MATLAB AND LaTeX EXAMPLES

ERNIE AND RYAN
UNIVERSITY OF WASHINGTON

Abstract. This seems as good a place as any for an abstract. Our goal is to present some examples that may facilitate the use of LaTeX and MATLAB. This document itself *is* the LaTeX example. The .tex file this document was created from is in fact included in the appendix (non-recursively of course). A few tricky LaTeX points are also emphasized.

We also present several pieces of MATLAB code that illustrate how to write scripts, functions and even use the symbolic toolbox to tackle an example inverse problem.

Date: June 2002.

Contents

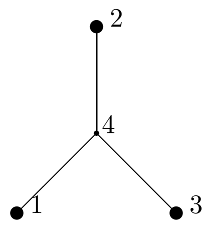

FIGURE 1. Y network

1. Introduction

We present four main examples:

1. Our first example will be to write a simple MATLAB function that computes the Dirichlet to Neumann map given the Kirchhoff matrix and the number of boundary nodes. We will also present some background information on this topic to show how it can be written up in LaTeX.
2. We present code for another MATLAB function that generates prime numbers. This is a good illustration of how flow control works in MATLAB.
3. There is also a MATLAB script that generates the Kirchhoff matrix for an example network (pictured in the script itself).
4. We will also take advantage of MATLAB's symbolic manipulator to solve the inverse problem for a simple Y network as shown in figure 1.

Finally we will briefly illustrate how this document was produced in LaTeXand give some useful LaTeX tricks, such as how to include graphics.

2. Example 1: Computing the Response Matrix

[1] For a network with n nodes, the Kirchhoff matrix K is an $n \times n$ symmetric matrix formed by taking for $i \neq j$

$$K(i,j) = \left\{ \begin{array}{ll} -\gamma(i,j) & \text{if there is an edge from i to j} \\ 0 & \text{if no such edge exists} \end{array} \right\}$$

Then the diagonal entries are chosen such that each row sums to zero. K operates on the vector of potentials u to give a vector of currents ϕ coming out of each node. ($Ku = \phi$) Naturally the current out of interior nodes is set to zero to agree with Kirchhoff's Law. It's useful to write the Kirchhoff matrix in the following block form:

$$K = \left[\begin{array}{cc} A & B \\ B^T & C \end{array} \right]$$

If all interior nodes are numbered such that they appear in the C block and all boundary nodes are in the A block, then the response matrix is just the Schur Compliment of K in C.

$$\Lambda = A - BC^{-1}B^T \tag{1}$$

[1]Background information is from [1].

It is shown in [1] that C is invertible because K is positive semi-definite. Λ acts on the vector of boundary potentials to give the vector of boundary currents. This is therefore the Dirichlet to Neumann map.

2.1. **MATLAB Code for Computing Λ.** The following assumptions are necessary:

- The Kirchoff matrix K is given
- K has the block form as above
- We know there are n boundary nodes

The following is a function written for MATLAB that will compute Λ. Note that functions and scripts are saved with a .m extension and can be run from the prompt in MATLAB.

```
%function L = getL(K,n)
function L = getL(K,n)
A = K(1:n,1:n);
C = K((n+1):end,(n+1):end);
B = K(1:n,(n+1):end);
L = A - B*inv(C)*B';
```

3. Example 2: A Prime Number Generator in MATLAB

This program does what the name suggests and also illustrates the syntax for for loops and if statements in MATLAB.

```
%function p=prime(n)
function p=prime(n)

% First prime number.
p = 2; j = 1;

while length(p) < n
    % Next number to consider.
    j = j+2;
    divisible = 0;

    for i = 1:length(p)

        if p(i) > sqrt(j)
            break;
        end

        if mod(j,p(i)) == 0
            divisible = 1;
            break;
        end

    end

    if divisible == 0
        p(length(p)+1) = j;
    end
end
```

4. Example 3: A MATLAB script

An advantage (or disadvantage) of a script over a function is that the variables are global. Any previously defined variable can be accessed by the script. Likewise, variables created or altered by the script can be accessed at the prompt.

