Technical Report 2009-02

A Cosimulation Approach for the Time Analysis of a Biomechanically-Inspired System with Rigid and Deformable Bodies

Anne Schmitz[1], Arman Pazouki[2], Darryl Thelen[1], Dan Negrut[2]

[1] UW Neuromuscular Biomechanics Lab, University of Wisconsin Madison, Madison, WI
[2] Simulation Based Engineering Laboratory, University of Wisconsin, Madison, WI

March 23, 2009

Abstract

The goals of this study were to develop a simplified finite element knee model, cosimulate this model with the motion of knee flexion/extension to demonstrate the concept of cosimulation, and choose an integrator to maximize the efficiency of the cosimulation process. A simplified model of the knee was used to simulate flexion/extension in FEBio. The femur was modeled as a fixed, rigid body, the knee complex soft tissue, and the tibia/fibula complex as a rigid body with prescribed motion to rotate about the knee center. Both a fourth order Runge-Kutta (RK4) integrator and Hilber-Hughes-Taylor (HHT) integrator were used to solve for rigid body dynamics in Matlab. Both the RK4 and HHT integrators produced the same angular position time trace for each damping situation but did not use the same time steps. For the underdamped system, RK4 was faster. However, as damping ratio increased, RK4 needed smaller step sizes to maintain stability and therefore became slower that HHT.

Contents

1 Introduction

Many gait simulation/motion capture techniques have been established with simplified knee joint models (Anderson and Pandy, 1999; Anderson and Pandy, 2001; Delp, et al., 2007; Delp, et al., 1990; Pandy, 2001; Pandy, 2003; Sun and Metaxas, 2001; Thelen and Anderson, 2006; Thelen, et al., 2003). However, it is not sufficient to use a simplified joint model where configuration is load dependent. Mathematical knee models (Beillas, et al., 2004; Caruntu and Hefzy, 2004; Dhaher and Kahn, 2002; Donahue, et al., 2002; Hirokawa and Tsuruno, 2000; Li, et al., 2001; Sun and Dhaher, 2006) have also been developed but independently of gait simulation. It seems intuitive to use gait simulation to solve for muscle actuations that would output a desired gait motion and feed these into the mathematical knee model to analyze stresses and deformations during gait. However, this poses a problem because the joint configuration changes with each muscle force and is especially sensitive to hamstring activation (Harner, et al., 1998; Kwak, et al., 2000; Li, et al., 1999). Cosimulating a finite element analysis of the knee and gait kinetics/kinematics can provide a more accurate and detailed model of how the knee performs during gait. Cosimulation requires modeling geometry and soft tissue mechanics in a finite element software and solving rigid body equations of motion with numerical integrators.

A wide range of numerical integrators exists; these integrators are commonly classified as explicit or implicit. In implicit methods, the value of a general coordinate at a time step depends on its value at previous steps and its current value. Therefore, these kinds of methods require matrix algebra at each step, thus increasing the computational efforts. However, convergence occurs faster in terms of the total number of steps. Moreover, these methods are stable for larger time steps with respect to their explicit counterparts. In explicit methods, the value of a general coordinate at a time step depends only on the previous steps. Given an initial condition, these methods can be solved autonomously until a desired time. These methods tend to be faster at each step compared to implicit methods but may diverge and increase CPU time due to smaller time steps.

The objective of this study was to demonstrate cosimulation on a simple knee model. Numerically integrating the set of differential equations required in cosimulation first involves picking a proper integrator for the problem at hand. A detailed investigation and comparison of the efficiency between implicit and explicit integrators is essential to maximize cosimulation performance and make this simulation practical. The Hilber-Hughes-Taylor (HHT) method (implicit integrator) (Negrut, et al., 2007) and fourth order Runge-Kutta method (explicit integrator) (Butcher, 2000) were investigated since these are commonly used in practice and simple to implement.

2 Cosimulation Method

Figure 1 illustrates the process of cosimulation. The appendix shows Matlab code used to implement this simple model with no damping and using a Matlab integrator (ODE45) and fourth order Runge-Kutta.

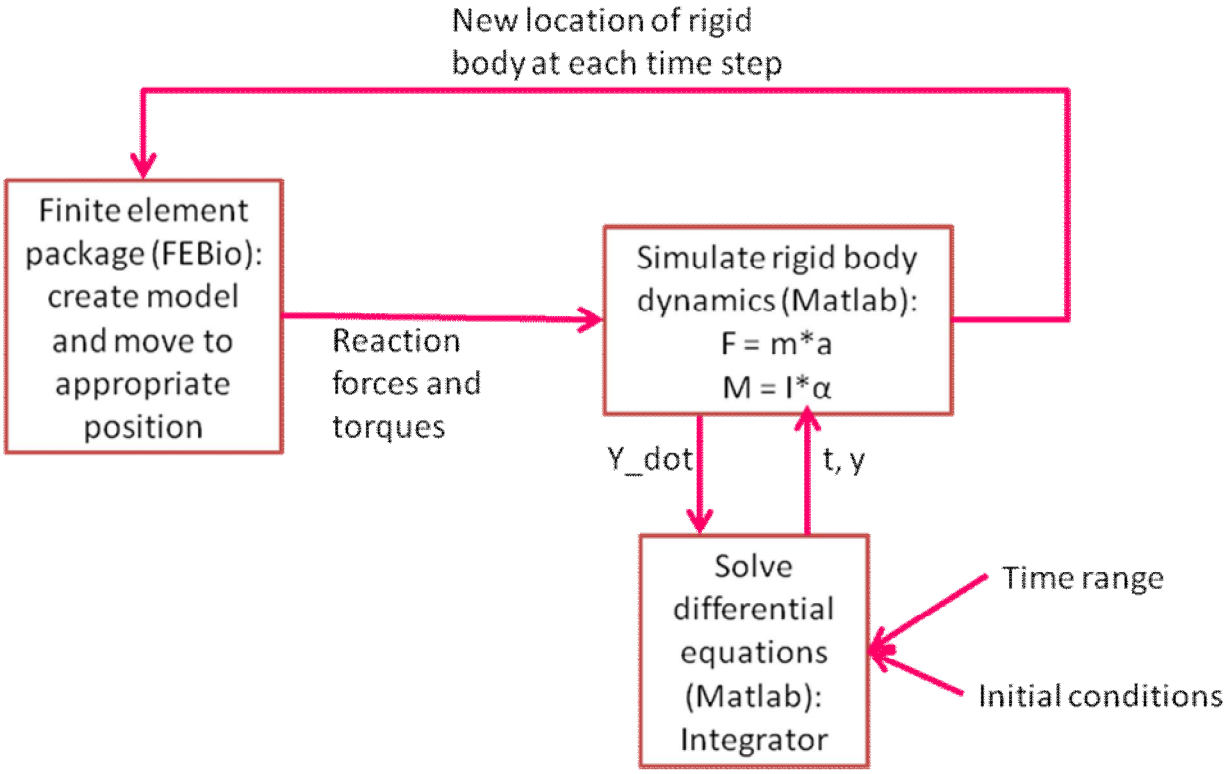

Figure 1: This flowchart shows the basic methodology of cosimulation.

