HW4 Extrapolation, Padé, and Continued Fractions

Fei Ding

October 20, 2009

Problem1

Using the transformation from continued fraction to Padé, we can see that here $a_1 = 1, a_n = (n-1)^2 \ \ (n \neq 1); bn = 2n - 1$. So

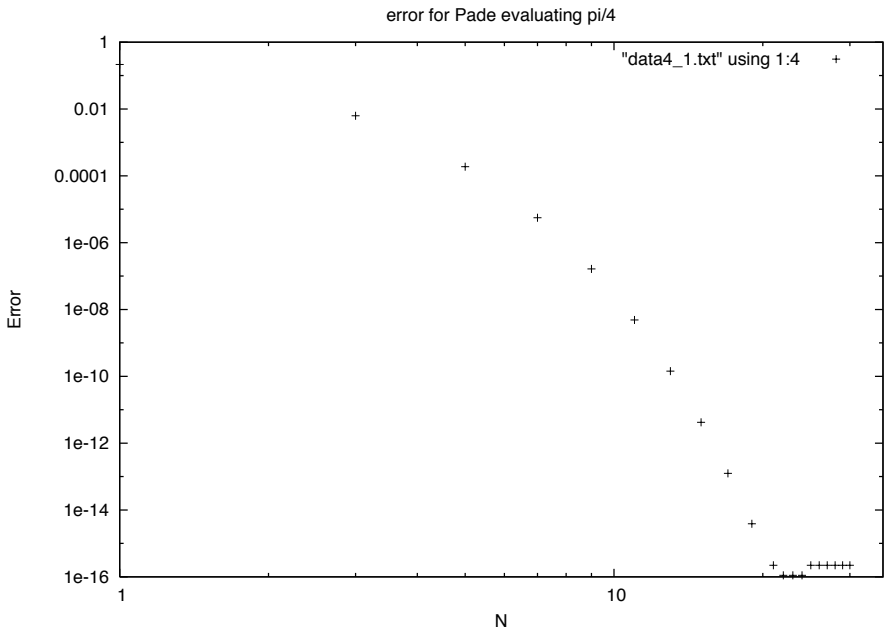
$$\begin{aligned} A_1 &= b_0 A_0 + a_0 A_{-1} = A_0 + A_{-1} \\ A_n &= b_n A_{n-1} + a_n A_{n-2} = (2n-1)A_{n-1} + (n-1)^2 A_{n-2} \quad (n \neq 1) \end{aligned}$$

$$\begin{aligned} B_1 &= b_0 B_0 + a_0 B_{-1} = B_0 + B_{-1} \\ B_n &= b_n B_{n-1} + a_n B_{n-2} = (2n-1)B_{n-1} + (n-1)^2 B_{n-2} \quad (n \neq 1) \end{aligned}$$

The code:

```
program quarterPi
integer::N
double precision:: A,A_1,A_2,B,B_1,B_2,pi,Pade,error
A_1=0d0
A_2=1d0
B_1=1d0
B_2=0d0
pi = 4.d0*datan(1.d0)
do N=1, 30
   if (N==1) then
      A=A_1+A_2
      B=B_1+B_2
   else
      A=(2*N-1)*A_1+(N-1)**2*A_2
      B=(2*N-1)*B_1+(N-1)**2*B_2
   end if
   A_2=A_1
   A_1=A
   B_2=B_1
   B_1=B
   Pade=A/B
```

```
   Error=Pade-pi/4.d0
   write(1,*) N,Pade*4.d0, pi, Error
end do
end program quarterPi
```



The plot shows that while N increases, the error decreases. As N reaches 30, the error almost hits the machine accuracy. So the error kind of reaches some degree and doesn't decrease anymore.

Problem2

i. expanding ψ in Chebyshev polynomials of the second kind.

