**Simulating the Ramsey-Cass-Koopmans Model Using MATLAB and Simulink**

By Sonia Bridge and Ken Deeley, MathWorks

Many economic and financial models, such as those for resource allocation or optimal growth, involve systems of differential equations with no explicit analytical solution. Solving these systems numerically and visualizing results are important tasks for economists and other financial analysts.

Using the fundamental Ramsey-Cass-Koopmans (RCK) model as an example, this article describes two workflows for creating, simulating, and visualizing a system of ordinary differential equations (ODEs). One approach is based on MATLAB, the other on Simulink. The MATLAB approach uses programming techniques familiar to financial professionals who work in a technical computing environment. The Simulink approach uses a visual modeling environment and graphical representation of the system.

Simulink is a block diagram environment used for modeling time-varying systems with feedback. Such systems are typical in control engineering applications, which have for many years influenced economic modeling [1]. Financial models can involve large-scale systems of ODEs with many equations and dependencies. Simulink provides convenient features such as subsystems and model referencing to support the modeling of large systems.

The code and models used in this article are available for download.

**The Ramsey-Cass-Koopmans Model**

The RCK model aims to explain long-term economic growth in terms of capital accumulation and consumption growth [2-4]. The core RCK model is two-dimensional, comprising two coupled ODEs for per-capita wealth () and per-capita consumption (). Figure 1 shows the phase portrait of the system.

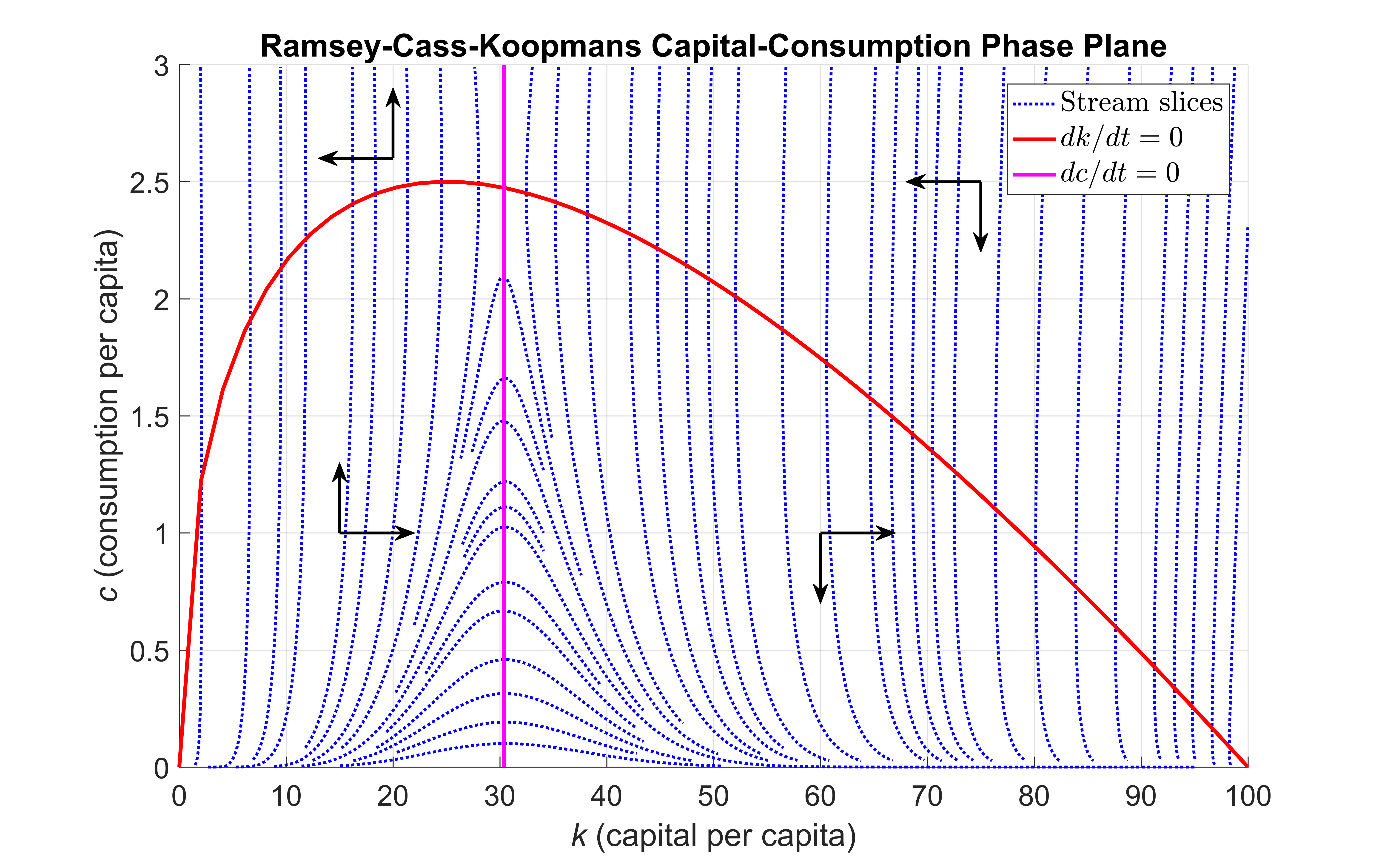


Figure 1: Phase portrait of the Ramsey-Cass-Koopmans system of ordinary differential equations.

