Matlab Basic Stuff:

- When you assign them, Matlab automatically makes space for new variables and chooses the right type of variable. It will even change the type of variable if necessary
- Undeclared variables will cause errors
- To eliminate all variables - 'clear all'
- We can force variables to take a particular type
- A long line can be split by putting '...' at the end of a line
- Commands can be stored in a file, and writing the name of the file in the command window executes all of the commands
- The default output is 4 decimal digits. We get more using 'format long' ('format short' to go back to 4)
- Basic Input/Output:
    - Basic input can be an explicit assignment or the input command that uses the keyboard
    - Output can be simply typing the variable name, or 'disp(name)'
    - Whitespace can be minimized using 'format compact'
    - We can also make nicely formatted output: see the lab
- Every function you've ever heard of, and many more, are built into Matlab. We can use documentation to find precise names.
- Standards are:
    - sin(x)
    - cos(x)
    - tan(x)
    - asin(x)
    - acos(x)
    - atan(x)
    - atan2(x, y) [radians]
    - exp(x)
    - log(x) [natural]
    - log10(x)
    - sqrt(x)
    - abs(x)
    - mod(x, y)
        - Modulus is the remainder when doing integer division (a = qb + r)
    - floor(x)
        - Goes down to the nearest integer
    - ceil(x)
        - Goes up to the nearest integer
- For Loops:
    - Commands so far are only executed once, for loops execute them multiple times
    - For var = a:b
        - Commands
    - End
    - $Var\ takes\ the\ values\ a,\ a\ +\ 1,\ a\ +\ 2,\ \ldots,\ b\ -\ \ each\ time\ the\ commands\ are\ executed$

- $Always\ make\ a\ \&\ b\ integers,\ with\ a\ \leq\ b$
- $If\ a\ >\ b,\ do\ nothing$
- Variable step size:
  - For var = initial:step:final
    - Commands
  - End

Taylors Theorem:

$Functions\ well\ -\ behaved\ at\ a\ point\ can\ be\ approximated\ by\ a\ polynomial$:

$f(x)\ =\ P_n(x)\ +\ R_n(x),\ where$

$$P_n(x)\ =\ \sum_{k=0}^{n}\frac{1}{k!}f^{(k)}(x_0)(x\ -\ x_0)^k$$

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

$$R_n(x)\ =\ f^{(n+1)}(c)\frac{(x-x_0)^{n+1}}{(n+1)!}\ \{x\ \leq\ c\ \leq\ x_0\}$$

$Replace\ x\ with\ (x\ +\ h)\ and\ x_0\ with\ x,\ so\ x\ -\ x_0\ =\ h,\ such\ that$

$$f(x\ +\ h)\ =\ f(x)\ +\frac{h}{1!}f'(x)\ +\frac{h^2}{2!}f''(x)\ +\frac{h^3}{3!}f'''(x)\ +\ldots+\frac{h^n}{n!}f^{(n)}(x)\ +\frac{h^{(n+1)}}{(n+1)!}f^{(n+1)}(c)\ \{x\ \leq\ c\ \leq\ x\ +\ h\}$$

$$Set\ n\ =\ 1:\ f(x\ +\ h)\ =\ f(x)\ +\ hf'(x)\ +\frac{h^2}{2}f''(c)$$

$$hf'(x)\ =\ f(x\ +\ h)\ -\ f(x)\ -\frac{h^2}{2}f''(c)$$

$$Equation\ 1:\ f'(x)\ =\frac{1}{h}(f(x\ +\ h)\ -\ f(x))\ -\frac{h}{2}f''(c)$$

$$Equation\ 2\ (Replace\ h\ with\ -\ h):\ f'(x)\ =\frac{1}{h}(f(x)\ -\ f(x\ -\ h))\ +\frac{h}{2}f''(c_2)\{x\ -\ h\ \leq\ c_2\ \leq\ x\}$$

$Equation\ 1\ is\ called\ 1st\ order\ forward\ difference\ approximation\ to\ f'(x)$

$Equation\ 2\ is\ called\ 1st\ order\ backward\ difference$

$Small\ h\ means\ small\ range\ of\ c,\ c_2,\ f''(c),\ f''(c_2)\ roughly\ constant,\ error\ is\ proportional\ to\ h$

