Class notes, ECE 719. University of Wisconsin, Madison

Nasser M. Abbasi

December 30, 2019 Compiled on December 30, 2019 at 11:20pm [public]

Contents

List of Figures

List of Tables

Chapter 1

Class notes

Local contents

Summary table

These are my class lecture notes written from the lectures of ECE 719 optimal systems course given by Professor B. Ross Barmish at University of Wisconsin, Madison in Spring 2016. Any errors in these notes, then all blames to me and not to the instructor.

#	date	event	Topic
1	Tuesday, 1/19/2016	First class	Introduction, handouts
2	Thursday, 1/21/2016	Multilinear	Tractable, Farming example, Multilinear, det(M) example
3	Tuesday, 1/26/2016	Real analysis	Reader on Farming, level sets, Pareto, Existence of optimal, real analysis, B-W, sub-sequences
4	Thursday, 1/28/2016	Quadratic forms	Coercivity, classical existence theorem, Quadratic forms, starting convex sets
5	Tuesday, 2/2/2016	Mixtures	Polytope, Mixtures, Extreme points, Started convex functions, maximum of collection of convex functions is convex function. epi graph.
6	Thursday, 2/4/2016	Convex	Convex functions, properties, indexed collection, Hessian theorem: $J(u)$ is convex iff, the Hessian is positive semi-definite.
7	Tuesday, 2/9/2016	Hessian	Bridging lemma. Proof of Hessian theorem for $n > 1$. Strong local minimum theorem.
8	Thursday, 2/11/2016	optimal gain	optimal gain problem.
9	Tuesday, 2/16/2016	Gradient	Gradient based optimization
10	Thursday, 2/18/2016	Exam 1	
11	Tuesday, 2/23/2016	Steepest	Handout amplifier. Finish Steepest descent. Start on Newton-Raphson
12	Thursday, 2/26/2016	Convergence	Handout Newton. Derivation of step size for Newton-Raphson. Super-linear convergence.
13	Tuesday, 3/1/2016	Gradient direction	Gradient direction, quadratic convergence, mutually conjugate vectors, quadratic convergence theorem
14	Thursday, 3/3/2016	LP	Starting linear programming
15	Tuesday, 3/8/2016	LP	patrol sector problem, mechanics of LP
16	Thursday, 3/10/2016	LP	Squeeze method, basic and feasible solutions
17	Tuesday, 3/15/2016	LP simplex	optimality theorem, extreme point theorem, unit simplex, mechanism of simplex
18	Thursday, 3/17/2016	Complete LP	Tableau method with optimality
19	Tuesday, 3/22/2016		No class. Thanks giving
20	Thursday, 3/24/2016		No class. Thanks giving
21	Tuesday, 3/29/2016	LP in control	Example using LP in control, minimum fuel. Ending LP, starting dynamic programming, review.
22	Thursday, 3/31/2016	Exam 2	
23	Tuesday, 4/5/2016	Dynamic programming	First example in dynamic programming, trip from NY to San Francisco. Toll fee optimization.
24	Thursday, 4/7/2016	No class	
25	Tuesday, 4/12/2016	D.P. and special problem	describe special problem. Dynamic programming.
26	Thursday, 4/14/2016	D.P.	LQR example. Oil and real estate example
27	Tuesday, 4/19/2016	D.P. and LQR	LQR using D.P., long example

28	Thursday, 4/21/2016	D.P. floor function	D.P. examples for variation of dynamic programming
29	Tuesday, 4/26/2016	Steady state	Finish Floor problem. Start on steady state, iterative method
30	Thursday, 4/28/2016	Steady state, Riccati	Closed form, guess method, infinite time LQR, Riccati.
31	Tuesday, 5/3/2016	Review of course	Special problem review, class review and prep for final exam
32	Thursday, 5/5/2016	Final exam	Exam

1.1 Lecture 1. Tuesday, January 19, 2016

This course is on finite dimensional optimization, which means having finite number of parameters. We now went over the syllabus. Here is a summary:

1. Convex sets and functions
2. How to certify your solution?
3. Linear programming.
4. Dynamic programming (at end of course)
5. Three exams and a final special project/problem.

Homeworks will be graded using E,S,U grades. Most will get S, few will get E. Matlab will be used.

Cardinal rule Develop an answer using given material in class only. Can use other basic things like Laplace transform, etc...

A wise person once said: “Fundamental difficulties are invariant under reformulation”.

1.1.1 Objective functions, constraints and variables

In a problem, identify the objective function, constraints and variables. Optimization problems from different fields can be formulated into a common framework for solving using optimization methods.

Ingredients in this case: Set $U \subseteq \Re^n$, the constraint set. Problem has n parameters (the decision variables). Therefore $u \in U$. One dimensional problem has $n = 1$. An example is to find optimal resistor value where $U = 100 \cdots 200$ Ohms. For $n = 2$, an example is to find two resistors $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ with $100 \le u_1 \le 200$ and $300 \le u_2 \le 400$.

Reader Often we describe U graphically in $\Re^n$. Typically for only $n = 1, 2, 3$. Do this for the above example.

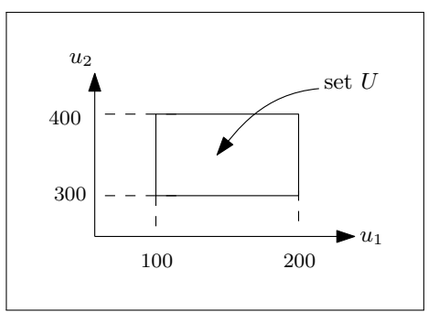


Figure 1.1: Set U with constraints

Reader design a bandpass filter with passband from ω_1 to ω_2 with $\omega_1, \omega_2 > 0$. Describe graphically the set U.

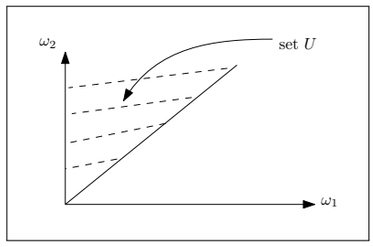


Figure 1.2: Lecture one, set U second diagram

Reader Often U is sphere in $\Re^n$ described by $\sum_{i=0}^{n} \left(u_i - u_{i,0}\right)^2 \leq R$ where $u_{i,0}$ are coordinates of center of sphere.

Reader Suppose we are designing an input voltage $u(t)$ on $t \in [0,1]$ such that $\int_0^1 u^2(t)\,dt \leq 5$. This does not fit in above framework. This is function space problem.

An important class of U: A generalization of rectangle in 2D to hypercube in $\Re^n$ with $\left|u_i - u_{i,0}\right| \leq r_i$ for $i = 1 \cdots n$.

Reader For $n = 3$ and $u_0 = \begin{bmatrix} u_{1,0} \\ u_{2,0} \\ u_{3,0} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $r_1 = 1, r_2 = 2, r_3 = 3$, then sketch U.

1.1.2 Constrained and Unconstrained problems

Unconstrained problem is one when we say $U = \Re^n$. There are no constraints on U. These are easier to solve than constrained problems. In practice, there will always be constraints. U is sometimes called the set of decision variables, or also called the input. The other criteria, is the objective function $J(u)$, or more formally $J : \Re^n \to \Re$. The objective function $J(u)$ is obtained from your goals.

Example we want to go from A to B in least time. Often the selection of $J(u)$ is not straight forward. Selection of $J(u)$ can be difficult in areas such as social or environmental science.

In stock markets, $J(u)$ is given and we just use it. Example is $J(u) = u^2 + 6u + e^{-u}$ in $\Re^n$. Often we test the algorithm first in $\Re^1$ or $\Re^2$ before going to higher dimensions. In $\Re^4$ an example is

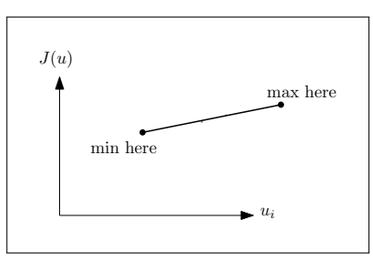
$$J(u) = u_1u_2 - 3u_1u_3u_4 + 6u_2u_3 - 6u_1 - 4u_2 - u_3u_4 + 12$$

This has 16 vertices.

Reader Can you find max $J(u)$ subject to U described by $|u_i| \leq i$ using common sense?

Answer It will be on the vertices of hypercube.

In VLSI, with hundreds of resistors, there are 2^n vertices, so the problem becomes computationally harder to solve very quickly. To proof that the optimal value is at a vertex, the idea is to freeze all other variables except for one at a time. This gives a straight line in the free variable. Hence the optimal is at ends of the line.



The main case problem: Find u^* that minimizes $J(u)$ with the constraints $u \in U$. u^* might not exist. When it exists, then

$$J(u^*) = J^* = \min_{u \in U} J(u)$$

When we begin, we do not know if u^* exist. We write

$$J(u^*) = \inf_{u \in U} J(u)$$

Example $U = \Re^n$, $J(u) = e^{-u}$. So u^* do not exist. We do not allow u^* to take values $\pm\infty$. But we allow $J(u^*)$ itself to become $\pm\infty$. For example, if $J(u) = u^2$ then $u^* = 0$.

1.2 Lecture 2. Thursday, January 21, 2016

1.2.1 Existence of optimal solution, explicit and implicit $J(u)$

Tractability. $U \subseteq \Re^n$, the decision variables. $J : \Re^n \to \Re$. Problem: find u^* such that $J(u^*) = \inf_{u \in U} J(u)$.

u^*, when it exists, is called the optimal element. We do not allow $u^* = \pm\infty$, but allow $J(u^*) = \pm\infty$. For example. $J(u) = \frac{1}{u}$ on $U = (0, \infty)$. We say $\sup_{u \in U} J(u) = J^* = 0$ but $u^* = \infty$ do not exist. Hence J^* is a limiting supremum value. Another example is $J(u) = -u$ on $\Re$. Therefore $J^* = -\infty$ but u^* do not exist.

Reader: We can consider $\max_{u \in U} J(u) \equiv \sup_{u \in U} J(u)$. Note that

$$\max_{u \in U} J(u) = -\min_{u \in U} J(u)$$

But $u^*_{\min} \neq u^*_{\max}$.

Some problems will be tractable and some are not. Some problems will not have defined algorithms and some might not have certified algorithms. Some problems have algorithms but are not tractable. NP hard problems can be either tractable or not tractable. What about stochastic problems? If u is random variable, we can not write $J(u)$. But instead we work with $\tilde{J}(u) = E(J(u))$ where E is expectation. So now $\tilde{J}(u)$ fits in the frameworks we used earlier.

We also need to make distinction between explicit and implicit $J(u)$. When we write $J(u) = u_1^2 + e^{u_2} \cos u_1 + u_2$, then $J(u)$ here is explicit. But if we have a circuit as below, where $J(u) = V_{out}$, then here $J(u)$ is implicit, since we have to solve the complicated RLC circuit to find $J(u)$, so we implicitly assume $J(u)$ exists.

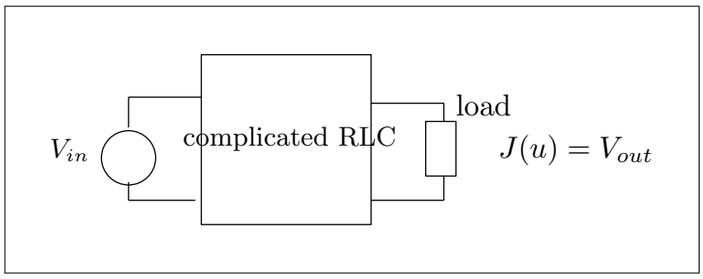


Figure 1.3: Complicated RLC

1.2.2 Farming problem

First detailed example Optimal farming example. Let $y(k)$ be the annual crop value where k is the year. This is the end of year value of the crop. At end of year, we use some of this to invest and the rest we keep as profit. Let $u(k) \in [0,1]$ be the fraction of $y(k)$ invested back. Therefore $(1 - u(k))$ is the fraction of $y(k)$ which is taken out as profit. Dynamics of the problem are: $y(k+1) = y(k)$ if we invest nothing (i.e. $u = 0$). But if we invest, then

$$y(k+1) = y(k) + \omega(k) \overbrace{u(k) y(k)}^{\text{amount invested}}$$

Where $\omega(k)$ is an independent and identically distributed (i.i.d.) random variable which depends on the weather and other variables. Let $E(\omega(k)) = \bar{\omega}$. We have N years planning horizon. What about $J(u)$? If we model things as a convex problem, we can solve it, but if we do not, it becomes hard to solve.

$$J(u) = E\left(y(k) + \sum_{k=0}^{N-1} y(k)(1 - u(k))\right) \tag{1}$$

The set U here is $\{u(0), u(1), \cdots, u(N-1)\}$. In this example $J(u)$ is implicit. We need to make $J(u)$ explicit. Let us now calculate $J(u)$ for $N=2$

$$\begin{aligned} y_1 &= y_0 + \omega_0 u_0 y_0 \\ &= y_0 (1 + \omega_0 u_0) \end{aligned}$$

In class, we assumed that $y_0 = 1$. But will keep it here as Y, which is the initial conditions.

$$y_1 = Y(1 + \omega_0 u_0) \tag{2}$$

Now for the second year, we have

$$\begin{aligned} y_2 &= y_1 + \omega_1 u_1 y_1 \\ &= y_1 (1 + \omega_1 u_1) \end{aligned}$$

Substituting (2) into the above gives

$$\begin{aligned} y_2 &= Y(1 + \omega_0 u_0)(1 + \omega_1 u_1) \\ &= Y + Y\omega_0 u_0 + Y\omega_1 u_1 + Y\omega_0 \omega_1 u_0 u_1 \end{aligned}$$

Therefore from (1) we have for $N=2$

$$\begin{aligned} J(u) &= E\left(y_2 + \sum_{k=0}^{1} y_k (1 - u_k) \right) \\ &= E\left(\overbrace{Y + Y\omega_0 u_0 + Y\omega_1 u_1 + Y\omega_0 \omega_1 u_0 u_1}^{y} + \left[Y(1-u_0) + y_1 (1-u_1) \right] \right) \\ &= E\left(Y + Y\omega_0 u_0 + Y\omega_1 u_1 + Y\omega_0 \omega_1 u_0 u_1 + \left[Y(1-u_0) + Y(1+\omega_0 u_0)(1-u_1) \right] \right) \\ &= E\left(Y + Y\omega_0 u_0 + Y\omega_1 u_1 + Y\omega_0 \omega_1 u_0 u_1 + Y - Yu_0 + Y - Yu_1 + Y\omega_0 u_0 - Y\omega_0 u_0 u_1 \right) \\ &= Y\left(3 + \varpi u_0 + \varpi u_1 + \varpi^2 u_0 u_1 - u_0 - u_1 + \varpi u_0 - \varpi u_0 u_1 \right) \\ &= Y\left(3 + 2\varpi u_0 + u_1(\varpi - 1) + \varpi^2 u_0 u_1 - u_0 - \varpi u_0 u_1 \right) \\ &= Y\left(3 + u_0(2\varpi - 1) + u_1(\varpi - 1) + u_0 u_1 \left(\varpi^2 - \varpi \right) \right) \end{aligned}$$

Reader Use syms to find $J(u)$ for $N=4$.

If we want maximum profit, then common sense says that ϖ should be large (good weather). Then $u^* = (1,1)$. This says $u_0 = 1$ and $u_1 = 1$. In other words, we invest everything back into the crop each year. The above was obtained by maximizing term by term since all terms are positive. If $\varpi < \frac{1}{2}$ (bad weather), then $u^* = (0,0)$ and all coefficients are negative.

In the above, $J(u)$ is multilinear function in u_0, u_1. If all u_k are held constant except for one at a time, then $J(u)$ becomes linear in the free parameter. In HW 1, we need to proof that extreme point of a multilinear function is at a vertex. So for this problem, the possible solutions are $(0,0), (0,1), (1,0), (1,1)$.

Reader Is there a value of ϖ that gives $(1,0)$ and $(0,1)$?

For arbitrary N there are 2^N vertices in hypercube for a multilinear $J(u)$ where $u \in \Re^n$.So for large n, a multilinear problem is much harder to solve than a convex optimization problem. Even simple problems demonstrate intractability. Let $M = N \times N$ matrix. Let each every m_{ij} be known within bounds $m_{ij}^- \le m_{ij} \le m_{ij}^+$. The question is: What are the bounds on the determinant of matrix M?

Reader ponder this question in the context of multilinear problem. To determine this for small n in Matlab, here is some code for $n=4$ (uses `allcomb` which is a file exchange file)

```
a = repmat({[-1 1]},16,1);
v = allcomb(a{:});
r = arrayfun(@(i) det(reshape(v(i,:),4,4)),1:size(v,1));
max(r)
min(r)
```

.

Running the above gives

```
ans =
    16
```

```
ans =
   -16
```

1.3 Lecture 3. Tuesday, January 26, 2016

1.3.1 Multilinear functions, level sets, contours

Today will be on multilinear functions, then back to infimum of $J(u)$. Then we will go over real analysis to see when a min is reached. We need to find first if there exists a minimum before starting to search for it. For the farming problem we looked at, say $\varpi = 2$. We want to look at contours in $\Re^2$ and $\Re^3$. Try first in $\Re^2$.

Reader Obtain level sets, contour lines, defined as $\Lambda_\gamma = \{u : J(u) = \gamma\}$. In words, contour line is curve of equal values of $J(u)$. For farming problem, show contour lines of $J(u)$ looks like

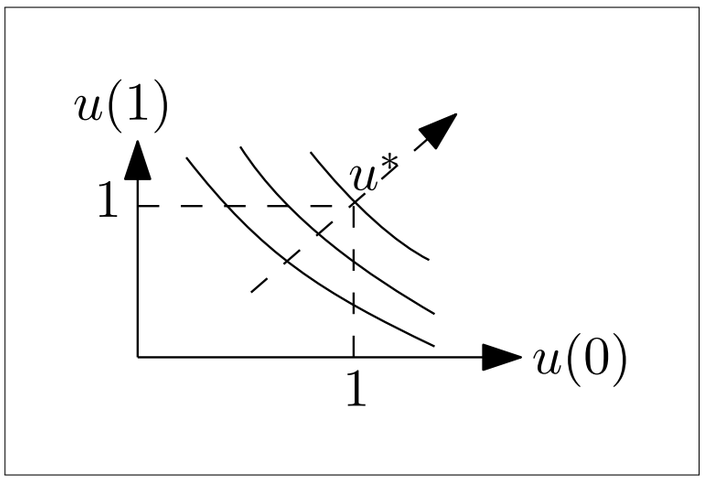


Figure 1.4: Level sets

For $\varpi = 2$, and using results from last lecture, where we had

$$J(u) = Y\left(3 + u_0(2\varpi - 1) + u_1(\varpi - 1) + u_0 u_1\left(\varpi^2 - \varpi\right)\right)$$

Where $Y = y(0) = 1$, the above becomes

$$J(u) = 3 + u_0(2\varpi - 1) + u_1(\varpi - 1) + u_0 u_1\left(\varpi^2 - \varpi\right) \tag{1}$$

Reader Use Matlab syms to obtain $J(u)$ for any N

Answer Below is a function which generates $J(u)$ for any N. Function called `nma_farm(N,y0)`, takes in N, which is how many years and $y(0)$, the initial value. $\omega 0$ below is ϖ, the mean of the ω random variable).

```
function nma_farm(N,initial)
%reader anwer for farming problem, lecture 1/21/16, ECE 719
%N is number of years
%Initial, is y(0). See class notes for ECE 719 for
%description of the problem. Second lecture.
%Nasser M. Abbasi

y = []; w=[]; u=[];
idx = [];
syms y(idx) w(idx) u(idx)
tot = 0;

for k = 0:N-1  %main loop
    tot = tot + Y(k)*(1-u(k));
end
J = Y(N)+ tot;

for i = 0:N  %use mean
```

```
    J = subs(J,w(i),'w0');
end

J=subs(J,y(0),initial);
expand(J)

%-------------------------
%   recursive function internal function
    function r = Y(k)
        if k ==0
            r = y(0);
        else
            r = Y(k-1)*(1+w(k-1)*u(k-1));
        end
    end

end
```

Here are example run outputs

```
>> nma_farm(2,1)
2*w0*u(0) - u(1) - u(0) + w0*u(1) + w0^2*u(0)*u(1) - w0*u(0)*u(1) + 3

>> nma_farm(4,1)
4*w0*u(0) - u(1) - u(2) - u(3) - u(0) + 3*w0*u(1) + 2*w0*u(2)
+ w0*u(3) + 3*w0^2*u(0)*u(1) + 2*w0^2*u(0)*u(2) + w0^2*u(0)*u(3)
+ 2*w0^2*u(1)*u(2) + w0^2*u(1)*u(3) + w0^2*u(2)*u(3) -
w0*u(0)*u(1) - w0*u(0)*u(2) - w0*u(0)*u(3) - w0*u(1)*u(2) -
w0*u(1)*u(3) - w0*u(2)*u(3) - w0^2*u(0)*u(1)*u(2) -
w0^2*u(0)*u(1)*u(3) + 2*w0^3*u(0)*u(1)*u(2) - w0^2*u(0)*u(2)*u(3)
+ w0^3*u(0)*u(1)*u(3) - w0^2*u(1)*u(2)*u(3) + w0^3*u(0)*u(2)*u(3)
+ w0^3*u(1)*u(2)*u(3) - w0^3*u(0)*u(1)*u(2)*u(3) +
w0^4*u(0)*u(1)*u(2)*u(3) + 5
>> nma_farm(5,1)

5*w0*u(0) - u(1) - u(2) - u(3) - u(4) - u(0) + 4*w0*u(1)
+ 3*w0*u(2) + 2*w0*u(3) + w0*u(4) + 4*w0^2*u(0)*u(1) +
3*w0^2*u(0)*u(2) + 2*w0^2*u(0)*u(3) + 3*w0^2*u(1)*u(2) +
w0^2*u(0)*u(4) + 2*w0^2*u(1)*u(3) + w0^2*u(1)*u(4) +
2*w0^2*u(2)*u(3) + w0^2*u(2)*u(4) + w0^2*u(3)*u(4) - w0*u(0)*u(1)
- w0*u(0)*u(2) - w0*u(0)*u(3) - w0*u(1)*u(2) - w0*u(0)*u(4)
- w0*u(1)*u(3) - w0*u(1)*u(4) - w0*u(2)*u(3) - w0*u(2)*u(4)
- w0*u(3)*u(4) - w0^2*u(0)*u(1)*u(2) - w0^2*u(0)*u(1)*u(3) +
3*w0^3*u(0)*u(1)*u(2) - w0^2*u(0)*u(1)*u(4) - w0^2*u(0)*u(2)*u(3)
+ 2*w0^3*u(0)*u(1)*u(3) - w0^2*u(0)*u(2)*u(4) - w0^2*u(1)*u(2)*u(3)
+ w0^3*u(0)*u(1)*u(4) + 2*w0^3*u(0)*u(2)*u(3) - w0^2*u(0)*u(3)*u(4)
- w0^2*u(1)*u(2)*u(4) + w0^3*u(0)*u(2)*u(4) + 2*w0^3*u(1)*u(2)*u(3)
- w0^2*u(1)*u(3)*u(4) + w0^3*u(0)*u(3)*u(4) + w0^3*u(1)*u(2)*u(4)
- w0^2*u(2)*u(3)*u(4) + w0^3*u(1)*u(3)*u(4) + w0^3*u(2)*u(3)*u(4)
- w0^3*u(0)*u(1)*u(2)*u(3) - w0^3*u(0)*u(1)*u(2)*u(4)
+ 2*w0^4*u(0)*u(1)*u(2)*u(3) - w0^3*u(0)*u(1)*u(3)*u(4)
+ w0^4*u(0)*u(1)*u(2)*u(4) - w0^3*u(0)*u(2)*u(3)*u(4)
+ w0^4*u(0)*u(1)*u(3)*u(4) - w0^3*u(1)*u(2)*u(3)*u(4) +
w0^4*u(0)*u(2)*u(3)*u(4) + w0^4*u(1)*u(2)*u(3)*u(4)
- w0^4*u(0)*u(1)*u(2)*u(3)*u(4) + w0^5*u(0)*u(1)*u(2)*u(3)*u(4) + 6
```

At $u^* = (u_0, u_1) = (1,1)$ then (1) becomes

$$\begin{aligned} J^* &= 3 + (4-1) + (2-1) + (4-2) \\ &= 9 \end{aligned}$$

Reader Look at $J(u)$ in $\Re^3$ for the farming problem. Today's topic: When does the optimal

element u^* exist?

1.3.2 Pareto optimality

When we have more than one objective function, $J_i(u) : U \subseteq \Re^n \rightarrow \Re$, for $i = 1 \cdots m$. We call $u^* \in U$ a Pareto optimal, if the following holds: Given any other $u \in U$, we can not have the following two relations be true at the same time:

1. $J_i(u) \leq J_i(u^*), i = 1 \cdots m$
2. $J_k(u) < J_k(u^*)$ for some k.

This is related to efficiency in economics. So something that is not Pareto optimal, can be eliminated.

Reader Say $U = (0, \infty)$, $J_1(u) = u + 1, J_2(u) = \frac{1}{u+1}$, describe the set of Pareto optimal solutions.

Reader By creating $J(u) = \alpha_1 J_1(u) + \alpha_2 J_2(u)$, with $\alpha_i \geq 0$, show the solution (optimal $J(u)$), depends on α_i which reflects different utility functions.

1.3.3 compact and bounded sets, open and closed sets

Now we will talk about existence of optimal element u^*. We will always assume that $J(u)$ is continuous function in $u \in U$. Two different conditions on the set U can be made

1. Is the set compact?
2. Is the set Unbounded?

In both cases, we still want conditions on $J(u)$ itself as well. We begin with the definition of

$$\boxed{J^* = \inf_{u \in U} J(u)}$$

Remember that we do not allow $u = \pm\infty$, but J^* can be $\pm\infty$.

Example $J(u) = -u$ on $\Re$. Then $J^* = -\infty$, but there is no optimal element $u^* = -\infty$. But it is always true that there is a sequence $\left\{u^k\right\}_{k=1}^{\infty} = u^k$ such that $J\left(u^k\right) \rightarrow J^*$ even though there is no u^* element. We write

$$\lim_{k \rightarrow \infty} J\left(u^k\right) = J^*$$

A bad set U is often one which is not closed. Example is of a gas pedal which when pressed to the floor will cause the car to malfunction, but the goal is to obtain the shortest travel time, which will require one to press the gas pedal to the floor in order to obtain the highest car speed.

This shows that there is no optimal u^* but we can get as close to it as we want. This is an example of a set U that is not closed on one end. We always prefer to work with closed sets. An open set is one such as $U = (0, \frac{\pi}{4}]$, and a closed set is one such as $U = \left[0, \frac{\pi}{4}\right]$.

Definition of continuity If $u^k \rightarrow u_0$ then $J\left(u^k\right) \rightarrow J(u_0)$, we write $J\left(u^0\right) = \lim_{k \rightarrow \infty} J\left(u^k\right)$. This is for every sequence u^k.

Equivalent definition: There is continuity at u^0 if the following holds: Given any $\varepsilon \geq 0$ there exists δ such that $\left|J(u) - J\left(u^0\right)\right| < \varepsilon$, whenever $\left|u - u^0\right| < \delta$.

Closed sets Includes all its limit points. Examples:

1. [0,1] is closed set.
2. [0,1) not closed. Because we can approach 1 as close as we want, but never reach it.
3. $[0, \infty)$ is closed. Since it includes all its limit points. Remember we are not allowed to use $u^* = \infty$.
4. $\Re$ is closed and open at same time.

A set can be both open and closed. $\Re$ is such a set. To show a set is closed, we need to show that the limit of any sequence is also in the set.

The intersection of closed sets remains closed, but the union can be an open set. This is important in optimization. If U_i represent constraint sets, then the intersection of the constraints remain a closed set. Closed sets also contain its boundaries.

Now we will talk about boundness of set. A set is bounded if we can put it inside a sphere of some radius. We always use the Euclidean norm. If a set is bounded, using one norm, then it is bounded in all other definitions of norms. There is more than one definition of a norm. But we will use Euclidean norm.

We like to work with compact sets. A Compact set is one which is both bounded and closed. These are the best for optimization.

1.3.4 B-W (The Bolzano-Weierstrass) theorem

Each bounded sequence in $\Re^n$ has a convergent sub-sequence. This is useful, since it says if the sequence is bounded, then we can always find at least one sub-sequence in it, which is convergent. For example $u^k = \cos k$. This does not converge, but is has a subsequence in it which does converge. The same for $u^k = (-1)^k$.

1.4 Lecture 4. Thursday, January 28, 2016

1.4.1 Existence of optimal solutions

We will spend few minutes to review existence of optimal solutions then we will talk about computability and convexity. We know now what $J(u)$ being continuous means. Closed sets are very important for well posed optimization problems. Typical closed set is $u \le k$, where k is constant. If the function $J(u)$ where $u \in U$, is a continuous function, then the set U must be closed.

Reader Give a proof.

