



2. The SATURN Suite – An Overview

Mini-Contents Page

| | | |
|-----------|---|------------|
| 2. | The SATURN Suite – An Overview | 2-0 |
| 2.1 | The Structure of Assignment Models | 2-1 |
| 2.2 | Trip Matrices in SATURN | 2-3 |
| 2.3 | Networks in SATURN | 2-4 |
| 2.4 | Route Choice in SATURN | 2-5 |
| 2.5 | Analysis in SATURN | 2-6 |
| 2.6 | General Advice on Using SATURN | 2-6 |
| 2.7 | Getting Started; Example Files | 2-7 |
| 2.8 | Text Files, Fixed Columns, Text Editors and Word Processors | 2-8 |
| 2.9 | Errors and Warnings: Fatal, Non-Fatal and Serious | 2-10 |
| 2.10 | Version Control | 2-12 |

INTRODUCTION

This section is intended, firstly, to introduce some of the functions and capabilities of **SATURN** to new users and, secondly, to illustrate how this manual may be used and where certain pieces of information are likely to be found.

It follows, roughly speaking, the authors' standard "Introduction to **SATURN**" presentation at meetings, courses etc., and - we would like to think - such presentations may provide a slightly more user-friendly intro to **SATURN** than meeting the documentation "cold". Equally reading an article such as that in Traffic Engineering and Control, 1980 (see reference 1 in Appendix C) may provide an easier starting point.

Alternatively there are a number of introductory **SATURN** courses available within the UK, including ones run by the Institute for Transport Studies and independently by Atkins; details available from Dirck Van Vliet (D.V.V.) or Ian Wright (I.W.) as appropriate.

2.1 The Structure of Assignment Models

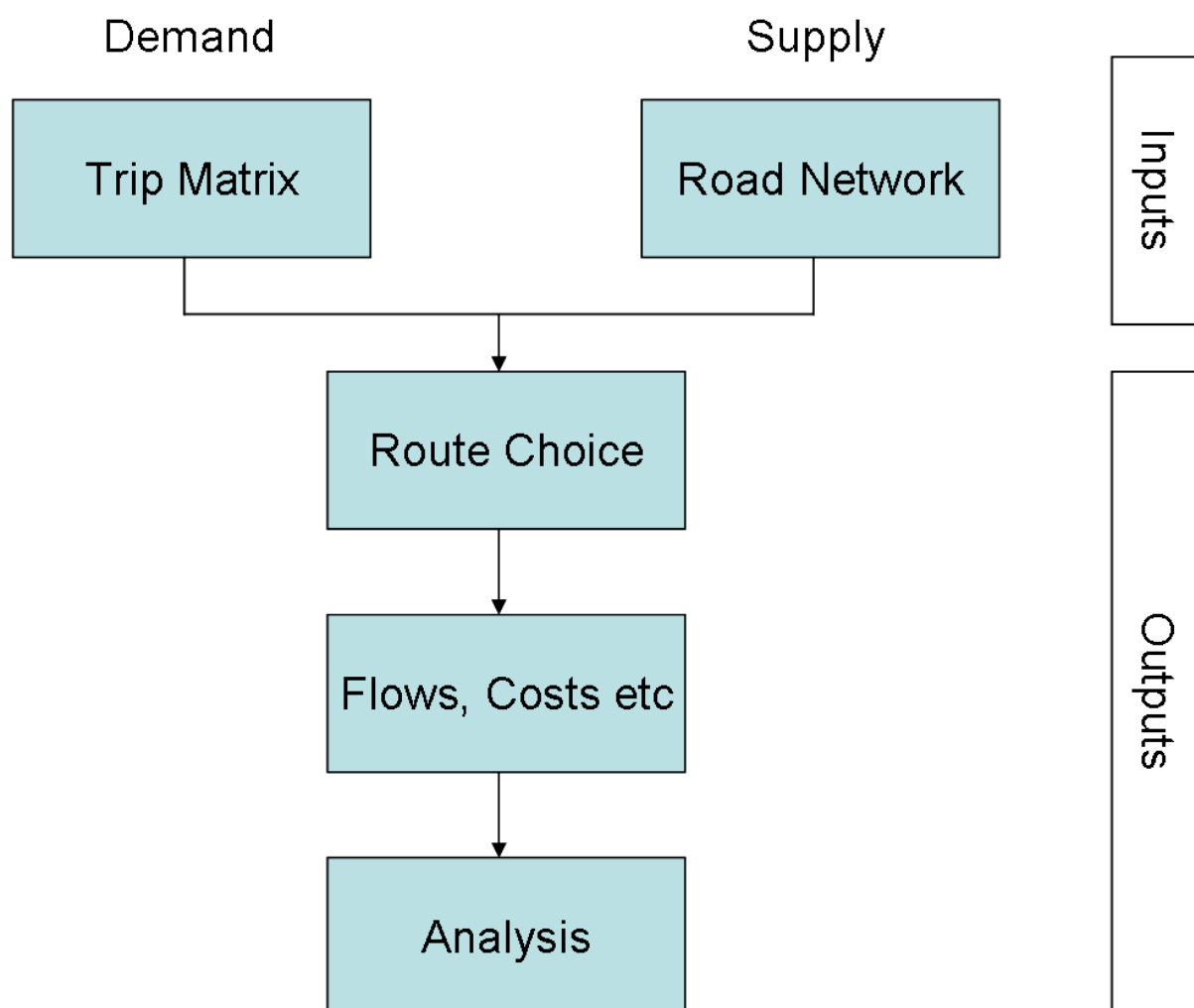
There are clearly lots of things that you can do with **SATURN**, some of which were described in Section 1. However to most users **SATURN** is primarily a traffic assignment model, the general structure of which is illustrated in Figure 2.1.

