AE2-202 Numerical Analysis

Coursework

Lecturer: Dr Ajit Panesar

Your name:

Jo Wayne Tan

Your CID:

01327317

Due date: Monday, 18 Feb 2019 (via Blackboard)

Each individual problem is marked out of a total of 100% and broken down as indicated in the right-hand margin.

You should submit the provided document for this coursework:

- Do not modify the margins, use a font Arial (12) and use a line spacing set at 1.15.
- Do not exceed the given page numbers.
- The pdf document should be self-contained and markable without reference to the entire code and without the need to run them. However, please provide your MATLAB code in the appendix (use font size 8).
- Clearly display the plot with large enough line width, font size, grid and axis labels.
- Do not present the information as a report. Be specific and provide evidence to your argument.
- Export file as a professionally-typed pdf document (no hand-writing, or smartphone photos of hand-written solutions).
- Note: the programs and question answers will be checked for similarities.

Question 1 (worth 40% of total)

a) An electric sports car has two 6-bit sensors installed. One reports the time (in minutes) for journey made and another the distance travelled (in kilometers), both are limited by the capacity of the on-board battery.

Time	0	1	0	1	0	0
Distance	1	1	0	1	1	0

It has been brought to your attention that the highlighted bits are faulty (i.e. for both Time & Distance measurements) and displays a random 0/1 value with equal probability.
What is the expectation for the computed speed (in km/min) and the associated error in its value. Justify your approach. [30%]

010100 is $2^4 + 2^2 = 16 + 4 = 20$ (maximum time)

010000 is $2^4 = 16$ (minimum time)

110110 is $2^5 + 2^4 + 2^2 + 2^1 = 32 + 16 + 4 + 2 = 54$ (maximum distance)

110010 is $2^5 + 2^4 + 2^1 = 32 + 16 + 2 = 50$ (minimum distance)

Speed = Distance/Time

Largest speed possible = $\frac{\max distance}{\min time} = \frac{54}{16} = 3.375\ km/min$

Smallest speed possible = $\frac{\min distance}{\max time} = \frac{50}{20} = 2.5\ km/min$

Expectation for the computed speed is the average between largest and smallest value → $\frac{3.375}{2.5} = 2.9375\ km/min$

Error is difference between largest or smallest value and the average value calculated → 3.375 – 2.9375 = 0.4375 km/min

ANS: $2.9375 \pm 0.4375\ km/min$

b) Given a quadratic function y=f(x) (see table for values), report how many stencil points would be sufficient for obtaining an exact value for $\frac{dy}{dx}_{x=2}$ (employing finite difference scheme) and what would this derivative value be [10%]

x	1.8	2	2.1	2.2
y	2.64	3	3.21	3.44

Forward Difference:

$$\frac{df}{dx}(x_i) \approx \frac{f_{i+1} - f_i}{\Delta x}$$

$$\frac{dy}{dx}_{x=2} \approx \frac{3.21 - 3}{0.1} = 2.1$$

Backward Difference:

$$\frac{df}{dx}(x_i) \approx \frac{f_{i+1} - f_i}{\Delta x}$$

$$\frac{dy}{dx}_{x=2} \approx \frac{3 - 2.64}{0.2} = 1.8$$

Central Difference:

$$\frac{df}{dx}(x_i) \approx \frac{f_{i+1} - f_{i-1}}{2\Delta x}$$

$$\frac{dy}{dx}_{x=2} \approx \frac{3.44 - 2.64}{2(0.2)} = 2$$

The central difference approx is second-order accurate while the forward and backward difference approx is first-order accurate.

Generally speaking, higher-order accuracy is better. Also, since f(x) is quadratic and not linear, 3 stencil points would be sufficient for obtaining an exact value, because $\epsilon_{trunc} = 0$

So, the derivative value is 2.

c) Find the approximation to $\frac{dy}{dx}$ for y = sin(x) at x = 0.625 using finite difference methods, specifically, forward difference (FD), backward difference (BD) and central difference (CD). You are allowed to evaluate the function value precisely but are limited to express the value of x with only 5-bits for the fractional part. Clearly show the decimal numbers you obtain for x, report the derivative values (7 decimal place accuracy) for the schemes employed and their errors from the analytical value. Comment on the trend you see in the errors, even for FD & BD. [20%]

The value of x in 5-bits binary is 0.10100. Hence, the value after smallest increment and decrement, x_{i+1} and x_{i-1}, in 5-bits is 0.10101 and 0.10011, which translate to 0.65625 and 0.59375. Using finite difference approximation formulas from 1b, FD, BD and CD are respectively

$$\frac{dy}{dx}_{x=0.625} \approx \frac{\sin(0.65625) - \sin(0.625)}{0.03125} = 0.8016897$$

$$\frac{dy}{dx}_{x=0.625} \approx \frac{\sin(0.625) - \sin(0.59375)}{0.03125} = 0.8199725$$

$$\frac{dy}{dx}_{x=0.625} \approx \frac{\sin(0.65625) - \sin(0.59375)}{2(0.03125)} = 0.8108311$$

The percentage error of FD, BD and CD from the analytical value,

cos(0.625) = 0.8109631, are 1.14%, 1.10% and 0.016%.

