Parts (d) and (e)

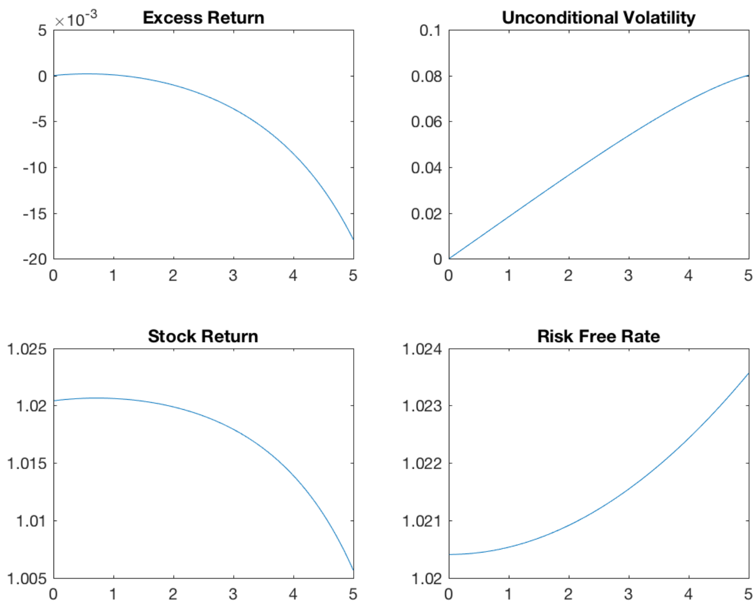
The table below reports the obtained values in the different specifications:

	R^f	q	ER	$ER - R^f$
μ_h	1.0576	49.404	1.0564	−0.0012268
μ_l	0.98418	67.155	0.98334	−0.0008398
Unconditional	1.0209	58.28	1.0199	−0.0010333

Part (f)

From Figure 1 we can clearly see that as γ increases the expected excess return, the expected return of the stock decrease while unconditional volatility and the risk-free rate increase.

Figure 1: Excess Return, Unconditional Volatility, Stock Return, Risk Free Rate for different γ



Code

```
%% Macroeconomics II - Part II %%
% Problem Set 1
% Gualtiero Azzalini
addpath('/Users/Gualtiero/Dropbox/A NYU/Macro/gilchrist/psets/1')
addpath('/Users/Gualtiero/Documents/MATLAB/Add-Ons/mfe_toolbox/utility')
clear all;
%% Exercise 2 - Computational Part

% Set Parameters %
```

```
beta   = 0.98;
gamma = 2;
S      = 2;

% Number of states

mu     = zeros(S,1);    % Vector of states
Q      = zeros(S,S);    % Transition matrix

% Define states and transition probabilities

state1 =  0.02;     % High state
state2 = -0.02;     % Low state
p11    =  0.95;
p12    =  1-p11;
p22    =  p11;
p21    =  1-p22;

% Filling matrices
for k=1:S
        mu(k,1) = eval(['state',num2str(k)]);
end
for j=1:S
        for i=1:S
                Q(i,j) = eval(['p',num2str(i),num2str(j)]);
        end
end
for k=1:500
        gamma    = k*0.01;
        % Compute risk-free return - this is contingent on state at t so # is S
        R_f = zeros(S,1);
        Id  = eye(S,S);
        R_f = beta^(-1)*((exp(-gamma*mu')*Q')').^(-1);
        % Compute price-dividend ratio
        A = beta*Q*(Id.*(exp((1-gamma)*mu)));
        % Matrix of function of states
        B = beta*Q*(exp((1-gamma)*mu));
        % Matrix of constants
        q = inv(Id-A)*B;
        % Compute expected return
```

```
        Pd_inv = Id.*q.^(-1);
        Pd_1   = Id.*(q+ones(S,1));
        R      = Pd_inv*Q'*Pd_1*exp(mu);
        % Compute excess return
        RE = R-R_f;
        % Compute invariant distribution
        [V,D]      = eig(Q');
        [foo , tp] = sort(diag(D));
        PI         = (V(: , tp(end))/sum(V(: , tp(end))));
        % Compute unconditional expected return and excess return
        R_f_unc = PI'*R_f;
        q_unc   = PI'*q;
        R_unc   = PI'*R;
        RE_unc  = R_unc-R_f_unc;
        % Unconditional variance
        Var_unc = sqrt(PI'*(R.^(2))-R_unc^(2));
        % Filling containers
        R_f_it(k,1) = R_f_unc;
        q_it(k,1)   = q_unc;
        R_it(k,1)   = R_unc;
        RE_it(k,1)  = R_unc-R_f_unc;
        Var_it(k,1) = Var_unc;

        if k==200
                table(R_f,q,R,RE)
                table(R_f_unc,q_unc,R_unc,RE_unc)
        end
end
figure(1)
subplot(2,2,1);plot(linspace(0.01,length(R_f_it)/100,length(R_f_it)),RE_it);
title('ExcessReturn');
subplot(2,2,2);plot(linspace(0.01,length(R_f_it)/100,length(R_f_it)),Var_it);
title('UnconditionalVolatility');
subplot(2,2,3);plot(linspace(0.01,length(R_f_it)/100,length(R_f_it)),R_it);
title('StockReturn');
subplot(2,2,4);plot(linspace(0.01,length(R_f_it)/100,length(R_f_it)),R_f_it);
title('RiskFreeRate');
saveas(figure(1),'unconditional.png');
```