Math 425 Computation Linear Algebra

HW5

Brent A. Thorne

brentathorne@gmail.com

Basis, Othogonality, Projection, Least-squares, Factorization, and SVG.

In [1]:
```
# environment setup, try to make it clear which library I'm using for what
import numpy as np  # nice arrays and other stuff
import scipy as sci # like numpy but nicer
import sympy as sym # symbollic maths
from sympy.matrices import Matrix # pretty matrices
from sympy import Eq # pretty equations
from sympy.physics.quantum.dagger import Dagger # we'll want this later...
from math import e, pi, sqrt # Mathy math math
from mpl_toolkits.mplot3d import Axes3D # we like 3d quivers for tutorials
import matplotlib.pyplot as plt # old standby for plotting like a villian
from IPython.display import display, Math, Latex # used to display formatted re
sults in the console
sym.init_printing()  # initialize pretty printing
```

1. Find an orthogonal basis for the column space of matrix $A = \begin{bmatrix} -1 & 6 & 6 \\ 3 & -8 & 3 \\ 1 & -2 & 6 \\ 1 & -4 & -3 \end{bmatrix}$.

In [2]:
```
A = Matrix([[-1,6,6],[3,-8,3],[1,-2,6],[1,-4,-3]])
print("Show columns are independent:")
A.rref() # columns are independent
```

```
Show columns are independent:
```

Out[2]:

$$\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, (0,\ 1,\ 2)\right)$$

In [3]:
```
# do it semi-manually
v1 = A.col(0)
v2 = A.col(1) - v1*A.col(1).dot(v1)/v1.dot(v1)
v3 = A.col(2) - v1*A.col(2).dot(v1)/v1.dot(v1) - v2*A.col(2).dot(v2)/v2.dot(v2)
print("Show our semi-manual result:")
v1,v2,v3
```

```
Show our semi-manual result:
```

Out[3]:

$$\left(\begin{bmatrix}-1\\3\\1\\1\end{bmatrix}, \begin{bmatrix}3\\1\\1\\-1\end{bmatrix}, \begin{bmatrix}-1\\-1\\3\\-1\end{bmatrix}\right)$$

In [4]:
```
# show sympy results
print("Show sympy result:")
sym.GramSchmidt([A.col(0),A.col(1),A.col(2)])
```

```
Show sympy result:
```

Out[4]:

$$\left[\begin{bmatrix}-1\\3\\1\\1\end{bmatrix}, \begin{bmatrix}3\\1\\1\\-1\end{bmatrix}, \begin{bmatrix}-1\\-1\\3\\-1\end{bmatrix}\right]$$

2. Find an orthonormal basis for the column space of the matix $A = \begin{bmatrix}3 & -3 & 0\\-4 & 14 & 10\\5 & -7 & -2\end{bmatrix}$.

In [5]:
```
print('Show columns are NOT independent:')
A = Matrix([[3,-3,0],[-4,14,10],[5,-7,-2]])
A, A.rref()
```

```
Show columns are NOT independent:
```

Out[5]:

$$\left(\begin{bmatrix}3 & -3 & 0\\-4 & 14 & 10\\5 & -7 & -2\end{bmatrix}, \left(\begin{bmatrix}1 & 0 & 1\\0 & 1 & 1\\0 & 0 & 0\end{bmatrix}, (0, 1)\right)\right)$$

In [6]:
```
# do it semi-manually
v1 = A.col(0)
v2 = A.col(1) - v1*A.col(1).dot(v1)/v1.dot(v1)
print('Show our semi-manual orthoginal result:')
v1,v2
```

```
Show our semi-manual orthoginal result:
```

Out[6]:

$$\left(\begin{bmatrix} 3 \\ -4 \\ 5 \end{bmatrix}, \begin{bmatrix} 3 \\ 6 \\ 3 \end{bmatrix} \right)$$

In [7]:
```
print('Show the orthonormal result:')
v1/sym.sqrt(v1.dot(v1)), v2/sym.sqrt(v2.dot(v2))
```

```
Show the orthonormal result:
```

Out[7]:

$$\left(\begin{bmatrix} \frac{3\sqrt{2}}{10} \\ -\frac{2\sqrt{2}}{5} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \begin{bmatrix} \frac{\sqrt{6}}{6} \\ \frac{\sqrt{6}}{3} \\ \frac{\sqrt{6}}{6} \end{bmatrix} \right)$$

