# Contents

# Chapter 1

# Integrator

## 1.1 Basic Knowledge

For the matrix exponential we have

$$e^A = \sum_{i=0}^{\infty} \frac{A^i}{i!} \tag{1.1}$$

$$(e^A)' = e^{A'} \tag{1.2}$$

$$\frac{d(e^{Ax})}{dx} = e^{Ax}A \tag{1.3}$$

The solution of linear differential equation

$$\dot{x} = Ax \tag{1.4}$$

is

$$x = e^{At}x_0 \tag{1.5}$$

For 2D system, we have the properties

| **Eigenvalue** | **Property** |
|---|---|
| real, positive | unstable node |
| real, negative | stable node |
| real, one positive, one negative | saddle |
| complex, positive real part | unstable spiral sink |
| complex, negative real part | stable spiral source |
| pure imaginary | center |
| real, positive, equal | unstable focus or improper node |
| real, negative, equal | stable focus or improper node |

When all eigenvalues have positive real part, it is called *source*; When all eigenvalues have negative real part, it is called *sink*;

## 1.2 Linear System with Uncertain Input

The problem is given the current reachabe space (face) $\mathcal{S}$ represented as `LP`

$$Px_0 \leq q \tag{1.6}$$

and an valid linearized model with uncertain input,

$$\dot{x} = Ax + u \tag{1.7}$$

where $u \in \mathcal{U}$ which is convex and compact, how to compute the forward reachable space $\delta_t(\mathcal{S})$ after time $t$?

Let us move froward each face separately. A face $\mathcal{F}$ of $\mathcal{S}$ is represented by its supporting hyper-plane:

$$\langle n, x \rangle = \langle n, y \rangle \tag{1.8}$$

where $\langle x, y \rangle$ denotes the inner product of $x$,$y$, $n$ is the outward normal to $\mathcal{F}$ and $y$ is an arbitrary point on the face[1].

The key idea is *there exists and input $u^* \in \mathcal{U}$ such that computing the successors of its supporting plane under $u^*$ is sufficient to derive a tight polyhedral approximation of forward face.*

It is proved that the evolution of the normal to $\mathcal{F}$ depends only on the linear term:

$$\dot{x} = -A'x \tag{1.9}$$

and the normal of forward face for any time $t$ is

$$n(t) = e^{-A't}n \tag{1.10}$$

It is clear that *the normal of forward face does not depend on the input* as shown in figure 1.1.

By integrating equation 1.7, we have

$$\begin{aligned}
x_u(t) &= e^{At}x_0 + \int_0^t e^{A(t-s)}u(s)\,ds \\
\langle n(t), x_u(t) \rangle &= \langle n(t), e^{At}x_0 \rangle + \left\langle n(t), \int_0^t e^{A(t-s)}u(s)\,ds \right\rangle \\
&= \left\langle e^{-A't}n, e^{At}x_0 \right\rangle + \dots \\
&= (e^{-A't}n)' \cdot e^{At}x_0 + \dots \\
&= n'e^{-At}e^{At}x_0 + \dots \\
&= \langle n, x_0 \rangle + \left\langle n(t), \int_0^t e^{A(t-s)}u(s)\,ds \right\rangle
\end{aligned}$$

[1] $n = P(i,:)'$ and $\langle n, y \rangle = q_i$

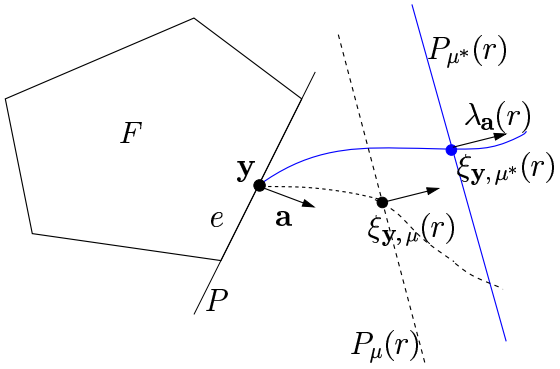


Figure 1.1: Forward faces with different inputs

similarly

$$\langle n(t), y_u(t)\rangle \quad = \quad \langle n, y_0\rangle + \left\langle n(t), \int_0^t e^{A(t-s)}u(s)\,ds \right\rangle$$

Any points $x_0 \in \mathcal{S}$ satisfies $\langle n, x_0\rangle \leq \langle n, y_0\rangle$, therefore, for all inputs $u \in \mathcal{U}$, for all $x_0 \in \mathcal{S}$, and for all $t \geq 0$ we have:

$$\langle n(t), x_u(t)\rangle \leq \langle n(t), y_u(t)\rangle$$

For an given an input $u$, the forward space $\delta_t^u(\mathcal{S})$ is bounded by the forward face $\delta_t^u(\mathcal{F})$:

$$\begin{aligned}\langle n(t), x\rangle &= \langle n(t), y_u(t)\rangle \\ &= \langle n, y_0\rangle + \left\langle n(t), \int_0^t e^{A(t-s)}u(s)\,ds \right\rangle\end{aligned}$$

It remarks that the normal is independent of input, therefore for all $u \in \mathcal{U}$ the forward face $\delta_t^u(\mathcal{F})$ are parallel to each other as shown in figure 1.1.

