1 Introduction

This handout is an introduction to using MATLAB for material balance, energy balance, physical separation, and other process calculations.

1.1 Why MATLAB?

Material and energy balance problems frequently lead to sets of linear and nonlinear equations. Phase equilibrium relationships are usually given in terms of graphs or complicated, empirically-derived functions. This has led to the development of the “traditional” graphical analysis techniques in which the two governing processes

1. Material and energy balances;
2. Phase equilibrium relationships;

are plotted together in the solution procedure. Material and energy balances connect the stages (operating lines) and the stages themselves are assumed to be at equilibrium. MATLAB is ideally suited to plotting and solving equations, and makes subsequent analysis of the computed solution easy.

1.2 Getting started

The information provided in this document should work without any problems on the network of Unix workstations in University of Maryland (the Glue project). The programs should also work with student and other PC versions.

If you are logged into a Glue workstation, you can start a MATLAB session by typing at the Unix prompt (:)

: tap matlab sets up paths, prints messages
: matlab starts program, more msgs, graphics window appears/disappears
>> prompt from MATLAB

From this point on, we will focus on examples of MATLAB use over descriptions of the language itself. However, the most useful command is help, help help, and especially helpwin. Also, see www.mathworks.com for useful information.
## 2 Variables and basic operations

Consider the table of elements, their atomic numbers, and atomic weights, which reflects the naturally occurring isotopic compositions of the elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Atomic number</th>
<th>Atomic weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>hydrogen</td>
<td>1</td>
<td>1.0079</td>
</tr>
<tr>
<td>helium</td>
<td>2</td>
<td>4.0026</td>
</tr>
<tr>
<td>oxygen</td>
<td>8</td>
<td>15.9994</td>
</tr>
<tr>
<td>titanium</td>
<td>22</td>
<td>47.90</td>
</tr>
<tr>
<td>copper</td>
<td>29</td>
<td>63.546</td>
</tr>
<tr>
<td>silver</td>
<td>47</td>
<td>107.868</td>
</tr>
<tr>
<td>iodine</td>
<td>53</td>
<td>126.9044</td>
</tr>
</tbody>
</table>

Suppose we understand nothing about nuclear physics, but need to estimate the atomic weight of molybdenum, which has an atomic number 42. We can use simple linear interpolation by defining $x_1 = 29$, $x_2 = 47$, $y_1 = 63.546$, and $y_2 = 107.868$ and the interpolation formula

$$y_i = \frac{y_2 - y_1}{x_2 - x_1} (x_i - x_1) - y_1$$

using the MATLAB commands

```matlab
>> x1=29
x1 =
    29
>> x2=47;
>> y1=63.546; y2=107.868;
```

noting the effect of the ; character in echoing output and allowing more than one command per line. We can find out what variables are currently in use in our workspace with the who command:

```matlab
>> who
Your variables are:
x1      x2      y1      y2
```

The interpolation is carried out using

```matlab
>> xi=42;
>> yi=(y2-y1)*(xi-x1)/(x2-x1)+y1
yi =
    95.5563
```

which is not a bad interpolation, when compared to the true atomic weight of 95.94.

### 2.1 Vectors

We now demonstrate how to set the data up as column vectors, and several of the basic vector operations, such as selecting a subset of the vector elements, and determining the vector (or matrix) size

```matlab
>> x=[1; 2; 8; 22; 29; 47; 53]
x =
```

noting how the semi-colon (;) between the array elements produces a column array. The first three elements can be selected with the colon notation (:), and the number rows and arrays is determined with the size function.

>> x(1:3)
ans =
    1
    2
    8
>> size(x)
ans =
   7  1

We note that the convention is that the first direction in an array represents the number of rows, while the second denotes columns. We can define row arrays by separating elements with commas (,)

>> y=[1.0079, 4.0026, 15.9994, 47.90, 63.546, 107.868, 126.9044]
y=
  1.0079  4.0026  15.9994  47.9000  63.5460  107.8680  126.9044

which we will convert to column format using the transpose operation – the single quote (′):

>> y=y′;

What makes MATLAB so powerful and useful are its plotting capabilities. We demonstrate with a simple plot, followed by a more informative plot

>> plot(x,y,xi,yi,'go')
>> plot(x,y,x,y,'r*',xi,yi,'go')
>> grid, xlabel('Atomic No.'), ylabel('Atomic Weight'), title('The data')

2.1.1 Intrinsic functions

Matlab has numerous built in mathematical functions. For example, our one-dimensional interpolation can be done automatically by

>> interp1(x,y,xi)
ans =
    95.5563
>> yi=interp1(x,y,xi)
yi =
    95.5563
2.2 Introduction to linear algebra and matrices

