solowpy Documentation

Release 0.2.0-alpha

David R. Pugh

February 20, 2017

Contents

CHAPTER 1

Contents:

License

solowpy

solowpy package

Submodules

solowpy.ces module

Solow model with constant elasticity of substitution (CES) production:

$$F(K, AL) = \left[\alpha K^{\rho} + (1-\alpha)(AL)^{\rho}\right]^{\frac{1}{\rho}}$$

where $0 < \alpha < 1$ and

$$\rho = \frac{\sigma - 1}{\sigma}$$

where $-\infty \leq \rho \leq 1$ and $0 \leq \sigma \leq \infty$ is the elasticity of substitution between capital and effective labor in production.

class `solowpy.ces.CESModel`(*params*)

Bases: *solowpy.model.Model*

Attributes

`effective_depreciation_rate`	Effective depreciation rate for capital stock (per unit effective labor).
`intensive_output`	Symbolic expression for the intensive form of aggregate production.
`ivp`	Initial value problem
`k_dot`	Symbolic expression for the equation of motion for capital (per unit effective labor).
`marginal_product_capital`	Symbolic expression for the marginal product of capital (per unit effective labor).
`output`	Symbolic expression for the aggregate production function.
`params`	Dictionary of model parameters.
solow_residual	Symbolic expression for the Solow residual which is used as a measure of technology.
`speed_of_convergence`	The speed of convergence for the Solow model.
steady_state	Steady state value of capital stock (per unit effective labor).

Methods

`evaluate_actual_investment`(k)	Return the amount of output (per unit of effective labor) invested in the productio
`evaluate_consumption`(k)	Return the amount of consumption (per unit of effective labor).
`evaluate_effective_depreciation`(k)	Return amount of Capital stock (per unit of effective labor) that depreciaties due t
`evaluate_intensive_output`(k)	Return the amount of output (per unit of effective labor).
`evaluate_k_dot`(k)	Return time derivative of capital stock (per unit of effective labor).
`evaluate_mpk`(k)	Return marginal product of capital stock (per unit of effective labor).
`evaluate_output_elasticity`(k)	Return elasticity of output with respect to capital stock (per unit effective labor).
`evaluate_solow_residual`(Y, K, L)	Return Solow residual.
`find_steady_state`(a, b[, method])	Compute the equilibrium value of capital stock (per unit effective labor).
`linearized_solution`(t, k0)	Compute the linearized solution for the Solow model.
`plot_factor_shares`(ax[, Nk])	Plot income/output shares of capital and labor inputs to production.
`plot_intensive_investment`(ax[, Nk])	Plot actual investment (per unit effective labor) and effective depreciation.
`plot_intensive_output`(ax[, Nk])	Plot intensive form of the aggregate production function.
`plot_phase_diagram`(ax[, Nk])	Plot the model's phase diagram.
`plot_solow_diagram`(ax[, Nk])	Plot the classic Solow diagram.

solow_residual

Symbolic expression for the Solow residual which is used as a measure of technology.

Getter Return the symbolic expression.

Type sym.Basic

steady_state

Steady state value of capital stock (per unit effective labor).

Getter Return the current steady state value.

Type float

Notes

The steady state value of capital stock (per unit effective labor) with CES production is defined as

$$k^* = \left[\frac{1-\alpha}{\left(\frac{g+n+\delta}{s}\right)^{\rho}-\alpha}\right]^{\frac{1}{\rho}}$$

where s is the savings rate, $g+n+\delta$ is the effective depreciation rate, and α controls the importance of capital stock relative to effective labor in the production of output. Finally,

$$\rho = \frac{\sigma-1}{\sigma}$$

where σ is the elasticity of substitution between capital and effective labor in production.

solowpy.cobb_douglas module

Solow growth model with Cobb-Douglas aggregate production:

$$F(K, AL) = K^{\alpha}(AL)^{1-\alpha}$$

where $0 < \alpha < 1$.

class `solowpy.cobb_douglas.`**`CobbDouglasModel`**(*params*)

Bases: *`solowpy.model.Model`*

Attributes

`effective_depreciation_rate`	Effective depreciation rate for capital stock (per unit effective labor).
`intensive_output`	Symbolic expression for the intensive form of aggregate production.
`ivp`	Initial value problem
`k_dot`	Symbolic expression for the equation of motion for capital (per unit effective labor).
`marginal_product_capital`	Symbolic expression for the marginal product of capital (per unit effective labor).
`output`	Symbolic expression for the aggregate production function.
`params`	Dictionary of model parameters.
`solow_residual`	Symbolic expression for the Solow residual which is used as a measure of technology.
`speed_of_convergence`	The speed of convergence for the Solow model.
`steady_state`	Steady state value of capital stock (per unit effective labor).

Methods

`analytic_solution`(t, k0)	Compute the analytic solution for the Solow model with Cobb-Douglas productio
`evaluate_actual_investment`(k)	Return the amount of output (per unit of effective labor) invested in the productio
`evaluate_consumption`(k)	Return the amount of consumption (per unit of effective labor).
`evaluate_effective_depreciation`(k)	Return amount of Capital stock (per unit of effective labor) that depreciaties due t
`evaluate_intensive_output`(k)	Return the amount of output (per unit of effective labor).
`evaluate_k_dot`(k)	Return time derivative of capital stock (per unit of effective labor).
`evaluate_mpk`(k)	Return marginal product of capital stock (per unit of effective labor).

