Direct Modelling and Programming of ODE – and CA – Results for ARGESIM Benchmark C17 'SIR-Type Epidemic' in Modelica/Dymola

Stefan Emrich, Ch. Simon, Th. Kadiofsky, J. Vilsecker, Vienna Univ. of Technology, Austria
Stefan.emrich@*tuwien.ac.at*

Simulator: Modelica is a free object-oriented modeling AND programming language designed for modeling large and complex systems in control, mechanics, electronics, etc and supports a-causal physical modeling. But Modelica allows also 'classic' programming with scalers, vectors and matrices. And in addition to direct text mode programming, graphical modeling is accessible through several front ends.

Simulations have been performed in Dymola, a 'full' Modelica simulator (in principle, Dymola has initiated the development of Modelica as 'standard' for physical modeling.

Model: Using the Modelica text mode, the given system of ODEs ([1]) can be solved numerically without difficulties by defining the model with the following equations:

```
equation
  der(S)= -r*S*I;
  der(I)= r*S*I-a*I;
  der(R) = a*I;
end SIR_epicemic;
```

Also the LGCA can be implemented with the Modelica language: The data structure used to represent the cells is a three dimensional array of integers. The first two dimensions contain the cells of the lattice and the third one represents all possible directions of particles within the cell, which count either 4, in the case of a HPP model, or 6 in the case of a FHP model. Each entry of this matrix has one of the following values: 0 empty, 1 susceptible, 2 infected, 3 recovered. Every time step consists of two phases: intercellular and inner-cellular motion. The former is realized by exchanging corresponding blocks of the matrix being aware of the given cell structure and the periodic boundary conditions, as shown in the following extract:

```
dim1:=size(World,1); dim2:=size(World,2);

// left-right-motion
World_new[:,1:dim2-1,2]:= World[:,2:dim2,5];
World_new[:,2:dim2,5]  := World[:,1:dim2-1,2];
World_new[:,dim2,2]    := World[:,1,5];
World_new[:,1,5]       := World[:,dim2,2];

// diagonal motion
for i in 1:dim1-1 loop
  // odd lines
  if mod(i,2) == 1 then
    World_new[i+1,:,6] := World[i,:,3];
    World_new[i,:,3]   := World[i+1,:,6];
    World_new[i+1,1:dim2-1,1]
                       := World[i,2:dim2,4];
    World_new[i+1,dim2,1]:= World[i,1,4];
    World_new[i,2:dim2,4]:=
                  World[i+1,1:dim2-1,1];
    World_new[i,1,4]     := World[i+1,dim2,1];
    // even lines
  else ..........
    ...
  end if; end for;
```

The inner-cellular motion is realised cell-wise. Concerning the FHP model there can be either a 3-particle head-on collision or a 2-particle head-on collision.

```
temp := World[i,j,:];
// 2-particle head-on collision
if ((temp[1]<>0) and (temp[4]<>0)
    and (temp[2]==0) and (temp[3]==0)
    and (temp[5]==0) and (temp[6]==0)) then ...
end if;
```

Recovery of infected individuals and infection of susceptible individuals happen according to specific rates. But a susceptible person only becomes infected with a certain probability, if there is at least one infected person in the same cell.

```
if I and S then
  k := 1;
  while PosS[k] <> 0 loop
    z := FHP.rand();
    if z < Inf_Rate*100 then
      World[i,j,PosS[k]]:=2;
    end if;
    k := k+1;
  end while;
end if;
```

In the last step of the inner-cellular activities, the particles move on through the cell along their directions, which means that the incoming particles become outgoing ones.

```
temp2 := World[:,:,1];
World_new[:,:,1] := World[:,:,4];
World_new[:,:,4] := temp2;
...
```

