NASA CR-181,677

NASA Contractor Report 181677

ICASE REPORT NO. 88-39

ICASE

HIGH DEGREE INTERPOLATION
POLYNOMIAL IN NEWTON FORM

Hillel Tal-Ezer

Contract No. NAS1-18107
June 1988

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

HIGH DEGREE INTERPOLATION POLYNOMIAL IN NEWTON FORM

Hillel Tal-Ezer
Division of Applied Mathematics
Brown University
and
Institute for Computer Applications in Sciences and Engineering

ABSTRACT

Polynomial interpolation is an essential subject in numerical analysis. Dealing with a real interval, it is well-known that even if $f(x)$ is an analytic function, interpolating at equally spaced points can diverge [Davi75]. On the otherhand, interpolating at the zeroes of the corresponding Chebyshev polynomial will converge. Using the Newton formula, this result of convergence is true only on the theoretical level. It is shown that the algorithm which computes the divided differences is numerically stable only if: 1.) the interpolating points are arranged in a certain order, 2.) the size of the interval is 4.

This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18107 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665. Additional support was provided by the Air Force Office of Scientific Research Grant No. 85-0303.

N88-25249 #

1. INTRODUCTION

Let $f(x)$ be a real function defined on the interval $[a,b]$ and $\{x_i\}_{i=o}^{N}$ be a set of $N+1$ points in $[a,b]$ then the general formulation of Newton interpolating polynomial of degree N is

$$P_N(x) = \sum_{k=0}^{N} a_k R_k(x) \tag{1.1}$$

where a_k are the divided differences

$$a_k = f[x_0, \cdots, x_k] \tag{1.2}$$

and

$$R_0(x) = 1 \tag{1.3}$$

$$R_k(x) = \Pi_{i=0}^{k-1}(x - x_i). \tag{1.4}$$

If $E_N(\overline{x})$ is the error at point $\overline{x}\epsilon[a,b]$ then we have the following theorem [CodB82].

Theorem 1.1: Let $f(x)$ be a real valued function defined on $[a,b]$ and $N+1$ times differentiable on $[a,b]$. If $P_N(x)$ is the polynomial of degree $\leq N$ which interpolates $f(x)$ at $N+1$ distinct points $x_0, \cdots, x_N$ in $[a,b]$, then for all $\overline{x}\epsilon[a,b]$, there exists $\xi = \xi(\overline{x})\epsilon(a,b)$ such that

$$E_N(\overline{x}) = f(\overline{x}) - P_N(\overline{x}) = \frac{f^{(u+1)}(\xi)}{(u+1)!} R_{N+1}(\overline{x}). \tag{1.5}$$

It is well-known that if $x_0, \cdots, x_N$ are equally spaced points then $max|R_{N+1}(\overline{x})|$ increases as $\overline{x}$ moves towards the ends of the interval and divergence can occur. A well-known remedy for this phenomenon is to choose as interpolating points the zeroes of the corresponding scaled and translated Chebyshev polynomial

$$x_i = \frac{1}{2}\left[(b-a)cos\frac{(2i+1)\pi}{2N+2} + b + a\right] \qquad i = 0, \cdots, N. \tag{1.6}$$

Using (1.6) results in a uniform distribution of the error and convergence is achieved.

This result of convergence is true only in theory. Practically, an interpolating polynomial based on (1.6) <u>will not</u> converge to $f(x)$ because of the finite accuracy of the computer. The numerical instability can be traced to two sources:

1.) The algorithm which computes the divided differences is very vulnerable to roundoff errors and overflow. The super sensitivity to roundoff errors is explained by the fact that for N large, the first k points ($k \ll N$) are very close to each other ($min|x_{i+1} - x_i| \sim 1/N^2$). But even if roundoff errors were eliminated, we still would face overflow because the first k points ($k \ll N$) are concentrated on one side of the interval. This distribution will lead to nonuniformity of $R_k(x)$ and $E_k(x)$. It is this nonuniformity which will cause overflow at intermediate stages of the interpolation process.

2.) Eliminating the first source of numerical instability (by taking the points in a different order) results in "almost" uniformity of $R_k(x)$ $k = 1, .., N$, but still, $R_k(x)$ satisfies (see Section 3)

$$|R_k(x)| \sim 2^{-k}. \tag{1.7}$$

Observing (1.1) and (1.7) it is obvious that we will face overflow (for k large enough) while computing the $a_k's$.