Table 1.4 – continued from previous page

evaluate_output_elasticity(k)	Return elasticity of output with respect to capital stock (per unit effective labor).
evaluate_solow_residual(Y, K, L)	Return Solow residual.
find_steady_state(a, b[, method])	Compute the equilibrium value of capital stock (per unit effective labor).
linearized_solution(t, k0)	Compute the linearized solution for the Solow model.
plot_factor_shares(ax[, Nk])	Plot income/output shares of capital and labor inputs to production.
plot_intensive_investment(ax[, Nk])	Plot actual investment (per unit effective labor) and effective depreciation.
plot_intensive_output(ax[, Nk])	Plot intensive form of the aggregate production function.
plot_phase_diagram(ax[, Nk])	Plot the model's phase diagram.
plot_solow_diagram(ax[, Nk])	Plot the classic Solow diagram.

analytic_solution(*t*, *k0*)

Compute the analytic solution for the Solow model with Cobb-Douglas production technology.

Parameters **t** : *numpy.ndarray*

Array of points at which the solution is desired.

k0 : (float)

Initial condition for capital stock (per unit of effective labor)

Returns **analytic_traj** : *numpy.ndarray* (shape=t.size, 2)

Array representing the analytic solution trajectory.

steady_state

Steady state value of capital stock (per unit effective labor).

Getter Return the current steady state value.

Type float

Notes

The steady state value of capital stock (per unit effective labor) with Cobb-Douglas production is defined as

$$k^* = \left(\frac{s}{g+n+\delta}\right)^{\frac{1}{1-\alpha}}$$

where s is the savings rate, $g+n+\delta$ is the effective depreciation rate, and α is the elasticity of output with respect to capital (i.e., capital's share).

solowpy.impulse_response module

Classes for generating and plotting impulse response functions.

class solowpy.impulse_response.**ImpulseResponse**(*model*)

Bases: object

Base class representing an impulse response function for a Model.

Attributes

`impulse`	Dictionary of new parameter values representing an impulse.
`impulse_response`	Impulse response functions generated by a shock to model parameter(s).
`kind`	The kind of impulse response function to generate.

Methods

`plot_impulse_response`(ax, variable[, log])	Plot an impulse response function.

N = 10

T = 100

impulse

Dictionary of new parameter values representing an impulse.

Getter Return the current impulse dictionary.

Setter Set a new impulse dictionary.

Type dictionary

impulse_response

Impulse response functions generated by a shock to model parameter(s).

Getter Return the current impulse response functions.

Type *numpy.ndarray*

kind

The kind of impulse response function to generate. Must be one of: 'levels', 'per_capita', 'efficiency_units'.

Getter Return the current kind of impulse responses.

Setter Set a new value for the kind of impulse responses.

Type str

plot_impulse_response(*ax*, *variable*, *log=False*)

Plot an impulse response function.

Parameters ax : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

variable : str

Variable whose impulse response functions you wish to plot.

impulse : dict

Dictionary of new parameter values representing the impulse whose model response you wish to plot.

kind : str (default='efficiency_units')

Whether you want impulse response functions in 'levels', 'per_capita', or 'efficiency_units'.

log : boolean (default=False)

Whether or not to have logarithmic scales on the vertical axes. Useful when plotting impulse response functions with kind='per_capita' or kind='levels'.

Returns A list containing:

irf_line : maplotlib.lines.Line2D

A Line2D object representing the impulse response for the requested variable.

bgp_line : maplotlib.lines.Line2D

A Line2D object representing the pre-impulse balanced growth path for the model.

solowpy.model module

The following summary of the [solow1956] model of economic growth largely follows [romer2011].

Assumptions

The production function The [solow1956] model of economic growth focuses on the behavior of four variables: output, *Y*, capital, *K*, labor, *L*, and knowledge (or technology or the "effectiveness of labor"), A. At each point in time the economy has some amounts of capital, labor, and knowledge that can be combined to produce output according to some production function, F.

$$Y(t) = F(K(t), A(t)L(t))$$

where *t* denotes time.

The evolution of the inputs to production The initial levels of capital, K_0, labor, L_0, and technology, A_0, are taken as given. Labor and technology are assumed to grow at constant rates:

$$\begin{aligned}\dot{A}(t) =& gA(t)\\ \dot{L}(t) =& nL(t)\end{aligned}$$

where the rate of technological progress, *g*, and the population growth rate, *n*, are exogenous parameters.

Output is divided between consumption and investment. The fraction of output devoted to investment, $0 < s < 1$, is exogenous and constant. One unit of output devoted to investment yields one unit of new capital. Capital is assumed to decpreciate at a rate $0 \le \delta$. Thus aggregate capital stock evolves according to

$$\dot{K}(t) = sY(t) - \delta K(t).$$

Although no restrictions are placed on the rates of technological progress and population growth, the sum of *g*, *n*, and δ is assumed to be positive.

The dynamics of the model

Because the economy is growing over time (due to exogenous technological progress and population growth) it is useful to focus on the behavior of capital stock per unit of effective labor, $k \equiv K/AL$. Applying the chain rule to the equation of motion for capital stock yields (after a bit of algebra!) an equation of motion for capital stock per unit of effective labor.

$$\dot{k}(t) = sf(k) - (g + n + \delta)k(t)$$

References

class `solowpy.model.`**`Model`**(*output*, *params*)

Bases: `object`

Attributes

effective_depreciation_rate	Effective depreciation rate for capital stock (per unit effective labor).
intensive_output	Symbolic expression for the intensive form of aggregate production.
ivp	Initial value problem
k_dot	Symbolic expression for the equation of motion for capital (per unit effective labor).
marginal_product_capital	Symbolic expression for the marginal product of capital (per unit effective labor).
output	Symbolic expression for the aggregate production function.
params	Dictionary of model parameters.
solow_residual	Symbolic expression for the Solow residual which is used as a measure of technology.
speed_of_convergence	The speed of convergence for the Solow model.
steady_state	Steady state value of capital stock (per unit effective labor).

