```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ÉQUILIBRE DU MODÈLE D'INDUSTRIE %
%      AVEC SORTIES EXOGÈNES.     %
%                                 &
% Basé sur les acétates de cours. %
% Ecriture matricielle compacte.  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear; clc


% PARAMÉTRISATION %
%%%%%%%%%%%%%%%%%%

% Préférences
A = 2.14;       % utilité u = ln(x) -  A.N
beta = 1/1.04;  % taux d'escompte

% Technologie
theta = 0.64;             % part du capital

% Politiques économiques
tau = 0.9;  seuil = 15;   % coûts de licenciement
ce = 48.9;                % coûts d'entrée

% Processus stochastiques
nu = [0.314; 0.514; 0.140; 0.030; 0; 0; 0; 0.0042; 0.0018; 0; 0];  % entrants

Q = [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0;          % firmes en place
    .1, .8, .1, 0, 0, 0, 0, 0, 0, 0, 0;
    .1, .09, .75, .06, 0, 0, 0, 0, 0, 0, 0;
    .05, 0, .115, .75, .085, 0, 0, 0, 0, 0, 0;
    .03, 0, 0, .125, .75, .095, 0, 0, 0, 0, 0;
    .03, 0, 0, 0, .125, .75, .095, 0, 0, 0, 0;
    .03, 0, 0, 0, 0, .125, .75, .095, 0, 0, 0;
    .02, 0, 0, 0, 0, 0, .13, .75, .1, 0, 0;
    .02, 0, 0, 0, 0, 0, 0, .13, .75, .1, 0;
    .02, 0, 0, 0, 0, 0, 0, 0, .13, .75, .1;
    .015, 0, 0, 0, 0, 0, 0, 0, 0, .185, .8];

% Paramètres IFV
Nn = 222;
taille_min = 1; taille_max = 5000; bornes = [taille_min; taille_max];
bornes = log(bornes) / log(10); n = logspace(bornes(1), bornes(2), Nn-1);
n = [0,n]';          % Nn x 1    grille d'emploi

Ns = 11;
s = [0; 2.4; 4.1; 5; 5.6; 6.6; 7; 8; 12.5; 15; 27.5];  % grille des chocs

% Remarque sur la calibration des processus stochastiques (et de s):
% Faite par tâtonnement. Peut être de façon plus systématique.
% L'idée est de mimiquer (i) la distribution de la taille des firmes,
```

```
% (ii) la contribution des différentes catégories de taille à l'emploi,
% (iii)la distribution de taille parmi les entrants et (iv) les taux de
% sortie suivant la taille des firmes. Ces données peuvent se trouver.
% Le processus est supposé persistent et les sauts de productivité
% majoritairement modérés.

tolerance = 1e-4;                                          % convergence

w_inf_init = 0.9;  w_sup_init = 1.1;                       % initialisation


% ETAPE 1: PROBLÈME DE LA FIRME ET LIBRE ENTRÉE %
%          Détermination de V, Ve, w et nd      %

% Préliminaires %
production = n.^theta * s';  % Nn x Ns

travail = repmat(n,[1,Nn]);                                % Nn x Nn
couts_licenciement = travail - travail';                   % Nn x Nn
couts_licenciement = tau * max(couts_licenciement,0);      % Nn(-1) x Nn
sans_cout = find(travail < seuil);  couts_licenciement(sans_cout) = 0;
couts_licenciement = repmat(couts_licenciement,[1,1,Ns]);  % Nn(-1) x Nn x Ns
couts_licenciement = permute(couts_licenciement,[1,3,2]);  % Nn(-1) x Ns x Nn

% Itération sur la condition de libre entrée %
w_inf = w_inf_init;  w_sup = w_sup_init;

w = w_inf;
compteur_w = 1;
libre_entree = 1;
while abs(libre_entree) > tolerance
    % Début du bloc de réactualisation de w %
    if compteur_w == 2
        w1 = w;
        libre_entree_1 = libre_entree;
        w = w_sup;
    elseif compteur_w >= 3
        pente = (libre_entree_1 - libre_entree) / (w1 - w);
        intersection = libre_entree_1 - pente * w1;
        if abs(libre_entree) < abs(libre_entree_1)
            w1 = w;
            libre_entree_1 = libre_entree;
        end
        w = - intersection / pente;
    end
    % Fin du bloc de réactualisation de w %

    % Initialisation de V (première itération sur w) %
    if compteur_w == 1
        V = production - repmat(w*n,[1,Ns]);
        V = V / (1-beta);  % Nn(-1)  x Ns
    end
```

```
    % Itération sur la fonction de valeur
    produit_net = production - repmat(w*n,[1,Ns]);  % Nn x Ns
    produit_net = repmat(produit_net,[1,1,Nn]);     % Nn x Ns x Nn(-1)
    produit_net = permute(produit_net,[3,2,1]);     % Nn(-1) x Ns x Nn

    deviation_V = 1;
    while max(abs(deviation_V(:))) > tolerance
        valeur_esperee = repmat(V*Q',[1,1,Nn]);            % Nn x Ns x Nn(-1)
        valeur_esperee = permute(valeur_esperee,[3,2,1]);  % Nn(-1) x Ns x Nn
        Bellman = produit_net - w * couts_licenciement ...
