APPROXIMATION THEORY HOMEWORK 3

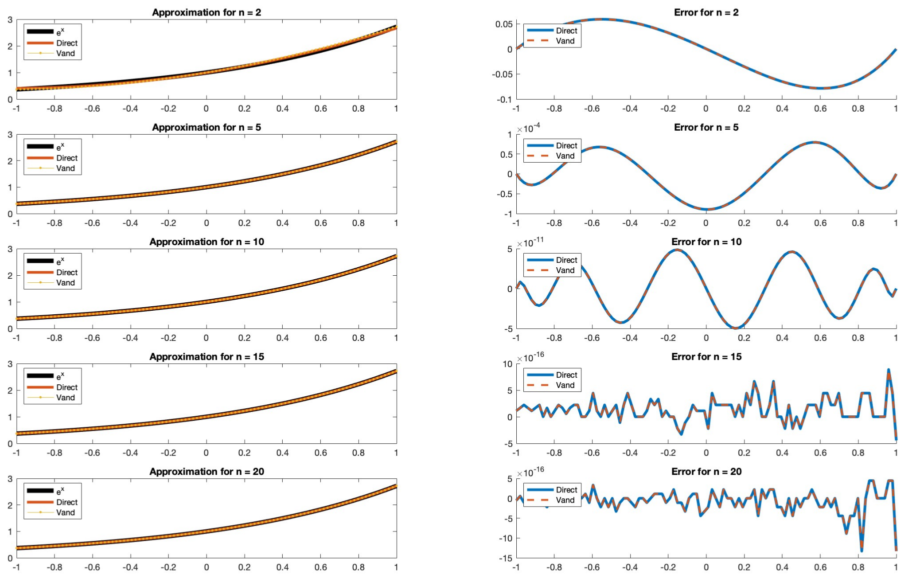
TOMMENIX YU
ID: 12370130
STAT 31220
DUE WED APR 12TH, 2023, 11PM

Discussed with classmates.

Exercise 1.

Proof.

(a): So we just plot them (generated by the file "TommenixYu_q1a"):



(b):

We know it's an ellipse with radius ρ and foci at ± 1, so it's "longer radius" (I don't know how to call it properly) is ρ and the "shorter radius" is $\sqrt{\rho^2 - 1}$ and hence the expression would be

$$\frac{\mathrm{Re}(x)^2}{\rho^2} + \frac{\mathrm{Im}(x)^2}{\rho^2 - 1} = 1.$$

(c):

Using the Bernstein bound we have

$$||f - f_n|| \lesssim \frac{2e^{\rho}\rho^{-n}}{\rho - 1}$$

where the $\sim$ part we just neglect the $e^{-\rho}$ term. Doesn't affect order at all so just for easier calculation. And since e^x is holomorphic we know that we can choose any ρ. By taking derivatives we get

$$\frac{d}{d\rho}\frac{2e^{\rho}\rho^{-n}}{\rho - 1} = \frac{\rho^{-n-1}e^{\rho}(\rho^2 - (n+2)\rho + n)}{(\rho - 1)^2}$$

for which we set to 0 and get $\rho^2 + (n-2)\rho - n = 0$ so the only > 1 root is

$$\rho = \frac{2 + n + \sqrt{n^2 + 4}}{2}$$

and plugging in the bound is

$$||f - f_n|| \leq \frac{2e^{\frac{2+n+\sqrt{n^2+4}}{2}}\left(\frac{2+n+\sqrt{n^2+4}}{2}\right)^{-n}}{\frac{2+n+\sqrt{n^2+4}}{2} - 1}.$$

(d):

For $n = 1$, we have

$$T_1 = \frac{1}{2}(-1)^0\frac{(1-0-1)!}{0!(1-0)!}(2x)^{1-2*0} = x = cos(\arccos(x)).$$

