1. Upload the notebook of week three.

03.26

solve system of linear equation

problem 1

|  |
| --- |
| # Ax = B  # T1 + m1 \* a = m1 \* g \* (np.sin(theta) - mu1 \* np.cos(theta))  # -T1 + T2 + m2 \* a = m2 \* g \* (np.sin(theta) - mu2 \* np.cos(theta))  # -T2 + T3 + m3 \* a = m3 \* g \* (np.sin(theta) - mu3 \* np.cos(theta))  # -T3 + m4 \* a = - m4 \* g  # m = [10, 4, 5, 6]  # mu = [0.25, 0.3, 0.2]  # g = 9.8 m/s^2  # Find A, B |

In [1]:

1. import numpy as np
3. mass = np.array([10, 4, 5, 6]) # kg
4. mu = np.array([0.25, 0.3, 0.2])
5. gval = 9.8 # m / s^2
6. theta = np.pi/4
8. aa = np.array([[ 1, 0, 0, mass[0]],
9. [-1, 1, 0, mass[1]],
10. [ 0,-1, 1, mass[2]],
11. [ 0, 0,-1, mass[3]]])
13. def f(x,i):
14. return (np.sin(x)-mu[i]\*np.cos(x))
16. bb = np.array([ mass[0]\*gval\*f(theta,0),
17. mass[1]\*gval\*f(theta,1),
18. mass[2]\*gval\*f(theta,2),
19. -mass[3]\*gval])
21. np.linalg.solve(aa,bb)

Out [1]: array([35.85477069, 48.81074968, 68.47054664, 1.61175777])

T1 = 35.85477069, T2 = 48.81074968, T3 = 68.47054664, a = 1.61175777

problem 2

|  |
| --- |
| # x1 \* (k1 + k2) - (x2 - x1) \* k3 - (x3 - x1) \* k5 - W1 = 0  # k3 \* (x2 - x1) - k4 \* (x3 - x2) - W2 = 0  # k4 \* (x3 - x2) + k5 \* (x3 - x1) - W3 = 0 |

In [2]:

1. k = [1,2,3,4,5]
2. W = [1,2,3]
4. aa = np.array([[k[0]+k[1]+k[2]+k[4], -k[2], -k[4]],
5. [ -k[2],k[2]+k[3], -k[3]],
6. [ -k[4], -k[3],k[3]+k[4]]])
8. bb = np.array(W)
10. np.linalg.solve(aa,bb)

Out [2]: array([2. , 2.63829787, 2.61702128])

x1 = 2, x2 = 2.63829787, x3 = 2.61702128

03.27

solving linear equation algorithm

In [1]:

1. import numpy as np
2. a = np.array([[ 4.,-2., 1.],
3. [-2., 4.,-2.],
4. [ 1.,-2., 4.]])
5. b = np.array([11.,-16.,17.])
6. x = np.linalg.solve(a,b)
7. ainv = np.linalg.inv(a)
8. x

 Out [1]: array([ 1., -2., 3.])

In [2]:

1. np.matmul(a,x) # Ax = B

Out [3]: array([ 11., -16., 17.])

In [4]:

1. np.matmul(ainv,b) # A^(-1)Ax = B

Out [4]: array([ 1., -2., 3.])

Gauss elimination

In [5]:

1. # Initializing parameters
2. ipiv = 0
3. nsize = 3
4. bb = np.zeros(nsize)
6. d = np.array(a)
7. f = np.array(b)
9. # eliminate 2nd row
10. rnum = 1
11. lam = a[ipiv+rnum,ipiv]/a[ipiv,ipiv]
12. c = a[ipiv+rnum,0:nsize] - lam \* a[ipiv,0:nsize]
13. d[rnum] = c
14. f[rnum] = f[rnum] - lam \* f[ipiv]
15. print(d,'\n',f,'\n')
17. # eliminate 3rd row
18. rnum = 2
19. lam = a[ipiv+rnum,ipiv]/a[ipiv,ipiv]
20. c = a[ipiv+rnum,0:nsize] - lam \* a[ipiv,0:nsize]
21. d[rnum] = c
22. f[rnum] = f[rnum] - lam \* f[ipiv]
23. print(d,'\n',f,'\n')
25. # eliminate 2nd column 3nd row
26. ipiv = 1
28. rnum = 2
29. lam = d[rnum,ipiv]/d[ipiv,ipiv]
30. c = d[rnum,0:nsize] - lam \* d[ipiv,0:nsize]
31. d[rnum] = c
32. f[rnum] = f[rnum] - lam \* f[ipiv]
33. print(d,'\n',f)

Out [5]:

[[ 4. -2. 1. ]

[ 0. 3. -1.5]

[ 1. -2. 4. ]]

[ 11. -10.5 17. ]

[[ 4. -2. 1. ]

[ 0. 3. -1.5 ]

[ 0. -1.5 3.75]]

[ 11. -10.5 14.25]

[[ 4. -2. 1. ]

[ 0. 3. -1.5]

[ 0. 0. 3. ]]

[ 11. -10.5 9. ]

Cholesky decomposition

Ax = b => LUx = b (A = LU, transpose(L) = U)

Ly = b => Ux = y

In [6]:

1. chole = np.linalg.cholesky(a)
2. cholet = np.transpose(chole)
3. amat = np.matmul(chole,cholet)
5. print(chole,'\n\n',cholet,'\n\n',amat)

Out [6]:

[[ 2. 0. 0. ]

[-1. 1.73205081 0. ]

[ 0.5 -0.8660254 1.73205081]]

[[ 2. -1. 0.5 ]

[ 0. 1.73205081 -0.8660254]

[ 0. 0. 1.73205081]]

[[ 4. -2. 1.]

