1 Finding Null Spaces

(a) 3

For any 3x5 matrix, the column vectors are 3x1 vectors, so they would at most span $(\mathbb{R}^3, \mathbb{R})$. Moreover, we have that $[1\ 0\ 0]^T, [0\ 1\ 0]^T, [0\ 0\ 1]^T$ is a Basis for $\mathbb{R}^3$, by definition of Basis, so this means that the maximum possible number of linearly independent column vectors is 3.

(b). Colspace (A) = span $\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} \right) = (\mathbb{R}^2, \mathbb{R})$.

$\boxed{2}$ unique vectors are required to span the column space of A.

(c). By definition, $A\vec{x} = 0$. so $\begin{bmatrix} 1 & 1 & 0 & -2 & 3 \\ 0 & 0 & 2 & -2 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$, which turn into

this augmented matrix: $\left[\begin{array}{ccccc|c} 1 & 1 & 0 & -2 & 3 & 0 \\ 0 & 0 & 2 & -2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$. Divide R_2 by 2: $\left[\begin{array}{ccccc|c} 1 & 1 & 0 & -2 & 3 & 0 \\ 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$

So we have: $\begin{cases} x_1 + x_2 - 2x_4 + 3x_5 = 0 \\ x_3 - x_4 + x_5 = 0 \end{cases} \Rightarrow \begin{array}{l} x_1 = -x_2 + 2x_4 - 3x_5 \\ x_3 = x_4 - x_5 \end{array}$

$$\text{so, } \vec{x} = \begin{bmatrix} -x_2 + 2x_4 - 3x_5 \\ x_2 \\ x_4 - x_5 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} x_2 + \begin{bmatrix} 2 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} x_4 + \begin{bmatrix} -3 \\ 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} x_5$$

Thus, vectors that span the null space of A: $\left(\begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -3 \\ 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \right)$.

and the dimension is $\boxed{3}$.

(e). By definition, $C\vec{x} = 0$, so $\begin{bmatrix} 2 & -4 & 4 & 8 \\ 1 & -2 & 3 & 6 \\ 2 & -4 & 5 & 10 \\ 3 & -6 & 7 & 14 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$. equivalent to this augmented matrix:

$\left[\begin{array}{cccc|c} 2 & -4 & 4 & 8 & 0 \\ 1 & -2 & 3 & 6 & 0 \\ 2 & -4 & 5 & 10 & 0 \\ 3 & -6 & 7 & 14 & 0 \end{array}\right]$. R_4: $3 \cdot R_3 - 2R_4$. R_3: $2 \cdot R_2 - R_3$, R_2: Subtract $\frac{1}{2} \cdot R_1$, which gives us $\Rightarrow \left[\begin{array}{cccc|c} 2 & -4 & 4 & 8 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 2 & 0 \end{array}\right]$

Then, R_1: Divide by 2
R_3: Subtract R_2
R_4: Subtract R_2. $\Rightarrow \left[\begin{array}{cccc|c} 1 & -2 & 2 & 4 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right]$, So: $\begin{array}{l} x_1 - 2x_2 + 2x_3 + 4x_4 = 0 \\ x_3 - 2x_4 = 0. \end{array}$

so, $x_3 = 2x_4$ and so $x_1 = 2x_2 - 2x_3 + 4x_4 = 2x_2 - 4x_4 + 4x_4 = 2x_2$.

$$\text{So, } \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2x_2 \\ x_2 \\ 2x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix} x_2 + \begin{bmatrix} 0 \\ 0 \\ 2 \\ 1 \end{bmatrix} x_4.$$

Thus, the vectors that span N(C) is . $\left(\begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 2 \\ 1 \end{bmatrix} \right)$

2. Technical Issues.

(a). Suppose originally, A is (x_a, y_a). Using my Basis, so $\vec{a_1} = x_a \vec{v_1} + x_b \vec{v_2}$

Since in my view, $\vec{a_1} = \begin{bmatrix} -2 \\ 3 \end{bmatrix} = x_a \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + y_a \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ so $x_a = -2$, $y_a = 3$

and we have that $A(-2, 3)$ originally. Similarly, we obtain $B(0, 2)$, $C(2, -3)$, $D(0, -2)$ before applying my Basis.

