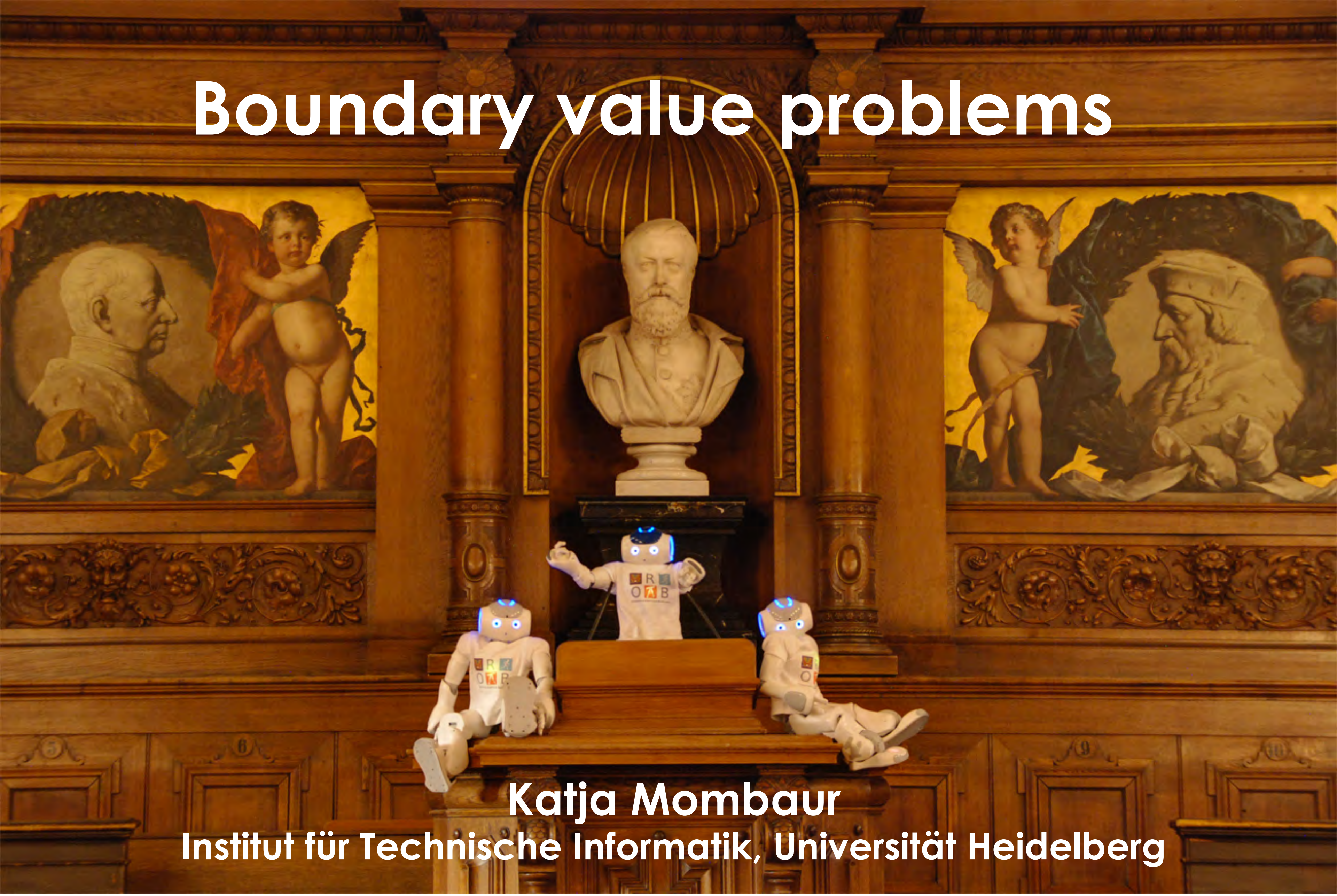


Boundary value problems

Katja Mombaur
Institut für Technische Informatik, Universität Heidelberg

Robotics 2 - 27 May 2019

Boundary value problems (BVP) – standard forms

- **2 point BVP**

$$\boxed{\begin{aligned} \dot{x} &= f(t,x) \\ r(x(t_0), x(t_{end})) &= 0 \end{aligned}} \qquad (1)$$

- **Multipoint BVP:**

$$\boxed{\begin{aligned} \dot{x} &= f(t,x) \\ r(x(t_0), x(t_1), \ldots, x(t_{end})) &= 0 \end{aligned}} \qquad (2)$$

