A Fast Björck-Pereyra-type algorithm for solving Complex-Vandermonde Systems

Austin Ogle and Jacky Qi Huang
Faculty Advisor; Sirani M. Perera
Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, FL.

INTRODUCTION

Modeling phenomenon of the interpolation problems can be seen in propagation of waves, weather conditions, real-time traffic patterns, etc. There are different interpolation techniques like polynomial interpolation, spline interpolation, rational interpolation, exponential interpolation, trigonometric interpolation, etc. One can see a polynomial interpolation problem as a system of equations problem. In here, we derive a fast $O(n^2)$ Björck-Pereyra-type algorithm to solve the system of equations.

Polynomials	Paper
Monomials	Bjorck-Pereyra (1970)
Real Orthogonal	Higham (1988)
Chebyshev	Rachel-Opfer (1991)
Szego	Bella et al. (2007)
Quasiseparable	Bella et al. (2009)
Complex	**????**

METHODOLOGY

We present the most general trigonometric interpolation problems to solve complex-Vandermonde system. We derive a fast $O(n^2)$ algorithm for solving a system of equations $W\vec{a} = \vec{f}$ where the coefficient-matrix is a complex Vandermonde matrix $W = [\omega_j^k]_{j,k=0}^{n-1}$ having $\omega = e^{-ix}$ and $i^2 = -1$. This method is much more favorable than Gaussian elimination which requires $O(n^3)$ complexity. This result generalizes the classical Björck-Pereyra algorithm from monomials to complex system $\{1, \omega, \omega^2, \cdots, \omega^{n-1}\}$, where nodes are taken along the unit circle. The new algorithm applies to a fairly general class to solve trigonometric interpolation problems. We present numerical experiments together with the better forward error bound than the Gaussian elimination.

Our Problem is to construct a complex polynomial

$$P(\omega) = \beta_0 + \beta_1\omega + \beta_2\omega^2 + \cdots + \beta_{n-1}\omega^{n-1}$$

where, $\omega^k = e^{-ix_k}$, $x_k = \frac{2\pi k}{n}$, and $i^2 = -1$, using the given data points (ω_k, f_k) for $k = 0,1,2,\ldots,n-1$.

One can convert the corresponding system of equations into the matrix-vector form as follows;

$$\begin{bmatrix} 1 & e^{-ix_0} & e^{-2ix_0} & \cdots & e^{-(n-1)ix_0} \\ 1 & e^{-ix_1} & e^{-2ix_1} & \cdots & e^{-(n-1)ix_1} \\ 1 & e^{-ix_2} & e^{-2ix_2} & \cdots & e^{-(n-1)ix_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-ix_{n-1}} & e^{-2ix_{n-1}} & \cdots & e^{-(n-1)ix_{n-1}} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n-1} \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{bmatrix}$$

RESULTS

To solve the system of equations, two factorizations were found; Type 1 & Type 2. Type 1 solves the system iteratively. This is a generalization of the original 1970 Bjork-Pereyra algorithm to the complex plane. Type 2 solves the system recursively. The factorization of Type 2 holds the subtraction to a single value for the calculation of lower triangular matrices. This reduces the point floating point error in the sequent calculations and even leads to more accurate algorithm.

Sparse Factorization: Type 1

$$V(\omega_{1:n})^{-1} = U_1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & V(\omega_{2:n})^{-1} \end{bmatrix} \cdot \tilde{L}_1$$

$$U_k = \begin{bmatrix} I_{k-1} & & & & \\ & 1 & -\omega_i & & \\ & & 1 & \ddots & \\ & & & \ddots & -\omega_i \\ & & & & 1 \end{bmatrix}$$

$$\tilde{L}_k = \begin{bmatrix} I_k & & & \\ & \frac{1}{\omega_{k+1} - \omega_1} & & \\ & & \ddots & \\ & & & \frac{1}{\omega_n - \omega_{n-k}} \end{bmatrix} \cdot \begin{bmatrix} I_{k-1} & & & & \\ & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}$$

Sparse Factorization: Type 2

$$U_k = \begin{bmatrix} I_{k-1} & & & & \\ & 1 & -\omega_i & & \\ & & 1 & \ddots & \\ & & & \ddots & -\omega_i \\ & & & & 1 \end{bmatrix}$$

$$D_k = \begin{bmatrix} I_k & & & \\ & \frac{1}{\omega_{k+1} - \omega_k} & & \\ & & \ddots & \\ & & & \frac{1}{\omega_n - \omega_k} \end{bmatrix}, L_k = \begin{bmatrix} I_{k-1} & & & \\ & 1 & & \\ & -1 & 1 & \\ & \vdots & & \ddots & \\ & -1 & & & 1 \end{bmatrix}$$

