■ 技 術 動 向 ■

제어 및 시스템이론에서의 컴퓨터 계산방법

李 夫 熙*

■ 차 례 ■

1. 서 론

지금까지 제어나 시스템에 관한 문제들이 이론적으로는 대부분 훌륭히 다루어졌다. 그러나 이러한 문제들은 이론적으로 완벽하지만 실제적으로 컴퓨터를 이용해서 문제를 풀 경우에 나타나는 뉴메리컬(numerical)특성에 대해서는 거의 이해되지 않았고 따라서 이러한 문제들을 푸는 수학적인 소프트웨어 (software)는 아직도 초기단계에 있다고 볼 수 있다. 여기서 우리는 제어나 시스템 문제를 컴퓨터로 푸는방법과 이때 나타나는 여러가지 제반문제에 관해 언급하고자 한다.

여기서는 주로 전형적인 선형모델을 다루기로 한다.

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (1)$$

$$Y(t) = Cx(t) + Du(t) \quad (2)$$

여기서 $x(t) \in \mathbf{R}^n$ 은 상태벡터이고 $u(t) \in \mathbf{R}^m$ 은 입력벡터이고 $Y(t) \in \mathbf{R}^r$ 은 출력벡터이다.

여기에 대응되는 discrete − time 모델은 다음과 같다.

$$x_{k+1} = A x_k + B u_k \quad (3)$$

$$y_k = C x_k + D u_k \quad (4)$$

시스템이나 제어에 관한 문제들은 문제에 따라 적합한 앨고리즘(Algorithm)을 가지고 있다. 대부분 어떤 경우 (예를 들면 " small order ")에는 훌륭한 결과를 얻게되지만 경우에 따라서는 (예를 들면" large order ") 심각한 문제를 야기하게 된다. 이와같은 결과가 나타내는 근본적인 이유는 바로 컴퓨터가 가지고 있는 본질적 특성인 " finite word length" 때문이다. 간단한 예를 들어보자. 우리가 e^A 를 IBM 370 컴퓨터를 이용해서 single precision으로 구하고자 할때 대략 6자리 까지의 유효숫자를 갖게된다.

$A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix}$ 로 주어졌을때 우리는 다음 공식을 이용해서 e^A 를 구해보자.

$$e^A = \sum_{k=0}^{\infty} \frac{1}{ki} A^k \quad (5)$$

컴퓨터로 이것을 계산할때 처음의 60항만 합해보면 충분하게 되는데 이유는 나머지 항들이 10^{-7} 오더 (order)이므로 무시되기 때문이다.

그 결과는 $\begin{pmatrix} -22.2588 & -1.43277 \\ -61.4993 & -3.47428 \end{pmatrix}$ 이 되는데

불행히도 실제 정확한 답은

$\begin{pmatrix} -0.785789 & 0.551819 \\ 1.47152 & 1.10364 \end{pmatrix}$ 가 된다. 문제는 중간

* 正会員 : 西江大 理工大 電子工學科 助教授 · 工博

항들의 값이 매우 크기때문인데 예를 들면 17번과 18번 항의 값들이 대략 10^7 오더를 갖고 있으며 부호는 반대가 되기때문에 합산과정에서 큰 오차를 유발하게 된다.

다행히 이 문제는 다른 방법으로 정확한 값을 구할 수 있었지만 대부분의 문제들이 손쉽게 결과가 틀렸다는 것을 측정하기가 어렵게 된다. 따라서 신뢰성있고 효율적인 알고리즘을 개발하는 문제는 앞으로 중요 연구과제가 될 것이다.

여기서 앞으로 전개해 나갈 내용을 알아 보자. 2장에서는 뉴메리칼 "안정도" (Numerical stability)와 주어진 문제의 "조건" (conditioning)에 대해 언급하고 3장에서는 선형대수에서의 제반문제들을 알아보고 4장에서 기타 제어 및 시스템에서의 응용문제를 다루고 마지막으로 5장에서 결론을 내려보았다.

