HSBI

Hochschule Bielefeld

University of Applied Sciences and Arts

# Efficient Model Evaluation with the GBODE multi-rate Solver

Philip Hannebohm, Bernhard Bachmann

05.02.2026

iDaS

Institute for Data Science Solutions

Hochschule Bielefeld
University of Applied Sciences and Arts

- What is GBODE
  - Multi-rate Simulation
- Selective Evaluation of RHS
  - Dependency Graph
  - Dependency Propagation
  - Selective Evaluation
- Saving the result
  - Global equidistant grid
  - Collocation nodes

- What is GBODE
  - Multi-rate Simulation
- Selective Evaluation of RHS
  - Dependency Graph
  - Dependency Propagation
  - Selective Evaluation
- Saving the result
  - Global equidistant grid
  - Collocation nodes

# GENERIC BIRATE ODE SOLVER

Introduced by B. Bachmann in 2022/23.

# GENERIC BIRATE ODE SOLVER

Introduced by B. Bachmann in 2022/23.

Single-rate Mode of GBODE:

- Can select integration scheme from a variety of butcher tableaus
- In principle new tableaus can be added quite easily

# GENERIC BIRATE ODE SOLVER

Introduced by B. Bachmann in 2022/23.

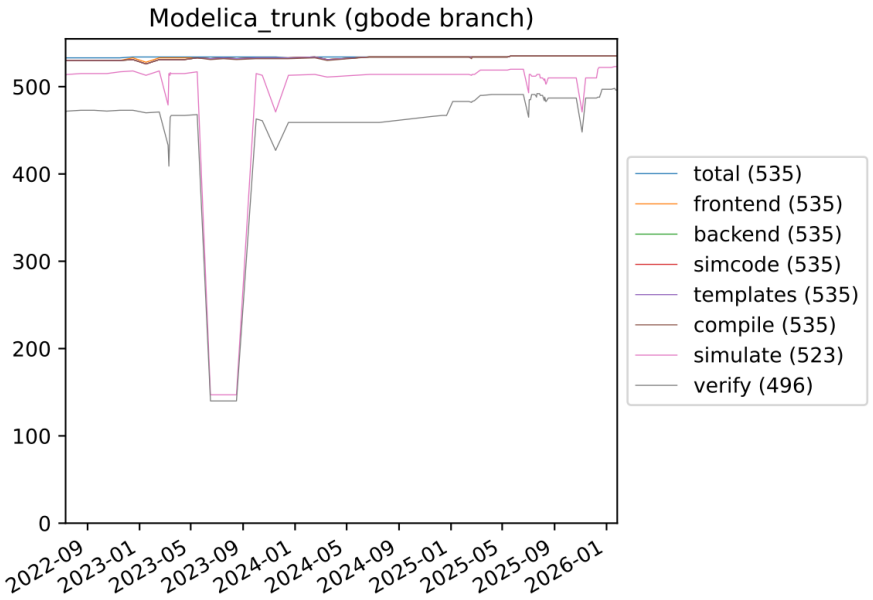
Single-rate Mode of GBODE:

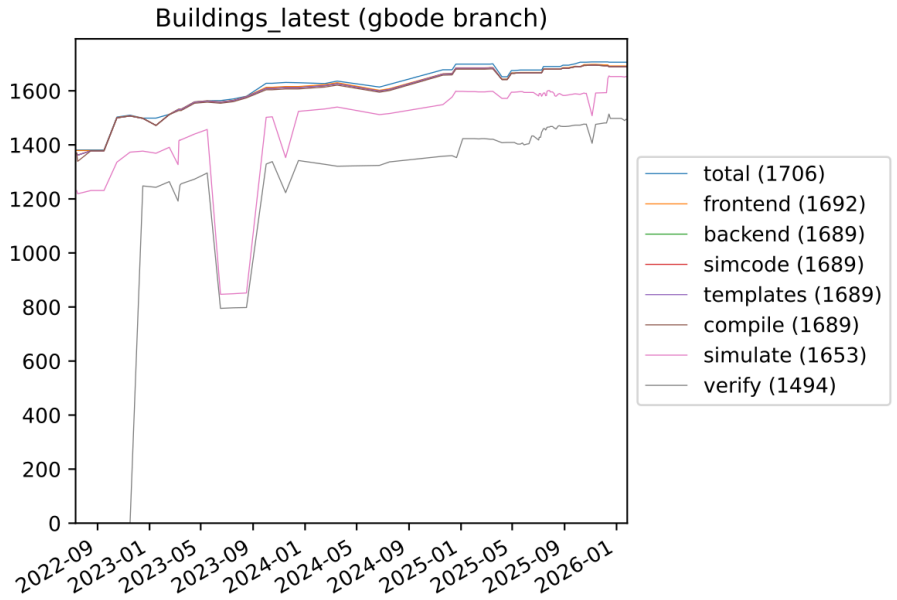
- Can select integration scheme from a variety of butcher tableaus
- In principle new tableaus can be added quite easily

