Topics in Macroeconomics

Lecture 3: Function Approximation, Neoclassical Model in Discrete Time, and Complete Markets

Diego de Sousa Rodrigues
de_sousa_rodrigues.diego@uqam.ca

ESG-UQAM

Fall 2025

Road Map

1. Global approximation.
2. Interpolation.
3. Solving Neoclassical Growth Model using hybrid root finding method (`fsolve`).
4. Complete Markets.

1. Global approximation

The goal

- Suppose $y = g(x)$ but $g(\cdot)$ is unknown. Observed data: $D = \{(x_0, y_0), \ldots, (x_n, y_n)\}$ with $y_i = g(x_i)$ and $x_i \neq x_j$ for $i \neq j$.
- **Task:** Find a function $\hat{g}(x)$ that approximates $g(x)$ as closely as possible.
- **Two broad types of global approximations:**
  - **Regression:** some information about $g(x)$ pins down $n < m$ free parameters that generate an approximation.
  - **Interpolation:** some information about $g(x)$ pins down n free parameters that generate an approximation.

	info known	**points x_i**
Regression	$g(x_i)$ at $m > n$ points	given by data
Interpolation	$g(x_i)$ at n points	selected

Global approximation

Typically we approximate a function $g : [a, b] \to \mathbb{R}$ by:

$$\hat{g}(x) = \sum_{j=1}^{m} c_j \, \phi_j(x),$$

where:

- m is the degree of interpolation,
- $\{\phi_j\}$ are basis functions,
- $\{c_j\}$ are basis coefficients.

Regression

- **Regression analysis** looks for $E[y \mid x] = \hat{g}(x)$ with $y = g(x) + \varepsilon$.
- It solves:
$$\min_{\theta} \sum_{i=1}^{n} \left(g(x_i;\theta) - y_i\right)^2.$$
The resulting $\hat{g}(x)$ is parameterized (e.g., by estimated coefficients in a linear model).
- Notes:
  - A large number of observations is typically needed for a good fit.
  - If you can choose data points (numerical analysis), interpolation theorems can yield very good approximations even with few points.

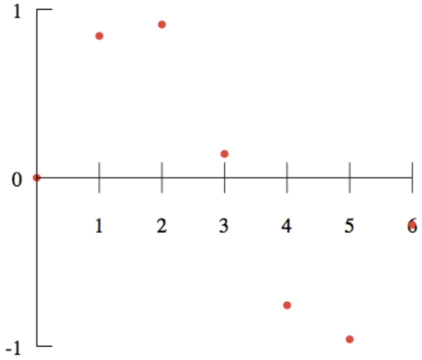
2. Interpolation

Interpolation

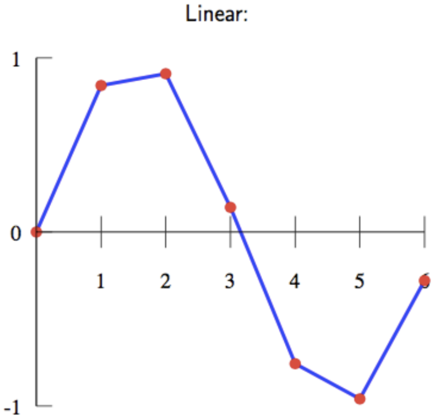
- Construct $\hat{g}$ so that $y_i = \hat{g}(x_i)$; the interpolant and the underlying function agree at finitely many points (optionally add derivative/smoothness restrictions).
- Main techniques (by popularity):
  1. **Linear interpolation;**
  2. **Spline interpolation;**
  3. **Polynomial interpolation (including Chebyshev interpolation).**
- Spline and polynomial interpolations differ (details later).

Interpolaton Examples

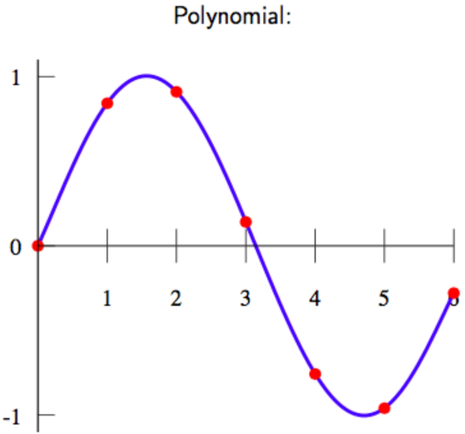
Raw data:



Interpolaton Examples



Interpolaton Examples



Piecewise Linear Interpolation

- Simplest way to interpolate: connect the points with straight lines.
- For $x \in [x_i, x_{i+1}]$, the interpolant is:

$$\hat{g}(x) \; = \; y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i). \quad \text{(piecewise linear)}$$

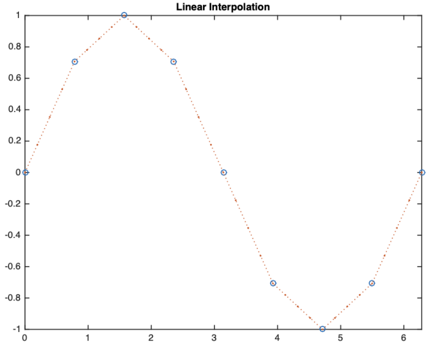
- "Kindergarten procedure of connecting the dots."
- MATLAB has the built-in function `interp1` for linear interpolation.

Linear Interpolation in MATLAB

- `vq = interp1(x, v, xq)` returns interpolated values at query points using linear interpolation. Vectors: `x` are sample points, `v` are values $v(x)$, `xq` are query points.

```
% Generates a linear approximation of a sine function
x  = 0:pi/4:2*pi;      % sample points
v  = sin(x);           % values at sample points
xq = 0:pi/16:2*pi;     % query points (can be same or finer)
vq = interp1(x, v, xq);    % Linear interpolation
plot(x, v, 'o', xq, vq, ':.')  % visualize
% xlim([0 2*pi]);      % optional
% title('Linear Interpolation'); % optional
```

Linear Interpolation Example



Linear Interpolation: Advantages and Disadvantages

Advantages (why it's popular):

- Very fast.
- Extremely simple.
- Preserves weak concavity and monotonicity.
- Easily generalizes to the multi-dimensional case.

Disadvantages: not smooth - does *not* preserve differentiability.

Spline Interpolation

- Biggest deficiency of linear interpolation: it yields a **non-differentiable** interpolant.
- Splines produce an interpolant that is *continuously differentiable* up to a chosen order by using piecewise polynomials on each interval $[x_{i-1}, x_i]$.
- Formal definition (preview): a spline of order n is piecewise polynomial of degree $n-1$ on each subinterval, with global smoothness C^{n-2}. (Next slides.)

Splines

Definition. A spline of order n is a piecewise-polynomial real function $s : [a, b] \to \mathbb{R}$ with global smoothness $s \in C^{n-2}$: there exists a grid $a = x_0 < x_1 < \cdots < x_m = b$ such that on each subinterval $[x_{j-1}, x_j]$ the function is a polynomial of degree $n - 1$.

Example. Order 2 $\Rightarrow$ standard piecewise-linear interpolation (connect the dots at the grid points). This differs from *polynomial* interpolation, which uses a single high-order polynomial on $[a, b]$.

Spline interpolation

Orders / names:

- **Linear spline** (order 2): degree 1 piece on each interval, globally C^0.
- **Quadratic spline** (order 3): parabolic pieces, globally C^1.
- **Cubic spline** (order 4): cubic pieces, globally C^2 (most popular).

MATLAB (cubic) example. `yy = spline(x,Y,xx)` performs cubic spline interpolation of Y at query points `xx` with breakpoints `x`.

```
x  = 0:pi/4:2*pi;             % breakpoints
v  = sin(x);                   % data at breakpoints
xq = 0:pi/32:2*pi;             % query points (finer grid)
vq = spline(x, v, xq);         % cubic spline interpolation
plot(x, v, 'o', xq, vq, ':.'); grid on
title('Cubic spline vs. data'); xlim([0, 2*pi])
```

Cubic Splines

- Focus on **cubic** splines. On each interval $[x_{i-1}, x_i]$:

$$s(x) = a_i + b_i x + c_i x^2 + d_i x^3.$$

- Each interval i has its own coefficients (a_i, b_i, c_i, d_i). There are $n+1$ data points, n intervals, and thus $4n$ unknown coefficients. Task: *how to find the coefficients?*

Conditions (cubic spline)

Condition 1 (interpolation): $s(x_i) = g(x_i) = y_i$, i.e.

$$y_i \;=\; a_i + b_i x_i + c_i x_i^2 + d_i x_i^3, \quad i = 1, \ldots, n. \quad \text{(on each interval)}$$

Condition 2 (The polynomial pieces must connect):

$$y_i \;=\; a_{i+1} + b_{i+1} x_i + c_{i+1} x_i^2 + d_{i+1} x_i^3, \quad i = 1, \ldots, n.$$