We talked about U being closed and bounded (i.e. compact). Compact sets are best for optimization. We talked about sequence $\left\{u^k\right\}_{k=1}^{\infty} = u^k$ and a subsequence in this sequence $\left\{u^{k_i}\right\}_{i=1}^{\infty} = u^{k_i}$. If u^k converges to u^* then we say $\lim_{k\to\infty} \left\|u^k - u^*\right\| = 0$. Finally, we talked about Bolzano-Weierstrass theorem.

1.4.2 Classical existence theorem

Suppose $J : U \to \Re$ is continuous and assume U is compact (i.e. bounded and closed) and non-empty, then there exists an optimal element $u^* \in U$ such that $J(u^*) = J^* = \min_{u\in U} J(u)$. This does not say that u^* is unique. Just that it exists.

Proof Let $u^k \in U$ be a sequence such that $J\left(u^k\right) \to J^*$, then by Bolzano-Weierstrass u^{k_i} be a convergent subsequence with limit $u^* \in U$.

Reader Show that $J\left(u^{k_i}\right)$ also converges to J^*. Note: $J\left(u^{k_i}\right)$ is sequence of real numbers, which converges to a real number J^*.

Example $u^k = (-1)^k$. Let $J(u) = e^{-u^2}$ then $J(u)$ converges, but u^k does not. Hence we need to look for subsequence u^{k_i} in u^k. Now, by continuity, $J(u^*) = \lim_{i\to\infty} J\left(u^{k_i}\right) = J^*$.

There are many problems where the set is open, as in unconstrained problems. These are called open cone problems.

1.4.3 Coercive functions and Coercivity theorem

Coercive function Suppose $U \subset \Re^n$ and $J : U \to \Re$. We say that J is positive coercive if

$$\boxed{\lim_{\|u\|\to\infty} J(u) = \infty} \tag{*}$$

Initially, think of U as the whole $\Re^n$ space. So $J(u)$ blows up at ∞. Note: there is a type of uniform continuity implied by Eq (*). What Eq (*) means is that given any $\gamma \ggg 0$, arbitrarily large, there exists radius R such that $J(u) > \gamma$, whenever $\|u\| > R$. This basically says that for a Coercive function we can always find a sphere, where all values of this function are larger than some value for any $\|u\| > R$. This is useful, if we are searching for a minimum, in that we can obtain a cut off on the values in U to search for.

Example $J(u) = u^2$ is positive coercive, but $J(u) = e^u$ is not (since we can't find a sphere to limit values within it to some number), since as $u \to -\infty$, the function e^u does not blow up.

$$J(u) = au^2 + bu + c$$

Is coercive only when $a > 0$. The most famous coercive function is the positive definite quadratic form. Let A be $n \times n$ symmetric and positive definite matrix and let $b \in \Re^n$. Let c be any real number, then $J(u) = u^T Au + b^T u + c$ is coercive. This is essential to quadratic programming.

Why is $u^T Au + b^T u + c$ coercive? From matrix algebra, if A is $n \times n$ symmetric and $x \in \Re^n$, then $\lambda_{\min} \|x\|^2 \le x^T Ax \le \lambda_{\max} \|x\|^2$. For $x, y \in \Re^n$, by Schwarz inequality, $\left|\langle x, y\rangle\right|^2 \le \langle x, x\rangle \langle y, y\rangle$ or $x^T y \le \|x\| \|y\|$, to establish $J(u) = u^T Au + b^T u + c$ notice then $J(u) \ge \lambda_{\min}(A) \|u\|^2 - \|b\| \|u\| + c$. Since A is positive definite, then $\lambda_{\min}(A) > 0$ (smallest eigenvalue must be positive). So this is the same as scalar problem $au^2 + by + c$. Hence $J(u)$ is coercive function in u.

Reader What if we have a physical problem, leading to $u^T Au + b^T u + c$, but A is not symmetric, what to do? Solution: We can always work with the symmetrical part of A using $u^T Au = \frac{1}{2} u^T \left(A + A^T\right) u$, hence we work with

$$J(u) = \frac{1}{2} u^T \left(A + A^T\right) u + b^T u + c$$

Instead.

1.4.4 Convex sets and Coercivity theorem

Coercivity theorem Suppose $J : U \to \Re$ is a continuous function and coercive. And $U \subseteq \Re^n$ is closed. Then an optimal element $u^* \in U$ exist minimizing $J(u)$. **Reader** Consider the maximization problem instead of minimization.

Now we start on a new topic: Convexity. Toward finding optimal. A set $U \subseteq \Re^n$ is said to be convex set if the following holds: For any $u^0, u^1 \in U$, and $\lambda \in [0,1]$ then it follows that $\lambda u^0 + (1 - \lambda) u^1 \in U$. In words, this says that all points on the straight line between any points in the set, are also in the set. Examples:

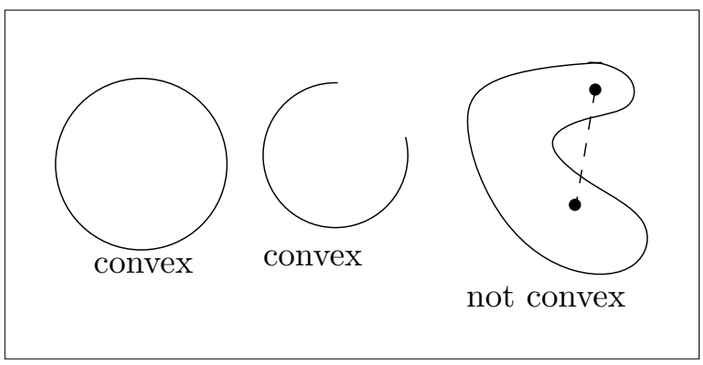


Figure 1.5: Convex sets

Reader If U_1, U_2 are both convex then show the intersection is also convex.

What about countable intersections $\cap_{i=1}^{\infty} U_i$?

Answer Yes. Example $U_i = \left\{u \in \Re^n : \|u\| \le e^{-i}\right\}$. The union is not a convex set.

Constraints using OR are union. So harder to work with OR constraints, since union of convex sets is not convex.

Reader Describe all possible convex sets on $\Re^1$.

Reader Suppose U_i is defined by set of inequalities $a_i^T u \leq b_i$, $i = 1 \cdots m$, these are intersections of sets. This is used in Linear programming.

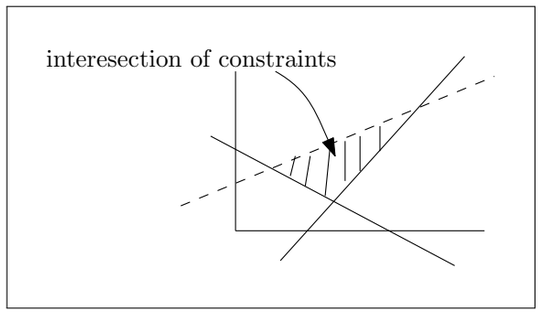


Figure 1.6: Intersection of constraints

1.5 Lecture 5. Tuesday, February 2, 2016

Back to the most important topic in optimization, which is convexity. We want to build on convex sets.

Definition A set $U \subseteq \Re^n$ is convex if the following holds: Given any $u^0, u^1 \in U$ and $\lambda \in [0,1]$, then $u^\lambda = (1-\lambda) u^0 + \lambda u^1$ is also in U. We will discuss convex function also soon.

Reader Show that the set $\{u \in \Re^n, \|u\| \leq r\}$ is convex.

Answer: Let $\|u_1\| \leq r, \|u_2\| \leq r$. Then $\|(1-\lambda) u_1 + \lambda u_2\| \leq \|(1-\lambda) u_1\| + \|\lambda u_2\|$ by triangle inequality. Hence

$$\|(1-\lambda) u_1 + \lambda u_2\| \leq (1-\lambda) \|u_1\| + \lambda \|u_2\|$$

If some center point, say $\hat{u}$ is given, then $\{u : \|u - \hat{u}\| \leq r\}$ is convex set. The translation $u + \hat{u}$ is also convex set. Other norms on $\Re^n$ are important. $\|u\|_1 = \sum_{i=1}^{n} |u_i|$ and $\|u\|_\infty = \max \{|u_1|, |u_2|, \cdots, |u_n|\}$.

Reader In $\Re^2$, sketch unit ball $\{u : \|u\| \leq r\}$ using norms $\|u\|_1$ and $\|u\|_\infty$

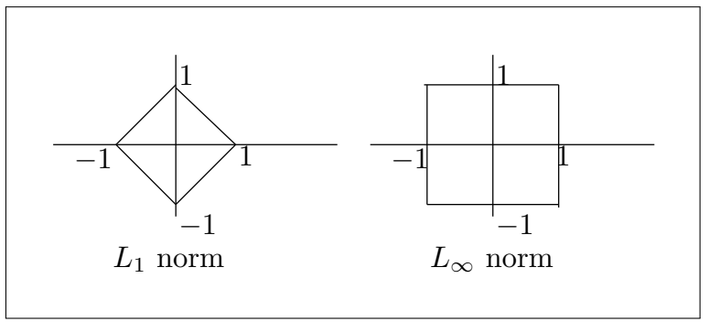
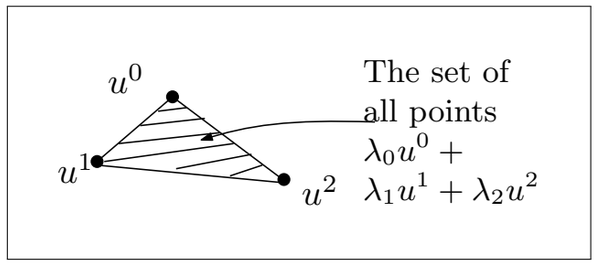


Figure 1.7: L_1 and L_∞ norms

Reader Do the above for $\Re^3$.

Reader Suppose U is convex, and $u^0, u^1, \cdots, u^m \in U$ and $\lambda_0, \lambda_1, \cdots, \lambda_m \geq 0$ with $\sum_{i=1}^{m} \lambda_i = 1$, then show that $\sum_{i=1}^{m} \lambda_i u^i \in U$. (See HW 2). For three points, u^0, u^1, u^2, then $\sum_{i=1}^{m} \lambda_i u^i$ is the mixture, which is convex set between all the three points:

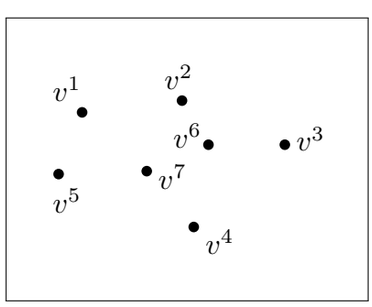


1.5.1 Polytope

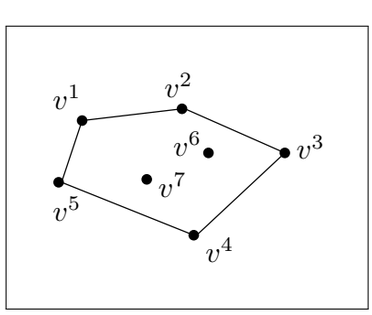
In words, these are flat sided shapes, which are convex. Let $v^1, v^2, \cdots, v^m \in \Re^n$ be given. We call set U a polytope generated by v^i if U is a set of mixtures of v^i. That is,

$$U = \left\{ \sum_{i=1}^{m} \lambda_i v^i : \lambda_i \geq 0 \text{ and } \sum_{i=1}^{m} \lambda_i = 1 \right\}$$

The following are some illustration. Given these points

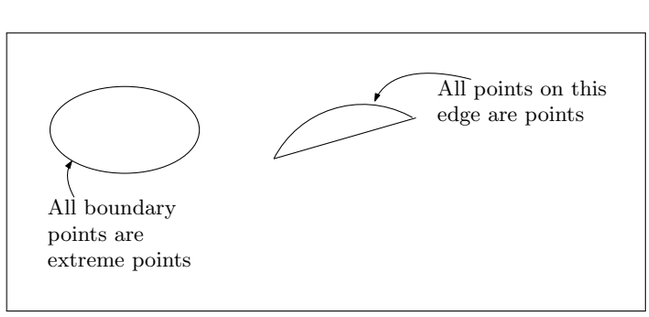


The generated polytope is



Notice that the points v^6, v^7 are redundant and have not been used. In higher dimensions, it will be harder to know which are the vertices of the extreme points.

Extreme points Let $U \subseteq \Re^n$ be convex set. A point $u \in U$ is said to be an extreme point if the following holds: u can ***not*** be written as a convex combination of other points $u^0, u^1, \cdots$. Examples:

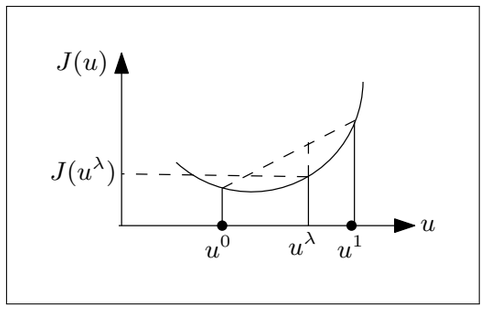


1.5.2 Convex functions

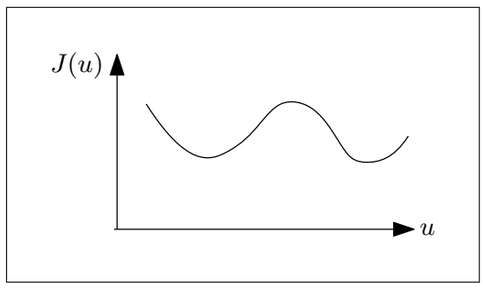
So far we talked about convex sets. Now we will talk about convex functions. $J(u) : \Re^n \to \Re$ is said to be convex function is the following is satisfied. Given any $u^0, u^1 \in U \subseteq \Re^n$ and $\lambda \in [0,1]$ then

$$\boxed{J\left((1-\lambda)u^0 + \lambda u^1\right) \leq (1-\lambda) J\left(u^0\right) + \lambda J\left(u^1\right)}$$

In words, it means the function value $J(u)$ between 2 points, is always below the straight cord points joining $J\left(u^0\right)$ and $J\left(u^1\right)$



For example, the following is not a convex function.

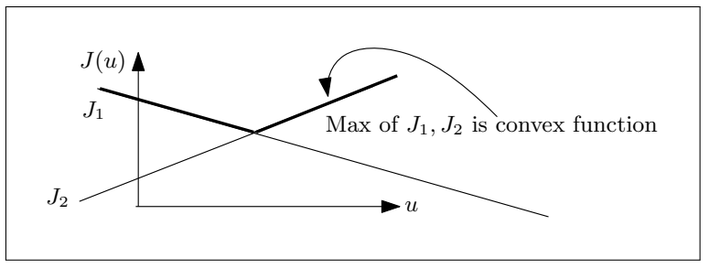


But the above is convex over some regions. But overall, it is not a convex function. We can not use this definition to check if a function is convex for higher dimensions. In that case, we have to use the Hessian to check.

Reader A function $J(u)$ is concave if $-J(u)$ is convex. Example: e^{au} is convex. Any linear function is convex function. $a^T u + b$ is convex. Also $-\log(u)$ is a convex function. And $J(u) = au^2 + bu + c$ with $a > 0$ is a convex function. What about the following?

$$\max\{J_1, J_2\} = \max\left\{(a_1)^T u + b_1, (a_2)^T u + b_2\right\}$$

Is it convex function? The pointwise maximum of two or more convex functions is a convex function.



Reader Suppose $J_i : \Re^n \to \Re$ are convex functions, not necessarily linear, for $i = 1, \cdots m$, then $J(u) = \max\{J_1(u), \cdots, J_m(u)\}$ is convex function. Hint, use $(1-\lambda)J(u)+\lambda J(u) = (1-\lambda)\max(\cdots)+\cdots$.

Next is to connect convex functions to convex sets.

1.6 Lecture 6. Thursday, February 4, 2016

1.6.1 Convex functions and convex sets

An interpretation of convex function in 1D is bowl shaped. But pictures are only for low dimensions. Convex applications have taken off in the last 15 years. For example, CVX software. Why is convex so great? There are useful properties of convex functions

1. Every local minimum is also global. This is important, since once we converge to a minimum, we can stop, as there will not be any better.

2. If the objective function is strict convex, then the minimum found is unique. If it is not strict convex, then there are other minimums of the same value, hence the minimum is not unique.

3. In quadratic programming, positive definite is the same as convex.

Reader Suppose $U \subseteq \Re^n$ is convex set, and $J : U \to \Re$ is convex function. Then show the set of minimizer elements $U^* = \left\{u \in U, J(u) = \min_{u \in U} J(u)\right\}$ is a convex set.

Other nice properties of convex functions is that point wise maximum of convex functions is also a convex function. The maximum over an indexed collection is also a convex function. Let I be a set, perhaps uncountable, for each $i \in I$, suppose we have a convex function $J_i : \Re^n \to \Re$ is convex. Let $J(u) = \sup_{i \in I} J_i(u)$ and assume $J(u) < \infty$.

Reader Show that $J(u)$ is convex function. Example: $J_q(u) = 6u^2 + \left(6q - \cos q\right) + e^{-q}$. Where $\|q\| \leq 1$. Then $J(u) = \max_{|q| \leq 1} J_q(u)$.

1.6.2 convex functions and convex sets relation

Given $J : \Re^n \to \Re$, define *epiJ* as set

$$\boxed{\left\{(u, v) \in \Re^{n+1} : v \geq J(u)\right\}}$$

$J(u)$ is convex function and *epiJ* is a convex set. See HW2 *epi* problem.

1.6.3 Criterion for convexity, Gradient and Hessian

Begin with $J : \Re^n \to \Re$. How to check $J(u)$ is convex? We assume $J(u)$ is twice differentiable, called C^2. And also assume U is open and convex set.

Definition: Gradient $\nabla J(u) = \left[\frac{\partial J}{\partial u_1} \quad \frac{\partial J}{\partial u_1} \quad \cdots \quad \frac{\partial J}{\partial u_n}\right]^T \in \Re^n$.

The Hessian

$$\nabla^2 J(u) = \begin{bmatrix} \frac{\partial J^2}{\partial u_1^2} & \frac{\partial^2 J}{\partial u_1 \partial u_2} & \cdots & \frac{\partial^2 J}{\partial u_1 \partial u_n} \\ \frac{\partial^2 J}{\partial u_2 \partial u_1} & \frac{\partial J^2}{\partial u_2^2} & \cdots & \frac{\partial^2 J}{\partial u_2 \partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u_n \partial u_1} & \frac{\partial^2 J}{\partial u_n \partial u_2} & \cdots & \frac{\partial J^2}{\partial u_n^2} \end{bmatrix}$$

Reader Why the Hessian is always symmetric? (Order of differentiation does not matter).

Positive definite Hessian is the same as saying second derivative is greater than zero.

1.6.4 Hessian theorem

Suppose $J : \Re^n \to \Re$ is C^2. Then J is convex function in U iff $\nabla^2 J(u)$ is positive semi-definite matrix for all $u \in U$.

Proof We first proof for $n = 1$. Sufficiency: Suppose $\nabla^2 J(u) \geq 0$ for all $u \in U$, (i.e. this is the same as saying $\frac{\partial^2 J}{\partial u^2} > 0$, and let u^0, u^1 be given in U. We need to show that for $\lambda \in [0,1]$, that $J\left(\lambda u^0 + (1-\lambda) u^1\right) \leq \lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right)$. We write

$$J\left(u^\lambda\right) = J\left(u^0\right) + \int_{u^0}^{u^\lambda} J'(\xi)\, d\xi$$

Since $J'(\xi)$ non-decreasing and $J''(\xi) > 0$, so the above can be upper bounded as

$$J\left(u^\lambda\right) \leq J\left(u^0\right) + J'\left(u^\lambda\right)\left(u^\lambda - u^0\right) \tag{1}$$

Similarly,

$$\begin{aligned} J\left(u^1\right) &= J\left(u^\lambda\right) + \int_{u^\lambda}^{u^1} J'(\xi)\, d\xi \\ J\left(u^1\right) &\geq J\left(u^\lambda\right) + J'\left(u^\lambda\right)\left(u^1 - u^\lambda\right) \\ J\left(u^\lambda\right) &\leq J\left(u^1\right) + J'\left(u^\lambda\right)\left(u^\lambda - u^1\right) \end{aligned} \tag{2}$$

Therefore $\lambda \overbrace{J\left(u^\lambda\right)}^{(1)} + (1-\lambda) \overbrace{J\left(u^\lambda\right)}^{(2)}$ using (1) and (2) gives

$$\lambda J\left(u^\lambda\right) + (1-\lambda) J\left(u^\lambda\right) \leq \lambda\left[J\left(u^0\right) + J'\left(u^\lambda\right)\left(u^\lambda - u^0\right)\right] + (1-\lambda)\left[J\left(u^1\right) + J'\left(u^\lambda\right)\left(u^\lambda - u^1\right)\right]$$

Hence

$$J\left(u^\lambda\right) \leq \lambda\left[J\left(u^0\right) + u^\lambda J'\left(u^\lambda\right) - u^0 J'\left(u^\lambda\right)\right] + \left[J\left(u^1\right) + J'\left(u^\lambda\right)\left(u^\lambda - u^1\right)\right] - \lambda\left[J\left(u^1\right) + J'\left(u^\lambda\right)\left(u^\lambda - u^1\right)\right]$$

Therefore

$$\begin{aligned} J\left(u^\lambda\right) &= \lambda J\left(u^0\right) + \lambda u^\lambda J'\left(u^\lambda\right) - \lambda u^0 J'\left(u^\lambda\right) + J\left(u^1\right) + u^\lambda J'\left(u^\lambda\right) - u^1 J'\left(u^\lambda\right) - \left[\lambda J\left(u^1\right) + \lambda J'\left(u^\lambda\right)\left(u^\lambda - u^1\right)\right] \\ &= \lambda J\left(u^0\right) + \lambda u^\lambda J'\left(u^\lambda\right) - \lambda u^0 J'\left(u^\lambda\right) + J\left(u^1\right) + u^\lambda J'\left(u^\lambda\right) - u^1 J'\left(u^\lambda\right) - \lambda J\left(u^1\right) - u^\lambda \lambda J'\left(u^\lambda\right) + u^1 \lambda J'\left(u^\lambda\right) \\ &= \lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right) - \lambda u^0 J'\left(u^\lambda\right) + u^\lambda J'\left(u^\lambda\right) - u^1 J'\left(u^\lambda\right) + u^1 \lambda J'\left(u^\lambda\right) \\ &= \left[\lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right)\right] + \lambda J'\left(u^\lambda\right)\left(u^1 - u^0\right) + J'\left(u^\lambda\right)\left(u^\lambda - u^1\right) \end{aligned}$$

But $u^\lambda = \lambda u^0 + (1-\lambda) u^1$, hence the above becomes

$$\begin{aligned} J\left(u^\lambda\right) &\leq \left[\lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right)\right] + \lambda J'\left(u^\lambda\right)\left(u^1 - u^0\right) + J'\left(u^\lambda\right)\left(\lambda u^0 + (1-\lambda) u^1 - u^1\right) \\ &= \left[\lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right)\right] + \lambda J'\left(u^\lambda\right)\left(u^1 - u^0\right) + J'\left(u^\lambda\right)\left(\lambda u^0 + u^1 - \lambda u^1 - u^1\right) \\ &= \left[\lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right)\right] + \lambda J'\left(u^\lambda\right)\left(u^1 - u^0\right) + J'\left(u^\lambda\right)\left(\lambda u^0 - \lambda u^1\right) \\ &= \left[\lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right)\right] + \lambda J'\left(u^\lambda\right)\left(u^1 - u^0\right) - \lambda J'\left(u^\lambda\right)\left(u^1 - u^0\right) \\ &= \lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right) \end{aligned}$$

Hence we showed that $J\left(u^\lambda\right) \leq \lambda J\left(u^0\right) + (1-\lambda) J\left(u^1\right)$. Same idea can be used to establish necessity. We now establish $\nabla^2 J(u) > 0$ and convexity next time. We carry this to n dimensions.

1.7 Lecture 7, Tuesday, February 9, 2016

For $n = 1$, Let $J : U \to \Re^n$, where U is open set (so that we can differentiate on it), and convex set. Let J is C^2. J is convex iff $\frac{d^2 J}{du^2} \geq 0$ for all $u \in U$. We are trying to establish the Hessian theorem. Now we want to show the above for $n > 1$. We start with the bridging lemma, which will use to proof the Hessian theorem.

1.7.1 The Bridging Lemma

Given $J : U \to \Re^n$, and U is convex set, we want to know if J is a convex function. The lemma says that J is convex iff the following condition holds:

Given any $\boldsymbol{u} \in U$, $z \in \Re^n$, then the function $\tilde{J}(\lambda) = J(\boldsymbol{u} + \lambda z)$ is convex on the set $\Lambda = \{\boldsymbol{u}, \lambda z \in \boldsymbol{U}\}$. Notice that the function $\tilde{J}(\lambda)$ is scalar valued. It depends on scalar λ. Hence we say $\tilde{J}(\lambda) \doteqdot \Re \to \Re$. This lemma says that the function $\tilde{J}(\lambda)$ is convex in any direction we move to from $\boldsymbol{u}$ in the direction of z within the set U iff J is convex function.

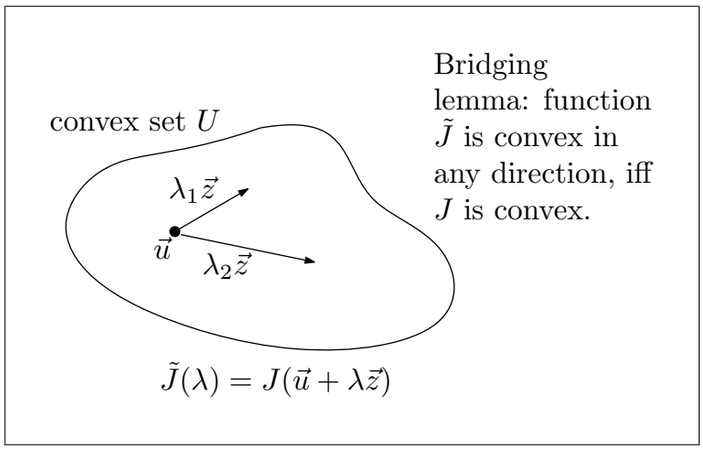


Figure 1.8: Bridging lemma

Proof See also the handout. Necessity: Assume J is convex function. We must show that $\tilde{J}$ is convex function. Pick $z \in \Re$ and $\lambda \in [0,1]$ and any scalars $\alpha^0, \alpha^1 \in \Lambda$. We must show that

$$\tilde{J}\left(\lambda\alpha^0 + (1-\lambda)\alpha^1\right) \leq \lambda\tilde{J}\left(\alpha^0\right) + (1-\lambda)\tilde{J}\left(\alpha^1\right)$$

Indeed, from $\tilde{J}(\lambda) \doteqdot J(\boldsymbol{u} + \lambda z)$, then

$$\tilde{J}\left(\lambda\alpha^0 + (1-\lambda)\alpha^1\right) \doteqdot J\left(\boldsymbol{u} + \left(\lambda\alpha^0 + (1-\lambda)\alpha^1\right)\boldsymbol{z}\right) \tag{1}$$

$$\doteqdot J\left(\lambda\left(\boldsymbol{u} + \alpha^0\boldsymbol{z}\right) + (1-\lambda)\left(\boldsymbol{u} + \alpha^1\boldsymbol{z}\right)\right) \tag{2}$$

Reader Going from (1) to (2) above is just a rewriting and manipulation only. Now since J is assumed convex, then

$$J\left(\lambda\left(\boldsymbol{u} + \alpha^0\boldsymbol{z}\right) + (1-\lambda)\left(\boldsymbol{u} + \alpha^1\boldsymbol{z}\right)\right) \leq \lambda J\left(\boldsymbol{u} + \alpha^0\boldsymbol{z}\right) + (1-\lambda)J\left(\boldsymbol{u} + \alpha^1\boldsymbol{z}\right)$$

Therefore (1) becomes

$$\begin{aligned}\tilde{J}\left(\lambda\alpha^0 + (1-\lambda)\alpha^1\right) &\leq \lambda J\left(\boldsymbol{u} + \alpha^0\boldsymbol{z}\right) + (1-\lambda)J\left(\boldsymbol{u} + \alpha^1\boldsymbol{z}\right)\\ &\leq \lambda\tilde{J}\left(\alpha^0\right) + (1-\lambda)\tilde{J}\left(\alpha^1\right)\end{aligned}$$

Hence $\tilde{J}$ is convex function. QED. Now to proof sufficiency. Assume that $\tilde{J}$ is convex, we need to show that this implies that J is convex on U. Since $\tilde{J}$ then

$$\begin{aligned}\tilde{J}\left((1-\lambda)\alpha^0 + \lambda\alpha^1\right) &\leq (1-\lambda)\tilde{J}\left(\alpha^0\right) + \lambda\tilde{J}\left(\alpha^1\right)\\ &= (1-\lambda)J\left(u + \alpha^0 z\right) + \lambda J\left(u + \alpha^1 z\right)\end{aligned} \tag{3}$$

But

$$\tilde{J}\left((1-\lambda)\alpha^0 + \lambda\alpha^1\right) = J\left(u + \left[(1-\lambda)\alpha^0 + \lambda\alpha^1\right]z\right) \tag{4}$$

$$= J\left((1-\lambda)\left(u + \alpha^0 z\right) + \lambda\left(u + \alpha^1 z\right)\right) \tag{5}$$

Where the main trick was going from (4) to (5) by just rewriting, so it match what we have in (3). Now replacing (5) into LHS of (3) we find

$$J\left((1-\lambda)\left(u + \alpha^0 z\right) + \lambda\left(u + \alpha^1 z\right)\right) \leq (1-\lambda)J\left(u + \alpha^0 z\right) + \lambda J\left(u + \alpha^1 z\right)$$

Let $\left(u + \alpha^0 z\right) \equiv u^0, u + \alpha^1 z \equiv u^1$, both in U, then the above becomes

$$J\left((1-\lambda)u^0 + \lambda u^1\right) \leq (1-\lambda)J\left(u^0\right) + \lambda J\left(u^1\right)$$

Hence J is convex function. QED. Now the bridging lemma is proved. we use it to proof the Hessian theorem.

