EE5110: Special Topics in Automation and Control

Segment C: Control Optimization

Lecture Two

Xiang Cheng

Associate Professor

Department of Electrical & Computer Engineering
The National University of Singapore

Phone: 65166210 Office: Block E4-08-07
Email: elexc@nus.edu.sg

First, recall what problem was solved in lecture one?

For a first order system,

$$\dot{y} = u$$

We want to make both the output and input small on average.

$$J(y) = \int_0^T (y^2 + u^2)dt = \int_0^T (y^2 + \dot{y}^2)dt$$

How to find out the minimal?

For simple functions, we need to evaluate the change of the function around the optimal point :

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{f''(x)}{2}\Delta x^2 + \cdots$$

For functional, we introduce a way to compute the variations of the functional around the critical point!

$$J(y + \delta y) \quad \Longrightarrow \quad \delta y = \varepsilon z$$

From this, we derive the Euler Equation as well as Euler-Lagrange Equation.

Today, we will try to solve the same problem with a totally different point of view:

For a first order system, $\dot{y} = u$, we want to minimize

$$J(y) = \int_0^T (y^2 + u^2)dt = \int_0^T (y^2 + \dot{y}^2)dt$$

over all y(t) subject to the initial condition y(0) = c.

In place of asking for the solution y as a function of time we ask for the most efficient control at each point in the state space. At any particular point in the history of the process, we want instructions as to what to do next.

A control process is thus considered to be a multistage decision process consisting of the following operations:

(a) At time t, the state of the system is observed.

(b) Based on this information and a control law, a decision is made.

(c) This decision produces an effect upon the system.

We can constrain our attention to the first control action, i.e., the initial slope, the value u(0)=$\dot{y}(0)$.

What does this control action, u(0), depend upon?

The control action depends upon the initial state y(0) and the time horizon, T.

We can regard this as a function of c, the initial state, and T, the duration of the process. This basic function, which we denote by u(c, T), is called a **policy.**

A policy determines the control action given the state and the time horizon.

The objective is to find out the optimal policy.

In dynamic programming, we do not try to find the optimal policy directly. Instead, we will try to find out the optimal value function first!

The optimal policy can then be determined from the optimal value function.

What does the minimal value of J(y) depend upon?

The initial state c and the duration of the process T.

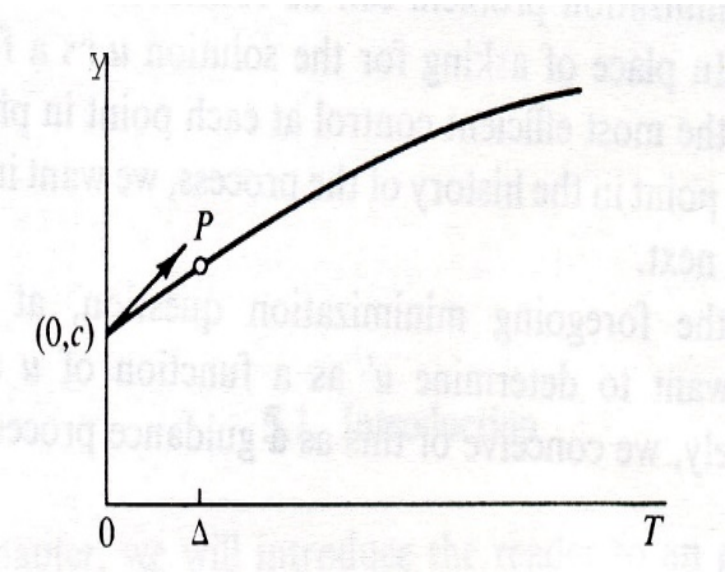
Optimal Value function:

$$\mathrm{V}(c,T) = \min_{y} J(y). \qquad (4.4.1)$$

This function plays the critical role in dynamic programming.

The key idea of dynamic programming is to find out the equation describing this optimal value function!

