Initialization within OpenModelica

Lennart A. Ochel, M.Sc.
Bielefeld University of Applied Sciences

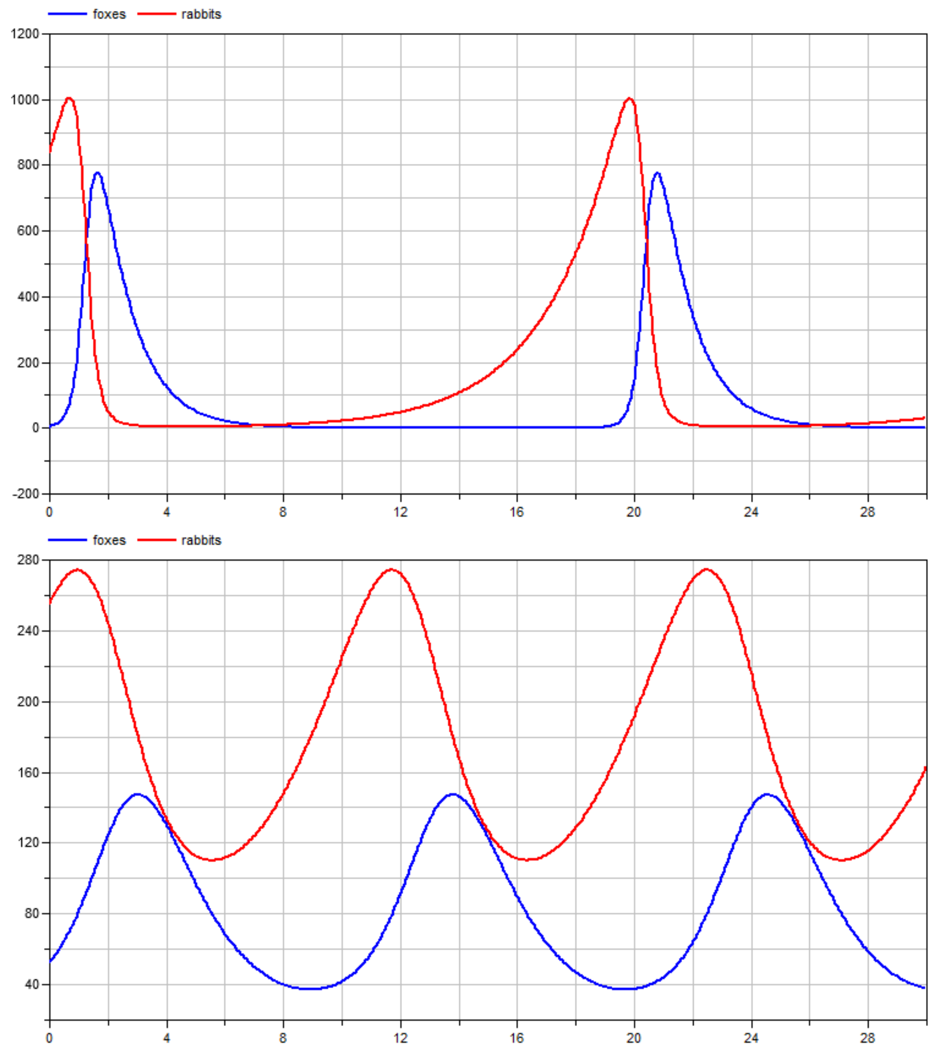
Motivation for Initialization

ordinary differential equations

- **initial value problem**

high-level description of **parameters**

high-level descripton of **discrete** variables



Outline

Modelica and Initialization

Mathematical Representation

Numeric Method

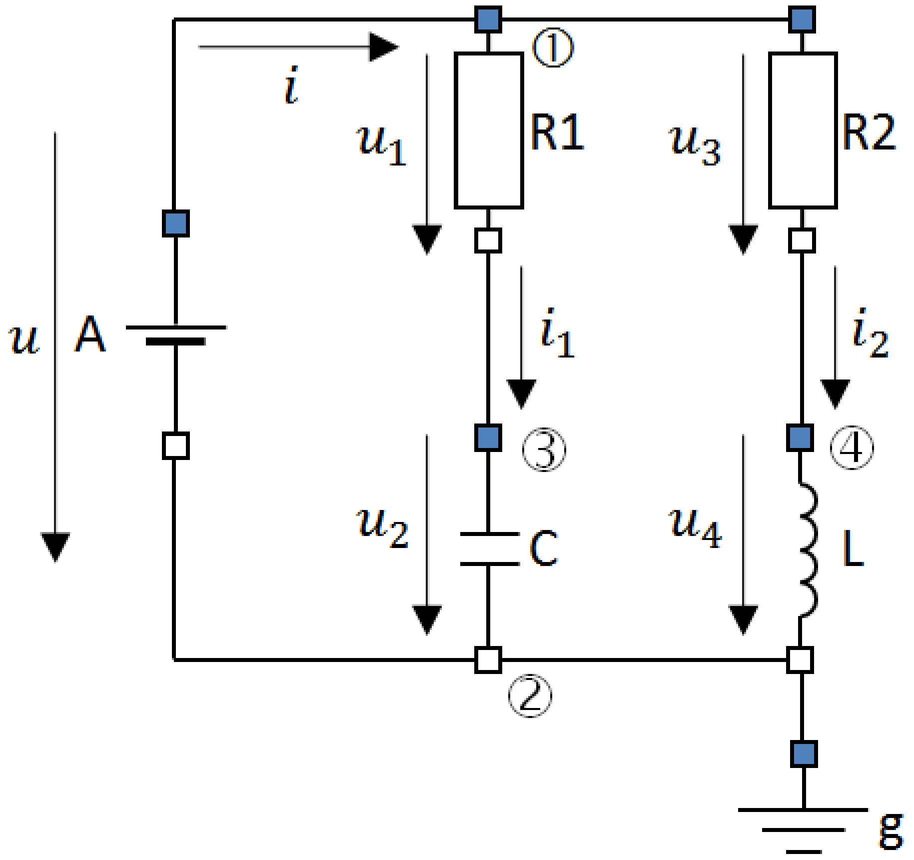
Symbolic Method

Conclusion and Outlook

Modelica and Initialization

Example

```
model InitSampleModel
  Resistor R1(R=270);
  Resistor R2(R=470);
  Capacitor C(C=1e-6);
  Inductor L(L=1e-3);
  Vsource A(U(fixed=false, start=1));
  Ground g;
initial equation
  der(L.i1) = 0;
  der(C.v) = 0;
  R1.v = 2;
equation
  connect(A.p, R1.p);
  connect(A.p, R2.p);
  connect(A.n, C.n);
  connect(A.n, L.n);
  connect(A.n, g.p);
  connect(R1.n, C.p);
  connect(R2.p, L.p);
end InitSampleModel;
```



