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 11, 2015

Review

Last time:

- Problem behaves like high-index problem (sup. alg.)
- Maybe Pacejka's Magic formula is almost high index?

Preview

This time:

- Revisit simulation results, looking at order and step length
- Discuss “almost high index”
  - Results indicate that Magic formula is harmless
  - Results indicate that problem is not almost high index

If time permits:

- Discuss optimization formulation and solution procedure
- Discuss if the observed simulation phenomena regarding DAE vs. ODE are related to optimization

Simulation results

Radau5 tolerance chosen to get reference solution, the rest to get similar accuracy

Setup	tol	Time [s]	Steps [1000]
Radau5 DAE	1e-12	69	17
IDA DAE	1e-6	59	26
IDA DAE sup. alg.	1e-8	2.8	2.9
IDA DAE par.	1e-6	6.3	8
IDA DAE par. sup.	1e-8	1.6	3.2
IDA ODE	1e-8	0.9	3.4

Radau5 DAE

```
Final Run Statistics: ---

 Number of steps                                 : 16701
 Number of function evaluations                  : 231074
 Number of Jacobian evaluations                  : 12338
 Number of function eval. due to Jacobian eval.  : 407154
 Number of error test failures                   : 1040
 Number of LU decompositions                     : 21266

Solver options:

 Solver                     : Radau5 (implicit)
 Tolerances (absolute)      : [  1.00000000e-12]
 Tolerances (relative)      : 1e-12
```

IDA DAE

```
Final Run Statistics: ---

 Number of steps                                 : 26361
 Number of function evaluations                  : 67823
 Number of Jacobian evaluations                  : 25991
 Number of function eval. due to Jacobian eval.  : 857703
 Number of error test failures                   : 12046
 Number of nonlinear iterations                  : 67823
 Number of nonlinear convergence failures        : 0

Solver options:

 Solver                         : IDA (BDF)
 Maximal order                  : 5
 Suppressed algebr. variables   : False
 Tolerances (absolute)          : 1e-06
 Tolerances (relative)          : 1e-06
```

IDA DAE sup. alg.

```
Final Run Statistics: ---

 Number of steps                                 : 2923
 Number of function evaluations                  : 7072
 Number of Jacobian evaluations                  : 1065
 Number of function eval. due to Jacobian eval.  : 35145
 Number of error test failures                   : 424
 Number of nonlinear iterations                  : 7072
 Number of nonlinear convergence failures        : 0

Solver options:

 Solver                         : IDA (BDF)
 Maximal order                  : 5
 Suppressed algebr. variables   : True
 Tolerances (absolute)          : 1e-08
 Tolerances (relative)          : 1e-08
```

IDA DAE par.

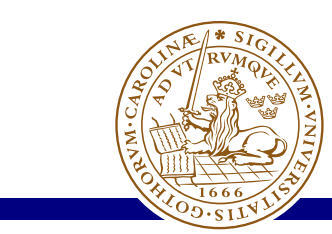
```
Final Run Statistics: ---

 Number of steps                                 : 8489
 Number of function evaluations                  : 18467
 Number of Jacobian evaluations                  : 5952
 Number of function eval. due to Jacobian eval.  : 83328
 Number of error test failures                   : 2867
 Number of nonlinear iterations                  : 18467
 Number of nonlinear convergence failures        : 0

Solver options:

 Solver                         : IDA (BDF)
 Maximal order                  : 5
 Suppressed algebr. variables   : False
 Tolerances (absolute)          : 1e-06
 Tolerances (relative)          : 1e-06
```

IDA DAE par. sup.

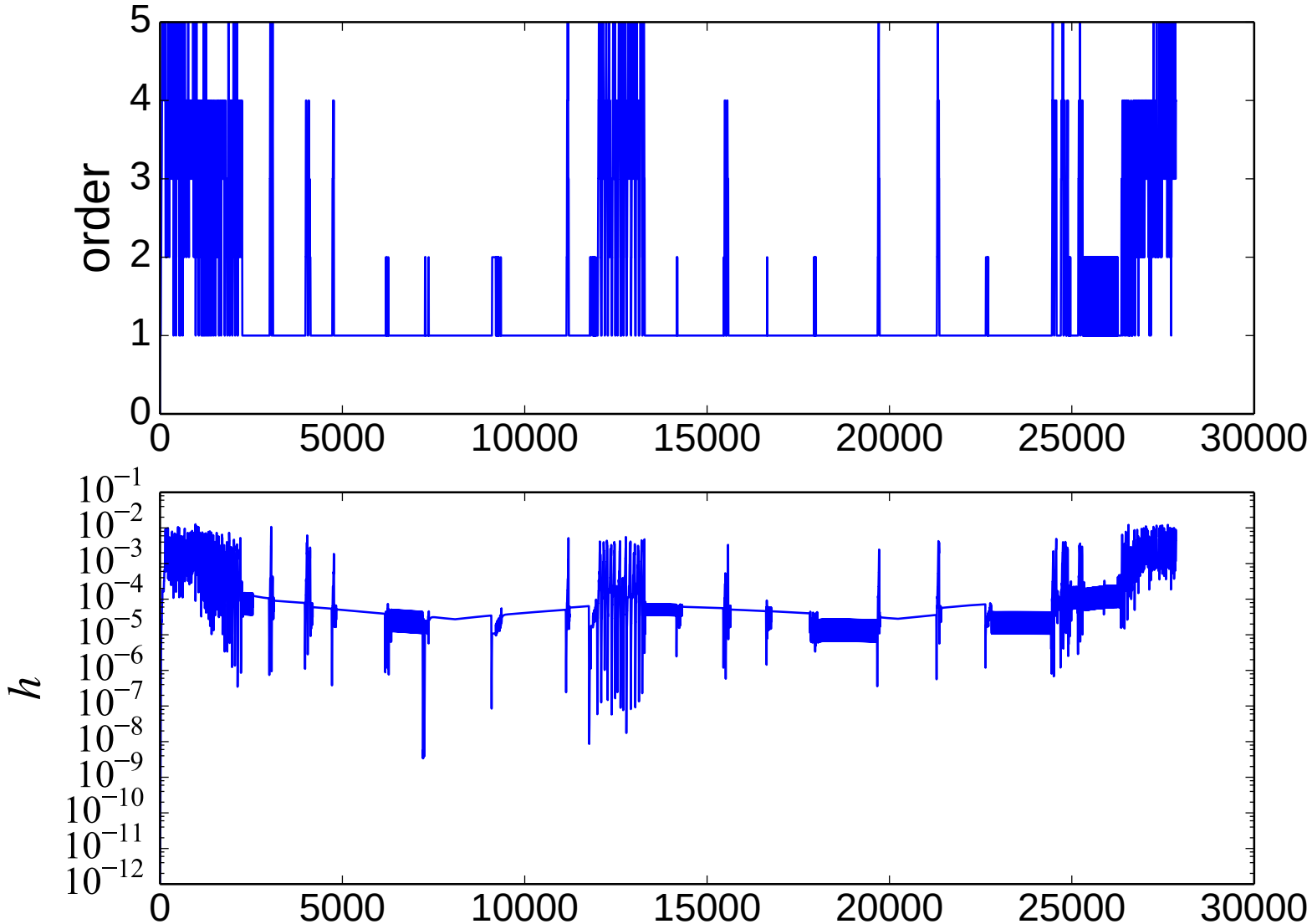
```
Final Run Statistics: ---

 Number of steps                                 : 3240
 Number of function evaluations                  : 7617
 Number of Jacobian evaluations                  : 1120
 Number of function eval. due to Jacobian eval.  : 15680
 Number of error test failures                   : 476
 Number of nonlinear iterations                  : 7617
 Number of nonlinear convergence failures        : 0

Solver options:

 Solver                         : IDA (BDF)
 Maximal order                  : 5
 Suppressed algebr. variables   : True
 Tolerances (absolute)          : 1e-08
 Tolerances (relative)          : 1e-08
```

