Simulation of Lava Flows using Cellular Automata
A Term Project at the University of Iceland in Spring 2006

Thomas Christian Chust

May 11, 2006

Abstract

Lava flows are a very common volcanic phenomenon. Lava is also a very complex fluid and influenced by a combination of effects from different areas of physics. The simulation of lava flows is thus an interesting research topic and it has a wide range of applications from more theoretical aspects to applied topics like risk analysis. Consequently about as many different approaches to the simulation of lava flows exist as behaviours of lava – but no definitive solution.

In this paper we want to analyze some simple ways of looking at a lava flow, what key aspects must be taken into account, how far the applicability of the models reaches and what the path to a really full featured model of lava flows could look like.

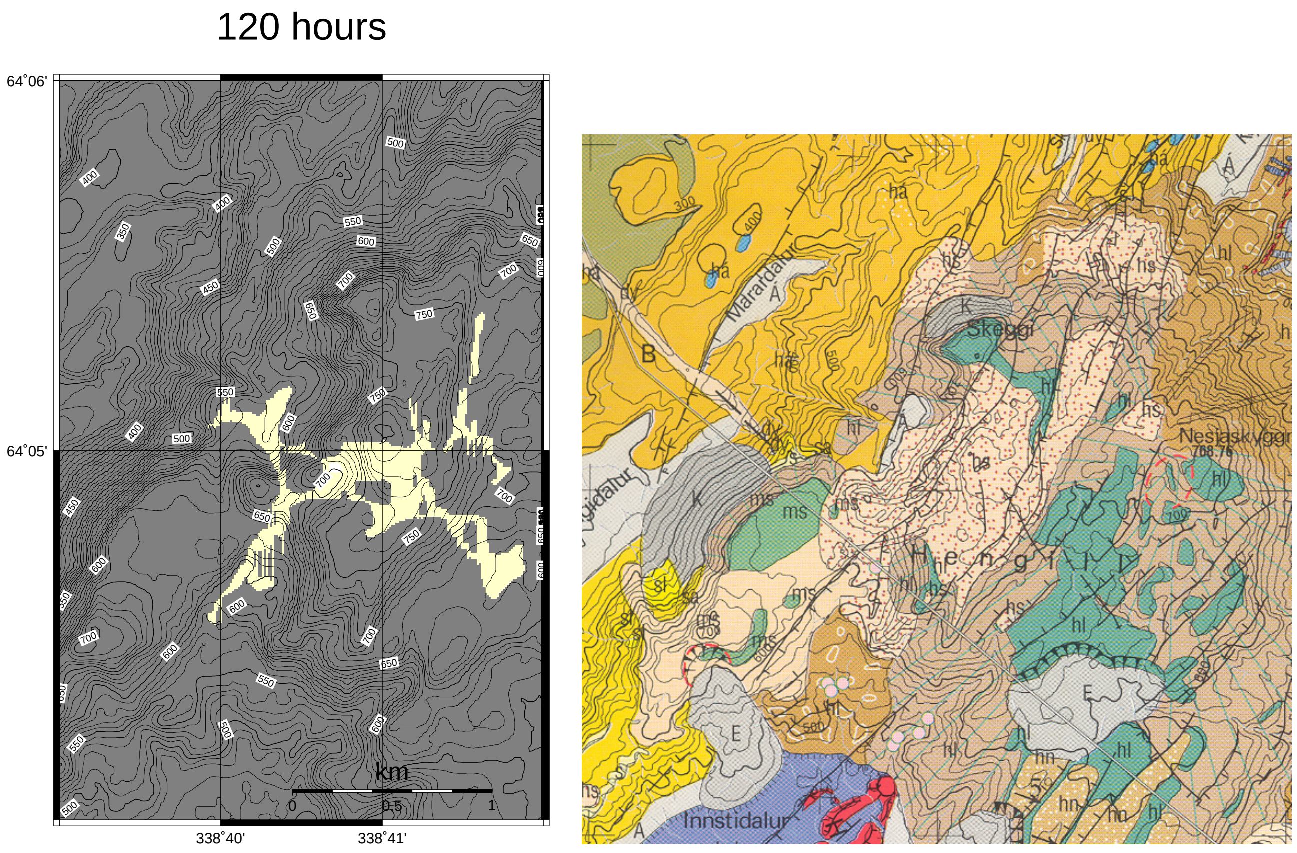

Figure 1: Distribution of a hypothetical Hengill lava flow next to a geological map [Sæ95] of the same area (see Table 1 for parameters)

