Simulating the Ramsey-Cass-Koopmans Model using MATLAB and Simulink

By Sonia Bridge and Ken Deeley, MathWorks

Many economic and financial models involve systems of differential equations with no explicit analytical solution. Solving these systems numerically is a key challenge for economists and other financial professionals.

The fundamental Ramsey-Cass-Koopmans (RCK) model aims to explain long-term economic growth in terms of capital accumulation and consumption growth [1-3]. The core RCK model is two-dimensional, comprising two coupled ordinary differential equations (ODEs) for per-capita wealth (*k*) and per-capita consumption (*c*). The phase portrait of the model is shown in figure 1.

This article presents a complete workflow showing how both MATLAB and Simulink can be used to create, simulate and visualize the RCK model. Simulink is a block diagram environment that can be used for modeling time-varying systems with feedback. Although Simulink provides an intuitive visual representation of models, it is not a tool typically used by financial analysts. However, for financial models comprising ordinary differential equations, Simulink can be utilized as an appropriate modeling and presentation environment.

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

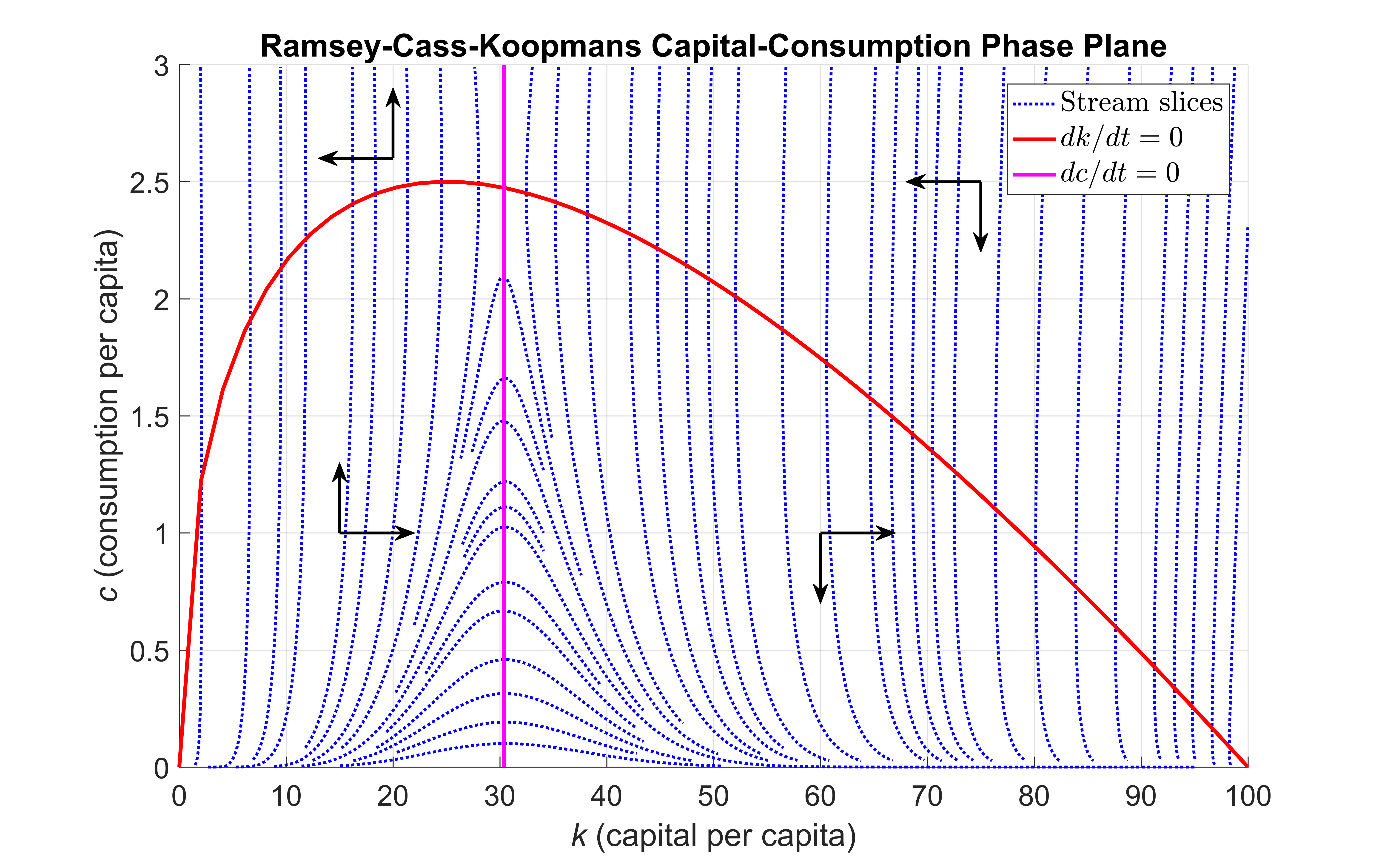


Figure 1: Phase portrait of the Ramsey-Cass-Koopmans system of ordinary differential equations

**The Ramsey-Cass-Koopmans Model**

The core RCK model equations for per-capita wealth (*k*) and per-capita consumption (*c*) are as follows.

Since *k* and *c* 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 *k* and a capital elasticity parameter (the responsiveness of the output production to changes in the input capital);
* is the growth rate of labor productivity (e.g., due to technological innovation or efficiency improvements);
* is the growth rate of labor supply (e.g., due to migration or population increase);
* is the depreciation rate of capital (e.g., 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 (e.g., 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 that they are expressed in the standard form to be solved on the time interval and subject to the initial condition We note that and are vector-valued if there are multiple unknown functions of time.

We begin by defining the necessary model parameters in a structure variable params, and write a vector-valued function RCK\_Equations representing the right-hand side of the standard differential equation This function returns a 2-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. Encapsulating these functions in separate files makes it 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 parametrizing RCK\_Equations with the predefined parameters structure params. This function is required to be a function of time (t) and state (Y) only.

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

We ensure that both the per-capita wealth and consumption remain nonnegative over time using odeset.

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

Taking the initial conditions to be and and allowing time to run from 0 to 1.5 units, we can now solve the system using ode45. The outputs of ode45 are time and state, so because we create a time vector t directly, we only request the second output 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

The MATLAB visualization function comet allows us to create an animated trajectory of the solution path. 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 as shown in figure 1.

