Module 4 Solutions

Modern Navigation Systems – EN.525.645.81

Submitted: 10/2/2018

**Kyle Mercer**

1. Use Newton’s method to find the value of x for which the curve y = 2x + 3x2 is equal to 0.

Let x0 = 1

Indeed, the function y = 2x + 3x2 is equal to 0 when x = 0.

Now let x0 = -1

Indeed, the function y = 2x + 3x2 is equal to 0 when x = -0.6667 or 2/3.

1. For M = E - e sin E, solve for E when M = pi/2 and e = .3

Let E0 = 1:

Verifying our iterative solution for E = 1.8585

1. Rewrite the Newton’s method equations for the 2 dimensional case for yourself explicitly, without using matrix notation. Doing this yourself instead of simply being a spectator to the instructor’s example problems will be of significant help when completing assignments for future modules.

Setting up our equations for a circle centered at x0 and y0:

Next we write Newton’s equations for a vector:

Where JF(xn) is the Jacobian matrix of F(xn) and x is a vector.

So, for the input ranges, α ∈ [x,y] and β ∈[x,y], provided by two GPS satellites we have:

Let’s use these two equations to define the vector F(xn)

So the elements of JF(xn) is as follows:

1. Define two circles in a two-dimensional plane that overlap. Specify the centers of the two circles as points x1, y1 and x2, y2. Use Newton’s method to find the two points x01, y01 and x02, y02 at which the circles intersect. This is how GPS works. Specifically, the locations of the GPS satellites are known. A GPS receiver measures the distance to each satellite, thus defining a circle of position on the surface of the earth that corresponds to each satellite. The points at which the circles from multiple satellites intersect are computed, thus defining one’s location on the surface of the earth.

The following MATLAB script allows the user to enter in initial values x1, y1 and x2, y2 (defined as xa, ya, and xb, yb respectively) as well as a radius for each circle, ra and rb, as well as an initial guess, x0 and y0. The algorithm then iterates until the precision of the DeltaX of the Newtown equation becomes 0 (or the inverse of the Jacobian can’t be found).

The two circles chosen for this problem are as follows:

|  |
| --- |
| % Problem4:  %% User defined inputs  % Initial inputs for centers of two circles  xa = 5;  ya = 5;    xb = 10;  yb = 10;    ra = 5;  rb = 5;    % Enter guess for intersection x0 and y0  x0 = 4;  y0 = 5;    %% Algorithm starts here  % Iterative input  X = [x0; y0];  DeltaX = [Inf; Inf];    %% Iterative portion starts here  loop = 0;  while (norm(DeltaX) ~= 0 && ~isnan(norm(DeltaX)))    % Equations for our two circles are:  F1 = (X(1) - xa)^2 + (X(2) - ya)^2 - ra^2;  F2 = (X(1) - xb)^2 + (X(2) - yb)^2 - rb^2;  F = [F1; F2];    % Jacobian, J, for F(x0,y0) is:  j11 = 2\*(X(1) - xa);  j12 = 2\*(X(2) - ya);  j21 = 2\*(X(1) - xb);  j22 = 2\*(X(2) - yb);  J = [j11, j12; j21, j22];    % Newton's equation for a vector is defined as:  % J\*(x2 - x1) = -F  % Or Delta X = inv(J)\*-F    disp('==================');  loop = loop+1  % Calculate Delta X  DeltaX = J\-F % Equivalent to inv(J)\*-F;    % Calculate new X to feed into next iteration  X = X + DeltaX  disp('==================');  end |

The output of the script for the two intersecting points is shown below:

Initial guess for x0 and y0 was [1,6]. The intersection converged to [5,10]

|  |
| --- |
| >> problem4\_script  ==================  loop =  1  DeltaX =  0.8000  7.2000  X =  1.8000  13.2000  ==================  ==================  loop =  2  DeltaX =  2.3018  -2.3018  X =  4.1018  10.8982  ==================  ==================  loop =  3  DeltaX =  0.7795  -0.7795  X =  4.8813  10.1187  ==================  ==================  loop =  4  DeltaX =  0.1160  -0.1160  X =  4.9973  10.0027  ==================  ==================  loop =  5  DeltaX =  0.0027  -0.0027  X =  5.0000  10.0000  ==================  ==================  loop =  6  DeltaX =  1.0e-05 \*  0.1447  -0.1447  X =  5.0000  10.0000  ==================  ==================  loop =  7  DeltaX =  1.0e-12 \*  0.4185  -0.4192  X =  5  10  ==================  ==================  loop =  8  DeltaX =  0  0  X =  5  10  ================== |

Second guess was placed at [15,6]. The intersection now converged to [10,5]

|  |
| --- |
| >> problem4\_script  ==================  loop =  1  DeltaX =  -3.5556  -2.4444  X =  11.4444  3.5556  ==================  ==================  loop =  2  DeltaX =  -1.1800  1.1800  X =  10.2645  4.7355  ==================  ==================  loop =  3  DeltaX =  -0.2518  0.2518  X =  10.0127  4.9873  ==================  ==================  loop =  4  DeltaX =  -0.0126  0.0126  X =  10.0000  5.0000  ==================  ==================  loop =  5  DeltaX =  1.0e-04 \*  -0.3185  0.3185  X =  10.0000  5.0000  ==================  ==================  loop =  6  DeltaX =  1.0e-09 \*  -0.2029  0.2029  X =  10  5  ==================  ==================  loop =  7  DeltaX =  0  0  X =  10  5  ================== |