FDU 数值算法 7. 矩阵函数

本文根据邵老师授课内容整理而成，并参考了以下教材:

- Matrix Analysis (R. Horn & C. Johnson) Chapter 3
- 矩阵分析 (R. Horn & C. Johnson) 第 3 章
- Matrix Computations (3rd Edition, G. Golub & C. Van Loan) Chapter 11
- 矩阵计算 (第 3 版, G. Golub & C. Van Loan) 第 11 章

欢迎批评指正!

7.1 矩阵函数

7.1.1 基本定义

(复述 FDU 高等线性代数 2. 范数中的结论)
一般地，设 $A\in\mathbb{C}^{n\times n}$ 可对角化，$A=S\Lambda S^{-1}$，$\Lambda=\text{diag}(\lambda_1,\dots,\lambda_n)$
给定一个定义域包含 $\lambda_1,\dots,\lambda_n$ 的复值函数 f
我们定义**初等矩阵函数**为: $f(A):=Sf(\Lambda)S^{-1}=S\text{diag}\{f(\lambda_1),\dots,f(\lambda_n)\}S^{-1}$
可以证明这个定义与相似矩阵 S 的选择是无关的 (即不同的 S 定义出的 $f(A)$ 是一致的)
这表明可对角化的矩阵的初等矩阵函数具有良好的定义.

特殊地，如果 f 在包含 $\lambda_1,\dots,\lambda_n$ 的某个开域解析，则我们有:

$$\begin{aligned}f(A)&:=Sf(\Lambda)S^{-1}\\&=S\text{diag}\{f(\lambda_1),\dots,f(\lambda_n)\}S^{-1}\\&=S\{\frac{1}{2\pi i}\oint_\Gamma f(\lambda)(\lambda I-\Lambda)^{-1}d\lambda\}S^{-1}\\&=\frac{1}{2\pi i}\oint_\Gamma f(\lambda)S(\lambda I-\Lambda)^{-1}S^{-1}d\lambda\\&=\frac{1}{2\pi i}\oint_\Gamma f(\lambda)(\lambda I-A)^{-1}d\lambda\end{aligned}$$

其中 Γ 是一个包含 A 的所有特征值的正向围道.

在 $f(A)$ 作为初等矩阵函数 (而不是作为幂级数) 的定义中，我们对函数 f 的要求很少 (它不必是解析的)
但我们对矩阵要求的更多 (它必须是可对角化的)
不可对角化的矩阵的初等矩阵函数可以定义，但我们必须要求函数 f 的解析性.
此时我们有 $f(A):=\frac{1}{2\pi i}\oint_\Gamma f(\lambda)(\lambda I-A)^{-1}d\lambda$
其中 Γ 是一个包含 A 的所有特征值的正向围道.

若 $A\in\mathbb{C}^{n\times n}$ 可以对角化，且解析函数 $f(z)=\sum_{k=0}^\infty a_kz^k$ 是由一个收敛半径大于 $\rho(A)$ 的幂级数所定义的，
则可以证明 $f(A)$ 的初等矩阵函数定义与其幂级数定义是一致的.
具体来说，设 $A=S\Lambda S^{-1}$，则我们有:

$$\sum_{k=0}^\infty a_kA^k=\sum_{k=0}^\infty a_k(S\Lambda S^{-1})^k=S(\sum_{k=0}^\infty a_k\Lambda^k)S^{-1}=S\text{diag}\{\sum_{k=0}^\infty a_k\lambda_1^k,\dots,\sum_{k=0}^\infty a_k\lambda_n^k\}S^{-1}=S\text{diag}\{f(\lambda_1),\dots,f(\lambda_n)\}S^{-1}$$

对于特定的函数，矩阵函数可能有不同的定义方式.
例如矩阵指数函数 $e^{At}=\lim_{n\to\infty}(I+\frac{At}{n})^n$
对于不同的定义，我们只需证明它与基于解析函数的定义一致即可.

矩阵函数的可交换性:

$$f(A)g(A)=g(A)f(A)$$

例如 Cayley 变换中: (可用 Neumann 级数理解，待补充)

$$(I+A)(I-A)^{-1}=(I-A)^{-1}(I+A)$$

7.1.2 理论计算

下面我们使用 Jordan 标准型给出矩阵函数的计算原理:
设复方阵 $A \in \mathbb{C}^{n\times n}$ 的 Jordan 标准型为:

$$S^{-1}AS = J = \begin{bmatrix} J^{(1)}(\lambda_1) & & & \\ & J^{(2)}(\lambda_2) & & \\ & & \ddots & \\ & & & J^{(d)}(\lambda_d) \end{bmatrix}$$

其中 $\lambda_1,\dots,\lambda_d$ 互不相同，$J_1(\lambda_1),\dots,J_d(\lambda_d)$ 为对应的 Jordan 矩阵:

$$\begin{cases} J^{(1)}(\lambda_1) = J_{n_1^{(1)}}(\lambda_1) \oplus \cdots \oplus J_{n_1^{(p_1)}}(\lambda_1) \text{ where } n_1 = \sum_{i=1}^{p_1} n_1^{(i)} & (n_1^{(1)} \geq \cdots \geq n_1^{(p_1)} \geq 1) \\ \quad\cdots \\ J^{(d)}(\lambda_d) = J_{n_d^{(1)}}(\lambda_d) \oplus \cdots \oplus J_{n_d^{(p_d)}}(\lambda_d) \text{ where } n_d = \sum_{i=1}^{p_d} n_d^{(i)} & (n_d^{(1)} \geq \cdots \geq n_d^{(p_d)} \geq 1) \end{cases}$$

给定解析的标量函数 f，我们定义矩阵函数 $f(A)$ 为:
(这个定义与相似变换矩阵 $S \in \mathbb{C}^{n\times n}$ 的选取无关)

$$f(A) := Sf(J)S^{-1}$$
$$f(J) := f(J^{(1)}(\lambda_1)) \oplus \cdots \oplus f(J^{(d)}(\lambda_d))$$
$$\begin{cases} f(J^{(1)}(\lambda_1)) = f(J_{n_1^{(1)}}(\lambda_1)) \oplus \cdots \oplus f(J_{n_1^{(p_1)}}(\lambda_1)) \\ \qquad\qquad\cdots \\ f(J^{(d)}(\lambda_d)) = f(J_{n_d^{(1)}}(\lambda_d)) \oplus \cdots \oplus f(J_{n_d^{(p_d)}}(\lambda_d)) \end{cases}$$

因此问题归结为如何计算关于特征值 λ 的 m 阶 Jordan 块 $J_m(\lambda)$ 的矩阵函数 $f(J_m(\lambda))$
考虑函数 f 在 $x=\lambda$ 处的 Taylor 展开式:

$$f(x) = f(\lambda) + \sum_{k=1}^{\infty} \frac{1}{k!} f^{(k)}(\lambda)(x-\lambda)^k$$

我们可以类似地写出矩阵函数 $f(J_m(\lambda))$ 的展开式:

$$\begin{aligned} f(J_m(\lambda)) &= f(\lambda)I_m + \sum_{k=1}^{\infty} \frac{1}{k!} f^{(k)}(\lambda)(J_m(\lambda)-\lambda I_m)^k \\ &= f(\lambda)I_m + \sum_{k=1}^{\infty} \frac{1}{k!} f^{(k)}(J_m(0))^k \end{aligned}$$

回忆起幂零 Jordan 块的性质:
(幂零 Jordan 块的性质, Matrix Analysis 引理 $3.1.4$)
给定正整数 $n \geq 2$ 和 $x \in \mathbb{C}^n$，记 $\mathbb{C}^n$ 的第 i 个标准单位基向量为 e_i，则我们有:

- $J_n(0)e_{i+1} = e_i \ (\forall\, i = 1,\dots,n-1)$
- $J_n(0)^{\mathrm{T}} J_n(0) = \begin{bmatrix} 0 & \\ & I_{n-1} \end{bmatrix}$
- $[I_n - J_n(0)^{\mathrm{T}} J_n(0)]x = (x^{\mathrm{T}} e_1)e_1$
- $J_n(0)^k = \begin{cases} I_n & \text{if } k=0 \\ \begin{bmatrix} & I_{n-k} \\ 0_{k\times k} & \end{bmatrix} & \text{if } 1 \leq k < n \\ 0_{n\times n} & \text{if } k \geq n \end{cases}$

我们有:

$$
\begin{aligned}
f(J_m(\lambda)) &= f(\lambda)I_m + \sum_{k=1}^{\infty}\frac{1}{k!}f^{(k)}(\lambda)(J_m(\lambda)-\lambda I_m)^k \\
&= f(\lambda)I_m + \sum_{k=1}^{\infty}\frac{1}{k!}f^{(k)}(\lambda)(J_m(0))^k \\
&= f(\lambda)I_m + \sum_{k=1}^{m-1}\frac{1}{k!}f^{(k)}(\lambda)(J_m(0))^k \\
&= f(\lambda)I_m + \sum_{k=1}^{m-1}\frac{1}{k!}f^{(k)}(\lambda)\begin{bmatrix} & I_{m-k} \\ 0_{k\times k} & \end{bmatrix} \\
&= \begin{bmatrix}
f(\lambda) & f'(\lambda) & \frac{1}{2!}f''(\lambda) & \cdots & \frac{1}{(m-1)!}f^{(m-1)}(\lambda) \\
 & f(\lambda) & f'(\lambda) & \ddots & \vdots \\
 & & \ddots & \ddots & \frac{1}{2!}f''(\lambda) \\
 & & & f(\lambda) & f'(\lambda) \\
 & & & & f(\lambda)
\end{bmatrix}
\end{aligned}
$$

(Matrix Computation 定理 $11.1.1$ 提供的另一种证明)
考虑计算关于特征值 λ 的 m 阶 Jordan 块 $J_m(\lambda)$ 的矩阵函数 $f(J_m(\lambda))$
假设 $(zI-J_m(\lambda))$ 非奇异，则我们有:

$$
\begin{aligned}
(zI-J_m(\lambda))^{-1} &= \sum_{k=0}^{\infty}\frac{(J_m(\lambda)-\lambda I_m)^k}{(z-\lambda)^{k-1}} \\
&= \sum_{k=0}^{\infty}\frac{(J_m(0))^k}{(z-\lambda)^{k-1}} \quad (\text{note that } (J_m(0))^k = \begin{cases} I_m & \text{if } k=0 \\ \begin{bmatrix} & I_{m-k} \\ 0_{k\times k} & \end{bmatrix} & \text{if } 1\le k<m) \\ 0_{m\times m} & \text{if } k\ge m \end{cases} \\
&= \sum_{k=0}^{m-1}\frac{(J_m(0))^k}{(z-\lambda)^{k-1}}
\end{aligned}
$$

根据 **Cauchy 积分公式**可知:

$$
\begin{aligned}
f(J_m(\lambda)) &= \frac{1}{2\pi i}\oint_{\Gamma} f(z)(\lambda I-J_m(\lambda))^{-1}dz \\
&= \frac{1}{2\pi i}\oint_{\Gamma} f(z)\sum_{k=0}^{m-1}\frac{(J_m(0))^k}{(z-\lambda)^{k-1}}dz \\
&= \sum_{k=0}^{m-1}\left[\frac{1}{2\pi i}\oint_{\Gamma}\frac{f(z)}{(z-\lambda)^{k+1}}dz\right](J_m(0))^k \\
&= \sum_{k=0}^{m-1}\left[\frac{1}{k!}\cdot\frac{k!}{2\pi i}\oint_{\Gamma}\frac{f(z)}{(z-\lambda)^{k+1}}dz\right](J_m(0))^k \\
&= \sum_{k=0}^{m-1}\frac{f^{(k)}(\lambda)}{k!}(J_m(0))^k
\end{aligned}
$$

> **(Cauchy 积分公式, Complex Variables and Applications 第 54 节)**
> 设函数 f 在由一条**正向** (即逆时针方向) 的简单闭围道 C 围成的闭区域 $\mathrm{cl}(C)$ 上解析.
> 若 z_0 是 C 内部的任意一点，则我们有:
>
> $$f(z_0) = \frac{1}{2\pi i}\oint_C \frac{f(z)}{z-z_0}dz$$
>
> - Cauchy 积分公式告诉我们:
>   若函数 f 在一条由简单闭围道 C 围成的闭区域 $\mathrm{cl}(C)$ 上解析，
>   则 f 在 C 内部的取值完全由 f 在 C 上的取值所确定.
>
> **(Cauchy 积分公式的推广, Complex Variables and Applications 第 55 节)**
> 设函数 f 在由一条**正向** (即逆时针方向) 的简单闭围道 C 围成的闭区域 $\mathrm{cl}(C)$ 上解析.
> 若 z_0 是 C 内部的任意一点，则我们有:
>
> $$f^{(k)}(z_0) = \frac{k!}{2\pi i}\oint_C \frac{f(z)}{(z-z_0)^{k+1}}dz \ \ (k=0,1,\dots)$$

7.1.3 数值稳定性

理论计算的结果体现了矩阵函数 $f(A)$ 与 A 的特征系统 (Jordan 标准型) 之间的紧密联系.
然而不幸的是，利用 Jordan 标准型计算矩阵函数是数值不稳定的.
一方面，Jordan 标准型的抗干扰能力相当差 (部分原因是秩的数值计算对扰动很敏感)，
另一方面，运算结果也会受到相似变换矩阵 S 病态程度的影响 (因为 S^{-1} 的存在要求我们求解涉及 S 的线性方程组)
具体来说，至少会有 $\kappa_2(S)\text{eps}$ 量级的舍入误差.
其中 $\kappa_2(S) := \|S\|_2\|S^{-1}\|_2 = \frac{\sigma_{\max}(S)}{\sigma_{\min}(S)}$，而 eps 代表机器精度.

下面的例子说明计算矩阵函数时应避免病态的相似变换:

$$A = \begin{bmatrix} 1+10^{-15} & 1 \\ & 1-10^{-15} \end{bmatrix}$$

易知其特征值为 $1+10^{-15}$ 和 $1-10^{-15}$，对应的特征向量为 $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ 和 $\begin{bmatrix} -1 \\ 2\times 10^{-15} \end{bmatrix}$

于是我们有:

$$A = S\Lambda S^{-1}$$
$$\text{where } S = \begin{bmatrix} 1 & -1 \\ 0 & 2\times 10^{-15} \end{bmatrix} \text{ and } \Lambda = \begin{bmatrix} 1+10^{-15} & \\ & 1-10^{-15} \end{bmatrix}$$

下面考虑计算 S 的条件数并估计舍入误差的量级:

$$S^{\mathrm{T}}S = \begin{bmatrix} 2 & -2\times 10^{-15} \\ -2\times 10^{-15} & 4\times 10^{-30} \end{bmatrix}$$

