AN INVESTIGATION INTO THE PERFORMANCE OF INTEGRATION METHODOLOGIES FOR REAL TIME MASS-SPRING CLOTH SIMULATIONS

by

CHRISTOPHER PHILLIPS
ID: 15022229

A dissertation submitted in partial fulfilment of the
requirements for the award of

MASTER OF SCIENCE IN COMPUTER GAMES PROGRAMMING

September 2016

Department of Computing
Staffordshire University
Stafford ST18 0AD

Supervised by: Christopher McCreadie
Word count: 12762

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Christopher Phillips
September 2016

Abstract

There are many models that have been proposed for cloth simulation but the majority at too computationally expensive to be used in real time. The Mass-Spring model is perhaps the only model that can be used for real time cloth simulations. This project investigates the performance of Mass-Spring model; in particular looking at the numerical integration methods that must be used to solve the differential equation inherent to the model.

The four most common explicit integrators, explicit Euler, Midpoint, fourth order Runge-Kutta and Verlet, were implemented for the Mass-Spring model. Data about the performance of each integrator was extracted using two simulation scenarios. The data showed that the choice of numerical integrator can have a great impact on the performance of the model and the results were different from what was expected. The expensive Midpoint and fourth order Runge-Kutta integrators could not use a sufficiently large time step to offset their high cost and so the cheap explicit Euler and Verlet integrators are recommended for real time simulations.

Acknowledgements

With thanks to Dr Christopher McCreadie

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The visual fidelity of video games has increased dramatically in recent years largely due to developments in discrete graphics hardware, or graphics processing units (GPUs). As a result, there is a need for realistic cloth in order to sell the appearance of video game characters and scenes. It is essential that the cloth should react to dynamic behaviour, such as a character moving or the wind blowing, and it must be animated in real time, i.e. at a minimum of 30 frames per second (FPS). These requirements render many cloth simulation techniques inappropriate, as they are too computationally expensive to be run in real time. The Mass-Spring model is one technique that is appropriate for use in video games, and is the most popular method for handling cloth for games. Whilst not necessarily providing 100% accuracy, Mass-Spring models result in real time frame rates and give visually pleasing animation, which is good enough for games.

The traditional Mass-Spring model uses forces, calculated using Newtonian mechanics, to animate the cloth. This results in a series of differential equations that must be approximated by a numerical integrator at discrete time intervals. There are many different integrators that can be used, the choice of which can directly affect the performance of the simulation. Some integrators must necessarily use small time intervals in order to maintain the stability of the cloth, and this increases the frequency of the integration calculations thus reducing performance.

This project will investigate the effect different integration methods have on a real time cloth simulation using the Mass-Spring model.

1.1 Project Aims

The aim of this project is to investigate the performance effects of different integration methods on real time cloth simulation using the Mass-Spring model.

Based on this aim, the objectives of the project are as follows:

- To research cloth simulation and numerical integration techniques

- To implement the Mass-Spring model for cloth simulation and several integrators, including explicit Euler

- To investigate the performance effects of the implemented integration methods on the simulation

1.2 Hypotheses

Volino and Magnenat-Thalmann (2001) were the first to investigate the performance impact of different integrators on cloth simulation. Their results, however, are extremely outdated, due to the vast increase in CPU power since 2001; from a 200MHz workstation CPU to the 3GHz+ CPUs available in modern desktops and laptops. Later work by Wang, Hu, and Zhuang (2009) also investigated the impact of different integrators, and found that integrator choice still has an impact on simulation performance. However they make no reference to the hardware on which their experiments were run.

As a result, two hypotheses are proposed for this project, a null and an alternative.

1.2.1 Null Hypothesis

The null hypothesis is that all integration methods result in a real time cloth simulation when running on modern hardware.

This was chosen because Wang, Hu, and Zhuang (ibid.) make no reference to the hardware used, and the simulation developed will be run on an Intel I7 4770K at 4.2GHz, so it may be the case that there are no performance concerns with this modern hardware.

1.2.2 Alternative Hypothesis

The alternate hypothesis is that some integration methods are prohibitively expensive for real time cloth simulation and other methods provide better performance.

Some researchers, such as Zeller (2005) and Tang et al. (2013), have proposed GPU accelerated approaches to Mass-Spring models which suggests that performance is still a consideration. Third party physics engines, such as NVIDIA® PhysX®, also use GPU accelerated models (Kim 2011) again suggesting performance of Mass-Spring models are still a concern.

1.3   Report Structure

The remainder of this report will be structured as follows:

- Chapter 2: literature review. An overview of cloth simulation techniques will be given and the Mass-Spring model and some numerical integrators described in detail

- Chapter 3: design and implementation. This chapter will describe the development methodology, design and some of the implementation details of the project

- Chapter 4: test plan. Details on how the hypotheses will be tested are described

- Chapter 5: data analysis and evaluation. The data gathered from the testing process is presented and analysed, and the hypotheses evaluated

- Chapter 6: conclusion. The results of the project are summarised and some suggestions for future work proposed

Chapter 2

Literature Review

2.1 Cloth Simulation

The simulation of cloth is a relatively old and well studied field with applications in many different areas, including, but not limited to:

- Virtual Garment Design

- Virtual Fitting Rooms

- Films

- Video Games

Different use cases require different things from the cloth simulation. For example, in virtual garment design, the physical accuracy of the simulation is paramount whereas cloth simulation for video games prioritises real-time simulation, sacrificing accuracy. Hence, many different models for cloth simulation have been proposed.

Since this project is concerned with the real-time simulation of cloth, only those models appropriate for real-time simulations have been studied in detail. However, a brief overview of other techniques will be provided. For a more detailed overview of cloth simulation techniques, see Ng and Grimsdale (1996).

2.1.1 Cloth Properties

Cloth has several properties that should be considered for modelling.
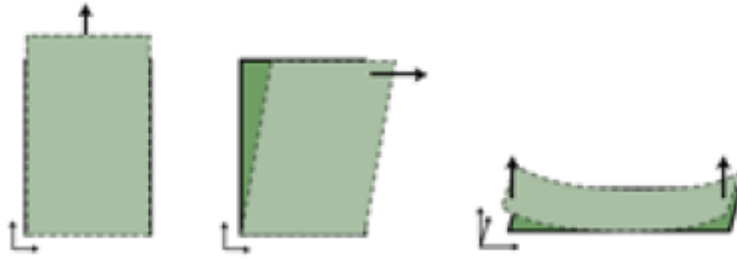
Figure 2.1: Mechanical properties of cloth (Yalçin and Yildiz 2009, p. 1)

2.1.1.1   Mechanical Properties

Cloth has three mechanical properties that control its behaviour; stretching, shearing and bending, Fig 2.1 shows how each property affects the cloth.

Stretching is the displacement of the cloth in either the horizontal or vertical direction. Most cloth has a high resistance to stretching and can typically only be stretched by 10% (Yalçin and Yildiz 2009; Provot 1995).

Shearing is the displacement of the cloth in a diagonal direction. Again, most cloths have high shearing resistance and this, coupled with high stretch resistance, makes cloth incompressible.

Finally, bending is the overall curvature of the cloth surface. Typically, cloth has low bending resistance and so is easily folded.

2.1.1.2   Visual Properties

The mechanical properties of cloth, discussed above, cause cloth to exhibit two visual properties. These properties arise from the fact that cloth is typically non-elastic, due to stretch and shear resistances, but highly flexible, due to low bend resistance.

Firstly, cloth will drape over objects and secondly the cloth will form many folds and wrinkles. Fig 2.2 demonstrates the visual properties of cloth.

2.1.2   Cloth Models

Techniques for modelling cloth are usually classified as either Geometric or Physically-based, and the choice of which modelling method to use depends on the use case for the simulation.
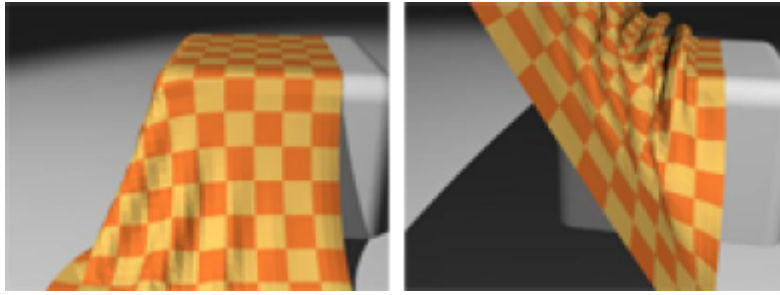
Figure 2.2: Visual properties of cloth (Yalçin and Yildiz 2009, p. 1)

2.1.2.1 Geometric Models

This family of techniques were the first models used to simulate cloth. They model the cloth using geometric equations and are especially good at modelling folds and wrinkles.

According to Ng and Grimsdale (1996), Weil was the first to propose a geometric model in 1986 and uses catenary curves to model the drape and folds of a hanging cloth. Following on from Weil, a number of other geometric models were proposed (see Ng and Grimsdale (ibid.) for more information).

All geometric techniques focus on simulating the appearance of cloth, rather than the physical properties. As such, geometric models are typically more computationally efficient than physically-based models, as there is no need to solve a series of complex equations. However, geometric techniques are unable to accurately simulate the motion of cloth, (Mongus et al. 2012; Zhang and Yuen 2001; Xinrong et al. 2009), and so are mostly useful for static cloth simulations.

As such, geometric models have not been considered for this project.

2.1.2.2 Physically-based Models

By contrast, physically-based models are concerned with the accurate modelling of the physical properties of the cloth and can therefore be used to produce realistic animations.

These models typically use a system of partial differential equations (PDE), or other differential equations, to model the cloth. These equations cannot be solved analytically and therefore the system requires discretisation to solve the equations at specific points in space and time. Following discretisation, a physically-based model typically involves the solving of an ordinary differential equation (ODE) of

the form (Baraff and Witkin 1998, p. 1):

$$\ddot{x} = M^{-1} \left(-\frac{\delta E}{\delta x} + F \right)$$

where:

x is a vector representing the geometric state of the system

(2.1)

M is a diagonal matrix representing the mass distribution of the system

E is a function of x which yields the internal cloth energy

F is a function of x and $\dot{x}$ which describes other forces

Physically-based models can be classified as either Continuum or Discrete.

2.1.2.2.1 Continuum Models

Continuum models were the first physical models to be proposed. Techniques in this family model cloth as a continuous surface and utilise continuum mechanics to calculate its behaviour; the Lagrange equations are most commonly used.

To discretise the continuous model, a numerical technique, such as a finite element method (FEM), is used. This is one of the advantages of continuum methods; they allow the use of a low resolution discretisation without sacrificing the accuracy of the simulation (Wacker, Thomaszewski, and Keckeisen 2005).

Another advantage of continuum models are that they are accurate; "they provide accurate models of the material derived directly from mechanical laws and models of material properties" (Magnenat-Thalmann and Thalmann 2004, p. 200).

This accuracy comes at the cost of computational performance, the main disadvantage of these techniques. The accuracy also renders these models inappropriate for use in dynamic simulations; "the formal and analytical description they require for the mechanical behavior of the material cannot easily be altered to represent transitory and non-linear events. Hence, phenomena such as frequent collisions or other highly variable geometrical constraints cannot be conveniently taken into account" (ibid., p. 200). Hence, continuum models are typically only considered appropriate for static simulations, or simulations where accuracy is paramount. As a result, continuum models have not been considered.

2.1.2.2.2 Discrete Models

According to Choi and Ko (2002, p. 2), "Cloth is not a homogeneous continuum. Therefore modeling

fabrics as a continuum and employing FEM or FDM has several potential drawbacks". As a result several discrete, or particle, models have been proposed.

With these techniques the discretisation in space is carried out by modelling the cloth as a discrete mesh, either regular or triangular, of point masses, called particles. This sacrifices some of the accuracy of continuum models, as the accuracy will depend on the number of particles used. However, this loss in accuracy is traded off against better computational performance; physical models are the only models that can be used for dynamic, real-time simulations.

The discretisation of the cloth has a direct affect on the performance of the simulation; Volino and Magnenat-Thalmann (2001) have shown this to be of order O(n^3). Hence, there is a trade off to be made between accuracy and performance when using discrete models, depending on the use case.

By far the most popular technique is the mass-spring model. This is because, according to Zink and Hardy (2007, p. 2), "Mass-spring models are the most efficient as well as the simplest of the cloth models" and "is one of the most popular techniques for simulating cloth, especially when interactive frame rates are required". Thus, it has been chosen for this project and will now be described in more detail.

2.1.3   Mass-Spring Models

Mass-spring models were first proposed for use in cloth simulation in Provot (1995). Using these models, the cloth is discretised as a 2-dimensional mesh of point masses, either regular or triangular, connected by linear springs.

2.1.3.1   Provot Model

Using the mass-spring model proposed in Provot (ibid.) the cloth is modelled as a regular mesh of point masses. The points are connected together using three different types of springs:

- Structural springs, connecting particle [i, j] to particles [i + 1, j] and [i, j + 1]. These springs resist structural deformations of the cloth, and provide the overall cloth structure. On their own they are not enough to provide a realistic cloth model; Fig 2.3 shows the results of running a cloth simulation with structural springs only. As can be seen, this does not produce a realistic image

- Shear springs, connecting particle [i, j] to particle [i + 1, j + 1] and particle [i + 1, j] to particle [i, j + 1]. These springs provide shearing resistance for the cloth. By adding shear springs, the realism

(a) Initial configuration   (b) Result of running the simulation
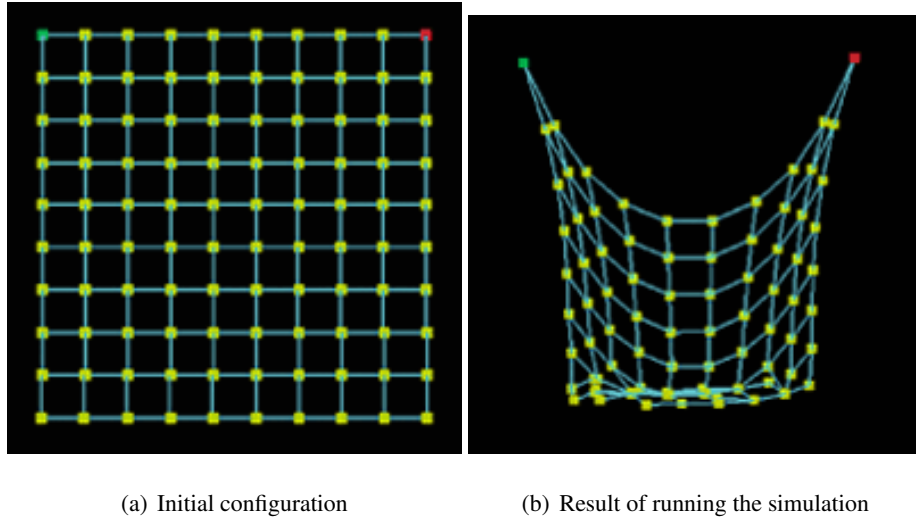
Figure 2.3: Cloth with structural springs only (Lander 2000b, p. 2)

(a) Initial configuration   (b) Result of running the simulation
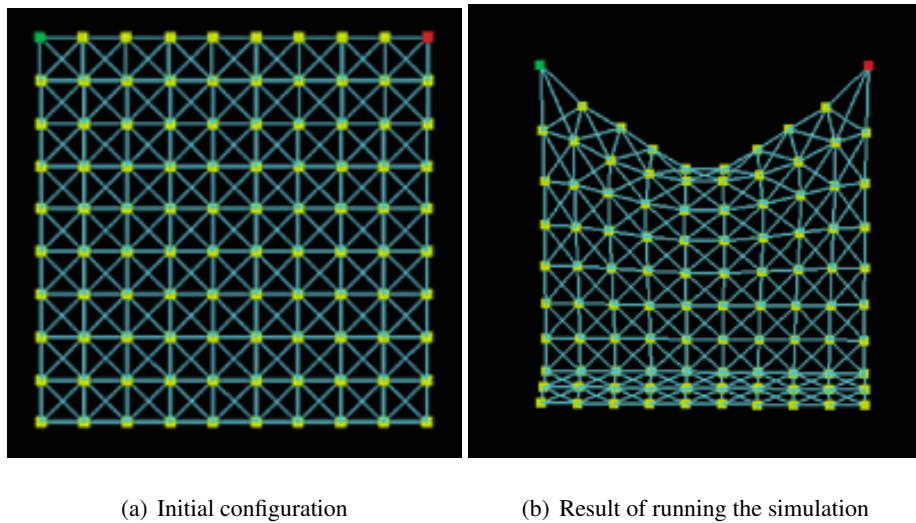
Figure 2.4: Cloth with structural and shear springs (Lander 2000b, p. 2)

of the model is improved, as shown in Fig 2.4

- Bend springs, connecting particle [i, j] to particles [i + 2, j] and [i, j + 2]. These springs model bend resistance

Fig 2.5 shows the arrangement of these springs for a small cloth model.
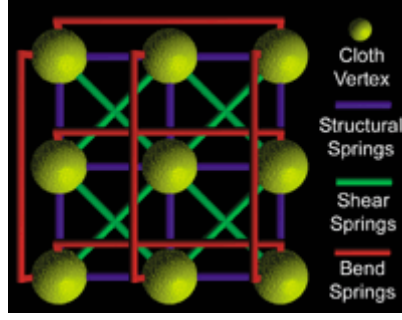
Figure 2.5: Provot cloth model (Lander 2000b, p. 2)

The total number of each type of spring can be calculated with:

$$Num_{structural} = 2((m-1)(n-1)) + (m-1) + (n-1)$$

$$Num_{shear} = 2((m-1)(n-1))$$

$$Num_{flexion} = 2((m-2)(n-2)) + 2((m-2)+(n-2)) \qquad (2.2)$$

where:

m and n are the vertical and horizontal dimensions of the mesh

To animate the mesh, forces are applied to the particles and are calculated using Newton's second law:

$$F_{ij} = m_{ij}a_{ij}$$

where:

F_{ij} is the total sum of forces acting on particle ij $\qquad (2.3)$

m_{ij} is the mass of particle ij

a_{ij} is the acceleration of particle ij

The total force acting on a particle is defined as:

$$F_{total} = \Sigma F_{external} + \Sigma F_{internal} \qquad (2.4)$$

$F_{external}$ are external forces acting on the mesh, such as gravity and wind.

Gravity is calculated by:

$$F_g = m_{ij}G$$

where: $\qquad (2.5)$

G is the gravitational constant

Wind is calculated by:

$$F_{wind} = w(n_{ij} \bullet \overrightarrow{W})$$

where:

n$_{ij}$ is the surface normal of particle ij

$\overrightarrow{W}$ is the wind direction vector

w is the wind constant

(2.6)

F$_{internal}$ are the resultant forces of the springs connecting the mesh.

The spring force is calculated using the Hooke equation (Parent 2012, p. 201):

$$F_{spring} = -k_s(L_c - L_r)\frac{p_2 - p_1}{\| p_2 - p_1 \|}$$

where:

k$_s$ is the spring stiffness coefficient

L$_c$ is the current length of the spring

L$_r$ is the initial, or rest, length of the spring

p$_1$ & p$_2$ are the positions of the two connected particles

(2.7)

Using this equation, the cloth will be modelled with pure elastic springs and will oscillate indefinitely. However, as mentioned in 2.1.1.2 and Provot (1995), cloth is a non-elastic medium and therefore the model needs to account for the energy lost due to internal friction.

This is typically modelled as an extra internal damping force, calculated using (Parent 2012, p. 201):

$$F_{damping} = -k_d(\dot{p_2} - \dot{p_1}) \bullet \left(\frac{p_2 - p_1}{\| p_2 - p_1 \|} \right) \left(\frac{p_2 - p_1}{\| p_2 - p_1 \|} \right)$$

where:

k$_d$ is the spring damping coefficient

$\dot{p_1}$ & $\dot{p_2}$ are the velocities of the two connected particles

(2.8)

2.1.3.2 Choi Ko Model

A different mass-spring model was proposed in Choi and Ko (2002) and aims to improve the buckling behaviour of the cloth, resulting in more realistic draping and wrinkling behaviour.

The cloth is modelled in a similar way to the Provot method, but additional bend springs are added, connecting particle [i, j] to particle [i + 2, j + 2] and particle [i + 2, j] to particle [i, j + 2]; Fig 2.6 shows the arrangement of springs.
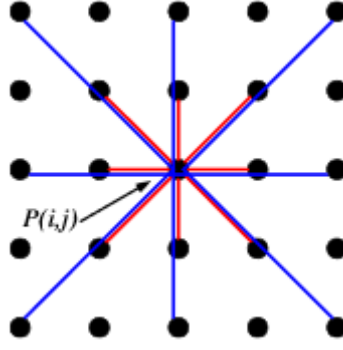
Figure 2.6: Choi Ko cloth model (Choi and Ko 2002, p. 2)

This model uses an energy-based approach, of the general formula 2.9(Bartels 2014, p. 3), to calculate the forces acting on individual particles.

$$F = \left(\frac{\delta E(S)}{\delta x}, \frac{\delta E(S)}{\delta y}, \frac{\delta E(S)}{\delta z} \right)$$

where: (2.9)

E(S) is an energy function of S, a representation of the cloth's state

Two types of interactions are defined, type 1 and type 2. Type 1 interactions model stretch and shear resistances, the red lines in 2.6, and are represented by a linear spring model. Type 2 interactions model bend forces, the blue lines in 2.6, and helps prevent the so called post-buckling instability problem. The interested reader should see Choi and Ko (2002) for more information on the energy functions for each interaction type.

2.1.3.3   Justification of Choice

Mass-spring models are suitable for use in this project as they are efficient and simple to implement; "Mass-spring models are the most efficient as well as the simplest of the cloth models. This method is one of the most popular techniques for simulating cloth, especially when interactive frame rates are required" (Zink and Hardy 2007, p. 2). As this project is concerned with the real time animation of cloth, mass-spring models are therefore the obvious choice.

Mass-spring models are also the most common method of modelling cloth in video games, used in games such as Alan Wake, (Enqvist 2010), and Hitman: Codename 47, (Jakobsen 2005), as well as commercial physics engines for games, such as PhysX. Again, since this project is concerned with the simulation of cloth for use in a video game, mass-spring models are the logical choice.

In particular, the Provot model was used for this project as the force-based approach requires the solving

(a) Super-elastic effect   (b) Result of correcting super-elasticity
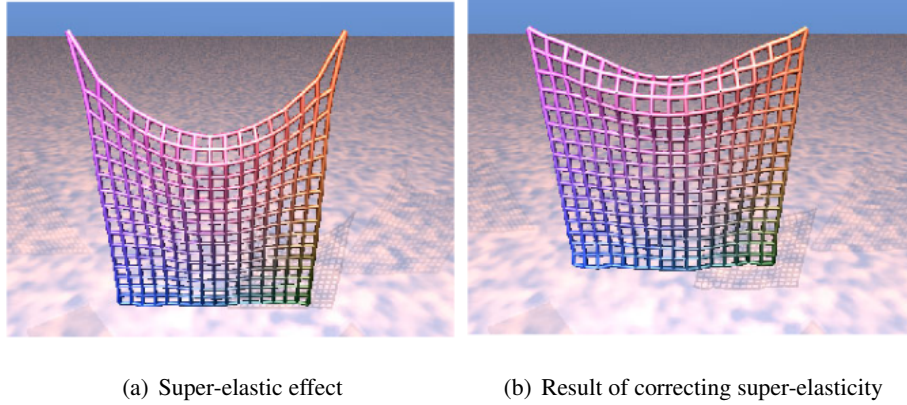
Figure 2.7: Super-elasticity problems (Provot 1995, pp. 4,6)

of a much simpler series of equations than the energy-based approach of the Choi Ko model, and is therefore likely to be more computationally efficient.

One disadvantage of the mass-spring model is that it does not achieve realistic animation of cloth as "mass-spring systems do not model any specific material and are not related to measured properties of real clothes" (Wacker, Thomaszewski, and Keckeisen 2005, p. 3). However, by careful tuning of the spring stiffness coefficients pleasing results can be achieved; Mongus et al. (2012) have shown that, with tuning, mass-spring models can reproduce the drape of a cloth with an accuracy of 97%.

Another disadvantage of mass-spring models is what Provot calls 'super-elasticity'; springs are allowed to deform too much, leading to unrealistic results. This is caused because "the springs are "ideal" and they have unlimited linear deformation rate" (Vassilev, Spanlang, and Chrysanthou 2001, p. 3). To counter this effect, Provot suggests enforcing length constraints on structural and shear springs. First the position of the particles are updated, then the deformation of the springs are calculated. If this deformation value is greater than some threshold, τ_c, then the position of the connected particles are adjusted so the deformation rate equals τ_c. Fig 2.7 shows the super-elasticity problem and the results of employing Provot's corrective method. As can be seen, correcting the super-elasticity results in a much more realistic model.

2.1.3.4 Mass-Spring Models for Video Games

Both the Provot and Choi Ko Mass-Spring models can be separated into two component, the modelling component and the animation component. The modelling component describes how the cloth will be represented internally, i.e. the disrectisation by point masses and the interconnectivity of the springs,

and the animation component details how the cloth will be animated over time; for the Provot model, using the force equations detailed in the previous sections.

For video games, where performance is a key factor, using the force based animation methods in the Provot and Choi Ko models may be prohibitively expensive over a certain mesh size. This results from the fact that the number of springs grows rapidly as the number of particles increases, leading to more internal force calculations per time step. For example, for a cloth represented by a 50^2 mesh of particles connected as in the Provot model, there are 14502 springs, and therefore Equations 2.7 and 2.8 must be calculated 14502 times at every time step. For the Choi Ko model, the internal force cost would only rise, as not only are there more springs in their model, but also their force calculations are more expensive. Thus, another approach to animating Mass-Spring models is usually taken for video games, called Position Based Dynamics (PBD).

PBD were first used by Jakobsen (2005) to implement cloth in Hitman: Codename 47 and have since gone on to become the standard for cloth animation in games (Enqvist 2010) and commercial physics engines (Kim 2011).

When using PBD, the cloth is modelled either using the Provot method (Enqvist 2010) or using a simplified version of the Provot model, where the flexion springs are omitted (Zeller 2005; Kim 2011). Instead of being modelled as Hooke springs, the spring connections are instead treated as distance constraints of the form

$$\| x2 - x1 \| = d$$

where:

$x2$ and $x1$ are the positions of the particles connected by the constraint

(2.10)

d is the length of the constraint

The work of Jakobsen (2005) was developed in Müller et al. (2006) resulting in a more general PBD model that supports both equality and inequality constraints. For games however, modelling cloth as a mesh of particles connected by simple distance constraints is enough.

To animate the cloth using PBD, external forces are first applied to the cloth, integrated using Verlet integration and the particle positions updated. Since the particles have moved, it is possible that one or more of the spring constraints are violated so they are checked and the positions of the particles corrected so none are violated; this is called constraint relaxation and is shown in Fig 2.8 . The constraints are correct using Equation 2.11(ibid., p. 4); stiffness can also be added to control the appearance of the cloth.
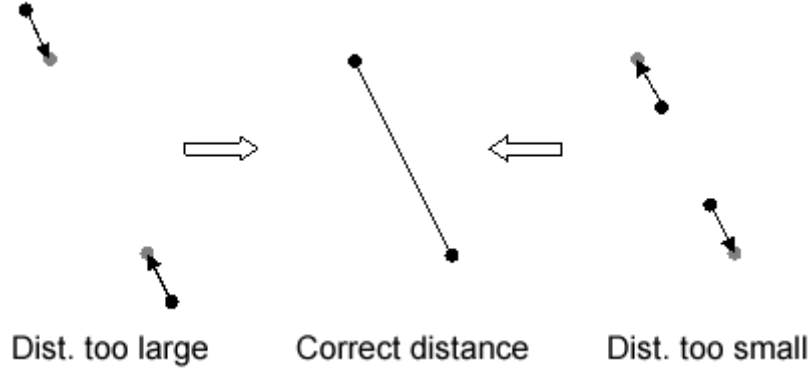
Figure 2.8: Constraint relaxation (Jakobsen 2005, p. 1)

$$\Delta x1 = -\frac{w1}{w1+w2}(\|\,x2-x1\,\| - d)\frac{x2-x1}{\|\,x2-x1\,\|}$$
$$\Delta x2 = \frac{w2}{w1+w2}(\|\,x2-x1\,\| - d)\frac{x2-x1}{\|\,x2-x1\,\|}$$

(2.11)

where:

w1 and w2 are the weights of the respective particles

This method of iteratively checking constraints and relaxing them immediately is called the Gauss-Seidel method and has one main problem; relaxing one constraint may violate other, already examined, constraints and therefore multiple relaxation passes may be needed. According to Jakobsen (2005) however, for most cloth only one relaxation pass is necessary for visually pleasing animation, although this is contradicted by (Kim 2011), who suggests that multiple passes are needed when using a Gauss-Seidel approach.

