PROJECT REPORT

HOMEWORK 1

Written and Prepared by:
Roxanne Lutz
University of New Mexico
September 6, 2025

TEXTBOOK PROBLEMS

See following pages for the handwritten, worked problems.

HW 1 : 1.1 : ~~8~~, ~~10~~, ~~17~~, ~~19~~ checked ✓
~~Compact 1~~
~~compact 2~~
Ask to be sure this is correct.

Roxanne Lutz
MATH 375
9/2/25

1.1.2 Show following have @least one solution in given intervals.

(a) $\sqrt{x} - \cos x = 0$, $[0,1]$.

let $x = 0 \rightarrow \sqrt{0} - \cos(0) = f(0) = -1 < 0$

let $x = 1 \rightarrow \sqrt{1} - \cos(1) = f(1) = 1 > 0$.

There are no values of $x \in [0,1]$ st $f(x)$ is discontinuous.

The function $f(x)$ is continuous on $[0,1]$; and since $f(0) < 0 < f(1)$, there is a number $c \in (0,1)$ such that $\sqrt{c} - \cos c = 0$. (Intermediate Value Theorem) (IVT)

(b) $e^x - x^2 + 3x - 2 = 0$, $[0,1]$

let $x = 0 \rightarrow f(0) = e^0 - 0^2 + 3(0) - 2 = 1 - 2 = -1 < 0$.

let $x = 1 \rightarrow f(1) = e^1 - 1^2 + 3(1) - 2 = e - 1 + 3 - 2 = e - 1 + 1$
$= e > 0$

The function $f(x)$ is continuous on $[0,1]$; and since $f(0) < 0 < f(1)$, there is a number $c \in (0,1)$ such that $e^c - c^2 + 3c - 2 = 0$. (IVT)

There are no values $x \in [0,1]$ st $f(x)$ is discontinuous.

(c.) $-3\tan(2x)+x=0$, $[0,1]$.

let $x=0 \to f(0)=-3\tan(2(0))+0=-3\tan(0)=-3(0)=0$.

There is a solution → let $c=0$, $f(c)=0$, therefore we have found at least 1 solution on $[0,1]$.

(d.) $\ln x - x^2 + \frac{5}{2}x - 1 = 0$, $[\frac{1}{2}, 1]$.

let $x=\frac{1}{2} \to f(\frac{1}{2}) = \ln(\frac{1}{2}) - (\frac{1}{2})^2 + \frac{5}{2}(\frac{1}{2}) - 1$

$= \cancel{\ln(1)}^{0} - \ln(2) - \frac{1}{4} + \frac{5}{4} - \frac{4}{4}$

$= -\ln(2) + 0 = -\ln(2) < 0$.

let $x=1 \to f(1) = \cancel{\ln(1)}^{0} - 1^2 + \frac{5}{2}(1) - 1$

$= -1 + \frac{5}{2} - 1 = -2 + \frac{5}{2} = -\frac{4}{2} + \frac{5}{2} = \frac{1}{2} > 0$.

no values of $x \in [\frac{1}{2}, 1]$ s.t. $f(x)$ is discontinuous.

The function $f(x)$ is continuous on $[\frac{1}{2}, 1]$; and since $f(\frac{1}{2}) < 0 < f(1)$, there is a number $c \in (\frac{1}{2}, 1)$ such that $\ln(c) - c^2 + \frac{5}{2}c - 1 = 0$. (IVT)

1.1.10 Find third Taylor polynomial $P_3(x)$ for $f(x) = \sqrt{x+1}$, $x_0 = 0$. Approx: $\sqrt{0.5}$, $\sqrt{0.75}$, $\sqrt{1.25}$, $\sqrt{1.5}$ → find actual errors

$$P_3(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \frac{f'''(x_0)}{3!}(x-x_0)^3 + R_3(x)$$

$= \downarrow$

$$= f(0) + f'(0)(x) + \frac{f''(0)}{2!}(x)^2 + \frac{f'''(0)}{3!}(x)^3$$

→ Aside: $f'(x) = (x+1)^{1/2} \Rightarrow \frac{1}{2}(x+1)^{-1/2} = \frac{1}{2\sqrt{x+1}}$

$f''(x) = \frac{1}{2}(x+1)^{-1/2} \Rightarrow -\frac{1}{4}(x+1)^{-3/2} = -\frac{1}{4(x+1)^{3/2}}$

$f'''(x) = -\frac{1}{4}(x+1)^{-3/2} \Rightarrow \frac{3}{8}(x+1)^{-5/2} = \frac{3}{8(x+1)^{5/2}}$

→ $f(0) = \sqrt{0+1} = \sqrt{1} = 1$

$f'(0) = \frac{1}{2\sqrt{0+1}} = \frac{1}{2\sqrt{1}} = \frac{1}{2}$

$f''(0) = -\frac{1}{4(0+1)^{3/2}} = -\frac{1}{4\sqrt{1^3}} = -\frac{1}{4}$

$f'''(0) = \frac{3}{8(0+1)^{5/2}} = \frac{3}{8\sqrt{1^5}} = \frac{3}{8}$

- taking positive only $\sqrt{1}$ answers from this solution set.

→ S①: $P_3(x) = 1 + \frac{1}{2}x - \frac{1}{4}\left(\frac{1}{2}\right)x^2 + \frac{3}{8}\left(\frac{1}{6}\right)x^3$

$$\boxed{P_3(x) = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3}$$

$\sqrt{0.5} = \sqrt{x+1}$

$0.5 = x + 1$

$-0.5 = x$

$P_3(-0.5) = 1 - \frac{1}{2}\left(\frac{1}{2}\right) - \frac{1}{8}\left(\frac{1}{4}\right) - \frac{1}{16}\left(\frac{1}{8}\right)$

$= 1 - \frac{1}{4} - \frac{1}{32} - \frac{1}{128}$

$\boxed{P_3(-0.5) \approx 0.7109375}$

$f(-0.5) = \sqrt{(-0.5)+1} = \sqrt{0.5}$

$f(-0.5) \approx 0.7071068$

$\boxed{\text{Actual Error} \approx 3.831 \times 10^{-3}}$

$\sqrt{0.75} = \sqrt{x+1}$
$0.75 = x+1$
$-0.25 = x = -\frac{1}{4}$

$P_3(-0.25) = 1 - \frac{1}{2}\left(\frac{1}{4}\right) - \frac{1}{8}\left(\frac{1}{16}\right) - \frac{1}{16}\left(\frac{1}{64}\right)$
$= 1 - \frac{1}{8} - \frac{1}{128} - \frac{1}{1024}$.

$P_3(-0.25) \approx 0.8662109$.

$f(-0.25) = \sqrt{-0.25+1} = \sqrt{0.75}$
$f(-0.25) \approx 0.8660254$

Actual Error $\approx 1.855 \times 10^{-4}$.

$\sqrt{1.25} = \sqrt{x+1}$
$1.25 = x+1$
$x = 0.25$

$P(0.25) = 1 + \frac{1}{2}\left(\frac{1}{4}\right) - \frac{1}{8}\left(\frac{1}{16}\right) + \frac{1}{16}\left(\frac{1}{64}\right)$
$= 1 + \frac{1}{8} - \frac{1}{128} + \frac{1}{1024}$.

$P_3(0.25) \approx 1.1181641$

$f(0.25) = \sqrt{0.25+1} = \sqrt{1.25}$
≈ 1.1180340

Actual Error: 1.301×10^{-4}

$\sqrt{1.5} = \sqrt{x+1}$
$1.5 = x+1$
$0.5 = x$.

$P_3(0.5) = 1 + \frac{1}{2}\left(\frac{1}{2}\right) - \frac{1}{8}\left(\frac{1}{4}\right) + \frac{1}{16}\left(\frac{1}{8}\right)$
$= 1 + \frac{1}{4} - \frac{1}{32} + \frac{1}{128}$

$P_3(0.5) \approx 1.2265625$

$f(0.5) = \sqrt{0.5+1} = \sqrt{1.5}$.
≈ 1.2247449

Actual Error $\approx 1.818 \times 10^{-3}$

1.1.11 Let $n=2$, find $P_2(x)$ for $f(x)=e^x\cos x$ about $x_0=e$

$$P_2(x)=f(e)+f'(e)x+\frac{f''(e)}{2!}x^2+R_2(x)$$

→ Find values needed.

$$f'(x)=e^x\cos x \Longrightarrow e^x(-\sin x)+e^x\cos x$$
$$=e^x(\cos x-\sin x)$$

$$f''(x)=e^x(\cos x-\sin x)\Longrightarrow e^x(-\sin x-\cos x)+e^x(\cos x-\sin x)$$
$$=-e^x(\sin x+\cos x)+e^x(\cos x-\sin x)$$
$$=e^x(\cos x-\sin x)-e^x(\sin x+\cos x)$$
$$=e^x[(\cos x-\sin x)-(\sin x+\cos x)]$$
$$=e^x(\cancel{\cos x}-\sin x-\sin x-\cancel{\cos x})$$
$$=-2e^x\sin x.$$

$$f(0)=e^0\cos(0)=1(1)=1$$
$$f'(0)=e^0(\cos(0)-\sin(0))=1(1-0)=1$$
$$f''(0)=-2e^0\sin(0)=-2(1)(0)=0.$$

$$\boxed{P_2(x)=1+x}$$

$$R_2(x)=\frac{f'''(z)(x^3)}{3!}$$
$$=\frac{-2(e^z\cos z+e^z\sin z)x^3}{6}$$

remainder term → $$\boxed{R_2(x)=-\frac{e^z(\cos z+\sin z)x^3}{3}}$$, z is between x & 0.

(a) Approx $f(0.5)$:

$$P_2(0.5) = 1 + 0.5 = 1.5$$

Upperbound for error $|f(x) - P_2(x)|$:

$$K_2(0.5) = -\frac{e^3(\cos 3 + \sin 3)\, x^3}{3}$$

→ When $3 = \pi/4$, $(\cos 3 + \sin 3)$ reaches a max. However, 3 must be between $x_0(0)$ and $x(0.5)$ → $(\pi/4 > 0.5)$. So, let $3 = 0.5$ instead, since that is the highest we can go. e^3 has no clear bound to consider here (growing exponentially)

Thus: $K_2(0.5) = -\frac{e^{0.5}(\cos(0.5) + \sin(0.5))}{3}(0.5)^3$

$$\approx -0.093222$$

And therefore, the upper bound for error

$$|f(x) - P_2(x)| \leq 0.0932$$

Actual Error $= |f(0.5) - P_2(0.5)|$

$$= |e^{0.5}\cos(0.5) - 1.5| \approx 0.053111$$

The actual error is below the upperbound from our error formula, as should be expected.