Now, consider my partner's view.

$$\vec{a_2} = -2 \cdot \vec{u_1} + 3 \cdot \vec{u_2} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{Eq (1)}$$
$$\vec{b_2} = 0 \cdot \vec{u_1} + 2 \cdot \vec{u_2} = \begin{bmatrix} 0 \\ -2 \end{bmatrix} \quad (2)$$
$$\vec{c_2} = 2 \cdot \vec{u_1} + (-3) \cdot \vec{u_2} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad (3)$$
$$\vec{d_2} = 0 \cdot \vec{u_1} + (-2) \cdot \vec{u_2} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad (4)$$

So we can obtain that $\vec{u_2} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ from Eq (2), and then from Eq (1), we have $\vec{u_1} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{3}{2} \end{bmatrix}$

Thus, $\boxed{\vec{u_1} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{3}{2} \end{bmatrix}, \quad \vec{u_2} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}}$

(b) Since we have that $A_{v \to u} \cdot \vec{V} = \vec{U}$, so $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & 0 \\ -\frac{3}{2} & -1 \end{bmatrix}$

so we obtain equations:

$$\begin{aligned} 1 \cdot a_{11} + 0 \cdot a_{12} &= -\tfrac{1}{2} \\ 0 \cdot a_{11} + 1 \cdot a_{12} &= 0 \\ 1 \cdot a_{21} + 0 \cdot a_{22} &= -\tfrac{3}{2} \\ 0 \cdot a_{21} + 1 \cdot a_{22} &= -1. \end{aligned} \Rightarrow \begin{aligned} a_{11} &= -\tfrac{1}{2} \\ a_{12} &= 0 \\ a_{21} &= -\tfrac{3}{2} \\ a_{22} &= -1. \end{aligned}$$

Thus, $A_{v \to u} = \boxed{\begin{bmatrix} -\frac{1}{2} & 0 \\ -\frac{3}{2} & -1 \end{bmatrix}}$

(c) Using the given information, so

$$-2 \cdot \vec{w_1} + 3 \cdot \vec{w_2} = \overrightarrow{A_{new}} = \begin{bmatrix} -2 \\ -3 \end{bmatrix} \quad (5)$$
$$0 \cdot \vec{w_1} + 2 \cdot \vec{w_2} = \overrightarrow{B_{new}} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad (6)$$
$$2 \cdot \vec{w_1} + (-3) \cdot \vec{w_2} = \overrightarrow{C_{new}} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad (7)$$
$$0 \cdot \vec{w_1} + (-2) \cdot \vec{w_2} = \overrightarrow{D_{new}} = \begin{bmatrix} 0 \\ -2 \end{bmatrix} \quad (8).$$

From Eq (6), we can solve that $\vec{w_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, substitute in Eq (5), so $\vec{w_1} = \begin{bmatrix} 1 \\ -3 \end{bmatrix}$

Thus, the new basis vectors are $\boxed{\vec{w_1} = \begin{bmatrix} 1 \\ -3 \end{bmatrix}, \quad \vec{w_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}}$

(d). Since we have that $A_{w\to u}\cdot \vec{W} = \vec{U}$, so $\begin{bmatrix} a_{w1} & a_{w2} \\ a_{w3} & a_{w4} \end{bmatrix}\cdot\begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & 0 \\ -\frac{3}{2} & -1 \end{bmatrix}$

so we obtain equations:

$$\begin{aligned} 1\cdot a_{w1} - 3a_{w2} &= -\frac{1}{2} \\ 0\cdot a_{w1} + 1\cdot a_{w2} &= 0. \\ 1\cdot a_{w3} - 3\cdot a_{w4} &= -\frac{3}{2} \\ 0\cdot a_{w3} + 1\cdot a_{w4} &= -1 \end{aligned} \Rightarrow. \quad \begin{aligned} a_{w1} &= -\frac{1}{2} \\ a_{w2} &= 0 \\ a_{w3} &= -\frac{9}{2} \\ a_{w4} &= -1 \end{aligned}$$

Thus, $A_{w\to u} = \boxed{\begin{bmatrix} -\frac{1}{2} & 0 \\ -\frac{9}{2} & -1 \end{bmatrix}}$

3. Traffic Flows

(a). Given that $\begin{cases} t_1 + t_3 = 0 & (1) \\ t_2 - t_1 = 0 & (2) \\ -t_3 - t_2 = 0 & (3) \end{cases}$ and that $t_1 = 10$, so substituting into Eq (2) and (3),

We have that $\boxed{t_2 = 10,\ t_3 = -10}$.

