# The Matlab Workbook - A Supplement for Calculus, Differential Equations and Linear Algebra

Steve Chapin and Todd Young

Department of Mathematics, Ohio University, Athens, OH 45701,

January 2, 2003



*E-mail address*: chapin@math.ohiou.edu

*E-mail address*: young@math.ohiou.edu

# Acknowledgment

The second author is grateful to Mary Beth Young who originally conceived of simple computer homework as contained in the exercises in this book. We are grateful to Ohio University for support of this project through the 1804 Fund. Thanks to Tsun-ho Liu who provided technical assistance and to Kiffany Keyes to proofread the assignments. We are also grateful to our colleagues and students at Ohio University for many helpful comments. Finally, special thanks go to Larry Snyder, who helped with the preparation of many of the homework assignments contained in this book.

# Contents

CHAPTER 1

# Introduction

## A Very Brief Intro to Matlab

### A few general principles

- Type commands at the prompt and press Enter.
- Unless declared otherwise, variables are row vectors ($1 \times n$ arrays).
- The command `syms x` declares `x` to be a symbolic variable.
- MATLAB is case sensitive, i.e. $X \not\equiv x$.
- The command `clear` will clear all variables. Always `clear` before starting a new computation. The command `clear` will not clear the screen.
- Ending a command with a semicolon "`;`" suppresses the output.
- Enter all commands exactly as given in the assignments.
- `ans` indicates the output from the preceeding command. Two useful commands are `pretty(ans)` and `simple(ans)`.
- MATLAB does both symbolic and numerical calculations.
- When you make a mistake, you do not have to retype the whole command. Use $\uparrow$ and $\downarrow$ to return to a line, correct the errors and re-press Enter. (Sometimes you also need to `clear`.)
- Access `Help` by clicking Help $\rightarrow$ Help Window, or by typing `helpdesk` or `helpwin`.
- Text may be added to your work after the symbol `%`.
- Save, print and exit by clicking the File icon.
- Many advanced procedures may be accomplished using Toolboxes.
- MATLAB may be used as a programming language.
- For a more details on MATLAB and how to use it we suggest: *A Guide to* MATLAB *for Beginners and Experienced Users*, by B. Hunt, R. Lipsman, and J. Rosenberg, Cambridge Univeristy Press, New York, 2001.

**Some basic commands using a symbolic variable** (try them).

- `syms x` — This makes a symbolic variable.
- `f = x*sin(x)` — This makes `f` a symbolic function.
- `f1 = diff(f)` — `f1` is the derivative of `f`.
- `f2 = diff(f,2)` — `f2` is the second derivative of `f`.
- `F = int(f)` — `F` is the antiderivative of `f`.
- `int(f,0,pi)` — This is a definite integral.
- `limit(log(cos(x))/x^2,0)` — MATLAB uses L'Hopital's rule to find limits.
- `limit(log(x)^2/x,inf)` — Also for $\infty/\infty$.
- `ezplot(f)` — Plot a graph using the default interval.
- `ezplot(f,0,4*pi)` — Plot a graph for specified interval.
- `polyn = x^5 - x^4 - 7*x^3 + x^2 + 6*x`
  `factor(polyn)`
  `solve(polyn)` ..... This solves the equation "`polyn = 0`"
- `expr = cos(x)^5 + sin(x)^4 + 2*cos(x)^2 - 2*sin(x)^2 - cos(2*x)`
  `simple(expr)`
- `ode = 'Dx = -a*x'`
  `dsolve(ode,'x(0)=3')`

**Some basic commands using arrays.**

- `t = 0:.01:8*pi;` — Makes `t` a vector with entries from 0 to $8\pi$ in .01 increments.
  `y = t.*sin(t);` — This evaluates $t\sin(t)$ for each entry of t.
  `plot(t,y)` — This plots the pairs of points $(t(k), y(k))$ for $k = 1, 2, \ldots$.
- `x = -2:.05:2; y = x;`
  `Z = sin(x'*y); mesh(Z)` — ' means transpose.
  A figure window will appear with a graph. Click on Tools and select Rotate 3D. Point the cursor at the graph and "click and drag" to rotate the graph.
- `A = [1 2 3; 4 5 6; 7 8 10], C = [1 2; 3 4; 5 6]`
  `A*C` — multiplies the matrices.
  `b = [1 2 3]', A\b` — solves $Ax = b$ by Gaussian elimination.

CHAPTER 2

# Preliminary Exercises – Do Not Skip!!

In these two preliminary exercises we consider one of the foundational issues that is needed to use MATLAB intelligently. Namely

- **Symbolic vs. Numeric computation**

To illustrate the difference between the two, consider the equation

$$x^2 - 2 = 0.$$

Manipulating this equation symbolically, we arrive at two solutions,

$$x = \pm\sqrt{2}.$$

If we enter the square root of 2 on a calculator and obtain

(1) $$1.4121356237$$

then the calculator has performed a numerical calculation and output a numerical answer. What we will mean by symbolic calculation is to manipulate symbols in a exact fashion. By numeric, we mean to manipulate floating point numbers with a fixed number of digits, which usually implies rounding off. Sometimes very simple operations qualify as both symbolic and numeric, such as solving the equation $2x = 4$, but usually an operation can be classified as one or the other.

We need to distinguish the two types of computation because modern software packages like MATLAB are capable of doing both types and there are important differences between the two.

The chief advantage of a symbolic computation is obvious; it is perfectly accurate, assuming that it was performed correctly. Why then do anything else? One reason is that symbolic computations are not always possible. For instance in the relatively simple case of finding the roots of a polynomial. If the degree is greater than four, then it is known that in general the roots cannot be found with symbolic computations. In this case we have no choice but to approximate the roots using numerical computations.

Whereas symbolic computations are exact, numeric computations are usually approximations. This is fine for most applications, but care must always be taken to ensure that the approximation is accurate enough to suit the need. This can be tricky. For instance in the second

exercise we encounter graphing a function. When a computer graphs a function it does a finite number of numeric approximations and then "connects the dots". As you will see, this can have pitfalls.

## 1. Factoring Expressions and Solving Equations

(1) At the prompt, type the following commands and press Enter:

```
clear
syms x
expr1 = (x-1)*(x-2)*(x-3)*(x-4)*(x-5)
expr2 = expand(expr1)
factor(expr2)
solve(expr2)  ..... This solves the equation expr2 = 0
```

Explain what happened. What is the relationship between solving and factoring?

(2) Type and enter:

```
expr3 = x^4 + 3*x^3 + 3*x^2 + x + 3
factor(expr3)
solve(expr3)
double(ans)
```

Explain what happened. Explain why an exact, symbolic solution may not be as useful as an approximation.

(3) Try to solve `expr3 - 3`. Why is the answer so nice?

(4) Make `expr4` be equal to `expr1 + 1` . ( `expr4 = expr1 + 1` )
Try to factor `expr4` and to solve `expr4 = 0` .
Why do you think MATLAB produces a numerical solution (for `solve`), rather than symbolic? Hint: Is it possible in this case to give a symbolic solution? Why?

(5) Prepare a brief (less than 1 page) written report answering all the questions. Use complete sentences and standard mathematical notation. **Do not get a printout.**

The user learns basic algebraic manipulation commands and is led to consider the difference between numerical and symbolic solving techniques. The user must confront the foundational fact that a symbolic solution is not always possible.

## 2. Defining, Evaluating and Plotting Functions

(1) At the prompt type: `syms x` and then press Enter.
Now type `f = sin(x)` and then press Enter.

(2) Type (at the prompt and then press Enter):

```
subs(f, 2)
subs(f, '2')
double(ans)
```

Which of the above answers are numerical and which are symbolic? (You may want to type: `help subs` and `help double` for explanations)

(3) Enter: `ezplot(f)`

(4) Following the example above, define and plot the function g(x) = exp(x) by typing:

```
syms x
g = exp(x)
ezplot(g)
```

Then adjust the domain in the plot by typing: `ezplot(g, -2, 2)`

(5) Enter: `ezplot(x^2)`

(6) Plot function `sqrt(x^2-.00001)` by typing: `ezplot(sqrt(x^2 - .00001))`.
Plot the function `x^7 - x` by typing: `ezplot(x^7 - x)`.
Because of the domain chosen by the computer, important features of the graphs are missing. What are they? Try adjusting the domains until these features are shown.

(7) Plot the function `sin(x^5)` by typing: `ezplot(sin(x^5))`. A computer plots a function by locating a finite number of points and "connecting the dots". How does this go wrong for `sin(x^5)`?

(8) Prepare a brief (< 2 page) written report describing what happened, answering all the questions and sketching the plots. Use complete sentences and standard mathematical notation. **Do not get a printout.**

These exercises introduce basic commands for defining and plotting functions. They consider the difference between numerical and symbolic evaluation of a function and the processes by which the software makes plots. They address issues of scale and the effects of rapid oscillation on plotting.

CHAPTER 3

# Differential Calculus

One of the great successes of symbolic computation programs is that the derivative of almost any function can now be produced symbolically at incredible speed and with absolute reliability. It is when one begins to use the derivative in applications that problems begin. For instance, when one wants to determine an extreme value of a functions, then one usually must try to solve the equation

$$f'(x) = 0.$$

As seen in the previous chapter, solving equations can quickly become a nontrivial affair.

