# Shock Elasticities Toolbox

Yiran Fan, Paymon Khorrami, Joseph Huang [*]

March 22, 2018

## Contents

[*]Please email `paymon@uchicago.edu` or `jhuang12@uchicago.edu` with any comments or questions.

## 1 Introduction

The shock elasticities toolbox is a set of user-friendly MATLAB functions that compute shock elasticities defined as follows (for a full discussion on shock elasticities, please refer to Borovička and Hansen (2016), Borovička, Hansen, and Scheinkman (2014), and Hansen (2012)).

Assume we are given stochastic processes

$$\begin{aligned} dX_t &= \mu_X(X_t)dt + \sigma_X(X_t)dW_t \\ d\log C_t &= \mu_C(X_t)dt + \sigma_C(X_t)dW_t \\ d\log S_t &= \mu_S(X_t)dt + \sigma_S(X_t)dW_t \end{aligned} \tag{1}$$

where $X$ is n-dimensional, the Brownian motion $dW_t$ has $k$ shocks (hence $k$ dimensions), and

$$\begin{aligned} \mu_X &: \mathbb{R}^n \mapsto \mathbb{R}^n \\ \sigma_X &: \mathbb{R}^n \mapsto \mathbb{R}^{n\times k} \\ \mu_C, \mu_S &: \mathbb{R}^n \mapsto \mathbb{R} \\ \sigma_C, \sigma_S &: \mathbb{R}^n \mapsto \mathbb{R}^{1\times k} \end{aligned}$$

In the equations above, $C_t$ is the cash flow or consumption process and $S_t$ is the stochastic discount factor. We are interested in computing the **shock exposure elasticities** and **shock price elasticities**.

For a given cash flow or consumption process, its shock **exposure elasticity** can be computed as

$$\varepsilon_C^1(x,t) = \nu(x)\cdot\left[\sigma_X'(x)\left(\frac{\partial}{\partial x}\log E\Big[\frac{C_t}{C_0}|X_0=x\Big]\right) + \sigma_C(x)\right] \tag{2}$$

$$\varepsilon_C^2(x,t) = \frac{E\Big[\frac{C_t}{C_0}\nu(X_t)\cdot\sigma_C(X_t)|X_0=x\Big]}{E\Big[\frac{C_t}{C_0}|X_0=x\Big]} \tag{3}$$

where $\nu : \mathbb{R} \mapsto \mathbb{R}^k$ encodes the direction of the shocks.

To compute the price elasticities for a given cash flow process $C_t$ and stochastic discount factor $S_t$, we first compute the shock cost elasticities by substituting process $C$ in (2) and (3) with a new process $SC$ (and define them as $\varepsilon_{SC}^1$ and $\varepsilon_{SC}^2$). To get the **price elasticities**, we compute:

$$\varepsilon_C^1(x,t) - \varepsilon_{SC}^1(x,t) \tag{4}$$

and

$$\varepsilon_C^2(x,t) - \varepsilon_{SC}^2(x,t) \tag{5}$$

All expressions in (2) and (3) can be computed explicitly, with the exception of conditional expectations

$$E\Big[\frac{C_t}{C_0}|X_0=x\Big] \tag{6}$$

$$E\Big[\frac{C_t}{C_0}\nu(X_t)\cdot\sigma_C(X_t)|X_0=x\Big] \tag{7}$$

We compute the conditional expectations by using the Feynman-Kac formula.

Define

$$\phi_t(x) \equiv E\Big[\frac{C_t}{C_0}\phi_0(X_t)|X_0=x\Big]$$

Then $\phi_t(x)$ satisfies:

$$\frac{\partial\phi}{\partial t} = \Big(\mu_C + \frac{1}{2}|\sigma_C|^2\Big)\phi + \Big(\mu_X + \sigma_X\sigma_C\Big)\frac{\partial\phi}{\partial x} + \frac{1}{2}|\sigma_X|^2\frac{\partial^2\phi}{\partial x^2} \tag{8}$$

Let $\phi_0(x) = 1$ and $\phi_0(x) = \nu(x)\cdot\sigma_C(x)$ for (6) and (7) respectively.