$$\psi(x) = \sum_{n=0}^{\infty} a_n U_n(x)$$

where $U_n(x)$ are the second kind Chebyshev polynomials satisfying properties as following,

$$U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x)$$

$$\int_{-1}^{1} \sqrt{1-x^2} U_n U_m \, dx = \frac{\pi}{2} \delta_{nm}$$

$$(1-x^2)U_n'' - 3xU_n' + n(n+2)U_n = 0.$$

substitute the expansion of ψ into the eigenvalue problem,

$$(x^2-1)\psi'' + 3x\psi' + \beta^2(2-2x-\frac{aE}{\beta})\psi = 0$$

we get (set $a = \beta = 1$)

$$\sum_{n=0}^{\infty} a_n[(x^2-1)U_n'' + 3xU_n' + (2-2x-E)U_n] = 0$$

, apply the third property of $U_n, (1-x^2)U_n'' - 3xU_n' + n(n+2)U_n = 0$, we get

$$\sum_{n=0}^{\infty} a_n[n(n+2) + 2 - 2x - E]U_n = 0$$

apply the first property of $U_n, U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x)$, we get

$$a_0(2-2x-E)U_0 + \sum_{n=1}^{\infty} a_n[(n^2+2n+2-E)U_n - U_{n-1}(x) - U_{n+1}(x)] = 0$$

Since we know the detailed form of U_0 and U_1, $U_0(x) = 1, U_1(x) = 2x$, the first term can be rewritten as $a_0[(2-E)U_0 - U_1]$, then apply the second property of $U_n, \int_{-1}^{1} \sqrt{1-x^2} U_n U_m \, dx = \frac{\pi}{2}\delta_{nm}$, that is multiplying the equation by $U_m\sqrt{1-x^2}$ and integrate from -1 to 1, we get a set of equations of coefficients a_m,

$$\begin{aligned} (2-E)a_0 - a_1 = 0, \quad m = 0 \\ (m^2+2m+2-E)a_m - a_{m-1} - a_{m+1} = 0, \quad m \geq 1 \end{aligned}$$

ii. establishing a tri-diagonal form for the ode Rearrange the equations and write the index as n instead of m, we get,

$$\alpha_0 a_0 + \beta_0 a_1 = E a_0$$

$$\vdots$$

$$\beta_{n-1} a_{n-1} + \alpha_n a_n + \beta_n a_{n+1} = E a_n$$

$$\cdots$$

where $\alpha_n = n^2 + 2n + 2, \beta_n = -1$.

So we have the equation $\hat{H}\vec{x} = E\vec{x}$, which can be written as the tri-diagonal form for ode if we cut off the polynomial at the Nth term,

$$\begin{pmatrix} 2 & -1 & 0 & 0 & \cdots\cdots & 0 \\ -1 & 5 & -1 & 0 & \cdots\cdots & 0 \\ 0 & -1 & 10 & -1 & \cdots\cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & \alpha_{N-1} & -1 \\ 0 & 0 & \cdots & 0 & -1 & \alpha_N \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N-1} \\ a_N \end{pmatrix} = E \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N-1} \\ a_N \end{pmatrix}$$

iii. converting this to a continued fraction, from the equation $\beta_{n-1}a_{n-1} + \alpha_n a_n + \beta_n a_{n+1} = Ea_n$, divided by a_n get,

$$\beta_{n-1}\frac{a_{n-1}}{a_n} + \alpha_n + \beta_n \frac{a_{n+1}}{a_n} = E,$$

so

$$\frac{a_n}{a_{n-1}} = \frac{\beta_{n-1}}{E - \alpha_n - \beta_n \frac{a_{n+1}}{a_n}},$$

apply this equation from n=1 and iterate,

$$\frac{a_1}{a_0} = \frac{\beta_0}{E - \alpha_1 - \beta_1 \frac{a_2}{a_1}} = \frac{\beta_0}{E - \alpha_1 - \frac{\beta_1^2}{E - \alpha_2 - \frac{\beta_2^2}{E - \alpha_3 - \cdots}}},$$

from the very first equation $\alpha_0 a_0 + \beta_0 a_1 = Ea_0$, divided by a_0 get,

$$\frac{a_1}{a_0} = \frac{E - \alpha_0}{\beta_0},$$

compare these two equation, we get a equation for E,

$$E = \alpha_0 + \frac{\beta_0^2}{E - \alpha_1 - \frac{\beta_1^2}{E - \alpha_2 - \frac{\beta_2^2}{E - \alpha_3 - \cdots}}}$$

$$= 2 + \frac{1}{E - 5 - \frac{1}{E - 10 - \frac{1}{E - 17 - \cdots}}}$$

iv. solving this using Padé approximants and bisection root finding. The continued fraction $\frac{1}{E-5-\frac{1}{E-10-\frac{1}{E-17-\cdots}}}$ has coefficients $a_1 = 1, a_n = -1 \ (n \neq 1); b_n = E - (n^2 + 2n + 2)$. convert this continued fraction to Padé and let $f(E) = 2 + \frac{1}{E-5-\frac{1}{E-10-\frac{1}{E-17-\cdots}}} - E$, using bisection root finding to find the lowest 8 roots of $f(E)$ for large N.