The core RCK model equations for per-capita wealth () and per-capita consumption () are:

Since and appear in both equations, the two ODEs are coupled. The terms in these equations are as follows:

* is a production function measuring the relative economic output in terms of and a capital elasticity parameter (the responsiveness of the output production to changes in the input capital).
* is the growth rate of labor productivity (for example, due to technological innovation or efficiency improvements).
* is the growth rate of labor supply (for example, due to migration or population increase).
* is the depreciation rate of capital (for example, due to inflation).
* is the derivative of the production function
* is an elasticity parameter indicating the tendency of consumers to smooth out their consumption over time.
* is the rate at which consumers discount their future consumption (for example, by indicating a preference for immediate consumption or attempting to preserve their long-term average consumption).

**Creating the RCK Model Using MATLAB**

We can solve many systems of ODEs directly using the MATLAB function ode45, provided we express the system in the standard form on the time interval and subject to the initial condition Note that and are vector-valued if there are multiple unknown functions of time in the system of equations.

We begin by defining the model parameters in a structure variable params, and then write a vector-valued function RCK\_Equations representing the right-hand side of the standard differential equation This function returns a two-element vector containing the values of and at each time step.

We also write two auxiliary functions RCK\_f and RCK\_df returning the values of the production function and its derivative respectively. We save each function in a separate file so that it will be easy to investigate the effect of different production functions on the numeric results.

function dY\_dt = RCK\_Equations(t, Y, params)

%RCK\_EQUATIONS Function defining the right-hand sides of the two %coupled ordinary differential equations defining the Ramsey-Cass-%Koopmans model.

% Extract k and c.

k = Y(1);

c = Y(2);

% Write down the equations for dk/dt and dc/dt.

dY\_dt(1, 1) = RCK\_f(k, params) - c - ...

(params.phi + params.xi + params.delta) \* k; % dk/dt

dY\_dt(2, 1) = ( ( RCK\_df(k, params) - params.theta - params.xi - ...

params.delta ) / params.rho - params.phi ) \* c; % dc/dt

end % RCK\_Equations

The next step is to create a function handle (@) containing the input function for ode45 by parameterizing RCK\_Equations with the predefined parameters structure params. This function has to be a function of time (t) and state (Y) only.

RCK\_Fun = @(t, Y) RCK\_Equations(t, Y, params);

We ensure that both per-capita wealth and consumption remain nonnegative over time by modifying the solver options with odeset.

opts = odeset('NonNegative', [1, 2]);

Taking the initial conditions to be and and creating a time vector from 0 to 1.5 units, we can now solve the system numerically using ode45. The outputs of ode45 are time and state. Note that since we are creating a time vector t for ode45 directly, it is only necessary to return the second output argument Y from ode45.

Y0 = [25; 2];

t = linspace(0, 1.5, 5000);

[~, Y] = ode45(RCK\_Fun, t, Y0, opts);

k\_out = Y(:, 1); % Output per-capita wealth

c\_out = Y(:, 2); % Output per-capita consumption

We use the MATLAB visualization function comet to create an animated trajectory of the solution path, showing the time evolution of capital and consumption. The final frame of this animation, superimposed on the phase plane, is shown in Figure 2. The red line is a small portion of the curve shown in Figure 1.