We are going to approach this phenomenon in the more general context of interpolation in the complex plane. Background material is given in Section 2. Based on the theory we show that by:

(1) arranging the interpolating points in a certain order,

(2) making a simple change of variables such that the interval is of size 4,

we get

$$|R_k(x)| \sim 1 \qquad 1 \leq k \leq N. \tag{1.8}$$

Thus these two modifications result in a stable Newton interpolation process. Algorithm 2.1 presented in Section 2 generates N interpolating points such that (1.8) is satisfied. In practice we would like to be able to add points, if necessary, without restarting the interpolation process. It is well-known that the Newton algorithm potentially has this feature; adding another point results in computing only one additional term. (In contrast to the Lagrange process where one has to start the calculations all over again.) But by using Chebyshev zeroes, (1.6), we would not capitalize on this property since increasing

N results in changing all the previous points. To this end, we construct Algorithm 2.2 which enables us to add interpolating points such that (1.8) is asymptotically satisfied. Observe from (1.1) and (1.8) that the a_k's behave asymptotically like the interpolation error. Thus one can use the algorithm which computes the divided differences (related to the points generated by Algorithm 2.2) as a numerical device for estimating the degree of the polynomial needed to achieve a given accuracy. This technique can be very useful when the mathematical expression of the error (1.5) is difficult to analyze.

The need for high degree interpolation is confronted in approximating a finite operator which can be presented as $f(A)$ where A is another finite operator. (The case $f(z) = exp(z)$ is a popular example [MoVa78].) Approximating $f(A)$ can be reduced to a problem of approximating $f(z)$ where z belongs to a domain D in the complex plane which includes all the eigenvalues of A. A possible approach is to expand the function as a sum of orthogonal polynomials. In [Tale86], [Tale85] we have used it for the function $f(z) = exp(tz)$ where the domains were [-iR,iR], [-R,0] respectively. The algorithm which results make use of the three term recurrence relation satisfied by the polynomials (scaled and translated Chebyshev). For more complicated domains, the related polynomial of degree k satisfies a k term recurrence relation and therefore the expansion approach is not suitable. An alternative way is to use interpolation. A brief description of the method is given in Section 4.

A particular and very important case of polynomial approximation of $f(A)$ is an iterative solution to a linear system $Ax = b$. Finding optimal parameters α_k for solving $Ax = b$ by the Richardson algorithm

$$x^{k+1} = x^k - \alpha_k(Ax^k - b) \tag{1.9}$$

can be achieved by considering polynomial interpolation to the function $f(z) = 1/z$ [Tale87]. When D is on one side of the real line, this method is widely treated in the numerical analysis literature (Chebyshev acceleration [HaYo81]). It is also known that the fact of having to decide on the number of iterations before starting the algorithm reduces its efficiency. A Richardson process which uses points generated by Algorithm 2.2 is free of this disadvantage. (For a more elaborate discussion and numerical examples see [Tale87].) We conclude the paper in Section 5 by giving some numerical results.

2. INTERPOLATION IN THE COMPLEX PLANE

Let D be a bounded continuum in C such that the complement of D is simply connected in the extended plane and contains the point at infinity. Considering interpolation in the domain D, one is faced with the problem – which are the "good" interpolating points? The solution to this question is based on the following:

Let $\phi(z)$ be a conformal mapping which maps the complement of D to the complement of a disc of radius ρ such that

$$\lim_{z\to\infty} \frac{\phi(z)}{z} = 1. \tag{2.1}$$

ρ is the logarithmic capacity of D [SmLe68]. (Having a domain D, ϕ and ρ are defined uniquely.) Define $\psi(\omega)$ to be the inverse of $\phi(z)$. Then we have [Wals56]:

Definition 2.1: Let Γ_R be the image under ψ of the circle $|w| = R \quad (R > \rho)$ and I_R be the closed Jordan region whose boundary is Γ_R. If $f(z)$ is single valued and analytic on I_R then the sequence of polynomials $P_m(z)$ is said to converge to $f(z)$ on D maximally if

$$|f(z) - P_m(z)| \le C(\rho/R)^m \qquad z \epsilon D \tag{2.2}$$

where C depends on ρ/R but not on m or z.

Definition 2.2: The set of interpolating points $z_j = \psi(w_j)$ is said to be uniformly distributed on Γ_D (the boundary of D) if w_j are equally distributed on the circle $|w| = \rho$.

This set (known also as Fejer points) is one possible set of "good" points. The polynomial which interpolates $f(z)$ at these points satisfies (2.2) [Wals56].

