Homework 2

Life-Cycle Heterogeneous Agents Models

Carlos Rojas Quiroz

January 2023

Homework 2

- I chose exercise 2 to solve in the computer

Codes can be found here: 

- For exercises 1 and 3, I wrote "pseudo-codes" with a `Matlab`-style
- Also, I've updated the results of Hugget's model from homework 1 (see the new PDF file here).

Exercise 1: Projection methods

Exercise 1

The residual function

- I define the residual function as a first step. From the consumer's problem:

$$\max_{a'} \frac{(a-a')^{(1-\sigma)}}{1-\sigma} + \beta S_j \mathbb{E}\left(\frac{((1+r)a' + \exp(\varepsilon))^{1-\sigma}}{1-\sigma}\right)$$

where $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Then, the FOC is:

$$(a-a')^{-\sigma} = \beta S_j (1+r)\mathbb{E}\left((1+r)a' + exp(\varepsilon)\right)^{-\sigma}$$

which is the Euler equation. Therefore, the residual function is defined as follows:

$$\mathbb{R} = \mathbb{E}\left\{\frac{(a-a')^{-\sigma}}{\left((1+r)a' + exp(\varepsilon)\right)^{-\sigma}} - \beta S_j (1+r)\right\} \tag{1}$$

Exercise 1

The residual function

- Now, I define the approximate solution as:

$$a' \approx \hat{a}'(a) \tag{2}$$

- Plugging 2 in 1:

$$\mathbb{R}(a) = \mathbb{E}\left\{ \frac{(a - \hat{a}'(a))^{-\sigma}}{\left((1+r)\hat{a}'(a) + exp(\varepsilon)\right)^{-\sigma}} - \beta S_j (1+r) \right\} \tag{3}$$

Exercise 1

Approximate solution

- For this exercise I define the following:
  - A basis: monomials
  - An order of approximation: $p = 3$
  - An interval over which we approximate the function: $[a_{min}, a_{max}] = [1, 20]$
- Then equation 2 becomes:

$$\hat{a}'(a; \gamma) = \gamma_1 + \gamma_2 a + \gamma_3 a^2 \tag{4}$$

- I understand "[s]olve the model" as computing policy functions.

Exercise 1.1: Collocation/ Dirac Delta

Gauss-Hermite approximation

- To deal with expectations in equation 3, I apply the Gauss-Hermite approximation. Let be $x = \frac{\varepsilon}{\sigma_\varepsilon \sqrt{2}}$, then:

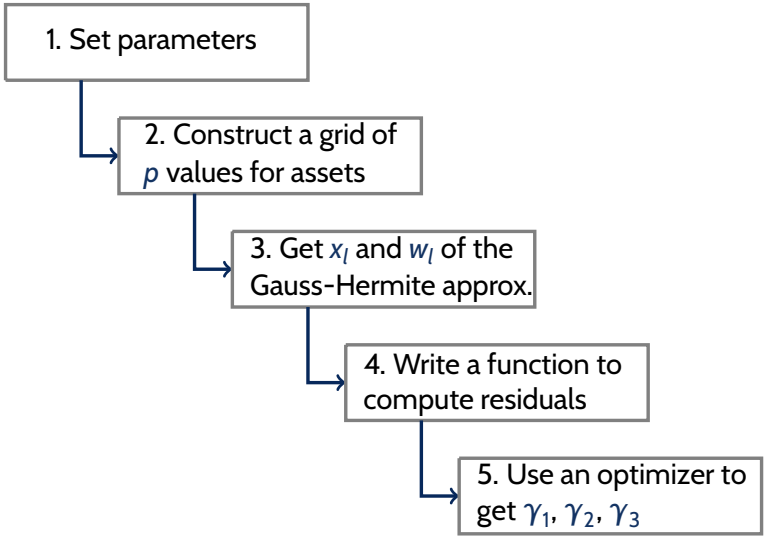
$$\mathbb{R}(a;\gamma) = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{\pi}} \exp(-x^2) \left\{ \frac{(a - \hat{a}'(a;\gamma))^{-\sigma}}{\left((1+r)\hat{a}'(a;\gamma) + exp(x\sigma_\varepsilon\sqrt{2})\right)^{-\sigma}} - \beta S_j(1+r) \right\}$$

- Then the Gauss-Hermite approximation is defined as follows:

$$\mathbb{R}(a;\gamma) \approx \sum_{l=1}^{n_{GH}} \frac{1}{\sqrt{\pi}} w_l \left\{ \frac{(a - \hat{a}'(a;\gamma))^{-\sigma}}{\left((1+r)\hat{a}'(a;\gamma) + exp(x_l\sigma_\varepsilon\sqrt{2})\right)^{-\sigma}} - \beta S_j(1+r) \right\} \quad (5)$$

Exercise 1.1: Collocation/ Dirac Delta

Pseudo-code



Exercise 1.1: Collocation/ Dirac Delta

Pseudo-code

1. Set parameters:

```
a_min=1; % borrowing constraint
a_max=20; % assets maximum value
p=3; % order of approximation
n_GH=1000; % number of integration points in Gauss-Hermite
```

   and others defined in the exercise's description (σ_ε, β, r, etc).