As we can see CD is far more accurate than FD and BD, with FD and BD having similar values.

d) Consider the differential equation $\frac{dU}{dt} - \frac{d^4U}{dx^4} = 0$. Use the systematic method to construct the finite-difference approximation of the spatial derivative using the following stencil (-2,-1,0,1,2). Report the coefficients obtained and the order of the truncation error in this approximation. Comment under which circumstance will this approximation provide exact solution. [40%]

$$\frac{d^4U}{dx^4} = a_{-2}f_{i-2} + a_{-1}f_{i-1} + a_0 f_i + a_1 f_{i+1} + a_2 f_{i+2}$$

$$f_{i-2} = f_i + f_i' \frac{-2\Delta x}{1!} + f_i'' \frac{(-2\Delta x)^2}{2!} + f_i''' \frac{(-2\Delta x)^3}{3!} + f_i'''' \frac{(-2\Delta x)^4}{4!}$$

$$f_{i-1} = f_i + f_i' \frac{-\Delta x}{1!} + f_i'' \frac{(-\Delta x)^2}{2!} + f_i''' \frac{(-\Delta x)^3}{3!} + f_i'''' \frac{(-\Delta x)^4}{4!}$$

$$f_{i+1} = f_i + f_i' \frac{\Delta x}{1!} + f_i'' \frac{(\Delta x)^2}{2!} + f_i''' \frac{(\Delta x)^3}{3!} + f_i'''' \frac{(\Delta x)^4}{4!}$$

$$f_{i+2} = f_i + f_i' \frac{2\Delta x}{1!} + f_i'' \frac{(2\Delta x)^2}{2!} + f_i''' \frac{(2\Delta x)^3}{3!} + f_i'''' \frac{(2\Delta x)^4}{4!}$$

$$a_{-2} + a_{-1} + a_0 + a_1 + a_2 = 0$$

$$-2a_{-2} - a_{-1} + a_1 + 2a_2 = 0$$

$$\frac{4}{2}a_{-2} + \frac{1}{2}a_{-1} + \frac{1}{2}a_1 + \frac{4}{2}a_2 = 0$$

$$-\frac{8}{6}a_{-2} - \frac{1}{6}a_{-1} + \frac{1}{6}a_1 + \frac{8}{6}a_2 = 0$$

$$\frac{16}{24}a_{-2} + \frac{1}{24}a_{-1} + \frac{1}{24}a_1 + \frac{16}{24}a_2 = \frac{1}{(\Delta x)^4}$$

Using MATLAB to solve the simultaneous equations in matrix form,

a_{-2} & $a_2 = \frac{1}{(\Delta x)^4}$, a_{-1} & $a_1 = -\frac{4}{(\Delta x)^4}$, $a_0 = \frac{6}{(\Delta x)^4}$

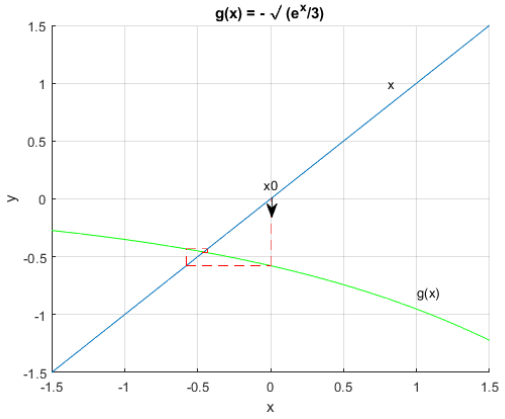
Order of the truncation error is second-order accurate, $O(\Delta x^2)$.

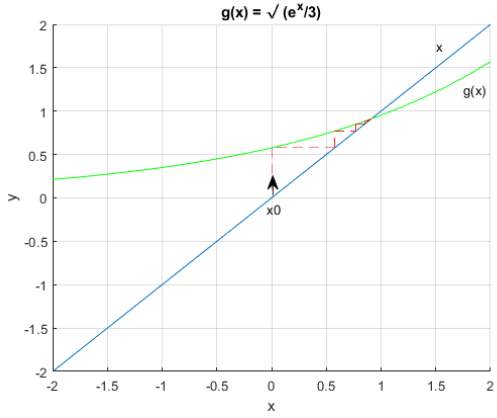
The approximation will provide an exact solution if the function is quartic.