Modelica and Initialization

Example

```
model InitSampleModel
  Resistor R1(R=270);
  Resistor R2(R=470);
  Capacitor C(C=1e-6);
  Inductor L(L=1e-3);
  Vsource A(U(fixed=false, start=1));
  Ground g;
initial equation
  der(L.i1) = 0;
  der(C.v) = 0;
  R1.v = 2;
equation
  connect(A.p, R1.p);
  connect(A.p, R2.p);
  connect(A.n, C.n);
  connect(A.n, L.n);
  connect(A.n, g.p);
  connect(R1.n, C.p);
  connect(R2.p, L.p);
end InitSampleModel;
```

Linguistic Devices

- `initial equation`
- `initial algorithm`
- `initial()`
- `homotopy(…)`
- variable-attributes
  - `fixed`
  - `start`
  - `nominal`
  - `min`/`max`

Mathematical Representation

Mathematical Representation

Variables

name	description
$x(t)$	vector of all states
$\dot{x}(t)$	vector of all derived states
$y(t)$	vector of all algebraic variables
$\mathrm{d}(t)$	vector of all discrete variables
p	vector of all parameters
t	simulation time
t_0	initialization time

Equation Systems

$$0 \overset{!}{=} F(x(t), \dot{x}(t), y(t), d(t), d^{pre}(t_e), p, t)$$

$$z(t) = \tilde{f}(x(t), d(t), d^{pre}(t_e), p, t)$$
$$\Leftrightarrow z(t) = \hat{f}\left(\omega, d(t), p^{fixed}, t\right)$$

$$0 \overset{!}{=} res \coloneqq H(x(t_0), \dot{x}(t_0), y(t_0), d(t_0), d^{pre}(t_0), p, t_0)$$
$$\Leftrightarrow 0 \overset{!}{=} res \coloneqq \widehat{H}\left(\omega, \mathrm{z}(t_0), d(t_0), p^{fixed}, t_0\right)$$

$$p \coloneqq (p^{fixed} \quad p^{free})^\top$$
$$\omega \coloneqq (x(t_0) \quad p^{free} \quad d^{pre}(t_0))^\top$$
$$z(t) \coloneqq (\dot{x}(t) \quad y(t))^\top$$

Mathematical Representation

Non-linear System of Equations

$$0 \overset{!}{=} \begin{pmatrix} F(x(t_0), \dot{x}(t_0), y(t_0), d(t_0), d^{pre}(t_0), p, t_0) \\ H(x(t_0), \dot{x}(t_0), y(t_0), d(t_0), d^{pre}(t_0), p, t_0) \end{pmatrix}$$

Non-linear Optimization

$$\min_{\omega, z(t_0)} f\left(\omega, z(t_0), d(t_0), d^{pre}(t_0), p^{fixed}, t_0\right)$$

s.t.

$$g\left(\omega, z(t_0), d(t_0), d^{pre}(t_0), p^{fixed}, t_0\right) = 0$$
$$\omega^{min} \leq \omega \leq \omega^{max}$$
$$z^{min} \leq z \leq z^{max}$$

$$p \coloneqq (p^{fixed} \quad p^{free})^{\top}$$
$$\omega \coloneqq (x(t_0) \quad p^{free} \quad d^{pre}(t_0))^{\top}$$
$$z(t) \coloneqq (\dot{x}(t) \quad y(t))^{\top}$$

Mathematical Representation

Non-linear System of Equations

$$0 \overset{!}{=} \hat{H}\left(\omega, \mathrm{z}(t_0), d(t_0), p^{fixed}, t_0\right)$$

with

$$z(t_0) \coloneqq \hat{f}\left(\omega, d(t_0), p^{fixed}, t_0\right)$$

Non-linear Optimization

$$\min_{\omega}\left\{\sum_i \hat{H}_i\left(\omega, \mathrm{z}(t_0), d(t_0), p^{fixed}, t_0\right)^2\right\} \rightarrow 0$$

s.t.

$$z(t_0) = \hat{f}(\omega, d(t_0), p^{fixed}, t_0)$$
$$\omega^{min} \leq \omega \leq \omega^{max}$$

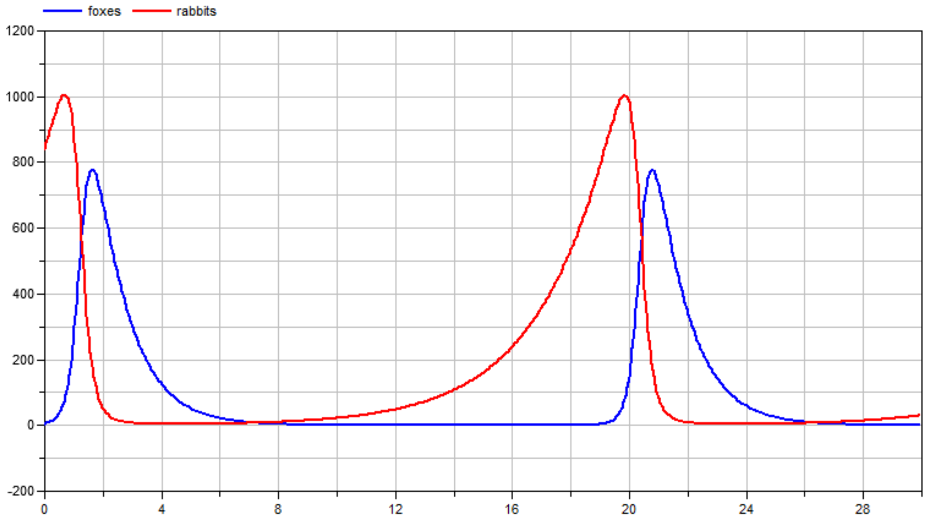
$$p \coloneqq (p^{fixed} \quad p^{free})^\top$$
$$\omega \coloneqq (x(t_0) \quad p^{free} \quad d^{pre}(t_0))^\top$$
$$z(t) \coloneqq (\dot{x}(t) \quad y(t))^\top$$