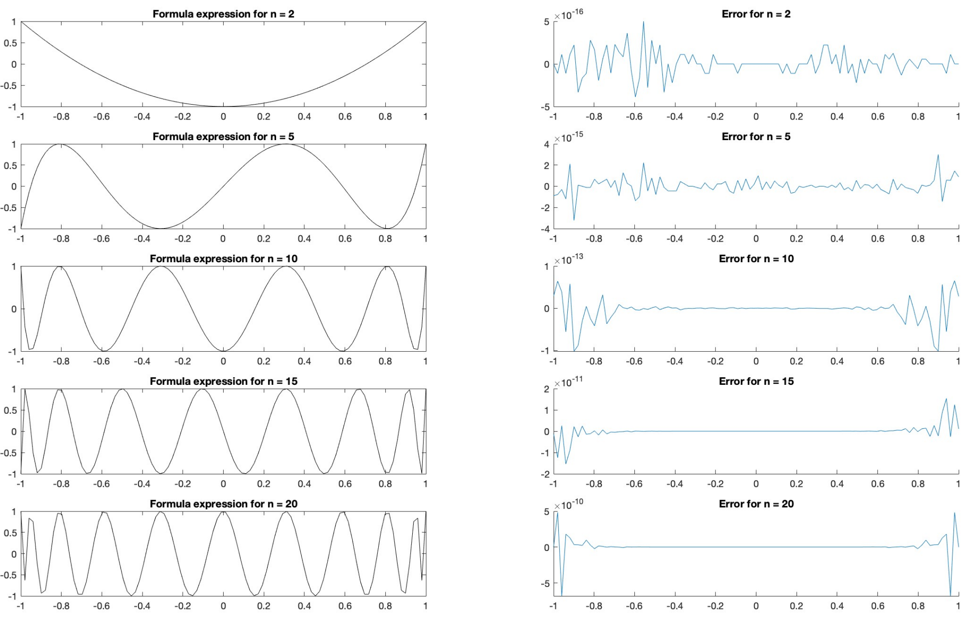
For $n = 2$ we have

$$T_2 = 1\left((-1)^0\frac{(2-0-1)!}{0!(2-0)!}(2x)^2 + (-1)^1\frac{(2-1-1)!}{1!(2-2)!}(2x)^0\right) = 2x^2 - 1 = cos(2\arccos(x)).$$

So indeed it holds. For larger terms an induction with some careful calculation will show the result.

(e):

The plot I had is the following (generated by the file "TommenixYu_q1e"):



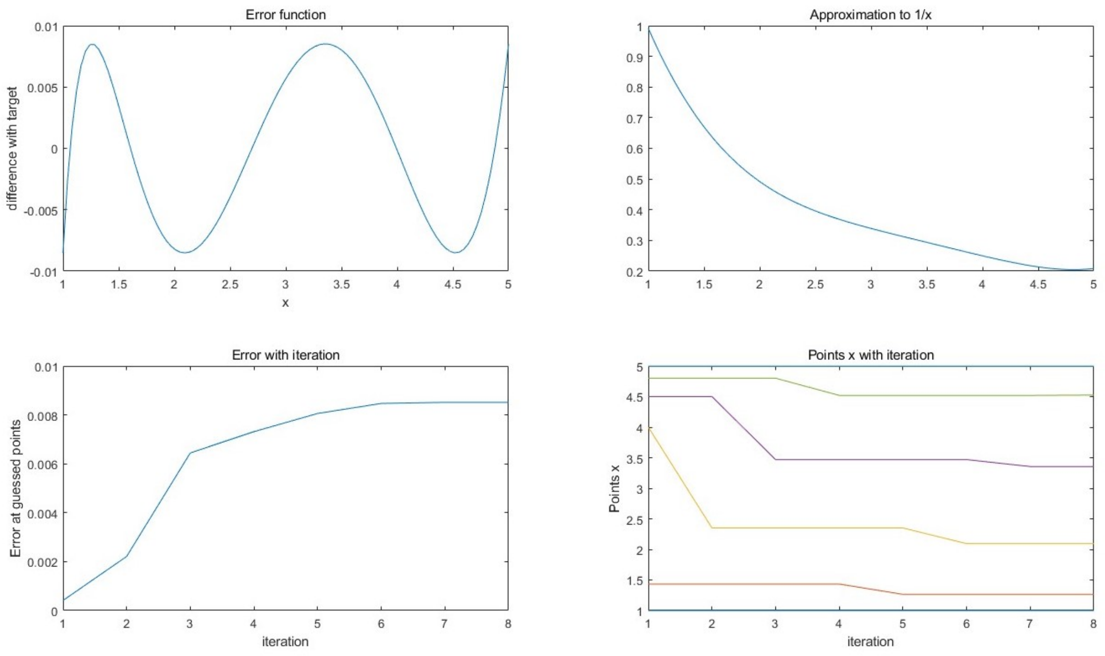
and we see that the error is tremendously larger for more points, at the end points. It's not really surprising because both Runge phenomenon and that machine precision is already capped but then there's always the increment of $\log n$ order as n goes up (of course if machine error is not capped, then this is way slower than convergence rate).

