Name: _____

1. (45 points) Evaluate by **hand** the **definite integrals** below. Subsequently, **approximate** them by using the **composite Trapezoidal** rule $T_n(f)$ for $n = 2, 4, 8, 16, 32$ and 64, where $h = (b - a)/n$.

   (a) (15 points) $I = \int_0^1 (3x + 1)\, dx$

   (b) (15 points) $I = \int_0^1 x e^{-x^2}\, dx$

   (c) (15 points) $I = \int_0^{2\pi} (\cos(x) + 1)\, dx$

   To do so, write a MATLAB function (or in **any** other programming language) stored as `trap.m` whose first line should read

   ```
   function [ y ] = trap( f, a, b, n )
   ```

   Include a copy of your code. For each of the cases above make a table with columns:

   - column 1: n
   - column 2: h
   - column 3: T_n (this is the approximation, i.e., the output `y`)
   - column 4: |error| (this is the **absolute** error)
   - column 5: $|\text{error}|/h^2$

   Are the numbers in the last column **converging**, and if so, **what does this mean**? Specifically, comment on the behavior of the error for (a) and (b). If your code is correct, you will notice that for (c) the last column is **not** converging, and that the approximation is **very accurate**. Can you explain **briefly** why?

Solution: The main driver and the MATLAB function `trap.m` are given below:

```
clearvars; close all; clc; format long;

   n = 6;
   a = 0; b = 1;      % Interval: [a,b]

func = @(x) 3*x + 1; % Integrand f(x).
Ifex = 2.5;          % Exact value.

fprintf('n     h   T_{n}  |error|                |error|/h^2 \n  ');
fprintf('-------------------------------------------------------\n ...
    ');
for i = 1:n
    N = 2^i;                   % Note that N=2^1, 2^2, 2^4 etc.
    y = trap( func, a, b, N ); % Call the function.
    h = ( b - a ) / N;         % Lattice spacing h.
  err = abs( Ifex - y );       % Absolute error.
fprintf ('%d   %e   %e   %e   %e\n  ', N, h, y, err, err/h^2  );
end
```

```
function [ y ] = trap( f, a, b, N )

    h = ( b - a ) / N;     % Distance between mesh points.
    x = a:h:b;             % 1-D equidistant mesh.
 func = f(x);              % Evaluate the function on this mesh.
     y = h * sum(func) - ...
            0.5 * h * ( func(1) + func(end) );
end
```

Note that the driver corresponds to part (a), although one can obtain the results for parts (b) and (c) too by making minor changes therein. Subsequently, the respective numerical results for the cases (a)-(c) are presented in the following:

(a) $I = \int_0^1 (3x+1)\, dx = 2.5.$

n	h	T_{n}	|error|	|error|/h^2
2	5.000000e-01	2.500000e+00	0.000000e+00	0.000000e+00
4	2.500000e-01	2.500000e+00	0.000000e+00	0.000000e+00
8	1.250000e-01	2.500000e+00	0.000000e+00	0.000000e+00
16	6.250000e-02	2.500000e+00	0.000000e+00	0.000000e+00
32	3.125000e-02	2.500000e+00	0.000000e+00	0.000000e+00
64	1.562500e-02	2.500000e+00	0.000000e+00	0.000000e+00

(b) $I = \int_0^1 xe^{-x^2}\,dx = \dfrac{e-1}{2e} \approx 0.31606027941427883.$

n	h	T_{n}	|error|	|error|/h^2
2	5.000000e-01	2.866701e-01	2.939022e-02	1.175609e-01
4	2.500000e-01	3.088826e-01	7.177655e-03	1.148425e-01
8	1.250000e-01	3.142759e-01	1.784387e-03	1.142008e-01
16	6.250000e-02	3.156148e-01	4.454786e-04	1.140425e-01
32	3.125000e-02	3.159489e-01	1.113311e-04	1.140031e-01
64	1.562500e-02	3.160324e-01	2.783038e-05	1.139932e-01

