An Economic Policy Simulator of Stock-Flow Consistent Models

*

Yaku Fernandez
dept. HEC Lausanne
University of Lausanne
Lausanne, Switzerland
yaku.fernandezlanda@unil.ch

Frank Aquinga Melo
HEC - Department of Finance
University of Lausanne
Lausanne, Switzerland
Frank.AlquingaMelo@unil.ch

***Abstract*—After the great recession, the Stock-Flow Consistent approach (SFC) has been increasingly used as an alternative to the Dynamic Stochastic General Equilibrium models (DSGE). Our project consists in creating a program that serves as a step-by-step guide for non-economists, allowing to visualize and get a quick, accurate and simple approximation of simulated exogenous shocks in SFC models. To do that, we implement our program in the Python programming language taking the equations and intuition described on Godley Lavoie (2007) specifically, chapters 3 and 4. Explicitly, the two mean models we replicate are called Simple with Expectations (SIMEX) and Portfolio Choice with Expectations (PCEX). The types of simulations that our application can achieve include fiscal policy (change in public expending), monetary policy (change in interest rate) and agents behaviour parameters (change for instance in marginal propensity to consume). Our main result consists in a program that enables to simulate external shocks in a friendly environment. Moreover, we show that Python can be used as a complement for educational purposes.**

***Index Terms*—SFC, DSGE, SIMEX, PCEX**

I. Introduction

In spite of Economics as a social science includes a plenty of different theories and approaches, it is commonly accepted that a specific orientation governs the teaching of this discipline. The so called neoclassical economics is pretty used as the cornerstone of the modern economic analysis. Nevertheless there exists a huge diversity of theories trying to overcome some problematic aspects that the neoclassical approach cannot explain. Since the last financial crisis in 2008 one of these heterodox approaches has gained ground: the Stock-Flow Consistent (SFC) approach.

The understanding of non-meanstream theories becomes difficult since there is not enough sources or teaching materials. The vast majority of textbooks in economics are written only taking into consideration neoclassical theories, ignoring the plural nature of economics. For that, our objective is to create an application that enables the user to simulate external shocks in a SFC economy. There is a large potential positive effect as this program makes possible an easy comprehension of the mean features of SFC models. The target group is anyone who wants to learn about new economic theories without a specific background in Economics as well as undergraduate students.

The two models on which we are going to implement our application are the Simple with Expectations (SIMEX) and Portfolio Choice with Expectations (PCEX). These models will permit to do simulations on fiscal policy (changes in Government expenditure), monetary policy (changes in interest rate) and agents behaviour parameters.

Our simulator is going to permit to estimate the impact of several exogenous shocks, even an hypothetical scenario where all exogenous variables have changed. Likewise, the user will be able to establish in what period the shock happens. The outputs that the program is going to provide are a graph with the evolution of selected variables (before and after the shock), a brief text with the description of what have occurred in the economy when the specific shock has been introduced and a summary table showing the values of the old and new steady states for the main endogenous variables.

In order to implement our application we are going to use a well known programming language : Python. In the following sections we explain the mean features of the SFC approach (section II), then, the methodology of our program is showed. The details of the implementation is explained file-by-file in section IV. Section V considers the practicability of maintain and update the code. Section VI details the results of our project, showing the interface and the outputs of our program. Finally, we include a brief conclusion.

II. Literature

A. Research Question

The aim of this project is to develop and implement an economic simulator of exogenous shocks. As this program is for any student or person eager to get a first approximation on heterodox economic theories, the challenge is to deliver an easy-to-use application, highlighting the main features of the economic models and how the exogenous shocks affect the final state of the simulated economies variables such as total income (GDP), taxes, wealth of households and so on. This first version of our program is going to include two SFC models following the Godley and Lavoie's book. Future version of the simulator would add and show the dynamics of another SFC model or a different type of economic theories.

B. Relevant Literature and Model Approach

The SFC approach provides a framework to analyse jointly the two sides of the economy: the real sector and the financial sector. The analysis is carry out by the use of a specific accounting method where the principle of everything comes from somewhere and everything goes somewhere is strictly followed [1]. Modeling a given economy requires a full integration of the balance sheet and the transactions flow of sectors or agents. The basic sectors in SFC models are households, firms and government.

In Godley and Lavoie (2007) the chapters 3 and 4 describe the fundamentals of SIMEX and PCEX models respectively. The following table shows the stocks of assets and liabilities for each sector and their logical interaction among agents in a SIMEX model. As we can see all columns and rows sum zero. That implies the economy is consistent in the sense that there is not loose ends: everything in the economy effectively goes somewhere including financial assets.

