First and foremost, the lecture notes will be my own as well as the errors. The first part will cover the RBC model and the New Keynesian model and several numerical methods on how to solve them.

1 Part 1

1.1 Lecture 1: Basic RBC Model

The RBC model falls into new classical macroeconomics, where people have:

1. Rational preferences
2. Maximize Utility and firms maximize profits
3. People act independently on full (and assumed) relevant information

From the name, it can be deduced that this model is used to measure business cycle fluctuations - movements which are done by real (instead of nominal) shocks. These models are considered real because they do not represent market failures, but rather it reflects the most efficient possible operation of the economy, given the structure of the economy. It rejects the real effectiveness of monetary policy. New classicals do not want the government to intervene in the short-run, and would rather have the government focus on long-run structural policies. They believe only technological shocks can influence and change the decisions of all factors in an economy. Innovations, bad weather, oil price increase, regulations, etc are examples. Pretty much a productivity shock is anything that affects the effectiveness of capital and labor.

The model begins two components to keep track of:

1. The household problem $\max_{c_t,k_{t+1},l_t} E_0 \sum_{t=0}^{\infty} \beta^t \left(\frac{c_t^{1-\sigma}-1}{1-\sigma} + \psi \log(1-l_t) + \theta g_t\right)$
2. The firm problem $\max \pi = \max_{k_t,l_t} Ae^{z_t} k_t^{\alpha} l_t^{1-\alpha} - w_t l_t - R_t k_t$

1.1.1 The Household Problem

The productivity shock equations (stochastic variables $\hat{g}_t$ and z_t). The resource constraint: $c_t + i_t = w_t l_t + R_t k_t - T$ where $\gamma_x k_{t+1} = (1-\delta)k_t + i_t$ Taking FOC conditions of the utility function with respect to c_t, k_{t+1}, l_t gets us the Euler and the labor supply equation:

$$c_t^{-\sigma} = \frac{\beta}{\gamma_x} c_{t+1}^{-\sigma}(1 + r_{t+1})$$

$$\frac{\psi}{1-l_t} = c_t^{-\sigma} w_t \;\; or \; w_t = \frac{-u_l}{u_c}$$

1.1.2 The Firm Problem

The firms problem is the production function minus the costs to production. They will maximize profits with respect to labor and capital. They find the optimal wage and rental price, which shows that the factors are paid the marginal products.

$$w_t = (1-\alpha)Ae^{z_t} k_t^{\alpha} l_t^{-\alpha}$$

$$R_t = \alpha Ae^{z_t} \left(\frac{k_t}{l_t}\right)^{\alpha-1}$$

1.1.3 Variable Descriptions

The level of Productivity, A_t, is exogenous. The level for g_t and thus, $\hat{g}$ is exogenous. This is analogous for z_t. The level of initial capital, k_0, we choose, thus is exogenous.
The level of capital is determined by the model, thus is endogenous, k_t. We want to maximize over consumption, capital tomorrow and labor, c_t, k_{t+1}, l_t, thus they are our control variables.
The rental price, R_t, price of labor w_t and taxes T_t are exogenous.

1.1.4 Market Clearing

There are three markets:

1. The Labor market (wages are just plugged into the labor equation). $\frac{\psi c_t^\sigma}{1-l_t} = (1-\alpha)Ae^{z_t}k_t^\alpha l_t^{-\alpha}$
2. The capital market (the rental price of capital is just plugged into the Euler).
$$c_t^{-\sigma} = \frac{\beta}{\gamma_x} c_{t+1}^{-\sigma}(\alpha A e^{z_{t+1}} \left(\frac{k_{t+1}}{l_{t+1}}\right)^{\alpha-1} + 1 - \delta)$$
3. The goods market ($c + i + g = Y$).
$$c_t + \gamma_k k_{t+1} - k_t(1-\delta) + g_t e^{\hat{g}_t} = Ae^{z_t}k_t^\alpha l_t^{1-\alpha}$$
4. Just a note: these three expressions and the exogenous processes of $\hat{g}_t$ and z_t make up the system of equations.

In solving this model, we have to log-linearize. The initial condition is thus important. Otherwise, we would have a global solution, but this is difficult to attain non-numerically. The local approximation around the steady state is accurate, and robust to the curse of dimensionality (meaning, I can use multiple state variables without any complications). Log linearization is just the total differential of each expression (i.e. the ones that clear the market). We attain an expression for c_t (forward looking) and k_{t+1} (backward looking). Putting the expressions together leads to the backward looking matrix formulation. The point, though, is to find the policy function for your control variables c_t and k_{t+1}, which we do throughout the course.

$$\hat{c}_t = \phi_{ck}\hat{k}_t + \phi_{cz}z_t + \phi_{cg}\hat{g}_t$$
$$\hat{k}_{t+1} = \phi_{kk}\hat{k}_t + \phi_{kz}z_t + \phi_{kg}\hat{g}_t$$

1.2 Lecture 2

The RBC model is a real business cycle model; thus it does not feature monetary shocks. This is known as monetary neutrality. **The basic New Keynesian Model** features money non-neutrality because of the stickiness in prices. Nominal shocks have strong and persistent effects. Therefore, RBC models have this **price flexibility**. A nominal shock occurs then prices adjust (i.e. wages for example). This is obviously not the case.
To model the price stickiness:

1. Imperfect competition: if you have i firms selling good i, then we have an economy of price setters.
2. For the sake of the model, they also use a la calvo pricing (1983) to define the model; where some firms have the right to reset their price and others don't. The implication is that when the firm is given that right to change prices, it also has to acknowledge that it could be some time before it changes its price again.

1.2.1 The model: Households

We have a continuum of firms and infinitely lived households, hence the integrals and the index $i \in [0,1]$. Households maximize their discounted sum of utility, where consumption takes on a different form. The constraint is the resource constraint. Optimality conditions are founded under the CRRA utility function, where as σ increases, the more risk averse you are. This is the process:
Note 1: The exponent is always the inverse of the outer exponent

$$E_0 \sum_{t=0}^{\infty} \beta^t U(C_t, N_t) \quad C_t = \left(\int_0^1 C_t(i)^{1-\frac{1}{\varepsilon}} di\right)^{\frac{\varepsilon}{\varepsilon-1}}$$

Note 2: What consumers buy today in goods and bonds cannot exceed the wages, dividends they earned and the bonds they bought last period.

$$\text{Budget Constraint} \rightarrow \int_0^1 P_t(i)C_t(i)di + Q_t B_t \leq B_{t-1} + W_t N_t + D_t$$

The resulting optimality conditions given the price level:

$$P_t = \left(\int_0^1 P_t(i)^{1-\varepsilon} di\right)^{\frac{1}{1-\varepsilon}}$$

is

$$C_t(i) = \left(\frac{P_t(i)}{P_t}\right)^{-\varepsilon} C_t$$

$$\frac{-U_n}{U_c} = \frac{w_t}{P_t} = \text{slope of the indifference curve} = \text{Real wage}$$

$$Q_t = \beta E_t \left(\frac{U_{c,t+1} P_t}{U_{c,t} P_{t+1}}\right)$$

Applying the log to this expression, leads to:

$$c_t = E_t\{c_{t+1}\} - \frac{1}{\sigma}(i - E_t\{\pi_{t+1}\} - \rho)$$

where log consumption is equal to the expectation of log consumption minus some proportion of the real interest rate and ρ. Also, assume the ad-hoc money demand is $\frac{M_t}{P_t} = C_t \frac{1}{\eta_{it}}$, where η is the interest semi-elasticity of money demand (i.e. if the interest rate increased by 1% or 100 basis points, then the money demanded will go up by β_1%

1.2.2 The model: Firms

Suppose we have many firms, differentiated goods and the production function is labor intensive (i.e. no capital). Calvo (1983) proposed each firm may reset its price with probability $1 - \theta$. Firms have a static problem where they maximize profits with respect to price. The price is found to be a constant markup dependent on the degree of competition (i.e. ε) times marginal cost. The constant markup is greater than 1. Finding the optimal price is just a discount term times profit. Taking FOC with respect to price finds a new expression (chain rule required). FIRST, multiplying by $\frac{Y_{t+k|t}}{Y_{t+k|t}}$ to the discount term, then distributing the denominator helps us simplify some terms into elasticities. We will eventually arrive to the famous price a la calvo. We also find the Philips Curve

$$\pi_t = \beta E_t\{\pi_{t+1}\} + \kappa \tilde{y}_t$$

$\tilde{y}_t$ is called the output gap. It is considered instead of what we initially had (real marginal cost) because it is much easier to measure. It compares the output from staggered prices and the natural output under full price flexibility (potential output). κ just denotes the sensitivity or slope of the Phillips curve. Essentially, the output gap has a good indicator of the demand of an economy and inflation (since demand can drive prices).

1.3 Lecture 3: Solving Linear Rational Expectations Models

The model consists of 4 equations:

1. New Keynesian Phillips Curve
$$\pi_t = \beta E_t\{\pi_{t+1}\} + \kappa \tilde{y}_t$$

2. Dynamic IS equation
$$y_t = E_t\{y_{t+1}\} - \frac{1}{\sigma}(i_t - E_t\{\pi_{t+1}\})$$

3. Monetary Policy Rule or Taylor Rule
$$i_t = \phi \pi_t$$
4. The natural rate of output
$$y_t^n = \rho y_t^n + u_t$$

There are 3 ways to solve the Linear Rational Expectations model:

1. Method of undetermined coefficients: we guess the form of the solution and solve for the coefficients which express inflation and the dynamic IS equation as proportions of natural output. Thus, the guess takes some proportion of our *exogenous state*
2. Stable/unstable Decoupling: This is the eigenvalue approach, where we express the equations in matrix form. The exogenous state equation, the dynamic IS equation and the Phillips curve is expressed in matrix form in a backward looking fashion.
3. Sims Toolbox

1.3.1 Method of undetermined coefficients

We first guess the form of the solution and solve for the coefficients which express inflation and the dynamic IS equation as proportions of natural output. We elect to express it in this form since y_t^N is the only state variable. Recall, that we can simply remove the third equation, i_t, by substituting it in to the second equation. The remaining equations are expressed in these guesses:

- $y_n^t = \rho y_t^n + u_t$
- $\pi_t = a y_t^n$
- $y_{t+1} = b y_{t+1}^n$

Substituting these into the equations above arrives at the following:

$$a y_t^n = \beta \rho a_t^n + \kappa (b y_t^n - y_t^n)$$
$$b y_t^n = \rho b y^n - \frac{1}{\sigma}(\phi a y_t^n - \rho a y_t^n)$$

Solving for a and b finds the coefficients. Notice that the y_t^n's cancel.

1.3.2 Method of Stable/unstable decoupling

Recall that the model is now 3 equations, structured so that it is backward looking. In matrix notation, we have:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \beta & 0 \\ 0 & \frac{1}{\sigma} & 1 \end{bmatrix} \begin{bmatrix} y_{t+1}^n \\ E_t \pi_{t+1} \\ E_t y_{t+1} \end{bmatrix} = \begin{bmatrix} \rho & 0 & 0 \\ -\kappa & 1 & \kappa \\ 0 & \phi\sigma & 1 \end{bmatrix} \begin{bmatrix} y_t^n \\ \pi_t \\ y_t \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u_t$$

- Reduce the notation into two vectors: one of predetermined variables and one of the forward looking variables.
$$\begin{bmatrix} x_{t+1}^1 \\ E_t x_{t+1}^2 \end{bmatrix}$$
- Pre-multiply the equation by the A_0^{-1}
$$\begin{bmatrix} x_{t+1}^1 \\ E_t x_{t+1}^2 \end{bmatrix} = A \begin{bmatrix} x_t^1 \\ x_t^2 \end{bmatrix} + C u_{t+1}$$
- Find the eigenvalues of the new $A = A_0^{-1} A_1$

- Find the schur decomposition of this matrix as well, which finds the unitary matrix (defined as a matrix who when multiplied by its inverse, returns the identity matrix) and the T matrix. $A = ZTZ^{-1}$
- We then substitute the expression above for A and pre-multiply everything by Z^{-1}. This cancels some matrices on the RHS and we allow:

$$Z^{-1}\begin{bmatrix} x^1_{t+1} \\ E_t x^2_{t+1} \end{bmatrix} = \begin{bmatrix} E_t\theta_{t+1} \\ \delta_{t+1} \end{bmatrix}$$

- The new formulation is: $\begin{bmatrix} \theta_{t+1} \\ E_t\delta_{t+1} \end{bmatrix} = T\begin{bmatrix} \theta_t \\ \delta_t \end{bmatrix} + Z^{-1}Cu_{t+1}$
- We then solve for δ_t using this equation: $E_t\delta_{t+1} = T_{22}\delta_t$. It is only T_{22} since T is a diagonal matrix.
- $\delta_t = T_{22}^{-1}E_t\delta_{t+1}$ where in the limit, T_{22}^{-T} converges to zero ($T_{22} > 1$).
- We use this and recall that $\begin{bmatrix} \theta_t \\ \delta_t \end{bmatrix} = Z^{-1}\begin{bmatrix} x^1_t \\ x^2_t \end{bmatrix} \longrightarrow \begin{bmatrix} \theta_t \\ 0 \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}\begin{bmatrix} x^1_t \\ x^2_t \end{bmatrix}$

  We have that $0 = Q_{21}x^1_t + Q_{22}x^2_t$, and solving for the forward looking, we have $x^2_t = -Q_{22}^{-1}Q_{21}x^1_t$
- This helps us solve for the predetermined variable. We solve from the original matrix formulation.

$$x^1_{t+1} = A_{11}x^1_t + A_{12}x^2_t + C_1u_{t+1} \longrightarrow x^1_{t+1} = (A_{11} - A_{12}Q_{22}^{-1}Q_{21})x^1_t + C_1t_{t+1}$$

1.3.3 Sim's Toolbox: using Gensys

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \beta & 0 \\ 0 & \frac{1}{\sigma} & 1 \end{bmatrix}\begin{bmatrix} y^n_t \\ \pi_t \\ y_t \end{bmatrix} = \begin{bmatrix} \rho & 0 & 0 \\ -\kappa & 1 & \kappa \\ 0 & \phi\sigma & 1 \end{bmatrix}\begin{bmatrix} y^n_{t-1} \\ \pi_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}u_t + \begin{bmatrix} 0 & 0 \\ \beta & 0 \\ \sigma & 1 \end{bmatrix}\begin{bmatrix} \eta^1_t \\ \eta^2_t \end{bmatrix}$$

In this variation, we keep it backward looking, but reflect the expectation through expectational errors. The equation in simplified, matrix form is: $\Gamma_0 Y_t = \Gamma_1 Y_{t-1} + C + \Psi u_t + \Pi\eta_t$. Matlab will do the rest if you define these matrices.

1.3.4 Condition for Existence and Uniqueness

Lets take a model as such: $Ey_{t+1} = \alpha y_t + \theta_t$. We're looking at the parameter α and just simply by observing, make the decision of whether it explodes or converges given the cases.

1. **Case 1: y_t is predetermined** This is just like k_{t+1} in a neo-classical growth model.

$$y_{t+1} = \alpha y_t + \theta_{t+1}$$

   Through progressive repetition into y, you get

$$y_t = \alpha^{t+1}y_{-1} + \sum_{j=0}^{t}\alpha^j\theta_{t-j}$$

   In this case, when α is less than 1, the equation converges. Since the equation was predetermined, the solution already existed.

