Exercises

3.2

(a)

$$Ax = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = b$$

(b)
First we set up a new system by,

$$A^T A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 4 & 10 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 11 \end{bmatrix}$$

Such that,

$$A^T Ax = \begin{bmatrix} 3 & 4 \\ 4 & 10 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 6 \\ 11 \end{bmatrix} = A^T b \; A^T Ax = \begin{bmatrix} 3 & 4 \\ 4 & 10 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 6 \\ 11 \end{bmatrix} = A^T b$$

Therefore, the normal equations are:

$$1) \quad 3x_0 + 4x_1 = 6$$
$$2) \quad 4x_0 + 10x_1 = 11$$

(c)
To solve this via Cholesky factorization, we first need to find a lower triangular matrix L such that,

$$A^T A = LL^T$$

We can do this directly by,

$$A^T A = \begin{bmatrix} 3 & 4 \\ 4 & 10 \end{bmatrix} = \begin{bmatrix} a & 0 \\ b & c \end{bmatrix} = \begin{bmatrix} a & b \\ 0 & c \end{bmatrix} = \begin{bmatrix} a^2 & ab \\ ab & b^2 + c^2 \end{bmatrix}$$

From this we obtain 3 useful equations,

$$a^2 = 3 => a = \sqrt{3}$$
$$ab = 4 => \sqrt{3}b = 4 => b = \frac{4}{\sqrt{3}}$$
$$b^2 + c^2 = 10 => (\frac{4}{\sqrt{3}})^2 + c^2 = 10 => c^2 = \frac{14}{3} => c = \frac{\sqrt{14}}{\sqrt{3}}$$

Thus,

$$L = \begin{bmatrix} \sqrt{3} & 0 \\ \frac{4}{\sqrt{3}} & \frac{\sqrt{14}}{\sqrt{3}} \end{bmatrix}$$

Next we use forward substitution to solve for z in $Lz = A^T b$,

$$\begin{bmatrix} \sqrt{3} & 0 \\ \frac{4}{\sqrt{3}} & \frac{\sqrt{14}}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 11 \end{bmatrix}$$

From this we obtain,

$$\sqrt{3}z_1 = 6 => z_1 = \frac{6}{\sqrt{3}} \sqrt{3}z_1 = 6 => z_1 = \frac{6}{\sqrt{3}}$$

$$\frac{4}{\sqrt{3}}z_1 + \frac{\sqrt{14}}{\sqrt{3}}z_2 = 11 => \frac{24}{3} + \frac{\sqrt{14}}{\sqrt{3}}z_2 = 11 => \frac{\sqrt{14}}{\sqrt{3}}z_2 = 3 => z_2 = \frac{3\sqrt{3}}{\sqrt{14}}$$

Now we use backwards substitution to solve for x in $L^T x = z$

$$\begin{bmatrix} \sqrt{3} & \frac{4}{\sqrt{3}} \\ 0 & \frac{\sqrt{14}}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \frac{6}{\sqrt{3}} \\ \frac{3\sqrt{3}}{\sqrt{14}} \end{bmatrix}$$

From this we obtain,

$$\frac{14}{\sqrt{3}}x_2 = \frac{3\sqrt{3}}{\sqrt{14}} => x_2 = \frac{9}{14}$$

$$\sqrt{3}x_1 + \frac{14}{\sqrt{3}}x_2 = \frac{6}{\sqrt{3}} => \sqrt{3}x_1 + \frac{18}{7\sqrt{3}} = \frac{6}{\sqrt{3}} => x_1 = \frac{24}{21}$$

Therefore, the solution to $A^T Ax = A^T b$ for x is,

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{9}{14} \\ \frac{24}{21} \end{bmatrix}$$

Or in equation form,

$$f(x) = \frac{9}{14} + \frac{24}{21}x$$

3.4

We would normally set up the system as,

$$Ax = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = b$$

But upon further inspection of the matrices it is clear that there is no solution to this system. To combat this, we would implement least squares to find a solution that gets us closest to the real solution. Since, the system is overdetermined, we can inspect the columns of the A matrix and note that its columns are linearly independent. This ensures that $A^T A$ is invertible and we can implement the left pseudo inverse of A to find a solution in the form $x = (A^T A)^{-1} A^T b$. Since, we can guarantee that $A^T A$ is invertible and the original system didn't have a solution, we can guarantee that this inverse is unique. Therefore the least squares method produces a unique solution to the system.