Thus there are two general inputs - the "trip matrix" T_{ij} which specifies the number of trips from zone i to zone j ; and the "network" which specifies the physical structure of the roads, etc. upon which trips take place. These may be thought of as the "demand" and "supply" inputs; it is the job of the user to provide these inputs.

Both the matrix and network are input to a "route choice" model which allocates trips to "routes" through the network, as a result of which total flows along links in the network may be summed and the corresponding network "costs" (e.g., times) calculated. This is essentially where **SATURN** takes over, operating of course under instructions from the user.

Once the assignment has been carried out the user may then "analyse" the resulting flow pattern and, in effect, ask the **SATURN** programs for whatever information they require.

Figure 2.1 General Structure of an Assignment Model



The above structure is of course perfectly general and may be applied to any suite of network analysis programs, not just SATURN. What distinguishes different suites is, e.g., the precise manner in which data must be prepared for input, the options that are available to undertake route choice or to analyse the results, etc. However one important common strand is to note that errors in the outputs can arise from four general sources

- ◆ errors in the trip matrix,
- ◆ errors in the network specification,
- ◆ errors in the route choice model specification (e.g., the definition of generalised cost), and
- ◆ errors in the numerical calculations (i.e., non-convergence).

Effectively errors (i) to (iii) are down to the user to correct, whereas any problem in the numerical solution of the problem specified by the user (i.e., non-convergence) is largely the responsibility of **SATURN**. (Although – see Section 9.5 – there are a lot of things users can do to minimise problems of convergence.) Clearly the

programs should - and do - assist users to minimise input errors, while the users should choose the appropriate options within the assignment routines.

The important point is that you should not necessarily blame the program if the results are not to your liking - the universal computing principle of “Garbage in, Garbage out” applies very much to the use of **SATURN**. This is not to say that the **SATURN** programs are perfect - far from it, as you shall no doubt find out! - but it is important to realise that there are a host of reasons why models fail and to learn - from experience, largely - how to minimise them.

