1991

Analysis of multichannel oceanographic data set from San Francisco Bay by frequency, variance and time-domain techniques

Alan Murray Swithenbank
San Jose State University

Order Number 1345825

Analysis of a multichannel oceanographic data set from San Francisco Bay by frequency, variance and time-domain techniques

Swithenbank, Alan Murray, M.S.

San Jose State University, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

ANALYSIS OF A MULTICHANNEL OCEANOGRAPHIC DATA SET FROM SAN FRANCISCO BAY BY FREQUENCY, VARIANCE AND TIME-DOMAIN TECHNIQUES

A Thesis

Presented to

the Faculty of Moss Landing Marine Laboratories

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Marine Science

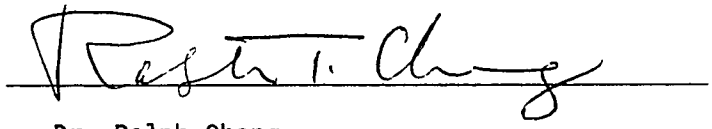
By

Alan Murray Swithenbank

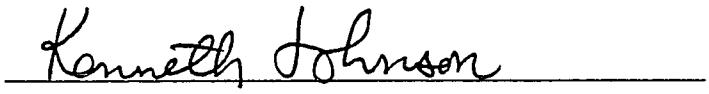
August, 1991

APPROVED FOR MOSS LANDING MARINE LABORATORIES

Dr. William Broenkow

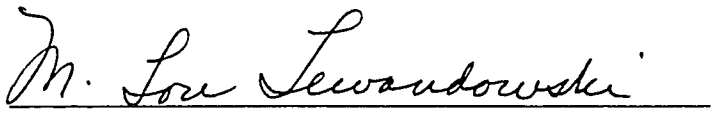


Dr. Ralph Cheng



Dr. Kenneth Johnson

APPROVED FOR THE UNIVERSITY



ABSTRACT

Fifteen-minute-interval tidal-excursion, temperature, and conductivity data, collected in San Francisco Bay between 23 February 1989 and 25 April 1989 from electronic sensors 1.2 m and 5.6 m below MLLW at Point San Pablo water quality monitoring station, were analyzed as multichannel data. Three time-series techniques were employed: spectral analysis, factor analysis, (as R-mode principle components analysis with rotation), and time-domain filtering. To obtain a 'measure of usefulness' for application in estuarine research each method's ability to separate tidal phenomena from superimposed non-tidal storm-signals was examined. Analysis techniques were also compared in terms of their computation and interpretation overhead. The results are presented as a data-analysis tutorial, including all preanalysis data preparation steps. From this study, if only one analysis technique could be applied to variables from a tidally-influenced data set, formation of power-spectra and periodograms via the Fast-Fourier-Transform would be most useful in determining the nature of underlying forcing functions. These analyses may be enhanced by phase-lag information available from cross-correlations of signals. Filtering, by time-domain techniques, or, (preferably), Fourier-Transform techniques, to remove tidal-components may also aid in identifying forcing functions. Factor analysis provides basically the same information as tidal filtering. Due to the amount of computation and interpretation overhead, versus the amount of information gained, factor analysis should be the last choice of a method

from the list of those examined here. That position may change when significantly longer records are examined. Along with investigating analysis techniques, physical processes in operation at the data-collection site were examined. That work involved comparing results of individual technique analyses to results obtained from combining techniques, and comparing analysis-results and record data to meterological and delta outflow data for the collection-period. The station record exhibited depression in salinity, particularly at low water, and long-period cycling in temperature. The long-period processes seemed to have greater effect on upper-sensor records than on lower-sensor records. Three major modes appeared to cause variation in data: tides, long-term forcing correlating with precipitation and delta-outflow, and long-term forcing correlating with variations in ambient air temperature.

ACKNOWLEDGEMENTS

Thanks to my original committee-members, all of whom I am pleased to be able to refer to as friend:

Dr. William Broenkow (who suffered the brunt of all this)

Dr. Ralph Cheng

Dr. David Peterson

Plus thanks to my 'eleventh-hour' member, who very graciously stepped in to eliminate a few bureaucratic complications:

Dr. Kenneth Johnson

And to the far-sighted and dedicated gentleman from DWR, often described as one of the 'good-guys':

Dr. Randall Brown

Along with a few friends, from or formerly of 'the survey,' for their much appreciated help, comments and unending moral-support:

Ms. Laura Benninger, Mr. Jon Burau, Ms. Anastasia Chapralis,

Mr. Tom Chase, Mrs. Jeanne DiLeo-Stevens, Mr. John Duff,

Mr. John Fitzpatric, Mr. Fred Murphy, Dr. Marlene Nobel,

Mr. Allan Ota, Dr. Robert Rosenbauer, Ms. Barbara Seekins,
Mr. Brian Swarthout

Add a couple more from DWR, dealing with whom also made all this much easier:

Mr. Heinrich Gebhard, Ms. Sheila Greene

Plus a long list of professors, teachers, T.A.'s, and in particular my friends from Humboldt State, who provide a most excellent learning environment; enabling me after a couple of years out of it, to come back and figure out that oceanography really is what I like to do:

Dr. Jeffrey Borgeld, Dr. George Crandell, Dr. James Gast,
Dr. Robert Hodgson, Ms. Debra Mondeel-Jarvis, Dr. John Pequegnat

Close to last, but certainly not least, much thanks and love to 'Auntie Kay' for allowing me to keep her garage full of myself and my stuff for the years I put into the project.

And, finally, acknowledgement to all those not named here, but without whose efforts things could well have turned out quite differently.

DISCLAIMER

Any manufacturers' names or product references contained in this thesis are for identification purposes only. They do not constitute endorsement by the author, the United States Geological Survey Water Resources Division, or the California State Department of Water Resources.

TABLE OF CONTENTS

	Page
CHAPTER 1. INTRODUCTION	1
CHAPTER 2. METHODS	5
Station Equipment, Location and Data Retrieval	5
System Calibrations	8
Introduction	8
Field Calibrations	8
Field Service Equipment Calibration	11
Sampling Scheme	11
Raw Data Corrections	13
Introduction	13
Manual Corrections	15
Drift Corrections	16
Fouling Corrections	18
Post-Correction Pre-Analysis Data Preparations	19
Introduction	19
Padding	20
Low-Pass Filtering	20
Unit Conversion	21
Mean removal	22
Detrending	22
Final Prepared Data Set	23
CHAPTER 3. ANALYSES	26
Introduction	26

Page

Spectral Analysis 27

Introduction 27

Variance, Correlation and Autocorrelation 29

Cross-correlation 35

Power-Spectrum Analysis 45

Fourier-Transform Filter 53

Summary 56

Factor Analysis 59

Introduction 59

Principle Components Analysis 61

Rotation 64

Analysis Results 65

Time-Domain Filtering 73

Introduction 73

Time-Domain Tide-Removal Filters 74

Analysis Results 77

CHAPTER 4. PROCESS INVESTIGATION 79

Introduction 79

Preliminary-Analysis Observations 80

Introduction 80

Spectral Analysis 80

Factor Analysis 83

Time-Domain Filtering 84

Adjunct Data 85

Page

Introduction 85

Adjunct-Data Analyses and Comparisons 90

Process Investigation Summary 103

CHAPTER 5. SUMMARY 106

Review of Techniques 106

Autocorrelation and Cross-correlation 106

Harmonic Analysis 107

Fourier-Transform Filtering 109

Factor Analysis 110

Time-Domain Filtering 112

Final Comments 113

REFERENCES 117

APPENDIX A. DATA COLLECTION AND CORRECTION A-1

Introduction A-1

Sampling Scheme A-1

Initial Inspection A-3

Manual Corrections A-5

Computer corrections A-10

Introduction A-10

Reference-Based Outlier Corrections A-10

Long-Term-Drift Corrections A-13

Statistical Outlier Removal A-20

Data Manipulations due to Constraints of Analysis Techniques .. A-21

Introduction A-21

Page

Establishing Record Length A-23

Padding A-24

Low-Pass Filtering and Decimation A-25

Scaling and Unit-Conversion A-28

Mean Removal and Detrending A-31

Final Prepared Data Set A-32

Final Comments A-32

APPENDIX B. SPECTRAL ANALYSIS B-1

Introduction B-1

Variance, Covariance, Autocovariance and Autocorrelation B-3

Introduction B-3

Variance and Covariance B-5

Autocovariance and Autocorrelation B-8

Cross-correlation B-19

Harmonic Analysis B-30

Introduction B-30

Fourier Series B-31

Continuous-Fourier-Transform B-44

Validity of the Periodogram Technique B-50

Windowing, Leakage and the Gibbs Phenomena B-51

Discrete-Fourier-Transform B-58

The Fast Fourier-Transform B-60

Digital Filters B-62

APPENDIX C. R-MODE FACTOR ANALYSIS C-1

Page

Introduction C-1

Mathematical Foundations of Factor Analysis C-6

Introduction C-6

The Structure of Linear Systems C-7

The Correlation Matrix C-11

Components Analysis C-16

Factor Rotation C-23

R-Mode Factor Analysis Example C-28

Introduction C-28

R-Mode Factor Analysis Using the Covariance Matrix C-34

initial inspection C-34

principle components analysis C-35

factor rotation C-41

factor interpretation C-45

R-Mode Factor Analysis Using the Correlation Matrix C-51

introduction C-51

initial inspection C-52

initial-factoring and rotation C-53

Factor Interpretation C-59

APPENDIX D. TIME DOMAIN FILTERING D-1

Introduction D-1

Low-Pass Filtering D-3

Introduction D-3

The Nyquist-Criterion and Aliasing D-3

Page

Recursive and Nonrecursive Filters D-14

Phase Distortion and Roll Off D-16

Low-Pass Filters D-20

Introduction D-20

The Z-Transform D-22

convolution-summation property of the z-transform D-24

transform pairs D-25

Transfer Functions D-27

Response Decomposition and Eigenfunctions D-28

Eigenvectors and Eigenvalues D-33

Magnitude and Phase D-39

Bandwidth D-42

Transversal Filters D-43

Summary of Appendix-A Low-Pass Filter Properties D-46

The Godin Filter D-57

Introduction D-57

Digitization of Band-Limited Functions D-62

Truncation of Band-Limited Functions D-67

Time-Domain Filters D-70

APPENDIX E. MATHEMATICAL NOTATION AND DETAILS E-1

Introduction E-1

Variables, Mean, Variance, Covariance and Correlations E-2

Matrix Algebra E-6

Matrices and Vectors E-6

Page

Vector and Matrix Operations E-7

Vector and Matrix Addition E-8

Vector and Matrix Transpose E-10

Scaler Multiplication of Vectors and Matrices E-11

Vector Multiplication E-11

minor products E-11

major products E-12

vector product moments E-12

Vector Geometry E-13

Types of Matrices E-15

Matrix Multiplication E-16

Linear Systems E-18

Simultaneous Linear Equations E-18

Linear Algebra E-19

matrix inverse E-20

Vector Spaces and Subspaces E-22

Linear Independence E-23

Basis Vectors E-25

Matrix Rank E-26

Orthogonality and Least Squares Approximations E-28

Determinants E-32

Eigenvalues and Eigenvectors E-33

Statistics in Matrix Form E-38

Partial Fraction Expansion E-44

Page

APPENDIX F. PROGRAM LISTINGS F-1

ADDTEXT.BAS F-2

AUTOCOVCOR F-6

CONSAL.F F-9

CROSSCOR F-14

FACAN.M F-23

FFTSPEC.M F-27

FOURIER.FILTER F-29

GODIN.FOR F-55

HARMONICS F-65

PADGAPS.BAS F-76

PADGAP11.BAS F-78

PADGAP12.BAS F-80

PADGAP2.BAS F-83

PRNCMP.M F-85

VARIMAX.BAS F-89

9PTFLTR.BAS F-94

LIST OF TABLES

Table		Page
1	Raw match-positions for the first major-peak in cross-correlograms produced from data for Upper-Temperature (Tu), Upper-Salinity (Su), Lower-Temperature (Tl), Lower-Salinity (Sl), and Tidal-Excursion (Z) at the Point San Pablo monitoring station. Match-position gives the number of hours by which a lagged-variable appears to lead a fixed-variable	42
2	Correlations of the Point San Pablo station data matrix variables with factors from R-mode principle components analysis	72
A-1	Interpolation corrections to the 5345-point raw series. Record numbers refer to full 5828-point data-set span.	A-8
A-2	Manual-interpolations of 13 short gaps in the raw-data time-series. Record numbers refer to the full 5828-point data-set span	A-9
A-3	Sample raw-data output from the Point San Pablo water-quality monitoring-station.	A-12
A-4	System calibration-and-correction data from the Point San Pablo water-quality-monitoring station for the period 23 February 1990 to 25 April 1990. There were four raw-data record subperiods. The main record periods are subdivided by remote access data download breaks, but the same prorating is applied to each subperiod within a main period. Reference salinities are converted to specific-conductance ($\mu S\ cm^{-1}$) referenced to 25° C for use in the correction-algorithms.	A-15
A-5	Location of data-gaps remaining after manual and computer corrections to the raw data-set from the Point San Pablo station for the period 23 February 1990 to 25 April 1990. Record numbers refer to the full 5828-point data-set.	A-25

Table | Page

A-6 Maximum, minimum and standard-deviation values for the Point San Pablo station data-set after manual and computer-corrections, and low-pass filtering and decimation to 1024-point records. The values for the conductivity-columns after conversion to salinity are included. A-30

B-1 Cross-correlation values at lags of maximum and minimum correspondence for 10 cycles of a 19-point per-cycle sinusoid against itself, and for a copy of the sinusoid with random-noise injected against the clean sinusoid .. B-24

B-2 Data for the 30 largest harmonically-related sinusoids from a 1024-point (1023-hour) tidal-excursion record. The table-data were generated using the program HARMONICS (Appendix F). The tidal-data used in producing the table are the corrected values from this thesis. The table-data are in descending power-spectral-estimate (power) order. B-39

B-3 Comparison of accepted astronomical partial-tidal-constituents (Cheng and Gartner 1984) with data for harmonics derived using the program HARMONICS (Appendix F) with the corrected tidal-excursion data of this thesis. B-43

C-1 Calculated primary-constituents for example problem. C-30

C-2 Data-matrix for R-Mode factor analysis example problem. ... C-31

D-1 Common sampled-functions, and their discrete-sample equivalents with z-transform and radius of convergence . D-26

D-2 Gain-factor for 9-pt. Boxcar and 9-pt. QUadratic low-pass filters with sampling period T = 0.25 h. D-49

Table | Page

D-3 First 16 ranges for frequency-components that may alias one-hour interval data from 0.5 to 0.6 cycles/hour when these data are derived by using a 9-point, quadratic-weight, noniterative, transversal filter stepped every four points through 15-minute interval data. These ranges were generated using the folding-sequence: w, $\pi/T-1$, $w+\pi/T$, $2\pi/T-w$, $w+2\pi/T$, ... on the endpoints of the range $3.14 \leq w \leq 3.77$ radians/hour, (which corresponds to $0.5 \leq f \leq 0.6$ cycles/hour), with $T = 1$ hour. The ranges were sorted in ascending order. D-56

LIST OF FIGURES

Figure		Page
1	Data collection network station location chart. Inset: Point San Pablo (PSP) station area detail -- station is loacted on the end of terminal number 4 pier adjacent to the navigation light marked "4".	2
2	Sensor elevation diagram for the Point San Pablo Water quality monitoring station	6
3	Raw data records from the Point San Pablo water quality monitoring station for the period 23 February 1989 at 1715 (ST) to 25 April 1989 at 1000 (ST).	14
4	Prepared data records from the Point San Pablo water quality monitoring station for the period 06 March 1989 a 0315 (ST) to 17 April 1989 at 1815 (ST).	24
5	Correlograms for lags 0 to 256 hours of all variables from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).	34
6	Cross-correlograms for match-positions 0 to 60 hours of all variables from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). Autocorrelations (diagonals) intentionally left blank.	39
7	Position diagrams for relative maxima and minima from the Cross-correlograms of Figure 6.	41

Figure Page

8 Signal phase relationships diagram derived using lag 0 to lag 60 (hours) data from Cross-correlations for all variables at the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). These data exhibit a 25 hour cycle with a 1 hour lead of tide (Z) over upper and lower salinity (Su,Sl), (the salinities appear in phase), a 7 hour lead of upper temperature (Tu) over the salinity signals, and a 1 hour lead of lower temperature (Tl) over upper temperature. 44

9 Significance-plots for match-positions 0 to 60 hours for all variables from the Point San Pablo water quality monitioring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). 46

10 Periodograms and power-spectra for all variables from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). .. 51

11 Fourier filtered data records from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). 57

12 R-Mode principle component scores from Point San Pablo water quality monitoring station data for the period 06 March 1989 at 0315 to 17 April 1989 (ST). 67

13 Rotated R-Mode factor scores from Point San Pablo water quality monitoring station data for the period 06 March 1989 at 0315 to 17 March 1989 at 1815 (ST). 68

14 Moving-average diurnal filter applied to a data set with four samples per day. 76

15 Godin filtered data records from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). 78

Figure		Page
16	Sacramento-San Joaquin river delta outflow for water year 1989.	87
17	Location chart for the Point San Pablo water quality monitoring station and NOAA weather stations at Richmond and the Martinez Water-Plant.	88
18	Plots of NOAA daily weather data from the Richmond and Martinez Water-Plant stations for the period February 1989 through April 1989.	89
19	Scaled Fourier-Transform filtered data from the Point San Pablo water quality monitoring station, with interpolated and scaled Sacramento-San Joaquin Delta-outflow overlay. 30 points have been removed from each end of the filtered records to eliminate the filter taper.	91
20	One hour interval weather data for the period 01 March 1989 at 0315 to 102 April 1989 at 1815 (ST). These data were interpolated from daily NOAA weather station data values.	93
21	Periodograms of Fourier-filtered Point San Pablo water quality monitoring station data spanning 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).	95
22	Periodograms for interpolated Delta-outflow and interpolated NOAA weather-station data.	96
23	Overlays of scaled Fourier-Transform filtered salinity and temperature data from Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).	98

Figure Page

24 Overlays of scaled interpolated NOAA weather-station data from Richmond Station and Martinez Water Plant with scaled interpolated Sacramento-San Joaquin River Delta outflow data. Outflow data span 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). There is a 5 day lead of weather station data over outflow data, hence the time axes are identified by record-number rather than day-of-year. .. 99

A-1 Raw data records from the Point San Pablo water quality monitoring station for the period 23 February 1989 at 1715 (ST) to 25 April 1989 at 1000 (ST). A-4

A-2 Raw data records after manual editing process. A-7

A-3 Geometric diagram of data fixed-offset and prorated-drift corrections scheme. A-19

A-4 Data records after manual editing and computer correction processe. A-22

A-5 Data gap interpolation process diagram for a data-gap less than 24-hours wide. This extends to larger gaps as a weighted interpolation process. A-26

A-6 Corrected data records after padding and decimation. A-29

A-7 Prepared data records from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 (ST) to 17 April 1989 at 1815 (ST). A-33

B-1 Change in alignment of peaks and troughs due to stepping a copy of a signal at equal-interval time lags, τ, relative to the original signal. B-9

B-2 Autocovariance-function, (lags 0 to 256 hours), for tidal excursion data from the Point San Pablo water quality monitoring station, 06 March 1989 at 0315 (ST) to 17 April 1989 at 1815 (ST). B-11

Figure		Page
B-3	Constituents from the factor analysis example problem (Appendix C), a trend line, and their correlograms. ...	B-14
B-4	Sums of constituent 1 with constituent 2 and with trend line, and sum correlograms.	B-16
B-5	Sums of constituents 1,2 and trend-line wiht constituent 3 (noise), and and sum correlograms.	B-18
B-6	10 cycles of a 19-point/cycle sinusoid, cross-correlogram of the sinusoid with itself, and significance-plot. ...	B-22
B-7	10 cycles of a 19-point/cycle sinusoid with injected random noise,cross-correlogram of the clean sinusoid versus the noisy sinusoid, and significance-plot.	B-23
B-8	Cross-correlogram for Su versus Tu, significance-plot for Su versus Tu for lags to n/4 = 256 hours, first 256 hours of the tidal-excursion record.	B-25
B-9	First 60 hours of tidal-excursion, Su, and Tu.	B-26
B-10	Cross-correlogram and significance plot for Su versus Tu..	B-27
B-11	Overlay of first 60 hours of Tu, Su and tidal-excursion. .	B-29
B-12	Diagram of effect on a sinusoid of change in phase angle, $2\pi p$, $0 \leq p \leq 1$.	B-34
B-13	Power-spectrum and periodogram for first 150 harmonics from the 1024-point (1023-hour) tidal-excursion record.	B-40
B-14	Power-spectrum and periodogram for harmonics 25 to 100 from the 1024-point (1023-hour) tidal-excursion record.	B-42

Figure — Page

B-15 Extension of single-pulse record to an infinite record with period T. B-48

B-16 Approximation of the sum of the C_k Fourier-coefficients. . B-49

B-17 Example periodic waveform, the squarewave. B-54

B-18 The Dirichlet Kernel. B-56

B-19 Squarewave exhibiting Gibbs phenomena ringing as the Dirichlet Kernel. B-57

B-20 Overlay unfiltered and Fourier-Transform filtered tidal-excursion record from the Point San Pablo water quality monitoring station, 06 March 1989 at 0315 (ST) to 17 April 1989 at 1815 (ST). B-65

B-21 Low-pass transfer function for 30 hour stop-period 40 hour pass-period FFT based tide-removal filter. B-66

C-1 A 3x7 data-matrix and the graphical expression of it's column-vectors in 3-dimensional space. Note that the data were plotted using a left-handed coordinate-system, as is typically done for studies in physical-oceanography. C-14

C-2 Representation of a two-factor model. C-18

C-3 Two-factor model for three variables. C-19

C-4 Example of unrotated and rotated factor-axes for the 2 factor, 3 variable case. C-27

C-5 41-point primary-constituent waveforms constructed for the R-Mode factor-analysis example-problem. C-29

Figure Page

C-6 Plotted column-vectors of the data-matrix derived from the 41-point primary-constituent waveforms for the R-mode factor-analysis example-problem. C-32

C-7 Unrotated-scores from the initial-factorization of the R-Mode factor analysis example problem covariance-matrix. C-48

C-8 Rotated-scores from the R-mode factor analysis example problem covariance-matrix. C-49

C-9 Unrotated-scores from the initial-factorization of the R-mode factor analysis example problem correlation-matrix. ... C-60

C-10 Rotated-scores from the R-mode factor analysis example problem correlation-matrix. C-61

D-1 First seven pleats of the Nyquist Folding Diagram. D-6

D-2 Errors introduced by sampling a waveform at less than or equal to the Nyquist frequency. D-8

D-3 Graphical low-pass filtering of the sum of two sinusoids . D-10

D-4 Typical mixed-tide record for the California Coast showing semidiurnal, diurnal, and spring-neap cycles. D-12

D-5 Filter-introduced phase-distortion. D-17

D-6 Ideal low-pass filter representation in the time and frequency domains. D-19

D-7 Non-ideal low-pass filter representation in the time and frequency domains. D-21

Figure Page

D-8 Gain-factor plot for a 9-point, least-squares fit, quadratic-weight, low-pass filter with period T = 0.25 h. D-47

D-9 Gain-factor plot for a 9-point, boxcar low-pass filter with period T = 0.25 h. D-48

D-10 Gain-factor plot for a 9-point, least-squares fit, quadratic-weight, low-pass filter with period T=0.25 hours. Here the plot is made out to 3 times the Nyquist frequency to show periodicity of the gain factor function. D-53

D-11 The diffraction function. D-60

D-12 Periodic repetition (ghosts) for the Fourier-series of a function which is non-zero over only a finite interval. D-64

D-13 Line spectrum, and spectrum of a truncated band-limited function. D-68

D-14 The spectrum of A_n, operating as a low-pass filter as n approaches infinity. D-73

D-15 The spectrum of the 'Godin Filter' operator. D-74

E-1 Vector projections E-30

E-2 Geometric interpretation of eigenvectors and eigenvalues . E-37

CHAPTER 1

INTRODUCTION

As part of continuing research on the San Francisco Bay estuary conducted by the United States Geological Survey, Water Resources Division (WRD), a network of continuous-monitoring water-quality stations, extending from south bay to north bay (Figure 1), is maintained in cooperation with the California State Department of Water Resources (DWR). Readings of temperature, specific conductance and, (at some stations), tidal excursion are taken automatically at each site from electronic sensors operated by a microcomputer controlled, multichannel data acquisition system developed at WRD for use in estuarine research (Swithenbank 1990). Sampled data are archived as part of a developing data base to provide baseline information for future bay studies. They are also analyzed as time-series in the frequency and time domains in order to look at long-term trends and short-term phenomena in physical properties of the bay. To determine a 'measure of usefulness' for numerical and statistical techniques that may be applied in bay studies, some research is directed towards examining analysis procedures used to manipulate long-term, time-series information. That work forms the basis of this thesis. As such, the main thrust here is not to analyze collected data, but to present, in a tutorial fashion, the analysis techniques themselves.

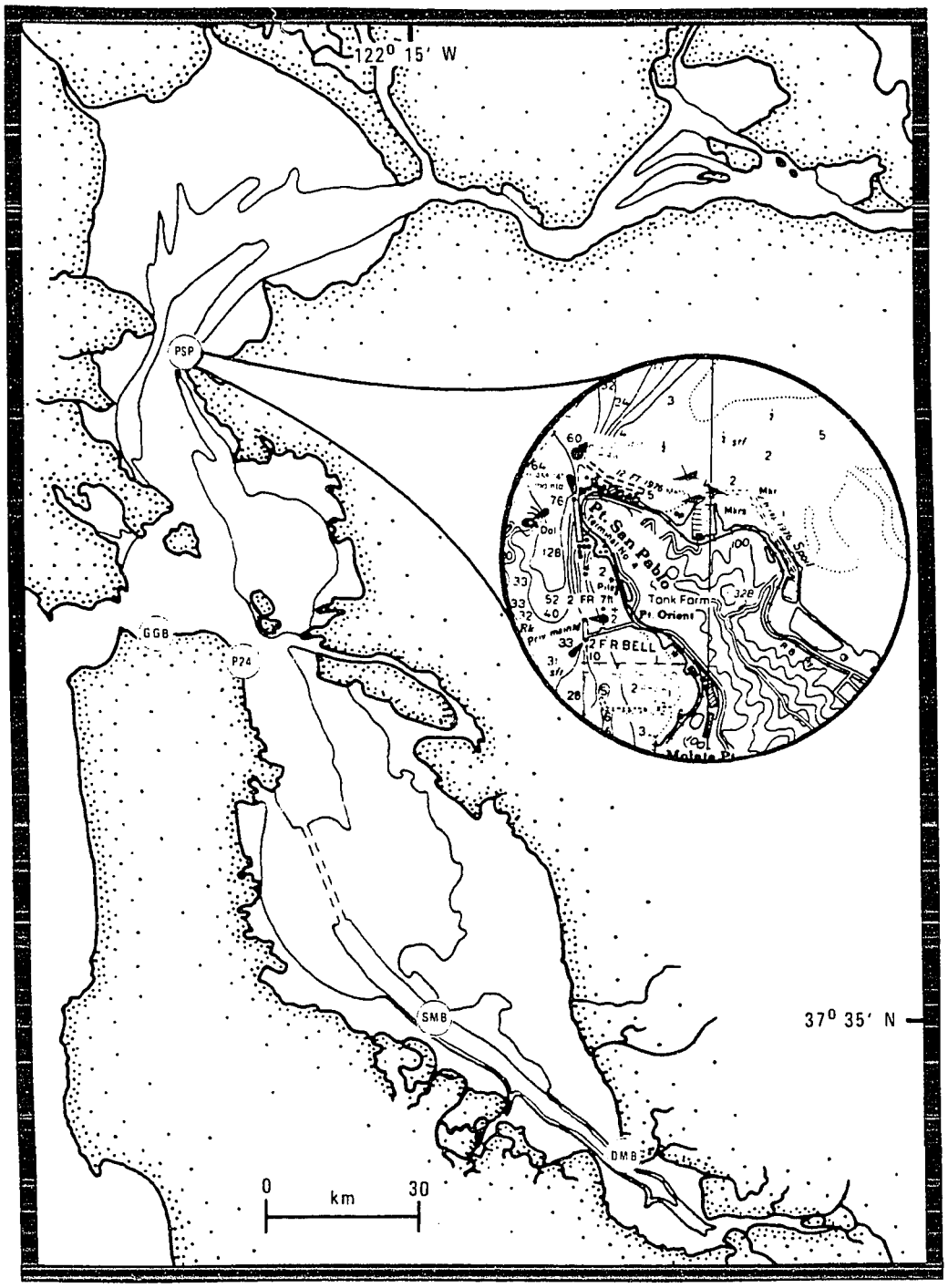


FIGURE 1: Data collection station location chart. Inset: Point San Pablo (PSP) station area detail -- station is located on the end of terminal number 4 pier adjacent to the navigation light marked "4".

Data discussed in this report were collected between 23 February 1989 and 25 April 1989, at the Point San Pablo monitoring station, near Richmond California (inset Figure 1). The collection period included a moderate storm event, resulting in increased freshwater flow past the sampling-station. During periods of low freshwater flow, tides are the main factor influencing physical-data records taken from the San Francisco Bay estuary. During high-inflow periods, records may be affected by freshwater input, primarily from the Sacramento-San Joaquin River delta. Of all sites in the network, the Point San Pablo station is closest to this added input source, and hence most influenced by it. Since tidal and non-tidal phenomena could be readily observed within data collected at the Point San Pablo station during the indicated period these data were chosen for analysis.

Three time-series analysis techniques were examined: spectral analysis, factor analysis (as R-mode principle components analysis with rotation), and time-domain filtering. The advantages and disadvantages of each method were determined by using them individually on the same data-set. Along with the examination of analysis techniques, a study of physical process in operation at the collection site is included in this thesis. That investigation involved comparing results of the individual technique analyses to results from combining techniques, and comparing analysis results and station records to meterological and delta outflow records for the collection period.

This thesis is divided into five chapters. Chapter 1, (INTRODUCTION), which concludes with this paragraph, gives an overview of the presented study. Chapter 2, (METHODS), describes the station location, station equipment, data-collection procedures, system calibrations, sampling scheme, data corrections, and preanalysis data preparations. The magnitude of applied data corrections may be seen in Appendix A. Chapter 3, (ANALYSES), covers the analysis techniques, and their application to the corrected data-set. The examination of physical process in operation at the sampling station during the collection-period is presented in Chapter 4, (PROCESS INVESTIGATION). The last section, Chapter 5, (SUMMARY), is an overall set of conclusions based on the various analyses and examinations contained in this thesis. There is at least one appendix associated with each analysis-technique. These appendices, and others, give greater detail on the data-manipulations and analysis techniques than is provided in the main text.

CHAPTER 2

METHODS

Station Location, Equipment and Data Retrieval

The Point San Pablo water-quality monitoring station is located on the PakTank Corporation pier (inset Figure 1) on Western Drive, near the Standard Oil refinery at Richmond California (37° 58' N, 122° 26' W). Here specific-conductance[1] and temperature are monitored at two depths by pairs of fixed-altitude sensors, along with tidal excursion relative to the upper sensor-pair (Figure 2).

Temperature at the Point San Pablo station is measured using YSI 44018 thermistor-pairs in a linearizing network, specific conductance referenced to 25° C is determined with Foxboro Model 1210 EC meters,

---

[1] The prefix 'specific' applied to any parameter means that values are normalized to some standard unit, e.g., unit-volume, unit-area, etc. The unit of conductance used in this thesis is the microsiemen (µS). The siemen is a measure of conductance in inverse-ohms (mhos). Standardizing conductance to unit-volume (1 cubic-cm) yields specific-conductance as siemens/cm, or, in our case, µS/cm. Conductivity measurements are dependent on geometry of the measurement-volume, however, readings converted to specific-conductance are always comparable regardless of cell-geometries, so long as temperature and pressure affects taken into account. Here conductivities are converted to specific-conductance in µS/cm and referenced to 25° C by circuits within the equipment.

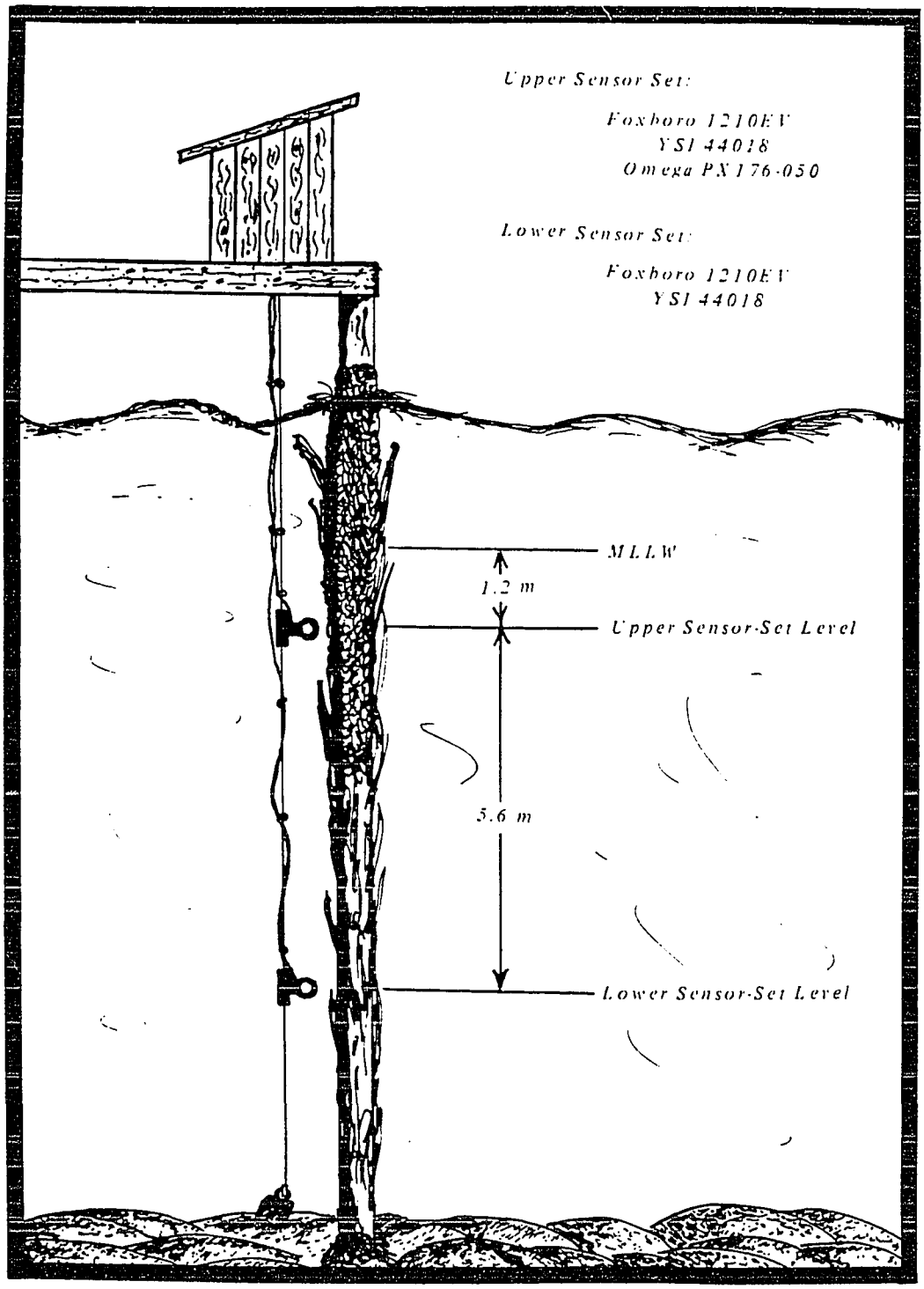


FIGURE 2: Sensor elevation diagram for the Point San Pablo water quality monitoring station.

and tidal excursion is calculated from the output of an Omega PX176-050 pressure sensor. Outputs from the sensors are read every fifteen minutes as dc-voltages. Sensor output-voltages are scaled to fall within a 0 to +5 volt range. Full-scale range for temperature is -5° to +45° C. Full-scale range for specific conductance is 0 to 60000 microsiemens (μS) cm^{-1} referenced to 25° C. Full-scale range for tidal excursion is 0 to 35.14 m. Scaled voltages are put through a 12-bit analog-to-digital conversion, giving a resolution of 0.0015 volts, or 0.03% of the full-scale output for a sensor. Digitized voltage-readings are stored as 16-bit values within the data acquisition system for later analysis. On retrieval, the appropriate readings are converted into temperature (° C), specific conductance (μS/cm referenced to 25° C), and depth (m) values. Data are collected approximately every four days via a phone-modem link to the station.

System output is a nine-column record of raw fifteen minute interval data consisting of: day of year, time of reading, upper temperature reading, upper conductivity reading, lower temperature reading, lower conductivity reading, upper sensor depth, a temperature insensitive fixed-voltage reference reading, (nominally 3.468 volts), and a ground (zero-volt) level reference reading. One such record, spanning from 23 February 1989 (day 54) at 1715 PST to 25 April 1989 (day 115) at 1000 PST, contains the data examined here.

System Calibrations

Introduction

There are two levels of calibration associated with the WRD data acquisition system. First is the in-field calibrations performed on each system as part of its periodic maintenance schedule. Second is calibration of equipment used for the periodic field-calibrations.

Information gathered during field-calibration is used to make corrections to raw data. Calibrations of field-service equipment are used to adjust station field-calibration numbers before raw data corrections are performed. Using this two-step process, outputs are compensated for offsets that may exist for a monitored parameter at the time of its raw voltage reading.

Field Calibrations

At approximately two-week intervals, instruments at each station are physically inspected. The time between inspections constitutes a record period. At the end of each record, before any sensor fouling has been removed, each sensor-set is pulled up to just below the surface and allowed to equilibrate. After equilibration, simultaneous readings of temperature and conductivity are taken from the system and from an RS5-3 Beckman portable salinometer. These readings are recorded for the

two salinity and two temperature data-channels, along with time and date for each comparison. Time, date and readings for depth are also recorded with the upper sensor-set suspended at 1 m and 2 m below the surface, according to marks on the cable. After this, any fouling is removed from the sensors and a similar comparison-reading set is made and recorded for the clean sensors. Bottle samples for salinity, and bucket thermometer readings are taken at the same time as the comparison readings. Both the before and after cleaning comparison-sets are used in later raw-data corrections. The clean sensor comparisons, bottle sample analysis results and bucket temperatures are used to determine if any recalibration of the station electronics package or the RS5-3 unit is required. When the difference between a clean-sensor reading and its comparison value is greater than 3% of full-scale for that sensor, an in-field recalibration of sensor electronics is performed.

Temperature recalibration consists of obtaining the digitized system-voltage output for a clean thermistor-pair after it has been rinsed thoroughly in distilled water and immersed in a continuously stirred distilled water and ice slurry in an insulated container. The thermistor-pair is then placed in a stirred bucket of ambient temperature bay water, and a second reading of the digitized system-voltage output taken. The ice-point reading is assumed to be 0° C. The bucket-temperature is read with the temperature-circuit of the RS5-3 unit. These values are recorded and used to calculate a two-point linear voltage-to-temperature conversion equation for the thermistor-pair.

The pressure sensor is calibrated by reading its digitized voltage output with the sensor 1 m and 2 m below the surface. These two readings are used to calculate a two-point linear voltage-to-depth conversion equation for the pressure sensor.

When required, a conductivity-unit is first electronically recalibrated so, when read directly with a voltmeter, its analog-voltage output is zero when the sensor-head is out of the water and dry. Then it is adjusted so its analog-voltage-output, divided by the unit's full-scale voltage (10 V) and multiplied times the full-scale reading-value (60000 µS/cm at 25° C) matches a simultaneous reading of salinity from the Beckman RS5-3 salinometer converted to specific-conductance at 25° C, when the sensor-head is immersed. Once this is accomplished, a system-reading of the scaled, digitized conductivity-unit output-voltage is made taken with the sensor-head immersed. This digitized sensor output voltage, and a simultaneous Beckman salinometer reading converted to specific conductance at 25° C are used to calculate a two-point linear voltage-to-specific-conductance conversion equation, assuming that zero conductivity water gives a digitized reading of zero volts.

The station-computer operating system is designed to accept, as in-field programming, slope and intercept information for data channels and use them for subsequent data-output. Readings of the RS5-3 salinometer are corrected using data from its most recent calibration before they are used to make corrections to the system output-equations.

Field Service Equipment Calibration

A full calibration check of the RS5-3 salinometer is made every six months. RS5-3 unit readings are taken of a bay-water dilution-series over the approximate range S = 30.0 to S = 4.0 in steps of approximately 2.0 S-units. The RS5-3 salinity readings are compared to values obtained from bottle samples filled at the same time and analyzed using a Guildline Autosal model 8400A laboratory-salinometer with a previously analyzed bay-water substandard. A temperature-reading series is made in a Masterline 2095 temperature-bath over the approximate range 5° C to 35° C in approximately 5° C steps. RS5-3 temperature readings are compared to readings from an NBS traceable 46965 Trident laboratory mercury thermometer. Linear equations are fit to calibration-data via least-squares. Any corrections required for the RS5-3 salinometer readings based on these equations are made before the salinometer readings are used to correct digitized sensor-readings. Single-point spot checks of the RS5-3 are made approximately every 2 weeks using salinity bottle samples, and bucket temperature readings, taken in the field, while stations are being serviced.

Sampling Scheme

The sampling scheme for time-series information must be considered carefully in view of proposed data analysis techniques. The data of this report are samples equally-spaced in time. This simplifies spectral and

time-domain analysis. The fifteen-minute sample period is more than a factor-of-two shorter than the periods of tides, related cycles, and storm generated signals we are examining. Thus, the sample frequency is more than a factor-of-two greater than the highest expected frequency of interest[2] in the data set. This fills the Nyquist-criterion, which says that all signals of interest can be reproduced from these data. The length of the record is significantly greater than the periods of interest. The predominant tidal components are diurnal and semidiurnal. Storm related pulses within the data were approximately five days duration. The length of the corrected data set is approximately 42 days. This means aliasing, (introduction of false signal-frequencies), can be minimized in the analyses. Thus, the sampling-scheme used for the data of this thesis fits a reasonable set of criterion for use in time-series analysis. But, it in no way implies that data, as they come off the system, are ready for analysis. Data must first be corrected to compensate for inherent errors in the sampling-system, and further manipulated to fit any requirements of analysis techniques.

---

[2] Frequencies higher than those of interest (to us), e.g., seiches, may be found in the data-set. If these frequencies are higher than the Nyquist-frequency for the sampled data, and of sufficient energy, they will interfere with analyses for the frequencies of interest, e.g., tides and storms. Some of the data preconditioning steps to be discussed later reduce this possibility.

Raw Data Corrections

Introduction

The first step in any analysis is to look at the raw data. Regardless of the need for unbiased reports, investigators should have some idea of what is expected, and decide to continue with an analysis, or consider some other approach, based on the state of the raw information. The raw data of this report (Figure 3) appear to represent an acceptable set of time-series records for measurements in an embayment on the California coast. There is an apparent semidiurnal period in the records, and values for readings both of temperature and specific-conductance[3] fall within the expected range for such waters.

---

[3] The station-equipment described in this thesis directly outputs specific-conductance (µS/cm) referenced to 25° C. Since this conductivity is referenced to a single temperature, the form of a specific-conductance-plot conveys the same information as the form of a salinity-plot for the same waters. For a quick, preliminary check on salinity values, a heuristic conversion factor, (derived from the difference in corresponding specific conductance values for salinity in one-unit increments from S = 1 to S = 40, at 25° C), is 0.000675. Multiplying specific conductance values referenced to 25° C by this factor will give a rough conversion to PSS-78 values for the unitless practical salinity. As part of data-preconditioning, raw specific conductance values were converted to salinity using the complete PSS-78 equation set (Fofonoff and Millard 1983). This was done not because of any real need to have salinity values in the analyses, but as a matter of scaling to prevent analysis problems that could be generated by the large difference in magnitude of variances calculated for specific-conductance versus those calculated for temperature and tidal-excursion. Scaling problems are explained in the text and Appendix C.

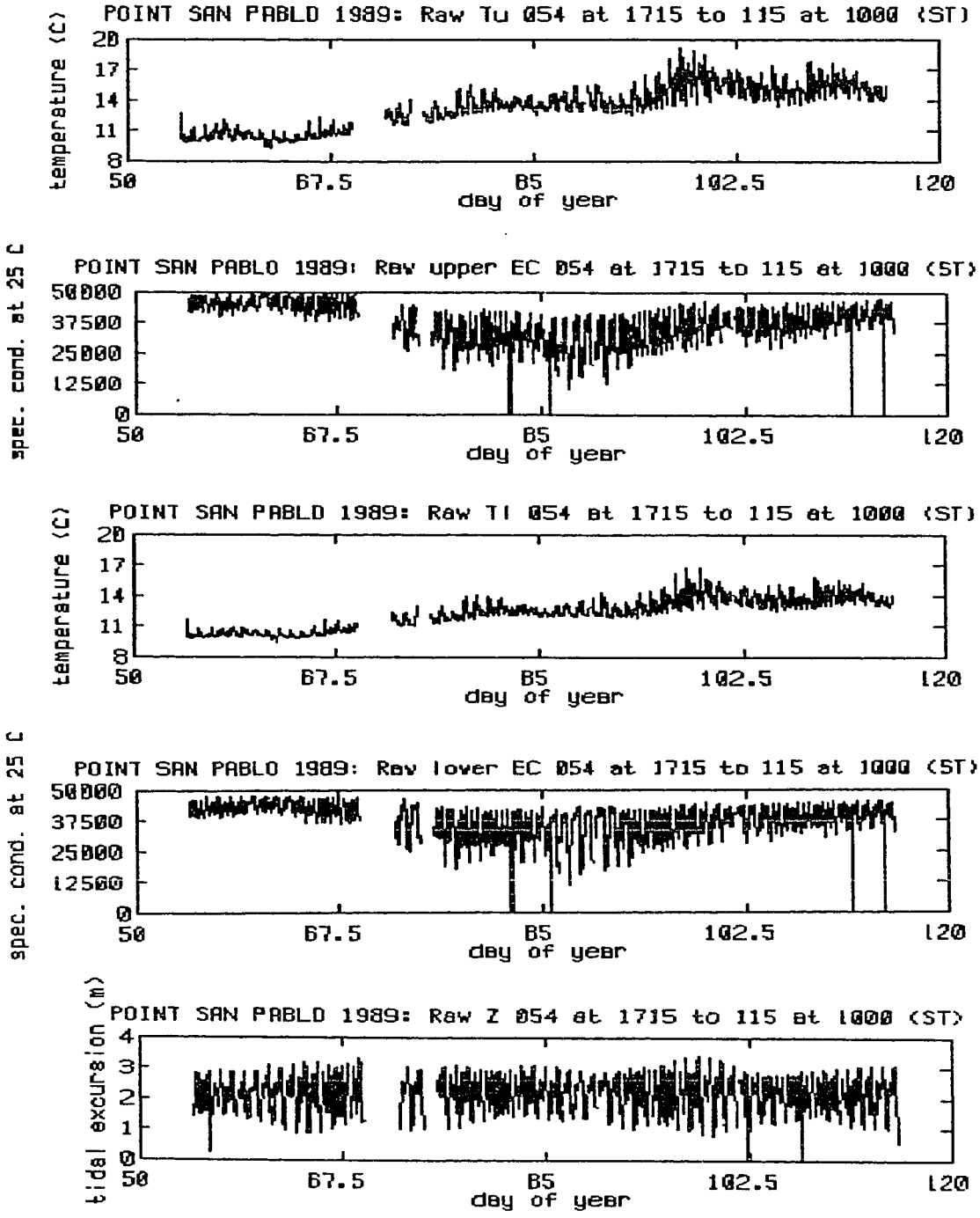


FIGURE 3: Raw data records from the Point San Pablo water quality monitoring station for the period 23 February 1989 at 1715 (ST) to 25 April 1989 at 1000 (ST).

Given this, it might seem reasonable to jump into an analysis of the records. However, that is not the case. Several correction-steps must be performed on data values before they can used. General information on corrections used for the data set of this thesis are given in the following three sections, in the order they were applied to the data. Detailed information on each step is given in Appendix A.

Manual Corrections

Corrections to raw-data made by running fixed-algorithms within programs on a computer are not perfect. This is seen most vividly when trying to remove outlier-values through an automated process. Cases exist where, though it is obvious to a person looking at a plot of the data, a value in a raw-data set will not be seen as an outlier by a correction program, even if it is a physical impossibility. The correction programs and data-set of this thesis are no exceptions.

With experience correcting data sets of a particular type using the same program, data-points that may be potential problems start becoming visually apparent in raw-data plots. The first step in data correction here was to look at plots of the raw data and manually interpolate through points that appeared as outliers to the eye which the correction routines might 'have trouble with,' and through several short gaps in the records, (a few minutes to a few hours), caused when the station was shut down for normal service.

Drift Corrections

Variation in data obtained from acquisition-equipment may arise from sources completely unrelated to natural variation in parameters being measured. These variations, blanketly referred to as 'drift,' must be removed as completely as possible before any analysis is performed. The following potential drift problems were addressed in correcting the data of this thesis: instantaneous drift (system noise), system (long-term) drift, and sensor response.

Instantaneous drift is the case where a momentary noise signal is superimposed on a sensor-reading, causing the value to shift outside the expected range of values in that portion of the record, i.e., producing an 'outlier.' There are many possible reasons for outlier values. From an electronic viewpoint, outliers may be caused by surges in power-supply or reference-voltage levels, (spikes), that temporarily affect the stability and accuracy of measurement circuits. Rapid temperature changes, as well as electromagnetic-radiation, may also induce outlier-values into records from an electronic system.

Two automated techniques for dealing with outliers, one heuristic and one statistical, were applied to the data. The heuristic method was used for making corrections that relate to electronic-spike errors. Based on the difference between the expected values for reference voltages, and values for those voltages stored with each data-record, offsets and

scaling factors are calculated to apply to the raw data values. The statistical technique was used to remove some outliers that the heuristic method can not detect. It is based on dividing the record into segments and linearly-interpolating through portions of a segment where raw-data values fall outside a range based on multiples of the standard-deviation for that segment of the record.

System, or long-term, drift is the tendency for analog circuits to slowly change the readings they give of a nonchanging input. The amount and rate of drift is a function of equipment type and quality. It essentially puts a change, prorated over time, into system readings away from true values of parameters being measured. Readings may drift away from and back towards true values. The correction technique applied here assumes system-drift, once begun, continues to move in the same direction as a linear function, in time. Each raw data-point was modified based on the value calculated at corresponding times for a line derived from the cleaned-sensor calibration readings at the beginning of the record, and the fouled-sensor calibration readings at the end of the record.

Sensor response refers to the speed with which a sensor can respond to changes in a measured-parameter. It may be, for example, that a temperature-sensor is not capable of tracking the rate of temperature-change to which it is exposed. This can induce offsets in a record that require a great deal of work to remove. The sensors used

here have responses on the order of a few seconds, and the parameters of interest change at rates measured in minutes, so correction for this type of error was ignored.

Fouling Corrections

Even if calibrations were perfect and all electronic errors in a data-acquisition system could be eliminated, at least one source of error in readings will not be eliminated from any piece of equipment that comes in physical contact with the marine environment, that is: sensor fouling. Sensor fouling may result from chemical reactions such as oxidation, but, the major fouling processes are biological. Simply, stuff grows anywhere stuff can find a place to grow, and a nice clean sensor is an ideal spot for an opportunistic organism to get in and really muck things up. This form of error would intuitively be logarithmic in time. However, the observations and calibrations required to make the appropriate correction-calculations are time consuming, difficult, and seldom made. In general, sensor fouling is treated as another linear-error in system output-values. And, as such, here, was lumped in with the long-term drift corrections.

A phenomena associated with fouling-drift is sensor-degradation. Over time many types of sensors used for physical and chemical measurement simply 'wear out,' causing sensor-readings to move away from true-readings. The nature of this type of drift is a function of

sensor type and environment, and can be quite complicated. From experience, if they fail, the types of sensors used for collecting the data of this thesis tend to do so catastrophically, rather than by degrading. So, no attempt was made to correct for this form of drift.

Post-Correction Pre-Analysis Data Preparations

Introduction

Data analysis requires not just corrected data, but corrected data of the proper form and format for the analysis technique. Data must be properly conditioned and scaled to fit a technique before any analysis can be performed. When, as here, more than one technique is to be applied to the same data set, that data set must be forced to fit the constraints of all techniques simultaneously or comparisons between the methods may not be valid.

The manipulations used to fit the data of this thesis to the analysis techniques were: padding, low-pass filtering, unit conversion, mean removal, and detrending. These techniques, briefly described in the paragraphs that follow, are used to help eliminate possible analysis problems such as interfering-frequencies, and those that arise due to noise, and channel-variance or channel-unit differences. The mechanics and further rational for applying these operations are presented in Appendix A. Programs to implement them are listed in Appendix F.

Padding

Many time-series analysis techniques, e.g., the Fast-Fourier-Transform (FFT), do not perform well or will not work at all if there are gaps in the data set or an inappropriate number of data-points. Padding a data set means simply to extend its length or fill in gaps with values that either represent expected values for the missing data, or at least will not interfere with calculations for a chosen analysis technique. Here, a series of linear-interpolation schemes was used to fill in data-gaps with representative values.

Low-Pass filtering

Low-pass filtering is used to remove undesirable high-frequency components, e.g., seiches, from a time-series data set. That is one reason for using this technique to precondition data. Another reason for using a low-pass filter at this point in data preparation for this thesis was to reduce the number of data-points to be analyzed. This reduction was required to overcome memory constraints of the statistical software package used to implement factor analysis programs written for this thesis. A low-pass filter in the form of a running, weighted-average was use on the data set to convert from 15-minute-interval data to hourly-interval data. This technique and the filter used to perform the operation are described in Appendix D.

Unit Conversion

The relative amplitude of signals in a data set can be important to many time-series analysis techniques, particularly those that rely on calculation and simultaneous comparison of variance for each signal. Such techniques, e.g., factor analysis, can give erroneous results for the case where relative changes in each signal are similar, but magnitudes of calculated signal variances are different because of different measurement units. Without scaling, signals with large magnitude variances dominate in these analyses. They appear as controlling factors even when their true contribution to overall variation in a data set may be small. When signal units are scaled to match, calculated variances are closer in magnitude, and a more true picture of the relative importance of each signal to the overall variation in a data set may be obtained. This phenomena is described in Appendix C.

Here, specific conductance is the offending variable, having units several orders of magnitude larger than temperature or tidal excursion. Frequently all that is required to bring signal variances in line is algebraic scaling, i.e., multiplying each signal value by some fixed factor. This technique can sometimes lead to confusion due to unspecified unit changes that result from the multiplications. Algebraic scaling would have reduced the magnitude of variance calculated for specific conductance data. It would also have left these signals with arbitrary units. Unit conversion from specific conductance to practical salinity was

performed, bringing the magnitude of specific conductance closer to the magnitude of temperature and tidal excursion. This was not a scaling operation. It was an application of the Practical Salinity Scale (IEEE 1980). Specific conductance data were processed through a multiple term, fractional-order polynomial equation (Fofonoff and Millard 1983).

Mean Removal

Means were removed from all signals in the data set. Removing means is primarily an aid to spectral analysis. The offset in a record exhibited as non-zero mean can be picked up as a very low-frequency, high-amplitude, nonexistent signal, by the numerical techniques used in spectral analysis. This phenomena is described in Appendix B.

Detrending

Detrending is the process of removing from a data-record, the slope (trend) calculated for the line from a least-squares linear-fit to the data. Detrending is used for reasons similar to those for removing the mean from each signal in the data set (see Appendix B). This aids spectral analysis. It may also influence factor analysis, as a trend can effect calculated variances. Although the trend in the data removed by this technique is real, it represents, in this case, an extremely low-frequency signal that could interfere with analysis for tide and storm signals, and was removed.

Final Prepared Data Set

The drift-corrected, fouling-corrected, padded, filtered, unit-converted, mean-removed, detrended data (Figure 4) are slightly different in form than the raw-data they were derived from (Figure 3). However, they now fit the constraints of the various analysis techniques, and no longer contain most of the information, (real or not), that might interfere with analysis for the frequencies of interest.

Final Comments on Data Corrections

The corrections just presented give a sequence of steps that, barring unusual problems, produce data suitable for analysis. Ideally, correcting then appending records should give a data set ready for analysis. However, raw data collected from the network frequently require 'special handling.' For example, interpolations used to pad data had to be 'custom fit' to record gaps. This meant writing four separate computer programs, only of use for these data. Correcting segmented data records requires careful attention to the alignment of the tail of one segment with the head of the segment that follows it. Here, records are approximately two weeks in length. If errors in correction cause small offsets between records, then the two week maintenance cycle may be interpreted as a spring-neap effect. Examination of tail alignments should be made on a segment by segment basis: looking at expanded plots of corresponding segment ends. If there are alignment errors, calibration

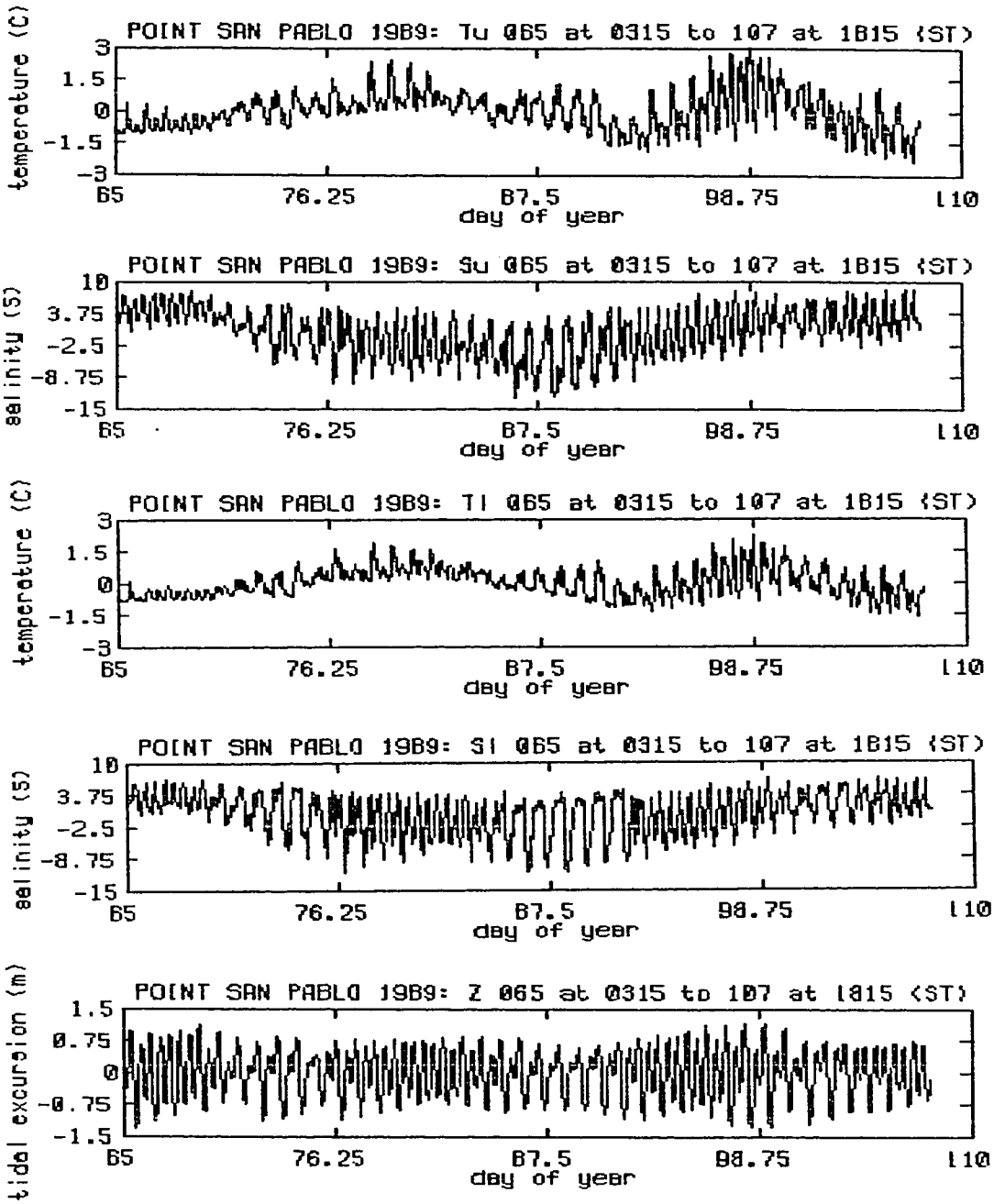


FIGURE 4: Prepared data records from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 (ST) to 17 April 1989 at 1815 (ST)

data and correction values must be examined. If no errors are found, then sensing equipment may be suspect. Failing to let equipment properly equilibrate before calibration readings are made will introduce cyclic offset errors in corrected data. If precision requirements allow, such errors may sometimes be compensated for by extrapolating the proper values from the closest segment of a record where the readings appear stable. This can be difficult to detect in tidally-influenced data. Tidal filtering of appended raw-data records and appended corrected-data records can help in identifying these segments. The temptation to 'fudge' should be resisted. Often data must be rejected, rather than corrected, because of this type of error. Data corrections can only compensate for 'reasonable' errors. The greater the attention paid to sampling and calibration procedures, the greater the probability of successful data correction. This cannot be over emphasized.

CHAPTER 3

ANALYSES

Introduction

This section covers the various time-series data-analysis techniques under discussion. It does not constitute an adequate analysis of the data from an oceanographic view-point. It is intended to show use and utility of the analysis techniques on an individual basis. However, showing use of the techniques on the data-set provides some insight to processes in operation at the monitoring-station at the time the data were collected. This may be considered 'preliminary work' as would be performed in the process of writing a research-paper, but not (normally) published in conjunction with that paper. The PROCESS INVESTIGATION chapter might be considered a 'report' based, in-part, on the preliminary investigations presented here.

The time-series analysis techniques examined here were: (1) spectral analysis, (2) factor analysis (as R-mode principle components analysis with rotation), and, (3) time-domain filtering. Each analysis, including results, has been given its own section in the text that follows. Separate appendices (noted within each section) provide greater detail for each time-series technique. Listing for programs developed to implement the analyses have been included in Appendix F.

Spectral Analysis

Introduction

We tend to think of waveforms more in terms of period (time) than frequency. However, any signal represented by a time-series record can be defined as a (possibly infinite) sum of sinusoids (sine and cosine 'waves') of different frequencies, amplitudes, and phases. The frequency-domain description of a signal, in terms of parameters of the sinusoidal-components in such a sum, is just as valid as a time-domain (point-by-point) description. In fact, its frequency-domain description often gives us significant clues to the underlying forces driving a signal; which could only be guessed at from the time-domain point-of-view.

The idea behind spectral (harmonic, Fourier, frequency) analysis is to take a function, $x(t)$, and express it as a linear combination of preselected functions, $y_i(t)$, such that:

$$x(t) = \Sigma X_i y_i(t), \qquad (1)$$

where X_i represent the 'importance' of components, $y_i(t)$, in forming the sum (1) for $x(t)$. The X_i may be considered 'weights' for the $y_i(t)$ components in the sum for $x(t)$. When values for the X_i are plotted as vertical-lines along a horizontal axis with-respect-to i, they display the spectrum of $x(t)$. The spectrum, X, through its spectral-values, X_i,

tells us the relative importance of the spectral-components, $y_i(t)$, in forming the signal, x(t), and, hence, something about the nature of x(t).

Formalization of frequency-domain descriptions for signals is credited to the French mathematician J.B.J. Fourier (1768-1830). The Fourier-Series is an infinite-sum of harmonically related sinusoids. A family of sinusoids is harmonically related if the frequencies of all its members are integer-multiples of a fundamental (lowest) frequency. The harmonically related sinusoids in a Fourier-Series expression represent the frequency-domain components of a signal, and the Fourier-Series sum returns a time-domain signal from the combination of frequency-domain elements. From the Fourier-Coefficients (weights) in the Fourier-Series we determine the spectrum of a signal. The magnitude of a spectral-component gives the 'energy contained' in the sinusoidal-component of the Fourier-Series sum it is related to. Energy, or 'power', and the statistical property variance are synonymous. Thus, investigation of a signal though spectral-analysis can be considered a comparison of relative levels of variance within its underlying sinusoidal-components.

Most spectral analysis techniques stem from the concept of variance. From variance is developed correlation, autocorrelation and cross-correlation. Often autocorrelations will give much the same information about underlying structure of signals as spectrums generated through Fourier techniques. Periodic-similarities between, as well as phase-differences for, multiple signals may be derived from cross-

correlations of those signals. These interrelationships may not be directly observable from individual spectums of signals without further comparison and analysis. So, prior to, or at least along with, generating spectrums of signals, one should consider producing their autocorrelations and cross-correlations. This is where we begin our investigation of spectral-analysis techniques. For greater detail on the procedures described above and below, see Appendix B.

Variance, Correlation and Autocorrelation

Any sampled time-series record, $\{x_i\}$, i=1,2,...n, is, by definition, a portion of the larger population of all possible readings from a signal. A sample-mean is given by:

$$\bar{x} = (\frac{1}{n})\sum_{i=1}^{n} x_i , \qquad (2)$$

The variance of a data-set is a measure of dispersion of sample-points, x_i, away from their mean, $\bar{x}$. The deviation of a single data-point, x_i, from the mean (2) of its sampled data-set is:

$$d_i = (x_i - \bar{x}). \qquad (3)$$

The unbiased-estimate of variance for a sampled data-set is given by:

$$s^2 = (\frac{1}{n-1})\sum_{i=1}^{n} d_i^2 , \qquad (4)$$

which is the 'average' value for the square-of-deviations of a set of data-points from their mean. The deviation-values must be squared for the calculation, as the direct sum of all deviations is zero. The square-root of a variables's variance (4) is its standard-deviation, s.

Covariance is a measure of the joint-variation of two variables about their common mean. That is, it is a measure of the tendency for corresponding values in two variables to 'track' each other. If a common mean is calculated as the mean of all values from two variables, and values for one variable always tend to plot on the same side of this common mean as corresponding values from the other variable, then the two variables covary, and will have a high covariance. Covariance of two variables, X_j, X_k, is calculated as:

$$cov_{jk} = \frac{n\sum_{i=1}^{n} X_{ij}X_{ik} - \sum_{i=1}^{n} X_{ij} \sum_{i=1}^{n} X_{ik}}{n(n-1)}, \qquad (5)$$

For two variables with similar-magnitude units, a large covariance-value means they are linearly-dependent. A low covariance implies that variables are independent. Covariance-values are scale-dependent. If one of the variables is measured in units which have a much greater magnitude than the other variable, then even if the variables are poorly related their calculated covariance may be numerically 'large.' To avoid problems with scale-dependence between two variables, we use the

correlation-coefficient, r, which is covariance divided by the product of the standard-deviations of the two variables:

$$r_{jk} = \frac{cov_{jk}}{s_j s_k}. \qquad (6)$$

A correlation of +1 indicates a perfect correspondence between variables. A correlation of -1 indicates one variable changes inversely with respect to the other variable. A correlation of 0 indicates there is no linear relationship between the variables what-so-ever. We will use the correlation-counterpart to any covariance-related calculations in the remainder of this thesis.

The prefix 'auto' is a combining form meaning 'by itself'. So, the term 'autocorrelation' means 'correlation by itself', that is the correlation of a variable with itself. The direct calculation of the correlation of a variable with itself will always give a value of r = 1. In order to make this calculation useful, we introduce lags. A lag is an offset between a variable and the copy of that variable being used in the autocorrelation calculation. Since we are dealing with time-series, a lag is an offset in time, τ, which is always selected to be some multiple of the equal-interval time-step used for sampling during data collection. The autocorrelation process is to calculate a series of correlations of the original variable with a copy of that variable for increasing lag-values. Autocorrelation is calculated as:

$$r_\tau = \frac{cov_\tau}{var(X)} = \frac{\sum_{t=1+\tau}^{n} X_t X_{t-\tau} - (\frac{1}{n})\sum_{t=1}^{n} X_t \; (\frac{1}{n-\tau})\sum_{t=1+\tau}^{n} X_{t-\tau}}{\sum_{t=1}^{n} (X_t - (\frac{1}{n})\sum X_t)^2}. \qquad (7)$$

A plot of lag-value versus autocorrelation is a correlogram. There are (at least) two ways to use correlograms. They may be compared to the correlograms of idealized-model signals in order to determine the underlying structure of the original signal. (The representation of a complex time-series can be built up as a linear combination of simple models.) And they may be used to determine periodicities within the original signal. We will consider the later application.

At lag $\tau = 0$, there is a perfect correspondence between the original and lagged copy of a signal, and hence an autocorrelation value of 1. As τ increases the alignment of peaks and troughs in the inputs changes and the autocorrelation value increases and decreases accordingly. If there is a 'good' correspondence between peaks in the original signal and a lagged-copy, then there is a period underlying the signal with a value near the lag-value, τ. There may also be lag-positions for 'good' correspondences between peaks and troughs in the original and lagged copies of a signal. These positions will generate large negative autocorrelation values. When a signal has a strong periodic component then its correlogram will show high indications of periodicity at each lag-value, τ, that is a multiple of this principle underlying period. Tide

records are one of many signals that exhibit this property. A recorded parameter which is driven by a semidiurnal tide is strongly periodic near 12-hours-and-25-minutes and 24-hours-and-fifty-minutes. A correlogram for that parameter will exhibit high autocorrelation values at lag values which are multiples of these periods, throughout the entire length of the correlogram. We know that strong indications of periodicity beyond the near 14-day spring-neap cycle should not be tidally-related, but any true long-period information is likely masked by the repeating shorter-period cycles. So, we cannot use autocorrelation to specifically identify long-period non-tidal information in the prepared data of this thesis (Figure 4) But, we can use it to examine the relative significance of the expected semidiurnal components within these data.

Autocorrelations are, in general, not reliable for long lag values, regardless of the strength of periodicities in a signal. So, correlograms are usually produced for lags from $\tau = 0$ to no greater than $\tau = \delta n/4$, where δ is the record time-step, and n is the number of points in the record. For the data of this thesis $\delta n/4 = 256$ hours. Correlograms out to lag-value 256-hours (Figure 5), produced from the output of the program AUTOCOVCOR (Appendix F), show high autocorrelation-values at the expected major semidiurnal component periods, and their multiples. The multiple-period phenomena is, not surprisingly, seen most strongly in the correlogram for the tidal-signal itself. This same periodicity is seen for the length of the temperature and salinity correlograms.

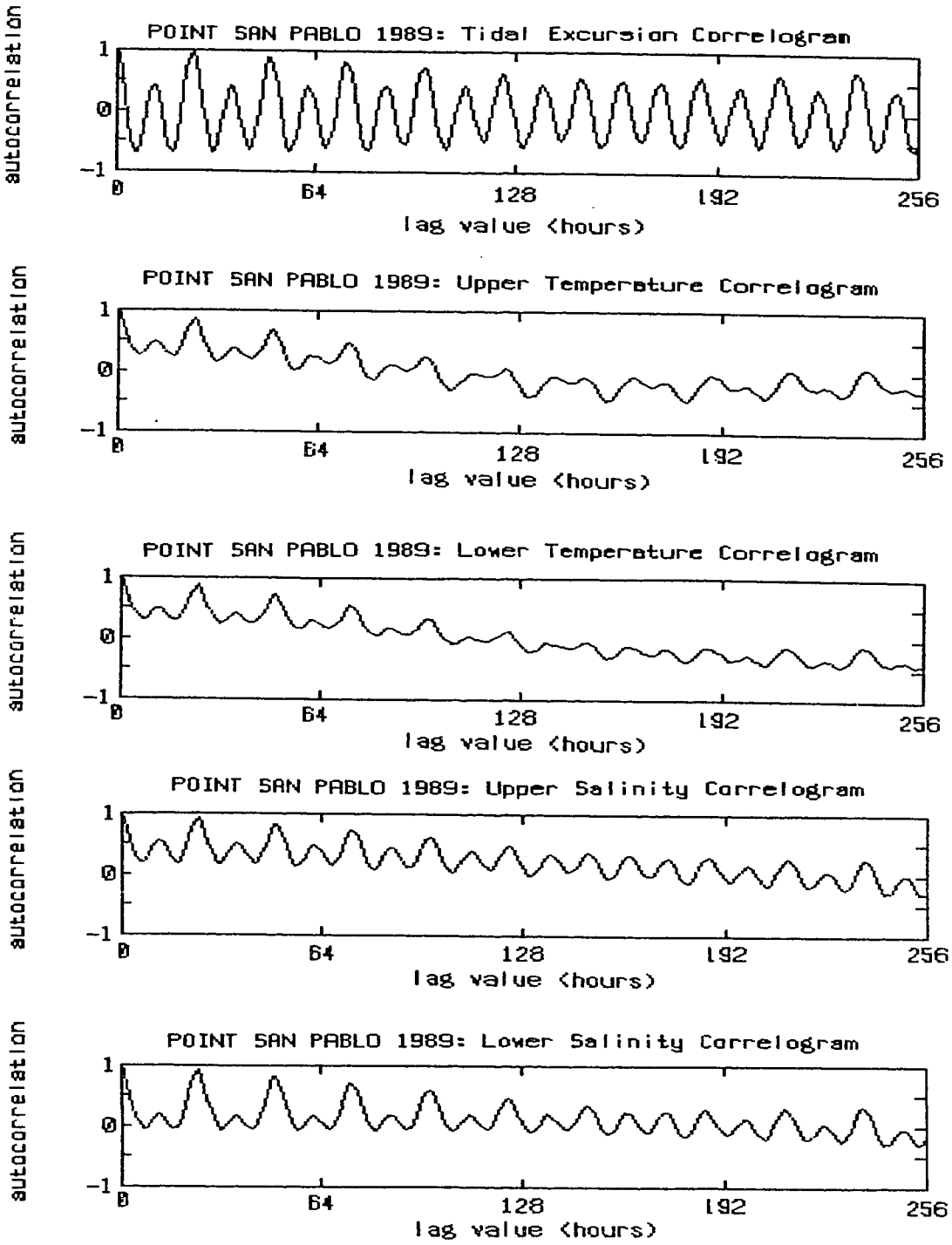


FIGURE 5: Correlograms for lags 0 to 256 hours of all variables from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).

There are notable differences between the tide correlogram and those for temperature and salinity (Figure 5). The salinity correlograms more closely resemble the tide correlogram than do the temperature correlograms. Apparently the salinity and temperature signals were not solely tidally driven, and the non-tidal forcing more greatly affected temperature than salinity during the sampling period. The effects seen in the temperature and salinity correlograms are the affects of long-period forcing on these records. This forcing is observable in the prepared data plots. These (Figure 4) show a stronger variation away from the form of the tide-plot for temperature than for salinity. Without adjunct information we can make no hypotheses as to the nature of this forcing. Our initial-investigation via autocorrelations serves mainly to verify what is observed in the data plots. The tide record is indeed a clean tidal signal. The temperature and salinity records show a strong tidal influence but are not solely tidally-driven. And, whatever forcing-functions beyond tides there are, they seem to have more strongly influenced temperature than salinity during the sampling-period.

Cross-correlation

Cross-correlation is similar to autocorrelation except comparisons are made between one signal and lagged-copies of another signal. Thus we can examine two different signals for periodic-similarities. We use the statement: 'variable-X versus variable-Y', with regard to cross-correlation, to mean that X was the fixed-variable, and variable-Y had

lag-shifts applied during the calculations. There are some difficulties with cross-correlations. Often it is not possible to determine a zero-lag position, because either series may lead the other. Since the two series are not identical the resulting cross-correlogram is not symmetric, but depends upon to which series the lag shifts are applied. Also, the two series may not be the same length.

If, for a variable of size n, we use the notation:

$$\Sigma X_1 = \sum_{i=1}^{n} X_{1i}, \tag{8}$$

and the notation n^* to mean the number of overlapped positions between two series, X_1, X_2, then the cross-correlation for lag m between the two series is:

$$r_m = \frac{COV_{1,2}}{s_1 s_2} = \frac{n^* \Sigma X_1 X_2 - \Sigma X_1 X_2}{((n^* \Sigma X_1^2 - (\Sigma X_1)^2)(n^* \Sigma X_2^2 - (\Sigma X_2)^2))^{1/2}}, \tag{9}$$

where, $COV_{1,2}$ means the covariance of the overlapped-segments of X_1 and X_2, s_1 and s_2 are the corresponding standard-deviations, and the summations extend only over the portions of the series that overlap at lag-position m. A cross-correlogram is plotted as match-position, m, versus cross-correlation value, r_m. Each peak in such a plot represents a match-position where there is close alignment between peaks in the fixed-signal and peaks in the lagged-copy of the signal it is being

compared to. A repeating time-step between similar appearing peaks in a cross-correlogram indicates a period of that length common to both signals. Within the data we cannot define a true lag-0 position. But, a lagged-signal is moved to the right by increasing lag-increments, so the match-position of a peak gives the time-difference by which peaks in the lagged-signal lead corresponding peaks in the fixed-signal. Similar to autocorrelation-functions, cross-correlograms suffer from long-period information being masked by strong short-period components within the input-signals. Cross-correlations are also prone to errors at long lag-values due to the resulting short-overlap between the records. Indications of high cross-correlation for long-lags should be ignored.

The significance of a cross-correlation coefficient can be determined from the approximate test:

$$t = r_m \frac{(n^* - 2)^{1/2}}{(1 - r_m^2)^{1/2}}, \qquad (10)$$

which has $(n^* - 2)$ degrees of freedom (Davis 1973). This test is related to the F-test for significance of correlation between two samples drawn from normal-populations. We expect a test-value of 0 at any match-position m if the two series are independent and random. A plot of match-position versus significance-value will show peaks and troughs corresponding to the peaks and troughs in a cross-correlogram produced plotting match-position versus r_m. Thus, the same phase and period information is available in significance-value plots and cross-correlograms.

Significance-value plots can emphasize match-positions where there is a strong correlation between signals and show that signals are different in form even when similarly periodic.

Using outputs from the program CROSSCOR (Appendix F), cross-correlograms were produced for each variable versus all other variables in the prepared data-set for lags 0 to 60 (Figure 6). Their general form is an alternating sequence of major and minor peaks. These plots show indications of strong-correlation with periods near 12 and 25 hours for any combination of signals (in all plots, consecutive peaks are separated by about 12 hours, while the most similar appearing peaks are separated by about 25 hours). This tells us only that these are, as expected, predominantly tidally-driven signals. The indicated periods correspond, within the resolution of our one-hour-increment lag-values, to the main semidiurnal and diurnal periods of approximately 12-hours-and-25-minutes and 24-hours-and-50-minutes, respectively. The alternating height of peaks in the cross-correlograms is due to the combination of these two major tidal-components, as is the case for the plot of a semidiurnal tide signal.

Selecting the tide signal as a reference we can determine the phase and period relationships for the other variables relative to the tides. From these we can find direct phase-relationships between the variables themselves. Consider the first major peak in each cross-correlogram for tidal-excursion versus other parameters (Figure 6) to mark the beginning

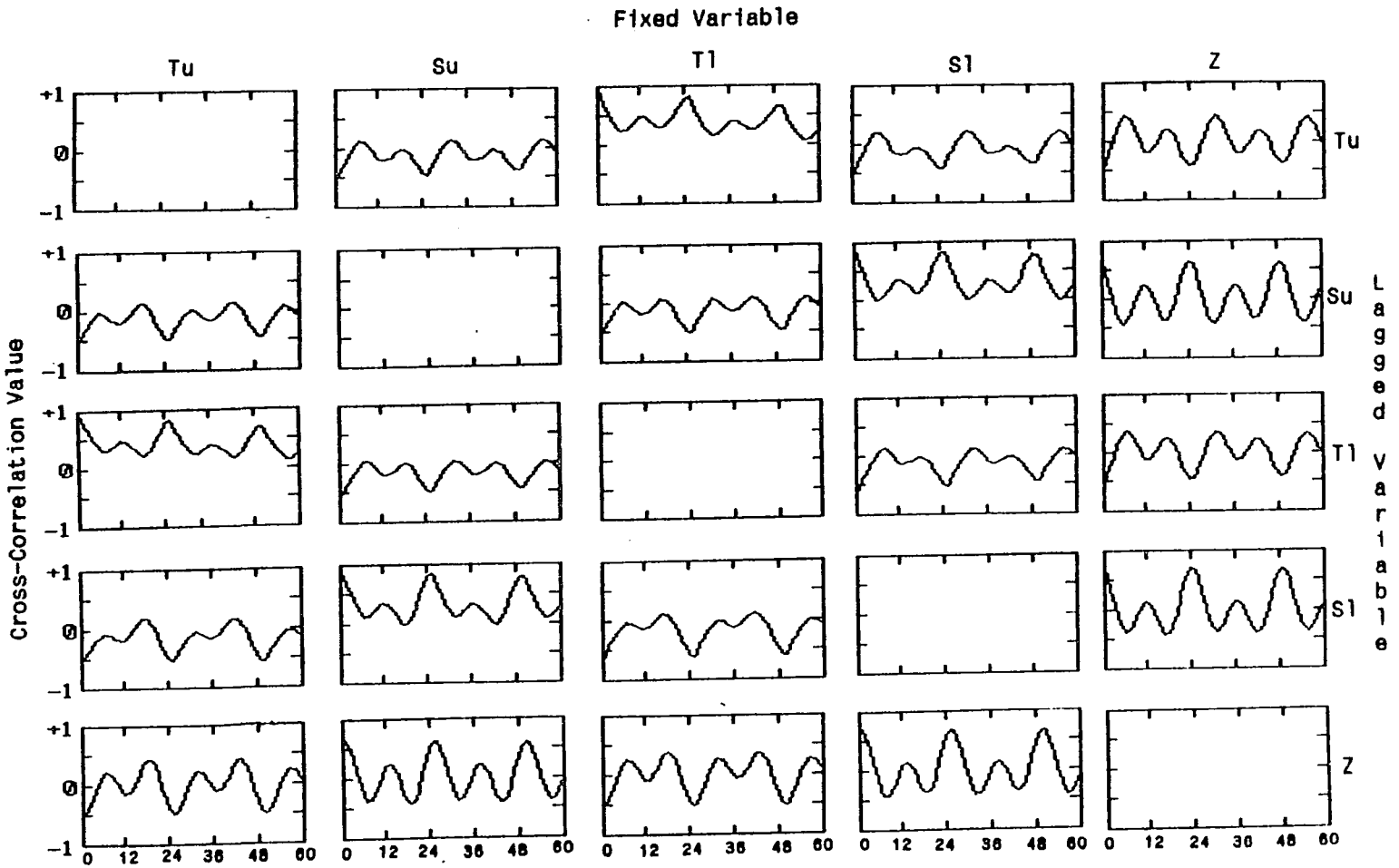


FIGURE 6: Cross-correlograms for match-positions 0 to 60 hours of all variables from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). Autocorrelations (diagonals) intentionally left blank.

of a cycle. This peak is at match-position 24-hours for both salinity signals, and at match positions 7-hours and 6-hours for the upper-temperature and lower-temperature signals, respectively. Given just lag-value information we might make the statement that if salinity signals lead the tide signal by 24 hours and temperature signals lead the tide signal by 6 or 7 hours, then salinity must lead temperature by about 18 hours. This is technically correct, but misleading. We are using the tide signal as a reference. However, in our case, its starting position is arbitrary within the tidal-cycle. The match position where signals come into synchronization depends both upon cycle length, and the position within a cycle for the start of the reference signal. Tide and salinity signals are, in general, closely correlated. Apparently, the starting point of our tide record is very close to, but slightly after the beginning of a full diurnal cycle, and essentially a full diurnal cycle of lags had to be stepped through before the major peaks in the salinity signal came into alignment with the major peaks in the tide signal. This gave their high cross-correlation-value at lag-24. The true lag-difference between the tide and salinity signals is closer to zero. This being the case, the lead-lag relationship between temperature and salinity must be close to that for temperature and tide. Which says that, within our data segment, it is more correct to state temperature leads salinity by about 7 hours.

To aid examining cross-correlograms for phase-relationships between variables you may plot only the positions of relative maximums and minimums in a cross-correlogram as vertical bars: up corresponding to a

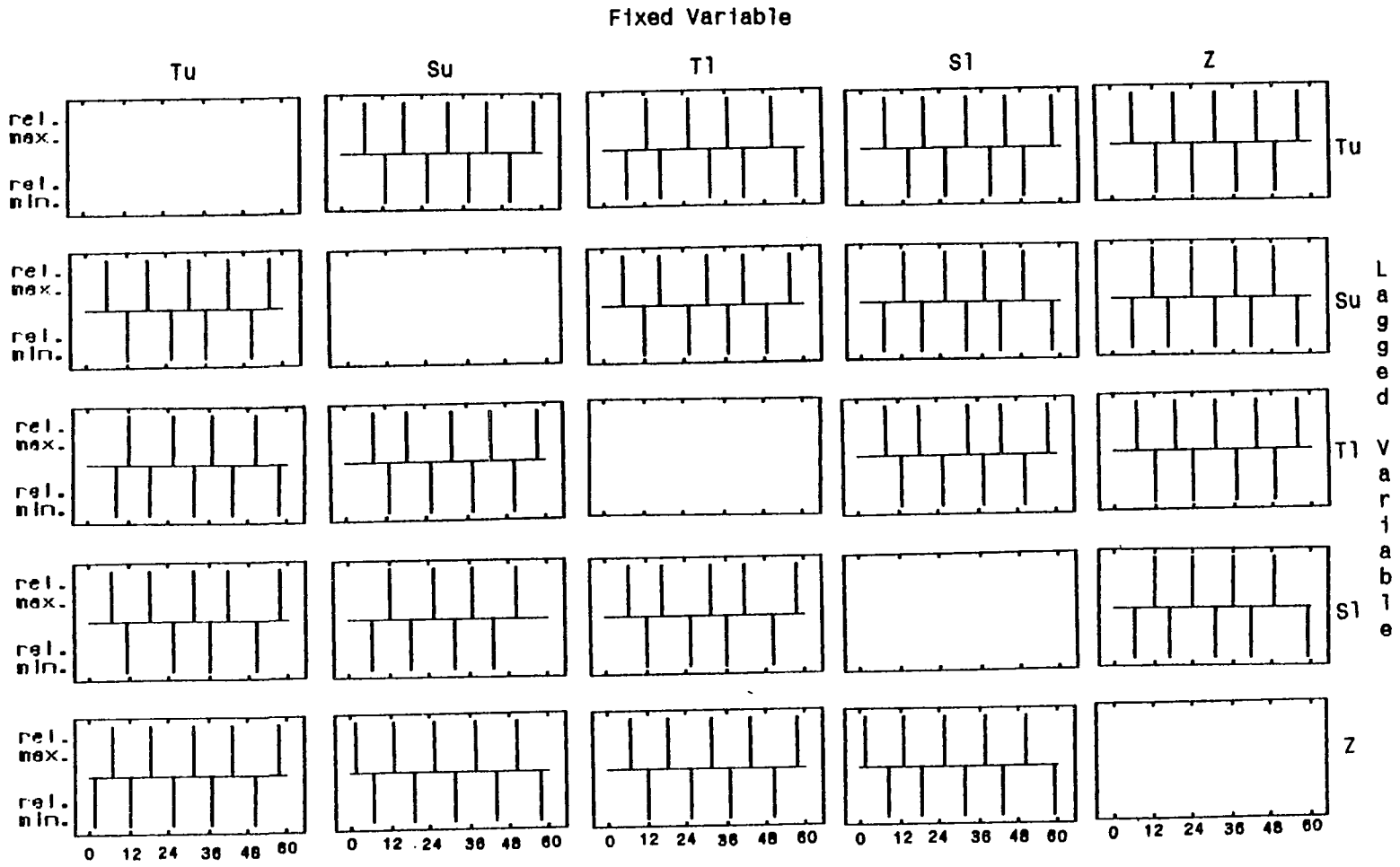


FIGURE 7: Position diagrams for relative maxima and minima from the Cross-correlograms of Figure 6.

local-maximum and down signifying a local-minimum. We can readily find the match-position of peaks by comparing such plots (Figure 7) to the original cross-correlograms (Figure 6). Note the position of the first major peak in each cross-correlogram. Given the signals are predominantly tidally-driven we might expect a fairly symmetric arrangement for these peaks. This is not the case (Table-1). As previously pointed out, in selecting the first major peak in the cross-correlograms for reference, we have not chosen a true 0-lag position. The tidal-cycle we are attempting to reference does not coincide with the

TABLE 1: Raw match-positions for the first major-peak in cross-correlograms produced from data for Upper-Temperature (Tu), Upper-Salinity (Su), Lower-Temperature (Tl), Lower-Salinity (Sl), and Tidal-Excursion (Z) at the Point San Pablo monitoring station. Match-position gives the number of hours by which a lagged-variable appears to lead a fixed-variable.

		Lagged Variable				
		Z	Tu	Su	Tl	Sl
Fixed Var	Z	--	6	24	7	24
	Tu	18	--	18	25	18
	Su	1	7	--	8	25
	Tl	18	25	17	--	17
	Sl	1	7	25	8	--

the start of the data-records, and there are inherent phase-differences between the signals. To look at these differences we must first determine, (to the resolution of our 1-hour lag-intervals), the length of

one full-cycle. From the match-position data, we see the tide signal leads both salinity signals by 1 hour, while both salinity signals lead the tide signal by 24 hours. This puts the length of a full-cycle at 25 hours, and implies the beginning of a tidal-cycle is 1 hour before the start of a cycle in salinity, though it is not a definitive statement. With these reference points we can determine phase-relationships between the other variables (Figure 8).

The largest peak in upper-temperature (Tu) occurs 6 hours before the largest-peak in the tide. The largest-peak in lower-temperature (Tl) occurs 7 hours before the largest peak in the tide. That Tu and Tl both peak and then drop before the tide peaks, with Tl starting to drop before Tu, is consistent with normal tidal action in an estuary. Salinity reaching a maximum shortly after the maximum tide is also reasonable. Riverine influence at the station could, through mixing processes, delay the maximum in salinity relative to the tide. The upper-salinity (Su) and lower-salinity (Sl) signals appear to be exactly in-phase. One might expect Sl to lead Su. Sensors are separated by 5.6 m in water not greatly more than twice that depth at the end of a pier with many piles. Under these conditions, with a moderate tide, vortices generated as water passes around pilings could mask the phase-difference between Su and Sl. It may also be phase-differences between signals are smaller than the 1 hour lag-interval used in generating their cross-correlograms. Such differences could not be observed in the given data. In fact, since all smaller phase-differences are 1 hour, the length of the lag-interval, they

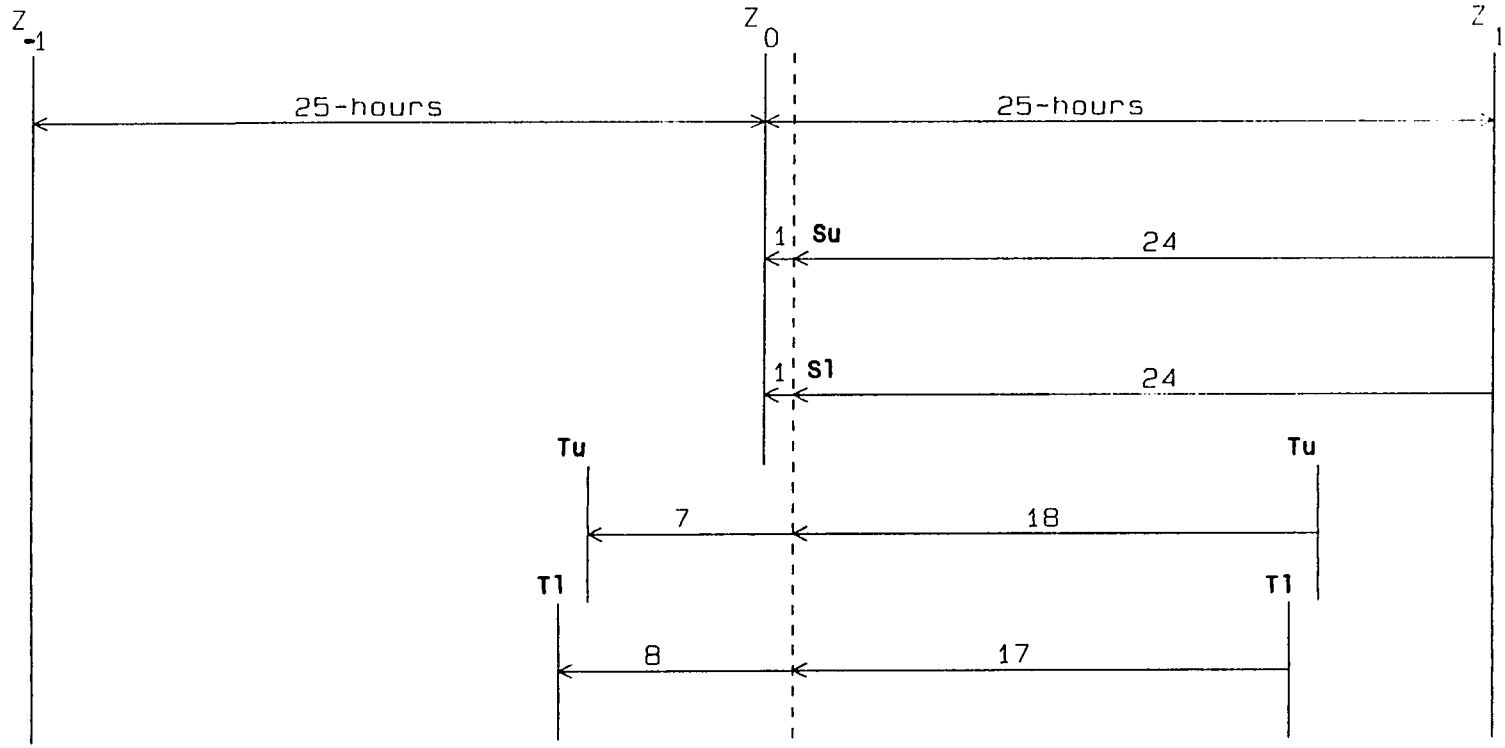


FIGURE 8: Signal phase relationships diagram derived using lag 0 to lag 60 (hours) data from Cross-correlations for all variables at the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). These data exhibit a 25 hour cycle with a 1 hour lead of tide (Z) over upper and lower salinity (Su,Sl), (the salinities appear in phase), a 7 hour lead of upper temperature (Tu) over the salinity signals, and a 1 hour lead of lower temperature (Tl) over upper temperature.

may be nothing more than 'round off' error. Rather than try to give physical justifications, it is better to 'note-then-ignore' them. From the investigation through cross-correlation we see all signals appear to be strongly periodic with a diurnal and semidiurnal component, salinity and tide signals are closely correlated, temperature signals are closely correlated, and there is a 6 to 7 hour lead of temperature signals over salinity and tide signals. These same relationships can be found by a similar examination of the corresponding significance-plots (Figure 9).

Power-Spectrum Analysis

The spectrum of a signal is a plot of frequency versus variance for frequency-components underlying that signal. The variance of a sinusiodal signal is related to the square of its amplitude, as is the energy or power of that signal. So, the spectrum is a plot of variance, or relative power-levels, for sinusoidal components of a signal. The terms 'power' and 'variance' are equivalent with reference to spectral-analysis. The term power arises from electrical-engineering, the field where these types of analyses were first developed for electronic signals. Coefficients of the Fourier-Series can be used to determine the magnitude of spectral-components, and hence to produce the spectrum of a signal. Since this spectrum defines power-levels, it is called the 'power-spectrum,' and frequently production of a spectrum by Fourier-techniques is called 'power-spectrum analysis.'

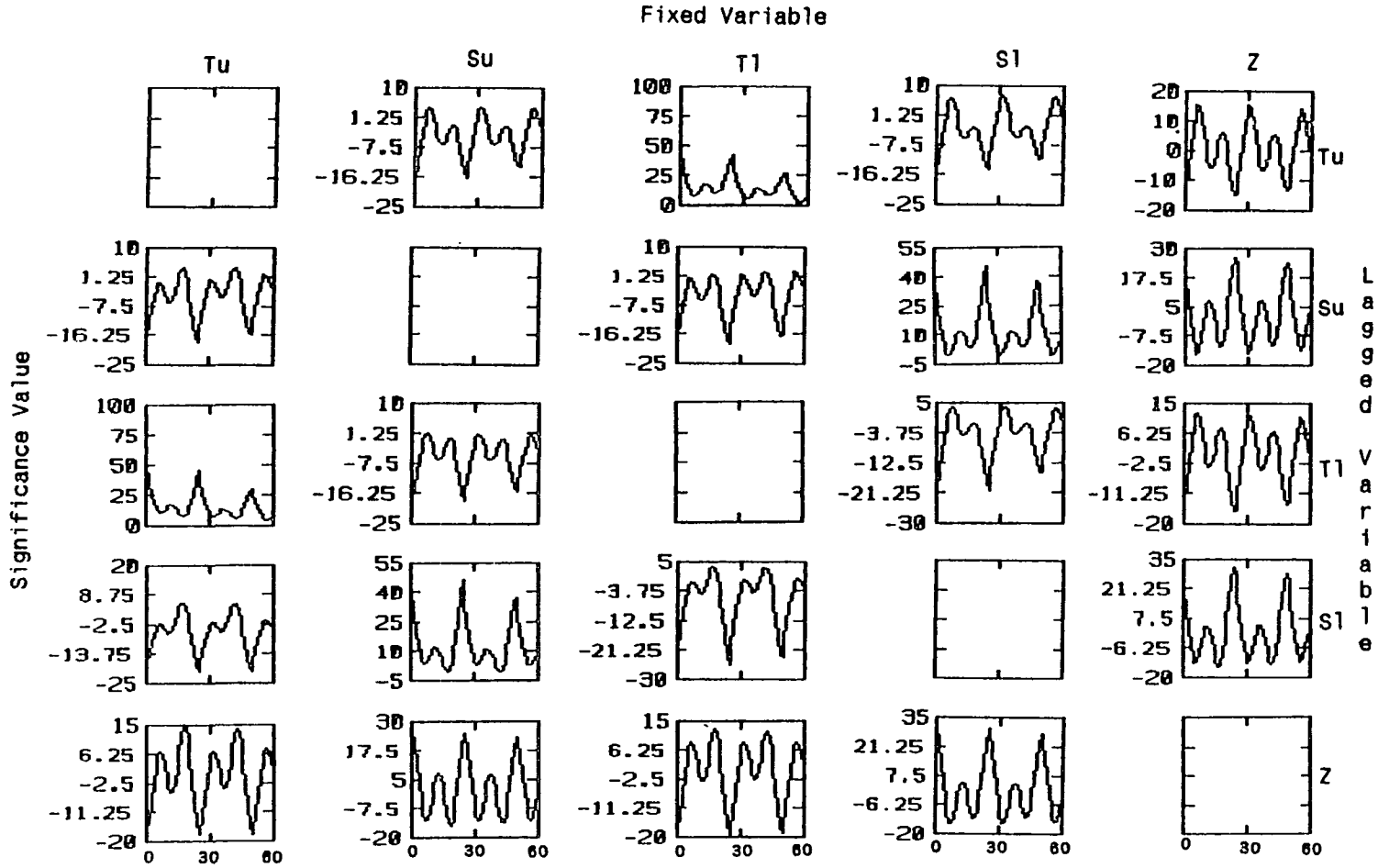


FIGURE 9: Significance-plots for match-positions 0 to 60 hours for all variables from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).

Any single-frequency signal of a particular power-level can be pictured in the time-domain as a plot of the cosine function with proper parameters. The input to the function relates to the frequency of the signal, and the energy (power) contained in the signal is expressed in the amplitude-factor applied to the function. The domain of cosine is 0 to 2π, and its range is -1 to +1. If T is the desired period for a sinusoid, then applying the factor $2\pi/T$ to the input t (time) to cosine will give an output of period T, starting from time t = 0. Adding a fixed-value to the input will shift the starting point of the cycle away from zero. For a period T, amplitude A, and p the portion of a cycle the initial zero-point is displaced from t = 0, the sinusoid may be expressed as:

$$x(t) = A\cos(\frac{2\pi}{T}t - 2\pi p). \qquad (11)$$

Period, T, is the number of time-units per cycle. So, 1/T is the number of cycles per unit-time, or frequency, f, of a sinusoid. $2\pi/T = 2\pi f$ is the radian-frequency of the sinusoid. If p in the expression for our arbitrary-period sinusoid (10) is 0, and we start at t = 0, then at t = T, $(2\pi/T)t = 2\pi$, and one full-cycle is complete. If p is not 0, then the first zero of the sinusoid is at time t = Tp, and one full-cycle is complete at time t = T + Tp. The value $2\pi p$ is the phase-angle, or phase-lag, of the sinusoid. Using the notation $\theta = 2\pi/T$ and $\phi = 2\pi p$, we can express a family of sinusoids:

$$x_k(t) = A_k\cos(k\theta t - \phi_k), \qquad (12)$$

where k, used as a multiplier and a subscript, identifies a particular member of the family. Note that the radian-frequency of all members of the family is some multiple, k, of θ, k=1,2,... This family of sinusoids (12) is harmonically-related to the fundamental-frequency $\theta = 2\pi/T$. The parameter k is the harmonic-number for a member of the family. The harmonic-number gives the number of cycles for a member-sinusoid that occur within a time-period equal to the period, T, of the fundamental, θ.

