



---

## 9. Assignment / Simulation Loops – The Role of SATALL

### Mini-Contents Page

<b>9.</b>	<b>Assignment / Simulation Loops – The Role of SATALL</b>	<b>9-0</b>
9.1	General Principles	9-1
9.2	Monitoring Convergence	9-3
9.3	Averaging Flows: The Use of KOMBI and AUTOK	9-10
9.4	Continuing a Previous Assignment (The DIDDLE Option)	9-13
9.5	Making Convergence Easier	9-14
9.6	Elastic Assignment/Simulation Loops	9-19
9.7	SATALL: General Functions	9-20
9.8	SATALL Run-time Convergence Statistics	9-21
9.9	SATALL Convergence Statistics: Full Line Printer Listings	9-22
9.10	Elastic Assignment within SATALL	9-25
9.11	Multiple User Class Assignment within SATALL	9-26
9.12	Special SATALL Extensions	9-26
9.13	The SATALL Batch Procedure	9-28
9.14	The SATURN Batch Procedure	9-29
9.15	SATALL: Technical specifications	9-30
9.16	Version Control	9-33

## INTRODUCTION

This section describes the general principles which underpin the assignment-simulation loop which is necessary for networks with a simulation component. It covers both the use of separate programs (the original method) plus the use of the single program **SATALL** (the strongly recommended technique).

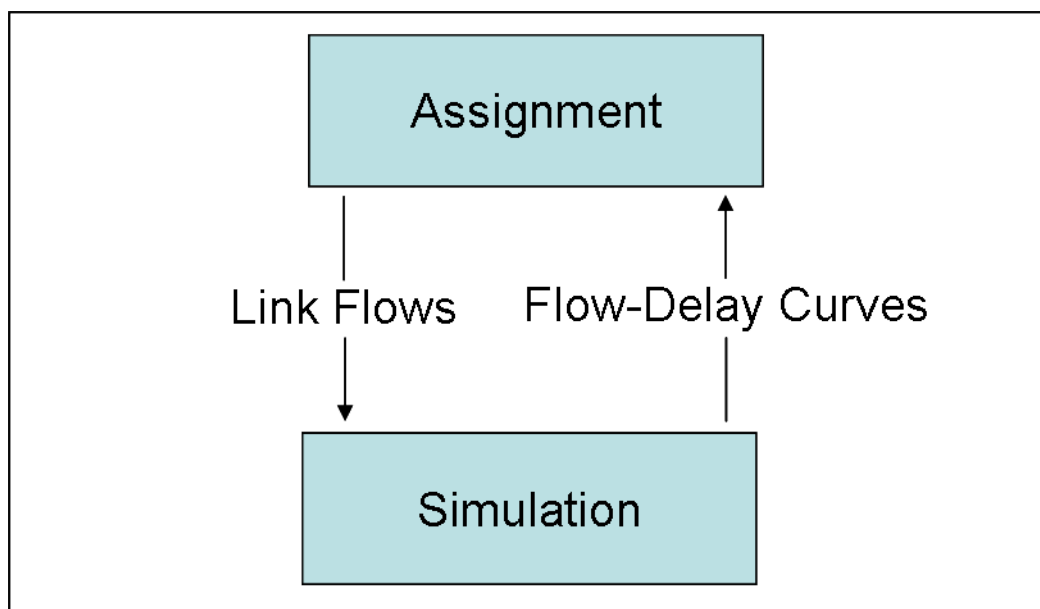
### 9.1 General Principles

#### 9.1.1 Assignment-Simulation Loops

The iterative loops between simulation and assignment have already been noted in Section 3.1 and are illustrated again in Figure 9.1 below. Thus the assignment sub-stage supplies the link flows which are needed by the simulation which in turn supplies the flow-delay curves for simulated turning movements which the assignment requires.

This loop may exist either as a loop between two distinct programs **SATEASY** and **SATSIM** or, in versions of **SATURN** from 9.1 onwards, within a single combined program **SATALL**. In both cases the basic principles are the same although, in certain respects, there is greater potential for flexibility within the one program **SATALL**. The use of **SATALL** is now virtually universal.

**Figure 9.1 - The Simulation-Assignment Loop**



#### 9.1.2 Why Loops are Necessary

The reason why the loops are necessary is essentially due to the fact that the turn-based flow-delay curves used by the assignment are only approximations in that they ignore the “interactions” between links in the determination of delays. To illustrate a very simple case, consider a T-junction with one minor (give-way) arm and one major arm. Here the delay on the minor arm will depend both on the flow along the minor arm as well as the flow on the major arm (and indeed the latter effect will probably dominate). However the flow-delay relationship set by

simulation only includes the effect of the minor arm flow - the implicit assumption is that the major flow is fixed throughout the assignment. If the flow does remain constant on two successive assignments then the assumption is correct; if it does not then the routes generated by the assignment are, to a greater or lesser extent, inconsistent with the simulated delays.

There are a considerable number of other sources of “interaction” between links, for example, shared lanes, blocking back, signal optimisation and co-ordination, merging and weaving, etc. etc. These interactions may be highly asymmetric and sometimes much stronger than the “intra-link” effects which means that, from a theoretical point of view, the ultimate convergence of the process has very few mathematical guarantees.

In order to deal with the interactions **SATURN** loops between assignment and simulation until (reasonably) steady flows are obtained, at which point the model is judged to be “self-consistent” or “in equilibrium”; i.e., the flows that go into the simulation produce delays which in turn produces the same flows on assignment. (Technically this approach is known as the “diagonalisation method”).

The (main) parameter used to monitor the rate of convergence is the percentage of link flows which vary by less than, say, 5% (the parameter PCNEAR) between loop  $n$  and loop  $n-1$ . If this exceeds the (integer) parameter ISTOP then the process is judged to have converged satisfactorily. If the criteria is satisfied on NISTOP successive loops then the process is terminated (where, by default and in all versions prior to 10.3, NISTOP = 1 so that the loops stop as soon as the ISTOP criterion is met).

Note that with release 11.1 the critical integer value ISTOP has been converted into a **real** value RSTOP; see 9.2.6.

In addition a number of alternative convergence criteria (e.g., the “gap” parameter) have been introduced at a later stage of **SATURN** development and these are described further in section 9.2.3.

Alternatively the procedure will always terminate if a maximum number of loops (the parameter MASL) has been reached.

Whether or not convergence will be achieved by this method is hard to determine in advance. Essentially the process works well when the “interaction” effects referred to above are relatively weak, but when they are strong problems of oscillations can occur. For example, networks with a large number of give-ways may create problems. See 9.5.

Clearly good convergence is highly desirable; poor convergence means that the results are imprecise / “in error”. A more precise description of the statistics used to monitor convergence is given in 9.2 and advice on how to achieve good convergence (or, conversely, how to avoid poor convergence) is given in 9.5. On the other hand, poor convergence is not the only source of error in the model outputs (cf. data input errors, section 2.1) and good convergence generally requires longer run times so that, in practice, the target level of convergence will represent a compromise between accuracy and run times.

### 9.1.3 SATALL: General Features

The program **SATALL**, first introduced with **SATURN 9.1**, in effect combines the programs **SATASS/SATEASY** and **SATSIM** into a single program and carries out a full assignment/simulation convergence loop internally. Thus, as shown in Figure 3.1, taking as input a .ufn network file built by **SATNET**, it both assigns and simulates so as to produce an output .ufs file containing converged assigned flows plus the corresponding simulated delays.

It may also carry out additional post-loop calculations and produce additional files, in particular an extra final (SAVEIT) assignment in order to facilitate future analyses of the assigned path flows; see 15.23.2

In addition to the final flows, travel times etc., the output .ufs files also contain a wide range of convergence plus selected aggregate statistics from each individual assignment/simulation loop which may accessed via **SATLOOK** and/or **P1X**. The aggregate statistics from several loops may also be averaged; see 17.9.2.

By combining two programs into one **SATALL** should be both faster and, ultimately, “more clever” in terms of the steps that can be introduced in order to improve the rates of convergence. For example it can combine the DIDDLE option with elastic assignment which is otherwise impossible; see 9.10. All the various distinct options for assignment and/or simulation that can be invoked with the separate programs **SATEASY** and **SATSIM** may now be carried out within **SATALL**. Indeed, as with the elastic DIDDLE (what a great name!) mentioned above, there are extra options only available with **SATALL**; see 9.12.

### 9.1.4 Assignment-Simulation Control Procedures

The traditional ‘control procedure’ to automatically run the loops between the distinct assignment and simulation programs, **SATURN8**, is described in Section 14.3. The equivalent and current standard procedure, **SATURN**, which runs **SATALL** is described in Section 9.14.

## 9.2 Monitoring Convergence

The rate of convergence of the assignment/simulation loops can best be monitored using 2 tables of convergence statistics which are (a) included within the .lpt files but (b) may also be viewed interactively using either option 8 of **SATLOOK** or under **P1X** Information and/or Convergence Options.