	1. Households	2. Production	3. Government	Σ
1. Consumption	$-C_d$	$+C_s$		0
2. Govt. expenditures		$+G_s$	$-G_d$	0
3. [Output]		$[Y]$		
4. Factor income (wages)	$+W \cdot N_s$	$-W \cdot N_d$		0
5. Taxes	$-T_s$		$+T_d$	0
6. Change in the stock of money	$-\Delta H_h$		$+\Delta H_s$	0
Σ	0	0	0	0

Fig. 1. Behavioural transactions matrix.

The National Income is represented in the first five rows: consumption (C), government expenditures (G), factor income (W=wages) and taxes (T). The line of Change in the stock provide a description of the stocks movement. In the simplest model of SIMEX there exists just one financial asset, i.e. money bills. Next to each cell on the matrix there is a positive or negative sign, that indicates the direction of the flow. For instance, households buy goods by spending in consumption (negative sign), that consumption goes to production sector as an income (positive sign) for firms in this economy. The same occurs for Government: this sector spends G (minus sign) and gets tax revenues (positive sign). The importance of creating this kind of matrix relies on the fact that any internal transaction and, consequently, any inter-sector relationship is employed to get a precise picture of the economy and the agents behavior. A well built behavioural transaction matrix enables constructing the models equations (See appendix).

In the SIMEX model, as we said before, there is only one stock, money bills. The exogenous variables are the variables that would create shocks in this economy and they include the government expenditure, the level of taxes, the propensity to consume of households either from disposable income and from assets (money). The endogenous variables include national income (Y), the evolution of consumption, tax revenues, stock flows and stock level.

On the other hand, we develop the same simulator structure for a PCEX model. Here the economy has an additional exogenous variable, i.e. the Central Banks interest rate. This new variable introduces a more realistic environment to our simulation because households now has to decide on how much of money and short-term bills they hold. Money do not gain interests, but short term bills, that are issued by the government, pay interest at a given rate. Now in PCEX model the wealth or stock is represented by the sum of money and bills.

Furthermore, PCEX model introduce a random component in the sense that households cannot perfectly forecast their future disposable income. Because of that, the evolution towards the steady state of the model does not goes smoothly. This last feature allow us estimating a model closer to the real-world economy than the first version of the SFC we have taken. With the PCEX model we can simulate fiscal and monetary policies by changing the level of government expenditure (the exogenous variable G) or by changing the level of interest rate (exogenous variable r). Consistently with the economic intuition a raise in interest rate creates a short-term decline of the total income, whereas a lower interest rate create a temporary boom.

III. Methodology

In order to begin with the construction of our application, we decided to divide the work into three mutually complementary phases with an specific purpose. These phases are the following:

1) Models.
   It constitutes the first stage in our project. The aim of this part is to implement the construction of the two models, with their respective equations and numerical technique to obtain simulations results.

2) Complements.
   Once the models phase finished, we develop complementary functions to show the results of each model. These results are a graph describing the evolution of the variables, sentences that express in words what was the effect of each shock selected by the user and a summary table with the steady states values and final impacts.

3) Interface.
   The final stage in our project contemplates the construction of an interface capable of integrating both two first phases and interacting with the user by taking the different user's inputs with the corresponding simulations. The main purpose of the interface is to provide an easy and intuitively manipulation of our application.

In order to develop our application and having in mind the objective of keeping a well-organized work, we decided to separate the program into several file scripts permitting to improve the code comprehension. That is largely facilitated as each script has a particular area of work and a specific purpose inside each work phase. These scripts will be imported by the *main.py* script (interface) which will make the scripts work together.

The following part details the area of work and the purpose of each file script inside every work phase.

A. Models

We create an individual file script to each model. Both of them follows the same basic structure but they differ in the sense that each model include its particular equations and variables. The models files could be seen as the engine of our program. That means inside these files there is the numerical procedure to simulate the shocks and the evolution of the economy. Each file contains a function where the exogenous and endogenous variables of each model are defined, as well as the main loop to obtain results of the simulations. Not only the variables evolution but also the final steady state produced by the shocks.

B. Complements

Display

This file script enables to visualize the simulations results. It contains two functions *display* and *display_with_without_shock* which permit to plot the evolution of the variables without any shock and with a shock respectively. It is important to mention that the final output will be dependent on the users inputs.

Explanation

This file script focus into give detailed text information in all the simulations about the steady state of the variables and the final impact of any shock if this exists. In order to do that, two functions were created *dic_steady_states* and *explain* which give an output adapted in function of the users inputs.

C. Interface

Main

This is the principal file script which enables the execution of the entire application. Thus, this file imports all the precedent scripts and make them work together.

