Background Information

There are some integrals, like $C(t)=\int_0^t \cos\left(x^2\right)dx$ and $S(t)=\int_0^t \sin\left(x^2\right)dx$, whose functions look like they should have an antiderivative which is an elementary function, but they don't. Integrals of this type are called **nonelementary integrals** and it is important to be able to solve them, because they are useful in a number of different fields [1].

Figure 1 shows a parametric plot of $S(t)$ versus $C(t)$. This curve is called an Euler spiral or clothoid [2]. The Euler spiral has the property that its curvature at any point is proportional to the distance along the spiral, measured from the origin. This means a driver can experience a smooth ride by turning the steering wheel with a constant speed, when transitioning from one direction to another direction [3].

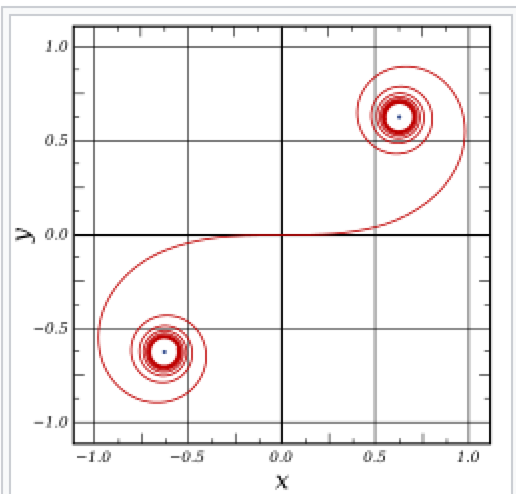

Figure 1 Euler Spiral

Another useful nonelementary integral is $\int e^{-x^2}dx,$ which is used in the definition of the normal probability distribution function, an important distribution in statistics and probability. If the function being integrated is written as

$$f(x)=\frac{e^{-x^2/2}}{\sqrt{2\pi}},$$

the graph of the function gives the bell-shaped curve shown in Figure 2. The integral of the function gives the probability that a statistic is observed below, above, or between values on the standard normal distribution. [4].

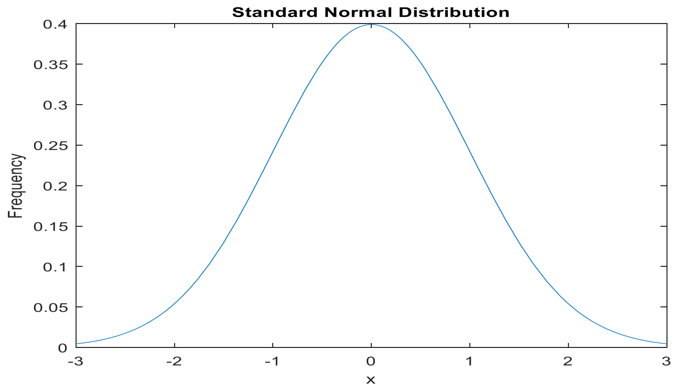

Figure 2 Graph of Standard Normal Distribution

Project Description

The goal of this project, is to evaluate $\int_0^1 e^{-x^2}\,dx$ using

- Simpson's Rule, and
- a Taylor polynomial to approximate e^{-x^2}.

For each method, you will be looking at how the error decreases when more intervals are used in Simpson's Rule, and when more terms are used in a Taylor polynomial. The exact value of the integral will be taken as the value calculated using the MATLAB command **integral**.

Learning Goals

By the end of the project, students should be able to

- write simple programs in MATLAB,
- write a technical paper describing the results of a project, and
- use their knowledge of MATLAB to solve scientific and mathematical problems in other classes.

Knowledge

This assignment will help students become familiar with

- using an error bound to estimate the error involved in integrating a function using Simpson’s Rule;
- estimating the error in approximating a function using a Taylor polynomial;
- integrating a Taylor polynomial, a function using Simpson’s Rule, and a function using a MATLAB commands; and
- the MATLAB programming language.

Skills

The assignment will help students develop computational and quantitative skills which are important in all STEM fields. Building computational skills helps develop

- quantitative self-efficacy,
- problem-solving skills, and
- logical thinking and reasoning skills.

Part I Introduction to MATLAB

Task

1. Register with MathWorks to take their free MATLAB Onramp course.
https://matlabacademy.mathworks.com/?s_tid=srchtitle

This is a web-based, interactive tutorial provided by the company that sells MATLAB. I suggest that you do all 15 tutorials, but if you don’t have the time, at least do lessons 1, 2, 3, 6, 7, 8, 9, and 13. The lessons are self-paced, and you can start and stop the lessons at any time. It takes about two hours to work through all of the lessons. When you are finished, upload a screen shot on Canvas showing the modules you have completed.

Part II Determine Error Bounds

Task

1. Review Simpson’s Rule

Simpson’s Rule is a method to numerically integrate a function where the function is approximated using parabolas. The formula for implementing Simpson’s Rule is

$$\int_a^b f(x)\,dx \approx S_n = \frac{\Delta x}{3}\left[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+\cdots+2f(x_{n-2})+4f(x_{n-1})+f(x_n)\right]$$

where n is the number of intervals and $\Delta x = \dfrac{b-a}{n}$. [4]

Note: n must be an even number.

