Problem 1.

Find the degree 2 Taylor polynomial for $f(x) = e^x \sin(x)$, about the point $a = 0$. Bound the error in this approximation when $-\pi/4 \leq x \leq \pi/4$.

Solution

The second degree Taylor polynomial centered at 0 is given by

$$p_2(x) = f(0)(x-0)^0 + \frac{x-0}{1!}f^{(1)}(0) + \frac{(x-0)^2}{2!}f^{(2)}(0) \tag{1}$$

Compute

$$\begin{aligned} f^{(0)}(x) &= e^x \sin(x) & f(0) &= 0 \\ f^{(1)}(x) &= e^x(\cos(x) - \sin(x)) & f^{(1)}(0) &= 1 \\ f^{(2)}(x) &= 2e^x \cos(x) & f^{(2)}(0) &= 2 \\ f^{(3)}(x) &= -2e^x(\cos(x) - \sin(x)) \end{aligned}$$

So

$$p_2(x) = 0 + \frac{x}{1} + \frac{x^2}{2}2 = x + x^2 \tag{2}$$

Next bound the error term. Theorem 1.2.1 (Taylor's remainder theorem) can be utilized to establish a remainder as

$$R_2 = \frac{(x-0)^3}{3!}f^{(3)}(c_x) \tag{3}$$

for some c_x between 0 and x. To establish a bound, we must compute the smallest bound b such that

$$b \geq \max_{x \in [-\frac{\pi}{4}, \frac{\pi}{4}]} |f^{(3)}(x)|$$

(Ideally b is the least upper bound and there is a strict equality.) As $|f^{(3)}|$ is a (non-constant) continuous function, it must have a maximum value over an interval, either a local extrema or an endpoint (a result from calculus).

$$f^{(4)} = -4e^x \sin(x),\ f^{(4)} = 0 \implies x = 0$$

f has exactly one local extrema in the interval.

$$f^{(5)}(x) = -4e^x(\cos(x) - \sin(x)),\ f^{(5)}(0) = -4e^0(1-0) = -4 < 0$$

So the local extrema at $x = 0$ is a maximum. As no other extrema exist, neither interval endpoint can be the maximum value. Then $b = 1$, so

$$R_2 \leq \frac{(x-0)^3}{3!}2e^0(1-0) = \frac{x^3}{3} \tag{4}$$

Then the maximum error bound occurs at $x = \pi/4$, so

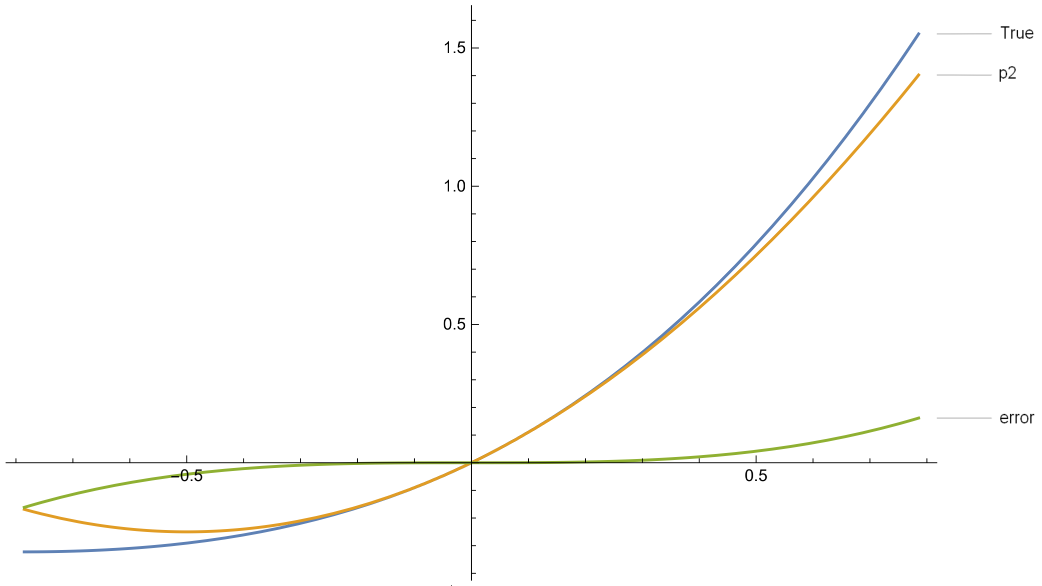
$$R_2 \leq \frac{1}{3}\left(\frac{\pi}{4}\right)^3 = \frac{\pi^3}{192} \approx 0.161491024376562$$