2. **Case 2: y_t is forward looking** The difference between a forward looking variable and a predetermined variable is the expectations operator.

$$E_t y_{t+1} = \alpha y_t + \theta_t$$

   In this case, we can solve forward or backward.

- **Case I: solving forward** In this scenario, we solve for y_t, and if $|\alpha| > 1$ then we have a solution (since α is in the denominator) that is **unique**.
- **Case II: Solving backward** This is similar to Sims model in that it represents the expectation through an expectations error.

$$y_{t+1} = \alpha y_t + \theta_t + \xi_{t+1}$$

  In this scenario, $|\alpha| < 1$ converges, however this holds for any expectation error that satisfies its expectation being 0. Any of them could be a solution. This is called a **sun spot solution**. We have existence, but **not uniqueness**

For forward looking variables, the solution always exist.

3. **Case 3: Multivariate Case**

$$E_t Y_{t+1} = AY_t + Cu_t$$

According to Blanchard and Kahn, existence and uniqueness in this scenario depend on the eigenvalues. Think about Macro I and how we always dealt with 3 by 3 matrices with 1 unique eigenvalue and had 2 forward looking variables and 1 predetermined. The main equation to consider is:

$$n_s + n_u = n_b + n_f$$

The number of stable and unstable eigenvalues must equal the number of backward and forward variables. **They key is: the number of backward variables must correspond to the number of stable eigenvalues**.

- When the number of forward looking variables equals the number of eigenvalues outside the unit circle, $n_f = n_u$, then we have a **unique solution**
- When the number of forward looking variables is less than the number of eigenvalues outside the unit circle, $n_f < n_u$, then $n_b > n_s$ then no solution exists
- when $n_f > n_u$, then there exists infinitely many solutions since $n_b < n_s$.

1.4 Lecture 4: Dynare

Dynare is a package in matlab used to do what we just did above. What we should take away is how we structure our *model.mod* file, which we then execute as *dynaremodel.mod.*

1. We define the endogenous, exogenous and parameter variables
2. The next block assigns values to these parameters
3. We then define our equations (for the NK, there are the 4 equations) in the next block, indicating (+1) for expectations and (-1) for predetermined.
4. We initiate guesses for each dependent variable (all 4 equal 0)
5. Then we initiate what dynare should do with these equations (i.e. resid;, steady;, check;,shocks; var u=1)
6. Then we run our simulation using the function $stoch\_simul(order = 1, irf = 30)\ \pi\ y\ i\ y_n$, which says to run a first order approximation simulation for 30 periods given these variables. In other words, it linearizes our model around the steady state and solves for the laws of motion, while plotting the impulse responses.

1.5 Lecture 5: SVAR - Empirics

This lecture compares the RBC model with the NK model. This is evidence from Structural autovector regressive DSGE models. The Structural vector autoregressive model (SVAR) takes the form:

$$B_0 y_t = B_1 y_{t-1} + ... + B_p y_{t-p} + \xi_t$$

where ξ_t is the structural shock. Pre-multiplying by B_0^{-1} leads to the reduced form VAR:

$$y_t = A_1 y_{t-1} + ... + A_p y_{t-p} + B_0^{-1}\xi_t$$

$$Y_t = AY_{t-1} + u_t$$

where $A_1 = B_0^{-1}B_1$. Notice now that the error term changed. It is some matrix multiplied by the structural shock. Recall that we've just been analyzing impulse response functions of DSGE models, which are shocks to productivity (from IS curve) and monetary shocks (from Phillips curve equation). We want to do the same with the DSGE model generated by data. To be able to analyze the structural shock, we must solve for the Variance -covariance matrix;

$$\hat{\Omega} = B_0^{-1}B_0^{-1}$$

This is done for two reasons:

- We can observe u_t (the new error term). We cannot observe the structural shock or B_0^{-1}. We need two of these three, so we **exploit the VCV matrix form** to solve for B_0^{-1}.
- The VCV matrix is symmetric (recall that the variance is the expectation of the error term squared), which saves brain power in trying to solve and rationalize the entries of the matrix

$U_t = B_0^{-1}\xi_t \longrightarrow \begin{bmatrix} u_t^y \\ u_t^\pi \\ u_t^m \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_t^y \\ \varepsilon_t^\pi \\ \varepsilon_t^m \end{bmatrix} \longrightarrow \hat{\Omega} = E(U_t U_t') \longrightarrow B_0^{-1}B_0^{-1}$ So B_0^{-1} is a 3x3 matrix.

We can get 6 values from the VCV matrix $\hat{\Omega}$, but there are 9 unknown in B_0^{-1}. This is the **identification issue**. We must set **restrictions** on 3 values of B_0^{-1}. This would allow for the **Cholesky Decomposition**, which splits a matrix into the product of two of the same matrices (think of square rooting). Taking the Cholesky of the VCV matrix, gives us B_0^{-1}.

The following are different examples or different models, where they've empirically tested restrictions in the short run and the long run.

1.5.1 Monetary Shocks

Lets begin with the DSGE model, where we look at the data generating process for y_t, π_t, i_t. Thus, the shocks are the following: $U_t = B_0^{-1}\xi_t \longrightarrow \begin{bmatrix} u_t^y \\ u_t^\pi \\ u_t^m \end{bmatrix} = \begin{bmatrix} b_{11} & 0 & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_t^y \\ \varepsilon_t^\pi \\ \varepsilon_t^m \end{bmatrix}$ To note are the zeros in the upper right triangle. These are short-run restrictions. The structural shock needs to have a contemporaneous relationship to have an entry with respect to the weighted, mutually correlated error term U_t.
Macroeconomic intuition: A shock to inflation today or any monetary shock **does not** affect GDP today. Hence, b_{12} and b_{13} are zero. Likewise, a monetary shock like a change in the money supply or a change in interest rates **do not** affect inflation today.
This section was on monetary shocks and as we can see, monetary shocks only impact the monetary component, which is represented by the interest rate i_t, contemporaneously. Essentially, the corresponding structural shock affects its corresponding variable today, except for the interest rate, which reacts to output, inflation and monetary shocks. $\begin{bmatrix} u_t^y \\ u_t^\pi \\ u_t^m \end{bmatrix} = \begin{bmatrix} b_{11} & 0 & 0 \\ 0 & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_t^y \\ \varepsilon_t^\pi \\ \varepsilon_t^m \end{bmatrix}$
The goal was to find B_0^{-1} which is achieved by taking the cholasky of the the VCV matrix after the restrictions are made.

1.5.2 Fiscal Shocks

$\begin{bmatrix} u_t^y \\ u_t^g \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{bmatrix} \begin{bmatrix} \varepsilon_t^y \\ \varepsilon_t^g \end{bmatrix}$
Fiscal shocks are changes in taxes and government expenditure. The above matrix representation captures government expenditure only. Of course, shocks to output affect output today of cource, and shocks to government expenditure affects output today too (i.e. if the government just today opened a facility that hired 2000 people). But, output does not have affect government expenditure.The government decides when to spend or how to spend before the initiation of the plan. So at time t, we initiate the plan, where we have time t output, but our decision was based on output before time t; rather time $t-2$ or whatever.

1.5.3 Oil Shocks - short run analysis

One of the more conflicting resources is oil; in that, every country depends on it, while very few control most of the power. This lends to fragility of this market with respect to those few countries. Any shock to oil can have global economic effects, which may be strong and persistent. Lets take a simplified model, where we look at the price of oil, the real global economic activity, and the production of oil. We want to see if shocks to the oil supply, or demand shocks affect these three variables.
$\begin{bmatrix} u_t^{prod} \\ u_t^{rea} \\ u_t^{rpoil} \end{bmatrix} = \begin{bmatrix} b_{11} & 0 & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_t^{oilsupply} \\ \varepsilon_t^{aggregatedemand} \\ \varepsilon_t^{oil-specificdemand} \end{bmatrix}$ Demand shocks do not affect production in the short run.
This is obvious. Aggregate demand or specific demand for oil can go down in time t, but oil production is planned in advance, so it also costs money to adjust oil production. It is in their best interest to adjust if the economy continues down the same path. Thus, in the short run, demand does not affect production (supply).
Global real economic activity today is affected of course by the supply of oil and of course by aggregate demand. Yet, the global economy will not react if demand for oil decreases or increases today. The global economy will adjust after some time if the demand continues in whatever path it was on. These changes in demand of course happen because of real price changes.

1.5.4 Productivity Shocks - Long-Run

In the long-run, only shocks to productivity have an effect. So any shock to the factors of production; productivity, labor and capital. Capital is relatively constant, so we will consider a model with labor. The reduced form VAR is:

$$\begin{bmatrix} \Delta a_t \\ \Delta n_t \end{bmatrix} = C_1 \begin{bmatrix} \Delta a_{t-1} \\ \Delta n_{t-1} \end{bmatrix} + A^{-1} \begin{bmatrix} \varepsilon_t^a \\ \varepsilon_t^n \end{bmatrix}$$

We are looking at the first difference of the model, where a denotes productivity and n denotes labor. Remember, we are considering impulse responses, thus let $U_t = \begin{bmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \varepsilon_t^a \\ \varepsilon_t^n \end{bmatrix}$ be the focus. Only a shock to productivity affects technology in the long run. Labor shocks have no effect on technology in the long run. Labor is affected by both technological and labor shocks in the long run. Specifically, a structural shock will change Y_t in the long run by:

$$\frac{\partial Y_{t+\infty}}{\partial \xi_t} \longrightarrow \sum_{t=0}^{\infty} \frac{\partial \Delta Y_{t+\infty}}{\partial \xi_t} \longrightarrow = A^{-1} + +C_1 A^{-1} + C_1^2 A^{-1} ...$$

$$L = (I - C_1)^{-1} A^{-1}$$

provided Y_t is stationary, so $|C_1| < I$. This is simply taking the partial derivative as we continuously input for $\begin{bmatrix} \Delta a_t \\ \Delta n_t \end{bmatrix}$. Essentially then, the long-run effect can be represented as the coefficient Matrix: $\begin{bmatrix} a_{t+\infty} \\ n_{t+\infty} \end{bmatrix} = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \varepsilon_t^a \\ \varepsilon_t^n \end{bmatrix}$ In this model, L_{12} is zero. L$_{21}$ is negative. **This means a positive shock to productivity has a negative effect on labor, which the RBC model does not show. This means the**

RBC must show another driver for productivity to eliminate this possible omitted variable bias. The VCV matrix of $\begin{bmatrix} a_{t+\infty} \\ n_{t+\infty} \end{bmatrix}$ is:

$$LL = (I - C_1)^{-1} A_1^{-1} A_1^{-1'} (I - C_1)^{-1'}$$

$$(I - C_1)^{-1} \Omega (I - C_1)^{-1'}$$

Taking the cholasky of $\Omega = A_1^{-1} A_1^{-1'}$ will find $A_1^{-1} = [(I - C_1)^{-1}]\hat{L}$

1.6 Solving Non-Linear System of Equations

When solving for steady states, we are first taught the analytic solution where we take the lagrange over our control variables, make the variables time-invariant and solve for the variables respectively. Here, we will solve them numerically. For whatever expression, given some parameters and their values, we can find the values of X that satisfy the equation:

$$f(X) = 0$$

This is called **Rootfinding**. This is similar to what you probably did in school. In this section, let X =

$$\begin{bmatrix} C \\ K \\ L \end{bmatrix}$$

where we want to find the values for c, k, l that satisfy the above equation. First and foremost, to understand the method, we have a simple example. Suppose we have the simple space in R^2 (i.e. the Cartesian plane), where we have some polynomial. Lat a and b be the end points for this polynomial and their values -2 and 2 respectively. By the intermediate value theorem, and continuity of the function, there must exist a point in between such that $f(x) = 0$. To find this point, we follow a simple process:

- Your first point to evaluate is decided by: $\frac{(a+b)}{2}$
- The first distance defined is $d^0 = \frac{b-a}{2}$
- After evaluating the point, there are two possibilities:
  - if the point is negative, we do $x_0 + d^1$ where $d^1 = \frac{d^0}{2}$
  - if the point is positive, we subtract to get to zero, so $x_0 - d^1$
- after we move whatever distance, we reevaluate the point and re-do the above steps until we approach $f(x) = 0$

1.6.1 Newton's Method - Multivariate Case

Suppose again we have some polynomial (think of an exponential function). In the multivariate case, we have a function of several variables. Here, we also begin with a guess x_0, except we take the local linear approximation or the 1st order Taylor expansion of the function to approximate the value. For example

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

where $f'(x)$ is the Jacobian matrix of first derivatives. Since we are looking for the point where $f(x) = 0$, we set the LHS to zero, which reveals something neat. Since the approximation is linear, the line must cross zero at some point (assuming the function is not a constant). We then solve for x_1 which is just x in our above approximation, but esoterically, the point where $f(x) = 0$ for the approximation. We now use this point to evaluate another approximation of the function until we found a point such that the approximation is near zero.

There are some drawbacks to this process. Some functions exist where we take the 1st order taylor expansion, the approximation line at this point strikes $f(x) = 0$ at such a point that if we take the approximation again using this new point, it is like we are going in circles. In this case, we must do a backstep where we deviate a bit from the new x we generated, that way it takes us eventually to the point where $f(x) = 0$.

The other issue arises from the Jacobian matrix. This can be very costly to the computer, since the Jacobian is the matrix of first derivatives and there could be several variables. To be efficient, we approximate the Jacobian as some matrix A_0 and we raise it to the k power for the kth iteration.

1.7 Solving models with the ZLB constraint -Deterministic NK Model

There are models with occasionally binding constraints. In this section, we look at the nominal rate hitting the zero-lower bound. For example, if we have a negative shock to inflation or the natural interest rate, then the nominal rate can fall under zero. In this approach, we forbid that from happening and see how this affects the model. This is also similar to when we do interpolation and the next state has less capital than before. The interpolation will sometimes drive that below the budget constraint, but we have to account for that in our code. The most recent example has been in the US, where they were practically operating the economy at the zero lower bound. Interest rates had been so low, and the economy was not exactly in a boom nor a bust, but in ambiguous waters; thus, reducing rates was not possible.

For right now, we focus on a deterministic NK model with the ZLB constraint on the interest rate using two methods:

- OccBin Method using matlab
- Complementarity Problem (noth by hand and matlab)

The deterministic model assumes no error (hence deterministic) and is fully explained by the following:

- $y_t = y_{t+1} - \frac{1}{\sigma}(i_t - \pi_{t+1} - \rho + u_t)$
- $\pi_t = \pi_{t+1} + \kappa y_t$
- $i_t = \max(\rho + \phi_\pi \pi_t, 0)$ where $\phi_\pi > 1$

Let u_t be the vector of shocks and is known. The interest rate is bounded below by zero of course and we make sure the parameter is greater than one for a solution to exist.

