
mathcad_homework_in_Matlab.m

Dr. Dave S#

Table of Contents

Basic calculations - solution to quadratic equation: $a*x^2 + b*x + c = 0$	1
Plotting a function with automated ranges and number of points	2
Plotting a function using a vector of values, with custom display	3
Using units and formatted display	4
Symbolic algebra	4
Symbolic calculus	5
Vector and matrix calculations	7
Programming a piecewise function	7
General programming problem example	8
Finding roots	9
Solving a set of nonlinear equations	10
Iterative calculations	12
Finding an optimal solution given constraints	13
Clean up windows (NOTE - I/O functions don't work in publish mode)	13

Basic calculations - solution to quadratic equation: $a*x^2 + b*x + c = 0$

```
clc                % clear the command window
clear             % clear all variables
close all        % close any existing windows
format compact % prevent extra blank lines in the output
display 'solution to quadratic equation:'
syms a b c x;
pretty (a*x^2 + b*x + c)
a=1, b=2, c=3
x_1st = (-b + sqrt(b^2 - 4*a*c)) / (2*a);
disp (['x_1st = ' num2str(x_1st)]);
x_2nd = (-b - sqrt(b^2 - 4*a*c)) / (2*a);
disp (['x_2nd = ' num2str(x_2nd)]);

% checking results in my_quadratic function:

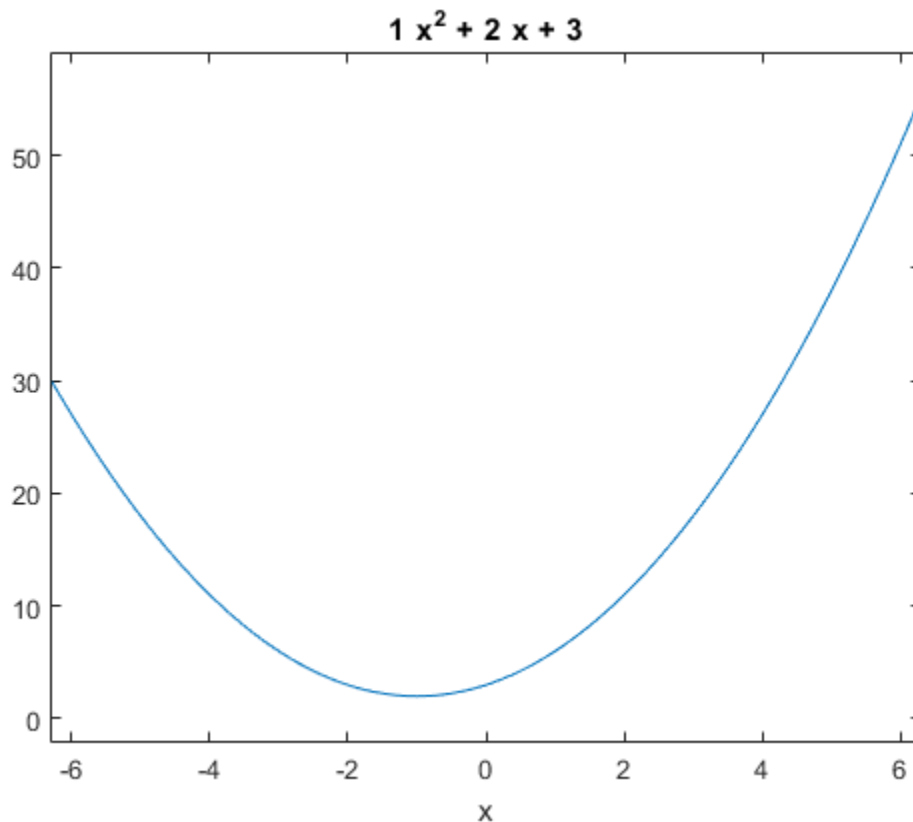
% function [f] = my_quadratic(x, a, b, c)
% % Function to evaluate the quadratic function with predefined a, b,
% c
% f = a*x.^2 + b*x + c;
% end

my_quadratic(x_1st, a, b, c);
```

```
disp(['f(x_1st) = ' num2str(my_quadratic(x_1st, a, b, c))]);  
my_quadratic(x_2nd, a, b, c);  
disp(['f(x_2nd) = ' num2str(my_quadratic(x_2nd, a, b, c))]);  
  
solution to quadratic equation:  
      2  
a x  + b x + c  
  
a =  
  1  
b =  
  2  
c =  
  3  
x_1st = -1+1.4142i  
x_2nd = -1-1.4142i  
f(x_1st) = -4.4409e-16  
f(x_2nd) = -4.4409e-16
```

