# Implicit vs. explicit ODE

**Fredrik Magnusson**[1] **Karl Berntorp**[2]

[1]Department of Automatic Control
Lund University, Sweden

[2]Mitsubishi Electric Research Laboratories
Cambridge, MA

November 18, 2015

# Review

Last time:

- Problem does not seem almost high index
  - Even if it did, variable scaling should solve the problem
- Test snow

# Preview

This time:

- Proper absolute tolerances solves everything
- Still some open questions, including optimization

# Absolute tolerance

- Previously used `atol = rtol`
- Model contains decent nominal values `nom` for all variables
- Instead use `atol = 10*nom*rtol`

# Simulation results

Radau5 is reference solution, others have same accuracy

| Setup | tol | Steps [1000] |
|---|---|---|
| Radau5 DAE | 1e-14 | 4.3 |
| IDA DAE | 1e-7 | 2.2 |
| IDA DAE sup. alg. | 1e-8 | 2.0 |
| IDA DAE par. | 1e-7 | 2.0 |
| IDA DAE par. sup. | 1e-8 | 2.2 |
| IDA ODE | 1e-8 | 2.3 |

Conclusion: Algebraic error control was running amok due to unreasonable tolerances for algebraics

# Iteration matrix condition number

Consider

$$\begin{aligned} F(\dot{x}, x, y) &= 0 \\ G(x, y) &= 0 \end{aligned}$$

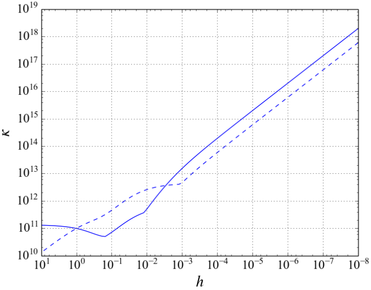
- BDF iteration matrix:

$$J = \begin{bmatrix} \frac{\alpha_0}{h}\nabla_{\dot{x}} F + \nabla_x F & \nabla_y F \\ \nabla_x G & \nabla_y G \end{bmatrix} \tag{1}$$

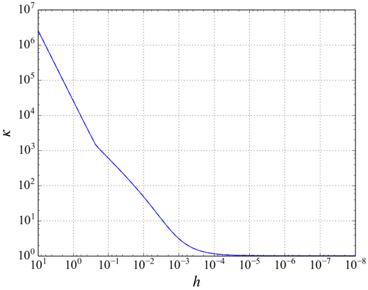
- (Theorem 5.4.1) Condition number of iteration matrix is $\mathcal{O}(h^{-\nu})$
- Conjecture: If $\nabla_{\dot{x}} F = I$ and first row of $J$ is scaled by $h$, condition number is instead $\mathcal{O}(h^{-\nu-1})$
  - What to do if $\nabla_{\dot{x}} F$ is not square? What does IDA do?
  - Maybe I should do something like this in my collocation implementation?

# Condition number

DAE




ODE



# Commutativity

- Previously: Does BLT and fixed-step collocation commute?
- Consider single integration step of

$$\begin{aligned} \dot{x} &= f(x, y, u) \\ y &= g(x, u), \end{aligned} \tag{2}$$

and

$$\dot{x} = f(x, g(x, u), u) \tag{3}$$

# Commutativity

Explicit DAE:

$$\dot{x}_k = f(x_k, y_k, u_k), \tag{4a}$$

$$y_k = g(x_k, u_k), \tag{4b}$$

$$\dot{x}_k = \frac{1}{h} \cdot \sum_{n=0}^{n_c} \alpha_{n,k} x_n, \tag{4c}$$

$$\forall k \in [1..n_c] \tag{4d}$$

Explicit ODE:

$$\dot{x}_k = f(x_k, g(x_k, u_k), u_k), \tag{5a}$$

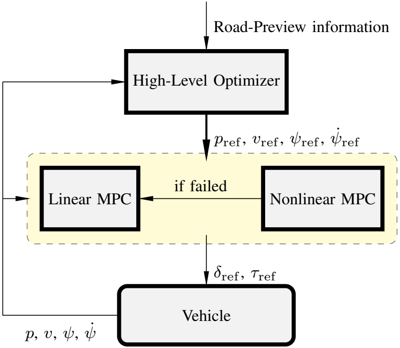
$$\dot{x}_k = \frac{1}{h} \cdot \sum_{n=0}^{n_c} \alpha_{n,k} x_n, \tag{5b}$$

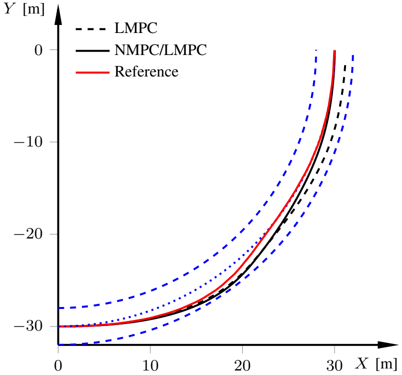
$$\forall k \in [1..n_c] \tag{5c}$$

Clearly commutative?