FEBio (FEBio v1.1, Salt Lake City, Utah) was used to create a finite element model of the knee (Figure 2). The femur was modeled as a fixed, rigid body and the tibia/fibula complex as a rigid body with freedom to rotate about the knee center in flexion/extension only. The knee complex was composed of transversely isotropic, Mooney-Rivlin tissue with parameters from Gardiner and Weiss for a medial collateral ligament (Gardiner and Weiss, 2003). Figure 3 shows the orientation of the fibers for the soft tissue and Figure 4 the parameters used.

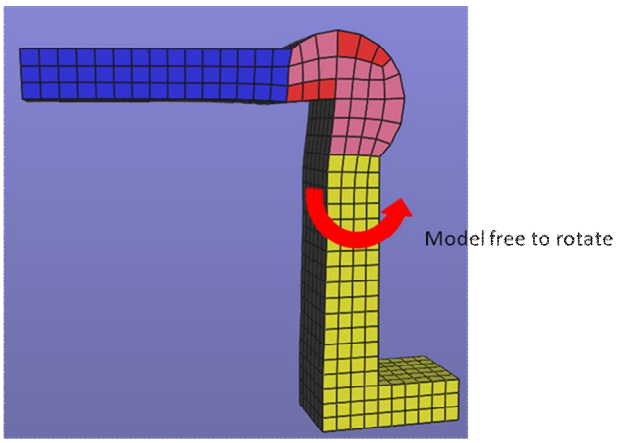

Figure 2: A simple knee model was created in FEBio.

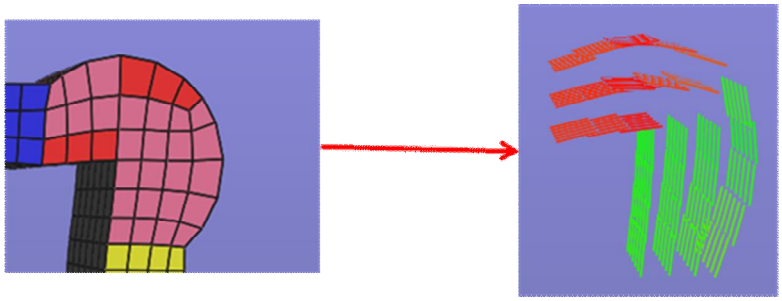

Figure 3: The soft tissue of the knee was composed of fibers in a specified direction.

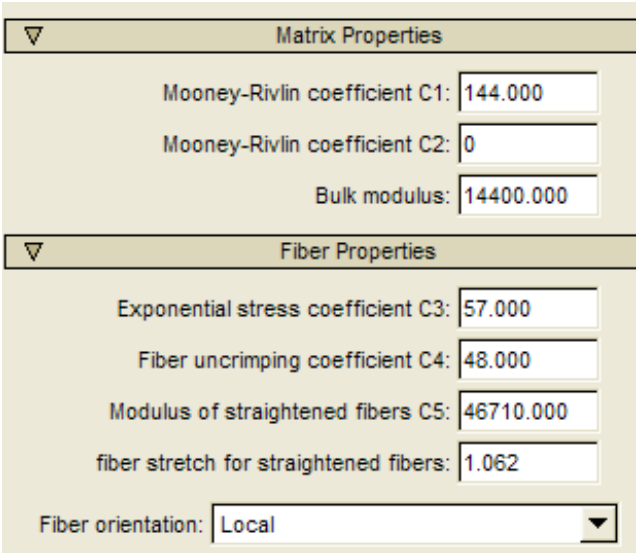

Figure 4: The soft tissue was modeled as isotropic, Mooney-Rivlin tissue with paramters from Gardiner and Weiss for a medial collateral ligament (Gardiner and Weiss, 2003).

Using both implicit and explicit methods of integration, the rigid body dynamics were simulated in Matlab (2007a, The MathWorks, Natick, MA). First, the knee was initially moved 1.0 radian in FEBio and the torque about the knee measured. For a 1.0 radian displacement, a torque resulted in 23 Nm, which seems reasonable (Ferrari, et al., 2008). This torque was then used in the equation of motion of the shank to solve for the new angular position of the shank at the next time point. Inertial properties were taken from deLava assuming a 63.5 kg subject (the size of a female graduate student) (de Leva, 1996). The FEBio model was then rebuilt with this new angular position to obtain a new torque output. This process was iterated for the desired time period.

3 Quantitative Analysis of the Numerical Integrators

The applicability of some integration methods is discussed in appendix A. Here, the accuracy and applicability of RK4—as an explicit method—and HHT—as an implicit method—will be discussed more quantitatively. The equation of motion for this model is shown in Equation (1). This model was treated as a 2D-planar problem with only flexion/extension motion considered.

$$I\ddot{\theta} = M(\theta) - c\dot{\theta} + F(t) \tag{1}$$

In this equation, $M(\theta)$ is moment, c is damping factor, and $F(t)$ is the excitation force. The fourth order Runge-Kutta (RK4) and Hilber-Hughes-Taylor (HHT) integrators were applied using different damping ratios, ζ, of 0.50, 2.00, and 100.00. The initial values of position and velocity were 0.1 and 0.0, respectively. Moreover, $F(t)$ was chosen (**Error! Reference source not found.**) to generate a harmonic motion with an amplitude of $\sim 0.02\sin(80t)$.

$$F(t) = 12.31\sin(80t) \tag{2}$$

Both the RK4 and HHT integrators produce the same angular position time trace for each damping situation (Figure 5 through Figure 7). However, these results were not derived for the same time steps in both integrators.

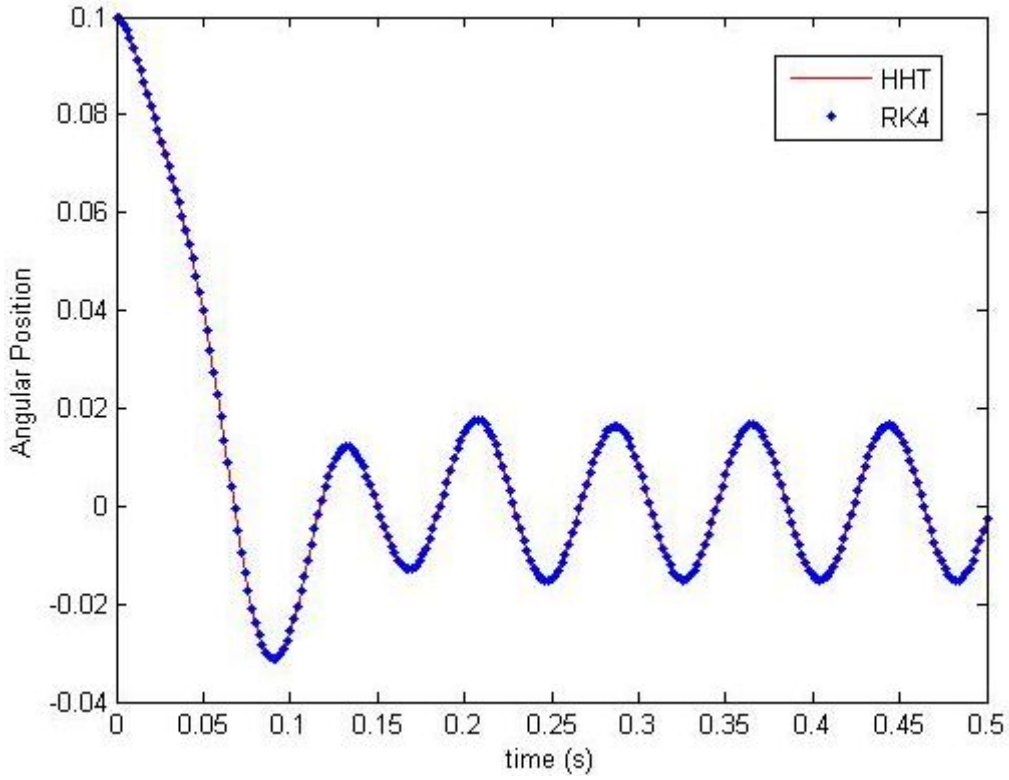

Figure 5: HHT and RK4 results for the underdamped system with $\zeta = 0.5$.

