A Brief tutorial for LIMS 5.0

P.Simacek, E.M. Sozer and S.G. Advani

Introduction

This tutorial deals with LIMS 5.0 basics. As such it is oriented to isothermal and mostly two-dimensional problems. There are three parts of this tutorial:

- In the part one we will deal with a simple 1D (linear) injection and we will learn what type of inlets are possible, how to trace the inlet information throughout the injection and how to post-process filling data into contour plots. Most of this part is carried over from the tutorial in LIMS 4.0 documentation.
- In the part two we will look at 2D radial injection into a non-homogenous planar region. We will learn how to use LIMS GE graphical interface to set gates and run problems and how to deal with material parameters to simulate racetracking, inserts and how to use "Regions". This part is not covered in the LIMS 4.0 documentation, but it is covered in this leaflet.
- In the part three you have several more realistic meshes (selected to provide reasonable solution time) to play with. This part is essentially an improvisation on a given topic.

Note that a significant part of LIMS functionality is provided by LBASIC scripts. If deeper understanding is desired, the source for the scripts used in tutorial should be studied as well – the scripts are fairly easy to understand, all of them shorter than one screen listing.

Part I

Tutorial files should be in subdirectory *examples\1D*. We will start this tutorial on the part shown in Figure 1, a very simple one-dimensional injection. Fluid injection comes from nodes 1 and 32. The mold is initially vacuumed. Input data for this problem is in *verif_1d_bare.dmp*.

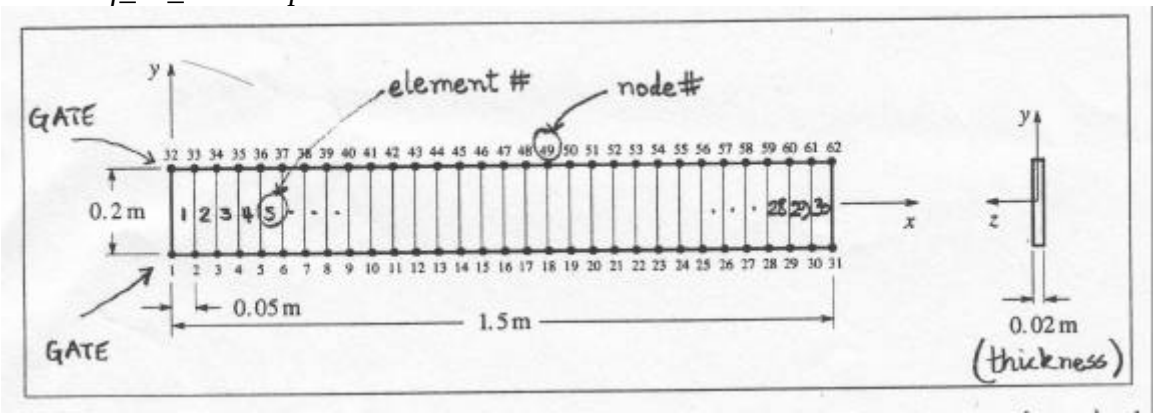

Figure 1.: 1D Problem geometry

The input file contains node and element data, but no results, gates, etc. If you want to see how complete file looks, look at *verif_1d.dmp* or (for IN version) *verif_1d.in*. To simulate the filling of this part we will need to start LIMS and get it to the proper directory. To start LIMS, click on the proper icon on desktop. Now you can type commands as in the usual operating system shell. To change the directory, type

```
##> Changedir "c:/lhome/examples/1D"
```

The simplest filling process uses the input file with gates. The commands to issue would be

```
##> read "verify_1d.dmp"
##> load auto
##> auto

       program outputs filled nodes

##> setouttype "tplt"
##> write "verify_1d"
```

What did happened?

- **`read "verify_1d.dmp"`** reads the input file *verif_1d.dmp*. Answer *yes* (press Y) to restore the simulation settings to keep the gates.
- **`load auto`** reads LBASIC script file *auto.lb* into LIMS. Note no quotes and missing extension – command *load* is directive and its syntax is somehow different than that of *read*. Script file contains LBASIC commands that are executed just as if they were typed from command line. As it comes, the file *auto.lb* contains procedure (macro), that will be stored and available for execution later. Its contents is:

```
proc auto
defint i
do
  solve
  for i=1 to sonumberfilled
    print sonextfilled
  next i
  print sonumberempty()," left"
loop while ((sonumberfilled>0)and(sonumberempty()>0))
erase i
endproc
```

It contains a definition of a single procedure called *auto* that uses no arguments. This procedure solves for pressure and advances the flow front, then it prints all the nodes filled within that step. The whole process is

repeated as long as we manage to fill some additional nodes and there are some empty nodes remaining.

- **`auto`** executed the freshly loaded procedure. You could use **`call auto`** instead.
- **`setouttype "tplt"`** sets the output file format to be TECPLOT file.
- **`write "verif_1d"`** writes the output file to disk. If there is no such file, it will appear in the current directory *C:\lhome\examples\1D*, otherwise the existing file is overwritten. The output can be load to TECPLOT, giving Figure 2.:

