Analysis of Generalized-α vs. θ-methods in Physics-based Computer Simulation of Soft-body Materials

Ruining (Ray) Wu

April 23, 2018

Abstract

The generalized-alpha method is a current favourite of the computer graphics community for solving structural dynamics problems due to its nice algorithmic damping properties: It avoids damping of the low-frequency nodes, while damping the high-frequency nodes. In this work, we define a second order theta-method and introduce a set of problems where it outperforms the generalized alpha method both in terms of algorithmic damping and by other metrics that we define. From our results, we show the benefits of this method, and point out areas where our new method may be used as a suitable replacement for the generalized alpha method.

1 Introduction

Physics-based graphics simulation of nonrigid materials involves applying a finite-element discretization, where we regard physical bodies as collections of point-masses in space, and applying the laws of physics (i.e. $\vec{F} = M\vec{x}''$) to calculate their positions over time, usually with some sort of differential equation solver. Specifically, in this paper, we focus on physical systems of the form

$$Mx'' + Dx' + Kx = F(t, x, x') \tag{1}$$

where M is the mass matrix, D is the damping matrix (such as friction or linear air resistance), and K is the stiffness matrix (such as spring stiffness for linear spring). F is a vector function of applied forces, dependent on time, position, and velocity. Typically, the solution curve $x = x(t)$ is a complex exponential function, which means that the physical motion of these systems are oscillatory in nature.

This work focuses on physics-based simulation of flexible soft objects, whose motion is characterized by a high degree of stiffness, meaning that they have a wide range of frequency modes: both rapid and slow oscillations exist in their motion. Stiffness of a differential equation system makes numerical integration computationally difficult. In particular, it is necessary to use implicit methods, such as the trapezoidal rule (TR) or the family of backwards differentiation formulas (BDF) methods to allow us to take large timesteps. Using explicit methods would force us to take very small time steps to maintain stability. Although implicit methods allow us to take larger timesteps, a key drawback is that we are required to solve nonlinear systems at every timestep, which is computationally expensive. To make matters worse, these systems potentially have high condition number which makes obtaining their solution difficult. Another issue that we run into is algorithmic damping, which is a phenomenon that results in a loss of energy in the computer

animation of the physical system, when in reality there is none according to the laws of physics. However this doesn't apply to all implicit methods, since the TR, being an implicit method, does not exhibit any algorithmic damping. Additionally, this phenomenon is not restricted to implicit methods, since Runge-Kutta (RK) schemes also exhibit this phenomenon.

We analyze and compare the generalized-α method (a one-step multivalue method) and a 2nd order θ-method (an implicit method) that combines BDF-2 and TR. In this work, we compare these algorithms based on the total number of iterations required in solving nonlinear systems, the progression of condition number over time, and the resultant artificial damping. We want our ideal algorithm to have visually correct behaviour, fewer total iterations, lower condition number, and a reasonable degree of damping in the high-frequency modes, but very little in the low-frequency modes.

2 Methodology

This section describes the methodology used in the research of this paper. It describes the nonlinear solvers used, formulates the two algorithms that are being investigated, states the metrics of comparison of the algorithms, and defines the example problems that were designed.

2.1 Nonlinear Solver

Since implicit methods require us to solve nonlinear equations, we cannot avoid a discussion of nonlinear solvers. The purpose of the nonlinear solver is to solve an equation $f(x) = 0$ for x that we define based on the implicit method we are using to find the next value.

There are many solvers for nonlinear equations out there. Due to the continuity of the motion of the physical system, we decided to use Newton's method for nonlinear systems: given a function $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ and an initial guess x_0, we do the following iteration:

$$x_{k+1} = x_k - J^{-1}(x_k)f(x_k) \tag{2}$$

until the sequence converges to a value x that satisfies $\|f(x)\| \leq \epsilon$ for some tolerance ϵ, or after a maximum number of iterations has been reached. For our initial guess, we choose the last value that we have in the sequence: If we are defining a nonlinear function to solve for v_{k+1}, then our initial guess is chosen to be v_k. We believe this to be a reasonable choice due to the continuity of physical motion. In addition to this usual definition, we define a *failure* to be one of the following events:

- No decrease in the objective value is observed after three iterations (i.e. the values $\|f(x_3)\|$, $\|f(x_2)\|$, $\|f(x_1)\|$ are all greater than $\|f(x_0)\|$), or
- The algorithm has failed to converge within the maximum number of iterations.

These events occur when this subproblem is so ill-conditioned that finding a solution is not computationally feasible. A failure will cause us to disregard the output from the current step, and take two equally spaced sub-steps to get us to the next step. This is the approach that is used in [2]. For example, if our stepsize is 0.1, and we try the Newton method and observe a failure, then we change our step size temporarily to 0.05 and repeat this process twice. After we have passed this step, we revert our step size to 0.1.

2.1.1 Jacobian Matrix

The matrix J we used in our iteration is numerically generated from a three-point rule. We choose our stencil width h to be $\sqrt{\epsilon}\|x\|_\infty$, where ϵ is machine epsilon. On double precision systems, ϵ is approximately 10^{-16}.

2.2 Generalized Alpha

Generalized-α [3] is currently a favourite of the graphics community. It is a generalization of the algorithms defined in [4], [6], and [7], hence its name. In addition to a damping parameter ρ_∞. The algorithm accepts initial conditions x_0 and v_0. We define $a_0 = M^{-1}(F(t_0, x_0, v_0) - Dv_0 - Kx_0)$. Let h be the step size. The update equations are as follows:

$$x_{n+1} = x_n + hv_n + \frac{1}{2}h^2((1-2\beta)a_n + 2\beta a_{n+1}) \tag{3}$$

$$v_{n+1} = v_n + h((1-\gamma)a_n + \gamma a_{n+1}) \tag{4}$$

$$Ma_{n+1-\alpha_m} + Dv_{n+1-\alpha_f} + Kx_{n+1-\alpha_f} = F(t_{n+1-\alpha_f}) \tag{5}$$

Where we define the following:

$$a_{n+1-\alpha_m} = (1-\alpha_m)a_{n+1} + \alpha_m a_n \tag{6}$$

$$v_{n+1-\alpha_f} = (1-\alpha_f)v_{n+1} + \alpha_f v_n \tag{7}$$

$$x_{n+1-\alpha_f} = (1-\alpha_f)x_{n+1} + \alpha_f x_n \tag{8}$$

$$t_{n+1-\alpha_f} = (1-\alpha_f)t_{n+1} + \alpha_f t_n \tag{9}$$

When we implement this algorithm, we first substitute eqns (6), (7), (8), (9) into eqn (5) and then we substitute eqns (3) and (4) into (5). This makes eqn (5) a nonlinear equation for a_{n+1}, and once we solve this, we can easily get v_{n+1} and x_{n+1} using eqns (3) and (4) respectively.

One of the good properties about generalized-α is that it supposedly damps high frequency modes while avoiding damping on low-frequency modes. This property is one that we will be investigating in this paper.