# 2 Installation

## 2.1 Adding Toolbox

After downloading the toolbox, unzip and place the scripts in a folder. Afterwards, add the folder to your MATLAB search path by doing [1]

```
addpath('yourpath');
```

where `yourpath` is the path of the toolbox folder.

## 2.2 Creating MEX Function

We use the finite differences method to solve the PDE (8). To improve performance, the PDE solver is built in C++ and a C++ package called Eigen developed by Guennebaud, Jacob, et al. (2010). However, no knowledge of C++ is required, as the toolbox can pass data from MATLAB into C++ through a MEX function. All you need to do is to build the MEX function by calling

```
mex constructSolve.cpp
```

After that, you should see the following message:

```
MEX completed successfully.
```

The installation is complete. The next step is optional.

## 2.3 Linking with Pardiso (Optional)

To solve higher dimensional problems [2], it would be useful to employ Pardiso, a high performance matrix solver to solve PDE (8). To link with Pardiso, first go to www.pardiso-project.org and download the library by requesting an academic license for free, after which you should receive an email with the download link and a license key (a long string of numbers and letters). Follow the steps below:

- Copy and paste the license key into a plain file and save it as `pardiso.lic` in the path of the toolbox.
- Download the correct Pardiso library on the download page based on your operating system and compiler; save it in the path of the toolbox.
- Execute the following command in MATLAB

  ```
  mex constructSolvePardiso.cpp
  -L[path of toolbox] -l[name of library]
  -llapack -lblas -lpthread
  ```

  **Note**: Replace `[path of toolbox]` with the path of the toolbox and `[name of library]` with the file name of the library downloaded **excluding `lib` in the beginning and `.dylib` at the end** . For example, if the toolbox is located in `/Users/Yourname/shockElas` and the filename of the library is `libpardiso500-MACOS-X86-64.dylib`, compile with the following command:

  ```
  mex constructSolvePardiso.cpp
  -L/Users/Yourname/shockElas -lpardiso500-MACOS-X86-64
  -llapack -lblas -lpthread
  ```

After that, you should see:

```
MEX completed successfully.
```

This step is then complete.

[1](Optional): To avoid repeating this command in each session that this toolbox is employed, try to add the folder onto the search path permanently (link to instructions).

[2]Pardiso could be complicated to set up initially, so if your model is less than two dimensions, it may not be necessary to use Pardiso.

# 3 Using the Toolbox

## 3.1 Computing Shock Elasticities: `computeElas`

This is the function that computes the shock exposure and price elasticities (both the first and second types). It takes in the following arguments:

- `domain`: An $S \times n$ matrix that has all the grid points in the state space. $n$ is the number of dimensions and S is the total number of grid points initialized. The grid points must be vectorized from MATLAB's `ndgrid` format[3]. For example, suppose the model has two state variables and each state variable has 100 grid points. The `domain` argument would be a $10,000 \times 2$ matrix.

- `model`:
  A structure that contains the information described in (1). Specifically, it has the following fields

  - `muX`: A function handle[4] that computes $\mu_X$ as in (1) for each of the grid points in `domain`; it takes `domain` as the argument and returns an $S \times n$ matrix;
  - `sigmaX`: A cell of function handles that compute $\sigma_X$ for each of the grid points in `domain`; each element of the cell, a function handle, takes `domain` as the argument and returns a matrix. For example, suppose there are two state variables and three shocks. There should be two function handles in `sigmaX` which would each return an $S \times 3$ matrix.
  - `muC`: A function handle that computes $\mu_C$ as in (1) for each of the grid points in `domain`; it takes `domain` as the argument and returns an $S \times 1$ matrix;
  - `sigmaC`: A function handle that computes $\sigma_C$ as in (1) for each of the grid points in `domain`; it takes `domain` as the argument and returns an $S \times k$ matrix (assuming there are $k$ shocks);
  - `muS`: A function handle that computes $\mu_S$ as in (1) for each of the grid points in `domain`; it takes `domain` as the argument and returns an $S \times 1$ matrix;
  - `sigmaS`: A function handle that computes $\sigma_S$ as in (1) for each of the grid points in `domain`; it takes `domain` as the argument and returns an $S \times k$ matrix (assuming there are $k$ shocks);
  - `dt`: The length of time step;
  - `T`: Total number of time periods. Suppose `dt = 1/12` implies one month and `model.T = 120`. The total time that the elasticities are solved for is 10 years.