In order to show the simulation results of our two models and with the objective to keep the utilization of our program easy to understand, we decided to implement in the *main* file script an interface which would permit the user to navigate through our program and to play with the variables of the models to simulate different scenarios in the economy.

The aim of this interface was to minimize the exposure to the programming code and at the same time allowing the user to experience all types of numerical simulations.

In this way, we developed an interface inspired by the environment of a web page. That would permit the user to easily discover all the functionalities of the application. Having this vision in mind, we created several windows pages which present the models and show the required steps to complete the simulations. The process to link each windows page to each other in order to construct the web environment is assured by buttons methods.

The interface presents the following organization:

Home Page. It is a welcoming page which will present the app and will show to the user the firsts options such as the choice of the model to simulate (SIMEX/PCEX pages).

PCEX/SIMEX pages. Once the model is chosen from the Home Page another windows will show up where there will be the possibility to select the type of simulation to undertake (baseline/shock simulation).

Baseline page. This window will permit the user to modify the input variables value of the model chosen and to select the variables to examine graphically. Once the inputs are given by the user and the variables to analyze are chosen the user will be able to visualize the information by clicking on a button which will show a plot of the selected variables evolution over time. Besides, an additional numerical information about the steady states of the variables chosen as well as a summarize table of the steady state of all variables will be given.

Shock page: This page will contain the same functionalities as the Baseline page but at the same time, it will permit the possibility to introduce a shock in the economy. Thus, the user will be able to choose over which variable the shock will be settled, the amount and the year in which the shock will affect the economy. As the Baseline page there will be the possibility to visualize the variables evolution as well as the information of the steady states. Moreover, an additional numerical information will be given about the numerical impact of the shock on the variables.

About page This page will be accessible from the home page and it will contain additional relevant information for the user.

In any page, the user will always have the possibility of going back to the Home Page and from this page the option to quit the program (GoodBye page).

In order to have a simplified view of the entire interface we made a summarizing diagram. This one permit us to quickly visualize the structure organization as well as the different windows composing the interface (Fig. 2).

IV. Implementation of the algorithm

At this stage is time to talk about how the algorithm was implemented. As said before, we organized our work into three phases. Because of that we are going to discuss the implementation of the code in each script following the phases organization.

A. Models

PCEX.py and *SIMEX.py*

First of all and before to define the function for each model the code imports three needed packages, to name, numpy, pandas and random. This last one package is used only in the *PCEX.py* file script because here we include forecasting error of agents. The numpy package is used for creating endogenous variables before running the loop. Basically we create vectors containing zeros with a length of T, that is the number of years taken into consideration in the simulation. This T is endogenously defined according to which was the user decision. Pandas package enables to get a Data Frame with the evolution of variables since the period zero to the final period. This step is implemented as a last step in the script and it is critically important for the program due to the fact that enables to display afterward the graph with the variable's evolution.

The main function in each model establish the exogenous values of parameters used, as well as the exogenous values by default. After creating the endogenous variables the function creates two for loop iterations, one for the whole period before the shocks and one for the whole period after the shocks. The interactions try to find out the value for each year or period by reducing the difference between a given initial national income value and a national income that takes into account consumptions. When the loop provides a value with an insignificance discrepancy then the obtained national income for any year is introduced in its respective vector. Since the path of national income is obtained, we can calculate the path for the rest of endogenous variables. An additional part inside the function enables to estimate the final steady states values of the simulation. This step is straightforward as we just need to use exogenous values according to the Godley Lavoie equations.

It is important to mention that for each model the same model function permits us to produce the two type of simulations:

1) Baseline. In which none shocks are introduced in the economy.
2) Shock. In which multiple shocks can be introduced in the economy.

We have implemented this dual-functionality of the model function by putting all the variables required to produce the baseline and shock simulations into the arguments position of the function. In this way, we have to mandatory introduce *initial variables* (arguments) of the baseline and *modified variables* (arguments) of the shock. Then to accomplish this dual-functionality we have developed a trick. This trick consists, on one hand, in setting *modified variables* to the same values as *initial variables*, in order to make the function simulate the baseline scenario. On the other hand, if *modified variables* contain values different from *initial variables*, then the function is going to simulate the shock scenario.

B. Complements

display.py

In this script we import *matplotlib.pyplot* which is a python 2D plotting library that permits to produce publication quality figures. The module *pyplot* provides a MATLAB-like interface which permit us to have full control of line styles. We used this library to generate the different simulation plots. To do that we have created two functions:

- *display*.
  This function shows the simulation results of an economy free of any shock. It takes as arguments a *df* which is dataframe containing all the simulation results of the model (given by the model function) and *variables* which is a list containing the name of the variables. The final output is a plot showing in the same graph the evolution of each variable along time.

