Journal of Information & Communication Technology
Vol. 6, No. 1, (Spring 2012) 18-26

Teaching Optimization Techniques using Matlab and Mathematica: A Comparative Study

Ingila Rahim *
Mathematical Sciences Research Center.
Federal Urdu University of Arts, Science and Technology, Karachi, Pakistan.

Muhammad Arif Hussan *
Institute of Business & Technology (IBT), Karachi, Pakistan.

ABSTRACT

This paper aims to explore how technology-based teaching (with *Matlab* and *Mathematica*) helps students in understanding optimization methods. Students solve linear programming problems using *Matlab* and *Mathematica*. Some examples and case study are used to discuss and explain issues that have been observed while using technology in completing this course. Results show that *Mathematica* linear programming algorithm is better than the algorithm for the same command in *Matlab*. Students also learn to counter check important results of optimization problems at hand. This sort of experimental mathematics helped students in improving their confidence levels in attacking optimization problems.

INSPEC Classification : C1180, A0140G, A0150H, A0150K

Keywords : *Matlab*, *Mathematica*, optimization problem, comparison of *Matlab* and *Mathematica*

1. INTRODUCTION

Computer algebra systems (CAS) are gaining increasing interest in the every day work of students. The use of these application software help students a lot in understanding the concepts of physics and mathematics. However, students erroneously assume that CAS tools always know more than they do about how to compute a mathematical expression. It is recommended for students to perform calculations in more than one ways in order to counter check final results of problems at hand (Aguirregabiria JM, Hernandez A, and Rivas M, 1994).

Research (Hoyles C. & Noss R., 2003), (Kendal M. & Stacey K., 1999) shows that students learn more from an organized lecture when they are technology-based. Other group says that (Wicks, R. J., 1999), (Wicks, R. J., 1996) *Mathematica* and Maple are two such systems with which we can create rich learning experience for our students. There has been a wide

*The material presented by the authors does not necessarily portray the viewpoint of the editors and the management of the Institute of Business and Technology (IBT) or Federal Urdu University of Arts, Science and Technology.

* Muhammad Arif Hussain : wmarif2002@yahoo.com
* Ingila Rahim : Ingila.rahim@gmail.com

© JICT is published by the Institute of Business and Technology (IBT).
Ibrahim Hydri Road, Korangi Creek, Karachi-75190, Pakistan.

range of computer activities (Shaw W T and Tigg J., 1994)(Davis B, Porta H and Uhl J., 1994) used in teaching mathematical concepts. In order to enhance students' understanding of abstract linear algebra concepts, the author worked on classroom activities supported by *Matlab* and *Mathematica* to provide inquiry-based learning environments. *Mathematica* ahs also powerful graphics capability (Maeder R, 1999).

According to a comparison of the two CAS tools, *Mathematica* can do all the numeric math and matrix work that *Matlab* can do, as fast and as accurately. Many of Mathematica's functions switch automatically between different algorithms, whereas in *Matlab* one must always choose the algorithm. Matlab requires one to manually linearize ordinary differential equations (ODEs) to a system of first order equations before it can solve them, but *Mathematica* accepts any order ODE. *Matlab* can do integration up to only three dimensions (with triplequad), but *Mathematica* handles n-dimensional (http://www.helium.com/items/895074-comparing-matlab-and-mathematica, 2012).

Other viewpoint is that Mathematica is easier to use. It is better for symbolic and analytic job, whereas Matlab is good for numeric computations (http://www.physicsforums.com/showthread.php?t=196740 , 2012).