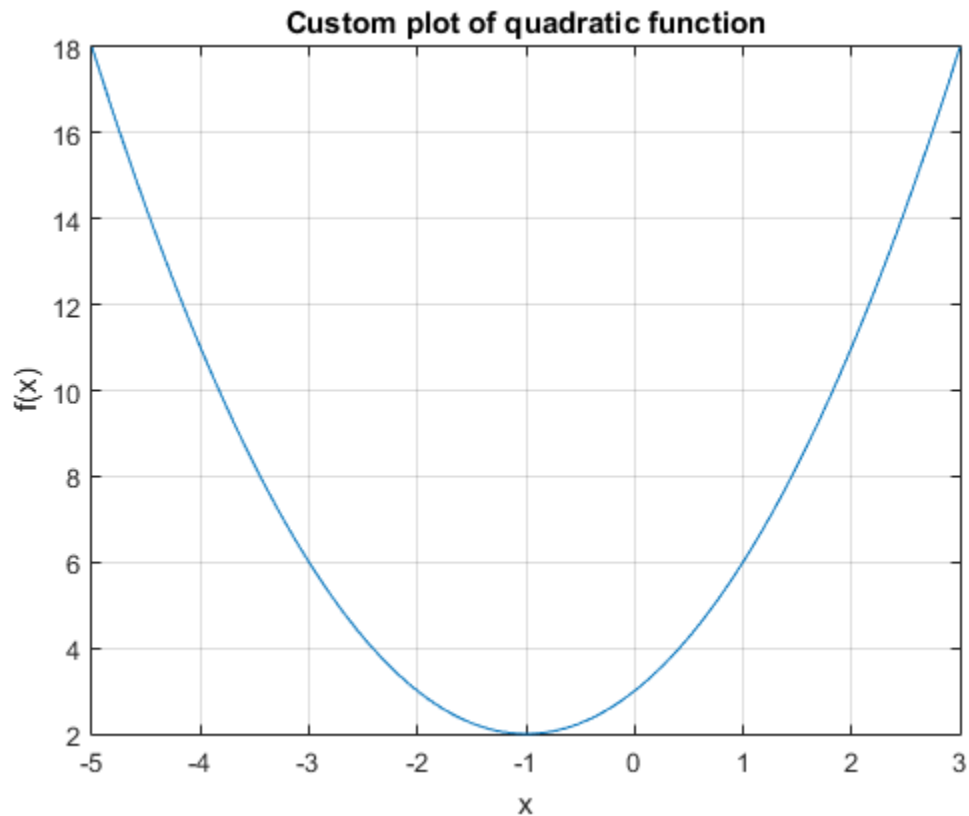
Plotting a function with automated ranges and number of points

```
ezplot('1*x^2 + 2*x + 3');  
snapnow; % causes plots to appear immediately during publish
```



Plotting a function using a vector of values, with custom display

```
figure      % open new figure window (to prevent previous from being
lost)
x = -5 : 0.05 : 3;
y = my_quadratic(x, a, b, c);
plot (x, y)
title ('Custom plot of quadratic function');
xlabel('x');
ylabel('f(x)');
grid on
snapnow;   % causes plots to appear immediately during publish
% Display both ends of x vector
x_length = length(x);
for i = 1:15
    x_lower(i) = x(i);
    x_upper(i) = x(x_length - 15 + i);
end
disp(' ');
x_lower
x_upper
```



```
x_lower =
```

```
Columns 1 through 7
-5.0000 -4.9500 -4.9000 -4.8500 -4.8000 -4.7500 -4.7000
Columns 8 through 14
-4.6500 -4.6000 -4.5500 -4.5000 -4.4500 -4.4000 -4.3500
Column 15
-4.3000
x_upper =
Columns 1 through 7
2.3000 2.3500 2.4000 2.4500 2.5000 2.5500 2.6000
Columns 8 through 14
2.6500 2.7000 2.7500 2.8000 2.8500 2.9000 2.9500
Column 15
3.0000
```