(b) Bound for error: $|f(x) - P_2(x)|$ on interval $[0,1]$

$$R_2(x) = -\frac{e^{\xi}(\cos\xi + \sin\xi)x^3}{3}$$ 1.252

This term maxes when $\xi = 1$, $x = 1$

Consider when $(\cos\xi + \sin\xi)$ maxes: $\xi = \pi/4$, $x = 1$

↳ $\left| -\frac{e^{\pi/4}(\cos\pi/4 + \sin\pi/4)\,1^3}{3} \right| = \frac{e^{\pi/4}(\sqrt{2})}{3} \approx 1.034$

Consider when e^{ξ} maxes: $\xi = 1$, $x = 1$

↳ $\left| -\frac{e(\cos(1) + \sin(1))\,1^3}{3} \right| \approx 1.252$ ← greater upper bound

Therefore, $|f(x) - P_2(x)| \leq 1.252$ when $x \in [0,1]$.

(c) Approximate $\int_0^1 f(x)dx$ w/ $P_2(x)$

$$\int_0^1 P_2(x)dx = \int_0^1 1 + x\,dx = x + \frac{x^2}{2}\Big|_0^1 = 1 + \frac{1}{2} - 0 = 1.5$$

Therefore: $\int_0^1 f(x)dx \approx 1.5$

(d) Find upper bound for (c) w/ $\int_0^1 |R_2(x)dx|$ & compare w/ actual

$$\int_0^1 \frac{e^{\xi}(\cos\xi + \sin\xi)}{3} x^3 dx = \frac{e^{\xi}(\cos\xi + \sin\xi)}{3} \frac{x^4}{4}\Big|_0^1$$

$$= \frac{e^{\xi}(\cos\xi + \sin\xi)}{12}$$ → as found in b, this term maxes when $\xi = 1$, therefore, upperbound:

$$\left| \int_0^1 f(x)dx - \int_0^1 P_2(x)dx \right| \leq \frac{e(\sin(1) + \cos(1))}{12} \approx 0.313$$ (7)

→ The actual error:

$$\int_0^1 f(x)\,dx \Rightarrow \int_0^1 e^x \cos x\,dx$$ ⌐ let $u = e^x$ $dv = \cos x\,dx$, $du = e^x dx$ $v = \sin x$

$$\int_0^1 e^x \cos x\,dx = e^x \sin x\Big|_0^1 - \int_0^1 e^x \sin x\,dx$$ ⌐ $u = e^x$ $dv = \sin x\,dx$, $du = e^x dx$ $v = -\cos x$

$$\int_0^1 e^x \cos x\,dx = e^x \sin x\Big|_0^1 - \left(-e^x \cos x\Big|_0^1 + \int_0^1 e^x \cos x\,dx\right)$$

$$\int_0^1 e^x \cos x\,dx = \left[e^x \sin x + e^x \cos x\right]_0^1 - \int_0^1 e^x \cos x\,dx$$

$$2\int_0^1 e^x \cos x\,dx = e^x(\sin x + \cos x)\Big|_0^1$$

$$\int_0^1 e^x \cos x\,dx = \frac{e^x(\sin x + \cos x)}{2}\Big|_0^1 = \frac{e(\sin(1)+\cos(1))}{2} - \frac{1(\sin(0)+\cos(0))}{2}$$

$$\approx 1.378$$

So: **Actual Error $\approx |1.378 - 1.5| \approx 0.122$.**

↳ Which is lower than our upper bound 0.313 ✓

1.1.14 $f(x) = 2x\cos(2x) - (x-2)^2$, $x_0 = 0$.

(a) Find $P_3(x)$ and approx $f(0.4)$

$R_4(x)$

$$P_3(x) = f(0) + f'(0)x + \frac{f''(0)x^2}{2} + \frac{f'''(0)x^3}{6} + \frac{f^{IV}(\xi)x^4}{4!}$$

→ find the needed values:

$2x \cdot \sin(2x) \cdot 2$

$$f'(x) = 2\cos(2x) - 4x\sin(2x) - 2(x-2)$$
$$= 2\cos(2x) - 4x\sin(2x) - 2x + 4.$$

$4x(\cos(2x))$

$$f''(x) = -4\sin(2x) - (4\sin(2x) + 8x\cos(2x)) - 2.$$
$$= -8\sin(2x) - 8x\cos(2x) - 2.$$

$f'''(x) = -16\cos(2x) - (8\cos(2x) - 16x\sin(2x))$

$= -24\cos(2x) + 16x\sin(2x)$.

$f^{iv}(x) = 48\sin(2x) + (16\sin(2x) + 32x\cos(2x))$

$= 64\sin(2x) + 32x\cos(2x)$

$f(0) = 0 - (0-2)^2 = -(-2)^2 = -4$

$f'(0) = 2\cos(0) - 0 - 0 + 4 = 2 + 4 = 6$.

$f''(0) = -8\sin(0) - 0 - 2 = 0 - 2 = -2$.

$f'''(0) = -24\cos(0) + 0 = -24$

$$P_3(x) = -4 + 6x - \frac{2x^2}{2} - \frac{24}{6}x^3 + \frac{64\sin(2\xi) + 32x\cos(2\xi)}{24}x^4$$

$$\boxed{P_3(x) = -4 + 6x - x^2 - 4x^3} + \underbrace{\frac{8\sin(2\xi) + 4\xi\cos(2\xi)}{3}x^4}_{R_3(x).}$$

Approx $f(0.4)$:

$P_3(0.4) = -4 + 6(0.4) - (0.4)^2 - 4(0.4^3)$

$\boxed{P_3(0.4) = -2.016.}$

(b) upper bound for error $|f(0.4) - P_3(0.4)|$

$R_3(x) = \frac{8\sin(2\xi) + 4\xi\cos(2\xi)}{3}x^4$, such that ξ is between $x_0(0)$ and $x(0.4)$

We will hit a max when $\xi = 0.4$, so:

$R_3(0.4) = \frac{8\sin(2(0.4)) + 4(0.4)\cos(2(0.4))}{3}(0.4)^4 \approx 0.0585$.

Therefore, our upperbound error is

$$|f(x) - P_3(x)| \leq 0.0585$$

Actual Error: $|2(0.4)\cos(2*0.4) - (0.4-2)^2 - (-2.016)|$

$= |+2.0026 + 2.016| \approx 0.0134 = \text{actual error}$

well under our upper bound.

(c) $P_4(x) = -4 + 6x - x^2 - 4x^3 + \frac{f^{iv}(0)}{4!}x^4 + \frac{f^{v}(\xi)}{5!}x^5$

$f^{iv}(0) = 8\sin(0) + 4(0)\cos(0)$

$= 0$

$f^{v}(x) = 64\sin(2x) + 32x\cos(2x) \rightarrow 128\cos(2x) + (32\cos(2x) - 64x\sin(2x))$

$= 160\cos(2x) - 64x\sin(2x)$

$$P_4(x) = -4 + 6x - x^2 - 4x^3 + 0 + \underbrace{\frac{160\cos(2\xi) - 64\xi\sin(2\xi)}{120}x^5}_{R_4(x)}$$

$$P_4(0.4) = P_3(0.4) = -2.016.$$

(d) upper bound error $|f(0.4) - P_4(0.4)|$

Let $\xi = 0$ be our max for $f^{v}(\xi)$ this time.

$$R_4(0.4) = \frac{20\cos(2(0)) - 8(0)\sin(2(0))}{15}(0.4)^5 = \frac{20}{15}(0.4)^5$$

≈ 0.0137

Therefore: $|f(0.4) - P_4(0.4)| \leq 0.0137$

Actual error (same as b) ≈ 0.0134 ← 0.0134 is still $\leq$ our upper bound 0.0137.

COMPUTER/MATLAB PART 1

ORIGINAL PROBLEM STATEMENT

"In example 3 (section 1.1 page 9), they show 2 graphs in Figure 1.9 for cos(x) and a corresponding quadratic approximation."

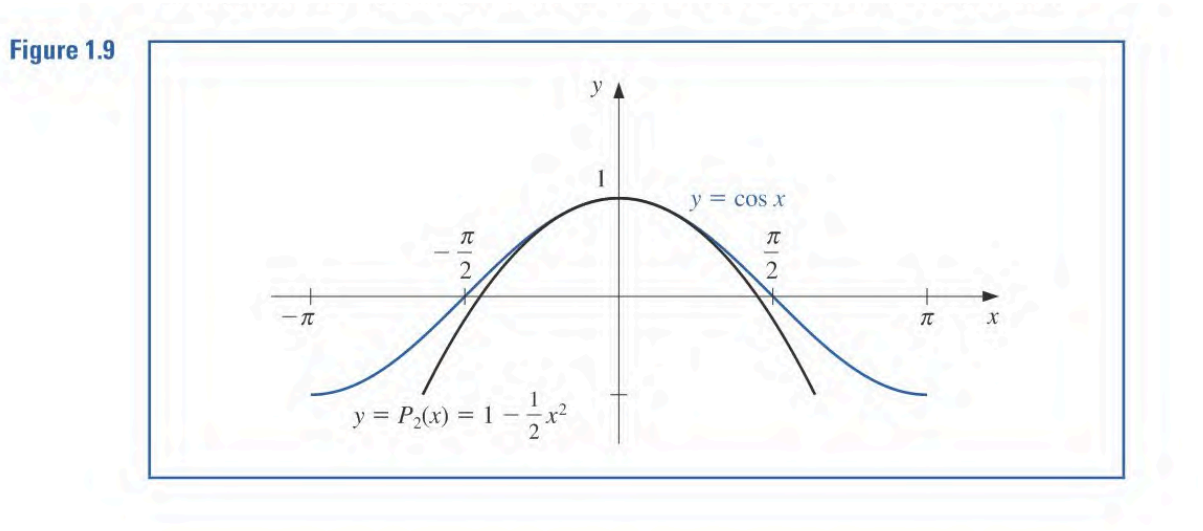


Figure 1.9 from "Numerical Analysis", Burden, et al.