- `bc`:
  A structure that specifies the boundary conditions in this form:

$$0 = a_0(t) + a_1(t)\phi(x) + a_2(t)\frac{\partial}{\partial x}\phi(x) + a_3(t)\frac{\partial^2}{\partial x^2}\phi(x) \tag{9}$$

  where $a_0$ is a scalar, $a_1$, $a_2$, and $a_3$ are $1 \times n$ vectors ($n$ being the number of dimensions). The user should configure `bc` in this way:

  - `a0`: a scalar that corresponds to $a_0$ in (9)
  - `level`: a $1 \times n$ that corresponds to $a_1$ in (9)
  - `first`: a $1 \times n$ that corresponds to $a_2$ in (9)
  - `second`: a $1 \times n$ that corresponds to $a_3$ in (9)
  - `natural`: a boolean (`true` or `false`) that determines whether to use natural boundaries[5]; if `true`, the fields above would be ignored.

  For example, to implement the boundary conditions $0 = \frac{\partial}{\partial x}\phi(x)$ (i.e. first derivatives are set to zero), one should set up `bc` by doing:

```
bc.a0 = 0; bc.first = [1 1]; bc.second = [0 0];
bc.level = [0 0]; bc.natural = false;
```

[3]For example, `[X,Y] = ndgrid(1:2:19,2:2:12); domain = [X(:)  Y(:)]`.

[4]If you only have numerical values of the drift(s) of your state variable(s), i.e. the state variable(s) are endogenous, you can create an interpolant by using MATLAB function `griddedInterpolant`; the same applies to the items below.

[5]Natural boundaries simply take the limit of PDE (8), where second derivatives at the boundary are approximated by one grid point inside the boundary.

- `x0`: A matrix that contains all the starting points (i.e. $x$ in (2)). For $l$ starting points, `x0` should be an $l \times n$ matrix ($n$ being the number of dimensions).
- `optArgs` (optional): A structure that contains optional arguments. It has two fields -
  - `usePardiso`: a boolean that determines whether to use Pardiso to solve high dimensional and large matrix systems when solving the PDE (8). It is `false` by default. To use this feature, one must complete section 2.3.
  - `priceElas`: a boolean that determines whether to compute price elasticities. By default, the function returns both exposure and price elasticities (i.e. `priceElas` is set to `true` by default).

The function returns exposure and price elasticities. Suppose the user inputs $l$ starting points (i.e. `x0` has $l$ rows) and $T$ time periods and chooses to compute price elasticities (the default option), the function will return two cells:

- `expoElas`: a cell of $l$ elements where each element is a structure that contains two fields `firstType` and `secondType`. Each field is a $T \times n$ matrix ($n$ being the number of dimensions). As the name suggests, `firstType` is the first type shock exposure elasticities for the $i$th starting point (the $i$th row in input `x0`) up to $T$ time periods, whereas `secondType` is the second type shock exposure elasticities[6].
- `priceElas`: similar to `expoElas`, but contains the price elasticities.

## 3.2 Computing Stationary Distribution: `simStatDent`

The toolbox uses simulations to compute stationary density as it is an approach that has guaranteed success if implemented correctly. Given the stochastic process $dX_t$ specified in (1), function `simStatDent` simulates by

1. Given a starting point $x$ at $t = 0$ (labeled $x_0$), it determines the drift $\mu_X$ and $\sigma_X$;
2. Draw the Brownian vector $dW_t \sim \mathcal{N}(\mathbf{0}_k, dt \times \mathbb{I}_k)$ and compute $dX_t$, where $\mathbf{0}_k$ is a $k \times 1$ vector and $\mathbb{I}_k$ is the $k \times k$ identity matrix ($k$ being the number of shocks) so that $x$ at $t + dt$ can be determined;
3. Repeat the two steps above $T/dt$ times so that it has a history of $T/dt$ time periods.