### 9.2.1 Table 1 Convergence Statistics by Sub-Module and Loops

The contents of Table 1 may vary depending on the precise assignment algorithm being applied. In the case of standard Wardrop Equilibrium the output plus interpretation is as given below:

- ◆ Assignment - Delta Function (%) / Number of iterations
- ◆ Simulation - Final Average Absolute Change in OUT CFP (PCU/HR) / Number of Iterations
- ◆ A/S Step - Step length used per loop / Simulation repetitions

- ◆ Assignment/ Simulation Loop - % Link flows differing by < 5% between successive assignments
- ◆ Assignment/Simulation Loop - % Turn delays differing by < 5% between the assignment and subsequent simulation
- ◆ Variational Inequality % - should be > 0
- ◆ Wardrop Equilibrium % Gap Function

N	Assignment	Simulation	% Flows	% Delays	% V.I.	% GAP
1	0.109/9	0.008/5		51.3		16.488
2	0.356/9	0.672/5	71.0	79.0	0.002	5.798
3	0.229/9	0.522/5	72.9	86.6	-0.001	0.222
4	0.351/9	0.379/5	92.8	87.4	0.000	0.332
5	0.399/9	0.305/5	93.5	93.3	0.000	0.230
6	0.327/9	0.508/5	96.9	94.1	0.000	16.488
7	0.300/6	0.522/5	60.4	95.0	0.045	5.798

Thus column 1 monitors the “delta function” which is internal to the assignment and measures the degree to which the routes assigned traffic are minimum cost routes; see section 7.1.4. If, as a rule of thumb, these figures are much in excess of 1% then you should consider increasing the parameter NITA to obtain better internal convergence. Otherwise the “uncertainty” in each assignment may be adversely affecting the overall convergence. (For an alternative point of view see 9.5.4.)

See 7.1.4 for more information on delta.

Similarly column 2 monitors the successive internal convergence of each simulation using the parametric measure discussed in 8.3. Again, as a rule of thumb, if these figures are much in excess of 1.0 you should consider increasing NITS.

Columns 1 and 2 also give the numbers of assignment and simulation internal iterations undertaken (for which NITA and NITS set upper limits).

Column 3 reports the factor used to average successive assignment flows between loops under KOMBI or AUTOK (see 9.3) and, in the case of AUTOK, the number of internal repetitions of the simulation in order to calculate the optimum weights (9.3.2).

The remaining 4 columns all monitor the convergence of the assignment/simulation loops and should, roughly speaking, tell a similar story.

Firstly “% FLOWS” reports the fraction of assignment links whose assigned flows changes by less than 5% (or, strictly speaking, less than PCNEAR%) from one simulation-assignment loop to the next. This is a somewhat arbitrary function but one which has been used in **SATURN** from the beginning and has thereby acquired a certain historical momentum. It is (by default, see 9.2.3) the “main”

convergence parameter in that it is used to stop the loops once the figure exceeds the parameter ISTOP (or RSTOP; see 9.2.6); typical values for ISTOP are 90 or 95%. ISTOP and/or RSTOP are set in &PARAM; see 6.3.2. See also 9.2.3 for alternative stopping criteria.

“% DELAYS” is similar to “% FLOWS” but is based on the fraction of simulation turns whose delays change by less than 5% (i.e. PCNEAR%) from those calculated by the previous assignment. Note that the simulation turns are a subset of the assignment links; hence the latter measure is based on a different - and larger - “sample” than the former. % DELAYS has no effect on stopping the loops.

These two measures, while generally similar, can present different viewpoints under certain circumstances. Thus in highly congested networks, where delays are a very sensitive function of flows, it is quite possible for the flows to settle down (high %FLOWS) but for the delays to fluctuate wildly (low %DELAYS).

Alternatively if the delay-flow relationship is very “flat” it is possible that the delays will be stable (high %DELAYS) but for the flows to wander (low %FLOWS). Thus if only 1 of these measures is high it probably implies that your overall convergence is acceptable even though either flow or delay is uncertain.

“% V.I” is more complicated to explain. In essence it compares the costs on the currently assigned routes using the current simulated delays to the costs on the previously assigned routes using the same delays. One should therefore expect the current routes to be “cheaper” if we are getting nearer to Wardrop equilibrium; if %VI is positive this is true, if negative, then the former routes are actually cheaper than the current routes. This situation arises if the flow pattern has changed too drastically from one loop to the next - one solution is to use the KOMBI or AUTOK parameters to “dampen down” the changes in flow. Thus if %VI goes strongly negative on, say, loop 6 then that is a strong argument for setting KOMBI to 6. We return to this question in Section 9.3 below.

Finally the “% GAP” is a generalisation of the delta function - column 1 - to include the interaction effects within the simulation. It is, firstly, the difference between the current total vehicle costs on the assigned routes and the total vehicle costs if all drivers were to use minimum cost routes with the costs FIXED. This measure is then “normalised” by dividing by the total vehicle costs and expressing it as a %. It is therefore the same as equation (7.3) for delta (Section 7.1.4) except that the costs are calculated after the simulation rather than after the assignment.

Generally speaking the GAP values will be greater than the DELTA values since the routes chosen based on the assignment cost estimates will tend to be slightly worse when the costs are further changed by the simulation. The difference between GAP and DELTA is therefore an indication as to how far the assignment and simulation stages “disagree” on the calculation of delays; for further statistics on the level of disagreement and the turning movements which may be causing it please see Section 9.9.1.

As with delta (7.1.4) a gap of under 1% may be regarded – at least for some purposes – as satisfactory. For example it is much better than the ability of drivers in real life to choose “true” minimum cost routes which may be categorised as around 5%. However, for other modelling purposes, significantly lower GAP values need to be achieved and, indeed, should be achieved. A GAP value of

under 0.1% or even 0.01% should be regarded as a suitable target. See 9.2.4 for a more complete discussion and 9.5 for advice on improving convergence.

Note that the latter two measures assume that the assignment is trying to assign all trips to minimum cost routes. As this is not true under stochastic assignment % VI and % GAP are not reported there.

### 9.2.2 Table 2

The second table contains extra convergence statistics relating to the progress of the assignment-simulation loops:

ASS-HRS	The total pcu-hr/hr from the buffer + simulation networks calculated by the assignment
CHANGE	% Change in ASS-HRS between successive loops
SIM-HRS	Total pcu-hr/hr from the simulation network only
SIM-KMS	Total pcu-km/hr from the simulation network only
GEHBAR	Mean GEH Statistic comparing link demand flows between two successive assignments
AAD	Average Absolute Difference in link demand flows between two successive assignments (PCU/HR)
RAAD	% Relative Average Absolute Difference in Link flows
XMSD	% Relative Standard Deviation in link flows
SAD	Mean Absolute difference in delays between the assignment and simulation (seconds)
RSAD	% Relative Mean Absolute Difference in ASS/SIM delays

Certain of the above measures are those recommended by the DfT for monitoring the degree of convergence of any 'congested assignment model' as set out in 4.4.19-4.4.28, Chapter 4, Section 2, Volume 12 of DMRB.

### 9.2.3 Terminating the Loops: Alternative Convergence Criteria

The simulation-assignment loops terminate under the following conditions:

- 1) The number of loops exceeds MASL;

Or

- 2) The number of loops does not exceed MASL **and**;
  - a) %FLOWS exceeds ISTOP / RSTOP on NISTOP successive loops,
  - b) GAP is less than STPGAP on NISTOP successive loops.
  - c) The total CPU time exceeds STPCPU,
  - d) Various and/or combinations of the above 3 criteria.

3) A minimum number of loops, MASL\_M, is satisfied (added in 11.1).

The choice between which of conditions 2 is applied is set by the parameter KONSTP which may (post release 11.1) lie in the range 0 to 6. All five control parameters KONSTP, ISTOP/RSTOP, NISTOP, STPGAP and STPCPU are set under &PARAM in **SATNET** (or **SATALL**) and default to 0, 95, 4, 1.0 and 1000.0 respectively.

The allowed values of KONSTP have the following interpretation:

0 – ISTOP/RSTOP only is required

1 – GAP only is required

2 – Total CPU time is used as the sole stopping criterion.

3 – ISTOP with an upper limit on the total CPU time

4 – GAP with an upper limit on the total CPU time

5 – **Both GAP and ISTOP** must be satisfied (independent of CPU)

6 – **Either GAP or ISTOP** must be satisfied

Note that tests based on GAP are not always available, depending on the exact form of assignment algorithm used. Thus GAP is not calculated under Stochastic Assignment or Multiple User Class Elastic Assignment.

Historically **SATURN** used %FLOWS as its stopping criteria (in addition to MASL) – so that KONSTP = 0 - although it is a fairly simplistic measure with very little theoretical pedigree. For example, having a fixed cut-off between “OK” and “not OK” means that it fails to distinguish links that fail marginally and links that fail by a large amount. Essentially it was introduced at a very early stage of program development when a rule was needed and it just hung about until it was so deeply entrenched it was difficult to get rid of! On the other hand, in its favour, it is easy to calculate and understand and works in all possible situations, not just Wardrop-based models.

A further problem with the use of %FLOWS as the stopping criterion is that it may depend on the “accuracy” of the assignment method used. Thus if one uses an extremely accurate assignment such as OBA (see Section 21) the **true** difference in link flows between loops n-1 and n will be obtained (to a good approximation) whereas with a less accurate technique, such as the default Frank-Wolfe algorithm, there is less scope for the assignment in loop n to move away from the assignment in loop n-1 (which is used as its starting point, assuming of course that DIDDLE = T as it should be); hence the differences in link flows tend to be reduced and %FLOWS measure increased. Hence, despite being a better assignment method with better convergence properties, OBA may perversely appear less convergent than Frank-Wolfe in terms of %FLOWS.

On the other hand, GAP does have a definite theoretical interpretation, does differentially weight “good” and “bad” fits and is easy to compare between networks of wildly different shapes and sizes. It is also more “neutral” with respect to the problem of assignment accuracy discussed above.

On balance, therefore, our current “best buy” for a stopping criterion is the GAP (set KONSTP = 1 or 4) although we recognise that there is a strong case for carrying on with %FLOWS for historical continuity and the default continues to be %FLOWS (KONSTP = 0).

However, whichever stopping criterion users choose, they should always view GAP as their most important **single** indicator of overall convergence.