From Fourier we know that any continuous time-series, x(t), which has only one value for each time, t, can be expressed as an infinite-sum of harmonically-related frequency-components. The Fourier-series expression is:

$$x(t) = A_0 + \sum_{k=1}^{\infty} \alpha_k \cos(k\theta t) + \beta_k \cos(k\theta t). \qquad (13)$$

The Fourier-coefficients are α_k and β^k. The amplitude of harmonic-k can be calculated from the Fourier-coefficients as:

$$A_k = (\alpha_k^2 + \beta_k^2)^{1/2}, \qquad (14)$$

and its phase-angle as:

$$\phi_k = \arctan\left[\frac{\beta_k}{\alpha_k}\right]. \qquad (15)$$

For a periodic, discrete-uniformly-sampled time-series, $\{x_m\}$, m = 1,2,...,n, the α_k,β_k Fourier-coefficients can be determined from the formulas:

$$\alpha_k = (\frac{2}{k})\sum_{m=1}^{n} x_m \cos(2\pi(m-1)k/n), \qquad (16)$$

and,

$$\beta_k = (\frac{2}{k})\sum_{m=1}^{n} x_m \sin(2\pi(m-1)k/n), \qquad (17)$$

with the A_0 term being the arithmetic-mean:

$$A_0 = (\frac{1}{n})\sum_{m=1}^{n} x_m. \qquad (18)$$

For a single frequency sinusoid uniformly sampled at discrete points, the amplitude of the waveform is related to the variance of the samples. In the limit, (i.e., as the number of samples becomes very large), the sample variance becomes half the square of the amplitude of the waveform:

$$s_k^2 = \frac{A_k^2}{2} = \frac{\alpha_k^2 + \beta_k^2}{2}. \qquad (19)$$

Since the Fourier-series is a linear-combination, the variance of the sum must equal the sum of the variances of the harmonic-components. A

plot of harmonic-number versus variance (power) is a periodogram. If harmonic-number is converted into frequency, the plot is commonly referred to as a power-spectrum. Due to the nature of inverses, ($f = 2\pi/T$), a periodogram tends to show detail for long-period energy, while the spectrum tends to show short-period energy more clearly. Plots (periodograms or spectrums) generated using Fourier-coefficients are subject to large random-error. They show the relative importance of the spectral-elements, but the implied harmonic-components are not well suited to reconstructing the original waveform as a sum.

Spectrum data were generated from the prepared data using the PC-MatLab script file FFTSpec.m (Appendix F) which employs the PC-MatLab fft-function. Prepared data were low-pass filtered to help eliminate periods shorter than 2 hours, so spectrum data for periods less than 2 hours were eliminated. As pointed out in the data preparation section and Appendix A, the filtered data set was reduced to 1024 points to fit constraints of the FFT. Since it is not possible to resolve a period longer than one-half the record length, spectrum data for periods longer than 512 hours were not generated. From the spectrum data, periodograms from periods 2 hours to 512 hours and the corresponding power-spectrums for frequencies 0.001 cycles/hour to 0.5 cycles/hour were produced (Figure 10).

The periodogram for the tide signal shows indications of high energy only at the expected semidiurnal and diurnal periods. The

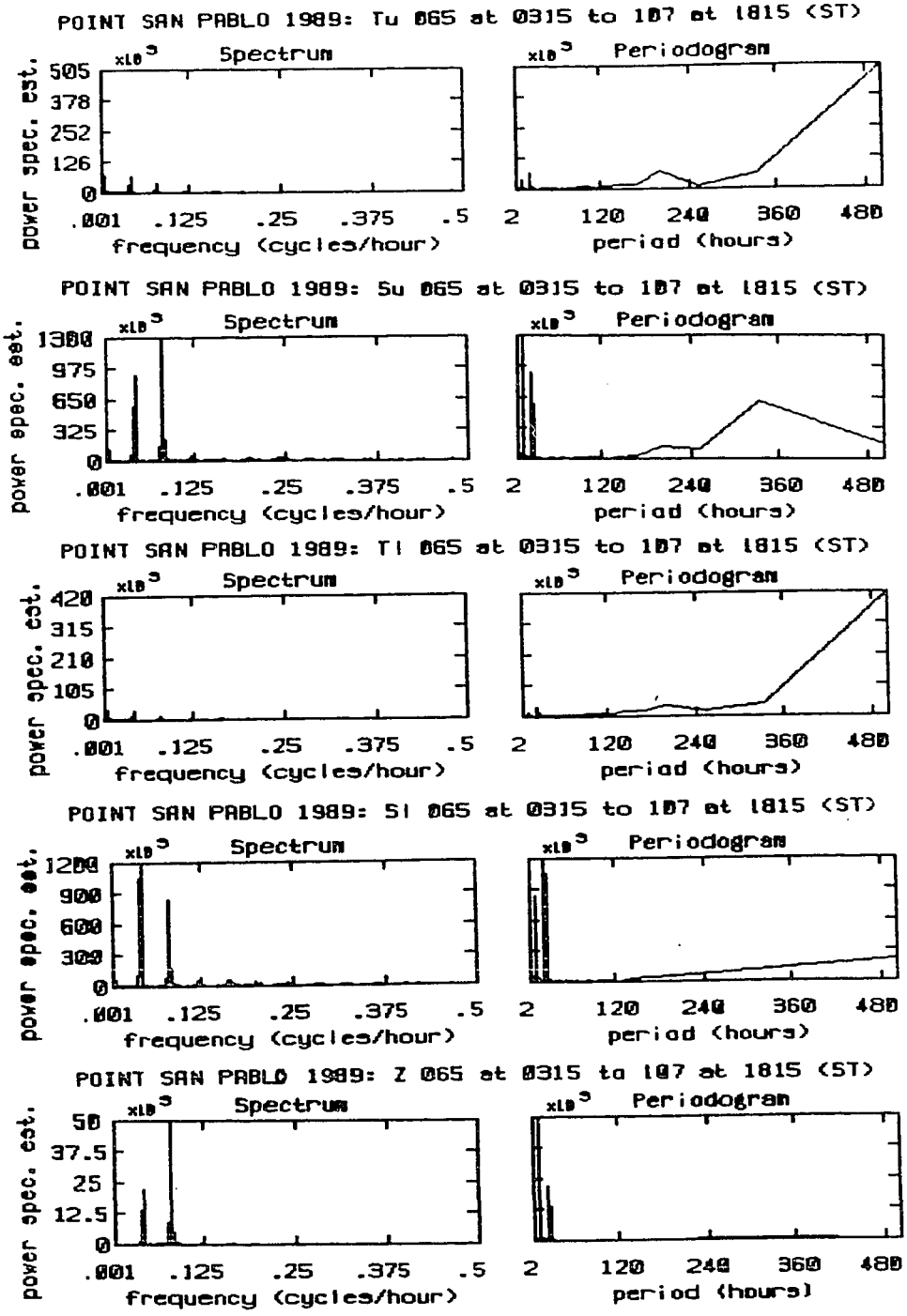


FIGURE 10: Periodograms and power-spectra for all variables from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).

spectrum for the tide signal shows energy only at the frequencies corresponding to the semidiurnal and diurnal periods. The other periodograms and spectra show peaks that correspond to the two peaks in the tide signal periodogram and spectrum. Thus all other signals are at least partially tidally-driven. The salinity signals appear more closely related to the tide signal than the temperature signals. Both salinity signals show very high energy at periods matching the periods of the tide signal. They also show some long-period energy not seen for the tide signal. Sl shows less long-period energy than Su. Long-period energy for Sl appears mainly at a period of 512 hours. For Su, some long-period energy is indicated at the 512 hour period, however, it shows most strongly at a period of about 340 hours, with lesser indications near periods of 200 hours and 257 hours. Relative to long-period energy indications, tide-related energy in the temperature signals is virtually 'swamped out'. The indicated levels of tidal-period energy in the temperature signals are close to those seen in the salinity signals, (since all signals have been scaled for similar magnitudes, that is a valid statement). However, the magnitude of the long-period energy-levels are so much higher than the tidal-energy level, the tidal-energy peaks are nearly lost. The greatest indication of energy for both Tu and Tl is at the 512 hour period. Similar to Su, both Tu and Tl show high-energy at periods near 200, 257 and 340 hours, with the 340 hour period having highest energy next to the 512 hour period. The long-period energy indications are all higher for Tu than Tl. This examination gives much the same information as the analysis using autocorrelations, but provides

more details on the nature of long-period forcing. Here we see not just that long-period affects have greater effects on temperature than salinity, but that they act more strongly on the upper-level signals than they do on lower-level signals. And we obtained estimates of periods for the long-period forcing-signals. Here we do not obtain any phase-relationship information, such as that obtained from the cross-correlation analysis.

Fourier-Transform Filter

The Fourier-series is actually applicable only to periodic time-series of infinite length. Extending the Fourier-series to encompass nonperiodic signals, signals in the presence of random-noise, and records of finite length results in the Continuous-Fourier-Transform (CFT) which converts the Fourier-series sums into integral-form. Using Euler's identities within the cosine and sine expressions of the CFT, the equations reduce to a transform-pair:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(w) e^{iwt} dw, \qquad (20)$$

$$X(w) = \int_{-\infty}^{\infty} x(t) e^{-iwt} dt, \qquad (21)$$

where X(w) (21) is the Fourier-transform of x(t), and x(t) (20) is the inverse-Fourier-transform of X(w). If x_n is a discrete-time signal of

length N, $0 \leq n \leq N-1$, then, letting $2\pi/N = \theta$, the discrete form of the inverse-Fourier-transform (20) is:

$$x_n = (\frac{1}{N})\sum_{k=0}^{N-1} X_k e^{i\theta kn}, \qquad (22)$$

and the coefficients, X_k, are given by the discrete-Fourier-transform:

$$X_k = \sum_{n=0}^{N-1} X_k e^{-i\theta kn}, \qquad (23)$$

With the discrete-Fourier-transform (DFT) pair (22, 23), we may construct the spectrum of nonperiodic, discretely-sampled signals. And given the N-point DFT of a function, we can always construct a Fourier-series that recovers the function exactly, at the sample-points.

With the ability to transfer back and forth between the time and frequency-domains for discrete nonperiodic-signals via the DFT, we are given a way to filter specific frequency-components from data records. The basic procedure is to generate a frequency-domain spectrum for the signal to be filtered, remove from that spectrum the spectral-elements corresponding to the frequency-components to be filtered, then convert back to the time-domain using the remaining elements from the spectrum. Since the DFT recovers a signal exactly at the sample-points, the conversion back to the time-domain from the spectrum with elements removed produces a signal that, at the sample-points, is equivalent to

the original signal with the periods corresponding to the spectral-elements eliminated in the frequency-domain removed. This picture is complicated by leakage and Gibbs-phenomena. Leakage is ringing (noise) in the Fourier-transform of a signal caused by discontinuities in the time-domain signal. Gibbs-phenomena is ringing in the inverse-Fourier-transform caused by discontinuities in the frequency-domain spectrum. Gibbs-phenomena ringing occurs near discontinuities in the time-domain signal.

All time-series records have at least two abrupt discontinuities: the beginning and end. To reduce leakage in the spectrum, the ends of a time-domain record are 'tapered' before the record is Fourier-transformed. That is, a function is applied at each end of the record that produces smooth transitions to zero. The act of eliminating spectral-elements as a filtering-technique introduces discontinuities into the spectrum. To reduce the resulting Gibbs-phenomena in the inverse-transform, the spectrum is tapered to cause smooth transitions to zero at the location where spectral-elements were removed. Both the tapering applied to reduce leakage, and the Gibbs-phenomena, adversely affect filter-results near the ends of the record. Some portion of each end of the filtered-record must be discarded or ignored.

Any type of filter may be implemented using the Fourier-transform technique. Here we are interested in determining what if any non-tidal forcing there is in the signals represented by our time-series records. In our case, non-tidal means low-frequency, so we filter to remove the

predominant tidal-components and any higher-frequencies. That is we implement a low-pass filter. Via the program FOURIER.FILTER (Appendix F), 3% of each end of the data-records was cosine-tapered and the tapered records Fourier-transformed. The resulting spectrums were zeroed for all components of period 30-hours or less, and a cosine-taper applied to the spectral-elements for period 30-hours to period 40-hours (see Appendix B). These spectrums were then inverse-Fourier-transformed giving records for each signal with the principle tidal-components removed according to the stop-period of 30-hours and pass-period of 40-hours (Figure 11). These records support what we have seen in the previous examinations. The tide-signal is relatively clean of frequencies lower than the diurnal-cycle. The temperature and salinity signals show effects due to long-period forcing. These effects seem more pronounced in temperature than salinity, and more pronounced in upper-records than lower-records. There appear to be several non-tidal forcing-periods, the primary one being the longest; about one-half the length of the data-segment, or 512-hours.

Summary

Each of the spectral-analysis techniques investigated here gives some useful information about the data-set. All give indications of periods underlying the data. However, phase-information is available only through cross-correlation. If limited to one technique, forming periodograms and spectrums via the Fourier-transform will give the most

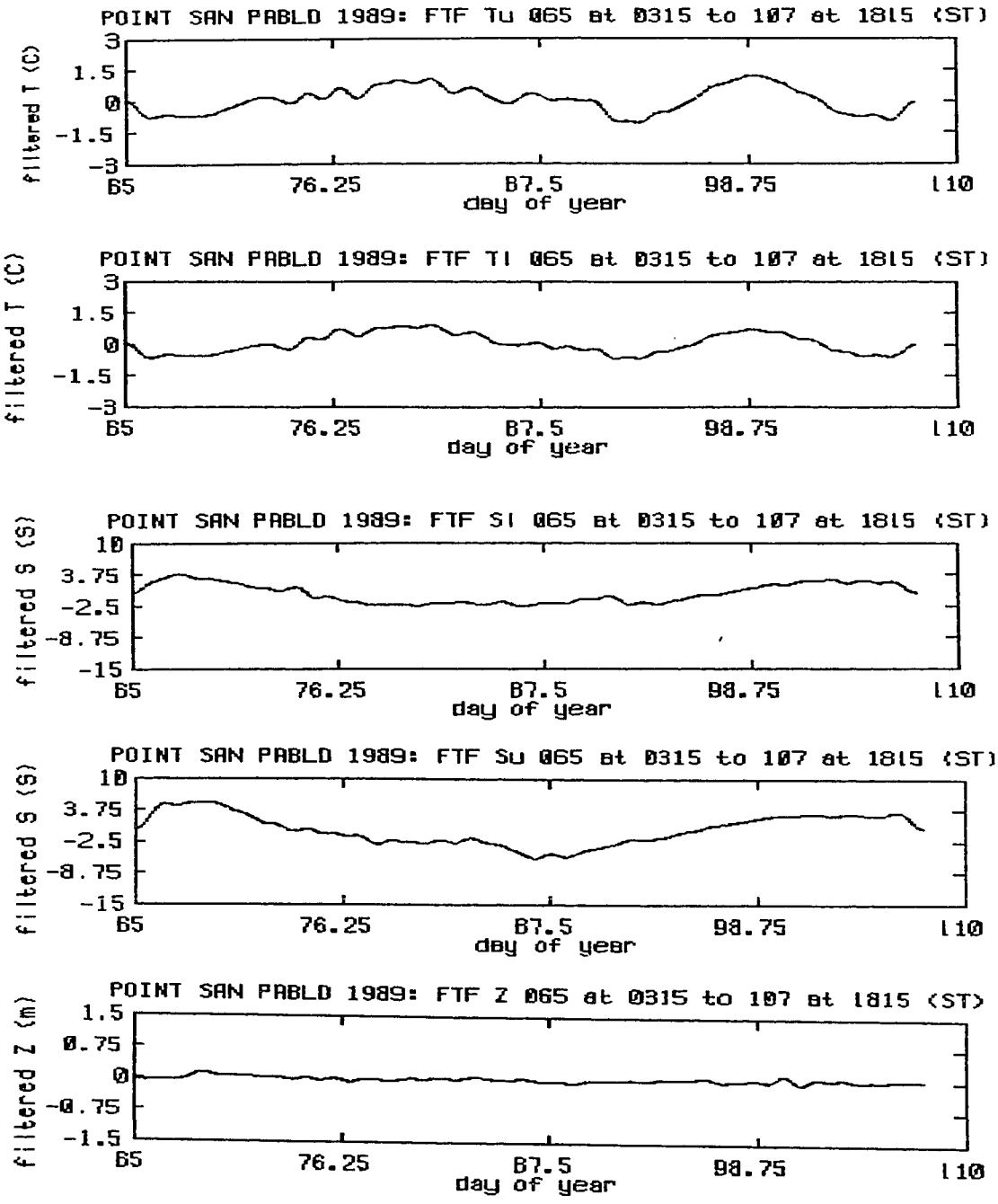


FIGURE 11: Fourier filtered data records from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).

information. These can show what periods are present in the data-set, and how significant these periods are to signals within the data-set. Autocorrelation can also show these aspects of the data-set, but spectral-data are less prone to having long-period information masked by repeating shorter-periods. Given no requirement to write the programs, there is little difference in the effort required to perform the rote operations for any of the analyses. Cross-correlations require more effort to interpret than the other techniques, so if phase-information is not required it is the least desirable. The Fourier-transform filter is the best of these techniques for seeing the form of non-tidal forcing, but, without running it a number of times with different stop and pass period values, gives the least amount of actual numerical information.

Factor Analysis

Introduction

Often the term 'variable' is used interchangeably to express one data value within a data record, or to identify the entire data record that contains an individual data value. Usually this presents no problem, as the meaning is clear in context. However, some techniques, such as factor analysis, require more rigid definitions. Here we will use 'variable' to mean an entire data record, containing all the readings of a measured parameter.

We form a data matrix by placing variables side-by-side as an array. A covariance or correlation matrix of a data matrix is an array formed of the covariance or correlation values for all possible combinations of variables in the data-matrix. This will be a square matrix whose size is the number of variables in the data matrix. Factor analysis is a multivariate statistical technique that attempts to find the underlying structure of a data matrix based on an examination of the covariance or correlation matrix of that data matrix. By underlying structure we mean a set of variables that can be used in a linear combination to reconstruct the original set of variables in the data matrix. This may be considered analogous to producing a complex waveform as the sum of its underlying harmonic components through spectral analysis. Here our underlying components are not necessarily sinusoids, and they are used to generate

multiple variables, not just one. We look for a set of underlying structure variables explaining the variance seen in our original variables, which is fewer in number, and, by some criterion, of 'simpler' form, than our original variable set.

There are two major forms of factor analysis: R-mode and Q-mode, with many variations on a theme. R-mode methods look for relationships between variables in a data matrix. Q-mode methods look for relationships between rows, (referred to as 'objects'), in a data matrix. Factor analysis is an approximation or modeling technique similar to regression-analysis, only there is no distinction between dependent and independent-variables. All variables are considered equally in factor analysis. It is assumed all data variables are described by some combination of all underlying structure variables. R-mode techniques are used to develop models for the variables of a data matrix. Q-mode methods consider each object in the data-matrix as an entity and look for relationships to describe variations in the sequence of objects contained in the data matrix. As is common for studies in physical oceanography and meteorology, we are looking for ways to describe our variables as individual entities and choose R-mode analysis. In particular, here we use R-mode principle components analysis, with rotation. This technique and the underlying mathematics are described in Appendix C, (which includes a fully worked example small enough to be preformed by hand), and Appendix E. Below are given only broad descriptions of the operations, and the results of applying them to the prepared data.

Principle Components Analysis

A variable may be viewed as a vector which describes the location of a point in n-dimensional space, where n is the number of elements in that vector. Finding a linear system that represents the underlying structure of a matrix amounts to finding a basis-vector set for the column-vectors of that matrix. What that means is our analysis looks for a set of 'axes' that may be used to describe the matrix variables in a manner analogous to using x, y, and z-coordinates to describe a point in 3-dimensional space.

One way to describe a set of variables is to determine the amount of variation in the variables that is common to all variables. That is we determine a single variable that will, in terms of modeling as a linear combination with a least-squares fit, most fully describe all the variables in the data-matrix, i.e., determine a single variable whose 'wiggles' match as closely as possible the 'wiggles' in all other variables, simultaneously. Put most simply, we find the 'average' component. Given this first 'principle component,' which, in terms of variance, most closely matches all variables, we can determine a set of weights to apply to this component so that the smallest possible residual can be determined from each original variable and the weighted principle component. For this set of 'smallest possible' residuals, a new principle component can be determined, along with a new set of weights to apply in forming the next set of 'smallest possible' residuals. This process can be continued

until some desired level of explanation of variance in the original data-matrix is reached. This is the basic idea of principle components analysis. Each component explains the maximum possible amount of variation in a data matrix after the variation explained by the previous component has been removed. The components form an ordered set; the first (average) component explaining most of the variance seen in the data matrix, the next explaining the next greatest amount of variance in the data matrix, and so on. This set of components and their corresponding weights can be used in a linear combination to model the original data matrix to any desired level of explanation of variance. The principle components are a set of axes underlying the data-matrix. These axes are in 'variance space,' and the weights represent the 'distance' (amount of variance) to 'travel along' (attribute to) an axis in finding a point in the space represented by the data matrix variables. The weights are referred to as 'factor loadings,' and an 'axis-weight' combination is a 'factor' in determining a variable. The number of factors required to explain a significant portion of the variance observed in the data matrix is usually less than the number of original variables.

The relationships with respect to variance between variables in a data matrix are expressed in the covariance or correlation matrix of that data matrix. The correlation or covariance matrix is called a similarity matrix for the data matrix. The choice of using covariance or correlation as the similarity index is made using considerations like those for performing correlations over covariances for a pair of variables. The

values within a similarity matrix are subject to the same scale dependence affects as their bi-variate counterparts. If covariance is used, then the produced axes and weights can, to a chosen level of explanation of variance, be used in a linear combination to reproduce the original data matrix variables. However, simply reproducing the original variables is not the point of factor analysis. Factor analysis is used to approximate the form of the underlying structure of a data matrix in order to determine processes underlying the data matrix variables. A set of variables that can reproduce, via linear combination, another set of variables may or may not say anything about underlying processes. When correlations are used, factors are normalized. They will not, as a linear combination, reproduce the original data-matrix variables. However, they do express the level of interdependence between the original variables and the factors, without scaling errors. That is, their forms more closely expresses the 'true' forms of the underlying structure variables. If variance levels are close for the individual variables in a data matrix, then covariance is appropriate. If the magnitudes of these levels are 'significantly' different, then correlation must be used. Since we are interested in the form of underlying processes, and not just arbitrarily explaining variance, correlation is most often used for the similarity index, regardless of the units of the variables.

In examining a similarity matrix we determine a set of principle component axes based on the variance relationships expressed there, and avoid working directly on a potentially very large matrix. (The 5x1024

data matrix of this thesis is reduced to a 5x5 similarity matrix.) The axes and weights are determined as the eigenvectors and eigenvalues of the similarity matrix, respectively. (This technique is sometimes referred to as eigen-analysis.) The eigenvectors form an orthogonal basis-vector set for the columns of the similarity matrix, and the eigenvalues express the amount of variance that may be attributed to the axis represented by its corresponding eigenvector. Through matrix operations these 'axes' and 'weights' can be used to determine (approximations to) the underlying structure variables of the data matrix. These variables will explain, to some selected level, the variance seen in the original data matrix. However, the technique is indeterminant. That is, there is an infinite number of possible solutions to the problem. The derived principle component axes may have no relation to processes in action within the variables of the data-matrix. They must be subjected to interpretation. If their form makes no sense with respect to the form of the original data matrix variables, then the initial analysis results are meaningless, regardless of how well they explain the observed variance. In this (frequent) case, rotation of factors may result in new forms for the approximations to the underlying structure variables that 'make sense' with regard to the type of variables under investigation.

Rotation

Rotation of factors is simply what it sounds like, a rigid rotation of the produced factor axes. There are several accepted rotation techniques.

Here we use normalized varimax-rotation (Kaiser 1959) with the modification suggested by Wingersky (Harman 1976). The purpose of rotation is to simplify structure in the factor loadings (i.e., to reduce the number and magnitude of the weights applied in generating the underlying structure variable approximations). Rotation attempts to force the maximum amount of variance explained for each variable to come as much as possible from only one factor. The basic procedure is to, in pairs, reorient the factor axes to minimize the variance explained for a variable by one axis, while maximizing the variance explained for that variable by the other factor. This process is continued iteratively until a 'maximum of simplicity' has been obtained according to the normalized varimax simplicity criterion. Once this criterion has been achieved, new factors are calculated based on the new factor loadings resulting from the axes rotation process. At this point all that is possible through factor analysis has been done. It remains the job of the investigator to determine if the rotated factors seem to be a more, or less, reasonable explanation of variance seen in the data matrix than the original principle components.

Analysis Results

The PC-Matlab script FACAN.M and its adjunct programs, PRNCMP.M, VARIMAX.BAS and ADDTEXT.BAS (Appendix F), were used to produce a set of principle component and rotated factor axes for the prepared data, using correlation as the symmetry index. Three principle

components were sufficient to explain 96.0% of the variance observed in the original data-set, at levels of 70.6%, 16.9%, and 8.5% variance explained. After rotation the corresponding variance levels for the factors were 32.4%, 40.9%, and 21.8%, respectively. Apparently rotation 'spread out' the distribution of variance. If we assume there should be one primary source of variation we have lost, not gained information by rotation. There is no guarantee that rotation will improve results, and this may be a case where it does not. However, it may be influences on variance are acting somewhat equally within the data-set and the rotated axes express this. Further interpretations are necessary before any definitive statements can be made.

A factor-score is a value calculated from the linear-combination approximating an underlying structure-variable at a particular position in the time-series sequence. Looking at a plot of time versus factor-score is the easiest way to examine the results of a factor analysis procedure. These plots show directly the form of the approximations to underlying structure-variables. Factor score plots from the principle-component axes (Figure 12) show forms that are similar to the forms of the original variables (Figure 4), as do the plots for the rotated-component axes (Figure 13).

The first principle component scores (Figure 12) appear in form most like the salinity signals (Figure 4), while the second principle component scores (Figure 13) look most like the temperature signals (Figure 4).

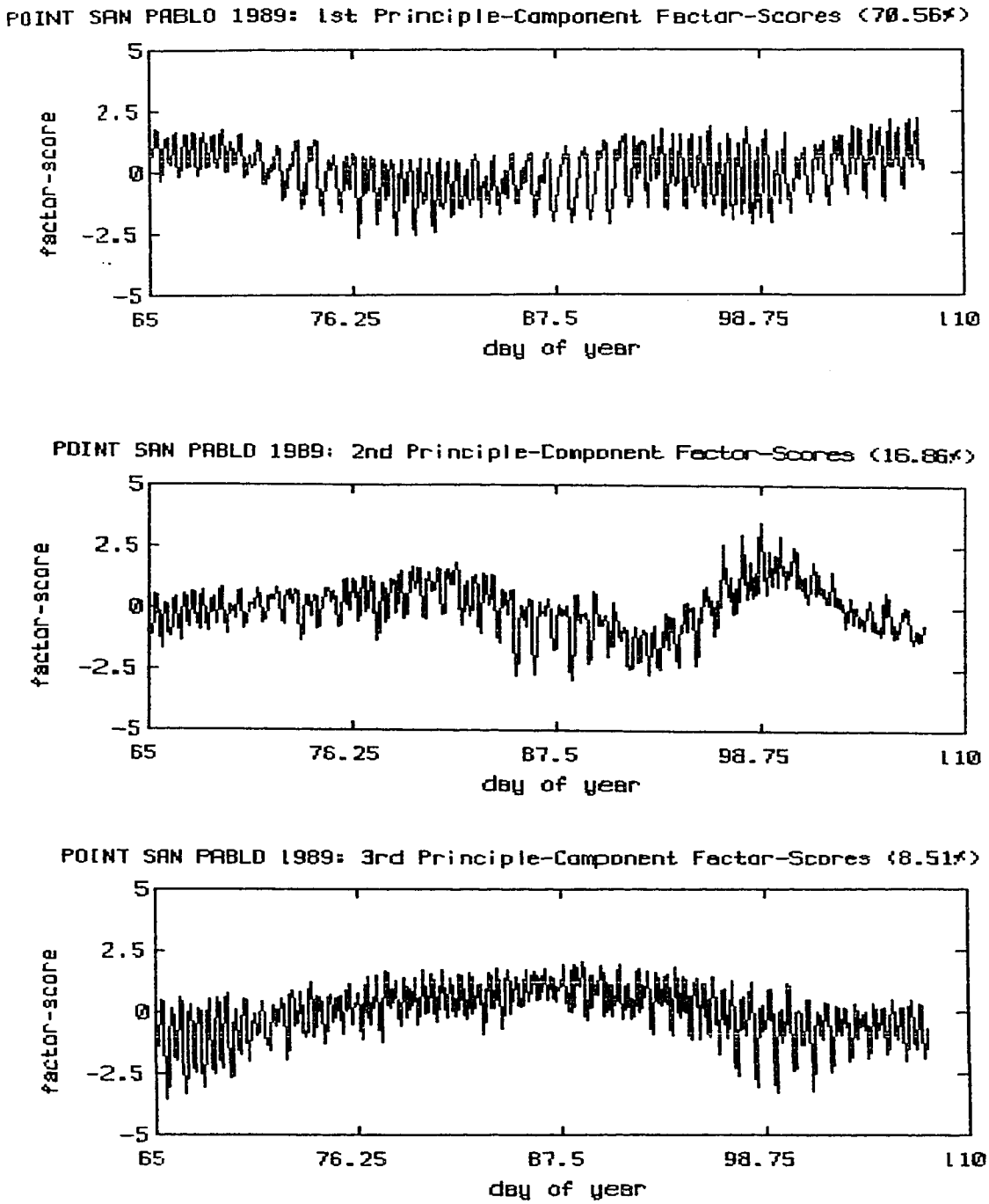


FIGURE 12: R-Mode principle component scores from Point San Pablo water quality monitoring station data for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST)

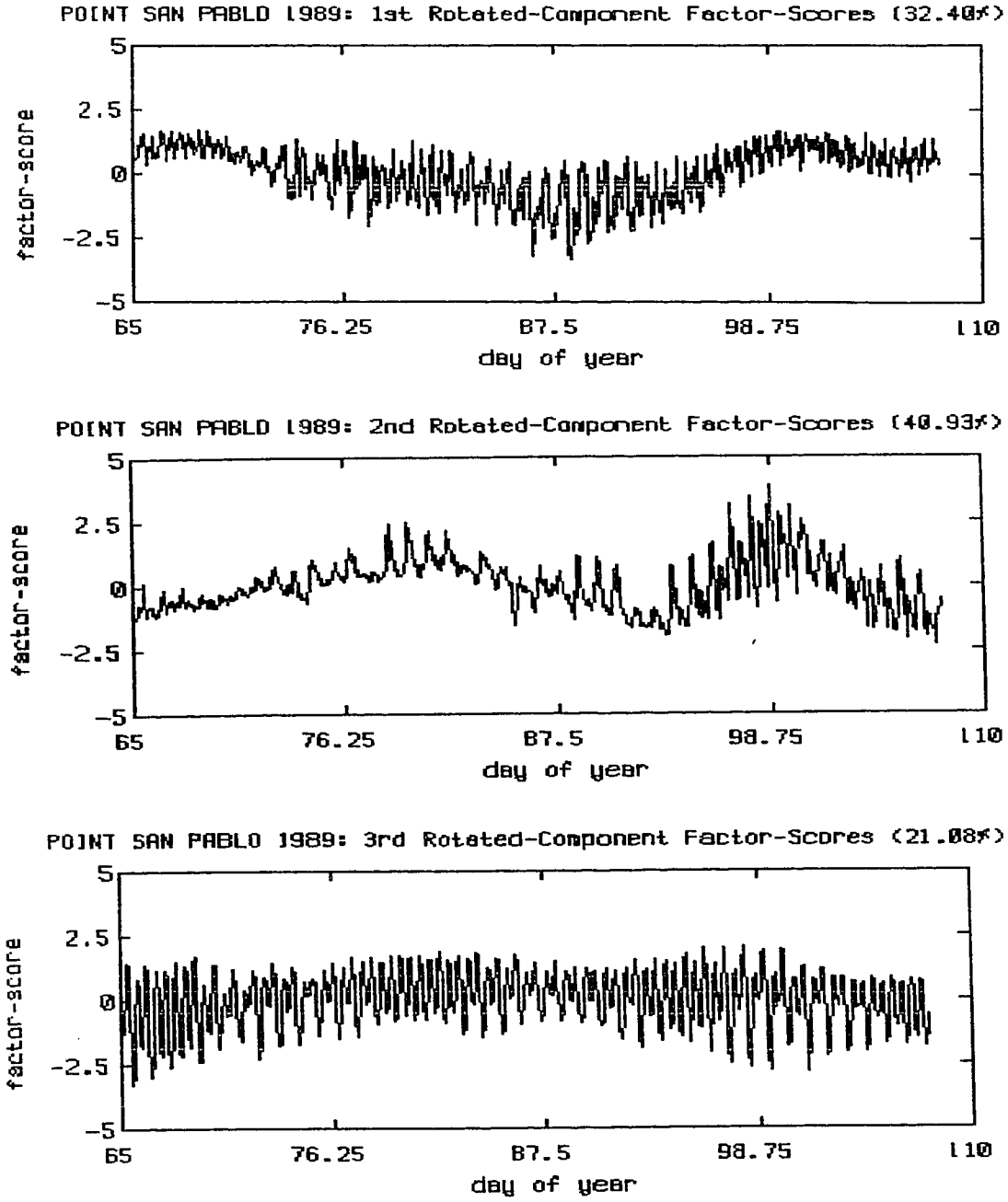


FIGURE 13: Rotated R-Mode factor scores from Point San Pablo water quality monitoring station data for the period 06 March 1989 at 0315 to 17 March 1989 at 1815 (ST).

Factors related to these two scores explain 87.4% of the total variance seen in the original variables. This is marginally useful information, stating mainly that our signals look a lot like our signals. 'Squinting' slightly we could say the first principle component scores plot appears strongly related to the tides. Ignoring high frequency variations, the second principle component scores plot exhibits the underlying form of the temperature signals. We might infer signals are predominantly tidal in nature, but significant long period forcing acted during the sampling period, mainly affecting temperature.

Scores plots for the axes after rotation resemble scores plots for the axes before rotation. However a significant change in levels of variance explained was indicated for corresponding factors. The form of the scores plot for the third factor showed the greatest change from the form of its unrotated counterpart. The third unrotated scores plot looks a little like an inverted version of the second unrotated scores plot. The scores plot for the rotated third axis looks very much like the original tide signal. The plot for the second rotated axis looks like that for the second unrotated axis, but with much less high frequency information; exhibiting the underlying 'wave' seen in the temperature signals more clearly than does the plot for the unrotated axis. The rotated scores plot for the first component looks more like the salinity signals than the corresponding plot for the unrotated axes. The form of scores plots for the rotated axes indicate three major forcing modes were in operation during the sample period. Tides, by form related to rotated component

three appear to explain about 21% of the total variance, and thereby have the least significant effect on the form of the variables in the data matrix. The rotated first component explains approximately 32% of the observed variance, and in form appears related to depression in the salinity signal. The most significant factor in the rotated set is related to the second component. This explains nearly 41% of the variance in the original signals, and in form appears related to long term pulses in temperature.

Although the unrotated information gives a single factor that can be use to explain over 70% of variation observed in the data matrix, the rotated information provides more satisfactory explanations for the form of the original variables from a physical point-of-view. One factor, appearing to be a tidal mode, was separated, leaving two other distinct modes, one appearing related to depression in salinity, and one appearing related to pulsing in temperature. The separate modes all show some diurnal and semidiurnal variation. We have detected modes that appear believable. But, we have no information as to their nature or origin. We can compare the form of these modes to the form of adjunct variables such as precipitation and insolation data, and perhaps determine their source. Factor analysis by itself says nothing about processes. It only finds groupings of variables in a data matrix that are significantly related in terms of their variance. It produces a smaller set of variables that will explain the variance observed in the original variables up to some predetermined level. Statistical procedures, including factor analysis,

cannot be used to find causal relationships. It is always the job of the investigator to interpret factors.

The correspondences we developed above through visual examination of factor-scores plots and comparing their form to the form of plots of the original data matrix variables can also be seen in correlations of the variables with the factors. These come out as part of the factor-analysis process (Table 2). Rotated factor-3 is most strongly related to the tides ($r = 0.91$). Rotated factor-2 is most strongly related to temperature (Tu $r = 0.94$, Tl $r = 0.90$). Rotated factor-1 is most strongly related to salinity (Su $r = 0.92$, Sl $r = 0.76$). Further relationships may be seen in these correlations, e.g., by correlations relative to the third rotated factor, salinity signals are more strongly related to tides than are the temperature signals, and the relation between temperature and tide tends to be negative.

For a data matrix whose variables are only moderately long and contain a great deal of high-frequency information, e.g., a 42-day record from a water quality monitoring station on San Francisco Bay, interpretation of the results of a factor analysis procedure requires somewhat a willingness to believe what you believe; high-frequency variations override the underlying low-frequency variations we are trying to separate from the record which makes interpretation difficult. Factor analysis requires a significant amount of computation and effort in interpretation. It is of limited use for direct application to short records

of tidally driven parameters. However, it can be used to help determine the nature of long-period forcing, by applying it to records which have been low-pass filtered to remove tidal variations (Walters 1982).

TABLE-2: Correlations of the Point San Pablo station data matrix variables with factors from R-mode principle components analysis with rotation.

	factor							
	principle				rotated			
variable	1	2	3	4	1	2	3	4
Upper-T	-.82	0.53	-.03	0.16	-.25	0.94	-.19	0.08
Upper-S	0.85	0.34	-.32	-.23	0.92	-.28	0.27	-.04
Lower-T	-.86	0.47	-.00	-.16	-.26	0.90	-.20	-.25
Lower-S	0.88	0.32	-.19	0.25	0.76	-.30	0.36	0.43
Tide	0.77	0.35	0.54	-.04	0.33	-.23	0.91	0.07

Time-Domain Filtering

Introduction

The primary application of time-domain filtering in this thesis was to precondition data for analysis. The data-record time-step was increased from 15-minutes to 1-hour, and periods shorter than two hours were removed or reduced to less than their original half-power levels using a 9-point binomially-weighted transversal filter. The rational for this operation is covered in Appendix A. The filter is described in Appendix D.

To investigate time-domain filtering as an analysis technique, the prepared data were passed through a low-pass tide-removal filter implemented, following the principles in Godin (1972), as a weighted running-average to remove periods shorter than 24 hours. Although we define the, so called, Godin filter in terms of its frequency-domain operations, it is considered a 'time-domain' filter because it operates directly on time-domain information. There is no explicit transform to the frequency-domain to remove specified frequency components when using this filter. The Godin filter is described in detail in Appendix D. Below we present only basic principles and the results of using it on the prepared data of this thesis.

Time-Domain Tide-Removal Filters

Formally, time-domain filtering is referred to as convolution. A convolution is simply a weighted-sum of a set of values. We don't normally think of it as such, but finding an average value is performing a convolution of a data set. Each element is multiplied (weighted) by 1/n, where n is the number of elements in the set, and those weighted elements are summed to form the average. So, averaging is a convolution, convolution is filtering, and, in fact, averaging is low-pass filtering.

When we form a simple average, we make the implicit assumption that the numbers being averaged are spaced at equal intervals of time, distance, or whatever unit is appropriate for separation of the values. An average not only finds the 'mid-value' of a data set, it also by implication places that mid-value at the midpoint of the 'space' occupied by that data set. If we average a set of readings taken at fixed increments over a 24 hour period, the average value should be placed at 'noon' relative to the readings being averaged. If readings are not at uniform increments, then placement of the average must be made with consideration for spacing of the values being averaged. Here we assume that all readings are a fixed interval.

If we perform a 24-hour average and end up with one point at noon in place of the original 24-hours' data, then we obviously no longer see

variation in the data with a period shorter than 24 hours. If the data set is several days long we can perform a sequence of 24-hour averages, stepping through the data at an increment equal to the sample interval. This will generate a new sequence of points separated by the sample interval, but filtered of variations with periods shorter than 24 hours. The first and last points of the filtered sequence will be at noon of the first and last day of the record. We lose one-half day's length from each end of the data by this diurnal filtering (Figure 14). This is the basic principle behind a time-domain tide-removal filter. However, since a diurnal cycle is not exactly 24 hours, our simplistic 24-hour filter will not perform an adequate job of tide removal.

The Godin filter is a low-pass time-domain filter that removes diurnal and semidiurnal tidal components. It is based on running multiple weighted-averages of one-hour increment data over filtering intervals of 24 hours and 25 hours. It requires 72 points for one iteration. The first filtered point falls 35.5 hours after the beginning of the input record, the last filtered point falls 35.5 hours before the end of the input record, for 71 hours of data lost. The program GODIN.FOR (Appendix F) uses the algorithm for this tide-removal filter as presented in Godin (1972, pp 65-66). Input data start on day 065 at 0315, so the first filter output is on 066 at 1445. Filter outputs are offset 30 minutes from the input data. They are interpolated in GODIN.FOR to match the times of the input-data starting on day 066 at 1515.

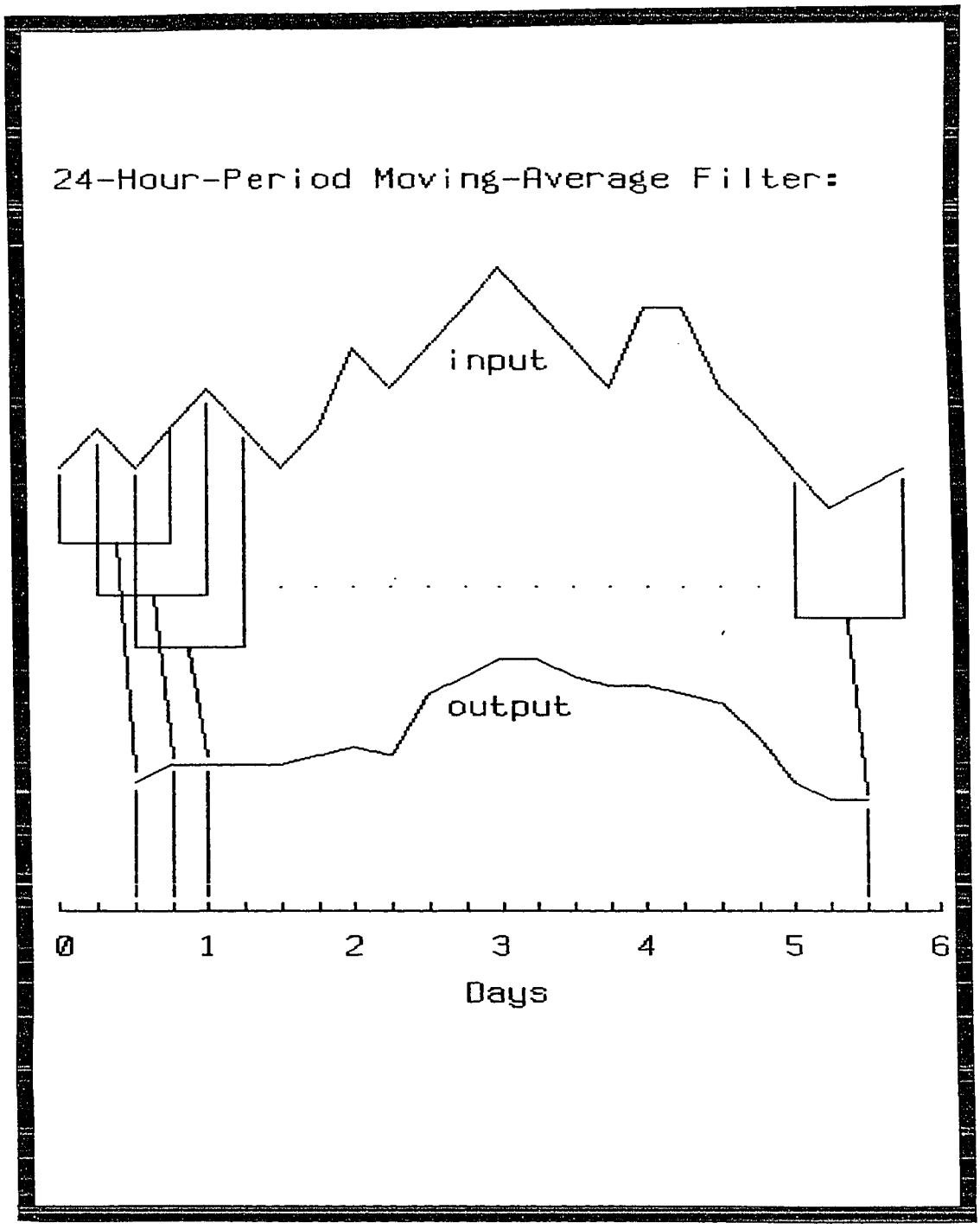


FIGURE 14: Moving-average diurnal filter applied to a data set with four samples per day.

Analysis Results

The Godin filter results (Figure 15) look very similar to the outputs from the Fourier-transform filter (Figure 11). The Godin filter outputs are slightly 'smoother' than the Fourier-transform filter results. This means they have less short-period information than the Fourier-transform filter outputs. Both filters remove diurnal and semidiurnal components, so the increased detail available from the Fourier-transform filter output may be an asset in analysis for residuals. The Godin filter must lose approximately 1.5 days from each end of the input-record just by the nature of its algorithm. Three-days loss may be significant to studies involving short records. The Fourier-transform filter output is the same length as its input-record. While the ends of Fourier-transform filter output records are in question due to possible Gibbs-phenomena effects, and from the affects of the taper applied to the records before performing the transform, the amount of data that must be ignored or questioned is less than that completely lost when using the Godin filter. Given the higher-detail output, flexibility in selecting stop period, pass period and data interval, and the smaller amount of data loss for the Fourier-transform filter over the Godin filter, the Fourier-transform filter is more desirable for residual analysis of the type being examined in this thesis than the Godin filter. The Fourier-transform filter is also computationally more efficient than the Godin filter.

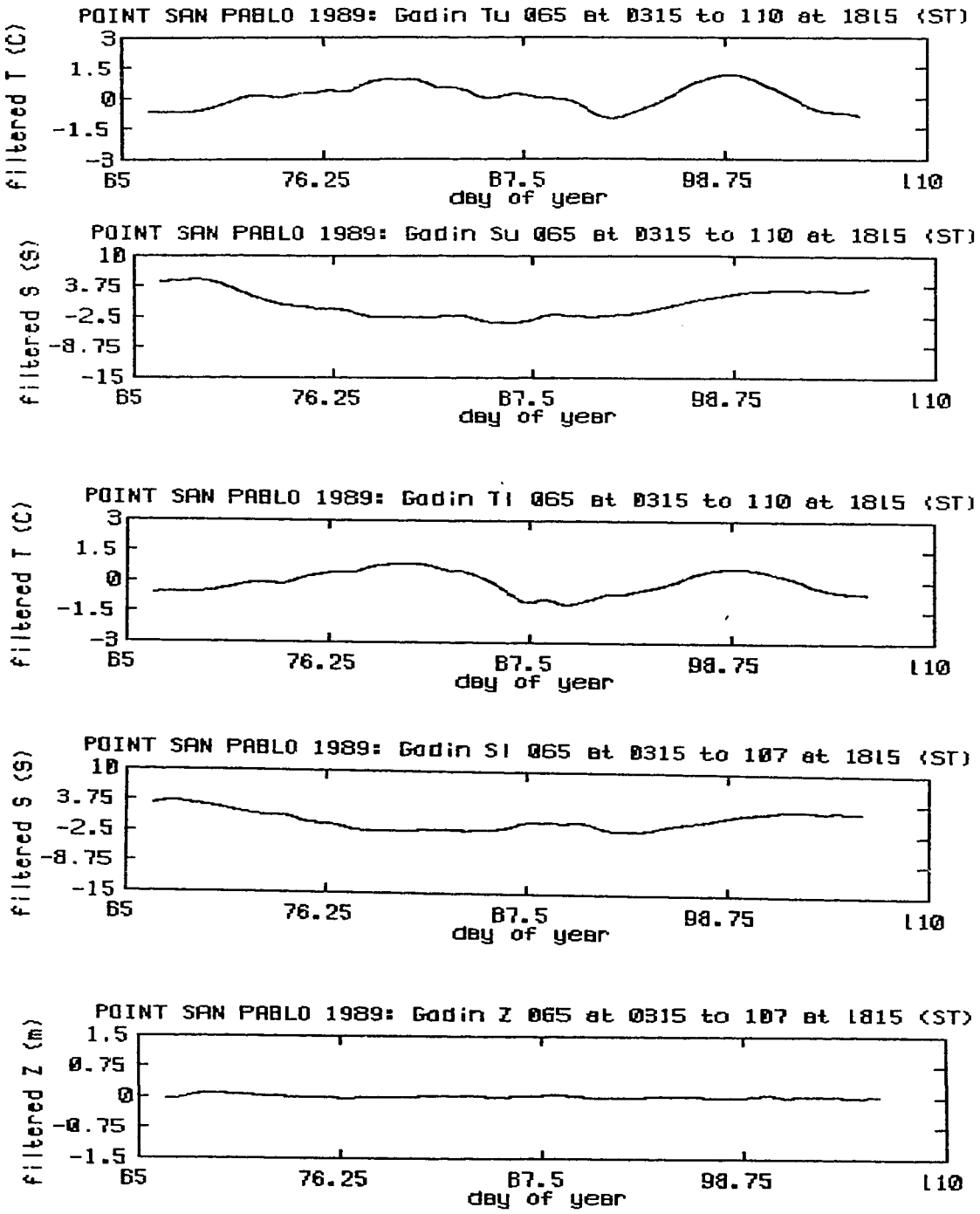


FIGURE 15: Godin filtered data records from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).

CHAPTER 4

PROCESS INVESTIGATION

Introduction

Prior to this point we have examined various time series analysis techniques on an individual basis. Within the investigations a number of features of the data set have been described and discussed, but we hypothesized no causal relationships for these features. This is in-part due to our emphasis on technique rather than result. It is also due to the nature of analysis. Within themselves the analysis techniques do not explain causes for features they discern and describe. Through analysis by a variety of techniques, comparison of results, and comparisons to adjunct data and their analysis, explanations of underlying forces behind variation seen in a data set may be found. Below we briefly reiterate the form of the data set and results of application of each analysis technique individually these data. We then make comparisons of these data, and their analysis results, to meteorological data, delta outflow data and results of analyses of these adjunct data.

Preliminary-Analysis Observations

Introduction

Data presented here (Figure 3) were collected from the Point San Pablo water quality monitoring station (inset Figure 1), between 23 February 1989 at 1715 PST and 25 April 1989 at 1000 PST. This sample period contained a moderate storm event. In preparation for analysis, data records were corrected to remove outlier values, padded to fill record gaps, had their mean removed, were detrended, and low-pass filtered of components with period 2-hours or less (Appendix A). The prepared data set exhibits indications of non-tidal forcing. Tidal excursion appears to be a relatively clean semidiurnal signal. There is significant depression in salinity at low water for the mid-portion of the record and an underlying cycle in temperature with an apparent period of about one-half the prepared record length (Figure 4). Prepared data were subjected to individual spectral analysis, factor analysis, and time-domain filtering procedures.

Spectral Analysis

Autocorrelations of prepared data-records indicate periods near 12 and 25 hours. These correspond, within the resolution of a 1-hour lag-interval, to the main semidiurnal and diurnal tide cycles. Forms of the correlograms for temperature and salinity seem to show effects of non-

tidal long-period forcing when compared to the autocorrelation-function of the tide. Relationship to the tide-signal appears stronger for salinity-signals than for temperature-signals (Figure 5).

Cross-correlograms for tide versus temperature compared to cross-correlograms for tide versus salinity (Figure 6) show strong correlations for both cases. The relationship of tide to temperature is near the inverse of the relationship of tide to salinity. Significance-plots for the data (Figure 9) express more clearly the inverse-relationships for salinity and temperature by the signs of significance-values. Forms of cross-correlograms and corresponding significance-value plots show apparent non-tidal forcing, with a stronger correlation between salinity and tide than between temperature and tide. They also indicate a stronger relation between the lower-level signals and tide, than between the upper-level signals and tide. A diurnal cycle of about 25 hours being coupled with a semidiurnal cycle near 12 hours is indicated for all signals. A phase-lead of 6 to 7 hours is shown for temperature over salinity and tide (figure 8). An inverse relationship giving a 6 to 7 hour phase-lead of temperature over both salinity and tides is reasonable. Salinity should closely track tides. As cooler oceanic waters move in with a flooding tide, temperature will decrease as salinity increases. We are looking at effects of a semidiurnal tide, and see leads of 6 to 7 hours for temperature; rather than the 12 to 13 hours expected for a diurnal tide. There is indication of a 1 hour phase-lead of tide over salinity. Since the lag resolution is 1 hour, this may not be significant.

Periodograms and power-spectrums show indications of tidal energy in each data record (Figure 10). The tide spectrum shows energy only at the semidiurnal and diurnal periods with most of the energy in the semidiurnal component. Both salinity spectra show high energy-peaks matching the position of the energy-peaks for the tide signal. For Sl the diurnal component shows greater energy than the semidiurnal component. Salinity signals show long-period energy-peaks of smaller magnitude than the tidal-energy indications in those signals. For Sl the non-tidal energy-peak is at a period of 512 hours. Su also shows energy at the 512 hour period. However, Su shows more significant energy at a period of about 340 hours, with lesser indications near 200 hours and 257 hours. Relative to long-period energy-indications, tide related energy in the temperature signals is virtually 'swamped out'. The largest indication of energy in both temperature signals is at the 512 hour period. Similar to Su, both Tu and Tl show significant energy at periods near 200, 257, and 340 hours, with the 340 hour period having highest energy next to the 512 hour period. The long-period energy-indications are all higher for Tu than for Tl. These observations confirm the inferences made from looking at autocorrelation-functions and cross-correlograms. Though tidally-driven, temperature and salinity are influenced by long-period forcing. Long-period energy has a greater influence on temperature than salinity. And this energy has a greater influence on the upper-level signals than on the lower-level signals. If we lump semidiurnal and diurnal energy together as tidal-energy, then there appear to be at least three major modes within the data: tidal-forcing, forcing with a period

near 512 hours primarily affecting temperature, and forcing with period near 340 hours record primarily affecting upper-salinity. Though we have yet to present any explanation for it, a cycle with period near 512 hours is observable in plots of Tu and Tl (Figure 4). Energy with period near 340 hours may indicate spring-neap affects. Non-tidal effects seen more strongly in upper level signals may be a result of stratification. These points are discussed in more detail below.

Fourier-transform low-pass-filtering gives further evidence for three primary modes in the data (Figure 11). This tidal-filtering removed essentially all information from the tide-record. Both filtered temperature-records show a cycle with period very near one-half the length of the record, which coincides well with the 512-hour period pointed out above. This cycle appears more significant in the upper-record than the lower-record. Filtered-salinities show a cycle very near the full-length of the records which is larger in upper-salinity. Frequency-analysis techniques will not clearly resolve a period greater than one-half the record-length. Energy for periods less than 512-hours may, in-part, be artifacts of 'aliasing' from this long-cycle.

Factor Analysis

Factor analysis as R-mode principle components analysis with rotation (Appendix C), applied here as a confirmatory technique, also shows three major modes acting to provide variance within the data.

Factor scores plots for the rotated axes (Figure 13) show a mode appearing most related to tides explains 21.1% of variance seen in the data set, a mode appearing most related to the 512 hour period observed in temperature explains 40.9% of variance seen in the data, and a mode appearing most related to the long-cycle in salinity explains 32.4% of variance seen in the data. Correlations of factors with variables (Table 2) show the lower-level parameters to be more closely related to tides than the upper-level parameters, a positive relationship between tide and salinity, and a negative relationship between temperature and salinity.

Time Domain Filtering

Using a Godin filter for tide removal (Appendix D), results (Figure 15) are much the same as from Fourier-transform filtering (Figure 11). Godin filter outputs also indicate 3 major modes in the data. They are 'smoother' than the Fourier-filter outputs, indicating a more severe attenuation of near-tidal periods for the Godin-filter over the Fourier-filter. Although near-tidal periods may be important for some analyses, either filter shows non-tidal periods adequately for our purposes.

Adjunct Data

Introduction

In the preliminary analyses we determined there are at least 3 major forcing-modes producing variation within the data: tides, a mode related to depression in salinity acting over nearly full-length of the 1023 hour record, and a mode related to pulses in temperature with a period near 512 hours. Upper-sensor records seem more affected than lower-sensor records. And, referenced to the tide, temperature seems more affected than salinity, showing significant-influence from 512 hour period forcing. But, we have no causal relationships. The sampling period contained a moderate storm event. It is reasonable to assume this could have had significant affect on station records, and begin looking for related information that correlates with observed non-tidal forcing.

River flow is storm influenced. In terms of outflow from the Sacramento-San Joaquin River Delta (DWR 1990), the sample period storm event was the largest for water-year[4] 1989 (Figure 16). Since this

---

[4] A water-year is the period between October of one calendar-year and September of the next calendar-year. The majority of freshwater input occurs in the winter, which is split between calendar-years. The water-year is used to keep records which have the major amount of freshwater-input all within the same time-period. A water-year is named by the calendar-year of it's last month.

outflow affects salinity levels at the Point San Pablo station it is a primary candidate for a forcing-function. There is some controversy over these values, particularly during the irrigation season. The method for estimating daily-delta-discharge (State of California 1986) involves uncertain assumptions about consumption in the delta and traveltime of water through the delta (Smith 1987). Regardless of their absolute-accuracy, the numbers do show variation in flow due to storms, and that is sufficient for our current purpose. Delta-outflow values are an integration of affects acting over the delta. Salinity can be depressed locally by immediate-runoff. So, local precipitation is another possible forcing-function. Although there may be a significant correlation between precipitation at the station and delta-outflow for some lag-values, the effects of immediate runoff deserve investigation. NOAA weather-stations located at Richmond CA (37° 57' N, 122° 21' W) and at the Martinez CA Water-Plant (38° 01' N, 122° 07' W) were the selected (Figure 17) for daily precipitation data (NOAA 1989). The Richmond weather station is closest to the Point San Pablo water quality monitoring station. The weather station at the Martinez Water-Plant was chosen as a remote comparison point. Both precipitation data sets show storm influence during the sample period, but are slightly different in overall form (Figure 18).

Records of insolation from the sampling station the weather stations would be useful for investigating additions of energy to the water column. However, only daily maximum and minimum ambient-temperature

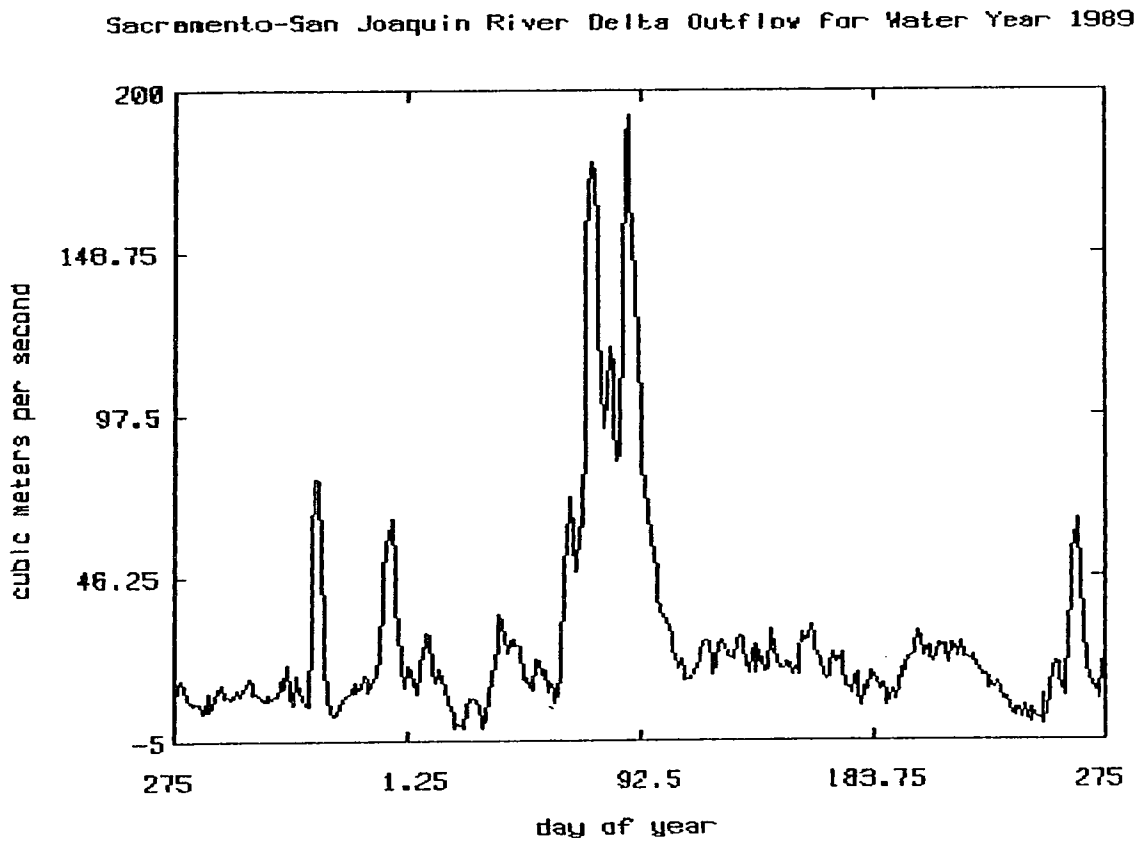


FIGURE 16: Sacramento-San Joaquin river delta outflow for water year 1989.

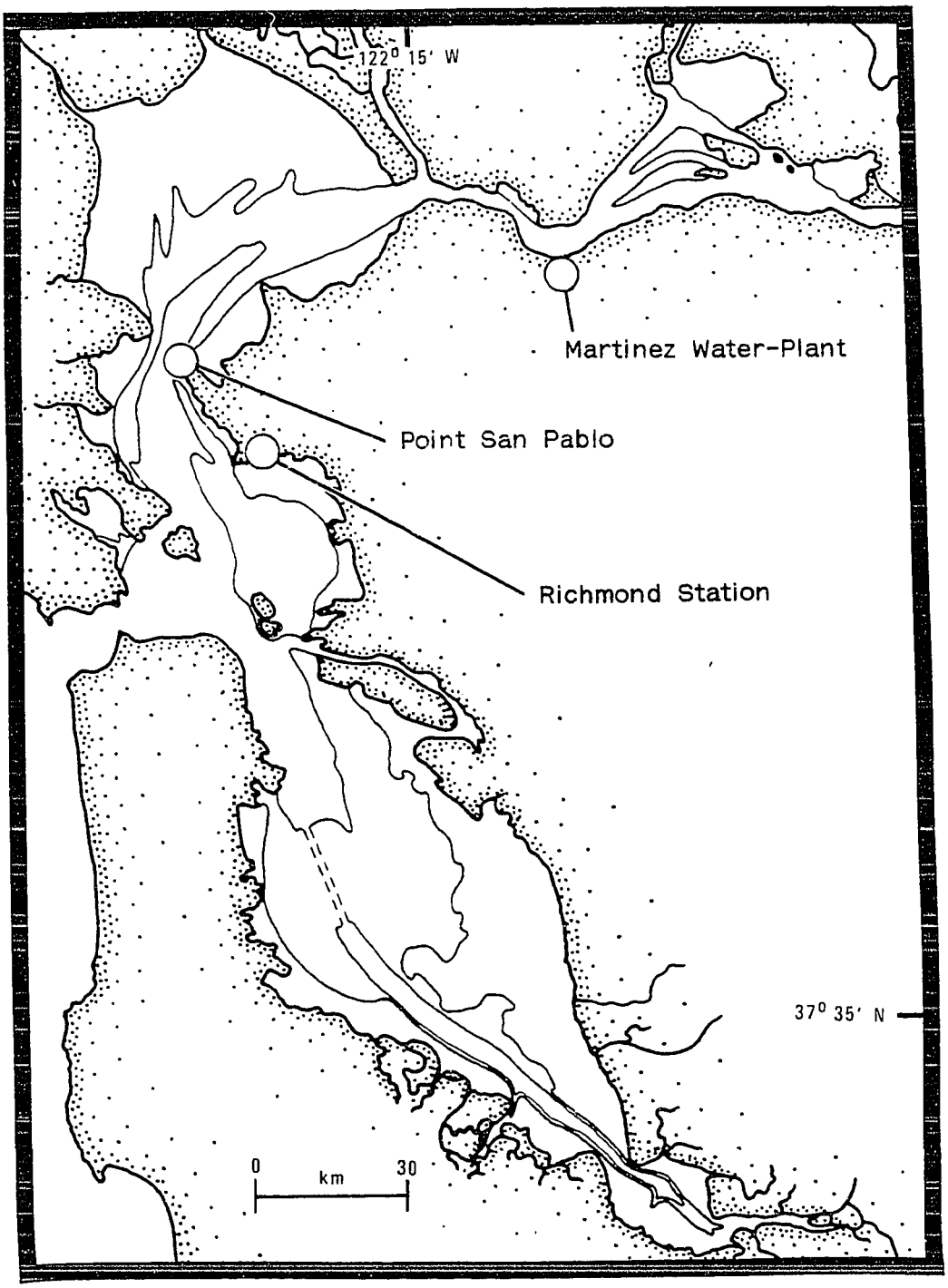


FIGURE 17: Location chart for the Point San Pablo water quality monitoring station and NOAA weather stations at Richmond and the Martinez Water-Plant.

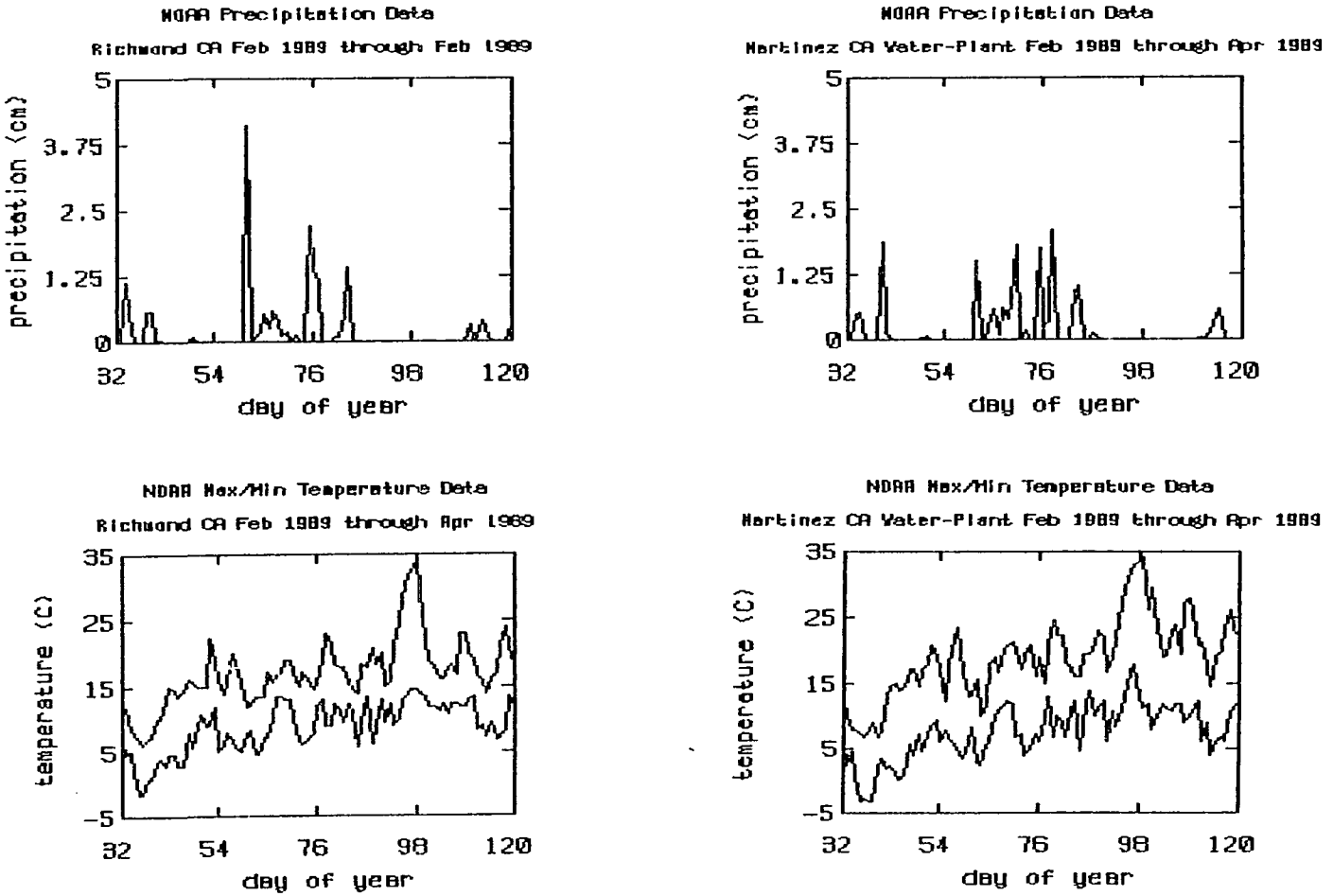


FIGURE 18: Plots of NOAA daily weather data from the Richmond and Martinez Water-Plant stations for the period February 1989 through April 1989.

data from Richmond and the Martinez Water-Plant are available. Ambient air temperature is strongly related to insolation, so this temperature was chosen for comparison as a forcing-function. Temperature data from both stations appear similar in form and magnitude (Figure 18).

Adjunct-Data Analyses and Comparisons

Tidal-forcing within the prepared data set thesis requires little explanation. It is simply expected variation in parameters measured within a tidally-influenced embayment. We are interested in explanations for long-period non-tidal affects. Delta outflow information and meteorological parameter data were obtained as daily averages, so they are diurnally filtered. Hence, here we make comparisons of adjunct-data to the Fourier-transform-filtered parameters from the prepared data set. Effects seen within the filtered data set that correlate with observations of the adjunct parameters can only relate to forcing with periods longer than 24-hours.

An important observation is that within the time span of the prepared data set, major peaks in delta outflow fall within the period of depression in salinity, and, taking into account about 10 day's lead, correspond well to peaks in filtered temperature data (Figure 19). Major peaks in precipitation records (Figure 18), while displaced relative to one another in time by about the same amount as peaks in delta outflow, start their sequence about 3 days before the sample period; leading the

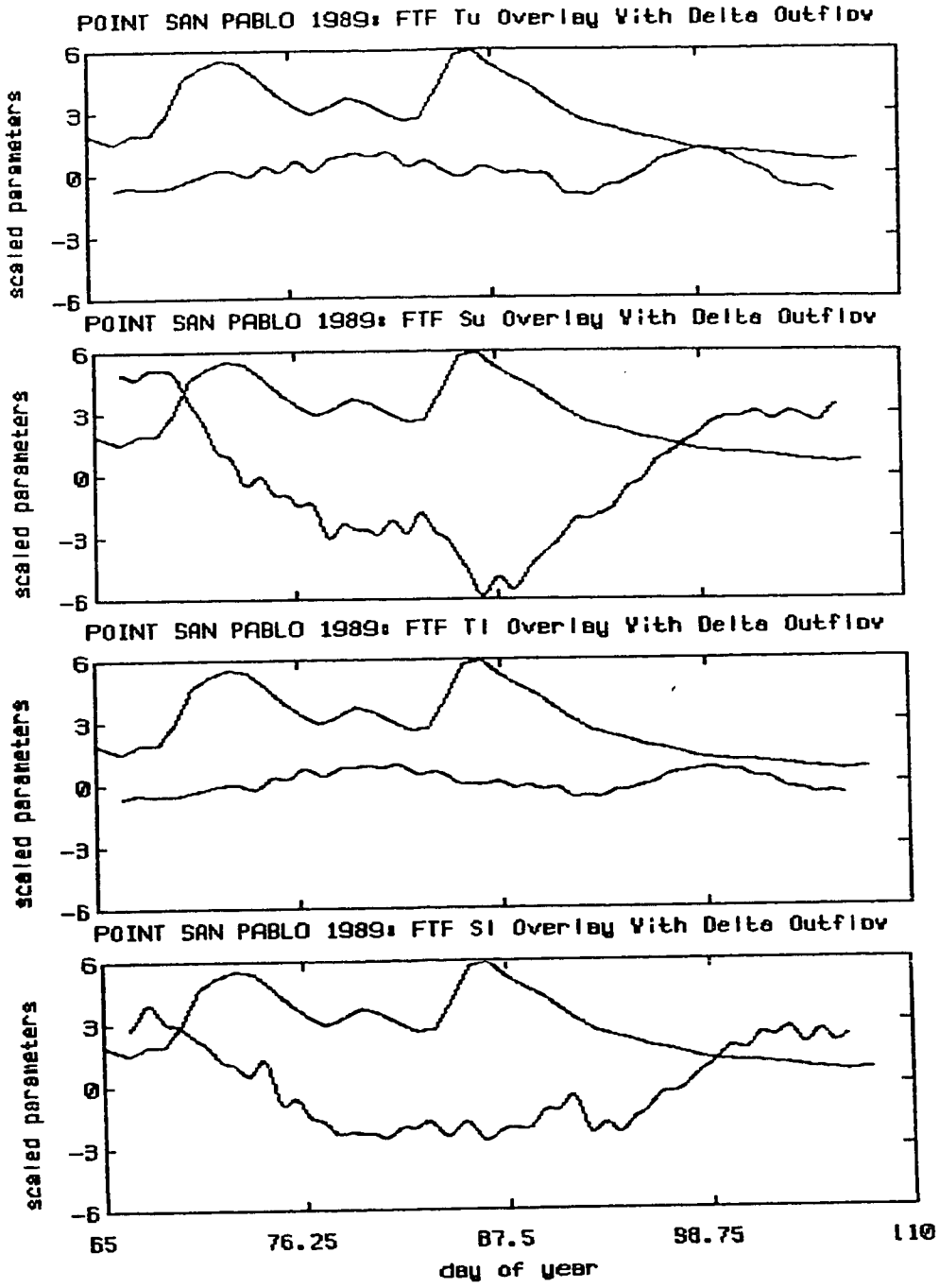


FIGURE 19: Scaled Fourier-Transform filtered data from the Point San Pablo water quality monitoring station, with interpolated and scaled Sacramento-San Joaquin Delta-outflow overlay. 30 points have been removed from each end of the filtered data records to eliminate the filter taper.

outflow peak sequence by about 13 days. The leading edge of the major peak in precipitation starts near day 60. This peak is an obvious reference point. The sample period starts on day 65. To simplify computations and comparisons, the section of each weather station record from day 60 to day 102 was extracted (Figure 20) for use in conjunction with the filtered-variable and delta-outflow records from day 65 to day 107. Where appropriate, as a reminder of the induced five-day offset, time-axes on plots have been identified by record-number rather than day-of-year.

Since the adjunct-parameters have only one value per-day relative to time 1200-hours, they were interpolated to match the filtered-data in number and relative-time within records. The pre-filtering taper of records input to the Fourier-transform filter causes a drop to zero at each end of the output-records. 30 hours were eliminated from the ends of the output-records to remove these drops to zero; leaving them with a span from day 66 to day 106. Corresponding-points were removed from the interpolated adjunct-parameter data. The resulting 964-point (963-hour) variables were used in producing periodograms for the analyses that follow.

Since we anticipate forcing-periods greater than a lag-value of n/4 = 963/4 = 240.75 hours, autocorrelations and spectrums of the variables will show little useful information. However, periodograms will show long-period energy. The program HARMONICS (Appendix F), (which does not

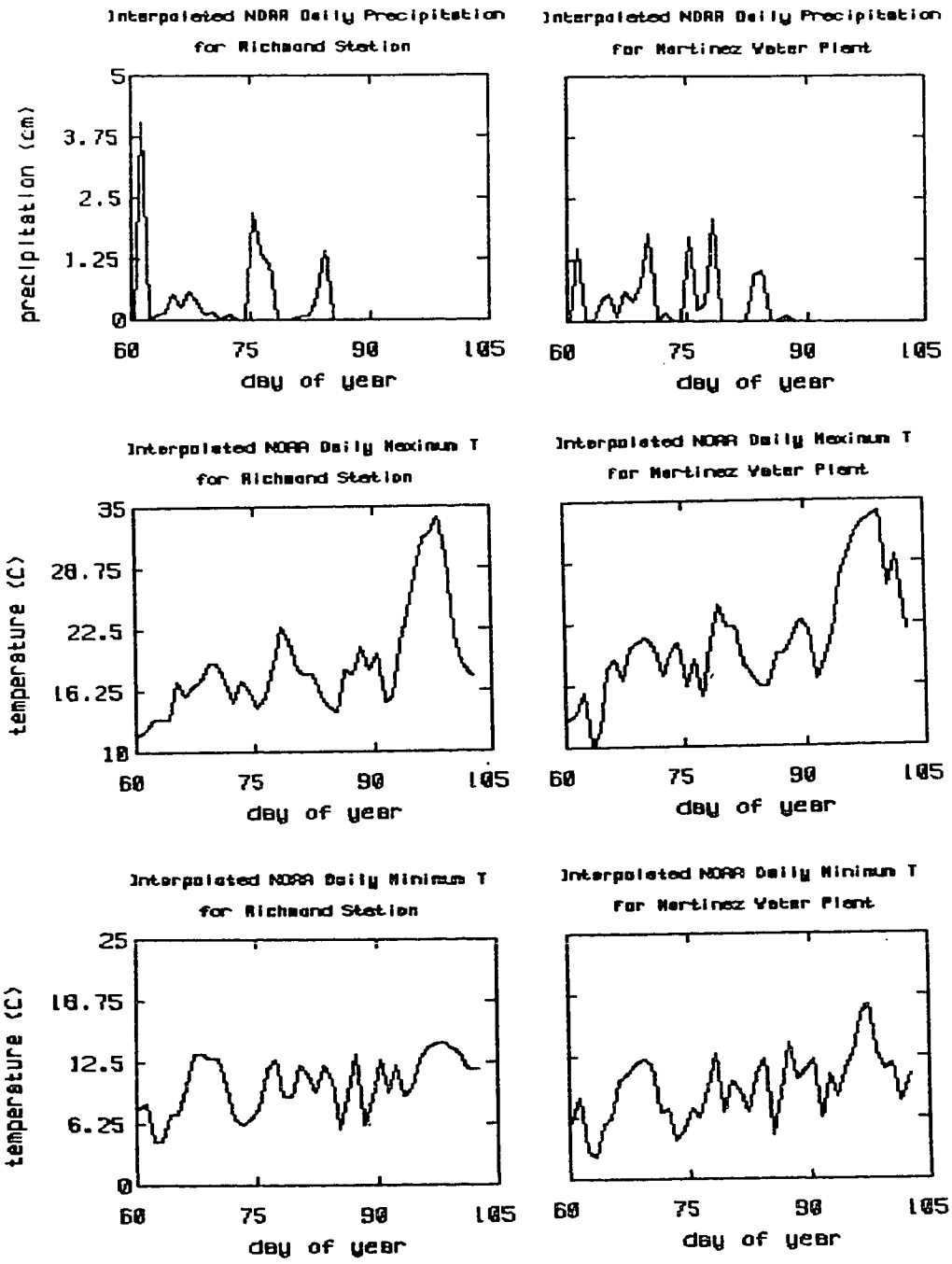


FIGURE 20: One hour interval weather data for the period 01 March 1989 at 0315 to 102 April 1989 at 1815 (ST). These data were interpolated from daily NOAA weather station data values.

require a does not require a power-of-2 number of variables to generate a periodogram), was used to generate the first 150 harmonically-related components for the truncated data. Ignoring fundamentals, these components span periods from 481.5 hours to 6.42 hours. Periodograms from these results show no diurnal or semidiurnal energy-peaks for Fourier-transform-filtered variables (Figure 21), and non-tidal energy-peaks very similar to those from unfiltered-data using the FFT (Figure 10). Also, a small energy-peak can now be seen for tidal-excursion near a period of 321 hours. Within the resolution of calculations this corresponds to the spring-neap cycle. Differences in peak period-values between filtered and unfiltered data are due to the change in resolution caused by using 964-points rather than 1024-points in the calculations. Comparing periodograms for filtered-variables (Figure 21) to the periodogram for delta-outflow (Figure 22), it appears, in terms of energy-content, non-tidal portions of the upper-temperature, lower-temperature, and lower-salinity signals are strongly related to delta-outflow, while the most significant energy-level in the upper-salinity signal is related to forcing with the same period as the small energy-peak discovered in the tide-signal after diurnal-filtering (Figure 21). Periodograms for data from the Richmond weather-station (Figure 22) and the Martinez-Water-Plant weather-station (Figure 22) show temperature signals at both stations to be very similar. The corresponding periodograms for precipitation show significant energy-peaks with a period near 187 hours. The Richmond-station also shows energy-peaks in the precipitation signal with periods near 321 hours and 481 hours not seen

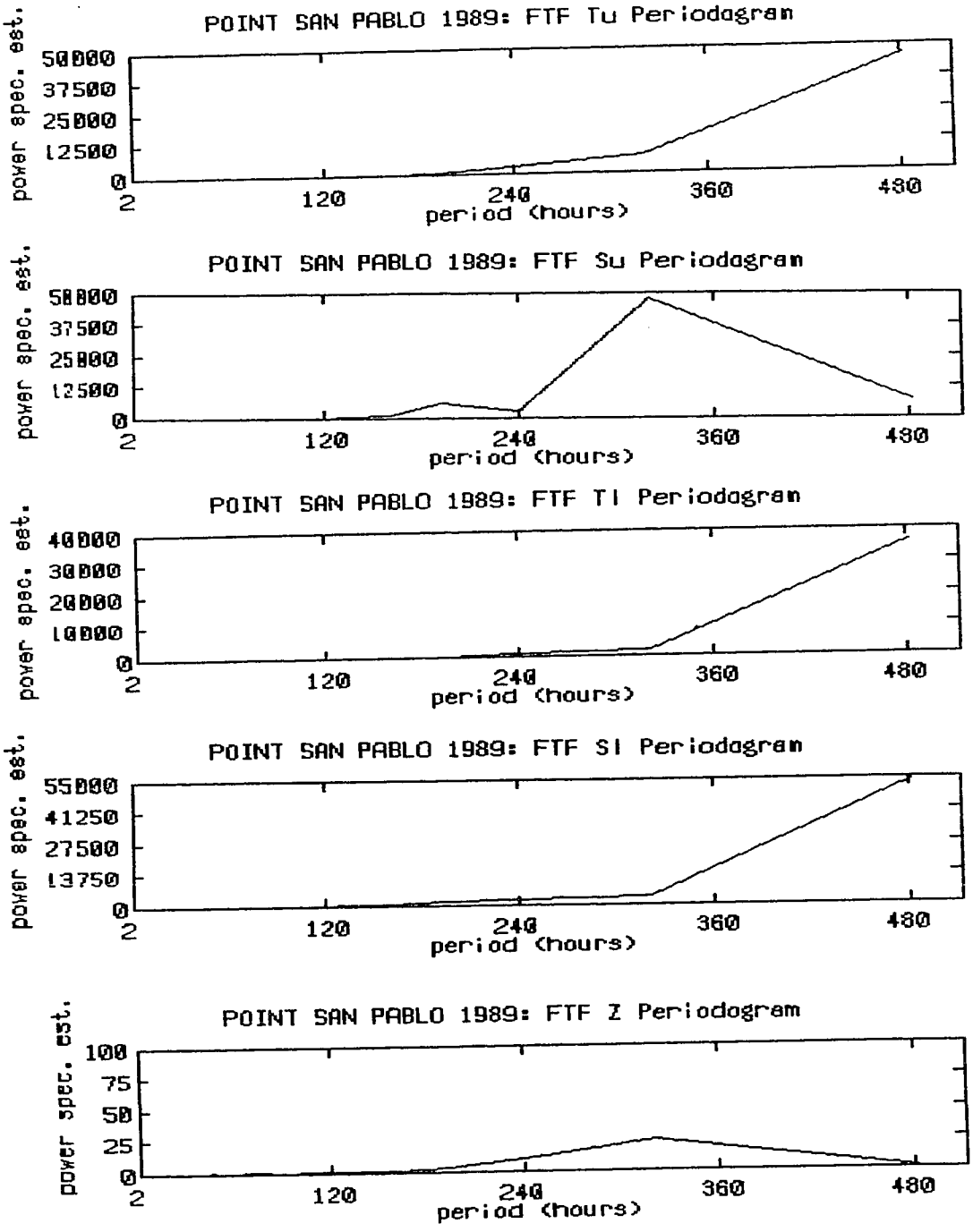


FIGURE 21: Periodograms of Fourier-filtered Point San Pablo water quality monitoring station data spanning 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).

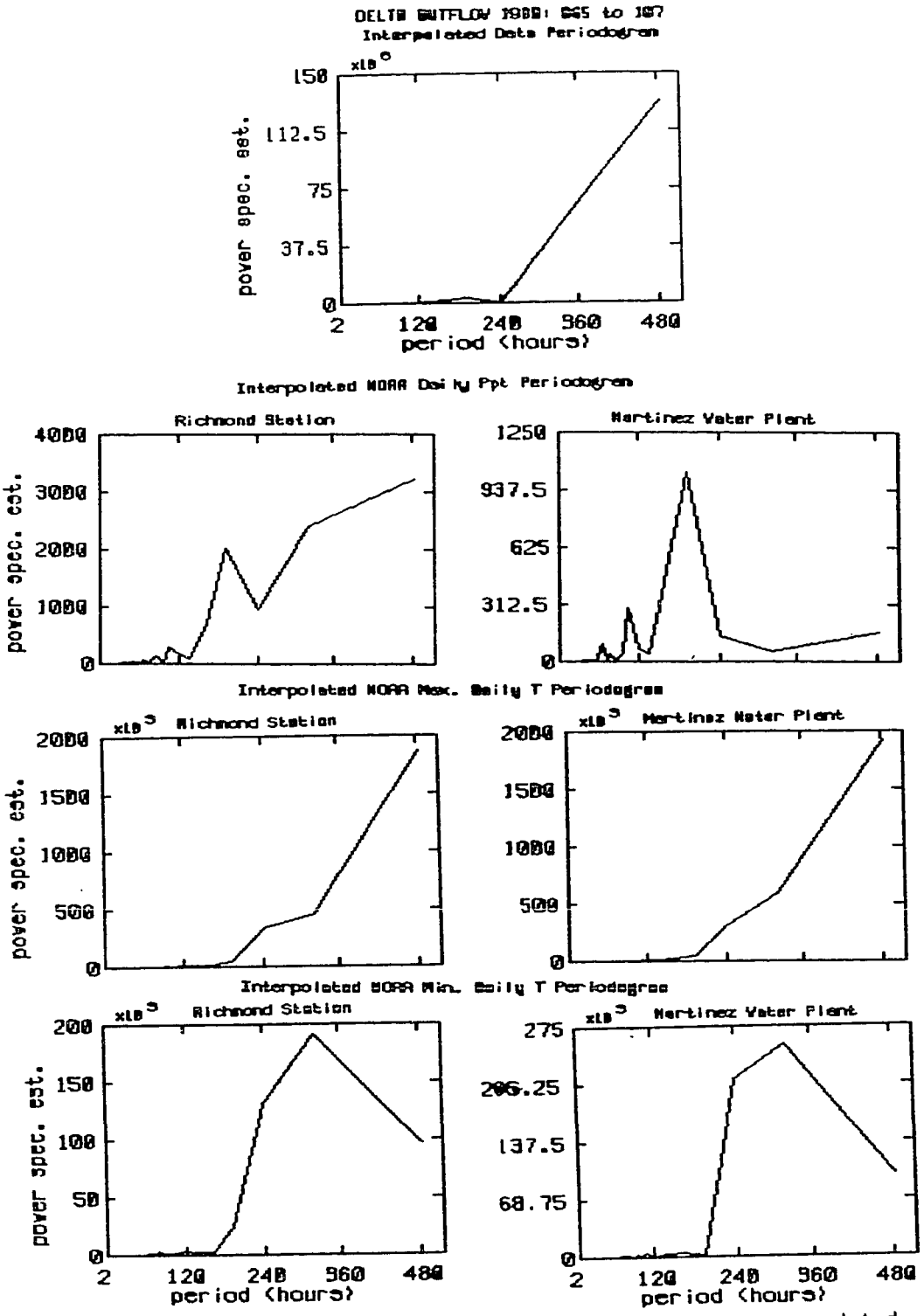
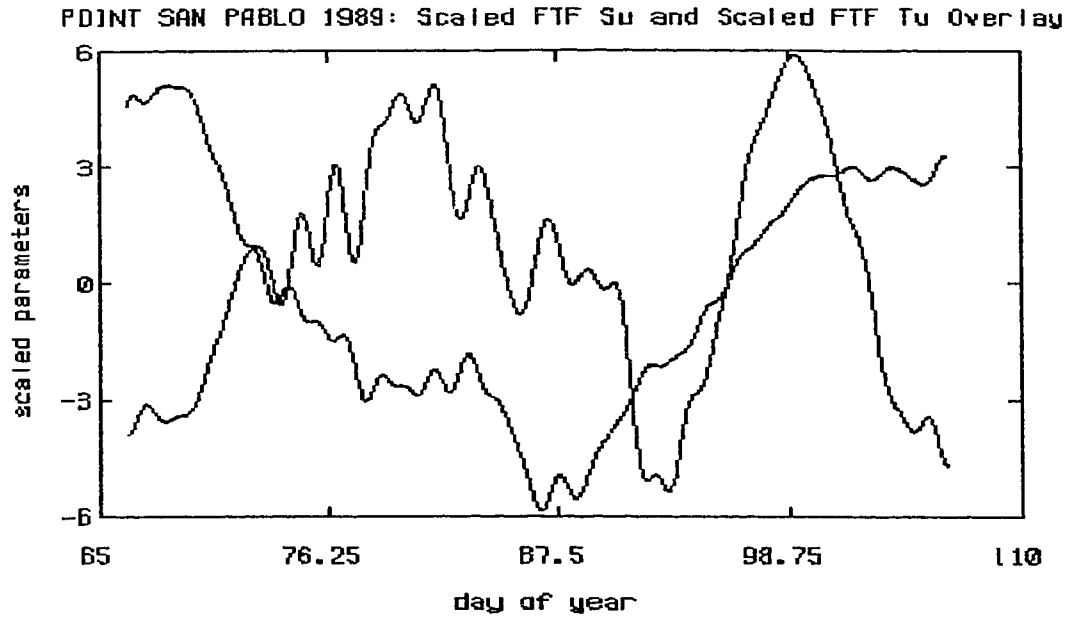


FIGURE 22: Periodograms for interpolated Delta-outflow and interpolated NOAA weather-station data.

for the Martinez Water-Plant station; the 481 hour period exhibiting the greatest energy-level, and the 187 hour period the lowest.

Cross-correlations between filtered-parameters and adjunct-data records will suffer from problems already mentioned for autocorrelations. The forcing-periods under investigation are very close to or greater than the maximum-acceptable n/4 lag-value for match-position. Thus, for these data, cross-correlations cannot be relied on to provide information on similarities in period between records. For relatively-smooth data, simple overlay-plots may sometimes be used to provide much of the information obtained from cross-correlograms. We have already examined lead-versus-lag relationships between filtered variables and adjunct parameters using direct or implied overlays (Figures 18,19); noting in particular that onset of major-peaks in precipitation leads the sample-period by about five days. Comparing filtered variables directly through overlays (Figure 23) there is a near inverse association between salinity and temperature for the early portion of the sample period, and a more direct correspondence between these variables in the latter portion of the record. These relationships hold more strongly for lower-level signals than upper-level signals. Using delta-outflow as a reference-overlay for the adjunct-data records (Figure 24), further relationships between variables can be determined.

Bearing in mind the previously introduced 5-day offset, there appears to be a strong correlation between daily-minimum ambient



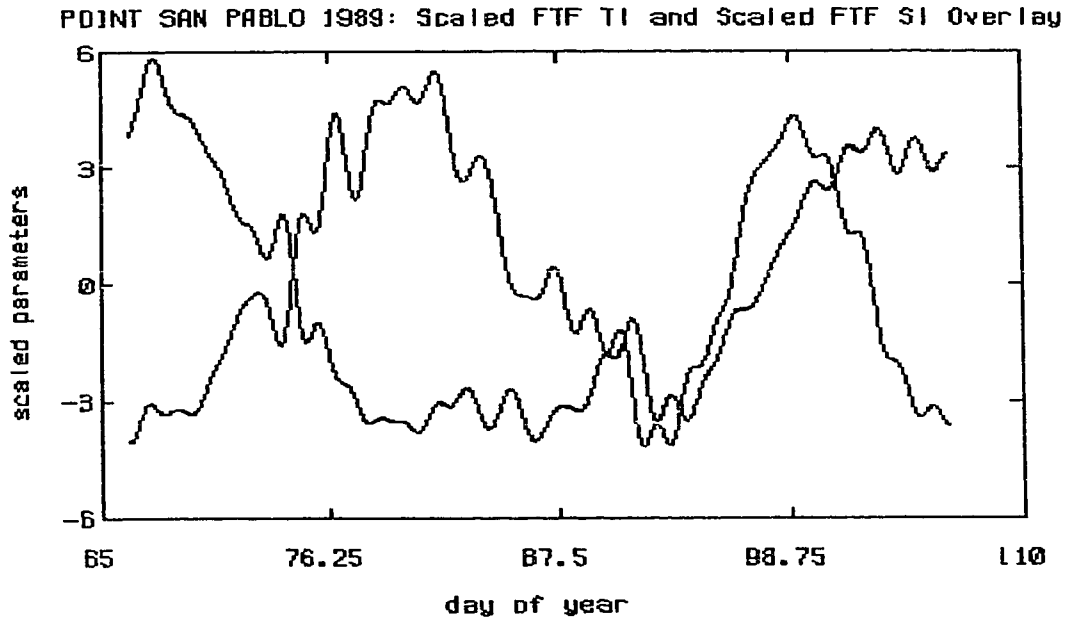


FIGURE 23: Overlays of scaled Fourier-Transform filtered salinity and temperature data from Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST).

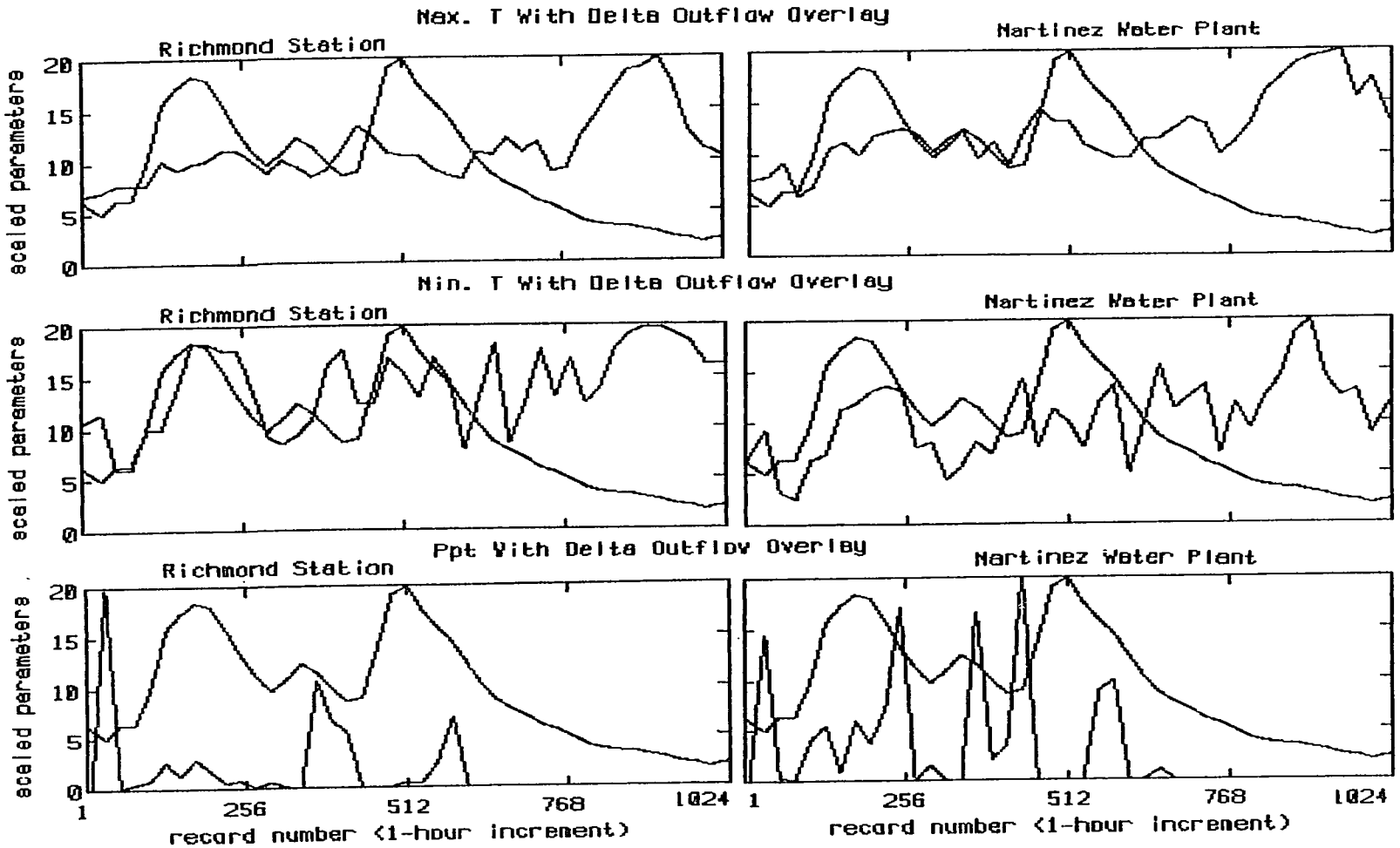


FIGURE 24: Overlays of scaled interpolated NOAA Weather-station data from Richmond Station and Martinez Water Plant with scaled interpolated Sacramento-San Joaquin River Delta outflow data. Outflow data span 06 March 1989 at 0315 to 17 April 1989 at 1815 (ST). There is a 5 day lead of weather station data over outflow data, hence the time axes are identified by record-number rather than day-of-year

temperature and delta outflow. There is a lesser but still discernable relationship between daily-maximum ambient temperature and delta outflow; stronger for Martinez Water-Plant data than Richmond data. Though there is some difference in overall form of the records between stations, there is good correspondence between major and minor peaks in precipitation at Richmond and Martinez Water-Plant. With an additional 158 hours lead time there is very good correspondence between the major and minor peaks in precipitation at Richmond and in delta outflow. Further comparisons show an overall increase in ambient temperature between peaks in precipitation (Figure 18). With the 278 hour lead seen for precipitation over delta-outflow applied to these temperature-peaks, they match the peaks observed in the filtered-temperature records, which occur between and after the peaks in delta-outflow (Figure 19). These may provide some explanation for the observed cycle in the temperature signal. However, there are large peaks in daily-maximum ambient temperature that, along with peaks in daily-minimum ambient temperature, correspond well with peaks in the station water-records; leading by about 2.5 days.

In real-time, there is little correspondence between local precipitation, ambient air temperature, and the form of station records after Fourier-transform low-pass filtering to remove tidal-components. Long-period non-tidal forcing in temperature-signals corresponds most strongly with peaks in ambient air-temperature records, which lead the temperature-signal peaks by about 2.5 days. Long-period non-tidal

forcing observed in salinity-records seems most strongly related to storm-affects beginning on about day 060 and ending on about day 085. These affects are strongly-coupled to station signals through changes in delta-outflow, which follow peaks in precipitation by about 278 hours, or 11.6 days. However, the couplings are not strictly flow related.

Delta outflow shows peaks corresponding to lagged-peaks in precipitation. Salinity signals show only a depression starting near the time of the first peak in delta outflow, and not recovering until after the last peak in outflow. The maximum flow rate during the sample-period was near 193 m^3/s. Delta-outflows near 200 m^3/s are considered low-discharge values (Smith 1987). For low-discharge conditions, estimates of mean residence-times are on the order of 20 days in San Pablo Bay (Denton and Hunt 1986). Since the major-peaks in delta outflow are 14 days apart, with a smaller peak centered between the two major-peaks, recovery of the system should not have, and did not occur between the periods of increased delta-outflow. There was a strong decrease in Su corresponding to the second peak in delta-outflow, that was not observed in Sl. The peaks in delta-outflow both occurred near the 'neap' portions of the spring-neap cycle. The second outflow-peak occurred at the time of a weaker set of neap-tides than the first peak. At low-flow the estuary tends to stratify. The decrease in mixing due to weaker neap-tides, within a system already partially-stratified, could result in greater effects in Su than Sl for increased freshwater-input with the second peak in delta-outflow. This would explain the difference in

periodograms and plots for upper and lower salinity-signals. The major energy-peak seen for Su occurs at a period near 340 hours. This corresponds well to the approximately 14-day spring-neap cycle, and to the spacing of the major-peaks in delta-outflow. The 340-hour period energy-peak is not seen in the periodogram for Sl.

Peaks in temperature signals come after peaks in delta outflow and seem to correspond best, with a lag of 2.5-days, to increases in daily maximum and minimum ambient air-temperature. These peaks in ambient temperature follow precipitation-peaks by about 17.3-days. Assuming average ambient air temperature is related to insolation, a simultaneous increase in minimum and maximum ambient-temperature means increased potential for overall heat-transfer to the bay through radiative heating. San Pablo Bay has large shoal-areas. Transfer of heat-energy into broad shallow areas is more efficient than transfer into narrow deep areas. Such heating can be significant in shoal areas, and effects in temperature-signals from the bay are possible. As was the case for salinity and delta-outflow, the lagged-affect of the second peak in daily ambient temperature records appears greater for Tu than for Tl. The second set of peaks in ambient-temperature records are greater in magnitude that the first set, which reasonably explains the temperature signal peak differences. We might also relate part of the difference in temperature signal peaks to the spring-neap cycle. The peaks in ambient temperature records occurred close to the 'spring' portions of the spring-neap cycle. The second set of peaks in ambient-temperature records

occurred near stronger spring-tides than the first set. This means shallower water and increased heating in shoal-areas at low-tide. The same stratification arguments made for salinity hold for greater effects in Tu than Tl.

Process Investigation Summary

Examinations of data from the Point San Pablo water quality monitoring station from 1989, day 065 at 0315 PST to day 107 at 1815 PST showed the tidal-influence expected for these values, and evidence of long-period non-tidal forcing. A depression in salinity, particularly at low-water, occurred just after the beginning of the sample-period, and only made a partial-recovery by the end of the record. A long-period cycle was observed in temperature. And apparent long-period affects seemed to have a greater effect on the upper-sensor records than the lower-sensor records.

By several means we determined there should be three modes causing the observed-variation in these data: tides, and two long-period non-tidal modes related to forcing in salinity, and forcing in temperature. Further investigations indicated non-tidal modes were related to affects due to storms shortly before and during the early-portion of the sampling-period, and correlated with changes in ambient air-temperature after the storms. Forcing in salinity seemed to relate to delta-outflow, which correlated strongly to precipitation with a lag of

278 hours. Forcing in temperature seemed most strongly related to simultaneous increases in maximum and minimum ambient air-temperature occurring approximately 415 hours after peaks in precipitation, and leading the peaks in temperature-signals by about 60 hours.

Although delta-outflow had two peaks, salinity-signals exhibited one depression and recovery. The literature (Denton and Hunt 1986) indicates a 20 day or greater mean residence time for San Pablo Bay at the observed delta-outflow levels. The major peaks in delta outflow were near 14 days apart, separated by a smaller peak. The system should not have recovered between the major-peaks in delta-outflow. Peaks in temperature corresponded, with a 60-hour lag, to simultaneous peaks in daily maximum and minimum ambient air temperature. The mechanism is likely radiative-heating of water in shoal areas due to increased insolation, rather than atmospheric heat conduction, but no insolation data are available.