Now, the problem is how to find the *critical input* $u^*$ to maximize

$$\begin{aligned}&\left\langle n(t), \int_0^t e^{A(t-s)}u(s)\,ds \right\rangle \\ = &\int_0^t \left\langle e^{-A't}n, e^{At}e^{-As}u(s)\right\rangle\, ds \\ = &\int_0^t \left\langle n, e^{-As}u(s)\right\rangle\, ds \\ = &\int_0^t \left\langle e^{-A's}n, u(s)\right\rangle\, ds \\ = &\int_0^t \langle n(s), u(s)\rangle\, ds\end{aligned}$$

Obviously, the solution is

$$u^*(t) \quad \in \quad argmax\{\langle n(t), u\rangle \,|u \in \mathcal{U}\} \tag{1.11}$$

which is the optimal point of $\mathcal{U}$ with the optimal direction $n(t)$ as shown in figure 1.2.

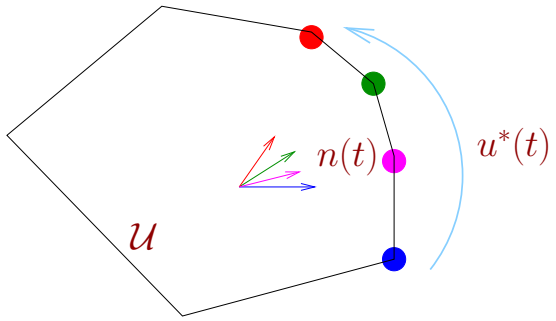


Figure 1.2: The critical input

With the critical input, the most outward forward face is

$$\begin{aligned} \delta_t(\mathcal{F}) &= \langle n, y_0\rangle + \int_0^t \langle n(s), u^*(s)\rangle \; ds \\ &= \langle n, y_0\rangle + \int_0^t (n(t)' u^*(s)) \; ds \end{aligned} \tag{1.12}$$

Finally, we have

$$\delta_t(\mathcal{S}) \subseteq \bigcap_{i=1}^{m} \delta_t(\mathcal{F}) \tag{1.13}$$

Of course, the result is over approximated because the input $u$ may not maximize all $\langle n(t), u\rangle$ for all faces. (It is not over approximated, because we have to contain all possisble reachable space for all possible inputs).

## 1.3 Algorithm

As shown above, the only problem is how to find the critical input $u^*(t)$ to compute

$$\int_0^t \langle n(t), u^*(s)\rangle \; ds \quad = \quad \int_0^t (n(t)' u^*(s)) \, ds. \tag{1.14}$$

We know $u^*(t)$ is the optimal point of $\mathcal{U}$ on $n(t)$ direction. $\mathcal{U}$ is convex polyhedron described by `LP`, the optimal point *jumps* rather than always change

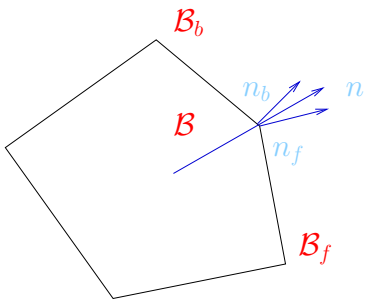


Figure 1.3: Critical Time to Force a New Pivot

with $t$. The difficult problem is find the *critical time* $t_i$ when the optimal direction $n(t)$ makes the optimal point (basis) change. Then the optimal point in $t_i, t_{i+1}$ can be found using LP solver. Thus, the critical input is determined.

We know for an optimal basis $\mathcal{B}$, its corresponding optimal direction is inside an cone defined as :

$$\lambda_1 n_1 + \ldots + \lambda_d n_d \quad \lambda_i \geq 0, \forall i \in [1, \ldots, d]$$

where $n_d$ is the normal (coefficients) of the $d^{th}$ half-plane (constraint) of $\mathcal{B}$. A 2D example is shown in figure 1.3.

