# Stochastic growth model: an R package for a solution by discretization

Thomas Vigié

**Abstract**

This paper presents an R package that solves stochastic growth models by discretization. The model allows for different specifications, and vectorized code makes its use fast and straightforward. Keywords: Stochastic Growth model, Macroeconomics

## 1 Introduction

The stochastic growth model is one of the workhorse models in macroeconomics, used to analyze intertemporal decision making in the presence of uncertainty. Closed form solutions are available for particular values of the parameters, often unrealistic. Three main approaches have been used to find its solutions: polynomials, first order disturbances and value function iteration. The first two approaches are extensively described in Fernández-Villaverde *et al.* (2016) but no algorithm has been coded for a general use, due to the specific nature of each model. The standard stochastic growth model is a reference however, so we present its solution by value function iteration (actually, iteration on the decision rule) for a general set of parameters that the user has to supply. Default values are included.

## 2 The generic stochastic growth model

Consider an infinitely lived representative consumer who seeks to solve:

$$\underset{\{c_t,k_{t+1}\}_{t=0}^{\infty}}{Max}\left\{\mathbb{E}_0\left[\sum_{t=0}^{\infty}\beta^t u\left(c_t\right)\right]\right\}$$

subject to

$$\begin{cases} c_t + k_{t+1} = (1-\delta)\,k_t + f\left(k_t, z_t\right) & \forall t \\ ln\left(z_t\right) = \rho ln\left(z_{t-1}\right) + \varepsilon_t & \varepsilon_t \sim \mathcal{N}\left(0,\sigma^2\right)\ \forall t \\ k_{t+1} \geq 0 & \forall t \\ k_0 > 0 \end{cases}$$

where $\beta \in [0,1)$ is the discount factor, $u\left(.\right)$ is quasi concave and differentiable in its argument, $f\left(.\right)$ is a production function and $\delta \in [0,1]$ is the depreciation rate. The operator $\mathbb{E}_t$ refers to the expectation operator conditional on the information set at time $t$, i.e. the set of variables known at time $t$. Such a problem is now common in the macroeconomic literature and can be solved by hand for $\delta = 1$. For more general cases, one can use Bellman's principle of optimality, a Hamiltonian approach or a Lagrange approach. We present the latter here by introducing the Lagrangean function:

$$\mathcal{L}\left(\{c_t, k_{t+1}, \lambda_t\}_{t=0}^{\infty}\right) = \mathbb{E}_0\left[\sum_{t=0}^{\infty}\beta^t u\left(c_t\right) + \lambda_{t+1}\left(\left(1-\delta\right)k_t + f\left(k_t,z_t\right) - c_t - k_{t+1}\right)\right]$$

Conditions on $c_t$ and $k_{t+1}$ can be checked after looking at the first order conditions that lead to the two optimality conditions:

$$\begin{cases} \mathbb{E}_t \left[ \beta \frac{u'(c_{t+1})}{u'(c_t)} \left( 1 + f' \left( k_{t+1}, z_{t+1} \right) - \delta \right) - 1 \right] = 0 & \forall t \\ c_t + k_{t+1} = (1 - \delta) k_t + f \left( k_t, z_t \right) & \forall t \end{cases}$$

While the second equation is the budget constraint of the consumer presented above, the first equation, called the Euler equation, determines the tradeoff between consumption at time $t$ and $t+1$ along the optimal trajectory. Intuitively, it equates the mariginal utility from consuming an extra unit of consumption today and saving this extra unit to consume it tomorrow, delivering a discounted utility $u' \left( c_{t+1} \right)$ multiplied by the unit of consumption plus its return net of depreciation. Those equations deliver the policy functions as a function of the state variables:

$$\begin{cases} c_t^* = g_c \left( k_t, z_t \right) \\ k_{t+1}^* = g_k \left( k_t, z_t \right) \end{cases}$$