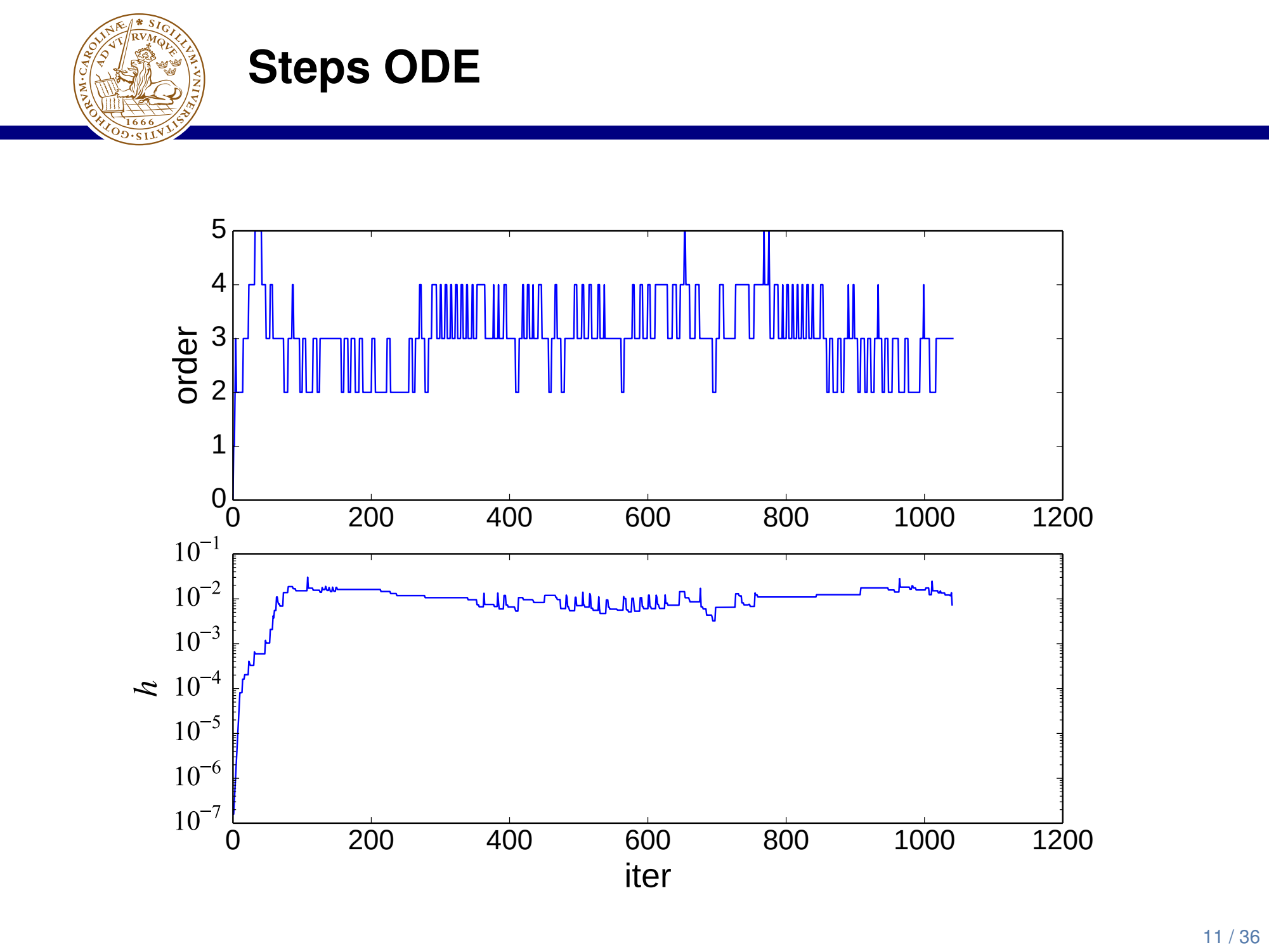
IDA ODE

```
Final Run Statistics: ---

 Number of steps                                 : 3428
 Number of function evaluations                  : 5524
 Number of Jacobian evaluations                  : 919
 Number of function eval. due to Jacobian eval.  : 9190
 Number of error test failures                   : 477
 Number of nonlinear iterations                  : 5524
 Number of nonlinear convergence failures        : 0

Solver options:

 Solver                         : IDA (BDF)
 Maximal order                  : 5
 Suppressed algebr. variables   : False
 Tolerances (absolute)          : 1e-08
 Tolerances (relative)          : 1e-08
```