1.7.2 The Hessian Theorem, strong local minimum

Let $J = U \to \Re$, where U is open set in $\Re^n$. Hessian theorem says that J is convex function on U iff $\nabla^2 J(u)$ is PSD (positive semi-definite) evaluated at each $u \in U$.

Reader Suppose $\nabla^2 J(u)$ is PSD, does this imply strict convexity on $J(u)$? Answer: No. Need an example.

See handout Hessian for proof.

Algorithms We will now start new chapter. Looking at algorithms to find optimal of $J(u)$.

Preliminaries: Begin with $J:\Re^n\to\Re$.

Strong local minimum u^* is strong local minimum if there exists $\delta>0$ such that $J(u^*)<J(u)$ for all u such that $\|u^*-u\|<\delta$.

We say u^* is global minimum if $J(u^*)\le J(u)$ for all u. Henceforth, $J(u)$ is C^2. From undergraduate calculus, u^* is strong local minimum if the following is satisfied: (for $n=2$)

1. $\left.\frac{\partial J}{\partial u_1}\right|_{u^*}=0,\ \left.\frac{\partial J}{\partial u_2}\right|_{u^*}=0$
2. $\left.\frac{\partial^2 J}{\partial u_1^2}\right|_{u^*}>0, \left(\frac{\partial^2 J}{\partial u_1^2}\right)\left(\frac{\partial^2 J}{\partial u_1^2}\right)-\left(\frac{\partial^2 J}{\partial u_1\partial u_2}\right)^2>0$

For $J:\Re^n\to\Re$, in other words, in higher dimensions, define gradient

$$\boxed{(\nabla J(u))^T=\begin{bmatrix}\frac{\partial J}{\partial u_1}\\ \frac{\partial J}{\partial u_2}\\ \vdots\\ \frac{\partial J}{\partial u_n}\end{bmatrix}}$$

Normally we consider gradient as column vector. Then we say that a point $u^*\in\Re^n$ is strong local min. if $\nabla J(u)=0$, and $\nabla^2 J(u)>0$.

Proof: Suppose u^* is strong local min. Then looking at neighborhood of u^*, let $v\in\Re^n$ be arbitrary. Look at $J(u^*+v)$ and expand in Taylor series.

$$J(u^*+v)=J(u^*)+\nabla J(u^*)v+v^T\nabla^2 J(u)v+H.O.T\left(O\left(\|v\|^3\right)\right)$$

Since u^* is strong local min. then $\nabla J(u^*)=0$. Hence $J(u^*+v)=J(u^*)+v^T\nabla^2 J(u)v+H.O.T$. Since $J(u^*+v)>J(u^*)$ (since strong local minimum), then this implies that

$$v^T\nabla^2 J(u)v>0$$

Since $v^T\nabla^2 J(u)v$ dominate over $H.O.T.$. This complete the proof.

1.8 Lecture 8. Thursday, February 11, 2016

Reminder, test 1 next Thursday Feb. 18, 2016. Up to and including HW 3. Today's lecture on gradient based optimization. We developed two conditions. u^* is strong local minimum when $\nabla J(u^*)=0$ and $\nabla^2 J(u^*)<0$.

1.8.1 gradient based optimization and line searches

We mention line searches. It is about optimization for one variable functions only. There will be reading assignment on line search. Some methods used are

1. Golden section.
2. Fibonacci.
3. Bisection

And more.

1.8.2 Optimal gain control problems, Lyapunov equation

We will now set up application areas. Optimal gain control and circuit analysis problems. In general we have

$$\dot{x}=Ax+Bu$$

Where $\dot{x}$ is $n\times 1$, A is $n\times n$, B is $n\times m$ and u is $m\times 1$. System has n states. u is the input. This can be voltage or current sources. u is the control and x is the state. We want to select u so that $x(t)$ behaves optimally. Classical setup is to use state feedback

$$u=kx+v$$

Where k is $m \times n$ is called the feedback gain matrix and v is extra input but we will not use it. It is there for extra flexibility if needed. We use optimization to determine k. Entries of k_{ij} are our optimization variables.

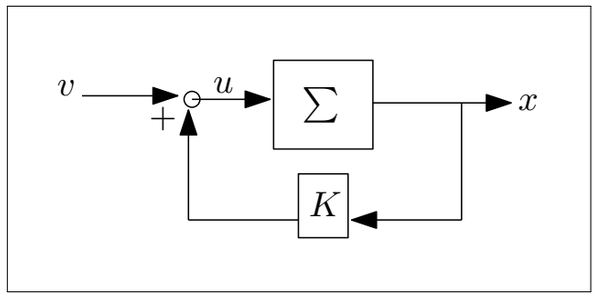


Figure 1.9: State feedback

Often, with $u = 0$, the system $\dot{x} = Ax$ may be unstable or have overshoot. We will set up a performance objective aimed at reducing or eliminating the badness of the original response (with no feedback control). Let

$$J(k) = \int_0^\infty \overbrace{x^T(t)\,x(t)}^{\|x\|^2} dt$$

In the above, $J(k)$ is implicit function of k. This cost function was found to work well in practice. We now want to make $J(k)$ explicit in k. We can solve for $x(t)$ from

$$\begin{aligned}\dot{x} &= (A + Bk)\,x\\ x &= x(0)\,e^{(A+Bk)t}\end{aligned}$$

Then $J(k) = \int_0^\infty x^T(0)\,e^{(A+Bk)^T t} dt$. But this is not practical to use. This is not closed form and hard to compute. So how can we come up with closed form for $J(k)$ which is easier to work with? Let us look at the closed loop. Let $v = 0$ and we have

$$\begin{aligned}\dot{x} &= Ax + Bkx\\ &= (A + Bk)\,x\\ &= A_c x\end{aligned}$$

Where A_c is the closed loop system matrix. Let us find a matrix P if possible such that

$$d\left(x^T(t)\,Px(t)\right) = -x^T(t)\,x(t)$$

So that now

$$\begin{aligned}J(k) &= \int_a^b x^T(t)\,x(t)\,dt\\ &= -\int_a^b d\left(x^T(t)\,Px(t)\right)\\ &= \int_b^a d\left(x^T(t)\,Px(t)\right)\\ &= x^T(a)\,Px(a) - x^T(b)\,Px(b)\end{aligned}$$

Can we find P?

$$\begin{aligned}d\left(x^T Px\right) &= x^T P\dot{x} + \dot{x}^T Px \stackrel{?}{=} -x^T x\\ &= x^T P(A_c x) + (A_c x)^T Px \stackrel{?}{=} -x^T x\\ &= x^T P(A_c x) + \left(x^T A_c^T\right) Px \stackrel{?}{=} -x^T x\end{aligned}$$

Bring all the x to LHS then

$$A_c^T x + PA_c = -I$$

Where I is the identity matrix. This is called the Lyapunov equation. This is the equation to determine P. Without loss of generality, we insist on P being symmetric matrix. Using

this P, now we write

$$\begin{aligned} J(k) &= \int_0^\infty x^T(t)\, x(t)\, dt \\ &= -\int_0^\infty d\left(x^T P x\right) \\ &= x^T P x\Big|_\infty^0 \\ &= x^T(0)\, P x(0) - x^T(\infty)\, P x(\infty) \end{aligned}$$

For stable system, $x(\infty) \to 0$ (remember that we set $v = 0$, so there is no external input, hence if the system is stable, it must end up in zero state eventually). Therefore

$$J(k) = x^T(0)\, P x(0)$$

With k satisfying

$$A_c^T(k)\, x + P A_c(k) = -I$$

Example Let $y'' = u$. Hence $x_1' = x_2, x_2' = u$. Note, this is not stable with $u = 0$. Using linear state feedback,

$$\begin{aligned} u &= kx \\ u &= \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned}$$

Hence

$$\begin{aligned} x' &= Ax + Bu \\ &= Ax + Bkx \\ &= (A + Bk)\, x \\ \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} &= \left(\overbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}^{A} + \overbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}^{B} \overbrace{\begin{bmatrix} k_1 & k_2 \end{bmatrix}}^{k} \right) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \left(\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ k_1 & k_2 \end{bmatrix} \right) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \overbrace{\begin{bmatrix} 0 & 1 \\ k_1 & k_2 \end{bmatrix}}^{A_c} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned}$$

For stable closed loop, we need $k_1 < 0, k_2 < 0$ by looking at characteristic polynomial roots. Now we solve the Lyapunov equation.

$$A_c^T P + P A_c = -I$$

$$\begin{bmatrix} 0 & 1 \\ k_1 & k_2 \end{bmatrix}^T \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ k_1 & k_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & k_1 \\ 1 & k_2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ k_1 & k_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Solving for P gives

$$P = \begin{bmatrix} \frac{k_2^2 - k_1 + k_1^2}{2k_1 k_2} & -\frac{1}{2k_1} \\ -\frac{1}{2k_1} & \frac{1 - k_1}{2k_1 k_2} \end{bmatrix}$$

With $x(0)=\begin{bmatrix}1\\1\end{bmatrix}$, then

$$\begin{aligned}J(k) &= x(0)^T P x(0)\\ &= \begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}\frac{k_2^2-k_1+k_1^2}{2k_1k_2} & -\frac{1}{2k_1}\\ -\frac{1}{2k_1} & \frac{1-k_1}{2k_1k_2}\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\\ &= \frac{k_1^2+k_2^2-2k_1-2k_2+1}{2k_1k_2}\end{aligned}$$

Here is Matlab script to generate the above

```
syms k1 k2 p11 p12 p21 p22;
Ac   = [0 1;k1 k2];
P    = [p11 p12;p21 p22];
eq   = Ac.'*P+P*Ac==-eye(2);
sol  = solve(eq,{p11,p12,p21,p22});
P    = subs(P,sol)
x0   = [1;1];
J    = simplify(x0'*P*x0)
```

.

Let $k_1=k_2=k<0$, we obtain

$$J(k)=\frac{2k^2-4k+1}{2k^2}$$

As $k\to-\infty$ then $J^*\to 1$. Therefore J^* can never get to zero. This means there is no k_1^*,k_2^* such that $\nabla J(k^*)=0$. Set k is not compact. Not coercive either. This is ill posed problem. This can be remedied by changing the control to

$$J(k)=\int_0^\infty x^T x dt+\lambda\int_0^\infty u^T u dt$$

1.9 Lecture 9. Tuesday, February 16, 2016

1.9.1 keywords for next exam 1

1. Common sense optimization. Farming problem. Explicit vs. Implicit.
2. Minimizing $J(u)$, $\inf, \sup$
3. We want to know ahead of time if minimum can be attained. J^* but u^* might not exist.
4. Multilinear function. u^* is at a vertex. But they grow as 2^n where n is number of variables.
5. When can we be sure u^* exist? if the set is compact, we talked about W-B theory, which is used to show u^* exist always for compact sets. If the set is not compact but coercive, then we can compact it.
6. Convex sets and convex functions. For convex sets, when we find u^* then the local minimum is also a global minimum.
7. Special case of convex sets is polytope. Polyhedron is a polytope but can be unbounded.
8. Strong local minimum is when $\nabla J(u)=0$ and $\nabla^2 J(u)>0$. To test for convexity, find the Hessian. If the Hessian is semi positive definite, then it is convex.
9. Optimal gain control, Lyapunov equation.
10. If there are proofs, they will be simple, such as show the sum of two convex functions is also convex function.

1.9.2 Gradient based optimization

Starting new chapter. Gradient based optimization. Many algorithms involve line searches. In other words, optimization in $\Re^n$ is often solved by performing many optimization (line searches) in $\Re$.

Optimization algorithm: Starting with $\boldsymbol{u}^k$ and direction $\boldsymbol{v}$ we study $J\left(\boldsymbol{u}^k + h\boldsymbol{v}\right)$ where h is the step size. This is called line search. $h \in [0, h_{\max}]$. We want to use optimal step size h^*. Once found, then

$$\boxed{\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + h^*\boldsymbol{v}}$$

Reader Read and learn about line search. Bisection, Golden section, Fibonacci and many more. We will not cover this in this course. We also want to minimize the number of function evaluations, since these can be expensive.

One way to do line search, is to do $h = 0\delta h_{\max}$ and then evaluate $J(h)$ and pick the minimizing h^*. For stopping criteria, we can check for the following

1. $\left\|u^{k+1} - u^k\right\| \le \delta$
2. $\left\|J\left(u^{k+1}\right) - J\left(u^k\right)\right\| \le \delta$
3. $\left\|\frac{J\left(u^{k+1}\right) - J\left(u^k\right)}{J\left(u^{k+1}\right)}\right\| \le \delta$
4. $\left\|\nabla J\left(u^k\right)\right\| \le \delta$

We also need to pick a starting point for the search. This is $\boldsymbol{u}^0$. What if we do not know where to start? We can pick multiple starting locations. And pick the best result obtained.

Reader Find $\min_{\|v\|=1} J(\boldsymbol{u} + \boldsymbol{v})$. Show optimal $\boldsymbol{v}$ is

$$\boxed{v^* = \frac{-\nabla J(u)}{\|\nabla J(u)\|}}$$

This is called myopic local terrain. Gets us to local minimum. The steepest descent algorithm is the following:

1. Select u^0 (starting point)
2. Find step size h
3. Iterate. While $\left\|\nabla J\left(u^k\right)\right\| > \delta$ then $\boxed{u^{k+1} = u^k - h\frac{\nabla J(u)}{\|\nabla J(u)\|}}$
4. Update counter and go back to step 3 above.

See my class study notes for detailed algorithm of all search methods we did in this course.

Later we will study conjugate gradient methods.

Example: Let $u^0 = [1,1]$. Let $h = 0.1$. Let $J(u) = u_1^2 + 2u_2^2 - 6u_1u_2 + 2u_1 + u_2 + 4$. Then

$$\nabla J(u) = \begin{bmatrix} 2u_1 - 6u_2 + 2 \\ 6u_2 - 6u_1 + 1 \end{bmatrix}$$

So $\nabla J\left(u^0\right) = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$ and $\left\|\nabla J\left(u^0\right)\right\| = \frac{1}{\sqrt{5}}$. Hence

$$\begin{aligned} u^1 &= u^0 - h\begin{bmatrix} -2 \\ 1 \end{bmatrix}\frac{1}{\sqrt{5}} \\ &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0.1\begin{bmatrix} -2 \\ 1 \end{bmatrix}\frac{1}{\sqrt{5}} \\ &= \begin{bmatrix} 1.0894 \\ 0.95528 \end{bmatrix} \end{aligned}$$

Reader Find u^2.

1.10 Lecture 10. Thursday, February 18, 2016 (Exam 1)

Exam 1

1.11 Lecture 11. Tuesday, February 23, 2016

Watch for HW4 going out today. Implementation of steepest descent with optimal step size. See handout circuit for 2 stage amplifier.

1.11.1 Steepest descent

As the number of stages increases it becomes harder to analytically determine the optimal capacitance of each stage to produce maximum power. For two stages, by direct circuit analysis we obtain

$$J(u) = (11 - u_1 - u_2)^2 + (1 + u_1 + 10u_2 - u_1u_2)^2$$

Where u_i is capacitance. There are two optimal values, they are $u^* = (10,1)$ which is a maximum and $u^* = (13,4)$ which is a minimum. There is also a minimum at $(7,-2)$.

Geometric insight on what might go wrong with steepest descent: Gradient algorithms work best from a far but as they get close to the optimal point, there are better algorithms such as the generalized Newton Raphson method which works best when close to the optimal point. If the step size h is big, we approach the optimal fast, but because the step size is large, we can overshoot and will end up oscillating around the optimal point. If h is too small, the search will become very slow. Hence we use steepest descent but with optimal step size, where the step size is calculated at each step. Ingredients of the steepest descent algorithm are:

1. Initial guess u^0
2. maximum step size H. Here we have h_k which is the step size used at each iteration.
3. Iteration step. When at u^k define $\boxed{\tilde{J}(h) = J\left(u^k - h\frac{\nabla J\left(u^k\right)}{\left\|\nabla J\left(u^k\right)\right\|}\right)}$ and carry a line search to find optimal h which minimized $\tilde{J}(h)$, then $h_k = h_*$
4. $u^{k+1} = u^k - h_k\frac{\nabla J(u^k)}{\left\|\nabla J(u^k)\right\|}$
5. Stopping criteria. Decide how to stop the search. $\left\|\nabla J\left(u^k\right)\right\| \le \varepsilon$

Reader: Consider oscillation issue.

Convergence result. From Polak. Let $J(u)$ be smooth and differentiable. Let u^* be strong local minimum. Assume constant $0 \le m \le M$ s.t.

$$mu^Tm \le u^T\nabla^2 J(u)\,u \le Mu^TM$$

In neighborhood of u^*. This criteria says that there is a good convexity and a bad convexity. What does this mean? We'll say more about this. In the neighborhood of u^* let $\theta = \frac{m}{M}$. Interpretation:

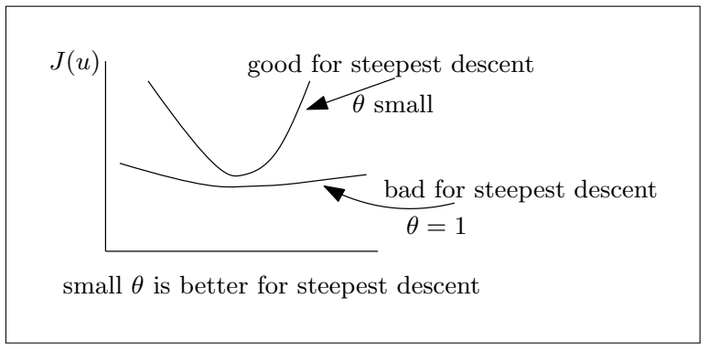


Figure 1.10: Steepest descent diagram

Define $E = J\left(u^0\right) - J(u^*)$ then Polak says

$$0 \leq J\left(u^k\right) - J(u^*) \leq E\theta^k$$

Best case is when E is small and θ is small. This is local result.

1.11.2 Classifications of Convergence

Convergence can be

1. Linear
2. Quadratic
3. Superlinear

These are the three convergence types we will cover. The second algorithm has quadratic convergence, which is the generalized Newton-Raphson method. We will start on this now but will cover it fully next lecture.

The idea is to approximate $J(u)$ as quadratic at each step and obtain h_k. By assuming $J(k)$ is quadratic locally, we approximate $J(u)$ using Taylor and drop all terms after the Hessian. Now we find where the minimum is and use the step size to find u^{k+1}. More on this next lecture.

1.12 Lecture 12. Thursday, February 25, 2016

1.12.1 Quadratic optimization, superlinear convergence

We will start the class with a reader problem. Consider

$$J(u) = \frac{1}{2}u^T Au + b^T u + c \tag{1}$$

$$\nabla J(u) = Au^k + b \tag{2}$$

with A being symmetric positive definite (PSD) matrix. This is classic quadratic objective function. You can take a complete course on quadratic optimization. The global optimal is at the solution for $\nabla J(u) = 0$. Hence we write

$$\begin{aligned} \nabla J(u) &= 0 \\ Au^* + b &= 0 \\ u^* &= -A^{-1}b \end{aligned} \tag{3}$$

Note that since A is PSD (the Hessian is PSD), then we know that $J(u)$ is convex. Hence the local minimum is also a global minimum. Now we imagine we are doing steepest descent on this function and we are at iterate u^k with optimal step size, which we can make as large as we want. Hence we need to optimize

$$\tilde{J}(h) = J\left(u^k - h\left[\nabla J(u)\right]\right)$$

for h. Notice we did not divide by $\|\nabla J(u)\|$ here, since the step size is free to be as large as needed. Expanding the above using (2) gives

$$\begin{aligned}\tilde{J}(h) &= J\left(u^k - h\left(Au^k + b\right)\right)\\ &= J\left((I - hA)u^k - hb\right)\end{aligned}$$

Using (1) for RHS of the above gives

$$\tilde{J}(h) = \frac{1}{2}\left((I - hA)u^k - hb\right)^T A\left((I - hA)u^k - hb\right) + b^T\left((I - hA)u^k - hb\right) + c$$

The above is quadratic in h. The optimal h we are solving for. Simplifying gives

$$\tilde{J}(h) = \frac{1}{2}\left(Au^k + b\right)^T A\left(Au^k + b\right)h^2 - \left(Au^k + b\right)^T\left(Au^k + b\right)h + \text{ constant terms}$$

To find optimal h, then

$$\begin{aligned}\frac{d\tilde{J}(h)}{dh} &= 0\\ \left(Au^k + b\right)^T A\left(Au^k + b\right)h - \left(Au^k + b\right)^T\left(Au^k + b\right) &= 0\\ h^* &= \frac{\left(Au^k + b\right)^T\left(Au^k + b\right)}{\left(Au^k + b\right)^T A\left(Au^k + b\right)}\end{aligned}$$

In practice, we would need to check $\frac{d^2\tilde{J}(h)}{dh^2}$ also to make sure h is minimizer.

Reader Why does it take multiple iterations to get the common sense answer $u^* = -A^{-1}b$?

For quadratic objective function $J(u)$ we can obtain u^* in one step, using $u^* = -A^{-1}b$. This is the idea behind the generalized Newton-Raphson method. $J\left(u^k\right)$ is approximated as quadratic function at each step, and h^* is found from above. To elaborate, expanding by Taylor

$$J\left(u^k + \Delta u\right) = J\left(u^k\right) + \nabla J\left(u^k\right)^T \Delta u + \frac{1}{2}\Delta u^T \nabla^2 J\left(u^k\right)\Delta u + HOT$$

We approximate as quadratic by dropping higher order terms and optimize for Δu (same as h used earlier), and here $\nabla^2 J\left(u^k\right)$ is same as the A above also. Therefore we find

$$\Delta u^* = -\left[\nabla^2 J\left(u^k\right)\right]^{-1}\nabla J\left(u^k\right)$$

This converges in one step Δu^* if $J(u)$ was actually a quadratic function. Notice that Newton method is expensive if used repeatedly, as it requires finding Hessian at each step and also finding the inverse of it. The algorithm is: Initialize u^0. Then iterate, where

$$u^{k+1} = u^k - \left[\nabla^2 J\left(u^k\right)\right]^{-1}\nabla J\left(u^k\right) \tag{4}$$

Then check for convergence. If Hessian fails to be PSD, in this case $J\left(u^{k+1}\right)$ can end up increasing not decreasing. How to stop? We can try more iterations to see if $J\left(u^k\right)$ will decrease again.

Example Let

$$J(u) = (11 - u_1 - u_2)^2 + (1 + 10u_2 + u_1 - u_1u_2)^2$$

Pick $u^0 = (18, 3)$ then

$$\begin{aligned}\nabla J\left(u^0\right) &= \begin{bmatrix}40\\100\end{bmatrix}\\ \nabla^2 J\left(u^0\right) &= \begin{bmatrix}10 & 44\\44 & 130\end{bmatrix}\end{aligned}$$

We see here that $\nabla^2 J\left(u^0\right)$ is not PSD (determinant is negative). Now we do the iterate

equation (4), obtaining

$$\begin{aligned} u^1 &= u^0 - \begin{bmatrix} 10 & 44 \\ 44 & 130 \end{bmatrix} \begin{bmatrix} 40 \\ 100 \end{bmatrix} \\ &= \begin{bmatrix} 18 \\ 3 \end{bmatrix} - \begin{bmatrix} 10 & 44 \\ 44 & 130 \end{bmatrix} \begin{bmatrix} 40 \\ 100 \end{bmatrix} \\ &= \begin{bmatrix} 19.3 \\ 1.8 \end{bmatrix} \end{aligned}$$

If we look at the contour plot, we will see this point made $J(u)$ larger (away from optimal) but if we let it iterate more, it will turn and move back to the optimal point.

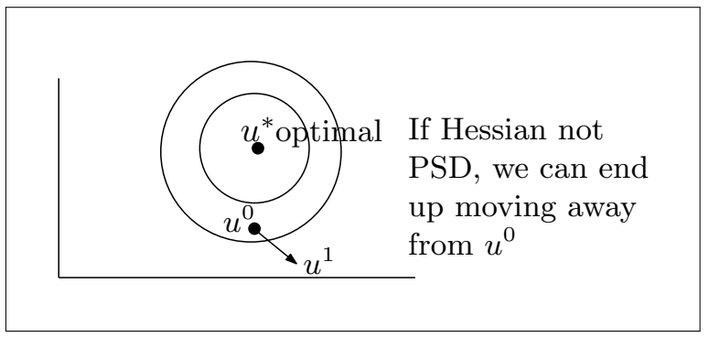


Figure 1.11: Hessian and optimal solution diagram

Now we will start on the third algorithm. Conjugate gradient algorithms (CG) are family of algorithms with property of superlinear convergence. It also has quadratic convergence.

1.12.2 Quadratic convergence

Quadratic convergence says the following: If $J(u)$ is positive definite quadratic form, the iterate $u^k \to u^*$ completes in finite number of steps N. This means if we give the algorithm a quadratic form function, it will converge in N steps to the optimal. The difference between this and Newton-Raphson, is that there is no Hessian to be calculated using this algorithm as the case was with Newton-Raphson. The conjugate direction algorithms work well from far away and also when close to the optimal point. (note: Steepest descent worked well from a far, but not when getting close to the optimal point u^*). The CG algorithms also have the property of superlinear convergence. This property do not apply to steepest descent.

1.12.3 Superlinear convergence

What is superlinear convergence? A sequence $\left\{u^k\right\}$ in $\Re^n$ is said to converge super-linearly to u^* if the following holds: Given any $\theta \in (0,1]$, then $\frac{\|u^k - u^*\|}{\theta^k} \to 0$. The important part of this definition is that the above should go to zero for any $\theta \in (0,1]$ and not some θ. Examples below illustrate this. Let $\left\{u^k\right\} = \frac{1}{k}$. Hence the sequence is $\left\{1, \frac{1}{2}, \frac{1}{3}, \cdots\right\}$. Clearly this sequence goes to $u^* = 0$. Does it converge super-linearly to u^*? Applying the definition

$$\frac{\left\|\frac{1}{k} - 0\right\|}{\theta^k} = \frac{1}{k\theta^k}$$

If we can find one θ that do not converge to zero, then we are done. Trying $\theta = \frac{3}{4}$, then the above becomes $\frac{4^k}{k3^k}$ which do not go to zero as $k \to \infty$. Hence this is not superlinear. How about $\left\{u^k\right\} = \frac{1}{k^2}$. This is still not superlinear. Similarly $\left\{u^k\right\} = \frac{1}{k^m}$. What about $\left\{u^k\right\} = \frac{1}{e^k}$. Here we get

$$\frac{\left\|\frac{1}{k} - 0\right\|}{\theta^k} = \frac{e^{-k}}{\theta^k}$$

Trying $\theta = \frac{1}{2}$ gives $\frac{2^k}{e^k}$ which is not superlinear (do not go to zero for large k). But if $\theta = \frac{2}{3}$ it will converge. But it has to converge for all θ, so $\frac{1}{e^k}$ is not superlinear. How about $\left\{u^k\right\} = \frac{1}{e^{k^2}}$

here we find it is superlinear. We obtain $\frac{e^{-k^2}}{\theta^k}$ and this goes to zero for any θ. To show this, use $\log$ on it and simplify. (Reader).

Next time we will go over conjugate direction algorithm in more details.

1.13 Lecture 13. Tuesday, March 1, 2016

1.13.1 Conjugate direction algorithms

Today lecture will be devoted to conjugate direction (C.D.) algorithms. We will start by remembering that there are many C.D. algorithms. From last lecture, be aware of: Superlinear convergence and quadratic convergence. The quadratic convergence concept is that on a P.S.D. (positive symmetric definite) form, the algorithm will converge in n steps or less (where n is the size A). Using exact arithmetic (not counting for floating point errors). We will proof this today. We will some preliminaries, then go over ingredients and go over examples, then go over properties of conjugate gradient.