(b). Similar to part (a), we obtain this set of equations from the graph: $\boxed{\begin{cases} t_1 + t_3 - t_4 = 0 & (1) \\ -t_1 + t_2 = 0 & (2) \\ -t_2 - t_3 + t_5 = 0 & (3) \\ t_4 - t_5 = 0. & (4) \end{cases}}$

→ $\boxed{\text{Yes.}}$ it is possible with the Berkeley suggestion because, we can rewrite Eq (1),(2),(4) so that $t_2 = t_1$, $t_3 = -t_1 + t_4$, $t_5 = t_4$.

Thus, given the measurements of t_1 and t_4, we can determine all traffic flows, $[t_1, t_2, t_3, t_4, t_5]^T$.

→ $\boxed{\text{No}}$ it's not possible with the Stanford suggestion because we can write our system of linear equations as $\begin{cases} t_3 - t_4 = -t_1 \\ 0 = -t_1 + t_2 \\ -t_3 + t_5 = t_2 \\ t_4 - t_5 = 0. \end{cases} \Leftrightarrow \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} t_3 \\ t_4 \\ t_5 \end{bmatrix} = \begin{bmatrix} -t_1 \\ -t_1 + t_2 \\ t_2 \\ 0 \end{bmatrix}$

which can be turned into: $\left[\begin{array}{ccc|c} 1 & -1 & 0 & -t_1 \\ 0 & 0 & 0 & -t_1+t_2 \\ -1 & 0 & 1 & t_2 \\ 0 & 1 & -1 & 0 \end{array}\right]$ If only the measurements t_1, t_2 are given.

R_3: Add R_1.

Swap R_2 and R_4.

$\Rightarrow \left[\begin{array}{ccc|c} 1 & -1 & 0 & -t_1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & -t_1+t_2 \\ 0 & 0 & 0 & -t_1+t_2 \end{array}\right]$ R_2: Add R_3. $\Rightarrow \left[\begin{array}{ccc|c} 1 & -1 & 0 & -t_1 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -t_1+t_2 \\ 0 & 0 & 0 & -t_1+t_2 \end{array}\right]$ Now we have two rows of 0s.

If $(-t_1+t_2) \neq 0$, then we have no solutions; if $(-t_1+t_2) = 0$, then we only have two independent equations with 3 variables, which means that we can't deduce a unique solution to determine all traffic flows $[t_1, t_2, t_3, t_4, t_5]^T$.

(c).

From the graph, we can construct and from the system of equation in part (b), so, $\boxed{B = \begin{bmatrix} 1 & 0 & 1 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}}$

(d). From the system of equations in part (b), so $\begin{cases} t_3 = -t_1 + t_4 \\ t_2 = t_1. \\ t_5 = t_4. \end{cases}$ and with $t_1 = \alpha$, $t_4 = \beta$

Thus, $\vec{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{bmatrix} = \begin{bmatrix} t_1 \\ t_1 \\ -t_1 + t_4 \\ t_4 \\ t_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} \alpha + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \beta$ Thus, the subspace can be expressed as $\boxed{\left(\operatorname{span}\left(\begin{bmatrix} 1 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}\right), \mathbb{R}\right)}$. Justification provided in part (b).

Scanned with CamScanner

(e). For $B\vec{t} = \vec{0}$, we can transform this system of linear equations into an augmented matrix with our calculated matrix B and the definition of matrix-vector multiplication, So:

$$\left[\begin{array}{ccccc|c} 1 & 0 & 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{array}\right]$$

R_3: Add R_1 and R_2.
then R_2: Add R_1 $\Rightarrow$

$$\left[\begin{array}{ccccc|c} 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{array}\right]$$

R_4: Add R_3 $\Rightarrow$.
R_3: $\times(-1)$

$$\left[\begin{array}{ccccc|c} 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

So, t_3, t_5 are free variables and we have that

$$\begin{cases} t_1 + t_3 - t_5 = 0 & (1) \\ t_2 + t_3 - t_4 = 0 & (2) \\ t_4 - t_5 = 0 \Rightarrow t_4 = t_5 & (3) \end{cases}$$

Substitute Eq(3) into Eq(1)(2) and we have: $t_1 = -t_3 + t_5$ and $t_2 = -t_3 + t_4 = -t_3 + t_5$

Thus, $\text{Nullspace}(B) = \begin{bmatrix} -t_3+t_5 \\ -t_3+t_5 \\ t_3 \\ t_5 \\ t_5 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} t_3 + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} t_5$.

which gives us that the dimension of Nullspace(B) is $\boxed{2.}$

Scanned with CamScanner

and a basis for it is $\boxed{\left\{ \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right\}}$

$\boxed{\text{Yes,}}$ it does match my answer in part (d) since $\begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$,

So the two basis representations are essentially the same

(f). $\boxed{\text{No,}}$ this doesn't. Consider the network of Figure 5 used in part (b), graph G.
We have shown that its incidence matrix B_G has a 2-dimensional null space.
Yet, if we measure just t_1 and t_2 (as the Stanford suggestion), we have proved in part (b) that we can't recover the exact (unique) flows.