N.B. If STPGAP is used as the stopping criterion for assignment-simulation loops then it is also recommended that the parameter UNCRTS which sets (one of) the stopping criteria for the assignment stage on its own should be less than STPGAP, otherwise the overall convergence will be restricted by the assignment convergence. Alternatively setting AUTONA = T (See 9.5.4) allows the program itself to correct for too high values of UNCRTS during each assignment loop.

## 9.2.4 Convergence Criteria: What is “Good Enough”?

We consider here the question of what sort of values are “acceptable” for the various stopping criteria used not only in the assignment-simulation loops (ISTOP etc.) but also in the assignment and simulation sub-stages themselves.

Such questions are intimately connected with the purposes for which the model is being run. For example, if you wish to do a very broad-brush “quick and dirty” estimate of what traffic conditions may look like in 20 years time the results will, of necessity, be very inexact and there is no point in imposing very strict convergence criteria. On the other hand calibrating a present-day network where you have extremely good data may justify very strict criteria.

One particular case where very good convergence – and therefore very strict criteria – is absolutely required is the comparison of “with” and “without” scheme networks where the differences are likely to be relatively small and can only be accurately measured if both sets of results are extremely accurate. Otherwise the differences will simply get lost in the “noise”.

It can be argued that, as a very general rule of thumb, the reduction in total vehicle-costs due to the “scheme” should be ten times larger than the “noise” in the model (as outlined in WebTAG Variable Demand Modelling, Section 3.10.4). This implies that if a scheme reduces total travel cost by, say, 1%, then you require a GAP value of 0.1% or better to achieve a satisfactory evaluation.

**It might also be argued that there is no such thing as a “minimum acceptable” level of convergence and that modellers should always try to reduce GAP values to the absolute minimum values which are technically possible, limited only by practical considerations of time and/or money.** Section 9.5.3 gives general advice as to how to achieve “extremely good” convergence.

## 9.2.5 Guidance on Convergence

### 9.2.5.1 Current Guidance

The current advice on model convergence was set out in DMRB 12.2.1 Table 4.1 (Chapter 4, Vol 12, Section 2, Part 1) and reproduced below. The advice was originally issued in 1997 and has not been updated since - it is widely recognised

that the convergence targets are set substantially below the level required in order to produce robust estimates of traffic flows and costs for model development and appraisal.

Type	Convergence Measure	Suggested Values
Proximity	Delta (or %GAP for SATURN)	Less than 1% or at least stable with convergence fully documented and all other criteria met
Stability	Percentage of links with flow change (P) < 5%	Four consecutive iterations (SATALL loops) greater than 90%
	Percentage change in total user costs (V)	Four consecutive iterations (SATALL loops) less than 1% (SUE only)

### 9.2.5.2 Emerging Guidance

The latest, emerging guidance from the transport practitioners (and summarised in TAG Unit 3.19c) suggests a more stringent set of standards as described below for **illustrative purposes** only. The emerging guidance notes that different levels of convergence may be required through the course of the study. For example, a more relaxed convergence level may be appropriate to ensure sufficient number of loops of matrix estimation may be practically undertaken whilst more stringent criteria would be applicable for economic appraisal.

Type	Convergence Measure	Suggested Values
Proximity	Delta (or %GAP for SATURN)	Less than 1% or at least stable with convergence fully documented and all other criteria met
Stability	Percentage of links with flow change (P1) < 1%	Four consecutive iterations (SATALL loops) greater than 98%
	Percentage of links with cost change (P2) < 1%	Four consecutive iterations (SATALL loops) greater than 98%
	Percentage change in total user costs (V)	Four consecutive iterations (SATALL loops) less than 0.1% (SUE only)

**In all circumstances, the onus remains on the SATURN user to ensure that their assignment is converged to an appropriate level.**

### 9.2.6 ISTOP vrs RSTOP

In release 11.1 an alternative form of the ISTOP stopping criteria, RSTOP, has been introduced. Basically RSTOP performs the same function as ISTOP, the one difference being that ISTOP may take only integer values whereas RSTOP may take real values (e.g., 96.7 vrs 96 or 97). In effect ISTOP has always been treated as a real variable rounded down from its integer value, e.g., ISTOP = 97 was effectively 96.5. The reason for this was that if the percentage of OK links had

been 96.6 then, when it was compared to ISTOP, it would be rounded up to the nearest integer, i.e., 97%, and compared to the integer value ISTOP, e.g., it would satisfy ISTOP = 97. Equally 96.4 would be rounded down to 96% and would not pass ISTOP = 97. Hence the effective critical value would be 96.5. So whether we round ISTOP down to a half-integer value to be compared with the exact (real) OK value, or round the OK value to the nearest integer, the end effect is the same.

Thus in Version 11.1 the comparison is always done using real values where the critical stopping criteria RSTOP may either be defined directly by including a variable RSTOP under &PARAM or, if RSTOP is not explicitly set, then RSTOP = ISTOP – 0.5.

Note that this should have no impact on re-running old networks if RSTOP is not defined since the stopping criteria is effectively unchanged.

### 9.3 Averaging Flows: The Use of KOMBI and AUTOK

Clearly non-convergent flows are undesirable. One way of trying to deal with the problem is to average the assigned flows over successive loops. Thus if after  $n$  assignment-simulation loops the link flows are  $V_a^{(n)}$  and we carry out a further assignment (using whatever assignment method -- Wardrop, Stochastic, etc) to obtain flows  $F_a^{(n+1)}$  then we take a strict 50:50 average of the two flows to obtain:

#### Equation 9.1

$$V_a^{(n+1)} = (F_a^{(n+1)} + V_a^{(n)}) / 2$$

or (post-SATURN 10.4) a  $\Lambda$ -weighted average :

#### Equation 9.2

$$V_a^{(n+1)} = (1 - \Lambda)V_a^{(n)} + \Lambda F_a^{(n+1)}$$

The first method is associated with the KOMBI parameter and the second with AUTOK as explained below. If neither is used then:

#### Equation 9.3

$$V_a^{(n+1)} = F_a^{(n+1)}$$

Empirically such methods appear to improve the convergence of “badly behaved” networks, in effect by damping down flows which otherwise may oscillate wildly over successive loops. On the other hand the averaging may actually slow down convergence for “well behaved” networks.

#### 9.3.1 Use of KOMBI

The loop at which 50:50 averaging first occurs is set by the parameter KOMBI; thus setting KOMBI = 3 allows 3 assignments with the “normal” method before averaging is introduced. If, of course, convergence (relative to ISTOP) is achieved within KOMBI loops then no averaging takes place.

Provided that convergence does occur naturally then there are strong reasons for not invoking KOMBI (see below). Our advice to users is to first test networks with KOMBI = 99 (or 0, the effect is the same) and if convergence is seen to be

proceeding happily enough (see 9.2) then leave well enough alone. If however the flow-convergence is seen to decrease, say, after loop 5 then consider setting KOMBI to 5.

Alternatively use AUTOK which **should** be an improvement!

### 9.3.2 Use of AUTOK

The AUTOK facility (AUTOMATIC Kombi) was first added in **SATURN** 10.4 as a method for automatically determining (a) at which point averaging should be introduced and (b) the appropriate  $\Lambda$ -weights so that the user would not need to make such decisions as to appropriate values of KOMBI via a process of trial and error. Its theory is very similar to that used under the ROSIE option, 7.1.3.

Thus if AUTOK = T (the default is F) then after each assignment a full simulation is carried out using the latest assigned flows (i.e., without any averaging), at which point a test is carried out on  $\% \Phi$  (as described qualitatively in 9.2.1 and defined more precisely below, equation (9.5)) to test whether averaging would improve convergence. If  $\% \Phi$  is positive then no further action is taken; if, however,  $\% \Phi$  is (significantly) negative then the flows are averaged as per equation (9.2) with an “optimum” value of  $\Lambda$ .

The criterion on which the optimum value of  $\Lambda$  is derived from the same optimising rule as applied within the Frank-Wolfe algorithm for networks with “separable” cost-flow curves; see 7.1.3. This in turn is based on a 1987 paper by one Dirck Van Vliet (“The Frank-Wolfe Algorithm for Equilibrium traffic Assignment Viewed as a Variational Inequality”, Transportation Research 21B, 87-89) which in turn was based on “Viewing Wardrop Equilibrium as a Variational Inequality” (Smith, 1979). Thus the Frank-Wolfe rule for combining the current assigned flows  $V_a$  with the latest all-or-nothing assigned flows  $F_a$  may be written as the solution to:

#### Equation 9.4

$$\sum c_a(\lambda) \{V_a - F_a\} = 0$$

where  $c_a(\lambda)$  represent the link costs with the flows averaged as per equation (7.2).

We may think of the (negative) costs as representing a “Social Force Field” which is pushing the current solution  $V_a$  in the direction of the cheaper routes represented by  $F_a$ . The solution (9.4) represents the point at which the social force field has shifted such that the all-or-nothing routes are no longer the cheapest routes (because they have been allocated extra traffic and the original routes less) and the force field is now at right angles (“normal”) to the direction of flow change.

The equivalent rule when applied to two successive sets of fully assigned flows from assignments  $n$  and  $n+1$  would be to require that:

#### Equation 9.5

$$\sum c_a(\Lambda) \{V_a^{(n)} - F_a^{(n+1)}\} = \Phi(\Lambda) = 0$$

where now the costs  $c_a(\Lambda)$  are those derived from simulation  $n+1$  based on  $\Lambda$ -averaged flows. Again we may think of the costs as a force field pushing the

solution from  $V_a^{(n)}$  towards  $F_a^{(n+1)}$  and that the simulation is changing the direction of the force field by calculating different costs.