- *display_with_without_shock*.
  This function permits to visualize the simulation results when a shock is added to the economy. Thus, additionally to the arguments needed for the *display* function it takes *df_with_shock* which is a dataframe containing the results of the model modified by the shock (given by the model function) and *T_shock* which is an integer showing the time where the shock takes place. This final argument is important because it permits us to put a vertical line showing the year of the shock. The final output is a plot which compares the evolution of the economy with shock and without shock (visualized through choppy lines).

It is important to mention that both functions can be used for the two models.

Explanation.py

In this script we have two functions:

- *dic_steady_states*.
  It takes as argument *variables* which is a list containing the names of the variables and *steady_data* which is a

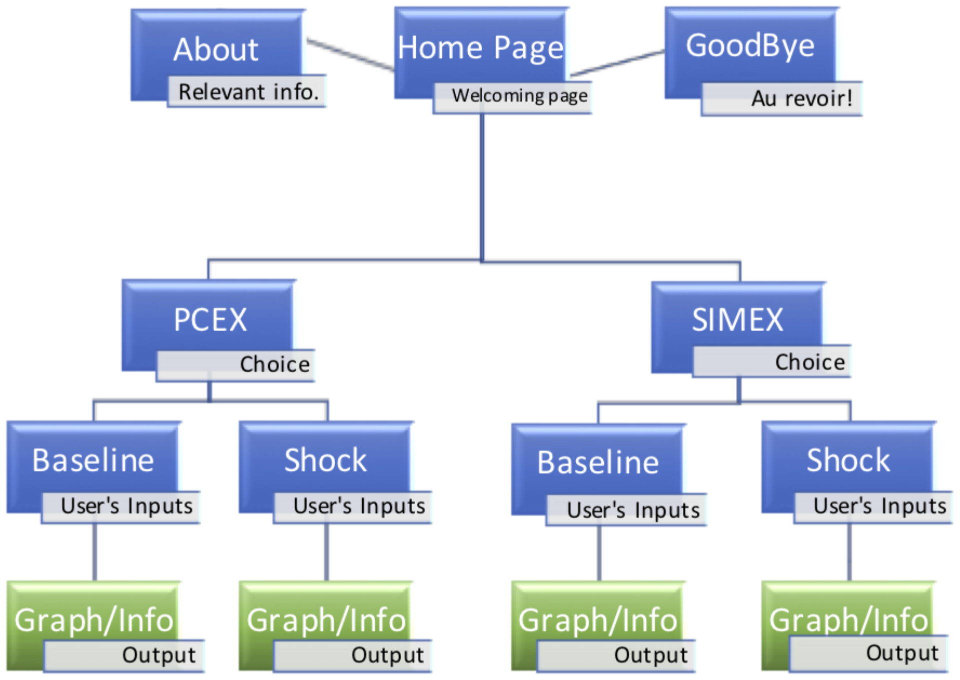


Fig. 2. Diagram representation of the interface.

dataframe (given by the model function) containing the steady states information for each variable. With these two arguments the function takes charge of putting all the information in a dictionary object. The final output is a dictionary having as keys the variables names and as values a list containing the steady states information.

- *explain*.
  A function which takes in charge the formatting of the text message that the user finally sees in the console. For that it takes as arguments *dictionary* (given by the *dic_steady_states* function), *Shock* which is a boolean object indicating whether there is shock or not (it is False by default) and *T_shock* which is an integer indicating the year of the shock (it is 0 by default). Thus, with these three arguments and wit the help of *if* conditions the function is going to be able to give as output an adapted message to the users simulations.

C. Interface

main.py

This script executes the entire application. To do that it imports all the file scripts mentioned before and *tkinter* which is an interface package.

To organize the code we decided to implement classes object. Thus, each interface windows correspond to one class in this script. That implies that the content in each class depends on what we want to show on the windows interface. The working of this interface is made specially through two types of objects:

1) *Checkbuttons, buttons and scale widgets*.
   These are specific objects from the *tkinter* package which can directly be seen in the interface windows. The checkbuttons permit us to implement on-off selections. The buttons permit us to call the functions which are going to execute the action. And the scales which allows the user to select a numerical value by moving a slider knob along a scale. All theses three widgets together make possible that the user can arbitrary modify the variables values in order to simulate different states in the economy.

2) *functions*
   These objects are not directly showed to the user. They have a precise work to do and they are called through the button objects. Thus, for example the *show_frame* function makes a specific window appear (depending on the windows name passed through the button) when a specific button calls him to do it. This call mechanism is triggered when the user presses the button. An specific class (*SeaofBTCapp)* was created which contains all the different functions.

