# Output

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Problem 1  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
x=3.14  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Order 3

8.4466e-07

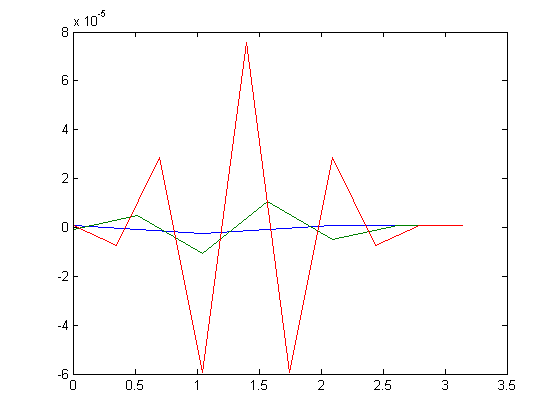
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Order 6  
  
 8.4557e-07  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Order 9  
  
 8.4466e-07

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Problem 2  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Values u = v = 0  
J =  
 1 0  
 0 -1  
  
calc\_eigs =  
 1  
 -1  
  
true\_eigs =  
 -1  
 1  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Values u =2, v = 0  
J =  
 -1 -2  
 0 1  
  
calc\_eigs =  
 -1  
 1  
  
true\_eigs =  
 -1  
 1  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Values u = 1 v = 1/2  
J =  
 -0.5000 -1.0000  
 0.5000 0

calc\_eigs =  
 -0.0000  
 -0.5000

true\_eigs =  
 -0.2500 + 0.6614i  
 -0.2500 - 0.6614i  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



# Driver.m

fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nProblem 1\n')  
fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nx=3.14\n')  
  
fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nOrder 3\n')  
Lagrange(@mysinc, 3.14159, 0, pi, 3)  
hold all  
fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nOrder 6\n')  
Lagrange(@mysinc, 3.14159, 0, pi, 6)  
hold all  
fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nOrder 9\n')  
Lagrange(@mysinc, 3.14159, 0, pi, 9)  
  
fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nProblem 2\n')  
fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nValues u = v = 0')  
J = Jacobian\_Matrix(0,0)  
calc\_eigs = QR\_Method(J, 20, 10^-16)  
true\_eigs = eig(J)  
  
fprintf('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nValues u =2, v = 0')  
J = Jacobian\_Matrix(2,0)  
calc\_eigs = QR\_Method(J, 20, 10^-16)  
true\_eigs = eig(J)  
  
  
fprintf('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nValues u = 1 v = 1/2')  
J = Jacobian\_Matrix(1,1/2)  
calc\_eigs = QR\_Method(J, 20, 10^-16)  
true\_eigs = eig(J)  
  
fprintf('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n')

# Jacobian\_Matrix.m

function [ J ] = Jacobian\_Matrix( u, v )  
%JACOBIAN\_MATRIX generates values for the jacobian of problem 2  
syms a b;  
equations = [((a - a^2/2) - a\*b); ((a-1)\*b)];  
J = jacobian(equations);  
J = subs(J, [a b], [u v]);  
  
end

# mysinc.m

function [ y ] = mysinc (x)  
if x == 0  
 y = 1;  
else  
 y = sin(x) / x;  
end

# Lagrange.m

function [garbage] = Lagrange( func, x, xmin, xmax, order )  
%LAGRANGE Generates a Lagrange polynomial for a given function.  
garbage = 0;  
h = (xmax - xmin)/order;  
p = 0;  
for i = 0:order  
 L = 1;  
 xcurr = xmin + i\*h;  
 for j = 0:order  
 if j ~= i  
 xj = xmin + j\*h;  
 L = L\*((x - xj)/(xcurr - xj));  
 end  
 end  
 p = p + func(xcurr)\*L;  
 M(i+1,2) = xcurr;  
 M(i+1,1) = p;  
end  
  
plot(M(:,2),M(:,1))

# QR\_Method.m

function [ eig ] = QR\_Method( A, max\_iter, tol )  
%QR\_METHOD - finds the eigenvalues of a given matrix using the QR method  
error = 999;  
A\_new = A;  
iter = 1;  
while (iter <= max\_iter && error > tol)  
 A\_old = A\_new;  
 [Q R] = myQR(A\_old);  
 A\_new = inv(Q)\*A\*Q;  
 error = abs((norm(A\_old)-norm(A\_new))/norm(A\_old));  
 iter = iter+1;  
end  
  
eig = diag(A\_new);  
  
end

# myQR.m

function [ Q R ] = myQR( A )  
%QR Decomposes a matrix into Q and R components  
  
[m n] = size(A);  
if m ~= n  
 error('Matrix is not square.')  
end  
  
Q = zeros(m);  
R = zeros(m);  
  
%Initialize Q's first row for recursion  
u = A(:,1);  
q = -u/norm(u);  
Q(:,1) = q;  
  
%Create matrix Q  
for i = 2:n  
 v = A(:,i);  
 dotp = 0;  
 for j = 1:i-1  
 q = Q(:,j);  
 dotp = dotp + dot(v,q)\*q;  
 end  
 u = v - dotp;  
 q = u/norm(u);  
 Q(:,i) = q;  
end  
  
%Create matrix R  
for k = 1:n  
 e = Q(:,k);  
 for l = 1:n  
 R(k,l) = dot(A(l,:),e);  
 end  
end  
  
end