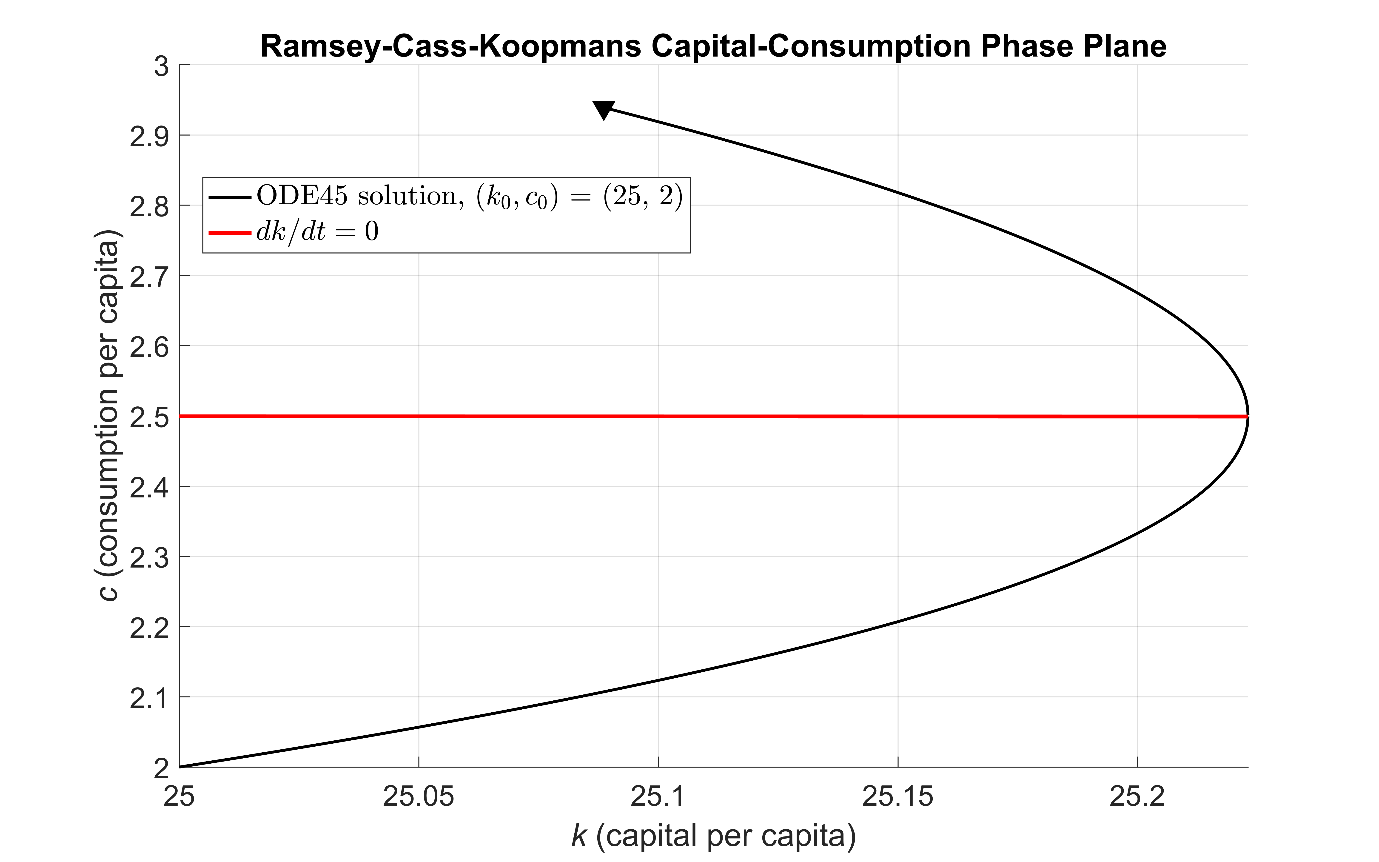


Figure 2. Solution trajectory starting from the point obtained by solving the coupled system of equations directly using ode45.

**Finding the Steady States and Solving the System Using Time Elimination**

We can find the steady state(s) of the system by solving the equations . Solving yields the curve (the red curve in Figure 1). Solving yields a single value for namely

This value defines the vertical line in Figure 1. We can use the meshgrid function to create lattices K and C of capital/consumption grid points. After computing the differentials dK and dC as defined by the RCK equations, we can then use the streamslice visualization function to render the streamlines in the plane.

streamslice(K, C, dK, dC)

Overlaying the curves and creates the phase portrait shown in Figure 1.

As Carrol shows [3], there is no analytical solution for the model’s transition to its steady state. However, we can use the time elimination technique to obtain the following:

Integrating with respect to gives a solution trajectory for as a function of . To avoid problems evaluating when its numerator or denominator are zero, we split the -domain into two parts, one to the left of and one to the right [2]. Applying ode45 gives the solution trajectory (Figure 3).

