Scientific Computation II HW2

Michael Nameika

February 2023

Section 5 Problems

3. Let $E_N = \inf_p \|p(x) - e^x\|_\infty$, where $\|f\|_\infty = \sup_{x\in[-1,1]} |f(x)|$, denote the error in degree N minimax polynomial approximation to e^x on $[-1, 1]$. (a) One candidate approximation $p(x)$ would be the Taylor series truncated at degree N. From this approximation, derive the bound $E_N < ((N + 2)/(N + 1))/(N + 1)!$ for $N \geq 0$. (b) In fact, the truncated Taylor series falls short of optimal by a factor of about 2^N, for it is known (see equation (6.75) of [Mei67]) that as $N \to \infty$, $E_N \sim 2^{-N}/(N + 1)!$. Modify Program 9 to produce a plot showing this asymptotic formula, the upper bound of (a), the error when $p(x)$ is obtained from interpolation in Chebyshev points, and the same for equispaced points, all as a function of N for $N = 1, 2, 3, \ldots, 12$. Comment on the results.

(a) Recall that

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Let $p(x)$ be the N^{th} degree Taylor polynomial expansion of e^x. Then

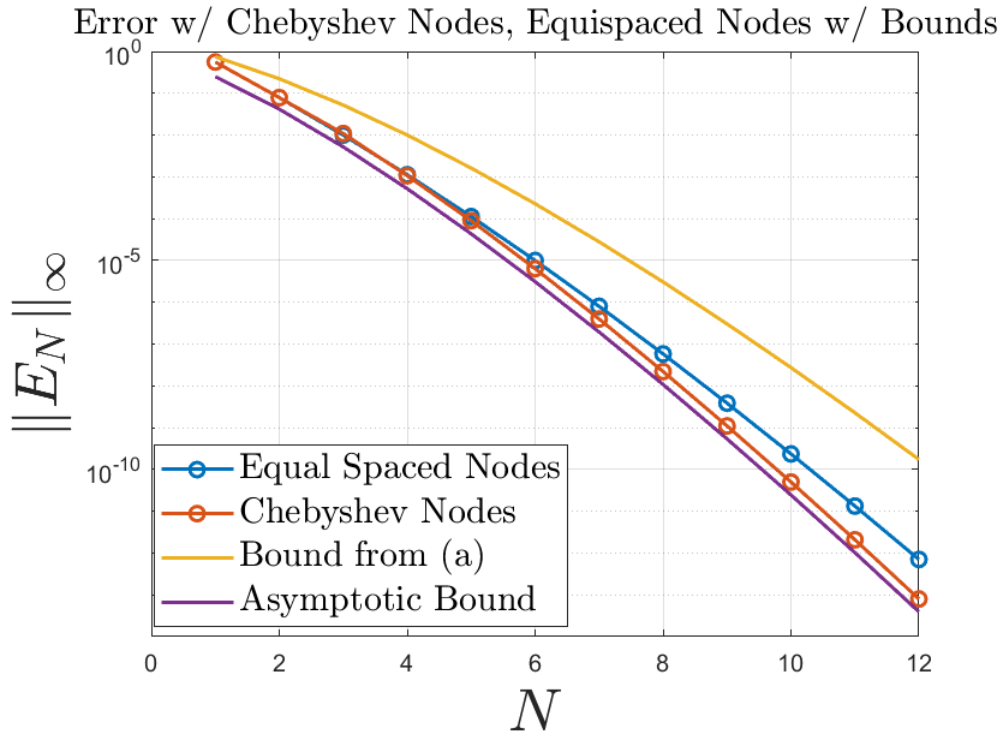
$$\begin{aligned}
\|p(x) - e^x\|_\infty &= \sup_{x\in[-1,1]} \left| \sum_{k=N+1}^{\infty} \frac{x^k}{k!} \right| \\
&= \sup_{x\in[-1,1]} \left| \frac{1}{(N+1)!} + \frac{1}{(N+2)!} + \ldots \right| \\
&= \frac{1}{(N+1)!} + \frac{1}{(N+2)!} + \cdots \\
&= \frac{1}{(N+1)!} \left(1 + \frac{1}{N+2} + \frac{1}{(N+3)(N+2)} + \ldots \right) \\
&< \frac{1}{(N+1)!} \left(\sum_{k=0}^{\infty} \frac{1}{(N+2)^k} \right) \\
&= \frac{1}{(N+1)!} \left(\frac{1}{1 - \frac{1}{N+2}} \right) \\
&= \frac{1}{(N+1)!} \left(\frac{N+2}{N+1} \right) \\
&= \frac{\frac{N+2}{N+1}}{(N+1)!}
\end{aligned}$$