In [8]:
```
print("Show sympy orthonormal result:")
sym.GramSchmidt([A.col(0),A.col(1)], orthonormal=True)
```

```
Show sympy orthonormal result:
```

Out[8]:

$$\left[\begin{bmatrix} \frac{3\sqrt{2}}{10} \\ -\frac{2\sqrt{2}}{5} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \begin{bmatrix} \frac{\sqrt{6}}{6} \\ \frac{\sqrt{6}}{3} \\ \frac{\sqrt{6}}{6} \end{bmatrix} \right]$$

3. Let $u_1, \ldots, u_p$ be an orthogonal basis for the subspace W of R^n, and let $T : \mathrm{R}^n \rightarrow \mathrm{R}^n$ be defined by $T(x) = proj_W x$.

Show that T is a linear transformation.

If $\{u_1, \ldots, u_p\}$ is and orthogonal basis for the subspace W of R^n, then each x in W can be form as a linear combination of each basis vector.

$x = c_1u_1 + c_2u_2 + \ldots + c_pu_p$

From the above we see that the weights of the constants are given by,

$c_j = \frac{x \circ u_j}{u_j \circ u_j}$ $(j = 1, \ldots, p)$, which by construction is $proj_W x_j$.

defining another linear combination y as,

$y = b_1v_1 + b_2v_2 + \ldots + b_pv_p$

...we can clearly see $T(cx + y) = cT(x) + T(y)$, thus proving T is a linear tranformation.

4. Let $A = \begin{bmatrix} 1 & 2 \\ -1 & 4 \\ 1 & 2 \end{bmatrix}$ and $b = \begin{bmatrix} 3 \\ -1 \\ 5 \end{bmatrix}$.

Find (a) the orthogonal projection of b onto $ColA$ and (b) a least-squares solution of $Ax = b$.

In [9]:
```
A = Matrix([[1,2],[-1,4],[1,2]])
b = Matrix([3,-1,5])
A,b
a1, a2 = A.columnspace()
a1 = Matrix(a1)
a2 = Matrix(a2)
b_hat = a1*b.dot(a1)/a1.dot(a1) + a2*b.dot(a2)/a2.dot(a2)
print('(a) Show b projected on ColA:')
b_hat
```

```
(a) Show b projected on ColA:
```

Out[9]:

$$\begin{bmatrix} 4 \\ -1 \\ 4 \end{bmatrix}$$

In [10]:
```
print('(b) Recall A.T*Ax = A.Tb (least squares):')
A.T*A, A.T*b, (A.T*A).row_join(A.T*b), (A.T*A).row_join(A.T*b).rref()
```

```
(b) Recall A.T*Ax = A.Tb (least squares):
```

Out[10]:

$$\left(\begin{bmatrix} 3 & 0 \\ 0 & 24 \end{bmatrix}, \begin{bmatrix} 9 \\ 12 \end{bmatrix}, \begin{bmatrix} 3 & 0 & 9 \\ 0 & 24 & 12 \end{bmatrix}, \left(\begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & \frac{1}{2} \end{bmatrix}, (0, 1)\right)\right)$$

In [11]:
```python
print('(b) continued...')
print('Show x as the least-squares solution: (also show that Ax=b, as seen above)')
x = (A.T*A).row_join(A.T*b).rref(pivots=False).col(-1) # yes, ugly but we're just messing about
x, A*x
```

```
(b) continued...
Show x as the least-squares solution: (also show that Ax=b, as seen above)
```

Out[11]:

$$\left(\begin{bmatrix}3\\\frac{1}{2}\end{bmatrix}, \begin{bmatrix}4\\-1\\4\end{bmatrix}\right)$$

5. Let $A = \begin{bmatrix}2 & 1\\-2 & 0\\2 & 3\end{bmatrix}$ and $b = \begin{bmatrix}-5\\8\\1\end{bmatrix}$. Find the least-square solution of $Ax = b$.