If, as mentioned above,  $\Phi(1) > 0$  then a step-size of 1 is used and no averaging takes place. N.B. This is the most frequent result; see below. Alternatively, if  $\Phi(1) < 0$ , then equation (9.5) is solved by a heuristic method of successive interpolations.

Thus we first calculate  $\Phi(0)$ ; this may be done, in fact, immediately after simulation  $n$  where the simulated costs are, in effect,  $c_a(0)$ . In theory  $\Phi(0) > 0$  (although it is possible that lack of assignment convergence and/or rounding errors might drive it marginally positive) so that our first estimate of the optimum  $\Lambda_1$  is simply based on a linear interpolation between  $\Phi(0)$  and  $\Phi(1)$ :

#### Equation 9.6

$$\Lambda_1 = \Phi(0) / (\Phi(0) - \Phi(1))$$

and we carry out another simulation with weighted flows and obtain the “correct” value for  $\Phi(\Lambda_1)$ .

If  $\Phi(\Lambda_1)$  is zero, or very near zero, then we terminate. If, however,  $\Phi(\Lambda_1)$  is significantly different from zero we may estimate an improved value  $\Lambda_2$  by approximating  $\Phi(\Lambda)$  as a quadratic function using the three points we have already simulated at  $\Lambda = 0$ ,  $\Lambda_1$  and 1. If, having carried out a further simulation with  $\Lambda = \Lambda_2$ ,  $\Phi(\Lambda_2)$  is not sufficiently near zero then the process of quadratic approximations and re-simulation continues using the last three estimated points until the zero-point is obtained with sufficient accuracy. Empirically it would appear that the solution is obtained “most of the time” with a simple linear interpolation or a small number of quadratic steps.

### 9.3.2.1 Accelerated Factoring of $\Lambda$

If an optimum value of  $\Lambda$  is not obtained after, say, 2 or 3 iterations and the same behavior is noted consistently over several assignment-simulation loops then it may be possible to “accelerate” the estimation of  $\Lambda$  by introducing an additional empirical factor based on the ratio of the final  $\Lambda$  value to the initial linear interpolation given by equation 9.6. Thus if we consistently observe that the final value is 0.5 times the initial value then it may well save time (by reducing the number of repeated simulation steps) if on the next application of AUTOK we reduce the initial estimate by a factor of somewhere between 0.5 and 1.0.

This factor is referred to as the “AUTOK AVERAGE STEPS FACTOR” in the .LPT output files and is not a constant but varies throughout based on the most recent experiences.

This “fix” was first introduced in 10.9.17 and has been found to marginally reduce the number of repeated simulations and therefore reduce CPU time.

### 9.3.2.2 The Role of $\Lambda$ : The Use of AK\_MIN

Post 10.9 a minimum value may be imposed on  $\Lambda$  by setting a Namelist &PARAM parameter AK\_MIN in the network .dat file with a default value of 0.0 (which means it will never be applied). The rationale behind setting a minimum value of  $\Lambda$

is that a value near zero implies that we virtually retain the previous flows and are therefore making very little progress towards convergence, due, most likely, to some sort of “quirk” in the simulation. A small value of AK\_MIN may therefore provide a little “impetus” for the model to converge. A value of 0.10 has been found to be appropriate in one particular network which was frequently generating values of (virtually) 0.0 but little comprehensive empirical testing has been done.

It should, however be noted that, again empirically, that in many networks averaging is **never** required and that in the remaining networks averaging is only required on a small fraction of the loops. Whether or not averaging is needed is essentially down to how much “interaction between links” is being introduced by the simulation as described in 9.1. If it is very strong then averaging may be necessary to “dampen it down”.

Somewhat perversely it also appears empirically that averaging is required more often if the assignments are extremely well converged internally, e.g., if you are using OBA assignment or PARTAN. This may be interpreted as being due to a poorly converged assignment not taking you very far from the current solution and therefore not as far as the “correct” solution. However a more accurate assignment may actually cause you to “overshoot” the correct solution, in which case the averaging is effectively bringing you back towards the previous solution.

Information on the  $\Lambda$ -weights used to average assigned flows and the number of internal loops used to calculate those weights on each loop are displayed within the standard table of assignment-simulation convergence statistics (see 9.2.1) as printed within the .lpt files or as displayed by request by **P1X** or **SATLOOK**.

### 9.3.3 Possible Problems with KOMBI or AUTOK

One minor reason for NOT using KOMBI is that, if you are using link-based assignment (the default), any SAVEIT-based analysis of the route pattern AFTER the end of the loops, for example a PIJA analysis, printing a forest or cordoning a trip matrix, becomes approximate although, see 15.23, it should be a very good approximation. However if using KOMBI is the price that has to be paid for achieving good convergence then it is a price well worth paying - the problems introduced by an approximate “SAVEIT” are relatively minor.

The same problem does not arise with AUTOK as long as averaging was not applied on the final assignment - simulation loop.

However this objection does **not** apply if you are using either path-based or origin-based assignment where the path information is preserved exactly.

## 9.4 Continuing a Previous Assignment (The DIDDLE Option)

As mentioned in 7.1.2 the Frank-Wolfe algorithm normally starts with an all-or-nothing assignment based on free-flow link costs. However an alternative starting point is to use the final assigned flows from the previous assignment (on loops after the first). This has the advantage that the previous flows are likely to be “nearer” to the desired flows on this assignment loops than is an all-or-nothing solution so that starting with them should reduce the number of internal iterations required and/or lead to a more accurate solution. This option is selected by setting the parameter DIDDLE to .TRUE.

DIDDLE is a relatively new **SATURN** option but it has proved to be highly effective both in terms of reducing the number of internal assignment iterations and improving the assignment/simulation convergence loops. With version 9.3 (and later) DIDDLE also works with elastic assignment when the loops are within **SATALL** (but not with **SATEASY**). It will not however function with stochastic assignment.

An earlier problem that SAVEIT could not be used in conjunction with DIDDLE no longer applies as (see 15.23) the route flows are estimated at the end of a full assignment whenever DIDDLE is in effect. As with KOMBI (last paragraph in 9.3), the use of DIDDLE introduces an element of approximation into the saved routes but these problems are minor compared to problems of convergence.

Given the intrinsic advantages of using DIDDLE its default value is set to .TRUE.

## 9.5 Making Convergence Easier

### 9.5.1 Improved Convergence

Unfortunately there are no precise rules for what to do if your network does not converge as well as you might expect or require (see 9.2.4). “All happy networks are alike but an unhappy network is unhappy after its own fashion”. The following are therefore only suggestions as to what you might try; if they work, fine - if they don't, keep on trying!

This section is primarily concerned with networks which appear to be converging slowly and/or irregularly between the assignment and the simulation; e.g., their %FLOWS reach 90% but fail to progress much further. The next two sections provide advice on what to do with (a) really badly converged networks and (b) very well-converged networks.

- 1) Check all warnings generated by **SATNET**, particularly Serious Warnings and Non-fatal Errors. Coding errors are the single most common cause of convergence problems. For example multiple turning movements sharing a single lane can cause severe convergence problems; see 6.4.9. N.B. The easiest way to detect and correct coding errors is to use the facilities to **P1X** to “highlight” nodes with specific errors and to automatically loop over those nodes within node graphics; see 11.6.5.4.
- 2) Check the “Parameter Examination” advice as provided interactively under **P1X** / Information and/or listed in .LPN and .LPT files. Many of the checks included there also appear below.
- 3) Use the various “10 Worst Converged” lists available in **P1X** / Convergence (see 11.15) in order to identify particular trouble spots in the network which may require corrective action. The lists of “worst” locations may be supplemented by the option to “highlight” (see 11.6.5.4) the locations of the nodes where they occur, following which the option “Hilite Nodes Loop Graphix” allows the user to automatically “loop” over those nodes in order to demonstrate, using node graphics, potential problems. Note, in particular, all error messages at such nodes even if they may not be the direct cause of the convergence problems.

- 4) Check that the global levels of network congestion “look right”. Highly congested networks tend to converge erratically so if, for example, there are too many trips in the trip matrix the resulting congestion may adversely affect convergence. (So, possibly, you need to consider elastic assignment.)
- 5) Check that individual assignment and simulation sub-stages are converging during the later loops at least; if not, increase **NITS** and/or **NITA** (but see point 6 below and 9.5.4 re NITA). (Poor convergence on early loops may not be a problem if they eventually come right.)
- 6) Increase **MASL** - after a large amount of wandering about in the wilderness you may strike it lucky!
- 7) Use (strongly recommended!) **AUTOK** (Section 9.3.2) in preference to **KOMBI**, **AUTONA** (Section 9.5.4) and – definitely - **DIDDLE** (9.4)
- 8) When using **DIDDLE** there is a strong case for reducing the maximum number of internal iterations per assignment, **NITA**, but increasing the number of assignment/simulation loops, **MASL**, since the total number of all-or-nothing assignments used in the final solution is the product of the two. Introducing more frequent updates of the cost-flow curves via the simulation may possibly reduce the total cpu time required to converge. See 9.5.4 below and also see NITA\_S, 15.23.3. On the other hand it may sometimes happen with **DIDDLE** that the overall convergence is impeded not by the lack of convergence between the simulation and the assignment but by the fact that assignment terminates after only 1 or 2 loops and does not allow sufficient change in the assigned flows. This may be corrected either by reducing the stopping criteria parameters **FISTOP**, **XFSTOP** and/or **UNCRTS** or, more straightforwardly, by setting the minimum number of assignment iterations (**NITA\_M**) to, say, 3. See also 9.5.3. N.B. **UNCRTS** may be adjusted automatically by the program under **AUTONA** = T; see 9.5.4.
- 9) Check the stage times and offsets at signalised nodes in order to identify possibly badly coded signals since poor coding may lead to certain stages becoming (incorrectly) heavily overloaded which in turn may lead to poor overall convergence. Thus you may need to consider the use of **SIGOPT** and/or **SATOFF** parameters within your network so that signal optimisation is carried out internally within **SATALL** or else consider the use of signal optimisation externally. See Sections 12.2 and 15.31. (Note that optimising stage times generally has a much greater impact on convergence than offset optimisation.)
- 10) Increase the dependence of cost on distance (i.e. increase **PPK**) since distance (clearly) is fixed and does not vary between loops as does time so that the assigned routes are less sensitive to time changes. (Although your choice of **PPM** and **PPK** may well be constrained by other factors such as evaluation.)
- 11) Make sure that **LTRP** > 0 as this helps to reduce “discontinuities” in cost-flow curves, particularly at  $V = C$  for major arms at priority junctions

where there are very few other causes of below-capacity delays; see 8.6.1. Equally set **RTP108** = T; see 8.6.3.