A PBD approach is unconditionally stable, as any instability will be corrected by the relaxation process, and can therefore use a large time step to reduce the performance cost of the simulation. Whilst implicit integrators, discussed below, offer unconditional stability for the traditional Mass-Spring models, they are much more difficult to implement, so the fact that PBD are stable and simple explains their popularity for use in video games. The need for multiple Gauss-Seidel passes can be used to control the performance hit of the cloth as well; the amount of frame time available for cloth simulation could be used to adjust the number of relaxation passes, so that if more time is available, more accurate cloth is produced.

Despite its advantages, PBD was not chosen for this project as it is somewhat of a closed problem for cloth simulation in video games. PBDs are the standard method of animating cloth for games, and have

been implemented efficiently, and parallelised, in commercial physics engines. By contrast, the Provot Mass-Spring model is not conventionally used, as it may be too expensive for certain mesh sizes. However, the research into the performance of the Provot model is either old, or makes no reference to the hardware used, and therefore there is scope for further work to investigate whether performance is still a concern with modern hardware.

2.1.4 Numerical Integration for Mass-Spring Models

As mentioned above, physically-based models for cloth simulation require solving a series of differential equations, discretised in space and time. According to Wacker, Thomaszewski, and Keckeisen (2005, p. 5), "since particle systems already represent a disrectisation in space, only a system of ordinary differential equations has to be solved". For the Mass-Spring model using Newtonian mechanics a series of second order ODEs, of the form 2.12, must be solved (Zink and Hardy 2007, p. 5).

$$\frac{\delta^2 x}{\delta t^2} = M^{-1} F(x, v) \tag{2.12}$$

This can be converted into a coupled series of first order ODEs by separating the position and velocity (ibid., p. 5):

$$\frac{\delta}{\delta t} \begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v \\ M^{-1} F(x, v) \end{pmatrix} \tag{2.13}$$

Equations 2.12 and 2.13 cannot be solved analytically, and therefore it is necessary to use a numerical method, or integrator, to approximate them at discrete time intervals.

There are many integration methods that could be chosen, typically classified as either explicit or implicit, and the most popular choices will be described now.

2.1.4.1 Explicit Integrators

Most of the early work on physically-based cloth simulation used explicit integrators as they are simple and easy to implement; they only require information about the state of the system at the previous interval to calculate the current state.

The most commonly used explicit integrators are the Runge-Kutta family of integrators.

2.1.4.1.1 Euler

The first order Runge-Kutta integrator, or explicit Euler, was used in Provot (1995) to approximate a Mass-Spring system.

Equation 2.13 is approximated by (Wang, Hu, and Zhuang 2009, p. 3):

$$v_{i+\Delta t} = v_i + \Delta t F(t_i, v_i)$$

where:

v_i is the velocity of a particle at time interval i

Δt is the time step

(2.14)

When applied to the Provot Mass-Spring model, this gives (Provot 1995, p. 3):

$$a_{i,j}(t + \Delta t) = \frac{1}{m_{i,j}} F_{i,j}(t)$$

$$v_{i,j}(t + \Delta t) = v_{i,j}(t) + \Delta t a_{i,j}(t + \Delta t)$$

$$x_{i,j}(t + \Delta t) = x_{i,j}(t) + \Delta t v_{i,j}(t + \Delta t)$$

where:

$a_{i,j}$, $m_{i,j}$, $v_{i,j}$ and $x_{i,j}$ are the acceleration, mass, velocity and position of particle i, j respectively

(2.15)

The explicit Euler method is computationally cheap but can result in numerical instability if too large a time step is used. Mathematically, the explicit Euler method is stable only if the time step is less than the natural period of the system, approximated as $\pi \sqrt{\frac{m}{K}}$, where K is the maximum stiffness in the system. Vassilev, Spanlang, and Chrysanthou (2001, p. 2) found that in fact explicit Euler is only stable for Δt values less than $0.4\pi \sqrt{\frac{m}{K}}$.

As cloth generally does not stretch easily, this results in high stiffness in the structural and shear springs, which necessitates the use of a small time step if the explicit Euler integrator is chosen. This can impact the overall performance of the simulation, as while this integrator is cheap, the frequency of calculations is high as a result of the time step limitations.

2.1.4.1.2 Midpoint

The explicit Midpoint integrator is a second order Runge-Kutta method and modifies the Euler integrator to give greater stability.

Equation 2.14 is modified to give the following (Wang, Hu, and Zhuang 2009, p. 3):

$$v_{i+\Delta t} = v_i + \Delta t F\left(t_i + \frac{\Delta t}{2}, v_i + \frac{\Delta t}{2} F(t_i, v_i)\right)$$

(2.16)

Since this method requires two derivatives, the computational cost is greater than Euler. However, because the midpoint method affords greater numerical stability (see Fig 2.9), a larger time step can be used which increases the overall simulation performance; Wang, Hu, and Zhuang (ibid.) have shown that the midpoint integrator offers close to twice the simulation performance over explicit Euler.
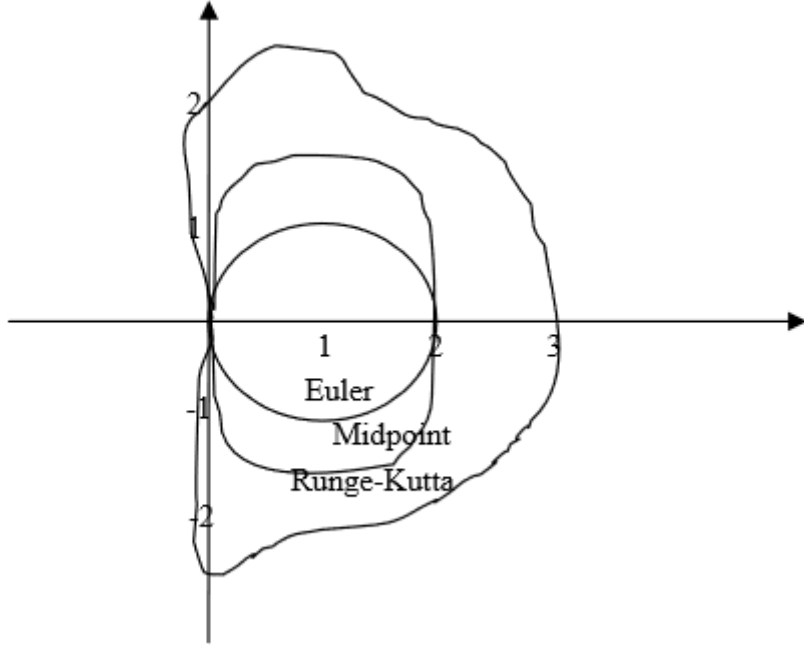
Figure 2.9: Explicit integrator stability regions (Wang, Hu, and Zhuang 2009, p. 4)

2.1.4.1.3 Fourth order Runge-Kutta

The Fourth order Runge-Kutta (RK4) integrator offers greater stability using larger time steps over the midpoint method and is formulated as (Wang, Hu, and Zhuang 2009, p. 3):

$$v_{i+\Delta t} = v_i + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = F(t_i + v_i)$$

$$k_2 = F\left(t_i + \frac{\Delta t}{2}, v_i + \frac{\Delta t}{2}k_1\right)$$

$$k_3 = F\left(t_i + \frac{\Delta t}{2}, v_i + \frac{\Delta t}{2}k_2\right) \tag{2.17}$$

$$k_4 = F\left(t_i + \Delta t, v_i + \frac{\Delta t}{2}k_3\right)$$

This integrator has a significantly higher computational cost than the other methods discussed, however Volino and Magnenat-Thalmann (2001) have shown that the RK4 supports time steps almost six times larger than the midpoint method. Therefore, given that RK4 is three times as computationally expensive as midpoint, this suggests that this integrator can lead to twice the overall simulation performance. Wang, Hu, and Zhuang (2009) have also shown that RK4 offers performance gains over midpoint.

2.1.4.1.4 Verlet

Verlet integration is an alternative to the Runge-Kutta family of integrators. It avoids velocity calcula-

tions by approximating the velocity of a particle using its previous positions.

A particle's new position is approximated by (Mongus et al. 2012, p. 2):

$$x_{i+\Delta t} = 2x_i - x_{i-\Delta t} + a_{i+\Delta t}\Delta t^2 \qquad (2.18)$$

Since Verlet is a velocity less integrator, the linear damping shown in Equation 2.8 will not provide any damping to the system. In order to prevent the system from oscillating infinitely, a damping factor is applied to the approximated velocity, thus Equation 2.18 becomes

$$x_{i+\Delta t} = x_i + dampingFactor(x_i - x_{i-\Delta t}) + a_{i+\Delta t}\Delta t^2 \qquad (2.19)$$

This method is computationally fast and reasonably stable, as "velocity is implicitly given and consequently it is harder for velocity and position to come out of sync" (Jakobsen 2005, p. 1). However it does still suffer from time step issues; figures 11 and 13 in Wacker, Thomaszewski, and Keckeisen (2005, pp. 14-15) show that below a certain threshold Verlet integration is less stable than explicit Euler, but more stable over that threshold.

2.1.4.2 Implicit Integrators

According to Baraff and Witkin (1998, p. 1), "Explicit methods are ill-suited to solving stiff equations because they require many small steps to stably advance the simulation forward in time". Therefore they propose an implicit integrator for use in cloth simulation since implicit integrators are unconditionally stable regardless of step size.

However implicit integrators are computationally more expensive than their explicit equivalents, as "they involve the resolution of a large and sparse linear equation system for each iteration" (Volino, Cordier, and Magnenat-Thalmann 2005, p. 4). Since they are unconditionally stable however, this can be countered by simply using a larger time step. The guaranteed stability of these methods also reduces the accuracy of the simulation, as they introduce inherent numerical damping, which increases as the time step increases (see Volino and Magnenat-Thalmann (2001, p. 4)). As such, there is a balance to be found between performance and accuracy of the simulation with implicit integrators.

The most common implicit integrator is the implicit, or backward, Euler method which will be described now. It should be noted that there are many other implicit integrators available.

2.1.4.2.1 Euler

First proposed for use in cloth simulation in Baraff and Witkin (1998), the implicit, or backward, Euler

method is an adaptation of the explicit Euler method.

Equation 2.15 is modified to give (Kang, Choi, and Cho 2000, p. 3):

$$v_i^{t+\Delta t} = v_i^t + F_i^{t+\Delta t} \frac{\Delta t}{m_i} \tag{2.20}$$

$F_i^{t+\Delta t}$ cannot be calculated at the current time step, and so must be approximated as (ibid., p. 3):

$$F^{t+\Delta t} = F^t + \frac{\delta F}{\delta x} \Delta x^{t+\Delta t} \tag{2.21}$$

$\Delta x^{t+\Delta t}$ can be written as $\Delta t(v^t + \Delta v^{t+\Delta t})$ and so Equation 2.21 can be rewritten, giving the series of linear equations (ibid., p. 3):

$$\left(I - \frac{\Delta t^2}{m} \frac{\delta F}{\delta x}\right) \Delta v^{t+\Delta t} = F^t \frac{\Delta t}{m} \tag{2.22}$$

where:

I is the identity matrix

Thus, implicit Euler involves calculating $\Delta v^{t+\Delta t}$ every iteration using:

$$\Delta v^{t+\Delta t} = \left(I - \frac{\Delta t^2}{m} \frac{\delta F}{\delta x}\right)^{-1} F^t \frac{\Delta t}{m} \tag{2.23}$$

$\frac{\delta F}{\delta x}$ is the negated Hessian matrix, denoted as H, and can be approximated as (ibid., p. 3):

$$H_{ij} = \begin{cases} k_{ij} & \text{if } i \neq j \\ -\sum_{i \neq j} k_{ij} & \text{if } i = j \end{cases} \tag{2.24}$$

where:

k_{ij} is the spring stiffness

As a result, the implicit Euler method requires the computation of an n x n matrix each iteration, and thus increasing the size of the mesh greatly impacts the performance of this method.

However, if the stiffness of the springs and mass of the particles are constant throughout the simulation then $\left(I - \frac{\Delta t^2}{m} H\right)^{-1}$ can be precomputed, giving performance gains.

This method has been shown by Volino and Magnenat-Thalmann (2001) to be stable for any time step value, however as the time step increases, the accuracy decreases rapidly over a certain threshold. Using the method described above, it is also not possible to use an adaptive time step or to vary the mass or stiffness of the model as the cost of computing H every iteration is high. As such, many researchers, such as Mesit, Guha, and Chaudhry (2007) and Kang, Choi, and Cho (2000), have presented faster approximation methods for the implicit Euler integrator.

2.1.4.3 Chosen Integration Methods

Four integrations methods were chosen for investigation by this project.

- Explicit Euler. This is the simplest integrator and one of the most popular in the literature but it is also most likely to impact simulation performance due to its reliance on small time steps

- Midpoint. Computationally more expensive than explicit Euler, but stable with larger time steps, therefore offering an interesting comparison between computational cost and simulation performance

- Fourth order Runge-Kutta

- Verlet. Chosen as it is an explicit integrator that is not part of the Runge-Kutta family

Only explicit integrators have been selected as they are considered to be inappropriate when solving stiff differential equations, such as those in cloth, due to the necessity of small time steps.

Chapter 3

Design and Implementation

3.1 Development Methodology

Software development methodologies break the development of a piece of software down into a number of phases with the aim of improving design and code quality as well as aiding in project management. There are a large number of different development methodologies, although they can typically be classified as either sequential, also called waterfall, or cyclical, also called spiral. The choice of which method to use will depend on the specifics of the software project as well as developer preference and team size.

3.1.1 Sequential

The sequential, or waterfall, approach uses a number of strict, sequential phases for developing software; each phase must be completed before the next phase can begin. Fig. 3.1 shows the typical structure of this methodology.

This approach works well for projects were the requirements are strictly defined, the project's subject area well known and the technologies involved well understood by the developers. In this case, the software can be designed well from the start, so there is little need for iterative development. If the project's requirements are likely to change, or the subject area and technologies are ill-understood or new then a waterfall approach is inappropriate. The rigid structure of the waterfall method does not allow requirements to be easily changed, or poor design choices corrected; in order to change requirements or correct poor design the whole development process must be started again, as there is no method of feedback between development phases.
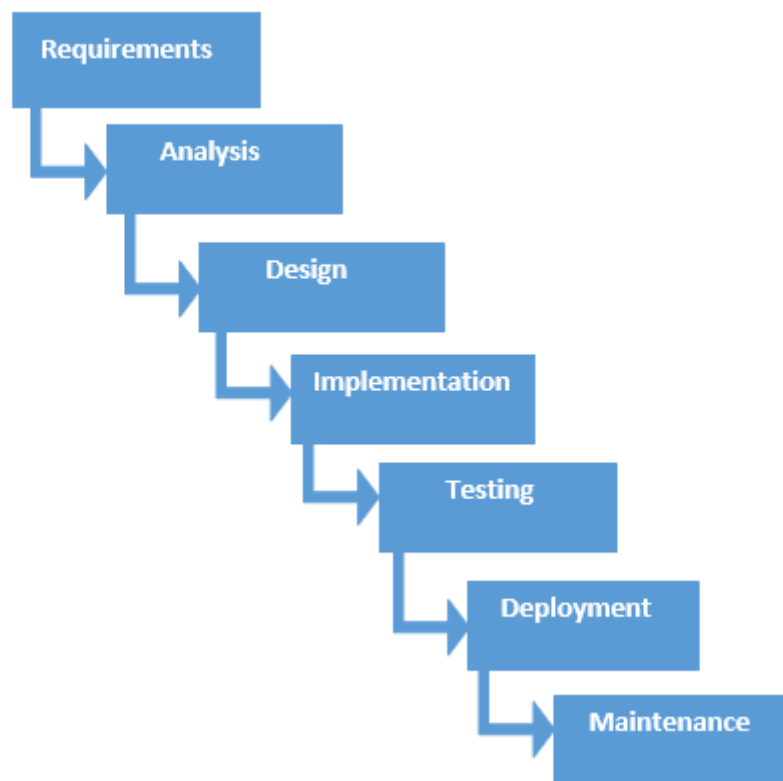
Figure 3.1: Waterfall development methodology (Naveen 2015)

3.1.2 Cyclical

Cyclical approaches, as the name suggests, develop software using iterative cycles. At the start of each cycle requirements and design are set and at the end, a working software prototype is produced. This prototype is used as feedback for the next cycle, adjusting the requirements and correcting design mistakes and issues as necessary. These methods counteract the disadvantages of the waterfall approach; by using an iterative approach it is easy to add or change requirements or redesign areas of the project.

There are many development methods which are derived from a cyclic approach, including Rapid Application Development (RAD) and Scrum.

RAD is a development method which favours rapid prototyping with minimal planning. Since there is less focus on planning at each iteration it is extremely easy to incorporate design and requirement changes. Prototype development will start earlier than other methods and this enables issues to be caught early, and the design to be altered appropriately. There is a requirement when using RAD of skilled and experience developers and designers, as the prototypes developed must be reusable so as to avoid wasting time.

Requirement	Type
Must model cloth using the Provot Mass-Spring model	Functional
Must be able to render cloth to the screen	Functional
Must be able to pin particles so they are unaffected by forces	Functional
Must perform in real time	Non-functional

Table 3.1: First cycle requirements

Using Scrum, a project is broken down into sprints and each sprint is timeboxed with a duration, i.e. 24 hours, 4 weeks. At the end of each sprint a working prototype is produced and as the sprints are completed the project is designed and developed over time.

3.1.3 Chosen Methodology

For this project, cyclical methods were preferred over waterfall. This was partly due to the author's preference for cyclical methods and partly because there were some unknowns, particularly related to rendering, that made rigidly designing the system difficult. Using the more agile cyclical methods allowed for some investigative work before finalising the system design.

In particular, RAD was chosen as the development methodology.

3.2 Development

Since RAD was the chosen development methodology, the artefact was developed over several cycles.

3.2.1 First Cycle

In the first cycle the Mass-Spring model was implemented. Springs were implemented using the Hooke equation (Equation 2.7) to calculate the internal spring force with linear damping as an additional internal force using Equation 2.8. Gravity, implemented as Equation 2.5, was the only external force. Rendering of the cloth was also implemented.

Table 3.1 shows the initial requirements of the first development cycle.

Before designing this cycle, an initial investigation into the rendering component was carried out, as the choice between DirectX11 or OpenGL would affect the design of the system.

As this project is not concerned with rendering cloth, but simulating it, it was decided that the cloth

would be rendered as a mesh of lines, representing the structural and shear springs (as in Figs 2.4 and 2.7). Since the positions of the particles, and therefore vertices, evolve over time, a DirectX11 implementation would have to use a dynamic vertex buffer and remap the vertex data every frame. This is not an issue in OpenGL, since it is possible to draw vertices directly with the glVertex3f function. It was unknown by the author what the cost of this remapping would be, so simple OpenGL and DirectX11 implementations were created and their performance compared. For a 500^2 mesh, with a debug build, the OpenGL implementation rendered the structural and shear springs at 80FPS and the DirectX11 version at 450FPS. Hence, DirectX11 was chosen as the rendering language. It should be noted that the OpenGL implementation was very naive, due to the author's inexperience, and this is most likely the reason for the poor performance. It is probable that a more robust implementation, using more advanced features, would perform much more closely to the DirectX11 implementation.

The initial design can be seen in Fig A.1; the rendering component, constructors, accessors and mutators have been excluded for brevity.

Since DirectX11 was the rendering language, DirectXMath types were used for the key variables in the Particle class. This gives performance gains over manually implementing a 3-dimensional vector and functions, as the functions that operate on DirectXMath types are compiled into SIMD instructions, allowing the calculations to be completed in fewer CPU cycles.

An std::vector was initially used to store the springs as it is not necessary to separate the springs into types, since the internal forces are calculated the same for every spring type. Also, the equations for calculating the amount of each spring (see Equation 2.2) were unknown during the initial design process, so an std::vector was chosen as it would allow an any number of springs to be stored.

During implementation, the initial design had to be adapted as a result of performance concerns; these optimisations will be discussed in the section 3.3.

Fig A.2 shows the final design of this cycle with the optimisation changes. Timing code was also added to allow performance data necessary for testing the hypotheses to be extracted from the cloth. The test plan for this project will be detailed in the next chapter. Code for automating the testing process using an XML file was also added, again, details of this will be discussed in the next chapter.

Requirement	Type
Must implement the Explicit Euler and Verlet integrators	Functional
Must be able to dynamically switch integrators	Functional

Table 3.2: Second cycle requirements

3.2.2 Second Cycle

During the second development cycle the model developed in the first cycle was adapted to use different integration methods. The explicit Euler and Verlet integrators were implemented as described in 2.1.4.1. Wind was also added as an additional external force.

The requirement for this development cycle are shown in Table 3.2.

The initial design of this development cycle can be seen in Fig A.3. For brevity, only the changes introduced for this cycle are shown for the classes developed previously; it can be assumed that the Particle and Cloth classes are unchanged other than the documented additions.

Two separate designs were initially proposed, one using static classes and storing a function pointer for the appropriate integrator, and the second using an interface and virtual functions. For the interface design, the inheriting classes were designed using the Singleton design pattern. Since particles are passed as parameters to the integrate function, there is no need for each particle to have their own instance of a particular integrator. Therefore it makes sense for integrators to be Singletons, and this helps to reduce the memory footprint of the application as well as the cost of dynamically switching integrators; as each integrator only needs to be dynamically allocated once, rather than for every particle.

The two different designs were suggested as the aim was to use the most efficient implementation possible and it was unknown during the design process which would be more efficient. As such, both designs were implemented and then some tests run using the explicit Euler integrator to determine which was most efficient. Compared with implementing explicit Euler directly in Particle, both implementations increased total update time by approximately 0.4ms for a 50^2 mesh on a debug build. This translated to an approximate drop of 4-5 FPS, and was expected due to the fact that both designs must call accessors to use Particle's data.

Compared with each other, the results were more variable. For some runs the static class design was more efficient and for others the interface design more so. Ultimately, the efficiency difference was

small, at most a 0.1ms difference between them, so the conclusion was reached that both designs were equally efficient. However, the explicit Euler integrator is the cheapest integrator, so for more expensive integrators, such as RK4, a wider performance gap may be found. Therefore these experiments will be run again during the third development cycle using RK4 and the design retroactively changed if a significant efficiency gulf is found.

Since both designs were equally efficient, the interface design was chosen as the author felt that it was, ostensibly, a better quality approach than using static classes. Using interfaces means Particle conforms much more to the Open/Closed SOLID principle; Since Particle stores a pointer to the interface, and only calls virtual functions, new integrators can easily be added and used without any changes needed in Particle.

One minor optimisation was made to the initial design and is documented in the next section. As a result of this optimisation, the chosen design now performs almost as well as coding the integrator directly in Particle.

Wind was originally intended to be calculated using a static normal assigned at Particle creation, however on reflection it was realised that this would not work, as the normal of the particles will change as they move over time. Therefore it is necessary to calculate the surface normal for the particles every time step. Using the code downloaded from Mosegaard (2009) as a reference, the particles are split into triangles and the normal calculated with the equation below.

$$\overrightarrow{P1ToP2} = p2 - p1$$
$$\overrightarrow{P1ToP3} = p3 - p1 \tag{3.1}$$
$$\overrightarrow{normal} = \overrightarrow{P1ToP2} \times \overrightarrow{P1ToP3}$$

Equation 2.6 had to be modified slightly as well in order to give a vector, rather than a scalar, force. Again, using Mosegaard (ibid.) as a reference, the equation was modified to:

$$F_{wind} = n_{ij}(w(\| n_{ij} \| \bullet \overrightarrow{W})) \tag{3.2}$$

This change is support by Provot (1995), who arrives at a similar equation for calculating a wind force.

The final design can be seen in Fig A.4. In addition to the optimisation changes, another change was made as a result of poor understanding of the integration equations. Initially the design was to store the time step in the integrator classes and have the time step checking code in the integrate function. The

algorithm would then proceed as follows for every frame:

```
/* In update */
```

timeSinceLastIntegration += deltaTime

for *every spring* **do**
| calculateInternalForce

end

for *every particle* **do**
| addGravity

end

if *timeSinceLastIntegration >= timeStep* **then**

    for *every particle* **do**
    | do integration calculations

    end

    timeSinceLastIntegration = 0.0

end

Algorithm 1: Original integration algorithm

Thus, forces were calculated and added every frame, resulting in a huge performance decrease as mesh size increased and overly elastic cloth as the time step was increased so the integration wasn't calculated every frame. By examining the integration equations again, it was realised that calculating the forces should be included as part of the integration step, and therefore only performed at every time step, rather than every frame. As such, the timeStep variable was moved from the integrators to a global variable, and the time step checking code moved to the global update function (not shown in the design diagrams). Now, Cloth's update function is called every time step, rather than every frame, and the cloth behaves as expected as the time step is varied.

3.2.3   Third Cycle

In the final development cycle the Midpoint and RK4 integrators were implemented.

The initial design is shown in Fig A.5. As with the second phase, only those changes introduced in this phase are shown.

In order to implement both Midpoint and RK4, it was necessary to make additional changes to the algo-

rithm flow. Following the changes detailed in the previous section, the cloth animation algorithm would proceed as follows every time step:

```
/* In Cloth.update */
```

for *every spring* **do**
  | calculateInternalForce

end

for *every particle* **do**
    addGravity

    if *flag scenario* **then**
    | addWind

    end

    do integration calculations
end

Since both integrators require multiple derivations, i.e. multiple calculations of the total force, it was necessary to redesign the system.

The integrate function is now passed a Cloth object rather than a Particle and Cloth has a new function, calcForces, which calculates the total force in the mesh. This allows an integrator to calculate the total force multiple times if necessary, but requires it to loop through every particle in order to apply the appropriate integration calculations. As such, the integrator classes are also friend classes of Cloth, allowing them access to the particles array directly. This breaks the Open/Closed principle slightly, so removing the friend relationship and requiring integrators to call accessors may be a better approach. The friend design was kept in order to give as much performance as possible, but it should be noted that caching the particle array in a local variable would lead to only one accessor call per time step, which would most likely result in near identical performance to the friend approach.

Since integrate is now passed a Cloth object, the chosen integrator is stored outside of Cloth and Particle, in the global encapsulating class (not shown in the diagrams for brevity). Integrate is then called every time step in place of Cloth's update method.

As mentioned in the previous section, the performance of the static class and interface designs were compared again, in order to see if a larger performance delta appeared with a higher cost integrator. The results of this comparison were similar to those of the second cycle. The results were still very variable,

but ultimately there was less than a frame difference between both implementations. Therefore, the interface design was kept and the final design can be seen in A.6.

Unit testing was also added during this cycle, in order to validate the integrator implementations. The force, velocity (where appropriate) and positions of the particles at the end of an integration step are compared with values calculated manually; an accuracy of four decimal places is used to account for the small floating point rounding errors inherent in computing.

3.3 Profiling and Optimisation

3.3.1 First Cycle

Several optimisations were made to the design of the first development cycle as a result of profiling the application.

Firstly the std::vector was replaced with dynamically allocated arrays in the Cloth class. Iterating through the vector to calculate the spring forces proved prohibitively expensive for meshes over a certain size, even when calcSpringForce was an empty function; a 100^2 mesh was the largest mesh that supported real time frame rates, running at 40FPS on a debug build. Profiling showed that std::vector iterator functions were the most expensive execution path, as a range-based for loop was used to iterate over every spring. This can be seen in Fig B.1 in the appendices. Hence, the vector was replaced with dynamically allocated arrays and the equations listed in 2.2 identified. Fig B.2 shows the profiling results using dynamic arrays. As can be seen, the high cost execution paths have moved away from iterating over the springs and into calculating the spring forces themselves.

