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

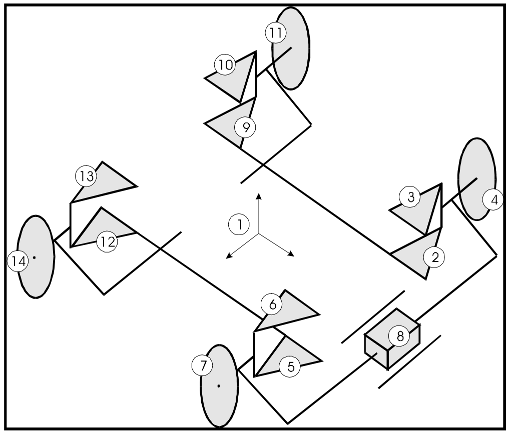
Dan Negrut

The University of Iowa

July 1998

MOTIVATION
HMMWV14 RESULTS - EXPLICIT INTEGRATION

US Army HMMWV

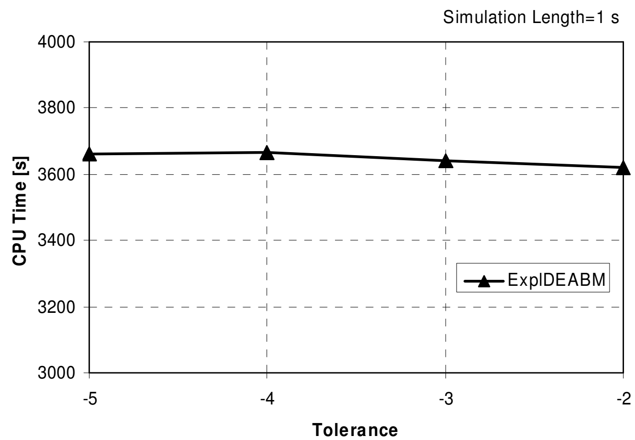


Topology Graph HMMWV14

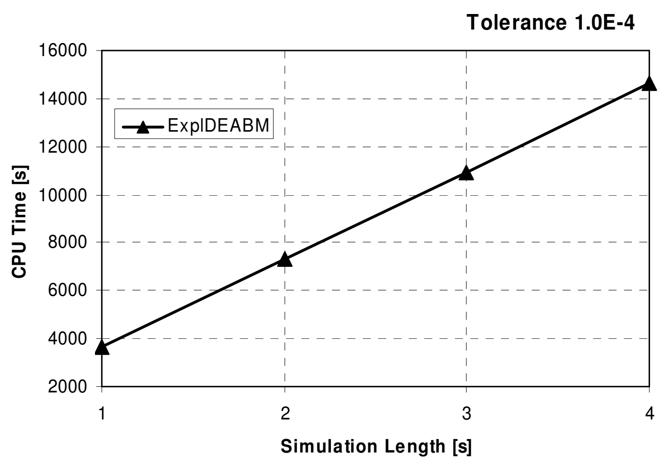
- Vehicle Driven Straight At 10mph
- Hits Bump (Half Cylinder, Diameter 0.2m)
- Simulate With *ExplDEABM*
  - Explicit Code
  - Based On DDEABM Of Shampine And Watts
- 1 Second Of Simulation Requires About 1 Hour CPU Time
- The Code Takes Approximately 160,000 Time Steps For Any Tolerance Between 1.0E-2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

MOTIVATION

HMMWV14 RESULTS - EXPLICIT INTEGRATION (Contd.)



CPU Time vs. Integration Tolerance



CPU Time vs. Simulation Length

- 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*

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

$$\mathbf{\Phi}(\mathbf{q}) = \mathbf{0}$$

- Velocity Kinematic Constraint Equation

$$\mathbf{\Phi}_{\mathbf{q}}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0}$$

- Acceleration Kinematic Constraint Equation

$$\mathbf{\Phi}_{\mathbf{q}}(\mathbf{q})\ddot{\mathbf{q}} = \boldsymbol{\tau}(\mathbf{q}, \dot{\mathbf{q}})$$

- Equations Of Motion

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}(\mathbf{q})\boldsymbol{\lambda} = \mathbf{Q}^{\mathrm{A}}(t, \mathbf{q}, \dot{\mathbf{q}})$$

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 $\ddot{\mathbf{v}}$:

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

$$\hat{\mathbf{M}} = \mathbf{M}^{\mathbf{vv}} - \mathbf{M}^{\mathbf{vu}}\boldsymbol{\Phi}_{\mathbf{u}}^{-1}\boldsymbol{\Phi}_{\mathbf{v}} - \boldsymbol{\Phi}_{\mathbf{v}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathbf{u}}^{-\mathrm{T}}(\mathbf{M}^{\mathbf{uv}} - \mathbf{M}^{\mathbf{uu}}\boldsymbol{\Phi}_{\mathbf{u}}^{-1}\boldsymbol{\Phi}_{\mathbf{v}})$$

$$\hat{\mathbf{Q}} = \mathbf{Q}^{\mathbf{v}} - \boldsymbol{\Phi}_{\mathbf{v}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathbf{u}}^{-\mathrm{T}}\mathbf{Q}^{\mathbf{u}} - (\mathbf{M}^{\mathbf{vv}} - \boldsymbol{\Phi}_{\mathbf{v}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathbf{u}}^{-\mathrm{T}}\mathbf{M}^{\mathbf{uu}})\boldsymbol{\Phi}_{\mathbf{u}}^{-1}\boldsymbol{\tau}$$

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)

$$\begin{aligned}\boldsymbol{\Psi}_{\ddot{\mathbf{v}}} = {} & \mathbf{M}^{\mathbf{vv}} + \mathbf{M}^{\mathbf{vu}}\mathbf{H} + \mathbf{H}^{T}(\mathbf{M}^{\mathbf{uv}} + \mathbf{M}^{\mathbf{uu}}\mathbf{H}) + \gamma h(\mathbf{M}^{\mathbf{vu}}\mathbf{N} + \mathbf{H}^{T}\mathbf{S} - \mathbf{Q}^{\mathbf{v}}_{\dot{\mathbf{v}}} - \mathbf{Q}^{\mathbf{v}}_{\dot{\mathbf{u}}}\mathbf{H}) \\ & + \beta h^{2}[(\mathbf{M}^{\mathbf{v}}\ddot{\mathbf{q}})_{\mathbf{u}}\mathbf{H} + (\mathbf{M}^{\mathbf{v}}\ddot{\mathbf{q}})_{\mathbf{v}} + \mathbf{M}^{\mathbf{vu}}\mathbf{L} + \mathbf{H}^{T}\mathbf{R} + (\boldsymbol{\Phi}^{T}_{\mathbf{v}}\boldsymbol{\lambda})_{\mathbf{u}}\mathbf{H} + (\boldsymbol{\Phi}^{T}_{\mathbf{v}}\boldsymbol{\lambda})_{\mathbf{v}} \\ & - \mathbf{Q}^{\mathbf{v}}_{\mathbf{u}}\mathbf{H} - \mathbf{Q}^{\mathbf{v}}_{\mathbf{v}} - \mathbf{Q}^{\mathbf{v}}_{\dot{\mathbf{u}}}\mathbf{J}]\end{aligned}$$

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

$$\Psi_{\begin{bmatrix}\ddot{\mathbf{q}}\\ \lambda\end{bmatrix}} = \begin{bmatrix} \mathbf{M} - \gamma h \mathbf{Q}^{A}_{\dot{\mathbf{q}}}\hat{\mathbf{H}} + \beta h^2 \{[(\mathbf{M}\ddot{\mathbf{q}})_{\mathbf{q}} + (\mathbf{\Phi}^{T}_{\mathbf{q}}\lambda)_{\mathbf{q}} - \mathbf{Q}^{A}_{\mathbf{q}}]\hat{\mathbf{H}} - \mathbf{Q}^{A}_{\dot{\mathbf{q}}}\hat{\mathbf{J}}\} & \mathbf{\Phi}^{T}_{\mathbf{q}} \\ \mathbf{\Phi}_{\mathbf{q}} - \gamma h \boldsymbol{\tau}_{\dot{\mathbf{q}}}\hat{\mathbf{H}} + \beta h^2 \{[(\mathbf{\Phi}_{\mathbf{q}}\ddot{\mathbf{q}})_{\mathbf{q}} - \boldsymbol{\tau}_{\mathbf{q}}]\hat{\mathbf{H}} - \boldsymbol{\tau}_{\dot{\mathbf{q}}}\hat{\mathbf{J}}\} & \mathbf{0} \end{bmatrix}$$