Methods

evaluate_actual_investment(k)	Return the amount of output (per unit of effective labor) invested in the productio
evaluate_consumption(k)	Return the amount of consumption (per unit of effective labor).
evaluate_effective_depreciation(k)	Return amount of Capital stock (per unit of effective labor) that depreciaties due t
evaluate_intensive_output(k)	Return the amount of output (per unit of effective labor).
evaluate_k_dot(k)	Return time derivative of capital stock (per unit of effective labor).
evaluate_mpk(k)	Return marginal product of capital stock (per unit of effective labor).
evaluate_output_elasticity(k)	Return elasticity of output with respect to capital stock (per unit effective labor).
evaluate_solow_residual(Y, K, L)	Return Solow residual.
find_steady_state(a, b[, method])	Compute the equilibrium value of capital stock (per unit effective labor).
linearized_solution(t, k0)	Compute the linearized solution for the Solow model.
plot_factor_shares(ax[, Nk])	Plot income/output shares of capital and labor inputs to production.
plot_intensive_investment(ax[, Nk])	Plot actual investment (per unit effective labor) and effective depreciation.
plot_intensive_output(ax[, Nk])	Plot intensive form of the aggregate production function.
plot_phase_diagram(ax[, Nk])	Plot the model's phase diagram.
plot_solow_diagram(ax[, Nk])	Plot the classic Solow diagram.

`effective_depreciation_rate`

Effective depreciation rate for capital stock (per unit effective labor).

Getter Return the current effective depreciation rate.

Type float

Notes

The effective depreciation rate of physical capital takes into account both technological progress and population growth, as well as physical depreciation.

`evaluate_actual_investment`(*k*)

Return the amount of output (per unit of effective labor) invested in the production of new capital.

Parameters **k** : array_like (float)

Capital stock (per unit of effective labor)

Returns **actual_inv** : array_like (float)

Investment (per unit of effective labor)

evaluate_consumption(*k*)

Return the amount of consumption (per unit of effective labor).

Parameters **k** : *numpy.ndarray* (float)

Capital stock (per unit of effective labor)

Returns **c** : *numpy.ndarray* (float)

Consumption (per unit of effective labor)

evaluate_effective_depreciation(*k*)

Return amount of Capital stock (per unit of effective labor) that depreciaties due to technological progress, population growth, and physical depreciation.

Parameters **k** : array_like (float)

Capital stock (per unit of effective labor)

Returns **effective_depreciation** : array_like (float)

Amount of depreciated Capital stock (per unit of effective labor)

evaluate_intensive_output(*k*)

Return the amount of output (per unit of effective labor).

Parameters **k** : *numpy.ndarray* (float)

Capital stock (per unit of effective labor)

Returns **y** : *numpy.ndarray* (float)

Output (per unit of effective labor)

evaluate_k_dot(*k*)

Return time derivative of capital stock (per unit of effective labor).

Parameters **k** : *numpy.ndarray* (float)

Capital stock (per unit of effective labor)

Returns **k_dot** : *numpy.ndarray* (float)

Time derivative of capital stock (per unit of effective labor).

evaluate_mpk(*k*)

Return marginal product of capital stock (per unit of effective labor).

Parameters **k** : *numpy.ndarray* (float)

Capital stock (per unit of effective labor)

Returns **mpk** : *numpy.ndarray* (float)

Marginal product of capital stock (per unit of effective labor).

evaluate_output_elasticity(*k*)

Return elasticity of output with respect to capital stock (per unit effective labor).

Parameters **k** : array_like (float)

Capital stock (per unit of effective labor)

Returns **alpha_k** : array_like (float)

Elasticity of output with respect to capital stock (per unit effective labor).

Notes

Under the additional assumption that markets are perfectly competitive, the elasticity of output with respect to capital stock is equivalent to capital's share of income. Since, under perfect competition, firms earn zero profits it must be true capital's share and labor's share must sum to one.

evaluate_solow_residual(*Y*, *K*, *L*)

Return Solow residual.

Parameters **k** : array_like (float)

Capital stock (per unit of effective labor)

Returns **residual** : array_like (float)

Solow residual

find_steady_state(*a*, *b*, *method='brentq'*, ***kwargs*)

Compute the equilibrium value of capital stock (per unit effective labor).

Parameters **a** : float

One end of the bracketing interval [a,b].

b : float

The other end of the bracketing interval [a,b]

method : str (default='brentq')

Method to use when computing the steady state. Supported methods are *bisect*, *brenth*, *brentq*, *ridder*. See *scipy.optimize* for more details (including references).

kwargs : optional

Additional keyword arguments. Keyword arguments are method specific see *scipy.optimize* for details.

Returns **x0** : float

Zero of f between a and b.

r : RootResults (present if `full_output = True`)

Object containing information about the convergence. In particular, `r.converged` is True if the routine converged.

intensive_output

Symbolic expression for the intensive form of aggregate production.

Getter Return the current intensive production function.

Type sympy.Basic

Notes

The assumption of constant returns to scale allows us to work the the intensive form of the aggregate production function, F. Defining $c = 1/AL$ one can write

$$F\left(\frac{K}{AL}, 1\right) = \frac{1}{AL}F(A, K, L)$$

Defining $k = K/AL$ and $y = Y/AL$ to be capital per unit effective labor and output per unit effective labor, respectively, the intensive form of the production function can be written as

$$y = f(k).$$

Additional assumptions are that f satisfies $f(0) = 0$, is concave (i.e., $f'(k) > 0$, $f''(k) < 0$), and satisfies the Inada conditions:

$$\lim_{k \to 0} = \infty$$
$$\lim_{k \to \infty} = 0$$

The [inada1964] conditions are sufficient (but not necessary!) to ensure that the time path of capital per effective worker does not explode.

ivp

Initial value problem

Getter Return instance of the ivp.IVP class representing the model.

Type ivp.IVP

Notes

The Solow model with can be formulated as an initial value problem (IVP) as follows.

$$\dot{k}(t) = sf(k(t)) - (g + n + \delta)k(t),\ t \geq t_0,\ k(t_0) = k_0$$

The solution to this IVP is a function $k(t)$ describing the time path of capital stock (per unit effective labor).

k_dot

Symbolic expression for the equation of motion for capital (per unit effective labor).

