Sayısal Çözüm Yöntemleri Dersi Son Ödev

Alirıza Bilir 18014125

Eğri Uydurma Fonksiyonları:

Vandermonde Yöntemi

```
lineer.m × sor.m × gauss_seidel.m × jacobi.m × vandermondeTechnique.m × +
       function coefficients = vandermondeTechnique(x, y)
           n = length(x);
 2
 3
           if n ~= length(y)
 4
 5
               error('x ve y dizileri aynı uzunlukta olmalıdır.');
 6
 7
 8
           % Vandermonde matrisi oluşturma
           A = zeros(n, n);
 9
           for i = 1:n
10
11
               A(:, i) = x.^{(i-1)};
12
13
           % Katsayıları bulma
14
15
           coefficients = A \ y'; % Katsayıları bulmak için matris bölme işlemi
16
17
           % Sonuçları ekrana yazdırma
           disp('Polynomial coefficients:');
18
           disp(coefficients');
19
20
       end
```

Lagrange Yöntemi

```
lagrangeInterpolation.m × +
       function result = lagrangeInterpolation(x, y, targetX)
 1 -
 2
           n = length(x);
 3
            if n ~= length(y)
 4
 5
                error('x ve y dizileri aynı uzunlukta olmalıdır.');
 6
            end
 7
 8
            result = 0;
 9 🖹
            for i = 1:n
               term = y(i);
10
                for j = 1:n
11 🗀
                    if i ~= i
12
                        term = term * (targetX - x(j)) / (x(i) - x(j));
13
                    end
14
15
                end
                result = result + term;
16
17
18
            disp(['Interpolated value at x = ' num2str(targetX) ' is ' num2str(result)]);
19
20
```

Newton İnterpolasyon Yöntemi

```
numericalMethodsMenu.m 🗶 lineer.m 🗶 sor.m 🗶 gauss_seidel.m 🗶 jacobi.m 🗶 vandermondeTechnique.m 🗶 newtonInterpolation.m 🗶 secantMethod
       function result = newtonInterpolation(x, y, targetX)
           n = length(x);
 2
 3
 4
           if n ~= length(y)
 5
               error('x ve y dizileri aynı uzunlukta olmalıdır.');
 7
           % İleri farkları tablosu oluşturma
 8
           forwardDiffTable = zeros(n);
 9
10
           forwardDiffTable(:, 1) = y';
           for j = 2:n
               for i = 1:n-j+1
12
                   forwardDiffTable(i, j) = (forwardDiffTable(i+1, j-1) - forwardDiffTable(i, j-1)) \ / \ (x(i+j-1) - x(i)); 
13
               end
14
15
           end
16
17
           % Katsayıları ve interpolasyonu hesaplama
           coefficients = forwardDiffTable(1, :);
18
           result = coefficients(1);
19
20
           temp = 1;
           for i = 2:n
21
               temp = temp * (targetX - x(i-1));
22
               result = result + coefficients(i) * temp;
23
24
25
           disp(['Interpolated value at x = ' num2str(targetX) ' is ' num2str(result)]);
26
27
```

Lineer Denklem Sistemi Çözüm Yöntemleri:

Jacobi Yöntemi

```
numericalMethodsMenu.m × lineer.m × sor.m × gauss_seidel.m × jacobi.m × vandermondeTech
       function x = jacobi(A, b, x0, tol, max_iter)
           n = length(b);
 2
 3
           x = x0;
           iteration = 0;
 4
 5
           error_history = [];
 6
 7
           while iteration < max iter
 8
               x_old = x;
 9
               for i = 1:n
10
                  sigma = A(i, :) * x_old - A(i, i) * x_old(i);
11
                   x(i) = (b(i) - sigma) / A(i, i);
12
13
14
               iteration = iteration + 1;
15
               error = norm(x - x_old, inf);
16
               error_history = [error_history; error];
17
18
19
               if error < tol
20
                   break;
21
22
23
           fprintf('Jacobi method converged in %d iterations.\n', iteration);
24
25
           fprintf('Solution: ');
26
           disp(x);
27
28
           % Plot error vs iteration
29
           figure;
30
           semilogy(error_history);
           xlabel('Iteration');
31
32
           ylabel('Error');
33
           title('Error vs Iteration (Jacobi)');
34
```