Steps DAE


order
0
1
2
3
4
5
0
5000
10000
15000
20000
25000
30000
h
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-11}
10^{-12}
0
5000
10000
15000
20000
25000
30000



Steps ODE

order
0
1
2
3
4
5
0
200
400
600
800
1000
1200
h
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
200
400
600
800
1000
1200
iter

Index

- Key seems to be to deactivate error control for (some) algebraics
- Old conjecture: The problem is almost high index

Almost high index

This is my understanding of almost high index:

$$\dot{x} = x + y \tag{1}$$

$$y = ax \tag{2}$$

This is index 1 for all $\alpha \in \mathbb{R}$, but as $a \to \infty$, it becomes index 2.

Almost high index

- The change of variables $z = ax$ yields

$$\dot{z} = z + ay \tag{3}$$

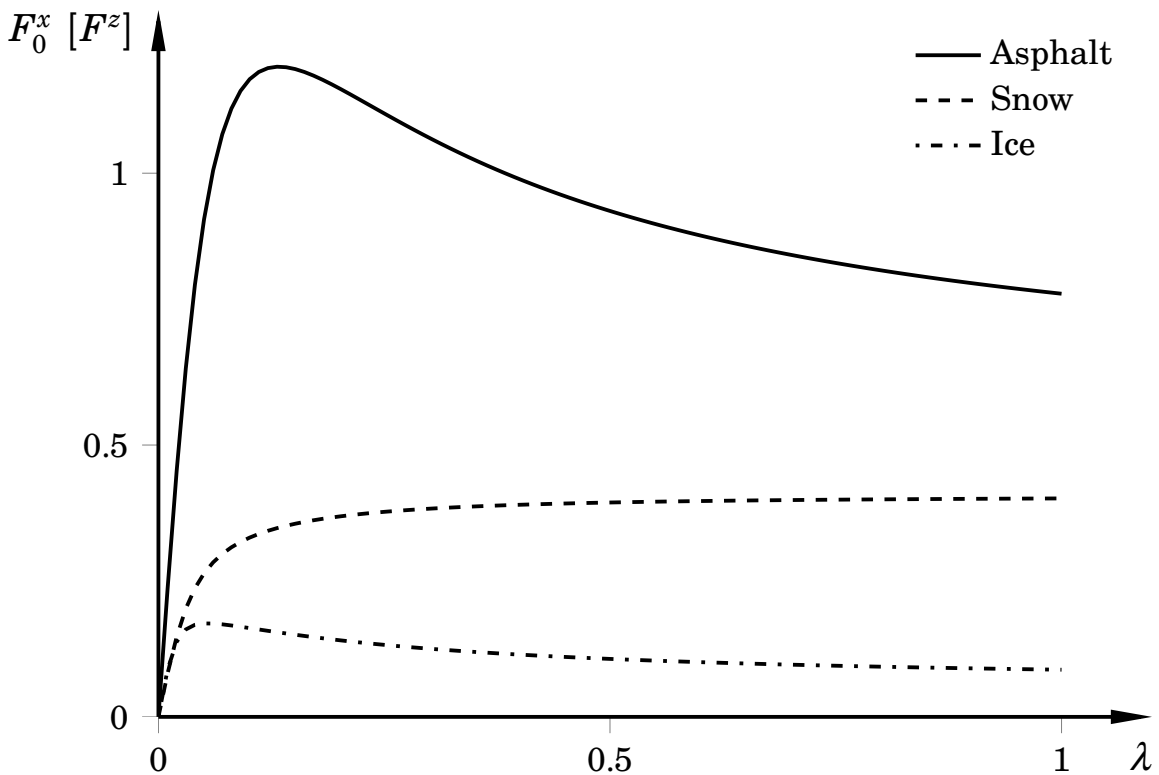
$$y = z \tag{4}$$

  which is index 1 even as $\alpha \to \infty$

- So is “almost high index” just a result of poor scaling?
- Also, it seems like transforming an almost high index DAE to an ODE will just result in a very stiff ODE

