Name: Manuel Tobon Bayona Subject: Numerical Methods Assessment: PEC_1

Exercise 1

The Bessel functions are defined byJk(x) =1π∫π0cos(x sinθ−(k−1)θ)dθ but they satisfy the forward recurrence relationJn(x) =2n−4xJn−1(x)−Jn−2(x)(1) that we will use for n ∈ N. Then, from the initial valuesJ1(1)'0.7651976865 and J2(1)'0.4400505857 you can obtainJ3(1),J4(1),... that, by their definition, they must be in [0,1].

a) Assuming round-off error in these two values, give an upper bound for their relative errors.

I have 10 decimals therefore the calculations is:

er= 0,5 *10 $^{1-t}$, given t = 10

er= 0,5*10^{-9}

b) Fill in the following table:
J1(1)(1)J2(1)J3(1)J4(1)J20(1)J21(1) 0.7651976865 0.4400505857.....…

values are taken from print statement of the “script1.b.py”

index: 3 : 0.11490348490000002
index: 4 : 0.01956335390000008
index: 5 : 0.002476638500000461
index: 6 : 0.0002497541000036074
index: 7 : 2.0902500035613336e-05
index: 8 : 1.075900423752607e-06
index: 9 : -5.839894103076837e-06
index: 10 : -9.4514206072982e-05
index: 11 : -0.0016954158152105991
index: 12 : -0.033813802098139
index: 13 : -0.7422082303438474
index: 14 : -17.7791837261542
index: 15 : -461.5165686496654
index: 16 : -12904.684738464477
index: 17 : -386679.02558528463
index: 18 : -12360824.133990644
index: 19 : -419881341.5300966
index: 20 : -15103367470.949486
index: 21 : -573508082554.5503

c) Using the formulas for the propagation of errors in the input data, give upper bounds of the relative error forJ3(1),J4(1),...J21(1). What do you observe in these results?

To solve this it is used the “script1c.py”
which is calculating the first 21 values for the Bessel series and its relative errors.
Print statement for this corresponding:

list of errors decimal places---

[5e-10, 5e-10, 5e-17, 5e-17, 5e-18, 5e-19, 5e-21, 5e-21, 5e-21, 5e-18, 5e-19, 5e-15, 5e-16, 5e-16, 5e-16, 5e-14, 5e-18, 5e-17, 5e-16, 5e-16, 5e-11]

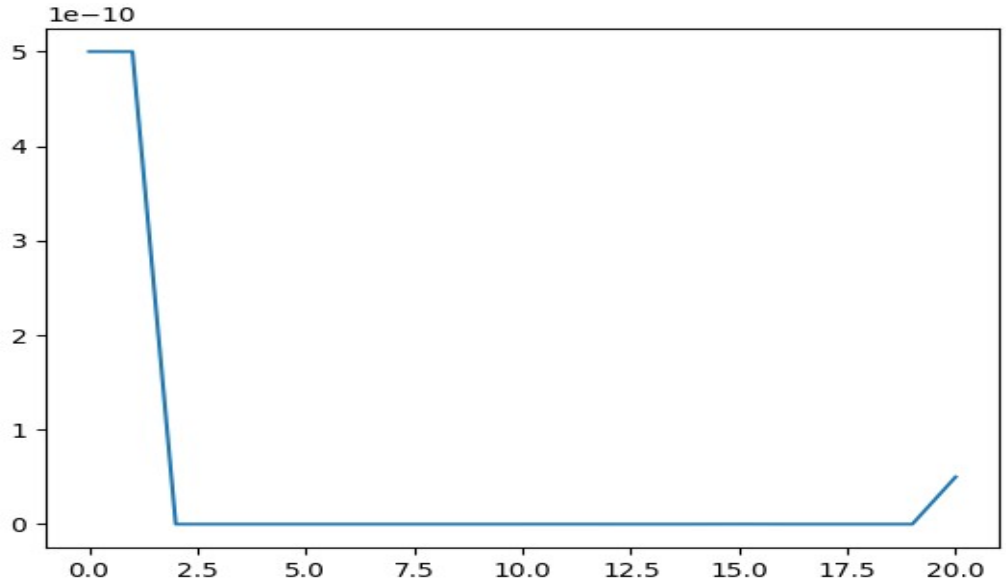

figure 1.c. this graph depicts how here is a minimal relative error point and then it is increasing again when the Bessel function is growing in dimensionality.
Is not a line , it is a curve more similar to quadratic functions.
X axis = position in j_n, Y axis = relative error.

d) From the valuesJ21(1)'3.873502 10−25andJ20(1)'1.548478 10−23, we could use the recurrence (1) in the reverse direction. From (1), deduce the expression of the backward recurrence and use it to obtainJ1(1) andJ2(1).
Re arrange the equations it obtains:

Equation backward Bessel: $J_{n-2} = 2n*J_{n-1} + J_n$

e) Using the formulas for the propagation of errors in the input data, give upper bounds of the relative error forJ19(1),J18(1),...J1(1) computed from the backward recurrence. What can you comment about them?

