HACETTEPE UNIVERSITY
ENGINEERING FACULTY
ELECTRICAL AND ELECTRONICS ENGINEERING PROGRAM

2023-2024

SPRING SEMESTER

ELE708

NUMERICAL METHODS IN ELECTRICAL ENGINEERING

HW10

N23239410 – Ali Bölücü

1) Computer Problems

1.a) 10.1. Solve the two-point BVP

$u'' = 10u3+3u+t2,\ 0<t<1,$

$u(0)=0, u(1)=1,$

a)Shooting method.

b) Finite difference method.

c)Collocation method

---

a) In the Shooting Method, we first make a initial guess for the y2(0) value, and using the "ode45" we are going to try each y2(0) value until we found y2(0) when y1(1) = 1.

a. Initial parameters

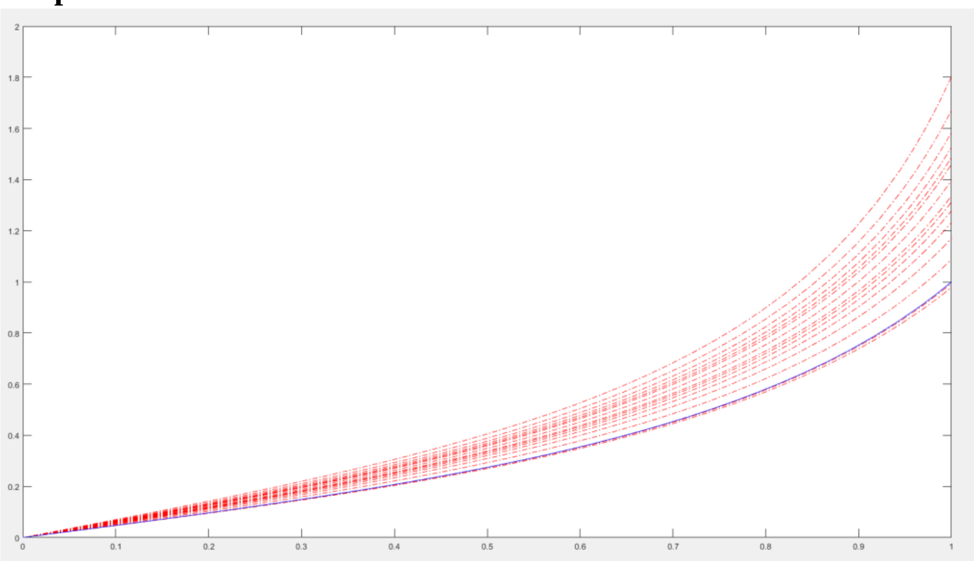
```
t0 = 0;
t1 = 1;
u0 = 0;
u1 = 1;
func = @(t, y) [y(2); 10*y(1)^3 + 3*y(1) + t.^2];
```

b. In order to find the y2(0) value when y1(1) = 1, we are using the "fzero" function.

```
% isZero degerinin 0 oludugu y2 degerini buluna kadar calisiyor
% y2(0)'ın initial degerini buluyor yani
% onu bulduktan sonra bvd'yi cozuyoruz
slope = fzero(@shooting_fun, 0.6);
    function [isZero] = shooting_fun(y2_initial)
        % Shooting method
        [t, y] = ode45(func,[t0,t1],[u0;y2_initial]);
        % Her denemeyi plot'luyor.
        if(u1 - y(end,1) == 0)
            plot(t, y(:,1), 'b-');
        else
            plot(t, y(:,1), 'r-.');
        end

        hold on;
        isZero = u1 - y(end,1);
    end
```

c. **Output**: Blue line is the answer.



b) In Finite Difference Method, we take advantage of that we have the second derivative of the function, using that we can find the correct curve.

a. Equation

$$\frac{y_2 - 2y_1 + y_0}{h^2} = 10\, y_1^3 + 3y_1 + t^2$$

b. Initial Parameters

```
t0 = 0;
t1 = 1;
u0 = 0;
u1 = 1;
func_initial = @(t, u) (10*u.^3+3*u+ t.^2);
func = @(t, y) [y(2); 10*y(1)^3 + 3*y(1) + t.^2];
```

c. When there is more than one step, in order to get the answer we need to solve system of non-linear equations. To create this in matlab we have the following code;