2. Construct a grid of p values for assets. For instance, I use the Chebyshev zeros:

```
for ii=1:p
    a_grid(ii)=((a_max-a_min)*cos((2*ii-1)/(2*p)*pi)+...
    (a_max+a_min))/2;
end
```

3. Get the integration nodes x_l and weights w_l of the Gauss-Hermite approximation. For instance, I can work with this function written for Matlab:

```
[x,w]=lgwt(n_GH,-1.96*sigma_e,1.96*sigma_e);
```

   Notice nodes are restricted to the interval $[-1.96\sigma_\varepsilon, 1.96\sigma_\varepsilon]$. Moreover, **x** and **w** are vectors of $1 \times n_{GH}$ values.

Exercise 1.1: Collocation/ Dirac Delta

Pseudo-code

4. Write a function to compute equation 5 for all the p assets values:

```
function R=Residual(G,a_grid,p,sigma,r,sigma_e,b_ta,Sj,w,x)
ap=G(1)+G(2)*a_grid+G(3)*a_grid.^2;
R=zeros(1,p);
for ii=1:p
    RR=(a_grid(ii)-ap(ii))^(-sigma)./(((1+rg)*ap(ii)+...
    exp(x*sigma_e*2^0.5)).^(-sigma))-b_ta*Sj*(1+r);
    R(ii)=(w'*RR).*1/(pi^0.5);
end
```

5. Use an optimizer to get values of γ_1, γ_2 and γ_3 that solve the system of p equations. However, take into account the next constraints:

$$\gamma_1 + \gamma_2 + \gamma_3 \geq 1 \quad \text{and} \quad \gamma_1 + 20\gamma_2 + 400\gamma_3 \leq 20$$

An option is `fmincon`. Another option is to use a two-step algorithm: running a global search over reasonable intervals for each parameter (for instance $[0, 1]$, $[0, 1]$ and $[0, 0.01]$, respectively), then use an optimizer like `fsolve` using the previous results as initial points.

Exercise 1.2: Least squares projection

Gauss-Chebyshev approximation

- In this case, the function to minimize is defined as follows:

$$\min_{\gamma} \int_{a_{min}}^{a_{max}} \mathbb{R}(a;\gamma)^2 da \tag{6}$$

  where $\mathbb{R}(a;\gamma)$ comes from equation 5.

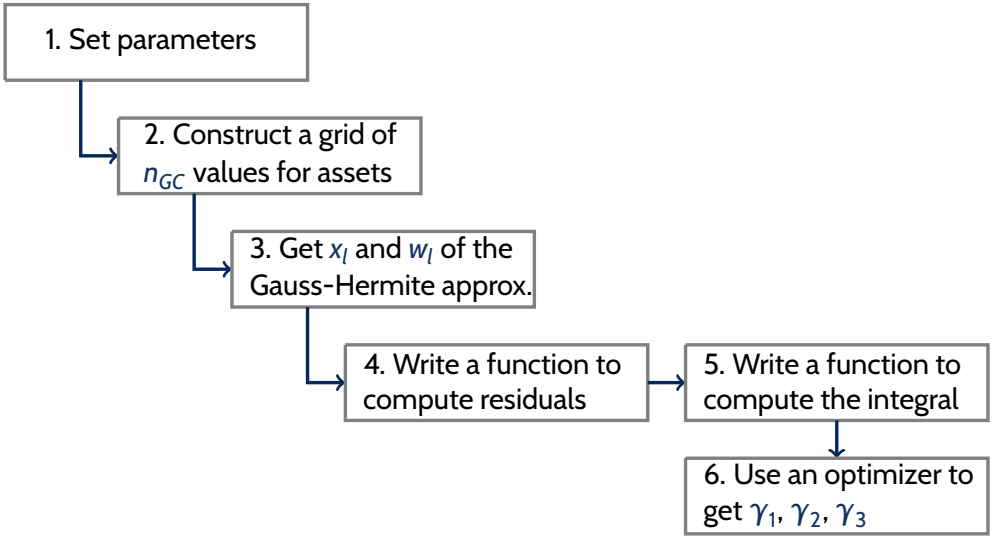
- In order to compute the integral in equation 6, I apply the Gauss-Chebyshev quadrature formula:

$$\int_{a_{min}}^{a_{max}} \mathbb{R}(a;\gamma)^2 da = \frac{\pi(a_{max} - a_{min})}{2n_{GC}} \sum_{h=1}^{n_{GC}} \mathbb{R}(a_h;\gamma)^2 \sqrt{1+\tilde{a}_h} \tag{7}$$

  where $\tilde{a}_h$ are the Chebyshev zeros.

Exercise 1.2: Least squares projection

Pseudo-code



Exercise 1.2: Least squares projection

Pseudo-code

1. Set parameters:

```
a_min=1; % borrowing constraint
a_max=20; % assets maximum value
n_GH=1000; % number of integration points in Gauss-Hermite
n_GC=1000; % number of integration points in Gauss-Chebyshev
```

   and others defined in the exercise's description (σ_ε, β, r, etc).