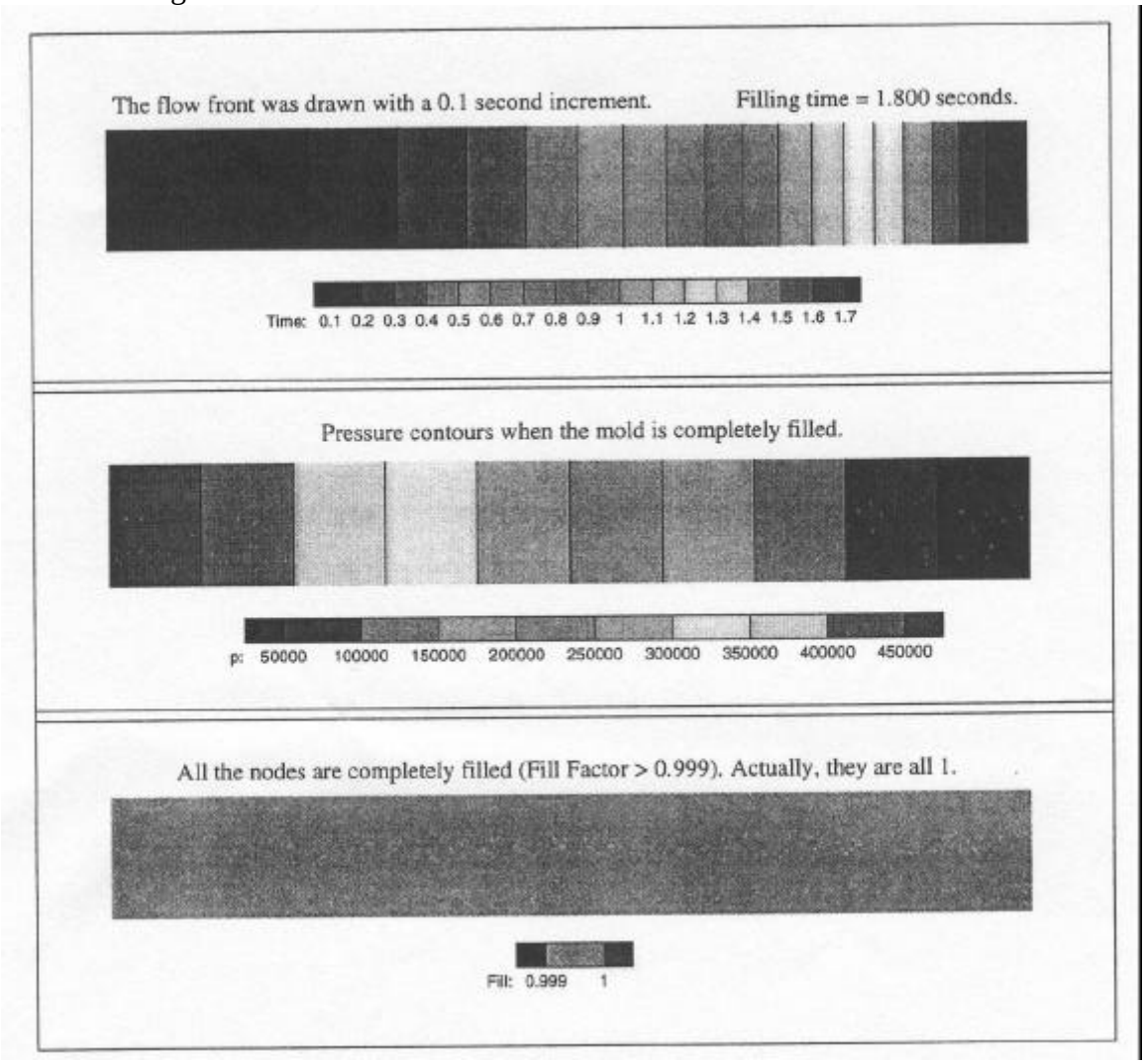

Figure 2..: Results from 1D filling (flow front, pressure, fill factor)

LBASIC script provides flexible way to control the filling simulation, data acquisition and so on. We did not do much controlling in the example above, so lets re-run the problem with some extra user intervention. We will set the gates and record pressure and flow rate at one of the gates. To do this, we will use procedures *rg* (record gate) and *srg* (stop recording gate) that are supplied in file *rg.lb*. To do so, we need to run:

```
##> read "verify_1d_bare.dmp"
##> load rg
##> settime 0
##> setgate 1,1,500000
##> setgate 32,1,500000
##> rg "consp",1
##> auto
       program outputs filled nodes
##> srg "consp",1
```

We have loaded file without gate information this time – thus no questions were asked. Then we had to specify the gates "manually" by *setgate*. The command *settime* sets the current simulation time – otherwise the times would be appended to the previous fill time. The remaining new commands start and stop recording of pressure and flow front at node (gate) 1 and should give you files *consp1p.txt* and *consp1q.txt* in current directory. These files contain time-pressure and time-flow rate data at given node as two columns that can be plotted (for example by TECPLOT), as shown in Figure 3.:

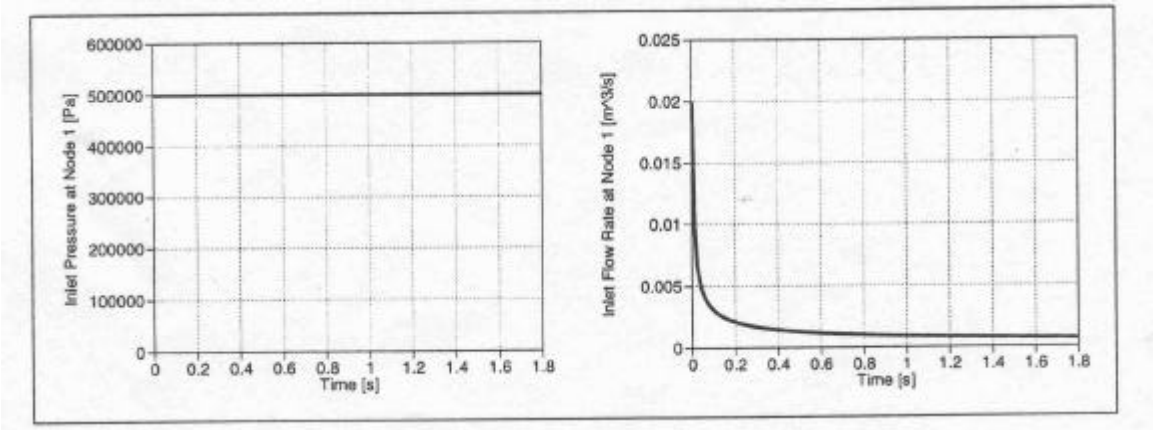

Figure 3: Inlet pressure and flow rate - constant pressure injection

We might want to inject resin in some different way, say using constant injection of rate 0.001 cubic units per second. We may do it as an exercise:

```
##> read "verify_1d_bare.dmp"
##> settime 0
##> setgate 1,2,0.0005
##> setgate 32,2,0.0005
##> rg "consq",1
##> auto
       program outputs filled nodes
##> srg "consq",1
```

Note that we do not need to load any scripts – they are already loaded. Now we should have pressure and flow rate information for constant volume injection in files *consq1p.txt* and *consq1q.txt* (Figure 4).

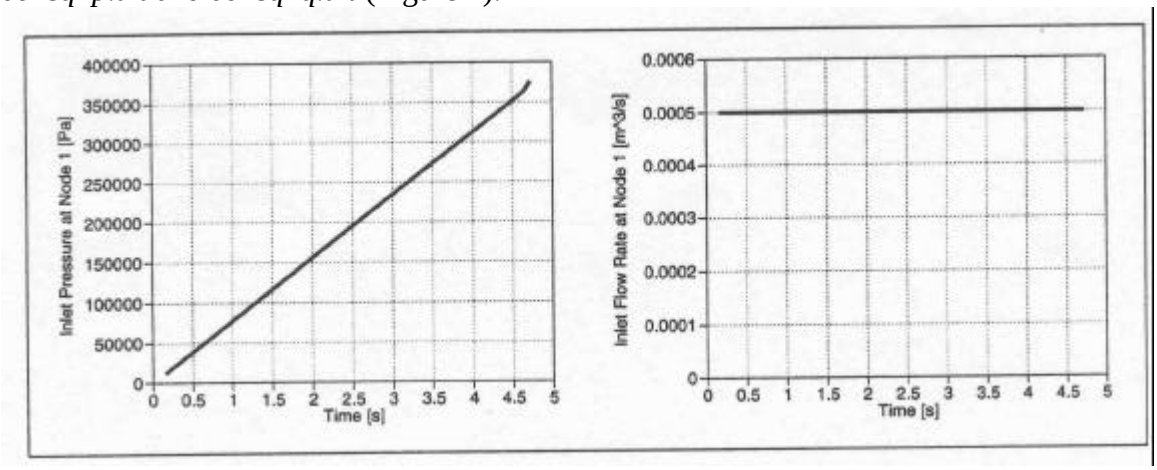

Figure 4.: Inlet pressure and flow rate - constant flow rate injection

There is one more built-in type of injection gate: one where flow rate depends on the injection pressure – this might be useful when the hydraulic pressure loss in the tubing is significant. Assuming $Q_{inlet}=0.0005-1.0x10^{-9}$ P_{inlet} we can utilize this type of gate as follows:

```
##> read “verify_1d_bare.dmp”
##> settime 0
##> setgate 1,6,0.0005,-1E-9
##> setgate 32,6,0.0005,-1E-9
##> rg “mixed”,1
##> auto
       program outputs filled nodes
##> srg “mixed”,1
```

The results are in Figure 5.