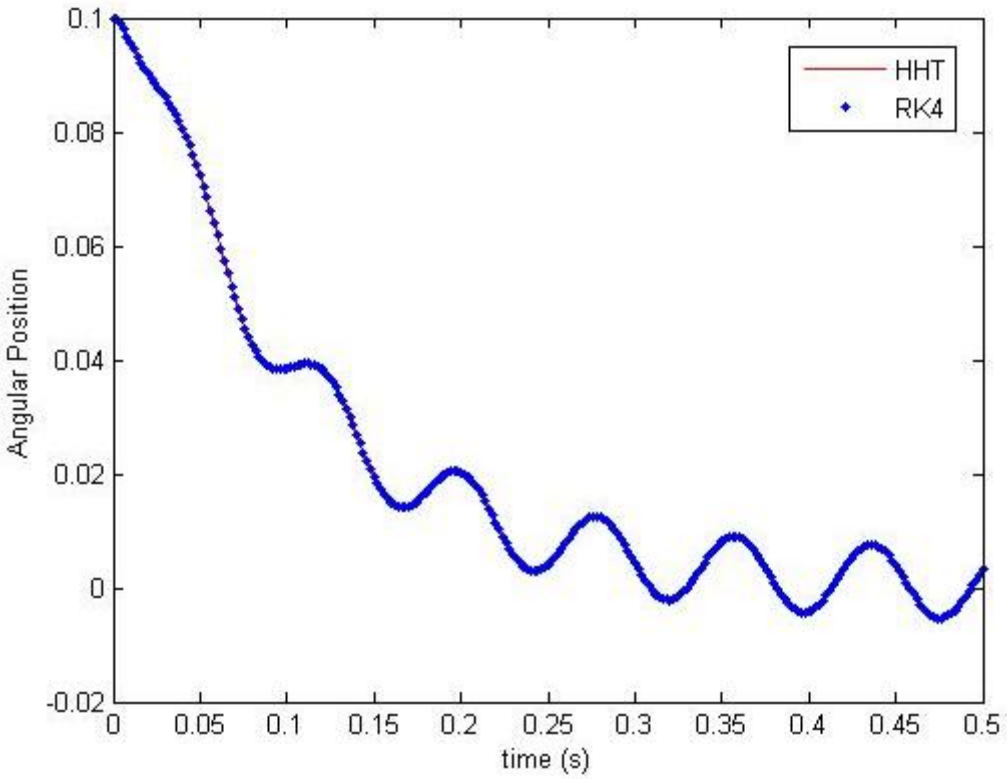

Figure 6: HHT and RK4 results for the overdamped system with $\zeta = 2.0$.

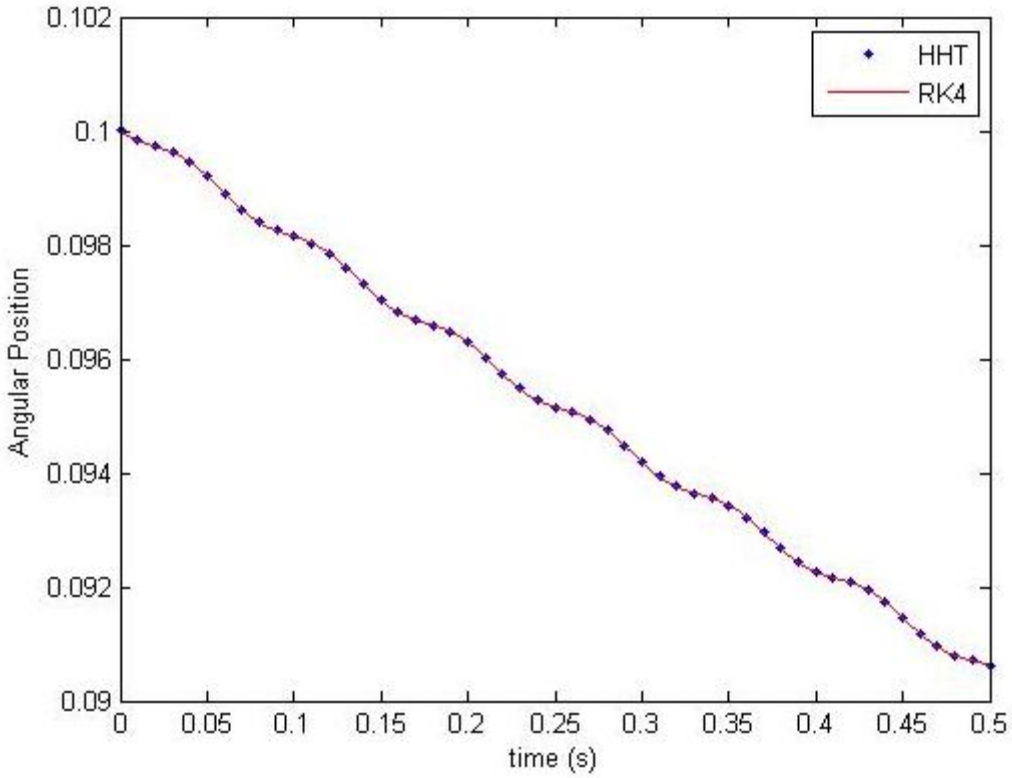

Figure 7: HHT and RK4 results for the overdamped system with $\zeta = 100.00$.

3.1 CPU Time Analysis

The time traces in Figure 5 through Figure 7 were obtained with different time steps for each integrator. Therefore, the CPU times of the two integrators may differ greatly. To calculate the CPU time of the two cases for an underdamped system with a damping ratio of 0.50, Δt of the RK4 method was chosen as 0.002 sec and 0.0002 sec for the HHT method to obtain the same accuracy (which will be discussed in more detail later). Figure 8 shows both integrators produce the same time trace of angular position, but the HHT method took 10.7 times longer.