Switching to arrays gave a 4x FPS increase, for the same mesh size, and a 2x increase in the maximum real time mesh size.

Secondly, profiling showed there were optimisations to be made in calcSpringForce.

The profile in Fig B.3 shows that calcSpringForce was the most expensive code path, and within it, addForce and XMLoadFloat3 were the most expensive functions. This results from the fact that XMFLOAT3s are used in Particle for variables such as position. XMFLOAT3s must be converted into XMVECTORs using XMLoadFloat3 in order to be able to use the DirectXMath vector functions. There-

fore, the position and velocity of every particle had to be converted every frame in order to implement Equations 2.7 and 2.8. Similarly, addForce must call XMLoadFloat3 and XMStoreFloat3 to first convert totalForce into an XMVECTOR for use in addition, and then convert the resultant XMVECTOR back into an XMFLOAT3. Again, this had to be done every frame, hence why Fig B.3 shows that XMLoadFloat3 and XMStoreFloat3 are the two most expensive functions in the entire system. As a result, XMVECTORs were used instead of XMFLOAT3s, removing the need for conversions, and giving reasonable performance gains.

Fig B.3 also shows that calls to the overloaded subtraction and multiplication operators were other expensive code paths. These simply call the appropriate DirectXMath function, and so calls to operators were replaced with direct calls to the DirectXMath function. For example,

```
XMVECTOR length = p1->getPosition() - p2->getPosition();
```

became

```
XMVECTOR length = XMVectorSubtract(p1->getPosition(), p2->getPosition());
```

Fig B.4 shows the profiling results after these changes were added. The most expensive code paths now are all DirectXMath functions directly involved in calculating Equations 2.7 and 2.8.

Implementing both of these changes awarded performance gains of 40FPS for a debug build with a 50^2 mesh.

A final optimisation step was changing the accessors and mutators in Particle to return and pass by reference. This improved performance slightly, giving gains of roughly 5FPS.

3.3.1.1 Second Cycle

One optimisation was made to the initial design of cycle two.

The VerletIntegrator and ExplicitEulerIntegrator classes were made friend classes of Particle. This allows the integrators to access Particle's private member variables directly, removing the overhead of having to call accessors. This change resulted in small performance increases for both integrators, reducing overall update time be approximately 0.3ms, for a 50^2 mesh on a debug build, which translated to an increase of 3-4FPS. This increase was also enough to move from unstable to stable cloth when using Verlet integration.

Chapter 4

Test Plan

4.1 Test Data

To evaluate the hypotheses, a number of data fields will be captured from the simulation

- Simulation frame rate. This will be affected by the cost and frequency of the integration calculations and is essential to capture to determine if the simulation is running in real time

- Average time spent in the various functions such as calcSpringForce and update. Extracting these timings will give a better understanding of where the time each frame is spent

The data mentioned above will be extracted by simulating the cloth in two different scenarios.

The first, or sheet, scenario will simulate a sheet hanging from a washing line. The top left and right particles of the cloth will be pinned, so as to be unaffected by any forces, and gravity applied as the sole external force.

The second scenario will simulate a flag on a flag pole flapping in the wind. Particles on the left edge of the cloth will be pinned with gravity and a wind force acting as external forces. The addition of wind will result in much more movement in the cloth so this scenario provides data on performance of a more active simulation.

4.2 Test Parameters

Mass-Spring models for cloth simulation have a number of different parameters which affect realism and performance, and are notoriously difficult to tune.

4.2.1 Mesh Size

This is the discretisation of the cloth, i.e. the number of particles used to represent the cloth. It is represented by two integers representing the number of rows and columns; the total number of particles in the cloth is therefore *rows × columns*.

Increasing mesh size will result in a more realistic simulation but there will be direct impacts on performance as a result. Adding more particles adds more springs, and therefore more calculations are needed every time step to calculate the internal forces. Adding more particles also increases the number of integration calculations needed every time step and both of these factors combined will result in a performance hit for the simulation.

Several different mesh sizes will be used when evaluating the project hypotheses. Data will be extracted for each integrator at each mesh size. This will allow a comparison of the performance impact of the integrators when mesh size is varied.

4.2.2 Integrator Time Step

The time step will vary from integrator to integrator and must be set carefully in order to maintain the stability of the cloth.

Varying the time step will affect the performance of the simulation as it will vary the frequency with which the integration calculations must be performed.

The maximum stable time step will be used along with a number of smaller time steps. This will allow a direct comparison of performance impact as time step decreases; the maximum time step value will be found by investigating the point at which the integrators become unstable.

4.2.3 Spring and Damping Coefficients

Varying the spring and damping coefficients of the springs will affect the realism of the simulation. If the stiffness is set too low, then the particle displacement will increase and may result in unrealistic deformations of the cloth. On the other hand, if the stiffness is set too high then this can lead to no displacement in the cloth at all. For damping, if it is set too low then the cloth will oscillate too much and appear too elastic, if set too high then the cloth will appear as if moving through a viscous fluid, such as oil.

Varying stiffness may also have an impact on the simulation's performance; increasing the stiffness may

require the use of smaller time steps in order to maintain stability, depending on the chosen integrator.

4.2.4 Particle Mass

Since this project is not concerned with truly accurate cloth modelling, the particles' mass will be defined uniformly; that is, a total mass will be defined for the cloth and then divided by the number or particles to give the mass of an individual particle.

Varying the mass of the particles may have an impact on simulation performance. As a result of Equation 2.5, increasing the mass of a particle will increase its displacement due to gravity and therefore, the stiffness of the springs may need to be increased in order to maintain the realism of the simulation.

Since this project is primarily concerned with measuring the performance of different integrators, mesh size and time step are the key parameters. As such, mass and spring and damping coefficients will be kept the same for each mesh size. In this way only one parameter will change between tests and therefore the true effect of the parameter on performance can be measured. The downside of keeping mass and the coefficients constant is that it may not lead to the most realistic cloth, however this is acceptable because the focus is on measuring performance rather than appearance.

4.3 Test Process

As the previous section details, the first stage of the testing process will be investigations into suitable values for the various test parameters. Following this, the test data will be extracted for every combination of mesh size and time step, in both the flag and sheet scenarios.

This, second, testing phase will be automated, using an XML file to detail the specific integrator and test parameters to use. The XML file will take the following form:

```xml
<test_list>
  <test id=''''>
    <integrator type='''' time_step=''''/>
    <cloth_params height='''' width='''' mass='''' wind_constant=''''>
      <top_left_position x='''' y='''' z='''' />
      <mesh_size rows='''' columns='''' />
      <structural spring_coefficient='''' damping_coefficient='''' />
```

```xml
            <shear spring_coefficient='''' damping_coefficient='''' />

            <flexion spring_coefficient='''' damping_coefficient='''' />

            <wind direction x='''' y='''' z='''' />

        </cloth_params>

    </test>

</test_list>
```

Each test element extracted from the test_list will be run in both the flag and sheet scenario. Scenarios will be run for a maximum of one minute, and when the run is completed, the appropriate test data will be extracted and stored in a CSV file. In total, all tests will be run six times, in order to sample as much data as possible and attempt to smooth any anomalous results.

Chapter 5

Data Analysis and Evaluation

This chapter will present and analyse the data gathered with the test plan detailed in the previous chapter. It will also evaluate the project hypotheses using the gathered data.

5.1 Chosen Parameters

The first stage of the data collection process was to choose suitable values for the parameters detailed previously.

5.1.1 Mesh Size

The maximum mesh size used was 300^2; this is the largest mesh size the can be run at real time frame rates with the most expensive integrator (RK4). This maximum was decreased by 50 in both dimensions, to a minimum size of 50^2, to give the six mesh sizes below.

- 300^2

- 250^2

- 200^2

- 150^2

- 100^2

- 50^2

5.1.2 Time Step

The maximum time step was 20ms. This was the largest time step where at least one integrator was still stable for a 50^2 mesh. This value was decreased by 5ms giving the range of five time steps listed below.

- 20ms

- 15ms

- 10ms

- 5ms

- 1ms

5.1.3 Mass and Spring and Damping Coefficients

These parameters were kept constant for every mesh size to avoid changing more than one variable between tests. Values were chosen that gave a reasonable cloth appearance. There are some visual issues, such as too much oscillation, with the values chosen at the larger mesh sizes but this is acceptable due to the focus of the project. The values used are as follows:

- Mass = 100

- Structural Stiffness = 20

- Structural Damping = 7.5

- Shear Stiffness = 20

- Shear Damping = 7.5

- Flexion Stiffness = 5

- Flexion Damping = 2.5

For Verlet, a constant damping factor of 0.5% was used.

5.2 Results

5.2.1 Explicit Euler

Fig C.1 shows the average simulation frame rate for every time step at every mesh size, in the sheet scenario. Only data for stable simulations are shown. The graph shows two things, firstly that as mesh size increases average frame rate decreases and secondly that as time step increases average frame rate increases.

The data shows that varying mesh size can have a dramatic effect on the frame rate, this is shown well by Fig C.2. This graph uses the average FPS for a 1ms time step, so that only one variable is changed. It has a strong negative correlation, with a Pearson's correlation coefficient of -0.8751. Also, it clearly shows a dramatic frame rate decrease as mesh size is increased to 100^2 and 150^2; for a 100^2 mesh the decrease is approximately 1.4 times and for 150^2 approximately 6.8 times.

The decrease in frame rate can be explained by examining where the time each frame is spent. Fig C.6 shows that the majority of the frame time is spent in the update function for a 300^2 mesh using a 1ms time step. This is the worst case test, but the conclusion is reflected across all other mesh sizes and time steps. By plotting the average time spent in the update function against mesh size, it is possible to explain why the frame rate decreases. Fig C.4 shows this graph; as with Fig C.2 it shows data for a 1ms time step. This shows a strong positive correlation, of strength 0.9626, thus it can be concluded that update time increases with mesh size. If the time for each update increases with mesh size, then the total time for each frame must increase also, thus reducing the frame rate of the simulation.

Digging a little deeper, Fig C.7 shows that calculating the internal forces, i.e. the spring forces, is the most expensive part of each update. By plotting this against mesh size, shown in Fig C.5, a strong positive correlation is again observed, with Pearson's coefficient 0.9302. This is easy to explain. As mesh size increases, the number of particles increases, and therefore the number of springs also increases. More springs means that Equations 2.7 and 2.8 must be calculated more, hence the time spent on internal forces increases.

The increase in update time with mesh size also explains the increase in frame rate as the time step increases. As the time step increases, the expensive update function is called less frequently, so overall frame time is decreased, thus increasing the frame rate. Fig C.3 shows the average frame rate plotted against the time step for a 300^2 mesh. This graph has a Pearson's correlation of 0.8955, thus supporting

that frame rate increases with time step. It also shows that frame rate only increases once the time step exceeds some threshold. Again, this result can be explained by examining the average update time. For the 300^2 mesh, the average update time was approximately 6ms, and frame rate only increased once the time step reaches 10ms. The reason for this should be obvious; both the 1ms and 5ms time steps are less than the total update time, therefore the update function will still be called every frame.

As mentioned earlier, only those simulations that are stable are shown in Fig C.1. From this graph, and the data in Table C.2 it is obvious that explicit Euler is quite unstable. It is only really stable for a 1ms time step, apart from for a 50^2 mesh, where it is stable for a 5ms step.

The data for the flag scenario shows similar correlations to the sheet scenario. Figs C.14 and C.15 show that as mesh size increases FPS decreases, with a strong negative correlation of strength -0.8163. The same initial dramatic decreases as the sheet scenario are still present; the decrease for a 100^2 mesh is approximately 2.8 times and approximately 4.9 times for 150^2. As with the sheet scenario, the majority of the frame time is still spent within update (see Fig C.19) and Fig C.17 displays a strong positive correlation as well, with Pearson's correlation of 0.9682.

The frame rates for the flag scenario are slightly lower than those for the sheet, this is explained by looking at Fig C.20. It shows a slightly different time breakdown within the update function. Calculating the internal forces is still the most expensive part, but the cost of calculating external forces has risen significantly over the sheet scenario; an approximate increase of 6.8 times. This is because the flag scenario includes wind as an additional external force. In order to apply wind, the particles must be split into triangles and the surface normal of every triangle calculated at every time step.

Figs C.14 and C.16 show that as time step increases the FPS increases with it, displaying a strong positive correlation of 0.9568. They also support the conclusion that frame rate only increases once the time step exceeds the total update time; again the frame rate for the 300^2 mesh only increased at 10ms, because the average update time was approximately 8ms.

The stability of the flag scenario tests, listed in Table C.4 are also similar to the sheet scenario, with the only difference being that a 5ms time step and 100^2 mesh was stable for the flag scenario.

Figs C.8 to C.13 and Figs C.21 to C.26 are the boxplots of the data for the explicit Euler integrator. They show that there is quite a lot of variance in the data gathered, particularly at lower mesh sizes. Mean values are used for the results detailed above, however examining the raw data, listed in Tables

C.1 and C.3, shows that the correlations are consistent for any particular test run.

Overall, explicit Euler is efficient. It easily supports the largest mesh size with a 1ms time step, running at 197 and 120FPS for the sheet and flag scenarios respectively, well over the 30FPS limit needed for a real time system. The update times for the same mesh are only 6 and 8ms, respectively, as well, which leaves approximately 27 and 25ms available each frame for other game related functions. The caveat is that it is only really stable for a 1ms time step, so the cloth will be updated frequently. However, as has been shown, the low cost of the integrator counters this disadvantage somewhat.

5.2.2 Verlet

The data for Verlet integration shows similar trends to explicit Euler.
Fig C.27 shows the average FPS decreases as mesh size increases for Verlet integration. This is supported by Fig C.28 which has a strong negative correlation with Pearson's coefficient of -0.8486. Both graphs also show a dramatic frame rate decrease as mesh size is increased to 100^2 and 150^2; approximately 2.1 and 6.5 times respectively with a 1ms time step. Again, this is explained by Figs C.32 and C.30 which show that the update function continues to be the most expensive code path and that update time has a positive correlation, of strength 0.9619, as mesh size increases. Calculating the internal forces continues to be the most expensive part of update, as shown in Fig C.33, and similar to explicit Euler, displays a positive correlation with mesh size (Fig C.31), with a Pearson's coefficient of 0.9679.
Fig C.27 also shows that frame rate increases as the time step increases. This is supported by Fig C.29 as it shows a positive correlation, with a coefficient of 0.8963, for FPS as time step increases. As with explicit Euler, both graphs show that the frame rate only increases once the time step exceeds some threshold. Again, this is because the update time for the 300^2 mesh is roughly 6ms.
The stability of Verlet can be seen in Table C.6 and Fig C.27. It shows that Verlet is much more stable than explicit Euler, being stable for every time step for the 50^2 mesh and even stable with a 5ms time step for 150^2 and 200^2 meshes. This increased stability may be a result of the damping factor added in Equation 2.19. Even with the small damping factor used (0.5%), the damping was much more noticeable than other integrators.

Similarly, the flag scenario also displays the same trends as the sheet scenario. Figs C.40 and C.41 both show a negative correlation for FPS as mesh size increases, with Pearson's correlation coefficient of -0.7709. Again large initial decreases are shown; approximately 4.2 times for 100^2 and approximately

4.8 times for 150^2. The majority of the frame time is still spent within update (see Fig C.45) and Fig C.43 has a Pearson's correlation coefficient of 0.9678 for update time.

As with explicit Euler, Fig C.46 shows that the cost of calculating external forces has risen significantly over the sheet scenario; approximately 6.8 times. This is reflected in the frame rate, and explains why the frame rates for the flag scenario are lower than the sheet scenario.

The trend that FPS increases with time step is also reflected in the flag scenario; Figs C.40 and C.29 show a positive correlation for FPS as time step increases, with a coefficient of 0.9579. Again, the same time step thresholds as explicit Euler are observed.

The stability of the flag scenario tests, listed in Table C.8 are identical to the sheet scenario, supporting the conclusion that Verlet is more stable than explicit Euler.

The boxplots for Verlet are shown in Figs C.34 to C.39 and Figs C.47 to C.52. Again there is quite a lot of variance in the data, particularly at lower mesh sizes. The correlations presented above however, are again reflected across individual tests in the data, listed in Tables C.5 and C.7.

Since Verlet is much more stable it offers performance advantages over explicit Euler. For example, for a 200^2 mesh, explicit Euler is limited to a 1ms time step which results in frame rates of 474 and 289 for the sheet and flag scenarios. By contrast, Verlet is stable for a 5ms time step leading to a frame rate of 6563 and 3574 for the two scenarios, an increase of approximately 13.8 and 12.4 times respectively. For mesh sizes over 200^2 however, Verlet is limited to a 1ms time step as well, so is more comparable to explicit Euler.

The data for the 50^2 mesh is quite anomalous. Comparing Verlet with explicit Euler reveals drastically different frame rates; this is shown well in Figs C.105 to C.106 and Figs C.111 to C.112. It is clear from these graphs that Verlet has much higher frame rates for a 50^2 mesh, with differences of roughly 3500-4000FPS. This is not expected, the expected behaviour is that Verlet has roughly the same performance as explicit Euler. The graphs show that other mesh sizes display the expected behaviour, so the results from the 50^2 mesh should be excluded as an anomaly.

Since the damping equation 2.8 is useless for Verlet, it may be possible to increase the performance by adding an if check to the calcSpringForce function in the Spring class that would not calculate 2.8 when using Verlet integration.

5.2.3 Midpoint

As with the previous integrators, the data for Midpoint displays similar trends.

Figs C.53, C.54, C.66 and C.67 all show that both the sheet and flag scenarios have a similar negative correlation to that displayed by the previous integrators, with coefficients of -0.7076 and -0.6941. Yet again there is a large initial drop in frame rate as the mesh size is changed to 100^2, but the decreases become smaller from that point. This is because the drop for the 100^2 mesh is much larger than explicit Euler or Verlet; approximately 10.5 and 13.8 times for the two scenarios. This explains why the Pearson's correlation coefficients for FPS as mesh size increases are smaller than the other integrators. With Midpoint, after the large initial frame rate drop, the changes become much smaller, which results, overall, in a less steep correlation.

Figs C.53 and C.66 continue to show the same positive correlation between FPS and time step for both scenarios, a conclusion also supported by Figs C.55 and C.68, which have correlation coefficients of 0.9651 and 0.9699 respectively. The same thresholding phenomenon is also observed, but it is more exaggerated for the Midpoint integrator. For the sheet scenario, the 300^2 mesh has an increased threshold of 10ms and for the flag scenario it has a threshold of 15ms.

Figs C.58 and C.71 explain why the threshold values have increased. The graphs show that the average time spent in the update function has increased over the previous integrators by almost 2 times. This is expected, as the Midpoint integrator involves two derivatives, so the forces acting on the cloth must be calculated twice. Consequently, it is expected that the average time spent calculating the forces should increase as well, a theory supported by Figs C.59 and C.72, both of which show that force calculation times have almost doubled. As a result of this increased update time, the frame rates for Midpoint are often significantly lower than both explicit Euler and Verlet.

The stability of the integrator is shown in Tables C.10 and C.12. It shows that Midpoint is only slightly more stable than explicit Euler but much less stable than Verlet; it is stable up to 10ms for the 50^2 mesh, but unstable with time steps greater than 1ms for every other mesh size.

The boxplots for Midpoint, shown in Figs C.60 to C.65 and Figs C.73 to C.78, present similar results as the other integrators. There is still a reasonably large variance in data, but the trends described are consistent in the raw data, listed in Tables C.9 and C.11, for individual tests

The data shows that, as expected, Midpoint is roughly twice as expensive as explicit Euler and Verlet and slightly more stable than explicit Euler. As a result, for small mesh sizes (50^2 or below), the

Midpoint integrator is a better choice than explicit Euler as it has increased performance due to the larger stable time step. However, as with Verlet, the data for the 50^2 mesh is highly anomalous, and so should not be used to draw conclusions. For anything other than small meshes, Midpoint is not recommended, as it is only stable with a 1ms time step which is too small to counteract the increased cost of the integrator.

5.2.4 Fourth Order Runge-Kutta

RK4 extends the trends displayed by Midpoint; notably lower frame rates, increased update times and increased time step thresholds.

Figs C.79 and C.92 show the lower frame rates of RK4. As with all the other integrators, there is still a clear negative correlation between average FPS and mesh size, a fact shown more clearly by Figs C.80 and C.93, with correlation coefficients of -0.6999 and -0.74 respectively. Large initial decreases in frame rate are observed, again with a 100^2 mesh giving the most dramatic decreases; roughly 12.7 and 6.3 times for the two scenarios. As with Midpoint, the size of this initial frame rate decrease explains why the correlation coefficients are slightly lower.

Figs C.79 and C.92 and C.81 and C.94 continue to show a positive correlation for frame rate as time step increases; the latter graphs have Pearson's coefficients of 0.8802 and 0.8889 respectively. The latter graphs also highlight the significantly lower frame rates for RK4; both scenarios show FPS numbers less than 60, with the flag scenario dropping below 30FPS for a 1ms time step.

RK4 requires four derivates, therefore it is expected that the total update time should be roughly twice as large as the Midpoint integrator. Figs C.84 and C.97 show that total update time is indeed roughly double that of Midpoint. These large update times explains why RK4 has many more time step thresholds than the other integrators. For both scenarios, the 300^2 mesh no longer has any time step that increases frame rate. The increase in time step thresholds explains why Fig C.79 shows an FPS drop for a 150^2 mesh with a 5ms time step. That mesh size has a time step threshold of 10ms, so it is expected that frame rate shouldn't increase for a 5ms time step. The slight performance decrease probably arises from expected variations in the data, but as the difference is not huge, this drop is not concerning.

The performance delta between the sheet and flag scenarios is much more pronounced for RK4. This is because there are larger differences between the update times for the scenarios using RK4 than other integrators. Figs C.84 and C.97 show that for a 300^2 mesh with a 1ms time step the update time difference between the scenarios is almost 10ms. This is much larger than Midpoint, where the difference between scenarios is only approximately 4ms. Figs C.82 and C.95 highlight the update time differences as well,

in particular highlighting that as mesh size increases, the time differences get wider.

Looking at the boxplots, Figs C.86 to C.91 show that for the sheet scenario, RK4 is consistent with the other integrators, having quite a lot of variance in the data. For the flag scenario, Figs C.99 to C.104 show less variance than the other integrators. This may be because the update times are so high that there is less possibility of variation. As with all the other integrators, examining the raw data in Tables C.13 and C.15 shows that the trends identified are consistent across multiple test runs, despite the variation.

Tables C.14 and C.16 list the stability of RK4 in both scenarios. They show that, as expected, RK4 is more stable than explicit Euler and Midpoint; stable up to 15ms for 50^2 meshes, and 5ms for mesh sizes up to 150^2. At 50^2 this stability is enough to counter the very high cost of RK4. However examinations of other integrators has chosen the 50^2 data to be anomalous so it cannot be used to recommend RK4. Sadly at all other mesh size, these time steps are not large enough to counter the very expensive computation of this integrator and therefore there are no situations in which RK4 should be chosen ahead of explicit Euler or Midpoint.

5.3   Evaluation

5.3.1   Null Hypothesis

The null hypothesis for this project is that all integration methods result in real time cloth simulation when running on modern hardware.

Whilst the data analysed above does indeed show that all the integrators result in real time simulations, this is only true for the small number of mesh sizes used in the testing process. The data, summarised in Figs C.105 and C.111, clearly show a negative correlation for frame rate as mesh size increases for all integrators. Hence, if mesh size were increased beyond the maximum used here, performance would decay even further, eventually resulting in non real time simulations. This frame rate decrease comes as a result of the cost of calculating each update step increasing with mesh size. Therefore, the only way to prevent the FPS loss is to use a larger time step to calculate less updates. However, the stability analysis shows that none of the integrators are stable with a time step greater than 1ms for larger mesh sizes, so there is no way of avoiding the performance loss. This disproves the null hypothesis and shows that performance is still very much a concern for Mass-Spring models, even when running on modern

hardware.

5.3.2 Alternative Hypothesis

The alternative hypothesis is that some integration methods are prohibitively expensive for real time simulations and that other methods give better performance.

This hypothesis is easily proven as the data clearly shows large performance deltas between some of the integrators (Figs C.105 and C.111 show this particularly well). There is also a clear negative correlation between FPS and mesh size, so if mesh size were increased beyond 300^2, some integrators would quickly not give real time results.

The results however are different from what was expected. At the beginning of this project it was expected that the explicit Euler and Verlet integrators would be the least performant, as they were dependant on small time steps only. Midpoint and RK4 were expected to be more performant, as they would be stable for larger time steps, which would counteract their increased computational cost. This turned out not to be the case. For larger mesh sizes, none of the integrators are stable for time steps larger than 1ms so the cheaper integrators end up giving the best performance results. Figs C.105 and C.106 and C.111 and C.112 suggest that for a 50^2 mesh the expensive integrators would give performance gains over explicit Euler, thanks to their increased stability. However, the data for this mesh size is considered highly anomalous. This is due to the extremely large performance deltas between Verlet and explicit Euler, which at all other mesh sizes are shown to have roughly identical performance. Therefore, the 50^2 mesh cannot be used to draw conclusions about integrator performance.

Chapter 6

Conclusion

The aim of this project was to investigate the performance effects of different integration methods on real time cloth simulation using the Mass-Spring model. With that in mind, two hypotheses were proposed, a null and an alternative. Two were proposed as it was unknown at the beginning whether there would still be performance concerns for Mass-Spring models due to the fact that the core literature is many years old.

The data extracted from the Mass-Spring model clearly disproves the null hypothesis, and thus proves the alternative.

At small mesh sizes, the null hypothesis is indeed true, all integrators lead to real time simulations. As mesh size is increased however, large performance deltas appear between the integrators, and at the largest mesh size used, the RK4 integrator drops below 30FPS in one scenario. Were mesh size to be increased beyond the maximum used in this project, all the integrators would eventually drop below the 30FPS threshold, thus, it is clear that even with modern hardware all integrators do not necessarily give real time results.

The fact that performance deltas appear between the integrators proves the alternative hypothesis, though the results were different from what was expected. During the initial stages of this project it was expected that explicit Euler and Verlet would be the more expensive integrators, as they would be limited to small time steps and therefore frequent updates. However, analysing the data shows that in fact Midpoint and RK4 are the more expensive integrators, with RK4 being borderline real time. This reversal is explained by looking at the stability of the integrators. It was expected that Midpoint and RK4 would be stable at higher time steps than explicit Euler; this is indeed the case. Verlet actually ends up more

stable than RK4 and Midpoint. However, neither of them are sufficiently more stable to counter their much higher update costs. This is particularly problematic at larger mesh sizes, where all integrators are only stable with a 1ms time step. Therefore, the cheaper integrators dominate here, resulting in frame rate increases of roughly 2 times and 4 times for Midpoint and RK4 respectively.

To conclude, this project has shown that performance for Mass-Spring models is still a concern when aiming for real time simulations, especially when running on a single thread. Integrator choice definitely makes a difference to overall performance, with different results than were expected. RK4 and Midpoint are not sufficiently more stable to counteract their computational cost, and therefore they are not recommended for use in real time Mass-Spring models, especially with large mesh sizes. Explicit Euler and Verlet, by contrast, are recommended as they are cheap, and result in frame rates well above 30FPS at mesh sizes up to 300^2. In particular, Verlet is recommended as the most efficient integrator, as it is more stable then explicit Euler up to 200^2 meshes, and therefore results in much larger frame rates for those mesh sizes.

6.1 Future Work

There is scope for a lot of additional research based off this project.

Firstly, the anomalous data for the 50^2 mesh should be investigated further, to try and determine why at this mesh size only explicit Euler has drastically worse performance than Verlet.

One simple area that could be researched further would be to investigate the mesh size at which all the integrators drop below 30FPS; this would give the upper bound of what is possible for single threaded Mass-Spring models.

A key area that should be researched is the impact of multi-threading and GPU acceleration. Mass-Spring models are highly parallelisable, so this project should be replicated for both multi-threaded CPU and GPU accelerated implementations. It may be the case that the improved performance of the parallel implementations increases the stability of the integrators and could lead to different conclusions.