In [12]:
```python
A = Matrix([[2,1],[-2,0],[2,3]])
b = Matrix([-5,8,1])
print('Recall A.T*Ax = A.Tb (least squares)...')
A.T*A, A.T*b, (A.T*A).row_join(A.T*b), (A.T*A).row_join(A.T*b).rref()
A,b
print('Show least squares solution:')
(A.T*A).row_join(A.T*b).rref(pivots=False).col(-1) # we're mirroring how it's done on paper
```

```
Recall A.T*Ax = A.Tb (least squares)...
Show least squares solution:
```

Out[12]:

$$\begin{bmatrix}-4\\3\end{bmatrix}$$

In [13]:
```python
print('Show a simpler method since we are using a computer:')
(A.T*A).inv()*A.T*b
```

```
Show a simpler method since we are using a computer:
```

Out[13]:

$$\begin{bmatrix}-4\\3\end{bmatrix}$$

6. Let $A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ **and** $b = \begin{bmatrix} 7 \\ 2 \\ 3 \\ 6 \\ 5 \\ 4 \end{bmatrix}$.

Describe all least-squares solutions of the equation $Ax = b$.

In [14]:
```
A = Matrix([[1,1,1,1,1,1],[1,1,1,0,0,0],[0,0,0,1,1,1]]).T
b = Matrix([7,2,3,6,5,4])
print('Show columns of A are dependent: (thus all solutions are approximate and have rank =2 so 4 free varibles)')
A, A.rref(), len(A.T.nullspace()) # think about a better way to 'describe' the form of these solutions
```

```
Show columns of A are dependent: (thus all solutions are approximate and have r
ank =2 so 4 free varibles)
```

Out[14]:

$$\left(\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \left(\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, (0,\ 1) \right), 4 \right)$$

In [15]:
```
print('Show our approximate solution:')
Q = Matrix([A.columnspace()])  # think about this, a bit more
x_hat = (Q.T*Q).inv()*Q.T*b
Q* x_hat
display(Latex('$\\hat{x}=$' + f'${sym.latex(x_hat)}$'))
```

```
Show our approximate solution:
```

$$\hat{x} = \begin{bmatrix} 5 \\ -1 \end{bmatrix}$$

7. Let $A = \begin{bmatrix} 1 & -1 \\ 1 & 4 \\ 1 & -1 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 0 & 5 \end{bmatrix}$**.**

Use the QR **factorization to find the least-squares solution of** $Ax = b$**.**

In [16]:

```
A = Matrix([[1,-1],[1,4],[1,-1],[1,4]])
Q = sym.Rational(1,2) * Matrix([[1,-1],[1,1],[1,-1],[1,1]])
R = Matrix([[2,3],[0,5]])
print('Show some truth:')
A,Q,R, Q.T*A
```

```
Show some truth:
```

Out[16]:

$$\left(\begin{bmatrix} 1 & -1 \\ 1 & 4 \\ 1 & -1 \\ 1 & 4 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \begin{bmatrix} 2 & 3 \\ 0 & 5 \end{bmatrix}, \begin{bmatrix} 2 & 3 \\ 0 & 5 \end{bmatrix} \right)$$

```
In [17]: print('Recall: R*x=Q.T*b')
b = Matrix([sym.symbols('b1 b2 b3 b4')]).T
R.inv()*Q.T, b, R.inv()*Q.T *b
display(Latex('Thus, $\hat{x}= R^{-1}Q^T b=$' + \
              f'${sym.latex(R.inv())} * \
              {sym.latex(Q.T)} * {sym.latex(b)} = \
              {sym.latex(R.inv()*Q.T *b)}$'))
```

```
Recall: R*x=Q.T*b
```

Thus, $\hat{x} = R^{-1}Q^T b = \begin{bmatrix} \frac{1}{2} & -\frac{3}{10} \\ 0 & \frac{1}{5} \end{bmatrix} * \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} * \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} \frac{2b_1}{5} + \frac{b_2}{10} + \frac{2b_3}{5} + \frac{b_4}{10} \\ -\frac{b_1}{10} + \frac{b_2}{10} - \frac{b_3}{10} + \frac{b_4}{10} \end{bmatrix}$

8. A healthy child's systolic blood pressure p (in millimeter of mercury) and weight w (in pounds) are approximately related by the equation

$$\beta_0 + \beta_1 \ln w = p$$

Use the following experimental data to estimate the systolic blood pressure of a healthy child weighting 100 pounds.