Running the “script1dnE.py” is used to print the calculations of backward Bessel Function of previous question. It is clear the backward Bessel function does NOT meet same results as forward one.

index19 : -2.3148731999999986e-24
index18 : 4.978278332e-22
index17 : -5.672740291999999e-22
index16 : 1.45064533588e-20
index15 : -2.9255578117999996e-20
index14 : 3.7741045872919993e-19
index13 : -1.0210331773251999e-18
index12 : 8.569242351909198e-18
index11 : -2.6947839543762797e-17
index10 : 1.6405571717430998e-16
index9 : -5.413254707869891e-16
index8 : 2.5099940768787086e-15
index7 : -7.9232487847486e-15
index6 : 2.800320139977827e-14
index5 : -7.554269410826987e-14
index4 : 1.8755549970738293e-13
index3 : -3.3864088792392266e-13

index2 : 3.3864088792392266e-13
index1 : 3.3864088792392266e-13

Exercise 2.
Given the function f(x) =e2xcos(3x), we want to compute its value at any irrational numberx0∈[0,π] using an approximation x∗0.
a) Using the formulas of the propagation of errors in the input data and an approximate value x∗0with an absolute errorεa(x0) =#, give an expression for an upper bound of the absolute error off(x0) and another one for its relative error.

Using the definition of error propagation, its calculated the derivates and then evaluated in a specific point times Es (error).

upper bound of the absolute error off(x0):

$E_a = |f'(x)| = |(2e^{2x})\cos(3x) - \text{sen}(3x)3e^{2x}| * Es$ where x is evaluated in x0.

$E_r = E_a(x0)/f(x0) = (|(2e^{2x})\cos(3x) - \text{sen}(3x)3e^{2x}| * Es) / (e^{2x}\cos(3x))$ where x= x0.

b) Using the formulas of the propagation of errors in the input data, deduce how far from x0 we can take x∗0 in order to assure that the approximation f(x0)'f(x∗0) will have an error smaller than 0.5·10−6

in order to know which is the max far value from x0 and fulfilling the condition of error, its needed apply the expression of in-equation:

Er <= 0.5·10−6
this means if we replace x=0 in our eq of Er we obtain:
$E_r = E_a(x0)/f(x0) = (|(2e^{2x})\cos(3x) - \text{sen}(3x)3e^{2x}| * Es) / (e^{2x}\cos(3x))$ x=0

Er= 2Es, so 2Es > or = than $0,5 * 10^{-6}$
turning out: Es <= $0,25 * 10^{-6}$ this is the maximum separated value from x0 to satisfiy the condition of Er.

Exercise 3.
Consider the problem of solving:
A= [2−1 1]
[2 2 2]
[−1−1 2]

b=[x, y, z]= [1 1 1]

a) Give the Jacobi's iterative matrix and Gauss-Seidel's iterative matrix for this system.

Sol:

Applying transformations to get Jacobi's Matrix:

where,
A= D+L+U

D= [2 0 0]
[0 2 0]
[0 0 2]

L= [0 -1 1]
[0 0 2]
[0 0 0]

U= [0 0 0]
[2 0 0]
[-1 -1 0]

b= [1 1 1]

R= L + U
C= D^{-1}.b
T= D^{-1}.R

Jacobi's Matrix and iterative Sol:

$x^{k+1} = D^{-1}. (b – R.x^k) = T.x^k + C$

Applying transformations to get Gauss-Seidel's Matrix:

$x^{k+1} = M.x^k + c$

where:

M= N^{-1}.P

b= [1 1 1]

N= [2 -1 1]
[0 2 2]
[0 0 2]

P= [0 0 0]
[2 0 0]
[-1 -1 0]

c = N^{-1}.b

b) Compute both spectral radius (if you want, you can use MATLAB to do it).

In order to obtain spectral radius , here it is used the expression of:
the solutions are provided by Matlab script (its used script3b.m)

Jacobi:
p(D^{-1}.(L+U)), this means its necessary to calculate the eigenvalues of a matrix and pick the max value.

