EE16B Course Notes

Anmol Parande

Spring 2019 - Professors Anant Sahai, Kris Pister, and Jaijeet Roychowdhury

1 Discrete Time Models

Although circuits and other systems function in continuous time, computers can only act in discrete time. This means the signals they send out to try and control the system are piecewise-continuous, and they can only sense the system at specified intervals δ At a discrete time-step i, the computer will measure $\vec{x}_d[i]$ and applies the control $\vec{u}[i]$ over the time-interval $[i, i+1]$. Keep in mind that $i = \delta t$ where t is the continuous time.

Definition 1 *The discrete time model is* $\vec{x}_d[i+1] = A_d\vec{x}_d[i] + B_d\vec{u}[i] + \vec{w}[i]$ *where* $\vec{w}[i]$ *is some disturbance.*

1.1 Converting between continuous time and discrete time

To see how to convert between a continuous time model to a discrete time model, consider the following system.

$$\frac{dx}{dt} = \lambda x + u, u = e^{st}$$
$$\left(\frac{d}{dt} - \lambda\right) x = u$$

Solving will give $x = \alpha e^{\lambda t} + \beta e^{st}$ Plugging this into the original equation

$$\alpha\lambda e^{st} + \beta s e^{st} = \lambda(\alpha e^{\lambda t} + \beta e^{st}) + e^{st}$$
$$\text{if } \lambda \neq s$$
$$\beta s = \beta\lambda + 1 \implies \beta = \frac{1}{s-\lambda}$$

Since the computer only outputs constant signals, $s = 0$, so $x = \alpha e^{\lambda t} - \frac{u}{\lambda}$ If $x(0) = x_0$

$$x(0) = \alpha - \frac{u}{\lambda} = x_0$$
$$x = (x_0 + \frac{u}{\lambda}e^{\lambda t}) - \frac{u}{\lambda}$$
$$x = x_0 e^{\lambda t} + u\left(\frac{e^{\lambda t}-1}{\lambda}\right)$$

Thus the discrete time model is

$$x[i+1] = e^{\lambda\delta}x[i] + \left(\frac{e^{\lambda t}-1}{\lambda}\right)u[i]$$

since the initial condition for the next time-step is simple the value at the previous time-step

1.2 Finding A and B

What would happen if we didn't know what A and B were though? It turns out that we can find it by gathering data about the system and then using least squares. In the scalar case:

$$\begin{gathered}
x[i+1] = ax[i] + bu[t] + w[i] \\
x[1] = ax[0] + bu[0] \\
x[2] = ax[1] + bu[1] \\
\vdots \\
x[m] = ax[m-1] + bu[m-1]
\end{gathered}$$

This means with m observations, we can construct a matrix D and a vector $\vec{s}$ such that we have the least squares problem

$$D\vec{p} = \vec{s}, \vec{p} = \begin{bmatrix} a \\ b \end{bmatrix}$$

The same thing can be done in the vector case. We just create multiple parallel systems by multiplying out the rows.

2 Control Theory

2.1 Controllability

Definition 2 *A system is controllable if* $\forall x^*, \forall x[0] \exists u[0], u[1], ..., u[k-1]$ *such that* $x[k] = x^*$ *given that the system starts at* $x[0]$

In words, this means that there exists a sequence of control actions that can eventually get the system anywhere we want it to be from any given initial state.

Theorem 1 *If a state is n-dimensional, then a system* $\vec{x}_d[i+1] = A\vec{x}[i] + B\vec{u}[i] + \vec{w}[i]$ *is controllable in k time-steps if* $span(B, AB, ..., A^{k-1}B) = \mathbb{R}^n$

Proof 1 *Without any noise,* $\vec{x}_d[i+1] = A\vec{x}[i] + B\vec{u}[i]$

$$\begin{gathered}
\vec{x}[1] = A\vec{x}[0] + B\vec{u}[0] \\
\vec{x}[2] = A\vec{x}[1] + B\vec{u}[1] = A^2\vec{x}[0] + AB\vec{u}[0] + B\vec{u}[i] \\
\vec{x}[3] = A\vec{x}[2] + B\vec{u}[2] = A^3\vec{x}[0] + A^2B\vec{u}[0] + AB\vec{u}[i] + B\vec{u}[2] \\
\vdots \\
\vec{x}[k] = A^k\vec{x}[0] + \textstyle\sum_{i=0}^{k-1} A^{k-i-1}B\vec{u}[i]
\end{gathered}$$