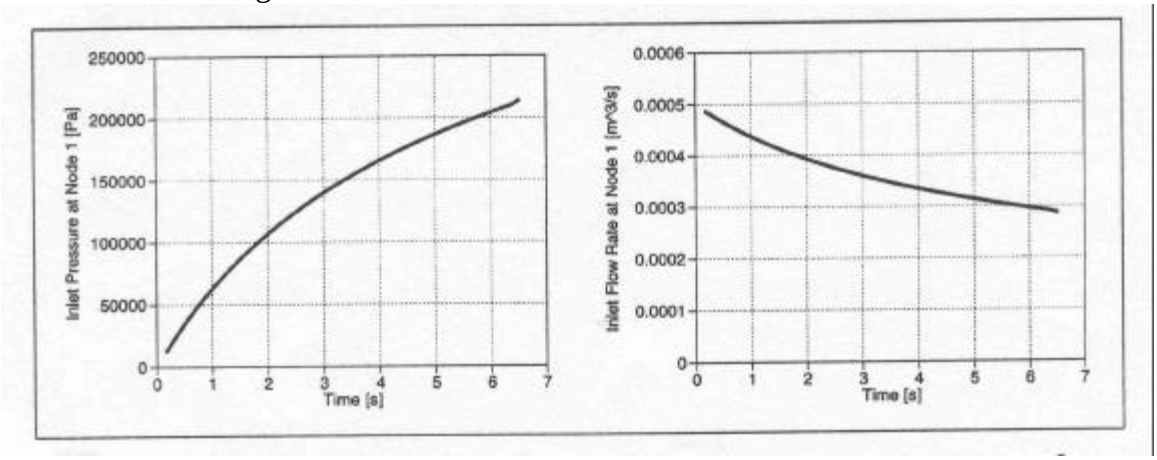

Figure 5.: Inlet pressure and flow rate - mixed boundary condition

This covers the built-in injection mechanisms, but we can use the scripting to add more sophisticated ones into our repertoire. Suppose that we inject at constant flow rate, say those $0.0005 m^3/s$. However, the pump is not capable to do this infinitely – there is bound to be a limit on pressure, too, possibly enforced by some safety valve. Lets set it to 250 kPa. Thus, whenever the pressure is under 250 kPa, pump works as a constant flow device. If the required pressure is higher, it turns into constant pressure pump supplying resin at limit pressure and lower flowrate.

The code that accomplishes this is within *kamgate.lb*. The procedure to call is *kgfill* and accepts, as arguments, desired flow rate, limit pressure and list of nodes that are supposed to serve as gates. Thus, the whole process can be simulated by entering

```
##> read "verify_1d_bare.dmp"
##> settime 0
##> setgate 1,1,500000
##> setgate 32,1,500000
##> load kamgate
##> rg "kaman",1
##> kgfill 0.0005,250000,1,32
##> srg "kaman",1
```

To find out how *kgfill* does it job you might want to look at the file *kamgate.lb*. Essentially, the filling loop checks every time whether pressure exceeds the given value. If this is the case, the constant flow rate gate is overwritten with constant pressure gate. Vice versa, if flow rate at constant pressure gate exceeds given flow rate, gate is set back to constant flow rate, though this is unlikely to happen unless the filling scheme is very complicated. The gate pressure and flow rate for our 1D case is in the Figure 6

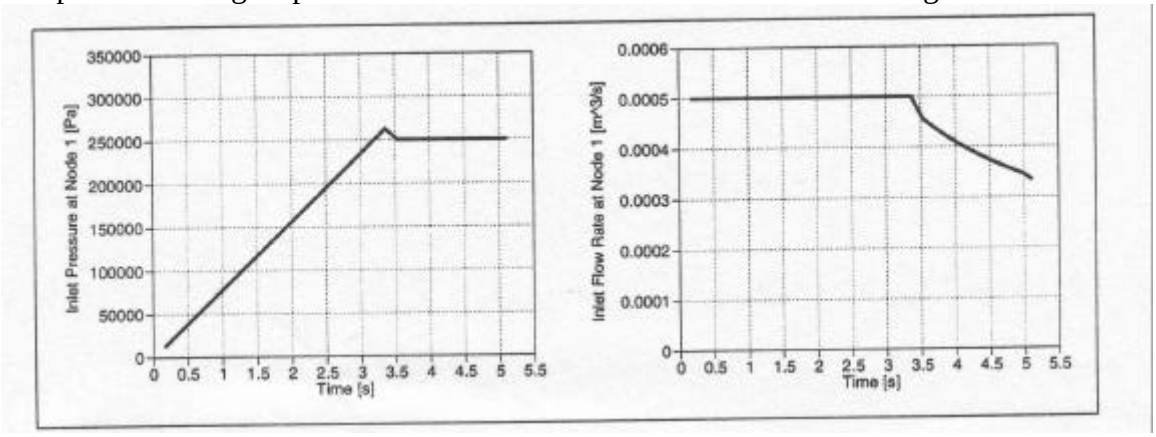

Figure 6.: Inlet pressure and flow rate - constant flow rate with pressure limit (Kaman)

Other things might be controlled during injection as well. We might want to open new gates or close the old ones as the filling progresses. Simple case is the sequential injection: new pressure gate gets opened when the associated node gets filled. This is done by another script, *owfgate.lb*, in procedure *owffill*. This procedure takes as its first argument the gate pressure and then a list of nodes, which serve as "open-when-filled" gates. Lets inject sequentially (from nodes 16 and 47) into our 1D mold:

```
##> read “verify_1d_bare.dmp”
##> settime 0
##> setgate 1,2,0.0005
##> setgate 32,2,0.0005
##> rg “Open”,1,16
##> owffill 500000,16,47
##> srg “Open”,1,16
```

The pressure and flow rate at nodes 1 and 16 is in the Figure 7. Most notably, the filling time was substantially reduced. Note also that while we did not switch off the original gate when we opened the new one (can be done), the flowrate from this gate went to 0 anyway. It may not work this way in more complex parts.