The four parameters of this algorithm are γ, β, α_m, α_f, and they can be defined in terms of one parameter ρ_∞ as follows:

$$\gamma = \frac{1}{2} - \alpha_m + \alpha_f \quad \beta = \frac{1}{4}(1 - \alpha_m + \alpha_f)^2 \quad \alpha_f = \frac{\rho_\infty}{\rho_\infty + 1} \quad \alpha_m = \frac{2\rho_\infty - 1}{\rho_\infty + 1}$$

Alternative parametrizations lead to the Newmark[6], Hilber-Hughes-Taylor[4], or Wood-Bossack-Zienkiewicz[7] algorithms. However, the above are, in some sense, an *optimal* choice of parameters. According to [3] and the analysis in [5], the choice of γ ensures that the algorithm is 2nd order accurate, and the remaining parameters are chosen to maximize the damping in high-frequency and minimize the damping in low frequency modes[3]. This set of parameters will be what we investigate in this work.

The user-defined high-frequency damping, denoted as $\rho_\infty \in [0, 1]$ is typically chosen to be between 0.6 and 0.9, although sometimes we use values as low as 0.3. It is important to note that when $\rho_\infty = 1$, we have no algorithmic damping, whereas for the θ-method discussed later, $\theta = 0$ (and not 1) corresponds to the case of no algorithmic damping.

2.3 Second-order theta-method

A θ-method is defined by a convex combination of two different integration rules: denote w_{n+1} and v_{n+1} as the values from the two integration schemes. Our update step is

$$y_{n+1} = \theta w_{n+1} + (1-\theta) v_{n+1} \tag{10}$$

In general, θ-methods are a relatively unexamined area for these kinds of structural dynamics problems. In this paper we examine a 2nd-order θ-method involving TR and BDF-2 as discussed in [1]. Let h be the step-size, as before, and let $\theta \in [0,1]$ be a damping parameter. Our 2nd-order θ-method has the following update formula:

$$y_{n+1} = y_n + (1-\theta)\left(\frac{h}{2}\right)[y'_n + y'_{n+1}] + \left(\frac{\theta}{3}\right)[y_n - y_{n-1} + 2hy'_{n+1}] \tag{11}$$

and, applying eqn (11) to x_{n+1} and v_{n+1} we have the following system of equations:

$$\begin{bmatrix} x_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} x_n + (1-\theta)(\frac{h}{2})(v_n + v_{n+1}) + (\frac{\theta}{3})(x_n - x_{n-1} + 2hv_{n+1}) \\ v_n + (1-\theta)(\frac{h}{2})(a_n + a_{n+1}) + (\frac{\theta}{3})(v_n - v_{n-1} + 2ha_{n+1}) \end{bmatrix} \tag{12}$$

where a can be written in terms of x and v by rearranging eqn (1)

$$a_n = M^{-1}(F(t_n, x_n, v_n) - Dv_n - Kx_n) \tag{13}$$

When we implement this, we apply the substitution of eqn (13) into eqn (12) (for a_{n+1} and a_n), and then (12) becomes a nonlinear system for x_{n+1} and v_{n+1}. To solve this, we substitute the expression for x_{n+1} into the equation for v_{n+1} and a nonlinear equation for v_{n+1} results. After solving it, we can get x_{n+1} and then a_{n+1} easily with eqns (12) and (13)

This θ-method, like the generalized-α method, is 2nd order accurate (since BDF-2 and TR both are 2nd order accurate). However, θ, unlike ρ_∞, goes from 0 to 1 rather than from 1 to 0 when damping increases: no damping occurs when θ is chosen to be 0 and increasing θ increases the damping. Additionally, the scale of the damping is different: the θ-method has a linear scaling of the algorithmic damping but the generalized-α method has a nonlinear scale of damping, as we see from [1]. However, we do not know the properties of high vs. low frequency damping, and this is the primary direction of our research.

2.3.1 Arriving at the second point

The θ-method involving BDF-2 requires two previous data points, since BDF-2 is a linear multistep method (LMM). Therefore, it is not possible to start the integration given the initial conditions. There are two possible schemes to arrive at the 2nd data point:

- TR
- First-order θ-method.

Note that the second method actually contains the first for a certain choice of θ ($\theta = 0.5$). The advantage of using the first scheme is that we retain 2nd order accuracy, however, there is no numerical damping which may cause problems if the problem has a high condition number. The second scheme avoids this problem by allowing us to control the algorithmic damping, however, an important drawback is that it is not 2nd order accurate. This is not a problem for the first step, since the overall order of accuracy is not affected by one step. See the section below regarding substepping for more a more detailed discussion.

2.3.2 Substepping

Substepping is not an issue in generalized-α (we can choose whatever step size we want at every step), however, in our θ-method we run into the problem of needing two previous data points, and if we are changing the step size, the previous data point is not readily available. This is, in fact, a general problem with LMMs. There are algorithms to do BDF-like methods with unevenly spaced data points, but in this work we focus on the same two schemes as we did at the first step. See the above section for a discussion of their advantages and disadvantages. However, what is notably different from that section is that if the first order θ-method is applied too many times, we **do** have a problem since overall 2nd order accuracy is not preserved. In the 2nd-order θ-method, we substep by using a 1st-order θ-method on the half step (to get from v_n to $v_{n+1/2}$), then, we use a 2nd order θ method to get from v_n and $v_{n+1/2}$ to v_{n+1}, as these points are evenly spaced.

2.3.3 First-order theta-method

This first-order θ-method is used when we do not have two previous data points, or choose not to use them. It combines the BE (BDF-1) and TR integration schemes. Our update formula is as follows:

$$\begin{bmatrix} x_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} x_n + h((1-\theta)v_n + \theta v_{n+1}) \\ v_n + h((1-\theta)a_n + \theta a_{n+1}) \end{bmatrix} \tag{14}$$

We solve this system in similar fashion to the θ-method involving BDF-2: first we substitute expressions for x and a into the formula for v, solve that, and get formulas for x and a. We consider choices of θ between 0.5 and 1.0, with 0.5 giving us no algorithmic damping (trapezoid rule) and 1.0 giving us considerable damping (since it becomes BE). Note that since BE is actually the same as a first-order BDF, it has much more algorithmic damping than BDF-2, so care should be taken to ensure that the damping value for this method isn't simply that of the 2nd order theta method plus 0.5. In this paper, we choose the damping value to be $0.5 + \frac{\theta}{50}$, where θ is the damping parameter for the 2nd order θ-method. The objective of this is to still have some damping while mostly retaining 2nd order accuracy.

2.4 Methods of Comparisons

We are most interested to see how much damping our θ-method gives to the high-frequency modes, while maintaining the same damping for low-frequency modes as generalized-α, for a set of parameters (θ', ρ'_∞). We are also interested in seeing if our θ-method requires fewer Newton iterations, or if it is better-conditioned than generalized-α. It is hypothesized that fewer Newton iterations are required when the problem is less stiff (see Problem 3 and 4 below). Of course, we also need to make sure that the visual results of the algorithm is correct.