To finish the interface, a style was implemented to differentiate the titles and the corps of the windows. This style was introduced at the beginning of the script.

V. Maintain and Update the Code

The aim of our application is to permit the simulation of two Stock-Flow Consistent models. As these models follow an essential theoretical approach with fixed established equations no updating methods are required in the functions that generate the models. At the same time, the functions produce its own data results, making our application be independent of external data sources. Therefore, we can see a limited need to maintain or update the code base. However, if we want to integrate another model to our application some changes would be needed. In particular, it would be necessary to add a specific code in the interface (main.py script) permitting the user to simulate the new model.

VI. Results

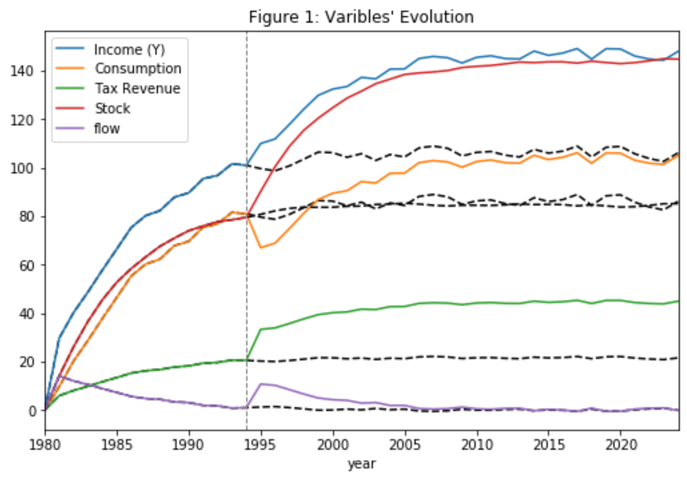
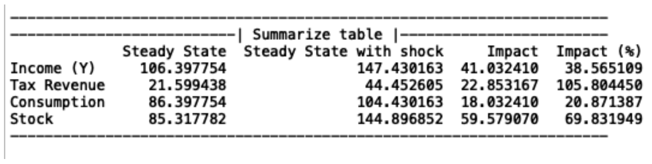
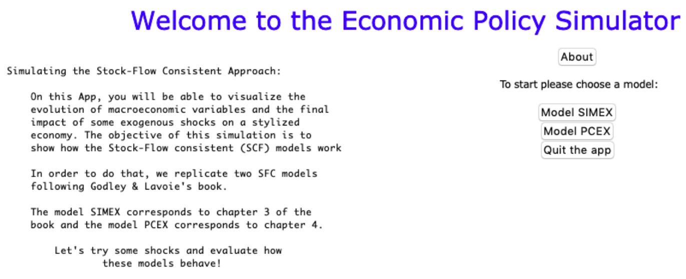


Fig. 3. Graph Output



```
---------------------------------------------------------------------
-------------------------| Summarize table |-------------------------
             Steady State  Steady State with shock    Impact  Impact (%)
Income (Y)     106.397754               147.430163 41.032410   38.565109
Tax Revenue     21.599438                44.452605 22.853167  105.804450
Consumption     86.397754               104.430163 18.032410   20.871387
Stock           85.317782               144.896852 59.579070   69.831949
---------------------------------------------------------------------
```

Fig. 4. Table Output

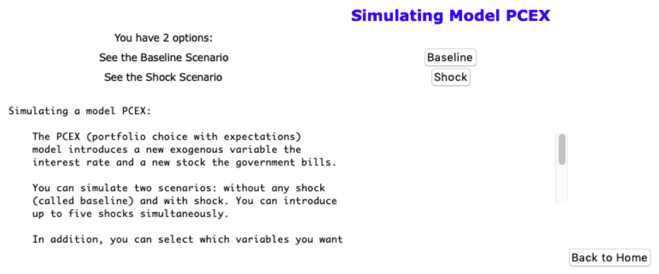


Welcome to the Economic Policy Simulator

```
Simulating the Stock-Flow Consistent Approach:

    On this App, you will be able to visualize the
    evolution of macroeconomic variables and the final
    impact of some exogenous shocks on a stylized
    economy. The objective of this simulation is to
    show how the Stock-Flow consistent (SCF) models work

    In order to do that, we replicate two SFC models
    following Godley & Lavoie's book.

    The model SIMEX corresponds to chapter 3 of the
    book and the model PCEX corresponds to chapter 4.

        Let's try some shocks and evaluate how
                these models behave!
```

Fig. 5. Visualization of the main page

The program we built consists in an user interface that enables to simulate external shocks on an hypothetical SFC-based economy. After coding the program we have obtained