The critical time to exit the core is when $n(t)$ is on the boundary of the core.

$$\begin{aligned} e^{-A't}n &= \sum_{i=1}^{d} \lambda_i n_i^{\mathcal{B}} \qquad (1.15) \\ \prod_{i=1}^{d} \lambda_i &= 0 \\ \lambda_i &\geq 0 \end{aligned}$$

The framework of our algorithm is shown in algorithm 1.

Solving equation 1.15 is nontrivial [2]. Of course, we can use numerical methods to approximate it.

Using Taylor approximation as shown in equation 1.1, equation 1.14 can be rewritten as

$$\begin{aligned} \int_0^t (n(t)'u^*(s))\,ds &= \int_0^t n' \sum_{i=0}^{\infty} \frac{(-At)^i}{i!} u^*(s)\,ds \\ &= \int_0^t n' \sum_{i=1}^{k} \frac{(-At)^k}{k!} u^*(s)\,ds + \int_0^t n' \sum_{i=k+1}^{\infty} \frac{(-At)^k}{k!} u^*(s)\,ds \end{aligned}$$

[2] I do not know how to get the close form solution

**Algorithm 1**: The Integral algorithm

**Input**: An linear system $\dot{x} = Ax + u(t)$, a half plane with normal $n$, and time step $T$

**Output**: $\int_0^T (n' e^{-As} u^*(s))\, ds$

```
t = 0, int = 0;
while t ≤ T do
    n(t) = e^{-A't} n;
    [pt, B] = lp_solve(U, n(t));
    ct = critical_time(n, B, A);
    nt = min(T, min(ct));
    int = int + ∫_t^{nt} (n' e^{-As} · pt) ds;
    t = nt;
```

$$\leq \int_0^t n' \sum_{i=1}^{k} \frac{(-At)^k}{k!} u^*(s)\, ds + \epsilon \tag{1.16}$$

The critical input $u^*(t)$ to maximize equation 1.16 has better properties. Equation 1.15 is now

$$\begin{aligned} \left( \sum_{i=0}^{k} \frac{(-A't)^k}{k!} \right) n &= \sum_{i=1}^{d} \lambda_i n_i^{\mathcal{B}} \\ \prod_{i=1}^{d} \lambda_i &= 0 \\ \lambda_i &\geq 0 \end{aligned} \tag{1.17}$$

The polynomial function is easier to solve.

Usually, the maximum eigenvalue of $At$ is less than 1 in Coho, thus, $k = 3, 4$ is accurate enough.

## 1.4 Special Cases

### 1.4.1 Linear System

When the input $u$ is a constant term $b$, the dynamics is

$$\dot{x} = Ax + b \tag{1.18}$$

The solution of forward face is

$$\begin{aligned} n' e^{-At} x &\leq n' y_0 + n'(I - e^{-At}) A^{-1} b \\ P(i,:) e^{-At} x &\leq q_i + P(i,:)(I - e^{-At}) A^{-1} b \end{aligned}$$

thus, the forward reachable space is in

$$Pe^{-At} x \leq q + P(I - e^{-At}) A^{-1} b \tag{1.19}$$

### 1.4.2 Linear Inclusion

The dynamics is

$$\dot{x} \quad = \quad Ax + b \pm u \tag{1.20}$$

$\mathcal{U}$ is a hyper-rectangle, the optimal point on any direction $d$ is $sign(d)u$. Thus, the critical input is

$$u^*(t) \quad = \quad sign(n(t)) \cdot * u \tag{1.21}$$

The critical time when any element of $n(t)$ changes its signum, which can be solved using numerical methods. Of course, we can also approximate $n(t)$ and find the critical time by solving

$$\sum_{i=0}^{k} \left( \frac{(-A')^i}{i!} n \right) t^k = 0. \tag{1.22}$$

The time $t$ which satisfies any equation of equation 1.22 is an critical time.

Then the integral of equation 1.14 is

$$\begin{aligned}
& \int_0^t (n(s)' \cdot (sign(n(s)) \cdot * u))\, ds \\
= & \int_0^{t_1} (n(s)' \cdot (sign(n) \cdot * e))\ ds + \\
& \int_{t_1}^{t2} (n(s)' \cdot (sign(n(t_1)) \cdot * e))\, ds + \\
& \cdots \\
& \int_{t_k}^{t} (n(s)' \cdot (sign(n(t_k)) \cdot * e))\, ds \\
= & \ n'(I - e^{-At_1}) A^{-1} (sign(n) \cdot * e) + \\
& \ n'(e^{-At_1} - e^{-At_2}) A^{-1} (sign(n(t_1)) \cdot * e) + \\
& \cdots \\
& \ n'(e^{-At_k} - e^{-At}) A^{-1} (sign(n(t_k)) \cdot * e)
\end{aligned}$$