$Halve\ h\ ->\ halve\ error$

$$h\ ->\frac{h}{10},\ \frac{error}{10}$$

$$Note\ \frac{d^2}{dx^2}(ax\ +\ b)\ =\ 0,\ so\ Equations\ 1\ \&\ 2\ are\ exact\ for\ linear\ functions$$

$Taylor's\ Theorem\ can\ approximate\ a\ function\ with\ a\ polynomial,\ and\ error/remainder\ has\ a\ given\ form$

$$Equation\ 3:\ Set\ n\ =\ 2:\ f(x\ +\ h)\ =\ f(x)\ +\ hf'(x)\ +\frac{h^2}{2}f''(x)\ +\frac{h^3}{6}f'''(c_3)$$

$$Equation\ 4\ (Replace\ h\ with\ -\ h):\ f(x\ -\ h)\ =\ f(x)\ -\ hf'(x)\ +\frac{h^2}{2}f''(x)\ -\frac{h^3}{6}f'''(c_4)$$

$$Equation\ 3\ -\ Equation\ 4:\ f(x\ +\ h)\ -\ f(x\ -\ h)\ =\ 2hf'(x)\ +\frac{h^3}{6}(f'''(c_3)\ +\ f'''(c_4)$$

$$Equation\ 5:\ f'(x)\ =\frac{1}{2h}(f(x\ +\ h)\ -\ f(x\ -\ h))\ -\frac{h^2}{12}(f'''(c_3)\ +\ f'''(c_4)$$

$Equation\ 5\ is\ the\ 2nd\ order\ central\ difference\ formula$

$Error\ is\ proportional\ to\ h^2,\ it\ is\ exact\ for\ quadratics$

$Equation\ 6\text{: }\ Set\ n = 3\text{: } f(x + h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f''''(c_5)$

$Equation\ 7\ (Replace\ h\ with\ -h)\text{: } f(x - h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) + \frac{h^4}{24}f''''(c_6)$

$Equation\ 6 + Equation\ 7\text{: } f(x + h) + f(x - h) = 2f(x) + h^2f''(x) + \frac{h^4}{24}(f''''(c_5) + f''''(c_6))$

$Equation\ 8\text{: } f''(x) = \frac{1}{h^2}(f(x + h) - 2f(x) + f(x - h)) - \frac{h^2}{24}(f''''(c_5) + f''''(c_6))$

$Equation\ 8\ is\ the\ 2nd\ order\ central\ difference\ formula\ for\ the\ 2nd\ derivative$

Functions and Modular Programming:

- So far, we have written code, saved it in a file, and then executed it as if we typed everything in the command window. However, we have also used some Matlab functions like sqrt, mod, and floor, which take input and give output. We can write our code that way as well.
    - Syntax: *function out = **function.name(in1, in2, …, inn)
    - *Commands, including 'out = …'
    - **Name of the m-file in the current directory
- Note that variables passed to functions don't change (only values are passed), and new memory is used for internal variables that are deleted or at the end of the output, more than one value, replace out with [out1, out2, …, outm]
- Note that "disp" only prints to the screen, it doesn't output anything we can use. In the code that calls the function, we need [out1, out2, …, outm] = … to get all the outputs.
- A good reason for functions is modular programming - Splits a task into smaller tasks that can be written and tested separately, and later combined.
- Once you have a function, you can just use it like the other functions, there is no need to write it again. (Ex. Which number from 1 to 911 has the largest sum of digits?)

Arrays:

- An array stores more than one value associated with a name - Remember sequences like:
  $s_i = \frac{1}{i},\ i = 1, 2, \ldots, 100\ (from\ Calc\ II,\ or\ vectors\ from\ Physics,\ Calc\ III,\ or\ Linear\ Algebra)$
- Instead of one "box" for the variable s, think of 100 in a row labeled $s_1, s_2, \ldots, s_{100}$, each with potentially different values.
- $[s_1, s_2, \ldots, s_{100}]$ is an example of a one-dimensional array/vector
- In Matlab, a variable can be assigned as a zero vector using s = zeros(numRows, numCols)
- Data can also be entered one element at a time, in a loop, or all at once.
- Matlab dynamically changes the size of arrays to match assignments.
- It is faster and often best to define space for an array using "zeros" rather than growing the array one at a time.
- Matlab arrays start with a first entry; other programming languages start with a zeroth entry.
- Is a sequence of numbers palindromic?