The following code created a Kirchhoff matrix for a stick-figure shaped electrical network. It also computes the response matrix for this network by calling the function getL, which was presented in example 1.

```
% Create Kirchhoff matrix and response matrix for stick figure matrix.
%
%   5    1    2
%    \   |   /
%     10--6--7         1 ~ 5  Boundary nodes
%      |     |         6 ~ 10 Interior nodes
%      9-----8
%     /       \
%   4          3
%

p = prime(10);

K = zeros(10,10);

for i = 1:10
    if i <= 5
        K(i,i+5) = -p(i);
    elseif i <= 9
        K(i,i+1) = -p(i);
    else
        K(6,10) = -p(10);
    end
end

K = K + K'; for i = 1:10
    K(i,i) = -sum(K(i,:));
end

L = getL(K,5);
```

5. Example 4: MATLAB's symbolic manipulator

Symbolic variables can be declared using the syms command in MATLAB (see help file, which contains a list of all commands in the symbolic toolbox). It's often useful to do symbolic calculations to eliminate round-off error, which can present a problem even with double precision. Here, we demonstrate how MATLAB can symbolically solve a small system of equations.

There are of course limitations to this. Large systems quickly become too memory intensive. Symbolic calculations are naturally much more computationally intensive. For example, symbolically computing determinants and inverses of matrices is not at all practical for matrices larger than 6×6. But MATLAB can easily solve the system of three equations associated with the Y network as the following script will demonstrate.

5.1. **Producing Kirchhoff and response matrices.** After declaring some symbolic variables, the script symbolically defines Kirchhoff and response matrices for the network.

$$K = \begin{bmatrix} g_{14} & 0 & 0 & -g_{14} \\ 0 & g_{24} & 0 & -g_{24} \\ 0 & 0 & g_{34} & -g_{34} \\ -g_{14} & -g_{24} & -g_{34} & g_{14}+g_{24}+g_{34} \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} -a-b & a & b \\ a & -a-c & c \\ b & c & -b-c \end{bmatrix}$$

5.2. **Recovering conductances via MATLAB.** The script then symbolically calculates the response matrix using K. When compared to Λ we have three equations corresponding to the three elements above the main diagonal. MATLAB then solves for g_{14}, g_{24} and g_{34} in terms of a, b and c. The unique solution to this system indicates the network is recoverable, which we knew already.

Given a response matrix, we can substitute the values for a, b and c into the following equations to recover the conductances.

$$g_{14} = -\frac{ca+bc+ba}{c}$$
$$g_{24} = -\frac{ca+bc+ba}{b}$$
$$g_{34} = -\frac{ca+bc+ba}{a}$$

The script below demonstrates a use of the subs command by substituting values from the following response matrix into the above equations.

$$ExampleL = \begin{bmatrix} 5/6 & -1/3 & -1/2 \\ -1/3 & 4/3 & -1 \\ -1/2 & -1 & 3/2 \end{bmatrix}$$

(This response matrix actually came from the Y network with $g_{14} = 1$, $g_{24} = 2$ and $g_{34} = 3$.)

The results fortunately agree with this.

```
%This script uses MATLAB's symbolic manipulator to recover
%conductances for a Y network from its response matrix

clear all

%define conductances, K, L, and entries in L
syms g14 g24 g34 K L a b c real

%construct symbolic Kirchhoff matrix K
```

```
K = sym(zeros(4,4));
K(1,4) = -g14;
K(2,4) = -g24;
K(3,4) = -g34;
K = K + K';
for i = 1:4
   K(i,i) = -sum(K(i,:));
end
K

%construct symbolic response matrix L
L = sym(zeros(3,3));
L(1,2) = a;
L(1,3) = b;
L(2,3) = c;
L = L + L';
for i = 1:3
   L(i,i) = -sum(L(i,:));
end
L

%calculate a response matrix A from K
syms A real
A = sym(getL(K,3));

%now solve the three equations we get by comparing L and A
%corresponds to entries above the main diagonal
[g14 g24 g34] = solve(A(1,2)-L(1,2), A(1,3)-L(1,3), A(2,3)-L(2,3),g14,g24,g34)

%now get the conductances for a specific L
exampleK = [1 0 0 -1;0 2 0 -2;0 0 3 -3;-1 -2 -3 6]
exampleL = getL(exampleK,3);
a = exampleL(1,2);
b = exampleL(1,3);
c = exampleL(2,3);
g14 = subs(g14)
g24 = subs(g24)
g34 = subs(g34)
```

6. Some LaTeX stuff

See the appendix for the .tex file used to create this document. Most of the LaTeX commands are probably best learned by example, but here are a few of the tricky points.

