Solution of Boundary Value Problems (BVP)

Moritz Diehl

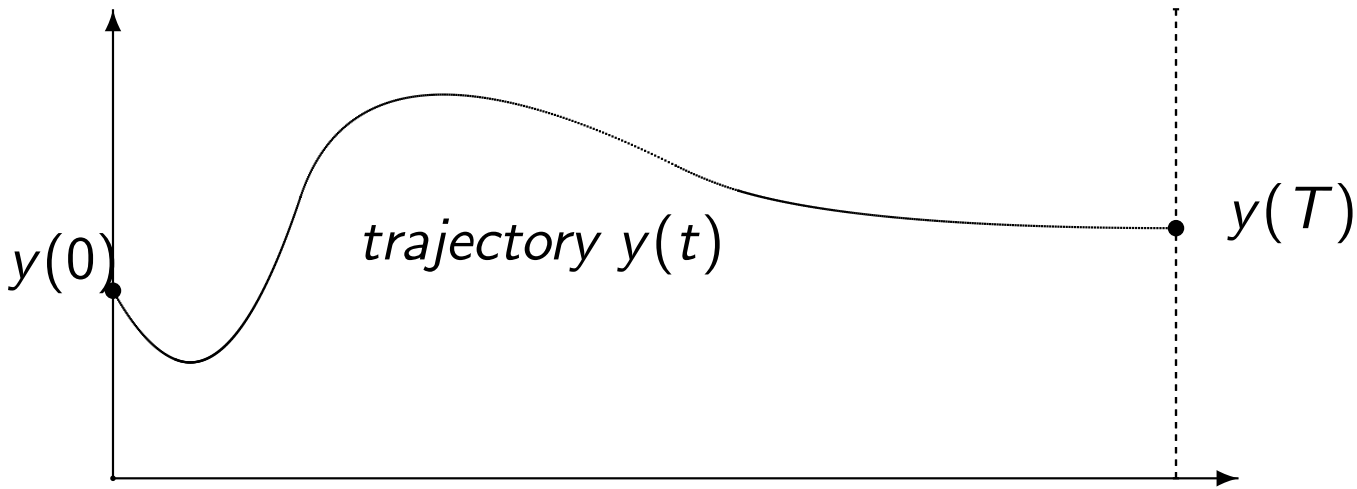
Overview

- Single Shooting
- ODE Sensitivities
- Collocation
- Multiple Shooting

Two Point BVP

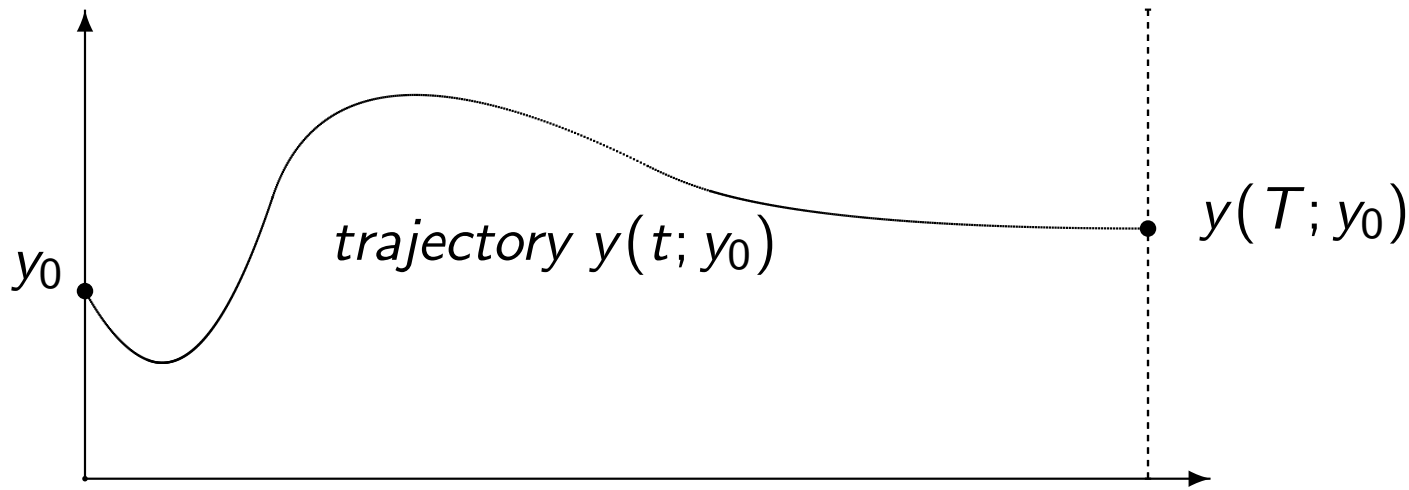
Find trajectory satisfying

$$\begin{aligned} 0 &= r(y(0), y(T)), && && \text{(boundary conditions)} \\ \dot{y}(t) &= f(y(t)) && t \in [0, T], && \text{(ODE model)} \end{aligned}$$



Single Shooting

Guess initial value for y_0. Use numerical integration to obtain trajectory as function $y(t; y_0)$ of y_0.



Obtain in particular terminal value $y(T; y_0)$.

Single Shooting (contd.)

The only remaining equation is

$$\underbrace{r(y_0, y(T; y_0)}_{=F(y_0)} = 0$$

which might or might not be satisfied for the guess y_0.
Fortunately, r has as many components as y_0, so we can apply Newton's method for root finding of

$$F(y_0) = 0$$

which iterates

$$y_0^{k+1} = y_0^k - \left(\frac{\partial F}{\partial y_0}(y_0^k)\right)^{-1} F(y_0^k)$$

Attention: to evaluate $\frac{\partial F}{\partial y_0}(y_0^k) = \frac{\partial r}{\partial y_0} + \frac{\partial r}{\partial y(T)} \frac{\partial y(T;y_0)}{\partial y_0}$ we have to compute ODE sensitivities.

ODE Sensitivities

How to compute the sensitivity

$$\frac{\partial y(T; y_0)}{\partial y_0}$$

of a numerical ODE solution $y(T; y_0)$ with respect to the initial value y_0? Four ways:

- External Numerical Differentiation (END)
- Variational Differential Equations
- Automatic Differentiation
- Internal Numerical Differentiation (IND)

External Numerical Differentiation

Perturb y_0 and call integrator several times to compute derivatives by finite differences:

$$\frac{y(T; y_0 + \epsilon e_i) - y(T; y_0)}{\epsilon}$$

Very easy to implement, but several problems:

- Relatively expensive, have overhead of error control for each varied trajectory.
- Due to adaptivity, each call might have different discretization grids: output $y(T; y_0)$ is not differentiable!