- [1, 2, 12, 2, 1] is
- [1, 2, 2, 3] isn't

Arithmetic:

- Arithmetic in base b is just like in base 10 but with different addition and multiplication tables.
  - Base 4 Ex.
    - Addition

+	0	1	2	3
0	0	1	2	3
1	1	2	3	10
2	2	3	10	11
3	3	10	11	12

    - Multiplication

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	10	12
3	0	3	12	21

    - $301 + 213 = 1120$
    - $301 * 213 = 131313$
- Computers use base 2 (binary)
  - Base 2 Ex.
    - Addition

+	0	1
0	0	1
1	1	10

    - Multiplication

*	0	1
0	0	0
1	0	1

    - $1 + 1 + 1 = 11$
- A binary digit is a bit
- The addition in each column takes 3 bits of input (carry) and outputs 2 bits.

    - Ex. $0110111 + 0010011 = 1001010.$
- Multiplication is just shifted addition, also very fast.
    - Ex. $11011 * 101 = 10000111$
- Subtraction by standard reallocation is very slow, but just as fast as addition when done by complements.
    - Reallocation Ex. $642 - 158 = 484$
    - Subtraction can also be done with equal additions and complements
        - Complement - All digits add to 9, then add 1
- In base 2, the complement of 0 is 1 and the complement of 1 is 0, then add 1
    - Ex. $000011011 - 000001101 \Rightarrow 000011011 + 111110010 \Rightarrow 1000001110$
    - $= 000001110 = 1110$
- Division in base 2 can be done by shifted subtraction, but the comparison is slow. Better: find $\frac{x}{y} = x * \frac{1}{y}, \frac{1}{y}$ by Newton's method.

Errors:

- There are many sources of error when solving scientific problems:
    - Models approximate reality
    - Errors in experimental data
    - Error in approximating functions
    - Round off due to finite computer precision
        - Given an exact value y and its approximation $y^* = sin(y)$:
            - The absolute error is $\delta y = |y - y^*|$
                - Measures magnitude relative to the precision
            - The relative error is $\varepsilon y = \frac{\delta y}{|y|} = \frac{|y-y^*|}{|y|} \approx \frac{|y-y^*|}{|y^*|}$, while $y \neq 0$
        - Computers use base 2, so most fractions aren't represented exactly
            - In Matlab:

              $3 * 5.4 \neq 16.2 \rightarrow (\frac{1}{5})_{10} = (0.2)_{10} = (0.001100110011\ldots)_2$

            - This means $\varepsilon y = 2^{-n}$ (n digits in mantissa)
        - **Round-off Propagation:**
            - Since numbers are approximated, the rules of algebra may be broken.
            - We've already seen that $3 * 5.4 \neq 16.2$, but the error is small (ex. 3-digit decimal mantissa)
            - $1 + 10^{-5} = 1.00001$, but $sn(1.00001) = 1.00 * 10^0$
            - So $1 + x = 1$, while $x \neq 0$ (small)
            - In general, $a + b = a$, while $b \neq 0$ when b has a magnitude much less than zero.
            - In general $a + (b + c) \neq (a + b) + c$, $\frac{(ab)}{c} \neq (\frac{a}{c})b \neq a(\frac{b}{c})$
            - Ex. $a = 1.23 * 10^0$, $b = 4.56 * 10^{-1}$, $c = 3.78 * 10^{-2}$ (exact)
                - $sn(a + sn(b + c))$

