Simulation Tools

Repetition: Multistep Methods

Claus Führer

Spring 2021

Initial Value Problems

(Time-)Simulation requires the solution of so-called initial value problems:

$$\dot{y}(t) = f(t, x(t)) \quad y(t_0) = y_0 \quad y(t) \in \mathbb{R}^n$$

- moderate size applications (this course):
  $n \leq 500$, rhs-function f expansive to evaluate, "linear algebra" subproblems cheap, problems stiff or nonstiff
- applications from PDE space discretization (not in this course):
  $1000 \leq n \leq 50000$, rhs-function f linear or cheap to evaluate, special algorithms for "linear algebra" subproblems, problems often stiff or highly oscillatory

Indata

Indata for a method:

- rhs-function: often given as a call to special purpose library functions
- time horizon: t_0, t_e
- initial value: y_0
- Communication interval Δt_{out}

Outdata:

- Approximations to $y(t_{\text{out}})$
- Error flags, performance statistics

Two method classes

Multistep methods

Multistep methods are methods, which require starting values from several previous steps.

There are two important families of multistep methods

- *Adams methods* (explicit: *Adams–Bashforth* as predictor, implicit: *Adams–Moulton* as corrector), used together with fixed point iteration for nonstiff problems,
- *BDF methods* (implicit), together with Newton iteration used for stiff problems

Adams methods

For deriving ADAMS methods we transform the ODE

$$\dot{y} = f(t, y) \quad \text{with} \quad y(t_0) = y_0$$

into its integral form

$$y(t) = y_0 + \int_{t_0}^{t} f(\tau, y(\tau))\, d\tau$$

and partition the interval into

$$t_0 < t_1 < \cdots < t_i < t_{i+1} = t_i + h_i < \cdots < t_e$$

Adams methods (Cont.)

Thus

$$y_{n+1} = y(t_{n+1}) = y_n + \int_{t_n}^{t_{n+1}} f(\tau, y(\tau))\, d\tau$$

Let $u_{n+1-i}, i = 1, \ldots, k$ be previously computed values and π_k^{p} the unique polynomial which interpolates

$$f(t_{n+1-i}, u_{n+1-i}), \quad i = 1, \ldots, k$$

The we define a method by

$$u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} \pi_k^{\mathrm{p}}(\tau)\, d\tau$$

Adams methods (Cont.)

Example (2-step method, $k = 2$):
Using the concept of Lagrange polynomials gives

$$\begin{aligned}\pi_k^{\mathrm{p}}(t) &= f(t_n, u_n)L_0(t) + f(t_{n-1}, u_{n-1})L_1(t) \\ &= f(t_n, u_n)\frac{t - t_{n-1}}{t_n - t_{n-1}} + f(t_{n-1}, u_{n-1})\frac{t - t_n}{t_{n-1} - t_n}\end{aligned}$$

Integration gives:

$$u_{n+1} = u_n + h(\frac{3}{2}f(t_n, u_n) - \frac{1}{2}f(t_{n-1}, u_{n-1})$$

This is the Adams–Bashforth-2 method.

Adams methods (Cont.)

Adams–Bashforth methods are explicit, their general form is

$$u_{n+1} = u_n + h \sum_{i=1}^{k} \beta_{k-i} f(t_{n+1-i}, u_{n+1-i})$$

Adams–Moulton methods are constructed in a similar way.

Here the unknown value $f(t_{n+1}, u_{n+1})$ is taken as an additional interpolation point.

This makes the method implicit.

Adams methods (Cont.)

Examples (Adams–Moulton):

$$k = 0: \quad u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}) \qquad \text{implicit Euler metho}$$

$$k = 1: \quad u_{n+1} = u_n + h\left(\tfrac{1}{2}f(t_{n+1}, u_{n+1}) + \tfrac{1}{2}f(t_n, u_n)\right) \qquad \text{Trapezoid}$$

$$k = 2: \quad u_{n+1} = u_n + h\left(\tfrac{5}{12}f(t_{n+1}, u_{n+1}) + \tfrac{8}{12}f(t_n, u_n) - \tfrac{1}{12}f(t_{n-1}, u_{n-1})\right.$$

The general form is

$$u_{n+1} = u_n + h\sum_{i=0}^{k} \bar{\beta}_{k-i} f(t_{n+1-i}, u_{n+1-i})$$

Starting a multistep method

To start a multistep method one applies

- the initial value
- then a one-step method, to get two values
- then a two-step method, to get three values
- and so on ...

