**ProblemSetSolutions/PS1.m**

```matlab
% This is code for PS1 coding part
% Author: Muhammad Bashir
% Date: 24 October 2024


% Parameters
pars.C_T = 2;
pars.r0 = 0.05;
pars.alpha = 0.01;
pars.theta = 2
pars.rho = 0.03;
pars.T = 10;

% iterate to find vector r =r0+alha*t
r = zeros(pars.T,1);
r(1) = pars.r0;
for t = 2:pars.T
    r(t) = pars.r0 + pars.alpha*t;
end

% Q.1 Solve Consumption equation analytically using integrating factor method
% C(t) = 1.4094*exp(0.5*(0.02t+0.01*t^2))

% Plot this solution function

t = 1:pars.T;
C = 1.4094*exp(0.5*(0.02*t+.5*0.01*t.^2));
figure
plot(t,C)
hold on
yline(2, '--r', 'LineWidth', 1.5, 'Label', 'Target Consumption at Time 10'); % Add horizontal line at y=2 with label
xlabel('Time (t)'); % Add x label
ylabel('Consumption (C)'); % Add y label
hold off

% Save picture
saveas(gcf, '/Users/muhammadbashir/GitHub/DynamicProgramming2024/ProblemSetSolutions/ConsumptionEvolutionAnalytical.png')

% Q.2 Solving numerically using Finite Difference Method
% create 1000 grid points between 0 and 10
pars.n = 100;
t = linspace(0, pars.T, pars.n);
pars.step_size = pars.T/pars.n;
% initialize consumption vector
C = zeros(pars.n,1);
% find (r(t)-rho)/theta for all t
discounter = (pars.r0 + pars.alpha*t - pars.rho)/pars.theta;
% initialize consumption at time T
C(pars.n) = 2;
% iterate backward to find consumption at all time periods
```

```
for i = pars.n-1:-1:1
    C(i) = C(i+1) - pars.step_size*(discounter(i)*C(i+1));
end

% Plot the solution along with the analytical solution
figure
scatter(t, C, 'o', 'LineWidth', .01) % Numerical solution as scatter plot with circles
hold on
plot(t, 1.4094*exp(0.5*(0.02*t+.5*0.01*t.^2)), '-r', 'LineWidth', 1) % Analytical solution as thick red line
xlabel('Time (t)'); % Add x label
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', 'Analytical Solution','Location', 'northwest')
hold off
% save
saveas(gcf, '/Users/muhammadbashir/GitHub/DynamicProgramming2024/ProblemSetSolutions/ConsumptionEvolutionNumerical_100Steps.png')
% Store this C as C100
C100 = C;
t100 = t;
% Q.3 Repeat with 10 grid points
pars.n = 10;
t = linspace(0, pars.T, pars.n);
pars.step_size = pars.T/pars.n;
% initialize consumption vector
C = zeros(pars.n,1);
% find (r(t)-rho)/theta for all t
discounter = (pars.r0 + pars.alpha*t - pars.rho)/pars.theta;
% initialize consumption at time T
C(pars.n) = 2;
% iterate backward to find consumption at all time periods
for i = pars.n-1:-1:1
    C(i) = C(i+1) - pars.step_size*(discounter(i)*C(i+1));
end

% Plot the solution along with the analytical solution
figure
scatter(t, C, 'o', 'LineWidth', .01) % Numerical solution as scatter plot with circles
hold on
scatter(t100, C100, 'x', 'LineWidth', .01) % Numerical solution as scatter plot with circles
plot(t, 1.4094*exp(0.5*(0.02*t+.5*0.01*t.^2)), '-r', 'LineWidth', 1) % Analytical solution as thick red line
xlabel('Time (t)'); % Add x label
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')
hold off
% save
```

```
saveas(gcf, '/Users/muhammadbashir/GitHub/DynamicProgramming2024/ProblemSetSolutions/ConsumptionEvolutionNumerical_10And100_Steps.png')
% Q.4 Discussion
% 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.
```