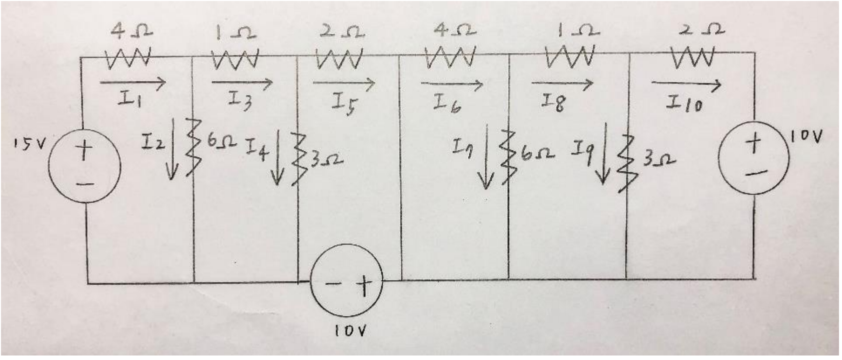
Linear Algebra Assignment 2 Report

106070038 杜葳葳

(a) your circuit



I1 = I2 + I3	I6 = I7 + I8
I3 = I4 + I5	I8 = I9 + I10
4I1 + 6I2 = 15	4I6 + 6I7 = 10
6I2 — I3 — 3I4 = 0	6I7 — I8 — 3I9 = 0
3I4 — 2I5 = 10	3I9 — 2I5 =10

(b) your matrix A and b

$$A = \begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & -1 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 4 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 & -1 & -3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & -2 \end{bmatrix}$$

B = [[0.0],[0.0],[15.0],[0.0],[10.0],[0.0],[0.0],[0.0],[0.0],[10.0]]

(c) A^{-1}=

```
[[ 0.29  0.16  0.18 -0.13 -0.08 -0.    0.   -0.    0.   -0.  ]
 [-0.19 -0.1   0.05  0.09  0.05  0.   -0.    0.   -0.    0.  ]
 [-0.52  0.26  0.13 -0.22 -0.13 -0.    0.   -0.    0.   -0.  ]
 [-0.21 -0.3   0.05 -0.09  0.15  0.   -0.    0.   -0.    0.  ]
 [-0.31 -0.44  0.08 -0.13 -0.28 -0.    0.   -0.    0.   -0.  ]
 [-0.    0.   -0.    0.   -0.    0.29  0.16  0.18 -0.13 -0.08]
 [ 0.   -0.    0.   -0.    0.   -0.19 -0.1   0.05  0.09  0.05]
 [-0.    0.   -0.    0.   -0.   -0.52  0.26  0.13 -0.22 -0.13]
 [ 0.   -0.    0.   -0.    0.   -0.21 -0.3   0.05 -0.09  0.15]
 [-0.    0.   -0.    0.   -0.   -0.31 -0.44  0.08 -0.13 -0.28]]
```

```
X= [[ 1.89]
    [ 1.24]
    [ 0.65]
    [ 2.26]
    [-1.61]
    [-0.78]
    [ 0.52]
    [-1.3 ]
    [ 1.48]
    [-2.78]]
```

(d) execution results for the third problem.(TA judge)

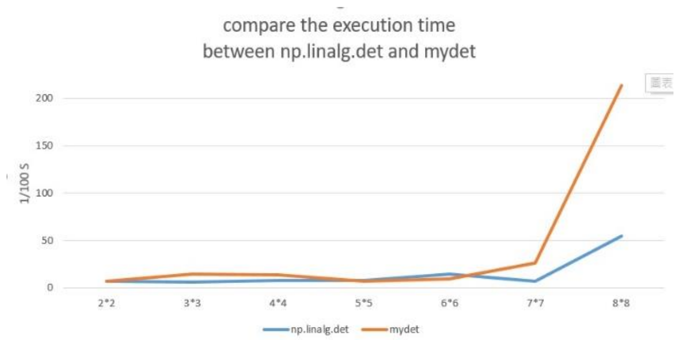
```
(2 x 2)
(mydetA): Success
(mysolve_cramer): Success
[[ 1.1 ]
 [-0.68]]
(mysolve_adj): Success
(3 x 3)
(mydetA): Success
(mysolve_cramer): Success
[[-0.96]
 [-1.97]
 [ 4.21]]
(mysolve_adj): Success
(4 x 4)
(mydetA): Success
(mysolve_cramer): Success
[[ 10.83]
 [ 20.97]
 [  4.9 ]
 [-18.79]]
(mysolve_adj): Success
(5 x 5)
(mydetA): Success
(mysolve_cramer): Success
[[ 2.28]
 [ 0.55]
 [-2.2 ]
 [ 0.43]
 [ 0.4 ]]
(mysolve_adj): Success
(6 x 6)
(mydetA): Success
(mysolve_cramer): Success
[[ 1.66]
 [-2.03]
 [-0.47]
 [ 1.27]
 [ 0.3 ]
 [ 0.29]]
(mysolve_adj): Success
(7 x 7)
(mydetA): Success
(mysolve_cramer): Success
[[-0.07]
 [-0.17]
 [ 0.28]
 [ 0.57]
 [-0.45]
 [ 0.78]
 [ 0.35]]
(mysolve_adj): Success

(Judge Test) Total Score: 18/18
```

