Simulation of Process Systems - A PhD Course

Nilsson, Bernt

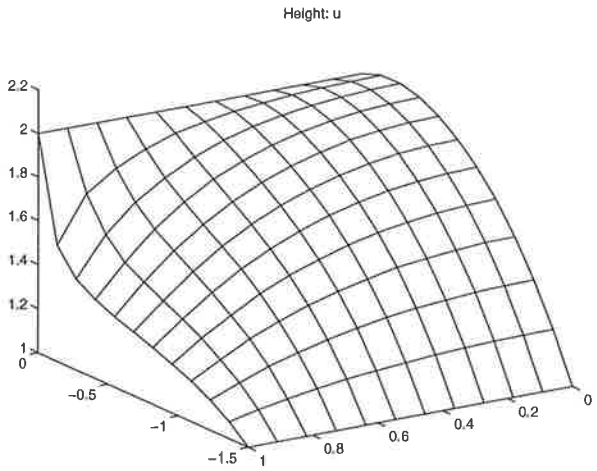
1996

Document Version:
Publisher's PDF, also known as Version of record

ISSN 0280–5316
ISRN LUTFD2/TFRT--7550--SE

Simulation of Process Systems — a PhD course

Bernt Nilsson



Department of Automatic Control
Lund Institute of Technology
August 1996

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 Lund Sweden

Document name
TECHNICAL REPORT

Date of issue
August 1996

Document Number
ISRN LUTFD2/TFRT--7550--SE

Author(s)
Bernt Nilsson

Supervisor

Sponsoring organisation

Title and subtitle
Simulation of Process Systems – a PhD course

Abstract

This report contains a set of lecture notes of a PhD course in Simulation of Process Systems for primary PhD students in Chemical Engineering. The course gives an overview of steady-state and unsteady-state simulation methods for lumped and distributed parameter systems. The lectures covers following areas: Linear and nonlinear algebraic equation systems, initial-value problems for ordinary differential equations, boundary-value problems for ordinary differential equations based on discrete variables and finite elements, simulation of parabolic and hyperbolic partial differential equations in one space dimension, partial differential equations in more then one space dimension. 5 computer exercises cover the material using MATLAB with ODE Suite and PDE toolbox.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
71

Recipient's notes

Security classification

Simulation of Process Systems

PhD course at Department of Chemical Engineering II
april and may1996.

Aim

The course gives an *overview* of common simulation methods in process engineering. The simulation methods are introduced and exemplified on chemical engineering problems. Examples are dynamic and static simulation problems as well as problems with lumped and distributed space descriptions. Mathematically this means algebraic equations (AE), ordinary and partial differential equations (ODE and PDE) both in initial value problem, IVP, and boundary value problem, BVP. The benefits and drawbacks of the methods are briefly discussed. Different classes of methods are tested on chemical engineering problems using **MATLAB** with ODE Suite and PDE-toolbox.

Lectures:

1. **Introduction to Simulation of Process Systems,** notes (pdf-format)
   `10.15 Wed. 10/4, K:I`
   - Process Modelling, Problem Formulations and Problem Solving Methods,
   - Simulation: History and State-of-the-Art
   - Computer Aided Process Engineering (CAPE) and Software Tools
2. **Steady-State Simulation of Algebraic Equations (AE),** notes (pdf-format)
   `10.15 Mon. 15/4, K:N`
   - Linear Equation Systems - LU decomposition and iteration methods
   - Sparseness
   - Stability of difference equations
   - Nonlinear Equation Systems - Fix-point and Newton methods
   - ***To read:*** *Fröberg, chap 10 and 11, Davis, app C, E and B*
3. **Dynamic Simulation of Ordinary Differential Equations (ODE)** (notes not available here)
   `10.15 Mon. 22/4, K:N`
   - Initial Value Problem (IVP) for ODEs.
   - Explicit and implicit solvers
   - One step and multistep methods
   - Stiff ODEs and Differential-Algebraic Equations (DAEs)
   - the MATLAB ODE Suite
   - ***To read:*** *Davis, chap 1*
4. **Steady-State Simulation of ODEs I - Discrete Variables** notes (pdf-format)
   `10.15 Mon. 29/4, K:N`
   - Boundary Value Problem (BVP) for ODEs.
   - Shooting Methods
   - Finite Difference Method (FDM)
   - Spacial Differentiation and Boundary Condition Approximations
   - ***To read:*** *Davis, chap 2*
5. **Steady-State Simulation of ODEs II - Finite Elements** notes (pdf-format)
   `10.15 Mon. 6/5, K:N`
   - Finite Element Method (FEM) and Methods of Weighted Residuals (MWR)
   - Piecewise Polynomials and B-splines
   - Galerkin and Collocation methods

- ***To read:*** *Davis, chap 3 and app D*

6. **Dynamic Simulation of PDEs - Method of Lines** notes (pdf-format)
   `10.15 Mon. 13/5, K:I`
   - Classification of PDEs.
   - Method of Lines (MOL) for the Heat Equation (parabolic PDE) in one space dimension
   - Finite Difference based MOL
   - Finite Element based MOL
   - Simulation of the Wave Equation (hyperbolic PDE)
   - MATLAB, ode suite, PDE-toolbox
   - ***To read:*** *Davis, chap 4*
7. **Simulation of PDEs in Higher Dimensions** notes (pdf-format)
   `10.15 Mon. 20/5, K:K`
   - PDE:s in 2 (and 3 space) dimensions
   - Steady state PDEs (Elliptic PDE)
   - Finite Differences and Boundary Conditions
   - Finite Element Methods
   - Irregular Boundaries
   - MATLAB PDE-toolbox
   - ***To read:*** *Davis, chap 5*

Lecture Problems:

1. **Methods for AE and ODE Systems** notes (pdf-format)
   - Methods for algebraic equations
   - Initial value problem methods for ODE:s (Davis chap 1)
   - Boundary value problem methods for ODE:s (Davis chap 2)
   - *Hand-in 13/5*
2. **Methods for PDE Systems** notes (pdf-format)
   - Finite element methods (Davis chap 3)
   - Method of lines for parabolic PDEs (Davis chap 4)
   - Elliptic and other PDE problems (Davis chap 5)
   - *Hand-in 31/5*

Computer Exercises:

The exercises can be done in the computer rooms on freday 8.30 (exercises 1-3 and 5).
The exercises can also be done at *home* and they require MATLAB 4 with ODE Suite. Some problems may require additional software.

1. **Steady-State Simulation of AEs** notes (pdf-format)
   `Fre. 19/4, Babaorum`
   - Linear equations - LU-decomposition, pivoting, iteration methods
   - Nonlinear equations - Newton methods.
   - additional functions: `fsolve` (optimization toolbox), `gseid`(Mathews, chap3)
2. **Dynamic Simulation of ODEs** (notes not available here)
   `Fre. 26/4, Babaorum`
   - Dynamic simulation with MATLAB
   - Explicit and implicit solver behaviour
   - non-stiff and stiff problems
   - Solvers in ODE Suite
   - additional functions: `eulers, rk4`(Mathews, chap9)

3. **Steady-State Simulation of ODEs** notes (pdf-format)
   `Fre. 3/5, Babaorum`
   - Shooting Methods
   - Finite Difference and boundary conditions
   - Sparse equation solving and sparse storage
   - MATLAB, sparse matrix technique
4. **Dynamic Simulation of PDE using FD based MOL** notes (pdf-format)
   `Wed. 15/5,` **`Lutetia`**
   - Finite difference and Method of Lines
   - Parabolic PDEs in one space dimension
   - Sparse ODEs
   - MATLAB with ODE Suite
   - additional functions: `finedif, forwdif, crnich` (Mathews, chap10)
5. **Simulation of PDEs using FEM** notes (pdf-format)
   `Fre. 24/5, Babaorum`
   - Finite Element Method - Galerkin
   - Static and Dynamic problems
   - MATLAB - PDE toolbox

Literature:

- **Davis (1984),** ***Numerical Methods and Modeling for Chemical Engineers***, Wiley
- **Fröberg (1985),** ***Numerical Mathematics, chap 10 and 11*** , Benjamin/Cummings
- A short bibliography

Examination:

1. *Passive:* the lectures and hand-in on the lecture problems (2 marks).
2. *Active:* 1 **and** hand-in on a selection (3) of the computer exercises (5 (2+3) marks).
3. *Research interested:* 2 **and one** theoretical seminar and **one** minor project (7 (2+3+2) marks).

Process Simulation extension (part 3, research interested): *In the case of interest, a continuation of the course is possible. The extension will then be a set of seminars with active students. Examples on seminars:*

- *Nonlinear equation solvers*
- *Flowsheeting methods, partitioning and tearing*
- *DAE-solvers*
- *MOL - accuracy and stability*
- *Adaptive regridding in PDE-solvers*

Projects after own choice.

Bernt Nilsson

Dept. of Chemical Engineering II, Lund Institute of Technology, Box 124, 221 00 LUND, Sweden

URL: `http://www.chemeng.lth.se/Pers/Bernt`**, E-mail:** `bernt.nilsson@chemeng.lth.se`**, phone: +46 46 22 236 27**

This page has the following URL:

`http://www.chemeng.lth.se/pers/bernt/pscourse.html`

Last update: May 15, 1996.
Bernt Nilsson
bernt.nilsson@chemeng.lth.se

Simulation of Process Systems

PhD course at Department of Chemical Engineering II
april and may 1996.

A short Bibliography

A (not complete list of) number of references and *personal reflections*

Course Literature:

Davis (1984), *Numerical Methods and Modeling for Chemical Engineers*, Wiley
A well organized book which gives an introduction to numerical methods for ODE and PDE problems. The whole book is used in the course. Sometimes to breif.

Fröberg (1985), *Numerical Mathematics, chap 10 and 11* , Benjamin/Cummings
A book that breifly discuss many areas in numerics (to breif in my opinion). Two chapters are used in the course which gives a breif presentation of methods for linear and nonlinear equation systems.

books in Chemical and Process Engineering:

Finlayson (1980): *Nonlinear Analysis in Chemical Engineering,* A book that could be used as the course book. More detailed then Davis.

Ramirez (1989): *Computational Methods for Process Simulation,* A well organized book and an interesting contents but are based on sequential thinging and FORTRAN coding which makes the text minor interesting *(an old book in a new cover).*

Villadsen and Michelsen (1978): *Solution of Differential Equation Models by Polynomial Approximations* , Prentice-Hall

books in Numerical Mathematics:

Mathews (1992), *Numerical Methods for mathematics, science and engineering,* Prentice-Hall. Gives a detailed presentaion of simple methods. There is a supplement with MATLAB examples and MATLAB codes, which are used in the course.

Golub and van Loan (1982): *Matrix Computations,* A bible in numerical linear algebra!

Dahlqvist and Björk (1974): *Numerical Methods,* A well written and condensed material. A dominating book on the subject in Sweden.

Schiesser (1991): *The Numerical Methods of Lines,* Academic Press. A book focused upon MOL and finite difference. *(to focused)*

Ames (1992 *(from 1969)*): *The Numerical Methods for Partial Differential Equations,* Academic Press, A book that covers a lot but the new book contains both new and old methods and it is not

obvious whats of interest today.

Hairer, Norsell and Wanner (1992): *Solving Ordinary Differential Equations I, nonstiff problems* Springer-Verlag, THE book on the subject. A good reference with detailed presentations and discassion

Hairer and Wanner (1992): *Solving Ordinary Differential Equations II, stiff and differential-algebraic problems* Springer-Verlag, part 2 of the previous one.

---

other good references:

Söderlind (1987), *Numerical Analysis of Ordinary Differential Equations* (lecture notes PhD course).

Söderlind (1989), *Stiff Differential Equations* (article).

Marquardt (1994), *Numerical Methods for the Simulation of Differential-Algebraic Process Models* (article).

Software Tools and Manuals:

MathWorks, MATLAB manuals.

Shampine and Reichelt: *the MATLAB ODE Suite*. Detailed presentations of the methods in the suite

Optimization toolbox contain a nonlinear equation solver `fsolve`

Spline toolbox contain piecewise polynomials and B-splines. There is an example how to solve ODE with BVP using the collocation method.

PDE-toolbox contains a well written chapter introducing Galerkin based Finite Element Method.

Back to
Simulation of Process Systens

Last update: May 23, 1996.
Bernt Nilsson
bernt.nilsson@chemeng.lth.se

SIMULATION OF PROCESS SYSTEMS

Introduction

Bernt Nilsson, Chemical Engineering II
Bernt.Nilsson@chemeng.lth.se

SIMULATION

from NATIONALENCYKLOPEDIN:

Simulering, att representera ett system med ett annat i avsikt att studera dess dynamiska uppförande eller för att under laboratorieförhållanden träna behärskandet av systemet. Motiven för simulering är att systemet är alltför komplext för en analytisk undersökning, ännu inte är tillgängligt, är för dyrbart eller för farligt. etc.

Simulator, apparat eller anläggning som helt eller delvis efterliknar komplicerade händelseförlopp och maskiner i samspel med människor. En simulator kan direkt upplevas och påverkas på samma sätt som sin förebild av förare eller operatör. etc.

SIMULATION

Contents:

- When is **simulation** useful?
- How is it done? History and State-of-the art.
- Simulation methods presented in this course
- Relations to other courses

SIMULATION OF PROCESS SYSTEMS

When is simulation useful?

- **Understanding**
  Verify that model and system agree. Understanding of model/system behaviour.
- **Virtuell experiments**
  make "experiments" on the system/model that are hard to do, expensiv, takes long time, dangerous,etc.
- Numerical **studies**
  Make detailed studies and analysis of the system.
- **Design**
  Design and configuration studies of a new system
- **Training** of operators
  Special design simulators for training
- games and entertainment

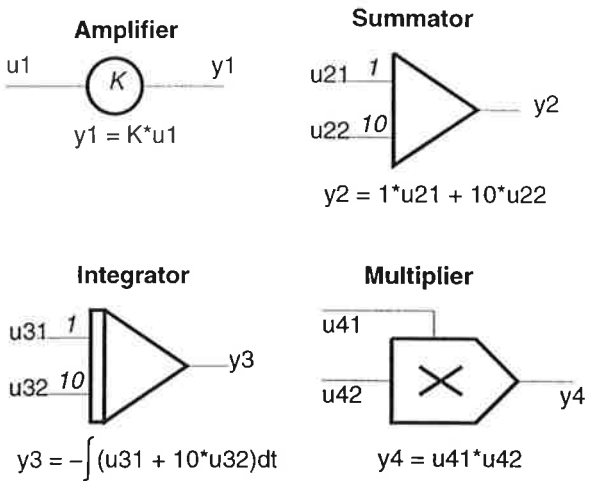
SIMULATION OF PROCESS SYSTEMS

How is simulation done?

- **"Physical" simulator**
  Representation of the system in another physical system. (diffusion problem rep. in a heating problem)
- **Analog computers**
  The mathematical model of the system is represented in an electrical circuit with the same behaviour.
- **Digital simulation**
  The use of numerical methods for equation solving of the mathematical model of the system.
- **"Hybrid" simulators**
  A combination of digital simulation and the use of "physical" simulation. (aircraft simulation using the real instrumentation)

SIMULATION I
Analog Computers

Electrical circuit components used in analog computer setups

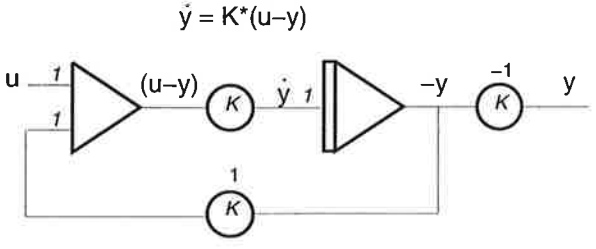


SIMULATION II
Analog Computers

The concentration dynamics of an ideal mixed tank model (constant flow and volume) can be described by the following first order differential equation

$$\frac{dc}{dt} = \frac{q}{V}(c_{in} - c)$$

The corresponding analog computer configuration becomes



SIMULATION III
State-of-the-art

Digital simulation of today:

- **Conventional programming languages**
  Creation of special designed programs with the use of public numerical routines
  ex. FORTRAN, C
- **Technical computing environments**
  Environments for mathematical problem solving.
  ex. MATLAB, Maple
- **Simulation environments**
  Special designed enviroments for simulation
  ex. MAX, Simnon (Omola and Dymola)
- **Real time control enviroments**
  General system for real time applications and operator support
  ex. G2, SattLine

SIMULATION IV
State-of-the-art

A CSTR example in MATLAB:

```
function [dx] = cstr(t,x)
% Adiabatic continuous stirred tank reactor
% reaction: A + 3B -> 2C ; r=k[A]
% states: 1=A, 2=B, 3=C and 4=T
k0 = 7.94e7; ea=49.46; R=8.314e-3; dh = -210;
q = 1.15e-3; V=20*q; m=1000*V; cp=4.17e-3;
nu  = [-1; -3; 2; -dh/(m*cp)];
qin = [10.61 40 0 25]';
k = k0.*exp(-ea./(R.*(x(4)+273));
prod = k*x(1).*nu;
dx = q/V.*(xin - x) + prod;
```

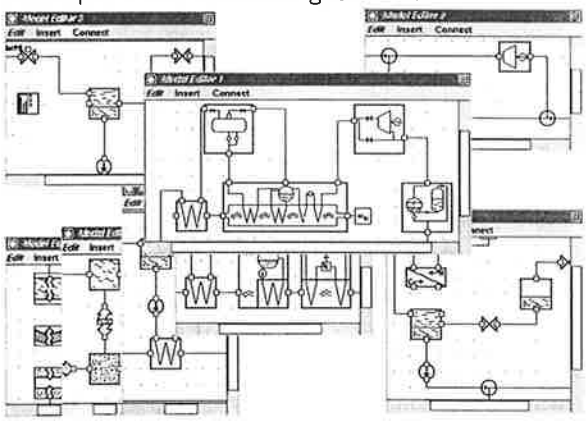
This description of a reactor is simulated with the following command:

```
ode23s('cstr',[0;50],[0;0;0;25])
```

- General numerical tool
- Toolboxes for simulation, optimization, splines etc.
- User M-file libraries

SIMULATION V
State-of-the-art

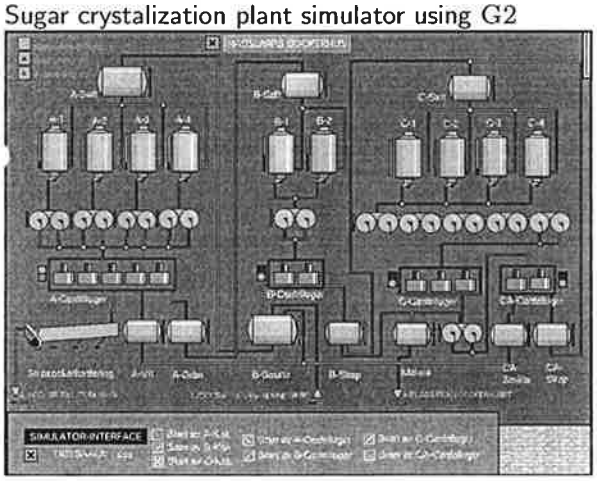
Power plant simulation using OMSIM



- High level model description
- Object-oriented modeling
- Differential-algebraic equation solvers
- Continuous and event simulation

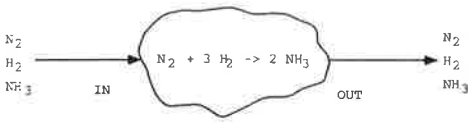
SIMULATION VI
State-of-the-art

Sugar crystalization plant simulator using G2



- Different programming styles
- Integrated graphical interface
- Real-time simulator
- Animation

SIMULATION PROBLEMS I
Algebraic System



In a system the following simple reaction occures.

$$N_2 + 3H_2 \longrightarrow 2NH_3$$

case 1: *Static overall model*
Assume

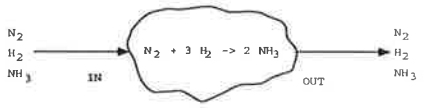
- static behaviour
- overall model

Put up three static species balances over the system

MB:	ACC	=	IN	-	OUT	+	PROD
N_2:	0	=	$N^{in}_{N_2}$	-	$N^{out}_{N_2}$	+	$(-1)\mathbf{r}$
H_2:	0	=	$N^{in}_{H_2}$	-	$N^{out}_{H_2}$	+	$(-3)\mathbf{r}$
NH_3:	0	=	$N^{in}_{NH_3}$	-	$N^{out}_{NH_3}$	+	$(+2)\mathbf{r}$

We have an algebraic equation set with 7 unknowns and 3 equations.

SIMULATION PROBLEMS II
Algebraic System



Assume the following

- N^{in} is known
- Specify the production of NH_3

Now the problem can be expressed as a linear equation system!

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} N_{N_2}^{out} \\ N_{H_2}^{out} \\ \mathbf{r} \end{bmatrix} = \begin{bmatrix} N_{N_2}^{in} \\ N_{H_2}^{in} \\ N_{NH_3}^{in} - N_{NH_3}^{out} \end{bmatrix}$$

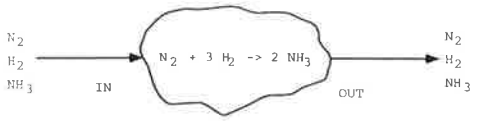
or on a more compact form

$$A\mathbf{x} = b$$

Steady-state simulation solves the algebraic equations for a set of different specifications, eg a number of different $N_{NH_3}^{out}$.