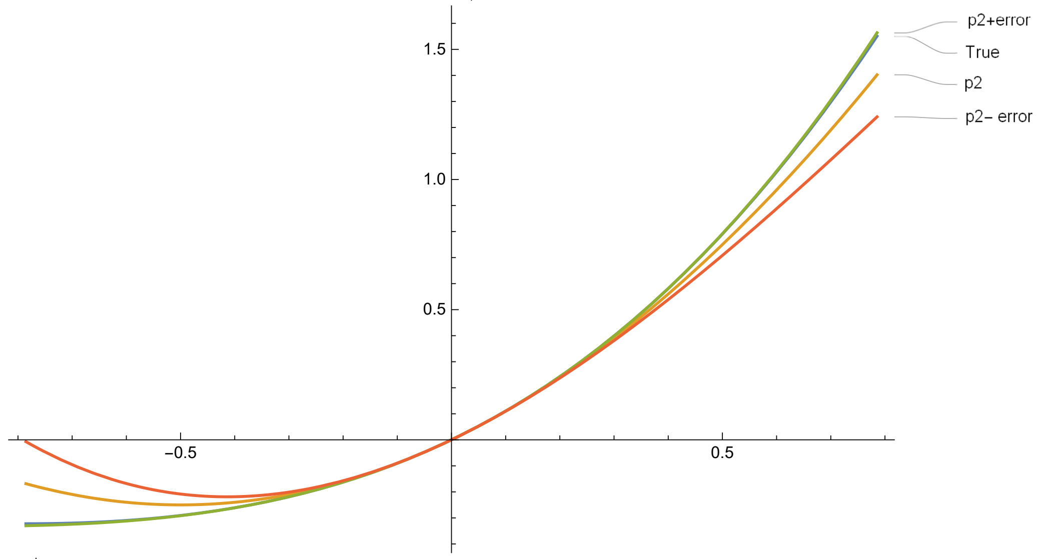
We can generate several plots (next page) to verify this result.

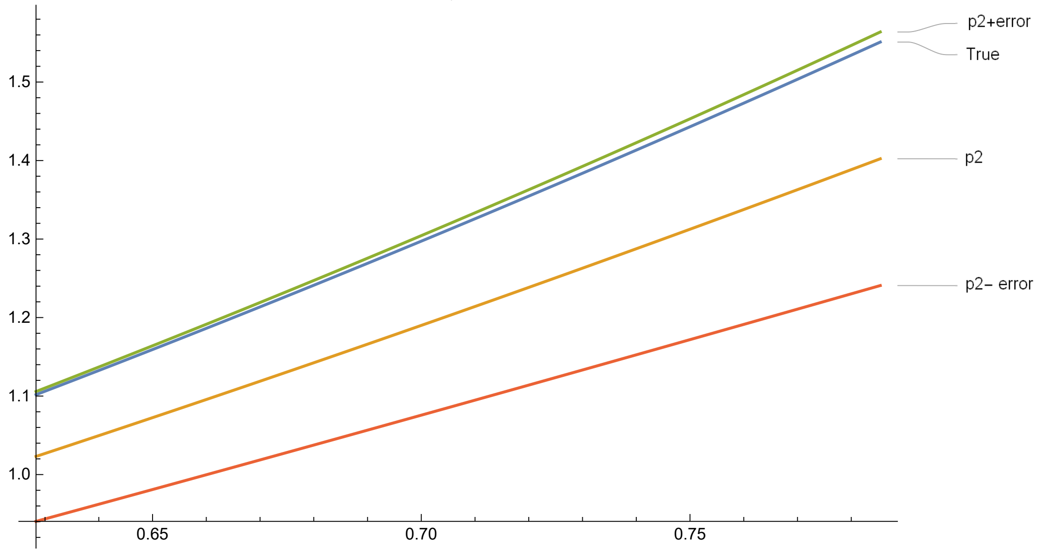
- True = $f(x)$

- p2 = $p_2(x)$

- error $= R_2(x)$


True
p2
error
1.5
1.0
0.5
-0.5
0.5


p2+error
True
p2
p2– error
1.5
1.0
0.5
-0.5
0.5


p2+error
True
p2
p2– error
1.5
1.4
1.3
1.2
1.1
1.0
0.65
0.70
0.75

Problem 2.

How large should the degree $2n-1$ be chosen such that

$$|\sin(x) - p_{2n-1}(x)| \leq 0.001, \quad x \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

Check your answer by evaluating the resulting p_{2n-1} at $x = \frac{\pi}{2}$.