A second important point to appreciate is that the programs in the **SATURN** suite are only tools capable of carrying out certain well-defined functions and it is the user who must specify how these tools are to be used. If you treat **SATURN** as a black box which operates in whatever default mode seems appropriate to us, the designers, then it may very well not do what you want it to do. You are the boss and the more you understand the underlying modelling principles which **SATURN** follows then the better your results will be.

The following sections explain in greater detail how the various components in Figure 2.1 are handled within **SATURN** and where relevant questions are treated in the documentation.

2.2 Trip Matrices in SATURN

There are, in general, two distinct schools of thought related to trip matrices

- ◆ that they should be obtained (as far as possible) from direct observation such as in-vehicle interviews or number-plate matching surveys;
- ◆ that they should be derived from some form of demand modelling process, e.g. trip distribution, modal split etc

A related issue is how one obtains future year forecasts given a validated base year matrix. Again there are two extreme solutions, simple growing-up or factoring methods or cost-related demand models (as discussed in Section 7.4).

In practice of course the techniques used in studies will incorporate elements of all the above procedures.

Every organisation using **SATURN** must therefore develop their own procedures for matrix estimation, e.g., from vehicle surveys, home interviews, and/or demand models. This, it must be stressed, is never an easy task and may involve even more resources, and possibly even more computing resources, than the network-based tasks.

Facilities within the **SATURN** Suite may be used to a greater or lesser extent in the estimation of matrices. At one extreme **SATURN** may be run with fixed trip matrices derived totally independently of **SATURN**. On the other hand there are a large number of matrix-based programs within **SATURN** which can assist in the derivation of trip matrices. Thus the general matrix manipulation program **MX** (see Section 10) contains virtually all the standard operations required to create trip matrices, for example matrix factoring routines to model growth in particular zones. In addition the **SATME2** program, described in detail in Section 13, uses link count data to estimate the “most likely” trip matrix. The elastic assignment

options (see Sections 7.4 and beyond) also allow the basic assignment function within **SATURN** to be interfaced with more general demand models such that **SATURN** can include, e.g., modal split or distribution facilities.

However for most users the creation of a trip matrix will involve the use of the “matrix build” procedure **M1** which converts a data file of individual o-d trips prepared by the user into a file suitable for handling within **SATURN**. How to use **M1** in this manner is described in Section 4.

We note, in passing, that it is important to remember that the recommended units for flows in **SATURN** are **pcus/hr**, not vehicles/hr (although it is possible to use vehicles if desired; see 15.17.1), so that it may be necessary to convert observed vehicle trip matrices into pcu matrices **prior** to running **SATURN**.