Question 2 (worth 30% of total)

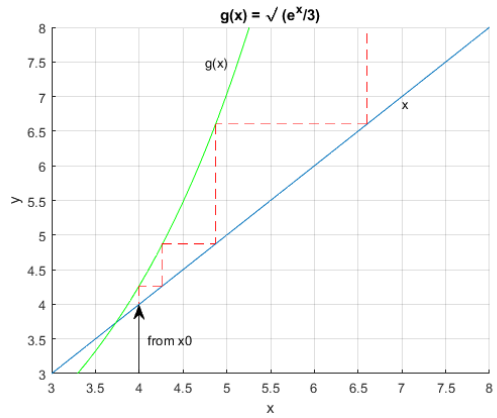
a) Consider $f(x) = e^x - 3x^2 = 0$ which has 3 roots. An obvious arrangement to find roots using fixed-point method is $x = g(x) = \pm\sqrt{\frac{e^x}{3}}$

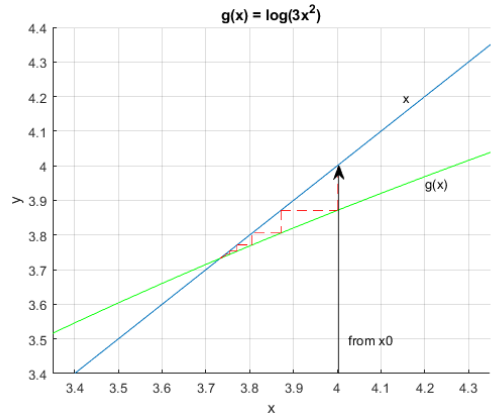
b) Show that convergence is to the root near -0.5 if we begin with $x_0 = 0$ and use the negative value of $g(x)$. Show also that convergence to a second root near 1.0 is obtained if $x_0 = 0$ and the positive value is used. Show, however, that this form does not converge to the third root near 4.0 even though a starting value very close to the root is used. Find a different arrangement that will converge to the root near 4.0. General hint: you may find that evaluating $g'(x)$ for x close to the required roots reveals important information. Use MATLAB plots to clearly show the root finding process [30%]





Convergence to the root -0.459 (left) and 0.910 (right)





Divergence when x0 = 4 (left). New arrangement of g(x) that converges to the root 3.7332, which is near 4.0 (right). Form diverges if abs(g'(x)) ≥ 1

$$g(x) = \pm\sqrt{\frac{e^x}{3}}, g'(x) = \pm\frac{1}{2}\left(\frac{e^x}{3}\right)^{\frac{1}{2}} \quad x = 0, g'(x) = 0.289 \quad x = 4, g'(x) = 2.13$$

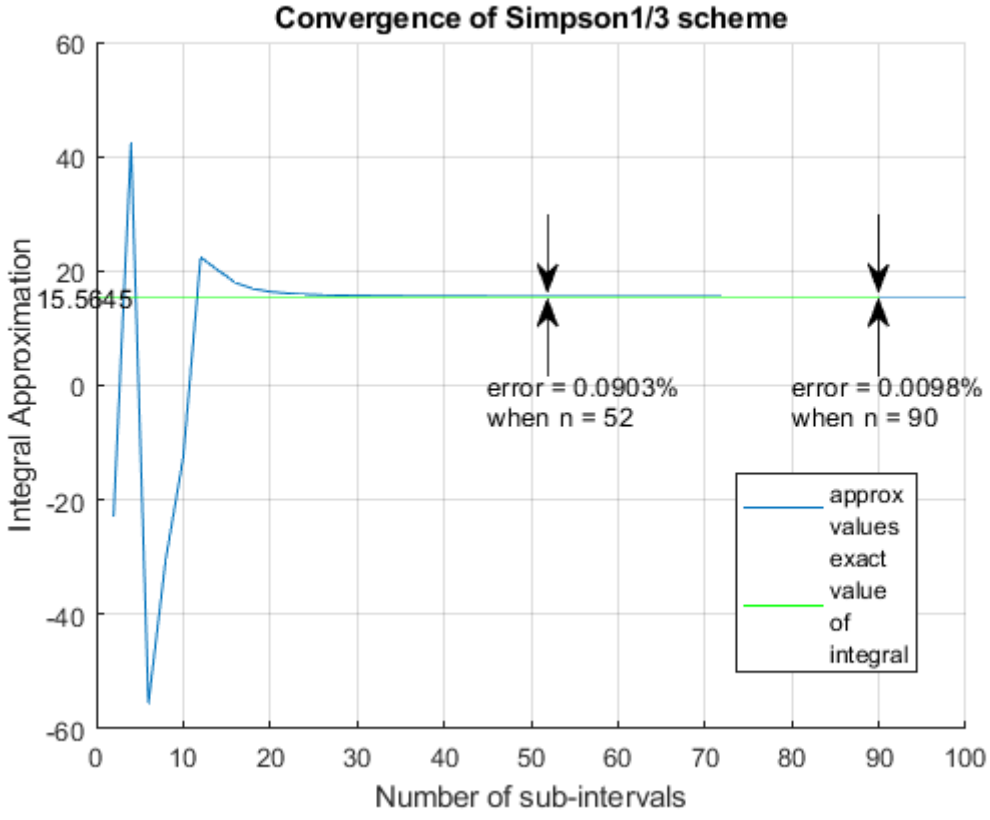
$$g(x) = \log(3x^2),\ g'(x) = 2/x \quad x = 4, g'(x) = 0.5$$

c) Find an approximation to the integral

$$\int_{-\frac{3}{2}\pi}^{0} \frac{\cos 7x}{e^x}\, dx$$

how many sub-intervals are required to have a relative error (take the approximate value) less than 0.1% and 0.01%, respectively? Starting with the minimum possible number of intervals. Show a MATLAB plot (displaying function value) for how the Simpson1/3 scheme converges. Also, specify the sub-interval number identified above where the criterion is met and the final value for integral.