Getter Return the current equation of motion for capital.

Type sympy.Basic

Notes

Because the economy is growing over time due to technological progress, g, and population growth, n, it makes sense to focus on the capital stock per unit effective labor, k, rather than aggregate physical capital, K. Since, by definition, $k = K/AL$, we can apply the chain rule to the time derative of k.

$$\begin{aligned}\dot{k}(t) =& \frac{\dot{K}(t)}{A(t)L(t)} - \frac{K(t)}{[A(t)L(t)]^2}\left[\dot{A}(t)L(t) + \dot{L}(t)A(t)\right] \\ =& \frac{\dot{K}(t)}{A(t)L(t)} - \left(\frac{\dot{A}(t)}{A(t)} + \frac{\dot{L}(t)}{L(t)}\right)\frac{K(t)}{A(t)L(t)}\end{aligned}$$

By definition, $k = K/AL$, and by assumption $\dot{A}/A$ and $\dot{L}/L$ are g and n respectively. Aggregate capital stock evolves according to

$$\dot{K}(t) = sF(K(t), A(t)L(t)) - \delta K(t).$$

Substituting these facts into the above equation yields the equation of motion for capital stock (per unit effective labor).

$$\begin{aligned}\dot{k}(t) =& \frac{sF(K(t), A(t)L(t)) - \delta K(t)}{A(t)L(t)} - (g+n)k(t)\\ =& \frac{sY(t)}{A(t)L(t)} - (g+n+\delta)k(t)\\ =& sf(k(t)) - (g+n+\delta)k(t)\end{aligned}$$

linearized_solution(*t*, *k0*)

Compute the linearized solution for the Solow model.

Parameters **t** : *numpy.ndarray* (shape=(T,))

Array of points at which the solution is desired.

k0 : float

Initial condition for capital stock (per unit of effective labor)

Returns **linearized_traj** : *numpy.ndarray* (shape=t.size, 2)

Array representing the linearized solution trajectory.

marginal_product_capital

Symbolic expression for the marginal product of capital (per unit effective labor).

Getter Return the current marginal product of capital.

Type sympy.Basic

Notes

The marginal product of capital is defined as follows:

$$\frac{\partial F(K, AL)}{\partial K} \equiv f'(k)$$

where $k = K/AL$ is capital stock (per unit effective labor).

output

Symbolic expression for the aggregate production function.

Getter Return the current aggregate production function.

Setter Set a new aggregate production function

Type sympy.Basic

Notes

At each point in time the economy has some amounts of capital, K, labor, L, and knowledge (or technology), A, that can be combined to produce output, Y, according to some function, F.

$$Y(t) = F(K(t), A(t)L(t))$$

where t denotes time. Note that A and L are assumed to enter multiplicatively. Typically *A(t)L(t)* denotes "effective labor", and technology that enters in this fashion is known as labor-augmenting or "Harrod neutral."

A key assumption of the model is that the function F exhibits constant returns to scale in capital and labor inputs. Specifically,

$$F(cK(t), cA(t)L(t)) = cF(K(t), A(t)L(t)) = cY(t)$$

for any $c \geq 0$.

params

Dictionary of model parameters.

Getter Return the current dictionary of model parameters.

Setter Set a new dictionary of model parameters.

Type dict

Notes

The following parameters are required:

A0: float Initial level of technology. Must satisfy $A_0 > 0$.

L0: float Initial amount of available labor. Must satisfy $L_0 > 0$.

g [float] Growth rate of technology.

n [float] Growth rate of the labor force.

s [float] Savings rate. Must satisfy *0 < s < 1*.

delta [float] Depreciation rate of physical capital. Must satisfy $0 < \delta$.

Although no restrictions are placed on the rates of technological progress and population growth, the sum of g, n, and δ is assumed to be positive. The user mus also specify any additional model parameters specific to the chosen aggregate production function.

plot_factor_shares(*ax*, *Nk=1000.0*, ***new_params*)

Plot income/output shares of capital and labor inputs to production.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

capitals_share_line : maplotlib.lines.Line2D

A Line2D object representing the time path for capital's share of income.

labors_share_line : maplotlib.lines.Line2D

A Line2D object representing the time path for labor's share of income.

plot_intensive_investment(*ax*, *Nk=1000.0*, ***new_params*)

Plot actual investment (per unit effective labor) and effective depreciation. The steady state value of capital stock (per unit effective labor) balance acual investment and effective depreciation.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

actual_investment_line : maplotlib.lines.Line2D

A Line2D object representing the level of actual investment as a function of capital stock (per unit effective labor).

breakeven_investment_line : maplotlib.lines.Line2D

A Line2D object representing the “break-even” level of investment as a function of capital stock (per unit effective labor).

ss_line : maplotlib.lines.Line2D

A Line2D object representing the steady state level of investment.

plot_intensive_output(*ax*, *Nk=1000.0*, ***new_params*)

Plot intensive form of the aggregate production function.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

intensive_output : maplotlib.lines.Line2D

A Line2D object representing intensive output as a function of capital stock (per unit effective labor).

plot_phase_diagram(*ax*, *Nk=1000.0*, ***new_params*)

Plot the model’s phase diagram.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

k_dot_line : maplotlib.lines.Line2D

A Line2D object representing the rate of change of capital stock (per unit effective labor) as a function of its level.

origin_line : maplotlib.lines.Line2D

A Line2D object representing the origin (i.e., locus of points where k_dot is zero).

ss_line : maplotlib.lines.Line2D

A Line2D object representing the steady state level of capital stock (per unit effective labor).

plot_solow_diagram(*ax*, *Nk=1000.0*, ***new_params*)

Plot the classic Solow diagram.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

actual_investment_line : maplotlib.lines.Line2D

A Line2D object representing the level of actual investment as a function of capital stock (per unit effective labor).

breakeven_investment_line : maplotlib.lines.Line2D

A Line2D object representing the "break-even" level of investment as a function of capital stock (per unit effective labor).

ss_line : maplotlib.lines.Line2D

A Line2D object representing the steady state level of investment.

solow_residual

Symbolic expression for the Solow residual which is used as a measure of technology.