Thus the current state is merely a linear combination of vectors in the column spaces of A^k *and* $B...A^{k-1}$*, so if this span is equal to* $\mathbb{R}^n$*, we can reach every state.*

Definition 3 *The controllability matrix is a the matrix whose column space is all the reachable states.*

$$\left[\ B \mid AB \mid A^2B \mid ... \mid A^{k-1}B\ \right]$$

However, if we want to check if a system is controllable in 100 time-steps for a 900 state system, it will be painful to calculate the column space of the controllability matrix. It turns out that we can stop checking when $dim(col(B, AB, ..., A^{p-1})) = dim(col(B, AB, ..., A^p)), (p \leq k)$ (i.e the dimension of the column space stops growing) This means we can build the controllability matrix up piece-by-piece and stop early rather than computing the full column space only to fail. Consider the following proof for why this works.

Proof 2 *If $\vec{b}$ is column vector of the matrix B, then $A^p\vec{b} = \sum_{i=0}^{p-1} \alpha_i A^i\vec{b}$ since we assume the dimension of the column space has stopped growing, so $A^p\vec{b}$ is a linear combination of the vectors preceding it. We want to show $\exists \beta_i$ such that $A^{p+1} = \sum_{i=0}^{p-1} \beta_i A^i\vec{b}$*

$$\begin{gathered}A^{p+1}\vec{b} = A(A^p\vec{b}) = \\ A\sum_{i=0}^{p-1} \alpha_i A^i\vec{b} = \sum_{i=0}^{p-1} \alpha_i A^{i+1}\vec{b} = \\ \alpha_{p-1}A^p\vec{b} + \sum_{i=0}^{p-2} \alpha_i A^i\vec{b} = \\ \alpha_{p-1}\sum_{i=0}^{p-1} \alpha_i A^i\vec{b} + \sum_{i=0}^{p-2} \alpha_i A^i\vec{b}\end{gathered}$$

By writing A^{p+1} as a linear combination of $B, AB, ..., A^{p-1}$, we have shown β_i exists, so by induction, all future vectors $p + m \leq k$ will be linear combinations of these same vectors, so the dimension will never grow again and we can be justified in stopping early.

2.2 Stability

The question of stability is whether or not disturbances will over time control a system if left alone. Specifically, it is asking if the disturbances are bounded, will the states be bounded too?

Definition 4 *Bounded Input, Bounded Output (BIBO) stability means if $|\vec{w}[i]| < \epsilon$, then $\exists \delta > 0$ such that $|\vec{x}[i]| < \delta$*

To see where this definition comes into play, consider the following scalar systems:

1. $x[i+1] = u[i] + w[i]$
2. $x[i+1] = \frac{1}{2}x[i] + u[i] + w[i]$ ("Weak" dependence on previous state)
3. $x[i+1] = 2x[i] + u[i] + w[i]$ ("Strong" dependenceon previous state)

System 1: In this system, there is no dependence on the previous state. Thus any disturbances are limited to affecting only the next state and no others.

This makes the system stable because disturbances will not add up. To see this mathematically, pretend $u[i] = 0$ since we want to see how the system evolves without doing anything.

$$|x[i+1]| = |w[i]| < \epsilon, \quad \therefore |x[i]| < \epsilon$$

Thus we have BIBO stability

System 2: Unlike the previous system, this system has a "weak" dependence on previous states. We consider it "weak" because only half of the previous state influences the next state. Intuitively, this system must be BIBO stability then because although disturbances will accumulate, they will be attenuated through an infinite geometric sum. Once again, we will pretend $\vec{u}[i] = 0$ to check this mathematically.

$$|x[i+1]| = \frac{1}{2}x[i] + w[i] < \frac{1}{2}x[i] + \epsilon$$

Unrolling the recursive relationship, we see

$$|x[i+1]| < \epsilon + \frac{1}{2}\epsilon + \frac{1}{4}\epsilon + ... + \left(\frac{1}{2}\right)^i \epsilon$$

$$\lim_{i->\infty} |x[i+1] < \frac{\epsilon}{1-\frac{1}{2}} = 2\epsilon$$

Thus the system is BIBO stability since $|x[i]| < 2\epsilon$

System 3: This system has a "strong" dependence on previous states since the contribution of previous states is doubled each times. Intuitively, this means the system must be unstable because small disturbances will be explode over time. To see this mathematically, once again consider what happens if we apply no inputs.

$$|x[i+1]| = 2x[i] + w[i] < 2x[i] + \epsilon$$

Unrolling the recursion,

$$|x[i+1]| < \epsilon + 2\epsilon + 4\epsilon + ... + 2^i\epsilon$$

$$\lim_{i->\infty} |x[i+1] < \infty$$

Thus $|x[i]|$ has no bound, so the system is not stable.