One caveat of this project is that it has not focused on the appearance of the cloth. For a game this

would be inappropriate. It is possible that by focusing on the appearance of the cloth as well as performance, different results may be found. For example, varying the spring stiffness may affect the stability of the cloth, so the more expensive integrators may become viable. Therefore, it would be interesting to research what effect focusing on appearance more would have on performance and integrator choice. This could possibly involve collaboration with an artist and development of a UI tool to control the cloth parameters dynamically.

Similarly, it may also be interesting to study the performance of integrators in a more game like environment. This project has focused on simulating the cloth on its own, but in a video game there would be many other demands on the available frame time. A game environment would also provide many more scenarios in which performance could be investigated, such as attaching the cloth to a game character as a cape and animating it with their movement.

One obvious extension is to also compare the performance of implicit integrators. Implicit integrators are unconditionally stable and are generally considered more appropriate for the stiff equations of Mass-Spring systems. However, they involve large matrix calculations which may render them inappropriate for real time use.

Finally, another necessary extension would be to compare this project against a PBD implementation. PDB are the standard for cloth in video games, so comparing them with traditional Mass-Spring models would show why PDB are preferred.

Appendix A

Class Diagrams

A.1 First Cycle Design
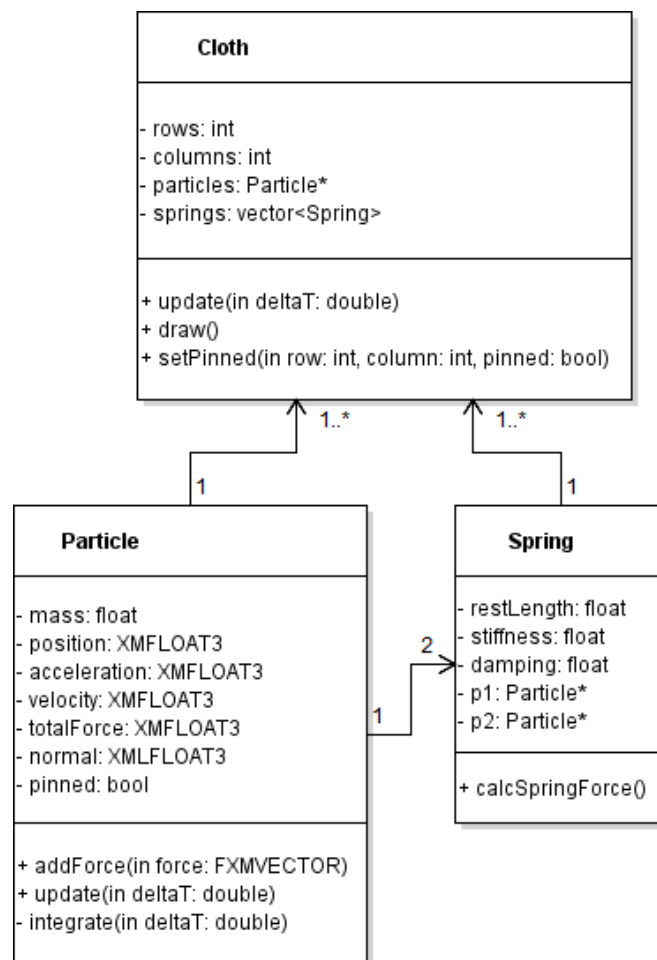
Figure A.1: Initial design for the first development cycle
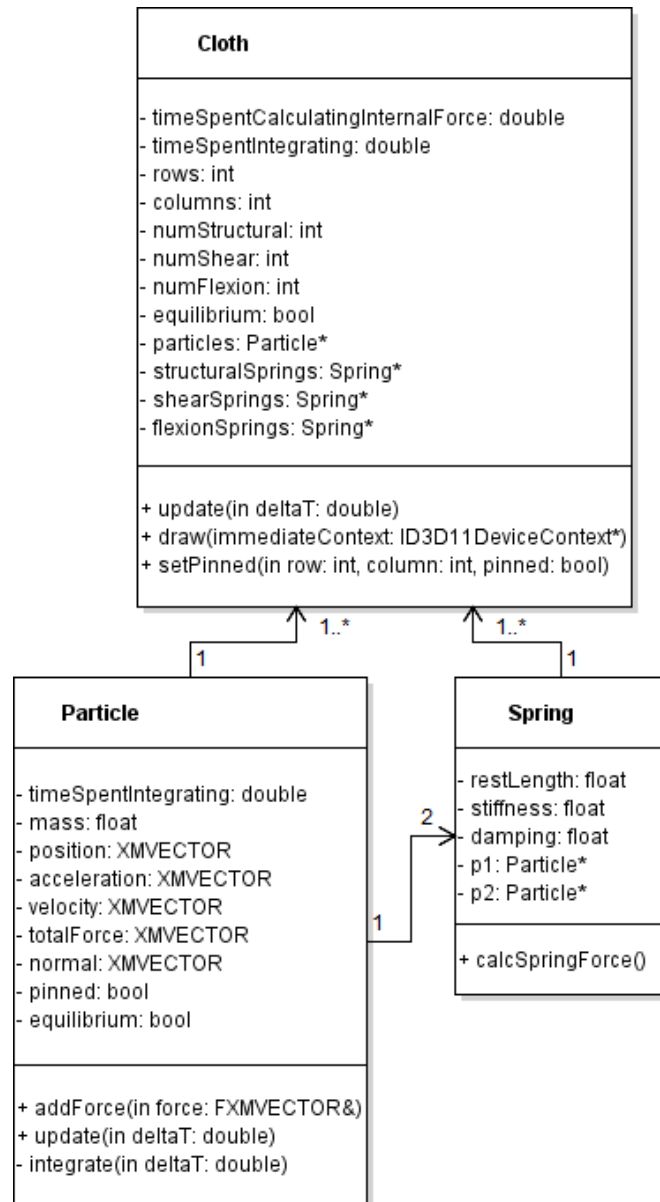
Figure A.2: Final design for the first development cycle

A.2   Second Cycle Design
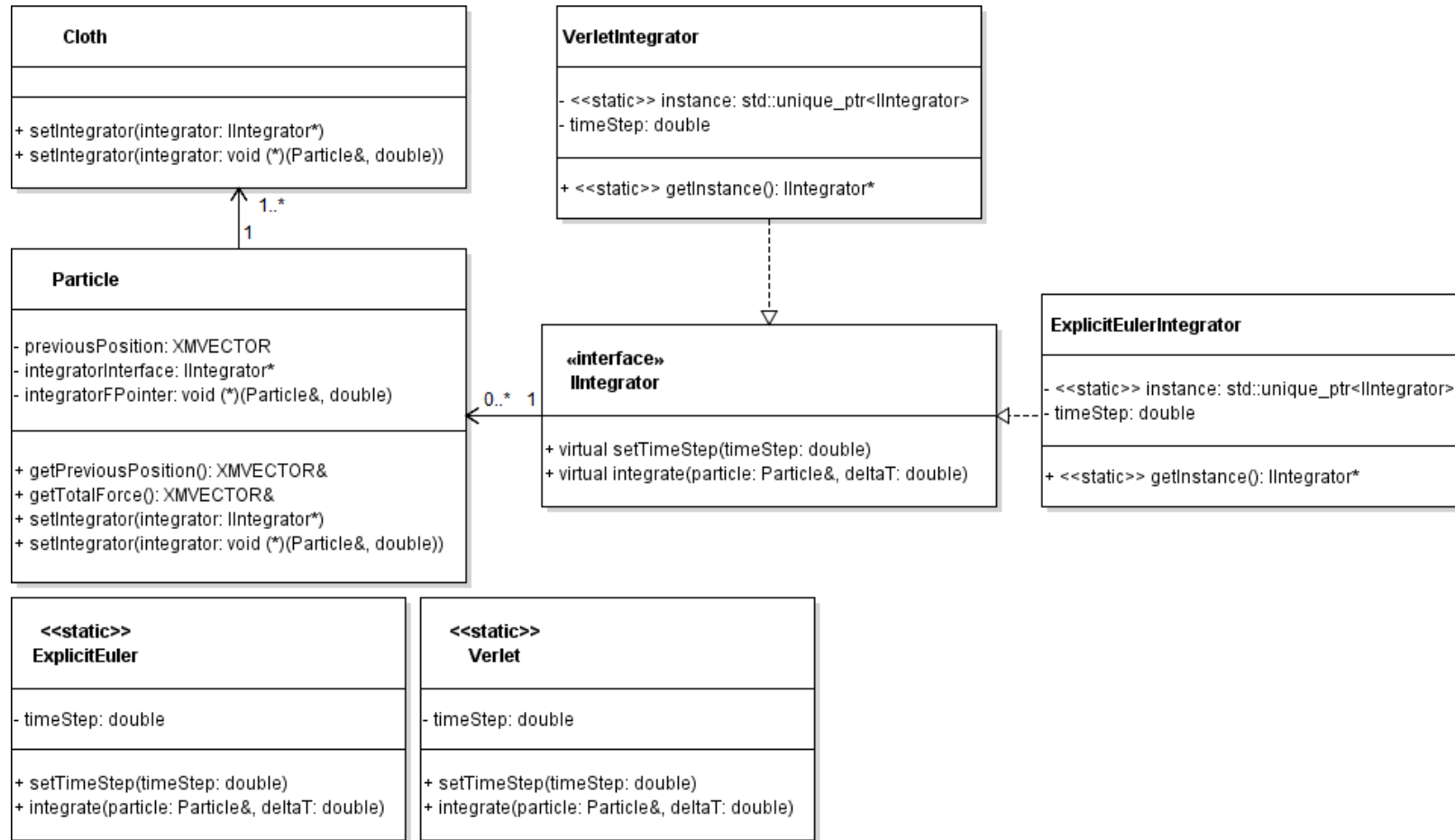
Figure A.3: Initial design for the second development cycle
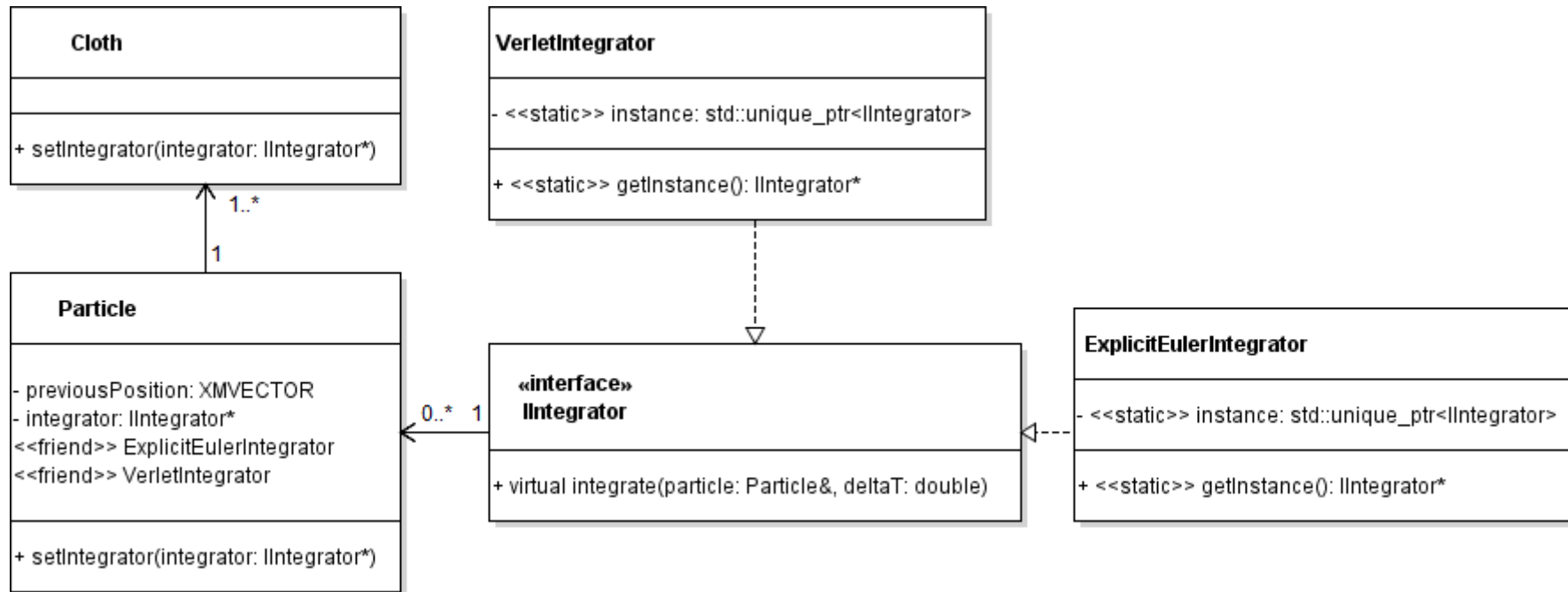
Figure A.4: Final design for the second development cycle

A.3 Third Cycle Design
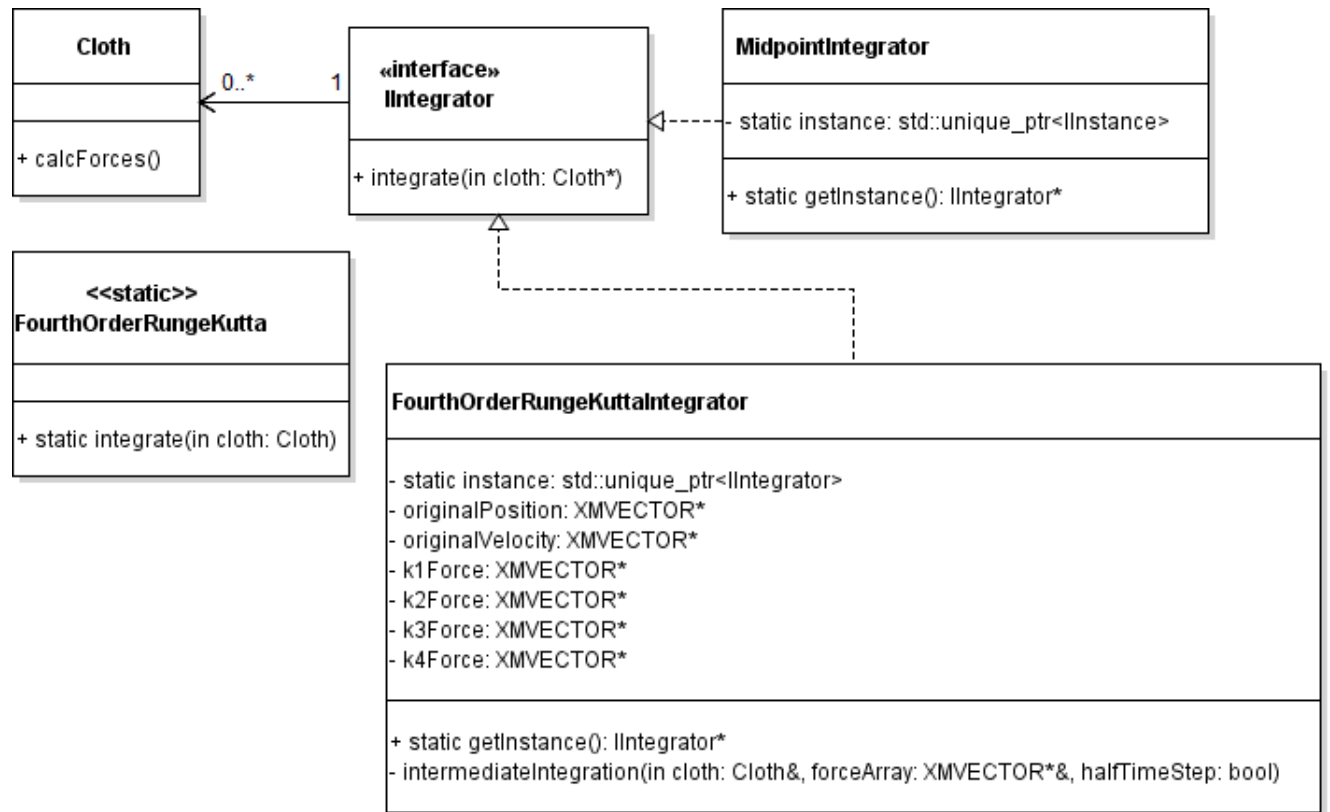
Figure A.5: Initial design for the third development cycle
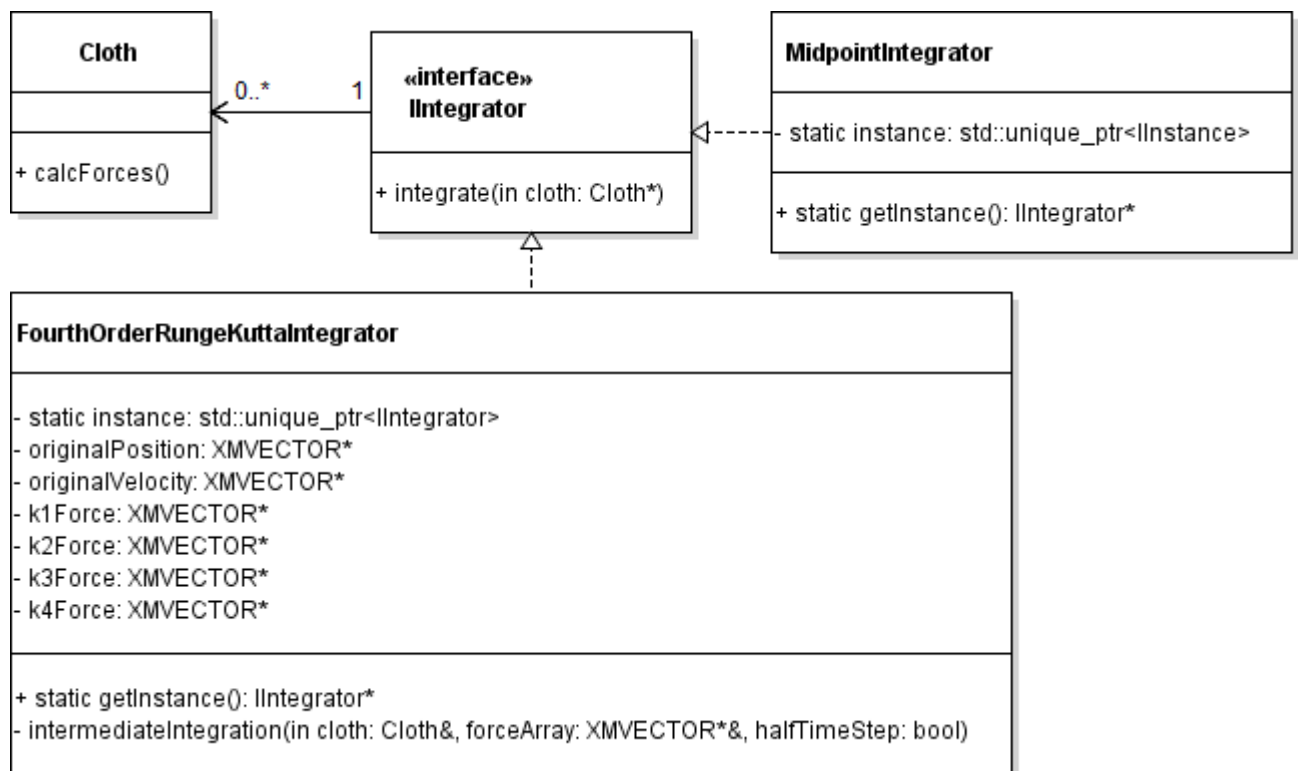
Figure A.6: Final design for the third development cycle

Appendix B

Profiling Results

B.1 First Cycle

Hot Path

Function Name	Inclusive Samples %	Exclusive Samples %
Cloth::update	93.00	1.11
std::_Vector_const_iterator<std::_Vector_val<std::_Simple_types<Spring> > >::operator!=	25.13	3.47
std::_Vector_iterator<std::_Vector_val<std::_Simple_types<Spring> > >::operator*	22.37	3.92
std::_Vector_iterator<std::_Vector_val<std::_Simple_types<Spring> > >::operator++	17.06	3.65
Spring::calcSpringForce	9.74	4.57

Figure B.1: Profiling results using a std::vector to store springs

Hot Path

Function Name	Inclusive Samples %	Exclusive Samples %
wWinMain	99.12	0.00
Application::update	96.81	1.91
Cloth::update	77.81	2.23
Spring::calcSpringForce	41.99	20.32
Particle::addForce	17.71	3.88

Figure B.2: Profiling results using dynamic arrays to store springs

Hot Path

Function Name	Inclusive Samples %	Exclusive Samples %
Spring::calcSpringForce	89.51	5.09
Particle::addForce	23.05	5.12
DirectX::XMLoadFloat3	12.17	12.17
DirectX::operator-	9.13	4.39
DirectX::operator*	8.63	4.26

Related Views: Call Tree   Functions

Functions Doing Most Individual Work

Name	Exclusive Samples %
DirectX::XMLoadFloat3	18.00
DirectX::XMStoreFloat3	8.45
Particle::addForce	5.86
DirectX::XMVectorAdd	5.75
Spring::calcSpringForce	5.19

Figure B.3: Profiling results for unoptimised calcSpringForce

Hot Path

Function Name	Inclusive Samples %	Exclusive Samples %
🔥 DirectX::XMVectorScale	8.88	8.88
🔥 DirectX::XMVectorSubtract	7.28	7.28
🔥 DirectX::XMVector3Length	6.53	6.53
🔥 DirectX::XMVector3Dot	4.80	4.80
🔥 Particle::getPosition	4.77	4.77

Related Views: Call Tree   Functions

Functions Doing Most Individual Work

Name	Exclusive Samples %
DirectX::XMVectorScale	11.54
DirectX::XMVectorAdd	9.86
Spring::calcSpringForce	9.74
DirectX::XMVectorSubtract	7.64
Particle::addForce	6.87

Figure B.4: Profiling results for optimised calcSpringForce

Appendix C

Test Results

C.1   Explicit Euler

C.1.1   Sheet Data

Rows	Columns	Time Step	Internal force (ms)	External force (ms)	Integrate (ms)	Update (ms)	Render (ms)	FPS	Update count
50	50	1	0.128615	0.005164	0.00868608	0.156632	0.100868	8268.3	57212
50	50	1	0.142372	0.00465691	0.00841708	0.167097	0.129328	6404.28	55975
50	50	1	0.142736	0.00457576	0.008418	0.166826	0.127135	6503.11	56703
50	50	1	0.128142	0.00511719	0.00864091	0.153111	0.119058	7070.92	55706
50	50	1	0.136302	0.00472693	0.00904226	0.164193	0.100677	8225.89	56909
50	50	1	0.129415	0.0045023	0.00849498	0.156785	0.0999752	8334.95	57604
50	50	5	0.125968	0.00515918	0.00879495	0.167273	0.100393	9590.8	11904

50	50	5	0.139694	0.00466677	0.00844583	0.178078	0.129721	7414.72	11859
50	50	5	0.140375	0.00459534	0.00846154	0.17746	0.12796	7519.88	11879
50	50	5	0.126731	0.0050855	0.00869745	0.166506	0.11904	8099.67	11878
50	50	5	0.127391	0.00454317	0.0085639	0.168509	0.10092	9542.57	11900
50	50	5	0.128197	0.00455854	0.00847804	0.170248	0.100271	9604.15	11898
50	50	10	0.119753	0.00517193	0.00880473	0.177195	0.100565	9733.7	5984
50	50	10	0.133649	0.00461083	0.00842911	0.185812	0.129613	7551.1	5970
50	50	10	0.13425	0.00459733	0.00846291	0.184474	0.128381	7627.8	5969
50	50	10	0.120623	0.00509719	0.00871628	0.175288	0.118932	8235.82	5974
50	50	10	0.120971	0.0045458	0.00858291	0.178856	0.101092	9688.4	5981
50	50	10	0.121658	0.0045216	0.00843943	0.181173	0.100487	9742.47	5980
50	50	15	0.118788	0.00513531	0.00877042	0.192235	0.100365	9800.3	3996
50	50	15	0.133633	0.00458522	0.00842181	0.199169	0.129537	7597.92	3989
50	50	15	0.13293	0.00456431	0.00841453	0.195633	0.127976	7694.93	3989
50	50	15	0.119784	0.00509553	0.00870429	0.18909	0.118943	8280.7	3987
50	50	15	0.120027	0.00454277	0.00857146	0.194847	0.100914	9749.95	3994
50	50	15	0.120678	0.00452713	0.00843417	0.197571	0.100332	9809.6	3993
50	50	20	0.114838	0.00517791	0.00886759	0.202998	0.10278	9598.15	2999
50	50	20	0.130242	0.00460061	0.00839896	0.205485	0.139744	7070.62	2993
50	50	20	0.129851	0.00458828	0.00842905	0.202839	0.135316	7303.17	2993
50	50	20	0.115379	0.00509642	0.00871972	0.198906	0.120033	8227.22	2995
50	50	20	0.115597	0.00454956	0.00857986	0.205574	0.103456	9540.51	2997
50	50	20	0.115717	0.0045517	0.0084265	0.209707	0.100757	9791.35	2997
100	100	1	0.50021	0.0250417	0.0367431	0.596335	0.0765951	5573.75	57748
100	100	1	0.514832	0.0251723	0.0370054	0.612511	0.0914453	4545.07	57070
100	100	1	0.515666	0.0250743	0.0368359	0.614561	0.0815862	5044.59	58056

100	100	1	0.502288	0.0249129	0.0368309	0.597887	0.113292	3776.3	57351
100	100	1	0.49928	0.0249818	0.0367029	0.594886	0.0675679	6320.38	57836
100	100	1	0.499808	0.0251597	0.0368722	0.596911	0.0690599	6179.37	57543
100	100	5	0.474015	0.0250705	0.0369404	0.589845	0.0687689	12681.6	11942
100	100	5	0.485705	0.0251561	0.0373023	0.599072	0.0868691	10080.3	11923
100	100	5	0.487466	0.0251163	0.0369153	0.600225	0.0832743	10522.8	11909
100	100	5	0.474668	0.0249071	0.0367763	0.584668	0.0940144	9361.45	11918
100	100	5	0.4729	0.0249976	0.0368693	0.586268	0.0672476	13026.7	11951
100	100	5	0.473421	0.0251752	0.0369446	0.588842	0.0657204	13343.2	11947
100	100	10	0.462763	0.0250988	0.0370636	0.600559	0.0702156	13306.2	5991
100	100	10	0.474766	0.0251945	0.037352	0.607567	0.0874179	10693.9	5980
100	100	10	0.476227	0.025096	0.0371006	0.614672	0.0789858	11821.5	5986
100	100	10	0.463763	0.0249026	0.0368456	0.595645	0.0846013	11066.5	5980
100	100	10	0.461878	0.0250276	0.0369636	0.595455	0.0681665	13720.8	5988
100	100	10	0.462265	0.0251808	0.0369779	0.599364	0.0675028	13858.6	5987
100	100	15	0.46245	0.0251022	0.0371451	0.622958	0.069685	13683.2	3999
100	100	15	0.474074	0.0251689	0.0373619	0.628673	0.0858	11113.3	3993
100	100	15	0.476084	0.0250906	0.0370946	0.634451	0.0801038	11896.1	3995
100	100	15	0.464404	0.0250113	0.0369353	0.616501	0.0833235	11463.5	3993
100	100	15	0.461497	0.0250315	0.0370508	0.615941	0.0680001	14031.6	3997
100	100	15	0.461899	0.0252166	0.03701	0.621804	0.0667296	14296.5	3996
100	100	20	0.465629	0.0251648	0.0372114	0.647722	0.0698311	13783.5	3002
100	100	20	0.474761	0.0251906	0.0374231	0.644402	0.0882953	10917.2	2998
100	100	20	0.476339	0.0250902	0.0370668	0.655153	0.0793703	12132.5	2999
100	100	20	0.464899	0.0249948	0.0369497	0.63632	0.0828481	11633	2999
100	100	20	0.461292	0.0250317	0.0370342	0.636176	0.0680484	14163.4	3000

