Journal of Kerbala University , Vol. 9 No.1 Scientific . 2011

BUILD A NEW BLOCK TO SOLVE A SYSTEM OF LINEAR EQUATIONS USING S-FUNCTION

بناء بلوك جديد لحل نظام من المعادلات الخطية باستخدام دالة-S

Riyad Mubarak and Mohammed A. Al-Taee
Computers Sciences/Education College / Mosul University

الملخص

في هذا البحث تم بناء بلوك جديد باستخدام دالة S لطريقة SMW التي تستخدم لحل نظام من المعادلات الخطية لعدم توفر هذا البلوك في الـ Simulink. تم تطبيق هذا البلوك الجديد على عدد من العينات ذات الأبعاد 2x2 إلى حد 9x9 من النظام الخطي, فضلاً عن ذلك يمكن تطبيقه على عينة *nxn* من المصفوفات, كما نجحنا ايضا في اختصار عامل الوقت.
نستطيع استخدام هذا النظام لحل المشاكل المتعلقة بمعالجة الصور الرقمية لأن الصور ممكن تحويلها إلى مصفوفات وحلها بهذا النظام.

Abstract

In this research done build a new block by use S-function for SMW method which use to solve linear equations system because not find this block in simulink . Done application this a new block on some samples 2x2 to 9x9 dimensions from linear system, and able to application to *n* x *n* matrixes moreover we pass docking in time.
We can use this system to solve devotes problems of digital images process because this images able to transfer to matrixes and solve it with this system.

1.1 INTRODUCTION

$A\underline{x} = \underline{b}$ equality is a form of linear equation stimulant which is usually used to resolve a technical issue. Equation $A\underline{x} = \underline{b}$ is a matrix multiplication effect because the number of columns from *n* matrix *A* is the first commensurate with the number of lines *n* matrix of the second $\underline{x}$. During this has been a lot of similarities $A\underline{x} = \underline{b}$ computerizes with the goal to $A\underline{x} = \underline{b}$ equation can be solved by using a computer. In connection with this settlement system equation Ax = b in computerizes, the authors are interested in creating a program for settlement of the system equation Ax = b using a device-based graphics software with the goal of the program in order to resolve the system equation $A\underline{x} = \underline{b}$ become more friendly to users.
In this paper we limit the resolution of problems in the model of linear equation ($A\underline{x} = \underline{b}$) using method of SMW (Sherman-Morrison-Woodbury). This method will modeling into a form blocks with mathematical diagrams using simulink is a facilities-based graphical user interface that is provided by device-MATLAB software version 7.0.

1.2 Form of the General System of Linear Equation

General form of linear equation, which consists of the *m* equation and *n* variables, can be written as follows:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_i \qquad (1.1)$$

The number *aij* is a coefficient to-*i* of variables to-*j*. While the b_i stated in constant segment to the right to equality to-*i*. Then, the coefficients can be formed in the matrix is a notation:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix} \quad (1.2)$$

Called the matrix coefficients. Coefficients for a variable that is not written, the coefficient matrix is written as the number zero. Constant in the right segment is often written as a column vectors

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_m \end{bmatrix} \quad (1.3)$$

Equation (1.2) can be written as a single matrix equation is as follows:

$A\underline{x} = \underline{b}$ (1.4)

With [1], [2]

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix} \quad \text{and} \quad \underline{b} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix} \quad (1.5)$$

1.3 Method SMW

Sherman, Morrison and Woodbury separately to find the relationship as below:-

$$(A + \underline{u}\ \underline{v}^{\mathrm{T}})^{-1} = A^{-1} - \frac{1}{1+\underline{v}^T A^{-1}\underline{u}}\ A^{-1}\underline{u}\ \underline{v}^T A^{-1}$$

$$\beta = \frac{1}{1+\underline{v}^T A^{-1}\underline{u}} \text{ It's a real number.} \quad (1.6)$$

Prove the truth of the relationship in nature is done by using the Inverse matrix

$$Q = (A + \underline{u}\ \underline{v}^T) = A^{-1} - \frac{1}{\beta}\ A^{-1}\underline{u}\ \underline{v}^T A^{-1} = I.$$

$$Q = A\,A^{-1} - \frac{1}{\beta}\ A\,A^{-1}\underline{u}\ \underline{v}^T A^{-1} + \underline{u}\ \underline{v}^T A^{-1} - \frac{1}{\beta}\ \underline{u}\ \underline{v}^T A^{-1}\underline{u}\ \underline{v}^T A^{-1}$$

$$= I - \frac{1}{\beta}\ \underline{u}\ \underline{v}^T A^{-1} + \underline{u}\ \underline{v}^T A^{-1} - \frac{\beta-1}{\beta}\underline{u}\ \underline{v}^T A^{-1}$$

$$= I - \left(\frac{1}{\beta} - 1 + \frac{\beta-1}{\beta}\right)\underline{u}\ \underline{v}^T A^{-1} = I \quad (1.7)$$

