MAT 201: Review 1

Introduction to Matlab

- Variable assignment (e.g., \( w = [1, 2, 3; 4, 5, 6] \) assigns a \( 2 \times 3 \) matrix to \( w \))
- Colon to create a vector (e.g., \( z = 6 : -2 : 1 \) assigns \([6, 4, 2]\) to \( w \))
- Standard mathematical operations \(+\, -\, \ast\, /\)
- Element-wise mathematical operations \( \ast\, ./\, .^2 \)
- Program structure
  
  ```matlab
  function [fOutSum, fOutProd] = MyFunction(fIn1, fIn2)
  fOutSum = fIn1 + fIn2
  fOutProd = fIn1 .* fIn2
  end
  ```
- Many built-in functions (\texttt{zeros}, \texttt{ones}, \texttt{size}, \texttt{length}, etc)
- Input and output via semi-colon and \texttt{disp} with \texttt{sprintf}
- \texttt{for} loop example
  
  ```matlab
  x = 10 : 0.1 : 10.5;
  for i = 1 : length(x)
    disp(sprintf('x(%d) = %0.1f', i, x(i)));
  end
  ```
- \texttt{while} loop example
  
  ```matlab
  x = 1;
  while x < 5
    disp(sprintf('x = %d', x));
    x = x + 1;
  end
  ```
- \texttt{if} statement example
  
  ```matlab
  if i == 1
    disp(sprintf('Yay i is 1'));
  elseif i < 3
    disp(sprintf('i is less than 3'));
  else
    disp(sprintf('i is greater than or equal to 3'));
  end
  ```
2.1 Floating-Point Numbers

- Conversion from binary to decimal numbers
- Floating-point form (fl\(x\) = \(\sigma \cdot \bar{x} \cdot b^e\) with sign \(\sigma\), mantissa \(\bar{x}\), base \(b\), exponent \(e\)) and restrictions
  - IEEE single precision (32-bit)
    (1 bit \(\sigma\), 23 bits \(\bar{x}\) excluding leading 1, 8 bits \(e \in [-126, 127]\))
  - IEEE double precision (64-bit)
    (1 bit \(\sigma\), 52 bits \(\bar{x}\) excluding leading 1, 11 bits \(e \in [-1022, 1023]\))
- Accuracy of floating-point numbers (as not all numbers are representable)
  - Machine epsilon = \(\epsilon_{\text{mach}}\) = (smallest representable number above 1) - 1 = \(2^{-n}\)
    if have \(n\) bits for \(\bar{x}\) excluding leading 1 and at least \(\lfloor \log_2 n \rfloor\) bits for \(e\)
  - Largest integer such that all integers 1 to \(M\) are representable = \(M = 2^n\)
    if have \(n - 1\) bits for \(\bar{x}\) excluding leading 1 and at least \(\lfloor \log_2 n \rfloor\) bits for \(e\)
- Chop or round to approximate \(x\) by fl\(x\) with fl\(x\) = \(x \cdot (1 + \epsilon)\) with \(\epsilon\) dependent on \(x\)
  - For chopping \(-2^{-n+1} \leq \epsilon \leq 0\)
  - For rounding \(-2^{-n} \leq \epsilon \leq 2^{-n}\)

2.2 Error Types

- Absolute error = Err\((x_A) = x_T - x_A\)
- Relative error = Rel\((x_A) = \frac{x_T - x_A}{x_T}\)
- Sources of error (human, modelling, measurement, truncation, floating-point)
- Catastrophic cancellation (computation with low error numbers gives high error result)

2.3 Propagation of Error

- Propagated error in mathematical operations
  - Propagated error = \(E = (x_T \circ y_T) - (x_A \circ y_A)\)
  - Propagated error can be bounded through interval analysis
  - Propagated relative error = Rel\((x_A \circ y_A) = \frac{(x_T \circ y_T) - (x_A \circ y_A)}{x_T \circ y_T}\)
  - Propagated relative error can be approximated using perturbation analysis
    \(x_T = x_A + \epsilon\) and \(y_T = y_A + \eta\)
  - Propagated error / propagated relative error occur as can’t represent inputs to an operation exactly
• Total computation error \( (x_T \circ y_T) - (x_A \circ y_A) = E + (x_A \circ y_A) - (x_A \circ y_A) \approx E \) occurs as also can’t represent the output of an operation exactly

• Error in function computation
  
  – Absolute error in function computation \( = f(x_T) - f(x_A) = f'(\xi) (x_T - x_A) \) with \( \xi \) some number between \( x_T \) and \( x_A \)
  
  – Relative error in function computation \( \approx \left( \frac{f'(\xi)}{f(x_T)} \right) (x_T) \text{Rel}(x_A) \) assuming \( x_T \approx x_A \) and \( f' \) well-behaved
  
  – Condition number \( \kappa \) represents inherent difficulty in computing \( f \)

2.4 Summation

• Summing \( n \) floating-point numbers \( a_i \) as \( S = \sum_{i=1}^{n} a_i \) involves \( n-1 \) addition operations, each with error

• Absolute error \( = S - S_n \approx -a_1(\epsilon_2 + \cdots + \epsilon_n) - \sum_{i=2}^{n} a_i(\epsilon_i + \cdots + \epsilon_n) \) with \( \epsilon_i \) arising from floating-point chopping/rounding error

• Summing in order from smallest to largest attempts to minimize error

1.1 Taylor Polynomials

• Often approximate a complicated function \( f \) by its Taylor polynomial \( p_n(x; a) = \sum_{i=0}^{n} \frac{f^{(i)}(a)}{i!} (x - a)^i \) based at \( a \)

1.2 Error in Taylor Polynomials

• Taylor remainder \( = R_n(x) = f(x) - p_n(x; a) \)

• If \( f \) has \( n+1 \) continuous derivatives on \( (\alpha, \beta) \) and \( a \in (\alpha, \beta) \) then there exists \( \xi \in (\alpha, \beta) \) with \( \xi \) between \( x \) and \( a \) (note, \( \xi \) is dependent on \( x \)) such that \( R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x-a)^{n+1} \)