We begin this chapter with some exercises involving limits and the definition of derivative. The main purpose is to assist students in grasping these topics that have proven to be difficult for many. Along the way we review issues of symbolic vs. numeric computation and difficulties in reliability of computer generated plots. We also explore Newton's Method, which is usually the student's first encounter with an actual numerical algorithm. We look at issues of speed of convergence as well as potential pitfalls of the method.

# Matlab Commands for Differential Calculus

## 1. Limits

(1) Try the following commands (at the prompt and press Enter):

(a) `x = sym('x')`

(b) `f = x^2`

(c) `limit(f, 2)`

(d) `limit(f, inf)`

(e) `limit(1/x, inf)`

(f) `limit(log(abs(x)), 0)`

(g) `limit(1/x, 0)`

(h) Explain what happened in each example, that is, why did it give the answer it did.

(2) Use MATLAB to find the limits of the following functions at the given points:

(a) `sqrt(x) at x = 0` (Type as: `limit(sqrt(x), 0)`)

(b) `sqrt(x^2 - .00001) at x = 0` (Type as: `limit(sqrt(x^2 - .00001), 0)`)

(c) `sqrt(x) at x = -1` (Type as: `limit(sqrt(x) , -1)`)

(d) `sin(x) at x = inf` (Type as: `limit(sin(x), inf)`)

(e) `sin(1/x) at x = 0` (Type as: `limit(sin(1/x), 0)`)

(f) Explain what happened in each example.

(3) Prepare a brief (less than 1 page) written report answering all the questions. Use complete sentences and standard mathematical notation. **Do not get a printout.**

The user encounters usual limits, limits at infinity and infinite limits, complex limits and oscillatory functions.

## 2. Limits and Derivatives

(1) Try the following commands:

```
syms  x   h
f = x^3 + x^2 + x + 1
m = (subs(f,x+h)-f)/h
f1 = limit(m, h, 0)
```

Explain what happened.

(2) Try the following sequence:

```
syms  x   h
f = exp(sin(x))
m = (subs(f,x+h)-f)/h
f1 = limit(m, h, 0)
subs(f1, pi)
X = -10:.05:10; ..... Makes an array of x values.
F = subs(f, X); ..... Makes an array of f(x) values.
F1 = subs(f1, X);
plot(X, F, 'b', X, F1, 'r')
```

Explain what happened.

(3) Now repeat the steps above for the function:

$$f(x) = (x-1)^2\sqrt{x} \qquad (\texttt{f = (x-1)\^{}2*sqrt(x)}).$$

Is the function defined for all real numbers? What about the derivative? How is the graph misleading?

(4) Next repeat this procedure for the function $f(x) = (x-1)^2 x^{1/3}$. Are the function and its derivative defined for all real numbers? How is this graph misleading?

(5) Use `ezplot(f)` and `ezplot(f1)` to get another picture for $f$ and $f'$ from (4). In what ways are these graphs misleading?

(6) Prepare a brief (less than 1 page) written report answering all the questions. Use complete sentences and standard mathematical notation. **Do not get a printout.**

This assignment is intended to reinforce the user's understanding of the definition of the derivative. They should think about the domains of a function and its derivative.

## 3. Derivatives

(1) Try the following commands:

(a) `syms x`

(b) `f = x^2`

(c) `f1 = diff(f)`

(d) `X = -3:.05:3;` ..... Makes `X` into an array with entries from `-3` to `3`

(e) `F = subs(f, X);`

(f) `F1 = subs(f1, X);`

(g) `plot(X, F, 'b', X, F1, 'r')`

(h) Explain exactly what happened.

(2) Repeat the above procedure for the function

$$g(x) = \frac{x^5 + x^3 + 2}{8x + 1}$$ `(Input as: g = (x^5 + x^3 + 2) / (8*x + 1)).`

(3) Use the command `ezplot(g1, [0 3])` and then change the interval until you can accurately guess a solution of g'(x) = 0. Then try:

(a) Enter `solve(g1)` and describe the results. Which part of the output is relevant? Did the computer find this output symbolically or numerically?

(b) What is the percentage error of your guess.

(4) Prepare a brief (less than 1 page) written report answering all the questions. Use complete sentences and standard mathematical notation. **Do not get a printout.**

The user must consider the derivative as a function, and they must consider issues of scale in plotting functions with asymptotes.

## 4. Newton's Method

(1) (a) Try the following commands (at the prompt and press Enter):

```
syms x
format long  ..... Sets displayed digits to 15.
f = x^3 - 3*x^2 + 1
f1 = simplify(diff(f))
g = simplify(x - f/f1)
p = .1
p = subs(g, p)
```

(b) Repeat the command `p = subs(g, p)` until `p` stops changing. (Use the up-arrow key to recall the command instead of typing it.)

(c) Assuming the final value is correct, how many steps did it take to get 7 decimal places of accuracy? How many steps for 14 decimal places?

(2) (a) Type `p = .5` and repeat `p = subs(g, p)` until `p` stops changing. To what do the approximations converge this time?

(b) Repeat, but start with `p = 3.0`.

(c) Why can Newton's method give three different answers for three different starting points? (Hint: Use `ezplot(f)` to look at $f(x)$.)

(3) Set `p = .11065934333376` and repeat `p = subs(g, p)` until it converges. How many iterations does it take this time?

(4) Repeat the process in (1), starting with `p = .1` for the function

$$f(x) = \frac{(x - 3/4)^{1/3}}{x^{1/3}} \qquad \left( \texttt{f = ((x-3/4)^(1/3))/(x^(1/3))} \right)$$

Write down the first 20 iterations. Do they seem to be converging to anything? Plot them on the interval $[0, 1]$.

Does $f(x) = 0$ have a solution on $[0, 1]$? Try that point as the initial guess and see what happens.

Next, try starting with `p = 0.0`. What is the value of $f$ at 0.0?

(5) Can one always rely on Newton's method? What are some things to be careful about?

(6) Prepare a brief (less than 1 page) written report answering all the questions. Use complete sentences and standard mathematical notation. **Do not get a printout.**

The user observes that Newton's method converges very fast for the certain functions and certain starting points. The convergence can be slow for other starting points and the final answer can depend on the starting point. Further, some functions lead to Newton's method iterations which are actually chaotic (random-like).

## 5. Exponentials vs. Powers

(1) Enter the following sequence commands:

**Important note:** Do not omit the semicolons! Also, do not omit the . before the ^ !

(a) `x1 = -1.15:0.01:1.15;` ..(This makes `x1` a vector with entries from $-1.15$ to 1.15 in .01 increments.)

(b) `x2 = -1.39:0.01:1.39;`

(c) `y1 = x1.^10;` .....(This evaluates `x1`$^{10}$ for each entry of `x1`.)

(d) `y2 = exp(x2);`

(e) `plot(x1, y1, 'b', x2, y2, 'r')`

These plots of $y = x^{10}$ and $y = e^x$ *suggest* that the equation $x^{10} = e^x$ has two solutions — one positive and one negative. Approximate these two solutions (to three decimal places) by "zooming". (To "zoom in" click on the button that looks like a magnifying glass with a plus sign, and then click on the graph. To "zoom out" select the magnifying glass with the minus sign.)

(2) Explain why there must be another positive solution of $x^{10} = e^x$ larger than the one that you found in #1.

By changing the beginning and ending values of `x1` and `x2` (you may leave the increments the same) and plotting as above, determine an interval that reveals this larger solution. (Note. You can use the up-arrow key to do this, but you must reevaluate `y1` and/or `y2` each time you change `x1` and/or `x2`.)

Approximate this solution (to two decimal places) by "zooming".

(3) Explain why it may be necessary to use several different domain intervals when studying computer plots.

(4) On a separate piece of paper, prepare a brief written report giving explanations where requested and answering all the questions. Include all of the approximate solutions. Use complete sentences and use standard mathematical notation. Do **not** hand in a printout.

This assignment reinforces the fact that the exponential function, exp(x), will eventually exceed any power of x. It also illustrates the importance of scale when considering computer plots.

CHAPTER 4

# Integral Calculus and Series

Since integration is just the inverse process of differentiation, one might guess that the techniques for computing them would be similar. In practice this is not the case. Whereas the derivative of any elementary function can be found symbolically, most elementary functions cannot be integrated symbolically. In fact, it is known that the antiderivatives of most elementary functions cannot be expressed in terms of elementary functions. In applications this often leads to the need to calculate definite integrals numerically. As with any other numeric calculation, accuracy becomes an issue. One of the main problems in obtaining accurate integrals numerically is similar to the problem of obtaining accurate graphs: the computer only handles a finite amount of discrete information. There might not be enough of this information to accurately determine what the function is doing.

In our exercises on series the main point we want to make is that speed of convergence matters. One of the primary uses of series is to approximate functions. The series is an infinite symbolic expression, but for approximations only a finite number of terms can be used. The accuracy of course depends on the number of terms used, and one must use as many terms as necessary to achieve the desired accuracy. The more terms needed for accuracy, the slower the computation will be. If the number of terms need is too large, then the method is not practical. Thus, the faster a series converges the better.

# Matlab Commands for Integral Calculus

## 1. Indefinite Integrals

(1) Enter the following sequence commands:
   (a) `syms x`
   (b) `int(x^2)`
   (c) `diff(ans)`
   (d) Explain exactly what happened.