(Use the expression $\frac{1}{50}(7e^{\frac{3\pi}{2}} - 1)$ for computing the exact integral) [30%]



$$\frac{1}{50}\left(7e^{\frac{3\pi}{2}} - 1\right) = 15.5645 \quad (exact\ value)$$

52 and 90 are the minimum possible number of sub-intervals, n, required to have a relative error less than 0.1% and 0.01% respectively.

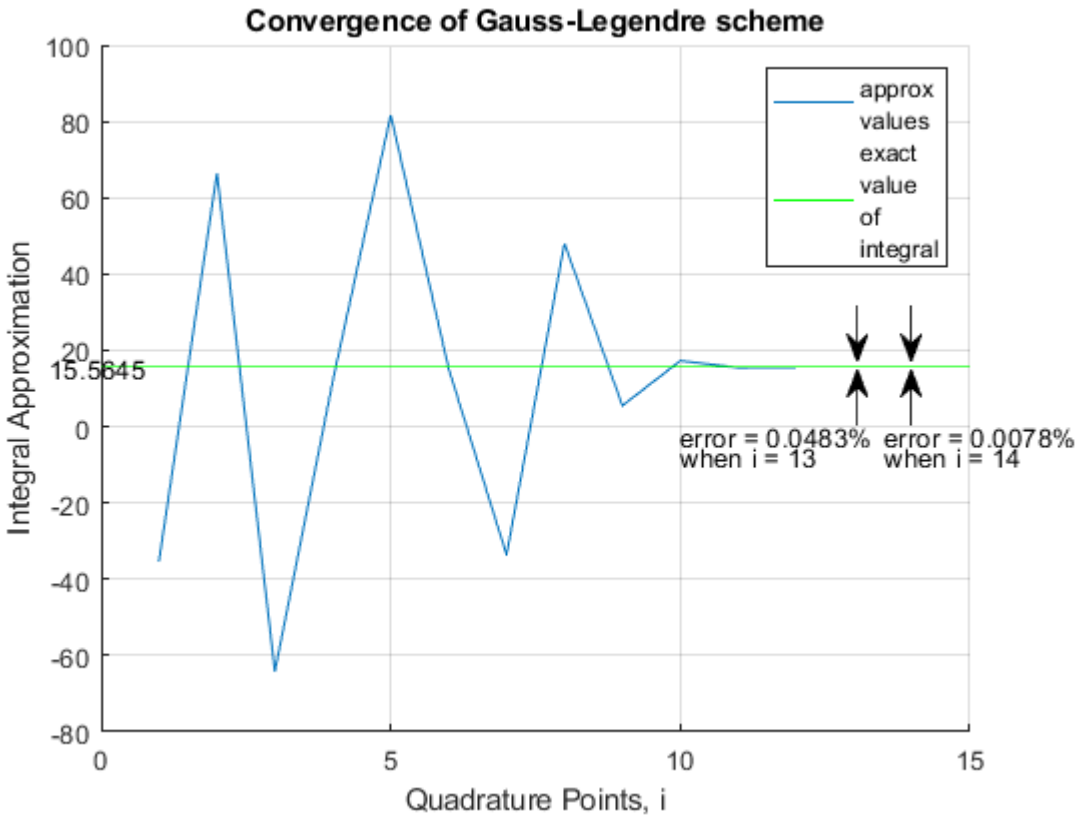
Approximate value of integral is 15.5785 when n = 52 and 15.5660 when n = 90. Final approximate value in the graph is 15.5655 when n = 100.

d) Vary the quadrature points in the Gauss-Legendre scheme to produce the same as requested in b). Contrast with b) and discuss why much fewer additional points are needed when considering relative error from 0.1% to 0.01% to obtain the approximation for this integral. [40%]

Using $x = \frac{a+b}{2} + \frac{b-a}{2}t$ and $dx = \frac{b-a}{2}dt$ to change the limits to the standard form of the quadrature, (-1,1) and evaluate the quadrature in the new t variable.

$$\int_{-\frac{3}{2}\pi}^{0} f(x)dx = \int_{-1}^{1} f(t)\left(\frac{-\frac{3}{2}\pi}{2}\right)dt = -\frac{3}{4}\pi\int_{-1}^{1}\frac{\cos 7(-\frac{3}{4}\pi - \frac{3}{4}\pi t)}{e^{(-\frac{3}{4}\pi - \frac{3}{4}\pi t)}}dt$$

$$\int_{-1}^{1} f(t)dt \approx \sum_{i=1}^{N} w_i f(\eta_i) \quad where \quad f(t) = \frac{\cos 7(-\frac{3}{4}\pi - \frac{3}{4}\pi t)}{e^{(-\frac{3}{4}\pi - \frac{3}{4}\pi t)}}$$



