A system of nonlinear equations

Newton's method

Levenberg-Marquardt method

Outline

- A system of nonlinear equations
- Matlab toolbox: fsolve
- fsolve by the Levenberg-Marquardt method
- Newton's method for nonlinear system solving
  - Updating rule
  - Matlab implementation
-

```
function F = myfun(x)
   F(1) =  x(1)^2 + x(2)^2-1;
   F(2) =  x(1)^2 - x(2)^2;
return
```

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 1$$

$$f_2(x_1, x_2) = x_1^2 - x_2^2$$

symbols

```
s1='x1^2+x2^2-1';
s2='x1^2-x2^2';
x1=sym('x1')
x2=sym('x2')
```

Inline Function

```
f=inline([str2sym(s1);str2sym(s2)]);
f(0,0)
```

fsolve

```
x=fsolve(@(x) [x(1)^2+x(2)^2-1 x(1)^2-x(2)^2],[1 1])
```

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0$$

$$f_2(x_1, x_2) = x_1^2 - x_2^2 = 0$$

```
x =

    0.7071    0.7071
```

```
s1='x1^2+x2^2-1';
s2='x1^2-x2^2';

x1=sym('x1')
x2=sym('x2')
f=inline([str2sym(s1);str2sym(s2)]
f(x(1),x(2))
```

```
ans =

   1.0e-11 *

    0.2282
         0
```

zeros

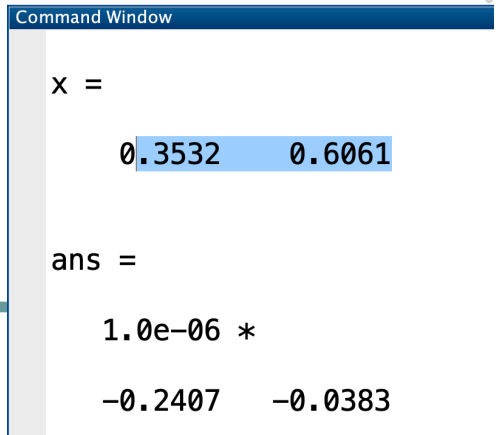
```
function demo_fsolve_0()
fun = @root2d;
x0 = [0,0];
x = fsolve(fun,x0)
root2d(x)

function F = root2d(x)

F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x(1)^2);
F(2) = x(1)*cos(x(2)) + x(2)*sin(x(1)) - 0.5;
```

$$e^{-e^{-(x_1+x_2)}} - x_2(1+x_1^2) = 0$$

$$x_1 cos(x_2) + x_2 sin(x_1) - \frac{1}{2} = 0$$



[1] Coleman, T.F. and Y. Li, "An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds," *SIAM Journal on Optimization*, Vol. 6, pp. 418-445, 1996.

[2] Coleman, T.F. and Y. Li, "On the Convergence of Reflective Newton Methods for Large-Scale Nonlinear Minimization Subject to Bounds," *Mathematical Programming*, Vol. 67, Number 2, pp. 189-224, 1994.

[3] Dennis, J. E. Jr., "Nonlinear Least-Squares," *State of the Art in Numerical Analysis*, ed. D. Jacobs, Academic Press, pp. 269-312.

[4] Levenberg, K., "A Method for the Solution of Certain Problems in Least-Squares," *Quarterly Applied Mathematics 2*, pp. 164-168, 1944.

[5] Marquardt, D., "An Algorithm for Least-squares Estimation of Nonlinear Parameters," *SIAM Journal Applied Mathematics*, Vol. 11, pp. 431-441, 1963.

[6] Moré, J. J., "The Levenberg-Marquardt Algorithm: Implementation and Theory," *Numerical Analysis*, ed. G. A. Watson, Lecture Notes in Mathematics 630, Springer Verlag, pp. 105-116, 1977.

[7] Moré, J. J., B. S. Garbow, and K. E. Hillstrom, *User Guide for MINPACK 1*, Argonne National Laboratory, Rept. ANL-80-74, 1980.

[8] Powell, M. J. D., "A Fortran Subroutine for Solving Systems of Nonlinear Algebraic Equations," *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, ed., Ch.7, 1970.

QUARTERLY
OF
APPLIED
MATHEMATICS
BROWN UNIVERSITY

Online ISSN 1552-4485; Print ISSN 0033-569X

A method for the solution of certain non-linear problems in least squares

An Algorithm for Least-Squares Estimation of Nonlinear Parameters

Donald W. Marquardt

Journal of the Society for Industrial and Applied Mathematics, Vol. 11, No. 2 (Jun., 1963), 431-441.

Stable URL:
http://links.jstor.org/sici?sici=0368-4245%28196306%2911%3A2%3C431%3AAAFLEO%3E2.0.CO%3B2-7

```
options = optimoptions('fsolve')
options = optimoptions('fsolve','Algorithm', 'levenberg-marquardt')
```

```
function demo_fsolve_1()
% problem setting
problem.x0 = [0,0];

display('fsolve by levenberg-marquardt');
options = optimoptions('fsolve','Algorithm', 'levenberg-marquardt')
```

```
FUN=@(x)root2d(x);
tic
x = fsolve(FUN,problem.x0,options);
toc
root2d(x)

end
```

```
function F = root2d(x)
F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x(1)^2);
F(2) = x(1)*cos(x(2)) + x(2)*sin(x(1)) - 0.5;
end
```

```
function demo_fsolve_2()
% problem setting
problem.x0=[0,0];
options = optimoptions('fsolve')
```

$$e^{-e^{-(x_1+x_2)}} - x_2(1+x_1^2) = 0$$

$$x_1 cos(x_2) + x_2 sin(x_1) - \frac{1}{2} = 0$$

```
FUN=@(x)root2d(x);
tic
x = fsolve(FUN,problem.x0,options);
toc
root2d(x)

end

function F = root2d(x)
F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x(1)^2);
F(2) = x(1)*cos(x(2)) + x(2)*sin(x(1)) - 0.5;
end
```

```
function demo_fsolve3()
% problem setting
x0 = [0,0,0];
options = optimoptions('fsolve','Algorithm', 'levenberg-marquardt')

display('fsolve by levenberg-marquardt');
c=[1 1 1 2 -1 1 1 3 -1];
f = @(x)root3d(x,c);
tic
x = fsolve(f,x0,options);
toc
root3d(x,c)

end

function F = root3d(x,c)
F(1) =c(1)*x(1)^2+ c(2)*x(2)^2+c(3)*x(3)^2-4;
F(2) =c(4)*x(1)+ c(5)*x(2)+c(6)*x(3)-1;
F(3) =c(7)*x(1)+ c(8)*x(2)+c(9)*x(3)-3;

end
```

$$x_1^2 + x_2^2 + x_3^2 = 4$$

$$2x_1 - x_2 + x_3 = 1$$

$$x_1 + 3x_2 - x_3 = 3$$

$$c_1x_1^2 + c_2x_2^2 + c_3x_3^2 = 4$$

$$c_4x_1 + c_5x_2 + c_6x_3 = 1$$

$$c_7x_1 + c_8x_2 + c_9x_3 = 1$$