(g). In essence, the null space of M and the null space of B_G must have $\boxed{\text{one and only one intersection}}$.
In other words, if we concatenate M and B_G in axis = 0, then the null space of the resulting matrix should have a unique solution.

4. Segway Tours

(a). Since $\vec{x}[n+1] = A\vec{x}[n] + \vec{b}u[n]$, so $\vec{x}[1] = \boxed{A\vec{x}[0] + \vec{b}u[0]}$

(b). Similarly, $\vec{x}[2] = A\vec{x}[1] + \vec{b}u[1] = A(A\vec{x}[0] + \vec{b}u[0]) + \vec{b}u[1]$

so, $\boxed{\vec{x}[2] = A^2\vec{x}[0] + A\vec{b}u[0] + \vec{b}u[1]}$

and so $\vec{x}[3] = A\vec{x}[2] + \vec{b}u[2] = A(A^2\vec{x}[0] + A\vec{b}u[0] + \vec{b}u[1]) + \vec{b}u[2]$

so, $\boxed{\vec{x}[3] = A^3\vec{x}[0] + A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2]}$

and $\vec{x}[4] = A\vec{x}[3] + \vec{b}u[3] = A(A^3\vec{x}[0] + A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2]) + \vec{b}u[3]$

so, $\boxed{\vec{x}[4] = A^4\vec{x}[0] + A^3\vec{b}u[0] + A^2\vec{b}u[1] + A\vec{b}u[2] + \vec{b}u[3]}$

(c). Thus, we can derive that:

$$\vec{x}[N] = A^N\vec{x}[0] + A^{N-1}\vec{b}u[0] + A^{N-2}\vec{b}u[1] + \cdots + A^2\vec{b}[N-3] + A\vec{b}[N-2] + \vec{b}u[N-1]$$

(d). Since we have that $\vec{x}[2] - A^2\vec{x}[0] = A\vec{b}u[0] + \vec{b}u[1]$.

and that we wish to reach $\vec{x}_f = \vec{0}$ in two time steps, so that $\vec{x}[2] = \vec{0}$.

So we have a linear equation to plug into iPython notebook,

and after Gaussian Elimination, we obtain:

$$\left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{array}\right]$$

Since we have a row of 0s with the right side being $1 \neq 0$, so there's no solution, which means that $\boxed{\text{No, I can't}}$ reach $\vec{x}_f$ in two time steps.

(e) Similarly, since we have that $\vec{x}[3] = A^3\vec{x}[0] + A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2]$

and that we wish to have $\vec{x}[3] = \vec{0}$, so $A^2\vec{b}u[0] + A\vec{b}u[1] + \vec{b}u[2] = -A^3\vec{x}[0]$

Using iPython notebook again,

after Gaussian Elimination, we have:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

Again, since we have a row of 0s with the right side being $1 \neq 0$,

so there's no solution, which means that $\boxed{\text{No, I can't}}$ again.

(f). Similarly, we have: $\vec{x}[4] = A^4\vec{x}[0] + A^3\vec{b}u[0] + A^2\vec{b}u[1] + A\vec{b}u[2] + \vec{b}[3]$.

with $\vec{x}[4] = \vec{0}$, and using iPython notebook to solve by Gaussian Elimination, we have:

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & -57.054 \\ 0 & 1 & 0 & 0 & 15.915 \\ 0 & 0 & 1 & 0 & -28.298 \\ 0 & 0 & 0 & 1 & 4.191 \end{array}\right]$$

Thus, $\boxed{\text{Yes, I can.}}$ with $u[0] = -57.054$, $u[1] = 15.915$

$u[2] = -28.298$, $u[3] = 4.191$.

(g). As found and stated in part f. via Gaussian Elimination (with iPython notebooks).
the control inputs are:

$$\boxed{\begin{cases} u[0] = -13.249 \\ u[1] = 23.733 \\ u[2] = -11.572 \\ u[3] = 1.465 \end{cases}}$$

Verified by the simulation.