One thing to look for is the different ways to work in math mode, which makes it possible to use mathematical symbols and characters and display equations properly. Numbered equations are designated by

`\begin{equation} the equation \end{equation}`

Other ways to declare math mode are with dollar signs or brackets, such as

```
 $math stuff$ or \[ more math stuff \]
```

It is possible to make macros to simplify typing in LaTeX commands. For example `\def\Rarrow{$\longrightarrow$}` means typing `\Rarrow` has the same effect as typing `$\longrightarrow$`

One can use the theorem environment by typing

```
\begin{theorem}The Theorem \end{theorem}
```

Custom environments can also be defined such as the proof environment below.

```
\newenvironment{myproof}{\textbf{Proof:} \textmd}{$\Box$}
```

The example below shows how this is displayed in LaTeX.

Theorem 6.1. *This statement is false.*

Proof: Otherwise if it were true, then it wouldn't be. $\Box$

When pasting in MATLAB code or other code, it is very useful to use the verbatim environment so that LaTeX displays the code as is, ignoring special characters such as $, & and #.

Encapsulated post script graphics can be inserted into LaTeX documents with the following code. (Figure 2 is the resulting graphic.)

```
\begin{figure}[h]
\begin{center}
\includegraphics[width=10cm]{comp.eps}
\caption{Frog licks monitor for unknown reason} \label{frog}
\end{center}
\end{figure}
```