100	100	20	0.462373	0.0252264	0.0370396	0.642698	0.0670208	14379.3	3000
150	150	1	1.08714	0.0566925	0.0825923	1.29726	0.0844184	736.65	43421
150	150	1	1.10248	0.0571319	0.0841166	1.32164	0.0878185	766.1	42566
150	150	1	1.10967	0.0569523	0.0831631	1.32709	0.0806708	787.85	42618
150	150	1	1.08771	0.0565917	0.0829925	1.29882	0.17784	702.2	40629
150	150	1	1.08063	0.0567398	0.0827397	1.29111	0.0820817	838	43690
150	150	1	1.08111	0.0571194	0.0832445	1.29227	0.0828494	797.983	43628
150	150	5	1.07668	0.0568541	0.0833727	1.29852	0.13282	5592.73	11776
150	150	5	1.13386	0.0615759	0.0883113	1.37683	0.140797	5146.57	11777
150	150	5	1.13924	0.0608693	0.0877071	1.37302	0.133314	5433.48	11792
150	150	5	1.07793	0.056962	0.0834797	1.29983	0.144025	5152.43	11766
150	150	5	1.07074	0.0569252	0.0829838	1.29125	0.136205	5437.42	11772
150	150	5	1.07076	0.0571872	0.0832443	1.29207	0.136572	5436.78	11764
150	150	10	1.06692	0.0571516	0.0836223	1.32154	0.0696779	12380.9	5984
150	150	10	1.07403	0.057229	0.0841817	1.32903	0.0888019	9705.15	5976
150	150	10	1.07785	0.0569725	0.0832342	1.32563	0.0840754	10257.7	5975
150	150	10	1.06373	0.0567704	0.0833085	1.30899	0.095309	9085.35	5972
150	150	10	1.05778	0.056876	0.0829886	1.30929	0.0678799	12708.1	5982
150	150	10	1.05875	0.0571627	0.083242	1.31299	0.067177	12852.2	5982
150	150	15	1.06815	0.057426	0.0836969	1.34509	0.069657	12989.2	3997
150	150	15	1.07505	0.0571288	0.0840977	1.34912	0.0887723	10197.7	3991
150	150	15	1.07909	0.057166	0.0832839	1.34541	0.0843817	10736.6	3991
150	150	15	1.06352	0.0571562	0.0833967	1.3274	0.0938246	9675.4	3990
150	150	15	1.06198	0.0572774	0.0834166	1.3351	0.0681589	13286.7	3995
150	150	15	1.06052	0.0571593	0.0832544	1.33689	0.0667814	13563.5	3994
150	150	20	1.07018	0.0577767	0.0839181	1.37055	0.0695069	13332.2	3001

150	150	20	1.07717	0.0578269	0.0839255	1.3717	0.0885381	10477.4	2997
150	150	20	1.07945	0.0571732	0.0833288	1.3644	0.0842716	11015.4	2997
150	150	20	1.06432	0.0573332	0.0834348	1.34475	0.0943868	9842.9	2995
150	150	20	1.06143	0.0568002	0.0831633	1.35405	0.067912	13661.4	2999
150	150	20	1.06213	0.0572555	0.0832937	1.36106	0.0667378	13901.6	2999
200	200	1	1.95413	0.104778	0.148641	2.33973	0.117445	487.883	24417
200	200	1	1.99599	0.107285	0.150545	2.40077	0.149581	455.667	23525
200	200	1	1.97059	0.104756	0.149374	2.3539	0.110811	479	24342
200	200	1	1.95061	0.10323	0.148556	2.32978	0.195538	403.7	23758
200	200	1	1.94577	0.10454	0.148732	2.3295	0.118045	535.483	24513
200	200	1	1.94179	0.10393	0.14907	2.32435	0.113833	485.983	24607
200	200	5	1.92935	0.10586	0.149001	2.33234	0.0709002	7496.1	11903
200	200	5	1.93676	0.10649	0.148707	2.34801	0.134698	3958.08	11803
200	200	5	1.96417	0.106265	0.148976	2.36651	0.0851045	6181.52	11876
200	200	5	1.91734	0.10373	0.148308	2.31132	0.0920302	5870.47	11875
200	200	5	1.9483	0.113891	0.155346	2.37752	0.0740104	7046.98	11879
200	200	5	1.9155	0.105509	0.149162	2.31727	0.0683443	7837.52	11887
200	200	10	1.94133	0.109748	0.151116	2.38167	0.070435	10714	5982
200	200	10	1.94682	0.112579	0.151517	2.38638	0.142851	5307.6	5958
200	200	10	2.03684	0.107776	0.150326	2.46866	0.0836876	8924.45	5977
200	200	10	1.92149	0.105546	0.148856	2.34046	0.0920961	8284.85	5971
200	200	10	1.93715	0.108519	0.150363	2.37241	0.0704464	10720.6	5974
200	200	10	1.9282	0.104518	0.148835	2.35197	0.0707199	10735.3	5973
200	200	15	1.94404	0.111236	0.151985	2.40702	0.077099	10827.8	3989
200	200	15	1.96589	0.106043	0.148658	2.40119	0.144915	5770.4	3982
200	200	15	2.11432	0.110783	0.152093	2.57682	0.0840341	9793.25	3989

200	200	15	1.92492	0.106619	0.149712	2.36509	0.0918499	9131.93	3986
200	200	15	1.938	0.108929	0.15043	2.38794	0.077202	10852.4	3987
200	200	15	1.92885	0.104003	0.148873	2.36929	0.0778796	10767	3986
200	200	20	1.9447	0.111514	0.151464	2.42282	0.0804087	10882.5	2995
200	200	20	1.966	0.105716	0.148499	2.41275	0.146134	5993.66	2991
200	200	20	2.18893	0.111935	0.153117	2.6774	0.0839945	10264.8	2995
200	200	20	1.92923	0.108473	0.150372	2.39083	0.0919629	9531.05	2993
200	200	20	1.94757	0.111225	0.151756	2.41941	0.0809524	10817.4	2993
200	200	20	1.93377	0.105474	0.148988	2.39139	0.0813481	10786	2993
250	250	1	3.08418	0.182782	0.246822	3.75981	0.173586	356.8	15254
250	250	1	3.16687	0.197024	0.259308	3.88623	0.243471	255.8	14529
250	250	1	3.10065	0.185383	0.248462	3.77634	0.167898	336.3	15211
250	250	1	3.07798	0.182707	0.245508	3.74736	0.253781	384.5	14996
250	250	1	3.08629	0.18605	0.24817	3.76506	0.174322	355.533	15230
250	250	1	3.06683	0.186924	0.248079	3.74641	0.176597	304.267	15294
250	250	5	3.05751	0.187729	0.249851	3.75391	0.0814721	3165.08	11806
250	250	5	3.12068	0.191736	0.254868	3.83152	0.101219	2436.75	11774
250	250	5	3.11574	0.188725	0.250138	3.81115	0.090901	2719.32	11806
250	250	5	3.04504	0.185082	0.246625	3.72955	0.100829	2600.75	11797
250	250	5	3.04605	0.183258	0.24593	3.7258	0.0804637	3284.58	11793
250	250	5	3.0386	0.186857	0.247725	3.73098	0.0782723	3380.85	11792
250	250	10	3.07186	0.201004	0.261647	3.83742	0.0837422	7271.78	5959
250	250	10	3.42961	0.198484	0.261089	4.18847	0.0879563	6577.13	5964
250	250	10	3.407	0.227755	0.286992	4.22662	0.0813083	7028.73	5964
250	250	10	3.06312	0.193748	0.252989	3.79331	0.0898306	6909.45	5962
250	250	10	3.06451	0.197424	0.258466	3.81484	0.0840582	7281.03	5954

250	250	10	3.05432	0.197824	0.257591	3.80619	0.0840639	7332.03	5948
250	250	15	3.07715	0.20055	0.261486	3.85542	0.100545	7371.66	3974
250	250	15	3.46747	0.233804	0.293241	4.34844	0.0963448	7336.27	3980
250	250	15	3.41866	0.224689	0.293707	4.28708	0.0969226	7336.57	3975
250	250	15	3.08381	0.202316	0.262605	3.86596	0.100725	7355.25	3975
250	250	15	3.0725	0.205373	0.264258	3.85529	0.100369	7387.77	3976
250	250	15	3.06142	0.203966	0.262263	3.84198	0.100387	7398.25	3978
250	250	20	3.08085	0.2046	0.265839	3.88544	0.108817	7385.13	2990
250	250	20	3.45353	0.23517	0.305929	4.38303	0.105861	7352.98	2988
250	250	20	3.45274	0.232259	0.301259	4.36033	0.106078	7357.5	2987
250	250	20	3.08666	0.201162	0.261554	3.87949	0.109122	7370.48	2989
250	250	20	3.08025	0.204587	0.265067	3.87951	0.108708	7398.45	2988
250	250	20	3.06561	0.205532	0.261605	3.85704	0.108729	7408.38	2988
300	300	1	4.80223	0.353893	0.428961	6.06601	0.197921	218.15	9579
300	300	1	4.83461	0.359853	0.437164	6.12223	0.210374	165.333	9475
300	300	1	4.81431	0.356429	0.43079	6.0796	0.195408	198.583	9562
300	300	1	4.81352	0.351226	0.426952	6.06781	0.283994	238.1	9446
300	300	1	4.80263	0.35189	0.426574	6.05617	0.200668	199.15	9590
300	300	1	4.76137	0.35572	0.424695	6.01429	0.199108	162.35	9657
300	300	5	4.71729	0.358629	0.432467	5.99137	0.192832	162.3	9700
300	300	5	4.74013	0.357931	0.434654	6.01875	0.208394	161.25	9634
300	300	5	4.7211	0.356094	0.429853	5.98699	0.19363	162.4	9706
300	300	5	4.70179	0.350677	0.423889	5.94767	0.276287	160.533	9639
300	300	5	4.70556	0.347797	0.425081	5.95607	0.202197	163.1	9741
300	300	5	4.66876	0.348872	0.422295	5.91153	0.202642	164.533	9811
300	300	10	4.74901	0.375124	0.453033	6.13049	0.0743465	5188.7	5959

300	300	10	5.08145	0.380255	0.453132	6.43367	0.0878853	4082.97	5957
300	300	10	5.07201	0.36577	0.438182	6.38925	0.0820663	4409.65	5956
300	300	10	4.72499	0.355613	0.428347	6.00655	0.0934473	4256.7	5952
300	300	10	4.72778	0.370681	0.44958	6.10033	0.0741941	5254.88	5958
300	300	10	4.69854	0.37232	0.445166	6.05805	0.0753774	5288.7	5934
300	300	15	4.74297	0.379159	0.457516	6.14611	0.111172	5279.57	3976
300	300	15	5.13869	0.421754	0.50207	6.67648	0.104468	5300.12	3975
300	300	15	5.10131	0.420554	0.49175	6.62004	0.105271	5285.57	3975
300	300	15	4.74181	0.375028	0.449344	6.12214	0.111483	5267.39	3976
300	300	15	4.73129	0.375718	0.453125	6.11588	0.111609	5284.8	3977
300	300	15	4.70019	0.3811	0.45124	6.08509	0.112076	5296.62	3978
300	300	20	4.74978	0.380027	0.457602	6.16323	0.130308	5306.43	2985
300	300	20	5.11336	0.426038	0.499694	6.671	0.125386	5311.53	2983
300	300	20	5.10323	0.428465	0.496571	6.65422	0.125587	5315.43	2981
300	300	20	4.73769	0.383656	0.455559	6.14907	0.130268	5317.8	2981
300	300	20	4.73566	0.380115	0.454533	6.14216	0.13066	5300.53	2985
300	300	20	4.69457	0.380791	0.451193	6.09217	0.131159	5300.47	2985