General case, nonlinear equation: $f(x) = 0$

SIMULATION PROBLEMS III
Dynamic Lumped System



case 2: *Dynamic overall model*
Assume lumped descriptions of the amount of the species in the system
Put up species balances over the dynamical system

MB:	ACC	=	IN	-	OUT	+	PROD
N_2:	$\frac{dn_{N_2}}{dt}$	=	$N_{N_2}^{in}$	-	$N_{N_2}^{out}$	+	$(-1)\mathbf{r}$
H_2:	$\frac{dn_{H_2}}{dt}$	=	$N_{H_2}^{in}$	-	$N_{H_2}^{out}$	+	$(-3)\mathbf{r}$
NH_3:	$\frac{dn_{NH_3}}{dt}$	=	$N_{NH_3}^{in}$	-	$N_{NH_3}^{out}$	+	$(+2)\mathbf{r}$

This results in 10 unknowns and 3 equations.

SIMULATION PROBLEMS IV
Dynamic Lumped System

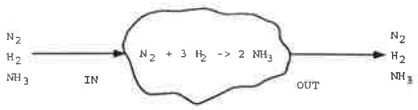
Assume the following

- homogenous concentrations and ideal mixing
- first order reaction kinetics
- constant volume

ACC	=	IN	-	OUT	+	PROD
$V\frac{dc_{N_2}}{dt}$	=	$qc_{N_2}^{in}$	-	qc_{N_2}	-	kVc_{N_2}
$V\frac{dc_{H_2}}{dt}$	=	$qc_{H_2}^{in}$	-	qc_{H_2}	-	$3kVc_{N_2}$
$V\frac{dc_{NH_3}}{dt}$	=	$qc_{NH_3}^{in}$	-	qc_{NH_3}	+	$2kVc_{N_2}$

If we know the inflow concentrations and assume a value on the concentrations inside the system at time t_0 we can calculate the derivitives of the concentrations.

SIMULATION PROBLEMS V
Dynamic Lumped System



A description like this is called a **state-space** description of first order **ordinary differential equation** system (ODE)

ACC	=	IN	-	OUT	+	PROD
$\frac{dc_{N_2}}{dt}$	=	$\frac{q}{V}c_{N_2}^{in}$	-	$\frac{q}{V}c_{N_2}$	-	kc_{N_2}
$\frac{dc_{H_2}}{dt}$	=	$\frac{q}{V}c_{H_2}^{in}$	-	$\frac{q}{V}c_{H_2}$	-	$3kc_{N_2}$
$\frac{dc_{NH_3}}{dt}$	=	$\frac{q}{V}c_{NH_3}^{in}$	-	$\frac{q}{V}c_{NH_3}$	+	$2kc_{N_2}$

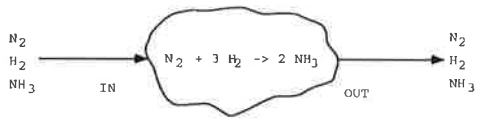
This can be rewritten as

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu$$

Dynamic simulation solves a set of **ordinary differential equations** (ODE) based on the initial values of the "states", the so called **initial value problem** (IVP)

Nonlinear state-space case: $\dot{x} = f(x, u)$

SIMULATION PROBLEMS VI
Distributed Parameter System



case 3: *Static space dependent model*
Assume the following

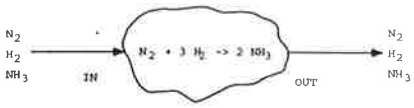
- one space dimension
- constant cross area (tube geometry)
- space dependent concentrations
- first order reaction kinetics

Species balances over a slice in the "tube" gives the following, if one assumes **steady-state** conditions in the system

ACC	=	IN	-	OUT	+	PROD
0	=	$q^z c^z_{N_2}$	-	$q^{z+dz} c^{z+dz}_{N_2}$	-	$Adzkc_{N_2}$
0	=	$q^z c^z_{H_2}$	-	$q^{z+dz} c^{z+dz}_{H_2}$	-	$Adz3kc_{N_2}$
0	=	$q^z c^z_{NH_3}$	-	$q^{z+dz} c^{z+dz}_{NH_3}$	+	$Adz2kc_{N_2}$

Let $dz \rightarrow 0$

SIMULATION PROBLEMS VII
Distributed Parameter System



We get a set of ODEs in space

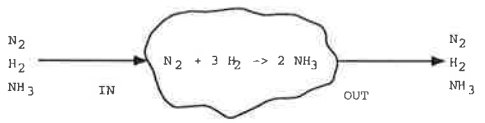
$$\begin{aligned}
\frac{d(qc_{N_2})}{dz} &= -kc_{N_2} \\
\frac{d(qc_{H_2})}{dz} &= -3kc_{N_2} \\
\frac{d(qc_{NH_3})}{dz} &= 2kc_{N_2}
\end{aligned}$$

Mathematically this is representaed as $\dot{\mathbf{x}} = A\mathbf{x}$.

We have 6 unknowns and 3 equations and we can assume that we know the variables at the inlet or/and at the outlet.

- c known at $z = z_{in} \Rightarrow$ IVP
- otherwise we have a **boundary-value problem** (BVP) where c at $z = z_{out}$ is known

SIMULATION PROBLEMS VIII
Distributed Parameter System



case 4: *Dynamic space dependent model*
Assume the following

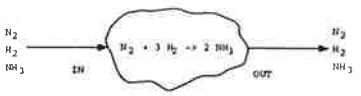
- one space dimension
- constant cross area (tube geometry)
- space dependent concentrations
- first order reaction kinetics

Dynamic species balances over a slice

ACC	=	IN	-	OUT	+	PROD
$\frac{d(Adzc_{N_2})}{dt}$	=	$q^z c^z_{N_2}$	-	$q^{z+dz} c^{z+dz}_{N_2}$	-	$Adzkc_{N_2}$
$\frac{d(Adzdc_{H_2})}{dt}$	=	$q^z c^z_{H_2}$	-	$q^{z+dz} c^{z+dz}_{H_2}$	-	$Adz3kc_{N_2}$
$\frac{d(Adzdc_{NH_3})}{dt}$	=	$q^z c^z_{NH_3}$	-	$q^{z+dz} c^{z+dz}_{NH_3}$	+	$Adz2kc_{N_2}$

Let $dz \rightarrow 0$

SIMULATION PROBLEMS IX
Distributed Parameter System



We get a set of **Partial Differential Equations** in time and one space dimension

$$\begin{aligned}
\frac{\partial c_{N_2}}{\partial t} &= -\frac{\partial (qc_{N_2})}{\partial z} - kc_{N_2} \\
\frac{\partial c_{H_2}}{\partial t} &= -\frac{\partial (qc_{H_2})}{\partial z} - 3kc_{N_2} \\
\frac{\partial c_{NH_3}}{\partial t} &= -\frac{\partial (qc_{NH_3})}{\partial z} + 2kc_{N_2}
\end{aligned}$$

Mathematically this is expressed as a linear (parabolic) PDE

$$\frac{\partial \mathbf{x}}{\partial t} = A\frac{\partial^2 \mathbf{x}}{\partial z^2} + B\frac{\partial \mathbf{x}}{\partial z} + B\mathbf{x}$$

We have 9 unknowns and 3 equations and we need to specify

- **initial values** in time
- **boundary values** in space

SIMULATION PROBLEMS X
Problem Formulations

The different cases show that the same application results in a number of different simulation problems. Depending on the choice of model description the simulation problem is changed

- Overall and steady-state
  ⇒ Algebraic equations, AE
- Overall and dynamic
  ⇒ Ordinary differential equations with initial value problem, ODE-ivp
- One space dimension and steady-state
  ⇒ Ordinary differential equations with boundary value problem, ODE-bvp
- Two or more space dimension and/or dynamic
  ⇒ Partial differential equations with boundary and/or initial value problem, PDE

COURSE CONTENTS
What the course contains

Main topics of the course:

- AE-systems
  - *Linear:* LU decomposition and iterativ methods
  - *Nonlinear:* Fix-point and Newton methods
- ODE-systems with IVP
  - *Explicit:* Runge-Kutta and multi step
  - *Implicit:* Gear and DAE-solvers
- ODE-systems with BVP
  - *rewrite as IVP:* Shooting methods
  - *Discretization:* Finite difference and finite element
- PDE-systems
  - *Parabolic:* Method of lines
- *How to do it in MATLAB*

(This list is **not** complete)

COURSE CONTENTS
What the course do NOT contain

- Modelling
- Flowsheeting
- Numerical analysis of method behaviour
- Implementation issues
- Stochastic simulation
- Discrete simulation

SIMULATION COURSES
at LTH

- Process simulation (at Chemical engineering I)
- Numerical analysis
- ○ Numerical methods, advanced course
- ○ Finite element method (at Solid or structural mechanics)
- \- Simulation (SAM)

SIMULATION
References

Books that covers the course:

FINLAYSON: *Nonlinear Analysis in Chemical Engineering*

RAMIREZ: *Computational Methods for Process Simulation*

Books that covers numerical aspects in the course:

DAHLQVIST AND BJÖRK: *Numerical Methods*

SIMULATION
Requirements for PhD points

Lecture Problems 1 and 2:
Solved by paper and pen (sometimes with the brain). No need for computer solutions. **Hand-In**

Computer Exercises, part A:
A number of problems to be solved by the use of MATLAB (and additional toolboxes) to illustrate theory and methods.

Computer Exercises, part B:
One problem to be solved by the use of MATLAB (or similar programs).

Passive (2 p): **Hand-In of 2 lecture problems**

Active (5 p): **Hand-In of 2 lecture problems and of 3 computer exercises** (select 3 out of 5)

Recommendation: for you that is not familiar with MATLAB go through the MATLAB-course material before Computer Exercise 1!

SIMULATION
MATLAB course

Lectures

Computer Exercises, part A:
A number of problems to be solved by the use of MATLAB (and additional toolboxes) to illustrate theory and methods.

Computer Exercises, part B:
One problem to be solved by the use of MATLAB (or similar programs).

SUMMARY
Simulation of Process Systems

Summary of lecture:

- Simulation: What, When, How?
- History and State-of-the-art
- Motivation for different methods
- Methods presented in this course

SIMULATION OF PROCESS SYSTEMS

Steady-State Simulation of Algebraic Equations

Literature: *Fröberg, chap 10 and 11*

Bernt Nilsson, Chemical Engineering II
Bernt.Nilsson@chemeng.lth.se

STEADY-STATE SIMULATION OF ALGEBRAIC EQUATIONS

Contents:

- Linear Equations
  - Gaussian elimination
  - LU decomposition and pivoting
  - Error analysis and condition numbers
  - Iteration methods
- Sparseness
- Difference Equations
- Nonlinear Equations
  - Fixed-point methods
  - Newton methods

LINEAR EQUATIONS I

A set of linear equations

$$\begin{cases} a_{11}x_1 & +a_{12}x_2 & +\ldots & +a_{1N}x_N & = b_1 \\ a_{21}x_1 & +a_{22}x_2 & +\ldots & +a_{2N}x_N & = b_2 \\ \vdots & & & & \\ a_{N1}x_1 & +a_{N2}x_2 & +\ldots & +a_{NN}x_N & = b_N \end{cases}$$

can be written on matrix form

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1N} \\ a_{21} & a_{22} & \ldots & a_{2N} \\ \vdots & & & \\ a_{N1} & a_{N2} & \ldots & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

and on a more compact form

$$Ax = b$$

LINEAR EQUATIONS II

Methods to solve $Ax = b$.

- Direct methods
  - Invert A and calculate $x = A^{-1}b$
  - Gaussian Elimination (LU decomposition)
- Indirect methods (iteration methods)
  - Jacobi
  - Gauss-Seidel
  - Succesive Over-Relaxation (SOR)

LINEAR EQUATIONS III
Gaussian Elimination

Illustration of Gaussian elimination on one example from Fröberg

$$\begin{bmatrix} 1 & -2 & 3 & -4 \\ 3 & -2 & 3 & -7 \\ 5 & -18 & 29 & -23 \\ 4 & -4 & 0 & -29 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \\ 1 \\ -25 \end{bmatrix}$$

Write in a form used in the linear algebra course:

$$\left[\begin{array}{cccc|c} 1 & -2 & 3 & -4 & 0 \\ 3 & -2 & 3 & -7 & 5 \\ 5 & -18 & 29 & -23 & 1 \\ 4 & -4 & 0 & -29 & -25 \end{array}\right]$$

Multiply row 1 with 3 and subtrac it from row 2 and so on...

$$\left[\begin{array}{c|cccc} 0 & \mathbf{1} & -2 & 3 & -4 & 0 \\ 3 & 0 & 4 & -6 & 5 & 5 \\ 5 & 0 & -8 & 14 & -3 & 1 \\ 4 & 0 & 4 & -12 & -13 & -25 \end{array}\right]$$

LINEAR EQUATIONS IV
Gaussian Elimination cont.

Continuation of the elimination example

$$\left[\begin{array}{cc|cccc} & & 1 & -2 & 3 & -4 & 0 \\ 3 & & 0 & 4 & -6 & 5 & 5 \\ 5 & -2 & 0 & 0 & 2 & 7 & 11 \\ 4 & 1 & 0 & 0 & -6 & -18 & -30 \end{array}\right]$$

$$\left[\begin{array}{ccc|cccc} & & & 1 & -2 & 3 & -4 & 0 \\ 3 & & & 0 & 4 & -6 & 5 & 5 \\ 5 & -2 & & 0 & 0 & \mathbf{2} & 7 & 11 \\ 4 & 1 & -3 & 0 & 0 & 0 & \mathbf{3} & 3 \end{array}\right]$$

We have obtained a triangular system, which can be solved by backsubstitution.

The diagonal elements, $1, 4, 2, 3$, are called **pivots**.

LINEAR EQUATIONS V
LU Decomposition

Gaussian elimination is also called *LU decomposition*

$$A = LU$$

where L is *Lower triangular* and U is *Upper triangular*

From the previous example we get

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 5 & -2 & 1 & 0 \\ 4 & 1 & -3 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & -2 & 3 & -4 \\ 0 & 4 & -6 & 5 \\ 0 & 0 & 2 & 7 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

LINEAR EQUATIONS VI
Pivoting and Permutations

Gaussian elimination is unstable because of the possibility of abitarily small pivots (diagonal elements). This is avoided by the permutation of A, i.e. make LU decomposition of PA where P is the permutationmatrix

Pivoting:

- *Complete pivoting:* search for the larges entry in the submatrix, both row and column permutations, stable,
- *Partial pivoting:* search for the largest entry in column, only row permutations

LINEAR EQUATIONS VII
Error Analysis

The condition number, μ, is defined as

$$\mu = ||A|| \cdot \left|\left|A^{-1}\right|\right|$$

The "relative" error can now be expressed as

$$\varepsilon \leq \frac{\mu}{1 - \varepsilon_1 \mu}(\varepsilon_1 + \varepsilon_2)$$

where $\varepsilon = \frac{||\delta x||}{||x||}$, $\varepsilon_1 = \frac{||\delta A||}{||A||}$ and $\varepsilon_2 = \frac{||\delta b||}{||b||}$

This means

- $\mu \geq 1$
- $\varepsilon \to \infty$ when $\mu \to \infty$
- A is ill-conditioned if μ is large

LINEAR EQUATIONS VIII
Direct Methods in MATLAB

- `[L,U,P]= lu(A)` makes a LU decomposition with partial pivoting.
- `x = A\b` uses `lu`
- `cond(A)` calculates the condition number.
- `rcond(A)` calculates the relative condition number. This is normally used as an indicator in MATLAB routines.
- `inv(A)` inverts the matrix A.

LINEAR EQUATIONS IX
Iteration Methods

Iteration methods can be used when

- A is diagonally dominant: $|a_{ij}| \geq \sum_{j \neq i} |a_{ij}|$
- A matrix is large and sparse

Simple version of iteration: Jacobi method.

$$\begin{aligned} x_1^{k+1} &= (b_1 - a_{12}x_2^k - a_{13}x_3^k)/a_{11} \\ x_2^{k+1} &= (b_2 - a_{21}x_1^k - a_{23}x_3^k)/a_{22} \\ x_3^{k+1} &= (b_3 - a_{31}x_1^k - a_{32}x_2^k)/a_{33} \end{aligned}$$

Previously calculated x^k are used to calculate new x^{k+1}

LINEAR EQUATIONS X
Iteration Methods cont.

Assume the following decomposition of th A matrix:

$$A = L_L + D + U_U$$

Lower triangular matrix

$$L_L = \begin{bmatrix} 0 & 0 & \dots & 0 \\ a_{21} & 0 & \dots & 0 \\ \vdots & & \dots & \\ a_{N1} & a_{N2} & \dots & 0 \end{bmatrix}$$

The diagonal matrix

$$D = diag(a_{11}, a_{22}, \dots, a_{NN})$$

Upper triangular

$$U_U = \begin{bmatrix} 0 & a_{12} & \dots & a_{1N} \\ 0 & 0 & \dots & a_{2N} \\ \vdots & & \ddots & \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

LINEAR EQUATIONS XI
Jacobi Iteration Method

The simple **Jacobi** iteration can now be rewritten on matrix form

$$Dx^{k+1} = -(L_L + U_U)x^k + b$$

- It is easy to invert D
- Do not use the latest x-values
- Can be extended to nonlinear iterations

LINEAR EQUATIONS XII
Gauss-Seidel Iteration Method

Makes use of the new x-values

$$x_1^{k+1} = (b_1 - a_{12}x_2^k - a_{13}x_3^k)/a_{11}$$
$$x_2^{k+1} = (b_2 - a_{21}\mathbf{x_1^{k+1}} - a_{23}x_3^k)/a_{22}$$
$$x_3^{k+1} = (b_3 - a_{31}\mathbf{x_1^{k+1}} - a_{32}\mathbf{x_2^{k+1}})/a_{33}$$

This can be expressed on matrix form

$$(D + L_L)x^{k+1} = -U_U x^k + b$$

- It is easy to invert $(D + L_L)$, (triangular)
- Use the latest x-values
- Can be extended to nonlinear iterations

LINEAR EQUATIONS XIII
Successive Over-Relaxation

The Gauss-Seidel method can sometimes converge very slow. To increase the convergence, different *relaxation methods* are developed.

The **successive over-relaxation** method (SOR) is expressed on matrix form as follows:

$$(D + \omega L_L)x^{k+1} = ((1-\omega)D - \omega U_U)x^k + \omega b$$

- $\omega > 1$ means over-relaxation
- $0 < \omega < 1$ means under-relaxation
- Theory for optimal ω

LINEAR EQUATIONS XIV
Summary: Iteration Methods

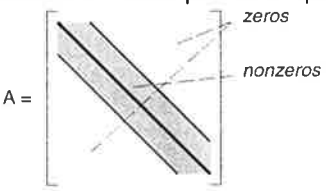
Iteration methods solve $Ax = b$ with the following iteration procedure

$$Mx^{k+1} = Nx^k + b$$

- Jacobi:
  $M_J = D$ and $N_J = -(L_L + U_U)$
- Gauss-Seidel:
  $M_{GS} = (D + L_L)$ and $N_{GS} = -U_U$
- Succesive over-relaxation, SOR:
  $M_{SOR} = (D + \omega L_L)$, $N_{SOR} = (I - \omega)D - \omega U_U$ and $b_{SOR} = \omega b$

SPARENESS

Banded matrix is an example of a sparse matrix.



Properties of **sparse matrices**

- Majority of the elements are zero
- (Sparseness) density is the relation non-zero and zero elements, `nnz`
- *Sparse matrix technique* only store and handle non-zeros.
- MATLAB has sparse matrix technique
- `sparse` convert a matrix to sparse form
- `full` do the opposite

DIFFERENCE EQUATION I
Eigenvalues and Stability

Assume the following linear difference equation system

$$x^{k+1} = \Phi x^k + \Gamma u^k$$

Steady-state solution for u^o:

$$x^{k+1} = x^k = (I - \Phi)^{-1}\Gamma u^o$$

Dynamic behaviour is captured in the eigenvalues λ

$$\Phi z = z\lambda \qquad \Rightarrow \qquad \det(\lambda I - \Phi) = 0$$

- $|\lambda| < 1$, stabil system
- $|\lambda| > 1$, unstabil system
- $\lambda < 0$, x changes sign every iteration

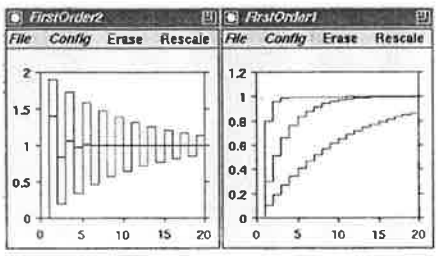
DIFFERENCE EQUATION II
First Order System

One linear difference equation with constant input, $u^o = 1$

$$x^{k+1} = ax^k + (1 - a)$$

$$\lambda = -a$$

$$x^\infty = \frac{1-a}{1-a} = 1$$

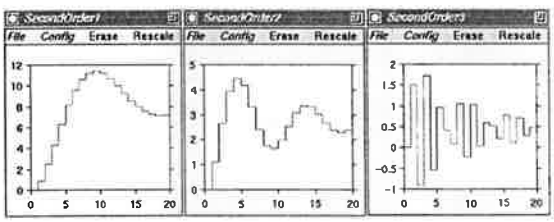


- *left:* negative real eigenvalues, $\lambda = -0.9, -0.4$
- *right:* positive real eigenvalues, $\lambda = 0.9, 0.7, 0.2$

DIFFERENCE EQUATION III
Second Order System

Assume the following difference equation

$$x^{k+1} = \begin{bmatrix} a_{11} & 1 \\ a_{21} & 0 \end{bmatrix} x^k + \begin{bmatrix} 0 \\ b_2 \end{bmatrix} u^k$$



- *left:* eigenvalues, $\lambda \approx 0.85 \pm 0.3i$
- *middle:* eigenvalues, $\lambda \approx 0.7 \pm 0.6i$
- *right:* eigenvalues, $\lambda \approx -0.8 \pm 0.4i$

NONLINEAR EQUATIONS
Iteration Methods

- Fix-point iteration
  - *Jacobi*
  - *Gauss-Seidel*
- Bracketing methods
  - Bisection
  - Regula falsi
- Newton methods
  - *Newton-Raphsson*
  - *Secant*
  - Acceleration methods
- Minimization methods
  - *Newton modifications*, Broyden
  - Gradient methods

NONLINEAR EQUATIONS I
Fix-point Iteration

Nonlinear equations can be written as

$$f(x) = 0$$

In the general case it is hard or impossible to find an analytical solution. Rewrite the nonlinear functions as

$$x = g(x) \qquad (= f(x) + x)$$

Guess a starting value of x^0 and iterate until it converges

$$x^{k+1} = g(x^k)$$

- Convergence: $|g'(x)| < 1$
- Illustration: see p 179 in Fröberg

NONLINEAR EQUATIONS II
Fix-point Iteration cont.