Using units and formatted display

(unit conversion functions available in Aerospace Toolbox only)

```
% m = convmass (100, 'lbm', 'kg');
m = 100 / 2.204622622; % conver lbm to kg
% v = convvel (60, 'mph', 'm/s');
v = 60 * 0.44704; % convert mph to mps
% a = convacc (20, 'ft/s^2', 'm/s^2');
a = 20 * 0.3048; % convert fps2 to mps2
p = m*v
F = m*a;
% convforce(F, 'N', 'lbf')
F = F / 4.448 % conver N to lbf

p =
1.2166e+03
F =
62.1650
```

Symbolic algebra

```
syms x y
eqn = x / (2*x - 3*x*y) == (x-2)^2/(y+2);
disp('solution:')
pretty (eqn);
x_ans = solve (eqn);
pretty (x_ans)
x_y = subs(x_ans, 'y', 5);
clear i;
eval(x_y(1))
eval(x_y(2))
```

solution:

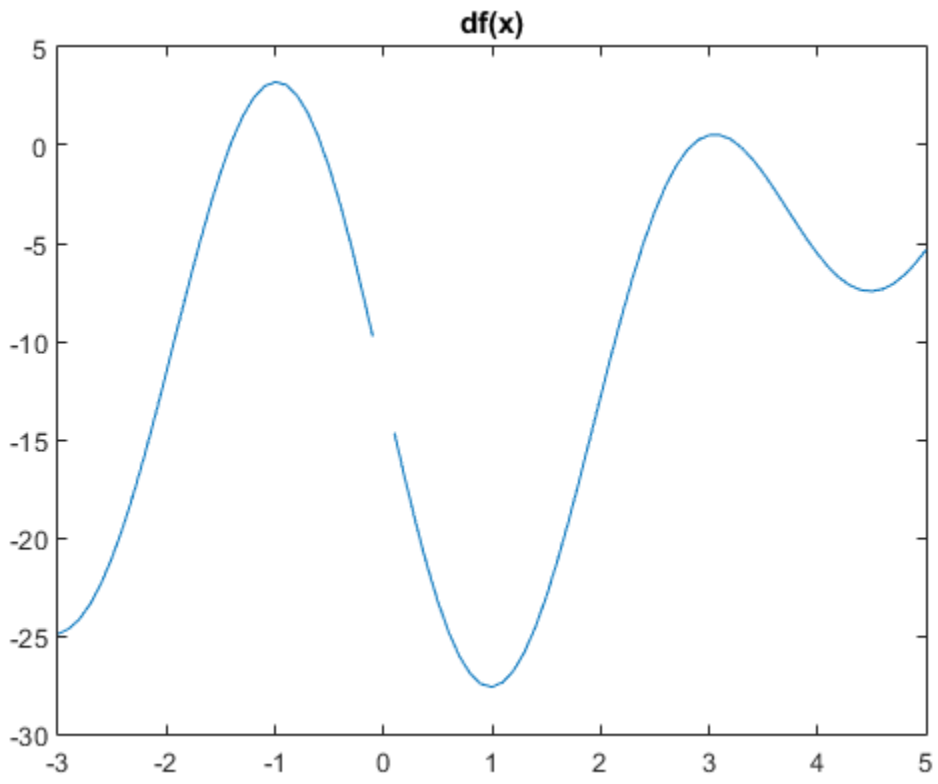
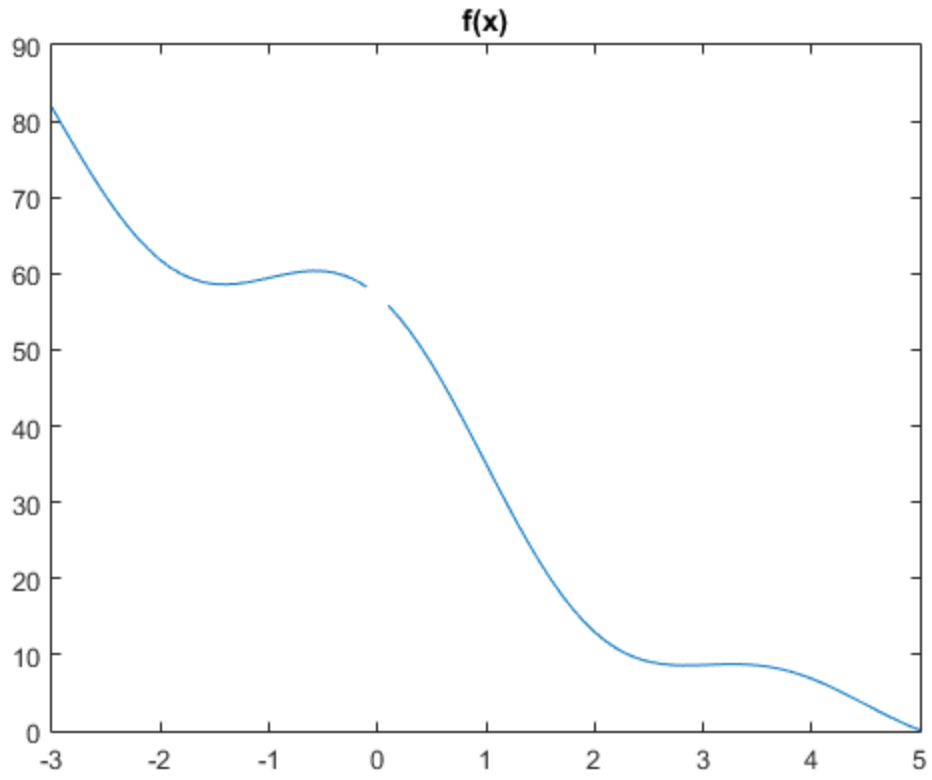
$$\frac{x}{2x - 3xy} == \frac{(x-2)^2}{y+2}$$