Note that in the case where $\delta = 1$, the policy functions for capital and consumption have a close form:

$$\begin{cases} g_c \left( k_t, z_t \right) = A z_t k_t^{\alpha} (1 - \beta \alpha) \\ g_k \left( k_t, z_t \right) = \beta \alpha A z_t k_t^{\alpha} \end{cases}$$

# 3 Discretization of the AR(1) process

In order to solve the model by value function iteration, i.e. find the policy function for any value of $k$ and $z$, we follow the contribution of Tauchen (1986) that proposes a method to convert an AR(1) process into a time-homogeneous Markov process with a finite number of states. The package Rtauchen implements that procedure in R. For each triplet $(m, \sigma, \rho)$ where $\sigma$ and $\rho$ are defined as in section 1 and $m$ is the number of states to create, the function Tgrid creates a vector of Markov states $z$, and the function Rtauchen creates the $[m, m]$ probability transition matrix $\Gamma$. For the remainder of the paper, we adopt the convention that the columns of $\Gamma$ sum to one, i.e. $\sum_{j=1}^{m} \Gamma_{i,j} = 1 \quad \forall i = 1, ..., m$. Then, the distribution of states at period $t+1$ given time $t$ is given by: $\pi_{t+1} = \Gamma \pi_t$. An extra parameter can be changed to control the range of values for the states around zero (see the Rtauchen documentation).

# 4 Solution by vectorization

Finding the policy functions through vectorization implies that the optimal values of $c$ and $k$ lie in a finite set, for which the expectation of the future value function is computed, and the values that maximize it are the optimal decisions[1].

The algorithm relies on the iteration of the value function maximization until convergence. We define the value function as

$$V(k, z) = \underset{\{c, k'\}}{Max} \left\{ u(c) + \beta \mathbb{E} \left[ V \left( k', z' \right) \right] \right\}$$

subject to

$$\begin{cases} c + k' = (1 - \delta) k + f(k, z) \\ k' \geq 0 \end{cases}$$

[1]The notation and description of the algorithm borrow heavily from Paul Klein's lecture notes of the course ECON 808 taught at Simon Fraser University in the fall 2014. URL: http://paulklein.ca/newsite/teaching/808.php

We can rewrite it as:

$$V(k,z) = \underset{\{k' \geq 0\}}{Max} \left\{ u\left( (1-\delta)k + f(k,z) - k' \right) + \beta \mathbb{E}\left[ V(k',z') \right] \right\} \tag{1}$$

where the expectation is taken over $z'$. Formulating the problem as above allows to solve it in one variable, from which the consumption policy function can be obtained through the budget constraint. Define now a capital grid $k = \left(k^1, k^2, ..., k^n\right)^t$. To avoid confusion in the notation, $a^t$ denotes the transpose of the vector or matrix $a$ whereas $a'$ is the notation for the Bellman formulation fo the problem. Analoguously, define $z = \left(z^1, z^2, ..., z^m\right)^t$. The vector $k$ is repeated $n$ times column-wise, and $m$ times row-wise (i.e for each of the state values $z$), in the $[mn, n]$ matrix $K$. Let $K'$ be a $[nm, n]$ matrix defined by repeating $k^t$ $nm$ times row-wise. The function $f(k,z)$ is evaluated at each pair of points $(k,z)$ in the matrix $y$ of size $[n,m]$ where $y_{i,j} = f\left(k^i, z^j\right)$. The matrix $Y$ is of size $[nm, n]$ is then obtained by repeating $vec(y)$ (of dimension $[nm, 1]$) $n$ times column-wise, thus matching the dimension of $K$ and $K'$. The $[nm, n]$ matrix $U$ representing the utility function is obtained by getting $u((1-\delta)K + Y - K')$, with each element of row i reporting the utility levels for the current capital value $k^i$ and all the possible values of $k'$. Every block of $n$ rows, $z$ changes and affects the utility through the production function. Hence, $U_{1,1}$ and $U_{n+1,1}$ are different due to $z$ having changed from $z_1$ to $z_2$ . Table 1 summarizes the dimensions of the objects defined and their description.