- How to chose perturbation stepsize? Rule of thumb: $\epsilon = \sqrt{\mathrm{TOL}}$ if TOL is integrator tolerance.
- Looses half the digits of accuracy. If integrator accuracy has (typical) value of $\mathrm{TOL} = 10^{-4}$, derivative has only two valid digits!

Variational Differential Equations

Solve additional matrix differential equation

$$\dot{G} = \frac{\partial f}{\partial y}(y)G, \quad G(0) = \mathbb{I}$$

Very accurate at reasonable costs, but:

- Have to obtain explicit expression for $\frac{\partial f}{\partial y}(y)$.
- Computed sensitivity is not 100 % identical with derivative of (discretized) integrator result $y(T; y_0)$.

Automatic Differentiation

Treat integration routine by Automatic Differentiation (AD), i.e. differentiate each step of the integration scheme. For illustration, regard Euler integrator (never used in practice!), which gives, when differentiated:

$$G(t_k + h) = G(t_k) + h\frac{\partial f}{\partial y}(y(t_k))G(t_k), \quad G(0) = \mathbb{I}$$

Very accurate, and up to machine precision 100 % identical with derivative of numerical solution $y(T; y_0)$, but:

- Have to obtain explicit expression for $\frac{\partial f}{\partial y}(y)$
- For Automatic Differentiation, need integrator and right hand side ($f(y)$) be written in same or compatible computer languages (e.g. C++ when using ADOL-C)

Internal Numerical Differentiation (IND)

Differentiate each step of the integration scheme numerically, or evaluate **simultaneously** all perturbed trajectories y_i. Like External Numerical Differentiation, but with **frozen** discretization grid and fixing also all other adaptivities. For illustration, regard Euler integrator (never used in practice!):

$$y_i(t_k + h) = y_i(t_k) + hf(y_i(t_k)), \quad y_i(0) = y_0 + \epsilon e_i$$

Very efficient, easy to use, and up to cancellation and linearization errors identical with derivative of numerical solution $y(T; y_0)$, but:

- How to chose perturbation stepsize? Rule of thumb: $\epsilon = \sqrt{\mathrm{PREC}}$ if PREC is **machine precision**.