The larger effects seen in latter-portions of signal-records may relate to the spring-neap cycle. This may have enhanced affects of delta-outflow on salinity, and affects correlated with ambient-temperature on water-temperature. The storm causing increased delta outflow was the largest of water-year 1989, but flow-levels are considered low. Larger effects of long-period forcing observed in upper-signal records over the corresponding lower-signal records may result from stratification at low delta-outflow levels constraining more of the affects to the upper-signals.

The above hypotheses are all only educated-inferences based on examination of limited-data. They are consistent with known processes. However, it would require a larger-scale study under similar conditions for reasonable verification.

CHAPTER 5

SUMMARY

Review of Techniques

Autocorrelation and Cross-Correlation

Autocorrelation gives information on periodicities in a record. This information can provide insight to underlying structure of a variable. It may be compared to periods of known processes as an aid in determining forcing-functions. For signals with a strong underlying short-period cycle, such as a tidally driven parameters, correlograms will show, throughout their length, indications of periodicity at lags which are multiples of the short-period cycle. This is a problem in general, but may be used to advantage in special cases. When the source of a short period cycle is know, variation away from the expected incremental indications of periodicity in a correlogram may give clues to the nature of other underlying forcing. This may be seen in correlograms for the prepared data (Figure 5). The expected form of short-period cycling is shown in the correlogram for the tide signal. Deviations from this form in correlograms for the other variables indicate long-period forcing which has a greater affect on temperature than salinity. A record of length n with sample-interval δ can have a period no greater than $n\delta/4$ reliably determined through autocorrelation. Thus, if the length of a record is

less than four-times the length of one of its underlying-periods, that period will not be seen in a correlogram.

Comparison of two variables through cross-correlation will show, in cross-correlograms, common periodicities and phase-differences between the variables. Cross-correlograms are asymmetric, their form depending on which variable is fixed and which variable is lagged. This can make interpretation difficult and doubles the computation requirement if both cross-correlograms must be generated. Cross-correlation also suffers from the same problems as autocorrelation. Strong short-period cycling in the signals may mask other information in cross-correlograms. To be reliable they are constrained to determining periods of no more than n/4 time-steps. This problem was seen in the PROCESS INVESTIGATION section of this thesis, where for filtered-variables, we had to resort to comparisons of data-plots to determine period and phase-relationships.

Harmonic Analysis

Producing power-spectra and periodograms for a variable gives information on periods contained within that variable, and the relative energy-levels of components corresponding to those periods. There is no problem with swamping by short-period information in a spectrum or periodogram plot. However, by the nature of inverses, power-spectra provide information most clearly for short-period components, and periodograms for long-period components. By expressing not only the

frequencies contained in a variable, but the relative energy-levels at those frequencies, harmonic-analysis allows detailed comparisons of signals and their interrelationships. Two signals that look much the same in their time-domain plots may be classified as different though harmonic-analysis. Here, upper and lower salinity-signal plots appeared quite similar (Figure 4). Their respective periodograms showed a similar frequency-structure, but different energy-content for the frequencies (Figure 10). Further investigations revealed that some aspects of forcing in salinity were, in fact, different for the upper and lower signals during the sampling-period.

Two different methods of producing spectra were used for this thesis. Spectrums generated from the Fast-Fourier-Transform (FFT) were presented in the main spectral-analysis section, and direct computation of Fourier-coefficients was employed in the PROCESS INVESTIGATION section. The techniques are equivalent, but the FFT requires a power-of-two number of values in its input record; while computationally much less efficient, the direct calculation technique may use any length record. After filtering to remove tidal-influence the variables were reduced from 1024 values to 964 values. Hence the switch in technique for computing spectrums. Power-spectrums (and periodograms) are subject to noise and frequency-resolution problems. Much of the noise is due to aliasing. Resolution in period or frequency is determined by the length of the record being analyzed. The fundamental-period is taken to be the length of the record, and higher-frequency components are harmonically related

to the fundamental, so that shorter periods come only at increments of one-half, one-third, one-quarter, etc. of the fundamental. Using two slightly different lenght records means even though the 'same' data are analyzed, slightly different periods for the components will be determined, with the greatest differences at the longest periods. In our case, frequency-structure and energy-content were such that the slight differences in indicated periods, (being approximate in any case), had no affect on the analysis.

Fourier-Transform Filtering

Fourier-transform filtering is a versatile method, employed here as a low-pass tidal-filter. After split-cosine tapering 3% of each end of the time-domain input records (Figure 4), a frequency-domain stop-period of 30 hours with a pass-period of 40 hours and split-cosine taper between was sufficient to remove virtually all traces of tidal-variation in the output-records (Figure 11); this was confirmed by the missing diurnal and semidiurnal peaks in their corresponding periodograms (Figure 21). This filter requires transformations between time and frequency domains, and computations in both, but, since the Fast-Fourier Transform is used, it is computationally efficient. Tapering of input-records causes ends of the output-records to drop to zero. Thirty hours (3%) was removed from each end of each output to eliminate these drops. This left 964-point filtered records for analysis and comparison to adjunct-data such as delta-outflow (Figure 19).

Removal of tidal-variation showed aspects of the data we inferred from looking at unfiltered-data and the previous analyses. The nearly pure semidiurnal nature of the tides, pulsing with a period near 512 hours in temperature, and a depression in salinity starting shortly after the beginning of the record and lasting nearly the length of the record can be seen in the filter outputs, as well as a difference in forcing-effects between the upper and lower signals.

Tidal filtering allows an examination of structure of non-tidal residuals with detail that is not available from autocorrelations, spectrums, and periodograms. The full sequence of variation is available, rather than just information on energy-levels and frequencies. In some cases, there may be enough information in the form of the residuals to identify forcing-functions. More often, as here, residuals must be compared to adjunct-data and further analyzed in order to identify sources of non-tidal forcing. As such, although very useful, Fourier-transform filtering could be considered an analysis-step, rather than an analysis-technique.

Factor Analysis

We inferred at least 3 modes for the data of this thesis from other analyses, and factor-analysis also showed 3 major-modes operating within these data: tidal-forcing, forcing related to depression in salinity, and forcing related to pulsing in temperature (Figure 13). However, except

for specific information on the statistical levels of significance for these modes, no more information was gained performing the factor analysis procedure than is available from a careful visual examination of plots of the prepared-variables (Figure 4). The apparent forcing determined from factor analysis is observable in the form of the data. That is an important point with regard to factor analysis. Factor analysis in no way provides any information about causal relationships within an investigation. It only provides, based on observed levels of variance, a set of variables that will explain the form of the original data. The form of these underlying variables may or may not relate to the form of any acting 'real' process. In fact, the apparent-mode determined for salinity did not have the same form as the most likely real-source of variation in salinity. Delta outflow showed two peaks and salinity only one depression. Factor analysis could not take into account the mean residence-time factor at low delta-outflow levels. Factor analysis could only detect the overall depression in salinity; a result of integrating affects of residence-time on delta outflow. The form of the result is correct for the form of salinity in station records, but it does not explain the more complicated underlying processes. Factor analysis is a sophisticated filtering procedure. Interpretations are required for modes observed in filter outputs, regardless of filter type.

The expected form of time-series plots for tidally-driven parameters is very familiar to anyone who has spent much time investigating these kind of data. If something is significantly 'not normal,' a process such

as factor analysis is not required to see its effects. For any data set where the major underlying forcing-functions can be reasonably predicted by other means, factor analysis will provide little if any additional information. Long-term time-series of tidally-driven parameters fall into this category. If the influence of predictable-forcing is removed from such records by, for example, Fourier-transform filtering, then factor analysis may help provide insights to the form of the remaining modes within the filtered-data. Here again we only obtain information as to possible forms for these modes, and no causal explanations. Factor analysis processing of filtered records was not demonstrated here. The forms of the filtered records showed acceptable forms for the underlying modes without further attempts at mode separation. In the case where acceptable forms for factors influencing variation in data are observable without 'factor analysis' then factor analysis is unlikely to produce any new information.

Time-Domain Filtering

Time-domain filtering (convolution) involves forming weighted averages of segments of input records. Time-domain tidal-filtering, (e.g., the Godin filter), uses multiple convolutions of 24 hour and 25 hour segments of the input data. As with Fourier-transform filtering, time-domain filtering allows an examination of details of the forms of non-tidal residuals. However, compared to Fourier-transform filtering it is computationally expensive, and inflexible.

The Godin filter gave results (Figure 15) similar to Fourier-transform filtering (Figure 11). They are somewhat smoother than the Fourier-transform results, indicating a stronger attenuation of near-tidal periods than is exhibited by the Fourier-filter. This additional frequency attenuation may or may not be desirable, depending on the analysis. In our case the additional smoothing would not have affected our analysis results. But, an aspect of the data which stands out clearly in the Fourier-transform filter outputs and was used in describing non-tidal forcing, the depression in upper-salinity with the second peak in delta-outflow, is less easy to see in the output of the Godin filter.

Final Comments

For investigations on a 'climatological scale', looking at perhaps a year of data rather than a few weeks, direct factor analysis of records of tidally-driven parameters may be able to separate tidal and non-tidal variance to the degree where their difference is readily observable in the form of factor-scores plots. As we saw here, tidal variation in short records will bleed into non-tidal modes determined from factor-analysis. Tidal-filtering of data before factor analysis will allow separation of strictly non-tidal forcing-modes from short records through factor analysis. However, unless the underlying modes are very subtle, the results of prefiltering may give as much information about the form of forcing as will be gained from factor analysis of the filtered information. For records such as those examined in this thesis, factor analysis is

probably most useful in confirmation of analysis results, rather than as an analysis tool. However, the amount of computations and memory required to recover factor scores for even a small data set limit its utility in analyses of the types presented here.

Tidal-filtering of data by any technique (time-domain filtering, factor analysis, or Fourier-transform filtering) is more an analysis-step than an analysis-technique. Removal of tidal-components may by itself provide, in the forms of the residuals, enough information to identify long-period non-tidal forcing. More likely, further analysis of residuals will be required to identify sources of non-tidal affects. Time-domain filtering and factor analysis are computationally more expensive and not as versatile as Fourier-transform-filtering.

Correlation techniques were among the more useful analysis methods investigated here. They require fewer computations and less memory than factor analysis. They are similar in computing requirements to harmonic analysis, and provide much of the same information that is available from periodograms and spectrums. Cross-correlation is, other than direct comparison of plots of data records, the only method used here that provided information about phase-relationships between variables.

If only one analysis technique was to be employed on variables from a tidally-influenced data-set, formation of power-spectra and periodograms would probably be most useful. These will separate diurnal

and semidiurnal-components from long-period components. Through an examination of frequencies and relative energy-levels of the non-tidal periods, they will also provide clues to the structure of non-tidal forcing and relationships of variables to that forcing. Comparing these results to the form of the original variables may provide enough information to identify non-tidal forcing.

In general more than one analysis-technique is required to specify forcing for complex processes such as tidal and non-tidal interactions. Depending on the form of the data, some techniques it might be desirable to apply will not work, and others will need to be substituted. For example, based on the length of records and the expected non-tidal periods determined from harmonic-analysis, cross-correlations could not be applied to the filtered and adjunct-data in the process investigation section of this thesis. In fact, to determine phase relationships between these records, none of the analysis techniques investigated here could be used. Cross-correlations would have shown phase differences between signals. However, this would only be after the signals were sufficiently prealigned and making the proper choice for which variable to lag. Cross-correlation could only have found phase relationships which we preordained. Still, making hypotheses for sources of non-tidal forcing in the data set would have been impossible without phase data. We had to resort to direct examination of data overlay plots to obtain this information.

Looking at plots of data is a viable and important data analysis technique. It should be the first step in any analysis of an oceanographic data-set, whether the observations are from simple two-dimensional figures, or multicolor three-dimensional displays. Regardless of available computing-equipment and analysis-techniques, direct examination of data-records can be invaluable, and should never be overlooked in deference to calculations.

REFERENCES

Bloomfield, P. 1976. *Fourier analysis of time series: an introduction.* John Wiley and Sons, New York.

Burrus, C.S. and T.W. Parks 1985. *DFT/FFT and convolution algorithms.* John Wiley and Sons, New York.

Cadzow, J.A. 1973. *Discrete time systems: an introduction with interdisciplinary applications.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Cheng, R.T and J.W. Gartner 1984. *Tides, tidal and residual currents in San Francisco bay California - results of measurements, 1979 - 1980, part 1, Description of data.* United States Geological Survey, water resources investigations report 84-4339.

Denton, R.A. and J.R. Hunt 1986. *Currents in San Francisco bay -- final report.* University of California at Berkeley, Hydraulic and Coastal Engineering Report UCB/HEL-86/01.

Davis, J.C. 1973. *Statistics and data analysis in geology, 2nd ed.* John Wiley and Sons, New York.

DWR 1990. Delta-Outflow (DAYFLOW) data for water-year 1989 obtained on request from the California State Department of Water Resources, Central District, 3251 S Street, Sacramento California 95816-7017

Evans, J.C. 1985. *Selection of a numerical filtering method: convolution or transform windowing?* J. of Geophys. Res. 90(C3):4991-4994

Fofonoff, N.P. and R.C. Millard Jr. 1983. *Algorithms for computation of fundamental properties of seawater.* UNESCO technical papers in marine science, 44.

Godin, G. 1972. *Analysis of tides.* University of Toronto Press, Toronto.

Guertin, W.H. and J.P. Bailey 1970. *Introduction to modern factor analysis.* Edwards Brothers, Ann Arbor, Mich.

Hamming, R.W. 1983. *Digital Filters.* Prentice-Hall, Inc., Englewood-Cliffs, New Jersey.

Harman, H.H. 1976. *Modern factor analysis, 3rd ed.* University of Chicago Press.

IEEE 1980. *Special issue on the practical salinity scale 1978.* IEEE J. Ocn. Eng., OE-5(1).

Jenkins, F.A. and H.E. White 1976. *Fundamentals of optics, 4th ed.* McGraw-Hill Book Company, New York.

Jennrich, R.R. and P.F. Sampson 1966. *Rotation for simple loadings.* Psychometrika, 31(3):313-323

Joreskog, K.G., J.E. Klovan and R.A. Reyment 1976. *Geological factor analysis.* Elsevier scientific publishing company, New York.

Kaiser, H.F. 1959. *Computer program for varimax rotation in factor analysis.* Ed. and Psych. Meas. 19(3):413-421

Kim, J. and C.W. Mueller 1978a. *Introduction to factor analysis: what it is and how to do it.* Sage University Papers.

Kim, J. and C.W. Mueller 1987b. *Factor analysis: statistical methods and practical issues.* Sage University Papers.

Kirby, W. 1974. *Algebraic boundedness of sample statistics.* Water Resources Research. 10(2):220-222

Kulhanek, O. 1976. *Introduction to digital filtering in geophysics.* Elsevier scientific publishing company, New York.

Marple Jr., S.L., 1987. *Digital spectral analysis with applications.* Prentice Hall Inc. Englewood Cliffs, New Jersey.

NOAA 1989. *Climatological data - California.* National Oceanic and Atmospheric Administration, National Climatic Data Center (NCDC), Ashville, North Carolina.

Press, W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling 1987. *Numerical recipes: the art of scientific computing.* Cambridge University Press.

Rayner, J.N. 1971. *An introduction to spectral analysis.* Pion Limited, London.

Schureman, P. 1940. *Manual of harmonic analysis and prediction of tides* (reprinted with correction 1976). United States Coast and Geodetic Survey, special publication 228.

Smith, L.H. 1987. *A review of circulation and mixing studies of San Francisco bay California.* United States Geological Survey circular 1015.

State of California 1986. *Dayflow program documentation and data summary user's guide.* California Department of Water Resources, Central District, Sacramento, California.

Spiegel, M.R. 1974. *Schaum's outline of theory and problems of Fourier analysis.* McGraw-Hill Book Company, New York.

Swithenbank, A.M. 1990. *A microcomputer for instrument control and data handling at water quality monitoring sites in San Francisco Bay.* United States Geological Survey, open file report, unpublished.

Thompson, R.O.R.Y. 1983. *Low-Pass filters to suppress inertial and tidal frequencies.* J. of Phys. Ocn., 13:1077-1083

Walters, R.A. 1982. *Low-frequency variations in sea level and currents in south San Francisco bay.* J. of Phys. Ocn., 12(7):658-668.

Walters, R.A. and C. Heston 1982. *Removing tidal-period variations from time-series data using low-pass digital filters.* J. of Phys. Ocn., 12(1):112-115

Yuen, K.C. and D. Fraser 1979. *Digital spectral analysis.* CSIRO, Australia. Pitman Publishing Limited, London. Fearon Pitman Publishers Inc., Belmont, California.

APPENDIX A

DATA COLLECTION AND CORRECTION

Introduction

Any data collection and analysis process must include several steps. Prior to data collection, data-requirements and potential results must be considered in order to formulate a sampling plan. An examination of possible analysis techniques should be included in these early phases as they may affect how data are collected and treated. And, it is necessary to consider potential sources of error in collected data and determine how to deal with them.

Determination of a sampling scheme, corrections to the raw data and final preparations of corrected data prior to analysis were briefly discussed in Chapter 2 (METHODS) under subheadings: Sampling Scheme, Raw Data Corrections, and Post-Correction Pre-Analysis Data Preparations. Below, details of each operation are presented, in the order they were applied to the data.

Sampling Scheme

A sampling scheme is determined by data-requirements along with consideration of potential analysis techniques. Here the

data-requirements are simple. Long-term time-series records of temperature and conductivity are needed to assess changes in the salinity of San Francisco Bay and what effects these changes may have on the bay. To make assessments data are looked at by themselves for trends, and are correlated with various biological, chemical, physical and meteorological observations taken throughout the bay and bay-delta region on an ongoing basis. Long-term salinity-trends (scale of years), as well as short-term phenomena, such as freshwater-input from storm-runoff (scales of hours to days), are important in the determinations.

While detailed sampling during periods of high freshwater-inflow or high freshwater-diversion, with lighter sampling at other times, might be appropriate, accurate prediction for periods requiring more intense sampling is difficult. Also, a variable rate sampling schedule may make long-term analysis more difficult. Many analysis techniques require, or at least perform better with, data evenly-spaced in time. So, the data-requirements and data analysis constraints indicate an equal-interval sampling-schedule with the period necessary to catch events of the shortest expected duration.

An intense storm of only an hour duration may deliver enough freshwater to temporarily affect surface-salinities by several salinity-units in areas of concentrated runoff. So, the quick-event period must be selected as, at most, a few hours. The sampling-theorem

for time-series analysis states that only those signals which have a period no less than one-half the sample-period used to collect the data can be accurately reproduced from an equal-interval sampled data set. Thus, to reproduce signals with a period of one hour, the sample-period for the data set must be no greater than one half-hour. This is an ideal condition. The sampling period should, in general, be shorter. A fifteen-minute-interval sample rate is used by the data collection stations discussed in this report.

Initial Inspection

Once a sampling scheme is established and a data set for time-series analysis collected, raw information must be looked at to determine if the obtained samples will be useful. It may be that a sampling scheme was inappropriate for an investigation, or perhaps collected data are too noisy to effectively analyze. Regardless of any need for unbiased reports, an investigator should have some idea of what is expected, and decide to continue with an analysis or consider some other approach, based on the condition of the raw information.

The raw data of this report (Figure A-1) represent a reasonable set of time-series records for measurements in an embayment on the California coast. There is an apparent semidiurnal period in the records. Values for the readings fall within a believable range for such waters. However, there are some notable problems within the data. There are

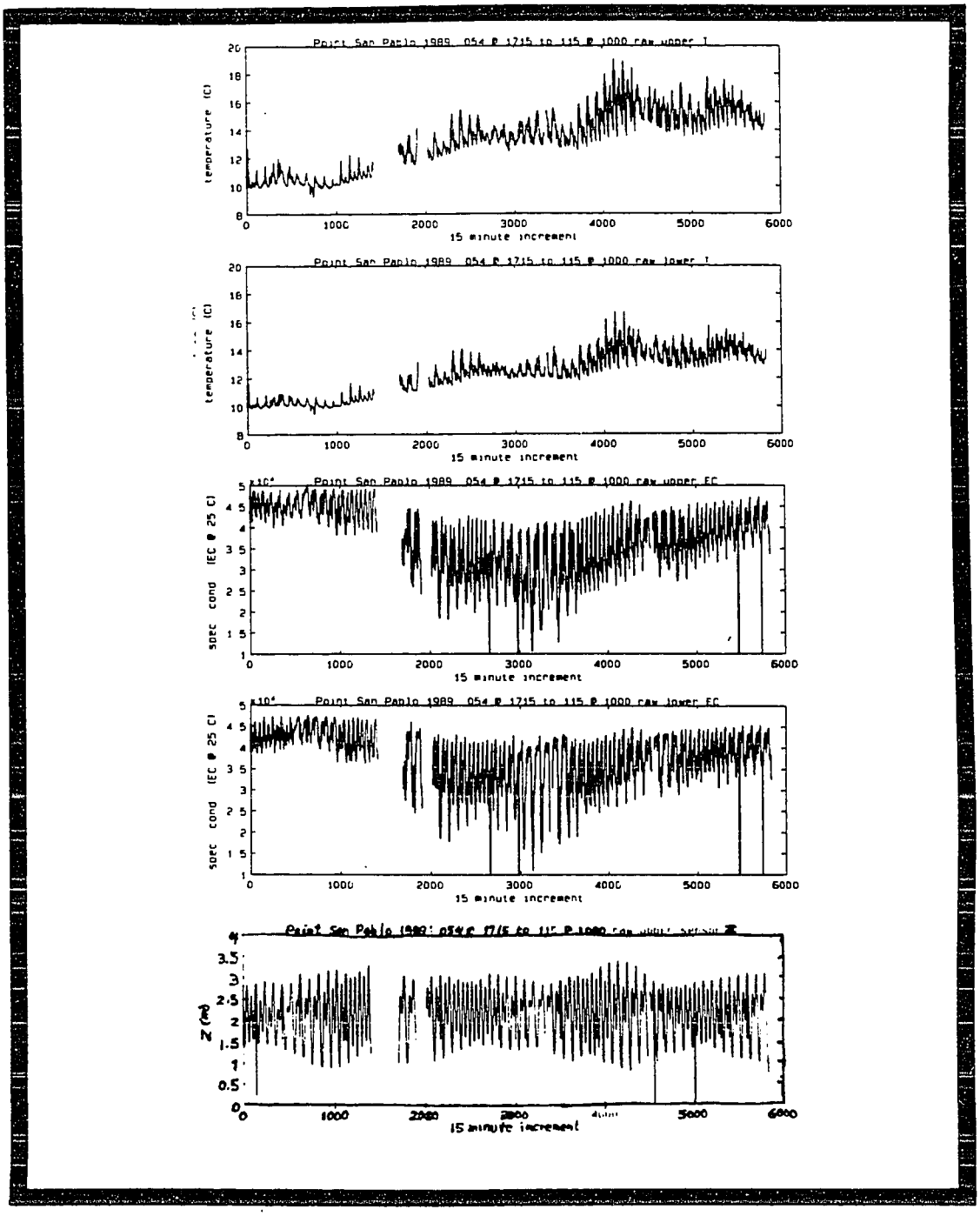


FIGURE A-1: Raw data records from the Point San Pablo water quality monitoring station for the period 23 February 1989 at 1715 (ST) to 25 April 1000 at 1815 (ST)

several large gaps in the records. From beginning to end the records should contain 5828 values. The raw data records contain 5345 values. 483 points, or 8.29 percent of each record was lost to events causing interruption of sampling. There are points where data values are not believable, (particularly in conductance and tidal-excursion). Also, there is a problem existing in, but not obvious from the plots: there are errors in the data set values. Readings contain both fixed and time-dependent offsets from actual values of the measured parameters. These are errors inherent in the sampling system itself, and errors from external sources such as sensor fouling. They must be addressed, and proper corrections made to the raw data set prior to data analysis.

Manual Corrections

Corrections to raw-data made by running fixed-algorithms within programs on a computer are not perfect. This is seen most vividly when trying to remove outlier-values through an automated process. Cases exist where, though it is obvious to a person looking at a plot of the data, a value in a raw-data set will not be seen as an outlier by a correction program, even if it is a physical impossibility. The correction programs and data set of this thesis are no exceptions.

With experience correcting data sets of a particular type using the same correction-scheme, data-points that may be potential problems start becoming visually apparent in raw-data plots. The first step in data

correction here was to look at plots of the data and visually interpolate through points that appeared to the eye as those which the correction routines might 'have trouble with.' No points were manually edited in the temperature records, 18 were edited in both of the conductivity records, and 21 were edited in the tidal-excursion data (Table A-1). Thirteen short gaps (16 records total) of 45 minutes or less, caused when sampling was interrupted while the station was remotely accessed for data-downloading, were also interpolated out in this manner (Table A-2).

Although interpolations were based on visual interpretations, a computer was used in the process. This work was performed on a Silicon Graphics IRIS workstation using a program (PLOTSAL) developed by Mr. Jon Burau of the USGS and modified by this author to suit corrections for data of the type in this thesis. Using PLOTSAL one can load data-files and display the records graphically on a high-resolution color-monitor. Once the data are 'on screen' a record can be expanded, scanned, and edited, under keyboard-and-mouse control. The program generates two files, one containing the edited data set (Figure A-2), and one documenting all the changes. The input data file is not changed.

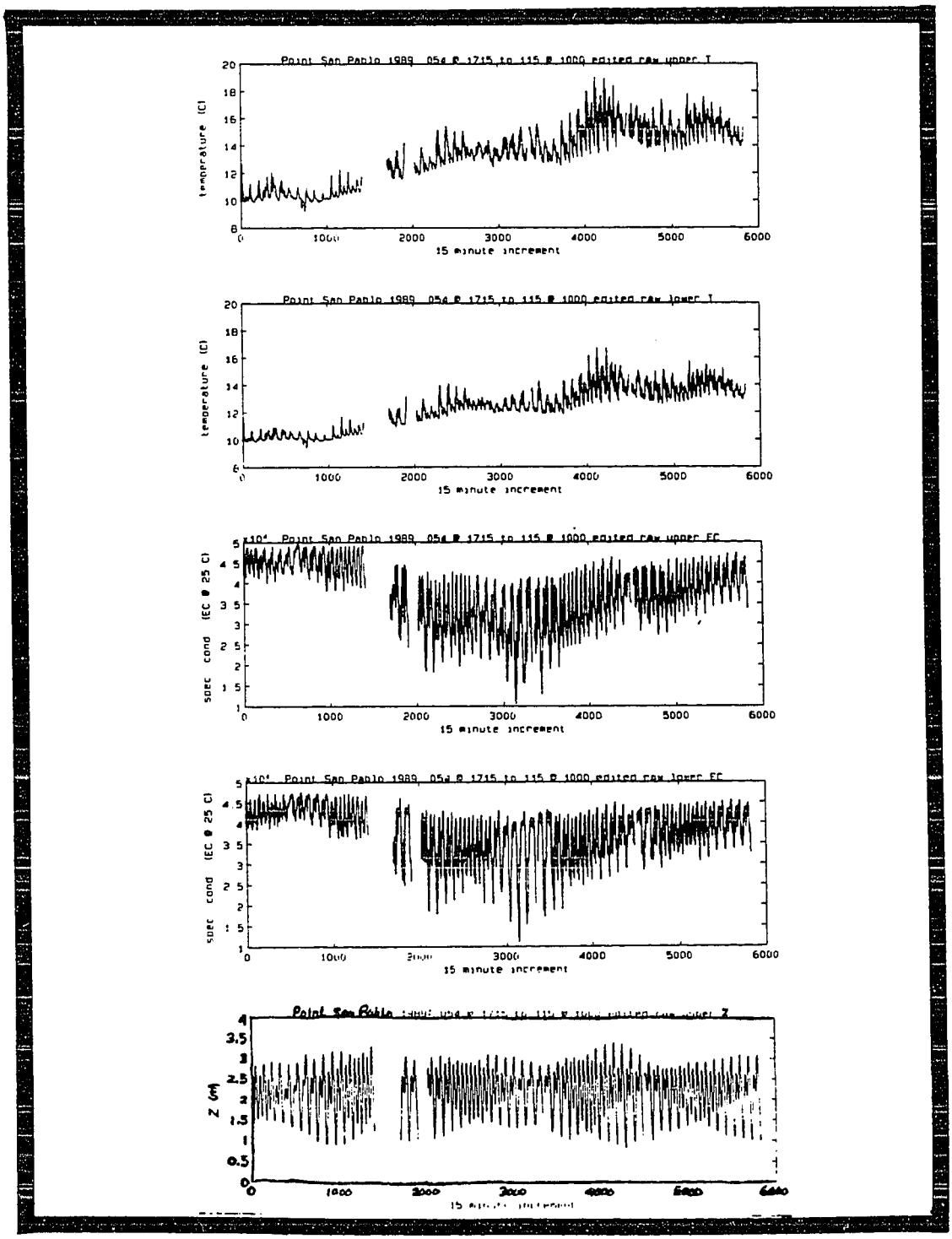


FIGURE A-2: Raw data records after manual editing process.

TABLE A-1: Interpolation corrections to the 5345-point raw series. Record numbers refer to full 5828-point data-set span.

record		Upper-C		Lower-C		Tide	
number	day time	from	to	from	to	from	to
134	056 0230					0.23	2.86
430	059 0430					1.53	2.85
622	061 0430					1.54	2.86
1125	066 1015					1.55	2.86
1136	066 1300					1.54	2.85
1386	069 0330					1.54	2.86
2172	077 0800					1.54	2.85
2659	082 0945	15	33821	15	34540		
2660	082 1000	15	33897	15	34753		
2661	082 1015	15	33770	15	34809		
2662	082 1030	15	33770	15	34921		
2663	082 1045	31	33821	15	34865		
2664	082 1100	15	34074	30	34809		
2823	084 0245					1.55	2.86
2979	085 1754	15	33394	30	38804		
2980	085 1800	15	33758	15	38965		
2981	085 1815	15	33987	30	39160		
3006	086 0030					1.53	2.84
3116	087 0400					1.54	2.86
3511	091 0645					1.55	2.86
3911	095 1045					1.54	2.85
4562	102 0530					-1.08	2.84
4660	103 0600					1.54	2.84
5011	106 2145					-2.40	2.86
5016	106 2300					0.22	2.86
5301	109 2215					1.55	2.86
5398	110 2230					1.54	2.86
5466	111 1530	31	44122	15	43165		
5467	111 1545	15	43419	15	42486		
5468	111 1600	15	42715	15	41806		
5469	111 1615	31	42012	30	41127		
5470	111 1630	15	41308	15	40447		
5471	111 1645	15	40404	15	39765		
5472	111 1700	15	39901	15	39088		
5591	112 2245					1.55	2.87
5604	113 0200					1.54	2.86
5732	114 1000	0	35433	30	34205	1.54	2.86
5733	114 1015	15	35497	15	34311	1.54	2.86
5828	115 1000					0.54	1.06

TABLE A-2: Manual-interpolations of 13 short gaps in the raw-data time-series. Record numbers refer to the full 5828-point data-set span

record		Upper-T	Upper-C	Lower-T	Lower-C	Tide
number	day time					
359	058 1045	11.30	43190	10.77	42559	1.40
648	061 1100	10.22	48083	10.15	46084	1.86
649	061 1115	10.28	47687	10.18	45749	1.78
1025	065 0915	10.13	47996	10.03	42125	2.93
2372	079 1000	12.69	35671	11.95	36497	2.82
2654	082 0830	13.54	23725	12.60	31449	1.68
3052	086 1200	13.66	21377	12.53	27260	1.76
3710	093 0830	12.81	36375	12.11	36056	2.88
4016	096 1300	14.12	40158	12.87	40149	2.88
4017	096 1315	13.99	40691	12.83	40472	2.85
4018	096 1345	13.87	41225	12.78	40796	2.81
4380	100 0800	16.63	35244	15.01	34495	1.24
4414	100 1630	16.09	39561	14.64	37449	2.63
5055	107 0845	14.40	39782	13.25	38965	2.60
5342	110 0830	15.53	37268	13.98	36969	1.82
5724	114 1230	14.59	38296	13.57	37441	1.05

Computer Corrections

Introduction

After manual corrections, the remaining data manipulations prior to analysis were performed using a computer. Algorithms in a program set written by this author and currently in use at the USGS for correcting data collected from the San Francisco Bay monitoring-station network were used to correct data for outliers, offsets, and long-term drift. Manual interpolations of data as outlined in the previous section are not performed on raw data placed in the network data base. The original raw data sets are retained. Manual editing is performed only on records that have been run through the correction algorithms. This eliminates the possibility of 'throwing away' something that looked 'bad' but was actually 'real.' The original raw data of this thesis were also retained. Manual and computer operations were performed on copies of those data.

Reference-Based Outlier Corrections

Output from the USGS data acquisition system (Table A-3) includes a pair of reference-voltage values, one a fixed-level and one the system-ground. Circuitry for the fixed-level is designed for extreme temperature insensitivity. Barring other effects, readings by the system that differ from the expected value for this reference (3.468 V) may be attributed to the effects of temperature on the rest of the system. The

ground-reference reading should always be zero. Nonzero system-readings of the ground-reference, barring other effects, may be considered some measure of the affects of power-surges and related events on the system. Corrections to raw data were made based on the differences of these references from their expected values. Corrections based on the ground-reference were made first, followed by corrections based on the reference-level. The reference-level was itself corrected using the ground-reference prior to reference-level corrections to the data.

A ground-correction was made only if the difference between zero and the ground-reference reading was greater than 0.001 V. The resolution of the 12-bit analog-to-digital conversion is approximately 0.0015 V, so, assuming a plus-or-minus 1 least-significant-digit accuracy for the digitizer, changes in readings on the order of 0.001 V are not significant. Sensor data-values were ground-corrected using the ratio of the difference between 0.001 V and the ground-reference reading, to the full-scale voltage of the digitization-process (4.998 V). This ratio was multiplied by the possible full-scale-range for a sensor (50° C for temperature, 60000 µS cm^{-1} for conductivity, 35.14 m for tidal-excursion), and that result subtracted from the corresponding sensor data-value.

Before making corrections using reference-level readings, the portion of any ground-reference reading greater than 0.001 V was subtracted from corresponding system-reading of the reference-level. The

TABLE A-3: Sample raw-data output from the Point San Pablo water-quality monitoring-station.

*** POINT SAN PABLO 23-FEB-1989(054) 1715(ST) PHONE RESTART ****

DAY	TIME	TOPT	TOPC	BOTT	BOTC	DPTH	V-REF	G-REF
54	1715	10.26	47166	10.14	45582	1.77	3.485	0.001
54	1730	10.26	47071	10.12	45429	1.69	3.484	0.000
54	1745	10.40	46707	10.16	45170	1.62	3.479	0.000
54	1800	10.31	46549	10.12	44972	1.54	3.485	0.001
54	1815	11.00	45506	10.18	44820	1.50	3.479	0.001
54	1830	11.75	44446	11.23	43236	1.46	3.483	0.000
54	1845	11.97	44209	10.90	43465	1.43	3.479	0.001
54	1900	12.72	43577	11.65	42597	1.39	3.479	0.001
54	1915	12.46	43450	11.23	42673	1.39	3.479	0.000
54	1930	12.23	43640	10.09	43343	1.39	3.479	0.001
54	1945	11.85	43798	10.14	41957	1.39	3.479	0.001
54	2000	11.09	42660	10.91	41333	1.40	3.479	0.001
54	2015	11.18	43561	10.01	41424	1.40	3.482	0.000
54	2030	11.18	43450	10.14	41302	1.43	3.479	0.000
54	2045	10.64	43324	10.11	41226	1.48	3.479	0.001
54	2100	10.45	43529	10.27	39018	1.54	3.479	0.001
54	2115	10.21	41995	10.18	39002	1.63	3.479	0.000
54	2130	10.21	41616	10.00	41591	1.69	3.479	0.000
54	2145	10.26	41395	10.05	40632	1.76	3.479	0.000
54	2200	10.26	41331	10.16	40114	1.84	3.479	0.001
54	2215	10.62	44114	10.03	40708	1.89	3.478	0.001
54	2230	10.64	44589	10.05	40647	1.96	3.479	0.001
54	2245	10.52	44715	10.11	40556	2.05	3.479	0.001
54	2300	10.26	44178	10.14	39977	2.16	3.479	0.000
54	2315	10.19	45031	10.05	40571	2.23	3.479	0.001
54	2330	10.12	46834	10.03	40723	2.29	3.479	0.000
54	2345	10.10	47055	10.05	40723	2.37	3.479	0.001
55	0	10.07	47435	10.00	41043	2.43	3.478	0.001
55	15	10.14	47229	10.03	41058	2.51	3.478	0.001
55	30	10.10	47530	10.03	41272	2.58	3.478	0.000
55	45	10.07	47719	9.98	41531	2.62	3.478	0.000
55	100	9.98	48099	9.94	41759	2.68	3.478	0.001
55	115	9.98	48289	9.94	42033	2.72	3.478	0.001
55	130	9.95	48415	9.92	42353	2.75	3.477	0.001
55	145	9.93	48668	9.90	42810	2.78	3.477	0.001
55	200	9.93	48731	9.90	43221	2.81	3.478	0.001
55	215	9.98	48273	9.90	43343	2.82	3.477	0.000
55	230	9.98	48162	9.90	45719	2.84	3.477	0.001

ground-corrected reference-level was then used to further correct raw-data values. This correction was to divide the expected reference-level (3.468 V) by the ground-corrected reference-level reading, then multiply this result into all data-values taken at the same time as the reference-level reading.

Making ground-level (offset) and reference-level (slope) corrections to the data amounts to a heuristic means of removing some outlier-values. However, these corrections have no affect on long-term drift-errors, and no affect on outliers that are produced from sources that do not affect the data-digitization process. In fact, these corrections may introduce greater error if not applied cautiously, and are optional operations in the corrections program. Reference-values are always examined to see if a problem with the reference-circuitry has developed before applying the reference-corrections. This was the case for early portions of the data set examined here. Reference-level corrections were not applied to portions of the record where the reference-level circuits were in need of repair.

Long-Term-Drift Corrections

At approximately two-week intervals, instruments at each station are physically inspected. The time between inspections constitutes a record period. Long-term drift corrections of temperature and conductivity were made with reference to comparison readings taken at

the beginning and end of each record using a Beckman model RS5-3 induction-salinometer. Corrections to tidal-excursion records were made with reference to readings for the depth sensor suspended 1.0 m and 2.0 m below the surface. Comparison readings for the beginning of a record were taken from clean sensors. The end-record comparison readings were taken before any fouling which developed during the record period was removed from the sensors.

Long-term drift corrections calculate the difference between sensor readings and their respective comparison readings (Table A-4), for the beginning and end of the record, and then prorate, in time, into each value in a record, the difference between differences (Figure A-3). After prorating, the offset between initial clean-sensor readings and their comparison-readings were removed from every value for each record. Removal of this initial offset-value was assumed to eliminate any fixed-difference between system-outputs and the actual value of an observed parameter. The prorating-correction was assumed to remove the effects of electronic drift and sensor fouling on system readings. The mean is removed from tidal-excursion records to eliminate small offsets due to inaccurate rehanging of the fixed-altitude sensors.

TABLE A-4: System calibration-and-correction data from the Point San Pablo water-quality-monitoring station for the period 23 February 1990 to 25 April 1990. There were four raw-data record subperiods. The main record periods are subdivided by remote access data download breaks, but the same prorating is applied to each subperiod within a main period. Reference salinities are converted to specific conductance ($\mu S\ cm^{-1}$) referenced to 25° C for use in the correction-algorithms.

RS5-3 portable salinometer corrections:

corrected-salinity = read-salinity - 0.24
corrected-temperature = read-temperature

Begin-Record: 23 Feb 1989 (054) @ 1330

sensor	corrected reference	system reading	time
TOPT	10.53	10.04	1325
TOPC	44697	49214	1326
BOTT	10.33	10.08	1258
BOTC	45071	44158	1257
DPTH	0.95	1.08	1300

bottle salinity	spec. cond. at 25° C	time
28.90	44745	1233

bucket thermometer	time
none	

End-Record: 15 Mar 1989 (074) @ 1315

sensor	corrected reference	system reading	time
TOPT	14.34	14.07	1338
TOPC	22995	24260	1340
BOTT	14.14	13.57	1357
BOTC	22995	21305	1400
DPTH	0.95	0.88	1350

bottle salinity	spec. cond. at 25° C	time
13.88	23010	1402

bucket thermometer	time
none	

TABLE A-4: Continued from previous page.

Begin-Record: 15 Mar 1989 (074) @ 1445

sensor	corrected reference	system reading	time
TOPT	14.38	13.62	1430
TOPC	22995	22562	1431
BOTT	14.25	12.98	1417
BOTC	22995	21038	1420
DPTH	0.95	0.93	1435

bottle salinity	spec. cond. at 25° C	time
13.88	23010	1402

bucket thermometer	time
none	

End-Record: 30 Mar 1989 (089) @ 1000

sensor	corrected reference	system reading	time
TOPT	14.26	13.58	1030
TOPC	15348	19760	1032
BOTT	14.50	13.19	1045
BOTC	16199	16789	1046
DPTH	0.95	0.87	1040

bottle salinity	spec. cond. at 25° C	time
9.36	16022	1103
9.17	15722	1123

bucket thermometer	time
14.40	1103
14.60	1123

TABLE A-4: Continued from previous page.

Begin-Record: 30 Mar 1989 (089) @ 1130

sensor	corrected reference	system reading	time
TOPT	14.71	14.39	1122
TOPC	15758	17939	1123
BOTT	14.48	13.37	1101
BOTC	15821	16345	1102
DPTH	0.95	0.89	1110

bottle salinity	spec. cond. at 25° C	time
9.36	16022	1103
9.19	15722	1123

bucket thermometer	time
14.40	1103
14.60	1123

End-Record: 11 Apr 1989 (101) @ 1000

sensor	corrected reference	system reading	time
TOPT	17.30	16.32	1020
TOPC	31383	34899	1123
BOTT	17.00	15.25	1041
BOTC	29858	30306	1039
DPTH	0.95	0.85	1120

bottle salinity	spec. cond. at 25° C	time
17.15	27914	1100
17.32	28166	1117

bucket thermometer	time
none	

TABLE A-4: Continued from previous page.

Begin-Record: 11 Apr 1989 (101) @ 1130

sensor	corrected reference	system reading	time
TOPT	16.91	16.24	1118
TOPC	28458	31880	1117
BOTT	16.86	15.07	1100
BOTC	27587	28050	1057
DPTH	0.95	0.83	1120

bottle salinity	spec. cond. at 25° C	time
17.15	27914	1100
17.32	28166	1117

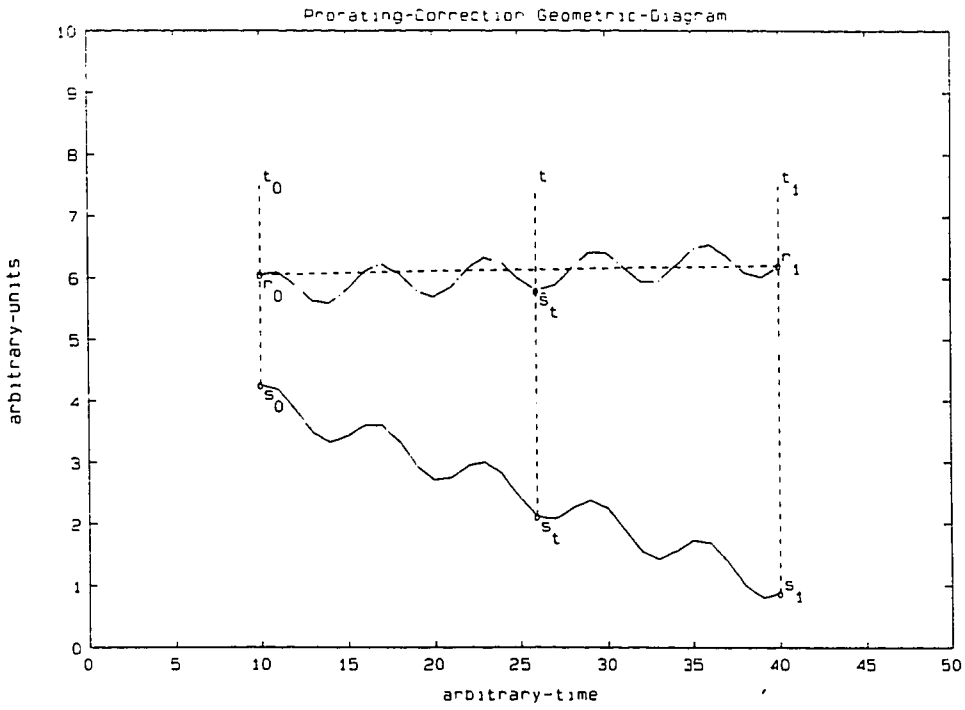
bucket thermometer	time
none	

End-Record: 25 Apr 1989 (115) @ 1000

sensor	corrected reference	system reading	time
TOPT	15.86	15.27	1011
TOPC	31863	34677	1013
BOTT	15.85	14.15	1035
BOTC	31703	32393	1036
DPTH	0.95	0.80	1017

bottle salinity	spec. cond. at 25° C	time
19.94	32013	1058
20.00	32101	1125

bucket thermometer	time
15.50	1056
15.40	1124



r_t = reference-reading at time t

s_t = signal-reading at time t

$o_i = r_0 - s_0$ = initial-fixed-offset

$$m_p = \frac{(s_1 - r_1) - (s_0 - r_0)}{(t_1 - t_0)} = \text{prorating slope-factor}$$

$P_t = (t - t_0)m_p$ = prorating-correction-value at time t

$\hat{s}_t = s_t - P_t + o_i$ = offset and prorating-corrected s_t

FIGURE A-3: Geometric diagram of data fixed-offset and prorated-drift corrections scheme.

Statistical Outlier Removal

Outliers-values may represent real events in a data record. However, a small subset of values far removed from the average of an entire record can have a large effect on the variance calculated for that record. In this case statistical computations made with regard to long-term phenomena in the record may be adversely affected. Thus, when looking at a long record for long-term phenomena, outlier values should be removed even though they may represent real short-term events. The algorithm for outlier-removal used here was:

1. calculate means of data-column segments
2. calculate standard-deviations of data-column segments
3. scan column segments for values which, in absolute-value, the value minus the column-mean is greater than 2.5 times the column-standard-deviation.
4. from surrounding non-outlier values interpolate a new value to replace the outlier-value

4a. If there are consecutive outlier values, then perform a weighted-interpolation to properly set the replacement values in time.

For a normal distribution, an absolute distance of 2.5 standard deviations from the mean says that a data point is outside the range of (approximately) 97.5 percent of the data set. Long-term time-series of tidally-driven data are not normally-distributed. But, experience with this algorithm has shown that under most conditions, when applied sectionally to record-segments no longer than 10 days, it will eliminate only values

that are 'well removed' from the bulk of data of the type under examination. Application of this algorithm to longer tidally-driven records can result in acceptable points being rejected as outliers. This is due to a range-constraint on standard deviation that relates to the square-root of the number of points in a record. Given a record of length N, with mean M and standard-deviation S, for any value in the record, V, $|V - M|/S \leq \sqrt{N}$ (Kirby 1974). So, the longer the record, the greater the possibility of a value being rejected as an outlier when a simple multiple of the record standard-deviation is used as the determining measure.

After manual editing and computer-corrections, (reference-based outlier removal, drift prorating, offset removal, and statistical outlier removal), the data-values (Figure A-4) are at least reasonably close to the true values of measured parameters. However, they are still not ready for analysis. Once we have reasonable data values, they must be fit to constraints placed on them by the analysis techniques.

Data Manipulations Due to Constraints of Analysis Techniques

Introduction

Data analysis requires not just corrected data, but corrected data of the proper form and format for the analysis technique. Data must be properly conditioned and scaled to fit a technique before any analysis

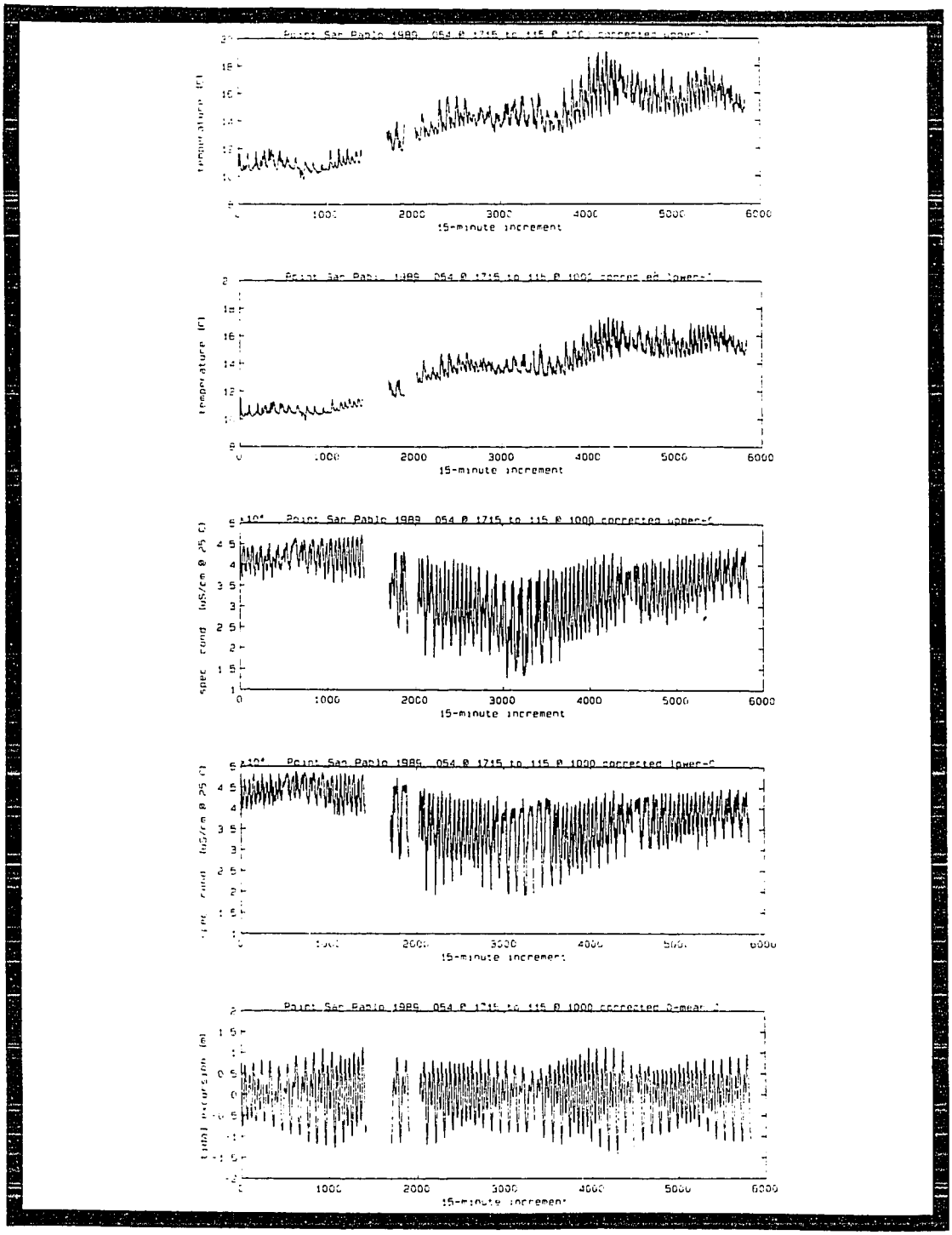


FIGURE A-4: Data records after manual editing and computer correction process.

can be performed. When, as here, more than one technique is to be applied to the same data set, those data must be forced to fit the constraints of all techniques simultaneously or comparisons between methods may not be valid.

Of the techniques examined in this thesis spectral analysis places the greatest constraints on data. These limitations arise from use of the Fast-Fourier-Transform (FFT). Setting up data for use in the FFT-algorithm nearly fits them to all constraints of the other analysis techniques examined in this thesis. Preparing data for FFT-algorithm processing, as well as fitting them to the data-limitations of the other analysis techniques is described below. Constraints due to the analysis-system hardware are also taken into account.

Establishing Record Length

The FFT requires the number of points in a record be an integer-power of two. The fifteen-minute interval data record from Point San Pablo 1989 day 054 at 1715 PST to day 115 at 1000 PST contains (theoretically) 5828 readings. The nearest power-of-two number of readings is 4096. To use only 4096 readings from 5828 means eliminating approximately 30 percent or about 18 days from the near 60-day record. Examining plots of the data corrected to this point (Figure A-4) shows that eliminating data from day 054 at 1715 (record 1) to day 065 at 0300 (record 1000), and eliminating data from day 107 at 1915 (record 5097) to

day 115 at 1000 (record 5828) leaves a 4096-point record that covers the apparent period of storm affects with a few days lead-in and tail-off time. This is acceptable for our purposes, so the first constraint of the FFT is met within the time span of the data record.

Padding

The FFT only works with equal interval data. We set out to collect 15-minute interval data and were reasonably successful. However, after corrections there are four gaps in the data (Table A-5), and the FFT algorithm will not tolerate them. The gaps must be filled in, or padded. Padding may be accomplished in several ways. Frequently one will see zeros being added to extend or fill in a record. However, this can cause aliasing and other problems in spectral analysis. Padding was performed here as a weighted-interpolation across the gaps point-by-point, relative to the 24 hour cycle of a parameter immediately before and immediately after a gap (Figure A-5). For Gap-1, the later portion of the leading 24 hour cycle had to be extrapolated from the corresponding portions of the two cycles preceding that cycle. Four BASIC-language programs were written to implement these procedures: PADGAPS.BAS, PADGAP2.BAS, PADGAP11.BAS and PADGAP12.BAS (Appendix F). After padding, the data are nearly ready for analysis. However, not only the analysis constraints, but constraints imposed by the equipment used to perform the analyses must be considered.

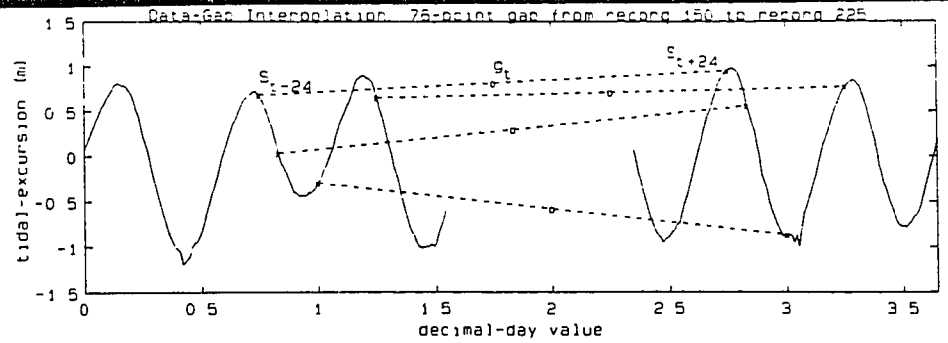
TABLE A-5: Location of data-gaps remaining after manual and computer corrections to the raw data-set from the Point San Pablo station for the period 23 February 1990 to 25 April 1990. Record numbers refer to the full 5828-point data-set.

	gap-start		gap-end	
Gap-ID	record number	day time	record number	day time
Gap-1	1410	069 0930	1702	072 1030
Gap-2	1906	074 1330	2024	075 1900
Gap-3	3333	089 1015	3359	089 1645
Gap-4	4485	101 1015	4512	101 1700

Low-Pass Filtering and Decimation

Low-Pass filtering of data intended for time series analysis is generally good practice. It helps to eliminate high-frequency noise affects and reduces the potential for aliasing (Appendix D). Besides low-pass filtering of data, the filtering technique used here also decimated the records. An output from the filter was generated only every four time-steps through the data. This, while maintaining the overall time-span of the record, converted the data set from 15-minute interval information to 1-hour interval information.

Much of the data-analysis was performed using a multivariate statistical analysis package, PC-Matlab, on an IBM-PC-Clone 80286-based



$$g_t = (S_{t-24} + S_{t+24})/2$$

where:

g_t = calculated gap-value at time t

S_{t-24} = existing signal-value 24-hours before time t

S_{t+24} = existing signal-value 24-hours after time t

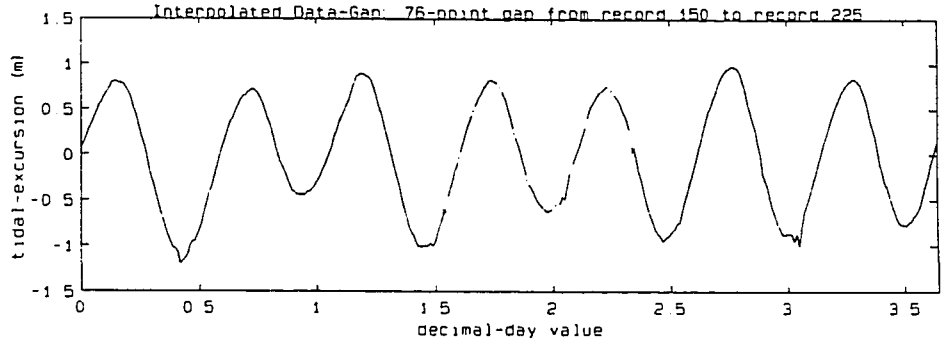


FIGURE A-5: Data gap interpolation process diagram for a data-gap less than 24 hours wide. This extends to larger gaps as a weighted interpolation process.

personal-computer. The need to decimate records arose from a constraint placed on factor analysis by a programming limitation placed on the analysis software by memory limitations of the MS-DOS operating system. PC-Matlab allows for approximately 8000 data-values in a variable, so our 4096-point records are not a problem for individual-analysis. However, the initial calculations of factor analysis (Appendix C) require all five 4096-point records be available at one time. This, after PC-Matlab itself is loaded, exceeds the 640K byte memory limit of the MS-DOS operating system. For the factor analysis routines to work, record size needed to be reduced from 4096 readings to 1024 readings. This could have been accomplished by simply 'throwing out' (decimating) all records whose readings did not fall on-the-hour. But, without low-pass filtering, this might have caused serious aliasing problems in the data (Appendix D). The data were passed though a filter that accomplished decimation and low-pass filtering in one step. The normalized, 9-point, least-squares fit, quadratic-weight transversal filter is analyzed in detail in Appendix-D. It has the following form:

$$y(t) = \sum_{i=-4}^{4} b_i x(t+i), \quad t=5+n*4, \quad n=0,1,2,... \qquad (A\text{-}1)$$

where: $y(t)$ = output at time-step t

$x(t)$ = input-variable value at time-step t

$\{b_i\} = \frac{1}{231}\{-21,14,39,54,59,54,39,14,-21\}$

Output values, by the increment of n on t in the above expression (A-1) come at one-hour time-steps, and by the weighting of the $\{b_i\}$, are low-pass filtered to contain little or no information having a period shorter than 2 hours (Appendix D). This filter was implemented as the BASIC-Language program 9PTFLTR.BAS (Appendix F). The result of applying this filter to the padded data is a set of 1456-point one-hour interval data records spanning from day 057 at 1815 to day 115 at 0915. The selected 1024-point records span from day 065 at 0315 to day 107 at 1815 (Figure A-6).

Scaling and Unit-Conversion

Scaling is the process of increasing or decreasing the overall amplitude of a record by multiplying all elements of that record by some fixed-value. For our purposes scaling is used to bring column-variances all within the same order-of-magnitude. This is important for factor analysis. The factor analysis technique used here is based on looking at column-variances. One column with a variance much greater than the variance of the other columns will dominate in the analysis. For the data preconditioned to this point, temperature columns and the depth column have standard deviations on the order of 1.0, while the conductivity columns have standard deviations on the order of 6000. The large-magnitude difference is due to different unit types for the columns; temperature and depth having values no greater than 19.0, while specific conductance has a numerical range of 13000 to 49000 (Table A-6).

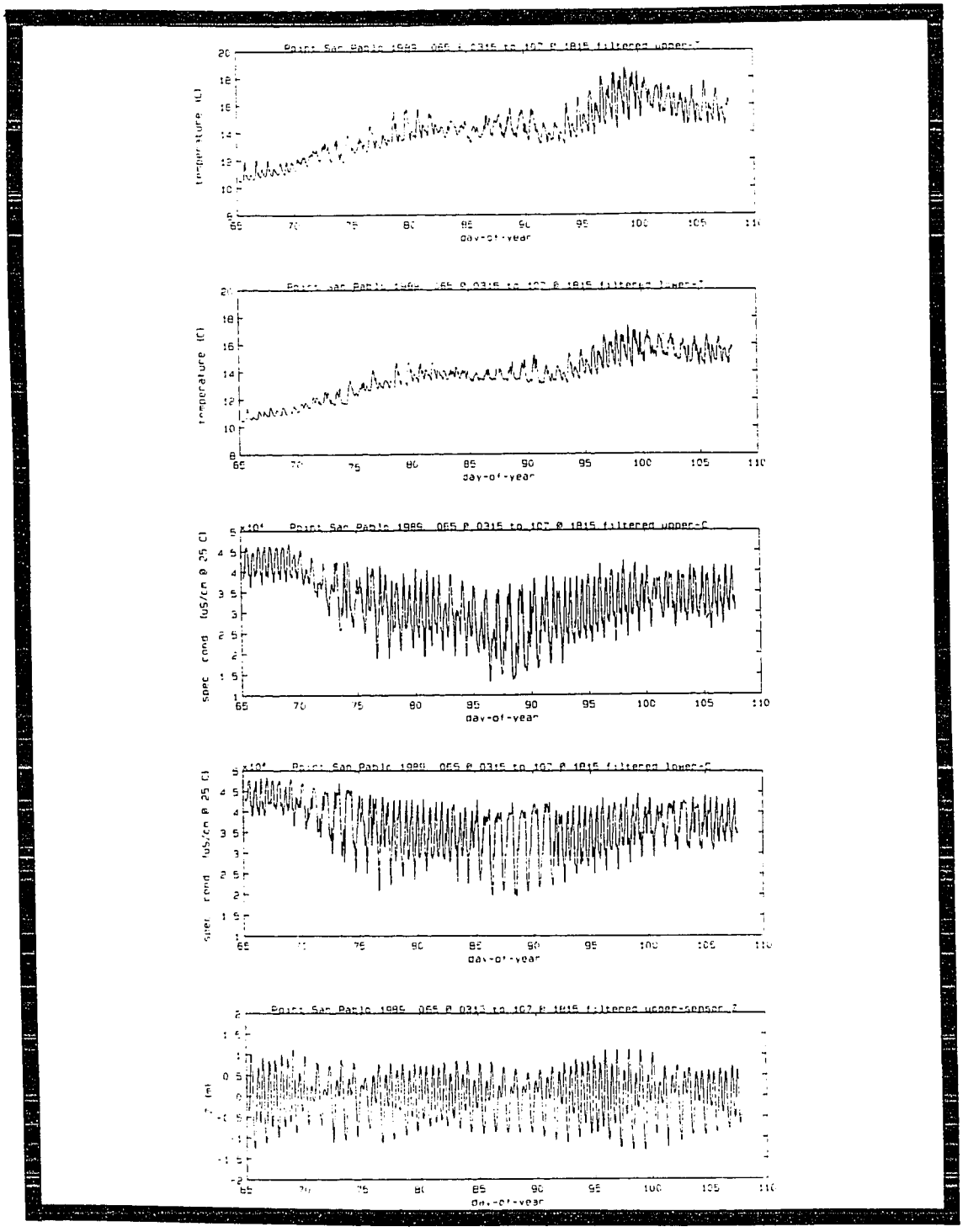


FIGURE A-6: Corrected data records after padding and decimation.

Specific conductance values must be scaled before factor analysis can be applied to the data.

TABLE A-6: Maximum, minimum and standard-deviation values for the Point San Pablo station data-set after manual and computer-corrections, and low-pass filtering and decimation to 1024-point records. The values for the conductivity-columns after conversion to salinity are included.

	TOP-T	TOP-C	BOT-T	BOT-C	DPTH	TOP-S	BOT-S
maximum	18.75	47045	17.38	48396	1.14	30.56	31.35
minimum	10.47	13215	10.45	19474	-1.31	7.38	11.52
std. dev.	1.78	6925	1.57	5906	0.55	4.75	4.10

Frequently all that is required to bring signal-variances in line is algebraic scaling, i.e., multiplying each signal value by some fixed factor. This technique can sometimes lead to confusion due to unspecified unit changes that result from the multiplications. Algebraic scaling would work to reduce the magnitude of variances calculated for specific conductance data, but leaves these signals with arbitrary units. Here specific conductance values are referenced to a single temperature (25° C) so they convey the same information as salinity. Salinity values are within the same order of magnitude as temperature values. So, a unit conversion from specific-conductance to practical-salinity was performed to bring the magnitude of specific-conductance signals closer in magnitude to temperature and tidal-excursion signals. Conversion from

specific-conductance (μS cm^{-1} at 25° C) to salinity (S), was accomplished using the Unesco/SCOR approved algorithms for the PSS-78 equations (Fofonoff and Millard 1983). This is a unit conversion and not a scaling operation. Data are passed through a multi-term, fractional-order polynomial. The algorithm is coded in the FORTRAN program CONSAL.F (Appendix F).

Mean Removal and Detrending

The last preconditioning steps applied to the data examined in this thesis were removal of column-means and detrending of the columns. Removal of column-means is primarily an aid in using the FFT. A large mean value for a data set may be interpreted by the algorithm as a very low-frequency of very high-power and swamp out frequencies of interest in the analysis. Detrending is removing a least-squares linear-fit from the data. The reason for doing this is very similar to the reason for removing the mean. An observable linear-trend in a record, i.e., the appearance in a plot of the record that data have been superimposed over a sloping-line, may be seen as a low-frequency cycle in spectral analysis, and as a source of variance by factor analysis. These trends may be quite real. However, one must always consider what they are looking for. Obvious trends can be seen in plots of the padded, decimated data (Figure A-6). These trends extend through the period of the record, and thus are of a frequency far lower than any we are interested in. They may be seasonal effects, but would only interfere in

our analyses for tides and storm generated phenomena. They were eliminated through detrending. Mean removal and detrending were accomplished using functions within the PC-Matlab analysis package.

Final Prepared Data Set

The drift corrected, fouling corrected, padded, filtered, unit converted, mean removed, detrended data (Figure A-7) are slightly different in form than the raw data they were derived from (Figure A-1). However, they now fit the constraints of the various analysis techniques, and no longer contain most of the information, (real or not), that might interfere with analyses for the frequencies of interest. The data are now ready for application of our chosen analysis techniques.

Final Comments

The correction presented above give a sequence of steps that, barring 'unusual' problems in station records, produce data suitable for analysis. Ideally, correcting each record, then appending them in order should give a data set ready for analysis. However, raw data collected from network stations frequently require some form of 'special handling'. For example, interpolation used to pad data had to be 'custom fit' to record gaps. This required writing one general purpose interpolator, PADGAPS.BAS, and three routines, PADGAP11.BAS, PADGAP12.BAS and PADGAP2.BAS, customized to the first and second record gaps. Using

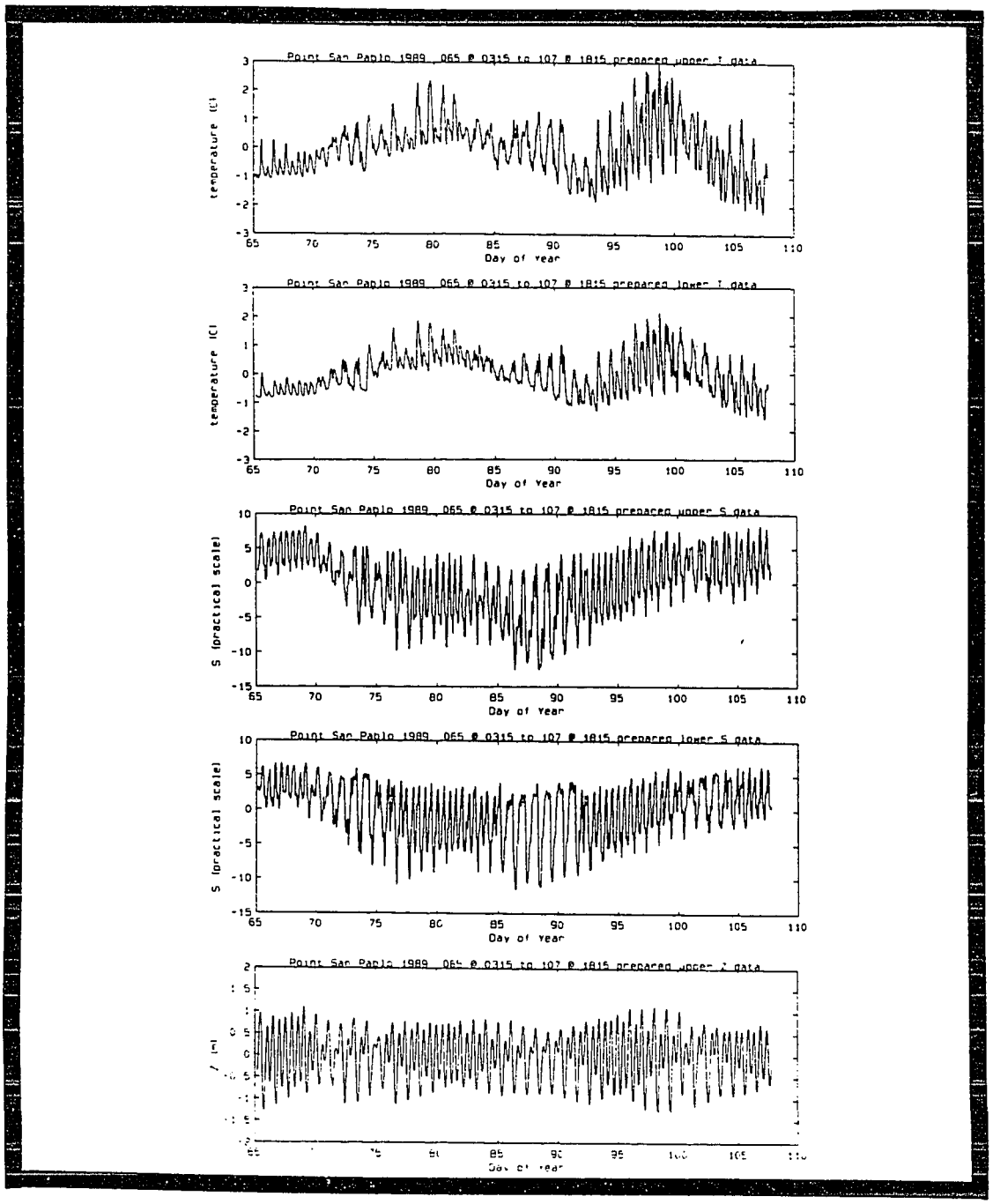


FIGURE A-7: Prepared data records from the Point San Pablo water quality monitoring station for the period 06 March 1989 at 0315 (ST) to 17 April 1989 at 1815 (ST).

these with other data sets would require some rewriting of the programs. Correcting segmented data records requires careful attention to the alignment of the tail of one segment with the head of the segment that follows it. Problems can arise in later analysis if there are errors. Here, records here are approximately two weeks in length. If there are errors in correction that cause small offsets between records, then the two week maintenance cycle may be interpreted as a spring-neap effect. Examination of tail alignments should be made on a segment by segment basis; looking at expanded plots of corresponding record ends. If there are alignment errors, then calibration data and correction values must be examined. If no errors are found there, then sensing equipment may be suspect. Errors introduced by failing to let equipment properly equilibrate before calibration readings are made will introduce this type of cyclic-offset error in corrected data. If precision requirements allow, such errors may sometimes be compensated for by extrapolating the proper values from the closest segment of a record where the readings appear stable; this can be difficult to detect in tidally-influenced data. Tidal filtering of appended raw data records and appended corrected data records can help in identifying these segments. The temptation to 'fudge' should be resisted. Often data must be rejected, rather than corrected, because of this type of error. Data corrections can only compensate for 'reasonable' errors. The greater the attention paid to sampling and calibration procedures, the greater the probability of successful data correction. This cannot be over emphasized.

APPENDIX-B

SPECTRAL ANALYSIS

Introduction

Any waveform, whether produced by a musical instrument or observed in a tidal record, may be thought of as the sum of sinusoidal waves of different frequencies, phases, and amplitudes. The purpose of spectral analysis (also harmonic, Fourier or frequency analysis) is to determine the frequencies and phases of the sinusoids that make up an observed waveform, and the energy contained in, (i.e., amplitude of), each of these component waves. The frequency of a component sinusoid gives us a clue to its source, (i.e., a tidal component, or a storm-driven component, etc.). The energy contained in a component gives its relative importance in producing the observed waveform. Spectral analysis is transforming a waveform from its time-domain representation (point-by-point description) to its frequency-domain description (displaying relative energy of underlying frequency components), so we can more readily determine structure within the signal.

We tend to think of waveforms more in terms of period (time) than frequency. However, the idea of multiple frequencies in a single data set becomes clear if one recalls that frequency is the inverse of period. When looking, for example, at a tide-record plot, multiple periods are

readily observable. Since period is the inverse of frequency, the data must contain multiple frequencies that produce the observed multiple periods in the plot. The expression of separate frequencies in a signal defines the spectrum of that signal.

When looking at predominantly tide-driven parameters, spectral analysis provides a means of examining nontidal components in an observed record. Using spectral analysis one may find the tidal components in a record. Once the tidal components are determined they may be removed from the record. The remaining (residual) part is the non-tidal portion of the signal. Given the residual one may, from its form, begin to ask questions such as: what, (if any), non-tidal processes were operating during the time the record was obtained?, and, how significant were these non-tidal processes to overall system-action at the location for that record?

For our purposes, spectral analysis is using one or more of a set of analysis techniques for separating an observed waveform into its sinusoidal components. These techniques do not work directly on time-domain data, but use various transforms or statistical measures of the time-domain information. Here we include autocovariance, autocorrelation, cross-correlation, the Fast-Fourier-Transform (FFT), and the digital filter. These techniques are all used in the main-text spectral analysis section of this thesis. In this appendix the listed techniques are described from the point-of-view of developing digital filters. Digital

filters have become a useful tool with the advent of the FFT and high-speed digital computers. They represent a powerful method for analyzing time-series data; rapidly gaining favor over more traditional statistical and time-domain filtering techniques.

Often one will see digital filters described beginning with explanations of aliasing, the Nyquist-criterion and the relation of input to output signals, similar to those given in Appendix D of this thesis for time-domain filtering. Here we take a (hopefully) more intuitive approach, and begin with descriptions of variance and covariance leading into autocorrelations, then to cross-correlations, harmonic analysis and the FFT, and finally to the concept of digital filters themselves.

Variance, Covariance, Autocovariance and Autocorrelation

Introduction

Given a data set $X = \{x_i\}$, $i=1,\ldots,n$. The mean of X is:

$$\mu = (\frac{1}{n})\sum_{i=1}^{n} x_i . \qquad \text{(B-1)}$$

The deviation of a single data-point, x_i, from the mean of its data-set, μ, is:

$$d_i = (x_i - \mu). \qquad \text{(B-2)}$$

The variance of X is:

$$\sigma^2 = (\frac{1}{n})\sum_{i=1}^{n} d_i^2. \qquad \text{(B-3)}$$

That is, variance is the mean of the square-of-deviations, d_i, of the data-points, x_i, away from the mean, μ, of variable X. Note that the sum of the deviations for a variable is zero. That is why deviations are squared when calculating variance. The unit of variance is the square of the unit of the variable it was calculated for. Standard-deviation, σ^2, is the square-root of variance; which has the same units as the original variable.

The concept of variance, a measure of dispersion of points in a data set away from the mean-value of that data set, forms the basis for most spectral analysis techniques. Variance, in and of itself, tells us very little about the spectrum of a variable. However the spectrum of a signal represented in a sampled data set can be determined via a systematic examination of variance and related measures. And, standardization of these examination procedures lead to autocorrelation techniques which gave rise, via the Fourier transform, to viable harmonic analysis methods.

Variance and Covariance

Variance was presented above as a simple descriptive statistic. That is, the equations imply we know the values for all points in our variable, X, and hence can calculate an exact mean, μ, and exact variance, σ^2, for that population of points. By definition our sampled data-sets cannot contain all possible points in any variable we are recording. Therefore we can only estimate the mean and variance from our sample of the population. We use $\bar{x}$ as the symbol for a sample-estimate of the mean, and s^2 as the symbol for a sample-estimate of variance. The calculation of sample-mean, $\bar{x}$, is the same as the calculation for population-mean, μ (B-1). However, simply using $\bar{x}$ in place of μ when calculating the d_i to apply in the equation for variance (B-3) tends to underestimate the variance of a population. That is the biased estimate of sample-variance. The so-called unbiased-estimate for variance from a sample is:

$$s^2 = \left(\frac{1}{n-1}\right)\sum_{i=1}^{n}(x_i - \bar{x}), \qquad \text{(B-4)}$$

Subtracting one from the sample-size in order to raise the value of the unbiased-estimate of variance (B-4) is not arbitrary. It relates to the degrees-of-freedom (df) available for making this calculation. Calculation of variance depends upon an estimate of the mean, but not vice-versa, thus the mean is said to be free. Since variance estimates depends on the sample-mean, the sample-estimate of variance loses one

degree of freedom. That is, the variance estimate has n-1 df. Every time an estimate of a parameter is used in calculating another parameter, the calculated parameter loses one degree of freedom.

It is important to note the concept of degrees-of-freedom and that the mean and variance calculated from a sampled variable are only estimates of the true mean and variance of the parameter being sampled. However, for a properly-sampled variable, (i.e., one who's sampling-scheme meets the Nyquist-criterion (see Appendix D)), where the sample size, n, is large, (i.e., greater than about 10), the calculation of a biased estimate of variance is virtually indistinguishable from the unbiased calculation.

Covariance is a measure of the joint-variation of two variables about their common mean. That is, it is a measure of the tendency for the corresponding values in the two variables to 'track' each other. If the values for one variable always tend to fall on the same side of their common mean as the corresponding values of the other variable, then the two variables covary and will have a high covariance.

For two variables, X_j, X_k, the covariance of X_j with X_k is calculated as:

$$cov_{jk} = \frac{n\sum_{i=1}^{n} X_{ij}X_{ik} - \sum_{i=1}^{n} X_{ij} \sum_{i=1}^{n} X_{ik}}{n(n-1)}. \qquad (B\text{-}5)$$

Given that two variables have similar units, a large covariance (B-5) means the two variables are linearly dependent, or perhaps that both are linearly dependent on another variable in a similar way. A low covariance implies that variables are independent. Covariance values must be used cautiously, as they are scale dependent. If one variable is measured in units of much greater magnitude than the other variable, then even if the variables are poorly related their calculated covariance may be numerically 'large,' To avoid scale problems when estimating two variables' interdependence we use the correlation-coefficient, r, which is the covariance of the variables divided by the product of their standard-deviations:

$$r_{jk} = \frac{cov_{jk}}{s_j s_k}. \qquad \text{(B-6)}$$

The correlation-coefficient is a ratio and therefore unitless. Covariance may never exceed the product of the standard-deviations of its variables. Hence, correlation (B-6) ranges from +1 to -1. A correlation of +1 indicates a perfect correspondence between the variables. A correlation of -1 indicates that one variable changes inversely with respect to the other variable. A correlation of zero indicates there is no linear-relationship between the variables.

Autocovariance and Autocorrelation

The prefix 'auto' is a combining form meaning 'by itself.' So, autocovariance means 'covariance by itself,' that is, the covariance of a variable with itself. Intuitively, calculating the covariance of a value with itself should give the maximum possible value for covariance. This is true. Calculating the covariance of a variable with itself is just the variance of that variable. In order to make this calculation useful we introduce the lag. A lag is an offset between a variable and the copy of that variable being used in the autocovariance calculation. We calculate autocovariance with a variable and a lagged-copy of that variable. Since we are dealing with time-series, the lag is an offset in time, τ. The lag τ is always selected to be some multiple of the equal-interval time-step used for sampling during data collection (Figure B-1).

For a variable X we calculate the autocovariance for all values X_t with all values $X_{t+\tau}$. The calculation takes the form:

$$cov_\tau = \frac{\sum_{t=1+\tau}^{n} X_t X_{t-\tau}}{(n-\tau)} - \frac{\sum_{t=1}^{n} X_t}{n} \frac{\sum_{t=1+\tau}^{n} X_{t-\tau}}{(n-\tau)}. \qquad \text{(B-7)}$$

When the series is long and the lag short, the means are essentially identical. If the lag is long relative to the length of the series, then the difference between the means is significant. To avoid problems with differences in the means, autocovariance (B-7) values are generally

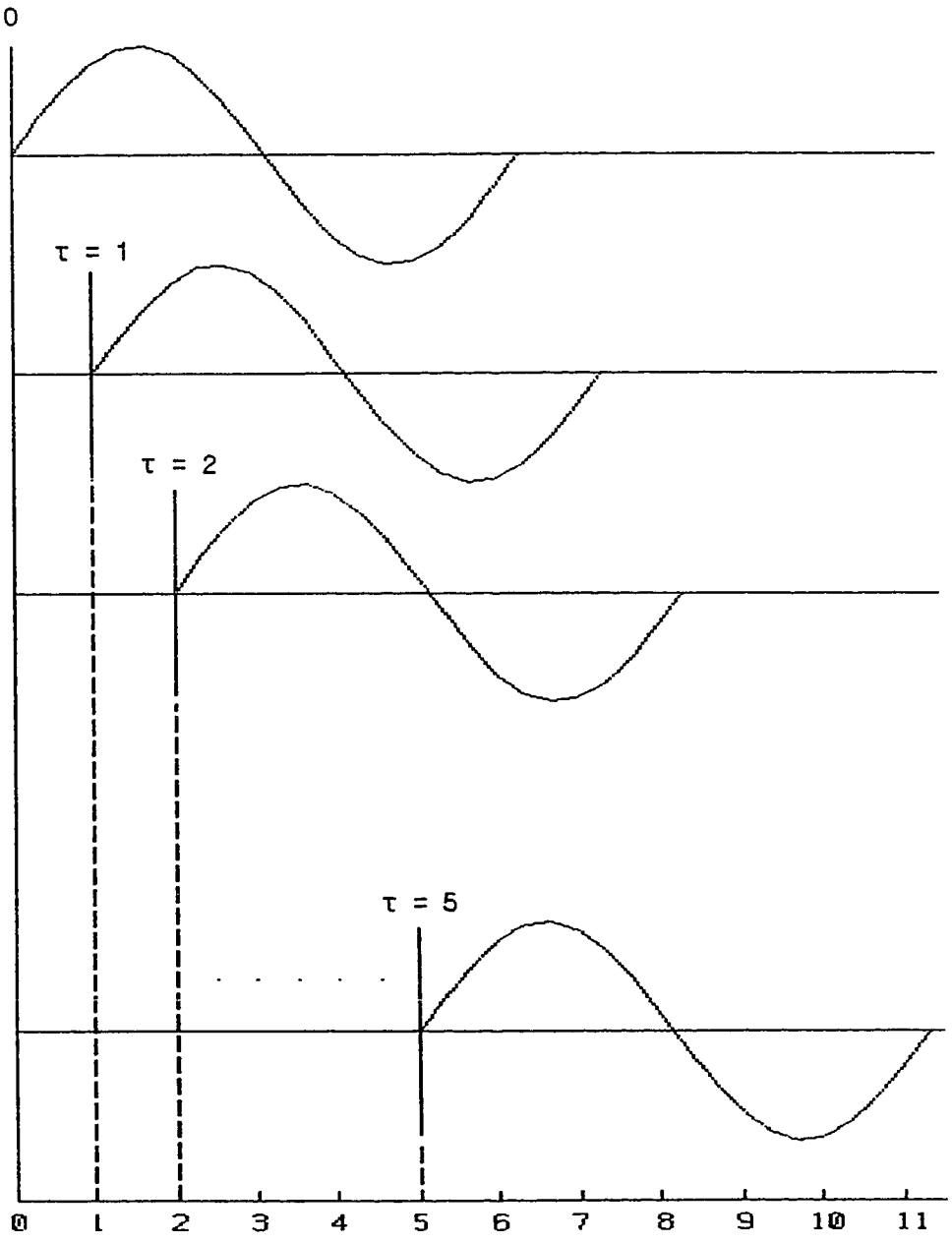


FIGURE B-1: Change in alignment of peaks and troughs due to stepping a copy of a signal at equal-interval time lags, τ, relative to the original signal.

calculated for lags from 0 to n/4, where n is the size of the variable. The results are displayed as a plot of lag against autocovariance-value, called an autocovariogram or autocovariance-function.

From its autocovariance-function, we can determine periodicities within a variable. For lag values that correspond to periods within the variable, the autocovariance-value will be high. For lag values which do not correspond to a period within the variable the autocovariance-value will be low. For example, in the autocovariogram of a semidiurnal tide-record we would expect to see indications of periodicity at lags near 12.5 hours and 25 hours. Using the program AUTOCOVCOR (Appendix F) with the prepared tide-data of this thesis we can produce an autocovariance-function which shows peaks near those lags. The plot also shows many peaks at lags that are multiples of the expected periodicities (Figure B-2). This is because the tidal-signal is exceptionally periodic. The covariance between the original variable and its lagged version is very high at each increment of those periods, throughout the record.

The unit of autocovariance is the square of the unit of the variable it is calculated for. This means autocovariance is scale-dependent and one cannot always compare two autocovariograms directly. However, if the variable is standardized by subtracting the mean from each element and dividing by the standard deviation, then the series will be in units of standard deviation, and the autocovariance will be in standardized form. The covariance of a standardized variable is the correlation, and the

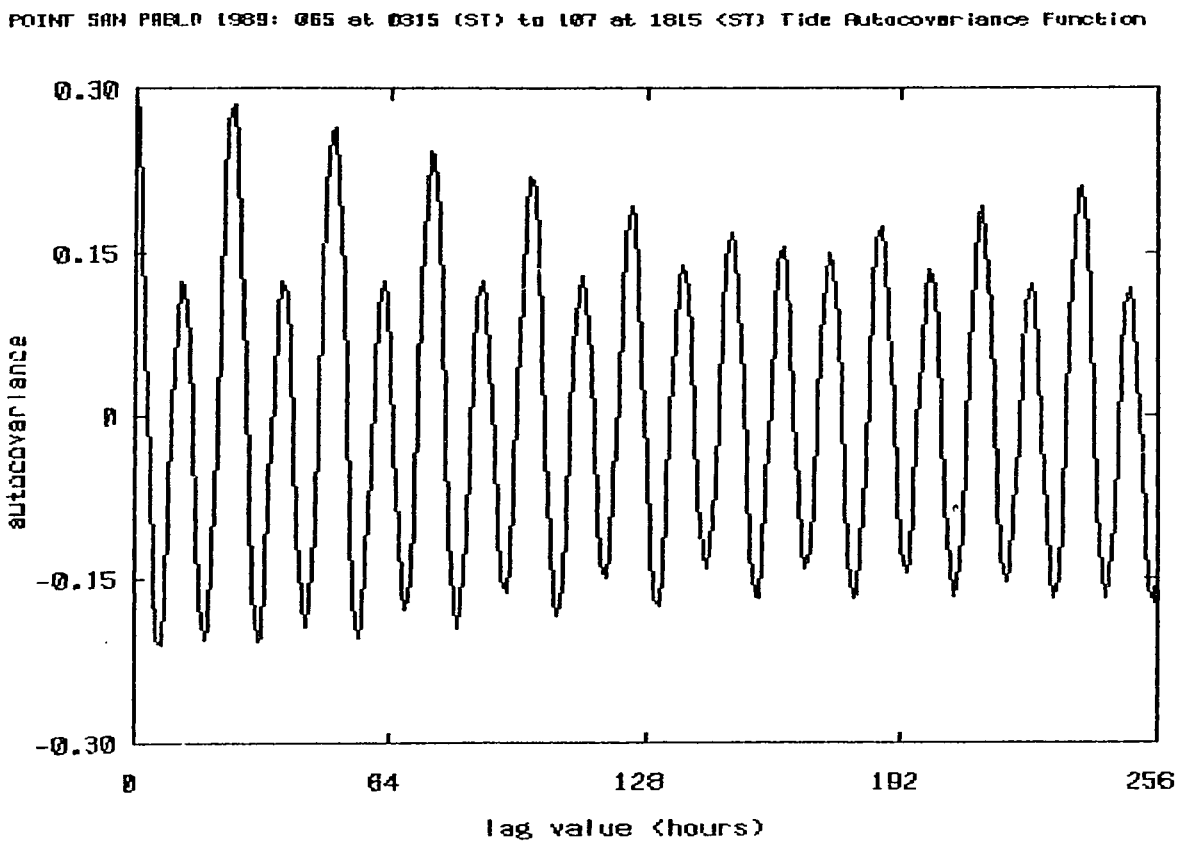


FIGURE B-2: Autocovariance-function, (lags 0 to 256 hours), for tidal excursion data from the Point San Pablo water quality monitoring station, 06 March 1989 at 0315 (ST) to 17 April 1989 at 1815 (ST).

autocovariance of a standardized variable is the autocorrelation. Rather than standardizing the variable first, the autocorrelation may be calculated by dividing the autocovariance by the variance of the variable. That is:

$$r_\tau = \frac{cov_\tau}{var(X)} = \frac{\sum_{t=1+\tau}^{n} X_t X_{t-\tau} - (\frac{1}{n})\sum_{t=1}^{n} X_t \; (\frac{1}{n-\tau})\sum_{t=1+\tau}^{n} X_{t-\tau}}{\sum_{t=1}^{n}(X_t - (\frac{1}{n})\sum_{t=1}^{n} X_t)^2}. \qquad (B\text{-}8)$$

A plot of lag versus autocorrelation (B-8) is a correlogram. A correlogram is actually two-sided, having a negative-half that is the mirror image of the positive side. In general we ignore the negative-half of the correlogram.

A correlogram is compared to correlograms of idealized models in order to determine characteristics of the time-series it was derived from. The simplest model is that the elements of a time-series are independent and normally-distributed. In this case there is no relationship between an observation at time t and any other observation at time $t + \tau$. The value of the correlogram at lag 0 is 1, and plots as the horizontal line $r = 0$ for all other lags. A sinusoidal model will produce a sinusoidal correlogram. The value of autocorrelation for a sinusoid is +1 when the lag results in an alignment of peaks in the original-series and the lagged-series. The value is -1 when the lag results in an alignment of peaks with troughs. There are many possible models. In general they are built

up from combinations of simpler models. A representation of a complex time-series can be formed in this way.

We can look at the results of producing a correlogram for combinations of simple models using the primary-constituent data constructed for the example factor analysis problem presented in Appendix C of this thesis, plus a linear trend-line. The three constructed waveforms and the trend-line are 41 points long, have zero mean, and range from -1 to +1 (Figure B-3). Constituent-1 is a sinusoid with a period of 10 units. Its correlogram shows a maximum autocorrelation at lag-values of 1 and 10, (alignment of peaks with peaks), and a minimum autocorrelation at a lag-value of 5 units, (alignment of peaks and troughs). Constituent-2 is a partial-cycle of a sinusoid with its trailing-end 'flattened.' The period of the full sinusoid is approximately 82 units. Its correlogram is a curve that shows decreasing autocorrelation with increasing lag. The autocorrelation value for constituent-2 is greater than 0 for all lags out to n/4 = 10. Constituent-3 represents a (theoretically) random, (i.e, independent), normally-distributed set of values. Its correlogram takes a form close to that described in the previous paragraph for such a set. The trend-line is just a positive-sloped line which has a point-value of -1 at n = 1 and +1 at n = 41. Its correlogram is a negatively-sloped line. As lag values increase the correspondence between the trend-line and its lagged copy always decreases. Note the similarity between the correlograms of constituent-2 and the trend-line. Since over its span constituent-2

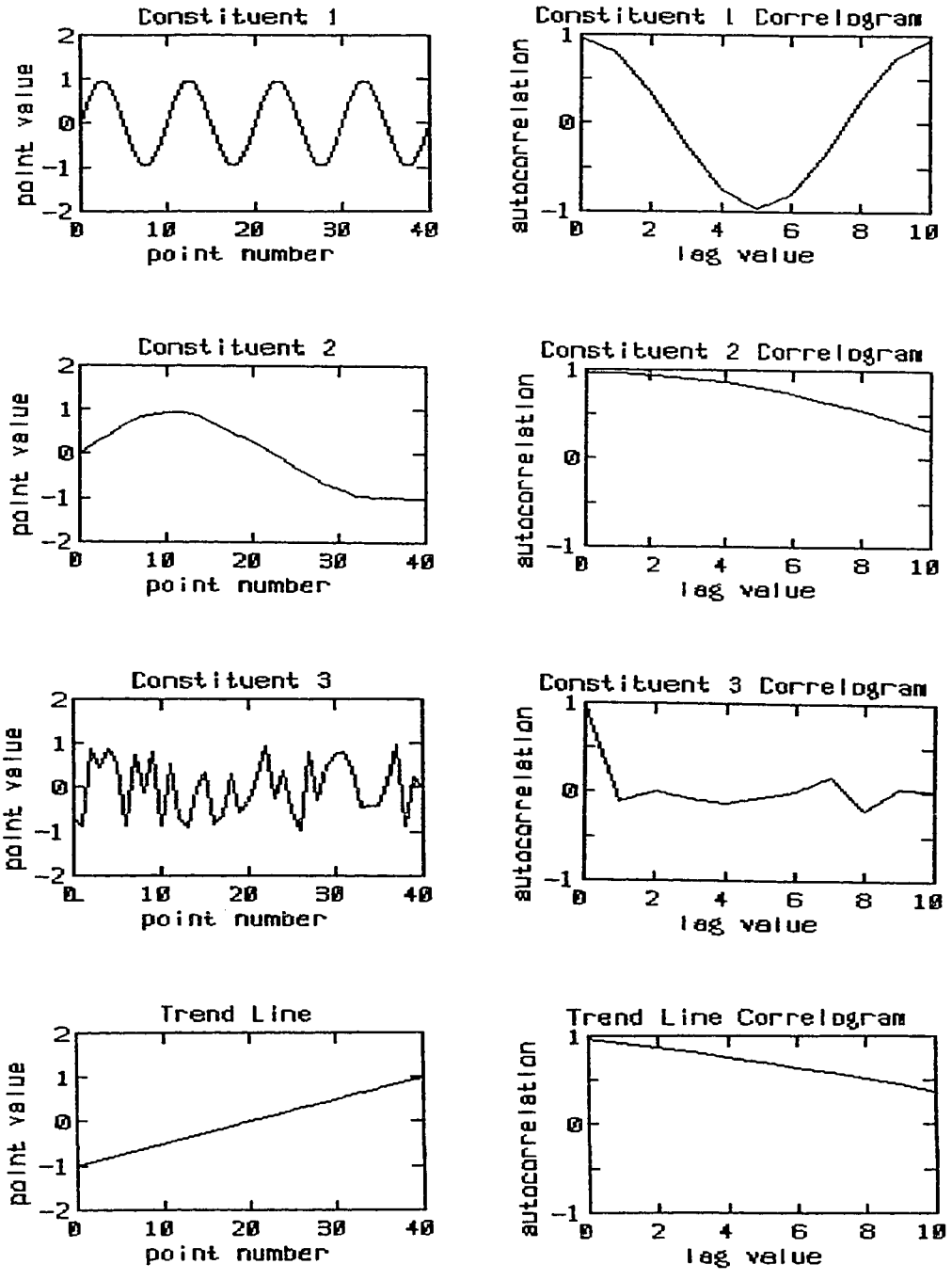


FIGURE B-3: Constituents from the factor analysis example problem (Appendix C), a trend line, and their correlograms.

contains less than one cycle, its addition to a signal will appear more as a trend than as an added periodic-component.

The sum of constituent-1 with the trend-line is a version of constituent-1 that slopes up to the right, (i.e., has a positive trend). The correlogram of this sum is similar to the correlogram for constituent-1 alone. However, the correlogram of the sum shows lower autocorrelation-value for increasing lag-value. This is the effect of a trend (Figure B-4). The horizontal-relation of peaks and troughs is identical for constituent-1 alone and for constituent-1 summed with the trend-line. However, as the lag-value is increased for the summed waveform, the vertical-correspondence between peaks and troughs in the original sum and its lagged counterpart becomes increasingly poor. This decreases the autocorrelation values. As was pointed out in the previous paragraph, constituent-2 should affect a sum of constituents very much like a trend-line. The sum of constituent-1 with constituent-2 looks like a copy of constituent-1 with a negative-trend a and a slight 'wave.' The correlogram of this sum looks very much like the correlogram for the sum of constituent-1 with the trend-line. There is some negative-correlation at lag-value 5, (alignment of peaks with troughs), for the sum of constituent-1 with the trend-line. For the sum of constituent-1 with constituent-2 the correlogram shows a value of about 0, or that there is no linear-dependence, at that lag value. This is in-part due to the 'wave' in the second sum. But, this appearance could be produced in the correlogram for the first sum by increasing the slope of the trend-line.

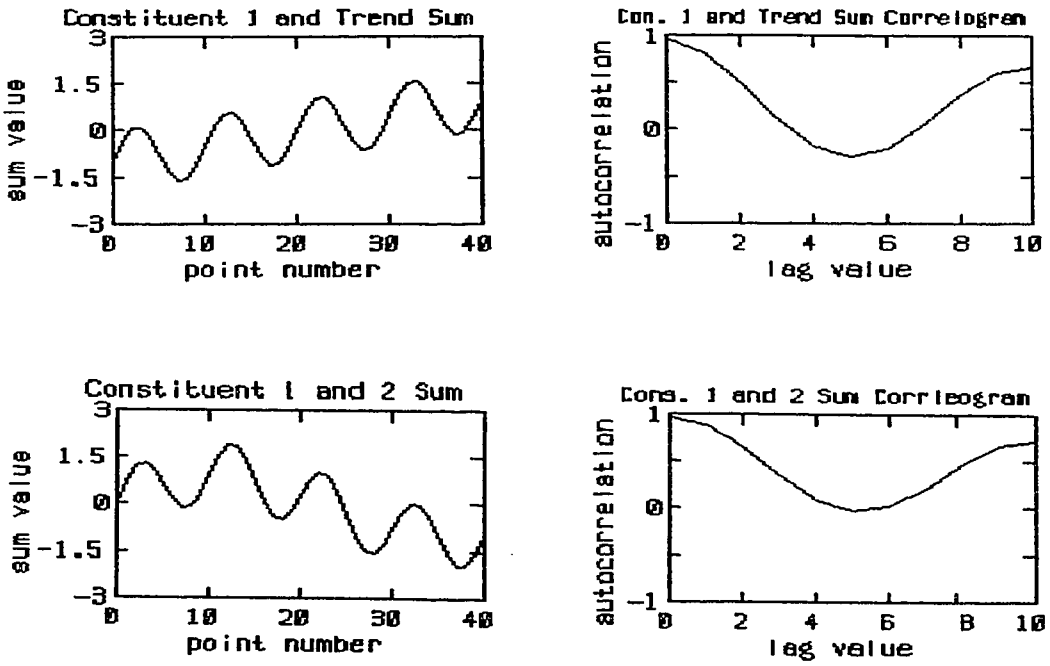


FIGURE B-4: Sums of constituent 1 with constituent 2 and with trend line, and sum correlograms.

This would cause the vertical-relationship to change more rapidly with increasing lag-values. Note that the correlogram gives some indication of the slope of a trend, but does not differentiate between positive or negative trends. The slope of the trend-line is positive. The implied slope from constituent-2 is negative. The correlogram for either sum only shows a decreasing correspondence with increasing lag-value.

A time-series signal whose properties do not change with time is said to be stationary. If a time-series is divided into small segments, and the means of the segments tend to be the same as the mean of the entire series, then that series is first-order-stationary. We remove linear-trends from data (see Appendix A) in an attempt to force a time-series to be first-order-stationary and reduce the affects of long-term trends in the data on spectral analysis. Combining constituent-1 with constituent-3 shows the effect of 'noise' on a signal. The periodicity of constituent-1 can be seen in the correlogram of the sum, but is slightly masked by the 'noise' of constituent-3; evidenced by the decreasing autocorrelation-values with increasing lag-value (Figure B-5). Adding constituent-3 to the sum of constituent-1 with the trend-line and to the sum of constituent-1 with constituent-2 shows another effect of injecting noise into a signal. In this case a beneficial effect. The correlograms for these sums without the addition of noise (Figure B-4) show very similar forms, but notably different autocorrelation-values. Correlograms for these same sums with the addition of noise (Figure B-5) show not only very similar forms, but a much closer correspondence between

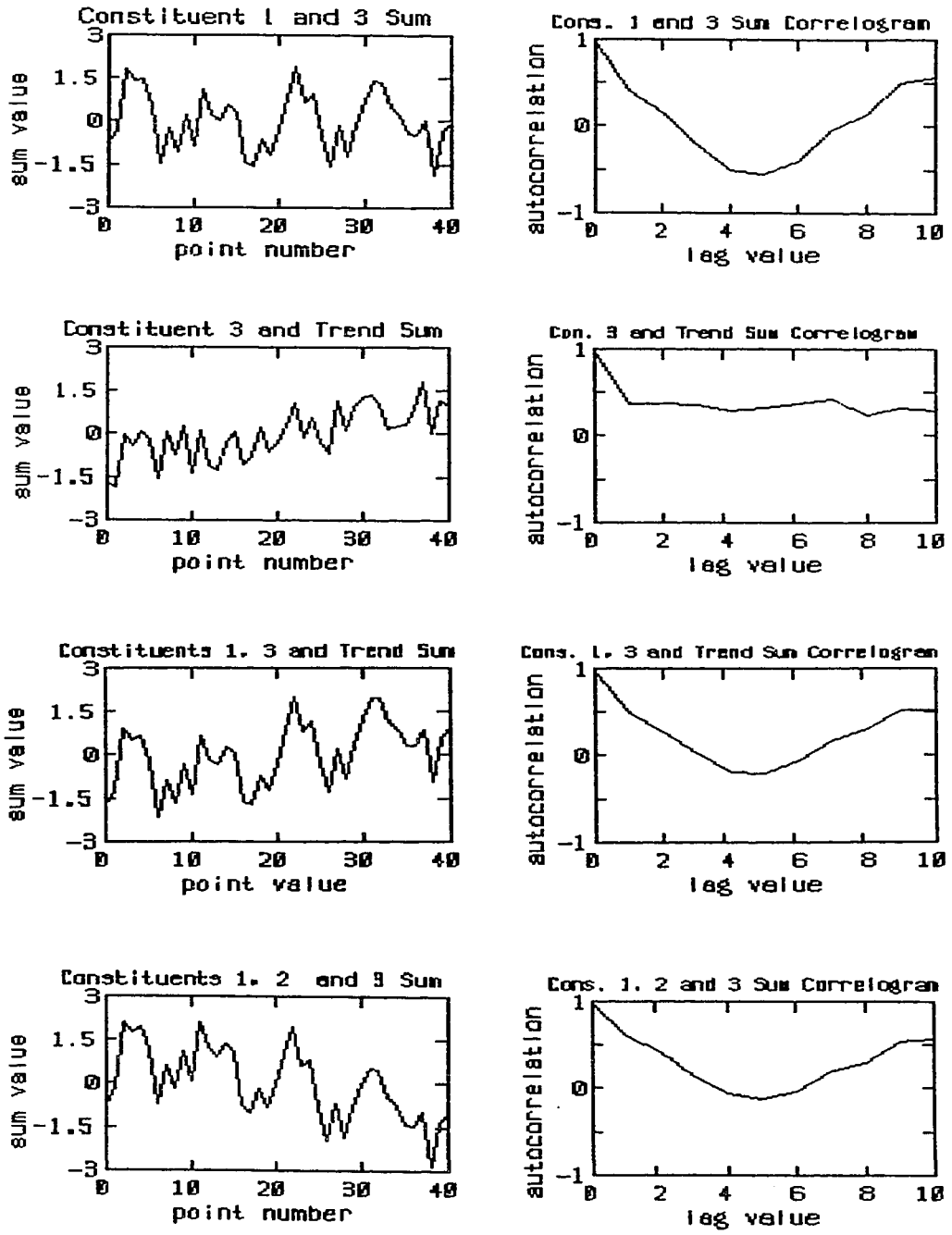


FIGURE B-5: Sums of constituents 1,2 and trend-line with constituent 3 (noise), and sum correlograms.

autocorrelation-values. This might be considered an (extreme) example of prewhitening. The judicious injection of noise into a signal (prewhitening) is a technique that may be used to reduce the affect of trends and undesired noise in a data-set. More formally, prewhitening is employed in the frequency-domain and involves amplification of high-frequencies and reduction of low-frequencies in the spectrum of a signal. The purpose is to 'flatten' the spectrum. (White-noise has a very flat spectrum, hence the term prewhitening.) The inverse process is post-darkening (Rayner 1971). A flat spectrum has several properties that make it desirable for signal-analysis. However, prewhitening was not employed in this thesis, and we will not discuss it further. It is used extensively in analyzing data from seismic-survey records.