여기서 다음 장으로 넘어가기 전에 주로 사용될 전형적인 표기에 대해 언급하겠다.

$\mathbf{F}^{n\times m}$: Field $\mathbf{F}$의 계수를 가진 $n\times m$ 행열의 집합 (여기서 $\mathbf{F}$는 주로 실수 $\mathbf{R}$ 이거나 복소수 $\mathbf{C}$)

$\mathbf{F}_r^{n\times m}$: rank 가 r 인 $n\times m$ 행열의 집합

A^T : 행열 A의 전치행열

A^+ : 행열 A의 Moore-Penrose Pseudoinverse

$\|A\|$: 행열 A의 Spectal norm

(즉 $\|A\| = \max \|Ax\|_2) \|x\|_2 = 1$

$\lambda_i(A)$: 행열 A의 i번째 고유치

끝으로, 앞으로 가끔 인용되는 중요용어 하나를 정의 하겠다. machince precision 은 사용하는 컴퓨터에 1을 더했을 경우 1보다 큰 값을 표시하는 가장 작은 양수 ε으로 정의된다. 다시 말하면 ε보다 작은 어떤 숫자 δ라도 1에 더할 경우 라운드어프에 의해 무시되고 결과는 1이 된다.

2. 안정도와 조건

이 장에서는 수치해석에서 중요한 역할을 하는 뉴메리칼 안정도와 문제의 조건에 대해 알아본다.[1-3] 지금 수학적으로 정의된 문제 f가 주어졌다고 가정하자. 이때 들어온 입력값이 d일때 문제의 해는 $f(d)$로 주어진다. d^*가 d의 근사치라고 했을때만약 $f(d^*)$값이 $f(d)$ 값과 매우 가까울때 우리는 이 문제 f가 "좋은 조건"으로 주어졌다고 한다. 만약 d^*값이 d값과 매우 가까운 데도 $f(d^*)$와 $f(d)$가 오차가 클 경우는 f가 "나쁜 조건"으로 주어졌다고 할다. 물론 여기서 "가깝다" 고하는 개념은 문제에 따라 달라질 수 있고 정확한 정의를 하기는 어렵게 된다.

나쁜 조건으로 주어진 문제를 예를 들어보자. 행열 $A\varepsilon R^{n\times n}$ 가 다음과 같이 주어졌을때 고유치(eigenvalue)를 구하려고 한다.

$$A=\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0\\ 0 & 0 & 1 & & \ddots & \vdots\\ \vdots & \vdots & & \ddots & & 0\\ \vdots & 0 & & & & 1\\ 0 & 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

이 경우 분명히 n개의 고유치값은 모두 0이 되는데 지금 주어진 입력에 작은 변화(Perturbation)가 일어난 경우를 고려한다. 행열 A의 마지막 행 첫번째 요소에 2^{-n} (Small Perturbation)을 더한 행열을 생각하면 우리는 이행열이 n개의 각각 다른 고유치를 갖는다는 것을 알 수 있으며

$\lambda_k=\frac{1}{2}\ exp\ (\frac{2k\pi}{n}j)$로 표시된다. 즉 이와같이 작은 변화가 입력에서 나타나도 결과에서는 매우 큰 오차를 발생하는 문제를 우리는 나쁜 조건으로 주어졌다고 한다.