Note: adaptivity of nominal trajectory only, reuse of matrix factorization in implicit methods, so not only more accurate, but also cheaper than END.

Integrator Types

Several types of integrator exist, and most come also in variants that deliver sensitivities:

- Explicit Runge-Kutta-Fehlberg (RKF) Methods, e.g. the famous RKF45 (Order 4 with Stepsize Control based on Order 5) (good for non-stiff systems)
- Implicit Runge-Kutta Methods
- Linear Multistep Methods like the famous Backwards-Differentiation-Formulae (BDF) Methods (good for stiff systems e.g. in chemical engineering) (DAESOL, DDASAC, DASSL, SUNDIALS, ...)
- Extrapolation Methods (LIMEX)
- ...

Models with Switches

If right hand side contains discontinuities, integrator with explicit treatment of switches must be used.
Typical grammar:

$$\dot{y} = \begin{cases} f_1(y) & \text{if} \quad s(y) \geq 0 \\ f_2(y) & \text{if} \quad s(y) < 0 \end{cases}$$

with "switching functions" $s(y)$.
Sensitivity update formulae can be derived, but are very complex.
Few integrators for switches and sensitivities exist.

Overview

- Single Shooting
- ODE Sensitivities
- **Collocation**
- Multiple Shooting

Collocation (Sketch)

- Discretize states on grid with node values $s_i \approx y(t_i)$.
- Replace infinite ODE

$$0 = \dot{y}(t) - f(y(t)), \quad t \in [0, T]$$

by finitely many equality constraints

$$\begin{aligned} c_i(s_i, s_{i+1}) &= 0, \quad i = 0, \ldots, N-1, \\ \text{e.g.} \quad c_i(s_i, s_{i+1}) &:= \frac{s_{i+1}-s_i}{t_{i+1}-t_i} - f\left(\frac{s_i+s_{i+1}}{2}\right) \end{aligned}$$

Higher Order Collocation

Typically have intermediate grid points, e.g. $M = 2, 3$ or 4 per subinterval. Denote s^0 as initial value at start time t^0 of interval. Collocation time points $t^1, \ldots, t^M$ have **unknown** node values $s^1, \ldots, s^M$.

Use interpolation polynomial $p(t; s^0, \ldots, s^M)$ of degree M satisfying

$$p(t^i; s^0, \ldots, s^M) = s^i, \quad i = 0, \ldots, M.$$

Determine node values uniquely by derivative conditions

$$\frac{\partial p}{\partial t}(t^i; s^0, \ldots, s^M) = f(s^i), \quad i = 1, \ldots, M$$

Can achieve high order by chosing t^i e.g. as Gauss-Integration points. Similar to implicit Runge-Kutta Integrators.

Couple start and end points of consecutive intervals, i.e. $s_k^M = s_{k+1}^0$.

Nonlinear Equation in Collocation

After discretization, obtain large scale, but sparse nonlinear equation system:

$$\begin{aligned} r(s_0, s_N) &= 0, && && \text{(boundary conditions)} \\ c_i(s_i, s_{i+1}) &= 0, && i = 0, \ldots, N-1, && \text{(discretized ODE)} \end{aligned}$$

Solve with Newton's method. Exploit sparsity in linear system setup and solution.