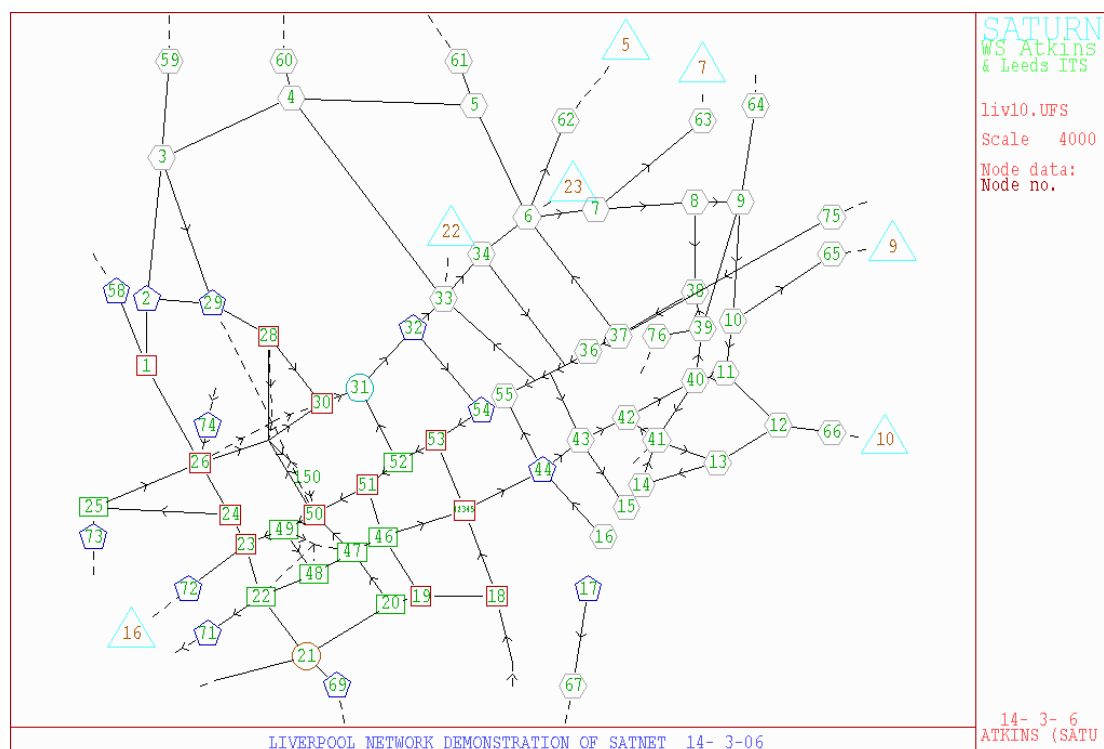
Finally, let us re-emphasise the point made in 2.1 that errors in trip matrices can have a significant effect on the overall accuracy of the full assignment procedure and that they should not be either ignored or under-estimated

2.3 Networks in SATURN

Networks on the other hand are central to **SATURN** and the way in which they are described and manipulated is one of the properties that distinguishes **SATURN** from other assignment models.

An example of a “typical” **SATURN** network (as output by program **PIX**) is given in Figure 2.2.

Figure 2.2 - A Typical SATURN Network



Note firstly that SATURN networks may be coded at two levels of detail:



- ◆ as an “inner” or “simulation” network in which considerable junction-based data in addition to road-based data must be provided by the user; and
- ◆ as a “buffer” network, normally surrounding the simulation network, which is coded in the more conventional sense of only requiring data to describe the roads as opposed to the junctions.

Thus in Figure 2.2 those nodes which are represented as hexagons, i.e. near the outside of the network, are buffer nodes and the links connecting them constitute the buffer network.

Junctions in the central area - rectangles, circles and squares - are the simulation junctions; the different shapes denote different junction types (see 11.6.5.1)

Note as well the presence of “zones” or “centroids” represented by triangles which are common to both the simulation and buffer networks.

Typically the simulation network is used to describe networks at the centre, say, of a traffic management scheme where the impacts are crucial and large, while the buffer network is used to describe, e.g., the inter-urban roads surrounding a town where the impacts of traffic management schemes are less critical. Very often, as in Figure 2.2, the full network resembles a “doughnut” with the simulation network in the middle.

It is not, however, necessary for every network to use both a buffer and simulation component. Large-scale studies of, say, inter-urban networks may only employ buffer networks, while very localised studies may set up a pure simulation network.

An early stage in any study is to decide upon the extent - and the type - of the network to be modelled. Following this you will need to obtain the data required to define the network(s) and to prepare appropriately formatted data files for entry into the **SATURN** network build procedures (i.e. into program **SATNET**).

More complete details on how networks are described are given in Section 5; in particular Section 5.6 provides a general introduction to the science - or art? - of building networks. New users should read this in full before moving on to Section 6 which describes the detailed formatting conventions used in preparing a network data file. You only need to read those parts of Section 6 which relate to your particular data inputs; for example only look at Section 6.9 if you wish to code bus routes.

Note as well that a lot of the drudgery associated with coding networks may be minimised by making use of the interactive network building and editing facilities in **PMAKE**; see Section 18. However, as with trip matrices, coding a network can be a long and involved process and there are very few shortcuts.