- Other special types of boundary conditions
  - **Linear 2 point boundary condition** $$Ax(t_0) + Bx(t_{end}) = c_1$$
  - **Linear separated boundary cond.** $$A_i x(t_i) = c_i \quad i = 1, .., k$$
  - **General separated boundary cond.** $$r(x(t_i)) = c_i \quad i = 1, .., k$$

Solution methods for boundary value problems

Katja Mombaur

Solution method for BVP: Single Shooting

- We consider again the 2 point BVP:

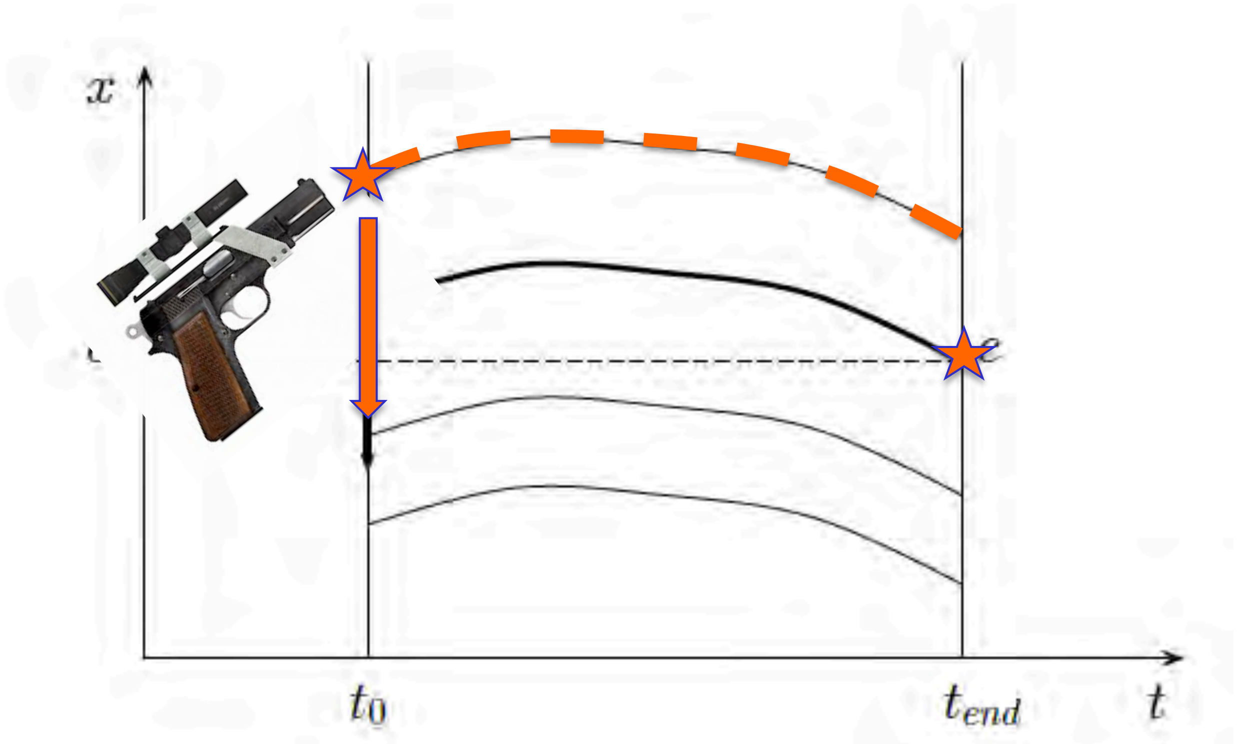
$$\begin{aligned} \dot{x} &= f(t, x(t)) \qquad f \in C^2 \\ r(x(t_0), x(t_{end})) &= 0 \end{aligned}$$

- Basic idea dea of Shooting methods: Trace BVP back to underlying IVP
- Hence the name „initial value methods for the solution of BVP“ for shooting methods
- Iterative determination of initial values such that the solution of the BVP exists