Jacobi p radius:: 1.118

Gauss-Seidel's:
p(N^{-1}*P)

Gauss-Seidel p radius:: 1.6514

c) Discuss, without solving the system, the convergence of the two methods applied to this system.
This linear system equations is not going to converge because:

1. The matrix of the system is NOT strictly or irreducibly diagonally dominant. The matrix's coefficients are not in the order to fulfill the condition and also there is not a transformation to apply here as swapping rows.
2. p radius are greater than 1 and that is violating the conditions of convergence.

Exercise 4.
We want to solve the following linear system using Jacobi's method andGauss-Seidel's method.

Provide the answer to the next questions for Jacobi's and Gauss-Seidel's ite-rative methods

a) Compute the iterative matrix and array

Jacobi method:

.T=-D^{-1}R
C=D^{-1}b
R=L+U
all of this to have the all matrix separated among them.

R = [0.25]
[0.5]
[0.]
[0.25]

T = [0. 0.25 0.25 0.]
[0.25 0. 0. 0.25]
[0.25 0. 0. 0.25]
[0. 0.25 0.25 0.]

$x^{k+1} = T.x^{k} + C$

so it gives a first result with $[0,0,0,0]^{T}$,

Jacobi Result for [[0 0 0 0]]
x^{1}:: [[0.25 0.5 0. 0.25]]

Gauss-Seidel method:

For this method , its necessary obtain all equation for each variable , x1, x2, x3 , x4.
Initial values:

x1,x2,x3,x4 = 0,0,0,0

equations:

x1 = (1+x2+x3)/4
x2=(2+x1+x4)/4
x3=(x1+x4)/4
x4=(1+x2+x3)/4

After run once , results are the following ones:

First Solution of Linear matrix with 0,0,0,0
21= 0.25 , x2=0.5625 , x3= 0.0625 , x4= 0.40625 .

for these both cases , they have a representation and model in python.

Jacobi script: "script4_a_jacobi.py" .

Gauss-Seidel script: "script4_a_GaussS.py" .

b) Fill in the following tables with the first four approximations of the solution, x(1),x(2),x(3)and x(4), their differences d(k)=x(k)−x(k−1) and their approximated relative errors.

x^0= [[0.25]
[0.5]
[0.]
[0.25]]

x^1 = [[0.375]
[0.625]
[0.125]
[0.375]]

x^2= [[0.4375]
[0.6875]
[0.1875]
[0.4375]]

x^3= [[0.46875]
[0.71875]
[0.21875]
[0.46875]]

After the run of the script for jacobi , 4 times we got same error distance for all d values at the same time:
d1 = d2 =d3=d4
and for each time that it was run:

[0.25, 0.125, 0.0625]

c) Without computing iterations, make a reasoned prediction about how many iterationsM would be necessary to have a solution with a relative error#(M)r≤0.5·10^{-14}

The solutions I propose is that after review the last measure of the error, I must say its probably that the error is decreasing in order of half for each time.

So the eq for calculate how many runs must be played is:

$2^N = 0{,}25/0.5\cdot 10^{-14}$

after used a calculator , the value of N = 46 (approx).

I concluded after N=50 its more than enough in order to get the desired relative error.

d) Check if your previous prediction is correct: compute the actual relative error for the approximation x(M)and compute the first approximation x(N)that satisfies the condition. Only here, you can use the exact solution (0.5,0.75,0.25,0.5)T.Fill in the following table:

Having the exact solutions make the code iterates several times until the differences between exact solutions and numerical solutions have a small error, this case I set the tolerance in
Es= 0,5 *10^{-8}

when the values do not go away from the exact solutions, we will get the numbers of iterations and stop the script.

The Script used for this operation is: “script4_d.py”
and the number of iterations is:
N= 26, after this number make more iterations and getting more resolutions in scale is not significant, here is already reach the exact solution vector.

Moreover, with more iterations, exactly with 47 its getting the error of previous point ($er=0.5\cdot10^{-14}$)
M=50
e(M)= 8.881784197001252e-16
e(M-1)= 1.7763568394002505e-15

N=26
e(N)= 3.725290298461914e-09

e(N-1) =7.450580596923828e-09

e) A criterion to stop the iterative process is thatE(k)would be smaller than the precision you want, in this case 0.5·10−14. Compute the number of iterations L required to satisfy this condition and fill in the following table:

Script used for this: “script4_e.py”
As I said before its needed 47 times of executions to have this error.

L=47
E(L)= 3.552713678800501e-15
E(L-1)=7.105427357601002e-15

Comparing point d and e, we see that after the prediction I made around 50, its only necessary 26 times for obtaining the exact solution with a good error rate.