MATVIEW 1.1
A Two Dimensional Matrix Visualization Tool

by

George V.J. Townsend
and
W-P. Tang

Research Report CS-89-36
August, 1989

A Two Dimensional
Matrix Visualization
Tool

MATVIEW1.1

USER'S GUIDE

George V. J. Townsend
W-P. Tang

Department of Computer Science
University of Waterloo

Table of Contents

1 An Introduction to MATVIEW

This chapter is an introduction to MATVIEW. We begin by discussing the importance of visualization tools to the researcher. The problems with some existing tools for displaying matrix data are discussed along with the features possessed by MATVIEW which allow it to overcome some of these problems.

In the next part of the chapter, some of the capabilities of MATVIEW are briefly covered. Although MATVIEW was designed as a visualization tool, it has many features which the user will find to be a helpful aid in allowing the details of boundary value problems to be easily specified.

The final portion of this chapter discusses some of the future plans to expand the features of MATVIEW.

2

The Motivation Behind MATVIEW

The results of scientific computation often yield large amounts of numerical data. It is very difficult for a user to quantitatively examine more than a tiny part of a given solution. In other words, it is very difficult to observe the global quantitative nature of solutions. The area of scientific visualization provides a solution to this problem. Even in its simplest form, that of a graph in an x-y Cartesian coordinate system, visualization is an invaluable aid in allowing a user to develop an intuitive understanding of the relationship between elements in a solution.

The graphs required by the modern computer user are more complex than a simple curve in an x-y Cartesian coordinate system since the relationships of interest tend to involve multidimensional quantities in both the domain and range of the graph. In investigating the trends found in two-dimensional matrices and their inverses, we are faced with a visualization problem in which we are interested in being able to discern the relationship between the location of an element in a matrix and its numerical value. In the case of a sparse matrix (or its inverse) resulting from a boundary value problem, we may be interested in how the numerical value of an element is related to some location in the physical problem and to neighbouring points.

In an effort to provide a solve problem, solutions have been developed which represent such information as a sort of three-dimensional graph in which the location of an element is represented in a very natural way as an x-y location in a plane, and the numerical value is represented as the height in the z direction. Although this is a useful representation, it suffers from one important problem. Generally we must convey this three-dimensional construction to the user on a two dimensional output device such as a printer or a CRT screen. In doing so, we must project it onto a two dimensional plane. As a result, points which may be of interest in a valley, may be hidden behind a peak in this representation.

The Motivation Behind MATVIEW

Animation may provide a partial solution to this problem, since the user may rotate the image to view all aspects of the construction. The obvious drawbacks of this approach are that it is compatible only with a CRT screen, and requires interaction with the user. It is not possible to easily adapt animation to printers and plotters which are able to only show a single view of such a construction. As a result, an author is unable to share this form of visualization in a publication. A more subtle drawback of animation, is that the user is not able to discern a global view of the data at a given instant in time. That is, as the construction is rotated to reveal one feature, a previously visible feature may become hidden.

In an effort to cope with the problem of visualizing matrix data, MATVIEW constructs a visual image which affords the user a single global view of the data. Approaches such as contour plots achieve this, but require interpretation on the part of the viewer and are therefore not as intuitively understood as the representation used by MATVIEW. This representation is two-dimensional in nature which makes it perfectly compatible with either a CRT screen or a printer, and more importantly allows a researcher to easily communicate research results in a publication. The location of an element is represented as a position in an x-y Cartesian coordinate system, while the numerical value is represented on a grey scale as the intensity of the point in this x-y plane. Primitive attempts have been made at this before by using various alphanumeric symbols to represent different values, but the images produced this way are very coarse, and difficult to notice trends in. MATVIEW uses a high quality graphics system of producing a grey scale in which the user is presented with an image in which it is much easier to appreciate any trends present in the data.

MATVIEW is a tool designed primarily to provide the user with a clean user friendly method of investigating the global view of the nature of a linear operators. As we know, a linear operator can be represented as a matrix or a template operator depending on the application [TANG87]. The sparsity structure of a sparse matrix may be viewed not only in

terms of zeros and non-zeros, but also in terms of the value of the non-zeros, shown in shades of grey. MATVIEW allows a user to see the inverse of a such a matrix, and note its exponential decay. The discrete Green Function (inverse template operator) may be viewed, as well as its counterpart which uses the inverse of the incomplete Cholesky factorization rather than the true inverse. The purpose of this is to help study the effect of the ordering of the unknowns on the preconditioning matrix.

MATVIEW is a run time statement interpreter that reads user input as a file of statements or a program, interprets each command, and executes the appropriate actions based on the command. The user's program may define the location of boundary conditions and regions of varying conductivity in either the X or Y direction, as well as sinks and sources. The number of grid lines desired on the discretization grid may be specified by the user, as well as the ordering to be used on the unknowns. The programming language used by MATVIEW consists of only two major groups of commands. These are the *Region and Boundary Definition* commands discussed in Chapter 3, and the *Action* commands discussed in Chapter 4.

By using the *Definition* commands, all the details of a physical problem may be specified. Setting the grid size, and selecting the ordering of the unknowns to be used is also accomplished by commands in this group. These definitions are always with respect to the "unit square". The user must then think of his physical region as being mapped onto, and/or imbedded in, the unit square. Therefore all horizontal and vertical co-ordinates within the region are real numbers in the range 0 to 1. In general, the user will specify the location of a region to be defined within his unit square by giving a pair of X co-ordinates followed by a pair of Y co-ordinates which define a rectangle within the unit square. There are predefined areas recognized by MATVIEW for the convenience of the user. An example of this is the command, *Region*, which may be used to refer to the entire unit square without having to use pairs of X-Y co-ordinates.

What Can MATVIEW Do?

Once the region has been fully defined by the user, the commands falling within the *Action* command group may be used to cause MATVIEW to perform calculations and generate output based on the definition of the user's physical problem description. This group of commands includes provisions for annotating the output generated by MATVIEW, and for producing graphical output of regions, matrices, and other data structures stored within and computed by MATVIEW. In order to keep track of which output plots are of which structure, MATVIEW maintains a title list in which the user may specify the desired title for a particular plot.

As a programming convenience, comments and blank lines may be used at will by the user throughout a MATVIEW program. Apart from recognizing blanks and tab characters as delimiters between words, MATVIEW ignores them. This feature allows comments or commands to easily be tabbed off to one side or the other to make a MATVIEW program more readable. Note: MATVIEW treats round parentheses, commas, and the equal sign as blanks which allows these characters to be used freely to increase the readability of a MATVIEW program. Another convenient feature is the *Trace* command described in Chapter 2 which is useful as an aid to debugging a MATVIEW program which will not execute due to user errors in the program.

Many physical aspects of a boundary value problem may viewed using MATVIEW. The regions of conductivity may be shown in shades of grey corresponding to the magnitude of the conductivity at each grid point in the physical problem. The ordering of the unknowns may be visualized in shades of grey. The solution, sources, and sinks may also be viewed using MATVIEW.

MATVIEW also provides user support for specifying a boundary value problem resulting from the discretization of partial differential equations. Given a user's definition of a physical region, the package will produce the corresponding sparse matrix structure. The user may have the package order the unknowns according to a number of ordering schemes, and solve the resulting sparse matrix using an iterative fast matrix solver. The package supports a wide range of output graphics devices such as laser printers in the form of files which can be processed and sent to suitable output device using the *University of Waterloo IM graphics tool kit.* The User may specify the conductivities in the X and Y directions of any rectangular area of a square grid. This grid may contain discontinuous regions or holes. Various combinations of Neumann and Dirichlet boundary conditions may be specified as well as flow sinks and sources located arbitrarily on the discretization grid. The resulting sparse matrix structure may be printed on a laser printer along with its inverse, or any row of the inverse mapped back onto the physical region as a template.

This manual will begin by covering a general discussion of what MATVIEW is and how it may be used. Chapter 2 will present an introduction to the Language used by MATVIEW. Chapter 3 will elaborate on this discussion with examples throughout. The commands used to describe the locations and types of regions within the discretization grid will be covered.

A different class of commands called *Action* commands will be covered in Chapter 4. These commands are used after a discretization grid has been specified in order to cause the package to operate on the data, solving the matrix and producing graphics output.

Chapter 5 will cover the popular *Stones* Problems, as well as some other examples to illustrate how to use this tool and the flexibility of the language. The final Chapter consists of a brief summary of all the commands accepted by MATVIEW.

The Future of MATVIEW

It is assumed that the user/reader is familiar with the terminology used in sparse matrix technology. No background material on this subject will be provided in this manual, however the reader may consult the bibliography for reference material on this subject [STONES68]. Furthermore, it is also assumed that the reader is familiar with FORTRAN, and with computer programming concepts in general.

MATVIEW interfaces with a package called MAT1 which is uses a fast iterative technique to find the solution to a matrix equation. More information on this package may be found in the reference section [KFOR89].

Future plans for MATVIEW include the addition of colour as an aid to visualization. Currently the pixel size is fixed. In the future the pixel size used by MATVIEW will be adjustable in order to increase the usefulness of the visualization features.

The current version of MATVIEW is intended to produce laser plots. In the future, a preview option will be added to permit viewing of a plot on an X window terminal.

Provision will be included in a future version of MATVIEW to support general second order boundary value problems including first order terms. We also plan on including more general discretization schemes such as finite element discretization.

2 ■ The Language of MATVIEW

The MATVIEW Language is described in this Chapter. It is simple to learn and easy to use due in part to the fact that is has a small vocabulary. One very nice feature of MATVIEW is the ability to run programs correctly with misspelled commands. This is because of the small vocabulary used in MATVIEW and because MATVIEW only checks enough letters in each of the user's commands to arrive at a minimally unique abbreviation for the command. As a result, a user may often desire to use commands which are more readable or more appropriate as desired in place of the "official" commands. For example, any command beginning with the letters "Tr" will be recognized as the *Trace* command by MATVIEW. Some users may then prefer to use the command "Tracking On" rather than "Trace On".

2 The Language of Matview

Throughout the remainder of this manual, when a command is given it the portion of the command line which is actually checked by MATVIEW to recognize the command will be shown in a larger typeface. For example, there is a *Trace* command which will be discussed later. Since only the first two letters are checked for this command, it will be shown as **TR**ACE. This notation will only apply to each command as it is first introduced, and in the command summary in chapter 7. The regular long form will be used elsewhere. The user may either use the minimally unique abbreviation shown in boldface by itself, or may use it as the prefix of any arbitrary string of characters. This allows the *Trace* command to be represented as "Tracing" or "Tracking" as desired. As a result, the user may customize MATVIEW commands slightly as desired.

