DEMONSTRATION OF THE STEEPEST GRADIENT DESCENT METHOD FOR SOLVING LINEAR SYSTEMS

- @file Gradient_Descent.m
- @brief

```
GRADIENT DESCENT - STEEPEST DESCENT DEMO SCRIPT

WE TRY TO SOLVE THE MINIMIZATION || y - Hx ||^2 USING THE
STEEPEST GRADIENT DESCENT METHOD

H is mxn, x is nx1 and y is mx1 and m > n
```

- @date 1/27/2019
- @author srijeshs

Contents

STEP 1: INITIALIZATION

```
clear all; clc;

% INITIALIZE THE PROBLEM DIMENSIONS
DIM_1 = 6;
DIM_2 = 4;

% GENERATE THE MATRIX H
H = rand(DIM_1,DIM_2);

% GENERATE THE TRUE VECTOR x
x_true = rand(DIM_2,1);

% GENERATE THE TRUE VECTOR y = Hx
y_true = H*x_true;

% INITIALIZE ESTIMATE OF X
x_est = ones(DIM_2,1).*5;

% INITIALIZE THE LEARNING RATE AND STOPPING THRESHOLD
lrate = 1e-1; haltThresh = 1e-6; nItermax = 1e+5;

% FUNCTION HANDLES FOR ERROR NORM, GRADIENT ALONG A DIMENSION
Err_Norm = @(Hmat,x_est,y_true) norm((y_true-Hmat*x_est),2);

% WE COMPUTE THE DERIVATIVE BY THE CENTRAL DIFFERENCE METHOD
Dim_Grad = @(Hmat,x_plus,x_minus,y_true,lrate) ...
            (Err_Norm(Hmat,x_plus,y_true) - Err_Norm(Hmat,x_minus,y_true))./2*(lrate);
```

STEP 2: BEGIN ITERATIONS

```
nIter = 0;
I_n = eye(DIM_2);

% INITIALIZE GRADIENT VECTOR
Grad_x = zeros(DIM_2,1);

while(norm(y_true - H*x_est)>=haltThresh && nIter <= nItermax)

    % UPDATE THE GRADIENT VECTOR BY COMPUTING GRAD ALONG EACH DIMENSION
    for iDim = 1:DIM_2

        % FORWARD INCREMENT AND BACKWARD DECREMENT VECTORS FOR THIS DIM
        xPlus  = x_est + (I_n(:,iDim)).*lrate;
        xMinus = x_est - (I_n(:,iDim)).*lrate;

        % GRADIENT ALONG THIS DIMENSION
        Grad_x(iDim) = Dim_Grad(H,xPlus,xMinus,y_true,lrate);

    end

    % UPDATE ESTIMATE OF X USING THE GRADIENT VECTOR CALCULATED
    x_est = x_est - Grad_x;

    % INCREMENT ITERATION COUNT
    nIter = nIter+1;
end
```

PRINT RESULTS

```
% ITERATION COUNT AND 2-NORM ERROR UPON TERMINATION
fprintf('No. of iterations = %d\nResidual norm = %f\n\n',nIter,norm(x_est-x_true,2));
fprintf('X_true:\n');
disp(x_true);
fprintf('X_est:\n');
disp(x_est);

% VERIFICATION WITH PSEUDO-INVERSE SOLUTION
fprintf('2-norm of error w.r.t pseudo inverse closed-form solution = %f\n',norm(x_est - (pinv(H)*y_true),2));
```

```
No. of iterations = 2126
Residual norm = 0.000003

X_true:
    0.2943
    0.1799
    0.9263
    0.0682

X_est:
    0.2943
    0.1799
    0.9263
    0.0682

2-norm of error w.r.t pseudo inverse closed-form solution = 0.000003
```

Published with MATLAB® R2018b