Solution

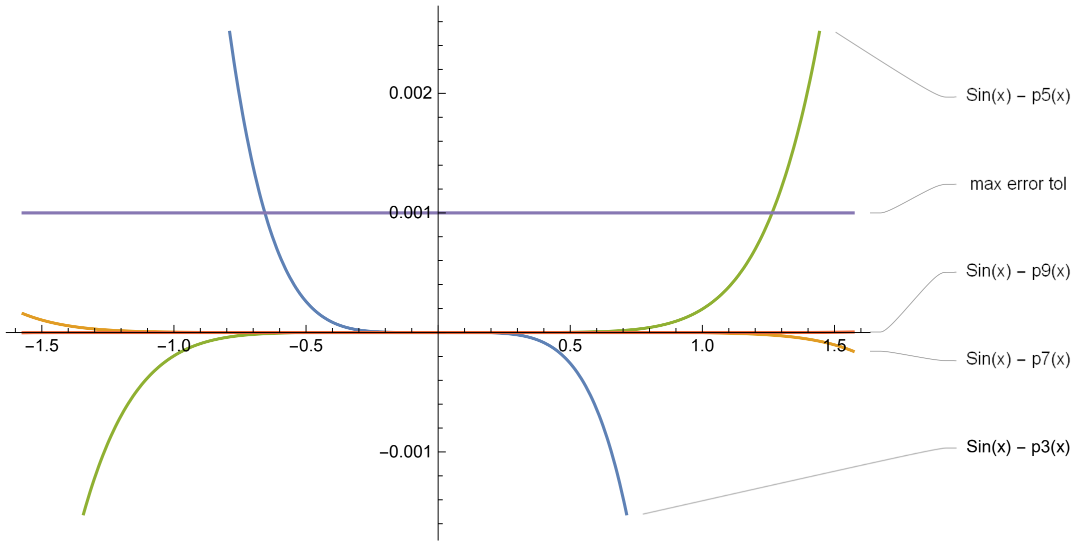
There are two possible approaches: plot $\sin(x) - p_{2n-1}(x)$ for $n = 1, 2, \ldots, N$ and determine the smallest n that satisfies the bound, or use the remainder theorem and establish a maximum possible error. We will use the first method to check our result from the later method.
The error term for p_{2n-1} is given in eqn. 1.14 (page 13) of the textbook as

$$|R_{2n-1}(x)| = \left|\frac{x^{2n+1}}{(2n+1)}\cos(c_x)\right| \leq \left|\frac{x^{2n+1}}{(2n+1)}\right|$$

The largest possible bound for the remainder occurs at the largest possible $|x|$. Define $g(n) = \frac{\left(\frac{\pi}{2}\right)^{2n+1}}{(2n+1)!}$. So $g(n) \geq R_{2n-1}\left(\frac{\pi}{2}\right)$. Then we can compute

$$g(1) = \frac{\left(\frac{\pi}{2}\right)^3}{(3)!} \approx 0.6454 > 0.001$$
$$g(2) = \frac{\left(\frac{\pi}{2}\right)^5}{(5)!} \approx 0.0796. > 0.001$$
$$g(3) = \frac{\left(\frac{\pi}{2}\right)^7}{(7)!} \approx 0.004681 > 0.001$$
$$g(4) = \frac{\left(\frac{\pi}{2}\right)^9}{(9)!} \approx 0.000164441 < 0.001$$

So a $2(4) - 1 = 7$th degree polynomial meets our error bound. We can test our bound numerically by computing $p_7(\pi/2) \approx 0.9998431013994987$, so the true error is approximately 0.0001568986005012774. We can then plot the true error terms to see that the bound works



And we see that p_7 meets the desired tolerance.

Problem 3

Compute the following sum

$$\sum_{k=1}^{10} \frac{e^k}{\left(\prod_{n=1}^{k}(2n+1)\right)}$$

Solution

We need to use an iteration method with a for loop. Both of the following methods produce the desired answer by iteration of partial sums. There is no requirement to determine the intermediate values, however they are often useful to store to help debug your code.

An iterative method contains an update rule ($s_{n+1} := s_n + a_n$, with base case $s_1 = a_1$) and stores the result of each iteration. To obtain full points, your writeup must include a description of what your code 'does' and explain the structure of your algorithm. Style or efficiency of your code was not considered.

Method A

```
format long %set print format (does not change machine arithmetic)

s = 0;
s_values = zeros(10,1);
divisor = 1;

for index = 1:10
divisor = divisor * (2 * index + 1);
s = s + exp(index) / divisor;
s_values(index) = s;
end
disp(s_values)
disp(s)
```

```
0.906093942819682
1.398697682748392
1.589988510588275
1.647764330729168
1.662041690912196
1.665027066969418
1.665568073201916
1.665654579520203
1.665666955759837
1.665668557764946

1.665668557764946
```

Method B

Slightly more complicated, but still allows us to examine the iterates.

```
format long %set print format (does not change machine arithmetic)

s = zeros(10, 1);
divisor = ones(10, 1);
divisor(1) = 3;
for index = 2:10
% Compute all divisors first
divisor(index) = divisor(index - 1) * (2 * index + 1);
end
s(1) = exp(1)/divisor(1);
for index = 2:10
s(index) = s(index - 1) + exp(index)/divisor(index);
end

disp(s)
disp(s(10))
```

```
0.906093942819682
1.398697682748392
1.589988510588275
1.647764330729168
1.662041690912196
1.665027066969418
1.665568073201916
1.665654579520203
1.665666955759837
1.665668557764946

1.665668557764946
```