2. Construct a grid of n_{GC} values for assets using the Chebyshev zeros:

```
for ii=1:n_GC
    a_tilde(ii)=cos((2*ii-1)/(2*n_GC)*pi);
    a_grid(ii)=((a_max-a_min)*a_tilde(ii)+(a_max+a_min))/2;
end
```

3. Get the integration nodes x_l and weights w_l of the Gauss-Hermite approximation:

```
[x,w]=lgwt(n_GH,-1.96*sigma_e,1.96*sigma_e);
```

4. Write a function to compute equation 5 for all the n_{GC} assets values (same function as step 4 from the previous algorithm, but instead of p I set n_{GC}).

Exercise 1.2: Least squares projection

Pseudo-code

5. Write a function to compute the integral in equation 7:

```
function S=Integral(G,a_grid,n_GC,sigma,r,sigma_e,b_ta,Sj,...
w,x,a_tilde,a_min,a_max)
R=Residual(G,a_grid,n_GC,sigma,r,sigma_e,b_ta,Sj,w,x);
S=pi*(a_max-a_min)/(2*n_GC)*sum((R.^2).*sqrt(1+a_tilde));
```

6. Use an optimizer to get values of γ_1, γ_2 and γ_3 that minimize equation 7. Take into account the constraints:

$$\gamma_1 + \gamma_2 + \gamma_3 \geq 1 \quad \text{and} \quad \gamma_1 + 20\gamma_2 + 400\gamma_3 \leq 20$$

An option is `fmincon`. Another option is to use a two-step algorithm: running a global search over reasonable intervals for each parameter (for instance $[0, 1]$, $[0, 1]$ and $[0, 0.01]$, respectively), then use an optimizer like `fminsearch` using the previous results as initial points.

Exercise 2: GMM estimation

Exercise 2

- There are six possible pair combinations. In the next tables, I call (x, y) to the estimation using moments $x - th$ and $y - th$. I also run an estimation using all four moments $(1, 2, 3, 4)$.
- I used a two-step algorithm to estimate parameters by GMM. In the first step, I ran a grid search of P and λ over reasonable intervals ($[1e-4, 4]$ and $[1e-4, 0.2]$ for P and λ, respectively). In the second step, I used the previous results as initial points in a `Matlab` built-in optimizer. I worked with `fminsearch` as default.
- I also used three new optimizers (`fminunc`, `fmincon`, and `patternsearch`) as a robustness check.

Exercise 2.1: results

- Results using the identity and the optimal matrices are shown in the following table (columns 1 and 2, respectively). Notice that tiny differences exist in the value of the estimated parameters between both methods.
- The main difference between estimations arises in the case of the over-identified system (4 moments > 2 parameters), where values of $\hat{P}$ and $\hat{\lambda}$ differ significantly between columns
- Regarding inference, in the case of the identity matrix, notice that both parameters are statistically significant at 95% level for most pairs. The only case where it is not is when the GMM estimator is computed using the second and fourth moments.
- The estimation using the second and fourth moments is susceptible to the choice of the initial parameters. Small changes at the initial point may get non-real (complex) values for the standard errors, as is observed in the robustness check table as well.
- This apparent problem is corrected with the optimal matrix. Using it, I obtained statistically significant parameters in all the cases.

Exercise 2.1: results

Moments	(1) W=Identity matrix		(2) W=Optimal	
	$\hat{P}$	$\hat{\lambda}$	$\hat{P}$	$\hat{\lambda}$
(1,2)	2.0568	0.0658	2.0568	0.0658
	(0.5035)	(0.0189)	(0.5035)	(0.0189)
(1,3)	2.4106	0.0771	2.4106	0.0771
	(0.6085)	(0.0256)	(0.6085)	(0.0256)
(1,4)	2.7720	0.0886	2.7720	0.0886
	(0.5860)	(0.0268)	(0.5860)	(0.0268)
(2,3)	2.2645	0.0713	2.2645	0.0713
	(0.5327)	(0.0211)	(0.5385)	(0.0216)
(2,4)	2.6090	0.0805	2.6091	0.0805
	(32.3094)	(0.8434)	(0.5221)	(0.0229)
(3,4)	3.0358	0.1018	3.0358	0.1018
	(0.6157)	(0.0292)	(0.6158)	(0.0292)
(1,2,3,4)	2.0583	0.0658	3.3589	0.1245
	(0.5035)	(0.0189)	(0.6463)	(0.0340)

NOTE. Standard errors are reported in parentheses.

Exercise 2.2: robustness check

- Robustness check using three new optimizers (`fminunc`, `fmincon`, and `patternsearch`) are shown in the next two slides for the case of identity (columns 3, 4, and 5) and optimal (columns 6, 7, and 8) matrices, respectively.
- Results are sensitive to using a solver when the weighting matrix is the identity one. However, the difference between them and `fminsearch` is small. Notice that I could not find real-valued standard errors when using the second and fourth moments (negative values in the diagonal of the var-covar matrix).
- Results are robust to the optimizer's choice when I used the optimal matrix and when the system is exactly identified. When the four moments are included, results differ slightly between different optimizers.