Let's start the decision at the initial time t=0, and take an action u(0) and evaluate its consequence.



$$\dot{y} = u$$

Let Δ be an infinitesimal, and let u = u(c, T) denote the control action at t=0, i.e., the initial slope at t= 0.

After this action, what is the state at next immediate time instant Δ , y(Δ)?

From the equation, $\dot{y} = u$, we can easily compute the state at t= Δ is c+Δu.

Now let's compute the cost caused by this action u(0).

Let's break down the cost into two terms:

$$J(y) = \int_0^{\Delta} + \int_{\Delta}^{T} . \qquad (4.4.2)$$

The first term corresponds to the immediate cost caused by the action. The second term corresponds to the long term cost caused by the actions after the first step.

Total Cost = Short term cost + Long term cost

$$J(y) = \int_0^{\Delta} \quad + \int_{\Delta}^{T} \quad .$$

Let's look at the short term cost occurred at the first step when u=u(0) is applied.

We are considering Δ to be an infinitesimal and systematically neglecting terms of order Δ^2 and higher. Hence, we can write

$$\int_0^{\Delta} [y^2 + \dot{y}^2] dt = (c^2 + u^2)\Delta + O(\Delta^2).$$

$$J(y) = \int_0^{\Delta} \quad + \int_{\Delta}^{T} \quad .$$

Total Cost = Short term cost + Long term cost

Computing the short term cost is easy. How to compute the long term cost?

In order to compute the long term cost, we need to know the control policy for the rest of the process.

Different policy will lead to different cost.

What is the control policy we are going to use after the first step?

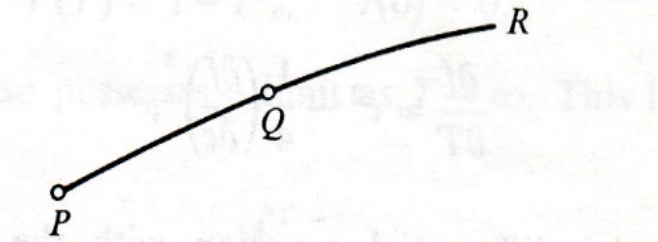
This is the most important step in Dynamic Programming.

Here we are going to use the Principle of Optimality to make the decision:

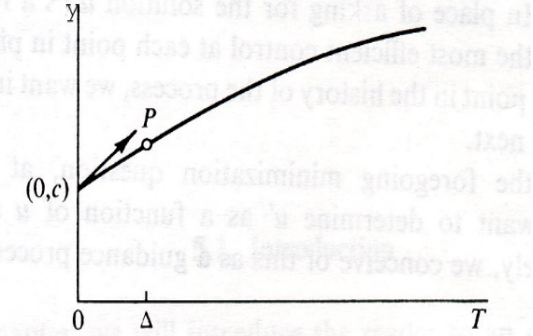
Principle of Optimality

An optimal policy has the property that whatever the initial state and initial decision are the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

If an optimal policy is conceived of as a means of tracing out an optimal trajectory in state space, say a geodesic as far as time or distance are concerned, then the principle becomes quite clear:



If PR is to be a path of minimum time, then clearly the last part, QR must also be a path of minimum time.



$$J(y) = \int_0^{\Delta} \quad + \int_{\Delta}^{T} \quad .$$

Regardless of how u has been chosen at t=0, we will continue from P at Δ in such a way as to minimize the cost incurred,

$$\int_{\Delta}^{T} [y^2 + \dot{y}^2] dt \, .$$

The long term cost is the minimal cost starting from the state at Δ!

Since it is the minimal cost, can we use the value function $V(c, T)$ to express this long term cost?

What is the state at t= Δ, y(Δ)? c + u Δ

What is the time horizon to reach the final state at T, starting from t= Δ? T-Δ

From the definition of the optimal value function, we have

$$\int_{\Delta}^{T} [y^2 + \dot{y}^2] dt \quad \rightarrow \quad V(c + u\,\Delta, T - \Delta)$$

$$J(y) = \int_0^{\Delta} \quad + \int_{\Delta}^{T} \quad .$$

In total we have

$$J(y) = (c^2 + u^2)\Delta + V(c + u\Delta, T - \Delta) + O(\Delta^2),$$

Total Cost = Short term cost + Long term cost

<u>Does the current action u(0) affect the long term cost?</u>

The control input u(0) will affect both the short term and long term costs!