PART A

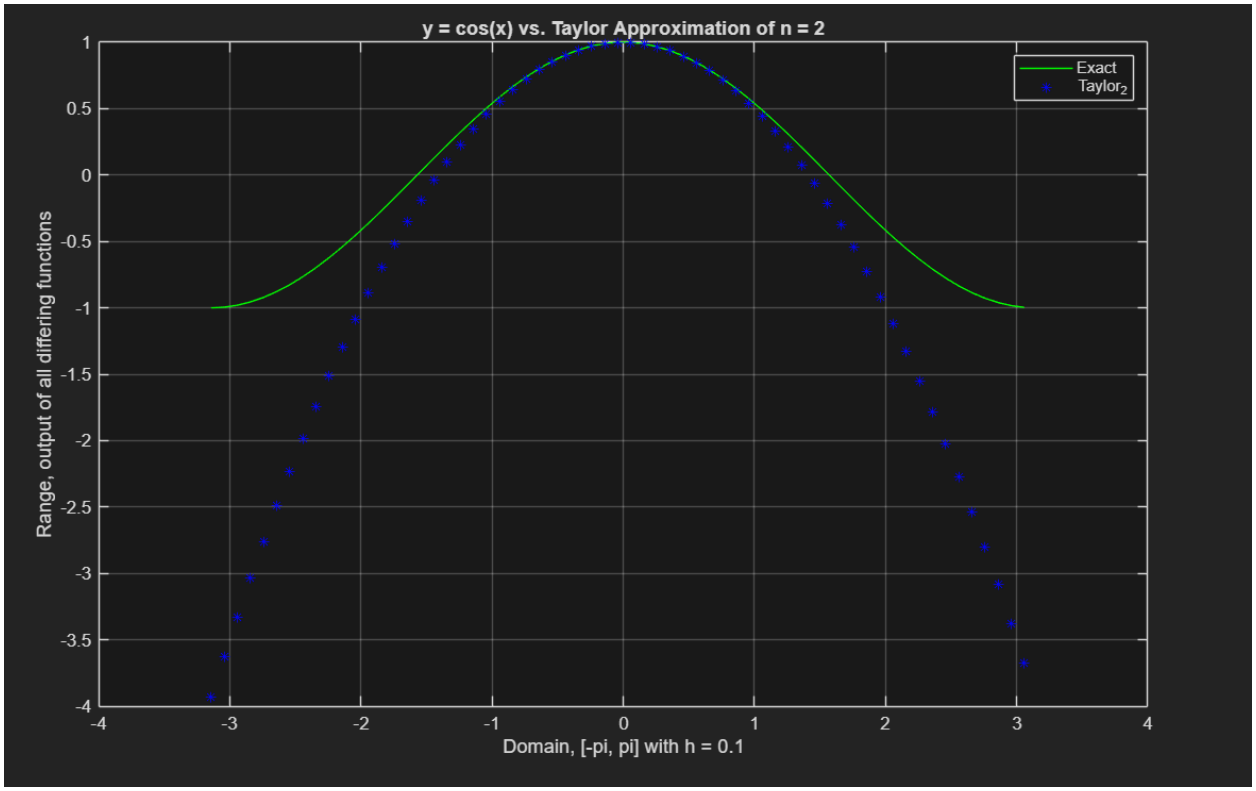
"Replicate the figure with Matlab using a domain spacing of h = 0.1 for the computational domain [−π, π]"

As follows is the code written to accomplish the requirements of part A.

```
% -------------------------------------
% Compare exact values of cos(x)
% to different Taylor approximations.
% -------------------------------------
h = 0.1; % domain spacing of 0.1
x = [-pi:h:pi]; % domain [-pi, pi]
y_exact = cos(x); % exact cos(x) function
y_taylor2 = 1 - (1/2)*(x.^2); % taylor approx, n = 2
```

```
% ------------------------------------
% plot part (a):
% plot exact y and taylor approximation on n = 2
% ------------------------------------
plot(x, y_exact, 'g-'); hold on;
plot(x, y_taylor2, 'b*'); hold on;
```

As follows is the figure plotted of just y = cos(x) and y = 1 - (1/2)*(x^2).



The Taylor approximation seems to get the closest to accurate around x = 0, which is the value used when building the Taylor series in figure 1.9. (Technically, it is the Maclaurin series for cos(x)).

Additionally, the further we get from x = 0, the less accurate the approximation becomes to the exact.

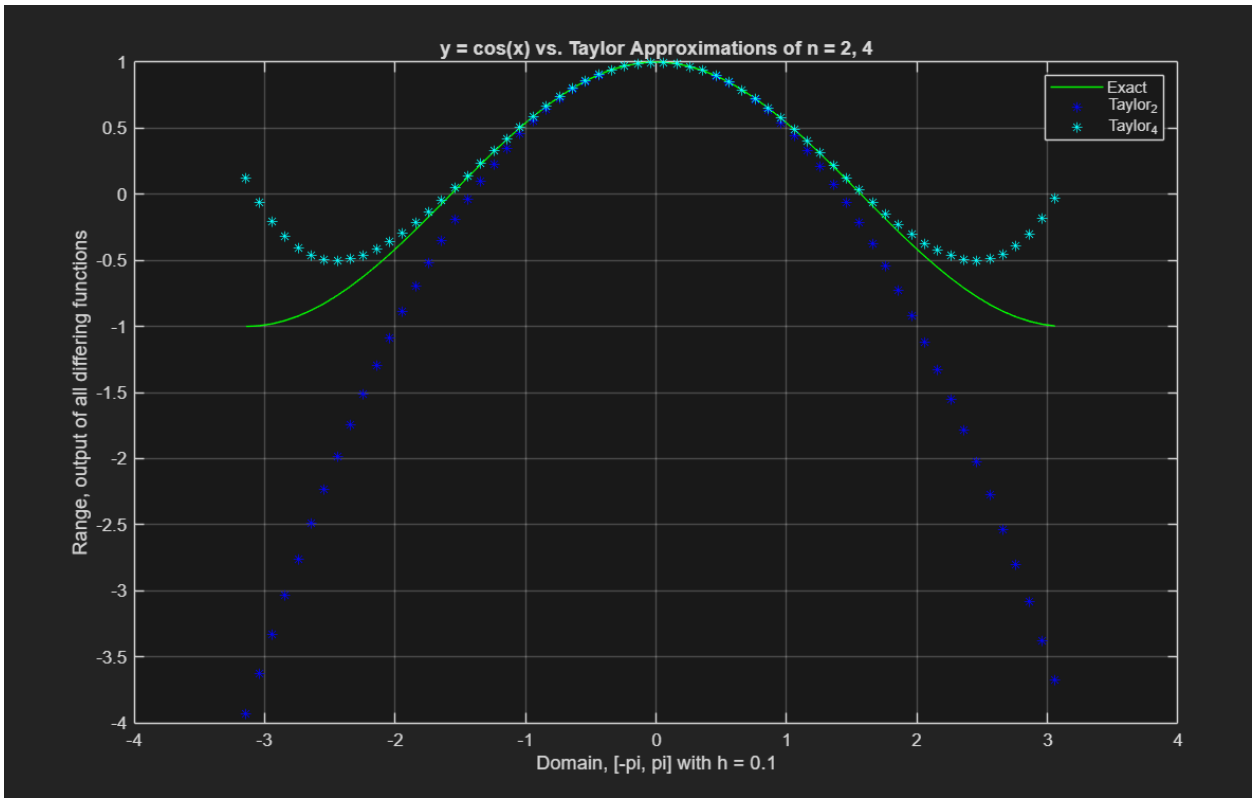
PART B

"Include in the plot an approximation of degree 4 for cos(x), that is plot P4(x). This figure should contain the original function f (x), P2(x) and P4(x). Describe your findings."

As follows is the code written to accomplish the requirements of part B.

```
% ------------------------------------
% plot part (b):
% plot taylor approximation of n = 4
% ------------------------------------
y_taylor4 = y_taylor2 + (1/24)*(x.^4);
plot(x, y_exact, 'g-'); hold on;
plot(x, y_taylor2, 'b*'); hold on;
plot(x, y_taylor4, 'c*'); hold on;
```

As follows is the figure plotted of y = cos(x), y = 1 - (1/2)*(x^2), and y = 1 - (1/2)*(x^2) + (1/24)*(x^4).

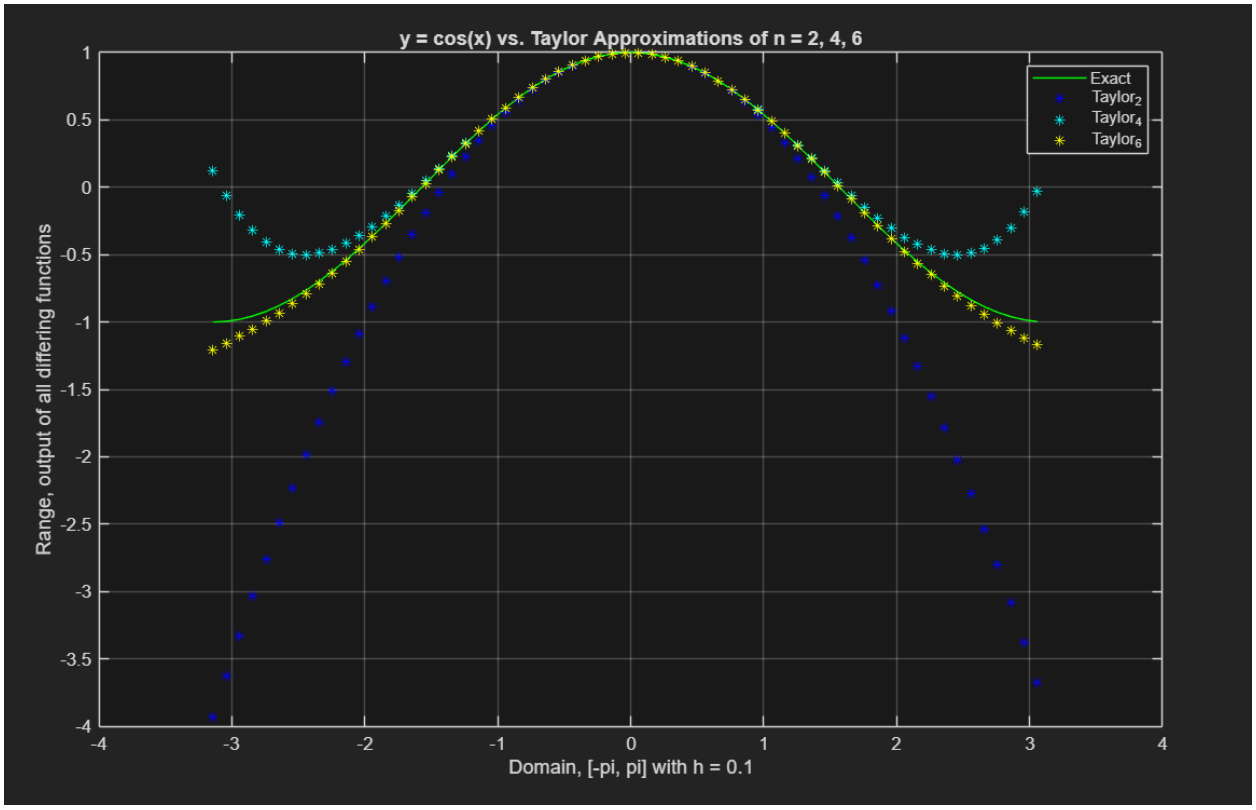


As we can visualize above, both of the Taylor approximations remain the most accurate around x = 0, the original value used to build the Taylor series.