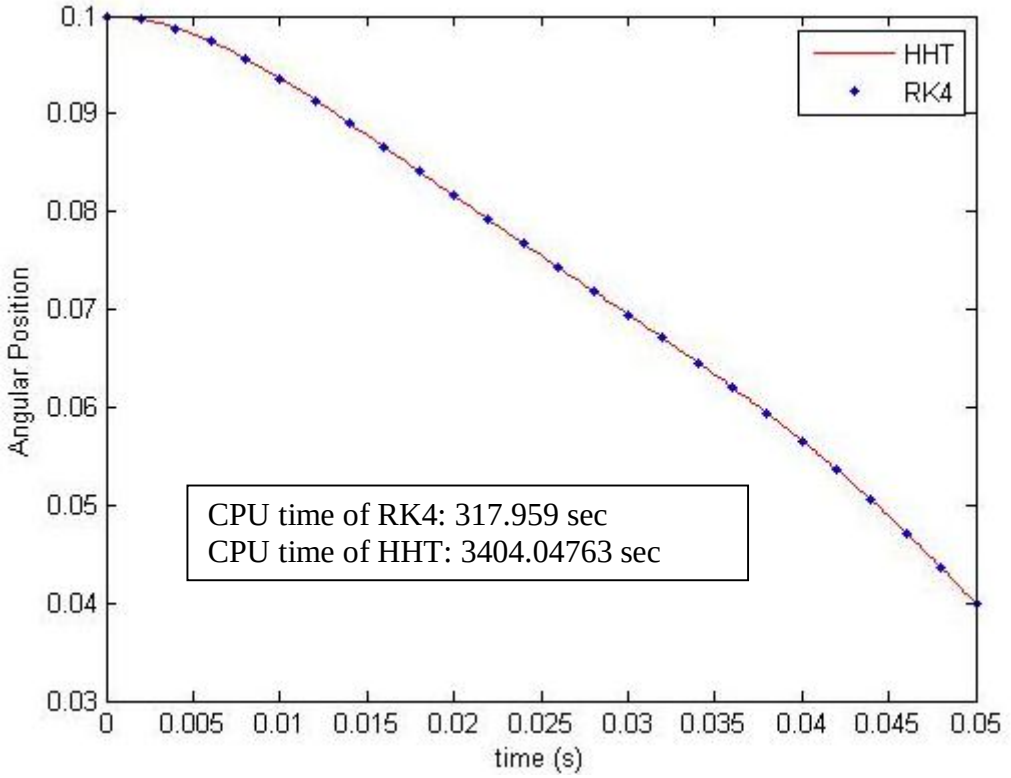

Figure 8: HHT and RK4 results for the underdamped system with $\zeta = 0.50$.

For a damping ratio of 100.00, the time step for HHT was 0.01 sec and 0.0003 sec for RK4. Any time steps larger than 0.0003 sec caused RK4 to diverge. The CPU times were computed as 2.3513e4 sec for RK4 and 8.804e2 sec for HHT. RK4 took 26.7 times longer to compute the solution.

By increasing the damping ratio, RK4 could not compete with HHT. In these overdamped cases, smaller step sizes should be chosen for RK4 to maintain stability and thus increases the CPU time. For the underdamped system, although HHT converged to the solution as well as RK4, it becomes weaker than RK4 due to the iterative process which HHT requires.

3.2 Error Analysis

3.2.1 Case 1: $\zeta = 0.5$

The HHT method was more sensitive to step size than the RK4. Table 1, and Table 2 present the quantitative comparison of the methods. Figure 9 and Figure 10 present the order analysis of the HHT method at $\zeta = 0.5$.

Table 1: Error values of HHT method for $\zeta = 0.5$ and different time step sizes

HHT		$\Delta t = 0.0005$	$\Delta t = 0.001$	$\Delta t = 0.002$	$\Delta t = 0.004$	$\Delta t = 0.008$
Position	Maximum Error	6.0e-6	2.5e-5	9.8e-5	3.94e-4	1.51e-3
	Maximum Error Regardless of Start	5.0e-6	2.0e-5	8.2e-5	3.27e-4	1.29e-3
	Error at the end	1.0e-6	4.0e-6	1.9e-5	7.7e-5	1.16e-4
Velocity	Maximum Error	2.84e-4	1.14e-3	4.62e-3	1.85e-2	7.09e-2
	Maximum Error Regardless of Start	2.50e-4	9.88e-4	3.95e-3	1.57e-2	6.24e-2
	Error at the end	2.19e-4	8.76e-4	3.50e-3	1.39e-2	6.17e-2

Table 2: Error values of RK4 method for $\zeta = 0.5$ and different time step sizes

RK4		$\Delta t = 0.0005$ (reference set)	$\Delta t = 0.001$	$\Delta t = 0.002$	$\Delta t = 0.004$	$\Delta t = 0.008$
Position	Maximum Error	0	1.0e-6	1.0e-6	4.0e-6	2.7e-5
	Maximum Error Regardless of Start	0	1.0e-6	1.0e-6	2.0e-6	1.5e-5
	Error at the end	0	0	0	0	5.0e-6
Velocity	Maximum Error	0	1.2e-5	5.5e-5	2.1e-4	1.29e-3
	Maximum Error Regardless of Start	0	1.1e-5	1.3e-5	8.2e-5	7.66e-4
	Error at the end	0	0	2.0e-6	7.0e-6	1.31e-4