1.7.1 OccBin

For the first method, OccBin is used (easily coded in matlab) which is a piecewise interpolation method. For intuition, this is the process:

- We have an input u_t. This is found in the dynamic IS equation. This is a shock variable. We select what period the shock takes place.
- we must guess in which period the interest rate is binding (i.e. we check the first 5 periods for example). In matlab, we check the first 19, then 59.
- Given the guess, find the path of x that solves the linear system $f(x) = 0$. Double check that the path of x is not negative. If it is, we have to make another guess. The solution are just three stochastic processes $\{y_t, \pi_t, i_t\}_{t=0}^{T}$. You'll notice that the interest rate converges to ρ since in the long run, π_t is zero. With the zero lower bound constraint, the interest does not fall under zero.

1.7.2 Complementarity Problem

This is a better method and definitely more intuitive. We construct the NK model into a root-finding problem like before, but using a different method to generate the optimal policy per period. Our model:

$$y_{t+1} - \frac{1}{\sigma}(i_t - \pi_{t+1} - \rho) - y_t = 0$$

$$\beta\pi_{t+1} + \kappa y + u_t - \pi_t = 0$$

$$-\pi - \frac{w}{\kappa}(y_t - y_{t-1}) = 0$$

Now we formulate a few things. We have a lower bound a and an upper bound b. We have an optimal policy $f(x) = -\pi - \frac{w}{\kappa}(y_t - y_{t-1}) = 0$. We construct an $\tilde{f}(x)_i = \min\{\max[f(x), a - x], b - x\}$. For the zero lower bound case, we do not have b. This eliminates the min component. What remains is:

$$\max[f(x), a - x]$$

$$\max[-\pi - \frac{w}{\kappa}(y_t - y_{t-1}), 0 - i_t]$$

This tells the function to pick the optimal policy, but zero if the interest rate is negative. This function maybe non-differentiable however so we reconstruct it in the following way:

$$\tilde{f}(x)_i = \phi^-[\phi^+(f(x), a - x), b - x]$$

Again, we do not have the ϕ^- part since we do not have an upper bound. So:

$$\tilde{f}(x)_i = \phi^+(f(x), a - x)$$

$$\tilde{f}(x)_i = \phi^+(-\pi - \frac{w}{\kappa}(y_t - y_{t-1}), -i_t) = \phi^+(u, v)$$

$$\phi^+(u, v) = u + v + \sqrt{u^2 + v^2} = -\pi - \frac{w}{\kappa}(y_t - y_{t-1}) + (-i_t) + \sqrt{(-\pi - \frac{w}{\kappa}(y_t - y_{t-1}))^2 + i_t^2}$$

The last expression eliminates the kinks and allows to keep interest rates above zero.

1.7.3 Solving the stochastic NK model with the ZLB constraint: Analytic Solution

The stochastic model takes expectations over the next period. The model is the same setup:

$$\pi_t = \beta E_t\{\pi_{t+1}\} + \kappa\tilde{y}_t$$

$$\tilde{y}_t = E_t\{\tilde{y}_{t+1}\} - \frac{1}{\sigma}(i_t - E_t\{\pi_{t+1}\} - r_t^n)$$

$$i_t = max(\rho + \phi_\pi\pi, 0)$$

Take a look at the dynamic IS equation. Let r_n^t be the natural interest rate, which is the rate under full employment.

When the ZLB is not binding, the expressions hold as they are and the solution is:

$$\pi_t = 0$$

$$\tilde{y}_t = 0$$

$$i_t = r_t^n$$

Yet, if r_t^n is negative by some shock, this would imply that the solution i_t is also negative. To avoid this negative interest rate, we have to formulate a short run solution and a long run solution.

In the short run, when the ZLB binds (when the interest rate attempts to go under), we have to consider a different method:

$$\pi_t^s = \beta E_t\{\pi_{t+1}\} + \kappa y^s$$

In the short run, how do we measure these expectations? Simply, we can expect things to not change to some degree and to another degree otherwise. This is reflected here by γ.

$$E_t\{\pi_{t+1}\} = \gamma\pi_{t+1}^s + (1-\gamma)\pi_{t+1}^*$$

Let $\pi*$ indicate reverting to the natural interest rate. Thus, with probability $(1-\gamma)$ we revert to the solution when the ZLB is not binding. The solution is zero, thus:

$$\pi_t^s = \beta\gamma\pi^s + \kappa y^s$$

The same process is done for the dynamic IS equation:

$$y^s = \gamma y^s + (1-\gamma) - [(\gamma(0 - \pi^s + \varepsilon)) + (1-\gamma)(\phi\pi^* - \pi^*)]$$

where $i_t = 0$, the expectation of inflation tomorrow is the same as inflation today π_s and it is a minus r_t^n and $r_t^n = -\varepsilon$ so we get a positive ε. This reduces to:

$$y^s = \gamma y^s - \gamma(\varepsilon - \pi^s)$$

1.8 Kalman Filter

My bestfriend. The kalman filter is a great tool to estimate models, whom otherwise cannot be measured. Specifically, the kalman filter estimates the unobserved using an observable; in some disciplines, it is known as signals. The state equation is the equation containing the unobserved and the measurement equation is the equation containing the signal, which attempts to estimate the unobserved.

$$X_t = AX_{t-1} + Cu_t$$

$$Z_t = DX_t + v_t$$

Here, Z_t is our signal, which is attempting to estimate the unobserved X_t. The errors are normally distributed. **Keep in mind that for the time being, I will make sure to respect notation; yet, since the estimates converge quite quickly to the true value, the time indices do not hold weight**. Lets begin by defining the equations necessary to estimate (the irony):

- $$X_{t|t} = AX_{t-1|t-1} + K(Z_t - DAX_{t-1|t-1})$$

  where K denotes the kalman gain and the quantity in the parenthesis is called the "surprise". This quantity attempts to represent the new difference between prior belief and today's signal. Thus, $X_{t|t}$ is the prior belief plus the weight we place on this new difference or surprise. $DAX_{t-1|t-1} = X_{t|t-1} =$ The expectation of X_t today conditional on beliefs at time t-1

- $$K_t = P_{t|t-1}D'(DP_{t|t-1}D' + \Sigma_{vv})^{-1}$$

  The Kalman gain can be understood as the variance of X_t divided by the summation of the variances (think about the optimal weight to attach to a signal). In this matrix formulation, more precisely, we have the variance of X_t divided by the variance of X_t and the variance of Z_t. The variance of X_t is known as the posterior uncertainty.

$$P_{t+1|t} = A[P_{t|t-1} - P_{t|t-1}D'(DP_{t|t-1}D' + \Sigma_{vv})^{-1}DP_{t|t-1}]A' + CC'$$

$$P_{t|t-1} = E(X_t - X_{t-1})E(X_t - X_{t-1})'$$

With these tools, we have the equations necessary to perform the Kalman filter. The Kalman filter actually solves for X_t recursively using the signal equations and initial conditions $X_{0|0}$ and $P_{0|0}$

1.8.1 Intuition

In my notes, there is a signal extraction problem with 2 signals. We attempt to estimate the difficulty of an exam given the lectures as the signals. Note that the optimal weight, K can be expressed in two ways. Given that the inverse of the variance is the precision, K can be expressed as the precision of component 1 divided by total precision or the variance of component 1 divided by the total variance.

1.8.2 The Scalar Kalman Filter

The scaler Kalman filter is basically a dynamic and recursive version of the previous filter:

$$x_t = \rho x_{t-1} + u_t$$

$$z_t = x_t + v_t$$

$x_{0|0}$ and $p_{0|0}$ is required. To begin the process, notice that:

$$x_{1|0} = \rho x_{0|0}$$

and

$$\begin{aligned} p_{1|0} &= \text{posterior uncertainty or uncertainty about what is going to happen tomorrow} \\ &= E(x_{1|0} - x_1)^2 \\ &= E(\rho x_{0|0} - \rho x_0 - u_1)^2 \\ &= \rho^2 E(x_{0|0} - x_0)^2 + \sigma_u^2 \\ &= \rho^2 p_{0|0} + \sigma_u^2 \\ p_{t|t-1} &= \rho^2 p_{t-1|t-1} + \sigma_u^2 \\ \text{Recall:} \longrightarrow p_{t|t} &= p_{t|t-1} - p_{t|t-1}^2 (p_{t|t-1} + \sigma_v^2)^{-1}) + \sigma_u^2 \end{aligned}$$

In the case where the posterior uncertainty equals today's uncertainty we have (which happens when the signals we get are useless or have a high variance):

$$p_{t|t-1} - \rho^2 p_{t-1|t-1} = \sigma_u^2 \longrightarrow p_{t|t-1} = (1-\rho^2)^{-1}\sigma_u^2$$

We then have to update. We know $x_{1|0}, z_1$ and $p_{1|0}$.

$$x_{1|1} = (1-\kappa)x_{1|0} + \kappa z_1$$

As we update, we get that:

$$\begin{aligned} x_{t|t} &= (1-\kappa_t)x_{t|t-1} + \kappa_t z_t \\ &= \rho x_{t-1|t-1} + k_t(z_t - \rho x_{t-1|t-1}) \end{aligned}$$

Likewise as we update the kalman gain, we get:

$$k_t = P_{t|t-1}(P_{t|t-1} + \sigma_v^2)^{-1}$$

which comes from the original big fraction we made earlier. The updated variance is

$$p_{1|1} = (1-\kappa)p_{1|0}$$

1.8.3 The Kalman Smoother

From the Kalman filter we get a real time estimate. So, what is the estimate today or tomorrow etc. But, sometimes we want to know the best estimate given the full sample, and not the estimate over time. Thus:

$$X_{t|T} = X_{t|t} + J_t(X_{t+1|T} - X_{t+1|t})$$

Then taking the variance of this (first subtracting terms to the LHS)

$$J_t = P_{t|t}A'P_{t|T}^{-1}$$

$$P_{t|T} = P_{t|t} + J_t(P_{t+1|T} - P_{t+1|t})J_t'$$

1.8.4 Gram-Schmidt Orthogonalization

This is projection theory. Basically, we derive the formulas in matrix notation.

Deriving K_1

$$Z_{1|0} = DAX_{0|0} \longrightarrow \tilde{Z}_1 = Z_1 - Z_{1|0} = Z_1 - DAX_{0|0}$$

Step 2

$$E(X_1|Z_1, X_{1|0})$$

$$E(X_1|\tilde{Z}_1, X_{0|0}) = E(X_1|\tilde{Z}_1) + E(X_1|X_{0|0})$$

Step 3

$$< X_1 - K\tilde{Z}_1, \tilde{Z}_1 >= 0$$

$$E(X_1 - K\tilde{Z}_1)\tilde{Z}_1 = 0$$

$$E(X_1\tilde{Z}_1') = K_1E(\tilde{Z}_1\tilde{Z}_1') \longrightarrow K_1 = E(X_1\tilde{Z}_1')[E(\tilde{Z}_1\tilde{Z}_1')]^{-1}$$

Step 4

$$E(X_1\tilde{Z}_1') = E[(\tilde{X}_1 + X_{1|0})\tilde{Z}_1] = E(\tilde{X}_1\tilde{Z}_1) = E(\tilde{X}_1(D\tilde{X}_1 + v_1))$$

$$= P_{1|0}D'$$

Step 5

$$E(\tilde{Z}_1\tilde{Z}_1) = E(D\tilde{X}_1 + v_1)(D\tilde{X}_1 + v_1) \longrightarrow DP_{1|0}D' + \Sigma_{vv}$$

Step 6: Putting it all together

$$K_1 = E(X_1\tilde{Z}_1')[E(\tilde{Z}_1\tilde{Z}_1')] \longrightarrow P_{1|0}D'[DP_{1|0}D' + \Sigma_{vv}]^{-1}$$

Deriving $X_{1|1}$

$$X_{1|1} = AX_{0|0} + K_1\tilde{Z}_1 \longrightarrow AX_{0|0} + K_1(Z_1 - DAX_{0|0})$$

Deriving $P_{1|1}$

$$X_{t|t} = X_{t|t-1} + K_t\tilde{Z}_t$$

Switching sides and adding X_t to both sides:

$$X_t - X_{t|t} = X_t - X_{t|t-1} + K_t\tilde{Z}_t$$

Taking the variance:

$$P_{t|t} + K_tE(\tilde{Z}_t\tilde{Z}_t')K_t' = P_{t|t-1}$$

$$P_{t|t} = P_{t|t-1} - P_{t|t-1}D'(DP_{t|t-1}D' + \Sigma_{vv})^{-1}$$

1.8.5 Constructing Likelihood Function under Kalman Filter

We have to find the parameters Θ which maximize the probability of or likelihood of Z_t. The likelihood is the result of the following process:

$$f(Z_i...Z_N|\Theta) = \prod_{i=1}^{N} f(Z_i|\theta_i)$$

$$\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \prod_{i=1}^{N} e^{\frac{-\tilde{Z}'\tilde{Z}}{2\sigma^2}}$$

Then taking the log:

$$-\frac{T}{2}\ln(2\pi) - \frac{T}{2}\ln[det(DP_{t|t-1}D' + \Sigma_{vv})] - \frac{1}{2}\sum_{t=1}^{T} \tilde{Z}'(DP_{t|t-1}D' + \Sigma_{vv})^{-1}\tilde{Z}$$

$$-\frac{T}{2}\ln(2\pi) - \frac{T}{2}\ln[det(\Omega)] - \frac{1}{2}\sum_{t=1}^{T} \tilde{Z}'(\Omega)^{-1}\tilde{Z}$$

In the maximum likelihood process, we typically take FOC with respect to the variance (omega in this case). Thus, we must find the variances which maximize the likelihood of Z. Typically, instead of an analytic solution, we must do a grid search to see which vector of variances (the variances for each error term) maximize the likelihood. Grid search requires setting the bounds to search, so we typically start with zero (since the variance is bounded below by zero) and set an upper bound equal to the sample variance (the variance of the measurement). With few parameters, this is a fairly simple process; but becomes computationally demanding if more are added.

1.9 Simulated Annealing

Simulated Annealing is used to find the series of $\hat{\theta}$ such that maximize our likelihood. Matlab literally searches through each parameter, between the bounds, until of course a vector is produced such that it maximizes the likelihood. The process for simulated annealing is comparable to the hook game. Imagine you have this little toy filled with water and there are little buckets floating. I can press two buttons which shoots these rings with the purpose of making them inside the bucket. At first, you're just pressing the buttons, making a few here and there. As you make them, your behavior changes and you only accept a few moves. Meaning, you wouldn't just press the button. Maybe you tilt the little toy to guide its direction. The point is: your process becomes more precise until all the rings are inside the bucket. Let

$$\theta_0$$

be the configuration now. Let $f(\theta_0)$ be the evaluation of this configuration. Suppose I press a button and then analyze my situation/configuration. If $f(\theta_1) > f(\theta_0)$ we accept the move. If it is less than, then we accepte with a certain probability which is our choice. This acceptance rule is defined below:

$$e^(\frac{f(\theta_1) - f(\theta_0)}{T} = a$$

where the numerator is Δf and the denominator is the Temperature. a is the acceptance probability we set exogenously. As temperature increases, we accept bad moves at a greater probability.

After making 20 or so many moves, we have to adjust our move size. Lets take this into the context of finding the θ that maximize our function. We essentially make smaller moves such that 50% of the moves are accepted. This is done to maximize our moves "late in the game".

After 5 iterations of adjusting the step size, we we reduce the temperature, which means we reduce our acceptance rule by the temperature reduction factor. Running this on matlab is pretty cool, but essentially we find a vector of the parameters that maximize our likelihood.