(c) $I = \int_0^{2\pi} (\cos(x) + 1)\,dx = 2\pi.$

n	h	T_{n}	|error|	|error|/h^2
2	3.141593e+00	6.283185e+00	0.000000e+00	0.000000e+00
4	1.570796e+00	6.283185e+00	0.000000e+00	0.000000e+00
8	7.853982e-01	6.283185e+00	0.000000e+00	0.000000e+00
16	3.926991e-01	6.283185e+00	0.000000e+00	0.000000e+00
32	1.963495e-01	6.283185e+00	1.776357e-15	4.607554e-14
64	9.817477e-02	6.283185e+00	1.776357e-15	1.843022e-13

In (a) we expect the Trapezoidal method to perform well since $f''(x) = 0$, so the method **is exact** (recall the error for the composite Trapezoidal method). For (b), one sees that $|\text{error}|/h^2$ is approaching a constant which indicates the expected (second order) convergence in this example. For (c), even though $f''(x) \neq 0$ we see that (except for roundoff when $n = 32$ and $n = 64$) the method **is** again **exact**. This is due to the fact that the Trapezoidal method is particularly accurate when the integrand is a *periodic function* which is the case here.

2. (25 points) Consider the basic **Simpson's rule**:

$$S_2(f) = \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right], \quad \text{with} \quad h = \frac{b-a}{2}. \tag{1}$$

This approximation satisfies:

$$E(f) = -\frac{f^{(4)}(\eta)}{90}h^5, \tag{2}$$

for some $\eta \in [a, b]$ which implies that Simpson's rule is **exact** if $f(x)$ is a polynomial of degree <= 3, i.e., $p_n(x)$ for $0 \leq n \leq 3$.

(a) (15 points) **Derive** Simpson's rule, i.e., Eq. (1).

(b) (10 points) Consider $f(x) = x^3$. Then, find I_{Simp} (**by hand** !) which approximates $I_f = \int_a^b f(x)\,dx$. Is the answer exact? **Justify your answer**.

Solution:

(a) As per our discussion about Simpson's method in class, we consider interpolation at the interval ends **and** the midpoint by using a **quadratic** interpolant. This way, $n = 2$, and the corresponding abscissae are $x_0 = a$, $x_1 = (a + b)/2$ and $x_2 = b$. Next, we form the **three** Lagrange polynomials:

$$L_0(x) = \frac{\left(x - \frac{a+b}{2}\right)(x - b)}{\left(a - \frac{a+b}{2}\right)(a - b)} \Rightarrow L_0(x) = \frac{(a + b - 2x)(b - x)}{(a - b)^2},$$
$$L_1(x) = \frac{(x - a)(x - b)}{\left(\frac{a+b}{2} - a\right)\left(\frac{a+b}{2} - b\right)} \Rightarrow L_1(x) = \frac{4(a - x)(x - b)}{(a - b)^2},$$
$$L_2(x) = \frac{(x - a)\left(x - \frac{a+b}{2}\right)}{(b - a)\left(b - \frac{a+b}{2}\right)} \Rightarrow L_2(x) = \frac{(a + b - 2x)(a - x)}{(a - b)^2},$$

and the corresponding **weights** given by $a_j = \int_a^b L_j(x)\,dx$ read:

$$a_0 = \frac{b - a}{6},$$
$$a_1 = \frac{2(b - a)}{3},$$
$$a_2 = \frac{b - a}{6}.$$

Finally we have:

$$S_2(f) = \sum_{j=0}^{2} a_j f(x_j) = \frac{b - a}{6} f(a) + \frac{2(b - a)}{3} f\left(\frac{a + b}{2}\right) + \frac{b - a}{6} f(b),$$

or, upon collecting terms

$$\boxed{S_2(f) = \frac{h}{3}\left[f(a) + 4f\left(\frac{a + b}{2}\right) + f(b)\right]},$$

with $h = (b - a)/2$.