Preliminaries: Let $A^{n\times n}$ be positive definite symmetric, then the pair of vectors u, v are said to be mutually conjugate w.r.t. A if

$$u^T A v = 0$$

This is generalization of orthogonality. Because we can take $A = I_n$ which is PSD.

Reader A set of distinct mutually conjugate vectors always exist for A. These are the eigenvectors of A. The proof starts with writing $Av = \lambda_1 v$ and $Au = \lambda_2 u$, then applying $u^T A v = 0$.

We will use this set of vectors as search directions. We will generate these vectors on the fly during the search and do line search along these directions. So instead of using $\nabla J(u)$ as the direction we did line search on when using steepest descent, we will now use the conjugate vectors instead.

Properties: Suppose $v^0, v^1, \cdots, v^{n-1}$ is a set of mutually conjugate vectors w.r.t A. (A is PSD). First step is to show these vectors are linearly independent.

Lemma: v^i are linearly independent. Proof: Suppose

$$\sum_{i=0}^{n-1} \alpha_i v^i = 0 \tag{1}$$

For scalars α_i. We must show that all $\alpha_i = 0$. Let use consider α_k. If we can show that $\alpha_k = 0$ for any k, then we are done. Multiply (1) by $\left(v^k\right)^T A$. Then

$$\left(v^k\right)^T A \sum_{i=0}^{n-1} \alpha_i v^i = 0$$

$$\sum_{i=0}^{n-1} \alpha_i \left(v^k\right)^T A v^i = 0$$

By mutual conjugate property, then all terms above vanish except $\alpha_k \left(v^k\right)^T A v^k$. Hence

$$\alpha_k \left(v^k\right)^T A v^k = 0$$

But A is PSD and $v^k \neq 0$, therefore $\alpha_k = 0$ is only choice. QED. We have proved that $v^0, v^1, \cdots, v^{n-1}$ are linearly independent So we can expand any vector $u \in \Re^n$ using these as basis vectors

$$u = \sum_{i=0}^{n-1} a_i v^i \tag{2}$$

Let us find the coefficients a_i. Premultiply by $\left(v^k\right)^T A$ both sides

$$\left(v^k\right)^T A u = \sum_{i=0}^{n-1} a_i \left(v^k\right)^T A v^i$$

As before, by mutual conjugate, the RHS becomes $a_k \left(v^k\right)^T A v^k$. Solving for a_k gives

$$a_k = \frac{\left(v^k\right)^T A u}{\left(v^k\right)^T A v^k}$$

Hence (2) becomes

$$u = \sum_{i=0}^{n-1} \frac{\left(v^i\right)^T A u}{\left(v^i\right)^T A v^i} v^i$$

This gives any vector u in terms of set of vectors v^i that are linearly independent

Conjugate directions algorithm ingredients are:

1. Initially given u^0. The starting guess vector
2. Iterative step u^k: Generate v^k a mutual conjugate vector to previous $n-1$ vectors v^i. For v^0 use $-\nabla J(u)$. Same as steepest descent.
3. Form line search with Max step H. To minimize $\tilde{J}(h) = J\left(u^k + h v^k\right)$. Notice there we used + sign and not − as with steepest descent. The direction takes care of the sign in this case.
4. Stopping criteria.

Example: Fletcher Reeves.

$$v^0 = -\nabla J(u)$$

$$v^{k+1} = -\nabla J\left(u^{k+1}\right) + \frac{\left\|\nabla J\left(u^{k+1}\right)\right\|^2}{\left\|\nabla J\left(u^k\right)\right\|^2} v^k$$

Reader Normalize above for implementation.

Reader What is A above? Where are these v^k vectors mutually conjugate? A is the Hessian. Note: These algorithms (C.D.) converge for convex $J(u)$. If $J(u)$ is not convex or we do not know, we need to put conditions to make sure it is converging.

For Polyak-Ribieve, see homework.

1.13.2 Quadratic convergence theorem

Consider quadratic form

$$J(u) = \frac{1}{2} u^T A u + b^T u + c$$

With $A = A^T$ and positive definite $n \times n$ matrix. Let $v^0, \cdots, v^{n+1}$ be mutually conjugate w.r.t. A. Let step size be as large as we want. The conjugate direction algorithm converges to optimal $u^* = -A^{-1} b$ in n steps or less

Proof

Let u^k be the k^{th} iterate. If $u^k = u^*$ and $k \le n$ then we are done. Without loss of generality, assume $u^k \neq u^*$. We must show that $u^n = u^*$. We first find h_k, the step size at iterate k. From

$$\begin{aligned} \tilde{J}(h) &= J\left(u^k + h v^k\right) \\ &= \frac{1}{2}\left(u^k + h v^k\right)^T A \left(u^k + h v^k\right) + b^T \left(u^k + h v^k\right) + c \end{aligned}$$

This is quadratic in h.

$$\tilde{J}(h) = \frac{1}{2}(v^k)^T A v^k h^2 + \left((v^k)^T A u^k + b^T v^k\right) h + \overbrace{\frac{1}{2} u^k A u^k + c}^{\text{constant term}}$$

Now taking derivative gives

$$\frac{d\tilde{J}(h)}{dh} = \left(v^k\right)^T A v^k h + \left(\left(v^k\right)^T A u^k + b^T v^k\right)$$

Setting this to zero and solving for h gives

$$\begin{aligned} h^* &= -\frac{\left(v^k\right)^T Au^k + b^T v^k}{\left(v^k\right)^T Av^k} \\ &= -\frac{\left(v^k\right)^T \left(Au^k + b\right)}{\left(v^k\right)^T Av^k} \end{aligned} \tag{3}$$

Hence

$$\begin{aligned} u^n &= u^0 + h_0 v^0 + \cdots + h_{n-1} v^{n-1} \\ &= u^0 + \sum_{k=0}^{n-1} h_k v^k \end{aligned}$$

Using (3) in the RHS of above, replacing each h_k with the optimal h at each iterate gives

$$\begin{aligned} u^n &= u^0 - \sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T \left(Au^k + b\right)}{\left(v^k\right)^T Av^k} \right) v^k \\ &= u^0 - \sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T b}{\left(v^k\right)^T Av^k} \right) v^k - \sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T Au^k}{\left(v^k\right)^T Av^k} \right) v^k \end{aligned}$$

Replacing u^k in the second term above in the RHS with $u^0 + \sum_{i=0}^{k-1} h_i v^i$ gives

$$u^n = u^0 - \sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T b}{\left(v^k\right)^T Av^k} \right) v^k - \sum_{k=0}^{n-1} \frac{\left(v^k\right)^T A \left(u^0 + \sum_{i=0}^{k-1} h_i v^i \right)}{\left(v^k\right)^T Av^k} v^k \tag{4}$$

But

$$\begin{aligned} \left(v^k\right)^T A \left(u^0 + \sum_{i=0}^{k-1} h_i v^i \right) &= \left(v^k\right)^T Au^0 + \sum_{i=0}^{k-1} h_i \left(v^k\right)^T v^i \\ &= \left(v^k\right)^T Au^0 \end{aligned}$$

Since all terms in $\sum_{i=0}^{k-1} h_i \left(v^k\right)^T v^i$ vanish by mutual conjugate property. Using this to simplify (4) gives

$$\begin{aligned} u^n &= u^0 - \sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T b}{\left(v^k\right)^T Av^k} \right) v^k - \sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T Au^0}{\left(v^k\right)^T Av^k} \right) v^k \\ &= u^0 - \sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T \left(Au^0 + b\right)}{\left(v^k\right)^T Av^k} \right) v^k \end{aligned}$$

Reader $\left(v^k\right)^T Au^0$ is expansion of u^0. Using this in the above reduces it to

$$u^n = -\sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T b}{\left(v^k\right)^T Av^k} \right) v^k$$

Insert AA^{-1} into the above gives

$$\begin{aligned} u^n &= -\sum_{k=0}^{n-1} \left(\frac{\left(v^k\right)^T AA^{-1} b}{\left(v^k\right)^T Av^k} \right) v^k \\ &= -\sum_{k=0}^{n-1} \left(\frac{\left(\left(v^k\right)^T A\right) \left(A^{-1} b\right)}{\left(v^k\right)^T Av^k} \right) v^k \\ &= -A^{-1} b \end{aligned}$$

But $-A^{-1}b = u^*$. QED.

Following some extra remarks added later from An introduction to optimization by Ching

and Zak, 1996:

1. Conjugate direction algorithms solve quadratics of n variables in n steps
2. Algorithm does not require calculation of Hessian.
3. Algorithm requires no matrix inverse and no storage for the $A^{n\times n}$ matrix.
4. C.D. Algorithms perform better than steepest descent but not as well as Newton-Raphson (when close to optimal).
5. For quadratic $J(u) = \frac{1}{2}x^T Ax - x^T b$, the best direction at each step is the mutually conjugate direction w.r.t. A.

See Example 10.1, page 133 of the above text for illustrations how to determine each of v^i vectors for given A matrix.

1.14 Lecture 14. Thursday, March 3, 2016

1.14.1 Constraints and linear programming

We are about to enter new phase of the course with constraints and linear programming. Until now we used iterative methods to solve unconstrained problems. These are gradient based. Also looked at Newton-Raphson. We used steepest descent and conjugate directions. These methods are mainly applied to problem without constraints. i.e. u is free, where u are the variables. But in HW4 we had problem where u was the capacitance. This can not be negative. But we did not account for this. When we have constraints and want to use the above iterative methods, there are ad hoc methods to handle this, but we will not cover these ad-hoc methods in this course, but will mention some of them.

We can check that no constraint is violated during the search and start a new search. There are literature on what is called "projection methods" and other names.

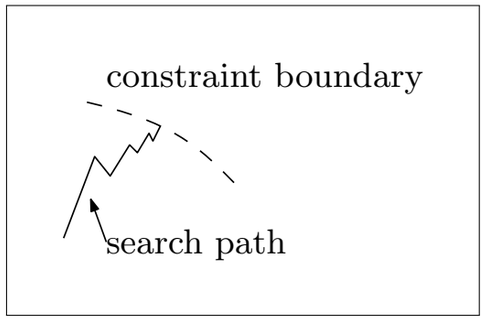


Figure 1.12: Search path near constraint

One good method is called the "penalty function method". This works as follows

$$\tilde{J}(u) = J(u) + J_{penalty}(u)$$

The original objective function is $J(u)$ and $J_{penalty}(u)$ is function we add such that it becomes very large when u constraint is violated ($u \notin U$). (assuming we are minimizing $J(u)$).

This method works on many problems. For example, if we do not want u_1 to be negative, we can add

$$J_{penalty}(u) = -100 \min(0, u_1)$$

This way, when $u_1 \le 0$, the result will be very large and positive. Hence $\tilde{J}(u)$ will become very large and the search will avoid this region and turn away during the line search.

But a theory with name of "Kohn Tucker" is the main way to handle such problems under heading of "non-linear programming".

Now we go back to linear programming which we will cover over the next 4–5 lectures. key points of linear programming are

1. Objective function is linear in u^i
2. Linear inequality constraints on u^i

1.14.2 History of linear programming

1. Dantzing introduced simplex algorithm. Matlab linprog implements this to find solution to L.P. problems.
2. Simplex algorithm has some problems related to what is called "klee-type pathologies". These days, L.P. have millions of variables. These days we want to solve many large scale L.P. The "Klee-type pathologies" says that there are some bad input to L.P. which causes it to become very slow. L.P. visits vertices of polytopes. We can do billion and more vertices these days on the PC with no problem. 10^{30} vertices is a small L.P. problem these days.

   L.P. works fast by not visiting each vertex. But with some input L.P. can become slow and force it to visit all vertices.
3. Khachian: 1970's. Front page of NY times. Introduced ellipsoidal algorithm to solve L.P. (Faster than L.P. on the worst case problems). But it turns out that in real world problems, simplex was still faster, unless the problem had "Klee-type pathologies".
4. Small. Gave a probabilistic explanation of the "magic of simplex algorithm". Considering average probabilities. In computer science, computational complexity is defined in terms of worst case.
5. Kharmarker, from Bell Labs. Came up with new approach. Developed scaling to L.P.

1.14.3 Polytopes

Polytopes are central to L.P since polytopes are described by constraints. See handout "Polytopes" taken from textbook Barmish, Robust Control.

Polytope is convex hull of finite point set. These are the generators $v^1, \cdots, v^N$. Polytopes have extreme vertices. L.P. visits vertices. If the set is bounded, we call it polytope, else we call it polyhedron. So with linear inequalities constraints and bounded, we have polytopes.

Reader A linear function $J(u) = a^T u$ on polytope P achieves its Max. or Min. at an extreme point. We showed that the Max. of convex function is at a vertex. we can also show that Min. of concave is at a vertex. Linear functions are both concave and convex. QED.

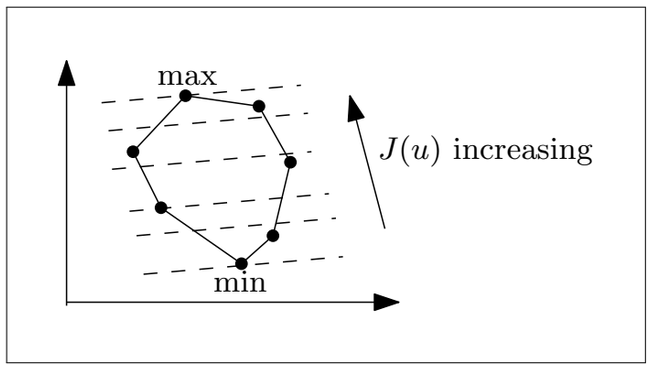


Figure 1.13: Increasing $J(u)$ diagram

Often we do not have list of vertices. Need to first generate them. They are generated from the constraint inequalities.

How many vertices to search? McMullen's Upper Bound Theorem gives us the answer. Assuming we have m constraints and n variables, where $m \geq n$. Then

$$V(m,n) = \begin{pmatrix} m - \left\lfloor \frac{1}{2}(n+1) \right\rfloor \\ m-n \end{pmatrix} + \begin{pmatrix} m - \left\lfloor \frac{1}{2}(n+2) \right\rfloor \\ m-n \end{pmatrix}$$

Example for $m = 20, n = 10$ we get 4004 vertices. So this is a small problem.

Reader Consider $n = 500, m = 2000$ which is very modest in terms of current technology, assuming it takes 1 microsecond per vertex, then it will take order of 10^{296} years to search all the vertices.

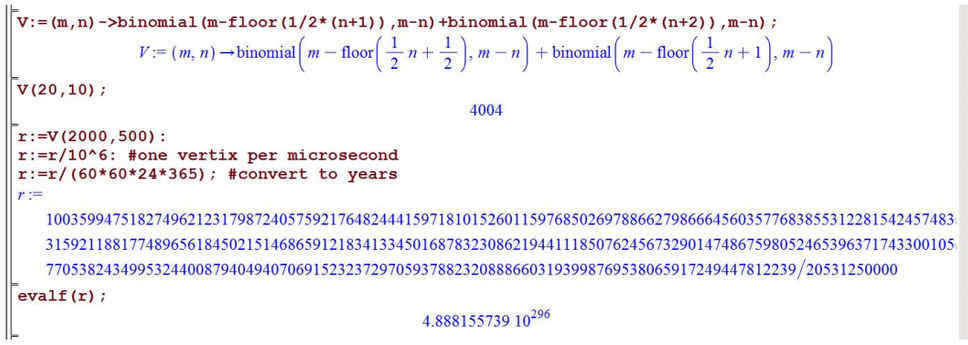


Figure 1.14: Verification of number of vertices using Maple

Up to last few years, L.P. was considered a completed research. But in the last 5–10 years, there is new L.P. research starting. Modern L.P. solvers use linear inequalities description as input. This is expressed and formulated as $Ax = b$. What if vertices or some vertices generating mechanism was given as input instead of the constraints themselves? How to convert the vertices to constraints? This is a difficult problem.

1.15 Lecture 15. Tuesday, March 8, 2016

1.15.1 Mechanism of linear programming

Today we will begin the mechanism of doing L.P. (linear programming). We know the extreme points is where the optimal will occur. But searching all extreme points is not practical for large N as shown before. One can take a whole course just on L.P. but here we will cover the main ideas. We basically have a linear objective function in u and linear constraints in u where u are the variables. This is called the raw L.P. formulation. This is converted to standard form L.P. and solved using the simplex method.

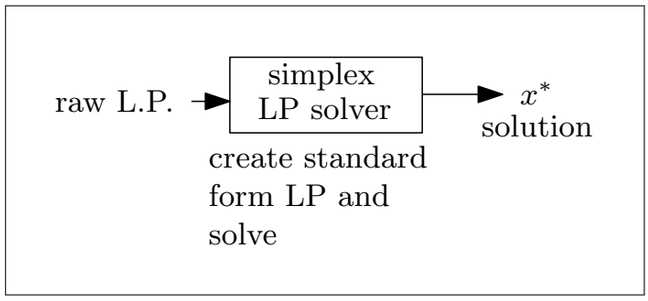


Figure 1.15: Simplex solver diagram

The solver finds the first vertex then in a clever way moves to another until it finds the optimal one. The solver solves two linear programming problems and these are:

1. First finds a feasible solution (basic)
2. moves from one vertex to another.

Standard form LP is

$$\boxed{\begin{array}{ll} \min & c^T x \\ st & Ax = b \end{array}}$$

Notice that solution might be infimum above. Example.

$$\begin{array}{ll} \min & x_1 \\ st & x_2 \le 5 \\ & x_1 < 0 \end{array}$$

The solution is $x_1 = -\infty$. This is closed by unbounded.

Ingredients of Linear programming are:

1. $n > m$ (number of variable is larger than number of constraints.). The matrix A is of order m by n. So A matrix is fat matrix and not thin.
2. No columns of A can all be zeros (non-degenerate).
3. Rank of A is m
4. $b \geq 0$
5. $x \geq 0$

What if we have some variables x_i which we want to be negative? We replace x_i with new variable $x_j = -x_i$. Now $x_j \geq 0$. Now in each place we have x_i which is negative and can't use, then we replace it with $-x_j$. This is now the same as before, but x_i is gone and replaced with $-x_j$ and x_j is now positive. So it is standard form.

At the end, when we obtain the solution, we replace x_j back to $-x_i$. (what about free variables?).

What if we have inequality in the raw L.P.? how to convert to equality for standard form? We use Slack variables and Surplus variables .

Example given $x_1 + 2x_2 - x_3 \leq 6$, then introduce new slack variable x_4 and rewrite the constraint as $x_1 + 2x_2 - x_3 + x_4 = 6$.

If we have constraint $x_1 + 2x_2 - x_3 \geq 6$, then we need surplus variable x_4. Rewrite as $x_1 + 2x_2 - x_3 - x_4 = 6$. Once we solve the LP problem and obtain x^*, we need to recover from this solution the actual variables of the raw LP (these are the u variables) and these do not contain any slack nor surplus variables.

1.15.2 Example, the sector patrol problem

See also handout sector patrol. The objective function is $E(T) = \frac{\frac{u_1}{3}}{10} + \frac{\frac{u_2}{3}}{5} = \frac{u_1}{30} + \frac{u_2}{15}$. Constraints are $u_1 \geq 0, u_2 \geq 0$ and

$$2u_1 + 2u_2 \geq 4$$
$$2u_1 + 2u_2 \leq 10$$

And as per handout, we need to add this constraint in order to obtain a realistic solution

$$u_2 \geq 1.5u_1$$

The above is the raw LP. Convert to standard form, using x as variables. It becomes

$$2x_1 + 2x_2 - x_4 = 4$$
$$2x_1 + 2x_2 + x_3 = 10$$
$$-1.5x_1 + x_2 - x_5 = 0$$

Where x_3, x_4, x_5 above were added to make it standard form. Writing it as $Ax = b$, the constraint equation is (we put the slack variables first by convention)

$$\begin{bmatrix} 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & 0 & -1 & 0 \\ -1.5 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 4 \\ 0 \end{bmatrix}$$

And $c^T x$ becomes (this is the objective function, notice we added the slack and surplus variables to it, but they are all zeros).

$$\begin{bmatrix} \frac{1}{30} & \frac{1}{15} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \end{bmatrix}$$

Reader find common sense solution working in u_1, u_2 domain. (will do this next lecture).

1.15.3 Basic and Feasible solutions

We will define two solutions: The basic solution, and basic feasible solution.

A vector x is said to be basic solution if it solves $Ax = b$ and the non-zeros elements of x correspond to the linearly independent columns of A.

Reader Is there a basic infeasible solution?

1.16 Lecture 16. Thursday, March 10, 2016

Recall, the standard LP problem is

$$\begin{aligned} \min \quad & c^T x \\ st \quad & Ax = b \end{aligned}$$

We talked about transforming the problem from raw LP to standard LP. The patrol sector problem, solved using common sense graphical approach is given below

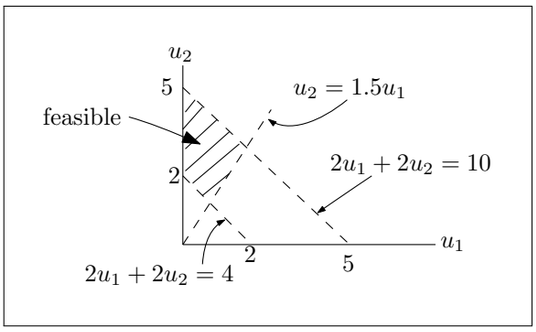


Figure 1.16: Patrol problem

The optimal u^* has to be at one of the vertices of the feasible region. It will be at the vertex shown

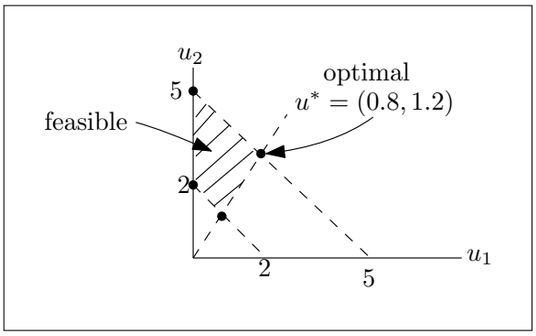


Figure 1.17: Patrol problem solution

Using Matlab, the above is solved as follows

$$\begin{bmatrix} 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & 0 & -1 & 0 \\ -1.5 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 4 \\ 0 \end{bmatrix}$$

And $c^T x$ becomes (this is the objective function, notice we add the slack and surplus variables to it, but they are all zeros).

$$\begin{bmatrix} \frac{1}{30} & \frac{1}{15} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \end{bmatrix}$$

The code is

```
f=[1/30,1/15,0,0,0];
A=[2,2,1,0,0,;
   2,2,0,-1,0,;
   -1.5,1,0,0,-1];
b=[10,4,0];
[X,FVAL,EXITFLAG,OUTPUT]=linprog(f,[],[],A,b,zeros(size(f)),[])
```

Result of above run

```
Optimization terminated.
X =
0.799999999994766
1.20000000008299
5.99999999984448
1.55520543353025e-10
9.08450336982967e-11
FVAL =
0.106666666672025
EXITFLAG =
1
OUTPUT =
iterations: 7
algorithm: 'interior-point-legacy'
cgiterations: 0
message: 'Optimization terminated.'
constrviolation: 8.88178419700125e-16
firstorderopt: 5.07058939341966e-12
```

In the above, we only need to map $x(1)$ to u_1 and $x(2)$ to u_2 to read the result. We see that Matlab result matches the graphical solution.

definition For LP, we say $\boldsymbol{x}$ is feasible if $\boldsymbol{x}$ satisfies the constraints.

For example, for the sector patrol, let U be the feasible set in $\Re^2$ (raw LP). However, in standard LP, the feasible set is in $\Re^5$.

Reader Obtain feasible set in $\Re^5$. Obtain feasible point with either x_3, x_4, x_5 nonzero.

Basic solution A vector x is basic solution if the non-zero components of x corresponds to the linearly independent columns of A. We do not require feasibility to be basic solution).

The difference in LP and standard $A\boldsymbol{x} = \boldsymbol{b}$ solution we have seen before many times in linear algebra, is that in LP, we want to solve $A\boldsymbol{x} = \boldsymbol{b}$ but with $\boldsymbol{x} \geq 0$ and at same time have $\boldsymbol{x}$ be optimal. This what makes LP different from standard methods of solving this problem.

The algorithm takes a solution which is feasible and makes it feasible basic solution. Then after that, we move from one basic feasible solution to another basic feasible solution while at the same time making $J(u)$ smaller until it reaches the optimal value.

Reader LP has at least one basic solution.

A has m linearly independent columns, since it has rank m.

$$A\boldsymbol{x} = \boldsymbol{b}$$

$$\begin{bmatrix} A_{basic} & A_{notbasic} \end{bmatrix} \begin{bmatrix} x_{basic} \\ x_{notbasic} \end{bmatrix} = \begin{bmatrix} b_{basic} \\ b_{notbasic} \end{bmatrix}$$

1.16.1 Linear programming feasible and basic solutions

Theorem Suppose LP has feasible solution, then it has a basic feasible solution. Remember: $\boldsymbol{x}$ is feasible if it is in the feasible region (satisfies constraints), and $\boldsymbol{x}$ is basic solution if the non-zero elements of $\boldsymbol{x}$ correspond to linearly independent columns of A.

Proof say $\boldsymbol{x}$ is feasible. Let $a^1, a^2, \cdots, a^p$ denote columns of A corresponding to nonzero entries of $\boldsymbol{x}$. Without loss of generality, say the first p columns of A. (we can always rearrange A to make it so). There are two cases:

case one The a^i above are linearly independent. We are done. x is therefore basic by definition.

case two The a^i are linearly dependent. Therefore there exist scalars y_i, not all zero, such that $\sum_{i=1}^{p} y_i a^i = 0$. (this is the definition of linearly dependent columns).

Now we do the squeezing process. For $\varepsilon > 0$ define vector η^ε with components

$$\eta^\varepsilon = \begin{matrix} x_i - \varepsilon y_i & \text{for } i \le p \\ 0 & i > p \end{matrix}$$

Reader η^ε is feasible for small ε. Hence

$$A\eta^\varepsilon = A\boldsymbol{x} - \varepsilon A \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Let $\varepsilon = \min\left\{\frac{x_i}{y_i}; y_i > 0\right\}$. Reader η^ε is basic and has at least one more zero entry than $\boldsymbol{x}$. So now $\boldsymbol{x}$ has $p-1$ columns of A corresponding to non-zero entries in $\boldsymbol{x}$. Continuing this process, we keep finding other basic feasible solutions.

example

$$\begin{bmatrix} 1 & 2 & 1 & 2 & 0 \\ 1 & 1 & 1 & 2 & 1 \\ 2 & -1 & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 4 \end{bmatrix}$$

Let starting $\boldsymbol{x} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ This is feasible since it satisfies $A\boldsymbol{x} = \boldsymbol{b}$ with $\boldsymbol{x} \ge 0$. But not basic, since the last 3 columns are not linearly independent. (the last three columns of A. Since these

are the ones that correspond to non-zero elements of x. we now write

$$y_3 a^3 + y_4 a^4 + y_5 a^5 = 0$$

Where a^3, a^4, a^5 represent the last three columns of A and y^i are the scalars we want to solve for. Solving, gives

$$\begin{aligned} y_3 &= 2 \\ y_4 &= -1 \\ y_5 &= 0 \end{aligned}$$

Hence, first find $\varepsilon = \min\left\{\frac{x_i}{y_i}; y > 0\right\}$, which we find to be $\varepsilon = \frac{1}{2}$. Now we find

$$\boldsymbol{x}^{new} = \eta^{\varepsilon} = \begin{bmatrix} x_1 - \varepsilon y_1 \\ x_2 - \varepsilon y_2 \\ x_3 - \varepsilon y_3 \\ x_4 - \varepsilon y_4 \\ x_5 - \varepsilon y_5 \end{bmatrix} = \begin{bmatrix} 0 - \frac{1}{2}(0) \\ 0 - \frac{1}{2}(0) \\ 1 - \frac{1}{2}(2) \\ 1 - \frac{1}{2}(-1) \\ 1 - \frac{1}{2}(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{3}{2} \\ 1 \end{bmatrix}$$

Since now a^4, a^5 are linearly independent (these are the fourth and fifth columns of A) and these correspond to the non zero of the new $x = \eta^{\varepsilon}$, then the new x is basic and feasible. So we have started with feasible solution, and from it obtained a basic feasible solution. This is not the final optimal solution x but we repeat this process now.

1.17 Lecture 17. Tuesday, March 15, 2016

If we have a feasible solution, we can obtain a basic feasible solution from it using the squeezing method. We have not talked about optimality yet. The next thing to consider is optimality. We will proof if we have an optimal feasible solution, then by obtaining a basic solution from it, the basic solution will remain optimal. This is called the optimality theorem. Then we will talk about extreme points.

1.17.1 Optimality theorem

If an optimal feasible solution exist, then an optimal feasible and basic solution exist as well.