Commands may be represented in either upper or lower case except where otherwise noted. Either case may be used for the portion of the command following its unique abbreviation. The first character may be either upper or lower case. The case of the remaining characters must all match the case of the second character. The second character may be either case unless the first character is lower case in which event it must also be lower case. The following represent valid representations of the *Trace* command:

```
TRAC
trac
Trac
Trace


TRACe
TracE
```

The last two look awkward, but are valid according to our rules above. The following example however is not valid:

```
tRACE
```

The basic rule of thumb to use is that if the case looks reasonable as with the first four examples, then it is probably acceptable.

■ Program Debugging and Documentation Conveniences

In order to enhance a MATVIEW program, documentation may be added to it in the form of comments. The symbol "#" may be used in a user's program to indicate a comment. A completely blank line is also considered to be a comment by MATVIEW, so that blank lines may be used to increase the readability of a program. Comments may be mixed on the same line as a command. In such a case all text following the comment symbol "#" on a line is ignored. Here are some examples:

```
# This is a comment.
          # This is a comment too!
      # another comment
```

Here are examples of comments occurring after a command on the same line. In these examples the *TRACE* command is used:

```
TRACE #This ends with a comment.
   TRACE # and So does this!
```

Another useful commenting feature of MATVIEW is that round brackets, commas, and equal signs are *invisible* characters, allowing these characters to be mixed together freely in a command in place of blanks.

An example of how this is useful for increasing program readability is given by the use of these characters in the *POINT* command. The purpose of this command will be discussed later. For now we will consider it just as an illustration. In its basic form, this command requires three real numbers as arguments. For example:

POINT 3.0 4.0 6.6

Inserting appropriate *invisible* characters allows the user to customize the command as desired while increasing program readability. The above command may be expressed:

POINT (3.0,4.0)=6.6

In the event that the user's program contains an error, the *Trace* command may be inserted into the program to aid in locating the error. When the tracing facility is turned on, MATVIEW will print the line number of the user's program currently being read as well as the contents of the line. The tracing facility may be turned on and off at will by the user's program. The tracing feature may be turned on and off by the following commands Note that comment lines have been included to further demonstrate the use of comments in a MATVIEW program:

```
Trace On
# Now Matview will list the lines and line numbers
#      as it executes.

...

Trace Off
# Now Matview will resume running quietly without listing
#      lines and line numbers.
```

In the case of the trace command, only the first two characters are checked. By default, the tracing feature is off. If the trace command is given with no parameter, the parameter "on" is assumed. All of the following are correct for turning the tracing feature on:

```
TRACE
TRACE ON
   TRACE ON
Trace On
trace on
tracking on
trGARBAGE on
TRjunk
```

■ Program Debugging and Documentation Conveniences

The *Trace* command causes MATVIEW to turn the tracing feature off if the last letter in the command portion of the command line is an upper or lower case "F", and turns it on otherwise. As a result the following command will turn the tracing feature on:

```
Trace offandOn
```

while these will turn it off:

```
Trace Off
TRACE onandoff
Tracx garbage off
Trac garbage xxf
```

The preceding examples are intended to illustrate how flexible MATVIEW is in terms of how it interprets commands.

14
Sequencing of Commands

It is important that the user not direct MATVIEW to perform some task which depends on some piece of information which has not yet been specified by the user or computed by MATVIEW. There is no error checking done by MATVIEW with regard to this. As a result, MATVIEW allows a maximum flexibility in terms of the order in which commands can be specified, and further allows (within reasonable restrictions) the user to generate some output, then change some parameters before regenerating the same output for these new parameters.

The user must exercise caution when doing this. Certain commands must be done in a particular order initially. For example, once a problem has been properly set up, it is possible to change the ordering of the unknowns, and recompute the discretization matrix, without requiring that the region be redefined. On the other hand, once the grid size is modified, the region must be redefined. If the user wishes to modify the grid size, this must be done prior to defining the physical regions in the problem.

The ordering of the unknowns to be used in producing the discretization matrix must be specified prior to closing the definition with the *Close* command. If the user wishes to modify the ordering, a new *Close* command must be issued after the ordering has been altered.

Plotting may be done at any time, but will produce garbage results if the data structure to be plotted has not yet been defined or computed. A user may plot the original region if plotting is done prior to issuing the *Close* command. MATVIEW will remove grid points from the problem if they do not contribute to the solution once the *Close* command is issued. This is why a plot of the physical region made prior to closing may differ from the same plot generated once the definition is closed.

Sequencing of Commands

The *Close* command should be issued before attempting to plot the discretization matrix, and the *Solve* command should be issued prior to plotting the solution.

The following is a guide to aid the user in ensuring that commands are sequenced properly. The user is urged to read the preceding section carefully. The following list of rules is only intended to be a guide, and should not be solely relied as an indication of how commands should be sequenced. If the user has adequate understanding of what each command does, that is all that will be required to understand how commands should be sequenced. This information can be found by carefully reading the command description in the manual for each command.

This must be done ...	Before this
Select the ORDERING	define the physical region
Set the GRIDSIZE	define the physical region
Issue the CLOSE command	issue the SOLVE command
Issue the RESET command	define a new region

3 ■ Region and Boundary Definition Commands

The following chapter gives the details of how to go about describing a boundary value problem. This includes how to specify the locations of areas and points within the unit square representing the grid. As we discuss the use of MATVIEW, we will use heat flow as a model for our discussion. MATVIEW may be used, of course, for any type of flow problem. We will discuss how to describe areas or points as being insulated regions, regions of known temperature, and sources or sinks of heat flow.

We will first discuss how to provide the locational information, then we will describe how to provide the descriptive information. In an actual MATVIEW program both pieces of information together form a command line. It will be easier to first discuss each half of this command line separately, then describe the full command line in the last part of the chapter.

All parameters accepted by MATVIEW are with respect to the unit square. Regions may be defined as boundary conditions or may be given as X-Y conductivities. The location of the region being specified is independent of the type of the region. As a result, we may discuss how the location of a region is defined, independently of the type of region being defined. Consequently, we may also discuss how to specify the type of region being described, independently of where it is located. Commands for specifying the location and type of a region are of the form:

Location Description

where *Location* and *Description* are each given by a sequence of real numbers in the range of 0 to 1 and/or a sequence of MATVIEW commands.

Physical regions in MATVIEW are defined by the user as points or rectangular areas. The user must specify the size and shape of the area, and its position within the unit square. In defining each region the user must also indicate what type of region is being defined. A region may be defined as any one of:

1/ A region containing grid points which represent unknowns to be solved,
2/ An insulated region or boundary,
or, 3/ An area or boundary of known temperature.

The second item in the above list represents Neumann boundary conditions, while the last item represents Dirichlet boundary conditions.

These three types of regions are mutually exclusive. If an area is defined to be of one type, then a new region is defined which overlaps the original area which is of a different type, then the area of the overlap will be redefined to assume the characteristics of the new region. This allows the user to set up irregular geometries consisting of smaller rectangles scattered throughout a larger region. To do this, the larger region is defined first, then the contained regions are defined, thereby "over writing" the original definition.

If the region is too difficult to describe in terms of simple rectangles the user may provide suitable functions to specify the region accordingly. Using this method, a solution region of any shape may be defined (see the last section of this chapter).

In addition to defining the physical characteristics of the problem region, points and/or areas of constant external heat flow such as sinks and sources may be specified independently of the physical description of the region. Unless specified otherwise by the user, all grid points are assumed to have no external heat flow associated with them.

The remainder of this chapter will begin by discussing the *Location* commands followed by the *Description* commands. These are really only "half" commands, since both together are required to form a valid MATVIEW command. The use of these command parts together will be considered to define a region within a physical problem.

■ Location Commands

The first series of commands which will be discussed are designed to specify the size, shape, and position of the rectangular region being described. As the method for doing this is discussed, please bear in mind that it represents only "half" commands. The actual full commands used by MATVIEW consist of locational information together with descriptive information. In this section we will only be interested in the locational information. The symbols $\#X_1$, $\#X_2$, $\#Y_1$, and $\#Y_2$ denote positive real numbers in the range of 0 to 1 provided by the user as part of a command.

$\#X_1$ $\#X_2$ $\#Y_1$ $\#Y_2$

The simplest method to specify the location of a region is to use a sequence of four real numbers in the range 0 to 1. X1 represents the X co-ordinate of the left boundary of the region, X2 represents the right boundary of the region, Y1 represents the lower boundary of the region, and Y2 represents the upper boundary of the region. It may be easier to remember the order of the parameters as LEFT,RIGHT,BOTTOM,TOP. Below are several examples of using this method of locating a region:

0 1 0 1 = the entire unit square
0 1 0.5 1.0 = the top half of the unit square
0 0.5 0 0.5 = the bottom left corner of the unit square
0 0.5 0.5 1.0 = the bottom right corner of the unit square
0.45 0.55 0.45 0.55= a square 0.1 X 0.1 located in the centre of the unit square

■ Location Commands

BOUNDARIES

The BOUNDARIES location refers simultaneously to all four external boundaries of the unit square. It provides a convenient way of setting up boundary conditions on all external boundaries of the unit square of either Dirichlet or Neumann boundaries as specified by the descriptive information provided in the rest of the command.

NORTH $\#X_1$ $\#X_2$
SOUTH $\#X_1$ $\#X_2$
EAST $\#Y_1$ $\#Y_2$
WEST $\#Y_1$ $\#Y_2$
(Note: $\#X_1$, $\#X_2$, $\#Y_1$, and $\#Y_2$ are all optional for these cases)

These locations refer to individual boundaries of the unit square. If the real numbers shown are omitted, MATVIEW will assume that the entire boundary indicated is being referred to by the command. If the real numbers are present, the portion of the boundary indicated by the limits provided will be referred to. Here are some examples:

NORTH = the entire northern boundary of the square
WEST 0.5 1 = the top half of the western boundary of the unit square
SOUTH 0.33 0.66 = the middle third of the southern boundary of the unit square

REGION

The REGION location represents the entire internal region of the problem or the complete unit square. It is a convenient way of referring to the whole grid to set uniform conditions for all the grid points, or to set up an initial background area in which other regions will be later defined.

POINT #X #Y

The POINT #X #Y command is useful for specifying the location of specific points within the problem grid as source or sink points. For example the point in the middle of the unit square may be specified as POINT 0.5 0.5 using this location method.

When specifying a location using a sequence of four real numbers, MATVIEW assumes that the location is internal to the unit square unless the specifies a line segment rather than an area by setting either $\#X_1 = \#X_2$, or $\#Y_1 = \#Y_2$. If either of these cases occurs, MATVIEW assumes that the resulting line segment is a boundary. As a result, internal regions may contained mixed boundary conditions by defining the boundary area as a rectangle, then modifying the desired edges of it as line segments to a different type of boundary condition. This will be covered in more detail in the last example of Chapter 5. If a line segment specified is situated on an edge of the unit square, MATVIEW will assume that the user wishes to define an external boundary rather than points on the unit square.