(b) For the function given, Simpson's rule yields

$$\begin{aligned} S_2(f) &= \frac{h}{3}\left[a^3 + 4\left(\frac{a+b}{2}\right)^3 + b^3\right] \\ &= \frac{b-a}{6}\left[a^3 + \frac{1}{2}\left(a^3 + 3a^2b + 3ab^2 + b^3\right) + b^3\right] \\ &= \frac{1}{4}\left(b^4 - a^4\right), \end{aligned}$$

after carrying out the algebra. Note that using Simpson's rule we obtain the **exact** result. This is expected since the error term for $S_2(f)$ given by Eq. (2) involves the **fourth derivative** of $f(x)$, which for x^3 **is identically zero**. This clearly shows that, while Simpson's rule corresponds to integration of a 2nd order in x interpolant, $S_2(f)$ is **exact for all polynomials of degree three or less**.

3. (45 points) Repeat the numerical evaluation of the integrals in parts (a)-(c) of Question 1 using the **composite Simpson's** rule $S_n(f)$ for $n = 2, 4, 8, 16, 32$ and 64, where $h = (b-a)/n$. To do so, write a MATLAB function (or in **any** other programming language) stored as `simp.m` whose first line should read

```
function [ y ] = simp( f, a, b, n )
```

Similary as in Question 1, and for each case, make a table with the same format as in Question 1 except of column 5 in which you must compute $|\text{error}|/h^4$. Include a copy of your code and compare your results with the **composite Trapezoidal** rule.

Solution: The main driver and the MATLAB function `trap.m` are given below:

```
1
2  clearvars; close all; clc; format long;
3
4     n = 6;
5     a = 0; b = 1;      % Interval: [a,b]
6
7  func = @(x) 3*x + 1; % Integrand f(x).
8  Ifex = 2.5;          % Exact value.
9
10 fprintf('n     h   S_{n}   |error|                |error|/h^4 \n  ');
11 fprintf('-----------------------------------------------------\n ...
       ');
12 for i = 1:n
```

```
    N = 2^i;                    % Note that N=2^1, 2^2, 2^4 etc.
    y = simp( func, a, b, N ); % Call the function.
    h = ( b - a ) / N;          % Lattice spacing h.
  err = abs( Ifex - y );        % Absolute error.
fprintf ('%d   %e   %e   %e   %e\n  ', N, h, y, err, err/h^2  );
end
```

```
function y = simp( f, a, b, N )

    h = ( b - a ) / N;     % Distance between mesh points.
    x = a:h:b;             % 1-D equidistant mesh.
 func = f(x);              % Evaluate the function on this mesh.
    y = h / 3 * ( func(1) + 4 * sum(func(2:2:end)) ...
                + 2 * sum(func(3:2:end-2)) + func(end) );

end
```

Furthermore, the respective numerical results for the cases (a)-(c) of Question 1 are presented in the following:

(a) $I = \int_0^1 (3x+1)\,dx = 2.5.$

n	h	S_{n}	|error|	|error|/h^4
2	5.000000e-01	2.500000e+00	0.000000e+00	0.000000e+00
4	2.500000e-01	2.500000e+00	0.000000e+00	0.000000e+00
8	1.250000e-01	2.500000e+00	0.000000e+00	0.000000e+00
16	6.250000e-02	2.500000e+00	0.000000e+00	0.000000e+00
32	3.125000e-02	2.500000e+00	0.000000e+00	0.000000e+00
64	1.562500e-02	2.500000e+00	0.000000e+00	0.000000e+00

(b) $I = \int_0^1 xe^{-x^2}\,dx = \dfrac{e-1}{2e} \approx 0.31606027941427883.$

n	h	S_{n}	|error|	|error|/h^4
2	5.000000e-01	3.209135e-01	4.853222e-03	7.765155e-02
4	2.500000e-01	3.162868e-01	2.265340e-04	5.799271e-02
8	1.250000e-01	3.160736e-01	1.336932e-05	5.476072e-02
16	6.250000e-02	3.160611e-01	8.241996e-07	5.401475e-02
32	3.125000e-02	3.160603e-01	5.133797e-08	5.383177e-02
64	1.562500e-02	3.160603e-01	3.205910e-09	5.378624e-02