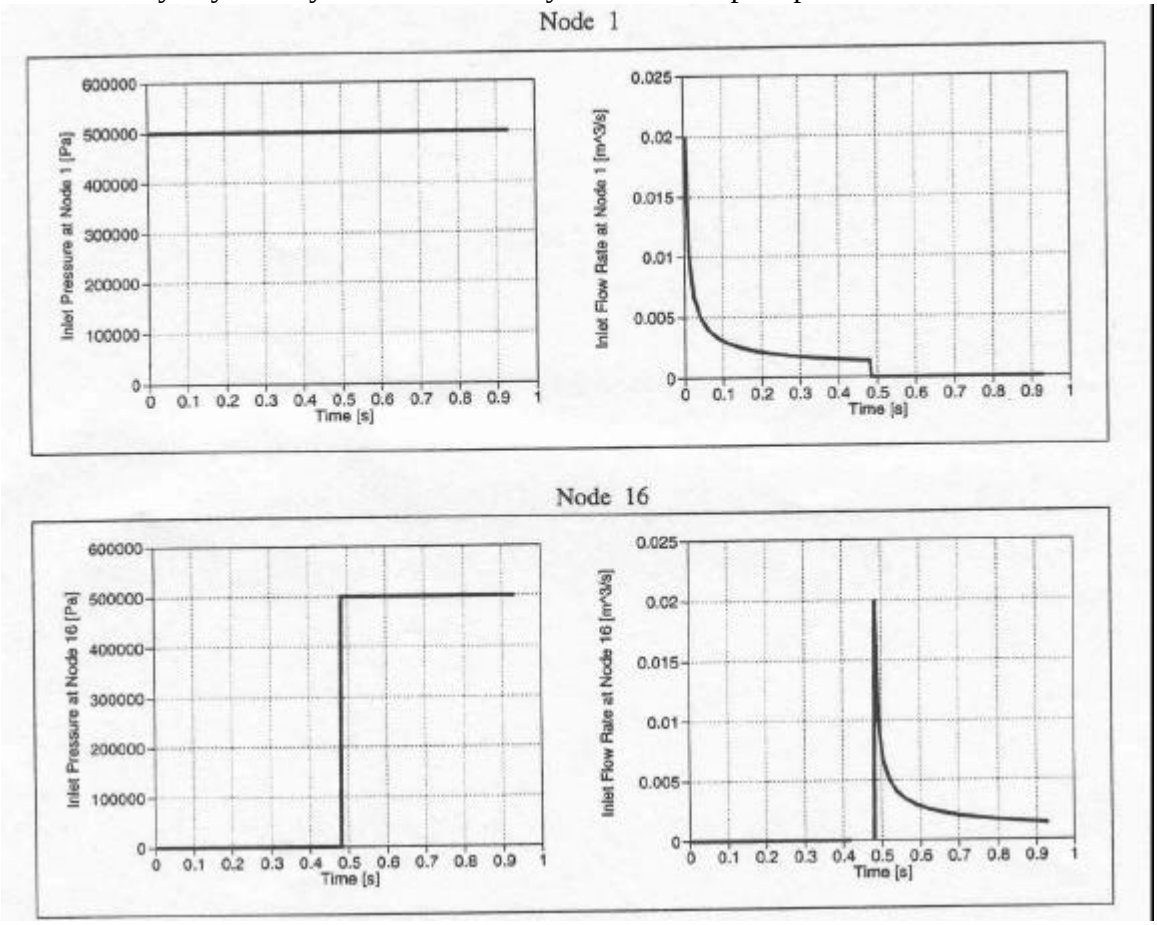

Figure 7.: Pressure and flow rate - multiple ports, sequential constant pressure injection

Part II

The two-dimensional problem to be studied is simple central injection into a plate (file *verif_2d.dmp*). The material of the plate is anisotropic and moreover non-homogenious – the upper part material differs from that of the lower part. The injection port is originally at the node 201 on the right edge (symmetry is being used). The drawing of this part is shown in Figure 8.

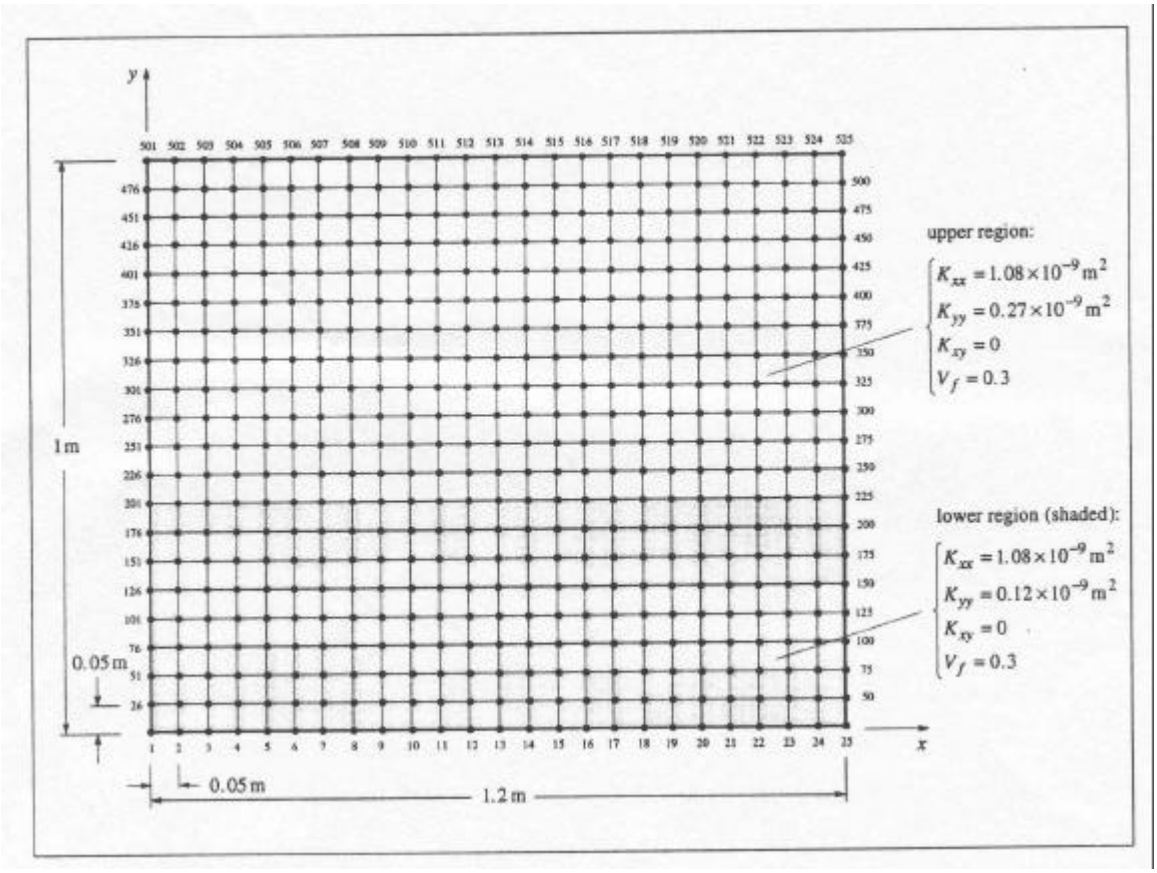

Figure 8.: 2D central injection problem schematics

Restart LIMS or *forget* all script words first. Then change the LIMS directory into *examples\2D*. Next, you might enter, similarly to 1D problem above:

```
##> read “verify_2d.dmp”
##> load auto
##> auto
##> setouttype “tplt”
##> write “verify_2d”
```

The resulting flow-fronts, shown in TECPLOT, are in the upper part of Figure 9. The bottom part of the figure shows the flow rate at gate (node 201). Try to obtain the data (using *rg* and *srg*) to plot this curve, too.