| object | dimension | description |
|---|---|---|
| $k$ | $[n, 1]$ | the initial capital grid |
| $z$ | $[m, 1]$ | the Markov states vector |
| $K$ | $[nm, n]$ | $K = \begin{pmatrix} k & k & \cdots & k \\ k & k & \cdots & k \\ \vdots & \vdots & \ddots & \vdots \\ k & k & \cdots & k \end{pmatrix}$ |
| $K'$ | $[nm, n]$ | $K' = nm$ times $\left\{ \begin{pmatrix} k^t \\ k^t \\ \\ \vdots \\ k^t \end{pmatrix} \right.$ |
| $y$ | $[n, m]$ | $y_{i,j} = f\left(k^i, z^j\right)$ |
| $Y$ | $[nm, n]$ | repetition of $vec(y)$, $n$ times column-wise |
| $\Gamma$ | $[m, m]$ | $\Gamma_{i,j} = \mathbb{P}\left(z_{t+1} = z^i \vert z_t = z^j\right)$ |
| $U$ | $[nm, n]$ | utility function evaluated at all the possible values of the budget constraint |

The algorithm goes as follows: consider the first row of the matrix $U$. Given the first value of the capital grid and the first Markov state for $z$, it associates a utility level for each value of $k'$. Given an initial guess for $V(k', z')$, say with a matrix of zeros of size $[mn, n]$, its expected value $\mathbb{E}\left[V(k', z')\right]$ is obtained by using the appropriate row of the probability transition matrix $\Gamma$ on each row of the matrix $U$. The $Max$ operator then picks, on each row of $U + \beta \mathbb{E}\left[V(k', z')\right]$, the highest element, and stores it in a vector of size $mn$, along with the coordinates of the chosen levels, the decision rule. The algorithm iterates over the latter vector, and updates the vector of value functions for each capital and total factor productivity level. Given that the conditions for the contraction mapping theorem to apply are satisfied, convergence of the value function is guaranteed in a finite number of iterations. Once convergence is reached, the decision rule indicates the level of capital $k'$ that solves (1) for a given $k$ and $z$. Using the budget constraint, the corresponding level of consumption can be obtained.

# 5 Data generating process

The decision rule allows one to generate data by starting at an initial level of capital and total factor productivity, say $(k_0, z_0)$. The optimal choice of capital for next period is determined through the decision rule and delivers $k_1$. Given the actual level of productivity $z_0$, the transition probability matrix is used to pick

another level for next period, $z_1$, and the output next period is computed by using $f(k_1, z_1)$. The procedure can be iterated an arbitrary number of times to generate samples.

# 6 Other functions

## 6.1 Stationary distribution of a Markov chain and moments

While conditional moments can be straightforwardly computed using the row of the probability transition matrix $\Gamma$ and the vector of states, for unconditional moments, one needs to compute the stationary distribution. It is the distribution of states $\pi_0$ such that: $\Gamma\pi_0 = \pi_0$ . It can be computed by iterating over the equation $\Gamma\pi = \pi$ as convergence is assured. Once it is obtained, one can compute unconditional moments using the stationary distribution and the column vector of states $z$. The unconditional mean $\mu$ is $\mu = \pi_0^t z = \sum_{i=1}^{m} \pi_{0i} z_i$. The unconditional variance $\sigma^2$ is $\sigma^2 = \sum_{i=1}^{m} \pi_{0i} z_i^2 - \mu^2$.

```
> library(sgmodel)
> a <- c(-1, 1)   # States of the Markov chain
> A <- matrix(c(0.5, 0.5, 0.6, 0.4), 2, 2)   # Probability transition matrix
> Markovmoments(a, A)

$Expectation
     [,1]
[1,]    0

$Variance
     [,1]
[1,]    1

$Autocovariance
     [,1]
[1,] -0.1

$Autocorrelation
     [,1]
[1,] -0.1

$`Stationary distribution`
     [,1]
[1,]  0.5
[2,]  0.5
```