(h). The condition is that span$\{\vec{b}, A\vec{b}, A^2\vec{b}, \cdots, A^{N-1}\vec{b}\}$ needs to contain $-A^N\vec{x}[0]$.
Since we have for $\vec{x}[N] = A^N\vec{x}[0] + A^{N-1}\vec{b}u[0] + A^{N-2}\vec{b}u[1] + \cdots + A\vec{b}u[N-2] + \vec{b}u[N-1]$
and we wish to have $\vec{x}[N] = \vec{x}_f = \vec{0}$.

Thus, we have. $$\begin{bmatrix} | & | & \cdots & | & | \\ A^{N-1}\vec{b} & A^{N-2}\vec{b} & \cdots & A\vec{b} & \vec{b} \\ | & | & \cdots & | & | \end{bmatrix} \cdot \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[N-2] \\ u[N-1] \end{bmatrix} = \begin{bmatrix} | \\ -A^N\vec{x}[0] \\ | \end{bmatrix}$$

which, for this to have solution, we need to have that

$$\boxed{\text{span}\{\vec{b}, A\vec{b}, \cdots, A^{N-1}\vec{b}\} \text{ contains } (-A^N x[0])}$$

(i). Similarly. this time we have: $A^{N-1}\vec{b}u[0] + \cdots + A\vec{b}u[N-2] + \vec{b}u[N-1] = \vec{x}[N] - A^N\vec{x}[0]$.
Since $\vec{x}[N]$ is any valid state vector (being 4×1 vector), so this means that $(\vec{x}[N] - A^N\vec{x}[0])$ could be any vector in $\mathbb{R}^{4\times1}$.
which means that, since $$\begin{bmatrix} | & | & \cdots & | & | \\ A^{N-1}\vec{b} & A^{N-2}\vec{b} & \cdots & A\vec{b} & \vec{b} \\ | & | & \cdots & | & | \end{bmatrix} \cdot \begin{bmatrix} u[0] \\ \vdots \\ u[N-2] \\ u[N-1] \end{bmatrix} = \begin{bmatrix} | \\ \vec{x}[N] - A^N\vec{x}[0] \\ | \end{bmatrix}$$

So this implies that $\boxed{\text{span}\{\vec{b}, A\vec{b}, \cdots, A^{N-1}\vec{b}\} = (\mathbb{R}^4, \mathbb{R}).}$

Scanned with CamScanner

6. Homework Process and Study Group

I worked alone without getting any help, except asking questions and reading posts (especially answers from the GSIs) on Piazza as well as reading the Notes of the course.

EE16A: Homework 4

Problem 6: Segway Tours

Run the following block of code first to get all the dependencies.

In [1]:
```
# %load gauss_elim.py
from gauss_elim import gauss_elim
```

In [2]:
```
from numpy import zeros, cos, sin, arange, around, hstack
from matplotlib import pyplot as plt
from matplotlib import animation
from matplotlib.patches import Rectangle
import numpy as np
from scipy.interpolate import interp1d
import scipy as sp
```

Dynamics

In [6]:
```
# Dynamics: state to state
A = np.array([[1, 0.05, -.01, 0],
              [0, 0.22, -.17, -.01],
              [0, 0.1, 1.14, 0.10],
              [0, 1.66, 2.85, 1.14]]);
# Control to state
b = np.array([.01, .21, -.03, -0.44])
nr_states = b.shape[0]

# Initial state
state0 = np.array([-0.3853493, 6.1032227, 0.8120005, -14])

# Final (terminal state)
stateFinal = np.array([0, 0, 0, 0])
```

Example of Gaussian Elimination

In [7]:
```
# System of example equations:
# x + y +  5z = 7
# x + 4y - z  = 4
# 3x + y - z  = 4

Q = np.zeros([3,3])

# Matrix construction by specifying column vectors
Q[:,0] = np.array([1, 1, 3]) # coefficients of x
Q[:,1] = np.array([1, 4, 1]) # coefficients of y
Q[:,2] = np.array([5, -1, -1]) # coefficients of z

m = np.array([7, 4, 4])

# Augmented Matrix for system of equations
Q_aug = np.c_[Q, m]

print('Augmented matrix for part f:')
print(Q_aug,'\n')

print('Matrix after Gaussian elimination:')
print(gauss_elim(Q_aug))
```

```
Augmented matrix for part f:
[[ 1.   1.   5.   7.]
 [ 1.   4. -1.   4.]
 [ 3.   1. -1.   4.]]

Matrix after Gaussian elimination:
[[1.    0.    0.    1.35]
 [0.    1.    0.    0.9 ]
 [0.    0.    1.    0.95]]
```

Part (d)