Section 2 gives some detail of linear programming mathematical formulation. In Section 3, basics commands of *Matlab* and *Mathematica* is discussed to attack mathematical problems. Finally, Section 4 concludes this paper

2. Linear Programming (LP) Formulation

In general, linear programming problems minimize a linear function of real variables over a region defined by linear constraints (Nash, S. and Sofer, A. 1996).

$$\min \quad c^T x \quad \textit{subject to} \quad Ax = b,\ x \geq 0,$$

where x is a vector of n real numbers, and $Ax = b$ is a set of linear equality constraints and $x \geq 0$ reveals that all variables are nonnegative. The dual of this problem can be formulated as

$$\max b^T \lambda \quad \textit{subject to} \quad A^T \lambda + s = c,\ s \geq 0,$$

where l is a vector of Lagrange multipliers (Smith K. J. and Bradley G. L., 1999) and s is a vector of dual slack variables. These two problems are intimately related, and algorithms typically solve both of them simultaneously. When the vectors x^*, l^*, and s^* satisfy the following optimality conditions:

$$\begin{aligned}
&A^T \lambda^* + s^* = c,\\
&Ax^* = b,\\
&x^* \geq 0, s^* \geq 0,\\
&\left(x^*\right)^T s^* = 0,
\end{aligned}$$

then x^* solves the primal problem and (l^*, s^*) solves the dual problem.

In the next section we summarize the basics of *Matlab* and *Mathematica* commands which help to solve linear and nonlinear problems.

3. *Matlab* and *Mathematica* Basics

This Section gives basics of popular computer algebra systems (CAS), namely *MATLAB* and *Mathematica* for the specification of numerical as well as symbolic calculations, and linear programming (LP) commands, which in turn helps in getting solution to a given optimization problem. These tools generally provide partial functionality of the kind to be created in many real world projects. We try to illustrate important features of these tools in LP problem solving. Students are encouraged to develop their own problem solving skills with CAS tools. In addition, all these computer activities are supposed to be augmented by classroom lessons where ideas are discussed and applied.

3.1. Some Basic Operations in *MATLAB*

Addition of numbers in Matlab.

```
>> 2 + 1/2

ans =

2.5000
```

This shows use of MATLAB as a calculator. Now, we look at the matrix representation in two different formats.

```
>> b=[1,2,3;2,0,9;0,0,3];
>> c=[1 2 3;2 0 9;0 0 3];

>> b'


ans =

1   2   0
2   0   0
3   9   3

>> c'

ans =

1   2   0
2   0   0
3   9   3
```

The commas in the specification of **b** can be replaced with spaces, as depicted by matrix **c**. Square brackets refer to vectors and parentheses are used to refer to elements within a matrix. The command **det** and **inv** return the determinant and inverse of a matrix respectively. The ' performs the transpose of a matrix. Transpose of matrices **b** and **c**, as shown above.

The **help** command returns information on different commands.

3.1.1. Calculus with *MATLAB*

Differentiation

For constructing symbolic objects, we define

```
>> syms x                                % defines x as symbolic variable
>> y=x*x; diff(y,x)                    % diff differentiates y w.r.t. x

ans =

2*x
```

For its second derivative we write

```
>> y=x*x;diff(y,x,2)

ans =

2
```

Integration

MATLAB uses **int** command to integrate. For example, int(*S*, *x*) is the indefinite integral of S with respect to *'x'*. Similarly, int(*S*, *v*, *a*, *b*) is the definite integral of S with respect to *v* from *a* to *b*. We demonstrate these with few examples.

```
>>    syms x t;
>>    int(1/(1+x^2))

ans =

atan(x)
```

3.1.2. Linear Programming in *Matlab*

In Matlab *LINPROG* (or *linprog*) command is used for Linear programming. The syntax is as follows (Venkataraman P., 2001).

X=LINPROG(f,A,b) attempts to solve the linear programming problem:

min f'*x   subject to: A*x $\leq$ b

Another option is:

X=LINPROG(f,A,b,Aeq,beq) solves the problem above while additionally satisfying the equality constraints Aeq*x = beq.

Similarly, we can use

X=LINPROG(f,A,b,Aeq,beq,LB,UB) defines a set of lower and upper bounds on the design variables, *X*, so that the solution is in the range LB$\leq$ *X*$\leq$ UB. Use empty matrices for LB and UB, if no bounds exist. Set LB(i) = -Inf if *X* (i) is unbounded below; set UB(i) = Inf if *X* (i) is unbounded above.