However, the approximation when n = 4 adheres closer to the exact values as we diverge from x = 0 than the n = 2 approximation. To further see the effect of adding terms to the approximation, we can go one further by adding another term to the Taylor series with the following.

```
y_taylor6 = y_taylor4 - (1/720)*(x.^6);
```

The addition can be seen in the following figure.



This highlights significantly that the longer our Taylor approximation becomes, the closer it gets to the exact values, making it a remarkably good approximation with more and more terms added.

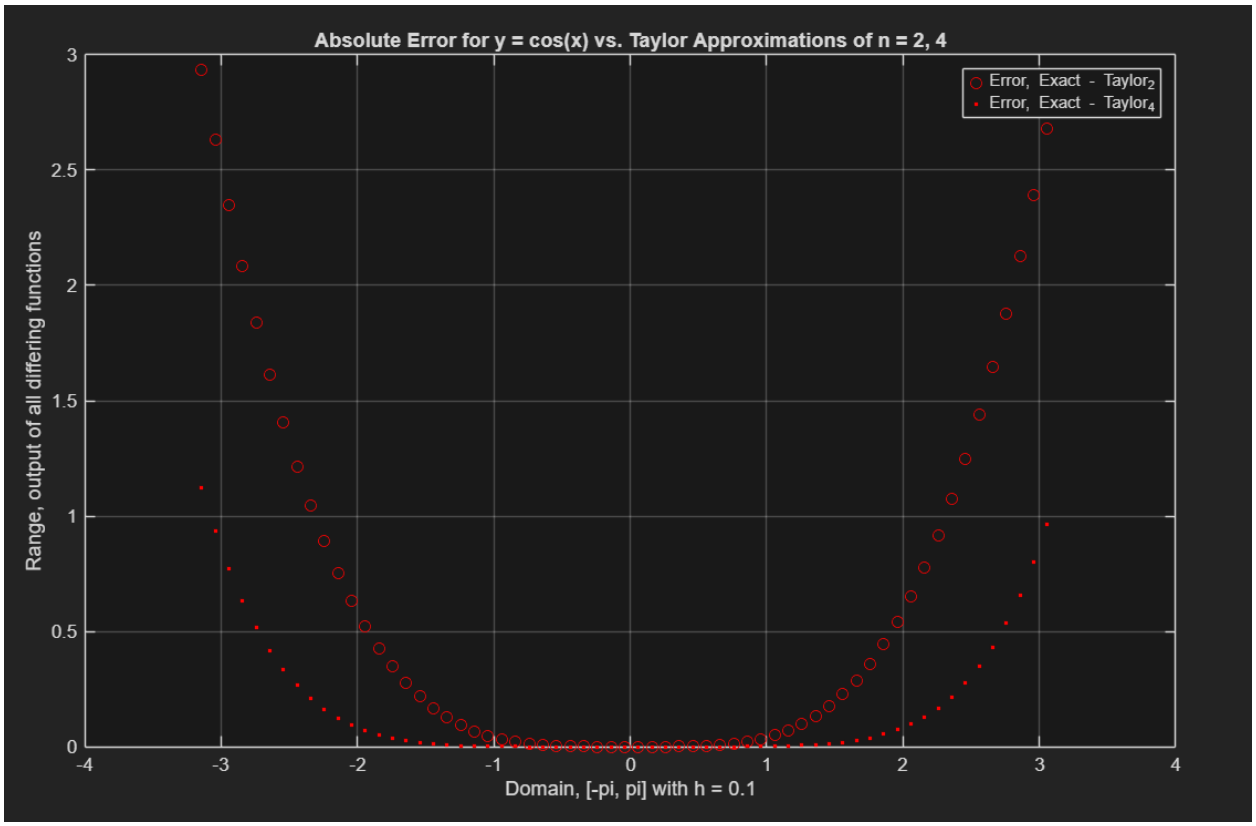
PART C

"Plot the absolute error e2(x) and e4(x) on the computational domain [−π, π] with h = .1. We use the definition from the absolute error which is given by en(x) = |f (x) − Pn(x)|. If you try smaller values of h does your error improve? Try h = 0.1, 0.01 and 0.001. Describe your findings."

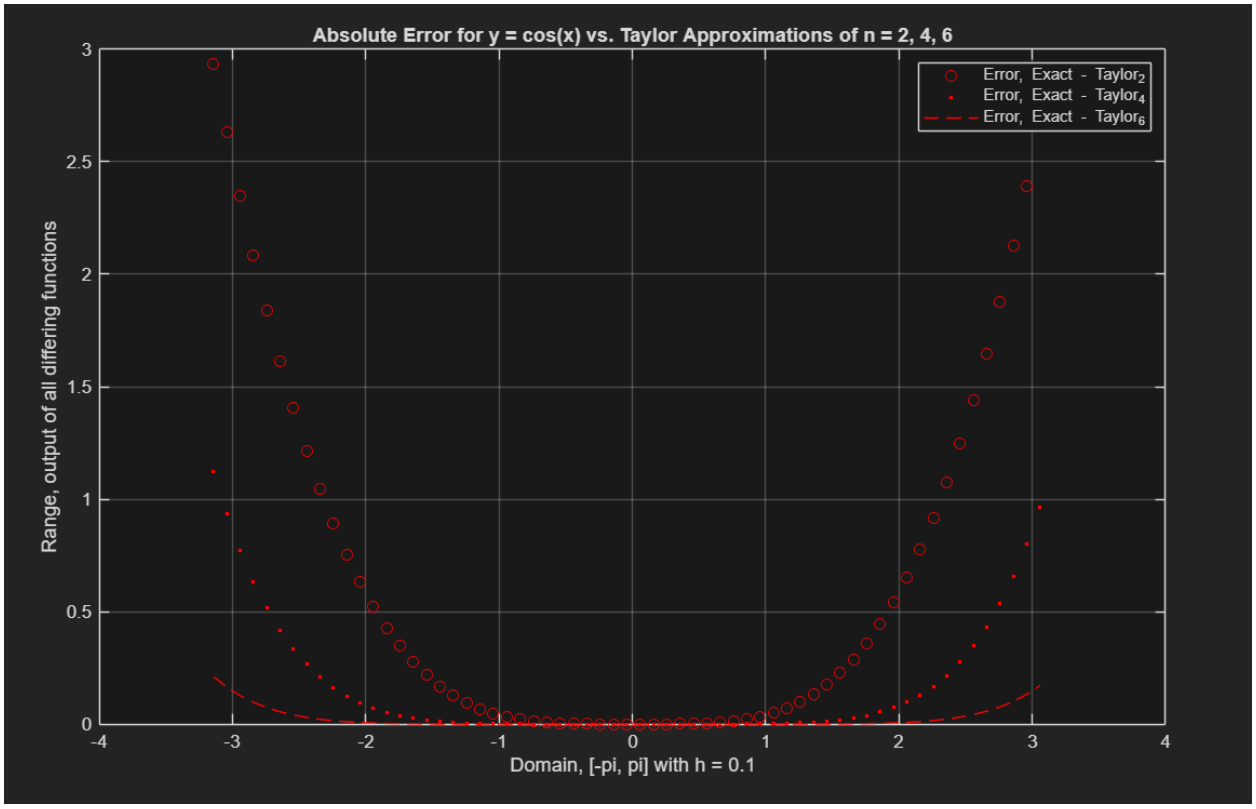
As follows is the code written to accomplish the requirements of part C.

```
% ------------------------------------
% plot part (c)
% plot absolute errors e_2 and e_4
% ------------------------------------
% formula for absolute/actual error: |f(x) - P_n(x)|
e_2 = abs(y_exact - y_taylor2);
e_4 = abs(y_exact - y_taylor4);
plot(x, e_2, 'ro', x, e_4, 'r.'); hold on;
```

When our domain spacing (h) is 0.1 for all of our functions, we found an absolute error for Taylor n = 2 and n = 4 as follows:

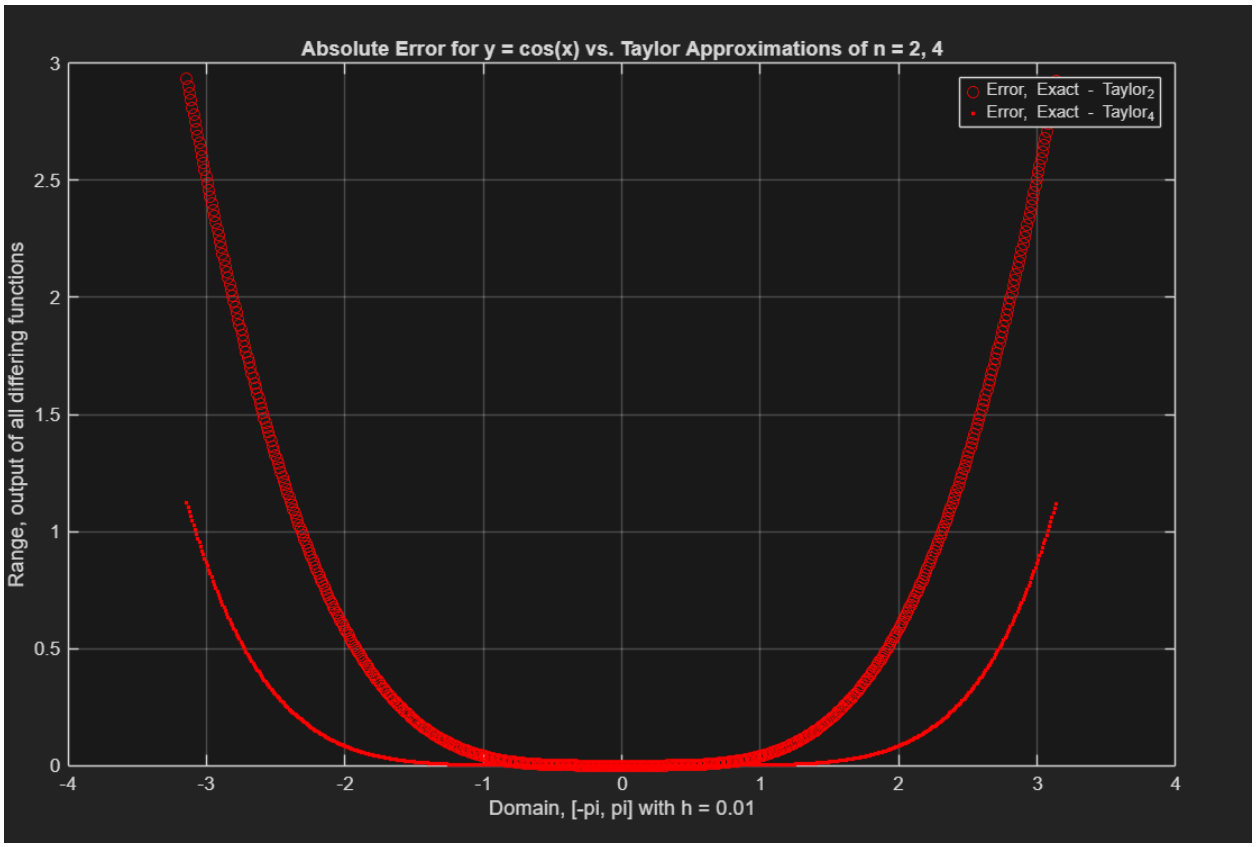


As we can see, the absolute error growth decreases when adding terms–this is a more numeric justification of the observations that when adding n = 6 to the approximation, the accuracy of the approximation increased. We can further support that idea by simply adding the absolute error plot for n = 6 to the other two to compare:

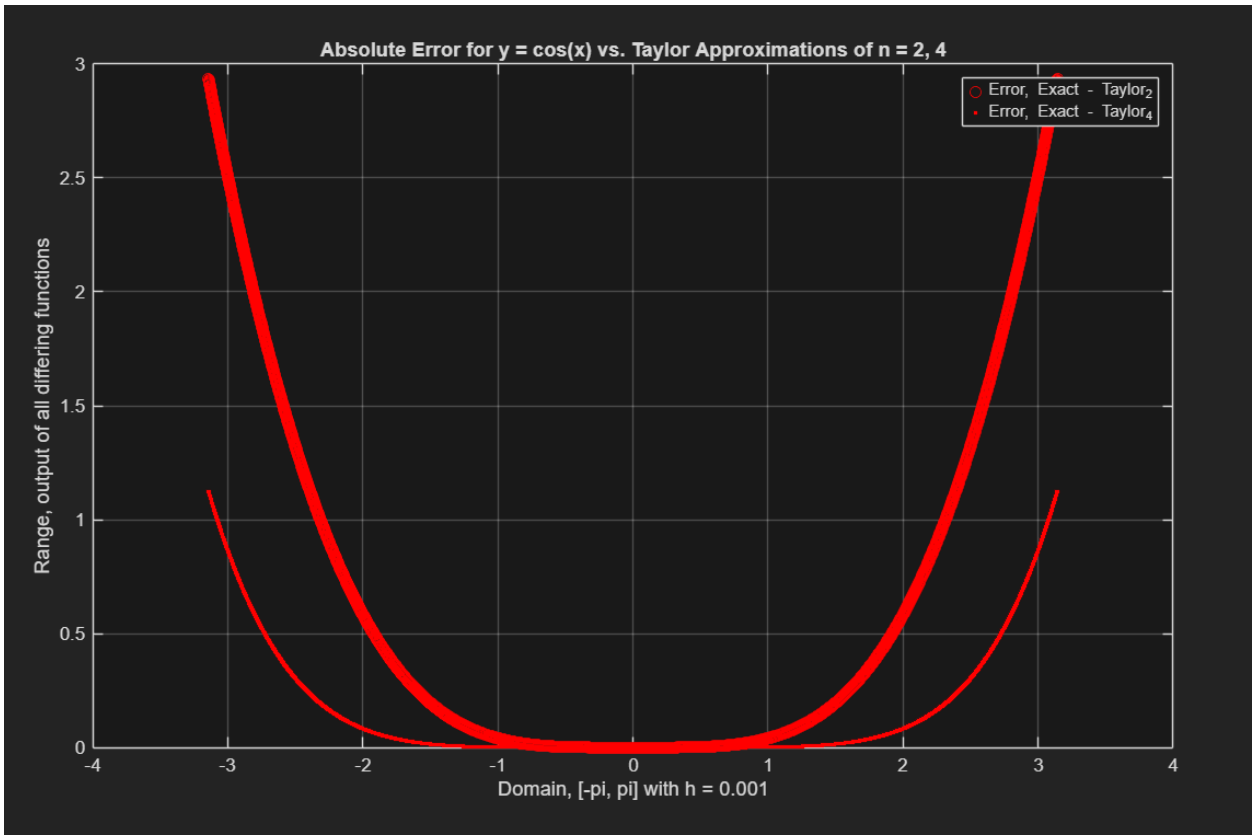


As we can see above, the error growth shrinks considerably with the Taylor approximation having more terms. Clearly, there is a relationship between increasing polynomial terms to the approximation meaning a stark decrease in the error between the approximation and exact values.

Now, we will explore the relationship between domain spacing and error. We will change h to 0.01 and see if the error is affected (moving forward with Taylor n = 2, 4 only for consistency):



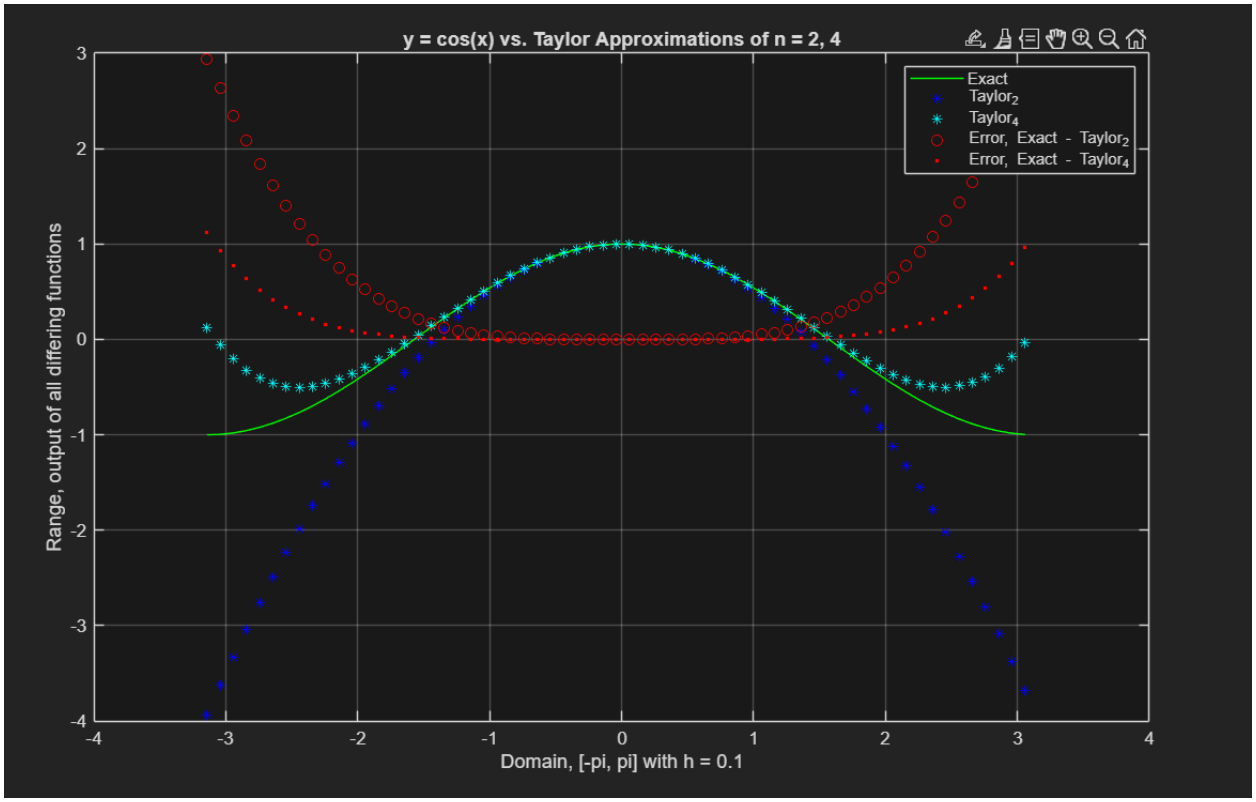
Further, we will try h = 0.001:



As we can see, there is no real change in the error measurements, implying that the domain spacing is independent of any effect on the accuracy of the approximation.

ADDITIONAL NOTES

As one final visual, all graphs (negating Taylor n = 6) can be seen plotted against each other here:



Seeing these all together provides a unified look of error versus the accuracy of the approximation against the exact. We can see very clearly the error of both approximations increase exponentially the further we retreat from x = 0. We can also see here the validation of the Taylor n = 4 approximation being clearly a better fit for the exact, with its error growth rate far lower.

COMPUTER/MATLAB PART 2

ORIGINAL PROBLEM STATEMENT

"Solve the following Initial Value Problem (IVP) given by the Logistic Model

$$N' = rN(1 - N/K), N(0) = N0$$

Using the built-in function ODE45 found in MATLAB."

A note for the following implementation: we will be using a step size (denoted as dt) of 0.01 for all schemes shown for part A, B, and C.

PART A

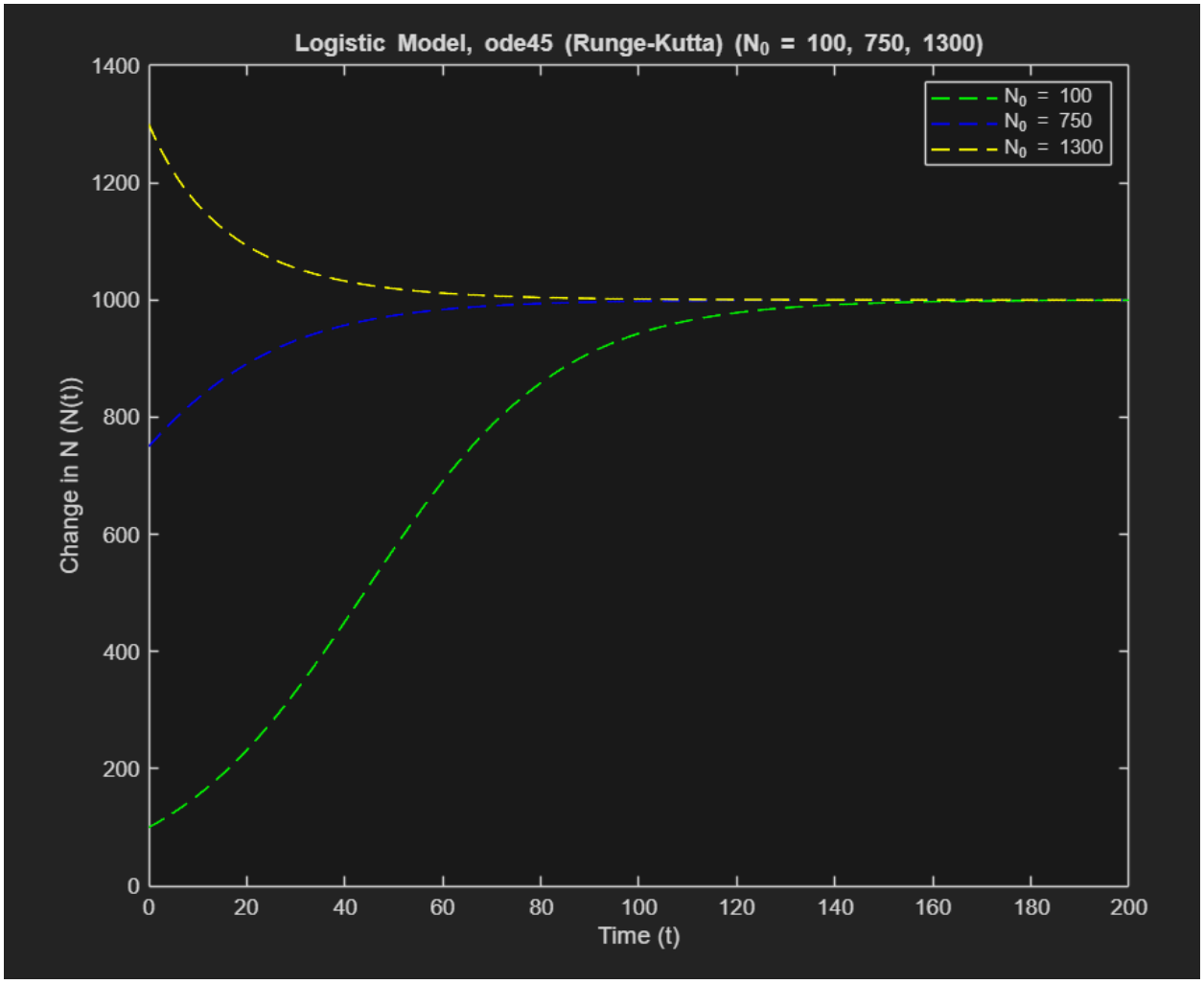
"Using r = 0.05, K = 1000, and N0 = 100 obtain a numerical solution and plot (N (t) vs t) in the computational domain for t on [0, Tf]. Here Tf will be selected in a way that your figure captures the solution approaching to a fixed/constant value. What happens if N0 = 750 or N0 = 1300 ? Describe your findings."