The purpose of our comparisons is as follows: the comparision of Newton iterations is a metric for the runtime of the algorithm, since the bulk of the work is from solving the nonlinear equations. Visual correctness is used to compare the correctness of the algorithm, since the purpose of the algorithm is to generate a visually correct animation. And finally, the condition number is used to judge the stability of the algorithm.

Our priority is on the comparison for visual correctness; this is the most important aspect. If on a certain problem, one algorithm gives the incorrect output while another gives the correct behaviour, then we do not need to look further; the condition number and Newton iterations are meaningless and do not need to be looked at, as the one that gave the correct answer is clearly

superior. Only when both algorithms give visually similar output, do we care about issues such as numerical damping and condition number.

2.5 Example Problems

We have four example problems, and they range in difficulty from trivial to fairly difficult, in terms of the relative stiffness of the problem.

2.5.1 Problem 1: Harmonic Oscillator

This problem is described by the differential equation $x'' + x = 0$. The initial data is $x_0 = 1$, $x_0' = 0$. No algorithm should have any difficulties with this problem: it represents a mass attached on a wall with a linear spring, with no surface friction or air resistance. The solution of this 2nd order differential equation is $x(t) = c_1 \sin(t) + c_2 \cos(t)$, but this simple problem allows us to demonstrate the effects of our damping parameters on the solution curve.

We will also look at fully decoupled systems that are similar to Problem 1 (i.e. $Mx'' + Kx = 0$ with M and D being diagonal matrices) in order to gain a better understanding of the high and low frequency damping properties of generalized-α and the 2nd order θ-method.

2.5.2 Problem 2: Pendulum

This problem is described by the differential equation $x'' = -g \sin(x)$. For simplicity, we will choose $g = 1.0$ instead of $g = 9.81$. The initial data is $x_0 = 1$, $x_0' = 0$

With this problem comes the introduction of nonlinear forces, so it makes sense to compare the total number of iteration counts. However, the system is 1x1 so condition number plots do not need to be looked at. Also, in this example there are no high frequency modes. So this is purely a test of the nonlinear solver.

2.5.3 Problem 3: Stiff Pendulum (Polar Coordinates)

This problem is described by the following differential equation ($r_0 = 1$):

$$\begin{bmatrix} r \\ \theta \end{bmatrix}'' = \begin{bmatrix} -\epsilon^{-2}(r - r_0) + r^{-3}\theta^2 + g\cos(\theta) \\ -gr^{-1}\sin(\theta) \end{bmatrix} \tag{15}$$

The initial data is an $\mathcal{O}(\epsilon)$ perturbation of $r(0) = 1$, $r'(0) = \frac{1}{\sqrt{2}}$, $\theta(0) = \frac{\pi}{4}$, $\theta'(0) = -\frac{1}{\sqrt{2}}$

This problem has a parameter ϵ, which is used to control the stiffness of the problem. We intend to vary the stiffness (ϵ is in a range of $[10^{-1}, 10^{-5}]$) to see the effect it has on our θ-method and generalized-α, in terms of condition number and Newton iteration count.

There are two modes, radius and angle of deviation. There are high oscillations in the radius mode (at a frequency corresponding to ϵ) and low oscillations in the angle mode (which is independent of ϵ), so we can compare the damping properties on the high-frequency modes, after fixing a certain level of damping on the low-frequencies.

Table 1: Summary of results for generalized-α-method

Test Example	Visual Correctness	Newton Iterations	Condition Number
Problem 1	Yes	Not needed	Not needed
Problem 2 ($x_0 = 1$)	Yes	50001	Not needed
Problem 2 ($x_0 = 2$)	Yes	50001	Not needed
Problem 3 ($\epsilon = 10^{-3}$)	Somewhat	2243	Low
Problem 3 ($\epsilon = 10^{-4}$)	No	N/A	N/A
Problem 3 ($\epsilon = 10^{-2}$)	Yes	1302	Low
Problem 4 ($\epsilon = 10^{-3}$)	No	N/A	N/A
Problem 4 ($\epsilon = 10^{-4}$)	No	N/A	N/A
Problem 4 ($\epsilon = 10^{-4}$)*	No	N/A	N/A
Problem 4 ($\epsilon = 10^{-2}$)	No	N/A	N/A
Problem 4 ($\epsilon = 10^{-2}$)*	Yes	2640	Low

2.5.4 Problem 4: Stiff Pendulum (Cartesian Coordinates)

This problem is the exact same as problem 3, however, instead of polar coordinates, we have cartesian coordinates. Let $\vec{q} = [q_1, q_2]^T$. The differential equation is:

$$\vec{q}'' = -\epsilon^2(r(\vec{q}) - r_0)\nabla r(\vec{q}) - g\cos(\theta(\vec{q}))\nabla r(\theta) + gr(\vec{q})\sin(\theta(\vec{q}))\nabla(\theta(q)) \tag{16}$$

where $\nabla r(\vec{q}) = \frac{1}{r}\vec{q}$, and $\nabla\theta(\vec{q}) = \frac{1}{r^2}(q_2, q_1)^T$

The initial data is an $\mathcal{O}(\epsilon)$ perturbation of $\vec{q}(0) = \frac{1}{\sqrt{2}}(1,1)^T$. $\vec{q}'(0) = (1,0)^T$. We do not believe that this change of coordinates will affect the damping, however, we do believe that a high degree of coupling is a more strenuous test on the conditioning and number of Newton iterations.

3 Results

Only results for the pair of parameters ($\rho_\infty = 0.6, \theta = 0.07$) are shown below and analyzed. Alternate sets of parameters with similar levels of damping can be found, and the results are roughly equivalent. Some other pairs of parameters with roughly equivalent numerical damping levels include ($\rho_\infty = 0.9, \theta = 0.001$), ($\rho_\infty = 0.3, \theta = 0.45$), ($\rho_\infty = 0.5, \theta = 0.15$). They were found experimentally on the test equation $x'' + x = 0$ for the maximum damping step size by fixing ρ_∞ and modifying θ such that the decay rate is similar based on observation.

The results are summerized in Table 1 for generalized-α and Table 2 for 2nd order θ method. We can see that the 2nd order θ algorithm is not worse than the generalized-α algorithm for any of the problems. We will compare the solution curves from the specific examples to gain a better understanding of the two algorithms under our comparison. Note that cases that have been marked with * are those where the stepsize has been adjusted from 0.05 to 0.02 in Problems 3 and 4.

3.1 Problem 1: Harmonic Oscillator

This problem is defined by the second-order differential equation $x'' + Kx = 0$, where K is a 2×2 diagonal matrix with diagonal entries $1, 100$. The initial conditions are $x_0 = (1,1)^T$, $x_0' = (0,0)^T$. There are two decoupled modes here, and the second one oscillates at 10 times the rate of the first.