Getter Return the symbolic expression.

Type sympy.Basic

speed_of_convergence

The speed of convergence for the Solow model.

Getter Return the current speed of convergence.

Type float

Notes

The following is a derivation for the speed of convergence λ:

$$\begin{aligned}\lambda \equiv -\frac{\partial \dot{k}(k(t))}{\partial k(t)}\Big|_{k(t)=k^*} =& -[sf'(k^*) - (g+n+\delta)] \\ =&(g+n+\delta) - sf'(k^*) \\ =&(g+n+\delta) - (g+n+\delta)\frac{k^*f'(k^*)}{f(k^*)} \\ =&(1-\alpha_K(k^*))(g+n+\delta)\end{aligned}$$

where the elasticity of output with respect to capital, $alpha_K(k)$, is defined as

$$\alpha_K(k) = \frac{k'(k)}{f(k)}.$$

steady_state
Steady state value of capital stock (per unit effective labor).

Getter Return the current steady state value.

Type float

Notes

The steady state value of capital stock (per unit effective labor), *k*, is defined as the value of *k* that solves

$$0 = sf(k) - (g+n+\delta)k$$

where *s* is the savings rate, *f(k)* is intensive output, and $g+n+\delta$ is the effective depreciation rate.

Module contents

models directory imports

objects imported here will live in the *solowpy* namespace

class `solowpy.Model`(*output*, *params*)
Bases: `object`

Attributes

effective_depreciation_rate	Effective depreciation rate for capital stock (per unit effective labor).
intensive_output	Symbolic expression for the intensive form of aggregate production.
ivp	Initial value problem
k_dot	Symbolic expression for the equation of motion for capital (per unit effective labor).
marginal_product_capital	Symbolic expression for the marginal product of capital (per unit effective labor).
output	Symbolic expression for the aggregate production function.
params	Dictionary of model parameters.
solow_residual	Symbolic expression for the Solow residual which is used as a measure of technology.
speed_of_convergence	The speed of convergence for the Solow model.
steady_state	Steady state value of capital stock (per unit effective labor).

Methods

evaluate_actual_investment(k)	Return the amount of output (per unit of effective labor) invested in the productio
evaluate_consumption(k)	Return the amount of consumption (per unit of effective labor).
evaluate_effective_depreciation(k)	Return amount of Capital stock (per unit of effective labor) that depreciaties due t
evaluate_intensive_output(k)	Return the amount of output (per unit of effective labor).
evaluate_k_dot(k)	Return time derivative of capital stock (per unit of effective labor).
evaluate_mpk(k)	Return marginal product of capital stock (per unit of effective labor).
evaluate_output_elasticity(k)	Return elasticity of output with respect to capital stock (per unit effective labor).
evaluate_solow_residual(Y, K, L)	Return Solow residual.
find_steady_state(a, b[, method])	Compute the equilibrium value of capital stock (per unit effective labor).
linearized_solution(t, k0)	Compute the linearized solution for the Solow model.
plot_factor_shares(ax[, Nk])	Plot income/output shares of capital and labor inputs to production.
plot_intensive_investment(ax[, Nk])	Plot actual investment (per unit effective labor) and effective depreciation.
plot_intensive_output(ax[, Nk])	Plot intensive form of the aggregate production function.
plot_phase_diagram(ax[, Nk])	Plot the model's phase diagram.
plot_solow_diagram(ax[, Nk])	Plot the classic Solow diagram.

effective_depreciation_rate

Effective depreciation rate for capital stock (per unit effective labor).

Getter Return the current effective depreciation rate.

Type float

Notes

The effective depreciation rate of physical capital takes into account both technological progress and population growth, as well as physical depreciation.

evaluate_actual_investment (*k*)

Return the amount of output (per unit of effective labor) invested in the production of new capital.

Parameters **k** : array_like (float)

Capital stock (per unit of effective labor)

Returns **actual_inv** : array_like (float)

Investment (per unit of effective labor)

evaluate_consumption (*k*)

Return the amount of consumption (per unit of effective labor).

Parameters **k** : *numpy.ndarray* (float)

Capital stock (per unit of effective labor)

Returns **c** : *numpy.ndarray* (float)

Consumption (per unit of effective labor)

evaluate_effective_depreciation (*k*)

Return amount of Capital stock (per unit of effective labor) that depreciaties due to technological progress, population growth, and physical depreciation.

Parameters **k** : array_like (float)

Capital stock (per unit of effective labor)

Returns **effective_depreciation** : array_like (float)

Amount of depreciated Capital stock (per unit of effective labor)

evaluate_intensive_output(*k*)

Return the amount of output (per unit of effective labor).

Parameters **k** : *numpy.ndarray* (float)

Capital stock (per unit of effective labor)

Returns **y** : *numpy.ndarray* (float)

Output (per unit of effective labor)

evaluate_k_dot(*k*)

Return time derivative of capital stock (per unit of effective labor).

Parameters **k** : *numpy.ndarray* (float)

Capital stock (per unit of effective labor)

Returns **k_dot** : *numpy.ndarray* (float)

Time derivative of capital stock (per unit of effective labor).

evaluate_mpk(*k*)

Return marginal product of capital stock (per unit of effective labor).

Parameters **k** : *numpy.ndarray* (float)

Capital stock (per unit of effective labor)

Returns **mpk** : *numpy.ndarray* (float)

Marginal product of capital stock (per unit of effective labor).

evaluate_output_elasticity(*k*)

Return elasticity of output with respect to capital stock (per unit effective labor).