Parameter	**Value**
Model	First model
Cell length	$10\ \mathrm{m}$
Timestep	$720\ \mathrm{s}$
Source position	$64°05\ '\mathrm{N}, 021°19\ '\mathrm{W}$
Influx	$5\ \mathrm{m/s}$ in four cells for $72\ \mathrm{h}$ simulated time
Influx temperature	$1373\ \mathrm{K}$
Solidus temperature	$1123\ \mathrm{K}$
Cooling coefficient	$6.15 \cdot 10^{-15}\ \mathrm{m/K^3}$
Adhesion model	$6.08 \cdot 10^{2}\ \mathrm{m} \cdot e^{-5.55 \cdot 10^{-3}\ \mathrm{K}T}$

Table 1: Simulation parameters for Figure 1

1 Introduction

Creating a simulation of lava flows requires many aspects to be considered: On the one hand there are the dynamics of flow itself, the topography modulating the driving forces of gravity and the intrinsic properties of lava, its rheological parameters. On the other hand there is the complex of thermodynamic effects, the lava starting to loose thermal energy through radiation, convecting air or water and conduction as soon as it leaves its vent. And the cooling lava changes with temperature, becoming more viscous and finally solidifying.

A model of a lava flow must at least represent dynamic mass transport, influences on it by the molten rocks' material properties, first and foremost viscosity, and the change of them with heat loss. The model will be governed by partial differential equations like the Navier-Stokes equation for the flow part.

Then there are several special features in lava flows that emerge from the interplay of the basic processes – formation of crust, flow tubes or levees to name important ones – which are not mere side effects but have decisive feedback effects on cooling and transport. They contribute to make lava a combination of phenomena of various nature.

In order to approximate the mathematical model numerically we can choose to directly attack the governing equations using finite difference or spectral solving strategies, or we can try to break down the problem into more manageable units before solving instead of using the global equations for the whole problem domain. This can be achieved with cellular automata or cellular nonlinear networks.

In addition we can choose to simplify parts of the model by not even attempting to obtain precise state information of a certain type at each point in the lava flow but rather working with bulk averages. In the case of lava flows the first big simplification one can make is the reduction to a two dimensional problem, as the primarily interesting feature is the geographical distribution of lava and its shape in the form of a thickness field. For this simplification we must average temperature and material properties over the whole thickness of the lava flow, though, and we must neglect certain properties of flow in three dimensions that are unlikely to affect lava. But it is necessary not to forget that we simplify a lot by looking at a 2D problem only.

1.1 Cellular automata

For this project a cellular automata approach was chosen. Cellular automata work by splitting a larger problem set into many small cells and assigning a set of state variables to each cell. Simulation proceeds in steps updating the state of every cell *at once*, calculating parameters for each cell locally, based only on the old state of the cell and its *neighbours*.

It is important to note the fact that state changes *at once* for all cells. This has the effect that the new state of a cell only ever depends on the old state of itself and its neighbours. This is the main difference to cellular nonlinear networks, for example, where this restriction is dropped, partially allowing the new state of a cell to depend on the new state of neighbours, which can facilitate implementation in software but complicate numerical properties.

The definition of how the cells look like, how they relate to the problem domain and what is a neighbour can be chosen arbitrarily, as long as those properties are well defined and consistent for the whole set of cells. For our lava flow simulations we will make the canonical choice of cells representing spacial units and neighbours in a geographical sense.

Cellular automata can be used to simulate a system which is dominated by local interactions as every cell is influenced directly only by adjacent cells. Not only does this apply to virtually any physical system viewed at a certain level of detail, but also it complies very well with the nature of transport and wave phenomena.

From the programmer's viewpoint a cellular automaton based simulation program has the main advantage that parallel execution of the algorithms can be achieved easily and on the same level of abstraction as the problem solution itself. With cellular automata we do not look at the physical system as a whole but break it down into manageable units, the cells. We store all state variables per cell, we apply the same laws of transformation from one time step to the next for every cell and the algorithm only needs data from the currently processed cell and its neighbours to do its job. Thus it is possible to distribute the set of cells into a large number of subsets each managed by a separate process for simulation. These processes only have to communicate information about cells that are neighbours to ones in another subset, which can conveniently be handled by shared memory or message passing architectures for parallel computing alike.

Concerning this direct approach to the problem and handling of the data in portions defined by the problem itself, cellular automata are quite the opposite of spectral partial differential equation solvers, for example, which lift the problem at hand from the time to the frequency domain and then apply methods of linear algebra, which constitute the only part in the algorithms that can effectively be parallelized, although that is still not trivial.

2 The first model

2.1 Idea

This model is an independent reimplementation of the concepts presented in [BC+93]. It was chosen mainly for the virtue of simplicity – both to ease the program development and to see how well a very simple model would describe the problem.

The model uses a two dimensional cellular automaton. The area of interest is gridded into square cells. For each cell, the four geographically adjacent cells directly to the north, east, south and west are considered its neighbours.