通过特征多项式解得 $S^{\mathrm{T}}S$ 的特征值约为 2 和 2×10^{-30}
可求得 $\kappa_2(S) = \frac{\sigma_{\max}(S)}{\sigma_{\min}(S)} \approx \frac{\sqrt{2}}{\sqrt{2\times 10^{-30}}} = 10^{15}$
由于 Matlab 的机器精度 $\text{eps} \approx 2.22\times 10^{-16}$，
故理论上舍入误差的量级约为 $\kappa_2(S)\text{eps} \approx 0.222$
我们使用 Matlab 代码进行验证:

```
A = [1+1e-15, 1;
     0, 1-1e-15];
S = [1, -1;
     0, 2e-15];
Lambda = diag([1+1e-15, 1-1e-15]);

% 验证谱分解
disp("A - S * Lambda * inv(S):")
disp(A - S * (Lambda / S));

% 机器精度
disp("eps of Matlab:");
disp(eps);

% 计算 f(A) = expm(A)
f_A = expm(A);  % 计算矩阵指数
disp('f(A) = exp(A):')
disp(f_A)

% 计算 S * f(Lambda) * S^{-1}
f_Lambda = diag([exp(Lambda(1,1)), exp(Lambda(2,2))]);  % 计算 Lambda 的指数
f_SLambdaS = S * (f_Lambda / S);  % 计算 S * f(Lambda) * S^{-1}
disp('S * f(Lambda) * S^{-1}:')
disp(f_SLambdaS)
```

运行结果:

```
A - S * Lambda * inv(S):
         0   -0.0625
         0         0
```

```
eps of Matlab:
   2.2204e-16

f(A) = exp(A):
    2.7183    2.7183
         0    2.7183

S * f(Lambda) * S^{-1}:
    2.7183    3.0000
         0    2.7183
```

我们发现 $(1,2)$ 位置上元素的误差约为 0.2827，与舍入误差量级的理论估计相符.

7.2 逼近法

本节我们讨论一类乍一看不涉及特征值的计算矩阵函数的方法.
这类方法的基本思想是:
若 $g(z)$ 在 $\mathrm{eig}(A)$ 上逼近 $f(z)$，则 $g(A)$ 逼近 $f(A)$，例如:

$$\begin{aligned} e^z &\approx 1+z+\frac{z^2}{2!}+\cdots+\frac{z^m}{m!} \\ e^A &\approx I+A+\frac{A^2}{2!}+\cdots+\frac{A^m}{m!} \end{aligned}$$

7.2.1 误差界

(Jordan 标准型导出的误差界, Matrix Computation 定理 $11.2.1$)
设复方阵 $A\in\mathbb{C}^{n\times n}$ 的 Jordan 标准型为:

$$S^{-1}AS = J = \begin{bmatrix} J^{(1)}(\lambda_1) & & & \\ & J^{(2)}(\lambda_2) & & \\ & & \ddots & \\ & & & J^{(d)}(\lambda_d) \end{bmatrix}$$

其中 $\lambda_1,\dots,\lambda_d$ 互不相同，$J_1(\lambda_1),\dots,J_d(\lambda_d)$ 为对应的 Jordan 矩阵:

$$\begin{cases} J^{(1)}(\lambda_1) = J_{n_1^{(1)}}(\lambda_1)\oplus\cdots\oplus J_{n_1^{(p_1)}}(\lambda_1) \text{ where } n_1=\sum_{i=1}^{p_1} n_1^{(i)} \quad (n_1^{(1)}\geq\cdots\geq n_1^{(p_1)}\geq 1) \\ \quad\cdots \\ J^{(d)}(\lambda_d) = J_{n_d^{(1)}}(\lambda_d)\oplus\cdots\oplus J_{n_d^{(p_d)}}(\lambda_d) \text{ where } n_d=\sum_{i=1}^{p_d} n_d^{(i)} \quad (n_d^{(1)}\geq\cdots\geq n_d^{(p_d)}\geq 1) \end{cases}$$

若标量函数 f,g 在包含 $\mathrm{eig}(A)$ 的某一开集上解析，则我们有:

$$\|f(A)-g(A)\|_2 \leq \kappa_2(S) \max_{\substack{1\leq i\leq d \\ 0\leq r\leq n_i^{(1)}-1}} \left\{ n_d^{(1)} \frac{|f^{(r)}(\lambda_i)-g^{(r)}(\lambda_i)|}{r!} \right\}$$

(Schur 型导出的误差界, Matrix Computation 定理 $11.2.2$)
设 $A\in\mathbb{C}^{n\times n}$ 的 Schur 分解为 $A=UTU^{\mathrm{H}}$
记 $T=\Lambda+N$ (其中 Λ 为 T 的对角部分，而 N 是 t 的严格上三角部分)
若 $f(z),g(z)$ 在一个内部包含 $\mathrm{eig}(A)$ 的闭凸集 Ω 上解析，则我们有:

$$\begin{aligned} \|f(A)-g(A)\|_{\mathrm{F}} &\leq \sum_{r=0}^{n-1} \delta_r \frac{\||N|^r\|_{\mathrm{F}}}{r!} \\ \text{where } \delta_r &:= \sup_{z\in\Omega} |f^{(r)}(z)-g^{(r)}(z)| \ (r=0,\dots,n-1) \end{aligned}$$

上述定理中的误差界表明，$g(z)$ 仅仅在 A 的谱 $\mathrm{eig}(A)$ 上逼近 $f(z)$ 是不够的.
特别地，如果 A 的特征矩阵是病态的或与正规矩阵的差距很大，
则 $f(A)$ 与 $g(A)$ 的差可能会比 $|f(z)-g(z)|$ 在 $\mathrm{eig}(A)$ 上的极大值大得多.
因此虽然逼近方法可以避免特征值的计算，但它会受到 A 的特征矩阵结构的影响.

7.2.2 矩阵求幂

要计算 A^{13}，只需要进行 5 次矩阵乘法即可:

$$\begin{aligned} A^2 &= A \cdot A \\ A^4 &= A^2 \cdot A^2 \\ A^8 &= A^4 \cdot A^4 \\ A^{13} &= A^8 \cdot A^4 \cdot A \end{aligned}$$

这称为**二进制求幂法** (Binary Powering):
(二进制求幂法, Matrix Computation 算法 $11.2.2$)
给定矩阵 $A \in \mathbb{R}^{n\times n}$ 和正整数 m，此算法计算幂 $F = A^m$

$$\begin{array}{l}
\text{function: } F = \text{Binary_Powering}(A, m) \\
\quad\quad \text{Let } m = \displaystyle\sum_{k=0}^{p} \beta_k 2^k \text{ be the binary expansion of } m \text{ (where } \beta_p = 1 \text{ and } \beta_k \in \{0,1\} \text{ for all } k = 0, \dots, p-1) \\
\quad\quad X = A \\
\quad\quad q = 0 \\
\quad\quad \text{while } \beta_q = 0 \\
\quad\quad\quad\quad X = X^2 \\
\quad\quad\quad\quad q = q + 1 \\
\quad\quad \text{end} \\
\quad\quad F = X \\
\quad\quad \text{for } k = q+1 : p \\
\quad\quad\quad\quad X = X^2 \\
\quad\quad\quad\quad \text{if } \beta_k \neq 0 \\
\quad\quad\quad\quad\quad\quad F = FX \\
\quad\quad\quad\quad \text{end} \\
\quad\quad \text{end} \\
\text{end}
\end{array}$$

这一算法至多需要 $2\lfloor \log_2(m) \rfloor$ 次矩阵乘法.
特殊地，若 m 是 2 的幂，则只需要 $\log_2(m)$ 次矩阵乘法.