Cross-Correlation

We can compare two separate series at different lags. This is cross-correlation. From such a comparison we can determine how strongly two series are related, and the lag between the series at the position of closest match. There are a number of difficulties with cross-correlation. It is often not possible to determine a zero-lag position, because either series may lead the other. Also, since the two series are not identical, a cross-correlogram is not symmetric, but depends upon to which series the lag shifts are applied. It may also be that the series are not the same length.

The equation for cross-correlation has the same form as the formula for the linear correlation-coefficient. If, for a variable, X_1, of size n, we use the notation:

$$\Sigma X_1 = \sum_{i=1}^{n} X_{1i}, \qquad \text{(B-9)}$$

and the notation n* to mean the number of overlapped positions between two series, X_1, X_2, then the cross-correlation for lag m between the two series is:

$$r_m = \frac{COV_{1,2}}{s_1 s_2} = \frac{n^* \Sigma X_1 X_2 - \Sigma X_1 X_2}{((n^* \Sigma X_1^2 - (\Sigma X_1)^2)(n^* \Sigma X_2^2 - (\Sigma X_2)^2))^{1/2}}, \qquad \text{(B-10)}$$

where, $COV_{1,2}$ means the covariance of the overlapped segments of XX and X_2, s_1 and s_2 are the corresponding standard-deviations, and the summations extend only over the portions of the series that overlap at lag-position m. The cross-correlogram is plotted as match-position, m, versus cross-correlation value, r_m (B-10).

The significance of the cross-correlation coefficient can be determined from the approximate test:

$$t = r_m \frac{(n^* - 2)^{1/2}}{(1 - r_m^2)^{1/2}}, \qquad \text{(B-11)}$$

which has (n* - 2) degrees of freedom (Davis 1973). This test is derived from the F-test for the significance of correlation between two samples

drawn from normal-populations. We expect a test-value of zero at any match position m if the two series are independent and random.

Using the program CROSSCOR (Appendix F), with two copies of a sinusoid as input, to calculate cross-correlation and significance values, and then plotting lag-value versus the results (Figure B-6), we see in this case the cross-correlogram is identical to the autocorrelation-function for the sinusoid. The significance-value plot has the same period as the cross-correlogram but emphasizes the lags of maximum and minimum correspondence. If we perform these operations using the same sinusoid and a copy with random-noise injected, the resulting cross-correlogram looks virtually identical to the cross-correlogram for the two clean sinusoids, (its amplitude is actually slightly less than the amplitude of the cross-correlogram for the two clean sinusoids (Table B-1)). However, while the significance-value plot has the same period as the cross-correlogram, its amplitude is notably smaller than the significance-value plot for the two clean sinusoids (Figure B-7). The cross-correlogram shows the correspondence between the signals with respect to period. The significance-value plot shows that while there is a strong periodic-correspondence between them, the waveforms are not identical.

Using the upper-sensor temperature and salinity records from the data set of this thesis as input to the program CROSSCOR we can produce values for a cross-correlogram and a significance-value plot showing the correspondence between these two signals. The n/4 lag

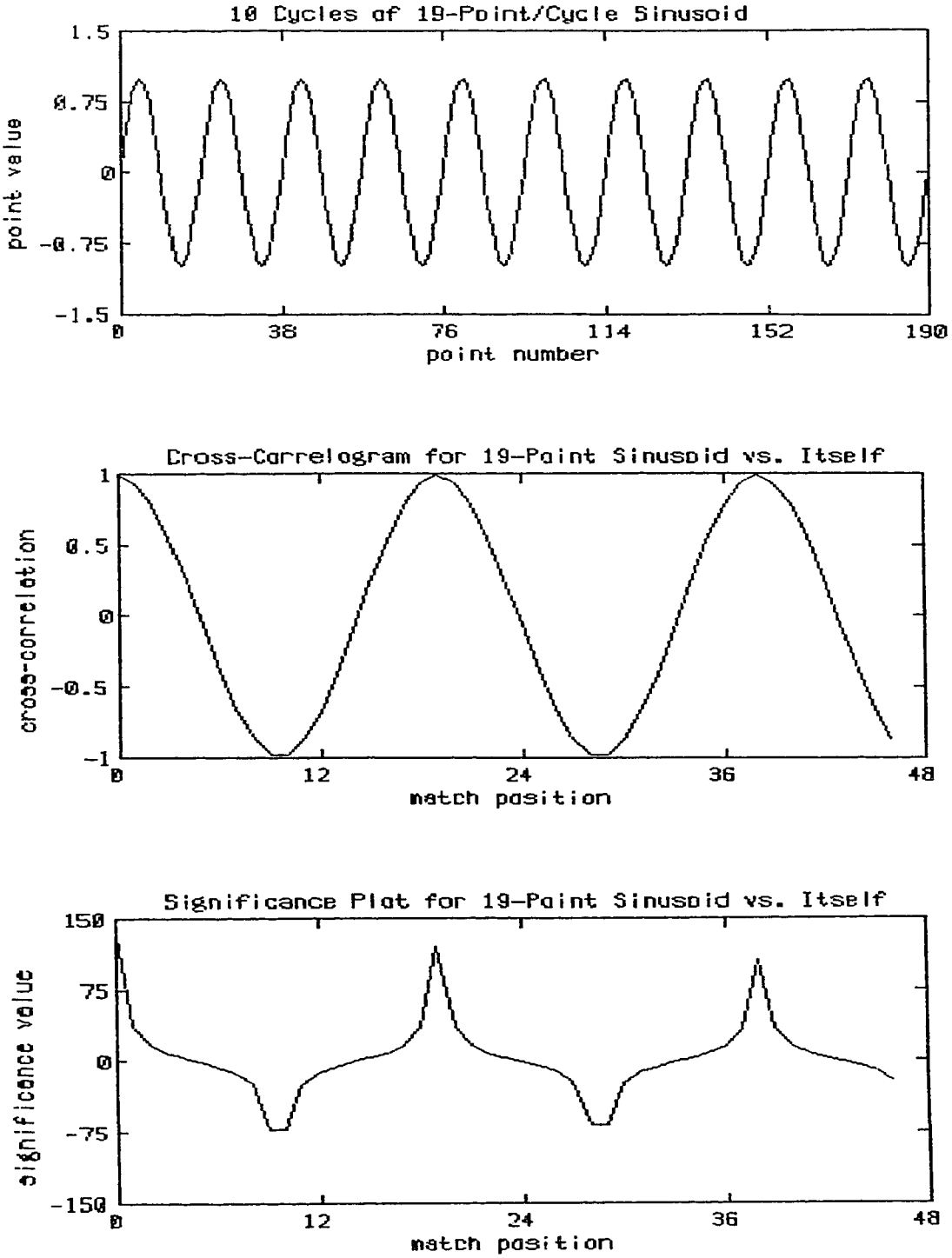


FIGURE B-6: 10 cycles of a 19-point/cycle sinusoid, cross-correlogram of the sinusoid with itself, and significance-plot.

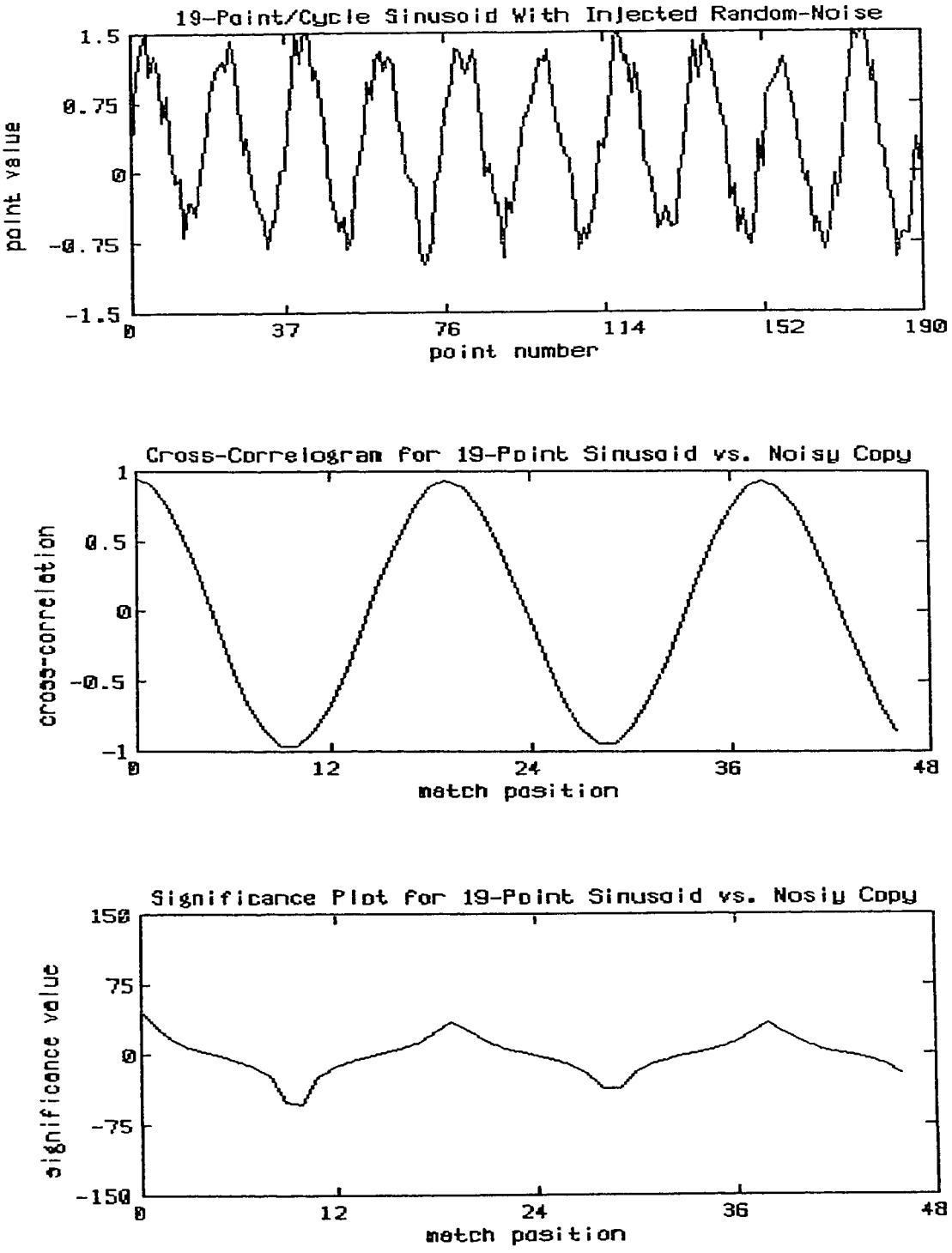


FIGURE B-7: 10 cycles of a 19-point/cycle sinusoid with injected random noise, cross-correlogram of the clean sinusoid versus the noisy sinusoid, and significance-plot.

TABLE B-1: Cross-correlation values at lags of maximum and minimum correspondence for 10 cycles of a 19-point per-cycle sinusoid against itself, and for a copy of the sinusoid with random-noise injected against the clean sinusoid.

Lag-Value	Cross-Correlation-Value	
	Clean/Clean	Noisy/Clean
0	0.995	0.959
10	-0.983	-0.970
19	0.994	0.937
29	-0.983	-0.943
38	0.993	0.938
47	-0.982	-0.965

value for these data is 256 hours (Figure B-8). However, these signals are predominantly tide-driven; hence strongly periodic near 12.5 and 25 hours (Figure B-9). We must consider high-correspondences at lags greater than 25 hours to be related mainly to cycling of the tide itself. Repetition of the form of the cross-correlogram from lag 0 to about lag 25, starting near lag 25 and ending near lag 50 supports this assertion. In any case we should ignore indications of high cross-correlation at long lag-values due to the corresponding short-overlap between the variables. Here we will examine cross-correlation and significance for lags out to 60 hours (Figure B-10).

The cross-correlation was made for salinity versus temperature. That is, the temperature-record was moved to the right past the salinity-record for each lag-value. Thus, lag-values for high

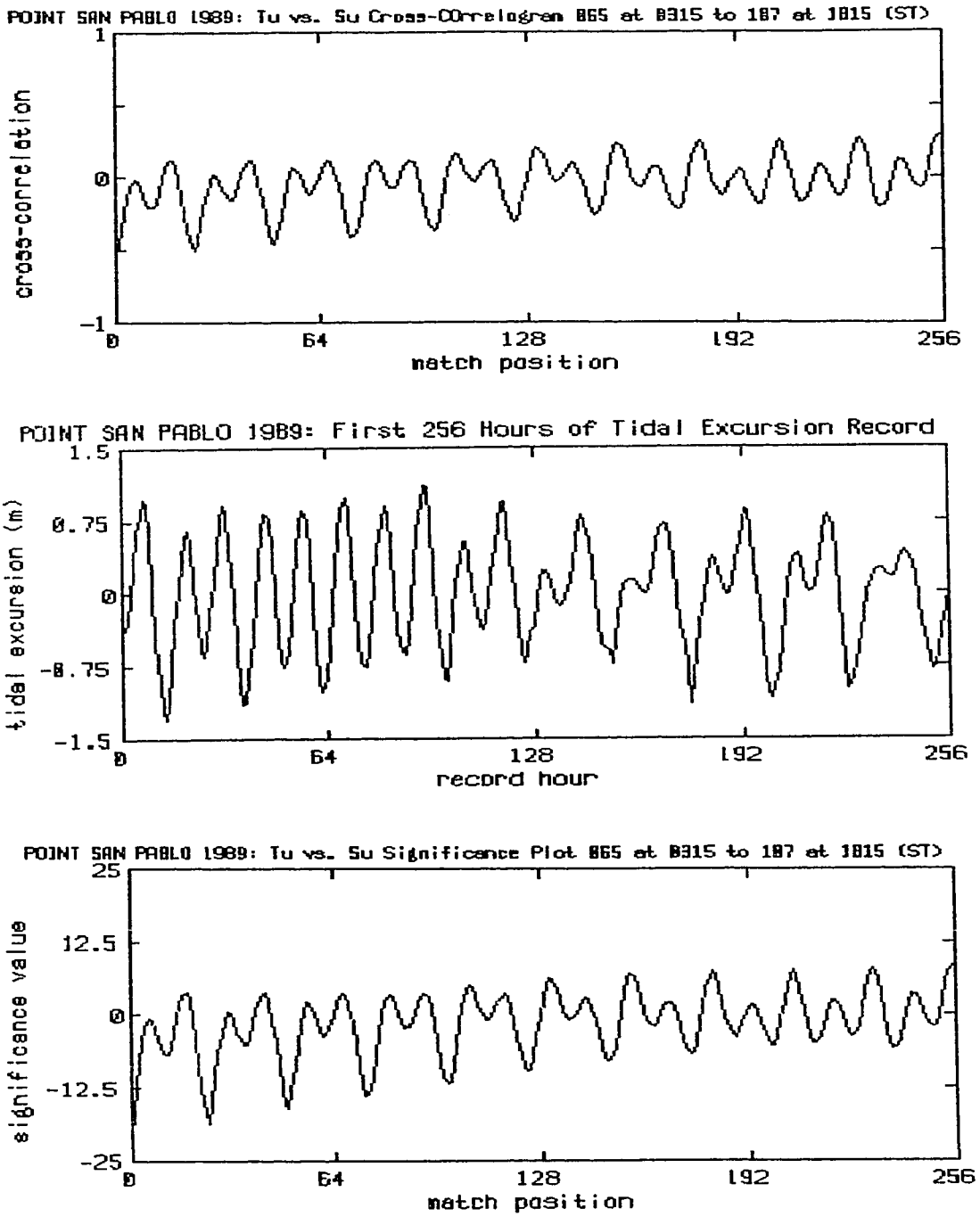


FIGURE B-8: Cross-correlogram for Su versus Tu, significance-plot for Su versus Tu for lags n/4 = 256 hours, first 256 hours of the tidal-excursion record.

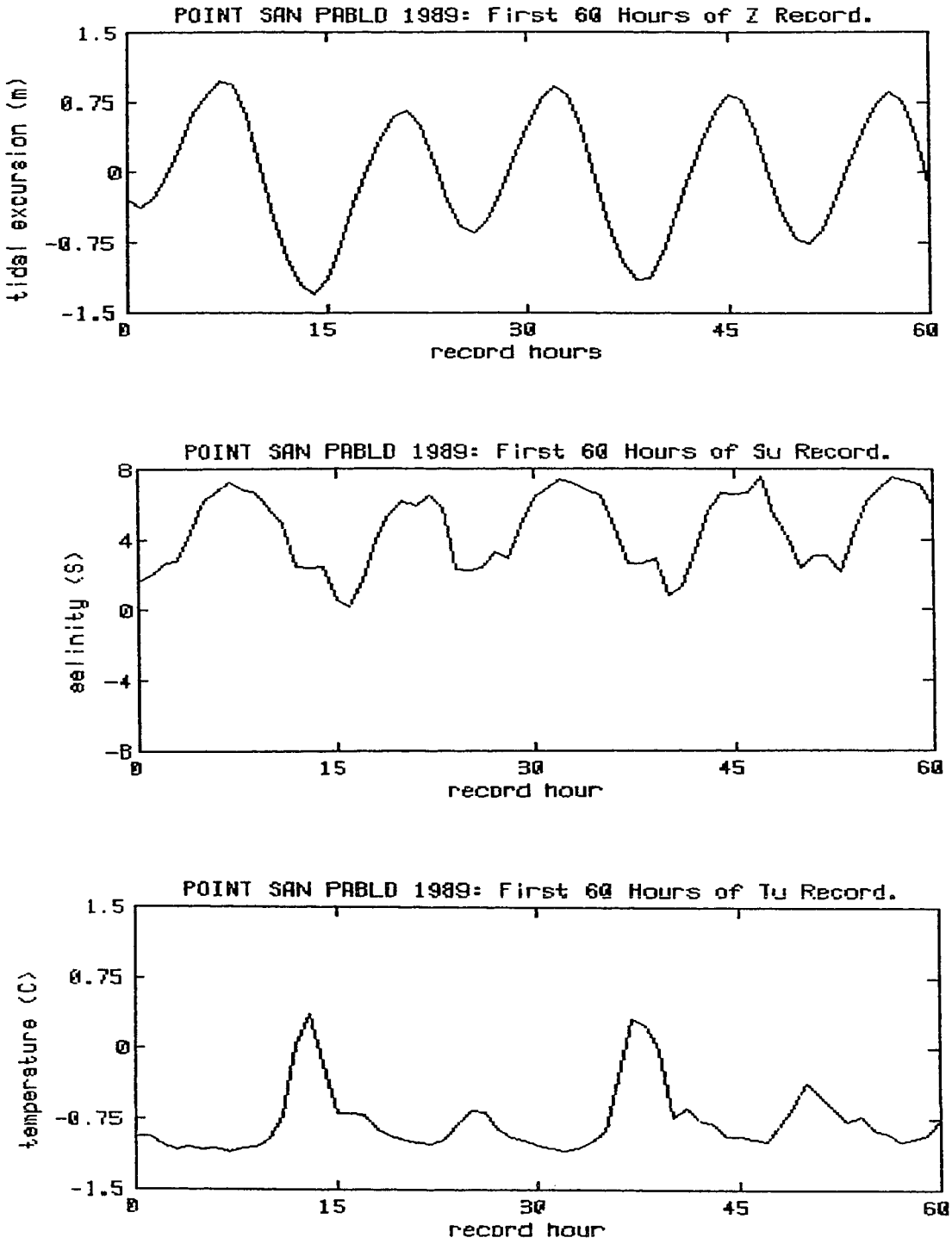
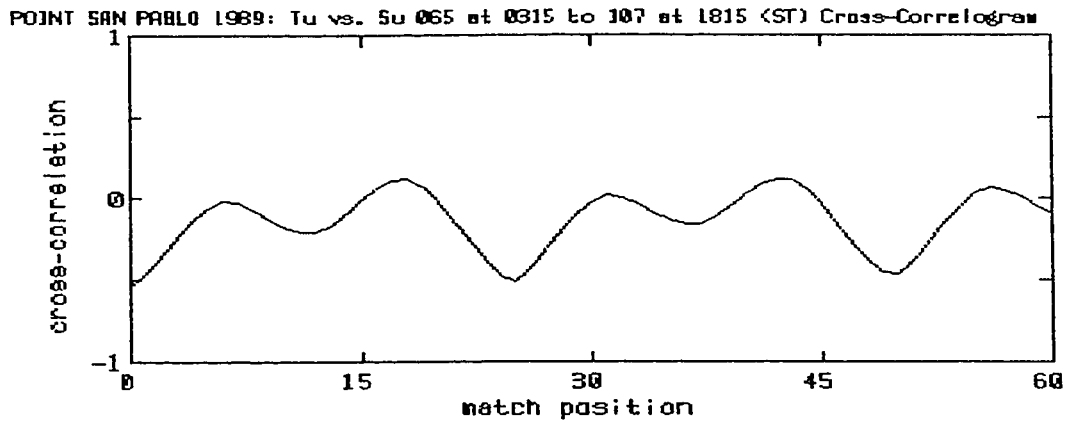


FIGURE B-9: First 60 hours of tidal-excursion, Su, and Tu.



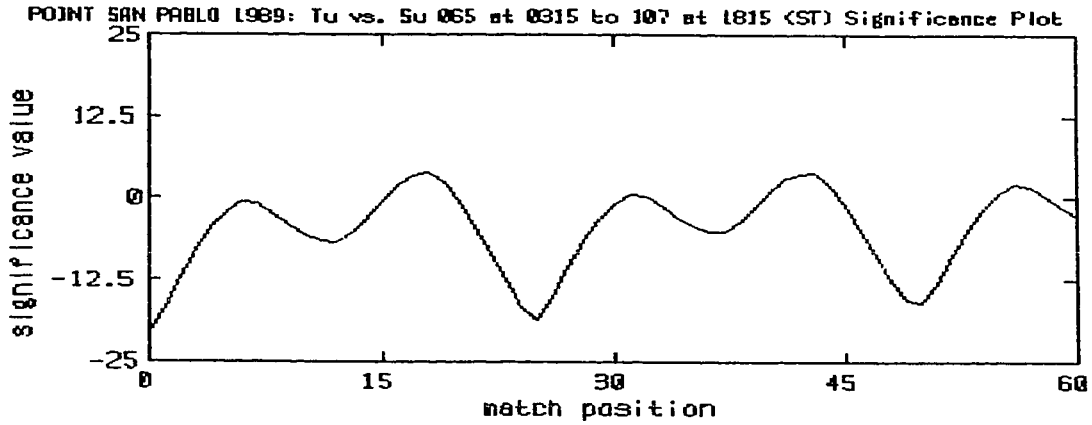


FIGURE B-10: Cross-correlogram and significance-plot for Su versus Tu.

correspondence indicate periods by-which the temperature-signal leads the salinity-signal. Peaks in the cross-correlogram occur at lag-values of 6 and 18 hours. This says that there are peaks in the salinity signal 6 and 18 hours after peaks in the temperature signal. The cross-correlation value for the 6-hour lag is lower than the value for the 18-hour lag. This says that there is a higher correspondence for the 18-hour period between peaks in the records than there is for the 6-hour period between peaks. We see these correspondences in an overlay plot of the first 60 hours of data (Figure B-11). There are 2 major peaks in the temperature record separated by about 25 hours. Each major peak is followed in about 12 hours by a minor peak. The major and minor peaks in temperature correspond well with the four troughs between the five peaks in the salinity-record, which are separated by about 12 hours. Sliding the temperature-record 6 hours to the right leaves its major-peaks still partially within the troughs of the salinity-record, but places the minor-peaks under peaks in the salinity record. Thus there is some increase in correspondence in the records at a lag of 6 hours. Moving the temperature-record another 12 hours to the right places the major and minor peaks in temperature under peaks in salinity, further increasing correspondence between the records at the 18-hour lag.

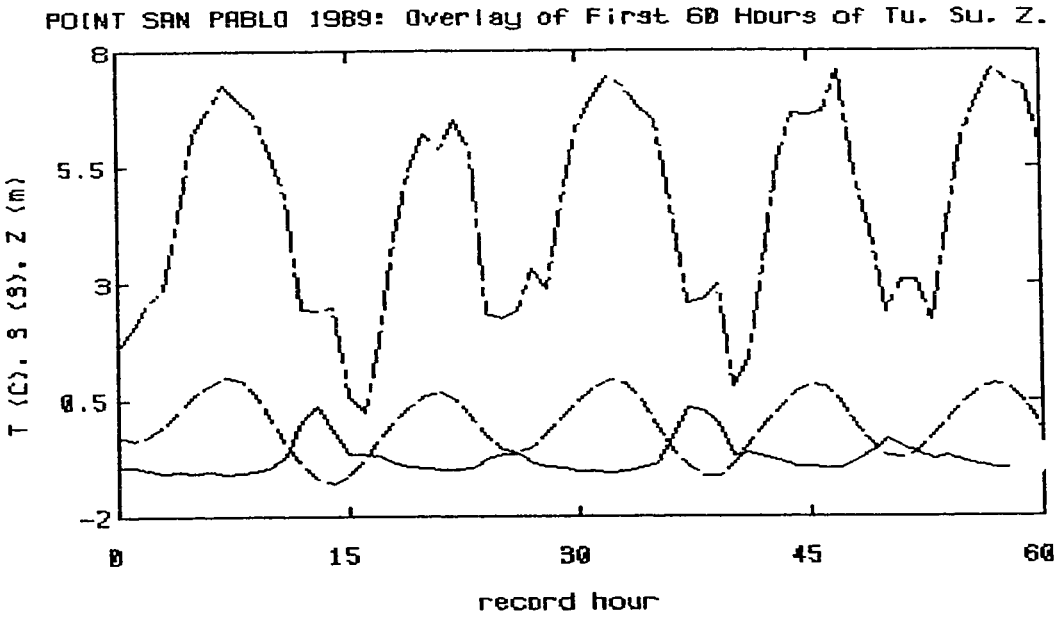


FIGURE B-11: Overlay of first 60 hours of Tu, Su and tidal-excursion.

Harmonic Analysis

Introduction

The idea behind spectral-analysis is to take a function, x(t) and express it as a linear-combination of preselected functions, $y_i(t)$, such that:

$$x(t) = \sum_i X_i y_i(t), \qquad \text{(B-12)}$$

where the values X_i represent the 'importance' of the components, $y_i(t)$, in forming the sum for x(t). The X_i may be considered 'weights' for the $y_i(t)$ components in the sum for x(t) (B-12). When values for the X_i are plotted as vertical-lines along a horizontal-axis with-respect-to i, they display the spectrum of x(t). The spectrum, X, through its spectral-values, X_i, tells us the relative importance of the spectral-components, $y_i(t)$, in forming the signal, x(t), and, hence, something about the nature of x(t) itself.

For a spectrum, X, to tells us something about a signal, x(t), the components, $y_i(t)$, should have some relation to the underlying-structure of x(t). In general, we have no knowledge of the underlying-structure (true components) of a signal. We assume these components are sinusoids. For most 'real' signals this is a reasonable approximation. For tidal and related phenomena it is quite good. Components, $y_i(t)$, are

sinusoids (sine and cosine). The subscript, i, gives the frequency of component-i. Generation of a spectrum, X, involves separating a signal, x(t) into its frequency-components, $y_i(t)$. This is analogous to separating a beam of white-light into its spectral-components using a prism.

In the introduction to this appendix, we identified harmonic analysis as a pseudonym for spectral analysis. Here, we use harmonic analysis to refer to a specific subset of spectral analysis: separating harmonically related sinusoidal components from a signal. Harmonically related means that the frequency of any component is an integer-multiple of the frequency of a single fundamental (lowest) component. Below we will refine this definition to encompass separation of harmonically related components from discrete uniformly-sampled signals.

Fourier Series

The French mathematician J.B.J. Fourier (1768-1830) discovered that any periodic-function can be described by an infinite-series of harmonically related frequencies. Before discussing an infinite-series of them, let us first look at a single sinusoid. A single-frequency sinusoidal signal with a particular energy-content can be pictured in the time-domain as a plot of the cosine (or sine) function with proper parameters. The input to the function relates to frequency of the signal, and energy contained in the signal is expressed in the amplitude-

factor applied to the function. Cosine evaluated at 0 is 0. Adding a fixed value to the variable input of the function allows us to shift the position of the initial function-zero way from the axis-zero. The range of cosine is from -1 to +1. We may multiply by a factor, A, and change the amplitude to range from -A to +A. Varying its input from 0 to 2π gives one cycle of the cosine function. The period, T, of a function is the time it takes to complete one full-cycle. Using time, t, as the input-variable to cosine, we may adjust the period of our plotted sinusoid to any number of time-units by applying the factor $2\pi/T$ to the input. With T as period, A the amplitude, and p the portion of a cycle away from the origin for the initial zero-point, our sinusoid may be expressed as:

$$x(t) = A\cos(\frac{2\pi}{T}t - 2\pi p). \qquad \text{(B-13)}$$

Period, T, is the number of time-units per cycle. So, 1/T is the number of cycles per time-unit, or frequency, f, of a sinusoid. $2\pi/T = 2\pi f$ is the radian-frequency of the sinusoid. The term radian refers to a measure of angle in terms of rotation. One full-cycle is 2π radians. The radian is associated with π, which is a proportion relating the radius of a circle to its circumference, and therefore unitless. So, the unit of radian-frequency is inverse-time, and multiplying radian-frequency by time gives some number of π radians. If p in the above equation (B-13) is 0, and we start at time t = 0, then at t = T, $(2\pi/T)t = 2\pi$, and one full-cycle is complete. If p is not 0, the first zero of the sinusoid is at time t = Tp, and one full-cycle is completed at time t = T + Tp. The value $2\pi p$ is the

phase-angle. Its measure is in radians (Figure B-12). If we use the notation $\theta = 2\pi/T$ and $\phi = 2\pi p$, we can express a family of sinusoids as:

$$x_k(t) = A_k \cos(k\theta t - \phi_k), \qquad \text{(B-14)}$$

where k, used as a multiplier and a subscript, identifies a particular member of the family. Note the radian-frequency of all members of this family is some multiple, k, of θ, k=1,2,... This family of sinusoids is harmonically related to the fundamental-frequency $\theta = 2\pi/T$, where k is the harmonic-number for a member of the family. The harmonic-number gives the number of cycles for a member sinusoid that occur within a time-period equal to the period, T, of the fundamental. The phase-angle, ϕ_k, may or may not exist and may or may not be different for each member of the family. Different phase-angles mean a different (fixed) portion, p, of a cycle in $\phi_k = 2\pi p$, for different harmonic-values, k.

Using the trigonometric-identity for the cosine of a difference:

$$\cos(X-Y) = \cos Y \cos X + \sin Y \sin X, \qquad \text{(B-15)}$$

with the expression for our harmonically related family (B-14), we produce the equivalent statement:

$$x_k(t) = A_k \cos(\phi_k)\cos(k\theta t) + A_k \sin(\phi_k)\sin(k\theta t). \qquad \text{(B-16)}$$

The A_k and $\cos(\phi_k)$ are parameters which are constant for each k.

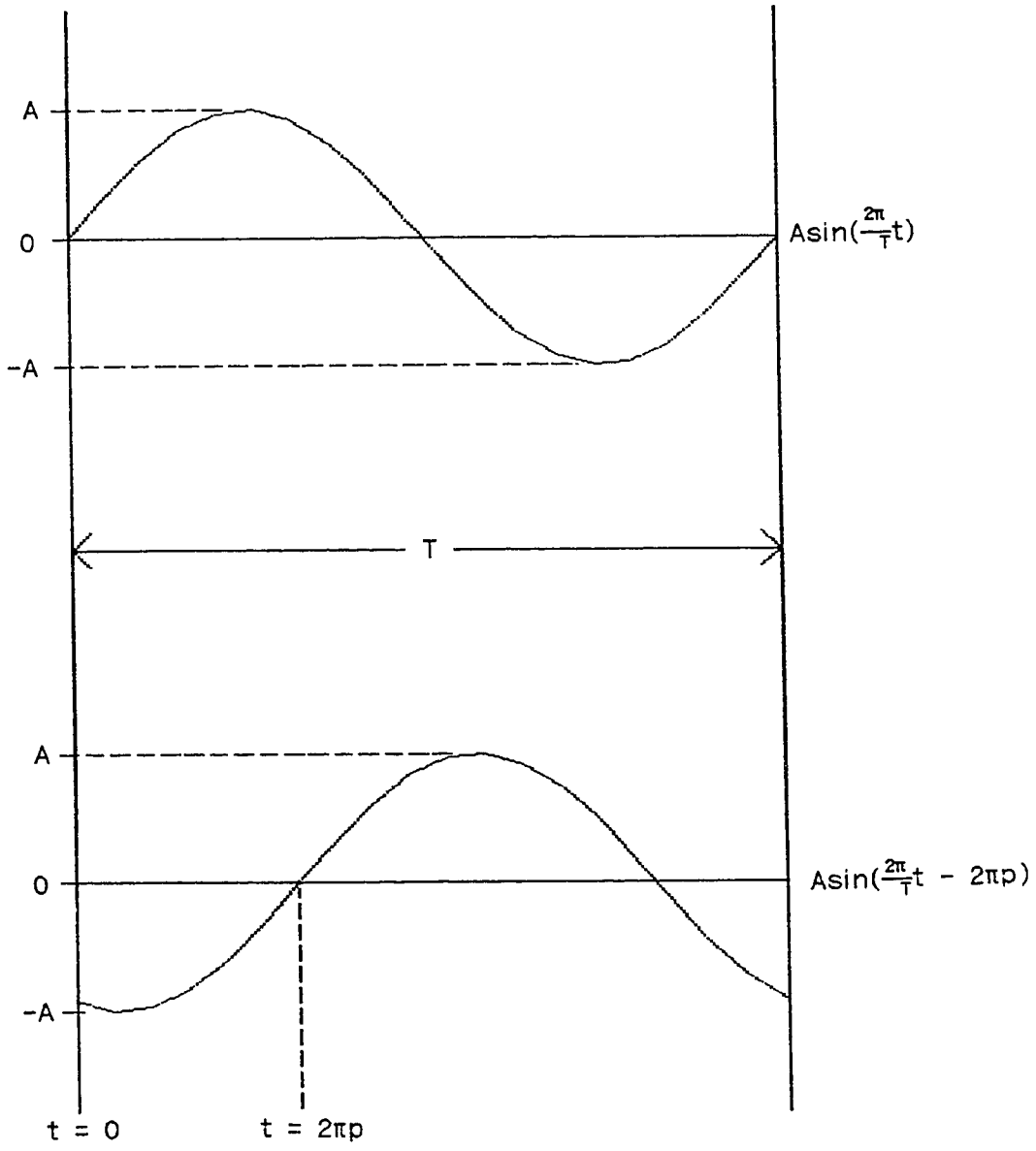


FIGURE B-12: Diagram of effect on a sinusoid of change in phase angle, $2\pi p$, $0 \leq p \leq 1$.

Employing the notation:

$$\alpha_k = A_k \cos(\phi_k), \qquad \text{(B-17)}$$

$$\beta_k = A_k \sin(\phi_k), \qquad \text{(B-18)}$$

in our equivalent statement (B-16) we obtain:

$$x_k(t) = \alpha_k \cos(k\theta t) + \beta_k \sin(k\theta t). \qquad \text{(B-19)}$$

If we sum our latest equation for a member of our harmonically-related family (B-18) from 0 to infinity, noting that cos(0) = 1 and sin(0) = 0, the expression of the Fourier-series emerges as:

$$x(t) = A_0 + \sum_{k=1}^{\infty} \alpha_k \cos(k\theta t) + \beta_k \sin(k\theta t). \qquad \text{(B-20)}$$

Any continuous time-series, x(t), which has only one value of x(t) for each time t, can be expressed as a Fourier-series sum (B-20). Harmonic-analysis is the process of comparing harmonically related sinusoids to a data set and retaining for the spectrum those frequencies that 'fit' the data-set within some predefined limit. The usual criterion is minimum mean-square-error, as determined by a least-squares procedure used to solve for the coefficients α_k, β_k. If very many harmonics are to be determined, a matrix-solution for the Fourier-coefficients, α_k, β_k, problem becomes quite formidable. However, the desired solution for the Fourier-coefficients is well known. They may be determined as:

$$A_0 = \frac{1}{T}\int_0^T x(t)dt, \qquad \text{(B-21)}$$

$$a_k = \frac{2}{T}\int_0^T x(t)\cos(k\theta t)dt, \qquad \text{(B-22)}$$

and,

$$\beta_k = \frac{2}{T}\int_0^T x(t)\sin(k\theta t)dt. \qquad \text{(B-23)}$$

Note the term A_0 is simply the mean of the signal x(t). If we have a periodic, discrete uniformly-sampled time-series, X, of length n, $X = \{x_m\}$, $m = 1,\ldots,n$, then the a_k, β_k Fourier-coefficients can be determined from the formulas:

$$a_k = \frac{1}{k}\sum_{m=1}^{n} x_m \cos(2\pi(m-1)k/n), \qquad \text{(B-24)}$$

and,

$$\beta_k = \frac{1}{k}\sum_{m=1}^{n} x_m \sin(2\pi(m-1)k/n), \qquad \text{(B-25)}$$

And, the A_0 term becomes the arithmetic-mean:

$$A_0 = \frac{1}{n}\sum_{m=1}^{n} x_i. \qquad \text{(B-26)}$$

Once we have determined its a_k (B-22 or B-24) and β_k (B-23 or B-25) Fourier-coefficients the amplitude of harmonic-k can be computed as:

$$A_k = (a_k^2 + \beta_k^2)^{1/2}, \qquad \text{(B-27)}$$

and its phase-angle as:

$$\phi_k = \arctan\left[\frac{\beta_k}{a_k}\right]. \qquad \text{(B-28)}$$

If a single-frequency sinusoid is uniformly sampled at discrete points, the amplitude of the waveform is related to the variance of the samples. In the limit, (i.e., as the number of samples becomes very large), the sample-variance becomes half-the-square of the amplitude of the waveform:

$$s_k^2 = \frac{A_k^2}{2} = \frac{a_k^2 + \beta_k^2}{2}. \qquad \text{(B-29)}$$

Since the Fourier-series is a linear combination, the variance of the sum must equal the sum of the variances of the harmonic-components. Therefore, the variance of harmonic-k can be expressed as a proportion of total-variance of the time-series. A plot of harmonic-number versus variance of the harmonic-components is a periodogram. The periodogram is sometimes referred to as a discrete-power-spectrum, or a line-power-spectrum. Variance and power are synonymous. The term 'power' was brought forth by electrical-engineers who developed the use of these techniques in analyzing electrical-signals.

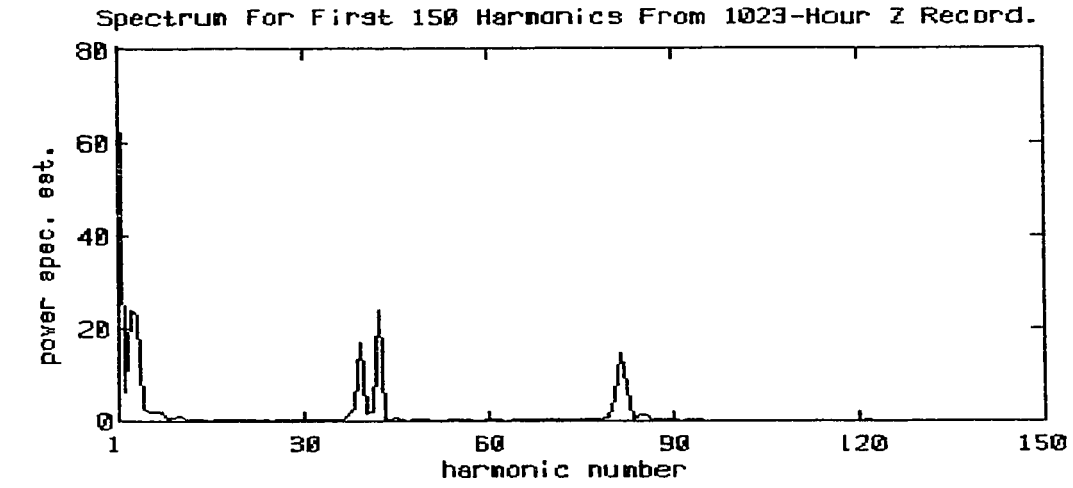
A periodogram as described above is referred to as the raw-spectrum. It is an estimate of the true or population-spectrum. These

estimates have very large standard-error. Some of this error is due to aliasing. That is, the injection of false low-frequencies into the spectrum, due to the presence of unresolvable high-frequencies, i.e., frequencies in the sampled-signal higher than the Nyquist-frequency. The Nyquist-frequency is one-half the uniform sampling-frequency (see Appendix D).

The program HARMONICS (Appendix F) implements the Fourier-coefficient calculations as expressed above for the discrete-uniformly-sampled case. We take the length of the 1024-point, 1-hour interval record (1023 hours) to be the period of the fundamental (harmonic-number k = 1) and divide this period by the harmonic-number of the harmonically-related sinusoids to determine the period of those sinusoids (Table B-2). A periodogram for the first 150 harmonics determined for the tidal-record (Figure B-13) gives us a spectrum showing power over the range of period 1023/1 = 1023 hours to 1023/150 = 6.82 hours. Note the high power indications for the longest periods (smallest harmonic-numbers) relative to the shorter periods (largest harmonic-numbers). The highest power is indicated at harmonic-1, with a level at least three times higher than the level at any other harmonic. This is not a real effect. The Fourier-series is defined for records of infinite-length. In using a finite-segment of a tide-record we have essentially created a signal that is a 'pulse' the length of our tidal-segment, and forced in a

TABLE B-2: Data for the 30 largest harmonically-related sinusoids from a 1024-point (1023-hour) tidal-excursion record. The table-data were generated using the program HARMONICS (Appendix-F). The tidal-data used in producing the table are the corrected values from this thesis. The table-data are in descending power-spectral-estimate (power) order.

k	Period (hours)	α_k	β_k	A_k	ϕ	Power
1	1023.00	12.68	-0.82	12.70	-0.06	80.70
3	341.00	-3.97	5.68	6.93	-0.96	24.01
43	23.79	6.19	3.05	6.90	0.46	23.80
4	255.75	-6.04	-2.91	6.71	0.45	22.52
40	25.56	-5.54	1.72	5.80	-0.30	16.84
82	12.47	-4.14	3.45	5.40	-0.70	14.56
83	12.32	2.88	-2.49	3.81	-0.71	7.24
2	511.50	-3.33	1.24	3.56	-0.36	6.32
81	12.63	-3.04	0.87	3.16	-0.28	5.01
39	26.23	2.04	-1.20	2.37	-0.53	2.80
5	204.60	-2.17	0.14	2.17	-0.06	2.36
42	24.36	-1.98	0.38	2.02	-0.19	2.04
8	127.88	-0.24	-1.90	1.91	1.44	1.83
6	170.50	-1.11	-1.56	1.91	0.95	1.83
7	146.14	-0.27	-1.85	1.87	1.42	1.74
41	24.95	-1.60	-0.46	1.66	0.28	1.38
85	12.04	-1.35	-0.89	1.61	0.58	1.30
38	26.92	1.56	0.29	1.58	0.18	1.25
86	11.90	1.56	-0.23	1.58	-0.15	1.25
11	93.00	0.42	-1.11	1.19	-1.21	0.71
10	102.30	1.05	-0.49	1.16	-0.44	0.68
46	22.24	-0.21	1.02	1.04	-1.37	0.54
80	12.79	-0.84	0.41	0.93	-0.45	0.43
79	12.95	-0.71	0.43	0.83	-0.54	0.34
44	23.25	0.68	0.43	0.80	0.57	0.32
87	11.76	0.71	-0.32	0.78	-0.43	0.30
78	13.11	-0.70	0.30	0.76	-0.40	0.29
88	11.62	0.65	-0.30	0.72	-0.43	0.26
84	12.18	0.44	-0.54	0.70	-0.89	0.24
89	11.49	0.58	-0.32	0.66	-0.50	0.22



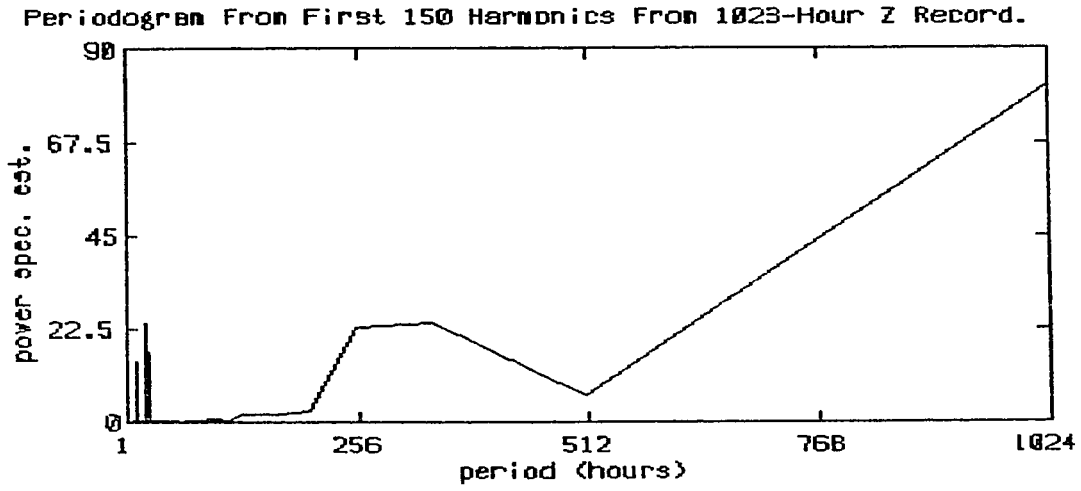
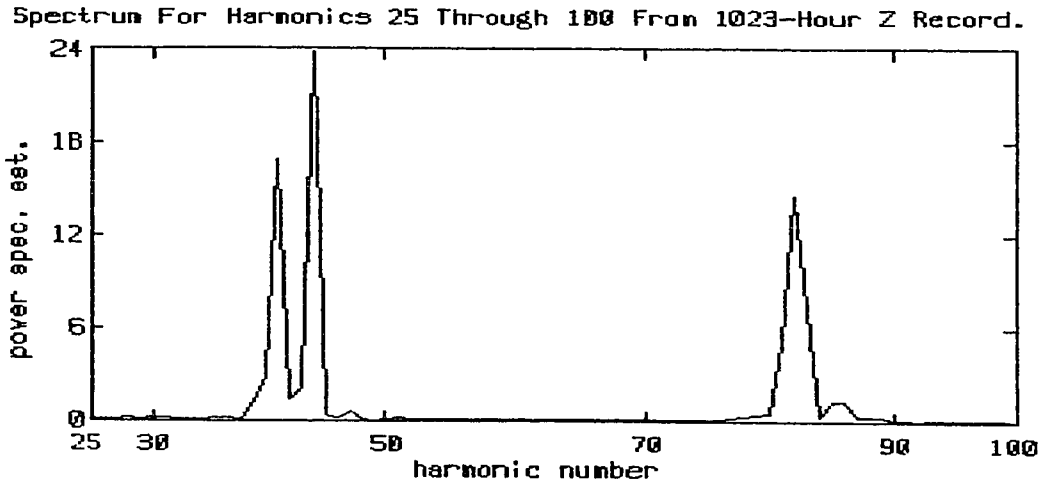


FIGURE B-13: Power-spectrum and periodogram for first 150 harmonics from the 1024-point (1023-hour) tidal-excursion record.

period which is the length of that pulse. This is an 'aliasing' effect (see Appendix D), and although we use the length of the fundamental (harmonic-1) in calculating the period of the other harmonics, we ignore this fundamental in producing harmonic-sums to approximate the original signal. We must use no lower than the second-harmonic, which has a period one-half (frequency twice) the fundamental. Another aliasing-related affect that forces us to ignore long-period harmonics when using this technique is 'quantizing' of harmonic-frequencies. That is, e.g., if we divide the fundamental period 1023-hours by the harmonic-value 3, we get 341 hours, if we divide 1023 hours by 4 we get 255.75 hours. The period of the Lunar-fortnightly astronomical partial-tidal-constituent (Mf) is 321.86 hours. It is not possible with our 1023-hour-span, one-hour-increment data to resolve a period between 341 hours and 255.75 hours, therefore, it is not possible to properly resolve this constituent from our data. As harmonic-number increases, the quantizing affect decreases, and we are better able to resolve short-period components. Looking at the spectrum from harmonic-value 25 to harmonic-value 100 we see peaks near the 12.5 and 25-hour periods, as expected for a semidiurnal-tide (Figure B-14). Comparing our calculated harmonics with accepted values for tidal-harmonics (Cheng and Gartner 1984) we see that our periodogram peaks are close but not quite the same (Table B-3). Due to the errors in period and the large standard-error in spectrum readings, sums formed using harmonics calculated from periodogram data have very large residuals when compared to the original data. The above example is only a sketch of what is implied by a harmonic-analysis of tidal-excursion



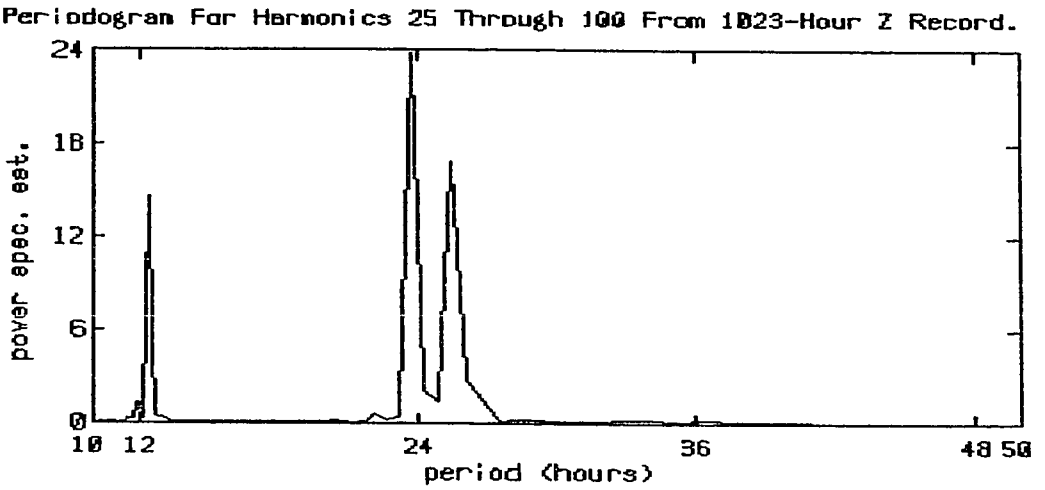


FIGURE B-14: Power-spectrum and periodogram for harmonics 25 to 100 from the 1024-point (1023-hour) tidal-excursion record.

TABLE B-3: Comparison of accepted astronomical partial-tidal-constituents (Cheng and Gartner 1984) with data for harmonics derived using the program HARMONICS (Appendix-F) with the corrected tidal-excursion data of this thesis.

astronomical		closest calculation	
symbol	period (hours)	harmonic number	period (hours)
K1	23.93	43	23.79
O1	25.89	40	25.58
P1	24.07	42	24.36
Q1	26.87	38	26.92
J1	23.10	44	23.25
M1	24.83	41	24.95
M2	12.42	82	12.48
S2	12.00	85	12.04
N2	12.66	81	12.63
K2	11.97	86	11.90
v2	12.63	80	12.79
L2	12.19	84	12.18
T2	12.02	85	12.04
α2	12.87	79	12.95
MK3	8.18	125	8.18
Mf	327.86	3	341.00

data. Calculations for such an analysis are usually made using data with a sample-interval of less than 1 hour and least-squares matrix-solution calculations for the Fourier-coefficients of the major-constituents, minimizing the square-error between field-readings and corresponding estimates (Cheng and Gartner 1984). Minor-constituents are generated through inference using Newton's equilibrium-theory, and additional factors applied to amplitude and phase (Schureman 1940).

Continuous-Fourier-Transform

The Fourier-series requires a periodic signal. Pulses and noise are not periodic. Therefore, harmonic analysis, as outlined above, of a pulse-signal, or a signal with random-noise (virtually any 'real' signal), may give unreliable results. Returning to the integral-forms of the Fourier-coefficients (B-21, B-22, B-23) we can extend these definitions to allow for the analysis of nonperiodic signals. The integrals are expressed evaluated from 0 to the period, T, of the signal, x(t). Since the signal,x(t), is periodic, the integration result is the same over any period.If we alter the limits of integration from 0 to T, to -T/2 to +T/2, and note that $1/T = \theta/2\pi$, where θ is the radian-frequency of the fundamental, the equations for the Fourier-coefficients become:

$$A_0 = \frac{\theta}{2\pi} \int_{-T/2}^{T/2} x(t)dt, \qquad \text{(B-30)}$$

$$\alpha k = \frac{\theta}{\pi} \int_{-T/2}^{T/2} x(t)\cos(k\theta t)dt, \qquad \text{(B-31)}$$

and,

$$\beta k = \frac{\theta}{\pi} \int_{-T/2}^{T/2} x(t)\sin(k\theta t)dt, \qquad \text{(B-32)}$$

If k is allowed to take negative values, then, since $\cos(k\theta t) = \cos(-k\theta t)$, and $\sin(k\theta t) = -\sin(-k\theta t)$,

$$\alpha_{-k} = \alpha_k, \tag{B-33}$$

and,

$$\beta_k = -\beta_{-k}. \tag{B-34}$$

This says that α_k is an even-function, and β_k is an odd-function. The same is true of cos(kθt) and sin(kθt), respectively. So, terms

$$\alpha_k \cos(k\theta t), \tag{B-35}$$

and,

$$\beta_k \sin(k\theta t), \tag{B-36}$$

are the same function of t, regardless of k being positive or negative. (The same as making the algebraic statements: 'positive' times a 'positive' is 'positive,' and 'negative' times a 'negative' is 'positive.') Using negative and positive values of k, the Fourier-coefficients can be reduced by one-half. Thus, the constants in front of the integrals for α_k (B-31) and β_k (B-32) become $\theta/2\pi$, which is the same as the constant for A_0 (B-30). The Fourier-series equationsincluding negative k are:

$$x(t) = A_0 + \sum_{\substack{k=-\infty \\ k\neq 0}}^{\infty} \alpha_k' \cos(k\theta t) + \beta_k' \sin(k\theta t), \tag{B-37}$$

$$A_0 = \frac{\theta}{2\pi} \int_{-T/2}^{T/2} x(t)dt, \tag{B-38}$$

$$\alpha_k' = \frac{\theta}{2\pi} \int_{-T/2}^{T/2} x(t)\cos(k\theta t)dt, \tag{B-39}$$

and,

$$\beta_k' = \frac{\theta}{2\pi} \int_{-T/2}^{T/2} x(t)\sin(k\theta t)dt, \qquad \text{(B-40)}$$

Noting that β_0 is always 0, cos(0) = 1, and sin(0) = 0, the definition of A_0 (B-37) can be included in the definition of α_k, and, with the change of variables:

$$C_k = \frac{2\pi a_k'}{\theta}, \qquad \text{(B-41)}$$

$$S_k = \frac{2\pi \beta_k'}{\theta}, \qquad \text{(B-42)}$$

the Fourier-series and its coefficients become:

$$x(t) = A_0 + \sum_{k=-\infty}^{\infty} C_k\cos(k\theta t) + S_k\sin(k\theta t). \qquad \text{(B-43)}$$

$$C_k = \int_{-T/2}^{T/2} x(t)\cos(k\theta t)dt, \qquad \text{(B-44)}$$

and,

$$S_k = \int_{-T/2}^{T/2} x(t)\cos(k\theta t)dt, \qquad \text{(B-45)}$$

The main difference between our new equations for the Fourier-series (B-43, B-44, B-45) and the previous versions (B-20, B-21, B-22, B-23) is negative-values of k were introduced so the sum goes over each absolute-value of k twice, except for k = 0. And, the size of coefficients was

reduced by half to compensate for the extra terms. The new equations may still be used to calculate the Fourier-series of a periodic-signal.

Consider a nonperiodic-signal, e.g., one everywhere zero except for a region composed of a half-circle. This signal can be approximated by a periodic-signal that is everywhere zero except for two regions composed of half-circles, separated by T, the period of the signal (Figure B-15). As T becomes infinite, the periodic-approximation to the nonperiodic-signal becomes nearly exact. Note that as T becomes infinite the fundamental-frequency, $\theta = 2\pi/T$, approaches 0, and the harmonics, $k\theta$, become infinitesimally close together. For T large and a frequency interval dw that is large compared to θ but small enough so that there is little change in C_k across it, the sum of the C_k (B-44) for the values of k that relate to the interval dw can be approximated by:

$$\Sigma C_k \approx C_k \frac{dw}{\theta}, \qquad \text{(B-46)}$$

since the term dw/θ gives the number of gaps between the C_k values within the interval dw (Figure B-16). Letting $k\theta = w$, $C_k = C(w)$, and making similar statements for the S_k (B-45), then the sum for the Fourier-series (B-43) can be replaced by the integral-equation:

$$x(t) = \frac{\theta}{2\pi} \int_{-\infty}^{\infty} \left[C(w)\cos(wt) \frac{dw}{\theta} + S(w)\sin(wt) \frac{dw}{\theta} \right], \qquad \text{(B-47)}$$

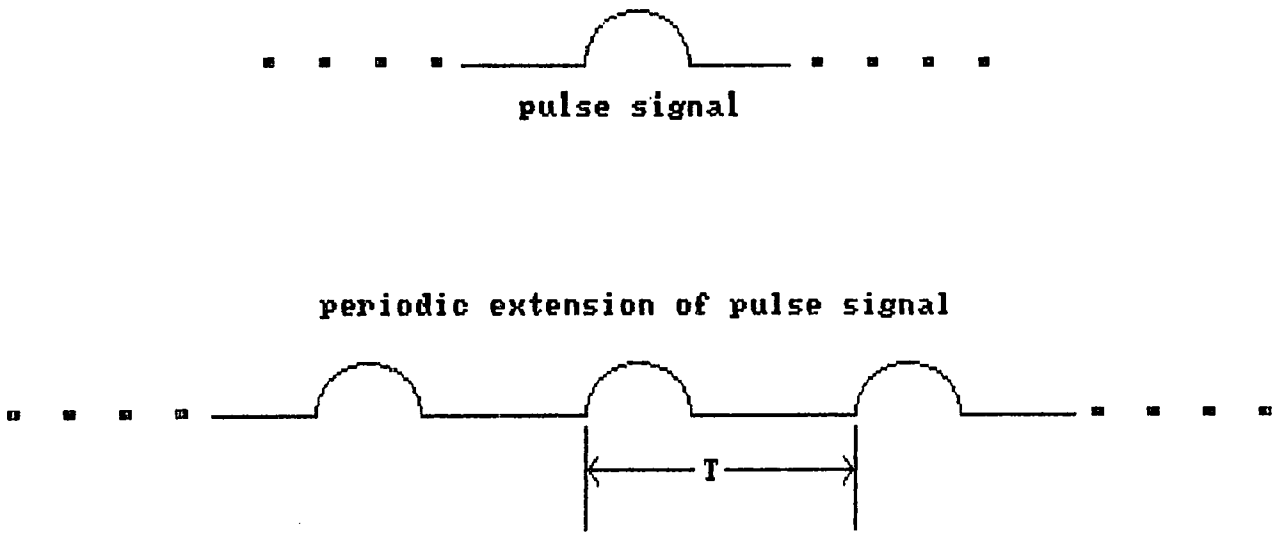
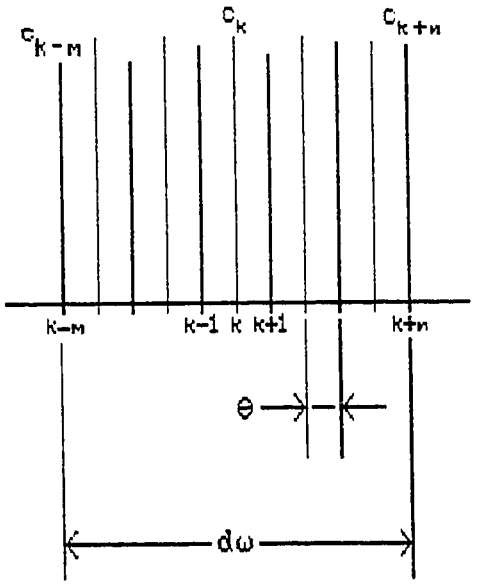


FIGURE B-15: Extension of single-pulse record to an infinite record with period T.



$$d\omega = [\,(k-m) - (k+n)\,] = \Theta(n+m)$$

$$\frac{d\omega}{\Theta} = (n+m)$$

FIGURE B-16: Approximation of the sum of the C_k Fourier-coefficients.

which allows us to find frequency-components for nonperiodic signals. Simplifying the integral-equation (B-47) with respect to θ, we can make the statements:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} [C(w)\cos(wt)dw + S(w)\sin(wt)dw], \qquad (B\text{-}48)$$

$$C(w) = \int_{-T/2}^{T/2} x(t)\cos(w)dt, \qquad (B\text{-}49)$$

and,

$$S(w) = \int_{-T/2}^{T/2} x(t)\sin(w)dt, \qquad (B\text{-}50)$$

These equations (B-48, B-49, B-50) can be used to determine the spectrum for periodic or nonperiodic signals. However, the signal, x(t), must be at least piecewise-continuous, hence integrals are required for a solution. Unless the form of the signal, x(t), is known, a numerical approximation is usually required for the Fourier-coefficients (B-49, B-50) of the Continuous-Fourier-Transform (CFT) (B-48).

Validity of the Periodogram Technique

A periodogram can be developed for periodic and nonperiodic signals using the CFT, just as one develops a spectrum for a periodic signal using the Fourier-series. However, as we saw in our example analyzing a tidal-excursion record using Fourier-series, the results may be subject to large errors. Many authors mention the instability of solutions using

direct Fourier-analysis at some length; some go so far as to say most of the structures one sees in a periodogram are "pure garbage" (e.g., Yuen and Fraser 1979). Besides being unreliable, the periodogram requires about n^2 multiplications and additions to Fourier transform n numbers; a computationally-expensive process for sizeable n.

The problems of instability of periodograms produced by direct Fourier-transform techniques and the large number of calculations required to produce the periodogram were first solved with the discovery that a periodogram can be computed by Fourier-transforming the autocorrelation of the input-data. The autocorrelation was found to be fairly reliable for small values of argument but erratic for large arguments (Yuen and Fraser 1979). So for this technique only the values of autocorrelation for small argument are Fourier-transformed, and the rest are discarded. This not only improves stability, but reduces the number of calculations required to produce the periodogram.

Windowing, Leakage and the Gibbs Phenomena

A spectrum produced using the Fourier-transform of an autocorrelation can be improved by averaging neighboring-values in the spectrum, or by multiplying the autocorrelation-function by a function that reduces the input to the Fourier-transform gradually, rather than abruptly cutting off values for large arguments. The two techniques, known as windowing, are equivalent. Abruptly cutting off values in the

autocorrelation-function introduces discontinuities. Discontinuities in a signal cause ringing (noise) in its Fourier-transform. This form of ringing is known as leakage. For a spectrum derived from Fourier-transforming portions of an autocorrelation-function, improvements through windowing come about, in part, due to a reduction of affects of discontinuities on the Fourier-transform. If we consider the Fourier-transform as a weighted, nonrecursive filter, then leakage comes about from formation of large side-lobes in the system-transfer-function of the filter (see Appendix B). These side-lobes cause large contributions to the transfer-function from frequency ranges distant from the center frequency (Kulhanek 1976).

An effect similar to leakage, only arising in the time-domain-inverse of the Fourier-transform is the Gibbs phenomena. Consider the truncated Fourier-series in exponential-form:

$$\tilde{x}(t) = \sum_{k=-N}^{N} X_k(t) e^{i2\pi kt}, \qquad \text{(B-51)}$$

where $\tilde{x}(t)$ is the 2N+1 term approximation to the infinite Fourier-series-sum for x(t). This series may be considered the result of multiplying the elements of the sequence of Fourier-coefficients, X_k, by corresponding elements of a sequence of zeros and ones, all 0 except for a segment of 2N+1 ones (Hamming 1983), which generates abrupt terminations in the frequency-domain. We noted that abrupt termination of the time-domain input to the Fourier-transform caused ringing in the frequency-domain.

The ringing observed in the time-domain reconstruction of a signal from a truncated Fourier-series, near discontinuities in the original-signal, is known as Gibbs phenomena, (after J. Willard Gibbs). It is a result of abrupt terminations in the frequency-domain.

Consider using the truncated Fourier-series (B-51) to approximate a signal, x(t), which is periodic, piecewise-continuous and contains abrupt discontinuities, the squarewave:

$$x(t) = \begin{cases} 1, & nT \le t < (n+1)T, \quad n=0 \text{ or } n \text{ even} \\ 0, & nT \le t < (n+1)T, \quad n \text{ odd} \end{cases} . \qquad \text{(B-52)}$$

We know that our squarewave (B-52, Figure B-17) can be reproduced exactly by an infinite Fourier-series sum. However, if fewer than the infinite-sum of harmonics is used, there will be some error, and only an approximation can be generated. It can be shown that at a discontinuity, the truncated Fourier-series will converge to the mean-value between the points at each end of the discontinuity. Close to the discontinuity the series does not converge well, so immediately on either side of the discontinuity ringing known as Gibbs phenomena will be observed in a plot of the truncated Fourier-series sum. Away from a discontinuity, the sum for a reasonable number of terms will converge fairly rapidly towards the signal we are approximating. The form of Gibbs-phenomena ringing is:

$$g(t) = \frac{\sin((2N+1)\pi t)}{\sin(\pi t)} . \qquad \text{(B-53)}$$

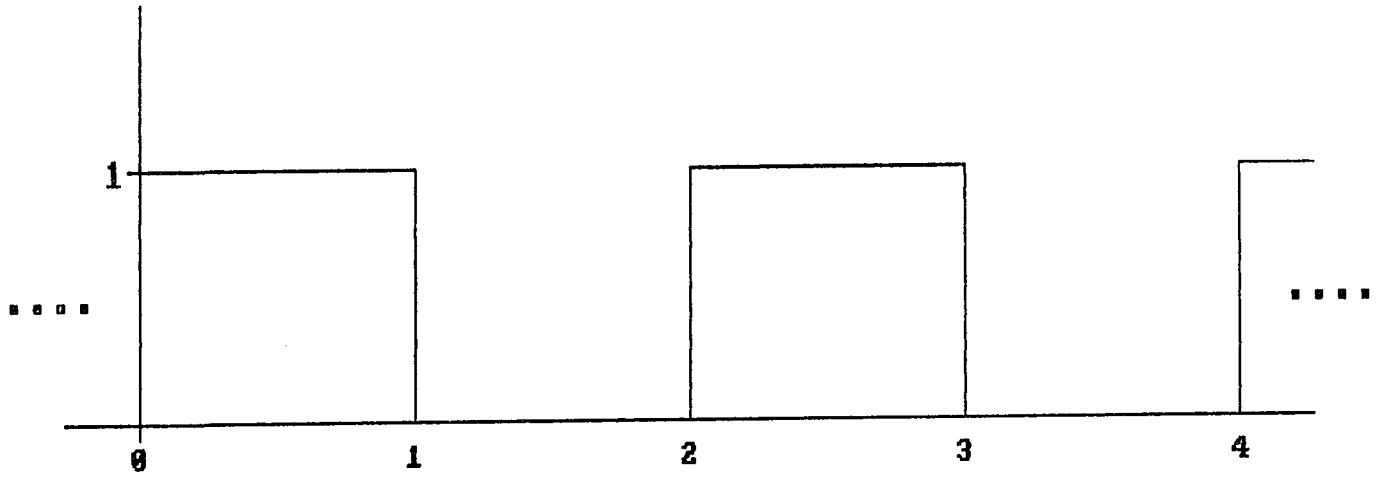


FIGURE B-17: Example periodic waveform, the squarewave.

This is a periodic waveform, (the Dirichlet Kernel), with peaks of width 2/(2N+1) centered at zero, and at each period, flanked by peaks of alternating sign and decreasing amplitude of width 1/(2N+1) (Figure B-18). Regardless of N, where 2N+1 is the number of harmonics used in the truncated Fourier-series (B-51), error remains large near discontinuities (Figure B-19). Increasing N decreases peak widths, but not amplitudes.

We can never form an infinite-series, so Fourier-series results are always subject to the Gibbs phenomena. Also, since we are dealing with sampled data-sets, there will always be at least two abrupt discontinuities in our record: the beginning and the end. Therefore, when filtering data using the Fourier-transform, the filter results for the beginning and end of the record are questionable; these output-data are frequently eliminated. To help reduce the amount of data that must be discarded, raw input records are tapered. That is, a function is applied to data that smoothly forces values towards zero near the ends of the record, but does not alter inner values. This reduces the sharpness of discontinuities at the ends of records, which improves convergence. Many different tapering-filters have been proposed and employed. One that has gained some favor is the split-cosine-bell:

$$u_k = \begin{cases} 0.5(1-\cos(\pi(k-1)/m), & k = 0,1,\ldots,m-1 \\ 1.0, & k = m,m+1,\ldots,n-m-1 \\ 0.5(1-\cos(\pi(n-k+1)/m), & k = n-m,\ldots,n-1, \end{cases} \qquad \text{(B-54)}$$

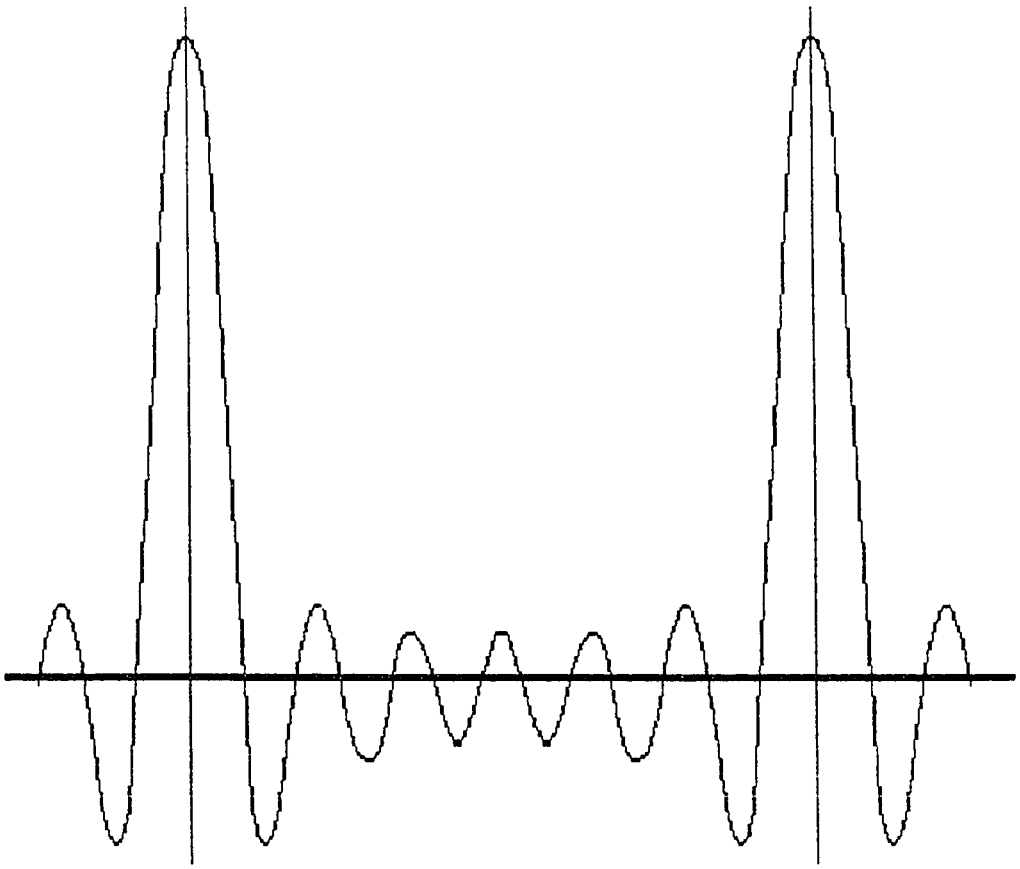


FIGURE B-18: The Dirichlet Kernel.

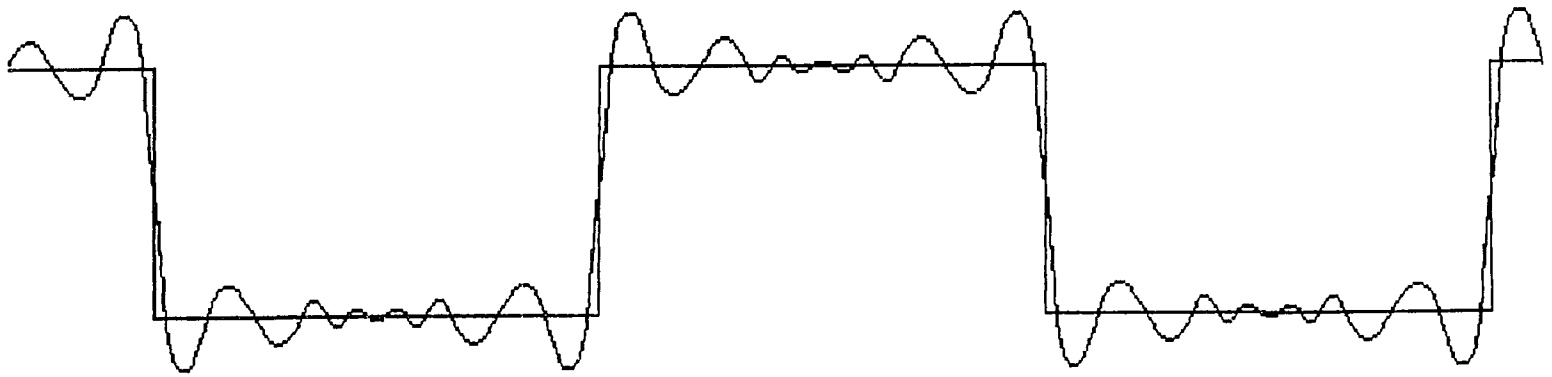


FIGURE B-19: Squarewave exhibiting Gibbs phenomena ringing as the Dirichlet Kernel.

where n is the length of the record to be tapered, m is the number of points tapered on each end of the record (Bloomfield 1976). For the split-cosine-bell (B-54), a rule of thumb is 2m/n should be 10% to 20% of the record length. An optimal tapering value can be found by trial-and-error.

Windowing, as described above while discussing leakage, corresponds to averaging over sets of values in the directly calculated periodogram, which causes erratic fluctuations to cancel out among themselves (Yuen and Fraser 1979). So, one can improve a calculated spectrum by computing a periodogram directly then smoothing it. However, until implementation of the Fast-Fourier-Transform (FFT) on digital-computers (around 1965), the autocorrelation technique for producing a periodogram was the method of choice. Then, with approximately only $n(\log_2(n))$ multiplications and additions required for the FFT versus the near n^2 such operations for a straight Fourier-transform of n values, the direct calculation techniques became practical. Tapering of input-data, calculation of the FFT, windowing in the frequency-domain, and direct smoothing of the periodogram has largely replaced the autocorrelation technique.

Discrete-Fourier-Transform

The Discrete-Fourier-Transform (DFT) may be viewed as an approximation to the CFT based on a finite number of samples (Marple

1987). We can develop the discrete form of the Fourier-transform after converting the continuous form from the integral of a sum of sines and cosines (B-40) to the integral of complex-exponentials. Using Euler's identities:

$$\cos X + i\sin X = e^{iX}, \qquad \text{(B-55)}$$

$$\cos X - i\sin X = e^{-iX}, \qquad \text{(B-56)}$$

where i is the imaginary-value equal to $\sqrt{-1}$, the Fourier equations (B-40, B-41, B-42) become an equivalent transform-pair:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(w)e^{iwt}dw, \qquad \text{(B-57)}$$

$$X(t) = \int_{-\infty}^{\infty} X(w)e^{-iwt}dw, \qquad \text{(B-58)}$$

X(w) (B-58) is the Fourier-transform of x(t), and x(t) (B-57) is the inverse-Fourier-transform of X(w) (Spiegel 1974).

If x_n is a discrete-time signal of length N, $0 \leq n \leq N-1$, then, letting $2\pi/N = \theta$, the discrete-form of the complex-exponential expression of the continuous inverse-Fourier-transform (B-45) is:

$$x_n = \left(\frac{1}{N}\right)\sum_{k=0}^{N-1} X_k e^{i\theta kn}, \qquad \text{(B-59)}$$

and the coefficients, X_k, are given by the Discrete-Fourier-Transform:

$$X_k = (\frac{1}{N})\sum_{k=0}^{N-1} x_n e^{-i\theta kn}, \qquad \text{(B-60)}$$

With the discrete Fourier-transform pair (B-47, B-48), we may construct the spectrum of nonperiodic, discretely-sampled signals.

Given the N-point DFT of a function, we can always construct a Fourier-series that recovers the function exactly, at the sample points. If the function has a finite Fourier-transform confined within the within the indices -N/2 to N/2, then it can be exactly reconstructed from N samples taken at t = k/N, k=0,1,2,...,N. This is known as the sampling theorem (Yuen and Fraser 1979).

The Fast Fourier-Transform

The Fast-Fourier-Transform (FFT) is not another type of Fourier transform. Rather, it is a generic term representing several algorithms for efficient computation of the Discrete-Fourier-Transform (DFT). The basic idea of the FFT is to divide an N-point DFT into two or more smaller DFT expressions. Each smaller DFT can be individually solved, and the results linearly combined to give the DFT of the original N-point series. These smaller problems can be further subdivided (Marple 1987). The technique used (here) to form the separate problems is multidimensional-index-mapping (a change of variables) to convert the

single DFT into a 2 or higher-dimension DFT. The goal is to separate one difficult problem into several simple problems. This is possible because of the structure of the DFT (Burrus and Parks 1985). The DFT is the multiplication of an NxN matrix into an N-element vector (B-60). We take an N-point DFT and converted it into 2 N/2-point DFT's. These N/2-point DFT's can themselves be divided, and so on. N N-point DFT values can be formed by summing the results of N/2 N/2-point DFT computations. When N is a power of 2, an FFT can be reduced to a series of length-2 DFT's, which require no multiplications. This is the Cooley-Tukey FFT algorithm; the algorithm generally being referred to when one simply states 'FFT.' It is a radix-2 FFT. The radix of the FFT is the dimension used for each of the DFT sub calculations. It is not a requirement, but the most efficient FFT algorithms use all dimensions the same. The Cooley-Tukey FFT is called a decimation in frequency (DIF) FFT, as each of the half-length DFT's give samples of the original DFT. One gives the odd-values and the other gives the even-values (Burrus and Parks 1985).

The computations described above appear on first exposure to be somewhat complicated. However, coding (originally in FORTRAN) for the process has been developed that is concise and straight-forward Examples exist throughout the literature (e.g., Press et al. 1987), and the FFT is available precoded in many software-packages.

Digital Filters

At this point we have covered enough ground to give a description of one form of digital-filter: the Fourier-Transform-Filter. The program FOURIER.FILTER (Appendix F) is an implementation of this type of filter intended for removal of tidal-components from an input-record. Rather than reiterate the above more precisely in terms of the Fourier-transform-filter, we will describe the Fourier-transform-filter in terms of operation of the program FOURIER.FILTER. Coded in AmigaBASIC and run under the AC-BASIC compiler, FOURIER.FILTER is a port to the Amiga-2000 computer of a FORTRAN program, FTF.F. FTF.F was implemented under UNIX on a Silicon Graphics IRIS workstation by Dr. Jia Wang during his stay with the USGS Deer Creek office hydrodynamics-group in 1989, under Dr. Ralph Cheng. FOURIER.FILTER is around 1500 lines of code. The length is mainly in the user interface, which generates a number of different 'gadgets' on the Amiga screen so the Amiga mouse may be used for adjusting filter parameters. The actual filtering-code is around 300 lines. It is well identified in the listing. The salient subroutines were derived from those in FFT.F; which came from Press et al. (1987).

The operation of FOURIER.FILTER is basically the procedure as outlined in the section above titled: Windowing, Leakage and the Gibbs Phenomena, for producing a periodogram using the FFT. Prior to performing the FFT, the input-data are (optionally) detrended, have their

mean removed, and split-cosine-bell tapered in order to reduce leakage in the calculated spectrum. As a power-of-two record-length is required for implementation of the FFT, the record is extended by zero-padding if necessary. Some authors swear by zero-padding, others consider it the root-of-all-evil. Properly applied it does have its use. Zero-padding may introduce a discontinuity in data that can cause leakage in a calculated spectrum. Here, since it is assumed each end of the record is tapered to zero, the adverse effects of zero-padding are minimized. Once the data are prepared, Fourier-coefficients are determined using the FFT. This spectrum gives us the frequencies and power of the underlying components of the input-signal. We know the DFT (calculated via the FFT) will give us a Fourier-series that, at least at the sample-points, reproduces the input-data exactly. If we eliminate one of the elements from the spectrum, the inverse-Fourier-transform will, at the sample-points, produce an output that is the input-signal without that component. By eliminating spectral-components, we produce, via the inverse-Fourier-transform, a copy of the input-signal filtered of the time-domain components relating to removed spectral-components. Removing elements causes discontinuities in the frequency-domain, just as zero-padding causes discontinuities in the time-domain. Discontinuities in the frequency-domain cause ringing (Gibbs phenomena) in the time-domain near discontinuities in the original input-signal. The ends of the input-record are discontinuities in the time-domain. Before performing the inverse-Fourier-transform either a linear or cosine-taper is applied to the spectrum at the position of the eliminated spectral-elements to reduce the

Gibbs phenomena, and hence the amount of data lost at each end of the filtered-record (Figure B-20).

FOURIER.FILTER is a tide-removal filter. Being a tide-removal filter means that it is a low-pass filter whose transfer-function has its stop-frequency and pass-frequency set to eliminate components with periods of less than around 30 hours. For a low-pass Fourier-transform-filter, the stop-frequency is the frequency above which all spectral-elements are set to zero, and the pass-frequency is the frequency below which all spectral-elements are unaltered. The band between the stop-frequency and pass-frequency is the region over which tapering is applied to the spectrum. For FOURIER.FILTER the default stop-frequency relates to a period of 30-hours, and the default pass-frequency relates to a period of 40-hours (Figure B-21); the values suggested by Walters and Heston (1982).

The Fourier-transform-filter is not limited to low-pass filtering. If we reverse the stop-frequency and pass-frequency so that only frequencies above the pass-frequency are unaffected, then we have a high-pass filter. Using multiple stop and pass-frequencies, we can generate band-pass and notch filters.

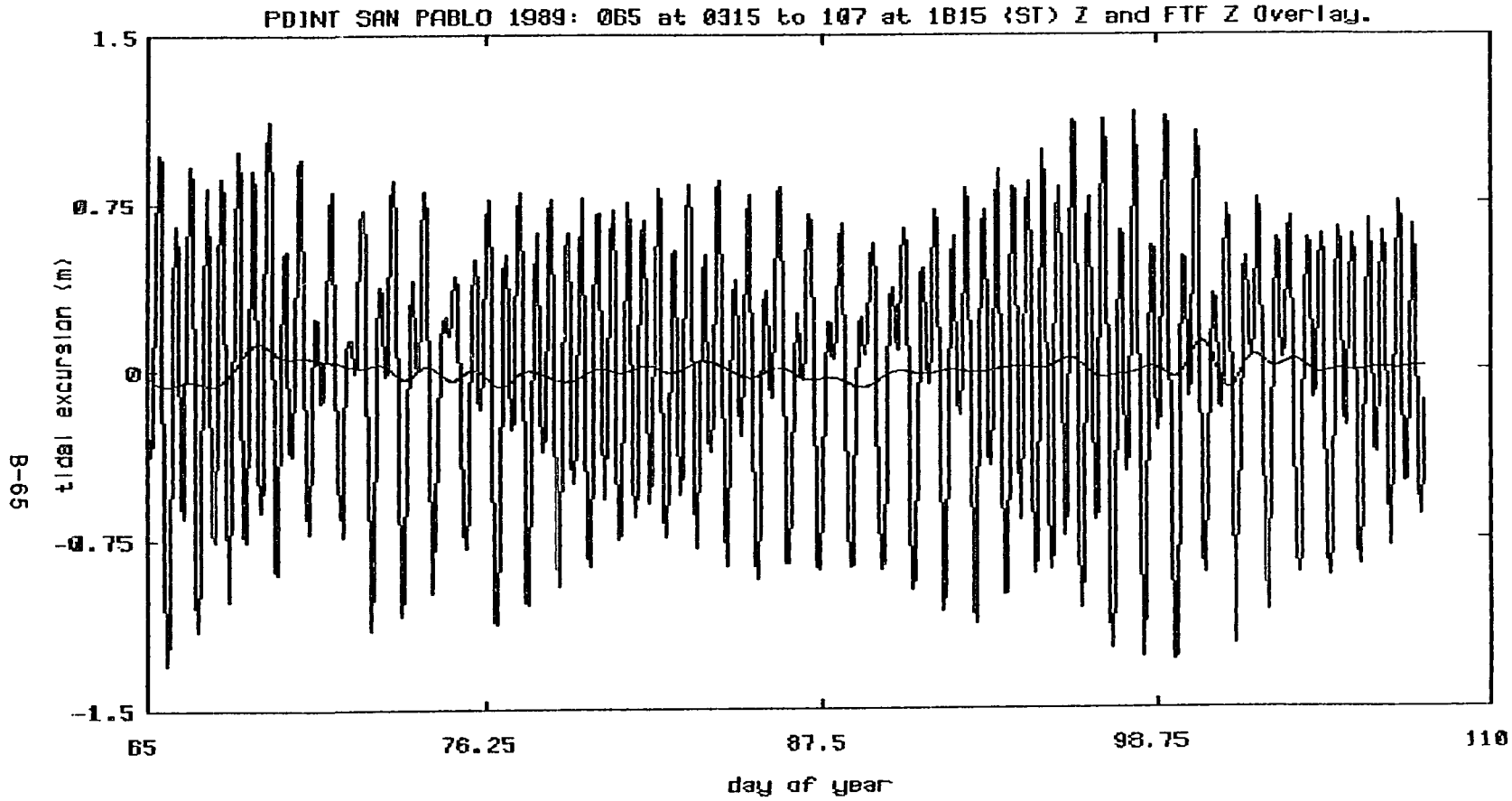


FIGURE B-20: Overlay of unfiltered and Fourier-Transform filtered tidal-excursion record from the Point San Pablo water quality monitoring station, 06 March 1989 at 0315 (ST) to 17 April 1989 at 1815 (ST).

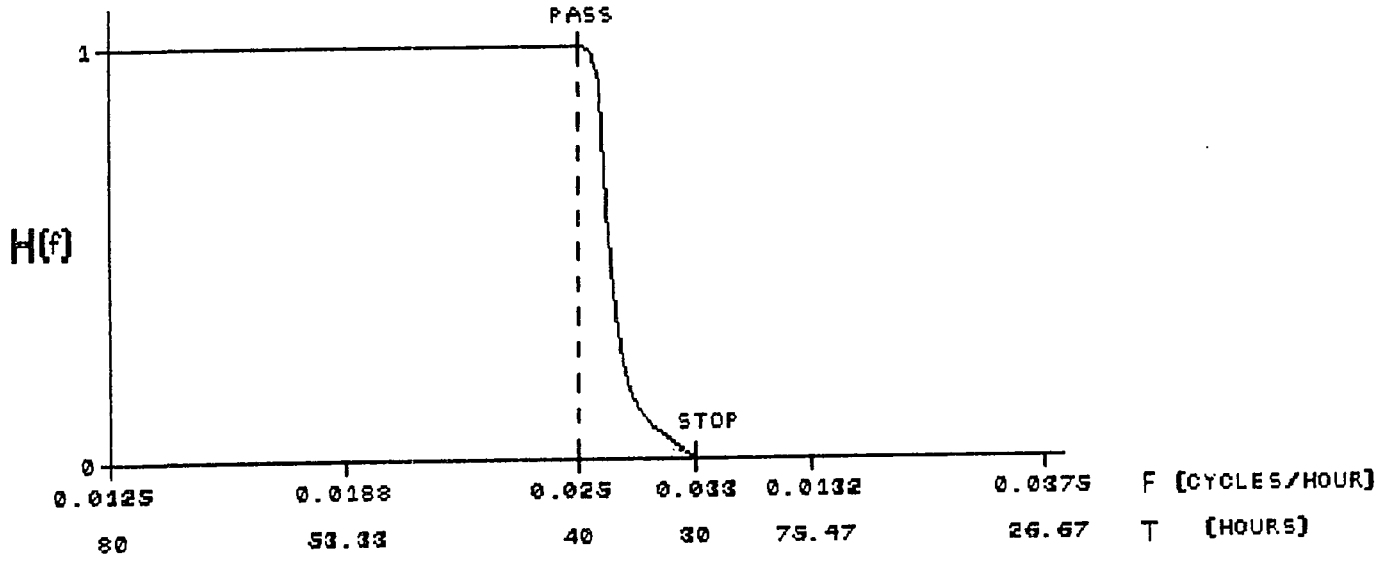


FIGURE B-21: Low-pass transfer function for 30 hour stop-period 40 hour pass-period FFT based tide-removal filter.

APPENDIX C

R-MODE FACTOR ANALYSIS

Introduction

In the most general sense factor analysis is a means of expressing how to get from 'here' to 'there.' Factor analysis, in a way, puts a set of coordinate axes onto your data. You may then, in a manner analogous to looking up a location on the earth based on latitude and longitude, find a variable by looking so far in the direction of one axis, then so far in the direction of another axis, etc., until you come to the desired point.

The 'axes' that come from factor analysis are not necessarily straight lines at right angles to our eyes. However, they do, in the space that can be represented by the variance within a data set, possess the same properties as rectangular coordinates in physical space. They are orthogonal, and contain sufficient information for identifying uniquely each point in space. An 'axis' combined with the 'distance' you must go along that axis on your way to a point, whether in physical or some other 'space,' partially describes that point, i.e., this axis-distance combination is a 'factor' in describing a point. And, a summation of all these factors completely describes the point.

The reason for putting a rectangular coordinate system on physical space is to simplify position descriptions. Which is the same reason for using factor analysis on a data set. Not just to describe the position of a point, but to describe that point in some meaningful way that is more comprehendible and useful than simply looking at the totality of all points in the space. In this sense, oceanographers have been doing a form of factor analysis since the production of the first T-S diagram.

Plotting temperature versus salinity produces a curve that can be used to determine physical properties in the sea that would not readily be seen by placing a column of salinity values next to a column of temperature values and staring at them. The curve plotted on a T-S diagram is a common-factor from two variables. The variables are actually columns of numbers, one set representing temperature and one set representing salinity. This is an important point with regard to factor analysis. We are not considering temperature or salinity to be variables, per se, rather, we define the entire column of temperature values to be one variable, and the entire column of salinity values to be the other variable. By considering our temperature and salinity data as columns rather than on a point-by-point basis we are able to produce a common element from them, our diagram curve. This curve may be viewed as a column of numbers. Thus by our previous definition it is also a variable. However, this new variable by itself contains the information present in our original two variables in a form that makes much of this information available by simple, visual inspection. We, in

making a T-S diagram, have produced a new variable that contains the information of our two original variables in a 'simpler' form. This is the essence of factor analysis. Taking a data set through certain procedures reducing the number of variables that must be considered in interpreting that data set, where the new variables are produced entirely from the original variables, and the new variables describe the original data set in a 'more simple way,' is factor analysis.

The 'simpler form variables' (factors) from factor analysis may or may not have physical meaning. In fact, the curve of a T-S diagram represents density which has definite physical meaning. Density is difficult to observe in the water-column. But, with or without the ability to find the curve directly, it can still be used to analyze the information contained within the original data set variables.

Factor analysis actually refers to a variety of statistical techniques used to produce a representation of a set of variables that is a set of variables smaller than the original set and developed from the original set (Kim and Mueller 1978a). Here the term variable is somewhat broad. A variable in factor analysis is the set of values in one column of a table of data. These variables can represent virtually anything that may be correlated with other similar variables, ranging from political opinions to current meter readings. No distinction is made between variables in terms of being independent or dependent, as is done for regression models. The variables are considered as a single set to be summarized by

a smaller, more basic and meaningful set of 'underlying' variables.

The beginnings of factor analysis date back to at least 1901, when Karl Pearson set forth the method of principle-axes. This was fully developed in the 1930's by Harold Hotelling. The beginning for a limited factor analysis technique, the Two-Factor Theory, was in 1904, when Charles Spearman noted that interrelations among exam scores in subjects such as the classics, French and English fell into a pattern that could be accounted for by a single underlying factor (Harman 1976). Factor analysis as we know it today was introduced first by L.L. Thurstone in 1931 (Guertin and Bailey 1970). Despite its origins, much of the work using factor analysis in meteorology and oceanography tends to imply that it is a more recent development; even giving it a new name: 'Empirical Orthogonal Function (EOF) Analysis.' EOF analysis, as commonly applied in oceanography, is actually R-Mode factor analysis. The R-Mode methods of factor analysis involve looking at the relationships between the columns of a data matrix by examining either the correlation matrix or covariance matrix of the data matrix. The factor variables underlying the data matrix column-variables are constructed from the results of this examination.

There is a debate over whether some procedures commonly referred to as factor analysis actually are 'true' factor analysis, or 'something else.' This arises from within the formal definitions of factor analysis as a multivariate statistical technique. Some methods, such as the one

applied in this thesis, contain a great deal of rote mathematical procedure; leaving statistics beyond the initial calculation of a correlation or covariance matrix hidden in the process. To avoid this issue we simply note that all 'factor analysis' techniques are contained within the realm of the field of latent variable analysis, and hence are related, regardless of any fine points of debate best left to those working at the theorem level.

This discussion will, in general, be limited to the form of factor analysis used for this thesis, i.e., R-Mode principle components analysis with rotation. Here, we generate principle components through calculation of eigenvalues and eigenvectors for the chosen similarity matrix, i.e., either the correlation matrix or covariance matrix of the data matrix. Often one will see this technique referred to simply as eigen-analysis.

Eigen-analysis produces an ordered set of vectors (axes) and associated weights (loadings) so that each vector-weight-combination (component) explains the maximum amount of variance in the data set possible, after the variance explained by any preceding components has been removed. This in some sense says that each component explains the variance seen in the data set when looking only in the direction of that component. It also says that the first component explains the greatest portion of the variance in the data matrix. This first component may be considered the average component for the data matrix.

Factor-analysis is an indeterminant process. That is, as with trying to solve a set of simultaneous-equations with more unknowns than equations, there is an infinite number of possible solutions. We attempt to solve the problem of indeterminancy using a technique known as rotation. Rotation, here, refers to rotating the principle-component axes generated during eigen-analysis to some orientation that produces a 'simplest form' according to criterion to be discussed later.

At this point the reader is assumed to have a background in matrix-algebra sufficient to follow the notation and mathematics presented below. Appendix E provides details of the notation, and on mathematical and statistical operations that will be found in what follows, from the point-of-view of applying them to factor-analysis. There is an emphasis on the geometric interpretation of eigenvalues and eigenvectors and their relationship to linear-systems. Listings and descriptions of the programs developed for the analysis in this thesis are provided in appendix F.

Mathematical Foundations of Factor Analysis

Introduction

An understanding of basic matrix algebra, and the step from simple statistics to multivariate statistics expressed in matrix form is required to follow mathematical discussions of factor analysis. The key to

understanding the mathematical foundation of factor analysis is understanding some of the basic characteristics of linear combinations of variables, in particular, the covariance (or correlation) structure of linear-systems (Kim and Mueller 1978a). Of course, with the advent of computers and prepackaged software, a complete understanding of the mathematical details of factor analysis is not required to apply factor-analysis, any more than such an understanding is required for other analysis techniques. However, as with other techniques, applying factor analysis without some knowledge of basic principles can lead to poor results. This is particularly true of factor analysis, since proper interpretation of factors is an integral part of the analysis process.

The Structure of Linear Systems

The structure of linear systems can be discussed in terms of regression, and multiple and partial correlations. This is the foundation for the basic ideas of factor-analysis. In Appendix E the underlying structure of linear systems is discussed in terms of eigenvalues and eigenvectors. There we develope a set of eigenvectors that fit as closely as possible to a theoretical set of underlying variables which the original data-matrix variables are all (at least partially) correlated with. We examine the variance in common between the original variables and the theoretical variables using the approximating set of eigenvectors.

The theoretical variable set we are trying to correlate with the observed variables is the so-called underlying structure of the linear system represented by a data matrix. Finding the underlying structure of a matrix amounts to finding a basis-vector set for the column-vectors of that matrix. That is, a set of vectors in the minimum number required so that every column-vector in the matrix can be represented as a linear combination of the basis-vector set. Finding a set of eigenvalues and eigenvectors for the covariance matrix of a data matrix produces not just a basis for that matrix, but, in fact, an orthogonal-basis for that matrix. If the number of variables present in a data matrix is greater than the number of basis-vectors required to describe the space of that matrix, then the eigenvectors calculated for the unnecessary dimensions will come back as the null-vector. The number of non-null eigenvectors calculated for a matrix is equal to dimension of the underlying basis-set, which is referred to as the rank of the original matrix.

Eigenvectors are fit to the theoretical underlying variables in a least-squares sense, which makes the process a regression procedure. A regression procedure requires a model. That is, some idea (guess) as to what the underlying structure of a system may be. Linear systems are discussed at length in Appendix E because factor analysis assumes a linear model. Any linear model is a linear system.

In a standard linear regression procedure on multiple variables one assumes not just a linear model for their system, but also which

variables are dependent and which are independent. An independent variable is said to affect the value of a dependent variable, e.g., in the linear equation: $y = mx + b$, y is the dependent variable, x is the independent variable, and we vary x to get new values of y. The number of coefficients for the regression, (i.e., the column-dimension of the assumed basis-space), is generally taken to be the number of independent variables.

If we alter four parameters in an experiment and watch the effects in a single parameter which we do not alter, then, when we perform a linear-regression, we are assuming a model of the form:

$$\overline{v} = a_1\overline{w} + a_2\overline{x} + a_3\overline{y} + a_4\overline{z} \qquad (C\text{-}1)$$

where $\overline{v}$ is a vector containing the readings of the dependent variable, $\overline{w}$, $\overline{x}$, $\overline{y}$, $\overline{z}$ are vectors containing the readings of the independent variables, and the a_i are loadings required to fit the model to the data. The loadings are the unknowns sought by the regression process. The above regression model equation (C-1) can be expressed in matrix form and matrix procedures used to generate a solution. But direct matrix calculations will work only if there is an exact solution to the equation as written.

For real data, measurement errors alone will generally make matrix equation solutions impossible, even if the model is 'perfect.' Some process is required to make a model for the results that produces errors,

(i.e., differences between the output of the experiment and the output of the model), that are by some criterion the smallest errors possible. The least-squares criterion is used in linear regression analysis.

The least-squares criterion for minimum error comes about from the simultaneous solution of multiple linear equations using determinants through a procedure know as Cramer's rule. Cramer's rule is a rote process which uses ratios of the determinant formed from all the columns of the coefficient matrix, save one, (with the right-hand-side vector taking the place of the missing vector), to the determinant formed from all the columns of the coefficient matrix. These ratios give the values of the unknown weights. For a particular ratio, the column position of the right-hand-side vector in the upper determinant gives the index for the calculated weight in the model equation. The calculation of determinants is shown in Appendix E. The important thing to note here is that using a regression model amounts to using the independent variables as basis-vectors, which means using the independent variables as axes. Using a least-squares fit to data in several dimensions results in errors in fitting the model to the data, (taken to be the squares of the difference between model results and corresponding data values), being as small as possible with regard to every axis. When the axes are orthogonal the calculations for a least-squares fit become very simple, (though perhaps tedious), amounting to calculation of vector projections for each dependent-variable point onto all the independent-variable axes, then minimizing these values by an iterative process.