Parameters **k** : array_like (float)

Capital stock (per unit of effective labor)

Returns **alpha_k** : array_like (float)

Elasticity of output with respect to capital stock (per unit effective labor).

Notes

Under the additional assumption that markets are perfectly competitive, the elasticity of output with respect to capital stock is equivalent to capital's share of income. Since, under perfect competition, firms earn zero profits it must be true capital's share and labor's share must sum to one.

evaluate_solow_residual(*Y, K, L*)

Return Solow residual.

Parameters **k** : array_like (float)

Capital stock (per unit of effective labor)

Returns **residual** : array_like (float)

Solow residual

find_steady_state (*a*, *b*, *method='brentq'*, ***kwargs*)
Compute the equilibrium value of capital stock (per unit effective labor).

Parameters **a** : float

One end of the bracketing interval [a,b].

b : float

The other end of the bracketing interval [a,b]

method : str (default='brentq')

Method to use when computing the steady state. Supported methods are *bisect*, *brenth*, *brentq*, *ridder*. See *scipy.optimize* for more details (including references).

kwargs : optional

Additional keyword arguments. Keyword arguments are method specific see *scipy.optimize* for details.

Returns **x0** : float

Zero of f between a and b.

r : RootResults (present if `full_output = True`)

Object containing information about the convergence. In particular, `r.converged` is True if the routine converged.

intensive_output
Symbolic expression for the intensive form of aggregate production.

Getter Return the current intensive production function.

Type sympy.Basic

Notes

The assumption of constant returns to scale allows us to work the the intensive form of the aggregate production function, F. Defining $c = 1/AL$ one can write

$$F\left(\frac{K}{AL}, 1\right) = \frac{1}{AL}F(A, K, L)$$

Defining $k = K/AL$ and $y = Y/AL$ to be capital per unit effective labor and output per unit effective labor, respectively, the intensive form of the production function can be written as

$$y = f(k).$$

Additional assumptions are that f satisfies $f(0) = 0$, is concave (i.e., $f'(k) > 0, f''(k) < 0$), and satisfies the Inada conditions:

$$\lim_{k \to 0} = \infty$$
$$\lim_{k \to \infty} = 0$$

The [inada1964] conditions are sufficient (but not necessary!) to ensure that the time path of capital per effective worker does not explode.

ivp
Initial value problem

Getter Return instance of the ivp.IVP class representing the model.

Type ivp.IVP

Notes

The Solow model with can be formulated as an initial value problem (IVP) as follows.

$$\dot{k}(t) = sf(k(t)) - (g + n + \delta)k(t),\ t \geq t_0,\ k(t_0) = k_0$$

The solution to this IVP is a function $k(t)$ describing the time path of capital stock (per unit effective labor).

k_dot

Symbolic expression for the equation of motion for capital (per unit effective labor).

Getter Return the current equation of motion for capital.

Type sympy.Basic

Notes

Because the economy is growing over time due to technological progress, g, and population growth, n, it makes sense to focus on the capital stock per unit effective labor, k, rather than aggregate physical capital, K. Since, by definition, $k = K/AL$, we can apply the chain rule to the time derative of k.

$$\begin{aligned}\dot{k}(t) =& \frac{\dot{K}(t)}{A(t)L(t)} - \frac{K(t)}{[A(t)L(t)]^2}\left[\dot{A}(t)L(t) + \dot{L}(t)A(t)\right] \\ =& \frac{\dot{K}(t)}{A(t)L(t)} - \left(\frac{\dot{A}(t)}{A(t)} + \frac{\dot{L}(t)}{L(t)}\right)\frac{K(t)}{A(t)L(t)}\end{aligned}$$

By definition, $k = K/AL$, and by assumption $\dot{A}/A$ and $\dot{L}/L$ are g and n respectively. Aggregate capital stock evolves according to

$$\dot{K}(t) = sF(K(t), A(t)L(t)) - \delta K(t).$$

Substituting these facts into the above equation yields the equation of motion for capital stock (per unit effective labor).

$$\begin{aligned}\dot{k}(t) =& \frac{sF(K(t), A(t)L(t)) - \delta K(t)}{A(t)L(t)} - (g + n)k(t) \\ =& \frac{sY(t)}{A(t)L(t)} - (g + n + \delta)k(t) \\ =& sf(k(t)) - (g + n + \delta)k(t)\end{aligned}$$

linearized_solution (*t*, *k0*)

Compute the linearized solution for the Solow model.

Parameters **t** : *numpy.ndarray* (shape=(T,))

Array of points at which the solution is desired.

k0 : float

Initial condition for capital stock (per unit of effective labor)

Returns **linearized_traj** : *numpy.ndarray* (shape=t.size, 2)

Array representing the linearized solution trajectory.

marginal_product_capital

Symbolic expression for the marginal product of capital (per unit effective labor).

Getter Return the current marginal product of capital.

Type sympy.Basic

Notes

The marginal product of capital is defined as follows:

$$\frac{\partial F(K, AL)}{\partial K} \equiv f'(k)$$

where $k = K/AL$ is capital stock (per unit effective labor).

output

Symbolic expression for the aggregate production function.

Getter Return the current aggregate production function.

Setter Set a new aggregate production function

Type sympy.Basic

Notes

At each point in time the economy has some amounts of capital, *K*, labor, *L*, and knowledge (or technology), *A*, that can be combined to produce output, *Y*, according to some function, *F*.

$$Y(t) = F(K(t), A(t)L(t))$$

where *t* denotes time. Note that *A* and *L* are assumed to enter multiplicatively. Typically *A(t)L(t)* denotes "effective labor", and technology that enters in this fashion is known as labor-augmenting or "Harrod neutral."