7.2.3 矩阵多项式

在超越矩阵函数的逼近中经常涉及计算多项式.
考虑以下矩阵多项式:

$$p(A) = c_d A^d + c_{d-1} A^{d-1} + \cdots + c_1 A + c_0 I$$

其中 $c_0, c_1, \dots, c_{d-1}, c_d$ $(c_d \neq 0)$ 是给定的实数.

(秦九韶算法/Horner 技巧, Matrix Computation 算法 $11.2.1$)
给定 $A \in \mathbb{R}^{n\times n}$ 和 $c_0, c_1, \dots, c_{d-1}, c_d$ (用向量 $c \in \mathbb{R}^{d+1}$ 传入)
此算法计算多项式 $F = p(A) = c_d A^d + c_{d-1} A^{d-1} + \cdots + c_1 A + c_0 I$:

$$\begin{array}{l}
\text{function: } F = \text{Qin_Jiushao}(A, c) \\
\quad\quad F = c_d A + c_{d-1} I \\
\quad\quad \text{for } k = d-2 : -1 : 0 \\
\quad\quad\quad\quad F = AF + c_k I \\
\quad\quad \text{end} \\
\text{end}
\end{array}$$

上述算法需要 $d-1$ 次矩阵乘法.
但和标量的情形不同的是，上述求和过程并不是最优的，我们可以适当增加 "步长".
以 $d = 4$ 为例 (只需要 2 次矩阵乘法):

$$
\begin{aligned}
p(A) &= c_4A^4 + c_3A^3 + c_2A^2 + c_1A + c_0I_n \\
&= A^2(c_4A^2 + c_3A + c_2I_n) + c_1A + c_0I_n
\end{aligned}
$$

$$
\begin{aligned}
A_1 &= A \\
A_2 &= A^2 \\
F_1 &= c_4A_2 + c_3A_1 + c_2I_n \\
F &= A_2F_1 + c_1A_1 + c_0I_n
\end{aligned}
$$

一般来说，考虑计算 d 次矩阵多项式:

$$
p(A) = c_dA^d + c_{d-1}A^{d-1} + \cdots + c_1A + c_0I
$$

可取最优步长 $t = \lfloor \sqrt{d} \rfloor$ 并计算:

$$
\begin{aligned}
A_1 &= A \\
A_2 &= AA_1 \\
A_3 &= AA_2 \\
&\cdots \\
A_t &= AA_{t-1}
\end{aligned}
$$

这需要 $t-1$ 次矩阵乘法.
记 $r = \lfloor \frac{d}{t} \rfloor$ 并执行以下迭代:

$$
\begin{aligned}
&F = c_dA_{d-rt} + c_{d-1}A_{d-rt-1} + \cdots + c_{rt+1}A_1 + c_{rt}I \\
&\text{for } k = r-1 : -1 : 0 \\
&\qquad F = A_tF + (c_{kt+t-1}A_{t-1} + \cdots + c_{kt+1}A_1 + c_{kt}I) \\
&\text{end}
\end{aligned}
$$

这需要 r 次矩阵乘法.
因此总共需要 $r+t-1$ 次矩阵乘法.

7.2.4 Taylor 逼近

一种逼近矩阵函数 (例如 e^A) 的常用方法是使用截断的 Taylor 级数.
一个矩阵函数 $f(A)$ 存在 Taylor 级数展开的条件是很容易找到的.

(Matrix Computation, 定理 $11.2.3$)
若解析函数 $f(z) = \sum_{k=0}^{\infty} c_kz^k$ 是由一个收敛半径大于 $\rho(A)$ 的幂级数所定义的，则我们有:

$$
f(A) = \sum_{k=0}^{\infty} c_kA^k
$$

- **证明:**
  考虑关于 λ 的 p 阶 Jordan 块 $J_p(\lambda) = \lambda I_p + J_p(0)$，容易验证:

$$
\begin{aligned}
(J_p(\lambda))^m &= \sum_{k=0}^{\min\{p-1,m\}} \binom{m}{k} \lambda^{m-k} (J_p(0))^k \\
\text{where } J_p(0)^k &= \begin{cases} I_p & \text{if } k=0 \\ \begin{bmatrix} & I_{p-k} \\ 0_{k\times k} & \end{bmatrix} & \text{if } 1 \le k < p \\ 0_{p\times p} & \text{if } k \ge p \end{cases}
\end{aligned}
$$

  结合 $f(J_p(\lambda)) = \sum_{k=0}^{p-1} \frac{f^{(k)}(\lambda)}{k!}(J_m(0))^k$ 的结论可知:

$$
\begin{aligned}
\sum_{m=0}^{\infty} c_m (J_p(\lambda))^m &= \sum_{m=0}^{\infty} c_m \left\{ \sum_{k=0}^{\min\{p-1,m\}} \binom{m}{k} \lambda^{m-k} (J_p(0))^k \right\} \\
&= \sum_{k=0}^{p-1} \left\{ \sum_{m=k}^{\infty} c_m \binom{m}{k} \lambda^{m-k} \right\} (J_p(0))^k \\
&= \sum_{k=0}^{p-1} \left\{ \frac{1}{k!} \sum_{m=k}^{\infty} c_m \frac{m!}{(m-k)!} \lambda^{m-k} \right\} (J_p(0))^k \\
&= \sum_{k=0}^{p-1} \left\{ \frac{1}{k!} f^{(k)}(\lambda) \right\} (J_m(0))^k \\
&= f(J_p(\lambda))
\end{aligned}
$$

因此 $f(J_p(\lambda)) = \sum_{k=0}^{\infty} c_k (J_p(\lambda))^k$
于是对于 A 的 Jordan 标准型 $J = S^{-1}AS$ 我们有 $f(J) = \sum_{k=0}^{\infty} c_k J^k$
进而有 $f(A) = f(SJS^{-1}) = Sf(J)S^{-1} = S\{\sum_{k=0}^{\infty} c_k J^k\}S^{-1} = \sum_{k=0}^{\infty} c_k A^k$
命题得证.

几个重要的矩阵函数的 Taylor 级数展开:
(A^{-1} 我们通常不是按照矩阵函数的方法研究的，因此没有列在其中)

$$
\begin{aligned}
\exp(A) &= \sum_{k=0}^{\infty} \frac{A^k}{k!} \\
\cos(A) &= \sum_{k=0}^{\infty} (-1)^k \frac{A^{2k}}{(2k)!} \\
\sin(A) &= \sum_{k=0}^{\infty} (-1)^k \frac{A^{2k+1}}{(2k+1)!} \\
\log(I-A) &= \sum_{k=1}^{\infty} \frac{A^k}{k} \quad (\text{under the condition of } \rho(A) < 1) \\
(I-A)^{-1} &= \sum_{k=0}^{\infty} A^k \quad (\text{under the condition of } \rho(A) < 1) \\
A^{\frac{1}{2}} &= I - \sum_{k=1}^{\infty} \frac{1}{2^k} (I-A)^k \quad (\text{where } A \text{ is positive semi-definite}) \\
A^{-1}(e^A - I) &= \sum_{k=0}^{\infty} \frac{A^k}{(k+1)!}
\end{aligned}
$$

注意 Hermite 半正定阵 A 的 Hermite 半正定 p 次根 $A^{\frac{1}{p}}$ 是唯一的.