The Correlation Matrix

Before proceeding with the discussion of extracting underlying structure from a matrix by the calculation of eigenvalues and eigenvectors, we must first examine the correlation matrix for a geometric interpretation of what it means. If $\mathbf{Y}$ is a deviate scores matrix, i.e., a data matrix where the individual column means have been removed, then the individual correlation coefficients for its correlation matrix can be determined from:

$$r_{ij} = \overline{y}_i'\overline{y}_j / [(\overline{y}_i\overline{y}_i)^{1/2}(\overline{y}_j\overline{y}_j)^{1/2}]. \qquad \text{(C-2)}$$

That is, the correlation coefficient (C-2) is the minor-product-moment of two vectors, divided by the product of their lengths. This is exactly the formula for the cosine of the angle between two vectors. So, the correlation coefficient uses as its measure of similarity for two vectors, the angle between the vectors. If the angular-distance between two vectors is small, then their correlation coefficient is large, and vice versa. The correlation matrix, $\mathbf{R}$, contains the cosines of all angles between all variables. Intuitively, variables (vectors) that have only a small angle between them must be similar, i.e., correlated.

Unless the variables have been standardized, there is no guarantee on the lengths of vectors being nearly the same even though they may

have only a slight, or even an angle of zero between them. So, the correlation-coefficient can be interpreted, not only as the angle between two vectors, but, as a measure of the tendency for points to group along a line. Any row in a correlation-matrix gives the correlations of all the variables in a data-matrix with the variable in the data-matrix whose index corresponds to the index of that row in the correlation-matrix, e.g., say we have the correlation-matrix:

$$\mathcal{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \qquad \text{(C-3)}$$

for some matrix, $\mathcal{R}$ then row-two of this matrix (C-3), $[\ r_{21}\ \ r_{22}\ r_{23}\ ]$, gives the correlations of variable $\bar{X}_2$ with $\bar{X}_1$, variable $\bar{X}_2$ with $\bar{X}_2$ (the variance of $\bar{X}_2$), and variable $\bar{X}_2$ with $\bar{X}_3$. Speaking geometrically, row-two of the correlations matrix gives the tendencies for all vector-variables in the data matrix to lie along the line of the vector-variable of column-two from the data matrix. If the correlations in row-two are all large, then the points described by the other column vectors will tend to lie along the line of the vector in column-two. If these correlation are not close in value, then the points described by the other column vectors will tend to be <u>scattered</u> away from the line of the column-two vector. The idea of point-scattering based on correlations is helpful in describing the underlying-structure of a matrix as determined through extraction of eigenvalues and eigenvectors of a correlation or covariance-matrix of the original-matrix.

Imagine a data matrix as its component-vectors with their ends all attached to an origin point. Consider some of these vectors may be grouped so the points they describe fall in a relatively straight line, and, in fact, there may be several such groupings within these data (Figure C-1). Say we form a basis-vector set for each one of these groupings, so linear combinations for each of the groupings exist, (i.e., do a least-squares fit for each grouping). Then these basis-sets come close to being the underlying structure of the data matrix. However, these sets may or may not overlap between groupings, (i.e., using the simplistic technique just described, there may be no basis-vector in the set for one grouping that is also a basis-vector in the set for any other grouping). For the true underlying-structure, we need one basis-vector set that fits the whole set of data-matrix vectors.

The groupings as described above are expressed within a correlations matrix generated from a data matrix. They can be determined from a systematic examination of the correlation matrix, and the 'best-fit' lines then determined for these groupings. Formed as just described, (through a regression-process), the best-fit lines for these groupings may not be orthogonal, although they could still be used to describe how the variances in the data are related. This process will determine an approximate basis-set for the entire data matrix, that approximately describes the underlying structure of the matrix. To carry this one step further to factor analysis, we need first to point out explicitly the assumptions of factor-analysis: the underlying structure of a

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \begin{bmatrix} 4.0 & 2.8 & -5.6 & -4.0 & 9.0 & 9.0 & 5.4 & -5.8 \\ -3.8 & 3.0 & 3.8 & 4.4 & -9.0 & -9.0 & 0.0 & 2.0 \\ 2.8 & -2.0 & -6.6 & -5.3 & -5.0 & 0.0 & -6.2 & -5.8 \end{bmatrix}$$

$$i = 1, \quad 2, \quad 3, \quad 4, \quad 5, \quad 6, \quad 7, \quad 8$$

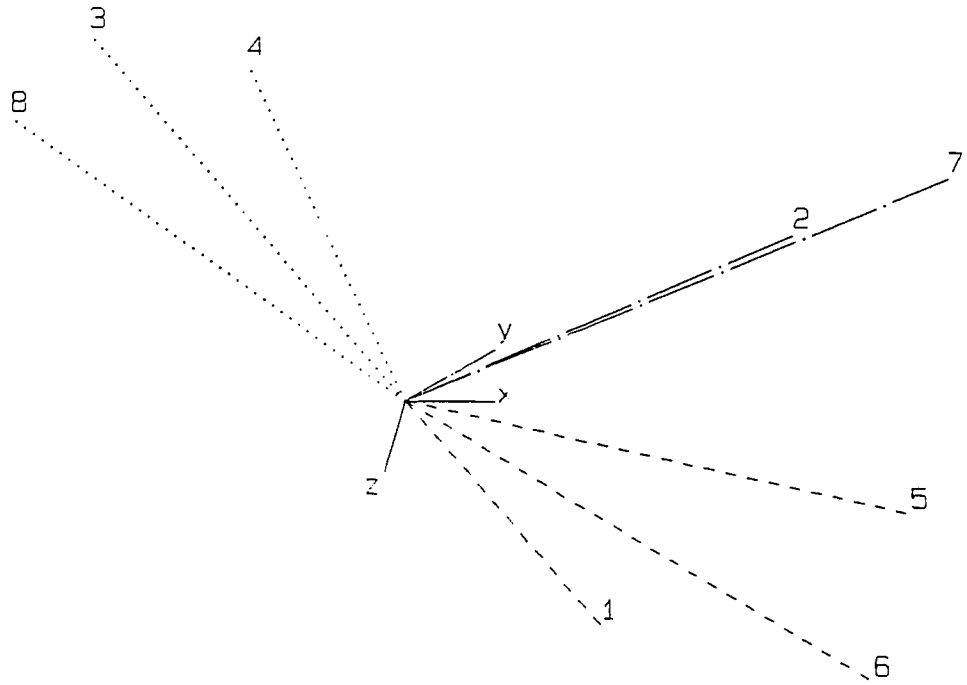


FIGURE C-1: A 3x7 data-matrix and the graphical expression of it's column-vectors in 3-dimensional space. Note that the data were plotted using a left-handed coordinate-system, as is typically done for studies in physical-oceanography.

data matrix is assumed to be linear, and variations in data away from the regression-lines for the underlying structure are caused only by random error. So, factor analysis, like regression analysis, is a modeling process and therefore never exact.

To simplify calculations, (and interpretations), we select lines-of-fit for each grouping that are orthogonal to the lines-of-fit for every other grouping, and force this set of lines to fit the data groupings in a least-squares sense. For factor analysis as applied in this thesis, these lines-of-fit approximate the true underlying structure for the data matrix given one more assumption: that these lines-of-fit (factors) are uncorrelated with each other. This is guaranteed by their orthogonality. The implied inspection of the correlation or covariance matrix for variable-groupings, the determination of an orthogonal-basis that approximates the underlying structure for these groupings in a least-squares sense, and the rank of the data matrix all come out 'for free' in the calculation of eigenvalues and eigenvectors for the correlation matrix or covariance matrix of a data-matrix.

Unlike regression analysis, where there is an assumed set of independent variables and an assumed set of dependent variables, the examination of covariance structure or correlation structure treats all variables equally. Every correlation is compared to every other correlation in attempting to determine underlying structure without considering which variable may be forcing changes in the variance of a

data set, and which variables are reacting to those forcing-functions. In factor analysis the forcing-functions are assumed to be all the variables in the underlying structure, and all observed variables react to them.

Components Analysis

After generating a similarity matrix, (i.e., for us, a correlation-matrix or covariance matrix), for a data matrix, the next step in factor analysis, as presented here, is to find a set of principle components using that similarity matrix. This is components analysis. Components analysis is one of several forms of initial-factoring that may be used in determining a component set such that a maximum amount of the variance in a data matrix is explained by a minimum number of factors (Kim and Mueller 1978b).

In components analysis, factors are produced to account for maximum variance of all the observed variables. Residual terms are assumed to be small in components analysis. Basically this says components analysis accepts that a large part of the total variance of a variable is important and in common with other observed variables. Residuals are assumed to be uncorrelated with the factors. And here we assume that residuals themselves are uncorrelated to simplify calculations and interpretations. However, in components analysis there actually is no assumption about correlations among the residuals.

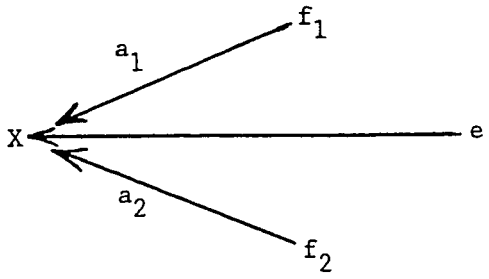
In a single-variable, two-factor model (Figure C-2) factor-1 has no connection to factor-2, and the noise (random error) element is associated only with the single variable. The factors (F_1, F_2) together with the error-element (e) combine to produce the variable (X). The factor-loadings (a_i) and factor-scores (f_i) values combine to form the factors (F_i) , thus,

$$X = F_1 + F_2 + e = a_1 f_1 + a_2 f_2 + e. \qquad \text{(C-4)}$$

This scheme readily expands to multiple variables (Figure C-3).

In factor-analysis a <u>factor-loading represents a correlation of a variable with a factor</u>. The factor-loading of a variable can give what percent of the variance in that variable is predictable by a factor if you simply square the appropriate loading. A squared (orthogonal) factor-loading gives a value analogous to a coefficient-of-determination (Guertin and Bailey 1970). So, there is a correspondence between factor-models and covariance-structures. Using figure C-3 as an example, first note that correlation between factors and random-components is 0, e.g., $COV(a_{11}f_1, d_1e_1) = 0$, and the correlation between random-components is zero, e.g., $COV(d_1e_1, d_2e_2) = 0$. Thus, <u>with the assumptions of linearity and orthogonality</u>, the equations governing the situation are:

$$\begin{aligned} X_1 &= a_{11}f_1 + a_{12}f_2 + d_1e_1 \\ X_2 &= a_{21}f_2 + a_{22}f_2 + d_2e_2 \\ X_3 &= a_{31}f_3 + a_{32}f_3 + d_3e_3. \end{aligned} \qquad \text{(C-5)}$$



$$X = F_1 + F_2 + e = a_1 f_1 + a_2 f_2 + e$$

where: X = observed-variable

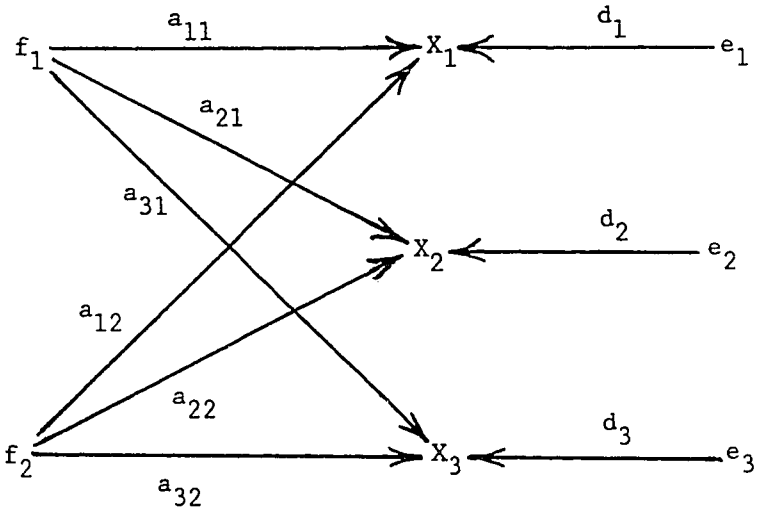
F_i = factor i

e = random-noise term

a_i = factor-loading for F_i

f_i = factor-score for F_i

FIGURE C-2: Representation of a two-factor model.



$$X_1 = a_{11}f_1 + a_{12}f_2 + d_1e_1$$
$$X_2 = a_{21}f_1 + a_{22}f_2 + d_2e_2$$
$$X_3 = a_{31}f_1 + a_{32}f_2 + d_3e_3.$$

where: X_i = observed-variable i

a_{ij} = factor-loading on factor-score j to variable i

f_i = factor-score i

d_ie_i = noise element for X_i

FIGURE C-3: Two-factor model for three variables.

The a_{ij} are factor-loadings, the f_i are factor-scores, (the a_{ij} and f_i combine to form the common-factors, F_i), and the d_ie_i are the unique-factors, or noise-elements for each variable X_i (C-5). Hence, the decomposition of the variance for each variable is:

$$VAR(X_i) = a_{i1}^2 + a_{i2}^2 + d_i^2. \qquad (C-6)$$

The proportion of variance (C-6) for an observed-variable, X_i, explained by the common-factors, i.e., the communality of X_i (denoted h_i^2), is:

$$h_i^2 = a_{i1}^2 + a_{i2}^2. \qquad (C-7)$$

And, the covariance between and two observed-variables, X_i, X_j, is:

$$r_{ij} = a_{i1}a_{j1} + a_{i2}a_{j2}. \qquad (C-8)$$

Thus, given the variances (C-6) and covariances (C-8) of the X_i, some model for the underlying structure of these variables, and a set of components to fit that model, it is possible, essentially through a process of back-calculation, to generate a factor-loading matrix for the model for one possible set of factors. In the case of Figure C-3, this matrix is:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \qquad (C-9)$$

where each of a_{ij} gives the correlation between a common-factor, F_j, and a variable, X_i. The primary factor-loading matrix is identical to the

primary factor-structure (Jennrich and Sampson 1966). With this it is further possible to estimate the factors and model-outputs relative to the used principal-components.

Eigen-analysis is, here, the method used to extract the initial (principal) components necessary to perform the above described calculations. The formation of eigenvectors and eigenvalues involves calculating determinants of the correlation, (or covariance), matrix in a special form which includes introducing an unknown into the matrix. As mentioned previously, determinants come about from the solution of linear equations. They fit a solution to a linear-system model that is 'best' in a least-squares sense. Thus the resultant eigenvectors from principal-components analysis are guaranteed orthogonal, and have a least-squares fit to the model. The ordering of the eigenvectors from most-variance explained to least-variance explained comes about from the calculation of the eigenvalues associated with each eigenvector. The largest-magnitude eigenvalue signifying the eigenvector associated with the underlying-factor explaining the greatest proportion of the total-variance in the data-matrix.

A matrix can be viewed as representing coordinates of points in n-dimensional space, each row representing one point. For 2 dimensions you can think of the 2 points in a 2x2 matrix as lying on the boundary of an ellipse whose origin is the center of the coordinate system. The eigenvalues represent the magnitude or lengths of the major and minor

axes of this ellipse. The corresponding eigenvectors give the directions of these axes away from the origin of the coordinate system. This holds true for any number of dimensions. When eigenvectors are determined for a symmetric matrix, they are guaranteed to be orthogonal. Since in R-mode factor analysis, we are dealing with a covariance or correlation (i.e., symmetric) matrix from the original data set, when we calculate eigenvectors and eigenvalues and perform the operations already described, we end up with mutually-orthogonal vectors (factors) whose lengths are proportional to the variation in the data set one sees 'looking in the direction of that vector.' Stating this all another way, each element of a factor (a factor-score) is weighted proportionally to the square-root of the amount of variance contributed by that variable to the factor (a factor-loading), and that amount of variance is given by the eigenvalue associated with a calculated eigenvector. Thus, in generating eigenvectors and eigenvalues from our similarity matrix we end up with an ordered, preliminary set of orthogonal components that may be used to explain the variance seen in the original data matrix.

Producing factors does not, in and of itself, produce answers. Generation of factors is a rote process. Factors may, or may not, relate to any active processes in an area of study. Factors only give an indication of what part of the total variance found in a data set is common between variables from the data-matrix. This does put a 'coordinate system' on the data, but it is in variance-space only. No physical or biological processes are implied. The important part of the

analysis is developing interpretations for factors. The interpretation process must be considered throughout the analysis, and in a complete report of the results for a factor analysis procedure, the factor interpretation process must be included. The point that can not be over emphasized is: factors must be interpreted. The main tool for interpretation of factors is rotation. It is the rotation and interpretation of initial factors that turns components analysis into what is more reasonably called factor analysis.

Factor Rotation

All that factor analysis does is examine a matrix which expresses the correlations of each variable a data matrix with every other variable in that data matrix. Factor analysis is simply a formal decision making process for selecting subsets of covarying variables, no matter how numerous they may be (Kim and Mueller 1978a).

Factor analysis assumes that the observed (measured) variables are linear combinations of some underlying source variables (factors). That is, it assumes the existence of a system of underlying variables for the system of observed variables. There is a certain correspondence between these two systems and factor analysis 'exploits' this correspondence to arrive at conclusions about the factors (Kim and Mueller 1978b). Factors are constructed in a way that reduces the overall complexity of the data by taking advantage of inherent interdependencies. As a result a small

number of factors will usually account for approximately the same amount of information as a much larger set of original observations. Thus, factor analysis is, in this sense, a multivariate method of data reduction (Joreskog, et al. 1976).

In a physical study, factor analysis solves for the properties of the sources of variation in a data set, and the amount of each of these sources at each site in the study. (Which means if you already know the sources of variation in your data, then you should not be using factor analysis.) The procedure may be used in an exploratory or confirmatory sense; either looking for possible sources of variation, or trying to confirm the presence of assumed sources. (The latter is the application used in this thesis.)

It may be initial factors do not appear to have any relation to believable sources of variation in a data set. For example, in analyzing tide data, the principal components determined may not look like tidal-components. It is not reasonable to assume some new driving force for tides has just been discovered. Rather, the factors determined are just one set from infinitely many possible sets that explain variance in the data, and are not the set we were hoping to find. In this case, the initial factors are rotated to try and produce a 'believable' set of factors.

Here rotation of factors is simply what it sounds like, a rigid-rotation of the produced factor-axes. There are several accepted

procedures for rotation, including straight geometric ones. For this thesis, normalized varimax-rotation (Kaiser 1959), with the modification suggested by Wingersky (Harman 1976) was the method of choice. The procedure is to perform an iterative examination of the factor-loadings matrix, and is well described in the references. Only the general concepts behind this rotation are discussed here. A BASIC-language program (VARIMAX.BAS) for performing normalized varimax-rotation is included in Appendix F.

The purpose of any rotation process is to simplify structure in the factor-loadings. By rotating factor-axes we attempt to force the maximum amount of variance explained for each variable to come as much as possible from only one factor. That is, in pairs, we attempt to reorient the factor-axes so we maximize the variance explained for a variable by one factor, while minimizing the variance explained in that variable by the other factor (figure C-4). This process is continued iteratively until a 'maximum of simplicity' has been obtained for the factor-loadings matrix according to the normalized varimax simplicity-criterion, which involves sums and differences for the factor-loadings raised to the fourth-power. Once this criterion has been achieved, new factors are calculated based on the new factor-loadings resulting from the axes-rotation process. At this point all that is possible through factor-analysis has been done. <u>It remains the job of the investigator</u> to determine if the rotated-factors seem to be a more, or less, reasonable explanation of variance seen in the data-matrix, than the

original principal-components.

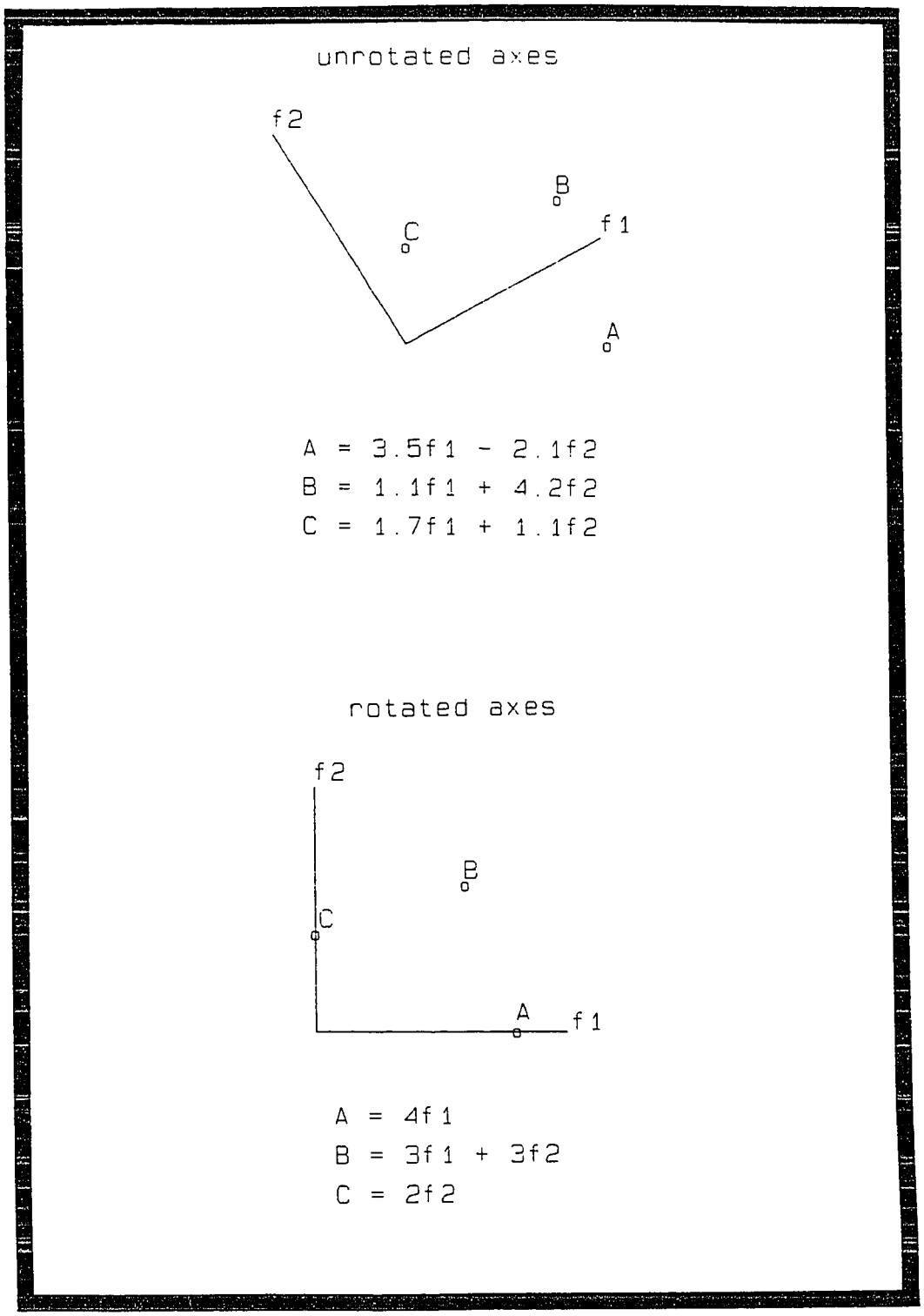


FIGURE C-4: Example of unrotated and rotated factor-axes for the 2 factor, 3 variable case.

R-Mode Factor Analysis Example

Introduction

To construct a data matrix for an example of factor analysis small enough to be 'doable' by hand, three 41-point waveforms (Figure C-5, Table C-1) were generated to have zero mean and an amplitude-range from -1 to 1. The first of these primary-constituent waveforms is four cycles of the sine-function. The second is also a sine-function with period slightly longer than the 41-point record. The truncated-end of this curve was 'flattened' to force a zero-mean for the record. The third constituent is the result of generating 41 values of two-times the output from the pseudorandom-number generator of a scientific-calculator, minus one ($C_{3i} = 2X_{ri} - 1$, $i=1,\ldots,41$). Six of these values were adjusted to force a zero-mean for the record. These three constituents were weighted and summed to form four columns for a data-matrix, $\mathbf{X}$, (Table C-2, Figure C-6) according to the following model:

$$\begin{aligned} \vec{X}_1 &= 85\vec{C}_1 + 10\vec{C}_2 + 5\vec{C}_3 \\ \vec{X}_2 &= 65\vec{C}_1 + 30\vec{C}_2 + 5\vec{C}_3 \\ \vec{X}_3 &= 35\vec{C}_1 + 60\vec{C}_2 + 5\vec{C}_3 \\ \vec{X}_4 &= 25\vec{C}_1 + 70\vec{C}_2 + 5\vec{C}_3 \end{aligned} \qquad \text{(C-10)}$$

where: $\vec{X}i$ = data-matrix column-vector i,

$\vec{C}i$ = primary-constituent vector i,

and vector operations are implied.

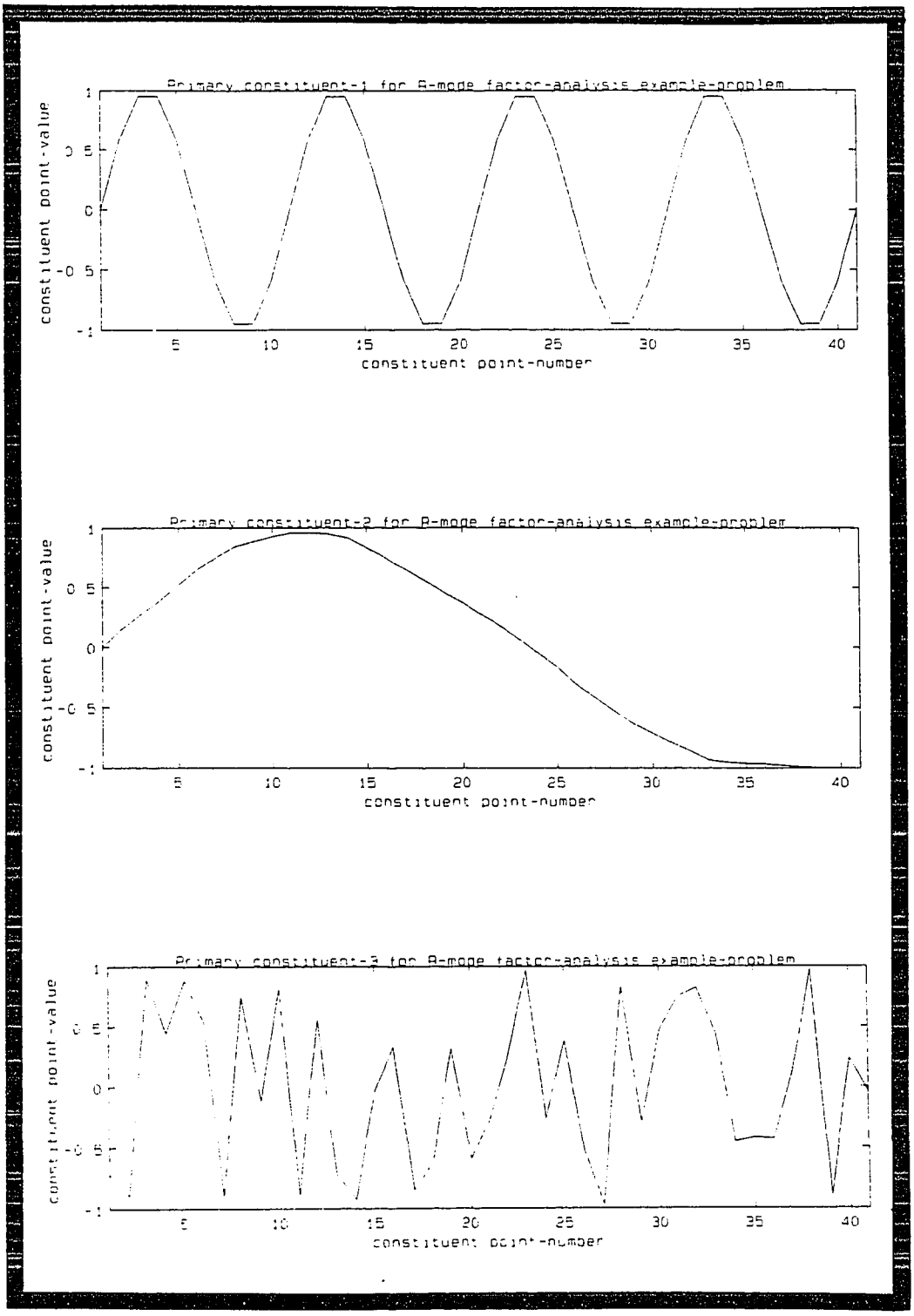


FIGURE C-5: 41-point primary-constituent waveforms constructed for the R-Mode factor-analysis example-problem.

TABLE C-1: Calculated primary-constituents for example problem.

Calculation	C_{1i}	C_{2i}	C_{3i}
1	0.000	0.000	-0.718
2	0.588	0.150	-0.896
3	0.951	0.275	0.892
4	0.951	0.390	0.450
5	0.588	0.520	0.884
6	0.000	0.650	0.546
7	-0.588	0.750	-0.896
8	-0.951	0.840	0.748
9	-0.951	0.885	-0.110
10	-0.588	0.925	0.814
11	0.000	0.960	-0.886
12	0.588	0.960	0.560
13	0.951	0.950	-0.720
14	0.951	0.915	-0.924
15	0.588	0.835	-0.026
16	0.000	0.745	0.334
17	-0.588	0.650	-0.848
18	-0.951	0.560	-0.613
19	-0.951	0.465	0.318
20	-0.588	0.370	-0.588
21	0.000	0.275	-0.282
22	0.588	0.175	0.248
23	0.951	0.065	0.966
24	0.951	-0.050	-0.262
25	0.588	-0.165	0.382
26	0.000	-0.310	-0.530
27	-0.588	-0.420	-0.970
28	-0.951	-0.530	0.831
29	-0.951	-0.635	-0.286
30	-0.588	-0.715	0.470
31	0.000	-0.790	0.750
32	0.588	-0.860	0.820
33	0.951	-0.935	0.438
34	0.951	-0.955	-0.452
35	0.588	-0.965	-0.420
36	0.000	-0.970	-0.431
37	-0.588	-0.985	0.114
38	-0.951	-0.995	0.974
39	-0.951	-1.000	-0.896
40	-0.588	-1.000	0.235
41	0.000	-1.000	-0.038
mean	0.000	0.000	0.000
std. dev.	0.698	0.722	0.642

TABLE C-2: Data-matrix for R-Mode factor analysis example problem.

Observation	X_{1i}	X_{2i}	X_{3i}	X_{4i}
1	-3.59	-3.59	-3.59	-3.59
2	47.00	38.24	25.10	20.72
3	88.04	74.52	54.24	47.48
4	86.98	75.76	58.93	53.32
5	59.60	58.24	56.20	55.52
6	9.23	22.23	41.73	48.23
7	-46.96	-20.20	19.94	33.32
8	-68.70	-32.88	20.86	38.76
9	-72.54	-35.82	19.26	37.62
10	-36.66	-6.40	38.99	54.12
11	5.17	24.37	53.17	62.77
12	62.38	69.82	80.98	84.70
13	86.74	86.72	86.68	86.68
14	85.36	84.64	83.56	83.20
15	58.20	63.14	70.55	73.02
16	9.12	24.02	46.37	53.82
17	-47.72	-22.96	14.18	26.56
18	-78.30	-48.08	-2.75	12.36
19	-74.60	-46.28	-3.80	10.36
20	-49.22	-30.06	-1.32	8.26
21	1.34	6.84	15.09	17.84
22	52.97	44.71	32.32	28.19
23	86.32	68.60	42.02	33.16
24	79.02	59.00	28.98	18.96
25	50.24	35.18	12.59	5.06
26	-5.75	-11.95	-21.25	-24.35
27	-59.03	-55.67	-50.63	-48.95
28	-81.98	-73.56	-60.93	-56.72
29	-88.62	-82.30	-72.82	-69.66
30	-54.78	-57.32	-61.13	-62.40
31	-4.15	-19.95	-43.65	-51.55
32	45.48	16.52	-26.92	-41.40
33	73.68	35.96	-20.63	-39.48
34	69.02	30.90	-26.28	-45.34
35	38.23	7.17	-39.42	-54.95
36	-11.86	-31.26	-60.36	-70.06
37	-59.26	-67.20	-79.11	-83.08
38	-85.92	-86.80	-88.12	-88.56
39	-95.32	-96.30	-97.76	-98.26
40	-58.80	-67.04	-79.40	-83.52
41	-10.19	-30.19	-60.19	-70.19
mean	0.0014	0.0188	0.0410	0.0480
std. dev.	61.41	52.66	52.00	55.34

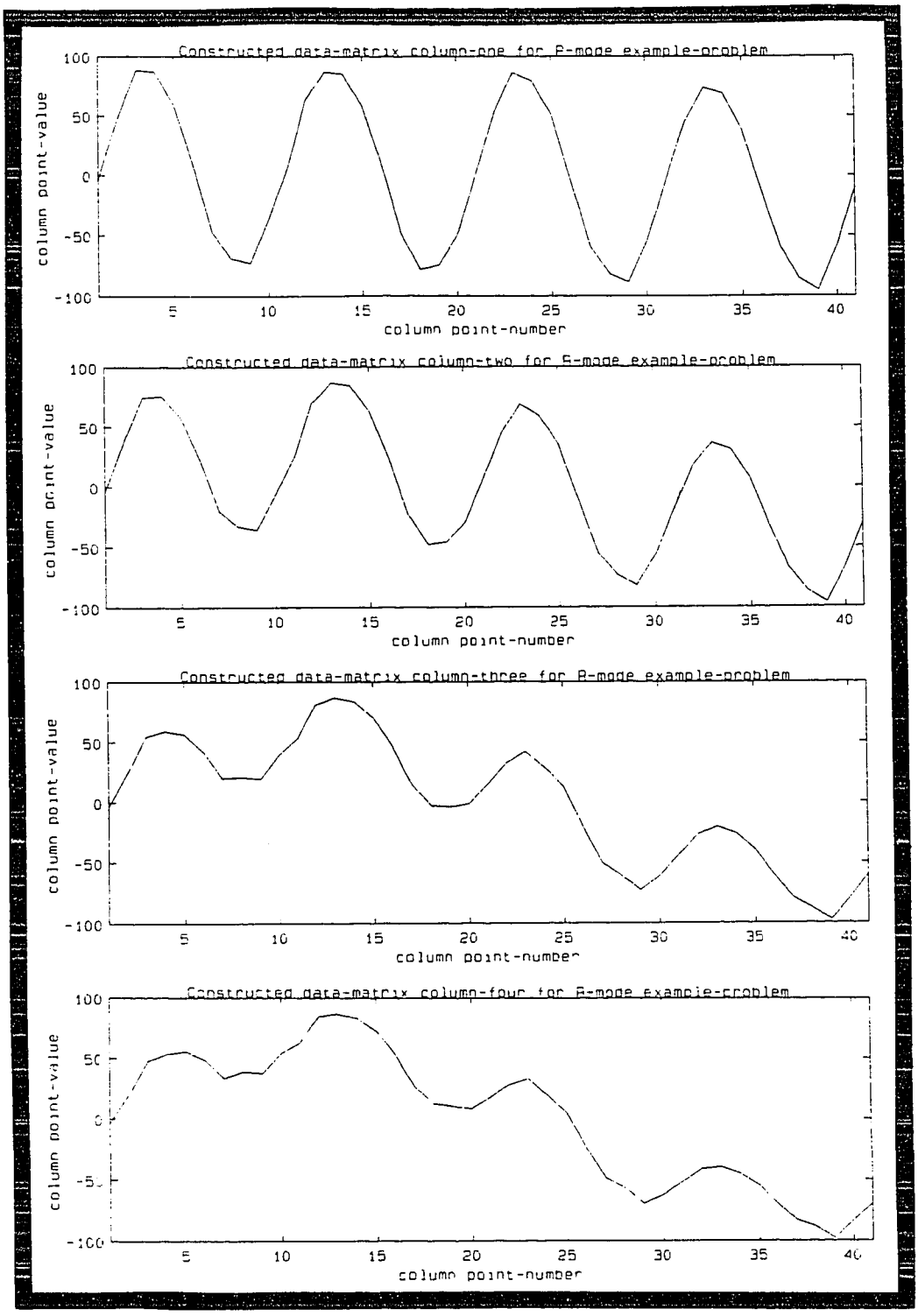


FIGURE C-6: Plotted column-vectors of the data-matrix derived from the 41-point primary-constituent waveforms for the R-Mode factor analysis example problem.

The weights were chosen to sum to 100. And so that the first data-matrix column-vector was most strongly influenced by the first primary-constituent while the last was most strongly influenced by the second primary-constituent, with a gradation between these two for data-matrix column-vectors two and three. Primary-constituent three was given the same small weight in all four data-matrix column-vectors. It represents noise. Note that this construction violates our assumption that the noise-elements in each column-vector are uncorrelated. We, as would be the case for real data, assume noise effects are small and simply don't worry about this.

The example consists of two parts. First is analysis of the constructed data matrix using its covariance matrix. Followed by an analysis of this data matrix using its correlation matrix. The example using the covariance matrix contains the details of the operation. The correlation matrix example is presented only showing the differences between it and the covariance matrix example. The mathematical details are left to Appendix E.

The factor analysis procedure used here follows the presentation in Joreskog, et al. (1976, pp 68-85). It was performed on the constructed data matrix using programs written for this thesis: FACAN.M, PRNCMP.M, VARIMAX.BAS, and ADDTEXT.BAS (Appendix F). FACAN.M and PRNCMP.M are PC-Matlab script-files. VARIMAX.BAS and ADDTEXT.BAS were written in Microsoft GW-BASIC. FACAN.M is the primary program.

It calls PRNCMP.M, VARIMAX.BAS and ADDTEXT.BAS during the analysis.

R-Mode Factor Analysis Using the Covariance Matrix

initial inspection

The covariance-matrix, $\boldsymbol{S}$, from this analysis is:

$$\boldsymbol{S} = \begin{bmatrix} 3679 & 3006 & 1998 & 1662 \\ 3006 & 2706 & 2254 & 2103 \\ 1998 & 2254 & 2638 & 2766 \\ 1662 & 2103 & 2766 & 2987 \end{bmatrix}. \qquad \text{(C-11)}$$

$\boldsymbol{S}$ is symmetric and square. The number of rows and the number of columns in $\boldsymbol{S}$ equals the number of variables in the matrix it was generated from. Ignoring our a-priori knowledge of the model for our data matrix, $\mathbf{X}$ (C-10), (something we normally would not have), we can get a hint of the underlying-structure of $\mathbf{X}$ by examining $\boldsymbol{S}$. Looking at the covariance-values in the rows, $\overline{r}_i'$, for our variables, $\overline{X}_i$, in $\overline{r}_1'$ we see that $\overline{X}_2$ is more closely related to $\overline{X}_1$ than are $\overline{X}_3$ or $\overline{X}_4$. And $\overline{X}_4$ is the variable most poorly related to $\overline{X}_1$. These correspondences are confirmed in the other rows of $\boldsymbol{S}$. If we normalize the rows of $\boldsymbol{S}$, generating $\underline{\boldsymbol{S}}$, by dividing each element of a row by the element with the greatest magnitude in that row, the variable-associations are easier to see.

$$\underline{\boldsymbol{S}} = \begin{bmatrix} 1.00 & 0.83 & 0.54 & 0.45 \\ 1.00 & 0.90 & 0.75 & 0.70 \\ 0.72 & 0.81 & 0.95 & 1.00 \\ 0.56 & 0.70 & 0.93 & 1.00 \end{bmatrix}. \qquad \text{(C-12)}$$

We must be careful to avoid use of the term correlation at this point. $\mathfrak{L}$ is not related to the correlations-matrix, R, in any well-defined way. The variable-associations in $\mathfrak{L}$ are just that, associations, and are valid only on a row-by-row basis. We can see that the associations in the first two rows are weighted most strongly towards $\overline{X}_1$, while the associations in the second two rows are weighted most strongly towards $\overline{X}_4$. The sharp break in direction of weighting tends to imply there are two major-variables in our underlying-structure. One that looks something like $\overline{X}_1$ in our original data-matrix, and one that looks something like $\overline{X}_4$ in our original data-matrix. The two middle-rows of $\mathfrak{L}$ show closer associations between the variables than do the upper and lower rows. This coupled with the direction of the weightings tends to imply that there may be a gradation from $\overline{X}_1$ through $\overline{X}_4$ in application of the underlying-structure variables to produce the observedvariables. This is the limit to any unbiased estimate we can make for the underlying structure of the data matrix based on our simplistic examination of the covariance matrix. However, it is important to at least attempt such an estimate so that some reference will be available when later interpreting factors.

principle components analysis

The process of finding factors is described below following the steps as presented in the program FACAN.M, including those steps, in order, from PRNCMP.M and VARIMAX.BAS. We first produce the

deviate-scores matrix. This is simply the original data matrix with the individual column-means removed. This does not affect column-variances, and eliminates potential offsets between the variables that may exist even when column-variances are similar. From the deviate-scores matrix, $\mathbf{Y}$, of our data-matrix, $\mathbf{X}$, we produce the covariance-matrix, $\mathbf{S}$.

Having $\mathbf{S}$ we perform an initial factorization via the method of principle components. We begin by generating the eigenvectors and eigenvalues of the similarity matrix, $\mathbf{S}$. The eigenvectors are produced as the columns of a matrix, $\mathbf{\Lambda}$.

$$\mathbf{\Lambda} = \begin{bmatrix} 0.528 & -0.660 & 0.000 & 0.534 \\ 0.511 & -0.241 & -0.196 & -0.802 \\ 0.484 & 0.388 & 0.784 & -0.774 \\ 0.475 & 0.597 & -0.588 & 0.267 \end{bmatrix}. \qquad \text{(C-13)}$$

$\mathbf{\Lambda}$ (C-13) has a corresponding ordered set of eigenvalues, λ_i = 9911, 2099, 6.86×10^{-6}, -6.69×10^{-14}. The proportion of the variation in the data matrix explained by a principle component is given by the eigenvalue related to that component divided by the sum of all the eigenvalues. The programs used in the analysis are written to accept the input of a minimum-level for explanation of variance. The proportions of variance explained are then summed until this level is met or exceeded. The index of the eigenvalue in the sum that meets or exceeds this level is the index for the last principle-component to be retained in the analysis. Here we arbitrarily set a minimum explanation-level of 97.5 percent. This results

in the first two principle-components being retained, explaining 100 percent of the variance at levels of 82.5 percent and 17.5 percent, respectively. This gives us the set of eigenvectors, $\vec{\Lambda}_i$, and eigenvalues, λ_i, used to generate the principle-component scores and loadings.

$$\vec{\Lambda}_1 = \begin{bmatrix} 0.528 \\ 0.511 \\ 0.484 \\ 0.475 \end{bmatrix}, \quad \lambda_1 = 9911, \tag{C-14}$$

$$\vec{\Lambda}_2 = \begin{bmatrix} -0.660 \\ -0.241 \\ 0.388 \\ 0.597 \end{bmatrix}, \quad \lambda_2 = 2099, \tag{C-15}$$

If $\boldsymbol{U}$ is a two-column matrix formed from $\vec{\Lambda}_1$ (C-14) and $\vec{\Lambda}_2$ (C-15),

$$\boldsymbol{U} = \begin{bmatrix} 0.528 & -0.660 \\ 0.511 & -0.241 \\ 0.484 & 0.388 \\ 0.475 & 0.597 \end{bmatrix}, \tag{C-16}$$

and Γ is a diagonal-matrix whose diagonal-elements are the square-roots of the eigenvalues, λ_1 , λ_2,

$$\Gamma = \begin{bmatrix} 99.554 & 0.000 \\ 0.000 & 45.185 \end{bmatrix}, \tag{C-17}$$

then the loading-matrix for the principle-components is $\boldsymbol{A} = \boldsymbol{U}\Gamma$.

$$\mathbf{A} = \begin{bmatrix} 5.26 & -3.02 \\ 5.08 & -1.10 \\ 4.82 & 1.78 \\ 4.73 & 2.74 \end{bmatrix}. \qquad \text{(C-18)}$$

The factor-scores are obtained, in matrix-form, as $\boldsymbol{F} = \boldsymbol{Y}\boldsymbol{\Lambda}^{-1}$.

$$\boldsymbol{F} = \begin{bmatrix}
-0.0726 & -0.0074 \\
0.6659 & -0.3962 \\
1.3392 & -0.5826 \\
1.3906 & -0.4581 \\
1.1527 & 0.0339 \\
0.5956 & 0.7310 \\
-0.0973 & 1.3845 \\
-0.2472 & 1.8429 \\
-0.2959 & 1.8853 \\
0.2201 & 1.5959 \\
0.7101 & 1.0645 \\
1.4867 & 0.5229 \\
1.7398 & 0.1573 \\
1.6900 & 0.1164 \\
1.3238 & 0.3777 \\
0.6535 & 0.8353 \\
-0.1757 & 1.2731 \\
-0.6169 & 1.5172 \\
-0.6027 & 1.4196 \\
-0.3829 & 0.9624 \\
0.2002 & 0.3041 \\
0.8016 & -0.3578 \\
1.1719 & -0.8167 \\
0.9528 & -0.9565 \\
0.5318 & -0.7367 \\
-0.3119 & -0.3523 \\
-1.0792 & 0.0755 \\
-1.3799 & 0.3117 \\
-1.5795 & 0.1840 \\
-1.1804 & -0.2411 \\
-0.5833 & -0.8773 \\
-0.0031 & -1.5099 \\
0.2860 & -1.9399 \\
0.1799 & -1.9704 \\
-0.2150 & -1.6387 \\
-0.8518 & -1.0894 \\
-1.4410 & -0.5463 \\
-1.7529 & -0.2070 \\
-1.9447 & -0.2297 \\
-1.4412 & -0.5619 \\
-0.8372 & -1.1194
\end{bmatrix}. \qquad \text{(C-19)}$$

If $\mathcal{D}$ is a diagonal matrix who's diagonal-elements are the standard deviations of the corresponding column-vector variables of the deviate scores matrix, $\mathcal{Y}$ then the matrix of correlations between the variables and the factors is obtained as $\mathcal{C} = \mathcal{D}^{-1}\mathcal{A}$

$$\mathcal{D} = \begin{bmatrix} 61.41 & 0.00 & 0.00 & 0.00 \\ 0.00 & 52.66 & 0.00 & 0.00 \\ 0.00 & 0.00 & 52.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 55.34 \end{bmatrix}. \qquad \text{(C-20)}$$

$$\mathcal{C} = \begin{bmatrix} 0.86 & -0.49 \\ 0.96 & -0.21 \\ 0.93 & 0.34 \\ 0.86 & 0.49 \end{bmatrix}, \qquad \text{(C-21)}$$

where each C_{ij} gives the correlation of variable-i with factor-j.

We previously stated that the communality of the factors for each variable is given by $h_i^2 = a_{i1}^2 + a_{i2}^2$, where the a_{ij} are elements of the factor-loadings-matrix. This is strictly true only when using correlation as the similarity-index. Here we are using covariance, and must normalize the squared-loadings before calculating the communalities. Normalization of a_{ij}^2 is accomplished through dividing a_{ij}^2 by the covariance of $\overline{X}_i$ For this example all the $h_i^2 = 1.00$, which says that the two factors explain all the variation in all the observed-variables.

We will use the subscript $_{\sim}$ to signify the rotated-form of our parameters. Rotation of the factor-loadings matrix, $\mathcal{A}$ (C-18), is (here) accomplished using normalized varimax-rotation, producing $\mathcal{A}_{\sim}$ The eigenvectors are rotated along with the factor-loadings matrix. So, the rotation-process does not require the transform matrix, $\mathcal{T}$, that would convert the unrotated eigenvector-matrix, $\mathcal{U}$, into its rotated form, $\mathcal{U}_{\sim}$ by direct multiplication. However, the conversion of the unrotated eigenvalues and the unrotated factor-scores to their rotated forms is not performed by the varimax-algorithm as presented in Kaiser (1959). The transformation of the factor-scores may be accomplished using the transform matrix. So, in addition to performing rotation of the factor-loadings matrix and the retained eigenvectors, the program VARIMAX.BAS generates the transform matrix, $\mathcal{T}$, via an augmented-matrix technique.

$$\mathcal{T} = \begin{bmatrix} 0.7151 & 0.6991 \\ -0.6991 & 0.7151 \end{bmatrix}. \qquad \text{(C-22)}$$

$$\mathcal{A}_{\sim} = \begin{bmatrix} 5.26 & -3.02 \\ 5.08 & -1.10 \\ 4.82 & 1.78 \\ 4.73 & 2.74 \end{bmatrix}. \qquad \text{(C-23)}$$

$$\mathcal{U} = \begin{bmatrix} 0.839 & -0.102 \\ 0.533 & 0.185 \\ 0.075 & 0.616 \\ -0.077 & 0.759 \end{bmatrix}. \qquad \text{(C-24)}$$

The communalities for the rotated factors are generated from the rotated factor-loadings in the same way they are generated from the unrotated factor-loadings. Here the $h_i^2 = 1.00$ for all the variables. The rotated factor-scores-matrix, $\mathcal{Z}$, is obtained by multiplying the inverse-of-the-transpose-of the transform-matrix, T'^{-1}, by the original factor-loading matrix, , i.e., $\mathcal{Z} = AT'^{-1}$. The matrix of correlations of the variables with the rotated-factors, $\mathcal{C}$, is generated in the same way as it's counterpart from the unrotated case. $\mathcal{C} = T^{-1}A$

A factor-loading matrix is obtained by post-multiplying a matrix of component-eigenvectors by a row-vector of the square-roots of their corresponding eigenvalues. This is true whether or not we are dealing with rotated-forms. So, we generate the corresponding eigenvalues for the rotated-eigenvectors by the inverse operation, a matrix-division of the rotated-eigenvector matrix, $\mathcal{U}$, by the rotated factor-loading matrix, A. This gives a square matrix that has the square-root of the eigenvalues on its main-diagonal. Squaring these diagonal-values gives the eigenvalues, λ_1, λ_2, that correspond to the rotated-eigenvectors, $\bar{A}_1$, $\bar{A}_2$. λ_1 = 5732 and explains 50.84 percent of the variance in the observed-variables. λ_2 = 5195 and explains 49.16 percent of the variance in the observed-variables.

$$\mathfrak{L} = \begin{bmatrix} 0.86 & -0.49 \\ 0.96 & -0.21 \\ 0.93 & 0.34 \\ 0.86 & 0.49 \end{bmatrix} \qquad \text{(C-25)}$$

$$\mathcal{E} = \begin{bmatrix} -0.0467 & -0.0561 \\ 0.7532 & 0.1822 \\ 1.3649 & 0.5196 \\ 1.3146 & 0.6446 \\ 0.8006 & 0.8300 \\ -0.0851 & 0.9391 \\ -1.0374 & 0.9220 \\ -1.4651 & 1.1450 \\ -1.5295 & 1.1413 \\ -0.9582 & 1.2951 \\ -0.2364 & 1.2576 \\ 0.6975 & 1.4132 \\ 1.1341 & 1.3287 \\ 1.1271 & 1.2646 \\ 0.6825 & 1.1955 \\ -0.1167 & 1.0541 \\ -1.0157 & 0.7875 \\ -1.5018 & 0.6537 \\ -1.4233 & 0.5937 \\ -0.9466 & 0.4206 \\ -0.0694 & 0.3574 \\ 0.8233 & 0.3045 \\ 1.4090 & 0.2352 \\ 1.3500 & -0.0180 \\ 0.8953 & -0.1550 \\ 0.0232 & -0.4700 \\ -0.8245 & -0.7004 \\ -1.2046 & -0.7418 \\ -1.2581 & -0.9726 \\ -0.6755 & -0.9975 \\ 0.1962 & -1.0351 \\ 1.0533 & -1.0818 \\ 1.5606 & -1.1872 \\ 1.5060 & -1.2832 \\ 0.9918 & -1.3220 \\ 1.5249 & -1.3745 \\ -0.6485 & -1.3979 \\ -1.1087 & -1.3734 \\ -1.2300 & -1.5237 \\ -0.6377 & -1.4093 \\ 0.1838 & -1.3857 \end{bmatrix}. \qquad \text{(C-26)}$$

factor interpretation

We know that both sets of factors generated above explain 100 percent of the variance in the observed variables. A set of factors that explains 100 percent of the observed variance does nothing more than that. We must make interpretations of the factors to see if they might be reasonable models for the actual processes that are producing variation in our data. Factor interpretation begins with plotting the values from each column of the factor-scores matrix. We use the factor-score values as the dependent-variable. If our variables are readings from sample sites separated by known distances, then we would use distance as the independent-variable. In the case of time-series data, the data time-step is used incrementally along the independent-axis. More esoteric variables, like weighted political-opinion values, would be plotted against an arbitrary fixed-step increment on the independent-axis. We plot the factor-scores this way to see their shape, or form. The reasoning is analogous to that for looking at the form of the curve on a T-S diagram. The single-curve on a T-S diagram gives a reduction of the two-dimensional T-S-space to a one-dimensional space; the 'space' of density. Factor-scores represent the reduction of the multi-dimensional variance-space of our data-matrix much like the curve on a T-S diagram represents the reduction of two-dimensional T-S-space.

Factor-scores are not the underlying-structure variables, and they are not the factor-axes. The eigenvectors, (rotated or unrotated),

represent the factor-axes. Each one analogous to a set of values in the input-parameters for a regression-model. The factor-loadings are weights we apply to the factor-axes to generate our model of the input data-matrix. These being analogous to the unknowns determined in a regression-analysis. Each factor-score represents something like the product of a calculated-unknown times the vector of its corresponding input-parameter values in a regression-analysis. Each value in a factor-scores variable is a normalized estimate of the total variance in a data matrix explained by the underlying-factor corresponding to the factor-score. The placement of that value along the independent-variable plotting-axis tells us 'when' that estimate was generated, ('when' may be 'where' or 'something-else-entirely'). That is, as we look along the plot of a factor-score we see a curve representing a sequence, (in our case a time-sequence), of estimates to the variance contributed to our observed-variables by the corresponding underlying-structure variables. Since our underlying-structure axes are linearly independent and orthogonal we see only information due to a single underlying-factor variable in the plot of a factor-score. Thus, we may INFER the FORM of a FORCING-FUNCTION that represents the true underlying-structure variable of our data-matrix by looking at the affects of this variable on the FORM of the VARIANCE in our observed-variables. Those affects are seen in the form of a plot of a factor-scores matrix variable. Our factor-interpretation is based on examining the form of the underlying-structure variables which we infer from the changes in total-variance-explained looking along the plot of a factor-scores variable.

The plots of factor-scores from the initial-factorization (Figure C-7) and the plots of the corresponding rotated factor-scores (Figure C-8) are somewhat different. The unrotated-scores appear as two modulated sinusiods approximately 180 degrees out-of-phase. The first rotated-score appears as a reasonably-unmodulated sinusoid of the same period as the first unrotated-score. The second rotated-score appears as a noisy sinusoid with period slightly longer than the record length. We now must decide which set would be a better model for the processes we might expect to be generating the variance in our data matrix. This decision must largely be based on the type of data that makes up our observed variables, and on our understanding of processes related to those types of data. Our example data is of no specific type. So, without prior knowledge of the model we have only our initial interpretations of $\underline{\mathbf{L}}$ to guide us. We stated that $\underline{\mathbf{L}}$ might indicate two major factors, one looking something like $\overline{X}_1$ and the other looking something like $\overline{X}_4$, and that a graded-application of these two factors might account for the variance in our four observed variables. This would indicate that the rotated-scores represent 'better' models for the factors than do the unrotated-scores. We also could invoke Occam's Razor and say that simpler explanations are best. Our variables, as we look from $\overline{X}_1$ to $\overline{X}_4$, actually have forms something like the shorter-period and longer-period sinusoids from the rotated factor-scores. Two equal-period, modulated sinusoids 180 degrees out-of-phase producing 82.5 and 17.5 percent, respectively, of the observed variation in the data matrix is not as 'simple' an explanation for the forces giving form to our

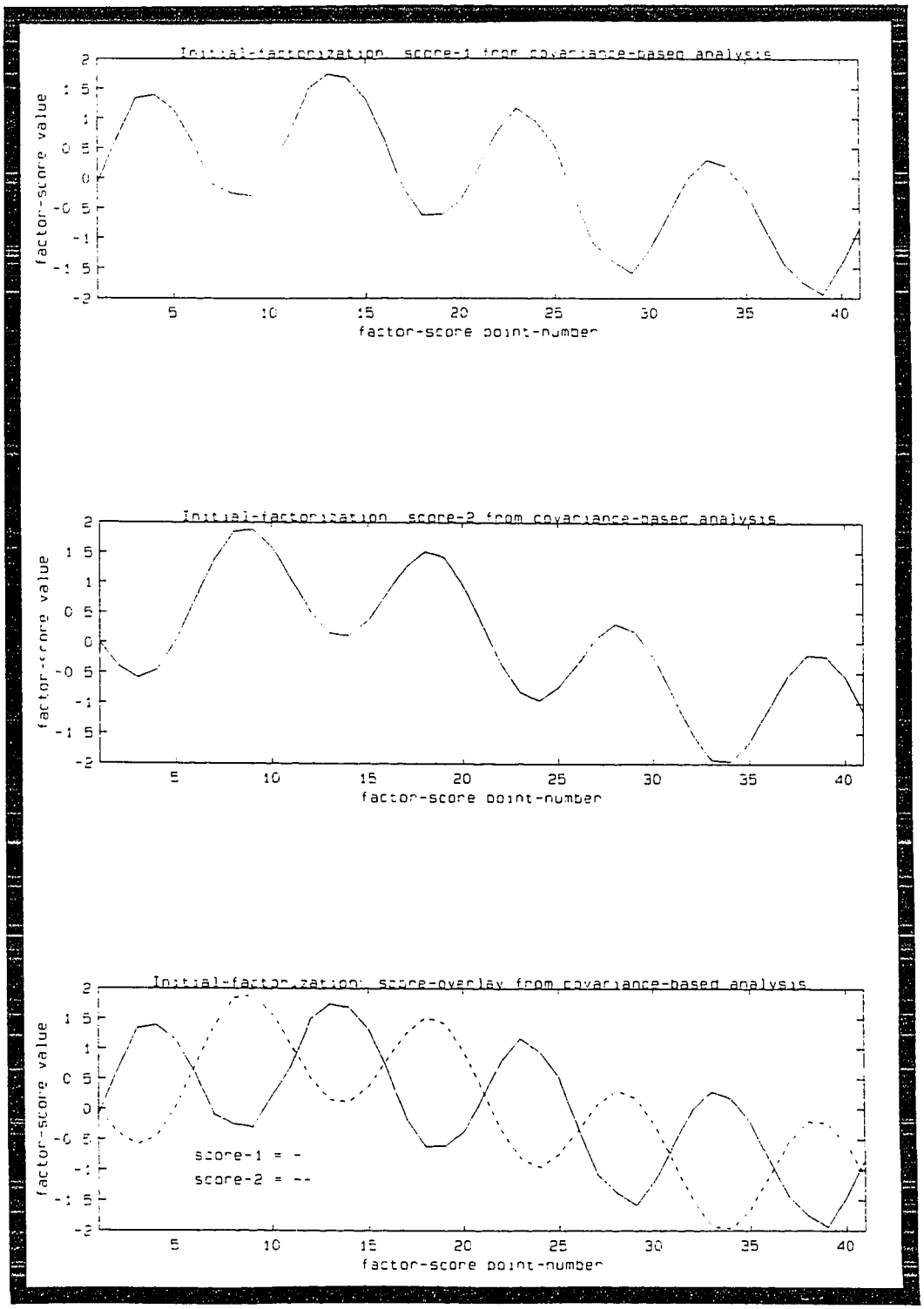


FIGURE C-7: Unrotated-scores from initial-factorization of the R-Mode factor analysis example problem covariance-matrix.

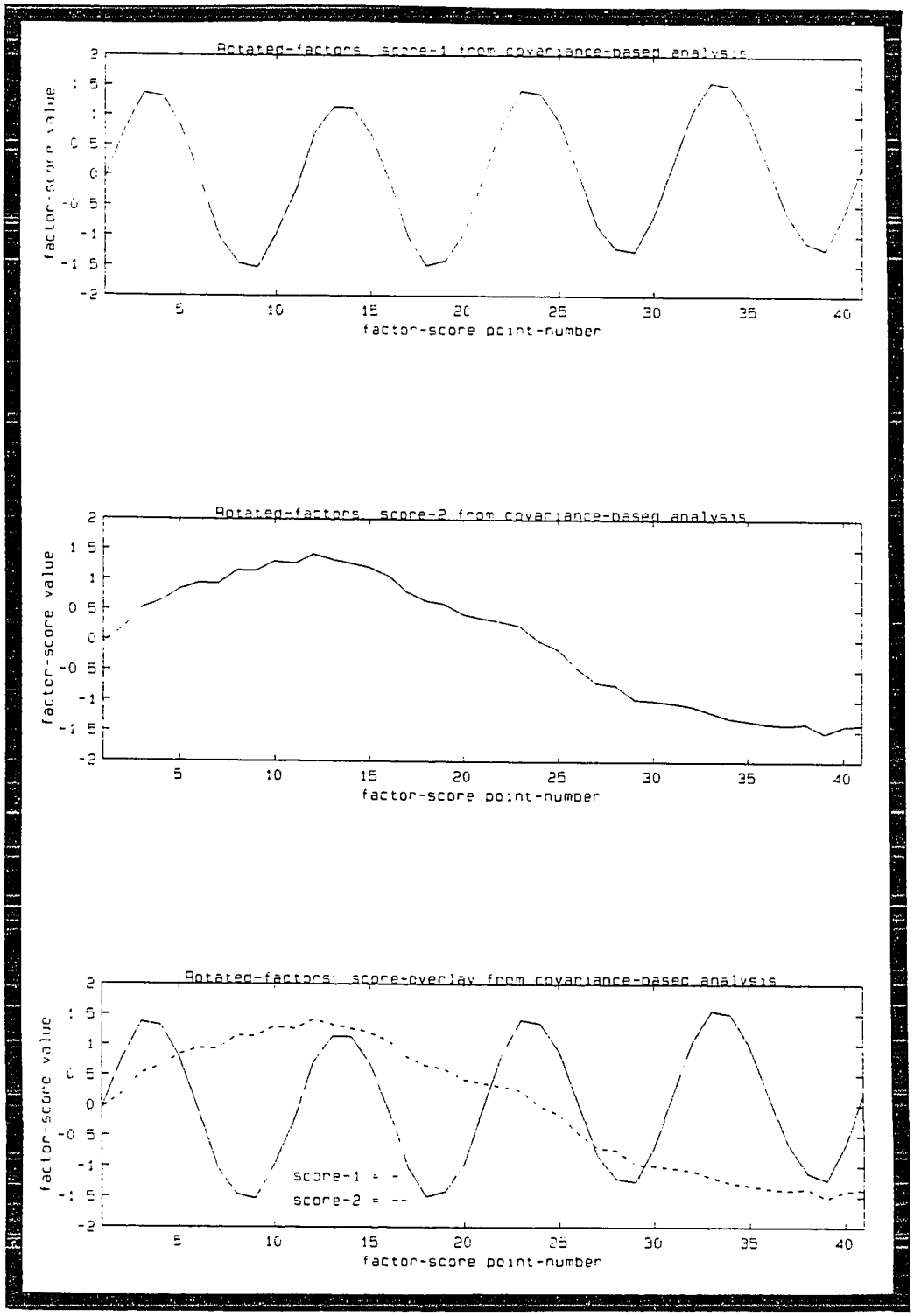


FIGURE C-8: Rotated-scores from the R-Mode factor analysis example problem covariance-matrix.

observed variables as two sinusoids that explain close to one-half the observed variance each, and look something like the original variables.

Factor-scores are ordered just as the eigenvalues and eigenvectors are ordered. We infer the form of the underlying variable with the largest influence on variance in the data matrix from the plot of the first factor-scores variable. The plot of the second factor-scores variable gives the form of the underlying-structure variable that explains the second-greatest amount of variance in the data matrix. And so on. Further, when we use <u>covariance</u> as our similarity index, we <u>preserve amplitude information</u>. So, in this case, if we apply the factor-loadings to the factor-scores we get back estimates of a factor's contribution to the model of the deviate-scores matrix. Summing these contributions for each variable gives us the complete model-estimate for each variable. That is, the sums $a_{i1}F_1 + a_{i2}F_2$ give us, in our two-factor case, the model-estimates to our each of our observed-variables, $\overline{X}_i$. In this case, 100 percent of the variance was explained, so our estimates are identical to the original-variables. This does not often happen with real data.

Scale greatly affects variance-values. A variable with relatively small variation through its record may have a <u>large magnitude</u> variance-value if the variable itself is expressed in large magnitude units. Conversely, a variable with a great deal of variation through its record may have a <u>small magnitude</u> variance-value if the variable itself is expressed in small magnitude units. As we saw in our preliminary

examination of the covariance-matrix, $\mathcal{S}$, via the matrix, $\mathcal{L}$, (having row-normalized covariance-values), the relative magnitudes of the values in the matrix determines how we interpret the form of the underlying-structure of the data-matrix. So, in a factor analysis procedure, a variable with small variation but large magnitude may (inappropriately) out-weigh the effects of a variable with large variation, but small magnitude. We combat this, in-part, by removing columns-means and performing the covariance-based analysis on the deviate-scores matrix, $\mathbf{Y}$, rather than on the original data matrix, $\mathbf{X}$. However, this is frequently not enough to eliminate the problem. Scaling of the data is also used to bring closer the relative-magnitudes of the variances and covariances of the variables. For example, in this thesis conductivity was converted to salinity solely to bring the magnitude of its variance closer to the magnitude of the variance determined for temperature and tidal-excursion. Such scaling problems are eliminated when the analysis is based on the correlation matrix of the data matrix.

R-Mode Factor-Analysis Using the Correlation Matrix

introduction

Use of the correlation matrix in factor-analysis eliminates potential bias due to scaling problems that can occur in a covariance-based analysis. We get a more true interpretation of the forms of our underlying-structure variables, and a more dependable expression of their

relative importance in explaining the variance in the observed variables. However, we lose the amplitude information that would enable us to generate model-estimates of our original variables. With R-mode factor analysis we are trying to find the form of 'believable' forcing-functions. So, in most studies a reliable estimate of the relative weight of forcing-functions in producing variance in the observed variables is more important than being able to reproduce the original variables from the analysis results. More often than not, the correlation-based analysis is used over the covariance-based analysis.

initial inspection

The correlation-matrix, $\mathcal{R}$, from this analysis is:

$$\mathcal{R} = \begin{bmatrix} 1.00 & 0.95 & 0.64 & 0.50 \\ 0.95 & 1.00 & 0.84 & 0.74 \\ 0.64 & 0.84 & 1.00 & 0.98 \\ 0.50 & 0.74 & 0.98 & 1.00 \end{bmatrix}. \qquad \text{(C-27)}$$

From $\mathcal{R}$ we can see correlations among our variables, $\bar{X}_i$, that basically match the associations we saw in $\mathcal{S}$. $\bar{X}_1$ and $\bar{X}_4$ are the least-well correlated of the observed-variables. The middle-variables are fairly well correlated with each other. However, $\bar{X}_3$ is very strongly correlated with $\bar{X}_4$ and poorly correlated with $\bar{X}_1$. While $\bar{X}_2$ is very strongly correlated with $\bar{X}_1$ and poorly correlated with $\bar{X}_4$. As was the case for inspection of the covariance-matrix, we might infer two major-factors with a graded application of these factors across our observed-variables.

initial-factoring and rotation

The correlation-based analysis is identical to the covariance-based analysis, except that factor-scores are determined using the normalized deviate-scores matrix. That is:

$$\boldsymbol{F} = \boldsymbol{Z}\boldsymbol{A}\boldsymbol{\Lambda}^{-1}, \qquad \text{(C-28)}$$

where:

$$\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{D}^{-1/2}, \qquad \text{(C-29)}$$

and $\boldsymbol{D}$ is a diagonal matrix whose diagonal elements are the diagonal elements of the covariance matrix (i.e., the column-vector standard deviations from the deviate scores matrix). From the correlation based analysis of the example data matrix, using notation identical to that of the covariance based analysis, we see:

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0.461 & -0.661 & 0.009 & 0.592 \\ 0.529 & -0.314 & -0.206 & -0.761 \\ 0.521 & 0.375 & 0.767 & 0.000 \\ 0.485 & 0.569 & -0.608 & 0.266 \end{bmatrix}, \qquad \text{(C-30)}$$

$$\lambda_i = 3.34,\ 0.658,\ 2.48\text{x}10^{-9},\ 8.36\text{x}10^{-18}. \qquad \text{(C-31)}$$

To explain at least 97.5 percent of the variance in the observed variables we retain the first two eigenvalues. $\overline{\lambda}_1 = 3.34$ explains 83.5 percent of the variance in the observed variables and $\overline{\lambda}_2 = 0.685$ explains 16.5 percent of that variance, for a total of 100 percent of the observed variance. The remaining results were:

$$\vec{\Lambda}_1 = \begin{bmatrix} 0.462 \\ 0.529 \\ 0.521 \\ 0.485 \end{bmatrix}, \quad \lambda_1 = 3.34, \tag{C-32}$$

$$\vec{\Lambda}_2 = \begin{bmatrix} -0.661 \\ -0.314 \\ 0.375 \\ 0.569 \end{bmatrix}, \quad \lambda_2 = 0.658, \tag{C-33}$$

$$\boldsymbol{U} = \begin{bmatrix} 0.528 & -0.660 \\ 0.511 & -0.241 \\ 0.484 & 0.388 \\ 0.475 & 0.597 \end{bmatrix}, \tag{C-34}$$

$$\Gamma = \begin{bmatrix} 3.34 & 0.000 \\ 0.00 & 0.658 \end{bmatrix}, \tag{C-35}$$

$$\boldsymbol{A} = \begin{bmatrix} 0.84 & -0.54 \\ 0.97 & -0.26 \\ 0.95 & 0.30 \\ 0.89 & 0.46 \end{bmatrix}, \tag{C-36}$$

$$
\boldsymbol{F} = \begin{bmatrix}
-0.0729 & -0.0042 \\
0.6477 & -0.4252 \\
1.3121 & -0.6412 \\
1.3690 & -0.5190 \\
1.1531 & -0.0171 \\
0.6273 & 0.7040 \\
-0.0360 & 1.3875 \\
-0.1656 & 1.8520 \\
-0.2123 & 1.8965 \\
0.2904 & 1.5846 \\
0.7564 & 1.0321 \\
1.5083 & 0.4567 \\
1.7450 & 0.0832 \\
1.6934 & 0.0416 \\
1.3392 & 0.3189 \\
0.6897 & 0.8056 \\
-0.1193 & 1.2797 \\
-0.5493 & 1.5430 \\
-0.5395 & 1.4448 \\
-0.3400 & 0.9784 \\
0.2134 & 0.2950 \\
0.7850 & -0.3928 \\
1.1347 & -0.8677 \\
0.9096 & -0.9977 \\
0.4988 & -0.7595 \\
-0.3272 & -0.3382 \\
-1.0748 & 0.1231 \\
-1.3648 & 0.3723 \\
-1.5698 & 0.2535 \\
-1.1898 & -0.1887 \\
-0.6214 & -0.8507 \\
-0.0697 & -1.5082 \\
0.2001 & -1.9506 \\
0.0927 & -1.9764 \\
-0.2871 & -1.6276 \\
-0.8991 & -1.0508 \\
-1.4637 & -0.4821 \\
-1.7603 & -0.1294 \\
-1.9529 & -0.1436 \\
-1.4646 & -0.4978 \\
-0.8858 & -1.0813
\end{bmatrix}, \qquad \text{(C-37)}
$$

$$\mathbf{D} = \begin{bmatrix} 61.41 & 0.00 & 0.00 & 0.00 \\ 0.00 & 52.66 & 0.00 & 0.00 \\ 0.00 & 0.00 & 52.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 55.34 \end{bmatrix}, \tag{C-38}$$

$$\mathbf{C} = \begin{bmatrix} 0.83 & -0.53 \\ 0.96 & -0.25 \\ 0.94 & 0.30 \\ 0.88 & 0.46 \end{bmatrix}, \tag{C-39}$$

and,

$$h_i^2 = 1.00. \tag{C-40}$$

The correlation-based results after rotation are:

$$\mathbf{T} = \begin{bmatrix} 0.7299 & -0.6835 \\ 0.6991 & 0.7299 \end{bmatrix}, \tag{C-41}$$

$$\mathbf{A} = \begin{bmatrix} 0.25 & -0.97 \\ 0.53 & -0.85 \\ 0.90 & -0.43 \\ 0.96 & -0.27 \end{bmatrix}, \tag{C-42}$$

$$h_i^2 = 1.00, \tag{C-43}$$

$$\mathbf{V} = \begin{bmatrix} -0.115 & -0.798 \\ 0.171 & -0.591 \\ 0.636 & -0.083 \\ 0.743 & 0.084 \end{bmatrix}, \qquad \text{(C-44)}$$

$\lambda_1 = 1.83$, explaining 52.5 percent of the observed variance, $\lambda_2 = 1.65$, explaining 47.5 percent of the observed variance,

$$
\underline{\mathcal{E}} = \begin{bmatrix}
-0.0561 & 0.0467 \\
0.1822 & -0.7532 \\
0.5196 & -0.1365 \\
0.6446 & -1.3146 \\
0.8300 & -0.8006 \\
0.9391 & 0.0851 \\
-0.9220 & 1.0374 \\
1.1450 & 1.4650 \\
1.1412 & 1.5295 \\
1.2950 & 0.9582 \\
1.2576 & 0.2364 \\
1.4132 & -0.6975 \\
1.3287 & -1.1341 \\
1.2646 & -1.1271 \\
1.1955 & -0.6825 \\
-1.0541 & 0.1166 \\
0.7875 & 1.0157 \\
0.6536 & 1.5018 \\
0.5937 & 1.4233 \\
0.4206 & 0.9466 \\
0.3574 & 0.0694 \\
0.3045 & -0.8233 \\
0.2352 & -1.4089 \\
-0.0180 & -1.3500 \\
-0.1550 & -0.8953 \\
-0.4700 & -0.0232 \\
-0.7004 & 0.8245 \\
-0.7418 & 1.2046 \\
-0.9726 & 1.2581 \\
-0.9975 & 0.6755 \\
-1.0351 & -0.1962 \\
-1.0818 & -1.0533 \\
-1.1872 & -1.5606 \\
-1.2383 & -1.5060 \\
-1.3220 & -0.9918 \\
-1.3745 & -0.1525 \\
-1.3979 & 0.6485 \\
-1.3734 & 1.1087 \\
-1.5237 & 1.2300 \\
-1.4093 & 0.6377 \\
-1.3857 & -1.8382
\end{bmatrix}, \qquad \text{(C-45)}
$$

and,

$$\mathcal{L} = \begin{bmatrix} 0.25 & -0.96 \\ 0.52 & -0.84 \\ 0.89 & -0.42 \\ 0.95 & -0.26 \end{bmatrix}.$$

Factor Interpretation

Factors scores from the initial factorization of the correlation matrix (Figure C-9) are very nearly identical to those produced by the initial factorization of the covariance matrix (Figure C-7). The rotated factor-scores from the correlation based analysis (Figure C-10) are very similar to their counterparts from the covariance based analysis (Figure C-8). The arguments for which set is more representative of the form of underlying factor-variables are the same for either analysis; leading us to choose the rotated-set.

The important differences between the analyses are associated with the rotated factor-scores. The ordering of the rotated factor-scores is reversed in this analysis relative to the covariance based analysis. With scaling removed, the longer-period sinusoid explains more of the total variance in the observed variables than does the shorter-period sinusoid. And, the phase of the shorter-period, second rotated factor-score from the correlations matrix is inverted relative to the shorter-period, first rotated factor-score from the covariance matrix. The change in ordering

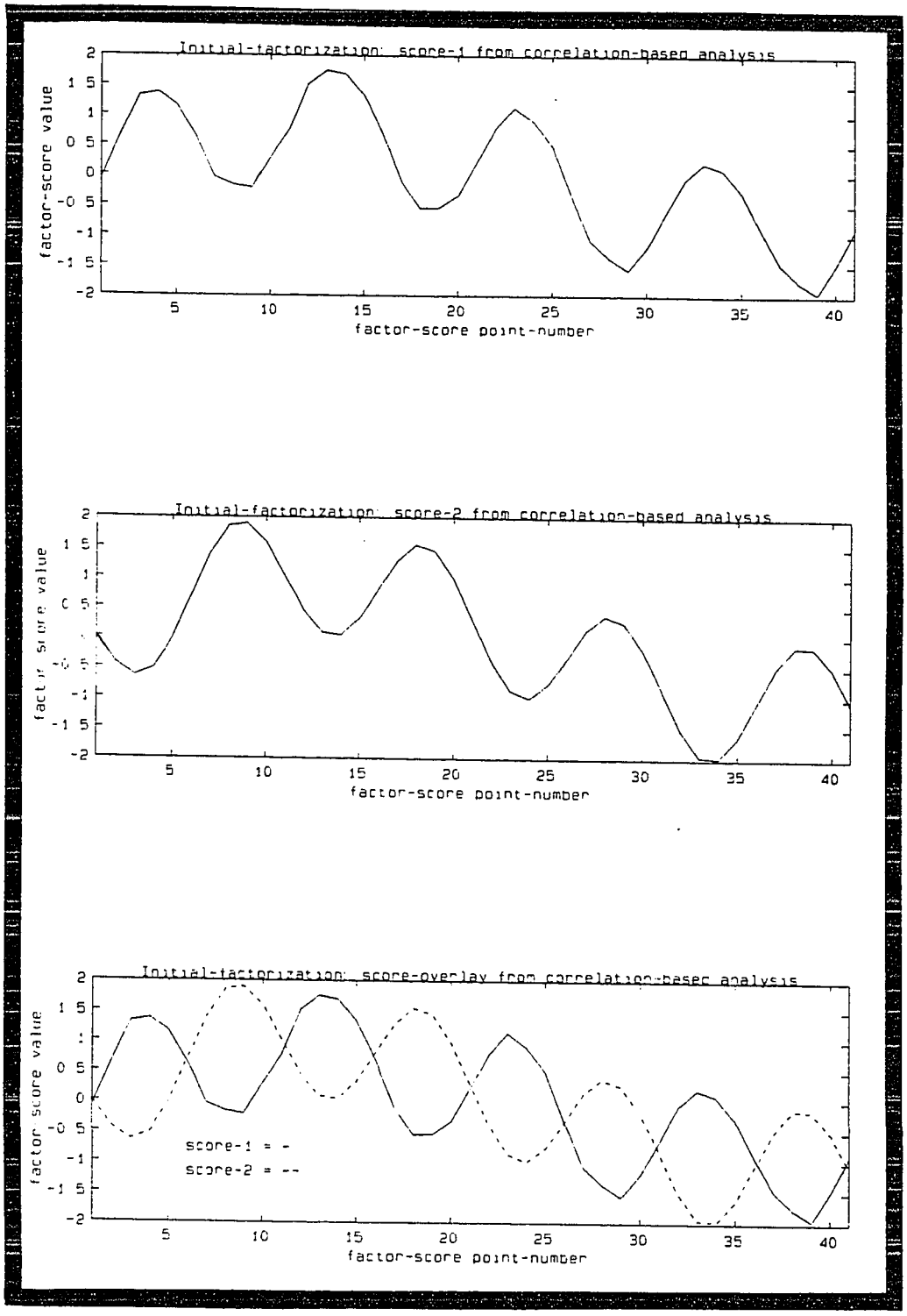


FIGURE C-9: Unrotated-scores from initial-factorization of the R-Mode factor analysis example problem correlation-matrix.

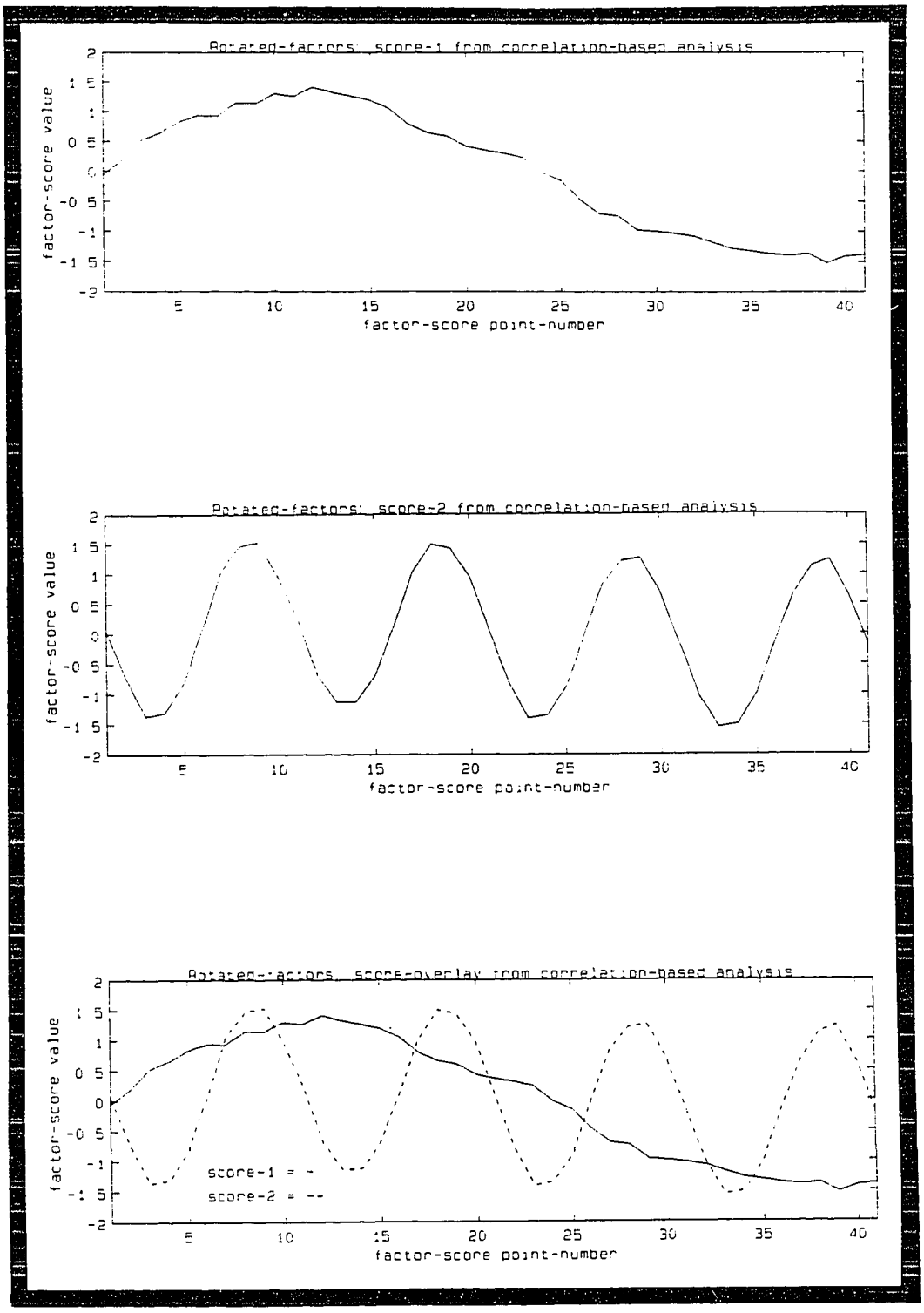


FIGURE C-10: Rotated-scores from the R-Mode factor analysis example problem correlation-matrix.

appears appropriate when we look at the original variables (Figure C-6). All of them seem to have a slight 'hump' that could be associated with the form of the longer-period factor-score. The reason for the inversion of phase for the shorter-period factor-score is a little more difficult to see immediately. This inversion puts its phase opposite to the apparent phase of the shorter-period aspect of the original variables (i.e., where the variables are 'up,' the shorter-period, rotated factor-score from the correlations analysis is 'down'). But, this is, in fact, the required phase for factors of this form to produce the data matrix variables as weighted-sums of the factors. The negative score-values from the shorter-period score 'pull-down' the positive score-values from the longer-period score, and vice-versa. The forms of the rotated factor-scores from the correlations based analysis are what is required to generate the proper form, as a weighted-sum, of the original variables in terms of the placement of their peaks and troughs, when the variables are all given equal-weight in terms of their individual variances. The 'wrong phase' being generated in the covariance based analysis is an artifact of scaling differences between the variances of the original variables.

APPENDIX D

TIME-DOMAIN FILTERING

Introduction

Filtering means removal of some part of a collection of items, where the separated portions are 'different' by some pre-established criterion. Whether one is removing sediment from a water sample, noise from a stereo system, or the tidal component from a long-term time-series data set, the process is referred to as filtering.

Two methods are commonly used to filter time-series data; applying weights to the original data (convolution or time-domain filtering), or applying weights to the Fourier transform of the data (transform 'windowing' or 'tapering' (spectral filtering)). The two techniques are equivalent, but depending on the data set being analyzed, may be very different in their ease and efficiency of use (Evans 1985). In this appendix we restrict ourselves to time-domain filtering. This means that during the process we operate algebraically on the time-series data themselves, and do not involve calculations of frequency or variance in the filtering operation.

There are many reasons for filtering time-series data, ranging from removal of 'noise' in a record, to establishing a new time-step between points in the series. More to the oceanographic point, there is an increasing interest in long-term (residual) circulation in estuaries (Walters and Heston 1982). Thus, an important use for data filtering in estuarine research is the removal of diurnal and shorter-period tides from a time-series record. Tides and inertial motions usually are a strong 'high frequency noise' in current meter, temperature, or sea level records used to study low frequency motions (Thompson 1983). Removal of 'high frequency' components means performing a low-pass filter operation.

All time-domain filter operations used in this thesis were low-pass functions. The discussions that follow describe time-domain filtering mainly from the point of view of low-pass applications. The symmetric, least-squares-fit, quadratic-weight filter used to prepare data for analysis is employed as an example throughout the general discussion of low-pass filters that follows. The Godin-filter utilized for analyses is described below in its own section.

Low-Pass Filtering

Introduction

Low-pass data filtering means using some technique to remove high frequency components from a data set. That is, only low frequency components are 'allowed to pass,' through the filtering process. A low-pass filter is any 'device' that will separate high and low frequency components from a signal, discarding high frequency parts in favor of the low frequencies. A low-pass device may be a physical unit or a mathematical abstraction. Here we concentrate on the mathematical filter, which may be considered a model of some physical system.

The terms 'high frequency' and 'low frequency' are relative. Shortwave radio operators consider a frequency of 500-kilocycles per second to be low. A climatologist may look at frequencies near one-cycle per week as high. In this thesis a storm-influenced tidal record was examined. So, we take more the climatologist's point of view. In the discussions that follow, frequencies on the order of cycles per hour are considered 'high.'

The Nyquist-Criterion and Aliasing

Although we are discussing the time-domain, what actually happens during a time-domain filtering process may be more easily understood

after an excursion into the frequency-domain and a description of the Nyquist-criterion. The Nyquist-criterion states for a discretely sampled, equal-interval time-series, the highest frequency that may be reproduced from those data is one-half the sampling frequency used to collect the data. For example, if data are sampled with a frequency of one-reading per 30 minutes, the highest frequency one may reliably resolve from those data is one-cycle per hour. This frequency, one-half the sampling-frequency, is the Nyquist frequency.

The Nyquist frequency is also known as the folding-frequency. Folding refers to the apparent introduction of false frequency components into a record when discrete, equally-spaced sampling is used. Specific frequencies, dependent upon sample-rate, will be 'folded back,' giving the appearance of some lower frequency in a time-series record. These so-called 'folded frequencies' are more commonly said to be aliased. High frequencies are aliased into low frequencies. Aliasing due to sampling, or simply 'aliasing,' is a result solely of the discrete sampling process (Hamming 1983).

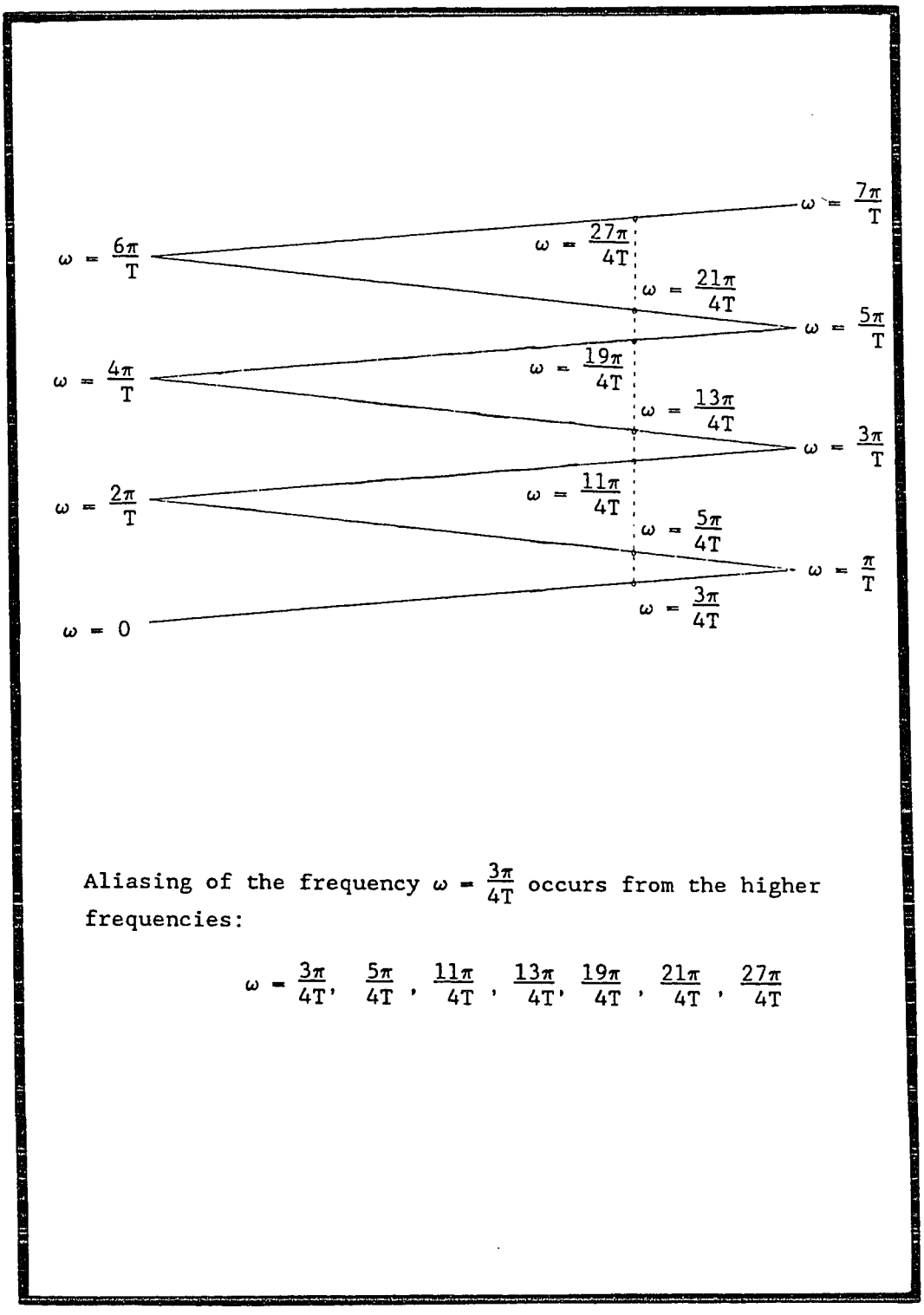
Uniform sampling can lead to large errors in recording a signal if the sampling period is too large. By the sampling theorem, in order to uniformly sample a continuous signal, g(t), with no loss of information the sampling period, T, must be selected so that $T \leq \pi/w_o$, where w_o is

the highest radian-frequency[1] in g(t).

If $g(t) = \sin(w_1 t+\theta)$ then to sample g(t) with no loss of information the sampling period, T, must be selected so that $0 \leq w_1 \leq \pi/T$. The resulting sampled-sequence will be a set of points $\{g(k)\} = \{\sin(kw_1T+\theta)\}$, k=1,2,... Now consider the continuous-signal $h(t) = \sin([w_1+2\pi N/T]t+\theta)$, where N is any positive-integer. If we sample h(t) with the same sampling-period we used to sample g(t) the resulting sampled-sequence is $\{h(k)\} = \{\sin([w_1+2\pi N/T]kT+\theta)\} = \{\sin(kw_1T+\theta+2\pi Nk)\}$. But, the term $2\pi Nk$ is just some number of complete 2π revolutions, and has no affect on calculated sine-values. So, $\{h(k)\} = \{\sin(kw_1T+\pi)\} = \{g(k)\}$, k=1,2,... That is, it is impossible to differentiate between {g(k)} and {h(k)}. So, the process of uniform-sampling is unable to distinguish between two sinusoids whose radian-frequencies have a sum or difference equal to an integral-multiple of $2\pi/T$. Effectively, all radian-frequencies are 'folded' into the interval $0 < w < \pi/T$ by the process of uniform-sampling (Figure D-1). The frequencies which fold into a specific frequency w_1, where $0 \leq w_1 \leq \pi/T$, are: w_1, $\pi/T-w_1$, $w_1+\pi/T$, $2\pi/T-w_1$, $w_1+2\pi/T$, ... (Cadzow 1973).

---

[1] Radian-Frequency, w, refers to counting cycles as full circular arcs in radians-pre-unit time. One revolution around a circle sweeps an arc of 2π radians. So conversion from radian-frequency to the more common expression for rotational-frequency, $f = 1/T$, is $f = 2\pi/w$. Which says that period $T = w/2\pi$.



Aliasing of the frequency $\omega = \frac{3\pi}{4T}$ occurs from the higher frequencies:

$$\omega = \frac{3\pi}{4T}, \frac{5\pi}{4T}, \frac{11\pi}{4T}, \frac{13\pi}{4T}, \frac{19\pi}{4T}, \frac{21\pi}{4T}, \frac{27\pi}{4T}$$

FIGURE D-1: First seven pleats of the Nyquist Folding Diagram.

If frequency-folding is to have no affect, then a uniformly-sampled signal must have no component with a radian-frequency greater than π/T, where T is the sampling-period. Unfortunately, any real signal will have some frequency-content across the entire spectrum. So, we can only select sampling-periods to minimize affects of frequency-folding. That is, T must be selected so that most of the signal's energy lies in components within the frequency-range $0 \leq w_1 \leq \pi/T$. Which says that the signal frequency-components with the greatest amplitudes must fall within the frequency-range $0 \leq w_1 \leq \pi/T$. If w_o is the highest frequency supplying significant energy to the signal, then we must select $T \leq \pi/w_o$. $w_o = 2\pi f_o$, where f_o is the rotational-frequency associated with w_o, therefore we must select $T \leq 1/2f_o$. This is a statement of the Nyquist-criterion. There must be at least two samples of the highest-frequency component present in a signal in each sample-period if there is to be no loss of information in the sampling-process. Or, only sinusoids with frequencies low enough for at least two samples to occur in each period will not be aliased (Figure D-2).

The Nyquist-criterion is usually applied from the point of view of determining the sampling rate required to accurately record phenomena up to some maximum expected frequency. However, through the idea of aliasing and the Nyquist-criterion, one can see that reducing the number of data points in a time-series while maintaining the overall time-span of the data reduces the maximum frequency that may be resolved from that series. This is because in doing so the sampling rate for the data set is

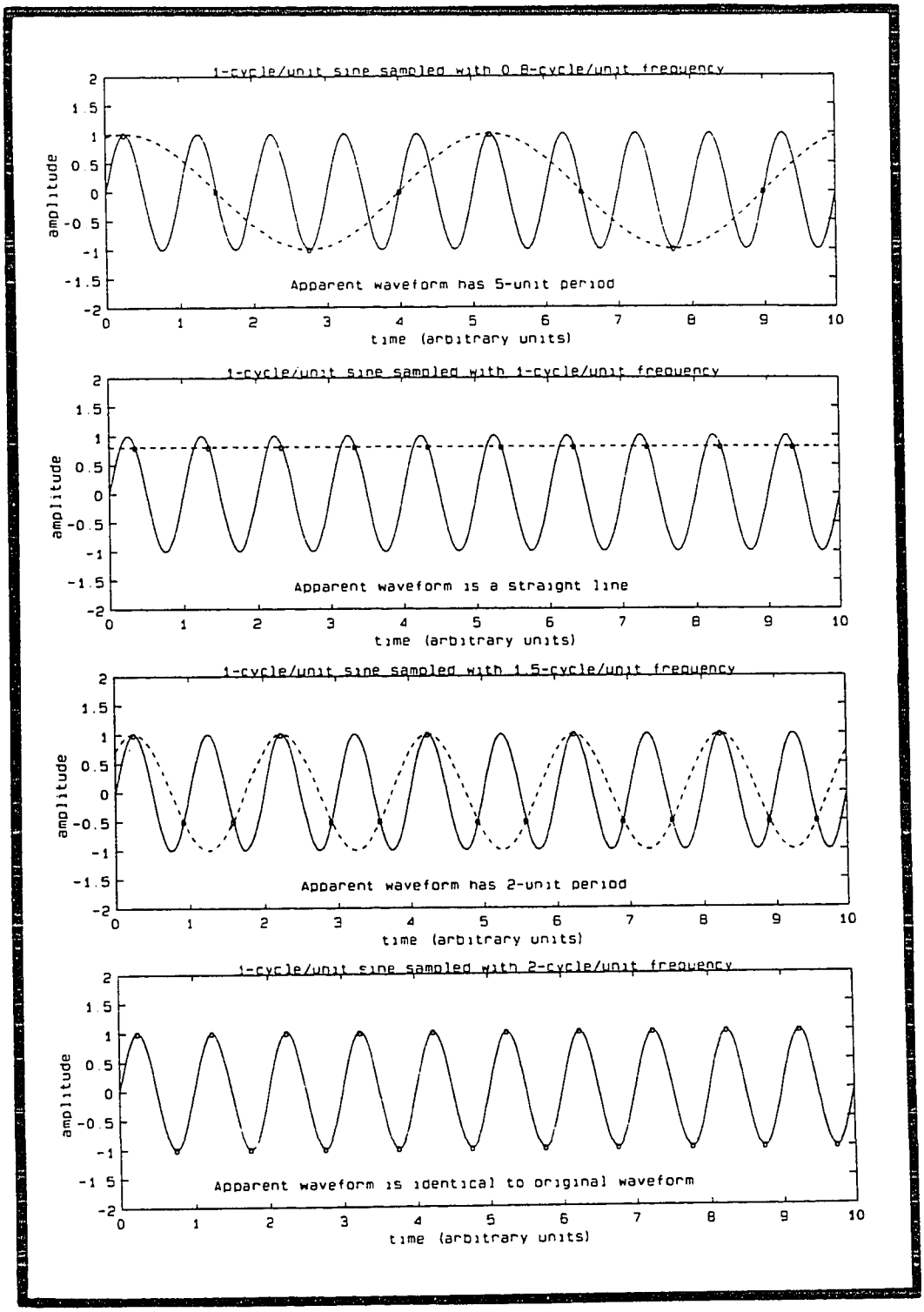


FIGURE D-2: Errors introduced by sampling a waveform at less than or equal to the Nyquist frequency.

effectively decreased. Since frequency and period are simple inverses, decreasing the maximum frequency increases the minimum period. Increasing the minimum period means that short-period information is lost. Hence, with due consideration for aliasing, removing data points as described above performs a low-pass filter-function on the time-series. Longer-period and lower-frequency are synonymous, so we speak of a low-(frequency)-pass filter even when we are working in the time-domain.

Arbitrary point removal from a time-series data set will give arbitrary filter results. However, when a logical point-removal scheme is followed, low-pass filtering can be obtained. The term decimation is often seen in conjunction with simple, logical elimination of data values. While to the ancient Roman soldier, decimation meant punishment of his unit for military cowardice, (by separating it into groups of ten members each, who then, by casting lots, selected a member to be killed by the remaining nine; typically by skinning him alive), it came to mean throwing out every tenth value from a data set. Decimation is now generally accepted to mean throwing out every other point from a data set. Eliminating every other point is a common technique for throwing away (low-pass filtering) data. Graphical techniques may also be used to remove data points for filtering.

An example of graphical low-pass filtering is removing the high-frequency component from the sum of a long-period, high-amplitude sinusoid and a short-period, low-amplitude sinusoid (Figure D-3). We

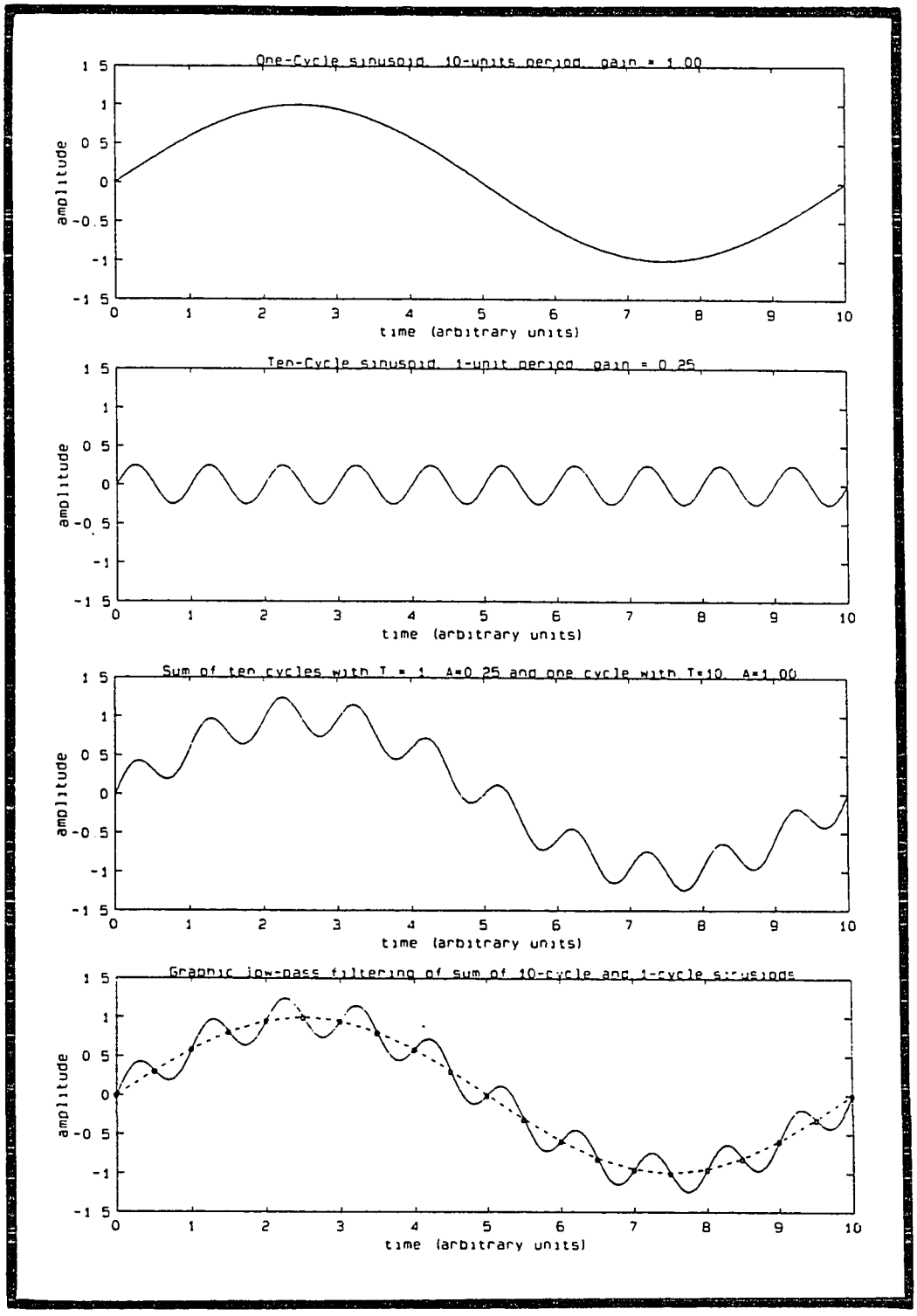


FIGURE D-3: Graphical low-pass filtering of the sum of two sinusoids.

simply draw a line following the middle of the high-frequency information, regardless of where in the plot it is placed relative to zero. This line gives us the form of the long-period component of the sum. If we plot only the points where our drawn-line intersects the plot of the summed sinusoids, we get back the long-period information. Effectively a low-pass filter has been applied to the original time-series. One feature of that record has been separated from it, while working graphically and only within the time-domain. Simple, logical elimination of data points, and no variance or frequency analysis was employed to graphically separate the long-period component from the original time-series.