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

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 $\rightarrow$ Cartesian Representation
  - 1:2 $\rightarrow$ 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

(*DESCRIPTOR FORM METHOD*-BASED ALGORITHMS)

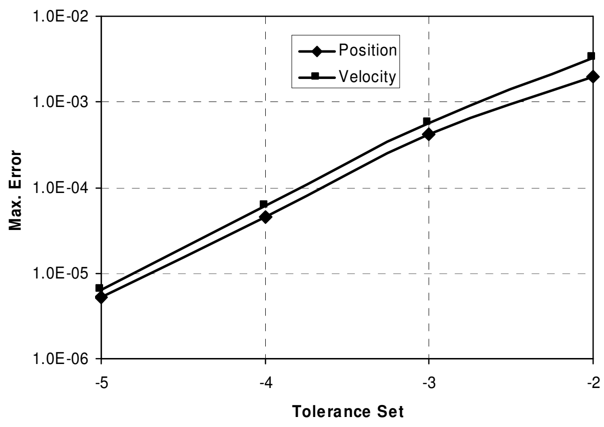
- Validation Done For *InflTrap* And *InflSDIRK* Using HMMWV14 Model.
- Reference Solution Obtained With *ForSDIRK* With Integration Tolerance Set To 1.E-8
- Largest Error Over All Grid Points Reported As

$$\Delta^{(k)} \equiv \max_{1 \le i \le n} | E_i^* - e_i |$$

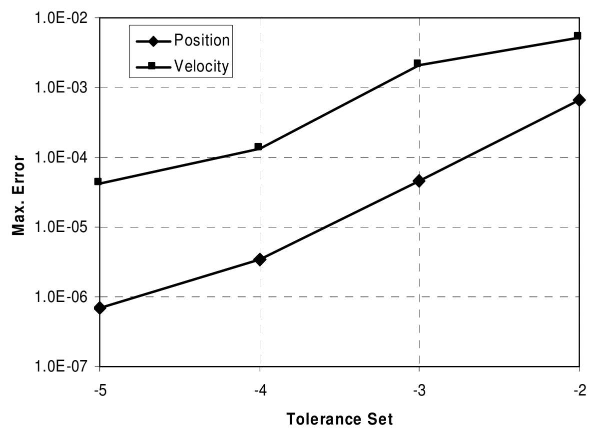
- Reference Solution At Off-Grid Points Obtained Via Cubic Spline Interpolation
- Simulation Length - 2 Seconds. Time Steps Taken (Reference Solution) - 579,779

ALGORITHM VALIDATION
(Contd.)

InflTrap		
Tol. (*k*)	Max. Error Position	Max. Error Velocity
-2	0.001987670207	0.003299978287
-3	0.000412496160	0.000570435797
-4	0.000044509076	0.000061351047
-5	0.000005203414	0.000006412237



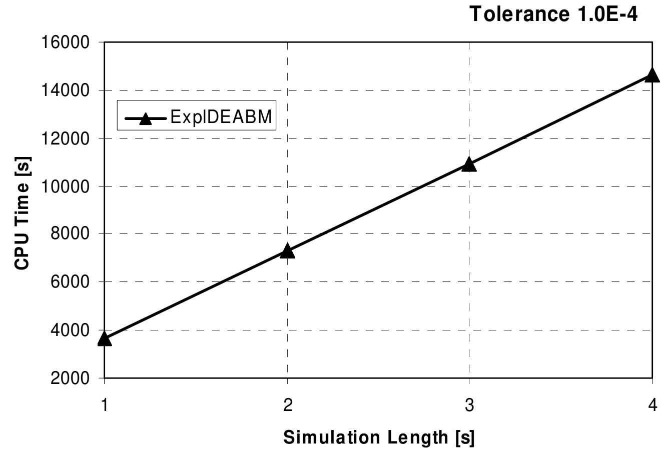
InflSDIRK		
Tol. (*k*)	Max. Error Position	Max. Error Velocity
-2	0.000657081652	0.005078716955
-3	0.000045498338	0.002071677946
-4	0.000003370069	0.000134605346
-5	0.000000696890	0.000041544247