Conditions (cubic spline) - smoothness

Condition 3 (First and secod derivative have to agree at knots x_i):

$$b_i + 2c_i x_i + 3d_i x_i^2 \; = \; b_{i+1} + 2c_{i+1} x_i + 3d_{i+1} x_i^2, \quad i = 1, \ldots, n{-}1,$$

$$2c_i + 6d_i x_i \; = \; 2c_{i+1} + 6d_{i+1} x_i, \quad i = 1, \ldots, n{-}1.$$

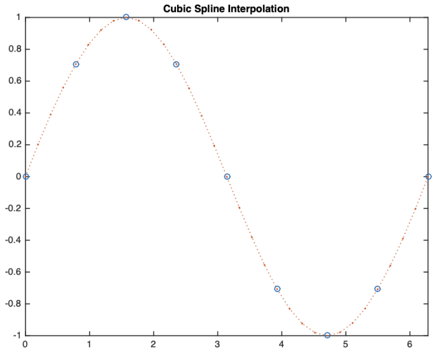
Conditions - counting equations & boundaries

- From the three blocks above we have $4n-2$ linear equations in $4n$ unknowns. We need **two more** conditions.
- A common choice (problem-dependent): $s'(x_0) = 0$ and $s'(x_n) = 0$ (a "natural" boundary condition).

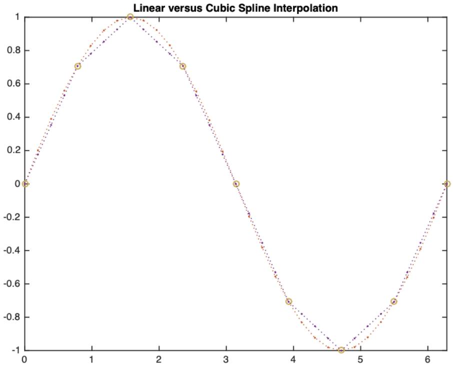
Linear vs. Cubic Spline Interpolation

```
% Compare linear (interp1) vs cubic spline (spline)
x    = 0:pi/4:2*pi;                 % breakpoints
v    = sin(x);                      % data at breakpoints
xq   = 0:pi/16:2*pi;                % query grid (finer)

v_lin = interp1(x, v, xq);    % linear interpolation
v_cub = spline(x, v, xq);     % cubic spline interpolation

plot(x, v, 'o', xq, v_lin, '--', xq, v_cub, '-.'); grid on
legend('data','linear','cubic spline','Location','best')
title('Linear vs. Cubic Spline Interpolation'); xlim([0 2*pi])
```

Cubic Spline Example



Cubic versus Linear Spline Example



Polynomial Interpolation

Suppose we want to approximate $f : [a, b] \to \mathbb{R}$ with $f \in C[a, b]$. The space $C[a, b]$ can be spanned by monomials $\{1, x, x^2, \ldots\}$, so we can write:

$$\hat{f}(x) = \sum_{i=0}^{n} \theta_i x^i \quad \text{or more generally} \quad \hat{f}(x) = \sum_{i=0}^{n} \theta_i \, \phi_i(x),$$

where $\{\phi_i\}$ is a polynomial basis.

Least Squares Method

Given nodes $\{x_i\}_{i=1}^n$ and a polynomial family $\{\phi_j\}_{j=1}^m$, choose $\theta = \{\theta_j\}_{j=1}^m$ to solve:

$$\min_\theta \sum_{i=1}^n \left(f(x_i) - \sum_{j=1}^m \theta_j \, \phi_j(x_i) \right)^2, \qquad (m < n).$$

The normal-equations solution is $\theta = (\Phi'\Phi)^{-1}\Phi' y$, where $\Phi_{ij} = \phi_j(x_i)$ and $y_i = f(x_i)$.

Which polynomials?

Monomial (Vandermonde) basis $\{1, x, \ldots, x^m\}$ is often ill-conditioned (e.g., powers become nearly collinear for large $|x|$) $\Rightarrow$ unreliable estimates.

Orthogonal polynomials avoid this issue (Legendre, Chebyshev, Laguerre, Hermite). A family $\{\phi_i\}$ is orthogonal w.r.t. weight $\omega(x)$ on $[a, b]$ if:

$$\int_a^b \omega(x)\,\phi_i(x)\,\phi_j(x)\,dx = 0 \quad \text{for } i \neq j.$$

These bases greatly improve numerical stability and approximation quality.

