VISUALIZATION OF THE STEEPEST GRADIENT DESCENT METHOD FOR SOLVING LINEAR SYSTEMS

- @file Gradient_Visualize.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
```

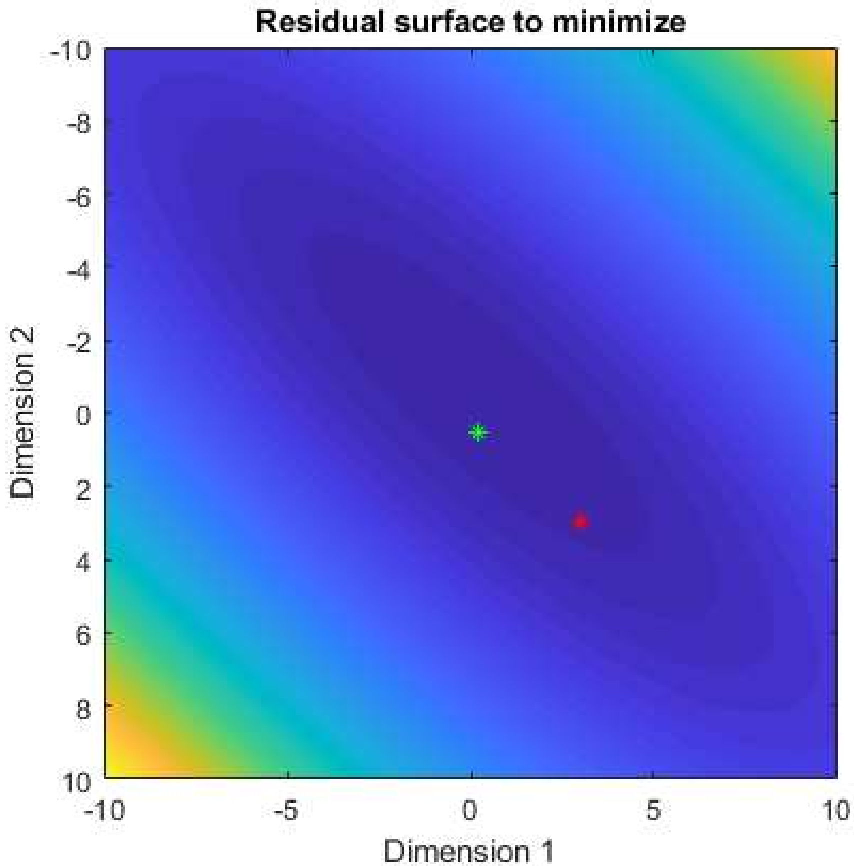
- @date 1/27/2019
- @author srijeshs

Contents

STEP 1: INITIALIZATION

```
clear all; clc;

% SETUP DIMENSIONS OF THE DEMO
DIM_1 = 2;
DIM_2 = 1;

% INITIALIZE X Y and H
H = rand(DIM_1);
cond_H = cond(H);

% FOR A VISUAL DEMONSTRATION, WE CHOOSE A RELATIVELY WELL-CONDITIONED H
while(cond_H > 5)
    H = rand(DIM_1);
    cond_H = cond(H);
end

X_true = rand(2,1);
Y = H*X_true;

% INITIALIZE THE RESIDUAL FUNCTION SPACE FOR A RANGE OF (X1,X2) PAIRS
residual_x = @(H,X1,X2,Y) [(Y(1) - X1.*H(1,1) + X2.*H(1,2)).^2 + (Y(2) - X1.*H(2,1) + X2.*H(2
,2)).^2];