1.10 DSGE In State Space

This was on the previous exam. Below is our system of equations:

- $$x_t = \rho x_{t-1} + u_t^x$$
- $$y_t = E_t y_{t+1} - \frac{1}{\gamma}(i_t - E_t \pi_{t+1}) + u_t^y$$
- $$\pi_t = E_t \pi_{t+1} + \kappa(y_t - x_t) + u_t^\pi$$
- $$i_t = \phi \pi_t + u_t^m$$

The idea here is we are trying to estimate potential output or natural output, denoted by x_t. To express this in **state space form**, we have to put all terms today, t, on the LHS and then pre-multiply the coefficients matrix by its inverse. The result is below, where the coefficients matrix is A_0 and A_0^{-1} is A_1:

$$X_t = AX_{t-1} + Cu_t \longleftrightarrow x_t = \rho x_{t-1} + u_t^x$$

$$Z_t = DX_t + A_1 v_t \longrightarrow \begin{bmatrix} i_t \\ \pi_t \\ y_t \end{bmatrix} = \begin{bmatrix} \frac{\phi\kappa\gamma(1-\rho)}{-c} \\ \frac{\kappa\gamma(1-\rho)}{-c} \\ \frac{\kappa(\phi-\rho)}{-c} \end{bmatrix} x_t + \begin{bmatrix} 1 & -\phi & 0 \\ 0 & 1 & -\kappa \\ \frac{1}{\gamma} & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u_t^r \\ u_t^\pi \\ u_t^y \end{bmatrix}$$

Step 2: Computing the Likelihood function using KF. This is identical to section 1.8.5. The computation requires FOC of the MLF with respect to the variance (Ω) and the other parameters: $\{\rho, \gamma, \kappa, \phi, \sigma_x, \sigma_y, \sigma_\pi, \sigma_r\}$.

1.11 Bayesian Estimation of DSGE Models

The issue with performing MLE and using the kalman filter to estimate the states is high dimensionality which may lead to ill-behaved models and possibly many local maxima. maximizing a complicated, highly dimensional function like the likelihood of a DSGE model. It is actually much harder than it is to integrate it, which is what we do in a Bayesian exercise. First, the likelihood of DSGE models is, as I have just mentioned, a highly dimensional object, with a dozen or so parameters in the simplest cases to close to a hundred in some of the richest models in the literature. Any search in a high dimensional function is fraught with peril. More pointedly, likelihoods of DSGE models are full of local maxima and minima and of nearly flat surfaces. This is due both to the sparsity of the data (quarterly data do not give us the luxury of many observations that micro panels provide) and to the flexibility of DSGE models in generating similar behavior with relatively different combination of parameter values .... Moreover, the standard errors of the estimates are notoriously difficult to compute and their asymptotic distribution a poor approximation to the small sample one.
The solution is the **Bayesian approach**, which looks at both the data and the parameters as *random variables*. Thus, these parameters, instead of being a fixed value, have a representative distribution. This is different from the **Frequentist approach**, where we view the data as the random variable and the parameters as fixed, but unknown quantities. We typically generate an estimate $\hat{\theta}$, but this changes between the samples. The **sampling distribution** is the result of all these estimates and essentially captures the uncertainty of our estimate. Instead of giving it a fixed value, the bayesian approach assigns the parameter a distribution, which does not require data. Suppose we have a quarter. If we toss the coin, it has to land on heads or tails. Suppose I also do not have any data, but I decide to represent this as a uniform distribution where the probability of getting heads is anywhere between 0 and 1 with equal probability. This is my **prior** - a mathematical representation of my beliefs. A better prior would be a **beta distribution** with $\alpha = 30$ and $\beta = 30$, which gives an expected value of 0.5, as we would expect as the probability of either getting heads or tails. Now, suppose I run an experiment and have data. I get 4 heads and 6 tails out of the 10 tosses. I can now make a **Likelihood** estimation given these results (the likelihood is just the probability

of observing the data given these parameters), which I find to be a **binomial distribution** since we are dealing with a discrete measure. We then update our belief by considering this experiment, which leads to our **posterior distribution**.

Comparative Statics: An informative prior and a small sample size even can get us a reliable posterior distribution. An uninformative prior distribution is evident when I multiply it by the likelihood function, and it generates a posterior nearly identical to the likelihood. Leaving the likelihood constant, if I improve the prior, the posterior also improves. If I increase the sample size of the likelihood function, given some prior, it will also influence the posterior a lot. Thus it is always a balance between an informative prior and sample size. To begin, we define **Bayes Rule**:

$$P(A,B) = P(B|A)P(A) \longrightarrow P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

$$p(\theta|y^T) = \frac{L(y^T|\theta)P(\theta)}{P(y^T)}$$

This is what it means to update your posteriors; the LHS $p(\theta|y^T)$ gives the **posterior distribution** of the now RV theta parameters given the data or *evidence*, y^T. This implies that our prior is $P(\theta)$. The denominator is considered some normalizing constant as it is independent of the parameters. To estimate the posterior distribution, it is complex and not easily tractable. We wish to consider just the **mean and the variance** in characterizing the distribution.
The expectation and variance, as always, is calculated as:

$$E(\theta|y^T) = \int \theta p(\theta|y^T)d\theta \approx \frac{1}{S}\sum_{s=1}^{S} g(\theta_s)$$

$$var(\theta|y^T) = E(\theta^2|y^T) - [E(\theta|y^T)^2]$$

To simulate the posteriors to construct the distribution, we need to generate a time series $\{\theta\}_j^S = 0$. We do this with the following algorithm.

1.11.1 Metropolis Hastings Algorithm

Think of **Monte Carlo Simulation**. Suppose we have

$$\theta_t \sim \mathcal{N}(0.5, \sigma)$$

We make a proposal distribution, which is selected as the normal distribution. After drawing a series of θ_t, we have created a histogram of those values. After drawing $10,000$ times, so 10000 thetas, we have made a nice distribution. centered at .5. Consider now a **Markov Chain** where θ_t depends on its previous value θ_{t-1}.

$$\theta_t \sim \mathcal{N}(\theta_{t-1}, \sigma)$$

This dependency typically causes problems and does not match the proposal density well. In some cases, the random number generation process can take on a random walk, and thus, the resulting density plot does not resemble the proposal distribution. Thus, the birth of the **Metropolis Hastings Algorithm**. It accepts and rejects proposed values of θ. Consider again the markov chain, but with this acceptance rule:

$$\theta_t \sim \mathcal{N}(\theta_{t-1}, \sigma)$$

$$\alpha = \min\left(1, \frac{p(\theta^*|y)}{p(\theta^{s-1}|y)}\frac{q(\theta^{(s-1)}|\theta^*)}{q(\theta^*|\theta^{(s-1)})}\right)$$

The way the rule works: The first fraction is the posterior of the new divided the posterior of the old theta. If this fraction is greater than 1, meaning the new theta represents the distribution better (greater posterior probability), then we always accept. Recall that we do not actually have to know the form of this distribution- we can just calculate it using the Log-likelihood and the prior distribution. The prior distribution is our choice, our belief so we do not need data for it (could be normal, beta, fischer distribution etc). If the fraction is less than 1, then we draw a **randon number** from a uniform distribution $U(0, 1)$ (since the fraction is between 0 and 1) and if the number is less than the fraction, then we accept the new theta θ^*. If the number is greater than the fraction ,we reject and keep $\theta^{(s-1)}$. **Note**: Each θ that we draw is a sample, and has its own distribution. This is the **proposal density** $q(\theta^*|\theta^{(s-1)})$ - typically normally distributed. Eventually, after repeating this process a finite S number of times, we have created a sample of the posterior distribution of the accepted θ's.

This process also has its problems: The sample depends on the starting value. If its a bad guess, then we could generate a poor distribution. So, we consider a **burn in period**, which is the part of the sample we *burn away.* The result is stability and a stabilized/intelligent guess. **STATA**'s default is to burn the first 2500 observations. **The other problem** is auto-correlation between the thetas. We noted this above. Essentially, thinning could be used, where suppose we want to draw $10,000$ thetas/numbers. We can increase the draws, to lets say $30,000$ and just pull out every third number. Thus, **thinning** the sample. This typically used, however, when the model has been correctly specified and auto-correlation strongly persists.
Consider the DSGE model above. Recall that we can use previous data, independent evidence or literature to formulate our priors. As already noted, a good prior makes for reliable results. When choosing a prior, you need **specify the functional form** and **the hyperparameters**.

1.11.2 Types of Priors

:

- *Normal Distribution*: Defined by its first two moments, unbounded
  - **When is it useful?** Feedback parameters, where the sign is unknown
- *Uniform Distribution*: Constant pdf, bounded. All points equally likely. This is the **uninformative prior** (unless the parameters are truly believed to be equally likely in the interval).
- *Beta Distribution*: Bounded on $[0, 1]$.
  - **When is it useful?** This is typically useful with parameters which are also bounded like the discount term, the calvo parameters and autoregressive parameters.
- *Gamma Distribution*: Supports $[0, \infty]$ which is good for variances and Taylor Rule feedback parameters. *The inverse gamma distribution* also exists, also used for variances.
- *Inverse Gamma Prior*: theoretically a better choice for variances if you want to prevent **stochastic singularity**, which in very layman terms is when the number of shocks is less than the number of observables because the shocks of the observables could simply be multiples of each other, which of course would not reflect on the data, where in that case likelihood estimation not possible.

1.11.3 Simulating the Posterior

As we mentioned earlier, we have the **Metropolis-Hastings Algorithm**, which can simulate based on an acceptance rule. We then do some convergence checks, where for each draw, we have a mean and a variance and see whether it converges to some number. Read **Herbst and Schorfheide (2015)**: Bayesian Estimation of DSGE Models.

2 Part 2: Block 1

Before we get into the topic, it is important to note that these are the basics for dynamic problems. We look at value function iteration and exploit the fact that we can setup the problem (under certain conditions) as a contraction mapping, which implies that a fixed point exists. We get into what that means later. In addition, this block will cover basic numerical tools like functional approximation, some useful algebra, rootfinding, numerical integration and doing so efficiently.

2.0.1 Warm-up: Incomplete Markets Model

The markets are incomplete since you cannot buy contingent claims to insure idiosyncratic risk. Households face an exogenous stochastic process for labor supply, hence there is uncertainty about how much labor a household allocates. Since you cannot insure against this through future claims, the markets are incomplete. What households *can* do is decide between investing and consuming today. Such a decision may insure you if you work less the next period (in this case, if you consumed less, then you might have some cushion next period to take on the possibility of allocating less labor). Labor supply $n_t \in [n_{min}, n_{max}]$ is the exogenous process.
Also note, there is a debt limit b, typically set to zero, which means you cannot incur debt at all. This is a lower bound. $a' \geq -b$ means as long as the amount of asset holdings does not go under zero, you are ok.
The household problem is:

$$\max_{a_t} E_0 \sum_{t=0}^{\infty} \beta^t u(w_t n_t + a_t(1+t_t) - a_{t+1})$$

where consumption, $c_t = w_t n_t + a_t(1+t_t) - a_{t+1}$ and a_{t+1} is predetermined.

2.1 Dynamic Programming

2.1.1 Dynamic Estimation Methods: Finite Time Horizon

So we have our typical, dynamic finite horizon problem, where the utility is dependent on a **predetermined** state variable (x_t) and a **control** state variable (x_{t+1}). This is a **recursive problem**; meaning, the current state is the only determinant of today.

$$\max \sum_{t=0}^{T} \beta^t u(x_t, x_{t+1})$$

To work with this, we convert this to a value function. Value functions are the max functions and are represented as indirect utilities. Meaning, if I like a certain bundle, then I must also have a utility for the prices and wealth combinations that allows for me to consume this bundle. As a consumer, I want to maximize my consumption, but indirectly, given certain states of the world, I can maximize this preference. The **Bellman Equation** shows just that:

$$V(x_t) = \max_{x_{t+1}} u(x_t, x_{t+1}) + \beta V(x_{t+1}) \longrightarrow \text{s.t. } x_{t+1} \in \Gamma_t(x_t)$$

The constraint just says that the state in the future, my choice, must be feasible with the existing state today. The max exists (hence the value function exists V) since the "budget", $\Gamma_t(x_t)$, is compact. There are only so many possible realizations from this budget set. This is the **theorem of the maximum**. Specifically, this is needed for the continuity of solutions given a change in the parameters. The maximum theorem says that the set of maximizers and the value of the problem change continuously with the parameters provided the objective function is continuous and the constraint correspondence is compact valued and continuous. Under the finite time horizon, the process is non-stationary, meaning the first and second moments may change depending on the time index. When the problem is infinite, we are considering all time periods always, so there is just a mean and a variance for all time - thus stationary (or we can at least make it stationary through first differencing or whatever).

2.1.2 Stationary Dynamic Programming

$$V(x) = \max_{y} u(x, y) + \beta V(y) \text{ s.t. } y \in \Gamma(x)$$

where $\Gamma(x)$ is convex and compact. To motivate this, consider the **neo-classical growth model** from Macro I; where we have our concave utility function.

$$V(k_0) = \max_{K_t} \sum_{t=0}^{\infty} \beta^t u(C(K_t, K_{t+1}))$$

We convert this intuitively. We have the same problem everyday of maximizing our utility with respect to the state today and the state tomorrow. We seek a time-invariant policy function which takes us from today's state to the next. This is essentially an infinite sequence of controls, that maximizes our value function each period. We also seek the convergence of the value functions, where **the solution is a function** that when mapped into the same space is itself. It is uniquely strictly concave. Using this idea, lets remove today's maximization problem and add the remaining.

$$= \max_{K_1} u(K_0, K_1) + \max_{\{K_t\}_{t=2\ldots\infty}} \beta \sum_{t=0}^{\infty} \beta^t u(K_t, K_{t+1})$$

$$= \max_{K' \in \Gamma_t(K_t)} u(K, K') + \beta V(k')$$

When does a solution exist?

Let the Bellman be a mapping, which means if I apply this operator to the value function, it takes me to another utility function.

$$T(V(x)) = U(x)$$

$$T[V(x)] = \max_y u(x, y) + \beta V(y)$$

If T is a **contraction mapping**, then a solution exists to the above equation. The solution is extracted from the set of **bounded** functions (which is a metric space with the sup-norm as a metric). The **sup-norm**, in simple terms, is the distance between two functions in a bounded space, expressed as $|f(x) - g(x)|$. Mentioning contraction mapping requires some prerequisites:

- Recall that a **cauchy sequence** is a sequence on a metric space such that $\forall \varepsilon > 0$, there exists an $n(\varepsilon)$ such that for all $n, m \geq n(\varepsilon)$, $d(x_n, x_m) < \varepsilon$
- All cauchy sequences converge in a metric space.
- The mapping T is a contraction if any $x, y \in C \longrightarrow T(x), T(y) \in C$ and for some $0 < \rho < 1$, $d(Tx, Ty) \leq \rho d(x, y)$
- If T is a contraction, then $T^n x$ is a Cauchy sequence.