Proof Suppose x is optimal and feasible. If x is basic, we are done. If not, now we need to do the squeeze process to make it basic, but now we have to do the squeeze making sure it remains optimal. Say $a^1, a^2, \cdots, a^p$ are the columns associated with non-zero entries in x. As before, we way, WLOG these are the first p columns in A. Hence there exist scalars y_i, not all zero, such that $\sum_{i=1}^{p} y_i a^i = 0$. Define η^{ε} for scalar ε such that

$$\eta^{\varepsilon} = \begin{matrix} x_i - \varepsilon y_i & \text{for } i \le p \\ 0 & i > p \end{matrix}$$

Reader For small ε, say $|\varepsilon| \le \delta$, then η^{ε} is still feasible. Claim: For ε suitable small, η^{ε} is optimal. It suffice to show that $c^T y = 0$ with $y_i = 0$ for $i > p$. This being the case, then $c^T \eta^{\varepsilon} = c^T x = J^*$. By contradiction: Say $c^T y \neq 0$. Let $\varepsilon = \delta sgn\left(c^T y\right)$. Let us show that η^{ε} is better than x. This contradicts optimality. Now

$$\begin{aligned} c^T \eta^{\varepsilon} &= c^T \left(x - \varepsilon y\right) \\ &= c^T x - c^T \left(\delta sgn\left(c^T y\right)\right) y \\ &= c^T x - \delta sgn\left(c^T y\right)\left(c^T y\right) \\ &= J^* - \delta \left|c^T y\right| \\ &< J^* \end{aligned}$$

This contradicts optimality. QED.

Now we will talk about extreme points. Extreme points and basic feasible solution are the same thing.

1.17.2 The extreme point theorem

Let $P = \{x \in \Re^n, x \geq 0, Ax \leq b\}$. This polyhedron is the feasible set. The set of extreme points of P are the basic feasible solution.

Proof See handout extreme send today. We need to get to the first feasible solution. This can be hard to obtain. Once we find a feasible solution, then we use it to find the first basic feasible solution, and from them we repeat the process (using the squeeze method). As we move from one basic feasible solution to another, we do this by making $J(u)$ smaller.

Example Consider LP with constraints

$$\begin{aligned} x_i &\geq 0, i = 1,2,3 \\ 2x_1 + 3x_2 &= 1 \\ x_1 + x_2 + x_3 &= 1 \end{aligned}$$

Note that $x_1 + x_2 + x_3 = 1$ is common in LP optimization. It is called unit simplex. Here is a plot of the above.

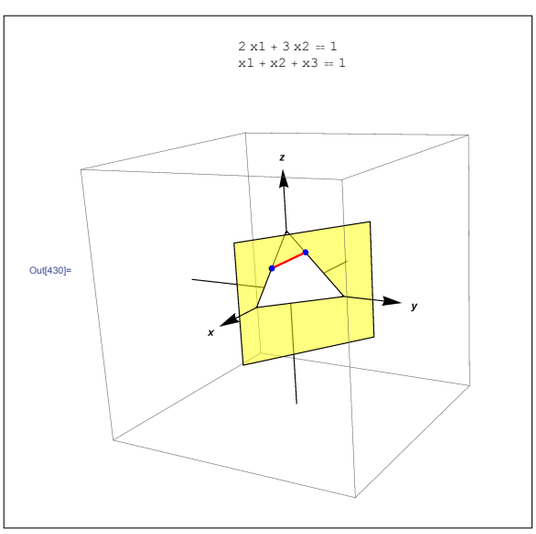


Figure 1.18: Unit simplex

Reader Plane for $2x_1 + 3x_2 = 1$ intersect the unit simplex else no feasible region exist. So there are two basic feasible solutions at $\left(0, \frac{1}{3}, \frac{2}{3}\right)$ and $\left(\frac{1}{2}, 0, \frac{1}{2}\right)$. These are the two red points shown in the above plot.

1.17.3 Mechanism the simplex method

We will use the sector patrol problem to show the mechanisms of simplex.

$$Ax = b$$

$$\begin{bmatrix} 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & 0 & -1 & 0 \\ -1.5 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 10 \\ 4 \\ 0 \end{bmatrix}$$

Step 1. Form partial tableau.

$$\begin{bmatrix} A & b \end{bmatrix} = \begin{bmatrix} 2 & 2 & 1 & 0 & 0 & 10 \\ 2 & 2 & 0 & -1 & 0 & 4 \\ -1.5 & 1 & 0 & 0 & -1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

We need to identify an identify matrix. By row operations we obtain

$$\begin{bmatrix} r_1 \\ -r_2 \\ -r_3 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 1 & 0 & 0 & 10 \\ -2 & -2 & 0 & 1 & 0 & -4 \\ 1.5 & -1 & 0 & 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} r_1' \\ r_2' \\ r_3' \end{bmatrix} \tag{1}$$

Now we can read out $\boldsymbol{x}^1$ and we see that $x_3 = 10, x_4 = -4, x_5 = 0$. These are the entries in x which corresponds to the columns of the unit matrix inside A. All other entries in x are assumed zero. Hence

$$\boldsymbol{x}^1 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 10 \\ -4 \\ 0 \end{bmatrix}$$

Note this is not feasible but basic. Now we go to next step. We have to remove an entry from x^1 and move in its place another entry. Let us pick x_3 to kick out and move in x_1. By row operations applied to (1) we obtain

$$\begin{bmatrix} \frac{r_1'}{2} \\ r_2' + 2r_1'' \\ r_3' - 1.5r_1'' \end{bmatrix} = \begin{bmatrix} 1 & 1 & \frac{1}{2} & 0 & 0 & 5 \\ 0 & 0 & 1 & 1 & 0 & 6 \\ 0 & -2.5 & -0.75 & 0 & 1 & -7.5 \end{bmatrix} \rightarrow \begin{bmatrix} r_1'' \\ r_2'' \\ r_3'' \end{bmatrix}$$

Hence the new identity matrix is a^1, a^4, a^5 and therefore $x_1 = 5, x_4 = 6, x_5 = -7.5$, These are the entries in x which corresponds to the columns of the unit matrix inside A. All other entries in x are assumed zero. Hence

$$\boldsymbol{x}^2 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 0 \\ 6 \\ -7.5 \end{bmatrix}$$

This is the second basic $\boldsymbol{x}$ we found.

Reader Find basic solution via pivoting row operations. Now we want to redo the above, with feasibility in mind which we did not consider above when moving elements out and selecting which one to move in.

Example

$$Ax = b$$

$$\begin{bmatrix} 1 & 0 & 0 & 4 & 3 & 2 \\ 0 & 1 & 0 & 2 & -1 & 2 \\ 0 & 0 & 1 & -1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} A & b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 4 & 3 & 2 & 1 \\ 0 & 1 & 0 & 2 & -1 & 2 & 2 \\ 0 & 0 & 1 & -1 & 2 & 1 & 3 \end{bmatrix}$$

Now we need to start with feasible solution. Last example we did not care about this, but now we need the first solution to be feasible. Here $x_1 = 1, x_2 = 2, x_3 = 3$ (read out, from the identity matrix, since the first three columns are linearly independent). We we use the squeeze process, to decide which one to kick out and which to move in. Let us for now choose arbitrarily x_4 (fourth column) to move in and we have to kick out a column. Need ε^* to decide.

$$\varepsilon^* = \min\left\{\frac{1}{4}, \frac{1}{2}\right\} = \frac{1}{4}$$

Hence it is the first column to kick out. Remember that when doing the above to determine which x to kick out, we only divide by those entries in the column which are positive. If there is a negative entry, then do not use it. That is why in the above, we did not write $\varepsilon^* = \min\left\{\frac{1}{4}, \frac{1}{2}, \frac{3}{-1}\right\}$. We will continue next lecture.

1.18 Lecture 18. Thursday, March 17, 2016

Class planning:

1. special problem towards end of course
2. Test 2 after spring break. Up to and including LP.
3. HW 6 will be send soon, due right after spring break.

Exercise to do

> A good exercise, to be done by hand, is the following: For the sector patrol problem which we considered in class, solve the Phase One LP to obtain a basic feasible solution.
> Then, use this first basic feasible solution as a starting point for the original LP and solve it via a sequence of tableau. Since you already know the answer, you will get feedback whether your calculations produce the right result.

1.18.1 Simplex method examples

We still need to know how to find first feasible solution. In the following example

x_1	x_2	x_3	x_4	x_5	x_6	b
1	0	0	2	4	6	4
0	1	0	1	2	3	3
0	0	1	–1	2	1	1

A solution which is basic and feasible is $x_1 = 4, x_2 = 3, x_3 = 1$ and all others $x_i = 0, i = 4 \cdots 6$.

$$x = \begin{pmatrix} 4 \\ 3 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

This was just read out from the identity matrix in the first 3 columns above. Let us now assume we want fourth column to be in the basis. We have to remove one of the current columns in the basis in order to let another column in. $\min\left\{\frac{4}{2}, \frac{3}{1}\right\} = 2$. This means the first row is the pivot row. Notice we do not consider $\frac{1}{-1}$ when doing the minimum operation. Any negative value in a column is bypassed. Now we know that first row is pivot row and that we want fourth column in. This is all what we need to go to the next step. We know need to normalize entry $(1,4)$ to one. (before it was 2). After normalizing the pivot row (by dividing the whole row by 2) we obtain

x_1	x_2	x_3	x_4	x_5	x_6	b
$\frac{1}{2}$	0	0	1	2	3	2
0	1	0	1	2	3	3
0	0	1	–1	2	1	1

Only now we start applying row operations, with row one as pivot row, we make all other entries in fourth column below $(1,4)$ zero, This gives the following

	x_1	x_2	x_3	x_4	x_5	x_6	b
r_1	$\frac{1}{2}$	0	0	1	2	3	2
$r_2 - r_1$	$-\frac{1}{2}$	1	0	0	0	0	1
$r_3 + r_1$	$\frac{1}{2}$	0	1	0	4	4	3

Now that we have a new identity matrix, we read out the new solution which is $x_2 = 1, x_4 = 2, x_3 = 3$ and all other x entries zero.

$$x = \begin{pmatrix} 0 \\ 1 \\ 3 \\ 2 \\ 0 \\ 0 \end{pmatrix}$$

We will now revisit this, with optimality in mind. This means we need to know which column to bring into the basis. The question is, which x_i to bring in. Set up this tableau[1]

$$\overbrace{\begin{pmatrix} 1 & \cdots & 0 & a(1,m+1) & \cdots & a(1,n) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \cdots \\ 0 & \cdots & 1 & a(m,m+1) & \cdots & a(m,n) \end{pmatrix}}^{A} \overbrace{\begin{pmatrix} y(1,0) \\ y(2,0) \\ \vdots \\ y(m,0) \end{pmatrix}}^{b}$$

Therefore, the current feasible and basic solution is $x_1 = y(1,0), x_2 = y(2,0), \cdots, x_m = y(m,0)$. All other $x_i = 0$. We need now to do a feasibility preserving perturbation.

Allow $x_{m+1}, x_{m+2}, \cdots, x_n$ to be part of the solution.

$$\begin{aligned} x_1 + \overbrace{\sum_{i=m+1}^{n} a(1,i)\, x_i}^{\text{we allow this in}} &= y(1,0) \\ x_2 + \sum_{i=m+1}^{n} a(2,i)\, x_i &= y(2,0) \\ &\vdots \\ x_m + \sum_{i=m+1}^{n} a(m,i)\, x_i &= y(2,0) \end{aligned}$$

Now $x = \begin{pmatrix} x_1 & x_2 & \cdots & x_m & 0 & \cdots & 0 \end{pmatrix}^T$ is still feasible, but no longer basic. We know that $J = c^T x$, hence

$$J = c_1 \left(y(1,0) - \sum_{i=m+1}^{n} a(1,i) x_i \right) + c_2 \left(y(2,0) - \sum_{i=m+1}^{n} a(2,i) x_i \right) + \cdots + c_m \left(y(m,0) - \sum_{i=m+1}^{n} a(m,i) x_i \right) + c_{m+1} x_{m+1} + \cdots +$$

Hence

$$J = \overbrace{\sum_{i=1}^{m} c_i y(i,0)}^{\text{current } J_0 \text{value}} - c_1 \sum_{i=1}^{m} a(1,i) x_i - c_2 \sum_{i=1}^{m} a(2,i) x_i - \cdots - c_m \sum_{i=1}^{m} a(m,i) x_i + c_{m+1} x_{m+1} + \cdots + c_n x_n$$

Therefore

$$J = J_0 - \left(-c_{m+1} + c_1 a(1,m+1) + c_2 a(2,m+1) + \cdots + c_m a(m,m+1) \right) x_{m+1} - \left(-c_{m+1} + c_1 a(2,m+1) + \cdots + c_m a(m,\right.$$

Define cost coefficients, for $j = m+1, \cdots, n$

$$r_j = c_j - \sum_{i=1}^{m} c_i a(i,j)$$

Hence

$$J = J_0 + \sum_{j=m+1}^{n} r_j x_j$$

[1] in class, we used $y(1, m+1)$ etc.. for entries in A matrix. I changed it in these notes to $a(1, m+1)$ etc... not to confuse myself with the RHS $y(1,0)$, etc.. entries in the b vector.

To minimize J we need to pick the most negative r_j. This tells us which x_j we need to bring into the basis in order to reduce J. Now we add extra row (last row) which is $J(u)$ to the table, to keep cost in it. Here is the table with the cost coefficient row added

$$\overbrace{\begin{pmatrix} 1 & \cdots & 0 & a(1,m+1) & \cdots & a(1,n) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \cdots \\ 0 & \cdots & 1 & a(m,m+1) & \cdots & a(m,n) \\ 0 & 0 & 0 & r_{m+1} & \cdots & r_n \end{pmatrix}}^{A} \overbrace{\begin{pmatrix} y(1,0) \\ y(2,0) \\ \vdots \\ -J_0 \end{pmatrix}}^{b}$$

state street vendor example Selling rings and bracelets. Ring has 3 oz. gold., 1 oz. silver. Bracelet has 1 oz. gold, 2 oz. silver. Profit on ring is \$4 and profit on bracelet is \$5. Initially we have 8 oz. gold and 9 0z silver. How many rings and bracelets to produce to maximize profit? This is the heart of many production problems. Note: we need integer LP to solve this. But if the quantities are very large, it will make little difference. So for now we can ignore the issue of integer programming.

$$\begin{aligned} u_1 &= \text{number of rings} \\ u_2 &= \text{number of bracelets} \\ J(u) &= 4u_1 + 5u_2 \end{aligned}$$

Since we want to maximize this, then

$$J(u) = -4u_1 - 5u_2$$

With $u_i \geq 0$. Constraints are $3u_1 + u_2 \leq 8$ and $u_1 + 2u_2 \leq 9$. We convert to standard LP with slack variables and make the first table

	x_1	x_2	x_3	x_4	b
row 1	3	2	1	0	8
row 2	1	2	0	1	9
$J(u)$	−4	−5	0	0	0

The first basic feasible solution is $x_3 = 8, x_4 = 9$. To decide which column to bring in, we see the most negative is column 2 (last row is −5). To find the pivot row, we see it is first row since $\min\left\{\frac{8}{2}, \frac{9}{2}\right\} = \{4, 4.5\} = 4$. Now we do the first stage.

Reader obtain the following

	x_1	x_2	x_3	x_4	b
row 1	2.5	0	1	−0.5	3.5
row 2	0.5	1	0	0.5	4.5
$J(u)$	−1.5	0	0	2.5	22.5

Hence $x_2 = 4.5, x_3 = 3.5$. Since there is still a negative entry in the last row we need to repeat the process again. We Keep doing this until there are no negative entries in the last (third) row.

Now we will now talk about how to find the first basic feasible solution. There are two cases. If the number of slack variables is m then first basic feasible solution can be read out. This means there is no phase one LP. Case two. The number of slack variables is $z < m$. Now we need to solve the first phase LP. Then use its result to solve second phase LP. In phase one, we introduce new artificial variables y_i as many as $m - z$ and new artificial cost function $J(y)$ which we want to minimize to zero.

See HW6, first problem for an example of how to solve first phase LP.

1.19 Lecture 19. Tuesday, March 22, 2016 (No class)

No class.

1.20 Lecture 20. Thursday, March 24, 2016 (No class)

No class.

1.21 Lecture 21. Tuesday, March 29, 2016

Coming up soon is the special problem. It is like one HW but can count up to two HW's weight. Note: Possible rescheduling for April 6, 2016 remain in place.

1.21.1 Second exam keywords

1. Test two will be more application oriented.
2. local minimum. Strong and weak. We had sufficient conditions. Gradient, Hessian.
3. If we have convexity, we can do much better. If not, we need iterative algorithms. Line search is central to iterative search. Step size, optimal step size.
4. we looked at steepest descent with and without optimal step size. For simple problems we can easily find optimal step size.
5. We talked about convergence for iterative algorithms. For steepest descent we talked about $E\theta^k$.
6. We talked about generalized Newton-Raphson. We talked about quadratic convergence. Conjugate direction method has quadratic convergence (we proved this). It will converge in N steps or less, where N is A matrix size. Newton-Raphson will converge in one step for quadratic function. We also talked about superlinear convergence.
7. This brought us to the end of iterative algorithms. Then we went to Linear programming. The number of vertices is large. So trying to check them all is not possible. Polytope is central to LP. The basic theory of LP
   (a) Feasible solutions
   (b) Basic solutions
   (c) basic and feasible solutions
   (d) The method of squashing using η^ϵ
   (e) Basic feasible $\Leftrightarrow$ extreme points.
   (f) Relative cost coefficients.
8. Then we talked about simplex algorithm.

Now we will start on today lecture. Often a problem is given to you, but it is not an LP problem. Sometimes it is not obvious how to convert it to an LP problem. Sometime we need algebra or cleaner reformulation of the problem to make it an LP problem. For example, in HW6, we had a min-max problem but it is was possible to convert it to an LP problem. See key solution for HW6.

1.21.2 Application of Linear programming to control problems

Another application area for LP is control. Example is the minimum fuel problem. In this, we want to go from some state to final state with minimum control effort. The control effort is generic name which can mean many things depending on the problem itself. We also want to do this in minimal time. We begin with the discrete state equation

$$x(k+1) = Ax(k) + Bu(k)$$

With $x(0)$ given as the initial state. In the above, x is an $n \times 1$ vector, and A is $n \times n$ and B is $n \times m$ where m is number of inputs, and u is $m \times 1$ input vector.

We want to select $u(k)$ sending $x(0)$ to some target x^* at some future time $k = N$. With N being minimal, and control effort minimum. Assume for now that u is scalar, which means one input, then an energy measure is

$$\sum_{k=0}^{N-1} |u(k)|^2$$

On the other hand, a peak measure is

$$\max\{u(k), k = 0, \dots, N-1\}$$

But we will consider the fuel measure given by

$$\sum_{k=0}^{N-1} |u(k)|$$

We will use fuel measure in the LP problem. The constraint is

$$|u(k)| \le U \tag{$*$}$$

Which says that control is bounded. Note that is (A, B) is controllable, we can get from initial state to final state in one step if we want, but the effort will be very large. We also want $x(N) = x^*$. The above two are the constraints in this problem. The objective function is

$$J(u) = \sum_{k=0}^{N-1} |u(k)|$$

Therefore

$$\begin{aligned}
x(1) &= Ax(0) + Bu(0)\\
x(2) &= A^2x(0) + ABu(0) + Bu(1)\\
x(3) &= A^3x(0) + A^2Bu(0) + ABu(1) + Bu(2)\\
&\vdots\\
x(N) &= \underbrace{A^Nx(0) + \sum_{k=0}^{N-1} A^{N-1-k}Bu(k)}_{x^* \text{This is the linear constraint}}
\end{aligned}$$

Now we rewrite the constraint $|u(k)| \le U$ as

$$u(k) = u_p(k) - u_n(k)$$

with $u_p(k), u_n(k)$ being positive. The objective function becomes (where we now put N as parameter, to say this is for a specific value of N

$$\begin{aligned}
J_N(u) &= \sum_{k=0}^{N-1} |u(k)|\\
&= \sum_{k=0}^{N-1} u_p(k) + u_n(k)
\end{aligned}$$

Equation $*$ above becomes

$$\begin{aligned}
u_p(k) &\le U\\
u_n(k) &\le U
\end{aligned}$$

So minimizing $J_N(u)$ is now an LP problem in $2N$ raw variables (we still need to add the needed slack variables). So by doubling the number of variables, we were able to convert this control problem to an LP problem. Let

$$N^* = \inf\{N : LP_N \text{ feasible}\}$$

Reader Argue that l^∞ measure also lead to an LP problem. l^∞ measure is $\max\{u(0), u(1), \dots\}$.

Example Let $A = \begin{pmatrix} 1 & 1 - e^{-T} \\ 0 & e^{-T} \end{pmatrix}$ where $T = 1$ is the sample time. And let $B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. The bound on $u(k) = 1$. This means $U = 1$. Let $x(0) = \begin{pmatrix} -40.91 \\ 43.50 \end{pmatrix}$ and let the target $x^* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Find u^* and N^*. Running LP for different values of N from $1, \dots, 4$ we find the first feasible solution at

$N = 4$. These is the resulting optimal effort $u(k)$

$$u(0) = -0.3009$$
$$u(1) = -1$$
$$u(2) = -0.2999$$
$$u(3) = -1$$

And the corresponding optimal objective function $J^* = 2.6008$. In the above, we have priorities on minimal time first.

1.21.3 Starting dynamic programming

Now we will start on the next topic, which is dynamic programming. This involves discrete decisions. In this course we will cover only discrete dynamic programming and not continuous dynamic programming. We will be making sequential decisions in time. For example, if $u(k) = \{-1, 0, 1\}$ then the decision tree will look like

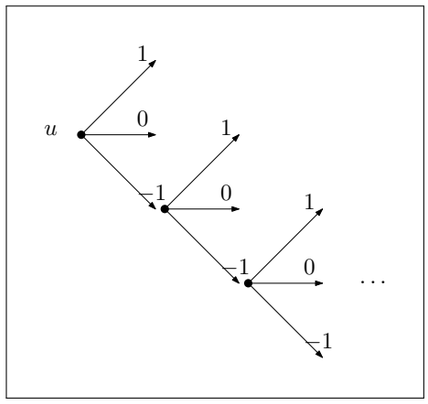


Figure 1.19: decision tree

We will get tree with potentially large number of branches. Combinatories arise. We get the curse of dimensionality problem again. For dynamic programming, Bellman is considered the person who originated the subject.

1.22 Lecture 22. Thursday, March 31, 2016. Second midterm exam

Exam.

1.23 Lecture 23. Tuesday, April 5, 2016

For steepest descent problem, with optimal step size, max is 1, need to use $\|u^{k+1} - u^k\| \leq 1$.

Next we will have special problem. Expect it next week.

Back to dynamic programming. Bellman secret is simple but theory is powerful. Main things to take from this course are

1. Linear programming.
2. Iterative solutions to optimization problems
3. dynamic programming

1.23.1 First dynamic programming problem, trip from NY to SFO

Suppose we want to take trip from NY to San Francisco, such that the total toll is minimum. We also must go west at each step we take once we start from NY. Not allowed to go east direction, even if the cost might be lower.

This diagram shows the possible routes and the cost (toll) for each segment.

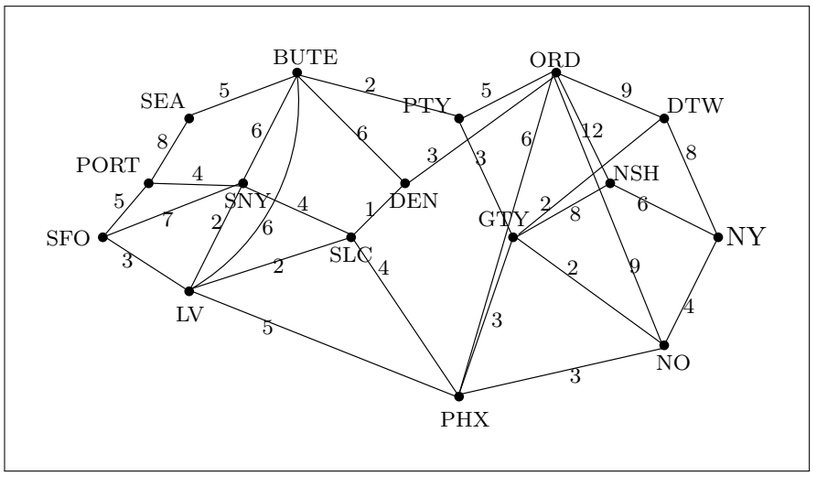


Figure 1.20: NY to SF tree one

Dynamic programming is now used to find optimal route in the above problem. (i.e. the route with least toll (cost) from NY to SFO). Dynamic programming is based on what if decisions. Instead of starting from NY and trying every possible route, we instead start backwards, and ask, what if we were in PORT, which route would we take?

Clearly the only route PORT to SFO with cost of 5 exist. Then we ask, what if we were in LV, which route would we take? we see it is LV to SFO with cost of 3. Then we ask what if we were in SNY? Then since LV had cost 3, then the route SNY $\rightarrow$ LV $\rightarrow$ SFO would be the one to take, with cost of $2+3=5$. Each time we find the cost from one city to SFO, we label the city with this cost. We keep moving back to the east, doing the same. When we arrive all the way to JFK, then we see that the lowest cost is

$$J^* = \text{JFK} \rightarrow \text{NO} \rightarrow \text{PHX} \rightarrow \text{LV} \rightarrow \text{SFO}$$

The following diagram shows the route above, with the cost of moving from each city to SFO given next to each city name on the diagram.

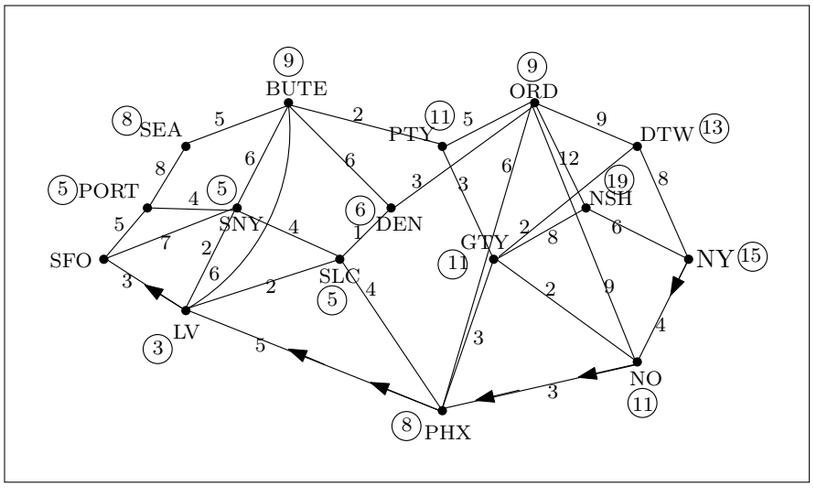


Figure 1.21: NY to SF tree two

If we were to solve the above problem using direct evaluation the number of computations is of order (assuming even n is $\frac{(n-1)!n!}{\frac{n}{2}!\frac{n}{2}!}$ while with dynamic programming method as explained above, it is $\frac{n^2}{2}+n$. For 20 cities, this given 220 for dynamic programming compare to over one million computation for the direct approach (trying all the possible routes).

We will only consider discrete dynamic programming. This is an optimization problem. The variables are not continuous. The variable take in discrete fixed values of choices each time. These applications are useful for integrate programming problems. An integer programming problem is much harder than continuous ones with much larger complexity.

When we are given an integer programming problem which is hard, we can try to approximate it to continuous programming problem and solve it more easily that way. For example

$$\min_x c^T x \quad \text{subject to} \quad Ax = b$$

where x is allowed to be integers, is a hard problem. But if we relax it and allow x to take any value so that the problem becomes continuous, then it will become much easier to solve using Linear Programming.

There are papers written on when we can approximate integer programming problems as continuous.

We will be making sequential decisions. $u_0, u_1, \dots, u_{N-1}$ i.e. N decisions where $u \in \mathbb{R}^n$. If we are given a problem which is not sequential, we can treat it as one for this purpose. For example, the car toll problem above, we formulate it to sequential but we did not have to do this. But the final answer J^* should come out the same no matter how it was formulated of course.

There will be states $x(k)$, $k = 0, 1, \dots, N$ where $x(N)$ is the terminal state. The constraints are state dependent. Because it depends on where we are when making the decision. For the car toll problem above, the decision depended on which city we were in. We denote the decision as $u(k) \in \Omega$ and the state equation is

$$x(k+1) = f(x(k), u(k), k)$$

and the objective function is

$$J(u) = \underbrace{\sum_{k=0}^{N-1} J(x(k), u(k), k)}_{\text{stage cost}} + \underbrace{\psi(x(N))}_{\text{terminal cost}}$$

The terminal cost, is a cost applied once we reach the terminal state.