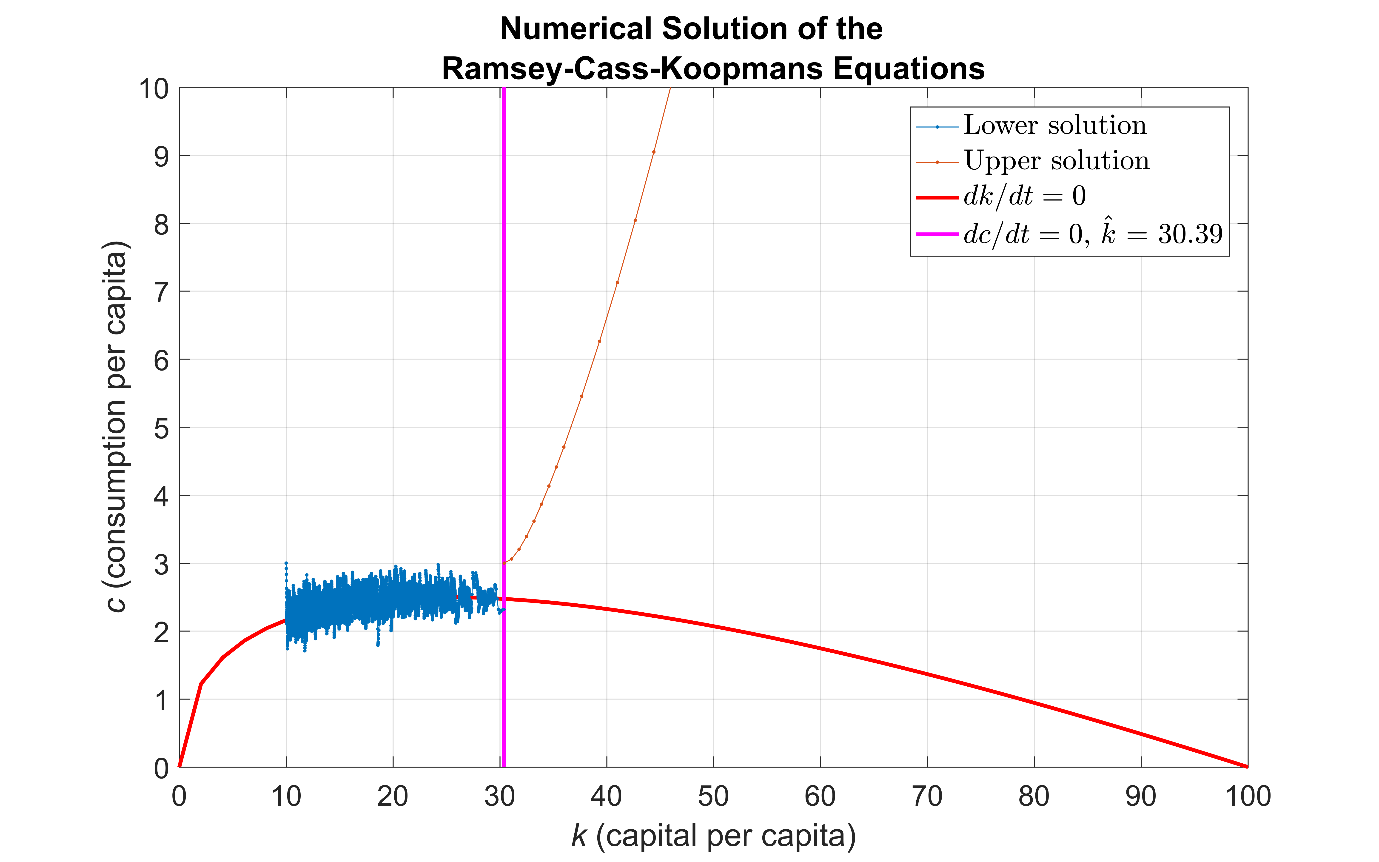


Figure 3. Upper and lower solution paths for a consumption strategy obtained using the time-elimination method.

Note that the upper solution path is smooth, whereas the lower solution path suffers from numerical instabilities in the vicinity of , a characteristic of certain stiff systems. In our example, instead of using ode45, we will use a solver designed to handle stiff systems, such as ode15s. To improve the reliability of the solution trajectory, we compute the Jacobian of the system and pass it to the solver via odeset. The resulting smooth path is shown in Figure 4. For larger or more complex systems, we could use Symbolic Math ToolboxTM to compute analytic Jacobians without manual calculation.

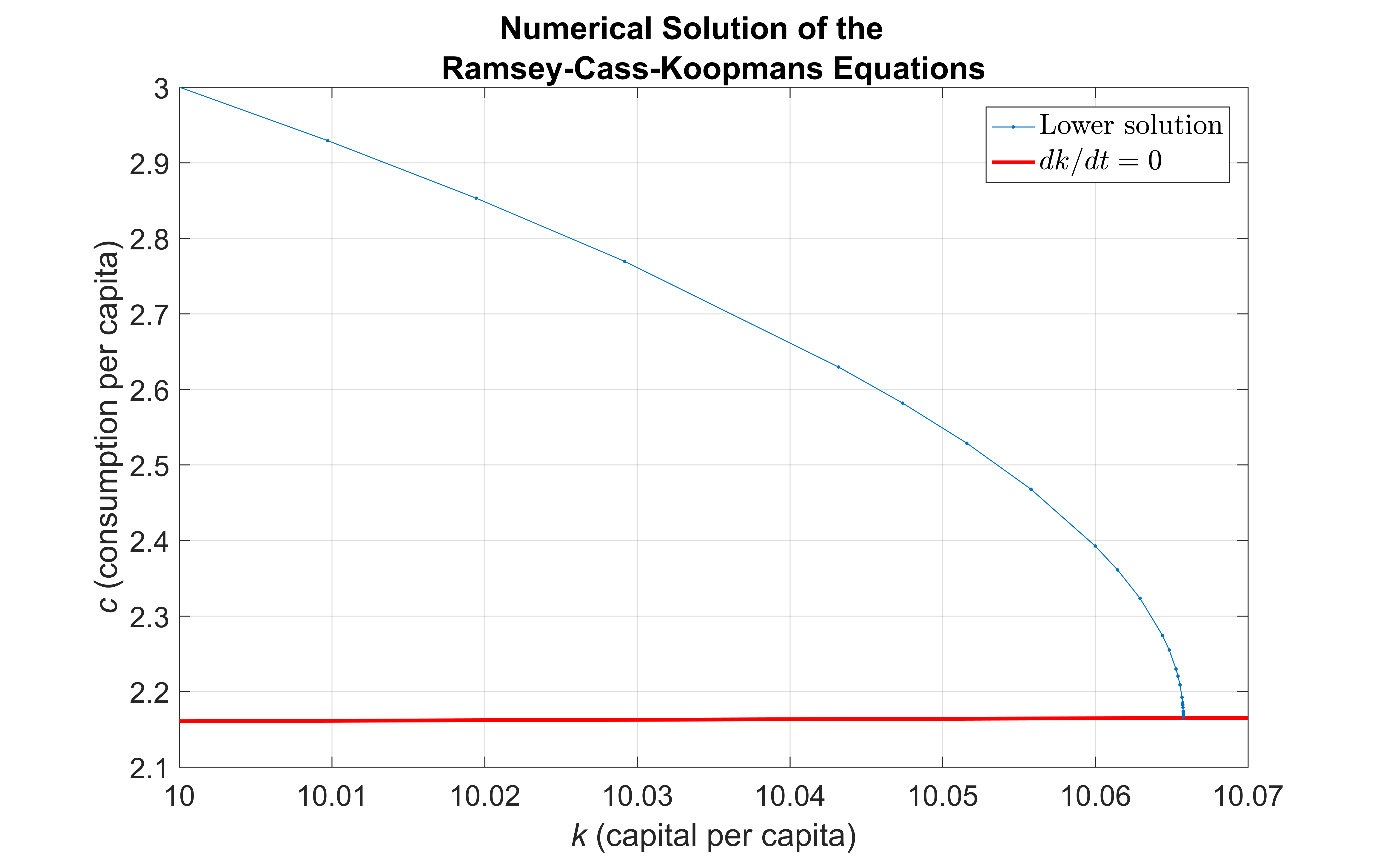


Figure 4. Lower solution path obtained using the stiff solver ode15s.