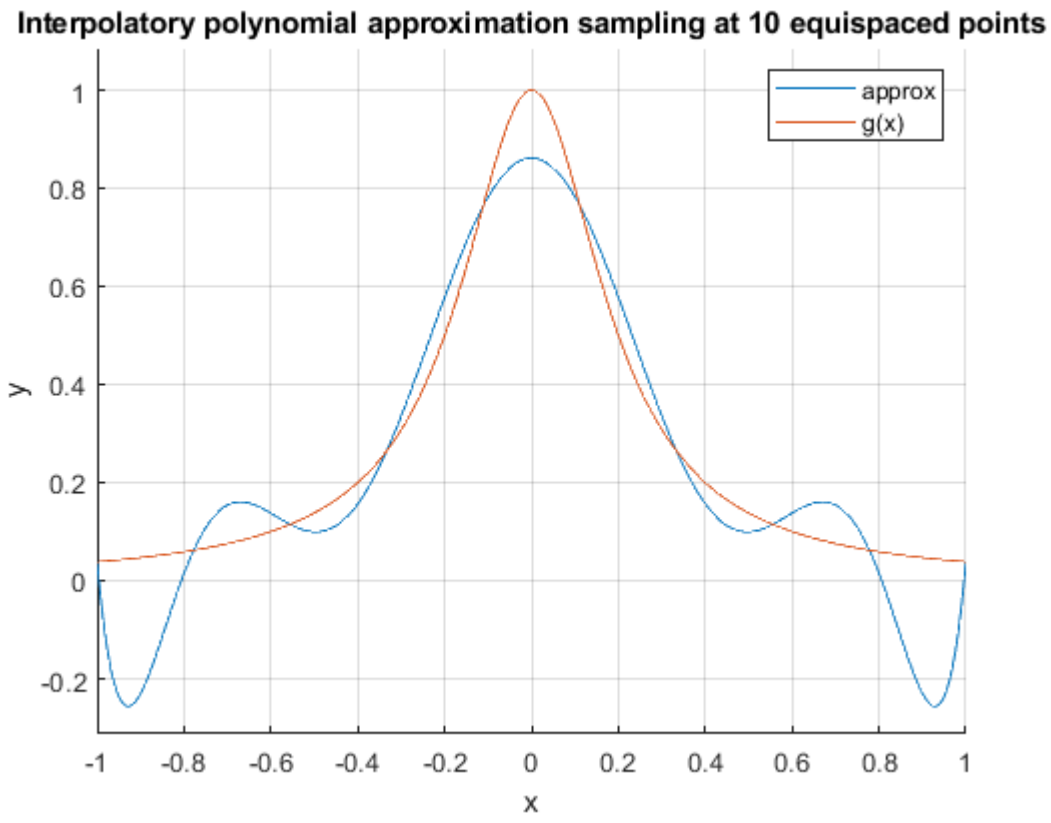
13 and 14 are the quadrature points required to have a relative error less than 0.1% and 0.01% respectively with approximate values of 15.572 and 15.5633 The Gauss-Legendre graph produces more oscillations with a larger approximation range compared to Simpson1/3 scheme but converges faster.

The Gauss-Legendre quadrature with N points can integrate polynomials of degree 2N – 1 exactly whereas the Simpson1/3 rule with the same N points can only integrate polynomials of degree N – 1, therefore requiring much more points to get similar relative errors.

Question 3 (worth 30% of total)

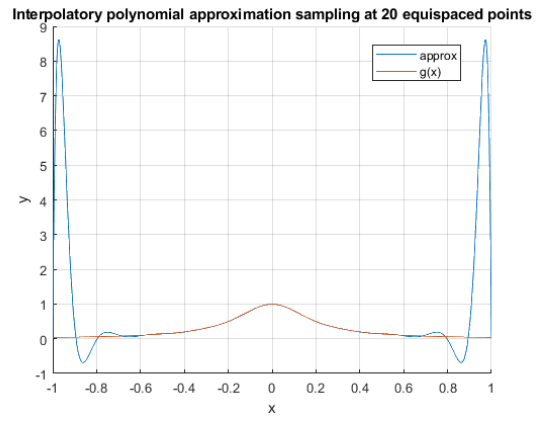
The following question is about constructing interpolatory polynomial approximations of arbitrary functions using Lagrange polynomials.

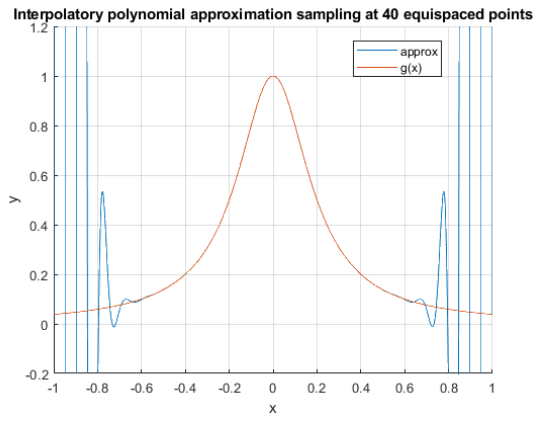
a) Produce the interpolatory polynomial approximation of $g(x) = \frac{1}{(1+25x^2)}$ in the standard interval -1 ≤ x ≤ 1 by sampling it at 10 equispaced points (include the end points). Plot the interpolatory polynomial approximation on top of the true function $g(x)$ using 1001 equispaced points in the given interval. Additionally, calculate the L_2 and L_∞ norm (check definition for vectors! - http://mathworld.wolfram.com/L2-Norm.html) of the error norm between the interpolatory polynomial approximation and $g(x)$ within the standard interval. Note: if you plotted $g(x)$ and the interpolatory function using 1001 points, your error vector should also be 1001 points long. [20%]