2.4 Route Choice in SATURN

In contrast to the definition of the networks and trip matrices where the onus is decidedly on the user to do hard work, the route choice in **SATURN** is handled by the computer programs. However the user will still have several important decisions to make; e.g



- ◆ whether to use an “equilibrium” or “stochastic” framework;
- ◆ how to define “cost”
- ◆ how many iterations, etc, to allow

Details on the options available and the theoretical background to the choices available are contained in Section 7 of the Manual. It should, however, be stressed that this documentation is not intended to provide a complete coverage of assignment theory; for that users should consult standard text books such as those written by Yosef Sheffi of M.I.T. or Roy Thomas of Salford

2.5 Analysis in SATURN

Once the assignment is complete - or indeed at any stage during the process - there is a wealth of **SATURN** analysis programs designed to allow the user to “interrogate” the system in order to discover not only WHAT the output values are but also WHY they take the values they do. A central precept of **SATURN** is that any internal processes should be transparent to the user.

The analysis programs may be used either to display information on the screen or to send it to a “line printer” file for external printing or viewing. Screen output may be either “alpha-numeric” output or, wherever possible, a “graphical display”.

Section 11 of the manual describes both the general features of the main interactive program **PIX** plus certain specific features of sub- programs such as **SATDB** or **SATLOOK**. However the best way to appreciate what can be done and how is simply to “browse” through **P1X**, using the on-line “help system” when the succinct menu entry does not fully convey the function of the option.

2.6 General Advice on Using SATURN

Firstly, **SATURN** is (obviously!) a suite of computer programs so users must (obviously!) be able to use a computer. This is not to say that you have to be a computer whizz kid in order to use **SATURN** - in fact you can get by with fairly basic computer skills - but the more you know about computers, the better. For example, you may be able to make the data input stages much simpler by using spread sheets to code matrix data or CAD packages to digitise networks.

As a minimum you must understand how files are named, how to manipulate them (e.g., to move files between different directories, etc.), how to use a text editor in order to prepare data files and how to examine output files. You do not, on the other hand, need to know anything about programming in order to run the programs.

Sections 3.3 to 3.6 of the manual describe the basic computer conventions within **SATURN**; read these at an early stage. Section 3 also includes an introduction to **SATWIN** which should be the normal point of access to all things **SATURN**.

Secondly, it is very important to read the relevant sections of the manual before embarking on any **SATURN** job. For new users finding their way around the manual is not that easy but the Table of Contents at the beginning and an Index at the end should make life a bit easier. Certainly Sections 1 to 5 should be read at a very early stage, Sections 7, 8 and 9 (assignment, simulation and their



combination respectively), slightly later but certainly before you become involved in any “serious” work. Equally read Section 19.1 and the relevant subsections in 11 before using any of the interactive programs such as **PIX** and remember to use the “Help Facilities” with these programs.

Finally, you will find a slightly warped sense of humour and a knowledge of who’s who on The Muppets to be useful assets! Oh yes, and don’t get upset by the insults.

2.7 Getting Started; Example Files

New **SATURN** users, particularly those operating with only this manual as a starting point, may find difficulties in establishing exactly what **SATURN** provides. To assist them - and, indeed, all users - two “standard” data files are provided

- ◆ a network file “Epsom98net.dat”; and
- ◆ a matrix file “Epsom98mat.dat”.

Thus, having read in all the files and set up the appropriate directory structure (which should be done for you by the Installation procedures) and wishing to run a sample **SATURN** job, you must first create a trip matrix file (see Section 4). This can be done through the Command Line (in **SATWIN**) by typing:

M1 Epsom98mat

followed by the command:

SATURN *Epsom98net Epsom98mat*

to assign the trip matrix to the *Epsom98net* network (see Section 3.1).

This process is also described in Section 3.6 where the facilities for running **SATURN** modules from **SATWIN** are detailed.