The interesting idea behind this model, though, is to greatly simplify the description of lava by several bulk averages and creatively chosen parameters. In particular, any type of explicit differential equation solving to describe the motion of the lava or the heat transport in it is avoided.

2.2 Algorithm

For each cell we store the following parameters:

- The height of cell floor above mean sea level,
- the thickness h of the lava layer,
- the mean temperature T of the lava layer,
- the thickness of lava flowing in from the source every step,
- the temperature of lava flowing in from the source every step and
- the thickness of lava flowing out of the cell in the four main geographical directions in the next step.

Of course lava heights always correspond directly to volumes, as we work with two dimensional cells of fixed base area.

Although we have to follow the restriction of updating the whole board at once, sensible use of one additional temporary variable per cell saves us from having to do double buffering with an additional copy of the complete cell array.

In detail the simulation algorithm works as follows for every timestep:

1. `tf_mix` in `ca-lava-transform.h` is called for every cell. The amount of lava flowing in from the source (if present) and from each of the neighbours is looked up and added together with the old height of lava in the cell.

   The temporary parameter of the cell is updated to hold an average of all the lava temperatures, weighed by the amounts of lava.

2. `tf_cool` is called for every cell. By averaging out the temperature in the first step, we assume that the system always has enough time to become thermally well mixed. It is then sensible to assume that the lava cools uniformly. In this model we only consider radiative cooling, which follows the law

   $$\varrho c V \frac{dT}{dt} = -\varepsilon \sigma A T^4,$$

   where ϱ is the density of the lava, c the specific heat capacity, V the volume of the cell, t the time, ε the surface emmissivity, σ the Boltzmann constant and A the area emitting radiation, the surface of the cell. This is a separable differential equation so we can solve

   $$\int_{t_0}^{t_1} -\frac{\varepsilon\sigma}{\varrho c} \underbrace{\frac{A}{V}}_{=\frac{1}{h}} \, dt = \int_{T_0}^{T_1} T^{-4} \, dT$$

   to obtain

   $$T_1 = T_0 \left(3 \frac{\varepsilon\sigma(t_1 - t_0)}{\varrho c} \frac{T_0^3}{h} + 1 \right)^{-\frac{1}{3}} . \tag{1}$$

   Radiative cooling is by the far the most effective process for heat loss of lava as long as the flow does not have a thick crust or is emplaced under exceptional conditions such as strong artificial or natural water cooling. Therefore if we reduce our cooling model to only one process, radiative cooling is the way to go. [KD96]

   Finally, if the cooling let the temperature drop below a certain threshold, we solidify the lava by adding its thickness to the geographical height and resetting the thickness and source influx variables of the cell to zero.

3. `tf_equalize` is called for every cell to determine the dynamics of the lava. This routine could be considered the core of the algorithm. Still thinking in bulk averages we note that if left to itself under the influence of gravitation, any liquid will with time level out to one flat equipotential surface with minimum potential energy. Our algorithm uses this principle to determine the fluxes of lava from one cell to another: An average height – geographical plus lava – of the current cell and its neighbours is calculated and every neighbour with more height than the average is discarded for the rest of this computation, then the averaging is repeated until no neighbours are left or all neighbours are lower than the average. Lava outflux to the remaining neighbours is now calculated proportional to their height below average level.

   To take viscosity into account, a minimum retained thickness of lava decreasing exponentially with temperature is calculated.

   Depending on the local topography this tends to produce a locally flat surface.

In addition to the actual calculation routines a fair amount of support code is needed to control loading and storing of data to and from the disk, managing the network interactions for distributed processing (namely the communication of state along boundaries of sub-grids), running integrity checks on the data if desired and printing status information.