% x1, x2 DISCRETE SPACE DEFINITION
x1 = linspace(-10,10,1000);
x2 = linspace(-10,10,1000);
[X2,X1] = meshgrid(x2,x1);
```

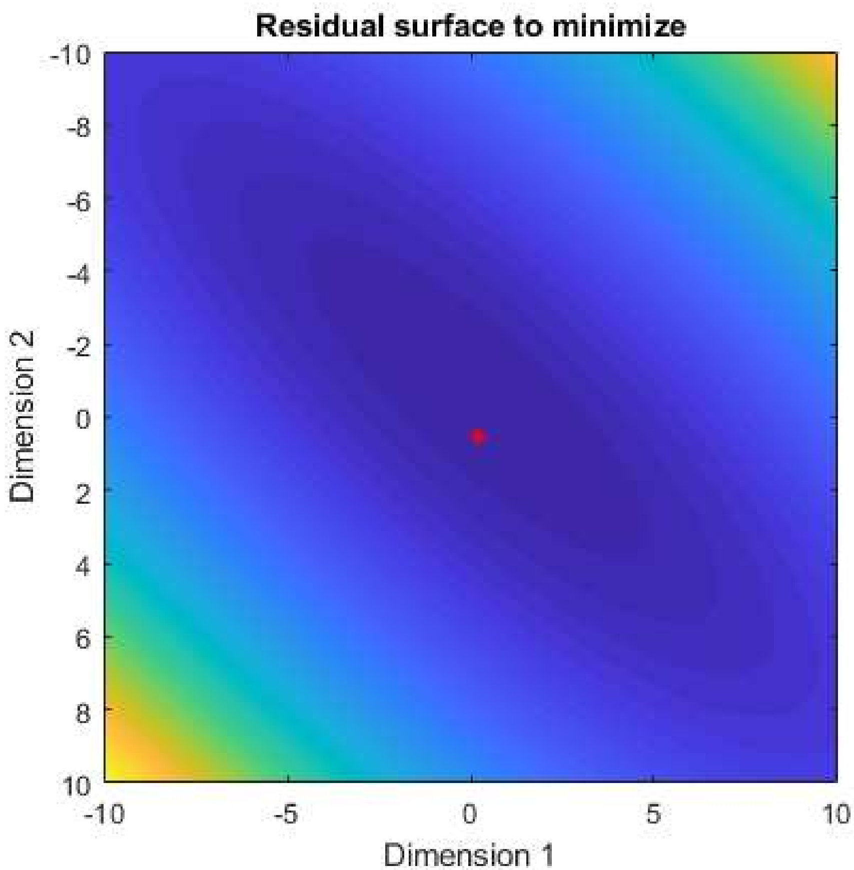
```
% FOR EACH (X1,X2) PAIR IN OUR SURFACE,
% COMPUTE RESIDUAL Y - HX, WITH X = [X1 ; X2]
residual_space = residual_x(H,X1,X2,Y);

% PLOT THE RESIDUAL SURFACE GENERATED W.R.T THE TWO DIMENSIONS
figure(1);
imagesc(x1,x2,residual_space');
axis equal tight;
title('Residual surface to minimize');
xlabel('Dimension 1');
ylabel('Dimension 2');
hold on;

% PLOT THE TRUE X VECTOR THAT OUR SOLUTION MUST CONVERGE TO
plot(X_true(1),X_true(2),'-*g');

% INITIALIZE ESTIMATE OF X
X_est = ones(2,1).*3;

% PLOT THE INITAL ESTIMATE OF THE X VECTOR
h0 = plot(X_est(1),X_est(2),'-*r');

% INITIALIZE THE LEARNING RATE AND STOPPING THRESHOLD
lrate = 1e-1; haltThresh = 1e-5; 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 = 1;
I_n = eye(DIM_1);
delete(h0);

% INITIALIZE GRADIENT VECTOR
Grad_x = zeros(DIM_1,1);

while(norm(Y - H*X_est)>=haltThresh && nIter <= nItermax)

    % PLOT CURRENT POSITION OF OUR ESTIMATE
    h1 = plot(X_est(1),X_est(2),'*r');
    pause(0.01); % Pause for visualization
    delete(h1);   % Delete this iteration's vector

    % UPDATE THE GRADIENT VECTOR BY COMPUTING GRAD ALONG EACH DIMENSION
    for iDim = 1:DIM_1

        % 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,lrate);

    end

    % UPDATE ESTIMATE OF X USING THE GRADIENT VECTOR CALCULATED
```

```
    X_est = X_est - Grad_x;

    % INCREMENT ITERATION COUNT
    nIter = nIter+1;
end

% PLOT THE FINAL CONVERGED SOLUTION
plot(X_est(1),X_est(2),'*r');
hold off;
```



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),2));
```

```
No. of iterations = 987
Residual norm = 0.000029

X_true:
```

```
    0.2124
    0.5433

X_est:
    0.2124
    0.5433

2-norm of error w.r.t pseudo inverse closed-form solution = 0.000029
```

Published with MATLAB® R2018b