Bi-rate Mode of GBODE:

- Same generic functionality as single-rate
- Dynamic separation of fast and slow states (inner/ outer integration)
- ...still not quite as robust as single-rate

# COVERAGE




https://libraries.openmodelica.org/branches/history/gbode/

# COVERAGE

## Modelica_trunk

| | Total | Frontend | Backend | SimCode | Templates | Compilation | Simulation | Verification |
|---|---|---|---|---|---|---|---|---|
| master | 535 | 535 | 535 | 535 | 535 | 535 | 525 | 499 |
| gbode | 535 | 535 | 535 | 535 | 535 | 535 | 523 | 496 |
| cvode | 535 | 535 | 535 | 535 | 535 | 535 | 504 | 477 |

## Buildings_latest

| | Total | Frontend | Backend | SimCode | Templates | Compilation | Simulation | Verification |
|---|---|---|---|---|---|---|---|---|
| master | 1706 | 1692 | 1689 | 1689 | 1689 | 1689 | 1655 | 1506 |
| gbode | 1706 | 1692 | 1689 | 1689 | 1689 | 1689 | 1653 | 1494 |
| cvode | 1706 | 1692 | 1689 | 1689 | 1689 | 1689 | 1628 | 1484 |

# MULTI-RATE SIMULATION

States with dynamic behavior on different time scales. Use different time steps for slow and fast states $y_s$, $y_f$ to solve

$$y' = \begin{pmatrix} y'_s \\ y'_f \end{pmatrix} = \begin{pmatrix} f_s(y_s, y_f, t) \\ f_f(y_s, y_f, t) \end{pmatrix} = f(y, t), \qquad y(t_0) = y_0$$

where $y_s$ is interpolated when evaluating $f_f(y_s, y_f, t)$.

But we don't know the partitioning a priori. So identify fast and slow variables at runtime based on some error measure.

Bonaventura et al. (2025) "Self-Adjusting Multi-Rate Runge-Kutta Methods: Analysis and Efficient Implementation in An Open Source Framework" In: Journal of Scientific Computing 105. DOI: 10.1007/s10915-025-03049-y.

What is GBODE
Multi-rate Simulation

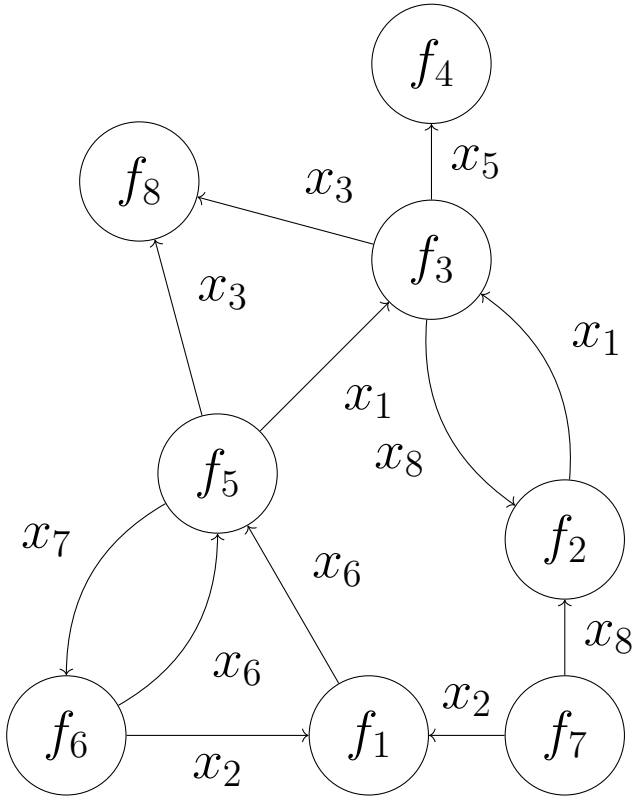
Selective Evaluation of RHS
Dependency Graph
Dependency Propagation
Selective Evaluation