                        + beta * valeur_esperee;           % Nn(-1) x Ns x Nn

        TV = max(Bellman,[],3);  % Nn(-1) x Ns

        deviation_V = TV - V;  % Nn(-1) x Ns
        V = TV;                % Nn(-1) x Ns
    end

    Ve = sum(V(1,:) .* nu');  % 1 x 1  % n(1) = 0 ou 1??
    libre_entree = beta * Ve - ce;

    compteur_w = compteur_w + 1;
end

disp('Boucles sur V et w terminées.')

% Règles de décision %
[V, nd_index] = max(Bellman,[],3);  % Nn(-1) x Ns
nd = n(nd_index);                   % Nn(-1) x Ns

message = ['Règle de décision obtenue.' newline 'Fin étape #1.'];  disp(message)


% ETAPE 2: DÉRIVATION DE L'ÉTAT STATIONNAIRE                              %
%          Détermination de mu et des agrégats, au nombre d'entrants près %

% Nouvelle formulation de la règle de décision
% Si i = n(-1) et k = s, index de nd(n(-1),s) = j? oui = 1, non = 0
indices = repmat(1:Nn,[Nn,1,Ns]);

matrice = repmat(nd_index,[1,1,Nn]);  % Nn(-1) x Ns x Nn
matrice = permute(matrice,[1,3,2]);   % Nn(-1) x Nn x Ns
regle_emploi = (indices == matrice);  % Nn(-1) x Nn x Ns

% Itération vers l'état stationnaire (mesure par entrant = 1)
% Initialisation de mu %
mu = ones(Nn,Ns)/(Nn*Ns);  % Nn(-1) x Ns

% Itération sur mu
mudeviation = 1;
 while max(abs(mudeviation(:))) > tolerance
    mu_plus_entree = mu + cat(1,nu',zeros(Nn-1,Ns));        % Nn(-1) x Ns
```

```
    % Les firmes font leur choix d'emploi.
    %
    mu_plus_entree(:,1) = 0;
    %
    mutilde = repmat(mu_plus_entree,[1,1,Nn]);              % Nn(-1) x Ns x Nn
    mutilde = permute(mutilde,[1,3,2]);                     % Nn(-1) x Nn x Ns
    mutilde = regle_emploi .* mutilde;
    mutilde = sum(mutilde,1);  mutilde = squeeze(mutilde);  % Nn x Ns

    % Les firmes tirent une nouvelle productivité
    muprime = mutilde * Q;  % Nn x Ns'

    mudeviation = muprime - mu;
    mu = muprime;  % Nn(-1) x Ns = Nn x Ns'
 end

espace = ' ';
message = [espace newline 'Boucle sur mu par entrant terminée.'];  disp(message)

% Agrégats par entrant %
offre_bien_agregee_entrant = s' .* nd.^theta;                                % Nn↙
(-1) x Ns
offre_bien_agregee_entrant = mu_plus_entree .* offre_bien_agregee_entrant;  % Nn↙
(-1) x Ns
offre_bien_agregee_entrant = sum(offre_bien_agregee_entrant(:));

transferts_agreges_entrant = tau * w * max(repmat(n,[1,Ns]) - nd, 0);  % Nn(-1) x↙
Ns
transferts_agreges_entrant = mu .* transferts_agreges_entrant;         % Nn(-1) x↙
Ns
transferts_agreges_entrant = sum(transferts_agreges_entrant(:));

profits_agreges_entrant = s' .* nd.^theta - w * nd;                   % Nn(-1) x Ns
profits_agreges_entrant = mu_plus_entree .* profits_agreges_entrant;  % Nn(-1) x Ns
profits_agreges_entrant = sum(profits_agreges_entrant(:));
profits_agreges_entrant = profits_agreges_entrant ...
