ON THE IMPLICIT INTEGRATION OF DIFFERENTIAL-ALGEBRIAC EQUATIONS OF MULTIBODY DYNAMICS

Dan Negrut
The University of Iowa
July 1998
**MOTIVATION**

**HMMWV14 RESULTS - EXPLICIT INTEGRATION**

- Vehicle Driven Straight At 10mph
- Hits Bump (Half Cylinder, Diameter 0.2m)
- Simulate With *ExpDEABM*
  - Explicit Code
  - Based On DDEABM Of Shampine And Watts
- 1 Second Of Simulation Requires About 1 Hour CPU Time
- The Code Takes Approximately 160,000 Time Steps For Any Tolerance Between 1.0E-2 And 1.0E-5

---

*The University of Iowa*
MOTIVATION
HMMWV14 RESULTS - EXPLICIT INTEGRATION (Contd.)

- CPU Times Are Very Large
- Poor Performance Determined By Inability Of Analysis Tools To Deal With The Problem, And Not By The Complexity of The Model
- Performance Of The Overall Algorithm Limited By Embedded ODE Integrator
  - Stiffness Is Causing The Trouble
- Implicit Integrators Deal With Stiffness Effectively

The University of Iowa
MOTIVATION AND GOALS

- Motivation For Implicit Integration
  - Industrial Grade Applications Are Often Stiff
    - Bushings
    - Tire Modeling
    - Flexible Components

- Goal - Develop Mathematical Algorithms And Software Libraries To Allow Implicit Integration Of Differential-Algebraic Equations Of Multibody Dynamics (DAEMD)
  - Reliability
  - Generality
  - Efficiency
Differential-Algebraic Equations of Multibody Dynamics

- Position Kinematic Constraint Equation
  \[ \Phi(q) = 0 \]

- Velocity Kinematic Constraint Equation
  \[ \Phi_q(q)q = 0 \]

- Acceleration Kinematic Constraint Equation
  \[ \Phi_q(q)\ddot{q} = \tau(q, \dot{q}) \]

- Equations Of Motion
  \[ M(q)\ddot{q} + \Phi_q^T(q)\lambda = Q^A(t, q, \dot{q}) \]
METHODS USED FOR SOLUTION OF DAEMD

- Constraint Stabilization Techniques
- Projection Methods
  - Derivative Projection
  - Position Projection
- State-Space Methods
STATE SPACE REDUCTION VIA
GENERALIZED COORDINATE PARTITIONING

- The DAEMD Are Reduced To The State-Space Second Order Differential Equations (SSODE) In Independent Positions $\ddot{v}$:

$$\hat{M}\ddot{v} = \hat{Q}$$

$$\hat{M} = M^{vv} - M^{vu}\Phi_u^{-1}\Phi_v - \Phi_v^T\Phi_u^{-T}(M^{uv} - M^{uu}\Phi_u^{-1}\Phi_v)$$

$$\hat{Q} = Q^v - \Phi_v^T\Phi_u^{-T}Q^u - (M^{vv} - \Phi_v^T\Phi_u^{-T}M^{uu})\Phi_u^{-1}\tau$$
METHODS DEVELOPED

- All Methods Are State-Space Methods
  - State-Space Reduction Method
  - Descriptor Form Method
  - First-Order Reduction Method
STATE SPACE REDUCTION METHOD

- DAE Reduced To ODE Of Dimension Equal To The Number Of Degrees Of Freedom Of The Mechanical System Model

- **Pros:**
  - Small Dimension
  - Fast Linear Algebra

- **Cons:**
  - Expensive Integration Jacobian Computation (55% Of Simulation CPU Time)

\[
\Psi_v = M^{\nu\nu} + M^{\nu u} H + H^T (M^{u\nu} + M^{uu} H) + \gamma h (M^{\nu u} N + H^T S - Q^y_v - Q^y_u H) \\
+ \beta h^2 [(M^{\nu \bar{q}})_u H + (M^{\nu \bar{q}})_v + M^{\nu u} L + H^T R + (\Phi^T_v \lambda)_u H + (\Phi^T_v \lambda)_v \\
- Q^y_u H - Q^y_v - Q^y_u J]
\]
**Descriptor Form Method**

