(Mostly) Review

Lectures

1. Intro & Julia programming
2. Finite difference methods, boundary conditions, and convergence notions
3. Dynamic programming (& iterative methods)
4. Value iteration and policy iteration
5. Continuous time, policy iteration

PDE solving: vectorize

- We approximate derivatives in terms of $\hat{v}$:

$$\frac{\partial^2 v}{\partial x^2}(x_i) \approx \frac{\hat{v}_{i+1} - 2\hat{v}_i + \hat{v}_{i-1}}{h^2} \quad \text{for } 1 \leq i \leq N-1.$$

- The equation $0 = \Delta v = \frac{\partial^2 v}{\partial x^2}$ can thus be written for one x_i as

$$0 = \begin{bmatrix} \underbrace{0 \cdots 0}_{i-1} & 1/h^2 & -2/h^2 & 1/h^2 & \underbrace{0 \cdots 0}_{N-1-i} \end{bmatrix} \begin{bmatrix} \hat{v}_0 \\ \vdots \\ \hat{v}_{i-1} \\ \hat{v}_i \\ \hat{v}_{i+1} \\ \vdots \\ \hat{v}_N \end{bmatrix}$$

for $1 \leq i \leq N-1$.

PDE solving: matrix equation

- Stacking these on top of each other for all i, we get

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1/h^2 & -2/h^2 & 1/h^2 & & & \\ & 1/h^2 & -2/h^2 & 1/h^2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1/h^2 & -2/h^2 & 1/h^2 \end{bmatrix} \begin{bmatrix} \hat{v}_0 \\ \vdots \\ \hat{v}_N \end{bmatrix}$$

- We now only need to add boundary conditions ($i \in \{0, N\}$)

Dirichlet: $v(x_i) = f(x_i)$

Add a row with the i^{th} entry 1 and the rest 0.

Add $f(x_i)$ to the 'other side' of the equation.

Neumann and Robin: $av(x_i) + bv'(x_i) = f(x_i)$

Introduce a **ghost point**.

Use the boundary condition to solve for the ghost point.

Use $\frac{\partial^2 v}{\partial x^2}(x_i) = 0$ and plug in the ghost point solution to get an equation only in point of the domain.

Add the terms from the equation to 'the other side'.

PDE solving: multiple dimensions

- In multiple dimensions, the equation must still be stored as a matrix A, a vector $\hat{f}$, and a vector of unknowns $\hat{v}$.
- To do this, a mapping from the multi-dimensional indices to 1-D indices is needed.
- In 2-D, this could look like $(i, j) \mapsto (j-1)(N_x+1)+i$, but any (bijection) works.
- This gives the transformation

$$\begin{bmatrix} v(x_0, y_0) & \cdots & v(x_0, y_{N_y}) \\ v(x_1, y_0) & \cdots & v(x_1, y_{N_y}) \\ \vdots & \cdots & \vdots \\ v(x_{N_x}, y_0) & \cdots & v(x_{N_x}, y_{N_y}) \end{bmatrix}$$

$$\mapsto$$

$$\begin{bmatrix} v(x_0, y_0) & \cdots & v(x_{N_x}, y_0) & v(x_0, y_1) & \cdots & v(x_{N_x}, y_1) & \cdots & v(x_0, y_{N_y}) & \cdots & v(x_{N_x}, y_{N_y}) \end{bmatrix}^\top$$

Optimal control: discrete time, finite horizon

- **Dynamic system**:

$$X_{k+1} = f(X_k, \alpha_k, W_{k+1}), \quad X_0 = x$$

- Restriction to **admissible controls**: $\alpha_k \in \mathcal{A}_k(x_k)$.
- With $\pi = (\alpha_0, \dots, \alpha_{N-1})$ the (finite horizon) problem has a **payoff functional**

$$J_0(x_0; \pi) = \mathbb{E}\Big[\sum_{k=0}^{N-1} g_k(x_k, \alpha_k) + \varphi(x_N)\Big]$$

- We seek $V_i(x_i) = \sup_\pi J_i(x_i; \pi)$, in particular $V_0(x_0)$.
- We 'derived' the **dynamic programming principle**, or the **dynamic programming algorithm**:

$$V_i(x_i) = \sup_{\alpha_i} \mathbb{E}\Big[g_i(x_i, \alpha_i) + V_{i+1}(f(x_i, \alpha_i, W_i))\Big], \quad V_N = \varphi$$

Optimal control: discrete time, discounted

- Consider $g_k = \gamma^k g$ for some function g.
- The payoff functional is

$$J_0(x_0; \pi) = \mathbb{E}\left[\sum_{k=0}^{N-1} \gamma^k g(x_k, \alpha_k) + \gamma^N \varphi(x_N)\right]$$

- The dynamic programming principle is then

$$V_i(x_i) = \sup_{\alpha_i} \mathbb{E}\Big[g(x_i, \alpha_i) + \gamma V_{i+1}(f(x_i, \alpha_i, W_i))\Big], \quad V_N = \varphi$$

- If $\gamma < 1$, W_k are i.i.d., and $N \to \infty$, we expect...