Table 2: Summary of results for 2nd-order θ-method

Test Example	Visual Correctness	Newton Iterations	Condition Number
Problem 1	Yes	Not needed	Not needed
Problem 2 ($x_0 = 1$)	Yes	69295	Not needed
Problem 2 ($x_0 = 2$)	Yes	87052	Not needed
Problem 3 ($\epsilon = 10^{-3}$)	Yes	1946	Low
Problem 3 ($\epsilon = 10^{-4}$)	Yes	N/A	N/A
Problem 3 ($\epsilon = 10^{-2}$)	Yes	1940	Low
Problem 4 ($\epsilon = 10^{-3}$)	Yes	N/A	N/A
Problem 4 ($\epsilon = 10^{-4}$)	No	N/A	N/A
Problem 4 ($\epsilon = 10^{-4}$)*	Yes	N/A	N/A
Problem 4 ($\epsilon = 10^{-2}$)	No	N/A	N/A
Problem 4 ($\epsilon = 10^{-2}$)*	Yes	2534	Low

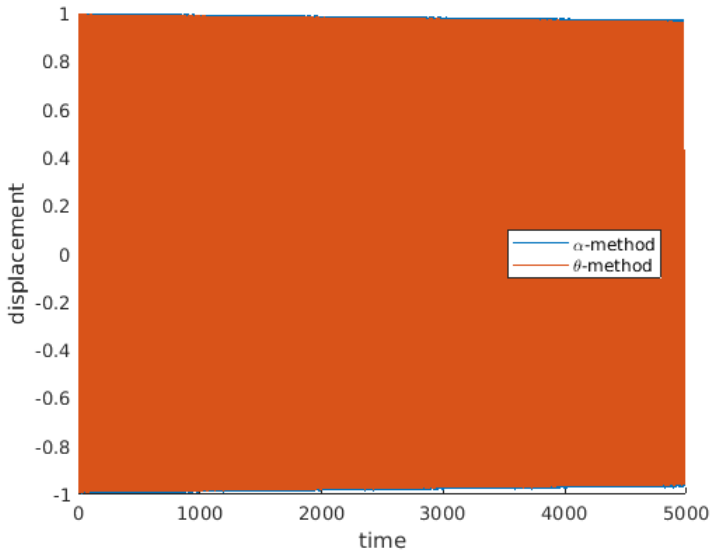

Figure 1: Low-Frequency damping of harmonic oscillator

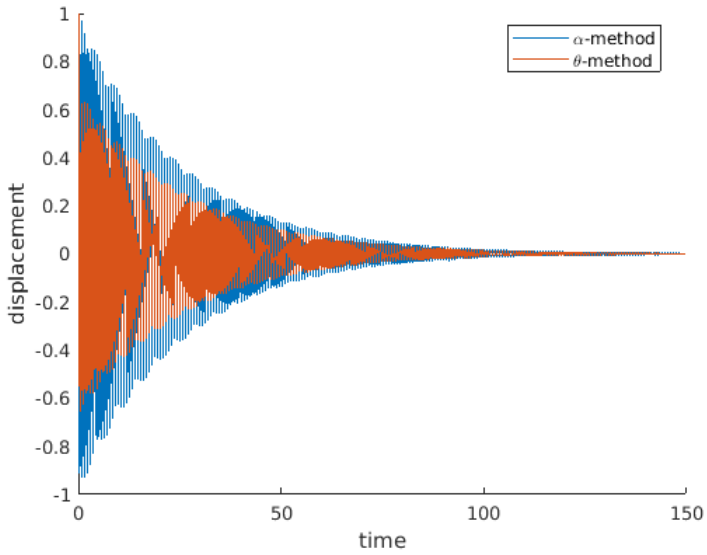

Figure 2: High-Frequency damping of harmonic oscillator

Figure 1 is the plot of the low-frequency mode. The solution curve from generalized-α is in blue, and that of 2nd order θ is in orange and overlays it. It can be seen that the orange just barely covers the blue, which indicates that the low-frequency damping is roughly equivalent. When we look at Figure 2, we see that for the high-frequency domain, the algorithmic damping is far more pronounced in the 2nd order θ-method compared to generalized-α, which is the behaviour that we would like to see.

The condition number remains constant throughout due to linearity. Linearity is also the reason why there is no need to look at the number of Newton iterations: it is n, where n is the number of steps we integrated, since Newton iterations solve linear systems exactly in 1 iteration.

3.2 Problem 2: Simple Pendulum

This is for a simple pendulum described by the differential equation $x'' = -g\sin(x)$ with initial conditions $x_0 = 1, x_0' = 0$. There are no high-frequency modes. The purpose of this problem is only to compare the nonlinear equations of the 2nd order θ-method vs. the generalized-α method in terms of the Newton iterations.

For approximately the same levels of damping, we see that the 2nd-order θ-method has 69295 nonlinear iterations to the 50001 used by generalized-α. It appears that the equation that generalized-α has set up is very close to linear, and we also examined x_0 much larger than 1: we set $x_0 = 2$ and keep the other parameters the same:

Our results are that the 2nd order θ-method uses 87052 iterations compared to the same 50001 iterations used by generalized-α. So, 2nd order θ-method uses 39% and 74% more iterations respectively. It seems that generalized-θ is better, but this is far from the full story as we will show

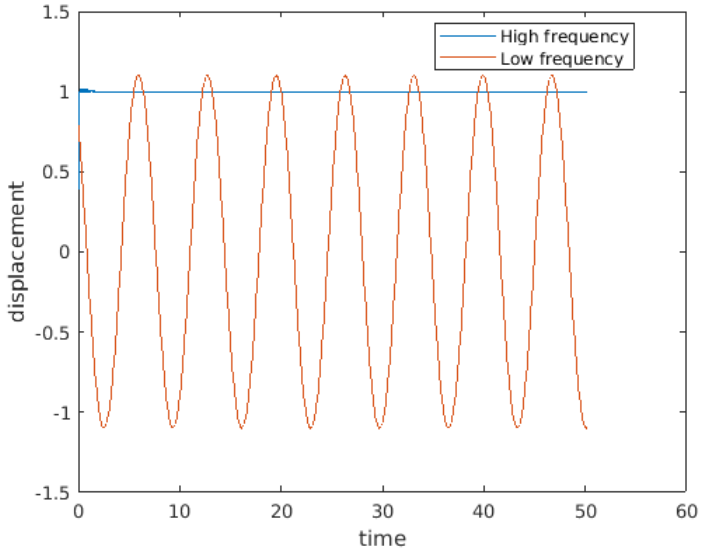

Figure 3: Problem 3, θ-method with $\epsilon = 10^{-3}$, stepsize $h = 0.05$

in our more complex problems. There is no need to compare condition number plots, as we have a 1x1 system.