Single Shooting

- A simple example with one variable and one end point condition

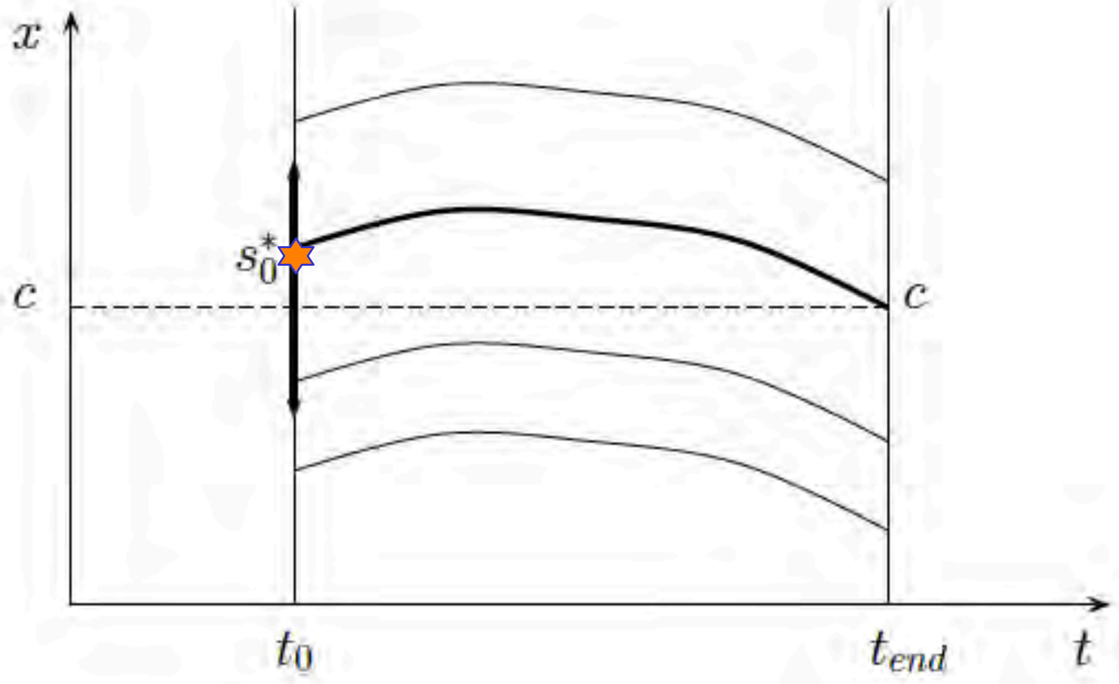
$$\begin{aligned} \dot{x}_1(t) &= f(t, x_1(t)) \\ r &= x(t_{end}) = c \end{aligned}$$



Single Shooting

- A simple example with one variable and one end condition

$$\begin{aligned} \dot{x}_1(t) &= f(t, x_1(t)) \\ r &= x(t_{end}) = c \end{aligned}$$



Single Shooting Algorithm

1. Select initial value (vector) s_0
2. Solve Initial value problem

$$\begin{aligned} \dot{x}(t) &= f(t,x) \\ x(t_0) &= s_0 \end{aligned}$$

$\Rightarrow$ Solution $\bar{x}(t;t_0,s_0)$

1. Is boundary condition satisfied?

$$r(s_0, x(t_{end};t_0,s_0)) = 0`$$

yes? $\Rightarrow$ **Stop**

2. Otherwise: Solve nonlinear system of equations numerically:

$$F(s_0) = r(s_0, x(t_{end};t_0,s_0)) = 0$$

Method of choice?

Newton's method

Numerical solution of nonlinear equations by Newton's method

- Iterative solution

$$F(s_{0[k+1]}) = F(s_{0[k]}) + \nabla F(s_{0[k]})\,\Delta s_{0[k]} \overset{!}{=} 0$$

$$\Delta s_{0[k]} = -\nabla F(s_{0[k]})^{-1} F(s_{0[k]})$$

$$s_{0[k+1]} \quad = \quad s_{0[k]} + \Delta s_{0[k]}$$

- Every step of this iteration requires a new
  - integration
  - evaluation of boundary conditions
  - computation of derivatives

Computation of ∇F

$$\nabla F(s_{0[k]}) = \boxed{\frac{\partial r}{\partial x_0}(s_{0[k]}, x(t_{end}; t_0, s_{0[k]}))} + \boxed{\frac{\partial r}{\partial x_{end}}(s_{0[k]}, x(t_{end}; t_0, s_{0[k]}))} \cdot \boxed{\frac{\partial x(t_{end}; t_0, s_{0[k]})}{\partial x_0}}$$

This requires:

- Partial derivatives of boundary conditions with respect to initial and final values (in many cases analytical derivatives, see example)
- Computation of sensitivity matrix of end values of trajectory with respect to initial values, i.e. derivatives of integrated trajectory (more complex)

Example:

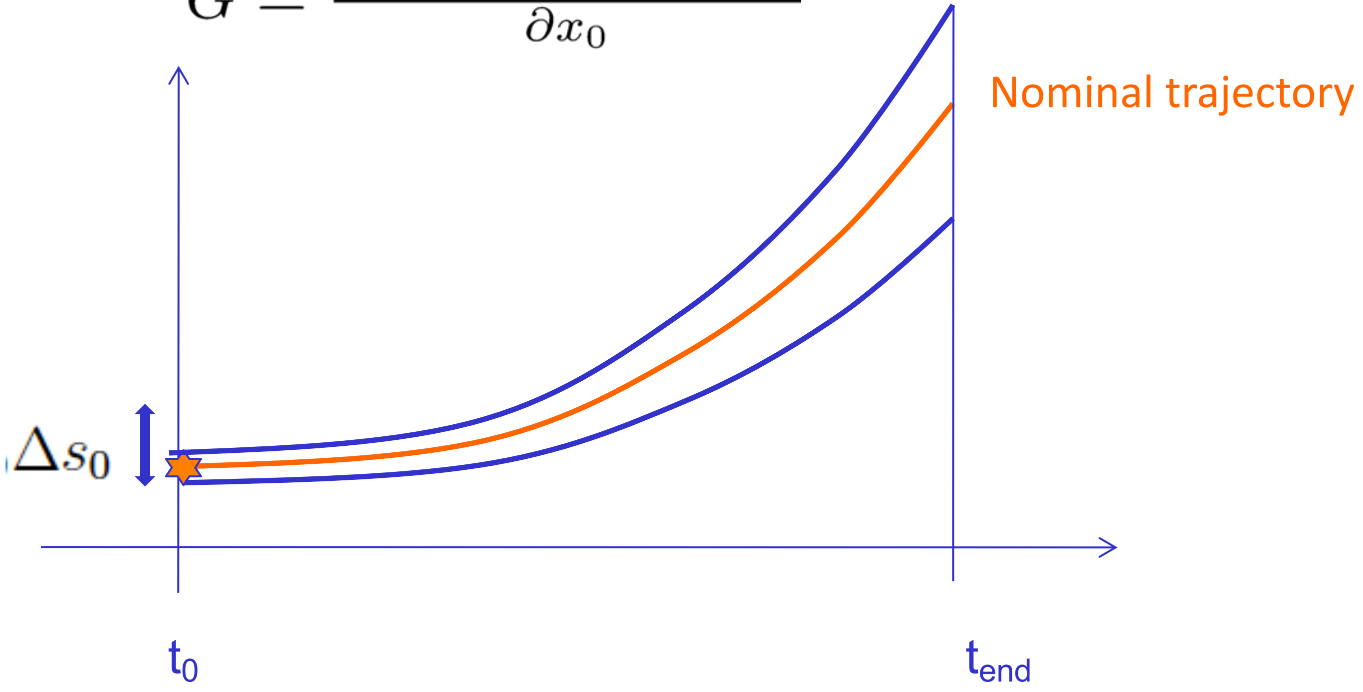
$$F = r(x_0, x_e) = x_0 - x_e = 0$$

$$\nabla F(s_0[k]) = \boxed{1} + \boxed{(-1)} \cdot \boxed{\frac{\partial x(t_{end}; t_{0,[k]}, s_{0,[k]})}{\partial x_0}}$$