So we have

$$\|p(x) - e^x\|_\infty < ((N + 2)/(N + 1))/(N + 1)!$$

Which is what we sought to show.

(b) Modifying program 9 to use the function e^x and plotting the error from the interpolation on equispaced nodes and Chebyshev nodes, we find the following semilog plot:



Notice that the error from the Chebyshev nodes appears to follow closely to the asymptotic result as N increases, as we would expect. Further notice how the error from the equispaced points appear to follow the trend of the bound we found in part (a).

Section 6 Problems

1. If $x_0, x_1, \ldots, x_N \in \mathbb{R}$ are distinct, then the *cardinal function* $p_j(x)$ defined by

$$p_j(x) = \frac{1}{a_j} \prod_{\substack{k=0 \\ k \neq j}}^{N} (x - x_k), \qquad a_j = \prod_{\substack{k=0 \\ k \neq j}}^{N} (x_j - x_k)$$

is the unique polynomial interpolant of degree N to the values 1 at x_j and 0 at x_k, $k \neq j$. Take the logarithm and differentiate to obtain

$$p_j'(x) = p_j(x) \sum_{\substack{k=0 \\ k \neq j}}^{N} (x - x_k)^{-1}$$

and from this derive the formula

$$D_{ij} = \frac{1}{a_j} \prod_{\substack{k=0 \\ k \neq i,j}}^{N} (x_i - x_k) = \frac{a_i}{a_j (x_i - x_j)} \quad (i \neq j)$$

and

$$D_{jj} = \sum_{\substack{k=0 \\ k \neq j}}^{N} (x_j - x_k)^{-1}$$

Taking the logarithm of the equation for $p_j(x)$ above, we find

$$\begin{aligned}\ln\left(p_j(x)\right) &= \ln\left(\frac{1}{a_j}\sum_{\substack{k=0\\k\neq j}}^{N}(x-x_k)\right)\\ &= \ln\left(\frac{1}{a_j}\right)+\sum_{\substack{k=0\\k\neq j}}^{N}\ln\left(x-x_k\right)\end{aligned}$$

Differentiating, we find

$$\begin{aligned}\frac{1}{p_j(x)}p_j'(x) &= \sum_{\substack{k=0\\k\neq j}}^{N}(x-x_k)^{-1}\\ p_j'(x) &= p_j(x)\sum_{\substack{k=0\\k\neq j}}^{N}(x-x_k)^{-1}\end{aligned}$$

Plugging in x_i, $0 \leq i \leq N$, $i \neq j$, for x, we find

$$\begin{aligned}p_j'(x_i) &= p_j(x_i)\sum_{\substack{k=0\\k\neq j}}^{N}(x_i-x_k)\\ &= \frac{1}{a_j}\prod_{\substack{k=0\\k\neq j}}^{N}(x_i-x_k)\left(\sum_{\substack{k=0\\k\neq j}}^{N}(x_i-x_k)^{-1}\right)\\ &= \frac{a_i}{a_j(x_i-x_j)}\end{aligned}$$

(since all terms go to zero except for the $k=i$ term)
If $i=j$, then

$$\begin{aligned}p_j'(x_j) &= \frac{1}{a_j}\prod_{\substack{k=0\\k\neq j}}^{N}(x_j-x_k)\left(\sum_{\substack{k=0\\k\neq j}}^{N}(x_j-x_k)^{-1}\right)\\ &= \frac{a_j}{a_j}\sum_{\substack{k=0\\k\neq j}}^{N}(x_j-x_k)^{-1}\\ &= \sum_{\substack{k=0\\k\neq j}}^{N}(x_j-x_k)^{-1}\end{aligned}$$