Multistep methods in General

The general form of a linear multistep method reads

$$\sum_{i=0}^{k} \alpha_{k-i} u_{n+1-i} - h_n \sum_{i=0}^{k} \beta_{k-i} f(t_{n+1-i}, u_{n+1-i}) = 0.$$

For starting a multistep method k starting values $u_0, \ldots, u_{k-1}$ are required.

Global Error

The quantity of interest is the *global error* of the method at a given time point t_n

$$e_n := y(t_n) - u_n,$$

with $n = t_n/h$.
If for exact starting values $e_n = \mathcal{O}(h)$, then the method is said to be *convergent*. More precisely, a method is *convergent of order* p, if

$$e_n = \mathcal{O}(h^p).$$

Local Residual

To make a statement about the behavior of the global error, we have to introduce and study first the *local residual*:
Let y be a differentiable function, then the quantity

$$l(y, t_n, h) := \sum_{i=0}^{k} \alpha_{k-i} y(t_{n-i}) - h \sum_{i=0}^{k} \beta_{k-i} \dot{y}(t_{n-i})$$

is called the *local residual* of the method.

Asymptotic Form of Local Residual

The local residual of a method with order of consistency p takes the form

$$l(y, t, h) = c_{p+1} h^{p+1} y^{(p+1)}(t) + \mathcal{O}(h^{p+2}).$$

Adams–Bashforth methods have order of consistency k, Adams–Moulton methods have order of consistency $k + 1$, and BDF methods have order of consistency k

The Global Error Increment

Every step contributes to the global error by the *global error increment*. These increments are amplified or damped by the differential equation's and method's stability properties.
The global error increment is defined as the difference between the numerical solution and the locally exact solution:

$$\varepsilon_n := \bar{y}_n - y(t_n)$$

where

$$0 = \alpha_k \bar{y}_n - h f(t_n, \bar{y}_n) + \sum_{i=1}^{k} \alpha_{k-i} y(t_{n-i}) - h \sum_{i=1}^{k} \beta_{k-i} f(t_{n-i}, y(t_{n-i}))$$

The Global Error Increment and Local Residual

The is a method dependent matrix A_k such that

$$\varepsilon_n = A_k^{-1} l(y, t_n, h) + \mathrm{higherorderterms}$$

or alternatively

$$\varepsilon_n = A_k^{-1} c_{p+1} h^{p+1} y^{(p+1)}(t_n) + \mathcal{O}(h^{p+2})$$

Iterations

Nonlinear System

$$y_n = h\beta_k f(y_n) + \text{old values}$$

Fixed point iteration (functional iteration, `CV_FUNCTIONAL`)

$$y_n(i+1) = h\beta_k f(y_n^{(i)}) + \text{old values}$$

Newton iteration (`CV_NEWTON`)

$$G'(y_n^{(i+1)})\Delta y = -G(y_n^{(i)})$$

with $G(y) = y - h\beta_k f(y) - \text{old values}$.

variable coefficient vs fixed leading coefficient

Coefficients depend on step size ratios:

$$r_i = \frac{h_{n-i}}{h_{n-i-1}}$$

$$\alpha_{k-i} = \alpha_{k-i}(r_0, r_1, \ldots, r_k) \quad \beta_{k-i} = \beta_{k-i}(r_0, r_1, \ldots, r_k)$$

In CVODE fixed leading coefficient,i.e.

$$\beta_k = \beta_k(r_0)$$

(saves Jacobian evaluations).