Another possibility is to interpolate at the zeroes of the corresponding Faber polynomial. Let D and $\phi(z)$ be as defined previously. We have [Mark77]

$$[\phi(z)]^m = C_m z^m + \cdots + C_0 + C_{-1}/z + C_{-2}/z^2 + \cdots. \tag{2.3}$$

The Faber polynomial of degree m related to D is the polynomial part of (2.3)

$$F_m(z) = C_m z^m + \cdots + C_0. \tag{2.4}$$

Interpolating at the zeroes of $F_m(z)$ satisfies (2.2) [Mark77]. When

$$D = \{z| -1 \leq Rez \leq 1 \qquad I_m z = 0\}, \tag{2.5}$$

then

$$F_m(z) = T_m(z) = cos(mcos^{-1}(z)), \tag{2.6}$$

and the m zeroes are the Chebyshev points (1.6).

For a general domain in the complex plane finding the zeroes of $F_m(z)$ for large m can be troublesome. Thus, it is preferable to use Fejer points since only knowledge of $\psi(w)$ is required.

Assume now that $z_j, \quad 1 \leq j \leq m,$ are uniformally distributed points. Then [Wals56]:

$$|R_m(z)| \sim \rho^m. \tag{2.7}$$

From (2.7) it is clear that in order to satisfy (1.8) we need:

1. $\rho = 1$
2. Every subset of interpolating points $z_i, \cdots, z_k, \qquad 1 \leq k \leq N$, has to be uniformly (or "almost" uniformly, see Algorithm 2.1) distributed.

Hence, by making the change of variables

$$\hat{z} = z/\rho \tag{2.9}$$

and arranging the interpolating points such that the second requirement is satisfied we eliminate the numerical instability mentioned in Section 1. The following algorithm is designed for this purpose.

Algorithm 2.1: [Generates m uniformly distributed points (m even)]

$$\theta_1 = 0 \qquad j = \ell = 1 \qquad \delta\theta = 2\pi/m. \tag{2.13}$$

Find the largest k such that k is power of 2 and $k < m/2$

$$\theta k = k \times \delta\theta \tag{2.14}$$

```
1       For $i = 1$ until $\ell$ do:
                    $\eta = \theta_i + \theta k$
                    $if(\eta \geq \pi)$ go to 1
                    $j = j + 1$                              (2.15)
                    $\theta_j = \eta$
        end do
                    $\ell = j$
                    $\theta k = \theta k/2$
                    $if(\theta k < \delta\theta) stop$
                    go to 1
```

Algorithm 2.1 generates $\frac{m}{2}$ arguments of points on the upper part of the unit circle (includes 1 but not -1). Thus

$$w_1 = 1 \qquad z_1 = \psi(w_1) \tag{2.16a}$$

$$w_2 = -1 \qquad z_2 = \psi(w_2) \tag{2.16b}$$

$$w_{2j-1} = exp(i\theta_j) \qquad z_{2j-1} = \psi(w_{2j-1}) \qquad 2 \leq j \leq \frac{m}{2} \tag{2.16c}$$

$$w_{2j} = -w_{2j-1} \qquad z_{2j} = \psi(w_{ij}) \qquad 2 \leq j \leq \frac{m}{2}. \tag{2.16d}$$

Using Algorithm 2.1 results in uniform distribution of $z_1, \cdots, z_m$ while $z_1, \cdots, z_k (k < m)$ are "almost" uniformly distributed.

As mentioned in the introduction, we would like to have an interpolating process which will allow us to add points if desired. To this end, we present the next algorithm. It generates an infinite set of interpolating points. Using this algorithm we do not have decide on the degree of the polynomial ahead of time. The set is asymptotically uniformally distributed and therefore the interpolating polynomial satisfies (2.2).

Algorithm 2.2:

```
                    $\delta\theta = \pi$
                    $k = 1$
                    $z_1 = \psi(1)$
1   For $i = 1$ until $k$ or until satisfied do
                    $\theta_{k+i} = \theta_i + \delta\theta$
                    $z_{k+i} = \psi(exp(i\theta_{k+i}))$
    end do
                    $\delta\theta = \delta\theta/2$
                    $k = 2k$
                    go to 1
```

The set of points generated by Algorithm 2.2 is uniformally distributed when the number points is a power of two and "almost" uniformally distributed otherwise.

Remark: In order to implement the algorithms presented in this section, we need the conformal mapping function. For some domains, we do have an analytic expression of this function. In more complicated situations, one has to resort to numerical techniques. When D is a polygon, $\psi(w)$ is a Schwartz-Cristoffel transformation. Numerical routines for this case, written by L. N. Trefethen (based on [Tref80]), are available through the Netlib facilities. (See also [Tale87] for a description of how to implement the routines.)