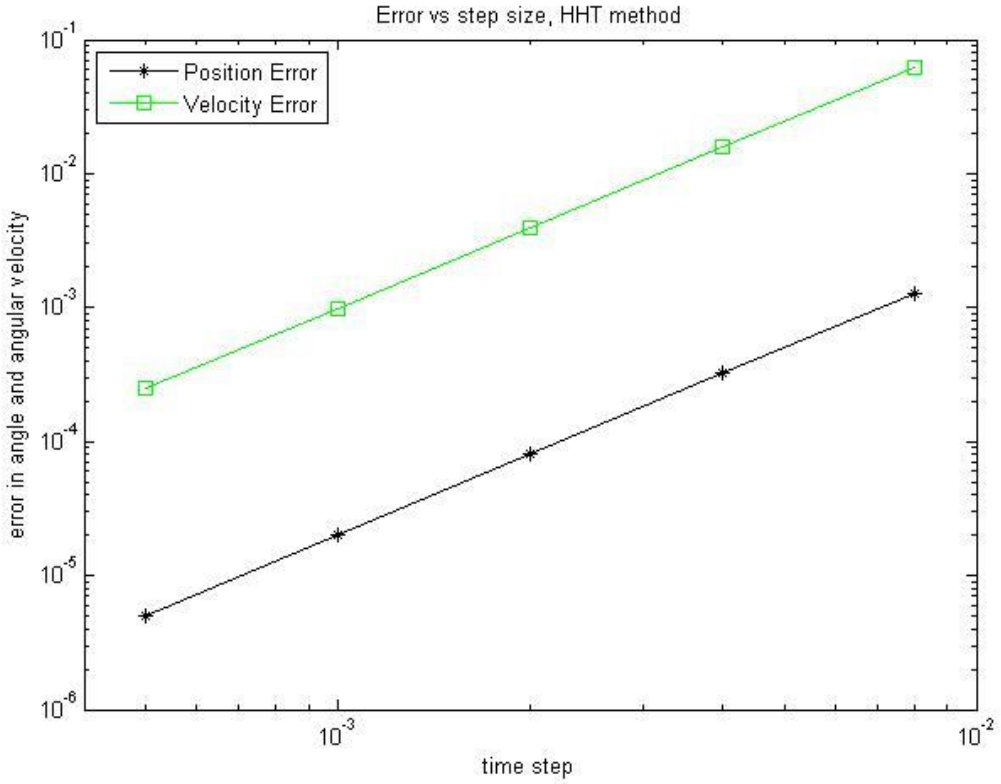

Figure 9: HHT order analysis $\zeta = 0.5$

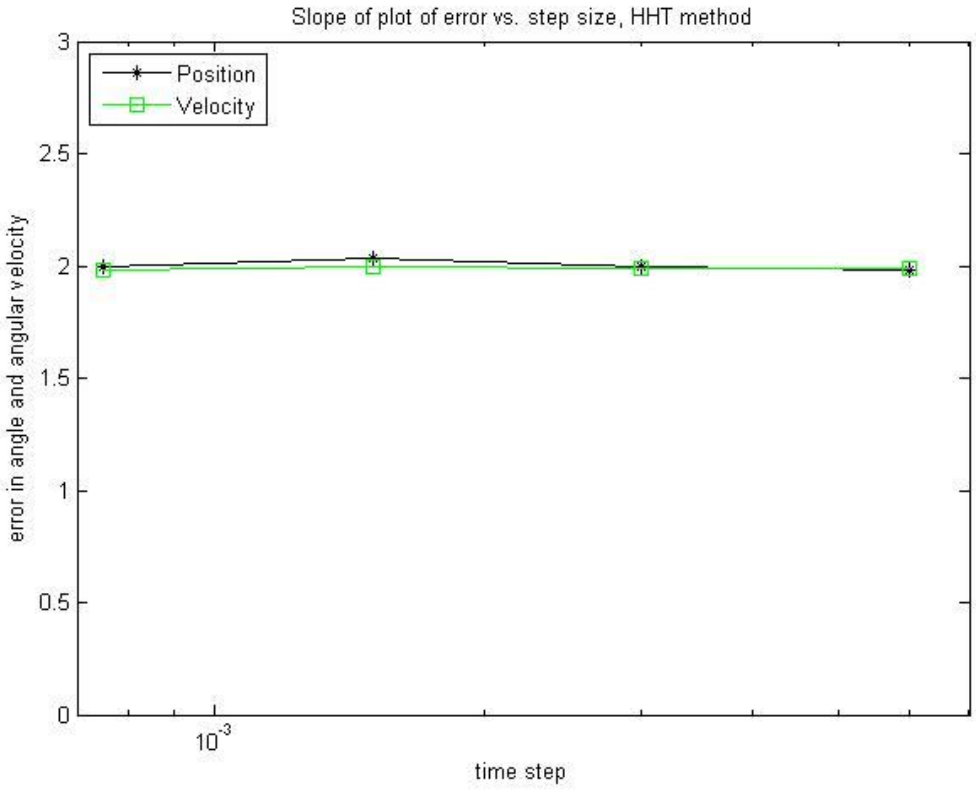

Figure 10: HHT convergence order $\zeta = 0.5$

3.2.2 Case 2: $\zeta = 100.0$

Increasing the damping caused RK4 to diverge (Figure 11, Table 3, and Table 4).

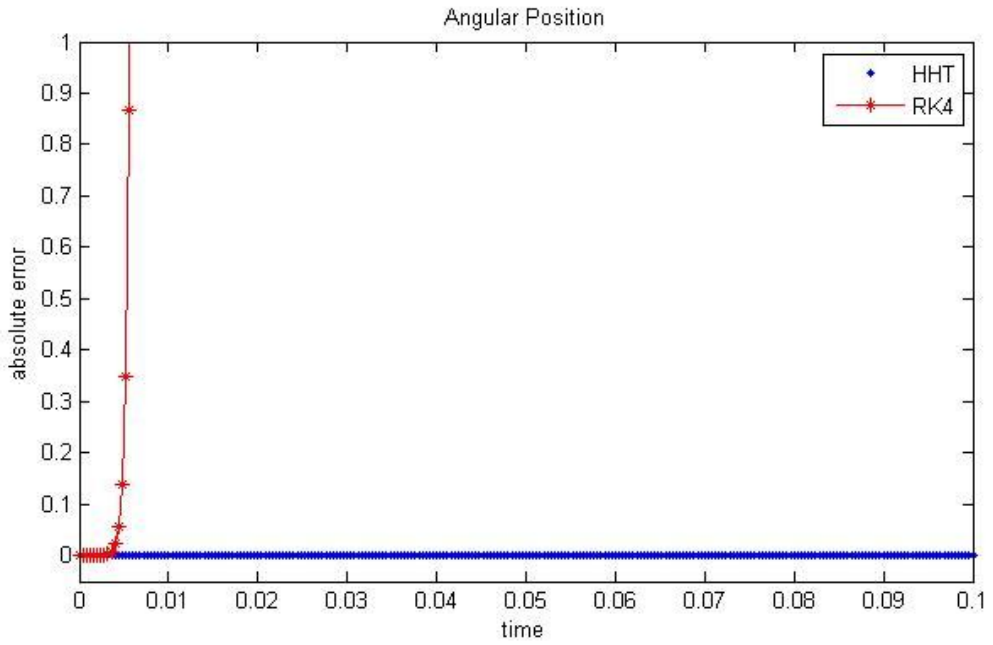

Figure 11: Comparison of the position errors of HHT and RK4 methods for $\zeta = 100.0$ and time step size $\Delta t = 0.0004$

Table 3: Error values of HHT method for $\zeta = 100.0$ and different time step sizes

HHT		$\Delta t = 0.0001$ (reference data)	$\Delta t = 0.0002$	$\Delta t = 0.0003$	$\Delta t = 0.0004$	$\Delta t = 0.01$
position	Maximum Error	**0**	**0**	**1.0e-6**	**1.0e-6**	**1.6e-5**
	Maximum Error Regardless of Start	**0**	**0**	**0**	**0**	**1.0e-5**
	Error at the end	**0**	**0**	**0**	**0**	**1.0e-5**
velocity	Maximum Error	**0**	**1.7e-3**	**3.8e-3**	**5.8e-3**	**1.9e-2**
	Maximum Error Regardless of Start	**0**	**1.0e-6**	**1.0e-6**	**1.0e-6**	**1.6e-2**
	Error at the end	**0**	**0**	**0**	**0**	**1.2e-2**

Table 4: Error values of RK4 method for $\zeta = 100.0$ and different time step sizes

RK4		$\Delta t = 0.0001$	$\Delta t = 0.0002$	$\Delta t = 0.0003$	$\Delta t \geq 0.0004$
position	Maximum Error	**0**	**0**	**1.0e-6**	**Diverges**
	Maximum Error Regardless of Start	**0**	**0**	**0**	
	Error at the end	**0**	**0**	**0**	
velocity	Maximum Error	**5.6e-4**	**2.3e-3**	**1.3e-2**	
	Maximum Error Regardless of Start	**1.0e-6**	**1.0e-6**	**4.0e-6**	
	Error at the end	**0**	**0**	**4.0e-6**	

Figure 12 and Figure 13 present the order analysis of HHT method at $\zeta = 100.0$.