SMW excess of the relationship, that is if there are two matrix *A* and *B*, with only *B* different from *A* of $\underline{u}\ \underline{v}^T$ course, the $B = A + \underline{u}\underline{v}^T$. SMW relations show if a known inverse, the Inverse of *B* can be calculated by making corrections on the Inverse of *A*. Based on the facts above, the Inverse random matrix Z (Inverse if it exists) can be determined by the application of repeated relations SMW, contrary to the facts from the beginning, for example, that the inverse of matrix unit is also matrix unit [2].

1.3.1 SMW Algorithm

In the SMW method, steps in the algorithm among others:

0. Initial steps: the initial matrix A

- $B = A\text{-}I;$
- $A^{-1} = I;$
- $n = size(n)$

1. Iteration to do k:= 1 to n

$\underline{u} = k$ from the field to
column k from B
$v^T = \underline{e}^T \times k$
calculate $\beta = 1 + v^T * A^{-1} \times \underline{u}$;
{ if $\beta = 0$ then
print 'no inverse'
go to 2
end }

calculate $A^{-1} = A^{-1} - \frac{1}{\beta} \underline{u}\ \underline{v}^T A^{-1}$

2. Completing the equation by using the linear nature of Inverse matrix:

$A\underline{x} = \underline{b}$
$\underline{x} = A^{-1}\underline{b}$

1.4 Simulink

Simulink using blocks that familiar diagram, the system that illustration in the form of writing can be easily implemented in Simulink. Simulation system that has been done is interactive, meaning that the parameters block diagram can be changed and changed every time its parameters can be directly seen the changes that occur in the output.

For modeling, Simulink provide blocks diagrams that have the nature of the GUI (Graphical User Interface) provides many options on the block, splitting a problem and blocks the only need to be connected with a line connecting a mouse [3].

1.4.1 Elements in a Model

A simulink model consists of three types of elements, namely the sources, which it used to modeling system, and the purpose:

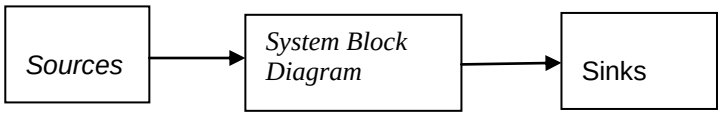

Figure 1.1 Elements of a model in Simulink.

Figure 1.1 illustrates the relationship between the three types of elements that are in a Simulink model. Elements that are in the middle of the system block diagram is a representation of the blocks diagram that will be used for the model of a dynamic system. Is the source of entries in a dynamic system? Source can consist of constant, function generators such as sine wave, or input signal generated in the MATLAB.

1.4.2 Programming S-function

S-function is the programming language used to describe a block. If in the use of a Simulink blocks, there are no blocks that are needed so users can create their own block of programming required to use the s-function. Basically programming s-function is the same with the m-file in the normal MATLAB, only difference is in how book. S-function written with a special template that can function on the block Simulink. S-function has several main parameters that must be the:

- Input parameter = *t*, *x*, *u*, and *flag*.
- Output parameter =*sys* and *x*0.

So in programming s-function is the minimum

function [*sys*, *x*0] = *Fung_name* (*t*, *x*, *u*, *flag*)

where:

t = *Time*
x = *state*
u = *input*

flag = calling mode function
*x*0 = *state* awal
sys = have a value depending on the value of the *flag*
In addition, in the s-function programming known flags. Flags are working to call each of which has a function duties. In the *s*-function, there are 9 flags and flags of 9, only 4 flags needed by the author in solving the equation *Ax* = *b*, namely

- ***Flag* 0:**

To initiate the initial work. Each time *s-function* is called, then the flag will call this function mdlInitializeSizes. Contents mdlInitializeSizes functions, namely: sys and *x*0.

- ***Flag* 2:**

Work to run and modify a process that is made from the blocks. This flag will call the function mdlUpdate. When mdlUpdate function is called, then the calculation process will be conducted.

- ***Flag* 3:**

Working to get the output from the process started. Flag call this function mdlOutputs. After the flag is called, then the sys will contain the results of the process.