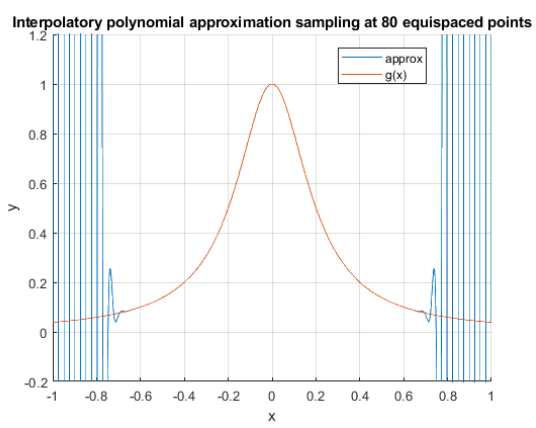


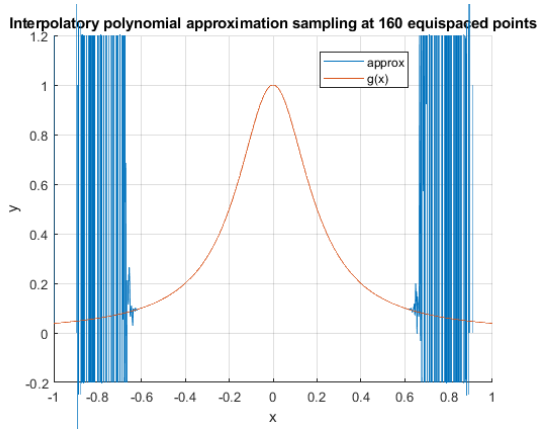
L_2 and L_∞ norm of the error norm between the interpolatory polynomial approximation and $g(x)$ within the standard interval is 3.4658 and 0.3003 respectively.

b) Now construct interpolatory polynomial approximations of g(x) in the standard interval by sampling it at 20, 40, 80, and 160 equispaced points respectively (include the end points). Plot the resulting interpolatory polynomial approximations such that they can be compared clearly with the true function g(x). Additionally, calculate the L_2 and L_∞ norm of the error between each interpolatory polynomial approximation and g(x) when evaluated at the 1001 points. Make plots to show how each of these norms vary as the number of points used to sample g(x) is increased. Comment on the results. [40%]