Saving the result
Global equidistant grid
Collocation nodes

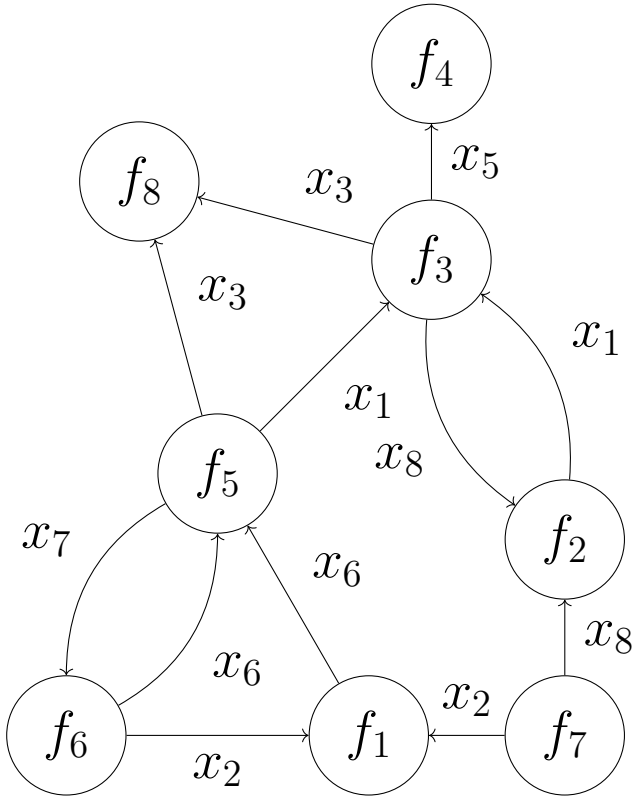
# CAUSALIZATION

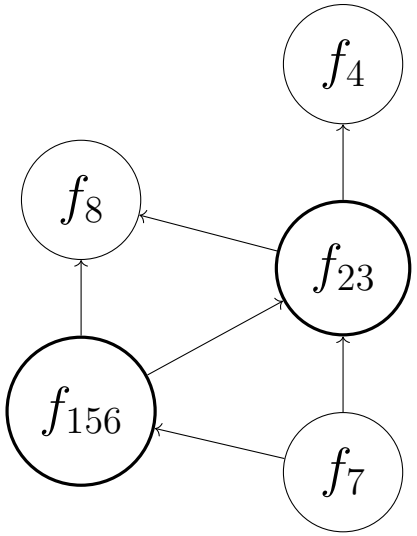
$$
\begin{aligned}
0 &= f_1(\underline{x_2}, x_6) & 0 &= f_5(x_1, x_3, \underline{x_6}, x_7)\\
0 &= f_2(x_1, \underline{x_8}) & 0 &= f_6(x_2, x_6, \underline{x_7})\\
0 &= f_3(\underline{x_1}, x_3, x_5, x_8) & 0 &= f_7(x_2, \underline{x_4}, x_8)\\
0 &= f_4(\underline{x_5}) & 0 &= f_8(\underline{x_3})
\end{aligned}
$$