As follows is the code written to accomplish the requirements of part A.

```
% --------------------------------------
% FUNCTION IN SEPARATE M FILE
% --------------------------------------

% logistic model for growth implementation
%
% given information/equation:
% N' = rN(1-(N/K)), N(0) = N_0
function [result] = logModel(N, r, K)
   % logistic model
   result = r * N * (1 - N/K);
end

% --------------------------------------
% CODE WITHIN MAIN PROB2.M SCRIPT
% --------------------------------------

% initial values given
N0_1 = 100;
```

```
N0_2 = 750;
N0_3 = 1300;
% logistic model constants given
r = 0.05;
K = 1000;
% ---------------------------------------
% plot part (a):
% use: r = 0.05, K = 1000, N_0 = 100
% to find solution and plot.
% plot on domain [0, T_f].
% change values of N_0 to 750, 1300...
% ---------------------------------------
t_f = 200; % setting to 200 shows convergence clearly
dt = 0.01; % setting with somewhat small steps
ts = [0:dt:t_f];
% solve the ode with the runge-kutta scheme, ode45
[t1, N1_rk] = ode45(@(t, N) logModel(N, r, K), ts, N0_1);
[t2, N2_rk] = ode45(@(t, N) logModel(N, r, K), ts, N0_2);
[t3, N3_rk] = ode45(@(t, N) logModel(N, r, K), ts, N0_3);
% plot the models
figure(1);
plot(ts, N1_rk, 'g--', ts, N2_rk, 'b--', t3, N3_rk, 'y--',
'LineWidth', 1); hold on;
```

As follows is the figure plotted of the three initial values used in the approximation of the logistical model solution.



What we can notice above is that all three initial values end up converging to K (given as 1000), which is to be expected as it is the behavior of this model. If N0 is larger than K, the model exponentially decays until converging to K. It is similar for when N0 is less than K: exponential growth until converging to K.

ode45 is a built-in method to numerically solve ordinary differential equations, just like the logistic model. It implements Runge-Kutta which is a fourth-order scheme, with approximation error being O(dt^4), therefore expected to be highly accurate with a small step size (dt), and even accurate with a large one.

PART B

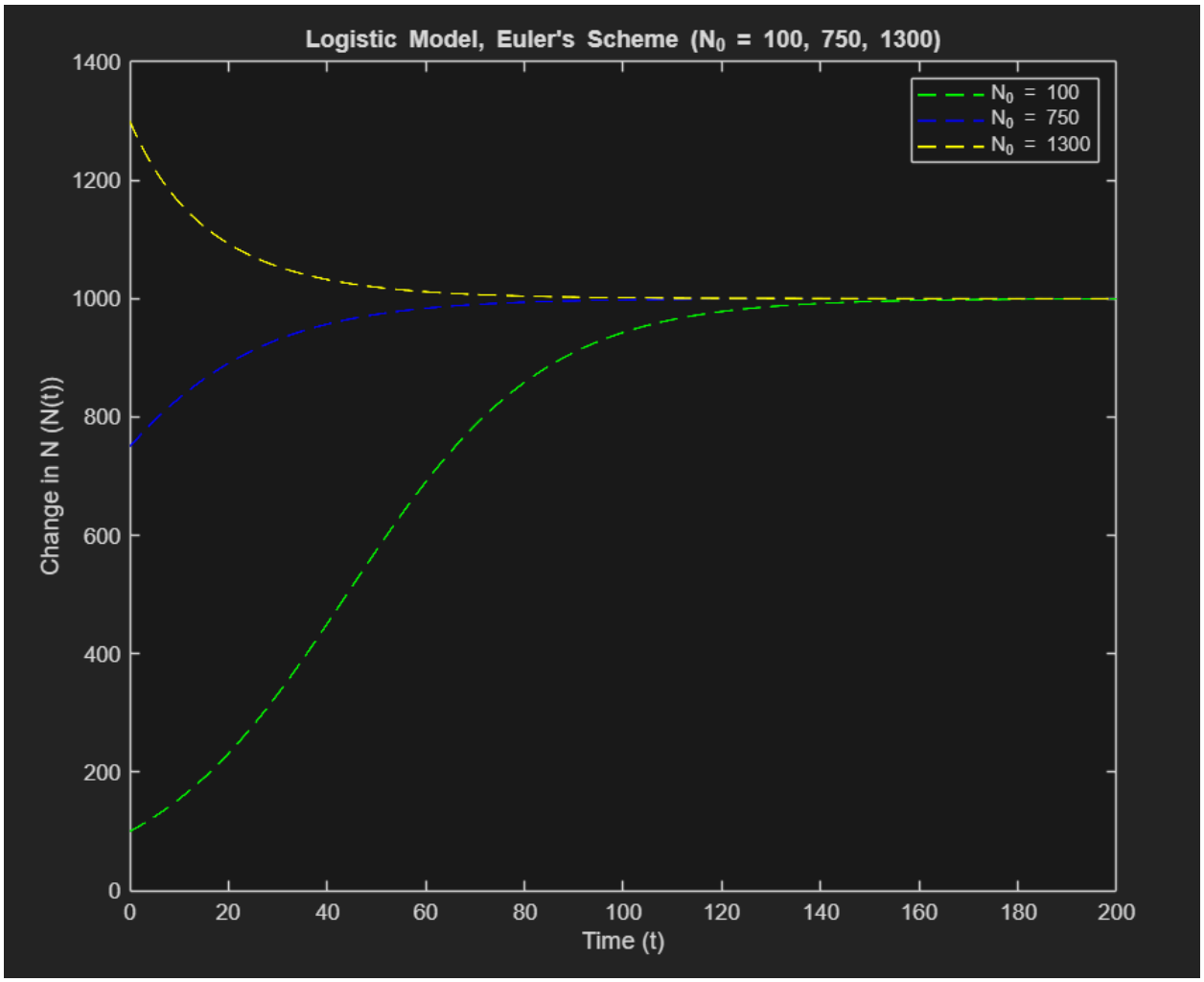
"Repeat part a) by implementing Euler's Scheme."

As follows is the code written to accomplish the requirements of part B.

```
% -------------------------------------
% FUNCTION IN SEPARATE M FILE
% -------------------------------------

% Euler's scheme for ODE approximations
%
% Euler's base scheme is as follows:
% x_n+1 = x_n + dt * f'(x_n), x(0) = x_0
%
% iterations -> how many times to run the scheme
% x_0 -> initial value
% dt -> the dt step to use in implementing euler scheme
% func -> ode function we are approximating

function [result] = euler(iterations, x_0, dt, func)
   x_curr = x_0; % init x_curr
   result = zeros(iterations, 1); % init vector
   result(1) = x_0; % init value in answer vector

   for i = [2:iterations]; % skip init value iteration
       x_next = x_curr + dt * func(x_curr);
       result(i) = x_next; % save result
       x_curr = x_next; % init curr for next iteration
   end;
end

% -------------------------------------
% CODE WITHIN MAIN PROB2.M SCRIPT
% -------------------------------------

% -------------------------------------
% plot part (b):
% repeat part (a) but by implementing
% Euler's scheme.
% -------------------------------------
iterations = length(ts); % how many times to run the scheme
N1_eu = euler(iterations, N0_1, dt, @(N) logModel(N, r, K));
N2_eu = euler(iterations, N0_2, dt, @(N) logModel(N, r, K));
N3_eu = euler(iterations, N0_3, dt, @(N) logModel(N, r, K));
% plot euler's scheme
```

```
figure(2);
plot(ts, N1_eu, 'g--', ts, N2_eu, 'b--', ts, N3_eu, 'y--',
'LineWidth', 1); hold on;
```

As follows is the figure plotted of the logistic model solved with Euler's scheme with the given different initial values.



As we can see, this behaves and looks very similarly to our solution using ode45. However, it is worth noting that we are looking at this at a larger scale and the solutions found are not exactly the same. But, from a distance, they appear to be comparable methods.

This method is first-order accurate, meaning the approximation error is O(dt). This should be noted as a critical difference between the methods.

PART C

"Using a single initial condition, plot both solutions at once using Euler's Scheme and Improved Euler's scheme. Describe your findings."