Note that in the scalar case, as long as $|\lambda| < 1$ for the system $x[i+1] = \lambda x[i] + bu[i] + w[i]$, the system will be stable. Note that this is $\lambda < 1$, not $\lambda \leq 1$. That is because when $\lambda = 1$, we can see the errors will add up. It turns out that in the vector case, for the system $\vec{x}[t+1] = A\vec{x}[t] + B\vec{u}[t] + \vec{w}[t]$, the system will be stable if the eigenvalues of A are less than 1.

Proof 3 *If $\exists\, n\, \vec{v}_i$ such that $A\vec{v}_i = \lambda_i \vec{v}_i$, then changing coordinates to the eigenbasis gives*

$$\tilde{A} = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}$$

This means $\vec{\tilde{x}}[t+1] = \tilde{A}\vec{\tilde{x}}[t] + V^{-1}B\vec{u}[t] + V^{-1}\vec{w}[t]$ where V is the matrix whose columns are A's eigenvectors. This is merely n different scalar cases, but we still have to make sure that $|\vec{w}[t]| < \epsilon \implies |V^{-1}\vec{w}[t]| < \kappa$ where κ is some bound. Lets say $m = V^{-1}_{ij}$ is the largest entry in V^{-1}. In the worst case all $\vec{w}_i[t]$ and all $V_{ij} = m$. Then $\forall i$, $\left[V^{-1}\vec{w}[t]\right]_i \leq nm\epsilon$ where n is the dimension of V, which means $|V^{-1}\vec{w}[t]| < \kappa$ Thus we can say that $\tilde{x}[t]$ is BIBO implies that $\vec{x}[t]$ is BIBO.

2.2.1 Stability in the General Case

Notice that checking stability based on the eigenvalues requires the matrix A to be diagonalizable becase we need to check all of its eigenvalues. This leaves open the question of what happens when A is not diagonalizable, meaning there are not n distinct eigenvalues. In order to resolve this problem, we are going to need a new matrix form called Shurr Upper Triangular Form

Definition 5 *The Shurr Upper Triangular form of a matrix A is*

$$\begin{bmatrix} \lambda_1 & * & * & \dots \\ 0 & \lambda_2 & * & \dots \\ 0 & 0 & \ddots & \\ \vdots & \vdots & & \end{bmatrix}$$

It is an upper triangular matrix with eigenvalues on the diagonal. The eigenvalues may equal = 0.

Shurr form allows us to easily find the characteristic polynomial of the matrix and solve the problem of stability. To see why it helps with stability, consider the following proof.

Proof 4

$$\tilde{x}[t+1] = \tilde{A}\tilde{x}[t] + \tilde{w}[t]$$

If $\tilde{A}$ is in Shurr form, then the nth component is

$$\tilde{x}_n[t+1] = \lambda_n \tilde{x}_n[t] + \tilde{w}_n[t]$$

so if $|\lambda_n| < 1$, then $\tilde{x}_n$ is BIBO stable. Likewise, for the n-1th component

$$\tilde{x}_{n-1}[t+1] = \lambda_{n-1}\tilde{x}_{n-1}[t] + s\tilde{x}_n[t] + \tilde{w}_{n-1}[t]$$

s is one of the $*$ entries in the upper part of Shurr form. It doesn't matter what it is because $\tilde{x}_n[t]$ is BIBO, so $s\tilde{x}_n[t]$ is also BIBO. As a result

$$\tilde{x}_{n-1}[t]$$

will be BIBO if $|\lambda_{n-1}| < 1$ and $|\lambda_n| < 1$ Since the component $\tilde{x}_{n-i}[t+1]$ depends on $\tilde{x}_n[t], \tilde{x}_{n-1}[t], ...,$ as long as $\tilde{x}_n[t], \tilde{x}_{n-1}[t], ...$ are stable, $\tilde{x}_{n-i}[t+1]$ will be stable too. Thus as long as all $|\lambda_i| < 1$, then Shurr form tells us whether or not the system is stable.