The code:

```
program eigenvalue
IMPLICIT NONE
double precision:: a,b,c,d,root
double precision, EXTERNAL:: pade,pA,pB
integer::cond,kl,kr
write (*,*) 'Input the range for root finding A & B:'
read (*,*) A,B
call BISEC(pade,A,B,1d-7,c,d,cond,kl,kr)
print *, cond
if (cond==1) then
   print *, 'Wrong range!'
else if (cond==2) then
   root=c
   print *, 'root=', root
else if (cond==3) then
   root=(a+b)/2.0
   print *, 'root=',root
else if (cond==4) then
   print *, 'Not continued!'
end if
end program

    SUBROUTINE BISEC(pade,A,B,Delta,C,D,Cond,KL,KR)
    double precision, PARAMETER:: Big=1d30
    INTEGER Cond,K,KL,KR,Max
    double precision A,B,C,D,Delta,YA,YB,YC
    double precision, EXTERNAL:: pade,pa,pb
    K=0
    KL=0
    KR=0
    YA=pade(pa,pb,A)
    YB=pade(pa,pb,B)
    D=B-A
    Cond=0
    Max=1+INT((dLOG(D)-dLOG(Delta))/dLOG(2d0))
    IF (YA*YB.GE.0) THEN
        Cond=1
        RETURN
    ENDIF
    DO WHILE ((Cond.EQ.0).AND.(K.LT.Max))
        C=(A+B)/2d0
        YC=pade(pa,pb,C)
            IF (YC.EQ.0d0) THEN
                A=C
                B=C
                Cond=2
```

```
            ELSEIF ((YB*YC).GT.0d0) THEN
                B=C
                YB=YC
                KR=KR+1
            ELSE
                A=C
                YA=YC
                KL=KL+1
            ENDIF
        K=K+1
        WRITE(1,*) k,a,c,b
    ENDDO
    D=B-A
    IF (D.LT.Delta) THEN
        IF (Cond.NE.2) Cond=3
        IF (dABS(YA-YB).GT.Big) Cond=4
    ENDIF
    RETURN
   END SUBROUTINE

double precision function pade(pA,pB,E)
IMPLICIT NONE
integer::n=100
double precision,external:: pA,pB
double precision::E
pade=pA(n,E)+pB(n,E)*(2d0-E)
return
end function pade

recursive double precision function pA(n,E) result(Avalue)
IMPLICIT NONE
integer:: n
double precision:: E, Avalue
if (n==0) then
  Avalue=0d0
else if (n==1) then
  Avalue=1d0
else
  Avalue=(E-(n**2+2*n+2))*pA(n-1,E)-pA(n-2,E)
end if
return
end function pA

recursive double precision function pB(n,E) result(Bvalue)
IMPLICIT NONE
integer:: n
```

```
double precision:: E, Bvalue
if (n==0) then
  Bvalue=1d0
else if (n==1) then
  Bvalue=E-5d0
else
  Bvalue=(E-(n**2+2*n+2))*pB(n-1,E)-pB(n-2,E)
end if
return
end function pB
```

The bisection subroutine is from

http://math.fullerton.edu/mathews/n2003/bisection/BisectionProg/Links/BisectionProg_lnk_5.ht

I used mathematica to plot $f(E)$ first so I knew the roots are very close to α_n, so for the first 8 α_n, they are 2, 5, 10, 17, 26, 37, 50. So basically I typed in 8 ranges by hand and got the lowest 8 eigenvalues:

```
 Input the range for root finding A & B:
1,2
 root=   1.6867202818393707
 Input the range for root finding A & B:
5,6
 root=   5.1130088269710541
 Input the range for root finding A & B:
10,11
 root=   10.057352870702744
 Input the range for root finding A & B:
17,18
 root=   17.031789809465408
 Input the range for root finding A & B:
26,27
 root=   26.020212918519974
 Input the range for root finding A & B:
37,38
 root=   37.013989597558975
 Input the range for root finding A & B:
50,51
 root=   50.010257810354233
```

I set N=10 because N=100 didn't work for me.