$\square$

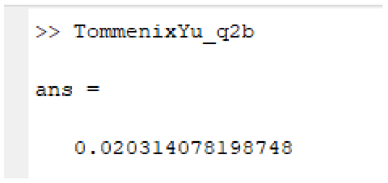
Exercise 2.

Proof.

(a): Adapted (generated by the file "TommenixYu_q2a"):



(b): Plugging in the matrix polynomial and the result of inverse is (with seed 1) (generated by the file "TommenixYu_q2b"):



```
>> TommenixYu_q2b

ans =

   0.020314078198748
```

This makes sense because we just do EVD and see that (norm of b is computed, like 3.8 or so)

$$||Pf(A)|| = ||QPf(\Lambda)Q|| \cdot ||b|| \approx 3 * ||Pf(\Lambda)|| = 0.03$$

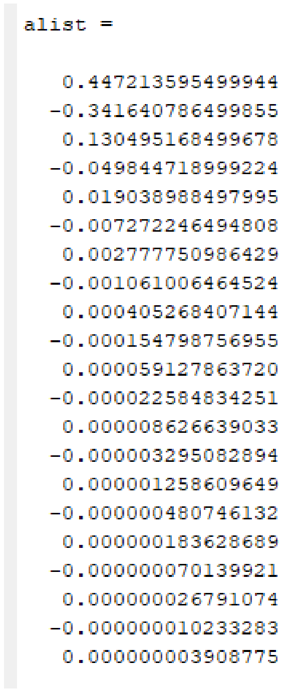
and this error makes very good illustration of that.

(c): Shifting $\frac{1}{x}$ to $\frac{1}{2x+3}$ does the job here.

So what we have should be

$$a_k = \frac{2}{\pi}\int_{-1}^{1}\frac{\frac{1}{2x+3}T_k(x)}{\sqrt{1-x^2}}dx$$

for $k > 0$ and a_0 is one half of the above expression. The list is (generated by the file "TommenixYu_q2c"):



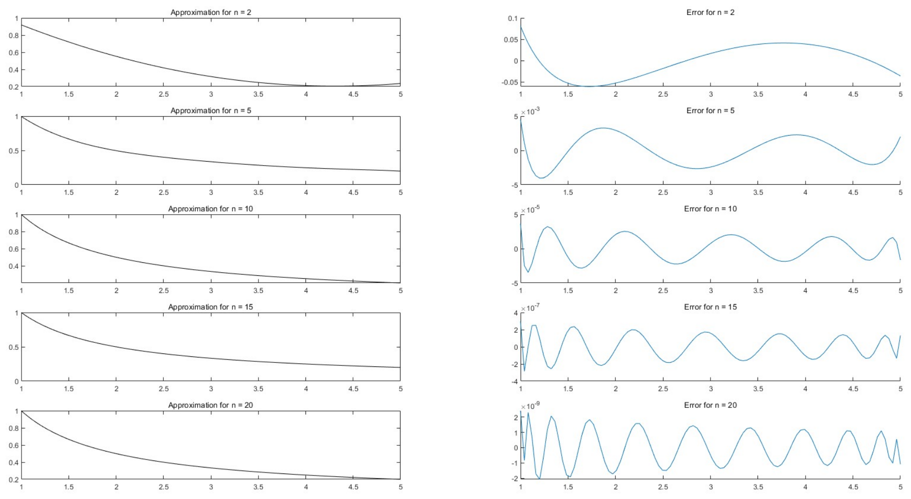
```
alist =

   0.447213595499944
  -0.341640786499855
   0.130495168499678
  -0.049844718999224
   0.019038988497995
  -0.007272246494808
   0.002777750986429
  -0.001061006464524
   0.000405268407144
  -0.000154798756955
   0.000059127863720
  -0.000022584834251
   0.000008626639033
  -0.000003295082894
   0.000001258609649
  -0.000000480746132
   0.000000183628689
  -0.000000070139921
   0.000000026791074
  -0.000000010233283
   0.000000003908775
```