## MPC

Bilevel control: Single-track used on high level to generate reference trajectory, double-track used on low level to follow trajectory



# Convergence

- High level needs to be solved over free time horizon $\implies$ difficult
- Low level has fixed time horizon (MPC) $\implies$ less difficult
- Newton's method does not converge at all for high-level problem if we use full DAE, but converges for ODE
- Usually converges for low level even with DAE, but ODE more robust and faster

# High level

Will today focus on high-level problem:

$$
\begin{aligned}
\underset{\delta_{\text{ref}},\tau_{f,\text{ref}},\tau_{r,\text{ref}}}{\text{minimize}} \quad & \int_{t_0}^{t_f} (\kappa_1 e^2 + \kappa_2 \beta^2)\,\mathrm{d}t \\
\text{subject to} \quad & |\tau_{i,\text{ref}}| \leq \tau_{i,\max},\ \forall i \in \{f, r\}, \\
& |\delta_{\text{ref}}| \leq \delta_{\max}, \\
& \|p(t_f) - p_{t_f}\| \leq \epsilon, \\
& \Gamma(p) \leq 0, \quad x(t_0) = x_0, \\
& F(\dot{x}, x, y, \delta_{\text{ref}}, \tau_{f,\text{ref}}, \tau_{r,\text{ref}}) = 0
\end{aligned}
\tag{6}
$$

# Dynamic optimization

More abstract/compact formulation ($\boldsymbol{z} := (\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u})$):

$$
\begin{aligned}
&\text{minimize} && \int_{t_0}^{t_f} L(\boldsymbol{z}(t))\,\mathrm{d}t, && \text{(7a)}\\
&\text{with respect to} && \boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^{n_x}, \quad \boldsymbol{y} : [t_0, t_f] \to \mathbb{R}^{n_y}, &&\\
& && \boldsymbol{u} : [t_0, t_f] \to \mathbb{R}^{n_u}, \quad t_f \in \mathbb{R}, &&\\
&\text{subject to} && \boldsymbol{F}(\boldsymbol{z}(t)) = \boldsymbol{0}, \qquad \boldsymbol{F}_0(\boldsymbol{z}(t_0)) = \boldsymbol{0}, && \text{(7b)}\\
& && \boldsymbol{z}_L \le \boldsymbol{z}(t) \le \boldsymbol{z}_U, && \text{(7c)}\\
& && \boldsymbol{g}(\boldsymbol{z}(t)) \le \boldsymbol{0}, \qquad \boldsymbol{G}(\boldsymbol{z}(t_f)) \le \boldsymbol{0}, && \text{(7d)}\\
& && \forall t \in [t_0, t_f]. &&
\end{aligned}
$$

# NLP

Discretize differential equations to get a finite-dimensional nonlinear program (NLP) ($\boldsymbol{z}_{i,k} \approx \boldsymbol{z}(t_{i,k})$, where $t_{i,k}$ is collocation point $k$ in element $i$):

$$\text{minimize} \quad \sum_{i=1}^{n_e} h_i \cdot (t_f - t_0) \cdot \sum_{k=1}^{n_c} \omega_k \cdot L\left(\boldsymbol{z}_{i,k}\right), \tag{8a}$$

$$\text{with respect to} \quad \boldsymbol{z}_{i,k} \in \mathbb{R}^{2n_x+n_y+n_u}, \quad \boldsymbol{x}_{i,0} \in \mathbb{R}^{n_x}, \quad t_f \in \mathbb{R},$$

$$\text{subject to} \quad \boldsymbol{F}(\boldsymbol{z}_{i,k}) = \boldsymbol{0}, \qquad \boldsymbol{F}_0(\boldsymbol{z}_{1,0}) = \boldsymbol{0}, \tag{8b}$$

$$\boldsymbol{u}_{1,0} = \sum_{k=1}^{n_c} \boldsymbol{u}_{1,k} \cdot \ell_k(0) \qquad \boldsymbol{z}_L \leq \boldsymbol{z}_{i,k} \leq \boldsymbol{z}_U, \tag{8c}$$

$$\boldsymbol{g}(\boldsymbol{z}_{i,k}) = \boldsymbol{0}, \qquad \boldsymbol{G}(\boldsymbol{z}_{n_e,n_c}) \leq \boldsymbol{0}, \tag{8d}$$

$$\forall (i,k) \in \{(1,0)\} \cup ([1..n_e] \times [1..n_c]),$$

$$\dot{\boldsymbol{x}}_{j,l} = \frac{1}{h_j \cdot (t_f - t_0)} \cdot \sum_{m=0}^{n_c} \boldsymbol{x}_{j,m} \cdot \frac{\mathrm{d}\tilde{\ell}_m}{\mathrm{d}\tau}(\tau_l), \tag{8e}$$

$$\forall (j,l) \in [1..n_e] \times [1..n_c], \tag{8f}$$

$$\boldsymbol{x}_{n,n_c} = \boldsymbol{x}_{n+1,0}, \quad \forall n \in [1..n_e - 1]. \tag{8g}$$

# NLP solution

After further abstraction, the NLP is:

$$
\begin{aligned}
\text{minimize} \quad & f(x), \\
\text{with respect to} \quad & x \in \mathbb{R}^m, \\
\text{subject to} \quad & x_L \leq x \leq x_U, \\
& g(x) = 0, \\
& h(x) \leq 0.
\end{aligned}
$$

- Solved by IPOPT
- Lots of complicated details, but essentially Newton's method is applied on KKT optimality conditions
- See bonus slides for details

# Convergence

Why convergence issues are more prominent in optimization than simulation in general:

- Larger system of equations (dual variables + TBVP)
- No good initial guess (at least not for high-level problem)
- Inherently ill-conditioned (see bonus slides)

## Restoration phase

- For the high-level problem, IPOPT fails in restoration
- Restoration is triggered for various reasons, but usually because of ill-conditioned Jacobian
- Restoration means that IPOPT stops solving the optimization problem and instead solves

$$
\begin{array}{ll}
\text{minimize} & ||g(x)||_1 + ||h(x) - y||_1 + 0.5\zeta ||D_R(x - x_R)||_2^2, \\
\text{with respect to} & x \in \mathbb{R}^m, \quad y \in \mathbb{R}^n \\
\text{subject to} & x_L \leq x \leq x_U, \\
 & y \leq 0.
\end{array}
$$

- IPOPT finds a local minimum to this problem which is not feasible; failure