Assignment 1
Matrix Computations and Applications
v. 2 — 2013-09-10

The deadline for this assignment can be found at:
http://www8.cs.umu.se/kurser/5DA002/HT13/timetable.html
(Link Planning and Readings on the course homepage.)

The submission should consist of:

• A hardcopy of the complete report, including an appendix with the source code, to be put in the assignment box labeled 5DA002 — Matrix Computations and Applications located at level 4 in the MIT building (just outside the support staff offices).

• The source code should be available in a folder called edu/5da002/assN in your home folder, where N is the assignment number.
1 Introduction

This assignment consists of three parts: Theory, algorithms, and applications. The purpose of the assignment is primarily to assess your theoretical knowledge, your ability to express yourself clearly in writing, your ability to understand and implement numerical algorithms, and your ability to solve practical problems using the tools of linear algebra.

Unless otherwise stated, the assignment should be performed individually. You are allowed (indeed encouraged) to discuss the assignment with other students. However, all solutions, theory, code, etc. should be constructed by yourself. You may be required to explain your submitted assignment.

2 Theory

1. (Laws of matrix addition) Matrix addition obeys the following laws: The commutative law, the distributive law (with respect to scalar multiplication), and the associative law. State and prove all three laws from the corresponding laws of scalar addition. (Hint: See §2.4 in Strang.)

2. (Laws of matrix multiplication) Matrix multiplication obeys the following laws: The distributive law from the left, the distributive law from the right, and the associative law. State all three laws and prove the left or right distributive law. (Hint: See §2.4 in Strang.)

3. (Block multiplication) Matrix multiplication can be applied to matrices partitioned into blocks if and only if the block shapes match. In particular,

\[
\begin{pmatrix}
A & B \\
C & D
\end{pmatrix}
\begin{pmatrix}
I \\
0
\end{pmatrix}
= \begin{pmatrix}
AC + BD \\
0
\end{pmatrix}
\]

Perform the block multiplication

\[
\begin{pmatrix}
A & B \\
C & D
\end{pmatrix}
\begin{pmatrix}
I & -A^{-1}B \\
0 & I
\end{pmatrix}
\]

The block that appears in the bottom right corner is known as the Schur complement of the matrix on the right.

4. (Back substitution) Upper triangular linear systems can be solved quickly using the back substitution algorithm. Solve the upper triangular linear system

\[
\begin{align*}
-2x + 4y + 1z &= 9 \\
4y - 3z &= -1 \\
2z &= 6
\end{align*}
\]

by back substitution. Specifically, start with the third equation and solve for z. Substitute z in the other equations. Solve the second equation for y and substitute y in the first equation. Finally, solve the first equation for x. For n equations, how many arithmetic operations does the back substitution algorithm perform?

5. (Elimination without pivoting) Transform the general linear system

\[
\begin{align*}
x + y + 2z &= 7 \\
2x + 3y + 5z &= 17 \\
x + 3y + 5z &= 14
\end{align*}
\]

to an equivalent upper triangular system by elementary row operations. Specifically, annihilate x from the second equation by adding a suitably chosen multiple of the first equation to
the second equation. Annihilate $x$ from the third equation by adding a suitably chosen multiple of the first equation to the third equation. Finally, annihilate $y$ from the third equation by adding a suitably chosen multiple of the second equation to the third equation.

6. (Block elimination) Consider the square block matrix

$$A = \begin{pmatrix} B & C \\ D & E \end{pmatrix},$$

where both blocks $B$ and $E$ are square. If we pre-multiply $A$ by the conforming block matrix

$$\begin{pmatrix} I & X \\ 0 & I \end{pmatrix},$$

then we get

$$\begin{pmatrix} I & X \\ 0 & I \end{pmatrix} \begin{pmatrix} B & C \\ D & E \end{pmatrix} = \begin{pmatrix} B + XD & C +XE \\ D & E \end{pmatrix}.$$ 

Note that the product is identical to $A$ except in the first block row. Find the matrix $X$ such that the $(1,2)$ block vanishes. Which assumption(s) did you have to make? You have just performed one step of block elimination. Compare with one step of ordinary elimination at the element level.

3 Algorithms

1. (Back substitution) Implement back substitution in MATLAB. Your function header should look like this:

```matlab
function x = backsubst( A, b )
```

where $A$ is the coefficient matrix (upper triangular) and $b$ is the vector of right-hand sides. The output vector $x$ is the computed solution to the upper triangular linear system $Ax = b$. Try to use MATLAB vector notation where appropriate (i.e. use as few for loops as possible). Recall that row $i$ and column $j$ of $A$ in MATLAB notation are given by $A(i,:)$ and $A(:,j)$, respectively. Compare your results against MATLAB’s backslash operator on random upper triangular matrices. Your report should include your code and the results of your tests.

