26/10/2024, 12:39 PS1.m

ProblemSetSolutions/PS1.m

```
1 % This is code for PS1 coding part
 2 % Author: Muhammad Bashir
 3 % Date: 24 October 2024
 4
 5
 6 % Parameters
 7 pars.C_T = 2;
 8 pars.r0 = 0.05;
 9 pars.alpha = 0.01;
10 pars.theta = 2
11 | pars.rho = 0.03;
12 pars.T = 10;
13
14 % iterate to find vector r =r0+alha*t
15 r = zeros(pars.T,1);
16 \mid r(1) = pars.r0;
   for t = 2:pars.T
17
18
        r(t) = pars.r0 + pars.alpha*t;
19
20
21
   % Q.1 Solve Consumption equation analytically using integrating factor method
   % C(t) = 1.4094 * exp(0.5 * (0.02t + 0.01 * t^2))
22
23
24 % Plot this solution function
25
26 | t = 1:pars.T;
27 | C = 1.4094 * exp(0.5 * (0.02 * t + .5 * 0.01 * t.^2));
28 figure
29
   plot(t,C)
30
   hold on
   yline(2, '--r', 'LineWidth', 1.5, 'Label', 'Target Consumption at Time 10'); % Add
31
   horizontal line at y=2 with label
   xlabel('Time (t)'); % Add x label
32
33
   ylabel('Consumption (C)'); % Add y label
34
   hold off
35
36
  % Save picture
37
   saveas(gcf, '/Users/muhammadbashir/GitHub/DynamicProgramming20↔
   24/ProblemSetSolutions/ConsumptionEvolutionAnalytical.png')
38
39
   % Q.2 Solving numerically using Finite Difference Method
40 % create 1000 grid points between 0 and 10
41 pars.n = 100;
42 t = linspace(0, pars.T, pars.n);
43 pars.step_size = pars.T/pars.n;
44 % initialize consumption vector
45 C = zeros(pars.n,1);
46 % find (r(t)-rho)/theta for all t
47 | discounter = (pars.r0 + pars.alpha*t - pars.rho)/pars.theta;
48 % initialize consumption at time T
49 C(pars.n) = 2;
50 % iterate backward to find consumption at all time periods
```

```
51 for i = pars.n-1:-1:1
52
       C(i) = C(i+1) - pars.step_size*(discounter(i)*C(i+1));
53
   end
54
55
   % Plot the solution along with the analytical solution
56
   scatter(t, C, 'o', 'LineWidth', .01) % Numerical solution as scatter plot with
57
   circles
  hold on
58
   plot(t, 1.4094*exp(0.5*(0.02*t+.5*0.01*t.^2)), '-r', 'LineWidth', 1) % Analytical
   solution as thick red line
60
   xlabel('Time (t)'); % Add x label
  ylabel('Consumption (C)'); % Add y label
61
   yline(2, '--r', 'LineWidth', 1.5, 'Label', 'Target Consumption at Time 10'); % Add
62
   horizontal line at y=2 with label
63
   legend('Numerical Solution', 'Analytical Solution', 'Location', 'northwest')
64
  hold off
65 % save
   saveas(qcf, '/Users/muhammadbashir/GitHub/DynamicProgramming20↔
66
   24/ProblemSetSolutions/ConsumptionEvolutionNumerical 100Steps.png')
67 % Store this C as C100
68 \mid C100 = C;
69 \mid t100 = t;
70 % Q.3 Repeat with 10 grid points
71 pars.n = 10;
72 t = linspace(0, pars.T, pars.n);
73 pars.step_size = pars.T/pars.n;
74 % initialize consumption vector
75 | C = zeros(pars.n,1);
76 \% find (r(t)-rho)/theta for all t
77
  discounter = (pars.r0 + pars.alpha*t - pars.rho)/pars.theta;
78 % initialize consumption at time T
79
   C(pars.n) = 2;
80 % iterate backward to find consumption at all time periods
   for i = pars.n-1:-1:1
81
82
       C(i) = C(i+1) - pars.step_size*(discounter(i)*C(i+1));
83
   end
84
85
   % Plot the solution along with the analytical solution
86
87
   scatter(t, C, 'o', 'LineWidth', .01) % Numerical solution as scatter plot with
   circles
   hold on
88
   scatter(t100, C100, 'x', 'LineWidth', .01) % Numerical solution as scatter plot with
89
90
   plot(t, 1.4094*exp(0.5*(0.02*t+.5*0.01*t.^2)), '-r', 'LineWidth', 1) % Analytical
   solution as thick red line
91 xlabel('Time (t)'); % Add x label
92
   ylabel('Consumption (C)'); % Add y label
   yline(2, '--r', 'LineWidth', 1.5, 'Label', 'Target Consumption at Time 10'); % Add
   horizontal line at y=2 with label
   legend('Numerical Solution (10 points)', 'Numerical Solution (100 points)',
    'Analytical Solution', 'Location', 'northwest')
95 hold off
96 | % save
```

26/10/2024, 12:39 PS1.m

97 saveas(gcf, '/Users/muhammadbashir/GitHub/DynamicProgramming20↔ 24/ProblemSetSolutions/ConsumptionEvolutionNumerical_10And100_Steps.png')

- 98 % Q.4 Discussion
- 99 % The numerical solution with 10 grid points is not as accurate as the one with 100 grid points. As we see once we increase no of grid points, we reach closer to the analytical solution. This is because the finite difference method is an approximation method and the accuracy of the solution depends on the number of grid points used. The more the grid points, the more accurate the solution will be.