Linear Systems

MAT 67L, Laboratory II

Contents

Instructions

(1) Read this document.
(2) The questions labeled "Experiments" are not graded, and should not be turned in. They are designed for you to get more practice with `MATLAB` before you start working on the programming problems, and they reinforce mathematical ideas.
(3) A subset of the questions labeled "Problems" are graded. You need to turn in `MATLAB` M-files for each problem via Smartsite. You must read the "Getting started guide" to learn what file names you must use. Incorrect naming of your files will result in zero credit. Every problem should be placed is its own M-file.
(4) Don't forget to have fun!

Linear Systems

Linear systems amounts to the study of functions

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$$

subject to the linearity property

$$f(\alpha x + \beta x') = \alpha f(x) + \beta f(x') \,.$$

Here α and β are real numbers and x, x' are points in $\mathbb{R}^n$ while the addition rule is the usual componentwise one. To check your understanding, ask yourself what sort of objects $f(x)$ and $f(x')$ are? The classic linear systems problem is to solve

$$f(x) = y$$

for the vector x given the vector y.

For example consider $f : \mathbb{R}^3 \longrightarrow \mathbb{R}^2$ given by

$$f(x_1, x_2, x_3) = (2x_1 - x_2, x_1 + x_2 + x_3) \,.$$

and

$$y = (1, -1)\,.$$

Lets try to solve $f(x) = y$. For that we need to write our two equations (because both $f(x)$ and y live in $\mathbb{R}^2$):

$$\begin{cases} 2x_1 - x_2 &= 1 \\ x_1 + x_2 + x_3 &= -1\,. \end{cases} \tag{1}$$

We could now solve these by hand using substitution or Guassian elimination. However, once the number of equations and variables becomes larger, it is better to use our friend MATLAB. For that we first rewrite our system in terms of MATrices!

The information of our example linear system (1) is encoded by the coefficients of x_1, x_2, x_3 on the left hand sides of the equations which we can arrange in a rectangular array, or matrix

$$M := \begin{pmatrix} 2 & -1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

and the vector y on the right hand side which we write as a column vector

$$y := \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

It is common to think of the unknowns x_1, x_2, x_3 also as a column vector $x := \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ and rewrite our equation as

$$\begin{pmatrix} 2 & -1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}\,.$$

(The juxtaposition of a 2×3 matrix and a 3×1 column vector denotes matrix multiplication and is defined exactly so as to produce a 2×1 column vector whose entries are precisely the left hand sides of the pair of equations in (1)).

Entering matrices and vectors in MATLAB

Lets solve for x for $Mx = y$ using `MATLAB`. First, we need to type M and y into `MATLAB`. We enter matrices and vectors by placing the numbers between the "[" and "]" symbols. Rows are separated semicolons and columns are separated by spaces or commas.

```
>> M=[2 -1 0; 1 1 1]

M =

     2    -1     0
     1     1     1
```

```
>> y=[1; -1]

y =

     1
    -1
```

Solving linear systems

First let us solve this system the "easy" way. Type:

```
>> x=M\y

x =

     0
    -1
     0
```

Hence $M(0,-1,0)^T = y$. But the story is not over. Is the solution unique? Next type

```
>> null(M)

ans =

   -0.2673
   -0.5345
    0.8018
```

The null command returns an orthonormal basis for the null space of M. This means M times a null space vector should give zero. Hence we can add a null space vector to our x vector and still get a solution for y. Type the above null space vector in `MATLAB` into a variable called z

```
>> M*(x + z)

ans =

    0.9999
   -1.0000
```

Hence $(0,-1,0)^T + (-0.2673,-0.5345,0.8018)^T$ is another solution. This is expected because two equations with three unknowns is not expected to have a unique solution.

In the following sections we will solve this system again in a different way that will give us more information about its solution space.

Gaussian Elimination

Many of you probably already have seen how to solve a linear system using Gaussian elimination. If not, watch this short video

http://www.math.ucdavis.edu/~linear/67/videos/chapter1_RREF.mp4

For our example system (1) we would begin with an augmented matrix and then perform row operations to achieve reduced row echelon form:

$$\left(\begin{array}{ccc|c} 2 & -1 & 0 & 1 \\ 1 & 1 & 1 & -1 \end{array}\right) \sim \cdots \sim \left(\begin{array}{ccc|c} 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & \frac{2}{3} & -1 \end{array}\right)$$

In this display, the $\cdots$ represent a bunch of row operations which would be laborious if performed by hand. Your friend MATLAB can help.

Reduced row echelon form in MATLAB

First make an augmented matrix called A from the matrix M and the right-hand-side vector y. Then type the command "rref" to row reduce the matrix A

```
>> A=[M y]

A =

     2    -1     0     1
     1     1     1    -1
>> format rat
>> rref(A)

ans =

     1     0     1/3     0
     0     1     2/3    -1
```

The "format rat" command forces MATLAB to print fractions. If you want decimal notation, type "format short".

Hence the solution is:

$$\begin{aligned} x_1 &= 0 - 1/3x_3 \\ x_2 &= -1 - 2/3x_3 \\ x_3 &= \text{anything} \end{aligned}$$

Or in vector notation

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} + w \begin{pmatrix} -1/3 \\ -2/3 \\ 1 \end{pmatrix},$$

where w is any number. Notice that solving the system with the "rref" command, we can easily find *all* solutions. Also the vector $(-1/3, -2/3, 1)^T$ is a scalar multiple of the null space vector we found above.