(c) $I = \int_0^{2\pi} (\cos(x) + 1)\, dx = 2\pi$.

n	h	S_{n}	|error|	|error|/h^4
2	3.141593e+00	4.188790e+00	2.094395e+00	2.150102e-02
4	1.570796e+00	6.283185e+00	0.000000e+00	0.000000e+00
8	7.853982e-01	6.283185e+00	0.000000e+00	0.000000e+00
16	3.926991e-01	6.283185e+00	1.776357e-15	7.469485e-14
32	1.963495e-01	6.283185e+00	0.000000e+00	0.000000e+00
64	9.817477e-02	6.283185e+00	0.000000e+00	0.000000e+00

For the integral of part (a), we expect similar behavior with the Trapezoidal rule. Indeed, Simpson's method **is exact** since its error involves $f^{(4)}(x)$ which is 0 for the linear in x integrand given. On the other hand, and as per the integral of part (b), Simpson's method produces much more accurate results compared to the ones obtained from the Trapezoidal rule (see, the errors in column 4 therein). Furthermore, the normalized error of column 5 gradually converges (as is the case with the Trapezoidal rule although to a different number). Finally, Simpson's rule produces accurate results for the periodic integrand of part (c) but it should be noted that the Trapezoidal rule is quite robust and performs even better than the Simpson's rule. Thus, and for periodic integrands at hand, the Trapezoidal rule is preferred!

4. (15 points) Besides the Trapezoidal and Simpson's rules, there exists another approximation to $\int_a^b f(x)\, dx$ by replacing $f(x)$ with the **constant** $f((a+b)/2)$ on $[a, b]$.

   (a) (5 points) Show that the aforementioned process leads to the numerical integration formula

   $$M_1(f) = (b-a)\, f\left(\frac{a+b}{2}\right).$$

   This is called the **basic Midpoint rule**.

   (b) (10 points) Derive the **composite version** of the Midpoint rule

   $$M_n(f) = h\left[f(x_1) + f(x_2) + \ldots + f(x_n)\right],$$

   where

   $$h = (b-a)/n, \quad x_j = a + (j - 1/2)\, h, \quad j = 1, \ldots, n.$$

Solution:

(a) We replace $f(x)$ with $f((a+b)/2)$ and immediately obtain

$$I(f) = \int_a^b f(x)\,dx \approx \int_a^b f\left(\frac{a+b}{2}\right) dx \approx \boxed{\frac{a+b}{2}\,(b-a) \equiv M_1(f)}.$$

(b) Upon introducing $x_j = a + (j-1/2)h$ being the jth **midpoint** of the jth sub-interval $[a+(j-1)\,h, a+j\,h]$, with $j = 1, \ldots, n$, we have

$$\begin{aligned} I(f) &= \int_a^b f(x)\,dx \\ &= \int_a^{a+h} f(x)\,dx + \int_{a+h}^{a+2h} f(x)\,dx + \ldots + \int_{a+(n-1)h}^{a+nh} f(x)\,dx \\ &\approx \int_a^{a+h} f\left(\frac{2a+h}{2}\right) dx + \int_{a+h}^{a+2h} f\left(\frac{2a+3h}{2}\right) dx + \ldots \\ &+ \int_{a+(n-1)h}^{a+nh} f\left(\frac{2a+(2n-1)h}{2}\right) dx \\ &\approx f(x_1)\int_a^{a+h} dx + f(x_2)\int_{a+h}^{a+2h} dx + \ldots + f(x_n)\int_{a+(n-1)h}^{a+nh} dx \Rightarrow \end{aligned}$$