다음에는 f^*가 f를 계산하기 위해 임플리멘트 되어진 알고리즘이라고 하자. d가 주어졌을때 $f^*(d)$는 알고리즘을 적용해서 입력치가 d일때의 값이된다. 만약 모든 $d\varepsilon D$에 대해 그 값과 가까운 $d^*\varepsilon D$가 존재해서 $f^*(d)$와 $f(d^*)$ 값이 가까울때 우리는 알고리즘 f^*가 "안정하다" 고한다. 여기서 D는 입력값의 집합이다. 만약 f가 좋은조건일 경우 $f(d^*)$와 $f(d)$는 매우 가깝게 되고 따라서 $f^*(d)$는 $f(d)$와 가깝게 되는데 다시 말하면 f^*는 그자체로서 더 이상의 큰 오차를 유발하지 않게된다. 물론 안정한 알고리즘을 사용한다고 해서 나쁜 조건으로 주어진 문제를 오차없이 풀 수 있는것은 아니지만 불안정한 알고리즘은 좋은 조건의 문제도 오차를 많이 내게된다. 그러므로 정확한 결과를 얻기위해서는 문제의 조건과 알고리즘의 안정도가 중요한 의미를 갖는다.

라운드 어프(round off) 오차가 알고리즘의 안정도를 결정하는데 주요원인이 된다. 그러나 실제 모든 오퍼레이션(operation) 마다 (예를 들면 곱하기, 더하기 등) 야기되는 오차를 일일히 계산하는 것 (forward error analysis)은 실제 불가능하다. 대신 J•H• Wilkinson 등 여러사람들은 라운드 어프 오차를 해석하기위해 backward error analysis를 이

용하고 있다.[1)]

3. 선형대수 방정식과 최소자승법

3.1 선형대수 방정식과 최소자승법

수치해석에서 가장 기본적인 문제는 행열 $A \varepsilon \boldsymbol{R}^{n\times n}$ 가 주어졌을때 다음 수식을 만족시키는 벡터 x 를 구하는 것일 것이다.

$$Ax = b \tag{6}$$

n 이 200 이하일 경우 가장 흔히 사용되는 앨고리즘은 Gauss 소거법이다. 이것은 행열 A를 하삼각행열 (lower triangular matrix)인 L과 상삼각행열 (upper triangular matrix)인 U의 곱으로 분해하는 것에 기초를 두고있다. 이 앨고리즘의 안정도에 대해 알아보자. 우리는 계산된 해가 다음방정식의 정확한 해 (true solution)의 근사치가 된다는 것을 쉽게 증명할 수 있다[1)].

$$(A+E)\ x = b \tag{7}$$

여기서 행열 E의 요소들을 e_{ij} 라 하면 $e_{ij} \leq \phi(n)\cdot\gamma\cdot\beta\cdot\varepsilon$ 으로 주어진다. $\phi(n)$은 n 값의 적당한 함수로서 작은 값이며 γ는 "growth factor"로서 일반적으로 작은 값으로 주어진다. 또 β는 행열 A의 "norm"과 같이 변화하며 ε은 컴퓨터의 machine precision이다[1)]. 따라서 특별한 경우가 아니면 E는 ε오더의 작은 값으로 이루어 지는 행열이 된다. 이때 $K(A) = \|A\|\ \|A^{-1}\|$라고 정의하고 일반적인 오차이론을 적용하면 A혹은 b의 오차때문에 나타나는 계산결과는 대략 정확한 해의 $K(A)$의 곱이된다. 여기서 $K(A)$는 문제의 조건을 나타내는 값이 되며 우리는 이것을 "조건치"라고 한다. 따라서 조건치가 작을 경우 앨고리즘은 안정하게 된다. 여기서 $K(A)$값 (또는 A^{-1})을 구하는 것이 문제를 해석하는데 중요한 역활을 하게된다. 이런 문제들을 고려해서 신뢰성있고 안정한 앨고리즘을 개발한 것이 LINPACK[4)]이다.