1.23.2 Subproblem in dynamic programming

Now we need to define a subproblem. Notion of subproblem: Will begin at $k = L$ and will be in state $x(L)$. The cost when in state L is therefore

$$\boxed{J_L(u) = \sum_{k=L}^{N-1} J(x(k), u(k), k) + \psi(x(N))}$$

Suppose u^* is the optimal decision when we are at state $x(k)$. Let x^*k be the optimal trajectory from $x(k)$. i.e. $x^*(k)$ is corresponding state path beginning at given $x(0)$. Hence

$$x^*(k+1) = f(x^*(k), u^*(k), k), \ k = 0, 1, \dots, N-1$$

1.23.3 Bellman principle of optimality

If the subproblem begins at $x(L) = x^*(L)$ i.e. we being subproblem along optimal trajectory of the original problem, then $u^*(L), u^*(L+1), \dots, u^*(N-1)$ is optimal for the subproblem.

What all this means, is that if the subproblem is optimal, then its trajectory has to be part of the overall problem optimal trajectory. An optimal subproblem, can not become sub-optimal when viewed as part of the main problem.

1.24 Lecture 24. Thursday, April 7, 2016 (No class)

No class.

1.25 Lecture 25. Tuesday, April 12, 2016

The first part of the lecture was on describing the special problem we have to do. This is described in the special problem HW itself included in HW chapters, under special

problem.

1.25.1 Dynamic programming state equation

Now we go back to dynamic programming. The state equation is

$$x(k+1) = f(x(k), u(k), k) \qquad k = 0, 1, \cdots N \quad \text{and} \quad u(k) \in \Omega_k$$

And the objective function is

$$J = \Psi(x(N)) + \sum_{k=0}^{N-1} J_k(x(k), u(k))$$

Where $\Psi(x(N))$ is the cost of the terminal stage.

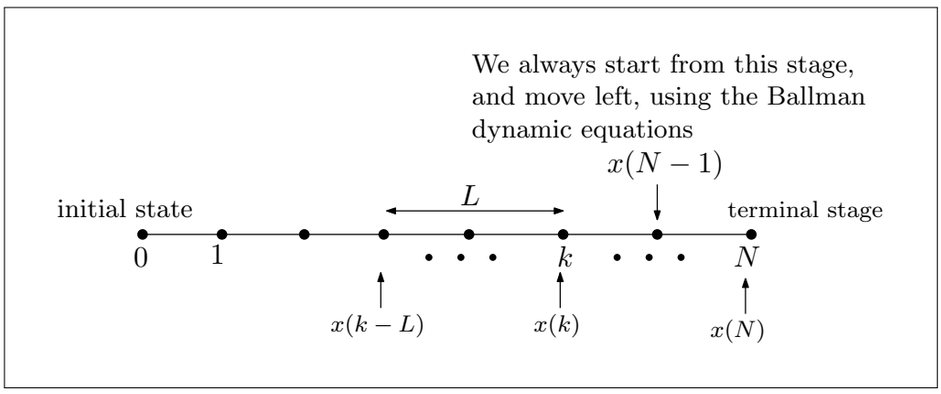


Figure 1.22: Showing dynamic programming block diagram

1.25.2 Subproblems and principle of optimality (POO)

A subproblem is defined at intermediate point.

P.O.O. (principle Of optimality): if initial state of a subproblem is on the optimal trajectory of original problem, then the subproblem is also optimal.

Proof by contradiction Let

$$u(k) = \begin{cases} u^*(k) & k = 0, \cdots, L-1 \\ u^*_{new}(k) & k = L, \cdots N-1 \end{cases}$$

Plug the above in the original problem. We will get a suboptimal solution.

Translating POO. to dynamic programming.

$$I(x(N-1), 1) = \min_{u(N-1) \in \Omega_{N-1}} \left\{ \overbrace{J(x(N-1), u(N-1))}^{\text{cost of one step}} + \Psi(x(N)) \right\}$$

The above is the optimal cost with one step to terminal stage. This is similar to what we did for the routing problem from NY to San Francisco before. The above is when we are standing in Portland and looking for the last step to take to San Francisco.

To minimize the above, we have to express everything in the same state $x(N-1)$. So we write the above, using the state equation, as

$$I(x(N-1), 1) = \min_{u(N-1) \in \Omega_{N-1}} \left\{ J(x(N-1), u(N-1)) + \Psi\left(f(x(N-1), u(N-1))\right) \right\} \tag{1}$$

Now we find $u^*(N-1)$ of the above, using

$$\frac{dI(x(N-1), 1)}{du(N-1)} = \frac{d}{du(N-1)} J(x(N-1), u(N-1)) + \Psi\left(f(x(N-1), u(N-1))\right) = 0$$

And solve for $u(N-1)$ using standard calculus. Once we find $u^*(N-1)$, we plug it back into (1) and obtain

$$I^*(x(N-1),1) = J(x(N-1),u^*(N-1)) + \Psi\left(f(x(N-1),u^*(N-1))\right)$$

Notice now there is no $\min_{u(N-1)\in\Omega_{N-1}}$ since we already done this. We now apply POO. again to find cost from stage $N-2$

$$I(x(N-2),2) = \min_{u(N-2)\in\Omega_{N-2}} \{J(x(N-2),u(N-2)) + I^*(x(N-1),1)\}$$

Notice the difference now. For all stages back, beyond $N-1$, we use the cost found from the ahead stage, which is $I^*(x(N-1),1)$ in the above case. We now repeat the process, and find optimal $u^*(x(N-2),2)$. See HW 7, problem 2 for detailed example how to do this. More generally,

$$I(x(L),N-L) = \min_{u(L)\in\Omega_L} \{J(x(L),u(L)) + I^*(x(L+1),N-L-1)\}$$

The above is called the dynamic programming equation.

1.26 Lecture 26. Thursday, April 14, 2016

1.26.1 Stages in dynamic programming

At stage L the optimal cost from stage L with $N-L$ steps to go is

$$\boxed{I(x_L,N-L) = \min_{u(N-L)\in\Omega_L} \{J(x_L,u(L)) + I(x_{L+1},N-(L+1))\}}$$

With appropriate initialization.

Some comments: The trickiest part is how to use this equation. Must be careful. Think of $u(L)$ as feedback. We call the optimal $u^*(L)$.

Warning. There is a constraint on u. Do not use derivative to find optimal without being careful about the limits and constraints. For example, if $|u(L)| \le 1$ and we have quadratic form. We will now use an example to show how to use these equations. Let

$$x_{k+1} = x_k - u_k \tag{1}$$

Where u_k is free to take any value. Let the objective function be

$$J(x_k,u_k) = \sum_{k=0}^{N-1} (x_{k+1}-u_k)^2 + u_k^2 \tag{2}$$

Reflecting a simple tracking mechanism. We always start at $x(N-1)$ with one stage to go. Hence the optimal cost from x_{N-1} with one step to go is

$$I(x_{N-1},1) = \min_{u(N-1)\in\Omega_1} \{J(x_{N-1},u_{N-1}) + \Psi(x_N)\}$$

$\Psi(x_N)$ is the terminal cost. Let us now remove it from the rest of the computation to simplify things. We also replace J in the above from (2) and obtain

$$\begin{aligned} I(x_{N-1},1) &= \min_{u(N-1)} \left\{ \left(x_{((N-1)+1)} - u_{N-1}\right)^2 + u_{N-1}^2 \right\} \\ &= \min_{u(N-1)} \left\{ (x_N - u_{N-1})^2 + u_{N-1}^2 \right\} \\ &= \min_{u(N-1)} \left\{ (x_N - u_{N-1})^2 + u_{N-1}^2 \right\} \end{aligned}$$

We want everything in terms of x_{N-1}. So we use (1) to write $x_N = x_{N-1} - u_{N-1}$ and plug it back in the last equation above to obtain

$$\begin{aligned} I(x_{N-1},1) &= \min_{u_{N-1}} \left\{ (x_{N-1} - u_{N-1} - u_{N-1})^2 + u_{N-1}^2 \right\} \\ &= \min_{u_{N-1}} \left\{ (x_{N-1} - 2u_{N-1})^2 + u_{N-1}^2 \right\} \end{aligned} \tag{3}$$

Only now to take derivative, in order to find u^*_{N-1}. therefore

$$\begin{aligned}\frac{dI(x_{N-1},1)}{u_{N-1}} &= 0\\ 2(x_{N-1}-2u_{N-1})(-2)+2u_{N-1} &= 0\\ u^*_{N-1} &= \frac{2}{5}x_{N-1}\end{aligned}$$

Now that we found the optimal u^*_{N-1} we go back to (3) and replace u_{N-1} in (3) by u^*_{N-1}. Hence

$$\begin{aligned}I(x_{N-1},1) &= \left(x_{N-1}-2\left(\frac{2}{5}x_{N-1}\right)\right)^2+\left(\frac{2}{5}x_{N-1}\right)^2\\ &= \frac{1}{5}x^2_{N-1}\end{aligned}$$

Now we backup one step. We need to find

$$I(x_{N-2},2)=\min_{u_{N-2}}\{J(x_{N-2},u_{N-1})+I(x_{N-1},1)\} \tag{4}$$

Notice that we used $I(x_{N-1},1)$ in place of what we had before, which was the terminal cost $\Psi(x(N))$. Since now we are two steps behind. All the work before was for finding optimal $I(x_{N-1},1)$. So now (4) becomes

$$I(x_{N-2},2)=\min_{u_{N-2}}\left\{J(x_{N-1},u_{N-1})+\frac{1}{5}x^2_{N-1}\right\} \tag{5}$$

But from (2)

$$J(x_{N-1},u_{N-2})=(x_{N-1}-u_{N-2})^2+u^2_{N-2}$$

Hence (5) becomes

$$I(x_{N-2},2)=\min_{u_{N-2}}\left\{(x_{N-1}-u_{N-2})^2+u^2_{N-2}+\frac{1}{5}x^2_{N-1}\right\}$$

We need to use the state equation $x_{k+1}=x_k-u_k$ to rewrite x_{N-1} in the above, since we want everything in $N-2$ terms. Therefore the above becomes

$$\begin{aligned}I(x_{N-2},2) &= \min_{u_{N-2}}\left\{((x_{N-2}-u_{N-2})-u_{N-2})^2+u^2_{N-2}+\frac{1}{5}(x_{N-2}-u_{N-2})^2\right\}\\ &= \min_{u_{N-2}}\left\{\frac{26}{5}u^2_{N-2}-\frac{22}{5}u_{N-2}x_{N-2}+\frac{6}{5}x^2_{N-2}\right\}\end{aligned} \tag{6}$$

Now we take derivative, to find u^*_{N-2}

$$\begin{aligned}\frac{dI(x_{N-2},2)}{u_{N-2}} &= 0\\ \frac{52}{5}u_{N-2}-\frac{22}{5}x_{N-2} &= 0\\ u^*_{N-2} &= \frac{11}{26}x_{N-2}\end{aligned}$$

Now that we found the optimal u^*_{N-2}, we go back to (6) and replace u_{N-2} there with u^*_{N-2}

$$\begin{aligned}I(x_{N-2},2) &= \frac{26}{5}\left(\frac{11}{26}x_{N-2}\right)^2-\frac{22}{5}\left(\frac{11}{26}x_{N-2}\right)x_{N-2}+\frac{6}{5}x^2_{N-2}\\ &= \frac{7}{26}x^2_{N-2}\end{aligned}$$

Reader Carry out one more stage and obtain $J^*=(x(0),3)$

Answer

$$\begin{aligned}I(x_{N-3},3) &= \min_{u_{N-3}}\{J(x_{N-3},u_{N-3})+I(x_{N-2},2)\}\\ &= \min_{u_{N-3}}\left\{J(x_{N-3},u_{N-3})+\frac{7}{26}x^2_{N-2}\right\}\end{aligned}$$

But from (2) $J(x_{N-3},u_{N-3})=(x_{N-2}-u_{N-3})^2+u^2_{N-3}$, hence the above becomes

$$I(x_{N-3},3)=\min_{u_{N-3}}\left\{(x_{N-2}-u_{N-3})^2+u^2_{N-3}+\frac{7}{26}x^2_{N-2}\right\}$$

We need to use the state equation $x_{k+1}=x_k-u_k$ to rewrite x_{N-2} in the above, since we want

everything in $N-3$ terms. Therefore the above becomes

$$\begin{aligned} I(x_{N-3},3) &= \min_{u_{N-3}} \left\{ ((x_{N-3}-u_{N-3})-u_{N-3})^2 + u_{N-3}^2 + \frac{7}{26}(x_{N-3}-u_{N-3})^2 \right\} \\ &= \min_{u_{N-3}} \left\{ \frac{137}{26}u_{N-3}^2 - \frac{59}{13}u_{N-3}x_{N-3} + \frac{33}{26}x_{N-3}^2 \right\} \end{aligned} \tag{7}$$

Now we take derivative, to find u_{N-3}^*

$$\begin{aligned} \frac{dI(x_{N-3},3)}{u_{N-3}} &= 0 \\ \frac{274}{26}u_{N-3} - \frac{59}{13}x_{N-3} &= 0 \\ u_{N-3}^* &= \frac{59}{137}x_{N-3} \end{aligned}$$

Replace this back in (7)

$$\begin{aligned} I(x_{N-3},3) &= \frac{137}{26}\left(\frac{59}{137}x_{N-3}\right)^2 - \frac{59}{13}\left(\frac{59}{137}x_{N-3}\right)x_{N-3} + \frac{33}{26}x_{N-3}^2 \\ &= \frac{40}{137}x_{N-3}^2 \end{aligned}$$

Let us do one more one, $N=4$.

$$\begin{aligned} I(x_{N-4},4) &= \min_{u_{N-4}} \{ J(x_{N-4},u_{N-4}) + I(x_{N-3},3) \} \\ &= \min_{u_{N-4}} \left\{ J(x_{N-4},u_{N-4}) + \frac{40}{137}x_{N-3}^2 \right\} \end{aligned}$$

But from (2) $J(x_{N-4},u_{N-4}) = (x_{N-3}-u_{N-4})^2 + u_{N-4}^2$, hence the above becomes

$$I(x_{N-4},4) = \min_{u_{N-4}} \left\{ (x_{N-3}-u_{N-4})^2 + u_{N-4}^2 + \frac{40}{137}x_{N-3}^2 \right\}$$

We need to use the state equation $x_{k+1} = x_k - u_k$ to rewrite x_{N-3} in the above, since we want everything in $N-4$ terms. Therefore the above becomes

$$\begin{aligned} I(x_{N-4},4) &= \min_{u_{N-4}} \left\{ ((x_{N-4}-u_{N-4})-u_{N-4})^2 + u_{N-4}^2 + \frac{40}{137}(x_{N-4}-u_{N-4})^2 \right\} \\ &= \min_{u_{N-4}} \left\{ \frac{725}{137}u_{N-4}^2 - \frac{628}{137}u_{N-4}x_{N-4} + \frac{177}{137}x_{N-4}^2 \right\} \end{aligned} \tag{8}$$

Now we take derivative, to find u_{N-4}^*

$$\begin{aligned} \frac{dI(x_{N-4},4)}{u_{N-4}} &= 0 \\ \frac{1450}{137}u_{N-4} - \frac{628}{137}x_{N-4} &= 0 \\ u_{N-4}^* &= \frac{314}{725}x_{N-4} \end{aligned}$$

Therefore (8) becomes

$$\begin{aligned} I(x_{N-4},4) &= \frac{725}{137}\left(\frac{314}{725}x_{N-4}\right)^2 - \frac{628}{137}\left(\frac{314}{725}x_{N-4}\right)x_{N-4} + \frac{177}{137}x_{N-4}^2 \\ &= \frac{217}{725}x_{N-4}^2 \end{aligned}$$

A table of the summary

L	$I(x_{N-L},L)$
1	$0.2\ x_{N-1}^2$
2	$0.2692\ x_{N-2}^2$
3	$0.29197\ x_{N-3}^2$
4	$0.29931\ x_{N-4}^2$

So for $N=4$

L	$I(x_{4-L}, L)$
1	$0.2\ x_3^2$
2	$0.2692\ x_2^2$
3	$0.29197\ x_1^2$
4	$0.29931\ x_0^2$

Using $x_{k+1} = x_k - u_k$ to write everything in terms of x_0

L	$I(x_{4-L}, L)$
1	$0.2\ (((x_0 - u_0) - u_1) - u_2)^2$
2	$0.2692\ ((x_0 - u_0) - u_1)^2$
3	$0.29197\ x_1^2\ (x_0 - u_0)^2$
4	$0.29931\ x_0^2$

So total cost is

$$I = 0.2\ (((x_0 - u_0) - u_1) - u_2)^2 + 0.2692\ ((x_0 - u_0) - u_1)^2 + 0.29197\ x_1^2 (x_0 - u_0)^2 + 0.29931\ x_0^2$$

This example is special case of LQR. What happens in this example above, or more generally when $N \to \infty$? As $N \to \infty$ we will see later that the feedback gains become time invariant. This is called steady state LQR and will we arrive at the Riccati equations.

1.26.2 Allocation problem, applying DP to investment problem

Next example is allocation problem We will do it in two steps. We can solve this without using D.P. but will use D.P. to illustrate the method. Consider two investments.

1. Invest \$1 in real estate, with return of \$2.
2. Invest \$1 in oil, with return of \$4.

Let say with start with fixed amount of money k dollars. We have constraint: b_r is maximum allowed amount of investment in real estate, b_o is the maximum amount allowed for oil investment. To avoid trivial solution, assume also that $b_r + b_o > k$. Let u_r the amount invested in real estate, and let u_o amount invested in oil.

Common sense solution is $u_0^* = k - b_o$ and $u_1^* = b_o$ since investment in oil has higher return. u_1 is investment in oil, and u_0 is investment in real estate. Let do this using D.P. The objective function is

$$J(u) = 2u_0 + 4u_1$$

And the state equation is (we only have two states x_1, x_0)

$$x_1 = x_0 - u_0$$

Initial state is $x_0 = k$ which is the money we have at the start. There are two stages. Hence $N = 2$. We start with

$$\begin{aligned} I(x_{N-1}, 1) &= I(x_1, 1) \\ &= \max_{u_1 \in \Omega_1} \{J(u)\} \end{aligned}$$

i.e. we assume we have one stage to go, and that we have made initial investment in real estate and now we are making investment in oil. Hence

$$\Omega_1 = [0 \cdots \min\{b_o, x_1\}]$$

Hence $u_1^* = \min\{b_o, x_1\}$ and

$$I(x_1, 1) = 4 \min\{b_o, x_1\}$$

Now backup one step.

$$\begin{aligned} I(x_0, 2) &= \max_{u_0 \in \Omega_0} \{J(u) + I(x_1, 1)\} \\ &= \max_{u_0 \in \Omega_0} \{2u_0 + 4 \min\{b_o, x_1\}\} \end{aligned}$$

Where $\Omega_0 = [0 \cdots b_o]$. Therefore since $x_1 = x_0 - u_0$ the above becomes

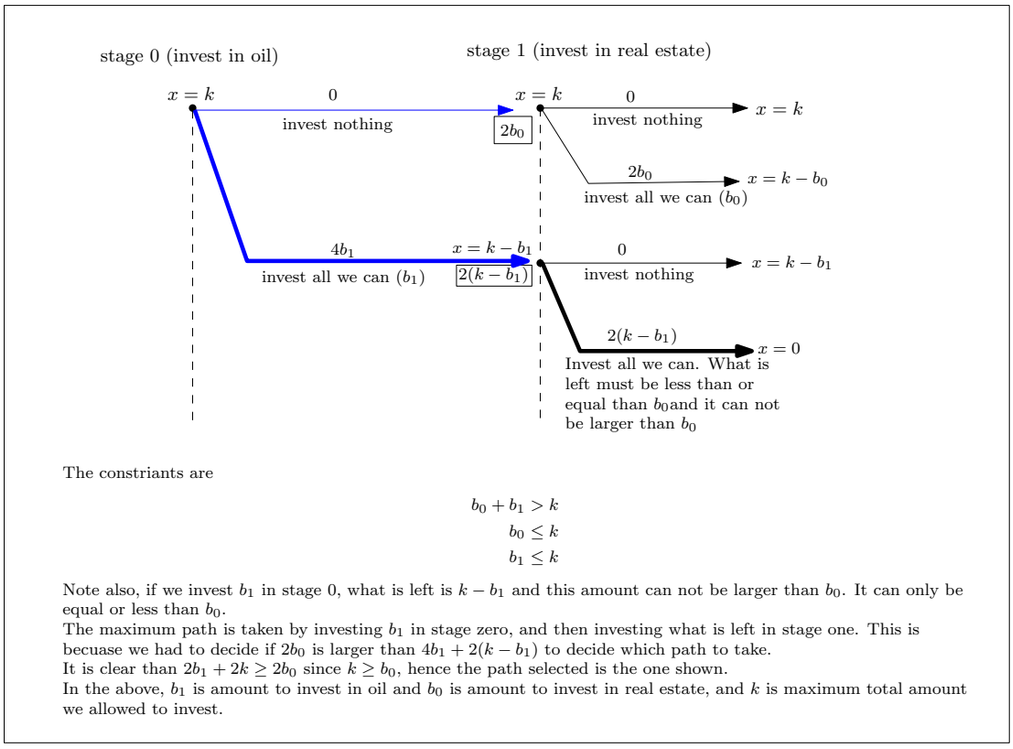
$$I(x_0, 2) = \max_{u_0 \in \Omega_0} \{2u_0 + 4 \min \{b_o, x_0 - u_0\}\}$$

Let

$$\begin{aligned} F(u_0) &= 2u_0 + 4 \min \{b_o, x_0 - u_0\} \\ &= 2u_0 + 4 \min \{b_o, k - u_0\} \end{aligned}$$

Find the maximum using graphics method. This gives $u_o^* = k - b_o$ which is the same using the common sense approach.

The following diagram shows the solution of the above using the dynamic graph method.



The constriants are

$$\begin{aligned} b_0 + b_1 &> k \\ b_0 &\le k \\ b_1 &\le k \end{aligned}$$

Note also, if we invest b_1 in stage 0, what is left is $k - b_1$ and this amount can not be larger than b_0. It can only be equal or less than b_0.
The maximum path is taken by investing b_1 in stage zero, and then investing what is left in stage one. This is becuase we had to decide if $2b_0$ is larger than $4b_1 + 2(k - b_1)$ to decide which path to take.
It is clear than $2b_1 + 2k \ge 2b_0$ since $k \ge b_0$, hence the path selected is the one shown.
In the above, b_1 is amount to invest in oil and b_0 is amount to invest in real estate, and k is maximum total amount we allowed to invest.

Figure 1.23: Solution to the oil and real estate problem using Branch and Bound graph method

1.27 Lecture 27. Tuesday, April 19, 2016

1.27.1 LQR and dynamic programming

In the following, some of the terms were re-written with the index being as subscript, as it is easier to see on the screen, Hence instead of $x(k)$ as was done in the lecture, it becomes x_k and $u(k)$ becomes u_k and so on.

Now we start with the state equation

$$\boxed{x_{k+1} = Ax_k + Bu_k}$$

In the above, A is $n \times n$ and B is $n \times m$ where m is the number of inputs, and u_k is column vector of size m, and similarly for x_{k+1} and x_k. In continuous time, the above is

$$\boxed{\dot{x}(t) = Ax(t) + Bu(t)}$$

We can also allow time varying A, B in the above, but in this discussion we assumed these are constant matrices. The goal is to make $x(k)$ track, or approach some desired $x^d(k)$ with as little effort $u(k)$ as possible (what about also as fast as possible at same time?). $u(k)$ is called the control effort. This diagram illustrates this

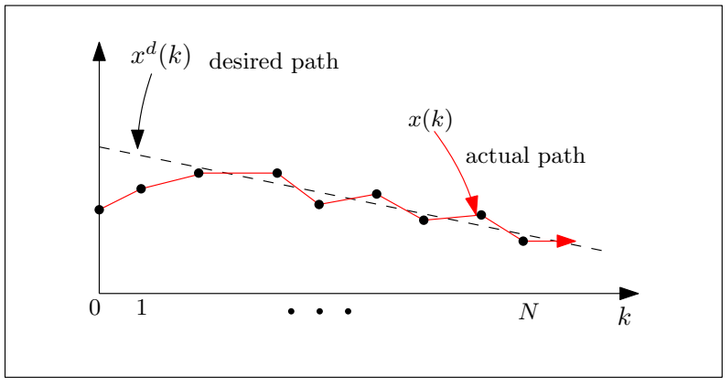


Figure 1.24: Goal is to track desired path

When $x^d(\infty) = 0$, this means we want to bring the system to stable state. We want now to quantify the goal $J(u)$. So the problem is to bring the state $x(k)$ to zero using as small effort u as possible. We write the cost $J(u)$ as

$$\boxed{J(u) = \sum_{k=0}^{N-1} x_{k+1}^T Q x_{k+1} + u_k^T R u_k}$$

This is just something we have to accept as given. We did not derive this, but it makes sense. Notice for example, when $x(k)$ is small, then $J(u)$ is small. But from now on, we just use the above as given. In the above, Q and R are called weighting matrices. For example, if $u(1)$ is more important than $u(2)$, we adjust the values in Q to reflect different weights we want to assign. In the above, Q is $n \times m$ and R is $m \times m$. Both Q and R are positive definite and symmetric. Now we start using the Bellman dynamic equations.

$$\begin{aligned} I(x_{N-1}, 1) &= \min_{u_{N-1} \in \Omega_{N-1}} \{J(x_{N-1}, u_{N-1}), \Psi(x_N)\} \\ &= \min_{u_{N-1} \in \Omega_{N-1}} \left\{x_N^T Q x_N + u_{N-1}^T R u_{N-1}\right\} \end{aligned} \tag{1}$$

We ignore the terminal cost $\Psi(x_N)$ for now. Be careful with the indices. Notice in the above, after replacing $J(u)$, that x has index N and the u has the $N-1$ index on it. This is due to how $J(u)$ is given to us to use, which has x_{k+1} in it already. EQ (1) is our starting point. Now we start applying the dynamic equations recursively. First , we replace the state equation in the above and obtain

$$I(x_{N-1}, 1) = \min_{u_{N-1} \in \Omega_{N-1}} \left\{(Ax_{N-1} + Bu_{N-1})^T Q (Ax_{N-1} + Bu_{N-1}) + u_{N-1}^T R u_{N-1}\right\} \tag{2}$$

Notice that now all indices on x are $N-1$, this is because the state equation being $x_{k+1} = Ax_k + Bu_k$. Notice that (2) is quadratic in u_{N-1}. This is important. Doing one more simplification

on (2) gives (where the leading $\min_{u_{N-1}\in\Omega_{N-1}}$ is now removed, just to make the equations fit), but it is assumed to be on each equation on what follows

$$\begin{aligned}
I(x_{N-1},1) &= \left((Ax_{N-1})^T + (Bu_{N-1})^T\right) Q\left(Ax_{N-1} + Bu_{N-1}\right) + u_{N-1}^T R u_{N-1} \\
&= \left(x_{N-1}^T A^T + u_{N-1}^T B^T\right) Q\left(Ax_{N-1} + Bu_{N-1}\right) + u_{N-1}^T R u_{N-1} \\
&= \left(x_{N-1}^T A^T Q + u_{N-1}^T B^T Q\right)\left(Ax_{N-1} + Bu_{N-1}\right) + u_{N-1}^T R u_{N-1} \\
&= \left(x_{N-1}^T A^T Q + u_{N-1}^T B^T Q\right)\left(Ax_{N-1}\right) + \left(x_{N-1}^T A^T Q + u_{N-1}^T B^T Q\right)\left(Bu_{N-1}\right) + u_{N-1}^T R u_{N-1} \\
&= x_{N-1}^T A^T Q A x_{N-1} + u_{N-1}^T B^T Q A x_{N-1} + x_{N-1}^T A^T Q B u_{N-1} + u_{N-1}^T B^T Q B u_{N-1} + u_{N-1}^T R u_{N-1} \\
&= \left(u_{N-1}^T B^T Q B u_{N-1} + u_{N-1}^T R u_{N-1}\right) + u_{N-1}^T B^T Q A x_{N-1} + x_{N-1}^T A^T Q B u_{N-1} + x_{N-1}^T A^T Q A x_{N-1} \\
&= u_{N-1}^T \left(B^T Q B + R\right) u_{N-1} + \left(u_{N-1}^T B^T Q A x_{N-1} + x_{N-1}^T A^T Q B u_{N-1}\right) + \left(x_{N-1}^T A^T Q A x_{N-1}\right) \\
&= u_{N-1}^T \left(B^T Q B + R\right) u_{N-1} + \left(u_{N-1}^T B^T Q A x_{N-1} + \left((QBu_{N-1})^T \left(x_{N-1}^T A^T\right)^T\right)^T\right) + x_{N-1}^T A^T Q A x_{N-1} \\
&= u_{N-1}^T \left(B^T Q B + R\right) u_{N-1} + \left(u_{N-1}^T B^T Q A x_{N-1} + \left((QBu_{N-1})^T \left(Ax_{N-1}\right)\right)^T\right) + x_{N-1}^T A^T Q A x_{N-1} \\
&= u_{N-1}^T \left(B^T Q B + R\right) u_{N-1} + \left(u_{N-1}^T B^T Q A x_{N-1} + \left(\left(u_{N-1}^T (QB)^T\right)\left(Ax_{N-1}\right)\right)^T\right) + x_{N-1}^T A^T Q A x_{N-1} \\
&= u_{N-1}^T \left(B^T Q B + R\right) u_{N-1} + \left(u_{N-1}^T B^T Q A x_{N-1} + \left(u_{N-1}^T \left(B^T Q^T\right)\left(Ax_{N-1}\right)\right)^T\right) + x_{N-1}^T A^T Q A x_{N-1} \\
&= u_{N-1}^T \left(B^T Q B + R\right) u_{N-1} + \left(u_{N-1}^T B^T Q^T A x_{N-1}\right)^T\right) + x_{N-1}^T A^T Q A x_{N-1}
\end{aligned}$$