- Discretization Done At The Index One DAE Level
- Newton-Like Approach Solves For Generalized Accelerations And Lagrange Multipliers

**Pros:**
- Simpler Integration Jacobian Computation

**Cons:**
- Large Dimension Of Resulting Numerical Problem
- Costly Linear Algebra

\[
\Psi = \begin{bmatrix}
M - \gamma h Q_q^A \dot{\hat{H}} + \beta h^2 \{(M \ddot{q})_q + (\Phi_q^T \lambda)_q - Q_q^A \hat{H} - Q_q^A \dot{J}\} & \Phi_q^T \\
\Phi_q - \gamma h \tau_q \dot{\hat{H}} + \beta h^2 \{(\Phi_q \ddot{q})_q - \tau_q \hat{H} - \tau_q \dot{J}\} & 0
\end{bmatrix}
\]
FIRST-ORDER REDUCTION METHOD

- DAE Reduced To First Order System Of ODE In Independent Positions And Velocities
- Any Classical ODE Numerical Integration Code Used To Integrate The First Order ODE
- **Pros:**
  - Small Dimension
  - Fast Linear Algebra
- **Cons:**
  - Expensive Integration Jacobian Computation
- Algorithms For Efficient Acceleration Computation Are Required

The University of Iowa
ACCELERATION COMPUTATION

- Methods For Acceleration Computation Have Been Developed For Mechanical Systems Modeled Using
  - Cartesian Representation
  - Relative Coordinates Representation
- Speed-Ups Compared To Older Implementations
  - 1:4 → Cartesian Representation
  - 1:2 → Relative Coordinates Representation
ACCELERATION COMPUTATION
(Contd.)

- Speed-Ups Due To
  - Topology Based Linear Algebra
    - Sparsity Exploited
  - Computational Sequences That Take Advantage Of Problem Structure
    - Problem Reduction (For Cartesian Formulation)
## Implicit Integration of DAEMD

### Algorithms Implemented

- **Constructing An Algorithm:**
  
  **DAE Method + ODE Integrator + Linear Algebra Support**

- **All Algorithms Posses** **Error Control** Via Step-Size Selection

  - **State-Space Reduction-Based Algorithms:**
    - **SspTrap** - Trapezoidal Formula, Order 2, A-Stable
  
  - **Descriptor Form Method-Based Algorithms:**
    - **InflTrap** - Trapezoidal Formula, Order 2, A-Stable
    - **InflSDIRK** - SDIRK 5 Stage, Order 4, Stiffly Accurate, A, L-Stable
  
  - **First-Order Reduction-Based Algorithms:**
    - **ForSDIRK** - SDIRK 5 Stage, Order 4, Stiffly Accurate, A, L-Stable
    - **ForRosen** - Rosenbrock Order 4, 4 Stage, Stiffly Accurate, A, L-Stable
ALGORITHM VALIDATION
(DESCRIBER FORM METHOD-BASED ALGORITHMS)

- Validation Done For InflTrap And InflSDIRK Using HMMWV14 Model.
- Reference Solution Obtained With ForSDIRK With Integration Tolerance Set To 1.E-8
- Largest Error Over All Grid Points Reported As
  \[ \Delta^{(k)} \equiv \max_{1 \leq i \leq n} |E^*_i - e_i| \]
- Reference Solution At Off-Grid Points Obtained Via Cubic Spline Interpolation
- Simulation Length - 2 Seconds. Time Steps Taken (Reference Solution) - 579,779
ALGORITHM VALIDATION
(Contd.)