$$\boxed{I(f) \approx M_n(f) \equiv h\sum_{j=1}^{n} f(x_j)}.$$

5. (20 points) Consider the integral of part (b) of Question 1:

$$\int_0^1 xe^{-x^2}\,dx.$$

and the **error formulas** for the Trapezoidal and Simpson's rules:

$$E_n^T(f) \equiv \int_a^b f(x)\,dx - T_n(f) = -\frac{h^2(b-a)}{12} f''(c_n),$$
$$E_n^S(f) \equiv \int_a^b f(x)\,dx - S_n(f) = -\frac{h^4(b-a)}{180} f^{(4)}(c_n),$$

respectively, where $c_n \in [a,b]$. Then, find the value of n (number of sub-intervals) to be chosen in order to ensure

$$|E_n^T(f)| \le 10^{-4}, \quad |E_n^S(f)| \le 10^{-4}.$$

Compare your theoretical results with the numerical findings of Question 1.

Hint: You should find the extrema of $f''(x)$ and $f^{(4)}(x)$ and determine their maximum values. To do so, graphing $f''(x)$ and $f^{(4)}(x)$ might help and you could use Newton's method to find their extrema!

Solution:

(a) Let us work first with the error formula for the Trapezoidal method. Direct computation of $f''(x)$ yields

$$f''(x) = 2e^{-x^2}x\left(2x^2 - 3\right).$$

Our goal is to maximize $f''(x)$ and to that effect, we calculate the zeros of

$$f'''(x) = e^{-x^2}\left(-8x^4 + 24x^2 - 6\right),$$

via Newton's method. Indeed, and for $c_n \in [0,1]$ we find that $c_n^{1,2} \approx \pm 0.52465$ such that $|f''(\pm 0.52465)| \approx 1.952$ (both roots work since we consider the absolute value but the one with the "+" is accepted since $c_n \in [0,1]$!). This way, and from the error bound we have

$$\begin{aligned} |E_n^T(f)| \leq 10^{-4} \Rightarrow \frac{h^2}{12}|f''(c_n)| \leq 10^{-4} \Rightarrow \\ h^2 \cdot 0.16265 \leq 10^{-4} \Rightarrow \\ \frac{1}{n^2} \leq 0.000614822 \Rightarrow \\ n^2 \geq 1626.49 \Rightarrow \\ n \geq 40.3297 \Rightarrow \boxed{n \geq 40}. \end{aligned}$$

(b) Similarly, for Simpson's method, we compute $f^{(4)}(x)$ directly yielding

$$f^{(4)}(x) = 4xe^{-x^2}\left(4x^4 - 20x^2 + 15\right),$$

which is extremized at the zeros of

$$f^{(5)} = -4e^{-x^2}\left(8x^6 - 60x^4 + 90x^2 - 15\right).$$

Upon using graphing $f^{(4)}$ and using Newton's method on $f^{(5)}$ we find $c_n \approx 0.4361$ such that $|f^{(4)}| \approx 16.357$. Similarly, from the error bound for Simp-

son's method we have

$$|E_n^S(f)| \leq 10^{-4} \Rightarrow \frac{h^4}{180}|f^{(4)}(c_n)| \leq 10^{-4} \Rightarrow$$
$$h^4 \cdot 0.090872 \leq 10^{-4} \Rightarrow$$
$$\frac{1}{n^4} \leq 0.0011 \Rightarrow$$
$$n^4 \geq 908.72 \Rightarrow$$
$$n \geq 5.49044 \Rightarrow \boxed{n \geq 6}.$$

Upon comparing the theoretical results with the numerical findings, we see a perfect agreement. Indeed, for the Trapezoidal method (although we used sub-intervals of powers of 2) for $n = 32$ the (absolute) error is $\approx 1.1 \times 10^{-4}$ but for $n = 64$ it dropped down to $\approx 2.8 \times 10^{-5}$. In a similar vein, and as far as the Simpson's method is concerned, we see that for $n = 4$ the error is $\approx 2.26 \times 10^{-4}$ whereas for $n = 8$ it dropped down to $\approx 1.337 \times 10^{-5}$. You should try your codes for the theoretically predicted n's to convince yourself!