        - $= sn(1.23 + sn(0.456 + 0.0378))$
        - $= sn(1.23 + sn(0.4938))$
        - $= sn(1.23 + 0.494)$
        - $= sn(1.724)$
        - $= 1.72$
    - $sn(sn(a + b) + c)$
        - $= sn(sn(1.23 + 0.456) + 0.0378)$
        - $= sn(sn(1.686) + 0.0378)$
        - $= sn(1.69 + 0.0378)$
        - $= sn(1.7278)$
        - $= 1.73$
    - $1.72 \neq 1.73$
    - True value is $a + b + c = 1.7238,\ sn(1.7238) = 1.72$
    - In general, only some of the digits of a small number are combined with digits in a big number, so it's best to add from smallest to largest.
- **Multiplication:**
    - $Let\ x = x^* + \delta x,\ y = y^* + \delta y$
        - Where:
            - $x\ \&\ y\ are\ the\ true\ values$
            - $x^*\ \&\ y^*\ are\ approximate\ values$
            - $\delta x\ \&\ \delta y\ are\ the\ error\ values$
    - $Then\ xy - x^*y^* = (x^* + \delta x)(y^* + \delta y) - x^*y^*$
        - $= x^*y^* + x^*\delta y + \delta x y^* + \delta x\delta y - x^*y^*$
        - $= x^*\delta y + \delta x y^* + \delta x\delta y$
            - $\delta x\delta y\ is\ very\ small,\ basically\ 0$
        - $= x^*\delta y + y^*\delta x$
    - $So\ \frac{xy - x^*y^*}{xy} \approx \frac{x^*\delta y}{xy} + \frac{y^*\delta x}{xy} \approx \frac{\delta x}{x} + \frac{\delta y}{y}$
    - Adds relative error - Same for division.
- **Addition:**
    - $Let\ x = x^* + \delta x,\ y = y^* + \delta y$
        - Where:
            - $x\ \&\ y\ are\ the\ true\ values$
            - $x^*\ \&\ y^*\ are\ approximate\ values$
            - $\delta x\ \&\ \delta y\ are\ the\ error\ values$
    - $Then\ (x + y) - (x^* + y^*) = x^* + \delta x + y^* + \delta y - x^* - y^* = \delta x + \delta y$
        - Adds absolute errors.
    - $So\ \frac{(x+y) - (x^* + y^*)}{x+y} = \frac{\delta x + \delta y}{x+y}$

- But if $x \approx- y$ relative errors can get very big (**Catastrophic Cancellation**)
  - Ex. $\sqrt{2} - 1.414213$ with 8-digit precision
    - $\approx 1.4142136 - 1.4142130 = 0.0000006$
    - $= 6.0 * 10^{-7}$
    - True value is $5.6237310 * 10^{-7}$ (lost 7 digits of accuracy)
  - Ex. Derivatives, $f(x) = x^2$, $f'(x) = 2x$, $f'(1) = 2$
    - $f'(1) = \lim_{h \to 0} (\frac{(1+h)^2 - 1^2}{h}) \approx \frac{(1+h)^2 - 1}{h}$
      - The numerator subtracts nearly equal numbers.

Fixed Point Iteration:

- $f(x) = 0 \Rightarrow x = g(x)$
  - Iterate $x_{n+1} = g(x_n)$
    - Hopefully converges
  - But finding g(x) is hard
    - When does it work?
- **Theorem:**
  - $Let\ p = g(p)\ such\ that\ p\ is\ a\ fixed\ point\ of\ g$
    - $If\ g\ is\ continuous,\ and\ if\ x \in [a,\ b],\ and\ if\ g(x) \in [a,\ b]$
      - $Then\ there\ is\ a\ fixed\ point\ in\ [a,\ b]$
    - $If\ |g'(x)| \leq k < 1$
      - $Then\ it\ is\ unique$
    - $If\ |g'(x)| \leq k < 1\ \forall x \in [a,\ b]$
      - $Then\ x_{n+1} = g(x_n)\ converges\ to\ p$
        - $Called\ an\ attractor$
    - $If\ |g'(x)| > 1$
      - $Then\ x_{n+1} = g(x_n)\ diverges$
        - $Called\ a\ repulsar$
- **Proof:**
  - **Existence**
    - $If\ g(a) = a\ or\ g(b) = b$
      - $[a,\ b]\ are\ the\ fixed\ points$
    - $Otherwise,\ g(a) > a\ and\ g(b) < b$
      - $Let\ h(x) = g(x) - x\ such\ that\ h(a) > 0\ and\ h(b) < 0$
      - $By\ IVT,\ there\ is\ at\ least\ one\ p \in [a,\ b]\ such\ that$:
        - $h(p) = 0$
        - $g(p) - p = 0$
        - $p = g(p)$
  - **Uniqueness**
    - $Assume\ for\ contradiction\ that$:

- $|g'(x)| \leq k < 1$
- $p = g(p)$
- $q = g(q)$
- $p \neq q$

■ $By\ MVT,\ \frac{g(p)-g(q)}{p-q} = g'(c)$

- $p \leq c \leq q$

■ $So\ |p-q| = |g(p) - g(q)| = |g'(c) * (p-q)| = |p-q| * |g'(c)| < |p-q|$

- $Contradiction:\ p = q\ (one\ fixed\ point)$

○ **Convergence**

■ $If\ x_{n+1} = g(x_n)$

■ $Then\ x_{n+1} - p = g(x_n) - p$

- $x_{n+1} - p = g(x_n) - g(p)$
- $x_{n+1} - p = \frac{g(x_n)-g(p)}{(x_n-p)} * (x_n - p)$
- $x_{n+1} - p = g'(c) * (x_n - p)$