2.1.3 Value Function Iteration - Algorithm to solve the Bellman Equation

We construct an infinite sequence of value functions and associated policy functions. We iterate on the bellman, starting with $V_0 = 0$. To prove the existence of the solution for the Bellman, we saw that a contraction is necessary. Starting with **any** initial guess, any V^0 from the set of continuous and bounded functions, we need to find an operator such that $V_t = T(V_{t-1})$. After a large number of iterations, V_n becomes "close" to the solution V^*. Formally, we take the supremum norm of the difference between the iterations. Lets define this as d_∞. The distance is defined since we are operating on a complete metric space (closed, bounded thus all Cauchy sequences converge). This is **Value Function Iteration**.

Blackwell's Sufficient Conditions: For T to be a contraction mapping on the set of bounded and continuous functions, to must preserve boundedness, continuity, monotonicity, and discounting. This means:

- For some $v(x) \leq w(x)$ we have
$$Tv = \max\{u(k, k') + \beta v(k')\}$$
- Thus, $Tw = \max\{u(k, k') + \beta w(k')$

- which means $Tv \geq Tw$, proving monotonicity
- It also means for: $T(v+c) = \max\{u(k,k') + \beta[V(k') + c]\}$
- $= \max\{u(k,k') + \beta V(k') + \beta c]\}$
- $= Tv + \beta c$, which implies the operator discounts.
- This satisfies both blackwell conditions

Policy Function:

$u(x, x')$		$\Gamma(x)$
Real-valued		Compact Valued
Continuous	The set of states is convex	Continuous
Strictly Concave		Convex
Bounded		Non-Empty

Then, $\longrightarrow V(x)$ is unique, continuous, and strictly concave. The optimal policy is a continuous, single valued function:

$$\phi(x) = argmax\ u(x,y) + \beta V(y)$$

$\phi(x)$ can be represented as $g(k_t) = k_{t+1}$. It is the optimal policy "kprime" which maximizes the value function at each period.

The Numerical Process

First, we force the state vector to live on a finite, discrete grid of points. This is done to dampen the curse of dimensionality, which occurs when the state space is too large. (The second way is to use polynomials to approximate the value function, but we will get to that later). We also define another state variable, which we'll consider income states, decided by some probability matrix. We are given capital and income state at time $t = 0$. We assume $\beta(1 + r) < 1$, which means the time preference rate is greater than the interest rate. THE REASON WHY THIS IS DONE WILL BE EXPLAINED LATER.
Thus, our bellman is: $V(k,z) = \max\{u(k,k',z) + \beta V(k',z)\}$
Or rather, for each state $i \in [1...m]$ and for each grid point (which is in itself a state, multiple starting points) $h \in [1,...n]$

$$V(k_h, z_i) = \max\{u(k_h, k', z_i) + \beta \sum_{j=1}^{m} P_{ij} V(k', z_j)\}$$

where j denotes the next state in the sequence.

We define, thus, 2 nx1's, for our value function iteration, one for each state j. Each row defines a grid point i, which is some state for the asset.
The process itself is performed backward. We start with a guess V^{n-1}. In our matlab code, this is just an Nxz matrix, where N are the number of grid points and z are the number of states. Specifically, it is an Nxz matrix of zeros. Thus we have: $V^n(s) = \max_{k' \in \Gamma(k)} u(f(K) - K') + \beta \cdot 0$. $f(K)$ is resources we have left over, NOT OUTPUT. The general setup is:

$$V^n(s) = \max_{s' \in \Gamma(s)} u(s,s') + \beta V^{n-1}(s')$$

Notice how the value function is one less on the inside, but dependent on the later state. Intuitively, we setup our value function for some period. Then we take FOC with respect to consumption and our capital choice tomorrow with the production constraint. Finding the explicit solutions, we replace it into the original value function and repeat the process. We can then find the policy function given each iteration (for each value function). **The idea mostly is: we make a guess, solve for the policy function, s_{t+1}, and evaluate the value function to compare the previous value function (in our case, zero). We replace our guess with this new value function until the value functions are very close which will occur due to the contraction mapping theorem.**

$$V_i^n = \max\left(U(i) + \beta \mathbf{V}^{n-1}\right) \quad \text{where i is the state of the world}$$

Stacking all the states, we form the matrices and introduce i_N which is a column vector of 1's:

$$\mathbf{V}^n = \max\left(\mathbf{U} + \beta i_N \mathbf{V}^{n-1}\right)$$

In Matlab, the process is the following:

- We define the continuation value: $\beta\mathbf{Vz}$ where $\mathbf{z}$ is our 2x2 matrix of stochastic productivity. $\mathbf{V}$ is a Nx2 in this case, where N is the number of grid points and 2 describes the number of possible productivity states. In our case, the EV is zero.
- We then reshape the matrix of continuation values and make copies of them. So above, we produced a 100x2, so 2 columns of 100 values. We want to reshape this into two 1x100's then make 100 copies of it for each grid point. This is the i_N. We call the new matrix "EVFull"
- Then we define "Vnew" which is the value function ahead and EVFull captures the value function behind. We take the max line by line. $Vnew = max(U + EVFull)$ Vnew is the new guess.
- We then measure the distance between Vnew and V until it reaches the critical point.
- "kprime" is extracted, which is the argmax of the value function iterations.
- The fixed point between x_{t+1} and x_t are points of convergence or rather, the steady states.

Analysis:

- Value function iteration must always work, but could be computationally demanding, hence slow.
- The speed of convergence can be solved for depending on the guess we make and of course knowing already the solution.

  $T(\hat{V}) = \max(u(s,s')+\beta(V^*+c)) \longrightarrow T(\hat{V}) = V^*+\beta c, \ \ where\ (1-\beta)^n c$ is the dist from the solution after n iter.

- **Curse of Dimensionality**: with too many endogenous states, the process can be too computationally demanding.

2.2 Off-grid Search

At the moment, we defined our grid and told Matlab to calculate considering every grid point (i.e. the capital state) and each income state. This is known as discretization. **Off Grid Search** considers the points in between the grid and makes a weighted average to build values for those *points in between.*

2.2.1 Solving for the Value Function Parametrically

Typically we are given a bellman equation and are asked to solve. We can always use value function iteration, but in the literature, efficiency is important. So instead, we estimate the value function's functional form. This is possible since the value function is **continuous**.

- This is our problem.

$$V(s) = \max_{s'\in\Gamma(s)} u(s,s') + \beta V(s')$$

- **Step 1**: We define a new function $\hat{V}(s'|\theta_n)$, which says we construct V using parameters θ_n, where θ is the universal notation for parameters. Using these parameters to estimate the state, we construct our new value function.
- The new function:

$$V_i^{n+1} = \max_{s'\in\Gamma(s_i)} u(s_i,s') + \beta\hat{V}(s'|\theta_n)$$

  The parametric form is a guess and we need to update it to maximize the value function every period.

- **How do we update?**: We first of all need the proper functional forms for V^*. Parametric families to consider (where a 1-to-1 mapping needs to exist between V^n to θ^{n+1}) are:
  - Chebyshev Polynomials of order m
  - Interpolations
- For Chebyshev polynomials, this mapping: V^n to θ^{n+1} is necessary since if I expressed V^* as $y = \theta x + b$, the optimal θ which minimizes the residual sum of squares is the famous OLS estimator:

$$\underline{\theta^{n+1}} = (P'P)^{-1}P'\underline{V^n}$$

- For interpolation, using Matlab, we can construct $\hat{V}^{n+1} = griddedinterpolant(grid.s, V^n)$

2.3 Interpolation - Functional Approximation

Since computers are discrete machines, we have to rewrite this **still** in a limited way, but in the sense that we can only use a finite number of parameters when estimating the functional form of the expected value. A low-dimensional vector of parameters would be ideal.

The idea with **interpolation** is we pick a few points, lets say 10, and evaluate our function at those 10 points, approximating the values in between. Let $x_{i=1..N}$ be the set of points where we evaluate the interpolation. $f_i(x_i)$ is the evaluation. Two problems arise:

- How to choose the functional approximation
- How to choose the grid points and how much of them

2.3.1 How to choose the functional approximation

Naturally, when we think of approximating a function, we have either some linear combination or non-linear so to some m degree. Formally, it can be expressed as:

$$\hat{f}(x) = \sum_{j=1}^{n} \psi_j c_j(x)$$

Of course, the basis function $c_j(x)$ can be x, x^2 etc and ψ_j is the coefficient or weight applied to the **basis function**. The above form is just some polynomial. Of particular interest are **Chebyshev Polynomials**.

$$c_j(x) = \cos(j \ \arccos x)$$

Let $\mathbf{C}$ be the matrix of chebyshev polynomials for all grid points i and degree j.

$$\mathbf{C} = [c_j(x_i)]_{i=1...M, j=1...n}$$

The function itself and the optimal estimator which minimizes the MSE is

$$\hat{\mathbf{f}} = \mathbf{C}\psi \longrightarrow \psi^* = (\mathbf{C}'\mathbf{C})^{-1}\mathbf{C}'\mathbf{f}$$

Problems can immediately occur if

- the functional form is incorrect or we evaluated at weird grid points of course.
- Inversion is also tricky, and may not be possible.
- If the number of grid points is greater than the order of the polynomial, $\hat{\mathbf{f}} \neq \mathbf{f}$
- **Runge's Phenomenon**: Oscillating behavior. Imagine trying to approximate a simple curve with a 7-degree polynomial. It is like you're squeezing the line; it is going to oscillate. So, higher order doesn't always mean better fit and neither does having more grid points.

2.3.2 Solution to Interpolation Problems

- **Use Chebyshev polynomials** as the basis functions
- The grid points should be: $x_i = \cos(\frac{2i-1}{2N})$ for however many i.

2.3.3 Another interpolation Method: Discrete Cosine Transform

This is actually pretty cool and used for compressing and decompressing photos. One example is JPEG. With any photo, we have low frequency parts (like a flat color) or high frequency part (something with detail). What **DCT** does is, using cosine functions, they transform the image to look virtually the same but with less computation needed. It does this by reassigning weights to certain areas depending on frequency; sometimes even setting some weights to zero. The result is an image that is virtually unchanged and computationally less demanding. We visualizing this on a graph, imagine a smooth curve and then all of a sudden it becomes less smooth. The smooth part maybe easier to approximate since the behavior is not changing (low frequency), but for the less smooth part, because it is carrying a lot of uncertainty or rather, higher frequency, we apply less weight to control the possible sporadic nature that may occur if we did not adjust the weight.

In this case, we have grid points x_i and we transform them to $g(y)$. Let y be the chebyshev nodes, where also:

$$c_j(x) = \cos(j \arccos g^{-1}(x))$$

The inverse is taken for all x, but at the grid points x_i, the inverse is y_i. In the end, it does not matter what g is, but rather just know that $y_i = \cos(\frac{2i-1}{2N})\pi$ and the chosen grid values x_i.

2.3.4 Overall

Interpolation is simply easy and less computationally demanding. Naturally, $f(x_i) = f(x)$ for every grid point. The only difference between this approximation and the true function is the values in between the grid points. These values are evaluated as:

$$f_{i-1} + c_{i-1}(x)$$

where f_{i-1} is the function value at the previous grid point and x is an off grid point. This makes the function $\hat{f}$ piece-wise defined.

There are several kinds of interpolation, but lets focus on two:

- **Linear Interpolation**: Uses linear functions to connect the grid-points. We have: $f_i + c_i(x_{i+1}) = f_{i+1}$ where f_i is the evaluation of the function at x_i (which is a constant of course) plus this term:

  $$c_i(x) = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}(x - x_i)$$

  $c_i(x)$ is just the slope of the two grid points multiplied by the difference between the off grid point and the on-grid point. Together, these terms just make a linear function in slope-intercept form.

- **Cubic Spline Interpolation**: More or less the same thing here, except we have cubic spline piece-wise functions estimating the functional. Cubic implies we are dealing with three parameters, which also follow this condition: $f_i + c_i(x_{i+1}) = f_{i+1}$. Another condition required is **smooth pasting**. This means:

  $$c_i'(x_{i+1}) = c_{i+1}'(x_i) \quad \text{and} \quad c_i''(x_{i+1}) = c_{i+1}''(x_i)$$

  Cubic spline does run the risk of oscillation between grid points and is more tedious. Overall, both methods do really well for well behaved functions and in these scenarios, more gridpoints are better.

2.4 Eigenvalues, Eigenvectors and Eigenvalue Decomposition

Your typical setup to solve by method of eigenvalue extraction is:

$$Ax = \lambda x$$

If A has full rank then we can write $A = X\lambda X^{-1}$. From Macro I, you can simply raise the eigenvalue to a number n which describes the number of deviations you wish to calculate to find the deviations overtime as we approach the steady state. The eigenvalue of course must be less than 1; otherwise, it will not converge to 0 distance. When we have the greatest eigenvalue equal to 1, then the limit of A as we raise it to ∞ is:

$$A^{\infty} = X\lambda^{\infty}X^{-1}$$

$$[x_1, 0, ...0]X^{-1} \neq 0$$

2.5 Rootfinding

From your typical Algebra II class, you have a function - we set it to zero and find the roots, which are the points such that satisfy $f(x) = 0$. In this case, it is no different, just that it requires numerical calculation of the non-linearity of these systems. This is also analogous to a fixed point problem. The intermediate value theorem, given a continuous function where the end points are opposite signs, satisfies the existence condition for a root.

We have some approaches on how this can be done:

- **Bi-section Search**: To begin, this is for the univariate or 1 dimensional case. The approach requires some iterations of some process which ceases when it reaches some predetermined distance from zero. Quick note: Formally, the powerpoint uses y to capture the next candidate point, but I will use x_{c_i}. Here are two formulas you have to know:
  - **Solving for x-value**:$x_{c_1} = \frac{a+b}{2}$.
  - **Solving for step or distance to take**:$d = \frac{a-b}{2}$
  - **Process**Use $a = -2$ and $b = 5$. The idea is one finds the x-value to which to evaluate and the other finds the next step or distance to take.
  - The next step is to evaluate the function at the new point. First think about pinching someone's cheeks with both hands. Where you pinch are the points. The new candidate point x_{c_1} is evaluated to be 1.5. If this function is positive, look at where you're pinching and move your right hand to this new point and leave your left hand as is. We shrunk the distance between the two pinching points. Using the distance formula again, $d = \frac{d_1}{2}$, moving to the left since the evaluation was positive, we find a new candidate point x_{c_2}. If this point is evaluated to be a negative number, look at where you're pinching again. Move your left hand to this point. Now we are looking at a smaller window and can see precisely where the root might be. We continue iterating until the root is found.
  - We just keep evaluating until the distance to $f(x) = 0$ reaches that critical value near zero or similarly when $|x_2^{(n)} - x_1^{(n)}| < \varepsilon$.
  - The takeaway is that we can actually calculate how many iterations it will take to find the root. $n^* = \frac{\log d^0 - \log(\varepsilon)}{\log 2}$
- **Newton Algorithm**: This of course is for n-dimensional functions. Suppose we have some non-linear function. We perform a Taylor approximation of the function around some candidate point (a guess): $x^{(0)}$.

  $$f(x) \approx f(x^*) + f'(x^*)(x - x^*)$$

  or as we iterate,

  $$0 = f(x^0) + f'(x^0)(x^1 - x^0)$$

Using this formulation, we can find the new point to calculate our next candidate point. In this case:

$$x^1 = x^0 - \frac{f(x^0}{f'(x^0)}$$

. This approach can be problematic, so we introduce the problems and possible solutions known as the **Quasi Newton Methods**.