Methods for nonlinear equation systems

- Jacobi: $x_i^{k+1} = g_i(x_1^k, \ldots, x_N^k)$
- Gauss-Seidel: $x_i^{k+1} = g_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i^k, \ldots, x_n^k)$

NONLINEAR EQUATIONS III
Newton Methods

Make a Taylor expansion of the function f:

$$f(x) = f(x^0) + \frac{\partial f}{\partial x}(x - x^0) + \frac{\partial^2 f}{\partial x^2}(x - x^0)^2 + \ldots$$

Make following assumptions

- Neglect second order and higher terms
- $f(x) = 0$
- $f'(x^0) = (\frac{\partial f}{\partial x})_{x^0}$

$$\begin{aligned} 0 &= f(x^0) + f'(x^0)(x - x^0) \\ x &= x^0 - \frac{f(x^0)}{f'(x^0)} \end{aligned}$$

Let us use this expression and create an iteration procedure

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

This is the **Newton-Raphson** method

NONLINEAR EQUATIONS IV
Newton Methods cont.

The Newton method can easily be generalized for nonlinear equation systems with the following notation

- function vector: $f(x)$
- Jacobian: $J(x) = \frac{\partial f}{\partial x}$

Iteration procedure

$$x^{k+1} = x^k - J^{-1}(x^k)f(x^k)$$

or more computational correct

$$\begin{aligned} 0 &= f(x^k) + J(x^k)(x^{k+1} - x^k) \\ J(x^k)x^{k+1} &= f(x^k) - J(x^k)x^k \end{aligned}$$

Interpreted as $A^k x^{k+1} = b^k$, where A and b change in each interation step.

NONLINEAR EQUATIONS V
Secant Method

The secant method can be seen as a modification of the Newton method with an approximation of derivitive

$$f'(x^k) \approx \frac{(f(x^k) - f(x^{k-1}))}{(x^k - x^{k-1})}$$

This give us the following iteration procedure

$$x^{k+1} = x^k - f(x^k)\frac{(x^k - x^{k-1})}{(f(x^k) - f(x^{k-1})}$$

NONLINEAR EQUATIONS VI
Minimization Methods

Rewrite the equation as a minimization problem

$$f(x) = 0 \qquad \Rightarrow \qquad \min f^2(x)$$

Notation

- function vector: $g(x) = f^2(x)$
- Jacobian: $J(x) = \frac{\partial g}{\partial x}$
- Hessian: $H(x) = \frac{\partial^2 g}{\partial x^2}$

Taylor expansion in several dimensions

$$g(x) = g(x_0) + (x - x_0)^T J(x_0) + \frac{1}{2}(x - x_0)^T H(x_0)(x - x_0) + \ldots$$

The iteration procedure now becomes

$$x^{k+1} = x^k - H(x^k)^{-1}J(x^k)$$

Newton method in optimization

NONLINEAR EQUATIONS VI
MATLAB routines

- `fzero`, zero of a function of one variable (combination of bisection,secant and interpolation methods)
- `roots` finds the roots to a rational polynomial.
- `fmins` minimize a function of several variables (Nelder-Meade simplex search)
- `fsolve` uses a least square minimization method. (in optimization toolbox: Gauss-Newton or Levenberg-Marquardt)

SUMMARY
Simulation of AE Systems

Summary of lecture:

- Linear Equations
  - LU Facorization
  - Iteration methods
- Sparseness
- Difference Equations
- Nonlinear Equations
  - Fixed-point methods
  - Newton methods

SIMULATION OF PROCESS SYSTEMS

Dynamic Simulation of Ordinary Differential Equations

Literature: *Davis, chap 1*

Bernt Nilsson, Chemical Engineering II
Bernt.Nilsson@chemeng.lth.se

DYNAMIC SIMULATION OF ORDINARY DIFFERENTIAL EQUATIONS

Contents:

- Integration Method Classes
  - Explicit Methods
  - Implicit Methods
  - Multistep Methods
- Stiffness
- Methods in MATLAB
- Differential-Algebraic Equations
- OmSim and Dymola

DYNAMIC SIMULATION I
Initial Value Problem of ODEs

Initial value problem of ODE is expressed mathematically on **state-space** as

$$\frac{dy(x)}{dt} = f(x,y)$$
$$y(x_0) = y_0$$

which represents a set of nonlinear differential equations on matrix form.

$$y(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_m(x) \end{bmatrix}, \quad f(x) = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \\ \vdots \\ f_m(x,y) \end{bmatrix},$$

$$y_0 = \begin{bmatrix} y_1(x_0) \\ y_2(x_0) \\ \vdots \\ y_m(x_0) \end{bmatrix}$$

INTEGRATION METHODS I
Euler Method

Assume an ODE on **state-space** form

$$\frac{dy(x)}{dt} = f(x,y)$$

Make a Taylor expansion of the solution y at a point

$$y(x_{i+1}) = y(x_i) + y'(x_i)(x_{i+1} - x_i) + y''(\xi_i)\frac{(x_{i+1} - x_i)^2}{2!}$$

where $x_i \leq \xi_i \leq x_{i+1}$. Use the ODE in the Taylor expression and truncating after the second term. We get an approximation $u_i \approx y_i$ and

$$u_{i+1} = u_i + hf_i$$

This is the **Euler** method.

- $h = \frac{(x_N - x_0)}{N}$
- h is called the **step-size**
- N is the number of **steps**

INTEGRATION METHODS II
An Overview

- Explicit methods
  Uses only old values to calculate new ones
  (Euler: $u_{i+1} = u_i + hf_i$)
- Implicit methods
  Uses new values to calculate new ones, requires iterations
  (Implicit Euler: $u_{i+1} = u_i + hf_{i+1}$)
- One-step methods
  Uses only the prevouis step i to calculate the new one
- Multistep methods
  Uses several steps (k-step method)
- Low/High order methods
  Refers to accuracy/step-size relation

Euler is an explicit, one-step, low order method (first order)

EULER METHOD I
Error Analysis

Assume that u_i is exact, the we compute u_{i+1} which creates an error called **local truncation error**,

$$e_{i+1} = z(x_{i+1}) - u_{i+1}$$

This gives in this case

$$z'(x) = f(x, z)$$
$$z(x_i + h) = u_i + hf(x_i, u_i) + \frac{h^2}{2!}z''(\xi_i)$$
$$e_{i+1} = \frac{h^2}{2!}z''(\xi_i) = o(h^2)$$

We say that a method is pth-order accurate

$$e_{i+1} = o(h^{p+1})$$

Which means that **Euler is first order** accurate!

EULER METHOD II
Error Analysis and Stability

The **global error** is defined as

$$\epsilon_{i+1} = y(x_{i+1}) - u_{i+1}$$

and is difference between the true solution and the numerical solution

Stability: Assume a first order ODE with the true solution and the corresponding Euler approximation

$$y'(x) = -ay$$
$$y(x) = y_0 e^{-ax}$$
$$u_{i+1} = u_i - hau_i = (1 - ha)u_i$$

EULER METHOD III
Stability

This means that

$$u_{i+1} = (1-ha)u_i = (1-ha)^2 u_{i-1} = \ldots = (1-ha)^{i+1}u_0$$

The global error now becomes (assume $e_0 = y_0 - u_0$)

$$\begin{aligned} \epsilon_{i+1} &= y(x_{i+1}) - u_{i+1} \\ &= y_0 e^{-(i+1)ha} - (1 - ha)^{i+1}(y_0 - e_0) \\ &= (e^{-(i+1)ha} - (1 - ha)^{i+1})y_0 + (1 - ha)^{i+1}e_0 \end{aligned}$$

We see that the global error becomes abitarily large if $|1 - ha| < 1$, which means for **stability**

$$0 \leq ha \leq 2 \qquad \Rightarrow \qquad h \leq \frac{2}{a}$$

(see table 1.1 in Davis p. 8)

RUNGE-KUTTA METHODS I
General Description

RK-methods are **explicit, one-step** methods which uses function evaluations at points between x_i and x_{i+1} to increase the accuracy.

$$u_{i+1} = u_i + \sum_{j=1}^{\nu} \omega_j K_j$$

where

$$\begin{aligned} K_j &= hf(x + c_j h, u_i + \sum_{l=1}^{j-1} a_{jl} K_l) \\ c_1 &= 0 \end{aligned}$$

RUNGE-KUTTA METHODS II
General Description

Euler: is the simplest RK-method with $\nu = 1, \omega_1 = 1$ and $K_1 = hf(x_i, u_i)$

RK2: $\nu = 2$ means $K_1 = hf(x_i, u_i)$, $K_2 = hf(x_i + c_2 h, u_i + a_{21} K_1)$ and

$$u_{i+1} = u_i + \omega_1 hf(x_i, u_i) + \omega_2 hf(x_i + c_2 h, u_i + a_{21} K_1)$$

and ω, c and a are derived after the selection of function evaluation points and comparision with the Taylor expansion.

(see the steps in proof of two RK2-methods discussed on p. 12-13)

(see Figure 1.2 p. 13)

RUNGE-KUTTA METHODS III
Accuracy

pth order accuracy requires ν (function evaluations) large enough for available parameters for obtaining agreement with the Taylor expansion (truncated after h^p).

p *(order)*	2	3	4	5	6	...
ν *(points)*	2	3	4	6	8	...

Notice the jump from 4 to 6 and the increase in number of function evaluation over order 4.

- Runge-Kutta-Gill is a 4th order RK (RK4).
- More efficient to increase the accuracy then decrease the step-size (see table 1.4)

Higher order methods increase accuracy!

RUNGE-KUTTA METHODS IV
Error Control

Error estimation for error control and step-size selection

- calculate u_{i+1} and u^*_{i+1} for h and $\frac{h}{2}$.
- estimate $e_{i+1} \approx u^*_{i+1} - u_{i+1}$

Runge-Kutta-Fehlberg (RKF) is a method that efficiently makes a error estimate (see **1.36**).
Note:

- "real" step is 4th order accurate
- error estimate is of 5th order

A method like this is called **RK45**-method

IMPLICIT METHODS I
Implicit Euler

Assume an ODE on **state-space** form

$$\frac{dy(x)}{dt} = f(x, y)$$

Make a Taylor expansion of the solution y at the new point

$$y(x_i) = y(x_{i+1}) + y'(x_{i+1})(x_i - x_{i+1}) + y''(\xi_i)\frac{(x_i - x_{i+1})^2}{2!}$$

where $x_i \leq \xi_i \leq x_{i+1}$. Use the ODE in the Taylor expression and truncating after the second term. We get an approximation $u_i \approx y_i$ and

$$u_{i+1} = u_i + h f_{i+1}$$

This is the **implicit Euler** method.

- Nonlinear equation
- Newton iteration in each step
- Converge fast (u_i often near u_{i+1})

IMPLICIT METHODS II
Implicit Euler cont.

Use **Newton** and make a Taylor expansion f_{i+1} over different iteration steps s

$$f_{i+1}^{s+1} = f_{i+1}^{s} + \frac{\partial f^s}{\partial y}_{i+1}(u_{i+1}^{s+1} - u_{i+1}^{s})$$

and replace f in the **implicit Euler** algorithm

$$u_{i+1}^{s+1} = u_i + h(f_{i+1}^{s} + \frac{\partial f^s}{\partial y}_{i+1}(u_{i+1}^{s+1} - u_{i+1}^{s}))$$

Express it as a linear equation $A^s x^{s+1} = b^s$

$$(I - h\frac{\partial f^s}{\partial y}_{i+1})u_{i+1}^{s+1} = u_i + h f_{i+1}^{s} - h\frac{\partial f^s}{\partial y}_{i+1} u_{i+1}^{s}$$

Note the use of a **Jacobian**!

IMPLICIT METHODS III
Stability

Assume the following first order linear ODE ($a > 0$)

$$\frac{dy(x)}{dt} = -ay$$

and put it in the **implicit Euler** approximation

$$u_{i+1} = u_i - h a u_{i+1}$$

Calculate the eigenvalue of this difference equation

$$(1 + ha)u_{i+1} = u_i \Rightarrow \lambda = \frac{1}{(1 + ha)}$$

The eigenvalue is always $0 < \lambda < 1$ which means **always stable**

Implicit methods increase the stability

IMPLICIT METHODS IV
Implicit Runge-Kutta

Runge-Kutta methods can also be rewritten on implicit form.

The Runge-Kutta family

- ERK is explicit RK
- IRK the implicit RK
- DIRK is a diagonally IRK ($a_{ii} \neq 0$)
- SDIRK a special DIRK

Note the need for a Jacobian, analytical or numerical.

MULTISTEP METHODS I
General Description

A *k-step linear multistep method* has the form

$$\begin{aligned} \alpha_0 u_{i+1} \quad + \quad & \alpha_1 u_i + \ldots + \alpha_k u_{i+1-k} \\ = \quad & h(\beta_0 f(x_{i+1}, u_{i+1}) + \beta_1 f(x_i, u_i) \\ & + \ldots + \beta_k f(x_{i+1-k}, u_{i+1-k})) \end{aligned}$$

or more compact

$$\sum_{j_1=0}^{k_1} \alpha_{j_1} u_{i+1-j_1} = h \sum_{j_2=0}^{k_2} \beta_{j_2} f(x_{i+1-j_2}, u_{i+1-j_2})$$

The coefficients, α, β, are derived after matching with the Taylor expansion

- Right hand side is a sum of solution points
- Left hand side is a sum of function evaluations
- $k_1 = 1$ and $k_2 = 1, \beta_0 = 0$ is a Euler
- $k_1 = 1, \beta_0 = 0$ are explicit **Adams-Bashforth**
- $\beta_0 \neq 0$ means an implicit method
- $k_1 = 1$ and $k_2 = 0$ is an implicit Euler

High order explicit multistep methods have poor stability

MULTISTEP METHODS II
Predictor and Corrector Methods

Example of a simple implicit 2-step method

$$u_{i+1} = u_i + \frac{h}{2}(f(x_i, u_i) + f(x_{i+1}, u_{i+1}))$$

Predict a first approximation

$$u^0_{i+1} = u_i + h f_i$$

Then compute a **corrected** value

$$u^{s+1}_{i+1} = u_i + \frac{h}{2}(f_i + f(u^s_{i+1}))$$

Iterate to convergence (one or two iterations).

A *predictor-corrector* method like this is called **Adams-Moulton**.

Implicit multistep methods have high accuracy but are *not* good at stiff problems.

MULTISTEP METHODS III
BDF Methods

Backward difference formula (BDF) of a k-step method is

$$\sum_{j=0}^{k_1} \alpha_j u_{i+1-j} = h f(x_{i+1}, u_{i+1})$$

- Right hand side is a sum of solution points
- Left hand side in only one function evaluation
- Implicit methods, also called **Gear** methods

Gear methods have high stability and are *good* at stiff problems

STIFFNESS I
Problem

Assume a linear differential equation system

$$\dot{x} = Ax$$

The solution of the system is

$$x(t) = x_0 e^{At}$$

Assume that we solve the problem with an explicit Euler *then* the step-size depends on the largest eigenvalue of A,

$$h < \frac{1}{|\lambda|_{max}}$$

Stiffness problem:

- largest step-size is governed by the largest eigenvalue
- solution is govern by the smallest eigenvalue

STIFFNESS II
Solution

Stiffness ratio:

$$SR = \frac{|\lambda|_{max}}{|\lambda|_{min}}$$

which means that

- non-stiff: $SR = 20$
- stiff: $SR = 10^3$
- very stiff: $SR = 10^6$

Implicit methods outperform explicit methods on stiff problems because in explicit methods *stability control* the step-size and *not accuracy*

MATLAB ROUTINES I
Pure MATLAB

Two simple Runge-Kutta solvers are available.

- ode23 is a Runge-Kutta with 2nd order accuracy
- ode45 is a Runge-Kutta with 4nd order accuracy

Function call:
`ode23('fun',t0,t1,[x0 x1])`

MATLAB ROUTINES III
SIMULINK and s-files

- `euler`
- `rk23`, is a Runge-Kutta with 2nd order accuracy
- `rk45`, is a Runge-Kutta with 4nd order accuracy
- `adams`, multistep method for non-stiff problems
- `gear`, BDF method for stiff problems
- `linsim`

Function call with the use of a **s-file** `funs`:
`adams('funs',[t0 t1],[x0 x1])`

MATLAB ROUTINES II
the ODE Suite

A suite of 5 new ODE solvers by Shampine and Reichelt

- ode23, is a 2nd order Runge-Kutta (new)
- ode45, is a 4nd order Runge-Kutta (new)
- ode113, explicit variable order Adams-Bashforth-Moulton method for non-stiff problems
- ode23s, an implicit one-step method (Rosenbrock) with 2nd order accuracy
- ode15s, variable order BDF method for stiff problems
- odesl, uses SIMULINK methods

Function call with M-file, `fun`:
`ode23s('fun',[t0 t1],[x0 x1])`

DIFFERENTIAL-ALGEBRAIC EQUATIONS I
Problem and Solvers

Extreme stiff problem is when one eigenvalue goes to infinite.

$$\begin{aligned} \frac{dy}{dt} &= f(x,y) \\ 0 &= g(x,y) \end{aligned}$$

(Part of the dynamics are so fast that it is described as instantanous or algebraic)

Available DAE solvers

- BDF based solvers like DASSL
- Implicit Runge-Kutta

OMOLA
OmSim Numerical Solvers

OMOLA is an object-oriented Modelling language and Omola models can be simulated in OMSIM. OMSIM contains the following solvers:

- `dasrt`, is a DASSL (BDF) version
- `radau5`, is an implicit Runge-Kutta for DAEs
- `dopri45`, a RK45 method
- `RKsuite`, a variabel order Runge-Kutta

DYMOLA
DymoSim Numerical Solvers

DYMOLA is a commercial dynamic modelling laboratory which contains **DymoSim** for simulation

- `deabm`, variable order method
- `lsode`, variable order methods
- `dopri`-methods (RK)
- `dassl, mexx`, are DAE solvers

SUMMARY
Dynamic Simulation of ODEs

Summary of lecture:

- Low/High order Methods
  - low order are simpler
  - high order increase accuracy
- Explicit/Implicit Methods
  - explicit are simpler
  - implicit has better stability
- One-step/Multistep Methods
  - One-step is simpler
  - Multistep increase accuracy
- Stiffness
  - Use implicit methods like Gear of IRK
- DAE problems
  - DASSL is the most famous

SIMULATION OF PROCESS SYSTEMS

Steady-State Simulation of Ordinary Differential Equations

PART I: DISCRETE VARIABLE METHODS

Literature: *Davis, chap 2*

Bernt Nilsson, Chemical Engineering II
Bernt.Nilsson@chemeng.lth.se

STEADY-STATE SIMULATION OF ODES

Contents:

- Boundary-Value Problems, BVP
- Initial-Value Methods for BVP
  - Shooting methods
  - Superposition
- Finite Difference Methods
  - Difference Approximations
  - Boundary Conditions
- *Finite Element Methods (next lecture)*

STEADY-STATE SIMULATION
Boundary-Value Problem for ODEs

A set of linear ODEs on state-space form are expressed as (for $a < x < b$)

$$\frac{dy}{dx} = F(x)y + z(x)$$

with the boundary conditions, BC

$$Ay(a) + By(b) = \gamma$$

(separated BCs: $Ay(a) = \gamma_1$ and $By(b) = \gamma_2$)

- number of ODEs = number of BCs
  $n_{ODE} = n_{BC}$

◦ only BCs at $x = a$ results in pure IVP

INITIAL-VALUE METHODS

- Shooting methods
  - Linear problems
  - Nonlinear problems
  - Multiple shooting
- Superposition
  Solve the fundamental and particular solutions and add them to fulfil the BCs

SHOOTING METHODS I
Linear Problems

Linear second-order equation

$$-y''(x) + p(x)y'(x) + q(x)y(x) = r(x)$$

with the linear two-point (separated) BC

$$a_0 y(a) - a_1 y'(a) = \alpha$$
$$b_0 y(b) - b_1 y'(b) = \beta$$

- $b_0 = b_1 = 0 \Rightarrow$ ODE-IVP
- $b_0 \neq 0$ or/and $b_1 \neq 0 \Rightarrow$ ODE-BVP

SHOOTING METHODS II
Linear Shooting

Define two functions $y^{[1]}(x)$ and $y^{[2]}(x)$ as the solution of two initial-value problems

- first IVP

$$\begin{aligned} -y^{[1]''} + py^{[1]'} + qy^{[1]} &= r \\ y^{[1]}(a) &= -\alpha C_1 \\ y^{[1]'}(a) &= -\alpha C_0 \end{aligned}$$

- second IVP

$$\begin{aligned} -y^{[2]''} + py^{[2]'} + qy^{[2]} &= 0 \\ y^{[2]}(a) &= a_1 \\ y^{[2]'}(a) &= a_0 \end{aligned}$$

with the following BC-relation (initial-values)

$$a_1 C_0 - a_0 C_1 = 1$$

SHOOTING METHODS III
Linear Shooting

The solution now becomes

$$y(x) = y^{[1]}(x) + sy^{[2]}(x)$$

and it satisfies BC at $x = a$

$$a_0 y(a) - a_1 y'(a) = \alpha$$

and will be a solution if s is chosen such that

$$\theta(s) = b_0 y(b; s) + b_1 y'(b; s) - \beta = 0$$

This equation is linear in s and we get

$$s = \frac{\beta - (b_0 y^{[1]}(b) + b_1 y^{[1]'}(b))}{(b_0 y^{[2]}(b) + b_1 y^{[2]'}(b))}$$

SHOOTING METHODS IV
State-Space Description

The linear second order ODE-BVP is rewritten on state-space form and results in a system with four states.

$$\begin{bmatrix} w^1 \\ v^1 \\ w^2 \\ v^2 \end{bmatrix}' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ q & p & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & p & q \end{bmatrix} \begin{bmatrix} w^1 \\ v^1 \\ w^2 \\ v^2 \end{bmatrix} + \begin{bmatrix} 0 \\ r \\ 0 \\ 0 \end{bmatrix}$$

with the following initial-values

$$\begin{bmatrix} w^1(a) \\ v^1(a) \\ w^2(a) \\ v^2(a) \end{bmatrix} = \begin{bmatrix} -\alpha C_1 \\ -\alpha C_0 \\ a_1 \\ a_0 \end{bmatrix}$$

The solution becomes

$$y(x) = w^1(x) + sw^2(x)$$

where

$$s = \frac{\beta - (b_0 w^1(b) + b_1 v^1(b))}{(b_0 w^2(b) + b_1 v^2(b))}$$

SHOOTING METHODS V
Multiple Shooting

Assume that the solution grows in magnitude as $x \to b$ and that $b_1 = 0$.

$$s = \frac{\beta - b_0 w^1(b)}{b_0 w^2(b)} \approx -\frac{w^1(b)}{w^2(b)}$$

If $|\beta| \ll \left|b_0 w^1(b)\right|$ we get numerical problems when $x \to b$

$$y(x) = w^1(x) - \frac{w^1(b)}{w^2(b)} w^2(x)$$

Multiple shooting decrease this problem:

1. divide the interval into subintervals
2. adjust the initial values to satisfy BC
3. satisfy continuity conditions

SHOOTING METHODS VI
Nonlinear Problems

Nonlinear second-order ODE

$$y'' = f(x, y, y')$$

with the linear two point BC

$$\begin{aligned} a_0 y(a) - a_1 y'(a) &= \alpha \\ b_0 y(b) - b_1 y'(b) &= \beta \end{aligned}$$

is rewritten as an **IVP on state-space form**

$$\begin{bmatrix} w \\ v^1 \end{bmatrix}' = \begin{bmatrix} v \\ f(x, w, v) \end{bmatrix}$$

with the initial values

$$\begin{aligned} w(a) &= a_1 s - c_1 \alpha \\ v(a) &= a_0 s - c_0 \alpha \end{aligned}$$

and the relations $a_1 c_0 - a_0 c_1 = 1$

SHOOTING METHODS VII
Nonlinear Problems cont.