When the above run is complete, you will find that a number of files have been created, e.g., a file *Epsom98net.lpn* which is a line printer output file from the **SATURN** program **SATNET** which converts the *Epsom98net* network data file into a “binary file” *Epsom98net.ufn* and a *Epsom98net.lpt* output file from **SATALL**. Line printer files may be either printed or viewed in a standard text editor. File name conventions are described in Section 3.3.

All essential outputs from the assignment are now stored in the file *liv97.ufs* and the results may be analysed by a set of analysis programs. To experiment try the following commands

P1X *Epsom98net*

SATLOOK *Epsom98net*

MX *Epsom98mat*

Details on the functions of the above programs may be found elsewhere in the manual.



Please note however that the *Epsom98net* file is NOT intended as an illustration of good coding practice; merely an example of some network coding. Equally, although superficially similar to the Epsom town centre from which it was derived, the actual data is almost totally artificial.

2.8 Text Files, Fixed Columns, Text Editors and Word Processors

SATURN programs make extensive use of “ascii” or “text” files which users need to prepare in order to run **SATURN** programs, e.g. the network .dat files used to define network structure as input to **SATNET**.

2.8.1 Fixed Column versus Comma Separated Variables (CSV)

Very often **SATURN** data files are based on “fixed column” conventions; i.e., a particular piece of input data **must** appear within particular columns of an input record; if it appears in the correct order but outside the required columns it will be misread. Specific column rules are documented throughout the Manual; see, in particular, Section 6 for network .dat file conventions.

An alternative convention is that of CSV or “Column Separated Variables” where the order of the variables is specified but the “width” of each is not and each variable is separated and distinguished from its neighbours by a comma (or a blank).

CSV is a more modern convention than fixed column and, while fixed columns is generally the standard format throughout **SATURN** data files, CSV is very often available as an alternative.

One advantage of fixed column inputs (DVV’s viewpoint!) is that it is much easier to spot mistakes visually in files, e.g., if a value in a particular column is out of range, whereas with a CSV file it is very difficult to distinguish the 5th from the 6th variable on a particular line. Another is that a blank column or field is always read as “zero” and need not be explicitly input as a zero whereas with CSV files all values need to be explicitly included, if only by an extra comma; this allows data fields to be “added” in unused columns at later stages of development whilst maintaining upwards compatibility with “old” data files. On the other hand fixed columns set limits on the “range” of values which may be input so that, for example, it is not possible to define a value of 12345 in 3 fixed columns; with CSV this is not a problem.

For you brought up in the Bill Gates era of MS Excel etc. etc. fixed column inputs may well seem strange – get used to it!

2.8.2 Text Editors and Word Processors

Generally speaking it is NOT a good idea to prepare such files using “word processing” facilities as opposed to “text editors”. The reason is that very often word processors - for very good reasons as far as they are concerned! - insert special, non-printing characters into files such as tab characters in order to give the desired fixed-column appearance on the screen. Unfortunately, such characters are virtually always misinterpreted by a FORTRAN read statement, particularly when data files are in “fixed format”, i.e., data items must appear within specific columns. Do not be misled by the fact that your file “looks” correct - it aint! Fortunately, some Text Editors (but not MS Wordpad or MS Notepad) are



capable of removing these 'hidden' tab characters and replacing them with spaces (eg UltraEdit and Textpad for example).

This is not to say that it is impossible to use a word processor to create a **SATURN** data file, but you will have to choose your text specifications with care! For example using MS Word you need, at a minimum, to use .txt outputs (which, following **SATURN** conventions, should subsequently be renamed as .dat files). Similarly, when exporting information from MS Excel, worksheets should be saved as "Formatted Text (Space Delimited) (*.prn)" files with the appropriate column widths. For example, a data file requiring data in columns 1 to 5 may be exported in the correct format by setting the width of Column A in the worksheet to 5.0cm (ie a one cm column is the equivalent of a single space in a text file).