Chebyshev polynomials

Definition on $[-1,1]$: $T_n(x) = \cos\big(n \arccos x\big)$. They are orthogonal on $[-1,1]$ with weight $\omega(x) = \dfrac{1}{\sqrt{1-x^2}}$:

$$\int_{-1}^{1} \frac{T_i(x)\, T_j(x)}{\sqrt{1-x^2}}\, dx = \begin{cases} 0, & i \neq j, \\ \pi, & i = j = 0, \\ \frac{\pi}{2}, & i = j \geq 1. \end{cases}$$

(Equivalently, one can state the weight as $\omega(x) = (1-x^2)^{-1/2}$.)

Chebyshev polynomials: useful properties

- **Range:** $T_n(-1) = -1$, $T_n(1) = 1$, and $T_n(x) \in [-1, 1]$.
- **Extrema:** T_n has $n+1$ extrema, each equal to ± 1.
- **Roots:** T_n has n distinct roots in $[-1, 1]$ at:

$$x_i = -\cos\left(\frac{(2i-1)\pi}{2n}\right), \quad i = 1, \ldots, n.$$

- **Discrete orthogonality (at roots x_k):**

$$\sum_{k=1}^{n} T_i(x_k) T_j(x_k) = \begin{cases} 0, & i \neq j, \\ n, & i = j = 0, \\ \frac{n}{2}, & i = j \geq 1. \end{cases}$$

Chebyshev on a general interval $[a, b]$

Affine map $h : [a, b] \to [-1, 1]$ and its inverse:

$$z = h(x) = \frac{2(x-a)}{b-a} - 1, \qquad x = \frac{(z+1)(b-a)}{2} + a.$$

Define generalized Chebyshev $\tilde{T}_n(x) = T_n\big(h(x)\big)$ on $[a, b]$. They are orthogonal on $[a, b]$ with weight:

$$\omega(x) = \frac{1}{\sqrt{1 - \left(\frac{2x-(a+b)}{b-a}\right)^2}}.$$

Which coefficients?

We approximate $g : [a, b] \to \mathbb{R}$ by a degree-m Chebyshev expansion:

$$\hat{g}(x) = \sum_{j=0}^{m} \theta_j \, \tilde{T}_j(x),$$

and choose the $m+1$ coefficients $\theta = \{\theta_j\}_{j=0}^{m}$ using values of g at well-chosen nodes in $[a, b]$.

$$\#\text{nodes} = n \quad \Rightarrow \quad \begin{cases} n = m + 1 & \text{(interpolation)} \\ n > m + 1 & \text{(regression)} \end{cases}$$

Chebyshev Regression Algorithm (steps)

Task: Choose n nodes to construct a degree $m < n$ approximation of f on $[a, b]$.

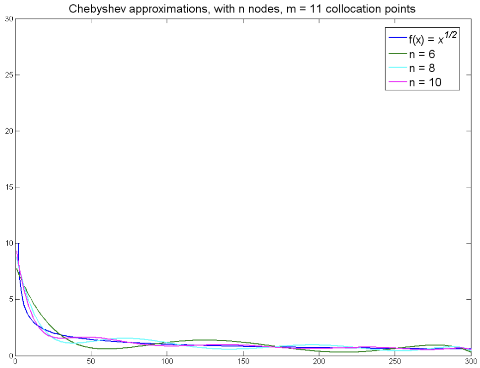
1. Compute $n \geq m+1$ Chebyshev nodes on $[-1, 1]$:
$$z_i = -\cos\left(\frac{(2i-1)\pi}{2n}\right), \quad i = 1, \ldots, n.$$
2. Map them to $[a, b]$: $x_i = \dfrac{(z_i+1)(b-a)}{2} + a$.
3. Evaluate (or obtain) $y_i = f(x_i)$.
4. Compute coefficients (one practical recipe):
$$\theta_j = \frac{\sum_{i=1}^{n} y_i \, T_j(x_i)}{\sum_{i=1}^{n} T_j(x_i)^2}, \quad j = 0, \ldots, m,$$
and form
$$\hat{f}(x) = \sum_{j=0}^{m} \theta_j \, T_j\left(\tfrac{2x-a-b}{b-a}\right).$$

Example and comparison

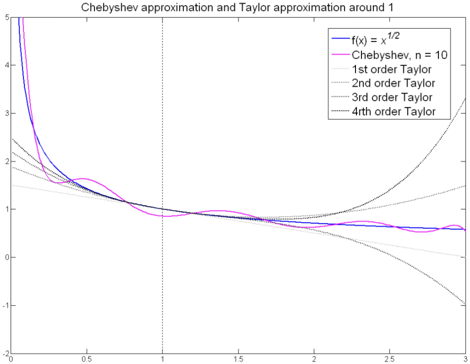
Example: Approximate $f\left(x^{-\frac{1}{2}}\right)$ on $[a, b]$ (using Chebyshev nodes and the steps above). Compute $y_i = f(x_i)$, obtain $\{\theta_j\}$, and evaluate $\hat{f}(x)$ on a fine grid for inspection.