**InflTrap**

<table>
<thead>
<tr>
<th>Tol. $(k)$</th>
<th>Max. Error Position</th>
<th>Max. Error Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>0.001987670207</td>
<td>0.003299978287</td>
</tr>
<tr>
<td>-3</td>
<td>0.000412496160</td>
<td>0.000570435797</td>
</tr>
<tr>
<td>-4</td>
<td>0.000044509076</td>
<td>0.000061351047</td>
</tr>
<tr>
<td>-5</td>
<td>0.000005203414</td>
<td>0.000006412237</td>
</tr>
</tbody>
</table>

**InflSDIRK**

<table>
<thead>
<tr>
<th>Tol. $(k)$</th>
<th>Max. Error Position</th>
<th>Max. Error Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>0.000657081652</td>
<td>0.005078716955</td>
</tr>
<tr>
<td>-3</td>
<td>0.000045498338</td>
<td>0.002071677946</td>
</tr>
<tr>
<td>-4</td>
<td>0.000003370069</td>
<td>0.000134605346</td>
</tr>
<tr>
<td>-5</td>
<td>0.000000696890</td>
<td>0.000041544247</td>
</tr>
</tbody>
</table>
EXPLICIT vs. IMPLICIT COMPARISON

**ExplDEABM**

<table>
<thead>
<tr>
<th>TOL</th>
<th>10^{-2}</th>
<th>10^{-3}</th>
<th>10^{-4}</th>
<th>10^{-5}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 s</td>
<td>3618</td>
<td>3641</td>
<td>3667</td>
<td>3663</td>
</tr>
<tr>
<td>2 s</td>
<td>7276</td>
<td>7348</td>
<td>7287</td>
<td>7276</td>
</tr>
<tr>
<td>3 s</td>
<td>10865</td>
<td>11122</td>
<td>10949</td>
<td>10965</td>
</tr>
<tr>
<td>4 s</td>
<td>14480</td>
<td>14771</td>
<td>14630</td>
<td>14592</td>
</tr>
</tbody>
</table>

CPU Results Explicit Integration [seconds]

**ForSDIRK**

<table>
<thead>
<tr>
<th>TOL</th>
<th>10^{-2}</th>
<th>10^{-3}</th>
<th>10^{-4}</th>
<th>10^{-5}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 s</td>
<td>10.1</td>
<td>21</td>
<td>33</td>
<td>57</td>
</tr>
<tr>
<td>2 s</td>
<td>25.3</td>
<td>49</td>
<td>78</td>
<td>139</td>
</tr>
<tr>
<td>3 s</td>
<td>29.5</td>
<td>58</td>
<td>92</td>
<td>166</td>
</tr>
<tr>
<td>4 s</td>
<td>30</td>
<td>61</td>
<td>94</td>
<td>184</td>
</tr>
</tbody>
</table>

CPU Results ForSDIRK [seconds]
EXPLICIT vs. IMPLICIT COMPARISON (Contd.)

Chassis Height HMMWV14

Step-Size History:
ForSDIRK and InflTrap

The University of Iowa
CONCLUSIONS

- A *Mathematically Rigorous* Approach For Variable Step Size Implicit Integration Of DAE Of Multibody Dynamics, With Error Control, Was Developed

- A Family Of *Generally Applicable* Implicit Integrators Was Implemented

- *Two Orders Of Magnitude* Speed Up Were Obtained When Compared To State Of The Art Explicit Integrators
DIRECTIONS OF FUTURE WORK

- Implement methods for parallel computation of the integration Jacobian
- Investigate and implement the tangent-plane parametrization-based DAE-to-ODE reduction method
- Embed topology based linear algebra routines in numerical implementations of the First Order Reduction Method
- Improve stopping criteria for algorithms based on State-Space Reduction and Descriptor Form Methods
- Modify the step-size controller of the algorithm ForRosen, to eliminate its conservative estimate
- Apply methods developed for numerical solution of systems that include flexible bodies and intermittent motion