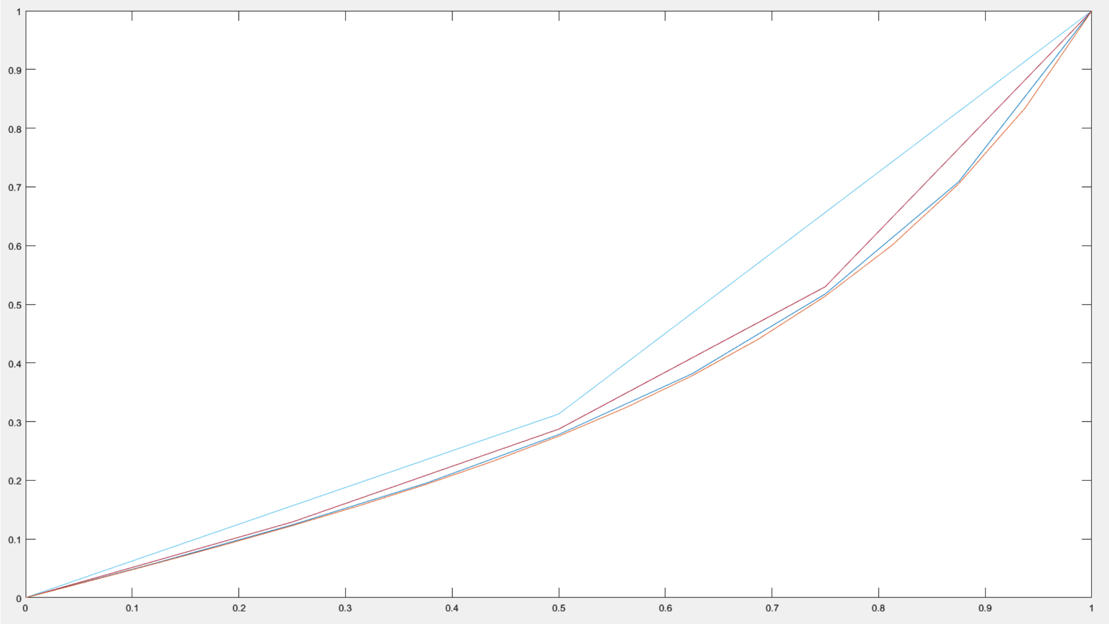
i. The middle_vals is the system of non-linear equations, we give this function to “fsolve” function to solve the system of equations.

```
function [middle_vals] = finite_difference(y)
    n_fd = length(y);
    h = 1/(n_fd+1); % step_size
    t_fd = linspace(h,1-h,n_fd);
    f = func_initial(t_fd,y);
    if n_fd > 1
        middle_vals = [(y(2)-2*y(1)+u0)/h^2-f(1); % ilk denklem
            ((y(3:n_fd)-2*y(2:n_fd-1)+y(1:n_fd-2))/h^2-f(2:n_fd-1))'; % aradaki denklemler
            (u1-2*y(n_fd)+y(n_fd-1))/h^2-f(n_fd)]; % son denklem
    else
        middle_vals = (u1-2*y+u0)/h^2-f;
    end
end
```

ii. The equation for n=3 reflected by the code is as follows, since we only now y(0) and y(1), this turn into system of equation.

$$\begin{bmatrix} \dfrac{y_{0,5} - 2y_{0,25} + y_0}{(0,25)^2} \\[2ex] \dfrac{y_{0,75} - 2.y_{0,5} + y_{0,25}}{(0,25)^2} \\[2ex] \dfrac{y_1 - 2.y_{0,75} + y_{0,5}}{(0,25)^2} \end{bmatrix} = \begin{bmatrix} f(0,25) \\ f(0,5) \\ f(0,75 \end{bmatrix}$$

d. **Output**, as we increased the step_size, the curve converges the correct one