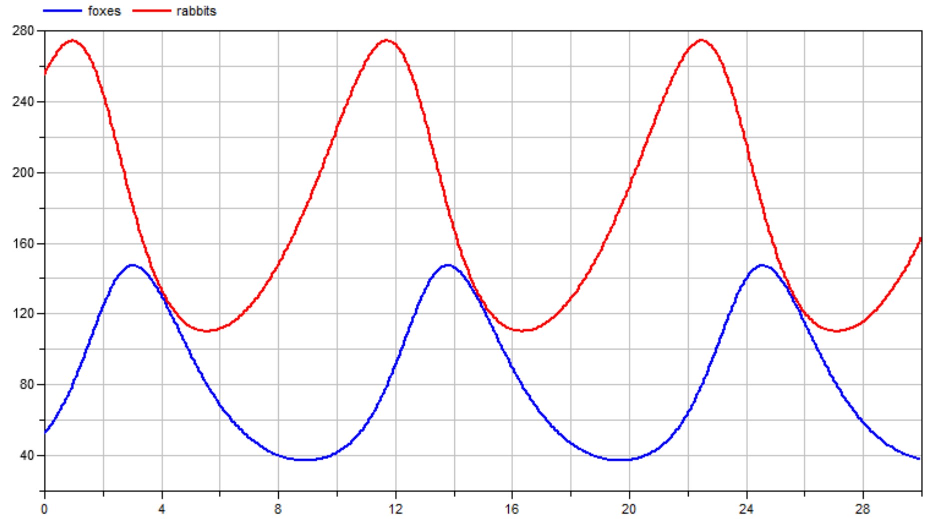
Numeric Method

Numerical Method

Example `[-iim=numeric -iom=nelder_mead_ex]`

```
model forest
  Real foxes;
  Real rabbits;
  Real population;
  Real value;
  […]

initial equation
  der(foxes) = 20;
  value = 11000;

equation
  der(rabbits) = rabbits*g_r - rabbits*foxes*d_rf;
  der(foxes)   = -foxes*d_f + rabbits*foxes*d_rf*g_fr;
  population   = foxes+rabbits;
  value        = priceFox*foxes + priceRabbit*rabbits;
end forest;
```

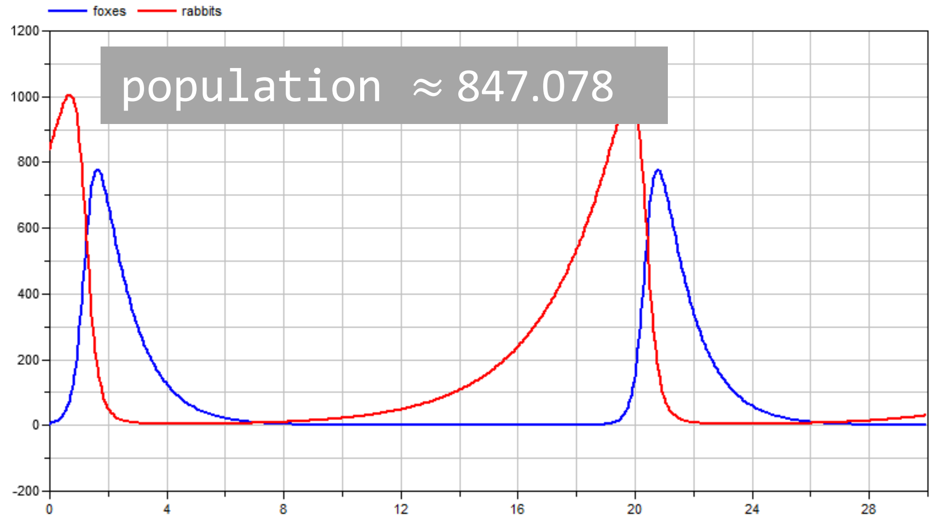


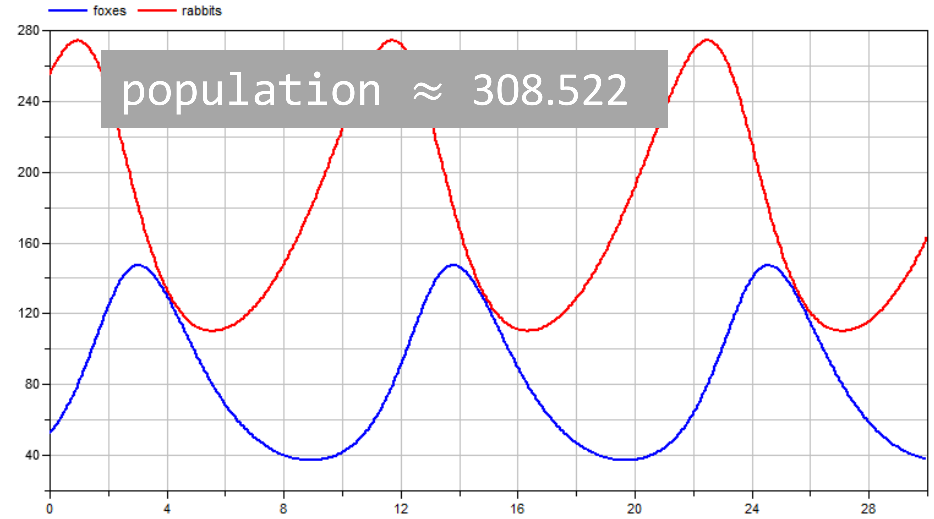


Numerical Method

Example [-iim=numeric -iom=nelder_mead_ex]