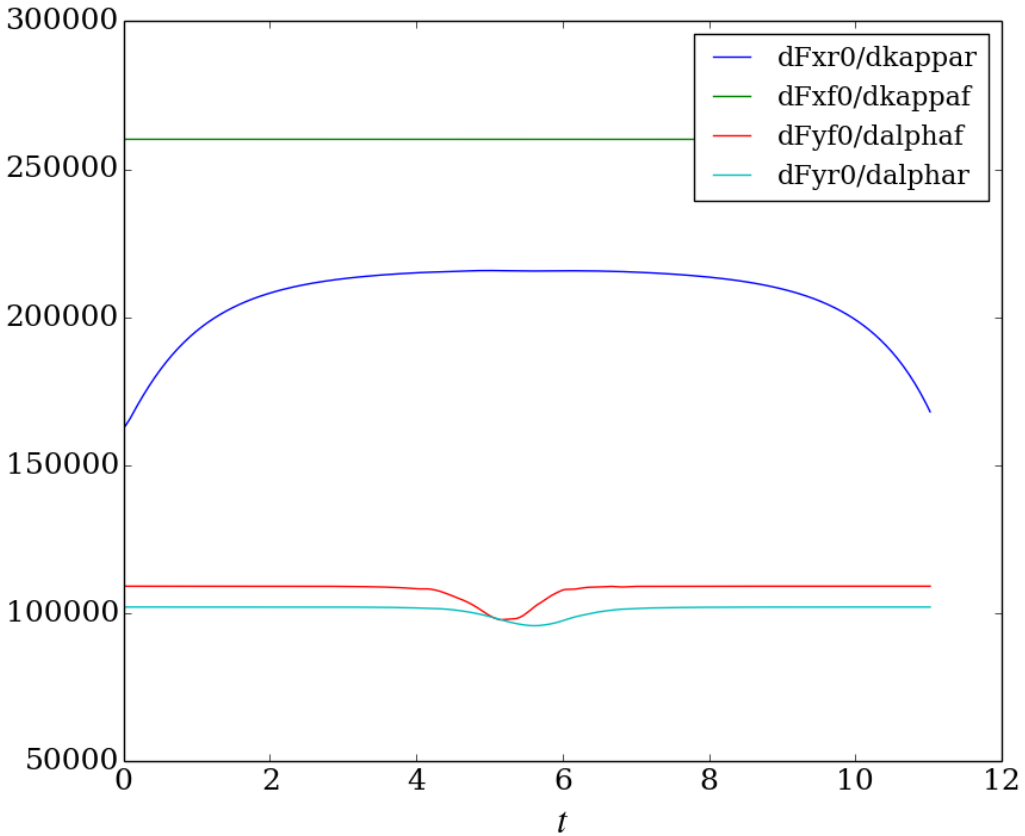
Magic formula

$$F_0^x = \mu_x F^z \sin\Big(C_x \arctan\big(B_x\lambda_i - E_x(B_x\lambda - \arctan B_x\lambda)\big)\Big) \quad (5)$$



Magic formula

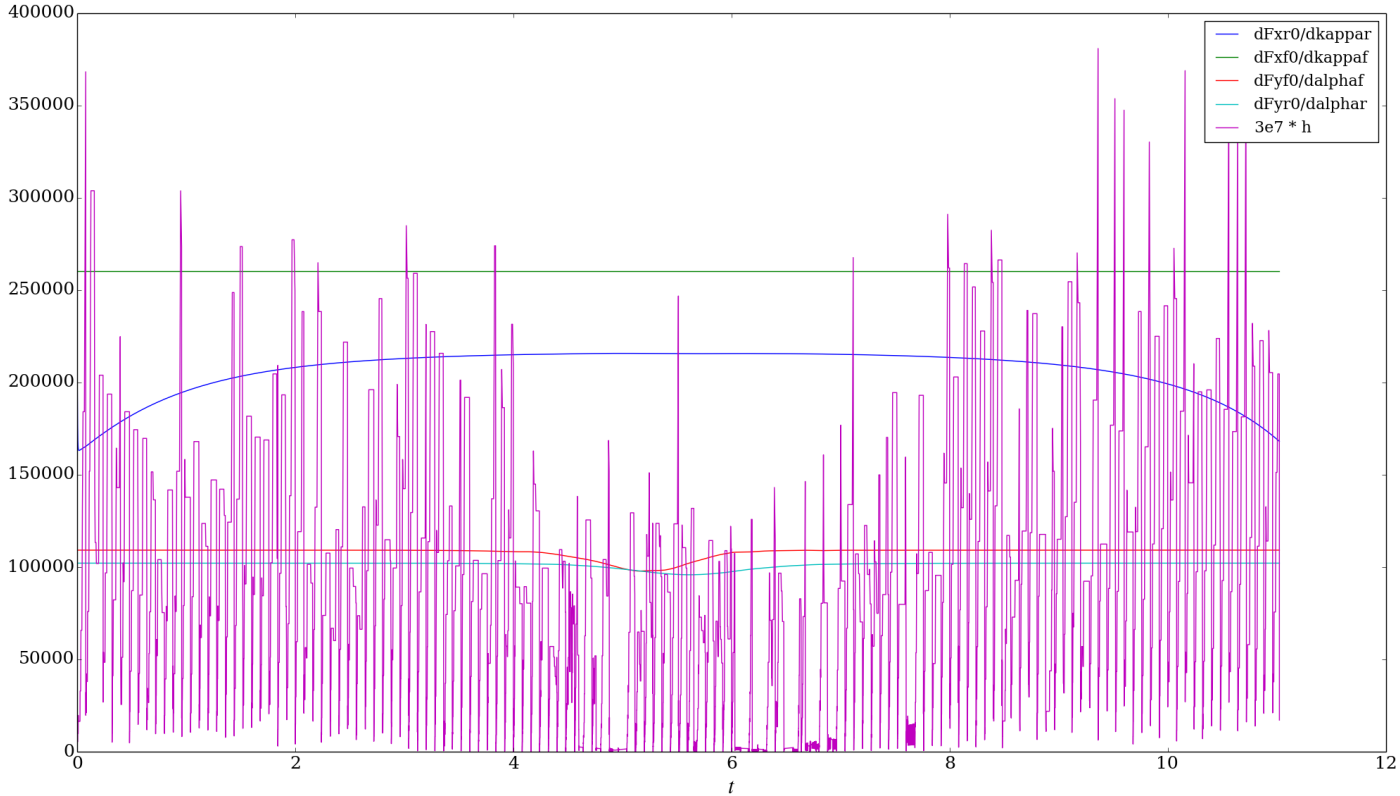
During simulation, λ stays below 0.05



Almost linear, with very high eigenvalues (can be remedied by variable scaling)

Magic formula

Changes in gradient not cause of trouble:



Measuring index

- How to measure distance to index 2? Two ideas:
  - Compute condition number of iteration matrix in discretization method and see how it varies with h
  - Analyze singular values of Jacobian of derivative array equations