w	ln w	p
44	3.78	91
61	4.11	98
81	4.41	103
113	4.73	110
131	4.88	112

In [18]:
```
p = Matrix([91,98,103,110,112])
w = [44,61,81,113,131]
ln_w = [3.78,4.11,4.41,4.73,4.88]

# cast into familar form
y = p
X = Matrix([sym.ones(1,5), w, ln_w]).T
y,X
display(Latex('Recall: $X\\beta=y$'))
display(Latex('Thus, $X^TX\\beta=X^Ty$'))

beta = ((X.T*X).inv() * X.T*y)
display(Latex('Or, $\\beta = (X^TX)^{-1}X^Ty$' + \
              f'$= {sym.latex(beta.n(3))}$'))
```

Recall: $X\beta = y$

Thus, $X^TX\beta = X^Ty$

Or, $\beta = (X^TX)^{-1}X^Ty = \begin{bmatrix} 13.0 \\ -0.0211 \\ 20.9 \end{bmatrix}$

In [19]:
```
print('Test our results on a known results: (looks good)')
(X*beta).n(3)
```

```
Test our results on a known results: (looks good)
```

Out[19]:

$$\begin{bmatrix} 91.1 \\ 97.6 \\ 103.0 \\ 110.0 \\ 112.0 \end{bmatrix}$$

In [20]:
```
print('Show estimated systolic blood pressure of a 100lb child: (with design ma
trix, x)')
x = Matrix([1,100, sym.ln(100)]).T
x, (x*beta).n(3)
```

```
Show estimated systolic blood pressure of a 100lb child: (with design matrix,
x)
```

Out[20]: $\left(\begin{bmatrix} 1 & 100 & \log(100) \end{bmatrix}, \begin{bmatrix} 107.0 \end{bmatrix}\right)$

9. To measure the takeoff performance of an airplane, the horizontal position of the plane was measured every second, from $t = 0$ to $t = 12$. The positions (in feet) were: 0, 8.8, 29.9, 62.0, 104.7, 159.1, 222.0, 294.5, 380.4, 471.1, 571.7, 686.8, 809.2.

(a) Find the least-squares cubic curve $y = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3$ for these data.

(b) Use the result of (a) to estimate the velocity of the plane when $t = 4.5$ seconds.

In [21]:
```
t = range(12+1)
p = [0, 8.8, 29.9, 62.0, 104.7, 159.1, 222.0, 294.5, 380.4, 471.1, 571.7, 686.
8, 809.2]

# design our matrix
x0 = sym.ones(1,len(t))
x1 = [e for e in t]
x2 = [e**2 for e in x1]
x3 = [e**3 for e in x1]
X = Matrix([x0,x1,x2,x3]).T

# cast vresults to a familar form
y = Matrix(p)

display(Latex('(a) Recall: $X\\beta=y$'))
display(Latex('Thus, $X^TX\\beta=X^Ty$'))

beta = ((X.T*X).inv() * X.T*y)
display(Latex('Or, $\\beta = (X^TX)^{-1}X^Ty$' + \
              f'$= {sym.latex(beta.n(3))}$'))
```

(a) Recall: $X\beta = y$

Thus, $X^TX\beta = X^Ty$

Or, $\beta = (X^TX)^{-1}X^Ty = \begin{bmatrix} -0.856 \\ 4.7 \\ 5.56 \\ -0.0274 \end{bmatrix}$

In [22]:
```
display(Latex(f'(b) Test our results on a known results: \
$y={sym.latex(y)} ≈ {sym.latex((X*beta).n(3))}$ \
* checks out!'))

display(Latex(f'Now find velocity at $t=4.5$:'))
epsilon = 0.001 # estimation interval
t0 = 4.5
t1 = t0 + epsilon
x0 = Matrix([1, t0, t0**2, t0**3]).T
x1 = Matrix([1, t1, t1**2, t1**3]).T

d0 = (x0*beta)[0]
d1 = (x1*beta)[0]
v_avg = d0/t0
v_inst = (d1-d0)/(t1-t0) # velocity for our interval

display(Latex(f' Our position at $t={t0}$ is ${sym.latex(round(d0,1))}$ft, \
thus our averge velocity is ${sym.latex(round(v_avg,1))}$' + '$\\frac{ft}{s}$ \
and our instantanious velocity is ' + f'${sym.latex(round(v_inst,1))}$' + '$\\f
rac{ft}{s}$. *'))
display(Latex('* Displayed results rounded to one decimal point.'))
```

(b) Test our results on a known results: $y = \begin{bmatrix} 0 \\ 8.8 \\ 29.9 \\ 62.0 \\ 104.7 \\ 159.1 \\ 222.0 \\ 294.5 \\ 380.4 \\ 471.1 \\ 571.7 \\ 686.8 \\ 809.2 \end{bmatrix} \approx \begin{bmatrix} -0.856 \\ 9.38 \\ 30.6 \\ 62.5 \\ 105.0 \\ 158.0 \\ 221.0 \\ 295.0 \\ 378.0 \\ 472.0 \\ 574.0 \\ 687.0 \\ 808.0 \end{bmatrix}$ * checks out!

