Numerical methods for dynamical systems

Alexandre Chapoutot

ENSTA Paris
master CPS IP Paris

2020-2021

Part IV

Numerical methods for discontinuous IVP-ODE

Recall our starting point is the IVP of ODE defined by

$$\dot{\mathbf{y}} = f(t, \mathbf{y}) \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0 \ , \tag{1}$$

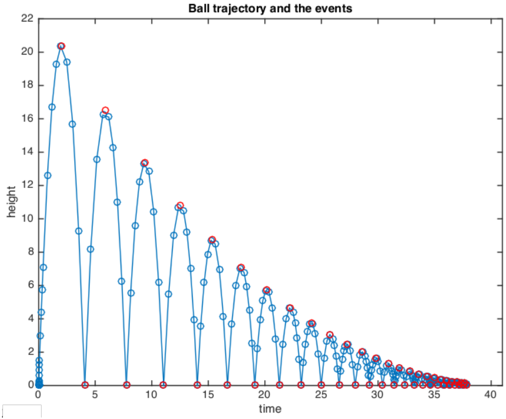
for which we want the solution $\mathbf{y}(t; \mathbf{y}_0)$ given by numerical integration methods i.e. a sequence of pairs $(t_i, \mathbf{y}_i)$ such that

$$\mathbf{y}_i \approx \mathbf{y}(t_i; \mathbf{y}_0) \ .$$

Why do we consider discontinuities?

Need to model

- non-smooth behaviors, *e.g.*, solid body in contact with each other
- interaction between computer and physics, *e.g.*, control-command systems
- constraints on the system, *e.g.*, robotic arm with limited space



There are two kinds of events:

- **time event:** only depending on time as sampling
- **state event:** depending on a particular value of the solution of ODE or DAE.

To handle these events we need to adapt the simulation algorithm.

- Time events are known before the simulation starting. Hence we can use the step-size control to handle this.
- State event should be detect and handle on the fly. New algorithms are needed.

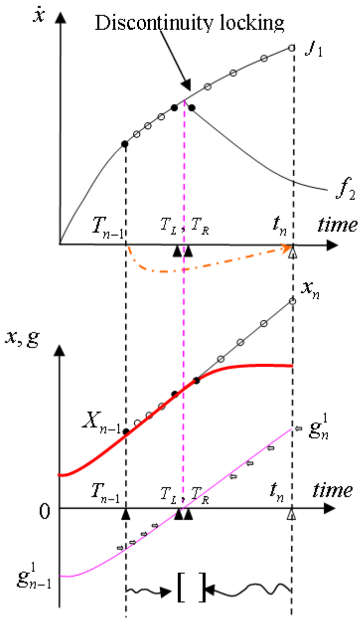
An IVP for ODE with discontinuities is defined by

$$\dot{\mathbf{y}} = \begin{cases} f_1(t, \mathbf{y}) & \text{if } g(t, \mathbf{y}) \geqslant 0 \\ f_2(t, \mathbf{y}) & \text{otherwise} \end{cases} \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0 \ , \tag{2}$$

for which we want the solution $\mathbf{y}(t; \mathbf{y}_0)$ given by numerical integration methods i.e. a sequence of pairs $(t_i, \mathbf{y}_i)$ such that

$$\mathbf{y}_i \approx \mathbf{y}(t_i; \mathbf{y}_0) \ .$$

Example: zero-crossing detection



A simple example

$$\dot{\mathbf{y}} = \begin{cases} f_1(t, \mathbf{y}) & \text{if } g(\mathbf{y}) \geqslant 0 \\ f_2(t, \mathbf{y}) & \text{otherwise} \end{cases}.$$

Legend

- ○ Minor step state x
- ● Major step in X
- Search process
- Zc value pair
- First trial step from Tn-1 to tn
- Integration results

Zero-crossing event detection

Main steps

- **Detection** of zero-crossing event
  Is one of the zero-crossing changed its sign between $[t_n, t_n + h_n]$?
- **Localization**: if detection is true
  Bracket the most recent zero-crossing time using bisection method.
- **Pass through** the zero-crossing event in two steps:
  - Set the next major output to the left bound of the bracket time.
  - Reset the solver with the state estimate at the right bound of bracket time.