Now that we know how it works, we need to know how to find a basis U such that $\tilde{A} = U^{-1}AU$. As a bonus, it would be helpful if U is orthonormal. To make things easier later, we'll define $M_n = A$. Since all matrices have at least one eigenpair, we can choose

$$M\vec{v}_1 = \lambda_1\vec{v}_1, ||\vec{v}_1|| = 1$$

Using Gram-Schmidt, we can now pick $n-1$ vectors $\vec{r}_i$ such that

$$U_1 = \vec{v}_1, \vec{r}_1, ..., \vec{r}_{n-1}$$

is an orthonormal basis. Note, the vectors $\vec{r}_i$ form a $n \times (n-1)$ matrix R_{n-1} Lets look at M_n in the U_1 basis.

$$U_1^{-1}M_nU_1 = \begin{bmatrix} \vec{v}_1^T \\ \\ R_{n-1}^T \end{bmatrix} M_n \begin{bmatrix} \vec{v}_1 & R_{n-1} \end{bmatrix} = \begin{bmatrix} \vec{v}_1^T M_n \vec{v}_1 & \vec{v}_1^T M_n R_{n-1} \\ \\ R_{n-1}^T M_n \vec{v}_1^T & R_{n-1}^T M_n R_{n-1} \end{bmatrix} = \left[\begin{array}{c|c} \lambda_1 & * \\ \hline \\ 0 & R_{n-1}^T M R_{n-1} \end{array}\right]$$

We can see that we are getting closer to where we want to be, so lets continue with the matrix M_{n-1}. We can find a vector of size $n-1$ such that

$$M_{n-1}\vec{v}_2 = \lambda_2\vec{v}_2, ||\vec{v}_2|| = 1$$

Extending this to an orthonormal basis U_2, we get a matrix R_{n-2}

$$\begin{bmatrix} \vec{v}_2^T \\ \\ R_{n-2}^T \end{bmatrix} M_{n-1} \begin{bmatrix} \vec{v}_2 & R_{n-2} \end{bmatrix} = \begin{bmatrix} \vec{v}_2^T M_{n-1} \vec{v}_2 & \vec{v}_2^T M_{n-1} R_{n-2} \\ \\ R_{n-1}^T M_{n-1} \vec{v}_1^T & R_{n-1}^T M_{n-1} R_{n-1} \end{bmatrix} = \left[\begin{array}{c|c} \lambda_2 & * \\ \hline \\ 0 & R_{n-2}^T M_{n-1} R_{n-2} \end{array}\right]$$

We can keep repeating this by saying $M_{n-2} = R_{n-2}^T M_{n-1} R_{n-2}$, but we need to know if our λ_2 has any relation to the eigenvalues of A. Lets say

$$U_2 = \vec{v}_1, \vec{x}, rest$$

is a basis where $\vec{x}$ has some dependence on $\vec{v}_2$. There aren't many options for what $\vec{x}$ and the rest of the basis vectors could be, so let's try.

$$\vec{x} = R_{n-1}\vec{v}_2, \ rest = R_{n-1}R_{n-2}$$

because this will get us the appropriate sized vectors. Notice this basis will also be orthonormal because $\vec{v}_1^T R_{n-1} = 0$ and $\vec{v}_2 R_{n-1}^T R_{n-1} R_{n-1} = \vec{v}_2{}^T R_{n-2} = 0$

Lets see how M_n lookins in the basis U_2 to see if we are getting closer to Shurr form.

$$U_2^-1 M_n U_2 = \begin{bmatrix} \vec{v}_1^T \\ \\ (R_{n-1}\vec{v}_2)^T \\ \\ (R_{n-1}R_{n-2})^T \end{bmatrix} M_n \begin{bmatrix} \vec{v}_1^T & R_{n-1}\vec{v}_2 & R_{n-1}R_{n-2} \end{bmatrix} =$$

$$\begin{bmatrix} \vec{v}_1^T M_n \vec{v}_1 & \vec{v}_1^T M_n R_{n-1}\vec{v}_2 & \vec{v}_1^T M_n R_{n-1}R_{n-2} \\ \\ (R_{n-1}\vec{v}_2)^T M_n \vec{v}_1^T & (R_{n-1}\vec{v}_2)^T M_n R_{n-1}\vec{v}_2 & (R_{n-1}\vec{v}_2)^T M_n R_{n-1}R_{n-2} \\ \\ (R_{n-1}R_{n-2})^T M_n \vec{v}_1 & (R_{n-1}R_{n-2})^T M_n R_{n-1}\vec{v}_2 & (R_{n-1}R_{n-2})^T M_n R_{n-1}R_{n-2} \end{bmatrix} =$$

$$\left[\begin{array}{c|c|c} \lambda_1 & * & * \\ \hline \\ 0 & \lambda_2 & * \\ \hline \\ 0 & 0 & (R_{n-1}R_{n-2})^T M_n R_{n-1}R_{n-2} \\ \vdots & \vdots & \end{array}\right]$$