**Creating the RCK Model Using Simulink**

Using MATLAB, we solved the RCK system using techniques familiar to financial analysts. We’ll now take a less familiar but more visual approach: we’ll implement the system dynamically using libraries of predefined blocks in Simulink.

In Simulink, data is represented as signals and as parameters. Signals are the lines connecting blocks, and represent time-varying data, such as the derivative . Parameters are system values stored inside blocks--for example, the initial condition .

When modeling ODEs in Simulink, we begin with the Integrator block from the Continuous library. This block numerically integrates its input signal (the derivative). Since the system has first-order derivatives for *k* and *c*, we use two integrator blocks. The initial conditions and are assigned as parameters inside the blocks (Figure 5).

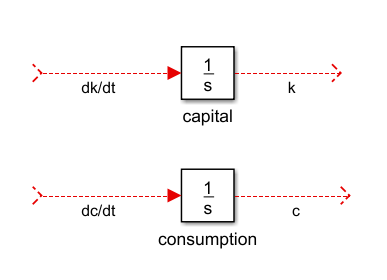


Figure 5. Integrator blocks for k and c. The red lines indicate signals not yet connected to other blocks.

We implement the right-hand sides of the RCK equations using blocks from the Simulink library (Table 1).

|  |  |  |
| --- | --- | --- |
| **Block** | **Symbol** | **Purpose** |
| Constant |  | Reference model parameters (e.g., params.phi) |
| Product |  | Multiply signals |
| Sum |  | Add or subtract signals |
| Gain |  | Multiply or divide a signal by a constant |
| Math Function |  | Mathematical operations (e.g., powers and logarithms) |
| Outport |  | Pass results to the MATLAB workspace (e.g., and ) |
| Derivative |  | Numerically approximate derivatives |
| Subsystem |  | Group functionally-related blocks |

Table 1. Simulink blocks used to implement the right-hand sides of the RCK equations.

As our model increases in size and complexity, we can simplify it by grouping blocks into subsystems using the Subsystem block. We encapsulate the production function in a subsystem (Figure 6).

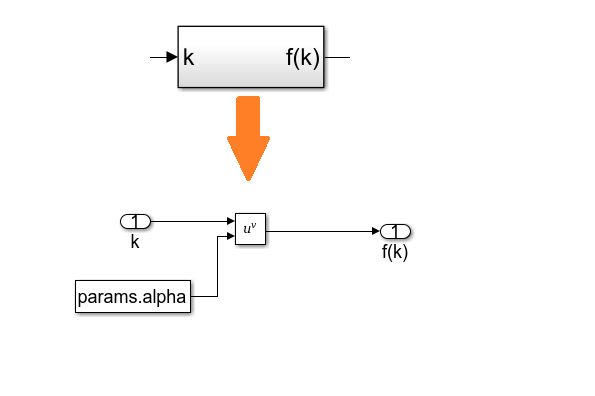


Figure 6. The production function implemented as a subsystem.

We approximate the derivative using the Derivative block. Note that the Derivative block has no initial condition parameter and so should not be used as a starting point for modeling ODEs.

Figure 7 shows the complete RCK model.

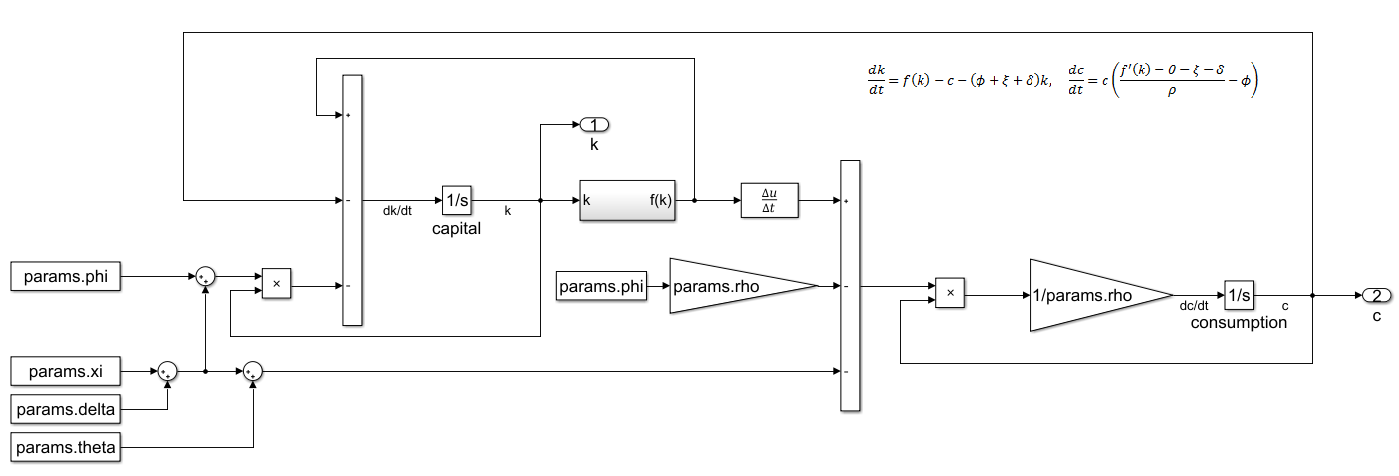


Figure 7. The complete Simulink RCK model.