Why Chebyshev nodes? They mitigate oscillations and improve stability/accuracy compared to evenly-spaced nodes (useful for high-degree global approximations).

Chebyshev Approximation Example



Comparison of Methods



3. Solving Neoclassical Model using fsolve

Powell's Hybrid Method (intuition)

- Newton on $f(x) = 0$ may fail to converge; when it works, it is fast.
- Minimizing $SSR(x) = \sum_i f_i(x)^2$ always moves to lower SSR, but can be slow and find only a local minimum.
- Powell's hybrid blends both ideas for robustness.

Powell's Hybrid Method (mechanics)

Given x_k, take a Newton step $s_k = -J(x_k)^{-1} f(x_k)$ and check a line search on SSR:

$$x_{k+1} = x_k + \lambda s_k, \qquad \min_{\lambda} SSR(x_k + \lambda s_k).$$

If the full Newton step ($\lambda = 1$) does not reduce SSR, shrink λ. This yields robust progress (lower SSR), though not guaranteed to find a root.

MATLAB: `fsolve`

- `x = fsolve(fun,x0)` solves `fun(x)=0` starting at `x0` using a Powell-hybrid (trust-region dogleg) strategy.

Example: two-equation system

```
% fsolve demo: solve
% 2*x1 - x2 - exp(-x1) = 0
% -x1 + 2*x2 - exp(-x2) = 0
myfun   = @(x)[2*x(1)-x(2)-exp(-x(1)); -x(1)+2*x(2)-exp(-x(2))];
x0      = [-5; -5];                             % start
options = optimset('Display','iter');     % show iterations
[x,fval,exitflag] = fsolve(myfun,x0,options)
```

growthsolve.m: setup & call to fsolve

```
% growthsolve.m  Solve basic growth model with fsolve
A=1; alpha=0.4; delta=0.06; eta=0.99;
beta=0.96;  T=100;
kss = ((A*beta*alpha)/(1-(1-delta)*beta))^(1/(1-alpha));
k0  = 0.10*kss;

% seed (linear path from k0 to kss)
x0 = zeros(T,1);
for j=1:T
  x0(j) = k0*(1-j/T) + (j/T)*kss;
end

param   = [T A alpha delta eta k0 beta kss]';
options = optimoptions('fsolve','Display','iter');
sol     = fsolve(@(z)focg(z,param), x0, options);
```

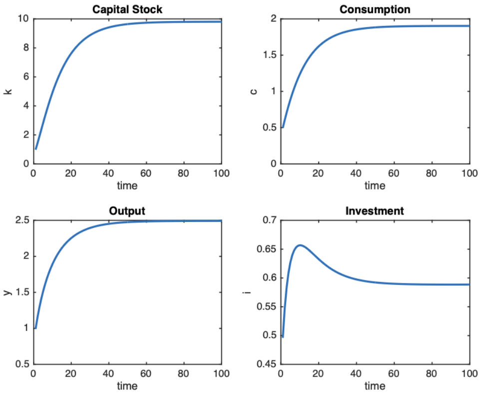
focg.m: equilibrium conditions

```
function f = focg(z,p)
T=p(1); A=p(2); alpha=p(3); delta=p(4);
eta=p(5); k0=p(6); beta=p(7); kss=p(8);

% unpack path for k and append terminal k_{T+1}=kss
k        = zeros(T+1,1);
k(1:T)   = z(:);
k(T+1)   = kss;

% Euler conditions (t=1 and t=2..T)
f     = zeros(T,1);
f(1)  = beta*(A*k(1)^alpha+(1-delta)*k(1)-k(2))^(-eta) ...
        *(alpha*A*k(1)^(alpha-1)+(1-delta)) ...
        -(A*k0^alpha+(1-delta)*k0-k(1))^(-eta);
for t=2:T
  f(t)= beta*(A*k(t)^alpha+(1-delta)*k(t)-k(t+1))^(-eta) ...
        *(alpha*A*k(t)^(alpha-1)+(1-delta)) ...
        -(A*k(t-1)^alpha+(1-delta)*k(t-1)-k(t))^(-eta);
end
```

Recovering series (k, y, i, c)

```
% After fsolve returns 'sol' for k(1..T):
k = [k0; sol; kss];
y = A*k.^alpha;
i = k(2:T+1) - (1-delta)*k(1:T);
c = y(1:T) - i;
```

Neoclassical Growth Model Example



In-class Exercise: Shock Experiments

Goal: Use the provided code to generate and interpret transition paths after shocks.