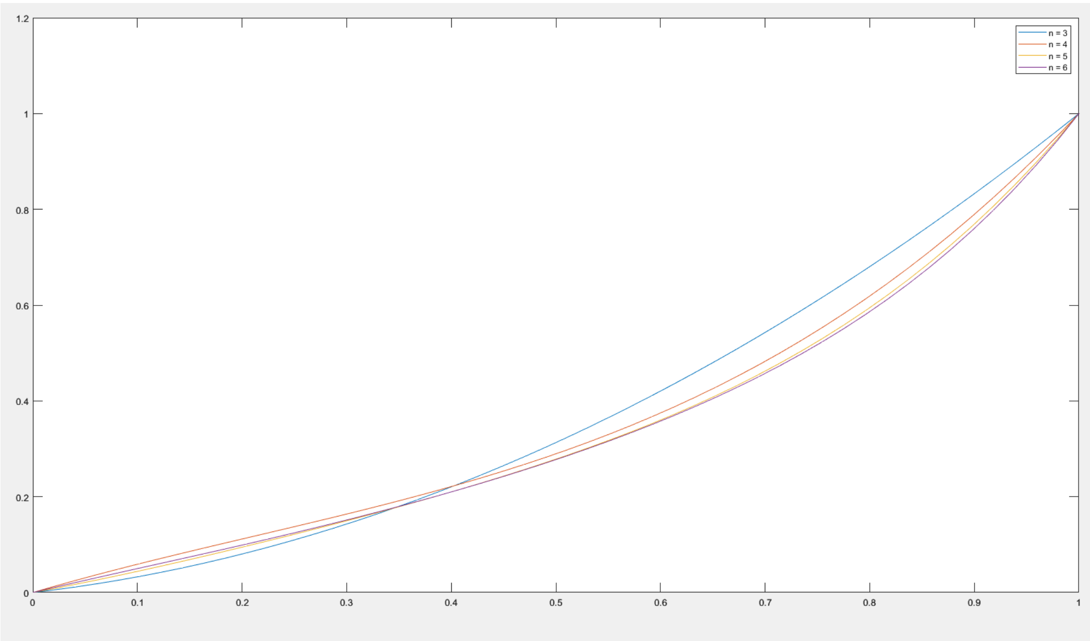


c) In Collocation Method, we try to find unknowns of the equation by fitting the values. The method we are using similar to previous one but this time we are going to use "polyval" function to find coefficient of equations.

a. With this code, we are creating coefficient vector for coefficient values that came from first and second order derivatives and then using these values we apply "polyfit" to find unknown values of the coefficients. As a result, we are able to construct the polynomial and plot the curve.

```
function [z] = collocation_method(x)
    x
    n = length(x);
    h = 1/(n-1);
    t = linspace(t0,t1,n)';
    d = (n-1:-1:1)'.*x(1:n-1);
    d2 = (n-2:-1:1)'.*(d(1:n-2));
    z = [polyval(x,t0)-u0;
        polyval(d2,t(2:n-1))-func_initial(t(2:n-1),polyval(x,t(2:n-1)));
        polyval(x,t1)-u1];
end
```

b. **Output**. As the we use more sub-interval for the finding coefficients, the curve converges the real one.



1.b) 10.7. The time-independent Schrodinger equation in one dimension

$$-\psi''(x) + V(x)\psi(x) = E\psi(x),$$

$$-\psi''(x) = E\psi(x), \qquad 0 < x < 1,$$

$$\psi(0) = 0, \qquad \psi(1) = 0.$$

How do your computed eigenvalues and eigenvectors compare with these analytical values as the mesh size of your discretization decreases? Try to characterize the error as a function of the mesh size.

When we apply the finite difference method to this question, what we get is

$$-\frac{\psi(x_{\{i+1\}}) - 2\psi(x_{\{i\}}) + \psi(x_{\{i-1\}})}{h^2} = E\,\psi(x_{\{i\}})$$

And after converting this to a matrix

$$\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -2 \end{pmatrix} \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \vdots \\ \psi_N \end{pmatrix} = E \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \vdots \\ \psi_N \end{pmatrix}$$

Now we can find the eigen values of the matrix.

After finding eigenvalues both with the matrix and the using (k^2 x pi^2), we get the results below. As we increased the mesh size, the eigenvalues converge the exact eigenvalues. The error decrease rate is at O(h^2). As the eigenvalues are increased, the error rate also increased.

```
------------------------------------------------------------
n:2
computed
     9    27

exact
    9.8696   39.4784

------------------------------------------------------------
n:3
computed
    9.3726   32.0000   54.6274

exact
    9.8696   39.4784   88.8264

------------------------------------------------------------
n:4
computed
    9.5492   34.5492   65.4508   90.4508

exact
    9.8696   39.4784   88.8264  157.9137

------------------------------------------------------------
n:5
computed
    9.6462   36.0000   72.0000  108.0000  134.3538

exact
    9.8696   39.4784   88.8264  157.9137  246.7401

------------------------------------------------------------
```