(e) 比較 np.linalg.det 和 mydet 的執行時間

從2*2的矩陣到10*10的矩陣，可發現使用 np.linalg.det 執行時間大多比 mydet 快，且矩陣越大執行時間差越大。

```
(2 x 2)
0.2678103598726722
Using np.linalg.det,time to solve det is  0:00:00.007016  seconds.
0.2678103598726722
Using mydet,time to solve det is  0:00:00.007137  seconds.
(3 x 3)
0.027352149988940814
Using np.linalg.det,time to solve det is  0:00:00.005980  seconds.
0.027352149988940828
Using mydet,time to solve det is  0:00:00.014504  seconds.
(4 x 4)
0.030217798576239057
Using np.linalg.det,time to solve det is  0:00:00.007469  seconds.
0.03021779857623906
Using mydet,time to solve det is  0:00:00.013463  seconds.
(5 x 5)
-0.02553496564181468
Using np.linalg.det,time to solve det is  0:00:00.007980  seconds.
-0.025534965641814653
Using mydet,time to solve det is  0:00:00.007018  seconds.
(6 x 6)
0.002989638593517613
Using np.linalg.det,time to solve det is  0:00:00.014028  seconds.
0.002989638593517622
Using mydet,time to solve det is  0:00:00.009606  seconds.
(7 x 7)
0.006265436427282263
Using np.linalg.det,time to solve det is  0:00:00.006931  seconds.
0.006265436427282271
Using mydet,time to solve det is  0:00:00.026277  seconds.
(8 x 8)
-0.03521925762178388
Using np.linalg.det,time to solve det is  0:00:00.054221  seconds.
-0.0352192576217839
Using mydet,time to solve det is  0:00:00.213624  seconds.
(9 x 9)
0.0042511263867378935
Using np.linalg.det,time to solve det is  0:00:00.006038  seconds.
0.004251126386737917
Using mydet,time to solve det is  0:00:01.328202  seconds.
(10 x 10)
-0.004523545109875117
Using np.linalg.det,time to solve det is  0:00:00.005984  seconds.
-0.004523545109875182
Using mydet,time to solve det is  0:00:13.155335  seconds.
```



用我自己設的10*10矩陣，mydet較快。

```
52899.99999999993
Using numpy,time to solve det is  0:00:00.020343  seconds.
52900.0
Using mydet,time to solve det is  0:00:00.009016  seconds.
```

(f) 比較 np.linalg.solve、mysolve_adj、mysolve_cramer 的執行時間

因為有些function運算的執行時間太短，會顯示0，所以我運用迴圈(for i in range(0,100))將所有運算都跑100次，以下實際執行時間都要除以100才是正確的。很明顯的可以看出來，np.linalg.solve的執行速度小於其他兩者。

```
Using numpy,time to solve det is  0:00:00.000997  seconds.
Using mydet,time to solve det is  0:00:00.112700  seconds.
Using np.linalg.solve, time to solve Ax=b is  0:00:00.000883  seconds.
Using mysolve_adj, time to solve Ax=b is  0:00:04.832379  seconds.
Cramer's rule, Execution Time =  0:00:03.825193  seconds.
```