Table C.1: Raw data for explicit Euler (sheet)
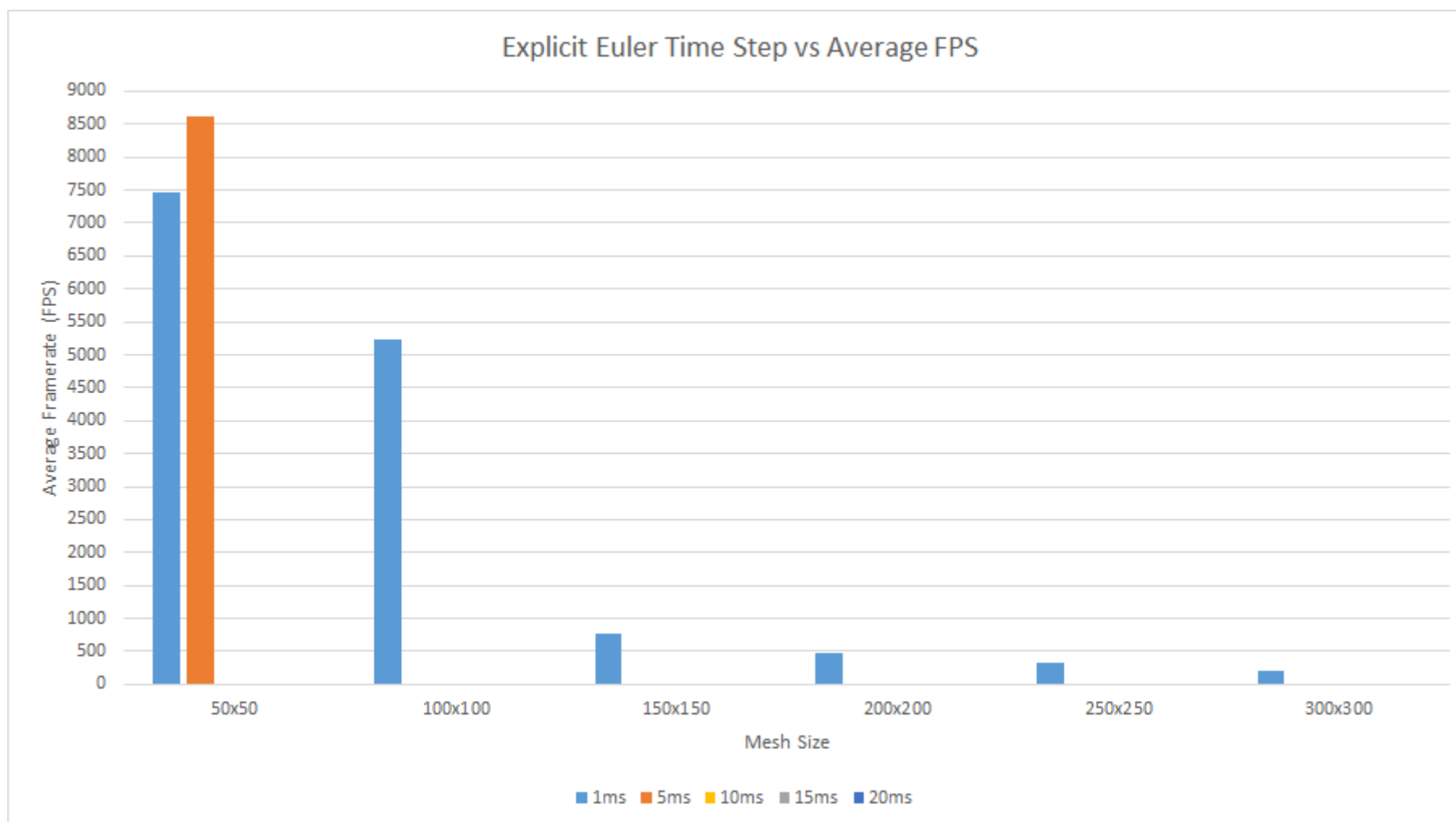
Figure C.1: Explicit Euler time step against average FPS (sheet)
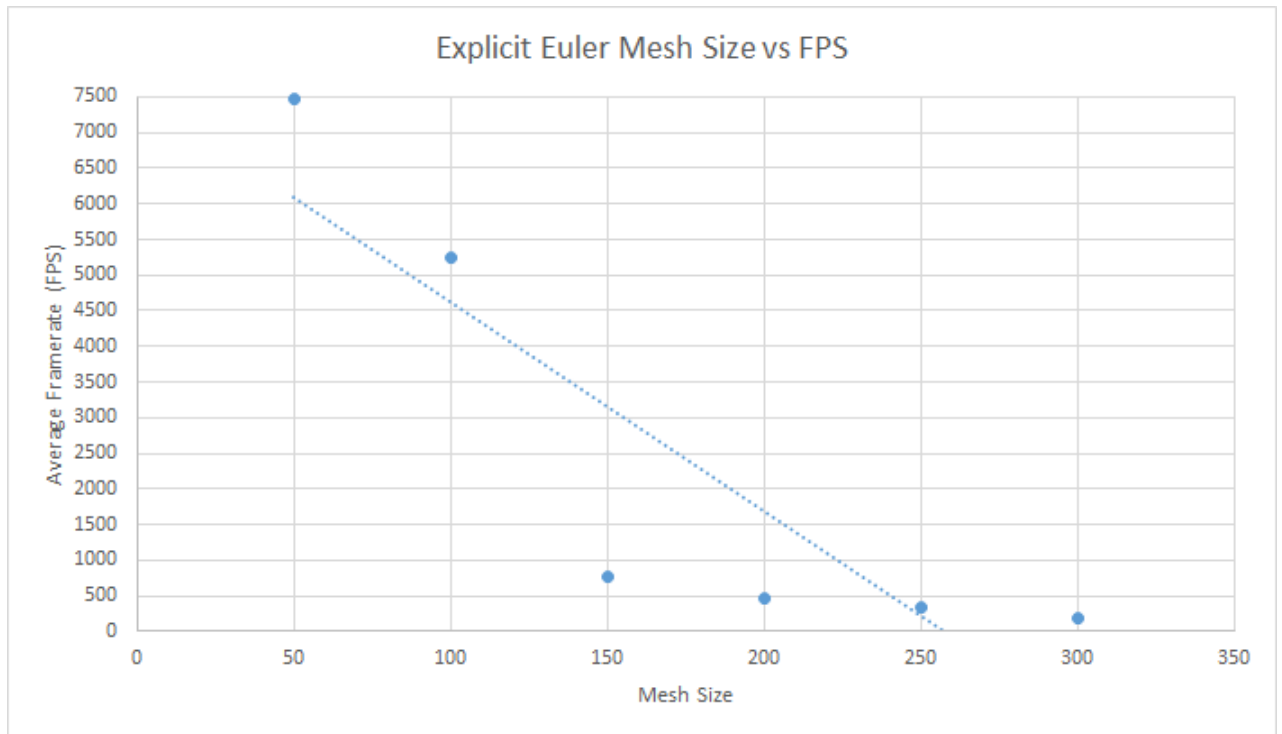
Figure C.2: Explicit Euler mesh size against average FPS (sheet)
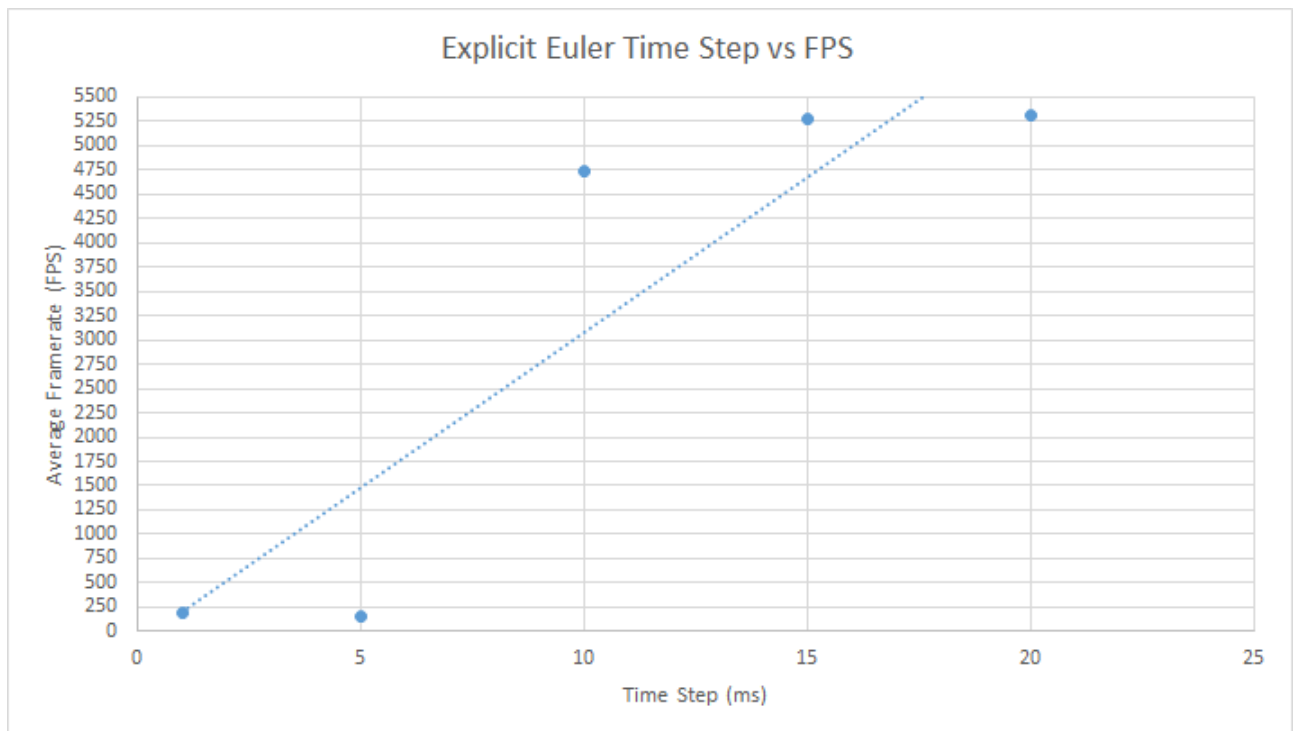
Figure C.3: Explicit Euler time step against average FPS for a 300^2 mesh (sheet)
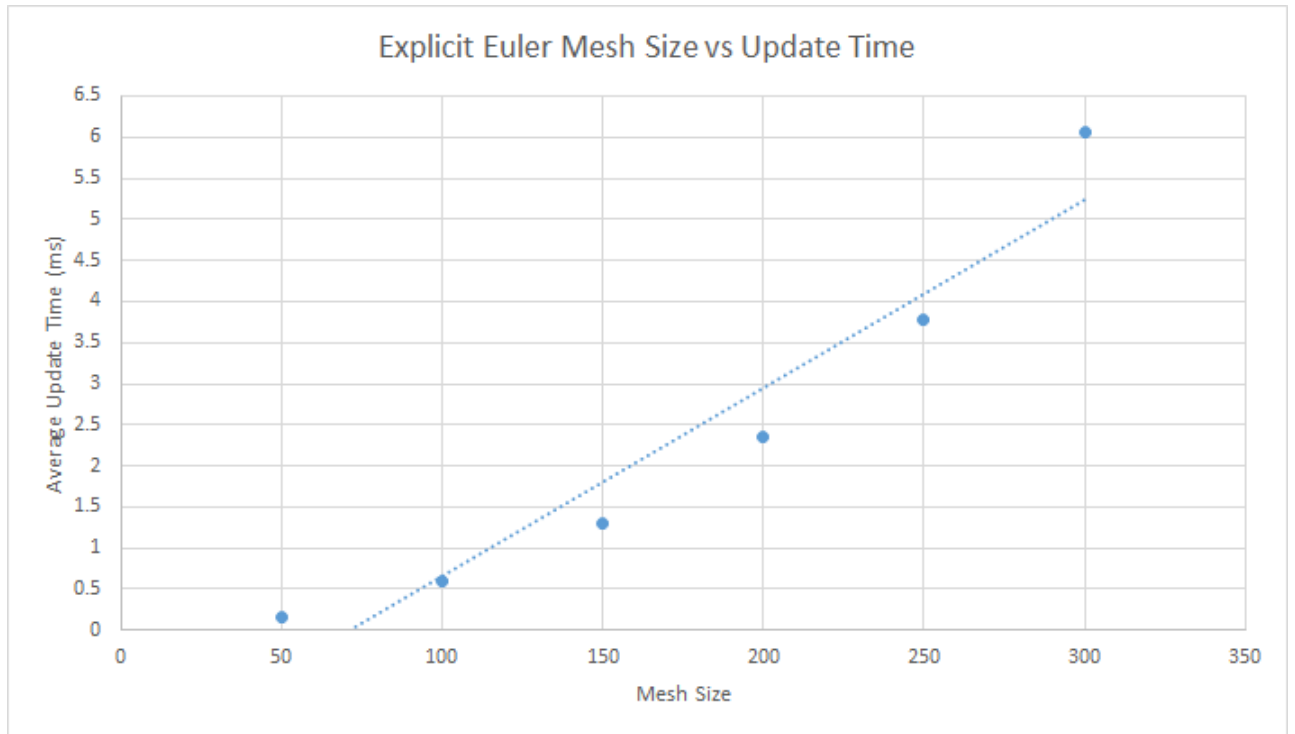
Figure C.4: Explicit Euler mesh size against average update time (sheet)
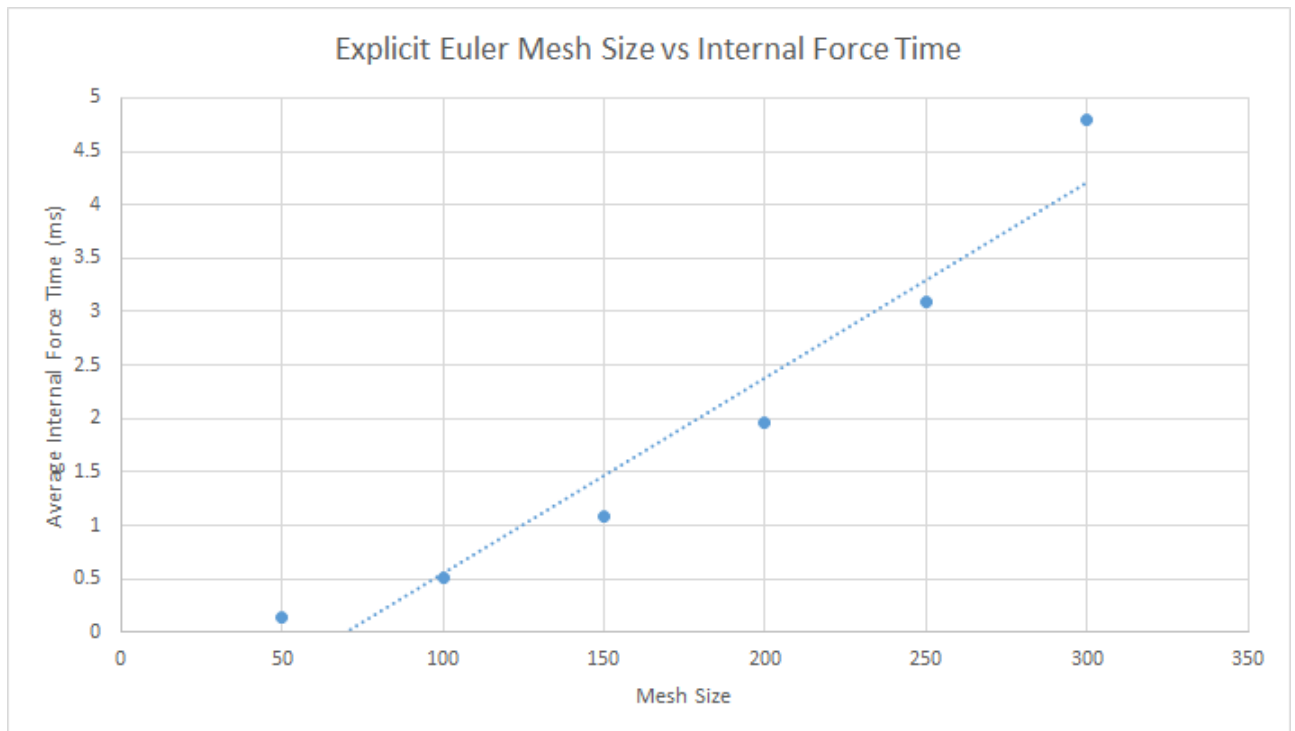
Figure C.5: Explicit Euler mesh size against average internal force time (sheet)
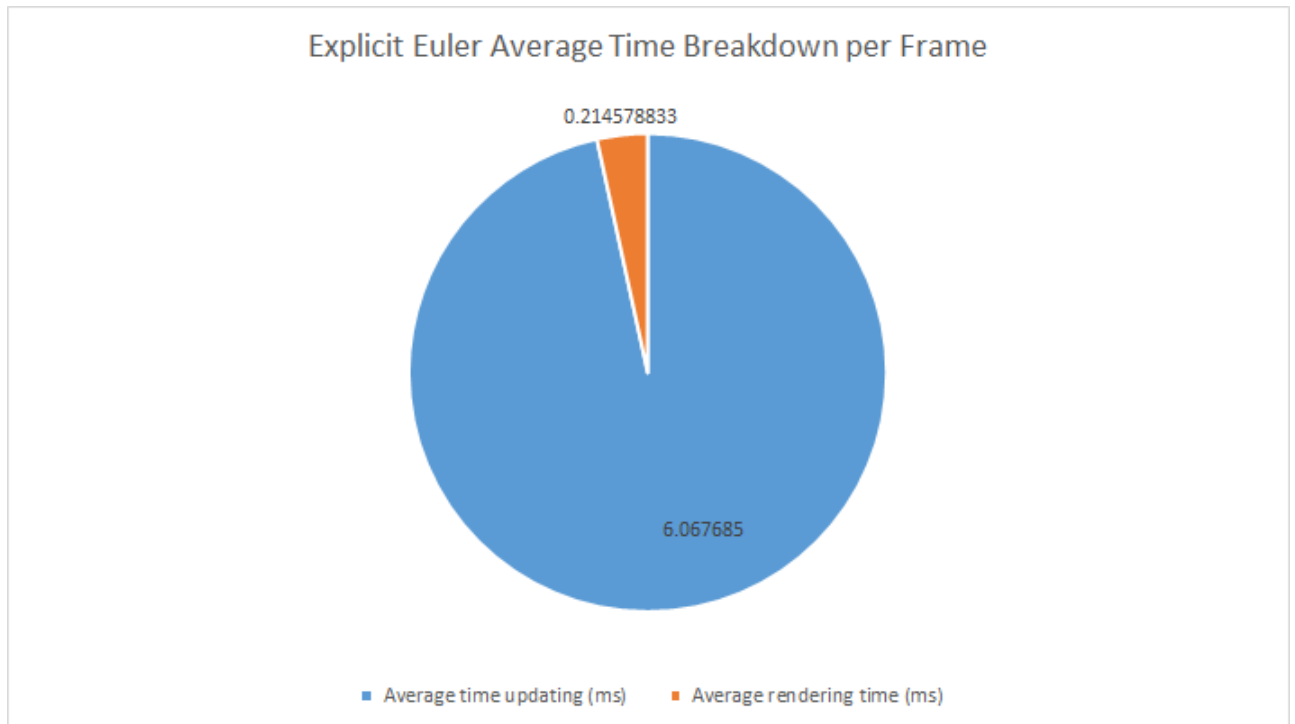
Figure C.6: Explicit Euler frame time breakdown (sheet)
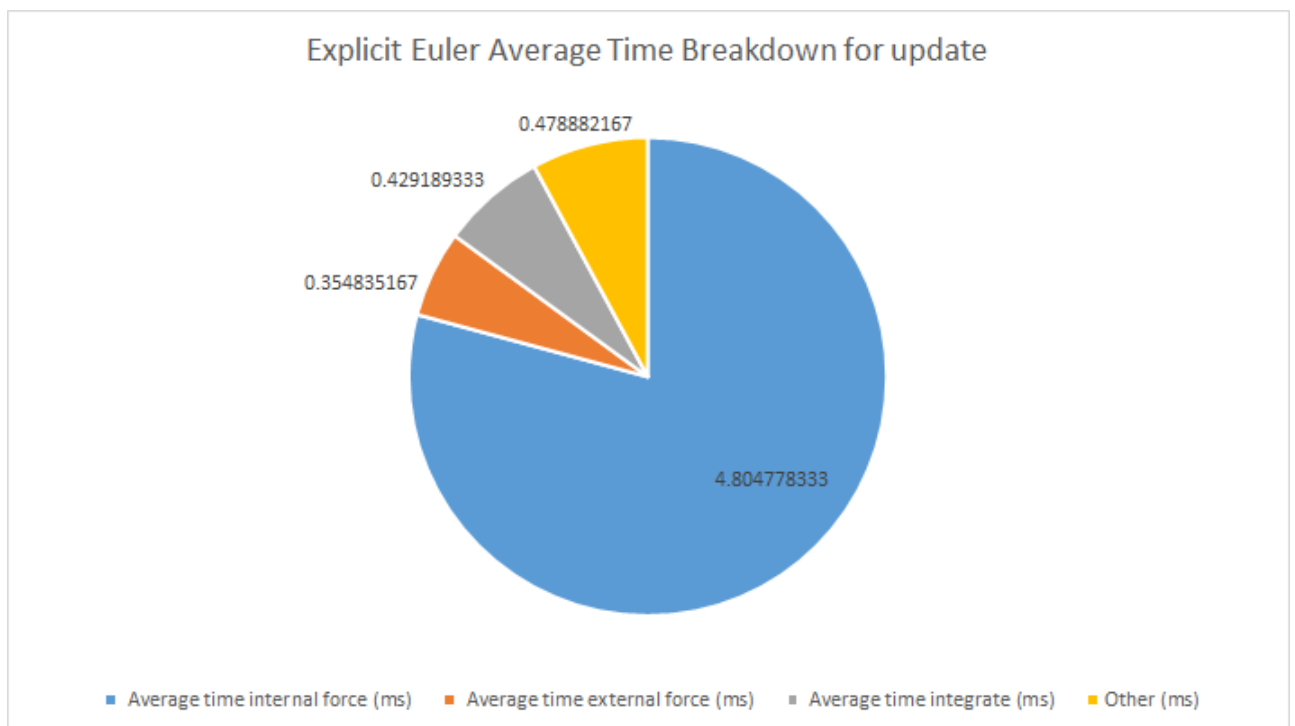
Figure C.7: Explicit Euler update time breakdown (sheet)
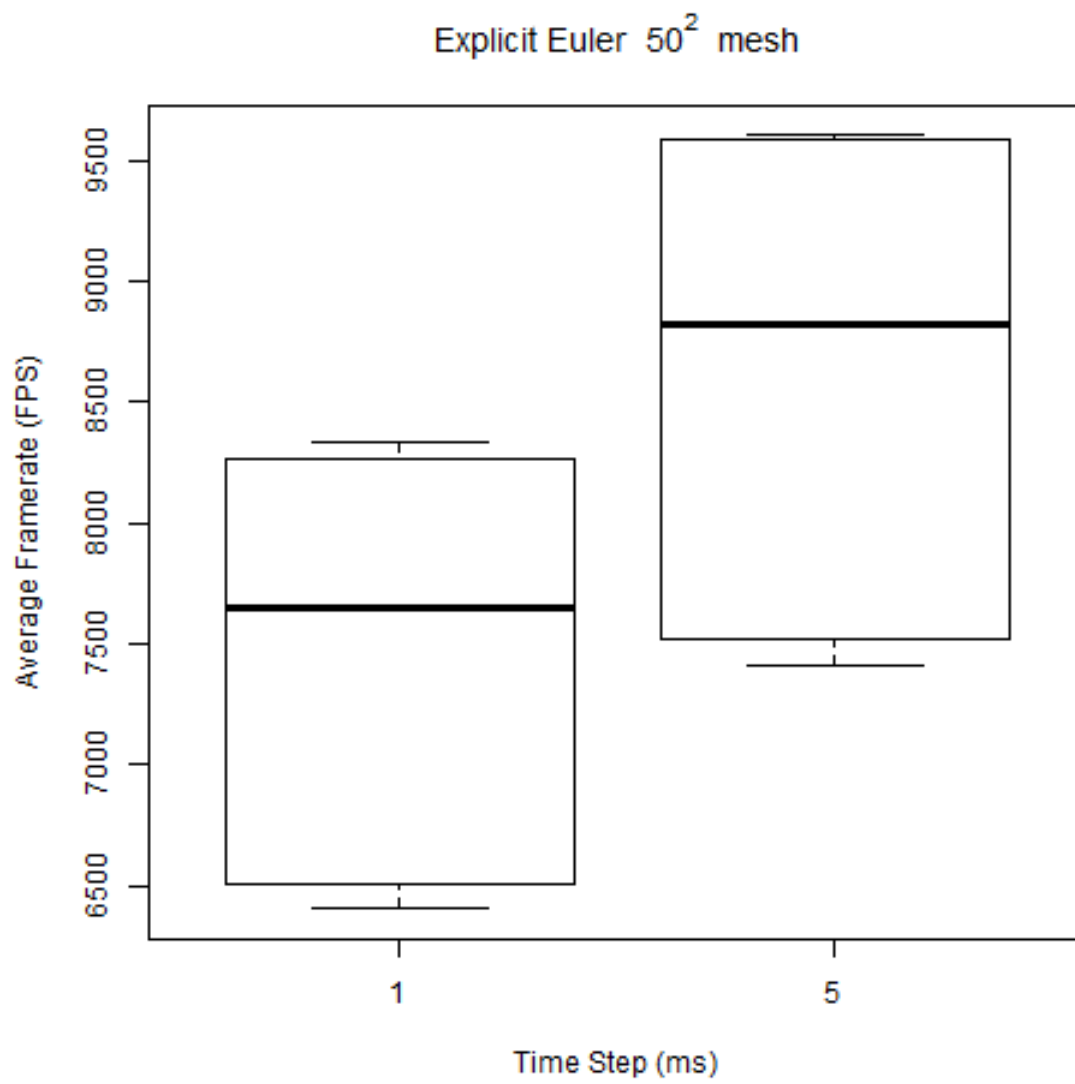
Figure C.8: Explicit Euler box plot for a 50^2 mesh (sheet)
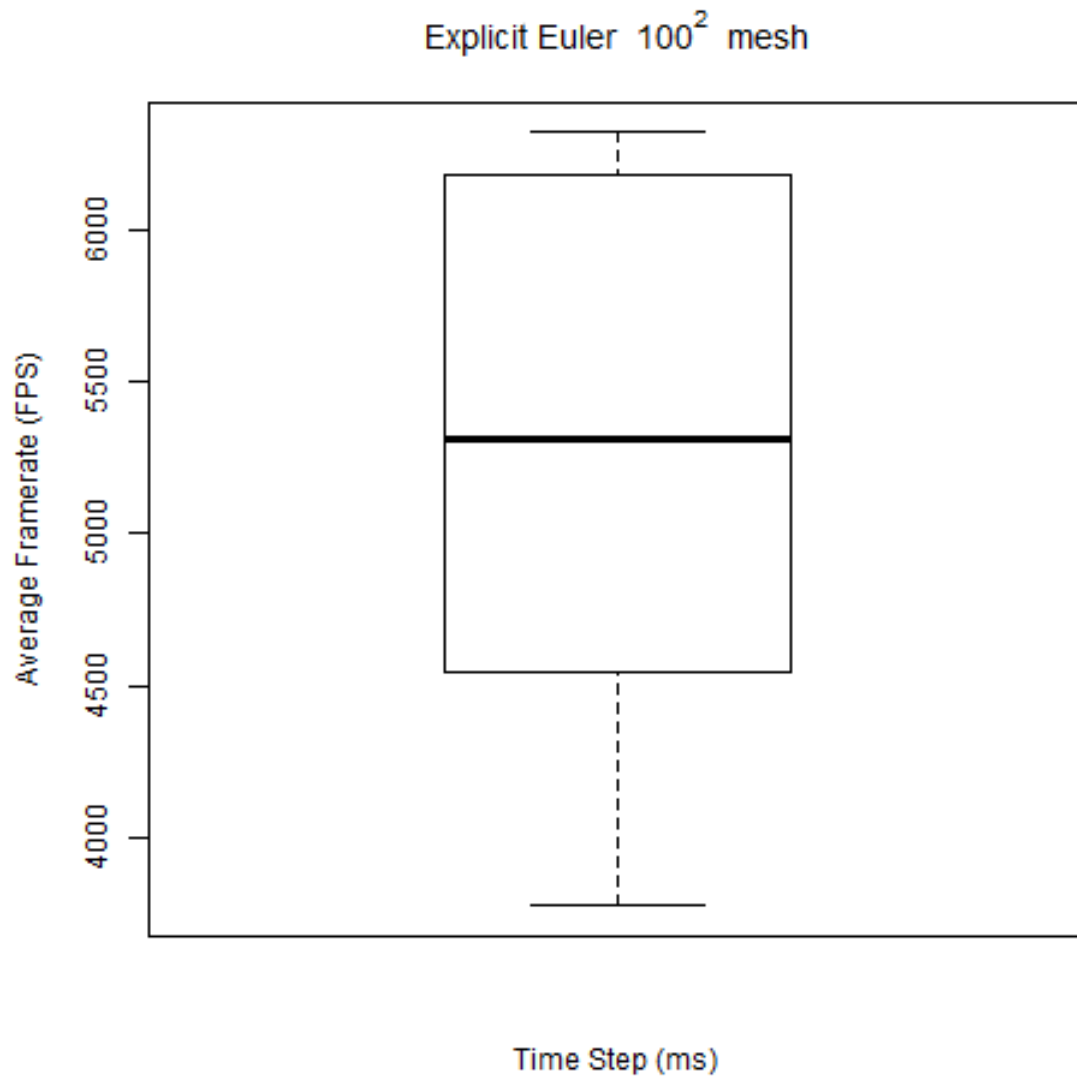
Figure C.9: Explicit Euler box plot for a 100^2 mesh (sheet)
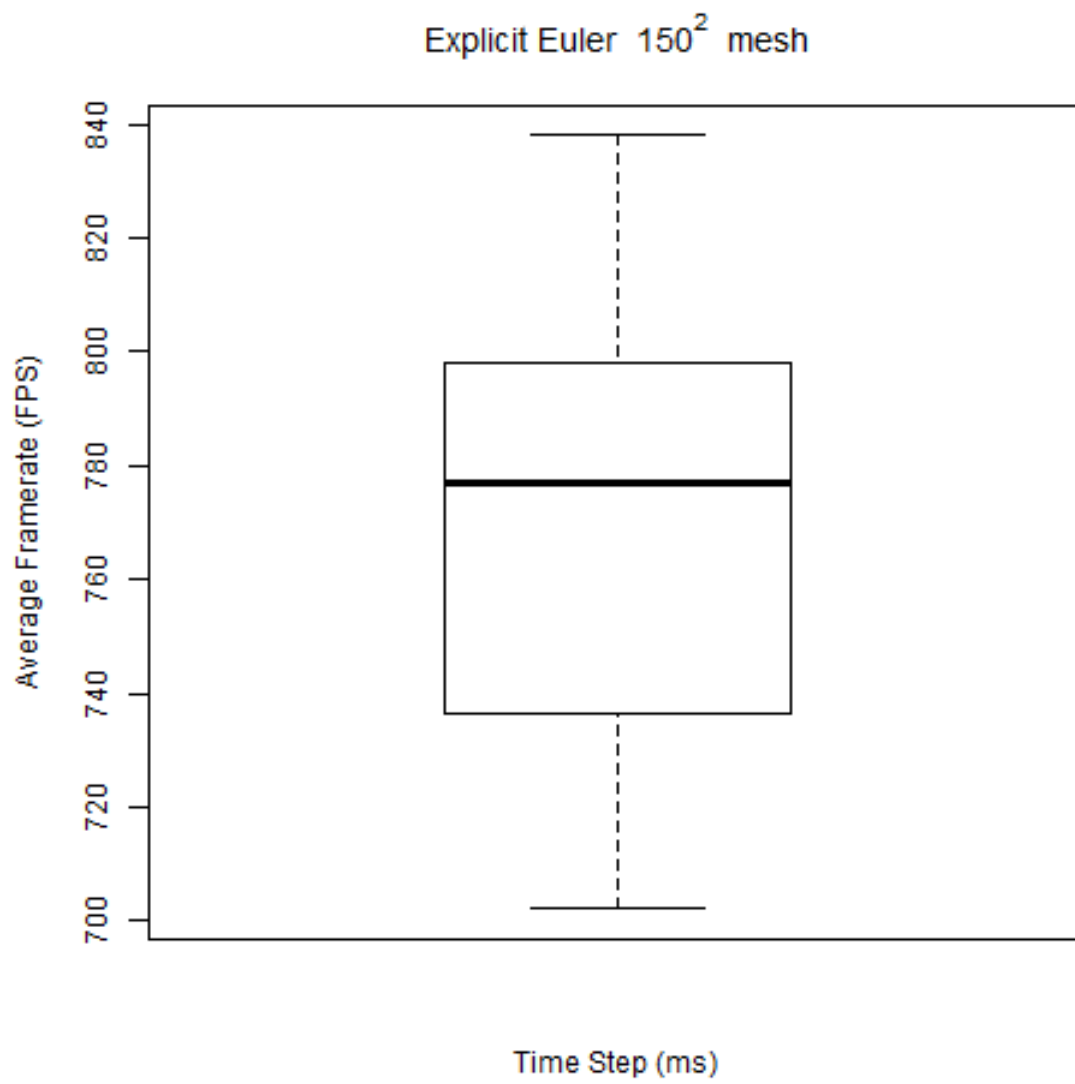
Figure C.10: Explicit Euler box plot for a 150^2 mesh (sheet)
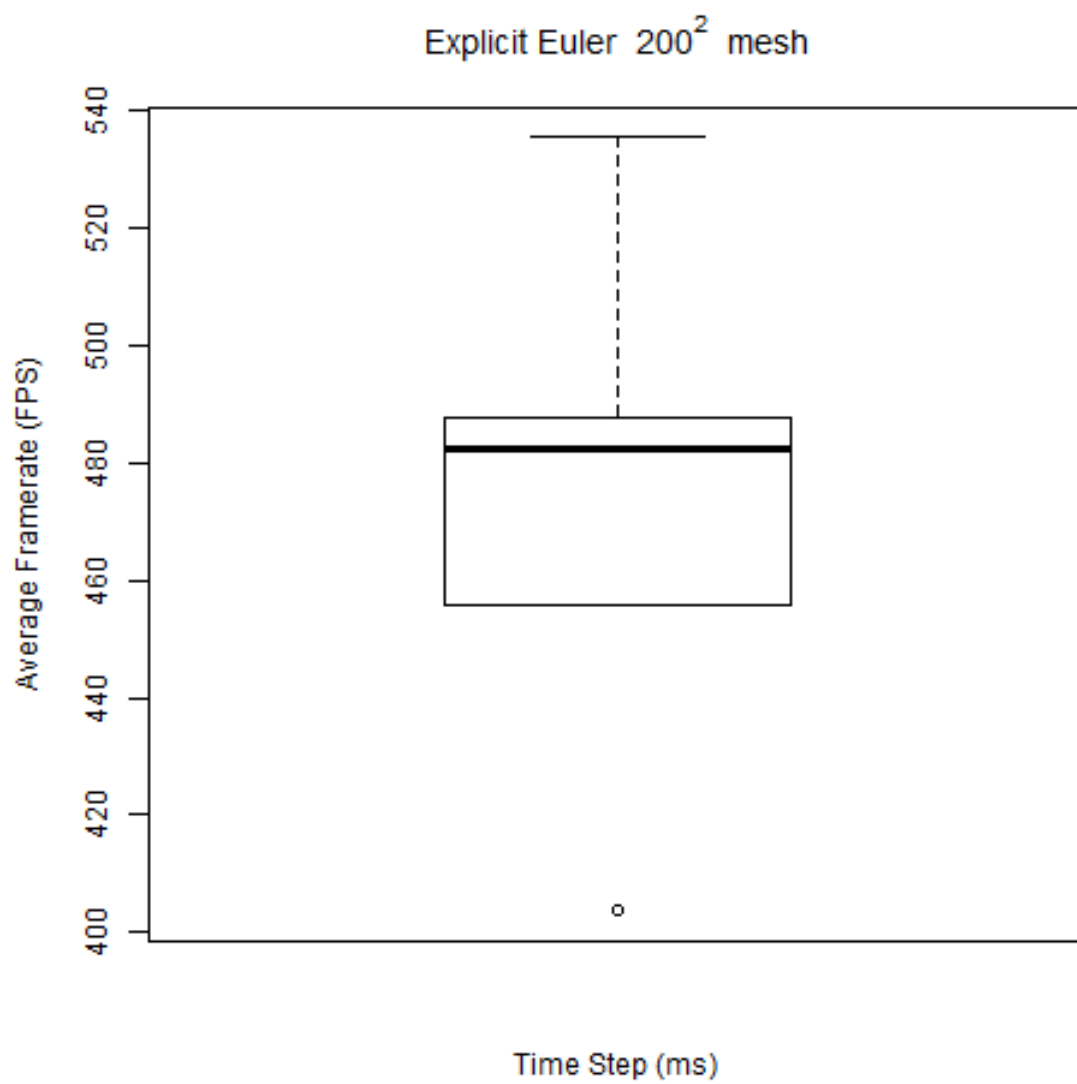
Figure C.11: Explicit Euler box plot for a 200^2 mesh (sheet)
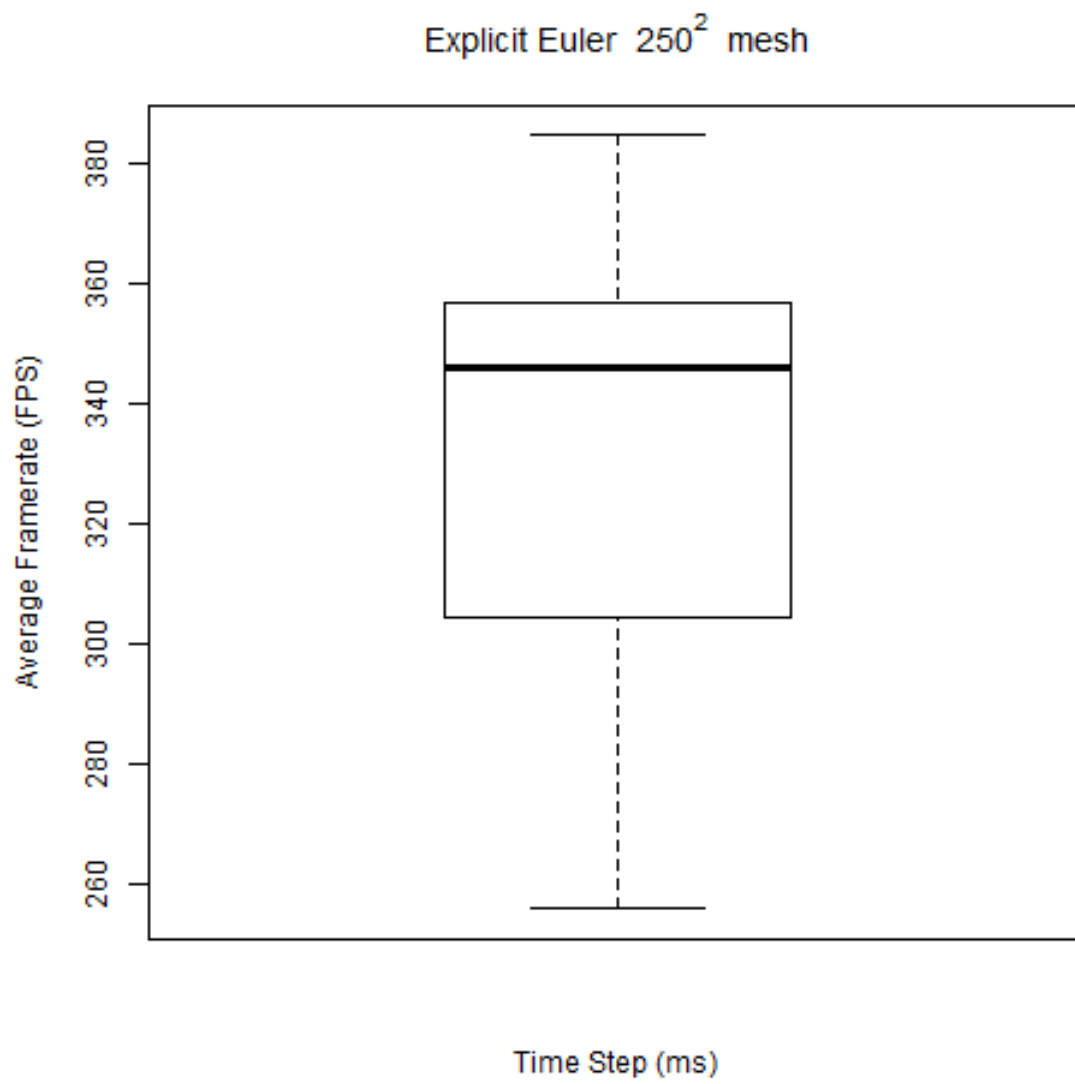
Figure C.12: Explicit Euler box plot for a 250^2 mesh (sheet)
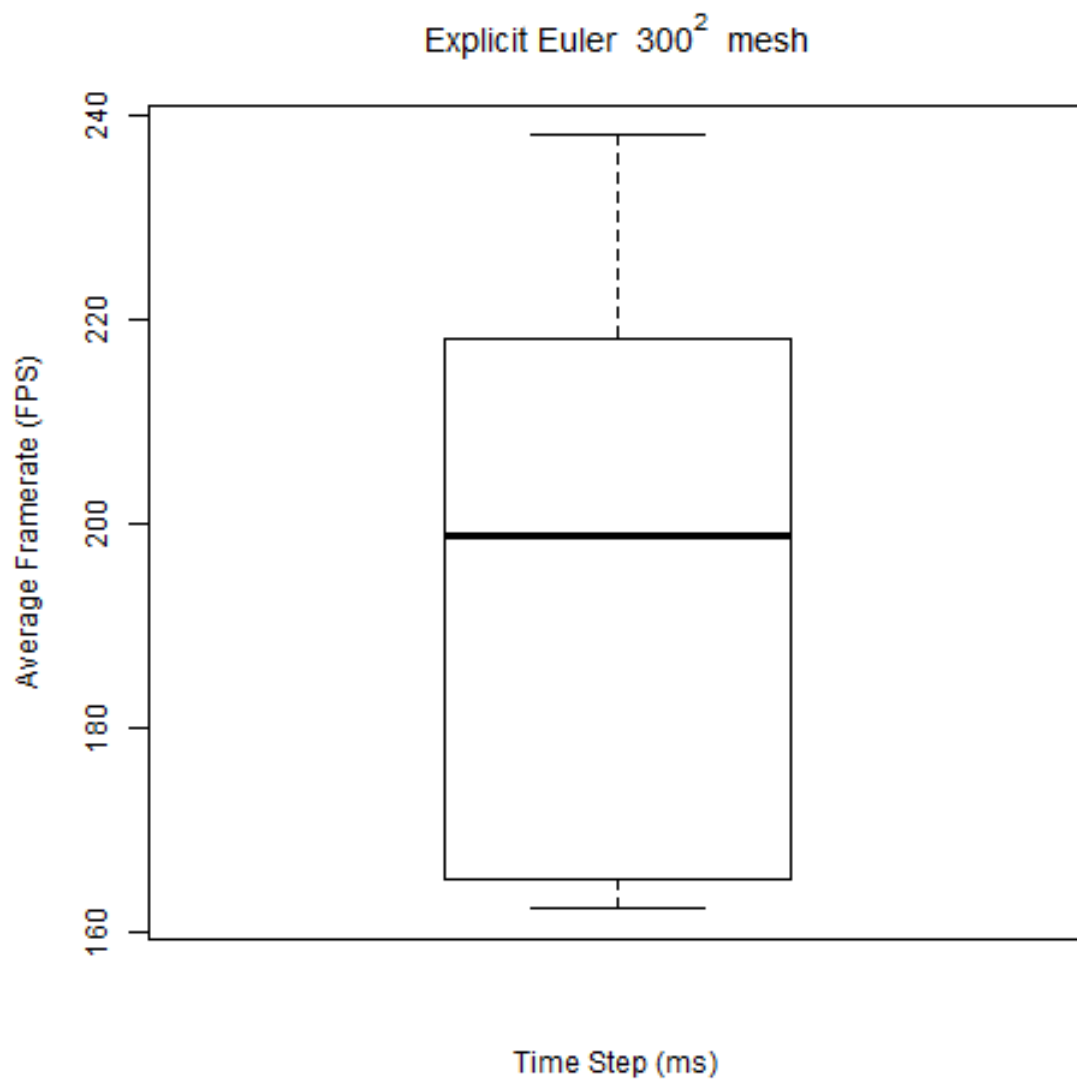
Figure C.13: Explicit Euler box plot for a 300^2 mesh (sheet)