Gauss Yöntemi

```
numericalMethodsMenu.m × lineer.m × sor.m × gauss_seidel.m × jacobi.m × vandermondeTechnique
        function x = gauss_seidel(A, b, x0, tol, max_iter)
 1 -
 2
            n = length(b);
           x = x0;
 4
           iteration = 0;
 5
           error_history = [];
 6
           while iteration < max iter
 7
 8
                x_old = x;
 9
 10
                for i = 1:n
                   sigma = A(i, 1:i-1) * x(1:i-1) + A(i, i+1:end) * x_old(i+1:end);
 11
                   x(i) = (b(i) - sigma) / A(i, i);
12
                end
13
14
15
                iteration = iteration + 1;
 16
                error = norm(x - x_old, inf);
                error_history = [error_history; error];
17
18
                if error < tol
19
20
                   break;
21
                end
22
 23
            fprintf('Gauss-Seidel method converged in %d iterations.\n', iteration);
24
25
            fprintf('Solution: ');
26
            disp(x);
27
 28
            % Plot error vs iteration
29
            figure;
 30
            semilogy(error_history);
            xlabel('Iteration');
 31
            ylabel('Error');
 32
            title('Error vs Iteration (Gauss-Seidel)');
 33
 34
```

SOR Yöntemi

```
numericalMethodsMenu.m × lineer.m × sor.m × gauss_seidel.m × jacobi.m × vandermondeTechnique.m
       function x = sor(A, b, x0, omega, tol, max_iter)
           n = length(b);
           x = x0;
3
4
           iteration = 0;
           error_history = [];
5
 6
           while iteration < max_iter
7
8
               x_old = x;
9
10
                for i = 1:n
                   sigma = A(i, :) * x - A(i, i) * x(i);
12
                    x(i) = (1 - omega) * x_old(i) + (omega / A(i, i)) * (b(i) - sigma);
13
14
               iteration = iteration + 1;
15
                error = norm(x - x_old, inf);
16
               error_history = [error_history; error];
17
18
               if error < tol
19
20
                   break;
21
                end
22
           end
23
24
           fprintf('SOR method converged in %d iterations.\n', iteration);
25
            fprintf('Solution: ');
26
           disp(x);
27
           % Plot error vs iteration
28
29
           figure;
           semilogy(error_history);
30
           xlabel('Iteration');
ylabel('Error');
31
32
           title('Error vs Iteration (SOR)');
33
34
```

Nonlineer Denklem Sistemi Çözüm Yöntemleri:

Sabit Nokta İterasyon Yöntemi

```
function [root, iterations] = fixedPointIteration(g, x0, tol, maxIter)
           iterations = 0;
2
3 🖨
           while iterations < maxIter
               x = g(x0);
4
               if abs(x - x0) < tol
5
 6
                  root = x;
7
                   return;
8
               end
9
               x0 = x;
10
               iterations = iterations + 1;
11
           end
12
           root = NaN;
       end
13
```

Newton Yönteimi

```
function [root, iterations] = newtonMethod(func, derivative, x0, tol, maxIter)
 2
           iterations = 0;
          while iterations < maxIter
3
4
              fx = func(x0);
 5
              if abs(fx) < tol
 6
                 root = x0;
 7
                  return;
 8
9
10
              fprime_x = derivative(x0);
11
              if fprime_x == 0
12
                 error('Derivative is zero. Division by zero error.');
              end
13
14
              x = x0 - fx / fprime_x;
15
              if abs(x - x0) < tol
16
17
                  root = x;
18
                  return;
              end
19
20
21
              x0 = x;
              iterations = iterations + 1;
22
23
           root = NaN;
25
       end
```

Nonlineer Denklem Çözüm Yötemleri:

Secant Yöntemi

```
function [root, iterations] = secantMethod(func, x0, x1, tol, maxIter)
 2
           iterations = 0;
          while iterations < maxIter
3 -
4
              fx0 = func(x0);
 5
              fx1 = func(x1);
              x = x1 - (fx1 * (x1 - x0)) / (fx1 - fx0);
 6
              if abs(x - x1) < tol
 7
 8
                 root = x;
9
                  return;
              end
10
11
              x0 = x1;
              x1 = x;
12
13
              iterations = iterations + 1;
          end
14
15
          root = NaN;
16
       end
```

