**Ex2.1**

**2.**



**5.**



This implies that the value of residual vectors doesn't imply the closeness to the exact solution.

**6.**

**a.**



**b.**



**c.**



**Ex2.2**

**2.**



**14.**





**23.**

**a.**



**b.**



same since just the coefficients are divided by certain values.

**Com**

**4.**

refer to 2.2.4.py



n 2

x [1.0000000000000004, 0.9999999999999993]

n 3

x [0.9999999999999986, 1.0000000000000095, 0.99999999999999]

n 4

x [0.999999999999976, 1.0000000000002687, 0.999999999999351, 1.0000000000004226]

n 5

x [0.9999999999999204, 1.0000000000013662, 0.9999999999944567, 1.0000000000079992, 0.9999999999962228]

n 6

x [0.999999999999128, 1.0000000000244573, 0.9999999998362857, 1.0000000004227552, 0.9999999995358216, 1.0000000001821694]

n 7

x [0.9999999999941098, 1.0000000002321485, 0.9999999977816244, 1.0000000085746124, 0.9999999843481169, 1.000000013479372, 0.9999999955860505]

n 8

x [0.9999999999629096, 1.0000000020206763, 0.9999999733186062, 1.000000145577689, 0.9999996055966544, 1.0000005608647378, 0.9999995992268776, 1.0000001134611476]

n 9

x [0.9999999997143316, 1.0000000195858796, 0.9999996699969852, 1.0000023479257762, 0.9999914100871531, 1.0000175034755743, 0.9999799311179818, 1.0000121054884321, 0.9999970124141845]

n 10

x [0.9999999994365795, 1.0000000483187004, 0.9999989769226189, 1.0000092556038491, 0.9999560325804482, 1.0001204457317074, 0.9998029795158074, 1.0001898973174332, 0.9999005364046613, 1.000021828535638]

n 11

x [0.9999999906480918, 1.0000009834059937, 0.9999743874451622, 1.0002873530537422, 0.9982828954270323, 1.0060532743213821, 0.986788857584985, 1.0180482527417325, 0.9849802903408135, 1.0069607958176419, 0.998622913062291]

n 12

x [0.9999999793651592, 1.0000025399149661, 0.9999219272333931, 1.0010443795695307, 0.9924584900301034, 1.0327198786824772, 0.9098067254203248, 1.1617615222440647, 0.8118715523707013, 1.1368135799071313, 0.9434708220702548, 1.010128614208432]

n 13

x [1.0000001451655824, 0.9999770568221985, 1.0008897501502898, 0.98511210231129, 1.134255110595516, 0.26911032689557185, 3.5580324483407386, -4.948225019051731, 10.286338456794896, -8.620361730941452, 7.3421622759125205, -1.408152235984949, 1.4008614537127868]

n 14

x [1.0000000250901557, 0.9999958698341075, 1.000165783282548, 0.9971317886892425, 1.026802240359601, 0.848052105509815, 1.5581975409133801, -0.37872524872398355, 3.3279544672219106, -1.6839816613710972, 3.0686598077856124, -0.012946798377926654, 1.2823053288111672, 0.9663887691730095]

n 15

x [0.9999999844303006, 1.0000000413199344, 1.0000829405288594, 0.9972546945637085, 1.0369912017989378, 0.7306470438066601, 2.184577145170565, -2.298101282591433, 6.820614553709128, -5.07405936244288, 3.572039694023297, 2.7651913871993714, -2.0273428292290805, 2.611135128543185, 0.680969578262405]

when n is bigger than 10, some of the values are far from 1.



this matrix is almost singular or ill-conditioned so after the gaussian elimination, the resulting b values tend to get very smaller than in the beginning, which means the significances are lost in the process.

**9.**

refer to 2.2.9.py

scaled partial pivoting gives [0.21602476700841294, -0.007915106087778418, 0.6352433264931048, 0.7461742760857158] as the result.

relative errors are 1.8553967405475773e-14%, 0.0%, 1.5500495980804978e-14%, and 4.235471720076896e-14%.

naive gives [0.2160247669961285, -0.007915106087324868, 0.6352433264885665, 0.7461742760893735]

relative erros are 0.0%, 4.3516928528595966e-11%, 2.1297991487545654e-09%, and 1.318107035766064e-09%.

scaled partial pivoting gives more accuracy especially when there is some very small coefficients in row-equilibrated form.

**Ex2.3**

**3.**



**6.**



dominance is preserved.

**Com**

**6.**

refer to 2.3.6.py

since this symmetric tridiagonal system has only two values, there is no need to use any vectors to express the matrix.

ratios and diagonals can be reused for different bs.

new bs are processed with ratios and then the solutions are calculated with diagonal information.

Ax=b is shown below.

1.0000000000000002

2.000000000000001

2.999999999999999

3.9999999999999987

4.999999999999999

6.000000000000001

7.0

8.0

9.0

9.999999999999996

11.000000000000004

12.000000000000002

12.999999999999996

13.999999999999996

15.0

15.999999999999996

16.999999999999996

17.999999999999996

19.0

20.0

20.999999999999993

21.999999999999996

22.999999999999993

24.0

25.0

25.999999999999993

26.999999999999996

27.999999999999993

29.000000000000007

29.999999999999993

**10.**

refer to 2.3.10.py

for the following configuration, n=7, b=[12,23,42,51,33,24,15]

code showed almost no error.

12.0

23.0

42.0

51.0

33.0

24.0

15.000000000000002