

MATH 100 – Introduction to the Profession

Linear Equations in MATLAB

Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Fall 2012



Where do systems of linear equations come up?

Everywhere!

- They appear straightforwardly in
 - analytic geometry (intersection of lines and planes),
 - traffic flow networks,
 - Google page ranks,
 - linear optimization problems,
 - statistical data fitting,
 - Leontief's input-output model in economics,
 - electric circuit problems,
 - the steady-state analysis of a system of chemical or biological reactors,
 - the structural analysis of trusses (see Exercise 5.6),
 - and many other applications.
- They are solved as the main step in the **linearization** of many nonlinear systems (in the solution of differential equations, optimization, etc.).
- One can certainly refer to them as **one of the workhorses of applied mathematics**.



Representation of Linear Systems

- Equation form:

$$x_1 + 2x_2 + 3x_3 = 1$$

$$2x_1 + x_2 + 4x_3 = 1$$

$$3x_1 + 4x_2 + x_3 = 1$$

- Matrix form: $A\mathbf{x} = \mathbf{b}$, with

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 4 \\ 3 & 4 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Remark

We always think of vectors as *column vectors*. If we need to refer to a row vector we use the notation \mathbf{x}^T (in mathematics) or \mathbf{x}' (in MATLAB).

Never use A^{-1} to solve $A\mathbf{x} = \mathbf{b}$

In linear algebra (MATH 332) you will learn that the solution of

$$A\mathbf{x} = \mathbf{b}$$

is given by

$$\mathbf{x} = A^{-1}\mathbf{b}.$$

This is correct, but **inefficient** and **more prone to roundoff errors**.

Always **use special algorithms** (preferably with some decomposition method such as LU, QR or SVD) **to solve linear systems** – even to compute the inverse itself (should you actually happen to need it).



Never use A^{-1} to solve $Ax = b$ (cont.)

Example

Consider the trivial “system” $7x = 21$ and compare solution via the “inverse” and by straightforward division.

Solution

- Division immediately yields $x = \frac{21}{7} = 3$.
- Use of the “inverse” yields

$$x = 7^{-1} \times 21 = 0.142857 \times 21 = 2.999997.$$

Clearly, use of the inverse requires **more work** (first compute the inverse, then multiply it into the right-hand side), and it is **less accurate**. This holds even more so for larger systems of equations.

Note that MATLAB is “smarter” than this, so that $7^{(-1)} * 21$ is still equal to 3.

How to solve linear systems by “division” in MATLAB

In order to mimic what we do (naturally) for a single equation, MATLAB provides two very sophisticated *matrix division* operators:

- For systems $A\mathbf{x} = \mathbf{b}$, we have the **backslash** (or `mldivide`) operator, i.e.,

$$\mathbf{x} = A \backslash \mathbf{b},$$

- and $\mathbf{x}^T A = \mathbf{b}^T$ is solved using a **forward slash** or (`mrdivide`) operator, i.e.,

$$\mathbf{x}^T = \mathbf{b}^T / A.$$

Remark

*These operators provide **black boxes** for the solution of (possibly even non-square or singular) systems of linear equations. They can be even extended to the cases $AX = B$ (see Exercise 5.4) and $XA = B$.*

Cramer's rule is especially inefficient!

n	Flops				
	10^9 (Giga)	10^{10}	10^{11}	10^{12} (Tera)	10^{15} (Peta)
10	10^{-1} sec	10^{-2} sec	10^{-3} sec	10^{-4} sec	negligible
15	17 hours	1.74 hours	10.46 min	1 min	$0.6 \cdot 10^{-1}$ sec
20	4860 years	486 years	48.6 years	4.86 years	1.7 day
25	o.r.	o.r.	o.r.	o.r.	38365 years

Table : Computer times for solving $n \times n$ linear systems using Cramer's rule on various computers ("o.r." stands for "out of reach"). Borrowed from [Scientific Computing with MATLAB and Octave (2010)].

- "Flops" stands for "floating point operation per second".
- Standard desktop PCs and laptops (Intel i5, i7) currently can perform on the order of about 10-50 gigaflops.
- Today's fastest supercomputer (LLNL's IBM BlueGene/Q *Sequoia*, see <http://top500.org/>) runs at 16 petaflops.



The World's Simplest Impossible Problem [Cleve's Corner]

"I'm thinking of two numbers and their average is 3."

What's the issue here?

- Not enough information is given (it is **under-determined**).
- So the problem is **not well-posed**.
- It does not have a unique solution, but has infinitely many solutions.

In MATLAB we get **different answers using different algorithms**:

- Using the **backslash operator**:

$$A = [1/2 \ 1/2], \quad b=3$$

$$x = A \backslash b$$

- Using the **pseudo-inverse**:

$$A = [1/2 \ 1/2], \quad b=3$$

$$x = \text{pinv}(A) * b$$



Compressed Sensing

An interesting recent article relating these two different algorithms (especially the backslash algorithm) to the hot research area of **compressed sensing** is

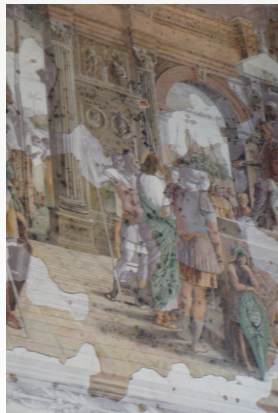
<http://www.mathworks.com/company/newsletters/articles/clevescorner-compressed-sensing.html>.

There are additional links at the end of this article.

The main idea of compressed sensing is to be able to **accurately reconstruct objects from very sparse information**.



Massimo Fornasier is a researcher in Linz, Austria, who does lots of work in compressed sensing.



[Math enters the picture] describes the reconstruction (using matrix encoding and so-called *circular harmonics*) of the Italian renaissance frescoes by Andrea Mantegna in the Ovetari Chapel in Padua.



Summary scripts

The basic commands for dealing with systems of linear equations in MATLAB are summarized in

- `lin_sys.m` (solving linear systems on the MATH 100 website)
- `linear_recap.m` (on the ExM website)



References I



T. A. Driscoll.

Learning MATLAB.

SIAM, Philadelphia, 2009.

http://epubs.siam.org/ebooks/siam/other_titles_in_applied_mathematics/ot115



D. J. Higham and N. J. Higham.

MATLAB Guide (2nd ed.).

SIAM, Philadelphia, 2005.

http://epubs.siam.org/ebooks/siam/other_titles_in_applied_mathematics/ot92



C. Moler.

Numerical Computing with MATLAB.

SIAM, Philadelphia, 2004.

http://www.mathworks.com/moler/index_ncm.html



References II



C. Moler.

Experiments with MATLAB.

Free download at

<http://www.mathworks.com/moler/exm/chapters.html>



A. Quarteroni, F. Saleri and P. Gervasio.

Scientific Computing with MATLAB and Octave (3rd ed.).

Springer, Berlin, 2010.



M. Fornasier.

Mathematics enters the picture.

Proceedings of the conference *Mathknow 2008*. <http://www.ricam.oeaw.ac.at/people/page/fornasier/mathsinpict.pdf>.



C. Moler.

Cleve's Corner. The World's Simplest Impossible Problem.

The MathWorks News & Notes, Vol.4 No.2, 1990. http://www.mathworks.com/company/newsletters/news_notes/pdf/dec1990cleve.pdf



References III



The MathWorks.

MATLAB 7: Getting Started Guide.

http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