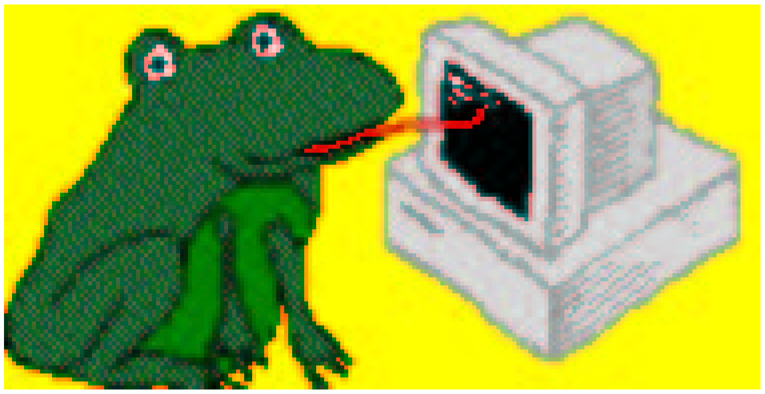

Figure 2. Frog licks monitor for unknown reason

Appendix A. The Tex file for this document (almost)

```
\documentclass[titlepage]{amsart}
\title{MATLAB and \LaTeX\  Examples}
\author{Ernie and Ryan \\ University of Washington}
\date{June 2002}

% Check if we are compiling under latex or pdflatex, and include the
% appropriate graphics package
\ifx\pdftexversion\undefined
  \usepackage[dvips]{graphicx}
\else
  \usepackage[pdftex]{graphicx}
\fi

% Need to do this if not using amsart class
%\usepackage{amsmath}
%\usepackage{amssymb}

\usepackage{verbatim} % needed to include source for this document at the end

%defining your own environment, not that you need it
\newenvironment{myproof}{\textbf{Proof:} \textmd}{$\Box$}

%user macros
\def\gam{$\gamma$}
\def\Lam{$\Lambda$}
\def\Rarrow{$\longrightarrow$}

%this command allows the margin at the bottom of the page to vary
\raggedbottom

%Apply the theorem environment to lemmas, theorems and conjectures
%\usepackage{amsthm}
\newtheorem{lemma}{Lemma}[section]
\newtheorem{theorem}{Theorem}[section]
\newtheorem{conjecture}{Conjecture}[section]

\begin{document}

\begin{abstract}
This seems as good a place as any for an abstract.  Our goal is to
present some examples that may facilitate the use of \LaTeX\ and
MATLAB.  This document itself \emph{is} the \LaTeX\ example.  The
.tex file this document was created from is in fact included in
the appendix (non-recursively of course).  A few tricky \LaTeX\
points are also emphasized.

We also present several pieces of MATLAB code that illustrate how
```

```
to write scripts, functions and even use the symbolic toolbox to
tackle an example inverse problem.
\end{abstract}

%include the given information in the title page
\maketitle



%let latex automatically create a table of contents
\tableofcontents

%put the table of contents on its own page.
\newpage

\section{Introduction}

We present four main examples:

%create a numbered list
\begin{enumerate}

\item Our first example will be to write a simple MATLAB function
that computes the Dirichlet to Neumann map given the Kirchhoff
matrix and the number of boundary nodes.  We will also present
some background information on this topic to show how it can be
written up in \LaTeX.

\item We present code for another MATLAB function that generates
prime numbers.  This is a good illustration of how flow control
works in MATLAB.

\item  There is also a MATLAB script that generates the Kirchhoff
matrix for an example network (pictured in the script itself).

\item We will also take advantage of MATLAB's symbolic manipulator
to solve the inverse problem for a simple Y network as shown in
figure 1.

\end{enumerate}

%create a figure and draw a picture
\begin{figure}
\begin{center}
\begin{picture}(100,100)(0,0)

\put(50,50){\circle*{2}}
\put(52,50){4}
```

```
\put(50,90){\circle*{5}}
\put(55,90){2}
\put(20,20){\circle*{5}}
\put(25,20){1}
\put(80,20){\circle*{5}}
\put(85,20){3}

\put (20,20){\line(1,1){30}}
\put (80,20){\line(-1,1){30}}
\put (50,50){\line(0,1){40}}

\end{picture}
\caption{Y network} \label{Y}
\end{center}
\end{figure}

Finally we will briefly illustrate how this document was produced
in \LaTeX and give some useful \LaTeX\ tricks, such as how to
include graphics.


\section{Example 1: Computing the Response Matrix}
\footnote{Background information is from \cite{book}.}
For a network with $n$ nodes, the Kirchhoff matrix $K$
is an $n \times n$ symmetric matrix formed by taking for $i \neq
j$
\[ K(i,j) = \left \{ \begin{array}{ll} -\gamma(i,j) & \mbox{if there is
an edge from i to j} \\ 0 & \mbox{if no such edge exists}
\end{array} \right\} \]
Then the diagonal entries are chosen such
that each row sums to zero. $K$ operates on the vector of
potentials $u$ to give a vector of currents $\phi$ coming out of
each node. ($Ku = \phi$)\  Naturally the current out of interior
nodes is set to zero to agree with Kirchhoff's Law. It's useful
to write the  Kirchhoff matrix in the following block form:
\[ K = \left[ \begin{array}{ll} A & B \\ B^{T} & C \end{array} \right] \]
If all interior nodes are numbered such that they appear in the
$C$ block  and all boundary nodes are in the $A$ block, then the
response matrix is just the Schur Compliment of $K$ in $C$.
%the \begin{equation} command creates a numbered equation
\begin{equation} \Lambda = A - BC^{-1}B^{T} \end{equation}
 It is shown in \cite{book}