[-2. 4. -2.]

[ 1. -2. 4.]]

# Use the Cholesky matrix to obtain y, and then x

In [7]:

1. chole = np.linalg.cholesky(a)
2. cholet = np.transpose(chole)
4. nsize = 3
6. # L\*y = b
7. ys = np.zeros(nsize)
8. ys[0] = b[0]/chole[0,0]
9. ys[1] = ( b[1] - chole[1,0]\*ys[0] ) / chole[1,1]
10. ys[2] = ( b[2] - chole[2,0]\*ys[0] - chole[2,1]\*ys[1]) / chole[2,2]
11. print(ys)
12. # U\*x = y
13. xs = np.zeros(nsize)
14. xs[2] = ys[2]/cholet[2,2]
15. xs[1] = ( ys[1] - cholet[1,2]\*xs[2] ) / chole[1,1]
16. xs[0] = ( ys[0] - cholet[0,1]\*xs[1] - cholet[0,2]\*xs[2]) / cholet[0,0]
17. print(xs)

Out [7]:

[ 5.5 -6.06217783 5.19615242]

[ 1. -2. 3.]

2. Write a code for the gauss elimination method and apply it to solve a non-singular matrix. Verify the correctness of the result comparing with linalg.solve

HW3\_2.py

1. import numpy as np
3. def triangular(A,B):
4. # Initializing parameters
5. nsize = len(A)
6. a = np.array(A)
7. b = np.array(B)
9. for ipiv in range(0,nsize-1):
10. for rnum in range(ipiv+1,nsize):
11. lam = a[rnum,ipiv]/a[ipiv,ipiv]
12. a[rnum] = a[rnum,0:nsize] - lam \* a[ipiv,0:nsize]
13. b[rnum] = b[rnum] - lam \* b[ipiv]
15. return [a,b]
17. def solve(A,B):
18. # Initializing parameters
19. nsize = len(A)
20. a = np.array(A)
21. b = np.array(B)
23. # Triangular
24. for ipiv in range(0,nsize-1):
25. for rnum in range(ipiv+1,nsize):
26. lam = a[rnum,ipiv]/a[ipiv,ipiv]
27. a[rnum] = a[rnum,0:nsize] - lam \* a[ipiv,0:nsize]
28. b[rnum] = b[rnum] - lam \* b[ipiv]
30. x = np.zeros(nsize)
31. x[nsize-1]=b[nsize-1]/a[nsize-1][nsize-1]
32. for n in range(2,nsize+1):
33. sum, i = 0, nsize - n
34. for j in range(i,nsize):
35. sum = sum + a[i][j] \* x[j]
36. x[i] = (b[i] - sum) / a[i][i]
38. return x
40. def solve0(A,B): # Use triangular function
41. # Initializing parameters
42. nsize = len(A)
43. C = triangular(A,B)
44. a = C[0]
45. b = C[1]
47. x = np.zeros(nsize)
48. x[nsize-1]=b[nsize-1]/a[nsize-1][nsize-1]
49. for n in range(2,nsize+1):
50. sum, i = 0, nsize - n
51. for j in range(i,nsize):
52. sum = sum + a[i][j] \* x[j]
53. x[i] = (b[i] - sum) / a[i][i]
55. return x

$ python

>>> from HW3\_2 import \*

>>> import numpy as np

>>> a = np.array([[4,-2,1],[-2,4,-2],[1,-2,4]])

>>> b = np.array([11,-16,17])

>>> solve(a,b)

>>> array([ 1., -2., 3.])

>>> np.linalg.solve(a,b)

>>> array([ 1., -2., 3.])

error is too small because a and b is integer

testing random 4\*4 matrix

test.py

1. import numpy as np
2. from HW3\_2 import \*
4. a, b = [], []
6. A = np.array([[0.,0.,0.,0.],[0.,0.,0.,0.],[0.,0.,0.,0.],[0.,0.,0.,0.]])
8. # Make 10 random array b
9. for i in range(0,10):
10. b.append(np.random.random(4))
12. for i in range(0,10):
13. a\_sample = np.random.random(16)
15. for j in range(0,4):
16. A[j,0:4] = a\_sample[j\*4:(j+1)\*4]
18. a.append(A)
20. num = 0
21. results = np.zeros(4)
22. for i in range(0,10):
23. x = np.linalg.solve(a[i],b[i])
24. solve = solve0(a[i],b[i])
25. results += x - solve
26. num =+ 1
28. print(results/num)

$ python test.py

[-2.91433544e-15 -1.72084569e-15 1.04916076e-14 -5.05151476e-15]

error is very small.

3. Write the backward and forward substitution codes to obtain y and then x, for the solutions of Ax=b, where A=LU, such that Ux=y, and Ly=LUx=b. Use linalg.cholesky to generate the L and U matrices, and verify against the solutions in problem 2.

In [8]:

1. def chole(a,b):
2. L = np.linalg.cholesky(a)
3. U = np.transpose(L)
4. nsize = len(a)
6. y = np.linalg.solve(L,b)
7. x = np.linalg.solve(U,y)
9. return x
11. print(solve(a,b) - np.linalg.solve(a,b))
12. print(chole(a,b) - np.linalg.solve(a,b))

Out [8]:

[0. 0. 0.]

[0.0000000e+00 0.0000000e+00 4.4408921e-16]

Cholesky algorithm’s error is higher than gauss elimination

error ware maked np.linalg.cholesky() function.

I coudn’t make random 4\*4 positive definite matrix, so coudn’t get general error