(2) Repeat steps (b)-(c) for the function:

$$\frac{x}{(x-1)(x+2)(x^2-1)(x+1)}$$

(Typing: `int(x / ((x-1)(x+2)(x^2-1)(x+1)))` for command(b))
Then enter the command: `simplify(ans)`

(3) Repeat the above sequence for the following functions:
   (a) $1/(1+3x+x^5)$ (Typing: `int(1/(1 + 3*x + x^5))` for command (b))
   (b) $\sin(\sin(x^3))$ (Typing: `int(sin(sin(x^3)))` for command (b))
   (c) $(1+x^6)^{3/4}$ (Typing: `int((1 + x^6)^(3/4))` for command (b))
   Why do you think that MATLAB was not able to find antiderivatives for some of these functions? Why was it successful for the rational function in #2, but not successful for the rather simple functions in #3?

(4) Prepare a brief ($< 1$ page) written report answering all the questions. Do **not** get a printout. Use complete sentences and standard mathematical notation.

This assignment introduces the command for indefinite integrals. MATLAB is not able to find an integral for some functions. It is a fundamental fact that not all functions have an antiderivative in terms of elementary functions. The difference between this concept and the concept of integrability should be considered.

## 2. Definite Integrals and Numerical Approximations

(1) Enter the following sequence commands:

```
format long
syms x
int(x^2, 0, pi)  ..... Computes symbolically.
double(ans)  ..... Converts to a numerical value.
quad('x.^2', 0, pi)  ..... Computes numerically using Simpson's method.
```

(2) Use both `int` and `quad` to integrate the following function on the interval $[0, 1/2]$ (When you use `quad`, you must follow the example and enclose the function in ' ', and the operations: "`* / ^`" must be typed as: "`.* ./ .^ `". This is because `quad` treats $x$ as a matrix.):

$$\frac{x}{(x-1)(x+2)(x^2-1)(x+1)}$$

Which answer do you trust?

(3) Use both `int` and `quad` to integrate the following functions on the interval [0, 1]:

(a) $1/(1+3x+x^5)$ . Type as:

```
int(1/(1 + 3*x + x^5), 0, 1)
double(ans)
quad('1./(1 + 3.*x + x.^5)', 0, 1)
```

(b) $\sin(x^3)$ . Type as:

```
int(sin(x^3), 0, 1)
double(ans)
quad('sin(x.^3)', 0, 1)
```

(c) $(1+x^6)^{3/4}$ . Type as:

```
int((1 + x^6)^(3/4), 0, 1)
double(ans)
quad('(1 + x.^6).^(3./4)', 0, 1)
```

How many decimal places of accuracy does it seem like you get in each? Which way is more accurate for these integrals?

(4) Prepare a brief (< 1 page) written report describing what happened and answering the questions. Use complete sentences and standard mathematical notation. Do **not** get a printout.

The user should observe that even for some relatively simple integrands, the integrals cannot be found in terms of elementary functions. However the computer can obtain a numerical answer using piecewise polynomial approximations to the integrand (Simpson's method).

## 3. Numerical Integration

(1) Enter the following sequence commands:

```
maple('with(student)') ..... This adds the Maple 'student package'.
syms x
maple('rightsum(3*x^2, x=0..2, 10)')
maple('evalf(%)')
```

Here `10` is the number of intervals used. How close is the right sum to the exact value of the integral?

(2) Next try:

(a)
```
maple('leftsum(3*x^2, x=0..2, 10)')
maple('evalf(%)')
```

(b)
```
maple('trapezoid(3*x^2, x=0..2, 10)')
maple('evalf(%)')
```

(c)
```
maple('middlesum(3*x^2, x=0..2, 10)')
maple('evalf(%)')
```

(d)
```
maple('simpson(3*x^2, x=0..2, 10)')
maple('evalf(%)')
```

What are the errors in each of the above? (Compare it with the exact value that you can calculate by hand.) Explain why the approximation gets better as we go down the list.

(3) Repeat the above sequence but change the number of intervals used from `10` to `1000` in the command.

(4) Use `trapezoid` and `simpson` with 1000 intervals on each of the following functions:

(a) $\sin(\sin(x))$ with `x=0..2`, by typing the following commands:

```
maple('trapezoid(sin(sin(x)), x=0..2, 1000)')
maple('evalf(%)')
maple('simpson(sin(sin(x)), x=0..2, 1000)')
maple('evalf(%)')
```

(b) $x^5 \cos(x^6)$ with `x=0..5*Pi^(1/6)` ..... Maple uses `Pi`, MATLAB `pi` by typing the following commands:

```
maple('trapezoid((x^5)*cos(x^6)), x=0..5*Pi^(1/6), 1000)')
maple('evalf(%)')
maple('simpson((x^5)*cos(x^6)), x=0..5*Pi^(1/6), 1000)')
maple('evalf(%)')
```

(5) Use the MATLAB command `int(f(x), a, b)` to evaluate the integral for each of the two functions in #4. For example, for the first one use `format long` followed by `int(sin(sin(x)), 0, 2)` and `double(ans)`. How close were the approximations in 4. to the approximations obtained here? Was 1000 big enough for these integrals?

(6) Using complete sentences and standard mathematical notation, prepare a brief ($< 1$ page) written report answering all the questions. Do **not** get a printout.

The user compares some basic numerical schemes and considers their accuracy. The effect of partition size and the problem of rapid oscillations are also considered.

## 4. Monte Carlo Integration

(1) Enter the following sequence commands:
   (a) `n = 10`
   (b) `total=0;  for i=1:n, total=total+rand^3; end; avg=total/n`
   (c) Use the `up-arrow key` to recall this line and then press enter again.
   (d) Obtain 10 estimates this way and record the values you get along with the absolute error of each estimate. You can have MATLAB calculate the absolute error for you conveniently by including at the end of line of `#1(b): error = abs(.25 - avg)`.
   (e) Explain why this is an approximation of $\int_0^1 x^3\,dx$.

(2) Enter the command `n = 100` and use the `up-arrow key` to recall the line in #1(b) again. Press the enter key to execute this line. Obtain and record 10 estimates this way along with the absolute errors.

(3) Repeat this process using `n = 1000`, `n = 10000`, and `n = 100000`.

(4) Make a chart showing the relationship between the sample size `n` and the arithmetic mean of the absolute errors of the estimates with sample size `n`. Use the data to write a formula which approximately describes the relationship (Try $E_n \approx Kn^{-r}$ and use logarithms to determine $K$ and $r$).

(5) Compare the accuracy of this method with the Trapezoid and Simpson's methods of numerical integration. (For the Trapezoid rule $r = 2$ and for Simpson's rule $r = 4$.)

(6) Prepare a brief written report answering all the questions. Use complete sentences and standard mathematical notation. Do **not** get a printout.

Many people are surprised that this technique for numerically approximating an integral is used sometimes in practice because it is efficient in higher dimensions.

## 5. Hyperbolic Functions and the Gateway Arch

The Gateway Arch in St. Louis has the shape of an inverted catenary. Rising 630 feet at its center and stretching 630 feet across its base, the arch's shape can be described by

$$y = -127.7\cosh(x/127.7) + 757.7 for - 315 \leq x \leq 315.$$

(1) Enter the following sequence commands:
   (a) `syms x`
   (b) `f = -127.7*cosh(x/127.7) + 757.7`
   (c) `ezplot(f, -315, 315)`

   Does the graph look like an arch?

(2) To compute the area beneath the Gateway Arch enter the following sequence of commands:
   (a) `int(f, -315, 315)`
   (b) `double(ans)`

(3) To compute the length of the Gateway Arch enter the following sequence of commands:
   (a) `int(sqrt(1+diff(f)^2), -315, 315)`
   (b) `double(ans)`

   Are you surprised that the symbolic answer is expressed in terms of exponential functions rather than hyperbolic functions? Why or why not?

(4) On a separate piece of paper, prepare a brief written report describing what happened and answering all the questions. Include the results of all of the computations. **Use complete sentences and use standard mathematical notation. Do not get a printout.**

Students may have little or no exposure to hyperbolic functions in their calculus courses. This assignment gives a real-life application of hyperbolic functions.

## 6. Improper Integrals

(1) Enter the following sequence of commands:

```
syms x
int(1/sqrt(x^6+1), 0, inf)  .....Calculates symbolically.
double(ans)  .....Converts to a numeric format.
quadl('1./sqrt(x.^6+1)', 0, inf)  .....Calculates numerically.
```

(2) Use the commands above to evaluate the following integrals (you will encounter error messages in some of them):

(a) $\int_0^\infty \frac{1}{x^{2/3}}dx$ .....(Use `1/x^(2/3)` .)

(b) $\int_1^\infty \frac{1}{x+1}dx$

(c) $\int_1^\infty \frac{\ln x}{x^2}$ .....(Use `log` for natural logarithm.)

(d) $\int_0^\infty \sin^2(x)dx.$

(3) Try to use MATLAB to evaluate the following functions using commands in #1:

(a) $\int_{-1}^1 \frac{1}{x^2}dx$

(b) $\int_0^1 \frac{1}{\sqrt{x}}dx.$

(4) What are some problems with calculating improper integrals numerically?

(5) Try the following:

```
int(1/x^5, 1, inf)
int(sin(x^3)/x^5, 1, inf) .
```

Comparing the integrands of these two integrals, should the second one converge? What might be causing MATLAB to make this mistake?