After constructing the model, we specify the simulation stop time as 500 time units and start the simulation. Figure 8 shows the resulting trajectory starting from the point

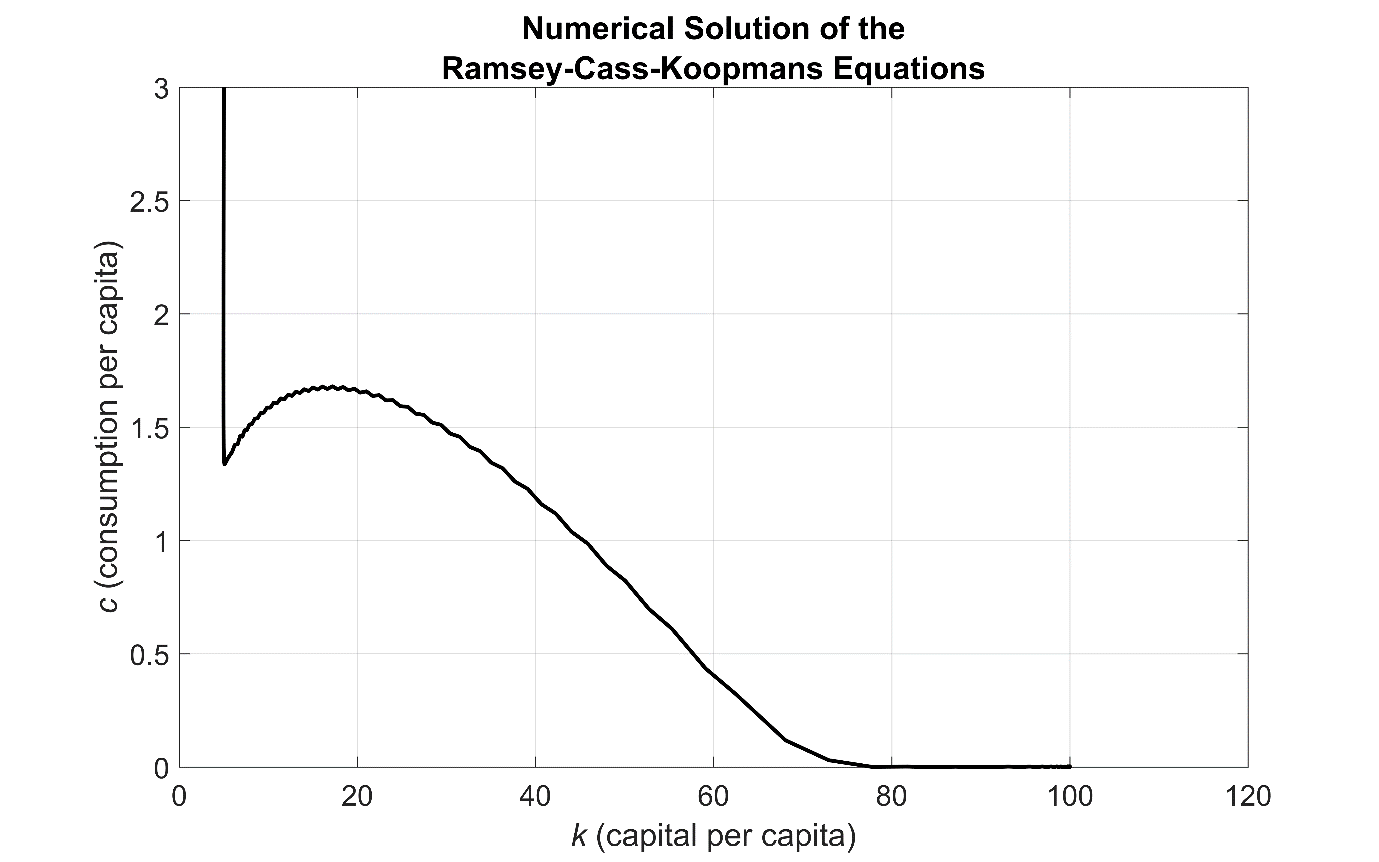


Figure 8. Solution trajectory starting from the point .

**Running Simulations in Parallel**

As part of the model analysis, we may want to investigate the dependency of the model on its parameters by running simulations using different sets of parameter values. Each simulation can be run independently and implemented in parallel using the parfor construct from Parallel Computing ToolboxTM.

Starting with the MATLAB based model implementation, we create lattices of grid points K0 and C0 representing the different initial conditions we want to investigate. Within each iteration of the parfor loop, we select a different initial condition Y0 and store the outputs k\_out and c\_out using cell arrays.

RCK\_Fun = @(t, Y) RCK\_Equations(t, Y, params);

opts = odeset('NonNegative', [1, 2]);

t = linspace(0, 1.5, 1000);

parfor k = 1:numel(K0)

% Initial values for per-capita wealth and consumption.

Y0 = [K0(k); C0(k)];

% Solve the coupled system.

[~, Y] = ode45(RCK\_Fun, t, Y0, opts);

k\_out{k} = Y(:, 1); % Output per-capita wealth

c\_out{k} = Y(:, 2); % Output per-capita consumption

end % parfor

Using 100-by-100 lattices of initial conditions means that we perform 10,000 parallel simulations of the model. These simulations produce the solution trajectories shown in Figure 9.