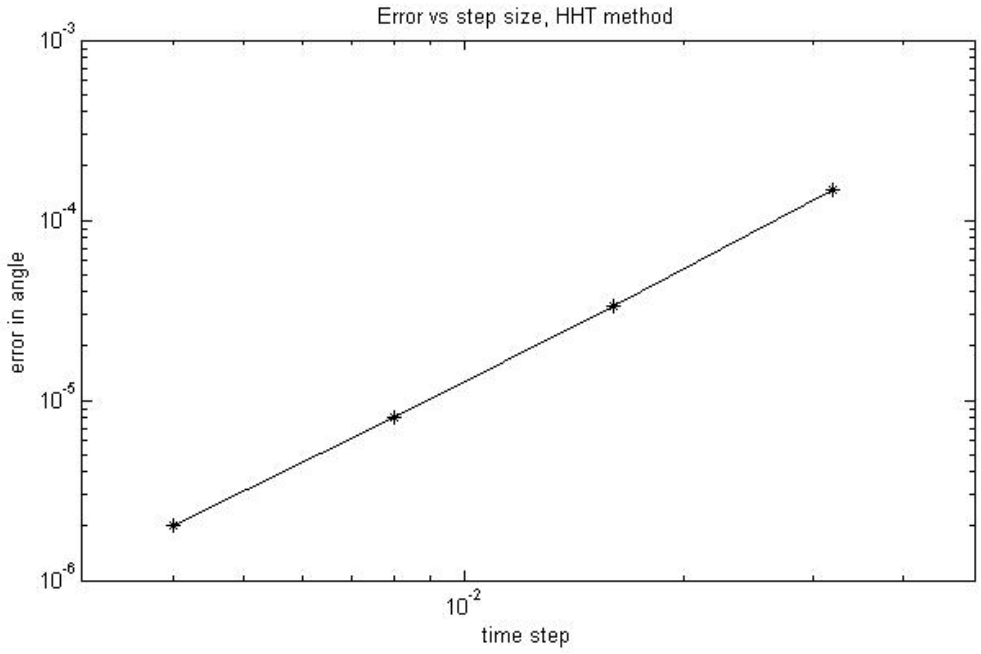

Figure 12: HHT order analysis $\zeta = 100.0$

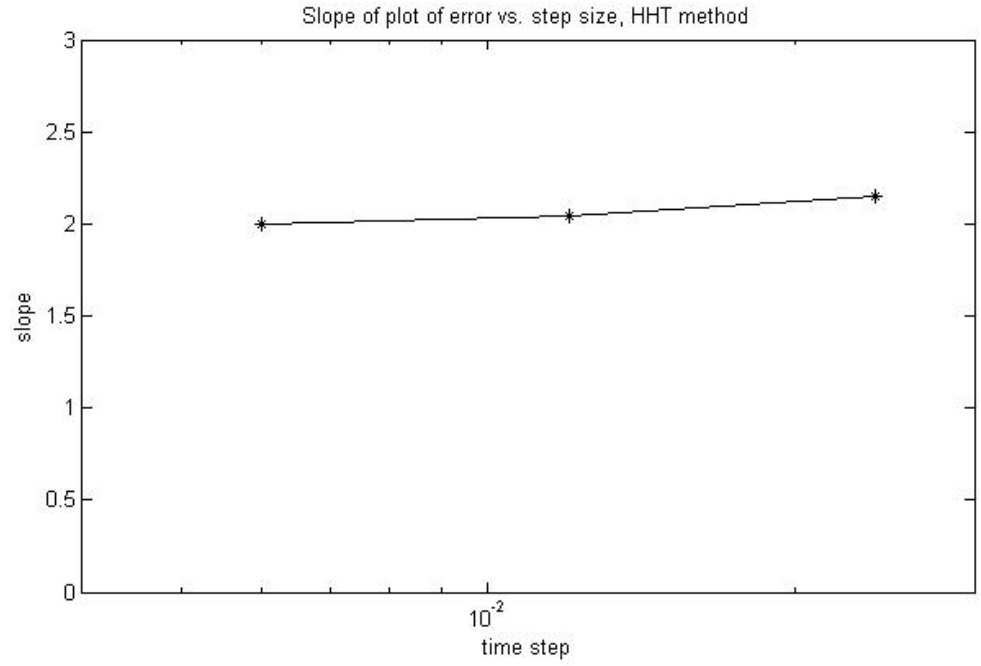

Figure 13: HHT convergence order $\zeta = 100.0$

For overdamped systems, HHT remained stable, even for big step sizes. However, for big step sizes at the beginning of the integration, although the position did not have considerable errors (Figure 14) but the velocity response showed large errors (Figure 15), with these errors eventually disappearing. In the HHT method, a combination of

velocities and accelerations of the previous and current time steps are used to calculate the position. So the value of the velocities and accelerations at a time step are not as important as the combination values.

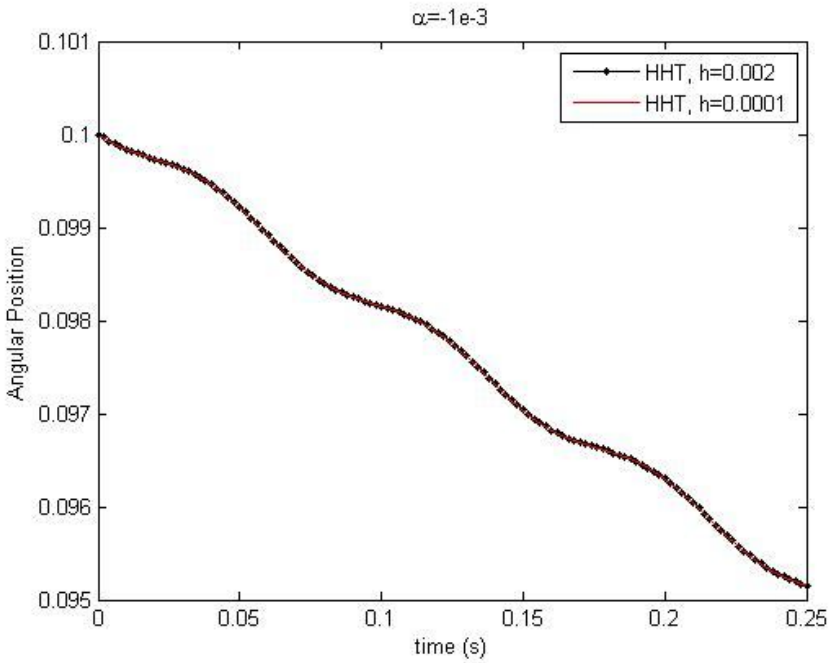

Figure 14: Angular position for $\alpha = -0.001$ (controls the numerical damping of the method).

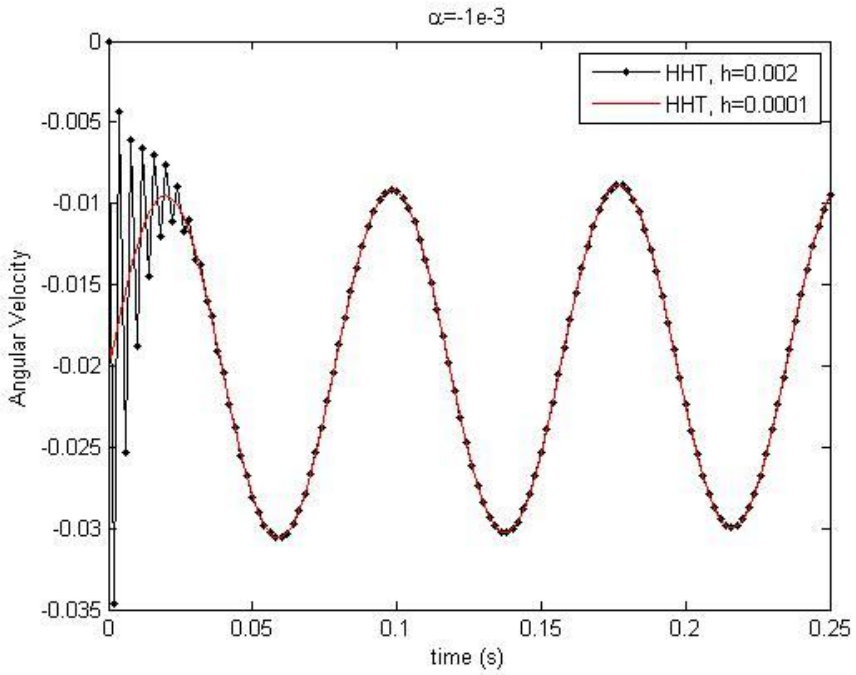

Figure 15: Angular velocity for $\alpha = -0.001$ (controls the numerical damping of the method).