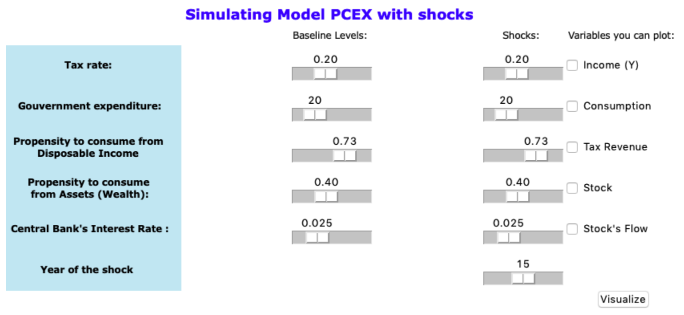


Simulating Model PCEX

You have 2 options:
See the Baseline Scenario
See the Shock Scenario

```
Simulating a model PCEX:

    The PCEX (portfolio choice with expectations)
    model introduces a new exogenous variable the
    interest rate and a new stock the government bills.

    You can simulate two scenarios: without any shock
    (called baseline) and with shock. You can introduce
    up to five shocks simultaneously.

    In addition, you can select which variables you want
```

Fig. 6. Welcome page for PCEX model



Simulating Model PCEX with shocks

	Baseline Levels:	Shocks:	Variables you can plot:
Tax rate:	0.20	0.20	☐ Income (Y)
Gouvernment expenditure:	20	20	☐ Consumption
Propensity to consume from Disposable Income	0.73	0.73	☐ Tax Revenue
Propensity to consume from Assets (Wealth):	0.40	0.40	☐ Stock
Central Bank's Interest Rate :	0.025	0.025	☐ Stock's Flow
Year of the shock		15	

Fig. 7. Page PCEX with shocks

an easy way to get the main features of this kind of models. Figure 5 shows the main page of the interface where on the left hand side the user can read a brief presentation. On the right hand side, the buttons "about", "Model SIMEX", "Model PCEX" and "Quit the App" are placed.

Selecting any of the models leads user to the "model page" (figure 6) with a description of the intuition and logic of the selected model and two simulation's options: the baseline and the shock scenario. Clicking on each of these scenarios opens a new page that looks like Figure 7.

Figure 7 is integrated by four columns, the first one shows the list of exogenous variables that can be changed by the user in order to generate shocks. The last element on this column is the "Year of the shock" that makes sense only under the shock scenario. Second and third columns show elements which corresponds to each line or row from first column.

The second column enables the user chooses the value of the variables for the baseline scenario, whereas the third column allows the same selection for the shock scenario. Finally, the user can choose what are the endogenous variables that he want to see as the output of the simulation. If the user do not select any variable the output he gets as a graph will be an empty figure.

When the user selects the button "visualize" the program start the process of calculate the endogenous variables, once this part is finished the display part takes the values delivered by the SIMEX or PCEX file script and the user can see the impact of the selected shocks. Figure 3 and 4 show an example of a shock' output for a simulation with five simultaneous shocks. Figure 3 plots the evolution of the selected variables as well as the counterfactual evolution (dashed lines) that describes the economy as if there were not shocks. Figure 4 summarizes in a table the steady states values for four different endogenous variables, to name Income, Tax Revenue, Consumption and Stock. The final impact of the shocks is showed in levels (third column) and as a percentage from initial state(fourth column).

By selecting the button ”change scenario” or ”back to home”, the user can estimate a completely new shock (either by a PCEX or SIMEX model) or create a baseline scenario.

VII. Conclusion

We developed an easy-to-use program on Python to simulate external shocks based on the Stock-Flow Consistent approach. This required to plan several steps. After getting a numerical method to estimate the mathematical model, the main challenge was to build the interface of our program. The use of the *tkinter* package was a critical aspect in our project because it allow us to create a friendly appearance for our program. In this project we demonstrated Python could be a powerful tool to develop programs and applications with educational purposes, making possible a better diffusion of heterodox economics approaches. The way the program was built permits to include, in future versions, more economic models to offer additional scenarios with their particular internal dynamics.

References

[1] W. Godley and M. Lavoie, “Monetary Economics: An Integrated Approach to Credit, Money, Income, Production and Wealth ,” London: PALGRAVE MACMILLAN, 2007.