                                - transferts_agreges_entrant - ce;

demande_travail_agregee_entrant = mu_plus_entree .* nd;                    % Nn(-1)↙
x Ns
demande_travail_agregee_entrant = sum(demande_travail_agregee_entrant(:));

message = ['Agrégats par entrant obtenus.' newline 'Fin étape #2.'];  disp(message)


% ETAPE 3: PROBLÈME DU MÉNAGE ET ÉQUILIBRAGE DES MARCHÉS     %
%          Détermination de x et de M, de mu et des agrégats %

x = w/A;
M = x / (offre_bien_agregee_entrant - ce);

offre_bien_agregee = M * offre_bien_agregee_entrant;
transferts_agreges = M * transferts_agreges_entrant;
```

```
profits_agreges = M * profits_agreges_entrant;
demande_travail_agregee = M * demande_travail_agregee_entrant;

message = [espace newline 'Consommation, entrée et agrégats obtenus.' ...
                      newline 'Fin étape #3.' newline espace];  disp(message)


% AUTRES VALEURS D'ÉQUILIBRE %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Valeurs moyennes
taille_moyenne = demande_travail_agregee_entrant / sum(mutilde(:));
productivite_moyenne = offre_bien_agregee_entrant /↙
demande_travail_agregee_entrant;

% Distribution de la taille des firmes
bornes = [0; 5; 10; 20; 50; 100; 250; 500; 1000; 2500; 5000];

firmes = M * sum(mutilde,2);  % Nn x 1

for i = 1:9
    selection = (n >= bornes(i) & n < bornes(i+1));
    mesure_firmes(i) = sum(selection .* firmes);
end

selection = (n >= bornes(10) & n <= bornes(11));
mesure_firmes(10) = sum(selection .* firmes);

distribution_firmes = mesure_firmes /sum(mesure_firmes);

% Destruction d'emploi
destruction_emploi = repmat(n,[1,Ns]) - nd;              % Nn x Ns
destruction_emploi = max(0, destruction_emploi);         % Nn x Ns
destruction_emploi = M * mu .* destruction_emploi;  % Nn x Ns
taux_destruction_emploi = sum(destruction_emploi(:)) ./ demande_travail_agregee;

% Taux d'emploi
% Choix de h1=0.5: huit heures sur un temp disctétionnaire quotidien de 16 heures.
h1=.5;
taux_emploi = demande_travail_agregee / h1;


% PRÉSENTATION DES RÉSULTATS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Politiques économiques
message = [espace newline 'coûts de licenciement (mois): ', num2str(12*tau) ...
                      newline 'seuil (employés): ', num2str(seuil) ...
                      newline 'coûts entrée: ', num2str(ce)];  disp(message)

% Ménages
message = [espace newline 'consommation: ', num2str(x) ...
                  newline 'salaire réel: ', num2str(w)];  disp(message)
```

```
% Agrégé
message = [espace newline 'production agrégée: ', num2str(offre_bien_agregee) ...
                  newline 'productivité moyenne: ', num2str(productivite_moyenne)↙
...
                  newline 'taille moyenne (employés): ', num2str(taille_moyenne)];↙
disp(message)

% Marché de l'emploi
message = [espace newline 'taux emploi (%): ', num2str(100*taux_emploi) ...
                  newline 'taux de destruction emploi (%): ', num2str↙
(100*taux_destruction_emploi) ...
                  newline 'entrée: ', num2str(M)];  disp(message)

% message = [espace newline 'distribution de la taille des firmes (%):'];  disp↙
(message)
% distribution = 100 * distribution_firmes'
% disp(espace)

message = [espace newline 'distribution de la taille des firmes (%):'];  disp↙
(message)
classes = ['0 à 5      '; '5 à 10     '; '10 à 20    '; '20 à 50    '; '50 à 100↙
'; '100 à 250  '; '250 à 500  '; '500 à 1000 '; '1000 à 2500'; '2500 à 5000'];
classes = string(classes);  distribution = num2str(100*distribution_firmes');
%tableau =
[classes, distribution]
disp(espace)
```