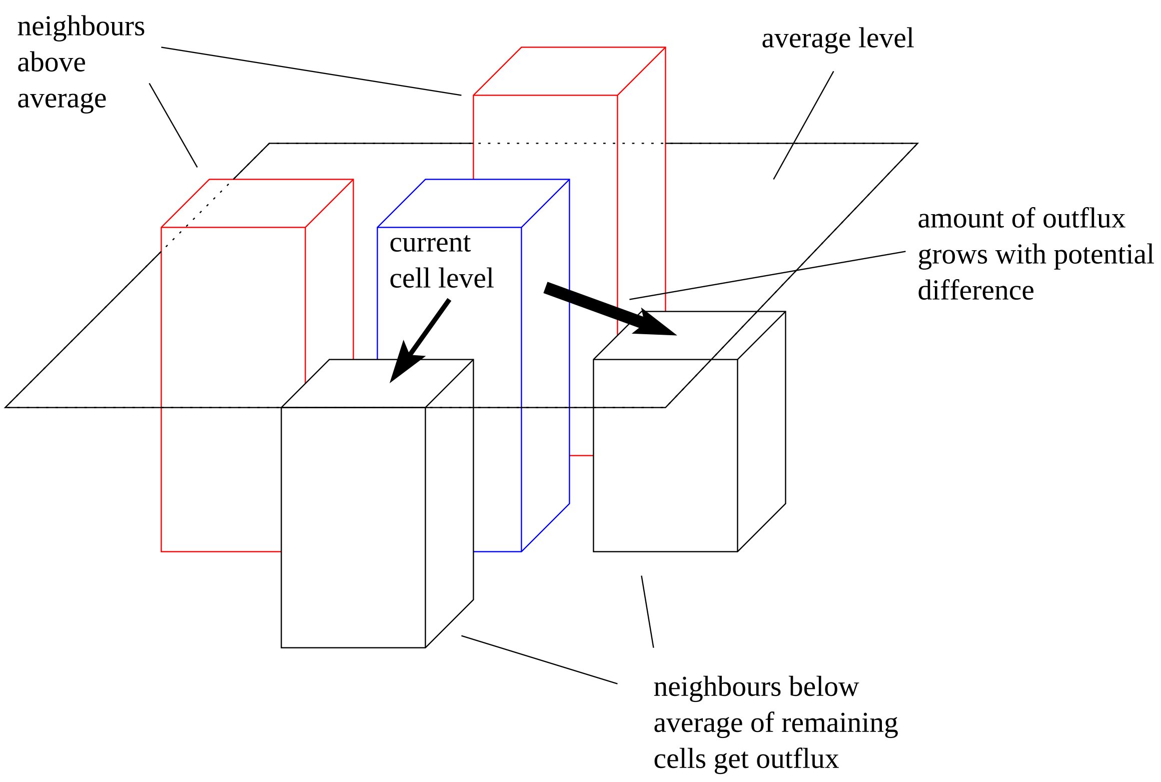
neighbours above average
average level
current cell level
amount of outflux grows with potential difference
neighbours below average of remaining cells get outflux

Figure 2: Principle of height balancing

2.3 Tuning the parameters

Of course a simulation program is only useful when fed some external parameters. We can see that in our case the following parameters must be specified:

- The timestep length,
- the side length of input cells,
- the influx amount and temperature from the lava source,
- the cooling parameter $3\frac{\varepsilon\sigma(t_1-t_0)}{\varrho c}$ from (1) and
- the two viscosity coefficients for the exponential dependence of retained lava on temperature.

Influxes and their temperature can be chosen relatively arbitrarily to match the eruption characteristics we want to simulate.

Once the time step has been fixed, the cooling parameter can be calculated from what is known about the thermodynamic properties of the lava.

The choice of the viscosity parameters poses a greater problem, because the model's way of describing viscosity is quite unusual. To figure out an estimate we consider a viscous fluid flowing down a slope under the influence of gravity. In case of steady state, the Navier stokes equation for this situation concerning the velocity parallel to the ground can then be written as

$$\mu\partial_z^2 v = \varrho g \cos\alpha,$$

where μ is the dynamic viscosity, z the perpendicular distance to the ground, v the aforementioned speed, ϱ the density, g gravity acceleration and α the slope of the hill. Integrating this in z we can claim that

$$z \approx \sqrt{\frac{v\nu}{g}},$$

where ν is the kinematic viscosity used for convenience. We can now plug in the free flow speed for v to get a rough estimate for the retained lava height.

The remaining problem is now to choose a cell length and timescale for the model. This is less trivial than one may think. On the one hand the time scale has to be suitably long to allow for mixing of the lava in every step, on the other hand it may at least not be longer than it takes the lava at free flow speed to flow through a complete cell length. But the cell length should not be chosen too large to get a good spacial resolution and numerical stability.

As the model does not simulate the mechanics of the lava but uses the height balancing algorithm described above instead, the progress of time only manifests itself explicitly in the model through the cooling parameter.

We see that the timescale should be chosen with care. In this model as well as in the original reference work a timestep and cell length that empirically fit well together are used.

