**Objectives:**

* Understand what a matrix is and how it corresponds to a transformation.
* Explain and calculate inverse and determinant of matrices
* Identify and explain how to find inverses computationally and what goes wrong.

**Python Code for Inverting Matrices**

import numpy as np

A = [[1, 1, 1],

[3, 2, 1],

[2, 1, 2]]

Ainv = np.linalg.inv(A)

**Python Code for Solving Linear Systems with Matrices**

import numpy as np

A = [[4, 6, 2],

[3, 4, 1],

[2, 8, 13]]

s = [9, 7, 2]

r = np.linalg.solve(A, s)

**Python Code for Checking if Matrix is Singular and Fixing it if it is not**

# GRADED FUNCTION

import numpy as np

# Our function will go through the matrix replacing each row in order turning it into echelon form.

# If at any point it fails because it can't put a 1 in the leading diagonal,

# we will return the value True, otherwise, we will return False.

# There is no need to edit this function.

def isSingular(A) :

B = np.array(A, dtype=np.float\_) # Make B as a copy of A, since we're going to alter it's values.

try:

fixRowZero(B)

fixRowOne(B)

fixRowTwo(B)

fixRowThree(B)

except MatrixIsSingular:

return True

return False

# This next line defines our error flag. For when things go wrong if the matrix is singular.

# There is no need to edit this line.

class MatrixIsSingular(Exception): pass

# For Row Zero, all we require is the first element is equal to 1.

# We'll divide the row by the value of A[0, 0].

# This will get us in trouble though if A[0, 0] equals 0, so first we'll test for that,

# and if this is true, we'll add one of the lower rows to the first one before the division.

# We'll repeat the test going down each lower row until we can do the division.

# There is no need to edit this function.

def fixRowZero(A) :

if A[0,0] == 0 :

A[0] = A[0] + A[1]

if A[0,0] == 0 :

A[0] = A[0] + A[2]

if A[0,0] == 0 :

A[0] = A[0] + A[3]

if A[0,0] == 0 :

raise MatrixIsSingular()

A[0] = A[0] / A[0,0]

return A

# First we'll set the sub-diagonal elements to zero, i.e. A[1,0].

# Next we want the diagonal element to be equal to one.

# We'll divide the row by the value of A[1, 1].

# Again, we need to test if this is zero.

# If so, we'll add a lower row and repeat setting the sub-diagonal elements to zero.

# There is no need to edit this function.

def fixRowOne(A) :

A[1] = A[1] - A[1,0] \* A[0]

if A[1,1] == 0 :

A[1] = A[1] + A[2]

A[1] = A[1] - A[1,0] \* A[0]

if A[1,1] == 0 :

A[1] = A[1] + A[3]

A[1] = A[1] - A[1,0] \* A[0]

if A[1,1] == 0 :

raise MatrixIsSingular()

A[1] = A[1] / A[1,1]

return A

# This is the first function that you should complete.

# Follow the instructions inside the function at each comment.

def fixRowTwo(A) :

# Insert code below to set the sub-diagonal elements of row two to zero (there are two of them).

A[2] = A[2] - A[2,0] \* A[0]

A[2] = A[2] - A[2,1] \* A[1]

# Next we'll test that the diagonal element is not zero.

if A[2,2] == 0 :

# Insert code below that adds a lower row to row 2.

A[2] = A[2] + A[3]

# Now repeat your code which sets the sub-diagonal elements to zero.

A[2] = A[2] - A[2,0] \* A[0]

A[2] = A[2] - A[2,1] \* A[1]

if A[2,2] == 0 :

raise MatrixIsSingular()

# Finally set the diagonal element to one by dividing the whole row by that element.

A[2] = A[2] / A[2,2]

return A

# You should also complete this function

# Follow the instructions inside the function at each comment.

def fixRowThree(A) :

# Insert code below to set the sub-diagonal elements of row three to zero.

A[3] = A[3] - A[3,0] \* A[0]

A[3] = A[3] - A[3,1] \* A[1]

A[3] = A[3] - A[3,2] \* A[2]

# Complete the if statement to test if the diagonal element is zero.

if A[3,3] == 0 :

raise MatrixIsSingular()

# Transform the row to set the diagonal element to one.

A[3] = A[3] / A[3,3]

return A

**Testing Code:**

A = np.array([

[2, 0, 0, 0],

[0, 3, 0, 0],

[0, 0, 4, 4],

[0, 0, 5, 5]

], dtype=np.float\_)

isSingular(A)

A = np.array([

[0, 7, -5, 3],

[2, 8, 0, 4],

[3, 12, 0, 5],

[1, 3, 1, 3]

], dtype=np.float\_)

fixRowZero(A)

fixRowOne(A)

fixRowTwo(A)

fixRowThree(A)











