$$\frac{\dot{c}}{c_t} = \frac{f'(k) - \rho - \theta g}{\theta}$$

$$\dot{k}_t = f(k) - c - (n+g)k$$

In steady state both k, c grow at rate 0.

$$Ak^{\alpha} - c + (n+g)k = 0$$

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

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

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

$$c_{ss} = A k_{ss}^{\alpha} + (n+g) k_{ss}$$

$$c_{ss} = A\left(\frac{\rho + \theta g}{\alpha A}\right)^{\frac{\alpha}{\alpha-1}} + (n+s)\left(\frac{\rho + \theta g}{\alpha A}\right)^{\frac{1}{\alpha-1}}$$

Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MATLAB Script: ramsey.m
%
% Author: Muhammad Bashir (Follows closely Kiyea Jin's version)
% date: November 2nd 2024
%
% description:
% this script solves the ramsey-cass-koopmans model
%
% the model consists of two equations:
%   (dc/dt)/c = (f'(k) - rho - theta*g)/theta
%   dk/dt = f(k) - c - (n+g)k
% where an initial condition for capital k0 is provided,
% and intertemporal budget constraint with equality is imposed as a terminal condition.
%
% parameters:
% - Discount rate (rho): 0.03
% - Inverse of IES (theta): 1
% - Technology growth rate (g): 0.02
% - Population growth rate (n): 0.02
% - Capital share (alpha): 1/3
% - TFP (A): 1
% - Initial boundary condition: K0 = 10
%
% Code Structure:
% 1. DEFINE PARAMETERS
% 2. INITIALIZE GRID POINTS
% 3. STEADY STATES
% 4. SHOOTING ALGORITHM
% 5. PLOT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;
```

1. DEFINE PARAMETERS

```
p = define_parameters();

%%%%%%%%%%%%%%%%%%%%%%% Q 2 fsolve to get SS k and c %%%%%%%%%%%%%%%%%%%%%%%%%
% We need to essentialy find k and c that satisfy the following two equations:
% 1. f(k) - c - (n+g)k = 0
% 2. f'(k) - rho - theta*g = 0
% We use functions f and f_prime defined in define_parameters.m to calculate f(k) and f'(k)

% Define the objective function that calculates the difference between the two equations
% Note: kss and css are functions of k and c
diff = @(x) [p.f(x(1)) - x(2) - (p.n + p.g) * x(1); p.f_prime(x(1)) - p.rho - p.theta * p.g];  % x(1) = k, x(2) = c

% Initial guess for k and c
x0 = [p.k0; p.k0];  % Initial guess for k and c

% Solve the system of equations using fsolve
options = optimoptions('fsolve', 'Display', 'off');
x = fsolve(diff, x0, options);

% Steady state k and c and interest rate
kss = x(1);
css = x(2);
disp(['Steady state k: ', num2str(kss)]);
disp(['Stedy state c: ', num2str(css)]);
disp(['Newton Method Interest rate(%): ', num2str(100*p.f_prime(kss))]);

% Also print analytical sollution values for k and c
kss_analytical = ((p.rho + p.theta * p.g) / (p.alpha * p.A))^(1 / (p.alpha - 1));
css_analytical = p.f(kss_analytical) - (p.n + p.g) * kss_analytical;
disp(['Analytical Steady state k: ', num2str(kss_analytical)]);
disp(['Analytical Stedy state c: ', num2str(css_analytical)]);
disp(['Newton Method Interest rate(%): ', num2str(100*p.f_prime(kss_analytical))]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Q 3 Solve Again using Newton's Method %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x0 = [p.k0; p.k0];  % Initial guess for k and c
error = inf;
iter = 0;  % Initialize iteration counter
x = x0;  % Initialize x
while iter < p.maxit
```

```
    % Calculate the Jacobian matrix
    J = [p.f_prime(x(1)) - (p.n + p.g), -1; -1, 1 / p.theta];

    % Calculate the difference
    error = diff(x);

    % Update x
    x = x - J \ error;

    % Check convergence
    if norm(error) < p.tol
        break;
    end

    iter = iter + 1;  % Increment iteration counter
end

% display results kss, css including number of iterations and error at convergence
disp(['Newton Method Steady state k: ', num2str(x(1))]);
disp(['Newton Method Stedy state c: ', num2str(x(2))]);
disp(['Newton Method Interest rate(%): ', num2str(100*p.f_prime(x(1)))]);
disp(['Number of iterations: ', num2str(iter)]);
disp(['Error at convergence: ', num2str(norm(error))]);

%%%%%%%%%%%%%%%%%%%%%%% Q.5 Discuss the results %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The results are pretty close but Newton's method is very senstive to number of iterations. When we increase iterations to 10000, we get the same

%%%%%%%%%%%%%%%%%%%%%%% Q.6 Shooting Algorithm %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% We will use shooting algorithm to solve the model. We will use the following boundary conditions:
% 1. k0 = 10
% 2. kT = kss
% 3. cT = css
```