(6) Prepare a brief (< 1 page) written report describing what happened and answering all the questions. Use complete sentences and standard mathematical notation. **Do not get a printout.**

This exercise explores improper integrals both symbolically and numerically. Evaluating improper integrals symbolically is precarious because it is hard for the computer to handle the symbol $\infty$ correctly. Evaluating numerically is difficult because one cannot actually compute all the way to $\infty$, one must stop at some finite place.

## 7. Summation of Series

(1) Enter the commands:

```
syms x k
format long
```

(2) Enter: `symsum(.5^k,0,inf)`

What kind of series is this? Is the result of the computation an approximation or is it exact? Was it done numerically or symbolically?

(3) Enter: `symsum(.5^k, 0, 10)` followed by `double(ans)`.

Here $n = 10$. Increase $n$ until 5 decimal places of accuracy are reached.

(4) Enter: `symsum(.99^k, 0, inf)` and `symsum(.99^k, 0, 10)`

Again increase $n$ until 5 decimal places of accuracy are reached. Compare this with the value of $n$ in the previous computation, i.e., what is the difference and what causes it?

(5) Try to repeat the process used in #2 and #3 for the series $\sum_{k=1}^{\infty} \frac{1}{k^{1.1}}$. Using an integral estimate (by hand), how many terms are needed for 5 decimal places accuracy? What makes the convergence so slow?

(6) Try to guess what the results of the command: `symsum(x^k/sym('k!'),k,0,inf)` will be, then enter it. Think about how amazing the program is.

(7) Prepare a brief (< 1 page) written report describing what happened and answering the questions. Use complete sentences and standard mathematical notation. Do **not** get a printout.

Rate of convergence is considered. For series to be useful for calculations which are used often, convergence must be relatively fast.

## 8. Taylor Series

MATLAB has an interactive Taylor series calculator called `taylortool`. It plots $f$ and the $N$-th degree Taylor polynomial on an interval. After `taylortool` is started, we can change $f$, $N$, the interval, or the point $a$.

(1) (a) Enter the command: `taylortool('sin(x)')`

(b) In the taylortool window, change $N$ to be 3. You can change the degree $N$ using the buttons `>>` or `<<`. Also you can just enter the value for $N$ in the box for $N$.

(c) For what domain does the Taylor polynomial appear to be a good approximation of the function?

(d) Now use the button `>>` to increase $N$ until the approximation appears to be accurate on the whole interval.

(e) For the degree $N$ above, use Taylor's Formula (by hand) to find an upper bound on the error of the approximation.

(2) In the `taylortool` window, change the function to $f(x) = e^x$ (use `exp(x)`), the interval to [-3, 3] and N to 3. Repeat the process above.

(3) Repeat the above process for $\sin(e^x)$ on the interval $[0, 3]$. What problems do you encounter. What do you think causes this? Does $\sin(e^x)$ equal its Taylor series? For roughly what range of $x$ and $N$ would $T_N(x)$ be a practical approximation tool? What might be a more reasonable strategy for approximating $\sin(e^x)$?

(4) Prepare a brief ($< 1$ page) written report describing what happened and answering the questions. Use complete sentences and standard mathematical notation. Do **not** get a printout.

The taylortool can help us gain some appreciation for the loss of accuracy of the Taylor approximation as $x$ varies farther from the approximation point $a$. We also encounter the difficulty of approximating a function that oscillates. Although a Taylor Series does actually equal a certain function, computers can only do polynomial operations. So for instance, the sine function on calculators or computers **must** be approximated using polynomial computations and knowing the accuracy is important.

CHAPTER 5

# Multiple Variable Calculus

One of the good features of MATLAB and similar programs is that they can be used to visualize two and three dimensional mathematical objects. However, just as computer plots of single variable functions can be misleading or even wrong, so too can plots of higher dimensional object have problems. The main difficulty is the same as for single variable plots, i.e. the computer can only represent a finite amount of information, but a function contains infinite information. A problem which is unique to three dimensional object, such as the graph of a function of two variables is that one must consider it from various angles. MATLAB allows the user to rotate an object and one should always take advantage of this feature.

Applications of multiple variable calculus, such as the method of Lagrange multipliers, often lead to the need to solve a system of equations with multiple variables. Unless the system is linear, solving the equations can be difficult and often requires numerical method, such as a higher dimensional version of Newton's method.

# Matlab Commands for Multiple Variable Calculus

## 1. Plotting Curves

(1) Enter the commands:

```
ezplot('x^2 + y^2 = 9')
ezplot('x^2 + x*y + y^2 = 9')
ezplot('x^2 - y^2 = 9')
ezplot('x^2 - y = 9')
```

What is the geometric object in each of the above?

(2) Use `ezplot` to plot the equations:

$\sin^2 x + sin^2 y - log(xy) = 0$ (Use `ezplot('sin^2(x)+sin^2(y)-log(x*y)=0')`)

$x^5 + y^5 + xy = 0$ (Use `ezplot('x^5 + y^5 + x*y = 0')`)

x + y = 100 (Use `ezplot('x + y = 100')`)

Are these plots produced numerically or symbolically? Sketch the graphs for your report and discuss any problems encountered.

(3) Try the following commands:

```
t = 0:pi/50:10*pi;
plot3(sin(t), cos(t), t)
```

Sketch the graph for your report.

(4) Try the following commands:

```
t = 0:1:10*pi;
plot3(sin(t.^2), cos(t.^3), t)
```

Eplain why the plot looks as it does.

(5) Try the following commands:

```
t = 0:pi/50:10*pi;
plot3(sin(t.^2), cos(t.^3), t)
```

Explain why the plot looks as it does.

(6) Using complete sentences and standard mathematical notation, prepare a brief written report. Do **not** get a printout.

The user plots curves given both by equations and by parametric functions. The user also encounters difficulties with under-sampling and with choice of domain.

## 2. Polar Coordinates

(1) Enter the following sequence commands:

```
syms t
r = cos(4*t) ..... Use t in place of θ.
ezplot(r*cos(t), r*sin(t), [0,2*pi])
```

Explain exactly what happened.

(2) Plot the polar equations r = cos(nθ) and r = sin(nθ) for several positive integers n. (Use the ↑ key.) Find a formula for the number of loops.

(3) Plot the polar equation r = sin(pθ/q) for various integers p and q, satisfying p > q > 0. Write p/q in lowest terms and plot over the interval [0, 2πq]. Find a formula for the number of loops.

(4) Plot the polar equation r = sin($\sqrt{2}\theta$) on the interval [0, 100π]. Explain the resulting plot.

(5) Plot the polar equation $r = e^{cos\theta} - 2cos4\theta + sin^5(\theta/12)$ for $0 \leq \theta \leq 24\pi$. (This curve was discovered by Temple H. Fay.)

(6) On a separate piece of paper, prepare a brief written report describing what happened and answering all the questions. Use complete sentences and use standard mathematical notation. Hand-in sketches of graphs or computer plots as directed by your instructor.

Polar equations can be plotted by transforming them into parametric equations.

### 3. Defining and Plotting a Function of Two Variables

(1) Enter the following commands:

```
syms x y
ezmesh(sin(x)*cos(y),[0,10,0,10])
```

(2) Click on Tools and then click Rotate 3D. Point at the graph, press the left mouse button and hold it down, and then move the pointer around slowly. The graph should rotate. Move it until you find the best picture possible (in your opinion).

(3) Use ezmesh to plot the function $f(x, y) = x^2 - y^2$. First use the same domain as in #1 and then use [-2, 2, -2, 2]. Which is a better picture?

(4) Plot the function $f(x, y) = \sin x^5 * \cos y$ using the default domain, i.e. omit it from the command. What are the problems with the resulting graph?

(5) Carefully sketch by hand the "best" graphs for the two functions in #1 and #3. Be sure to clearly label axes.

(6) Explain briefly why the plot of the function in #4 does not represent the true graph of the function. Use complete sentences and standard mathematical notation.

The goal of this project is to familiarize the user with the higher dimensional plotting capabilities of the program and to introduce them to the notion that views and domains must be adjusted to obtain a useful picture. The problem of plotting rapid oscillations is reviewed.

## 4. Contour Plots

(1) Enter the following commands:

```
[X, Y] = meshgrid(-1:.2:1);
Z = X.^2 - Y.^2;
contour(Z)
```

Notice the labelling of the axes. In order to fix this enter instead: `contour(X, Y, Z)`

(2) Also try the following variations and report what happens:

```
contourf(X, Y, Z)
contour(X, Y, Z, 10)
contour(X, Y, Z, 20)
contourf(X, Y, Z, 20)
```

What is the problem with the last couple of plots?

(3) Now try the following alternative way to make contour plots:

```
syms x y
ezcontour(x^2 - y^2)
```

(4) Try both commands above to plot the level curves of $z = \sqrt{1 - x^2 - y^2}$. Notice the squiggles in the curves near the edge. Should those be there? Can you find a way to improve this?

(5) Write a brief report, using complete sentences and standard mathematical notation.

The goal of this project is to familiarize the user with the contour plot capabilities of the program. They should notice that the methods the program uses to produce the plots have limitations.