6. (20 points) Use the **method of undetermined coefficients** to derive the approximation for the first derivative of a function $f(x)$ at x_0:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h} + O(h^2).$$

Study the error in the above formula and determine it in terms of $f(x)$ and its derivatives. Then, if $f(x) = e^x$ and $x_0 = 1/2$ write a MATLAB script which does the following: approximates the first derivative of $f(x)$ given using the above formula and shows in a **log-log** plot the **absolute error** by halving h at each step, i.e., use $h = 0.1$, $h = 0.05$, $h = 0.025$, $h = 0.0125$, and so on. In addition, the script should plot the theoretical prediction for the error on top of the previous graph in order to compare the findings. Finally, show that your approximation has $O(h^2)$ truncation error, i.e., the order of accuracy is 2. When your approximation starts getting worse and **why does this happen**?

Hints: Note that if error $\approx Ch^q$ where q is the order of accuracy, then $\log(\text{error}) \approx \log C + q \log h$. Therefore in the log-log plot, you must determine the slope of the straight line obtained therein which eventually is the value of q. Recall that for a straight line of the form of $y = b + ax$, two points can be selected on the x-axis, e.g., x_1 and x_2 and we can find the corresponding ordinates $y_1 = y(x_1)$ and $y_2 = y(x_2)$. Then, we obtain the slope via

$$a = \frac{y_2 - y_1}{x_2 - x_1}.$$

Solution: At first, we write

$$f'(x_0) \approx D_h^{(1)} f(x_0) \equiv Af(x_0 + h) + Bf(x_0 - h), \tag{3}$$

where A and B are coefficients to be determined. Then, we consider the Taylor polynomial approximations:

$$f(x_0 - h) \approx f(x_0) - hf'(x_0) + \frac{h^2}{2} f''(x_0) - \frac{h^3}{6} f'''(x_0) + \frac{h^4}{24} f^{(4)}(x_0),$$
$$f(x_0 + h) \approx f(x_0) + hf'(x_0) + \frac{h^2}{2} f''(x_0) + \frac{h^3}{6} f'''(x_0) + \frac{h^4}{24} f^{(4)}(x_0),$$

and upon plugging them into Eq. (3) we arrive at

$$D_h^{(1)} f(x_0) \approx (A + B) f(x_0) + h (A - B) f'(x_0) + \frac{h^2}{2} (A + B) f''(x_0) + \frac{h^3}{6} (A - B) f^{(3)}(x_0)$$
$$+ \frac{h^4}{24} f^{(4)}(x_0) (A + B). \tag{4}$$

Then, we require

$$A + B = 0 : \text{coefficient of } f(x_0),$$
$$h(A - B) = 1 : \text{coefficient of } f'(x_0),$$

and thus obtain $A = 1/(2h)$ and $B = -A$. This way, the approximation of the first derivative from Eq. (3) becomes

$$\boxed{f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h} + O(h^2)}.$$

As for the error, and based on the values of A and B we obtained, the first non-zero term in Eq. (4) becomes

$$\frac{h^2}{6} f'''(x_0) = \mathcal{O}(h^2), \tag{5}$$

which signals that the approximation is **second-order** accurate!