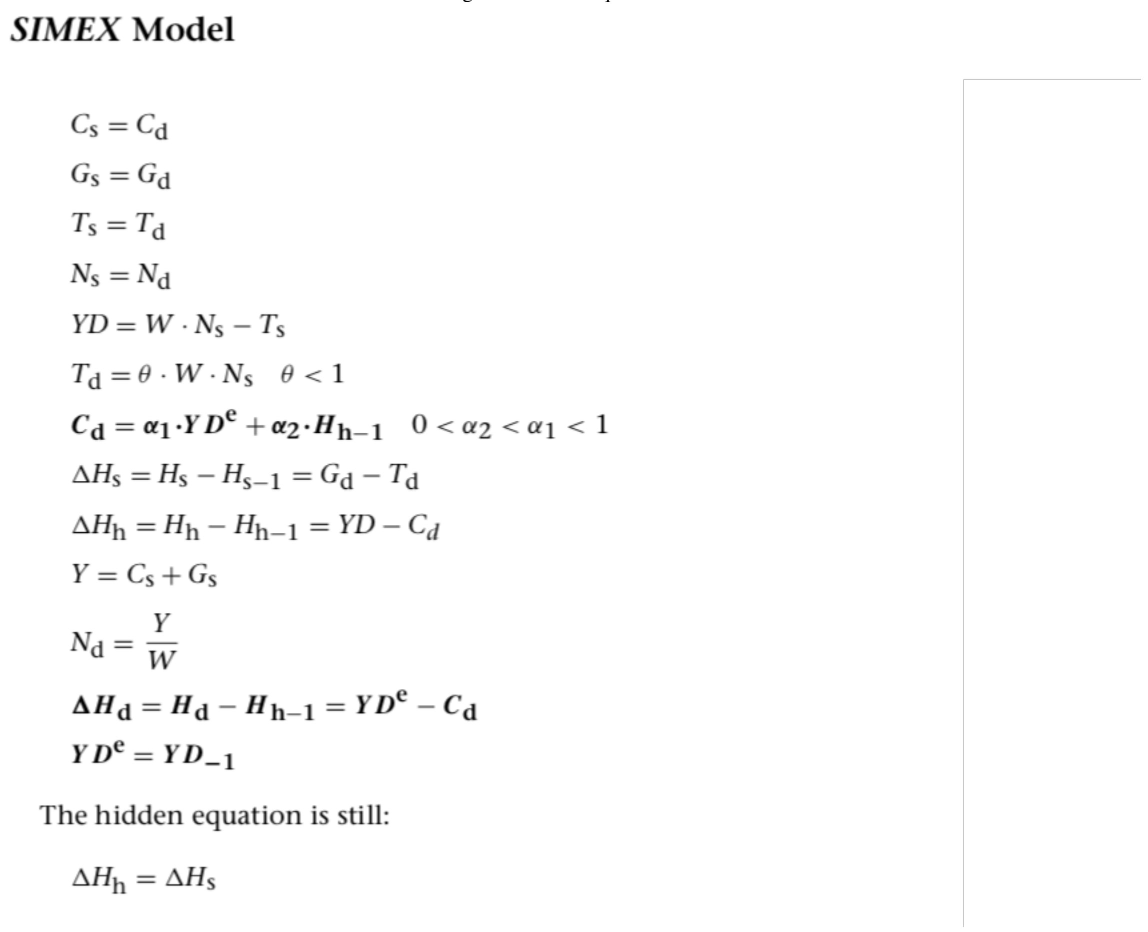
[2] M. Nikiforos G. Zezza, “Stock-flow Consistent Macroeconomic Models: A Survey,” The Levy Institute Working Paper Collection, 2017

VIII. Appendix

A. *SIMEX equations*

B. *PCEX equations*

Fig. 8. SIMEX equations



SIMEX Model

$$C_s = C_d$$

$$G_s = G_d$$

$$T_s = T_d$$

$$N_s = N_d$$

$$YD = W \cdot N_s - T_s$$

$$T_d = \theta \cdot W \cdot N_s \quad \theta < 1$$

$$\boldsymbol{C_d = \alpha_1 \cdot YD^e + \alpha_2 \cdot H_{h-1}} \quad 0 < \alpha_2 < \alpha_1 < 1$$