Dynamic programming is about how to achieve the perfect balance between the long term and short term goals!

$$J(y) = (c^2 + u^2)\Delta + V(c + u\Delta, T - \Delta) + O(\Delta^2)$$

How should u be chosen?

We choose u to minimize the total cost. We must balance the cost of control over [0,Δ] against the cost of optimal control over [Δ, T]. Then the total cost will be minimized.

Hence, we have

$$\mathrm{V}(c, T) = \min_u [(c^2 + u^2)\Delta + V(c + u\Delta, T - \Delta) + O(\Delta^2)]. \qquad (4.4.6)$$

How to relate $V(c + u\Delta, T - \Delta)$ with $V(c, T)$?

How to express f(x+Δx) around the point x?

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{f''(x)}{2}\Delta x^2 + \cdots$$

Expanding in a Taylor series for multi-variable function, we have

$$f(x + \Delta x, y + \Delta y) = f(x, y) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y + O(\Delta^2)$$

$$\mathrm{V}(c + u\Delta, T - \Delta) = V(c, T) + \frac{\partial V}{\partial c}u\Delta - \frac{\partial V}{\partial T}\Delta + O(\Delta^2)$$

Hence, (4.4.6) becomes

$$\mathrm{V(c,T)} = \min_{u}[(c^2+u^2)\Delta + V(c,T) + \frac{\partial V}{\partial c}u\Delta - \frac{\partial V}{\partial T}\Delta + O(\Delta^2)]. \qquad (4.4.8)$$

Ignoring the higher order terms of Δ, we have

$$\frac{\partial V}{\partial T} = \min_{u}[(c^2+u^2) + \frac{\partial V}{\partial c}u]. \qquad (4.4.9)$$

The righthand side is a standard minimization problem, how to solve it?

Take the derivative with respect to u, we have

$$2\mathrm{u} + \frac{\partial V}{\partial c} = 0 \quad \Longrightarrow \quad u = -\frac{1}{2}\frac{\partial V}{\partial c}$$

Plug this into (4.4.9), we have

$$\frac{\partial V}{\partial T} = c^2 - \frac{1}{4}(\frac{\partial V}{\partial c})^2 \qquad (4.6.2)$$

$$\frac{\partial V}{\partial T} = c^2 - \frac{1}{4}\left(\frac{\partial V}{\partial c}\right)^2$$

It is a nonlinear partial differential equation!

What is the initial condition V(c, 0)?

Since the time duration for the integral is zero, V(c, 0) = 0.

If we compare it to the Euler equation in calculus of variations,

$$-\ddot{y}_0 + y_0 = 0$$

It looks more complicated! But do not worry. Let's proceed to find out the final solution!

$$\frac{\partial V}{\partial T} = c^2 - \frac{1}{4}(\frac{\partial V}{\partial c})^2 \qquad (4.6.2)$$

What is the most commonly used technique for solving PDE?

Separation of Variables!

The trick of solving PDE is to reduce the problem into solving ODE, and separation of variables is the most direct way.

Assume that the solution of the PDE takes the form

$$V(c,T) = f(c)r(T) \qquad (4.7.2)$$

Plug it into (4.6.2), we have

$$f(c)\,r' = c^2 - \frac{1}{4}(f'(c))^2\, r^2 \qquad (4.7.3)$$

What is next?

For the method of separation of variables to work, it is clear that all the three terms should contain c^2 such that they can cancel out, and we can get separate equations for f(c) and r(T).

We can simply try $f(c) = c^2$

Then $f'(c) = 2c$ and $\frac{1}{4}(f'(c))^2 = c^2$ So it works!