■ $Let\ the\ error\ be\ e_n = x_n - p$

- $e_{n+1} = g'(c) * e_n$
- $|e_{n+1}| = |g'(c)| * |e_n|$
- $|e_{n+1}| < |e_n|$

<u>Root-finding:</u> f(x) = 0

- By IVT, if f is continuous on [a, b], and if f(a) and f(b) have opposite signs, then there is at least one $c \in [a, b]$ such that f(c) = 0.
- To reduce the interval, consider $m = \frac{a+b}{2}$
  - Based on the sign of f(n), either $c \in [a, m]\ or\ c \in [m, b]$
  - Ex. $f(x) = x^3 + 4x^2 - 10,\ f(1) = -5 < 0,\ f(2) = 14 > 0 \Rightarrow [1, 2]$
    - $f(1.5) = 2.375 > 0 \Rightarrow [1, 1.5]$
    - $f(1.25) = -1.7969 < 0 \Rightarrow [1.25, 1.5]$
- This is called the method of bisection because we bisect the interval
  - At every step, the width of the interval halves, so $e_{n+1} = \frac{1}{2}e_n$
  - Slow, but guaranteed to work if f(a) and f(b) have different signs

<u>Newton's Method:</u>

- If $f(x_0)$ is close to zero, hopefully x_1(where the tangent line crosses the x-axis) is better.
- Tangent line is: $y - f(x_0) = f'(x_0)(x - x_0)$
- If crosses x-axis at $(x_1, 0)$:
  - $0 - f(x_0) = f'(x_0)(x_1 - x_0)$

- $x_1 - x_0 = \frac{-f(x_0)}{f'(x_0)}$
- $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

- Ex. $\sqrt[k]{C}$: $f(x) = x^k - c = 0,\ f'(x) = kx^{k-1}$
  - $x_{n+1} = x_n - \frac{x_n^k - C}{kx_n^{k-1}} = \frac{1}{k}\left[(k-1)x_n + \frac{C}{x_n^{k-1}}\right]$
- Ex. $\sqrt[3]{10}$: $x_{n+1} = \frac{1}{3}\left(2x_n + \frac{10}{x_n^2}\right)$
- **Convergence:**
  - Let $e_n = p - x_n$
  - Then $e_{n+1} = p - x_{n+1} = p - x_n + \frac{f(x_n)}{f'(x_n)}$ *or* $e_{n+1} = e_n + \frac{f(x_n)}{f'(x_n)}$
  - To explain Newton's Method convergence:
    - $f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(c)$
    - Taylor:
      - $f(p) = 0 = f(x_0) + (p - x_0)f'(x_0) + \frac{1}{2}(p - x_0)^2 f''(c)\ \{p \le c \le x_0\}$
      - Since $f(p) = 0$, then $f(x_n + e_n) = 0$
        - $(e_n = p - x_n)$
      - Or $f(x_n) + e_n f'(x_n) + \frac{1}{2}e_n^2 f''(c_n) = 0\ \{p \le c_n \le x_n\}$
      - So

        $e_{n+1} = p - x_{n+1} = p - x_n + \frac{f(x_n)}{f'(x_n)} = e_n + \frac{f(x_n)}{f'(x_n)} = e_n - \frac{e_n f'(x_n) - \frac{1}{2}e_n^2 f''(c_n)}{f'(x_n)}$

        - $= e_n - e_n + \frac{1}{2}e_n^2 \frac{f''(c_n)}{f'(x_n)} = \left[\frac{f''(c_n)}{2f'(x_n)}\right]e_n^2$
          - Square error at each step
- If there is a double root $(f'(p) = 0)$, Newton's Method is only linear, and you only get half the digits of accuracy.
- Has the disadvantage that you need f'(x) and x.
  - Not always available
- Idea:
  - Approximate f'(x) by the slope of the line through two points.
    - Specifically $(x_n,\ f(x_n))$, $(x_{n-1},\ f(x_{n-1}))$
  - Then $f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$
  - Applied to Newton's method, this is called the <u>secant method</u>.
- Newton: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
- Secant: $x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n+1})}{f'(x_n) - f(x_{n+1})}$