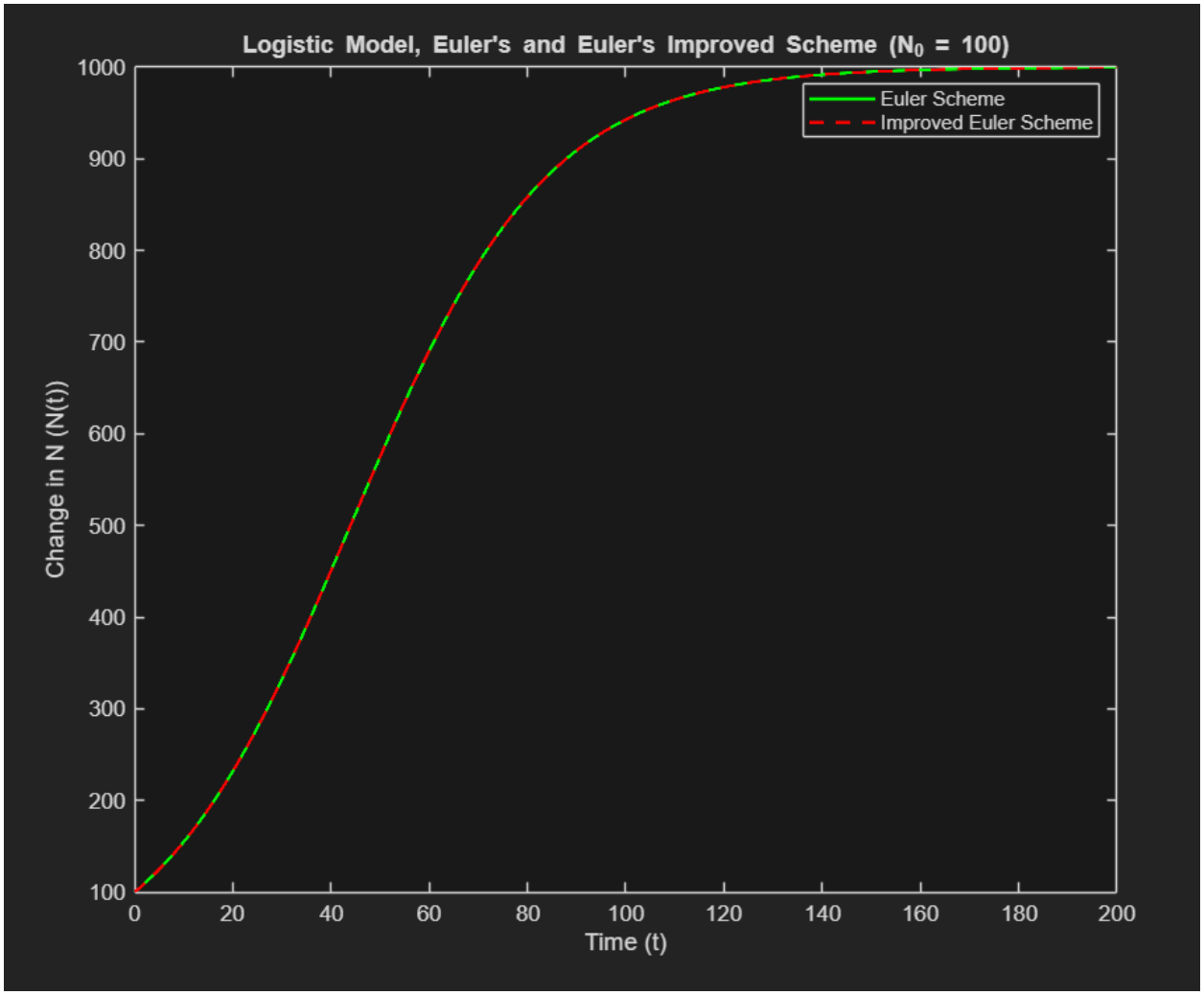
We will use N0 = 100 for this part.

As follows is the code written to accomplish the requirements of part C.

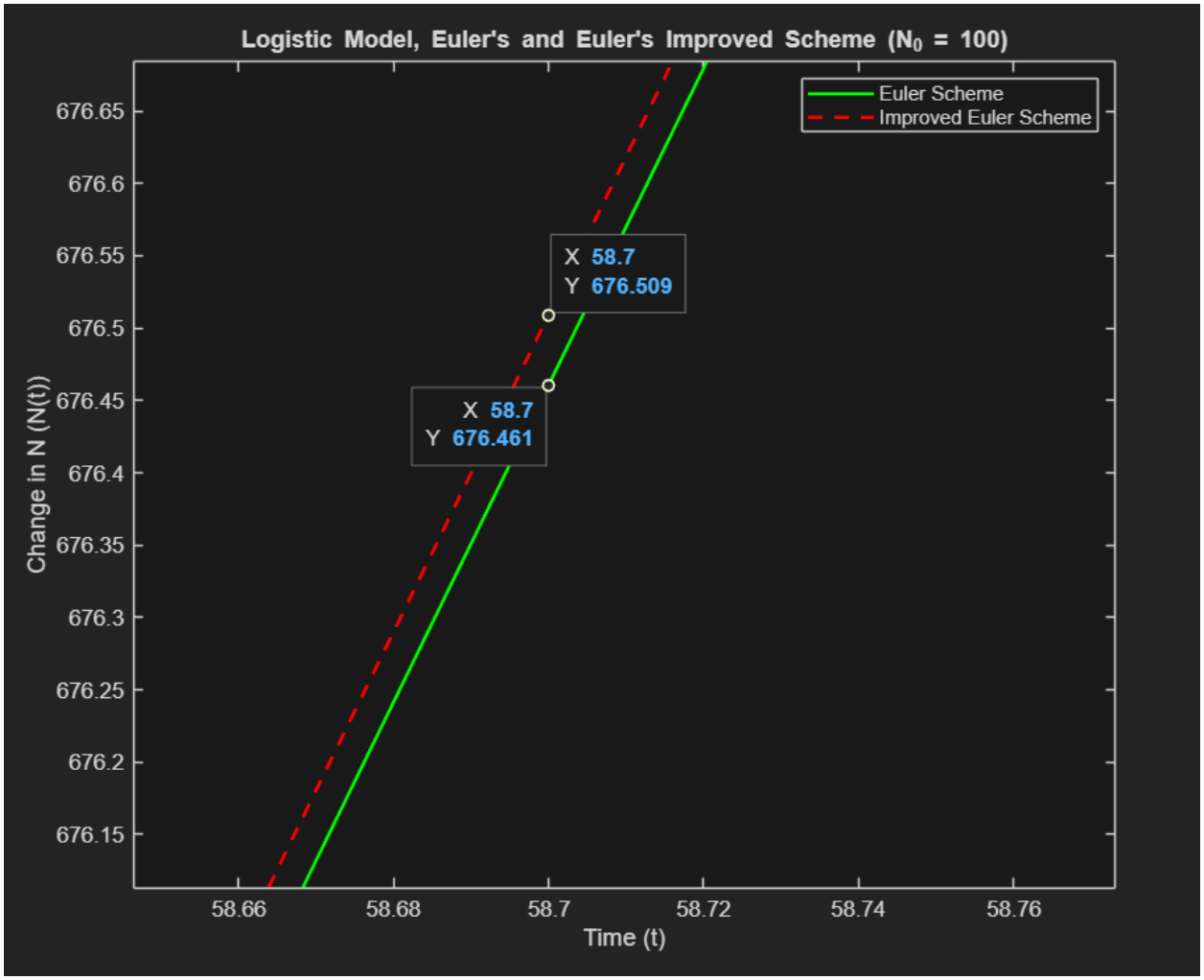
```
% --------------------------------------
% FUNCTION IN SEPARATE M FILE
% --------------------------------------

% Euler's IMPROVED scheme for ODE approximations
%
% Euler's improved scheme is as follows:
% x~ = x_n + dt * f'(x_n), x(0) = x_0
% x_n+1 = x_n + (dt / 2) * (f'(x_n) + f'(x~))
%
% iterations -> how many times to run the scheme
% x_0 -> initial value
% dt -> the dt step to use in implementing euler scheme
% func -> ode function we are approximating

function [result] = eulerImproved(iterations, x_0, dt, func)

   x_curr = x_0; % init x_curr
   result = zeros(iterations, 1); % init vector
   result(1) = x_0; % init value in answer vector

   for i = [2:iterations]; % skip init value iteration
       x_tilde = x_curr + dt * func(x_curr); % trial step
       x_next = x_curr + (dt / 2) * (func(x_curr) +
func(x_tilde)); % real step
       result(i) = x_next; % save result
       x_curr = x_next; % init curr for next iteration
   end;

end
```

```
% ---------------------------------------
% CODE WITHIN MAIN PROB2.M SCRIPT
% ---------------------------------------

% ---------------------------------------
% plot part (c):
% use a single initial condition
% and plot both Euler's and Improved Eulers.
% ---------------------------------------
% we will choose N0 = 100
% keep other variables unchanged for consistency
N1_euImproved = eulerImproved(iterations, N0_1, dt, @(N)
logModel(N, r, K));
% plot improved euler against euler at N_0 = 100
figure(3);
plot(ts, N1_eu, 'g-', ts, N1_euImproved, 'r--', 'LineWidth',
1.5); hold on;
```

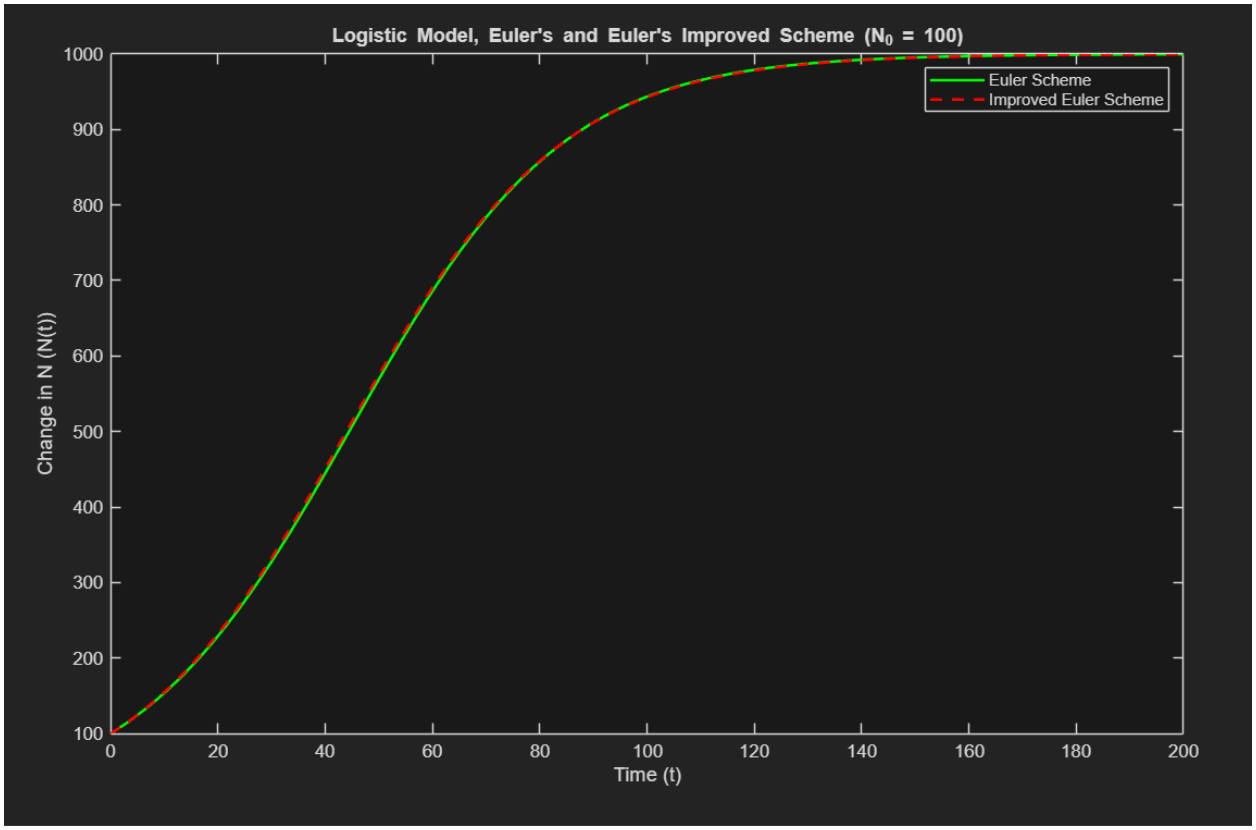
As follows is the figure plotted for Euler's and Euler's Improved schemes with N0 = 100.