- 12) The basic reason why networks do not converge is because of the interaction effects between flows and delays; reducing the level of interaction may therefore help (but on the other hand it may make your network representation less precise!). Therefore you could try to:
  - Reduce the number of give-way turns (i.e. remove turn priority markers);
  - Reduce the critical gap values (the default values are, in all likelihood, to be too high);
  - Reduce lane sharing (and/or introduce flaring);
  - Do not use blocking back (set ALEX to zero but **not** recommended);
  - Check all simulation nodes where **MAXQCT** is invoked.
  - Do not use blocking back (set ALEX to zero but **not** recommended);
- 13) (Repeat of 1!). Check the error messages in your .lpn file and/or use the Highlight facility in **P1X** – most of the time that's where the problem lies! The next section illustrates one or two possible extreme cases.

### 9.5.2 Very, Very Bad Convergence: What to do

Inevitably there are certain networks whose convergence behaviour can only be described as “terrible”. For example, their %FLOWS figures get to the mid 80's and then suddenly shoot back to the 70's.

The main cause for such instabilities is, almost always, blocking back, aided and abetted by coding “peculiarities”. For example, if a simulation link has been given a link distance of, say, 10 metres and one lane then its default stacking capacity will be less than 2 pcus. In this case if the exit turns go from V/C ratios of 0.99 to 1.01 then the link will start to block back, the blocking back factor may be extremely low in order to reduce the queue to under 2 pcus and the resulting queue may extend backwards through a long series of links. If V/C drops back from 1.01 to 0.99 then the queues disappear. In this case the modeller has to ask whether or not a link distance of 10 metres is realistic. (In some cases, of course, it may be, but it may also well be that the link was simply added as some sort of pseudo-link, 10 might be a typo for 100, etc., etc.).

Instabilities in blocking back may be detected from the table which displays the 10 links whose blocking back factors change most over a single assignment-simulation loop. In particular look for any links whose stacking capacity is low.

Any link with a stacking capacity of less than, say, 5 pcus is an accident waiting to happen and should be carefully vetted.

The converse situation may also occur, i.e., the link distance and stacking capacity are correct but the queue is unrealistically long due to coding errors at the junction leading to very low capacities. To illustrate the point from one (anonymous!) network: a priority T-junction on a major road into town was coded

such that the X-marker which was meant to indicate that right-turning traffic leaving town had to give way to traffic entering town was inadvertently coded on the opposite arm which meant that the major traffic into town had to give way to right-turning traffic from the other direction. As a result the capacity for traffic into town was reduced from what should have been its saturation flow of 1800 down to 100 or 200, the resulting queue stretched back for 10 km and the whole model became highly unstable. (The coding error, by the way, is now detected as Serious Warning 137 in **SATNET**).

The morale of the story is that even very small errors in network coding can lead to very large network problems and, unfortunately, users have to: (a) be very, very careful in their coding and (b) look very, very carefully at their outputs.

### 9.5.3 Very, Very Good Convergence: How to Get It!

Most networks, provided that most of the coding “funnies” have been removed, should be capable of converging to a very high degree; e.g., gap values of 0.01% or better, %FLOWS of 100%, etc., etc. However, in order to achieve such convergence levels, three conditions have to be satisfied:

- ◆ The assignment must be very well converged;
- ◆ The simulation must be very well converged; and
- ◆ The assignment-simulation loops must be very well converged.

The first may be achieved most easily using origin-based assignment (OBA, Section 21) which can reduce the assignment convergence (i.e., Delta 7.1.4) to (effectively) zero, the second is generally just a question of setting sufficiently tight simulation convergence criteria (described next), while the third is best obtained by making use of AUTOK (9.3.2).

In addition there is almost always a fourth condition which is that the network must be well coded such that, even if there are no fatal or semi-fatal errors detected by **SATNET**, certain Serious Warnings and/or Non-Fatal Errors (mostly involving X-turns at signals) are removed.

To obtain extremely good simulation convergence it is normally sufficient to specify the **minimum** number of simulation iterations, NITS\_M, to, say, 5. Otherwise, as one nears global convergence and the flows being simulated are relatively stable, the normal simulation convergence criteria will be met after possibly 2 or 3 iterations and, to go beyond that, extra iterations are required.

It may also be useful, if not using OBA, to set NITA\_M, the minimum number of assignment iterations to, say, 3 or 4, otherwise the assignment may stop after a single iteration which does not allow a sufficient improvement in the assigned flows to take place.

Finally, use of AUTOK guarantees optimum combinations of successive assigned flows and the most rapid approach to global equilibrium.

Currently, it is probably safe to say, most networks are not run to anywhere near their potential convergence levels – all it needs is a bit of ambition, confidence and application of the above rules. Go for it!

### 9.5.4 Reduced CPU

Various “tricks” exist to try to minimise the cpu time required by **SATALL** to reach the required level of assignment-simulation convergence, particularly for “large” networks where run times become a practical consideration. Most of these methods are based on the empirical observations that, certainly for large networks, the assignments take much more cpu than the simulations (since the number of assignment calculations is roughly speaking proportional to zones times links whereas the simulation is proportional to links only).

**(1) Increased MASL, decreased NITA** - The first suggestion is, by using the DIDDLE = T option to continue one assignment from the end point of the previous assignment loop (see 9.4), the full assignment is made up of (in effect)  $NITA * MASL$  all-or-nothing iterations with the simulation introduced after every NITA iterations in order to update the cost-flow curves. By decreasing NITA and increasing MASL in proportion the same number of total iterations may be achieved in roughly the same cpu time (any increases in cpu will be due only to the extra simulations) and with - hopefully! - the same overall convergence. Thus instead of setting, say, NITA = 50 and MASL = 20 we would recommend using NITA = 10 (or even 5) and MASL = 100 in the hope that convergence would be achieved in far fewer than 100 loops.

**(2) Use of UPDATE** - Alternative suggested methods of improving cpu are to make greater use of the UPDATE facility (see 15.3) or to use the perturbation techniques available within the path-based algorithms (see 21.3).

**(3) Sliding Convergence Criteria** - UPDATE may be particularly effective in situations where one basic network is being continuously edited (calibration, validation, etc.) and one version is much the same as the previous version. One useful “trick” here is to gradually “tighten up” the convergence criteria as the network nears its final state. For example, start with ISTOP = 80% and gradually increase in steps of, say, 5% as your network coding gets better and better. There is no point in carrying out a very long, very accurate assignment to a network which still has glaring coding errors. A similar principle is also used by CASSINI in order to minimise CPU time for supply-demand feedback loops; see 15.54.

**(4) AUTONA** – Setting a parameter AUTONA = T (default F) in &PARAM in the network .dat file allows **SATALL** to select “optimum” values of NITA and/or UNCRTS at each assignment based on: (a) the best GAP value to date and (b) the “history” of how delta and/or epsilon varied with iteration number in the previous assignment. The idea is that if, say, the current best GAP value is 1.0% then there is probably very little point in having a highly accurate assignment down to delta/epsilon values of, say, 0.001% since, as soon as we change the cost-flow relationships with the following simulation, the GAP value will increase over Epsilon. A more realistic objective is to set a “target” of, say, Epsilon = 0.01% for the assignment which is one tenth the value of the current GAP. By analysing the “history” from the previous assignment we can estimate how many iterations should be required to reach this value and set NITA accordingly (subject to the requirement that is  $\leq$  the global maximum value of NITA and  $\geq$  NITA\_M). Empirically using AUTONA = T has been found to significantly reduce CPU times.

In 10.9 AUTONA also automatically controls the assignment stopping criterion UNCRTS (see 7.1.5) in addition to NITA by reducing it to a value comparable to the current best GAP value. (The effect of reducing UNCRTS is generally to **increase** the number of assignment iterations in order to achieve a sufficiently well converged assignment).

The same principles of “relaxed convergence” are also used by CASSINI in order to minimise CPU time for supply-demand feedback loops; see 15.54.

Note that AUTONA cannot be used with any form of elastic assignment.

