CS170A — Mathematical Modeling & Methods for Computer Science

HW#0 *Matlab and Basic Linear Algebra*

Due: 3:59pm Wednesday January 22, 2014

D. Stott Parker, Xingyu Ma, Costas Sideris

`stott@cs.ucla.edu`, `maxy12@cs.ucla.edu`, `costas@cs.ucla.edu`

Using CourseWeb, please turn in your Matlab code and resulting output (enough to show that the program is working) as a document, such as a MSWord file or PDF document.

These problems draw on the *Matrix Algebra Review* for this class that was posted on courseweb.

1. **Outer Products (10 points)**
   The *Matrix Algebra Review* defines banded matrices with constraints on subscripts i and j:

Kind of matrix A	*condition on elements a_{ij} of A*
Diagonal	$a_{ij} = 0$ if $\|i - j\| > 0$
Tridiagonal	$a_{ij} = 0$ if $\|i - j\| > 1$
Upper Triangular	$a_{ij} = 0$ if $i - j > 0$
Lower Triangular	$a_{ij} = 0$ if $i - j < 0$
Upper Hessenberg	$a_{ij} = 0$ if $i - j > +1$
Lower Hessenberg	$a_{ij} = 0$ if $i - j < -1$

   (a) Write a Matlab function `Banded(A, Type)` that takes a $n \times n$ matrix `A` as input, and a string `Type` that can take any of the values `'Diagonal'`, `'Tridiagonal'`, `'Upper Triangular'`, `'Lower Triangular'`, `'Upper Hessenberg'`, `'Lower Hessenberg'`. The function should then yield a matrix like `A` except that all values outside the specified band set to zero.

   For example, the function call `Banded( ones(5), 'Tridiagonal' )` should yield the matrix

```
1   1   0   0   0
1   1   1   0   0
0   1   1   1   0
0   0   1   1   1
0   0   0   1   1
```

   In order to make this interesting, your function cannot use for loops or while loops — it must use Matlab operators only.

   Hint: the `cauchy` function provided with this assignment may help. (The function should also be in your Matlab installation in `.../matlab/elmat/private/cauchy.m`)

   (b) Also write a related Matlab function `TestBanded(A, Type)` that takes the same inputs as `Banded`, but yields the value 1 if the matrix `A` is of the Bandedness type specified, and 0 if not. To make this interesting, you must implement this in a single line by filling in the '_'s in this function definition:

   `TestBanded = @(A,Type)` __________ `Banded(A,Type)` __________

   This is a Matlab way of defining a function without using the usual `function` declaration.

   Hint: consider using the `prod` function on a logical matrix.

2. **Euler and Tait-Bryan Angles**

   Yaw, pitch, and roll are Aircraft principal Axes. As illustrated in Figure 1, assuming the object (like an airplane) is directed on its axis of motion, these three rotations are as follows:

yaw	ϕ	rotating left and right, i.e., rotating horizontally
roll	θ	'banking', i.e., rotating around the axis of motion
pitch	ψ	rotating up and down, i.e., rotating vertically.

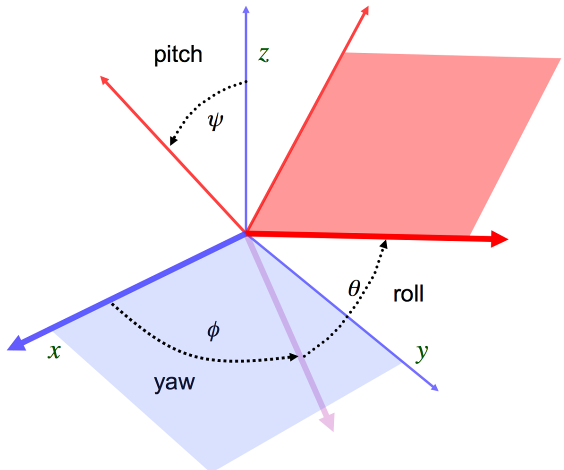

Figure 1: Three-dimensional rotation as a composite of pitch, roll, and yaw

A sequence of 2-dimensional rotations can produce any 3-dimensional rotation. For example consider a product of three 2D rotations:

$$R_{123}(\phi,\ \theta,\ \psi) \;=\; \underbrace{\begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{R_{12}(\psi)} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}}_{R_{23}(\theta)} \underbrace{\begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{R_{12}(\phi)}$$

This 3D transformation rotates first in dimensions 1 and 2 xy plane (around the z axis), then in 2 and 3 (around the x axis), and finally in 1 and 2 again. Any rotation of 3-space can be put in this form. The angles $(\phi,\ \theta,\ \psi)$ are known as the *Euler angles*. But in spatial navigation, a different convention for the sequence of rotations is used:

$$R_{123}(\phi,\ \theta,\ \psi) \;=\; \underbrace{\begin{pmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{pmatrix}}_{R_{13}(\psi)} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}}_{R_{23}(\theta)} \underbrace{\begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{R_{12}(\phi)}$$

(a) Write Matlab functions for both of these rotation matrices, and find angles showing they are different.

(b) For the first rotation matrix, prove that the angles (ϕ, θ, ψ) and $(-\psi, -\theta, -\phi)$ yield matrices that are inverses of each other.

The point here: the convention of using the first and third rotation in the same 2D space permits easy construction of inverse rotations.

(c) **Extra Credit (5pts)**: Write a Matlab function that, given a 3D rotation matrix `R`, yields the corresponding angles (ϕ, θ, ψ).

3. **Determinants**

Assuming A and B are square real symmetric matrices, prove that

$$\det(\,A'\,B'\,) \;=\; \overline{\det(A)}\;\overline{\det(B)}.$$

Hint: use eigenstructure.

4. **Moler**
The course syllabus includes a reference to an online book:

   Cleve Moler, *Numerical Computing with MATLAB*, 2004. (online text, free access)
   - `http://www.mathworks.com/moler/index.html` The book's home page
   - `http://www.mathworks.com/moler/lu.pdf` chapter on Linear Systems solution.
   - `http://www.mathworks.com/moler/eigs.pdf` chapter on Eigenvalues.

   (a) Download the chapter on Eigenvalues and Singular Values (`eigs.pdf`) and do problem 10.1.

   (b) Download the chapter on Linear Systems (`lu.pdf`), read section 2.11 on Page Rank, and do problem 2.23.

   The programs mentioned in these questions can also be downloaded from `http://www.mathworks.com/moler/ncmfilelist.html`

5. **Matrix Computations**
The file `http://www.cs.cornell.edu/courses/CS4220/2013sp/CVLBook/chap5.pdf` has a very nice introduction to matrix computations in Matlab, from a book by Charles van Loan. At the end it includes a comparison of several ways of computing matrix products; one of these uses outer products:

```
function C = MatMatOuter(A,B)
% This computes the matrix-matrix product C = A*B (via outer products) where
% A is an m-by-p matrix, B is a p-by-n matrix.
[m,p] = size(A);
[p,n] = size(B);
C = zeros(m,n);
for k=1:p
   % Add in k-th outer product
   C = C + A(:,k)*B(k,:);
end
```

   (a) What is the result of `MatMatOuter( ones(3,2), ones(2,4) )` ?

   (b) Prove that the result of `MatMatOuter(A,B)` is the same as `A * B`.