The solution becomes $y(x) = w(x)$ if s is a root of

$$\theta(s) = b_0 u(b; s) + b_1 u'(b; s) - \beta = 0$$

This is a *nonlinear* function in s! Iteration procedure is required, for example **Newton**

$$s^{[k+1]} = s^{[k]} - \frac{\theta(s^{[k]})}{\theta'(s^{[k]})}$$

How can we find $\theta'(s)$

$$\theta'(s) = b_0 \frac{\partial u}{\partial s} + b_1 \frac{\partial u'}{\partial s} = b_0 \frac{\partial w}{\partial s} + b_1 \frac{\partial v}{\partial s} = b_0 \xi + b_1 \eta$$

The **two new states** are expressed by a linearization of the original nonlinear ODEs

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix}' = \begin{bmatrix} \eta \\ (\frac{\partial f}{\partial v})\eta + (\frac{\partial f}{\partial w})\xi \end{bmatrix}$$

with the $\xi(a) = a_1$ and $\eta(a) = a_0$

SHOOTING METHODS VIII
Summary

The shooting method procedure

1. rewrite as two initial-value problems
   - *Linear:* particular and fundamental ODEs
   - *Nonlinear:* original ODEs and linearized ODEs
2. solve the two initial-value problems
3. adjust to satisfy the boundary conditions
   - *Linear:* solve one equation
   - *Nonlinear:* iteration procedure and resolve the IVPs

SHOOTING METHODS IX
MATLAB

There are no direct methods for shooting problems in MATLAB and in toolboxes.

- Linear case can be solved directly
  1. Use odexx to solve the two IVPs.
  2. Adjustment to fit BC using the simulation end-point.
- Nonlinear case requires iteration.
  1. Simulate the two IVPs with odexx.
  2. Make new calculation of the BC and resolve IVP.
- Mathews routines
  - Linear shooting method using a RK4-method. (`rks4`, Mathews, chap9)

FINITE DIFFERENCE METHODS I
General Description

Consider the linear second order ODE again

$$-y''(x) + p(x)y'(x) + q(x)y(x) = r(x)$$

with the Dirichlet BC

$$\begin{aligned} y(a) &= \alpha \\ y(b) &= \beta \end{aligned}$$

Impose a unitform mesh on the interval $[a, b]$

$$\begin{aligned} x_i &= a + ih, \qquad i = 0.1, \ldots, N+1 \\ h &= \frac{b-a}{N+1} \end{aligned}$$

(h = mesh-size, N = interior mesh points)

FINITE DIFFERENCE METHODS II
General Description

The problem now becomes a linear equation system

$$Au = z$$

where

- $u = [u_1, \ldots, u_N]$ are the approximated solutions on the mesh points.
- z contains constant part of ODE and the BC.
- A represents the derivative approximations

DIFFERENCE APPROXIMATIONS I
First-Order Approximations

Consider the following two Taylor expansions

$$y(x+h) = y(x) + hy'(x) + \tfrac{h^2}{2!}y''(x) + \tfrac{h^3}{3!}y'''(\xi_1)$$
$$y(x-h) = y(x) - hy'(x) + \tfrac{h^2}{2!}y''(x) - \tfrac{h^3}{3!}y'''(\xi_2)$$

From these expansions we develop **forward** and **backward** (Euler) difference equations

$$y'(x) = \tfrac{y(x+h)-y(x)}{h} - \tfrac{h}{2!}y''(x) - \tfrac{h^2}{3!}y'''(\xi_1)$$
$$y'(x) = \tfrac{y(x)-y(x-h)}{h} + \tfrac{h}{2!}y''(x) - \tfrac{h^2}{3!}y'''(\xi_2)$$

These approximations are **first-order accurate**

DIFFERENCE APPROXIMATIONS II
Second-Order Approximations

Take the mean of the two first-order approximations

$$2y'(x) = \frac{y(x+h)-y(x)}{h} + \frac{y(x)-y(x-h)}{h} - 2\frac{h^2}{3!}y'''(\xi)$$
$$y'(x) = \frac{y(x+h)-y(x-h)}{2h} - o(h^2)$$

This *centered-difference approximation* is **second-order accurate**

Second order derivatives are approximated in a similar way. Add the previous two Taylor expansions

$$y(x+h) + y(x-h) = 2y(x) + h^2y''(x) + \frac{h^4}{4!}(y''''(\xi_1) + y''''(\xi_2))$$

$$y''(x) = \frac{y(x+h) - 2y(x) + y(x-h)}{h^2} + o(h^2)$$

and this is a **second-order** accurate approximation of the *second order derivative*.

FINITE DIFFERENCE METHODS III
Linear Problems

Consider the linear second order ODE

$$-y''(x) + k_1y'(x) + k_2y(x) = z$$

Apply the second-order approximations u of y

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + k_1\frac{u_{i+1} - u_{i-1}}{2h} + k_2u_i = z$$

with the boundary conditions

$$u_0 = \alpha$$
$$u_{N+1} = \beta$$

FINITE DIFFERENCE METHODS IV
Linear Problems cont.

This can now be described on matrix form

$$\frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \dots & & -1 & 2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} +$$

$$\frac{k_1}{2h}\begin{bmatrix} 0 & -1 & 0 & \dots & 0 \\ -1 & 0 & -1 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \dots & & -1 & 0 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} +$$

$$k_2\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \dots & & 0 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} z+\alpha \\ z \\ \vdots \\ z+\beta \end{bmatrix}$$

Or written more compact

$$Au = z$$

FINITE DIFFERENCE METHODS V
Boundary Conditions

Classification

- *Dirichlet:* $Ay(a) + By(b) = \gamma_D$
- *Neumann:* $Ay'(a) + By'(b) = \gamma_N$
- *Robin* or mixed:
  $A_1y(a) + A_2y'(a) + B_1y(b) + B_2y'(b) = \gamma_R$

Implications on finite difference method, FDM:

∘ **Dirichlet** BCs are values on the boundary and therefore specifies the values of u_0 and u_{N+1}

∘ **Neumann** BCs describe relations between solution points round the boundary. u_0 and u_{N+1} becomes unknown and different approximation techniques for the BC can be used.

BOUNDARY CONDITIONS I
Method of False Boundary

Assume a Neumann BC:

$$\frac{dy}{dx} = 0, \qquad x = b$$

Create a *fictitious* point at $N+2$ and use a center-difference approximation of a first-order derivative

$$\frac{u_N - u_{N+2}}{2h} = 0$$

In this simple example the *fictitious* point is expressed in known points as

$$u_{N+2} = u_N$$

This is now used in the governing equation of the problem.

BOUNDARY CONDITIONS II
Taylor Series based Method

Another method, that avoid fictitious points are based on Taylor expansions for points near the boundary.

Assume a Neumann BC:

$$\frac{dy}{dx} = 0, \qquad x = b$$

Put up the following Taylor series

$$y(x_N) = y(x_{N+1}) + hy'(x_{N+1}) + \tfrac{h^2}{2!}y''(x_{N+1}) + \dots$$
$$y(x_{N-1}) = y(x_{N+1}) + 2hy'(x_{N+1}) + \tfrac{(2h)^2}{2!}y''(x_{N+1}) + \dots$$

A second-order accurate approximation of the Neumann BC are the following

$$4y(x_N) - 3y(x_{N+1}) - y(x_{N-1}) = 2hy'(x_{N+1}) + o(h^3)$$

which results in

$$y'(x_{N+1}) \approx \frac{4y(x_N) - 3y(x_{N+1}) - y(x_{N-1})}{2h}$$

FINITE DIFFERENCE METHODS VI
Nonlinear Problems

Consider the nonlinear second order ODE

$$-y'' + f(x, y, y') = 0$$
$$y(a) = \alpha, \quad y(b) = \beta$$

Uniform mesh and *centered-difference* approximations

$$-\tfrac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + f(x_i, u_i, \tfrac{u_{i+1} - u_{i-1}}{2h}) = 0$$
$$u_0 = \alpha, \qquad u_{N+1} = \beta$$

This is a set of nonlinear equations in u

$$\Phi(u) = 0$$

Iteration required, for example Newton

$$u^{[k+1]} = u^{[k]} - J^{-1}(u^{[k]})\Phi(u^{[k]})$$

FINITE DIFFERENCE METHODS VII
State-Space Problems

A system of ODEs on state-space form

$$y' = f(x, y)$$
$$Ay(a) + By(b) = \alpha$$

Uniform mesh and *centered-difference* approximations

$$\tfrac{u_i - u_{i-1}}{h} = f(x_{i+\frac{1}{2}}, \tfrac{u_i + u_{i-1}}{2})$$
$$u_0 = \alpha, \qquad u_{N+1} = \beta$$

This is a **nonlinear equation system** which requires a nonlinear solver.
Dimension:

- m states or number of ODEs
- m number of BC
- $N+2$ gridpoints

$\Rightarrow$ $m(N+2)$ discrete variables to find

DIFFERENCE APPROXIMATIONS III
High Order Methods

Fourth-Order Approximation can be found directly by the use of Taylor series.
(as in the previous discussed approximations)

$$y'(x) = \frac{1}{4!h}(-2u_{i+2} + 16u_{i+1} - 16u_{i-1} + 2u_{i-2}) + o(h^4)$$

$$y''(x) = \frac{1}{4!h^2}(-2u_{i+2} + 32u_{i+1} - 60u_i + 32u_{i-1} - 2u_{i-2}) + o(h^4)$$

This is also called **5-point differentiation formula**

- To increase the accuracy even futher we require more gridpoints in the approximation.
- There are 7, 9 and 11 point approximations
- *Note that BC-approximations also must have the same accuracy!*

DIFFERENCE APPROXIMATIONS IV
High Order Methods

Richardsson extrapolation uses two BVP solutions with mesh-sizes h and $\frac{h}{2}$ to increace the accuracy. Express the error in a gridpoint as

$$E_i = h^2 a_1(x_i) + h^4 a_2(x_i) + \ldots$$

Use this error expression for the two FD-solutions

$$\begin{aligned} y(x_i) - u_i(h) &= h^2 a_1(x_i) + h^4 a_2(x_i) + \ldots \\ y(x_i) - u_i(\frac{h}{2}) &= (\frac{h}{2})^2 a_1(x_i) + (\frac{h}{2})^4 a_2(x_i) + \ldots \end{aligned}$$

Eliminate a_1-term and we get

$$y(x_i) = \frac{4u_i(\frac{h}{2}) - u_i(h)}{3} + o(h^4)$$

Futher subdivition increase the accuracy to $o(h^6)$ and so on.

Pereyra's method is proven to be superiour, which is an iteration procedure that estimates the truncation error in each step to increase the accuracy.

FINITE DIFFERENCE METHODS MATLAB

- Linear case: $Ax = z$ is sparse or very sparse
  - `sparse(A)` convert to sparse notation
  - `x = A\b` solve the problem (sparse or non-sparse)
- Nonlinear case: $\Phi(u) = 0$
  - `fsolve` is a multi-dimensional nonlinear solver
  - There is no Newton method, based on *sparse matrix technique*, in MATLAB
- Mathews routines
  `findiff` is a second-order FDM for linear problems

SUMMARY

Steady-State Simulation of ODEs:

- Initial-Value Methods for BVP
  - *Shooting methods*
    Rewrite the ODE-BVP as **two ODE-IVP** and adjust the solution to fulfil BC
- Finite Difference Methods
  - Approximations of ODEs and BCs into *discrete variables*
  - Solve an **algebraic equation system**
- *Finite Element Methods (next lecture)*

SIMULATION OF PROCESS SYSTEMS

Steady-State Simulation of Ordinary Differential Equations

PART II: FINITE ELEMENT METHODS

Literature: *Davis, chap 3, app D*

Bernt Nilsson, Chemical Engineering II
Bernt.Nilsson@chemeng.lth.se

STEADY-STATE SIMULATION OF ODES

Contents:

- *Boundary-Value Problems, BVP*
- *Initial-Value Methods for BVP*
- *Finite Difference Methods*
- Finite Element Methods
  - General Description
  - Method of Weighted Residuals
  - Basis Functions
  - Galerkin
  - Collocation

STEADY-STATE SIMULATION Finite Elements for ODE-BVP

Assume one second order linear ODE (for $0 < x < 1$)

$$\frac{d^2y}{dx^2} = y + f(x)$$

with the homogenous Dirichlet boundary conditions

$$y(0) = 0, \qquad y(1) = 0$$

Approximate the solution, $y(x)$, with a set of **piecewise polynomials (pp)**.

$$u(x) = \sum_{j=1}^{m} a_j \phi_j(x)$$

- A piecewise polynomial is a function defined on a partition of the interval
- ϕ_j is piecewise continuously differentiable
- ϕ_j are called **basis functions**
- a_j is unknown constants

FINITE ELEMENT METHODS I Collocation Method

The set of unknown a_j is determined by satisfying BVP exactly at m points, x_i, **the collocation points**. The problem becomes

$$u''(x_i) - u(x_i) - f(x_i) = 0$$

and using the pp-approximation

$$\sum_{j=1}^{m} a_j[\phi_j''(x_i) - \phi_j(x_i)] - f(x_i) = 0$$

and by the use of matrix notation

$$A^C a = f$$

where

- $A_{ij}^C = \phi_j''(x_i) - \phi_j(x_i)$
- $a = [a_1, \ldots, a_m]^T$
- $f = [f(x_1), \ldots, f(x_m)]^T$

ODE-BVP becomes an **algebraic equation system**

FINITE ELEMENT METHODS II
Galerkin Method

Put up the following integrals for $i = 1, \ldots, m$.

$$\int_0^1 [y''(x) - y(x) - f(x)]\phi_i(x)dx = 0$$

(Use the following integral to rewrite the above)

$$\int_0^1 y''(x)\phi_i(x)dx = [y'(x)\phi_i(x)]_0^1 - \int_0^1 y'(x)\phi_i'(x)dx$$

We now get *(BC = 0)*

$$\int_0^1 y'(x)\phi_i'(x)dx + \int_0^1 [y(x) + f(x)]\phi_i(x)dx = 0$$

and use the following notation for convenience

$$(y', \phi_i') + (y, \phi_i) + (f, \phi_i) = 0$$

This is called the weak form of the ODE

FINITE ELEMENT METHODS III
Galerkin Method

The set of unknown a_j is determined by satisfying BVP in the integral,

$$(u', \phi_i') + (u, \phi_i) + (f, \phi_i) = 0$$

and using the pp-approximation we get

$$(\sum_{j=1}^m a_j\phi_j', \phi_i') + (\sum_{j=1}^m a_j\phi_j, \phi_i) + (f, \phi_i) = 0$$

and on matrix form

$$A^G a = -g$$

where

- $A_{ij}^G = (\phi_j', \phi_i') + (\phi_j, \phi_i)$
- $a = [a_1, \ldots, a_m]^T$
- $g = [\overline{f_1}, \ldots, \overline{f_m}]^T$ where $\overline{f_i} = (f, \phi_i)$

ODE-BVP becomes an **algebraic equation system**

METHODS OF WEIGHTED RESIDUALS

Apply the pp-approximation in the ODE which creates a **residual**

$$\sum_{j=1}^m a_j[\phi_j''(x) - \phi_j(x)] - f(x) = R(x)$$

Let us tune the unknown coefficients a in order to fulfil the integral of **weighted residuals**

$$\int_0^1 W_k R(x) = 0$$

How to choose the weights, W?

- $W_k = \delta(x - x_k)$, pulses at collocation points
  $\Rightarrow$ *Collocation method.*
- $W_k = \phi(x)$, weights are the basis functions
  $\Rightarrow$ *Galerkin method.*
- $W_k = \frac{\partial R}{\partial a}$
  $\Rightarrow$ *Least-square minimization.*
- Other MWR are; *method of moments, subdomain method* etc

PIECEWISE POLYNOMIALS I
Definitions

Interval partition π

$$a = x_1 < x_2 < \ldots < x_{\ell+1} = b$$

Piecewise polynomial (pp) function, $F(x)$

$$F(x) = P_j(x),$$
$$x_j < x < x_{j+1}, \quad j = 1, \ldots, \ell$$

- $P_j(x)$ where $j = 1, \ldots, \ell$, are any sequence of ℓ **polynomials**.
- $P_j(x)$ are of **order** k (degree $\leq k - 1$)
- $\ell + 1$ **breakpoints** of $F(x)$ at x_j

By convention

$$F(x) = \begin{cases} P_1(x), & x \leq x_1 \\ P_\ell(x), & x \geq x_{\ell+1} \end{cases}$$

and with right continuity $F(x_i) = P_i(x_i)$

PIECEWISE POLYNOMIALS II
Basis Functions

Assume a set of functions

$$S = \{\lambda_j(x); j = 1, \ldots, L\}$$

The class of functions $\wp$ is the set of all function

$$f(x) = \sum_{j=1}^{L} \alpha_j \lambda_j(x)$$

$\wp$ is called a **linear function space**.

If λ_j in S are linearly independent then

- S is a basis for $\wp$
- L is the dimension of the space $\wp$
- λ_j is a **basis function**

$\wp_k(\pi)$ is the set of **all pp-function of order** k with the dimension

$$\dim \wp_k(\pi) = k\ell$$

PIECEWISE POLYNOMIALS III
Continuity

Let $\nu = \{\nu_j; j = 2, \ldots, \ell\}$ such that

$$\text{jump}_{x_j} \frac{d^{i-1}}{dx^{i-1}}[f(x)] = 0$$

(for $i = 1, \ldots, \nu_j$ and $j = 2, \ldots, \ell$)

The jump condition is defined as

$$\text{jump}_{x_j} \tfrac{d^{i-1}}{dx^{i-1}}[f(x)] = \tfrac{d^{i-1}}{dx^{i-1}}[f(x_j^+)] - \tfrac{d^{i-1}}{dx^{i-1}}[f(x_j^-)]$$

ν specifies the continuity of f and its derivatives at the breakpoints.

$\wp_k^\nu(\pi)$ is a subspace of $\wp_k(\pi)$ that satisfies the jump condition specified by ν

PIECEWISE POLYNOMIALS IV
Linear Basis Functions

$\wp_2^1(\pi)$ is a space of piecewise linear functions.

$$f(x) = \sum_{j=1}^{\ell+1} \alpha_j w_j$$

- see table 3.1
- see figure 3.2 (a)
- **no continuity at breakpoints** for derivatives of $f(x)$

PIECEWISE POLYNOMIALS V
Hermite Cubic Basis Functions

The Hermite cubic space, $\wp_4^2(\pi)$, has a basis of *value*, v_j, and *slope*, s_j, functions.

$$f(x) = \sum_{j=1}^{\ell+1} [\alpha_j^1 v_j + \alpha_j^2 s_j]$$

- see table 3.2
- see figure 3.2 (b)
- **continuity of the first derivative**

PIECEWISE POLYNOMIALS VI
B-splines

For higher continuity specifications B-splines are used for $\wp_k^\nu(\pi)$.

$$f(x) = \sum_{j=1}^{N} \alpha_j B_j(x)$$

where $N = \dim \wp_k^\nu(\pi) (< k\ell)$

A **spline** f is specified by its **knot** sequence t and by its B-spline coefficient sequence a.
Properties:

- Each $B_i(x) = 0$ when $x < t_i$ or $x > t_{i+k}$ (local support)
- $\sum_{alli} B_i(x) = 1$
- Each B_i satisfies $0 \geq B_i(x) \geq 1$ in the subinterval
- can have **continuity up to the** $(\nu - 1)$ **derivative**
- *see appendix D for more details*

PIECEWISE POLYNOMIALS VII
B-splines cont.

