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

October 1, 2015

Introduction

- Want to solve optimal control problems for vehicle maneuvers (optimize wheel torques and steer angle)
- Modeled in Modelica
- Models of varying complexity, 10-100 equations
- All of them are index-one

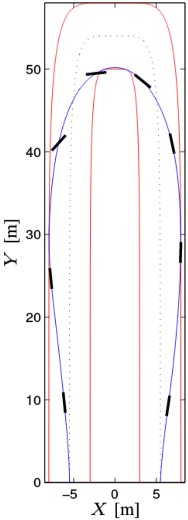
The model

- Will today focus on a single-track model using weighting functions for tire dynamics
- 10 differential variables, 23 algebraic variables
- Implicit DAE can be transformed to explicit DAE:

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

  with closed-form expression for f and g
- Then trivial to get explicit ODE:

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



Time discretization

- Discretize dynamics with collocation (Radau5)
- Can either discretize the full DAE, or the transformed explicit ODE
- Interested in how this transformation affects optimization performance
- Experience is that the explicit ODE formulation usually is superior
    - Requires fewer iterations
    - Cheaper iterations
    - More likely to succeed

Implicit vs. explicit

- Want to understand why explicit ODE formulation is superior
- Probably not as simple as the problem being smaller
- Have noticed that the ODE formulation also is superior for simulation purposes
- If you help us understand why, maybe we can understand why ODE formulation is superior for optimization

Equations

Chassis dynamics (7 equations):

$$\dot{v}^X - v^Y \dot{\psi} = F_g^x \tag{1}$$

$$F_g^x = \frac{1}{m}(F_f^x \cos(\delta) + F_r^x - F_f^y \sin(\delta)) \tag{2}$$

$$\dot{v}^Y + v^X \dot{\psi} = F_g^y \tag{3}$$

$$F_g^y = \frac{1}{m}(F_f^y \cos(\delta) + F_r^y + F_f^x \sin(\delta)) \tag{4}$$

$$I_{ZZ}\ddot{\psi} = M_g^z \tag{5}$$

$$M_g^z = l_f F_f^y \cos(\delta) - l_r F_r^y + l_f F_f^x \sin(\delta) \tag{6}$$

Nominal tire forces (4 equations):

$$F_0^x = \mu_x F^z \sin\left(C_x \arctan\left(B_x \lambda_i - E_x(B_x \lambda - \arctan B_x \lambda)\right)\right) \tag{7}$$

$$F_0^y = \mu_y F^z \sin\left(C_y \arctan\left(B_y \alpha - E_y(B_y \alpha - \arctan B_y \alpha)\right)\right) \tag{8}$$

Equations cont.

Actual tire forces (12 equations):

$$F^{x,y} = F_0^{x,y} G_m \tag{9}$$
$$G_m = \cos(C_m \arctan(H_m m)) \tag{10}$$
$$H_m = B_{m1} \cos(\arctan(B_{m2} m)) \tag{11}$$

Wheel dynamics (2 equations):

$$\tau = I_w \dot{\omega} - R_w F^x, \tag{12}$$

Equation cont.

Slip (4 equations):

$$\dot{\alpha}_i \frac{\sigma}{v_i^x} + \alpha_i := -\arctan\left(\frac{v_i^y}{v_i^x}\right) \tag{13}$$

$$\lambda_i := \frac{R_w \omega_i - v_i^x}{v_i^x} \tag{14}$$

Vehicle sideslip (1 equation):

$$\beta = \arctan\left(\frac{v^Y}{v^X}\right) \tag{15}$$

Velocities (3 equations):