Algorithms for Type 1 and Type 2

Type 1

Input: n, f_k, $x_k = \frac{-2\pi(k-1)}{n}$, for k = 1,2,…,n

Recursion:

for k = 1,2,…,n
  $\beta_k = f_k$
end

for k = 1,2,…,n-1
  for j = n, n-1,…,1
    $\beta_j = \frac{\beta_j - \beta_{j-1}}{e^{ix_j} - e^{ix_{j-k}}}$
  end
end
for k = n-1, n-2,…,1
  for j = k, k+1,…,n-1
    $\beta_j = \beta_j - \beta_j e^{ix_k}$
  end
end

Output: β_k, for k = 1,2,…,n

Type 2

Input: n, f_k, $x_k = \frac{-2\pi(k-1)}{n}$, *for k = 1,2,…,n*

Recursion:

for t = 1, 2, …, n-1
  $u = I_n$
  for j = t, t+1, …, n-1
    $u_{(n+1)j} = -e^{ix_t}$
  end
  $U_t = u$
  $d = I_n$
  for j = t+1, t+2, …, n
    $d_{n(j-1)+j} = \frac{1}{e^{ix_j} - e^{ix_t}}$
  end
  $D_t = d$
  $l = I_n$
  $l_{t+1:end,t} = -1$
  $L_t = l$
end

$\underline{\beta} = U_1U_2 \ldots U_{n-1}D_{n-1}L_{n-1}D_{n-2}L_{n-2} \ldots D_1L_1\underline{f}$

Output: $\underline{\beta} = \beta_k$, *for k = 1,2,…,n*

RESULTS

Forward Error Bound for Both Algorithms

$$n \geq 4, \quad u := \text{unit roundoff}$$

For $\underline{\beta} = V^{-1}\underline{f}$

$$\frac{\|\beta - \hat{\beta}\|_2}{\|\beta\|_2} \leq (1-5u)^{1-n}\|U_1\|_2 \cdots \|U_{n-1}\|_2\|D_{n-1}\|_2\|L_{n-1}\|_2 \cdots \|D_1\|_2\|L_1\|_2$$

Numerical Results

Size	BP-Type 1	BP-Type 2	Gaussian Elimination	BP-Type 1 Leja Ordered	BP-Type 2 Leja Ordered
10×10	1.491E-06	1.843E-08	2.645E-08	1.648E-07	1.894E-08
20×20	3.876E-04	1.619E-08	2.391E-08	4.228E-07	2.445E-08
30×30	8.302E-02	7.940E-09	2.517E-08	5.560E-07	1.955E-08
35×35	**3.425E+00**	2.266E-08	1.733E-08	5.657E-07	2.933E-08
40×40	**2.443E+02**	1.338E-07	3.334E-08	1.098E-06	1.262E-08
50×50	**7.089E+06**	3.193E-05	1.123E-05	1.316E-06	4.034E-08
60×60	**1.657E+12**	1.389E-02	3.916E-04	1.708E-06	1.433E-08
70×70	**9.646E+16**	**3.948E+00**	6.277E-03	1.863E-06	2.476E-08
80×80	**1.898E+21**	**3.533E+04**	3.991E-02	3.741E-06	2.226E-08
90×90	**6.079E+26**	**6.264E+09**	1.168E-01	3.155E-06	4.361E-08
100×100	**7.919E+31**	**9.359E+14**	2.844E-01	2.604E-06	1.474E-08
105×105	**6.272E+34**	**1.781E+17**	**2.624E+00**	3.422E-06	1.467E-08
150×150			**8.979E+00**	6.321E-06	1.558E-08
200×200			**3.171E+01**	9.046E-06	1.840E-08
250×250			**1.953E+02**	1.138E-05	4.871E-08
300×300			**1.087E+02**	1.841E-05	2.596E-08

CONCLUSION

Results on polynomial interpolation from real to complex plane leads to:

- Fast $O(n^2)$ Algorithms
- Sparse Factorizations
- Iterative Algorithms
- Stable Algorithms with Leja Ordering
- Accurate Algorithms beyond Gaussian Elimination for $n > 100$

REFERENCES

- [1] A. Bjork and V. Pereyra, Solution of Vandermonde Systems of Equations, Mathematics of Computation (American Mathematical Society) 24 (112), (1970)
- [2] N. J. higham, Fast Solution of Vandermonde-Like Systems Involving Orthogonal Polynomials, IMA Journal of Numerical Analysis 8 (4):473-486(1988)
- [3] I. Gohberg and V. Olshevsky, Fast inversion of Chebyshev-Vandermonde matrices, Numerische Mathematik 61 (1), 71-92, (1994).
- [4] T.Bella, Y.Eidelman, I. Gohberg, I. Koltracht and V.Olshevsky, A Bjork-Pereyra –type algorithm for Szego-Vandermonde matrices based on the properties of unitary Hessenberg matrices, Linear Algebra and Applications 420(2-3):634-647, (2007)
- [5] T.Bella, Y.Eidelman, I.Gohberg, I Koltracht and V.Olshevsky, A fast Bjork-Pereyra-type algorithm for solving Hessenberg-quasieparable-Vandermonde systems, SIAM, J. Matrix Anal. And Appl. 31(2):790-815, (2009).