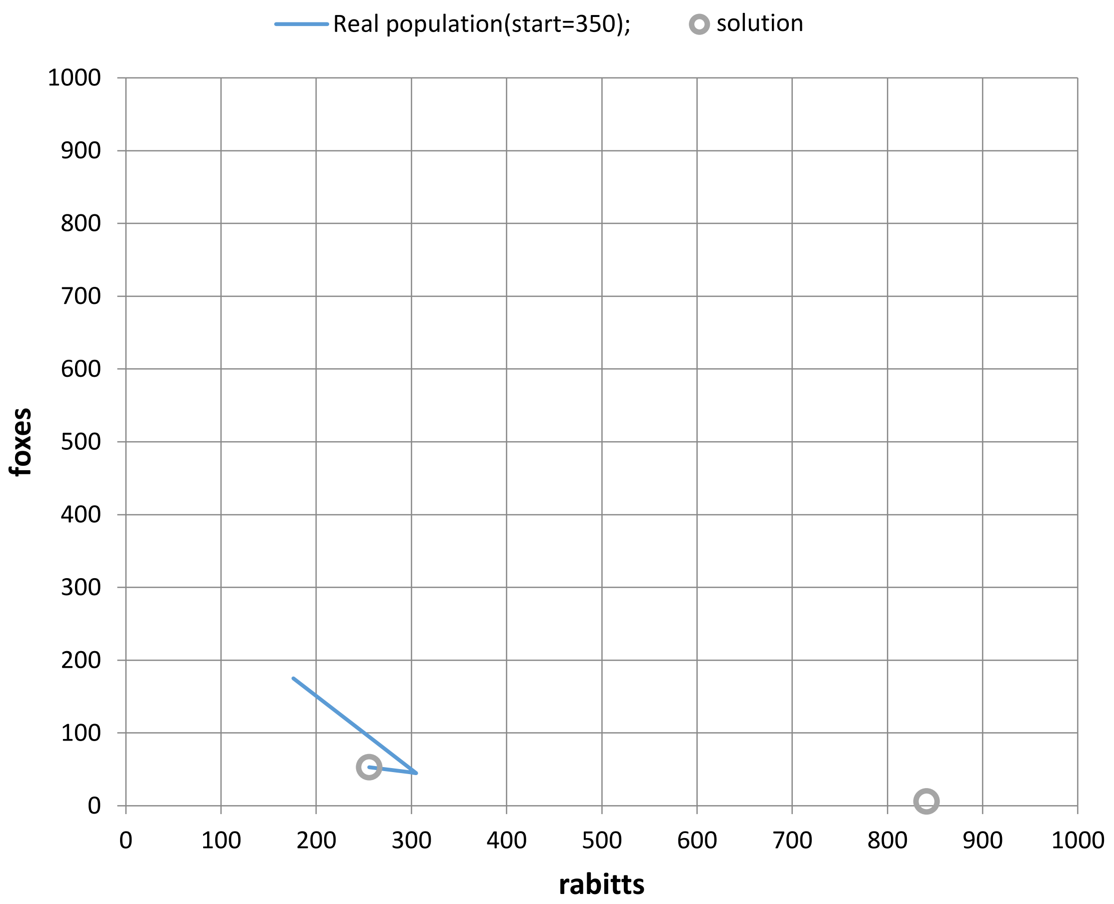
```
model forest
  Real foxes;
  Real rabbits;
  Real population(start=350);
  Real value;
  […]

initial equation
  der(foxes) = 20;
  value = 11000;

equation
  der(rabbits) = rabbits*g_r – rabbits*foxes*d_rf;
  der(foxes)   = -foxes*d_f + rabbits*foxes*d_rf*g_fr;
  population   = foxes+rabbits;
  value        = priceFox*foxes + priceRabbit*rabbits;
end forest;
```