## 5. Partial Derivatives

(1) Enter the following commands:

```
syms x y
f = x*y*(x^2-y^2)/(x^2+y^2)
fx = diff(f ,x)
fx = simplify(fx)
subs(fx, {x, y}, {0, y})
```
..... This is $f_x(0, y)$.

(2) Define $f(0,0) = 0$ and compute, by hand,

$$f_x(0,0) = \lim_{h\to 0} \frac{f(h,0) - f(0,0)}{h}.$$

Why is it necessary to use the definition to compute $f_x(0,0)$?

(3) Try: `fy = simplify(diff(f,y))`

`subs(fy, {x, y}, {x, 0})` ..... This is $f_y(x, 0)$.

Then, compute $f_y(0,0)$ by hand.

(4) Compute, by hand,

$$f_{xy}(0,0) = (f_x)_y(0,0) = \lim_{k\to 0} \frac{f_x(0,k) - f_x(0,0)}{k} \quad \text{and} \quad f_{yx}(0,0) = \lim_{h\to 0} \frac{f_y(h,0) - f_y(0,0)}{h}.$$

What do you notice about $f_{xy}(0,0)$ and $f_{yx}(0,0)$?

(5) Try: `fxy = diff(fx, y)`

```
fxy = simplify(fxy)
ezmesh(fxy)
```

What do you notice about the graph of $f_{xy}$?

(6) Either obtain a printout of the graph, or, carefully sketch it by hand, making sure to clearly label axes.

(7) Using complete sentences and standard mathematical notation, write a brief report, showing your hand calculations and answering all the questions.

The user is reminded of the definition of derivative and encounters a situation where it must be used. The user also encounters a situation where second derivatives are not continuous and $f_{xy} \neq f_{yx}$.

## 6. Gradients

(1) Enter the following commands:

```
[X, Y] = meshgrid(-2:.2:2);
Z = X.*exp(-X.^2 - Y.^2);
mesh(X, Y, Z)
```

Rotate this plot into various positions until you fully understand the shape.

(2) Next enter the following:

```
[DX, DY] = gradient(Z);
contour(X,Y,Z)
hold on
quiver(X,Y,DX,DY)
hold off
```

(3) Repeat the above steps for the functions: $f(x, y) = \exp(-x^2 - y^2)$ and $f(x, y) = x + y$.

(4) Write a brief report, using complete sentences and standard mathematical notation.

The goal of this project is to familiarize the user with the vector field plot capabilities of the program and aid in the understanding of the meaning of the gradient.

## 7. Lagrange Multipliers

(1) To find the points on the ellipse $4x^2 + 9y^2 = 36$ that are nearest to and farthest from the point $(1, 1)$, using the method of Lagrange multipliers, one needs to solve the system of equations

$$2(x-1) - 8\lambda x = 0$$
$$2(y-1) - 18\lambda y = 0$$
$$4x^2 + 9y^2 - 36 = 0$$

Carefully **derive** this system by hand. Do **NOT** try to **solve** the system by hand. Instead, solve the system using the commands:

- `syms L x y` ..... (Note that we use "L" instead of "$\lambda$".)
- `[L,x,y] = solve(2*(x-1)-8*L*x, 2*(y-1)-18*L*y, 4*x^2+9*y^2-36)`
- `double([L,x,y])` (Elements in square brackets must be in alphabetical order.)

Explain what happened. What is the nearest point? What is the farthest point? Give solutions to four decimal places.

(2) Adapt the procedure in #1 to find the points on the ellipsoid

$$64x^2 + 144y^2 + 36z^2 = 576$$

that are nearest to and farthest from the point $(1, 1, 1)$. Write down the system you are solving and answer the questions above for this example.

(3) What are your observations about symbolic versus numerical computations from #1 and #2?

(4) Using complete sentences and standard mathematical notation, write a brief report (1 page only), showing your hand calculations and answering all the questions.

The system of equations resulting from relatively straightforward Lagrange multiplier problems can be very difficult, if not impossible, to solve in closed form. In this exercise MATLAB is used solve such systems. Students are asked to compare symbolic versus numerical solutions.

## 8. Double Integrals

(1) Enter the following commands:

```
format long
```
.....Sets the number of digits displayed to 15.

```
f = inline('x*y^2')
```
.....Defines a function.

```
f(2, 3)
dblquad(f, 0, 1, 0, 1)
```

Calculate this double integral by hand to confirm the answer. To get an explaination of the command, you may want to enter: `help dblquad`

(2) Next try to use `dblquad` to integrate the following functions on the same domain as above:

```
f = inline('x^2*y^2')
f = inline('x.^*y^2')
f = inline('x.^*y.^2')
```

Calculate this double integral by hand to determine which of the outputs is correct. You should conclude the following: for the `dblquad` command to work properly, `x` must be a vector and `y` must be a scalar.

(3) The following is a trick that can be used for double integrals on regions that are not square:

```
f = inline('(x.^2*y^2).*(x+y <= 1)')
dblquad(f, 0, 1, 0, 1)
```

This should calculate the integral of $x^2y^2$ on the triangle with corners at (0, 0), (1, 0), and (0,1). Calculate this integral by hand to comfirm the calculation above.

(4) Write a brief report including your hand calculations. Use complete sentences and standard mathematical notation.

The goal of this project is to familiarize the user with the double integration capabilities of the program. Just as hand calculations of multiple integrals are tricky, so are computer calculations.

CHAPTER 6

# Differential Equations

MATLAB can be used both symbolically and numerically to study differential equations. For instance, if the equation is separable, then one can simply use MATLAB's symbolic integration command, '`int`', to find the antiderivatives of both sides. MATLAB also has a symbolic command '`dsolve`', that tries to solve initial value problems symbolically, even when the equation, or the initial conditions involve symbols. For solving numerically, MATLAB provides several algorithms, including '`ode45`', which employs the Runga-Kutta45 method, the most commonly used algorithm in applications.

When using '`ode45`' for second or higher order equations, the equation must first be put into the form of a system of equations. For instance consider the equation

$$x'' + ax' + bx = \sin x. \tag{2}$$

As with all second order equations one converts to system form by the substitution

$$x_1 = x, \qquad \text{and} \qquad x_2 = x'. \tag{3}$$

With this substitution, one arrives at the equations

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= -ax_2 - bx_1 + \sin x_1. \end{aligned} \tag{4}$$

**Matlab Commands for Differential Equations**

Below are two ways to solve the initial value problem:

$$\frac{dy}{dt} = y, \qquad y(0) = 1.$$

Solving symbolically:

- `y = dsolve('Dy = y','y(0)=1')`
- `ezplot(y,-4,4)`

Solving numerically:

- `F = inline('y','t','y')` ..... Makes a function $F(t, y) = y$.
- `T = -4:.1:4;` ..... Make a vector of time values.
- `[T,Y] = ode45(F,T,1);`
- `plot(T,Y)`

## 1. Separation of Variables

- Enter the following sequence of commands:

```
syms  t
f = sqrt(9-t^2)
int(f) ..... An antiderivative of f.
int(f, -3, 3) ..... The definite integral.
g = t*acos(t) ..... acos(t) is arccos t
diff(g)
pretty(ans)
diff(g, 2) ..... The second derivative of g.
pretty(ans)
diff(g, 3) ..... The third derivative of g.
pretty(ans)
ezplot(f, [-3, 3])
ezplot(g, [-1, 1])
```

- Remarks.

  The graph of `f` should be the upper half of a circle. It will be distorted because of the default scale on the y-axis. Display the graph again and in the `Figure window`, click on Edit. Pull down to `Axis Properties`. Reset the y-limits to be -1.5 and 3.1, click on Apply and then OK. The graph should now appear more like a semicircle.

  Type `help sym/diff` or `help int` in MATLAB for more info on the use of `diff` or `int`.

Following the methodology above, using a separate piece of paper, do the following.

(1) Find the particular solution to the ODE $y'' = \sec y'$ that is tangent to the $t$-axis at the origin. Use the method of separation of variables, and make sure to include all of the steps. Use MATLAB to compute the appropriate integrals.

(Hint. Let $u = y'$ and remember to add a constant of integration where appropriate.)

(2) Find the area under the graph of the solution of the IVP in part (a) on the interval $[-1, 1]$. Make sure to write the formula you use, not just the answer. Again, use MATLAB to compute the appropriate integrals.

(3) Use MATLAB to plot the solution of the IVP in part (a) on the interval $[-1, 1]$. Sketch the graph, *by hand*. DO **not** get a printout.

(4) Find the absolute maximum value of the solution on the interval $[-1, 1]$.

## 2. Direction Fields

- `dfield6` is a MATLAB program for MATLAB Version 6 that may be retrieved from the web site at `http://math.rice.edu/~dfield/` and other versions are also available at this site. If you don't have it, copy it into `C:\Matlab\Work` (or `C:\MatlabR12\Work`).
- In MATLAB, enter the command: `dfield6`
- A `DFIELD Setup` window appears.
- The differential equation `x' = x^2 - t` appears in the boxes for `The differential equation`.
- Using MATLAB notation, change these entries to enter the differential equation `y' = sin y`.
- The independent variable by default is t so leave that entry unchanged.
- For `The display window` settings,
  - enter `-5` for `The minimum value of t`
  - enter ` 5` for `The maximum value of t`
  - enter `-2*pi` for `The minimum value of y`
  - enter ` 2*pi` for `The maximum value of y`.
- Click on the `Proceed` button. The direction field for your differential equation will appear in another window.
- At the top of this window, you can click on `Options` and pull down to `Window` settings. Here you can select `Arrows` instead of `Lines` for your direction field plot.
- If you click at any point in the direction field plot, a solution curve through that point is plotted. Several solution curves can be plotted by clicking on more than one point.

