Homework 2, #2

```
% % System of equations solved by Newton's Method
% % Bhairav Mehta, MATH371 HW2 W16
%

%Problem 2: fn = (x-1)^2 +y^2 = 4; xy = 1
% Jacobian: 2*(x-1) 2y
%            y       x

%%% 2a. Newton method for a convergent guess

% % fn handles, take a 2-vector x as input
fun = @(x) [(x(1)-1).^2 + x(2).^2 - 4; x(1).*x(2) - 1];
% %compute jacobian analytically
Jfun = @(x) [2*(x(1)-1) 2*x(2) ; x(2) x(1)];
%
% % try newton's starting from 3,0 with 5 steps
x0 = [3 0]'; tol=1e-10; nmax=5; verb=1;
[r,rn] = newton_method_nd(fun,Jfun,x0,tol,nmax,verb);

clear all;

%%% 2b. Newton method for a nonconvergent guess
%now a nonconvergent initial guess

% % fn handles, take a 2-vector x as input
fun = @(x) [(x(1)-1).^2 + x(2).^2 - 4; x(1).*x(2) - 1];
% %compute jacobian analytically
Jfun = @(x) [2*(x(1)-1) 2*x(2) ; x(2) x(1)];
%

x0 = [3 0]'; tol=1e-10; nmax=5; verb=1;
[r,rn] = newton_method_nd(fun,Jfun,x0,tol,nmax,verb);

clear all;

% fn handles, take a 2-vector x as input
fun = @(x) [(x(1)-1).^2 + x(2).^2 - 4; x(1).*x(2) - 1];
%compute jacobian analytically
Jfun = @(x) [2*(x(1)-1) 2*x(2) ; x(2) x(1)];

%creates a meshgrid of guesses that will be used in the algorithm
[x, y] = meshgrid(-1.95 : .25 : 3.95, -1.95 : .25 : 3.95);
tol=1e-10; nmax=20; verb=1;
X = [x(:) y(:)];
figure;
clear r rn;

%starts guessing
for i=1:length(X)
    x0 = X(i,:)';
    %runs current guess
    [r(:,i),rn{i}] = newton_method_nd(fun,Jfun,x0,tol,nmax,0);
```

```
    if r(1,i)>2.9714 && r(1,i)<2.9715 && r(2,i)<.337 && r(2,i)>.335
        %converges to root from part a
        plot(X(i,1),X(i,2),'.g');
        hold on;
    elseif r(1,i)>.5153 && r(1,i)<.5154 && r(2,i)>1.9403 && r(2,i) < 1.95
        %converges to .5154, 1.94309
        plot(X(i,1),X(i,2),'.b');
        hold on;
    else
        %did not converge
        plot(X(i,1),X(i,2),'.r');
        hold on;
    end
end

axis equal; axis tight; %fixes the axes
```

Homework 2, #2a output

```
% Homework 2 #2a

% |--n--|----xn----|---yn---|----|f(xn)|----|---|g(xn)|---|
% |--0--|3.0000000|0.0000000||0.0000000|-1.0000000|
% |--1--|3.0000000|0.3333333||0.1111111|0.0000000|
% |--2--|2.9716981|0.3364780||0.0008109|-0.0000890|
% |--3--|2.9714832|0.3365323||0.0000000|-0.0000000|>>
```

Homework 2, #2b explanation

```
% No it does not converge. It says the matrix is singular to working
precision
% which means that after the first step, the Jacobian matrix is updated and
% contains a row of all 0s or a column of all 0s, which makes the matrix rank
deficient.
% This happens when the determinant is = 0, which renders the matrix
uninvertible.
```

Homework 2, #3b

```
% % HW2, 3b and Extra Credit
% % Bhairav Mehta, MATH371 HW2 W16

%creates the matrix
A = [2 3 -1; 4 4 -3; -2 3 -1];

%assigns the lower and upper matrices based on matlabs lu fn
[L, U, p] = lu(A, 'vector');

%original b
b = [5 3 1]';
```

```
%permuted b
bp = b(p);

%solving with forward and backwards substituion.
y = L\bp;
x = U\y;

%the error and resuidual are both equal to 0 in this example.
%the example below is for extra credit, and we will be able to see
%when the error and residual are NOT equal to zero.

clear all;

%%%%%%%%%%%%%%%%%%%%%% Extra Credit %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%creates the matrix
A = [2.5 3 -1; 4 4 -3; -2 3 -1];

%assigns the lower and upper matrices based on matlabs lu fn
[L, U, p] = lu(A, 'vector');

%original b
b = [5 3 1]';

%permuted b
bp = b(p);

%solving with forward and backwards substituion.
y = L\bp;
x = U\y;
```

Homework 2, Extra Credit Explanation

```
% After a row reduction step, a value started to approach zero in the matrix
% introducing error into the system.
% As the calculations contained, the error snowballed, which is why
% both the error and residual are non zero.
%
```