Problem3

Use the quotient-difference algorithm to convert the power series expression of $exp(-x) = sum_n c_n x^n$ to continued fraction, where $c_n = \frac{(-1)^n}{n!}$:

$$e_0^n = 0$$

$$
\begin{aligned}
q_1^n &= \frac{c_{n+1}}{c_n} \\
e_m^n &= q_m^{n+1} - q_m^n + e_{m-1}^{n+1} \\
q_{m+1}n &= \frac{e_{m+1}^n}{e_m^n} q_m^{n+1}
\end{aligned}
$$

I did this first by hand and found the rules, here are the results,

$$e_0^n = 0$$

$$q_1^n = -\frac{1}{n+1}$$

$$e_m^n = \frac{m}{(n+2m-1)(n+2m)}$$

$$q_m^n = -\frac{n+m-1}{(n+2m-2)(n+2m-1)}$$

so here $b_n = 1$, $a_1 = 1, a_n = q_{n/2}^0$ (n is even), $a_n = e_{(n-1)/2}^0$ (n is odd).

Then I tried two recursive function e and q to let them call each other to get the value of es and qs but failed so I used two arrays e and q to do the iteration and set another array to locate a_n. Here are the results of es, qs and a_n from the program, which do correspond to the results I got by hand.

```
 m          n
1           0  q= -1.00000000000000000
1           1  q= -0.50000000000000000
1           2  q= -0.33333333333333331
1           3  q= -0.25000000000000000
1           4  q= -0.20000000000000001
1           5  q= -0.16666666666666666
1           6  q= -0.14285714285714285
1           7  q= -0.12500000000000000
1           8  q= -0.11111111111111110
1           9  q= -0.10000000000000001
1          10  q= -9.09090909090909116E-002
1          11  q= -8.33333333333333287E-002
1          12  q= -7.69230769230769273E-002
1          13  q= -7.14285714285714246E-002
1          14  q= -6.66666666666666574E-002
1          15  q= -6.25000000000000000E-002
1          16  q= -5.88235294117647051E-002
1          17  q= -5.55555555555555525E-002
1          18  q= -5.26315789473684181E-002
1          19  q= -5.00000000000000028E-002
1           0  e=  0.50000000000000000
1           1  e=  0.16666666666666669
1           2  e=  8.33333333333333148E-002
```

```
1           3  e=  4.99999999999999889E-002
1           4  e=  3.33333333333333537E-002
1           5  e=  2.38095238095238082E-002
1           6  e=  1.78571428571428492E-002
1           7  e=  1.38888888888888951E-002
1           8  e=  1.11111111111110994E-002
1           9  e=  9.09090909090909394E-003
1          10  e=  7.57575757575758291E-003
1          11  e=  6.41025641025640136E-003
1          12  e=  5.49450549450550274E-003
1          13  e=  4.76190476190475886E-003
1          14  e=  4.16666666666666574E-003
1          15  e=  3.67647058823529493E-003
1          16  e=  3.26797385620915259E-003
1          17  e=  2.92397660818713434E-003
1          18  e=  2.63157894736841536E-003
2           0  q= -0.16666666666666669
2           1  q= -0.16666666666666660
2           2  q= -0.14999999999999999
2           3  q= -0.13333333333333344
2           4  q= -0.11904761904761897
2           5  q= -0.10714285714285710
2           6  q= -9.72222222222223070E-002
2           7  q= -8.88888888888887535E-002
2           8  q= -8.18181818181819426E-002
2           9  q= -7.57575757575758013E-002
2          10  q= -7.05128205128203456E-002
2          11  q= -6.59340659340661300E-002
2          12  q= -6.19047619047617681E-002
2          13  q= -5.83333333333333551E-002
2          14  q= -5.51470588235294379E-002
2          15  q= -5.22875816993464276E-002
2          16  q= -4.97076023391812491E-002
2          17  q= -4.73684210526314764E-002
2           0  e=  0.16666666666666677
2           1  e=  9.99999999999999223E-002
2           2  e=  6.66666666666665408E-002
2           3  e=  4.76190476190478246E-002
2           4  e=  3.57142857142856845E-002
2           5  e=  2.77777777777776375E-002
2           6  e=  2.22222222222224486E-002
2           7  e=  1.81818181818179103E-002
2           8  e=  1.51515151515152352E-002
2           9  e=  1.28205128205130386E-002
2          10  e=  1.09890109890106169E-002
2          11  e=  9.52380952380986467E-003
```

```
2          12  e=  8.33333333333317189E-003
2          13  e=  7.35294117647058293E-003
2          14  e=  6.53594771241830519E-003
2          15  e=  5.84795321637433113E-003
2          16  e=  5.26315789473690704E-003
3           0  q= -9.99999999999998251E-002
3           1  q= -9.99999999999998945E-002
3           2  q= -9.52380952380959128E-002
3           3  q= -8.92857142857137603E-002
3           4  q= -8.33333333333329401E-002
3           5  q= -7.77777777777790280E-002
3           6  q= -7.27272727272707808E-002
3           7  q= -6.81818181818196756E-002
3           8  q= -6.41025641025648740E-002
3           9  q= -6.04395604395572272E-002
3          10  q= -5.71428571428612905E-002
3          11  q= -5.41666666666635599E-002
3          12  q= -5.14705882352950936E-002
3          13  q= -4.90196078431373514E-002
3          14  q= -4.67836257309946352E-002
3          15  q= -4.47368421052631998E-002
3           0  e=  9.99999999999998529E-002
3           1  e=  7.14285714285705226E-002
3           2  e=  5.35714285714299771E-002
3           3  e=  4.16666666666665048E-002