- If you store $f(x_{n+1})$ you only need one function evaluation per step
    - While Newton's requires 2
- Similar working to Newton shows $e_{n+1} \approx Ke_n e_{n-1}$
    - Or $e_{n+1} \approx Ke_n^{\phi}$, $\phi = \frac{1+\sqrt{5}}{2} \approx 1.608\ (golden\ ratio)$

Accelerating Convergence:

- A linearly converging sequence satisfies $e_{n+1} \approx Ke_n$
    - $\Rightarrow p - x_{n+1} \approx K(p - x_n)$
    - $\Rightarrow K \approx \frac{p-x_{n+1}}{p-x_n}$
    - If K is constant:
        - $K \approx \frac{x_{n+1}-p}{x_n-p} \approx \frac{x_n-p}{x_{n-1}-p}$
        - $\Rightarrow (x_{n+1} - p)(x_{n-1} - p) \approx (x_n - p)^2$
- $x_{n+1}x_{n-1} - x_n^2 \approx p(x_{n+1} - 2x_n + x_{n-1})$
    - $p \approx \frac{x_{n+1}x_{n-1}-x_n^2-2x_{n-1}x_n+x_{n-1}^2+2x_{n-1}x_n-x_{n-1}^2}{x_{n+1}-2x_n+x_{n-1}}$ (Catastrophic cancellation)
        - $= x_{n-1} - \frac{(x_n-x_{n-1})^2}{x_{n+1}-2x_n+x_{n-1}}$
- **This is called Aitken's Δ^2 method**
    - $\Delta x_{n-1} = x_n - x_{n-1}$
    - $\Delta^2 x_{n-1} = \Delta(\Delta x_{n-1}) = x_{n+1} - 2x_n + x_{n-1}$
- **Steffensen's Method:**
    - Used Aitken's method to turn linear convergence (or divergence) into quadratic convergence.
    - Given x_0, $x_1 = g(x_0), x_2 = g(x_1)$:
        - New $x_0\ is\ x_0 - \frac{(x_1-x_0)^2}{(x_2-2x_1+x_0)}$
- Ex. $x^3 + 4x^2 - 10 = 0\ near\ x_0 = 1.5$
    - $x = \frac{1}{2}\sqrt{10 - x^3}$
        - Converged slowly
- Ex. $x = x^3 - 4x^2 - 10 + x \Rightarrow diverges$

Linear Algebra:

- **Matrices:**
    - An mxn matrix is an mxn array where all elements are numbers.
    - We usually represent them with a capital letter.

- M_{mxn} is the set of all mxn matrices
- Ex. $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \in M_{2x3}$
- If mxm, then the matrix is square
- If $m = 1$, then the matrix is a row vector
- If $n = 1$, then the matrix is a column vector
- If $m = n = 1$, then the matrix is a scalar
- **Equality:**
  - If $A, B \in M_{mxn}$, $A = B \; iff \; a_{ij} = b_{ij} \; \forall i, j$
    - Matlab: a(i, j) == b(i, j)
- **Addition:**
  - If $A, B \in M_{mxn}$, $then \; C = A + B \in M_{mxn} \; s.t. \; c_{ij} = a_{ij} + b_{ij} \; \forall i, j$

Midterm Topics:

- Be able to write very simple Matlab commands and functions.
  - Use the computer to do so
  - Write in the solution (no email/canvas)
- Memorize and be able to use Taylor's Theorem
  - $f(x + h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \ldots + \frac{h^n}{n!}f^{(n)}(x) + \frac{h^{n+1}}{(n+1)!}f^{(n+1)}(c) \; \{x < c < x + h\}$
- Be able to convert numbers between base (integer/fraction).
  - Let's skip arithmetic in different bases.
- Be able to work with a computer approximation of a real number.
  - Stick to decimal (max/min, arithmetic, approximation)
- Errors:
  - $x + y = x \; for \; small \; y$
  - Catastrophic cancellation
- Skip logistics and chaos
- Fixed point iteration:
  - Do it in Matlab, and see what happens.
  - $x = g(x) \; converges \; when \; |g'(x)| < 1 \; near \; p, \; p = g(p)$
- Explain why IVT leads to the method of bisection, be able to do a few steps by hand
- Be able to graphically derive Newton's method
- Given Newton's method, use it to solve f(x) = 0 (using Matlab)
- Write solutions on scrap paper.
  - Order doesn't matter