Following the methodology above, do the following.

(a) Print out or carefully sketch by hand the direction field of the differential equation

$$y' = \frac{2y}{t}$$

$$(Choose -5 \leq t \leq 5 and -10 \leq y \leq 10.)$$

(b) Superimpose some solutions (say, two above the t-axis and two below the t-axis) on the direction field in part (a).

(c) Use the information in parts (a) and (b) to guess a one-parameter family of solutions of the differential equation.

## 3. Homogeneous ODEs with Constant Coefficients

Try the following in MATLAB:

```
syms m
eqn1 = 'm^2 - 3*m-1 = 0'
eqn2 = 'm^4 - 4*m^3 + 14*m^2 - 20*m + 25 = 0'
solve(eqn1)
solve(eqn2)
```

For each of the following differential equations:

- Write down the auxiliary equation.
- Write down, in standard mathematical notation, all of the solutions to the auxiliary equation. (Use MATLAB to find the solutions.)
- Write down the general solution of the differential equation.

(1) y"' + y" - 6y' - 18y = 0

(2) $y^4$ - 2y"' - 6y" + 16y' - 8y = 0

(3) $y^4$ - 3y"' + 7y" + 21y' - 26y = 0

(4) $y^5$ - 2$y^4$ + 2y"' - 4y" + y' - 2y = 0

(5) 2$y^5$ - $y^4$ - 4y"' + 3y" - 8y' - 12y = 0

## 4. Plotting Solutions to First Order Initial Value Problems

Enter the following sequence of commands:

```
F = inline('sin(y)', 't', 'y') .....Defines a function of two variables.
T = 0:.01:10; .....Defines a vector. Do not skip the semicolon.
[T, Y] = ode45(F, T, 1);
plot(T, Y)
```

**Remarks**

1. If you skip the semicolon, you will get a list of the values in `T`.

2. The third statement tells MATLAB to numerically solve the IVP:

$$y' = F(t, y), \quad y(0) = 1.$$

By using `T` as the second argument in the call to `ode45` we are indicating that we want the values of `Y` at the times given in the vector `T`. If you want more info on the use of `ode45`, issue the command `help ode45`.

3. The fourth statement plots a graph of the points

$$(T(1), Y(1)), (T(2), Y(2)), \ldots, (T(1000), Y(1000)).$$

It should appear that the solution has a horizontal asymptote. Try extending the range of the t values to go from 0 to 20. You can re-type the second statement as `T = 0:.01:20;` or you can use the up-arrow key until the statement `T = 0:.01:10;` reappears and then use the left-arrow key to move the cursor left and change the 10 to 20, then press the Enter key. Next you can again use the up-arrow key to recall `ode45` and then press the Enter key. Plot the new values. The up-arrow key and the down-arrow key allow the user to move up and down through the list of previous commands. A command does not get entered until you press the Enter key.

4. What would you guess for the value of the horizontal asymptote?

Using the methodology described above, sketch *by hand*, on a separate piece of paper, the solution of the given initial-value problem on the given interval. DO **not** get a printout.

Make sure you include appropriate numerical values along the axes.

(1) $y' = \dfrac{1}{2} - \cos t, \qquad y(0) = 1, \qquad [0, 30]$

(2) $\dfrac{dy}{dt} = \dfrac{2}{t+1} - y^2, \qquad y(0) = 2, \qquad [0, 30]$

(3) $y' - y = t \cos t, \qquad y(0) = 0, \qquad [0, 20]$

(4) $t\dfrac{dy}{dt} + y = t, \qquad y(1) = 2, \qquad [1, 10]$

(1) Enter the following commands:
   (a) `y = dsolve('Dy=-0.1*y', 'y(0)=1')`
   (b) `ezplot(y, [-20,20])`
   (c) Explain exactly what happened.

(2) Repeat the above procedure to solve and plot the solutions for the following differential equations. Use the same initial condition as above.
   (a) $y'(t) = \sin t$
   (b) $y'(t) = -0.1y + \sin t$
   (c) Explain exactly what happened in each example.

(3) Compare the differential equations in the three examples. Then compare the graphs of the solutions in the three examples. What do you observe from these comparisons?

(4) Prepare a brief (less than 1 page) written report answering all the questions and sketching the graphs carefully by hand. Use complete sentences and standard mathematical notation. Do **not** get a printout.

Students observe that for a linear differential equations qualitative features of solutions tend to "add" as terms are added to the righthand side.

## 5. Linear First-order Equations

(1) Enter the following commands:
   (a) `y = dsolve('Dy=-0.1*y', 'y(0)=1')`
   (b) `ezplot(y, [-20, 20])`
   (c) Explain exactly what happened.

(2) Repeat the above procedure to solve and plot the solutions for the following differential equations. Use the same initial condition as above.
   (a) y'(t) = sin(t)
   (b) y'(t) = -0.1 y + sin(t)
   (c) Explain exactly what happened in each example.

(3) Compare the differential equations in the three examples. Then compare the graphs of the solutions in the three examples. What do you observe from these comparisons?

(4) Prepare a brief (< 1 page) written report answering all the questions and sketching the graphs carefully by hand. Use complete sentences and standard mathematical notation. Do **not** get a printout.

Students observe that for a linear differential equations qualitative features of solutions tend to "add" as terms are added to the righthand side.

## 6. Linear Second-order Equations

(1) Enter the following commands:
   (a) `y = dsolve('D2y+y=0', 'y(0)=1', 'Dy(0)=1')`
   (b) `ezplot(y, [0,100])`
   (c) Explain exactly what happened.

(2) Repeat the above procedure to solve and plot the solutions for the following differential equations. Use the same initial condition as above.
   (a) $y''(t) + y(t) = \sin t$
   (b) $y''(t) + 0.1y' + y(t) = 0$
   (c) $y''(t) + 0.1y' + y(t) = \sin t$

(3) Compare the differential equations in the four examples. Then compare the graphs of the solutions in the examples. Based on things you have learned in class, explain the differences between the examples.

(4) Prepare a brief (less than 1 page) written report answering all the questions and sketching the graphs carefully by hand. Use complete sentences and standard mathematical notation. **Do not get a printout.**

Students explore the interaction of damping, restoring, and forcing effects on the solution.

## 7. A Spring-Mass System

(1) Type the following commands (at the prompt and then press Enter):
   (a) `dsolve('2*D2y + .5*Dy + 5*y = sin(a*t)', 'y(0)=1', 'Dy(0)=1')`
   (b) `y1 = sub(y, a, 1)` ..... Substitutes "1" for $a$.
   (c) `ezplot(y1, [0,50])`
   (d) Explain exactly what happened.

(2) Repeat (b) and (c) for different values of `a`, both more and less than 1. By trial and error find a value of $a$ that maximizes the amplitude of the solution. From the equation, what is its 'natural' or 'resonant' frequency? What should happen when `a` is set to this value? Test your hypothesis.

(3) Prepare a brief (less than 1 page) written report answering all the questions. Use complete sentences and standard mathematical notation. Do **not** get a printout.

The user examines what happens when a system is excited at different frequencies, the relationship between natural frequency and amplitude of the forced, damped oscillator.

## 8. Laplace Transforms

## 9. Linear versus Nonlinear

(1) Try the following commands (at the prompt and then press Enter):
   (a) `syms t y`
   (b) `dsolve('D2y + y = 0', 'y(0) = 2', 'Dy(0) = 2')`
   (c) `ezplot(ans, [0,50])`
   (d) Change the initial conditions to $y(0) = .2$, $y'(0) = .2$. How does this affect the solution?
   (e) Explain exactly what happened.

(2) Repeat the above procedure to solve the the following differential equation. Use the initial conditions: $y(0) = 1$, $y'(0) = 1$.

$$y''(t) - y(t) + y^3(t) = 0$$

Why is MATLAB unable to solve this equation symbolically?

(3) Note that the equation in #2 may be written as a system by the substitution $y_1 = y$, $y_2 = y'$. This produces the system:

(5)
$$\frac{y_1}{dt} = y_2$$
$$\frac{y_2}{dt} = y_1 - y_1^3$$

Now try the following:
   (a) `F = inline('[y(2);y(1) - y(1)^3]','t','y')` ..Makes `F` the r.h.s. of (1).
   (b) `T = 0:.01:50;` .....Don't skip the semicolon!
   (c) `[T, Y] = ode45(F, T, [2,2]);`
   (d) `plot(T, Y(:,1))`

Try changing the initial conditions to $y(0) = .2$, $y'(0) = .2$. How does this effect the solution? How does this differ from the linear case?

(4) Use the commands you learned in #3 to numerically solve and plot:

$$y''(t) - y(t) + y^3(t) = \sin t, \qquad y(0) = 1, \quad y'(0) = 1$$

on the interval $t = [0, 100]$. How does the graph of this solution differ from all the graphs of solutions you have seen for linear equations.

(5) Prepare a brief (less than 1 page) written report answering all the questions. Use complete sentences and standard mathematical notation. Do **not** get a printout.