3           4  e=  3.33333333333315496E-002
3           5  e=  2.72727272727306957E-002
3           6  e=  2.27272727272690156E-002
3           7  e=  1.92307692307700367E-002
3           8  e=  1.64835164835206854E-002
3           9  e=  1.42857142857065536E-002
3          10  e=  1.25000000000075953E-002
3          11  e=  1.10294117647016382E-002
3          12  e=  9.80392156862832514E-003
3          13  e=  8.77192982456102138E-003
3          14  e=  7.89473684210576648E-003
4           0  q= -7.14285714285705503E-002
4           1  q= -7.14285714285747136E-002
4           2  q= -6.94444444444419495E-002
4           3  q= -6.66666666666630436E-002
4           4  q= -6.36363636363760504E-002
4           5  q= -6.06060606060414842E-002
4           6  q= -5.76923076923207961E-002
4           7  q= -5.49450549450673092E-002
4           8  q= -5.23809523809078970E-002
4           9  q= -5.00000000000610720E-002
```

```
4          10  q= -4.77941176470086521E-002
4          11  q= -4.57516339869506530E-002
4          12  q= -4.38596491228012836E-002
4          13  q= -4.21052631578996903E-002
4           0  e=  7.14285714285663592E-002
4           1  e=  5.55555555555627412E-002
4           2  e=  4.44444444444454106E-002
4           3  e=  3.63636363636185428E-002
4           4  e=  3.03030303030652620E-002
4           5  e=  2.56410256409897036E-002
4           6  e=  2.19780219780235236E-002
4           7  e=  1.90476190476800977E-002
4           8  e=  1.66666666665533786E-002
4           9  e=  1.47058823530600152E-002
4          10  e=  1.30718954247596372E-002
4          11  e=  1.16959064327776946E-002
4          12  e=  1.05263157894626147E-002
5           0  q= -5.55555555555692360E-002
5           1  q= -5.55555555555475797E-002
5           2  q= -5.45454545454236647E-002
5           3  q= -5.30303030304005371E-002
5           4  q= -5.12820512819040716E-002
5           5  q= -4.94505494506334645E-002
5           6  q= -4.76190476192076134E-002
5           7  q= -4.58333333328359641E-002
5           8  q= -4.41176470595338113E-002
5           9  q= -4.24836601300809108E-002
5          10  q= -4.09356725149831249E-002
5          11  q= -3.94736842103812421E-002
5           0  e=  5.55555555555843975E-002
5           1  e=  4.54545454545693256E-002
5           2  e=  3.78787878786416704E-002
5           3  e=  3.20512820515617275E-002
5           4  e=  2.74725274722603108E-002
5           5  e=  2.38095238094493747E-002
5           6  e=  2.08333333340517471E-002
5           7  e=  1.83823529398555313E-002
5           8  e=  1.63398692825129158E-002
5           9  e=  1.46198830398574231E-002
5          10  e=  1.31578947373795774E-002
6           0  q= -4.54545454545392039E-002
6           1  q= -4.54545454543203997E-002
6           2  q= -4.48717948724421251E-002
6           3  q= -4.39560439551067578E-002
6           4  q= -4.28571428574984620E-002
6           5  q= -4.16666666683737502E-002
```

```
6           6  q= -4.04411764658487508E-002
6           7  q= -3.92156862814803595E-002
6           8  q= -3.80116958996453286E-002
6           9  q= -3.68421052677069191E-002
6           0  e=  4.54545454547881297E-002
6           1  e=  3.84615384605199451E-002
6           2  e=  3.29670329688970948E-002
6           3  e=  2.85714285698686066E-002
6           4  e=  2.49999999985740864E-002
6           5  e=  2.20588235365767465E-002
6           6  e=  1.96078431242239226E-002
6           7  e=  1.75438596643479466E-002
6           8  e=  1.57894736717958326E-002
7           0  q= -3.84615384601241783E-002
7           1  q= -3.84615384652865835E-002
7           2  q= -3.80952380901918020E-002
7           3  q= -3.75000000002197395E-002
7           4  q= -3.67647058978977648E-002
7           5  q= -3.59477123785213806E-002
7           6  q= -3.50877193582513000E-002
7           7  q= -3.42105262530932736E-002
7           0  e=  3.84615384553575398E-002
7           1  e=  3.33333333439918764E-002
7           2  e=  2.91666666598406690E-002
7           3  e=  2.57352941008960612E-002
7           4  e=  2.28758170559531307E-002
7           5  e=  2.04678361444940032E-002
7           6  e=  1.84210527695059731E-002
8           0  q= -3.33333333525970485E-002
8           1  q= -3.33333333104581450E-002
8           2  q= -3.30882352805183175E-002
8           3  q= -3.26797386864503409E-002
8           4  q= -3.21637423893237281E-002
8           5  q= -3.15789478329313797E-002
8           0  e=  3.33333333861307798E-002
8           1  e=  2.94117646897804966E-002
8           2  e=  2.61437906949640378E-002
8           3  e=  2.33918133530797434E-002
8           4  e=  2.10526307008863517E-002
9           0  q= -2.94117646230105251E-002
9           1  q= -2.94117645358481054E-002
9           2  q= -2.92397669756481078E-002
9           3  q= -2.89473663396739612E-002
9           0  e=  2.94117647769429162E-002
9           1  e=  2.63157882551640354E-002
9           2  e=  2.36842139890538900E-002
```