A **spline** f is specified by its **knot** sequence t and by its B-spline coefficient sequence a.

- the coefficients a are **control points** for the curve
- a spline is pp of **order k**
- knots are almost as **breakpoints** t
  (but knots are different in that they may be repeated)
- **knot multiplicity + condition multiplicity = order**
- *see figure D.2 in appendix D*

GALERKIN METHOD I
Linear Problems

Example 1:
Second-order ODE with homogenous Dirichlet BCs.

$$\begin{aligned} -y''(x) &= 1 \\ y(0) &= 0 \\ y(1) &= 0 \end{aligned}$$

Set up the matrix problem, $A^G a = -g$, using $\wp_2^1(\pi)$ (piecewise linear function space).

$$u(x) = \sum_{j=1}^{\ell+1} a_j w_j$$

For homogenous BCs the first and last basis functions are excluded, $u(x) = \sum_{j=1}^{\ell-1} a_j w_j$.

GALERKIN METHOD II
Linear Problems cont.

The elements in the A^G-matrix are

$$A_{ij}^G = \int_0^1 \phi_j' \phi_i' dx$$

Each basis function is supported on only two subintervals $A_{ij}^G = 0, \quad |i-j| > 1$

A^G becomes tridiagonal with the following diagonal elements

$$A_{ii}^G = \int_0^1 (\phi_i')^2 dx =$$
$$\int_{x_{i-1}}^{x_i} [\tfrac{1}{x_i - x_{i-1}}]^2 dx + \int_{x_i}^{x_{i+1}} [\tfrac{-1}{x_{i+1} - x_i}]^2 dx = \tfrac{1}{h_i} + \tfrac{1}{h_{i+1}}$$

and following super diagonal elements

$$A_{i,i+1}^G = \int_0^1 \phi_i' \phi_{i+1}' dx =$$
$$\int_{x_i}^{x_{i+1}} [\tfrac{-1}{x_i - x_{i-1}}][\tfrac{1}{x_i - x_{i-1}}] dx = -\tfrac{1}{h_{i+1}}$$

GALERKIN METHOD III
Linear Problems cont.

and to follow up the last page the subdiagonal elements becomes

$$A^G_{i,i-1} = -\frac{1}{h_i}$$

The g-vector are $[\overline{f}_1, \ldots, \overline{f}_m]^T$

$$\overline{f}_i = \int_0^1 f(x)\phi_i(x)dx = \int_0^1 \phi_i(x)dx = \frac{1}{2}(h_i + h_{i+1})$$

GALERKIN METHOD IV
Linear Problems cont.

The linear ODE-BVP problem are written as

$$A^G a = -g$$

where the a:s are the unknowns which results in

$$\begin{bmatrix} (\frac{1}{h_1}+\frac{1}{h_2}) & -\frac{1}{h_2} & 0 & \ldots & 0 \\ -\frac{1}{h_2} & (\frac{1}{h_2}+\frac{1}{h_3}) & -\frac{1}{h_3} & \ldots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \ldots & & -\frac{1}{h_{\ell-2}} & (\frac{1}{h_{\ell-2}}+\frac{1}{h_{\ell-1}}) \end{bmatrix}$$

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{\ell-1} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(h_1+h_2) \\ \frac{1}{2}(h_2+h_3) \\ \vdots \\ \frac{1}{2}(h_{\ell-2}+h_{\ell-1}) \end{bmatrix}$$

and for a uniform grid the equation system becomes

$$\frac{1}{h}\begin{bmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \ldots & & -1 & 2 \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{\ell-1} \end{bmatrix} = h\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

which *corresponds to a second-order correct finite difference method.*

GALERKIN METHOD V
Accuracy and Nonlinear Problems

In general, the Galerkin method using $\wp_k^\nu(\pi)$ gives an error such that

$$||y - u|| \leq Ch^k$$

which means that the **order of accuracy** of the Galerkin method are equal to the **order of the basis functions**

Nonlinear ODEs with homogenous BCs

$$y'' = f(x, y, y')$$

Generates a nonlinear matrix problem like

$$Aa + H(a) = 0$$

which must be solved by a nonlinear equation solver (for sparse systems).

GALERKIN METHOD VI
Boundary Conditions

Assume an ODE-BVP

$$(a(x)y'(x))' + b(x)y(x) + c(x) = 0$$
$$y(0) = \Psi_1, \qquad y(1) = \Psi_2$$

which results in the integral using B-splines

$$\int_0^1 [(a(x)y'(x))' - b(x)y(x) - c(x)]B_i(x)dx = 0$$

Rewrite to weak form using

$$\int_0^1 (a(x)y'(x))'B_i(x)dx = [a(x)y'(x)B_i(x)]_0^1 - \int_0^1 a(x)y'(x)B_i'(x)dx$$

and the *weak form* now becomes

$$[a(x)y'(x)B_i(x)]_0^1 - (ay', B_i') + (by, B_i) + (c, B_i) = 0$$

The BC defines the first term

GALERKIN METHOD VII
Boundary Conditions cont.

For B-splines the following is valid

$$B_1(0) = 1, \qquad \sum_{j=2}^{N} B_j(0) = 0$$
$$B_N(0) = 1, \qquad \sum_{j=1}^{N-1} B_j(0) = 0$$

Dirichlet: this gives direct that $a_1 = \Psi_1$ and $a_N = \Psi_2$ which is used in the futher evaluation of the integrals. (number of unknowns becomes $N - 2$)

Neumann: requires a manipulation if the BC before using it in the integrals.
(number of unknowns becomes N)

COLLOCATION METHOD I
General Description

Consider the nonlinear ODE-BVP

$$y'' = f(x, y, y')$$
$$\eta_1 y + \beta_1 y' = \gamma_1, \qquad x = a$$
$$\eta_2 y + \beta_2 y' = \gamma_2, \qquad x = b$$

Use a PP-approximation in $\wp_k^\nu(\pi)$.

Collocation defines $\{\alpha_j; j = 1, \ldots, N\}$ by **satisfying the ODE at N points**
How many and where?

- $k = 4$ and $\nu = 2$ then $N = 2\ell + 2$
- Satisfy **two BC**
- Satisfy ODE at **2 points in each ℓ subinterval**
- **Optimal position** for 2 $(k - M)$ Gaussian points are $-\frac{1}{\sqrt{3}}$ and $\frac{1}{\sqrt{3}}$ round the middle of the subinterval.

COLLOCATION METHOD II
Linear Problems and Accuracy

A linear problem results in a linear equation system

$$A^C \alpha = f$$

with the dimension

- $N = 2\ell$ for Dirichlet
- $N = 2\ell + 1$ for one Dirichlet and one Neumann (or Robin)
- $N = 2\ell + 2$ for Neumann (or Robin)

and with the **accuracy** same as **the order of the basis function** (same as for Galerkin).

FINITE ELEMENT METHODS in MATLAB

- **Spline toolbox** handles both pp-functions and B-splines. Can be used to solve nonlinear ODE-BVP using the **collocation method**
  - `spmak` makes a spline
  - `spcrv` display a spline
  - `spcol` spline collocation
- Partial Differential Equations can be handles in MATLAB by the **PDE toolbox** which uses the **Galerkin method**.
  Will be discussed in the following lectures and used in exercise 5.

SUMMARY

Steady-State Simulation of ODEs:

- *Initial-Value Methods for BVP*
- *Finite Difference Methods*
- Finite Element Methods
  - Method of Weighted Residuals is the general framework
  - Uses piecewise basis functions as approximations of the solution
  - Galerkin match the ODE with an integral
  - Collocation satisfy the ODE at a number of points
  - Hard to use by hand. Use computer tools!
  - MATLAB support: Spline and PDE toolboxes

SIMULATION OF PROCESS SYSTEMS

Dynamic Simulation of Partial Differential Equations

PART I: METHOD OF LINES

Literature: *Davis, chap 4*

Bernt Nilsson, Chemical Engineering II
Bernt.Nilsson@chemeng.lth.se

DYNAMIC SIMULATION OF PDES in One Space Dimension

Contents:

- Classifications of PDEs
- Method of Lines (MOL)
- Parabolic PDEs
  - Finite Difference based MOL
  - Finite Element based MOL
- Hyperbolic PDEs

PARTIAL DIFFERENTIAL EQUATIONS

Linear second order *partial differential equation*

$$a\frac{\partial^2 w}{\partial x^2} + 2b\frac{\partial^2 w}{\partial x \partial y} + c\frac{\partial^2 w}{\partial^2 y} + d\frac{\partial w}{\partial x} + e\frac{\partial w}{\partial y} + fw = g$$

- **parabolic** PDE: $b^2 - 4ac = 0$
  - also called *Diffusion Problem* or *Heat Equation*
  - first order in time and second order in space
- **hyperbolic** PDE: $b^2 - 4ac > 0$
  - also called the *Wave Equation*
  - second order in time and space or
  - (first order in time and space)
- **elliptic** PDE: $b^2 - 4ac < 0$
  - also called the *Laplace's Equation*
  - steady-state problems

DYNAMIC PDEs in One Space Dimension

Parabolic PDE: Dynamic diffusion

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial^2 x}$$

First-order Hyperbolic PDE: Perfect plug-flow model

$$\frac{\partial T}{\partial t} = -v\frac{\partial T}{\partial x}$$

Second-order Hyperbolic PDE: Gas dynamics (small disturbances)

$$\frac{\partial^2 v}{\partial t^2} = c^2\frac{\partial^2 v}{\partial^2 x}$$

Parabolic-Hyperbolic PDE: Dynamic dispersion model of a tubular reactor in one space (axial) dimension

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial^2 x} - v\frac{\partial c}{\partial x} - kc$$

METHOD OF LINES I
General Description

In general:

- Solution to an *algebraic equation* is one (or many) **point**
- Solution to an *ODE* is a function of one variable, **a line** (not a strait line).
- Solution to a *PDE* in two variables is a **surface**

Method of Lines discretize the PDE into a set of ODEs and the solution will be a set of lines.

- Finite difference in space
- Finite element in space of the time-variable solution

The surface is approximated with a set of lines.

METHOD OF LINES II
General Description

Consider a simple diffusion problem

$$\frac{\partial w}{\partial t} = D\frac{\partial^2 w}{\partial^2 x}$$
$$w(0,t) = \alpha, \quad w(1,t) = \beta$$
$$w(x,0) = \alpha + (\beta - \alpha)x$$

Define a **uniform grid** and make a **centered-difference** approximation of the space derivative $w(x_i,t) \approx u_i$

$$\frac{du_i}{dt} = D\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

- This results in a set of ODEs.
- Boundary conditions in space
- Initial conditions in time

METHOD OF LINES III
Example

A uniform grid and a centered-difference approximation of the space derivative on the previous diffusion problem using the following BCs

- BC: $u_1 = \alpha$ and $u_{N+1} = \beta$
- IC: $u_i = \alpha + (\beta - \alpha)x_i$ at $t = 0$

results in

$$\frac{du_2}{dt} = \frac{D}{h^2}(u_3 - 2u_2 + \alpha)$$
$$\frac{du_i}{dt} = \frac{D}{h^2}(u_{i+1} - 2u_i + u_{i-1})$$
$$\frac{du_N}{dt} = \frac{D}{h^2}(\beta - 2u_N + u_{N-1})$$

and on matrix form

$$\frac{du}{dt} = \frac{D}{h^2}\begin{bmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ \vdots & & & & \\ 0 & \dots & 0 & 1 & -2 \end{bmatrix} u + \frac{D}{h^2}\begin{bmatrix} \alpha \\ 0 \\ \vdots \\ \beta \end{bmatrix}$$

FINITE DIFFERENCES I
Forward Euler Approximation

Make a *Forward Euler Approximation* of the time derivative

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{D}{h^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})$$

Rewrite as a difference equation

$$u_{i,j+1} = (1 - 2D\frac{\Delta t}{h^2})u_{i,j} + D\frac{\Delta t}{h^2}(u_{i+1,j} + u_{i-1,j})$$

Explicit integration method:

- eigenvalue: $\lambda_i = 1 - 2D\frac{\Delta t}{h^2}$
- stability: $|\lambda_i| < 1$

$\Rightarrow$ $\frac{\Delta t}{h^2} < \frac{1}{D}$ (Courant condition)

- first order accurate in time and second in space.

FINITE DIFFERENCES II
Backward Euler Approximation

Make a *Backward Euler Approximation* of the time derivative

$$\frac{u_{i,j+1}-u_{i,j}}{\Delta t} = \frac{D}{h^2}(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1})$$

Rewrite on matrix form ($\tau = D\frac{\Delta t}{h^2}$)

$$\begin{bmatrix} 1+2\tau & -\tau & 0 & \dots & 0 \\ -\tau & 1+2\tau & -\tau & \dots & 0 \\ \vdots & & & & \\ 0 & \dots & 0 & -\tau & 1+2\tau \end{bmatrix} u_{j+1} =$$

$$u_j + \tau \begin{bmatrix} u_{1,j+1} \\ 0 \\ \vdots \\ u_{N+1,j+1} \end{bmatrix}$$

Implicit integration method:

- *Linear:* solve $Au = b$ in each iteration step
- *Nonlinear:* iteration to solve $f(u) = 0$
- **Unconditional stable**
- first order accurate in time and second in space.

FINITE DIFFERENCES III
Crank-Nicolson Method

Make a trapezoidal approximation in time (order 2)

$$\frac{du_i}{dt}_{j+\frac{1}{2}} = \frac{u_{i,j+1}-u_{i,j}}{\Delta t}$$

Together with centered-difference space approximation this results in the following

$$\begin{aligned} \frac{u_{i,j+1}-u_{i,j}}{\Delta t} \quad = \quad & \frac{D}{2h^2}((u_{i+1,j+1} + u_{i+1,j}) \\ & -2(u_{i,j+1} + u_{i,j}) \\ & +(u_{i-1,j+1} + u_{i-1,j})) \end{aligned}$$

Implicit integration method:

- **Unconditional stable**
- It is called the *Crank-Nicolson method*
- The *Theta method* is a generalization of the Euler methods and CN method.
- second order accurate in time and space.

METHOD OF LINES IV
High-Order Methods

The Euler and Crank-Nicolson methods are interesting of *historical* reasons and that they can be *numerically analysed*.

In practice **high-order ODE solvers**, presented earlier in the course, (chap 1) **are used** to solve the set of ODEs.

- one-step or multistep solvers
- high or low order methods
- explicit or implicit solvers
- *Note different orders of accuracy in time and space*

The selection rules for ODE-solvers should also be used in MOL!

METHOD OF LINES V
Nonlinear, Stiff and Sparse

Nonlinear problems occur frequently. Can cause problems for boundary value approximations and in discretization.

Stiff problems occur often. Stiffness can occur due to discretization effects.
Ex: unitform discretization of the radius of a sphere results in smaller and smaller volumes.

Sparse problems occur always. The space discretization results (almost) always in a sparse problem. *Sparseness is important for implicit solvers.*

METHOD OF LINES VI
Boundary and Initial Conditions

Mass transfer in a pipe is described by a parabolic PDE, *see p. 137.* (Note two dimensions in space)

$$
\begin{aligned}
v\frac{\partial y_A}{\partial t} &= \frac{D}{r}\frac{\partial}{\partial r}(r\frac{\partial y_A}{\partial r}) \\
&\frac{\partial y_A}{\partial r}(0,z) = 0 \\
&-D\frac{\partial y_A}{\partial r}(r_w,z) = k_g(y_A - y_A^o) \\
&y_A(r, z=0) = y_{A_1}
\end{aligned}
$$

Inconsistency in the initial and boundary conditions

1. Use the IC in BC at the wall
   $-D\frac{\partial y_A}{\partial r} = k_g(y_{A_1} - y_A^o)$
   This indicated that there is a gradient i the r-direction
2. IC defines y_A to be contant in r at $z = 0$

$\Rightarrow$ **1 and 2 are inconsistent!**

Recommendation: choose the boundary condition and change the inital condition to $y_A = y_A^o$ at the boundary.

MATLAB I
Finite Difference based MOL

- No **direct** support for space discretization

$$
\frac{\partial^2 c}{\partial x^2} \approx \begin{bmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ \vdots & & & & \\ 0 & \dots & 0 & 1 & -2 \end{bmatrix} c
$$

- Describe ODEs (in M-file) using sparse matrices

```
>> e=ones(n,1)
>> A=spdiags([e -2*e e],-1:1,n,n)
>> dx=A*x
```

- Solvers for ODE on state-space form (ODE suite)
  - `ode23s, ode15s` can use information about the sparseness of the Jacobian.
    `>> ode15s('fun',[t0 t1],Y0,'sparseJ',S)`

FINITE ELEMENTS I
Method of Lines

Finite element methods for **static problems** are based on the piecewise polynomial (pp) approximation of the solution in space.

$$u(x) = \sum_{j=1}^{m} \alpha_j \phi_j(x)$$

In **dynamic problems** basis functions in the pp-approximation are assumed to be time independent and that the dynamics is captured by the unknown coefficients.

$$u(x,t) = \sum_{j=1}^{m} \alpha_j(t) \phi_j(x)$$

Note:

- *Time independent basis functions*
- *Time-variable α-coefficients*

FINITE ELEMENTS II
Galerkin-MOL

Assume the following piecewise polynomial (pp) approximation of the solution

$$u(x,t) = \sum_{j=1}^{m} \alpha_j(t) \phi_j(x)$$

and consider the parabolic PDE

$$\frac{\partial w}{\partial t} = \frac{\partial}{\partial x}[a(x,w)\frac{\partial w}{\partial x}]$$

The **weak form** becomes

$$\left(a(x,w)\frac{\partial w}{\partial x}, \phi_i'\right) + \left(\frac{\partial w}{\partial t}, \phi_i\right) = 0$$

FINITE ELEMENTS III
Galerkin-MOL

The MOL formulation of the problem becomes

$$\left(a(x, \sum_{j=1}^{m} \alpha_j \phi_j(x)) \sum_{j=1}^{m} \alpha_j \phi_j'(x), \phi_i' \right) + \left(\sum_{j=1}^{m} \alpha_j' \phi_j(x), \phi_i \right) = 0$$

Write the problem on matrix form and we get an IVP with a set of ODEs

$$B\alpha'(t) + D(\alpha)\alpha(t) = 0$$

where

- $D_{i,j} = (a\phi_j', \phi_i')$
- $B_{i,j} = (\phi_j, \phi_i)$
- $B\alpha(0) = \beta = [(w_0, \phi_1), \ldots, (w_0, \phi_m)]^T$

FINITE ELEMENTS IV
Collocation-MOL

Consider a nonlinear parabolic PDE

$$\frac{\partial w}{\partial t} = f(x, t, w, \frac{\partial w}{\partial x}, \frac{\partial^2 w}{\partial x^2})$$

Make the following pp-approximation

$$u(x,t) = \sum_{j=1}^{m} \alpha_j(t)\phi_j(x)$$

Used the pp-approximation in the PDE and we get the following equation

$$\sum_{j=1}^{m} \alpha_j'(t)\phi_j(x_i) =$$

$$f(x_i, t, \sum_{j=1}^{m} \alpha_j(t)\phi_j(x_i), \sum_{j=1}^{m} \alpha_j(t)\phi_j'(x_i), \sum_{j=1}^{m} \alpha_j(t)\phi_j''(x_i))$$

FINITE ELEMENTS V
Collocation-MOL

Write the problem on matrix form and we get

$$A\alpha'(t) = F(t, \alpha)$$

- ϕ is local and A becomes **sparse**
- order of accuracy is $o(\Delta t^p + h^k)$
  - k is the order of the basis function $\wp_k^\nu$
  - p is the order of accuracy in the integration method

MATLAB II
Finite Element based MOL

- ODE solver for FE-MOL

$$Ay' = f(y)$$

  - A is sparse and should not be inverted directly.
  - ODE suite: `'mass'`-option to specify A.

- PDE-toolbox
  - Can solve *Parabolic* and *Hyperbolic* PDEs
  - `parabolic` and `hyperbolic` uses Galerkin-MOL
    - *(no convection terms in current version)*

HYPERBOLIC PDE I
Finite Difference

First-Order hyperbolic PDE

$$\frac{\partial c}{\partial t} = -v\frac{\partial c}{\partial x}$$

has the following simple solution

$$c(x,t) = f(t - \frac{x}{v})$$

- Propogates finite discontinuities along the tube.
- Finite difference approximations of second and fourth order creates **numerical oscillations**.
- *One of the most difficult classes of PDE to integrate numerically.*

HYPERBOLIC PDE II
Up-wind Approximations

To solve numerical oscillations there are a number of approaches

- First order **Up-wind approximation** creates *numerical diffusion*

$$\frac{\partial T}{\partial x} = \frac{T_{i,j} - T_{i-1,j}}{h}$$

- Recommendation by Scheisser is a combination of *five-point* approximation and a *three-point up-wind* approximation
  W.E. Schiesser: The Numerical Method of Lines.

HYPERBOLIC PDE III
Hopscotch Methods

Approximation of hyperbolic PDEs (waves) are close to the stability limit.
(Undamped waves oscillates)

Lax-Wendroff methods add an artificial second order term to the hyperbolic PDE which results in a hyperbolic-parabolic PDE

$$\frac{\partial c}{\partial t} = -v\frac{\partial c}{\partial x} - \frac{k}{2}\frac{\partial^2 c}{\partial x^2}$$

- The diffusion term damp oscillations
- k is used to tune stability.