3.11

A matrix is orthogonal if and only if $MM^T = I$

By using introducing this logic to our system, we can set up a new system using our partitions as,

$$\begin{bmatrix} A & B \\ 0 & C \end{bmatrix} \begin{bmatrix} A^T & B^T \\ 0^T & C^T \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} = I$$

Now we can multiply the matrices and obtain,

$$\begin{bmatrix} A & B \\ 0 & C \end{bmatrix} \begin{bmatrix} A^T & B^T \\ 0^T & C^T \end{bmatrix} = \begin{bmatrix} AA^T + BO^T & AB^T + BC^T \\ 0A^T + C0^T & 0B^T + CC^T \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

We are then given a few equations from which we can solve and extract information,

1) $AA^T + BO^T = AA^T + 0 = AA^T = I$, since 0 is the zero matrix.
2) $0B^T + CC^T = 0 + CC^T = CC^T = I$, since 0 is the zero matrix.

Thus, the matrices A and C are both non-zero and orthogonal by the same logic used to set up the new system. Since, A and C are non-zero matrices and $AB^T + BC^T = 0$. We can determine that B must be the zero matrix exactly. Therefore, A and C must be orthogonal and B must be the zero matrix.

3.23

To solve for $A = QR$ we follow the Gram-Schmidt process, by denoting the columns of A by a_i where a_i is the i[th] column of A.

$$r_{11} = ||a_1|| = \sqrt{a_1 * a_1} = \sqrt{1 + e^2} = \sqrt{1 + 0} = 1$$

$$q_1 = \frac{a_1}{r_{11}} = \frac{\begin{bmatrix} 1 & e & 0 \end{bmatrix}^T}{1} = \begin{bmatrix} 1 \\ e \\ 0 \end{bmatrix}$$

$$r_{12} = q_1 * a_2 = \begin{bmatrix} 1 \\ e \\ 0 \end{bmatrix}^T * \begin{bmatrix} 1 \\ 0 \\ e \end{bmatrix} = 1 + 0 + 0 = 1$$

$$r_{22} = ||a_2 - r_{12}q_1|| = || \begin{bmatrix} 1 \\ 0 \\ e \end{bmatrix} - \begin{bmatrix} 1 \\ e \\ 0 \end{bmatrix} || = || \begin{bmatrix} 0 \\ -e \\ e \end{bmatrix} || = \sqrt{1 + e^2 + e^2} = e\sqrt{2}$$

$$q_2 = \frac{a_2 - r_{12} * q_1}{r_{22}} = \frac{\begin{bmatrix} 1 \\ 0 \\ e \end{bmatrix} - \begin{bmatrix} 1 \\ e \\ 0 \end{bmatrix}}{e\sqrt{2}} = \frac{\begin{bmatrix} 0 \\ -e \\ e \end{bmatrix}}{e\sqrt{2}} = \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

Thus we see that Q and R are,

$$Q = \begin{bmatrix} q_1 & q_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ e & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$R = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & e\sqrt{2} \end{bmatrix}$$

Now we compute the determinant of R and obtain,

$$det(R) = 1 * e\sqrt{2} - 1 * 0 = e\sqrt{2} > \sqrt{e_{mach}} > 0$$

Thus, the determinant of R is not zero.
Therefore, R is not singular even in floating point arithmetic.