Sensitivities of trajectories

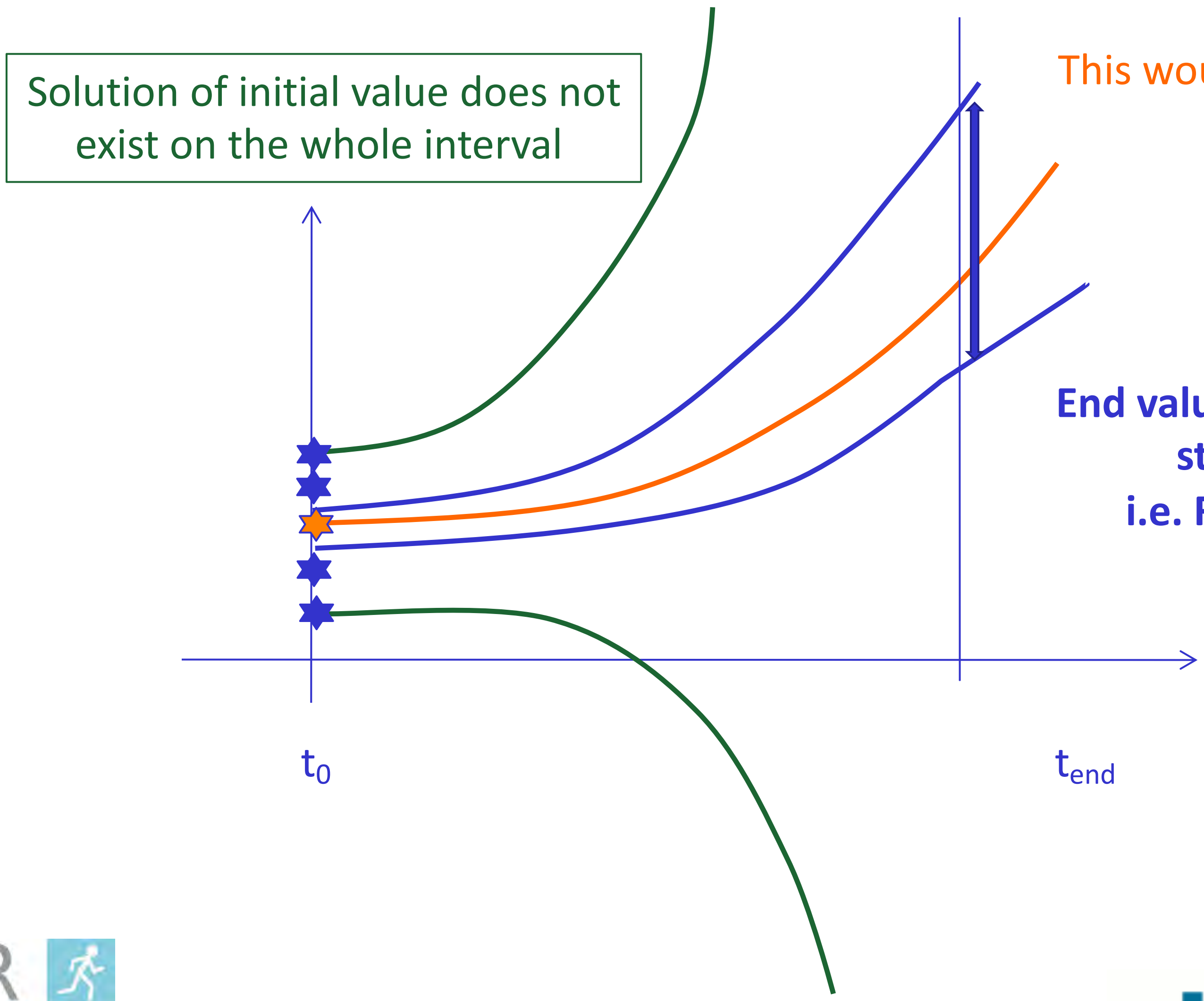
- How do end values of integration change if initial values change?

$$G = \frac{\partial x(t_{end}; t_{0,[k]}, s_{0,[k]})}{\partial x_0}$$



How this is done will be discussed later today or next week

Single Shooting – what might happen?



Solution of initial value does not exist on the whole interval
t_0
t_{end}

This would be the solution of the BVP

End values and sensititivies can still be computed, i.e. F and VF can still be evaluated

Advantages and disadvantages of Single Shooting

- **Advantages:**
  - Simple concept
  - Simple implementation possible; state of the art components can be used for solution of initial value problems and of nonlinear equations

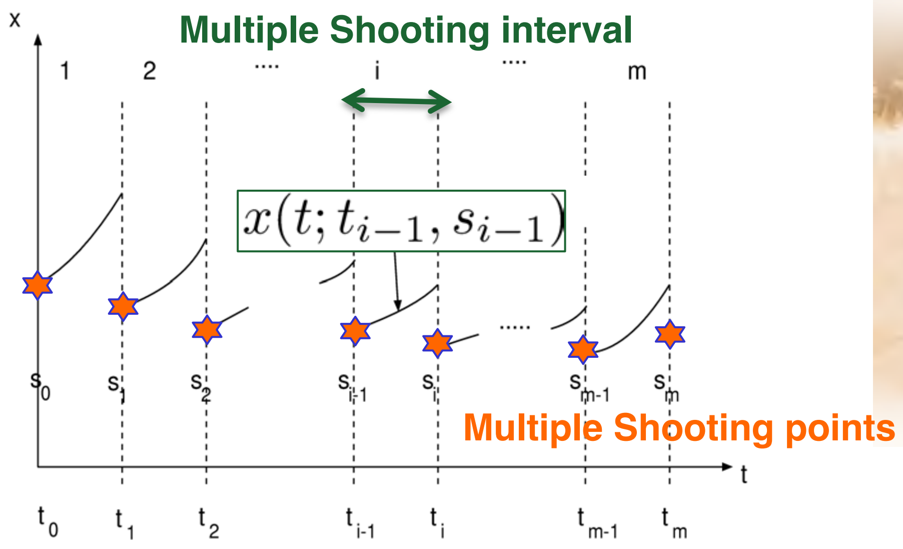
- **Disadvantages:**
  - Even for well conditioned BVD, the corresponding IVP can be unstable
  - For bad initial values there is no guarantee that the solution of the IVP exists on the whole interval [t_0, t_{end}], i.e. (F(s0) may be only defined small neighborhood of s*)
  - For bad starting data, Newton‘s method does not converge.