Tolerances and Norms

SUNDIALS uses weighted norms (state dependent weights)

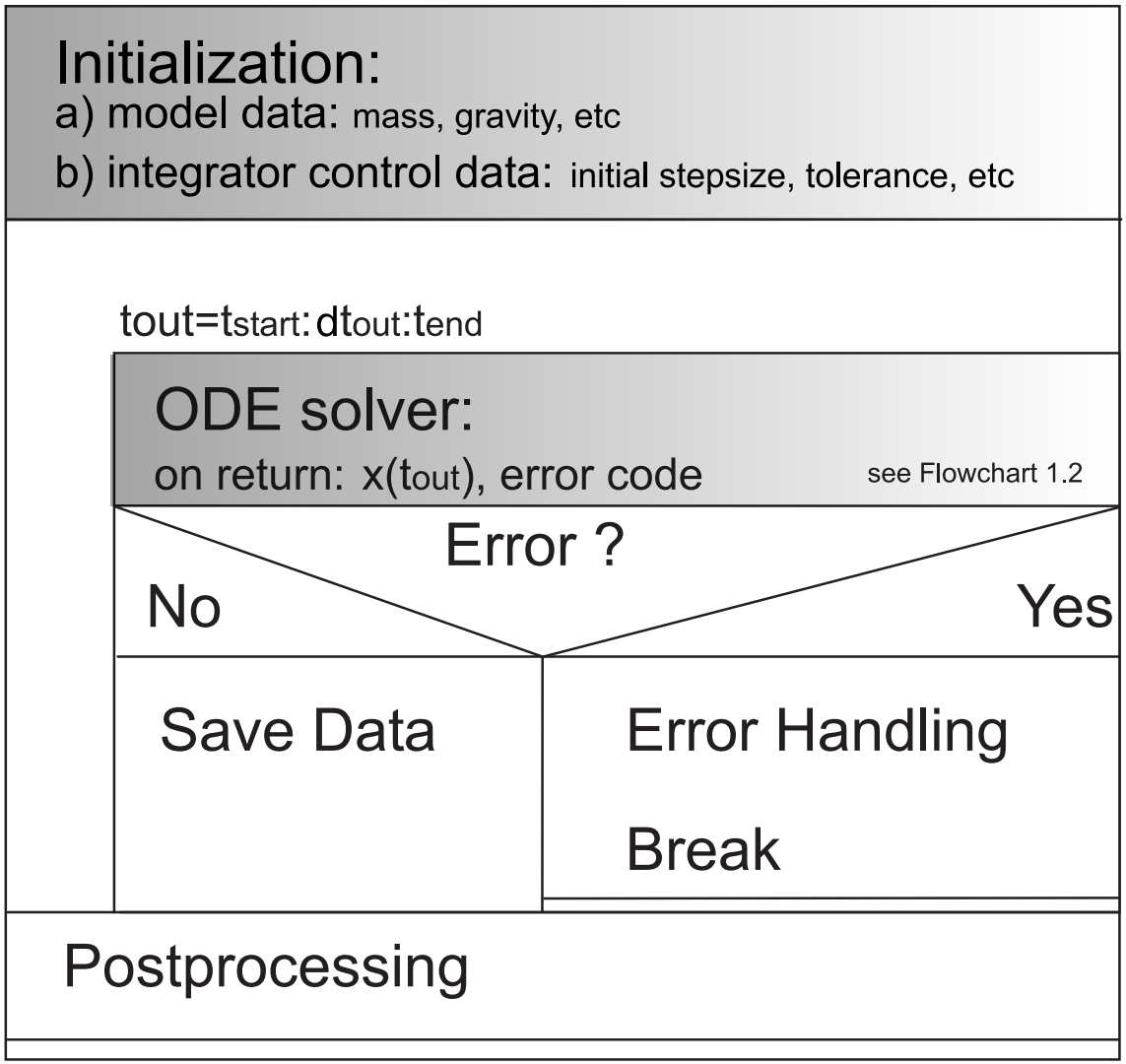
Weights

$$W_i = \mathrm{RTOL} \cdot |y_i| + \mathrm{ATOL}_i$$

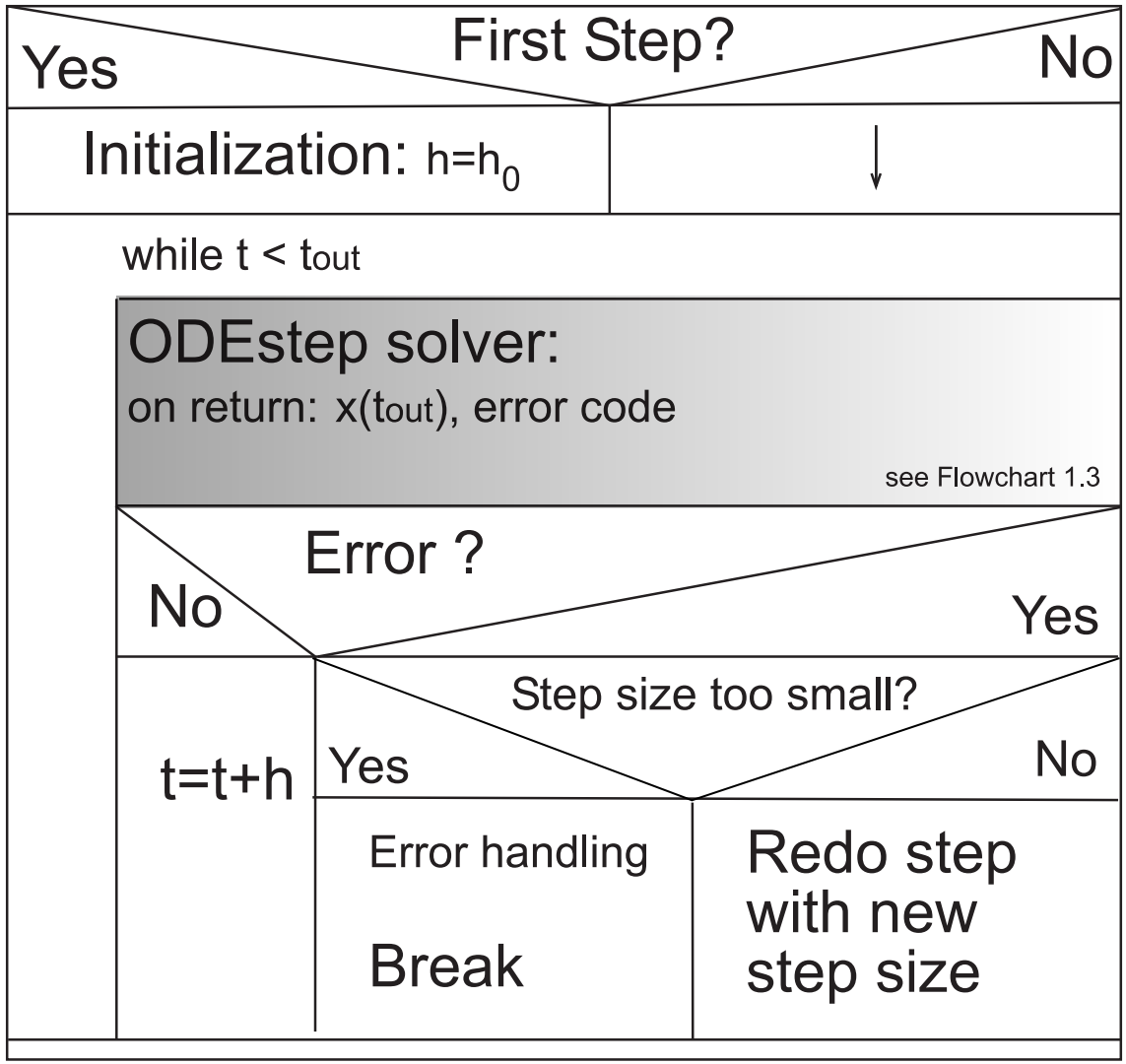
Weighted root mean square norm

$$\|v\| = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\frac{v_i}{W_i})^2}$$

Flowchart 1: Generic integrator call



Flowchart 1.2: Generic single step integrator call



Flowchart 1.3: Generic integrator organisation