2. (Triangular matrix inversion) Use your back substitution function to implement triangular matrix inversion in MATLAB. Recall that the inverse of a square matrix $A$ is that matrix $B$ for which $AB = I$ holds, where $I$ is the identity matrix. If we equate columns of this matrix equation, we get the $n$ equations

$$Ab_i = e_i \quad \text{for } i = 1, 2, \ldots, n,$$

where $b_i$ is the $i$-th column of $B$ and $e_i$ is the $i$-th column of the identity matrix. If $A$ is upper triangular, then we can use our backsubst function to solve the above equations for each column of the inverse in turn. Your MATLAB function for triangular inversion should have the function header

```matlab
function B = triinv( A )
```

where $A$ is an upper triangular matrix and $B$ is its inverse. You should verify correctness by checking the norm of $AB - I$ (which should be close to zero). Your report should include your code and the results of your tests.
4 Applications

4.1 Background

You have been hired by a company that produces software for an air traffic control system to work on the Early Collision Warning subsystem. Your first assignment is to write a piece of software that computes the point where two aircraft might collide. As input you receive the current positions $p_1, p_2 \in \mathbb{R}^2$ (unit: meter) and the current velocities $v_1, v_2 \in \mathbb{R}^2$ (unit: meter per second) of two aircraft that have the same altitude and might therefore collide. An illustration of the situation is shown in Figure 1.

![Figure 1: Schematic illustration of two aircraft on a potential collision course.](image)

The requirements for your module are as follows:

1. If the paths of the aircraft will not intersect in the future, output **No collision possible**.
2. If the two aircraft will not cross paths in the next 10 minutes, then output **No imminent collision**.
3. If the two aircraft cross paths within the next 10 minutes but will miss each other by at least 30 seconds, then output **Collision avoided by $\alpha$ seconds at $(x, y)$**, where $\alpha$ is the number of seconds by which the collision was avoided and $(x, y)$ is the point of intersection.
4. If the two aircraft cross paths within the next 10 minutes and will miss each other by less than 30 seconds, then output **Warning! Potential collision in $\alpha$ seconds at $(x, y)$**, where $\alpha$ is the number of seconds before the first aircraft arrives at the intersection point $(x, y)$.

While thinking about how to solve the problem, you realize that the position of aircraft $i$ at time $t$ is given by the function

$$p_i(t) = p_i + tv_i,$$

which describes a line in two-dimensional space. Most likely, these lines will neither be parallel nor collinear, which means that there is a unique point of intersection. Maybe it is possible to setup a linear system from the position of the first aircraft at time $t_1$ and the position of the second aircraft at time $t_2$ and solve for the unknown times? That would give the intersection point as well as the times at which the aircraft reach the intersection point.

4.2 Implementation

Implement a MATLAB function with the function header

```matlab
function [x, y, t1, t2] = collision( p1, v1, p2, v2 )
```
that computes the quantities mentioned above. Use this function to implement the required behavior. Test your implementation on carefully chosen test cases to make sure that it fulfills all four requirements in the list above. As with many linear algebra problems, plotting will help you to understand the problem and simplify a visual “reality check” of your results.

Hint: If \( \mathbf{a} \) and \( \mathbf{b} \) are 2-vectors in Matlab, a blue line between them is plotted by `plot([a(1),b(1)],[a(2),b(2)],'b')`. If you wish to plot individual points stored in a 2-by-N array \( \mathbf{X} \), e.g. a tick for each minute, you can use `hold on; plot(X(1,:),X(2,:),'r*'); hold off` to plot the points as red stars.

Your report should include your code and the results of your tests, including plots.

### 4.3 Connection to reality

The problem in this assignment is a simplification of the real-world problem of collision avoidance in Air Traffic Control (ATC). In reality, the aircraft are not always in straight-and-level flight and hence move in 3 dimensions. Furthermore, the exact movements of the aircraft during the coming 10 minutes are not always known. However, the ATC controllers have some tools at their disposal that are similar to what you have implemented. Figure 2 shows two such tools. Two aircraft (black diamonds) are visible; NTJ805 (NextJet flight 805) from Mariehamn to Stockholm/Arlanda, a Saab 340 at flight level 108 (FL108, 10800 feet), climbing towards FL120 with a ground speed of 184 knots, and TFL7620 (Arke Fly flight 7620), a Boeing 767 en route to Amsterdam/Shiphol, cleared to FL380, currently cruising at FL360 with a ground speed of 426 knots. In the left view, the 5-minute vectors alternate between black and white for every minute and clearly shows the speed difference between the aircraft. The plot shows that TFL7620 will arrive at the intersection point in about 4 minutes, 1 minute ahead of NTJ805. In the right view the minimum horizontal separation between the aircraft has been calculated to 2 nautical miles. This is below the horizontal separation limit of 5 nautical miles, but since the vertical separation is (a lot) more than 1000 feet, no action need to be taken by the controller to avoid a collision.

![Figure 2: Real-world radar views from Air Traffic Control Center (ATCC) Stockholm. The image shows the Swedish (light gray) and Finnish (dark gray) airspace over the Sea of Åland (Ålands hav). The island(s) of Åland is visible to the upper right. The Swedish coast near Norrtälje is to the left.](image-url)