Iteration matrix condition number

Consider

$$F(\dot{x}, x, y) = 0$$
$$G(x, y) = 0$$

- BDF iteration matrix:

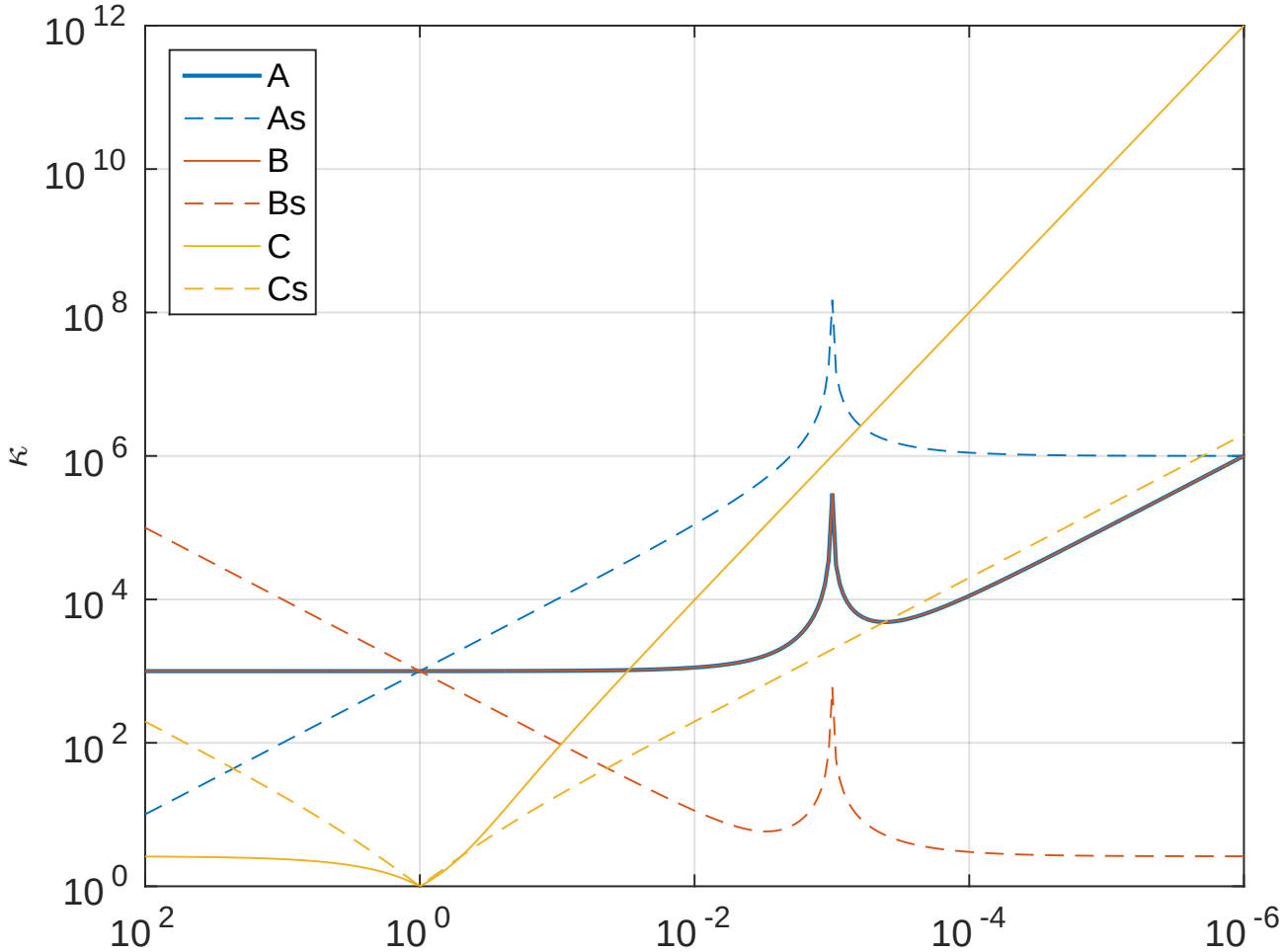
$$J = \begin{bmatrix} \frac{\alpha_0}{h}\nabla_{\dot{x}}F + \nabla_x F & \nabla_y F \\ \nabla_x G & \nabla_y G \end{bmatrix} \tag{6}$$