用截断的 Taylor 级数来逼近矩阵函数的一个缺点是只有在原点附近它才有意义.
某些情况下，我们可以通过变换可克服这一点.
例如计算 $\cos(A)$ 和 $\sin(A)$ 时我们可以重复应用倍角公式组装矩阵的余弦值和正弦值:

$$
\begin{aligned}
\cos(2A) &= 2(\cos(A))^2 - I \\
\sin(2A) &= 2\sin(A)\cos(A)
\end{aligned}
$$

这称为**缩放平方算法** (Scaling and Squaring)
其对于 $\cos(A)$ 和 $\sin(A)$ 的具体实现如下:

$$
\begin{aligned}
&m = \max(0, 1 + \lfloor \log_2(\|A\|_\infty) \rfloor) \\
&S_0 = \text{the Taylor approximation of } \sin(A/2^m) \\
&C_0 = \text{the Taylor approximation of } \cos(A/2^m) \\
&\text{for } j = 1 : m \\
&\qquad S_j = 2S_{j-1}C_{j-1} \\
&\qquad C_j = 2C_{j-1}^2 - I \\
&\text{end}
\end{aligned}
$$

对于 $\exp(A)$ 的具体实现如下:

$$
\begin{aligned}
&m = \max(0, 1 + \lfloor \log_2(\|A\|_\infty) \rfloor) \\
&E_0 = \text{the Taylor approximation of } \exp\left(A/2^m\right) \\
&\text{for } j = 1 : m \\
&\qquad E_j = E_{j-1}^2 \\
&\text{end}
\end{aligned}
$$

7.2.5 Padé 逼近

考虑矩阵指数函数 e^A 的计算 (Nineteen Dubious Ways to Compute the Exponential of a Matrix)
若 $g(z) \approx e^z$，则 $g(A) \approx e^A$
为达到这一目的，有一类非常有用的逼近函数——Padé 函数:

$$R_{p,q}(z) := (D_{p,q}(z))^{-1} N_{p,q}(z)$$

其中:

$$
\begin{aligned}
N_{p,q}(z) &= \sum_{k=0}^{p} \frac{(p+q-k)!}{(p+q)!} \binom{p}{k} z^k = \sum_{k=0}^{p} \frac{(p+q-k)!p!}{(p+q)!k!(p-k)!} z^k \\
D_{p,q}(z) &= \sum_{k=0}^{q} \frac{(p+q-k)!}{(p+q)!} \binom{q}{k} (-z)^k = \sum_{k=0}^{q} \frac{(p+q-k)!q!}{(p+q)!k!(q-k)!} (-z)^k
\end{aligned}
$$

特殊地，当 $q = 0$ 时即为 p 阶 Taylor 多项式:

$$R_{p,0}(z) = \sum_{k=0}^{p} \frac{z^k}{k!}$$

值得注意的是，Padé 逼近不仅仅适用于指数函数.
但与 Taylor 逼近不同的是，Padé 逼近函数有时不一定存在.

不幸的是，Padé 逼近仅在原点附近才非常好，下面的等式说明了这一点:

$$e^A = R_{p,q}(A) + \frac{(-1)^q}{(p+q)!} A^{p+q+1} (D_{p,q}(A))^{-1} \int_0^1 u^p (1-u)^q e^{A(1-u)} du$$

不过我们可以利用缩放平方算法 $e^A = (e^{\frac{A}{2^m}})^{2^m}$ 来克服这一困难.
整个算法的好坏取决于以下逼近的精度:

$$F_{p,q} = \left(R_{p,q}\left(\frac{A}{2^m}\right) \right)^{2^m}$$

可以证明:
若 $\frac{\|A\|_\infty}{2^m} \le \frac{1}{2}$，则存在矩阵 $\Delta A \in \mathbb{R}^{n \times n}$ 使得:

$$
\begin{aligned}
F_{p,q} &= e^{A + \Delta A} \\
A \Delta A &= \Delta A A \\
\|\Delta A\|_\infty &\le \varepsilon(p,q) \|A\|_\infty \\
\varepsilon(p,q) &= 2^{3-(p+q)} \frac{p!q!}{(p+q)!(p+q+1)!}
\end{aligned}
$$

进而可以证明不等式:

$$\frac{\|e^A - F_{p,q}\|_\infty}{\|e^A\|_\infty} \le \varepsilon(p,q) \|A\|_\infty \exp\{\varepsilon(p,q) \|A\|_\infty\}$$

参数 p, q 可根据所需的相对精度来确定.
鉴于计算 $F_{p,q}$ 大约需要 $m + \max(p,q)$ 次矩阵乘法，故我们最好令 $p = q$
上述结果构成了有效计算 e^A 并控制误差的方法的基础.

(Matrix Computation 算法 11.3.1)
给定矩阵 $A \in \mathbb{R}^{n \times n}$ 和 $\tau > 0$，此算法计算 $F = e^{A + \Delta A}$ (其中 $\|\Delta A\|_\infty \le \tau \|A\|_\infty$)

$m = \max(0, 1 + \lfloor \log_2(\|A\|_\infty) \rfloor)$
$A = A/2^m$
Let q be the smallest nonnegative integer such that $\varepsilon(q,q) \le \tau$
$N = I$
$D = I$
$X = I$
$c = 1$
for $k = 1 : q$
  $c = c(q-k+1)/[(2q-k+1)k]$
  $X = AX$
  $N = N + cX$
  $D = D + (-1)^k cX$
end
solve $DF = N$ for $F = D^{-1}N$ using Gaussian elimination
for $k = 1 : m$
  $F = F^2$
end

上述算法大约需要 $2(q+m+\frac{1}{3})n^3$ 的计算量.
此外我们还可利用 7.2.3 节的**改进的秦九韶算法**来加快矩阵多项式 $D = D_{q,q}(A)$ 和 $N = N_{q,q}(A)$ 的计算.

7.3 Schur-Parlett 算法

7.3.1 Parlett 递推

Shcur 型的数值稳定性要比 Jordan 标准型好得多.
利用 Schur 分解处理矩阵函数可避免用 Jordan 分解所带来的一些困难.
设 $A \in \mathbb{C}^{n\times n}$ 的 Schur 分解为 $A = UTU^{\mathrm{H}}$，则 $f(A) = Uf(T)U^{\mathrm{H}}$

注意到 T 和 $F := f(T)$ 可交换

> 这是因为 $F = f(T)$ 存在 Taylor 展开式 $F = f(T) = \sum_{k=0}^{\infty} f^{(k)}(0)T^k$
> 根据 Cayley-Hamilton 定理可知 A 高于 n 次的幂总能表示为 $I, A^1, \dots, A^{n-1}$ 的线性组合.
> 因此 $F = f(T) = \sum_{k=0}^{\infty} f^{(k)}(0)A^k$ 可以表示为 A 的次数不超过 $n-1$ 的多项式:
>
> $$F = f(T) = \sum_{k=0}^{\infty} f^{(k)}(0)A^k = \sum_{k=0}^{n-1} c_k A^k$$
>
> 所以 F 和 T 一定是可交换的.