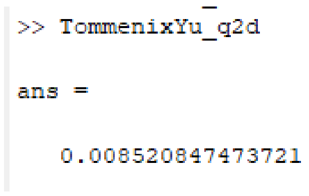
To use this list we have that on the interval $[1, 5]$

$$\frac{1}{x} \approx \sum_{k=0}^{n} a_k * \cos\left(k \arccos\left(\frac{x-3}{2}\right)\right)$$

and indeed the graph is correct (generated by the file "TommenixYu_q2c"):



(d): Plugging in the matrix polynomial and the result of inverse is (with seed 1) (generated by the file "TommenixYu_q2d"):



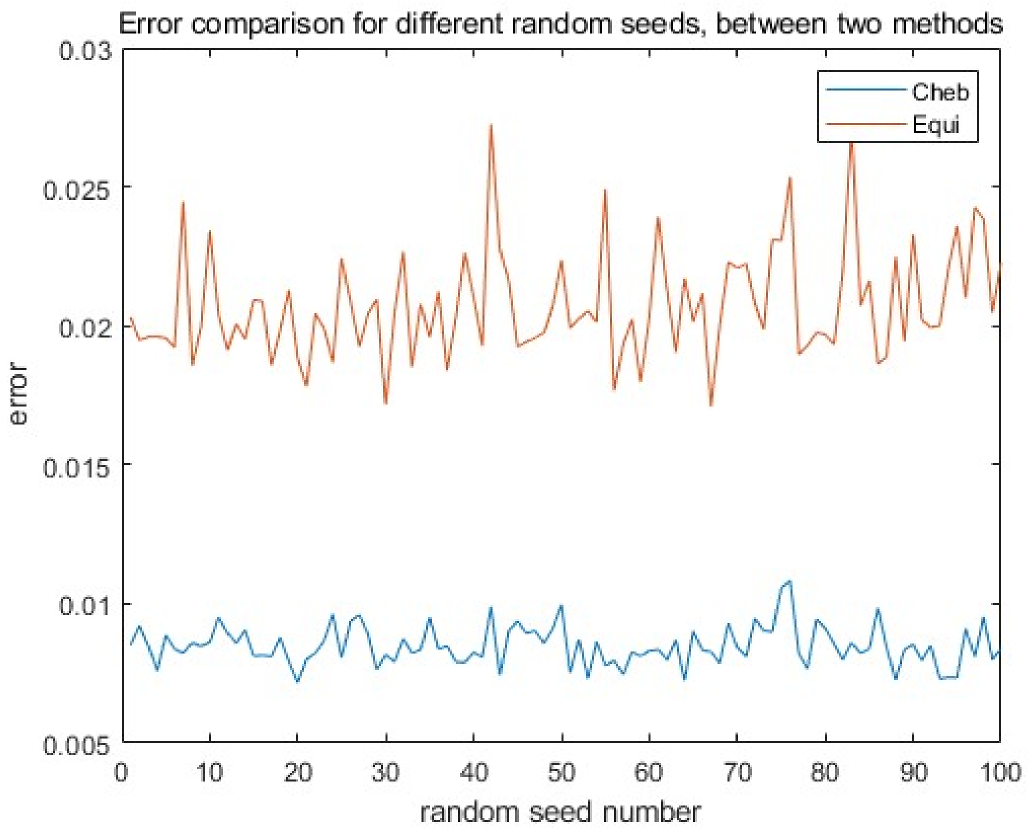
```
>> TommenixYu_q2d

ans =

   0.008520847473721
```

Here the error is better with seed 1.

I also did a test with random seeds from 1 to 100 and compared the error between the two methods, and it turns out that the Chebyshev method is better (generated by the file "TommenixYu_q2dcompare") (this might take a while):



And this very much sense. Explanation for error here is that I used only up till the 5th Chebyshev to save time running this code. And from last question we see that the norm is only about 0.005 and thus the same norm of b analysis yields the result. □

Exercise 3.

Proof.