Do this (5-10 min):

- Run the baseline ($s = 0$), then a **temporary shock** with $s \neq 0$ for a few periods (e.g., `s = -0.10`, `dur = 5`, `t0 = 1`).
- Plot $\{k_t, y_t, c_t, i_t\}$ in levels and % deviations (use the plotting blocks in the code).
- What would you do if you want to introduce a **permanent shock**?

Quick questions to answer:

1. Which drops more on impact: y or i? *Why?*
2. Does c overshoot?
3. How do results change if the shock is **permanent** instead of temporary?

Files to use: `growthsolve_shock.m` + `focg_shock.m`. The plotting blocks already included.

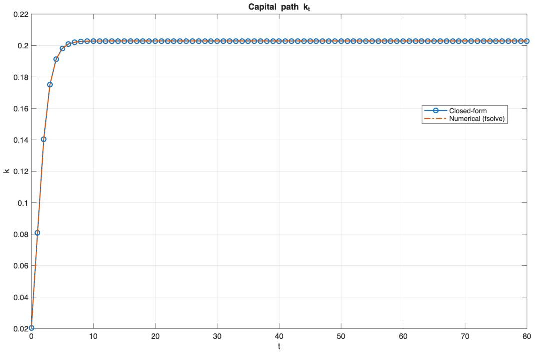
Numerical vs analytical (benchmark case)

We had seen that if $\eta \approx 1$ and $\delta = 1$, the policy reduces to:

$$k_{t+1} = \beta \alpha A \, k_t^{\alpha}.$$

Use this closed-form path to visually benchmark the `fsolve` trajectory.

Comparison Numerical vs Analytical



4. Complete Markets

The Baseline Model

The neoclassical growth model consists of the following objects:

1. **Production function:**

$$y_t = F\left(k_t, l_t\right).$$

The Baseline Model

The neoclassical growth model consists of the following objects:

1. **Production function:**

$$y_t = F\left(k_t, l_t\right).$$

2. **Consumers:**

$$c_t + i_t = y_t,$$

   where investment, i_t, obeys the law of motion for the capital stock

$$k_{t+1} = (1-\delta)k_t + i_t.$$

Arrow-Debreu Competitive Equilibrium

In a competitive equilibrium each household is faced with a **utility maximization problem**, subject to their own **preferences** and **budget constraints**.

Arrow-Debreu Competitive Equilibrium

In a competitive equilibrium each household is faced with a **utility maximization problem**, subject to their own **preferences** and **budget constraints**.

1. The Representative agent (AD):

$$\max \sum_{t=0}^{\infty} \beta^t u(c_t)$$
$$\text{s.t. } \sum_{t=0}^{\infty} p_t \left[c_t + k_{t+1} - (1-\delta)k_t\right] \leq \sum_{t=0}^{\infty} p_t \left[r_t k_t + w_t l_t\right]$$
$$0 \leq l_t \leq 1, c_t \geq 0, k_{t+1} \geq 0, k_0 \text{ known.}$$

Arrow-Debreu Competitive Equilibrium

2. **Firm:**

$$\begin{aligned} \max \textstyle\sum_{t=0}^{\infty} p_t \left[y_t - r_t k_t - w_t l_t \right] \\ \text{s.t.} \quad y_t = F\left(k_t, l_t\right) \forall t \end{aligned}$$

Arrow-Debreu Competitive Equilibrium

2. **Firm:**

$$\begin{array}{rl} \max & \sum_{t=0}^{\infty} p_t \left[y_t - r_t k_t - w_t l_t \right] \\ \text{s.t.} & y_t = F\left(k_t, l_t\right) \forall t \end{array}$$

3. **Resource feasibility constraint:**

$$y_t = c_t + k_{t+1} - (1-\delta) k_t \forall t$$

Arrow-Debreu Competitive Equilibrium

Definition

In this setting, an Arrow-Debreu Competitive Equilibrium is defined as a **household allocation** decision $Z^H = \{(c_t, k_{t+1}, l_t)\}_{t=0}^{\infty}$, a **firm allocation** decision $Z^F = \left\{\left(k_t^f, l_t^f\right)\right\}_{t=0}^{\infty}$, and a **price system** $\{(p_t, r_t, w_t)\}_{t=0}^{\infty}$, such that, given prices,