## 6.2 Utility functions

The package contains generic utility functions, including the most commonly used ones in the economic literature. The functions are defined for scalars if the argument `ngoods` is missing or set to zero. The utility functions available with the argument `type` are `log`(default `type` option), `CARA`, `CRRA`, `Cobb-Douglas` and `CES`(if `ngoods` is set to a higher value than one). If `ngoods` is higher than one, the utility function is separable in its arguments for the case `CARA`, `log` and `CRRA`. The following table describes the functional forms for the value of `prefparam` equal to $a$, and a number of goods equal to $n$:

| utility type | functional form $n = 1$ | functional form $n > 1$ |
|---|---|---|
| Cobb-Douglas | $u(x) \equiv Ax^a$ | $u(x) \equiv A \prod_{i=1}^{n} x_i^a$ |
| log | $u(x) \equiv Aln(x)$ | $u(x) \equiv A \sum_{i=1}^{n} ln(x_i)$ |
| CES | $u(x) \equiv Ax$ | $u(x) \equiv A \left( \sum_{i=1}^{n} x_i^{\frac{1}{a}} \right)^a$ |
| CRRA | $u(x) \equiv A\frac{x^{1-a}}{1-a}$ | $u(x) \equiv A \sum_{i=1}^{n} \frac{x_i^{1-a}}{1-a}$ |
| CARA | $u(x) \equiv Ae^{-ax}$ | $u(x) \equiv A \sum_{i=1}^{n} e^{-ax_i}$ |

```
> #install.packages("sgmodel")
> library(sgmodel)
> x <- c(exp(1), exp(1))
> A <- 2
> type <- "log"
> ngoods <- 2
> util(x = x, A = A, type = type, ngoods = ngoods)
```

```
[1] 4
```

# 7 The sgmodel package

The package contains as it main function the solution to a stochastic growth model, as exposed in the previous sections. It also contains the most common utility functions in the economic literature, functions to compute the moments of a stationary Markov chain and a function to generate samples according to the process exposed in section 4. The sgmodel is available on the CRAn website, and thus can be installed and loaded the usual way.

```
> library(sgmodel)
```

The main function is sgmodel() that returns a list summarizing the features of the model and the result of the optimization process. All the arguments have default values, except rho and sigma, which govern the TFP process. Hence, the following command is enough:

```
> model <- sgmodel(rho = 0.2, sigma = 0.02)
```