(g) residuals

運用||Ax-b|| (歐式距離)，計算誤差，mysolve_adj 與 mysolve_cramer 皆和 np.linalg.solve 的誤差差不多，皆很小，因此可推論準確度皆很高。

```
Using numpy,time to solve det is  0:00:00  seconds.
Using mydet,time to solve det is  0:00:00.001031  seconds.
Using np.linalg.solve, time to solve Ax=b is  0:00:00  seconds.
rediduals of np.linalg.solve
[[-2.22044605e-16]
 [ 4.44089210e-16]
 [ 0.00000000e+00]
 [-1.11022302e-16]
 [ 0.00000000e+00]
 [ 3.33066907e-16]
 [-4.44089210e-16]
 [ 0.00000000e+00]
 [ 0.00000000e+00]
 [ 0.00000000e+00]]
Using mysolve_adj, time to solve Ax=b is  0:00:00.051857  seconds.

rediduals of mysolve_adj
[[-2.22044605e-16]
 [-4.44089210e-16]
 [ 0.00000000e+00]
 [ 2.22044605e-16]
 [ 0.00000000e+00]
 [ 0.00000000e+00]
 [-2.22044605e-16]
 [-4.44089210e-16]
 [ 0.00000000e+00]
 [ 0.00000000e+00]]
Execution Time =  0:00:00.038945  seconds.

rediduals of mysolve_cramer
[[ 0.00000000e+00]
 [ 4.44089210e-16]
 [ 0.00000000e+00]
 [ 2.22044605e-16]
 [ 0.00000000e+00]
 [ 0.00000000e+00]
 [ 0.00000000e+00]
 [-4.44089210e-16]
 [ 0.00000000e+00]
 [ 0.00000000e+00]]
```

(h) Bonus

numpy.linalg.det是運用 **LU decomposition**這種演算法來解， LU decomposition就是先算上三角矩陣和下三角矩陣，然後用 det(A)=det(P)det(L)det(U) 或 det(A)=det(L)det(U) (如果沒有permutation matrix)，在矩陣較大時可以節省很多時間。下圖為numpy函式庫內對於det的實作，註解中有寫到是使用LUdecomposition來做。

```python
def det(a):
    """
    Compute the determinant of an array.

    Parameters
    ----------
    a : (..., M, M) array_like
        Input array to compute determinants for.

    Returns
    -------
    det : (...) array_like
        Determinant of `a`.

    See Also
    --------
    slogdet : Another way to represent the determinant, more suitable
      for large matrices where underflow/overflow may occur.

    Notes
    -----

    .. versionadded:: 1.8.0

    Broadcasting rules apply, see the `numpy.linalg` documentation for
    details.

    The determinant is computed via LU factorization using the LAPACK
    routine z/dgetrf.

    Examples
    --------
    The determinant of a 2-D array [[a, b], [c, d]] is ad - bc:

    >>> a = np.array([[1, 2], [3, 4]])
    >>> np.linalg.det(a)
    -2.0

    Computing determinants for a stack of matrices:

    >>> a = np.array([ [[1, 2], [3, 4]], [[1, 2], [2, 1]], [[1, 3], [3, 1]] ])
    >>> a.shape
    (3, 2, 2)
    >>> np.linalg.det(a)
    array([-2., -3., -8.])

    """
    a = asarray(a)
    _assertRankAtLeast2(a)
    _assertNdSquareness(a)
    t, result_t = _commonType(a)
    signature = 'D->D' if isComplexType(t) else 'd->d'
    r = _umath_linalg.det(a, signature=signature)
    r = r.astype(result_t, copy=False)
    return r
```

以下為一個LU decomposition的實例：

$$A = P.L.U$$

where

$$A = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 1 & 10 & 3 & 4 \\ 5 & 3 & 2 & -4 \\ 4 & 8 & 7 & 9 \end{pmatrix}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 4 & \frac{37}{32} & 1 & 0 \\ 5 & \frac{47}{32} & \frac{181}{135} & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} 1 & 10 & 3 & 4 \\ 0 & -32 & -5 & -7 \\ 0 & 0 & -\frac{135}{32} & -\frac{221}{32} \\ 0 & 0 & 0 & -\frac{602}{135} \end{pmatrix}$$

(g) bonus2

在mydet中我有運用條件判斷if A[0][i]!=0，可以節省執行時間，因為如果有很多0(sparse matrix)，計算det+=A[0][i]*((-1)**i)*mydet(minor)時，會做很多不必要的運算，因為A[0][i]如果是0，整項都會變0，就不用再算該項的minor，節省不必要的時間浪費。

(h) bonus reference

Finding determinant of 4*4 Matrix via LU Decomposition?
https://math.stackexchange.com/questions/831823/finding-determinant-of-44-matrix-via-lu-decomposition?fbclid=IwAR0RLFsgCs1nlfFVO1ixlSa8PAJgtS1TM5Uj5S23e5e0joLAyy-cO6hP2-E