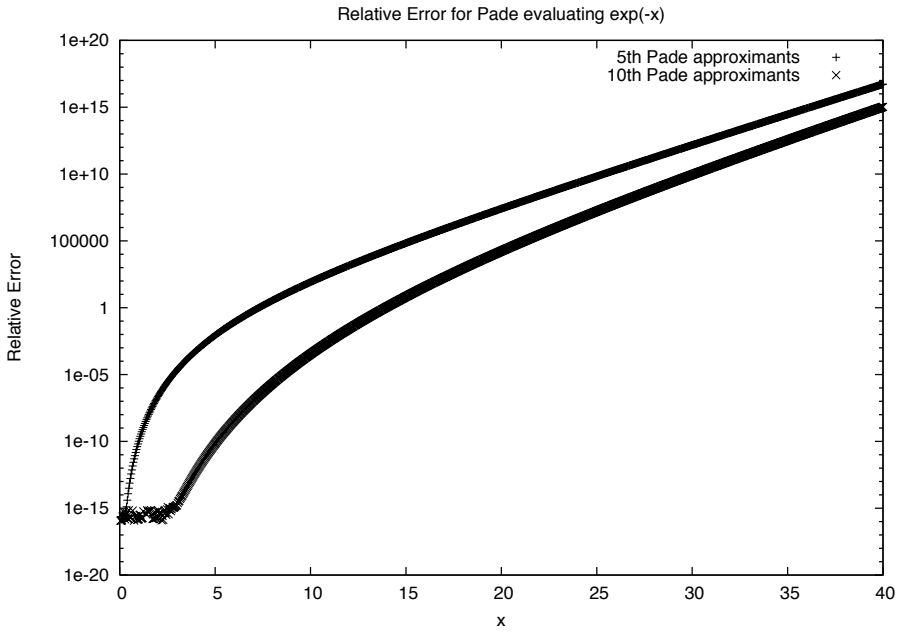
```
          10           0  q= -2.63157880394476358E-002
          10           1  q= -2.63157953440906409E-002
          10           0  e=  2.63157809505210304E-002
a1=  1.00000000000000000
a2=  1.00000000000000000
a3= -0.50000000000000000
a4=  0.16666666666666669
a5= -0.16666666666666677
a6=  9.99999999999998251E-002
a7= -9.99999999999998529E-002
a8=  7.14285714285705503E-002
a9= -7.14285714285663592E-002
a10=  5.55555555555692360E-002
a11= -5.55555555555843975E-002
a12=  4.54545454545392039E-002
a13= -4.54545454547881297E-002
a14=  3.84615384601241783E-002
a15= -3.84615384553575398E-002
a16=  3.33333333525970485E-002
a17= -3.33333333861307798E-002
a18=  2.94117646230105251E-002
a19= -2.94117647769429162E-002
a20=  2.63157880394476358E-002
a21= -2.63157809505210304E-002
```