比较等式 $TF = FT$ 两边 (i,j) (其中 $i<j$, 即严格上三角元) 位置上的元素可知:

$$\sum_{k=i}^{j} t_{ik}f_{kj} = \sum_{k=i}^{j} f_{ik}t_{kj}$$

$$\Leftrightarrow$$

$$t_{ii}f_{ij} + \sum_{k=i+1}^{j-1} t_{ik}f_{kj} + t_{ij}f_{jj} = f_{ii}t_{ij} + \sum_{k=i+1}^{j-1} f_{ik}t_{kj} + f_{ij}t_{jj}$$

$$\Leftrightarrow$$

$$(t_{ii} - t_{jj})f_{ij} = (f_{ii} - f_{jj})t_{ij} + \sum_{k=i+1}^{j-1}(f_{ik}t_{kj} - t_{ik}f_{kj})$$

因此如果 $t_{ii} \neq t_{jj}$ (即 T 的第 i,j 个特征值 λ_i, λ_j 不相同)，则我们有:

$$f_{ij} = \frac{f_{ii} - f_{jj}}{t_{ii} - t_{jj}} t_{ij} + \frac{1}{t_{ii} - t_{jj}} \sum_{k=i+1}^{j-1}(f_{ik}t_{kj} - t_{ik}f_{kj})$$

从中我们可以看出 f_{ij} 是它左侧元素 $\{f_{ik}\}_{k=i}^{j-1}$ 和下方元素 $\{f_{kj}\}_{k=i+1}^{j}$ 的组合.
(例如 f_{25} 依赖于 $f_{22}, f_{23}, f_{24}, f_{35}, f_{45}, f_{55}$ 的值)
因此在计算出 $f(T)$ 对角元 $f_{11}, f_{22}, \dots, f_{nn}$ 之后，
我们逐一计算每一条超对角线 (superdiagonal)，直至计算完整个严格上三角部分.

这样我们就得到了如下算法:

(Parlett 递推, Matrix Computation 算法 11.1.1**)**
给定对角元互不相同的上三角阵 $T \in \mathbb{C}^{n\times n}$，计算 $F = f(T)$:

$$\begin{aligned}
&\text{function: } F = \text{Parlett_Recursion}(T) \\
&\quad \text{for } i = 1:n \quad \text{(diagonal)} \\
&\quad\quad f_{ii} = f(t_{ii}) \\
&\quad \text{end} \\
&\quad \text{for } p = 1:n-1 \quad (p\text{-th superdiagonal}) \\
&\quad\quad \text{for } i: n-p \quad \text{(from bottom to top)} \\
&\quad\quad\quad j = i+p \\
&\quad\quad\quad \text{sum} = (f_{ii} - f_{ij})t_{ij} \\
&\quad\quad\quad \text{for } k = i+1 : j-1 \quad \text{(from left to right)} \\
&\quad\quad\quad\quad \text{sum} = \text{sum} + f_{ik}t_{kj} - t_{ik}f_{kj} \\
&\quad\quad\quad \text{end} \\
&\quad\quad\quad f_{ij} = \frac{1}{t_{ii} - t_{jj}}\text{sum} \\
&\quad\quad \text{end} \\
&\quad \text{end} \\
&\text{end}
\end{aligned}$$

上述算法所需的计算量为 $\frac{2}{3}n^3$
因此计算 $f(A)$ 的大多数工作量都花在了 Schur 分解 $A = UTU^{\mathrm{H}}$ 的计算上.

7.3.2 Parlett 分块递推

若矩阵 A 有相近的或相同的特征值，则对 A 直接进行 Parlett 递归的效果极差.
此时我们应当使用 Parlett 递归的分块形式.

设 Schur 分解 $A = UTU^{\mathrm{H}}$ 的划分如下:

$$T = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1d} \\ & T_{22} & \cdots & T_{2d} \\ & & \ddots & \vdots \\ & & & T_{dd} \end{bmatrix}$$

设 $T_{ii} \in \mathbb{C}^{n_i\times n_i}$ $(i = 1, \dots, d)$ 的特征值互不相交，而 T_{ii} 自身的特征值很相近.
我们记:

$$F = f(T) = \begin{bmatrix} F_{11} & F_{12} & \cdots & F_{1d} \\ & F_{22} & \cdots & F_{2d} \\ & & \ddots & \vdots \\ & & & F_{dd} \end{bmatrix}$$

既然 T_{ii} 的特征值很相近，我们可以使用逼近法计算 $F_{ii} = f(T_{ii})$
比较 $TF = FT$ 中 (i, j) 分块 (其中 $i < j$)，可得到如下递推关系:

$$\begin{gathered}
\sum_{k=i}^{j} T_{ik}F_{kj} = \sum_{k=i}^{j} F_{ik}T_{kj} \\
\Leftrightarrow \\
T_{ii}F_{ij} + \sum_{k=i+1}^{j-1} T_{ik}F_{kj} + T_{ij}F_{jj} = F_{ii}T_{ij} + \sum_{k=i+1}^{j-1} F_{ik}T_{kj} + F_{ij}T_{jj} \\
\Leftrightarrow \\
T_{ii}F_{ij} - F_{ij}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1} (F_{ik}T_{kj} - T_{ik}F_{kj})
\end{gathered}$$

我们可以使用 **Bartels-Stewart 算法**来求解上述 Sylvester 方程得到 F_{ij}
(参考 FDU 数值算法 5. 非对称特征值问题的解法)
于是我们得到 **Parlett 分块递推算法:**

function: $F = \text{Parlett\_Block\_Recursion}(T)$
    for $i = 1 : n \quad$ (diagonal)
        use approximation algorithm to calculate $F_{ii} = f(T_{ii})$
    end
    for $p = 1 : d-1 \quad$ (p-th superdiagonal)
        for $i : d-p \quad$ (from bottom to top)
            $j = i + p$
            $\text{sum} = F_{ii}T_{ij} - T_{ij}F_{jj}$
            for $k = i+1 : j-1 \quad$ (from left to right)
                $\text{sum} = \text{sum} + F_{ik}T_{kj} - T_{ik}F_{kj}$
            end
            $F_{ij} = \text{Bartels-Stewart}(T_{ii}, T_{jj}, \text{sum}) \quad (\text{solve } T_{ii}F_{ij} - F_{ij}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1}(F_{ik}T_{kj} - T_{ik}F_{kj}))$
        end
    end
end

上述算法在计算实矩阵的实函数时是很有用的.
在计算了实 Schur 分解 $A = QTQ^{\mathrm{T}}$ 后可用 Parlett 分块递推算法沿着 T 的对角线来处理 2×2 块的计算问题.

7.4 应用

7.4.1 扰动分析

矩阵指数函数 e^{At} 的扰动分析是大多数矩阵函数扰动分析的基础.
考虑初值问题:

$$\begin{aligned}\dot{X}(t) &= AX(t) \\ X(0) &= I\end{aligned}$$

其中 $A, X(t) \in \mathbb{C}^{n\times n}$
它具有唯一解 $X(t) = e^{tA}$
记 $F(t) := e^{t(A+E)} - e^{tA}$, 则我们有:

$$\begin{aligned}\frac{d}{dt}F(t) &= (A+E)e^{t(A+E)} - Ae^{tA} \\ &= A(e^{t(A+E)} - e^{tA}) + Ee^{t(A+E)} \\ &= AF(t) + Ee^{t(A+E)}\end{aligned}$$

于是我们有:

$$\begin{aligned}\frac{d}{dt}\{e^{-tA}F(t)\} &= -Ae^{-tA}F(t) + e^{-tA}\frac{d}{dt}F(t) \quad (\text{note that } Ae^{-tA} = e^{-tA}A) \\ &= e^{-tA}\left\{\frac{d}{dt}F(t) - Ae^{-tA}\right\} \\ &= e^{-tA}Ee^{t(A+E)}\end{aligned}$$

因此我们有:

$$e^{-tA}F(t) = \int_0^{\mathrm{T}} e^{-sA}Ee^{s(A+E)}ds$$
$$\Leftrightarrow$$
$$e^{t(A+E)} - e^{tA} = F(t) = e^{tA}\int_0^{\mathrm{T}} e^{-sA}Ee^{s(A+E)}ds = \int_{0^{\mathrm{T}}} e^{(t-s)A}Ee^{s(A+E)}ds$$