- **Problem #1: Going back to where you started**: This leads to no convergence. When taking a Taylor Expansion around some point, the line at that point will intersect the x-axis. This is your new guess. This is the intuitive explanation. What happens is, at this new x-guess, when you evaluate the real function at his guess to do a linear approximation, you'll find that where the line intersects the x-axis, this x point evaluated at the true function is the same point you started off at.
  - **Solution**: *Back-stepping*: If you feel or see that you have entered in such a non-convergent cycle, then you must adjust your x- values to get new evaluations and that should lead to a convergent sequence.
- **Problem #2: The Jacobian Matrix**: The Jacobian matrix is costly to calculate, especially if there are n-parameters to consider. Formally, we have:

$$0 = F(x^*) \approx F(x) + J(x)(x^* - x) \longrightarrow x^{n+1} = x^n - J(x)^{-1}F(x^n)$$

  - **Solution**: *Broyden's Method*: This Quasi-Newton Method replaces the inverse Jacobian matrix with an estimate B, which we can then continuously multiply with itself as we iterate i.e. $B^{(n)}$ for n iterations. This can be performed in Matlab with the steps below.
    - **Step 1**: Define all variables. $x^{n+1} = x^n - B^{-1}F(x^n)$ include: *xnow, xold, Bnow, Fold, Fnow*. Let *Bnow* be the estimation. $B^{(n)} = B^{(n-1)} + (Dx - B^{(n-1)})DF)\frac{(Dx'B^{(n-1)})}{Dx'B^{(n-1)}DF}$. We let the first approximation be the identity matrix.
    - Dx is the change in x: *xnow - xold*
    - DF is the change in F: *Fnow - Fold*
    - **Step 2**: Considering these elements, we are interested in the x-values and functional values as this equation approaches zero. Iteration reveals it.

2.6 Minimization Routines

Instead of finding the root, we take a similar approach minimizes the value of $f(x)$. In Matlab, this can be done using a solver. We first use Golden Search

- **In the 1-dimensional case, thus derivative free**: *Golden search* is like bi-section. We are trying to find the **minimum** of a function.
  Let x_1, x_2 be evaluation points and let x_c and x_m be in between x_1 and x_2. For simpler understanding, let $x_1 = a$, $x_2 = b$, $x_c = x_1$ and $x_m = x_2$.
- **Process**:
  - The *Golden Ratio* is $GR = \frac{1+\sqrt{5}}{2} \approx 1.618$.
  - Next, we find the distance. $d = GR(b - a)$
  - $x_1 = a + d$
  - $x_2 = b - d$
  - Evaluating the function, if:
    - $f(x_1) < f(x_2)$ then we remove whatever is outside of the greater evaluation, which is $f(x_2)$
    - $f(x_1) > f(x_2)$ then we remove whatever is also outside of the greater evaluation.
  - In our case, suppose we have the second condition, so we remove everything to the right (since $x_1 > x_2$). The greater evaluation point becomes our new $x_1 = b$. a does not change.

  - Then we recalculate the new d with different b and find new x_1 and x_2.

- **In the multi-dimensional case**: We must use the *Nelder-Mead Simplex Method* which is just FMIN-SEARCH(F,X0) on Matlab or BFGS which is FMINUNC(F,X0)

3 Part 2: Block 2 - Further methods to solve single agent decision problems

3.1 Policy Function Iteration

Also known as *Howard's Improvement Algorithm*, we apply the **policy function** forever, instead of once like in the value function iteration. We start off with an arbitrary policy, evaluate the value function at that policy, improve the policy by the sum overtime and evaluate the value function then again as a better guess until the policy can no longer be improved (or when the difference between policies reaches some critical value). Consider the following system:

$$h^{(n)}(s,z) = \text{argmax} u(s,s',z) + \beta E V^{(n)}(s',z')$$

$$V^{(n+1)}(s,z) = u[(s,h^{(n)}(s,z),z)] + \beta E V^{(n)}[h^{(n)}(s,z),z']$$

The aforementioned value function is evaluated at the argmax, which just replaces s'. In *policy function iteration* we do the same here:

$$h^{(n)}(s,z) = \text{argmax} u(s,s',z) + \beta E V^{(n)}(s',z')$$

We get our policy, except now, we take the discounted sum over time of our value function using this policy function.:

$$V^{(n+1)}(s,z) = E_0 \sum_{t=0}^{\infty} \beta^t u\left[s_t, h^{(n)}(s_t,z_t), z_t\right]$$

$$s_t = h^{(n)}(s_{t-1}, z_{t-1})$$

The above function has the predetermined state to be the policy function from the previous period and has added indices to states of the world. The idea is now we take this V^{n+1} and make this discounted sum our new guess. This cuts down on iterations.

The process for VFI is the following:

$$V^{(n+1)} = \overline{u}^{(n)} + \beta \Gamma^{(n)} V^{(n)}$$

$$V^{(n+1)} - V^{(n)} = \overline{u}^{(n)} - (I - \beta\Gamma^{(n)})V^{(n)}$$

The last line comes from subtracting $V^{(n)}$ from both sides. This is considered a newton method, since by subtracting from $V^{(n)}$ from both sides, we essentially look for a root. $\Gamma^{(n)}$ is our transition probability matrix here. $\overline{u}^{(n)}$ is the vector of payoffs $u[s,h^{(n)}(s,z),z]$ for all nodes/gridpoints under the optimal policy in iteration n. The process for PFI is:

$$V^{(n+1)} = \overline{u}^{(n)} + \sum_{t=1}^{\infty} \beta^t (\Gamma^{(n)})^t \overline{u}^{(n)} = (I - \beta\Gamma^{(n)})^{-1}\overline{u}^{(n)}$$

3.2 Multigrids

Thus far, all our algorithms attack the problem for a fixed and given resolution (think of just 1 grid of course). **Multigrid** methods speed up virtually any method and the idea is moving from one coarse grid to a finer grid. The solution generated using the coarse grid can then be used as a guess to run on a finer grid. The process is as follows:

- **Step 1**: We solve the problem through VFI. This helps us find $\hat{V}_0^*$. The index "0" here denotes the grid number.
- **Step 2:** We then increase the number of grid points for each dimension d (we typically work with 2 dimensions). The increase is by a factor of $2^{\frac{1}{d}}$

- **Step 3**: The initial guess *for the new grid* must come from the solution of the coarser grid. This solution would have been generated by interpolation.
- **Step 4**: With this guess, we perform the VFI on the new grid (which is finer thus larger) to get $\hat{V}_1^*$.
- **Step 5**: Repeat steps 2-4 until the grid is fine enough.
- **Something to note**: As we move from 1 grid to the next, since it is getting finer, our termination value must get smaller - smaller by a factor of 2.

3.3 Using the Euler Equation

For the past methods, we perform a VFI and extract the policy function. But what if I can find the policy function without actually performing VFI? The policy function is the only thing we care about. There are two methods to finding the policy function without performing VFI.

- Projection Methods: Uses the euler equation and then minimizes the error to find the policy function.
- Endogenous Grid

3.3.1 Projection Method

Consider the stochastic value function:

$$V(s,z) = \max u(s,s',z) + \beta EV(s',z')$$

Taking the first order condition with respect to s' reveals:

$$\frac{\partial u}{\partial s'}(s,s',z) + \beta E\frac{\partial V}{\partial s}(s',z') = 0$$

We leave it in this form to be general. The states can be anything, not necessarily the growth models from macro I. **Note that Bayer uses the following notation: taking the partial wrt to s is the first argument no matter what and s' is the second argument no matter what** Using the envelope theorem, which finds the value function under different parameter values, we arrive at the Euler equation:

$$\frac{\partial u}{\partial s'}(s,s',z) = -\beta E\frac{\partial u}{\partial s}(s',s'',z')$$

This is the inter-temporal optimality condition, which cannot account for occasionally binding constraints. Using the fact that s' is a result of the policy function, we have $s' = h(s,z)$. The point of this process is to exploit the fact that this relationship must hold and the simplification has allowed for the equation to be only in terms of s.

$$F(s) = \frac{\partial u}{\partial s'}(s,h(s,z),z) + \beta E\frac{\partial u}{\partial s}(h(s,z),h[h(s,z),z'],z') = 0$$

Just some notes: We are essentially finding the root of the Euler at all nodes k and z. In the code, we use **fsolve**, which will generate the values of *kprime* for which the euler error is near zero (depending on the critical value you set of course). *kprime* should be a vector, 1 for each state, thus 2 columns, where each row is for each grid point. Hence, the vector is of course increasing as the predetermined level of capital is increasing. The code labeled this the **Collocation Method**, which is not different generally from the one mentioned below. We need to find an approximation for this $h(.)$ function. Parametizing it is possible using chebyshev polynomials, where

$$\hat{h}(s,z) = \sum_{j=1}^{m}\sum_{i=1}^{n} \psi_{i,j} c_j(z) c_i(s)$$

Naturally, using this approach, we cannot expect for $F_{\hat{h}}(s,z)$ to always equal zero for all states since it is an approximation. Obviously again, we want it to be as close to zero as possible. Thus we **minimize the Euler error**.

$$\langle F,g\rangle = ||F|| = \int_A F(a)g(a)da$$

where g is some weighing function. We want these functions to be orthogonal. Using the terms above, we want:

$$\langle F(\hat{h}(s,z), g(s)\rangle = 0$$

From here, we have two types of projection method metrics to measure this error:

- **Least Square Metric**:
$$\min_{\psi}\langle F_{\hat{h}}, F_{\hat{h}}\rangle = \min_{\psi}\int [F_{\hat{h}}]^2 ds$$
This sets the weighing function to be the partial of F wrt parameters ψ, where ψ^* is the **argmin** of the metric.

- **Collocation Method**: This has the same goal: to find the $\hat{h}$ which solves the functional. This alternative metric uses the mass-point function as the weighing function. Defined as
$$\min_{\psi}\langle F_{\hat{h}}, \delta(X - X_i)\rangle$$
where if $X = X_i$, then the weighing function returns a 1 and 0 otherwise i.e. $g(0) = 1$. or rather:
$$F_{\hat{h}}(s,z) = F_{\hat{h}}(s_i, z_i)$$
δ is the Dirac delta function. Simply we zero out Euler equation errors (Set the residuals equal to zero) at certain specified points such as the grid points and we attempt to minimize the error of the the off-grid points. These grid points could be specified as chebyshev nodes (defined on $[-1,1]$ and thus, we have to find the ψ_i (the parameters) s.t. $F_{\hat{h}}$ is equal to zero. The integral here is not necessary to solve since the weighing matrix is defined in such a way that the result is just a system of n equations (the node equations) and n unknowns (the parameters). Thus, this reduces to finding the policies on grid with an interpolation rule off grid for $h[h(s,z), z']$

- **More Specifically**:
$$\hat{h}(s,z) = \sum_{j=1}^{m}\sum_{i=1}^{n}\psi_{i,j}c_j(z)c_i(s)$$
Where the function is evaluated first at every node for state $1 \in z$ and then at every node considering state $2 \in z$
$$F(X_1, \psi_{1,1}c_1(z)c_1(s)) = 0$$
$$...F(X_n, \psi_{n,m}c_m(z)c_n(s)) = 0$$

3.3.2 Endogenous Grid Method

Including the last method above, root-finding has been the premise. Instead, here, we make the grid endogenous and the policy exogenous. The idea comes from forward induction "Given the state I am in now, what must have been the state before if I acted rationally". In other words, all past play must have been the result of rational decision making. EGM is quite demanding, in the sense that it has greater restrictions: it may deal with occasionally binding constraints well, but requires differentiability of the value function everywhere, and monotone policy function (since the inverse must exist, hence, an isomorphism). Lets set up our problem:

$$V(s,z) = \max_{s' \leq b} u((1+r)s - s' + z) + \beta E V(s', z')$$

where we have s as savings, r as the interest rate return on our savings, and z being the stochastic income shock we get. FOC arrives at our Euler, which is:

$$u'((1+r)s - s' + z) = (1+r)\beta E u'((1+r)s' - s'' + z') + \lambda$$

where $\lambda = 0$ if $s' < b$ and if it binds, then λ can still be zero but also positive. This means the person is ok with binding at the borrowing constraint. Thus, overall, the grid is endogenously generated from a predetermined grid of values of end-of-period assets, k_{t+1}

The process is as follows:

- We start with a policy guess. $rk + z$ is a good guess. This is called **rolling over assets**. Essentially, we just consume based on our income and the returns from our income and "roll over" the assets of today to the next day. The iteration creates a vector of optimal consumption policies.
- We thereby get values for the RHS of the this equation for each grid point, s'. $(1+r)s^*(s', z) = s' - z + c^*(s', z')$
- Then we solve for s^* given c^* and z and s.t. the optimal condition above holds for each s'. This is the forward induction reasoning.
- This generates pairs $\{s', (s, z)\}$, or rather a mapping from s' to s^*. This inverts our grid, which is reflected in the code. We interpolate in between the s^* to find the off grid values.
- At these s^*, the budget constraint binds, which was mentioned above.
- Go back to step 2 and continue iteration
- **Overall**: we form an exogenous grid for the policy function or rather, the future state variables, then invert the utility expressions RHS above to derive the c^* and use that to find pairs s' which is exogenous and (s^*, z) which is endogenous. The s^* are typically off-grid. Imagine that they are functional values, where the functional is the linear interpolant result.

To illustrate, let $c^{(n)}$ be the consumption policy function at some iteration n. The resource constraint can be seen as:

$$(1+r)s^*(s', z) = s' - z + c^*(s', z')$$

where your *endogenous grid* is the LHS and you have resources left plus consumption after. This is our **budget constraint**. Then we have the inverse of the utility function to find the consumption policy:

$$c^*(s', z) = u'^{-1}\{(1+r)\beta E u'[c^{(n)}(s', z')]\}$$

We use this consumption policy and plug into the resource constraint above to solve for s^* given grid points s'. Thus, with the FOC, the budget constraint and the future guesses, we have pairs $(s', s^*(s', z))$ with which we can interpolate as before to generate an approximation for s^* for off grid points. This is then a mapping from the exogenous grid points s' to the endogenous s^*. The FOC lets us find the inverse where we find c, then we use the budget constraint and values c^* to solve for the state s^*.**Note**: For any $s < s^*(s'_1, z)$, households would like to choose assets lower than the borrowing limit, which is not possible so we set their policy to the borrowing constraint, setting $s' = b$. Households cannot have assets lower than the first s^*. That implies that they are consuming more and not saving enough. thus, they must go to the borrowing constraint to increase their asset levels to match the s^*. This happens because we are solving for s^* and we do not really know how that may correspond to the grid we already have in place. So for those starting grid points below the s^*, we flag and make sure they go to the borrowing constraint. The next update then becomes $c^{(n+1)} = (1+r)s^* + z - b$.