Bisection Yöntemi

```
bisectionMethod.m × +
       function root = bisectionMethod(func, a, b, tol, maxIter)
1 🖃
            if func(a) * func(b) > 0
 2
               error('f(a) \ ve \ f(b) \ isaretleri \ farklı \ olmalıdır.');
 3
 4
 5
           iterations = 0;
 6
 7
           while iterations < maxIter
 8
                c = (a + b) / 2;
                if func(c) == 0 \mid \mid (b - a) / 2 < tol
 9
10
                    root = c;
11
                    return;
                end
12
13
                iterations = iterations + 1;
14
15
                if func(c) * func(a) < 0
                   b = c;
16
17
                else
18
                    a = c;
19
                end
           end
20
21
           root = NaN;
22
```

Menü

```
numericalMethodsMenu.m × lineer.m × sor.m × gauss_seidel.m × jacobi.m ×
                function numericalMethodsMenu()
                       while true
disp('Numerical Methods Menu:');
                               disp('1. Nonlinear Equation Solution');
disp('2. Nonlinear Equation System Solution');
disp('3. Linear Equation System Solution');
   5
                               disp('4. Curve fitting');
disp('5. Exit');
                               choice = input('Enter your choice: ');
switch choice
   10
11
   12
13
                                      case 1
                                              nonlinearEquationSolutionMenu();
  14
15
16
17
18
19
20
21
                                              nonlinearEquationSystemSolutionMenu();
                                       case 3
                                             linearEquationSystemSolutionMenu();
                                       case 4
                                             curveFittingMenu();
                                              disp('Programdan Cikiliyor');
  22
23
  24
25
                                              disp('Geçersiz Seçim');
  26
27
                function nonlinearEquationSystemSolutionMenu()
disp('Nonlinear Equation Solution Menu:');
disp('1. Fixed Point Iteration Method');
disp('2. Newton Method');
methodChoice = input('Enter your choice: ');
  28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
                        switch methodChoice
                                case 1
                                      disp('Fixed Point Iteration Method');
                                      disp( Fixes orant teration method );
g = input('Enter the function g(x): ');
x0 = input('Enter initial guess x0: ');
tolerance = input('Enter tolerance: ');
maxIterations = input('Enter maximum number of iterations: ');
   38
39
  40
41
                                       fixed PointIteration(g,\ x0,\ tolerance,\ maxIterations);
                                      e 2
disp('Newton Method');
func = input('Enter the function f(x): ');
derivative = input('Enter the derivative of f(x): ');
x0 = input('Enter initial guess x0: ');
tolerance = input('Enter tolerance: ');
maxIterations = input('Enter maximum number of iterations: ');
newtonMethod(func, derivative, x0, tolerance, maxIterations);
arratise
  42
43
44
45
46
47
   48
49
                               otherwise
  50
51
                                       disp('Geçersiz Seçim');
 52
```

```
52
 53
 54 🖃
         function nonlinearEquationSolutionMenu()
 55
            disp('Nonlinear Equation Solution Menu:');
 56
            disp('1. Bisection Method');
 57
            disp('1. Secant Method');
 58
            methodChoice = input('Enter your choice: ');
 59
            switch methodChoice
 60
                case 1
 61
                    disp('Bisection Method');
                    func = input('Enter function (örn: @(x) x^2 - 4;): ');
 62
 63
                     a = input('Enter lower bound: ');
 64
                     b = input('Enter upper bound: ');
 65
                     tol = input('Enter tolerance: ');
 66
                     maxIter = input('Enter maximum number of iterations: ');
 67
                    bisectionMethod(func, a, b, tol, maxIter);
                 case 2
 69
                    disp('Secant Method');
 70
                     func = input('Enter function (örn: @(x) x^2 - 4;): ');
 71
                     x0 = input('Enter initial guess x0: ');
 72
                    x1 = input('Enter initial guess x1: ');
 73
                     tol = input('Enter tolerance: ');
 74