# CAUSALIZATION

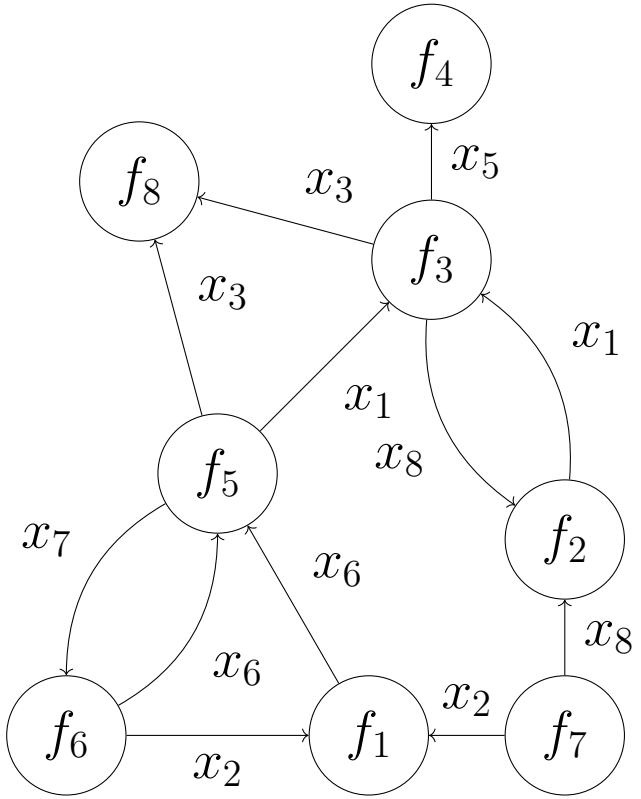


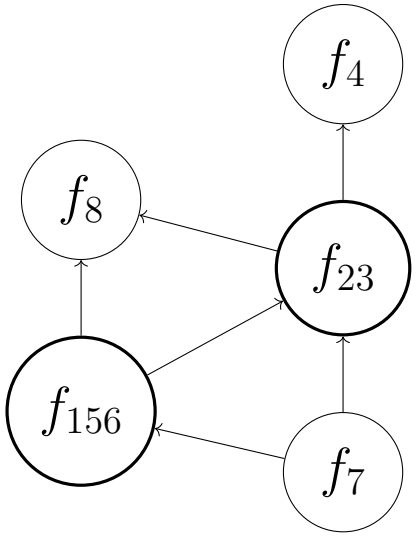
# CAUSALIZATION

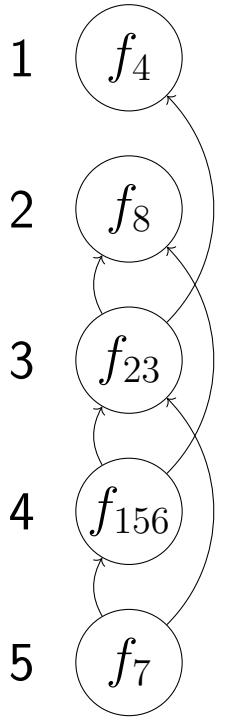


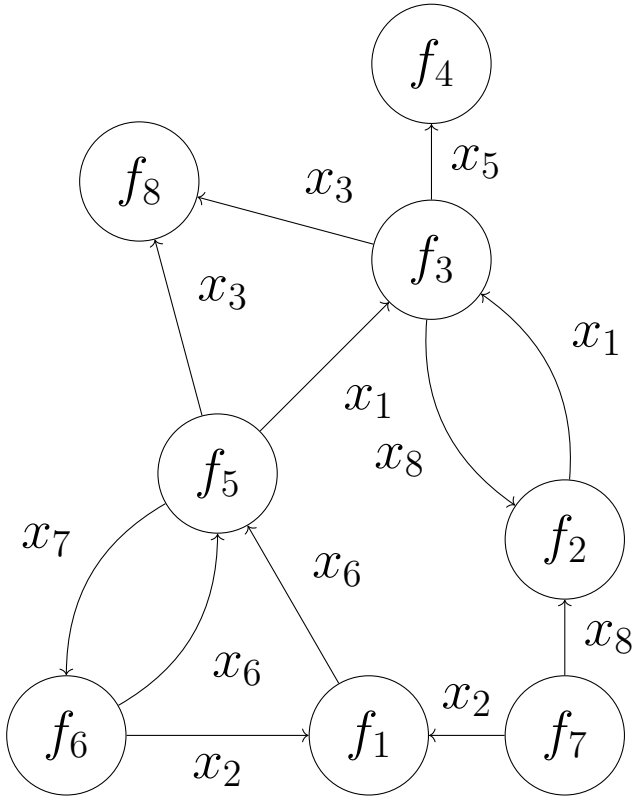
# CAUSALIZATION

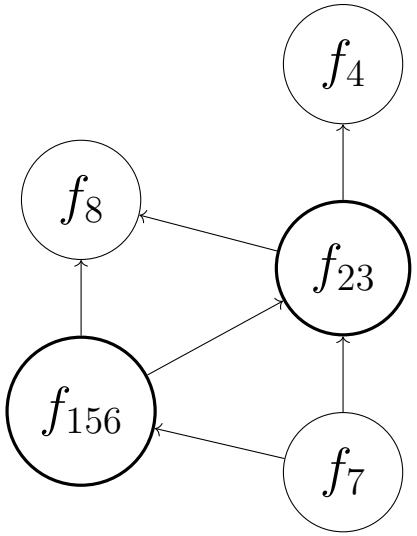


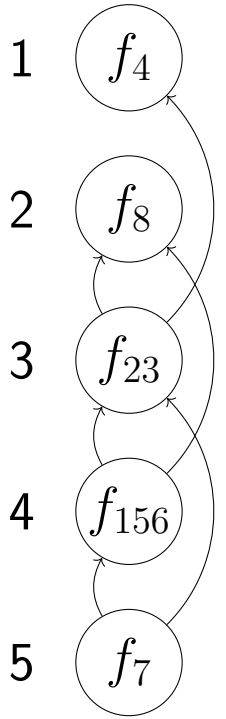



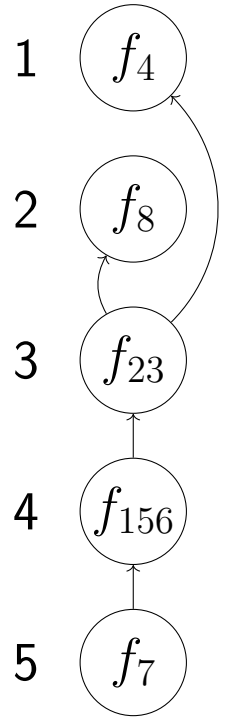