In addition Matlab has other commands as well for linear and nonlinear programming.

3.1.2.1. Linear Programming Examples

We solve following examples (Hoffmann LD and Bradley GL., 1995), with Matlab command *linprog*
Examples:

1) We solve the LP problem:
Maximize $P = 1.2\,H + L$

Subject to: $10\,H + 12\,L \leq 1920$
$5\,H + 3\,L \leq 780$
$H, L \leq 0$

Matlab Input:

```
>> f=[-1.2,-1];
A=[10,12;5,3];
b=[1920 780];
>> [x,fval]= linprog(f,A,b)
```

Matlab Output:

```
Optimization terminated.
x =
120.0000
60.0000

fval =
-204.0000
```

Minus sign shows that we solveded a problem of *Maximization.*

2) We solve the LP problem:
Minimize $F = 7\,x_1 + 20\,x_2$
Subject to: $x_1 + 2\,x_2 \geq 2$
$x_1 + 5\,x_2 \geq 3$
$x_1, x_2 \geq 0$

Solution to this problem is: $F = 16$, x_1, $= 4/3$, $x_2 = 1/3$

3) We solve the LP problem:
Maximize $F = 2x_1 + x_2 + 3\,x_3$
Subject to: $x_1 + 2x_2 + x_3 \leq 12$
$2\,x_1 + x_2 + x_3 \leq 20$
$x_1 + x_2 + 2\,x_3 \leq 20$
$x_1, x_2, x_3 \geq 0$

Matlab **linprog** command fails to solve this simple example. The same set of examples will be solved by *Mathematica* in the next section.

3.2. Symbolic Computations in *Mathematica*

[Input]: $x + x$
[Output]:$2x$

[Input]: $x = 3$; $y = 5$; $x + y$
[Output]:8

Mathematica has many powerful features which go far beyond even what is needed for calculus. We define some functions for the purpose of evaluation. The syntax for defining function is as follows:

g[x_] := x^3 + 5;
h[x_] := -2*x + 3;

Here, we defined two functions $g(x)$ and $h(x)$. It is necessary to use := and the underline after the variable. It is also important to note that functional dependence in *Mathematica* is always denoted by square brackets [ ].

[Input]: **g[2]**
[Output]: 13

Similarly,

[Input]: **h[2]**
[Output]: -1

Sign ? (or ??, for more detail) can be used to get help on any command. For instance, to get help on **FindRoot** command in *Mathematica* do the following.

[Input]: **?FindRoot**
[Output]: FindRoot[lhs==rhs, {x, x_0}] searches for a numerical solution to the equation lhs==rhs. Similarly, FindRoot[{*eqn1*, *eqn2*, …}, {{x, x_0},{y, y_0},…}] searches for a numerical solution to the simultaneous equations *eqni*.

3.2.1. Calculus in *Mathematica*

Differentiation

The derivative $g'(x)$ can be obtained by using command: **D**[ ] as given below.

[Input]: **D[g[x],x]**
[Output]:$3x^2$

Integration

We define another function $f(x)$: **f[x_] := 1 / (x + 5);**
Now, we perform integration in *Mathematica* environment.

[Input]: **Integrate[f[x],x]**
[Output]:Log[x+5]

The above result is exact. One can also obtain numerical integration of the same function as follows.

[Input]: **NIntegrate[f[x], {x,2,4}]**
[Output]:0.251314

3.2.2. Linear Programming in *Mathematica*

In Mathematica, there are several ways to solve Linear Programming problems. One command which is based matrix formulation is given below (Wolfram Research, Inc., 2003).
LinearProgramming[c, m, b]. This command finds vector x which minimizes cx.

Subject to the constraints mx ≥ b and x ≥ 0
Similarly, one can use

LinearProgramming[c, m, b, l]. This command finds vector x which minimizes cx.