So $$V(c,T) = c^2 r(T)$$

Then $$f(c)\,r' = c^2 - \frac{1}{4}(f'(c))^2\, r^2$$

becomes

$$\dot{r}(T) = 1 - r^2(T) \qquad (4.7.5)$$

What is the initial condition r(0)?

r(0)=0 since V(c, 0) = 0.

This is a **Riccati equation** in the math textbook.

In mathematics, a **Riccati equation** in the narrowest sense is any first-order ordinary differential equation that is quadratic in the unknown function.

In other words, it is an equation of the form

$$\dot{y} = q_0(x) + q_1(x)y(x) + q_2(x)y^2(x)$$

Riccati equation

$$\dot{r}(T) = 1 - r^2(T)$$

r(0)=0

How to solve Riccati equation?

Let's try the magic Laplace Transform, we have

$$\mathrm{sR(s)} = 1/\mathrm{s} - R(s) * R(s)$$

where * denotes convolution.

Can we find R(s) from above equation?

We cannot obtain the solution R(s) from above equation.

This is a good example to show why Laplace Transform does not work for nonlinear ODE.

We will skip the details on how to solve the Riccati equation, as this is not the main point.

Let's use the results from Calculus of Variations directly.

We have already established in section 3.2 of lecture notes that

$$\min_{y} J(y) = \int_0^T [y^2 + \dot{y}^2]dt = c^2 \tanh(\mathrm{T}) = c^2 \frac{e^T - e^{-T}}{e^T + e^{-T}}$$

which means that $V(c,T) = c^2 \tanh(\mathrm{T})$ (4.8.1)

thus

$$\mathrm{r(T)} = \tanh(\mathrm{T}) = \frac{e^T - e^{-T}}{e^T + e^{-T}}$$

You can verify that it indeed satisfies the Riccati equation,

$\dot{r}(T) = 1 - r^2(T)$, with initial condition r(o)=0.

The formalism is particularly simple in the case where $T \to \infty$.

In this case, does the optimal value function V(c,T) depend upon T?

V(c,T) becomes V(c).

$$\mathrm{V}(c) = \min_{y} \int_0^{\infty} (\dot{y}^2 + y^2) dt.$$

Then the same formalism as above yields the relation

$$\mathrm{V}(c) = \min_{u} [(u^2 + c^2)\Delta + V(c + u\Delta)] + O(\Delta^2),$$

hence

$$0 = \min_{u} [(u^2 + c^2) + u\dot{V}(c)]$$

Take the derivative respect to u gives $2u + \dot{V}(c) = 0$

So

$$u = -\frac{\dot{V}(c)}{2},$$

$$\dot{V}^2 = 4c^2$$

$$\dot{V}^2 = 4c^2$$

So we have two possibilities:

$$\dot{V}(c) = 2c \quad \text{and} \quad \dot{V}(c) = -2c$$

With condition V(0)=0, easily we can obtain two possible solutions:

$$V(c) = c^2 \quad \text{and} \quad V(c) = -c^2$$

Which one is the correct solution?

Since $V(c) \geq 0$, we see that $V(c) = c^2$

Once the optimal value is obtained, the optimal policy can be easily obtained as

$$u = -\frac{\dot{V}(c)}{2} = -c$$

Since y(0)=c, so we have $u(0) = -y(0)$. At any time t, we will have the feedback control law:

$$u(t) = -y(t).$$

Is this the same feedback control law we have obtained by calculus of variations?

Yes!

Break

State-of-the-art control systems

Future Technology

From previous example, it is not clear what is the advantage of Dynamic programming. The resulting nonlinear PDE is more complicated than the Euler equation.

The calculus of variations is only applicable to continuous process.

Dynamic programming is not only useful for continuous process, but also powerful tool for discrete process.

Discrete Control Processes

$$y(n+1) = ay(n) + u(n), y(0) = c \qquad (4.10.1)$$

and suppose that the u(n), the control variables are chosen so that the quadratic form

$$J_N(y,u) = \sum_{n=0}^{N} (y^2(n) + u^2(n)) \qquad (4.10.2)$$

is minimized.