The MATLAB script that computes the first derivative of e^x at $x_0 = 1/2$ is given next:

```
clearvars; close all; clc; format long;

% The function and its derivative:
   f = @(x) exp(x);
  fx = @(x) exp(x);

```

```
% Set-up:
  N = 14;  % How many halvations?
  h = 0.1; % Initial step-size.
 x0 = 1/2; % Point given.

% Pre-allocate arrays:
  fapprox = zeros(N,1);
   hstore = zeros(N,1);
    error = zeros(N,1);

% Do the task:
for i = 1:N
      hstore(i) = h;                                  % Store h.
     fapprox(i) = 0.5 * (f(x0+h) - f(x0-h) ) / h; % Approximate f'.
              h = h / 2;                              % Halve h.
       error(i) = abs(fx(x0)-fapprox(i));          % Absolute error.
end

% Plot the outcome:
figure;
set(gca,'FontSize',24,'Fontname','Times');
loglog(hstore,error,'-k','LineWidth',2);
grid on;
xlabel('$h$','Interpreter','latex');
ylabel('$\textrm  {error}$','Interpreter','latex');

% Estimate the truncation error by fitting:
log(error(5)/error(10))/log(hstore(5)/hstore(10))
% Alternative way: Use polynomial fitting
% [ q, s ] = polyfit(log(hstore),log(error),1);
```

The output of the above script is shown in Fig. 1. In particular, the top panel depicts the error as a function of h in a log-log scale. The slope q (or, the order of accuracy) can be determined either from the hint given or by using the built-in command `polyfit`. The slope we obtain is $q \approx 1.999998939009837$ which is in good agreement with the expected value of 2, i.e., the approximation is second-order accurate! Furthermore, the red open circles in the top panel correspond to the theoretical prediction given by Eq. (5). The perfect agreement can be clearly discerned from the figure.

However, if we keep decreasing h being equivalent to increasing `N` in the MATLAB script (you can experiment with this!), then, we obtain the result shown in the bottom panel of Fig. 1. The reason that this happens is due to roundoff error which eventually dominates the truncation error leading to **catastrophic phenomena**. This phenomenon is shown in the bottom panel of Fig. 1 where the error starts increasing as h decreases.

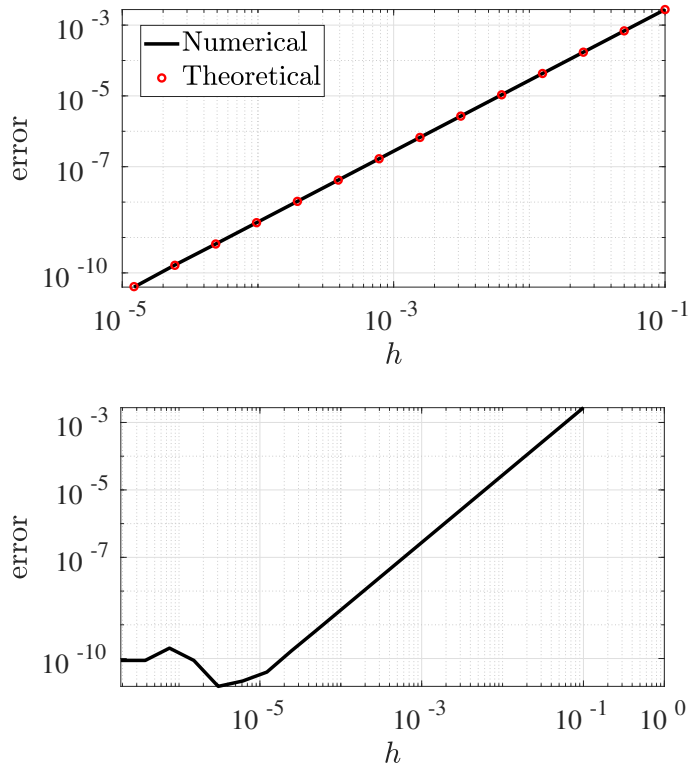

Figure 1: The log-log plot of the absolute error corresponding to the approximation of the first derivative of e^x as a function of h. The red open circles in the top panel correspond to the theoretical prediction of the error dictated by Eq. (5). On the other hand, and in the bottom panel, the approximation becomes worse past a value of h!

Date: February 22, 2020
Mathematics Department, California Polytechnic State University, San Luis Obispo, CA 93407-0403, USA
Email address: echarala@calpoly.edu