Arrow-Debreu Competitive Equilibrium

Definition

In this setting, an Arrow-Debreu Competitive Equilibrium is defined as a **household allocation** decision $Z^H = \{(c_t, k_{t+1}, l_t)\}_{t=0}^{\infty}$, a **firm allocation** decision $Z^F = \left\{\left(k_t^f, l_t^f\right)\right\}_{t=0}^{\infty}$, and a **price system** $\{(p_t, r_t, w_t)\}_{t=0}^{\infty}$, such that, given prices,

1. the representative agent maximizes utility subject to the budget constraint, resource constraints and the transversality condition
2. the firm maximizes profits subject to its resource feasibility constraint, and,
3. the aggregate resource feasiblity constraint is met (i.e. markets clear)

$$
\begin{aligned}
F(k_t, l_t) = c_t + k_{t+1} - (1-\delta)k_t & \qquad \textbf{(Goods)} \\
k_t^f = k_t & \qquad \textbf{(Capital)} \\
l_t^f = l_t. & \qquad \textbf{(Labor/Leisure)}
\end{aligned}
$$

Conditions for the Competitive Equilibrium

The representative agent chooses **consumption** and **capital** to maximize utility and the firm chooses **capital** and **employment** to maximize profits. Given a Lagrangean construction:

- $[c_t]$:
- $[k_{t+1}]$:
- $\left[k_t^f\right]$:
- $\left[l_t^f\right]$:

Conditions for the Competitive Equilibrium

Euler Equation:

$$\frac{u'(c_t)}{u'(c_{t+1})} = \beta\left[r_{t+1} + 1 - \delta\right] = \beta\left[F_k(k_{t+1}, 1) + 1 - \delta\right]$$

Feasibility Condition:

$$F(k_t, l_t) = c_t + k_{t+1} - (1 - \delta)k_t$$

How do the primitives affect behavior?

1. **Smooth consumption:** if the utility function is strictly concave the individual prefers a smooth consumption stream.
   Example:

2. **Impatience:** a low β will be associated with low c_{t+1} and high c_t.

3. **The return on savings:** since k_{t+1} is endogenous we have that $F_k(k_{t+1}, l_t)$ non-trivially depends on it, so the effect cannot be signed directly.

Sequential Market Equilibrium

- From a mechanical standpoint, **all the activity in the previous economy occurred at date 0**: the consumer faces a maximization problem with a single lifetime budget constraint.

Sequential Market Equilibrium

- From a mechanical standpoint, **all the activity in the previous economy occurred at date 0**: the consumer faces a maximization problem with a single lifetime budget constraint.
- Essentially, households take stock of future states of the world and make decisions about future consumption, labor and investment in the initial time period.

Sequential Market Equilibrium

- From a mechanical standpoint, **all the activity in the previous economy occurred at date 0**: the consumer faces a maximization problem with a single lifetime budget constraint.
- Essentially, households take stock of future states of the world and make decisions about future consumption, labor and investment in the initial time period.
- **This representation does not capture the normal way in which we imagine interaction in an economy.**

Sequential Market Equilibrium

1. **The Representative agent (SME):**

$$
\begin{gathered}
\max \sum_{t=0}^{\infty} \beta^t u\left(c_t\right) \\
\text{s.t. } c_t + k_{t+1} - (1-\delta)k_t \leq r_t k_t + w_t l_t \quad \forall t \\
0 \leq l_t \leq 1, c_t \geq 0, 0 \leq k_{t+1} \leq \overline{K}, k_0 \text{ known.}
\end{gathered}
$$

Sequential Market Equilibrium

Definition

In this setting, a Sequential Markets equilibrium is an **allocation** $\{(c_t, k_{t+1}, l_t)\}_{t=0}^{\infty}$ and a **price system** $\{(r_t, w_t)\}_{t=0}^{\infty}$, such that

Sequential Market Equilibrium

Definition

In this setting, a Sequential Markets equilibrium is an **allocation** $\{(c_t, k_{t+1}, l_t)\}_{t=0}^{\infty}$ and a **price system** $\{(r_t, w_t)\}_{t=0}^{\infty}$, such that