Since the minimum value of J(y, u) depends on c, the initial value, and N, the number of stages, we write the optimal value function as

$$V_N(c) = \min_{\{u(n)\}} J_N(y,u) \qquad (4.10.3)$$

Again we need to consider the cost associated with the initial control action u(0).

After u(0) is chosen, the new state of the system is y(1)=ac+u(0) . The cost function takes the form

$$c^2 + u^2(0) + \sum_{n=1}^{N} (y^2(n) + u^2(n)) . \qquad (4.12.1)$$

Total cost = short term cost + Long term cost

How to compute the long term cost?

What is the policy we should use after the first step?

Regardless of the choice of u(0), we proceed from the new state to minimize the remaining sum.

This is the trick for dynamic programming: when we compute the long term cost, we assume that we just take the optimal actions even though we have no idea about what the optimal policy is at this stage!

This is the power of algebra: we just take the problem as solved and find out the equation first.

What is the state at t=1?

$$y(1)=ay(0)+u(0)=ac+u(0)$$

How many steps are left to reach the final value after u(0) is chosen?

N-1

So the long term cost can be expressed as the optimal value starting from ac+u(0) with N-1 steps left.

$$\sum_{n=1}^{N}(y^2(n)+u^2(n))=V_{N-1}(ac+u(0)).$$

The total cost caused by the control action u(0):

$$c^2 + u^2(0) + V_{N-1}(ac + u(0)).$$

Now let's figure out what is the optimal action to take at time t=0.

The quantity u(0) is now to be chosen to minimize the total cost. Then

$$V_N(c) = \min_{u(0)}[c^2 + u^2(0) + V_{N-1}(ac + u(0))],$$

$$V_0(c) = c^2, N \geq 1.$$

To proceed further, we need to take a guess on the mathematical form of $V_N(c)$.

For the continuous case we have

$$V(c,T) = c^2\, tanh(T)$$

It is reasonable to guess that

$$V_N(c) = c^2 r_N$$

$$V_N(c) = \min_{u(0)}[c^2 + u^2(0) + V_{N-1}(ac + u(0))], \qquad (4.12.3)$$

Let $V_N(c) = c^2 r_N$

Then we have

$$c^2 r_N = \min_{u(0)}[c^2 + u^2(0) + (ac + u(0))^2 r_{N-1}], \qquad (4.12.4)$$

How to find out u(0) to minimize the righthand side?

The value of u(0) that minimizes is readily obtained by differentiation,

$$2u(0) + 2\,(ac + u(0))r_{N-1} = 0,$$

$$u(0) = \frac{-r_{N-1}ac}{1 + r_{N-1}} \qquad (4.12.5)$$

Using this value and a small amount of algebra in (4.12.4), we obtain the recurrence relation

$$r_N = (1 + a^2) - \frac{a^2}{(1 + r_{N-1})} \qquad (4.12.6)$$

$$r_0 = 1, N \geq 1$$

Once r_N is obtained from above equation, the input u(0) can be computed as

$$u(0) = \frac{-r_{N-1}ac}{1 + r_{N-1}}$$

Is this a feedback controller?

Note that the controller is in a feedback form since c=y(0). Overall, we have obtained the following optimal controller in a feedback fashion.

At each time t=k, measure the state variable y(k), and calculate

$$u(k) = \frac{-r_{N-k-1}ay(k)}{1 + r_{N-k-1}} \qquad (4.12.7)$$

It is noted that the gain of controller depends upon the time k and N.

For control process, we are particularly interested in the case where $N \to \infty$.

Let $r = \lim_{N\to\infty} r_N$. Then the recurrence equation

$$r_N = (1 + a^2) - \frac{a^2}{(1 + r_{N-1})}$$

Becomes

$$r = (1 + a^2) - \frac{a^2}{(1 + r)} \qquad (4.13.1)$$

r can be solved as the positive root of the above quadratic equation.