For each starting point $x_0$, the function will generate a history of time paths. We recommend that the user use multiple starting points as each history is embarrassingly parallel (so parallel computing can be used to speed up the process)[7].

The function takes in the following inputs: Let $n$ be the number of dimensions, $S$ total number of grid points, $k$ number of shocks, and $l$ number of starting points

- `stateSpace`: a cell of grid vectors for each of the state variables. It mirrors the input for MATLAB's `ndgrid` function[8].
- `hist0`: an $l \times n$ matrix that contains the starting points for the simulations . Having $l$ starting points will generate $l$ number of time paths.
- `dt`: the length of time step. This corresponds to $dt$ in (1) and the variance of the Brownian shock in the simulations.
- `T`: the total length of time for each time path. For each starting point, the function will simulate for $T/dt$ time periods. If you have $l$ starting points, the function will generate $l$ time paths and $l \times (T/dt)$ time points in total.
- `drifts`: A cell that contains $n$ elements where each element is either a function handle that computes the drift of the state variable or a column vector that contains the values of the drift for the state variable at each grid point of the state space;
- `vols`: A cell that contains $n$ elements where each element is either a function handle that computes the volatility of the state variable (a $1 \times m$ vector) or an $S \times m$ that contains the values of volatility for the state variable at each grid point of the state space;

The function outputs a cell of $l$ elements, where each element is a $(T/dt) \times n$ matrix, corresponding to each history of points.

[6]For example, to access the first type shock exposure elasticity of the second shock at the third starting point (in the third row of input `x0`), type `expoElas{3}.firstType(:,2)`

[7]By default, the function uses `parfor` which employs all cores available on your computer, assuming that MATLAB's Parallel Computing Toolbox is installed. To check whether it is installed, simply type `ver` in the MATLAB command window and you should see Parallel Computing Toolbox listed. If not, go to this link for more information

[8]For example, `stateSpace = {0:1:10,10:5:50}` is a valid input and creates two state variables.

## 4 Examples

### 4.1 He and Krushnamurthy (2013)

This example borrows from He and Krishnamurthy (2013). The model specifications are as follows.

The state variable is characterized by

$$\mu_X(x) = x\Big[ -\frac{\ell}{1+\ell}\rho + (\alpha(x)-1)^2\sigma_Y^2 \Big]$$
$$\sigma_X(x) = x(\alpha(x)-1)\sigma_Y$$

And the stochastic discount factor by:

$$\mu_S(x) = \frac{\rho}{1+\ell} + \mu_Y - \alpha(x)\sigma_Y^2 + \frac{1}{2}\alpha(x)^2\sigma_Y^2$$
$$\sigma_S(x) = \alpha(x)\sigma_Y$$

We are interested in two cash flows $G^1$ (aggregate endowment) and $G^2$ (expert consumption):

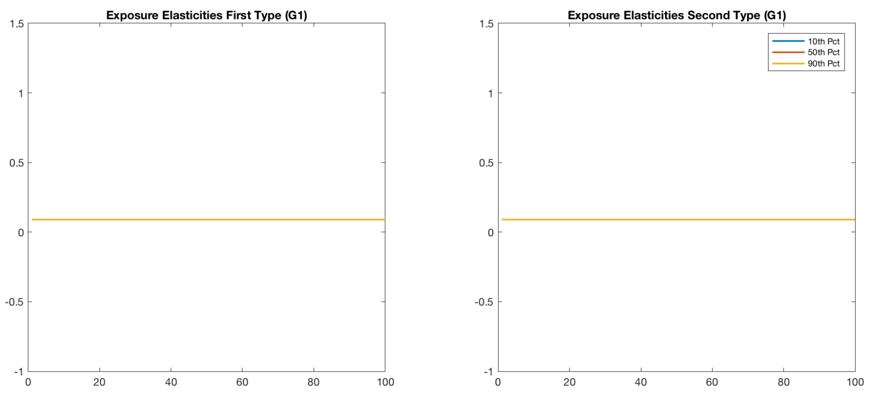
$$\mu_{G^1}(x) = \mu_Y - \frac{1}{2}\sigma_Y^2$$
$$\sigma_{G^1}(x) = \sigma_Y$$
$$\mu_{G^2}(x) = \mu_S(x) - \rho$$
$$\sigma_{G^2}(x) = \sigma_S(x).$$