The code:

```
program quotientDifference
integer,parameter:: NUM=30
integer::n,m,i,j,flag,pn,step
double precision::x,xmax,dx,error
double precision::pade,Avalue,Bvalue,Avalue_1,Avalue_2,Bvalue_1,Bvalue_2
double precision, dimension(0:NUM-1):: e,q
double precision, dimension(1:NUM):: a
!STEP 1, get an for cotinued fraction using quotient-difference algorithm
a(1)=1d0
!initial value for es
flag=1!flag is the index of array a, and also the total column of e and q
m=0!m is the lower index of e and p
e=0d0
!initial value for qs
flag=flag+1
m=m+1
do n=0,NUM-flag
   q(n)=-1d0/(dble(n)+1d0)
print *, m,n,'q=', q(n)
end do
```

```
a(flag)=-q(0)
!calculate es and qs and get the value of as
do i=3,NUM,2
   flag=i
   do n=0,NUM-flag
      e(n)=q(n+1)-q(n)+e(n+1)
   print *, m,n, 'e=',e(n)
   end do
   a(flag)=-e(0)
   m=m+1
   flag=flag+1
   do n=0,NUM-flag
      q(n)=e(n+1)/e(n)*q(n+1)
   print *, m,n, 'q=',q(n)
   end do
   a(flag)=-q(0)
end do
do i=1, NUM
   print*, a(i)
end do
!STEP 2, evaluate using pade
print *, 'enter the value of N for Pade:'
read *, pn
x=0d0
xmax=40d0
step=1000
dx=(xmax-0d0)/step
do i=0,step
   Avalue_1=0d0
   Avalue_2=1d0
   Bvalue_1=1d0
   Bvalue_2=0d0
   do j=1,pn
      if (j==1) then
         Avalue=Avalue_1+a(j)*Avalue_2
         Bvalue=Bvalue_1+a(j)*Bvalue_2
      else
         Avalue=Avalue_1+a(j)*x*Avalue_2
         Bvalue=Bvalue_1+a(j)*x*Bvalue_2
      end if
      Avalue_2=Avalue_1
      Avalue_1=Avalue
      Bvalue_2=Bvalue_1
      Bvalue_1=Bvalue
   end do
   pade=Avalue/Bvalue
```

```
   error=dabs(pade-dexp(-x))/dexp(-x)
   write(1,*) x,dexp(-x),pade,error
   x=x+dx
end do
end program quotientDifference
```



From the relative error plot, it can be seen that both 5th and 10th approximants are good estimation for $exp(-x)$ for small x, but bad for large x. The 10th approximants are about at least 2 order better than the 5th approximants for large x. And there are some small oscillation for 10th approximants at the very beginning.