This assignment demonstrates that the solutions of linear equations are very "tame" compared with solutions of nonlinear equations.

## 10. Special Functions

The antiderivatives of many elementary functions are not, themselves, elementary functions. Some of these antiderivatives arise frequently in certain subjects and have been given names. These are examples of what are called special functions, and other antiderivatives can sometimes be expressed in terms of these special functions.

Try the following MATLAB commands:

```
syms t
int(exp(-t^2))
int(sin(t^2))
```

Look up the definition of the special functions involved using the `mhelp` command which calls up the help feature in the Maple kernel.

For the following find the general solution on the indicated interval, by hand, using the method of variation of parameters, except using MATLAB to integrate $u_1'$ and $u_2'$.

**Make sure you show all your work! Look up and write down, in standard mathematical notation, any special functions that occur and any unfamiliar expressions or constants that appear in these definitions.**

(a) $y'' - 3y' = \dfrac{1}{t}, \qquad t > 0$

(b) $y'' - 2y' + 2y = \dfrac{e^t}{t}, \qquad t > 0$

CHAPTER 7

# Linear Algebra

MATLAB is short for "Matrix Laboratory" and the original program was designed specifically to handle matrix operations.

When dealing with matrices in real applications, size and speed are considerations. One should know that the process of solving equations by Gaussian elimination is a relatively fast operation, it requires approximately $n^2$ operations where $n$ is the size of the matrix. On the otehr hand, finding the determinant of a matrix in the concentional way requires $n!$ operations, which is impossible even for fast machines when $n$ is moderately large.

Symbolic computations with matrices are very limited because of inherent limitations such as the insolvability of quintic equations. So for instance, when MATLAB tries to find the eigenvalues of a matrix bigger than $4 \times 4$, it cannot do it symbolically with a determinant, but numerically with iterative approximations.

## Matlab Commands for Linear Algebra

**Making vectors:** Unless otherwise specified, variables are row vectors (`1 x n` arrays). Here are examples of ways to form vectors. Try them:

- `b = [1 2 3 4]`
- `b = b'`
- `xx = 0:.1:2`
- `yy = linspace(0,3,13)`

**Making matrices:**

- `A = [1 2 3; 4 5 6]`
- `C = eye(3)`
- `D = ones(4)`
- `E = zeros(5,3)`
- `F = rand(2,3)`
- `G = randn(5)`
- `H = hilb(5)`
- `P = pascal(4)`
- Commands for other speciality matrices include: gallery, hadamard, hankel, invhilb, magic, rosser, toeplitz , vander, wilkinson.

**Basic operations:**

- `B = A'`
- `A*C`
- `C*A` ..... Will not work, `C` is 3 by 3 and `A` is 2 by 3.
- `x = P \backslash b` ..... Solves `Px=b`.
- `P*x` ..... Checks the previous command.

**Some speciality commands**

- `[m n] = size(A)`
- `P = pascal(5), p = diag(P)`
- `diag(p)`
- `flipud(A)`
- `fliplr(A)`
- `v = randn(10,1), a = abs(v)`
- `s = sort(v), m = max(v)`
- `norm(v)`

- `norm(eye(4))`
- `D, N = Null(D), D*N`
- `rank(D)`
- `det(D)`
- `trace(D)`
- `inv(G), N*G, G*N`
- `cond(H)`

**Some matrix decompostions:**

- `[L U P] = lu(G)`
- `[V m] = eig(G)`
- `[U T] = shur(G)`
- `[Q R] = qr(G)`
- `[U S V] = svd(G)`

## 1. Matrix Operations

(1) Try the following commands (at the prompt and then press Enter):

```
clear
M = [1,3,-1,6;2,4,0,-1;0,-2,3,-1;-1,2,-5,1]
det(M)
inv(M)
```

(2) Repeat the above procedure for the matrix:

$$N = \begin{bmatrix} -1 & -3 & 3 \\ 2 & -1 & 6 \\ 1 & 4 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

(3) Multiply $M$ and $N$ using `M*N`. Can the order of multiplication be switched? Why or why not? Try it to see how MATLAB reacts.

(4) Find the determinant and inverse of the following matrix:

$$A = \begin{bmatrix} 1.2969 & .8648 \\ .2161 & .1441 \end{bmatrix}$$

(5) Let $B$ be the matrix obtained from $A$ by rounding off to three decimal places. Find the determinant and inverse of $B$. How do $A^{-1}$ and $B^{-1}$ differ? Explain how this happened.

(6) Prepare a brief ($< 1$ page) written report describing what happened and answering all the questions. Use complete sentences and standard mathematical notation. Do **not** get a printout.

This exercise introduces some basic matrix operations, the importance of matrix dimensions, and numerical sensitivity.

## 2. Solving Linear Systems

(1) Try the following commands (at the prompt and then press Enter):

(a) `A = [1.2969, 0.8648; 0.2161, 0.1441]`

(b) `b1 = [1.2969; 0.2161]`

(c) `x = A\b1`

(d) Repeat the process but with a vector $b2$ obtained from $b1$ by rounding off to three decimal places.

(e) Explain exactly what happened. Why was the first answer so simple? Why do the two answers differ by so much?

(2) Try the following commands:

(a) `B = sym(maple('matrix','2,2','(I,J)->sin(I*J)'))`

(b) `c = [1;2]`

and use `x = B\c` to solve $Bx = c$. Then change the 2's to 3's in the first line, change $c$ to $[1; 2; 3]$ and try to solve again. Use `x=double(x)` to obtain an approximate numerical value of the solution. Try the command `Bn = double(B)`, then `x = Bn\c`. When would an exact symbolic solution and when would an approximate numerical solution be more useful? For big matrices, which type of computation would be faster?

(3) Input the matrix:

$$C = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

and solve $Cx = d$ with $d1 = [4; 8]$ and $d2 = [1; 1]$. Use symbolic and non-symbolic versions of $C$. Explain the results. Which way gives more information?

(4) Prepare a report as follows:

(a) **Using standard mathematical notation**, write down the results of all the computations, except the symbolic solution to the $3 \times 3$ system in #2. Do **not** get a printout.

(b) **Using complete sentences**, briefly answer all of the questions. This includes giving explanations where requested.

The matrix in #1 is nearly singular, causing the linear system to be very sensitive to perturbations. Students are exposed to both symbolic and numerical solutions. The ideas of no solutions or infinitely many solutions are reinforced.

### 3. LU Decomposition

### 4. Least Squares

### 5. Eigenvalues and Eigenvectors

(1) Try the following commands
   (a) `digits(4)`
   (b) `A = sym([1,1; 0,1])`
   (c) `E = eig(A)`
   (d) `[V,E] = eig(A)`

   Find the eigenvalues and eigenvectors for this matrix by hand and interpret the output.

(2) Input the **symbolic** matrix (use `sym` as above):

$$B = \begin{bmatrix} 3 & -1 & -1 \\ -1 & 0 & 2 \\ 1 & 1 & -3 \end{bmatrix}$$

   and try the commands:
   (a) `SE = eig(B)` ..... Finds eigenvalues symbolically.
   (b) `NE = eig(vpa(B))` ..... `vpa` changes from symbolic to numeric.
   (c) `[SV,SE] = eig(B)`
   (d) `[NV,NE] = eig(vpa(B))`

(3) Create a matrix using the command: `C = sym(hilb(5))`,
   and repeat the process in the previous part.

(4) What are your observations about symbolic vs. numerical computations from the last two parts?

(5) Using complete sentences and standard mathematical notation, write a brief report. Show your hand calculations and answer all the questions.

In the first example students must consider multiplicities. The last part should lead to a discussion of the fact that polynomials of degree 5 or higher cannot in general be solved symbolically and so exact symbolic eigenvalues cannot be found for 5 by 5 matrices. They should also notice that symbolic solutions are sometimes too complicated to be useful.

## 6. Eigenvalue Power Method

(1) Enter the following sequence of commands:

```
format long
A = hilb(5);
m = eig(A)
v = ones(5,1)
w = v./norm(v);
```

(2) Next enter the the following sequence:

```
v = A*w;
w = v./norm(v);
ma = w'*A*w
```

(3) Repeat the steps in part 2 until the value of `ma` stops changing.How many iteratations did it take? Is this number close to one of the eigenvalues? How close?

(4) Repeat the above experiment for the Pascal matrix generated by: `A = pascal(5)`.

(5) Repeat the experiment for a larger matrix.

(6) Using complete sentences and standard mathematical notation, write a brief report.

This demonstrates the simplest form of the QR method. Most modern software including MATLAB's built-in function "eig" use improved versions of this algorithm.

## 7. Eigenvalues by the QR Method

(1) Enter the following sequence of commands:

```
format long
A = hilb(5)
m = eig(A)
m = flipud(m)
```

(2) Next enter the the following sequence:

```
[Q,R] = qr(A);
A = R*Q;
ma = diag(A);
e = norm(m-ma)
```

(3) Record the value of `e`. Repeat the steps in the above sequence until the value of `e` stops changing. Assume that the errors satisfies $e_{n+1} = Ke_n^r$ and use the recorded data to solve for r and K.

(4) Repeat the above experiment for the Pascal matrix generated by: `A = pascal(5)`.

(5) Repeat the experiment for a larger matrix.