The λ_2 showed up because

$$\vec{v}_2^T R_{n-1}^T M_n R_{n-1}\vec{v}_2) = \vec{v}_2 M_{n-1}\vec{v}_2 = \lambda_2$$

This tells us we can continue with this process to get to Shurr form. To get to Shurr Form, all we need to do is

```
    Let $M_n = A$
Loop:
    Find $\lambda_i, \vec{v}_i$ such that $M_{n+1-i}\vec{v}_i = \lambda_i \vec{v}_i,\ ||\vec{v}_i|| = 1$
    Find $R_{n-i}$ such that $R_{n-i}^T \vec{v}_i = 0$ and $R_{n-i}^T R_{n-i} = I_{n-1}$
    $M_{n-i} = R_{n-i}^T M_{n+1-i} R_{n-i}$
End Loop
    $U = \vec{v_1},\ R_{n-1}\vec{v}_2,\ R_{n-1}R_{n-2}\vec{v}_2,\ ...,\ (\prod_{j=1}^{k} R_{n-j})\vec{v}_{k+1}$
```

Thus $\tilde{A} = U^T A U$ will give the Shurr Form of the matrix A. Notice that $(U^T A U)^T = U^T A^T U$. This forms the basis for **Spectral Theorem**

Theorem 2 (Spectral Theorem) *Every symmetric matrix with real eigenvalues can be orthogonally diagonalized*

2.3 Closed Loop Control

Lets say we have a system which is modeled by $\vec{x}[t+1] = A\vec{x}[t] + B\vec{u}[t] + \vec{w}[t]$ We must choose our inputs $\vec{u}[t]$ based on only $\vec{x}[t]$ since we can't observe the disturbances. How should we choose our inputs to control the system? If we try $\vec{u}[t] = k\vec{x}[t]$, then

$$\begin{aligned} \vec{x}[t+1] &= A\vec{x}[t] + Bk\vec{x}[t] + \vec{w}[t] \\ \vec{x}[t+1] &= (A + Bk)\vec{x}[t] + \vec{w}[t] \end{aligned}$$

Which means that if we choose K appropriately, we can potentially change the behavior of the system. Consider the following simple scalar case example

$$\begin{aligned} x[t+1] &= 3x[t] + u[t] + w[t] \\ u[t] &= kx[t] \qquad\qquad c \\ x[t+1] &= (3 + k)x[t] + w[t] \end{aligned}$$

Notice that even if the system is initially unstable, we can set our k to make it stable since the domain of $\lambda = 3 + k$ is ∞ What if we have a system $\vec{x}[t+1] = A\vec{x}[t] + \vec{b}u[t] + \vec{w}[t]$ where $A, \vec{b}$ are controllable, but we only get a scalar input $u[t]$. Can we find a set of feedback gains $\vec{f}^T = [f_0, f_1, ..., f_{k-1}]$ such that $u[t] = -\vec{f}^T\vec{x}[t]$ makes $A - \vec{b}\vec{f}^T$ have the eigenvalues we want?

We know how to easily solve scalar cases like $z[t+1] = az[t] + u[t]$, but it is hard to get this from a vector case because we only have a simple input, so we will be limited in how we can set our eigenvalues this way. Since this won't work, consider the folowing recursive system.

$$z[t+1] = a_{k-1}z[t] + a_{k-2}z[t-1] + ... + a_0 z[t-(k-1)] + u[t]$$

This is definetly controllable since our input $u[t]$ could potentially be any value in $\mathbb{R}$. Moreover, if we choose $u[t] = -\sum_{i=0}^{k-1} f_i z[t-(k-1)+i]$, we can create any recurrence relation we want. For example, if we wanted $z[t+1] = d_{k-1}z[t] + d_{k-2}z[t-1] + ... + d_0 z[t-(k-1)]$, then $f_i = a_i - d_i$ Writing this in vector form, we get

$$\vec{z}[t+1] = A_z z[t] + \vec{b}u[t]$$

$$\vec{z}[t] = \begin{bmatrix} z[t-(k-1)] \\ \vdots \\ z[t-2] \\ z[t-1] \\ z[t] \end{bmatrix}$$

Notice that the vector $\vec{z}[t]$ here is different from the scalar $z[t]$ which it is made up of. Each "state" here is a scalar quantity, but to represent the recursive relationship properly, we collect all of the previous states in a vector to define the next state.