> **错误的做法:**
> (这是因为 $\exp(A+E) = \exp(A)\exp(E)$ 通常并不成立)

> $$\begin{aligned} e^{t(A+E)} - e^{tA} &= e^{tA}(e^{tE} - 1) \\ &= e^{tA} \int_0^t E e^{sE} ds \\ &= \int_0^t e^{(t-s)A} E e^{s(A+E)} ds \end{aligned}$$

根据上式可知:

$$\frac{\|e^{t(A+E)} - e^{tA}\|_2}{\|e^{tA}\|_2} \leq \frac{\|E\|_2}{\|e^{tA}\|_2} \int_0^{\mathrm{T}} \|e^{(t-s)A}\|_2 \|e^{s(A+E)}\|_2 ds$$

因此我们只需估计出被积函数中指数函数的范数 $\|e^{tA}\|_2$ 的上界，就可将上述结果进一步简化.
详细的讨论参见 **(Matrix Computation $11.3.2$ 节)**
当 A, E 为 Hermite 阵且可交换 (即 $AE = EA$) 时，我们有一个很强的界 (参见 Homework 10 Problem 4)
此外，我们还可不断代入得到更高阶的界 (以代入一次为例):

$$\begin{aligned} e^{t(A+E)} &= e^{tA} + \int_0^t e^{(t-s)A} E e^{s(A+E)} ds \\ &= e^{tA} + \int_0^t e^{(t-s)A} E \left\{ e^{sA} + \int_0^s e^{(s-w)A} E e^{w(A+E)} dw \right\} ds \\ &= e^{tA} + \int_0^{\mathrm{T}} e^{(t-s)A} E e^{sA} ds + \int_0^{\mathrm{T}} e^{(t-s)A} E \left\{ \int_0^s e^{(s-w)A} E e^{w(A+E)} dw \right\} ds \end{aligned}$$

与矩阵指数函数有关的复杂函数可以拉大得到矩阵形式.
以三角矩阵函数的扰动分析为例: **(存疑)**
根据 Euler 公式 $e^{iA} = \cos(A) + i \sin(A)$ 我们有:

$$\exp\left(\begin{bmatrix} 0 & A \\ -A & 0 \end{bmatrix} \right) = \begin{bmatrix} \cos(A) & \sin(A) \\ -\sin(A) & \cos(A) \end{bmatrix}$$

因此我们有:

$$\begin{bmatrix} \cos(A+\Delta A) & \sin(A+\Delta A) \\ -\sin(A+\Delta A) & \cos(A+\Delta A) \end{bmatrix} - \begin{bmatrix} \cos(A) & \sin(A) \\ -\sin(A) & \cos(A) \end{bmatrix} = \exp\left(\begin{bmatrix} 0 & A+\Delta A \\ -(A+\Delta A) & 0 \end{bmatrix} \right) - \exp\left(\begin{bmatrix} 0 & A \\ -A & 0 \end{bmatrix} \right)$$

这样三角函数扰动分析就可以归结为矩阵指数函数的扰动分析.

7.4.2 一阶线性自治系统

在控制论中，连续时间的一阶线性自治系统可表示为一阶微分方程:

$$x'(t) = \frac{dx(t)}{dt} = Ax(t)$$

其中 $x(t) \in \mathbb{C}^n$ 是状态向量，$A \in \mathbb{C}^{n \times n}$ 是系统矩阵.
记初始状态为 $x(0)$，可以验证 $x(t) = e^{tA} x(0)$ 是方程的解:

- 注意到 $e^{-tA} x(t) = x(0)$，两边同时求导得到:

  $$\begin{gathered} (e^{-tA} x(t))' = -Ae^{-tA} x(t) + e^{-tA} x'(t) = 0_n \\ \Leftrightarrow \\ x'(t) = e^{tA} A e^{-tA} x(t) = A e^{tA} e^{-tA} x(t) = Ax(t) \end{gathered}$$

  因此 $x(t) = e^{tA} x(0)$ 是一阶微分方程 $x'(t) = Ax(t)$ 的解.

(生成矩阵 & 随机矩阵)

- 若 $Q = [q_{ij}] \in \mathbb{R}^{n \times n}$ 满足 $q_{ij} \geq 0\ (i \neq j)$ (非对角元非负) 和 $Q 1_n = 0_n$ (所有行和均为 0)，
  则我们称 Q 是一个**生成矩阵** (generator matrix)
  (通常用于描述连续时间 Marcov 链的状态转移速率)
- 若 $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ 满足 $a_{ij} \geq 0\ (\forall\, i, j = 1, \ldots, n)$ (即 A 是非负矩阵) 和 $A 1_n = 1_n$ (所有行和均为 1)
  则我们称 A 是一个**随机矩阵** (stochastic matrix)
  (通常用于描述离散时间 Marcov 链的状态转移过程)

设 $Q \in \mathbb{R}^{n\times n}$ 是一个生成矩阵.
我们知道一阶微分方程 $x'(t) = Qx(t)$ 的解是 $x(t) = e^{tQ}x(0)$
可以证明 $\exp(Q)$ 是一个随机矩阵:
(类似地，可以证明 $\exp(tQ)$ 对于任意 $t \geq 0$ 都是随机矩阵)

- ① 首先证明 $\exp(Q)$ 是一个非负矩阵:
  由于 Q 的非对角元都是非负的，故我们可以取一个足够大的 $\alpha \in \mathbb{R}$ 使得 $Q + \alpha I_n$ 是一个非负矩阵.
  于是我们有:

$$\begin{aligned}\exp(Q) &= \exp(Q+\alpha I_n)\exp(-\alpha I_n) \\ &= \left(\sum_{k=0}^{\infty}\frac{(Q+\alpha I_n)^k}{k!}\right)\cdot e^{-\alpha}I_n \\ &= \left(\sum_{k=0}^{\infty}\frac{(Q+\alpha I_n)^k}{k!}\right)\cdot e^{-\alpha}\end{aligned}$$

  注意到 $\exp(Q+\alpha I_n) = \sum_{k=0}^{\infty}\frac{(Q+\alpha I_n)^k}{k!}$ 作为非负矩阵 $Q + \alpha I_n$ 的幂级数，也一定是非负矩阵.
  同时 $e^{-\alpha}$ 是一个正实数，故 $\exp(Q)$ 是一个非负矩阵.
- ② 其次证明 $\exp(Q)$ 的所有行和均为 1:

$$\begin{aligned}\exp(Q)1_n &= \left(\sum_{k=0}^{\infty}\frac{Q^k}{k!}\right)1_n \\ &= \left(I_n + \sum_{k=1}^{\infty}\frac{Q^k}{k!}\right)1_n \\ &= 1_n + \sum_{k=1}^{\infty}\frac{Q^k 1_n}{k!} \\ &= 1_n + \sum_{k=1}^{\infty}\frac{Q^{k-1}\cdot Q1_n}{k!} \quad (\text{note that } Q1_n = 0_n) \\ &= 1_n + \sum_{k=1}^{\infty}\frac{Q^{k-1}\cdot 0_n}{k!} \\ &= 1_n\end{aligned}$$

7.4.3 一阶线性时不变系统

我们可以将连续时间的一阶线性自治系统推广到**一阶线性时不变系统**.
线性时不变 (Linear Time-Invariant, LTI) 系统是指具有线性和时不变特性的动态系统.