# CAUSALIZATION

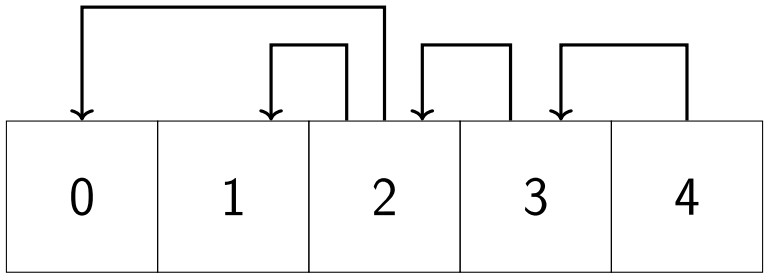


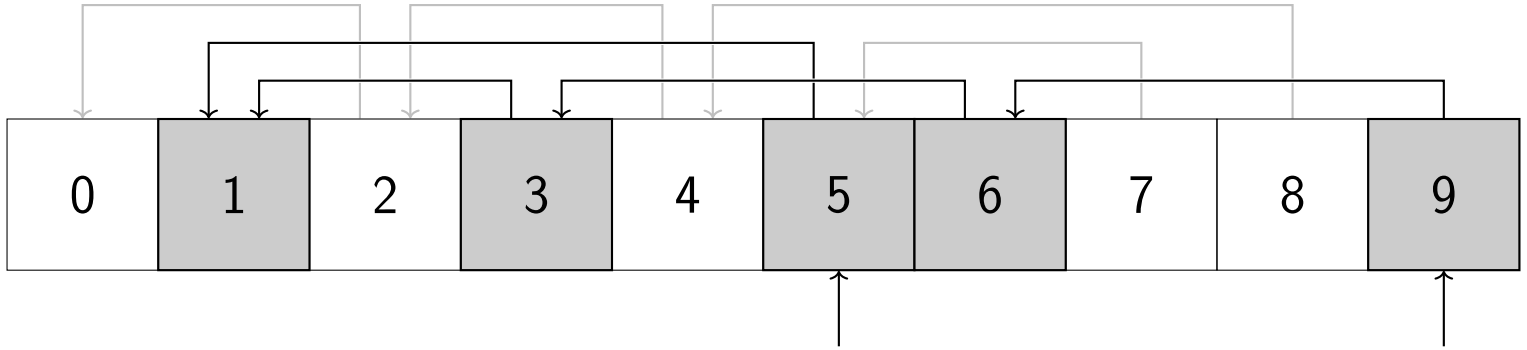
# DEPENDENCY GRAPH




The dependency graph is a DAG (Directed Acyclic Graph) Evaluating some equation requires other equations to be evaluated first.

# DEPENDENCY PROPAGATION

propagate by traversing linearly from larger to smaller index



# SELECTIVE EVALUATION

```
void functionODE()
{