A key assumption of the model is that the function *F* exhibits constant returns to scale in capital and labor inputs. Specifically,

$$F(cK(t), cA(t)L(t)) = cF(K(t), A(t)L(t)) = cY(t)$$

for any $c \geq 0$.

params

Dictionary of model parameters.

Getter Return the current dictionary of model parameters.

Setter Set a new dictionary of model parameters.

Type dict

Notes

The following parameters are required:

A0: float Initial level of technology. Must satisfy $A_0 > 0$.

L0: float Initial amount of available labor. Must satisfy $L_0 > 0$.

g [float] Growth rate of technology.

n [float] Growth rate of the labor force.

s [float] Savings rate. Must satisfy *0 < s < 1*.

delta [float] Depreciation rate of physical capital. Must satisfy $0 < \delta$.

Although no restrictions are placed on the rates of technological progress and population growth, the sum of g, n, and δ is assumed to be positive. The user mus also specify any additional model parameters specific to the chosen aggregate production function.

plot_factor_shares (*ax*, *Nk=1000.0*, ***new_params*)
Plot income/output shares of capital and labor inputs to production.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

capitals_share_line : maplotlib.lines.Line2D

A Line2D object representing the time path for capital's share of income.

labors_share_line : maplotlib.lines.Line2D

A Line2D object representing the time path for labor's share of income.

plot_intensive_investment (*ax*, *Nk=1000.0*, ***new_params*)
Plot actual investment (per unit effective labor) and effective depreciation. The steady state value of capital stock (per unit effective labor) balance acual investment and effective depreciation.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

actual_investment_line : maplotlib.lines.Line2D

A Line2D object representing the level of actual investment as a function of capital stock (per unit effective labor).

breakeven_investment_line : maplotlib.lines.Line2D

A Line2D object representing the "break-even" level of investment as a function of capital stock (per unit effective labor).

ss_line : maplotlib.lines.Line2D

A Line2D object representing the steady state level of investment.

plot_intensive_output (*ax*, *Nk=1000.0*, ***new_params*)
Plot intensive form of the aggregate production function.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

intensive_output : maplotlib.lines.Line2D

A Line2D object representing intensive output as a function of capital stock (per unit effective labor).

plot_phase_diagram(*ax*, *Nk=1000.0*, ***new_params*)
Plot the model's phase diagram.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

k_dot_line : maplotlib.lines.Line2D

A Line2D object representing the rate of change of capital stock (per unit effective labor) as a function of its level.

origin_line : maplotlib.lines.Line2D

A Line2D object representing the origin (i.e., locus of points where k_dot is zero).

ss_line : maplotlib.lines.Line2D

A Line2D object representing the steady state level of capital stock (per unit effective labor).

plot_solow_diagram(*ax*, *Nk=1000.0*, ***new_params*)
Plot the classic Solow diagram.

Parameters **ax** : *matplotlib.axes.AxesSubplot*

An instance of *matplotlib.axes.AxesSubplot*.

Nk : float (default=1e3)

Number of capital stock (per unit of effective labor) grid points.

new_params : dict (optional)

Optional dictionary of parameter values to change.

Returns A list containing...

actual_investment_line : maplotlib.lines.Line2D

A Line2D object representing the level of actual investment as a function of capital stock (per unit effective labor).

breakeven_investment_line : maplotlib.lines.Line2D

A Line2D object representing the "break-even" level of investment as a function of capital stock (per unit effective labor).

ss_line : maplotlib.lines.Line2D

A Line2D object representing the steady state level of investment.

solow_residual

Symbolic expression for the Solow residual which is used as a measure of technology.

Getter Return the symbolic expression.

Type sympy.Basic

speed_of_convergence

The speed of convergence for the Solow model.

Getter Return the current speed of convergence.

Type float

Notes

The following is a derivation for the speed of convergence λ:

$$\begin{aligned}\lambda \equiv -\frac{\partial \dot{k}(k(t))}{\partial k(t)}\Bigg|_{k(t)=k^*} =& -[sf'(k^*) - (g+n+\delta)] \\ =&(g+n+\delta) - sf'(k^*) \\ =&(g+n+\delta) - (g+n+\delta)\frac{k^* f'(k^*)}{f(k^*)} \\ =&(1-\alpha_K(k^*))(g+n+\delta)\end{aligned}$$

where the elasticity of output with respect to capital, $alpha_K(k)$, is defined as

$$\alpha_K(k) = \frac{k'(k)}{f(k)}.$$

steady_state

Steady state value of capital stock (per unit effective labor).

Getter Return the current steady state value.

Type float

Notes

The steady state value of capital stock (per unit effective labor), *k*, is defined as the value of *k* that solves

$$0 = sf(k) - (g+n+\delta)k$$

where *s* is the savings rate, *f(k)* is intensive output, and $g+n+\delta$ is the effective depreciation rate.

class solowpy.**CobbDouglasModel**(*params*)

Bases: *solowpy.model.Model*

Attributes

`effective_depreciation_rate`	Effective depreciation rate for capital stock (per unit effective labor).
`intensive_output`	Symbolic expression for the intensive form of aggregate production.
`ivp`	Initial value problem
`k_dot`	Symbolic expression for the equation of motion for capital (per unit effective labor).
`marginal_product_capital`	Symbolic expression for the marginal product of capital (per unit effective labor).
`output`	Symbolic expression for the aggregate production function.
`params`	Dictionary of model parameters.
`solow_residual`	Symbolic expression for the Solow residual which is used as a measure of technology.
`speed_of_convergence`	The speed of convergence for the Solow model.
`steady_state`	Steady state value of capital stock (per unit effective labor).