From single shooting to multiple shooting

?

Multiple Shooting

- Addresses the disadvantages of Single Shooting which result from the long integration time



Idea:

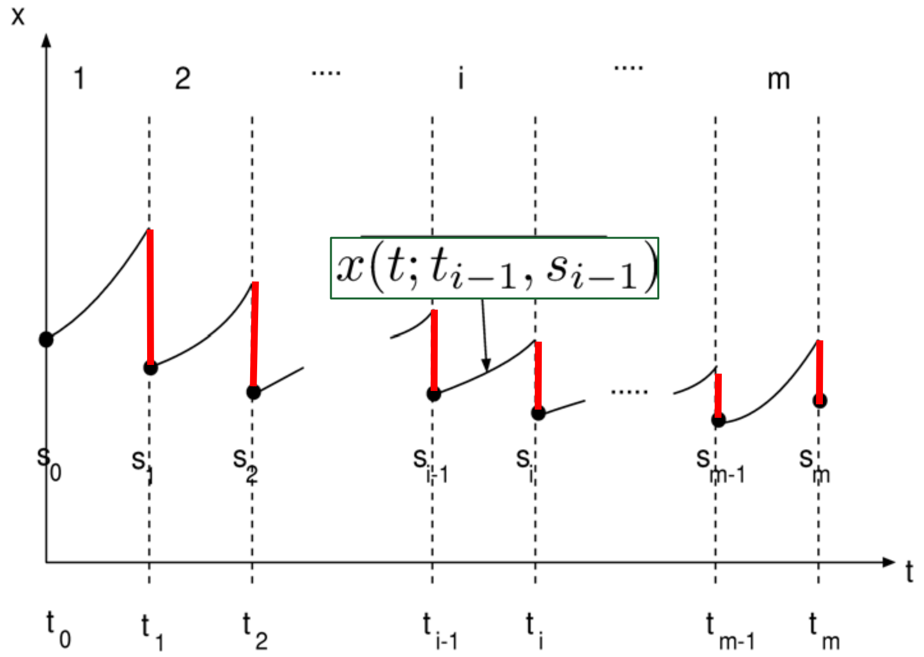
- Split the interval $[t_0, t_{end}]$ into m subintervals (= multiple shooting intervals)

$$[t_i, t_{i+1}],\ i = 0, \ldots, m-1, \qquad t_0 < t_1 < \cdots < t_m = t_{end}$$

- Instead of s_0 only, introduce parameterized initial values for all intervals $s_i \in \mathbb{R}^n$ at the so called multiple shooting points.

- Independent integrations („shots") are performed from all these points

Multiple Shooting



- Integration and sensitivity generation are performed seperately on each interval
- In order to guarantee equivalence to the original problem, continuitiy conditions have to be satisfied at the interval borders

$$x(t_{i+1}; t_i, s_i) - s_{i+1} = 0, \quad i = 0, \ldots, m-1$$

Multiple Shooting – Variables and constraints

- Compared to Single Shooting
  - n variables $s_0 \in \mathbb{R}^n$
  - n nonlinear equations $F(s_0) = r(s_0, x(t_{end}; t_0, s_0)) = 0$

- Multiple Shooting requires $s_i \in \mathbb{R}^n \quad i = 1, \ldots, m$
  - $n \cdot m$ **additional** variables
  - $n \cdot m$ **additional** constraints (the continuity conditions)

- The problem size has been considerably increased to $n \cdot (m+1)$
  But we will see later that suitable numerical techniques avoid a big increase in computational complexity.