$$\frac{6y + \sqrt{-(3y-2)(y+2)} - 4}{3y-2} - \frac{\sqrt{-(3y-2)(y+2)} - 6y + 4}{3y-2}$$

```
ans =  
2.0000 + 0.7338i  
ans =  
2.0000 - 0.7338i
```

Symbolic calculus

```
a_copy = a;  
clear x a  
syms x a  
fx = (x - a)^2 + 10*sin(2*x)/x  
dfx = diff (fx)  
x = -3:0.1:5;  
a = a_copy;  
y = eval(fx);  
dy = eval(dfx);  
figure  
plot (x, y)  
title ('f(x)')  
snapnow;  
figure;  
plot (x, dy)  
title ('df(x)')  
snapnow;  
  
fx =  
(10*sin(2*x))/x + (a - x)^2  
dfx =  
2*x - 2*a + (20*cos(2*x))/x - (10*sin(2*x))/x^2
```



Vector and matrix calculations

```
disp(' ');
vx = -1; vy = -2;
v = [vx; vy]
v = vx + j*vy;
v_mag = abs(v);
display(['|v| = ' num2str(v_mag)]);
display(['v dot v = ' num2str(dot(v,v))]);
v
v_ang = angle(v)*180/pi;
display(['angle of v = ' num2str(v_ang) ' deg']);
display(['polar form of v = ' num2str(v_mag) ' < ' num2str(v_ang)]);

A = [1 2 3; 2 1 5; 0 -2 3]
disp('A^-1');
A_inv = inv(A)
display('A * A^-1');
I = A * A_inv

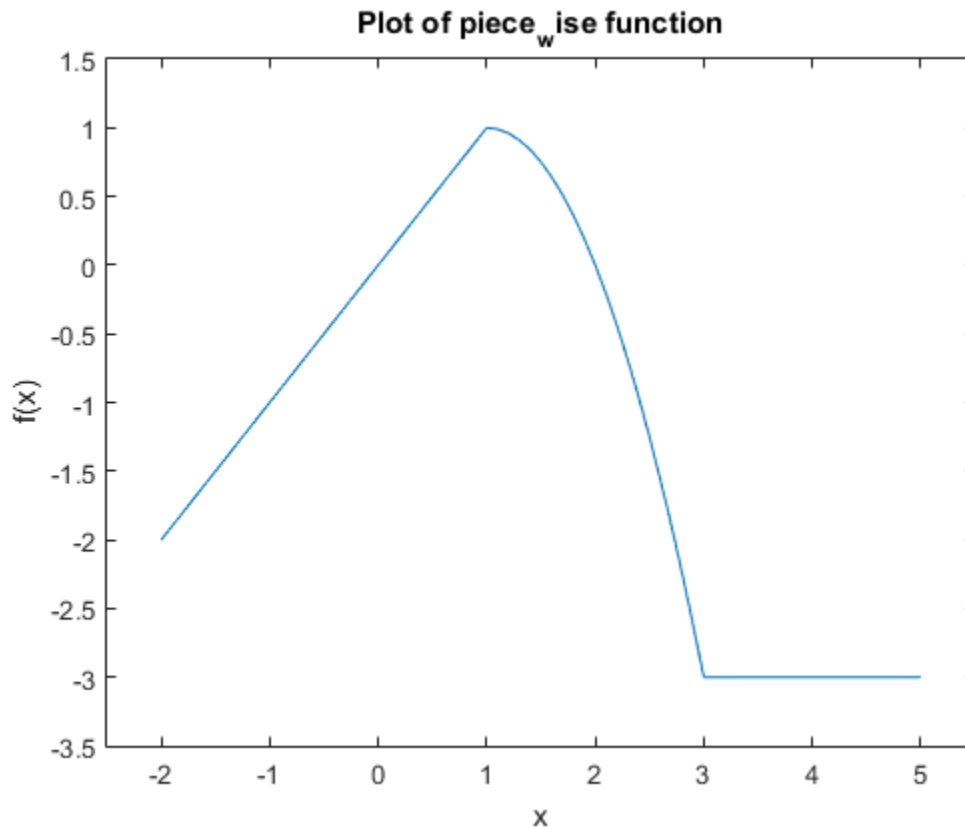
v =
    -1
    -2
|v| = 2.2361
v dot v = 5
v =
    -1.0000 - 2.0000i
angle of v = -116.5651 deg
polar form of v = 2.2361 < -116.5651
A =
     1     2     3
     2     1     5
     0    -2     3
A^-1
A_inv =
    -1.1818    1.0909   -0.6364
     0.5455   -0.2727   -0.0909
     0.3636   -0.1818    0.2727
A * A^-1
I =