When describing the type of region being defined, it is possible for the user to use either a real number represented as **#**, or a user function. The functions must be named after one of the following:

Fkx Fky Fdir Fsource.

If the user wishes to supply a function, the appropriate name must be used, and the compiled function must be linked with MATVIEW. All of these functions accept two double precision reals as arguments representing the X-Y co-ordinates of a point on the unit square, and return a double precision real result. Fkx and Fky return the conductivity of the region at the point X-Y in the X and Y directions respectively. Fdir returns the value of the known temperature of an X-Y point on a dirichlet boundary. Fsource returns the value of known external heat flow at the point X-Y.

In this section, the descriptive commands discussed will either allow a real number, or a function to be used interchangeably. This will be denoted as:

#/FUNCTION

which is intended to mean ONE of either **#**, or **FUNCTION**

#/FUNCTION #/FUNCTION

This specifies the conductivity in the X and Y directions respectively for the region being defined by the command at the grid point specified by the locational information. The parameters need not match one another, that is the conductivity of the region in the X direction may be supplied as a real number while the conductivity in the Y direction may be a function. The reserved name Fkx is used for the conductivity in the X direction, and the reserved function Fky is used similarly for the Y direction.

DIRICHLET #/FUNCTION

This describes a dirichlet boundary, that is a boundary of known temperature at the location given by the locational portion of the command. If the user wishes to use a function, a function named Fdir must be supplied.

NEUMANN

This specifies that an insulated boundary is to be specified at the location given by the locational information, that is, a Neumann boundary.

UNITY

This provides a short hand method for indicating that Kx=Ky=1 on the region indicated by the rest of the command. (Where Kx and Ky are the conductivity of the specified region in the X and Y directions respectively.)

ZERO

This provides a short hand method for indicating that Kx=Ky=0 on the region indicated by the rest of the command. (Where Kx and Ky are the conductivity of the specified region in the X and Y directions respectively.)

SINK #/FUNCTION

This describes a sink point. If a function is desired, it must be named Fsource. If the user supplies a real number, MATVIEW assumes that only the magnitude of the sink will be specified and not the sign. It will then negate the supplied value before recording it. NOTE: This is not true when a function is supplied. If a function is used it must supply both the magnitude and direction of the source or sink, since the same function will be used for both sources and sinks.

SOURCE #/FUNCTION

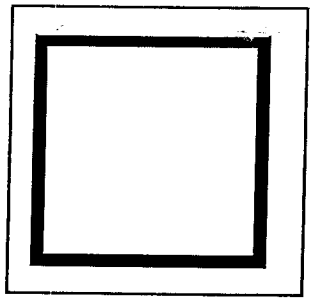
This is similar to the SINK command given above, with the exception that if a real number is supplied its sign will be left unchanged. Therefore the user may wish to use the SOURCE command with a negative argument instead of the SINK command. By the same token, the SINK command with a negative argument would actually specify a source.

■ Using Location and Description Commands to Define a Problem

In order to see how to use MATVIEW commands to define a problem, we will consider a number of examples. In each case, a command or a sequence of commands will be given, and a corresponding plots will be shown. In each case the plot will be labeled according to whether it is representing boundaries conditions, or regions of conductivity. Neumann conditions are shown in black, Dirichlet conditions are shown in a scale of grey according to the value of the dirichlet boundary specified, and grid points will be shown in white. Conductivity plots are show according on a grey scale according to the magnitude of the conductivity.

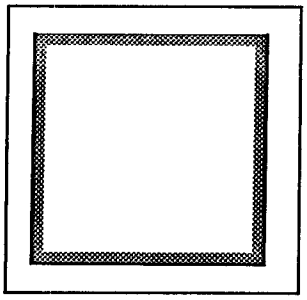
First only commands that effect the external boundaries will be shown. Following these, a plot showing an internal boundary will be given. In this plot, a large insulated area is placed in the middle of the grid, and a Dirichlet area in the lower left corner. Next, some sample commands to set up regions of conductivity will given. Finally, example commands will be given to set up some point sources and sinks and a source area.

BOUNDARIES NEUMANN
Here all four boundaries have been set to Neumann boundary condition. The corresponding plot shows these insulated regions in black. Grid points are shown in white.



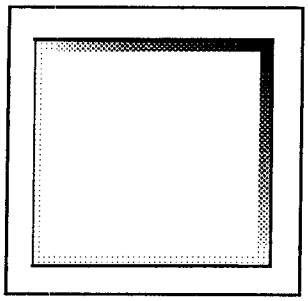
Plot of the Boundary Conditions

BOUNDARIES DIRICHLET 0.5
Here the command given above sets up all the boundaries as Dirichlet boundaries of value 0.5. Dirichlet boundary conditions are plotted in shades of grey relative to any other Dirichlet values present. Since there was only one Dirichlet value used, there is only one shade of grey.



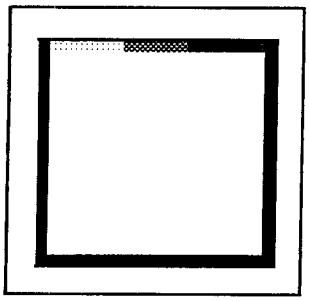
Plot of the Boundary Conditions

BOUNDARIES DIRICHLET FUNCTION
Here a Dirichlet function is used. The reserved function name *Fdir* was set to x*y for this plot. As the Dirichlet value varies, so does the level of Grey in the plot.



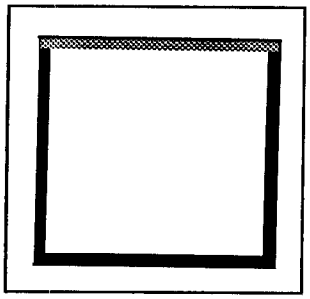
Plot of the Boundary Conditions

NORTH 0 0.33 DIRICHLET 0
NORTH 0.33 0.66 DIRICHLET 0.5
NORTH 0.66 1 DIRICHLET 1.0
SOUTH NEUMANN
EAST NEUMANN
WEST NEUMANN
Here different portions of the northern boundary are set up as three different Dirichlet Values. The remaining boundaries are set up as Neumann boundaries.



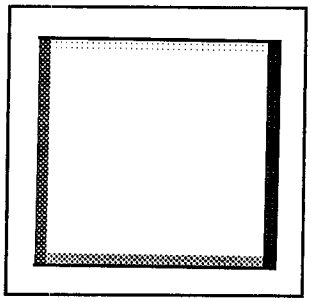
Plot of the Boundary Conditions

BOUNDARIES NEUMANN
NORTH DIRICHLET 0
Here all the boundaries are initially set up as Neumann boundaries. Since the NORTH command was given afterwards, its effect overwrites the northern boundary as a Dirichlet boundary.



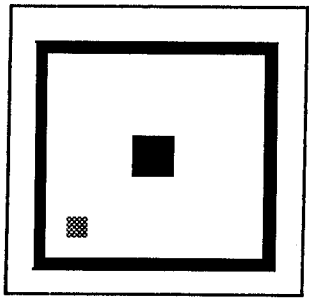
Plot of the Boundary Conditions

NORTH DIRICHLET 0
SOUTH DIRICHLET 10
EAST DIRICHLET 20
WEST DIRICHLET 30
Here Dirichlet Boundaries of higher values are set up to show how the grey scaling is relative to the magnitude of the values present.



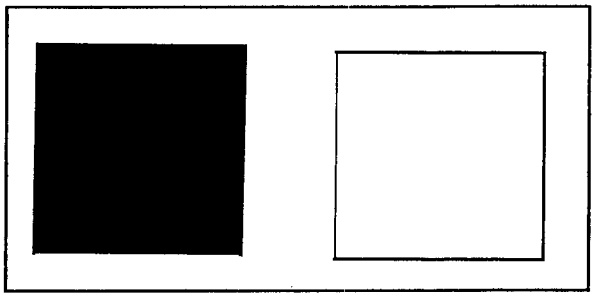
Plot of the Boundary Conditions

BOUNDARIES NEUMANN
0.4 0.6 0.4 0.6 NEUMANN
0.1 0.2 0.1 0.2 DIRICHLET 2.0
Here two areas representing internal boundary conditions are shown along with external boundary conditions. In place of the 2.0 which follows the keyword DIRICHLET, a function may also be used if desired.



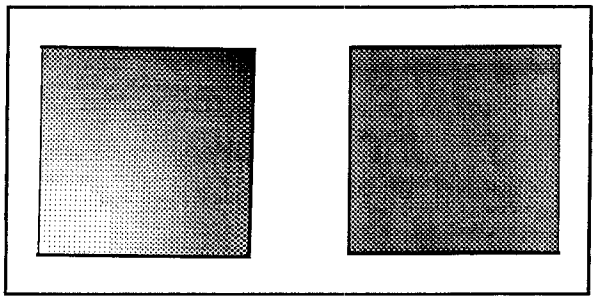
Plot of Internal and External Boundary Conditions

REGION 0.0 1.0
This command sets up the entire region with conductivity equal to unity in the X direction, and with no conductivity in the Y direction.



Conductivity in the X Direction (left) and in the Y Direction (right)

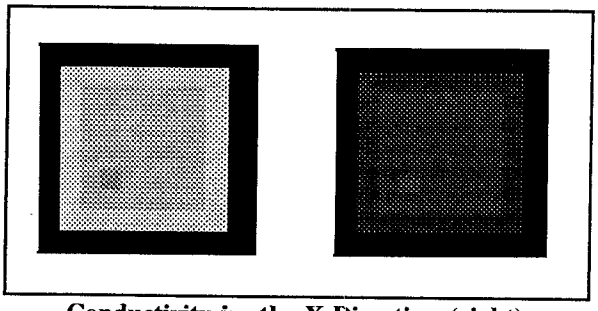
REGION FUNCTION 0.5
Here the reserved function *Fkx* was set to $Fkx=x^3+y^3$. This is the conductivity in the X direction. In the Y direction, the conductivity is 0.5.



Conductivity in the X Direction (left) and in the Y Direction (right)

REGION UNITY
0.1 0.9 0.1 0.9 0.9 0.1
0.2 0.8 0.2 0.8 0.8 0.2
0.3 0.4 0.3 0.4 0.7 0.3

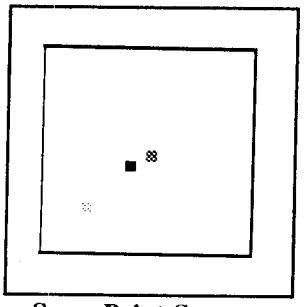
Here the whole region is initially set up with the conductivities in both directions equal to unity, then areas are overwritten with new conditions toward the centre.