Exercise 2.2: robustness check

identity matrix

Moments	(3) `fminunc`		(4) `fmincon`		(5) `patternsearch`	
	$\hat{P}$	$\hat{\lambda}$	$\hat{P}$	$\hat{\lambda}$	$\hat{P}$	$\hat{\lambda}$
(1,2)	2.0633	0.0659	2.0524	0.0656	2.0633	0.0659
	(0.5061)	(0.0190)	(0.5017)	(0.0188)	(0.5061)	(0.0190)
(1,3)	2.4026	0.0768	2.4062	0.0769	2.4026	0.0768
	(0.6042)	(0.0254)	(0.6061)	(0.0255)	(0.6043)	(0.0254)
(1,4)	2.7803	0.0889	2.5323	0.0810	2.7803	0.0889
	(0.5910)	(0.0269)	(0.4525)	(0.0227)	(0.5911)	(0.0269)
(2,3)	2.1801	0.0691	2.1801	0.0691	2.1801	0.0691
	(0.4655)	(0.0200)	(0.4466)	(0.0202)	(0.5337)	(0.0213)
(2,4)	2.7211	0.0835	2.7211	0.0835	2.7211	0.0835
	(–)	(–)	(–)	(–)	(–)	(–)
(3,4)	3.0284	0.1015	3.0345	0.1018	3.0287	0.1015
	(0.6112)	(0.0291)	(0.6149)	(0.0292)	(0.6114)	(0.0291)
(1,2,3,4)	2.0633	0.0659	2.0528	0.0657	2.0633	0.0659
	(0.5055)	(0.0190)	(0.5012)	(0.0188)	(0.5055)	(0.0190)

NOTE. Standard errors are reported in parentheses.

Exercise 2.2: robustness check

optimal matrix

Moments	(6) `fminunc`		(7) `fmincon`		(8) `patternsearch`	
	$\hat{P}$	$\hat{\lambda}$	$\hat{P}$	$\hat{\lambda}$	$\hat{P}$	$\hat{\lambda}$
(1,2)	2.0569	0.0658	2.0568	0.0658	2.0568	0.0658
	(0.5035)	(0.0189)	(0.5035)	(0.0189)	(0.5035)	(0.0189)
(1,3)	2.4106	0.0771	2.4106	0.0771	2.4106	0.0771
	(0.6085)	(0.0256)	(0.6085)	(0.0256)	(0.6085)	(0.0256)
(1,4)	2.7720	0.0886	2.7720	0.0886	2.7720	0.0886
	(0.5860)	(0.0268)	(0.5860)	(0.0268)	(0.5860)	(0.0268)
(2,3)	2.2645	0.0713	2.2645	0.0713	2.2645	0.0713
	(0.5385)	(0.0216)	(0.5385)	(0.0216)	(0.5385)	(0.0216)
(2,4)	2.6091	0.0805	2.6090	0.0805	2.6090	0.0805
	(0.5220)	(0.0229)	(0.5220)	(0.0229)	(0.5220)	(0.0229)
(3,4)	3.0358	0.1018	3.0358	0.1018	3.0358	0.1018
	(0.6157)	(0.0292)	(0.6157)	(0.0292)	(0.6157)	(0.0292)
(1,2,3,4)	3.3630	0.1246	3.3547	0.1243	3.3628	0.1246
	(0.6449)	(0.0340)	(0.6480)	(0.0341)	(0.6448)	(0.0340)

NOTE. Standard errors are reported in parentheses.

Exercise 3: Estimation of life-cycle model

Exercise 3.1: Euler equation to write a set of moment conditions

- From exercise 1 (equation 1), we know the Euler equation is defined as follows:

$$\mathbb{E}\left\{\frac{(a-a')^{-\sigma}}{\left((1+r)a'+exp(\varepsilon)\right)^{-\sigma}}-\beta S_j(1+r)\right\}=0 \tag{8}$$

  I will work with the "empirical" analogy of equation 8 to compute moments per age.
- In order to solve this problem, I set the following:
  - Lifetime's duration is J years.
  - Given that I have to "[k]ill individuals as they age", the survival probability across ages is now defined as S^j, where $j \in [1, J-1]$.
  - I will get as many policy functions as J years. Then, $\hat{a}'(a)$ is now a matrix with dimensions $n_{aa} \times J$.
  - Since equation 8 is the inter-temporal condition (relation between two adjacent periods) then I will able to compute $J-1$ moments.

Exercise 3.1: Euler equation to write a set of moment conditions

- If we preserve M individuals after discarding constrained agents, then the matrix of $J-1$ moments is defined as follows:

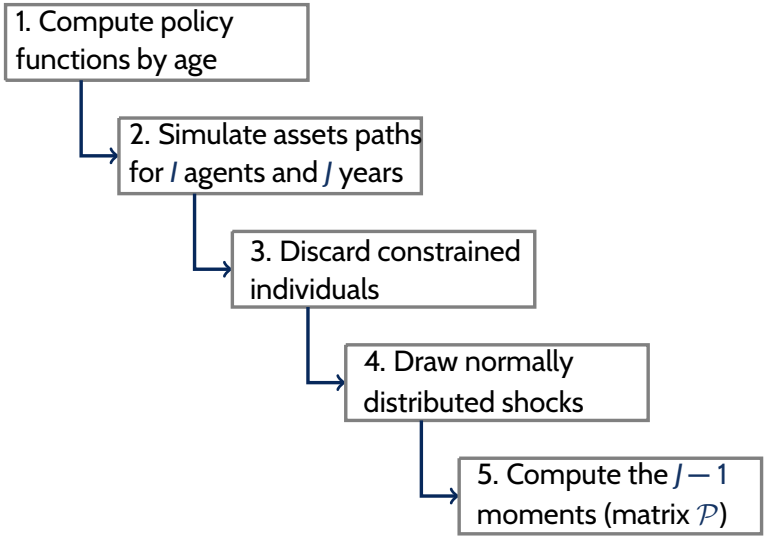
$$\mathcal{P}_{(J-1)\times 1} = \begin{bmatrix} \frac{1}{M}\sum_{m=1}^{M}\left\{\frac{(a_{m,1}-a_{m,2})^{-\sigma}}{\left((1+r)a_{m,2}+exp(\varepsilon_{m,1})\right)^{-\sigma}} - \beta S(1+r)\right\} \\ \frac{1}{M}\sum_{m=1}^{M}\left\{\frac{(a_{m,2}-a_{m,3})^{-\sigma}}{\left((1+r)a_{m,3}+exp(\varepsilon_{m,2})\right)^{-\sigma}} - \beta S^2(1+r)\right\} \\ \vdots \\ \frac{1}{M}\sum_{m=1}^{M}\left\{\frac{(a_{m,J-1}-a_{m,J})^{-\sigma}}{\left((1+r)a_{m,J}+exp(\varepsilon_{m,J-1})\right)^{-\sigma}} - \beta S^{J-1}(1+r)\right\} \end{bmatrix} \tag{9}$$

- The criterion to minimize is defined as follows:

$$\mathcal{C} = \mathcal{P}' \times \mathcal{W} \times \mathcal{P} \tag{10}$$

Exercise 3.1: Euler equation to write a set of moment conditions

Pseudo-code



Exercise 3.1: Euler equation to write a set of moment conditions

Pseudo-code

1. Compute the assets policy function per age. Now we have to include an external loop over steps 4 and 5 if we use the Collocation/Dirac Delta (from exercise 1.1), or over steps 5 and 6 if we apply the Least Squares Projection (from exercise 1.2). For the latter case:

```
g_a=zeros(n_aa,J);
G1=zeros(n_aa,J); G2=zeros(n_aa,J); G3=zeros(n_aa,J);
for jj=1:J-1
     % run steps 5 and 6 from exercise 1.2
     % get the policy functions per age:
     % g_a = G1+G2.*a_grid+G3.*a_grid.^2
     % where G1, G2 and G3 are (1 x J) matrices
     % with optimal coefficients per age
     % and a_grid is defined in step 2 from exercise 1.2
end
```

Exercise 3.1: Euler equation to write a set of moment conditions

Pseudo-code

2. Simulate assets trajectory for $I = 1000$ individuals and J years:

```
for ii=1:I
    for jj=1:J
        if jj==1
            a(ii,jj)=interp1(a_grid,g_a(:,jj),...
            a_grid(randi(1,n_GC),'linear','extrap');
        else
            a(ii,jj)=interp1(a_grid,g_a(:,jj),...
            a(ii,jj-1),'linear','extrap');
        end
    end
end
```

3. Discard those individuals who are constrained. Assume we preserve M individuals.
4. Draw normally distributed shocks per individual and age:

```
epsilon=normrnd(0,sigma_e,M,J-1);
```

Note epsilon is $M \times J - 1$ matrix.

Exercise 3.1: Euler equation to write a set of moment conditions

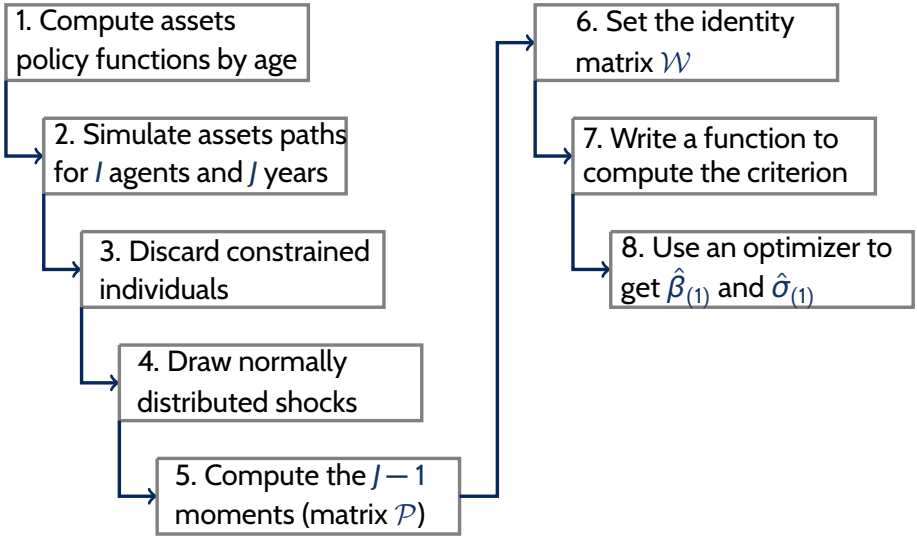
Pseudo-code

5. Compute the $J - 1$ moments (equation 9). For instance, we can use the following function:

```
function P=Pfun(a,epsilon,b_ta,sigma,S,r)
[M,J]=size(a);
for jj=1:J-1
 for mm=1:M
    Mat(mm,jj)=(a(mm,jj)-a(mm,jj+1))^(-sigma)/(((1+r)*...
    a(mm,jj+1)+exp(epsilon(mm,jj)))^(-sigma))-b_ta*S^(jj)*(1+r);
 end
end
aux=mean(Mat); P=aux';
```