3. INTERPOLATION ON THE REAL LINE

Let

$$D = [-b, b]. \tag{3.1}$$

The conformal mapping $\tilde{\psi}(w)$ which maps the complement of the unit disc on the complement of D is [Mark77]

$$\tilde{\psi}(w) = \frac{b}{2}(w + w^{-1}). \tag{3.2}$$

Using (2.1) we get that

$$\rho = b/2. \tag{3.3}$$

Therefore, one should make the following change of variables

$$\hat{x} = \frac{2x}{b} \tag{3.4}$$

in order to get

$$\hat{D} = [-2, 2] \tag{3.5}$$

$$\hat{\rho} = 1. \tag{3.6}$$

Similarly, if

$$D = [a, b] \tag{3.7}$$

then

$$\hat{x} = \frac{4x}{b - a}. \tag{3.8}$$

Thus, without loss of generality, we assume that

$$D = [-2, 2]. \tag{3.9}$$

Using (3.2), we get that Fejer points are

$$x_j = w_j + w_j^{-1} \tag{3.10}$$

where w_j can be generated by the algorithms given in Section 2. Observe that the x_j 's are double interpolation points (except for 2 and -2).

Faber polynomials which correspond to D are scaled Chebyshev polynomials [Mark77]

$$\tilde{T}_m(x) = cos(m(cos^{-1}(\frac{x}{2}))) \tag{3.11}$$

whose zeroes are

$$y_j = 2cos\frac{(2j-1)\pi}{2m} \qquad j = 1, \cdots, m. \tag{3.12}$$

Using Algorithm 2.1, we can get the zeroes of $\tilde{T}_m(x)$, arranged in a stable order, as follows:

$$x_j = 2cos(\theta_j + \frac{\delta\theta}{2}) \qquad j = 1, \cdots, m. \tag{3.13}$$

A popular set of interpolating points on the real line is the extremas of Chebyshev polynomial of degree N. This set is not exactly Fejer points but can be shown to satisfy (2.2). Using the algorithms given in Section 2 we have

$$x_1 = 2; \qquad x_2 = -2 \tag{3.14a}$$

$$x_j = 2cos\theta_j \qquad j \geq 3 \tag{3.14b}$$

where θ_j are generated by Algorithm 2.1. or

$$x_1 = 2 \qquad x_2 = -2 \tag{3.15a}$$

$$x_j = 2cos\theta_{2j+1} \qquad j \geq 1 \tag{3.15b}$$

where θ_{2j+1} are generated by Algorithm 2.2.

4. POLYNOMIAL APPROXIMATION OF A FUNCTION OF A MATRIX

Let A be an $N \times N$ matrix and $f(z)$ a function analytic in a domain D in the complex plane which includes all the eigenvalues of A. Let w be the vector which results from operating with $f(A)$ on a vector v

$$w = [f(A)]v. \tag{4.1}$$

Getting an approximation of w is a problem frequently confronted in applied mathematics. An elaborate description of this topic is given in [Tale87]. It is shown there that one can get an "almost" optimal polynomial algorithm by using the polynomial $P_m(z)$ which interpolates $f(z)$ at points uniformally distributed on the boundary of D. (Without loss of generality we assume that $\rho(D) = 1$. If not, we define $\hat{A} = \frac{1}{\rho}A$ and consider $\hat{f}(\hat{z}) = f(\rho\hat{z}) = f(z)$). Let $\tilde{w}$ be an approximation of w

$$\tilde{w} = [P_m(A)]v. \tag{4.2}$$

When A is normal, the error vector satisfies

$$\|w - \tilde{w}\| = \|(f(A) - P_m(A))v\| \le |f(z) - P_m(z)| \; \|v\|. \tag{4.3}$$

Hence

$$E = \frac{\|w - \tilde{w}\|}{\|v\|} \le |f(z) - P_m(z)|. \tag{4.4}$$

Using (2.2) we get that E is asymptotically bounded by $(\frac{1}{R})^m$. Thus one can get a prediction of the degree of the approximating polynomial. (When $f(z)$ is an entire function ($R = \infty$) one can use the idea of computing the corresponding divided differences in order to get an estimation of m.) In this case, we will use Algorithm 2.1 as a generator of the interpolating points z_i. Once we have z_i we compute the divided differences a_i. Having these two sets of numbers, we can write the following algorithm:

Algorithm 4.1:

```
        $u = v$
        $\tilde{w} = a_1 u$
For $i = 2, \cdots, m$ do
        $u = (A - z_{i-1} I)u$
        $\tilde{w} = \tilde{w} + a_i u$
end do
```

When A is diagonalizable but not normal we have the following

$$E \le \|T\| \; \|T^{-1}\| \; |f(z) - P_m(z)| \tag{4.5}$$

where T is the matrix which diagonalize A. Since $\|T\|\ \|T^{-1}\|$ is unknown we usually would not have an estimation of m before starting the algorithm. In this case we can use Algorithm 2.2 as a generator of points. The criterion for stopping will be the norm of the relative residual $\|a_i u\|/\|\tilde{w}\|$. (Because of the asymptotic behavior, the decision should be made checking by a few residuals.)