Conductivity in the X Direction (right) and in the Y Direction (left)

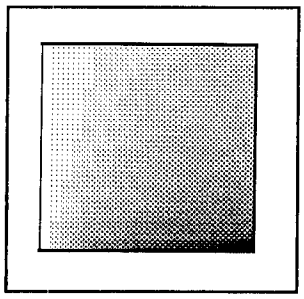
POINT 0.5 0.5 SOURCE 2.0
POINT 0.2 0.2 SINK 2.0
POINT 0.4 0.4 SINK 5.0

Here a number of sinks and sources are set up. Note that the second source could have been set up with POINT 0.2 0.2 SOURCE -2.0 if desired. Although it would be an unusual approach, the first point could have been set up as POINT 0.5 0.5 SINK -2.0 by the same token. In general, the point a user specifies will not necessarily coincide with a grid point. As a result MATVIEW always causes user specified points to shift to the closest grid point.



Some Point Sources and Sinks

REGION SOURCE FUNCTION
Here the reserved function name *Fsource* was set to Fsource=x/(0.1+y). Note that unlike the above example, SINK and SOURCE may be used interchangeably if a function is used. Therefore, the command REGION SINK FUNCTION has the same effect as the command REGION SOURCE FUNCTION.



Plot of a Source/Sink Area

32

■ Specifying an Irregular Solution Region

In order to understand how to specify an irregular solution, the user must first understand how MATVIEW represents the physical region of the problem internally. There is a data structure which represents the region conductivity in the X and Y directions. Insulated regions as well as Dirichlet regions are represented at a specific point by storing the value 999.999 for the value of the conductivity of the region in the X direction for that particular point. If the point is an insulated point, the conductivity in the Y direction is also represented as 999.999. If it is a Dirichlet point, the conductivity in the Y direction then represents this Dirichlet value rather than a conductivity.

If a user wishes to set up an irregular solution region, the areas of the unit square which do not include the region should be set at Insulation or as Dirichlet conditions. **NOTE:** The user must also set up the EXTERNAL boundaries separately by using either the *BOUNDARIES* command, or the *NORTH, SOUTH, EAST,* and *WEST* commands, **even if the solution region does not touch these boundaries.** This is because MATVIEW has no way of telling that you do not require these boundaries. If they are not needed, simply set them to insulation. Apart from using the above commands, the user has **no other control** over the external boundaries, that is boundaries outside the unit square.

For boundaries and unused regions the user does have control. Simply use the command "*REGION FUNCTION FUNCTION*", and write functions Fkx, and Fky which return either conductivity values for points in the solution region, or return the appropriate type and value of boundary conditions using the "boundary constant" 999.999 as described above. MATVIEW will then call the functions Fkx and Fky for each point of the discretization grid. Because this *is* a discretization, the user supplied function should return the appropriate boundary constants for an *area* outside the solution region to ensure that wherever the grid points fall, the correct boundary conditions will be realized. That is, the user functions should return boundary constants not only for points immediately outside the solution region, but for all points in an appropriate direction which fall outside the region.

4 ■ Action Commands

This chapter gives a description of the *Action* commands used to cause MATVIEW to perform calculations and generate results. There also a number of miscellaneous commands which fall into this group which are used to specify further details of the problem description as well as commands to annotate the output of the program and to maintain a title list of the plots generated.

As in the previous chapter, each command will be introduced with its unique minimal abbreviation in a larger typeface.

4 Action Commands

The CLOSE, END, and ECHO Commands

CLOSE

The CLOSE command indicates to MATVIEW that the region has been fully defined, the desired grid size has been selected, and the ordering of the unknowns has been chosen. When MATVIEW encounters a CLOSE command, it analyzes the physical description of the problem, removes unknowns which do not contribute to the solution, removes dirichlet boundary points from the unknown boundary points, and constructs a discretization matrix from the data. Once the CLOSE command has been executed, the region definition cannot be modified. A new ordering may, however be selected, and performing another CLOSE will then cause the new ordering to become effective.

END

The END command MUST be placed at the end of a MATVIEW program. It signifies that the program is finished. Without the END command present, MATVIEW will abort when reaching the end of the program, possibly without flushing some of the output generated during run time from its output buffer.

ECHO

The ECHO command will echo ALL remaining characters on the remainder of the command line into the output stream of the program. This includes any comment symbols and white space characters, with the exception that trailing blanks are ignored. A comment cannot be placed on the same line as an ECHO command, since it will simply be echoed to the program output. The ECHO command is useful in that it allows the user to annotate the output generated by MATVIEW.

■ The GRIDSIZE, RESET, SOLVE, and TITLE Commands

GRIDSIZE

The GRIDSIZE command is used to set the number of grid lines on the discretization grid. For this command, # must be an positive integer. The default grid size is 20 by 20 grid lines. The RESET command will reset the grid size to 20 X 20.

RESET (or RESTORE)

The RESET command causes MATVIEW to reset all internal data structures and variables to the initialized state. This allows a program to start over, re-defining a new region. It is almost like a new run of MATVIEW except that the title list of the plots generated thus far is not cleared, and the file number of the next plot to be generated is also maintained. MATVIEW continues executing at the next command line following the RESET command.

SOLVE

The SOLVE command causes the sparse discretization matrix to be solved. This must be done prior to plotting any data structures which depend on this.

TITLE

The TITLE command is very similar to the ECHO command, in that the remainder of the line with trailing blanks removed is echoed to the MATVIEW output. The difference here is that the string echoed is prefixed with a message indicating the plot file number of the next plot to be generated. MATVIEW produces plot files as a sequence of FORTRAN files starting with fort.21 with the number increasing by one for each new plot. If the next plot to be generated is fort.33, then the command:

TITLE This is a plot of the boundaries.

will cause MATVIEW to echo the following line to the output:

fort.33 : This is a plot of the boundaries.

PLOT *argument*

The PLOT command with no argument will cause MATVIEW to generate a plot of the boundary conditions of the physical region. If an argument is supplied, the plot generated will depend on the argument. The *argument* may be any one of the following:

BOUNDARIES	**GREEN #X #Y**	**INVERSE**
KX	**KY**	**MATRIX**
ORDERING	**PGREEN #X #Y**	**PINVERSE**
PMATRIX	**SINKS**	**SOLUTION**
SOURCES	**TRUE SOLUTION**	

Care must be taken by the user to ensure that MATVIEW has already computed the quantity to be plotted. These plots will be discussed individually in alphabetical order in the remainder of this chapter.

The Chapter on "How to Use MATVIEW" discusses how to actually output the plots produced by MATVIEW to an output device.

■ Plotting BOUNDARIES, GREEN #X #Y, and INVERSE

PLOT BOUNDARIES

BOUNDARIES is the default argument for the plot command. This plots all boundary conditions both on the external boundaries of the grid as well as any internal boundary conditions. Neumann conditions are shown in black, grid points are shown in white, and Dirichlet boundary conditions are shown as shades of grey according to the relative magnitude of the Dirichlet conditions present.

PLOT GREEN #X #Y

This command requires that a pair of real numbers be supplied representing an X-Y co-ordinate. A row of the inverse of the discretization matrix is computed. The row is given by the ordering number of the point located at the point specified in the physical grid by X and Y. This command assumes that the SOLVE command has already been issued. The inverse row is plotted, with each element mapped back into its corresponding point in the physical grid. The plot formed is known as the discrete Green function. The plot is done in shades of grey according to the absolute value of each element.

PLOT INVERSE

This command plots the full inverse of the discretization matrix. If the grid size is large, this command could generate a large number of files. Each file will be printed as a single sheet of paper. The order that the sheets are printed is first left to right, then top to bottom. Care should be taken in determining whether or not to use this command on a large sized matrix. MATVIEW will support the output of an arbitrarily large plot divided onto individual sheets of paper, so the responsibility rests with the user to decide how many pages of output are acceptable.

■ Plotting KX, KY, MATRIX, ORDERING, PGREEN and PINVERSE

PLOT KX, and

PLOT KY

These commands plot the conductivity in the X and Y directions respectively. This is scaled in shades of grey according to the relative magnitude of the different conductivities present in the region.

PLOT MATRIX

This plots the discretization matrix. The absolute value of each element is scaled in grey. Refer to the similar command PLOT INVERSE above for more details.

PLOT ORDERING

This command plots the ordering number of each unknown point on the grid scaled in shades of grey where the darkest point is the one with the highest ordering number. This command assumes that the CLOSE command has been executed.

PLOT PGREEN #X #Y

This command is similar to the PLOT GREEN #X #Y command given above except that rather than a row of the true inverse of the discretization matrix, the corresponding row of the inverse of the incomplete Cholesky factorization (precondition matrix) is used. Therefore the term "Precondition Green", or PGREEN for short, is used to describe it.

PLOT PINVERSE

This command will produce a plot of the inverse of the precondition or incomplete Cholesky factorization matrix. A full matrix is plotted much in the same way as in the case of the PLOT MATRIX and PLOT INVERSE commands above.

■ Plotting PMATRIX, SINKS/SOURCES, SOLUTION, & TRUE SOLUTION

PLOT PMATRIX

This command will produce a plot of the precondition or incomplete Cholesky factorization matrix. A full matrix is plotted much in the same way as in the case of the PLOT MATRIX and PLOT INVERSE commands above.

PLOT SINKS , or
PLOT SOURCES

Either of these commands will plot all the sinks and sources on the grid scaled in shades of grey according to the relative value at each point compared to the other points.

PLOT SOLUTION

This command assumes that the SOLVE command has been executed. It is similar to the PLOT KNOWN command given above, except the solved solution rather than the true solution is plotted.

PLOT TRUE SOLUTION

If the true solution is known, and the reserved function *Ftrue* is modified to return the true solution at the point X,Y, then this command will generate a plot of the true solution, scaled in grey, and mapped onto the appropriate grid points. This is useful for comparison between the MATVIEW solution and the true solution for verification purposes.

■ Selecting the Desired ORDERING of the Unknown Grid Points

ORDERING *argument*

The ORDERING command is used to select one of many different predefined orderings for the unknown grid points. If no argument is given, or an unrecognizable argument is given, row, or natural ordering will be used. This is also the default ordering in the event that the ORDERING command is not issued. If a new ordering is to be selected, it must be done prior to issuing the CLOSE command. **NOTE: The argument for an ordering MUST be in uppercase.**

Below is a list of the available orderings. A full description of the details of each ordering will not be given here, but may be found in [DUFF88].