These look almost exactly the same at this scale, but as we zoom into an arbitrary point, we can see that they are every so slightly different.

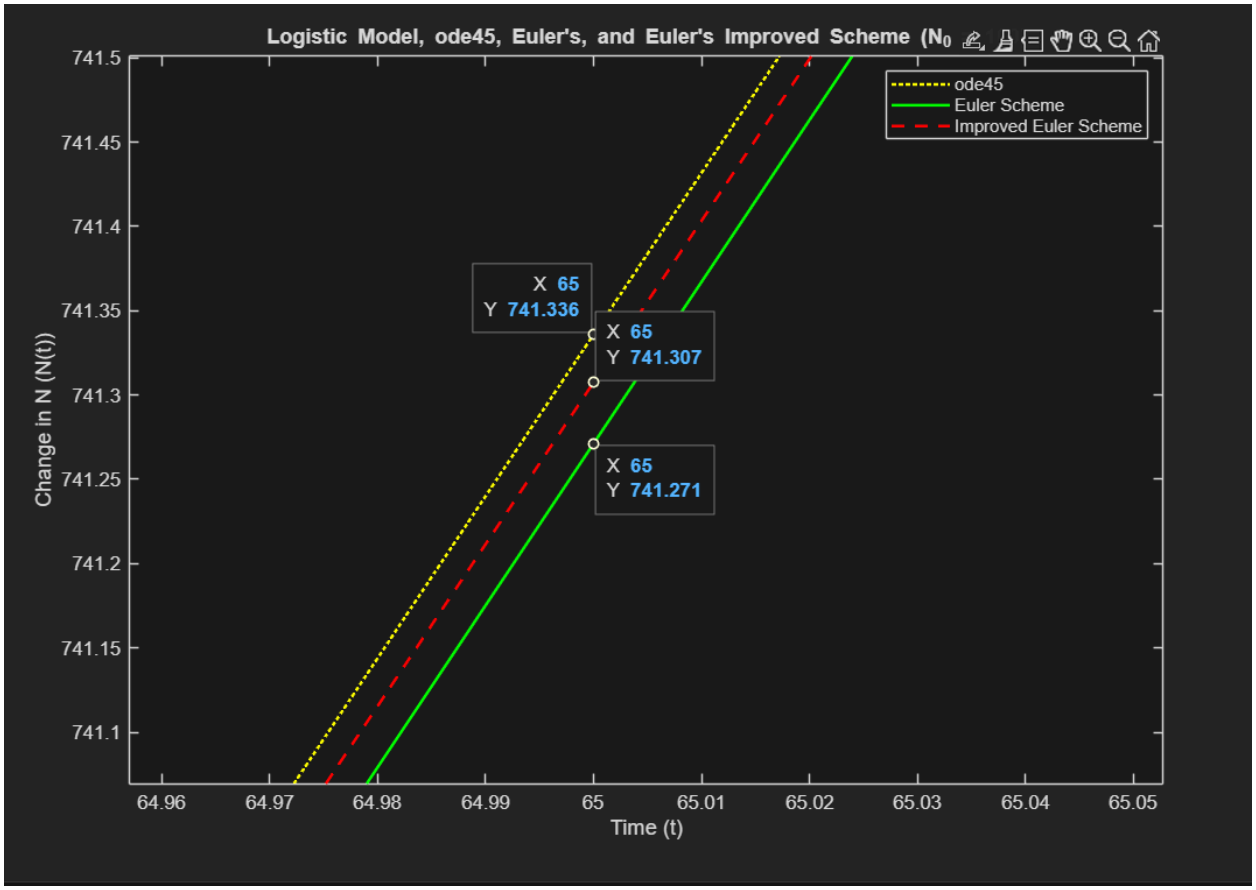


So, their solutions are not exactly the same, which begs the question: “which method is better”? As mentioned prior, we can make a mathematically safe assumption that Euler’s Improved scheme is better at dt = 0.01, since it is a second-order function (approximation error O(dt^2)) as opposed to Euler’s first-order. For small step sizes, this difference may not be completely palpable, but will likely be for large step sizes. If we change the step size to dt = 1, the difference between the two become more noticeable on large-scale.

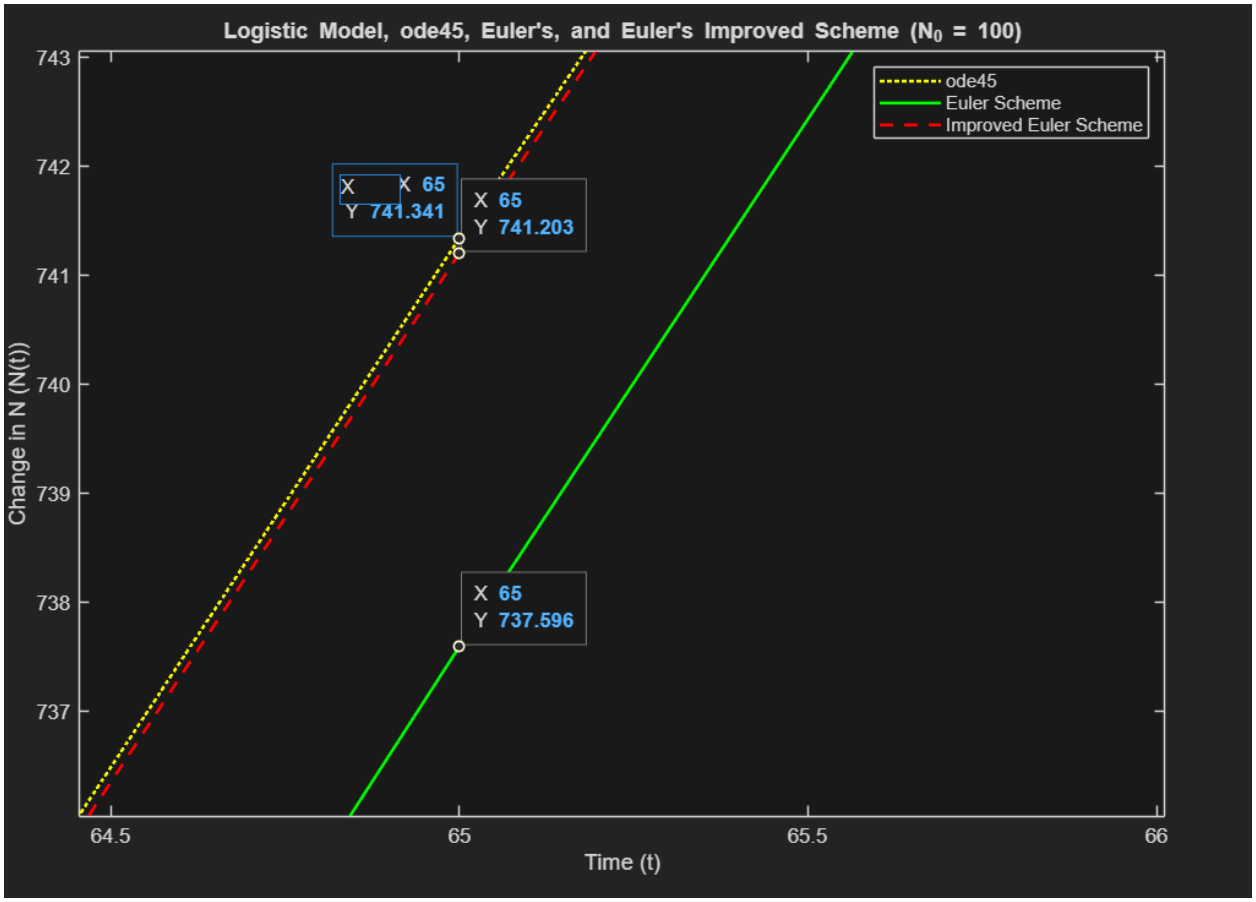


As can be seen above, a larger step size starts to affect the two approximations notably.

We can notice the same difference by looking at a plot including ode45 approximations for N0 = 100 as well against the other two. Again, with dt = 0.01, we can zoom to an arbitrary point for all three plots and see the slight difference.



Now, let’s increase the step size to 1 and find our arbitrary point.



Now we can see a considerable difference in the approximations. Further, Euler's approximation was the most affected by the change, its new approximation being different by almost 4 points, rather than the other two being different by an entire order of magnitude less (about <= 0.1) Clearly the step size is incredibly important in deciding what method to utilize.

ADDITIONAL NOTES

Ideally, we should be going one step further and plotting the absolute error of all three methods against the exact values of the ordinary differential equation solution. We are, for now, only considering the differences between the approximations in a broad way above. To see the true tradeoffs in terms of step size, error from the exact, computation cost, and any other considerations pertinent to the problem to solve, more analysis should be completed.

CONCLUSION

The running thread through both the textbook exercises is to explore the importance of considering what terms may affect the approximation methods used to solve different problems.

In both handwritten and coding exercises, we explored Taylor series approximations and exactly how close they were to a truly accurate function provided. We can see from data that, while the number of points, or step size, evaluated within the approximation did not affect the end result accuracy, the number of terms was of high importance.

Further, we investigated a different problem through the same eyes: generating numerical solutions to ordinary differential equations. In this we can see the critical understanding of what potential variables contribute to inaccuracies–in this case, step size appears to be a key variable when deciding which methodology to use in our solution. ode45 (Runge-Kutta), Euler's scheme, and Euler's improved scheme are all inherently dependent on step size. While the methods appeared close in their approximations on a large scale, more analysis should be done on the actual error against the exact answer since there were notable differences when altering the step sizes.

Despite these being different problems to solve through various methods to generate approximate solutions, it is crucial that the implementer considers the tradeoffs and variables when deciding which method to use. As we have seen from the above explorations, we need to be careful in that choice, since the wrong one could jeopardize data and missions in varying severity.