3.3 Problem 3: Stiff Pendulum (Polar Coordinates)

This is a pendulum with loosely coupled polar coordinates. A representative case is when we choose our stiffness to be $\epsilon = 10^{-3}$, and stepsize to be $h = 0.05$. Our θ-method does significantly better than the generalized-α method, because the generalized-α method is not representative of the what we expect to see according to the laws of physics in the high frequency domain. However, qualitatively correct behaviour is retained in the low frequency domain. As can be seen in Figure 3 below, the theta-method highly resembles what we should see: the low frequency doesn't dampen while the high frequency mode dampens away and converges to 1. However, the generalized-α algorithm doesn't do so well (see Figure 4) in view of the sporadic changes in the radius (e.g. extending to ± 6, which should not be possible, since the radius should never be negative). So, we conclude that the θ-method is superior for this problem. Such a failure in the generalized-α method means that the step-size taken is too large. So, we need to pick a smaller step size if we wanted to use this algorithm. Additionally, the 2nd order θ-method uses fewer nonlinear iterations (1946 to 2243, only 87% of the iterations that generalized-α does). Also, note that generalized-α has a large increase in the condition number progression in Figure 5, almost to 6.0×10^4, which is another reason to prefer the 2nd order θ-method.

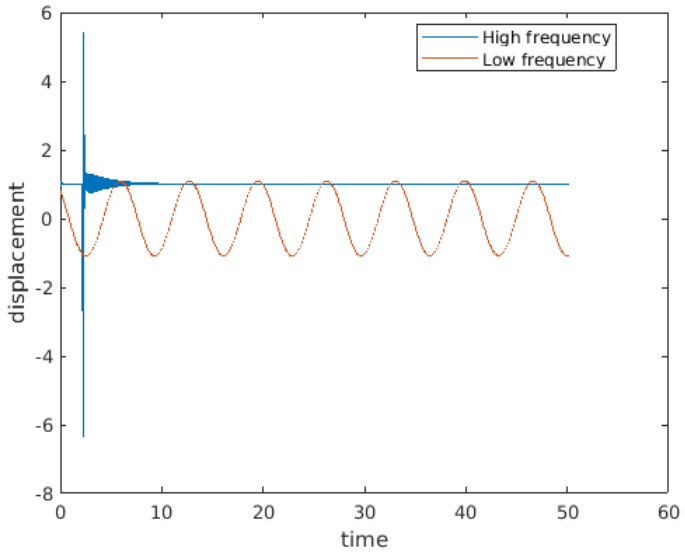

Figure 4: Problem 3, α-method with $\epsilon = 10^{-3}$, stepsize $h = 0.05$

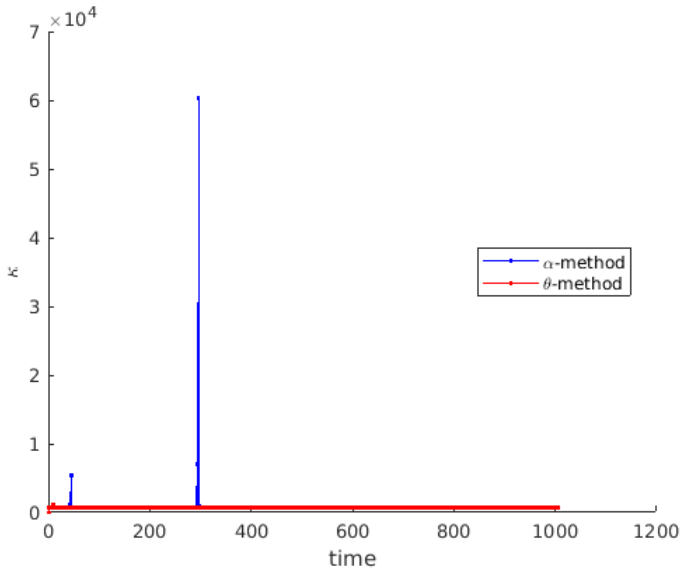

Figure 5: Problem 3 condition number, with $\epsilon = 10^{-3}$, stepsize $h = 0.05$

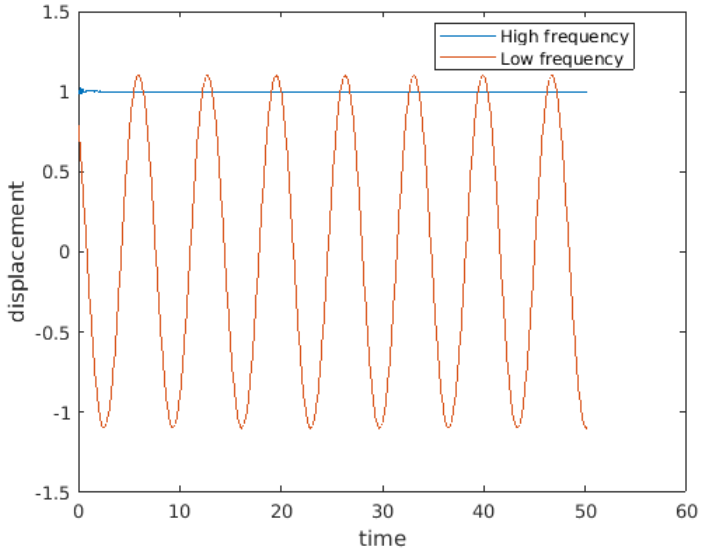

Figure 6: Problem 3, θ-method with $\epsilon = 10^{-2}$, stepsize $h = 0.05$

3.3.1 Changing epsilon to 10^{-4}

When we increase the difficulty of the problem, our θ-method successfully solves the problem (the plot looks very similar to the plot with the original problem), but the generalized-α method fails because the algorithm constantly subdivides with no end. This also attests to the superiority of the 2nd order θ-method.

3.3.2 Changing epsilon to 10^{-2}

When we decrease the difficulty of our original problem, we see that both algorithms integrate this successfully. Note that the generalized-α algorithm outperforms the 2nd order θ method in terms of Newton iterations. See Tables 1 and 2 for number of Newton iterations, and Figures 6, 7, 8 for the solution curves and condition number plots.

3.4 Problem 4: Stiff Pendulum (Cartesian Coordinates)

This is the same as problem 3, reformulated as cartesian coordinates. The parameters of the experiment remain the same. We see that initially, generalized-α gives correct behaviour (See Figure 10), but after about one period, the algorithm loses stability. The 2nd-order θ-method, on the other hand, maintains correct behaviour throughout as can be seen in Figure 9. Therefore, we can see that the 2nd order θ-method is superior. We do not need to compare the condition number plots, since generalized-α doesn't exhibit correct behaviour.