A sum (superposition) of sinusoids of different periods, amplitudes and phases can be found to reproduce any complex waveform recorded in a data set. Tides are no exception. A typical mixed-tide record (Figure D-4), as might be observed on the California coast, shows several periods that are readily discernable to the naked eye. The most obvious feature of such records is the approximately 12-hour-and-25-minute semidiurnal cycle of high-water to low-water. One can also see a diurnal cycle over this high-to-low period as two different highs and two different lows with a period near 24-hours-and-50-minutes. And, if the record is long enough these two periods are seen to be modulated by the 'spring-neap cycle,' where the overall amplitude of the record varies with about a two-week cycle. To separate the components from such a record by simple decimation or graphical techniques would be very difficult. Underlying the record are at least six significant components related to

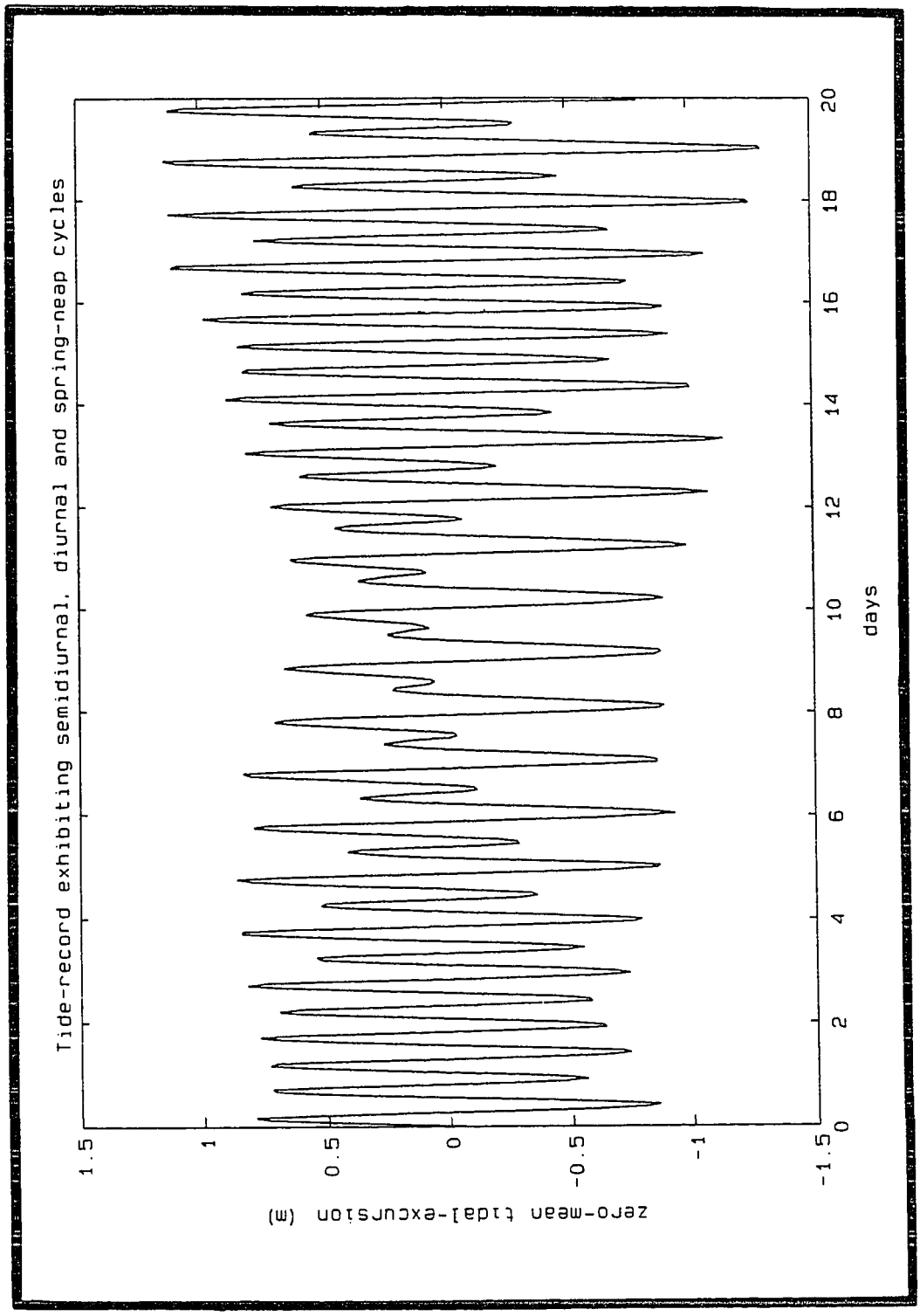


FIGURE D-4: Typical mixed-tide record for the California Coast showing semidiurnal, diurnal, and spring-neap cycles.

the rotation and orbit of the earth and the phase of the moon. For complex records, numerical calculations are required for any practical filtering process.

Averaging and Convolution

A simple average over the length of a time-series record is, in fact, a form of data filter. Admittedly it is a rather severe filter. Averaging reduces a time-series to one point with an associated time corresponding to the midpoint time of the original data set.

Superposition of sinusoids may be used to represent a complex waveform. Its average value represents the component in that sum with a frequency of zero. So, the average is a sinusoid with infinitely long period. This appears effectively as a straight, horizontal line for the duration of any finite length record. Removing their average shifts the individual values of a data set so that relative-amplitudes are maintained, while centering the values about zero. So, the average also represents a fixed-offset from zero for a data-record.

Forming simple averages through a data set over a specified time interval is a common time-domain filtering technique. For example, forming daily-averages of a parameter over a year is low-pass filtering of that parameter; expressing the offset from zero for the record on a day-by-day basis. In fact, most time-domain filtering techniques are

based on the calculation of multiple averages from a data-set. Generally, some multiplier (weight) is applied to the values used to form these averages. The process of forming weighted averages is more formally referred to as convolution. It is the idea of convolution that is central to time-domain filtering.

Recursive and Nonrecursive Filters

A filter is recursive (or iterative) if its output at a time-step, k, depends on some or all of the outputs for the previous time-steps k-1, k-2, ... , 0. Recursive filters have many applications, particularly in numerical prediction and approximation techniques. They were not used in this thesis, and are described no further in this appendix. A nonrecursive filter does not depend on previous outputs. It uses values as they are expressed in the input-sequence and generates its outputs from them.

Our symmetric, least-squares-fit, quadratic-weight data-preparation filter falls into the category of nonrecursive filters. Nine points from the input-sequence are weighted and summed to generate the output-sequence values. In general a filter of this type will be used to generate an output-sequence value relative to each input-sequence value. Some points on either end of the output-sequence will end up 'thrown out' relative to ends of the input-sequence, due to filter overlap. If we apply our filter in this manner, and step through the input-sequence

point-by-point, rather than four points at a time as we did in preparing the data of this thesis, the output-sequence loses just four points from each end relative to the input-sequence. While it would not be apparent we have decreased the sample-interval in the output-sequence, we still have performed a low-pass filter operation as a convolution-summation of the input-sequence. We will discuss this later. What is important to note here is in stepping the nonrecursive-filter through the input-data at some interval other than one time-step, the generated output-sequence values are the same as the corresponding points in the single time-step output-sequence. This is why we can use a nonrecursive filter to adjust a sequence time-step.

A nonrecursive filter can be low-, band-, or high-pass. If we use such a filter to increase the time-step in a data-set, then we must use a low-pass filter. This is because of aliasing. If we increase the time-step, then by the Nyquist-criterion we have reduced the maximum frequency that can be recovered from the sampled data set. For example, our original time-series data was 15-minute interval information, and the filtered data set is a one-hour interval sequence. This means frequencies up to two-cycles/hour may be recovered from the original-data, but filtered-data may be used to recover frequencies only up to one-half cycle/hour. At some point in simply eliminating values we could overstep the Nyquist-criterion boundary in an input data set, and aliasing would become a problem in the output-sequence. So, we use a nonrecursive, low-pass filter to increase a time-step, rather than simple

data-point elimination. This, along with increasing the time-step, reduces the energy from higher frequencies, which helps keep aliasing to a minimum.

Phase Distortion and Roll Off

Arbitrary application of even a well established filtering process to a data set can lead to undesirable results. Filters have effects beyond reducing the number of points in, or changing the period composition of, a data set. Two important considerations for a filter are the amount of phase distortion it produces, and its frequency roll off characteristics.

The relative placement of peaks and troughs for two periodic waveforms is defined as the phase difference between the waveforms. Phase difference, along with period and amplitude, determines the result of summing simple periodic components to form a complex waveform. A filter can affect the phase of the components it allows to pass relative to their original phase placements in the input record. For example, a noise filter in a stereo system may delay passing some frequencies more than it delays other frequencies, resulting in a relative phase shift between the output frequencies. Such an alteration in phase information can lead to a difference in form between the input and output of a filter (and to unpleasant sound). This form of difference is referred to as phase distortion (Figure D-5). The amount of phase distortion in its output is one of the factors determining what is a 'good' stereo system

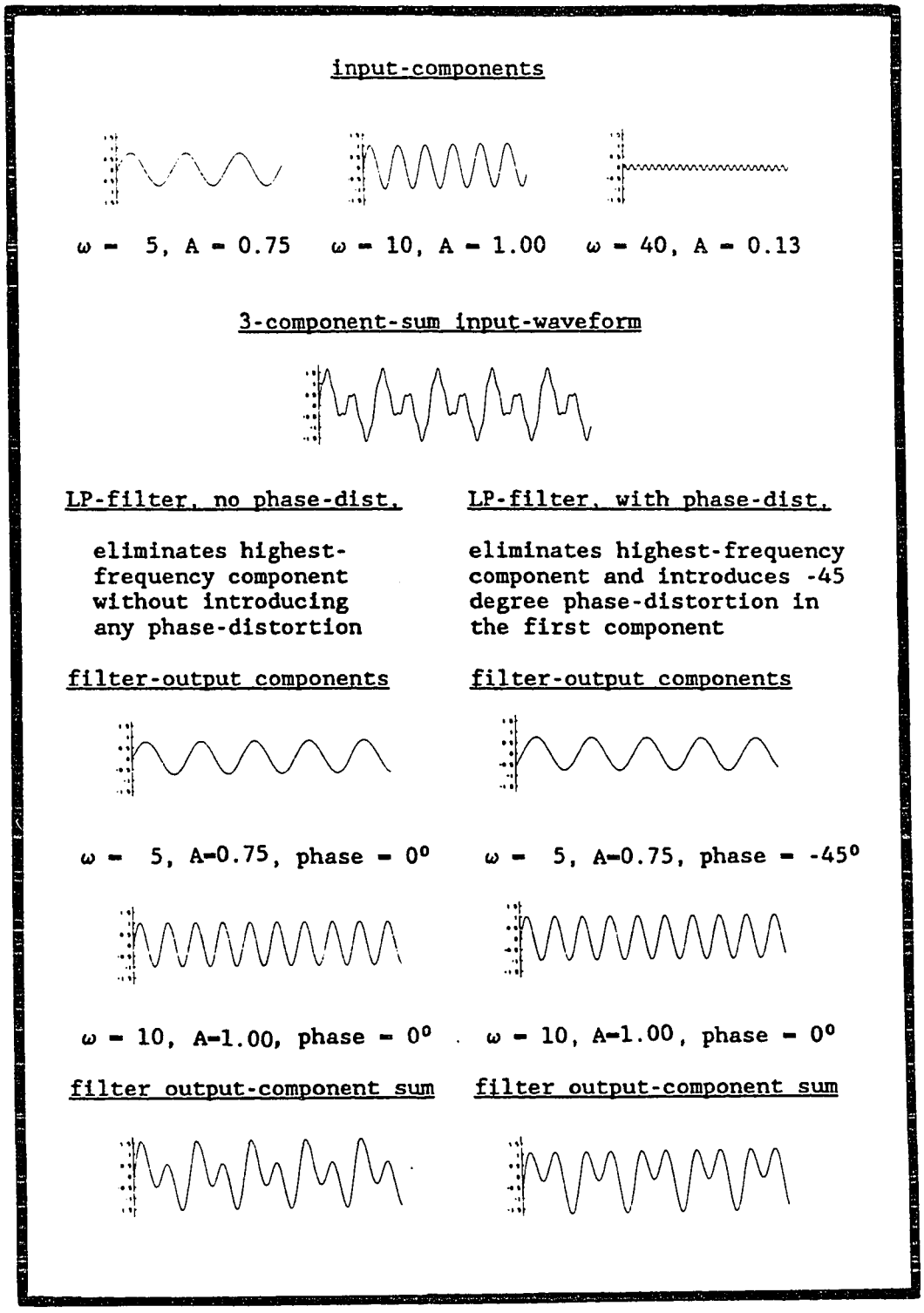


FIGURE D-5: Filter-introduced phase-distortion.

and what is a 'cheap' stereo system. A high-quality sound system has output filters that introduce only a small amount of phase distortion. Obviously, one would desire a 'high-quality' filter to retain proper phase relations between components separated from oceanographic-records by filtering.

Ideally a filter would send directly to its output all components it is designed to pass, without affecting the amplitude or phase of those components, and not pass any portion of those components it is designed to block. In the time-domain one can consider this ideal (non-amplifying) filter as a weighted sum of the periodic components from the complex waveform being filtered. The passed components are multiplied by one, and the blocked components are multiplied by zero. Note this means the ideal filter may be considered as a multiplication of the input record by a unit step-function in the frequency-domain (Figure D-6). Convolution in the time-domain is equivalent to multiplication in the frequency-domain.

Filters are not ideal. Besides the amount of phase distortion introduced, the degree to which undesired frequencies are passed and the amount desired frequencies are attenuated must be considered. This undesired passing and attenuation is a filter characteristic referred as frequency roll off. It is due to the frequency-domain description of a filter being not quite a step-function. A typical low-pass filter-function has a sloped region between the band of completely passed frequencies and the band of completely blocked frequencies. Components of the input

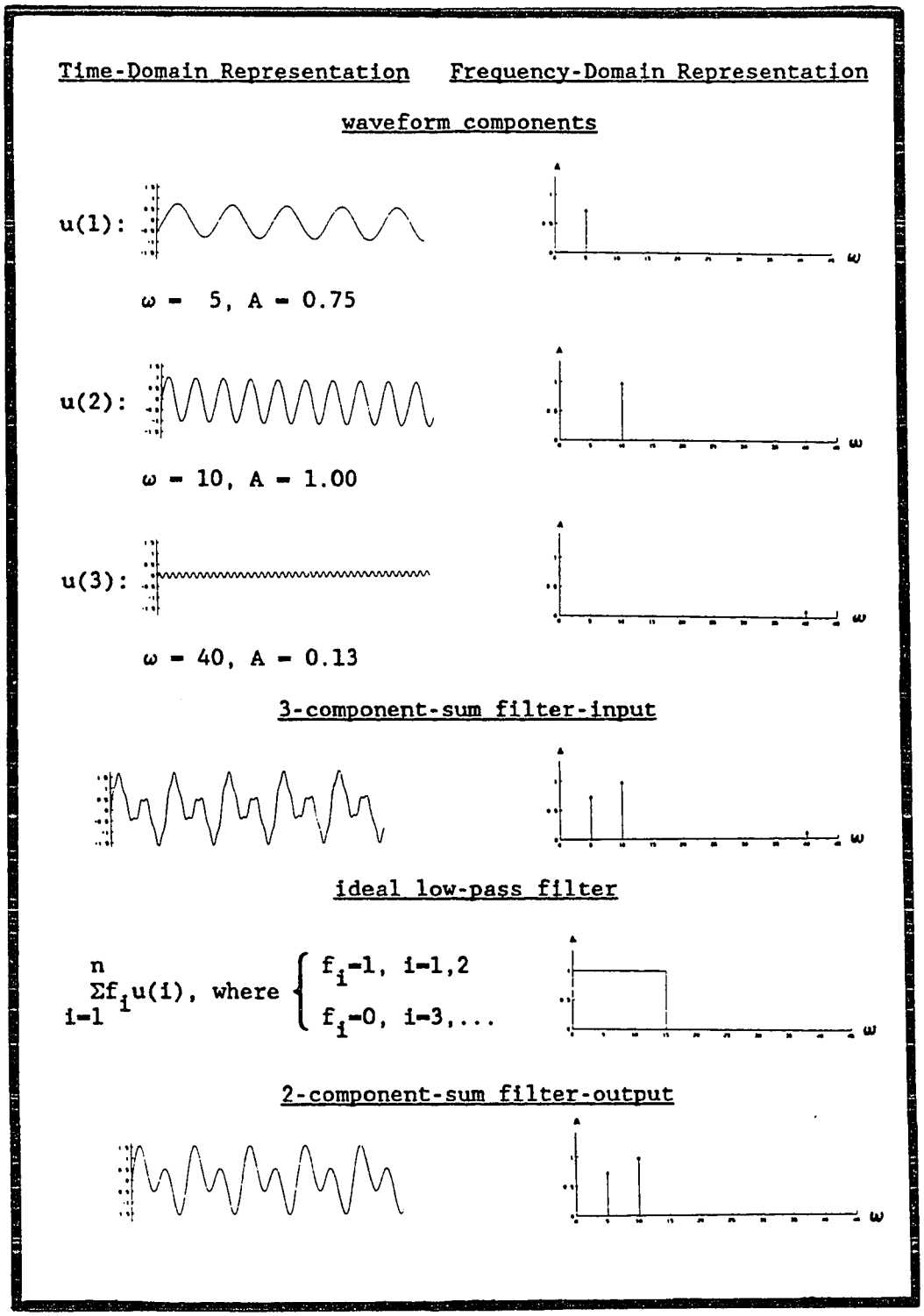


FIGURE D-6: Ideal low-pass filter representation in the time and frequency domains.

signal having frequencies covered by this sloped region are multiplied by some value less than one, but greater than zero. Hence, they are neither completely passed or totally blocked (Figure D-7).

There are a number of methods available to combat phase distortion and undesirable roll off characteristics in filters. To say how these techniques work requires a more rigorous description of how filters themselves work. In the sections that follows, the data-preconditioning filter used in this thesis is analyzed in order to describe low-pass filtering in more detail. In the process, the output results of using this symmetric, quadratic-weight convolution-filter are determined and summarized.

Low-Pass Filters

Introduction

By the definition of filtering as reduction in the number of points in a data set, it is clear why a simple average is a filter. A data set, no matter how large, is reduced to one value by averaging. More sophisticated filtering techniques involve averaging segments of a data set sequentially, and applying weights within the filter. By previous definition, these more sophisticated techniques are convolutions (as are simple averages). An application of this type of filtering was presented as a data-preconditioning technique in Appendix-A. A nine-point,

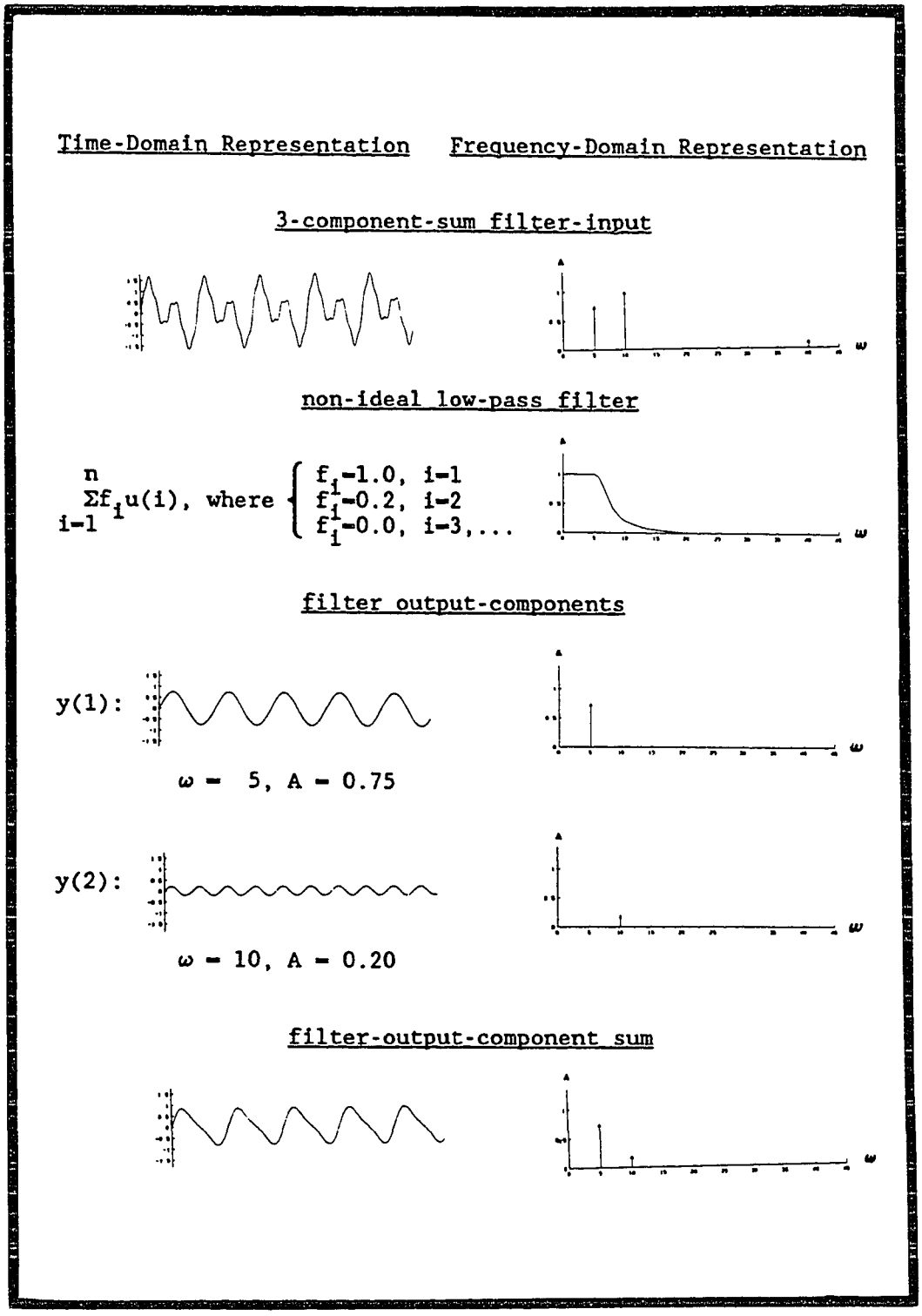


FIGURE D-7: Non-ideal low-pass filter representation in the time and frequency domains.

symmetric, quadratic-weight filter was stepped through the data to convert the information from a fifteen-minute interval set into a one-hour interval set.

The first thing to note about the data-preconditioning filter is that it is symmetric. The filter has an odd number of components, and corresponding weights on either side of the central-weight are identical. A symmetric-filter preserves phase information. That is, it produces no phase distortion in output-records. The next thing to note is that the sum of the filter-weights is one (i.e., the filter is normalized). A constraint for correctly passing low frequencies through a filter is that the weights sum to one (Thompson 1983). Given a symmetric filter who's weights sum to one, the number, spacing, and values of the weights determine features such as its roll off characteristics. In order to find roll off characteristics based on filter-weight information we must, once again, step out of the time-domain.

The Z-Transform

The z-transform has properties which enable one to solve linear difference-equations using simple algebraic manipulations. Thus, the z-transform lends itself well to investigation of discrete linear-systems, such as those represented by application of a convolution-summation filter to a discretely sampled data-set. Discrete linear-systems are characterized by linear difference-equations. The z-transform converts difference

equations into algebraic equations. So, given a convolution-summation filter and input-sequence, we can convert the implied difference equations into algebraic equations and generate the z-transform of the filtered output-sequence. From this, using tables or the inverse-z-transform, the time-domain representation of the output-sequence can be generated. Here we do not use the z-transform to simplify generating output-sequences for our filter with arbitrary inputs. Rather, we use specific input-sequences which enable us to determine the characteristics of our filter.

Any transform is simply a rule for changing between two different forms of the same thing. The z-transform is a rule for converting a sequence of numbers into a function of the complex-variable, z. For a sequence of numbers {f(k)}, where k represents discrete time-values, and f(k)=0 for k<0, the z-transform of the sequence is defined as:

$$Z[f(k)] = F(z) = f(0) + \frac{f(1)}{z} + \frac{f(2)}{z^2} + \frac{f(3)}{z^3} + \frac{f(4)}{z^4} + \cdots \quad (D\text{-}1)$$

where z is an arbitrary complex-number[2]. This can be put into more compact summation notation:

$$F(z) = \sum_{k=0}^{\infty} f(k)z^{-k}. \quad (D\text{-}2)$$

---

[2] Here z is the rectangular form of the complex variable $z = x + iy$, where $i = \sqrt{-1}$

F(z) (D-2) is a sum of complex-numbers, therefore F(z) is a complex-number. {f(k)} is called the generating-sequence for F(z). The region of convergence for F(z) is defined as the set of complex-numbers {z} for which F(z) has a finite-magnitude. It is defined as a radius, R, where: UzU > R, and R depends on the generating sequence, {f(k)}.

convolution-summation property of the z-transform

The are several important properties of the z-transform with regard to study of linear-discrete-systems. In the discussions that follow we make use of the convolution-summation relationship of input and output signals for the z-transform.

Consider two sequences {y(k)} and {u(k)} where y(k) = u(k) = 0 for all k < 0. If {y(k)} is related to {u(k)} through a weighting-sequence {h(k)} such that:

$$\{y(k)\} = \{h(0)u(k) + h(1)u(k-1) + h(2)u(k-2) + \ldots\}, \quad \text{(D-3)}$$

that is, each y(k) is a weighted-sum of elements of {u(k)}, then:

$$Y(z) = H(z)U(z) \quad \text{(D-4)}$$

This is the expression of the convolution-summation property (D-4) of the z-transform. If, in the time-domain, one time-series can be

represented as a weighted sum of another time-series, then the z-transform of the weighted series is the product of the z-transform of the weighting-sequence and the z-transform of the original sequence.

transform pairs

Given a sequence {f(k)}, its z-transform (D-2) may be determined through application of the definition and properties of the z-transform. Given the convolution-summation property of z-transforms (D-4), and the definition of our example low-pass filter as a weighted-sum, or convolution (D-3), it becomes clear why the z-transform is useful in describing properties of filters. We may convert our input-series and the filter weight-sequence into their z-transform representations, multiply these transforms, and detransform the result to obtain the time-domain representation of the output for the filter. If we select inputs with specific properties, e.g., frequency-content, amplitude, phasing, then the z-transform can be used to determine the characteristics of the filter. Rather than perform the calculations each time a filter response is determined, it is convenient to have a listing of pairs of common sampled-functions and their z-transforms (Table D-1).

TABLE D-1: Common sampled-functions, and their discrete-sample equivalents with z-transform and radius of convergence.

f(t)	f(kT)	F(z) = Z[f(kT)]	R
1	1	$\frac{z}{z-1}$	1
t	kT	$\frac{Tz}{(z-1)^2}$	1
t^2	$(kT)^2$	$\frac{T^2z(z+1)}{(z-1)^3}$	1
t^3	$(kT)^3$	$\frac{T^3z(z^2+4z+1)}{(z-1)^4}$	1
e^{-at}	e^{-akT}	$\frac{z}{z-e^{-aT}}$	$\lvert e^{-aT} \rvert$
te^{-at}	kTe^{-akT}	$\frac{zTe^{-aT}}{(z-e^{-aT})^2}$	$\lvert e^{-aT} \rvert$
sinwt	sinkwT	$\frac{z\sin wt}{z^2-2\cos wT+1}$	1
coswt	coskwT	$\frac{z(z-\cos wT)}{z^2-2\cos wT+1}$	1

Transfer Functions

We stated that the filter of Appendix-A produces no phase distortion and correctly passes low frequencies. We still must concern ourselves with how amplitude information for high frequencies is transferred through the low-pass filtering-process. This transfer is determined by the filter's frequency roll off characteristics. The roll off characteristics for a filter can be determined by an examination of the transfer function for that filter.

The transfer function for any system is a general description of the affects of that system on an input-signal. A discrete-linear-system, such as our data-preconditioning filter, is governed by a linear difference-equation that relates its input and output-signals. As such we may make good use of the z-transform and its convolution-summation property to determine filter characteristics. The z-transform (D-2) of a weighting-sequence (D-3) as defined above while describing the convolution-summation property (D-4) is the system transfer-function. That is, the system weighting-sequence and the system transfer-function are z-transform pairs.

Given $Y(z) = H(z)U(z)$, it is always true that $H(z) = Y(z)/U(z)$. That is, the system transfer-function $H(z)$ is the ratio of the z-transform of the response $Y(z)$ to the z-transform of the input $U(z)$. Since $H(z)$ is a ratio, the locations of the roots of its numerator and denominator have

important relationships to the transfer characteristics for H(z). The roots of the numerator are referred to as the zeros of H(z). The roots of the denominator are referred to as the poles of H(z). If we desire a specific response from a system, we can design an H(z) for the poles and zeros necessary to give us that response. Conversely, given an H(z) we can analyze it for the locations of its poles and zeros and determine the response characteristics of the associated system.

Response Decomposition and Eigenfunctions

It can be shown for the case Y(z) = H(z)U(z), that the system transfer-function has the general form:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} \qquad \text{(D-5)}$$

where this function is an exclusive function of the a_i and b_i coefficients governing the system difference-equation. Also, for most input-signals of any practical interest the z-transform will be a ratio of polynomials in z:

$$U(z) = \frac{N(z)}{(z - q_1)(z - q_2) \cdots (z - q_s)} \qquad \text{(D-6)}$$

(Cadzow 1973). Thus, the response of the system will be:

$$Y(z) = \left[\frac{b(z - z_1)\cdots(z - z_m)}{(z - p_1)\cdots(z - p_n)}\right]\left[\frac{N(z)}{(z - q_1)\cdots(z - q_s)}\right]. \qquad (D\text{-}7)$$

We can determine the output-sequence {y(k)} (D-3) by performing a partial-fraction expansion (Appendix E) of its z-transform Y(z) (D-7) and performing the inverse z-transform for each of the resulting simple z-transforms. The expansion will be of the form:

$$Y(z) =$$

$$e_0 + \left[\frac{e_1 z}{z - q_1} + \frac{e_2 z}{z - q_2} + \cdots + \frac{e_s z}{z - q_s}\right] \qquad (D\text{-}8)$$

$$+ \left[\frac{d_1 z}{z - p_1} + \frac{d_2 z}{z - p_2} + \cdots + \frac{d_n z}{z - p_n}\right] \qquad (D\text{-}9).$$

We have decomposed the response signal into two parts called the input-signal-mode (D-8) and the system-mode (D-9). The response of a linear-system to any input will contain only modes generated by the input-signal's poles and by the system-transfer-function's poles. Using the simple z-transform pairs presented in Table D-1, we can convert Y(z) (D-8,D-9) to the time-domain response-sequence:

$$\{y(k)\} =$$

$$\{e_0\delta(k) + [e_1(q_1)^k + \cdots + e_s(q_s)^k] \qquad \text{(D-10)}$$

$$+ [d_1(p_1)^k + \cdots + d_n(p_n)^k]\}. \qquad \text{(D-11)}$$

In performing the inverse z-transform from Y(z) to {y(k)}, giving the time-domain sequences for the input-signal-mode (D-10) and the system-mode (D-11), the e_0 term is multiplied by the Kronecker-delta, $\delta(k)$. This function is defined as:

$$\delta(k) = \begin{cases} 1 \text{ for } k=0, \\ 0 \text{ for } k=\pm1,\pm2,\ldots \end{cases} \qquad \text{(D-12)}$$

with corresponding z-transform = 1 and R=0.

When an input signal is applied to a system, the response is partly made up of the natural-modes[3] of the input-signal (via poles of U(z)),

---

[3] Natural modes are the resonant frequencies of a system. Consider a string fixed at both ends. Say there is a continuous source of vibration at one end of the string. A vibration traveling down the string as a wave will be reflected from the fixed ends. At each reflection the wave is inverted. Say the string is vibrated at a frequency, f_o, such that wavelength, $l_o = 1/f_o = 2L$, twice the length of the string, L. Then, on reflection at the end of the return trip for a wavecrest traveling the length of the string from the source and back, that wavecrest is inverted so that it leaves exactly in phase with the wavecrest just generated by the vibration source. By superposition the two wavecrests add positively, and the energy of the departing wave is increased by the energy in the arriving wavecrest. The wave in the string is a standing wave, and the addition of energy with each vibration is the phenomena of resonance. If the string is continuously driven at f_o, eventually enough energy will be added in the generated wavecrests that the string will break. Resonance occurs not just

(continues on next page)

and partly caused by the input-signal exciting the natural modes of the system (via poles of H(z)). In designing a filter, one locates the poles of the transfer-function near the values for the natural-modes of the input-signal it is desired to transmit. The zeros of the transfer-function are located near the values of the natural-modes of the input-signal to be rejected.

For a system to have an output that looks much like the input, the system-mode must decay rapidly to zero after only a few time-steps. Which, on examination of {y(k)} (D-10,D-11), says that all of the system transfer-function poles, p_i, must be sufficiently smaller than one in magnitude. A sluggish system is one who's system mode decays to zero very slowly. So, a sluggish system is one that has at least one of the p_i close to 1 in magnitude.

For a well-designed filter we can ignore the system-mode (also called the transient-response) after a few time-steps, concerning ourselves only with the input-signal-mode. And say that:

$$\{y(k)\} = \{e_0\delta(k) + [e_1(q_1)^k + \cdots + e_s(q_s)^k]\}. \qquad (D\text{-}13)$$

(continued from previous page)
at f_o, but for all frequencies giving wavelengths that are integral divisors of l_o. That is for $l_i = nl_o$, n = 1,2,... Resonant frequencies are represented by the poles of a system transfer function, and hence by the poles of the input mode and system mode of a decomposed Y(z).

We ignored the first few time-steps to avoid the system transient-response (D-11), So, because of the Kronecker-delta (D-12), we can also ignore the e_0 term in {y(k)} (D-13) and say:

$$\{y(k)\} = \{ \sum_{i=1}^{\infty} e_i (q_i)^k \}. \qquad \text{(D-14)}$$

An output-sequence {y(k)} (D-14) is related to an input-sequence {u(k)} via the convolution of {u(k)} with some weighting sequence {h(k)}. But, {y(k)} is also a function of k as expressed above, dependent solely on the e_i and q_i. For each value, k, y(k) and u(k) are fixed. So, in the case being considered, the convolution of {u(k)} and {h(k)} may be considered as a simple <u>proportion</u> of each u(k) giving each y(k), where that proportion is based on the values of e_i and q_i^k.

Let us call the summation above e(k). Then for each value k, the output y(k) is proportional to e(k), and, in this case, the input u(k) is also proportional to e(k). A term is an <u>eigenfunction</u> of a linear-system when both input and output are proportional to it (Marple 1987). Here, the inputs and outputs are both proportional to e(k), which represents the system-input-mode. So, <u>the system-input-mode is an eigenfunction of the linear-system represented by a convolution-summation</u>. An <u>eigenvalue</u> may be considered the 'constant-of-proportionality' relating the input and output to an eigenfunction. So, <u>the concept of a system transfer-function corresponds to the eigenvalues of the process</u> (Hamming 1983).

Eigenvectors and Eigenvalues

Eigenfunction (also characteristic, proper, or natural function) is the general term for a class of orthogonal-functions. Here we are concerned with the special case of eigenfunctions: eigenvectors, and their associated eigenvalues. Eigenvectors and eigenvalues, and their relationship to linear-systems, are covered in detail in Appendix E, within the context of factor-analysis. Here we briefly look at their properties as they relate to system transfer-functions and convolution.

If we multiply a square-matrix, $\mathbf{A}$, by a vector, $\bar{x}$, where $\mathbf{A}$ is nxn, and $\bar{x}$ is nx1, the product is another vector of dimension nx1:

$$\mathbf{A}\bar{x} = \bar{y}. \qquad \text{(D-15)}$$

In general, the output-vector, $\bar{y}$, will point in a different direction in n-dimensional space that the input-vector, $\bar{x}$. However, for most square matrices $\mathbf{A}$, of dimension n, that we expect to encounter in expressing convolution-summation filters as linear-systems in matrix form, there will be n different vectors, $\bar{x}$, such that the corresponding $\bar{y}$ will have the <u>same direction</u> as $\bar{x}$, though not necessarily the <u>same magnitude</u>. For these vectors:

$$\mathbf{A}\bar{x} = \bar{y} = \lambda\bar{x} \qquad \text{(D-16)}$$

or,

$$(\mathbf{A} - \lambda I)\overline{X} = 0 \qquad \text{(D-17)}$$

where I is the identity matrix, and 0 is the null vector. For this equation to have a non-zero solution it is necessary that the determinant of the system of equations:

$$|\mathbf{A} - \lambda I| = 0. \qquad \text{(D-18)}$$

The determinant is a polynomial in λ of degree n, and will in general have n distinct roots, λ_i, i=1,2,...,n, which may be either real or complex (the existence of multiple-roots is determined by the form of $\mathbf{A}$). So, there are in general n distinct λ_i, with corresponding vector solutions, $\overline{X}_i$. The λ_i are <u>eigenvalues</u>, and the $\overline{X}_i$ are <u>eigenvectors</u>.

Eigenvectors are <u>linearly-independent</u>, so they may be used as a <u>basis</u> to express any arbitrary vector, $\overline{X}$, in n-dimensional space. Thus we can say:

$$\overline{X} = \sum_{i=1}^{n} a_i \overline{X}_i . \qquad \text{(D-19)}$$

Or, applying $\mathbf{A}$ to both sides:

$$A\bar{X} = \sum_{i=1}^{n} a_i A\bar{X}_i = \sum_{i=1}^{n} a_i \lambda_i \bar{X}_i . \qquad \text{(D-20)}$$

where the $\bar{X}_i$ are <u>not</u> vector components of $\bar{X}$, but the eigenvectors of the previously expressed determinant (D-18). So, each eigenvector is multiplied only by its corresponding eigenvalue, each product being an eigenfunction. And, each eigenfunction is independent and unaffected by the presence of any other eigenfunction.

A convolution-summation filtering process may be represented by a discrete linear-system, therefore we can find some description of that process in terms of the eigenvectors and eigenvalues of that system. We have already shown that the system-input-mode from the decomposed transfer-function for a convolution-summation process is an eigenfunction of that process. Comparing the form of the decomposed system-input-response (D-14) to the form of the eigenvector expression above (D-20), it appears that eigenfunctions of the convolution-summation process must relate to the poles, q_i, of the z-transform of the input-signal, U(z). But, decomposition using the z-transform to find an eigenfunction expression for the input-signal-mode left us with powers-of-k of the q_i. We must find a means to express these relationships that will be useful for determining filter response-characteristics in terms of the q_i themselves.

Eigenfunctions of Translation

Classic methods for describing approximation use the sampling-polynomial:

$$\pi(x) = (x - x_1)(x - x_2)\cdots(x - x_n) \qquad \text{(D-21)}$$

Note that it vanishes at all the sample-points, x_i. If some power of x, say x^m, is divided by $\pi(x)$, resulting in a quotient, Q(x), and remainder R(x), then:

$$x^m = \pi(x)Q(x) + R(x) \qquad \text{(D-22)}$$

where R(x) is of degree less than n, the degree of $\pi(x)$. Note that at the sample points, $x^m = R(x)$. So, at the sample-points, the original single-power of x is aliased into a polynomial, R(x), which is a linear combination of 1, x, $x^2,\ldots,x^{(n-1)}$, and not into a single power. It is considerations such as these which lead to results from the z-transform having powers-of-k of the poles, q_i, rather than simple expressions of the q_i.

In describing the process of aliasing due to discrete, equal-interval sampling we noted any frequency higher than the Nyquist frequency is translated into some apparent lower frequency. And, that these translated-frequencies relate to the apparent lower-frequency by

differences that are integral-multiples of $2\pi/T$, where T is the period between samples. Aliasing takes any one of these higher frequencies, and, in the sense of having the same values at all the sample-points, transforms it into a single low-frequency function. The process of equally-spaced sampling followed by the reconstruction of a function of minimal-frequency results in a single sinusoid. That is, all the sampled-sequences of the high-frequency function are, in some sense, 'proportional' to the common sine and cosine functions. From this description of aliasing, it is reasonable to say that the common sine and cosine functions are the eigenfunctions of equal-interval sampling (Hamming 1983). If common sine and cosine functions are eigenfunctions of equal-interval sampling, then there must be some way to relate them to the eigenfunction-expression we developed for the system-input-mode via the z-transform (D-20). In fact, we may do this through the application of simple trigonometric identities, the Euler-identities, and the expression of complex-variables in exponential form. Using the addition formulas of trigonometry:

$$\sin(X+Y) = \sin X\cos Y + \cos X\sin Y, \qquad \text{(D-23)}$$

$$\cos(X+Y) = \cos X\cos Y - \sin X\sin Y, \qquad \text{(D-24)}$$

we can show that for a translation $X = X' + K$, $A\sin X + B\cos X$, becomes $A'\sin X' + B'\cos X'$, where:

$$A' = A\cos K - B\sin K, \qquad \text{(D-25)}$$

$$B' = A\sin K + B\cos K. \qquad \text{(D-26)}$$

Since, with a translation, the weighted-sum of cosX and sinX comes out as 'proportional' to that sum, the pair cosX and sinX constitute the eigenfunctions under translation. That is, if we translate the pair by an amount, K, we get back sin and cos after the operation is complete.

The Euler-identities are:

$$\cos X + i\sin X = e^{iX}, \qquad \text{(D-27)}$$

$$\cos X - i\sin X = e^{iX}. \qquad \text{(D-28)}$$

The Euler-identities (D-27,D-28) lead to:

$$\cos X = \tfrac{1}{2}(e^{iX} + e^{-iX}), \qquad \text{(D-29)}$$

$$\sin X = \tfrac{1}{2}(e^{iX} + e^{-iX}). \qquad \text{(D-30)}$$

Applying these (D-29,D-30) in the trigonometry addition formulas (D-25, D-26) gives:

$$e^{iX}e^{iY} = e^{i(X + Y)}. \qquad \text{(D-31)}$$

This (D-31) expresses a translation of the complex-exponential derived from the translation of sine and cosine. The result is a complex-exponential, so the complex-exponentials are the eigenfunctions of translation. The complex-exponential expression was derived from sine and cosine, implyng sine and cosine are <u>invariant</u> under translation. Being invariant means that any linear-combination of sine and cosine of a given frequency can still be written as a linear-combination of sine and cosine when an arbitrary translation of the coordinate-axis is made. <u>In our case a translation is a time-step, not a distance-step</u>.

Sine and cosine are <u>unique</u> in having the property of invariance under translation. So, we can express the eigenfunction-property in complex-exponential form:

$$u(t+K) = e^{iw(t+K)} = e^{iwK}e^{iwt} = \lambda(w)u(t), \qquad (D\text{-}32)$$

where $\lambda(w) = e^{iwK}$ is the eigenvalue, independent of t, and e^{iwt} is the eigenfunction of translation.

Magnitude and Phase

We now have a set of eigenfunctions e^{iwt}, with eigenvalues e^{iwK}, through which we can directly relate system-inputs and system-outputs. With these we can tie together previous concepts and develop a way to express both the <u>magnitude-response</u> and <u>phase-response</u> for a filter.

By magnitude-response we mean the amplification or gain-factor associated with each frequency input to the system. It is from a frequency vs. gain plot that we determine filter roll off characteristics. A plot of frequency vs. gain for a filter is commonly referred to as a gain-factor plot. Sometimes it will be seen referred as a filter-spectrum plot. This spectrum is related to the spectrum produced by using Fourier-analysis. Truncating the infinite-sum z-transform (D-2) to N values, 0 to N-1, (relating it to a length-N input-sequence), and using $z = e^{i2\pi k/N}$ (the eigenfunction), yields the Discrete-Fourier-Transform (Burrus and Parks 1985).

Phase-response refers to amount of phase-delay produced by a filter for each frequency input to the system. This, too, is most often expressed as a plot with frequency as the independent-variable. Phase-response plots are called phase-angle-factor plots. Sometimes they will be seen referred to as Bode plots.

Gain-factor and phase-angle-factor plots are produced by determining the system-response for various input-frequencies; generally from zero to the Nyquist-frequency. This means that we must apply pure sinusoidal-inputs with known frequency to a system and measure the output-responses. Doing this on a physical-system or even using its mathematical analog could be a very tedious process. However, with a system-transfer-function and associated eigenfunctions available it becomes a simple operation.

For any linear system, the steady-state response to a sinusoidal input is a sinusoidal output. That is, once the transient-response due to the poles of H(z) has died away, the system-output will be a sinusoid of the same frequency as the input, but perhaps at a different amplitude and different phase relative to the input. From previous discussion we know that if we apply one of its eigenfunctions to the system transfer-function, we will get that eigenfunction back as an output. So, of course, if we use the eigenfunction of translation, e^{iwt}, as input we will get back that eigenfunction as the output, perhaps different in amplitude by a multiplicative constant (the eigenvalue e^{iwK}). But, we know that the eigenfunction, e^{iwt}, is another way of expressing the pair sine and cosine. So, applying e^{iwt} to the system transfer-function means we are applying a sinusoid of a single frequency, w, to that function. Hence we get back a sinusoidal output that will reflect the input in terms of the system gain-factor and phase-angle responses at w. Further, a complex-number, z, may be expressed in the form e^{iwt}. So we simply put e^{iwT} in place of z in the expression for the z-transform of a system transfer-function, H(z), and we arrive at an expression in terms of w that may be used to evaluate the characteristics of our filter.

$H(e^{iwT})$ is a complex-number. Therefore it is a vector and may be expressed in terms of magnitude and angle. Its magnitude is $UH(e^{iwT})U$ which is the gain-factor at frequency w. The angle of $H(e^{iwT})$, expressed as $\angle H(e^{iwT})$, is determined as an arctangent from $H(e^{iwT})$.

Bandwidth

Bandwidth for a low-pass filter is a measure of the width of the band of frequencies for which at least one-half the power of an input sinusoid is transmitted to the output. The power contained in a sinusoidal-sequence {AsinkwT} is given by $A^2/2$. So, the bandwidth for a normalized low-pass filter corresponds to values of w such that:

$$|H(e^{iwT})| \geq 1/\sqrt{2} \qquad \text{(D-33)}$$

The frequency for the break-point in bandwidth is often referred to as the 3db-point. A reduction in power of 3-decibels translates to reducing energy by 1/2.

Determination of Filter Characteristics

Since e^{iwT} is a periodic-function of w with period $2\pi/T$, $H(e^{iwT})$ is a periodic-function of w with period $2\pi/T$. So, the system gain-factor and phase-angle-factor are both periodic-functions of w with period $2\pi/T$. That is, they produce the same value every integer multiple of $2\pi/T$ for any value of w.

Our sampling-frequency is $2\pi/T$. By the sampling-theorem the highest significant-frequency in our input should the Nyquist-frequency, which is one-half the sample-frequency, or π/T. Frequencies beyond this

range will be aliased. But, with proper selection of sampling period, T, aliasing is minimized. So, we look at the behavior of $H(e^{iwT})$ over the interval $0 \leq w \leq \pi/T$ to determine our filter-characteristics.

Transversal Filters

The data-preparation filter used for this thesis is a member of a class of nonrecursive filters know as transversal-filters. This refers to the fact that the weight-sequence transverses only a part of the input-sequence. Here our normalized, least-squares-fit, quadratic filter-weights are given by:

$$\{b_i\} = \frac{1}{231} \{-21,\ 14,\ 39,\ 54,\ 59,\ 54,\ 39,\ 14,\ -21\}. \qquad \text{(D-34)}$$

These values are derived by performing a least-squares-fit over nine points though the quadratic equation: $y = a + bx + cx^2$. They were chosen over the weights for a simple running-average (boxcar) filter, (i.e., all weights equal to one with a leading multiplier of 1/9), to help reduce some of the extreme initial roll-off that is given by the simpler boxcar-filter.

A transfer-function may be evaluated anywhere, so we choose as our indices for the b_i, $i = -4,-3,-2,-1,0,1,2,3,4$; to later take advantage of the Euler-identities. This gives us a filter of the form:

$$y(k) = u(k-4)b_{-4} + \cdots + u(k)b_0 + \cdots + u(k+4)b_4. \qquad \text{(D-35)}$$

We know from the convolution-summation property of the z-transform that in this case the system-transfer-function and the z-transform of the filter-weight-sequence are transform-pairs. So, applying the basic summation definition of the z-transform to our problem, we get:

$$H(z) = \sum_{k=-4}^{4} b_k z^{-k}. \qquad \text{(D-36)}$$

We know the b_i are symmetric around b_0, so we can say:

$$H(z) = b_0 + \sum_{k=1}^{4} b_k (z^{-k} + z^{k}). \qquad \text{(D-37)}$$

Substituting eiwT for z:

$$H(e^{iwT}) = b_0 + \sum_{k=1}^{4} b_k (e^{-iwkT} + e^{iwkT}). \qquad \text{(D-38)}$$

Now for the magic:

$$\begin{aligned} e^{iX} &= \cos X + i\sin X \\ +)\ e^{-iX} &= \cos X - i\sin X \\ \hline e^{iX} + e^{-iX} &= 2\cos X \end{aligned} \qquad \text{(D-39)}$$

So,

$$H(e^{iwT}) = b_0 + 2\sum_{k=1}^{4} b_k \cos wkT. \qquad \text{(D-40)}$$

Applying the above expression for the system-transfer function (D-40) with the weighting-sequence (D-34) for our low-pass filter (D-35):

$$H(e^{iwT}) = \frac{1}{231} \{59+108\cos wT+78\cos 2wT+28\cos 3wT-42\cos 4wT\}. \qquad (D-41)$$

The gain-factor, $|H(e^{iwT})|$, is obtained by substituting the sample-period value for T then evaluating the expression (D-41) for w, $0 \leq w \leq \pi/T$. The phase-angle-factor, $\angle H(e^{iwT})$, is dependent on the magnitude of the imaginary-part of the expression for $H(e^{iwT})$. That is, $\angle H(e^{iwT})$ is dependent on any isinwkT terms. Our filter is symmetric and isinwkT terms drop out via the Euler-identities. So, here, $\angle H(e^{iwT}) = 0$ for all w, $0 \leq w \leq \pi/T$.

Using T = 0.25 hour, we get a range for w, $0 \leq w \leq 12.566$ rad/hr. And, using T as 1/4 in our gain-factor expression, we obtain:

$$H(e^{iw/4}) = \frac{1}{231} \{59+108\cos w/4+78\cos 2w+28\cos 3w/4-42\cos w\} \qquad (D-42)$$

as our equation to be evaluated between the given limits on w.

Summary of Appendix-A Low-Pass Filter Properties

The gain-factor plot (Figure D-8) from the expression developed in the previous section (D-42) does not appear to be very much like that for the ideal low-pass filter (Figure D-6). Note that $H(e^{i\omega T})$ was plotted rather than $|H(e^{i\omega T})|$. This was done to simplify finding frequencies where $|H(e^{i\omega T})| = 0$. And explains the presence of negative peaks in the plot. Still, even accounting for the negative values, it is clear that our preconditioning low-pass filter is not ideal. This is not necessarily a problem. What is important is not how closely the gain-factor plot of a filter matches the ideal-case, but how well suited the gain-factor of a filter is to our intentions.

The simple boxcar-filter is commonly used as a data preconditioning filter. Its gain-factor plot for a nine-point average (Figure D-9) appears even 'less ideal' than the plot for our quadratic-filter. In our case either the boxcar-filter or the chosen quadratic-filter would serve our purposes in expanding the time-step and reducing the possibility of aliasing in later analyses. Both filters pass frequency components with periods of 12 hours or longer at much more than the one-half-power level, and frequency components higher than our theoretical one-half cycle/hour Nyquist limit for the one-hour interval data are attenuated. The main difference between the two filters is in their degree of attenuation of frequency-components between about 0.12 cycles/hour and 0.59 cycles/hour (Table D-2). The boxcar-filter attenuates all frequency

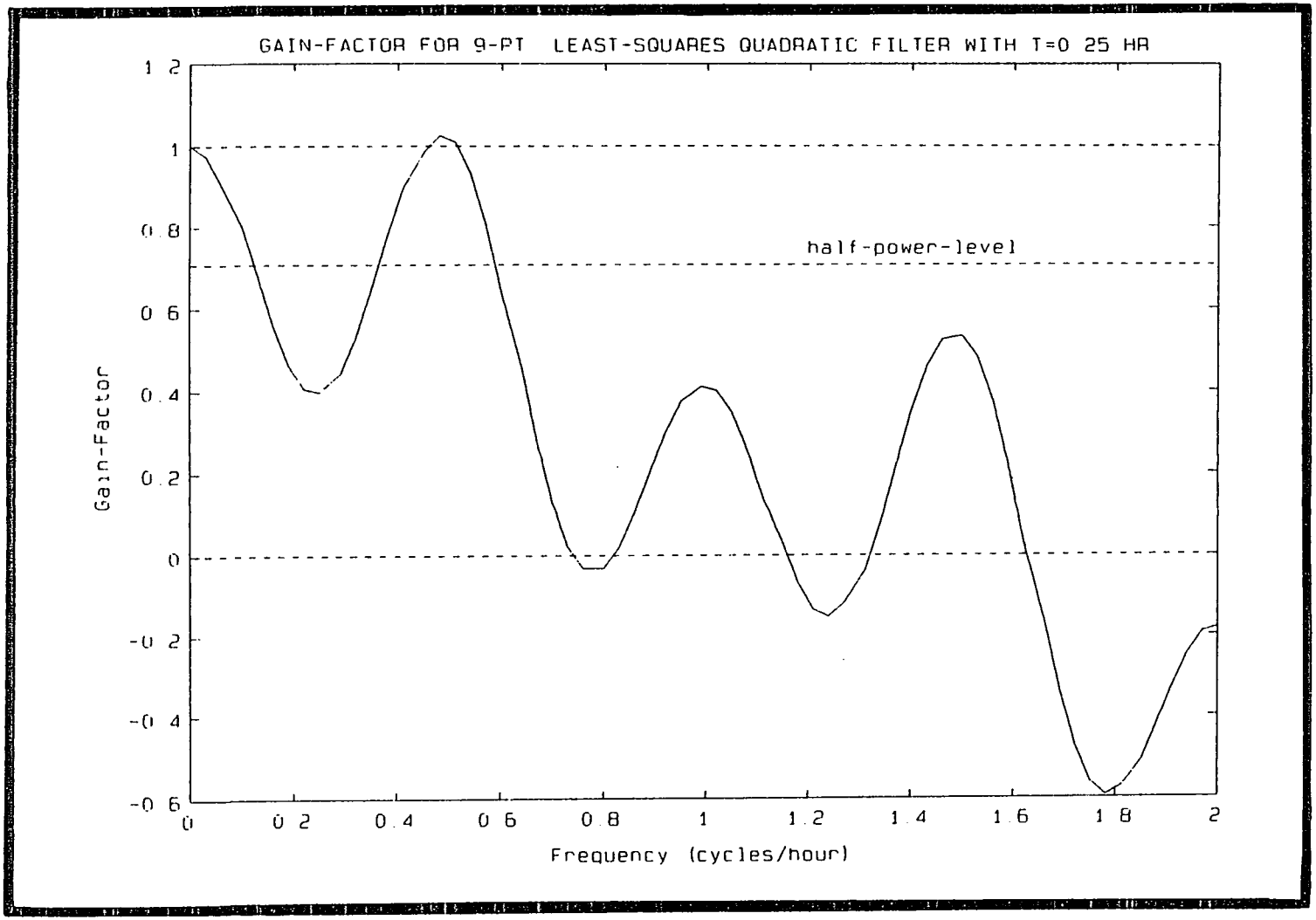


FIGURE D-8: Gain-factor plot for a 9-point, least-squares fit, quadratic-weight, low-pass filter with period T = 0.25 h.

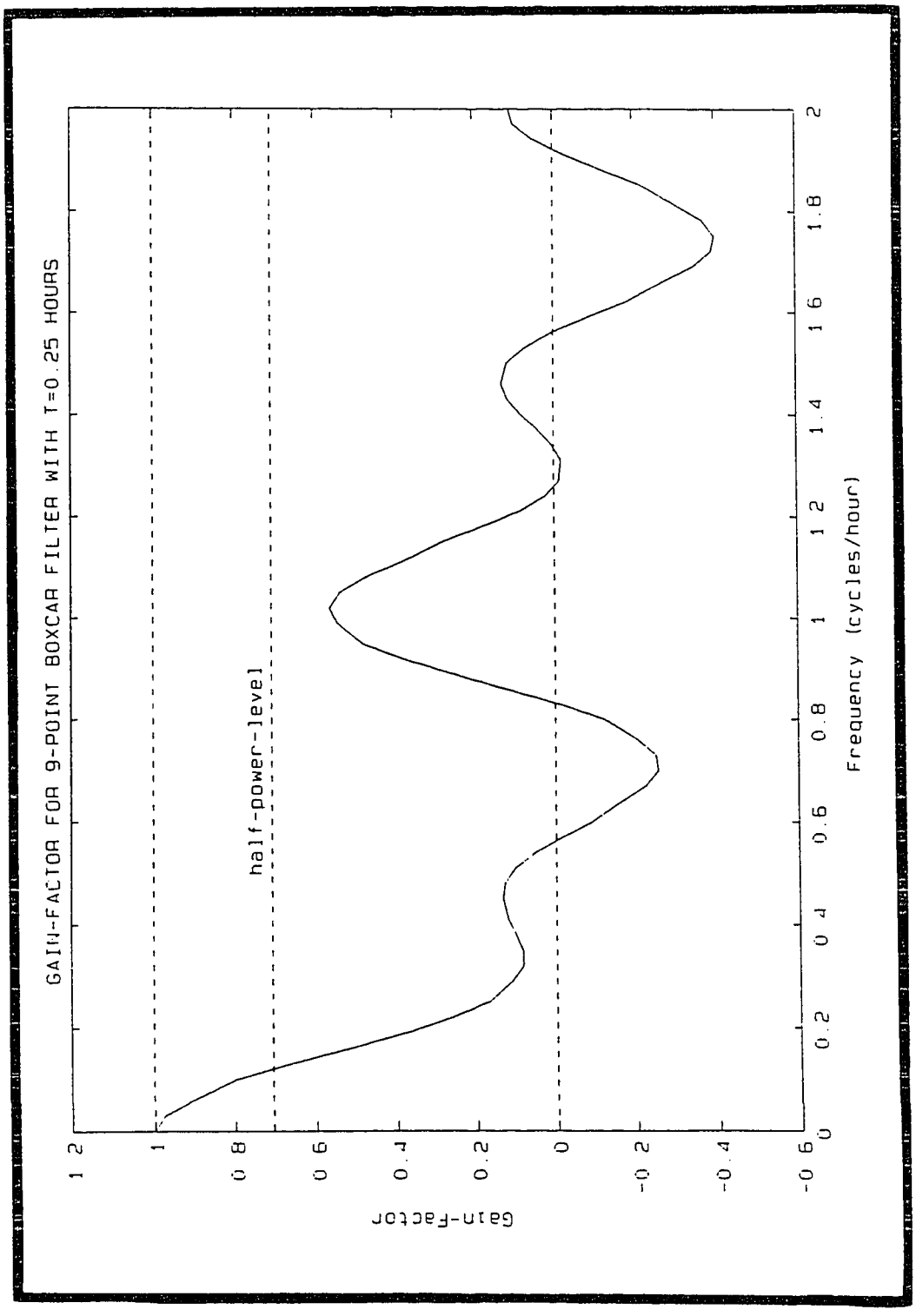


FIGURE D-9: Gain-factor plot for a 9-point, boxcar low-pass filter with period T = 0.25 h.

components with a period greater than about 5.6 hours to below the half-power level. Our preconditioning-filter attenuates frequency components with periods from about 8.4 hours to 2.7 hours to below the half-power-level, with a minimum at about T = 4.1 hours. It then passes those frequency-components with periods from about 2.7 hours to 1.7 hours at greater than the half-power level, with a maximum at about T = 2.2 hours. After frequencies with corresponding periods greater than about 1.7 hours are reached, all components are attenuated to less than the half-power level by the preconditioning-filter.

TABLE D-2 -- Gain-factor for 9-pt. Boxcar and 9-pt. Quadratic low-pass filters with sampling period T = 0.25 h.

f (c/h)	T (h)	$\lvert H(e^{i\omega/4})\rvert$ Boxcar	Quadratic
0.00	∞	1.00	1.00
0.12	8.33	0.69	0.70
0.24	4.17	0.21	0.40
0.36	2.75	0.09	0.69
0.45	2.22	0.13	0.99
0.50	2.00	0.11	1.02
0.51	1.99	0.10	1.01
0.57	1.75	0.01	0.82
0.59	1.69	0.06	0.72
0.74	1.35	0.24	0.00
0.83	1.20	0.00	0.02
1.00	1.00	0.56	0.41
1.25	0.80	0.01	0.15
1.33	0.75	0.00	0.05
1.56	0.64	0.00	0.37
1.75	0.57	0.40	0.56
1.95	0.51	0.06	0.23
2.00	0.50	0.11	0.18

By using the quadratic filter we have introduced the possibility for aliasing of the final one-hour interval data by higher-frequency components that alias the 15-minute interval data between 0.5 and about 0.6 cycles/hour. The gain-factor drops to below the half-power level at about 0.6 cycles/hour. However, some frequency components between 0.5 and 0.6 cycles/hour are not strongly attenuated by the quadratic filter. This is acceptable in the 15-minute data, but the Nyquist-limit for one-hour interval data is 0.5 cycles/hour. Thus, the components with frequencies from 0.5 to 0.6 cycles/hour passed while filtering the 15-minute interval data could alias into the one-hour interval data. The corresponding period-range is 1.67 to 2 hours. So, for example, a seiche[4] with period in the range 1.67 to 2 hours may alias longer-period components in the one-hour interval data. Using the sampling-period T = 1 hour, aliasing folds-back energy from components with periods from 1.67 to 2 hours into components with periods from 2.5 hours to 2 hours, respectively. For our purposes, components with periods shorter than 2.5 hours are 'high frequency.' If we accept, say, 3 hours as our minimum period and ignore any shorter-period information derived from the analyses, then any potential aliasing effects will be by-passed.

---

[4] A seiche is a standing-wave (see Footnote-3) that develops in a basin due to energy imparted by the tides, or atmospheric or seismic disturbances. They are resonance-phenomena and their occurence, persistence, frequency and the energy they contain are very dependent on basin geometry.

We can try and determine if, in fact, we should be concerned about potential aliasing into the 2.5 to 2 hour period-range. If a seiche is close to but slightly less than 2 hours period, then it is very close to the minimum-period resolution of the one-hour data. Therefore, through frequency-folding, it can only alias components with period very close to but slightly longer than 2.00 hours period. If we are only aliasing frequencies very close to our aliasing frequency, then we are doing minimal damage. This energy 'bleed over' will not greatly affect results based on total energy derived from components near that region. If the seiche has period closer to 1.67 hours, then it may alias data components with periods around 2.5 hours. A single seiche of period near 1.67 hours will be passed by the quadratic filter at very close to its half-power level. So, our hypothetical 1.67 hour period sieche would have to contain a great deal of energy before it transferred significant aliasing-energy into the 2.5 hour period-range. However, aliasing is a cumulative effect. That is, energy from multiple frequency-bands is added to a particular region affected by aliasing.

We must be concerned with all frequency bands that may contribute energy via aliasing into the 0.5 to 0.6 cycles/hour band. This corresponds to the range 3.14 to 3.77 radians/hour. Any aliasing frequencies will be a sum or difference of some integer multiple of π/T away from frequencies in this range (Table D-3). Using T = 1 hour, we see that the first higher frequency-range that could cause aliasing is 5.65 to 6.28 radians/hour, followed immediately by 6.28 to 6.91

radians/hour, giving a total range of 5.65 to 6.91 radians/hour. This first radial-frequency-range corresponds to the rotational-frequency limits of 0.9 to 1.1 cycles-per-hour. Frequency components in this range could certainly be found in tidal-data. But, the energy contained in this range is generally much lower than the energy found in the predominant tidal-frequency components. Also, the preconditioning filter strongly attenuates frequency components in this range. So, frequencies in this first range should not cause any appreciable aliasing of the one-hour interval data . The next higher rotational-frequency range that could cause aliasing in the decimated data is 1.4 to 1.6 cycles/hour. Here again these frequency-components will be of low-energy in tidal-data except for unusual circumstances, and the preconditioning filter attenuates these components to below the half-power level. The next range of frequencies that may cause aliasing is from 1.9 to 2.1 cycles/hour. Frequencies from 1.9 to 2 cycles/hour are strongly attenuated by the preconditioning filter. The frequencies from 2 to 2.1 cycles/hour are off-scale on our standard gain-factor plot. But, the gain-factor is periodic. If we fold-out a mirror-image of the gain-factor plot we can see the filter response at higher frequencies (Figure D-10). Frequencies from 2 to 2.1 cycles/hour are also attenuated to below the half-power-level.

Note the form of the 'folded out' gain-factor plot. There are three intervals where a component with frequency greater than 2 cycles/hour will not be attenuated to below the half-power-level: 3.41 to 3.64, 3.88 to 4.12, and 4.37 to 4.59 cycles/hour. The nearest ranges to these that may

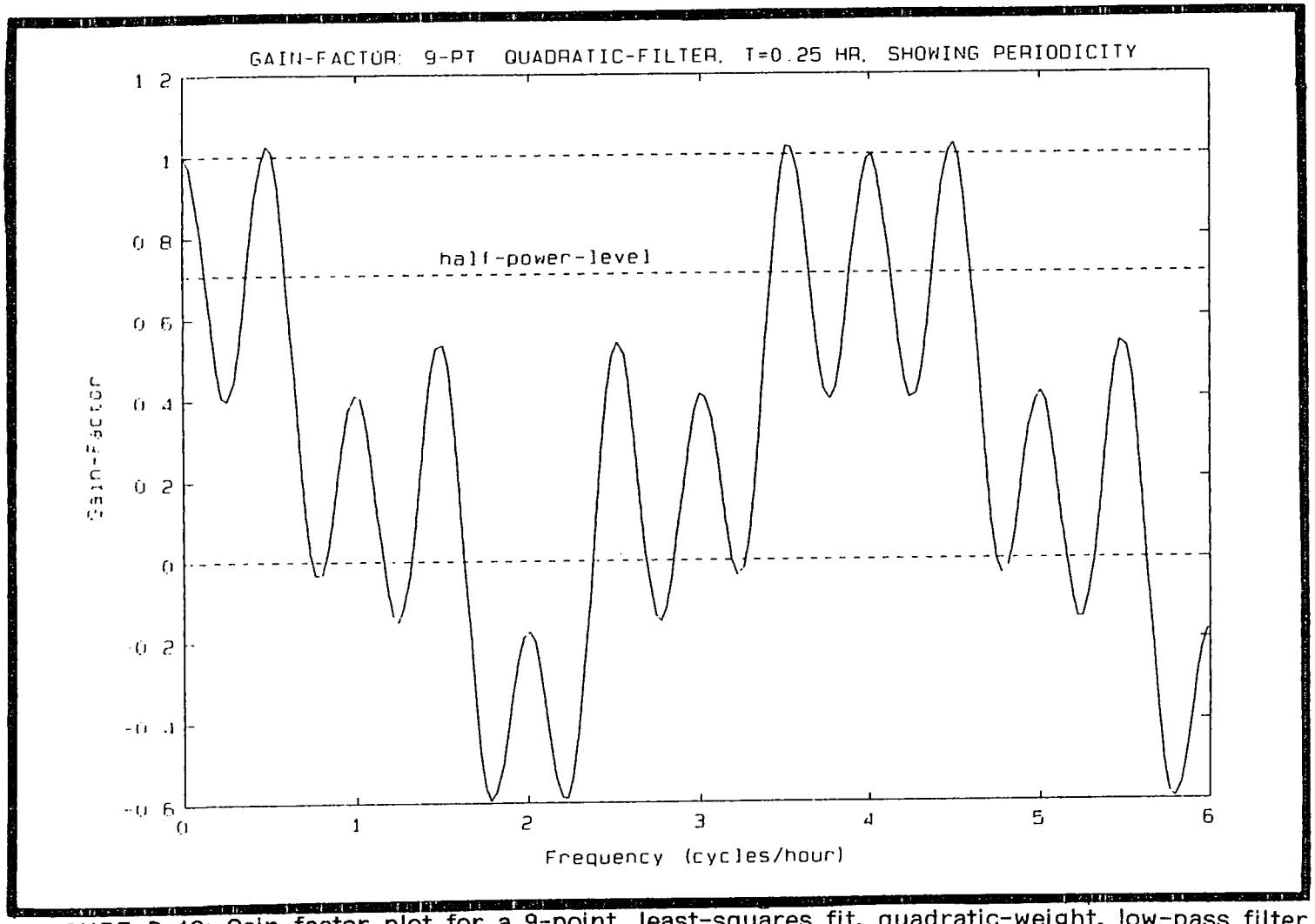


FIGURE D-10: Gain-factor plot for a 9-point, least-squares fit, quadratic-weight, low-pass filter with period T = 0.25 hours. Here the plot is made out to 3 time the Nyquist frequency to show periodicity of the gain factor function.

cause aliasing in the 1-hour interval data are: 3.4 to 3.6, 3.9 to 4.1, and 4.4 to 4.6 cycles/hour; for all intents and purposes, the same. The aliasing frequency-ranges relate to period-ranges of: 16.67 to 17.64, 14.63 to 15.38, and 13.04 to 13.63 minutes, respectively. Seiches may exist that have periods within these ranges. However, the probability of a seiche of high energy having a period very close to the center of one of these narrow power-transfer 'windows' is small. So, we do not have to be overly concerned about their aliasing of the 0.5 to 0.6 cycle/hour range. Above the frequency 4.60 cycles/hour all components are passed at well below the half-power-level until the next period of the gain-factor begins, at a frequency 4 cycles/hour higher. The first repeat of the three power-transfer windows will start at a frequency of 7.64 cycles/hour, with corresponding period-ranges: 7.85 to 8.1, 7.39 to 7.61, and 6.98 to 7.17 minutes. We may again argue that the probability of any one seiche existing near the center of one of these windows is small. Also, the water action in a basin with a seiche near these periods and energetic enough to add significant aliasing-energy to the normal tidal-energy would be very noticeable. For the next fold-out of the gain factor the first window has a corresponding center-period of 5.21 minutes. Any action of the water in an enclosed body such as the northern reaches of San Francisco Bay with a period near 5-minutes and energetic enough to significantly alias tidal-components having period near 2.5 hours could cause significant coastline erosion. This does not occur. So, we can with some confidence say that the potential aliasing-energy from the frequency-band 0.5 to 0.6 cycles/hour is not likely to affect our

analyses for and using frequencies from 0.5 cycles/hour or lower. In using the 9-point quadratic-filter to precondition data, we have changed the time-step from 15-minutes to one-hour in a manner consistent with the requirement that the majority of the energy in the resulting data set will be found in components below the Nyquist frequency of 0.5 cycles/hour. As such, aliasing of the lower-frequency components is minimized. We have also allowed for some the information near the Nyquist frequency to be available. This near-Nyquist information would have been lost using a simple-boxcar filter.

TABLE D-3 -- First 16 ranges for frequency-components that may alias one-hour interval data from 0.5 to 0.6 cycles/hour when these data are derived by using a 9-point, quadratic-weight, noniterative, transversal filter stepped every four points through 15-minute interval data. These ranges were generated using the folding-sequence: w, $\pi/T-1$, $w+\pi/T$, $2\pi/T-w$, $w+2\pi/T$, ... on the endpoints of the range $3.14 \leq w \leq 3.77$ radians/hour, (which corresponds to $0.5 \leq f \leq 0.6$ cycles/hour), with $T = 1$ hour. The ranges were sorted in ascending order.

radian-frequency	rotational-frequency
0.00 to -.63 rad/hr	0.00 to -.10 cycles/hr
2.51 to 3.14	0.40 to 0.50
3.14 to 3.77	0.50 to 0.60
5.65 to 6.28	0.90 to 1.00
6.28 to 6.91	1.00 to 1.10
8.80 to 9.42	1.40 to 1.50
9.42 to 10.05	1.50 to 1.60
11.94 to 12.57	1.90 to 2.00
12.57 to 13.19	2.00 to 2.10
15.08 to 15.71	2.40 to 2.50
15.71 to 16.34	2.50 to 2.60
18.22 to 18.85	2.90 to 3.00
18.85 to 19.48	3.00 to 3.10
21.36 to 21.99	3.40 to 3.50
21.99 to 22.62	3.50 to 3.60
24.50 to 25.13	3.90 to 4.00
25.13 to 25.76	4.00 to 4.10
27.64 to 28.28	4.40 to 4.50
28.28 to 28.90	4.50 to 4.60
30.79 to 31.42	4.90 to 5.00
31.42 to 32.04	5.00 to 5.10

THE GODIN FILTER

Introduction

We refer to the 'Godin-filter,' (named for Gabriel Godin, after work presented in The Analysis of Tides (Godin 1972)), as a time-domain filter because, in application, all operations take place as convolutions of time-domain data. The filtering technique is rooted in spectral analysis, being developed from examination of affects of mathematical operations, (addition and subtraction), on the spectrum of a data-sequence. Examination of these spectral-effects is made using Fourier-series (Appendix B). This description of time-domain low-pass tidal-filtering follows that presented in Godin (1972). The notation has been altered to agree with notation presented elsewhere in this thesis.

The Line-Spectrum

The Fourier-transform pair:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(w)e^{iwt}dw, \qquad \text{(D-43)}$$

$$X(w) = \int_{-\infty}^{\infty} x(t)e^{-iwt}dt, \qquad \text{(D-44)}$$

where: t is time, w is radian-frequency,
x(t) is the inverse-fourier-transform of X(w),
and, X(w) is the Fourier-transform of x(t),

was developed in Appendix B (B-52,B-53). If we consider a pure harmonic in exponential-form:

$$x(t) = e^{iw_0t}, \qquad \text{(D-45)}$$

where: $w_0 = 2\pi f_0$,
f_0 = the rotational-frequency of x(t),

and attempt to find its Fourier-Transform directly from the definition (D-44), the result is not very informative. However, if the limits of integration are allowed to approach infinity via:

$$X(w) = \lim_{T\to\infty} X_T(w), \qquad \text{(D-46)}$$

with,

$$X_T(w) = \int_{-T}^{T} e^{iw_0t}e^{-iwt}dt, \qquad \text{(D-47)}$$

then the integration (D-47) yields:

$$X_T(w) = \frac{\sin[2\pi T(w - w_0)]}{\pi(w - w_0)} \equiv 2T\mathrm{Dif}_T(w - w_0). \qquad \text{(D-48)}$$

The quantity $\mathrm{Dif}_T^2(w - w_0)$ is found in optics[5] with regard to

---

[5] with proper parameterization of ß in $\sin^2ß/ß^2$ this quantity provides the modulation-function that describes the intensity of the pattern seen from a single-slit diffraction-grating (Jenkins and White 1976). It is applied to the square of the amplitude of the incoming beam and hence relates to energy arriving at the viewing-screen behind the slit.

diffraction-gratings, and here, following Godin (1972), we refer to the quantity $Dif(w - w_o)$ as the diffraction-function. It has maximum of 1 at $w = w_o$, and decreasing-amplitude oscillations as $(w - w_o) \to \pm\infty$ (Figure D-11). As T gets large, the peak at $w = w_o$ and the function tends to become small everywhere except at $w = w_o$.

Using the diffraction-function (D-48), the Fourier-transform (D-47) may be expressed as:

$$X(w) = \lim_{T\to\infty} X_T(w) = \lim_{T\to\infty} 2TDif(w - w_o) \equiv \delta(w - w_o), \quad \text{(D-49)}$$

$\delta(w - w_o)$ is the delta-function:

$$\delta(w - w_o) = \begin{cases} \infty, & w = w_o, \\ 0, & w \neq w_o. \end{cases} \quad \text{(D-50)}$$

It can be shown that:

$$\int_{-\infty}^{\infty} \delta(w - w_o)dw = 1, \quad \text{(D-51)}$$

so,

$$\int_{-\infty}^{\infty} g(w)\,\delta(w - w_o)dw = g(w_o), \quad \text{(D-52)}$$

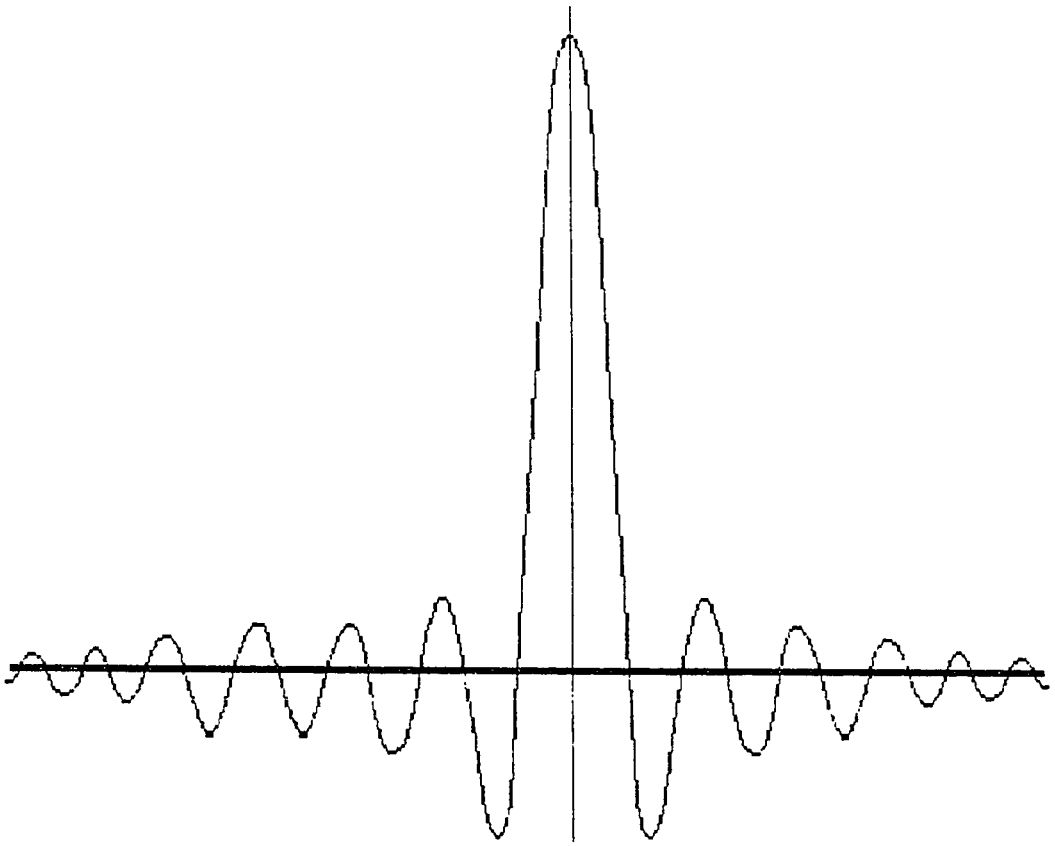


FIGURE D-11: The diffraction function.

for any function, g(w). In practical-work, the delta-function always takes the form of the truncated Fourier-Transform (D-48) because T is finite for real-data.

Using the delta-function (D-50), and extraction of sine and cosine from the Euler-identities (D-29,D-30) the Fourier-transform of sin$w_o t$ becomes:

$$\int_{-\infty}^{\infty} \sin(w_o t) e^{-i w_o t} dw = \frac{1}{2i}[\delta(w - w_o) - \delta(w + w_o)], \qquad \text{(D-53)}$$

and, the Fourier-transform of cos$w_o t$ becomes:

$$\int_{-\infty}^{\infty} \cos(w_o t) e^{-i w_o t} dw = \frac{1}{2}[\delta(w - w_o) - \delta(w + w_o)]. \qquad \text{(D-54)}$$

The Fourier transform of a constant-valued function, x(t) = a is:

$$X(w) = a\delta(w). \qquad \text{(D-55)}$$

Using the above information (D-53,D-54,D-55), the Fourier-transform of a function which is the sum of pure-harmonics:

$$x(t) = \sum_{k=-N}^{N} a_k e^{iwkt}, \qquad \text{(D-56)}$$

may be expressed as a sum of delta-functions:

$$X(w) = \sum_{k=-N}^{N} a_k \delta(w - w_k). \qquad (D\text{-}57)$$

This formulation is known as the line-spectrum. Graphically it is represented by a series of vertical-lines of height a_k plotted on a horizontal-axis relative to corresponding-frequencies w_k.

Digitization of Band-Limited Functions

A band-limited function is one whose frequency-content is limited to components within a specific range of frequencies. Virtually all real-signals are band-limited. Band-limiting may be intentional, a function of instrument-response, or both. For example, the pressure-sensor used to generate the tidal-excursion record for this thesis has some predefined (though high) rate above which it is not capable of responding to changes in pressure. Signals coming off the pressure sensor are put through a low-pass-filter implement as a 28-second period 15-reading box-car-average, which removes 'wave-chop' from records. So, tidal-excursion records from the monitoring-station accessed for this thesis lose high-frequency information through inevitable and intentional band-limiting. Thus, these signals will contain no frequency exceeding some maximum-frequency w_m. The spectrum of band-limited functions satisfy the condition:

$$X(w) = 0, \quad |w| > w_m. \qquad (D\text{-}58)$$

Digitization and sampling are synonymous. All sampled-signals are digitized. Digitization affects the spectrum of a signal. A spectrum that is nonzero only over the interval $(-w_s,w_s)$, must still have a Fourier-series of the form:

$$X(w) = \sum_{k=-\infty}^{\infty} a_k e^{-ikw/w_s}, \qquad \text{(D-59)}$$

where the a_k are given by:

$$a_k = \frac{\pi}{w_s} \int_{-w_s}^{w_s} X(w) e^{i\pi kw/w_s}\, dw. \qquad \text{(D-60)}$$

Although X(w) is nonzero only over the interval $(-w_s,w_s)$ the Fourier-series (D-59) is defined over $(-\infty,\infty)$ and may be interpreted as periodic repetitions of X(w) (ghosts) over the interval $(-\infty,\infty)$ (Figure D-12).

$w_s = 2\pi f_s$. f_s has units of time^{-1}, and, the quantity:

$$\tau_o = \pi/w_s = 1/2f_s \qquad \text{(D-61)}$$

is the minimum time-step for sampling the band-limited function x(t). This can be shown using (D-61) in (D-59) and (D-60). In that process it will also be seen that:

$$a_k = \tau_o x(k\tau_o), \qquad \text{(D-62)}$$

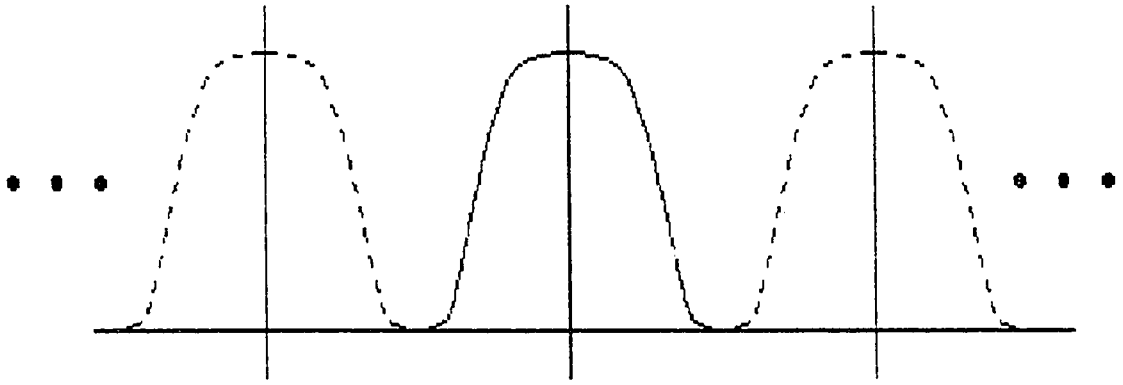


FIGURE D-12: Periodic repetition (ghosts) for the Fourier-series of a function which is non-zero over only a finite interval.

and further that:

$$x(t) = \sum_{k=-\infty}^{\infty} x(k\tau_o)\frac{\sin[\pi(k - t/\tau_o)]}{\pi(k - t/\tau_o)}. \qquad \text{(D-63)}$$

This (D-63) states that a function x(t) can be fully represented by its samples taken at times τ_o units apart provided (D-61) is satisfied. This is yet another expression of the sampling theorem. (D-63) holds for any time-interval $\tau \leq \tau_o$. w_s, as defined in presenting (D-61), is the folding-frequency. If X(w) is defined over the interval $(-w_m, w_m)$ with $w_m > w_s$, the Fourier-series (D-59) becomes:

$$X(w) = \sum_{k=-\infty}^{\infty} b_k e^{-ik\tau w}, \qquad \text{(D-64)}$$

with,

$$b_k = \tau \int_{-w_s}^{w_s} X(w) e^{ik\tau w} dw. \qquad \text{(D-65)}$$

where,

$$\tau \equiv \pi/w_m = 1/2f_m < \tau_o, \qquad \text{(D-66)}$$

and,

$$b_k/\tau = x(k\tau). \qquad \text{(D-67)}$$

Thus,

$$X(w) = \int_{-\infty}^{\infty} x(t)e^{-iwt}dt = \tau\sum_{k=-\infty}^{\infty} x(k\tau)e^{-ik\tau w}, \qquad (D\text{-}68)$$

which says that the samples $x(k\tau)$ determine the spectrum $X(w)$ exactly, provided,

$$\tau \leq \pi/2w_s = 1/2f_s, \qquad (D\text{-}69)$$

and,

$$X(w) = 0, \ |w| \geq w_s. \qquad (D\text{-}70)$$

Sampling at time-intervals shorter than τ_o simply moves the ghosts of $X(w)$ farther away from the central band $(-w_s, w_s)$. w_s is the same folding-frequency as described earlier here, and in Appendix-B, with regard to sampling and aliasing.

The line-spectrum (D-57) is subject to aliasing-phenomena as presented earlier in this appendix (Figures D-1,D-2) and as described for the power-spectrum presented in Appendix B.The inverse-Fourier-transform (D-63) is subject to Gibbs-phenomena ringing. Gibbs-phenomena is noise in the form of the Dirichlet-Kernel observed in the inverse-Fourier-transform at the relative positions of discontinulties in the original time-series. It is caused by the affects of truncation of the

input-sequence to the Fourier-transform on the spectrum, and is discussed more fully in Appendix B.

Truncation of Band-Limited Functions

In the prior discussions the length of a record-sequence has been assumed to be infinite. This is never true in real-life. Assume we have a sequence of length 2N+1. Then the Fourier-series (D-68) becomes:

$$X_{2N+1}(w) = \tau \sum_{k=-N}^{N} x(k\tau) e^{-ik\tau w}, \qquad \text{(D-71)}$$

which is most likely different than X(w). Using the expression for the Fourier-series of x(kτ) (D-63) in terms (D-68) (employing the diffraction-function) we can show for a set of n = 2N + 1 samples:

$$X_n(w) = \sum_{k=-N}^{N} a_k \frac{\sin[\frac{1}{2}n\pi(w - w_k)/w_s]}{\sin[\frac{1}{2}\pi(w - w_k)/w_s]} \frac{\pi}{w_s}. \qquad \text{(D-72)}$$

The spectrum of a truncated band-limited function (D-72) is not a true line-spectrum (D-57), but is a continuous-spectrum with discernable peaks centered at the discrete-frequencies implied by the summation over k (Figure D-13).

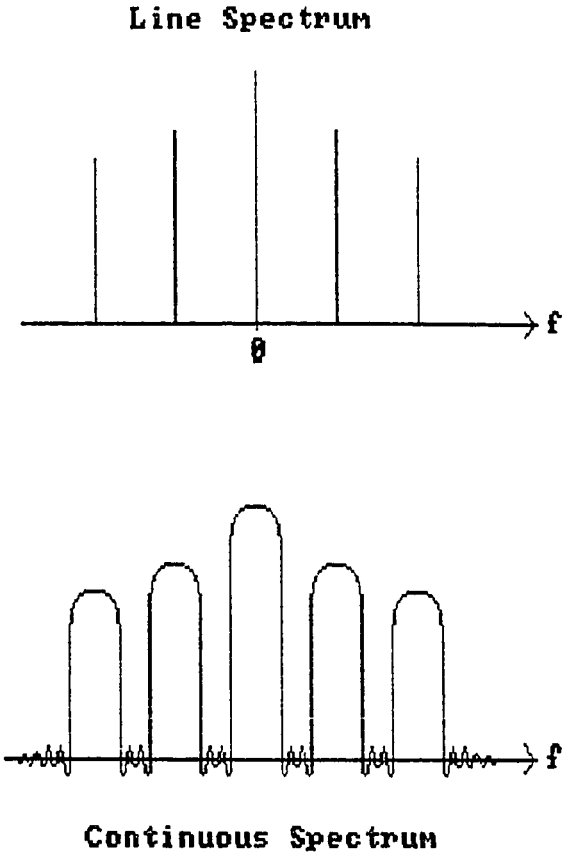


FIGURE D-13: Line spectrum, and spectrum of a truncated band-limited function.

In practice a true line-spectrum cannot be found for any record, rather only the continuous-spectrum as defined above (D-72). The peaks may overlap, and resolution of spectral-lines may be limited if they do. As n increases, the diffraction-function becomes sharper and spectral-lines are more easily discerned. The limit of resolution is dependent on n, which defines σ, the distance between two adjacent lines.

$\sin[\frac{1}{2}n\pi(w - w_k)/w_s] = 0$ for $(w - w_k) = \pm 2w_s/n$ if the first line is located at w_k. If the maximum for the peak corresponding to the second line (at w_{k+1}) is at or beyond the first zero of the diffraction-function pertaining to line w_k, there is a good chance that the second line will be recognized as distinct. This is the Rayleigh-criterion from optics, and is applicable here. If two lines fall on the frequencies w_k and w_{k+1}, they will be resolved if, noting $w = 2\pi f$:

$$|f_k - f_{k+1}| \geq 2f_s/n, \qquad \text{(D-73)}$$

or, since $n\tau$, the total time of sampling in units of τ, is $2T$,

$$n\tau|f_k - f_{k+1}| = 2T|f_k - f_{k+1}| \geq 1, \qquad \text{(D-74)}$$

which says,

$$|w_k - w_{k+1}| \geq 2\sigma, \qquad \text{(D-75)}$$

Note that:

$$\sigma = 1/4T, \qquad \text{(D-76)}$$

this is the elementary-frequency band. (D-74) says that the total relative change of phase between two frequencies, f_k and f_{k+1} should equal or exceed 1 cycle it they are to be resolved. (D-75) says that two lines are distinct if they are separated by two elementary-frequency bands.

σ is the lowest possible frequency that can be sensed from a record of duration 2T. Over an interval of 2T, the harmonic of frequency σ will barely have time to complete one-half oscillation. Any harmonic with a frequency of less than σ will be sensed as a trend in the record and cannot be resolved. σ is the minimum frequency that can be resolved and half the minimum frequency-step.

Time-Domain Filters

Any sequence of numbers has an associated spectrum (D-68). An operation or sequence of operations has its own spectrum and will modify the spectrum of a data-sequence $\{x_k\}$. The spectrum of an operation may or may not produce desirable effects in the spectrum of a data-sequence, which leads to the problem of finding a sequence $\{h_k\}$ that can be used as a filter.

If we add two observations $n\tau$ time-units apart, and equidistant from $x(k\tau)$, (which may or may not be observed), the result is:

$$x[(k + \tfrac{1}{2}n)\tau] + x[(k - \tfrac{1}{2}n)\tau]. \qquad \text{(D-77)}$$

This expression (D-77) may be viewed as the k^{th} term of the sequence resulting from the convolution of the sequence:

$$\{h_k\}, \quad k = \pm 1/2, \pm 3/2, \pm 5/2, \ldots \text{ for n even;}$$
$$k = 0, \pm 1, \pm 2, \pm 3, \ldots \text{ for n odd,} \qquad \text{(D-78)}$$

where,

$$h_k = \begin{cases} 1, & k = \pm n/2 \\ 0, & k \neq \pm n/2, \end{cases} \qquad \text{(D-79)}$$

with the infinite-sequence $\{x(k\tau)\}$, $k = 0, \pm 1, \pm 2, \ldots$ If the given operation of addition (D-77) is denoted by the operator A_n then the spectrum of A_n is:

$$A_n(w) = \sum_{k=-\infty}^{\infty} h_k e^{-k\tau w} = e^{-in\tau w} + e^{in\tau w} = 2\cos nw\tau/2. \qquad \text{(D-80)}$$

By similar means the spectrum for the summation of <u>n consecutive</u> observations can be determined as:

$$A_n(w) = \frac{\sin(n\tau w/2)}{\sin(\tau w/2)}. \qquad \text{(D-81)}$$

Note that $A_2 = A_1$, since the operation A_1 consists of adding 2 observations τ time-units apart. The form of the spectrum of A_n has

been presented before while discussing the Gibbs-phenomena in Appendix B, as the Dirichlet-kernel (B-58). Here we see that, considered as a filter, as $n \to \infty$, it tends to act as a low-pass function and reject all frequencies other than $w = 0$ (Figure D-14)

Particularly for small n, A_n allows passage of a number of unwanted frequencies. However, repeated convolutions in the time-domain are equivalent to applying a product of the spectrum of multiple operators, $A_n(w)$, to the spectrum of the signal, $X(w)$, in the frequency-domain. Using this fact, reasonable filters can be defined from their time-domain definition.

The form of the, so-called, Godin-filter used in this thesis is:

$$\frac{A_n A_{n+1}}{n(n+1)^2}. \qquad \text{(D-82)}$$

which is essentially an average of length n multiplied by two averages of length n+1. For $n = 24$, this operator's spectrum (D-82,Figure D-15) vanishes beyond $\tau w/\pi = 0.07$ and can be considered as a low-pass filter with a cutoff beyond this point. The operation requires a minimum of 3n observations. For $n = 24$ and $\tau = 1$ hour, it very effectively removes the semidiurnal and diurnal components of the tide.

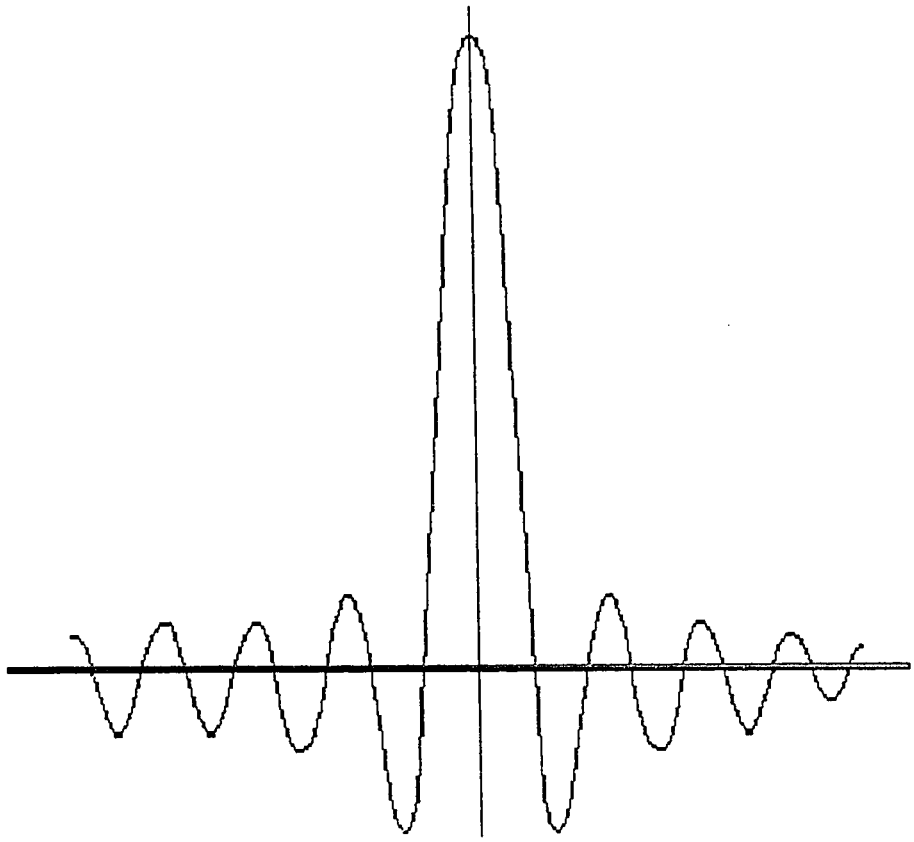


FIGURE D-14: The spectrum of A_n, operating as a low-pass filter as n approaches infinity.

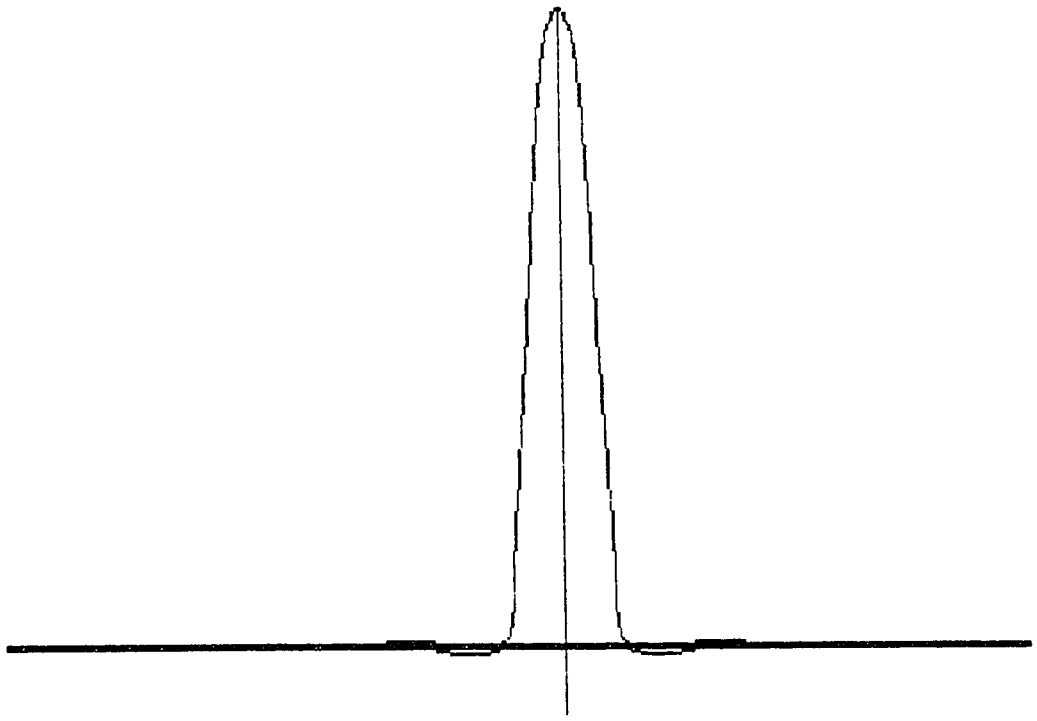


FIGURE D-15: The spectrum of the 'Godin filter' operator.

Consider 3n observations,

$$X_1, X_2, X_3, \ldots, X_n, \tag{D-83}$$

taken from $\{x(k\tau)\}$. Evaluating $A_{n+1}X$ requires calculating:

$$X_k = \sum_{j=0}^{n} X_{j+k}, \quad k = 1,2,\ldots,2n, \tag{D-84}$$

giving the sequence: $X_1, X_2, \ldots, X_{2n}$. Evaluating A_{n+1}^2X means calculating:

$$Y_m = \Sigma X_{k+m}, \quad m = 0,1,2,\ldots,n-1,n, \tag{D-85}$$

which yields the sequence: $Y_1, \ldots, Y_{n-1}, Y_n$. Evaluating $A_n A_{n+1}^2 X$ consists of calculating:

$$X_o = \sum_{k=1}^{n} Y_k. \tag{D-86}$$

The final result $\{A_n A_{n+1}/[n(n+1)^2]\}X$ is given by:

$$\frac{1}{n(n+1)^2} X_o = \bar{X}_{(3n+1)/2}, \tag{D-87}$$

which is the filtered value of x(t) falling at the center of the interval from which the input sequence (D-83) was chosen.

The above algorithm was implemented for $n = 24$, $\tau = 1$-hour in the FORTRAN program GODIN.FOR (Appendix F). The results of its application to the prepared data-set may be seen in the main-text section of this thesis entitled Time Domain Filtering.

APPENDIX E

MATHEMATICAL NOTATION AND DETAILS

Introduction

In this appendix mathematical details beyond those presented in the main text, and the other appendices, are presented. In the process of describing basic statistical, vector and matrix operations, as they relate to linear systems and the geometry of vector spaces, the notation used here and elsewhere in this thesis is presented. Matrix algebra, linear systems and statistics in matrix form are also covered. These discussions lead to the concept of orthogonality. Orthogonality, linear systems, and vector spaces all underlay the concept of eigenvalues and eigenvectors. The generation of eigenvalues and eigenvectors is the starting point for factor analysis as applied in this thesis. The geometric interpretation of eigenvalues and eigenvectors is extremely important for a good understanding of the results of factor analysis, and is covered here with particular emphasis on how this interpretation relates to factor analysis. Partial fraction expansion, mentioned in Appendix D as a tool for manipulation of the Z-transform, is presented via example.