Lets do another example that will help with the problems below. Let A defined below and use `MATLAB`'s "rref" command. We want to find all x vectors such that $Ax = 0$.

$$A = \begin{pmatrix} 16 & 80 & -73 & -16 & -45 \\ 18 & 90 & -117 & -18 & -39 \\ 2 & 10 & -98 & -2 & 24 \\ 18 & 90 & -243 & -18 & 3 \\ 12 & 60 & -219 & -12 & 21 \end{pmatrix}; rref(A) = \begin{pmatrix} 1 & 5 & 0 & -1 & -13/3 \\ 0 & 0 & 1 & 0 & -1/3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2}$$

Solving for x_1 and x_3 we get $x_1 = -5x_2 + x_4 + 13/3x_5$ and $x_3 = 1/3x_5$. Or in vector notation

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = x_2 \begin{pmatrix} -5 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + x_4 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + x_5 \begin{pmatrix} 13/3 \\ 0 \\ 1/3 \\ 0 \\ 1 \end{pmatrix}$$

Hence any linear combinations of these vectors will be mapped to 0 by A

Problems

Problem 1.

Given a matrix A and vector b, find the number of solutions x for $Ax = b$. The function will return 1 if there is a unique solution, 0 if there are no solutions, and -1 if the solution is not unique. The function signature is

```
%Input
%   A: m x n matrix
%   b: m x 1 vector
%Output
%   p: number of solutions.
%       1: there is a unique solution
%       0: there is no solution
%       -1: there are infinitly many solutions.
function [p] = ls_numberSolutions(A, b)
    code here
end
```

Problem 2.

Given a (maybe not square) matrix A, find A^{-1} without using the MATLAB command "inv". If you use "inv" that is cheating! If the inverse does not exist, return the empty list. The function signature is

```
%Input
%   A: n x m matrix
%Output
%   Ainv: A^-1 if this exist, otherwise [].
function [Ainv] = ls_inverse(A)
    code here
end
```

Hints:

(1) Use "rref" on a the matrix $[A\ B]$ where B is a special matrix.
(2) Non-square matrices are not invertible
(3) the i, j entry of a matrix in MATLAB is $A(i, j)$. The i^{th} row is $A(i, :)$, and the j^{th} column is $A(:, j)$.

Problem 3.

The goal of this problem is to solve $Ax = b$ for x, where A is a general $m \times n$ matrix. The answer should be an empty list if no solution exist, an $n-$vector if there is one unique solution, and a $n \times (1+k)$ matrix if the system is under determined where the first column $(x_{(p)})$ is a particular solution to $Ax_{(p)} = b$, and the other k columns $(x_{(n)})$ are linearly independent null space vectors (that is $Ax_{(n)} = 0$).

You are not allowed to use MATLAB's "null" function or your "ls_numberSolutions" function. You should use MATLAB's "rref" function, and then do a bunch of "if" and "for-loop" analysis on the resulting matrix to build up the answer.

The function signature is

```
%Input
%   A: m x n matrix
%   b: m x 1 vector
%Output
%   ansMatrix:
%       [] if there is no solution
%       column vector if there is a unique solution
%       else a matrix in the form [x, n1, n2, ..., nk]:
%           where Ax=b, Ani=0 for i=1 to k.
%           The set {n1, ..., nk} are linearly independent.
%Example
%   see below
function [n] = ls_findSolutions(A, b)
    code here
end
```

Consider example (2) of solving $Ax = 0$ (here $b = 0$). The particular solution is the zero vector, and the null space vectors are the ones computed in that example. If your MATLAB

function is given "ls_findSolutions(A, 0)," the output should be

$$\begin{pmatrix} 0 & -5 & 1 & 13/3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Problem 4.

A permutation matrix is a matrix where every row and column has exactly one 1 in it and all the outher elements are zero.

For example,

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

is a 3×3 permutation matrix. Note that $P^1 \neq I$ but $P^2 = I$. The smallest positive integer k such that $P^k = I$ is called the order of the permutation matrix P.

Write a function to find the order of a $n \times n$ permutation matrix. The function signature is

```
%Input
%   A: n x n permutation matrix
%Output
%   x: order of the permutation matrix.
%Example
%   ls_permutationOrder([1 0; 0 1]) gives 1
function [x] = ls_permutationOrder(P)
    code here
end
```

Hint: MATLAB's "eye" and "size" functions may be used.

Problem 5.

Consider the purmutation patrix P

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Notice that $Pe_1 = e_2$, $Pe_2 = e_3$, $Pe_3 = e_1$, $Pe_4 = e_5$, $Pe_5 = e_4$ where e_i is the i^{th} standard basis vector in $\mathbb{R}^5$. We say $(1, 2, 3)$ is a 3-cycle because P can be identified with mapping 1 to 2, 2 to 3, and 3 back to 1. Likewise, $(4, 5)$ is a 2-cycle.

Let an arbitrary permutation matrix have k_1 many 1-cycles, k_2 many 2-cycles, ..., and k_n many n-cycles. The cycle type of the permutation matrix is the row vector $(k_1, k_2, \ldots, k_n)$. For the example above, the cycle type is $(0, 1, 1, 0, 0)$

Write a function to find the cycle type vector of a $n \times n$ permutation matrix. The function signature is

```
%Input
%   A: n x n permutation matrix
%Output
%   t: cycle type vector.
%Example
%   for the permutation example above, the output is (0, 1, 1, 0, 0)
function [t] = ls_cycleType(P)
    code here
end
```