  eqFunction_1();
  // [...]
  eqFunction_N();
}
```

# SELECTIVE EVALUATION

```
void functionODE()
{
  eqFunction_1();
  // [...]
  eqFunction_N();
}
```

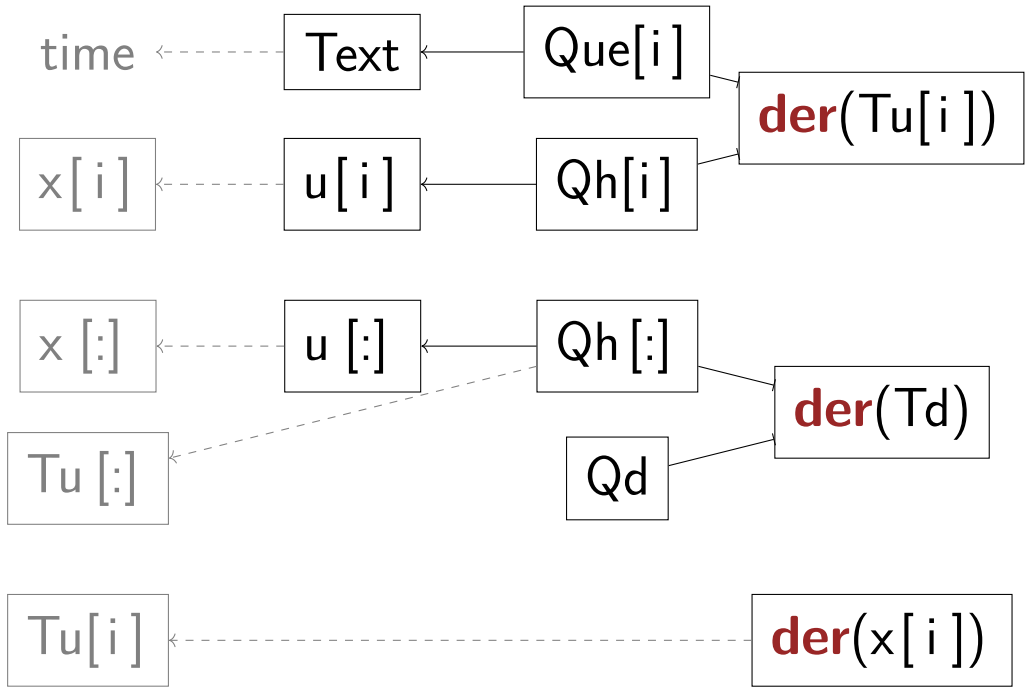
```
void functionODE()
{
  static f_ptr eqFunctions[N] = {
    eqFunction_1,
    // [...]
    eqFunction_N
  };
  for (int i = 0; i < evalN; i++)
    eqFunctions[evalI[i]]();
}
```