that C is invertible because K is positive semi-definite. \Lam\
acts on the vector of boundary potentials to give the vector  of
boundary currents. This is therefore the Dirichlet to Neumann map.

\subsection{MATLAB Code for Computing \Lam}
The following assumptions are necessary:
```

```
%create a bulleted list
\begin{itemize}
\item The Kirchoff matrix K is given
\item K has the block form as above
\item We know there are n boundary nodes
\end{itemize}

The following is a function written for MATLAB that will compute
\Lam.  Note that functions and scripts are saved with a .m
extension and can be run from the prompt in MATLAB.

%display the following text as is (like <pre> in html)
\begin{verbatim}
%function L = getL(K,n)
function L = getL(K,n)
A = K(1:n,1:n);
C = K((n+1):end,(n+1):end);
B = K(1:n,(n+1):end);
L = A - B*inv(C)*B';
\end{verbatim}

\section{Example 2: A Prime Number Generator in MATLAB}
This program does what the name suggests and also illustrates the
syntax for for loops and if statements in MATLAB.

\begin{verbatim}
%function p=prime(n)
function p=prime(n)

% First prime number.
p = 2; j = 1;

while length(p) < n
    % Next number to consider.
    j = j+2;
    divisible = 0;

    for i = 1:length(p)

        if p(i) > sqrt(j)
            break;
        end

        if mod(j,p(i)) == 0
            divisible = 1;
            break;
        end
    end

    if divisible == 0
        p(length(p)+1) = j;
    end
end
\end{verbatim}

\section{Example 3: A MATLAB script}
An advantage (or disadvantage) of a script over a function is that
the variables are global.  Any previously defined variable can be
accessed by the script.  Likewise, variables created or altered by
the script can be accessed at the prompt.

The following code created a Kirchhoff matrix for a stick-figure
shaped electrical network.  It also computes the response matrix
for this network by calling the function getL, which was presented
in example 1.

\begin{verbatim}
% Create Kirchhoff matrix and response matrix for stick figure matrix.
%
%   5    1    2
%    \   |   /
%    10--6--7         1 ~ 5  Boundary nodes
%     |     |         6 ~ 10 Interior nodes
%     9-----8
%    /       \
%   4         3
%

p = prime(10);

K = zeros(10,10);

for i = 1:10
    if i <= 5
        K(i,i+5) = -p(i);
    elseif i <= 9
        K(i,i+1) = -p(i);
    else
        K(6,10) = -p(10);
    end
end

K = K + K'; for i = 1:10
    K(i,i) = -sum(K(i,:));
end
```

```
L = getL(K,5);
\end{verbatim}


\section{Example 4:  MATLAB’s symbolic manipulator}
Symbolic variables can be declared using the syms command in
MATLAB (see help file, which contains a list of all commands in
the symbolic toolbox).  It’s often useful to do symbolic
calculations to eliminate round-off error, which can present a
problem even with double precision.  Here, we demonstrate how
MATLAB can symbolically solve a small system of equations.

There are of course limitations to this.  Large systems quickly
become too memory intensive.  Symbolic calculations are naturally
much more computationally intensive.  For example, symbolically
computing determinants and inverses of matrices is not at all
practical for matrices larger than $6 \times 6$.  But MATLAB can
easily solve the system of three equations associated with the Y
network as the following script will demonstrate.

\subsection{Producing Kirchhoff and response matrices}
After declaring some symbolic variables, the script symbolically
defines Kirchhoff and response matrices for the network.
\[ K = \left[ \begin{array}{llll} g_{14} & 0 & 0 & -g_{14} \\
                                0 & g_{24} & 0 & -g_{24} \\
                                0 & 0 & g_{34} & -g_{34} \\
                                -g_{14} & -g_{24} & -g_{34} &
                                g_{14} + g_{24} + g_{34}
                                \end{array} \right] \]

\[ \Lambda = \left[ \begin{array}{lll} -a-b & a & b \\
                                a & -a-c & c \\
                                b & c & -b-c \end{array} \right]
                                \]

\subsection{Recovering conductances via MATLAB}
The script then symbolically calculates the response matrix using
K. When compared to \Lam\ we have three equations corresponding to
the three elements above the main diagonal.  MATLAB then solves
for $g_{14}$, $g_{24}$ and $g_{34}$ in terms of a, b and c.  The
unique solution to this system indicates the network is