```
Steady state k: 17.2133
Stedy state c: 1.8935
Newton Method Interest rate(%): 5
Analytical Steady state k: 17.2133
Analytical Stedy state c: 1.8935
Newton Method Interest rate(%): 5
Newton Method Steady state k: 17.2127
Newton Method Stedy state c: 1.8935
Newton Method Interest rate(%): 5.0001
Number of iterations: 4854
Error at convergence: 9.9953e-07
```

2. INITIALIZE GRID POINTS

```
t = linspace(p.tmin, p.tmax, p.I)';
dt = (p.tmax-p.tmin)/(p.I-1);
```

4. SHOOTING ALGORITHM

```
        % 4-1. Find the initial value of consumption that satisfies terminal boundary condition

        tic;
        % Objective function that calculates the difference between terminal capital k(T) and steady-state kss
        % Note: k(T)-kss is a function of c0
        diff = @(c0) terminal_condition(c0, p.k0, kss, p.f, p.f_prime, p.rho, p.theta, p.g, p.n, dt, p.I);

        % Guess an initial value of consumption
        c0_guess = 1;

        % Use fsolve to find the initial consumption c0 that makes k(T) = k_ss
        % Note: X = fsolve(FUN, X0, options) starts at the matrix X0 and tries to solve the equations in FUN.
        % Set OPTIONS = optimoptions('fsolve','Algorithm','trust-region'), and then pass OPTIONS to fsolve.
        options = optimoptions('fsolve', 'TolFun', p.tol, 'Display', 'iter');
        c0 = fsolve(diff, c0_guess, options);

% 4-2. Forward simulate with the updated initial consumption

[k, c] = forward_simulate(c0, p.k0, p.f, p.f_prime, p.rho, p.theta, p.g, p.n, dt, p.I);
toc;
```

Implied interest rate

```
r_t = p.f_prime(k);
```

```
                                         Norm of      First-order   Trust-region
 Iteration  Func-count     ||f(x)||^2          step       optimality         radius
     0          2             6549.24                      2.67e+03              1
     1          3             6549.24             1        2.67e+03              1
     2          5             4481.43          0.25        7.09e+03           0.25
     3          6             4481.43         0.625        7.09e+03          0.625
     4          7             4481.43       0.15625        7.09e+03          0.156
     5          9             3832.38     0.0390625        9.82e+03         0.0391
     6         10             3832.38     0.0976562        9.82e+03         0.0977
     7         12              3282.9     0.0244141         1.3e+04         0.0244
```

```
     8          14              102.78       0.0610352          4.91e+04          0.061
     9          16              35.0699       0.00209295          8.1e+04          0.153
    10          18              0.935949      0.000433197          9.4e+03          0.153
    11          20          0.000872267       9.95919e-05              270          0.153
    12          22           8.1712e-10       3.23336e-06            0.261          0.153
    13          24           3.88012e-20        3.1355e-09          1.8e-06          0.153
    14          26           2.26637e-25       2.16067e-14          4.34e-09          0.153

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.

Elapsed time is 0.049052 seconds.
```

5. PLOT

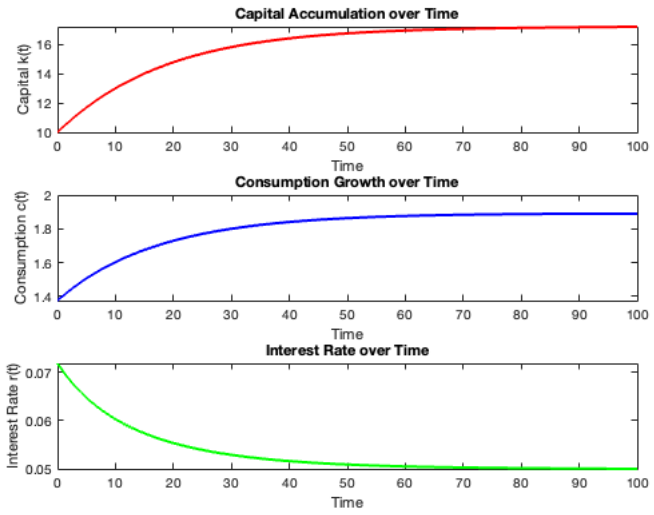
```
% 5-1. Evolution of capital and consumption

figure;
subplot(3,1,1);
plot(t, k, 'r-', 'LineWidth', 2);
xlabel('Time'); ylabel('Capital k(t)');
title('Capital Accumulation over Time');

subplot(3,1,2);
plot(t, c, 'b-', 'LineWidth', 2);
xlabel('Time'); ylabel('Consumption c(t)');
title('Consumption Growth over Time');

subplot(3,1,3);
plot(t, r_t, 'g-', 'LineWidth', 2);
xlabel('Time'); ylabel('Interest Rate r(t)');
title('Interest Rate over Time');
```

DISCUSSION %%% % We see achieving convergence around 60th time period and we also see that interest rate is decreasing over time. This is because the economy is converging to steady state and capital is increasing which leads to lower returns due to diminishing returns to capital.



Published with MATLAB® R2024b