Optimal control: discrete time, infinite horizon

- We expect to get the problem

$$V(x) = \sup_{\pi} \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k g(X_k, \alpha_k)\Big]$$

- The value is now independent of time. **Why?**
- We shall focus on $\pi = (\alpha(X_0), \alpha(X_1), \dots)$.
- The **dynamic programming principle** then becomes the **dynamic programming equation (DPE)**:

$$V(x) = \sup_{\alpha} \mathbb{E}\Big[g(x, \alpha) + \gamma V(f(x, \alpha, W))\Big]$$

- **This does not lead to a backward stepping algorithm!**
- Instead we develop iterative schemes...

Dynamic programming: vectorized

- Consider finitely many states: $\mathcal{X} = \{x^1, \ldots, x^n\}$ and $\bar{V} = (V(x^1), \ldots, V(x^n))$.
- Denote by $p(x^\ell | x^m, \alpha^m) = \mathbb{P}\Big(f(x^m, \alpha^m, W_1) = x^\ell\Big)$.
- For m, ℓ we put these values in a matrix:

$$P^{\bar{\alpha}} = \begin{bmatrix} p(x^1|x^1, \alpha^1) & \cdots & p(x^n|x^1, \alpha^1) \\ \vdots & \ddots & \vdots \\ p(x^1|x^n, \alpha^n) & \cdots & p(x^n|x^n, \alpha^n) \end{bmatrix} \in \mathbb{R}^{n \times n}$$

- With $\bar{g}(\bar{\alpha}) = (g(x^1, \alpha^1), \ldots, g(x^n, \alpha^n))$, we write the dynamic programming equation in vectorized form

$$\bar{V} = \sup_{\bar{\alpha}} \left[\bar{g}(\bar{\alpha}) + \gamma P^{\bar{\alpha}} \bar{V} \right]$$

- Or, element by element,

$$\forall i, \quad \bar{V}^i = \sup_{\bar{\alpha}} \left[\bar{g}(\bar{\alpha})^i + \gamma (P^{\bar{\alpha}} \bar{V})^i \right] = \sup_{\bar{\alpha}^i} \left[g(x^i, \bar{\alpha}^i) + \gamma \sum_j p(x^j | x^i, \bar{\alpha}^i) V^j \right]$$

Dynamic programming: fixed point structure

- Like previous weeks, we drop the 'bars'.
- Inspired by the DPP, define $T^\alpha U = g(\alpha) + \gamma P^\alpha U$, so that the DPP reads

$$V = \sup_\alpha T^\alpha V = T^{\alpha^*} V$$

  with α^* being optimal, i.e., $J^{\alpha^*} = V$.
- Similarly,

$$J^\alpha = T^\alpha J^\alpha$$

- The quantities we are interested in are fixed points to T^α!
- Moving V to the RHS, $0 = \sup_\alpha \left[T^\alpha V - V\right]$.
- Inspired by this, define $\mathcal{B}U = \sup_\alpha \left[T^\alpha U - U\right]$.
- The value function is a root of $\mathcal{B}$!

Value and policy iteration

- We iteratively both **evaluate** the policy and **update** the policy.
- **Value iteration** (with tolerance ε), starting with $m = 0$ and some V_0:
  1. $\alpha_{m+1} = \arg\max_\alpha T^\alpha V_m$
  2. $V_{m+1} = T^{\alpha_{m+1}} V_m$
  3. Return to 1 if unless $\|V_{m+1} - V_m\| = \max_{x \in \mathcal{X}} |V_{m+1}(x) - V_m(x)| < \varepsilon(1-\gamma)/2\gamma$
- Note that steps 1 and 2 above can be written as $V_{m+1} = \max_\alpha T^\alpha V_m$.
- **Policy iteration**, starting with $m = 0$ and α_0:
  1. $V_m = J^{\alpha_m} = (I - \gamma P^{\alpha_m})^{-1} g(\alpha_m)$
  2. $\alpha_{m+1} = \arg\max_\alpha T^\alpha V_m$
  3. Return to 1 unless $\alpha_{m+1} = \alpha_m$.
- Policy iteration is also known as **Howard's algorithm** after its pioneer Ronald A. Howard.