ORDERING **NATU**RAL

Natural or row ordering

ORDERING **CUTH**ILL

Cuthill McKee (1969) ordering

ORDERING **RCUT**HILL

Reverse Cuthill McKee, George (1971) ordering

ORDERING **BCUT**HILL

Block Cuthill McKee, Duff (1988) ordering

■ Selecting the Desired ORDERING of the Unknown Grid Points

ORDERING REDBLACK

Red black ordering

ORDERING ALTDIAG

Alternating Diagonal ordering

ORDERING- ZEBRA

Zebra ordering

ORDERING NESTED

Nested Dissection ordering

ORDERING OWD1

One way dissection (1 level)

ORDERING OWD2

One way dissection (2 level)

ORDERING SPIRAL

Spiral ordering

ORDERING FOURCOLOR

Four colour ordering

■ Selecting the Desired ORDERING of the Unknown Grid Points

ORDERING VDV1

Van der Vorst ordering (Version 1)

ORDERING VDV2

Van der Vorst ordering (Version 2)

ORDERING UNION

Union Jack ordering

ORDERING LRCO

Localized row/column ordering

ORDERING ALANGEORGE

Alan George ordering

ORDERING WPTANG

Wei-Pai Tang ordering

■ Adjusting and Selecting a Grey Scale for Plots

The grey scale used by MATVIEW may be adjusted by the user. This feature is particularly useful when the user desires to draw a meaningful comparison between several plots. In such cases, it is important that all the plots use the same grey scale. Up to 100 grey scales may be set by the user, either manually, or automatically according to the maximum and minimum values in a particular matrix.

SCALE *scale-number scale-range scale-mode*

The SCALE command is used to adjust the limits used for grey scaling a plot. The default scale used is "SCALE 0". If the SCALE command is used without a specific scale number as a parameter, the parameter "0" will be assumed. The scale number may be specified as 0 to 99 thereby allowing 100 scales to be set and used. The scale used for plotting is the last scale set or referred to by the user.

The default range and mode for SCALE 0 (which is the default scale) is "AUTOMATIC" and "ABSOLUTE". Therefore, if plots are generated without first setting the scale, these are the parameters that will be used. We will now discuss the ranges and modes for grey scales which may be set by the user. Whenever the SCALE command is used, the selected scale is set according to the "range" and "mode" specified by the user. If these parameters are not supplied, MATVIEW will assume that the scale has previously been set by the user, and use those values. If the scale has not been set, the default values will be used. The arguments to the SCALE command for each of the parameters above are:

Number	Range	Mode
#	**A**UTOMATIC	**A**BSOLUTE
	FIXED	**R**ELATIVE
	# #	**C**ENTRED

■ Selecting the RANGE for a Grey Scale

These parameters may be combined in any combination desired by the user. **N.B.** If no scale number is specified, then SCALE 0 will be assumed. **All of the following commands apply to SCALE 0 through to SCALE 99 equally.** Each range and mode will be discussed separately, however, in the actual usage of the command, the keyword "SCALE" followed by the scale-number, scale-range, and scale-mode will all be present together on a single command line to form the desired scaling command.

We will begin by discussing the "range" portion of the SCALE command. It will be difficult to discuss the "range" portion independently of the "mode" portion, since the appearance of the plot depends on both of these parameters. As a result, we will discuss the range command only in terms of two values that it produces, a MAX and a MIN. **The user is not expected to understand the relationship between the matrix, MAX, MIN, and the grey scaling until the "mode" command is discussed.**

AUTOMATIC

This is the default range. The matrix to be plotted is scanned to determine the largest and smallest values present. These values become the parameters MAX and MIN respectively. The exact appearance of the plot will depend on the mode that has been selected.

FIXED

This range is similar to the "AUTOMATIC" range, except the values MAX and MIN are set according to the next matrix that is plotted. These values are remembered and reused on other plots made under the same scale.

#

Here two real numbers are supplied. The first is the parameter MIN and the second is the parameter MAX. It is the responsibility of the user to ensure that MIX < MAX.

■ Selecting the MODE for a Grey Scale

Now the "mode" portion of the SCALE command may be discussed. Here the parameters MIN and MAX described above will be used to determine the appearance of the plot.

ABSOLUTE

In this mode, the elements in the matrix are plotted in shades of grey according to their absolute values. Zero is represented by white on the plot. The largest absolute value of MAX and MIN determines the value that will be represented by black. If the absolute value of an element is beyond this black value, black will be used for it. That is, values whose absolute value is larger than the parameters set will "top-out" at black.

CENTRED

In this mode, the grey scale is not uniform. The value MIN is represented by white, while the value MAX is represented by black. Zero is represented by a shade of grey which is exactly midway between black and white. As a result all the negative values are scaled uniformly between white and mid-grey, while all the positive values are scaled uniformly, but possibly according to a different scale, from mid-grey to black. This mode is useful for viewing the sparsity pattern while preserving an indication of the sign of the values in a sparse matrix. Values which are too large or too small for the parameters set will either "top-out" or "bottom-out" at black or white respectively.

RELATIVE

In the relative mode,the grey scale is uniform. The value MIN is represented by white, while the value MAX is represented by black. Intermediate values are represented by a uniform shading of grey. Values which are too large or too small for the parameters set will either "top-out" or "bottom-out" at black or white respectively.

46

■ Recording the MIN and MAX values used for a Grey Scale

When the range of a grey scale has been set according to either the FIXED or AUTOMATIC modes, it may be useful for the user to know what range and mode were used for the scale. There is a "SCALE OUTPUT" command that MATVIEW provides in which these parameters are printed out. The values last used with the scale are printed to the standard output device along with an explanatory message. The current mode of the scale is also printed. This command is used as follows:

SCALE *scale-number* **OUTPUT**

If the scale-number is omitted, then SCALE 0 is assumed.

Below are several examples of how the SCALE command may be used. Since no scale-number is present, all of these commands refer to SCALE 0.

SCALE 1.2 3.5 RELATIVE

Values less than or equal to 1.2 will be shown in white, while values greater than or equal to 3.5 will be shown as black. Intermediate values will be uniformly scaled in shades of grey.

SCALE AUTOMATIC ABSOLUTE

The absolute values of the elements will be shown uniformly in shades of grey from white, for zero, to black, for the highest absolute value in the quantity to be plotted.

SCALE FIXED CENTRED

Negative values will be scaled uniformly from white for the lowest value to mid-grey for zero. Positive values will be uniformly scaled from mid-grey for zero to black for the highest value. The highest and lowest values will be determined by the next matrix plotted. These values will be remembered for subsequent plots.

■ Inputting an Externally Generated Matrix

From time to time a user may desire to use MATVIEW only for its ability to represent a matrix. In such cases the user will have a matrix generated by some other program and will desire to import it into MATVIEW for displaying. The *INPUT* command will allow this to be done. Three different types of matrix structures may be input to MATVIEW by the user, as well as a vector representing the "Right Hand Side" of a matrix equation for which the unknowns are to be solved. If any physical plots are desired, the user must fool MATVIEW into thinking that it set up the matrix, by using the commands discussed in Chapter 3. The purpose of this is to indicate where the unknowns are in the physical region. Since MATVIEW recognizes unknowns by the fact that the conductivities are not "boundary constants", the easiest way to do this is simply use the following sequence of commands:

GRIDSIZE *n*
REGION UNITY
BOUNDARIES NEUMANN

then for internal points or areas not in the solution region, (including Dirichlet points) use:

POINT X Y NEUMANN, or X1 X2 Y1 Y2 NEUMANN

as appropriate to indicate that the point or points in the area are not unknowns.

Next select the desired ordering using the *ORDERING* command, and issue a *CLOSE* command. Now input the matrix using the description given below. If desired, a "Right Hand Side" may also be input, in which event the SOLVE command can be used to solve the problem.

Note: These steps are necessary only if a plot mapped onto a physical region is desired. If only plots of the matrix or its inverse are desired, these steps are not required.

■ Inputting an Externally Generated Matrix

INPUT

This command accepts one parameter indicating the type of matrix to be input, followed by the elements of the matrix. Each format is discussed below. In ALL cases, the elements in the matrix may be separated by blanks or commas, and the lines may contain comments. The number of elements per line is immaterial. MATVIEW continues reading as many lines as is necessary to input the indicated number of elements. **Note:** the line length of each line must be no more than 80 characters. Extra characters will be ignored.

INPUT FULL N

or

INPUT FULL FILE *filename*

Here "N" is the order of the matrix to be input, and "N" squared elements will then follow, with both zeros and non-zeros present. For example to input the following 3 X 3 matrix, the command show will be used. Note that in this example the matrix is input on two lines to emphasize that the number of lines used is immaterial. If the FILE option is used, the input will be taken from the user file specified by *filename*. In this case the input format is as described above, except that the first line of the file must contain a single integer representing N.

2 4 3
1 0 5
9 8 7

The above matrix may be input as follows:

INPUT FULL 3
2 4 3 1 0 5 9
8 7

Inputting an Externally Generated Matrix

INPUT SPARSE N M

or

INPUT SPARSE FILE *filename*

Here "N" is the order of the matrix to be input, and "M" is the number of non-zeros. The "M" non-zero elements follow next, then "M" integers in the same order as these elements indicating which column each of the non-zeros was in. Finally, N integers follow indicating the number of the element which begins each of the N rows of the matrix, then a final integer whose value is M+1 to complete the pattern. The example below should clarify this. The number of lines used for the input is immaterial, but each of the three component structures must begin on a new line. If the FILE option is used, the input will be taken from the user file specified by *filename*. In this case the input format is as described above, except that the first line of the file must only contain two integers representing N and M in that order.

2	**3.3**	0	0	0
1	**4**	0	0	0
0	**5**	**7.2**	0	0
0	0	**4.4**	**6**	**9**
0	0	0	0	**8.1**

```
INPUT SPARSE 5 9
2 3.3 1 4 5 7.2 4.4 6 9 8.1
1 2 1 2 2 3 3 4 5 5
1 3 5 7 10 11
```

INPUT BANDED N L U

or

INPUT BANDED FILE *filename*

Here "N" is the order of the matrix to be input, and "L" and "U" are the number of elements in the lower bands and upper bands of the matrix respectively. The elements of the matrix follow one row at a time in column order. If a zero is in the band, it must be present in the input data. If the FILE option is used, the input will be taken from the user file specified by *filename*. In this case the input format is as described above, except that the first line of the file must only contain three integers representing N, L, and U in that order.

9	**2**	**1**	0	0
3	**8**	**2**	**0**	0
0	**7**	**5**	**6**	**8**
0	0	**9**	**8**	**7**
0	0	0	**4**	**5**

INPUT BANDED 5 1 2

9 2 1 3 8 2 0 7 5 6 8 9 8 7 4 5

INPUT RHS N

or

INPUT RHS FILE

Here "N" is the number of equations in the matrix for which the right hand side is given. The values in the vector follow on an arbitrary number of lines, with no more than 80 characters per line. The order must be according to the row of the matrix with which the value is associated. If the FILE option is used, the input will be taken from the user file specified by *filename*. In this case the input format is as described above, except that the first line of the file must contain a single integer representing N.

5 ■ How to Use MATVIEW

In this chapter, we will cover the various procedures that a user will need to learn in order to use MATVIEW. The user must write a MATVIEW program using the MATVIEW language described in this manual. **N.B.** MATVIEW accepts its input from stdin (standard in). The user should therefore redirect stdin to come from the file containing the user's program. The program may be prepared using any standard text editor. It is assumed that the user is familiar with using a program editor to do this.