Algorithm 4.2:

```
                    $u = v$
                    $\tilde{w} = a_1 u$
For $i = 2, \cdots$,until satisfied do
                    $u = (A - z_{i-1} I) u$
                    $\tilde{w} = \tilde{w} + a_i u$
                    check for convergence
```

When A is a real matrix and $f(z)$ is real for z real, the algorithms have to be modified in order to eliminate complex arithmetic. (Observe that in this case the domain is symmetric around the real axis.) To this end, we first arrange the points such that each point with nonzero imaginary part will be followed by its conjugate. Thus, we change (2.16d) to read $w_{2j} = \overline{w}_{2j-1}$. The modified version of Algorithm 4.1 is:

Algorithm 4.3:

```
                    $w = [a_1 I + a_2(A - z_1 I)] v$
                    $r = (A - z_1 I)(A - z_2 I) v$
For $i = 1, \cdots, \frac{m-1}{2}$ do
                    $\tilde{r} = (A - z^R_{2i+1} I) r$
                    $\tilde{w} = \tilde{w} + a^R_{2i} r + a^R_{2i+1} \tilde{r}$
                    $r = (A - z^R_{2i+1} I)\tilde{r} + (z^I_{2i+1})^2 r$
end do
```

(The superscripts R, I stand for the real and imaginary parts respectively.)

Using Algorithm 2.2, we will order the points as follows: $z_1, z_2, z_3, \overline{z}_3, \cdots$,where z_{2i+1} are generated by the algorithm. The modified version of Algorithm 4.2 will read:

Algorithm 4.4:

```
        $w = [a_1 I + a_2(A - z_1 I)]v$
        $r = (A - z_1 I)(A - z_2 I)v$
For $i = 1, \cdots$,until satisfied do
        $\tilde{r} = (A - z_{2i+1}^R I)r$
        $\tilde{w} = \tilde{w} + a_{2i}^R r + a_{2i+1}^R \tilde{r}$
        $r = (A - z_{2i+1}^R I)\tilde{r} + (z_{2i+1}^I)^2 r$
        check for convergence
```

An iterative solution to a general linear system

$$Ax = b \tag{4.7}$$

is a particular case of approximating $f(A)$ where $f(z) = \frac{1}{z}$. Applying Algorithms 4.1 - 4.4 for this problem can be shown to be equivalent to a Richardson type process [Tale87]. Having an initial approximation x^1, Algorithm 4.1 will result in

Algorithm 4.5:

```
For j = 1 until $m$ do
        $x^{j+1} = x^j - \alpha_j(Ax^j - b)$
end do
```

Where the relaxation parameters are

$$\alpha_j = \frac{1}{z_j}. \tag{4.8}$$

Thus, for this particular function we do not have to compute the divided differences. Therefore, the change of variables (2.9) is not significant here. But still rearranging the points is important. Otherwise, we can face numerical instability due to the nonuniformity of the error at intermediate stages. This phenomenon is treated also in [AnGo72], [LeFi76], and [FiRe87]. The algorithm (based on bit-reversed binary representation of the iteration count) proposed in [FiRe87] results in the same set of parameters generated by Algorithm 2.2. When A is real, Algorithm 4.4 will read

Algorithm 4.6:

```
                 $x^2 = x^1 - \alpha_1(Ax^1 - b)$
                 $x^3 = x^2 - \alpha_2(Ax^2 - b)$
for $j = 2$ until satisfied do
                 $R^j = Ax^{2j-1} - b$
                 $x^{2j+1} = x^{2j-1} - [2\alpha_j^R - |\alpha_j|^2 A]R^j$
                 check the residuals
end do
```

5. NUMERICAL RESULTS

All the numerical experiments reported in this section were performed on an IBM 3270. Let $P_m(x)$ be the interpolating polynomial. The maximal error

$$E = \max_{x \epsilon [-1,1]} |f(x) - P_m(x)| \tag{5.1}$$

was approximated by

$$E_{ap} = \max_i |f(x_i) - P_m(x_i)| \tag{5.2}$$

where x_i are checkpoints

$$x_i = -1 + 2(i-1)/19 \qquad 1 \leq i \leq 20. \tag{5.3}$$

In the first set of experiments, the interpolated function is

$$f(x) = \frac{1}{0.005 + x^2} \qquad -1 \le x \le 1. \tag{5.4}$$

By the following change of variables

$$y = xb \tag{5.5}$$

we get

$$f(x) = g(y) = \frac{1}{0.005 + (\frac{y}{b})^2} \qquad -b \le y \le b. \tag{5.6}$$

The interpolating points are uniformally distributed on $[-b, b]$ (as defined in Section 2) but taken in two different orderings: y_j^1 are arranged in the standard order

$$y_j^1 = b\cos(\pi j/m) \qquad 0 \le j \le m \tag{5.7}$$

and y_j^2 are generated by Algorithm (2.1). In the first experiment we have used single precision. The polynomial is of degree 80. Selected divided differences and the numerical maximal error are presented in Table 1.