Exercise 3.2: Estimate the discount factor and risk aversion coefficient

Pseudo-code: identity matrix



Exercise 3.2: Estimate the discount factor and risk aversion coefficient

Pseudo-code: identity matrix

6. Set the weighting matrix $\mathcal{W}$:

```
W=eye(M,J-1);
```

7. Write a function to compute the criterion:

```
function C=criterion(H,a,epsilon,S,r,W)
b_ta=H(1); % discount factor
sigma=H(2); % risk aversion coefficient
P=Pfun(a,epsilon,b_ta,sigma,S,r); % matrix from 9
C=P'*W*P; % compute the criterion
```

8. Use an optimizer to get values of $\hat{\beta}_{(1)}$ and $\hat{\sigma}_{(1)}$ that minimize the criterion. Take into account reasonable values for both parameters. For instance:

$$\hat{\beta}_{(1)} \in [0.5, 1] \quad \text{and} \quad \hat{\sigma}_{(1)} \in [0, 3]$$

An option is `fmincon`:

```
fun=@(H) criterion(H,a,epsilon,S,r,W); x0=[0.96,1];
Hhat=fmincon(fun,x0,[],[],[],[],[0.5 0],[1 3],[],options);
beta_hat1=Hhat(1); sigma_hat1=Hhat(2);
```

Exercise 3.2: Estimate the discount factor and risk aversion coefficient

Pseudo-code: identity matrix

8. (Continued) Another option is to use a two-step algorithm: running a global search, then use an optimizer like `fminsearch` using the previous results as initial points:

```
% First step
b_ta=linspace(0.5,1,1000);
sigma=linspace(0,3,1000);
for i1=1:length(b_ta)
    for i2=1:length(sigma)
        H(1)=b_ta(i1);
        H(2)=sigma(i2);
        C(i1,i2)=criterion(H,a,epsilon,S,r,W);
    end
end
[i1,i2]=find(C==min(C(:)));
H0=[b_ta(i1) sigma(i2)];
% Second step
fun=@(H) criterion(H,a,epsilon,S,r,W);
Hhat=fminsearch(fun,H0,options);
beta_hat1=Hhat(1); sigma_hat1=Hhat(2);
```

Exercise 3.2: Estimate the discount factor and risk aversion coefficient

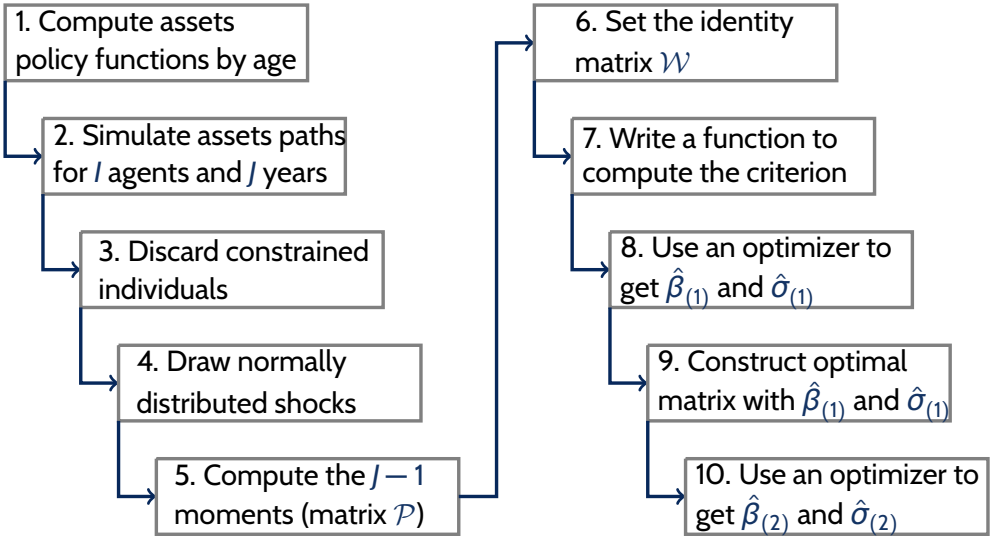
Pseudo-code: optimal matrix

- Use previous estimation of $\hat{\beta}_{(1)}$ and $\hat{\sigma}_{(1)}$ to construct the optimal weighting matrix:

$$W_{opt} = \frac{1}{M}\sum_{m=1}^{M} \begin{bmatrix} \left\{ \frac{(a_{m,1}-a_{m,2})^{-\hat{\sigma}_{(1)}}}{\left((1+r)a_{m,2}+exp(\varepsilon_{m,1})\right)^{-\hat{\sigma}_{(1)}}} - \hat{\beta}_{(1)} S(1+r) \right\} \\ \left\{ \frac{(a_{m,2}-a_{m,3})^{-\hat{\sigma}_{(1)}}}{\left((1+r)a_{m,3}+exp(\varepsilon_{m,2})\right)^{-\hat{\sigma}_{(1)}}} - \hat{\beta}_{(1)} S^2(1+r) \right\} \\ \vdots \\ \left\{ \frac{(a_{m,J-1}-a_{m,J})^{-\hat{\sigma}_{(1)}}}{\left((1+r)a_{m,J}+exp(\varepsilon_{m,J-1})\right)^{-\hat{\sigma}_{(1)}}} - \hat{\beta}_{(1)} S^{J-1}(1+r) \right\} \end{bmatrix} \times \begin{bmatrix} \left\{ \frac{(a_{m,1}-a_{m,2})^{-\hat{\sigma}_{(1)}}}{\left((1+r)a_{m,2}+exp(\varepsilon_{m,1})\right)^{-\hat{\sigma}_{(1)}}} - \hat{\beta}_{(1)} S(1+r) \right\} \\ \left\{ \frac{(a_{m,2}-a_{m,3})^{-\hat{\sigma}_{(1)}}}{\left((1+r)a_{m,3}+exp(\varepsilon_{m,2})\right)^{-\hat{\sigma}_{(1)}}} - \hat{\beta}_{(1)} S^2(1+r) \right\} \\ \vdots \\ \left\{ \frac{(a_{m,J-1}-a_{m,J})^{-\hat{\sigma}_{(1)}}}{\left((1+r)a_{m,J}+exp(\varepsilon_{m,J-1})\right)^{-\hat{\sigma}_{(1)}}} - \hat{\beta}_{(1)} S^{J-1}(1+r) \right\} \end{bmatrix}' \quad (11)$$

Exercise 3.2: Estimate the discount factor and risk aversion coefficient

Pseudo-code: optimal matrix



Exercise 3.2: Estimate the discount factor and risk aversion coefficient

Pseudo-code: optimal matrix

9. Construct the optimal weighting matrix with $\hat{\beta}_{(1)}$ and $\hat{\sigma}_{(1)}$:

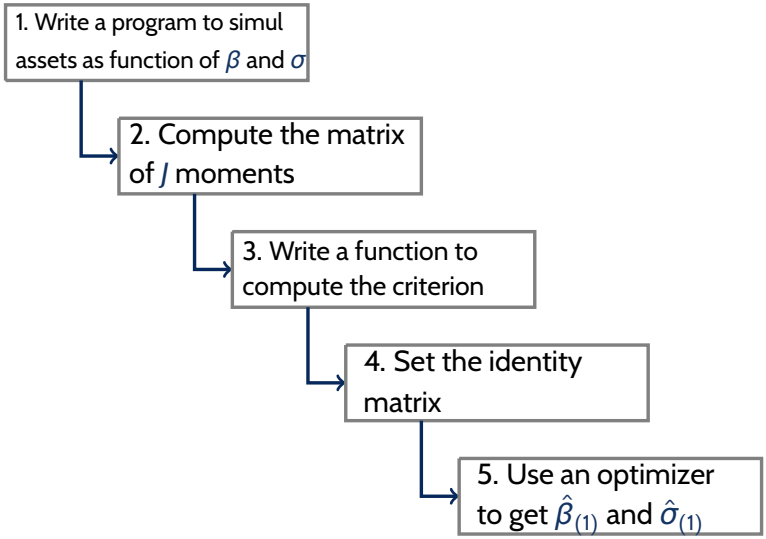
```
P=Pfun(a,epsilon,beta_hat1,sigma_hat1,S,r);
Wopt=P*P';
```

10. Use an optimizer to get values of $\hat{\beta}_{(2)}$ and $\hat{\sigma}_{(2)}$ that minimize the criterion. Same as step 3 from the previous algorithm but now using $\mathcal{W}_{opt}$ instead of $\mathcal{W}$. For instance, with `fmincon`:

```
fun=@(H) criterion(H,a,epsilon,S,r,Wopt); x0=[0.96,1];
Hhat=fmincon(fun,x0,[],[],[],[],[0.5 0],[1 3],[],options);
beta_hat2=Hhat(1); sigma_hat2=Hhat(2);
```

Exercise 3.3: Estimate the discount factor and risk aversion coefficient by matching the average wealth of individuals by age

Pseudo-code: identity matrix



Exercise 3.3: Estimate the discount factor and risk aversion coefficient by matching the average wealth of individuals by age

Pseudo-code: identity matrix

1. Write a program to compute assets trajectories as a function of β and σ:

```
function a=asimul(b_ta,sigma);
a_min=1; a_max=20; p=3; n_GH=1000;...
% Construct a grid of assets (with Chebyshev zeros)
for ii=1:p
   a_grid(ii)=((a_max-a_min)*cos((2*ii-1)/(2*p)*pi)+...
   (a_max+a_min))/2;
end
% Gauss-Hermite approximation
[x,w]=lgwt(n_GH,-1.96*sigma_e,1.96*sigma_e);
% Get coefficients by Least-Squares (per age)
for jj=1:J-1
 S=Sj^jj;
 fun=@(G) Integral(G,a_grid,n_GC,sigma,r,sigma_e,b_ta,S,...)
 Ghat=fminsearch(fun,G0,options);
 g1(1,jj)=Ghat(1);g2(1,jj)=Ghat(2);g3(1,jj)=Ghat(3);
end
```