Observe that u(0), the initial decision, also converges as $N \to \infty$, and

$$\lim_{N\to\infty} u(0) = -\frac{rac}{1 + r} \qquad (4.13.2)$$

Another way to show this

If we formally consider the infinite process, then value function becomes V(c), and

$$V(c) = \min_{\{u(n)\}} \sum_{n=0}^{\infty} (y^2(n) + u^2(n)), \tag{4.13.3}$$

We see that

$$V(c) = \min_{u(0)} [c^2 + u^2(0) + V(ac + u(0)) \tag{4.13.4}$$

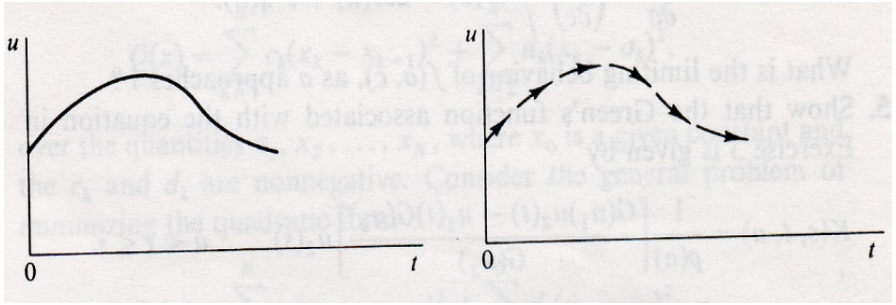
One solution of this equation is certainly $V(c) = rc^2$.

Therefore, for the infinite time process, the optimal feedback controller is simply,

$$u(k) = -\frac{ray(k)}{1 + r} \tag{4.13.5}$$

Comparison between calculus of Variations and Dynamic Programming

In the calculus of variations, we wish to minimize a functional J(u). The minimizing function is regarded as a point in a function space. In the dynamic programming approach, the tangent at a point (the control action at each state) is determined by the policy.



In the calculus of variations, the solution is the whole curve.

In dynamic programming, we consider the problem as multi-stage decision process and try to find out the optimal decision at every step.

The calculus of variations is applicable to continuous process only while dynamic programming can be applied to both continuous and discrete processes.

In summary, there are four steps involved in DP.

1. Try to translate the original optimization problem as a multi-stage decision process. Even though some problems do not involve any dynamic process, you can artificially break down the optimization problem into a sequence of decisions such that DP can be applied.

2. Define the optimal value function, V(x), where x represent all the information affecting the outcome. The value could be cost or utility, and the problem could be minimization (for cost) or maximization (for utility)

3. At state x, take any action u, and find out the immediate consequence associated with this action.

4. Assume that the rest of the actions will follow the **optimal policy**, which of course will depend upon the state driven by the current action. Then try to minimize (or maximize) the total value by choosing the best action, which will lead to the optimal cost. In this way, a recurrence relation will be derived to relate the current optimal value and the future value (immediately after the current action).

$$V(x) = \max_{u}[H(x,u,\mathrm{V}(\mathrm{T}(\mathrm{x},\mathrm{u})))] \qquad (4.15.1)$$

$$V(x) = \max_{u}[H(x,u,\mathrm{V}(\mathrm{T}(\mathrm{x},\mathrm{u})))]$$

The function T(x,u) describes the consequence resulting from a particular decision u, and H is the total value associated with this decision combined with the rest. In many case the total cost can be expressed as summation of short term cost and long term cost.

$$H(x,u,V(T(x,u))) = C(x,u) + V(T(x,u))$$

Total cost = short term cost + long term cost

The short term cost is the immediate cost associated with the action, and long term cost is the rest of the cost assuming the optimal policy is applied.

There are many applications which can be solved by dynamic programming. In the following, we will show one example.

The **knapsack problem** is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

The most common problem being solved is the **0-1 knapsack problem**, which restricts the number of copies of each kind of item to zero or one (either selected or not selected).

0-1 knapsack problem

Given a set of *n* items numbered from 1 up to *n*, each with a weight w_i and a value v_i, along with a maximum weight capacity *W*, we wish to determine a subset

$$S \subseteq \{1,2,\cdots n\} \qquad (4.15.2)$$

that maximizes

$$\sum_{i\in S} v_i \qquad (4.15.3)$$

subject to

$$\sum_{i\in S} w_i \leq W \qquad (4.15.4)$$

What are the possible solutions?