4 Stochastic Fluctuations and Markov Chains

Modern probability theory studies chance processes for which the knowledge of previous outcomes influences predictions for future experiments. All along we have been working with these sort of influences. Of one particular type are Markov chains. More formally, suppose we have a sequence of stochastic variables X_t, which take on values from our grid/state space. The movement from one state to the next, X_i to X_j is captured by the probability matrix P, where P_{ij} constitutes the probability of moving from state i to state j. Each row of the probability matrix sums to 1, so think of each row as being a probability mass function over n possible states. This is the tale of markov chains:

$$P(X_n | X_{n-1}, X_{n-2}, X_{n-3}, X_{n-4}, X_{n-5}...) = P(X_n | X_{n-1})$$

They have finite memory. All movements from one state to another state is captured by the *transition matrix*. To illustrate, suppose I am in state 1 today and I want to be in state 3 in 2 days. Each day the state is realized and the weather starts. What is the probability of it being snowy in 2 days, given state 3 means snowy?

$$p_{13}^{(2)} = p_{11}p_{13} + p_{12}p_{23} + p_{13}p_{33}$$

This is also the dot product of the first row and the third column of the transition matrix. Formally, you can express it as:

$$p_{ij}^{2} = \sum_{k=1}^{r} p_{ik}p_{kj}$$

where r denotes how many states there are. Notice how the inner indices are the same. This is how the process is done for 2 days ahead. What if we are considering the probability of being in state i at time t? Does it matter where we start from? If you have a result where after n steps, the answer is independent of where you started (meaning, the same probability distribution is generated after n steps) then you have a *regular chain.* Thus, we have P, and we have probability vectors (the rows) u as our starting distribution. After n steps, the probability that we are in state i is the ith entry in the vector $u^n = uP^n$. u^n just captures the distribution of the states, and u is the original probability vector. In the end, we reach the **stationary distribution**, which is $\pi = \pi P$. Large values in the diagonal shows strong persistence. To continue, there are certain types of Markov chains. Of main discussion will be the **ergodic** or **irreducible** Markov chain. This type of Markov chain means it is possible to go from every state to every state, but not necessarily in one move. It is just possible to move around and not get "stuck". **Aperiodic** means that at some point, so it can be in irregular intervals, the probability of returning to yourself is positive. So after sufficiently many iterations, if the probability to go from one state to another is strictly positive, then we have both irreducibility and aperiodicity. These have **unique** stationary distributions. Relating this back to our methods, stationarity and dependencies between states is exactly what we need to describe planning problems in recursive form. Thus, Markov processes are a must! In the cases we have been working on, we have a finite and discrete grid. This makes expressing conditional expectations much easier. To motivate, Markov chains really illustrate the states of the world, and reveals how long we may possibly stay in a state.

4.1 Approximating Continuous AR(1) processes using Discrete Markov Chain Approximation

This is what we've been doing - using a discrete grid, where the future state solely depends on the current state (thus recursive). We want to discretize the AR(1) process over N grid points x_i, where we partition the real line into bins (think of integration through riemann sums). Think of for example productivity. In the matlab code, we always discretize to 2 states; but, we can represent productivity as an AR(1) process. As always, from just looking at our value function, we want the expectation of some state given that we are at some discretize income state, but in the case where we are conditional on the AR(1), thus continuous, stochastic variable, how do we approximate? For our assignment (problem 9), we had to compare moments of this three state markov chain versus the moments of the continuous AR(1) process.

To continue precisely, we can approximate by taking the integral as (thus a **quadrature problem**, which means the process of determining area, thus integrals):

$$\int_{X_i} F(y')\phi(y'|y \in X_i) \approx w_{ji}F(x_i)$$

. You can visualize w as the base of a rectangle and $F(x_i)$ to be the value or height of the rectangle evaluated at each discrete state/grid point. To perform this approximation, there are 3 steps:

- **Step 1: Choosing a partition**: We must choose the grid/states to consider in our approximations of continuous processes. For example, which states to pick in analyzing the distribution of economic states (i.e. growth, mild recession, or severe would be a possible set). There are two types of partitions discussed
  - *Linear Grid*: we take the $\alpha\%$ mass interval and use equidistant points. This means truncate our distribution, maybe to 3 standard deviations. This cuts off the tails of course, which may be bad depending on what we are analyzing.

  - *Importance Sampling*: We split the grid into bins such that the likelihood of the bins are equal, thus $\frac{1}{N}$, in the ergodic distribution. Imagine your probability vector being 5 states and their unconditional expectations have the same probabilities. The tails are again not emphasized.

- **Step 2: Choosing the representative element**: Imagine a normal distribution, where we already divided the distribution into our bins. The representative element is suppose to capture the point where the curvature is the greatest or when the pdf is the greatest. This is denoted as the expectation. For simplicity, you can just imagine this as the middle between two bin bounds. Formally, the "middle" and "the expectation" respectively are:

$$x_i = \frac{\min(X_i) + \max(X_i)}{2} \quad x_i = E(x|x \in X_i)$$

  The choice of partition and representative element has to reflect the problem. In these partitions, consider the effect on the tails of the distribution. The grid centers or x_i must be those grid points that represent areas we reach often. With this method, you will find most grid centers within the first 2 standard deviations since this where the probability of being in such a state is high.

- **Step 3: Choosing the weights**

4.1.1 Importance Sampling

As previously mentioned, we have some bin bounds, which are elements of the grid X_i. These bounds are determined by the likelihood.

$$P(x \in X_i) = \Phi(y_{i+1}) - \Phi(y_i) = \frac{1}{N}$$

. Thus the probability between these two bounds must be $\frac{1}{N}$ in the ergodic distribution, where the bounds themselves do not have to be equidistant of course. Φ denotes the cdf. The cdf is the cumulative distribution; thus, $\Phi(y_{i+1} = \frac{i}{N}$, where as i increases, the cumulative distribution, of course, approaches 1. Taking the inverse, helps us find the bound points. $y_i = \Phi^{-1}(\frac{i-1}{N})$, where at $i = 1$, we have zero probability since this is first, leftmost bound. To find the "centers" of the bins, we take the conditional expectation of x_i, which is:

$$x_i = E(x|x \in X_i) \longrightarrow x_i = \frac{\int_{y_i}^{y_{t+1}} x\phi(x)dx}{P(x \in X_i})$$

The conditional expectation, where the condition is the space, is expressed in this way, where the denominator is the probability space the expectation depends on. Since $P(x \in X_i) = \frac{1}{N}$, we get that

$$x_i = N[\phi(y_{t+1}) - \phi(y_t)]$$

4.1.2 Choosing Weights

The weights are the base of the rectangle. In the context of our grid space, we need to **calculate the transitions** - moving from one bin to another. What is the conditional probability of moving from one bin to another bin?

$$p_{i,j} = \Phi\left(\frac{y_{j+1} - \rho x_i}{\sqrt{1-\rho^2}}\right) - \Phi\left(\frac{y_j - \rho x_i}{\sqrt{1-\rho^2}}\right)$$

To be precise, this is max shock and min shock, respectively, divided by the variance of the normally distributed error term.

4.1.3 Simulating a Markov Chain

Now that we found our probability matrix by finding the grid centers and bounds based on being equidistant or representing the greater probabilities, we can simulate a Markov chain.

1. We obtain the probability matrix.

2. Set t= 0

3. Pick any state $X_t = i$ to begin from.

4. For the remainder of time, for $t = 1...T$, draw from a uniformly distributed $(0, 1)$ variable u_t. Suppose we start off in state 1. Then we generate the second state (assuming we have just two states) from the uniform distribution with the probability vector equal to the row we obtained above.

Example of a Simulation P= $\begin{pmatrix} .30 & .70 \\ .50 & .50 \end{pmatrix}$
Suppose our state today, X, is $X_1 = 1$, where our state space is $X = \{1, 2\}$. We wish to simulate X_2. We have to generate a r.v. X distributed like the second row of $\mathbf{P}$ and set it equal to X_2. To generate such an X, we have to use the **inverse transform method**: We generate a uniform distribution. Let $X = 1$ or let the state stay 1 if $U \leq \mathbf{P}_{21} = .50$ or Let $X = 2$ if $U > .50$. We then choose and continue simulating based on the state. In the code, we can speed this up by running random draws at the beginning, instead of at each iteration.

How can we use this now?
The process of finding a probability matrix and simulating can actually be cut in half. Let me explain **Policy functions inducing a Markov chain**. Suppose we have interpolated our value function guesses and found the policy functions which maximized our value functions for the capital and income grid states we defined. Consider again our state space. It is defined by grid points. Let i^* denote the index of the next smallest element in S relative to s^*. If our grid starts with 0.3 and $s^* = .34$, the $i^* = 1$. 1 for the first grid point in our state space. Essentially, $s^* \in [s_i, s_{i+1})$. In the code, s^* could be anything, so to make sure it stays in the grid, we set the value of the index $s^* < s_1 = 1$ thus $s^* = s_1$. Likewise, for the $s^* > s_n = 99$ we give it the index 99. This whole index dilemma is needed to define **linear interpolation weights**. To be specific, let $w(s, z) = \frac{s^* - s_{i^*}}{s_{i^*+1} - s_{i^*}}$ be the weights. The idea here is to reform our value function, where the value function guess is a convex combination of the value function at s_{i^*} and s_{i^*+1} for some income state in the future. So we assign weights to our state today and state tomorrow, whose probabilities sum to 1 (think of bounds here as well). To find the stochastic matrix Γ, we have to multiply the weights by the probability matrix defined by the productivity states. So for each income state, we have a value function guess which a convex combination of the state today and tomorrow purely defined on the grid of s!

$$V_t = \max u(s, s', z) + \beta \Gamma_{s'} V_{t+1}$$

This actually gives us the ergodic distribution of states for the planning problem without simulating. We just linear interpolate, find weights, have an exogenous probability matrix for our process for productivity, and make the iterations.

5 Heterogeneous Agent Economies

Typically in rudimentary economic courses, we have the case where all agents are identical. These are described as the **representative agent problems**. Consider a heterogeneous agent model here, where we have agents reflecting the spectrum that is the Wealth Distribution. The moments of the wealth distribution determine demand. Let Aggregate demand depend on the distribution of wealth $\mu(W)$. Thus, prices $p = p(\cdot, \mu(W))$ also depend on the distribution of wealth. The trickiness here is the infinite dimensional state variable: **the wealth distribution function**. Also note, aggregate behavior is the result of market interactions among a large number of agents who have different wealth (because they endure different idiosyncratic shocks). Taking the expectation over the agents in this ergodic distribution would lead to aggregate capital supply (I'll explain below). We will first consider an economy with no aggregate fluctuations, where prices are constant. In these incomplete markets, we are solving for the equilibrium interest rate. Recall also that now consumption depends on history since we are not fully insured.

5.1 Consumption-Savings models with Heterogeneous agents

Suppose we have households supplying funds

$$K^S(P) = \sum_i k_i \mu^*(k_i, P) = E_{\mu^*(P)}(k)$$

What households supply depends on *prices* and the RHS is just the sum over all agents capital stock times the respective ergodic distribution of each agent (depending on which state they are in). This is essentially equivalent to the expectation of capital supply over the ergodic distribution, which spans the state space of course. This equation holds for the following models. What changes is *demand* for each model.

$$K^d = \left(\frac{R+\delta}{\alpha}\right)^{\frac{1}{\alpha-1}} N$$

This is for the aiyagari model. This is solved by solving for capital in the following equation:

$$r + \delta = \alpha k^{\alpha-1} N^{1-\alpha} = MPK$$

5.1.1 Aiyagari Model: the asset is physical capital

In general, in incomplete models, we have ex-ante homogeneous agents facing idiosyncratic risk and for the time being, no aggregate risk. Suppose we have 2 income states, really employment states, which dictate your earnings. Wages are ws_t, where $s_t = 1$ for employed, 0 otherwise so of course you do not receive wages. The combination of income and assets evolves jointly as a discrete markov chain. Prices remain constant since we just consider the stationary distribution of assets. And let $\beta(1+r) < 1$ for a solution to exist. Tersely, the Aiyagari model requires:

$$F_K(K, L) = r + \delta = MPK \longrightarrow F_L(K, L) = w = MPL = \text{wages}$$

$$c_t + k_{t+1} = ws_t + (1+r)k_t$$

The capital letters denote the aggregate. Aggregate labor supply is exogenously chosen, thus we have prices MPK and MPL being completely determined by *capital*. In this case, $K^S(P(K)) = K$. K is equal to the sum over all individual asset holdings times the probability of being in some asset state and income state.

$$K = \sum_{i,j} \overline{\lambda}(i,j)_{r,w} k_j$$

Aggregate labor is given as already mentioned, but is calculated as the long-run distribution or the ergodic distribution times the probability matrix of income states.

$$N = \sum \overline{\pi}_i s_i$$

Aiyagari notes:

- Aggregate behavior is the result of market interactions over a large number of people among which suffer idiosyncratic shocks. These models have individual dynamics from what we have seen in representative agent models, but not the aggregate dynamics. Let the idiosyncratic shocks be the uninsured labor endowment (thus we do not know whether we will have a job tomorrow. This lends to the implication that consumers smooth consumption heavily and carry tons of capital. this is known as **precautionary saving**).
- **we seek to determine the real interest rate and wage that clears the market, the law of motion for aggregate capital and the wealth distribution**. Also, we'd like to learn more about how individual risks and uncertainties affect aggregate savings. Thus, the sources of aggregate savings. Of course, bequests, inheritances, or interchangeably, the size of your endowment also play a role in aggregate saving, but nonetheless difficult to isolate.

- the steady state is some equilibrium interest rate, and since shocks are purely idiosyncratic, they "go away" in the aggregate. Naturally, also, the interest rate will be the net marginal product of capital (subtracting depreciation expense) and the aggregate amount of capital will be equivalent to the sum of all ind asset holdings.
- comparative statics on risk aversion, persistence of earnings and variability illustrate the effects on aggregate savings.
- literature and common sense finds that liquidity is important for those in the lower wealth distribution, while those at the top enjoy stock returns, significantly greater than the risk free rates. Because of the state of their wealth, of those that invest, those on the lower distribution stick to low-return, non-risky assets while those at the higher exhibit the opposite. under complete markets, risk aversion is the same, hence this type of portfolio dispersion would not exist.
- let earnings be MPL times labor supply (which is in hours)
- essentially, with the borrowing constraint, and no insurance markets for the uncertainty on earnings, we have the interest rate lower than the time preference rate (i.e. $\frac{1-\beta}{\beta}$ which in words is the intensity of our desire to satisfy our wants now divided by satisfying our wants in the future.) . Having low time preference means patience and saving behavior, which means money is in the bank. So much money accumulates, which makes it cheap to borrow. Thus, interest rates drop. This could possibly mean, if people are very patient, and the return is less than this time preference rate, they are risk averse. This means people will save more and aggregate savings will thus increase.
- having a negative interest rate means people can borrow and payback less. Thus, we must set a bound for the borrowing limit (otherwise, the limit is infinity for consumers consumption). We have to avoid ponzi schemes of course.
- **Putting everything together:**
$$E_0\sum_{t=0}^{\infty}\beta^t U(c_t)$$
and our resource constraint: $k_{t+1} + c_t = (1+r)k_t + wl_t$ and $c_t \geq 0$ with $a_t \geq -b$ and $r = MPK - \delta$
- households are identical in their preferences, the markov process decides their income (employment) and the prices they face are also identical. What is different is of course the stochastic process that is their employment opportunities and thus their capital accumulation. The behavior of these households determine the interest rate and wage.
- Lets assume an initial distribution across households as $\lambda(k, s)$. The average level of capital *per household*, K, satisfies
$$\sum_{k,s}\lambda(k,s)g(k,s)$$
, where $k' = g(k, s)$.
- Average Labor supply is $N = \xi_\infty s$, where ξ_∞ just captures our transition matrix to the ∞ power to make the **invariant distribution**. We multiply that by the grid of our employment/income/productivity states.
- The production function is $F(K, L) = AK^{1-\alpha}$. the MPL is the wage and the MPK is return on capital.
- **Equilibrium**: A stationary distribution is a policy function $g(k, s)$, aprobability distribution $\lambda(k, s)$ and positive real numbers (K, MPK, MPL)
  - prices satisfy $w = MPL$ and $r = MPK - \delta$
  - the policy function solves the household problem

– the probability distribution λ satisfies

$$\lambda(k', s') = \sum_{s,k} \lambda(k, s)\pi(s, s')$$

– the average value of K is implied by the average of the households' decisions

$$K = \sum_{k,s} \lambda(k, s)(s, s')$$

– **Computation**:

- The next step is to see the impact of the interest rate. Naturally, the interest rate if negative runs the risk of abuse under ponzi schemes. Now, if its positive, if the interest rate is positive, we require that the present value is positive. That, the lim $\frac{k_t}{(1+r)^t} \geq 0$. Thus, the period by period borrowing constraint is $a_t \geq \frac{-wl_{min}}{r}$ which comes from consumption when we roll over assets (we save our assets and just consume based on our income and the return on our assets.) Sometimes, the borrowing constraint is not binding, so we replace by the smaller amount equal to wl/r Thus, $k_t \geq -\phi$ where for $r > 0$ and $r \leq 0$ we have:

  $$\phi = \min[b, wl_{min}/r] \qquad \phi = b$$

  Essentially ϕ is a function of wages, labor and the interest rate.