- ① 线性性: 系统遵循叠加原理
  任意给定输入信号 $u_1(t)$ 和 $u_2(t)$，设其对应的响应为 $x_1(t)$ 和 $x_2(t)$
  对于任意 $\alpha, \beta \in \mathbb{R}$，输入信号 $u(t) = \alpha u_1(t) + \beta u_2(t)$ 的响应都是 $x(t) = \alpha x_1(t) + \beta x_2(t)$
- ② 时不变性: 系统的动态行为不随时间改变
  若输入信号 $u(t)$ 对应的响应为 $x(t)$，
  则对于任意 $\tau \geq 0$，延迟输入 $u(t-\tau)$ 对应的响应都是 $x(t-\tau)$

考虑连续时间的一阶线性时不变系统:

$$\begin{cases} x'(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

- $A \in \mathbb{C}^{n\times n}$ 代表系统矩阵，决定了状态向量 $x(t) \in \mathbb{C}^n$ 之间的动态关系.
- $B \in \mathbb{C}^{n\times m}$ 代表输入矩阵，描述输入信号 $u(t) \in \mathbb{C}^m$ 如何作用于状态向量 $x(t) \in \mathbb{C}^n$
- $C \in \mathbb{C}^{p\times n}$ 代表输出矩阵，决定状态向量 $x(t) \in \mathbb{C}^n$ 如何影响输出信号 $y(t) \in \mathbb{C}^p$
- $D \in \mathbb{C}^{p\times m}$ 代表直接传递矩阵，表示输入信号 $u(t) \in \mathbb{C}^m$ 对输出信号 $y(t) \in \mathbb{C}^p$ 的直接影响

一阶线性时不变系统可以通过复杂系统的一阶近似得到.
当输入信号 $u(t) \in \mathbb{C}^m$ 恒为零时，上述系统退化为一阶线性自治系统.

考虑求解上述系统:
对 $x'(t) = Ax(t) + Bu(t)$ 同时左乘 e^{-tA} 可得:

$$e^{-tA}x'(t) - e^{-tA}Ax(t) = e^{-tA}Bu(t)$$

于是我们有:

$$\begin{aligned}(e^{-tA}x(t))' &= -Ae^{-tA}x(t) + e^{-tA}x'(t) \\ &= -e^{-tA}Ax(t) + e^{-tA}x'(t) \\ &= e^{-tA}Bu(t)\end{aligned}$$

因此我们有:

$$\begin{aligned}e^{-tA}x(t) - x(0) &= \int_0^{\mathrm{T}} (e^{-sA}x(s))'ds \\ &= \int_0^{\mathrm{T}} e^{-sA}Bu(s)ds\end{aligned}$$

最终得到 $x(t) = e^{tA}x(0) + \int_0^t e^{(t-s)A}Bu(s)ds$
于是连续时间的一阶线性时不变系统的解为:

$$\begin{cases} x(t) = e^{tA}x(0) + \int_0^t e^{(t-s)A}Bu(s)ds \\ y(t) = Cx(t) + Du(t) \end{cases}$$

7.4.4 Lyapunov 方程

Lyapunov 方程是控制论中的重要工具，通常用于判断**线性时不变 (LTI) 系统**的稳定性.
考虑连续时间的一阶线性时不变系统:

$$\begin{cases} x'(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

- $A \in \mathbb{C}^{n\times n}$ 代表系统矩阵，决定了状态向量 $x(t) \in \mathbb{C}^n$ 之间的动态关系.
- $B \in \mathbb{C}^{n\times m}$ 代表输入矩阵，描述输入信号 $u(t) \in \mathbb{C}^m$ 如何作用于状态向量 $x(t) \in \mathbb{C}^n$
- $C \in \mathbb{C}^{p\times n}$ 代表输出矩阵，决定状态向量 $x(t) \in \mathbb{C}^n$ 如何影响输出信号 $y(t) \in \mathbb{C}^p$
- $D \in \mathbb{C}^{p\times m}$ 代表直接传递矩阵，表示输入信号 $u(t) \in \mathbb{C}^m$ 对输出信号 $y(t) \in \mathbb{C}^p$ 的直接影响

当 $u(t) \equiv 0$ (即没有外部输入) 时，我们称对应的响应 $x(t) = e^{tA}x(0)$ 为**自由响应**.
给定任意矩阵 $P \in \mathbb{C}^{n\times n}$ (应用中一种通常的构造是 Hermite 半正定阵 $P = vv^{\mathrm{H}}$，其中 $v \in \mathbb{C}^n$ 是给定向量)
若存在 $X_\star$ 是 Lyapunov 方程 $AX + XA^{\mathrm{H}} = P$ 的解，
则上述系统的自由响应 $x(t) = e^{tA}x(0)$ 渐近稳定 (即 $\lim_{t\to\infty} e^{tA} = 0_{n\times n}$)

可以证明当 A 是 **Hurwitz 矩阵** (即所有特征值的实部都为负数) 时，
Lyapunov 方程 $AX + XA^{\mathrm{H}} = P$ 具有唯一解 $X_\star = -\int_0^\infty e^{tA}Pe^{tA^{\mathrm{H}}}dt$

- > **(Sylvester 定理, Matrix Analysis 定理 $2.4.4.1$)**
  > 设 $A \in \mathbb{C}^{m\times m}, B \in \mathbb{C}^{n\times n}, C \in \mathbb{C}^{m\times n}$
  > Sylvester 方程 $AX - XB = C$ 有唯一解 $X \in \mathbb{C}^{m\times n}$ 当且仅当 A, B 没有公共特征值 (即 $\mathrm{eig}(A) \cap \mathrm{eig}(B) = \emptyset$)

  当 A 所有特征值的实部都为负数时，$-A^{\mathrm{H}}$ 所有特征值的实部都为正数.
  因此 $A, -A^{\mathrm{H}}$ 没有公共特征值，根据 Sylvester 定理可知 Lyapunov 方程 $AX + XA^{\mathrm{H}} = P$ 具有唯一解.
- 下面我们验证 $X_\star = -\int_0^\infty e^{tA}Pe^{tA^{\mathrm{H}}}dt$ 是 Lyapunov 方程的解.
  注意到:

  $$\begin{aligned}(e^{tA}Pe^{tA^{\mathrm{H}}})' &= (Ae^{tA})Pe^{tA^{\mathrm{H}}} + e^{tA}P(A^{\mathrm{H}}e^{tA^{\mathrm{H}}}) \\ &= Ae^{tA}Pe^{tA^{\mathrm{H}}} + e^{tA}Pe^{tA^{\mathrm{H}}}A^{\mathrm{H}}\end{aligned}$$

  于是我们有:

$$
\begin{aligned}
AX_\star + X_\star A^{\mathrm{H}} &= -A\int_0^\infty e^{tA}Pe^{tA^{\mathrm{H}}}dt - (\int_0^\infty e^{tA}Pe^{tA^{\mathrm{H}}}dt)A^{\mathrm{H}} \\
&= -\int_0^\infty (Ae^{tA}Pe^{tA^{\mathrm{H}}} + e^{tA}Pe^{tA^{\mathrm{H}}}A^{\mathrm{H}})dt \\
&= -\int_0^\infty (e^{tA}Pe^{tA^{\mathrm{H}}})'dt \\
&= -\{e^{tA}Pe^{tA^{\mathrm{H}}}\}\big|_0^\infty \\
&= -(0_{n\times n} - P) \\
&= P
\end{aligned}
$$

其中 "A 的特征值实部小于 0" 保证了 e^{tA} 在 $t \to \infty$ 时趋近于 $0_{n\times n}$
因此 $e^{tA}Pe^{tA^{\mathrm{H}}}$ 在 $t \to \infty$ 时趋近于 $0_{n\times n}$

The End