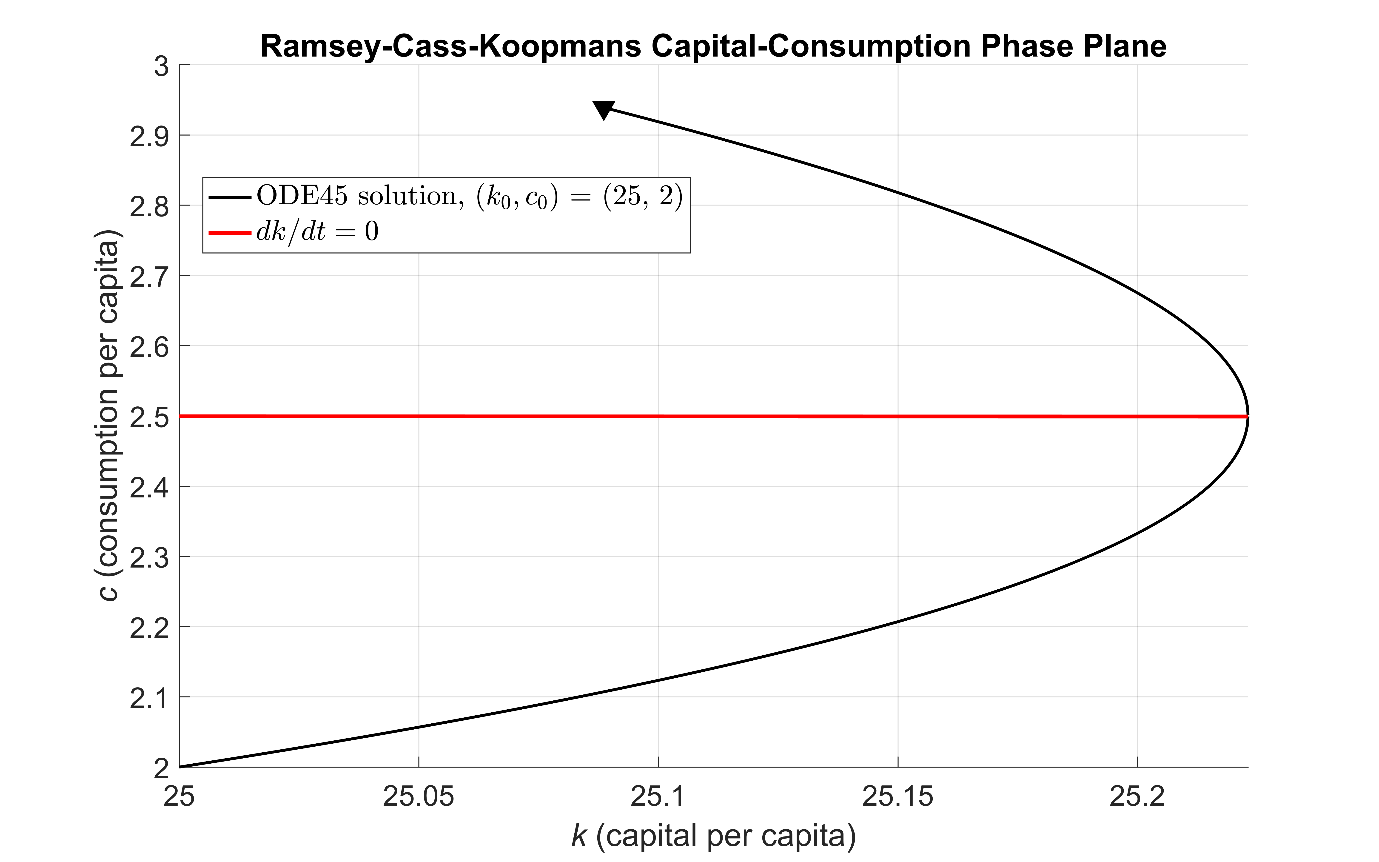


Figure 2: Solution trajectory starting from the point obtained by solving the coupled system of equations directly using ode45.

**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 This is 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 points. After computing the corresponding 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 mentioned in [2], there is no analytical solution for the model’s transition to its steady state. However, we can use the time elimination technique [2] to obtain the following:

.

Integrating with respect to allows us to obtain a solution trajectory for as a function of . To avoid problems evaluating when its numerator or denominator are zero, we split up the -domain into two parts: one to the left of and one to the right [2]. Applying the same technique (using ode45) as described in the previous section gives the solution trajectory as shown in figure 3.