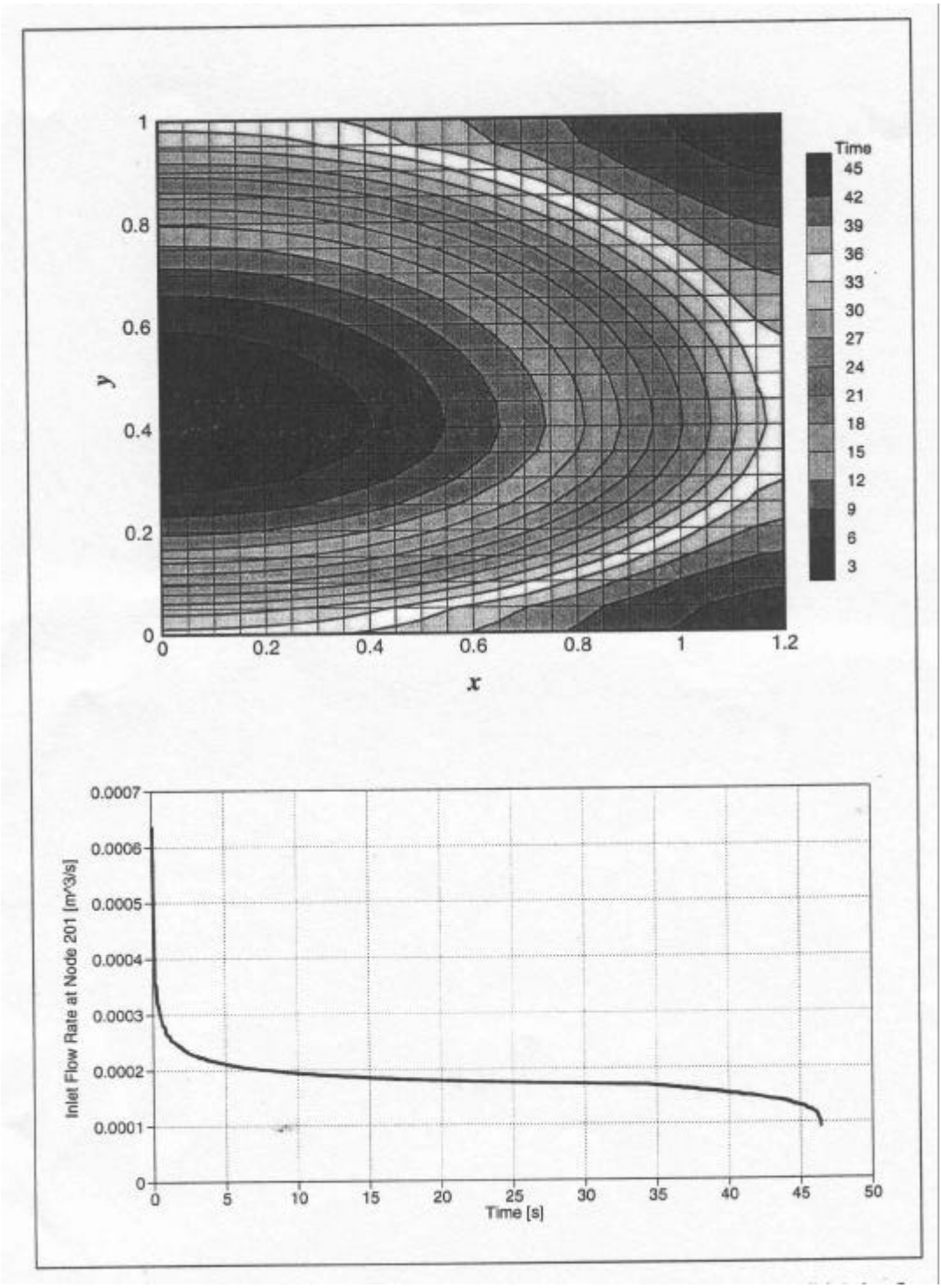

Figure 9.: Radial, constant pressure injection example - flow fronts and flow rate at inlet

Now we will look at a couple of variations of this basic case. At this point we should introduce another program, Lims GE. This program is a relatively simple OGL application that allows user to display and modify the mesh settings. The user manual should have been included in the handouts. Start this application by clicking on its icon,

go to the *file/open* menu and select *verif_2d_bare.dmp*. You should see the plate from Figure 8 displayed in the user area, as shown in Figure 10.

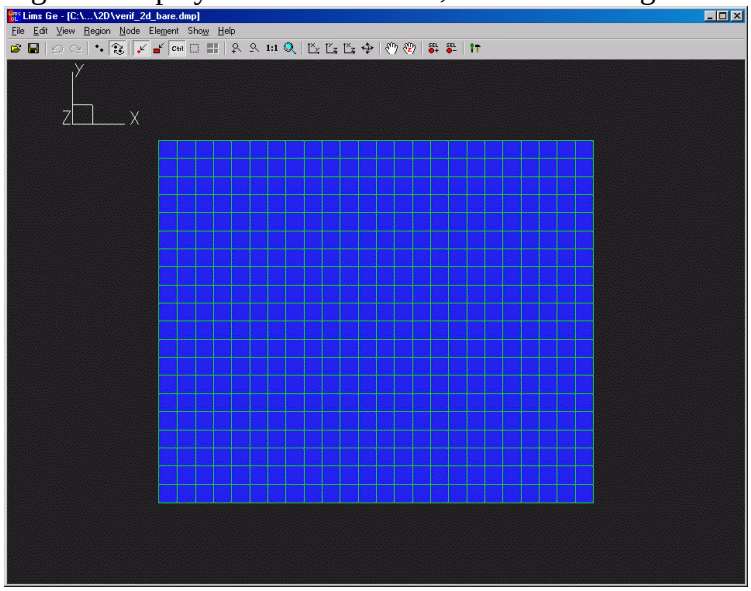

Figure 10.: 2D example loaded into LIMS GE

This is “bare” file, which does not have any gate or result information. Such files seem to be preferred by most users, since the gates can be added from LBASIC script after the file has been loaded. Thus the first task is to set gate. We may keep it at the old location – click on the corresponding node. It should appear as a magenta rectangle. Now that the node is selected, right-click it and, from the context menu, select *Set Gate*. From pull-down menu select type (pressure) and type in pressure value (100000). This point is captured in Figure 11.