- ***Flag* 9:**

Is a flag terminal to end the simulation. Functions that are called by the flag, this is a function mdlTerminate. In making blocks, the authors use the programming m-file programming because the m-file more concise and more easily understood. But the actual s-function can be made using the C language, C + +, Fortran, There are files, and MEX. MEX file is a file in the C language, which is compiling. Below is an example of programming m-files and files in the C s-making function [3]:

1.4.3 S-function Programming Using m-file in Matlab

```
% This function is called by the flag 0
% Works to initiate the initial value
function [sys,x0,str,ts]=mdlInitializeSizes(n)
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = n*n;
sizes.NumOutputs     = n*n;
sizes.NumInputs      = n*n;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = zeros(n*n,1);
str = [];
ts  = [0 1];
% This function is called by the flag 2
% Works to run the calculations
function sys=mdlUpdate(t,x,u,n)
for j=1:n,
   for i=1:n
      a(I,j)=u((i-1)*n+j);
   end
end
c=a-eye(n,n);
a=eye(n,n);
```

```
for k=1:n
   z=1+a(k,: ) *c(:,k);
   a=a-a*c(:,k)*a(k,:)/z;
end
for j=1:n,
   for i=1:n
      sys((i-1)*n+j)=a(i,j);
   end
end [4].
```

1.4.4 Blocks SMW Method

Blocks used in this method include:

- To input used DSP blocks constant. The process used in the block SMW, transpose block, block, block and reshape matrix multiplication.
- To block the display output is used. Model of the design method for squares matrix and subsystem blocks from the SMW is shown in Figure 1.2, 1.3, 1.4 and 1.5 the following:

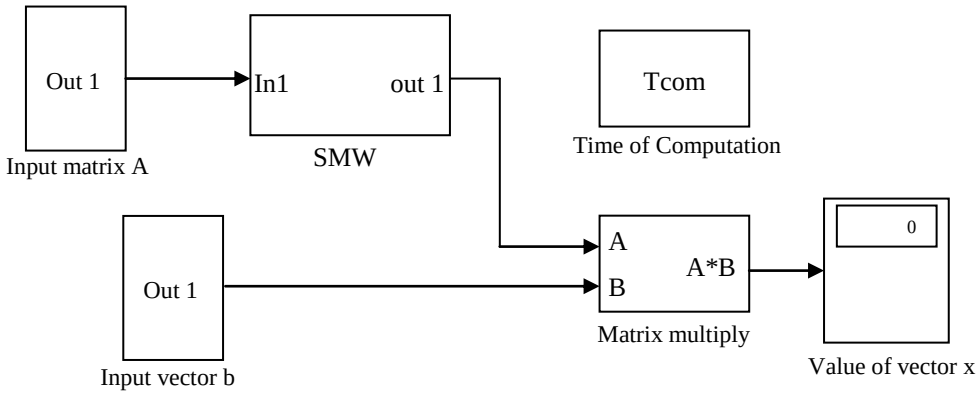

Figure 1.2 Model mathematical methods SMW for matrix square.

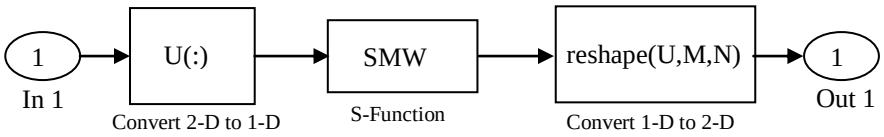

Figure 1.3 SMW subsystems Model.

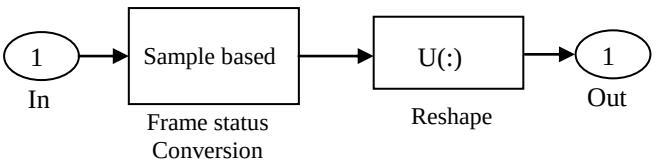

Figure 1.4 Subsystem model block convert 2-D to 1-D

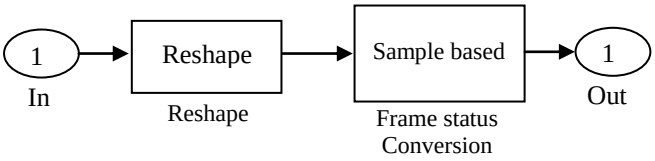

Figure 1.5 Subsystem model block convert 1-D to 2-D.