1. the households maximize the utility subject to the budget constraint at each period t and the constraints,
2. the firm maximizes, that is, $F_k(k_t^f, l_t^f) = r_t$ and $F_l(k_t^f, l_t^f) = w_t$,
3. the aggregate resource feasiblity constraint is met (i.e. markets clear)

$$
\begin{aligned}
F(k_t, l_t) = c_t + k_{t+1} - (1-\delta)k_t & \qquad \textbf{(Goods)} \\
k_t^f = k_t & \qquad \textbf{(Capital)} \\
l_t^f = l_t. & \qquad \textbf{(Labor/Leisure)}
\end{aligned}
$$

A Simplified Example

- To motivate the **equivalence between the two definitions of equilibrium**, consider the following example:

A Simplified Example

- To motivate the **equivalence between the two definitions of equilibrium**, consider the following example:
  - Let the **economy be deterministic** and have an **infinite horizon with a finite number of types of agents**. Let there be I types and assume that there is an **equal mass of each type**:

$$\sum_i c_t^i = \sum_i e_t^i$$

A Simplified Example

- To motivate the **equivalence between the two definitions of equilibrium**, consider the following example:

A Simplified Example

- To motivate the **equivalence between the two definitions of equilibrium**, consider the following example:
  - Thus, the allocation consists in choosing a sequence $\left\{\left\{c_t^i\right\}_t\right\}_i$ that solves

$$\max \sum_{t=0}^{\infty} u_i\left(c_t^i\right)$$
$$\text{s.t. } \sum_t p_t c_t^i \leq \sum_t p_t e_t^i$$

A Simplified Example

- Consider the **period-by-period** counterpart of the representative agent problem along with a new price q_t:

A Simplified Example

- Consider the **period-by-period** counterpart of the representative agent problem along with a new price q_t:
  - The problem now will be given by:

$$
\begin{aligned}
& \max \textstyle\sum_{t=0}^{\infty} \beta^t u_i\left(c_t^i\right) \\
\text{s.t.} \quad & c_t^i + q_t a_{t+1}^i \leq e_t^i + a_t^i
\end{aligned}
$$

A Simplified Example

- Consider the **period-by-period** counterpart of the representative agent problem along with a new price q_t:
  - The problem now will be given by:

$$\begin{aligned} & \max \textstyle\sum_{t=0}^{\infty} \beta^t u_i \left(c_t^i\right) \\ \text{s.t.} \quad & c_t^i + q_t a_{t+1}^i \leq e_t^i + a_t^i \end{aligned}$$

    We also need to impose a **Non-Ponzi** scheme such that

$$a_{t+1} \geq -\overline{A}, \quad \text{with} \quad \overline{A} \in \mathbb{R}_+$$

Solving the Simplified Example

...

Exercise

An economy consists of two infinitely lived consumers names $i = 1, 2$. There is one non-storable consumption good. Consumer i consumes c_t^i at time t. Consumer i ranks consumption streams by:

$$\sum_{t=0}^{\infty} \beta^t u\left(c_t^i\right),$$

where $\beta \in (0, 1)$ and $u(c)$ is increasing, strictly concave, and twice continuously differentiable. Consumer 1 is endowed with a stream of the consumption good $e_t^i = 1, 0, 0, 1, 0, 0, 1, \ldots$. Consumer 2 is endowed with a stream of the consumption good $e_t^i = 0, 1, 1, 0, 1, 1, 0, \ldots$ Assume that there are **complete markets with time 0 trading**.

- Define a competitive equilibrium.
- Compute a competitive equilibrium.

Exercise

Consider the following economy

$$u\left(c_0^i, c_1^i, \cdots\right) = \sum_{t=0}^{\infty} \beta^t \log c_t^i,$$

$$\text{where } \left(e_0^1, e_1^1, e_2^1, e_3^1, \cdots\right) = (6, 4, 6, 4, \cdots) \quad \text{and}$$

$$\left(e_0^2, e_1^2, e_2^2, e_3^2, \cdots\right) = (4, 6, 4, 6, \cdots)$$

Exercise

Consider the following economy

$$u\left(c_0^i, c_1^i, \cdots\right) = \sum_{t=0}^{\infty} \beta^t \log c_t^i,$$

$$\text{where } \left(e_0^1, e_1^1, e_2^1, e_3^1, \cdots\right) = (6, 4, 6, 4, \cdots) \quad \text{and}$$

$$\left(e_0^2, e_1^2, e_2^2, e_3^2, \cdots\right) = (4, 6, 4, 6, \cdots)$$

- Define a **Sequential Market Equilibrium** and find it.
- Define an **Arrow-Debreu Equilibrium** and find it.