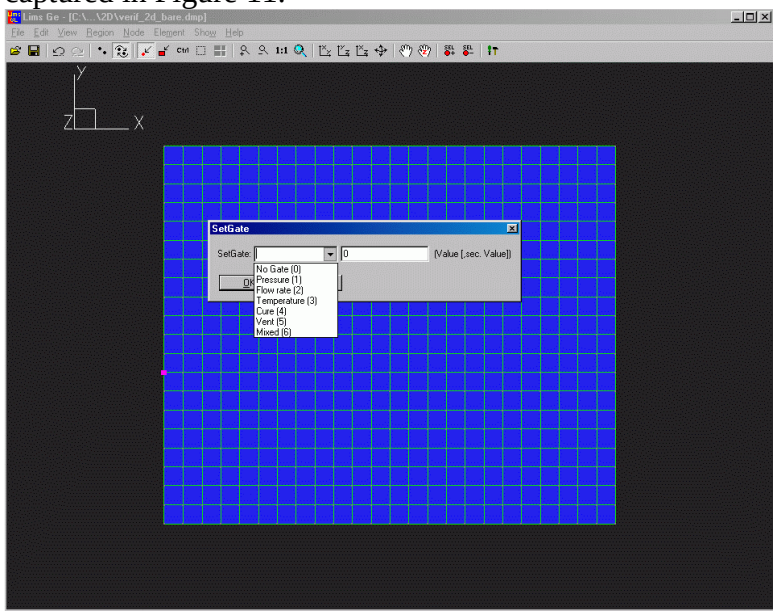

Figure 11.: Setting the gate

Now, let us assume that the situation in the mold is not perfect as usual, but that the preform is too small and leaves a channel on the top and right hand side for resin to

flow through – the dreaded racetracking phenomenon. How do we include this into our simulation? The easiest way is to artificially increase the permeability in the corresponding rows or columns of elements and that is what we are going to do. Note that there are other, more accurate ways to model this effect.

To set the permeability, we need to switch LIMS GE to element selection mode by clicking on appropriate icon. Then we need to shift- or ctrl-click all the elements we want to change. Note that when you shift-select two elements, LIMS_GE tries to include all elements between those two into selection. Once we are done with selecting, right-click can bring forth a context menu. Select *Set Permeability (global)* and type in appropriate permeability in the x-y coordinates. In our example, isotropic $1x10^{-6}$ (m^2)was selected, as shown in Figure 12.

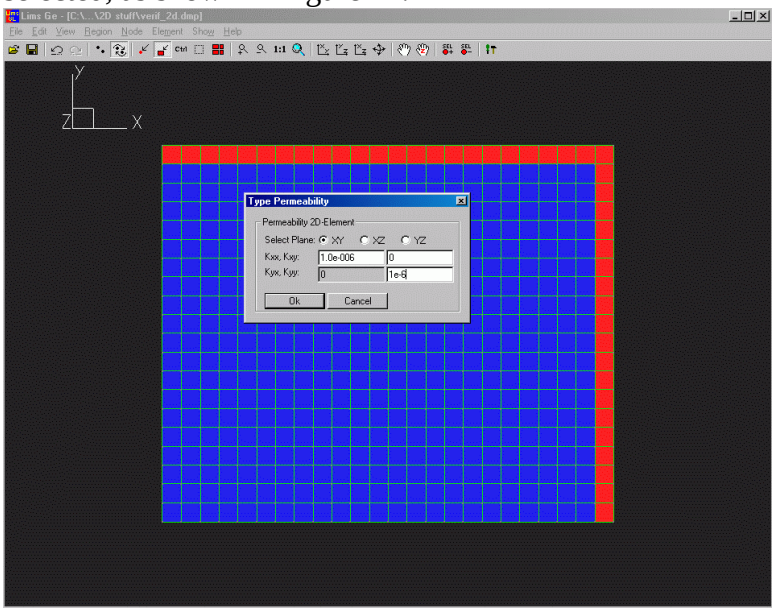

Figure 12.: Setting the racetracking permeability

Note that the actual permeability values are related to the size of the gap. This being done, our modified mesh is ready. Save it as *dmp* file *v2d_mod.dmp*, then once more as a LBASIC (*lb*) file *v2d_mod.lb*.

Why two files? LIMS GE writes modified geometry (including permeability) into *dmp* file. However, to set gates it uses procedure (named as the *.lb* file) in LBASIC script. This procedure also reads the proper *dmp* file and can be modified, for example to include call to *auto*.

Now we can start LIMS, load *auto* and *v2d_mod*, run *v2d_mod* and *auto* and then set the output type to TECPLOT and write the output file. Do not forget to reset time and forget *auto* if you did not restart LIMS.

```
##> read “v2d_mod.dmp”
##> load auto
##> load v2d_mod
##> v2d_mod
##> auto
##> setouttype “tplt”
##> write “v2d_mod”
```

The flow fronts are shown in Figure 13. In this figure, a few contour levels were added manually to make the figure more obvious.

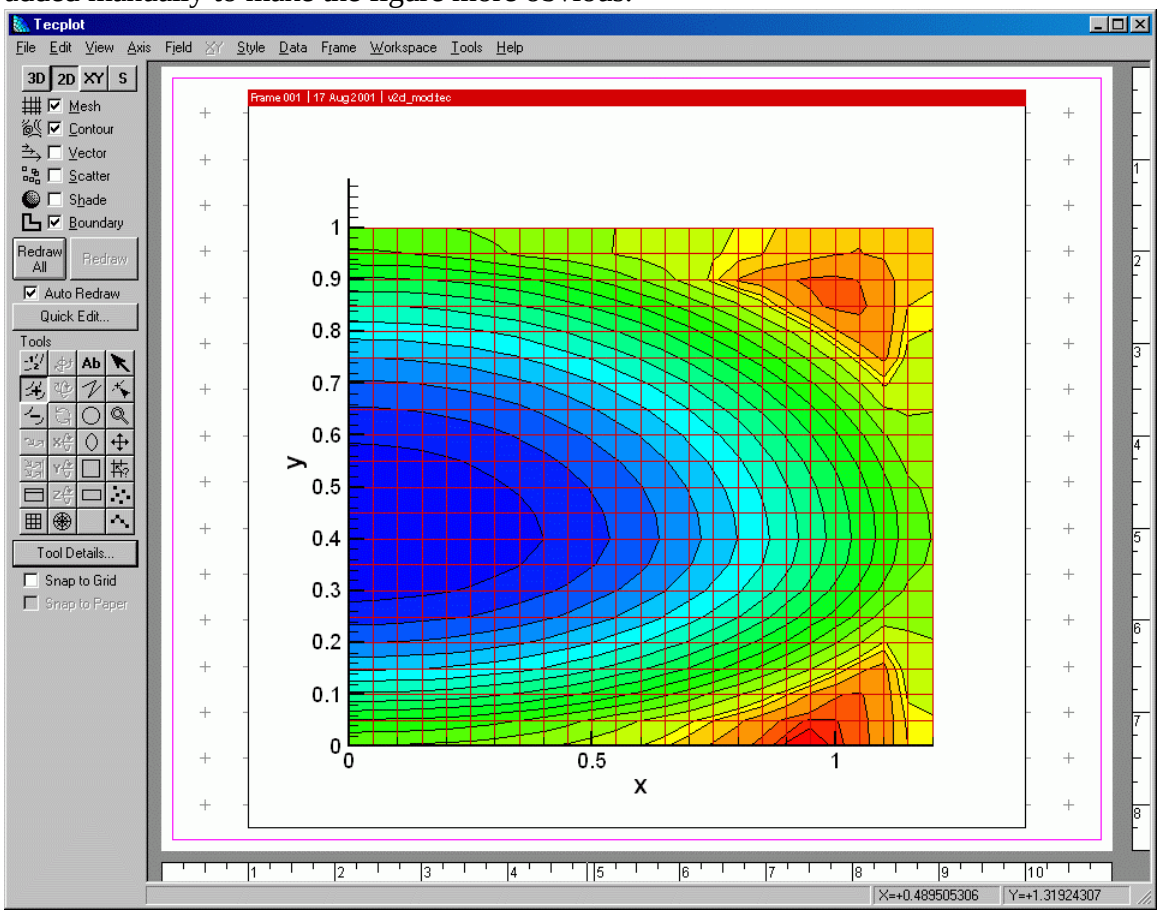

Figure 13.: Flowfronts for 2D racetracking case

Note that the last places to fill moved considerably from the original corners, thus creating potential voids.