선형대수에서 또다른 중요한 문제는 최소자승 문제이다. 즉 주어진 행열 $A \varepsilon \boldsymbol{R}_k^{m\times n}$ 와 $b \varepsilon \boldsymbol{R}^m$ 에서 $\|Ax - b\|_2$ 를 최소화시키는 벡타 $x \varepsilon \boldsymbol{R}^n$를 구하는 문제인데 일반적으로 $k = n \leq m$으로 주어진다[1),5)] 이 문제를 푸는방법은 여러가지가 있겠지만 보통 주어진 행열 A를 Orthogonal 행열 Q와 상삼각행열인 R로 분해한다. (QR factorization). 이와같이 분해하기 위해서는 n개의 Householder 행열 H_i를 연속적으로 사용해서 행열 A를 다음과 같이 변환한다.

$$H_n \cdot H_{n-1} \cdot \cdots H_1\ A = \begin{pmatrix} R \\ O \end{pmatrix} \tag{8}$$

각각의 Householder 변환 H_i는 $I - \dfrac{2uu^T}{u^T u}$ 와 같은 형으로 이루어져 있고 각각이 모두 orthogonal 행열이 된다.

$$H_n \cdot H_{n1} \cdot H_1 b = \begin{pmatrix} c \\ d \end{pmatrix} \tag{9}$$

$Rx = C$를 만족시키는 벡타 x가 원하는 해가 되며 이 앨고리즘은 안정하다는 것을 쉽게 증명할 수 있고 또 오차이론으로 부터 조건치는 $\|A\|\|A^+\|$ 가 된다. 여기서 A^+는 행열 A의 Moore - Penrose Pseudoinverse 를 나타낸다.

3.2 고유치(eigenvalue)와 일반화된 고유치 (generalized eigenvalue)문제

행열 $A \varepsilon \boldsymbol{R}^{n\times n}$ 이 주어졌을때

$$Ax = \lambda x \tag{10}$$

를 풀고자한다. 수식 (10)을 만족시키는 벡터 $x \varepsilon \boldsymbol{R}^n$을 고유벡터, 상수 $\lambda \varepsilon \boldsymbol{C}^1$을 고유치라 한다.

이러한 고유치 문제를 수치해석으로 푸는데는 라운드어프 오차가 존재하게 되며 이 오차를 이해하고 해석한 것은 아주 최근의 일이다. 아직도 이러한 고유치 문제의 한 분야가 되는 invariant subspace 문제들은 계속 연구중에 있다. 고유치문제를 푸는데 가장 많이 사용하고 있는 방법은 Francis의 QR 앨고리즘[6)]인데 이 앨고리즘을 적용하기 전에 일반적으로 앞에서 언급한 Householder 변환을 이용하여 행열 A를 상헤센버그 (upper Hessenburg) 형태로 우선 변환시키게 된다[7)]. 다음에 QR 앨고리즘을 적용시켜 불럭 (block) 상삼각행열인 S를 얻게되는데 행열 S는 실수 고유치를 갖는 1×1불럭과 복소수 고유치를 갖는 2×2 블럭이 된다. 결국 우리는 모든 변환을 함께 고려해서 다음과 같은 식을 얻게 된다.

$$U^T A U = S \tag{11}$$

QR앨고리즘, 그리고 이것과 연관된 앨고리즘에 관한 안정도, 고유치 및 고유벡터의 조건 등에 관한 해석은 참고문헌[8),9)] 등에서 찾을 수 있다. 고유치에 관한 수학적인 소프트웨어가 본격적으로 개발된 것은 1970년대 이후 부터이다. EISPACK Subroutine[10),11)]은 이 문제에 대한 소프트웨어의 집대성이라 볼 수 있다.

QR앨고리즘과 아주 비슷한 것은 다음과 같은 일반화된 고유치문제를 푸는데 사용하는 QZ 앨고리즘

이다[12].

$$Ax = \lambda Mx \qquad (12)$$

여기서 A, $M \in R^{n\times m}$ 이다.

이 앨고리즘은 orthogonal 행열 Q와 Z를 사용하여 QAZ를 블럭 상삼각행열, QMZ를 상삼각행열로 만들게 된다[13]. 일반화된 고유치문제 해석은 고유치문제보다 다루기가 힘든 대신에 일반적으로 응용분야가 넓게 된다.