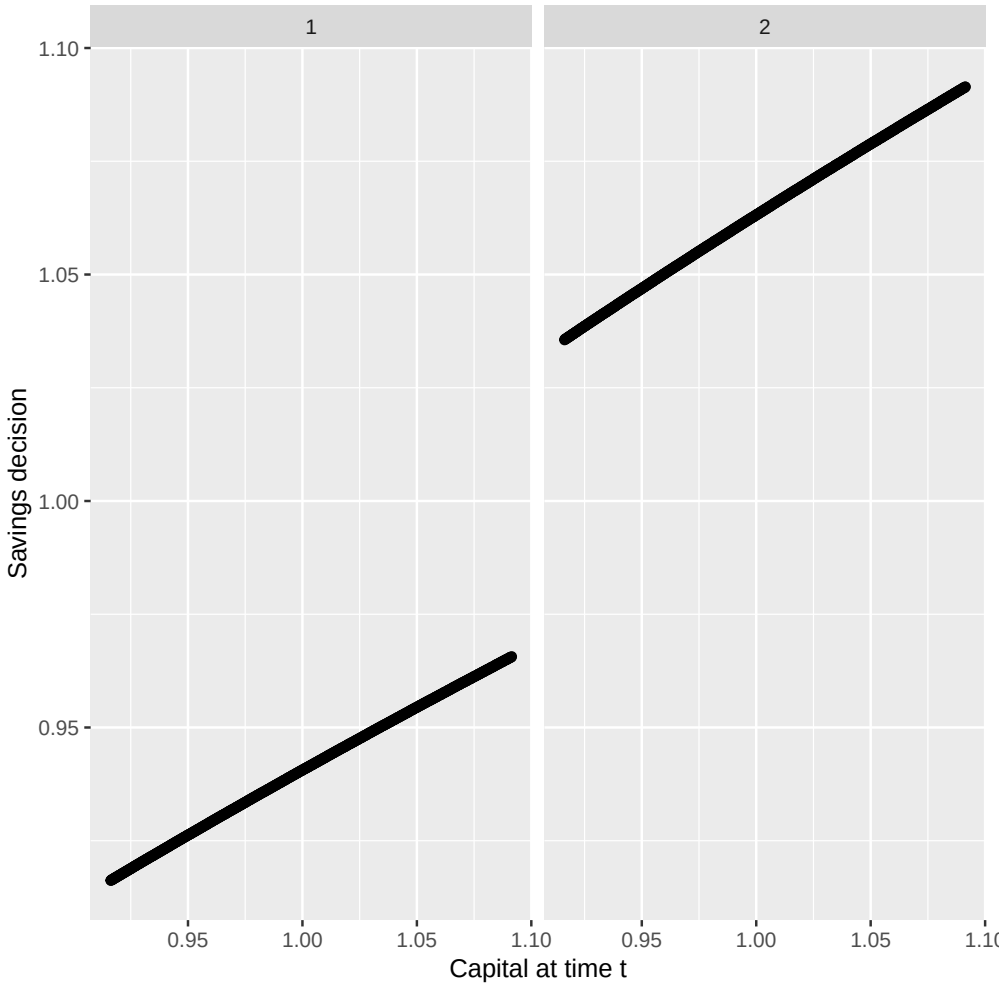
It returns a list whose first element is the capital grid used, $Savingsand$Consumption are the solution vectors to the problem, and the rest of the elements are the parameters of the model. The function summary$_sgmodalsoreportsthelistofparametersofthemodel.Thefunctionplot_sgmodplotsthe$Savings vector for each state of the Markov TFP process. The plot is made using ggplot from the ggplot2 package(see Wickham (2016)).

```
> summary_sgmod(model)
```

```
$description
$description$`Utility function`
[1] "log"

$description$`Capital share`
```

```
[1] 0.3

$description$`Discount factor`
[1] 0.95

$description$Depreciation
[1] 1

$description$Rho
[1] 0.2

$description$Sigma
[1] 0.02

$description$`Number of TFP states`
[1] 2


$call
NULL

attr(,"class")
[1] "summary.sgmod"
```

*> plot_sgmod(model)*




## 8 Conclusion

This package proposes a way to solve simple stochastic growth models, along with some functions that compute utility levels and moments of a stationary Markov process. The functions can be used to generate data and estimate structural parameters, or serve as a benchmark to solve other types of models. Using the same type of algorithm, one can extend this class of models to allow for heterogeneous agents regarding their productivity, and extend the set of decision variables to include labou supply.

## 9 Package details

The package sgmodel can be found on the comprehensive R archive network (CRAN). It has been written in R entirely under the GPL-3 license. It depends on the ggplot2( see Wickham (2016)), ramify, Rtauchen(Tauchen (1986)) and tidyverse (which contains ggplot2. See tidyverse) packages.

## References

Fernández-Villaverde, Jesús, Rubio-Ramírez, Juan Francisco, & Schorfheide, Frank. 2016. Solution and estimation methods for DSGE models. *Pages 527–724 of: Handbook of macroeconomics*, vol.

2. Elsevier.

Tauchen, George. 1986. Finite state markov-chain approximations to univariate and vector autoregressions. *Economics letters*, **20**(2), 177–181.

Wickham, Hadley. 2016. *ggplot2: elegant graphics for data analysis*. Springer.