Table 1. $(m = 80)$

k	b=1		b=2		b=3	
	$a_k(y^1)$	$a_k(y^2)$	$a_k(y^1)$	$a_k(y^2)$	$a_k(y^1)$	$a_k(y^2)$
10	-5.44+09	-4.75+03	-1.06+07	-9.28+00	-2.08+04	-1.81−02
20	-5.31+17	2.045+06	-1.02+12	3.89+00	-1.93+06	7.44−06
30	-1.75+21	-8.94+08	-3.26+12	-1.66+00	-6.07+03	-3.10−09
40	-1.46+22	3.45+11	-2.65 +10	6.36−01	-4.83−02	1.16−12
50	-2.54+21	-2.14+14	-4.50+06	-3.80−01	-8.00−09	-6.74−16
60	-4.28+21	9.27+16	-7.42+03	1.61−01	-1.29−14	2.80−19
70	-1.73+22	-3.43+19	-2.94+01	-5.82−02	-4.98−20	-9.86−23
80	-8.79+16	2.90+18	-1.46−07	4.80−02	-2.41−31	7.94−30
E_{ap}	4.68+23	3.48−01	4.68+23	3.48−01	—	3.48−01

Table 1 verifies the fact that y_j^1 are useless for high degree Newton interpolation. Since no overflow (underflow) has occurred in computing the divided differences, the error related to y_j^2 is the same in all the three cases. However, observe the "nice" behavior of the coefficients which correspond to $b = 2$. The last ones are close to the interpolation error, as predicted by the theory.

Next we took $m = 160$. The results are presented in Table 2.

Table 2. $(m = 160)$

k	b=1		b=2		b=3	
	$a_k(y^1)$	$a_k(y^2)$	$a_k(y^1)$	$a_k(y^2)$	$a_k(y^1)$	$a_k(y^2)$
20	3.91+29	2.42+06	7.46+23	4.62+00	1.42+18	8.81−06
40	—	5.44+11	6.49+33	9.90−01	1.18+22	1.78−12
60	—	1.36+17	9.31+34	2.35−01	1.65+17	4.08−19
80	—	2.34+17	7.43+30	3.88−02	1.23+07	6.43−26
100	—	7.24+27	2.10+23	1.14−02	3.31−07	1.80−32
120	—	1.67+33	3.28+13	2.51−03	4.93−23	3.77−39
140	—	—	1.01+02	2.45−04	1.45−40	0.00+00
160	—	—	-5.35−08	-2.60−05	0.00+00	0.00+00
E_{ap}	—	—	—	7.29−04	—	—

Now, overflow or underflow is taking over in all the cases except for y_j^2 and $b = 2$. As in the previous example, the last coefficients are close to the error itself.

Next we interpolated the function

$$f(x) = \cos(2000x) \qquad -1 \leq x \leq 1. \tag{5.8}$$

In this case, one needs a super high degree polynomial. We took $m = 2100$ and used double precision. The divided differences are presented in Table 3.

Table 3. $(m = 2100)$

k	$\lvert a_k\rvert$	k	$\lvert a_k\rvert$	k	$\lvert a_k\rvert$	k	$\lvert a_k\rvert$
1	3.67−01	801	2.36−02	1601	1.71−02	2031	1.98−05
201	8.52+00	1001	3.80−03	1801	4.47−02	2051	1.12−06
401	3.78−01	1201	1.51−02	2001	5.13−02	2071	5.77−08
601	5.69−03	1401	1.22−02	2011	2.78−03	2091	2.18−10

The numerical error is

$$E_{ap} = 5.89 - 09. \tag{5.9}$$

Observe that the asymptotic decay of the coefficients starts when k crosses 2000. One encounters a similar behavior of coefficients while expanding (5.8) in Chebyshev polynomials.

$$\cos(2000x) = \sum_{k=0}^{\infty} b_k T_k(x). \qquad -1 \le x \le 1 \tag{5.10}$$