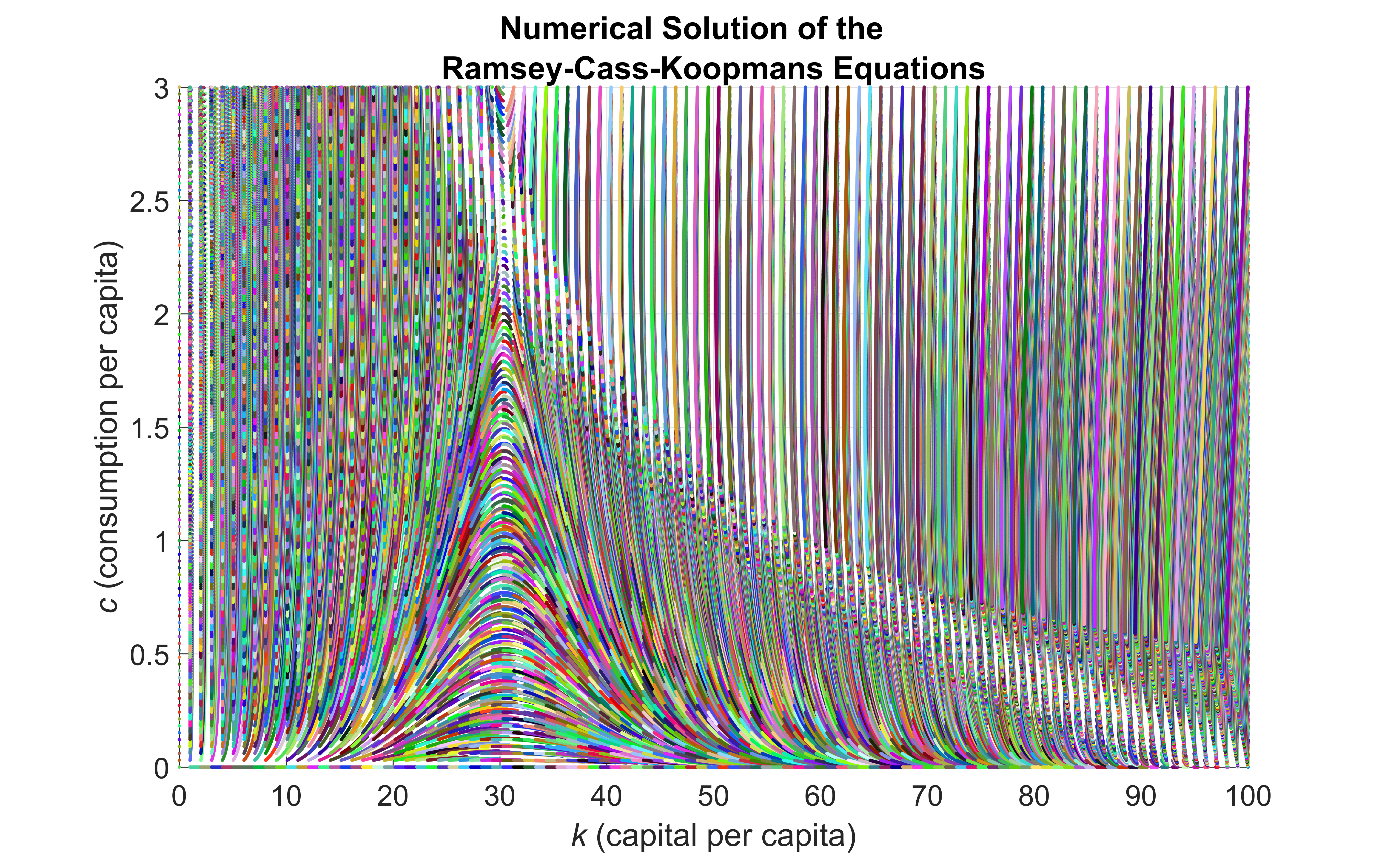


Figure 9. Solution paths of the RCK model starting from different initial conditions.

To simulate the Simulink RCK model in parallel we do the following:

1. Load the model on each worker using load\_system and the spmd construct.
2. Define arrays K0 and C0 for the initial conditions under which we want to simulate the model.
3. Write a function to simulate the model programmatically using the sim command. (Note that to set parameters programmatically in the model using set\_param, we need to convert numerical values to text. The try/catch construct safeguards against unexpected convergence issues for isolated sets of initial conditions.)
4. Within each iteration of the parfor loop, call the function with a different pair of initial conditions.

%% Load the model once per worker.

spmd

load\_system('RCK\_Model');

end % spmd

%% Perform the simulations in parallel.

parfor k = 1:numel(K0)

simout(k) = runSim(K0(k), C0(k));

end % parfor

function simout = runSim(k0, c0)

% RUNSIM Function simulating the model for different initial values

% for per-capita wealth and consumption using the stiff system solver

% ode15s and a stop time of 45 time units.

% Format the initial values for per-capita wealth and consumption as text.

k0 = num2str(k0);

c0 = num2str(c0);

set\_param('RCK\_Model/capital', 'InitialCondition', k0)

set\_param('RCK\_Model/consumption', 'InitialCondition', c0)

% Run the simulation.

try

simout = sim('RCK\_Model', 'Solver', 'ode15s', 'StopTime', '45');

catch

% If a simulation run fails to converge, assign an empty output.

simout = Simulink.SimulationOutput;

end % try

end % runSim

Figure 10 shows the solution trajectories for 10,000 parallel simulations of the model.