Finally, we have

$$\begin{aligned}D_{ij} &= \frac{a_i}{a_j(x_i-x_j)} \qquad (i\neq j)\\ D_{jj} &= \sum_{\substack{k=0\\k\neq j}}^{N}(x_j-x_k)^{-1}\end{aligned}$$

Which is what we sought to show.

8. Let D_N be the usual Chebyshev differentiation matrix. Show that the power $(D_N)^{N+1}$ is identically equal to zero. Now try it on the computer for $N = 5$ and 20 and report the compupted 2-norms $\|(D_5)^6\|_2$ and $\|(D_{20})^{21}\|_2$. Discuss.

Proof: Recall that the rows of the Chebyshev differentiation matrix are given by differentiating the Nth degree interpolating polynomial $p(x)$ and inputting the Chebyshev nodes. Thus, since $p(x)$ is an Nth degree polynomial,

$$\frac{d^{N+1}}{dx^{N+1}}p(x) = 0$$

That is, each element of the $(N+1)$th differentiation matrix is identically equal to zero.

Using my chebDiffMat.m file, we find the following for D_5, $(D_5)^6$, and $\|(D_5)^6\|_2$:

```
>> [diffMat] = chebDiffMat(5)

diffMat =

    8.5000  -10.4721    2.8944   -1.5279    1.1056   -0.5000
    2.6180   -1.1708   -2.0000    0.8944   -0.6180    0.2764
   -0.7236    2.0000   -0.1708   -1.6180    0.8944   -0.3820
    0.3820   -0.8944    1.6180    0.1708   -2.0000    0.7236
   -0.2764    0.6180   -0.8944    2.0000    1.1708   -2.6180
    0.5000   -1.1056    1.5279   -2.8944   10.4721   -8.5000
```

```
ans =

   1.0e-09 *

   -0.0564    0.0837   -0.0537    0.0346   -0.0241    0.0084
   -0.0255    0.0346   -0.0214    0.0140   -0.0086    0.0018
   -0.0014   -0.0018    0.0009   -0.0014    0.0039   -0.0019
    0.0017   -0.0034    0.0028   -0.0032   -0.0004    0.0034
   -0.0100    0.0255   -0.0318    0.0468   -0.0791    0.0510
   -0.0209    0.0500   -0.0632    0.0900   -0.1328    0.0820
```

```
>> norm(ans, 2)

ans =

   2.4631e-10
```

So we can see that $(D_5)^6$ is essentially the zero matrix. However, for D_{20}, we find the following for $\|(D_{20})^{21}\|_2$:

```
>> norm(ans)

ans =

   4.7884e+21
```

This tells us that $(D_{20})^{21}$ is not even close to the zero matrix, when we expect it should be! This is likely due to machine round off error compounding with each successive power of D_{20}.