Back to drawing board! Open LIMS GE and re-load *verif_2d_bare.dmp*. Lets try another variation on the old topics. First, lets change the point injection to line injection along all left edge. That should be relatively simple, apart from need to click all those points individually. Set the pressure gates to 100000 as shown in Figure 14.

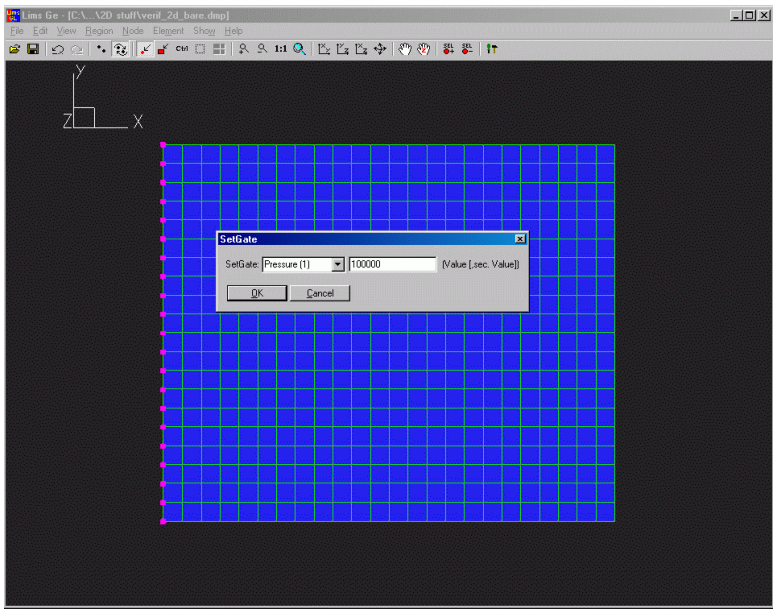

Figure 14.: Setting line injection gate along left edge

Now, if we try to fill this, we will see linear flow, because the x-component of permeability is the same throughout the part. Not too interesting. So we might need to upset the permeability values a bit.

We have previously assumed that we have gaps ,m where the permeability is very high. What about the opposite case? Assume you add a few preform layers or some insert over some area of your part. This should increase compaction and decrease permeability. Lets try so reduce permeability over rectangular zone as in Figure 15. This might simulate extra reinforcement of the area or, if the permeability value is very low, impermeable insert in the mold.

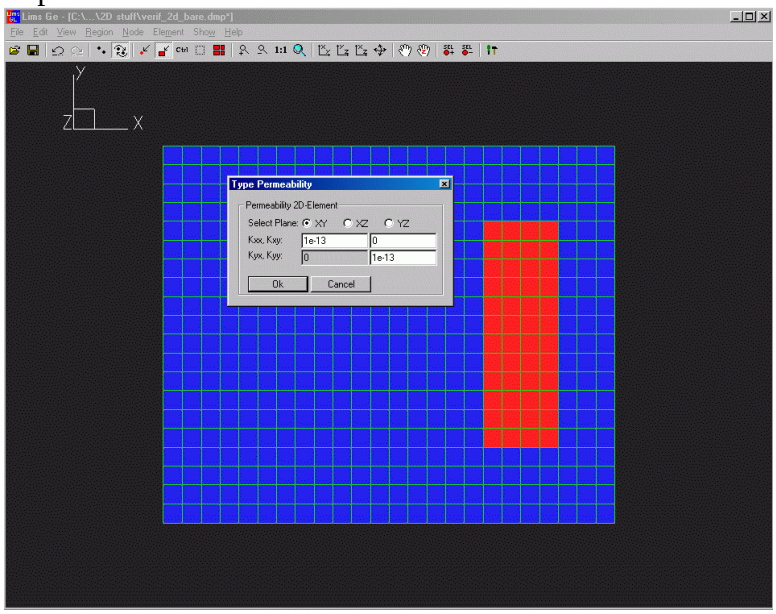

Figure 15.: Decreasing permeability over rectangular area

Set the permeability to $1x10^{-13}$ as shown. You might also want to set higher fiber volume fraction in this area by right-clicking, selecting *Set Volume Fraction* from context menu and setting the number (between 0 and 1, please). Then save file as *dmp* and *lb* again, load them into LIMS, run the proper procedure and export results to TECPLOT. Flowfronts should be similar to those in Figure 16. Note, that there were a few contour levels added in the figure to visualize the original linear flow before the flowfront reaches the insert.