Given our definition of $\vec{z}[t]$ (which are the scalar values of previous states), we can fill in values for A_z and $\vec{b}$

$$A_z = \begin{bmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 0 & 1 & ... & 0 \\ \vdots & & & & \ddots & \vdots \\ a_0 & a_1 & a_2 & ... & & a_{k-1} \end{bmatrix}, \vec{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

To understand where these come from, thing of the A_z matrix "scrolling down" the state vector $\vec{z}[t]$ to get how it influences $\vec{z}[t+1]$. All the rows except the last are 0's and 1's since the previous states can't change, but the last row determines the next state. Notice that all entries in the b-vector are 0 except the last since we can't affect the past with present control. We can assume $a_0 \neq 0$ since we defined the recurrence to have length k. Breaking A_z into its block components gives us some more insight to its structure.

$$A_z = \left[\begin{array}{c|c} 0 & I_{k-1} \\ \hline \multicolumn{2}{c}{\vec{a}^T} \end{array}\right]$$

We can check through multiplication that this system will be controllable. When the A matrix of a system is in this form, we say it is in **Controllable Canonical Form**.

Let's try finding the eigenvalues of A_z. If A_z is large, it'll be hard to calculate the characteristic polynomial, so instead lets guess their form and figure them out that way.

$$\vec{v}_\lambda = \begin{bmatrix} 1 \\ * \\ * \\ \vdots \end{bmatrix}$$

Where * is any element. We know $A_z\vec{v}_\lambda = \lambda v_\lambda$, and since A_z scrolls the vector down, we can see that the second * must be λ. Applying this same logic to the second *, it must be λ^2. Continuing this, we get

$$\vec{v}_\lambda = \begin{bmatrix} 1 \\ \lambda \\ \lambda^2 \\ \vdots \\ \lambda^{k-1} \end{bmatrix}$$

If we wanted to find the characteristic polynomial, we could look at the last entry in $A_z\vec{v}_\lambda$

$$\textstyle\sum_{i=0}^{k-1} a_i\lambda_i = \lambda * \lambda^{k-1} = \lambda_k$$
$$\textstyle\lambda_k - \sum_{i=0}^{k-1} a_i\lambda_i = \lambda * \lambda^{k-1} = 0$$

Now that we know about this matrix form, the question becomes: can we transform any controllable system with matrix A into controllable canonical form? (i.e can we make $\vec{x}[t+1] = A\vec{x}[t] + \vec{b}u[t]$ into $\vec{z}[t+1] = A_z\vec{z}[t] + \vec{b}_z u[t]$) We know that $A\vec{b}, A^2\vec{b}, ...A^{k-1}\vec{b}$ are linearly independent since our system is controllable. Let this define basis $\mathbb{G}$. If we write the matrix A in the basis of G, we get $\tilde{A} = G^{-1}AG$ The naming choice is very appropriate here because this matrix multiplication will be painful by hand, so lets look for another way to get $\tilde{A}$.

Remember that any linear transformation is defined by what it does to the basis vectors. We know

$$\tilde{b} = G^{-1}\vec{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

since $\vec{b}$ is the first basis vector of $\mathbb{G}$. Thus,

$$\tilde{A}\tilde{\vec{b}} = G^{-1}A\vec{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}, \tilde{A}^2\tilde{\vec{b}} = G^{-1}A^2\vec{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} ...$$

This gives us k columns of $\tilde{A}$. For the last column must be a linear combination of all the basis vectors, so $\tilde{A}^k\tilde{\vec{b}} = \sum_{i=0}^{k-1} \alpha_i \tilde{A}^i\tilde{\vec{b}}$ This means

$$\tilde{A} = \begin{bmatrix} 0 & 0 & ... & \alpha_0 \\ 1 & 0 & ... & \alpha_1 \\ 0 & 1 & ... & \alpha_2 \\ \vdots & & \ddots & \vdots \\ 0 & ... & 1 & \alpha_{k-1} \end{bmatrix}$$

which is A_z^T. Now we have a basis $\mathbb{G}$ with $(\tilde{A}, \tilde{b})$, but we want a basis $\mathbb{H}$ with $(\tilde{A}^T, \vec{b}_z)$ We want $\tilde{A}^T, \vec{b}_z$ to be controllable, which means $\vec{b}_z, \tilde{A}^T\vec{b}_z, ..., (\tilde{A}^T)^{k-1}\vec{b}_z$ are linearly independent.