On the other hand, it has a lower bound

$$\begin{aligned}
& \int_0^t \langle n(s), u^*(s) \rangle\ ds \\
= & \int_0^t \langle n(s), sign\,(n(s)) \cdot * u \rangle\ ds \\
= & \int_0^t \langle |n(s)|\,, u \rangle\ ds \\
= & \int_0^t (|n(s)'| \cdot u)\ ds
\end{aligned}$$

$$\begin{aligned} &\geq \left| \int_0^t (n(s)' \cdot u)\, ds \right| \\ &= \left| n'(I - e^{-At})A^{-1}u \right| \\ &\leq \left| n'(I - e^{-At})A^{-1} \right| u \end{aligned}$$

where $|A|$ denotes the element absolute function. However, $\left| n'(I - e^{-At})A^{-1} \right| u$, the approximation method we used before, is neither a lower bound nor a upper bound.

Thus, the forward reachable space $\delta_t(\mathcal{S})$ is

$$\begin{aligned} Pe^{-At}x \leq\ & q + P(I - e^{-At})A^{-1}b + \\ & P(I - e^{-At_1})A^{-1} \cdot * sign(P)u + \\ & \cdots \\ & P(e^{-At_m} - e^{-At})A^{-1} \cdot * sign(Pe^{-At_m})u \end{aligned} \tag{1.23}$$

Of course, when $t$ is tiny, $n(t)$ may have the same optimal point. The equation 1.23 is simplified as

$$\begin{aligned} \int_0^t (n(s)'u^*(s))\, ds &= n'(I - e^{-At})A^{-1}(sign(n) \cdot * u) \\ Pe^{-At}x &\leq q + P(I - e^{-At})A^{-1}b + P(I - e^{-At})A^{-1} \cdot * sign(\cdots) \end{aligned} \tag{1.24}$$

## 1.5 Implementation Issue

### 1.5.1 Numerical Computation

When $A$ is ill conditioned, $A^{-1}$ can not be computed by *inv* function in `matlab`. We solve the problem by

$$\begin{aligned} (e^{-At_0} - e^{-At_1})A^{-1} &= \left( \sum_{i=0}^{\infty} \frac{(-At_0)^i}{i!} - \sum_{i=0}^{\infty} \frac{(-At_1)^i}{i!} \right) A^{-1} \\ &= \sum_{i=0}^{\infty} \left( \left( \frac{(-At_0)^i}{i!} - \frac{(-At_1)^i}{i!} \right) A^{-1} \right) \\ &= \sum_{i=0}^{\infty} \left( \frac{(-A)^i(t_0^i - t_1^i)}{i!} A^{-1} \right) \\ &= \sum_{i=1}^{\infty} \frac{(-A)^{i-1}(t_1^i - t_0^i)}{i!} \\ (I - e^{-At})A^{-1} &= \sum_{i=1}^{\infty} \frac{(-A)^{i-1}t_1^i}{i!} \\ &= t_1 \sum_{i=1}^{\infty} \frac{(-At_i)^{i-1}}{i!} \end{aligned}$$

### 1.5.2 Experimental Result

We use one projectagon from passgate circuit as an example.

As shown in figure 1.4, the maximum eigenvalue of $At$ is 0.7 on average and at most 0.85.

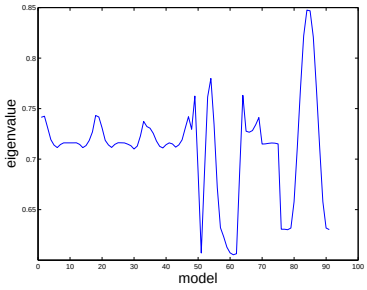


Figure 1.4: The maximum eigenvalue of $At$

We also find that usually $n(t)$ does not change sign within time $t$. Thus, the error between different methods (exact solution, numerical solution, previous approximation and lower bound) is small as shown in figure 1.5. Only the result of lower bound method is visible smaller than others. Thus, we use equation 1.24 as the default method[3].

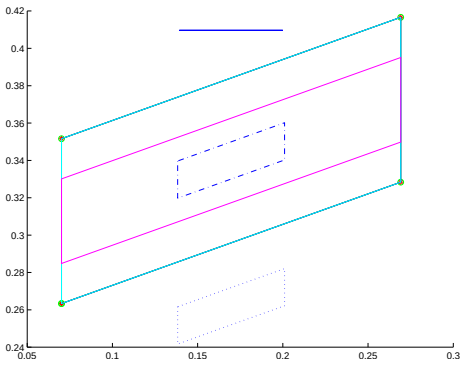

Figure 1.5: The result of different method

I think one main reason is that almost all edges are horizontal or vertical, thus 0 is already a critical time, it takes long time to change the normal by 90 degree.