$$v_f^x = v^X \cos(\delta) + (v^Y + l_f \dot{\psi}) \sin(\delta) \tag{16}$$

$$\dot{X} = v^X \cos(\psi) - v^Y \sin(\psi) \tag{17}$$

$$\dot{Y} = v^X \sin(\psi) + v^Y \cos(\psi) \tag{18}$$

BLT form

	beta	Bykf	der(psi)	vxf	kappaf	Gykf	Fyf0	Fyf	Bykr	kappar	Gykr	Fyr0	Fyr	Bxaf	Gxaf	Fxf0	Fxf	Mz_g	Bxar	Gxar	Fxr0	Fxr	Fx_g	Fy_g	der(vx)	der(vy)	der(dpsi)	der(omegaf)	der(omegar)	der(alphaf)	der(alphar)	der(X)	der(Y)
beta = atan(vy / vx)	O																																
Bykf = By1f * cos(atan(By2f * alphaf))		O																															
der(psi) = dpsi			O																														
vxf = vx * cos(delta) + (vy + lf * der(psi)) * sir			X	O																													
kappaf = (Re * omegaf - vxf) / vxf				X	O																												
Gykf = cos(Cykf * atan(Bykf * kappaf))		X			X	O																											
Fyf0 = muyf * Fzf * sin(Cyf * atan(Byf * alpha							O																										
Fyf = Gykf * Fyf0						O	O	O																									
Bykr = By1r * cos(atan(By2r * alphar))									O																								
kappar = (Re * omegar - vx) / vx										O																							
Gykr = cos(Cykr * atan(Bykr * kappar))									X	X	O																						
Fyr0 = muyr * Fzr * sin(Cyr * atan(Byr * alph												O																					
Fyr = Gykr * Fyr0											O	O	O																				
Bxaf = Bx1f * cos(atan(Bx2f * kappaf))					X									O																			
Gxaf = cos(Cxaf * atan(Bxaf * alphaf))														X	O																		
Fxf0 = muxf * Fzf * sin(Cxf * atan(Bxf * kappa					X											O																	
Fxf = Gxaf * Fxf0															O	O	O																
Mz_g = lf * Fyf * cos(delta) - lr * Fyr + lf * Fxf								O					O				O	O															
Bxar = Bx1r * cos(atan(Bx2r * kappar))										X									O														
Gxar = cos(Cxar * atan(Bxar * alphar))																			X	O													
Fxr0 = muxr * Fzr * sin(Cxr * atan(Bxr * kapp										X											O												
Fxr = Gxar * Fxr0																				O	O	O											
Fx_g = Fxf * cos(delta) + Fxr - Fyf * sin(delta								O									O					O	O										
Fy_g = Fyf * cos(delta) + Fyr + Fxf * sin(delt								O					O				O							O									
((- vy) * der(psi) + der(vx)) * m = Fx_g			X																				O		X								
(der(vy) + vx * der(psi)) * m = Fy_g			X																					O		X							
der(dpsi) * Iz = Mz_g																		O									O						
Twf - 2 * Iw * der(omegaf) - Fxf * Rw = 0																	O											O					
Twr - 2 * Iw * der(omegar) - Fxr * Rw = 0																						O							O				
der(alphaf) * sig / vxf + alphaf = delta - atan(			X	X																										O			
der(alphar) * sig / vx + alphar = - atan((vy - l			X																												O		
der(X) = vx * cos(psi) - vy * sin(psi)																																O	
der(Y) = vx * sin(psi) + vy * cos(psi)																																	O

Implementation

- We do not utilize JModelica's BLT and FMI framework
- Instead transfer the full symbolic DAE to CasADi Interface
- Perform our own BLT and ODE transformation using CasADi
- Then hook up the causalized system either to JModelica's collocation for optimization or Assimulo for simulation
- Can skip our own ODE transformation to expose the full DAE to the optimization or Assimulo

Simulation results

Setup	Time [s]	Steps [1000]	Evals [1000]
IDA DAE	43.4	33	674
IDA DAE sup. alg.	1.4	1.5	21
IDA ODE	0.5	1.8	8
Radau5 DAE	5.7	2.0	51
Radau5 ODE	0.4	0.2	4

IDA DAE

```
Final Run Statistics: ---

 Number of steps                                 : 32885
 Number of function evaluations                  : 67297
 Number of Jacobian evaluations                  : 18383
 Number of function eval. due to Jacobian eval.  : 606639
 Number of error test failures                   : 7482
 Number of nonlinear iterations                  : 67297
 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-08
```