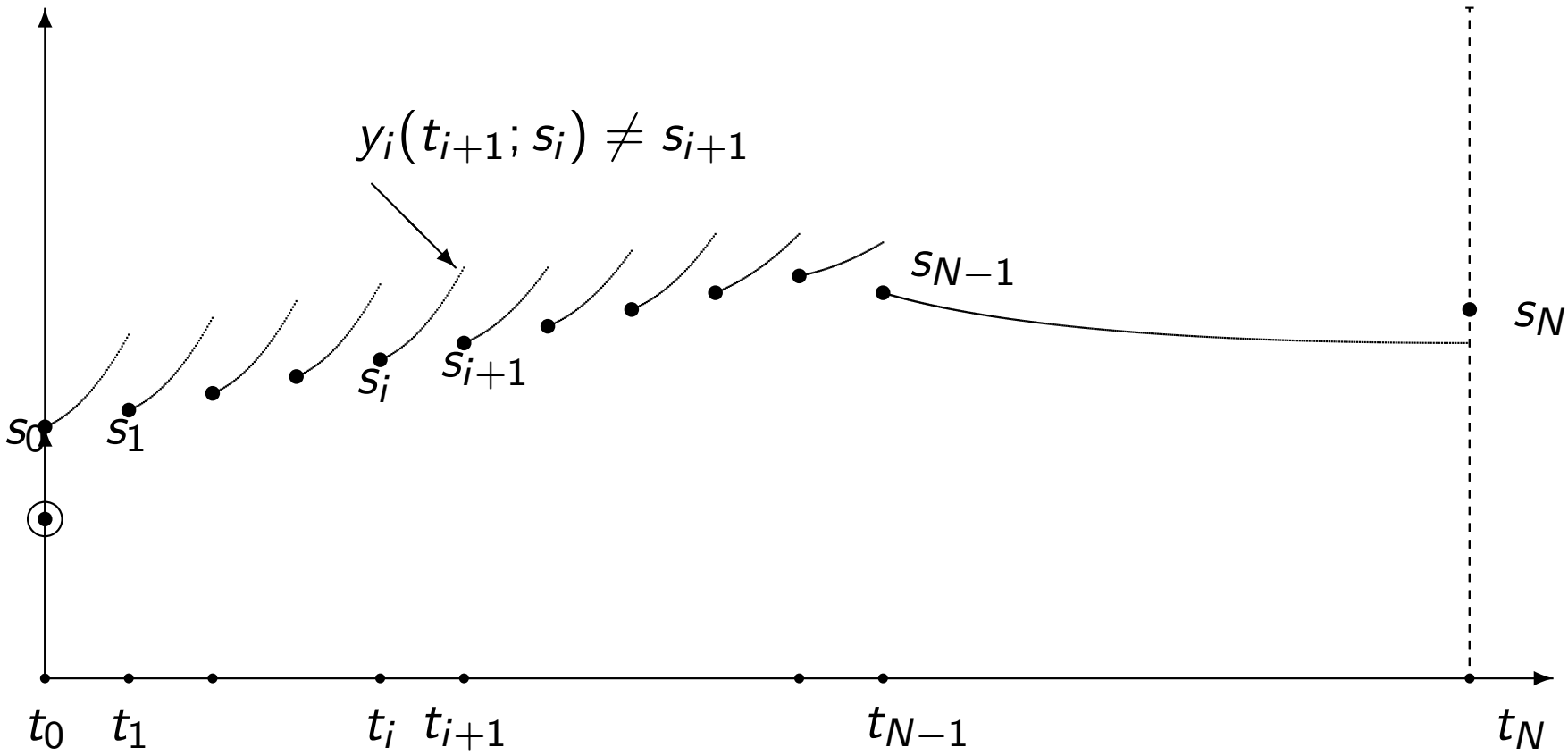
Multiple Shooting for BVPs

- Divide time horizon into intervals
- Solve ODE on each interval $[t_i, t_{i+1}]$ numerically, starting with artificial initial value s_i:

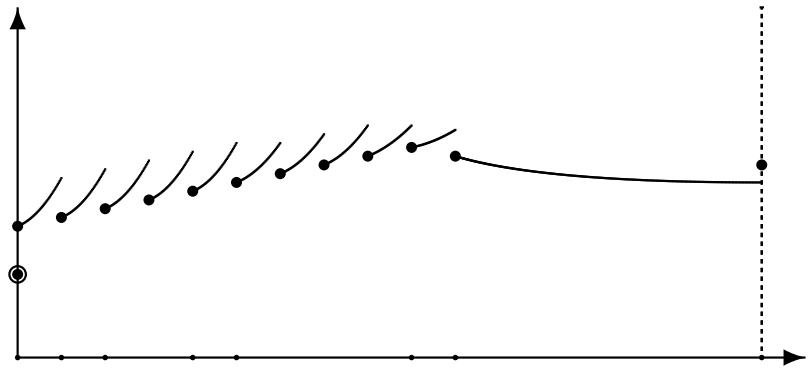
$$\begin{aligned} \dot{y}_i(t; s_i) &= f(y_i(t; s_i)), \quad t \in [t_i, t_{i+1}], \\ y_i(t_i; s_i) &= s_i. \end{aligned}$$

Obtain trajectory pieces $y_i(t; s_i)$.