We will begin by discussing how to run a MATVIEW program. Next we will discuss how to combine several plots together, and how to output a plot to a laser printer. Finally, we will cover the subject of linking user functions into MATVIEW.

---

5 How to Use Matview

■ Running MATVIEW Programs

As was noted in the introduction, MATVIEW accepts a program written in the MATVIEW programming language from stdin. A user may enter commands one line at a time either for experimental or for diagnostic purposes. In general, however, it will be much more useful for a user to redirect stdin to come from a file containing a MATVIEW program. To run the a user program called "myprog", execute the command "matview" at the operating system prompt and redirect stdin as shown:

```
matview < myprog
```

As plots are generated, one file for each page of each plot will be generated. These files will be named starting with fort.21, and continuing up to fort.999 as required, depending on the number of plots generated. Since the plots are not identified by the system, the user may wish to use the *Title* command described earlier to keep track of which plot belongs to which filename.

All non-graphics output from MATVIEW is directed to stdout (standard out), and may be redirected by the user to a file as desired. In the example above, any text output by MATVIEW will go to the users terminal device. In order to have this output stored in a file, stdout may be redirected. The output is sent to a file called "outfile" in the following example:

```
matview < myprog > outfile
```

■ Outputting and Combining Plots

The graphics output produced by MATVIEW will not be able to be sent directly to an output device. There are a number of utilities such as the *IM tool kit* which are documented elsewhere which will assist in this. The IM tool kit may be used to change the output files generated by MATVIEW into a format compatible with graphics output devices. As this manual focuses on MATVIEW, the reader may wish to consult other sources to gain information on the subject of producing plots from the graphics output generated by MATVIEW. This toolkit is documented fully elsewhere, [PAETH86] but sufficient information will be given here to allow the user to generate plots from MATVIEW.

When a plot is generated it is stored in a file as described earlier. A number of IM tools are required to generate output to a laser printer from these files. In order to facilitate this procedure, the user should place the following alias into a start up file for the account being used. MATVIEW is supplied with this alias in a file called "matview.alias". Note that the underlined portions of this alias represent device dependent portions of the command. These parts of the alias will need to be modified according to the output device being used.

```
alias plot set IMPARMS = `wc !*`;set IMLINES=`echo $IMPARMS | sed "s/^ *\([^ ]*\)
.*/\1/"`;set IMCOLS=`echo $IMPARMS | sed "s/^ *\([^ ]*\)  *\([^ ]*\).*/\2\/\1/"
| bc`; imtabin  -w $IMCOLS -h $IMLINES -p n8 <!* | imexpr -e  1-n | imsample -s
16 | imhalftone dot8 | immargin -m 4 -v 0 | imcanw -spool -x 800 -y 500 | lpr
-Pic
```

This alias allows the user to simply type:

```
plot fort.33
```

at the operating system prompt in order to generate a plot from fort.33 .

Plots of the same size may be combined to form one larger plot using the *combine* utility. **Note: In general boundary plots are larger than Kx, Ky, Green, and PGreen plots.** The following example combines three plots into a new one called "bigplot". A black boarder is placed around each individual plot as a separation within the new file.

```
combine fort.21 fort.22 fort.23 > bigplot
```

Linking User Functions

If the user desires to modify any of the reserved functions discussed in Chapter 3, it will be necessary to use the FORTRAN compiler to recompile these functions, and the linker to link the other MATVIEW object modules together with the new object modules of the modified functions.

There is one object module for each reserved function. The default functions simply return a value of 1.0 when called. Therefore nothing is lost if these are erased by the user, however, they will have to be replaced by the user's version of the function. In the following example, we assume the user had written a new function for returning the conductivity in the X-direction. This function must be named "Fkx" by the user, and must be contained in a file named "Fkx". Valid source code must still exist for all the other reserved functions even if the user will not be using them. At the unix prompt the following should be typed:

```
f77 -o matview matview.o Fkx.o Fkx.o Fdir.o Fsource.o Ftrue.o
```

6 ■ Sample MATVIEW Programs

This chapter gives sample MATVIEW programs for each of the four versions of the popular *Stones* problems. The reader unfamiliar with this set of problems should consult [STONE68]. In each case, the MATVIEW program is commented to explain what each command line does. Recall that the following characters:

= , ( )

are all ignored by MATVIEW and may therefore be used freely to make the code more readable. These characters are treated as blanks. Recall also the use of comments indicated by the "#" character.

Each program is followed with the plots it generated. This should aid the user in understanding what each program does. The chapter begins some typical discretization problems, and then gives examples of the Stones problems. Next, an example where a source function is applied to the entire region is given. For this example a solution is actually generated and plotted. The true discrete Green function is plotted as well as the equivalent plot resulting from using the inverse of the preconditioning matrix. The final example includes an internal boundary area with multiple boundary conditions.

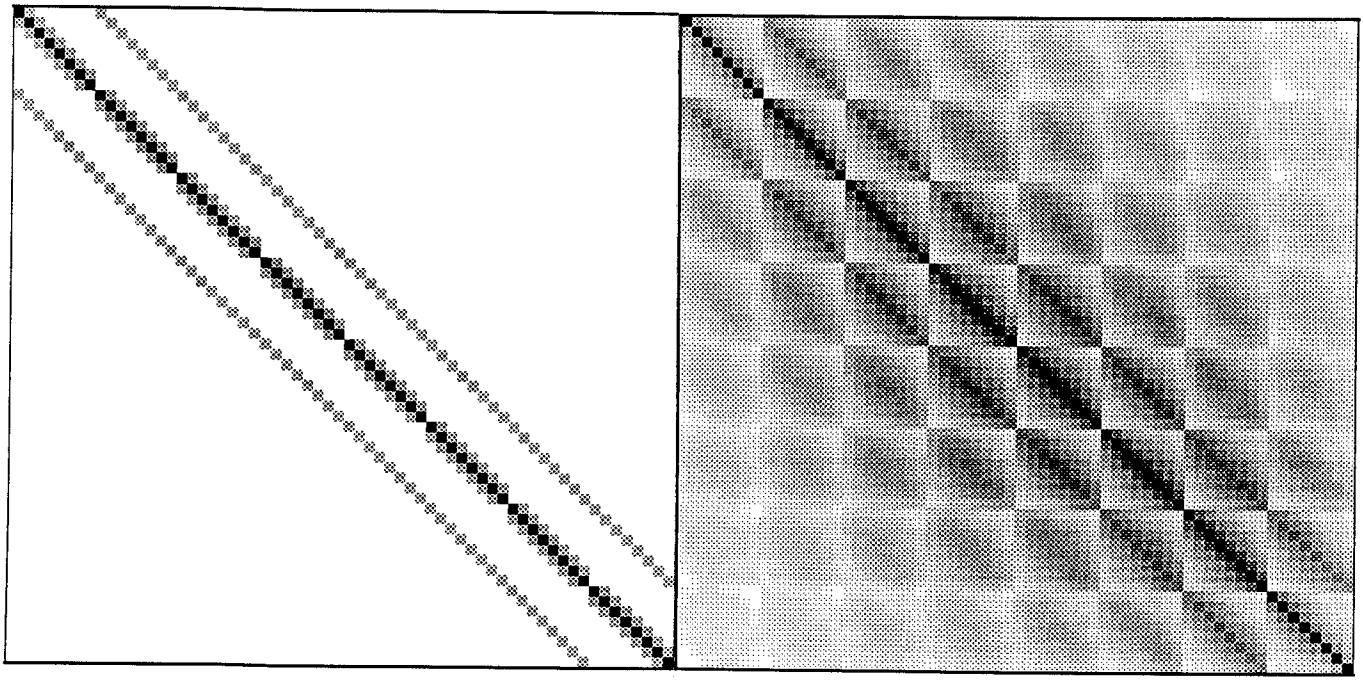
---

6 Sample Matview Programs

The Model Problem

```
# The Model Problem
# Here we show the discretization matrix
#  of the model problem and its inverse

GRIDSIZE=10
BOUNDARIES DIRICHLET 0
REGION UNITY
CLOSE
PLOT MATRIX
PLOT INVERSE
CLOSE
```

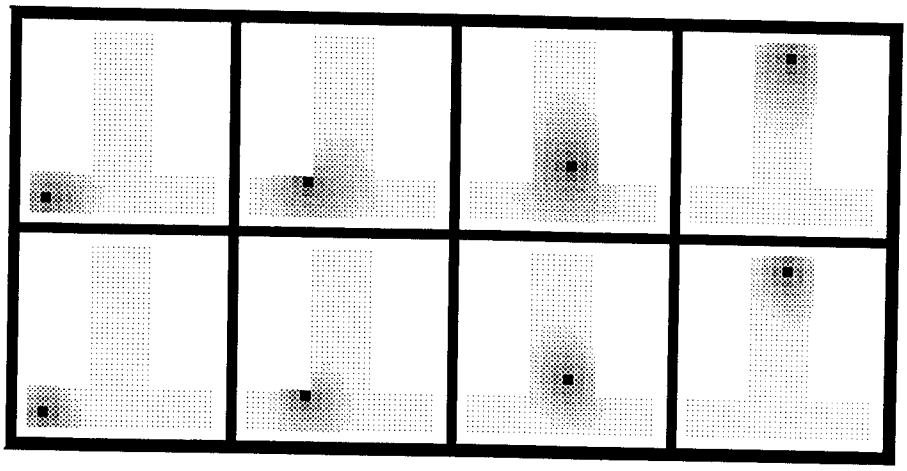
The two output plots generated by this program appear below.



A "T" Shaped Solution Region

```
# The "T" shaped problem
# Here we show the discrete Green function and the Preconditioned
#  equivalent for several points on a "T" shaped solution region.

BOUNDARIES DIRICHLET 0
REGION UNITY
0 0.3 0.3 1.0 DIRICHLET 0
0.7 1.0 0.3 1.0 DIRICHLET 0
CLOSE
PLOT GREEN(0.1,0.1)
PLOT GREEN(0.3,0.2)
PLOT GREEN(0.5,0.3)
PLOT GREEN(0.5,0.9)
PLOT PGREEN(0.1,0.1)
PLOT PGREEN(0.3,0.2)
PLOT PGREEN(0.5,0.3)
PLOT PGREEN(0.5,0.9)
END
```

The output plots generated by this program appear below. The plots of the discrete Green function are shown on the top, while the preconditioned version of each is shown below. These plots were combined using the "combine" command discussed in Chapter 5.

