Matlab Functions 2

ES115
Univ. Hartford, CETA
More Practice with Simple Functions

Let’s write a function that will convert temperature values given in the Fahrenheit scale to the Centigrade scale.

- $32^\circ F = 0^\circ C$
- $212^\circ F = 100^\circ C$

We can use the equation of a line to produce a conversion equation. Calculate the slope $m$, plug in the values for one temperature, and produce an equation for a line.
Working the Theory

- Start with
  \[ m = \frac{C_2 - C_1}{F_2 - F_1} \]

- Substitute only one set of values into
  \[ C_2 - C_1 = m(F_2 - F_1) \]

- Produce an equation that looks like this
  \[ C_2 = mF_2 - b \]

- Check your equation to make sure it’s right
Write the Function F2C.m

- Use previous notes to write a function that accepts F and returns C, based on your equation.
- Next write a script named F2C_test.m that tests your function.
- The script will produce a table with the left column with values 0, 20, 40, ..., 220.
- The right column will have the values in centigrade. Also plot the output.
Other plots

- Briefly try...
  - semilogx
  - semilogy
  - loglog

- These are outlined on page 155 of Moore.
Reconsider mysqrt.m

- A function is more useful if it can work with scalars or matrices.
- Why does mysqrt produce the wrong result for vectors?
- Let’s reconsider the algorithm:
  1. Start with $X_1 = 0$, $X_2 = 1$
  2. If $\text{abs}(X_2 - X_1)$ is larger than SMALL_VALUE then proceed to step 3 otherwise proceed to step 6
  3. Assign $X_1 = X_2$
  4. $X_2 = (X_1 + \text{value}/X_1)/2$
  5. Return to step 2
  6. $X_2$ is a reasonable approximation of the square root
The subroutine so far...

% mysqrt.m – Your name – The date
% My own square root function
function X2 = mysqrt(value)
    SMALL_VALUE = 1.0E-8;
    X1 = 0; X2 = 1;

    % Repeat until all values are close
    while abs(X2 - X1) > SMALL_VALUE
        X1 = X2;
        X2 = (X1 + value/X1)/2
    end

% end of mysqrt.m
Reconsider Step 1

- Step 1 provides initial values, like a guess, but it only makes sense when the value is a scalar
- Use the size function to assign the dimensions of ‘value’ to ‘value_size’
- Use the zeros function to assign a zeros matrix to X1
- Use the ones function to assign a ones matrix to X2
Reconsider Step 2

- The abs function works element by element, try it.
  
  ```
  >> abs([ -1 2 -3; 3 2 -1])
  ```

- The while statement is suspicious, what does this produce?
  
  ```
  >> [ 1 2 4 3 6 ] >= 4
  ```

- The comparison test produces a matrix??? Well, the value zero is `false`, and any non-zero value is `true`. **But** a matrix as a whole is `true` only when all its elements are `true`. This is a problem as the test produces a matrix with `true` and `false` values.
Reconsider Steps 2, 3, 4

- To cause the loop to repeat until all the values are close, we need a way to produce a single scalar `true/false` value based on whether or not the largest error is small... Ahh! Let’s use `max`.

```matlab
% Repeat until all values are close
while max(abs(X2 - X1)) > SMALL_VALUE
    X1 = X2;
    X2 = (X1 + value/X1)/2;
end
```

- What else is a problem here? Use `. /` to divide by the `X1` matrix. Try the new version.