Of course, the exhaustive search (brutal force) is to try all 2^n possible subsets.

What is the approach by human common sense?

Greedy algorithm

The greedy algorithm is to rank the items by the values, and place the items one by one from the top value to the lower ones until the bag is full. It is fast and very often gives a good solution, but there is no guarantee to get the optimal solution.

Question: Any better solution?

Yes–Dynamic programming (DP)!

Does this knapsack problem involve any dynamic process?

No.

The first step in DP is to translate the problem into a multi-stage decision process.

How to do it?

The knapsack problem itself seems a static process. But we will artificially imbue with a time-like property by requiring the allocations to be made one at a time. We make decisions on one item at a time, and then force it to be a multi-stage decision process.

Viewed in this fashion, we have a dynamic process.
That is why it is called dynamic programming!

Once we have a multi-stage decision process, the next step, also the most important step in DP, is to define the optimal value function V. And then the rest is to find out V using principle of optimality.

What are the factors affecting the optimal value?

The set of items and the total capacity.

Therefore, let's define the value function

$$V(i,w), \quad 1 \le i \le n \quad 0 \le w \le W \qquad (4.15.5)$$

as the maximal value of the bag with the available set of {1,2,3,…,i}, and the capacity of w.

We need to find out this function for all possible i and w. For many applications, the arguments are discrete type, like the knapsack problem. Then we need to use a table for storing all the outcomes.

The key is to use dynamic programming to form the recurrence relation such that all the entries can be computed.

Let's do it step by step

When i=1, how do we find out V(1,w)?

Can we simply put V(1,w)= v_1?

We can easily find out V(1,w) for $0 \leq w \leq W$ as follows

$$V(1,w) = \begin{cases} 0, & w_1 > w \\ v_1, & w_1 \leq w \end{cases} \qquad (4.15.6)$$

How do we find out V(2,w) for all $0 \leq w \leq W$?

Now there are two items {1,2} available, it appears that we need to make a decision on each of them. However, since we already know V(1,w), it turns out that we only need to make decision on item 2, which significantly reduces the computing time.

That is the reason DP is much better than exhaustive search! You only need to take one action at a time and simply assume the rest are optimal!

How many choices do we have on item 2?

Since it is just 1 or 0, there are only two choices for item 2, select it or not. So just need to compute the value for each of them and pick the bigger one.

If item 2 is confirmed to be selected, what is the total value?

If item 2 is confirmed to be selected, then the total capacity of the bag reduces from w to $w - w_2$, and the available set for selection is the set of items {1}, for which we already have the optimal value of $V(1, w - w_2)$, therefore the total value is $V(1, w - w_2) + v_2$.

If item 2 is not selected, what is the total value?

If item 2 is not selected, then the optimal value is simply V(1,w).

So we just select the bigger one as

$$\max(\mathrm{V}(1, w), V(1, w - w_2) + v_2)$$

Overall, we have

$$V(2, w) = \begin{cases} V(1, w) & if \quad 0 \le w < w_2 \\ \max(\mathrm{V}(1, w), V(1, w - w_2) + v_2) & if \quad w_2 \le w < W \end{cases}$$

Following the same argument, we can easily get the recurrence relation between V(i-1,w) and V(i,w). Assuming we have the complete information about the subset {1,2,…, i-1} with the optimal value V(i-1,w), how to compute V(i,w)?

Again, we only need to take action on item i!

How many choices do we have?

There are only two choices for item i, select it or not.

If item i is not selected, what is the total value?

If not selected, then the value is still V(i-1,w).

If item i is selected, what is the total value?

If selected, then the totally capacity of the bag reduces from w to $w - w_i$, and the available set for selection is the subset of items {1,2,..,i-1}, for which we already have the optimal value of $V(i-1, w-w_i)$, therefore the total value is $V(i-1, w-w_i) + v_i$.

