econ 202A; problem set 2, coding; second half

1. At steady state,

i. $\frac{\dot{c}(t)}{c(t)} = 0 \iff f'(k(t)) - \rho - \theta g = 0$

$\iff A\alpha k^{\alpha-1} - \rho - \theta g = 0$

$\iff k^{\alpha-1} = \frac{\rho+\theta g}{\alpha A} \iff \underline{k_{ss} = \left(\frac{\rho+\theta g}{\alpha A}\right)^{\frac{1}{\alpha-1}}}$.

Substituting our parameters gives,

$\underline{k_{ss} \approx 17.2133}$.

ii. $\dot{k}(t) = 0 \iff A k_{ss}^{\alpha} - c_{ss} - (n+g)k_{ss} = 0$

$\iff \underline{c_{ss} = A k_{ss}^{\alpha} - (n+g)k_{ss}}$. Substituting our parameters gives,

$\underline{c_{ss} \approx 1.8935}$.

iii. At steady state,

$\underline{r_{ss} = f'(k_{ss}) \approx .05}$.

For the approximations, see code for question 1.

2. Using fsolve, we obtain the same results as 1, where the system of equations is given in 3.

3. The steady state values for consumption & capital satisfy the following system of equations,

$$F = \begin{bmatrix} f'(k_{ss}) - (\rho + \theta g) \\ c_{ss} - (f(k_{ss}) - (n+g) k_{ss}) \end{bmatrix} = \vec{0}.$$

The associated Jacobian is,

$$J = \begin{bmatrix} f''(k_{ss}) & 0 \\ -f'(k_{ss}) + (n+g) & 1 \end{bmatrix}.$$

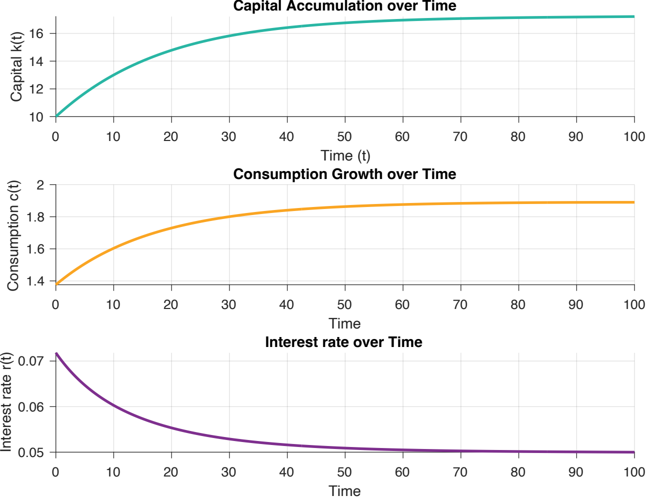
Using Newton's method, the results match 1 exactly.

4. Each approach gives approximately the same steady state values for each variable.

5. Using the finite difference method, we obtain

$$c(i+1) = c(i) + \Delta t \left[\frac{f'(k(i)) - \rho - \theta g}{\theta} c(i) \right]$$

$$k(i+1) = k(i) + \Delta t \left[f(k(i)) - c(i) - (n+g) k(i) \right].$$



Capital Accumulation over Time
Capital k(t)
10
12
14
16
0
10
20
30
40
50
60
70
80
90
100
Time (t)
Consumption Growth over Time
Consumption c(t)
1.4
1.6
1.8
2
Time
Interest rate over Time
Interest rate r(t)
0.05
0.06
0.07
Time