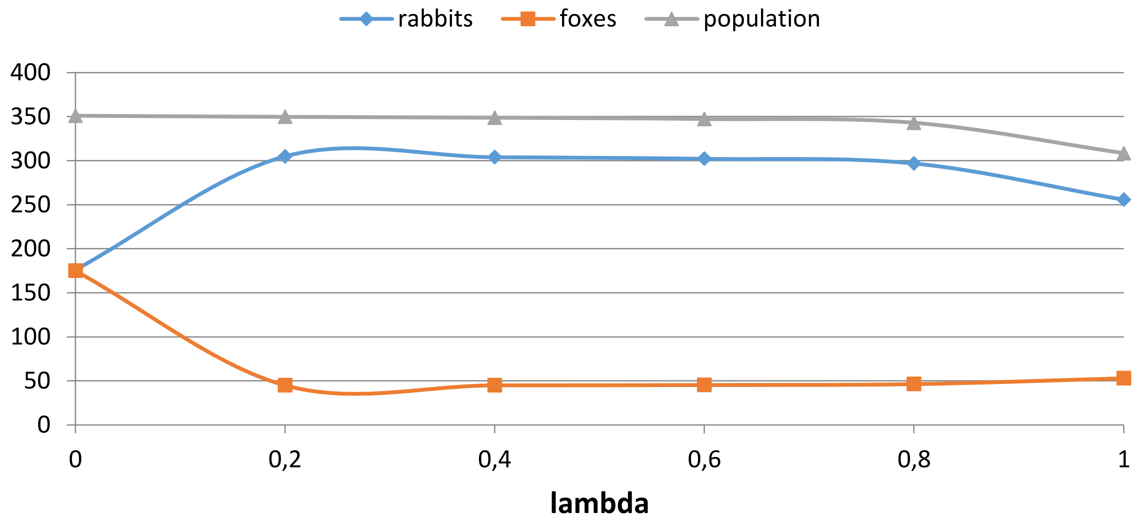




Numerical Method

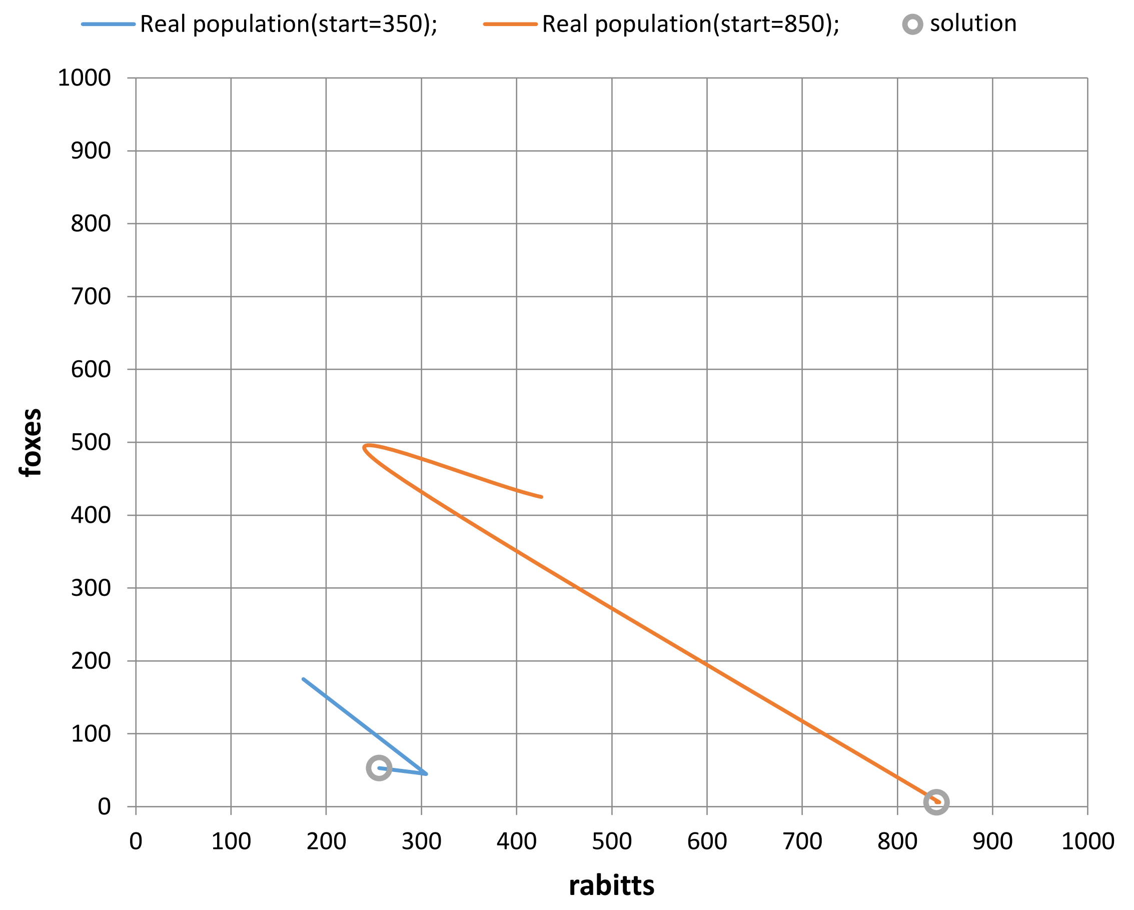
Global Homotopy

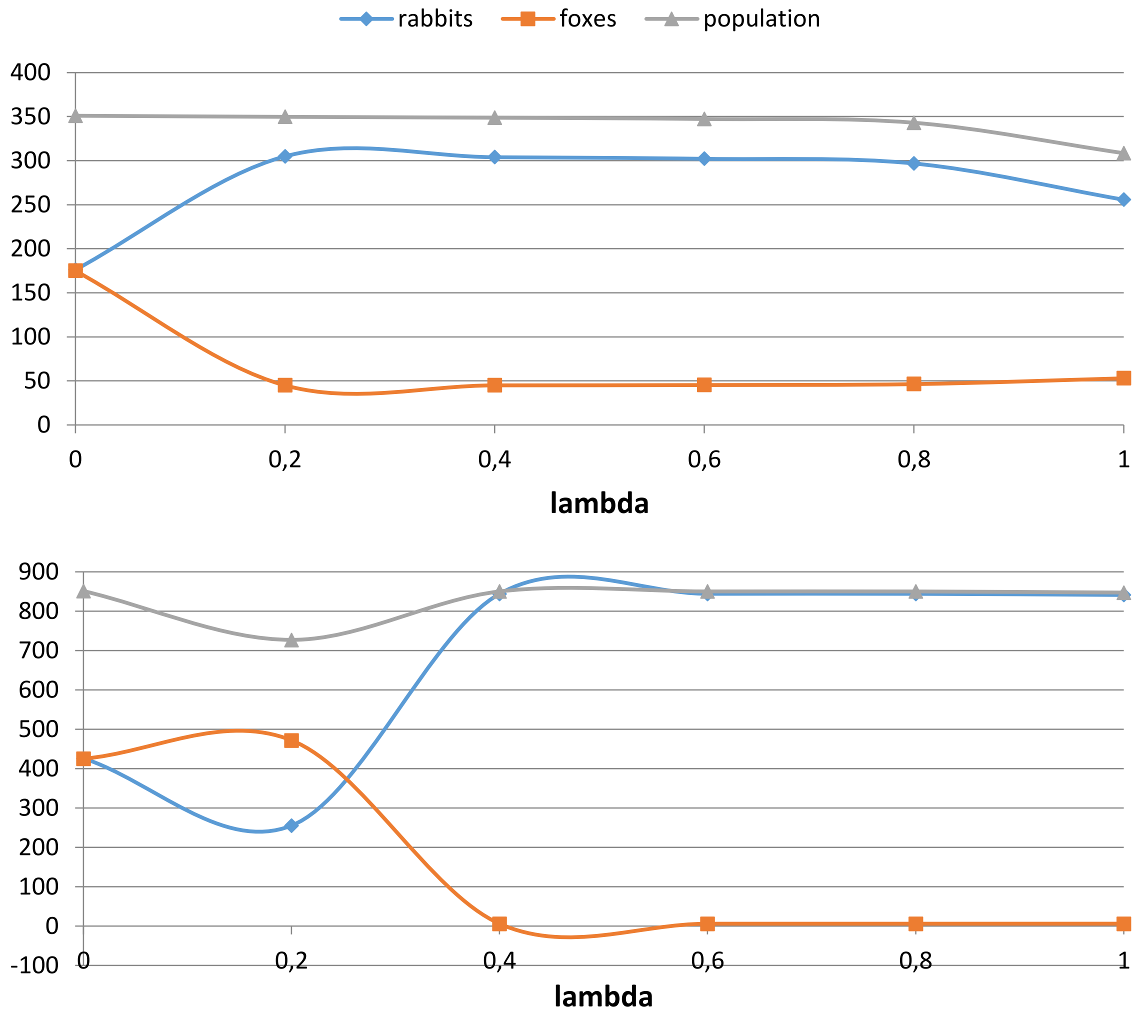




Numerical Method

Global Homotopy





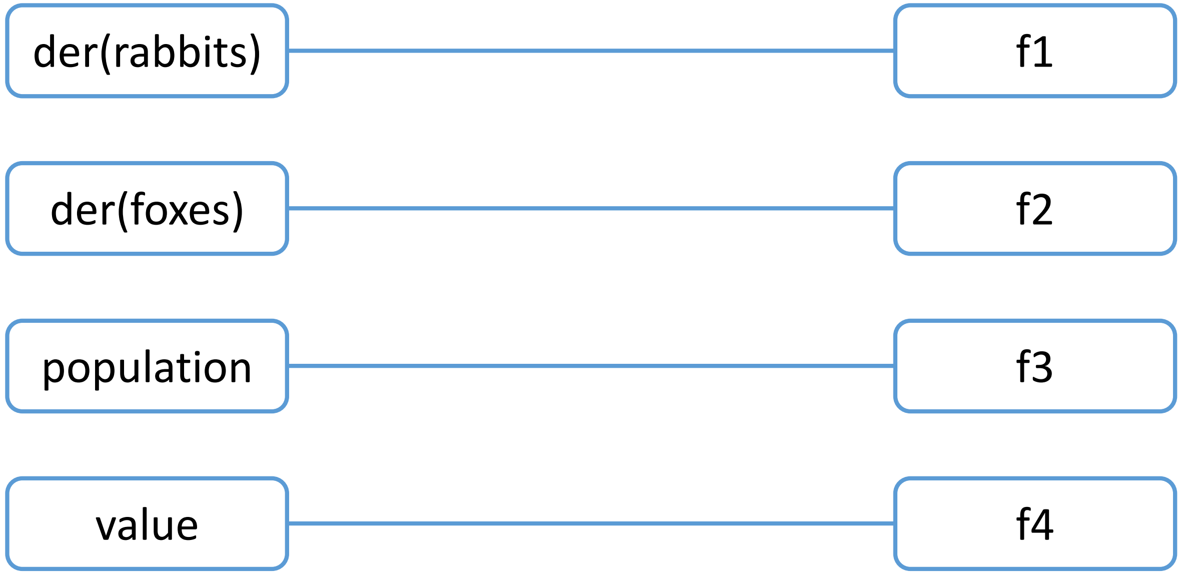
Symbolic Method

Symbolic Method

Example

```
    model forest
      Real foxes;
      Real rabbits;
      Real population;
      Real value;
      […]

    initial equation
      der(foxes) = 20;
      value = 11000;

    equation
f1    der(rabbits) = rabbits*g_r – rabbits*foxes*d_rf;
f2    der(foxes)   = -foxes*d_f + rabbits*foxes*d_rf*g_fr;
f3    population   = foxes+rabbits;
f4    value        = priceFox*foxes + priceRabbit*rabbits;
    end forest;
```

Matching (time-dependent system)