Overall, we have the recurrence equation to compute $V(i, w)$ from $V(i-1, w)$,

$$V(i,w) = \begin{cases} V(i-1,w) & if \quad 0 \le w < w_i \\ max(V(i-1,w), V(i-1,w-w_i) + v_i) & if \quad w_i \le w \le W \end{cases}$$

Tabulating the results from $V(i, 1)$ up through $V(i, W)$ gives the solution.

This solution will therefore run in $O(n\,W)$ time and $O(n\,W)$ space. Compare to the brutal force in the order of 2^n in time, the computational load is significantly reduced for large n.

After the optimal value function is found, how to use it to make the decision?

Once the optimal value function V(i,w) is found, it is very easy to find out whether item i should be chosen or not by working backwards from item n to item 1. For instance, if you want to decide whether item n should be included or not, you just compare V(n,W) and V(n-1,W) and use the following decision rule:

Item n should be selected if $V(n, W) > V(n-1, W)$

For the next item n-1, it will depend upon the choice on item n.

If item n is confirmed to be selected, then the total packing space reduces from W to $W - w_n$. Then using similar argument, you just compare $V(n-1, W - w_n)$ and $V(n-2, W - w_n)$, and use the decision rule that

Item n-1 should be selected if $V(n-1, W - w_n) > V(n-2, W - w_n)$

If item n is confirmed not to be selected, the total packing space remains as W. Then use the decision rule that

Item n-1 should be selected if $V(n-1, W) > V(n-2, W)$

Overall, to check whether item i should be selected, you need to know the choices of all the items from i+1 to n, and obtain the total packing space after packing the items i+1 to n as W_i, and then decide

Item i should be selected if $V(i, W_i) > V(i-1, W_i)$

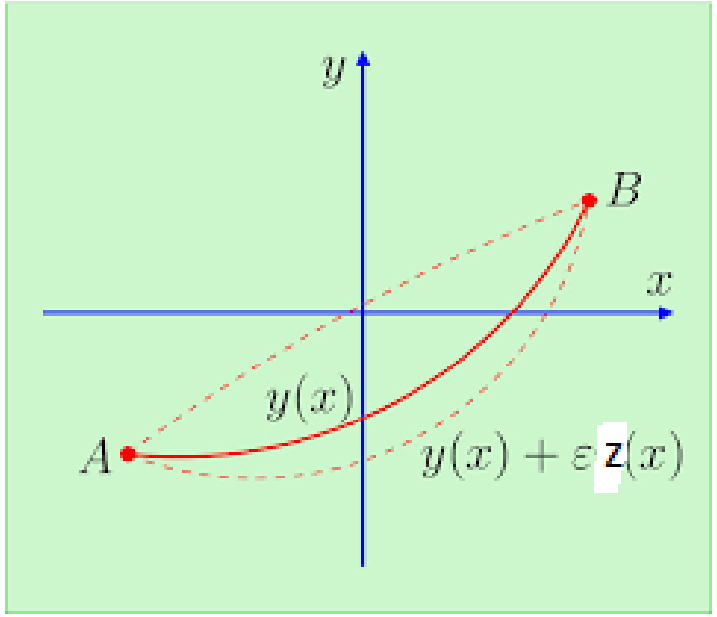
In summary, both calculus of variations and dynamic programming can be applied for solving optimization problems in continuous-time space, while dynamic programming is more suitable for discrete-time systems.

For calculus of variations, the key is to compute the variation of the value around the optimal solution:

$$J(y + \delta y)$$

How to express the change of the function δy?

$$\boldsymbol{\delta y = \varepsilon z}$$



For Dynamic Programming, the key is to formulate the problem as a multi-stage decision process and define the value function $V(x)$, and then use the principle of optimality.

You only need to consider one action at the current step and assume the rest of the actions are all optimal! Try to form a recurrence equation for the optimal value function.

Remember: if the optimization problem can be cast as a multistage decision process, dynamic programming should be the first choice of your weapon to attack the problem.

Q & A...