Multiple Shooting algorithm

- In analogy to Single Shooting, but
  - Integration of IVP + Sensitivity generation on all multiple shooting intervals i, starting from s_i . The integrations are completely independent, and can be easily parallelized.
  - Solution of Multiple Shooting system of equations:

$$F(s) = \begin{pmatrix} \underbrace{r(s_0, s_m)}_{\text{boundary conditions}} \\ x(t_1; t_0, s_0) - s_1 \\ x(t_2; t_1, s_1) - s_2 \\ \vdots \\ x(t_m; t_{m-1}, s_{m-1}) - s_m \end{pmatrix} = 0, \quad \text{with } s = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_m \end{pmatrix} \in \mathbb{R}^{n(m+1)}$$

boundary conditions

continuity conditions

nonlinear system of equations with $n \cdot (m+1)$ equations and unknowns

Multiple Shooting algorithm

- This system can also be solved by Newton's method

- Newton iteration $s_{[k+1]} = s_{[k]} + \Delta s_{[k]}$

  where $\Delta s_{[k]}$ is a solution of the following system of equations:

$$\nabla F(s_{[k]} \cdot \Delta s_{[k]} = -F(s_{[k]})$$

with the Jacobian

$$\nabla F(s) = \begin{pmatrix} A & 0 & \cdots & \cdots & 0 & B \\ G_0 & -I & 0 & \cdots & 0 & 0 \\ 0 & G_1 & -I & & \vdots & 0 \\ \vdots & & \ddots & \ddots & \vdots & \vdots \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & G_{m-1} & -I \end{pmatrix}$$

$$A = \frac{\partial r(s_0, s_m)}{\partial s_0}$$

$$B = \frac{\partial r(s_0, s_m)}{\partial s_m}$$

$$G_i = \frac{\partial x(t_{i+1}; t_i, s_i)}{\partial s_i}$$

Excursion: Solution of linear systems of equations

- For the solution of linear systems of equations there are many standard direct methods, such as Gauss elimination (= LR decomposition) which are implemented in many libraries

- But the numerical cost for such algorithms is quite high, e.g in the case of Gauss elimination $\sim \frac{2}{3} z^3$ flops (z: dimension of system)

  i.e. for the multiple shooting system it would be $\sim \frac{2}{3} (n(m+1))^3$ flops

  and therefore much higher than for the single shooting system: $\sim \frac{2}{3} n^3$ flops

- BUT: the **Jacobian** ∇F **has got a special structure** (= many zeroes at exactly known places), which can be exploited for the solution of the linear system

$\Rightarrow$ **Condensing**

Condensing (Block Gauss Decomposition)

- Goal: efficient structure expoiting decomposition of the nonlinear system of equations

$$\nabla F(s) \cdot \Delta s = -F(s)$$

$$\begin{pmatrix} A & 0 & 0 & \cdots & 0 & B \\ G_0 & -I & 0 & \cdots & \cdots & 0 \\ 0 & G_1 & -I & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & 0 \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & G_{m-1} & -I \end{pmatrix} \Delta s = - \begin{pmatrix} r \\ h_0 \\ h_1 \\ \vdots \\ \vdots \\ h_{m-1} \end{pmatrix}$$

$$r = r(s_0, s_m), \; h_i := x(t_{i+1}; t_i, s_i) - s_{i+1} \quad \text{und} \quad \Delta s^T = (\Delta s_0^T, \ldots, \Delta s_m^T)$$

Condensing algorithm - 1

- Multiply the last block row with B and add it to the first block row

$$\begin{pmatrix} A & 0 & \cdots & 0 & BG_{m-1} & 0 \\ G_0 & -I & 0 & \cdots & \cdots & 0 \\ 0 & G_1 & -I & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & 0 \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & G_{m-1} & -I \end{pmatrix} \Delta s = - \begin{pmatrix} r + Bh_{m-1} \\ h_0 \\ h_1 \\ \vdots \\ \vdots \\ h_{m-1} \end{pmatrix}$$

- Multiply the second last block row with BG_{m-1} nd add it to the first block row

$$\begin{pmatrix} A & 0 & 0 & BG_{m-1}G_{m-2} & 0 & 0 \\ G_0 & -I & 0 & \cdots & \cdots & 0 \\ 0 & G_1 & -I & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & 0 \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & G_{m-1} & -I \end{pmatrix} \Delta s = - \begin{pmatrix} r + Bh_{m-1} + BG_{m-1}h_{m-2} \\ h_0 \\ h_1 \\ \vdots \\ \vdots \\ h_{m-1} \end{pmatrix}$$

