MA 3457 Discussion 1

October 26th, 2022

Taylor's Theorem

(a) Find the third Taylor polynomial $P_3(x)$ for the function $f(x) = (x-1)\ln x$ about $x_0 = 1$.

(b) Approximate $f(0.5)$ using $P_3(0.5)$ and write an expression for the error when your approximation.

Answers:

(a)

$f(x) = (x-1)\ln x,\ f(1) = 0$

$f'(x) = \ln x + 1 - \dfrac{1}{x},\ f'(1) = 0$

$f''(x) = \dfrac{x+1}{x^2},\ f''(1) = 2$

$f'''(x) = -\dfrac{x+2}{x^3},\ f'''(1) = -3$

$f^{(4)}(x) = \dfrac{2(x+3)}{x^4}$

$$P_3(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2}(x-x_0)^2 + \frac{f'''(x_0)}{3!}(x-x_0)^3 = (x-1)^2 - \frac{1}{2}(x-1)^3$$

$$P_3(x) = (x-1)^2 - \frac{1}{2}(x-1)^3$$

(b)

The actual error is:

$\text{Error} = |f(0.5) - P_3(0.5)| \approx 0.034074$.

To write an expression for the error using Taylor's Theorem, first write the general expression for the remainder:

$$R_3(x) = \frac{f^{(4)}(\xi)}{(n+1)!}(x-x_0)^4 = \frac{2(\xi+3)}{\xi^4}(x-1)^4,$$

where $\xi \in (1, x)$ if $x > 1$ or $\xi \in (x, 1)$ if $1 > x$.

Error $= |f(0.5) - P_3(0.5)| = |R_3(0.5)| = \frac{2(\xi+3)}{\xi^4}(-1/2)^4 = \frac{2(\xi+3)}{16\xi^4}$ where $\xi \in (0.5, 1)$.

Machine Epsilon

Machine epsilon is the maximum relative error for a specified rounding procedure. Commonly, this is determined as the smallest floating point number that when added to 1, results in a floating point number greater than 1. In this problem you are exploring what machine epsilon is in Matlab.

(a) Write a program that will find machine epsilon. One way to do this is set an initial value $x := 1$, and repeatedly change it by making it smaller, for example $x := x/2$, and check whether MATLAB can still recognize that it is something greater than 0, that is, checking to see if $x + 1 > 1$.

(b) Compare the result with the built in Matlab command `eps` that determines machine epsilon.

```
% first program
x = 1;
while x + 1 > 1

    x = x / 2;

end
x = 2 * x
```

```
x = 2.2204e-16
```

```
eps
```

```
ans = 2.2204e-16
```

This program gets the same result as the eps command. Why was it necessary to multiply by 2 after the while loop ended?

```
% second program
x = 1;
while x/2 + 1 > 1

    x = x / 2;

end
x
```

```
x = 2.2204e-16
```

```
eps
```

```
ans = 2.2204e-16
```

This program gets the same result as the eps command but avoids taking "one step too many" so there's no need to multiply by 2 after the while loop ends.

Truncation Error

The following four expressions are exactly equal to zero:

$$x_1 = \left|2000 - \sum_{k=1}^{20,000} 0.1\right|$$

$$x_2 = \left|2000 - \sum_{k=1}^{16,000} 0.125\right|$$

$$x_3 = \left|2000 - \sum_{k=1}^{10,000} 0.2\right|$$

$$x_4 = \left|2000 - \sum_{k=1}^{8,000} 0.25\right|$$

However, computers store floating point numbers with a binary representation and only finitely many digits, so most decimal representations have a small truncation error. This error is usually too small to be noticeable, but it accumulates if you add up many copies of that number. As a result, x_1, x_2, x_3, x_4 might not be exactly zero if you calculate them with a computer. Verify this using a Matlab program. Can you explain the differences in these values?

```
s = 0; % let 's' stand for the s to avoid collision with the command 'sum'
for k = 1:20000
s = s + 0.1;
end
x1 = abs(2000 - s)
```

```
x1 = 7.2350e-10
```

```
s = 0;
for k = 1:16000
s = s + 0.125;
end
x2 = abs(2000 - s)
```

```
x2 = 0
```

```
s = 0;
for k = 1:10000
s = s + 0.2;
end
x3 = abs(2000 - s)
```

```
x3 = 3.1764e-10
```

```


s = 0;
for k = 1:8000
s = s + 0.25;
end
x4 = abs(2000 - s)
```

```
x4 = 0
```