Subject to the constraints $mx \geq$ b and x ≥ l

Mathematica also has other commands Minimize, Maximize, NMinimize, and NMaximize for the solution of linear and nonlinear optimization problems. Maximize and Minimize commands give exact (or symbolic) solutions to the problems, whereas NMinimize, and NMaximize commands show numeric solutions to the problems.

3.2.2.2. Examples

Here, we solve the same set of examples as given in Section 3.12.1.

1) We solve the LP problem:
Maximize $P = 1.2\,H + L$
Subject to: $10\,H + 12\,L \leq 1920$
$5\,H + 3\,L \leq 780$
$H, L \geq 0$

LinearProgramming[{-1.2 ,-1},{{-10,-12},{-5,-3}},{-1920,-780}]

Answer: $P = 204$, $H = 120$, $L = 60$.

2) We solve the LP problem:
Minimize $F = 7\,x_1 + 20\,x_2$
Subject to: $x_1 + 2\,x_2 \geq 2$
$x_1 + 5\,x_2 \geq 3$
$x_1, x_2 \geq 0$

Answer: $F = 16$, $x_1 = 4/3$, $x_2 = 1/3$.

3)We solve the LP problem:
Maximize $F = 2\,x_1 + x_2 + 3\,x_3$
Subject to: $x_1 + 2x_2 + x_3 \leq 12$
$2\,x_1 + x_2 + x_3 \leq 20$
$x_1 + x_2 + 2\,x_3 \leq 20$
$x_1, x_2, x_3 \leq 0$

Matlab was unable to solve this problem as we observed in the previous section. We solving this LP problem with Mathematica by two different commands.

Mathematica Solution:
We use the first command *LinearProgramming[ ]*

1) LinearProgramming[{-2,-1,-3},{{-1,-2,-1},{-2,-1,-1},{-1,-1,-2}},{-12,-20,-20}]
Output: {**4, 0, 8**}
Maximum *F* comes out to be **32**. at *x*1= 4, *x*2 = 0, and *x*3 = 8

And now we use another command *Maximize[ ]*
2) Maximize[{2a+b+3 c,a+2 b+ c≤ 12,2 a+ b+ c≤ 20,a+ b+2 c≤ 20,a≥ 0,b≥ 0,c≥ 0},{a,b,c}]

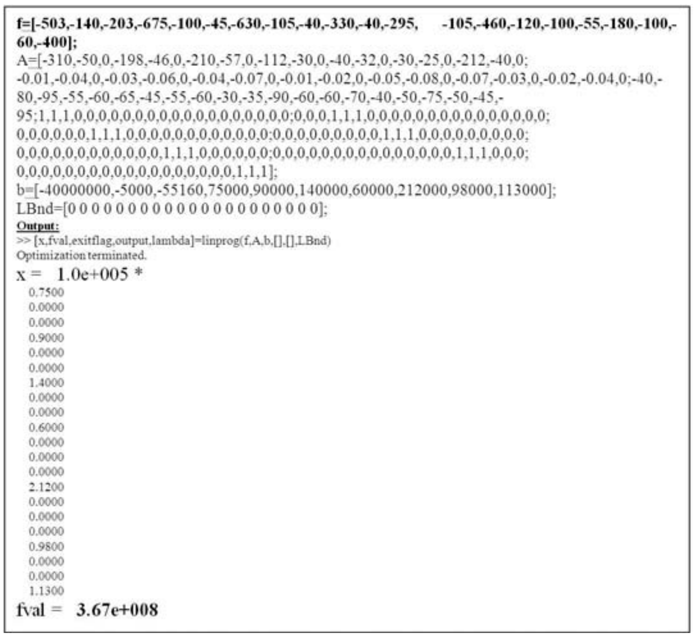
Here, a = x_1 b = x_2 , c = x_3 .
Output is: ***F* = 32,** a→**4,** b→**0,** c→**8**
.Above answer is same as given in the reference text. Next we solve a case study with Matlab and Mahematica.