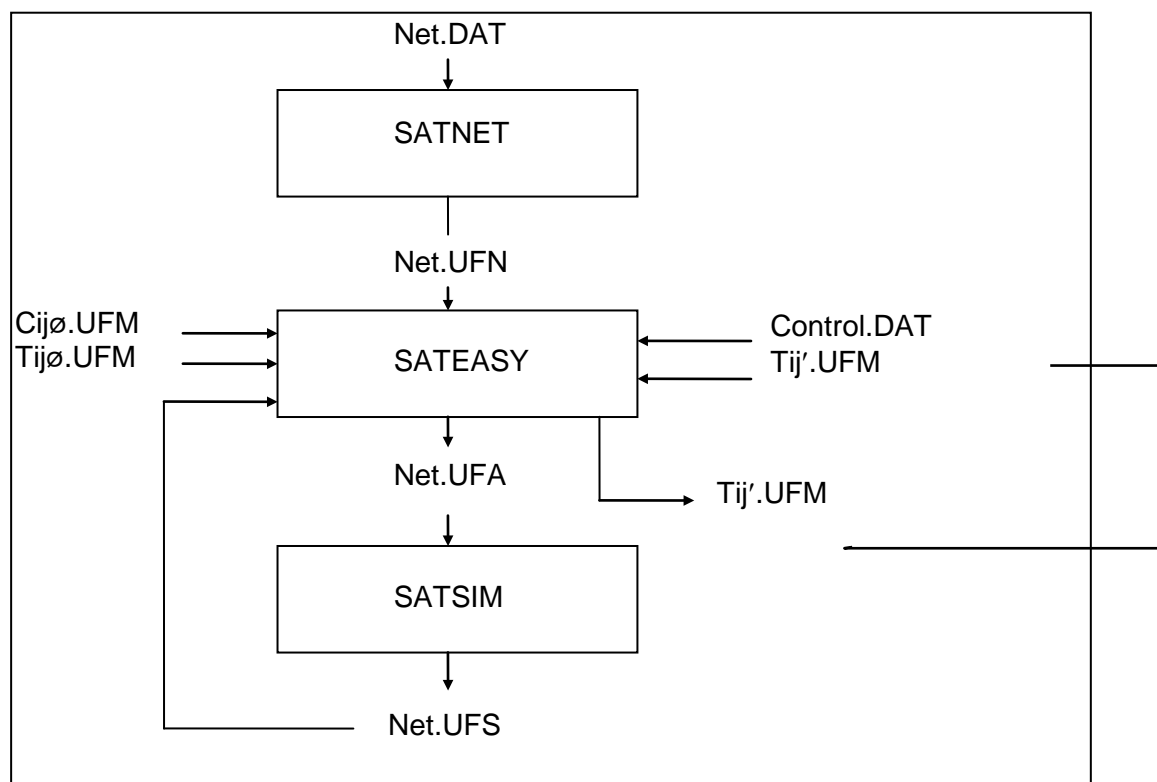
## 9.6 Elastic Assignment/Simulation Loops

If the assignment is based on an elastic demand matrix as opposed to a fixed trip matrix (see 7.4), the same principle of looping between assignment and simulation as illustrated in Figure 9.1 still applies, the only difference is that the assignment not only changes the link flows, it also changes the trip matrices. However the same loop convergence criteria as listed in 9.2.1 still apply - if the link flows from one loop to the next (or equivalently the cost-flow curves) are unchanged then the process has converged.

**SATALL** therefore proceeds in a virtually identical fashion with elastic assignment as with inelastic. Details of the changes required are given in 9.10.

Whether or not an elastic loop will converge better than an inelastic loop is difficult to say in advance; it is probably a question of “horses for courses”. Thus including trip matrix variability probably complicates matters; however the lower level of congestion to be expected with elastic assignment probably simplifies matters. Experience to date is limited.

For completeness we briefly consider here the alternative "DIY" approach of explicit loops between **SATEASY** and **SATSIM** as illustrated below with additional input and output files (although, see below, the use of **SATALL** is strongly recommended).



Thus starting from a network data file net.dat fed through **SATNET** the process starts with an initial elastic assignment using **SATEASY**. This requires input reference matrices  $c_{ij}^0$  and  $T_{ij}^0$  in order to define the demand function (best defined within net.dat; see 7.12.3). An initial estimate of the road trip matrix,  $T_{ij}'$ , may or may not be available. However on subsequent iterative loops the output trip matrix  $T_{ij}$ .ufm may be “re-cycled” back to the subsequent elastic assignment with REDMEN = T.

**SATSIM** is run in order to update the link flow-delay curves based on the elastic link flows and loops through the elastic assignment. Parameters MASL, ROSIE and KOMBI (but not DIDDLE or AUTOK) may be used to control convergence.

The DIY nature of this approach is further reflected in the fact that there are no dos-style bat procedures available to simplify the loops. This is left to your ingenuity!

However it needs to be strongly emphasised once again that an elastic assignment loop is much better undertaken using **SATALL** as described in Section 9.10, which is run as a single program and does not require any complex DIY procedures.

## 9.7 SATALL: General Functions

The program **SATALL**, first introduced with **SATURN** 9.1, in effect combines the programs **SATASS/SATEASY** and **SATSIM** into a single program and carries out a full assignment/simulation convergence loop internally. Thus, as shown in Figure 3.1, taking as input a .ufn network file built by **SATNET**, it both assigns and simulates so as to produce an output file containing converged assigned flows plus the corresponding simulated delays.

By combining two programs into one **SATALL** should be both faster and, ultimately, “more clever” in terms of the steps that can be introduced in order to improve the rates of convergence. For example it can combine the DIDDLE option with elastic assignment which is otherwise impossible; see 9.10. All the various distinct options for assignment and/or simulation that can be invoked with the separate programs **SATEASY** and **SATSIM** may now be carried out within **SATALL**. Indeed, as with the elastic DIDDLE (what a great name!) mentioned above, there are extra options only available with **SATALL**; see 9.12.

## 9.8 SATALL Run-time Convergence Statistics

The rates of convergence of the assignment-simulation loop as well as the internal convergence of both the simulation and assignment sub-stages are displayed during run time in three parallel “windows”. In all three cases, more complete convergence statistics are given within the .LPT file; see 9.9. The statistics provided within each “window” are as follows:

### 9.8.1 Assignment - Simulation Convergence

- (i) The “loop” number.
- (ii) The percentage of links whose flows differ by less than 5% (or, strictly speaking, PCNEAR %) between successive assignments; see 9.2.
- (iii) The average GEH parameter (see 15.6) indicating the differences in demand flows per link between successive assignment loops.

Note that both measures i) and ii) are used to determine stopping criteria; iii) is provided for information only. Convergence is reached as ii) goes to 100% and iii) goes to zero.

### 9.8.2 Assignment Convergence

The statistics displayed depend on the precise assignment option used. For Wardrop-based assignment the window displays:

- (i) iteration number;
- (ii) the lambda or step-length value (see 7.1.2);
- (iii) the delta function (see 7.1.4).

Of these only i) and ii) determine stopping conditions; iii) goes to zero as one approaches convergence.

For stochastic-based assignments (see 7.2.5):

- (i) iteration number;
- (ii) total pcu-costs on each assignment;
- (iii) the root mean squared differences in flows between successive assignments.

Only i) is used as a stopping criteria; ii) stabilizes and iii) goes to zero at convergence.

### 9.8.3 Simulation Convergence

The simulation statistics include:

- (i) the iteration number;
- (ii) the average absolute change in OUT profiles in pcu/hr (see 8.3);
- (iii) the number of simulation junctions simulated;

where, under iii), junctions are only simulated on a particular iteration if there has been a “significant” change in the flow profiles into that junction. A decreasing number of simulated junctions implies convergence.

Statistics i) and ii) determine the simulation stopping criteria; see 8.3. At convergence ii) approaches zero.

## 9.9 SATALL Convergence Statistics: Full Line Printer Listings

The line printer (.LPT) output file contains fully comprehensive statistics illustrating the convergence of both the individual assignment and simulation stages plus the loops and includes all the “window” data described in 9.8. To reduce the size of the file these are given in tables as far as possible.

### 9.9.1 Assignment - Simulation Loop Convergence

#### 9.9.1.1 Global Measures

At the end of each simulation a number of different loop convergence indicators are listed. A further summary table giving a sub-set of these statistics by iterative loop number is given at the end of the complete set of loops and may also be obtained as part of the convergence statistics output by **SATLOOK** (11.11.8) and/or **P1X**.

1. The percentage of links “PCOK” whose (demand) flows differ by less than 5% (or, strictly speaking, PCNEAR%) between successive assignments; see 9.2.1. The distribution of PCOK for a standard set of (in effect) PCNEAR values from 0.5% up to 50% is also given. In addition Table L(9) – see below - lists the 10 “worst” links.
2. Further comparison statistics of flow differences per link between the last two assignments, e.g.:

MEAN GEH STATISTIC =	0.83
MEAN ABSOLUTE DIFFERENCE =	15.96 %
RELATIVE MEAN ABS DIFFERENCE =	3.65 %
RELATIVE MEAN STANDARD DEVIATION =	7.37 %

3. Compare the turn delays as estimated by the assignment with the ‘correct’ delays calculated by the simulation; at convergence the two should be identical, e.g.:

MEAN SIMULATED TURN DELAY =	31.75 secs
MEAN ABS. DIFF. IN ASS/SIM DELAYS =	19.38 secs

RELATIVELY = 61.03%  
 NUMBER DIFFERING BY < 5.0% OR 1 96  
 SECOND =  
 RELATIVELY (PCOK) = 78.05%

The distribution of PCOK above as a function of the critical % is also given here while the 10 “worst” turns are listed separately in Table L(8) – see below.

4. The assignment/simulation gap function; see 9.2.1.
5. The assignment simulation variational inequality measure; see 9.2.1. NB: Measures 4) and 5) do not appear under certain options, eg stochastic or elastic assignment, where they are not relevant.
6. Eight standard simulation summary statistics between two successive iterations, as illustrated below.