Let's construct a matrix H which will take us from a coordinates in basis $\mathbb{H}$ to coordinates in the basis $\mathbb{G}$. **Important:** the matrix

$$\left[\begin{array}{c|c|c|c} \vec{b}_z & \tilde{A}^T\vec{b}_z & ... & (\tilde{A}^T)^{k-1}\vec{b}_z \end{array}\right]$$

is **not** the matrix H. To compute H, we have to see how it transforms the $\mathbb{G}$ basis vectors.

Once we have the matrix H, to get $(\tilde{A}^T, \vec{b}_z)$ from $(A, \vec{b})$, we can apply the change of coordinates $G^{-1}H$ so $\vec{x}_z[t] = HG^{-1}\vec{x}[t]$ and put our entire system into controllable canonical form.

The reason we can't find HG^{-1} in a single-step and must go through an intermediary basis is because we might not know what our system will look like in canonical form.

2.4 Linearization

Up until now, we have been assuming our systems are linear. To understand what this means, we must first understand what linearity means.

First, let's define what we know as a function.

Definition 6 *A function is a simple mapping from a domain to a range (i.e y(t)=L(u(t))). To evaluate y(t), we only need to know u(t), making the function "memory-less"*

Definition 7 *A function $f(x)$ is linear iff*

- $\forall\alpha, \forall x \in \mathbb{R}, f(\alpha x) = \alpha f(x)$ *(Scaling)*
- $\forall u, \forall v \in \mathbb{R}, f(u+v) = f(u) + f(v)$ *(Superposition)*

Notice that by this definition, any linear function must necessarily have $f(0) = 0$ since otherwise with $\alpha = 1, x = 0, f(1 * 0) = f(0) = 1 * f(0) \neq 0$ These conditions can be simplified down into a single expression $f(\alpha u + v) = \alpha f(u) + f(v), \forall u, \forall v, \forall \alpha \in \mathbb{R}$ If we can find a single setting of (u, v, α) for which this equation does not hold, then our function is not linear.

If we have a scalar nonlinear function we can linearize it in a simple way. Remember that the Taylor series expansion of a function around a certain point is

$$f(x) = \frac{f(c)}{0!} + \frac{f'(c)}{1!}(x-c) + \frac{f''(c)}{2!}(x-c)^2 + ... = \sum_{i=0}^{\infty} \frac{f^{(i)}}{i!}(x-c)^i$$

where c is some expansion point we choose. Lets say we choose some point x^* and decide to linearize around that point. The tangent line to the function is

$$f_L(x) = f'(x^*)(x - x^*) + f(x^*)$$

However, $f(0) \neq 0$, so this tangent line is still not linear. To linearize it, we can change coordinates.

$$\begin{gathered} m = \tfrac{df}{dx}|_x \\ f(x^* + \delta x) = f(x^*) + m\delta x \\ \delta y = f(x^* + \delta x) - f(x^*) \\ f(x^* + \delta x) = f(x^*) + \delta y = f(x^*) + m\delta x \\ \delta y = m\delta x \end{gathered}$$

Thus the linearized expression is in terms of the "error" δy, a.k.a how far it deviates from what it was at the point of linearization. While this makes sense for functions, how does it apply to systems. Systems are more than simple functions, they are functionals.

Definition 8 *A functional y(t) = L{u(t)} is a function which requires knowing the entire u in order to evaluate it.*

A good example of a functional is

$$L\{u(t)\} = \int_{\tau-1}^{\tau} u(t)d\tau$$

The definition of linearity for functionals is virtually the same as for a normal function.