3.3 특성치 분해(Singular Value Decomposition)

특정치 분해 문제는 통계학에서 오랜 역사를 가지고 있으며 최근에는 제어나 시스템 분야에서 각광을 받고 있는데 다음과 같이 나타낼 수 있다.

행열 $A \in \mathbf{R}^{n\times m}$이 주어지면 Orthogonal행열 $U \in \mathbf{R}^{n\times m}$와 $V \in \mathbf{R}^{n/m}$을 구해서 아래와 같은 결과를 얻는다.

$$U^{T} AV = \Sigma \qquad (13)$$

여기서 $\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}$ 이고 $S = \begin{pmatrix} \sigma_1 & & 0 \\ & & \\ 0 & & \sigma_r \end{pmatrix}$,

$\sigma_1 \geq \sigma_2 \geq \cdots\cdots \geq \sigma_r > 0$ 이다.

이 문제를 계산하기 위한 효율적이고 안정한 앨고리즘은 Golub과 Reinsch[14]에 의해 개발되었고 앨고리즘에 대한 임플리멘트는 참고문헌 4), 11)에서 찾을 수 있다. 또한 좀더 상세한 내용은 15)에서 알아볼 수 있을 것이다.

4. 기타 제어 및 시스템에서의 응용문제

4.1 Controllability와 Observability

선형제어문제나 시스템 이론을 다루는데 있어서 Controllability와 Observability는 매우 중요하게 된다. 이 문제를 다루는데 있어 앞에서 언급한 finite arithmatic으로 임플리멘트 하는데 여러가지 문제가 생긴다.

또한 Controllability, Observability와 관련된 새로운 시스템 이론 중에 "balancing"[16]이라는 것이 있는데 이것은 어떤 주어진 시스템을 balancing transformation 행열을 이용하여 Controllability와 Observability가 같은 값을 갖는 새로운 시스템으로 변환시킨 후 해석을 함으로써 특히 시스템의 오더가 매우 큰 경우 시스템의 오더를 거의최적 상태로 줄이는데 사용된다.

이러한 balancing transformation 행열을 구하는 앨고리즘은 참고문헌(17)에서 찾을 수 있다.

4.2 주파수 응답

선형 시스템이론에서 주파수 응답 행열 $G(jw)$는 자주 사용된다.

$$G(jw) = C(jw - A)^{-1} B \qquad (13)$$

실제 return difference 행열 $I + G(jw)$의 여러가지 norm은 선형모델의 안정도, noise 응답, 감도(Sensitivity) 등을 측정하는데 결정적 역할을 하게 되며[18] 따라서 여러가지 w에 대한 $G(jw)$의 값을 효율적으로 구할 수 있는 앨고리즘이 요구된다. 참고문헌 19)에서 새로운 앨고리즘을 발표했는데 이것은 직접 대수방정식 $(jwI - A)X = B$를 푸는 대신에 행열 A를 먼저 상헤센버그 형태로 변환시켜 푸는 방법이다.

4.3 Linear multivariable System의 Geometric theory에서의 문제

선형모델인 식 (1), (2)로 주어지는 시스템을 geometric 방법[20]으로 해석할 경우 여러가지 뉴메리칼 문제가 발생한다. 이중에서도 Supremal (A,B)-invariant subspace와 Controllability subspace를 구하는 문제에 가장 관심이 집중되고 있었는데 Van Dooren[21),22)]에 의해 거의 완벽하게 발전되었다. 그는 이 문제를 matrix pencil $(N - \lambda L)$로 해석을 했는데 이것을 이용하면 상당히 신뢰성있는 앨고리즘을 얻게 된다.