3.2.2.3. Case Study

Forest Service Allocation: The US forest services has used just such an allocation model to address the sensitive task of managing 191 million acres of national fresh land (Rardin R L.,1998). The forest service must tradeoff timber, grazing, recreational, environmental, national preservation, and other demands on forestland. For this purpose it defines a linear programming model in order to maximize total net present value (NPV) per acre of all uses in area. Twenty one variables (all positive) are defined.

The above problem is solved with *Matlab* and *Mathematica*. Results are more or less similar to the published report from the two CAS tools. However, *Mathematica* result sems to be better, as it is closer to the reported NPV in the report.

a) MATLAB Solution: **Box 1**: Malab **linprog** command to solve the case study.

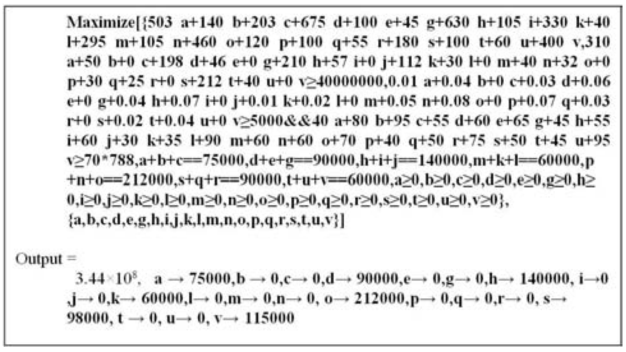


```
f=[-503,-140,-203,-675,-100,-45,-630,-105,-40,-330,-40,-295,    -105,-460,-120,-100,-55,-180,-100,-
60,-400];
A=[-310,-50,0,-198,-46,0,-210,-57,0,-112,-30,0,-40,-32,0,-30,-25,0,-212,-40,0;
-0.01,-0.04,0,-0.03,-0.06,0,-0.04,-0.07,0,-0.01,-0.02,0,-0.05,-0.08,0,-0.07,-0.03,0,-0.02,-0.04,0;-40,-
80,-95,-55,-60,-65,-45,-55,-60,-30,-35,-90,-60,-60,-70,-40,-50,-75,-50,-45,-
95;1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0;0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0;0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1];
b=[-40000000,-5000,-55160,75000,90000,140000,60000,212000,98000,113000];
LBnd=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
Output:
>> [x,fval,exitflag,output,lambda]=linprog(f,A,b,[],[],LBnd)
Optimization terminated.
x =   1.0e+005 *
   0.7500
   0.0000
   0.0000
   0.9000
   0.0000
   0.0000
   1.4000
   0.0000
   0.0000
   0.6000
   0.0000
   0.0000
   0.0000
   2.1200
   0.0000
   0.0000
   0.0000
   0.9800
   0.0000
   0.0000
   1.1300
fval =   3.67e+008
```

Box 1 gives the solution of the above cases tudy. Now, we perform calculation with *Mathematica.*

As *Maximize* command in *Mathematica* can accommodate all types of constraints found in the literature therefore, we solve this case study with *Maximize* command . **b**) Mathematica Solution:

Box 2. *Mathematica* notebook showing *Mathematica* command for Maximization.



```
Maximize[{503 a+140 b+203 c+675 d+100 e+45 g+630 h+105 i+330 k+40
l+295 m+105 n+460 o+120 p+100 q+55 r+180 s+100 t+60 u+400 v,310
a+50 b+0 c+198 d+46 e+0 g+210 h+57 i+0 j+112 k+30 l+0 m+40 n+32 o+0
p+30 q+25 r+0 s+212 t+40 u+0 v≥40000000,0.01 a+0.04 b+0 c+0.03 d+0.06
e+0 g+0.04 h+0.07 i+0 j+0.01 k+0.02 l+0 m+0.05 n+0.08 o+0 p+0.07 q+0.03
r+0 s+0.02 t+0.04 u+0 v≥5000&&40 a+80 b+95 c+55 d+60 e+65 g+45 h+55
i+60 j+30 k+35 l+90 m+60 n+60 o+70 p+40 q+50 r+75 s+50 t+45 u+95
v≥70*788,a+b+c==75000,d+e+g==90000,h+i+j==140000,m+k+l==60000,p
+n+o==212000,s+q+r==90000,t+u+v==60000,a≥0,b≥0,c≥0,d≥0,e≥0,g≥0,h≥
0,i≥0,j≥0,k≥0,l≥0,m≥0,n≥0,o≥0,p≥0,q≥0,r≥0,s≥0,t≥0,u≥0,v≥0},
{a,b,c,d,e,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v}]

Output =
  3.44×10^8,  a → 75000,b → 0,c→ 0,d→ 90000,e→ 0,g→ 0,h→ 140000, i→0
j→ 0,k→ 60000,l→ 0,m→ 0,n→ 0, o→ 212000,p→ 0,q→ 0,r→ 0, s→
98000, t → 0, u→ 0, v→ 115000
```