Recall our alternate formula for linear interpolation

\[ y_1 = mx_1 + b \]
\[ y_2 = mx_2 + b \]

or

\[ \bar{y} = Aq \quad A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \]

\[
\begin{bmatrix}
A = \\
29 & 1 \\
47 & 1
\end{bmatrix}
\]

Having set up the square array, we now demonstrate that the matrix product of an inverse of an array with itself \((A^{-1}A)\) gives the identity array when the array is nonsingular:

\[
\begin{bmatrix}
q = \text{inv}(A) \times \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \\
2.4623 \\
-7.8617
\end{bmatrix}
\]

We use this to solve the linear equations with the following operations

\[
\begin{bmatrix}
\bar{y} = [y_1; y_2]; \\
q = \text{inv}(A) \times \bar{y}; \\
m = q(1); \\
m = 2.4623 \\
b = q(2); \\
y_i = m \times x_1 + b \\
y_i = 95.5563
\end{bmatrix}
\]

Solution of linear equations done most efficiently using the backslash operator, which encompasses Gaussian Elimination, back substitution, and least squares solution methods.

\[
\begin{bmatrix}
A = \text{inv}(A) \times \bar{y}; \\
2.4623 \\
-7.8617
\end{bmatrix}
\]

\[
\begin{bmatrix}
A = \begin{bmatrix} x, \text{ones(size(x))} \end{bmatrix} \\
\text{A} = \\
1 & 1 \\
2 & 1 \\
8 & 1 \\
22 & 1 \\
29 & 1 \\
47 & 1 \\
53 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
q = \text{A} \div \bar{y}; \\
2.4623 \\
-7.8617
\end{bmatrix}
\]
q =
  2.3774
-2.5585
>> m=q(1); b=q(2);
>> xlsqt=[1:60];
>> ylsqt=m*x_lsqt+b;
>> plot(x,y,x,y,'r*',xlsqt,ylsq)

3 Interacting with your directory

We can execute a number of unix and DOS-like commands directly:

>> ls
ans =
acetone_ethanol
bin_sep_col.m
hw2_2.m
>> ls *.m
ans =
bin_sep_col.m
hw2_2.m
>> dir
. hw2_2.m temp.m
.. hw3_1.m water_eth
acetone_ethanol hw3_3.m x.mat
>> pwd
ans =
/afs/glue.umd.edu/home/isr/a/adomaiti/home/tex/teaching/ench426/matlab2
>> cd..

and so on. This means we can save data, Matlab scripts, and Matlab functions. For example, we could have saved everything from last time in file stuff.mat with

>> save stuff
>> clear
>> who
>> load stuff

4 Control Structures

As an example of more complex operations, we consider the problem of numerically determining the roots of the nonlinear equation

\[ f(T) = 8 - 5T - 6T^4 \]

or, in other words, finding the values of \( T \) for which \( f(T) = 0 \). We use Newton’s method, which is based on the Taylor’s series expansion of \( f \)

\[ f(T) \approx f(T_0) + \frac{df(T_0)}{dT}(T - T_0) + \text{higher order terms}. \]

This expansion is solved for the \( T \) that makes \( f(T) = 0 \):

\[ T = T_0 - \frac{f(T_0)}{df(T_0)/dT} \]
This procedure can be implemented, with a graphical display of the algorithm convergence, by

\[
\begin{align*}
&\text{>> } T=-2:0.01:2; \\
&\text{>> } f=8-5*T-6*T.^4; \\
&\text{>> } \text{plot}(T,f); \text{ grid}
\end{align*}
\]

which creates a plot of the function \( f(T) \) for \(-2 \leq T \leq 2\) spanned by a set of points placed at intervals of 0.01. We can then implement the Newton iterations, and plot the results on the current plot by

\[
\begin{align*}
&\text{>> } \text{hold on} \\
&\text{>> } T0 = 0.5; \\
&\text{>> for i=1:6} \\
&\quad \text{fnew} = 8-5*T0-6*T0^4; \\
&\quad \text{plot}(T0,fnew,'ro') \\
&\quad \text{pause(2)} \\
&\quad \text{dfnew} = -5-24*T0^3; \\
&\quad \text{Tnew} = T0-fnew/dfnew \\
&\quad T0 = \text{Tnew}; \\
&\text{end} \\
&\text{Tnew} = \\
&\quad 1.1406 \\
&\text{Tnew} = \\
&\quad 0.9471 \\
&\text{Tnew} = \\
&\quad 0.8855 \\
&\text{Tnew} = \\
&\quad 0.8801
\end{align*}
\]