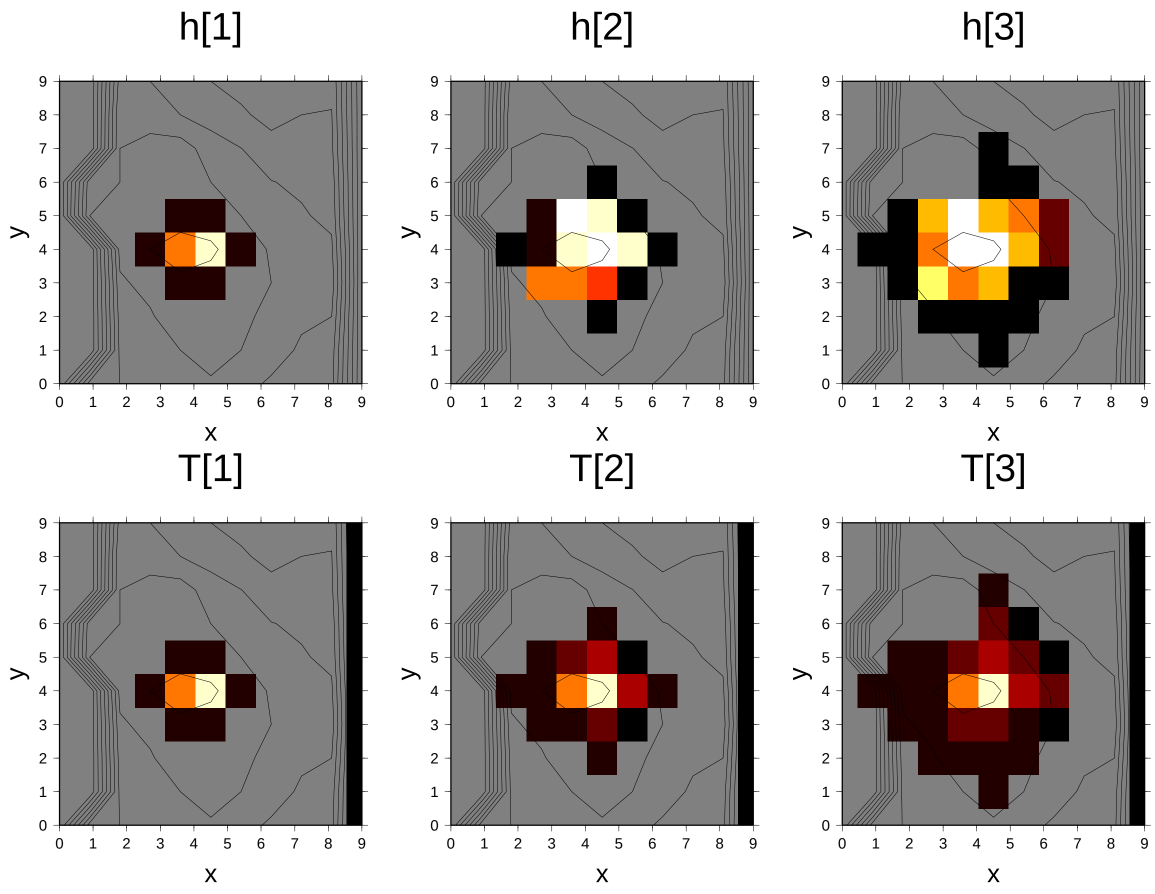

Figure 3: First model: Simulation with non-dimensional test values to illustrate the function, "height" h and "temperature" T have been plotted on schematic terrain

2.4 Observations

Several tests of the algorithm show that it gets the distribution of a lava flow right at a large scale, especially in the first phases of emplacement while the assumptions for cooling are valid best and the viscosity is not very high.

The nature of the computations can never deliver a precise flow simulation in detail, but it may be good enough for some sort of impact or affected area estimate.

Maybe the most problematic point about this approach is that the time and the velocity of the lava are not made explicit in the model, which in turn leads to an unusual viscosity model and general uncertainty about the best choice of parameters. When modelling an existing structure for reconstruction one can of course always fit the parameters by hand optimization or using an evolutionary algorithm in such a way as to produce an output in better accordance with the measurements. But the virtues of this process may be questionable because both the input parameters and outputs for temperature and thickness may become rather unrealistic while the simulation still yields a nice geographical fit.

It would give the model greater credibility if the result was less heavily affected by a heuristic choice of parameters.

3 The second model

3.1 Idea

Because of the observations of the first model's behaviour a modified algorithm was implemented that is based on the same two dimensional cellular automaton approach – and can thus reuse the first model's supporting code – but treats the motion of lava differently.

The primary goal was to base the simulation algorithm more closely on the physics of flow. For example it is desirable to have explicit velocity in the model to be able to ensure the fundamental principle of momentum conservation, which is completely neglected by the height balancing approach. Also rendering time and velocity explicit should render the model more robust concerning the choice of timescale.

Simulating the dynamics of flow explicitly allows us to model the viscosity a little more precisely, as well.

In addition, as crust formation is a very important process with a great impact on lava flow development, an automatic parameter modifying the radiative cooling depending on crust coverage estimated from flow speed has been added to the model.

3.2 Algorithm

In order to stay simple this second attempt uses a cell-wise forward difference solver for the Navier-Stokes equations at its core. The original Navier-Stokes equation is actually an acceptable approximation as the assumption of incompressibility is very reasonable for lava and Newtonian viscosity can at least describe the hot and fluid lava. The use of a Bingham model for fluid viscosity would be preferable, though.