Box 2 gives the input and output of the case study pergormed with *Mathematica*.

4. Conclusion and Future Outlook

We demonstrated three simple examples of optimization problems and a case study to guide students using *Matlab* and *Mathematica* optimization commands and helped them in technology-based learning environments. Comparison of *Matlab* and *Mathematica* reveals that *Mathematica* is better for exact and analytical calculations, whereas *Matlab* is faster for numeric calculations. Reported case study is the result of ongoing research investigating the effect of technology on teaching and learning optimization techniques. Results revealed that *Mathematica* linear programming algorithm wss better than the algorithm for the same command in *Matlab* In fact many factors influence the implementation of technology in imparting such know how to students. Finally, we can say that *Matlab* and *Mathematica* can be used to improve the students' understanding of optimization techniques course as well as to counter check important results of problems at hand. These application softwares help imparting knowledge to students in a shorter time frame. Further discussion on graphical method of solving optimization problems will be the topic of a future paper.

5. References

Aguirregabiria JM, Hernandez A, and Rivas M, 1994,"Are we careful enough when using computer algebra system", Computers in Physics, **8** (1)

Hoyles C. & Noss R., 2003, 'What can digital technologies take from and bring to research in mathematics education? In: A.J. Bishop, M.A. Clements, C. Keitel, J. Kilpatrick & F. Leung (Eds.)', *Second International Handbook of Mathematics Education* (Vol. 1, pp. 323-349). Dordrecht: Kluwer Academic Publishers.

Kendal M. & Stacey K., 1999, Varieties of teacher privileging for teaching calculus with computer algebra systems, *International Journal of Computer Algebra in Mathematics Education* 6, 233-247.

Wicks, R. J., 1999, *Mathematica* materials from MAA mini-course on creating interactive texts in *Mathematica*. MAA 1999 Joint meetings.

Wicks, R. J., 1996, Linear Algebra: an interactive laboratory approach with *Mathematica*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Shaw W T and Tigg J., 1994, "Applied *Mathematica*: Getting Started, Getting It Done", Addison-Wesley, Massachusetts..

Davis B, Porta H and Uhl J., 1994, "Calculus andd *Mathematica*:, Addison-Wesley, Massachusetts..

Maeder R, 1999, "Programming in *Mathematica*", Addison-Wesley, Massachusetts. http://www.helium.com/items/895074-comparing-matlab-and-mathematica, 2012. http://www.physicsforums.com/showthread.php?t=196740 , 2012.

Nash, S. and Sofer, A. 1996. *Linear and Nonlinear Programming*. McGraw-Hill. Smith K. J. and Bradley G. L., 1999, "A *Mathematica*:Approach To Calculs 2/e", Addison-Wesley, Massachusetts..

Venkataraman P., 2001, 'Applied Optimization with Matlab Programming', Wiley.

Hoffmann LD and Bradley GL, 1995, 'Finite Mathematics with Calculus', McGraw-Hill, NY., pp. 153-204.

Wolfram Research, Inc. 2003 , *Mathematica*, Version 5.0, Champaign, IL.

Rardin R L, 1998, "Optimization in Operations Research", Prentice-Hall, NJ, pp. 132-134..