2. Find Error Bound

The error bound E_S for Simpson's Rule is given by

$$|E_S| \le \frac{K(b-a)^5}{180n^4},$$

where $\left|f^{(4)}(x)\right| \le K$ for $a \le x \le b$ and $f^{(4)}(x)$ is the 4th derivative of $f(x)$. [4]

a) Find the 4th derivative of e^{-x^2} and then find the maximum value of the 4th derivative over the limits of integration.

b) Find n so that $|E_s| < 0.0001$.

3. Review Taylor Polynomials

The Taylor series of a function $f(x)$ that has a power series expansion at a, is of the form

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

$$= f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

a) Find the Taylor polynomial for $f(x) = e^{-x^2}$ at $a = 0$.

b) Integrate the Taylor polynomial.

c) Find the number of terms needed in a Taylor polynomial to have an error of less than 0.0001.

Since the Taylor polynomial is an alternating series, the Alternating Series Estimation Theorem can be used to determine the number of terms to include in the series [4].

If $s = \sum(-1)^{n-1}b_n$ is the sum of an alternating series that satisfies

- $b_{n+1} \le b_n$ and
- $\lim_{n\to\infty} b_n = 0$,

then

$$|R_n| = |s - s_n| \le b_{n+1}.$$

Part III Writing/Modifying Code

1. As a general rule, a MATLAB program will have five sections

- Documentation: Include the program name, the author, the date, a short description of what the program does, and what information needs to be provided to run the program.
- Input/Output: Input data needed to run the program can be included in the program, it can be loaded from an external file, or a prompt can be sent to the terminal requesting input from the user. If you are writing a function, state what information is passed out of the function back to the calling program.
- Computation/Algorithm: An algorithm is the step-by-step procedure needed to carry out the desired computation.
- Figures: Whenever possible, use figures as part of your output, they give information not apparent from an equation or table of data. Be sure all figures are properly labeled.
- Variable Output: State what information is returned to the user or written to a file.

2. Advice when writing programs:

- If a number is needed in more than one place in a program, you need to use a variable.
- Annotate all graphs completely, and give them meaningful names.
- Annotate all code completely. Use the percent sign (%) before any comment. The % sign tells MATLAB to ignore anything on the line after that symbol.

3. Creating an algorithm

Step 1: Define the problem
Step 2: Outline the solution
Step 3: Develop a high-level algorithm (pseudo code)
Step 4: Refine the algorithm by adding more detail
Step 5: Code the algorithm on computer
Step 6: Test the algorithm

4. Modifying code

Sometimes it is possible to find code that solves a problem similar to the one you are trying to solve. When that is the case, be sure to go through the code carefully to be sure you fully understand the program before trying to modify it to fit your specific needs.

MATLAB code to evaluate $\int_0^{\pi} \sin(x)\,dx$ using Simpson's Rule is given on the next page. Some parts of the code may not make sense to you, which is expected. The goal is to go through the program and understand what each line does, and figure out how to make changes to the code so you can use it to solve your problem.

If you do not understand what a MATAB command does, search online for information on that command, or use the **help** *command* feature of MATLAB.

```
% simpsonint.m
% Evaluates the integral of a function over an interval
% input: function f(x) = sin(x)
%           limits of integration: a – lower limit; b – upper limit
% output: the value of the integral

format long
clear
a=0;            % lower limit of integration
b=pi;
% enter the number of intervals from the keyboard
n=input('input number of intervals (n must be an even number) ->');

h = (b-a)/n;

fa = sin(a); % f(a)- function value at left endpoint
fb = sin(b);

fsum=0;  % the area of each interval will be stored in fsum. Start with 0 area

for i = 2:2:n;  % all 4*f(a+nh) terms to f(b) h = (1,3,5,7,...,n-1)
   x = (a+(i-1)*h);
   fx = sin(x);
   fsum = fsum + 4*fx;
end

for i = 3:2:n;  % all 2*f(a+nh) terms to f(b) h = (2,4,6,...,n-2)
   x = (a+(i-1)*h);
   fx = sin(x);
   fsum = fsum + 2*fx;
end

result = (h/3)*(fa+fb+fsum)
```

5. Modify the code by changing the necessary lines of the program to do the following:
   - Change the function from sin(x) to e^{-x^2}. **Note**: e^{x^2} in MATLAB is written as exp(x.^2).
   - Change the limits of integration to match the limits for $\int_0^1 e^{-x^2}dx$.

6. Finish commenting the code.

Part IV Use MATLAB to evaluate the integral, $\int_0^1 e^{-x^2}dx.$

Task
1. Start MATLAB.
2. Create a directory on your thumb drive or the desktop, where you can store your MATLAB files.
3. Use the MATLAB **addpath** command to tell MATLAB where your files are stored. For example, if your folder on the E: drive is *matlabfiles*, the command you need to type in the command window is addpath('E:/matlabfiles'). If your folder is on the desktop, use addpath('U:\Desktop\matlabfiles').