In terms of lava cooling still only energy loss through radiation is simulated, but now we modulate it directly by crust coverage.

In this new model the cell state now consists of

- the height of cell floor above mean sea level,
- the thickness h of the lava layer,
- the mean temperature T of the lava layer,
- the vectorial velocity v of the flowing lava parallel to the ground,
- the thickness of lava flowing in from the source every step and
- the temperature of lava flowing in from the source every step.

The set of external parameters that have to be fed to the simulation grows by two coefficients for the crust coverage estimate.

Finally the sequence of operation in every simulation step looks as follows now:

1. `tf_mix` is called for each cell to calculate an average temperature and new height of lava after applying in- and outflows to and from the cell.

   This step is very similar to the first algorithm, but the outward fluxes are no longer an intrinsic variable of the model here. They are calculated further down in the sequence of operation for the next simulation step.

2. `tf_cool` is called for each cell to determine the radiative heat loss. The formula used in the first algorithm is modified to scale the emitted energy down by a factor between 0 and 1 that falls exponentially with the absolute value of flow speed following a consideration presented in [HR01].

   If the temperature of the cell has dropped below solidus, the complete cell is assumed to become solid like in the first algorithm. Note, though, that this new model does use a somewhat more realistic representation of temperature dependent viscosity, so the step to solidification is not that sharp.

3. `tf_flow` performs the last part of board state transformation for each cell and contains the core of the new algorithm. Basically, the right hand side of the Navier-Stokes equation

   $$D_t v = -\frac{1}{\varrho}\,\mathrm{grad}\,p + \nu\Delta v + f,$$

   where p is the pressure and f the driving acceleration due to gravity influence, is calculated from difference quotients involving the values of the neighbouring cells for first derivatives and of the current and neighbouring cells for the Laplace operator.

   The result is multiplied with the timestep of the simulation and added to the current velocity of the cell. Then new outfluxes from the current cell are calculated for the next step, based on the total lava content of the cell and the lava flow speed.

3.3 Observations

Although this model does produce flowing lava as well as the first one, the numeric stability of the differential equation solver is a serious concern. It is of course possible to reduce the grid cell length and time step to a comfortable level, but this comes at high computational cost. It seems advisable to improve the code by using a different strategy which includes intelligent dampening terms, because as one may guess from Figure 4 the model tends to produce exaggerated velocity values.

In addition some features that would be nice to have, such as simulation of a Bingham fluid instead of a Newtonian one, would be good extensions to the model – but of course this also comes at the cost of added complexity in the algorithm.

With increasing complexity of the partial differential equations it will become even more important to choose a solid solution strategy. But it is non-trivial to combine the best available spectral solution methods with a cell-based local view of the model world, simply because a stepwise solution in terms of simulated time is not straightforward compatible with a solution in the frequency domain. It may still be possible, though, to combine both approaches cleverly.

4 Conclusion

When following the path of cellular automata based lava simulation further, we would want to make several additional improvements to our model. Starting with aspects of applied mathematics we saw that we should attend to the numerical stability of the model. Helpful in this respect and well suited to reduce computational cost would be a way to dynamically adapt the time step to the problem. As this adaption