(Note: not directly a method of lines)

METHOD OF LINES VII
Adaptive Grids

- **Moving grid** is used to move the grid points to the parts of the solution that have large changes. (can be viewed as changing dynamics in the ODEs)
- **Regridding** is used to change the number of grid points, change the number of ODEs.
  If the second order derivative in a grid point becomes large, new gridpoints are inserted.
  (can not use ordinary ODE solvers)

SUMMARY

Dynamic Simulation of PDEs in one space dimension:

- Classification of PDEs
- Method of Lines (MOL)
  - Discretize the solution surface in a set of solution lines
  - Finite Difference-MOL
  - Finite Element-MOL
- Inconsistent boundary and initial conditions can occur
- Parabolic PDEs
  - Can be solved successfully by MOL
  - MATLAB: FD-MOL using ODE suite
  - MATLAB: Galerkin-MOL using PDE-toolbox
- Hyperbolic PDEs
  - Can be hard to integrate.
    (propagation of discontinueties)

SIMULATION OF PROCESS SYSTEMS

Simulation of Partial Differential Equations

PART II: TWO SPACE DIMENSIONS

Literature: *Davis, chap 5*

Bernt Nilsson, Chemical Engineering II
Bernt.Nilsson@chemeng.lth.se

SIMULATION OF PDES in Two Space Dimension

Contents:

- Classifications of PDEs (again)
- Steady-State Simulation or Elliptic PDEs
  - Finite Difference Methods
  - Finite Element Methods
- Irregular Boundaries
- Dynamic Simulation of PDEs or Parabolic and Hyperbolic PDEs
- MATLAB PDE-toolbox
- Summary of Course
- Beyond the Scope

PARTIAL DIFFERENTIAL EQUATIONS

Linear second order *partial differential equation*

$$a\frac{\partial^2 w}{\partial x^2} + 2b\frac{\partial^2 w}{\partial x \partial y} + c\frac{\partial^2 w}{\partial^2 y} + d\frac{\partial w}{\partial x} + e\frac{\partial w}{\partial y} + fw = g$$

- **parabolic** PDE: $b^2 - 4ac = 0$
  - also called *Diffusion Problem* or *Heat Equation*
  - first order in time and second order in space
- **hyperbolic** PDE: $b^2 - 4ac > 0$
  - also called the *Wave Equation*
  - second order in time and space or
  - (first order in time and space)
- **elliptic** PDE: $b^2 - 4ac < 0$
  - also called the *Laplace's Equation*
  - steady-state problems

PDEs in 2 and 3 Space Dimensions

Elliptic PDE: Steady-State diffusion (in 2 dimensions)

$$0 = D(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2})$$

Parabolic PDE: Dynamic diffusion (in 2 dimensions)

$$\frac{\partial c}{\partial t} = D(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2})$$

Hyperbolic PDE: Gas dynamics (in 3 dimensions)

$$\frac{\partial^2 v}{\partial t^2} = c^2(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2})$$

PDEs in 2 and 3 Space Dimensions

Simulation methods

- **Elliptic PDE**: Steady-State Problems
  - Finite Difference Methods
  - Finite Element Methods
- **Parabolic PDE**: Method of Lines with
  - Finite Differences
  - Finite Elements
- **Hyperbolic PDE**:
  - MOL with Up-wind corrections
  - Lax-Wendroff (hopscotch) methods

In other words:
Generalization of previously discussed methods.

ELLIPTIC PDE in 2 space dimensions

One example of an elliptic PDE in $x - y$ plane

$$\frac{\partial}{\partial x}[a_1(x,y)\frac{\partial w}{\partial x}] + \frac{\partial}{\partial y}[a_2(x,y)\frac{\partial w}{\partial y}] = f(x,y,w,\frac{\partial w}{\partial x},\frac{\partial w}{\partial y})$$

with associated boundary condition (BC):

- Dirichlet BC: $w = f(x,y)$
- Neumann BC: $\frac{\partial w}{\partial n} = g(x,y)$
  ($\frac{\partial}{\partial n}$ refers to the normal direction of the boundary.)
- Robin BC: $\alpha(x,y)w + \beta(x,y)\frac{\partial w}{\partial n} = \gamma(x,y)$
  (or generalized Neumann)

FINITE DIFFERENCE METHOD I Laplace's Equation

Laplace's equation in a square plane

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = 0$$

A uniform grid with mesh-size h results in $(N-1)^2$ internal grid points.

Centered-difference approximation:

$$\frac{1}{(\Delta x)^2}[u_{i+1,j} - 2u_{i,j} + u_{i-1,j}] +$$
$$\frac{1}{(\Delta y)^2}[u_{i,j+1} - 2u_{i,j} + u_{i,j-1}] = 0$$

For the square problem we get ($\Delta x = \Delta y = h$)

$$u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1} = 0$$

(order of accuracy $o(h^2)$.)

FINITE DIFFERENCE METHOD II Dirichlet Problem

Dirichlet BC means **known values** on the boundary

$$u_{i,j} = f(x_i, y_j)$$

where (x_i, y_j) is **on boundary** and together with the difference approximation we get following matrix problem

$$Au = f_D$$

- Space discretization is captured by A
- The unknown **internal grid point** values are found in the u-vector.
- The Dirichlet BC is described in f_D.
- Number of equations are $(N-1)^2$
- *See the matrices on page 179.*

FINITE DIFFERENCE METHOD III
Neumann Problem

Discretize the Neumann BC and use method of false boundaries (square plane).

$$\frac{1}{2h}[u_{-1,j} - u_{1,j}] = g_{0,j}, \qquad \frac{1}{2h}[u_{i,-1} - u_{i,1}] = g_{i,0}$$

and together with the difference approximation we get following matrix problem

$$A_N u = 2hg$$

- Space discretization together with Neumann BC is captured by A_N
- The unknown **internal and boundary grid point** values are found in the u-vector.
- The right hand side of the Neumann BC is described in g.
- Note that mesh-size h becomes a parameter.
- Number of equations are $(N+1)^2$
- *See the matrices on page 180.*

FINITE DIFFERENCE METHOD IV
Example 1

Laplace's equation describing steady-state heat conduction. Assume a square plate and uniform grid.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

The boundary conditions are

$$T(0,y) = T_1, \qquad T(1,y) = T_2$$
$$\frac{\partial T}{\partial y}(x,0) = 0, \qquad \frac{\partial T}{\partial y}(x,1) = k[T(x,1) - T_2]$$

- Dirichlet BC in the x-dimension
- Neumann BC at $y = 0$
- Robin BC at $y = 1$
- *See Figure 5.1 on page 182.*
- *See the matrices on page 183.*

FINITE DIFFERENCE METHOD V
Practical Problems

Elliptic PDE in $x - y$ plane

$$\frac{\partial}{\partial x}[a_1(x,y)\frac{\partial w}{\partial x}] + \frac{\partial}{\partial y}[a_2(x,y)\frac{\partial w}{\partial y}] = f(x,y,w,\frac{\partial w}{\partial x},\frac{\partial w}{\partial y})$$

- **Variable Coefficients** change the values of the elements in the A-matrix of the linear equation system.
- **Nonlinear Terms**, as f, creates a nonlinear equation system and requires a *nonlinear solver and iteration*.
- **Nonuniform Grids** require careful discretization. (straightforward centered-differences not valid.)

FINITE DIFFERENCE METHOD VI
Irregular Boundaries

Dirichlet Conditions

1. Unequal mesh spacing.
   - Grid points on the boundary
2. Boundary interpolation.
   - interpolation of the *real boundary* to a new boundary on the uniform grid points

- *See Figure 5.4 on page 190.*

Neumann Conditions

- Line segment approximation.
  - Nonuniform grid points on the boundary
  - Approximate boundary segments as straight lines
- *See Figure 5.5 on page 191.*

FINITE ELEMENT METHOD I
General Description

Assume the PDE, $\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = -f(x,y)$, with the **piecewise polynomial** (pp) approximation of the solution in the $x-y$ plane

$$u(x,y) = \sum_{j=1}^{m} \alpha_j \phi_j(x,y)$$

Collocation method satisfies the PDE at m collocation points

$$A^C \alpha = -f$$

- $A^C_{ij} = \frac{\partial^2 \phi_j}{\partial x^2}(x_i, y_j) + \frac{\partial^2 \phi_j}{\partial x^2}(x_i, y_j)$
- $\alpha = [\alpha_1, \ldots, \alpha_m]^T$
- $f = [f(x_1, y_1), \ldots, f(x_m, y_m)]^T$

Galerkin method satisfies the integral of the weak form of the PDE

$$A^G \alpha = g$$

- $A^G_{ij} = (\nabla \phi_i, \nabla \phi_j)$
- $\alpha = [\alpha_1, \ldots, \alpha_m]^T$
- $g = [g_1, \ldots, g_m]^T$ where $g_i = (f, \phi_i)$

FINITE ELEMENT METHOD II
Basis Functions

Basis functions can be created by straightforward generalization

- Tensor product of one-dimensional basis functions
- Creates rectangles of piecewise polynomials
- Linear basis function *see page 201 and Figure 5.7*
- Hermite cubic space, $\wp^2$, results in **4 collocation points**
  $4(N_x+1)(N_y+1)$ unknown coefficients

FINITE ELEMENT METHOD III
Collocation

Elliptic PDE in **Example 3** defined on a square

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = \Phi$$

with homogenous Dirichlet on one side and Neumann on the other in both x and y-direction. Make a partition of the square into 4 subrectangles and use Hermite cubic space $\wp^2$

- $4(N_x+1)(N_y+1) = 36$ unknown coefficients
  - 4 collocation points in each subrectangles results in 16 equations
  - 8 internal boundary equations (4 * 2)
  - 12 boundary corner equations (4 * 3)
  - *See Figure 5.6*

Put up all these equation and the result is an linear equation system

$$A^C a = \Phi$$

FINITE ELEMENT METHOD IV
Galerkin

Consider a similar problem

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = -1$$

with homogenous Dirichlet which gives the weak form

$$\iint_R [\frac{\partial w}{\partial x}\frac{\partial \phi_i}{\partial x} + \frac{\partial w}{\partial y}\frac{\partial \phi_i}{\partial y}] dx\, dy = \iint_R \phi_i \, dx\, dy$$

Use the linear pp-approximation on a 2x2-partition of the square *(3x3 breakpopints)*

$$u(x,y) = \sum_{i=1}^{3}\sum_{j=1}^{3} u(x_i, y_j)\omega_{i,j}$$

and the resulting problem becomes,

$$A_{22} u_2 = g_2$$

where

$A_{22} = \iint [\frac{\partial \omega_2}{\partial x}\frac{\partial \omega_2}{\partial x} + \frac{\partial \omega_2}{\partial y}\frac{\partial \omega_2}{\partial y}] dx\, dy, \quad g_2 = \iint \omega_2 \, dx\, dy$

FINITE ELEMENT METHOD V
Galerkin cont.

Instead of integration of the whole domain, one solves the problem **element by element**. and the resulting problem becomes, *(see Figure 5.8)*

$$A_{22}u_2 = \sum_{e_i=1}^{4} A_{22}^{e_i}u_2 = \sum_{e_i=1}^{4} g_2^{e_i} = g_2$$

and for **element 1** we get, $h = 0.5$
(from the book, check these calculations, wrong?)

$$A_{22}^1 = \frac{1}{h^4}\int_{0.5}^{1}\int_{0.5}^{1}[(\frac{\partial\omega_2}{\partial x})^2 + (\frac{\partial\omega_2}{\partial y})^2]dx\,dy =$$
$$\frac{1}{h^4}\int_{0.5}^{1}\int_{0.5}^{1}[(1-y)^2 + (1-x)^2]dx\,dy = \frac{2}{3}$$

and

$$g_2^1 = \frac{1}{h^2}\int_{0.5}^{1}\int_{0.5}^{1}(1-x)(1-y)\,dx\,dy = \frac{h^2}{4}$$

FINITE ELEMENT METHOD I
Triangular Elements

The **Galerkin method** can be formulated on triangular elements

- Irregular regions can be covered by triangular elements, *see Figure 5.9*
- TN vertices *("internal nodes")*
- associated basis function, *see Figure 5.10*

Example 5 shows the use of triangulation. Only one internal vertex, all other on the boundary.

PARABOLIC PDEs
Method of Lines

Consider the unsteady-state diffusion problem in the plane

$$\frac{\partial w}{\partial t} = D(\frac{\partial^2 w}{\partial^2 x} + \frac{\partial^2 w}{\partial^2 x})$$

Define a **uniform grid** and make a **centered-difference** approximation of the space derivatives $w(x_i, y_j, t) \approx u_{ij}$

$$\frac{du_{ij}}{dt} = \frac{D}{h^2}((u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + (u_{i,j+1} - 2u_{i,j} + u_{i,j-1}))$$

- *Straightforward generalization of Method of Lines*
- This results in a set of ODEs.
- Boundary conditions in space
- Initial conditions in time
- Galerkin-MOL is just a generalization of 1-dimensional case

MATLAB
PDE-toolbox

PDE-toolbox in MATLAB is based on the **finite element method**

- **Triangular elements**
- **Piecewise linear** basis functions
- **Galerkin** method

The toolbox can be **command-driven** or be used though an **interactive graphical interface**, `pdetool`.

- Geometry and boundary conditions defined in M-files or by **graphics**
- `assempde` generates/solves $KU = F$ (elliptic)
- `parabolic` solves $M\frac{d}{dt}U + KU = F$
- `hyperbolic` solves $M\frac{d^2}{dt^2}U + KU = F$

SUMMARY
Chapter 5

Simulation of PDEs in **two** space dimension:

- Classification of PDEs
- Elliptic PDEs - Steady-State Problems
  - Finite Difference Methods
  - Finite Element Methods
- Nonuniform Grids and Irregular Boundaries
- Parabolic PDEs
  - Method of Lines
- MATLAB PDE-toolbox

SUMMARY I
Simulation of Process Systems

Introduction to **Simulation Methods** for Process Systems

- Algebraic Equations
  - Linear - LU-decomposition
  - Nonlinear - Newton method
- Initial-Value Problems of Ordinary Differential Equations
  - Explicit - Runge-Kutta/Adams
  - Implicit - IRK/Gear
- Boundary-Value Problems of Ordinary Differential Equations
  - Shooting methods
  - Finite difference
  - Finite elements

SUMMARY II
Simulation of Process Systems

Introduction to **Simulation Methods** for Process Systems

- Parabolic Partial Differential Equations
  - Method of Lines
    - Finite difference
    - Finite elements
- Hyperbolic Partial Differential Equations
  - Method of Lines
  - (in general hard to solve, special methods)
- Elliptic Partial Differential Equations
  - Finite difference methods
  - Finite elements methods

SCOPE

Scope of the course:

Deterministic models *are the basis for all the methods discussed in the course with* **no** *exeption.*

Steady-state *and* **Unsteady-state responses** *are the ways to study the performances of the models.*

Extensions of the course must go beyond these statements

EXTENSION I
Events and Noise

Extensions into **Discrete** and **Stochastic** simulation

- State-driven events in continuous time.
  - indicator functions (`gstop`)
- Sampling systems (digital devises)
  - one-step methods
- Stochastic systems
  - random number generation
  - one-step methods (Euler)
- Discrete systems
  - (*another framework*)

EXTENSION II

Extensions in **Continuous** simulation

- Differential-Algebraic Equation systems
  - equilibrium problems
  - DAE and PDAE-solvers
- Integral Partial Differential Equations
  - IPDE and IPDAE - The state is continuous distributed.

Extensions into frequency domain and **frequency responses**

- Complex functions
  - Amplitude and argument functions
- Power spectrum
  - FFT

WHAT'S LEFT?

- Computer Exercise 5 - PDE-toolbox
- Söderlind-lecture?
- Hand-In discussion (informal)
- For you that would like to do *part III*
  - "theoretical" seminar
  - "practical" project

SIMULATION
part III

Suggestions on subjects are *detailed discussion on material in the course* or *extensions of the material*

- Nonlinear equation solvers
- DAE-solvers
- Collocation in MATLAB
- Adaptive griding in MOL
- Solvers for hyperbolic problems

SIMULATION OF PROCESS SYSTEMS

Lecture Problems 1: Methods for AEs and ODEs

Aim: Some theoretical exercises to do by paper and pen and to **hand-in 13/5.**

Problem 1: A separator is fed with a mixture of 60 % A and 40 % B. 90 % of all entering B is condensed and removed in stream 2 and all A together with the rest of B is removed in stream 3. The resulting algebraic equation system is as follows (equation order: separation demand, A-mole and B-mole balances)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} N_B^{[2]} \\ N_B^{[3]} \\ N_A^{[3]} \end{bmatrix} = \begin{bmatrix} 36 \\ 60 \\ 40 \end{bmatrix}$$

Calculate the flows, N, and do a LU-decomposition of the A-matrix. Note that you have to make a partial pivoting.

Problem 2: The enthalpy in a liquid mixture is related to the temperature as a polynomial

$$H(T) = a_0 + a_1 T + a_2 T^2 + a_3 T^3$$

Assume that we know the enthalpy and we want to find the temperature. Put up the Newton-Raphson iteration formula for this problem.

Problem 3: A continuous stirred tank, CST, with cooling and constant volume and flow has the follows ODE describing the temperature dynamics (assume constant C_p).

$$\frac{dT}{dt} = \frac{q}{V}(T_0 - T) + \frac{kA}{\rho V C_p}(T_c - T)$$

For simplicity assume $\frac{q}{V} = 0.15$ and $\frac{kA}{\rho V C_p} = 0.1$. What are the restrictions on the step-sizes for an explicit and on an implicit Euler for this ODE?

Problem 4: What is a stiff problem? When does it occur? What implications has it on the integration methods? What is a DAE system? Give a process example of a stiff or DAE problem!

Problem 5: A heat transfer problem is discussed in Example 3 in Davis, chap 2, on pages 72-75. Formulate it as a shooting problem. (do not solve it!)

Problem 6: Follow the solution of the heat transfer problem on pages 72-75 and put up the resulting linear equation system, $Ax = b$, for $N = 5$. How does A, x and b look like? Assume $H = 1$. How is the boundary value $\frac{d\theta}{dx}(1) = 0$ approximated?

Bernt Nilsson, Chemical Engineering II and Automatic Control

SIMULATION OF PROCESS SYSTEMS

Lecture Problems 2: Methods PDEs

Aim: Some theoretical exercises to do by paper and pen and to **hand-in 31/5**.

Problem 7: One way to solve a parabolic PDE by the use of *method of lines* requires an approximation of the space dimension. The resulting problem that needs to be solved is a set of ODEs. Make a centered-difference approximation in space and put up the ODE system on state-space form, $\dot{x} = Ax + b$, for the following PDE ($0 \leq x \leq 1$ and $t \geq 0$).

$$\begin{aligned} \frac{\partial c}{\partial t} &= D\frac{\partial^2 c}{\partial x^2} + 1 \\ c(0,t) &= 0 \\ c(1,t) &= 0 \\ c(x,0) &= 0 \end{aligned}$$

Problem 8: Use the *Galerkin method* to rewrite the following ODE-BVP

$$-y''(x) = 1, \qquad y(0) = 0, \qquad y(1) = 0$$

as an algebraic equation system. Use piecewise linear basis functions $\wp_2^1$ to approximated the solution. Solve the problem with the mesh-size 1/4 (3 internal breakpoints). Plot (by hand) the solution and its pp-functions.

Problem 9: Use the Galerkin method in *method of lines* to rewrite the parabolic PDE in problem 7 into a set of ODEs. Use piecewise linear basis functions $\wp_2^1$. Write the problem as $B\dot{x} = Ax + b$ with the mesh-size 1/4 (3 internal breakpoints).

Problem 10: Solve Problem 1 on page 222 (Davis, chap 5) which is finite difference approximation of an elliptic PDE.

Problem 11: Describe a problem that **you** have encountered in your research, teaching or education that can be solved by a method presented in the course. (A solution is not necessary, but desired)

Bernt Nilsson, Chemical Engineering II and Automatic Control

SIMULATION OF PROCESS SYSTEMS

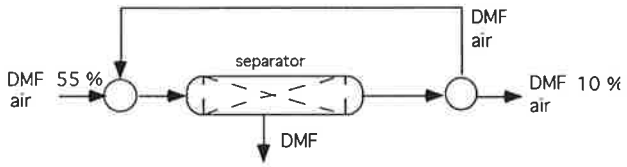
Computer Exercise 1:

Algebraic Equation Systems

Aim: The computer exercises in this course are composed of two parts, A and B. Part **A** introduce different methods, process application and the use of MATLAB. Part **B** is a home exercise to hand-in.

Part A: Algebraic Equations

Exercise 1: LU-decomposition
A purification system with recycle is used to recover the solvent DMF from a waste gas containing 55 % DMF in air. The product is to have only 10 % DMF. Calculate the recycle fraction assuming that the purification unit can remove two thirds of the DMF in the combined feed to the unit. (see the material balance problem 2.24 in Reklaitis.)
If you put up the species balances over the mixer, separator and splitter and



Figur 1: The purification system of DMF

the purification relation one get 7 linear equations. The outflow is assumed to be $N^5 = 1$ (the basis).
Solve the material balance problem seen below, $Ax = b$. Whats the relative condition number? Make a `lu`-decomposition of the A matrix. How does the permutation matrix P look like and what does it mean? Make a MATLAB function that plots the recycle fraction, $\frac{N^6}{N^1}$, as a function of the separation relation, ie change the value of `A(7,2)`.

$$\begin{bmatrix} 0.55 & -1 & 0 & 0 & 0 & 0 & 0.1 \\ 0.45 & 0 & -1 & 0 & 0 & 0 & 0.9 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -0.1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -0.9 \\ 0 & -\frac{2}{3} & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} N^1 \\ N^2_{DMF} \\ N^2_{air} \\ N^3 \\ N^4_{DMF} \\ N^4_{air} \\ N^6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.1 \\ 0.9 \\ 0 \end{bmatrix}$$

Exercise 2: Ill-condioned problems

Two fuels are mixed and burned with air to produce a flue gas analyzing 7% CO_2, 1% CO, 7% O_2 and the rest N_2. If the fuel compositions are 80% CH_4 and 20% N_2 and 60% CH_4, 20% C_2H_6 and 20% N_2, calculate the ratio of the feed rates of the two fuels. (see example 4.5 in Reklaitis.)