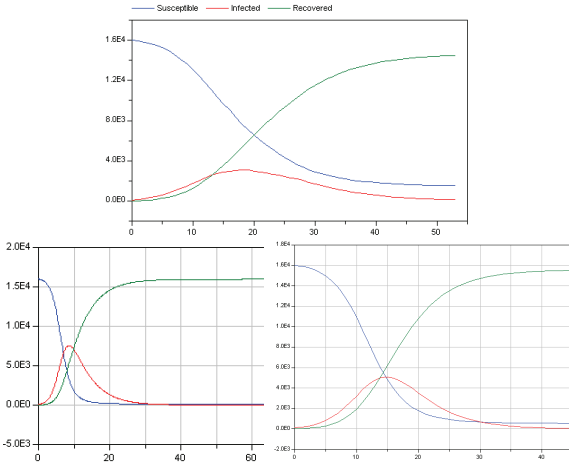

Figure 1. top: ODE, left: HPP, right: FHP solution

A-Task: Comparing the three plots in Figure 1 shows that the solutions of the HPP-model and the FHP-model deliver the same long-term-behavior. The main difference of the two models is the maximum number of infected individuals. The fact that this number is much higher in the FHP-model can be explained as follows: In the FHP-model there are more particles in one cell and hence more individuals can possibly be infected at once. The solution of the ODEs differs more significantly. Especially the velocity of propagation is much higher in the continuous model.

B-Task: To simulate the given vaccination strategies ([1]) in the FHP model, the initial distribution of the population was modified. Infected individuals were grouped together in one half of the domain and 4000 of the susceptible individuals were assumed to be vaccinated (recovered).

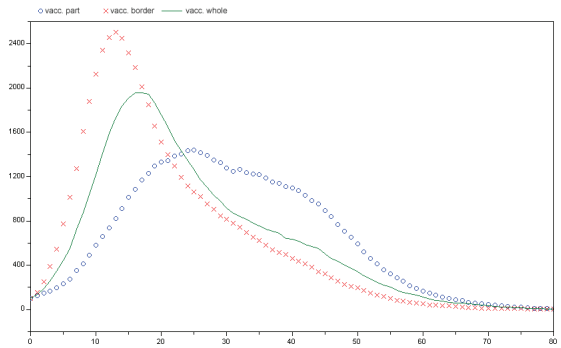

Figure 2: infected individuals – differtent strategies red: border vac., green: global vac., blue: partial vac.

Comparing the different strategies by opposing the number of infected individuals shows that vaccination in the 'epidemic' half of the domain yields a lower maximum peak of the number of infected people but at the same time the epidemic lasts longer. Border vaccination leads to a rapid breakout of the disease but also to a shorter lifetime. This fact can be explained by the initially unrestricted spread of the epidemic. The spatial isolation yields a fast decay of the disease.

C-Task: In contrast to the first tasks the whole population is redistributed uniformly after every time step. For this reason the infection process was modified here: In cells, which contain infected individuals, a certain fraction of the susceptible individuals becomes infected.

Figure 3 shows a comparison of the number of infected individuals in the ODE- (blue), DE- (red) and FHP-approach (green). The behaviour is not only qualitatively but also quantitatively similar. In contrast to task a, the velocity of propagation is nearly identical in all three approaches.

Resumé: The perhaps astonishing aspect of this solution is the fact, that Modelica can be used as technical programming language like MATLAB. The CA- models and update programs were directly translated from a MATLAB model / MATLAB algorithm. There, structures for vectors, matrices and their manipulation are very similar.

The Modelica-programmed algorithm is executed in a Dymola - ALGORITHM section. But some tricky modifications in the code were necessary to get correct results, because of ambiguities in interpretation of discrete equations.

Corresponding author: Stefan Emrich,
Vienna Univ. of Technology
Inst. for Real Estate Development and
Inst. f. Analysis and Scientific Computing /
Wiedner Hauptstrasse 8-10. 1040, Vienna
stefan.emrich@tuwien.ac.at

Received: January 17, 2008
Revised: June 20, 2008
Revised: November 5, 2008
Accepted: November 10, 2008