But $Q^T = Q$ and the above becomes

$$\begin{aligned}
I(x_{N-1},1) = u_{N-1}^T \left(B^T Q B + R\right) u_{N-1} + \left(u_{N-1}^T B^T Q A x_{N-1} + \left(u_{N-1}^T B^T Q A x_{N-1}\right)^T\right) \\
+ x_{N-1}^T A^T Q A x_{N-1}
\end{aligned} \tag{2}$$

Using

$$\boxed{H = \frac{1}{2}\left(H + H^T\right)}$$

On the middle term $\left(u_{N-1}^T B^T Q A x_{N-1} + \left(u_{N-1}^T B^T Q A x_{N-1}\right)^T\right)$, where $H = u_{N-1}^T B^T Q A x_{N-1}$ reduces (2) to

$$I(x_{N-1},1) = \min_{u_{N-1}\in\Omega_{N-1}} \left\{u_{N-1}^T \left(B^T Q B + R\right) u_{N-1} + 2\left(u_{N-1}^T B^T Q A x_{N-1}\right) + x_{N-1}^T A^T Q A x_{N-1}\right\} \tag{3}$$

The above is in the form $I = a_1 u^2 + 2a_2 u + a_3$, therefore it is quadratic in u_{N-1}. Taking derivative w.r.t. u_{N-1} since this is what we are minimizing I with respect to, we obtain from (3)

$$\begin{aligned}
\frac{\partial I(x_{N-1},1)}{\partial u_{N-1}} &= 0 \\
0 &= 2\left(B^T Q B + R\right) u_{N-1} + 2\left(B^T Q A x_{N-1}\right)
\end{aligned}$$

R is positive definite, and Q is positive definite. Solving for u_{N-1} gives

$$\begin{aligned}
u_{N-1}^* &= -\left(B^T Q B + R\right)^{-1}\left(B^T Q A x_{N-1}\right) \\
&= -\left(B^T Q B + R\right)^{-1} B^T Q A x_{N-1}
\end{aligned} \tag{4}$$

Reader $\left(B^T Q B + R\right)$ is the Hessian. Show it is positive definite matrix. Since the Hessian is P.D., then u_{N-1}^* is global min. Eq (4) is linear feedback on state $(N-1)$. i.e. we write

$$u^*(N-1) = K(N-1)\, x(N-1)$$

Where $K(N-1)$ is called the gain matrix which is

$$K(N-1) = -\left(B^T Q B + R\right)^{-1} B^T Q A$$

In all the expressions below, we see the term $\left(B^T Q B + R\right)$ repeating. So to simplify things and make the equations smaller, let

$$\boxed{\Phi = B^T Q B + R}$$

Hence

$$K(N-1) = -\Phi^{-1} B^T Q A$$

And therefore

$$\begin{aligned} u^*_{N-1} &= \left(-\Phi^{-1}B^TQA\right)x_{N-1} \\ &= -\left(B^TQB+R\right)^{-1}B^TQAx_{N-1} \end{aligned}$$

Now we go back to (3) and replace the u_{N-1} in that expression with u^*_{N-1} we found in (4). (we remove the $\min_{u_{N-1}\in\Omega_{N-1}}$ we had in (3), since it is now the minimum)

$$\begin{aligned} I^*(x_{N-1},1) &= u^{*T}_{N-1}\Phi u^*_{N-1} + 2\left(u^{*T}_{N-1}B^TQAx_{N-1}\right) + x^T_{N-1}A^TQAx_{N-1} \\ &= \left(\left(-\Phi^{-1}B^TQA\right)x_{N-1}\right)^T\Phi\left(-\Phi^{-1}B^TQA\right)x_{N-1} \\ &\quad + 2\left(\left(-\Phi^{-1}B^TQA\right)x_{N-1}\right)^T\left(B^TQAx_{N-1}\right) + x^T_{N-1}A^TQAx_{N-1} \\ &= -x^T_{N-1}\left(\Phi^{-1}B^TQA\right)^T\Phi\left(-\Phi^{-1}B^TQA\right)x_{N-1} - 2x^T_{N-1}\left(\Phi^{-1}B^TQA\right)^T\left(B^TQAx_{N-1}\right) \\ &\quad + x^T_{N-1}A^TQAx_{N-1} \\ &= x^T_{N-1}\left[\left(\Phi^{-1}B^TQA\right)^T\left(B^TQA\right) - 2\left(\Phi^{-1}B^TQA\right)^T\left(B^TQA\right) + A^TQA\right]x_{N-1} \\ &= x^T_{N-1}\left[\left(B^TQA\right)^T\left(\Phi^{-1}\right)^T\left(B^TQA\right) - 2\left(B^TQA\right)^T\left(\Phi^{-1}\right)^T\left(B^TQA\right) + A^TQA\right]x_{N-1} \\ &= x^T_{N-1}\left[(QA)^TB\left(\Phi^{-1}\right)^T\left(B^TQA\right) - 2(QA)^TB\left(\Phi^{-1}\right)^T\left(B^TQA\right) + A^TQA\right]x_{N-1} \\ &= x^T_{N-1}\left[A^TQ^TB\left(\Phi^{-1}\right)^T\left(B^TQA\right) - 2A^TQ^TB\left(\Phi^{-1}\right)^T\left(B^TQA\right) + A^TQA\right]x_{N-1} \end{aligned}$$

But $Q = Q^T$ and $\left(\Phi^{-1}\right)^T = \Phi^{-1}$. Note that Φ is the Hessian matrix, and it is positive definite, and assumed symmetric. That is why $\left(\Phi^{-1}\right)^T = \Phi^{-1}$. But we did not proof this. It was a reader to show this is positive definite. The above therefore becomes

$$\begin{aligned} I^*(x_{N-1},1) &= x^T_{N-1}\left[A^TQB\Phi^{-1}B^TQA - 2A^TQ^TB\Phi^{-1}B^TQA + A^TQA\right]x_{N-1} \\ &= x^T_{N-1}\left[A^TQ - A^TQB\Phi^{-1}B^TQA\right]x_{N-1} \\ &= x^T_{N-1}A^TQ\left[I - B\Phi^{-1}B^TQA\right]x_{N-1} \\ &= x^T_{N-1}A^TQ\left[Q^{-1} - B\Phi^{-1}B^T\right]QAx_{N-1} \end{aligned}$$

Let

$$M_{N-1} = A^TQ\left[Q^{-1} - B\Phi^{-1}B^T\right]QA$$

Then

$$I^*(x_{N-1},1) = x^T_{N-1}M_{N-1}x_{N-1}$$

Now that we found $I^*(x_{N-1},1)$, we go back one more step

$$\begin{aligned} I(x_{N-2},2) &= \min_{u_{N-2}\in\Omega_{N-2}}\left\{J(x_{N-2},u_{N-2}) + I^*(x_{N-1},1)\right\} \\ &= \min_{u_{N-2}\in\Omega_{N-2}}\left\{x^T_{N-1}Qx_{N-1} + u^T_{N-2}Ru_{N-2} + I^*(x_{N-1},1)\right\} \\ &= \min_{u_{N-2}\in\Omega_{N-2}}\left\{x^T_{N-1}Qx_{N-1} + u^T_{N-2}Ru_{N-2} + x^T_{N-1}M_{N-1}x_{N-1}\right\} \\ &= \min_{u_{N-2}\in\Omega_{N-2}}\left\{x^T_{N-1}(Q+M_{N-1})x_{N-1} + u^T_{N-2}Ru_{N-2}\right\} \end{aligned} \tag{6}$$

Think of $(Q+M_{N-1})$ as the new Q matrix at stage $N-2$. We need to replace everything to be at $N-2$ stage, using the state equation $x_{k+1} = Ax_k + Bu_k$ then

$$x_{N-1} = Ax_{N-2} + Bu_{N-2}$$

Hence (6) becomes

$$I(x_{N-2},2) = \min_{u_{N-2}\in\Omega_{N-2}}\left\{(Ax_{N-2}+Bu_{N-2})^T(Q+M_{N-1})(Ax_{N-2}+Bu_{N-2}) + u^T_{N-2}Ru_{N-2}\right\}$$

As above, remove $\min_{u_{N-2}\in\Omega_{N-2}}$ in what follows so that equations fit on the page and let

$$\boxed{Q' = Q + M_N - 1}$$

Then

$$\begin{aligned}
I(x_{N-2},2) &= \left((Ax_{N-2})^T + (Bu_{N-2})^T\right) Q'\left(Ax_{N-2} + Bu_{N-2}\right) + u_{N-2}^T R u_{N-2} \\
&= \left((Ax_{N-2})^T Q' + (Bu_{N-2})^T Q'\right)\left(Ax_{N-2} + Bu_{N-2}\right) + u_{N-2}^T R u_{N-2} \\
&= (Ax_{N-2})^T Q'\left(Ax_{N-2} + Bu_{N-2}\right) + (Bu_{N-2})^T Q'\left(Ax_{N-2} + Bu_{N-2}\right) + u_{N-2}^T R u_{N-2} \\
&= (Ax_{N-2})^T Q' Ax_{N-2} + (Ax_{N-2})^T Q' Bu_{N-2} + (Bu_{N-2})^T Q' Ax_{N-2} + (Bu_{N-2})^T Q' Bu_{N-2} + \\
&\quad u_{N-2}^T R u_{N-2} \qquad (1.1) \\
&= x_{N-2}^T A^T Q' Ax_{N-2} + x_{N-2}^T A^T Q' Bu_{N-2} + u_{N-2}^T B^T Q' Ax_{N-2} + u_{N-2}^T B^T Q' Bu_{N-2} + u_{N-2}^T R u_{N-2} \\
&= u_{N-2}^T \left(B^T Q' B + R\right) u_{N-2} + x_{N-2}^T \left(A^T Q' A\right) x_{N-2} + x_{N-2}^T A^T Q' Bu_{N-2} + u_{N-2}^T B^T Q' Ax_{N-2} \qquad (7)
\end{aligned}$$

But

$$\begin{aligned}
x_{N-2}^T A^T Q' Bu_{N-2} + u_{N-2}^T B^T Q' Ax_{N-2} &= \left(u_{N-2}^T \left(x_{N-2}^T A^T Q' B\right)^T\right)^T + u_{N-2}^T B^T Q' Ax_{N-2} \\
&= \left(u_{N-2}^T \left(\left(A^T Q' B\right)^T x_{N-2}\right)\right)^T + u_{N-2}^T B^T Q' Ax_{N-2} \\
&= \left(u_{N-2}^T \left((Q'B)^T Ax_{N-2}\right)\right)^T + u_{N-2}^T B^T Q' Ax_{N-2} \\
&= \left(u_{N-2}^T \left(B^T (Q')^T Ax_{N-2}\right)\right)^T + u_{N-2}^T B^T Q' Ax_{N-2}
\end{aligned}$$

But $Q' = (Q')^T$ since symmetric[2] then the above becomes

$$\begin{aligned}
x_{N-2}^T A^T Q' Bu_{N-2} + u_{N-2}^T B^T Q' Ax_{N-2} &= \left(u_{N-2}^T B^T Q' Ax_{N-2}\right)^T + u_{N-2}^T B^T Q' Ax_{N-2} \\
&= 2\left(u_{N-2}^T B^T Q' Ax_{N-2}\right) \qquad (8)
\end{aligned}$$

Using $H = \frac{1}{2}\left(H + H^T\right)$. Replacing (8) into (7) gives

$$I(x_{N-2},2) = u_{N-2}^T \left(B^T Q' B + R\right) u_{N-2} + x_{N-2}^T \left(A^T Q' A\right) x_{N-2} + 2\left(u_{N-2}^T B^T Q' Ax_{N-2}\right) \qquad (9)$$

We now find u_{N-2}^*

$$\begin{aligned}
\frac{\partial I(x_{N-2},2)}{\partial u_{N-2}} &= 0 \\
0 &= 2u_{N-2}\left(B^T Q' B + R\right) + 2B^T Q' Ax_{N-2}
\end{aligned}$$

Hence

$$u_{N-2}^* = \left(-\left(B^T Q' B + R\right)^{-1} B^T Q' A\right) x_{N-2}$$

Where $Q' = Q + M_{N-1}$. Hence

$$\begin{aligned}
K(N-2) &= -\left(B^T Q' B + R\right)^{-1} B^T Q' A \\
&= -\left(B^T \left(Q + M_{N-1}\right) B + R\right)^{-1} B^T \left(Q + M_{N-1}\right) A
\end{aligned}$$

Now we go back to $I(x_{N-2},2)$ and replace u_{N-2} with u_{N-2}^* we just found. From (9)

$$\begin{aligned}
I^*(x_{N-2},2) &= \left(-\left(B^T Q' B + R\right)^{-1} B^T Q' Ax_{N-2}\right)^T \left(B^T Q' B + R\right)\left(-\left(B^T Q' B + R\right)^{-1} B^T Q' Ax_{N-2}\right) + \\
&\quad x_{N-2}^T \left(A^T Q' A\right) x_{N-2} + 2\left(\left(-\left(B^T Q' B + R\right)^{-1} B^T Q' Ax_{N-2}\right)^T B^T Q' Ax_{N-2}\right) \\
&= \left(B^T Q' Ax_{N-2}\right)^T \left(B^T Q' B + R\right)^{-1}\left(B^T Q' B + R\right)\left(B^T Q' B + R\right)^{-1} B^T Q' Ax_{N-2} + \\
&\quad x_{N-2}^T \left(A^T Q' A\right) x_{N-2} - 2\left(\left(B^T Q' B + R\right)^{-1} B^T Q' Ax_{N-2}\right)^T B^T Q' Ax_{N-2} \\
&= \left(B^T Q' Ax_{N-2}\right)^T \left(B^T Q' B + R\right)^{-1} B^T Q' Ax_{N-2} + x_{N-2}^T \left(A^T Q' A\right) x_{N-2} \\
&- 2\left(B^T Q' Ax_{N-2}\right)^T \left(B^T Q' B + R\right)^{-1} B^T Q' Ax_{N-2} \\
&= x_{N-2}^T \left(B^T Q' A\right)^T \left(B^T Q' B + R\right)^{-1} B^T Q' Ax_{N-2} + x_{N-2}^T \left(A^T Q' A\right) x_{N-2} \\
&= x_{N-2}^T \left[\left(A^T Q'^T B\right)\left(B^T Q' B + R\right)^{-1} B^T Q' A + \left(A^T Q' A\right)\right] x_{N-2}
\end{aligned}$$

Reader Argue that $I^*(x_{N-2},2)$ looks like

$$I^*(x_{N-2},2) = x_{N-2}^T M_{N-2} x_{N-2}$$

Reader Find M_{N-2}

[2]Need proof

$$\begin{aligned} M_{N-2} &= \left(A^T Q'^T B\right)\left(B^T Q' B + R\right)^{-1} B^T Q' A + \left(A^T Q' A\right) \\ &= \left(A^T \left(Q + M_N - 1\right)^T B\right)\left(B^T \left(Q + M_N - 1\right) B + R\right)^{-1} B^T \left(Q + M_N - 1\right) A \\ &\quad + \left(A^T \left(Q + M_N - 1\right) A\right) \end{aligned}$$

But $M_N - 1 = A^T Q\left[Q^{-1} - B\Phi^{-1}B^T\right]QA$, hence

$$\begin{aligned} M_{N-2} = &\left(A^T \left(Q + \left(A^T Q\left[Q^{-1} - B\Phi^{-1}B^T\right]QA\right)\right)^T B\right)\left(B^T \left(Q + M_N - 1\right) B + R\right)^{-1} B^T \\ &\left(Q + \left(A^T Q\left[Q^{-1} - B\Phi^{-1}B^T\right]QA\right)\right) A + \left(A^T \left(Q + \left(A^T Q\left[Q^{-1} - B\Phi^{-1}B^T\right]QA\right)\right) A\right) \end{aligned}$$

But $\Phi = B^T QB + R$, hence the above becomes

$$M_{N-2} = \left(A^T \left(Q + \left(A^T Q\left[Q^{-1} - B\left(B^T QB + R\right)^{-1} B^T\right]QA\right)\right)^T B\right)\left(B^T \left(Q + M_N - 1\right) B + R\right)^{-1} B^T \left(Q + \left(A^T Q\left[Q^{-1} - B\left(B\right.\right.\right.$$

Summary

$$\boxed{\begin{aligned} u^*_{N-i} &= K_{N-i} x_{N-i} \\ I^*\left(x_{N-i}\right) &= x^T_{N-i} M_{N-i} x_{N-i} \end{aligned}}$$

Reader Find a formula for $K(N-3)$.

The expression for K_{N-i} is

$$\begin{aligned} K_{N-1} &= -\left(B^T QB + R\right)^{-1} B^T QA \\ K_{N-2} &= -\left(B^T \left(Q + M_{N-1}\right) B + R\right)^{-1} B^T \left(Q + M_{N-1}\right) A \\ &= -\left(B^T QB + R + B^T M_{N-1} B\right)^{-1} \left(B^T QA + B^T M_{N-1} A\right) \end{aligned}$$

Where $M_{N-1} = A^T Q\left[Q^{-1} - B\Phi^{-1}B^T\right]QA$ and $\Phi = B^T QB + R$, hence

$$\begin{aligned} K_{N-2} &= -\left(B^T QB + R + B^T \left(A^T Q\left[Q^{-1} - B\Phi^{-1}B^T\right]QA\right) B\right)^{-1} \\ &\quad \left(B^T QA + B^T \left(A^T Q\left[Q^{-1} - B\Phi^{-1}B^T\right]QA\right) A\right) \\ &= -\left(B^T QB + R + B^T \left(A^T Q\left[Q^{-1} - B\left(B^T QB + R\right)^{-1} B^T\right]QA\right) B\right)^{-1} \\ &\quad \left(B^T QA + B^T \left(A^T Q\left[Q^{-1} - B\left(B^T QB + R\right)^{-1} B^T\right]QA\right) A\right) \end{aligned}$$

The final optimal cost will be

$$J^* = x_0^T M_0 x_0$$

1.27.2 Example LQR using dynamic programming

Reader Let $A = \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, Q = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ and $R = 1$, the weight on input. For $N = 3$, find $K(2), K(1), K(0)$ solving for LQR problem. We will get optimal gain $K(i)$ and these will not be the same. We do not like time varying gains, as in this case. We like the gain matrix to be constant, as it is easier to manger and more safe to use. If we make N very large, then gain will become constant. We start from very large N and go back to zero.

Solution

$N = 3$,

$$K(N-1) = -\left(B^T Q B + R\right)^{-1} B^T Q A$$

$$K(2) = -\left(\begin{pmatrix}0\\1\end{pmatrix}^T \begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} + 1\right)^{-1} \begin{pmatrix}0\\1\end{pmatrix}^T \begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix}$$

$$= -(2+1)^{-1}\begin{pmatrix}-1 & 4\end{pmatrix}$$

$$= \frac{-1}{3}\begin{pmatrix}-1 & 4\end{pmatrix}$$

$$= \begin{pmatrix}\frac{1}{3} & \frac{-4}{3}\end{pmatrix}$$

Hence

$$\boxed{K(2) = \begin{pmatrix}\frac{1}{3} & \frac{-4}{3}\end{pmatrix} = \begin{pmatrix}0.33333 & -1.333\,3\end{pmatrix}}$$

Will do things from now on using the state equations directly, as it seems easier. Starting again

$$I(x(3-1),1) = \min_{u(2)}\{J(x_2,u_2)\}$$

$$I(x_2,1) = \min_{u_2}\left\{x_3^T Q x_3 + u_2^T R u_2\right\}$$

But $x_3 = Ax_2 + Bu_2$ from state equation, hence the above becomes

$$I(x_2,1) = \min_{u_2}\left\{(Ax_2+Bu_2)^T Q (Ax_2+Bu_2) + u_2^T R u_2\right\}$$

$$= \min_{u_2}\left\{\left(\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix}x_2 + \begin{pmatrix}0\\1\end{pmatrix}u_2\right)^T \begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\left(\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix}x_2 + \begin{pmatrix}0\\1\end{pmatrix}u_2\right) + u_2^T(1)\,u_2\right\}$$

$$= \min_{u_2}\left\{\left(\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix}\begin{pmatrix}x_{11}\\x_{21}\end{pmatrix} + \begin{pmatrix}0\\u_2\end{pmatrix}\right)^T \begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\left(\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix}\begin{pmatrix}x_{11}\\x_{21}\end{pmatrix} + \begin{pmatrix}0\\u_2\end{pmatrix}\right) + u_2^2\right\}$$

$$= \min_{u_2}\left\{\begin{pmatrix}x_{11}-2x_{21}\\u_2 - x_{11} + 3x_{21}\end{pmatrix}^T \begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\begin{pmatrix}x_{11}-2x_{21}\\u_2 - x_{11} + 3x_{21}\end{pmatrix} + u_2^2\right\}$$

$$= \min_{u_2}\left\{(x_{11}-2x_{21})(u_2 + x_{11} - x_{21}) + (2u_2 - x_{11} + 4x_{21})(u_2 - x_{11} + 3x_{21}) + u_2^2\right\}$$

$$= \min_{u_2}\left\{2u_2^2 - 2u_2x_{11} + 8u_2x_{21} + 2x_{11}^2 - 10x_{11}x_{21} + 14x_{21}^2 + u_2^2\right\}$$

$$= \min_{u_2}\left\{3u_2^2 - 2u_2x_{11} + 8u_2x_{21} + 2x_{11}^2 - 10x_{11}x_{21} + 14x_{21}^2\right\}$$

Hence

$$\frac{\partial I(x_2,1)}{\partial u_2} = 0$$

$$0 = 6u_2 - 2x_{11} + 8x_{21}$$

$$0 = 6u_2 + \begin{pmatrix}-2 & 8\end{pmatrix}\begin{pmatrix}x_{11}\\x_{21}\end{pmatrix}$$

$$0 = 6u_2 + \begin{pmatrix}-2 & 8\end{pmatrix}x_1$$

$$u_2^* = -\frac{1}{6}\begin{pmatrix}-2 & 8\end{pmatrix}x_1$$

$$= \begin{pmatrix}\frac{2}{6} & \frac{-8}{6}\end{pmatrix}x_1$$

Therefore

$$\boxed{u_2^* = \begin{pmatrix}\frac{1}{3} & \frac{-4}{3}\end{pmatrix}\begin{pmatrix}x_{11}\\x_{21}\end{pmatrix}}$$

And

$$\boxed{K(2) = \begin{pmatrix}\frac{1}{3} & \frac{-4}{3}\end{pmatrix} = \begin{pmatrix}0.33333 & -1.333\,3\end{pmatrix}}$$

Which is the same as above using $-\left(B^T Q B + R\right)^{-1} B^T Q A$.

Now we find $I^*(x_2,1)$ by using u_2^* found above back in $I(x_2,1)$

$$I^*(x_2,1) = \left(\begin{pmatrix}1 & -2\\ -1 & 3\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}u_2\right)^T \begin{pmatrix}2 & 1\\ 1 & 2\end{pmatrix} \left(\begin{pmatrix}1 & -2\\ -1 & 3\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}u_2\right) + u_2^T(1)\,u_2$$

$$= \left(\begin{pmatrix}1 & -2\\ -1 & 3\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}\begin{pmatrix}\frac{1}{3} & \frac{-4}{3}\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix}\right)^T \begin{pmatrix}2 & 1\\ 1 & 2\end{pmatrix}\left(\begin{pmatrix}1 & -2\\ -1 & 3\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}\begin{pmatrix}\frac{1}{3} & \frac{-4}{3}\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix}\right)$$

$$+ \left(\begin{pmatrix}\frac{1}{3} & \frac{-4}{3}\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix}\right)^T \begin{pmatrix}\frac{1}{3} & \frac{-4}{3}\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix}$$

$$= \begin{pmatrix}x_{11}-2x_{21}\\ \frac{5}{3}x_{21}-\frac{2}{3}x_{11}\end{pmatrix}^T \begin{pmatrix}2 & 1\\ 1 & 2\end{pmatrix}\begin{pmatrix}x_{11}-2x_{21}\\ \frac{5}{3}x_{21}-\frac{2}{3}x_{11}\end{pmatrix} + x_{11}\left(\frac{1}{9}x_{11}-\frac{4}{9}x_{21}\right) - x_{21}\left(\frac{4}{9}x_{11}-\frac{16}{9}x_{21}\right)$$

$$= (x_{11}-2x_{21})\left(\frac{4}{3}x_{11}-\frac{7}{3}x_{21}\right) + \left(\frac{1}{3}x_{11}-\frac{4}{3}x_{21}\right)\left(\frac{2}{3}x_{11}-\frac{5}{3}x_{21}\right) + x_{11}\left(\frac{1}{9}x_{11}-\frac{4}{9}x_{21}\right) - x_{21}\left(\frac{4}{9}x_{11}-\frac{16}{9}x_{21}\right)$$

$$= \frac{5}{3}x_{11}^2 - \frac{22}{3}x_{11}x_{21} + \frac{26}{3}x_{21}^2$$

We have to convert this to $x_1^T x_1$ to be able to use it in the next stage since we need to apply the state equation to it. We see that

$$\frac{5}{3}x_{11}^2 - \frac{22}{3}x_{11}x_{21} + \frac{26}{3}x_{21}^2 = \begin{pmatrix}x_{11} & x_{21}\end{pmatrix}\begin{pmatrix}c_{11} & c_{12}\\ c_{12} & c_{22}\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix}$$

Solving gives $\begin{pmatrix}c_{11} & c_{12}\\ c_{12} & c_{22}\end{pmatrix} = \begin{pmatrix}\frac{5}{3} & -\frac{22}{6}\\ -\frac{22}{6} & \frac{26}{3}\end{pmatrix}$, hence

$$\boxed{I^*(x_2,1) = x_2^T\begin{pmatrix}\frac{5}{3} & -\frac{22}{6}\\ -\frac{22}{6} & \frac{26}{3}\end{pmatrix}x_2}$$

Now we find $I(x(N-2),2) = I(x(3-2),2) = I(x_1,2)$

$$I(x_1,2) = \min_{u_1}\{J(x_1,u_1) + I^*(x_2,1)\} = \min_{u_3}\left\{x_2^T Q x_2 + u_1^T R u_1 + I^*(x_2,1)\right\}$$

But $x_2 = Ax_1 + Bu_1$ from state equation, hence the above becomes

$$I(x_1,2) = \min_{u_3}\left\{(Ax_1+Bu_1)^T Q(Ax_1+Bu_1) + u_1^T R u_1 + I^*(x_2,1)\right\}$$

$$= \left(\begin{pmatrix}1 & -2\\ -1 & 3\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}u_1\right)^T \begin{pmatrix}2 & 1\\ 1 & 2\end{pmatrix}\left(\begin{pmatrix}1 & -2\\ -1 & 3\end{pmatrix}\begin{pmatrix}x_{11}\\ x_{21}\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}u_1\right) + u_1^2 + x_2^T\begin{pmatrix}\frac{5}{3} & -\frac{22}{6}\\ -\frac{22}{6} & \frac{26}{3}\end{pmatrix}x_2$$

Now we have to use the state equations in $I^*(x_2,1)$ to update the last term above, this is

important, since everything should be at the same state

$$
\begin{aligned}
I(x_1,2) &= 3u_1^2 - 2u_1x_{11} + 8u_1x_{21} + 2x_{11}^2 - 10x_{11}x_{21} + 14x_{21}^2 \\
&+ \left(\begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_1\right)^T \begin{pmatrix} \frac{5}{3} & -\frac{22}{6} \\ -\frac{22}{6} & \frac{26}{3} \end{pmatrix}\left(\begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_1\right)
\end{aligned}
$$

$$
\begin{aligned}
&= 3u_1^2 - 2u_1x_{11} + 8u_1x_{21} + 2x_{11}^2 - 10x_{11}x_{21} + 14x_{21}^2 \\
&\left(\begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix} + \begin{pmatrix} 0 \\ u_1 \end{pmatrix}\right)^T \begin{pmatrix} \frac{5}{3} & -\frac{22}{6} \\ -\frac{22}{6} & \frac{26}{3} \end{pmatrix}\left(\begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix} + \begin{pmatrix} 0 \\ u_1 \end{pmatrix}\right)
\end{aligned}
$$

$$
\begin{aligned}
&= 3u_1^2 - 2u_1x_{11} + 8u_1x_{21} + 2x_{11}^2 - 10x_{11}x_{21} + 14x_{21}^2 \\
&\frac{26}{3}u_1^2 - \frac{74}{3}u_1x_{11} + \frac{200}{3}u_1x_{21} + \frac{53}{3}x_{11}^2 - \frac{286}{3}x_{11}x_{21} + \frac{386}{3}x_{21}^2
\end{aligned}
$$

$$
= \frac{35}{3}u_1^2 - \frac{80}{3}u_1x_{11} + \frac{224}{3}u_1x_{21} + \frac{59}{3}x_{11}^2 - \frac{316}{3}x_{11}x_{21} + \frac{428}{3}x_{21}^2
$$