In making blocks SMW, the author uses three flags, namely the flag 0, flag 2, and the flag 3. Here are the s-function blocks to the SMW

- Function name: *function* [*sys,x0, str, ts*] = *SMW* (*t, x, u, flag, n*)
- At *flag* 0

```
function [sys, x0, str, ts]=mdlInitializeSizes(n)
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = n*n;
sizes.NumOutputs     = n*n;
sizes.NumInputs      = n*n;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = zeros(n*n,1);
str = [];
ts  = [0 1];
```

Variable *n* is a variable to the size of the matrix will be processed, so block SMW not automatically made in the process matrix. This aims to introduce the one hand again in the programming Simulink. In other words, the value of the variable n will be set from the window handle and MATLAB graph using the set_param. For example, for matrix cleanup with size 4x4, then use the command set_param as follows: to block the SMW

Set_param ('Sherman/SMW','n','4')'

Set_param command means set the value of the parameter n in the block SMW files that are on the model given the name "Sherman" with a value of 4.

• At flags 2. This flag is used to call the function mdlUpdate that will make the process of calculation SMW as follows:

```
function sys=mdlUpdate(t,x,u,n)
for j=1:n,
   for i=1:n
      a(i,j)=u((i-1)*n+j);
   end
end
c=a-eye(n,n);
a=eye(n,n);
for k=1:n
   z=1+a(k,:)*c(:,k);
   a=a-a*c(:,k)*a(k,:)/z;
end
for j=1:n,
   for i=1:n
      sys((i-1)*n+j)=a(i,j);
   end
end
```

In this flag, the value sys is the result of the calculation.

•At flags 3. This flag is a flag to call the function mdlOutput that will send the results of calculation to block other, as follows:

```
function sys=mdlOutputs(t,x,u,n)
sys=x;
```

1.5 Numerical Results

We generate a few sample systems as follows [5]:

Table 1.1 Sample test matrix squares.

Orde Matrix	A	x	B
2×2	$\begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$	$\begin{bmatrix} 9 \\ 8 \end{bmatrix}$
3×3	$\begin{bmatrix} 4 & 3 & 2 \\ 2 & 1 & 1 \\ 1 & 2 & 2 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$	$\begin{bmatrix} 6 \\ 4 \\ 5 \end{bmatrix}$
4×4	$\begin{bmatrix} 4 & 7 & -7 & 2 \\ 6 & 5 & -3 & 3 \\ 5 & 6 & 7 & 2 \\ 4 & 5 & 5 & 54 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$	$\begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$
5×5	$\begin{bmatrix} 4 & 4 & 5 & 7 & 3 \\ 6 & 5 & -6 & 5 & 3 \\ 5 & 6 & -3 & 3 & 3 \\ 7 & 3 & 6 & 54 & 6 \\ 5 & 2 & 4 & 43 & 7 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 6 \\ 4 \\ 4 \\ 5 \end{bmatrix}$
6×6	$\begin{bmatrix} 4 & 76 & 6 & 4 & -6 & 5 \\ 6 & 6 & 5 & 6 & -5 & 4 \\ 5 & 4 & 5 & 6 & 54 & 54 \\ 6 & 5 & 6 & 5 & 5 & 4 \\ 6 & 3 & 5 & -7 & 4 & 4 \\ 5 & 8 & 5 & -5 & 4 & 4 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$	$\begin{bmatrix} 2 \\ 54 \\ 4 \\ 5 \\ 4 \\ 4 \end{bmatrix}$
7×7	$\begin{bmatrix} 5 & 6 & 2 & -7 & 4 & 5 & 5 \\ 6 & 2 & 4 & 8 & 3 & 2 & 4 \\ 5 & 1 & 3 & 6 & 2 & -8 & 3 \\ 4 & 3 & 5 & 5 & 3 & 6 & 4 \\ 7 & 2 & -9 & 5 & 4 & 5 & 5 \\ -2 & 4 & -7 & 4 & 4 & -4 & 5 \\ 7 & 3 & -5 & 4 & 4 & 6 & 4 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$	$\begin{bmatrix} 4 \\ 34 \\ 2 \\ 5 \\ 4 \\ -7 \\ -6 \end{bmatrix}$
8×8	$\begin{bmatrix} 4 & -9 & 4 & 3 & 4 & 4 & 1 & 4 \\ 5 & -8 & 6 & 5 & 56 & 4 & 4 & 34 \\ 4 & 9 & 5 & 4 & 6 & 3 & 12 & 3 \\ 6 & 10 & 9 & 6 & 7 & 6 & 11 & 4 \\ 5 & 10 & 8 & 5 & 6 & 7 & 15 & 5 \\ 3 & 3 & 12 & 4 & 5 & 8 & 2 & 5 \\ 4 & 0 & 23 & 4 & 4 & 1 & 3 & 5 \\ 3 & 5 & 4 & 4 & 4 & 3 & 2 & 6 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$	$\begin{bmatrix} 6 \\ 54 \\ 4 \\ 3 \\ 6 \\ 10 \\ 30 \\ 40 \end{bmatrix}$