# EXAMPLE MODEL

```
Text = 278.15 + 8*sin(2*pi*time/86400)"f1";
Cd*der(Td) = Qd - sum(Qh)                "f2";
Qd = sat(Kp*(Td0 - Td),0, Qmax)          "f3";
for i in 1:N loop
  Qh[i] = Gh*(Td - Tu[i])*u[i]           "f4";
  Que[i] = Gu*(Tu[i] - Text)             "f5";
  Cu[i]*der(Tu[i]) = Qh[i] - Que[i]      "f6";
  der(x[i]) = a*hist(x[i], Tu0 - Tu[i],
                     Teps)               "f7";
  u[i] = sat(b*x[i], -0.5, 0.5) + 0.5    "f8";
end for;
```

# EXAMPLE MODEL

```
Text = 278.15 + 8*sin(2*pi*time/86400)"f1";
Cd*der(Td) = Qd - sum(Qh)                "f2";
Qd = sat(Kp*(Td0 - Td),0, Qmax)          "f3";
for i in 1:N loop
  Qh[i] = Gh*(Td - Tu[i])*u[i]           "f4";
  Que[i] = Gu*(Tu[i] - Text)             "f5";
  Cu[i]*der(Tu[i]) = Qh[i] - Que[i]      "f6";
  der(x[i]) = a*hist(x[i], Tu0 - Tu[i],
                     Teps)               "f7";
  u[i] = sat(b*x[i], -0.5, 0.5) + 0.5    "f8";
end for;
```



# TIME MEASUREMENTS

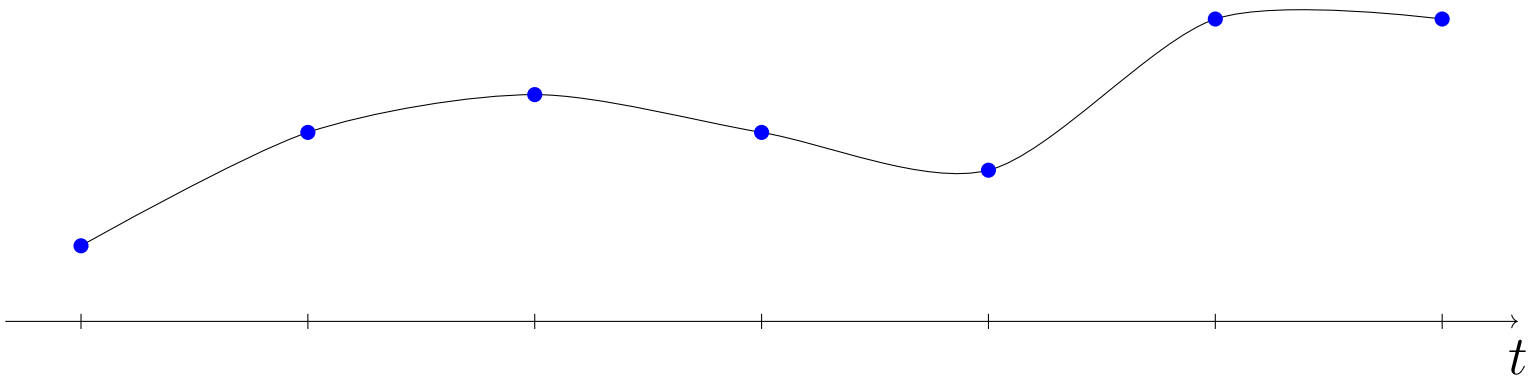
To come...

What is GBODE
Multi-rate Simulation

Selective Evaluation of RHS
Dependency Graph
Dependency Propagation
Selective Evaluation