problem_set_2_code.m

```matlab
%{
------------------------------------------------------------------------------------------------------------------------
Purpose: Coding part of problem set 2
Created: Nico Rotundo 2024-11-04 (Adapted from Kiyea's section code)
------------------------------------------------------------------------------------------------------------------------
%}

%{
------------------------------------------------------------------------------------------------------------------------
Define parameters
------------------------------------------------------------------------------------------------------------------------
%}
clear all;

% Initial condition
k_0 = 10;

% Time discount factor
rho = 0.03;

% Inverse of the intertemporal elasticity of substitution
theta = 1.0;

% Technological growth rate
g = 0.02;

% Population growth rate
n = 0.02;

% Capital share
alpha = 1/3;

% TFP
A = 1.0;

% Tolerance for Newton's method
tol = 1e-6;

% Production function
f = @(k) A * k.^alpha;

% MPK
f_prime = @(k) alpha * A * k.^(alpha - 1);

tmin = 0;
tmax = 100;

% The number of time steps
I = 300;

%{
------------------------------------------------------------------------------------------------------------------------
Question 1: Solve analytically for the steady-state values of capital and consumption.
------------------------------------------------------------------------------------------------------------------------
%}

% Define equations for steady state capital, consumption, and interest rate
k_ss = ((rho + theta * g) / (alpha * A))^(1 / (alpha - 1));
c_ss = f(k_ss) - (n+g) * k_ss;
r_ss = f_prime(k_ss);

% Display the results
fprintf('Steady-state capital (k_ss): %.4f\n', k_ss);
fprintf('Steady-state consumption (c_ss): %.4f\n', c_ss);
fprintf('Steady-state interest rate (r_ss): %.4f\n', r_ss);

%{
------------------------------------------------------------------------------------------------------------------------
Question 2: Solve for the steady-state values of capital and consumption using fsolve.
------------------------------------------------------------------------------------------------------------------------
%}

% Define and solve the system using fsolve with an inline function
options = optimoptions('fsolve', 'Display', 'iter', 'TolFun', 1e-6);
solution = fsolve(@(x) [
    f_prime(x(1)) - (rho + theta * g);              % Capital steady-state condition
    x(2) - (f(x(1)) - (n + g) * x(1))               % Consumption steady-state condition
], [10, 1], options);

% Extract the steady-state values of capital and consumption
k_fsolve = solution(1);
c_fsolve = solution(2);
r_fsolve = f_prime(k_fsolve);

% Display the last values
fprintf('Steady-state capital (k_fsolve): %.4f\n', k_fsolve);
fprintf('Steady-state consumption (c_fsolve): %.4f\n', c_fsolve);
fprintf('Steady-state interest rate (r_fsolve): %.4f\n', r_fsolve);

%{
------------------------------------------------------------------------------------------------------------------------
Question 5: Use the shooting algorithm to simulate dynamic paths for capital k(t) and consumption c(t). Then, calculate
the implied rate of return on capital r(t) = f′(k) and plot the dynamic paths of these three variables over time.
------------------------------------------------------------------------------------------------------------------------
%}

% Initialize grid points
t = linspace(tmin, tmax, I)';
dt = (tmax-tmin)/(I-1);

% Objective function that calculates the difference between terminal capital k(T) and steady-state k_ss
diff = @(c_0) terminal_condition(c_0, k_0, k_ss, f, f_prime, rho, theta, g, n, dt, I);

% Guess an initial value of consumption
c_0_guess = 1;

% Use fsolve to find the initial consumption c_0 that makes k(T) = k_ss
options = optimoptions('fsolve', 'TolFun', tol, 'Display', 'iter');
c_0 = fsolve(diff, c_0_guess, options);

[k, c, r] = forward_simulate(c_0, k_0, f, f_prime, rho, theta, g, n, dt, I);

% Extract the last values of k and c
k_final = k(end);
c_final = c(end);
r_final = f_prime(k_final);

% Display the last values
fprintf('Steady-state capital (k_final): %.4f\n', k_final);
fprintf('Steady-state consumption (c_final): %.4f\n', c_final);
fprintf('Steady-state interest rate (r_final): %.4f\n', r_final);

% 5-1. Evolution of capital, consumption, and interest rate
figure;
    subplot(3,1,1);
    plot(t, k, 'r-', 'LineWidth', 2, 'Color', [41/255, 182/255, 164/255]);