- resources today are: $z_t = wl_t + (1+r)k_t - r\phi$, where we must pay the interest on whatever we borrowed. $\hat{k_t} = k_t + \phi$ shows our assets today plus what we borrowed.

5.1.2 Hugett Model: Risk free rate puzzle, a model with credit

:
Suppose again we have incomplete markets, individuals cannot self-insure from the i.i.d. idiosyncratic shock to labor, so these consumers save a lot for in cases of downturns. A lot of savings will drive the risk free rate down essentially, since too many people are willing to do it. What is different in this model is that **only one** asset can be traded: *a one-period, un-contingent bond.* This is just **a claim** to one unit of consumption in one period's time, accessible through some credit-central authority. There are no outside assets, thus the **net bond position across all households is zero**. Households maybe borrowers or lenders, but the average bond position must be zero. We have the following problem:

$$V(a, y) = \max_{a' \geq \underline{a}} \left(U(c) + \beta \sum_t V(a', y')\pi(y', y) \right)$$

$$c + qa' \leq a + y$$

where the value function is defined for some stock of bonds, a_t, and income/initial endowment, y. Think of π as capturing the transition probabilities between income states. The resource constraint shows that what I consume today and plan to take in bonds tomorrow for some price q has to be less than the bonds I have today and the income I receive today. Of course, our bond choices will be restricted to a grid. We describe the households as a point in the joint distribution of AxY, which is our state space of assets and what we are endowed with. The distribution, μ_t, reveals how much of the population are in this state of the world and also, the likelihood of someone experiencing this point. **The stationary recursive competitive equilibrium** is one where, given a bond price, q, we iterate the value function to find our policy rule. The stationary distribution μ is induced via the exogenous Markov chain for y, y' ans the policy function. In the end, the bond market clears, where

$$\sum_a \sum_y g(a, y, q)\mu(a, y, q) = 0$$

describes the sum over all asset positions for all incomes is zero. The solution is a policy function $a' = g(a, y)$ that induces the stationary distribution $\mu(a, y)$. Increasing the bond price will naturally drive down demand (until zero, which is what we want). Iteration follows this process where we make a guess for the bond price, see that demand is positive then we increase our guess for the bond price and continue doing this process until we have **no excess demand**. Looking at some comparative statics, as we increase the borrowing constraint (more negative), the less likely we are as households to really this constraint. As such, the annual risk free rate and the bond price approach the representative agent benchmarks of 4% and $q = \frac{1}{1+r} = \beta$. As the borrowing constraint gets tighter, consumers will not be able to borrow as much, and must create these saving buffers to insure against these idiosyncratic shocks. Bond prices also increase since we experience negative risk free rates. The negative rate is dissuade people from getting bonds.

5.2 Heterogeneous Agent Economies Methods with Aggregate risk

So above, we assumed that idiosyncratic risk withered away in the aggregate, and in the aggregate, there was no uncertainty. Now we consider an economy with aggregate risk. For simplicity, we assume agents take on a finite number of "types" and thus, we discretize the aggregate state space. It'll be defined by some state vector S. The recursive equilibrium of such a model is:

- price functions dependent on the state $P(S,\xi)$
- An individual policy function that solves the DP problem $h(s,S,\xi)$
- A law of motion for $H(S,\xi)$ for the aggregate economy such that
  1. all agents optimize prices and H as given
  2. markets clear
  3. the aggregate law of motion and the individual policy functions are consistent $H(S,\xi) = h(S,S,\xi)$

5.2.1 Parameterized Expectations

As the title dictates, this is a projection method, where we attempt to project the expectation of our continuation value onto some space of parameters. This space of parameters should be able to minimize the squared errors but essentially, we are trying the aggregate policy $H(K,z)$ in this continuation value. Suppose we have our typical capital accumulation problem, except firms and households are separated. Our value function is:

$$V(k,K,z) = u_t\left(r(K) + (1-\delta)k - k' + \pi\right) + \beta EV(k',K',z')$$

where $r(K) = \alpha z K^{\alpha-1}k$. This is just the MPK times the capital units I have today. π is the share of profits the household receives $\pi = (1-\alpha)zk^{\alpha}$, which is just a percentage of the production function. Generally, the idea is to find the policy function $S' = H(S,z)$ or in our case $K' = H(K,z)$ and we do this through parameterization (linear or spline). The idea is we posit a functional form of this policy. Before we get into how we do that, lets consider the full model. Consider the neo-classical growth model with CRRA utility function.

$$E_0\left[\sum_{t=0}^{\infty}\beta^t \frac{c^{1-\gamma}-1}{1-\gamma}\right]$$

$$c_t + k_{t+1} = zk^{\alpha} + (1-\delta)k_t$$

$$\ln(z_t) = \rho\ln(z_{t-1}) + \sigma\varepsilon \quad \varepsilon - N(0,1)$$

We have to find the policy for our control variables: consumption and capital tomorrow. Recognize that the control variables are just functions of the state variables. For example, $c_t(k_t,z_t)$ and $k_{t+1} = k(k_t,z_t)$. These *functions* have to respect the first order conditions and budget constraint above..

$$c_t^{-\gamma} = E_t\left[\beta c_{t+1}^{-\gamma}(\alpha z k_{t+1}^{\alpha-1} + 1 - \delta)\right]$$

The issue then arises: how do we posit this functional form for the policy function when the space of functions is so vast? When we have **complete depreciation** and **log-utility** ($\gamma = 1$) then we have an analytical solution where $c_t(k_t,z_t) = (1-\alpha\beta)zk^{\alpha}$ and $k_{t+1} = k(k_t,z_t) = \alpha\beta zk^{\alpha}$. Consumption is a function of the state variables and since consumption and capital tomorrow are our only choices, they are both simply proportions of the output and split among each other by the parameters we have defined. This functional form satisfies the first order conditions and is justified by the **Weierstrass Theorem**, where for any continuous function on a closed interval, there exists an arbitrarily close function such that for some $\varepsilon > 0$, we have

$$\forall x \in [a,b], |f(x) - p_n(x)| \leq \varepsilon$$

Thus the class of polynomials is dense in the space of continuous functions. Reducing the function to this approximation will change the problem from finding a functional form to finding a finite number of coefficients for this n-order polynomial. Such approximations are taylor expansions and as we increase the order, we analyze how close we are to a solution.
Returning to our example, we have the Euler equation which forecasts our consumption tomorrow. The RHS of the Euler is equal to just the marginal utility of consumption; thus a function of consumption and consumption, as we saw, is a function of the state variables. Thus:

$$c(k_t, z_t)^{-\gamma} = g(k_t, z_t) = E_t\left[\beta c_{t+1}^{-\gamma}(\alpha z k_{t+1}^{\alpha-1} + 1 - \delta)\right]$$

$$c(k_t, z_t) + k(k_t, z_t) = (1-\delta)k_t + zk^{\alpha}$$

We can pick estimating the conditional expectation, the conusmption function and the capital function. Finding one will solve for the other. Yet, **we pick the choice which resembles a low order polynomial**. The reason for it, is because if we were to pick a random function, the remaining functions would actually be non-linear and the chosen function will be approximated. But if the chosen function is high order polynomial, then we lose. Thus, by making an educated choice of the functional form of the functions, we can minimize the error overall. This of course also depends on the model; where the choice might not play such a significant role.
Lets consider the analytical solution for a bit:

$$g(k_t, z_t) = c_t^{-\gamma} = c_t^{-1} = \frac{1}{(1-\alpha\beta)zk^{\alpha}}$$

$$= exp\{-\ln(1-\alpha\beta) - \alpha\ln(k_t) - ln(z_t)\}$$

This is equivalent to saying I want to estimate:

$$exp\{P_1 \ln(k_t), \ln(z_t); [-\ln(1-\alpha\beta), -\alpha, -1]'\}$$

Which is just saying I want to make a first-order approximation of the polynomials dependent on the logarithms of k_t and z_t, my state variables, given the vector of parameters.

5.2.2 The Algorithm

Given the structural parameters, the time series of realizations of the aggregate productivity process $\{\varepsilon\}_{t=2}^{T}$, an initial value for aggregate productivity, z_1, and k_1 plus the vector of coefficients, we can solve for a time series of consumption today and capital tomorrow.
The process

1. **Simulate**: Let F be the Euler RHS as above. We want to simulate this for some T.

$$F_{t+1} = E_t(\beta c_{t+1}^{-\gamma}(\alpha z k^{\alpha-1} + 1 - \delta))$$

2. **Updating Coefficients**: So as previously mentioned, we want to parameterize this RHS. We can express this as an econometric equation:

$$F_{t+1} = g(k_t, z_t) + u_{t+1}$$

   We attempt to approximate $g(k_t, z_t)$ by

$$F_{t+1} = exp(P_n(\ln(k_t), ln(z_t); \eta_n)) + u_{t+1}$$

   Simple OLS retrieves these coefficients

3. Of course the coefficients are the ones that minimize the residual sum of squares. In updating these coefficients, consider this equation:

$$\eta_n^{i+1} = \lambda\hat{\eta}_n + (1-\lambda)\eta_n^i$$

This says that the next iteration for some polynomial of order n is equal to some convex combination of the current argmin for the time series, and the current parameter value for the current point in time. This is similar to the kalman gain approach.

4. **Stopping Rule**: $max(\hat{\eta}_n(j) - \eta_n^i(j)) < .0001$

In our code, we first use EGM to find the sequence of consumption that satisfies the Euler equation. Then we use this sequence in the budget constraint to solve for the endogenous grid today given tomorrow's capital stock and income today. But here, we use kstar as our grid and interpolate over the grid tomorrow to find assets/savings tomorrow. The values generated are assets tomorrow which is then our kprime. We then find the policy function where $H(K,z) = h(k,K,z)$. The aggregate policy becomes h evaluated at the meshes. We compare that to our guess aggregate policy and generate a difference. The new aggregate policy is actually a convex combination of both. We then iterate again.

5.2.3 Perturbation Methods

Essentially, economic models can be written in the form of a non-linear difference equation. Meaning, we just set all non-linear equations equal to zero. This can be simplified as:

$$E_t F(s_{t+1}, s_t, c_{t+1}, c_t, \epsilon) = 0$$

where F is our system of equations expressed as functions of the state variables today and tomorrow and the control variables today and tomorrow. We must take expectations over them! The point is to take the expectation of the local approximation of this system. In the matlab code, we compare the social planner policy, the parameterized expectations approach and the perturbation method and they are all the same. The values get even closer as they approach the steady state.

6 Heterogenous Agent Economics with Aggregate Fluctuations

The distribution of wealth, before, would reach some steady state over time. Now, with aggregate fluctuations/uncertainty, the distribution of wealth will change over time (it is endogenous). The state space is thus infinite. The aggregate state variable, which is the distribution, truly can be anything. The **Krussell-Smith Algorithm** essentially reduces this infinite dimensional object by approximation. The idea is *boundedly rational parameterized expectations.* Lets illustrate with an example. Lets take our **Huggett Model** but with aggregate risk. Agents only care about prices and in this model of credit, we have just the real rate to care about. They're interested in forecasting future prices. They form **adaptive** expectations. They form these forecasts by only using the finite moments m of the distribution and some parametric model explained below.
Model Components

- *Income*: $y_{it} = exp(\xi_{it} + Z_t) + \tau$ is denoted as having an aggregate component Z_t which affects households equally and an idiosyncratic component ξ_{it}.
- *Idiosyncratic Component*: $\xi_{it} = \rho\xi_{it-1} + \sigma_\xi \varepsilon_{it}^\xi$
- *Aggregate Component*: $Z_t = \rho Z_{t-1} + \sigma_z \varepsilon_t^z$
- both components above are continuous
- **Value function**:

$$V(s,\xi,Z,\mu) = \max_{s' \in S} u[s + exp(\xi + Z) - \frac{s'}{1 + r(Z,\mu)}] + \beta E V(s', \xi', Z', H(Z,\mu))$$

The value function considers the entire distribution which is impossible to measure. Krusell and Smith, thus, simplify it by having agents make expectations over the moments of the distribution g_m **and also** forecast prices g_r. We thus parametize the distribution.

$$V(s,\xi,Z,m) = \max_{s' \in S} u[s + exp(\xi + Z) - \frac{s'}{1 + g_r(Z,m)}] + \beta EV(s',\xi',Z',g_m(Z,m))$$

We have also simplified $r(Z,\mu)$ to some function g, which is now considered the pricing rule and likewise, the same was done for the perceived law of motion for μ, $H(Z,\mu)$. It is now the perceived law of motion for m of course.
For the household, each period, the actual policy is:

$$h(s,\xi,Z,m,r) = argmax_{s' \in S} u[(1+r)s + exp(\xi + Z) - s'] + \beta E\overline{V}[s',\xi',Z',g_m(Z,m)]$$

This is the policy for the boundedly rational or limited information equilibrium. Limited information since we are *limited* to the moments of the distribution. We do not know the entire distribution.

The process in matlab:

1. we choose a set of moments to characterize our distribution of states
2. Draw a sequence of states for Z_t before (because it is efficient) and then choose an initial guess for our distribution. In the matlab code, we chose the representative agent under no aggregate fluctuations as our initial guess.
3. At the end, we ran a regression to derive values of $KPrime$, estimated by a constant of length the size of the mesh, the mesh of productivity and aggregate capital. The functional form selected for $g = KPrime$ is log-linear and we find initial parameters β_0 from Aiyagari model without risk. We use the values as our guess for the Krusell-Smith model.
4. We then run Aiyagari's model to solve for the equilibrium interest rate.