Ingredients for zero-crossing events – 1

Detection of the event.
We check that

$$g(t_n, \mathbf{y}_n) \cdot g(t_{n+1}, \mathbf{y}_{n+1}) < 0$$

We observe is there is a sign changement of the zero-crossing function g.

Remark this is a not robust method (is the sign changes twice for example)

Zero-crossing event detection

Main steps

- **Detection** of zero-crossing event
  Is one of the zero-crossing changed its sign between $[t_n, t_n + h_n]$?
- **Localization**: if detection is true
  Bracket the most recent zero-crossing time using bisection method.
- **Pass through** the zero-crossing event in two steps:
  - Set the next major output to the left bound of the bracket time.
  - Reset the solver with the state estimate at the right bound of bracket time.

Ingredients for zero-crossing events – 2

Continuous extension (method dependent) to easily estimate state.
For example, ode23 uses Hermite interpolation

$$p(t) = (2\tau^3 - 3\tau^2 + 1)\mathbf{y}_n + (\tau^3 - 2\tau^2 + \tau)(t_2 - t_1)f(\mathbf{y}_n) + (-2\tau^3 + 3\tau^2)\mathbf{y}_{n+1} + (\tau^3 - \tau^2)(t_2 - t_1)f(\mathbf{y}_{n+1})$$

with $\tau = \frac{t - t_n}{h_n}$

Zero-crossing event detection

Main steps

- **Detection** of zero-crossing event
  Is one of the zero-crossing changed its sign between $[t_n, t_n + h_n]$?
- **Localization**: if detection is true
  Bracket the most recent zero-crossing time using bisection method.
- **Pass through** the zero-crossing event in two steps:
  - Set the next major output to the left bound of the bracket time.
  - Reset the solver with the state estimate at the right bound of bracket time.

Ingredients for zero-crossing events – 2

The solve the equation

$$g(t, p(t)) = 0$$

instead of $g(t, y(t)) = 0$

Note: as this equation is 1D then algorithm as bisection or Brent's method can be used instead of Newton's iteration.

Main steps

- **Detection** of zero-crossing event
  Is one of the zero-crossing changed its sign between $[t_n, t_n + h_n]$?
- **Localization**: if detection is true
  Bracket the most recent zero-crossing time using bisection method.
- **Pass through** the zero-crossing event in two steps:
  - Set the next major output to the left bound of the bracket time.
  - Reset the solver with the state estimate at the right bound of bracket time.

Ingredients for zero-crossing events – 3

Enclosing the time of event produce a time interval $[t^-, t^+]$ for which we have

- the left limit of the solution $\mathbf{y}(t^-)$
- an approximation of the right limit of the solution $\mathbf{y}(t^+)$ which is used as initial condition for the second dynamics

Simulation algorithm

Data: f_1 the dynamic, f_2 the dynamic, g the zero-crossing function, $\mathbf{y}_0$ initial condition, t_0 starting time, t_{end} end time, h integration step-size, tol

```
t ← t_0;
y ← y_0;
f ← f_1;
while t < t_end do
    Print(t, y);
    y_1 ← Euler(f,t,y,h);
    y_2 ← Heun(f,t,y,h);
    if ComputeError(y_1, y_2) is smaller than tol then
        if g(y) · g(y_1) < 0 then
            Compute p(t) from y, f(y), y_1 and f(y_1);
            [t^-, t^+] = FindZero (g(p(t)));
            Print (t + t^-, p(t^-));
            f ← f_2;
            y ← p(t^+);
            t ← t + t^+;
        end
        y ← y_1;
        t ← t + h;
        h ← ComputeNewH (h, y_1, y_2);
    end
    h ← h/2
end
```

Remark

One-step methods are more robust than multi-step in case of discontinuities (starting problem)