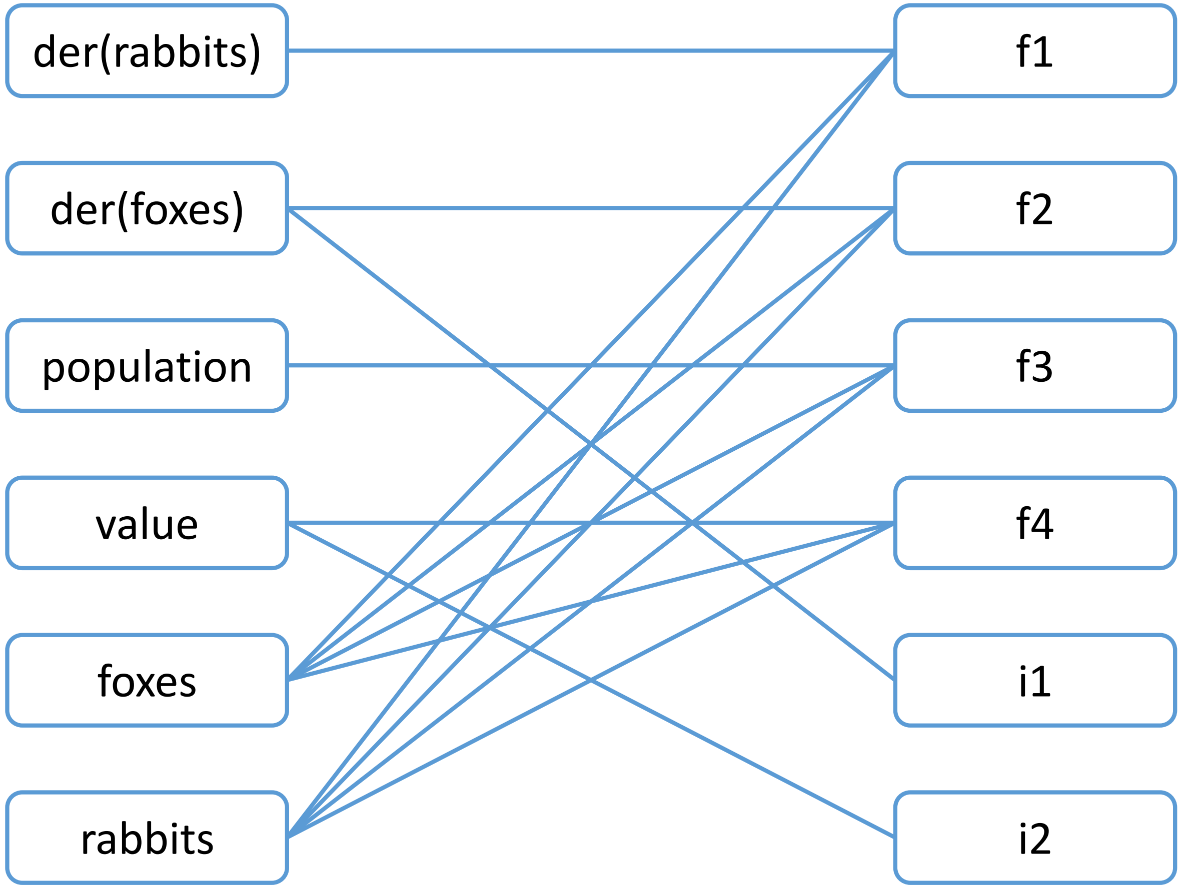


Symbolic Method

Example

```
    model forest
      Real foxes;
      Real rabbits;
      Real population;
      Real value;
      […]

    initial equation
i1    der(foxes) = 20;
i2    value = 11000;

    equation
f1    der(rabbits) = rabbits*g_r – rabbits*foxes*d_rf;
f2    der(foxes)   = -foxes*d_f + rabbits*foxes*d_rf*g_fr;
f3    population   = foxes+rabbits;
f4    value        = priceFox*foxes + priceRabbit*rabbits;
    end forest;
```

Matching (initial system)

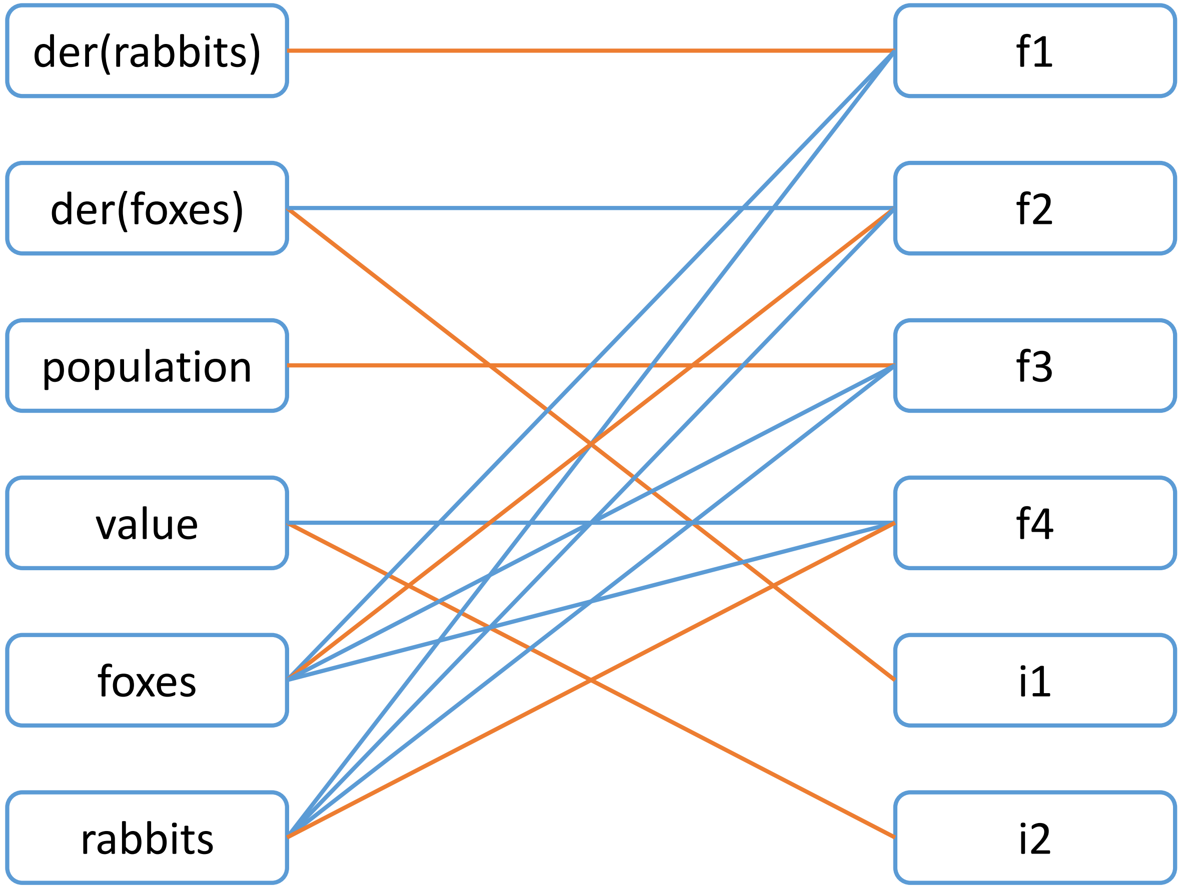


Symbolic Method

Example

```
    model forest
      Real foxes;
      Real rabbits;
      Real population;
      Real value;
      […]

    initial equation
i1    der(foxes) = 20;
i2    value = 11000;

    equation
f1    der(rabbits) = rabbits*g_r – rabbits*foxes*d_rf;
f2    der(foxes)   = -foxes*d_f + rabbits*foxes*d_rf*g_fr;
f3    population   = foxes+rabbits;
f4    value        = priceFox*foxes + priceRabbit*rabbits;
    end forest;
```

Matching (initial system)