Variables, Mean, Variance, Covariance and Correlations

Here we consider a variable as a collection of numerical values arranged in a column. For a variable, X, its component values are denoted x_i, where i runs from 1 to n, if n is the number of components in X. Two statistical properties of variables, important in determining the structure of linear systems, (which is the central problem of factor analysis), are the mean, (or expected value, a measure of central tendency), and variance, (a measure of the degree of dispersion within a variable).

The mean of a variable is computed as:

$$\bar{X} = \sum_{i=1}^{n} x_i/n = E(X) \qquad \text{(E-1)}$$

while its variance is given by:

$$VAR(X) = \sum_{i=1}^{n} (x_i - E(X))^2/n \qquad \text{(E-2)}$$

$$= E(X-E(X))^2$$

where n is the number of elements (observed values), x_i, in variable X.

If a variable is normally distributed, then the mean and variance are sufficient to characterize the entire probability distribution of the

variable. Variables with mean zero and variance 1 are standardized variables. Any variable can be transformed to a standardized variable by subtracting the variable's mean from the observed values and dividing those resulting values buy the square root of the variable's variance. No generality is lost in dealing with standardized variables.

The mean and variance of a single variable give rise to covariance, a measure of the correspondence of the variance in one variable to the variance in another variable, which is important in characterizing linear relationships between variables.

The covariance between two variables, X, Y, is expressed as:

$$COV(X,Y) = \sum_{i=1}^{n}((x_i - E(X)(y_i - E(Y))/n \qquad (E\text{-}3)$$

$$= E((X-E(X)(Y-E(Y))$$

Cases when elements in a variable equal the mean of the variable do not contribute to the magnitude of covariance. For cases when the respective element values are higher than the mean for one variable but lower than the mean for the other, a negative value is contributed to the covariance. In cases when their respective elements have either higher or lower value than the mean for both variables, then covariance is increased. Thus covariance measures the extent which values of one variable tend to covary (move) with values of another variable.

The covariance between standardized variables has a special name: the correlation coefficient or product-moment (Pearson's) correlation coefficient. Between two variables, X,Y, it is denoted: r_{XY}. Note that with E(X)=E(Y)=0, COV(X,Y)= E(XY), and with VAR(X)=VAR(Y)=1, then E(XY)=r_{XY}. The basic element with which factor analysis works is the correlation coefficient.

Correlation coefficients range from -1 to +1. If the correlation coefficient between two variables is negative, then a high score for one variable is accompanied by a low score for the other variable. A zero correlation coefficient means no prediction is possible with a linear equation, (i.e., the variables are statistically independent). A positive correlation coefficient means a high score for one variable is accompanied by a high score for the other and a low score for one variable is accompanied by a low score for the other. A correlation coefficient of +1 or -1 means that all the variance of one variable is predictable from the other variable, i.e., they are perfectly related. If it is possible for you to set the value of one of the variables then you have complete control over the other variable. If the distribution between two variables is bivariate normal then the means, variances and correlation between the two completely specify the bivariate distribution.

The relationship between two variables must be linear if the correlation coefficient is to serve as a valid index for that relationship. If the relationship between variables is nonlinear then the correlation

coefficient will be smaller than it should be. For variables to be considered linearly related, when one variable is plotted against another, the points should lie in a relatively straight line. It should be possible to draw a line through the plotted points so the sum of squares of deviations of points from this line is less than the sum of squares of deviations for any curve fitted to the plotted data. The line for this minimum sum of deviations is known as a regression line. All correlation procedures used in factor analysis assume the regression line is straight and only random error is responsible for deviations from it.

It is important to note that their covariation is independent of the underlying causal structure of two variables. That is, the covariance between two variables can express, e.g., how well two time series track each other, but it can not say why they track each other as well as they do. Variables may covary because one variable is a cause of the other, or because both variables share at least one common cause, or for both reasons.

The coefficient of determination, R^2, is equal to the correlation coefficient squared. It gives a measure of how predictable one variable is by knowing the score of the other variable. If one variable can be expressed as a linear function of the other, as in $Y = mX + b$, the correlation coefficient will be either +1 or -1 and the coefficient of determination will be 1.

Matrix Algebra

Matrices and Vectors

A matrix is a rectangular array of values composed of side-by-side variables. The matrix array is considered as a single unit rather than a collection of values, and is operated on as a unit. This results in great simplification in expressing complicated statements and relationships about the values within the matrix. Here we identify a matrix by script capital letters, such as $\mathcal{A}$. For a matrix, its element variables are identified by the capital letter corresponding to the script letter matrix identifier. A single subscript may be included with the variable identifier, e.g., A_i, which corresponds to the column position of the variable in the matrix, numbering from left to right. The individual values in a matrix are identified by double-subscripted small-case letters, e.g., a_{ij}, corresponding to the script matrix identifier. The first subscript identifies the row the value is in, and the second subscript identifies the value's column within the array. The maximum values for the row and column identifiers in a matrix are referred to as the dimension of the matrix. If there are m rows and n columns in a matrix, then that matrix is said to be of dimension m-by-n (dim mxn, or simply mxn)

A vector is the degenerate case for a matrix. That is, a vector is a linear collection of values, or, stated another way, a vector always has one dimension that is equal to one. A vector may be written either

horizontally, (a row vector), or vertically, (a column vector). For our purposes, the terms variable and column-vector are interchangeable. When meaning is not clear in context, vectors will often be seen expressed with an identifying arrow above the character naming the vector. Since vectors, e.g., $\vec{x}$ are linear arrays, their individual elements are identified by single subscripts, e.g., x_i. If a vector contains m elements, then its dimension is m. A vector with all elements equal to 1 is called the unit vector, (denoted $\vec{1}$). A vector with all elements equal to 0 is the null vector, (denoted $\vec{0}$).

Vector and Matrix Operations

Simple vector expressions and operations were used in previous sections while defining mean (E-1), variance (E-2) and covariance (E-3). Rather than write out definitions using summation notation, the operator, E (expected value), was defined to be the variable mean, containing the implied sum and division over the number of elements within a variable. This operator was applied in the subsequent definitions of variance and covariance, greatly simplifying their expressions. Operations on variables were assumed to take place element-by-element on the values contained within a variable, or between corresponding elements of variables.

A further example of simplification through these types of expressions was implied in the equation: $Y = mX + b$, presented above in describing the coefficient of determination. Since this equation contains

column variables, it implies the following set of equations:

$$\begin{aligned} y_1 &= mx_1 + b \\ y_2 &= mx_2 + b \\ &\vdots \\ y_n &= mx_n + b \end{aligned} \qquad \text{(E-4)}$$

The scaler values m, b, could themselves have been variables, M, B. In this case the individual m_i and b_i would have been included in the equation set (E-4).

The extension from these vector expressions and operations to true matrix expressions requires only that the vectors become matrices, i.e., row-and-column collections of values, and that the element-by-element operations are properly performed.

Vector and Matrix Addition

Vector addition is a direct-corresponding, element-by-element operation. Given vectors:

$$X = [1\ 2\ 3\ 4] \text{ and } Y = [5\ 6\ 7\ 8], \qquad \text{(E-5)}$$

then:

$$X + Y = [1+5\ \ 2+6\ \ 3+7\ \ 4+8] = [6\ \ 8\ \ 10\ \ 12] = Z. \qquad \text{(E-6)}$$

or, more formally:

$$Z = X + Y = [x_i + y_i],\ i=1 \text{ to } n, \qquad \text{(E-7)}$$

where the notation $[x_i + y_i]$ refers to the form of each element in the vector Z. This type of notation is used for vectors and matrices.

Given matrices:

$$\mathbf{X} = \begin{bmatrix} -6 & 3 \\ 2 & 1 \\ 0 & -1 \end{bmatrix}, \text{ and } \mathbf{Y} = \begin{bmatrix} 56 & -28 \\ -28 & 20 \\ 8 & 20 \end{bmatrix} \qquad \text{(E-8)}$$

then:

$$\mathbf{X} + \mathbf{Y} = \begin{bmatrix} -6+56 & 3+(-28) \\ 2+(-28) & 1+20 \\ 0+8 & -1+20 \end{bmatrix} = \begin{bmatrix} 50 & -25 \\ -26 & 21 \\ 8 & 19 \end{bmatrix} \qquad \text{(E-9)}$$

or, more formally:

$$\mathbf{X} + \mathbf{Y} = \mathbf{Z} = [x_{ij} + y_{ij}],\ i=1 \text{ to } n,\ j=1 \text{ to } m,\ \mathbf{X},\mathbf{Y}\ n\text{x}m. \qquad \text{(E-10)}$$

Note that the dimensions of both vectors, or both matrices, must match exactly to make their addition possible.

Vector and Matrix Transpose

The symbol ' is used to indicate the transpose operation for both vectors and matrices.

To transpose a vector means to rotate it ninety degrees, i.e., if it is a column vector turn it into a row vector, and if it is a row vector turn it into a column vector, e.g.,

$$\text{if } \bar{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}, \text{ then } \bar{X}' = [\, X_1 \quad X_2 \quad X_3 \,]. \qquad \text{(E-11)}$$

Note that $\bar{X}$ is a column vector and that $\bar{X}'$ is a row vector. In this thesis, assume that vectors without an indicated transpose operation are column vectors. A row vector will always include the transpose operator.

Transposing a matrix is similar to transposing a vector. Here all columns are changed into rows and all rows become columns, e.g,

$$\text{if } \mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_2{}^2 & X_{23} \\ X_{31} & X_{32} & X_{33} \\ X_{41} & X_{42} & X_{43} \end{bmatrix}, \text{ then } \mathbf{X}' = \begin{bmatrix} X_{11} & X_{21} & X_{31} & X_{41} \\ X_{12} & X_{22} & X_{32} & X_{42} \\ X_1{}^3 & X_{23} & x_{33} & X_{43} \end{bmatrix}. \qquad \text{(E-12)}$$

More formally:

$$\text{if } \mathbf{X} = [x_{ij}],\ \mathbf{X}\ n \times m, \text{ then } \mathbf{X}' = [x_{ji}],\ j=1 \text{ to } m,\ i=1 \text{ to } n. \qquad \text{(E-13)}$$

Scaler Multiplication of Vectors and Matrices

Multiplying a vector by a scaler means to multiply each element in the vector by the scaler value. For example:

$$\text{if } X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \text{ then } aX = \begin{bmatrix} ax_1 \\ ax_2 \\ ax_3 \end{bmatrix}. \qquad \text{(E-14)}$$

This concept carries over directly to the scaler multiplication of matrices by considering each column of a matrix to be a vector.

Vector Multiplication

Vector products can have several forms. Pre-multiplying a column vector by a row vector is called a minor (inner, dot or scaler) product. The result is a scaler value. Pre-multiplying a row vector by a column vector is called a major (outer, or cross) product. The result is a matrix.

minor products

Forming a vector minor product means multiplying all corresponding elements in two vectors, then summing all these products, e.g.,

$$\text{if } X' = [x_1\ x_2\ x_3],\ Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \text{ then } X'Y = x_1y_1+x_2y_2+x_3y_3. \quad \text{(E-15)}$$

Note that the dimension of the two vectors must be identical.

Note that the dimension of the two vectors must be identical.

major products

Forming a vector major product involves multiplying each element of the prefactor column vector across the values of the postfactor row vector. These cross products form the elements in the row of the product matrix that corresponds to the dimension of the element in the premultiplying column vector being used to form the cross products, e.g., using the notation of J. Imbre (from Joreskog, et al. 1976):

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} = \begin{bmatrix} x_1y_1 & x_1y_2 & x_1y_3 \\ x_2y_1 & x_2y_2 & x_2y_3 \\ x_3y_1 & x_3y_2 & x_3y_3 \end{bmatrix}. \qquad \text{(E-16)}$$

Note that the dimension of the two vectors must be identical.

vector product moments

When a vector is multiplied by its transpose, the result is referred to as a product moment. A minor product moment is a vector premultiplied by its transpose. A major product moment is a vector postmultiplied by its transpose.

Vector Geometry

A vector can be considered coordinates for the endpoint of a directed line segment starting at some origin in space. A directed line segment is a line segment that "points" in some direction. For vectors, their direction is from the origin to the coordinate values for the vector. Our familiar x and y coordinates are the endpoints of a vector in 2-dimensional space. This idea carries over to any number of dimensions. The concept of vectors as coordinates is helpful in understanding factor analysis.

Adding and subtracting vectors results in another vector. Multiplying a vector by a scaler value "stretches" or "contracts" the original vector.

The length (or magnitude) of a vector is determined by taking the square root of the squares of all its projections onto the coordinate axes, (i.e., the square root of the sum of the squares of all its component values). In two dimensions this amounts to applying the Pythagorean Theorem for a right triangle with one end of its hypotenuse fixed at the origin. Regardless of the number of dimensions, note that the length of a vector is the square root of its minor product moment, i.e., the length of a vector $\bar{x}$ is given by:

$$| \bar{x} | = (\bar{x}'\bar{x}) \qquad \text{(E-17)}$$

The cosine of the angle between two vectors is given by their inner product divided by the product of their lengths. In factor terminology, the angle between two vectors is the arccosine of the ratio of their minor product to the product of the square roots of their minor product moments. Given two vectors, $\overline{u}$, $\overline{v}$, if θ is the angle between them:

$$\cos\theta = \frac{\overline{v}'\overline{u}}{(\overline{v}'\overline{v})^{1/2}(\overline{u}'\overline{u})^{1/2}}. \qquad \text{(E-18)}$$

This expression (E-18), with rearrangement and use of the alternate (norm) notation given above for vector length (E-17) becomes the (perhaps) more familiar expression for the dot product from analytic geometry:

$$\overline{v}\cdot\overline{u} = |\overline{v}|\,|\overline{u}|\cos\theta \qquad \text{(E-19)}$$

The idea of the angle between two vectors becomes important when we examine a correlation matrix while performing a factor analysis procedure (Appendix C).

The Euclidian distance between two points is given by the square root of the minor product moment of their difference vector, i.e., for two vectors, V, U, considered as coordinates for two points in space, the distance between them is:

$$d = |\overline{v} - \overline{u}| = [(\overline{v} - \overline{u})'(\overline{v} - \overline{u})]^{1/2} \qquad \text{(E-20)}$$

Types of Matrices

A matrix with more columns than rows, or more rows than columns is called a rectangular matrix. If a matrix has the same number of rows as columns, then it is a square matrix. Note that for non-transposed square matrices, elements with identical-value subscripts will extend from the upper-left to lower-right corners, diagonally. These elements form the principle diagonal of a square matrix. The sum of elements on the principle diagonal is called the trace. The trace will be denoted here as Tr$\mathbf{x}$ for the matrix. $\mathbf{x}$ A symmetric matrix is one for which corresponding elements on opposite sides of the main diagonal are equal, i.e., if $x_{ij} = x_{ji}$ for all i,j, then $\mathbf{x}$ is symmetric. Note that a symmetric matrix equals its transpose. A diagonal matrix is a square, symmetric matrix that has all elements off the main diagonal equal to zero. The identity matrix is a diagonal matrix with all diagonal elements equal to 1. Here we denote the identity matrix as [1]. The identity matrix functions the same in matrix (linear) algebra as the value 1 does in scaler arithmetic. The null matrix is a matrix with all elements equal to zero. A null matrix works for matrix arithmetic as the value 0 does in scaler arithmetic. Here we denote the null matrix as [0].

Matrix Multiplication

Multiplying matrices amounts to forming all possible vector minor products between the rows and columns of the two matrices being multiplied. The rows of the prefactor matrix are taken to be the row vectors for the vector minor products, and the columns of the postfactor matrix taken to be the column vectors for the vector minor products. For example, if **X** is 3x4 and **Y** is 4x3, then:

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix} \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \\ y_{41} & y_{42} & y_{43} \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \end{bmatrix} \qquad \text{(E-21)}$$

where:

$$\begin{aligned} z_{11} &= x_{11}y_{11} + x_{12}y_{21} + x_{13}y_{31} + x_{14}y_{41} \\ z_{12} &= x_{11}y_{12} + x_{12}y_{22} + x_{13}y_{32} + x_{14}y_{42} \\ z_{13} &= x_{11}y_{13} + x_{12}y_{23} + x_{13}y_{33} + x_{14}y_{43} \\ &\vdots \\ z_{33} &= x_{31}y_{13} + x_{32}y_{23} + x_{33}y_{33} + x_{34}y_{43} \end{aligned} \qquad \text{(E-22)}$$

Since this (E-22) is a series of vector minor product moments, the row dimension of the prefactor matrix, **X**, must equal the column

dimension of the postfactor matrix, Y, for their multiplication to be possible. Note that the resultant matrix, Z, is square, with both its dimensions equal to the matching row and column dimensions of X and Y. In general, for an nxm matrix X and an mxn matrix Y, with product matrix XY = Z, Z mxm:

$$z_{ij} = \sum_{i=1}^{m} x_{ij} y_{ji}, \quad j = 1 \text{ to } n \qquad \text{(E-23)}$$

Note in some cases it is possible to reverse positions of matrices in the product, but in general this multiplication is not commutative, e.g.,

$$\begin{bmatrix} 3 & 17 & 10 \\ 12 & 14 & 11 \\ 13 & 29 & 6 \end{bmatrix} \begin{bmatrix} 4 & 27 & 18 \\ 12 & 25 & 12 \\ 10 & 23 & 16 \end{bmatrix} = \begin{bmatrix} 316 & 736 & 418 \\ 326 & 927 & 560 \\ 460 & 1214 & 678 \end{bmatrix} \qquad \text{(E-24)}$$

but,

$$\begin{bmatrix} 4 & 27 & 18 \\ 12 & 25 & 12 \\ 10 & 23 & 16 \end{bmatrix} \begin{bmatrix} 3 & 17 & 10 \\ 12 & 14 & 11 \\ 13 & 29 & 6 \end{bmatrix} = \begin{bmatrix} 570 & 968 & 445 \\ 492 & 1076 & 467 \\ 304 & 956 & 449 \end{bmatrix} \qquad \text{(E-25)}$$

Linear Systems

Simultaneous Linear Equations

A point of view useful in understanding factor analysis is that a set of n simultaneous linear equations in n unknowns may be considered a mapping for an n-dimensional vector over n independent n-dimensional vectors whose (weighted) sum in n-space is equal to the original vector. All that was simply a convoluted description of coordinate axes. For example, in 3-space (the space of our familiar 3-dimensional coordinate system, $\mathcal{A}^3$), such a set of equations might be:

$$\begin{aligned} 2a + 5b + \quad 6c &= 3 \\ 1a + 2b + \quad 3c &= 9 \\ -3a + 2b + (-2)c &= -4 \end{aligned} \qquad \text{(E-26)}$$

Using vector notation, this set of equations can be expressed as:

$$a\begin{bmatrix} 2 \\ 1 \\ -3 \end{bmatrix} + b\begin{bmatrix} 5 \\ 2 \\ 2 \end{bmatrix} + c\begin{bmatrix} 6 \\ 3 \\ -2 \end{bmatrix} = \begin{bmatrix} 3 \\ 9 \\ -4 \end{bmatrix} \qquad \text{(E-27)}$$

Giving the right-hand-side vector as linear combination of the three left-hand-side vectors. Which can be further reduced by vector notation to:

$$a\overline{x} + b\overline{y} + c\overline{z} = \overline{u} \qquad \text{(E-28)}$$

Recall that vectors are directed line segments. Consider the unknown weights, a,b,c, as distances. Then traveling from the origin a units in the direction of $\bar{x}$, then b units in the direction of $\bar{y}$, and c units in the direction of $\bar{z}$, you finally reach the endpoint of the directed line segment, $\bar{u}$.

Linear Algebra

The central problem of linear algebra is the solution of simultaneous linear equations. When the number of equations and unknowns becomes large, then scaler sums of products notation becomes unwieldy. A change to vector notation helps in this regard, but a further simplification in expression can be gained if the change is made to matrix notation for large equation sets.

Recall that the product of two matrices is a collection of vector minor products, and, further, that a vector is the degenerate case of a matrix. With these ideas in hand, looking at the original equation set (E-26) expressed as vector operations (E-27), one can see how this equation set may be expressed as a matrix product, i.e.,

$$\begin{bmatrix} 2 & 5 & 6 \\ 1 & 2 & 3 \\ -3 & 2 & -2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 3 \\ 9 \\ -4 \end{bmatrix} \qquad \text{(E-29)}$$

Which can then be simplified even further as:

$$\mathcal{B}\vec{a} = \vec{c}. \qquad \text{(E-30)}$$

The matrix, $\mathcal{B}$, is produced by combining the three column vectors formed from the coefficients of the origin three equations into a three column matrix, and the vector of unknown weights, $\vec{a}$, is formed as a column vector from the unknown scaler weights. The unknown weight vector is then premultiplied by the coefficient matrix so that the number of rows and columns is correct for the implied matrix multiplication. The result is the original vector, $\vec{c}$.

matrix inverse

Solutions to linear algebraic equations are analogous to solutions of simple algebraic equations. For the case of n equations in n unknowns, one solves for $\mathcal{B}$ by rewriting the equation $\mathcal{B}\vec{a} = \vec{c}$ as $\vec{a} = \mathcal{B}^{-1}\vec{c}$, where the notation $\mathcal{B}^{-1}$ refers to the <u>inverse</u> of $\mathcal{B}$. Multiplying by the inverse of a matrix is the analog of division in simple algebra. Note that:

$$\mathcal{B}^{-1}\mathcal{B} = \mathcal{B}\mathcal{B}^{-1} = [1]. \qquad \text{(E-31)}$$

In factor analysis we deal with correlation or covariance matrices, which are square. One technique for finding the inverse of square matrices, which lends itself well to use on computers, is the method of

row reduction of an augmented matrix. The augmented matrix is the matrix we wish to find the inverse of, with the identity matrix appended to its right side, e.g., if:

$$\mathcal{B} = \begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix} \text{ its augmented matrix is } \left[\begin{array}{cc:cc} 1 & 2 & 1 & 0 \\ 3 & 7 & 0 & 1 \end{array}\right]. \qquad \text{(E-32)}$$

Once the augmented matrix has been formed, then elementary row operations, (as used in Gaussian Elimination), executed across the entire matrix are used to reduce its left side to an identity matrix. For our example (E-32), the operations are as follows:

$$\left[\begin{array}{cc:cc} 1 & 2 & 1 & 0 \\ 3 & 7 & 0 & 1 \end{array}\right] \xrightarrow[(-1/3)r_2]{} \left[\begin{array}{cc:cc} 1 & 2 & 1 & 0 \\ -1 & -7/3 & 0 & -1/3 \end{array}\right] \qquad \text{(E-33a)}$$

$$\left[\begin{array}{cc:cc} 1 & 2 & 1 & 0 \\ -1 & -7/3 & 0 & -1/3 \end{array}\right] \xrightarrow[r_2 + r_1]{} \left[\begin{array}{cc:cc} 1 & 2 & 1 & 0 \\ 0 & -1/3 & 1 & -1/3 \end{array}\right] \qquad \text{(E-33b)}$$

$$\left[\begin{array}{cc:cc} 1 & 2 & 1 & 0 \\ 0 & -1/3 & 1 & -1/3 \end{array}\right] \xrightarrow[6r_2]{} \left[\begin{array}{cc:cc} 1 & 2 & 1 & 0 \\ 0 & -2 & 6 & -2 \end{array}\right] \qquad \text{(E-33c)}$$

$$\left[\begin{array}{cc:cc} 1 & 2 & 1 & 0 \\ 0 & -2 & 6 & -2 \end{array}\right] \xrightarrow[r_1 + r_2]{} \left[\begin{array}{cc:cc} 1 & 0 & 7 & -2 \\ 0 & -2 & 6 & -2 \end{array}\right] \qquad \text{(E-33d)}$$

$$\left[\begin{array}{cc:cc} 1 & 0 & 7 & -2 \\ 0 & -2 & 6 & -2 \end{array}\right] \xrightarrow[(-1/2)r_2]{} \left[\begin{array}{cc:cc} 1 & 0 & 7 & -2 \\ 0 & 1 & -3 & 1 \end{array}\right] \qquad \text{(E-33e)}$$

$$\text{So, if } \mathcal{B} = \begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix}, \text{ then } \mathcal{B}^{-1} = \begin{bmatrix} 7 & -2 \\ -3 & 1 \end{bmatrix}. \qquad \text{(E-33f)}$$

Checking the result:

$$\begin{bmatrix} 7 & -2 \\ -3 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = [1] \qquad \text{(E-34)}$$

Vector Spaces and Subspaces

Clearly a linear system such as $\mathcal{A}\vec{a} = \vec{c}$ is solvable for $\vec{a}$ only if the vector $\vec{c}$ can be expressed as some combination of the columns of the coefficient matrix $\mathcal{A}$. Each solution vector, $\vec{a}$, is one element in the set of all possible solutions to $\mathcal{A}\vec{a} = \vec{c}$. The totality of solutions, $\{\vec{a}\}$ forms the vector space defined by $\mathcal{A}\vec{a} = \vec{c}$. If $\mathcal{A}$ is composed of three columns that are mutually orthogonal 3-dimensional vectors, then the vector space defined by $\mathcal{A}\vec{a} = \vec{c}$ is our familiar 3-space. If $\mathcal{A}$ is formed from two 3-dimensional column vectors and $\mathcal{A}$ is 2-dimensional, then $\mathcal{A}\vec{a} = \vec{c}$ defines a set of three equations in 2 unknowns, e.g.,

$$a\begin{bmatrix} 2 \\ 1 \\ -3 \end{bmatrix} + b\begin{bmatrix} 5 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 9 \\ -4 \end{bmatrix} \qquad \text{(E-35)}$$

The solution to this expression (E-35) is a subset of all of vectors in 3-space, forming a plane in 3-space. This portion of 3-space is referred to as a vector subspace of 3-space. Note that vector spaces are closed

under addition and scaler multiplication, i.e., these operations will not generate a new vector that is outside the vector space, and that any vector space contains the null vector. We will use a script character with a superscript vector arrow, e.g., $\vec{\mathcal{A}}$ as notation for a vector space or subspace.

Linear Independence

If a set of vectors, $\vec{X}_i$, i=1 to n, is made up such that no linear combination of these vectors:

$$c_1\vec{X}_1 + c_2\vec{X}_2 + \ldots + c_n\vec{X}_n = 0, \qquad \text{(E-36)}$$

unless all the $c_i = 0$, then these vectors are said to be linearly independent. If a linear combination of these vectors can be found that equals zero when some of the $c_i \neq 0$, then (at least) one of the vectors is a linear combination of the others, and this set is said to be linearly dependent. The columns of:

$$\mathbf{X} = \begin{bmatrix} 6 & 3 & 9 \\ 2 & 7 & -3 \\ 4 & 8 & 0 \\ -1 & 16 & -14 \end{bmatrix} \qquad \text{(E-37)}$$

Are linearly dependent. Note that:

$$2X_1 + (-1)X_2 + X_3 = 0. \qquad \text{(E-38)}$$

The columns of an identity matrix, [1], of any dimension, are linearly independent. A special notation, e_i, is often given to the columns of the identity matrix, where i refers to the dimensional level of the element 1 in a particular column. These particular vectors are vectors of unit length in the coordinate directions. They are referred to as the coordinate vectors and useful as operators in linear algebra. For the three dimensional case:

$$[1] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{(E-39)}$$

and,

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \; e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \; e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \text{(E-40)}$$

Say that:

$$c_1 e_1 + c_2 e_2 + c_3 e_3 = \overline{c} \qquad \text{(E-41)}$$

Clearly $\overline{c} = \overline{0}$, if and only if all the $c_i = 0$, therefore, the e_i are linearly independent.

Basis Vectors

If all possible linear combinations of a set of vectors, $\vec{a}_i$, make up all the possible vectors within a vector space, $\vec{A}$, then the vectors, $\vec{a}_i$, are said to span the vector space, $\vec{A}$. This simply says that the vectors, $\vec{a}_i$, may be considered as a form of coordinate axes within the space represented by $\vec{A}$. If the vectors, $\vec{a}_i$, are linearly independent and span the space $\vec{A}$, then the $\vec{a}_i$ are said to form a basis for the vector space $\vec{A}$. This says that in some sense the $\vec{a}_i$ are the minimum set of vectors required to form a set of coordinate axes for the vector space $\vec{A}$. The coordinate vectors, e_i, i=1 to n, are linearly independent, span n-dimensional space, and therefore form a basis for n-dimensional space, R^n.

All bases for a vector space contain the same number of basis vectors. This number expresses the degrees of freedom of the vector space and is referred to as the dimension of the vector space. Any linearly independent set of vectors, $\vec{a}_i$, in a vector space, $\vec{A}$, can be extended to a basis for $\vec{A}$ by adding more linearly independent vectors if necessary. Any spanning vector set in a vector space can be reduced to a basis for that space by eliminating vectors if necessary.

Matrix Rank

A set of vectors that spans a space may be expressed in matrix form and then considered from the point of view of row space or column space, i.e., whether the rows make up the vectors that span the space, or the columns are the vectors that span the space. The dimension of its row space equals the rank, r, of a matrix, which is also equal to the dimension of the column space for the matrix.

The column space of a matrix, $\mathbf{X}$, is often referred to as the range of $\mathbf{X}$. This is analogous to the idea of the range of a function. For a function y=f(x), all values, x, for which the function is defined gives the domain of f, and the resulting values, y, give the range of f. Here we are considering vector functions, which we express as $f(\overline{X}) = \mathbf{A}\overline{X}$. For the n-dimensional case, its domain is all vectors, $\overline{X}$, in R^n, and its range is all vectors, $\overline{C}$, for which $\mathbf{A}\overline{X} = \overline{C}$ can be solved.

The row space or column space dimension for a matrix is not the simple dimension we have used before, expressed as mxn for a matrix of m rows by n columns. For a matrix, A, the number of nonzero rows in an upper-echelon matrix, $\boldsymbol{U}$, produced from $\mathbf{A}$ through elimination by elementary row operations, is the rank, r, of $\boldsymbol{U}$. A basis for $\boldsymbol{U}$ is formed by the nonzero rows of $\boldsymbol{U}$. Elementary row operations do not change the row space of a matrix. That is, each new row of $\boldsymbol{U}$ is simply a combination of rows of $\mathbf{A}$, so the new row space contains the old row

space. Also, because each elementary row operation is reversible, the old row space is contained in the new row space. Therefore, the row space of $\mathbf{A}$ has the same dimension, r, as the row space of $\mathbf{U}$, and the same basis as $\mathbf{U}$, because the two row spaces are the same. So, the rank of $\mathbf{A}$ is the same as the rank of $\mathbf{U}$, and the problem of finding the rank of $\mathbf{A}$ becomes the problem of generating the matrix $\mathbf{U}$.

Generating $\mathbf{U}$ from $\mathbf{A}$ by elementary row operations is simply approaching $\mathbf{A}$ as though attempting to solve a set of simultaneous equations by the elimination method. That is, use elementary row operations to force the rows of $\mathbf{A}$ to contain as many leading zeros as possible, i.e., forcing entries below the main diagonal to be zero, and forcing the nonzero entries into an echelon (or staircase) form. This procedure is that of factorizing a matrix into UL form. The mechanics, which can be found in any linear algebra text, are not critical to understanding factor analysis. The leading nonzero elements in each row are referred to as pivots. The number of pivots after the factorization is complete is the rank of the matrix.

Factorizing a rectangular matrix amounts to solving a set of equations where the number of equations is not equal to the number of unknowns. If after factorization, there were r nonzero pivots, and m zero rows, then there will be r basic variables (those for which values are determined), and m free variables (those whose values are arbitrary), in the solution.

The idea of the rank of a matrix is important in determining the underlying structure of a linear system. Relating to factor analysis, the rank of a matrix gives the minimum number of variables (axes) required to completely describe the variance in a data set. The determination of rank in factor analysis is not made explicitly, but "comes out for free" in the computation of eigenvalues and eigenvectors for a data matrix. The eigenvectors being a set of orthogonal basis vectors for the space of the data matrix, and eigenvalues being weights for the eigenvectors used in determining the principle factors for the data matrix.

Orthogonality and Least Squares Approximations

In the preceding sections you probably assumed orthogonality for coordinate axes as the various matrix manipulations were being described. Axes do not have to be orthogonal (at right angles), though if they are not, and they span a space, redundant position information is generated by using those axes within the space they span. Sometimes one will see non-orthogonal factor axes being used in factor analysis. This complicates the picture, and in most physical studies is not likely of any major benefit in determining processes, when an orthogonal set can be found.

Non-orthogonal vectors have some projection on each other. That is, a perpendicular through one vector will intersect the other vector. The line segment along the vector from the origin to this point of

intersection is called the projection of the first vector onto the second (figure E-1).

Two vectors, $\bar{v}$, $\bar{w}$, are said to be orthogonal if their inner product $\bar{v}'\bar{w}$ is zero. Recall that the cosine of the angle between two vectors is a ratio of this value to the product of their lengths. Since $\cos(90^0) = 0$, two vectors must be at right angles if their inner product is zero. When vectors are orthogonal, the length of the projection of one onto the other is zero.

A matrix is said to be orthogonal if its columns, as vectors, are mutually orthogonal. If the result of premultiplying a matrix by its transpose is a diagonal matrix, then the matrix is orthogonal. Note the significance of this operation. Only one value shows up in each column of the product matrix, where each column vector is multiplied by its transpose. The column vectors are linearly independent so they are correlated only with themselves. The diagonal values amount to the covariances for the matrix. The off diagonal correlations are zero.

A subspace, $\bar{\bar{Y}}$, of some space R^n is said to be orthogonal if every vector $\bar{y}_m$ in $\bar{\bar{Y}}$ is orthogonal to every vector $\bar{y}_j$ in $\bar{\bar{Y}}$, $i \neq j$. If also $\bar{\bar{Y}}$ spans R^n, then every vector in R^n can be described as a linear combination of the $\bar{y}_i$, and there is a simple formula for this combination for every vector in R^n. It amounts to determining the projection of the vector (point) in question onto every vector in $\bar{\bar{Y}}$.

$$r = \sqrt{(c-a)^2 + (d-b)^2}$$

$$d = r\sin(\theta)$$

$$L = \sqrt{r^2 - d^2}$$

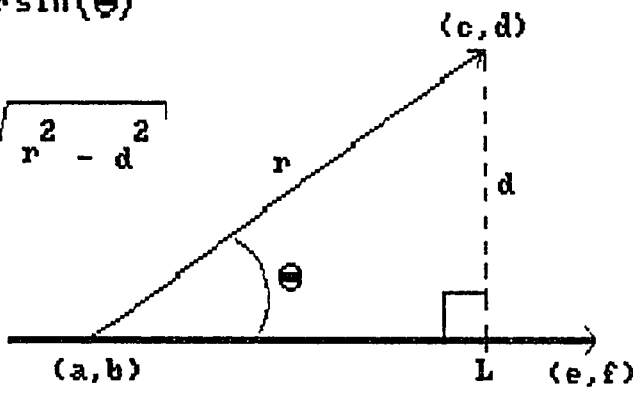


FIGURE E-1: Vector projections.

Orthogonal vectors are linearly independent. If a set of orthogonal vectors spans a space, then that set of vectors is a basis for the space, in fact, an orthogonal basis.

Suppose we have a multivariate data set, (i.e., a data set that can be written down in the form of a matrix whose columns are values of measurements on different variables), that would ideally be spanned by an orthogonal subspace of that set, but where errors in measurement have made this untrue. That is, some of the data values fall outside of the space spanned by the orthogonal subspace it was assumed should span the data space. If we have an orthogonal basis that nearly spans the data space, then, by proper manipulation, we can approximate points for the data that fall outside the space of the orthogonal basis such that the error between the data point and the approximated point is minimized in the least squares sense for every vector in our basis set. It is exactly this process being performed in extracting factors to analyze the variance in a data set by factor analysis.

Each factor explains a certain percentage of the variance in a data set. The complete set of factors explains all the variance in the data set. If only successive factors are retained until some preset limit on the percent of variance explained in the data set is reached, then the error in describing the variance in the data set by eliminating the smaller factors is the minimum possible in a least squares sense. As was the case for calculating the rank of a matrix, this least squares feature of

factor analysis is not made as explicit calculations, but "comes out for free" in the calculation of eigenvectors and eigenvalues for the correlation matrix from the data matrix.

Determinants

It can be shown that if a matrix, $\mathbf{A}$, has a nonzero determinant ($\det\mathbf{A} \neq 0$) then $\mathbf{A}^{-1}$ exists, i.e., $\mathbf{A}$ is invertible. If $\det\mathbf{A} = 0$, then $\mathbf{A}$ is singular, and no $\mathbf{A}^{-1}$ exists. Determinants can be used to solve simultaneous linear equations. Determinants are essential in factor analysis because they are used to derive eigenvalues and eigenvectors from a data matrix.

Determinants can be calculated recursively from rote formulas. The recursion stopping condition (simplest) determinant is the 2x2 case:

$$\det\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ab - cd \qquad \text{(E-42)}$$

Calculation of determinants for larger square matrices is made by row expansion or column expansion into minors until the simplest form is reached for the actual calculation as a weighted sum of these minors, e.g., expanding a 4x4 determinant by its first row:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} & a_{24} \\ a_{32} & a_{33} & a_{34} \\ a_{42} & a_{43} & a_{44} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} & a_{24} \\ a_{31} & a_{33} & a_{34} \\ a_{41} & a_{43} & a_{44} \end{vmatrix}$$

$$+ a_{13} \begin{vmatrix} a_{21} & a_{22} & a_{24} \\ a_{31} & a_{32} & a_{34} \\ a_{41} & a_{42} & a_{44} \end{vmatrix} - a_{14} \begin{vmatrix} a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{vmatrix} \quad \text{(E-43)}$$

where the 3x3 minors (E-43) are further reduced to a sum of 2x2 minors for solution.

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}$$

$$+ a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \quad \text{(E-44)}$$

Note carefully the alternation in sign between minors, and the subscript pattern within minors relative to their scaler premultiplier's subscripts. The scheme expands for any nxn matrix.

Eigenvalues and Eigenvectors

If:

$$\mathcal{A}\bar{u} = \lambda\bar{u}, \quad \text{(E-45)}$$

and $\bar{u} \neq \bar{0}$, where λ is an unknown scaler, then $\bar{u}$ is an <u>eigenvector</u> of $\mathcal{A}$. The problem is to find a value of λ that satisfies the eigenvector equation (E-45).

$\mathcal{A}\bar{u} = \lambda\bar{u}$ can be rewritten as $(\mathcal{A} - \lambda[1])\bar{u} = \bar{0}$, where [1] is the identity matrix. Note, if $(\mathcal{A} - \lambda[1])$ is nonsingular, then $(\mathcal{A} - \lambda[1])^{-1}$ can be used to form a solution. But, this is the trivial case: $\bar{u} = \bar{0}$. If we are to have $\bar{u} \neq \bar{0}$, then $(\mathcal{A} - \lambda[1])$ must be singular. This says that $\det(\mathcal{A} - \lambda[1]) = 0$.

As a set of linear equations for the 2x2 case, $(\mathcal{A} - \lambda[1])\bar{u} = \bar{0}$ becomes:

$$\begin{aligned} (r_{11} - \lambda)u_1 + r_{12}u_2 &= 0 \\ (r_{21} - \lambda)u_1 + (r_{22} - \lambda)u_2 &= 0 \end{aligned} \qquad \text{(E-46)}$$

and,

$$\det(\mathcal{A} - \lambda[1]) = \begin{vmatrix} r_{11}-\lambda & r_{12} \\ r_{21} & r_{22}-\lambda \end{vmatrix} = 0 \qquad \text{(E-47)}$$

or,

$$\lambda^2 - \lambda(r_{11} + r_{22}) + (r_{11}r_{22} - r_{12}r_{21}) = 0 \qquad \text{(E-48)}$$

The last result is called the characteristic equation of $\mathcal{A}$. Its roots are called the characteristic values or eigenvalues of $\mathcal{A}$. If $\mathcal{A}$ is a symmetric matrix, with real values, then the eigenvalues are real. Some may not be unique, and some may be zero.

The eigenvectors of $\mathcal{A}$ are determined by substituting the calculated eigenvalues back into the original equation $\mathcal{A}\bar{u} = \lambda\bar{u}$, and solving for $\bar{u}$. In general, $\mathcal{A}\bar{u}$ is not a simple multiple of $\bar{u}$, $\lambda\bar{u}$. This means that only certain special values are eigenvalues, and certain special

vectors are eigenvectors. Note that if $\bar{u}$ is a solution, then so is $c\bar{u}$, c a scalar. Because of this, eigenvectors are always normalized.

Eigenvalues and eigenvectors follow from solution of the general first-order initial-value problem, i.e.,

$$\frac{d\bar{u}}{dt} = A(t)\bar{u} + \bar{b}(t), \quad \bar{u} = \bar{u}_0 \text{ at } t = 0. \qquad \text{(E-49)}$$

which leads to $A\bar{x} = \lambda\bar{x}$, the fundamental equation for the eigenvector $\bar{x}$ and eigenvalue λ. The differential equation has pure exponential solutions:

$$\bar{u} = e^{\lambda t}\bar{x}. \qquad \text{(E-50)}$$

The eigenvalue gives the rate of growth or decay, and the eigenvector develops at this rate. The important thing to note here is that eigenvectors do not change direction. The are fixed in space.

Matrices can be interpreted geometrically, as a set of coordinate points in n-space. Consider the 2xn case for a matrix. The first row in the matrix can be considered the x,y coordinates of one point, and the second row the coordinates of another point, and so on. That is a 2xn matrix defines a set of vectors in a plane. This idea expands to any number of coordinates and any number of vectors. Eigenvectors and eigenvalues also have a geometric interpretation.

Consider the 2x2 case for a matrix, e.g.,

$$A = \begin{bmatrix} 3 & 6 \\ 6 & 3 \end{bmatrix} \qquad \text{(E-51)}$$

We can think of the two points, (3,6) and (6,3), as lying on a ellipse whose center is the origin of a coordinate system. The eigenvalues represent lengths of the major and minor axes of the ellipse. The eigenvalues of the example matrix (E-51) are: $\lambda_1 = 9$, $\lambda_2 = -3$. The first one represents the length of the major axis of the ellipse and the second represents the length of the ellipse's minor axis. Note that for a symmetric 2x2 matrix, as in our example, the eigenvector corresponding to the largest eigenvalue will bisect the vector represented by the matrix values. By definition the second eigenvector is orthogonal to the first (figure E-2).

The ratio of the length of the major axis of an ellipse to its minor axis length gives a measure of the elongation of the ellipse. If the two points on the ellipse represented by the values in our 2x2 matrix are equal, then the ellipse collapses into a line. If these points are the endpoints of perpendicular vectors of the same length, then the ellipse becomes a circle. The idea of the ratio of lengths of eigenvectors helps in explaining the underlying structure of linear systems and factor analysis. If the matrix represents the covariance of a set of variables, then the eigenvectors "point in the direction" of ooooofigure 2

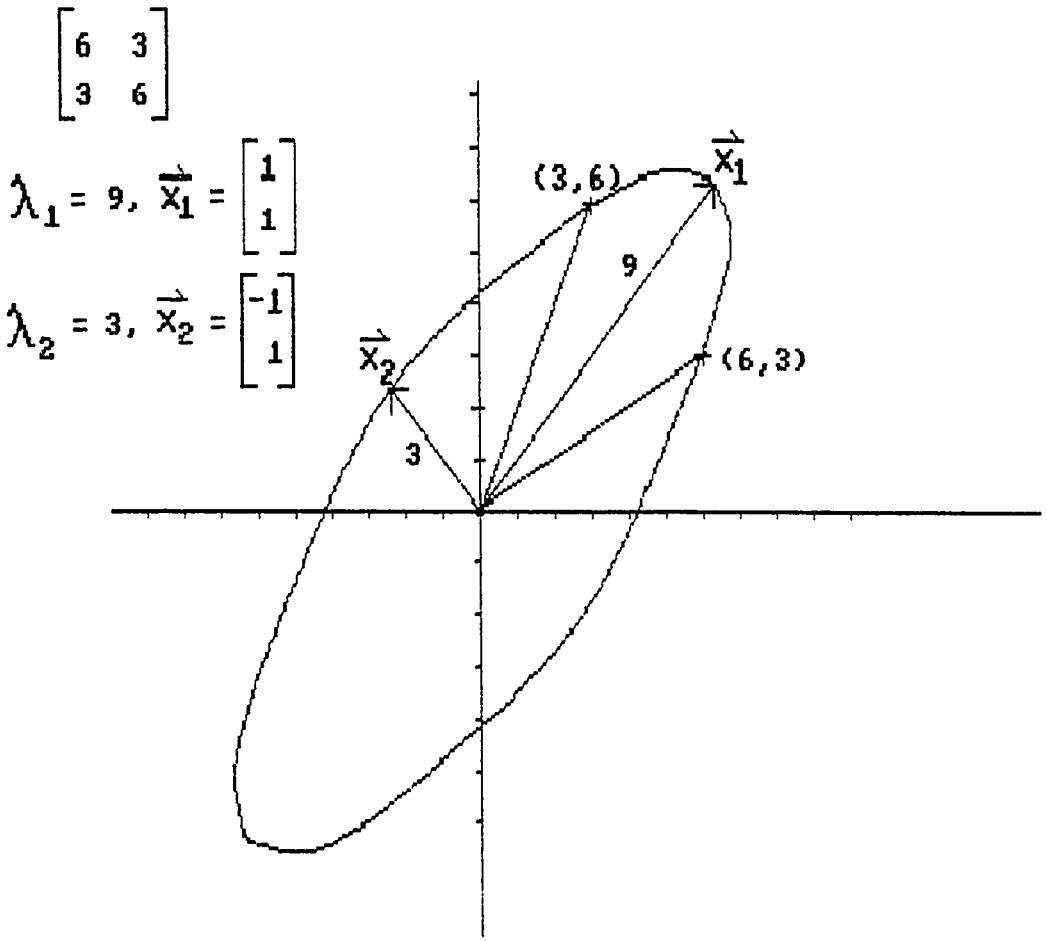


FIGURE E-2: Geometric interpretation of eigenvectors and eigenvalues.

collections of variables (columns, which themselves are vectors) in the matrix that tend to covary, that is, in the direction of variables that all seem to be explained by one major source of variation in the data. The length of an eigenvector, i.e., it's eigenvalue, expresses how strong this covariation among the variables is relative to the other apparent (orthogonal) sources of variation, i.e., the other eigenvectors. If the variance in a data set is chiefly explained by one source, then one eigenvalue will be much greater than all the others. If the variance is poorly explained, then all the eigenvectors will tend to have equal eigenvalues. The measures given by eigenvalues and eigenvectors fall as close as possible to a combined covariance between all the variables they are trying to explain the variance for, in a least-squares sense, and hence relate all these variables to a single theoretical variable (factor).

Statistics in Matrix Form

Performing statistics using the traditional methods and equations on very large data sets is tedious to say the least. And, the results tend to get lost in the calculations. Factor analysis is usually performed on very large data sets with many variables. In fact, since it is a statistical procedure, it works better if the number of variables, and the number of elements within variables is large. Hence, calculations are generally left to computers, and, for simplicity of expression, statistical and other operations applied in factor analysis are given in matrix form.

In R-mode factor analysis we are mainly concerned with the covariance or correlation matrix from a data set. Using one or the other says something about the kind of result you want from the analysis (see Appendix C).

The mean of a matrix is a vector containing the individual means of all the columns in the matrix. Expressing this in matrix notation:

$$\bar{X}' = \bar{T}'X/\bar{T}'\bar{T} \qquad \text{(E-52)}$$

Look carefully at how the notation works. The dimension of a unit vector is always assumed to be the correct dimension for the problem at hand. The vector of averages, (as a row vector), is the result of premultiplying the matrix to be averaged by the transposed unit vector, then dividing this by the vector minor product of the unit vector. The final values are the averages of each column in the original matrix in the position in the resulting vector that corresponds to the position of their column in the original matrix. Premultiplying a matrix by the transposed unit vector simply gives a vector that contains the column sums for the matrix. The vector minor product of the unit vector gives the number of rows in the matrix. So, the above expression is a very compact way of expressing the single variable mean for each column in a matrix via one equation.

The difference between an observation within a variable and the mean for that variable is the deviation for that observation. The original observations are called the raw scores, and the collection of deviations is called the deviate scores. For an nxm raw scores matrix, $\mathbf{X}$, the nxm deviate scores matrix, $\mathbf{Y}$, is given by:

$$\mathbf{Y} = [\, x_{ij} - \bar{X}_j \,], \; i=1 \text{ to } n, \; j=1 \text{ to } m, \qquad \text{(E-52)}$$

where: $\bar{X}_j$ is the j^{th} element of $\bar{X}$.

The variance of a variable is a measure of the deviation of individual elements in the variable about the mean of the variable. It is formally defined as the average of the sum of the squared deviate scores. When considering the variance of a matrix, we look, as was the case for the matrix mean, column-by-column. If $\mathbf{Y}$ represents the deviate scores for $\mathbf{X}$, then the variance in $\mathbf{X}$ is:

$$\text{VAR}(\mathbf{X}) = s^2 = \sum_{i=1}^{n} y_i / n \qquad \text{(E-53)}$$

Note two things here: (1) the use of s^2 to represent variance (E-53), and (2) division of the deviate scores by n rather than by n-1. s^2 is common statistical nomenclature for variance. Dividing the deviate scores by n rather that n-1 gives the so-called biased estimate of variance. However, once the number of elements used to calculate the variance becomes large, (i.e., > 10), this difference is usually of little concequence and the biased estimate is acceptable, (and used here).

To express the variance of the columns of a matrix, we use notation similar to that for calculating the matrix mean. Again assume $\mathbf{Y}$ is the deviate scores matrix for the nxm matrix $\mathbf{X}$. Then,

$$\bar{s}^2 = [\mathbf{y}_j'\mathbf{y}_j/\bar{1}'\bar{1}], \; j=1 \text{ to } m, \qquad \text{(E-54)}$$

where $\bar{s}2$ is a column vector whose elements s_j^2 are the variances for each column X_j, in $\mathbf{X}$.

Covariance is a measure of the relationship between two variables, expressing how well (or poorly) two variables change in conjunction with one another. It is defined as the average of the sum of the products of the deviate scores for all objects, (i.e., the sum of the products across the rows of a deviate scores matrix). For the individual elements in the resulting covariance matrix in the multivariable case, covariance is given by:

$$s_{jk} = \sum_{i=1}^{n} y_{ij}y_{ik}/n \qquad \text{(E-55)}$$

where $\mathbf{Y}$ represents the deviate scores matrix of $\mathbf{X}$.

Note that the formula for variance is simply the case of covariance where the subscripts are equal. Thus, by forming the covariance matrix as the minor product moment of the data matrix expressed in deviate form:

$$\mathbf{S} = \mathbf{Y}'\mathbf{Y}/\mathbf{T}'\mathbf{T}, \qquad \text{(E-56)}$$

we actually have the variances of the individual variables, s_i^2 along the main diagonal, with the off-diagonal elements representing covariances, s_{ij}, between the corresponding variables, X_i, X_j, in $\mathbf{X}$. What we refer to here as simply the covariance matrix is sometimes called the variance-covariance matrix.

Standard deviation is the positive square root of variance. If a variance value is given by s_i^2, then the standard deviation is s_i.

Recall the length of a vector is given by the square root of its minor product moment, i.e., $|\bar{y}| = (\bar{y}'\bar{y})^{1/2}$. If $\bar{y}$ is a vector of deviate scores for a variable, X, then, looking back at the various equations, we see the standard deviation of X is proportional to the length of $\bar{y}$, and,

$$s = |\bar{y}|n^{-1/2}. \qquad \text{(E-57)}$$

Thus we are able to interpret standard deviation in terms of the lengths of vectors expressed in deviate form. (This should bring to mind the geometric interpretation of eigenvalues and eigenvectors).

Converting a variable into standardized scores means to express the observations within the variable as deviate scores, then to put

divergence in units of standard deviation. The standardized score for a particular element, x_i, in a vector, $\overline{X}$, is:

$$z_i = (x_i - E(\overline{X}))/s \qquad \text{(E-58)}$$

where: $E(\overline{X})$ is the expected value (mean) of $\overline{X}$, and,
s is the standard deviation of $\overline{X}$.

Standardized variables have a mean of zero, and variance and standard deviation of one. The lengths of the vectors of standardized variables is $n^{1/2}$, n the dimension of the vector.

The formula for standardizing all the variables in a data matrix is:

$$\mathbf{Z} = \mathbf{Y}\mathbf{D}^{1/2} \qquad \text{(E-59)}$$

where: $\mathbf{D}$ is a diagonal matrix of the diagonal elements of the covariance matrix, $\mathbf{S}$, and $\mathbf{Y}$ is the deviate scores matrix.

We previously discussed the (Pearson product-moment) correlation coefficient, defining it as the covariance between two normalized variables, i.e., the covariance between two variables after their individual means have been removed, and individual variances forced to equal one. We can also defined correlation, r, to be the ratio of the covariances of two variables to the product of their standard deviations, i.e, for two variables X_i, X_j, their correlation is given by:

$$r_{ij} = s_{ij}/s_i s_j \qquad \text{(E-60)}$$

For a matrix of standardized data, the correlation matrix is given by:

$$\mathcal{R} = \mathcal{Z}'\mathcal{Z}/T'T. \qquad \text{(E-61)}$$

The correlation matrix is square symmetric. It is the minor product moment of the standardized data matrix, with each element divided by the dimension of the variables, i.e., divided by the size of the sample. The interpretation of the correlation matrix with respect to factor analysis may be found in Appendix C.

Some of the mechanics of programming the above statistical procedures may be found in the programs for factor analysis produced for this thesis. These are listed in Appendix F.

Partial Fraction Expansion

The transfer function, H, (Appendix D) is a ratio of polynomials, P/Q. If this ratio as a proper fraction, i.e., a fraction whose numerator is of degree less than the degree of it's denominator, (we can find a proper fraction by dividing Q into P if required), H can be expressed as a sum of partial fractions. The denominators of the partial fractions are obtained by factoring Q into a product of linear and quadratic factors.

Any polynomial with real coefficients can be written as the product of linear and quadratic factors so that each of the factors has real coefficients, (though this is not always easy to do). When the n^{th} degree polynomial Q(x), in H(x) = P(x)/Q(x), has been factored into products of linear and quadratic factors:

$$Q(x) = C_0x^n + C_1x^{n-1} + \cdots + C^{n-1}x + C_n, \qquad \text{(E-62)}$$

we can assume without loss of generality that the coefficient $C_0 = 1$, as we can always divide the numerator and denominator of P(x)/Q(x) by C_0.

If the factors of Q(x) are all linear and none repeat, i.e.,

$$Q(x) = (x - a_1)(x - a_2) \cdots (x - a_n) \qquad \text{(E-64)}$$

where none of the a_i (E-64) are identical, then we can write:

$$\frac{P(x)}{Q(X)} \equiv \frac{A_1}{x - a_1} + \frac{A_2}{x - a_2} + \cdots + \frac{A_n}{x - a_n} \qquad \text{(E-65)}$$

where the A_i are constants to be determined. For example, say:

$$H(x) = \frac{x - 1}{x^3 + x^2 + 2x} \qquad \text{(E-66)}$$

factoring the denominator of H(x) (E-66), we get:

$$H(x) = \frac{x - 1}{x(x - 2)(x + 1)} \qquad \text{(E-67)}$$

and state:

$$\frac{x - 1}{x(x - 2)(x + 1)} \equiv \frac{A}{x} + \frac{B}{x - 2} + \frac{C}{x + 1} \qquad \text{(E-68)}$$

which simplifies to:

$$x - 1 \equiv A(x - 2)(X + 1) + Bx(x + 1) + Cx(x - 2). \qquad \text{(E-69)}$$

The coefficients A,B,C are determined by substituting in the root values from factoring the denominator of H(x) (E-67), x=0,2,-1, into the simplified equation (E-69), giving A = 1/2, B = 1/6, and C = -2/3.

If factoring of Q(x) results a repeated factor, then this factor must itself be expressed as a sum of partial fractions. If $(x - a_i)$ is a p-fold factor then it's corresponding partial fraction sum will be:

$$\frac{A_1}{(x - a_i)^p} + \frac{A_2}{(x - a_i)^{p-1}} + \cdots + \frac{A_{p-1}}{(x - a_i)^2} + \frac{A_p}{x - a_i} \qquad \text{(E-70)}$$

For:

$$H(x) = \frac{x^3 - 1}{x^2(x - 2)^3} \equiv \frac{A}{x^2} + \frac{B}{x} + \frac{C}{(x - 2)^3} + \frac{D}{(x - 2)^2} + \frac{E}{x - 2} \qquad \text{(E-71)}$$

the terms with coefficients A,B correspond to the expansion of the repeated factor x^2, and the terms with coefficients C,D,E correspond to

the expansion of repeated factor $(x - 2)^3$. Multiplying both sides of the identity (E-71) by the lowest common denominator gives:

$$x^3 - 1 \equiv A(x-2)^3 + Bx(x-2)^3 + Cx^2 + Dx^2(x-2) + Ex^2(x-2)^2 \quad \text{(E-72)}$$

expanding:

$$x^3 - 1 \equiv A(x^3-6x^2+12x-8) + Bx(x^3-6x^2+12x-8) + Cx^2 + Dx^3 - Dx^2 + Ex^2(x^2-4x+4) \quad \text{(E-73)}$$

grouping like powers of x:

$$x^3 - 1 \equiv (B+E)x^4 + (A-6B+D-4E)x^3 + (-6A+12B+C-2D+4E)x^2 + (12A-8B)x - 8A \quad \text{(E-74)}$$

Equating coefficients of like powers of x on both sides of the identity (E-74) gives a system of equations:

$$\begin{aligned} B + E &= 0 \\ A - 6B + D - 4E &= 1 \\ -6A + 12B + C - 2D + 4E &= 0 \\ 12A - 8B &= 0 \\ -8A &= -1 \end{aligned} \quad \text{(E-75)}$$

Solving this system (E-75) gives: A = 1/8, B = 3/16, C = 7/4, D = 5/4, E = -3/16.

Partial fraction expansion allows simplification of a transfer function to forms of use with simple z-transforms (Appendix D).

APPENDIX F

PROGRAM LISTINGS

Listings of some of the programs developed for this thesis follow.

ADDTEXT.BAS

Purpose: adds text headers to output files created using FACAN.M.

Language: GW-BASIC

```
 1 ' Program file: ADDTEXT.BAS -- Alan Swithenbank, March 1990
 2 '
 3 ' This program adds informational headers to data-files created
 4 ' using FACAN.M.
 5 '
10 PRINT : PRINT "Adding text headers to output files:" : PRINT
20 OPEN "SIMTYPE.DAT" FOR INPUT AS #1
30 INPUT #1,ST
40 CLOSE #1
50 SHELL"del simtype.dat"
60 IF ST=1 THEN 70 ELSE 100
70   F$ = "DEVSCOR.DAT"
80   T$ = "% deviate-scores-matrix"
90   GOTO 120
95 'else if
100  F$ = "NORMDVSC.DAT"
110  T$ = "% normalized deviate-scores-matrix"
115 'end if
120 GOSUB 2000
130 IF ST=1 THEN 140 ELSE 170
140   SHELL"del devscor.dat"
150   SHELL"rename temp.dat devscor.dat"
160   GOTO 190
165 'else if
170   SHELL"del normdvsc.dat"
180   SHELL"rename temp.dat normdvsc.dat"
185 'end if
190 F$ = "STDDEVS.DAT"
195 IF ST = 1 THEN T$="% column standard-deviations from deviate-scores-matrix"
200 IF ST = 2 THEN T$="% column std.-devs. of normalized deviate-scores-matrix"
210 GOSUB 2000
220 SHELL"del stddevs.dat"
230 SHELL"rename temp.dat stddevs.dat"
240 IF ST=1 THEN 250 ELSE 280
250   F$ = "COVMTX.DAT"
260   T$ = "% covariance-matrix of deviate-scores-matrix"
270   GOTO 300
275 'else if
280   F$ = "CORMTX.DAT"
290   T$ = "% correlation-matrix of data-matrix"
295 'end if
300 GOSUB 2000
310 IF ST=1 THEN 320 ELSE 350
320   SHELL"del covmtx.dat"
330   SHELL"rename temp.dat covmtx.dat"
340   GOTO 370
345 'else if
350   SHELL"del cormtx.dat"
360   SHELL"rename temp.dat cormtx.dat"
365 'end if
370 F$ = "SIMDIAG.DAT"
380 T$ = "% diagonal elements of chosen similarity-matrix"
390 GOSUB 2000
400 SHELL"del simdiag.dat"
410 SHELL"rename temp.dat simdiag.dat"
420 F$ = "EVAL.DAT"
430 T$ = "% eigenvalues of chosen similarity-matrix"
440 GOSUB 2000
450 SHELL"del eval.dat"
460 SHELL"rename temp.dat eval.dat"
470 F$ = "EVCT.DAT"
480 T$ = "% eigenvectors of chosen similaritry-matrix"
490 GOSUB 2000
500 SHELL"del evct.dat"
510 SHELL"rename temp.dat evct.dat"
```

```
520 F$ = "FACVAR.DAT"
530 T$ = "% variance explained by each retained factor"
540 GOSUB 2000
550 SHELL"del facvar.dat"
560 SHELL"rename temp.dat facvar.dat"
570 F$ = "NUMFACS.DAT"
580 T$ = "% number of factors retained"
590 GOSUB 2000
600 SHELL"del numfacs.dat"
610 SHELL"rename temp.dat numfacs.dat"
620 F$ = "VAREXPLD.DAT"
630 T$ = "% total variance explained by the retained factors"
640 GOSUB 2000
650 SHELL"del varexpld.dat"
660 SHELL"rename temp.dat varexpld.dat"
670 F$ = "LOADING.DAT"
680 T$ = "% factor-loadings from principle-components"
690 GOSUB 2000
700 SHELL"del loading.dat"
710 SHELL"rename temp.dat loading.dat"
720 F$ = "COMUNAL.DAT"
730 T$ = "% communalities from factor-loadings"
740 GOSUB 2000
750 SHELL"del comunal.dat"
760 SHELL"rename temp.dat comunal.dat"
770 F$ = "MATSIZ.DAT"
780 T$ = "% row and column size of loadings matrix"
790 GOSUB 2000
800 SHELL"del matsiz.dat"
810 SHELL"rename temp.dat matsiz.dat"
820 F$ = "TRNSFRM.DAT"
830 T$ = "% transformation matrix"
840 GOSUB 2000
850 SHELL"del trnsfrm.dat"
860 SHELL"rename temp.dat trnsfrm.dat"
870 F$ = "PRNCORVF.DAT"
880 T$ = "% correlation of variables with factors for principle-components"
890 GOSUB 2000
900 SHELL"del prncorvf.dat"
910 SHELL"rename temp.dat prncorvf.dat"
920 F$ = "PRNSCORE.DAT"
930 T$ = "% factor-scores from principle-components"
940 GOSUB 2000
950 SHELL"del prnscore.dat"
960 SHELL"rename temp.dat prnscore.dat"
970 F$ = "ROTSCORE.DAT"
980 T$ = "% rotated factor-scores"
990 GOSUB 2000
1000 SHELL"del rotscore.dat"
1010 SHELL"rename temp.dat rotscore.dat"
1020 F$ = "ROTLOAD.DAT"
1030 T$ = "% rotated factor-loadings"
1040 GOSUB 2000
1050 SHELL"del rotload.dat"
1060 SHELL"rename temp.dat rotload.dat"
1070 F$ = "ROTCORVF.DAT"
1080 T$ = "% correlation of variables with factors from rotation"
1090 GOSUB 2000
1100 SHELL"del rotcorvf.dat"
1110 SHELL"rename temp.dat rotcorvf.dat"
1120 F$ = "ROTEVCT.DAT"
1130 T$ = "% rotated retained eigenvectors"
1140 GOSUB 2000
1150 SHELL"del rotevct.dat"
```

```
1160 SHELL"rename temp.dat rotevct.dat"
1170 F$ = "ROTEVAL.DAT"
1180 T$ = "% rotated retained eigenvalues"
1190 GOSUB 2000
1200 SHELL"del roteval.dat"
1210 SHELL"rename temp.dat roteval.dat"
1220 F$ = "RFACVAR.DAT"
1230 T$ = "% percent variance explained by each rotated factor"
1240 GOSUB 2000
1250 SHELL"del rfacvar.dat"
1260 SHELL"rename temp.dat rfacvar.dat"
1270 F$ = "ROTCMNL.DAT"
1280 T$ = "% communalities from rotated factor-loadings"
1290 GOSUB 2000
1300 SHELL"del rotcmnl.dat"
1310 SHELL"rename temp.dat rotcmnl.dat"
1320 PRINT : PRINT "Done!"
1330 SYSTEM
1996 ' ************************ SUBROUTINES ************************
1997 '
1998 ' Open files and write out headers.
1999 '
2000 OPEN "TEMP.DAT" FOR OUTPUT AS #1
2020 OPEN F$ FOR INPUT AS #2
2030 PRINT F$
2040 PRINT #1,"% " + F$
2050 PRINT #1, T$
2060 IF ST = 1 THEN PRINT #1,"(analysis using covariance-matrix)"
2070 IF ST = 2 THEN PRINT #1,"(analysis using correlation-matrix)"
2080 PRINT #1,
2090 GOSUB 3000
2100 RETURN
2996 '
2997 ' Read in data-file records as line inputs and write
2998 ' them out below added header in temporary file.
2999 '
3000 IF EOF(2) THEN 3040
3010 LINE INPUT #2, A$
3020 PRINT #1, A$
3030 GOTO 3000
3040 CLOSE
3050 RETURN
```

AUTOCOVCOR

Purpose: calculates autocovariance and autocorrelation

Language: AmigaBASIC

```
' AutoCovCor

' This program calculates autocovariance and autocorrelation for an
' input-variable file for lags 0 to n/4, where n is the length of the
' input file.

  DIM X(2000)

  LOCATE 10,10
  INPUT "Enter input variable filename: ",InpFil$

  OPEN InpFil$ FOR INPUT AS #1

  LOCATE 10,10
  PRINT SPACE$(60)
  LOCATE 10,10
  PRINT "Reading ";InpFil$

        n = 0
     SumX = 0
  SumXSqr = 0

  WHILE NOT(EOF(1))
    n=n+1
    INPUT #1,X(n)
    SumX = SumX + X(n)
    SumXSqr = SumXSqr + X(n)*X(n)
  WEND

  VarX = (n*SumXSqr - SumX^2)/(n*(n-1))

  CLOSE #1

  OPEN InpFil$+".autocovcor" FOR OUTPUT AS #1

  maxlag = n\4
     Tau = 0

LagLoop:

  IF Tau > maxlag THEN EndLagLoop

  LOCATE 10,10
  PRINT SPACE$(60)
  LOCATE 10,10
  PRINT "Calculating auto-functions for lag";Tau

    SumXtTau = 0
  SumXtXtTau = 0

  FOR t=1+Tau TO n
    SumXtTau = SumXtTau + X(t-Tau)
    SumXtXtTau = SumXtXtTau + X(t)*X(t-Tau)
  NEXT t

  Autocov = SumXtXtTau/(n-Tau) - (SumX/n)*(SumXtTau/(n-Tau))
  Autocor = Autocov/VarX

  PRINT #1,Autocov;Autocor

  Tau = Tau + 1

  GOTO LagLoop

EndLagLoop:
```

```
CLOSE #1

LOCATE 12,20
PRINT "All Done!"
LOCATE 14,10
PRINT "Output results are in ";InpFil$+".autocovcor"

END
```

CONSAL.F

Purpose: specific conductance in microseimens at 25°C —> salinity

Language: FORTRAN (Microsoft V4.01)

```
c**** ******************************************************************
c                          CONSAL.F
c                       Alan Swithenbank
c                          April 1990
c**** ******************************************************************
c This program converts a keyboard entered specific conductance into a
c PSS-78 salinity value using the UNESCO/SCOR approved algorithm (Fofonoff
c and Millard 1983) when the parameters have been precalculated for T=25, P=0.
c
c Paramaterizing the PSS-78 equation:
c
c   Rp at P=0 is 1.000000,  rt at T = 25 is 1.236537
c   leading delS multiplier at T = 25 is 8.605852
c
c**** ******************************************************************
      PROGRAM SALCON
c**** ******************************************************************

c variable declarations:

      CHARACTER*80 INPSTR

      INTEGER*4 ERRNUM

      REAL*8 INPEC, OUTSAL

c function declarations:

      CHARACTER*80 NUMSTR

      REAL*8 CONSAL
      REAL*8 STRNUM

c**** ******************************************************************

      PRINT*, 'This program converts specific conductance to salintiy'
      PRINT*, 'using the PSS-78 modified for use at 25 degrees C only.'
100   PRINT*
      PRINT*, 'Enter value for conductivity to salinity conversion:'
      PRINT*, '(QUIT to stop)'
      PRINT*

      READ(*,200) INPSTR
200   FORMAT(A)

      IF (     (INDEX(INPSTR,'quit') .NE. 0)
     &    .OR. (INDEX(INPSTR,'QUIT') .NE. 0)
     &   ) GOTO 2000

      PRINT*

c take string and convert to REAL*8:

      INPEC = STRNUM(INPSTR)

c convert conductance to salinity and output:

      WRITE(*,210) CONSAL(INPEC)
210   FORMAT(' Converted salinity is: ',F6.3)

      GOTO 100

c**** ******************************************************************

2000  PRINT*
```

```
      PRINT*, 'Bye Bye !'
      PRINT*

      END

c**** ******************************************************************
c**** ******************************************************************
c**** ******************************************************************
c**** ******************************************************************
c                    SUBROUTINE DEFINITONS
c**** ******************************************************************
c**** ******************************************************************
c**** ******************************************************************
c**** ******************************************************************
      FUNCTION CONSAL(SPCOND)
c**** ******************************************************************
c**** ******************************************************************
c This function converts specific conductance to salinty using
c simplified form of PSS-78 for use at 25 degrees C only.
c**** ******************************************************************

      REAL*8 K1, K2, K3, K4, K5, K6, R25, HALF, THROV2, TWO, FIVOV2
      REAL*8 C3525, CONSAL, SPCOND

c simplified PSS-78 coefficents:

      PARAMETER ( K1 =  0.0123,
     &            K2 = -0.2174,
     &            K3 = 25.3283,
     &            K4 = 13.7714,
     &            K5 = -6.4788,
     &            K6 =  2.5842 )

c conductivity for S = 35.000 water at 25 degrees C:

      PARAMETER ( C3525 = 53088 )

c double precision exponents:

      PARAMETER ( HALF   = 0.5,
     &            THROV2 = 1.5,
     &            TWO    = 2.0,
     &            FIVOV2 = 2.5 )

      R25 = SPCOND/C3525

      CONSAL = K1 +
     &         K2*(R25**HALF) +
     &         K3*R25 +
     &         K4*(R25**THROV2) +
     &         K5*(R25**TWO) +
     &         K6*(R25**FIVOV2)


      IF (CONSAL .LT. 2.0) CONSAL = 0.0

      RETURN
      END
```

```
c**** ******************************************************************
c**** ******************************************************************
      FUNCTION STRNUM(INPSTR)
c**** ******************************************************************
c**** ******************************************************************
c This function converts a valid string representation of a number into
c a REAL*8 value for return to the calling routine.  This function error
c traps invalid inputs automatically, but the calling routine must be able
c to handle errors on it's own.  The return value on a error input is a
c small number (-1.0e-100).  Through a READ the character string INPSTR is
c opened as an internal file and, using the F30.0 FORMAT line 1, converted
c from a character string to a REAL*8 numeric value.  The F30.0 specification
c is overridden when, for example, INPSTR is 123.456e-7, by the forced
c format of the string itself.  If INPSTR is not a valid numeric type
c representation, the conversion bombs and is trapped as an I/O error.
c**** ******************************************************************

      REAL*8 STRNUM
      CHARACTER*(*) INPSTR

100   READ ( UNIT = INPSTR,
     &       FMT = 1,
     &       ERR = 1000 ) STRNUM

1     FORMAT(F30.0)

      RETURN

1000  INPSTR = '-1.0E-100'
      GOTO 100

      END

c**** ******************************************************************
c**** ******************************************************************
      FUNCTION NUMSTR(NUMBER)
c**** ******************************************************************
c**** ******************************************************************
c This function takes a REAL*8 value and converts it to a string.
c The value is written to a string via an internal reference in a
c WRITE statement.  When numbers fall outside + or - E+010 range
c the string is automatically output in scientific notation.  Within
c that range, floating point format is used.  When the value input has
c no decimal point, then the decimal is left off the string.
c**** ******************************************************************

      CHARACTER*80 NUMSTR
      REAL*8 NUMBER

      INTEGER*4 PLACE

c make the conversion to a string:

      IF (NUMBER .EQ. 0) THEN
        NUMSTR = '    0'
        GOTO 20
      ELSEIF (      (ABS(NUMBER) .GT. 1.0E+10)
     &         .OR. (ABS(NUMBER) .LT. 1.0E-10)
     &       ) THEN
                    WRITE (UNIT = NUMSTR,
     &                      FMT = 2 ) NUMBER
      ELSE
                    WRITE (UNIT = NUMSTR,
     &                      FMT = 1 ) NUMBER
      ENDIF
```

```
1     FORMAT(F30.15)
2     FORMAT(E15.6)

c remove leading blanks:

10    PLACE = INDEX(NUMSTR,' ')
      IF (PLACE .NE. 1) GOTO 20
      NUMSTR = NUMSTR(2: )
      GOTO 10

20    CONTINUE

c remove trailing decimal:

      IF (       (INT(NUMBER) .EQ. NUMBER)
     &     .AND. (INDEX(NUMSTR,'.') .NE. 0)
     &    ) NUMSTR = NUMSTR(1:INDEX(NUMSTR,'.')-1)

      RETURN

      END

c**** **********************************************************************
c**** **********************************************************************
```

CROSSCOR

Purpose: calculates cross-correlations and significance-values

Language: AmigaBASIC

```
' CrossCor

' This program calculates cross-correlation and corresponding significance
' test values for autocorrelation of two selected data-column variables from
' a multicolumn input-file for lags 0 to n/4, where n is the column-size of
' the input file.  The variables may be up to 2000 records long, and the
' input file may contain up to 10 variables.

  CLEAR ,25000
  CLEAR ,125000&

  DIM InpDat(10,2000),C(10),ReadCol(10),t(500),r(500)

  TRUE = -1
  FALSE = 0

  MsgCol = 10
  MsgRow = 10
  ErrCol =  5
  ErrRow =  3

  ON ERROR GOTO ErrorTraps  ' Error trapping is used intentionally for
                            ' stopping data input with INPUT# -- do input
                            ' this way for increased speed (no string
                            ' parsing and character handling required for
                            ' variable number of data columns in input files).
                            ' EOF() does't work with INPUT# like it does with
                            ' LINE INPUT# (see program INPUT#_EOF_Error_Fix)

EnterInputName:

  LOCATE MsgRow,1
  PRINT SPACE$(78)
  LOCATE       ,1
  PRINT SPACE$(78)
  LOCATE MsgRow,MsgCol
  PRINT "Enter file containing variables for"
  LOCATE       ,MsgCol
  INPUT "cross-correlation: ",InpFil$
  LOCATE ErrRow,1
  PRINT SPACE$(78)

  CALL OldFile(InpFil$,FileOld) ' The name FileOld is NOT optional in
                                ' OldFile! -- if a file does not exist
                                ' then error-trapping takes over and the
                                ' OldFile() subroutine does not return.
                                ' FileOld is set FALSE followed by a RESUME
                                ' NEXT in ErrorTraps anytime a file is
                                ' is not found.  RESUME NEXT considers CALL
                                ' to be the error-line, not the OPEN in
                                ' the subroutine.  FileOld is returned as
                                ' TRUE by OldFile() if the file exists.

  IF NOT FileOld THEN
    CALL NotFound(InpFil$,ErrCol,ErrRow)
    GOTO EnterInputName
  END IF


'****** START FASTER AmigaBASIC DATAFILE READER ******:

ReadInputFile:

  CLS
```

```
  CALL BreakOut(InpFil$,FirstLine$,Col,ReadNum,ReadCol())

  LOCATE 18,5
  PRINT "Reading in selected columns from ";InpFil$

  Row = 0

  OPEN InpFil$ FOR INPUT AS #1 LEN = 16384 ' large buffer for faster I/O

  WHILE NOT EOF(1)

' read column values from current row:

    IF Col= 1 THEN INPUT#1,C(1)
    IF Col= 2 THEN INPUT#1,C(1),C(2)
    IF Col= 3 THEN INPUT#1,C(1),C(2),C(3)
    IF Col= 4 THEN INPUT#1,C(1),C(2),C(3),C(4)
    IF Col= 5 THEN INPUT#1,C(1),C(2),C(3),C(4),C(5)
    IF Col= 6 THEN INPUT#1,C(1),C(2),C(3),C(4),C(5),C(6)
    IF Col= 7 THEN INPUT#1,C(1),C(2),C(3),C(4),C(5),C(6),C(7)
    IF Col= 8 THEN INPUT#1,C(1),C(2),C(3),C(4),C(5),C(6),C(7),C(8)
    IF Col= 9 THEN INPUT#1,C(1),C(2),C(3),C(4),C(5),C(6),C(7),C(8),C(9)
    IF Col=10 THEN INPUT#1,C(1),C(2),C(3),C(4),C(5),C(6),C(7),C(8),C(9),C(10)

    Row = Row + 1

    LOCATE 19,25
    PRINT Row

' place selected column values into InpDat():

    FOR i=1 TO ReadNum
      InpDat(i,Row) = C(ReadCol(i)) ' ReadCol() holds selected col indices
    NEXT i                          ' ReadNum is the number of cols to read

  WEND

FastLoadOver:

  CLOSE #1

'****** END FASTER AmigaBASIC DATAFILE READER ******:

  IF ReadNum=1 THEN
    BEEP
    COLOR 3,2
    LOCATE 12,5
    PRINT "You need more than one variable for a cross-correlation!"
    COLOR 1,0
    dly& = TIMER
    WHILE TIMER < dly& + 6
    WEND
    CLS
    GOTO EnterInputName
  END IF

EnterOutputName:

  CLS

  LOCATE MsgRow,1
  PRINT SPACE$(78)
  LOCATE       ,1
  PRINT SPACE$(78)
  LOCATE MsgRow+1,MsgCol
```

```
  PRINT "(Hit return to bailout.)"
  LOCATE MsgRow,MsgCol
  INPUT "Enter file for output : ",OutFil$
  IF OutFil$ = "" THEN GOTO BailOut
  LOCATE ErrRow,1
  PRINT SPACE$(78)
  CALL OldFile(OutFil$,FileOld)

  IF FileOld THEN
    CLS
    BEEP
    LOCATE ErrRow,ErrCol
    COLOR 3,0
    PRINT OutFil$;" aleady exists!"
    COLOR 1,0
    LOCATE ErrRow+2,ErrCol
    INPUT "Do you wish to overwrite";Y$
    IF INSTR(UCASE$(Y$),"Y") = 0 THEN
      LOCATE ErrRow+2,1
      PRINT SPACE$(78)
      GOTO EnterOutputName
    END IF
  END IF

StartCross:

' Calculations are one-way only -- slide second-variable to right past
' first-variable -- scan: first-variable from 1+tau to n vs.
' second-variable from 1 to n-tau, tau = 0 to n\4.  The number of overlap
' points is then n-tau.  If you want to see the cross-correlogram the other
' way then swap variables at input.

  CLS

  LOCATE 2,5
  PRINT "There were";Row;"data rows input from ";InpFil$
  LOCATE 4,5
  PRINT InpFil$;" has";Col;"data columns."
  LOCATE 6,5
  PRINT "First row looks like:"
  LOCATE 8,5
  PRINT FirstLine$
  LOCATE 10,5
  IF ReadNum = 1 THEN
    PRINT "You selected column";
  ELSE
    PRINT "You selected columns";
  END IF
  FOR i=1 TO ReadNum
    PRINT ReadCol(i);
  NEXT i
  PRINT "for input."

SelectColumn1:

  LOCATE 13,1 : PRINT SPACE$(65)
  LOCATE 13,1 : INPUT "Enter first column index for input: ",Ci$

  IF Ci$ = "" THEN GOTO BailOut

  FOR i=1 TO Col
    IF VAL(Ci$) = ReadCol(i) THEN
      LOCATE 16,5 : PRINT SPACE$(65)
      C1 = ReadCol(i)                  ' Column indicies use indirection!
      V1 = i
```

```
      GOTO SelectColumn2
    END IF
  NEXT i

  LOCATE 16,5 : PRINT "Input Error! -- try again"
  GOTO SelectColumn1

SelectColumn2:

  LOCATE 13,1 : PRINT SPACE$(65)
  LOCATE 13,1 : INPUT "Enter second column index for input: ",Ci$

  IF Ci$="" THEN GOTO SelectColumn2

  FOR i=1 TO Col
    IF VAL(Ci$) = ReadCol(i) THEN
      LOCATE 16,5 : PRINT SPACE$(65)
      C2 = ReadCol(i)
      V2 = i
      GOTO DoubleCheck
    END IF
  NEXT i

  LOCATE 16,5 : PRINT "Input Error! -- try again"
  GOTO SelectColumn2

DoubleCheck:

  IF V2=V1 THEN
    LOCATE 16,5
    PRINT "There are better ways to compare a variable to itself!"
    GOTO SelectColumn2
  END IF

  CLS

  LOCATE 10,10
  PRINT "Performing cross-correlation for variables";C1;"and";C2

  maxlag = Row\4
     Tau = 0
       m = 1

  FOR Tau = 0 TO maxlag

    Lapn = Row - Tau

    LOCATE 12,10
    PRINT SPACE$(60)
    LOCATE 12,10
    PRINT "Calculating cross-correlation and significance for lag";Tau

    Sum1 = 0
    Sum2 = 0
    Sum3 = 0
    Sum4 = 0
    Sum5 = 0
    Sum6 = 0

    FOR i = 1 TO Lapn
      Sum1 = Sum1 +  InpDat(V1,Tau+i)*InpDat(V2,i)
      Sum3 = Sum3 + (InpDat(V1,Tau+i))^2
      Sum4 = Sum4 +  InpDat(V1,Tau+i)
      Sum5 = Sum5 + (InpDat(V2,Tau+i))^2
      Sum6 = Sum6 +  InpDat(V2,Tau+i)
```

```
    NEXT i

    Sum2 = Sum1

    denom = ((Lapn*Sum3 - Sum4^2)*(Lapn*Sum5 - Sum6^2))^(1/2)

    r(Tau+1) = (Lapn*Sum1 - Sum2)/denom
    IF ABS(r(Tau+1))>= 1 THEN r(Tau+1)=SGN(r(Tau+1))*.99999 ' round off error

    t(Tau+1) = r(Tau+1)*((Lapn-2)/(1-r(Tau+1)^2))^(1/2)      ' cause t blow-up

  NEXT Tau

WriteOut:

  OPEN OutFil$ FOR OUTPUT AS #1

  LOCATE 12,1
  PRINT SPACE$(78)
  LOCATE 12,5
  PRINT "Writing results to ";OutFil$

  FOR Tau=0 TO maxlag
      PRINT #1,Tau;r(Tau+1);t(Tau+1)
  NEXT Tau

  CLOSE #1

  GOTO EnterOutputName

BailOut:

  CLS
  ERASE InpDat,C,ReadCol,t,r
  LOCATE MsgRow,MsgCol
  PRINT "All Done!"
  CLEAR ,25000

END

'*******************************************************
'*******************************************************
'**                                                   **
'**                    ERROR TRAPS                    **
'**                                                   **
'*******************************************************
'*******************************************************

ErrorTraps:

IF ERR = 62 THEN      '  input past end error (traps the fast-load end)
  RESUME FastLoadOver
ELSEIF ERR = 53 THEN  ' file not found
  FileOld = FALSE
  RESUME NEXT
ELSEIF ERR = 55 THEN ' file already open
  CLOSE
  CLS
  BEEP
  LOCATE 3,5
  COLOR 3,0
  PRINT "You tried to use an open input for output -- start over!"
  COLOR 1,0
  RESUME EnterInputName
ELSE
```

```
  ON ERROR GOTO 0
END IF

'*****************************************************
'*****************************************************
'**                                                 **
'**                   SUBPROGRAMS                   **
'**                                                 **
'*****************************************************
'*****************************************************

SUB OldFile(FileName$,FileErr) STATIC

' Sets FileErr to FALSE then checks if file is present by
' attempting to open for input.  If present, then opens,
' closes, exits.   If not, get error.  Error trap sets
' FileErr FALSE, then RESUME NEXT in the error trap returs
' to the line after the call.  Return of FileErr FALSE means
' that the file does not exist, so it is safe to open
' for output.  FileErr = TRUE means that the file exists so
' check whats going on. The name 'FileErr' in this subroutine
' is not optional due to use of error-trapping.

  SHARED TRUE,FALSE

  FileErr = TRUE

  OPEN FileName$ FOR INPUT AS #99
  CLOSE #99

END SUB

'*****************************************************

SUB NotFound(InpFile$,MsgCol,MsgRow) STATIC

  CLS
  BEEP
  LOCATE MsgRow,MsgCol
  COLOR 3,0
  PRINT "I could not find ";InpFile$
  COLOR 1,0

END SUB

'*****************************************************

SUB BreakOut(DataFile$,FirstLine$,Col,ReadNum,ReadCol()) STATIC

' This subprogram determines the number of data values in FirstLine$,
' (which is the number of data columns in DataFile$), returning this
' value in Col.  Specific data columns may be selected for input.
' The number of selected columns is returned in ReadNum, and the indices
' for the selected columns are returned in ReadCol().

  SHARED TRUE,FALSE

  OPEN DataFile$ FOR INPUT AS #99
    LINE INPUT#99,FirstLine$
  CLOSE #99

  Col = 0
    i = 1

ParseLoop:
```

```
  WHILE i <= LEN(FirstLine$)
    i=i+1
    IF i>LEN(FirstLine$) THEN GOTO ParseOver
    IF MID$(FirstLine$,i,1) <> " " THEN
      Col = Col+1

SkipCol:

      IF MID$(FirstLine$,i,1) <> " " THEN
        i=i+1
        IF i>LEN(FirstLine$) THEN GOTO ParseOver
        GOTO SkipCol
      END IF

    END IF
  WEND

ParseOver:

  LOCATE 9,1
  PRINT DataFile$;" has";Col;"columns.  It's first row is:"
  LOCATE 11,5
  PRINT FirstLine$

  ReadNum = 0
  GotOne = FALSE

SelectColumn:

  LOCATE 13,1 : PRINT SPACE$(65)
  LOCATE 14,1 : PRINT "(just hit enter to quit)"
  LOCATE 13,1 : INPUT "Enter column index for input: ",Ci$

  IF Ci$ = "" AND GotOne = TRUE THEN GOTO SelectOver

  IF VAL(Ci$)<1 OR VAL(Ci$)>Col THEN
    LOCATE 16,5 : PRINT "Input Error! -- try again"
    GOTO SelectColumn
  END IF

  ReadNum = ReadNum + 1

  IF ReadNum > Col THEN
    LOCATE 16,5 : PRINT "You can't read in more columns than there are!"
    ReadNum = ReadNum - 1
    TooMany = TRUE
    GOTO SelectOver
  END IF

  IF ReadNum > 10 THEN
    LOCATE 16,5 : PRINT "Only 10 columns allowed for input selection!"
    ReadNum = ReadNum - 1
    GOTO SelectOver
  END IF

  ReadCol(ReadNum) = VAL(Ci$)

  GotOne = TRUE

  LOCATE 16,5 : PRINT SPACE$(65)

  GOTO SelectColumn

SelectOver:
```

```
  LOCATE 13,1 : PRINT SPACE$(65)
  LOCATE 14,1 : PRINT SPACE$(65)
  IF TooMany = FALSE THEN
    LOCATE 16,5
    PRINT SPACE$(65)
  END IF

END SUB

'*****************************************************
```

FACAN.M

Purpose: Main factor-analysis routine, calls others

Language: PC-MatLab .M script file

```
function f = facan(X)
%
% facan(X) performs R-mode factor analysis on a data matrix X.  This is not
% "true" factor analysis -- it uses prncmp(X) to get the principle components
% then uses varimax.bas to do a normalized varimax rotation with the loadings
% derived from the principle components analysis.  The output is in the form
% of ASCII flat files. (Alan Swithenbank, January 1988)
%
% modified June 1988 for variance explained per retained factor output (AMS)
%
% modified July 1988 to include communality output (AMS)
%
% modified September 1989 to use direct ASCII import/export under MatLab
% version 3.5 upgrade (AMS)
%
% modified March 1990 to use similarity-matrix type information from
% prncmp(X) return-string while generating outputs (AMS)
%
% modified March 1990 to take inv(diag(std-devs)) in prncmp(X) call (AMS)
%
% modified March 1990 to correct the sum of rotated eigenvalues so the
% proper variance explained by the rotated factors is calculated (AMS)
%
% ------------------------------------------------------------------------
%
  fprintf('\nPerforming Principle Components Analysis\n')
%
  [cc,cvf,fload,score,eval,evct,sim,cd,n,pv,sv,devsc,sd,isd,inveig] = prncmp(X);
%
  simmtx=sim; % give more descriptive variable names
  simdiag=cd;
  corvarf=cvf;
  clear sim cd cvf % and keep as much free memory as possible
%
  [r,c]=size(eval);
  facvar = [];
  count=0;
  while n>count % calculate percent variance explained by retained factors
    count=count+1; % count to go across columns of eval
    v=100*eval(r:r,count)/sv;  % variance = eigenvalue/(sum of eigenvalues)
    facvar=[facvar, v]; % build up value matrix by concatenation
    r=r-1; % eval is diagonal from bottom to top left to right, so go up rows
  end
%
%
  fprintf('\nfiling components data\n')
%
  S = size(fload);
  save matsiz.dat S /ascii
  clear S % clear everything that won't be used later
  save prncorvf.dat corvarf /ascii
  clear corvarf
  save loading.dat fload /ascii
  clear fload
  save prnscore.dat score /ascii
  save eval.dat eval /ascii
  clear eval
  save evct.dat evct /ascii
  clear evct
  save facvar.dat facvar /ascii

  if cc == 1
    save covmtx.dat simmtx /ascii
  end
  if cc == 2
```

```
    save cormtx.dat simmtx /ascii
  end
  clear simmtx
%
% save similarity-matrix diagonal even if used covariance-matrix because
% it is used in VARIMAX.BAS regardless
%
  save simdiag.dat simdiag /ascii
  clear simdiag
  save numfacs.dat n /ascii
  save varexpld.dat pv /ascii
  clear pv
  if cc == 1
    save devscor.dat devsc /ascii
  end
  if cc == 2 % deviate-scores have been normalized if correlation-matix used
    save normdvsc.dat devsc /ascii
  end
  clear devsc
  save stddevs.dat sd /ascii
  clear sd
%
 !cls
%
 !rbasic c:\aland\thesis\prog\varimax.bas
%
% rbasic is a .BAT file that uses the /f8 flag on GWBASIC
%
  fprintf('\nbringing in transform matrix\n')
%
  load trnsfrm.dat
  TM = trnsfrm;
  clear trnsfrm
%
 !cls
%
  fprintf('\ncalculating rotated scores\n')
%
  rotscore = score*inv(TM');
%
  fprintf('\nbringing in rotated loadings and eigenvectors\n')
%
  load rotload.dat
  load rotevct.dat
  rotevct  % send to screen
  rotload
%
 !cls
%
  fprintf('\ncalculating variable vs. rotated-factor correlations\n')
%
  rotcorvf = isd*rotload
%
  fprintf('\ncalculating rotated retained eigenvalues\n')
%
  roteval = rotevct\rotload;  % this is sqrt(rotated eigenvalues)
  roteval = roteval.*roteval  % so square for final result
%
% The total-variance explained by rotated factors = total-variance
% explained by the original factors.  But if the total variance explained
% by the retained original-factors is less than 100 percent, the sum of the
% rotated factors must be adjusted for the calculation of variance explained by
% the individual rotated factors. This is because just the retained
% original-factors are rotated and not all the original-factors. We must
% calculate X in sumrot/(sumrot+X) = sumret/sumtot. Where sumrot is the sum
```

```
% of the rotated eigenvalues, sumret is the sum of the retained original
% eigenvalues, and sumtot is the sum of all the original eigenvalues.
%
  varexp = sum(facvar)/100; % proportion of variance from retained originals
  clear facvar
  sumrot = sum(diag(roteval));
  X = sumrot/varexp - sumrot;
  totrot = sumrot + X;
%
% calculate percent variance explained by rotated factors
%
  r=1;
  rfacvar = [];
  count=0;
  while n>count
    count=count+1; % count to go across columns of roteval
    v=100*roteval(r:r,count)/totrot;
    rfacvar=[rfacvar, v]; % build up value-matrix by concatenation
    r=r+1; % roteval is diagonal top to bottom, right to left, so go down rows
  end
%
  fprintf('\nfiling rotation information\n')
%
  save rotscore.dat rotscore /ascii
  save rotload.dat rotload /ascii
  save rotcorvf.dat rotcorvf /ascii
  save roteval.dat roteval /ascii
  save rfacvar.dat rfacvar /ascii
  save simtype.dat cc /ascii
%
 !rbasic c:\aland\thesis\prog\addtext.bas
%
 !cls
%
  fprintf('\nthe results are in the following files (with .DAT extension):\n')
  if cc == 1
    fprintf('\n DEVSCOR = data deviate scores matrix\n')
  fprintf(' STDDEVS = column standard-deviations from deviate-scores-matrix\n')
    fprintf('  COVMTX = covariance-matrix of deviate-scores-matrix\n')
  end
  if cc == 2
    fprintf('\nNORMDVSC = normalized deviate-scores-matrix\n')
  fprintf(' STDDEVS = column std.-devs. of normalized deviate-scores-matrix\n')
    fprintf('  CORMTX = correlation-matrix of data-matrix\n')
  end
  fprintf(' SIMDIAG = diagonal elements of chosen similarity-matrix\n')
  fprintf('    EVAL = eigenvalues of chosen similarity-matrix\n')
  fprintf('    EVCT = eigenvectors of chosen similaritry-matrix\n')
  fprintf('  FACVAR = variance explained by each retained factor\n')
  fprintf(' NUMFACS = number of factors retained\n')
  fprintf('VAREXPLD = total variance explained by the retained factors\n')
  fprintf(' LOADING = factor-loadings from principle-components\n')
  fprintf(' COMUNAL = communalities from factor-loadings\n')
  fprintf('  MATSIZ = row and column size of loadings matrix\n')
  fprintf(' TRNSFRM = transformation matrix\n')
  fprintf('PRNCORVF = correlation of vars. with factors for princ. comps.\n')
  fprintf('PRNSCORE = factor-scores from principle-components\n')
  fprintf('ROTSCORE = rotated factor-scores\n')
  fprintf(' ROTLOAD = rotated factor-loadings\n')
  fprintf('ROTCORVF = correlation of variables with factors from rotation\n')
  fprintf(' ROTEVCT = rotated retained eigenvectors\n')
  fprintf(' ROTEVAL = rotated retained eigenvalues\n')
  fprintf(' RFACVAR = percent variance explained by each rotated factor\n')
  fprintf(' ROTCMNL = communalities from rotated factor loadings\n')
```

FFTSPEC.M

Purpose: produces FFT power spectra and periodograms

Language: PC-MatLab .M script file

```
function [P,f,t] = fftspec(X,fs)
%
%  fftspec automates the use of the PC-MatLab fft function to get out
%  power spectrum plots for the variable X with a sampling frequency fs.
%  For the Pier 24 data I want to plot in cycles/hour, so I use fs=2,
%  as they are 30 minute samples, which means 2/hour.
%  The outputs are:
%
%  P = the power spectral estimates for X
%  f = the frequencies (to Nyquist)
%  t = the periods related to f (except infintity)
%
%  spectrum plots vs. frequency and vs. period are automatically produced
%
%  all the requirements for fft hold (Alan Swithenbank, September 1988)
%
% ------------------------------------------------------------------------
%
  Y = fft(X);
  P = Y.*conj(Y);
  n = length(Y);
  m = n/2; % note: only get 1/2 the bins, because rest is symmetric
  f = fs*(0:m-1)/n;
  t = ones(1:m-1)./f(1,2:m); % note: period of infinity is skipped
  clg
  subplot(211)
  plot(f,P(1:m))
  title('POWER SPECTRUM PLOT TO NYQUIST BY FREQUENCY')
  ylabel('power')
  xlabel('frequency')
  subplot(212)
  plot(t,P(2:m))
  title('POWER SPECTRUM PLOT BY PERIOD')
  ylabel('power')
  xlabel('period')
```