Note: When only one or two lines of code are needed to solve a problem, commands can be typed directly into the Command Window. For longer programs, a script or function file should be used and commands entered into the Editor Window.

4. The **integral** command is a MATLAB command used to numerically integrate a function. Before using **integral**, the function to be integrated needs to be specified. This is done using an *anonymous function*, which is written as

```
fun = @ (x) sin(x)
```

where in this example, *fun* is the name given to the function sin(x).

The **integral** command is written as

```
q = integral(fun,xmin,xmax)
```

where *q* is the value of the integral, *fun* is the function to be integrated, *xmin* is the lower limit of integration, and *xmax* is the upper limit of integration.

5. Create a new MATLAB script and add the lines of code from the previous task to calculate the value of the integral using the **integral** command. Be sure to assign values to *xmin* and *xmax*. Save your program.

6. Run your program by clicking on the green arrow icon in the upper task bar. Assume the result is the exact value of the integral. It will be the reference value to determine the error when using Simpson's Rule and Taylor polynomials. Save a copy of the commands you used and the result in a WORD document.

7. Create a new MATLAB script and add the lines of code you modified in Part III using Simpson's Rule. Save the code in your *matlabfiles* folder.

8. Click on the green arrow to run the program. At the onscreen prompt, type in the number of intervals to be used (it has to be an even number), and record the result in a table in your WORD document. The table should have three columns (number of intervals, value of integral, error). You need to run the program several times using values of *n* less than and more than the value of *n* you predicted were needed from your error estimate.

9. To evaluate $\int_0^1 e^{-x^2}dx$ using a Taylor polynomial, it is first necessary to find a Taylor polynomial to approximate e^{-x^2}. The terms in a Taylor polynomial can be obtained using the MATLAB commands

```
syms x
expr = sin(x);
texpr = taylor(expr, 'Order', 8)
```

The **taylor** command computes the Taylor series for *expr* and returns all terms up to, but not including, order eight in the Taylor series for the function, $f(x)$, which in the example is sin(x).

10. Create a new script file and type in the lines of code to find a Taylor polynomial. Be sure to properly comment each of the above lines of code. If you don't know what a command does, use the '**help** command' in MATLAB or google that word along with the word MATLAB.

11. Find the integral of the Taylor polynomial using

```
int(texpr,var,a,b)
```

The **int** command computes the definite integral of *texpr* (the terms in the Taylor polynomial) with respect to the variable *var* from *a* to *b*. Add the **int** command to your script program in the

Editor Window. You need to replace *var* with the variable used in your function and be sure to specify the values of *a* and *b*. Also add a comment to this line of code.

12. Run your program with different numbers of terms in the polynomial. Use polynomials with less than and more than the predicted number of terms you found using the Alternating Series Estimation Theorem. Record your results in a new table.

Part V Report

Your report must be typed with a 12-point Times New Roman font, in a technical report format. Two people may work together to produce one report.

The format of your report should be:

1. Introduction
   Describe the general type of integral being evaluated, the specific integral you solved, what methods were used to solve the integral, and why it is necessary to sometimes solve integrals numerically.
2. Analysis
   Explain the two different ways used to evaluate the integral and show how the error is calculated for each method.
   2.1 Simpson's Rule – Explain how Simpson's rule is used to evaluate integrals, and show how the error is estimated based on the number of intervals used. Explain the algorithm that was used to implement Simpson's Rule.
   2.2 Taylor Polynomials – Give the definition of a Taylor series, how Taylor polynomials are used to approximate functions, and how the Alternating Series Estimation Theorem is used to determine the number of terms needed in the polynomial.
3. Discussion and Results
   The results should be written so that someone reading the text will know what you have done and what you found out. When stating your results, refer to the tables you created when using Simpson's Rule and Taylor polynomials. Each table should be numbered sequentially for easy reference in the text. Each table must have a caption that is located above the table.

   The discussion is where you explain your results in detail, speculating on trends, possible causes, and conclusions. Try to present the principles, relationships, and generalizations shown by the Results. Explain any unexpected results. State what conclusions can be drawn from the results. Comment on how your predicted error compares to the actual error. If it is not what you expected, explain why.

4. Conclusion
   The two main parts of a conclusion are to summarize the most important findings from your work, and to introduce new ideas on how your work can be extended in a future project.
5. References
6. Appendix
   List the MATLAB code (properly commented) to find the exact value of the integral, and the approximate value using Simpson's Rule and Taylor polynomials.

Grading

- Part I – Introduction to MATLAB: 20 points for completing lessons 1, 2, 3, 6, 7, 8, 9, 13 from the MATLAB Onramp course.
- Part V – Report: 100 points

References and Resources

1. Nonelementary Integral, Wikipedia
   https://en.wikipedia.org/wiki/Nonelementary_integral
2. Fresnel integral spiral
   https://en.wikipedia.org/wiki/Fresnel_integral
3. The Clothoid
   https://pwayblog.com/2016/07/03/the-clothoid/
4. Normal Distribution
   https://en.wikipedia.org/wiki/Normal_distribution
5. Stewart, James, Calculus Early Transcendentals, 7th ed., Brooks/Cole Cengage Learning, 2012