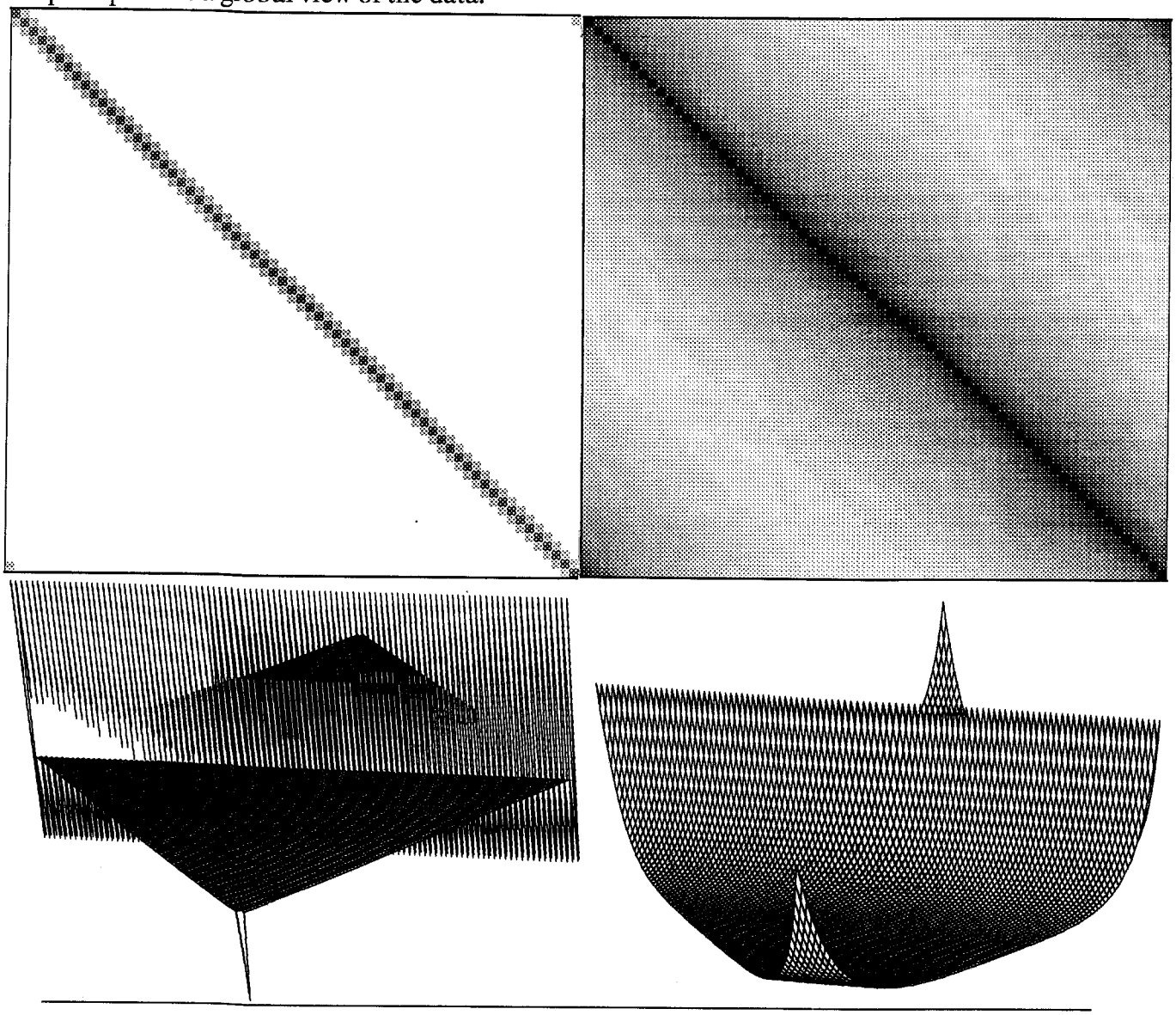


58

A Periodic Boundary Problem

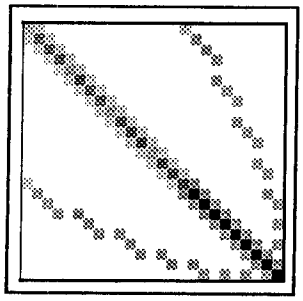
```
# A periodic boundary problem. Here we input an externally generated
#  discretization matrix and plot the matrix and its inverse.
INPUT FULL FILE periodic.problem
PLOT MATRIX
PLOT INVERSE
END
```

The output generated is shown below. Here, it is compared with the equivalent plots produced by a three dimensional visualization tool (shown under each MATVIEW plot). Note that there are many hidden features in the three dimensional representation. The MATVIEW plots present a global view of the data.



■ Stones Problem Version 1

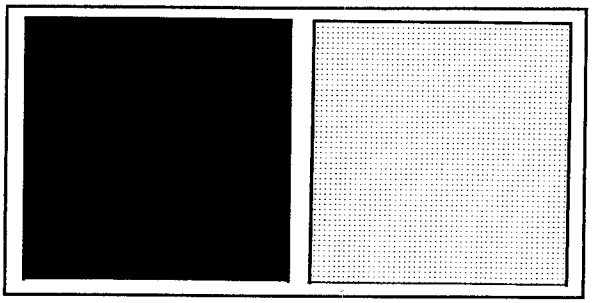
```
# Stones version 1 problem
# Since this version of the stones problem is so straightforward,
#     we will only plot the discretization matrix for it. We
#     will use a very coarse gridsize so that the plot
#     will fit comfortably onto this page.
#      The Stones problems are typically used as a test problem set
#      for  testing solutions for discretization matrices derived from
#     the discretization of Second Order Partial Differential Equations.
#      (see [STON68])

# First we set the gridsize
Gridsize=5

# Next we set the entire region to Unity
Region=Unity

#Now we will set up the external boundaries as dirichlet boundaries
BOUNDARIES DIRICHLET 0

# Since the discretization is not affected by sinks and sources,
#     we will not bother to set any up.

# Next we will select spiral ordering.
Ordering=SPIRAL

# Now we will echo a message to the output mentioning the order
#     that we selected.
Echo Spiral Ordering has been selected.

# Now tell Matview that were done defining the
problem.
Close

# Now we are all set to generate a plot
#     which we will title appropriately.
Title Stones version 1 Discretization Matrix
Plot Matrix # Here we plot it.

End
```



Plot of the Discretization Matrix for Spiral Ordering

Stones Problem Version 2

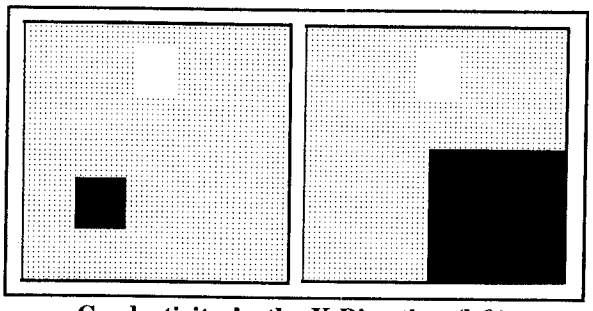
```
# Stones version 2 problem
# Since this version differs from version 1 only in that the
#     the Kx and Ky values differ, we will only plot the
#     regions of conductivity.

# First we set the gridsize
Gridsize=25

# Next we set the entire region to Kx=100, Ky=1
Region  100  1

# Next we set up the boundaries appropriately. This step isn't
#     really necessary since we will only be plotting Kx and Ky
Boundaries Dirichlet 0

# We will not need to select an ordering, or to "close"

# Now we are all set to generate a plot
PLOT Kx
PLOT Ky

END
```



Conductivity in the X Direction (left) and in the Y Direction (right)

Stones Problem Version 3

```
# Stones version 3 problem
# This version is a bit more interesting as it has an
#     irregular geometry. We will plot the regions
#     of conductivity only.

Gridsize=25
Boundaries Dirichlet 0

# First we'll set up the whole region as unity, then
#     overwrite the changes.
REGION UNITY

# now region B
0.48 1.0 0.0 0.52 1.0 100

# next region C
0.18 0.38 0.18 0.38 100 1.0

# lastly region D
0.42 0.62 0.72 0.92 ZERO

# Now we are all set to generate a plot
PLOT Kx
PLOT Ky

END
```



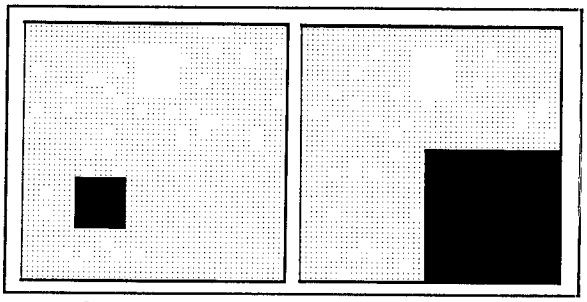
Conductivity in the X Direction (left) and in the Y Direction (right)

Stones Problem Version 4

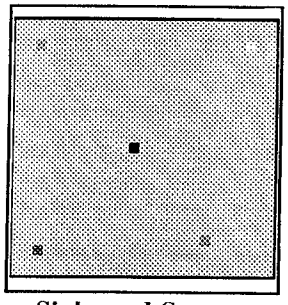
```
# Stones version 4 problem
# For the final stones problem, we will set up the problem
#      in its entirety, and produce a number of plots
#      of the physical details.
#      Note that this problem has the same geometry as
#      The previous one, except that Region A is set up
#      with random values of Kx and Ky. The values randomly
#      produced are set to zero if they are below 0.1 in
#      accordance with the specifications of the stones problem.
#      The function Fkx and Fkx were set up to return these values.

Gridsize=25
Boundaries Dirichlet 0

#Note that since F is the minimally unique abbreviation for FUNCTION
#      in Matview, it will be perfectly acceptable to use the specific
#      function name in its place since they all start with 'F'.
#      This will make the code much more readable.

# region A first
REGION Fkx Fky # (meaning "REGION FUNCTION FUNCTION")

# now region B
0.48 1.0 0.0 0.52 1.0 100

# next region C
0.18 0.38 0.18 0.38 100 1.0

# lastly region D
0.42 0.62 0.72 0.92 ZERO
```

■ Stones Problem Version 4

```
# Now we are all set to generate some plots
PLOT Kx
PLOT Ky

# Now we set up the sources. Note that the parentheses, commas, and equal
#     signs are all just for looks!
Point(0.1,0.1)=Source,1.0
Point(0.1,0.9),Source=0.5 # maybe this way looks nicer?
POINT 0.77 0.133 SOURCE 0.6 # this is the plain way.

#Next are sinks. We will set up one as a source with a negative flux rate.
Point(0.466,0.5)=sink 1.83
Point(0.9,0.9)=source -0.27

# Lets plot the sources/sinks
Plot sources
# and now we're all done
end
```



Conductivity in the X Direction (left) and in the Y Direction (right)