Methods

`analytic_solution`(t, k0)	Compute the analytic solution for the Solow model with Cobb-Douglas productio
`evaluate_actual_investment`(k)	Return the amount of output (per unit of effective labor) invested in the productio
`evaluate_consumption`(k)	Return the amount of consumption (per unit of effective labor).
`evaluate_effective_depreciation`(k)	Return amount of Capital stock (per unit of effective labor) that depreciaties due t
`evaluate_intensive_output`(k)	Return the amount of output (per unit of effective labor).
`evaluate_k_dot`(k)	Return time derivative of capital stock (per unit of effective labor).
`evaluate_mpk`(k)	Return marginal product of capital stock (per unit of effective labor).
`evaluate_output_elasticity`(k)	Return elasticity of output with respect to capital stock (per unit effective labor).
`evaluate_solow_residual`(Y, K, L)	Return Solow residual.
`find_steady_state`(a, b[, method])	Compute the equilibrium value of capital stock (per unit effective labor).
`linearized_solution`(t, k0)	Compute the linearized solution for the Solow model.
`plot_factor_shares`(ax[, Nk])	Plot income/output shares of capital and labor inputs to production.
`plot_intensive_investment`(ax[, Nk])	Plot actual investment (per unit effective labor) and effective depreciation.
`plot_intensive_output`(ax[, Nk])	Plot intensive form of the aggregate production function.
`plot_phase_diagram`(ax[, Nk])	Plot the model's phase diagram.
`plot_solow_diagram`(ax[, Nk])	Plot the classic Solow diagram.

`analytic_solution`(*t*, *k0*)

Compute the analytic solution for the Solow model with Cobb-Douglas production technology.

Parameters **t** : *numpy.ndarray*

Array of points at which the solution is desired.

k0 : (float)

Initial condition for capital stock (per unit of effective labor)

Returns **analytic_traj** : *numpy.ndarray* (shape=t.size, 2)

Array representing the analytic solution trajectory.

`steady_state`

Steady state value of capital stock (per unit effective labor).

Getter Return the current steady state value.

Type float

Notes

The steady state value of capital stock (per unit effective labor) with Cobb-Douglas production is defined as

$$k^* = \left(\frac{s}{g+n+\delta}\right)^{\frac{1}{1-\alpha}}$$

where s is the savings rate, $g+n+\delta$ is the effective depreciation rate, and α is the elasticity of output with respect to capital (i.e., capital's share).

class `solowpy.`**`CESModel`**(*params*)

Bases: *`solowpy.model.Model`*

Attributes

`effective_depreciation_rate`	Effective depreciation rate for capital stock (per unit effective labor).
`intensive_output`	Symbolic expression for the intensive form of aggregate production.
`ivp`	Initial value problem
`k_dot`	Symbolic expression for the equation of motion for capital (per unit effective labor).
`marginal_product_capital`	Symbolic expression for the marginal product of capital (per unit effective labor).
`output`	Symbolic expression for the aggregate production function.
`params`	Dictionary of model parameters.
`solow_residual`	Symbolic expression for the Solow residual which is used as a measure of technology.
`speed_of_convergence`	The speed of convergence for the Solow model.
`steady_state`	Steady state value of capital stock (per unit effective labor).

Methods

`evaluate_actual_investment`(k)	Return the amount of output (per unit of effective labor) invested in the productio
`evaluate_consumption`(k)	Return the amount of consumption (per unit of effective labor).
`evaluate_effective_depreciation`(k)	Return amount of Capital stock (per unit of effective labor) that depreciaties due t
`evaluate_intensive_output`(k)	Return the amount of output (per unit of effective labor).
`evaluate_k_dot`(k)	Return time derivative of capital stock (per unit of effective labor).
`evaluate_mpk`(k)	Return marginal product of capital stock (per unit of effective labor).
`evaluate_output_elasticity`(k)	Return elasticity of output with respect to capital stock (per unit effective labor).
`evaluate_solow_residual`(Y, K, L)	Return Solow residual.
`find_steady_state`(a, b[, method])	Compute the equilibrium value of capital stock (per unit effective labor).
`linearized_solution`(t, k0)	Compute the linearized solution for the Solow model.
`plot_factor_shares`(ax[, Nk])	Plot income/output shares of capital and labor inputs to production.
`plot_intensive_investment`(ax[, Nk])	Plot actual investment (per unit effective labor) and effective depreciation.
`plot_intensive_output`(ax[, Nk])	Plot intensive form of the aggregate production function.
`plot_phase_diagram`(ax[, Nk])	Plot the model's phase diagram.
`plot_solow_diagram`(ax[, Nk])	Plot the classic Solow diagram.

`solow_residual`

Symbolic expression for the Solow residual which is used as a measure of technology.

Getter Return the symbolic expression.

Type sym.Basic

steady_state

Steady state value of capital stock (per unit effective labor).

Getter Return the current steady state value.

Type float

Notes

The steady state value of capital stock (per unit effective labor) with CES production is defined as

$$k^* = \left[\frac{1-\alpha}{\left(\frac{g+n+\delta}{s}\right)^{\rho} - \alpha}\right]^{\frac{1}{\rho}}$$

where s is the savings rate, $g + n + \delta$ is the effective depreciation rate, and α controls the importance of capital stock relative to effective labor in the production of output. Finally,

$$\rho = \frac{\sigma - 1}{\sigma}$$

where σ is the elasticity of substitution between capital and effective labor in production.

Indices and tables

- genindex
- modindex
- search

Bibliography

[romer2011] 4. Romer. *Advanced Macroeconomics, 4th edition*, MacGraw Hill, 2011.

[solow1956] 18. Solow. *A contribution to the theory of economic growth*, Quarterly Journal of Economics, 70(1):64-95, 1956.

[inada1964] 11. Inda. *Some structural characteristics of Turnpike Theorems*, Review of Economic Studies, 31(1):43-58, 1964.

[inada1964] 11. Inda. *Some structural characteristics of Turnpike Theorems*, Review of Economic Studies, 31(1):43-58, 1964.

S

A

C

E

F

I

K

L

M

N

O

P

S

T