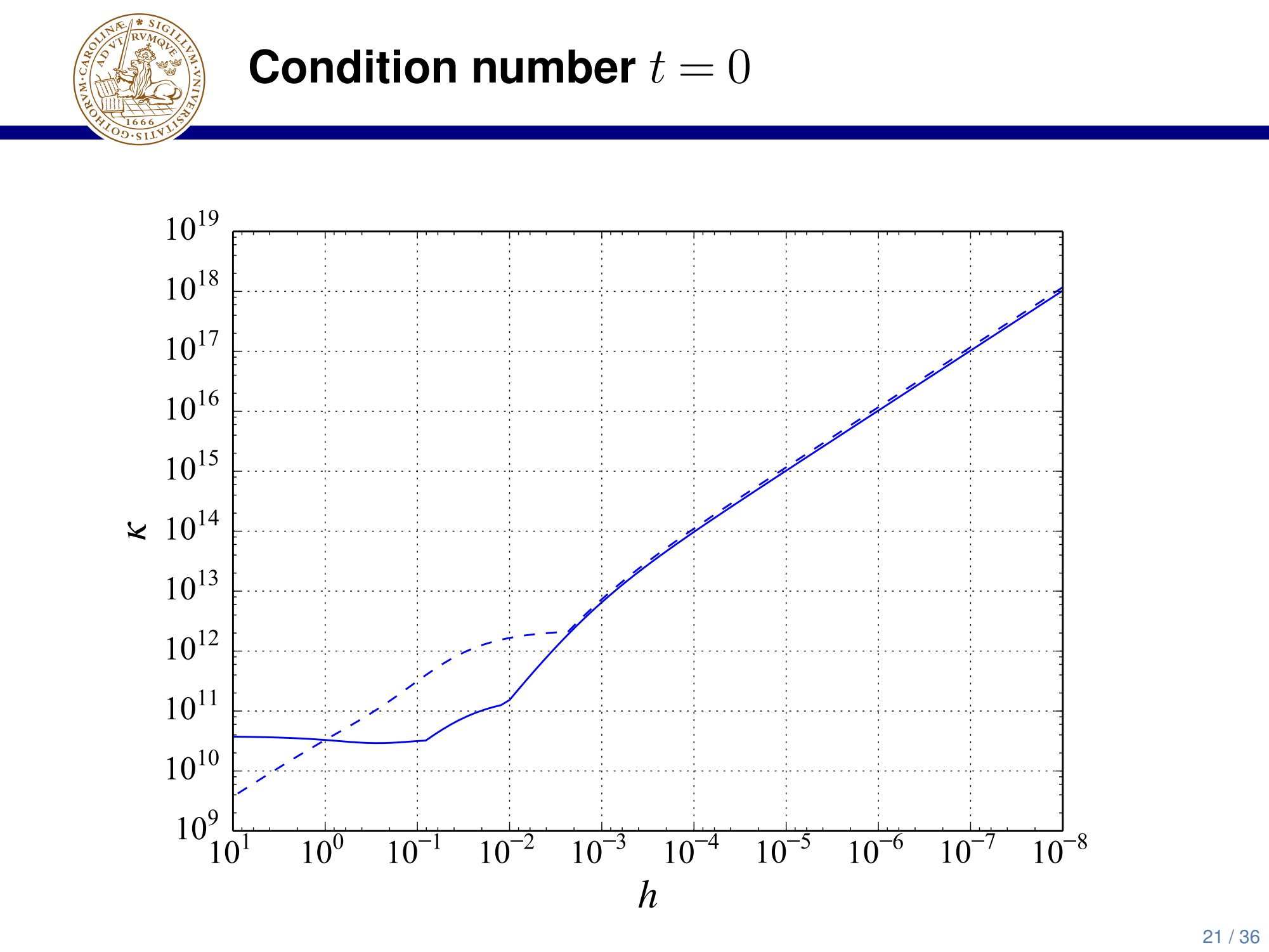
- (Theorem 5.4.1) Condition number of iteration matrix is $\mathcal{O}(h^{-\nu})$
- Conjecture: If $\nabla_{\dot{x}}F = I$ and first row of J is scaled by h, condition number is instead $\mathcal{O}(h^{-\nu-1})$
  - What to do if $\nabla_{\dot{x}}F$ is not square? What does IDA do?
  - Maybe I should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