In [91]:
```
# Compute the A^2
A2 = np.dot(A,A)
values = (-1) * np.dot(A2, state0)

coeff_0 = np.dot(A, b).reshape(4, 1)
coeff_1 = b.reshape(4, 1)

variables = np.concatenate((coeff_0, coeff_1), 1).reshape(4, 2)
augmented = np.c_[variables, values]

print("The augmented matrix is:\n", augmented, "\n")
print("After Gaussian Elimination:\n", gauss_elim(augmented))
```

```
The augmented matrix is:
 [[ 0.0208      0.01        0.02243475]
 [ 0.0557      0.21       -0.30785117]
 [-0.0572     -0.03        0.06193476]
 [-0.2385     -0.44        1.38671326]]

After Gaussian Elimination:
 [[ 1.  0.  0.]
 [ 0.  1.  0.]
 [-0. -0.  1.]
 [ 0.  0.  0.]]
```

Part (e)

In [92]:
```
# Compute the A^3
A3 = np.dot(A2,A)

values = (-1) * np.dot(A3, state0)

coeff_0 = np.dot(A2, b).reshape(4, 1)
coeff_1 = np.dot(A, b).reshape(4, 1)
coeff_2 = b.reshape(4, 1)

variables = np.concatenate((coeff_0, coeff_1, coeff_2), 1).reshape(4, 
augmented = np.c_[variables, values]

print("The augmented matrix is:\n", augmented, "\n")
print("After Gaussian Elimination:\n", gauss_elim(augmented))
```

```
The augmented matrix is:
 [[ 0.024157    0.0208      0.01        0.00642285]
 [ 0.024363    0.0557      0.21       -0.0921233 ]
 [-0.083488   -0.0572     -0.03        0.17849184]
 [-0.342448   -0.2385     -0.44        1.24633424]]

After Gaussian Elimination:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

Part (f)

In [96]:
```
# Compute the A^4
A4 = np.dot(A3,A)

values = (-1) * np.dot(A4, state0)

coeff_0 = np.dot(A3, b).reshape(4, 1)
coeff_1 = np.dot(A2, b).reshape(4, 1)
coeff_2 = np.dot(A, b).reshape(4, 1)
coeff_3 = b.reshape(4, 1)

variables = np.concatenate((coeff_0, coeff_1, coeff_2, coeff_3), 1).res
augmented = np.c_[variables, values]

print("The augmented matrix is:\n", augmented, "\n")
print("After Gaussian Elimination:\n", gauss_elim(augmented))
```

```
The augmented matrix is:
 [[ 2.62100300e-02  2.41570000e-02  2.08000000e-02  1.00000000e-02
    3.17637529e-05]
 [ 2.29773000e-02  2.43630000e-02  5.57000000e-02  2.10000000e-01
   -6.30740802e-02]
 [-1.26984820e-01 -8.34880000e-02 -5.72000000e-02 -3.00000000e-02
    3.18901788e-01]
 [-5.87888940e-01 -3.42448000e-01 -2.38500000e-01 -4.40000000e-01
    1.77659810e+00]]

After Gaussian Elimination:
 [[  1.            0.            0.            0.          -13.24875075]
 [  0.           1.            0.            0.           23.73325125]
 [  0.           0.            1.            0.          -11.57181872]
 [  0.           0.            0.            1.            1.46515973]]
```

Part (g)

Preamble

This function will take care of animating the segway. DO NOT EDIT!

In [97]:
```
# frames per second in simulation
fps = 20
# length of the segway arm/stick
stick_length = 1.

def animate_segway(t, states, controls, length):
    #Animates the segway

    # Set up the figure, the axis, and the plot elements we want to an
    fig = plt.figure()

    # some config
```

```
    segway_width = 0.4
    segway_height = 0.2

    # x coordinate of the segway stick
    segwayStick_x = length * np.add(states[:, 0],sin(states[:, 2]))
    segwayStick_y = length * cos(states[:, 2])

    # set the limits
    xmin = min(around(states[:, 0].min() - segway_width / 2.0, 1), arou
    xmax = max(around(states[:, 0].max() + segway_height / 2.0, 1), arc

    # create the axes
    ax = plt.axes(xlim=(xmin-.2, xmax+.2), ylim=(-length-.1, length+.1

    # display the current time
    time_text = ax.text(0.05, 0.9, 'time', transform=ax.transAxes)

    # display the current control
    control_text = ax.text(0.05, 0.8, 'control', transform=ax.transAxes

    # create rectangle for the segway
    rect = Rectangle([states[0, 0] - segway_width / 2.0, -segway_height
        segway_width, segway_height, fill=True, color='gold', ec='blue
    ax.add_patch(rect)