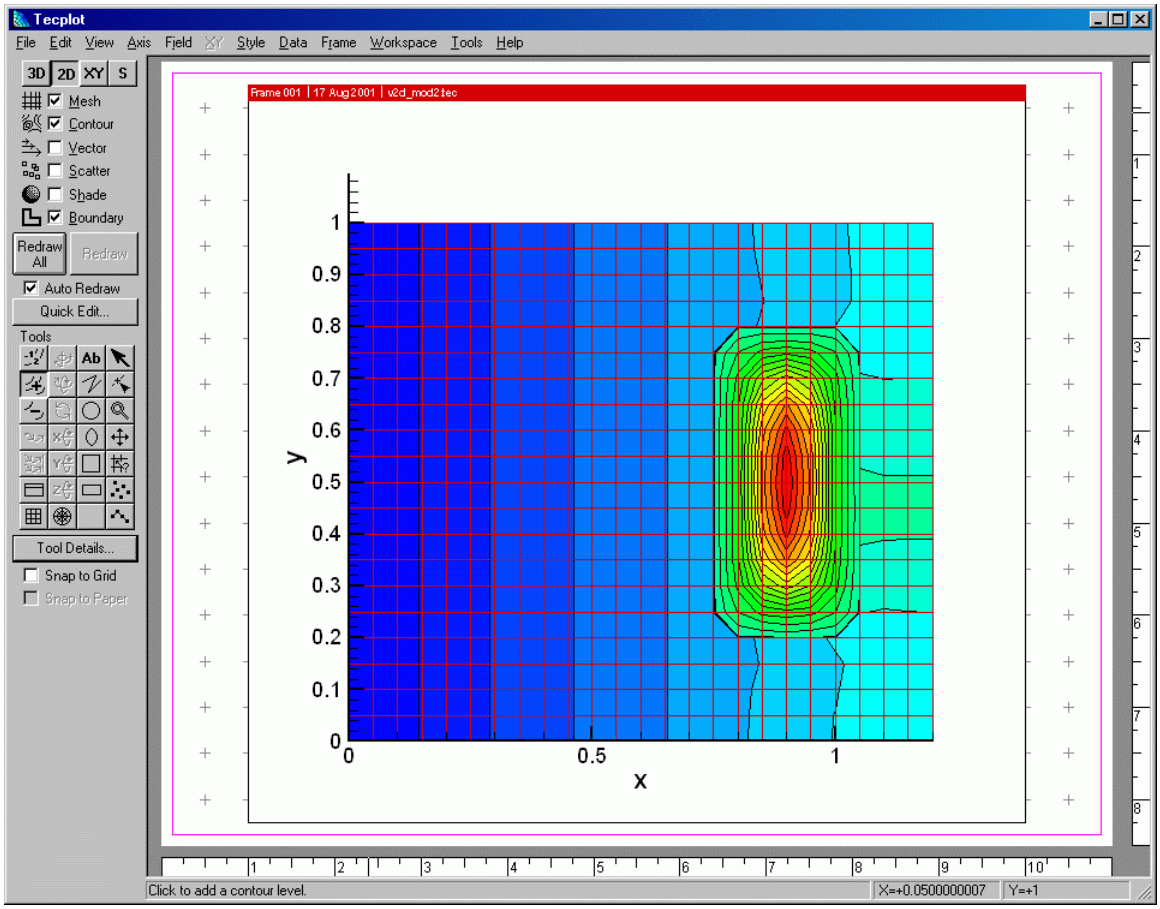

Figure 16.: Flowfronts for linear injection into a plate with rectangular insert of very low permeability

Part III

If you have got that far, you are partially on your own. You can ask questions or play with a few meshes provided in directory *examples\freestuff*. These meshes should be sufficiently small to guarantee short solution times.

File *bike_mesh.dmp*, the RTM formed bike frame from ETH Zurich, is seen in Figure 17. This is a complex shell like (2D) structure representing a real-world part. There is some suspicion that the file contains some forgotten disconnected elements.

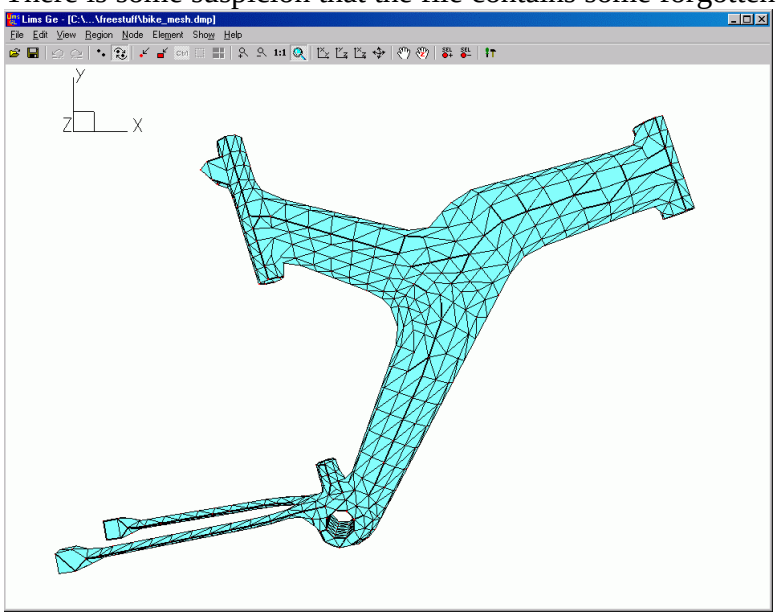

Figure 17.: Bike_mesh.dmp

If this seems too complex, *proto.dmp* is a mesh file of a plate with a set of stiffeners – see Figure 18.

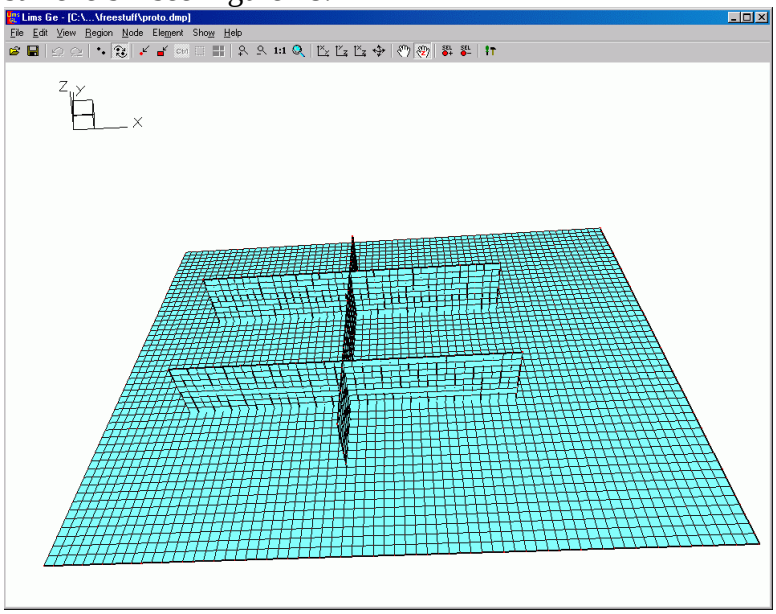

Figure 18.: Proto.dmp

The last file *small.dmp* is not as small as Figure 19 suggests, because it is a three-dimensional, 5-layer part with two top layers being thinner, representing distribution media in VARTM. Switch off the perspective projection and zoom a lot if you want too see the layered structure in LIMS GE (TECPLOT will magnify the z-dimension). Also, there are some 1D elements along y=0 on top surface, representing tubing. Given all these facts, it is not easy to work with this mesh in LIMS GE.

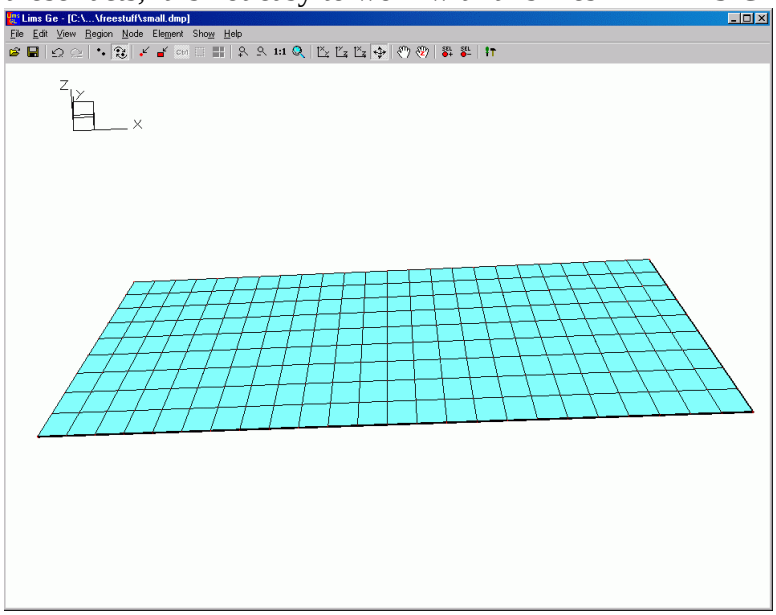

Figure 19.: Small.dmp