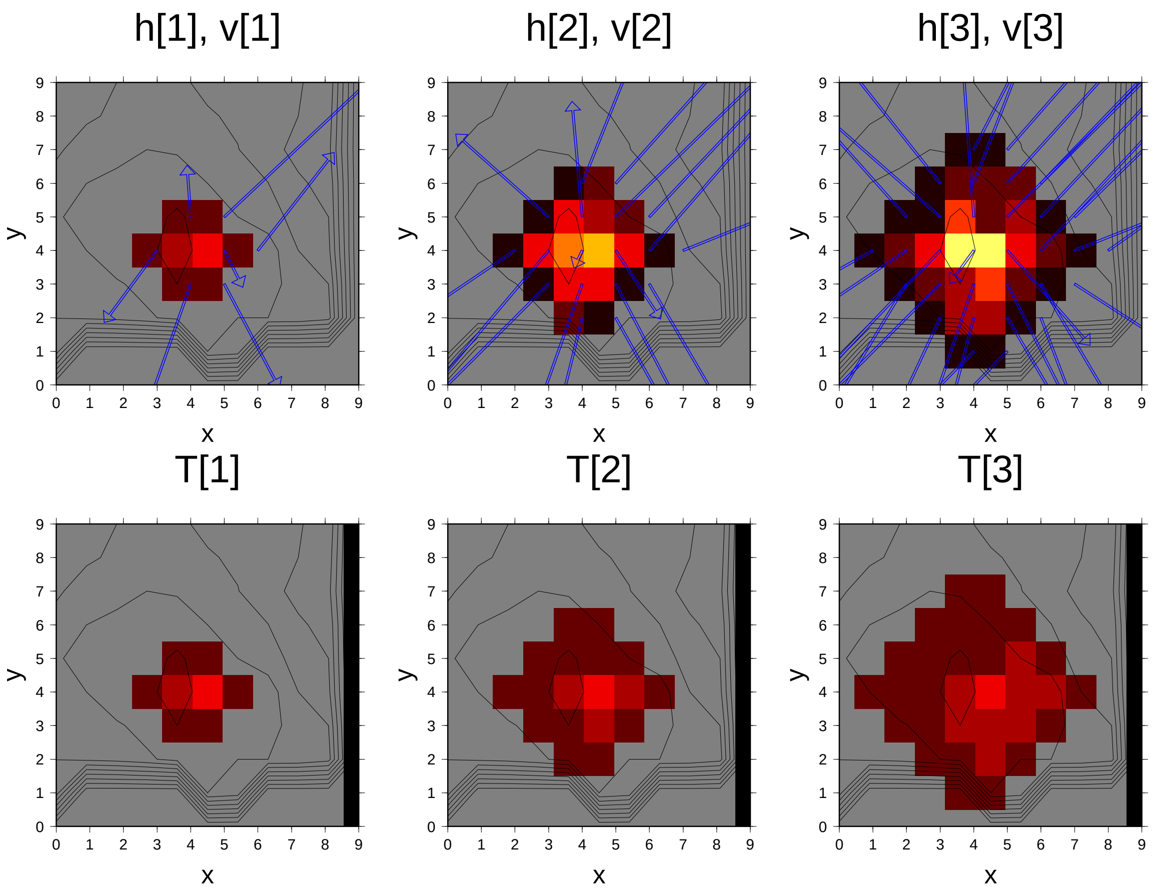

Figure 4: Second model: Simulation with non-dimensional test values to illustrate the function, "height" h, "velocity" v and "temperature" T have been plotted on schematic terrain

would have to occur cell-locally it would complicate control flow of the software a lot. But it should be possible to realize the necessary synchronizations within the presently applied worker-controller model of task separation. Especially with large grids and long running simulations the tradeoff of additional management overhead versus gained effective computation speed could be worth it.

Another fundamental improvement at the cost of code complexity would be to change from square cells to hexagonal cells. This has mainly the reason that a hexagonal cell grid offers a better angular resolution of flow directions and it does not force very ambiguous distribution of lava outflux towards a cell vertex. Together this would make the whole model more realistic.

With a side view towards the possible combination with a finite elements method for partial differential equation solving a hexagonal partitioning of the model space also promises better results than a square gridding.

But apart from these more technical issues we should also attend to physical problems with the model. We have seen that our models produce acceptable geographical distributions for lava flows but include many simplifications that do not only avoid difficulties for the calculations but also prevent the correct simulation of more complex detailed behaviour of the lava. Even when sticking close to the governing equations of fluid mechanics improvements would have to be made to the viscosity model and, maybe most important, the cooling model. In order to become capable of simulating things like crust formation correctly it is not sufficient to just derive the coverage from other model parameters, especially as the exact dependencies are not fully understood here. Such effects that involve partial solidification of the lava, require a much more fine grained cooling model that also takes conductive and convective effects into account plus it becomes more and more complicated to sustain the restriction to two dimensions when more complex lava phenomena like tubes should be simulated.

But we should note, that the requirements of which effects and state variables have to be modelled accurately and which are less important depend on the part of the lava flow we are looking at: Fast ‘a‘ā lava emplacement may be well modelled by the approaches described here, open channel flow may require a very similar approach but the rigid boundary conditions can make determination of flow direction even easier; flow underneath an insulating crust requires a different approach, most notably a different cooling model, while emplacement of a single pāhoehoe flow lobe could be treated similarly again, up to the point where crust tension becomes important; etc.

If we want to create a comprehensive model of lava flows it is probably a good idea to combine several modules that concern themselves with the different lava transport processes and are optimized for their specific purpose to avoid an explosion of computational cost due to an over-general model that tries to include all possible effects.

5 Acknowledgements

The author gratefully acknowledges the impulses, valuable ideas and support by Ármann Höskuldsson and Þorvaldur Þórðarson from the University of Iceland.

A Where are the simulation programs?

You should have obtained a copy of the source code for the simulation programs together with this report. If not, you can find links to packages containing the necessary files at `http://www.chust.org/projects/lava-flows/index.html`.

The packages contain documentation generated from the sources and the source is commented. Questions on algorithmic details that are not covered in this report can hopefully be answered quickly by looking at the code and comments.

B Compiling and running the models

To encourage everbody to compile highly optimized versions of the simulation code with tweaked compiler flags, and, even more important, to encourage people to actually look at the source code, no binary package for the programs is provided. It was however a fundamental design goal for all the code to be fairly portable.