9×9	$\begin{bmatrix} 5 & 6 & 2 & 18 & 4 & 9 & 1 & 4 & 5 \\ 4 & 3 & 1 & 11 & 3 & 6 & 3 & 6 & 6 \\ 6 & 4 & 3 & 34 & 56 & 5 & 2 & 5 & 3 \\ 7 & 5 & 4 & 4 & 6 & 4 & 4 & 5 & 6 \\ 6 & 6 & -9 & 6 & 4 & 3 & 3 & 6 & 8 \\ -8 & 5 & 0 & 5 & 7 & 3 & 4 & 7 & 1 \\ -6 & 1 & 8 & 4 & 6 & 3 & 0 & 6 & 6 \\ -5 & 2 & 9 & 5 & 8 & 24 & 43 & 5 & 68 \\ 8 & 3 & 7 & 6 & 7 & 3 & 5 & 6 & 54 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}$	$\begin{bmatrix} 7 \\ 6 \\ 5 \\ 43 \\ 3 \\ 55 \\ 5 \\ 5 \\ 9 \end{bmatrix}$

And the result showed in Table 1.2:

Table 1.2 Results test matrix squares.

Orde Matrix	**Results of Calculation**	**SMW**
	Value of x	Time
2×2	$\begin{bmatrix} 3 \\ 2 \end{bmatrix}$	0.22
3×3	$\begin{bmatrix} 1 \\ -2 \\ 4 \end{bmatrix}$	0.22
4×4	$\begin{bmatrix} 0.1325 \\ 0.3742 \\ -0.1256 \\ -0.01432 \end{bmatrix}$	0.28
5×5	$\begin{bmatrix} 1.794 \\ -1.19 \\ 0.1701 \\ -0.2369 \\ 1.131 \end{bmatrix}$	0.33
6×6	$\begin{bmatrix} -5.931 \\ -1.203 \\ 8.785 \\ 0.04308 \\ -5.894 \\ 5.789 \end{bmatrix}$	0.38

7 × 7	$\begin{bmatrix} 395.5 \\ 231.5 \\ -312.8 \\ 419.2 \\ -3906 \\ 4.212 \\ 659.4 \end{bmatrix}$	0.93
8 ×8	$\begin{bmatrix} -9.768 \\ -0.4947 \\ -0.2103 \\ 14.16 \\ -3.287 \\ -3.864 \\ 0.2634 \\ 6.701 \end{bmatrix}$	1.22
9 × 9	$\begin{bmatrix} -0.5091 \\ 11.2 \\ 5.548 \\ 5.161 \\ -2.867 \\ -17.23 \\ 9.944 \\ 0.3434 \\ -1.309 \end{bmatrix}$	1.55

1.6 Conclusions

We conclude from this research can add any block for any method with use S–function in the Simulink and done this method successfully to solve linear equations system with progressed in examples. Done application this a new block on some samples 2x2 to 9x9 dimensions from linear system, and able to application to *n* x *n* matrixes moreover we pass docking in time.

1.7 Future Works

Use this method in digital images process because digital images are matrixes, moreover can compare this method with another methods like LU factorization (to factor the *n*x*n* matrix $A = (a_{ij})$ into the product of the lower-triangular matrix $L=(l_{ij})$ and the upper-triangular matrix $U=(u_{ij})$; that is $A=LU$, where the main diagonal of either L or U consists of all one) and QR (is a matrix reduction technique used to simultaneously determine all the eigenvalues of a symmetric matrix) [2].

Reference

[1] John, H., M., and Kurtis D. F., Numerical Methods using Mathlab, third edition Prentice Hall.

[2] Richard L. B., and J. Douglas Faires, 2001, Numerical Analysis, Seventh Edition, USA.

[3] عبد الكريم البيكو, MATLAB 6.5 Tutorial and Reference ,2003, سوريه حلب.

[4] Simulink Dynamic System Simulation for MATLAB writing S-Function ver 3, 1998, MathWorks, Inc.

[5] MATLAB The Language of Technical Computing Computation, Visualization and Programming, 2002, MathWorks, Inc.