Definition 9 *A system $L\{\}$ is linear iff*

- $\forall \alpha \in \mathbb{R}, \forall u(t), L\{\alpha u(t)\} = \alpha L\{u(t)\}$ *(Scaling)*
- $\forall u(t), \forall v(t), L\{u(t) + v(t)\} = L\{u(t)\} + L\{v(t)\}$ *(Superposition)*

With this definition, we can now formally define how to linearize a non-linear system. Take the system $\frac{d}{dt}x(t) = f(x(t)) + bu(t)$ where f is a nonlinear function and b is a scalar multiplier

To linearize the system

1. Choose a constant with time input $u(t) = u^*$
2. Find a solution $x(t) = x^*$ to the system
$$\frac{dx^*}{dt} = f(x^*) + bu^* \implies f(x^*) = -bu^*$$
3. Linearize $f(x)$ around x^*
$$f(x) \approx f(x^*) + \frac{df}{dx}|_{x^*}(x - x^*)$$
4. Use the linearization of f to make the differential equation linear
$$\tfrac{dx}{dt} = f(x^*) + \tfrac{df}{dx}|_{x^*}(x(t) - x^*) + bu(t)$$
$$\text{Let } \delta x = x(t) - x^*, \delta u = u(t) - u^*$$
$$\tfrac{d}{dt}(\delta x + x^*) = f(x^*) + \tfrac{df}{dx}|_{x^*}\delta x + b(u^* + \delta u)$$
$$\tfrac{d}{dt}\delta x = \overset{0}{\cancel{f(x^*) + bu^*}} + \tfrac{df}{dx}|_{x^*}\delta x + b\delta u(t)$$
$$\tfrac{d}{dt}\delta x = \tfrac{df}{dx}|_{x^*}\delta x + b\delta u(t)$$

Note that this only works assuming that δx is small, so when linearizing our systems, we must check that this property is true otherwise our linearization may be invalid. To see why, consider the system $\frac{d}{dt} = -sin(x)$. Try linearizing about $x^* = 0$.

$$\tfrac{d}{dt}\delta x = -\delta x$$

$$\therefore \delta x = \delta x(0)e^{-t}$$

Since δx is the "error" of our approximation, we can see that because $\delta x = \delta x(0)e^{-t}$ will grow smaller over time, meaning the system is stable, and our approximation is legimitate. However, if we linearize about $x^* = \pi$

$$\tfrac{d}{dt}\delta x = \delta x$$

$$\therefore \delta x = \delta x(0)e^{t}$$

and this will grow very large as time goes on, so our approximate will be illegitimate.

These concepts of linearization extend beyond simple 1D functions. For example, the linearization of a 2D function is

$$f(x^* + \delta x, u^* + \delta u) \approx f(x^*, u^*) + \frac{\partial f}{\partial x}|_{x^*,u^*}\delta x + \frac{\partial f}{\partial u}|_{x^*,u^*}\delta u$$

We can further generalize the idea of linearization to vector functions.

$$f(\vec{x}) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} \text{ where } \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

If we linearize each of the functions inside the vector, we get

$$f(\vec{x^*} + \delta\vec{x^*}) = \begin{bmatrix} f_1(x_1^* + \delta x_1, x_2^* + \delta x_2) \\ f_2(x_1^* + \delta x_1, x_2^* + \delta x_2) \end{bmatrix}$$

$$f_1(x_1^* + \delta x_1, x_2^* + \delta x_2) \approx f_1(x_1^*, x_2^*) + \tfrac{\partial f_1}{\partial x_1}|_{x_1^*,x_2^*}\delta x_1 + \tfrac{\partial f_1}{\partial x_2}|_{x_1^*,x_2^*}\delta x_2$$

$$f_2(x_1^* + \delta x_1, x_2^* + \delta x_2) \approx f_2(x_1^*, x_2^*) + \tfrac{\partial f_2}{\partial x_1}|_{x_1^*,x_2^*}\delta x_1 + \tfrac{\partial f_1}{\partial x_2}|_{x_1^*,x_2^*}\delta x_2$$

$$\therefore f(\vec{x^*} + \delta\vec{x^*}) = \begin{bmatrix} f_1(x_1^* + \delta x_1, x_2^* + \delta x_2) \\ f_2(x_1^* + \delta x_1, x_2^* + \delta x_2) \end{bmatrix} = \begin{bmatrix} f_1(x_1^*, x_2^*) \\ f_2(x_1^*, x_2^*) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \end{bmatrix}|_{x_1^*,x_2^*} \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix}$$

The matrix with partial derivatives is known as the **Jacobian** and is denoted either ∇f or J Thus a succinct form the linearization of a vector system is

$$f(\vec{x^*} + \delta\vec{x}) = f(\vec{x^*}) + J_x f(\delta\vec{x})$$

For multiargument functions,

$$f(\vec{x^*} + \delta\vec{x}, \vec{u^*} + \delta\vec{u}) = f(\vec{x^*}, \vec{u^*}) + J_x\delta\vec{x} + J_u\delta\vec{u}$$