B.1 Prerequisites

Before you can compile and run the models yourself, you will need working installations of the Parallel Virtual Machine for parallel computing and build support [pvm], the NetCDF library and tools for portable binary data storage [net] and an ANSI C compiler.

B.2 Starting the build process

The code has been tested on a PowerPC machine running MacOS X and using the GNU C compiler as well as on an Intel based Linux cluster using the GNU and Intel C compilers.

If you are building on one of these architectures, you may want to copy or link the `custom-DARWIN.aimk` or `custom-LINUX.aimk` file in the source code directory to `custom.aimk` after unpacking the source package. These files contain some tweaked compiler settings. The Linux version also causes the Intel C compiler to be used, which yielded slightly better optimizations for my taste.

You can start the build with a command like `aimk RELEASE=1 all` in the source directory to create all tools and simulation programs with debugging features turned off. Take a look at `Makefile.aimk` and the source documentation to see which other top level targets may make sense for you.

B.3 Creating simulation grids

Templates for empty simulation grids are contained in the source packages in the file `sim-grid.cdl`. Run this file through the `ncgen` program to produce a usable binary simulation grid. You should also fill the variable data sections for geographical height and source influx with reasonable values. Furthermore you must define the global parameters and length of time steps.

The gridded values are assumed to be valid for the centers of the grid cells. Although you can specify any rectangular grid cell geometry, the simulation programs work under the assumption that the grid is equally spaced and has square cells.

Some utility programs for filling the simulation grid from a geographical grid in similar format and for setting single or multiple values in the grid are available. You should refer to the source documentation for more information here.

It is worth noting that the simulation grids are compatible with the Generic Mapping Tools [gmt]. If you use this suite of tools for graphical data visualization, you can directly operate on the simulation grids to extract visual representations of the result data.

B.4 Starting a simulation

A simulation can be started very simply by passing the simulation grid file and the information how many steps should be run in how many parallelly working task to the simulation controller program. Of course PVM has to be running for this to work. For example to run the simulation on 15 machines, splitting the grid equally into 3 columns and 5 rows, for 100 steps and dumping data back to the grid every 20 steps you would enter something like `$PVM_ARCH/ca-lava my-simulation-grid.nc 3 5 100 20`. To run the same simulation on a single machine without the help of PVM you would enter `$PVM_ARCH/ca-lava-serial my-simulation-grid.nc 100 20`.

The serial version of the simulation program is functionally equivalent to the parallel one. It is mainly useful if you want to run it inside a debugger or on a single machine, single processor system.

References

[AC$^+$05] M. V. Avolio, G. M. Crisci, et al. SCIARA γ2: An improved cellular automata model for lava flows and applications to the 2002 etnean crisis. *Computers & Geosciences*, 2005.

[BC$^+$93] D. Barca, G. M. Crisci, et al. Cellular automata methods for modelling lava flows: simulation of the 1986-1987 eruption, Mount Etna, Sicily. In C. R. J. Kilburn, editor, *Active Lavas: Monitoring and Modelling*, pages 291–310. UCL Press, Limited, 1993.

[Col62] G. H. A. Cole. *Fluid Dynamics*. Methuen & Co. Ltd., 1962.

[DNFV05] C. Del Negro, L. Fortuna, and A. Vicari. Modelling lava flows by cellular nonlinear networks (CNN): preliminary results. *Nonlinear Processes in Geophysics*, 12:505–513, 2005.

[gmt] The generic mapping tools. The GMT Home Page. http://gmt.soest.hawaii.edu/.

[HR01] A. J. L. Harris and S. K. Rowland. FLOWGO: a kinematic thermo-rheological model for lava flowing in a channel. *Bulletin of Volcanology*, 63:20–44, 2001.

[KD96] L. Keszthelyi and R. Denlinger. The initial cooling of pahoehoe flow lobes. *Bulletin of Volcanology*, 58:5–18, 1996.

[Kil00] C. R. J. Kilburn. Lava flows and flow fields. In H. Sigurdsson, B. F. Houghton, et al., editors, *Encyclopedia of Volcanoes*, pages 291–305. Academic Press, 2000.

[net] Unidata netcdf. The NetCDF Home Page. http://www.unidata.ucar.edu/software/netcdf/.

[pvm] Pvm: Parallel virtual machine. The PVM Home Page. http://www.csm.ornl.gov/pvm/.

[Sæ95] K. Sæmundsson. *Hengill, geological map (bedrock) 1:50000*. Orkustofnun, Hitaveita Reykjavíkur and Landmælingar Íslands, 1995.

[Wei] E. W. Weisstein. Navier stokes equations. A Wolfram Web Resource. http://scienceworld.wolfram.com/physics/Navier-StokesEquations.html.