Sinks and Sources

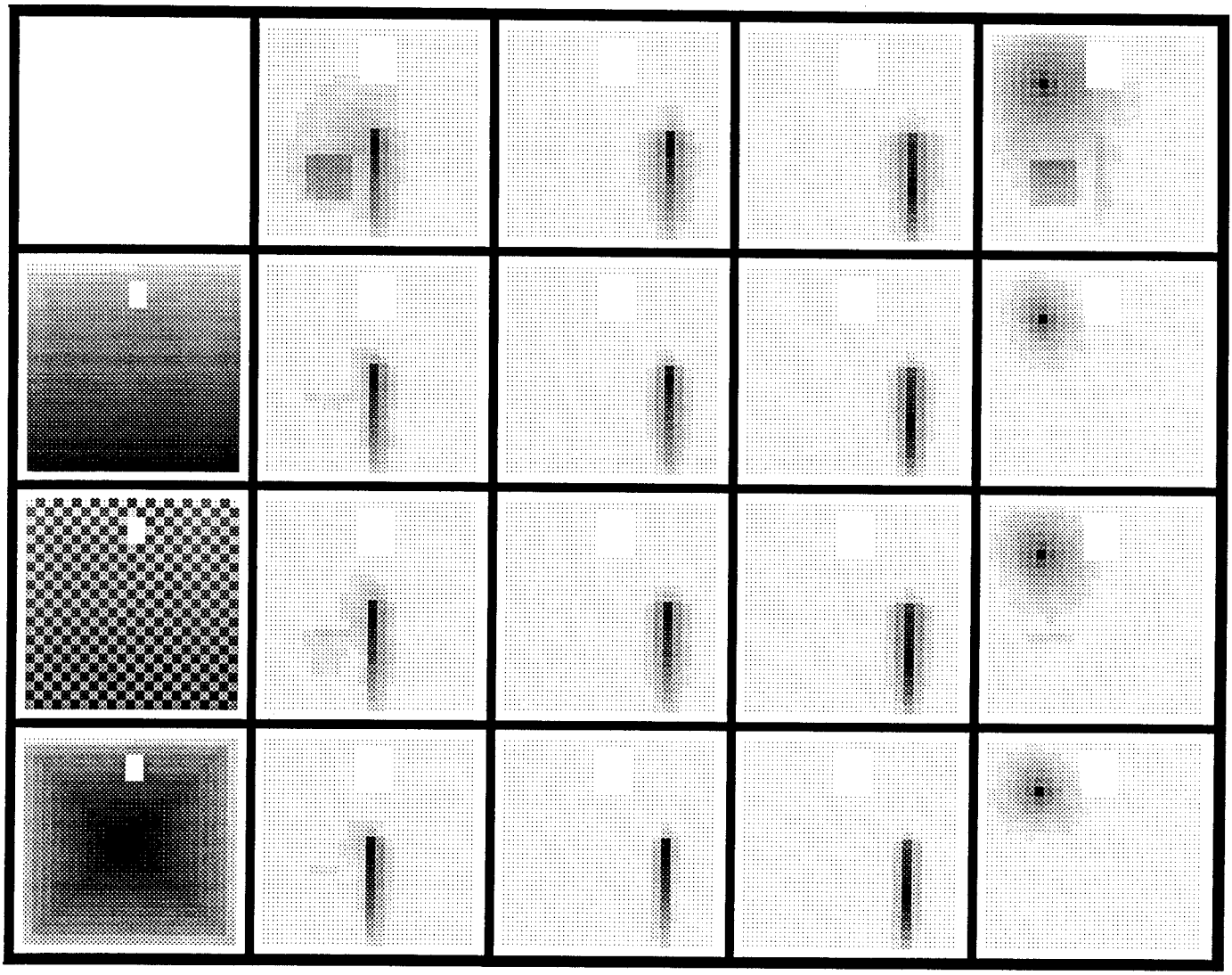
64

■ Some Sample Output From One of the Stones Problems

```
# Some Sample Output from the Stones Problem Version 3
Gridsize=25
# Make a blank filler plot
REGION ZERO
PLOT KX
Boundaries Dirichlet 0
REGION UNITY
0.48 1.0 0.0 0.52 1.0 100
0.18 0.38 0.18 0.38 100 1.0
0.42 0.62 0.72 0.92 ZERO
CLOSE
Plot Green(0.5,0.5)
Plot Green(0.75,0.5)
Plot Green(0.75,0.25)
Plot Green(0.25,0.75)
Plot Ordering
Plot Pgreen(0.5,0.5)
Plot Pgreen(0.75,0.5)
Plot Pgreen(0.75,0.25)
Plot Pgreen(0.25,0.75)
Ordering REDBLACK
CLOSE
Plot Ordering
Plot Pgreen(0.5,0.5)
Plot Pgreen(0.75,0.5)
Plot Pgreen(0.75,0.25)
Plot Pgreen(0.25,0.75)
Ordering SPIRAL
CLOSE
Plot Ordering
Plot Pgreen(0.5,0.5)
Plot Pgreen(0.75,0.5)
Plot Pgreen(0.75,0.25)
Plot Pgreen(0.25,0.75)
End
```

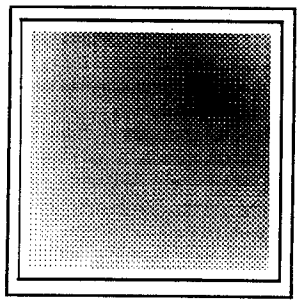
The output plots generated by this program appear below. They were combined using the combine command along with a blank plot for filler in the top right corner. The top row of plots consists of the true discrete Green function for a number of points as shown. The first plot in each of the remaining rows shows the ordering of the variables, and the remaining plots in each row are the Preconditioned template corresponding to the Green function above for each of these orderings.

■ Some Sample Output From One of the Stones Problems



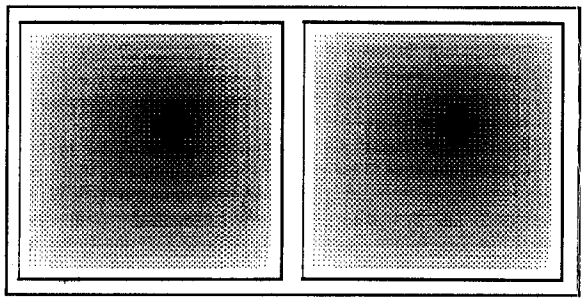
A Solved Example

```
# Regular geometry with source function
# Here the reserved functions were set as follows:

#     Fsource = (-sin(y)*exp(sin(x)+sin(y))*(x*x-x)*(y*y-y)
#     +cos(y)*cos(y)*exp(sin(x)+sin(y))*(x*x-x)*(y*y-y)
#     +2*cos(y)*exp(sin(x)+sin(y))*(x*x-x)*(2*y-1)
#     +2*exp(sin(x)+sin(y))*(x*x-x)
#     -sin(x)*exp(sin(y)+sin(x))*(y*y-y)*(x*x-x)
#     +cos(x)*cos(x)*exp(sin(y)+sin(x))*(y*y-y)*(x*x-x)
#     +2*cos(x)*exp(sin(y)+sin(x))*(y*y-y)*(2*x-1)
#     +2*exp(sin(y)+sin(x))*(y*y-y))

#     Ftrue=exp(sin(x)+sin(y))*(x*x-x)*(y*y-y)

Gridsize=25
Boundaries Dirichlet=0
Region Unity
Region Source=Fsource
Plot Sources
Close
Solve
Plot Solution
Plot True solution
Plot Pgreen(0.5,0.5)
Plot Green(0.5,0.5)
end
```

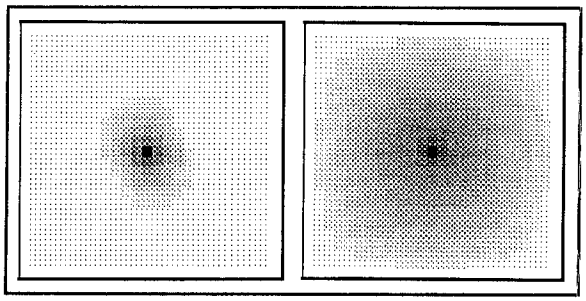


The Source Function

A Solved Example



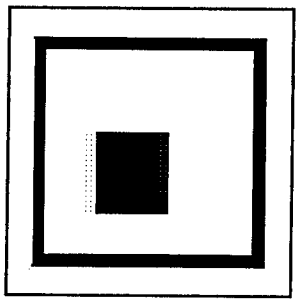
True Solution (left) and Solved Solution (right)



Template of Cholesky Inverse (left) and True Inverse (right)

■ Multiple Internal Boundary Conditions

```
# This program sets up some complicated boundaries
Boundaries Neumann
0.2 0.6 0.2 0.6 neumann
0.2 0.2 0.2 0.6 dirichlet 0
0.6 0.6 0.2 0.6 dirichlet 10
plot boundaries
end
```



External Neumann Boundary Conditions and Mixed Internal Boundary Conditions

6 ■ MATVIEW Command Summary

Below is a summary of the MATVIEW command set organized by command type and listed in alphabetical order.

Action Commands	Description Commands	Location Commands	Functions
CLOSE	DIRICHLET	BOUNDARIES	Fdir
ECHO	FUNCTION	EAST	Fkx
END	NEUMANN	NORTH	Fky
GRIDSIZE	SINK	POINT	Fsource
INPUT	SOURCE	REGION	Ftrue
ORDERING	UNITY	WEST	
PLOT	ZERO	#	
RESET	#	# #	
SCALE	# #		
SOLVE			
TITLE			
TRACE			

Input Arguments	Ordering Arguments	Plot Arguments	Scale Arguments
BANDED N L U	NATURAL	BOUNDARIES	ABSOLUTE
FULL N	CUTHILL	GREEN	AUTOMATIC
SPARSE N M	RCUTHILL	INVERSE	CENTRED
RHS N	BCUTHILL	KX	FIXED
	REDBLACK	KY	OUTPUT
	ALTDIAG	MATRIX	RELATIVE
	ZEBRA	ORDERING	#
	NESTED	PGREEN	# #
	OWD1	PINVERSE	
	OWD2	PMATRIX	
	SPIRAL	SINKS	
	FOURCOLOR	SOLUTION	
	VDV1	SOURCES	
	VDV2	TRUE SOLUTION	
	UNION		
	LRCO		
	ALANGEORGE		
	WPTANG		

7 ■ References

[DUFF88]
DUFF, I. S., AND MEURANT, G. A.
"The effect of ordering on preconditioned conjugate gradients,"
CEA, Centre d'Etudes de Limeil-Valenton, **September 1988**

[KFOR89]
KIGHTLEY, J. R., AND FORSYTH, P. A.
"MAT1 - Iterative Sparse Matrix Solver: User's Guide,"
Technical Report 3, **University of Waterloo, 1989**

[PAETH86]
PAETH, A. W.
"The IM Raster Toolkit Design, Implementation,and use,"
Technical Report, University of Waterloo, **December 1986**

[STONE68]
STONE, H. L.
"Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations,"
SIAM Journal of Numerical Analysis, **September, 1968**

[TANG87]
TANG, W. P.
"Schwarz Splitting and Template Operators,"
PhD thesis, **Stanford University, 1987**