#### CONVERGENCE OF BASIC SIMULATION TOTALS:

	Previous Iteration	Current Iteration	Difference	%
PRIMARY STOPS	14733.07	14164.58	568.49	4.01
SECONDARY STOPS	7805.46	6488.73	1316.74	20.29
TRANSIENT DELAYS	86.03	83.19	2.84	3.41
QUEUING DELAYS	231.75	217.65	14.10	6.48
CRUISE TIMES	136.83	135.99	0.84	0.62
TOTAL TIME	454.61	436.83	17.78	4.07
PCU-DISTANCE	4834.87	4816.13	9.74	0.39
SPEED	10.64	11.03	-0.39	-3.54

Note that whereas certain global statistics such as the pcu-distance may appear to stabilize rapidly others, such as those related to delays and stops, are considerably more variable.

7. List of the (up to) 10 largest changes in Blocking Back Factors used on the current and previous loops plus an r-squared value comparing the same factors.

### 9.9.2 Disaggregate Measures (10 Worst Tables)

Assignment-simulation statistics may also be calculated and displayed at the level of individual nodes, links or turns: for example, the difference in turn delay as calculated from the flow-delay curves on the previous assignment with that from the latest simulation.

Further disaggregate measures include “gap”, capacity and flow differences. The “Gap” is defined to be difference between the assigned and simulated delays (as above) multiplied by the demand flow. It is therefore similar, but not strictly identical, to the route-based contributions to the assignment delta values (Eqn. 7.3, section 7.1.4) and/or simulation-assignment gap function (9.2.1). Capacity differences are simply the differences in turn capacity as calculated on two

successive simulations (with an assignment stage in between) while flow differences are the differences between two successive assignments.

Two **SATALL** tables which relate to the convergence of the assignment – simulation loops at a disaggregate level are given in slightly different locations in the .lpt files.

Firstly, at the end of each simulation Table L(8) lists, in decreasing order, the 10 turning movements whose current simulated delays differ most from those calculated from the time-flow curves used in the previous assignment (as given in aggregate terms under (3) above). These differences indicate where the assignment and the simulation “disagree” in terms of how to define delays even though the demand flows are identical; this “disagreement” is the main cause of “gap” values which exceed the “delta” values; see 9.2.1. To help in identifying the likely cause of these differences the current and previous capacities plus the (demand) flows are listed.

Secondly, at the end of each assignment stage, Table L(9) lists the 10 (assignment) links whose (demand) flows differ most in terms of GEH statistics between the latest assignment and that on the previous loop.

The same two tables may also be displayed interactively within **P1X** (see 11.15) which also optionally offers two additional “10 worst” tables in terms of (a) “gaps” and (b) capacities.

In addition the various turn-based convergence statistics, e.g., delays, gaps, etc. may be generated and displayed within **P1X** as either turn or link annotation data which may, in turn, be saved as **SATDB** data columns. In the latter format they may be displayed in a window with the data ordered by, say, decreasing absolute value so that it becomes possible to view not just the 10 worst examples but the full list of worst examples.

Finally disaggregate convergence statistics may also be displayed per simulation node within the standard node text tables within **SATLOOK** (11.11.1).

### 9.9.3 Assignment Convergence

A summary table is given at the end of each assignment, the precise contents of which depend on the assignment technique used (e.g. Wardrop vrs Stochastic, elastic vrs fixed trip matrix, etc). For the simplest case of Wardrop Equilibrium the following measures are given for each iteration:

- ◆ LAMBDA: The “step length” at each Frank-Wolfe step; see 7.1.2.
- ◆ FRACTION: The ultimate fraction of trips assigned to this iteration; see 7.1.2.
- ◆ C1: The total cost associated with the current flows and the current costs.
- ◆ C2: The total cost if all trips could be assigned to the current minimum cost routes.
- ◆ DELTA =  $(C1 - C2)/C2$ ; see 7.1.4.
- ◆ Z: Current value of the objective function;

- ◆ DZ: The improvement to the objective function in this iteration.
- ◆ FDZ = DZ/Z (as a %), the fractional improvements in the objective function.
- ◆ ZLB: A lower bound on the objective function; see 7.1.5.
- ◆ ZULB: The upper lower bound on the objective function; see 7.1.5.
- ◆ EPS: Epsilon, the current “uncertainty” in the objective function; see 7.1.5.
- ◆ FIMP: The improvement in Z relative to epsilon; see 7.1.5.

### 9.9.4 Simulation Convergence

A standard summary table is given at the end of each simulation, listing for each iteration:

CC	average absolute change in the OUT profiles; see 8.3.
NO SIM	number of nodes simulated
NO BB	Numbers of links which block back;
SUM	Sum of absolute differences between queues and stacking capacities
NO >	links with queue greater than the stacking capacity
SUM >	total excess PCUs
NO <	Links with queue less than the stacking capacity
SUM <	total spare PCUs
NO =	Links with queue equal to the stacking capacity (to +- 0.1 PCU)
ABBC	The average absolute change in blocking back factors

Thus “No BB” has been subdivided into 3 categories, “NO =”, “NO <” and “NO >”, as has “SUM”.

### 9.10 Elastic Assignment within SATALL

Elastic assignment within **SATALL** is carried out in much the same way that it is carried out in **SATEASY** with the obvious proviso that it is part of the outer simulation/assignment loop, not an isolated elastic assignment.

Thus the same control parameters are used; eg MCGILL >0 designates the form of elastic demand function. BETA and POWER are elasticity - related parameters, etc. Similarly the share-based demand functions, the extended logit models (7.6) and the distribution models (7.10) may all be invoked within **SATALL**. All these and the necessary file names may be set either in the original .dat file - highly recommended, 7.12.3 - or input/re-defined in the **SATALL** control file.

There are, however, certain differences between **SATALL** and **SATEASY** as listed below.

On loops after the first **SATALL** always uses the REDMEN option of starting the latest elastic assignment with the estimate of the trip matrix from the previous assignment. The reasons for doing so are (a) it almost certainly helps convergence and (b) the previous trip matrix is already stored internally so no user intervention to define the matrix is required.

Hence if REDMEN = T and an estimated trip matrix is set in the original .dat file (or otherwise) this is only used on the very first elastic assignment. (NB. This is not a reason not to invoke REDMEN since using a good estimate of  $T_{ij}$  on the first assignment is still a very good thing).

**SATALL** can use the DIDDLE option such that, if DIDDLE = T, any inelastic assignment after the first will commence with the initial set of link flows equal to the flows from the previous assignment. This is very similar to the REDMEN option, the difference being that REDMEN specifies, in effect, the initial flows on the pseudo links while DIDDLE specifies the flows on the real links. Empirically using DIDDLE appears to improve convergence significantly.

## 9.11 Multiple User Class Assignment within SATALL

Multiple user class assignment within **SATALL** is essentially no different than with the separate assignment steps within **SATEASY**. Follow the instructions in 7.3.

Note that a separate set of assignment/simulation flow convergence statistics is given for each user class; i.e. items (1) and (2) as described in 9.9.1.

The same applies for elastic multiple user class assignment; see 7.9.

## 9.12 Special SATALL Extensions

We describe here a set of special “extensions” to the standard assignment - simulation procedures within **SATALL**.

### 9.12.1 Continuation Runs with SATALL

It is a frequent problem that having run a network through **SATALL** over, say, 20 assignment - simulation loops, you find that what you really wanted was to do 21 loops. An obvious solution is to change the convergence parameters in the original file, e.g. MASL to 21, and re-run but on large networks this is potentially very time consuming.

Alternatively the command

**SATALL net MASL 1**

will take the file net.ufs (which has been through 20 loops, say) and carry out one more simulation-assignment loop. The output file will also be named net.ufs and therefore over-writes the input file.

Using a parameter MASL 5 will run (up to) 5 extra loops; the actual number may be less if the loops terminate on the ISTOP criteria rather than MASL. It will, however, always carry out one additional loop even if the original ISTOP criterion was satisfied. One may circumvent this “problem” by using both a MASL parameter and the KR parameter to define a new control file which increases the

ISTOP value. Alternatively one can edit the network .ufs using **P1X** (11.9.11) to change parameter values such as ISTOP prior to the continuation run.

Strictly speaking the MASL n option increments the existing value of MASL by n; it does not guarantee that exactly n extra loops are run since, as mentioned above, the number of loops may be terminated by other criteria such as ISTOP. For example, if the original value of MASL was 20 but the loops stopped after 10 due to ISTOP, then using MASL 5 on the command line and resetting ISTOP to a higher value may actually result in 15 extra loops since the new value of MASL will be 25. In principle it should be possible to set up the continuation option such that “MASL 5” implies “run exactly 5 extra loops” or that it means “set MASL equal to the current number of loops plus 5”. But it doesn’t do that – it does what it says on the tin!

Note that in this case the output network file, net.ufs, has the same name as the input network file, i.e. it overwrites it. This creates a problem since both versions of the file need to co-exist at the same time so, to avoid this, the input file is first copied into net.ufn and that file is used as input. A consequence of using this option is that an existing net.ufn file will itself be over-written.

Clearly this facility depends on the network having a simulation component; a similar continuation option for buffer-only networks is described under WIDDLE in 7.11.6.

### 9.12.2 Outer Signal Optimisation Loops (NIPS, SATOFF and/or SIGOPT)

Prior to version 10.5 signal settings (either stage times if SIGOPT = T or offsets if SATOFF = T) could be continuously optimised during the assignment / simulation loops; i.e., once per loop. However, as explained in 15.31.1, this is almost certainly not a very realistic procedure for setting green times since it almost certainly leads to an overly optimistic view of network performance, nor is it very efficient in terms of cpu time.