4.4 Lyapunov, Sylvester, Riccati방정식

제어나 시스템 이론에서 자주 나타나는 행열 방정식에는 Lyapunov 방정식과 Sylvester 방정식이 있다.

$$FX + XF^{T} + H = 0 \qquad (14)$$

$$FX + XG + H = 0 \qquad (15)$$

그리고 discrete-time에서는 다음과 같이 나타난다.

$$FXF^{T} - X + H = 0 \qquad (16)$$

$$FXG - X + H = 0 \qquad (17)$$

그러나 놀랍게도 지금까지 이러한 방정식을 수치

해석으로 푸는 것에는 별로 관심이 집중되지 않았다.

대부분 특별한 경우에만 적용할 수 있는 것이 고작이었고 특히 안정도, 조건, 컴퓨터 임플리멘테이션 등에는 거의 연구가 되지 않았다. 지금까지 나와있는 앨고리즘 중에서 가장 효율적이고 신뢰성이 있는 것은 Bartels와 Stewart[23]에 의해 개발된 앨고리즘이다. 기본적인 아이디어는 행영 F를 블럭 상삼각행렬로 변환시켜 back substitution을 하는 것이다.

Sylvester 방정식의 경우에는 행렬 F와 G를 동시에 블럭 상삼각행렬로 변환시켜 back substitition을 하게된다. 그러나 아직도 이러한 방정식의 조건을 행렬의 계수로써 나타내는 문제는 계속 연구과제로 남아있다.

앞에서 언급한 Lyapunov와 Sylvester 방정식보다 더욱 자주 쓰이고 또 좀더 복잡한 것이 바로 Riccati 방정식이다. 이 방정식에 관한 문헌은 지금까지 수백편이상 나왔지만 수치해석의 견지에서 볼때는 아직도 불모지라 볼 수 있다. 대칭인 Riccati 방정식은 다음과 같이 주어진다.

$$XGX+FX+XF^T+H=0 \qquad (18)$$

또 discrete - time경우에는 다음과 같다.

$$FXF^T - X - FXG_1(G_2+G_1^TXG_1)^{-1} G_1^TXF^T = 0 \qquad (19)$$

대부분의 경우 유일한 해가 존재하기 위한 여러가지 가정이 주어지게 된다. Riccati 방정식을 푸는 방법중에 가장 신뢰성이 있는 것이 Schur 방법을 이용한 것인데[24] 이것도 역시 행렬을 블럭 상삼각행열로 변환시키는 것에 기초를 두고 있다. 그러나 Schur 방법을 사용하던지 또는 다른 방법을 사용하던지 Riccati 방정식의 조건을 구하는 것은 계속 좋은 연구분야로 남아있다.

4.5 선형 미분방정식

다음과 같은 방정식은 우리가 흔히 접하게 되는 선형 미분방정식이다.

$$\dot{X}(t)=AX(t)+f(t)\ ;\ X(o)=X_0 \qquad (20)$$

이 방정식을 푸는데 가장 좋은 앨고리즘이 어떤 것인가에 대해서는 아직도 여러가지 토론이 진행되고 있지만 보통 많이 사용하는 방법은 e^{At}를 사용하는 것이다. 이 문제에 관한 것은 참고문헌 25)에서 찾아볼 수 있다. 또한 시스템 이론에서 나타나는 미분방정식을 해석한 것으로는 26)이 있다.

5. 결 론

지금까지 선형대수에서의 수치해석과 이 아이디어의 응용문제에 관해 언급하였다. 중요한 문제는 원형 모델에서 계수가 어느정도 변화될 경우 어떻게 신뢰성있게 문제를 풀 수 있느냐 하는 것이다. 또 실제 문제를 푸는 앨고리즘과 신뢰성있는 수단(소프트웨어)을 개발하는 것이 결국은 마지막의 목적이 될것이다. 특히 주어진 문제의 조건을 결정하는 것은 새로운 앨고리즘을 개발하는데 큰 도움을 주게되고 나아가서 그 앨고리즘의 안정도를 측정하는데 중요한 역할을 하게 된다.