     1.0000         0   -0.0000
         0     1.0000   -0.0000
         0         0     1.0000
```

Programming a piecewise function

```
% function [f] = my_piece_wise(x, a, b, c)
% % Function to evaluate the quadratic function with predefined a, b,
% c
% if (x < 1)
%     f = x;
% elseif ((x >= 1) && (x <= 3))
```

```
%      f = -(x-1)^2 + 1
% else
%      f = -3
% end

x = -2 : 0.1 : 5;
clear y;
for (i = 1 : length(x))
    y(i) = my_piece_wise(x(i));
end
figure;
plot (x, y);
title ('Plot of piece_wise function');
xlabel('x');
ylabel('f(x)');
axis([-2.5 5.5 -3.5 1.5]);
snapnow;      % causes plots to appear immediately during publish
```



General programming problem example

Find the sum of the first N numbers divisible by 3

```
% function [i total] = my_program(N)
% % Function to calculate the sum of the first N numbers divisible by
% 3
% i = 0;
```



```
% n = 0;
% total = 0;
%
% while (n < N)
%     i = i + 1;
%     remainder = mod (i, 3);
%     if (remainder == 0)
%         total = total + i;
%         n = n + 1;
%     end
% end
```

```
N = 10000;
display 'i total:'
[i total] = my_program(N)
```

```
i total:
i =
    30000
total =
    150015000
```

Finding roots

```
% function [f] = my_root_func(x)
% % Function to evaluate the quadratic function with predefined a, b,
% c
% f = 2*x^2 - 4*sin(x) - 2;
% end
```