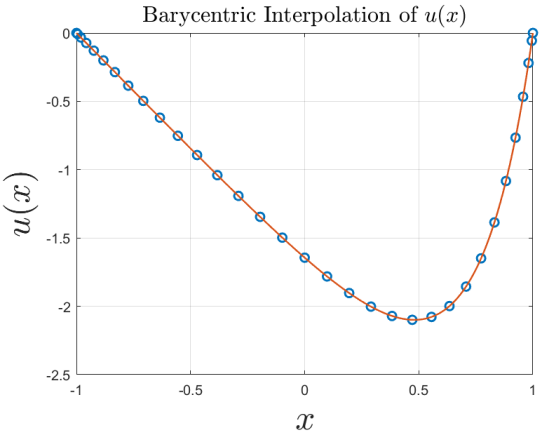
Section 7 Problems

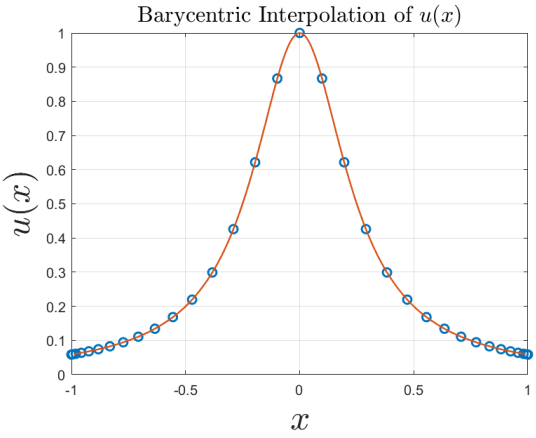
1. Modify Program 13 so that instead of polyval and polyfit, it uses the more stable formula of *barycentric interpolation* [Hen82]:

$$p(x) = \sum_{j=0}^{N} \frac{a_j^{-1} u_j}{(x - x_j)} \Bigg/ \sum_{j=0}^{N} \frac{a_j^{-1}}{x - x_j}$$

where $\{a_j\}$ are defined by (6.7). Experiment with various interpolation problems (such as that of Exercise 5.1) and find evidence of the enhanced stability of this method.

Implementing the barycentric interpolation formula (see the attached myProb13Script.m file), we find the following plots for the Runge function $f(x) = 1/(1+16x^2)$ and the solution to the differential equation $u_{xx} = e^{4x}$, $u(\pm 1) = 0$:

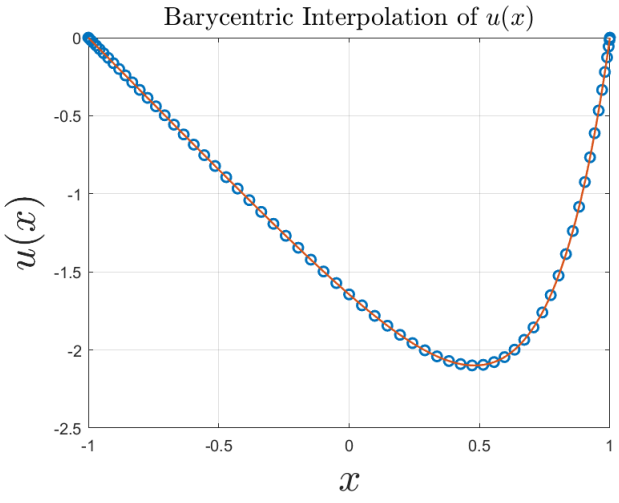




Increasing the number of interpolation points to $N = 64$, we find the following issue with the "normal" interpolation

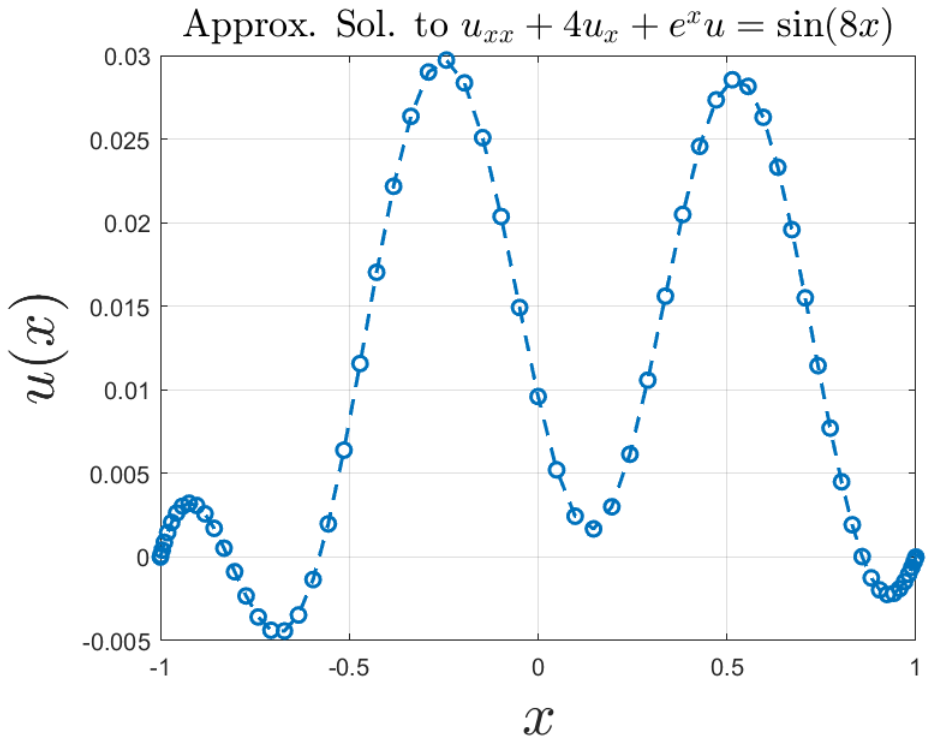
```
>> p13
Warning: Polynomial is badly conditioned. Add points
with distinct X values, reduce the degree of the
polynomial, or try centering and scaling as
described in HELP POLYFIT.
> In polyfit (line 84)
In p13 (line 13)
```