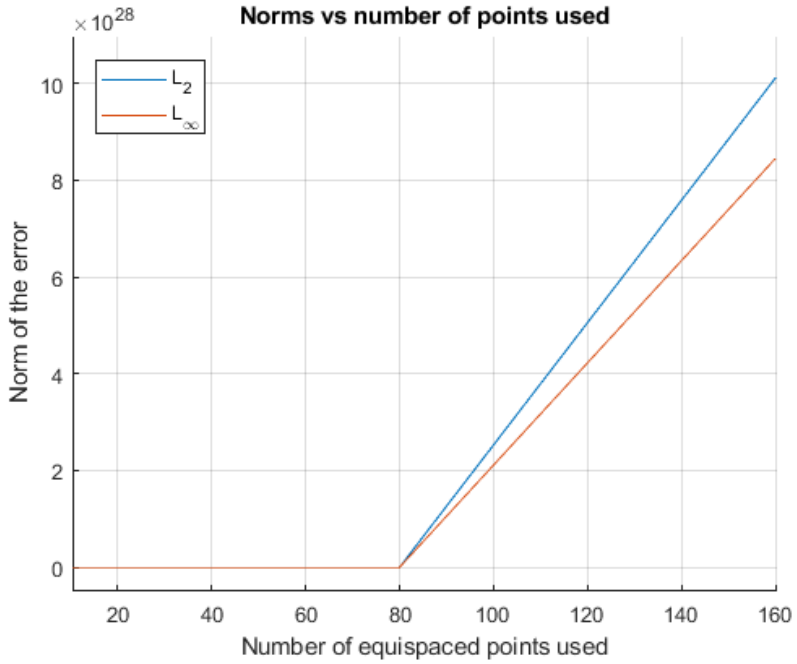




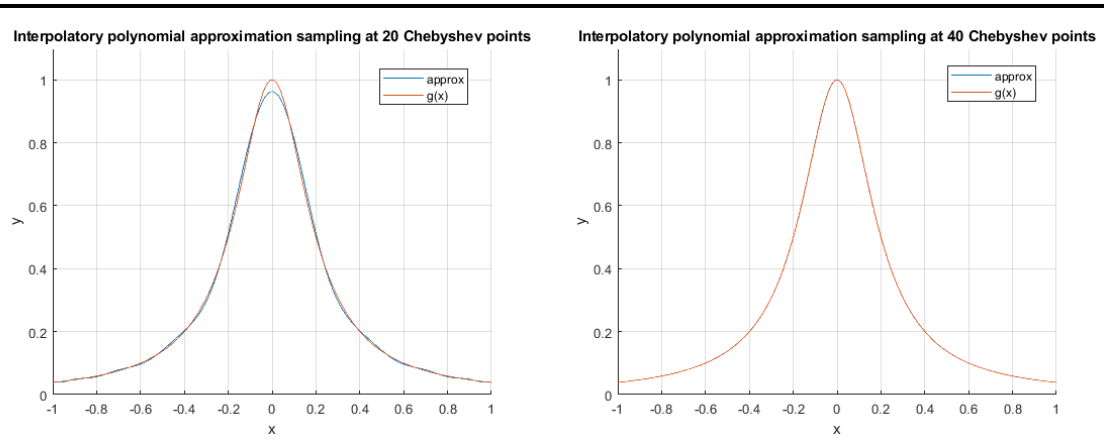


Starting from 40 equispaced points, y limit of graph changed to -0.2 to 1.2 so the plots can be compared clearly with the true function g(x). Frequency and amplitude of oscillations of approximation plots near the endpoints of the standard interval increases as the number of equispaced points increases.

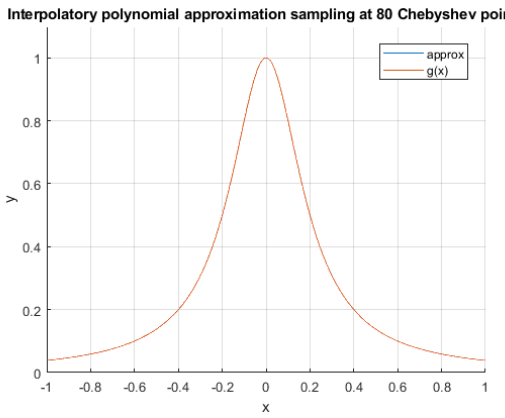
No. of points	20	40	80	160
L_2	56.2386	6.2097×10^4	2.1183×10^{11}	1.0112×10^{29}
L_∞	8.5756	1.4428×10^4	7.3032×10^{10}	8.4497×10^{28}

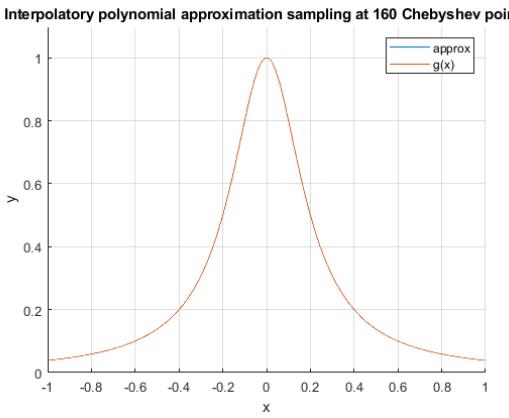


c) Repeat part (b). However, this time sample g(x) at Chebyshev points instead of equispaced points. What happens now? [40%]

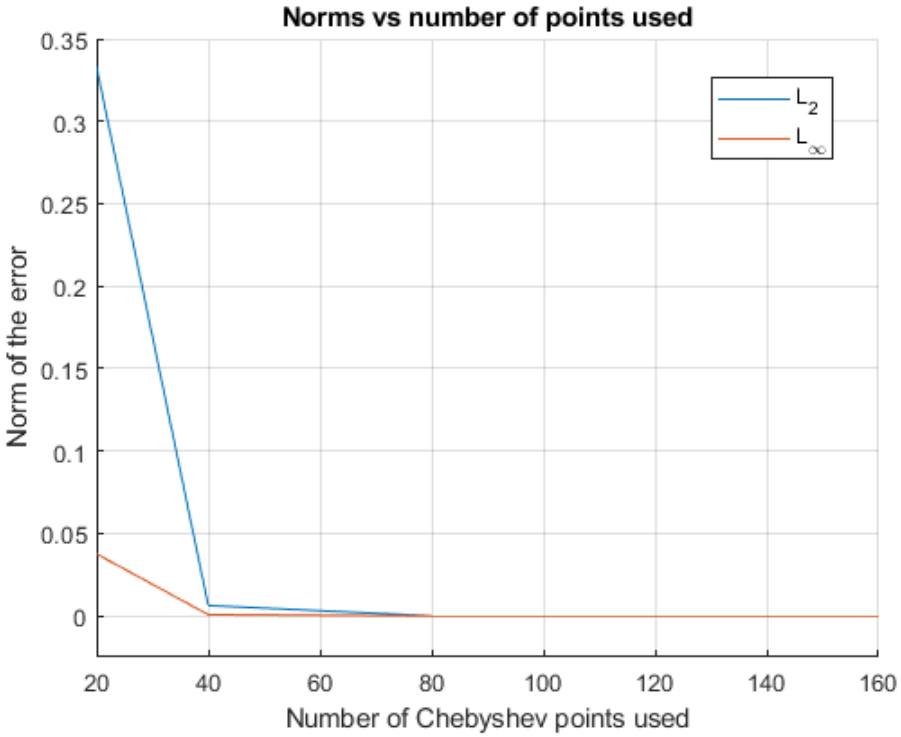


Results using Chebyshev points are much more accurate compared to sampling g(x) at equispaced points as shown. Approximation plot at 40 Chebyshev points onwards is pretty much the same as the true function g(x).