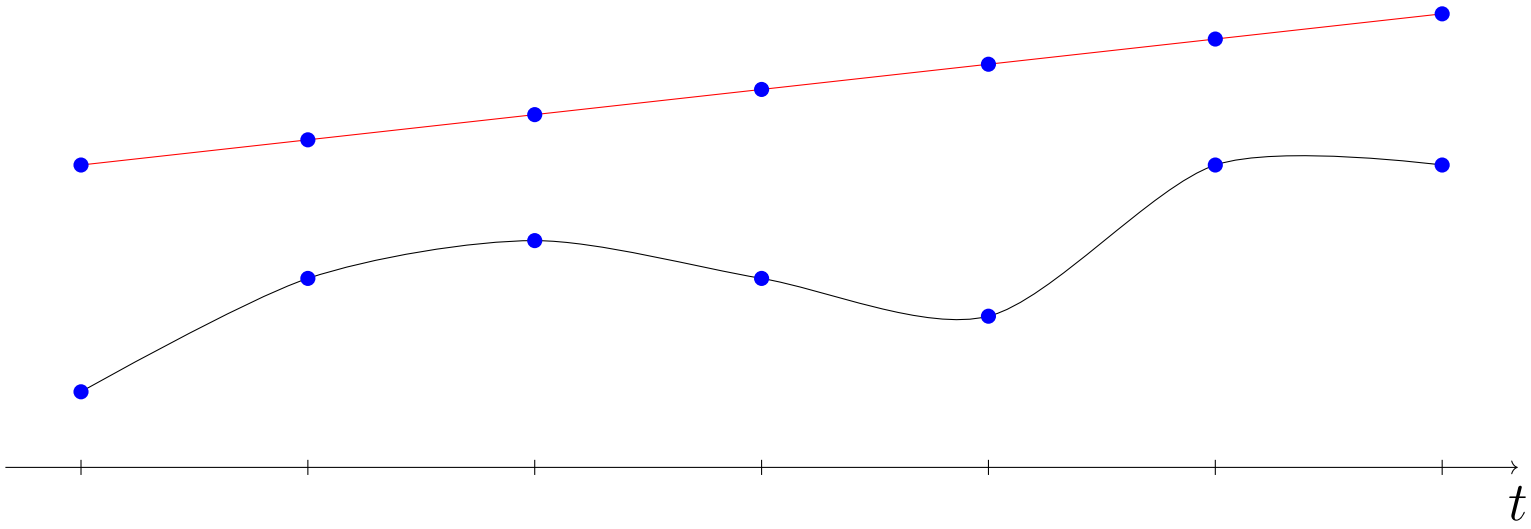
Saving the result
Global equidistant grid
Collocation nodes

# GLOBAL EQUIDISTANT GRID



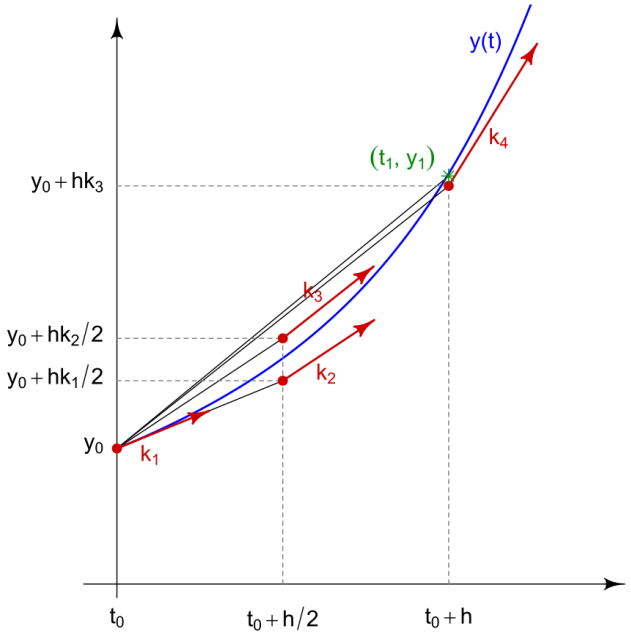

Currently this is the only way to store results with the de-facto standard mat-format.
Problem: when evaluating only fast states, other variables are unknown or equivalently: when emitting at finer grids the benefit of multi-rate is lost

# GLOBAL EQUIDISTANT GRID




Currently this is the only way to store results with the de-facto standard mat-format.
Problem: when evaluating only fast states, other variables are unknown or equivalently: when emitting at finer grids the benefit of multi-rate is lost

# COLLOCATION NODES




Dense Output can reconstruct the states smoothly from collocation nodes.

Post Processing will be needed to plot other algebraic variables.