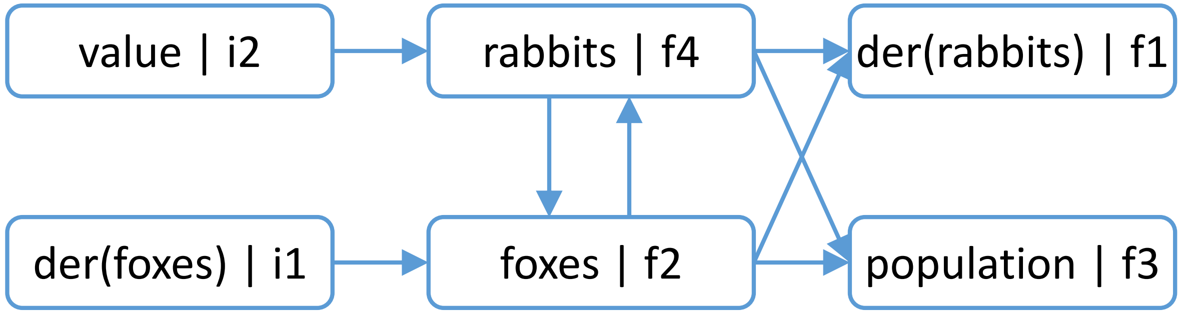


Symbolic Method

Example

```
    model forest
      Real foxes;
      Real rabbits;
      Real population;
      Real value;
      […]

    initial equation
i1    der(foxes) = 20;
i2    value = 11000;

    equation
f1    der(rabbits) = rabbits*g_r – rabbits*foxes*d_rf;
f2    der(foxes)   = -foxes*d_f + rabbits*foxes*d_rf*g_fr;
f3    population   = foxes+rabbits;
f4    value        = priceFox*foxes + priceRabbit*rabbits;
    end forest;
```

Strong Components



Symbolic Method

Example 2

```
model InitWhenModel

  Real lastTime;
  Real interval(start=0);
  Boolean condition(start=false, fixed=true);
  Integer counter(start=0);

equation

  condition = time - pre(lastTime) >= interval;
  when condition then
    lastTime = time;
    counter = pre(counter) + 1;
    interval = pre(interval) + counter;
  end when;

end InitWhenModel;
```

Symbolic Method

Example 2

```
model InitWhenModel

  Real lastTime;
  Real interval(start=0);
  Boolean condition(start=false, fixed=true);
  Integer counter(start=0);

equation

  condition = time – pre(lastTime) >= interval;
  when condition then
    lastTime = time;
    counter = pre(counter) + 1;
    interval = pre(interval) + counter;
  end when;

end InitWhenModel;
```

Discrete Initialization

```
pre(condition) = false; // start-expression
```

```
lastTime = pre(lastTime);

counter = pre(counter)

interval = pre(interval)
```

Symbolic Method

Example 2

```
model InitWhenModel

  Real lastTime;
  Real interval(start=0);
  Boolean condition(start=false, fixed=true);
  Integer counter(start=0);

equation

  condition = time – pre(lastTime) >= interval;
  when condition then
    lastTime = time;
    counter = pre(counter) + 1;
    interval = pre(interval) + counter;
  end when;

end InitWhenModel;
```

Initial System - under-determined

```
pre(condition) = false; // start-expression

lastTime = pre(lastTime);

counter = pre(counter)

interval = pre(interval)

condition = time – pre(lastTime) >= interval; // true
```

Symbolic Method

Example 2

```
model InitWhenModel

  Real lastTime;
  Real interval(start=0);
  Boolean condition(start=false, fixed=true);
  Integer counter(start=0);

equation

  condition = time – pre(lastTime) >= interval;
  when condition then
    lastTime = time;
    counter = pre(counter) + 1;
    interval = pre(interval) + counter;
  end when;

end InitWhenModel;
```

Initial System - under-determined

```
pre(condition) = false; // start-expression

lastTime = pre(lastTime);

counter = pre(counter)

interval = pre(interval)

condition = time – pre(lastTime) >= interval; // true
```

Additional Information needed

```
lastTime(fixed=true)

counter(fixed=true)

Interval(fixed=true)
```

Symbolic Method

Example 2

```
model InitWhenModel

  Real lastTime(fixed=true);
  Real interval(start=0, fixed=true);
  Boolean condition(start=false, fixed=true);
  Integer counter(start=0, fixed=true);

equation

  condition = time - pre(lastTime) >= interval;
  when condition then
    lastTime = time;
    counter = pre(counter) + 1;
    interval = pre(interval) + counter;
  end when;

end InitWhenModel;
```

Initial System

```
pre(lastTime) = 0.0;     // start-expression

pre(interval) = 0.0;     // start-expression

pre(condition) = false;  // start-expression

pre(counter) = 0;        // start-expression


condition = time - pre(lastTime) >= interval; // true

lastTime = pre(lastTime);

counter = pre(counter)

interval = pre(interval)
```

Symbolic Method

Example 2

```
model InitWhenModel

  Real lastTime;
  Real interval(start=0);
  Boolean condition(start=false, fixed=true);
  Integer counter(start=0);

equation

  condition = time - pre(lastTime) >= interval;
  when {initial(), condition} then
    lastTime = time;
    counter = pre(counter) + 1;
    interval = pre(interval) + counter;
  end when;

end InitWhenModel;
```

Initial System

```
pre(condition) = false; // start-expression

condition = time - pre(lastTime) >= interval; // true

lastTime = time;

counter = pre(counter) + 1;

interval = pre(interval) + counter;
```

Additional Information needed

```
lastTime(fixed=true)

counter(fixed=true)

Interval(fixed=true)
```

Conclusion and Outlook

Initialization within OpenModelica

Conclusion

- numeric method is not useable for discrete system-parts
- global-homotopy is quite user-intuitive
- symbolic method is very fast and accurate

Initialization within OpenModelica

Conclusion

- numeric method is not useable for discrete system-parts
- global-homotopy is quite user-intuitive
- symbolic method is very fast and accurate

Outlook

- introduce the global-homotopy-benefit within the symbolic method
- introduce handling for over-determined systems into the symbolic method

Questions?