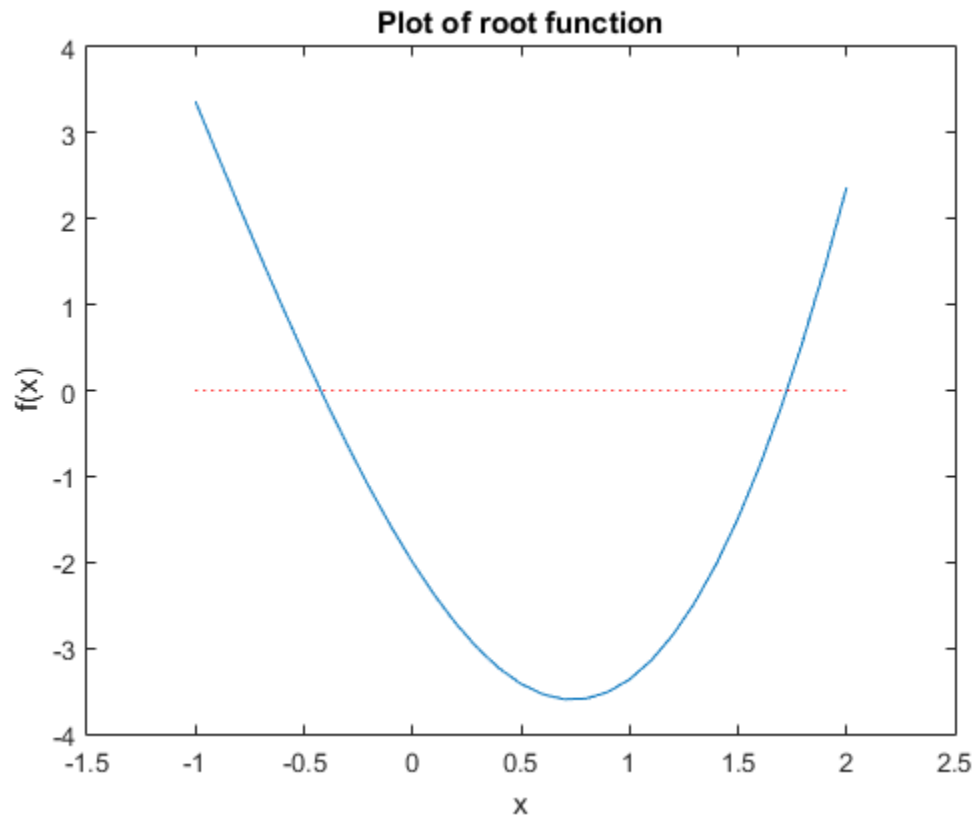
```
display 'f(x):'
syms x
pretty (2*x^2 - 4*sin(x) - 2);
display 'roots for different guesses:'
x0 = 1
fzero (@my_root_func, x0)
x0 = -1
fzero (@my_root_func, x0)
x = -1 : 0.1 : 2;
y = 2*x.^2 - 4*sin(x) - 2;
figure;
plot (x, y);
title ('Plot of root function');
xlabel('x');
ylabel('f(x)');
hold on;
x = [-1 2];
y = [0 0];
plot (x,y, 'LineStyle',':', 'Color',[1 0 0]);
axis([-1.5 2.5 -4 4]);
snapnow; % causes plots to appear immediately during publish
```

```
f(x):
    2
```

$$2x - 4\sin(x) - 2$$

roots for different guesses:

```
x0 =  
    1  
ans =  
    1.7252  
x0 =  
   -1  
ans =  
   -0.4230
```



Solving a set of nonlinear equations

```
syms x y;  
display 'solving:'  
pretty (x == 2 - y^2);  
pretty (y == sin(x)/x + x*y);  
  
% function [ F ] = my_non_linear_equations( x )  
% % Define set of nonlinear equations to be solved numerically  
% F = [x(1) - 2 + x(2)^2; x(2) - sin(x(1))/x(1) + x(1)*x(2)];  
% end  
  
x0 = [1; 1]; % initial guesses  
[x_sol,fval] = fsolve(@my_non_linear_equations,x0);
```

```
x_sol

% Checking results (solving symbolically and plotting)
syms x y;
fa = solve(x == 2 - y^2, y);
fb = solve(y == sin(x)/x + x*y, y);
display 'fa(x):'
pretty(fa);
display 'fb(x):'
pretty(fb);
x = x_sol(1)
display (['fa(x) = ' num2str(eval(fa(1)))]);
display (['fb(x) = ' num2str(eval(fb))]);
x = 0.01 : 0.04 : 0.5;
ya = eval (fa(1));
yb = eval (fb);
figure;
plot (x, ya);
hold on;
plot (x, yb, 'Color',[1 0 0]);
legend ('fa(x)', 'fb(x)');
```

solving:

$$x == 2 - y^2$$
$$y == \frac{\sin(x)}{x} + x y$$

Equation solved.