Now find velocity at $t = 4.5$:

Our position at $t = 4.5$ is 130.3ft, thus our averge velocity is $29.0\frac{ft}{s}$ and our instantanious velocity is $53.0\frac{ft}{s}$. *

* Displayed results rounded to one decimal point.

10. Find the singular values of the matrix $\begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}$.

In [23]:
```
A = Matrix([[-5,0],[0,0]])
A, A.rank(), A.T*A, (A.T*A).eigenvects()
#help(Matrix.eigenvects)
```

Out[23]:

$$\left(\begin{bmatrix}-5 & 0\\0 & 0\end{bmatrix}, 1, \begin{bmatrix}25 & 0\\0 & 0\end{bmatrix}, \left[\left(0, 1, \left[\begin{bmatrix}0\\1\end{bmatrix}\right]\right), \left(25, 1, \left[\begin{bmatrix}1\\0\end{bmatrix}\right]\right)\right]\right)$$

In [24]:
```
print('Show semi-manual process to find SVD:')
V = Matrix([[1,0],[0,1]]) # order our eigenvects

m,n = A.shape
sigma = sym.zeros(m,n) # our matrix for sigma is the same shape as A
sigma_1 = sym.sqrt(25) # made our sigmi
sigma_2 = 0
sigma[0] = sigma_1

u1 = 1/sigma_1*A*V.col(0) # Av_k
u2 = sym.zeros(m,1) # simga_2 is 0 so just cook up a zero vector
U = Matrix([u1.T,u2.T]).T # U is our non-zero Av_k vectors

A, U*sigma*V.T  # validate our result
display(Latex(f'$A=U\\Sigma V^T= \
{sym.latex(U)}{sym.latex(sigma)}{sym.latex(V.T)}=\
{sym.latex(U*sigma*V.T)}$ * Where the non-zero $\Sigma$ are the singular value
s.'))
display(Latex(f'$A={sym.latex(A)}$, *checks out!'))  # think about this, the si
gma are our singular values
```

```
Show semi-manual process to find SVD:
```

$A = U\Sigma V^T = \begin{bmatrix}-1 & 0\\0 & 0\end{bmatrix}\begin{bmatrix}5 & 0\\0 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \begin{bmatrix}-5 & 0\\0 & 0\end{bmatrix}$ * Where the non-zero Σ are the singular values.

$A = \begin{bmatrix}-5 & 0\\0 & 0\end{bmatrix}$, *checks out!

11. Suppose the factorization below is an SVD of a matrix A, with the entries in U and V rounded to two decimal places.

$$\mathbf{A = \begin{bmatrix}-0.86 & -0.11 & -0.50\\0.31 & 0.68 & -0.67\\0.41 & -0.73 & -0.55\end{bmatrix}\begin{bmatrix}12.48 & 0 & 0 & 0\\0 & 6.34 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}0.66 & -0.03 & -0.35 & 0.66\\-0.13 & -0.90 & -0.39 & -0.13\\0.65 & 0.08 & -0.16 & -0.73\\-0.34 & 0.42 & -0.84 & -0.08\end{bmatrix}}$$

(a) What is the rank of A?

(b) Use this decomposition of A, with no calculations, to write a basis for Col A and a basis for Nul A.

(a) The rank is 2 based on the Diagonal matrix in Σ.

(b) The basis of A is the first two columns of U, $\left\{ \begin{bmatrix} -0.86 & -0.11 \\ 0.31 & 0.68 \\ 0.41 & -0.73 \end{bmatrix} \right\}$.