2.8.3 FORTRAN Conventions for Reading "Real" Data Fields

Within FORTRAN numerical variables may be either "integer" or "real"; e.g., 1, 2, 3 as opposed to 1.25, 3.14159, etc. etc. It is important in preparing SATURN .dat files that when an **integer-only** variable is to be set, e.g., in a fixed range of columns, that it is written as an integer (i.e., without a decimal), otherwise a "FORTRAN Read Error" will result and, most likely, the program will terminate. For example, writing 1.0 instead of 1 will cause such an error, even if it is within the correct columns.

On the other hand the FORTRAN input conventions for **real** numbers are more forgiving so that either 1 or 1.0 will be correctly read as "one" as long as it is within the correct columns. In addition, the **number** of decimal places to be included is generally defined within the program for a real input field but need not be strictly adhered to. Thus a Fortran format of F5.2 would imply that a real value is required within a block of 5 columns with 2 characters after the decimal point; e.g., 1.23, not 1.234. And certainly if F5.2 were used as an **output** format 1.23 would always be output, never 1.234. However if 1.234 were to be input under Format F5.2 the third decimal place would be correctly read; equally 123.4 would be correctly read even though it only has one decimal place, and 123 would be read as 123.0 even though it doesn't have an explicit decimal.

Note, however, that the field "width" must always be strictly adhered to so that, for example, with F5.2 it would not be possible to input a value such as 123.45 which requires 6 columns, or indeed any value greater than 99999.0.

In very old versions of **SATURN** which were supplied to users as source code for the users to compile with their own compilers life was more complicated in that certain compilers would require that, with a format F5.2, the decimal place **must** appear in column 3 whereas others would be more forgiving as above. Therefore the Manual tends very often to specify a **fixed** decimal place whereas, with current .exe programs, the input formats are flexible as above.

In brief, you can input real numbers in virtually any format you like as long as you stay within the correct "width" of the input field.

Finally we note that there are a number of instances of inputs which are generally input as integers but which are stored as reals and therefore, strictly speaking, may be input as either an integer or a real. For example, traffic signal stage times (see Record Type 3, 6.4.1) are generally specified with sufficient precision as integer seconds but may, if desired, be input as a real value in order to achieve



greater precision. Similarly counts, which are stored as reals, are generally input as integers on the assumption that it is difficult to define a flow to greater than 1 PCU/hr.

Clearly, however, an input which defines, say, the number of data entries which follow can only be input as an integer.

2.9 Errors and Warnings: Fatal, Non-Fatal and Serious

Error messages in SATURN have five levels; in order of decreasing severity these are:

- ◆ Fatal errors.
- ◆ Semi-fatal
- ◆ Non-fatal errors.
- ◆ Serious warnings.
- ◆ Warnings.

Generally speaking these are applied to the processing and validation of input data files such as network files but they are also used in interactive programs.

Fatal errors, as the name implies, cause the program to terminate abnormally. This does not necessarily imply that the program itself cannot continue at that point; it may be that at a later point in that program or even in another program life would become impossible. In some situations a program may continue even after a fatal error has been encountered; for example, **SATNET** will continue to process data after a fatal error in order to detect other errors later on in data sets. However it would not carry on to the very end.

Semi-fatal errors were first introduced in **SATURN** 10.1. They are applied to errors in **SATNET**, which would prevent (or are expected to prevent) the assignment - simulation procedures from running normally but would not prevent the map network as required to **P1X** to be set up. Networks with semi-fatal errors may therefore be processed within **P1X** and the errors corrected interactively so that they may be processed by the assignment/simulation stages.

Non-fatal errors result from input which **SATURN** can handle without subsequent numerical problems but which, in the program's opinion, is almost certain to produce nonsensical results. Thus defining a free-flow speed of 0 kph would be a fatal (or semi-fatal) error if it would ultimately cause a program to divide by zero; a speed of 0.001 would be a non-fatal error since the program could cope numerically.

Clearly all fatal errors need to be corrected before continuing, and all non-fatal errors should at least be thoroughly checked and confirmed that they are deliberate.

