```
%% mathcad_homework_in_Matlab.m                                  Dr. Dave  S#

%% 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))]);


%% 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


%% 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


%% 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))

%% 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;

%% 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


%% 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)


%% 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


%% 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)');


%% 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


%% 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


%% 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
```