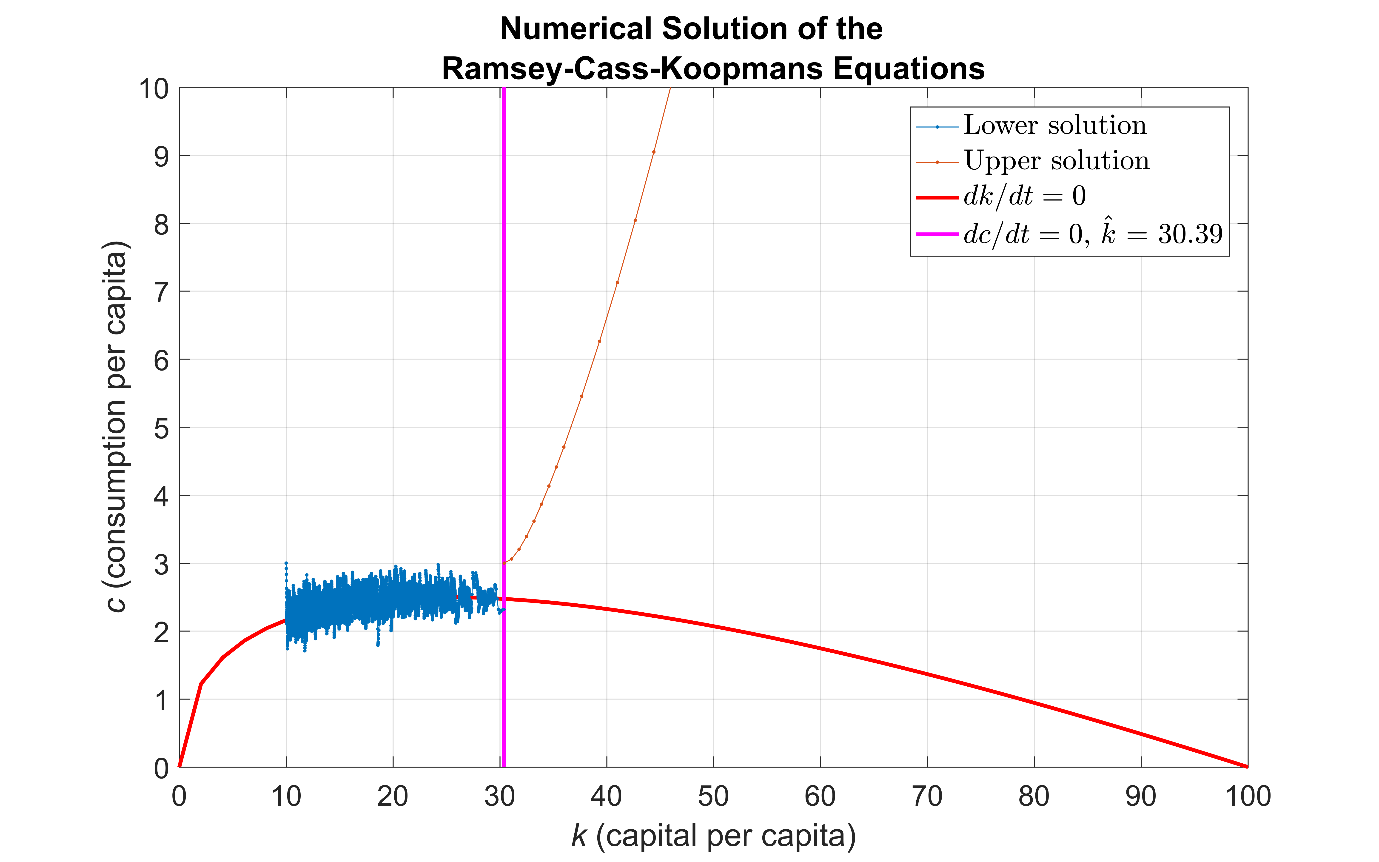


Figure 3: Upper and lower solution paths for a consumption strategy obtained using the time-elimination method.

We observe 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 [4]. In this case, instead of using ode45, we can 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. Note that for larger or more complex systems, we could use Symbolic ToolboxTM to compute analytic Jacobians without manual calculation.