IDA DAE sup. alg.

```
Final Run Statistics: ---

 Number of steps                                  : 1405
 Number of function evaluations                   : 3897
 Number of Jacobian evaluations                   : 525
 Number of function eval. due to Jacobian eval.   : 17325
 Number of error test failures                    : 215
 Number of nonlinear iterations                   : 3897
 Number of nonlinear convergence failures         : 0

Solver options:

 Solver                        : IDA (BDF)
 Maximal order                 : 5
 Suppressed algebr. variables  : True
 Tolerances (absolute)         : 1e-06
 Tolerances (relative)         : 1e-08
```

IDA ODE

```
Final Run Statistics: ---

 Number of steps                                    : 1789
 Number of function evaluations                     : 2972
 Number of Jacobian evaluations                     : 458
 Number of function eval. due to Jacobian eval.     : 4580
 Number of error test failures                      : 271
 Number of nonlinear iterations                     : 2972
 Number of nonlinear convergence failures           : 0

Solver options:

 Solver                        : IDA (BDF)
 Maximal order                 : 5
 Suppressed algebr. variables  : True
 Tolerances (absolute)         : 1e-06
 Tolerances (relative)         : 1e-08
```

Radau5 DAE

```
Final Run Statistics: ---

 Number of steps                                 : 1999
 Number of function evaluations                  : 17855
 Number of Jacobian evaluations                  : 1009
 Number of function eval. due to Jacobian eval.  : 33297
 Number of error test failures                   : 585
 Number of LU decompositions                     : 2264

Solver options:

 Solver                   : Radau5 (implicit)
 Tolerances (absolute)    : [  1.00000000e-06]
 Tolerances (relative)    : 1e-08
```

Radau5 ODE

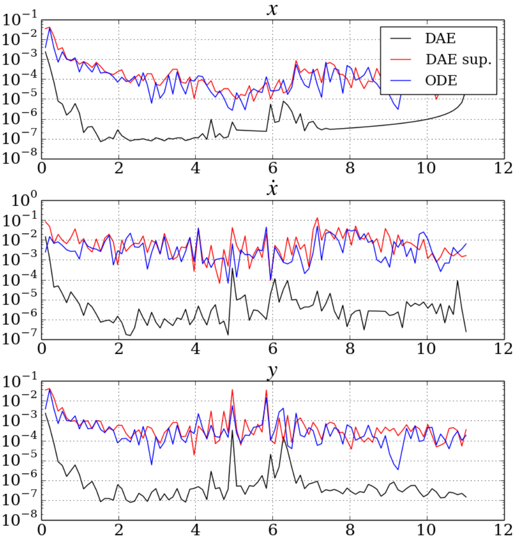
```
Final Run Statistics: ---

 Number of steps                                 : 207
 Number of function evaluations                  : 1994
 Number of Jacobian evaluations                  : 192
 Number of function eval. due to Jacobian eval.  : 1920
 Number of error test failures                   : 31
 Number of LU decompositions                     : 240

Solver options:

 Solver                       : Radau5 (implicit)
 Tolerances (absolute)        : [  1.00000000e-06]
 Tolerances (relative)        : 1e-08
```

Error

- ODE formulation behaves similarly to DAE formulation with suppressed algebraics (faster since Jacobian computation is cheaper)
- Does this mean we lose accuracy in the algebraics with the ODE formulation?
- First compute "reference" solution $\dot{x}, x, y$ with DAE formulation and `RTOL=1e-12, ATOL=1e-8`
- Compare this with DAE, DAE sup. alg., and ODE with default tolerances
- Compute relative error as function of time for each variable kind:

$$e_v(t) = \left\| \frac{v(t) - \hat{v}(t)}{v(t) + \epsilon_{\text{mach}}} \right\|_\infty, \quad \forall v \in \{\dot{x}, x, y\}$$

Variable kind errors



Variable kind errors

- Very roughly speaking, suppressing algebraics or transforming to ODE increases relative error from 10^{-6} to 10^{-3} for all variable kinds.
- Unexpected?

Optimization?

- We hoped that understanding these things would help us understand why ODE is good for optimization
- But now I think that these are separate issues, since we have fixed step length in optimization
- Some tolerance levels start giving nonlinear convergence failures for DAE
  - Unfortunately, unable to reproduce
  - Could be related to what we are seeing in optimization?
  - Is it worth analyzing (if we manage to reproduce)?
  - Problem conditioning is not straightforward to measure in optimization, but it seems like there is no significant difference between the formulations in this regard