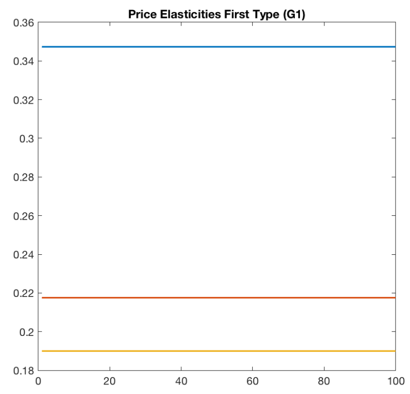
The parameters in the equations above are either computed or set as:

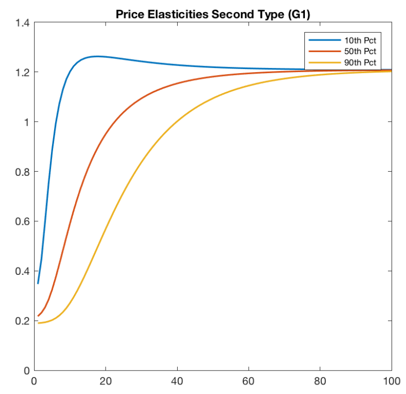
$$\mu_Y = 0.02 \quad \sigma_Y = 0.09 \quad m = 4$$
$$\lambda = 0.6 \quad \rho = 0.04 \quad \ell = 1.84$$
$$\alpha(x) = \begin{cases} [1-\lambda(1-x)]^{-1}, & \text{if } x > x^* \\ (1+m)^{-1}x^{-1}, & \text{if } x \le x^*. \end{cases}$$
$$x^* = \frac{1-\lambda}{1-\lambda+m} \in (0,1).$$

The file `HeKrushnamurthy.m` computes the shock elasticities for both cash flows and the stationary density through simulations. It will render charts similar to below (minor differences could be observed due to the nature of simulations).

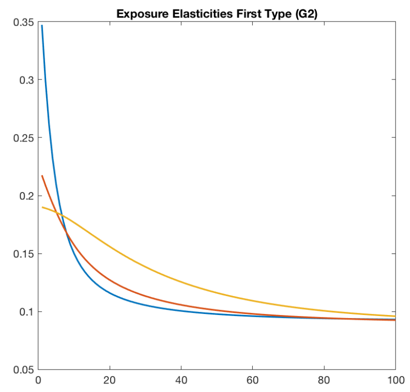
The elasticities of the first cash flow $G^1$:

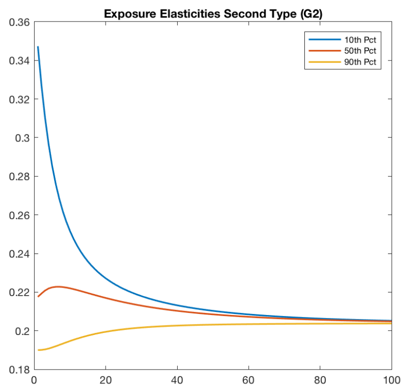


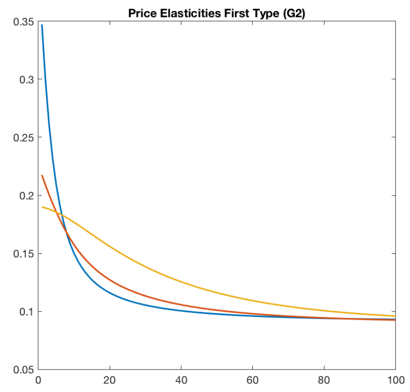


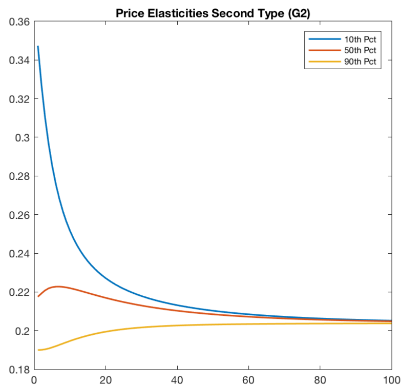


The elasticities of the second cash flow $G^2$:









## 4.2 Bansal and Yaron (2004)

This example `BansalYaron.m` is adapted from Bansal and Yaron (2004). Different from the example in 4.1, the model is two dimensional and has three shocks. The example computes elasticities for two utility functions (i.e. two SDFs). Note that we have created a detailed live script that shows how the script works step by step (click here).

The model has two state variables that follow stochastic processes:

$$
\begin{aligned}
dX_t^{[1]} &= -0.021 X_t^{[1]} dt + \sqrt{X_t^{[2]}} \begin{bmatrix} 0.00031 & -0.00015 & 0 \end{bmatrix} dW_t \\
dX_t^{[2]} &= -0.013 (X_t^{[2]} - 1) dt + \sqrt{X_t^{[2]}} \begin{bmatrix} 0 & 0 & -0.038 \end{bmatrix} dW_t
\end{aligned}
$$

Where aggregate consumption follows:

$$
d \log C_t = 0.0015 dt + X_t^{[1]} dt + \sqrt{X_t^{[2]}} \begin{bmatrix} 0.0034 & 0.007 & 0 \end{bmatrix} dW_t
$$

Please note that the parameter settings and specific continuous-time specification comes from the handbook chapter of Hansen, Heaton, Lee, and Roussanov (2007) to well approximate the dynamics used by Bansal and Yaron (2004). As in Hansen (2012), we transform two of the shocks to the macroeconomy in order that one is permanent and the other is transitory. Furthermore, the intertemporal elasticity parameter is unity and the risk aversion parameter is 8.

Suppose that the state dynamics are specified by

$$\mu(x) = \begin{bmatrix} \bar{\mu}_{11} & \bar{\mu}_{12} \\ 0 & \bar{\mu}_{22} \end{bmatrix} \begin{bmatrix} x^{[1]} + \iota_1 \\ x^{[2]} + \iota_2 \end{bmatrix} \text{ and } \sigma(x) = \sqrt{x^{[2]}} \begin{bmatrix} \bar{\sigma}_1 \\ \bar{\sigma}_2 \end{bmatrix}$$

and the consumption dynamics take the following form:

$$\beta(x) = \bar{\beta}_{c,0} + \bar{\beta}_{c,1}(x^{[1]} - \iota_1) + \bar{\beta}_{c,2}(x^{[2]} - \iota_2) \text{ and } \alpha(x) = \sqrt{x^{[2]}}\bar{\alpha}_c$$

We can recast the model by setting:

$$\begin{bmatrix} \bar{\mu}_{11} & \bar{\mu}_{12} \\ 0 & \bar{\mu}_{22} \end{bmatrix} = \begin{bmatrix} -0.021 & 0 \\ 0 & -0.013 \end{bmatrix}, \begin{bmatrix} \iota_1 \\ \iota_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \text{ and } \begin{bmatrix} \bar{\sigma}_1 \\ \bar{\sigma}_2 \end{bmatrix} = \begin{bmatrix} 0.00031 & -0.00015 & 0 \\ 0 & 0 & -0.038 \end{bmatrix}$$
$$\bar{\beta}_{c,0} = 0.0015;\ \bar{\beta}_{c,1} = 1;\ \bar{\beta}_{c,2} = 0$$
$$\bar{\alpha}_c = \begin{bmatrix} 0.0034 & 0.007 & 0 \end{bmatrix}$$

We compute the shock elasticities for both the recursive utility and power utility SDFs. The SDF of the recursive utility is given by:

$$S_t = \exp(-\delta t)\left(\frac{C_t}{C_0}\right)^{-1} \tilde{M}_t$$

where $d\log \widetilde{M}_t = -\frac{|\tilde{\alpha}|^2}{2} X_t^{[2]} dt + \sqrt{X_t^{[2]}}\tilde{\alpha} \cdot dW_t$.