EXPLICIT vs. IMPLICIT COMPARISON

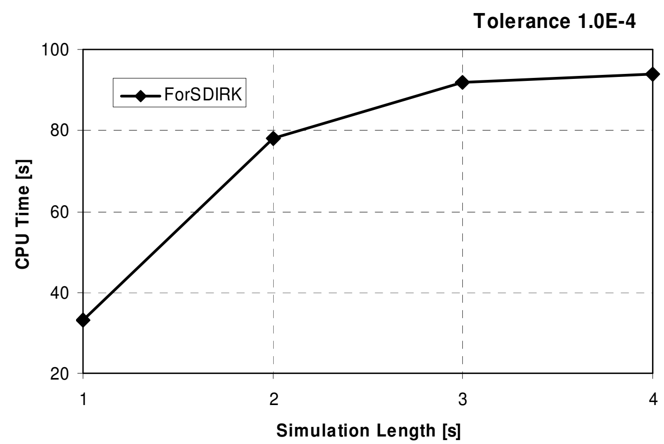
ExplDEABM				
TOL	*10^{-2}*	*10^{-3}*	*10^{-4}*	*10^{-5}*
1 s	3618	3641	3667	3663
2 s	7276	7348	7287	7276
3 s	10865	11122	10949	10965
4 s	14480	14771	14630	14592

CPU Results Explicit Integration [seconds]

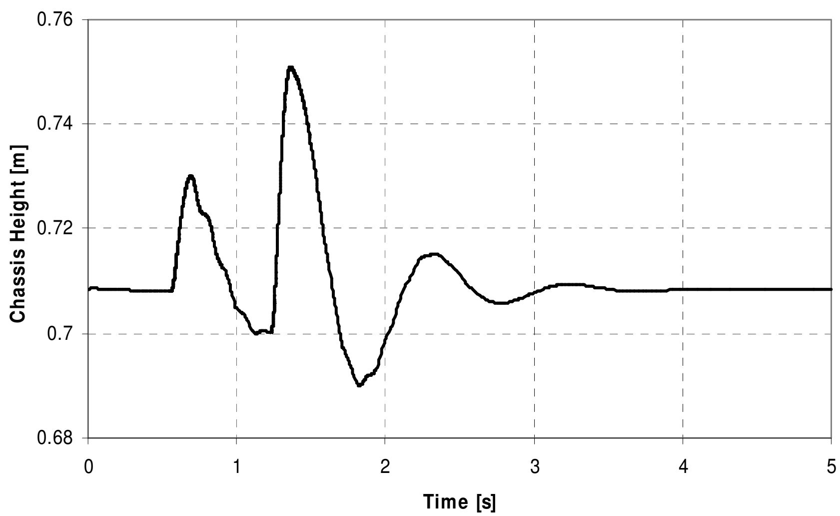


ForSDIRK				
TOL	*10^{-2}*	*10^{-3}*	*10^{-4}*	*10^{-5}*
1 s	10.1	21	33	57
2 s	25.3	49	78	139
3 s	29.5	58	92	166
4 s	30	61	94	184

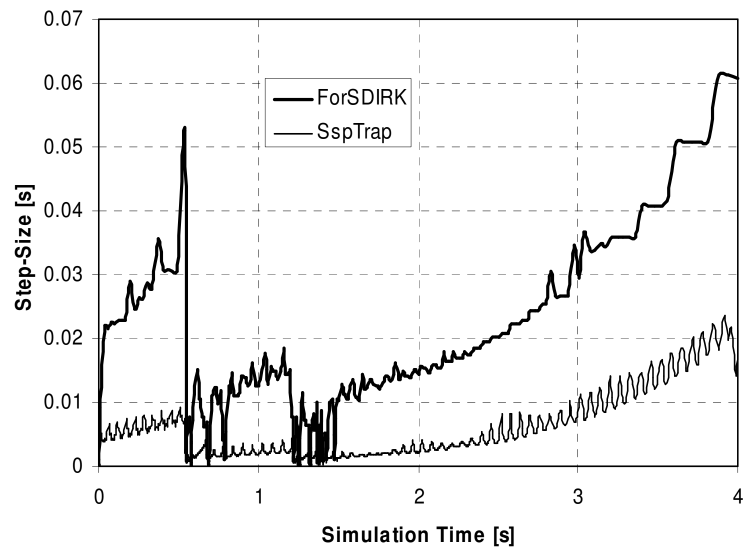
CPU Results *ForSDIRK* [seconds]



EXPLICIT vs. IMPLICIT COMPARISON (Contd.)



Chassis Height HMMWV14



Step-Size History:
ForSDIRK and *InflTrap*

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