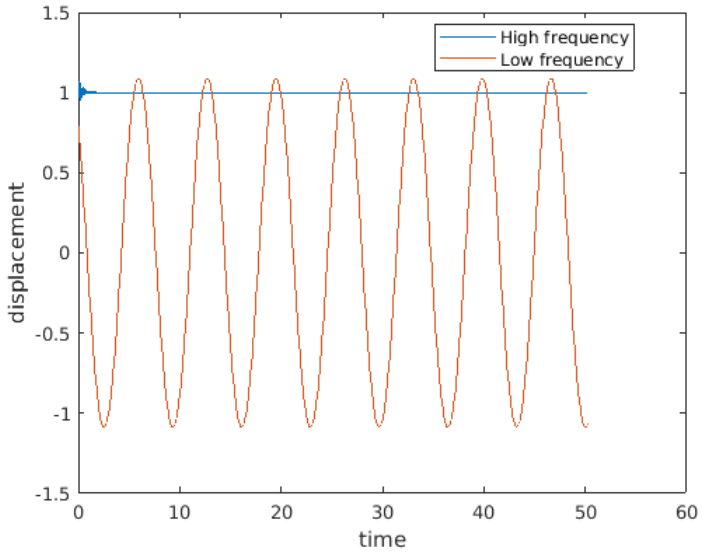

Figure 7: Problem 3, α-method with $\epsilon = 10^{-2}$, stepsize $h = 0.05$

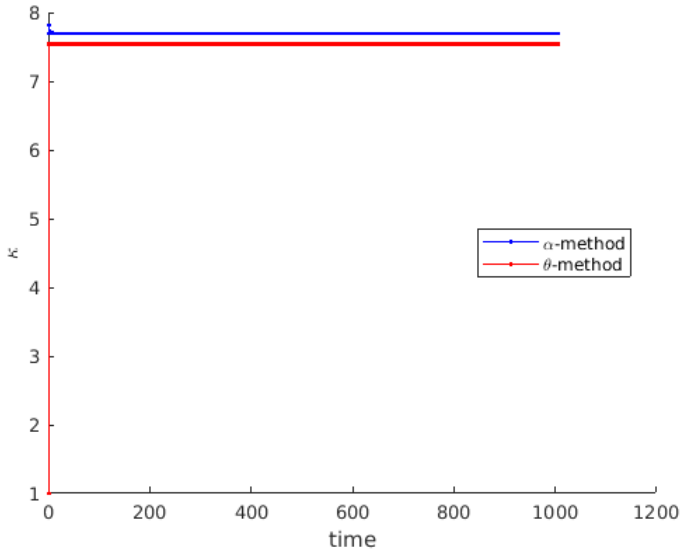

Figure 8: Problem 3 condition number, with $\epsilon = 10^{-2}$, stepsize $h = 0.05$

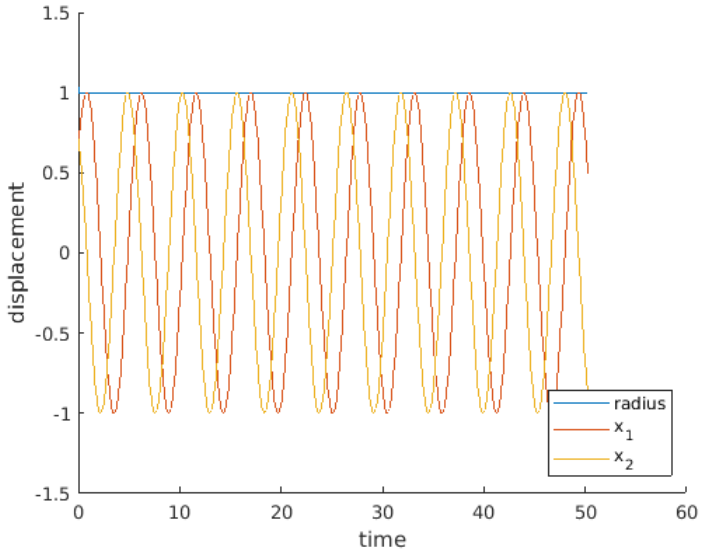

Figure 9: Problem 4, θ-method with $\epsilon = 10^{-3}$, stepsize $h = 0.05$

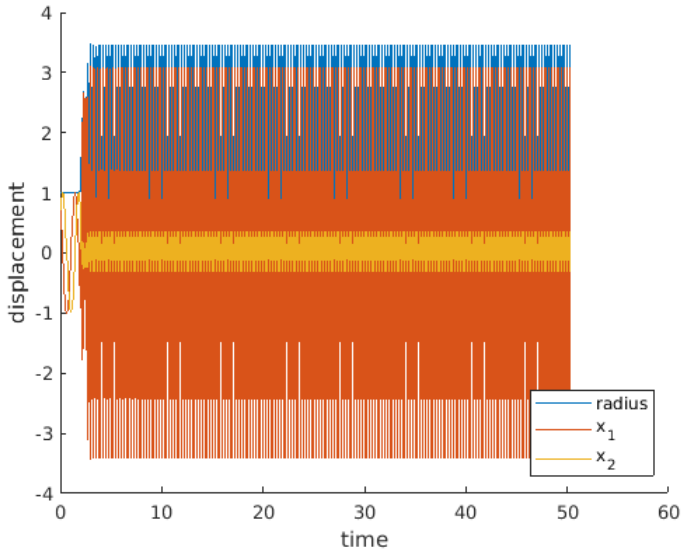

Figure 10: Problem 4, α-method with $\epsilon = 10^{-3}$, stepsize $h = 0.05$

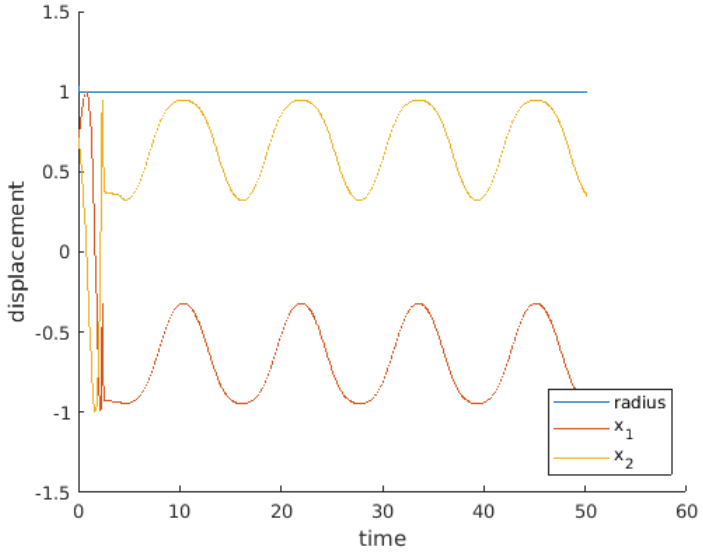

Figure 11: Problem 4, θ-method with $\epsilon = 10^{-4}$, stepsize $h = 0.05$

3.4.1 Changing epsilon to 10^{-4}

Neither the generalized-α method nor the θ method retains visually correct behaviour. See Figures 11 and 12 for the solution curves. Hence, we reduce the step-size to 0.02 and compare the algorithms again, when we can see that the θ-method is superior: the generalized-α method does not work for this step size. The solution curve for the θ-method in Figure 13 is visually very similar as before in Figure 9.

3.4.2 Changing epsilon to 10^{-2}

It seems that the damping for generalized-α on this problem is excessive, as can be seen from the solution curves for x_1 and x_2 in Figure 16. In comparison, our θ-method retains visually correct behaviour in Figure 15. Changing the step-size to 0.02 makes the problem sufficiently easy that both plots look fine, and the condition number is sufficiently low (< 10). See Figures 17, 18, 19 for the solution curves and condition numbers.