Problem4

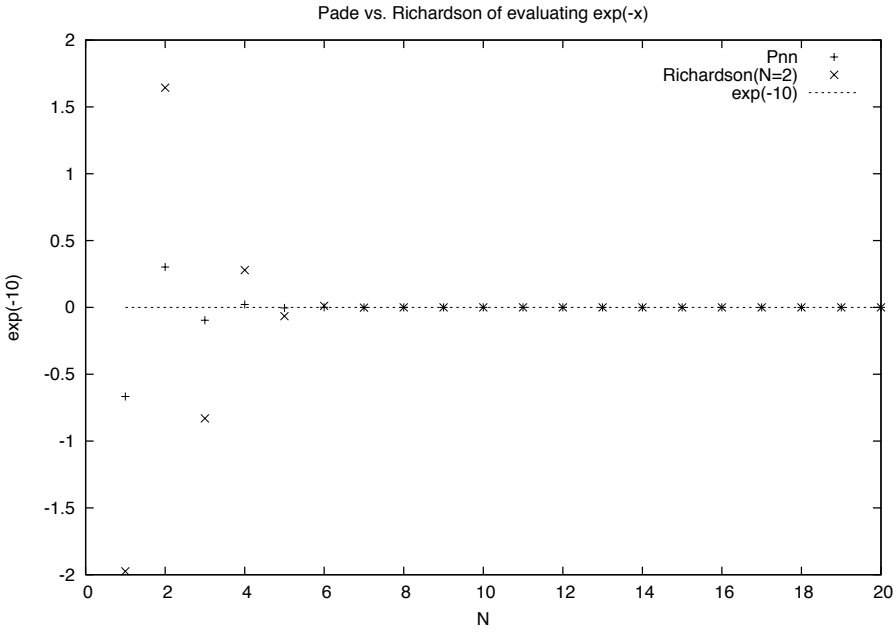
I set $A_n = P_{nn}$ here and fix the N to be 2 while varying n from 1 to 30 because this method kind of makes sense to me. Both of the Padé and Richardson jumps a little bit at the beginning and then converge to a quite good value. But I didn't see any obvious improvement from Padé to Richardson.

The code:

```
program main
 IMPLICIT NONE
 DOUBLEPRECISION :: richard
 INTEGER :: k, nlow, ncap
```

```
 ncap = 2!ncap is the number of Pade temrs being grouped together to form Richardson transfo
 WRITE(1,*) "N for Richardson is", ncap
 WRITE(1,*) "n,       Pade,       Richardson,       exp(-10)"
do nlow=1,20 !nlow is the N index of P_NN
!do Richardson series
   richard = 0D0
  DO k = 0, ncap
   richard = richard + pade(2*(k+nlow)+1) *dble(nlow+k)**ncap * (-1D0)**(k+ncap)/dble( facto
  ENDDO
  WRITE(1, '(I2, 1F20.15, 2ES25.14)') nlow, pade(2*nlow+1), richard, DEXP(-10D0)
end do
contains
double precision function pade(pn)
integer,parameter:: NUM=60
integer::n,m,i,flag,pn
double precision::x=10d0,Avalue,Bvalue,Avalue_1,Avalue_2,Bvalue_1,Bvalue_2
double precision, dimension(0:NUM-1):: e,q
double precision, dimension(1:NUM):: a
!STEP 1, get an for cotinued fraction using quotient-difference algorithm
a(1)=1d0
!initial value for es
flag=1!flag is the index of array a, and also the total column of e and q
m=0!m is the lower index of e and p
e=0d0
!initial value for qs
flag=flag+1
m=m+1
do n=0,NUM-flag
   q(n)=-1d0/(dble(n)+1d0)
end do
a(flag)=-q(0)
!calculate es and qs and get the value of as
do i=3,NUM,2
   flag=i
   do n=0,NUM-flag
      e(n)=q(n+1)-q(n)+e(n+1)
   end do
   a(flag)=-e(0)
   m=m+1
   flag=flag+1
   do n=0,NUM-flag
      q(n)=e(n+1)/e(n)*q(n+1)
   end do
   a(flag)=-q(0)
end do
!STEP 2, evaluate using pade
```

```
   Avalue_1=0d0
   Avalue_2=1d0
   Bvalue_1=1d0
   Bvalue_2=0d0
   do i=1,pn
      if (i==1) then
         Avalue=Avalue_1+a(i)*Avalue_2
         Bvalue=Bvalue_1+a(i)*Bvalue_2
      else
         Avalue=Avalue_1+a(i)*x*Avalue_2
         Bvalue=Bvalue_1+a(i)*x*Bvalue_2
      end if
      Avalue_2=Avalue_1
      Avalue_1=Avalue
      Bvalue_2=Bvalue_1
      Bvalue_1=Bvalue
   end do
   pade=Avalue/Bvalue
end function pade
integer FUNCTION factorial(nn)
  IMPLICIT NONE
  INTEGER :: iter,nn,fact
  fact=1
  DO iter = 1, nn
   fact = fact*DBLE(iter)
  ENDDO
   factorial = fact
   RETURN
 END FUNCTION
end program
```


Pade vs. Richardson of evaluating exp(-x)
Pnn
Richardson(N=2)
exp(-10)
exp(-10)
N
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
2
4
6
8
10
12
14
16
18
20