**MSDM5004 HW1 solution**

**Q1**

**Part (1) – Iteration**

Iteration via Newton’s method:

Values from calculator:

**Part (2)(i) – Netwon’s method**

f = @(x) 4\*x\*sin(x) - 4\*(sin(x))^2-x^2;

Df = @(x) 2\*(x-2\*sin(x))\*(2\*cos(x)-1);

x0 = 1.5;

err = 100;

while err > 10e-5

x = x0-f(x0)/Df(x0);

err = abs(x-x0);

x0 = x;

end

disp(x0)

**Console display:**

1.895402774843726

**Part (2)(i) – Secant method**

Another point other than is required for secant method. Here is chosen.

f = @(x) 4\*x\*sin(x) - 4\*(sin(x))^2-x^2;

Df = @(x) 2\*(x-2\*sin(x))\*(2\*cos(x)-1);

x1 = 1.5;

x2 = 1;

while abs(x2-x1) > 10e-5

x\_temp = x2 - f(x2)\*(x2-x1)/(f(x2)-f(x1));

x1 = x2;

x2 = x\_temp;

end

disp(x2)

**Console display:**

1.895505103915596

**Q2**

First derive the Jacobian matrix

Observing that it can be simplified by taking ,

Finding the inverse is then simple:

Newton method is iterated as

Here is the code:

f = @(x1,x2) [1 + x1^2 - 4\*x2^2 + exp(x1)\*cos(2\*x2);

4\*x1\*x2 + exp(x1)\*sin(2\*x2)];

a = @(x1,x2) 2\*x1 + exp(x1)\*cos(2\*x2);

b = @(x1,x2) 4\*x2 + exp(x1)\*sin(2\*x2);

J = @(x1, x2) [a(x1,x2), -2\*b(x1,x2); b(x1,x2), 2\*a(x1,x2)];

x = [-1;2];

for i=1:5

x = x - (J(x(1),x(2)))\f(x(1),x(2));

disp([x(1),x(2)])

end

**Console display:**

-0.504703140620673 1.006023370838592

-0.374867197610368 0.654165113071481

-0.301307093881118 0.587881951811803

-0.293178007168211 0.586317181051050

-0.293162686798362 0.586329908945809

**Q3**

Lagrange interpolating polynomial:

Value at is

**Q4**

Least square polynomial of degree 1 is equivalent to doing linear regression.

Then

Thus the polynomial is

Error computed as

**Q5**

First compute and :

Eigenvalues of are and . So the singular values are and .

Using the eigenvectors associated with and with norm 1 to construct :

Thus the first 2 columns of are

The third vector is found by taking cross product of the previous 2, then normalize:

Finally we have

**Q6**

Let the least squares polynomial as . Let

Perform SVD on to get

Then compute and

Finally the least squares coefficients are (displaying in long format)

The least square error is

Here is the code for all the computations:

deg = 3;

x = [1.0, 1.1, 1.3, 1.5, 1.9, 2.1]';

b = [1.84, 1.96, 2.21, 2.45, 2.94, 3.18]';

A = [ones(size(x))];

for i=1:deg

A = [A, x.^i];

end

[U,S,V] = svd(A);

c = U'\*b;

z = zeros(deg+1,1);

for i=1:deg+1

z(i) = c(i)/S(i,i);

end

a = V\*z;

disp(a)

err = norm(c(deg+2:end,1));

disp(err)