Value iteration and policy iteration: convergence

- We know that both value iteration and policy iteration converge to the value function V.
- Value iteration requires lower computational effort **per iteration**, but requires more iterations to converge.
- The policy iteration error decreases very fast **per iteration** once the moderately close to the correct solution.
- This very fast convergence is thanks to it being structurally similar to Newton's method.

Policy iteration: connection to Newton's method

Recall that $\mathcal{B}U = \max_\alpha T^\alpha U - U = \max_\alpha g(\alpha) + (\gamma P^\alpha - I)U$. And that V is its root.

Then $\mathcal{B}$ is "convex" in the following sense.

Lemma

For any U, U', and $\alpha \in \arg\max_\alpha g(\alpha) + (\gamma P^\alpha - I)U = \arg\max_\alpha T^\alpha U$,

$$\mathcal{B}U' \geq \mathcal{B}U + (\gamma P^\alpha - I)(U' - U)$$

By the following theorem, policy iteration is similar to **Newton's method**.

Theorem

Let (V_m, α_m) be generated by policy iteration. Then

$$V_{m+1} = V_m - (\gamma P^{\alpha_{m+1}} - I)^{-1}\mathcal{B}V_m.$$

Optimal control: discounted continuous time, finite horizon

- With $\pi = (\alpha_t)_{t\geq 0}$

$$\mathrm{d}X_t^\pi = \mu(t, X_t^\pi, \alpha_t)\,\mathrm{d}t + b(t, X_t^\pi, \alpha_t)\,\mathrm{d}W_t$$

- With **running reward** g and **terminal reward** φ, the **payoff functional** is

$$J(t,x;\pi) = \mathbb{E}_{t,x}\left[\int_t^T e^{-rt} g(X_t^\pi, \alpha_t)\,\mathrm{d}t + e^{-rT}\varphi(X_T^\pi)\right]$$

- The **value function** is the optimal payoff:

$$V(t,x) = \sup_\pi J(t,x;\pi)$$

- The value function solves the **HJB** PDE with $V(T,x) = \varphi(x)$:

$$\partial_t V(t,x) - rV + \sup_\alpha \left[g(x,\alpha) + \mu(t,x,\alpha)\partial_x V(t,x) + \frac{1}{2} b(t,x,\alpha)^2 \partial_{xx} V(t,x)\right] = 0$$

- How would you solve this numerically?

Optimal control: discounted continuous time, infinite horizon

- Like in discrete time, the dependence on time is expected to disappear as $T \to \infty$:

$$J(x;\pi) = \mathbb{E}\left[\int_0^\infty e^{-rt} g(X_t^\pi, \alpha_t)\,\mathrm{d}t \,\middle|\, X_t^\pi = x\right], \quad V(x) = \sup_\pi J(x;\pi)$$

- And the corresponding **HJB** equation is

$$0 = -rV(x) + \sup_\alpha \left[g(x,\alpha) + \mu(x,\alpha)\partial_x V(x) + \frac{1}{2} b(x,\alpha)^2 \partial_{xx} V(x)\right]$$

- Do the methods for solving the finite horizon problem work?

Discretization: one dimension

- Let h be the grid size and approximate the derivatives as

$$\frac{\partial V(x)}{\partial x} \approx \begin{cases} \dfrac{V(x+h)-V(x)}{h} & \text{if } \mu(x,\alpha) > 0, \\ \dfrac{V(x)-V(x-h)}{h} & \text{if } \mu(x,\alpha) < 0, \end{cases}$$

$$\frac{\partial^2 V(x)}{\partial x^2} \approx \frac{V(x+h)-2V(x)+V(x-h)}{h^2},$$