This can be solved by the use of element balances solving the equation $\alpha\Delta N = b$, where α is the atom matrix, Δ is the component change matrix and N is the unknown molar flows of the second fuel, air, dry flue gas and water. The molar flow of the first fuel is set to one (basis).

$$\alpha = \begin{bmatrix} 0 & 1 & 2 & 1 & 1 & 0 \\ 0 & 4 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 2 \\ 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \Delta = \begin{bmatrix} -0.2 & -0.79 & 0.85 & 0 \\ 0 & -0.21 & 0.07 & 0 \\ 0 & 0 & 0.07 & 0 \\ -0.6 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ -0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$N = \begin{bmatrix} N^2 & N^3 & N^4 & N^5 \end{bmatrix}^T \qquad b = \begin{bmatrix} 0.4 & 0 & 0.8 & 3.2 \end{bmatrix}^T$$

Solve the linear problem. Study the stability of the solution by making small pertubuations of the air composition, nitrogen, $\Delta(1,2)$, and oxygen, $\Delta(2,2)$. What is the relative condition number, `rcond`, of the A matrix.

Exercise 3: Gauss-Seidel iteration method

Gauss-Seidel is suited for diagonal dominant matricies. Create an linear equation system $Ax = b$ with the following A and b

```
A1 = [zeros(1,n);eye(n-1) zeros(n-1,1)];
A = -2*eye(n) + A1 + A1';
b =-[1; zeros(n-1,1)];
```

Solve it by the use of the Gauss-Seidel method `gseid`. (Implemented by Mathews.) How does this system converge?

(extra exercise: Solve the same problem as in *Exercise 1* with the Gauss-Seidel iteration method, `gseid`. Notice that this method requires non-zero diagonal elements. (suggestion: use the following column order $1, 7, 2, 3, 5, 6, 4$). Does the iteration converge? Plot the iteration results (of one variable).)

Exercise 4: Nonlinear equation

Phase equilibrium between liquid and vapour can be expressed as $y = K(T)x$. The equilibrium polynomial is of third order, $K = a+bT+cT^2+dT^3$. Calculate the bubble point in a liquid, with 20% n-hexan, (x_1), and 80% n-heptan, (x_2), using `fzero`. Unknowns are the vapour compositions, y_1 och

y_2, and the temperature. The three equations needed to solve the problem are the two equilibrium relations and that the vapour mole fractions must sum to one.

$$y_1 = K_1 x_1$$
$$y_2 = K_2 x_2$$
$$y_1 + y_2 = 1$$

where the equilibrium relations are

$$K_1 = 1.584 + 5.247\ 10^{-2}T - 4.067\ 10^{-4}T^2 + 7.259\ 10^{-7}T^3$$
$$K_2 = 1.415 + 3.955\ 10^{-2}T - 2.976\ 10^{-4}T^2 + 5.056\ 10^{-7}T^3.$$

(This is the same exercise as 11.2 in the MATLAB cource. For you that have done this exercise solve it by using `roots` *instead.)*

Exercise 5: Nonlinear equation system
In a continuous stirred tank reactor, CSTR, a first order exothermic reaction occurs. A mass balance over reactant A and an energy balance give the following algebraic equations at steady-state.

$$q(c_{A_0} - c_A) - kc_A V = 0$$
$$q\rho C_p(T_0 - T) - kc_A V\Delta H = 0$$

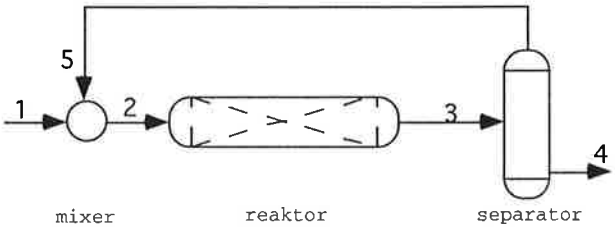
Assume Arrhenius temperature dependency, $k = k_0 e^{-\frac{E_a}{R(T+273)}}$.
Find the steady-state conditions for c_A and T, by the use of `fsolve`. Use the following data: $c_{A_0} = 3$, $\rho = 1000$, $V = 18e-3$, $C_p = 4.19$, $T_0 = 25$, $q = 60e{-}6$, $\Delta H = -2.09e5$, $K_0 = 4.48e6$, $E_a = 62800$ and finally $R = 8.314$.

Part B: Process Example

In a process for the catalytic hydration of ethylen to etyl alcohol, only a fraction of the ethylen is converted.

$$C_2H_4 + H_2O \longrightarrow C_2H_5OH.$$

The majority of the product is condensed and removed after each pass through the converter and the unconverted gases are recycled. (modification of example 3.11 in Reklaitis.)
Assume the following data



Figur 2: Flowsheet for etylen hydration process

- The recyle gases will contain 6.5 % water vapor.
- The convertion of ethylen per pass through the converter is 4.5 %.
- The molar ratio of water to ethylen in the feed to converter, after mixing the recycle gas with fresh feed, is 0.55.
- 90 % of all alcohol that enters the separator is condensed.
- Assume the product flow of alcohol to be 100

Application problems: Put up the species balances of ethylen, water and alcohol over the mixer, converter and separator together with the information relations above. Calculate all streams in the process.
Plot the recycle fraction as a function of the conversion.

Numerical problems: Solve this problem using `lu`-decomposition. The problem is quite sparse. Define the matricies as `sparse` and solve it using `lu` again. What is the sparseness (density) of the A matrix, `nnz`. Count the number of floating point operations, `flops`, needed for the solution using full and sparse matrices respectivly.

Bernt Nilsson, april 10
Chemical Engineering II, LTH

SIMULATION OF PROCESS SYSTEMS

Computer Exercise 2:

Dynamic Simulation of ODEs

Aim: The computer exercises in this course are composed of two parts, A and B. Part **A** introduce different methods, process application and the use of MATLAB. Part **B** is a home exercise to hand-in.

Part A: Initial value problems of ODEs

Exercise 1: Dynamic Simulation
A tank with constant volume has one inlet and one outlet and are cooled by a coil with constant temperature. The temperature of the tank, T, can be modelled with a first order ODE.

$$\frac{dT}{dt} = \frac{kA}{V\rho C_p}(T_c - T) + \frac{F}{V\rho C_p}(T_0 - T)$$

where

- initial value $T = 50$
- $T_c = 25$ is the cooling temperature.
- $T_0 = 50$ is the inlet temperature.
- assume $\frac{kA}{V\rho C_p} = 1$

a) Simulate the tank with no inlet, $\frac{F}{V\rho C_p} = 0$. Study changes of the initial value of T.

```
ode23('tank21',[0 10],50);
```

b) Simulate the tank with an inlet, $\frac{F}{V\rho C_p} = 1$.

c) Assume that the inlet is opened after 10 time units, $\frac{F}{V\rho C_p} = 1$. Study the dynamic behaviour.

```
function [dtemp]=tank21(t,temp);
% Tank model in exercise 2.1
Tc = 25; T0 = 50; k1 = 1;
if t<10 k2=0; else k2=1; end;
dtemp = k1*(Tc - temp) + k2*(T0 - temp);
```

Exercise 2: Step-sizes and Stability
The example 1 in Davis on page 5 to 7 results in the following ODE. $\dot{y} = -21.6y$. Verify the numbers in table 1.1 and plot the result using an Euler method with fix step-size, `eulers` (from the Mathews library). What is the largest step-size for stability and for non-oscillations?
Do the same simulation with a Runge-Kutta method with fix step-size, `rk4` (from the Mathews library). Use 10 steps.
extra: Make an implicit Euler approximation of the equation and simulate it for some different step-sizes. Make a M-file that do it (no predefined methods).

Exercise 3: Stiffness
Simulate the problem discussed on page 29 to 31 in Davis. Compare one explicit and one implicit method for the problem. Choose for example `ode23` and `ode23s`.

Exercise 4: Methods in ODE Suite
Different methods are good at different things. One can measure the number of floating points operations, `flops`, used by a solver in MATLAB. This is a measurement of how efficient the solver is for a particular problem.
Example:

```
>> flops(0);                      %reset counter
>> [t,x]=ode23('ex2',[0 1],1); %ode solver
>> nf23 = flops                   %flops measurement
```

Do a comparision of all available methods in MATLAB by running the examples in exercise 2 and 3 and count the floating point operations, `flops`.

ODE methods	ex.2 non-stiff	ex.3 stiff
ode23		
ode45		
ode113		
ode23s		
ode15s		
odesl		

Exercise 5:
In a continuous stirred tank reactor, CSTR, a first order exothermic reaction are taking place, see Exercise 1.5 (previous Computer Exercise). A mass balance over reactant A and an energy balance give the following nonlinear ordinary differential equations.

$$q(c_{A_0} - c_A) - kc_A V = \frac{d(Vc_A}{dt}$$
$$q\rho C_p(T_0 - T) - kc_A V\Delta H = V\rho C_p \frac{dT}{dt}$$

Assume Arrhenius temperature dependency, $k = k_0 e^{-\frac{E_a}{R(T+273)}}$.
The steady-state conditions for c_A and T was found by the use of `fsolve` in exercise 1.5. Use the following data: $c_{A_0} = 3$, $\rho = 1000$, $V = 18e-3$, $C_p = 4.19$, $T_0 = 25$, $q = 60e-6$, $\Delta H = -2.09e5$, $K_0 = 4.48e6$, $E_a = 62800$ and finally $R = 8.314$.
Make dynamic simulations of the tank reactor and simulate the behaviour round the different steady-states. Make a phase-plan flot (c_A vs. T).

Part B: Heated Vessel Example

Do the Problem 5 in Davis on page 47 and 48.

Numerical problems: Solve this problem using different ODE-solvers, one non-stiff, one stiff and one stiff with defined jacobian function. What is the stiffness ratio?

Application problems: The parameters in the example are not so realistic. (the tank and the termocouple have the same time constants). Assume the the termocouple are 10 times faster. Assume also that the flow/mass ratio is between 10 and 50.

Extra problems: Add dynamic mass balance (rewrite the old ones) and study the behavior of the temperature control at different tank levels.
(extra ++: Add integral action in the temperature controller).

Bernt Nilsson, april 18
Chemical Engineering II. LTH

SIMULATION OF PROCESS SYSTEMS

Computer Exercise 3: Steady-State Simulation of ODEs

Aim: Some computer exercises (Part **A**) to introduce different methods, process applications and how to do it in MATLAB. Part **B** is a home exercise to hand-in.

Part A: BVP for ODEs

Exercise 1: Linear shooting method
Dispersion model of an isothermal tubular reactor at steady-state with first-order kinetics is a linear second order ODE in one space dimension, $0 \geq x \geq 1$,

$$D\frac{d^2c}{dx^2} - v\frac{dc}{dx} - kc = 0$$

where D is the dispersion, v is the velocity, k is the reaction constant and c is the concentration. Assume $D = v = k = 1$ (for simplicity) and the feed concentration $c(0) = 1$. The general linear boundary conditions can be written as:

$$a_0c(0) - a_1c'(0) = \alpha$$
$$b_0c(1) + b_1c'(1) = \beta$$

Solve the boundary-value problem (BVP) for the ODE using the linear shooting method. Rewrite the problem as two IVPs. Here follows a suggestion of a state-space representation in a M-file.

```
function [dx]=tubivp(t,x)
% tub-BVP rewritten as two IVPs
k=1; D=1; v=1;
A=[0 1; k/D v/D];
dx=[A zeros(2,2); zeros(2,2) A]*x;
```

a) Assume the additional boundary condition $c(1) = 0.1$ (separated Dirichlet BCs). This means $a_0 = 1$, $a_1 = 0$, $\alpha = 1$, $b_0 = 1$, $b_1 = 0$, $\beta = 0.1$ and the IV-relation becomes $a_1C_0 - a_0C_1 = 1$ which results in $C_1 = -1$ and C_0 can be anything. Simulate the two IVP with the following inital-values

$$\begin{aligned} x_1(0) &= -\alpha C_1 = -1(-1) = 1 \\ x_2(0) &= -\alpha C_0 = 0 \\ x_3(0) &= -a_1 = 0 \\ x_4(0) &= -a_0 = 1 \end{aligned}$$

It is now straightforward to solve the problem. (note that $x_2(0)$ can be chosen to anything.)

```
[t,x]=ode23('tubivp',[0 1],[1 0 0 1]);
n= max(size(x));
s=(beta-(b0*x(n,1) + b1*x(n,1)))/(b0*x(n,3) + b1*x(n,4));
y=x(:,1) + s*x(:,3);
```

b) Assume the additional Neumann boundary condition $c'(1) = 0$. Solve the problem using the technique above.

extra problem: The problem becomes nonlinear if the reaction is of second-order ... $-kc^2 = 0$. Make a nonlinear shooting, see page 57 in Davis, to solve the same problem as above!

Exercise 2: Finite difference method
An isothermal plug-flow tubular reactor, PTR, with first-order reaction is modelled by a linear first order ODE, $v\frac{dc}{dx} = -kc$. Assume a feed concentration $c(0) = 1$.
Make a backward differentiation approximation of the ODE and solve the problem as a linear equation system. *(Note that this is the simplest finite difference approximation and it is sometimes called the static* **tank-series model**.*)*

```
function [x,u,A,b]=tubfdm(n)
% PTR solved by backward finite difference
k=1; v=1; u0=1;
h = (1-0)/(n+1);                              %mesh-size
x=0:h:1;                                      %gridpoints
Ah = [zeros(1,n); eye(n-1) zeros(n-1,1)];%help matrix
A0 = eye(n);                                  %I-matrix
A1 = A0 - Ah;                                 %1st der. approx
A  = A1*v/h + A0*k;                           %ODE to AE
b = [v/h*u0; zeros(n-1,1)];                   %BC
u1 = A\b;                                     %solve gridpoints
u = [u0; u1];                                 %boundary and grid
```

Solve the problem with $3, 10$ and 20 interiour gridpoints and plot the result in the same plot. Plot u for different k, D, v!

Exercise 3: Finite differences becomes sparse
For a large number of gridpoints the problem becomes sparse or very sparse. Solve the problem with 100 and 400 points and count the number of `flops` and the sparseness density for full and sparse matrix calculations. How many gridpoints are required in order to make sparse matrix technique superior?

extra problem: Solve this problem using the Gauss-Seidel method, `gseid`. Can it compete with LU-decomposition?

Exercise 4: Centered-difference approximation
Let us go back to the dispersion model in exercise 1,

$$D\frac{d^2c}{dx^2} - v\frac{dc}{dx} - kc = 0$$

Use the same parameters. Solve the problem using centered-difference approximation for the following BCs:
a) $c(0) = 1$ and $c(1) = 0.1$. See `tubCD` below.
b) $c(0) = 1$ and $c'(1) = 0$. Modify `tubCD` for Neumann BCs.

```
function [x,u,A,b]=tubCD(n)
% Tube model solved by centered-difference
% Dirichlet BCs
k=1; D=1; v=1; u0=1; uL=0.1;
h = (1-0)/(n+1);
x=0:h:1;
Ah = [zeros(1,n); eye(n-1) zeros(n-1,1)];
A0 = eye(n);
A1 = Ah' - Ah;
A2 = -2*A0 + Ah + Ah';
A  = A2*D/h^2 - A1*v/(2*h) - A0*k;
b = [(-D/h^2-v/(2*h))*u0; zeros(n-2,1); (-D/h^2+v/(2*h))*uL];
u1 = A\b;
u = [u0; u1; uL];
```

Part B: Dispersed Plug Flow Tubular Reactor

Assume a tubular reactor modelled by a dispersion model as in the last exercise with two reactions $A \rightarrow B \rightarrow C$ which are described as first-order reactions. This results in two coupled ODEs.

$$\begin{aligned} D_A \frac{d^2 c_A}{dx^2} - v\frac{dc_A}{dx} - k_1 c_A &= 0 \\ D_B \frac{d^2 c_B}{dx^2} - v\frac{dc_B}{dx} - k_2 c_B &= -k_1 c_A \end{aligned}$$

The BCs at the inlet are $c_A(0) = 1$ and $c_B(0) = 0$. Solve the problem using centered-difference approximations of second-order accuracy (or own choice). Select a BC at the outlet , *i:* Dirichlet $c_A(1) = 0, c_B(1) = 0.2$ or *ii:* Neumann $c'_A(1) = c'_B(1) = 0$.

Application problems: How does the tube profiles look like for different k-values (begin with $k_1 = 20, h_2 = 1$)? Put up the problem as $Au = b$. How does A look like for the two different u-vectors. Show it for $n = 5$. Visualize the A-matrix with `spy`.
i) $u_{band} = [u_{c_A} u_{c_B}]^T$ where $u_{c_A} = [u_{c_A,1} \ldots u_{c_A,N}]^T$.
ii) $u_{block} = [u_1 \ldots u_N]^T$ where $u_j = [u_{c_A,j} u_{c_B,j}]^T$.
Make the first reaction reversible, $A \rightleftharpoons B \rightarrow C$ and solve the problem. How does A look like using u_{band}

Numerical problems: Exercise 3 focus on the need for sparse matrix technique to efficiently solve problems with large numbers of gridpoint. Both of the suggested functions for finite differences in Exercises 2 and 4 creates 4 to 5 full matrices before solving. Make a function that never creates a full matrix and that uses sparse storage together with sparse solving. See the command `sparse, speye` etc. Visualize a large A-matrix with `spy`.

SIMULATION OF PROCESS SYSTEMS

Computer Exercise 4: Dynamic Simulation of PDEs

Aim: Some computer exercises (Part **A**) to introduce different methods, process applications and how to do it in MATLAB. Part **B** is a home exercise to hand-in.

Part A: Method of Lines

Exercise 1: Euler and Centered-Difference Approximations
Unsteady-state diffusion is described by a parabolic PDE ($0 \leq x \leq a$ and $0 \leq t \leq b$)

$$\frac{dT}{dt} = c^2 \frac{d^2T}{dx^2}$$

A centered-difference approximation in space and forward Euler in time is implemented in the M-file `forwdif` (from Mathews, chap 10). To simulate the problem three M-files must be defined, one for the initial value, `f`, and two for the boundary conditions, `g1` and `g2`.

```
u=forwdif('f','g1','g2',a,b,c,n,m);

function z=f(x)
% Initial values for forwdif
z = 4*x - 4*x^2;

function z=g1(x)
% Boundary condition 1 for forwdif
z = 0;

function z=g2(x)
% Boundary condition 2 for forwdif
z = 0;
```

Simulate the diffusion problem and make a `mesh` plot of u using two parameter sets. What are the Courant condition in the two cases, $\lambda = 1 - 2c^2 \frac{\Delta t}{h^2}$?
a) Gridpoints in space are 6 ($n = 6$) and in time 11 ($m = 11$)

```
u=forwdif('f','g1','g2',1,0.2,1,6,11);
```

b) Simulate to 1/3.

```
u=forwdif('f','g1','g2',1,0.3333,1,6,11);
```

Exercise 2: Diffusion and Nonlinear Kinetics
Chemical kinetic models are often benchmark problems in numerical analysis and you find a number of them in ODE suite. One is the *Brusselator model* described in `brussex` with its Jacobian defined in `brussjac`. The model is a parabolic PDE with nonlinear terms, diffusion and nonlinear kinetics, and with Dirichlet boundary conditions

$$\frac{dc_A}{dt} = D\frac{d^2c_A}{dx^2} - 4c_A + c_B c_A^2 + 1$$
$$\frac{dc_B}{dt} = D\frac{d^2c_B}{dx^2} + 3c_A - c_B c_A^2$$
$$c_A(0,t) = c_A(1,t) = 1, \quad c_A(x,0) = 1 + \sin((\frac{2\pi}{N+1}x)$$
$$c_B(0,t) = c_B(1,t) = 3, \quad c_B(x,0) = 3$$

The number of gridpoints in the space discretization is defined in the global variable N. To get the right starting values for simulation, run brussex with empty arguments. This command returns tspan, IC and options for ODE solvers in ODE suite. Simulate the Brusselator and plot the time responses.

```
global N
N = 10;
[out1,out2,out3]=brussex([],[]);
[t,x]=ode23('brussex',out1,out2,out3)
```

Use the solution above to make surface or mesh plots of $c_A(x,t)$ and $c_B(x,t)$. The result in `x` are $c_A(x,t)$ and $c_B(x,t)$ for the 10 gridpoints.

```
m1=[];
for i=1:2:2*N, m1=[m1 x(:,i)]; end;
m2=[];
for i=2:2:2*N, m2=[m2 x(:,i)]; end;
[tt,xx]=meshgrid(t,1:N);
mesh(tt,xx,m1')
```