(6) How do the computed values of r and K vary in your experiments?

(7) Using complete sentences and standard mathematical notation, write a brief report.

This demonstrates the simplest form of the QR method. Most modern software including MATLAB's built-in function "eig" use improved versions of this algorithm.

CHAPTER 8

# Programming in Matlab

## Commands for Matlab Programming

# 1. Programming in the Command Window

## 2. Writing and Running an m-file

## 3. Defining a function in an m-file

# 4. Determinates of Random Matrices

## 5. Cobweb graphs for discrete dynamical systems

## 6. Approximate Double Integrals

(1) Go to the O.U. Matlab website (`www.math.ohiou.edu/~matlab`). In 263D, under this assignment, click on: `lowerleft.m`
This will download a program file; save it to your working directory.

(2) Open Matlab. In the command window, check the current directory, and if needed, change to the directory where you saved `lowerleft.m`.

(3) In the command window, type the following commands:

```
format long
f = inline('x*y^2','x','y')
lowerleft(f,0,1,0,2,10,20)
```

Matlab should return the answer `1.1115000000000`. Calculate the integral exactly by hand and find the % error.

(4) Click on the file icon at the upper left corner and open the file `lowerleft.m`, this will be the program that was used in the command above by the same name. Read the program and comments.

(5) Save the program as `centerpoint.m`. Modify it to do Riemann sums using the centerpoint rather than the lowerleft point. You will need to change the name of the function in the first line to `centerpoint`, otherwise, you will need to change very little.

(6) Test your new program on $f(x, y) = xy^2$ by typing:

```
centerpoint(f,0,1,0,2,10,20)
```

This answer should be closer to the right answer than `lowerleft` Test this using the % error.

(7) Try out your new program on the following integral:

$$\int_0^2 \int_0^5 \sqrt{xy + y^5 + x^2}\, dy\, dx$$

Also try this integral using the command `dblquad` as in the previous homework. For `m` and `n` fairly large, does your program come close to Matlab's built in program?

(8) Using complete sentences and standard mathematical notation, write a brief report.

This assignment gives student a chance to work with Riemann sums and gives a very gentle introduction to programming in Matlab.

CHAPTER 9

# Sample Solutions

Some words about sample solutions.

## 1. Sample Solution For: 'Factoring Expressions and Solving Equations'

(1) The command `clear` clears all variables.

The command `syms x` declares $x$ to be a symbolic variable.

The command `expr1 = (x-1)*(x-2)*(x-3)*(x-4)*(x-5)` gives the label `expr1` to the expression $(x-1)(x-2)(x-3)(x-4)(x-5)$.

The command `expand` is used to expand or multiply out an expression. Expanding `expr1` yields

$$x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120$$

The command `factor` is used to factor an expression. Here we factor the expression that results from expanding `expr1`. Thereby, we recover `expr1`, which is what one would expect.

Solving `expr2 = 0` gives $x = 1, 2, 3, 4, 5$.

The relationship between solving and factoring is as follows: Let $p(x)$ be any polynomial. $x = x_0$ is a solution of $p(x) = 0$ if and only if $(x - x_0)$ is a factor of $p(x)$.

(2) MATLAB is not able to factor $x^4 + 3x^3 + 3x^2 + x + 3$.

MATLAB *is* able to solve $x^4 + 3x^3 + 3x^2 + x + 3 = 0$, symbolically; however, the solutions it gives are extremely long and complicated.

The command `double(ans)` numerically evaluates `ans`, in this case the symbolic solutions to $x^4 + 3x^3 + 3x^2 + x + 3 = 0$. (Note. `double(ans)` does not mean $2 \cdot \text{ans}$; `double` is short for double precision.) The numerical solutions we obtain are

$$x = 0.2289 \pm 0.8595i, \quad -1.7289 \pm 0.8959i$$

One reason an exact, symbolic solution may not be as useful as an approximation is that when we measure things we usually use decimals or very simple fractions.

(3) Solving `expr3-3` gives $x = 0, -1, -1, -1$. The reason the answer is so nice is that `expr3 - 3` is

$$x^4 + 3x^3 + 3x^2 + x = x(x^3 + 3x^2 + 3x + 1) = x(x+1)^3$$

(4) MATLAB is unable to factor `expr4` or to solve `expr4=0`, symbolically. MATLAB gives the numerical solutions (to four decimal places)

$$x = 0.9615, \quad 2.2093, \quad 2.7342, \quad 4.1510, \quad 4.9541$$

None of the algorithms MATLAB uses to obtain symbolic solutions to polynomial equations work for this equation, so MATLAB provides an approximate numerical solution.

It is known from higher mathematics that, for polynomial equations of degree five or higher, a symbolic solution is not always possible.

## 2. Sample Solution For: 'Defining, Evaluating, and Plotting Functions'

(1) The command `syms x` declares $x$ to be a symbolic variable. The command `f = sin(x)` makes $f$ the symbolic function $\sin x$.

(2) The command `subs(f, 2)` numerically evaluates $f(2)$. The command `subs(f, '2')` symbolically evaluates $f(2)$. The command `double(ans)` numerically evaluates `ans`, in this case $\sin 2$. (Note. `double(ans)` does not mean $2 \cdot \text{ans}$; `double` is short for double precision.)

(3) The command `ezplot(f)` plots $y = \sin x$ using a default domain interval.

(4) First, we plot $y = \exp(x) = e^x$ using a default domain interval. Second, we plot $y = \exp(x)$ using the domain interval $[-2, 2]$.

(5) This plots $y = x^2$ using a default domain interval.

(6) (a) The graph of $y = \sin(x^5)$ oscillates (goes up and down) quite rapidly for not too large values of $|x|$. (For example, between $x = 3$ and $x = 4$, the graph of $y = \sin(x^5)$ oscillates

$$\frac{4^5 - 3^5}{2\pi} \approx 124 \quad \text{times.)}$$

MATLAB cannot accurately portray the graph of a function that oscillates so rapidly.

(b) For $x$ not too close to 0,

$$\sqrt{x^2 - .00001} \approx \sqrt{x^2} = |x|,$$

and this is what the graph looks like using the default domain interval. However, if $|x|$ is close enough to 0, then the graph looks different (in fact,

$$\sqrt{x^2 - .00001} \quad \text{is not a real number if } x^2 < .00001.)$$

Some distinctive features of the graph are not apparent, because of the scale that MATLAB chooses.

(c) The value of $|x^7 - x|$ is relatively large for relatively small $|x|$. For example, $2^7 - 2 = 126$. Using the default domain it appears as though $x^7 - x = 0$ for $x$ in $[-1, 1]$, which, of course, is *not* the case. As in (b), some distinctive features of the graph are not apparent, because of the scale that MATLAB chooses.

(7) MATLAB plots function by locating points on the graph and connecting the points. If `ezplot` is used, MATLAB will choose a default domain interval if one is not specified.

If the true graph oscillates too rapidly, the computer may not fill in enough points to give an accurate representation. If the domain is not chosen properly, important features of the graph may be missed. Often, but not always, problems with plotting can be alleviated by choosing a different domain interval.

### 3. Sample Solution for 'Indefinite Integrals'

(1) (a) The command `syms x` declares x to be a symbolic variable.

(b) The command `int(x^2)` is used to find an antiderivative (indefinite integral) for $x^2$, namely, $\frac{1}{3}\,x^3$. Note that MATLAB omits "+C" from the answer.

(c) The command `diff(ans)` is used to differentiate the previous answer. The result is $x^2$.

The derivative of an indefinite integral of a function is the original function.

(2) Using the command `int` we obtain

$$\int \frac{x}{(x - 1)(x + 2)\,(x^2 - 1)(x + 1)}\,dx$$

$$= \frac{1}{(x - 1)} - \frac{1}{36}\ln(x - 1) - \frac{2}{9}\ln(x + 2) + \frac{1}{4(x + 1)} + \frac{1}{4}\ln(x + 1)$$

Note that MATLAB uses `log` for the natural logarithm which is denoted by ln in most calculus textbooks.

Using the command `diff(ans)` we obtain

$$\frac{1}{12(x - 1)^2} - \frac{1}{36(x - 1)} - \frac{2}{9(x + 2)} - \frac{1}{4(x + 1)^2} + \frac{1}{4(x + 1)}$$

The command `simplify(ans)` instructs MATLAB to attempt to simplify the previous answer. In this case, we obtain

$$\frac{x}{(x - 1)^2(x + 2)(x + 1)^2}$$

Since $x^2 - 1 = (x + 1)(x - 1)$, this is easily seen to be equal to the original function.

(3) (a) MATLAB gives a long answer in terms of the (unknown) roots of a fifth degree polynomial.

(b) MATLAB cannot find an explicit integral.

(c) MATLAB gives the answer in terms of another indefinite integral which is no simpler.

The function in #2 is a rational function for which the denominator can easily be written as the product of linear factors. There are well-known techniques for integrating such a function.

The function in #3(a) is also a rational function. However, there is no "formula" for factoring a general fifth degree polynomial into linear and irreducible quadratic factors. This accounts for the nature of the given solution.

For the functions in #3(b) and (c), none of the algorithms MATLAB uses to obtain explicit integrals are successful.

It is a fundamental fact that not all elementary functions have an antiderivative in terms of elementary functions.

CHAPTER 10

# Summary of Matlab Commands