The most conservative solution to this problem is to choose small values as the time step. However, there are two better solutions. First of all, the value of α, which controls the numerical damping of the method, can be tuned during the solution. For example, at the beginning, large values can be chosen for α, which increases the influence of the current velocities and accelerations (Figure 16 and Figure 17). The other solution is to solve the problem choosing different time steps at different time intervals.

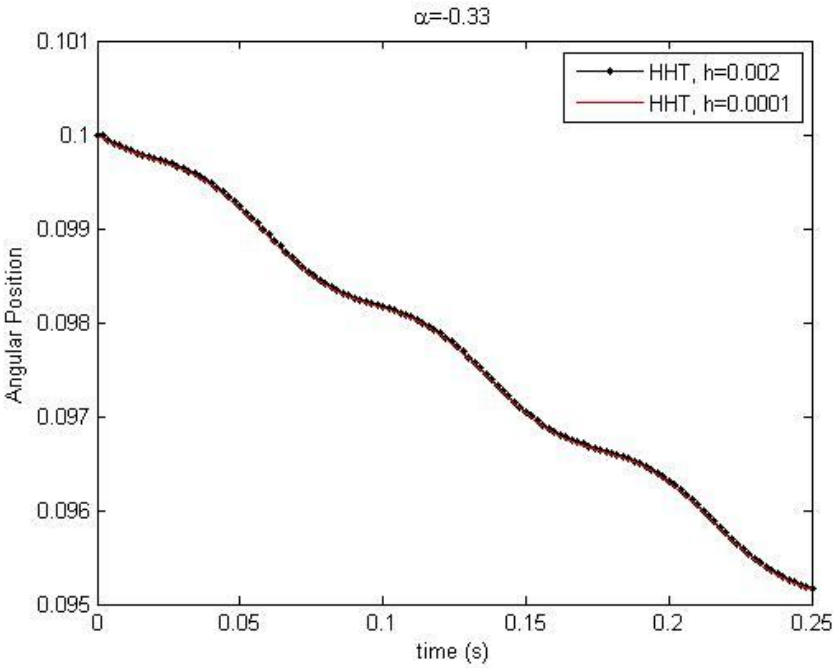

Figure 16: Angular position for $\alpha = -0.33$

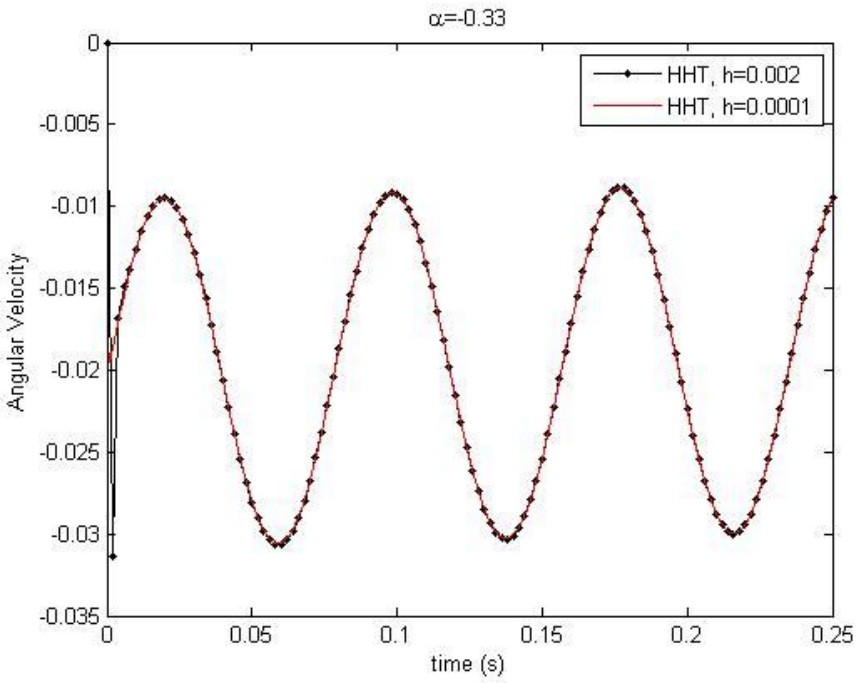

Figure 17: Angular velocity for $\alpha = -0.33$

Appendix A: integration Methods

A-1 Implicit methods

In some methods, the discretization is done in a way that the value of the variable depends on not only the previous steps, but also its current values. Therefore these kinds of methods require some matrix algebra at each step, which increases the computational efforts. But convergence occurs faster in terms of the total number of steps. Moreover they are stable for larger time steps than their explicit counterparts.
Among the different implicit methods, here we present the Hilber-Hughes-Taylor (HHT) method.

A.1.1 Hilber-Hughes-Taylor

The Hilber-Hughes-Taylor (HHT) method is an implicit method that has been used successfully in the field of classical mechanical simulation. Equation (A-1) shows how the positions, velocities and accelerations are updated at every time step.

$$\begin{aligned} Ma_{n+1} &= (1+\alpha)F_{n+1} - \alpha F_n \\ v_{n+1} &= v_n + \Delta t((1-\gamma)a_n + \gamma a_{n+1}) \\ r_{n+1} &= r_n + \Delta t v_n + \frac{\Delta t^2}{2}((1-2\beta)a_n + 2\beta a_{n+1}) \end{aligned} \qquad \textbf{(A-1)}$$

In this equation, $\gamma = \frac{1-2\alpha}{2}$ and $\beta = (\frac{1-\alpha}{2})^2$. The parameter α controls the amount of numerical dissipation that is introduced and is bounded by $\alpha \in [-\frac{1}{3}, 0]$. For $\alpha = 0$ this method becomes the well known trapezoidal method.

Using implicit methods requires the solution of a coupled set of nonlinear equations. One popular approach to solving such a system numerically is the family of methods known as Newton-like methods. Newton-like methods require the calculation of a Jacobian matrix. Equation (A-2) gives a derivation of the Jacobian for the HHT method.

$$\begin{aligned} \Psi &= Ma_{n+1} - (1+\alpha)F_{n+1} + \alpha F_n \\ J &= \frac{\partial \Psi}{\partial a_{n+1}} = M - (1+\alpha)\frac{\partial F_{n+1}}{\partial r_{n+1}}\frac{\partial r_{n+1}}{\partial a_{n+1}} - (1+\alpha)\frac{\partial F_{n+1}}{\partial v_{n+1}}\frac{\partial v_{n+1}}{\partial a_{n+1}} \end{aligned} \qquad \textbf{(A-2)}$$

So Jacobian can be written as equation (A-3).

$$J = M - (1+\alpha)\beta\Delta t^2 \frac{\partial F_{n+1}}{\partial r_{n+1}} - (1+\alpha)\gamma\Delta t \frac{\partial F_{n+1}}{\partial v_{n+1}} \qquad \textbf{(A-3)}$$

In which r_{n+1} and v_{n+1} were taken from equation (A-1). In practice, for complicated functional forms, numerical differencing must be used to approximate the derivatives.
There are several related ways to advance the simulation given an implicit method. The algorithm employed in our simulations is given in Algorithm 1.