4 Discussion

Our preliminary results show that the θ-method is superior to the generalized-α method on these problems that we have designed. In particular, we note that the θ method is superior for highly stiff problems, since it is the only algorithm that retains visually correct behaviour. However, when both algorithms work, we do see that sometimes the generalized-α algorithm has less Newton iterations compared to the 2nd order θ algorithm. We discuss some possible future directions of research below.

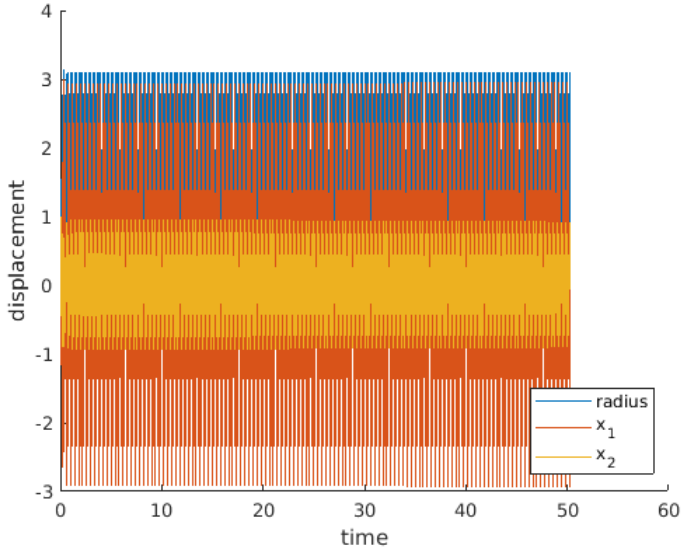

Figure 12: Problem 4, α-method with $\epsilon = 10^{-4}$, stepsize $h = 0.05$

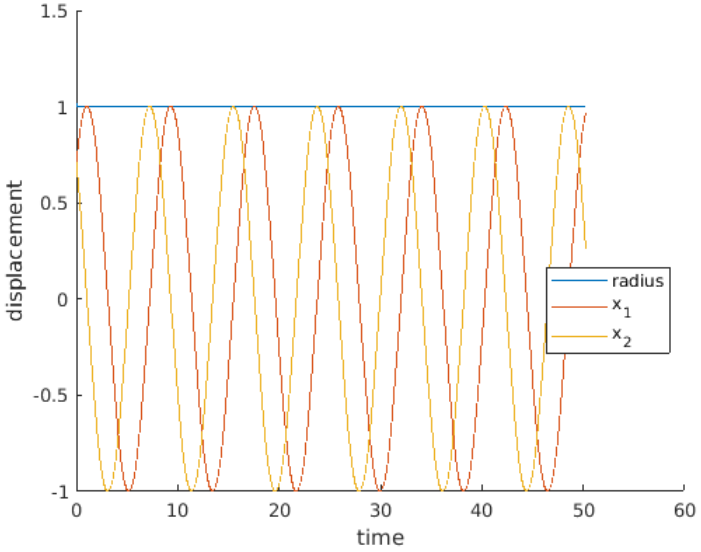

Figure 13: Problem 4, θ-method with $\epsilon = 10^{-4}$, stepsize $h = 0.02$

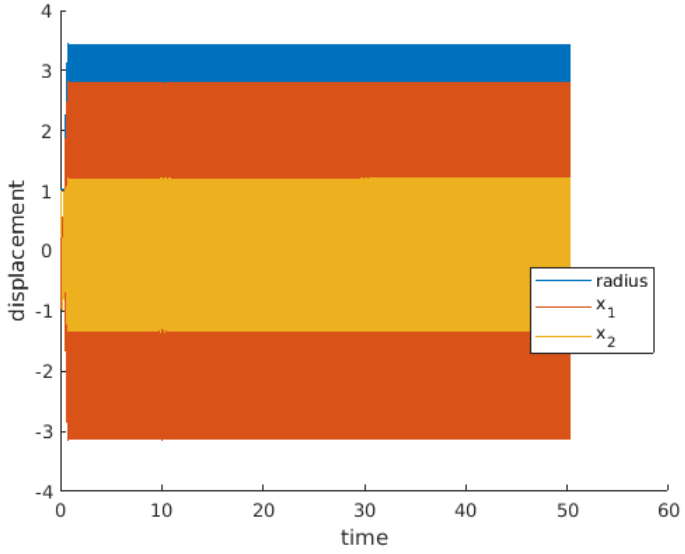

Figure 14: Problem 4, α-method with $\epsilon = 10^{-4}$, stepsize $h = 0.02$

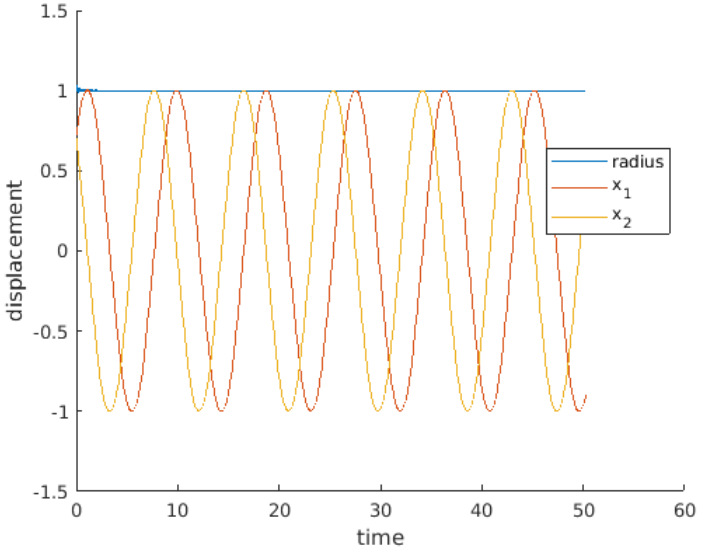

Figure 15: Problem 4, θ-method with $\epsilon = 10^{-2}$, stepsize $h = 0.05$

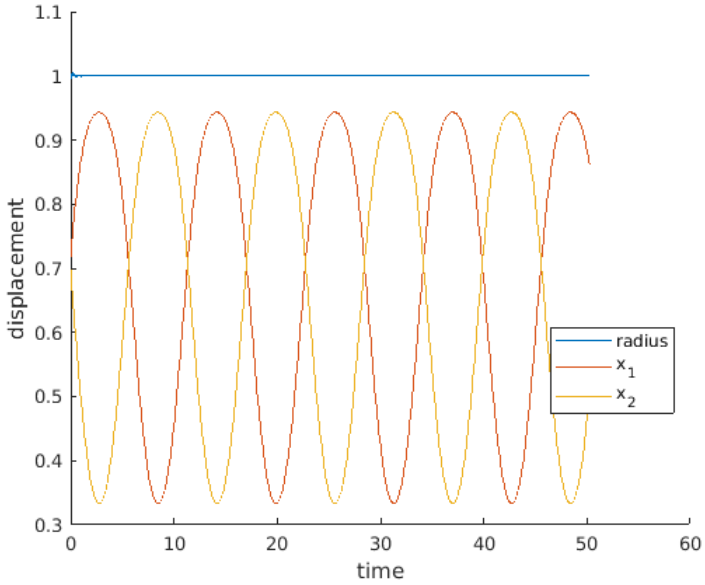

Figure 16: Problem 4, α-method with $\epsilon = 10^{-2}$, stepsize $h = 0.05$

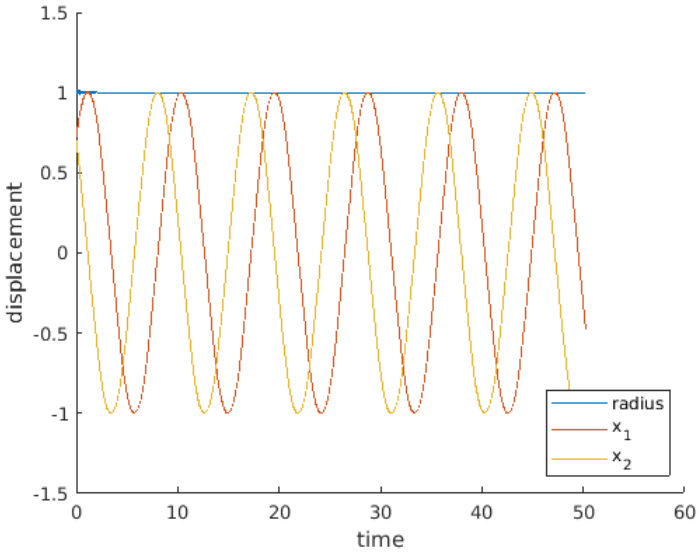

Figure 17: Problem 4, θ-method with $\epsilon = 10^{-2}$, stepsize $h = 0.02$

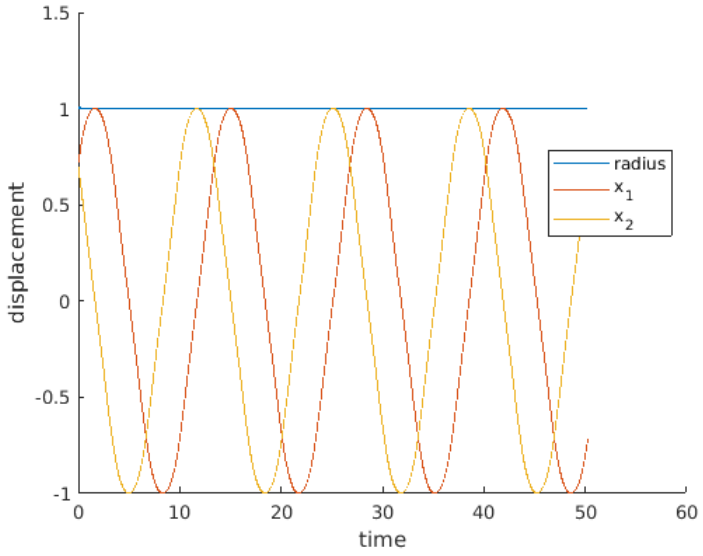

Figure 18: Problem 4, α-method with $\epsilon = 10^{-2}$, stepsize $h = 0.02$

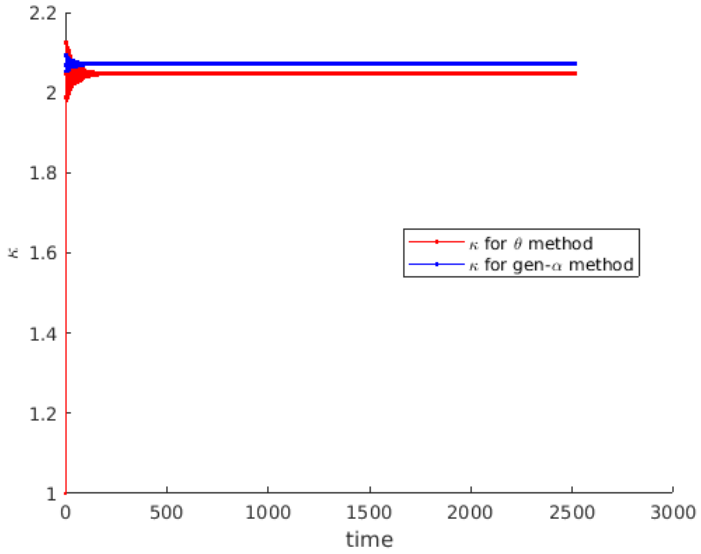

Figure 19: Problem 4 condition number, with $\epsilon = 10^{-2}$, stepsize $h = 0.02$

4.1 Future Research

- First, and most importantly, it remains to be seen if the 2nd order θ method keeps its advantage over generalized-α methods on highly conditioned, large-scale problems in the animation industry.
- It will also be of interest to see if the generalized-α algorithm uses less Newton iterations compared to the second-order θ algorithm for well-conditioned problems.