Back to Numerical Differentiation:

- Remember $f'(x) = \frac{1}{2h}(f(x + h) - f(x - h)) - \frac{h^2}{6}f'''(c)$
- Let $f(x) = f^*(x) + e(x) \Rightarrow True = Approx + Error$
- Then $f'(x) = \frac{1}{2h}(f^*(x + h) + e(x + h) - f^*(x - h) - e(x - h)) - \frac{h^2}{6}f'''(c)$

- $f'(x) = \frac{1}{2h}(f^*(x+h) - f(x-h) + \frac{1}{2h}(e(x+h) - e(x-h)) - \frac{h^3}{6}f'''(c_1)$
- $|f'(x) - \frac{1}{2h}(f^*(x+h) - f(x-h)| = |\frac{1}{2h}e(x+h) - \frac{1}{2h}e(x-h) - \frac{h^3}{6}f'''(c_1)|$
- $\leq \frac{|e(x+h)|}{2h} + \frac{|e(x-h)|}{2h} + \frac{h^3}{6}|f'''(c_1)|$
- $\leq \frac{e}{h} + \frac{h^2M}{6}$, $where\ e \sim e(x)\ \&\ f'''(c_1) \sim M$
    - Catastrophic cancellation

- Idea: We want to keep h big enough that roundoff errors don't matter, so return to Taylor Series:
    - Equation 1: $f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f''''(x) + \ldots$
    - Equation 2: $f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f''''(x) + \ldots$
    - $\frac{Equation\ 1 - Equation\ 2}{2h} = \frac{f(x+h)-f(x-h)}{2h} = f'(x) + \frac{h^2}{3!}f'''(x) + \frac{h^4}{5!}f^{(5)}(x) + \frac{h^6}{7!}f^{(7)}(x) + \ldots$
    - So $f'(x) = \frac{1}{2h}(f(x+h) - f(x-h)) - \frac{h^2}{3!}f'''(x) - \frac{h^4}{5!}f^{(5)}(x) - \frac{h^6}{7!}f^{(7)}(x) - \ldots$
        - Error is the infinite sum of even powers of h
- Richardson: Let N(h) approximate an unknown M with parameter h
    - Equation 3: $M = N(h) + K_1h^2 + K_2h^4 + K_3h^6 + K_4h^8 + \ldots$
    - Equation 4: $M = N(\frac{h}{2}) + K_1\frac{h^2}{4} + K_2\frac{h^4}{4^2} + K_3\frac{h^6}{4^3} + K_4\frac{h^8}{4^4} + \ldots$
    - $4(4) - (3) * (4-1)M = [4N(\frac{h}{2}) - N(h)] + K_2h^4(\frac{1}{4^{1-1}}) + K_3h^6(\frac{1}{4^{2-1}}) + K_4h^8(\frac{1}{4^{3-1}}) + \ldots$
    - Since $4 = (4-1) + 1$
    - $\div (4-1)$: $M = N(\frac{h}{2}) + \frac{1}{4-1}(N(\frac{h}{2}) - N(h)) + K_2'h^4 + K_3'h^6 + K_4'h^8 + \ldots$
    - Let $N_1(h) = N(h)\ and\ N_2(h) = N_1(\frac{h}{2}) + \frac{1}{4-1}(N_1(\frac{h}{2}) - N_1(h))$
    - So Equation 5: $M = N_2(h) + K_2'h^4 + K_3'h^6 + K_4'h^8 + \ldots$
    - Then Equation 6: $M = N_2(\frac{h}{2}) + K_2'\frac{h^4}{4^2} + K_3'\frac{h^6}{4^3} + K_4'\frac{h^8}{4^4} + \ldots$
    - $4^2(6) - (5)$: $(4^2-1)M = 4^2N_2(\frac{h}{2}) - N_2(h) + K_3'h^6(\frac{1}{4^{2-1}}) + K_4'h^8(\frac{1}{4^{3-1}})$
    - $\div (4^2-1)$: $M = N_2(\frac{h}{2}) + \frac{1}{4^{2-1}}(N_2(\frac{h}{2}) - N_2(h)) + K_3''h^6 + K_4''h^8$