Sketch of Multiple Shooting



Nonlinear Equation in Multiple Shooting



$$
\begin{aligned}
r(s_0, s_N) &= 0, && && \text{(boundary conditions)} \\
s_{i+1} - y_i(t_{i+1}; s_i) &= 0, && i = 0, \ldots, N-1, && \text{(continuity conditions)}
\end{aligned}
$$

Summarize all variables as $w = (s_0, \ldots, s_N)$, and nonlinear equations as

$$F(w) = 0$$

Structured Jacobian

Jacobian of this system is block sparse:

$$\frac{\partial F}{\partial w} = \begin{bmatrix} R_0 & & & & & & R_N \\ -A_0 & \mathbb{I} & & & & & \\ & -A_1 & \mathbb{I} & & & & \\ & & -A_2 & \mathbb{I} & & & \\ & & & & \ddots & & \\ & & & & & -A_{N-1} & \mathbb{I} \end{bmatrix}$$

Can exploit this in numerical solution procedure for Newton step

$$\frac{\partial F}{\partial w} \Delta w = -F(w)$$

Linearization = Linear Discrete Time System

For computation of Newton step

$$\Delta w = (\Delta s_0, \ldots, \Delta s_N)$$

via

$$\frac{\partial F}{\partial w} \Delta w = -F(w)$$

the linearized continuity conditions represent **linear discrete time system:**

$$\Delta s_{i+1} = (y_i(t_{i+1}; s_i) - s_{i+1}) + A_i \Delta s_i, \quad i = 0, \ldots, N-1.$$

Condensing

Can eliminate all $\Delta s_1, \ldots, \Delta s_N$ as function of Δs_0, by a vector and matrix recursion:

$$b_0 = 0, \quad b_{i+1} = (y_i(t_{i+1}; s_i) - s_{i+1}) + A_i b_i, \quad i = 0, \ldots, N-1.$$

$$G_0 = \mathbb{I}, \quad G_{i+1} = A_i G_i, \quad i = 0, \ldots, N-1.$$

to obtain

$$\Delta s_i = b_i + G_i \Delta s_0 \quad i = 0, \ldots, N.$$

This technique of eliminating the states is called "condensing".

Small Condensed Linear System

The linearized boundary equation was

$$R_0 \Delta s_0 + R_N \Delta s_N = -r(s_0, s_N).$$

In condensed form we obtain

$$(R_0 + R_N G_N)\Delta s_0 = -r(s_0, s_N) - R_N b_N.$$

This has exactly the same dimensions as before in single shooting!
Thus, have nearly same costs per iteration....

Why multiple shooting?

- More freedom in initialization.
- Avoid that a well posed BVP inherits bad conditioning of initial value problem (IVP). (example: unstable system with fixed terminal condition)
- Have faster Newton convergence even for single shooting initialization (example: $x^{16} - 2 = 0$).
- Can solve linear system with other approaches than condensing for even better numerical stability (e.g. structure preserving QR factorization of jacobian)
- in contrast to collocation, can use adaptive integrators

Summary

- Three numerical methods for solution of boundary value problems:
  - single shooting
  - collocation
  - multiple shooting
- shooting methods need ODE integrators with sensitivities
- Four methods to obtain ODE sensitivities:
  - External Numerical Differentiation,
  - Internal Numerical Differentiation,
  - Variational Differential Equations
  - Automatic Differentiation

References

- M.R. Osborne: On shooting methods for boundary value problems. Journal of Mathematical Analysis and Applications, Vol. 27, pp. 417–433, 1969.
- U. Ascher, B. Mattheij and B. Russell: Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, SIAM Classics, 1995.
- J. Albersmeyer and M. Diehl: The Lifted Newton Method and its Application in Optimization, SIAM J. Optim. Vol. 20, No. 3, pp. 1655-1684, 2010.