                     maxIter = input('Enter maximum number of iterations: ');
 75
                     secantMethod(func, x0, x1, tol, maxIter);
 76
                 otherwise
 77
                     disp('Geçersiz Seçim.');
 78
             end
 79
        end
 80
 81 -
        function linearEquationSystemSolutionMenu()
 82
            disp('Linear Equation System Solution Menu:');
 83
            disp('1. Jacobi Method');
 84
            disp('2. Gauss-Seidel Method');
 85
            disp('3. SOR Technique');
 86
            methodChoice = input('Enter your choice: ');
 87
            switch methodChoice
 88
                 case 1
 89
                    disp('Jacobi Method');
 98
                    A = input('Enter matrix A: ');
 91
                    b = input('Enter vector b: ');
 92
                    x0 = input('Enter initial guess x0: ');
 93
                    tol = input('Enter tolerance: ');
 94
                    max_iter = input('Enter maximum number of iterations: ');
 95
                    jacobi(A, b, x0, tol, max_iter);
 96
                case 2
 97
                    disp('Gauss-Seidel Method');
                    A = input('Enter matrix A: ');
 98
                    b = input('Enter vector b: ');
 99
100
                    x0 = input('Enter initial guess x0: ');
101
                    tol = input('Enter tolerance: ');
102
                    max_iter = input('Enter maximum number of iterations: ');
                    gauss_seidel(A, b, x0, tol, max_iter);
103
104
                case 3
105
                    disp('SOR Technique');
106
                    A = input('Enter matrix A: ');
107
                    b = input('Enter vector b: ');
102
                    x0 = input('Enter initial guess x0: ');
109
                    omega = input('Enter relaxation factor (omega): ');
110
                    tol = input('Enter tolerance: ');
111
                    max_iter = input('Enter maximum number of iterations: ');
112
                    sor(A, b, x0, omega, tol, max_iter);
113
                otherwise
114
                    disp('Gecersiz Secim');
115
            end
116
        end
117
```

```
function curveFittingMenu()
119
              disp('Curve Fitting Menu:');
120
              disp('1. Vandermonde Technique');
121
              disp('2. Lagrange Interpolation');
122
              disp('3. Newton Interpolation');
123
              methodChoice = input('Enter your choice: ');
124
              switch methodChoice
125
                   case 1
126
                       disp('Vandermonde Technique');
127
                       x = input('Enter x values (e.g., [x1, x2, x3]): ');
128
                       y = input('Enter corresponding y values (e.g., [y1, y2, y3]): ');
129
                       vandermondeTechnique(x, y);
130
                   case 2
                       disp('Lagrange Interpolation');
131
132
                       x = input('Enter x values (e.g., [x1, x2, x3]): ');
                       y = input('Enter corresponding y values (e.g., [y1, y2, y3]): ');
targetX = input('Enter the target x value for interpolation: ');
133
134
                       lagrangeInterpolation(x, y, targetX);
135
136
                   case 3
                       disp('Newton Interpolation');
137
                       x = input('Enter x values (e.g., [x1, x2, x3]): ');
y = input('Enter corresponding y values (e.g., [y1, y2, y3]):
targetX = input('Enter the target x value for interpolation: ');
138
139
140
141
                        newtonInterpolation(x, y, targetX);
142
                   otherwise
143
                       disp('Geçersiz Seçim');
144
              end
145
146
148
         % çağırmak için -> numericalMethodsMenu();
```

Test 1. Menü Arayıcılığı ile Vandermonde Yöntemi Kullanımı

```
Command Window
     numericalMethodsMenu();
 Numerical Methods Menu:
  1. Nonlinear Equation Solution
 2. Nonlinear Equation System Solution
 3. Linear Equation System Solution
 4. Curve fitting
 5. Exit
  Enter your choice: (4)
  Curve Fitting Menu:
     Vandermonde Technique
  2. Lagrange Interpolation
 3. Newton Interpolation
  Enter your choice: 1
  Vandermonde Technique
  Enter x values (e.g., [x1, x2, x3]): [1, 2, 3]
Enter corresponding y values (e.g., [y1, y2, y3]): [4, 7, 10]
  Polynomial coefficients:
               3
```

Test 2. Menü Arayıcılığı ile Jacobi Yöntemi Kullanımı