Exercise 3: Stiff and Sparse problems
For the problem in exercise 2 the explicit `ode23` take the fewest floating point operations. That is not the case if N and `tspan` is increased and the implicit `ode15s` becomes the most efficient. An implicit ODE solver needs a Jacobian to solve an implicit equations system one or many times during the simulation. This Jacobian can be defined analytically or derived numerically. For the Brusselator the analytical Jacobian is defined in `brussjac`. If the analytical Jacobian is not defined the implicit solvers uses a numerical Jacobian calculated by `numjac`. If the problem is sparse one can help `numjac` by defining the sparseness. The two ways are examplified in the MATLAB code on the next side.

```
options=odeset('analyticJ','brussjac');
[t,x]=ode15s('brussex,out1,out2,options);

B = ones(2*N,5);
B(2:2:2*N,2) = zeros(N,1);
B(1:2:2*N-1,4) = zeros(N,1);
S = spdiags(B,-2:2,2*N,2*N);
options=odeset('sparseJ',S);
[t,x]=ode15s('brussex,out1,out2,options);
```

Exercise 4: Centered-difference approximation
Let us go back to the dispersion model in exercise 3.4 and modify it to capture dynamic changes of the concentration.

$$D\frac{d^2c}{dx^2} - v\frac{dc}{dx} - kc = \frac{dc}{dt}$$

Use $D = v = k = 1$. Solve the problem using centered-difference approximation in space and simulate the resulting ODE system with a solver in ODE suite. Assume Dirichlet BC at the inlet, $c(0,t) = 1$, Neumann at the outlet, $c'(1,t) = 0$ and the initial condition to be $c(x,0) = 1$.
The tube model below can now be simulated with the following command `ode23('tubMOL',[0 1],ones(1,11))`. Visual the result in a `plot(t,u)` and in a `mesh(u)`.
Notice the similarities between `tubMOL` below and `tubCD` in Exercise 3.4.

```
function du=tubMOL(t,u)
% Dispersion tube model with centered-difference
% in space and Dirichlet at the inlet and Neumann
% at the outlet
nn=10;                                      % internal gridpoints
n = nn +1;                                  % nn plus outlet boundary
k=1; D=1; v=1; u0=1;                        % parameters
h = (1-0)/(nn+1);                           % mesh-size
x=0:h:1;                                    % space dimension
Ah = [zeros(1,n); eye(n-1) zeros(n-1,1)]; % help matrix
A0 = eye(n);                                % 0-order terms
A1 = Ah' - Ah;                              % 1-order terms
A2 = -2*A0 + Ah + Ah';                      % 2-order terms
A2(n,n-1) = 2;                              % Neumann BC
A  = A2*D/h^2 - A1*v/(2*h) - A0*k;          % space discretization
b = [(-D/h^2-v/(2*h))*u0; zeros(n-1,1)];  % Dirichlet BC
du = A*u - b;
```

(Extra) Exercise 5: Dynamic Dispersion Model
(This extra problem is discussed in more detail in Ramirez, Computational Methods for Process Simulation, pp 365-376. The exothermic reaction can cause very large temperature changes, *hot spot.*)
A dynamic tubular reactor is modelled by one species balance and one energy balance, both with dispersion. A simple exothermic reaction is assume to occur with temperature dependent reaction coefficient $k = k_0 e^{\frac{-E_a}{RT}}$.

$$\frac{dc_A}{dt} = D\frac{d^2c_A}{dx^2} - v\frac{dc_A}{dx} - kc_A$$
$$\rho C_p \frac{dT}{dt} = \lambda \frac{d^2T}{dx^2} - \rho C_p v \frac{dT}{dx} + (-\Delta H)kc_A$$

The model in Ramirez assumes a Neumann BC at the outlet, in other words no gradients at the outlet boundary. At the inlet a Robin BC is used to describe diffusion and conduction at the inlet boundary, example $-\lambda\frac{\partial T}{\partial z}(0) = \rho C_p v(T_{in} - T(0))$. The model is nondimensionalized and discretized into the MATLAB code, `dyntub`, found on the next page. See Ramirez for futher details.

```
[t,x]=ode15s('dybtub',[0 1],ones(1,22));
```

Part B: Hand-In Problems

Select one of the following problems. Based the solution on the material in chapter 4 and use MATLAB with ODE suite.

a) Simulate the spherical catalyst pellet example discussed and presented on pages 147-154 in Davis and in Problem 6 on page 169. Make a similar table as Table 4.4 based on `flops` in MATLAB.

b) Simulate diffusion in two immiscible solvent system. Solve Problem 3 and 4 on pages 167 and 168.
extra problem: assume another relation between C_A^1 and C_A^2 then the one used in the book.

c) Diffusion and adsorption in an idealized pore. Solve Problem 5 on pages 168 and 169. Discuss discretization, stiffness, sparseness and the use of ODE solvers in the problem.

Bernt Nilsson, Chemical Engineering II and Automatic Control

```
function[du]=dyntub(t,u)
% Dynamic dispersion model of a tubular reactor with species and energy
%   balances approximated with centered-FD and MOL
%   BC: Robin at the inlet and and Neumann at the outlet
% (code for extra exercise 5) from: Ramirez, Computational Methods for PS

% reactor parameters
a=1;   D=1;
r1=30; r2=30; cp=1;
C0=0.5; T0=510;
L=100; rho=60;
k=1.2e4;  q=17.6; dH=4e4;
% a second set
%k=1.2e6; q=23;   dH=4.32e4;

% equation parameters
B1 = (dH*k*C0*L^2)/(rho*cp*T0*D);
B2 = k*L^2/D;
Tin = 1;
% step responce after 0.1
if t<0.1 Cin=1; else Cin=1.1; end;

% grid
ni=20; n=ni+2;                          % Neumann/Robin grid
h=(1 - 0)/(ni+1);                       % mesh-size
x=0:h:1;                                % grid points
T=u(1:n);                               % temp.var.
C=u(n+1:2*n);                           % conc.var.

% FD-matrices
Ah=[zeros(1,n);eye(n-1) zeros(n-1,1)];  % help matrix
A0 = eye(n);                            % 0-order
A1 = -Ah + Ah';                         % 1-order
A2 = -2*A0 + Ah + Ah';                  % 2-order

% Neumann at x=L
bcn=zeros(n,1); bcn(n)=1;
Tbcn=(a/(D*h^2) - r2/(2*h))*T(n-1).*bcn;
Cbcn=(1/(h^2)   - r2/(2*h))*C(n-1).*bcn;

% Robin at x=0
bc0=zeros(n,1); bc0(1)=1;
Tbc0=(a/(D*h^2)+r2/(2*h))*(T(2)-2*h*r1*(T(1)-Tin)).*bc0;
Cbc0=(1/(h^2)  +r2/(2*h))*(C(2)-2*h*r2*(C(1)-Cin)).*bc0;

% ODEs
olin = C.*exp(-q./T);                    % temp dep. r
dT = (a/(D*h^2).*A2 - r2/(2*h)*A1)*T + B1*olin + Tbc0 + Tbcn;
dC = (    1/h^2.*A2 - r2/(2*h)*A1)*C - B2*olin + Cbc0 + Cbcn;
du=[dT;dC];
```

SIMULATION OF PROCESS SYSTEMS

Computer Exercise 5: Simulation of PDEs

Aim: Some computer exercises (Part **A**) to introduce different methods, process applications and how to do it in MATLAB. Part **B** is a home exercise to hand-in.

Part A: PDE toolbox

PDE-toolbox in MATLAB is based on **Finite Element Method** using *Galerkin*, *linear basis functions* and *triangular elements.*
The toolbox can be used through the ordinary command-line in MATLAB or in the interactive graphical interface

```
>> pdetool
```

PDEtool is a window which consists of a set of menus, one toolbar and one drawing area. The toolbar is just push buttons for some menu choises and it is optional.

Exercise 1: Steady-State Heat Transfer or Elliptic PDE
Solve Example 1 in Davis, page 181-183, using PDE-toolbox. Consider a square plate, ($0 \leq x \leq 1$ and $0 \leq y \leq 1$), with the heat conduction equation (Laplace's equation)

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

with the boundary conditions

$$T(0,y) = T_1 \qquad T(1,y) = T_2$$
$$\frac{\partial T}{\partial y}(x,0) = 0 \qquad \frac{\partial T}{\partial y}(x,1) = k[T(x,1) - T_2]$$

Choose for simplicity $T_1 = 2$, $T_2 = 1$ and $k = -5$.

1. Select the **rectangle** button on the toolbar. Draw a square from $(0,1)$ to $(1,0)$. You get a gray rectangle with a name (R1).
2. Select the **boundary** button ($\partial\Omega$). The segments of the boundary becomes red arrows. By dubble clicking on the boundaries a dialog window occurs.
   (a) Dubble click on the $(x = 0)$ boundary. Define a Dirichlet boundary condition with the value $r = 2$.

(b) Dubble click on the ($x = 1$) boundary and define a Dirichlet with $r = 1$.

(c) Dubble click on the ($y = 0$) boundary. Define a homogenous Neumann, $g = q = 0$. Note that the boundary arrow change colour to blue

(d) Dubble click on the ($y = 1$) boundary. Define a Robin BC (generalized Neumann), $q = g = 5$.

3. Choose an **elliptic PDE** equation in the dialog window under **PDE**. Coefficients are $c = 1$ and $a = f = 0$.

4. **Intitialize the mesh** by clicking on the triangle button, $\triangle$. It takes some seconds before the blue triangular mesh shows up.

5. **Solve** the problem by clicking on $=$.

6. **Visualize** the result by clicking on the "mesh-plot" icon.

   (a) `Color` indicate temperature by the use of colors

   (b) `Contour` results in a contour plot

   (c) `Height` results in a 3D plot. You can rotate the 3D plot by grabbing the plot with the cursor.

   (d) there is many ways to plot the result.

Exercise 2: Elliptic PDE and Boundary Conditions
Play around with the coefficients in the PDE and the BC definitions.

a) Assume that heat is constantly generated inside the plate. This means that the right-hand side of the PDE gets a constant value, $f = 10$. Solve and plot the result.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 10$$

b) Assume that heat is removed from the plate in the z-direction (vertical from the plate), $Q_z = k_1(T - T_0)$. Select $k_1 = 5$ and $T_0 = 0$ which means $a = 5$ and $f = 0$. Solve and plot the result.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + 2T = 0$$

c) Assume that the heat conduction coefficient is space dependent, $c = 0.1 + 0.9x$. Solve and plot the result.

d) Go back to the original PDE, $a = f = 0$ and $c = 1$. Change $k = 2$ in the Robin BC at $y = 1$, $(g = q = -2)$ Solve and plot the result.

e) Change the Dirichlet BC at $x = 0$ to $r = 1 + 0.3\sin(10y)$. Solve and plot the result.

f) Go back to the original problem once again. Add a nonlinear term

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} - 1.5T^2 = 0$$

and in **PDE** window add $a = -1.5u$. Go to the `Solve` menu and select `Parameters...`. Click on `Use nonlinear solver` and define an initial solution, starting value for the Newton solver (damped Gauss-Newton). Solve and plot the result.

Exercise 3: Unsteady-state Heat Transfer or Parabolic PDE
A dynamic version of the square plate, $(0 \leq x \leq 1$ and $0 \leq y \leq 1)$, with the unsteady-state heat conduction, a parabolic PDE or heat equation, is seen below

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}$$

with the boundary and initial conditions

$$\begin{aligned} &T(t,0,y) = T_1 \qquad && T(t,1,y) = T_2 \\ &\frac{\partial T}{\partial y}(t,x,0) = 0 \qquad && \frac{\partial T}{\partial y}(t,x,1) = k[T(t,x,1) - T_2] \\ &T(0,x,y) = T_2 \end{aligned}$$

Choose for simplicity $T_1 = 2$, $T_2 = 1$ and $k = -5$.

1. Define your boundaries as in Exercise 1.
2. Select **PDE** and parabolic PDE with $c = d = 1$.
3. **Initialize the mesh.**
4. Select `Solve` menu and `Parameters...` Define the simulation time `[0:0.02:0.2]` and the inital value `1` in $u(t0)$.
5. Solve, =.
6. Plot the result for different times, like 0.02, 0.1, 0.2. (You can also do animation of the solution)

Exercise 4: Catalyst pellet or Nonlinear Elliptic PDE System
Rewrite the spherical catalyst pellet model in Problem 4.6 (chap 4, page 169) to a steady-state elliptic PDE.

$$-\text{div}[(x^2+y^2)\text{gradc}] = (x^2+y^2)\Phi^2 e^{\gamma(1-\frac{1}{T})}c$$
$$-\text{div}[(x^2+y^2)\text{gradT}] = (x^2+y^2)\beta\Phi^2 e^{\gamma(1-\frac{1}{T})}c$$

Solve the problem with the nonlinear solver for elliptic PDEs.

1. Select `Generic system` on the toolbar (menu choice after the zoom button).
2. Define a circle with the radius 1.
3. Define Dirichlet BC to be equal to 1.
4. Select **PDE** and parabolic PDE
   (a) `c = x.^2+y.^2` for both PDEs (c_{11} and c_{22})
   (b) `f1 = -(x.^2+y.^2).*exp(18*(1-1./u(2))).*u(1))`
   (c) `f2 = 0.04*(x.^2+y.^2).*exp(18*(1-1./u(2))).*u(1)`
5. **Initialize the mesh**.
6. Select `Solve` menu and `Parameters...` and select nonlinear solver with initial value 1
7. Solve, =.
8. Plot the result for u and v.

Part B: Hand-In Problem

A Newtonian fluid flow example is discussed on page 215-218. Solve the problem in a number of different ways using the PDE-toolbox.

- Rewrite the problem as a steady-state elliptic PDE.
  - Solve it for the duct geometry
  - Solve only one corner (and use symmetry to derive BCs).
- Simulate the unsteady-state parabolic PDE for the case discussed in Davis.
- Animate the transient.
- Animate the solution for a sinusoidal pressure change over the duct.

Bernt Nilsson, Chemical Engineering II and Automatic Control

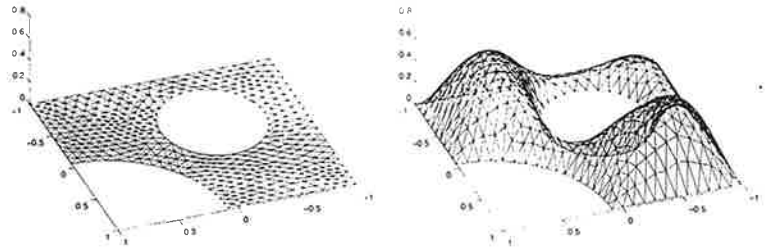
Basics of The Finite Element Method

The solutions of simple PDEs on complicated geometries can rarely be expressed in terms of elementary functions. You are confronted with two problems: First you need to describe a complicated geometry and generate a mesh on it. Then you need to discretize your PDE on the mesh and build an equation for the discrete approximation of the solution. The `pdetool` graphical user interface provides you with easy-to-use graphical tools to describe complicated domains and generate triangular meshes. It also discretizes PDEs, finds discrete solutions and plots results. You can access the mesh structures and the discretization functions directly from the command line (or M-file) and incorporate them into specialized applications.

Below is an overview of the *Finite Element Method* (*FEM*). The purpose of this presentation is to get you acquainted with the elementary FEM notions. Here you find the precise equations that are solved and the nature of the discrete solution. Different extensions of the basic equation implemented in the PDE Toolbox are presented. A more detailed description can be found in *The Finite Element Method* chapter.

You start by approximating the computational domain Ω with a union of simple geometric objects, in this case triangles. The triangles form a mesh and each vertex is called a node. You are in the situation of an architect designing a dome. He has to strike a balance between the ideal rounded forms of the original sketch and the limitations of his simple building-blocks, triangles or quadrilaterals. If the result does not look close enough to a perfect dome, the architect can always improve his work using smaller blocks.

Next you say that your solution should be simple on each triangle. Polynomials are a good choice: they are easy to evaluate and have good approximation properties on small domains. You can ask that the solutions in neighboring triangles connect to each other continuously across the edges. You can still decide how complicated the polynomials can be. Just like an architect, you want them as simple as possible. Constants are the simplest choice but you cannot match values on neighboring triangles. Linear functions come next. This is like using flat tiles to build a waterproof dome, which is perfectly possible.



A triangular mesh (left) and a continuous piecewise linear function on that mesh (right).

Now you use the the basic elliptic equation:

$$-\nabla \cdot (c\nabla u) + au = f \text{ in } \Omega.$$

If u_h is the piecewise linear approximation to u, it is not clear what the second derivative term means. Inside each triangle, ∇u_h is a constant (because u_h is flat) and thus the second-order term vanishes. At the edges of the triangles, $c\nabla u_h$ is in general discontinuous and a further derivative makes no sense.

What you are looking for is the best approximation of u in the class of continuous piecewise polynomials. Therefore you test the equation for u_h against all possible functions v of that class. Testing means formally to multiply the residual against any function and then integrate, i.e., determine u_h such that

$$\int_\Omega (-\nabla \cdot (c\nabla u_h) + au_h - f)\, v\, dx = 0,$$

for all possible v. The functions v are usually called *test functions*.

Partial integration (Green's formula) yields that u_h should satisfy

$$\int_\Omega (c\nabla u_h)\nabla v + au_h v\, dx - \int_{\partial\Omega} \vec{n} \cdot (c\nabla u_h) v\, ds = \int_\Omega fv\, dx \qquad \forall v,$$

where $\partial\Omega$ is the boundary of Ω and $\vec{n}$ is the outward pointing normal on $\partial\Omega$. Note that the integrals of this formulation are well-defined even if u_h and v are piecewise linear functions.

Boundary conditions are included in the following way. If u_h is known at some boundary points (Dirichlet boundary conditions), we restrict the test functions

to $v = 0$ at those points, and require u_h to attain the desired value at that point. At all the other points we ask for Neumann boundary conditions, i.e., $(c\nabla u_h)\cdot \vec{n} + qu_h = g$. The FEM formulation reads: Find u_h such that

$$\int_\Omega (c\nabla u_h)\nabla v + au_h v \, dx + \int_{\partial\Omega_1} qu_h v \, ds = \int_\Omega fv \, dx + \int_{\partial\Omega_1} gv \, ds \qquad \forall v,$$

where $\partial\Omega_1$ is the part of the boundary with Neumann conditions. The test functions v must be zero on $\partial\Omega - \partial\Omega_1$.

Any continuous piecewise linear u_h is represented as a combination $u_h(x) = \sum_{i=1}^{N} U_i\phi_i(x)$, where ϕ_i are some special piecewise linear basis functions and U_i are scalar coefficients. Choose ϕ_i like a tent, such that it has the "height" 1 at the node i and the height 0 at all other nodes. For any fixed v, the FEM formulation yields an algebraic equation in the unknowns U_i. You want to determine N unknowns, so you need N different instances of v. What better candidates than $v = \phi_j$, $j = 1, 2, \ldots, N$? You find a linear system $KU = F$ where the matrix K and the right-hand side F contain integrals in terms of the test functions ϕ_i, ϕ_j and the coefficients defining the problem: c, a, f, q, and g. The solution vector U contains the expansion coefficients of u_h, which are also the values of u_h at each node x_i since $u_h(x_i) = U_i$.

If the exact solution u is smooth, then FEM computes u_h with an error of the same size as that of the linear interpolation. It is possible to estimate the error on each triangle using only u_h and the PDE coefficients (but not the exact solution u, which in general is unknown).

The PDE Toolbox provides functions that assemble K and F. This is done automatically in the graphical user interface, but you also have direct access to the FEM matrices from the command-line function `assempde`.

To summarize, the FEM approach is to approximate the PDE solution u of by a piecewise linear function u_h. u_h is expanded in a basis of test-functions ϕ_i, and the residual is tested against all the basis functions. This procedure yields a linear system $KU = F$. The components of U are the values of u_h at the nodes. For x inside a triangle, $u_h(x)$ is found by linear interpolation from the nodal values.

FEM techniques are also used to solve more general problems. Below are some generalizations that you can access both through the graphical user interface and with command-line functions.

- Time-dependent problems are easy to implement in the FEM context. The solution $u(x,t)$ of the equation

$$d\frac{\partial u}{\partial t} - \nabla\cdot(c\nabla u) + au = f$$

can be approximated by $u_h(x,t) = \sum_{i=1}^{N} U_i(t)\phi_i(x)$. This yields a system of ordinary differential equations (ODE) $M\frac{d}{dt}U + KU = F$ which you integrate using ODE solvers. Two time derivatives yield a second order ODE $M\frac{d^2}{dt^2}U + KU = F$, etc. The toolbox supports problems with one or two time derivatives (the functions `parabolic` and `hyperbolic`).

- Eigenvalue problems: Solve

$$-\nabla\cdot(c\nabla u) + au = \lambda du$$

for the unknowns u and λ (λ is a complex number). Using the FEM discretization, you solve the algebraic eigenvalue problem $KU = \lambda_h MU$ to find u_h and λ_h as approximations to u and λ. A robust eigenvalue solver is implemented in `pdeeig`.

- If the coefficients c, a, f, q, or g are functions of u, the PDE is called nonlinear and FEM yields a nonlinear system $K(U)U = F(U)$. You can use iterative methods for solving the nonlinear system. The toolbox provides a nonlinear solver called `pdenonlin` using a damped Gauss-Newton method.
- Small triangles are needed only in those parts of the computational domain where the error is large. In many cases the errors are large in a small region and making all triangles small is a waste of computational effort. Making small triangles only where needed is called adapting the mesh refinement to the solution. An iterative adaptive strategy is the following: For a given mesh, form and solve the linear system $KU = F$. Then estimate the error and refine the triangles in which the error is large. The iteration is controlled by `adaptmesh` and the error is estimated by `pdejmps`.

Although the basic equation is scalar, systems of equations are also handled by the toolbox. The interactive environment accepts u as a scalar or 2-vector function. In command-line mode, systems of arbitrary size are accepted.

If $c \geq \delta > 0$ and $a \geq 0$, under rather general assumptions on the domain Ω and the boundary conditions, the solution u exists and is unique. The FEM linear system has a unique solution which converges to u as the triangles become smaller. The matrix K and the right-hand side F make sense even when u does not exist or is not unique. It is advisable that you devise checks to problems with questionable solutions.