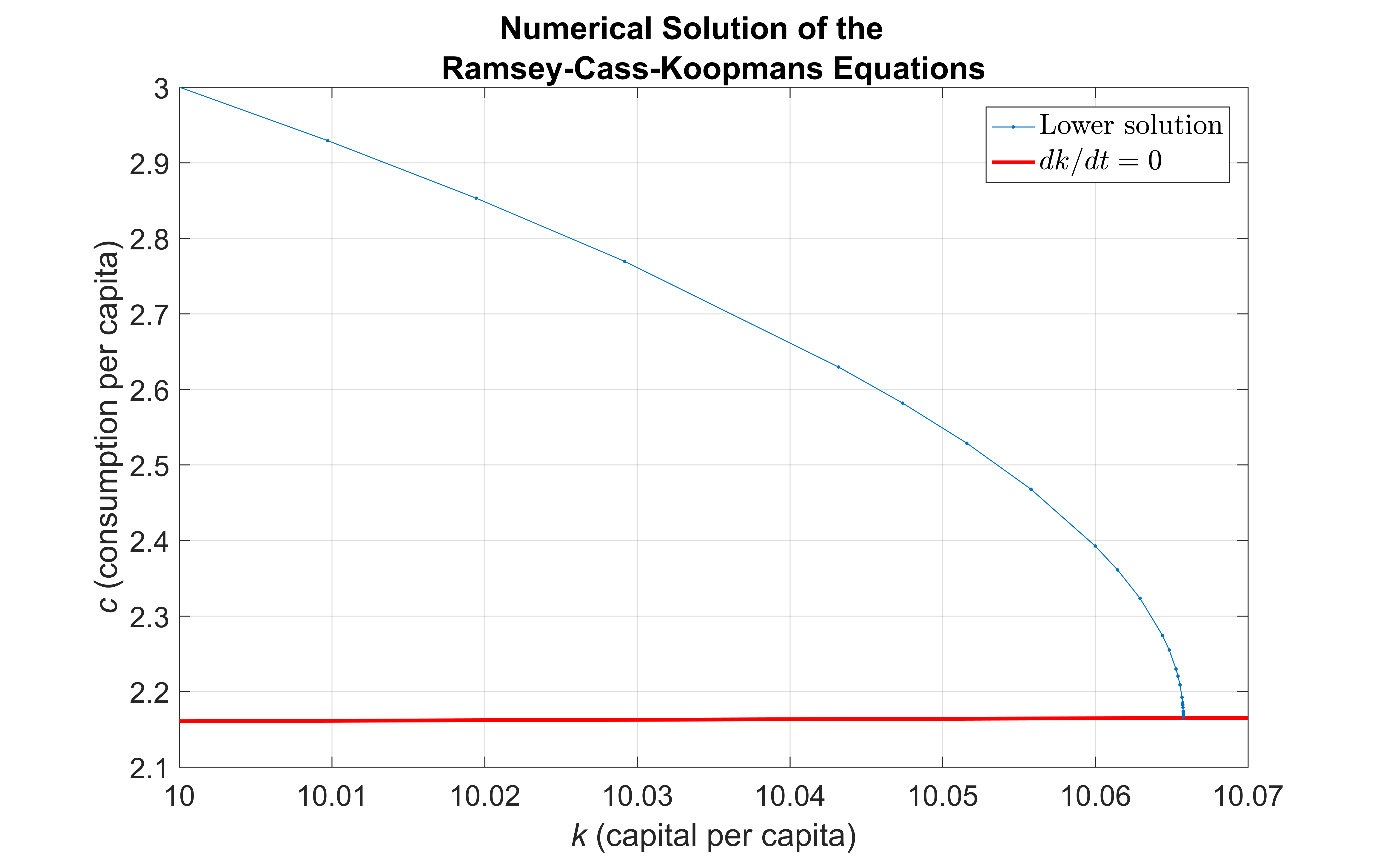


Figure 4: Lower solution path obtained using the stiff solver ode15s.

**Creating the RCK Model using Simulink**

**TODO: Sonia**

* + **More low-level details needed (intro Simulink, block diagrams)**
  + **More hand-holding**

**Running Simulations Efficiently 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 of these simulations can be run independently of the others, which is ideal for a parallel implementation 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 would like 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. This produces the solution trajectories shown in figure 5.