Thus in 10.5 an option has been introduced such that the optimisations only occur at the **end** of a fully converged simulation / assignment sequence, e.g., at the end of MASL loops. At this stage the stage times and/or offsets are optimised and the simulation /assignment loops re-started until convergence is again achieved. The “outer-outer” loop is repeated NIPS times, where NIPS is a parameter set under &PARAM in the original network .dat file.

Generally speaking it is recommended that NIPS should be a small number, e.g., 1 or 2, since a very large value will simply repeat the problems noted earlier. If NIPS = 0 the original method of continuous optimisation is followed. N.B. The original default value of 0 was chosen primarily to retain upwards compatibility and was not particularly recommended; post 10.9 the default was changed to 2.

We note that, if the optimisation changes are relatively small (as is generally to be expected) then subsequent should converge very quickly. There is a good case, therefore, for reducing MASL in proportion to NIPS (or, strictly NIPS+1). For example, to carry out a maximum of 60 simulation/assignments with NIPS = 2, then set MASL = 20 as that will give 20 loops followed by the first optimisation, 20 more followed by the second and 20 more to follow – a total of 60.

### 9.12.3 Assigning a Nil Matrix: The ZILCH Option

A new option, first included in version 10.6, allows the user to run **SATALL** without actually assigning any trips. In this case the only network flows will be those included as “fixed flows”, e.g., bus routes, pre-load flows, etc.

To request this option set the parameter `ZILCH = T` either within `&PARAM` in the network `.dat` file or within the **SATALL** control file.

If `ZILCH = T` **SATALL** basically runs through a single loop with, in effect, no assignment stage apart from a load of the fixed flows followed by a single simulation. In effect `MASL = 1`. Clearly no feedback is necessary since the simulated delays have no impact on the flows.

At first glance this may seem a bit of a silly option – why have an assignment model that doesn’t actually do any assignment? However there are a number of circumstances in which it could be useful.

For example, one might wish to cordon off a segment of a network (via **SATCH**) and run the simulation with the identical flows as in the “master” network since extracting and re-assigning a cordoned trip matrix (a) takes time and (b) is not guaranteed to give identically assigned flows. To do this the master network must be introduced as a pre-load network (see 15.5) to the cordoned network (with some care being taken that any bus flows etc. are not double-counted – more on that one later). Note as well that the pre-load flows for a cordoned network must be loaded via a **text file**, not a `.ufs` file, since the network structure will have changed under cordoning; see 15.5.4.

Equally one might wish to simulate traffic flows extracted from a different suite of programmes entirely within **SATURN** and transferred as a text file. (Recall the use of text files to define pre-load flows, not just `.ufs` files; 15.5.4.) Or, similarly, one might wish to test the effect of “pre-scheme” flows on a “with-scheme” network so that the network may have been altered but the link flows themselves remain unchanged.

Another application would be to do a genuine zero load in order to calculate delays at zero flow in order to accurately measure the effects of “congestion”. (For example, traffic signals always give some delays even with zero flows because of the finite red times so one might wish to subtract these delays from the “actual” delays.)

Note that because no assignment takes place certain options such as `SAVEIT` etc. are ignored and there will be no route information output. Equally any form of elastic assignment is ignored. On the other hand if there are multiple user classes flows (if any are input via pre-load) are retained.

### 9.13 The SATALL Batch Procedure

The program **SATALL** may be run by typing:

```
SATALL network trips      (KR control COST cost REDMEN tij1 TIJ tij2
FREEZE ice MASL n RESTART
```

where:

Network.UFN	Input network UF file from SATNET
Network.UFS	Output network file
Network.UFC	Output costs per Frank-Wolfe iteration under SAVEIT (15.23.1)
Network.LPT	Output line printer file
trips.UFM	Input trip matrix file (Optional)
control.DAT	Ascii Control file (Default: SATALL0.DAT)

Files used for elastic assignment:

cost.UFM	Input cost matrix $C_{ij}^0$
tij1.UFM	Estimated road trip matrix (REDMEN = T)
tij2.UFM	Output road trip matrix
ice.UFM	Input frozen cell matrix (ICING=T)
MASL n	Re-run with n more assignment/simulation loops (see 9.12.1)
RESTART	Input the previous network.ufs file; see 15.4

(N.B. Use of M 21 cost/M 22 tij1/M 23 tij2/M 24 ice also works)

If the KR option is not invoked on the command line, then **SATALL** expects to find the parameters in the default control file SATALL0.DAT. (See 9.15.2.)

Note that the input file has the “new” extension .UFN and that the output file is always .UFS whether or not the network is pure buffer or not. If network.UFN does not exist but network.UFS does then it is renamed.

## 9.14 The SATURN Batch Procedure

The special DOS .bat file, **SATURN** (also known as **SATURN9**), has been provided to run the complete set of network building (**SATNET**) plus assignment/simulation (**SATALL**) operations in sequence.

In the simplest case, where the trip matrix and all other file names have been defined within the network .dat file (recommended), use:

**SATURN** network

In more general terms to invoke a complete run use:

**SATURN** network trips (UPDATE net1 PASSQ net1 PS post

Files:

network.DAT	Input <b>SATURN</b> network data file
trips.UFM	Input unformatted trip matrix file

(Optional - not necessary if defined within network.DAT see 6.3.4)

network.UFN	Output SATURN UF file from <b>SATNET</b>
network.UFS	Output SATURN UF file from <b>SATALL</b>
network.UFC	Output route cost file
network.LPN	Output line printer files from <b>SATNET...</b>
network.LPT	and <b>SATALL</b>
net1.UFS	Input SATURN UFS file for UPDATE or PASSQ (Optional)
post.PS	Over-write file for input to <b>SATNET</b> ; see 17.4.1. (Optional)

Further details of the **SATURN** .bat procedure are given in Section 14.3 and special extensions thereto are described in Section 14.4.

## 9.15 SATALL: Technical specifications

### 9.15.1 SATALL Files

Channel Number	Remarks
1	The input UFN network file from <b>SATNET</b> . (Mandatory)
2	The output UFS file containing the assigned flows and simulated delays. (Mandatory)
3	The output UFC file containing the link costs by iteration number. (Optional: SAVEIT = T)
4	The output UFO file
5	The control file specifying the options and parameters for this run of <b>SATALL</b> as specified below. (Mandatory)
6	The output LPT line printer file. (Mandatory)
8	A scratch UF file to temporarily store iteration costs under SAVEIT. (Optional: SAVEIT = T)
9	Input UFM file containing the trip matrix being loaded (or the “reference” trip matrix $T^0_{ij}$ for elastic assignment). (Mandatory)
10	Scratch UFX files used for both input and output if ...
11	... insufficient RAM to store the trip matrix under ..
12	....multiple user class and/or elastic assignment
13	A further scratch UFX file used under MUC
15/14	Terminal (output only) (Optional (MODET ne 0))
19	Input cost matrix “CGHFIL” used under incremental distribution (MCUBC=1) (Optional)
20	Input cost matrix “CKLFIL” used at the lower level of a nested logit model (MCGILL=5) (Optional)

21	Input “reference” cost UFM matrix $c_{ij}^0$ used under Elastic Assignment (Mandatory under Elastic Assignment)
22	Input initial trip matrix estimate used under Elastic Assignment (Optional under Elastic Assignment (REDMEN = T))
23	Output road trip UFM matrix used under Elastic Assignment: trips by road (Mandatory under Elastic Assignment)
24	Input “freeze” matrix indicating which cells in an elastic assignment are to be frozen (ICING=T); see 7.5.5. (Optional)
28	A scratch UF file used under OBA plus AUTOK or KOMBI
29	Input update .UFS file used under Warm Starts

### 9.15.2 SATALL Control File Data Input

Input consists only of a set of Namelist Parameters associated with the name &PARAM and an optional new network title.

The parameters which may be defined here constitute a sub-set of those described in Section 6.3, more precisely:

- 1) Parameters relevant to the simulation:  
AFTERS, ALEX, CAPMIN, LRTP, MAXDTP, MAXQCT, MYTVV, NFT, NITS, NOPD, NOPMAX, NOTUK, QUEEN, SIGOPT, TAX and TDEL.
- 2) Parameters relevant to the assignment:  
AMY, ASHORT, DIDDLE, ERTM, EXPERT, FISTOP, GONZO, KINKY, KOB, KORN, MCALG, MULTIC, NITA, PARTAN, ROSIE, SAVEIT, SHADOW, SOWHAT, SUET, SUZIE, SUZIEQ, TIJMIN, UNCRTS and XFSTOP
- 3) Elastic assignment parameters described in 7.12.2.  
BETA, BETA\_2, BETA\_D, FRED, ICING, MCGILL, MCUBC, MASL\_F, NITA\_F, NITA\_S, POWER and REDMEN
- 4) Parameters relevant to the simulation/assignment loop:  
ISTOP, KOMBI, MASL, NIPS, NISTOP and PCNEAR
- 5) Miscellaneous parameters:  
MODET, TITLE (see below) and WINDY
- 6) Filenames:  
FILCGH, FILCIJ, FILCKL, FILICE, FILRED, FILTIJ, ROADIJ

#### Network Title

A new descriptive network title may be set by either:

- 1) A namelist parameter of the form:  
TITLE = 'nettitle'
- 1) A namelist designation of a logical variable:

TTLE = T

in which case the new network title is contained on the record immediately following the namelist records (i.e. following &END) and occupies columns 1 to 76.

**Default File**

The default control file SATALL0.DAT is as follows:

&PARAM

&END



## 9.16 Version Control

JOB NUMBER: 5101396		DOCUMENT REF: Section 9.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
3.2	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	19/03/08
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	22/10/09
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	06/12/10
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