Condensing algorithm - 2

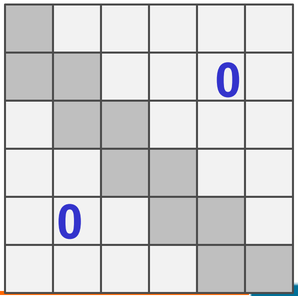
- ... etc., until only the first block in the first row has non zero entries

$$\begin{pmatrix} E & 0 & \cdots & \cdots & \cdots & 0 \\ G_0 & -I & 0 & \cdots & \cdots & 0 \\ 0 & G_1 & -I & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & & 0 \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & G_{m-1} & -I \end{pmatrix} \Delta s = - \begin{pmatrix} u \\ h_0 \\ h_1 \\ \vdots \\ \vdots \\ h_{m-1} \end{pmatrix}$$

$$E := A + BG_{m-1} \cdots G_0$$

$$u := r + Bh_{m-1} + BG_{m-1}h_{m-2} + \cdots + BG_{m-1} \cdots G_1 h_0$$

- This system has got band structure / a block diagonal shape



Condensing algorithm - 3

- Solve the first block row („the condensed system“)

$$E\Delta s_0 = -u$$

dense system of dimension n

- .. then the second block row ...

$$G_0\Delta s_0 - I\Delta s_1 = -h_0$$

$$\Delta s_1 = G_0\Delta s_0 + h_0$$

- ... and the remaining block rows in the same way :

$$\vdots$$

$$\Delta s_m = G_{m-1}\Delta s_{m-1} + h_{m-1}$$

Formulation as recursive algorithm

1. Backward recursion
   a) Set $E := B, u = r$
   b) for $i = m, \ldots, 1$:
   $$\begin{aligned} u &= u + EH_{i-1} \\ E &= EG_{i-1} \end{aligned}$$
   c) $E = E + A$

2. Solution of condensed system
   $$E\,\Delta s_0 = -u \quad \Rightarrow \quad \Delta s_0$$

3. Forward recursion

   for $i = 1, \ldots, m$:
   $$\Delta s_i = G_{i-1}\,\Delta s_{i-1} + h_{i-1}$$

Some remarks

- If along the exact solution of a BVP the matrix

$$E^* = A + BG(T_{end}; t_0)|_{x^*}$$

  is regular, then this property also holds in a neighborhood of the solution for

$$E = A + BG_{m-1} \cdots G_0$$

  and the linear system in step 2 (see previous slide) can be solved.

- **Computational cost is much lower than for solving uncondensed system**

- **Condensing can be generalized to multipoint BVPs**

Multiple Shooting – Advantages & Disadvantages

- Advantages:
  - Adaptive ODE solvers can be used for integration (error control of dynamic process, does not modify multiple shooting grid)
  - can use knowledge about x in initialization along all multiple shooting points, not only guess for first point (as in single shooting)
  - Can treat unstable systems well
  - Easy to parallelize

- Disadvantages:
  - Integration of dynamic processes is costly
    (this is not a disadvantage with respect to single shooting where also an integration is required, but with respect to collocation that comes next)

A different method for the solution of BVP: Collocation

$$\begin{aligned} \dot{x} &= f(t, x(t)) \\ r(x(t_0), x(t_{end})) &= 0 \end{aligned}$$

- Basic idea of collocation:
  - Choose approximate solution for BVP from a finite dimensional function space such that the differential equation is exactly satisfied at certain points – the so-called collocation points - but not at others
  - Choice of finite dimensional functions, e.g.
    - Polynomials (increase order for higher accuracy, or better:)
    - Piecewise polynomials (reduce length of pieces for higher accuracy)

$$\Pi_N : \; t_0 < t_1 < \cdots < t_N = t_{end}$$

  - Collocation points on interval N:

$$0 \leq \rho_1 \leq \rho_2 \leq \cdots \leq \rho_k \leq 1 \qquad t_i + \rho_j(t_{i+1} - t_i), \;\; j = 1, \ldots, k$$

A different method for the solution of BVP: Collocation

- Conditions:
  - Collocation points conditions:

$$\dot{x}_\Pi(t_{COL}) = f(t_{COL}, x(t_{COL})) \qquad N \cdot k \cdot n \quad \text{equations}$$

  - Continuity between polynomials

$$x_{\Pi_i}(t_{i+1}) = x_{\Pi_{i+1}}(t_{i+1}) \qquad (n-1) \cdot n \quad \text{equations}$$

  - Boundary conditions

$$n \quad \text{equations}$$

Total: $N \cdot (k+n) \cdot n$ equations

- Large and sparse system, can be solved by sparse solvers

- No integrators are necessary

Collocation – Advantages & Disadvantages

Advantages:

- Large but sparse system, general sparse solvers can be used
- Knowledge of state trajectory can be used in initialization
- Can treat unstable systems well
- No integration required (fast)

Disadvantages

- Adaptivity needs adjustments to grid, change in problem dimensions
- No error control of the whole dynamic process possible (as in the case of integrators) - dynamics are only satisfied exactly in collocation points and not in other places – > solution may be infeasible for a robot.

Thank you very much for your attention!