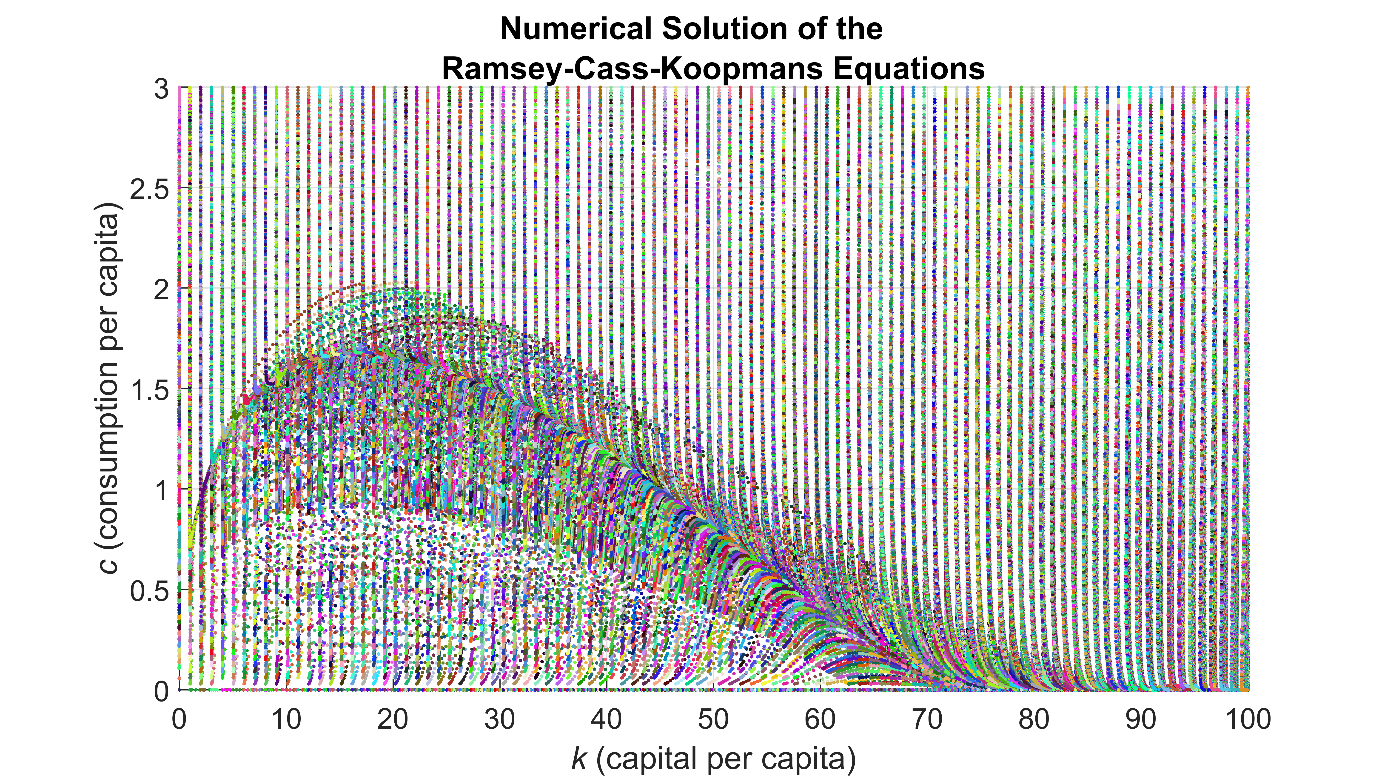


Figure 10. Solution paths of the RCK model starting from different initial conditions.

**Summary**

In this article we demonstrated that a system of coupled ODEs can be simulated using either MATLAB or Simulink, and that each approach brings benefits.

Using MATLAB, we transformed the equations to the form required by ode45 and used function handles. Using the Integrator block in Simulink, we implemented the differential equations in their native form.

Both MATLAB and Simulink enable us to compute derivatives automatically. In MATLAB, we use the gradient function in Symbolic Math Toolbox. In Simulink, we use the Derivative block and the solver Jacobian configuration setting of ode15s.

Simulink simplifies the modeling of a large or complex system of ODEs. Subsystems help us to organize our model by grouping functionally related blocks The Simulink model window provides an intuitive visual layout of the model.

Both approaches enable parallelization. In both MATLAB and Simulink we use the parfor construct. In Simulink we use the sim command to simulate the model programmatically.

Both MATLAB and Simulink provide an integrated modeling environment for solving and visualizing systems of ODEs. The method you use depends on the size and complexity of your model as well as your familiarity with each approach.

**Products Used**

MATLAB

Simulink

Parallel Computing Toolbox

**References**

1. David Kendrick, *Applications of Control Theory to Macroeconomics* (1976)

(<http://www.nber.org/chapters/c10438.pdf>)

1. C. Groth, *Lecture Notes in Macroeconomics*, 2011

(<http://www.econ.ku.dk/okocg/MAT-OEK/Mak%C3%98k2/Mak%C3%98k2-2011/Lectures%20and%20lecture%20notes/Ch10-2011-1.pdf>)

1. Christopher D. Carroll, The Ramsey/Cass-Koopmans (RCK) Model, November 2011 (<http://www.econ2.jhu.edu/people/ccarroll/public/lecturenotes/Growth/RamseyCassKoopmans.pdf>)
2. Pierre-Olivier Gourinchas, The Ramsey-Cass-Koopmans Model, 2014 (<http://eml.berkeley.edu/~webfac/gourinchas/e202a_f14/Notes_Ramsey_Cass_Koopmans_pog.pdf>)

**Learn More**

Stiff Differential Equations (<http://www.mathworks.com/company/newsletters/articles/stiff-differential-equations.html>)

Tips for Simulating Models in Parallel (<http://blogs.mathworks.com/simulink/2016/05/05/tips-for-simulating-models-in-parallel/>)