Mesh Size	Time Step (ms)	Stable/Unstable
50^2	1	Stable
50^2	5	Stable
50^2	10	Unstable
50^2	15	Unstable
50^2	20	Unstable
100^2	1	Stable
100^2	5	Unstable
100^2	10	Unstable
100^2	15	Unstable
100^2	20	Unstable
150^2	1	Stable
150^2	5	Unstable
150^2	10	Unstable
150^2	15	Unstable
150^2	20	Unstable
200^2	1	Stable
200^2	5	Unstable
200^2	10	Unstable
200^2	15	Unstable
200^2	20	Unstable
250^2	1	Stable
250^2	5	Unstable
250^2	10	Unstable
250^2	15	Unstable
250^2	20	Unstable
300^2	1	Stable
300^2	5	Unstable
300^2	10	Unstable
300^2	15	Unstable
300^2	20	Unstable

Table C.2: Stability results for explicit Euler (sheet)

C.1.2 Flag Data

Rows	Columns	Time Step	Internal force (ms)	External force (ms)	Integrate (ms)	Update (ms)	Render (ms)	FPS	Update count
50	50	1	0.126889	0.0631805	0.00856284	0.211306	0.10033	7940.68	57092
50	50	1	0.141106	0.0634383	0.00830033	0.224213	0.128951	6109.13	56348
50	50	1	0.141075	0.0636839	0.00837112	0.224216	0.12593	6286.53	55315
50	50	1	0.127209	0.0627942	0.00850107	0.20965	0.118803	6714.48	57423
50	50	1	0.135137	0.0642465	0.0089046	0.220546	0.0999958	7836.6	58322
50	50	1	0.12822	0.0632535	0.00832739	0.2133	0.0991516	8041.83	56458
50	50	5	0.124505	0.0631798	0.00868232	0.223518	0.10018	9507.32	11914
50	50	5	0.138414	0.0634127	0.00832853	0.235313	0.129115	7366.53	11865
50	50	5	0.138764	0.0636472	0.00828404	0.234729	0.126708	7508.07	11873
50	50	5	0.125351	0.0628338	0.00856448	0.222627	0.119061	8012.15	11876
50	50	5	0.125754	0.0630455	0.00838532	0.224948	0.100569	9471.67	11905
50	50	5	0.125847	0.0631604	0.00832051	0.226444	0.0995307	9569.17	11903
50	50	10	0.121694	0.0631812	0.00868426	0.236722	0.100151	9717.05	5984
50	50	10	0.136681	0.0634118	0.00830805	0.247416	0.129165	7533.05	5971
50	50	10	0.135794	0.0636943	0.00829456	0.244779	0.126618	7686.63	5971
50	50	10	0.122746	0.0628267	0.00857752	0.234755	0.118965	8189.4	5975
50	50	10	0.122957	0.063031	0.00839502	0.238887	0.100627	9668.33	5981
50	50	10	0.122955	0.0631344	0.0083078	0.24112	0.0993443	9795.85	5981
50	50	15	0.120917	0.0631623	0.00870006	0.252018	0.100155	9783.85	3996
50	50	15	0.135424	0.0634524	0.0083932	0.259916	0.12915	7590.12	3989
50	50	15	0.135161	0.0636884	0.00830613	0.257013	0.126598	7744.35	3989
50	50	15	0.121959	0.0628387	0.00857292	0.248671	0.11892	8248.6	3991
50	50	15	0.122179	0.0630491	0.00839099	0.254544	0.100472	9755.35	3995
50	50	15	0.122144	0.0631493	0.0083046	0.258243	0.0994405	9853	3994

50	50	20	0.115169	0.0614182	0.00873345	0.259956	0.1014	9700.2	2999
50	50	20	0.129796	0.0616938	0.00836336	0.262755	0.136943	7187.1	2995
50	50	20	0.129366	0.0621318	0.00829096	0.269745	0.126925	7685.18	2989
50	50	20	0.115374	0.0611245	0.00857983	0.257008	0.113778	8648.05	2992
50	50	20	0.115919	0.0612923	0.00839589	0.263554	0.0978553	10038.3	2992
50	50	20	0.115967	0.0614763	0.00830258	0.268728	0.0995333	9885.92	2998
100	100	1	0.490834	0.256816	0.0363974	0.817499	0.0710994	2854.3	58595
100	100	1	0.53242	0.256586	0.0371431	0.859918	0.0738518	2279.58	58026
100	100	1	0.536516	0.255505	0.0366543	0.862498	0.0688329	2494.27	57610
100	100	1	0.493646	0.255716	0.0366469	0.818908	0.134358	1716.97	56360
100	100	1	0.490316	0.256131	0.0365797	0.816166	0.0691346	2891.52	58851
100	100	1	0.49127	0.256426	0.0367062	0.818111	0.0685072	2898.63	58784
100	100	5	0.478664	0.257202	0.0365974	0.825366	0.068368	12174.4	11947
100	100	5	0.48818	0.256506	0.0370851	0.831873	0.0861403	9658.12	11910
100	100	5	0.492281	0.25563	0.0367267	0.834561	0.0825711	10067	11927
100	100	5	0.479556	0.255446	0.0365764	0.81914	0.0932946	8948.5	11918
100	100	5	0.477862	0.256218	0.036617	0.821252	0.0671148	12416.3	11943
100	100	5	0.478343	0.256732	0.0367178	0.82409	0.0660139	12615.8	11948
100	100	10	0.463419	0.249768	0.0367093	0.824857	0.0692155	13193.5	5998
100	100	10	0.474932	0.24973	0.037184	0.832232	0.085642	10660.5	5988
100	100	10	0.476996	0.248577	0.036856	0.835696	0.0773269	11804.6	5990
100	100	10	0.465617	0.248889	0.036796	0.82039	0.0833328	10976.5	5988
100	100	10	0.461861	0.248871	0.036769	0.818349	0.0675437	13539.7	5994
100	100	10	0.462519	0.249194	0.0367234	0.822689	0.0662882	13791.3	5994
100	100	15	0.464789	0.248928	0.0368516	0.847234	0.0693375	13543.1	4003
100	100	15	0.474042	0.249253	0.0371504	0.851062	0.087051	10800.6	3998

100	100	15	0.475834	0.248027	0.0368367	0.855762	0.0778931	12058.4	4000
100	100	15	0.464435	0.24855	0.0367584	0.838388	0.0826306	11388.5	3997
100	100	15	0.460938	0.248771	0.0368066	0.838185	0.0674177	13940.5	4000
100	100	15	0.461903	0.248834	0.0367428	0.843268	0.06638	14160.8	4000
100	100	20	0.465758	0.248978	0.0368933	0.870204	0.0694714	13703.8	3004
100	100	20	0.47466	0.249244	0.0372039	0.869774	0.0856658	11118.2	3000
100	100	20	0.476701	0.24808	0.0367869	0.906017	0.0787547	12050.5	3005
100	100	20	0.465018	0.248564	0.0367385	0.858681	0.0827635	11521.1	3000
100	100	20	0.461863	0.248646	0.0368581	0.859074	0.067703	14071.3	3003
100	100	20	0.462336	0.248867	0.0368093	0.865371	0.0665592	14312.9	3002
150	150	1	1.09998	0.57973	0.0822618	1.83258	0.0859599	521.483	31271
150	150	1	1.11744	0.579988	0.0839165	1.85875	0.0887618	514.533	30806
150	150	1	1.12086	0.578167	0.0829554	1.85288	0.0832781	517.883	30987
150	150	1	1.10047	0.57891	0.0828753	1.83385	0.172003	498.967	29910
150	150	1	1.09305	0.578992	0.082572	1.8253	0.0832979	525.667	31434
150	150	1	1.09334	0.579831	0.0829948	1.82716	0.0836744	524.4	31398
150	150	5	1.08766	0.580016	0.0829858	1.83741	0.0686337	9190.28	11935
150	150	5	1.0944	0.580102	0.0838362	1.85445	0.0878333	7138.85	11915
150	150	5	1.10046	0.578399	0.082931	1.84604	0.0834672	7540.48	11916
150	150	5	1.08537	0.57804	0.0828254	1.83137	0.085574	7402.55	11909
150	150	5	1.08023	0.579302	0.0827029	1.82799	0.0676601	9318.32	11936
150	150	5	1.08108	0.580396	0.0829579	1.83097	0.0662746	9518.92	11937
150	150	10	1.06678	0.563371	0.0832629	1.82356	0.0693068	11751.8	5993
150	150	10	1.07396	0.56302	0.0839456	1.83215	0.0881184	9246.88	5981
150	150	10	1.07804	0.560878	0.0830607	1.82599	0.0832585	9796.15	5986
150	150	10	1.06289	0.561162	0.083046	1.80945	0.0927812	8808.58	5979

150	150	10	1.05831	0.562308	0.0828574	1.8114	0.0677105	12055.8	5991
150	150	10	1.05837	0.56271	0.0829599	1.81372	0.0664148	12286.9	5991
150	150	15	1.06824	0.5629	0.0832894	1.84673	0.0690795	12637.3	4001
150	150	15	1.07518	0.563152	0.0839694	1.85095	0.0883169	9895.16	3996
150	150	15	1.07871	0.560708	0.0830127	1.84477	0.0837819	10431.8	3997
150	150	15	1.06276	0.560668	0.0830418	1.82615	0.0928281	9434.97	3994
150	150	15	1.05918	0.562024	0.0827627	1.83262	0.0677027	12908.1	3999
150	150	15	1.06041	0.562732	0.0830221	1.83835	0.0666909	13110.1	3999
150	150	20	1.07016	0.563065	0.0834242	1.87117	0.0691069	13050.5	3003
150	150	20	1.07667	0.563161	0.0833823	1.87021	0.088236	10232.8	2999
150	150	20	1.08081	0.560692	0.0831541	1.86603	0.0836471	10799.9	2999
150	150	20	1.06379	0.560544	0.0831226	1.84481	0.0922837	9806.08	2999
150	150	20	1.06187	0.562125	0.0829679	1.8561	0.0672709	13424.2	3002
150	150	20	1.06187	0.562903	0.0830401	1.86248	0.0663225	13614.7	3001
200	200	1	1.9745	1.03552	0.14812	3.28903	0.120043	294.15	17599
200	200	1	2.03521	1.04191	0.150457	3.37444	0.148415	284.567	17031
200	200	1	2.12621	1.03759	0.149849	3.4424	0.110561	282.433	16887
200	200	1	1.9695	1.03232	0.147687	3.27569	0.200314	287.8	17261
200	200	1	1.96751	1.03716	0.148666	3.28387	0.119586	295.033	17628
200	200	1	1.96317	1.03572	0.148094	3.27625	0.120651	295.167	17662
200	200	5	1.937	1.01008	0.149503	3.24147	0.0704393	5051.18	11932
200	200	5	1.93312	1.00454	0.14814	3.24004	0.126054	2865.93	11837
200	200	5	1.9504	1.00449	0.148615	3.24333	0.0843671	4223.73	11917
200	200	5	1.91801	1.00248	0.147621	3.20371	0.0922406	3946.6	11904
200	200	5	1.93138	1.00698	0.152229	3.2371	0.0708963	5048.85	11919
200	200	5	1.91487	1.00638	0.148222	3.20979	0.0677751	5341.47	11932

200	200	10	1.93668	1.00504	0.148407	3.25748	0.0691158	9737.4	5990
200	200	10	1.99056	1.00907	0.149987	3.3195	0.142958	4679.92	5962
200	200	10	2.02998	1.00447	0.148734	3.34927	0.0830442	7999.27	5983
200	200	10	1.91952	1.00152	0.147695	3.22585	0.0917345	7381.28	5977
200	200	10	1.94326	1.00616	0.152826	3.27193	0.0698808	9619.61	5986
200	200	10	1.92368	1.00595	0.148194	3.24243	0.0671648	10040.8	5988
200	200	15	1.94321	1.00634	0.149288	3.2913	0.0723178	10747.4	3997
200	200	15	1.96182	1.00513	0.148763	3.29557	0.14491	5381.87	3985
200	200	15	2.11203	1.00703	0.149194	3.45946	0.082934	9239.9	3997
200	200	15	1.92562	1.00164	0.148002	3.2511	0.0916712	8527.13	3993
200	200	15	1.93822	1.00525	0.148652	3.27599	0.0723035	10769.5	3995
200	200	15	1.92849	1.00617	0.148374	3.26861	0.072387	10769.2	3994
200	200	20	1.94708	1.00541	0.148617	3.30773	0.0773535	10743.3	3000
200	200	20	1.96477	1.00463	0.148503	3.30879	0.146404	5687.92	2992
200	200	20	2.19139	1.01072	0.149685	3.5673	0.0831503	9840.35	3001
200	200	20	1.92793	1.00174	0.148011	3.27146	0.0916773	9101.28	2998
200	200	20	1.95353	1.00769	0.151879	3.31806	0.0781592	10629	2999
200	200	20	1.93397	1.00666	0.149037	3.29283	0.0772839	10769.9	2998
250	250	1	3.11302	1.62171	0.240512	5.20875	0.174243	186.567	11146
250	250	1	3.14758	1.62452	0.24712	5.26296	0.181524	183.95	11020
250	250	1	3.13043	1.62428	0.24248	5.23326	0.172417	185.7	11099
250	250	1	3.11125	1.62105	0.240884	5.20914	0.253531	184.15	10983
250	250	1	3.1169	1.6236	0.242617	5.21928	0.173831	186.2	11125
250	250	1	3.09627	1.62571	0.239014	5.19563	0.175484	186.633	11171
250	250	5	3.04187	1.57492	0.241708	5.09756	0.172743	227.817	11382
250	250	5	3.07092	1.57959	0.247093	5.14285	0.183673	198.8	11262

250	250	5	3.05839	1.57647	0.247594	5.12625	0.166771	217.183	11334
250	250	5	3.03066	1.57215	0.242729	5.08286	0.25184	188.8	11244
250	250	5	3.03682	1.57367	0.241058	5.08628	0.17435	227.65	11403
250	250	5	3.02026	1.57671	0.238252	5.06743	0.173584	213.35	11446
250	250	10	3.0708	1.57696	0.242301	5.15436	0.0711735	6810.03	5985
250	250	10	3.348	1.58157	0.246363	5.44405	0.087733	5205.62	5979
250	250	10	3.36935	1.57757	0.246459	5.45888	0.0809367	5636.8	5979
250	250	10	3.06045	1.57423	0.242354	5.13661	0.0897538	5428.88	5977
250	250	10	3.06082	1.57468	0.243278	5.14423	0.0704907	6891.32	5983
250	250	10	3.04975	1.58289	0.245645	5.14485	0.0696185	6973.27	5983
250	250	15	3.07765	1.57848	0.248795	5.20405	0.0879117	7415.63	3986
250	250	15	3.43911	1.76734	0.270191	5.77557	0.0861042	7121.87	3994
250	250	15	3.41892	1.75442	0.282707	5.79614	0.0827542	7399.67	3990
250	250	15	3.06882	1.57626	0.247058	5.18661	0.088656	7363.12	3992
250	250	15	3.09151	1.58421	0.262508	5.24586	0.0888002	7312.03	3986
250	250	15	3.05796	1.58083	0.245333	5.1704	0.0881487	7427.93	3983
250	250	20	3.08398	1.58017	0.253092	5.23074	0.0992669	7418.85	2994
250	250	20	3.45298	1.77648	0.287492	5.88178	0.0952943	7382.3	2997
250	250	20	3.42842	1.75214	0.280574	5.80749	0.0957347	7395.53	2993
250	250	20	3.08419	1.58182	0.248688	5.22559	0.0996957	7393.3	2993
250	250	20	3.08327	1.5779	0.248951	5.21062	0.0993838	7422.48	2994
250	250	20	3.06371	1.58188	0.246463	5.19504	0.0994093	7430.67	2994
300	300	1	4.8473	2.42626	0.424046	8.16817	0.200034	119.95	7170
300	300	1	4.87882	2.43505	0.425804	8.21045	0.213911	118.817	7122
300	300	1	4.85572	2.43158	0.431433	8.19147	0.194902	119.567	7155
300	300	1	4.8472	2.42966	0.423805	8.17031	0.283102	118.783	7098

300	300	1	4.8436	2.42563	0.423282	8.16432	0.20429	119.833	7170
300	300	1	4.79875	2.42406	0.418037	8.10271	0.205201	120.5	7222
300	300	5	4.73164	2.35266	0.42312	7.98162	0.202434	123.6	7330
300	300	5	4.74445	2.36356	0.42669	8.0086	0.212301	122.6	7297
300	300	5	4.73057	2.35401	0.416084	7.9625	0.200096	123.7	7349
300	300	5	4.70413	2.35091	0.419937	7.93946	0.277454	122.567	7301
300	300	5	4.68648	2.3558	0.425236	7.93622	0.203416	124.183	7370
300	300	5	4.66938	2.35048	0.414061	7.89206	0.204302	124.567	7409
300	300	10	4.74684	2.35477	0.425322	8.01385	0.0774982	2624.25	5978
300	300	10	4.87881	2.36869	0.427029	8.16233	0.0947965	1981.07	5974
300	300	10	4.8815	2.35643	0.419896	8.14134	0.0863702	2205.28	5976
300	300	10	4.72368	2.35232	0.416036	7.96701	0.0975412	2151.32	5972
300	300	10	4.72343	2.35069	0.421439	7.9761	0.0768156	2690.78	5978
300	300	10	4.687	2.35388	0.41851	7.93861	0.0746133	2803.35	5978
300	300	15	4.74601	2.35802	0.45109	8.10921	0.0863129	5331.73	3983
300	300	15	5.1464	2.58264	0.46079	8.71463	0.0844088	4961.78	3992
300	300	15	5.10329	2.56545	0.486575	8.74352	0.0789175	5284.53	3991
300	300	15	4.74286	2.35761	0.440742	8.06162	0.0894031	5172.45	3991
300	300	15	4.73313	2.35571	0.4463	8.08325	0.0867767	5328.5	3980
300	300	15	4.70092	2.35659	0.438878	8.03449	0.0873419	5330.83	3981
300	300	20	4.75397	2.35772	0.451147	8.12704	0.111567	5323.62	2985
300	300	20	5.12018	2.57836	0.499309	8.81301	0.10497	5328	2989
300	300	20	5.10824	2.57073	0.490803	8.7761	0.105383	5328.45	2987
300	300	20	4.74325	2.35763	0.441928	8.09556	0.111884	5323.17	2985
300	300	20	4.74025	2.35583	0.448088	8.09512	0.112178	5315.03	2981
300	300	20	4.70013	2.35818	0.442914	8.05038	0.112472	5320.42	2982