참 고 문 헌

1) Stewart, G. W. ; "Introduction to matrix computation", Academic Press, New York, 1973.

2) Forsythe, G.E. ; M.A. Malcolm, and C.B. Moler ; "Computer methods for mathematical computations", Prentice-Hall, Englewood Cliffs, New Jersey, 1977.

3) Stoer, J.; and R. Bulirsch ; "Introduction to numerical analysis", Springer-Verlag, New York, 1980.

4) Dongarra, J., et al., ; "LINPACK User's guide," SIAM, Philadelphia, 1979.

5) Lawson, C.L., R.J. Manson ; "Solving least squares problems", Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

6) Francis, J.G.F. ; "The QR transformation I, II," Computer J., vol 4, 1961, 1962, pp. 265-271, 332-335.

7) Martin, R.S., and J.H. Wilkinson ; "Similarity reduction of a general matrix to hessenburg form," Numer. Math., vol 12, 1968, pp. 349-368.

8) Wilkinson, J.H. ; "The algebraic eigenvalue problem," Oxford University Press, London, 1965.

9) Parlett, B.N. ; "The symmetric Eigenvalue problem", Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

10) Smith, B.T., et al. ; "Matrix eigensystem routines EISPACK guide," 2nd ed., Lect. Notes in Comp. Sci., vol 6, Springer-Verlag, New York, 1976.
11) Garbow, B.S., et al., ; "Matrix eigen system routines-EISPACK guide extension," Lect. Notes in Comp Sci., vol 51, Springer-Verlag, New York, 1977.
12) Moler, C.B., and G.W. Stewart ; "An Algorithm for generalized matrix eigenvalue problem," SIAM JNA, vol. 10, 1973, pp. 241-256.
13) Laub, A.J., and B.C. Moore ; "Calculation of trans-mission zeros using QZ techniques," Automatica, vol 14, 1978, pp. 557-556.
14) Golub, G.H., and C. Reinsch ; "Singular value decomposition and least squares solutions," Numer. Math., vol 14, 1970, pp. 403-420.
15) Paige, C.C. ; "Properties of numerical algorithms related to computing controlabilitiy," IEEE TAC ; AC-26, 1981, pp. 130-138.
16) Moor, B.C. ; "Principal component analysis in linear systems in linear systems," IEEE TAC, AC-26, 1981, pp. 17-32.
17) Laub, A.J. ; "On computing balancing transformations," Proc. 1980. JACC, San Fancisco, CA, 1980. pp FA 8-E.
18) Safanov, M.G., et al. ; "Feedback properties of multivariable systems: the role and use of the return difference matrix," IEEE TAC, AC-26, 1981, pp. 47-65
19) Laub, A.J. ; "Efficient multivariable frequency response calculations," IEEE TAC, AC-26, 1981.
20) Wonham, W·M· ; "Linear multivariable control : A geometric approach," 2nd ed·, Springer-Verlag, New York, 1979.
21) Van Dooren, P· ; "The generalized eigenstructure problem in linear system theory," IEEE TAC, AC-26, 1981, pp. 111～129.
22) Van Dooren, P· ; "The generalized eigenstructure problem. applications in linear system theory," ph·D·Thesis, Katholieke Universiteit Leuven, Belgium, 1979.
23) Bartels, R·H·, and G·W· stewart ; "Solution of the matrix equation $AX+XB=C$," Comm. AcM, vol 15, 1972, pp. 820～826.
24) Laub, A·J· ; "A schur method for solving algebraic riccati equations," IEEE TAC, AC-24, 1979, pp. 913～921.
25) Moler, C·B·, and C·F·Van Loan ; "Nineteen dubious ways to compute the exponential of a matrix," SIAM review, vol 20, 1978, pp. 801～836.
26) Enright, W· ; "On the efficient and reliable numerical solution of large linear systems of l large linear systems of ODE's," IEEE TAC, AC-24, 1979, pp. 905～908.