Computer Problems

3.2

```
# initialization of matricies
A = np.array([[1, 0, 0, 0],
              [0, 1, 0, 0],
              [0, 0, 1, 0],
              [0, 0, 0, 1],
              [1, -1, 0, 0],
              [1, 0, -1, 0],
              [1, 0, 0, -1],
              [0, 1, -1, 0],
              [0, 1, 0, -1],
              [0, 0, 1, -1]])
b = np.array([2.95, 1.74, -1.45, 1.32, 1.23, 4.45, 1.61, 3.21, 0.45, -2.75]).reshape(-1,1)

# solving for both solutions using numpy's least sqaures method
s = np.linalg.lstsq(A, b, rcond=None)

# printing the results
print("The best values for the altitudes are", str(s[0][0][0])+",", str(s[0][1][0])+",", str(s[0][2][0]) + ", and", str(s[0][3][0]), "respectfully.")
print("These altitude values match extremly well with the direct measurements!")
```

```
The best values for the altitudes are 2.9599999999999995, 1.745999999999999, -1.4600000000000004, and 1.3139999999999994 respectfully.
These altitude values match extremly well with the direct measurements!
```

3.4

```
[41] # initialization of matricies
     A = np.array([[0.16, 0.10],
                   [0.17, 0.11],
                   [2.02, 1.29]])
     b = np.array([0.26, 0.28, 3.31]).reshape(-1, 1)
     b_p = np.array([0.27, 0.25, 3.33]).reshape(-1, 1)

     # solving for both solutions using numpy's least sqaures method
     s = np.linalg.lstsq(A, b, rcond=None)
     s_p = np.linalg.lstsq(A, b_p, rcond=None)
```

(a)

```
[42] print("The values for x1 and x2 using the non-preturbed RHS are", str(s[0][0][0]) + " and " + str(s[0][1][0]) + " respectfully.")
```

```
The values for x1 and x2 using the non-preturbed RHS are 0.9999999999999487 and 1.00000000000008 respectfully.
```

(b)

```
[43] print("The values for x1 and x2 using the preturbed RHS are", str(s_p[0][0][0]) + " and " + str(s_p[0][1][0]) + " respectfully.")
```

```
The values for x1 and x2 using the preturbed RHS are 7.008887308923287 and -8.395662993246317 respectfully.
```

(c)

```
[44] print("The results of both solutions are staggeringly different!")
     print("There are two possible issues with this system that could make this problem arise.\n")

     # first issue
     print("The first issue that can arise is that the b vector is nearly orthogonal to the span of A.")
     print("To verify this we will find a orthonormal basis of A and compare b to its vectors.")
     # compare RHS to basis of A via QR factorization and dot product
     ortho_basis = np.linalg.qr(A)[0]
     b1 = np.dot(ortho_basis[:,0],b)
     b2 = np.dot(ortho_basis[:,1],b)
     bp1 = np.dot(ortho_basis[:,0],b_p)
     bp2 = np.dot(ortho_basis[:,1],b_p)
     print("The dot products of the non-perterbed RHS to the basis vectors of A are: " + str(b1) + " and " + str(b2) + ".")
     print("The dot products of the perterbed RHS to the basis vectors of A are: " + str(bp1) + " and " + str(bp2) + ".")
     print("As we can see that in both cases the RHS is almost orthogonal to one of the basis vector of A.\n")

     # second issue
     print("The second issue that can arise is that the least squares problem is ill-conditioned.")
     print("To verify this we can calculate the condition value for the system and examine it.")
     # calculate cond value for A
     A_cond = np.linalg.cond(A)
     print("The condition value for this system is " + str(A_cond) + ", which is slightly high.\n")

     # conclusion
     print("I believe that a combination of both these issues is what is causing the drastic changes in solutions.")
```

```
The results of both solutions are staggeringly different!
There are two possible issues with this system that could make this problem arise.

The first issue that can arise is that the b vector is nearly orthogonal to the span of A.
To verify this we will find a orthonormal basis of A and compare b to its vectors.
The dot products of the non-perterbed RHS to the basis vectors of A are: [-3.33198037] and [-0.00260827].
The dot products of the perterbed RHS to the basis vectors of A are: [-3.35012691] and [0.02189815].
As we can see that in both cases the RHS is almost orthogonal to one of the basis vector of A.

The second issue that can arise is that the least squares problem is ill-conditioned.
To verify this we can calculate the condition value for the system and examine it.
The condition value for this system is 1097.5386765222238, which is slightly high.

I believe that a combination of both these issues is what is causing the drastic changes in solutions.
```

```
[45] ortho_basis
```

```
array([[-0.07868419,  0.83357047],
       [-0.08360195, -0.55206658],
       [-0.9933879 , -0.01956433]])
```