The basis for the Nul A is the last two rows of the V^T, $\begin{bmatrix} 0.65 & 0.08 & -0.16 & -0.73 \\ -0.34 & 0.42 & -0.84 & -0.08 \end{bmatrix}$ or rather more clearly stated, the last two columns of V,

$$\left\{ \begin{bmatrix} 0.65 \\ 0.08 \\ -0.16 \\ -0.73 \end{bmatrix} \begin{bmatrix} -0.34 \\ 0.42 \\ -0.84 \\ -0.08 \end{bmatrix} \right\}.$$

In [25]:
```python
print('Show the calculation to valid our results: (we are expecting floating po
int error)')
U = Matrix([[-0.86, -0.11, -0.50],[0.31,0.68,-0.67],[0.41,-0.73,-0.55]])
sigma = Matrix([[12.48,0,0,0],[0,6.34,0,0],[0,0,0,0]])
V = Matrix([[0.66,-0.03,-0.35,0.66],[-0.13,-0.90,-0.39,-0.13],[0.65,0.08,-0.1
6,-0.73],[-0.34,0.42,-0.84,-0.08]]).T
A = U*sigma*V.T

display(Latex(f'sympy.Matrix.rank() correctly displays the rank as {sym.latex
(A.rank())}.'))
display(Latex(f'$A^TA.eigenval()$ shows two very small eigenvalues, \
${sym.latex((A.T*A).eigenvals())}$, which provides further evidence that the ra
nk is indeed $(4-2)=2$.'))

v1 = A.col(0)
v2 = A.col(1) - v1 * A.col(1).dot(v1)/v1.dot(v1)
v1 = v1/v1.norm()
v2 = v2/v2.norm()
v1,v2

display(Latex('Show $UU^T, VV^T$: *Here we are demostrating U and V are orthono
rmal basises by rounding.'))
(U*U.T).applyfunc(lambda x: round(x,1)), (V*V.T).applyfunc(lambda x: round(x,
1))# A.col(0)/A.col(0).norm(), A.col(1)/A.col().norm()
```

```
Show the calculation to valid our results: (we are expecting floating point err
or)
```

sympy.Matrix.rank() correctly displays the rank as 2.

$A^TA.eigenval()$ shows two very small eigenvalues,

$\left\{ -2.13039473560594 \cdot 10^{-15} : 1,\ -2.00925079369198 \cdot 10^{-63} : 1,\ 40.3241537771784 : 1,\ 155.493056575486 : 1 \right\}$

, which provides further evidence that the rank is indeed $(4-2)=2$.

Show UU^T, VV^T: *Here we are demostrating U and V are orthonormal basises by rounding.

Out[25]:

$$\left(\begin{bmatrix} 1.0 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}, \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix} \right)$$

12. Suppose A is square and invertible. Find the singular value decomposition of A^{-1}.

Recall: $A = U\Sigma\ V^T$

also recall: $A^{-1} = V\Sigma^{-1}U^T$, where $\Sigma^{-1} = diag(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, ..., \frac{1}{\sigma_n})$

Working backwards see can see this is correct:

$A^{-1}A = (V\Sigma^{-1}U^T)(U\Sigma V^T)$

$= V\Sigma^{-1}(U^TU)\Sigma V^T)$

$= V(\Sigma^{-1}\Sigma)V^T)$

$= VV^T$

$= I$

To the Mathematician it ought to be obvious how this might be made into a proof, being a pedantic lot maybe we should just do this. Here we go...

Proof:

$A^{-1} = (U\Sigma V^T)^{-1}$

$= (V^T)^{-1}\Sigma^{-1}U^{-1}$, ($U$ and V are orthonormal, thus $(V^T)^{-1} = V$ and $U^{-1} = U^T$)

$= V\Sigma^{-1}U^T$, where $\Sigma^{-1} = diag(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, ..., \frac{1}{\sigma_n})$.

∎

'Ah! Never to escape from Being and Number!'

-Charles Baudelaire, The Void

- see also: Moore-Penrose inverse (pseudoinverse) where, $A^{\dagger} = V_r\Sigma^{-1}U_r^T$
- see also Lay, Ex 7.4.7

13. Show that if A is square, then $|detA|$ is the product of the singular values of A.

Recall: $A = U\Sigma V^T$

Also recall for a square matrix:

$\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$, where $\sigma_i = \sqrt{\lambda_i}$.

These λ_i are our eigenvalues given by our characteristic polynomial of A.

We also know that U and V^{-1} are orthonormal thus are rotations that will not scale Σ. Further we can note the determinate of an orthonormal basis is 1. This property means U and V are unitary.