Whereas the barycentric interpolation runs no problem:



2. Solve the boundary value problem $u_{xx} + 4u_x + e^x u = \sin(8x)$ numerically on $[-1, 1]$ with boundary conditions $u(\pm 1) = 0$. To 10 digits of accuracy, what is $u(0)$?

Implementing this ode in MATLAB using Chebyshev differentiation, we find the following solution plot:



And we find the following approximate value for $u(0)$:

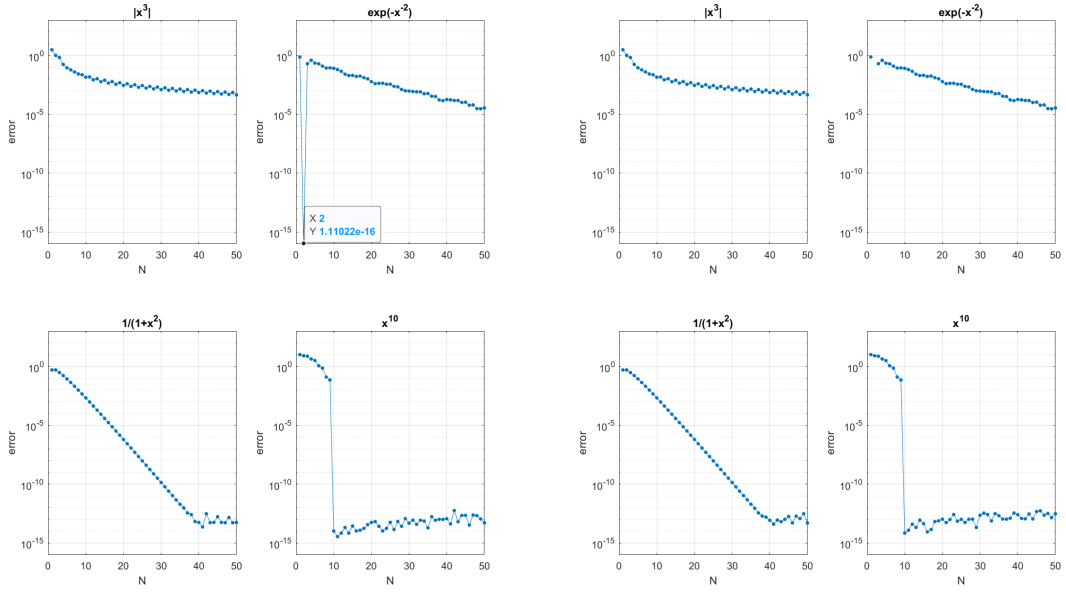
```
ans =

   0.009597857229504
```

For this problem, we used $N = 64$. See the attached probScript.m for additional details.

Section 8 Problems

3. Modify Program 12 (p. 57) to make use of chebfft instead of cheb. The results should be the same as in Output 12, except for rounding errors. Are the effects of rounding errors smaller or larger than before?