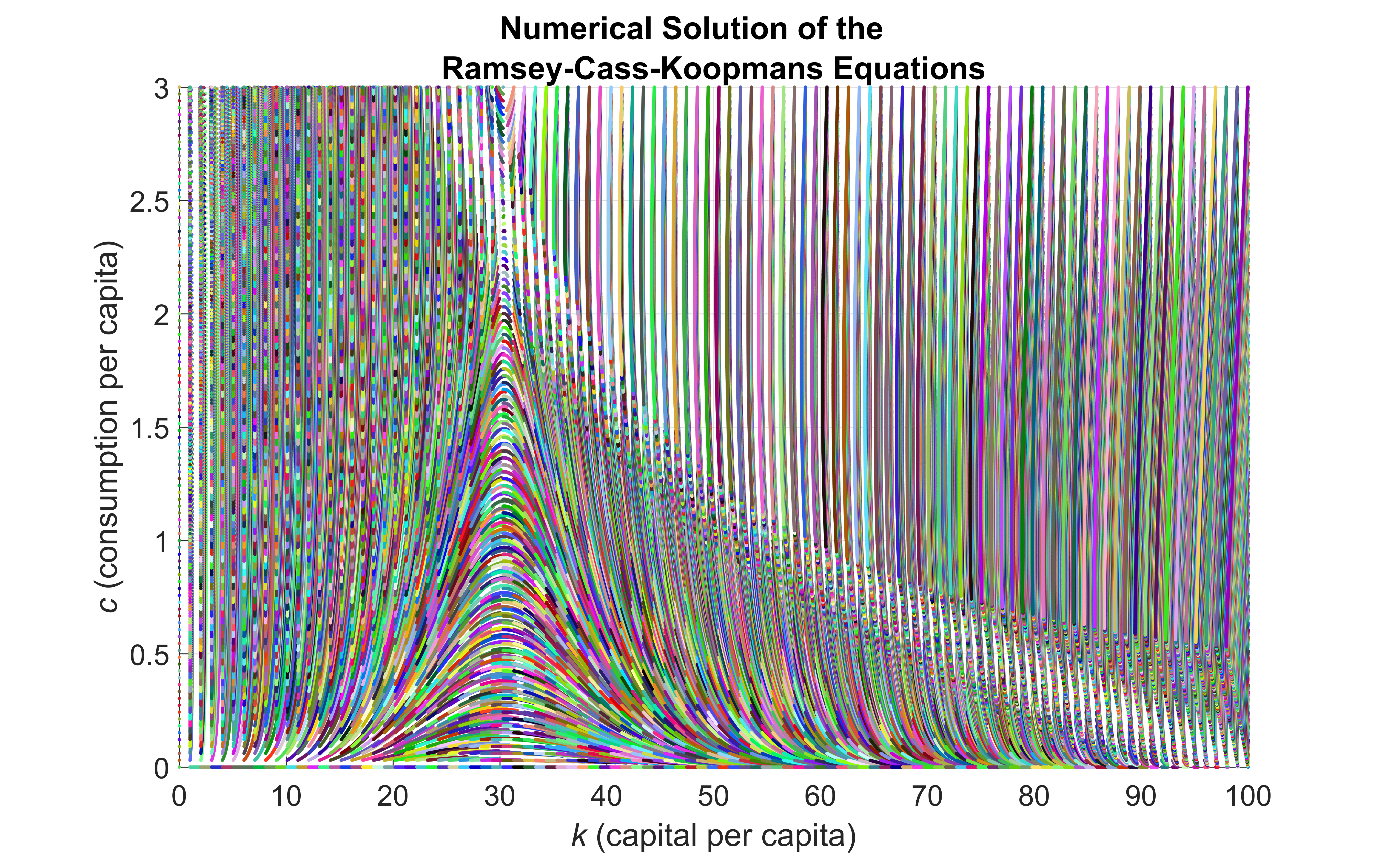


Figure 5: Solution paths of the RCK model starting from different initial conditions.

**TODO: Sonia**

**Parallelization of the Simulink model.**

**Summary and Next Steps**

**TODO**

* + Discussion: feedback loops clearly graphically represented in Simulink
  + 2x2 summary table (presentation styles/modelling styles – what types of models are suitable)? Different ways of thinking about the models.
  + Time-varying parameters: roughly the same for both approaches. Put in further improvements section.

% \* Simulink plus: work the equations "as is" - no transformation/rewriting

% of the equations required to work with Simulink, whereas with ODE45 you

% need to write the equations in standard form.

% \* Simulink plus: can compute the derivatives of f(k) automatically,

**Products Used**

MATLAB

Simulink

Parallel Computing Toolbox

**References**

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