When we take logs:

$$d\log(S_t) = -\delta dt - d\log(C_t) + \sqrt{X_t^{[2]}}\tilde{\alpha} \cdot dW_t - \frac{|\tilde{\alpha}|^2}{2} X_t^{[2]} dt$$
$$d\log(S_t) = -\delta dt - \left[\bar{\beta}_{c,0} dt + \bar{\beta}_{c,1} X_t^{[1]} dt + \bar{\beta}_{c,2}(X_t^{[2]} - 1)dt + \sqrt{X_t^{[2]}}\bar{\alpha}_c \cdot dW_t\right] + \sqrt{X_t^{[2]}}\tilde{\alpha} \cdot dW_t - \frac{|\tilde{\alpha}|^2}{2} X_t^{[2]} dt$$
$$= \left[-\delta - \bar{\beta}_{c,0} - \bar{\beta}_{c,1} X_t^{[1]} - \bar{\beta}_{c,2}(X_t^{[2]} - 1) - \frac{|\tilde{\alpha}|^2}{2} X_t^{[2]}\right] dt + \sqrt{X_t^{[2]}}(\tilde{\alpha} - \bar{\alpha}_c) dW_t$$

Where $\tilde{\alpha} = (1-\gamma)[(\bar{\sigma}_1)'\bar{v}_1 + (\bar{\sigma}_2)'\bar{v}_2 + \bar{\alpha}_c]$ and $\bar{v}_1$ and $\bar{v}_2$ are computed by:

$$\bar{v}_1 = \frac{-\bar{\beta}_{(c,1)}}{\bar{\mu}_{11} - \delta}$$
$$\bar{v}_2 = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Where

$$A = \frac{1-\gamma}{2}(\bar{\sigma}_1^2 + \bar{\sigma}_2^2)$$
$$B = -\delta + \bar{\mu}_{22} + (1-\gamma)(\bar{\alpha}_c)'(\bar{\sigma}_2)' + 2v_1\frac{1-\gamma}{2}\bar{\sigma}_1\bar{\sigma}_2$$
$$C = \bar{\mu}_{12}\bar{v}_1 + \bar{\beta}_{c,2} + (1-\gamma)(\bar{\alpha}_c)'(\bar{\sigma}_1)'\bar{v}_1 + \frac{1-\gamma}{2}(\bar{\sigma}_1^2\bar{v}_1^2 - \bar{\alpha}_c^2)$$

We also compute the shock elasticities for power utility, whose SDF is given by

$$S_t = \exp(-\delta t)\left(\frac{C_t}{C_0}\right)^{-\gamma}$$

Taking logs yields:

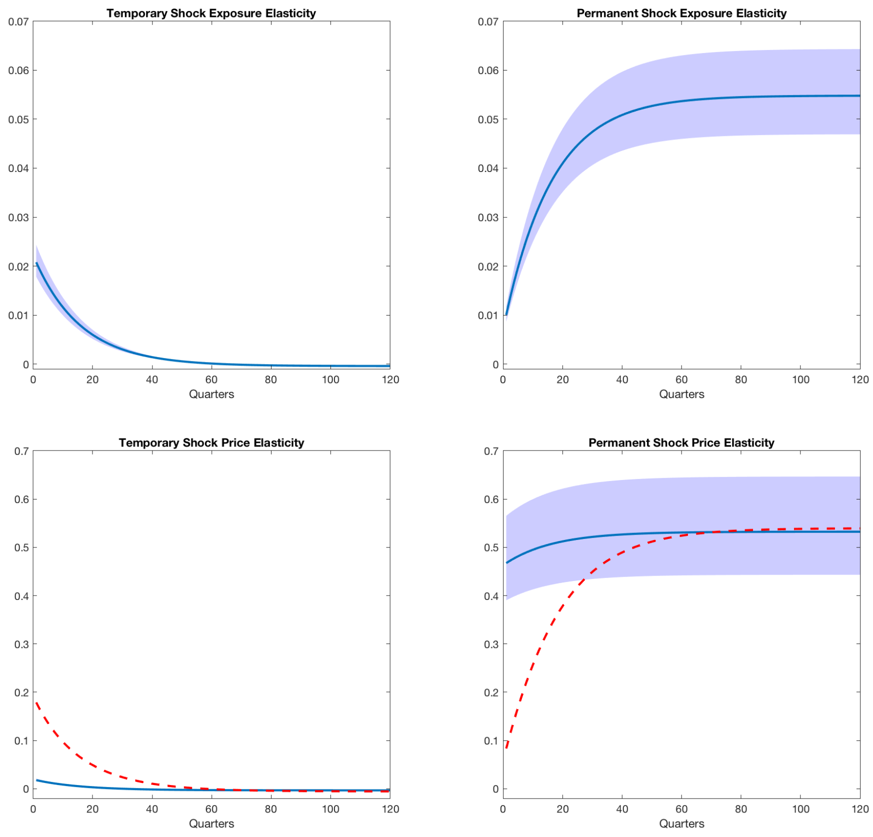
$$\begin{aligned} d\log(S_t) &= -\delta dt - \gamma d\log(C_t) \\ &= -\delta dt - \gamma\left[\mu_C(x)dt + \sqrt{X_t^{[2]}}\bar{\alpha}_c \cdot dW_t\right] \\ &= \left[-\delta - \gamma\mu_C(x)\right]dt + \gamma\sqrt{X_t^{[2]}}(-\bar{\alpha}_c)dW_t \end{aligned}$$

With these equations, we are ready to compute shock elasticities using the toolbox. We will compute the shock elasticities for starting points at the mean of $X^{[1]}$ and the 10th, 50th, and 90th percentiles of $X^{[2]}$ (3 starting points in total). In particular, some of the arguments for function `computeElas` are as follows -

Let $S$ be the total number of grid points:

- `model`
  - `muC`: a function handle that returns an $S \times 1$ matrix
  - `sigmaC`: a function handle that returns an $S \times 3$ matrix (remember, there are **3** shocks)
  - `muS`: a function handle that returns an $S \times 1$ matrix
  - `sigmaS`: a function handle that returns an $S \times 3$ matrix
  - `muX`: a function handle that returns an $S \times 2$ matrix (there are **2** state variables)
  - `sigmaX` a cell that contains 2 function handles, with each handle returning an $S \times 3$ matrix
- `x0`: a $3 \times 2$ matrix (there are **3** starting points and **2** dimensions)

You should expect the elasticities to look like:



## References

Bansal, Ravi, and Amir Yaron, 2004, Risks for the long run: A potential resolution of asset pricing puzzles, *Journal of Finance*.

Borovička, Jaroslav, and Lars Peter Hansen, 2016, Term structure of uncertainty in the macroeconomy, *Handbook of Macroeconomics* 2, 1641–1696.

Borovička, Jaroslav, Lars Peter Hansen, and José A. Scheinkman, 2014, Shock elasticities and impulse responses, *Mathematics and Financial Economics* 8.

Guennebaud, Gaël, Benoît Jacob, et al., 2010, Eigen v3, .

Hansen, Lars Peter, 2012, Dynamic Valuation Decomposition within Stochastic Economies, *Econometrica* 80.

——— , John Heaton, Junghoon Lee, and Nikolai Roussanov, 2007, Intertemporal substitution and risk aversion, *Handbook of Econometrics, Volume 6A*.

He, Zhiguo, and Arvind Krishnamurthy, 2013, Intermediary Asset Pricing, *The American Economic Review* 103(2).