- Substepping: Since the TR is unsatisfactory due to the lack of algorithmic damping, and a first-order θ method is also lacking due to the overall loss of accuracy, are there 2nd-order methods with controllable algorithmic damping for the first substep? This requires an stencil taking on the values at $\{t_{i-1}, t_i, t_{i+1/2}\}$

5 Conclusion

In conclusion, we see that for the small problems that we have designed, our new algorithm is not worse than the generalized-α method. We see that our second-order θ method never fails unless if generalized-α also does so for our complex pendulum either in cartesian or polar coordinates. It seems that our second-order θ method would be a good replacement for generalized-α in these situations. Additionally, our method accepts the same set of parameters (excluding the damping constant, which is on a different scale) and returns the same values, so this new algorithm can be easily incorporated into existing implementations.

References

[1] U. Ascher and H. Huang. Numerical analysis in visual computing: What we can learn from each other. *Vietnam Journal of Mathematics*, 2018.

[2] E. Chen, U. Ascher, and D. Pai. Exponential rosenbrock-euler integrators for elastodynamic simulation. *IEEE Trans. Computer Graphics*, 2017.

[3] J. Chung and G. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-alpha method. *Journal of Applied Mechanics*, 60, 1993.

[4] H. Hilber, T. Hughes, and R. Taylor. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering and structural dynamics*, 5, 1977.

[5] M. Kobis and M. Arnold. Convergence of generalized-alpha time integration for nonlinear systems with stiff potential forces. *Multibody Systems Dynamics*, 2016.

[6] N. Newmark. A method of computation for structural dynamics. *Proceedings of the American Society of Civil Engineers*, 3, 1959.

[7] L. Wood, M. Bossak, and O. Zienkiewicz. An alpha modification of newmark's method. *International Journal for Numerical Methods in Engineering*, 1980.