Theorem and conjecture verification

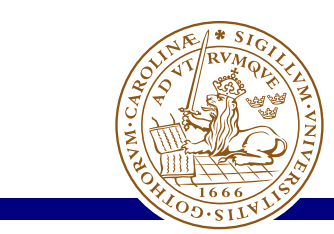
A: original problem B: variable scaling

C: $a = \infty$ s: conjecture scaling

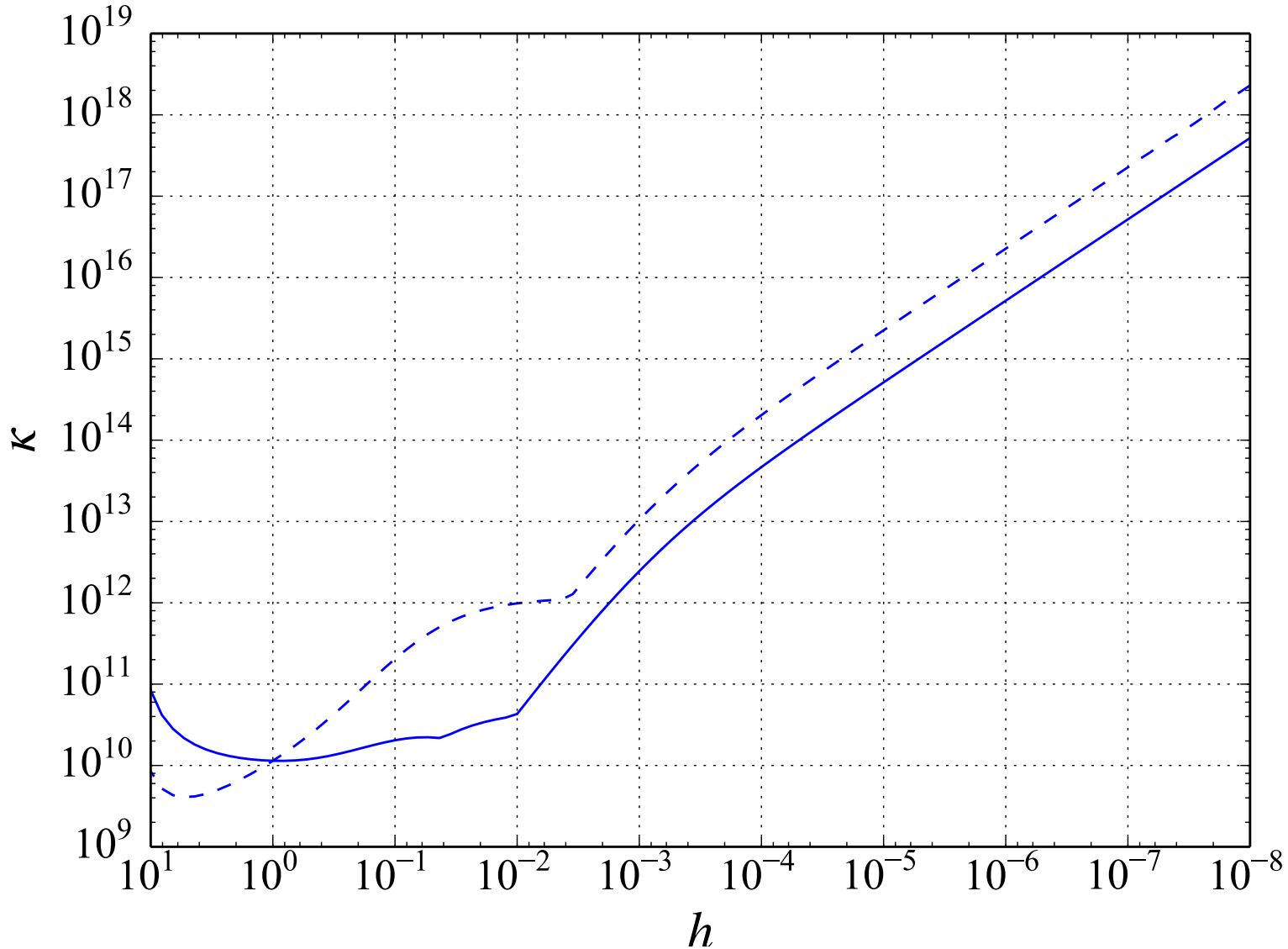




Condition number $t = 0$



Condition number $t = 6.21$



Derivative array equations

Consider $F(x, \dot{x}) = 0$

- Derivative array equations

$$\boldsymbol{F}(x, \dot{x}, \ddot{x}) = \begin{bmatrix} F(x, \dot{x}) \\ \frac{\mathrm{d}}{\mathrm{d}t} F(x, \dot{x}) \end{bmatrix} = 0 \tag{7}$$

- Proposition 2.5.1: Index is 1 if $\dfrac{\partial \boldsymbol{F}(x, \dot{x}, \ddot{x})}{\partial [\dot{x}, \ddot{x}]}$ is 1-full w.r.t. $\dot{x}$ and has constant rank
- Measure: $\dfrac{\sigma_1}{\sigma_k}$, where σ_k is smallest singular value whose singular vector involves $\dot{x}$

Derivative array equations

Problems with this measure:

- Depends on problem scaling (good or bad?)
- Requires $\ddot{x}$ (could be obtained by adding states? might interfere?)

Next steps

- How to proceed?
- Claus mentioned stability issues before...

Commutativity

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

$$\begin{aligned} \dot{x} &= f(x, y, u) \\ y &= g(x, u), \end{aligned} \tag{8}$$

and

$$\dot{x} = f(x, g(x, u), u) \tag{9}$$

Commutativity

Explicit DAE:

$$\dot{x}_k = f(x_k, y_k, u_k), \tag{10a}$$

$$y_k = g(x_k, u_k), \tag{10b}$$

$$\dot{x}_k = \frac{1}{h} \cdot \sum_{n=0}^{n_c} \alpha_{n,k} x_n, \tag{10c}$$

$$\forall k \in [1..n_c] \tag{10d}$$

Explicit ODE:

$$\dot{x}_k = f(x_k, g(x_k, u_k), u_k), \tag{11a}$$

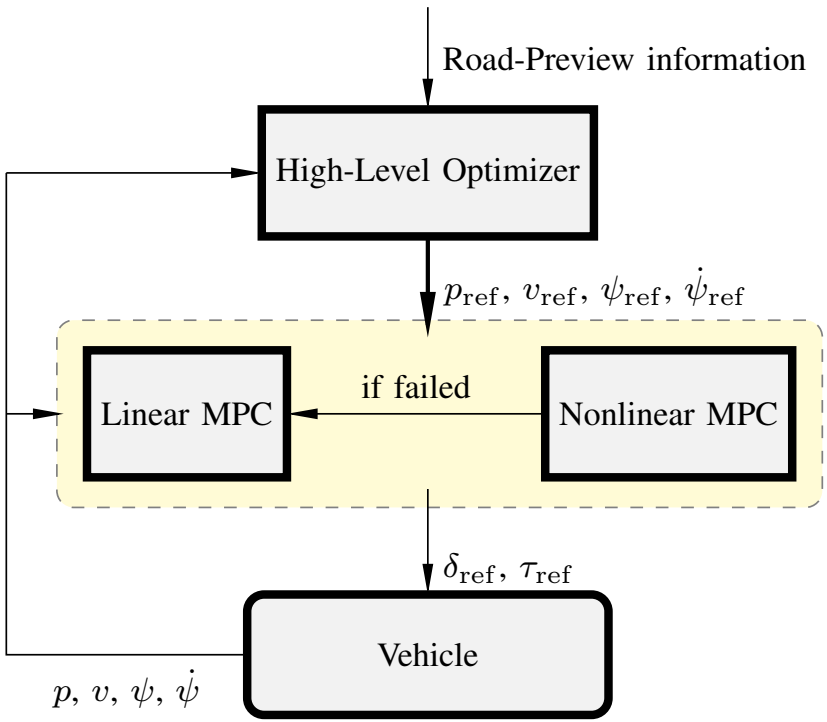
$$\dot{x}_k = \frac{1}{h} \cdot \sum_{n=0}^{n_c} \alpha_{n,k} x_n, \tag{11b}$$