It can be shown [AbSt72] that

$$b_k = \begin{cases} 0 & k \; odd \\ 2J_k(2000) & k \; even \end{cases} \tag{5.11}$$

where J_k are Bessel functions of order k. By asymptotic analysis of Bessel functions we know that $J_k(R)$ goes to zero exponentially fast only when $k > R$. This similarity between a_k and b_k could be anticipated since $T_k(x)$ satisfies

1.$|T_k(x)| \le 1 \qquad |x| \le 1$

2. Its zeroes are uniformally distributed.

Where $R_k(x)$ satisfies

1.$|R_k(x)| \sim 1 \qquad |x| \le 2$

2. Its zeroes are "almost" uniformally distributed.

Polynomial approximation of a function of a matrix

Let us consider the following parabolic P.D.E

$$U_t = U_{xx} + U_{yy} \qquad |x| < 1, |y| < 1 \tag{5.12a}$$

$$U(x,0) = \sum_{k=1}^{4} sin(\pi kx) sin(\pi ky) \tag{5.12b}$$

$U = 0$ on the boundaries of the unit square.

Using the following space discretization

$$U_{ij} = U(x_i, y_j, t) \tag{5.13a}$$

$$x_i = -1 + i\Delta x \qquad 1 \le i \le N \tag{5.13b}$$

$$y_j = -1 + j\Delta y \qquad 1 \le j \le N \tag{5.13c}$$

$$\Delta x = \Delta y = \frac{2}{N+1} \tag{5.13d}$$

where N is the number of grid points in each direction, we approximate the space derivatives by the standard second order central differencing

$$U_{xx} \sim (U_{i+1,j} - 2U_{ij} + U_{i-1,j})/\Delta x^2 \tag{5.14a}$$

$$U_{yy} \sim (U_{i,j+1} - 2U_{ij} + U_{i,j-1})/\Delta y^2. \tag{5.14b}$$

Thus we get the following semidiscrete representation of (5.12)

$$(U_N)_t = G_N U_N \tag{5.15a}$$

$$U_N^0 = U^0 \tag{5.15b}$$

where U_N is a vector of dimension N^2, G_N is a $N^2 \times N^2$ matrix which results from the space discretization and

$$U_{ij}^0 = U(x_i, y_j, 0) \qquad 1 \leq i, j \leq N. \tag{5.16}$$

The formal solution of (5.15) is

$$U_N = exp(tG_N)U_N^0 \tag{5.17}$$

Now, by using polynomial approximation of $exp(tG_N)$ we get

$$U_N^k = P_k(tG_N)U_N^0 \tag{5.18}$$

where $P_k(tz)$ approximates $exp(tz)$ in the domain which includes all the eigenvalues of G_N. The numerical solution was computed at $t = 0.1$. A simple Fourier analysis shows that D is on the negative real line. Therefore we took (3.12b) as interpolating points and used Algorithm 4.2 to get U_N^k. The exact solution of (5.12) is

$$U(x, y, t) = \sum_{k=1}^{4} exp(-(k\pi)^2 t) sin(k\pi x) sin(k\pi y). \tag{5.19}$$

The following tables present the dependence of three factors on k, the degree of the interpolating polynomial.

$|a_k|-$ the absolute value of the maximal divided difference.

r_k/u_k- the relative residual which represents the time accuracy.

e_k/u_k- the relative error which represents the space + time accuracy.

For $N=8$ D is

$$D=\{x|-160\le x\le -4\} \tag{5.20}$$

and the results are given in Table 4.

Table 4. $(N=8)$

k	$\lvert a_k\rvert$	r_k/u_k	e_k/u_k	k	$\lvert a_k\rvert$	r_k/u_k	e_k/u_k
2	2.45−01	7.79+00	7.01+00	12	2.73−03	1.85−02	8.30−02
4	1.79−01	5.42+00	3.27+00	14	2.25−05	1.92−03	8.23−02
6	8.29−02	2.32+00	1.74+00	16	1.11−06	1.48−04	8.23−02
8	1.31−02	5.81−01	2.54−01	18	6.15−08	1.77−06	8.23−02
10	2.40−03	9.18−02	1.06−01	20	2.97−09	1.94−07	8.23−02

For $N=16$ D is

$$D=\{x|-640\le x\le -4\} \tag{5.21}$$

and the results are given in Table 5.

