Erin Obermayer
HW6 Pseudocode

Gradient Descent 1

- function
- Inputs: tolerance and gamma
- Output: N (# of iterations)
- Initialize:
    - N = 0
    - Error
    - Initial guess vector
- While loop (run until error > tolerance)
    - Compute the next iteration using gradient descent formula: x(n+1) = x(n) - gamma*gradient(x(n))
    - Compute error using the l2-norm
    - Update xn to be the new min
    - Update the number of iterations (add 1)
- End loop
- End function

Gradient Descent 2

- Same as gradient descent one EXCEPT:
    - Only one input (tolerance), so gamma is initialized in the function
    - Gamma is updated in the while loop using the Barzilai-Borwein step-size
    - I made a separate function to compute the gradient (gradx) with pseudocode:
        - Input: x vector
        - Output: gradient
        - Redefine the x vector into two components
        - Compute the first partial derivatives of the function wrt the two components (x and y)

Vary Step-Size Gamma to Optimize

- Make vector of gamma values to iterate through
- Initialize
    - Minimum N value (number of iterations)
    - Minimum gamma value
- For loop from i = 1 to number of gamma values in vector
    - Run gradient descent code for ith gamma value, store in N vector
    - If the new N value is less than the minimum value
        - Make it the new minimum N
        - Make the corresponding gamma the minimum gamma
    - End
- End
- Plot number of iterations vs gamma values

Vary Error Tolerances to Compare

- Create error tolerance vector to iterate through
- For loop from i = 1 to number of error tolerance values in vector
    - Run gradient descent 1 for ith error tolerance (and input gamma), store as vector
    - Run gradient descent 2 for ith error tolerance, and store as vector
- End
- Plot both number of iteration vectors vs error tolerance on semilogx graph
- Plot both number of iteration vectors vs error tolerance on loglog graph

Fit Sine Series

- Initialize
    - N (number of data points)
    - M (number of sine terms in linear least squares eq)
- Create x and y data points
- For loop from i = 1 to M
    - Create the ith column of the matrix A using the formula: sin(i*xData)
- End
- use the pseudo-inverse to compute beta vector
- Compute the l2-norm of the residuals
- Plot the data (both the y-data and the predicted y values, A*beta)

NONLINEAR LEAST SQUARES PSEUDOCODE CAN BE FOUND ON A PDF IN THE ZIP FILE