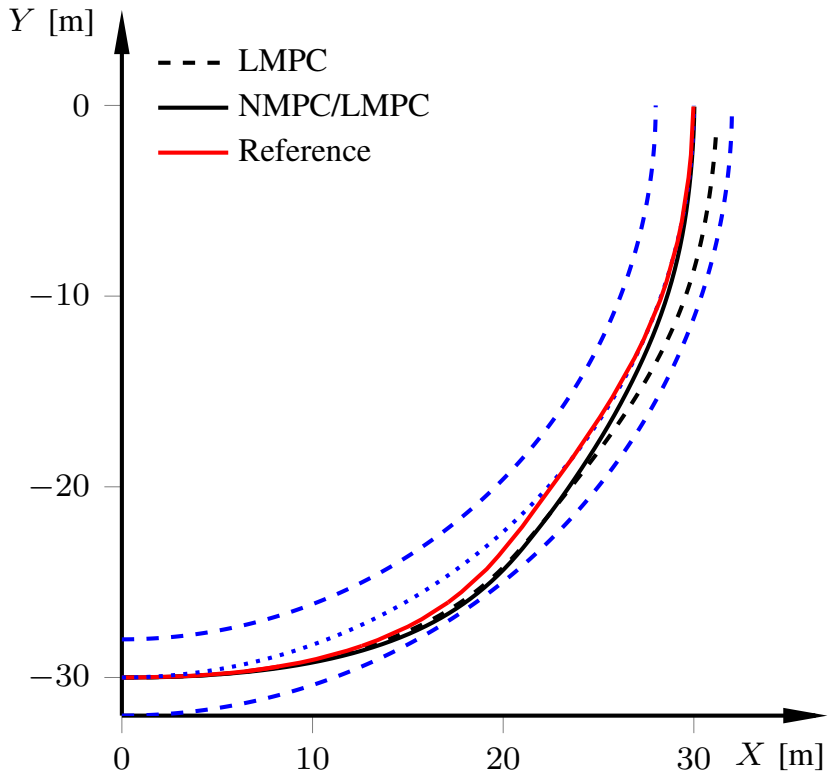
$$\forall k \in [1..n_c] \tag{11c}$$

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 $\Longrightarrow$ difficult
- Low level has fixed time horizon (MPC) $\Longrightarrow$ 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:

$$
\begin{aligned}
\underset{\delta_{\text{ref}},\tau_{f,\text{ref}},\tau_{r,\text{ref}}}{\text{minimize}} \quad & \int_{t_0}^{t_f} (\kappa_1 e^2 + \kappa_2 \beta^2)\, \mathrm{d}t \\
\text{subject to} \quad & |\tau_{i,\text{ref}}| \leq \tau_{i,\max},\ \forall i \in \{f, r\}, \\
& |\delta_{\text{ref}}| \leq \delta_{\max}, \\
& \|p(t_f) - p_{t_f}\| \leq \epsilon, \\
& \Gamma(p) \leq 0, \quad x(t_0) = x_0, \\
& F(\dot{x}, x, y, \delta_{\text{ref}}, \tau_{f,\text{ref}}, \tau_{r,\text{ref}}) = 0
\end{aligned}
\tag{12}
$$

Dynamic optimization

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

$$
\begin{aligned}
&\text{minimize} && \int_{t_0}^{t_f} L(\boldsymbol{z}(t))\,\mathrm{d}t, && && \text{(13a)}\\
&\text{with respect to} && \boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^{n_x}, && \boldsymbol{y} : [t_0, t_f] \to \mathbb{R}^{n_y}, &\\
& && \boldsymbol{u} : [t_0, t_f] \to \mathbb{R}^{n_u}, && t_f \in \mathbb{R}, &\\
&\text{subject to} && \boldsymbol{F}(\boldsymbol{z}(t)) = \boldsymbol{0}, && \boldsymbol{F}_0(\boldsymbol{z}(t_0)) = \boldsymbol{0}, && \text{(13b)}\\
& && \boldsymbol{z}_L \le \boldsymbol{z}(t) \le \boldsymbol{z}_U, && && \text{(13c)}\\
& && \boldsymbol{g}(\boldsymbol{z}(t)) \le \boldsymbol{0}, && \boldsymbol{G}(\boldsymbol{z}(t_f)) \le \boldsymbol{0}, && \text{(13d)}\\
& && \forall t \in [t_0, t_f]. &&
\end{aligned}
$$

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):

$$\text{minimize} \quad \sum_{i=1}^{n_e} h_i \cdot (t_f - t_0) \cdot \sum_{k=1}^{n_c} \omega_k \cdot L\left(\boldsymbol{z}_{i,k}\right), \tag{14a}$$

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

$$\text{subject to} \quad \boldsymbol{F}(\boldsymbol{z}_{i,k}) = \boldsymbol{0}, \qquad \boldsymbol{F}_0(\boldsymbol{z}_{1,0}) = \boldsymbol{0}, \tag{14b}$$

$$\boldsymbol{u}_{1,0} = \sum_{k=1}^{n_c} \boldsymbol{u}_{1,k} \cdot \ell_k(0) \qquad \boldsymbol{z}_L \le \boldsymbol{z}_{i,k} \le \boldsymbol{z}_U, \tag{14c}$$

$$\boldsymbol{g}(\boldsymbol{z}_{i,k}) = \boldsymbol{0}, \qquad \boldsymbol{G}(\boldsymbol{z}_{n_e,n_c}) \le \boldsymbol{0}, \tag{14d}$$

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

$$\dot{\boldsymbol{x}}_{j,l} = \frac{1}{h_j \cdot (t_f - t_0)} \cdot \sum_{m=0}^{n_c} \boldsymbol{x}_{j,m} \cdot \frac{\mathrm{d}\tilde{\ell}_m}{\mathrm{d}\tau}(\tau_l), \tag{14e}$$

$$\forall (j,l) \in [1..n_e] \times [1..n_c], \tag{14f}$$

$$\boldsymbol{x}_{n,n_c} = \boldsymbol{x}_{n+1,0}, \quad \forall n \in [1..n_e - 1]. \tag{14g}$$

NLP solution

After further abstraction, the NLP is:

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

- 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

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

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

Next steps

- Analyzing IPOPT convergence issues is complicated
  - Next step I see in that direction is to compare cond$(\nabla_z F)$ for DAE and ODE formulation
  - If significant difference, that is a likely explanation, but then we should try to figure out why there is a significant difference...
- Still don't understand why ODE is better for simulation
  - Probably not related to ill-conditioning?