Algorithm 1 Quasi-Newton algorithm: Hilber-Hughes-Taylor

Guess $a_n^{(0)} = a_{n-1}$

Guess $v_n^{(0)}$ (for example: $v_n^{(0)} = v_{n-1} + a_{n-1}\Delta t$)

Guess $r_n^{(0)}$ (for example: $r_n^{(0)} = r_{n-1} + v_{n-1}\Delta t + 0.5 \times \Delta t^2 \times a_{n-1}$)

Compute $\frac{\partial F_n}{\partial r_n}$ and $\frac{\partial F_n}{\partial v_n}$ numerically to obtain J

while $\left\|\Delta a_n^{(k)}\right\| > \varepsilon$ **do**

Solve $J\Delta a_n^{(k)} = -\Psi(r_n^{(k)}, v_n^{(k)}, a_n^{(k)})$

$a_n^{(k+1)} = a_n^{(k)} + \Delta a_n^{(k)}$

Update position and velocities using HHT formulas

end while

A-2 Explicit methods

In explicit methods, the governing equations are discretized in a way that the value of the variable depends only on the previous steps. So giving an initial condition, these kinds of methods can be solved autonomously until a desired step. Since there is no additional time consuming calculation between two steps, these methods are faster at each step; but they should be well treated to avoid divergence.

Among the several explicit methods, many of them are not applicable to our problem, which is of the form of equation (A-4).

$$M\vec{a} = \vec{F}(\vec{r}, \vec{v}; t) \quad \textbf{(A-4)}$$

For example, Velocity Verlet and Basic Verlet can not be implemented without using an iterative loop between two consecutive steps. Consequently, based on the applicability and memory usage, forth order Runge-Kutta method was selected for the current work.

A.2.1 Forth order Runge-Kutta

For the differential equation of (A-5), forth order Runge-Kutta (RK4) method can be applied as shown in equations (A-6).

$$y' = f(t, y) \quad \textbf{(A-5)}$$

$$
\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \\
k_3 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2) \\
k_4 &= f(t_n + h, y_n + hk_3) \\
y_{n+1} &= y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \\
t_{n+1} &= t_n + h
\end{aligned}
\qquad \textbf{(A-6)}
$$

The implementation of this method for the 2^{nd} order differential equations requires the definition of $2n$ phase space variables for the n spatial variables.

References

Anderson, F. C. and Pandy, M. G. 1999. A dynamic optimization solution for vertical jumping in three dimensions. Computer Methods in Biomechanics and Biomedical Engineering 2, 201-231.

Anderson, F. C. and Pandy, M. G. 2001. Dynamic optimization of human walking. Journal of Biomechanical Engineering 123, 381-390.

Beillas, P., Papaioannou, G., Tashman, S. and Yang, K. H. 2004. A new method to investigate in vivo knee behavior using a finite element model of the lower limb. J Biomech 37, 1019-1030.

Butcher, J. C. 2000. Numerical methods for ordinary differential equations in the 20th century. J Comp and App Math 125, 1-29.

Caruntu, D. and Hefzy, M. 2004. 3-D Anatomically Based Dynamic Modeling of the Human Knee to Include Tibio-Femoral and Patello-Femoral Joints. J Biomech Engr 126, 44-53.

de Leva, P. 1996. Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. J Biomech 29, 1223-1230.

Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. and Thelen, D. 2007. OpenSim: Open-source software to create and analyze dynamic simulations of movement. IEEE Transactions on Biomedical-Engineering (in press).

Delp, S. L., Loan, J. P., Hoy, M. G., Zajac, F. E., Topp, E. L. and Rosen, J. M. 1990. An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. IEEE Transactions on Biomedical-Engineering 37, 757-767.

Dhaher, Y. and Kahn, L. 2002. The Effect of Vastus Medialis Forces on Patello-femoral Contact: A Model-based Study. J Biomech Engr 124, 758-767.

Donahue, R., Hull, M., Rashid, M. and Jacobs, C. 2002. A Finite Element Model of the Human Knee Joint for the Study of Tibio-Femoral Contact. J Biomech Engr 124, 273-280.

Ferrari, A., Benedetti, M. G., Pavan, E., Frigo, C., Bettinelli, D., Rabuffetti, M., Crenna, P. and Leardini, A. 2008. Quantitative comparison of five current protocols in gait analysis. Gait & Posture 28, 207-216.

Gardiner, J. C. and Weiss, J. A. 2003. Subject-specific finite element analysis of the human medial collateral ligament during valgus knee loading. Journal of Orthopaedic Research 21, 1098-1106.

Harner, C. D., Hoher, J., Vogrin, T. M., Carlin, G. J. and Woo, S. L. Y. 1998. The Effects of a Popliteus Muscle Load on In Situ Forces in the Posterior Cruciate Ligament and on Knee Kinematics. Am J Sports Med 26, 669-673.

Hirokawa, S. and Tsuruno, R. 2000. Three-dimensional deformation and stress distribution in an analytical/computational model of the anterior cruciate ligament. J Biomech 33, 1069-1077.

Kwak, S. D., Ahmad, C. S., Gardner, T. R., Grelsamer, R. P., Henry, J. H., Blankevoort, L., Ateshian, G. A. and Mow, V. C. 2000. Hamstrings and iliotibial band forces affect knee kinematics and contact pattern. J Orthop Research 18, 101-108.

Li, G., Lopez, O. and Rubash, H. 2001. Variability of a Three-Dimensional Finite Element Model Constructed Using Magnetic Resonance Images of a Knee for Joint Contact Stress Analysis. J Biomech Engr 123, 341-346.

Li, G., Rudy, T. W., Sakane, M., Kanamori, A., Ma, C. B. and Woo, S. L. Y. 1999. The importance of quadriceps and hamstring muscle loading on knee kinematics and in-situ forces in the ACL. J Biomech 32, 395-400.

Negrut, D., Rampalli, R., Ottarsson, G. and Sajdak, A. 2007. On an Implementation of the Hilber-Hughes-Taylor Method in the Context of Index 3 Differential-Algebraic Equations of Multibody Dynamics (DETC2005-85096). J Comp and Nonlinear Dyn 2, 73-85.

Pandy, M. G. 2001. Computer Modeling and Simulation of Human Movement. Annual Reviews in Biomedical Engineering 3, 245-273.

Pandy, M. G. 2003. Simple and complex models for studying muscle function in walking. Phil Trans R Soc Lond 358, 1501-1509.

Sun, H. C. and Metaxas, D. N. 2001. Automating gait generation. Proceedings of the 28th annual conference on Computer graphics and interactive techniques 261-270.

Sun, Q. and Dhaher, Y. 2006. Three-dimensional Hyperelastic Model of the Human Knee: A Parametric Sensitivity Study. 7th World Congress on Computational Mechanics.

Thelen, D. G. and Anderson, F. C. 2006. Using computed muscle control to generate forward dynamic simulations of human walking from experimental data. J Biomech 39, 1107-1115.

Thelen, D. G., Anderson, F. C. and Delp, S. L. 2003. Generating dynamic simulations of movement using computed muscle control. Journal of Biomechanics 36, 321-328.