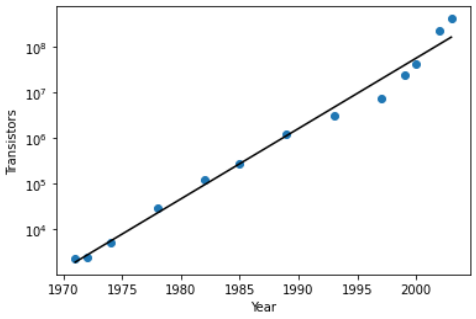
Bonus

```
# initialize matricies
start_year = 1970
A = np.array([[1, 1971 - start_year],
              [1, 1972 - start_year],
              [1, 1974 - start_year],
              [1, 1978 - start_year],
              [1, 1982 - start_year],
              [1, 1985 - start_year],
              [1, 1989 - start_year],
              [1, 1993 - start_year],
              [1, 1997 - start_year],
              [1, 1999 - start_year],
              [1, 2000 - start_year],
              [1, 2002 - start_year],
              [1, 2003 - start_year]])
b = np.log(np.array([2.25e3, 2.5e3, 5e3, 2.9e4, 1.2e5, 2.75e5, 1.18e6, 3.1e6, 7.5e6, 2.4e7, 4.2e7, 2.2e8, 4.1e8]).reshape(-1, 1))
```

(a)

```
# solve and print solution
s = np.linalg.lstsq(A, b, rcond=None)
theta_1 = s[0][0][0]
theta_2 = s[0][1][0]

# print theta values
print("Theta 1: ", math.pow(10, theta_1))
print("Theta 2: ", math.pow(10, theta_2), "\n")

# calculate line y values
y1 = math.pow(math.e, theta_1 + theta_2 * 1)
y2 = math.pow(math.e, theta_1 + theta_2 * 33)

# plotting
# scatterplot points
x = np.array([1971, 1972, 1974, 1978, 1982, 1985, 1989, 1993, 1997, 1999, 2000, 2002, 2003])
y = np.array([2.25e3, 2.5e3, 5e3, 2.9e4, 1.2e5, 2.75e5, 1.18e6, 3.1e6, 7.5e6, 2.4e7, 4.2e7, 2.2e8, 4.1e8])

# display plot
plt.scatter(x, y)
plt.plot([1971, 2003], [y1, y2], 'k-')
plt.yscale("symlog")
plt.xlabel("Year")
plt.ylabel("Transistors")
plt.show()
```

```
Theta 1:  15737763.16609627
Theta 2:  2.262767669811653
```



(b)

```
[48] # calculate transister count
transistor_count = math.pow(math.e, theta_1 + theta_2 * (2015 - 1970))

# calculate difference and report
dif = transistor_count - 4e9
print("Approximate number of transistors in 2015: ", transistor_count)
print("This is ", dif, "more transistors than were actually produced in 2015")
```

```
Approximate number of transistors in 2015:  11387036868.698685
This is  7387036868.698685 more transistors than were actually produced in 2015
```

(c)

```
[49] print("My model predicts that it will take about one year to double the number of transistors.")
print("This is about 33% faster than the prediction made by moore")
```

```
My model predicts that it will take about one year to double the number of transistors.
This is about 33% faster than the prediction made by moore
```