$$\Delta H_s = H_s - H_{s-1} = G_d - T_d$$

$$\Delta H_h = H_h - H_{h-1} = YD - C_d$$

$$Y = C_s + G_s$$

$$N_d = \frac{Y}{W}$$

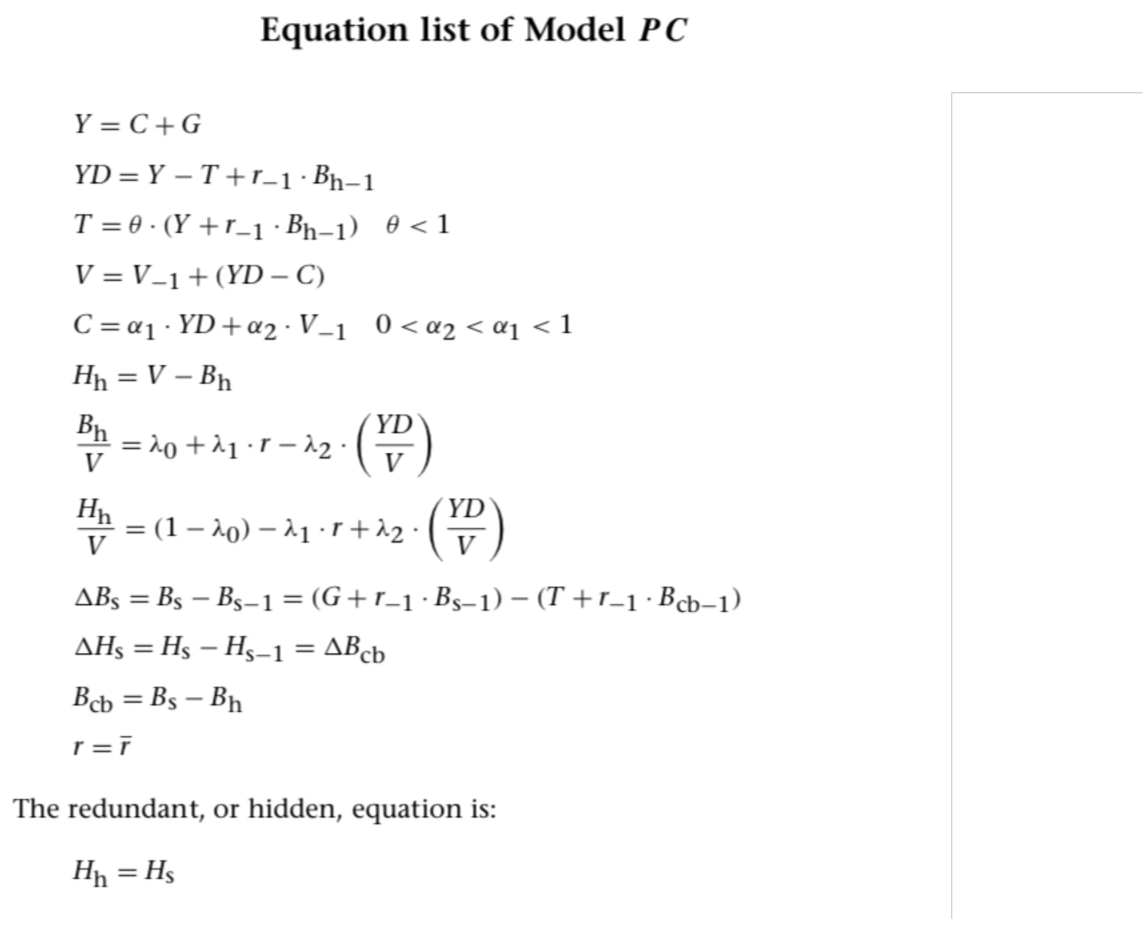
$$\boldsymbol{\Delta H_d = H_d - H_{h-1} = YD^e - C_d}$$

$$\boldsymbol{YD^e = YD_{-1}}$$

The hidden equation is still:

$$\Delta H_h = \Delta H_s$$

Fig. 9. PCEX equations



Equation list of Model *PC*

$$Y = C + G$$

$$YD = Y - T + r_{-1} \cdot B_{\text{h}-1}$$

$$T = \theta \cdot (Y + r_{-1} \cdot B_{\text{h}-1}) \quad \theta < 1$$

$$V = V_{-1} + (YD - C)$$

$$C = \alpha_1 \cdot YD + \alpha_2 \cdot V_{-1} \quad 0 < \alpha_2 < \alpha_1 < 1$$

$$H_{\text{h}} = V - B_{\text{h}}$$

$$\frac{B_{\text{h}}}{V} = \lambda_0 + \lambda_1 \cdot r - \lambda_2 \cdot \left(\frac{YD}{V}\right)$$

$$\frac{H_{\text{h}}}{V} = (1 - \lambda_0) - \lambda_1 \cdot r + \lambda_2 \cdot \left(\frac{YD}{V}\right)$$

$$\Delta B_{\text{s}} = B_{\text{s}} - B_{\text{s}-1} = (G + r_{-1} \cdot B_{\text{s}-1}) - (T + r_{-1} \cdot B_{\text{cb}-1})$$

$$\Delta H_{\text{s}} = H_{\text{s}} - H_{\text{s}-1} = \Delta B_{\text{cb}}$$

$$B_{\text{cb}} = B_{\text{s}} - B_{\text{h}}$$

$$r = \bar{r}$$

The redundant, or hidden, equation is:

$$H_{\text{h}} = H_{\text{s}}$$