Now we take derivative to find optimal u_1^*

$$
\begin{aligned}
\frac{\partial I(x_1,2)}{\partial u_1} &= 0 \\
0 &= \frac{70}{3}u_1 - \frac{80}{3}x_{11} + \frac{224}{3}x_{21} \\
u_1^* &= \frac{3}{70}\left(\frac{80}{3}x_{11} - \frac{224}{3}x_{21}\right) \\
&= \frac{8}{7}x_{11} - \frac{16}{5}x_{21} \\
&= \begin{pmatrix} \frac{8}{7} & -\frac{16}{5} \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix}
\end{aligned}
$$

Hence

$$
\boxed{u_1^* = \begin{pmatrix} \frac{8}{7} & -\frac{16}{5} \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix}}
$$

And

$$
K(1) = \begin{pmatrix} \frac{8}{7} & -\frac{16}{5} \end{pmatrix} = \begin{pmatrix} 1.1429 & -3.2 \end{pmatrix}
$$

Therefore, we find $I^*(x_1,2)$ using u_1^*

$$
I^*(x_1,2) = \left(\begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_1^*\right)^T \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\left(\begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_1^*\right) + u_1^{*T}(1)\, u_1^*
$$

But $u_1^* = \begin{pmatrix} \frac{8}{7} & -\frac{16}{5} \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix}$, hence the above becomes

$$
\begin{aligned}
I^*(x_1,2) &= \begin{pmatrix} x_{11} - 2x_{21} \\ \frac{1}{7}x_{11} - \frac{1}{5}x_{21} \end{pmatrix}^T \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} x_{11} - 2x_{21} \\ \frac{1}{7}x_{11} - \frac{1}{5}x_{21} \end{pmatrix} + \left(\begin{pmatrix} \frac{8}{7} & -\frac{16}{5} \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix}\right)^T \begin{pmatrix} \frac{8}{7} & -\frac{16}{5} \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix} \\
&= (x_{11} - 2x_{21})\left(\frac{15}{7}x_{11} - \frac{21}{5}x_{21}\right) - \left(\frac{1}{5}x_{21} - \frac{1}{7}x_{11}\right)\left(\frac{9}{7}x_{11} - \frac{12}{5}x_{21}\right) \\
&+ x_{11}\left(\frac{64}{49}x_{11} - \frac{128}{35}x_{21}\right) - x_{21}\left(\frac{128}{35}x_{11} - \frac{256}{25}x_{21}\right)
\end{aligned}
$$

$$
= \frac{178}{49}x_{11}^2 - \frac{82}{5}x_{11}x_{21} + \frac{478}{25}x_{21}^2
$$

We have to write the above as x^TCx in order to update the state the next stage. As before,

we solve for C from

$$\begin{aligned}\frac{178}{49}x_{11}^2 - \frac{82}{5}x_{11}x_{21} + \frac{478}{25}x_{21}^2 &= \begin{pmatrix} x_{11} & x_{21}\end{pmatrix}\begin{pmatrix} c_{11} & c_{12} \\ c_{12} & c_{22}\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix} \\ &= c_{11}x_{11}^2 + 2c_{12}x_{11}x_{21}c_{22}x_{21}^2\end{aligned}$$

Hence $c_{11} = \frac{178}{49}, c_{22} = \frac{478}{25}, c_{12} = -\frac{82}{10}$, therefore

$$I^*(x_1, 2) = \begin{pmatrix} x_{11} & x_{21}\end{pmatrix}\begin{pmatrix} \frac{178}{49} & \frac{82}{10} \\ \frac{82}{10} & \frac{478}{25}\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix}$$

Now we back one more final step to find $K(0)$. We find $I(x(N-3), 3) = I(x(0), 3)$

$$\begin{aligned}I(x_0, 3) &= \min_{u0}\left\{J(x_0, u_0) + I^*(x_1, 2)\right\} \\ &= \min_{u_0}\left\{x_1^T Q x_1 + u_0^T R u_0 + I^*(x_1, 2)\right\}\end{aligned}$$

But $x_1 = Ax_0 + Bu_0$ from state equation, hence the above becomes

$$\begin{aligned}I(x_0, 3) &= \min_{u_0}\left\{(Ax_0 + Bu_0)^T Q(Ax_0 + Bu_0) + u_0^T R u_0 + I^*(x_1, 2)\right\} \\ &= \left(\begin{pmatrix} 1 & -2 \\ -1 & 3\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix} + \begin{pmatrix} 0 \\ 1\end{pmatrix}u_0\right)^T \begin{pmatrix} 2 & 1 \\ 1 & 2\end{pmatrix}\left(\begin{pmatrix} 1 & -2 \\ -1 & 3\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix} + \begin{pmatrix} 0 \\ 1\end{pmatrix}u_0\right) + u_0^2 \\ &+ x_1^T\begin{pmatrix} \frac{178}{49} & \frac{82}{10} \\ \frac{82}{10} & \frac{478}{25}\end{pmatrix}x_1\end{aligned}$$

Now we have to use the state equations in $I^*(x_1, 2)$ to update the last term above, this is important, since everything should be at the same state

$$\begin{aligned}I(x_0, 3) &= \left(\begin{pmatrix} 1 & -2 \\ -1 & 3\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix} + \begin{pmatrix} 0 \\ u_0\end{pmatrix}\right)^T \begin{pmatrix} 2 & 1 \\ 1 & 2\end{pmatrix}\left(\begin{pmatrix} 1 & -2 \\ -1 & 3\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix} + \begin{pmatrix} 0 \\ u_0\end{pmatrix}\right) + u_0^2 \\ &+ \left(\begin{pmatrix} 1 & -2 \\ -1 & 3\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix} + \begin{pmatrix} 0 \\ u_0\end{pmatrix}\right)^T \begin{pmatrix} \frac{178}{49} & \frac{82}{10} \\ \frac{82}{10} & \frac{478}{25}\end{pmatrix}\left(\begin{pmatrix} 1 & -2 \\ -1 & 3\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix} + \begin{pmatrix} 0 \\ u_0\end{pmatrix}\right)\end{aligned}$$

$$\begin{aligned}&= (x_{11} - 2x_{21})(u_0 + x_{11} - x_{21}) + u_0^2 + (2u_0 - x_{11} + 4x_{21})(u_0 - x_{11} + 3x_{21}) \\ &+ (x_{11} - 2x_{21})\left(\frac{41}{5}u_0 - \frac{1119}{245}x_{11} + \frac{4247}{245}x_{21}\right) + \\ &\left(\frac{478}{25}u_0 - \frac{273}{25}x_{11} + \frac{1024}{25}x_{21}\right)(u_0 - x_{11} + 3x_{21})\end{aligned}$$

$$= \frac{553}{25}u_0^2 - \frac{596}{25}u_0x_{11} + \frac{2248}{25}u_0x_{21} + \frac{10\,232}{1225}x_{11}^2 - \frac{70\,132}{1225}x_{11}x_{21} + \frac{125\,208}{1225}x_{21}^2$$

Now we take derivative to find optimal u_0^*

$$\begin{aligned}\frac{\partial I(x_0, 3)}{\partial u_0} &= 0 \\ 0 &= 2\frac{553}{25}u_0 - \frac{596}{25}x_{11} + \frac{2248}{25}x_{21}\end{aligned}$$

Hence

$$u_0^* = \frac{596}{603}x_{11} - \frac{2248}{603}x_{21}$$

Therefore

$$\boxed{u_0^* = \begin{pmatrix} \frac{596}{603} & -\frac{2248}{603}\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix} = \begin{pmatrix} 0.98839 & -3.728\end{pmatrix}\begin{pmatrix} x_{11} \\ x_{21}\end{pmatrix}}$$

And

$$K(0) = \begin{pmatrix} 0.988\,39 & -3.728\end{pmatrix}$$

Matlab `dlqr` gives a slightly different result and the signs are switched. This needs to be looked at it more. Let us verify $K(1)$ using the Bellman dynamic equations we derived

earlier which is

$$K(1) = -\left(B^TQB + R + B^T\left(A^TQ\left[Q^{-1} - B\left(B^TQB + R\right)^{-1}B^T\right]QA\right)B\right)^{-1}$$
$$\left(B^TQA + B^T\left(A^TQ\left[Q^{-1} - B\left(B^TQB + R\right)^{-1}B^T\right]QA\right)A\right) \tag{10}$$

Let $\Delta = B^T\left(A^TQ\left[Q^{-1} - B\left(B^TQB + R\right)^{-1}B^T\right]QA\right)$, hence

$$K(1) = -\left(B^TQB + R + \Delta B\right)^{-1}\left(B^TQA + \Delta A\right)$$

We already found $K(1) = \begin{pmatrix}1.1429 & -3.2\end{pmatrix}$ using the direct method. Just need to verify using the dynamic equations found.

$$\Delta = \begin{pmatrix}0\\1\end{pmatrix}^T\left(\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix}^T\begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\left[\begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}^{-1} - \begin{pmatrix}0\\1\end{pmatrix}\left(\begin{pmatrix}0\\1\end{pmatrix}^T\begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} + 1\right)^{-1}\begin{pmatrix}0\\1\end{pmatrix}^T\right]\begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix}\right)$$
$$= \begin{pmatrix}0\\1\end{pmatrix}^T\begin{pmatrix}1 & -1\\-1 & 4\end{pmatrix}\begin{pmatrix}\frac{2}{3} & -\frac{1}{3}\\-\frac{1}{3} & \frac{1}{3}\end{pmatrix}\begin{pmatrix}1 & -1\\-1 & 4\end{pmatrix}$$
$$= \begin{pmatrix}-\frac{11}{3} & \frac{26}{3}\end{pmatrix}$$

Hence (10) becomes

$$K(1) = -\left(\left(\begin{pmatrix}0\\1\end{pmatrix}^T\begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} + 1 + \begin{pmatrix}-\frac{11}{3} & \frac{26}{3}\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\right)^{-1}\left(\begin{pmatrix}0\\1\end{pmatrix}^T\begin{pmatrix}2 & 1\\1 & 2\end{pmatrix}\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix} + \begin{pmatrix}-\frac{11}{3} & \frac{26}{3}\end{pmatrix}\begin{pmatrix}1 & -2\\-1 & 3\end{pmatrix}\right)$$
$$= -\begin{pmatrix}-\frac{8}{7} & \frac{16}{5}\end{pmatrix}$$
$$= \begin{pmatrix}1.1429 & -3.2\end{pmatrix}$$

Which matches the direct method used above. Similar results are obtained using Matlab. Matlab has the signs reversed, this needs to be investigated to find out why.

```
>> A=[1,-2;-1,3];
>> B=[0;1];
>> Q=[2,1;1,2];
>> R=1;
>> [k,s,e]=dlqr(A,B,Q,R)
k =
-1.3195    3.6050
s =
7.0346  -10.3887
-10.3887   28.3824
e =
0.2679
0.1270
```

In Mathematica:

```
ssm = StateSpaceModel[{{{1, -2}, {-1, 3}}, {{0}, {1}}}];
q = {{1, -2}, {-1, 3}};
r = {{1}};
DiscreteLQRegulatorGains[ssm, {q, r}, 1.0]
{{-1.44328, 3.8633}}
```

.

Next we will talk about variations of dynamic programming. Floor and ceiling. We might want to optimize for maximum of some variable. For economy, this could be ceiling of unemployment. We can modify the dynamic equations to handle these problems. Floor and ceiling violate the additive process we used in D.P. but we can still manage to use the dynamic equations with some modifications.

1.28 Lecture 28. Thursday, April 21, 2016

1.28.1 Variations of dynamic programming, floor and ceiling

Today's lecture is on variations of dynamic programming. Many integer programming problem can be cast as D.P. The emphasis will be heuristics more than proofs.

One such variation is when the objective function is in terms of the floor or ceiling of variable. Another variation is steady state (this is when the number of stages becomes very large and goes to infinity).

Floor and ceiling To begin, say that x_k is scalar. We are interested in the floor of x_k. This is $\min_k x_k$. We can also talk about ceiling. This is $\max_k x(k)$. More formally

$$x_{k+1} = f(x_k, u_k)$$

Where $u(k) \in \Omega$. We introduce a monitoring function $g(x(k))$. Therefore, the floor problem can be stated as

$$\max_{u_k \in \Omega_k} \min_{k=1\cdots N} g(x_k)$$

For example, the ceiling problem could be to minimize the maximum of unemployment, stated as

$$\min_{u_k \in \Omega_k} \max_{k=1\cdots N} g(x_k)$$

Modeling our D.P. analysis Assume we are doing the floor problem now. Then we write

$$I(x_{N-1}, 1) = \max_{u_{N-1} \in \Omega_{N-1}} g(x_N)$$

And for the ceiling

$$I(x_{N-1}, 1) = \min_{u_{N-1} \in \Omega_{N-1}} g(x_N)$$

From now on, we continue with the ceiling problem. For the next stage we obtain

$$I(x_{N-2}, 2) = \min_{u_{N-2} \in \Omega_{N-2}} \left\{ \max \left\{ g(x_N), I^*(x_{N-1}, 1) \right\} \right\}$$

And the recursion formula becomes

$$I(x_L, N-L) = \min_{u_L \in \Omega_L} \left\{ \max \left\{ g(x_{L+1}), I^*(x_{L+1}, N-L-1) \right\} \right\}$$

When applying the above, we see one difference from previous examples of D.P., now we have Ω_L which might depends on $u_{L-1}, u_{L-2}, \cdots$. Now we will go over one example. A simplified economy model example. Let N be the years of horizon planning. Let y_k be the national income for the k year. Let c_k be the consumer expenditure. Let I_k be the business investment. And let u_k be the government expenditure. The constraint is

$$\sum_{k=0}^{N-1} u_k \le U$$

Where U is the budget given and $u_k \ge 0$. Hence in a given year we have

$$y_k = c_k + I_k + u_{k-1} \tag{1}$$
$$c_k = \alpha y_{k-1} \tag{2}$$

Where α is propensity factor to consume.

$$I_k = \beta\left(c_k - c_{k-1}\right) \tag{3}$$

Reader Eliminate c_k, I_k from (1,2,3) to obtain

$$y_k = \left(1+\beta\right)\alpha y_{k-1} - \alpha\beta y_{k-2} + u_{k-1} \tag{4}$$

Our goal is to control the output y_k using the input u_k.

Reader Let $z_k = y_k - y_{k-1}$

Therefore, now

$$y_{k-2} = y_{k-1} - z_{k-1}$$

Substituting the above in (4) gives

$$\begin{aligned} y_k &= \left(1+\beta\right)\alpha y_{k-1} - \alpha\beta\left(y_{k-1} - z_{k-1}\right) + u_{k-1} \\ &= \alpha y_{k-1} + \alpha\beta z_{k-1} + u_{k-1} \end{aligned}$$

We have state equation

$$y_{k+1} = \alpha y_k + \alpha\beta z_k + u_k$$

Next we want state equation for z_k

Reader

$$z_{k+1} = \left(\alpha - 1\right) y_k + \alpha\beta z_k + u_k$$

Now define state $x_1(k) = y_k$ and $x_2(k) = z_k$, hence the state equation in matrix form becomes

$$x_{k+1} = \begin{pmatrix} \alpha & \alpha\beta \\ \alpha - 1 & \alpha\beta \end{pmatrix} x_k + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u_k$$

We want to maximize the floor of the national income y_k using D.P. To illustrate two stages, i.e. $N = 2$, let $U = 1$ dollar, and let $\alpha = \beta = \frac{1}{2}$ Hence

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{-1}{2} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u_k$$

Monitor function is $g(x(k)) = x_1(k)$

Begin with one stage to go

$$I(x(1),1) = \max_{u(1)\in\Omega_1} x_1(2) \tag{5}$$

We know $u(1) \le 1 - u(0)$

Hence (5) becomes

$$I(x(1),1) = \max_{u(1)\in[0\cdots 1-u(0)]} \left(\frac{1}{2}x_1(1) + \frac{1}{4}x_2(1) + u(1)\right)$$

The minimizer is $u^*(1) = 1 - u(0)$, therefore

$$I^*(x(1),1) = \frac{1}{2}x_1(1) + \frac{1}{4}x_2(1) + 1 - u(0)$$

Now we go back one more step

$$I(x(0),2) = \max_{u(0)\in\Omega_0} \min\left\{g(x(1), I(x(1),1))\right\}$$

Where $\Omega_0 = [0 \ldots 1]$ since we have one dollar to start with. Hence

$$I(x(0),2)=\max_{u(0)\in[0...1]}\min\left\{x_1(1),\frac{1}{2}x_1(1)+\frac{1}{4}x_2(1)+1-u(0)\right\}$$

Back everything to get an equation in $u(0)$

$$I(x(0),2)=\max_{u(0)\in[0...1]}\min\left\{\frac{1}{2}x_1(0)+\frac{1}{4}x_2(0)+u(0),\frac{1}{2}\left(\frac{1}{2}x_1(0)+\frac{1}{4}x_2(0)+u(0)\right)+\frac{1}{4}\left(-\frac{1}{2}x_1(0)+\frac{1}{4}x_2(0)+u(0)\right)1-\right.$$

This reduces to

$$I(x(0),2)=\max_{u(0)\in[0...1]}\min\{A+u(0),B\}$$

Where

$$\begin{aligned}A&=\frac{1}{2}x_1(0)+\frac{1}{4}x_2(0)+u(0)\\B&=\frac{1}{8}x(0)+\frac{3}{16}x_2(0)+1-\frac{1}{4}u(0)\end{aligned}$$

Hence the function to minimize is

$$F(u(0))=\min\{A+u(0),B\}$$

Consider case when $A\geq B$ and case $A<B$.

Reader work out the different cases.

1.29 Lecture 29. Tuesday, April 26, 2016

1.29.1 Detailed example for a floor problem

We will start today with one more dynamic problem which will be useful for HW problem. Then we will start on steady state. Example is a floor problem.

$$\begin{aligned}x_1(k+1)&=\min\{x_1(k),x_2(k)\}+u(k)\\x_2(k+1)&=x_1(k)\,u(k)\end{aligned}$$

And initial state is

$$\begin{aligned}x_1(0)&=1\\x_2(0)&=-1\end{aligned}$$

And

$$J=\min_{k=1,2}x_2(k)$$

With $|u(k)|\leq M$. One step to go is

$$\begin{aligned}I(x(1),1)&=\max_{u(1)\in\Omega_1}\{\min J\}\\&=\max_{u(1)\in\Omega_1}x_2(2)\\&=\max_{|u(1)|\leq M}x_1(1)\,u(1)\end{aligned}$$

Hence $u^*=M\ \text{sign}(x_1(x))$, therefore

$$I^*(x(1),1)=M\ abs(x_1(1))$$

With two steps

$$\begin{aligned}I(x(0),2)&=\max_{u(0)\in\Omega_0}\{\min\{J(x(1),I(x(1),1))\}\}\\&=\max_{u(0)\in\Omega_0}\min\{x_2(1),M|x_1(1)|\}\\&=\max_{u(0)\in\Omega_0}\min\{x_1(0)\,u(0),M|\min\{x_1(0),x_2(0)\}+u(0)|\}\\&=\max_{u(0)\in\Omega_0}\min\{u(0),M|u(0)-1|\}\end{aligned}$$

Where $F(u) = \min\{u(0), M|u(0)-1|\}$. Consider the case $0 < M \le \sqrt{2}$ and case $M > \sqrt{2}$. See key solution for HW 7 for the solution.

We now start talking about steady state. Notion of functional equations., then iterative solution to steady state problem, then Riccati equation. We begin with

$$x_{k+1} = f(x_k, u_k)$$

With a constraint on u_k given by $u_k \in \Omega_k$. The branch cost is

$$J = \Psi(x_N) + \sum_{k=0}^{N-1} J_k(x_k, u_k)$$

Then let $N \to \infty$. We are hoping to get J_N and obtain $u_N^* = u_0^*, u_1^*, \cdots, u_{N-1}^*$ optimal controls at each stage. Notice that when N changes, then the whole sequence u_N^* changes also, and not just one term. We now ask, does J_N converge? does u_N^* converge? To make sense of the above, we remove the terminal cost $\Psi(x_N)$ from this analysis. We also want Ω_k to be fixed for any N. This means we have the same constraints all the time.

If steady state solution exist, then it satisfies

$$I(x) = \min_{u\in\Omega}\left\{J(x,u) + I\left(f(x,u)\right)\right\} \tag{1}$$

This is a functional equation. Solving it means finding u^*. The solution u^* in (1) is a function of x. i.e.. u at state x is a feedback. Say $u^* = \sigma(x)$. Substitute this in (1) gives

$$I(x) = \min_{u\in\Omega}\left\{J(x,\sigma(x)) + I\left(f(x,\sigma(x))\right)\right\}$$

We do not know $I(x)$ here. Before, we knew I, but now we do not know I. So we have a function space issue to find $I(x)$.

Example

$$J(x,u) = x^2 + xu + u^2$$
$$f(x,u) = xu + x + u$$
$$x(k+1) = x(k)\,u(k) + x(k) + u(k)$$

Let u be free variable with no constraints. Hence (1) becomes

$$I(x) = \min_{u}\left\{x^2 + xu + u^2 + I(xu + x + u)\right\}$$

1.29.2 Functional equations in dynamic programming

This is a functional equation since $I(x)$, appears on both sides of the equation. There are two famous methods to solve functional equations. The first is the iterative method. Begin with initial $I_o(x)$, then $I_{k+1}(x) = \min_{u(k)}\left\{J(x,u) + I_k\left(f(x,u)\right)\right\}$. We get sequence of solutions of $u(k)$ and $I(x(k))$ and check for convergence.

Example

$$J(x,u) = u^2 + (x-u)^2$$
$$x(k+1) = x(k) - u(k)$$

$I_0(x) = 0$, hence

$$I_1(x) = \min_{u}\left\{u^2 + (x-u)^2\right\}$$

$\frac{d}{du} = 0$ gives $u^* = \frac{x}{2}$, hence $I_1^*(x) = \frac{x^2}{4} + \frac{x^2}{4} = \frac{x^2}{2}$. Next stage becomes

$$I_2(x) = \min_{u}\left\{u^2 + (x-u)^2 + I_1^*\right\}$$
$$= \min_{u}\left\{u^2 + (x-u)^2 + \frac{(x-u)^2}{2}\right\}$$

$\frac{d}{du} = 0$ gives $u^* = \frac{3}{5}x$, hence $I_1^*(x) = \frac{3}{5}x^2$.

Reader Continue this process. Does it converge? It will converge eventually leading to $u^* \to kx$

1.30 Lecture 30. Thursday, April 28, 2016

The special problem will be returned next Tuesday.

1.30.1 Steady state and functional equations

The plan for today: we have been talking about steady state. This lead to functional equations in D.P. so far, we talked about iterative solution. Today we will talk about closed form solution. Analogy between differential equations and functional equations. In differential equations, the iterative method is called Picard iterations method.

For linear state equations, we can get closed form solution for the functional equation. We start with a guess of the solution with a parameter to find. Now we will use our main example to illustrate this.

$$\begin{aligned} x_{k+1} &= x_k - u_k \\ J &= \sum u_k^2 + (x_{k+1} - u_k)^2 \end{aligned}$$

Notice the cross term with x and u in it. Now we guess a form for I. Let $I = px^2$ and then we try to find p. First write J above so that all indices are the same with the help of the state equation. This will reduce the chance of error later on

$$\begin{aligned} J &= \sum u_k^2 + ((x_k - u_k) - u_k)^2 \\ &= \sum u_k^2 + x_k^2 + 4u_k^2 - 4x_k u_k \\ &= \sum x_k^2 + 5u_k^2 - 4x_k u_k \end{aligned}$$

Consider

$$\begin{aligned} I(x) &= \min_{u\in\Omega} \left\{ J(x,u) + I\left(f(x,u)\right) \right\} \\ px^2 &= \min_{u\in\Omega} \left\{ \left(5u^2 + x^2 - 4xu\right) + p(x-u)^2 \right\} \end{aligned} \tag{1}$$

$$\frac{d\left(5u^2 + x^2 - 4xu\right) + p(x-u)^2}{du} = 0$$

$$0 = 2u - 4(x - 2u) - 2p(x-u)$$

Solving gives $u^* = \frac{2+p}{5+p}x$. Substitute back in (1)

$$px^2 = \left(5u^2 + x^2 - 4x\left(\frac{2+p}{5+p}x\right)\right) + p\left(x - \left(\frac{2+p}{5+p}x\right)\right)^2$$

And obtain an equation in p and solve for p. We find roots are $p = 0.302$ and $p = -3.3$. If everything was done correct, there should be a positive root. Always pick the positive one. This is special case of LQR. In LQR there is no cross term between x, u. While in the above there was. **Reader** For $x(0) = 1$ find J^*.

Example Consider

$$x(k+1) = x(k) + 2u(k)$$

With constraint $u(k) \in [-1, 1]$

$$J = \sum_{k=0}^{\infty} e^{x(k+1)}$$

Guess $I = ae^x$ then

$$I(x) = \min_{u\in[-1,1]} \left(e^{x+2u} + ae^{x+2u}\right)$$

Reader $u^* = -1$

Therefore

$$ae^x = \left(e^{x-2} + ae^{x-2}\right)$$

Solving gives

$$a = \frac{1}{e^2 - 1} > 0$$

For LQR, the steady state is given by

$$x(k+1) = Ax(k) + Bu(k)$$
$$J = \sum_{k=0}^{\infty} x^T(k+1)\,Qx(k+1) + u^T(k)\,Ru(k)$$

Where Q, R are weight matrices and are positive definite symmetric. I should be quadratic in the state $x(k)$.

$$I(x) = \min_u x^T Qx + u^T Ru$$

Guess $I = x^T Px$ and now solve for P, this leads to Riccati matrix equation.

$$I(x) = \min_u \left\{ x^T Qx + u^T Ru + (Ax + Bu)^T P(Ax + Bu) \right\} \tag{2}$$

Taking gradient w.r.t. u and setting to zero, gives

$$2Ru + 2B^T PBu + 2B^T PAx = 0$$
$$u^* = -\left(R + B^T PB\right)^{-1} B^T PAx$$

Back to (2) we find

$$x^T Px = x^T Qx + \left(-\left(R + B^T PB\right)^{-1} B^T PAx\right)^T R\left(-\left(R + B^T PB\right)^{-1} B^T PAx\right)$$
$$+ \left(Ax + B\left(-\left(R + B^T PB\right)^{-1} B^T PAx\right)\right)^T P\left(Ax + B\left(-\left(R + B^T PB\right)^{-1} B^T PAx\right)\right)$$

Solving to P, we obtain the Riccati matrix equation

$$\boxed{P = A^T PA - A^T PB\left(R + B^T PB\right)^{-1} B^T PA + Q}$$

In Matlab, use `dare()` to solve this for P.

Remarks Are we sure the solution exist? i.e. does there exist positive definite P that satisfies the Riccati equation above? Yes, a solution exist if (A, B) is controllable system. Notice that the Riccati equation is not linear in P. This is solved numerically. Consider the special case of LQR with one state and one input. Hence everything is scalar. We obtain

$$p = a^2 p - apb\frac{1}{r + b^2 p}bpa + q$$

Solving for p, show that there is solution $p > 0$. Assume $b > 0$ for controllability.

1.31 Lecture 31. Tuesday, May 3, 2016

This is the last lecture. Final exam is next lecture. Review of special problem and results obtained by different reports. General approaches to solving the special problem included: Cluster analysis, noisy gradient and random search.

1.31.1 Final review for final exam

We talked about steady state. Quadratic regulator has no cross coupling terms between x and u

$$J = \sum_{k=0}^{\infty} x^T(x+1)\,Qx(k+1) + u^T Ru$$

For general regulator, one can get a cross term as in $J = ax^2 + bxu + cu$ but we did not discuss this.

Test 3, will have 4 questions on dynamic programming. With dynamic programming, one can solve the problem using the Bellman equations or using the graphical method. If there are finite stages, and the state x is discrete $x(k+1) = f(x(k), u(k))$ and if u is discrete, then a graphical method can be used. If there are constraints, this will reduce the size of the tree more.

Course review

We can take an integer linear programming problem, which is hard to solve and treat it as continuous problem under special conditions and solve it much easier. We did not

discuss non-linear programming and kuhn-Tucker conditions. But for many non-linear programming problem, it is possible to use the penalty method. There is also large scale linear programming, where sparsity becomes important. Also parallel programming become important for these problems. For dynamic programming, most of the books are on continuous time, and very few discusses discrete dynamic programming problems.

1.32 Lecture 32. Thursday, May 5, 2016

Final exam