[3]We use $sign(n(t/2)P)u$ as the critical input because 0 is usually a critical time

Another problem is how many polynomial terms should be use to approximate $e^{-At}$. Figure 1.6 shown the critical time found by using different number of polynomial terms. We find there are *fake* critical time using the approximation. However, when the number of terms increase, the *fake* critical time is larger thus the error is small. Usually, 2 or 3 terms is enough.

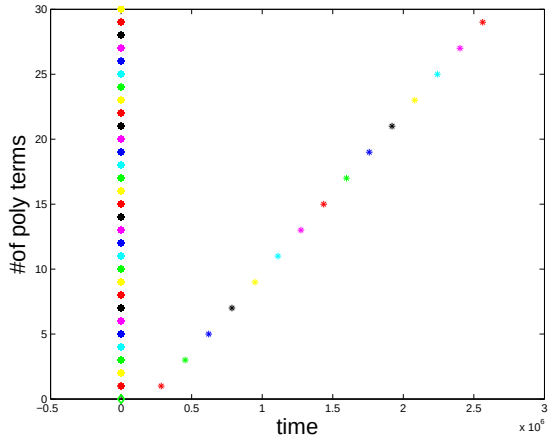


Figure 1.6: The critical time and number of polynomial terms

We also found the the approximated polynomial does not always find the critical times as shown in figure 1.7, where the green diamond is the critical time found by numerical method. Thus, the numerical method to find critical points is better than `roots` function which usually find complex roots.

### 1.5.3 Integration error

It is weird that even when $\dot{x} = Ax + b \pm u$ is always negative for all reachable space,i the projected polygon might increase, as shown in figure 1.8.

First, we compute the forward face with different time step $t$. We can see, at the beginning, $x$ is decreasing, however, it turns back when time step is larger.

Second, to analyze the reason, we remove the error term (shown as dotted polygon) and constant term (shown as the dashed polygon) from the inclusion. For the linear system, $x$ is always decreasing. The constant term only shifts the polygon. However, the error term bloat the polygon outward almost linearly as time advances. Thus, when the $\dot{x} = Ax$ decrease $x$ slower than linear speed, the error term catch up and $x$ value jumps back.

When $x$ is around 0.2 or above, $\dot{x}$ might be positive, thus $\dot{x} = Ax + b$ make the value of $x$ stop to decrease. However, the error term still bloat the polygon outward. Just imagine the time step is huge, thus the error term can destroy any progress made by the linear system.

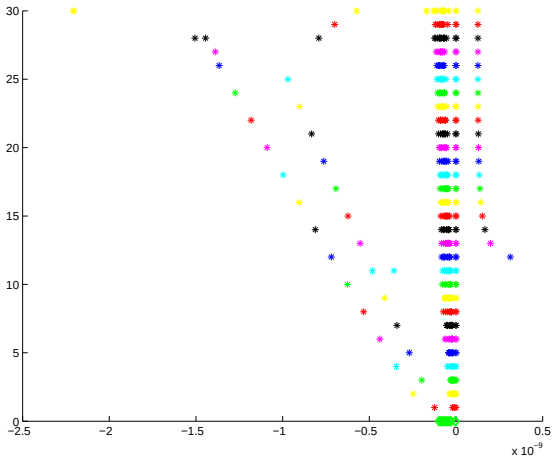


Figure 1.7: The critical time and number of polynomial terms

This is cause by the integration approximation error because the input can not move all faces outmost at the same time. Thus, the time step can not too large, otherwise the integration error is huge.

Then how about move several tiny steps without projection to replace one large step? `It is not helpful, the result are the same.` Combined with simulation?

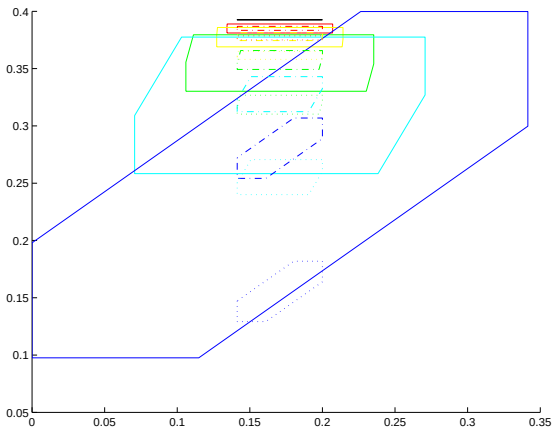


Figure 1.8: The integration of linear inclusion