Modifying Program 12 to use the chebfft file rather than cheb, we find the following plots:



Above: Left figure shows Program 12 using cheb and the right figure shows Program 12 using chebfft. Notice the extreme similarities between the two plots. We can see that the rounding errors are approximately the same between the two figures, with the chebfft differentiation having smaller rounding errors and larger rounding errors when compared to the cheb differentiation matrix.

5. Find a way to modify your program of Exercise 8.4, as in Exercise 3.8 but now for a second-order problem, to make use of matrix exponentials rather than time discretization. What effect does this have on the computation time?

Starting with the two dimensional wave equation with $c = 1$,

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

and reducing to a system of first order equations in time, we have

$$\frac{d}{dt}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 0 & I \\ D & 0 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Where D is an $(N+1)^2$ by $(N+1)^2$ differentiation matrix, 0 is the zero matrix with the same dimension, I is the identity matrix of the same dimension, and $v = u_t$.
Let $\mathcal{L} = \begin{bmatrix} 0 & I \\ D & 0 \end{bmatrix}$ and multiply the above equation by $e^{\mathcal{L}t}$, we get

$$e^{\mathcal{L}t}\frac{d}{dt}\begin{bmatrix} u \\ v \end{bmatrix} - \mathcal{L}e^{\mathcal{L}t}\begin{bmatrix} u \\ v \end{bmatrix} = 0$$

which becomes

$$\frac{d}{dt}\left(e^{\mathcal{L}t}\begin{bmatrix} u \\ v \end{bmatrix}\right) = 0$$

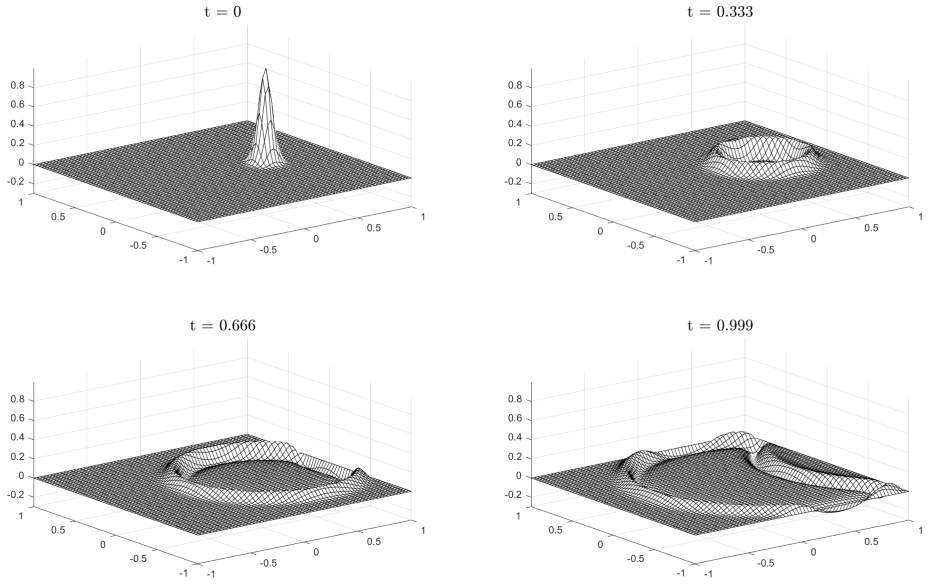
which gives us

$$e^{\mathcal{L}t}\begin{bmatrix} u \\ v \end{bmatrix} = C$$

and for $t = 0$, we find $C = \begin{bmatrix} u(x,y,0) \\ v(x,y,0) \end{bmatrix}$ Then the general solution takes the following form:

$$\begin{bmatrix} u \\ v \end{bmatrix} = e^{-\mathcal{L}t} \begin{bmatrix} u \\ v \end{bmatrix}$$

Implementing this into MATLAB, we find the following plot for $N = 64$:



Running it for $N = 64$ took nearly seven minutes, so this had a massive impact on the computation time!