

Smart Products Lab 1 & 2

Tyler Morrison

February 2, 2019

Section 1.

a) Data Structures Vectors are formed as vectors of floats using the STL container `vector<int>`. Matrices are vectors of vectors (i.e: `vector<vector<int>>>`).

b) Extra Functions

- `float dotv(const VecF_t, const VecF_t)`: Dot product of two vectors.
- `MatF_t initIdent(const unsigned int)`: Initialize an identity matrix.
- `void swapRows(MatF_t&, const int, const int)`: Swap two rows of a matrix.
- `float detm(const MatF_t)`: Compute determinant of a matrix. (Gauss-based)
- `void fwdElim(MatF_t&)`: Conduct forward elimination on matrix. (Gauss-based)
- `void bckElim(MatF_t&)`: Conduct backward elimination on matrix. (Gauss-based)
- `void rref(MatF_t&)`: Get reduced row-echelon form of matrix. (Gauss-based)
- `int rankm(const MatF_t)`: Get rank of a matrix. (Gauss-based)

c) Implementation of SPMath functions in SPmatrix The matrix class in `SPmatrix` calls functions from `SPmath` whenever it can so that functions are not redefined. For example:

```
1 MatF MatF::inv() {  
2     MatF B(invertm(mat_data_));  
3     return B;  
4 }
```

Section 2.

a) Matrix inversion algorithm To do matrix inversion, I used the forward elimination algorithm from Wikipedia: Gaussian elimination, and completed the backward elimination myself.

b) Technique name An algorithm for Gauss-Jordan elimination with pivots selected by numerical magnitude was used because it has relatively low complexity compared to other methods ($\mathcal{O}(n^3)$) and good numerical stability for poorly conditioned matrices.

c) Invertible check method A check was added to `invertm()` to determine if, after forward elimination, the left hand side of the composite matrix has any zeros in the diagonal.

If there are zeros, it throws a runtime error for a non-invertible matrix, otherwise it continues onto backward elimination.

Section 3.

a) **What info do you think would have helped you to better complete the lab?** Nothing that I can think of right now. The suggestion to practice on HackerRank helped me get a handle on C++.

b) **Output from labone.cpp and labtwo.cpp** Included below are the outputs of the compiled programs as they were piped out of the command line.

Program 1: Output of labone.cpp

```
1 Adding Vectors (C=A+B): C =
2 5
3 5
4 5
5 5
6 5
7 5
8 5
9 5
10 5
11 5
12 error!: addv(): Cannot add vectors of different dimensions!
13
14 Fat Matrix: Am =
15 2.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000
16 2.000 2.000
17 3.000 3.000 3.000 3.000 3.000 3.000 3.000 3.000
18 3.000 3.000
19 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000
20 5.000 5.000
21
22 transposed: Am =
23 2.000 3.000 5.000
24 2.000 3.000 5.000
25 2.000 3.000 5.000
26 2.000 3.000 5.000
27 2.000 3.000 5.000
28 2.000 3.000 5.000
29 2.000 3.000 5.000
30
31 Skinny Matrix: Bm =
32 1.000 1.000 1.000
33 2.000 2.000 2.000
34 3.000 3.000 3.000
35 4.000 4.000 4.000
36 5.000 5.000 5.000
```

```

37
38 transposed: Bm =
39 1.000      2.000      3.000      4.000      5.000
40 1.000      2.000      3.000      4.000      5.000
41 1.000      2.000      3.000      4.000      5.000
42
43 Square Matrix: Cm =
44 1.000      1.000      1.000
45 2.000      2.000      2.000
46 3.000      3.000      3.000
47
48 transposed: Cm =
49 1.000      2.000      3.000
50 1.000      2.000      3.000
51 1.000      2.000      3.000
52
53 Cm resized with pointer
54 1.000      2.000      3.000
55 1.000      2.000      3.000
56 1.000      2.000      3.000
57 0.0000     0.0000     0.0000
58 0.0000     0.0000     0.0000

```

Program 2: Output of labtwo.cpp

```

1 Matrix A =
2      5      3.3      2.34
3      6.7      7.32      9.3
4      8.123      7.345      22.34
5
6 Matrix Ainv = inverse(A)
7      0.4589     -0.2725     0.06536
8     -0.3573      0.4468     -0.1486
9     -0.0494     -0.04781     0.06984
10
11 Matrix B = A * Ainv
12      1 -7.451e-08  7.451e-08
13 -8.941e-08      1  5.96e-08
14      0 1.192e-07      1
15
16 Matrix C = inverse((transpose(A) * Ainv + A - B) * A)
17      2.089      -1.786      0.524
18     -2.216      1.94      -0.5782
19     -0.01374 -0.007176  0.007603

```