- Then, with $x^+ = \max\{0,x\}$ and $x^- = \min\{0,x\}$ (so $|x| = x^+ - x^-$),

$$\begin{aligned} 0 &= -rV(x) + \sup_\alpha \left[g(x,\alpha) + \mu(x,\alpha)\partial_x V(x) + \frac{1}{2} b(x,\alpha)^2 \partial_{xx} V(x) \right] \\ &= -rV(x) + \sup_\alpha \Bigg[g(x,\alpha) + \left(\frac{\mu(x,\alpha)^+}{h} + \frac{b^2}{2h^2} \right) V(x+h) + \left(-\frac{\mu(x,\alpha)^-}{h} + \frac{b^2}{2h^2} \right) V(x-h) \\ &\qquad - \left(\frac{|\mu(x,\alpha)|}{h} + \frac{b^2}{h^2} \right) V(x), \Bigg] \end{aligned}$$

which can be written on matrix form

$$0 = -rV + \sup_\alpha \left(g(\alpha) + A^\alpha V \right) = \sup_\alpha \left(g(\alpha) + (A^\alpha - rI)V \right)$$

Discretization: matrix form

- Indeed, indexing x^i as the ith grid point,

$$A^\alpha_{i,i+1} = \frac{\mu(x^i,\alpha)^+}{h} + \frac{b(x^i,\alpha)^2}{2h^2} \quad A^\alpha_{i,i-1} = \frac{-\mu(x^i,\alpha)^-}{h} + \frac{b(x^i,\alpha)^2}{2h^2} \quad -A^\alpha_{i,i} = \frac{|\mu(x^i,\alpha)|}{h} + \frac{b(x^i,\alpha)^2}{h^2}$$

  and $A^\alpha_{i,j} = 0$ otherwise.
- Clearly, $\sum_j A^\alpha_{i,j} = 0$ and $A^\alpha_{i,j} \geq 0$ whenever $i \neq j$.
- Moreover, $\sup_\alpha \max_i -A^\alpha_{i,i} < \infty$ is satisfied whenever $\|\mu\|_\infty, \|b\|_\infty < \infty$.

Discretization: matrix form

- Indeed, indexing x^i as the ith grid point,

$$A^\alpha_{i,i+1} = \frac{\mu(x^i,\alpha)^+}{h} + \frac{b(x^i,\alpha)^2}{2h^2} \quad A^\alpha_{i,i-1} = \frac{-\mu(x^i,\alpha)^-}{h} + \frac{b(x^i,\alpha)^2}{2h^2} \quad -A^\alpha_{i,i} = \frac{|\mu(x^i,\alpha)|}{h} + \frac{b(x^i,\alpha)^2}{h^2}$$

  and $A^\alpha_{i,j} = 0$ otherwise.
- Clearly, $\sum_j A^\alpha_{i,j} = 0$ and $A^\alpha_{i,j} \geq 0$ whenever $i \neq j$.
- Moreover, $\sup_\alpha \max_i -A^\alpha_{i,i} < \infty$ is satisfied whenever $\|\mu\|_\infty, \|b\|_\infty < \infty$.
- **The PDE boundary conditions must be encoded in A^α!**

Discretization: discrete time control reinterpretation

- Let $c = \max_i |A_{i,i}| < \infty$.
- For any α and U,

$$\bar{g}(\alpha) + (A^\alpha - rI)U = \bar{g}(\alpha) + \Big(c\underbrace{(I + A^\alpha/c)}_{\tilde{Q}^\alpha} - (r+c)I\Big)U$$

$$= (r+c)\left(\frac{\bar{g}(\alpha)}{r+c} + \Big(\frac{c}{r+c}\tilde{Q}^\alpha - I\Big)U\right)$$

- In particular, the discretized DPE can be rewritten as

$$0 = \sup_\alpha \left[\frac{\bar{g}(\alpha)}{r+c} + \frac{c}{r+c}\tilde{Q}^\alpha \bar{V} - \bar{V}\right]$$

- Or, equivalently, with $\gamma = \frac{c}{c+r}$ and $\tilde{g} = \frac{g}{c+r}$,

$$\bar{V} = \sup_\alpha \left[\tilde{g} + \gamma \tilde{Q}^\alpha \bar{V}\right]$$

- Because $\tilde{Q}^\alpha_{i,j} \in [0,1]$ for all i, j and $\sum_j \tilde{Q}^\alpha_{i,j} = 1$, it is a probability matrix.

Discretization: policy iteration

We therefore know that the following policy iteration scheme for the continuous time problem converges.

1. Start with $m = 0$ and any policy α_0.
2. Evaluate the policy by solving

$$0 = -rV_m + \bar{g}(\alpha_m) + A^{\alpha_m} V_m$$

3. Update the policy by finding

$$\alpha_{m+1} = \arg\max_{\alpha} -rV_m + \bar{g}(\alpha) + A^{\alpha} V_m$$

4. If $\alpha_{m+1} = \alpha_m$, then $V_m = V$. Otherwise, increment m and return to step 2.