Certain Non-Fatal Errors (plus certain Serious Warnings – see below) are optionally upgraded to Semi-Fatal status if a network parameter **WRIGHT** is set equal to T. This convention is strongly recommended. See Section 6.12.2.



Serious warnings are similar to non-fatal errors, but less obviously bound to produce nonsense, while warnings refer to input which looks strange but may be OK. Verification of both is recommended but not, if you like, mandatory.

However, it is worth bearing in mind that a warning at any level may not reflect the “true” error. For example, if a user wishes to define data for a link A,B and they accidentally input A,C then this may generate a warning if the input does not make much sense at C but simply correcting the input so that the data makes sense at C does not correct the basic problem of having input the wrong node number. Thus even warnings may be evidence of much more serious problems.


Summary statistics of all five types of errors are given at the end of each program plus, in **SATNET**, at intermediate stages. The same information may also be accessed within **P1X** under Information.

Note that non-fatal errors and warnings may be suppressed in **SATNET** by setting a parameter **ERRYES(n)** to F in the **&PARAM** inputs to **SATNET** to suppress error n. Alternatively, in a limited number of cases, setting **ERRYES(n) = F** may convert a semi-fatal error into a serious warning; e.g., setting **ERRYES(437) = F** allows unidentified links to appear in the 77777 count data without creating semi-fatal errors.

A range of error messages may be set in a single Namelist statement by using a colon; e.g., **ERRYES(33:66) = F** “turns off” all warnings in the range 33 to 66. See Appendix A.

In addition a count of the number of each category of errors detected per simulation node is included in the uf* files and may be displayed as part of the node print outs or node graphics. For information on which error numbers are which please see Appendix L (or, less conveniently, list the file SATTIT.DAT).

2.10 Version Control

| JOB NUMBER: 5101396 | | DOCUMENT REF: Section 2.doc | | | | |
|---------------------|-----------------------------|--|---------|----------|------------|----------|
| Revision | Purpose / Description |  | | | | |
| | | Originated | Checked | Reviewed | Authorised | Date |
| 1 | Re-formatted (Final to DVV) | TF / BG | NS | IW | IW | 06/05/06 |
| 3 | Upgrade to V2 Template | IW | | | | 22/06/06 |
| 3.1 | Update Figure 2.1 | IW | | | | 13/07/06 |
| 3.2 | Web release – Sept 06 | DVV | NP | IW | IW | 08/09/06 |
| 3.3 | Web release – Jan 07 | DVV | NP | IW | IW | 04/01/07 |
| 3.4 | SATURN v10.7 Release | DVV | NP | IW | iW | 12/03/07 |
| 3.5 | Web release – Jul 07 | DVV | NP | IW | IW | 19/07/07 |
| 3.6 | SATURN v10.8 Release | DVV | NP | IW | IW | 25/03/08 |
| 3.7 | Web release – Jul 08 | DVV | NP | IW | IW | 07/07/08 |
| 3.8 | Web release – Dec 08 | DVV | NP | IW | IW | 12/12/08 |
| 3.8.21 | Web release – Feb 09 | DVV | NP | IW | IW | 16/02/09 |
| 3.8.22 | Web release – Jun 09 | DVV | NP | IW | IW | 16/06/09 |
| 10.9.10 | SATURN v10.9 Release | DVV | DG | IW | IW | 04/09/09 |
| 10.9.12 | SATURN v10.9 Release (Full) | DVV | DG | IW | IW | 22/09/09 |
| 10.9.17 | Web release – Jun 10 | DVV | NP | IW | IW | 22/06/10 |
| 10.9.22 | Web release – Dec 10 | DVV | AG | IW | IW | 13/01/11 |
| 10.9.24 | SATURN v10.9 Release (Full) | DVV | AG | IW | IW | 06/05/11 |
| 11.1.09 | Saturn V11.1 Release (Full) | DVV | AG | IW | IW | 31/03/12 |