    xlabel('Time (t)');
    set(gca, 'TickDir', 'out');
    box off;
    ylabel('Capital k(t)');
    title('Capital Accumulation over Time');
    grid on;

    subplot(3,1,2);
    plot(t, c, 'b-', 'LineWidth', 2, 'Color', [250/255, 165/255, 35/255]);
    xlabel('Time');
    set(gca, 'TickDir', 'out');
    box off;
    ylabel('Consumption c(t)');
    title('Consumption Growth over Time');
    grid on;

    subplot(3,1,3);
    plot(t, r, 'b-', 'LineWidth', 2, 'Color', [0.4940 0.1840 0.5560]);
    xlabel('Time');
    set(gca, 'TickDir', 'out');
    box off;
    ylabel('Interest rate r(t)');
    title('Interest rate over Time');
    grid on;

% Export
exportgraphics(gcf, '/Users/nicorotundo/Documents/GitHub/DynamicProgramming2024/problem_sets/problem_set_2/output/question_5.pdf', 'ContentType', 'vector');


%{
------------------------------------------------------------------------------------------------------------------------
Question 3: Solve numerically for the steady-state values of capital and consumption by implementing Newton’s Method
without using fsolve
------------------------------------------------------------------------------------------------------------------------
%}

% Second derivative
f_double_prime = @(k) alpha * (alpha - 1) * A * k.^(alpha - 2);

% Initial guess vector
x = [k_0; c_0];

% Maximum iterations
max_iter = 100;

for iter = 1:max_iter
    % Step 1: Calculate F(x)
    F = [
        f_prime(x(1)) - (rho + theta * g);    % Capital steady-state condition
        x(2) - (f(x(1)) - (n + g) * x(1))     % Consumption steady-state condition
    ];

    % Check for convergence
    if norm(F, inf) < tol
        fprintf('Converged in %d iterations.\n', iter);
        break;
    end

    % Step 2: Calculate the Jacobian matrix J(x)
    J = [
        f_double_prime(x(1)), 0;
        -(f_prime(x(1)) - (n + g)), 1
    ];

    % Step 3: Update x using Newton's method: x_new = x - J(x)^{-1} * F(x)
    x_new = x - J \ F;  % Equivalent to solving J * delta_x = -F for delta_x and updating x

    % Check for convergence based on the update size
    if norm(x - x_new, inf) < tol
        fprintf('Converged based on update size in %d iterations.\n', iter);
        break;
    end

    % Update x for the next iteration
    x = x_new;
end

% Extract steady-state values of capital, consumption, and interest rate
k_newton = x(1);
c_newton = x(2);
r_newton = f_prime(k_newton);

% Display the results
fprintf('Optimal initial consumption (c_0_newton): %.4f\n', c_0);
fprintf('Final capital; Newton (k(T)): %.4f\n', k_newton);
fprintf('Final consumption; Newton (c(T)): %.4f\n', c_newton);
fprintf('Final interest rate; Newton (r(T)): %.4f\n', r_newton);

%{
------------------------------------------------------------------------------------------------------------------------
Define forward simulate function
------------------------------------------------------------------------------------------------------------------------
%}

% This function solves the two differential equations using forward simulation.
function [k, c, r] = forward_simulate(c_0, k_0, f, f_prime, rho, theta, g, n, dt, I)

    % Pre-allocate arrays for solution
    k = zeros(I, 1);
    c = zeros(I, 1);
    r = zeros(I, 1);
    k(1) = k_0;
    c(1) = c_0;
    r(1) = f_prime(k(1));

    for i = 1:I-1

        % Euler equation for consumption growth: (c(i+1)-c(i))/dt = c(i)*(f'(k(i)-rho-theta*g)/theta
        c(i+1) = c(i) + dt * (f_prime(k(i)) - rho - theta * g) / theta * c(i);

        % Capital accumulation equation:(k(i+1)-k(i))/dt = f(k(i))-c(i)-(n+g)k(i)
        k(i+1) = k(i) + dt * (f(k(i)) - c(i) - (n + g) * k(i));

        % Interest rate
        r(i+1) = f_prime(k(i+1));
    end

end


%{
------------------------------------------------------------------------------------------------------------------------
Define terminal condition function
------------------------------------------------------------------------------------------------------------------------
%}

% This function calculates the difference between terminal capital k(T) and steady-state k_ss
function difference = terminal_condition(c_0, k_0, k_ss, f, f_prime, rho, theta, g, n, dt, I)

    [k, ~, ~] = forward_simulate(c_0, k_0, f, f_prime, rho, theta, g, n, dt, I);
    k_T = k(end);  % Terminal capital k(T)

    % The difference between terminal k(T) and steady-state k_ss
    difference = k_T - k_ss;
end
```