Thus by construction,

$|detA| = \Sigma \circ \Sigma^T$

$= diag(\sigma_1{}^2, \sigma_2{}^2, \ldots, \sigma_n{}^2)$

$= diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$

∎

14. Find the minimal length least-squares solution of the equation $A\mathbf{x} = \mathbf{b}$, where

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 8 \\ 2 \end{bmatrix}.$$

In [26]:
```
display(Latex('Recall: $A^TAx=A^Tb$'))
display(Latex("Thus, $\\hat{x}=(A^TA)^{-1}A^Tb$, We've been setup! A.T*A is not invertible."))
A = Matrix([[1,1,1,1],[1,1,0,0],[0,0,1,1]]).T # Transposed to make it easiler to type
b = Matrix([1,3,8,2])
#x_hat = (A.T*A).inv() * A.T*b  # we've been setup, A.T*A is not invertible
x_hat = ((A.T*A).row_join(A.T*b)).rref(pivots=False)
A, x_hat, A*x_hat.col(-1)  # FIXME!!! Think about this result!
display(Latex('$\\hat{x}=$'+f'${sym.latex(x_hat.col(-2))},{sym.latex(x_hat.col(-1))}$'))
display(Latex('$\\hat{b}=$'+f'${sym.latex(A*x_hat.col(-2))},{sym.latex(A*x_hat.col(-1))}$'))
print('Think about this result!')
z1 = b-A*x_hat.col(-2)
z2 = b-A*x_hat.col(-1)
display(Latex('$\\hat{x}=$'+f'${sym.latex(x_hat.col(-1))}$'+' minimized the length of $b-\\hat{b}$.'))
```

Recall: $A^TAx = A^Tb$

Thus, $\hat{x} = (A^TA)^{-1}A^Tb$, We've been setup! A.T*A is not invertible.

$$\hat{x} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 5 \\ -3 \\ 0 \end{bmatrix}$$

$$\hat{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \\ 5 \\ 5 \end{bmatrix}$$

```
Think about this result!
```

$\hat{x} = \begin{bmatrix} 5 \\ -3 \\ 0 \end{bmatrix}$ minimized the length of $b - \hat{b}$.

In [27]:
```
print('More thoughts...') # A.T*A is our correlation matrix
A.rank(), A.nullspace(), (A.T*A).eigenvects(), A*A.T
```

```
More thoughts...
```

Out[27]:

$$\left(2, \left[\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}\right], \left[\left(0, 1, \left[\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}\right]\right), \left(2, 1, \left[\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}\right]\right), \left(6, 1, \left[\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}\right]\right)\right], \begin{bmatrix} 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix}\right)$$

Appendix 1. Practice Problems

```
In [28]: A = Matrix([[1,2,3],[4,5,6],[7,8,9]])
         In = sym.eye(3)
         In[2,2]=0
         In[1,1]=0

         A,In,A*In
```

Out[28]:

$$\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 0 & 0 \end{bmatrix} \right)$$

Lay Ex6.6.2

Suppose we wish to approximate the data by an equation of the form $y_1 = \beta_0 + \beta_1 x_1 + \beta_2 x^2$.

Describe the linear model that produces a "least-squares fit" of the data by the above equation.

The coordinate of data points (x_k, y_k) must statisfy the equations of the form $y_k = \beta_0 + \beta_1 x_k + \beta_2 x_k^2 + \epsilon_k$.

```
In [29]: y = Matrix(sym.symbols('y1 y2 y_n'))
         beta = Matrix(sym.symbols('beta:4'))
         epsilon = Matrix(sym.symbols('epsilon1 epsilon2 epsilon_n'))
         x0 = sym.ones(1,3)
         x1 = sym.symbols('x1, x2, x_n')
         x2 = [e**2 for e in x1]
         x3 = [e**3 for e in x1]
         X = Matrix([x0,x1,x2,x3]).T
         y, X, beta, epsilon
         display(Latex('$y=X\\beta+\\epsilon$'))  # note use of '\\beta' to escape '\' c
         haractor
         display(Latex("Where $y$ is 'observation vector', $X$ is the 'design matrix', \
         $\\beta$ is the 'parameter vector' and $\\epsilon$ is the 'residual vector'."))
         display(Latex(f'${sym.latex(y)}={sym.latex(X)}{sym.latex(beta)}+{sym.latex(epsi
         lon)}$'))
```

$y = X\beta + \epsilon$

Where y is 'observation vector', X is the 'design matrix', β is the 'parameter vector' and ϵ is the 'residual vector'.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_n \end{bmatrix}$$