6.6 – 1

A =

[[ 4 -2 1]

[ 3 6 -4]

[ 2 1 8]]

A-1 =

[[ 0.19771863 0.06463878 0.00760456]

[-0.121673 0.11406844 0.07224335]

[-0.03422053 -0.03041825 0.11406844]]

A\*A-1= should be the identity matrix

[[ 1.00000000e+00 -2.08166817e-17 0.00000000e+00]

[ -5.55111512e-17 1.00000000e+00 1.11022302e-16]

[ 0.00000000e+00 -2.77555756e-17 1.00000000e+00]]

A-1calculated – A-1exact =

[[ 0.00000000e+00 0.00000000e+00 6.07153217e-18]

[ 0.00000000e+00 0.00000000e+00 2.77555756e-17]

[ 0.00000000e+00 -3.46944695e-18 1.38777878e-17]]

The error in the inverse of the matrix is on the same order of magnitude as the error in the identity matrix; however, fewer terms are actually affected by the difference.

6.6-2

The vectors(going vertically) of the x solutions are:

[[ 1. 0.31178707 2.31939163]

[-2. -0.03802281 -2.96577947]

[ 4. 2.67680608 4.79087452]]

Can I just add that this is a dumb problem…

6.6-3

The eigen values for the matrix

|  |  |
| --- | --- |
| A | B |
| -B | A |

Is the same as for the matrix for

|  |  |
| --- | --- |
| 1 | 1 |
| -1 | 1 |

And then multiplied by the vector [A,B]

Eigen values : [ 1.+1.j ; 1.-1.j]

Eigen Vectors : [[ 0.70710678+0.j 0.70710678-0.j ]

[ 0.00000000+0.70710678j 0.00000000-0.70710678j]]

Which is just scaled sqrt[2]/2 – which is a normalization factor for the eigenvalues.

6.6.1

11.2.1 – 1

For 2 arrays that are 1000 x 1000 :

ABij = 12870.000000
for the regular summation,

the time is t2-t1 = 0.004723

ABij through slicing is 12870.000000
for the sliced summation,

the time is t2-t1 = 0.000041

Slicing is ~ 100x faster (115)

6.6.1,

I honestly don’t understand what the question is – we are provided the answer. So I just played around with it, until I felt that I built up an intuition on how it works.

 - 6.6.2 (extra credit)