FOURIER.FILTER

Purpose: programmable Fourier-Transform filter

Language: AmigaBASIC (intended for use with AC-BASIC compiler)

```
' Fourier.Filter -- Alan Swithenbank, June 1990

' A Fourier-Transform filter.  Prepares and Fourier-filters a time-series
' to remove tidal-components.  Maximum raw-series length is 8192 records.
' If the number of input-records is not a power-of-2, the input data are
' zero-padded to a power-of-2 length.  The input-data may optionally be
' detrended, have the mean removed, and have no taper or a cosine-taper
' applied.  The transform is windowed in the frequency-domain to eliminate
' high-frequency information in the spectrum, with the stop and pass periods
' selectable by the user.  The mean may be applied to the filtered data if
' it was removed before processing.

' This program is based on the FORTRAN program FTF.F produced by
' Dr. Jia Wang during his stay with the United States Geological
' Survey Deer Creek office hydrodynamics-group in 1989, under
' Dr. Ralph Cheng.

' A full-size window is explicitly opened by the interpreter.
' The ' AC-BASIC compiler does this automatically and the
' window-opening is $IGNOREd by the compiler.

'******************************************************
'******************************************************
'**                                                  **
'**                 INITIALIZATIONS                  **
'**                                                  **
'******************************************************
'******************************************************

  ON ERROR GOTO ErrorTraps

Restart:

  DIM SHARED DAT(16384&)

  MsgR = 10          ' message/prompt row and column
  MsgC = 10

  ErrR =  3          ' error-message row and column
  ErrC =  5

  LeftCol = 34       ' control-gadget area upper-left corner
   TopRow = MsgR+4

   TRUE = -1         ' this is MicroSoft's idea, not mine
  FALSE =  0

  Compiled = TRUE    ' run-time flagging of interpreter/compiler operation

'$IGNORE ON

  SCREEN 1,640,400,2,4
  WINDOW 2,"Fourier Filter",(1,1)-(617,385),22,1
  Compiled = FALSE

'$IGNORE OFF

InputName:

  CALL ClearRows(MsgR,1!)
  LOCATE MsgR,MsgC
  COLOR 3,1
  PRINT "Enter filename of data for Fourier filtering:";
  COLOR 1,0
  INPUT" ",InpFil$
```

```
 CALL OldFile(InpFil$,FileOld) ' The name 'FileOld' is NOT optional in
                               ' OldFile()! -- if a file does not exist
                               ' then error-trapping takes over and the
                               ' OldFile() subroutine does not return.
                               ' FileOld is set FALSE followed by a RESUME
                               ' NEXT in ErrorTraps anytime a file is
                               ' is not found.  RESUME NEXT considers CALL
                               ' to be the error-line, not the OPEN in
                               ' the subroutine.  FileOld is returned as
                               ' TRUE by OldFile() if the file exists.

 IF NOT FileOld THEN

   ' Error-trapping closes custom-windows under interpreter, so reopen
   ' if this is not a compiled program.

   IF NOT Compiled THEN WINDOW 2,"Fourier Filter",(1,1)-(617,385),22,1
   ErrMsg$ = "I could not find "+InpFil$
   CALL ErrorMsg(ErrMsg$,ErrC,ErrR,TRUE)
   GOTO InputName
 END IF

 CALL ClearRows(ErrR,1!)
 CALL ClearRows(MsgR,3!)

EnterInterval:

   P$ = "Sample-interval for "+InpFil$+" (minutes):"
   MC = MsgC
   MR = MsgR
  InF = TRUE  ' integer inputs
   DF = FALSE ' no default
   DV = 0
  MMF = TRUE  ' max and min range active (dummy since using slider)
  MME = TRUE  ' exclude exact max and min (dummy since using slider)
  min = 1
  max = 180   ' 180 minutes = 3 hours maximum sample-interval
   EC = ErrC
   ER = ErrR
   SF = TRUE  ' use slider -- since integers can eliminate 0 with min = 1
   SC = MC
   SS = 2     ' 2-pixels per integer-increment

  CALL NumericInput(IDT,P$,MC,MR,InF,DF,DV,MMF,MME,min,max,EC,ER,SF,SC,SS)

SelectTaper:

  Msg$ = "Cosine-taper raw-data from "+InpFil$+"?"

  CALL ClickYesNo(MsgC,MsgR,Msg$,CosTpr)

  IF NOT CosTpr THEN GOTO OutputFTFName

   P$ = "Percent of full data-set for one of the tapered-tails:"
  InF = FALSE ' decimal inputs
   DF = TRUE  ' allow default
   DV = 3
  MME = FALSE ' include exact max and min values
  max = 20
   SF = TRUE  ' use slider
   SC = MC
   SS = 5     ' slider-scale 5 means 0.2 increment on decimal-input

  CALL NumericInput(PctTpr,P$,MC,MR,InF,DF,DV,MMF,MME,min,max,EC,ER,SF,SC,SS)
```

```
OutputFTFName:

  CALL ClearRows(MsgR,1!)
  LOCATE MsgR+2,MsgC
  PRINT "(Default = ";InpFil$;+".FTF)"
  LOCATE MsgR,MsgC
  COLOR 3,1
  PRINT "Enter output file name for filter results:";
  COLOR 1,0
  INPUT" ",OutFTF$
  IF LEN(OutFTF$)=0 THEN OutFTF$=InpFil$+".FTF"

  CALL OldFile(OutFTF$,FileOld)

' not finding file is desired here, but still triggers error-trapping and
' have to reopen window if using interpreter.

'$IGNORE ON

  IF NOT FileOld THEN WINDOW 2,"Fourier Filter",(1,1)-(617,385),22,1

'$IGNORE OFF

FileOld = FALSE '*****

  IF FileOld THEN
    IF Compiled THEN
      scrn = -1
    ELSE
      scrn = 1
    END IF
    BEEP
    BMsg$ = "HEY WAIT A MINUTE!"
    EMsg$ = OutFTF$+" aleady exists!"
    CALL MakeWindow(OK,BMsg$,EMsg$,"Over Write","Bail Out!",MsgC,MsgR,scrn)
    IF NOT OK THEN GOTO OutputFTFName
  END IF

  CALL ClearRows(MsgR,3!)

  Msg$ = "Output frequency-domain data to a file?"

  CALL ClickYesNo(MsgC,MsgR,Msg$,FrqOut)

  IF NOT FrqOut THEN GOTO SelectFreqTaper

OutputFRQName:

  CALL ClearRows(MsgR,1!)
  LOCATE MsgR+2,MsgC
  PRINT "(Default = ";OutFTF$;+".FRQ)"
  LOCATE MsgR,MsgC
  COLOR 3,1
  PRINT "Enter output file name for frequency data:";
  COLOR 1,0
  INPUT" ",OutFRQ$
  IF LEN(OutFRQ$)=0 THEN OutFRQ$=OutFTF$+".FRQ"

  CALL OldFile(OutFRQ$,FileOld)

'$IGNORE ON

  IF NOT FileOld THEN WINDOW 2,"Fourier Filter",(1,1)-(617,385),22,1
```

```
'$IGNORE OFF

  IF FileOld THEN
    IF Compiled THEN
      scrn = -1
    ELSE
      scrn = 1
    END IF
    BEEP
    BMsg$ = "HEY WAIT A MINUTE!"
    EMsg$ = OutFRQ$+" aleady exists!"
    CALL MakeWindow(OK,BMsg$,EMsg$,"Over Write","Bail Out!",MsgC,MsgR,scrn)
    IF NOT OK THEN GOTO OutputFRQName
  END IF

  CALL ClearRows(MsgR,3!)

SelectFreqTaper:

  Msg$ = "Select frequency-domain tapering method"
  LOCATE MsgR,MsgC
  COLOR 3,2
  PRINT Msg$
  COLOR 1,0
  IF LEN(Msg$) < 18 THEN
    X1=MsgC
  ELSE
    X1=MsgC+(LEN(Msg$)-18)\2
  END IF

  X2=X1+11

  CALL DrawBox("Linear",X1,MsgR+3!,1!)
  CALL DrawBox("Cosine",X2,MsgR+3!,1!)

FreqTaperLoop:

  CALL MouseColRow(MCol,MRow)

  CALL CheckBox(MCol,MRow,"Linear",X1,MsgR+3!,RL)
  CALL CheckBox(MCol,MRow,"Cosine",X2,MsgR+3!,RC)

  IF RC THEN
    FTapr=TRUE
  ELSEIF RL THEN
    FTapr=FALSE
  ELSE
    GOTO FreqTaperLoop
  END IF

  CALL ClearRows(MsgR,1!)
  CALL EraseBox("Linear",X1,MsgR+3!)
  CALL EraseBox("Cosine",X2,MsgR+3!)

EnterStopPer:

   P$ = "Stop-Period for spectrum taper (hours):"
   DV = 30
  MME = FALSE ' include exact max and min values
  max = 60
   SF = TRUE  ' use slider

  CALL NumericInput(TSTOP,P$,MC,MR,InF,DF,DV,MMF,MME,min,max,EC,ER,SF,SC,SS)
  TMIN = TSTOP
```

```
EnterPassPer:

   P$ = "Pass-Period for spectrum taper (hours):"
   DV = 40

  CALL NumericInput(TPASS,P$,MC,MR,InF,DF,DV,MMF,MME,min,max,EC,ER,SF,SC,SS)
  TMAX = TPASS

  IF TPASS =< TSTOP THEN
    ErrMsg$ = "Stop-Period ="+STR$(TSTOP)+", Pass-Period must be greater!"
    CALL ErrorMsg(ErrMsg$,ErrC,ErrR,TRUE)
    GOTO EnterPassPer
  END IF
  CALL ClearRows(ErrR,1!)

  CALL CenterMsg("Detrend Options",MsgR,TRUE)

  GOSUB DrawControls

DetrendOptions:

  GOSUB CheckControls

  IF NOT Contn THEN GOTO DetrendOptions

'*****************************************************
'*****************************************************
'**                                                 **
'**                DATA PREPARATION                 **
'**                                                 **
'*****************************************************
'*****************************************************

  Msg$ = "Reading "+InpFil$
  CALL CenterMsg(Msg$,MsgR,TRUE)
  CALL FileRead(InpFil$,DAT(),NTOTAL)

  Msg$ = "I read"+STR$(NTOTAL)+" records from "+InpFil$
  CALL CenterMsg(Msg$,MsgR+1,TRUE)

  CALL Detrend(NTOTAL,B0,B1,R2,MY,RLSTr,REPTr,RMean,MsgR+2,NM)

  MR = MsgR+2+NM

  IF CosTpr THEN CALL CosTaper(NTOTAL,PctTpr,MR)

  NOUT = NTOTAL  ' save original input-length before zero-padding

  CALL ZeroPad(NTOTAL,MR+1)

'*****************************************************
'*****************************************************
'**                                                 **
'**                  FOURIER FILTER                 **
'**                                                 **
'*****************************************************
'*****************************************************

 ' The Fourier-filter performs a FFT on the prepared input-data,
 ' sets Fourier-coefficients to zero for high-frequencies (those
 ' with periods less than the specified stop-period), tapers
 ' coefficients, (linear or cosine-taper as specified in setup),
 ' from 0 at the specified stop-period, to no taper at the
 ' specified pass-period, and inverse-transforms the result to
 ' produce a verison of the input that has high-frequencies removed.
```

```
' Tapering of the coefficeints is done to reduce Gibbs phenomena
' ringing at the ends of the filtered-data record.  Generally, the
' input-data are detrended and split-cosine-bell tapered to reduce
' leakage in the frequency-domiain spectrum.
'
' Note: there was alot of switching betweem integer/real single/double
' values in the original FORTRAN code -- that is all ignored here, but
' there is an excess of variable names in this port as a result.

' Set max and min frequencies and DF:

  NN = 2*NTOTAL
  N2 = NTOTAL/2

  DT6 = IDT/60          ' assume input sample-interval is in minutes
   DF = 1/(NTOTAL*DT6) ' filter works in hours

' Above 2 expressions are correct, but, since FFT splits data into
' real and imaginary parts, we have to set the frequencies to twice what
' is specified:

  FMIN = 4/TMAX
  FMAX = 4/TMIN

  NMIN = INT(FMIN/DF)
  NMAX = INT(FMAX/DF)
    DN = NMAX-NMIN

  CALL CenterMsg("Performing FFT",MR+2,TRUE)
  CALL REALFT(NTOTAL,1!)

  IF FTapr THEN
    CALL CenterMsg("Performing Cosine-Taper of Fourier-Coeffs.",MR+3,TRUE)
    FOR j=NMIN TO NN
      Arg=(NMAX-j)/DN
      IF j < NMAX THEN A =.5-.5*COS(3.1416*Arg)
      IF j >= NMAX THEN A = 0
      DAT(j)=A*DAT(j)
    NEXT j
  ELSEIF NOT FTapr THEN
    CALL CenterMsg("Performing Linear-Taper of Fourier-Coeffs.",MR+3,TRUE)
    FOR j=NMIN TO NN
      Arg=(NMAX-j)/DN
      IF j < NMAX THEN A=Arg
      IF j >= NMAX THEN A 0
      DAT(j)=A*DAT(j)
    NEXT j
  END IF

FreqDomainOut:

  IF FrqOut THEN
    Msg$ = "Outputting tapered Fourier-coefficients to "+OutFRQ$
    CALL CenterMsg(Msg$,MR+4,TRUE)
    OPEN OutFRQ$ FOR OUTPUT AS #1
    PRINT#1,"* Results for data from: ";InpFil$
    PRINT#1,"*"
    PRINT#1,"* NTOTAL =";NTOTAL;" points.  DT =";IDT;" minutes."
    PRINT#1,"* NMIN =";NMIN;" NMAX =";NMAX;"-- transition-band in frq-domain"
    PRINT#1,"*"
    PRINT#1,"* Fourier-Coefficients in the frequency-domain"
    PRINT#1,"*"
    PRINT#1,"*   J       +FREQ          Ak        J      -FREQ         -Ak"
    PRINT#1,"*  ----  ----------  ----------  ----  ----------  ----------"
    FOR j=1 TO NTOTAL
```

```
      FREQ=DF*(j-1)
      FREQ1=-DF*(j-1)
      jj=NN+1-j
      IF FREQ = 0 THEN
        PRINT #1,USING"   ####  ###.######  #######.##";j;FREQ;DAT(j)
      END IF
      IF FREQ > 0 THEN
        PRINT #1,USING"   ####  ###.######  #######.##";j;FREQ;DAT(j);
        PRINT #1,USING"  ####  ###.######  #######.##";jj;FREQ1;DAT(j)
      END IF
    NEXT j
    CLOSE #1
    MR = MR + 1 ' to keep progress output-lines straight
  END IF

  CALL CenterMsg("Performing inverse-FFT",MR+4,TRUE)
  CALL REALFT(NTOTAL,-1!)

' since the factor '1/N/ is not included in REALFT() it is employed below

  A=1/NTOTAL

  IF AMean THEN
    Msg$ = "Adding mean back to filtered data"
    CALL CenterMsg(Msg$,MR+4,TRUE)
    MR = MR + 1
  ELSE
    MY = 0
  END IF

  FOR k=1 TO NTOTAL
    DAT(k)=MY+A*DAT(k)
  NEXT k

' Reset zeros from zero-padding

  IF NOUT < NTOTAL THEN
    FOR i=NOUT+1 TO NTOTAL
      DAT(i)=0
    NEXT i
  END IF

  Msg$ = "Outputting filtered-data to "+OutFTF$
  CALL CenterMsg(Msg$,MR+4,TRUE)

  NTAP = NOUT*PctTpr\100 ' length of tapered-tail

  OPEN OutFTF$ FOR OUTPUT AS #1

  PRINT#1,"* Fourier-transform filtered-series for ";InpFil$
  PRINT#1,"*"
  PRINT#1,"* stop-period =";TMIN;"hours, pass-period =";TMAX;"hours"
  PRINT#1,"* input-length =";NOUT;" filtered-length =";NTOTAL
  PRINT#1,"* raw-data taper-length =";NTAP;"points in each tail"
  FOR k = 1 TO NTOTAL
    PRINT#1,DAT(k)
  NEXT k
  CLOSE #1

  CLS

  Msg$ = "Do you wish to perform another analysis?"
  CALL ClickYesNo(MsgC,MsgR,Msg$,k)

  IF k THEN
```

```
    ERASE DAT
    GOTO Restart
  END IF

'$IGNORE ON       ' Interpreter explicitly closes full-size window and screen
   SCREEN CLOSE 1
   WINDOW CLOSE 2
'$IGNORE OFF

'  ERASE DAT
'  CLEAR ,25000
'  SYSTEM

END

'*****************************************************
'*****************************************************
'**                                                 **
'**                   ERROR TRAPS                   **
'**                                                 **
'*****************************************************
'*****************************************************

ErrorTraps:

' NOTE: If error-trapping occurs under the interpreter, custom-windows
'       are closed.  So, any that were in use at the time of the error
'       of the error have to be reopened.  The AC-BASIC compiler always
'       opens a full-sized window, and error-traps do not affect it.

  IF ERR = 53 THEN  ' file not found trap
    FileOld = FALSE
    RESUME NEXT
  ELSE              ' everything else crashes (but not much else can happen)
    ON ERROR GOTO 0
  END IF

'*****************************************************
'*****************************************************
'**                                                 **
'**            USER INTERFACE SUBROUTINES           **
'**                                                 **
'*****************************************************
'*****************************************************

DrawControls:

   Clr1 = 1 ' initial background-color for gadget-boxes
   Clr2 = 1
   Clr3 = 1
   Clr4 = 1
   Clr5 = 1

   RMean = FALSE ' initial logic-settings for gadget-arguments
   RLSTr = FALSE
   REPTr = FALSE
   AMean = FALSE
   Contn = FALSE

   CALL DrawBox(" Remove Mean ",LeftCol,TopRow   ,Clr1)
   CALL DrawBox("Least Squares",LeftCol,TopRow+4 ,Clr2)
   CALL DrawBox("  End Point  ",LeftCol,TopRow+8 ,Clr3)
   CALL DrawBox("Add Mean Back",LeftCol,TopRow+12,Clr4)
   CALL DrawBox("  Continue   ",LeftCol,TopRow+16,Clr5)
```

```
RETURN

'*************************************************

CheckControls:

' Use subprogram-pair CheckBoxT()/ToggleBox() to allow multiple
' selections and corrections -- keep toggling selections
' via main-program loop until 'Continue' is selected.

  CALL MouseColRow(MCol,MRow)

  IF MRow < TopRow-1 OR MRow > TopRow+17 THEN RETURN 'outside control area
  IF MCol < LeftCol-1 OR MCol > LeftCol+13 THEN RETURN

  CALL CheckBoxT(MCol,MRow," Remove Mean ",LeftCol,TopRow,R)
  IF R THEN
    CALL ToggleBox(LeftCol,TopRow," Remove Mean ",Clr1,RMean)
    IF AMean AND NOT RMean THEN  ' turn off add mean if turn off remove
      CALL ToggleBox(LeftCol,TopRow+12,"Add Mean Back",Clr4,AMean)
    END IF
    CALL ClearRows(ErrR,1!)
    RETURN
  END IF

  CALL CheckBoxT(MCol,MRow,"Least Squares",LeftCol,TopRow+4,R)
  IF R THEN
    CALL ToggleBox(LeftCol,TopRow+4,"Least Squares",Clr2,RLSTr)
    IF RLSTr AND REPTr THEN  ' turn off end-point if turned on least-squares
      CALL ToggleBox(LeftCol,TopRow+8,"  End Point  ",Clr3,REPTr)
    END IF
    CALL ClearRows(ErrR,1!)
    RETURN
  END IF

  CALL CheckBoxT(MCol,MRow,"  End Point  ",LeftCol,TopRow+8,R)
  IF R THEN
    CALL ToggleBox(LeftCol,TopRow+8,"  End Point  ",Clr3,REPTr)
    IF RLSTr AND REPTr THEN  ' turn off least-squares if turned on end-point
      CALL ToggleBox(LeftCol,TopRow+4,"Least Squares",Clr2,RLSTr)
    END IF
    CALL ClearRows(ErrR,1!)
    RETURN
  END IF

  CALL CheckBoxT(MCol,MRow,"Add Mean Back",LeftCol,TopRow+12,R)
  IF R THEN
    IF (RMean AND NOT AMean) OR AMean THEN
      CALL ToggleBox(LeftCol,TopRow+12,"Add Mean Back",Clr4,AMean)
      CALL ClearRows(ErrR,1!)
    ELSEIF NOT RMean THEN
      ErrMsg$ = "You have to remove a mean before you can add it back!"
      CALL ErrorMsg(ErrMsg$,ErrC,ErrR,FALSE)
      END IF
    RETURN
  END IF

  CALL CheckBox(MCol,MRow,"  Continue   ",LeftCol,TopRow+16,R)
  Contn = R
  CALL ClearRows(ErrR,1!)
  CALL EraseBox(" Remove Mean ",LeftCol,TopRow)
  CALL EraseBox("Least Squares",LeftCol,TopRow+4)
  CALL EraseBox("  End Point  ",LeftCol,TopRow+8)
  CALL EraseBox("Add Mean Back",LeftCol,TopRow+12)
  CALL EraseBox("  Continue   ",LeftCol,TopRow+16)
```

```
RETURN



'*****************************************************
'*****************************************************
'**                                                 **
'**           DATA PREPARATION SUBPROGRAMS          **
'**                                                 **
'*****************************************************
'*****************************************************

SUB Detrend(N,B0,B1,R2,MY,RLSTr,REPTr,RMean,MR,NM) STATIC

' Based on flags, remove least-squares-trend, remove end-point-trend,
' remove mean, or remove a trend and remove mean.  Messages are
' output beginning at row MR, and, since the number of output lines
' is variable, the number of messages is returned in MR.  If no action
' then no messages, and NM is returned as 0.  For end-point detrending
' B0 returns intercept, b, and B1 returns slope m.  For least-squares
' detrending, B0 and B1 return the regression-coefficients B0,B1 and
' R2 returns the 'goodness-of-fit', R-squared.  If removed without
' detrending, or if the mean is not removed, but there is detrending, MY
' returns the mean of the input-data.  MY returns the mean of the
' detrended-data if the mean is removed with detrending (detrending
' always happens first).

' The original FORTRAN code use a different algorithm for least-squares
' detrending and returned a factor that was used in producing the
' filtered output-data. For a filtered-value with index k, the extra factor
' times (K-1) was summed into that value.  This has been eliminated here.

  SHARED TRUE,FALSE

  NM = 0

  IF NOT RLSTr AND NOT REPTr AND NOT RMean THEN EXIT SUB

  IF RLSTr THEN
    Msg$ = "Removing least-squares trend-line, Y(i)' = B0 + B1*X(i)."
    CALL CenterMsg(Msg$,MR+NM,TRUE)
    NM = NM + 1
    SumXY = 0
    SumXX = 0
     SumX = 0
     SumY = 0
      SST = 0
      SSR = 0
    FOR i=1 TO N
      SumXY = i*DAT(i)
      SumXX = i*i
       SumX = SumX + i
       SumY = SumY + DAT(i)
    NEXT i
      B0 = (SumY*SumXX-SumXY*SumX)/(N*SumXX-SumX*SumX)
      B1 = (N*SumXY-SumX*SumY)/(N*SumXX-SumX*SumX)
      MY = SumY/N
    FOR i = 1 TO N
      SST = SST + (DAT(i)-MY)^2
       YI = B0 + B1*i
      SSR = SSR + (YI - MY)^2
      DAT(i) = DAT(i) - YI
    NEXT i
```

```
    R2 = SSR/SST
  END IF

  IF REPTr THEN
    Msg$ = "Removing end-point to end-point trend-line, Y = mX + b."
    CALL CenterMsg(Msg$,MR+NM,TRUE)
    NM = NM + 1
    X1 = 1
    Y1 = DAT(1)
    XN = N
    YN = DAT(N)
    B1 = (YN-Y1)/(XN-X1)
    B0 = Y1 - m*X1
    MY = 0
    FOR i = 1 TO N
      MY = MY + DAT(i)
      DAT(i) = DAT(i) - (B1*i + B0)
    NEXT i
    MY = MY/N
  END IF

  IF RMean THEN
    IF RLSTr OR REPTr THEN
      Msg$ = "Removing mean from detrended data."
    ELSE
      Msg$ = "Removing mean."
    END IF
    CALL CenterMsg(Msg$,MR+NM,TRUE)
    NM = NM + 1
    MY = 0
    FOR i = 1 TO N
      MY = MY + DAT(i)
    NEXT
    MY = MY/N
    FOR i = 1 TO N
      DATA(I)=DAT(I)-MY
    NEXT i
  END IF

END SUB

'*****************************************************

SUB CosTaper(N,PctTpr,MR) STATIC

' Apply split-cosine-bell taper to tails of input-data
' for specified percentage of total-record length.

  SHARED TRUE,FALSE

  pi = 3.141592

  NTapr = PctTpr*N\100

  IF NTapr = 0 THEN
    Msg$ = "0-point raw-data cosine-taper specified -- bypassing"
    CALL CenterMsg(Msg$,MR,TRUE)
    EXIT SUB
  END IF

  Msg$ = "Applying cosine-taper to"+STR$(NTapr)+"-point tails of record"
  CALL CenterMsg(Msg$,MR,TRUE)

  FOR i = 1 TO NTapr
    Arg = pi*(i-1)/NTapr
```

```
    Ah = .5-.5*COS(Arg)
    DAT(i) = Ah*DAT(i)
    DAT(N-i+1) = Ah*DAT(N-i+1)
  NEXT i

END SUB

'*****************************************************

SUB ZeroPad(N,MR) STATIC

' Zero-pad the input-data if not already a power-of-2 records long.
' The padded-length is returned in N.  Zero-padding is a good thing
' assuming data have been tapered, otherwise it will cause leakage.

  SHARED TRUE,FALSE

  NT = 1

  WHILE NT < N
    NT = NT*2
  WEND

  IF NT = N THEN
    CALL CenterMsg("Input data-set is a power-of-2 records long",MR,TRUE)
    EXIT SUB
  ELSE
    Msg$="Zero-Padding input-record from"+STR$(N)+" to"+STR$(NT)+" records"
    CALL CenterMsg(Msg$,MR,TRUE)
  END IF

  FOR i = N+1 TO NT
    DAT(i) = 0
  NEXT i

  N = NT

END SUB

'*****************************************************
'*****************************************************
'**                                                 **
'**            FOURIER FILTER SUBPROGRAMS           **
'**                                                 **
'*****************************************************
'*****************************************************

SUB FOUR1(NN,ISIGN) STATIC

' Replaces DAT() by its decrete-Fourier-transform if ISIGN is input as 1;
' Replaces DAT() by NN times its inverse-discrete-Fourier-Transform if
' ISIGN is input as -1.  DAT() is a complex array of lenght NN or,
' equivalently, A real array of length 2*NN.  NN must be A POWER of 2.

' Algorithm from: "Numerical Recipes" Pg. 394 by Bill Press et al.

  N=2*NN
  j=1

  FOR i=1 TO N STEP 2
    IF j > i THEN
      TEMPR=DAT(j)
      TEMPI=DAT(j+1)
      DAT(j)=DAT(i)
      DAT(j+1)=DAT(i+1)
```

```
      DAT(i)=TEMPR
      DAT(i+1)=TEMPI
    END IF
    m=N/2

FLoop1:

    IF((m > 2) AND (j > m))THEN
      j=j-m
      m=m/2
      GOTO FLoop1
    END IF
    j=j+m
  NEXT i

  MMAX=2

FLoop2:

  IF (N > MMAX) THEN
    ISTEP=2*MMAX
    THETA=6.2831853072#/(ISIGN*MMAX)
    WPR=-2*SIN(.5*THETA)^2
    WPI=SIN(THETA)
    WR=1
    WI=0
    FOR m=1 TO MMAX STEP 2
      FOR i=m TO N STEP ISTEP
        j=i+MMAX
        TEMPR=WR*DAT(j)-WI*DAT(j+1)
        TEMPI=WR*DAT(j+1)+WI*DAT(j)
        DAT(j)=DAT(i)-TEMPR
        DAT(j+1)=DAT(i+1)-TEMPI
        DAT(i)=DAT(i)+TEMPR
        DAT(i+1)=DAT(i+1)+TEMPI
      NEXT i
      WTEMP=WR
      WR=WR*WPR-WI*WPI+WR
      WI=WI*WPR+WTEMP*WPI+WI
    NEXT m
    MMAX=ISTEP
    GOTO FLoop2
  END IF

END SUB

'*******************************************************

SUB REALFT(N,ISIGN) STATIC

' Calcualtes the Fourier-transform a set of 2N real-valued data-points.
' Replaces these data (stored in array DAT()) by the positive-frequency
' half of its complex Fourier-transform.  The real-valued first and last
' components of the complex transform are returned elements DAT(1) and
' DAT(2) respectively.  N must be a power of 2.  This routine calculates
' the inverse-tranform of a complex data array if it is the transform
' of real data.  (Result in This case must be multiplied by 1/N)

' Algorithm from: "Numerical Recipes" Pg. 400 by Bill Press et al.

  THETA=6.2831853072#/2/N
  C1=.5
  IF (ISIGN = 1) THEN
    C2=-.5
    CALL FOUR1(N,+1!)
```

```
  ELSE
    C2=.5
    THETA=-THETA
  END IF
  WPR=-2*SIN(.5*THETA)^2
  WPI=SIN(THETA)
  WR=1+WPR
  WI=WPI
  N2P3=2*N+3
  FOR i=2 TO N/2+1
    I1=2*i-1
    I2=I1+1
    I3=N2P3-I2
    I4=I3+1
    WRS=WR
    WIS=WI
    H1R=C1*(DAT(I1)+DAT(I3))
    H1I=C1*(DAT(I2)-DAT(I4))
    H2R=-C2*(DAT(I2)+DAT(I4))
    H2I=C2*(DAT(I1)-DAT(I3))
    DAT(I1)=H1R+WRS*H2R-WIS*H2I
    DAT(I2)=H1I+WRS*H2I+WIS*H2R
    DAT(I3)=H1R-WRS*H2R+WIS*H2I
    DAT(I4)=-H1I+WRS*H2I+WIS*H2R
    WTEMP=WR
    WR=WR*WPR-WI*WPI+WR
    WI=WI*WPR+WTEMP*WPI+WI
  NEXT i
  IF (ISIGN = 1) THEN
    H1R=DAT(1)
    DAT(1)=H1R+DAT(2)
    DAT(2)=H1R-DAT(2)
  ELSE
    H1R=DAT(1)
    DAT(1)=C1*(H1R+DAT(2))
    DAT(2)=C1*(H1R-DAT(2))
    CALL FOUR1(N,-1!)
  END IF

END SUB

'*****************************************************
'*****************************************************
'**                                                 **
'**             USER INTERFACE SUBPROGRAMS          **
'**                                                 **
'*****************************************************
'*****************************************************

SUB MakeWindow(OK,BdrMsg$,WndMsg$,box1msg$,box2msg$,X,Y,scrn) STATIC

' Open a window to give a window-border title, a message and two
' gadgets to click on.  X,Y give the character-postion for the
' upper-left corner of the message-window.  The screen to open the
' window on is an option because the compiler uses the Workbench
' screen (ID = -1), while the interpreted version opens screen 1.

  SHARED TRUE,FALSE

  WndMsg$ = " " + WndMsg$ + " "

  WndWdth = 2 + LEN(box1msg$) + 4 + LEN(box2msg$) + 2

  IF LEN(WndMsg$)<WndWdth THEN
    X1 = 2
```

```
    X2 = 2+LEN(box1msg$)+4
  ELSE
    WndWdth = LEN(WndMsg$)+3
    X1 = (WndWdth - (LEN(box1msg$) + 4 + LEN(box2msg$)))\2 + 1
    X2 = X1 + LEN(box1msg$) + 4
  END IF

  CALL ChrToPixPos(X,Y,XP,YP)

  WINDOW 3,BdrMsg$,(XP,YP)-(XP+(WndWdth-1)*8,YP+55),22,scrn
  COLOR 3,2
  PAINT (1,1),2
  LOCATE 2,2
  PRINT WndMsg$
  CALL DrawBox(box1msg$,X1,5!,1!)
  CALL DrawBox(box2msg$,X2,5!,1!)

ActionLoop:

  CALL MouseColRow(MCol,MRow)
  CALL CheckBox(MCol,MRow,box1msg$,X1,5!,R)
  IF R THEN
    OK=TRUE
    GOTO DoAction
  END IF
  CALL CheckBox(MCol,MRow,box2msg$,X2,5!,R)
  IF R THEN
    OK=FALSE
    GOTO DoAction
  END IF

  GOTO ActionLoop

DoAction:

  WINDOW CLOSE 3

END SUB

'*****************************************************

SUB ClickYesNo(MsgCol,MsgRow,Msg$,k) STATIC

' Click Yes or No.

  SHARED TRUE,FALSE

  LOCATE MsgRow,MsgCol
  COLOR 3,2
  PRINT Msg$
  COLOR 1,0

  IF LEN(Msg$) < 11 THEN
    X1=MsgCol
  ELSE
    X1=MsgCol+(LEN(Msg$)-11)\2
  END IF

  X2=X1+7

  CALL DrawBox("Yes",X1,MsgRow+3!,1!)
  CALL DrawBox("No ",X2,MsgRow+3!,1!)

YesNoLoop:
```

```
  CALL MouseColRow(MCol,MRow)

  CALL CheckBox(MCol,MRow,"Yes",X1,MsgRow+3!,RY)
  CALL CheckBox(MCol,MRow,"No ",X2,MsgRow+3!,RN)

  IF RY THEN
    k=TRUE
  ELSEIF RN THEN
    k=FALSE
  ELSE
    GOTO YesNoLoop
  END IF

  CALL ClearRows(MsgRow,1!)
  CALL EraseBox("Yes",X1,MsgRow+3!)
  CALL EraseBox("No ",X2,MsgRow+3!)

END SUB

'*****************************************************

SUB NumericInput(V,P$,MC,MR,InF,DF,DV,MMF,MME,min,max,EC,ER,SF,SC,SS) STATIC

' Prevents AmigaBASIC error-traps on silly inputs so screen-format
' doesn't get screwed up.  Returns value in V.  Allows only intergers
' if InF is TRUE.  A default output (DV) is allowed if DF is TRUE.
' A maximum and minimum range is included in the error checks
' (Min =< V =< Max) if MMF is TRUE.  Leading spaces are stripped
' but spaces anywhere else are considered an error.  Exponentials (e.g.,
' -2.345E-67) allowed in single or double-precision (E or D) format.
' MC,MR are the column and row for promt-message P$.  EC,ER are the
' column and row for error-messages.  SF is a flag for optional
' mouse-slider input.  SC is the starting column for the slider if used.
' SS is the slider scale. Valid max and min values are required for
' slider use.  MME is a flag for excluding max and min from the acceptable
' input-range for key inputs, this allows, e.g., excluding a minimum of zero.

  SHARED TRUE,FALSE

DefaultInput:

  IF DF THEN

    IF DV<0 THEN
      DV$=" "+STR$(DV)
    ELSE
      DV$=STR$(DV)
    END IF

    LOCATE MR,MC
    COLOR 3,2
    PRINT P$
    COLOR 1,0

    D$ = "Default Value ="+DV$

    C = LEN(D$) - LEN(P$)  ' center default-string under promt-string
    C = MC - C\2           ' note: if C < 0 then sign causes shift right

    LOCATE MR+2,C
    PRINT D$

    C = 18 - LEN(P$)  ' use/change gadgets total 18 characters long

    C = MC - C\2
```

```
    CALL DrawBox(" Use  ",C,MR+5,1!)
    CALL DrawBox("Change",C+10,MR+5,1!)

DefaultLoop:

    CALL MouseColRow(MCol,MRow)
    CALL CheckBox(MCol,MRow," Use  ",C,MR+5,R)
    IF R THEN
      CALL EraseBox(" Use  ",C,MR+5)
      CALL EraseBox("Change",C+10,MR+5)
      CALL ClearRows(MR,3!)
      V = DV
      EXIT SUB
    END IF
    CALL CheckBox(MCol,MRow,"Change",C+10,MR+5,R)
    IF R THEN
      CALL EraseBox(" Use  ",C,MR+5)
      CALL EraseBox("Change",C+10,MR+5)
      CALL ClearRows(MR,3!)
      GOTO StartNumInput
    ELSE
      GOTO DefaultLoop
    END IF

  END IF

StartNumInput:

  IF SF THEN   ' use slider

     CALL Slider(SC,MR,min,max,InF,SS,P$,V,TRUE)

  ELSE         ' use key and <CR> input

    IF MMF THEN
      IF min<0 THEN
        min$=" "+STR$(min)
      ELSE
        min$=STR$(min)
      END IF
      IF max<0 THEN
        max$=" "+STR$(max)
      ELSE
        max$=STR$(max)
      END IF
    END IF
    LOCATE MR+2,MC
    IF MMF AND NOT DF THEN
      IF MME THEN
        PRINT "("+min$+" < Value <"+max$+")"
      ELSE
        PRINT "( Range ="+min$+" to"+max$+")"
      END IF
    END IF
    IF DF AND MMF THEN
      IF MME THEN
        PRINT "("+min$+" < Value <"+max$+" -- Default ="+DV$+")"
      ELSE
        PRINT "( Range ="+min$+"to"+max$+" -- Default ="+DV$+")"
      END IF
    END IF

    CALL ClearRows(MR,1!)
    LOCATE MR,MC
```

```
    COLOR 3,2
    PRINT P$;
    COLOR 1,0
    LINE INPUT;" ";V$

    IF LEN(V$) = 0 THEN  ' carriage-return is an error unless defaults active
      IF DF THEN
        V = DV
        GOTO NumInputOver
      END IF
      GOTO InputError
    END IF

    FOR i=1 TO LEN(V$)                          ' strip leading spaces
      IF MID$(V$,i,1)<>" " THEN GOTO StripOver
      V$ = MID$(V$,i+1 )
    NEXT i

    GOTO InputError                             ' all spaces is an error

StripOver:

    NumStr$ = "0123456789eEdD+-."               ' acceptable input characters

    FOR i=1 TO LEN(V$)
      IF INSTR(NumStr$,MID$(V$,i,1))=0 THEN GOTO InputError
    NEXT i

    V = VAL(V$)

    IF InF THEN
      IF V<>INT(V) THEN GOTO IntegerError
    END IF

    IF MMF THEN
      IF MME THEN
        IF V=<min OR V>=max THEN GOTO RangeError
      ELSE
        IF V<min OR V>max THEN GOTO RangeError
      END IF
    END IF

  END IF

NumInputOver:

  CALL ClearRows(ER,1!)
  CALL ClearRows(MR,3!)
  EXIT SUB

InputError:

  CALL ClearRows(ER,1!)
  CALL ErrorMsg("That is not an acceptable input!",EC,ER,TRUE)
  GOTO StartNumInput

IntegerError:

  CALL ClearRows(ER,1!)
  CALL ErrorMsg("Please enter an INTEGER value!",EC,ER,TRUE)
  GOTO StartNumInput

RangeError:

  CALL ClearRows(ER,1!)
```

```
  IF MME THEN
    ErrMsg$ = "Enter a value between"+min$+" and"+max$+"!"
  ELSE
    ErrMsg$ = "Enter a value from"+min$+" to"+max$+"!"
  END IF
  CALL ErrorMsg(ErrMsg$,EC,ER,TRUE)
  GOTO StartNumInput

END SUB

'*****************************************************

SUB Slider(SCol,SRow,min,max,IntFlg,scale,Msg$,LEVEL,ClearSlider) STATIC

' Mouse slider input routine. You must click on the slider before it will
' accept input.  Hold the left mouse button down until desired level is
' reached.  Release button and that level is returned.

' msg$ gets printed below the slider followed by the current slider-value.

' min and max are the range for the slider.

' level is the return slider value.'

' SRow and SCol are the screen row and column for the slider.

' Scale gives the number of horizontal pixels for one unit on the slider.

' IntFlg flags whether output is an integer (TRUE) or a floating (FALSE)
' value.  In integer mode, scale gives a range of pixels that relate to a
' single integer value.  In floating mode, scale gives the resolution of the
' decimal output, e.g., a scale of 4 means the pixel increment is in .25
' unit steps from min to max.  The decimal output display is limited to
' 3 places, but the full calculated decimal-value is returned.

' ClearSlider is a flag that optionally will leave the slider displayed on
' exit when FALSE.  Sometimes this is 'prettier' when using slider
' repeatedly for input with the same min and max, but different message.

  SHARED TRUE,FALSE

' X1 and Y1 are the pixel coordinates of the upper left corner of the slider

  CALL ChrToPixPos(SCol,SRow,X1,Y1)

' X2 and Y2 define the coordinates of the lower right corner of the slider

  W = scale*(max - min)

  IF (NOT IntFlg) AND (W <> INT(W)) THEN W = INT(W+1)

        ' For floating input, have to make sure get a whole number of pixels,
        ' or might not get full-scale reading on slider.  And in this case
        ' have to check that output doesn't overrun max value.

  X2 = X1 + W

  Y2 = Y1 + 5

' the B values define a 5 pixel wide border around the slider (also 5 wide)

  BX1 = X1 - 5 : BY1 = Y1 - 5
  BX2 = X1 + W + 5 : BY2 = Y1 + 10

  LINE (BX1,BY1)-(BX2,BY2),1,bf
```

```
  LINE (X1,Y1)-(X2,Y2),3,bf

' calculate column to center message under slider box.  If msg$ is longer
' than slider, is shifted left, if shorter than slider, is shifted right.

  CntrCol = (W/8 - LEN(Msg$))\2 + SCol
  IF CntrCol < 1 THEN CntrCol = 1

  LOCATE SRow+2,CntrCol
  COLOR 3,2
  PRINT Msg$;
  COLOR 1,0
  PRINT min

ActivateLoop:

  ClrMse = MOUSE(0)

  WHILE MOUSE(0)=0
  WEND

  X=MOUSE(1) : Y=MOUSE(2)

  IF X<X1 OR X>X2 THEN GOTO ActivateLoop
  IF Y>Y2 OR Y<Y1 THEN GOTO ActivateLoop

SetLevelLoop:

  ClrMse = MOUSE(0)

  X=MOUSE(1) : Y=MOUSE(2)

  IF MOUSE(0) >= 0 THEN GOTO LevelSet

  IF X<X1 OR X>X2 THEN GOTO SetLevelLoop
  IF Y>Y2 OR Y<Y1 THEN GOTO SetLevelLoop

  LINE(X1,Y1)-(X,Y2),2,bf
  IF X+1 <= X2 THEN LINE(X+1,Y1)-(X2,Y2),3,bf

  IF IntFlg THEN
    LEVEL = (X-X1)\scale + min
  ELSEIF NOT IntFlg THEN
    LEVEL = (X-X1)/scale + min
    IF LEVEL > max THEN LEVEL = max
  END IF

  LOCATE SRow+2,CntrCol+LEN(Msg$)
  PRINT SPACE$(10)

  LOCATE SRow+2,CntrCol+LEN(Msg$)
  IF IntFlg THEN
    PRINT LEVEL
  ELSEIF NOT IntFlg THEN
    LEVEL$ = STR$(LEVEL)
    LEVEL$ = MID$(LEVEL$,1,INSTR(LEVEL$,".")+3)
    PRINT LEVEL$
  END IF

  GOTO SetLevelLoop

LevelSet:

  IF ClearSlider=TRUE THEN
    CALL ClearRows(SRow-3,5!)
```

```
  END IF

END SUB

'*****************************************************

SUB MouseColRow(MCol,MRow) STATIC

' Get screen column-and-row-position mouse-pointer when last left-clicked.

  CALL PMouseColRow(PCol,PRow)  ' get pixel-coordinates

  CALL PixToChrPos(PRow,PCol,MRow,MCol) ' convert to screen column-and-row

END SUB

'*****************************************************

SUB PMouseColRow(PCol,PRow) STATIC

' Get screen X,Y pixel-position of mouse-pointer when last left-clicked.

' NOTE: The AC-BASIC compiler does not handle MOUSE() quite like the
'       AmigaBASIC interpreter.  Hence, this is not just a simple
'       WHILE MOUSE(0)=0 : WEND loop.  The routine below also works with
'       the interpreter.

MouseLoop:

  ClearMouse = MOUSE(0)

  Dly! = TIMER
  WHILE TIMER < Dly! + .5
  WEND

  MouseEvent = MOUSE(0)

  IF MouseEvent > 0 THEN  ' only looking for clicks, not holds.
    PCol = MOUSE(1)
    PRow = MOUSE(2)
  ELSE
    GOTO MouseLoop
  END IF

END SUB

'*****************************************************

SUB PixToChrPos(PRow,PCol,SRow,SCol) STATIC

' Convert from pixel-position to screen-character position.
' Converts to nearest character-position containing the pixel.
' (Assuming default Topaz font.)

  SRow = PRow\8 + 1
  SCol = PCol\8 + 1

END SUB

'*****************************************************

SUB ChrToPixPos(SRow,SCol,PRow,PCol) STATIC

' Convert from screen-character position to pixel-position.
' Gives values for upper-left of character position.
```

```
' (Assuming default Topaz font.)

  PRow = (SRow-2)*8
  PCol = (SCol-2)*8

END SUB

'*****************************************************

SUB DrawBox(Lab$,Col,row,Clr) STATIC

' Draw and label a box for clicking on.  Row and Col are the screen row
' and column for the box label.  Pixel positions are calculated here based
' on Row, Col and length of label.

  CALL ChrToPixPos(Col,row,X1,Y1)

  X2 = X1+(LEN(Lab$)+2)*8 : Y2 = Y1 + 22

  LINE (X1,Y1)-(X2,Y2),Clr,bf
  LOCATE row,Col
  COLOR 3,Clr
  PRINT Lab$
  COLOR 1,0

END SUB

'*****************************************************

SUB EraseBox(Lab$,Col,row) STATIC

' Erase a gadget-box previously drawn with label Lab$ starting at
' screen-position (Col,Row).

  CALL ChrToPixPos(Col,row,X1,Y1)

  X2 = X1+(LEN(Lab$)+2)*8 : Y2 = Y1 +22

  LINE (X1,Y1)-(X2,Y2),0,bf

END SUB

'*****************************************************

SUB CheckBox(MCol,MRow,Lab$,Col,row,R) STATIC

' Checks if last specified Mouse Col and Row are within
' a gadget-box with label Lab$ starting at Col and Row.
' R is returned TRUE and box background is changed to black,
' otherwise, R is returned FALSE.

  SHARED TRUE,FALSE

  R = FALSE

  X1 = Col-1 : Y1 = row-1
  X2 = Col+LEN(Lab$)+1 : Y2 = row+1

  IF MCol>=X1 AND MCol=<X2 AND MRow>=Y1 AND MRow=<Y2 THEN
    CALL DrawBox(Lab$,Col,row,2!)
    R = TRUE
  END IF

END SUB
```

```
'*****************************************************

SUB CheckBoxT(MCol,MRow,Lab$,Col,row,R) STATIC

' Checks if last specified Mouse Col and Row are within
' a gadget-box with label Lab$ starting at Col and Row.
' R is returned TRUE, otherwise, R is returned FALSE.
' The box is not redrawn here.  This routine is used in
' conjunction with ToggleBox() to allow reselection on
' multiple gadgets.

  SHARED TRUE,FALSE

  R = FALSE

  X1 = Col-1 : Y1 = row-1
  X2 = Col+LEN(Lab$)+1 : Y2 = row+1

  IF MCol>=X1 AND MCol=<X2 AND MRow>=Y1 AND MRow=<Y2 THEN R = TRUE

END SUB

'*****************************************************

SUB ToggleBox(Col,row,Lab$,Clr,Arg) STATIC

' If previous click set a gadget and it's argurment, then this
' resets, and vice-versa.  Use in conjunction with CheckBoxT() and
' then can make and correct multiple selections on a gadget-grid.

  SHARED TRUE,FALSE

  IF Clr = 1 THEN
    Clr = 2
  ELSEIF Clr = 2 THEN
    Clr = 1
  END IF

  IF Arg THEN
    Arg = FALSE
  ELSEIF NOT Arg THEN
    Arg = TRUE
  END IF

  CALL DrawBox(Lab$,Col,row,Clr)

END SUB

'*****************************************************

SUB ErrorMsg(ErrMsg$,MsgCol,MsgRow,ClrF) STATIC

  IF ClrF THEN CLS
  BEEP
  LOCATE MsgRow,MsgCol
  COLOR 2,3
  PRINT ErrMsg$
  COLOR 1,0

END SUB

'*****************************************************

SUB FileRead(InpFil$,FileRecs(),Count) STATIC
```

```
' Read in the rows of a file as character strings and count.
' Skips zero-length lines and lines that start with '*'
' This assumes a single-column format.

  Count = 0

  OPEN InpFil$ FOR INPUT AS #99 LEN=16384

  WHILE NOT(EOF(99))
    LINE INPUT #99,R$
    IF MID$(R$,1,1)="*" THEN SkipRow
    IF LEN(R$)=0 THEN SkipRow
    Count = Count+1
    FileRecs(Count)=VAL(R$)

SkipRow:

  WEND

  CLOSE #99

END SUB

'*****************************************************

SUB ClearRows(row,NumRows) STATIC

' Erease specified number of rows starting at specified row on screen.

  FOR i = 0 TO NumRows-1
    LOCATE row+i,1
    PRINT SPACE$(78)
  NEXT i

END SUB

 '*****************************************************

SUB OldFile(FileName$,FileOld) STATIC

' Sets FileOld to TRUE then checks if file is present by
' attempting to open for input.  If present, then opens,
' closes, exits.   If not, get error.  Error trap sets
' FileOld FALSE, then RESUME NEXT in the error trap returs
' to the line after the call.  Return of FileOld FALSE means
' that the file does not exist, so it is safe to open
' for output.  FileOld = TRUE means that the file exists so
' check whats going on. The variable name 'FileOld' in this
' subroutine is not optional due to use of error-trapping.

  SHARED TRUE,FALSE

  FileOld = TRUE

  OPEN FileName$ FOR INPUT AS #99
  CLOSE #99

END SUB

'*****************************************************

SUB CenterMsg(Msg$,MsgR,Bkgndon) STATIC

' Centers a message on the screen at the specified row, assuming
' 80-column display.  May use black background if desired.
```

```
  CALL ClearRows(MsgR,1!)
  IF Bkgndon THEN COLOR 3,2
  Col = 40 - LEN(Msg$)\2
  LOCATE MsgR,Col
  PRINT Msg$
  IF Bkgndon THEN COLOR 1,0

END SUB

'*****************************************************

SUB GenHarmonic(N,k,amp,phi,A0,HrmDat()) STATIC

' generate a sinusoid where n is the number of points in the record,
' k is the number of cycles in the record, amp is the amplitude,
' phi is the phase-angle, and A0 the offset.
' Results are returned in CosDat().

  pi = 3.141592

  T = N/k ' points in a period = number-of-points/harmonic-number

  FOR i=1 TO N
    HrmDat(i) = A0 + amp*COS(2*pi*(i-1)/T - phi)
  NEXT i

END SUB

'*****************************************************
```

GODIN.FOR

Purpose: A24A25[2] form 'Godin' tide removal filter

Language: FORTRAN (Microsoft V4.01)

```
c          ***********************************************
c          *                                             *
c          *                  GODIN.FOR                  *
c          *               Alan Swithenbank              *
c          *                February 1989                *
c          *                                             *
c          ***********************************************
c          *                                             *
c          * This program uses the A24A25^2 form of tide *
c          * removal filter suggested by Godin on page 65 *
c          * of THE ANALYSIS OF TIDES for hourly data.   *
c          *                                             *
c          * The counters are set up expecting no gaps in *
c          * the data.  Input arrays are dimensioned for *
c          * files no longer than 1024 records.          *
c          *                                             *
c          * This program was originally written in      *
c          * AmigaBASIC for 30 minute data. It was ported *
c          * for hourly data to MicroSoft FORTRAN v4.01  *
c          * August 1989.                                *
c          *                                             *
c          ***********************************************

c**** *****************************************************************

      PROGRAM GDNFLT

c**** *****************************************************************

c working variable declarations:

      REAL*8 Z(1024),X(48),Y(24),GODIN(952),ROWVAL(25)

      INTEGER*4 NUMCOL,FSTCOL,LSTCOL,COLNUM,NUMROW,I,J,K,L,IDX,COUNT
      INTEGER*4 ERRNUM

      CHARACTER*80 DATFIL,INPSTR,TMPSTR,OUTFIL

c function type declarations:

      CHARACTER*80 NUMSTR
      CHARACTER*80 CHGLWR

      REAL*8 STRNUM

      INTEGER*4 STRLEN
      INTEGER*4 GETCOL

c**** *****************************************************************

      PRINT*
      PRINT*,'Godin Filter -- for 1 hour interval, max 1024 record data'
      PRINT*
10    PRINT*
      PRINT*,'Enter data file name for filtering:'
      READ(*,100) DATFIL

c  get number of data columns in input file:

      OPEN (   UNIT = 31,
     &         FILE = DATFIL,
     &       STATUS = 'OLD',
     &          ERR = 1000,
     &       IOSTAT = ERRNUM )
```

```
20    READ (   UNIT = 31,
     &          FMT = 100,
     &          ERR = 1010,
     &       IOSTAT = ERRNUM,
     &          END = 1020 ) INPSTR

c skip blank lines and comments marked by '*' in data files:

      IF (STRLEN(INPSTR) .EQ. 0) GOTO 20
      IF (INDEX(INPSTR,'*') .NE. 0) GOTO 20

      CALL PARSE(INPSTR,ROWVAL,NUMCOL)
      TMPSTR = NUMSTR(DBLE(NUMCOL))

      CLOSE(31)

      PRINT*
      PRINT*,'I see ',TMPSTR(1:STRLEN(TMPSTR)),' columns in ',DATFIL
      PRINT*
      PRINT*,'The first row of ',DATFIL(1:STRLEN(DATFIL)),' is:'
      PRINT*
      WRITE(*,200) (ROWVAL(J),J=1,NUMCOL)

30    PRINT*
      PRINT*,'Enter number for first column to filter:'
      READ(*,100) TMPSTR
      FSTCOL = STRNUM(TMPSTR)
      IF (     (FSTCOL .LE. 0)
     &    .OR. (FSTCOL .GT. NUMCOL)
     &    ) THEN
         PRINT*,'Get Serious!'
         GOTO 30
      ENDIF

40    PRINT*
      PRINT*,'Enter number for last column to filter:'
      READ(*,100) TMPSTR
      LSTCOL = STRNUM(TMPSTR)
      IF (    (LSTCOL .LE. 0)
     &    .OR. (LSTCOL .GT. NUMCOL)
     &    .OR. (LSTCOL .LT. FSTCOL)
     &    ) THEN
         PRINT*,'Get Serious!'
         GOTO 40
      ENDIF

c input data:

      DO 90, COLNUM = FSTCOL,LSTCOL

      TMPSTR = NUMSTR(DBLE(COLNUM))
      PRINT*,'Reading column ',TMPSTR(1:STRLEN(TMPSTR)),' from ',DATFIL

      OPEN (   UNIT = 31,
     &         FILE = DATFIL,
     &       STATUS = 'OLD',
     &          ERR = 1000,
     &       IOSTAT = ERRNUM )

        NUMROW = 0

50    READ (   UNIT = 31,
     &          FMT = 100,
     &          ERR = 1010,
```

```
     &        IOSTAT = ERRNUM,
     &           END = 60 ) INPSTR

c skip blank lines and comments marked by '*' in data files:

      IF (STRLEN(INPSTR) .EQ. 0) GOTO 50
      IF (INDEX(INPSTR,'*') .NE. 0) GOTO 50

      NUMROW = NUMROW + 1

      CALL PARSE(INPSTR,ROWVAL,NUMCOL)
      TMPSTR = NUMSTR(DBLE(NUMCOL))

      Z(NUMROW) = ROWVAL(COLNUM)

      PRINT*, Z(NUMROW)

      GOTO 50

60    CONTINUE

      CLOSE(31)

      PRINT*
      PRINT*,NUMROW,' values input'

      PRINT*
      PRINT*,'Calculating filter outputs for column',COLNUM

      COUNT = 0
      IDX = 1

70    IF (IDX+24+48 .GT. 1024) GOTO 80

        COUNT = COUNT + 1

        Z0 = 0

        DO 9900 I=1,24
          Y(I) = 0
9900    CONTINUE

        DO 9910 I=1,48
          X(I) = 0
9910    CONTINUE

        DO 9920 K=1,48
          DO 9930 J = IDX,IDX+24
            X(K) = X(K) + Z(K+J)
9930      CONTINUE
9920    CONTINUE

        DO 9940 L = 1,24
          DO 9950 K = 0,24
            Y(L) = Y(L) + X(K+L)
9950      CONTINUE
9940    CONTINUE

        DO 9960 L = 1,24
          Z0 = Z0 + Y(L)
9960    CONTINUE

        GODIN(COUNT) = (1.0/(24.0*(25.0**2.0)))*Z0

        IDX = IDX + 1
```

```
      GOTO 70

80    CONTINUE

      TMPSTR = NUMSTR(DBLE(COLNUM))
      OUTFIL = 'COLUMN' // TMPSTR(1:STRLEN(TMPSTR)) //  '.GDN'

      PRINT*
      PRINT*,'Saving column ',TMPSTR(1:STRLEN(TMPSTR)),' outputs'
      PRINT*,'to ',OUTFIL

      OPEN (   UNIT = 32,
     &         FILE = OUTFIL,
     &       STATUS = 'UNKNOWN',
     &          ERR = 1000,
     &       IOSTAT = ERRNUM )

c  NOTE: Filter places value at 35.5 hours from beginning of each segment.
c        Thesis data start at 065 @ 0315 so first filter value is at
c        066 @ 1445.  Since are hourly data at 15 minutes after the hour
c        the Godin filter outputs are 30 m offset from the original data
c        so interpolate to get them to line up with the original data
c        starting at 066 @ 1515 (this makes later plotting, etc. much easier).

      DO 999 I=1,COUNT
        IF (I .LT. COUNT) THEN
          WRITE(32,200) (GODIN(I)+GODIN(I+1))/2.0
        ELSE
          WRITE(32,200) 1.5*(GODIN(I)-GODIN(I-1)) + GODIN(I-1)
      ENDIF

999   CONTINUE

      CLOSE(31)
      CLOSE(32)

90    CONTINUE

      GOTO 2000

c**** ***************************************************************
c**** ***************************************************************

c                          FORMAT STATEMENTS

c**** ***************************************************************

100   FORMAT(A)
200   FORMAT(25(F9.3,:))

c**** ***************************************************************
c**** ***************************************************************

c                          ERROR TRAPS

c**** ***************************************************************

1000  PRINT*,'I got an error ',ERRNUM,' trying to open ',DATFIL
      GOTO 10

1010  PRINT*,'I got an error ',ERRNUM,' trying to read ',DATFIL
      GOTO 10

1020  CONTINUE
```

```
      CLOSE(31)
      PRINT*
      PRINT*,'I can not see any data in ',DATFIL
      GOTO 10

c**** ******************************************************************
c**** ******************************************************************

2000  PRINT*
      PRINT*,'All Done!'
      PRINT*

      END

c**** ******************************************************************
c**** ******************************************************************

c          ***************************************************
c          ***************************************************
c          **                                               **
c          **                Utility Subprograms            **
c          **                                               **
c          ***************************************************
c          ***************************************************

c**** ******************************************************************
c**** ******************************************************************
      SUBROUTINE PARSE(INPSTR,ROWVAL,COLCNT)
c**** ******************************************************************
c**** ******************************************************************
c This subprogram parses numeric values out of the input row string,
c assuming the values are always separated one or more blanks.  The row
c values are returned in order in a numeric variable array. The number
c of data columns is also returned. Up to 25 input columns are allowed.
c**** ******************************************************************

      CHARACTER*80 INPSTR, OUTSTR, COLS(25)

      REAL*8 ROWVAL(25),STRNUM

      INTEGER*4 I,COLCNT,STRLEN

      LOGICAL*4 VALFLG

      COLCNT = 0
      VALFLG = .FALSE.

      DO 1 I=1,25
        COLS(I) = ' '
1     CONTINUE

      DO 10 I=1,STRLEN(INPSTR)

        IF (INPSTR(I:I) .NE. ' ' .AND. (.NOT. VALFLG)) THEN
          COLCNT = COLCNT+1
          COLS(COLCNT)=COLS(COLCNT)(1:STRLEN(COLS(COLCNT)))//INPSTR(I:I)
          VALFLG = .TRUE.
        ELSEIF (INPSTR(I:I) .NE. ' ' .AND.  VALFLG) THEN
          COLS(COLCNT)=COLS(COLCNT)(1:STRLEN(COLS(COLCNT)))//INPSTR(I:I)
        ELSEIF (INPSTR(I:I) .EQ. ' ') THEN
          VALFLG = .FALSE.
        ENDIF

10    CONTINUE
```

```
      DO 20 I=1,COLCNT
        ROWVAL(I) = STRNUM(COLS(I))
20    CONTINUE

      RETURN
      END


c**** ***************************************************************
c**** ***************************************************************
      FUNCTION STRNUM(INPSTR)
c**** ***************************************************************
c**** ***************************************************************
c This function converts a valid string representation of a number into
c a REAL*8 value for return to the calling routine.  This function error
c traps invalid inputs automatically, but the calling routine must be able
c to handle errors on it's own.  The return value on a error input is a
c small number (-1.0e-100).  Through a READ the character string INPSTR is
c opened as an internal file and, using the F30.0 FORMAT line 1, converted
c from a character string to a REAL*8 numeric value.  The F30.0 specification
c is overridden when, for example, INPSTR is 123.456e-7, by the forced
c format of the string itself.  If INPSTR is not a valid numeric type
c representation, the conversion bombs and is trapped as an I/O error.
c**** ***************************************************************

      REAL*8 STRNUM
      CHARACTER*(*) INPSTR

100   READ ( UNIT = INPSTR,
     &       FMT = 1,
     &       ERR = 1000 ) STRNUM

1     FORMAT(F30.0)

      RETURN

1000  INPSTR = '-1.0E-100'
      GOTO 100

      END

c**** ***************************************************************
c**** ***************************************************************
      FUNCTION NUMSTR(NUMBER)
c**** ***************************************************************
c**** ***************************************************************
c This function takes a REAL*8 value and converts it to a string.
c The value is written to a string via an internal reference in a
c WRITE statement.  When numbers fall outside + or - E+010 range
c the string is automatically output in scientific notation.  Within
c that range, floating point format is used.  When the value input has
c no decimal point, then the decimal is left off the string.
c**** ***************************************************************

      CHARACTER*80 NUMSTR
      REAL*8 NUMBER

      INTEGER*4 PLACE

c make the conversion to a string:

      IF (NUMBER .EQ. 0) THEN
        NUMSTR = '    0'
        GOTO 20
      ELSEIF (      (ABS(NUMBER) .GT. 1.0E+10)
```

```
     &          .OR. (ABS(NUMBER) .LT. 1.0E-10)
     &        ) THEN
                    WRITE (UNIT = NUMSTR,
     &                     FMT = 2 ) NUMBER
      ELSE
                    WRITE (UNIT = NUMSTR,
     &                     FMT = 1 ) NUMBER
      ENDIF

1     FORMAT(F30.15)
2     FORMAT(E15.6)

c remove leading blanks:

10    PLACE = INDEX(NUMSTR,' ')
      IF (PLACE .NE. 1) GOTO 20
      NUMSTR = NUMSTR(2: )
      GOTO 10

20    CONTINUE

c remove trailing decimal:

      IF (       (INT(NUMBER) .EQ. NUMBER)
     &     .AND. (INDEX(NUMSTR,'.') .NE. 0)
     &   ) NUMSTR = NUMSTR(1:INDEX(NUMSTR,'.')-1)

      RETURN

      END
c**** ******************************************************************
c**** ******************************************************************
      FUNCTION STRLEN(STRING)
c**** ******************************************************************
c**** ******************************************************************
c This function returns the actual character count length of a string.
c**** ******************************************************************

      INTEGER*4 STRLEN

c parameter declaration:

      CHARACTER*(*) STRING

c local declaration:

      INTEGER*4 I

      DO 10 I = LEN(STRING), 1, -1

        IF ( .NOT. (      (ICHAR(STRING(I:I)) .EQ. 32)
     &               .OR. (ICHAR(STRING(I:I)) .EQ.  0)
     &             )
     &     ) THEN
              STRLEN = I
              RETURN
        ENDIF

10    CONTINUE

      STRLEN = 0

      RETURN
      END
```

```
c**** ***************************************************************
c**** ***************************************************************
      FUNCTION GETCOL(STRING)
c**** ***************************************************************
c**** ***************************************************************
c This function finds the number of data columns in a record string.
c It assumes that at least one space separates values.  Commas are
c o.k. as long as they butt against one of the values they separate
c and there is a space between the comman and the other value.
c**** ***************************************************************

      INTEGER*4 GETCOL

c parameter declaration:

      CHARACTER*(*) STRING

c used function declaration:

      INTEGER*4 STRLEN

c local declarations:

      INTEGER*4 I

      LOGICAL*4 VALFLG

c**** ***************************************************************

      GETCOL = 0
      VALFLG = .FALSE.

      DO 10 I=1,STRLEN(STRING)

        IF (STRING(I:I) .NE. ' ' .AND. (.NOT. VALFLG)) THEN
          GETCOL = GETCOL+1
          VALFLG = .TRUE.
        ENDIF

        IF (STRING(I:I) .EQ. ' ') VALFLG = .FALSE.

10    CONTINUE

      RETURN
      END

c**** ***************************************************************
c**** ***************************************************************
      FUNCTION CHGLWR(INPSTR)
c**** ***************************************************************
c**** ***************************************************************

      CHARACTER*(*) INPSTR,CHGLWR

      INTEGER*4 STRLEN,I,TMPNUM

      CHGLWR = INPSTR

      DO 10 I=1,STRLEN(INPSTR)
        TMPNUM = ICHAR(INPSTR(I:I))
        IF (TMPNUM.GE.65 .AND. TMPNUM.LE.90) THEN
```

```
            TMPNUM = TMPNUM + 32
            CHGLWR(I:I) = CHAR(TMPNUM)
          ENDIF
10      CONTINUE

        RETURN
        END

C**** ***************************************************************
C**** ***************************************************************
```

HARMONICS

Purpose: Harmonic-analysis via calculation of Fourier-Coefficients

Language: AmigaBASIC (intended for use with AC-BASIC compiler)

```
'Harmonics  -- Alan Swithenbank, June 1990

' Does a harmonic-analysis of an input data-file of up to 2000 records
' for the first selected number of harmonics (up to 150).  The analysis is
' based on calculation of Fourier-coefficients for uniform discretly-sampled
' periodic-signals. The parameters for all calculated harmonics are output
' to a file, along with a separate sorted listing of the data for the
' largest ones, up to 30.  The largest, up to 15, are constructed and saved.

' A full-size window is explicitly opened by the interpreter.  The
' AC-BASIC compiler does this automatically and the window opening is
' $IGNOREd by the compiler.  This code is FAR from being as 'Amigized'
' as it should be (not even one mouse-click here).

  CLEAR,100000&

  DIM SHARED InpDat(2000)
  DIM OutDat(6,150),temp(2,150),Big10(6,30),HrmDat(2000),SumDat(2000)
  DIM RsdDat(2000)

  ON ERROR GOTO ErrorTraps

   TRUE = 1
  FALSE = 0

  Compiled = TRUE  ' run-time flag for interpreter/compiler operation

InputName:  ' put label above WINDOW incase got here by error-trapping
            ' RESUME statement which seems to close custom-windows

'$IGNORE ON
  SCREEN 1,640,400,2,4
  WINDOW 2,"Harmonics",(1,1)-(617,385),22,1
  Compiled = FALSE
'$IGNORE OFF

  MsgR = 10
  MsgC = 10

  ErrR =  3
  ErrC =  5

  CALL ClearRow(MsgR)
  LOCATE MsgR,MsgC
  INPUT"Enter file name for harmonic analysis: ",InpFil$

  CALL OldFile(InpFil$,FileErr) ' The name FileErr is NOT optional in
                                ' OldFile()! -- if a file does not exist
                                ' then error-trapping takes over and the
                                ' OldFile() subroutine does not return.
                                ' FileErr is set FALSE followed by a RESUME
                                ' NEXT in ErrorTraps anytime a file is
                                ' is not found.  RESUME NEXT considers CALL
                                ' to be the error-line, not the OPEN in
                                ' the subroutine.  FileErr is returned as
                                ' TRUE by OldFile() if the file exists.

  IF FileErr = FALSE THEN

'$IGNORE ON                                      ' interpreter error-trapping
    WINDOW 2,"Harmonics",(1,1)-(617,385),22,1   ' closes window, so reopen
'$IGNORE OFF

    ErrMsg$ = "I could not find "+InpFil$
    CALL ErrorMsg(ErrMsg$,ErrC,ErrR)
```

```
    GOTO InputName
  END IF
  CALL ClearRow(ErrR)

InputHarmonics:

  CALL ClearRow(MsgR)
  P$ = "Enter number of harmonics to calculate parameters for: "
  CALL NumericInput(nh,P$,MsgC,MsgR,TRUE,TRUE,30!,TRUE,1!,150!,ErrC,ErrR)

OutputName:

  CALL ClearRow(MsgR)
  LOCATE MsgR+1,MsgC
  PRINT "(Default = ";InpFil$;+".harm)"
  LOCATE MsgR,MsgC
  INPUT"Enter output file name: ",OutFil$

  IF LEN(OutFil$)=0 THEN OutFil$=InpFil$+".harm"

  CALL OldFile(OutFil$,FileErr)

  IF FileErr = TRUE THEN
    CLS
    BEEP
    LOCATE ErrR,ErrC
    COLOR 3,0
    PRINT OutFil$;" aleady exists!"
    COLOR 1,0
    LOCATE ErrR+2,ErrC
    INPUT "Do you wish to overwrite";Y$
    IF INSTR(UCASE$(Y$),"Y") = 0 THEN
      CALL ClearRow(ErrR)
      GOTO OutputName
    END IF
  END IF

  CLS

  LOCATE MsgR,MsgC
  PRINT "Reading ";InpFil$
  CALL FileRead(InpFil$,InpDat(),n)

  SumXSqr = 0
     SumX = 0

  FOR i=1 TO n
    SumXSqr = SumXSqr + InpDat(i)*InpDat(i)
       SumX = SumX + InpDat(i)
  NEXT i

  VarX = (n*SumXSqr - SumX^2)/(n*(n-1))

  A0 = SumX/n

  CALL ClearRow(MsgR)
  LOCATE MsgR,MsgC
  PRINT MID$(STR$(n),2);" records input from ";InpFil$

  LOCATE MsgR+2,MsgC
  PRINT "Total-Variance =";VarX,"A0 =";A0

  SumPwr = 0
      pi = 3.141592
```

```
FOR k=1 TO nh

  CALL ClearRow(MsgR+4)
  LOCATE MsgR+4,MsgC
  PRINT "Performing calculations for harmonic-";MID$(STR$(k),2)

  alphk = 0
  betak = 0

  FOR m=1 TO n
    alphk = alphk + InpDat(m)*COS(2*pi*(m-1)*k/n)
    betak = betak + InpDat(m)*SIN(2*pi*(m-1)*k/n)
  NEXT m

  alphk = 2*alphk/k
  betak = 2*betak/k

  Ak = SQR(alphk^2 + betak^2)

  phik = ATN(betak/alphk)

  pwrk = (Ak^2)/2

  SumPwr = SumPwr + pwrk

  OutDat(1,k) = k
  OutDat(2,k) = alphk
  OutDat(3,k) = betak
  OutDat(4,k) = Ak
  OutDat(5,k) = phik
  OutDat(6,k) = pwrk

NEXT k

CLS

OPEN OutFil$ FOR OUTPUT AS #1

LOCATE 2,5
PRINT "Harmonic-analysis results for ";InpFil$
PRINT #1,InpFil$
LOCATE 4,5
PRINT "There were";n;"records in ";InpFil$
PRINT #1,n
LOCATE  ,5
PRINT "Total-variance =";VarX,"A0 =";A0
PRINT #1,VarX,A0
LOCATE  ,5
PRINT "Parameters for";nh;"harmonics"
PRINT #1,nh
LOCATE 8,5
PRINT "    k       alpha     beta        A         phi       power"
LOCATE  ,5
PRINT "--------- --------- --------- --------- --------- ---------"

prncnt = 1

FOR k=1 TO nh
  LOCATE 9+prncnt,4
  FOR i=1 TO 6
    PRINT USING " ####.####";OutDat(i,k);
    PRINT #1, USING " ####.####";OutDat(i,k);
  NEXT i
  PRINT
  PRINT #1,
```

```
  prncnt=prncnt+1
  IF prncnt>30 AND k<nh THEN
    CALL WaitForKey(40!)
    FOR j=1 TO 30
      CALL ClearRow(9+j)
    NEXT j
    prncnt = 1
  END IF
NEXT k

LOCATE 9+prncnt+2,5
PRINT "Sum of power-spectral-estimates =";SumPwr
PRINT #1,SumPwr

CALL WaitForKey(9+prncnt+4)

CLS

IF nh < 30 THEN
  ne = nh
ELSE
  ne = 30
END IF

LOCATE MsgR,MsgC
PRINT "Sorting and extracting data for the";ne;"largest harmonics."

FOR i=1 TO nh
  temp(1,i)=OutDat(6,i) ' power-spectral-estimate
  temp(2,i)=OutDat(1,i) ' harmonic-number
NEXT i

FOR i = 1 TO nh-1
  FOR j=i+1 TO nh
    IF temp(1,i) < temp(1,j) THEN
       temp1=temp(1,i)
       temp2=temp(2,i)
       temp(1,i)=temp(1,j)
       temp(2,i)=temp(2,j)
       temp(1,j)=temp1
       temp(2,j)=temp2
     END IF
  NEXT j
NEXT i

FOR i=1 TO ne
  FOR j=1 TO 6
    Big10(j,i)=OutDat(j,temp(2,i))
  NEXT j
NEXT i

CLS

LOCATE 7,5
PRINT ne;"largest harmonics in descending order:"
PRINT #1,ne
LOCATE 9,5
PRINT "    k       alpha     beta        A        phi      power"
LOCATE  ,5
PRINT "--------- --------- --------- --------- --------- ---------"
FOR k=1 TO ne
  LOCATE 10+k,4
  FOR i=1 TO 6
    PRINT USING " ####.####";Big10(i,k);
    PRINT #1, USING " ####.####";Big10(i,k);
```

```
    NEXT i
    PRINT
    PRINT #1,
  NEXT k

  CALL WaitForKey(12!+ne)

  CLS

GenerateHarmonics:

  IF ne < 15 THEN
    ng = ne
  ELSE
    ng = 15
  END IF

  LOCATE MsgR,1
  PRINT SPACE$(78)
  LOCATE MsgR,MsgC
  PRINT "Do you wish to generate the";ng;"largest harmonics? ";

GenLoop:

  k$ = INKEY$
  IF k$ = "" THEN GOTO GenLoop:
  PRINT k$
  IF UCASE$(k$)="Y" THEN GOTO DoGeneration
  IF UCASE$(k$)="N" THEN
    CLOSE #1
    GOTO BailOut
  END IF

  GOTO GenerateHarmonics

DoGeneration:


  LOCATE MsgR,MsgC
  PRINT"Generating";ng;"largest harmonics"

  PRINT #1,ng      ' save the number of generated harmonics

  CLOSE #1

  FOR i=1 TO ng
    k=Big10(1,i)
     ClearRow(MsgR+2)
    LOCATE MsgR+2,MsgC
    PRINT "Performing calculations for harmonic-value";k
    amp = Big10(4,i)
    phi = Big10(5,i)
    CALL GenHarmonic(n,k,amp,phi,A0,HrmDat())
    IF k<10 THEN
      endstr$=".k0"+MID$(STR$(k),2)
    ELSE
      endstr$=".k"+MID$(STR$(k),2)
    END IF
    HrmFil$ = OutFil$+endstr$
    ClearRow(MsgR+2)
    LOCATE MsgR+2,MsgC
    PRINT "Sending results to ";HrmFil$
    OPEN HrmFil$ FOR OUTPUT AS #1
    FOR j=1 TO n
      PRINT #1,HrmDat(j)
```

```
    NEXT j
    CLOSE #1
  NEXT i

BailOut:

'$IGNORE ON        ' Interpreter explicitly closes full-size window
  SCREEN CLOSE 1
  WINDOW CLOSE 2
'$IGNORE OFF

  ERASE InpDat,OutDat
  CLEAR ,25000
  SYSTEM

END

'*******************************************************
'*******************************************************
'**                                                   **
'**                    ERROR TRAPS                    **
'**                                                   **
'*******************************************************
'*******************************************************

ErrorTraps:


' NOTE: Error trapping seems to close custom-windows. So any
'       that were in use have to be  reopened as required.

IF ERR = 53 THEN  ' file not found
  FileErr = FALSE
  RESUME NEXT
ELSEIF ERR = 55 THEN ' file already open
  CLOSE
  CLS
  BEEP
  LOCATE ErrR,ErrC
  COLOR 3,1
  PRINT "You tried to use an open input for output -- start over!"
  COLOR 1,0

'$IGNORE ON              ' if interpreter used then delay message on
  Dly& = TIMER           ' interpretrer-window before RESUME and reopen
  WHILE TIMER < Dly&+2   ' custom-window after RESUME line.
  WEND
'$IGNORE OFF

  RESUME InputName
ELSE
  ON ERROR GOTO 0
END IF

'*******************************************************
'*******************************************************
'**                                                   **
'**                   SUBPROGRAMS                     **
'**                                                   **
'*******************************************************
'*******************************************************

SUB NumericInput(V,P$,MC,MR,InF,DF,DV,MMF,Min,Max,EC,ER) STATIC

' Prevents AmigaBASIC error-traps on silly inputs so screen-format
```

```
' doesn't get screwed up.  Returns value in V.  Allows only intergers
' if InF is TRUE.  A default output on carriage-return (DV) is allowed
' if DF is TRUE.  A maximum and minimum range is included in the error
' checks (Min =< V =< Max) if MMF is TRUE.  Leading spaces are stripped
' but spaces anywhere else are considered an error.  Exponentials (e.g.,
' -2.345E-67) allowed in single or double-precision (E or D) format.
' MC,MR are the column and row for promt-message P$.  EC,ER are the
' column and row for error-messages.

  SHARED TRUE,FALSE

StartNumInput:

  CALL ClearRow(MR+1)
  IF DF=TRUE THEN
    LOCATE MR+1,MC
    PRINT "( Default =";DV;")"
  END IF
  IF MMF=TRUE THEN
    LOCATE MR+1,MC
    PRINT "( Range =";Min;"TO";Max;")"
  END IF
  IF DF=TRUE AND MMF=TRUE THEN
    LOCATE MR+1,MC
    PRINT "( Range =";Min;"TO";Max;" -- Default =";DV;")"
  END IF

  CALL ClearRow(MR)
  LOCATE MR,MC
  PRINT P$;
  LINE INPUT; V$

  IF LEN(V$) = 0 THEN  ' carriage-return is an error unless defaults active
    IF DF=TRUE THEN
      V = DV
      GOTO NumInputOver
    END IF
    GOTO InputError
  END IF

  FOR i=1 TO LEN(V$)                              ' strip leading spaces
    IF MID$(V$,i,1)<>" " THEN GOTO StripOver
    V$ = MID$(V$,i+1 )
  NEXT i

  GOTO InputError                                 ' all spaces is an error

StripOver:

  NumStr$ = "0123456789eEdD+-"                    ' acceptable input characters

  FOR i=1 TO LEN(V$)
    IF INSTR(NumStr$,MID$(V$,i,1))=0 THEN GOTO InputError
  NEXT i

  V = VAL(V$)

  IF InF=TRUE THEN
    IF V<>INT(V) THEN GOTO IntegerError
  END IF

  IF MMF=TRUE THEN
    IF V<Min OR V>Max THEN GOTO RangeError
  END IF
```

```
NumInputOver:

  CALL ClearRow(ER)
  CALL ClearRow(MR)
  CALL ClearRow(MR+1)
  EXIT SUB

InputError:

  CALL ClearRow(ER)
  CALL ErrorMsg("That is not an acceptable input!",EC,ER)
  GOTO StartNumInput

IntegerError:

  CALL ClearRow(ER)
  CALL ErrorMsg("Please enter an INTEGER value!",EC,ER)
  GOTO StartNumInput

RangeError:

  CALL ClearRow(ER)
  ErrMsg$ = "Enter a value from"+STR$(Min)+"to"+STR$(Max)+"!"
  CALL ErrorMsg(ErrMsg$,EC,ER)
  GOTO StartNumInput

END SUB

'*****************************************************

SUB OldFile(FileName$,FileErr) STATIC

' Sets FileErr to FALSE then checks if file is present by
' attempting to open for input.  If present, then opens,
' closes, exits.   If not, get error.  Error trap sets
' FileErr FALSE, then RESUME NEXT in the error trap returs
' to the line after the call.  Return of FileErr FALSE means
' that the file does not exist, so it is safe to open
' for output.  FileErr = TRUE means that the file exists so
' check whats going on. The name 'FileErr' in this subroutine
' is not optional due to use of error-trapping.

  SHARED TRUE,FALSE

  FileErr = TRUE

  OPEN FileName$ FOR INPUT AS #99
  CLOSE #99

END SUB

'*****************************************************

SUB ErrorMsg(ErrMsg$,MsgCol,MsgRow) STATIC

  CLS
  BEEP
  LOCATE MsgRow,MsgCol
  COLOR 3,1
  PRINT ErrMsg$
  COLOR 1,0

END SUB

'*****************************************************
```

```
SUB FileRead(InpFil$,FileRecs(),Count) STATIC

' Read in the rows of a file as character strings and count.
' Skips zero-length lines and lines that start with '*'

  Count = 0

  OPEN InpFil$ FOR INPUT AS #99 LEN=16384

  WHILE NOT(EOF(99))
    LINE INPUT #99,R$
    IF MID$(R$,1,1)="*" THEN SkipRow
    IF LEN(R$)=0 THEN SkipRow
    Count = Count+1
    FileRecs(Count)=VAL(R$)

SkipRow:

  WEND

  CLOSE #99

END SUB

'*****************************************************

SUB ClearRow(Row) STATIC

' Erease specified row on screen.

    LOCATE Row,1
    PRINT SPACE$(78)

END SUB


'*****************************************************

SUB GenHarmonic(n,k,amp,phi,A0,HrmDat()) STATIC

' generate a sinusoid where n is the number of points in the record,
' k is the number of cycles in the record, amp is the amplitude,
' phi is the phase-angle, and A0 the offset.
' Results are returned in CosDat().

  pi = 3.141529

  T = n/k ' points in a period = number-of-points/harmonic-number

  FOR i=1 TO n
    HrmDat(i) = A0 + amp*COS(2*pi*(i-1)/T - phi)
  NEXT i

END SUB

'*****************************************************

SUB WaitForKey(MsgLine) STATIC

' Puts a message MsgLine to wait for a  keystroke, then does so.

  PRINT
  COLOR 3,1
  LOCATE MsgLine,1
```

```
  PRINT "Hit any key to continue."
  COLOR 1,0

WaitForKeyLoop:

  IF INKEY$ = "" THEN GOTO WaitForKeyLoop

  CALL ClearRow(MsgLine)

END SUB

'*****************************************************
```

PADGAPS.BAS

Purpose: data-gap padding via linear interpolation

Language: GW-BASIC

```
 10 REM This program averages values on either side of a data gap in
 20 REM a thesis column file and inserts averaged values into the gap.
 30 REM This version is for gaps less than 24 hours long.
 40 DIM COLDAT(5828)
 50 INPUT "    Enter column file name: ",FILNME$
 60 INPUT "     Enter gap start index: ",GSIDX
 70 INPUT "       Enter gap end index: ",GEIDX
 80 INPUT "Enter file name for output: ",OUTFIL$
 90 PRINT "Reading ";FILNME$;" into array storage."
100 OPEN FILNME$ FOR INPUT AS #1
110 FOR I=1 TO 5828
120   INPUT #1,COLDAT(I)
130 NEXT I
140 CLOSE #1
150 PRINT "Calculating and inserting pad values."
160 REM There are 96 points from 0000 to 2345 at 15 minute increments
170 FOR I = GSIDX TO GEIDX
180   COLDAT(I) = (COLDAT(I-96)+COLDAT(I+96))/2
190 NEXT I
200 PRINT "Writing padded column data to ";OUTFIL$
210 OPEN OUTFIL$ FOR OUTPUT AS #1
220 FOR I=1 TO 5828
230   IF COLDAT(I) > 100 THEN PRINT #1, USING " #####";COLDAT(I)
235   IF COLDAT(I) < 100 THEN PRINT #1, USING " ##.##";COLDAT(I)
240 NEXT I
250 CLOSE #1
260 PRINT "DONE!"
270 END
```

PADGAP11.BAS

Purpose: data-gap padding via linear interpolation

Language: GW-BASIC

```
 10 REM This program pads gap1 in thesis columns up from 69 @ 0930 to
 20 REM 69 @ 1030 by extrapolating from days 67 and 68.
 30 INPUT "Enter column file name: ",FILNME$
 40 PRINT "Padding ";FILNME$;" from 69 @ 0930 to 69 @ 1030."
 50 DIM COLDAT(6000)
 60 OPEN FILNME$ FOR INPUT AS #1
 70 FOR I=1 TO 5828
 80   INPUT #1,COLDAT(I)
 90 NEXT I
100 CLOSE #1
110 FOR I = 1218 TO 1222
120  COLDAT(I+192) = 2*(COLDAT(I+96)-COLDAT(I)) +COLDAT(I)
130 NEXT I
140 PRINT "Writing padded data back to ";FILNME$
150 OPEN FILNME$ FOR OUTPUT AS #1
160 FOR I=1 TO 5828
170   IF COLDAT(I) > 100 THEN PRINT #1, USING " #####";COLDAT(I)
175   IF COLDAT(I) < 100 THEN PRINT #1, USING " ##.##";COLDAT(I)
180 NEXT I
190 CLOSE #1
200 PRINT"DONE!"
210 END
```

PADGAP12.BAS

Purpose: data-gap padding via linear interpolation

Language: GW-BASIC

```
 10 REM This program calculates and removes the means from the leading and
 20 REM days of gap1 in thesis data columns after PADGAP11.BAS has been run
 30 REM over the columns. It then interpolates the gap data inbetween the
 40 REM meanless portions of the record.  After this the apparent slope is
 50 REM put into the interpolated part of the record and these results
 60 REM are returned to the data column.
 50 DIM COLDAT(6000)
 60 INPUT "Enter column file name: ",FILNME$
 70 PRINT "Reading ";FILNME$
 80 OPEN FILNME$ FOR INPUT AS #1
 90 FOR I=1 TO 5828
100   INPUT#1,COLDAT(I)
110 NEXT I
120 CLOSE #1
130 PRINT "Performing statistics"
140 MAX1 = -99999
150 MIN1 = 99999
160 MAX2 = -99999
170 MIN2 = 99999
180 MEAN1=0
190 MEAN2=0
200 FOR I=1319 TO 1414
210  IF MAX1 < COLDAT(I) THEN MAX1 = COLDAT(I)
220  IF MIN1 > COLDAT(I) THEN MIN1 = COLDAT(I)
230  MEAN1 = MEAN1 + COLDAT(I)
240 NEXT I
250 FOR I=1703 TO 1798
260  IF MAX2 < COLDAT(I) THEN MAX2 = COLDAT(I)
270  IF MIN2 > COLDAT(I) THEN MIN2 = COLDAT(I)
280  MEAN2 = MEAN2 + COLDAT(I)
290 NEXT I
300 MEAN1 = MEAN1/96
310 MEAN2 = MEAN2/96
320 PRINT "Removing means"
330 FOR I = 1319 TO 1414
340   COLDAT(I) = COLDAT(I) - MEAN1
350   COLDAT(I+384) = COLDAT(I+384) - MEAN2
360 NEXT I
370 PRINT "Performing interpolations"
380 FOR I = 1319 TO 1414
390   COLDAT(I+ 96) = (1/4)*(COLDAT(I+384) - COLDAT(I)) + COLDAT(I)
400   COLDAT(I+192) = (2/4)*(COLDAT(I+384) - COLDAT(I)) + COLDAT(I)
410   COLDAT(I+288) = (3/4)*(COLDAT(I+384) - COLDAT(I)) + COLDAT(I)
420 NEXT I
430 PRINT "Restoring slope"
440 REM The first interpolated point is set to the same mean level as
450 REM the leading day.  The last interpolated point is set to the
460 REM same mean level as the trailing day. The slope between these
470 REM is applied to all points inbetween.
480 FOR I = 1319 TO 1414
490   COLDAT(I) = COLDAT(I) + MEAN1
500   COLDAT(I+384) = COLDAT(I+384) + MEAN2
510 NEXT I
520 M = (MEAN1 - MEAN2)/(1415 - 1702)
530 B = MEAN2 - M*1702
540 FOR I = 1415 TO 1702
550   COLDAT(I) = COLDAT(I) + (I*M + B)
560 NEXT I
580 PRINT "Writing padded data back to ";FILNME$
590 OPEN FILNME$ FOR OUTPUT AS #1
600 FOR I = 1 TO 5828
610   IF COLDAT(I) > 100 THEN PRINT #1,USING " #####";COLDAT(I)
615   IF COLDAT(I) < 100 THEN PRINT #1,USING " ##.##";COLDAT(I)
620 NEXT I
630 CLOSE #1
```

```
640 PRINT "DONE!"
650 END
```

PADGAP2.BAS

Purpose: data-gap padding via linear interpolation

Language: GW-BASIC

```
 10 REM This program is a fixed position version of PADGAPS.BAS for padding
 20 REM averages values on either side of data gap 2, which is slightly greater
 30 REM than 24 hours in a thesis data column.
 40 DIM COLDAT(5828)
 50 INPUT "    Enter column file name: ",FILNME$
 80 INPUT "Enter file name for output: ",OUTFIL$
 90 PRINT "Reading ";FILNME$;" into array storage."
100 OPEN FILNME$ FOR INPUT AS #1
110 FOR I=1 TO 5828
120   INPUT #1,COLDAT(I)
130 NEXT I
140 CLOSE #1
150 PRINT "Calculating and inserting pad values."
155 REM Data gap 2 is from day 74.5625 (1906) to day 75.7917 (2024)
160 REM There are 284 points from day 72.5625 (1714) to day 76.5625 (2098)
165 REM There are 192 points from day 72.5625 (1714) to day 74.5625 (1906)
170 FOR I = 1714 TO 1832
180   COLDAT(I+192) = (COLDAT(I)+COLDAT(I+384))/2
190 NEXT I
200 PRINT "Writing padded column data to ";OUTFIL$
210 OPEN OUTFIL$ FOR OUTPUT AS #1
220 FOR I=1 TO 5828
230   IF COLDAT(I) > 100 THEN PRINT #1, USING " #####";COLDAT(I)
235   IF COLDAT(I) < 100 THEN PRINT #1, USING " ##.##";COLDAT(I)
240 NEXT I
250 CLOSE #1
260 PRINT "DONE!"
270 END
```

PRNCMP.M

Purpose: Principle-components calculation (called by FACAN.M)

Language: PC-MatLab .M script file

```
function [cc,cvf,fload,score,eval,evct,sim,cd,n,pv,sv,DS,SD,ISD,LM] = prncmp(X)
%
% prncmp(X) takes in a raw data matrix, X, and performs a principle component
% analysis on X.  The percent variance explained by the components is a
% run time user option input.  The complete return list for prncmp(X) is:
% [cc, cvf, fload, score, eval, evct, simmtx, n, pv, sv, DS, SD, ISD, LM].
% See facan.m documentation for more details. (Alan Swithenbank, January 1988)
%
% output of diagonal elements of covariance matrix added July 1988 (AMS)
%
% use of correlation matrix included September 1989 (AMS)
% (corrected to use NORMALIZED data in this case March 1990 (AMS))
%
% return standard-deviations and inv(diag(std-devs)) March 1990 (AMS)
%
% ------------------------------------------------------------------------
%
  [r,c]=size(X); % get number of rows and columns
%
  DS=X-ones(1:r)'*mean(X); % form deviate scores matrix
%
  cc = 0;
%
  while (cc ~= 1) & (cc ~= 2)
    cc=input('enter 1 to use covariance, enter 2 to use correlation ');
  end
%
% note: a lot of the covariance stuff gets done to the correlation
%       matrix just to make life easier.

%
% generate covariance matrix regardless of selected similarity
%
% (do it this way instead of with cov() function so can use unbiased weight)
%
   m=r;
   if r<11  % if r "small" then used "unbiased" weight (1/(r-1)) for variance
     m=m-1;
   end
%
   N=1/m*ones(c);

   covmtx=N.*(DS'*DS);
%
  if cc == 1 %
    simmtx = covmtx;
  end
  if cc == 2 % generate correlation-matrix
    simmtx = corrcoef(X);
  end
%
  simmtx  % goes to screen after calculations
%
  cd=diag(simmtx); % get the cov(j,j) into a column vector
%
  [evt,evl]=eig(simmtx); % get the eigenvectors and eigenvalues
  evct=revcol(evt); % then reverse columns to get biggest ones first
  eval=revcol(evl)  % goes to screen on calculation
%
  pctvar=input('what minimum percent variance should the factors explain? ');
%
  e=ones(c)*eval; % put eigenvalues into column vector from the diagonal matrix
  ceval=diag(e);  % eval. ceval has elements from lower left to upper right
%
  seval=sum(ceval); % sum the eigenvalues
```

```
%
  varsum=0;
  count=0;
  while varsum<pctvar  % count up variance explained until pass prescribed limit
    count=count+1;
    varsum=varsum+100*ceval(count)./seval;
  end
%
  n=count;    % make shorter names to fit in function call line
  pv=varsum;
  sv=seval;
  sim=simmtx;
%
  !cls
%
  fprintf('\n%4.0f factors retained\n',n)
  fprintf('explaining %8.4f percent of the variance\n',pv)
%
  L=ceval(1:n); % get the retained eigenvalues
  U=evct(:,1:n); % get the retained eigenvectors
%
% get the factor loadings (type 1) per pg. 69 Geological Factor Analysis
%
  Lsqrt=sqrt(L)'; % get the square root of the eigenvalues as a row vector
  Lsqrm=ones(1,c)'*Lsqrt; % repeat into matrix for element by element operations
%
  fload=U.*Lsqrm;
%
% get the factor scores (type 1) per pg. 69 Geological Factor Analysis
%
  Linv=ones(1:n)./L'; % get the inverse of the eigenvalues as a row vector
  LM=ones(1:r)'*Linv; % repeat into a matrix for multiplication
%
% use unnormalized DS if using covariance-matrix (keep amplitude info)
% use normalized DS=Z if using correlation-matrix (lose amplitude info)
% where Z=(DS)(D**(-1/2)) where D is a diagonal-matrix of diagonal-elements
% of covarinace matrix
%
  if cc == 2
%
%   generate D**(-1/2)
%
    D=sqrt(diag(covmtx))';
    D=ones(1:c)'*D;
    I=eye(covmtx);
    D=I.*D;
    D=inv(D); % for diagonal matrix, inv(X) = X**(-1)
%
%   normalize DS
%
    DS = DS*D;
%
  end
%
  score=DS*fload.*LM;
%
% get the variable vs. factor correlations matrix for type 1 analysis
%
  SD=std(DS); % the std function has been modified for (1/(n-1)) if r small
  sddiag=diag(SD,0); % put the variable std. devs. into a diag matrix
  ISD=inv(sddiag); % for diag matrix inv is just element^(-1) for each element
%
  cvf=ISD*fload;
%
  fprintf('\nThe correlations of the factors with the variables are:\n\n')
```

```
  fprintf('           factor 1 ... factor n\n')
  fprintf('\nvariable 1    r11  . . .   r1n\n    .\n    .\n    .\n')
  fprintf('variable n    rn1  . . .   rnn\n')
%
  cvf  %  goes to screen even when use prncmp();
```

VARIMAX.BAS

Purpose: normalized varimax factor-rotation (called by FACAN.M)

Language: GW-BASIC

```
' Program file: VARIMAX.BAS  -- Alan Swithenbank, January 1988
'
' This program is a BASIC routine for doing a
' normalized varimax rotation of a factor loadings matrix.  It is based
' on the "verbal pseudocode" of Henry F. Kaiser (1959, Educational and
' Psychological Measurement, 19(3):413-420), but has the recomendation of
' B. Wingersky (see: H.H. Harman,1976. Modern Factor Analysis. pg. 294)
' implemented to determine the sine and cosines for the rotations. The
' variable names (attempt) to reflect the descriptions used in Kaiser's
' paper.  It should be noted that this paper was written at a time when
' computer programming required a more intimate knowledge on the part of
' the programmer of the inner workings of their computer than is expected
' today.  The references to base, size of p, etc., can be ignored as the
' overflow prevention scaling scheme described in Kaiser has been
' eliminated in this program.  Note that what Kaiser calls tests are
' variables here (tests refers to psychological tests).
'
' modified for communality output -- Alan Swithenbank, July 1988
'
' changed to double precision -- Alan Swithenbank, September 1988
'
DEFDBL A-H,L-Z : ' everything but I,J,K variables are double precision
PRINT : PRINT "doing normalized varimax rotation"
DIM LM(30,32) ' loading matrix 30 variables (rows) 30 factors (columns) max
'
' the extra columns hold communalities and sqrt(sum(a(i,j)**2))'s
'
DIM TM(30,30) ' transform matrix (generated as rotate loading matrix)
DIM EV(30,30) ' eigenvector matrix (will be rotated along with loadings)
EPSILON = 0.0005
'
' read in size of unrotated factor loadings matrix
'
OPEN "MATSIZ.DAT" FOR INPUT AS #1 ' file MATSIZ.DAT generated in PCMATLAB
INPUT #1, NUMVAR, NUMFAC
CLOSE #1
OPEN "MATSIZ.DAT" FOR INPUT AS #1
INPUT #1, INUMVAR, INUMFAC
CLOSE #1
'
' generate TM as an identity matrix to start
'
FOR I = 1 TO INUMFAC
  FOR J = 1 TO INUMFAC
    TM(I,J) = 0
    IF (I=J) THEN TM(I,J) = 1
  NEXT J
NEXT I
'
' read in the unrotated factor loadings matrix
'
OPEN "LOADING.DAT" FOR INPUT AS #1 ' file LOADING.DAT generated in PCMATLAB
FOR I = 1 TO INUMVAR
  FOR J = 1 TO INUMFAC
    INPUT #1, LM(I,J)
  NEXT J
NEXT I
CLOSE #1
'
' calculate communalities and sqrt(sum(a(i,j)**2)), where:
' communality(j) = sum(a(i,j)**2/covariance(j,j)), i = 1 to numvars
'
OPEN "SIMDIAG.DAT" FOR INPUT AS #1
OPEN "COMUNAL.DAT" FOR OUTPUT AS #2
FOR I = 1 TO INUMVAR
```

```
223 INPUT #1, COVJJ
230   HJSQR = 0
240   FOR J = 1 TO INUMFAC
250     HJSQR = HJSQR + LM(I,J)^2 ' sum squares of row elements
260   NEXT J
270   LM(I,31) = HJSQR/COVJJ : PRINT #2, HJSQR/COVJJ ' communality
280   LM(I,32) = SQR(HJSQR)                           ' sqrt(sum(a(i,j)**2))
290 NEXT I
291 CLOSE #1
292 CLOSE #2
307 '
308 ' normalize the loadings
309 '
310 FOR I = 1 TO INUMVAR
320   FOR J = 1 TO INUMFAC
330     LM(I,J) = LM(I,J)/LM(I,32)
340   NEXT J
350 NEXT I
357 '
358 ' read in the unrotated eigenvectors
359 '
360 OPEN "EVCT.DAT" FOR INPUT AS #1 ' file EVCT.DAT generated in PCMATLAB
370 FOR I = 1 TO INUMVAR
380  FOR J = 1 TO INUMVAR
390   INPUT #1,EV(I,J)
400  NEXT J
410 NEXT I
420 CLOSE #1
428 '
429 '
430 INUMROT = INUMFAC*(INUMFAC-1)/2 ' rot. count, reentry point (see Kaiser)
434 '
435 ' for each pair of factors successively generate A,B,C,D
436 ' note the factor comparison sequence, e.g., if 4 factors then:
437 ' 1 -> 2, 1 -> 3, 1 -> 4; 2 -> 3, 2 -> 4; 3 -> 4 -- then repeat
438 ' (see Kaiser)
439 '
440 FOR IFRSTFAC = 1 TO INUMFAC-1
450   FOR ISCNDFAC = IFRSTFAC+1 TO INUMFAC
460     A=0 : B=0 : C=0 : D=0
470     FOR K = 1 TO INUMVAR
480       AK1 = LM(K,IFRSTFAC) ' loading on variable K for first factor rotated
490       AK2 = LM(K,ISCNDFAC) ' variable K loading for second factor rotating
500       U = (AK1 + AK2)*(AK1 - AK2)
510       V = AK1*AK2 : V = V + V
520       U2MV2 = (U + V)*(U - V)
530       UV2 = U*V : UV2 = UV2 + UV2
540       A = A + U
550       B = B + V
560       C = C + U2MV2
570       D = D + UV2
580     NEXT K
587 '
588 ' rotate the current factor loadings pair (see: Kaiser; see: Harman)
589 '
590     NUM = D - 2*(A*B)/NUMVAR
600     DEN = C - (A*A-B*B)/NUMVAR
607 '
608 ' this is where the switch to Wingersky recommendation in Harmon is made
609 '
610     G = SQR(NUM*NUM + DEN*DEN)
615 '
620     IF (G=0) THEN 740 ' no rotation required
625  'else
630         COS4PHI = DEN/G
```

```
640         COS2PHI = SQR((1+COS4PHI)/2)
650          COSPHI = SQR((1+COS2PHI)/2)
660       ABSSINPHI = SQR((1-COS2PHI)/2)
670       IF (ABSSINPHI<EPSILON) THEN 740
675 '
680     IF (NUM>=0) THEN 710
685  'else
690       SINPHI = -ABSSINPHI
700       GOTO 720
705  'then
710       SINPHI =  ABSSINPHI
715 '
720     GOSUB 2000 ' rotate current factor pair
730     GOTO 750 ' get next pair to rotate
735 '
740     INUMROT = INUMROT - 1 '  there was no rotation for this pair
750   NEXT ISCNDFAC
760 NEXT IFRSTFAC
765 '
770 IF INUMROT <> 0 THEN 430 ' have not converged yet so do again
776 '
777 ' converged, so denormalize and return varimax loadings, communalities,
778 ' eigenvectors, and the transform matrix -- NOTE: PC-Matlab chokes
779 ' on D format exponential inputs, so final output is converted to single
780 ' single precision so will get E format instead
781 '
800 OPEN "ROTLOAD.DAT" FOR OUTPUT AS #1
810 OPEN "ROTCMNL.DAT" FOR OUTPUT AS #2
820 OPEN "TRNSFRM.DAT" FOR OUTPUT AS #3
840 OPEN "ROTEVCT.DAT" FOR OUTPUT AS #4
841 OPEN "SIMDIAG.DAT" FOR INPUT  AS #5
850 FOR I = 1 TO INUMVAR
851 INPUT #5, COVJJ
860   HJSQR = 0
870   FOR J = 1 TO INUMFAC
880     NEWLOD =  LM(I,J)*LM(I,32)
890     PRINT #1, CSNG(NEWLOD);
910     HJSQR = HJSQR + NEWLOD*NEWLOD
920     PRINT #4, CSNG(EV(I,J));
930   NEXT J
940   PRINT #1,
950   PRINT #2, CSNG(HJSQR/COVJJ)
960   PRINT #4,
970 NEXT I
980 FOR I = 1 TO INUMFAC
990   FOR J = 1 TO INUMFAC
1000    PRINT #3, CSNG(TM(I,J));
1010  NEXT J
1020  PRINT #3,
1030 NEXT I
1040 CLOSE
1050 PRINT : PRINT "done with rotation !"
1060 SYSTEM
1994 '
1995 ' ************************************************************************
1996 ' ************************************************************************
1997 '
1998 ' subroutine to rotate factor pairs
1999 '
2000 INUMROT = INUMFAC*(INUMFAC-1)/2
2007 '
2008 ' rotate appropriate columns of loadings matrix
2009 '
2010 FOR I = 1 TO INUMVAR
2020   LM1 = LM(I,IFRSTFAC) : LM2 = LM(I,ISCNDFAC)
```

```
2030   LM(I,IFRSTFAC) = LM1*COSPHI + LM2*SINPHI
2040   LM(I,ISCNDFAC) = LM2*COSPHI - SINPHI*LM1
2050 NEXT I
2057 '
2058 ' rotate appropriate columns of transform martix
2059 '
2060 FOR I = 1 TO INUMFAC
2070   TM1 = TM(I,IFRSTFAC) : TM2 = TM(I,ISCNDFAC)
2080   TM(I,IFRSTFAC) = TM1*COSPHI + TM2*SINPHI
2090   TM(I,ISCNDFAC) = TM2*COSPHI - TM1*SINPHI
2100 NEXT I
2103 '
2104 ' rotate appropriate columns of eigenvector matrix
2105 '
2106 ' NOTE: only the eigenvectors that relate to the retained eigenvalues
2107 ' will be rotated, although all the eigenvectors will be returned by
2108 ' this program in the rotated eigenvector files
2109 '
2110 FOR I = 1 TO INUMVAR
2120   EV1 = EV(I,IFRSTFAC) : EV2 = EV(I,ISCNDFAC)
2130   EV(I,IFRSTFAC) = EV1*COSPHI + EV2*SINPHI
2140   EV(I,ISCNDFAC) = EV2*COSPHI - EV1*SINPHI
2150 NEXT I
2155 '
2160 RETURN
2161 '
2162 ' ************************************************************************
2163 ' ************************************************************************
2164 '
```

9PTFLTR.BAS

Purpose: 9-point quadratic data-decimation filter

Language: GW-BASIC

```
 1 REM This program runs a 9 point weighted-average filter through a column file
 2 REM of 15 minute interval data, producing filtered 1 hour interval data.
 3 REM This is a least-squares, quadradic-fit, symmetric weighted filter, with the filter
 4 REM weights summing to 1. E.g., for a 17 point series with values:
 5 REM
 6 REM                a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q
 7 REM
 8 REM the output series contains 3 points which will be:
 9 REM
10 REM (-21*a + 14*b + 39*c + 54*d + 59*e + 55*f + 39*g + 14*h - 21*i)/231 @ e
11 REM (-21*e + 14*f + 39*g + 54*h + 59*i + 55*j + 39*k + 14*l - 21*m)/231 @ i
12 REM (-21*i + 14*j + 39*k + 54*l + 59*m + 55*n + 39*o + 14*p - 21*q)/231 @ m
13 REM
14 REM This was originally written in AmigaBASIC as a binomially-weighted
15 REM filter and ported to GWBASIC -- then convereted to least-squares.
16 REM
20 DIM F(6000)
30 LOCATE 5,5
40 INPUT "Enter input column file name: ",InpFile$
50 LOCATE 7,5
60 INPUT "Enter output column file name: ",OutFile$
70 OPEN InpFile$ FOR INPUT AS #1
80 OPEN OutFile$ FOR OUTPUT AS #2
90 CLS
100 LOCATE 5,5
110 PRINT "Reading ";InpFile$
120 ptcnt = 0
130 IF EOF(1) THEN GOTO 180
140   LINE INPUT #1, V$
150   F(ptcnt) = VAL(V$)
160   ptcnt = ptcnt +1
170 GOTO 130
180 LOCATE 7,5
190 PRINT InpFile$;" has";ptcnt;"records"
200 LOCATE 9,5
210 PRINT "Running filter to decimate data"
220 outcnt = 0
230 FOR i=1 TO ptcnt STEP 4
240  IF i+8 > ptcnt THEN GOTO 300
250  V = -21*F(i) + 14*F(i+1) + 39*F(i+2) + 54*F(i+3) + 59*F(i+4)
260  V = (V + 54*F(i+5) +39*F(i+6) + 14*F(i+7) - 21*F(i+8))/231
270  PRINT #2,V
280  outcnt = outcnt + 1
290 NEXT i
300 CLOSE #1
310 CLOSE #2
320 LOCATE 11,5
330 PRINT "Done! -- ";OutFile$;" has";outcnt;"records"
340 LOCATE 15,1
350 END
```