recoverable, which we knew already.

Given a response matrix, we can substitute the values for a, b and
c into the following equations to recover the conductances.

\[ g_{14} = -\frac{ca+bc+ba}{c} \]
\[ g_{24} = -\frac{ca+bc+ba}{b} \]
\[ g_{34} = -\frac{ca+bc+ba}{a} \]

%show how to calculate this by hand maybe

The script below demonstrates a use of the subs command by
substituting values from the following response matrix into the
above equations.

\[ ExampleL = \left[ \begin{array}{lll} 5/6 & -1/3 & -1/2 \\
                                    -1/3 & 4/3 & -1 \\
                                    -1/2 & -1 & 3/2 \end{array}
                                    \right] \]

(This response matrix actually came from the Y network with
$g_{14}=1$, $g_{24}=2$ and $g_{34}=3$.)

The results fortunately agree with this.

\begin{verbatim}
%This script uses MATLAB's symbolic manipulator to recover
%conductances for a Y network from its response matrix

clear all

%define conductances, K, L, and entries in L
syms g14 g24 g34 K L a b c real

%construct symbolic Kirchhoff matrix K
K = sym(zeros(4,4));
K(1,4) = -g14;
K(2,4) = -g24;
K(3,4) = -g34;
K = K + K';
for i = 1:4
   K(i,i) = -sum(K(i,:));
end
K

%construct symbolic response matrix L
L = sym(zeros(3,3));
L(1,2) = a;
L(1,3) = b;
L(2,3) = c;
L = L + L';
for i = 1:3
   L(i,i) = -sum(L(i,:));
end
```

```
L

%calculate a response matrix A from K
syms A real
A = sym(getL(K,3));

%now solve the three equations we get by comparing L and A
%corresponds to entries above the main diagonal
[g14 g24 g34] = solve(A(1,2)-L(1,2), A(1,3)-L(1,3), A(2,3)-L(2,3),g14,g24,g34)

%now get the conductances for a specific L
exampleK = [1 0 0 -1;0 2 0 -2;0 0 3 -3;-1 -2 -3 6]
exampleL = getL(exampleK,3);
a = exampleL(1,2);
b = exampleL(1,3);
c = exampleL(2,3);
g14 = subs(g14)
g24 = subs(g24)
g34 = subs(g34)
\end{verbatim}


\section{Some \LaTeX\ stuff}
See the appendix for the .tex file used to create this document.  Most of
the \LaTeX\ commands are probably best learned by example, but here are a few
of the tricky points.

One thing to look for is the different ways to work in math mode,
which makes it possible to use mathematical symbols and characters and
display equations properly.  Numbered equations are designated by
\begin{verbatim}
\begin{equation} the equation \end{equation}
\end{verbatim}
Other ways to declare math mode are with dollar signs or brackets,
such as \begin{verbatim} $math stuff$ or \[ more math stuff \] \end{verbatim}

It is possible to make macros to simplify typing in \LaTeX\ commands.
For example \verb+\def\Rarrow{$\longrightarrow$}+ means typing
\verb+\Rarrow+ has the same effect as typing \verb+$\longrightarrow$+

One can use the theorem environment by typing
\begin{verbatim}
\begin{theorem}The Theorem \end{theorem}
\end{verbatim}

Custom environments can also be defined such as the proof
environment below.
\begin{verbatim}
```

```
\newenvironment{myproof}{\textbf{Proof:} \textmd}{$\Box$}
\end{verbatim}

The example below shows how this is displayed in \LaTeX.
\begin{theorem}
This statement is false.
\end{theorem}
\begin{myproof}
Otherwise if it were true, then it wouldn't be.
\end{myproof}

When pasting in MATLAB code or other code, it is very useful to use
the verbatim environment so that \LaTeX\ displays the code as is, ignoring
special characters such as \$, \& and \#.

Encapsulated post script graphics can be inserted into \LaTeX\
documents with the following code.  (Figure 2 is the resulting graphic.)
\begin{verbatim}
\begin{figure}[h]
\begin{center}
\includegraphics[width=10cm]{comp.eps}
\caption{Frog licks monitor for unknown reason} \label{frog}
\end{center}
\end{figure}
\end{verbatim}


\begin{figure}[h]
\begin{center}
\includegraphics[width=10cm]{comp.eps}
\caption{Frog licks monitor for unknown reason} \label{frog}
\end{center}
\end{figure}

\pagebreak

\appendix
\section{The Tex file for this document (almost)}

\verbatiminput{schur.tex} % include this document literally

%how to cite references
\begin{thebibliography}{99}
\bibitem{book}
Edward B. Curtis and James A. Morrow \emph{Inverse Problems for Electrical Networks.}
Series on applied mathematics - Vol. 13.  World Scientific, \copyright 2000.
\end{thebibliography}
```

```
\end{document}
```

References

[1] Edward B. Curtis and James A. Morrow *Inverse Problems for Electrical Networks.* Series on applied mathematics - Vol. 13. World Scientific, ©2000.