Table 5. $(N=16)$

k	$\lvert a_k\rvert$	r_k/u_k	e_k/u_k	k	$\lvert a_k\rvert$	r_k/u_k	e_k/u_k
3	1.67−01	2.44−01	9.19+00	18	9.50−04	3.28−02	9.76−02
6	2.88−02	1.67−01	2.64+00	21	6.32−05	1.80−03	2.28−02
9	2.29−02	5.19−01	3.59+00	24	1.64−05	3.05−03	2.29−02
12	4.08−02	1.35+00	2.36−01	27	5.92−06	3.71−04	2.26−02
15	6.50−03	3.05−01	2.46−01	30	2.56−07	4.96−05	2.26−02

Observing Tables 4 and 5 we see that for $N=8$ we need 14 iterations to recover the space accuracy and for $N=16$ we need 27 iterations. This result compares favorably with a standard explicit algorithm like Forward Euler or Runge-Kutta where the number of iterations(time steps) is multiplied by 4 (due to stability limitations) whenever we increase the space resolution by two. In [Tale85] we describe an algorithm which is spectral in time. It is shown there that for a parabolic problem, the time resolution parameter depends almost linearly on the space resolution parameter. Since the algorithm used in the present paper can be considered also as spectral in time, it explains this improvement in the number of matrix vector multiplications needed to march the solution to a given time level.

References

[AbSt72] M. Abramowitz, I. A. Stegun, *Handbook of Mathematical Functions*, Dover Publications, Inc., New-York (1972).

[AnGo72] R. S. Anderssen, G. H. Golub, "Richardson's non-stationary matrix iterative procedure," Rep. STAN-CS-72-304, Computer Science Dept., Stanford University, Stanford, CA (1972).

[CodB82] S. D. Conte, C. de Boor, *Elementary numerical analysis*, McGraw-Hill International Student Edition, 1982.

[Davi75] P. J. Davis, *Interpolation and Approximation*, Dover Publication, Inc., New York, 1975.

[FiRe87] B. Fischer, L. Reichel, "A stable Richardson iteration method for complex linear systems," Preprint 10, Institute fur Angewandte Mathematik der Universitat Hamburg, 1987.

[HaYo81] L. Hayeman, D. Young, *Applied Iterative Methods*, Academic Press, New York, 1981.

[LeFi76] V. I. Lebedev, S. A. Finogenov, "Utilization of ordered Chebyshev parameters in iterative methods," U.S.S.R. Comput. Math. Phys., 16, No. 4, pp. 70-83, (1976).

[Mark77] A. I. Markushevich, *Theory of Functions of a Complex Variable*, Chelsea, New York (1977).

[MoVa78] C. B. Moler, C. F. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix," SIAM Review, 20, pp. 801-836 (1978).

[SmLe68] V. I. Smirnov, N. A. Lebedov, *Functions of a Complex Variable*, London ILIFFE Books Ltd. (1968).

[Tale86] H. Tal-Ezer, "Spectral methods in time for hyperbolic equations," SIAM J. Numer. Anal., 23, No. 1 (1986), pp. 11-26.

[Tale85] H. Tal-Ezer, "Spectral methods in time for parabolic problems," ICASE Report No. 85-9, Langley Research Center, Hampton, VA (1985).

[Tale87] H. Tal-Ezer, "Polynomial approximation of functions of matrices and its application to the solution of a general system of linear equations," ICASE Report No. 87-63, NASA Langley Research Center, Hampton, VA (1987).

[Tref80] L. N. Trefethen, "Numerical computation of the Schwarz-Christoffel transformation," SIAM J. Sci. Stat. Comput. 1 (1980), pp. 82-102.

[Wals56] J. L. Walsh, "Interpolation and approximation by rational functions in the complex domain," American Mathematical Society, Providence, Rhode Island, 1956.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA CR-181677 ICASE Report No. 88-39	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle HIGH DEGREE INTERPOLATION POLYNOMIAL IN NEWTON FORM		5. Report Date June 1988
		6. Performing Organization Code
7. Author(s) Hillel Tal-Ezer		8. Performing Organization Report No. 88-39
		10. Work Unit No. 505-90-21-01
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225		11. Contract or Grant No. NAS1-18107
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Richard W. Barnwell

Submitted to SIAM J. Sci. Comput.

Final Report

16. Abstract

Polynomial interpolation is an essential subject in numerical analysis. Dealing with a real interval, it is well-known that even if f(x) is an analytic function, interpolating at equally spaced points can diverge [Davi75]. On the other hand, interpolating at the zeroes of the corresponding Chebyshev polynomial will converge. Using the Newton formula, this result of convergence is true only on the theoretical level. It is shown that the algorithm which computes the divided differences is numerically stable only if: 1.) the interpolating points are arranged in a certain order, 2.) the size of the interval is 4.

17. Key Words (Suggested by Author(s)) interpolation, Newton form, divided differences	18. Distribution Statement 64 - Numerical Analysis Unclassified - unlimited

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 23	22. Price A02

End of Document