(a):

By computation we have

$$\left[-\frac{\gamma_n}{\alpha_n}, -\frac{\beta_n}{\alpha_n}, \frac{1}{\alpha_n}\right] \cdot \begin{bmatrix} q_{n-1}(x) \\ q_n(x) \\ q_{n+1} = q_n(x)(\alpha_n x + \beta_n) - \gamma_n q_{n-1}(x) \end{bmatrix} = x q_{n+1}(x)$$

thus the matrix we want is

$$A = \begin{bmatrix} -\frac{\beta_0}{\alpha_0} & \frac{1}{\alpha_0} & & & \\ -\frac{\gamma_1}{\alpha_1} & -\frac{\beta_1}{\alpha_1} & \frac{1}{\alpha_1} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\gamma_{n-2}}{\alpha_{n-2}} & -\frac{\beta_{n-2}}{\alpha_{n-2}} & \frac{1}{\alpha_{n-2}} \\ & & & -\frac{\gamma_{n-1}}{\alpha_{n-1}} & -\frac{\beta_{n-1}}{\alpha_{n-2}} \end{bmatrix}$$

where indeed the last term matches up.

It's full rank so it should be unique.

(b):

($\Leftarrow$:) Plugging in yields:

$$p(x_*) = \sum_{i=0}^{n-1} c_i q_i(x_*) - \frac{c_n}{c_n} \sum_{i=0}^{n-1} c_i q_i(x_*) = 0.$$

($\Rightarrow:$) Just writing things out we get

$$p(x_*) = \sum_{i=0}^{n-1} c_i q_i(x_*) + c_n q_n(x_*) = 0$$

$$\Rightarrow q_n(x_*) = -\frac{1}{c_n} \sum_{i=0}^{n-1} c_i q_i(x_*) = 0.$$

(c):

It must be a typo and the roots should be the eigenvalues of C not L.

($p(\lambda) = 0 \Rightarrow Cx = \lambda x$ for some x:)

We just pick $x = v$ in (a) then we get

$$Cv = Mv + Lv = \begin{bmatrix} \lambda q_0(\lambda) \\ \cdots \\ \lambda q_{n-2}(\lambda) \\ \lambda q_{n-1}(\lambda) - \frac{q_n(x)}{\alpha_n} + \frac{1}{\alpha_n}\left[-\frac{1}{c_n}\sum_{i=0}^{n-1} c_i q_i(x_*)\right] \end{bmatrix}$$

and by (b) we have

$$\lambda q_{n-1}(\lambda) - \frac{q_n(x)}{\alpha_n} + \frac{1}{\alpha_n}\left[-\frac{1}{c_n}\sum_{i=0}^{n-1} c_i q_i(x_*)\right] = \lambda q_{n-1}(\lambda)$$

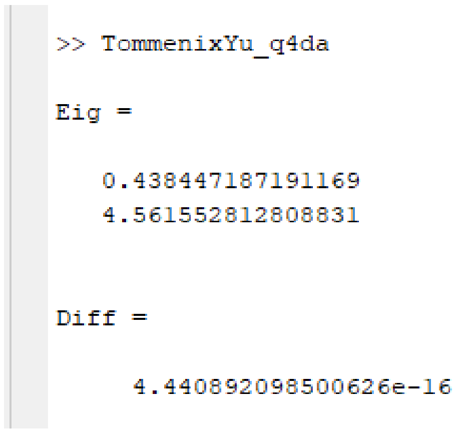
so indeed it is an eigenvalue.

($p(\lambda) = 0 \Leftarrow Cx = \lambda x$ for some x:)

We know already that p has distinct roots and thus we know already that those roots are eigenvalues of C. But p is a degree n polynomial and therefore we know n distinct eigenvalues of C, a n by n matrix. So we know the whole spectrum and there's no other eigenvalues.

(d):

a): For monomials $a_i = 1, b_i = 0, \gamma_i = 0$ and thus the matrix is just the upper off-diagonal of ones. The results are exactly the same (generated by the file "TommenixYu_q4da"):



```
>> TommenixYu_q4da

Eig =

   0.438447187191169
   4.561552812808831


Diff =

     4.440892098500626e-16
```

b): Here $a_i = 2, b_i = 0, \gamma_i = -1$ and $a_0 = 1$. Thus the matrix is upper off-diagonal of halves, and lower off diagonal of halves. We also need to change $M_{1,2} = 1$ of course. In our $n = 2$ case it's plainly

$$M = \begin{bmatrix} & 1 \\ \frac{1}{2} & \end{bmatrix}.$$

The results are exactly the same (generated by the file "TommenixYu_q4db"):

```
>> TommenixYu_q4db

Eig =

  -0.750000000000000
   0.500000000000000


Roots =

  -0.750000000000000
   0.500000000000000


Diff =

     0
```

□