Exercise 3.3: Estimate the discount factor and risk aversion coefficient by matching the average wealth of individuals by age

Pseudo-code: identity matrix

1. (Continued) Write a program to compute assets trajectories as a function of β and σ:

```
% Construct policy functions
g_a=g1+g2.*a_grid+g3.*a_grid.^2;
% Simulate assets for I agents and J years
for ii=1:I
    for jj=1:J
        if jj==1
            a(ii,jj)=interp1(a_grid,g_a(:,jj),...
            a_grid(randi(1,n_GC),'linear','extrap');
        else
            a(ii,jj)=interp1(a_grid,g_a(:,jj),...
            a(ii,jj-1),'linear','extrap');
        end
    end
end
```

Exercise 3.3: Estimate the discount factor and risk aversion coefficient by matching the average wealth of individuals by age

Pseudo-code: identity matrix

2. Compute the J moments:

$$\mathcal{P}_{J\times 1}=\begin{bmatrix} \frac{1}{I}\sum_{i=1}^{I} a_{i,1} \\ \frac{1}{I}\sum_{i=1}^{I} a_{i,2} \\ \vdots \\ \frac{1}{I}\sum_{i=1}^{I} a_{i,J} \end{bmatrix} \tag{12}$$

For instance:

```
P=(mean(a))';
```

3. Write a function to compute the criterion given values of β and σ:

```
function C=criterion(param,W)
b_ta=param(1); sigma=param(2);
a=asimul(b_ta,sigma);
P=(mean(a))';
C=P'*W*P;
```

Exercise 3.3: Estimate the discount factor and risk aversion coefficient by matching the average wealth of individuals by age

Pseudo-code: identity matrix

4. Set the identity matrix:

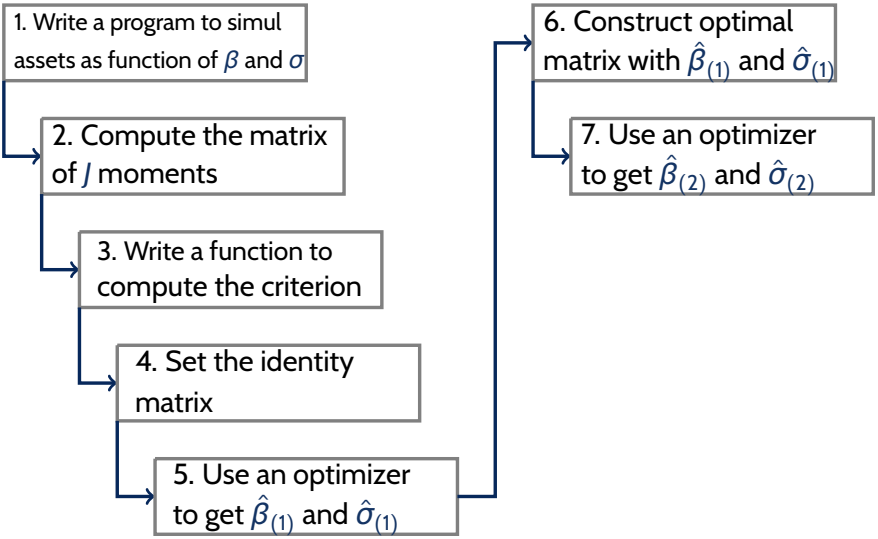
```
W=eye(I,J)
```

5. Use an optimizer to get values of $\hat{\beta}_{(1)}$ and $\hat{\sigma}_{(1)}$ that minimize the criterion. For instance, with `fmincon`:

```
fun=@(param) criterion(param,W); x0=[0.96,1];
paramhat=fmincon(fun,x0,[],[],[],[],[0.5 0],[1 3],[],options);
beta_hat1=paramhat(1); sigma_hat1=paramhat(2);
```

Exercise 3.3: Estimate the discount factor and risk aversion coefficient by matching the average wealth of individuals by age

Pseudo-code: optimal matrix



Exercise 3.3: Estimate the discount factor and risk aversion coefficient by matching the average wealth of individuals by age

Pseudo-code: optimal matrix

6. Construct an optimal matrix using $\hat{\beta}_{(1)}$ and $\hat{\sigma}_{(1)}$. It implies to simulate assets with those values and compute the matrix from equation 12:

```
a=asimul(beta_hat1, sigma_hat1);
P=(mean(a))    ;
Wopt=P*P';
```

7. Use an optimizer to get values of $\hat{\beta}_{(2)}$ and $\hat{\sigma}_{(2)}$ that minimize the criterion. For instance, with `fmincon`:

```
fun=@(param) criterion(param,Wopt); x0=[0.96,1];
paramhat=fmincon(fun,x0,[],[],[],[],[0.5 0],[1 3],[],options);
beta_hat2=paramhat(1); sigma_hat2=paramhat(2);
```