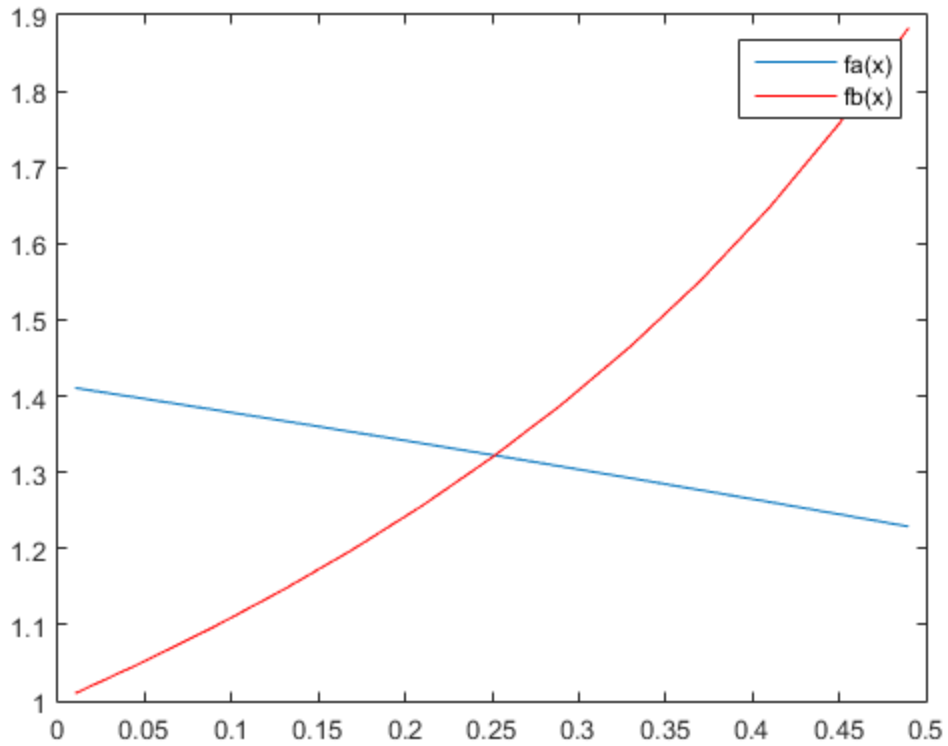
fsolve completed because the vector of function values is near zero as measured by the default value of the function tolerance, and the problem appears regular as measured by the gradient.

```
x_sol =
    0.2517
    1.3222
fa(x):
/ sqrt(2 - x) \
|              |
\ -sqrt(2 - x) /

fb(x):
sin(x)
-----
      2
- x  + x

x =
```

0.2517
 $fa(x) = 1.3222$
 $fb(x) = 1.3222$



Iterative calculations

```
clear x y;  
x(1) = 1, y(1) = 1  
for i = 1 : 6  
    x(i+1) = x(i) + 2;  
    y(i+1) = (x(i) + x(i+1)) / 2;  
end  
x  
y  
  
x =  
    1  
y =  
    1  
x =  
    1     3     5     7     9    11    13  
y =  
    1     2     4     6     8    10    12
```

Finding an optimal solution given constraints

(requires Optimization Toolbox)

```
% function [ F ] = my_objfun( x )
% % Function definition for constrained optimization problem
% % (minus sign in front for max vs. min)
% F = - ((x(1)-1)^2 - x(1)*sin(x(2)));
% end

% function [c, ceq] = my_confun(x)
% % Nonlinear inequality constraints
% c = [-x(1) - 2; x(1) - 2*x(2)^2 - 3; x(2) - 5; -x(2) - 3];
% % Nonlinear equality constraints
% ceq = [];

x0 = [1; 1];
[x,fval] = fmincon(@my_objfun,x0,[],[],[],[],[],[],[],@my_confun);
x
-fval    % minus for max vs. min
```

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to within the default value of the function tolerance, and constraints are satisfied to within the default value of the constraint tolerance.

```
x =
    -2.0000
     1.5708
ans =
    11.0000
```

Clean up windows (NOTE - I/O functions don't work in publish mode)

```
disp 'Hit Enter to close all windows and quit' pause
```

```
close all
```

Published with MATLAB® R2015a