    # blank line for the stick with o for the ends
    stick_line, = ax.plot([], [], lw=2, marker='o', markersize=6, colo

    # vector for the control (force)
    force_vec = ax.quiver([],[],[],[],angles='xy',scale_units='xy',sca

    # initialization function: plot the background of each frame
    def init():
        time_text.set_text('')
        control_text.set_text('')
        rect.set_xy((0.0, 0.0))
        stick_line.set_data([], [])
        return time_text, rect, stick_line, control_text

    # animation function: update the objects
    def animate(i):
        time_text.set_text('time = {:2.2f}'.format(t[i]))
        control_text.set_text('force = {:2.3f}'.format(controls[i]))
        rect.set_xy((states[i, 0] - segway_width / 2.0, -segway_height
        stick_line.set_data([states[i, 0], segwayStick_x[i]], [0, segwa
        return time_text, rect, stick_line, control_text

    # call the animator function
    anim = animation.FuncAnimation(fig, animate, frames=len(t), init_fu
            interval=1000/fps, blit=False, repeat=False)
    return anim
    #plt.show()
```

Plug in your controller here

In [98]:
```
# If you want to try zero control
# controls = np.zeros((4))

controls = [-13.249, 23.733, -11.572, 1.465]
print(controls)
```

```
[-13.249, 23.733, -11.572, 1.465]
```

Simulation

DO NOT EDIT!

In [99]:
```
# This will add an extra couple of seconds to the simulation after the
# the effect of this is just to show how the system will continue afte
controls = np.append(controls,[0, 0])

# number of steps in the simulation
nr_steps = controls.shape[0]

# We now compute finer dynamics and control vectors for smoother visua
Afine = sp.linalg.fractional_matrix_power(A,(1/fps))
Asum = np.eye(nr_states)
for i in range(1, fps):
    Asum = Asum + np.linalg.matrix_power(Afine,i)

bfine = np.linalg.inv(Asum).dot(b)

# We also expand the controls in the "intermediate steps" (only for vi
controls_final = np.outer(controls, np.ones(fps)).flatten()
controls_final = np.append(controls_final, [0])

# We compute all the states starting from x0 and using the controls
states = np.empty([fps*(nr_steps)+1, nr_states])
states[0,:] = state0;
for stepId in range(1,fps*(nr_steps)+1):
    states[stepId, :] = np.dot(Afine,states[stepId-1, :]) + controls_f

# Now create the time vector for simulation
t = np.linspace(1/fps,nr_steps,fps*(nr_steps),endpoint=True)
t = np.append([0], t)
```

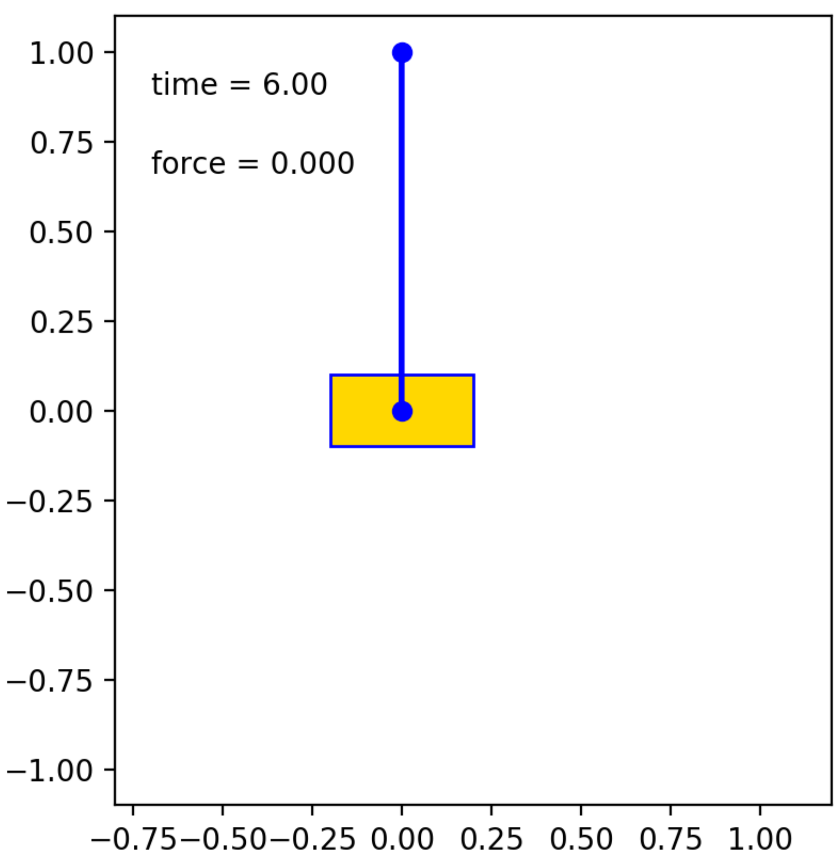
Visualization

DO NOT EDIT!

In [101]:

```
%matplotlib nbagg
# %matplotlib qt
anim = animate_segway(t, states, controls_final, stick_length)
anim
```

Figure 1



Out[101]:

```
<matplotlib.animation.FuncAnimation at 0xd248ac438>
```

In [ ]:

2. (a). Since for any point $X(x_0, y_0)$, we have $\vec{x} = x_0 \vec{u_1} + y_0 \vec{u_2} = x_0 \vec{v_1} + y_0 \vec{v_2}$

Plug in the values for points A, B, C, D: $\vec{u_1} = -2\vec{v_1} + 3\vec{v_2} \Rightarrow$ So, we have $\vec{u_2} = -\vec{v_2}$

$-2\vec{u_2} = 0\vec{v_1} + 2\vec{v_2} \Rightarrow$ and so $\vec{u_1} = -2\vec{v_1} + 3\vec{v_2}$

$-\vec{u_1} = 2\vec{v_1} - 3\vec{v_2}$ with $\vec{v_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\vec{v_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

$2\vec{u_2} = 0\vec{v_1} - 2\vec{v_2}$.

Thus, $\vec{u_1} = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$ and $\vec{u_2} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$

2. (b). Plugging values into the equation: $A_{v \to u} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 3 & -1 \end{bmatrix}$

Since $\begin{bmatrix} | & | \\ \vec{v_1} & \vec{v_2} \\ | & | \end{bmatrix}$ is the identity matrix, so $A_{v \to u} = \begin{bmatrix} -2 & 0 \\ 3 & -1 \end{bmatrix}$

2. (d). Again, plugging in values and we have: $A_{w \to u} \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 3 & -1 \end{bmatrix}$

Let $A_{w \to u} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$, and we have

$$a_1 - 3a_2 = -2, \quad a_2 = 0, \quad a_3 - 3a_4 = 3, \quad a_4 = -1 \Rightarrow \begin{cases} a_1 = -2 \\ a_2 = 0 \\ a_3 = 0 \\ a_4 = -1 \end{cases}$$

Thus, $A_{w \to u} = \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix}$

3. (g). Since we need recovery, and recovery fails iff $\exists$ two valid flows, which is equivalent to $\exists$ two distinct valid flows $\vec{f_1}$ and $\vec{f_2}$ such that $M\vec{f_1} = M\vec{f_2}$, $\vec{f_1} \neq \vec{f_2}$. $\Leftrightarrow M(\vec{f_1} - \vec{f_2}) = \vec{0}$.

Then since the nullspace of B_G is the set of possible flows, so the condition above is equivalent to having $\vec{f_1}, \vec{f_2} \in \text{Nullspace}(B_G)$, $\vec{f_1} \neq \vec{f_2}$ and $M(\vec{f_1} - \vec{f_2}) = \vec{0}$. Since Nullspace$(B_G)$ is a subspace, so $\vec{0} \in \text{Null}(B_G)$, and $M \cdot \vec{0} = \vec{0}$, so the above condition is now equivalent to that we can't have any solution other than $\vec{0}$ for $M\vec{f} = \vec{0}$ where $\vec{f} \neq \vec{0}$.

Thus, the condition is that Nullspace of M and Nullspace of B_G doesn't intersect except at $\vec{0}$.

4. (f). (Basically my original submission). Again, we have $\vec{x}[4] = A^4\vec{x}[0] + A^3\vec{b}u[0] + A^2\vec{b}u[1] + A\vec{b}u[2] + \vec{b}u[3]$

with $\vec{x}[4] = 0$, using iPython to solve the system of equations via Gaussian Elimination, so:

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & -13.249 \\ 0 & 1 & 0 & 0 & 23.733 \\ 0 & 0 & 1 & 0 & -11.572 \\ 0 & 0 & 0 & 1 & 1.465 \end{array}\right]$$

Thus, $\boxed{\text{Yes}}$ I can, with $u[0] = -13.249$, $u[1] = 23.733$, $u[2] = -11.572$, $u[3] = 1.465$