No. of points	20	40	80	160
L_2	0.3331	6.2557×10^{-3}	2.2114×10^{-6}	2.7654×10^{-13}
L_∞	0.0376	7.0702×10^{-4}	2.4994×10^{-7}	3.3307×10^{-14}



Norm of the error using Chebyshev points decreases exponentially and remains close to zero after it passes 80 points, explains why approximation is almost true g(x) if number of points is large. Norm of the error using equispaced points is the exact opposite, increasing exponentially with its values remaining almost zero from range 0 to 80 points, explains the huge increase in approximation value amplitude as number of points increases.

Q2) Please include your MATLAB codes here (you may use font size of 9)

2a)

```
x0 = 0; %initial x value
x1 = sqrt((exp(x0))/3); %initial g(x) value
i=1;
while abs(x1 - x0) > 0.0001 %error
        z(i,1)=x0; %storing data points
        z(i,2)=x1;
    g = sqrt((exp(x1))/3); %g(x) points
    x0 = x1;
    x1 = g;
    i=i+1;
end
x=[3:0.1:5];
y=x; %y=x line
g2 = sqrt((exp(x))/3); %g(x)
```

2b)

```
q = 0;
p = (-3*pi)/2;
n = 2; %number of intervals
exact = (1/50)*(7*(exp((3*pi)/2)) - 1);
error = 1;

while error >= 0.01
%for n = 1:100
    h = (q - p)/n;
    for j=1:n+1
    x(j) = p + h*(j-1);
    f(j) = (cos(7*x(j)))/(exp(x(j)));
    end
    simpson = (h/3)*(2*sum(f) - f(1) - f(n+1) + 2*sum(f(2:2:n)));
    z(n)=simpson;
    error = 100*abs((simpson - exact)/exact);
n = n+2;
end
disp(n-2)
disp(simpson)
```

2c)

```
exact = (1/50)*(7*(exp((3*pi)/2)) - 1);
error = 1;
i = 0;
while error >= 0.01
    sum = 0;
    i = i+1;
    n = s(i).f ; %no. of matrix in the structure array
for j=1:i
    sum = sum + n(j,2)*((cos(7*(((-3*pi)/4)-((3*pi*n(j,3))/4))))/(exp(((-3*pi)/4)-((3*pi*n(j,3))/4))));
end
z(i)=((3*pi)/4)*sum;
ans = ((3*pi)/4)*sum;
error = 100*abs((ans - exact)/exact);
end
disp(i)
disp(ans)
```

Q3) Please include your MATLAB codes here (you may use font size of 9)

3a)

```
x = linspace(-1, 1, 1001);
x2 = linspace(-1, 1, 10);
y = 1./(1 + 25*(x2.^2));
y1 = 1./(1 + 25*(x.^2));
L=1;
sum = 0;
for i=1:1001
   sum = 0;
   for j=1:10
      L = 1;
      for k=1:10
      if j~=k
         L = L*((x(i)-x2(k))/(x2(j)-x2(k)));
      end
      end
      sum = sum + (y(j)*L);
   end
   y2(i) = sum;
end
error = y2 - y1;
l2 = norm(error, 2);
linf = norm(error, inf);
```

3b)

```
x = linspace(-1, 1, 1001);
x2 = linspace(-1, 1, 20); %can be 20, 40, 80, 160
y = 1./(1 + 25*(x2.^2));
y1 = 1./(1 + 25*(x.^2));
L=1;
sum = 0;
for i=1:1001
   sum = 0;
   for j=1:20
      L = 1;
      for k=1:20
      if j~=k
         L = L*((x(i)-x2(k))/(x2(j)-x2(k)));
      end
      end
      sum = sum + (y(j)*L);
   end
   y2(i) = sum;
end
error = y2 - y1;
l2 = norm(error, 2);
linf = norm(error, inf);
```

3c)

```
x = linspace(-1, 1, 1001);
x2=-cos(((2*(0:19) + 1)/(2*19 + 2))*pi); %can be 19, 39, 79, 159, because n-1 points for n intervals
y = 1./(1 + 25*(x2.^2));
y1 = 1./(1 + 25*(x.^2));
L=1;
sum = 0;
for i=1:1001
   sum = 0;
   for j=1:20
      L = 1;
      for k=1:20
      if j~=k
         L = L*((x(i)-x2(k))/(x2(j)-x2(k)));
      end
      end
      sum = sum + (y(j)*L);
   end
   y2(i) = sum;
end
error = y2 - y1;
l2 = norm(error, 2);
linf = norm(error, inf);
```