Table C.3: Raw data for explicit Euler (flag)
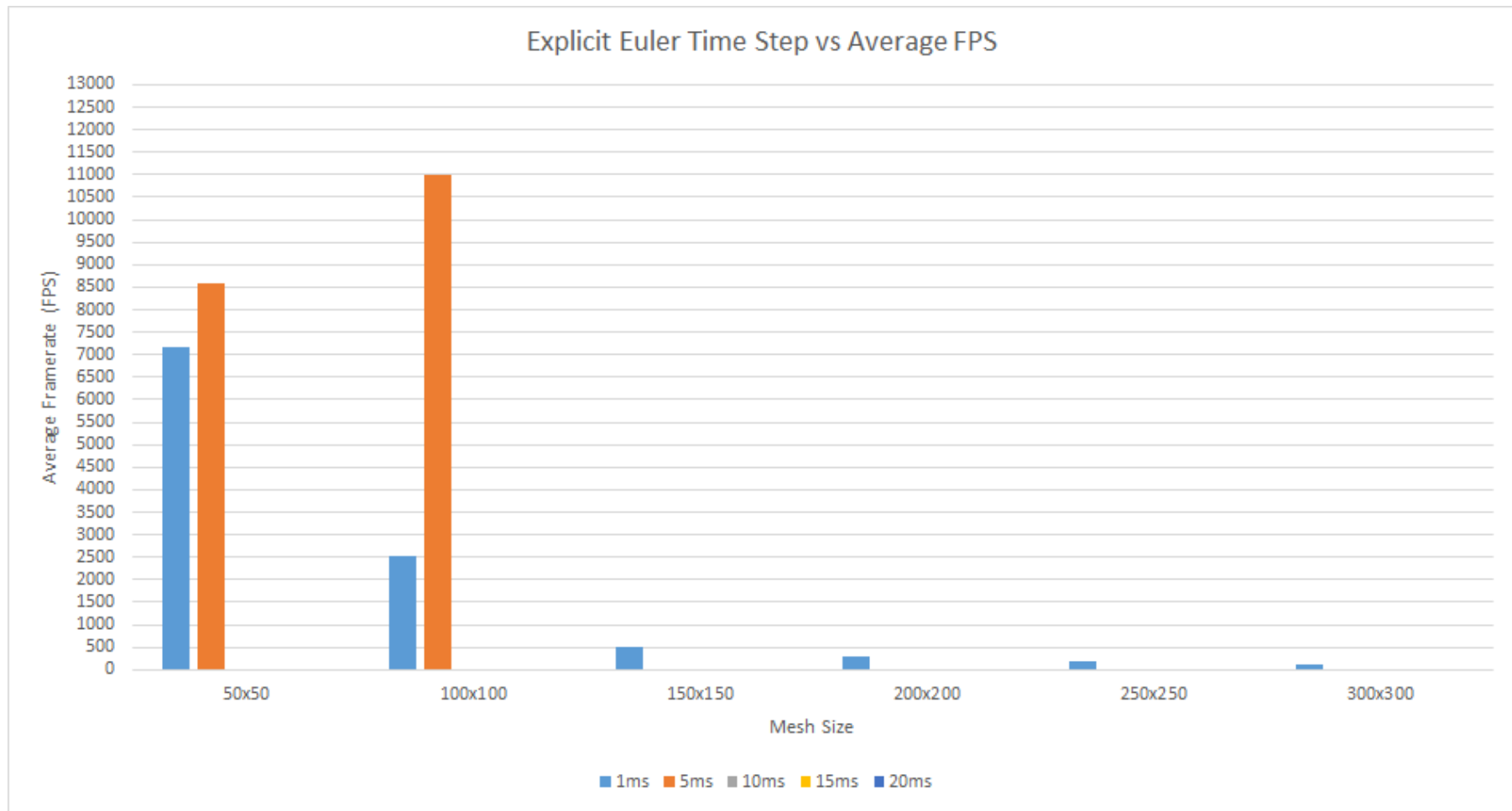
Figure C.14: Explicit Euler mesh size against average FPS (flag)
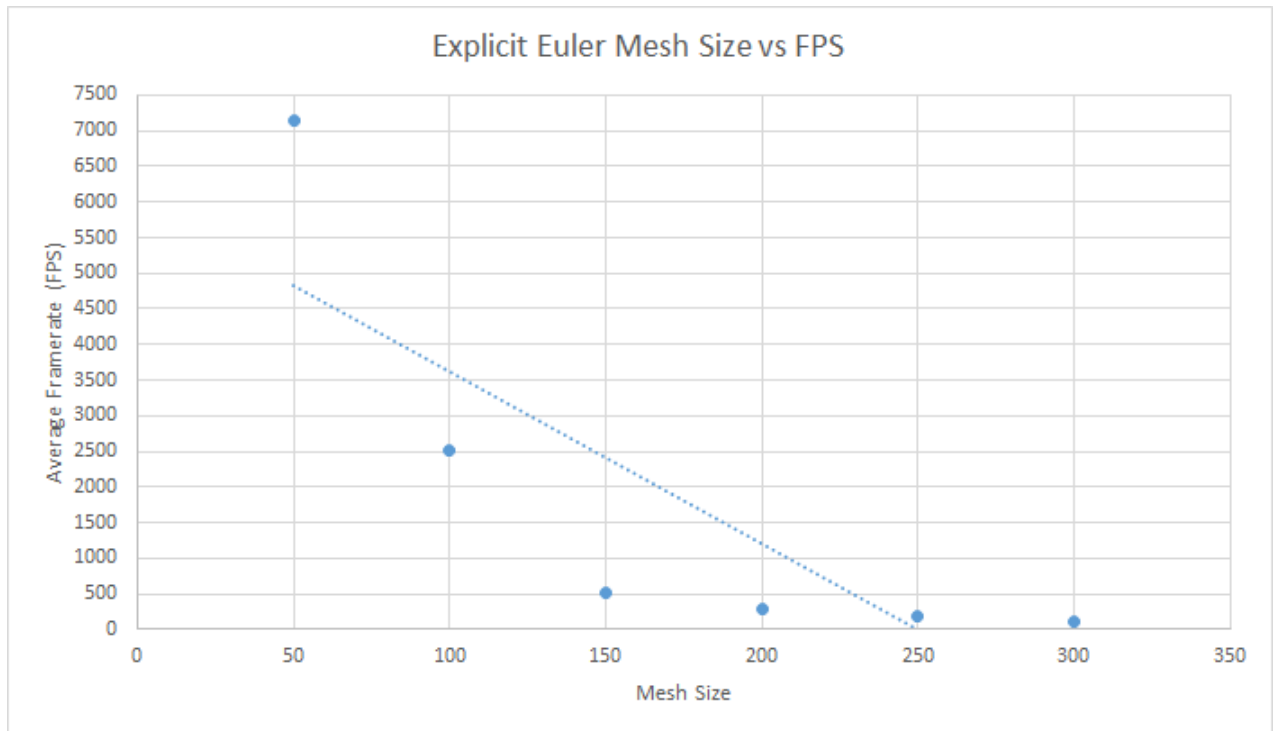
Figure C.15: Explicit Euler mesh size against average FPS (flag)
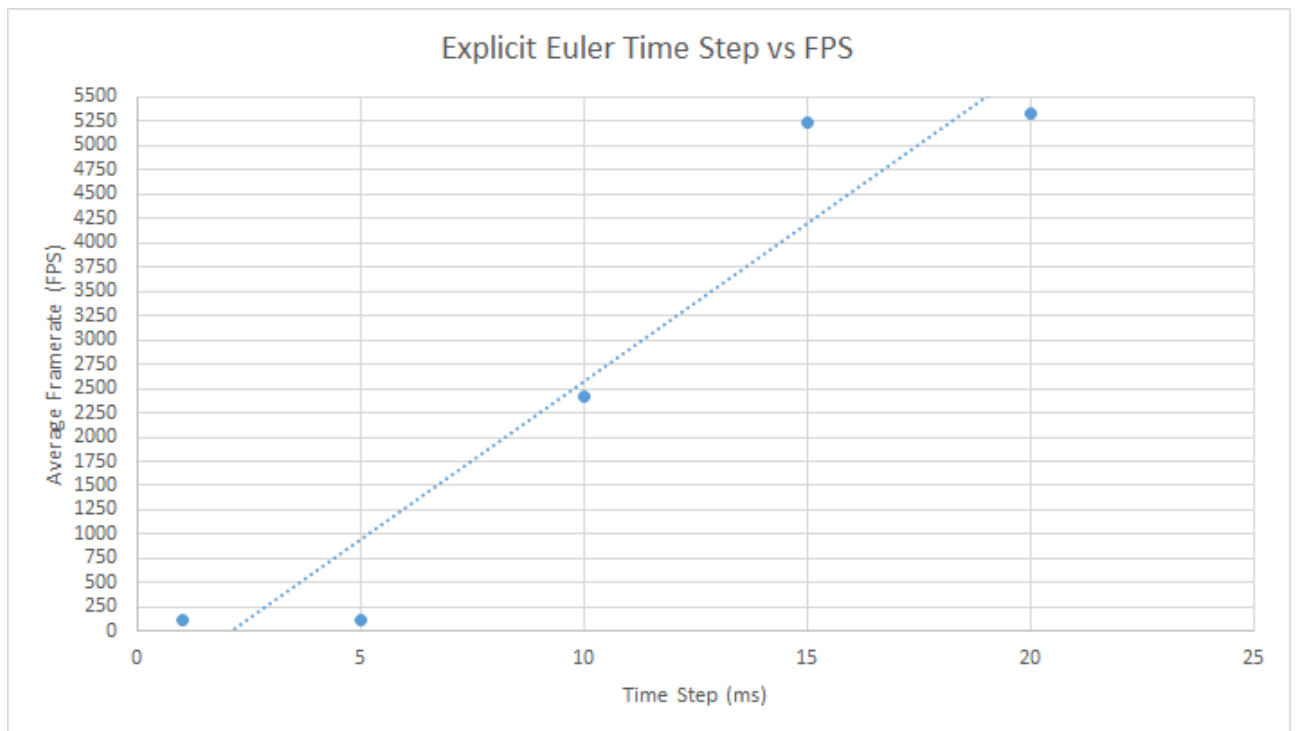
Figure C.16: Explicit Euler time step against average FPS for a 300^2 mesh (flag)
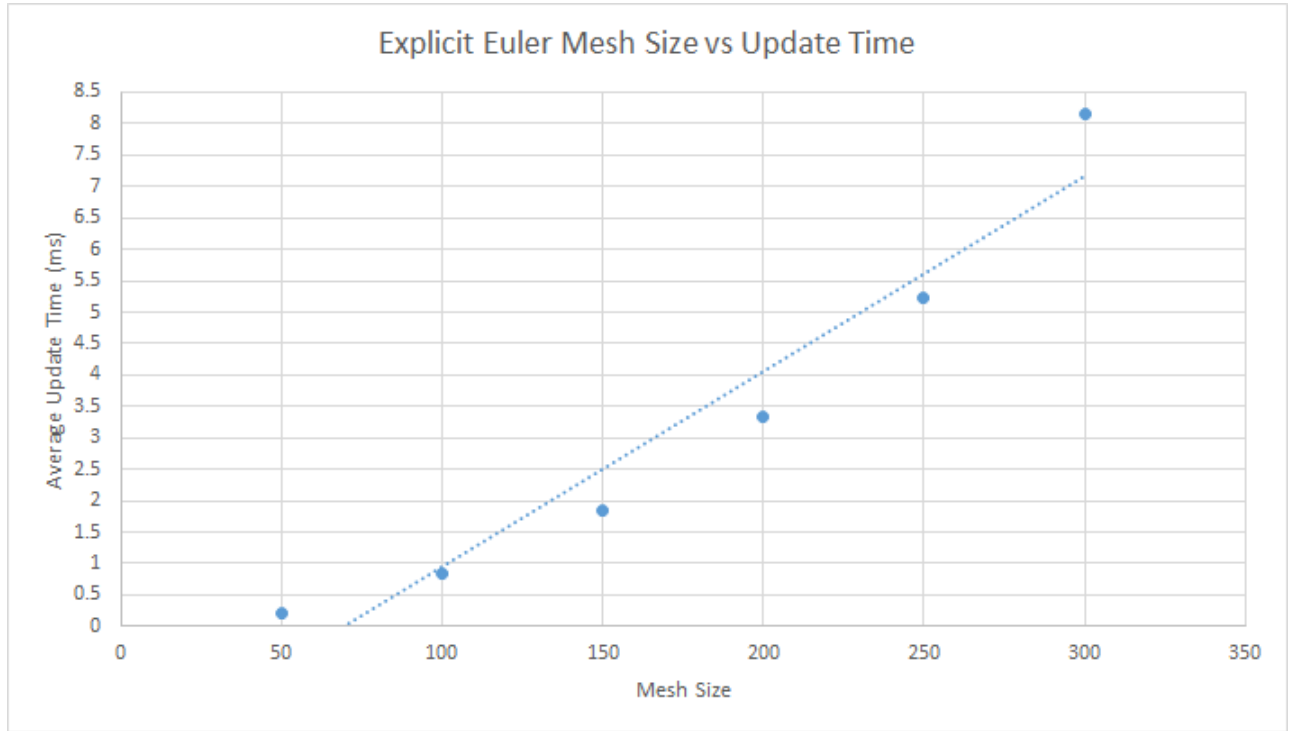
Figure C.17: Explicit Euler mesh size against average update time (flag)
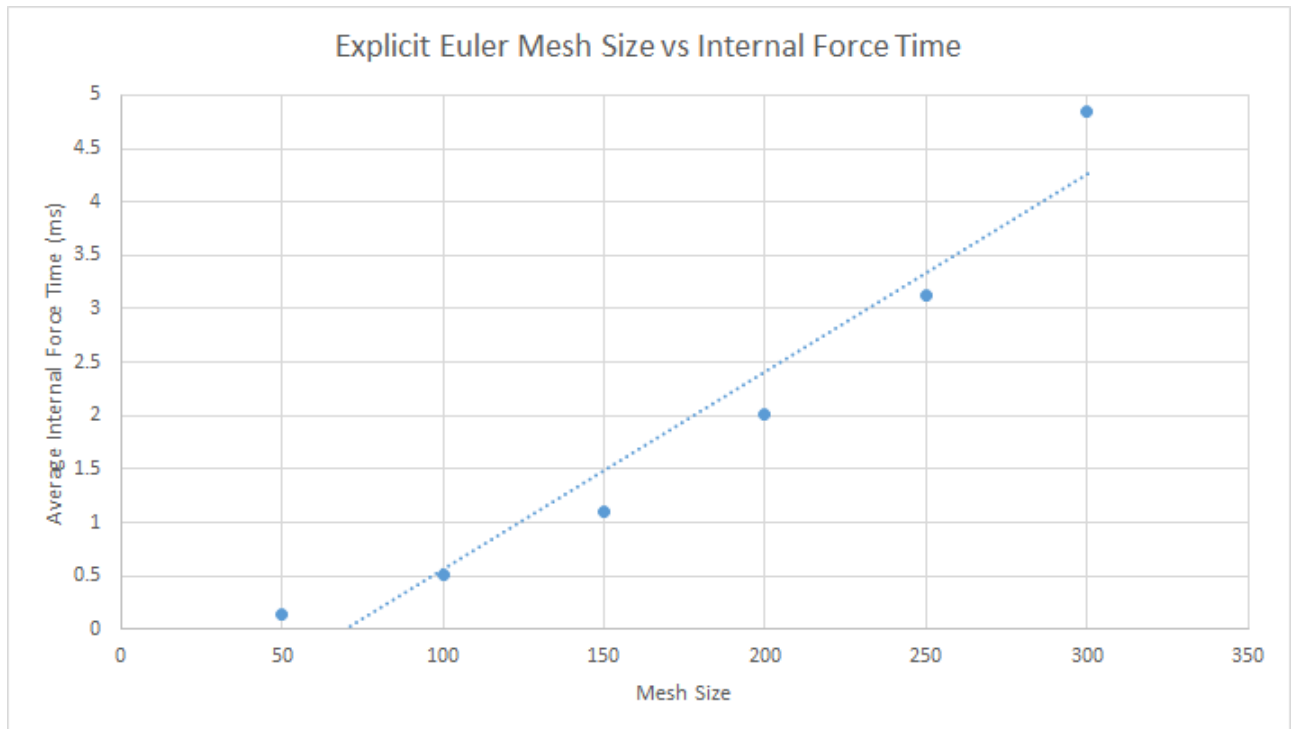
Figure C.18: Explicit Euler mesh size against average internal force time (flag)
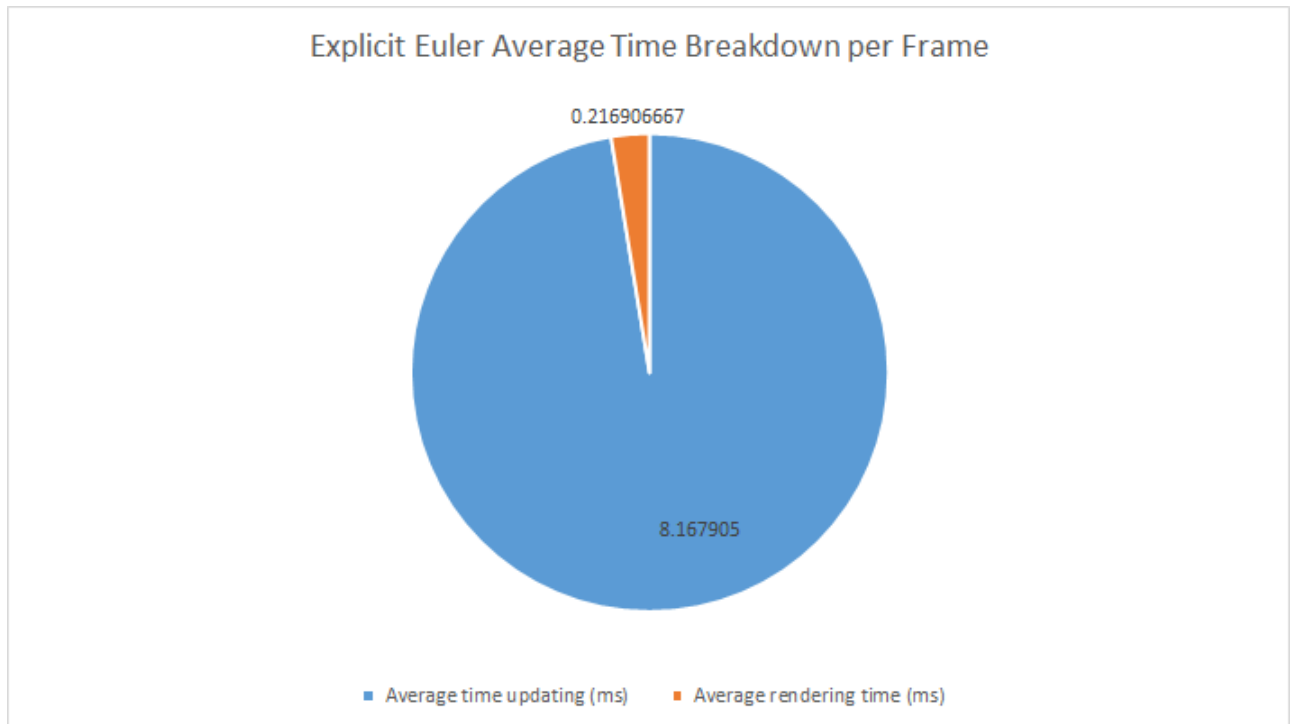
Figure C.19: Explicit Euler frame time breakdown (flag)
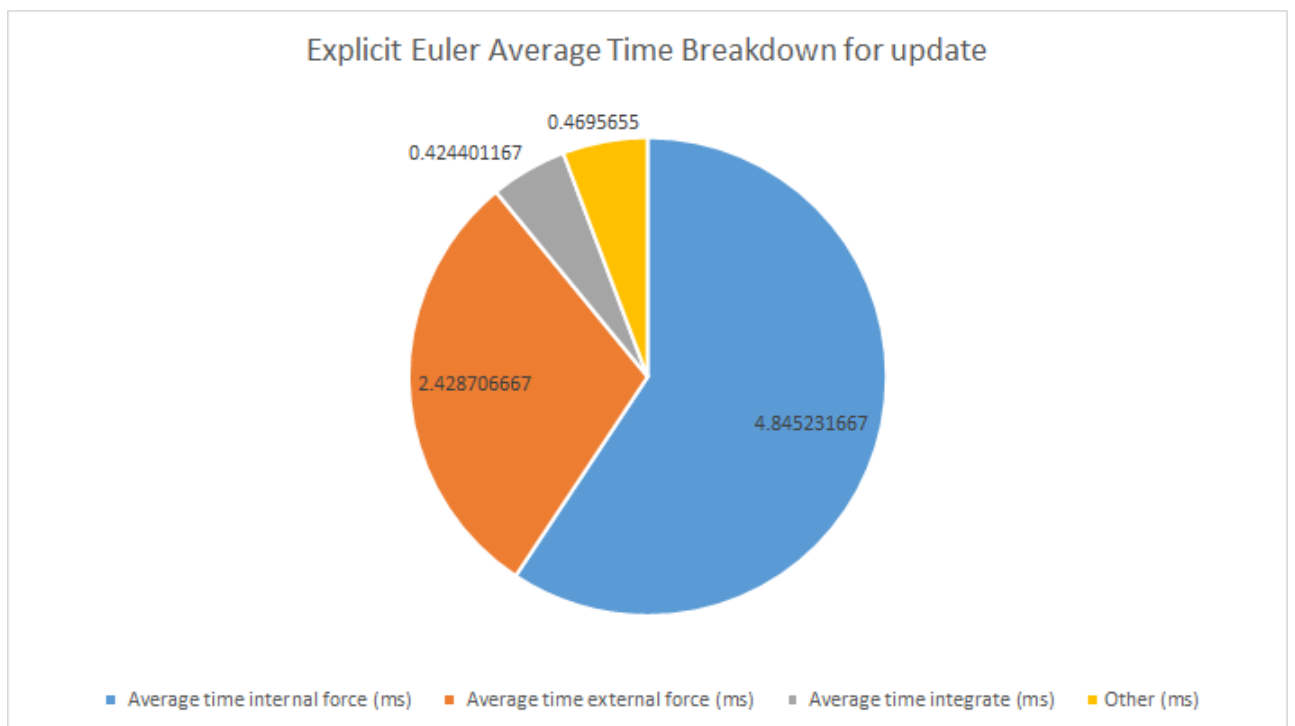
Figure C.20: Explicit Euler update time breakdown (flag)

104
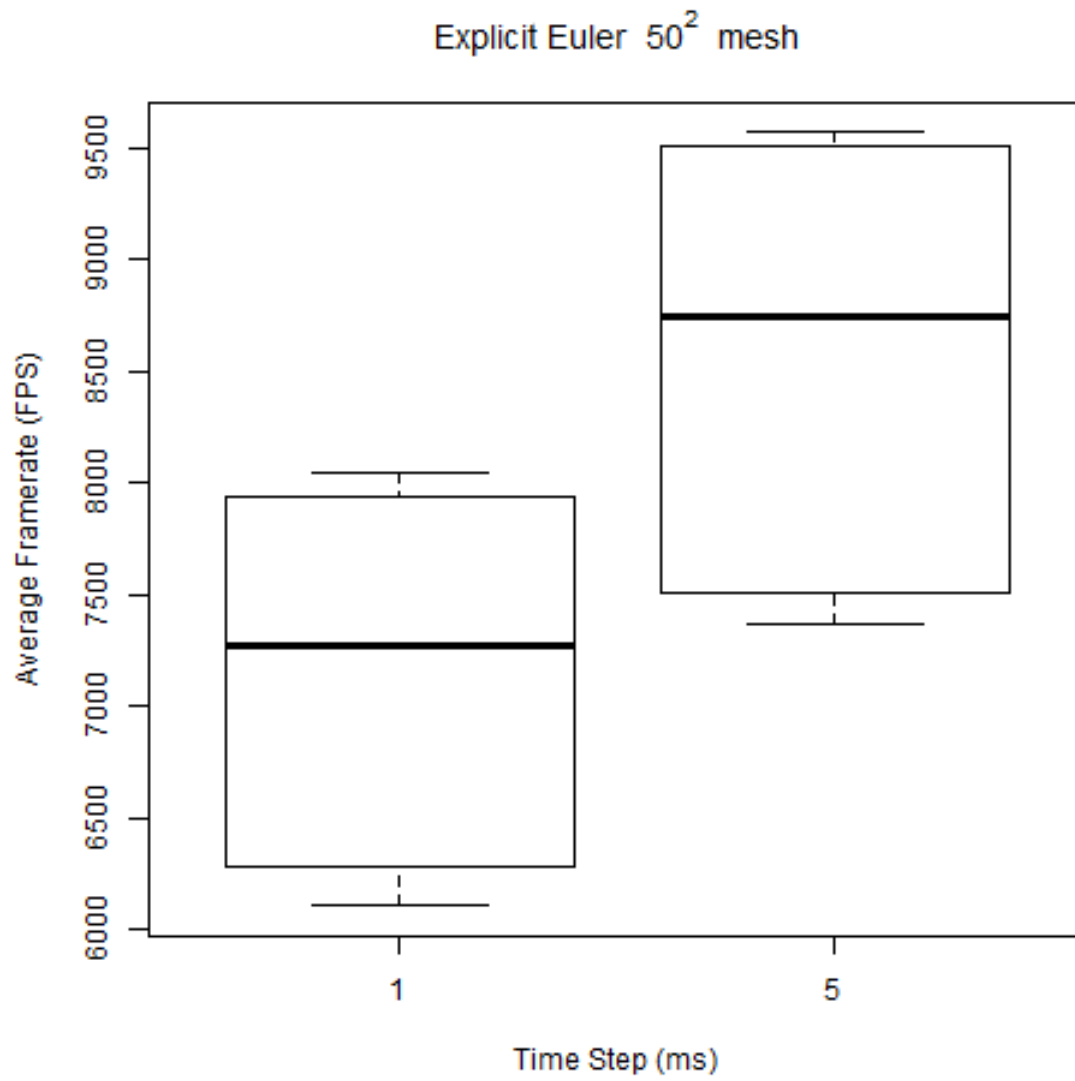
Figure C.21: Explicit Euler box plot for a 50^2 mesh (flag)
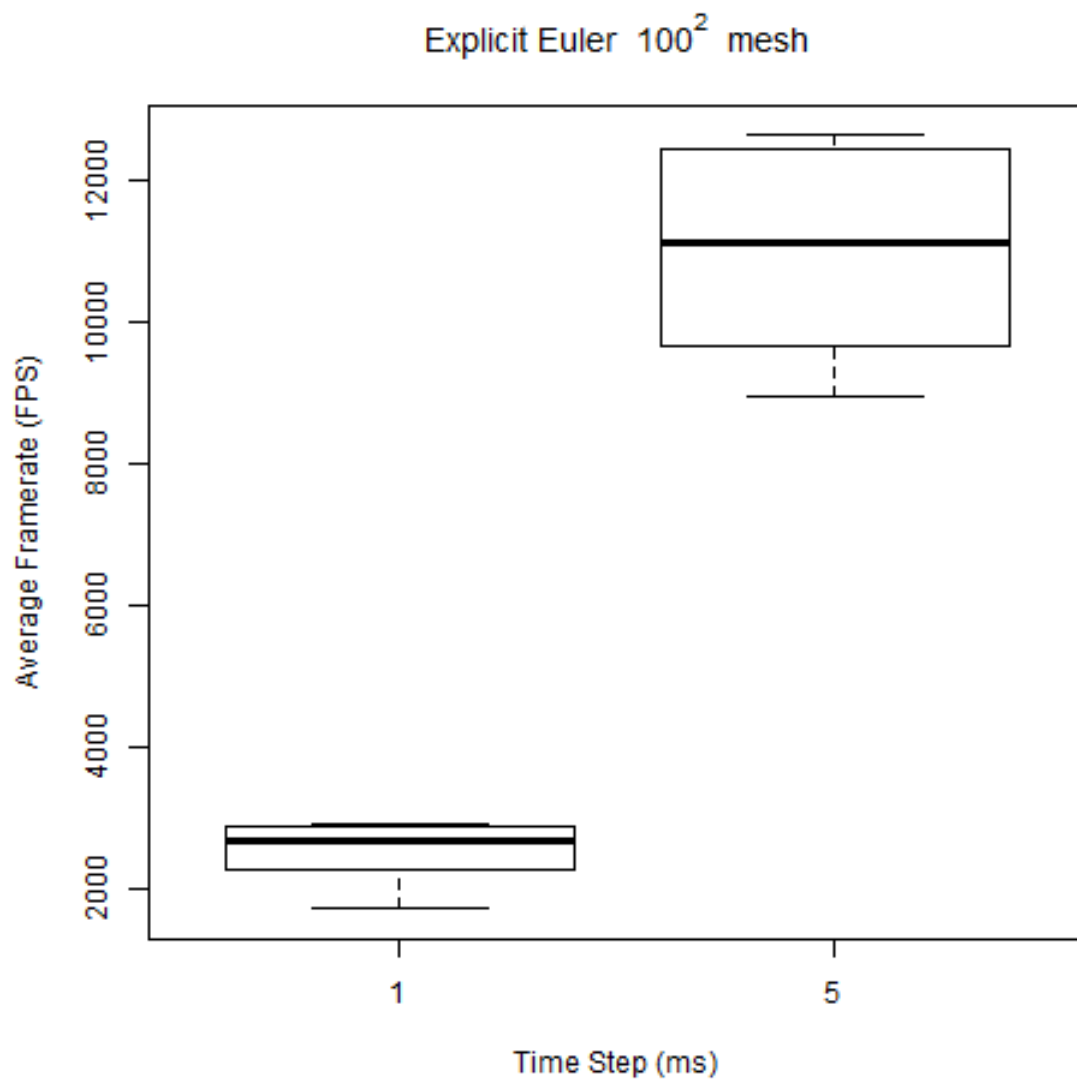
Figure C.22: Explicit Euler box plot for a 100^2 mesh (flag)
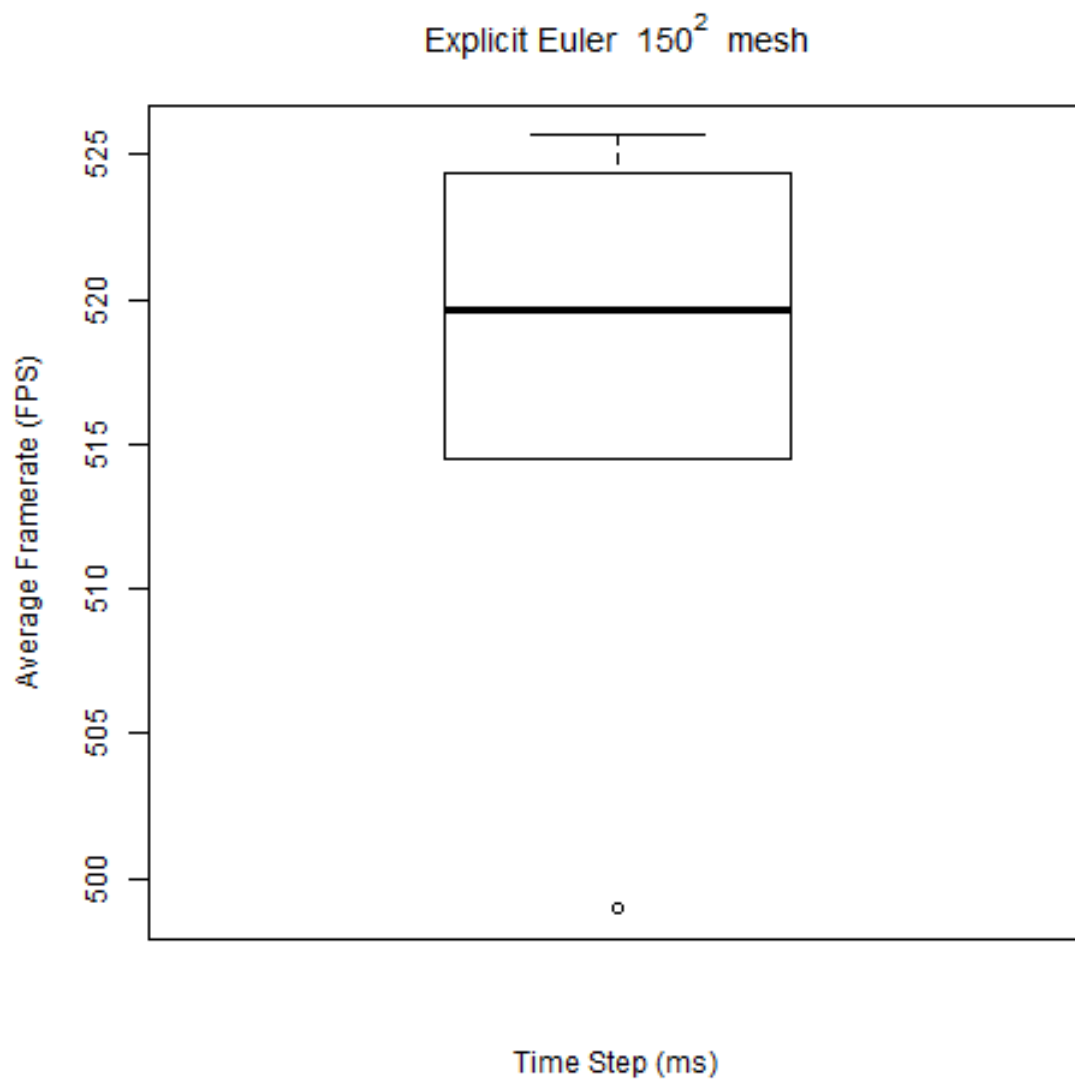
Figure C.23: Explicit Euler box plot for a 150^2 mesh (flag)
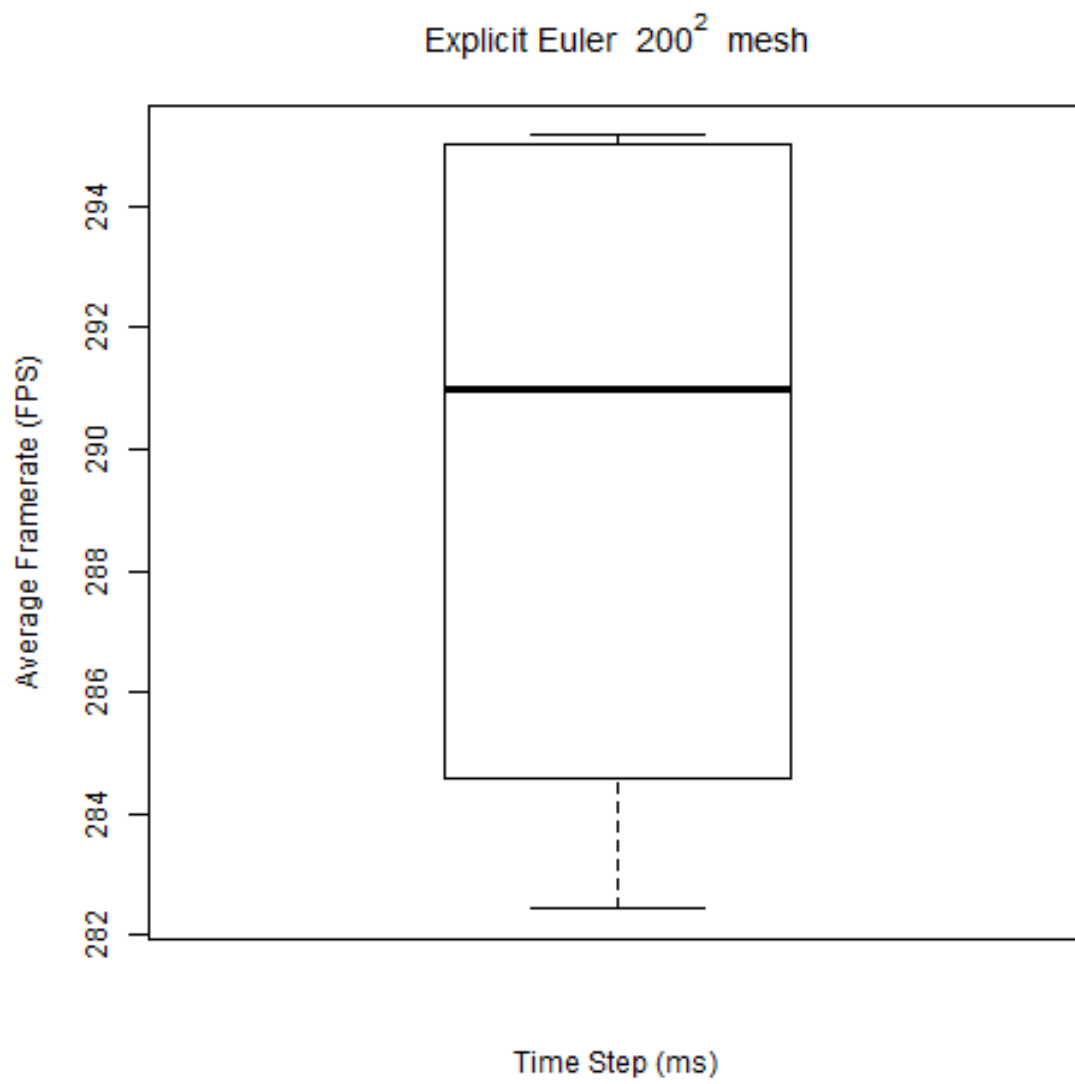
Figure C.24: Explicit Euler box plot for a 200^2 mesh (flag)
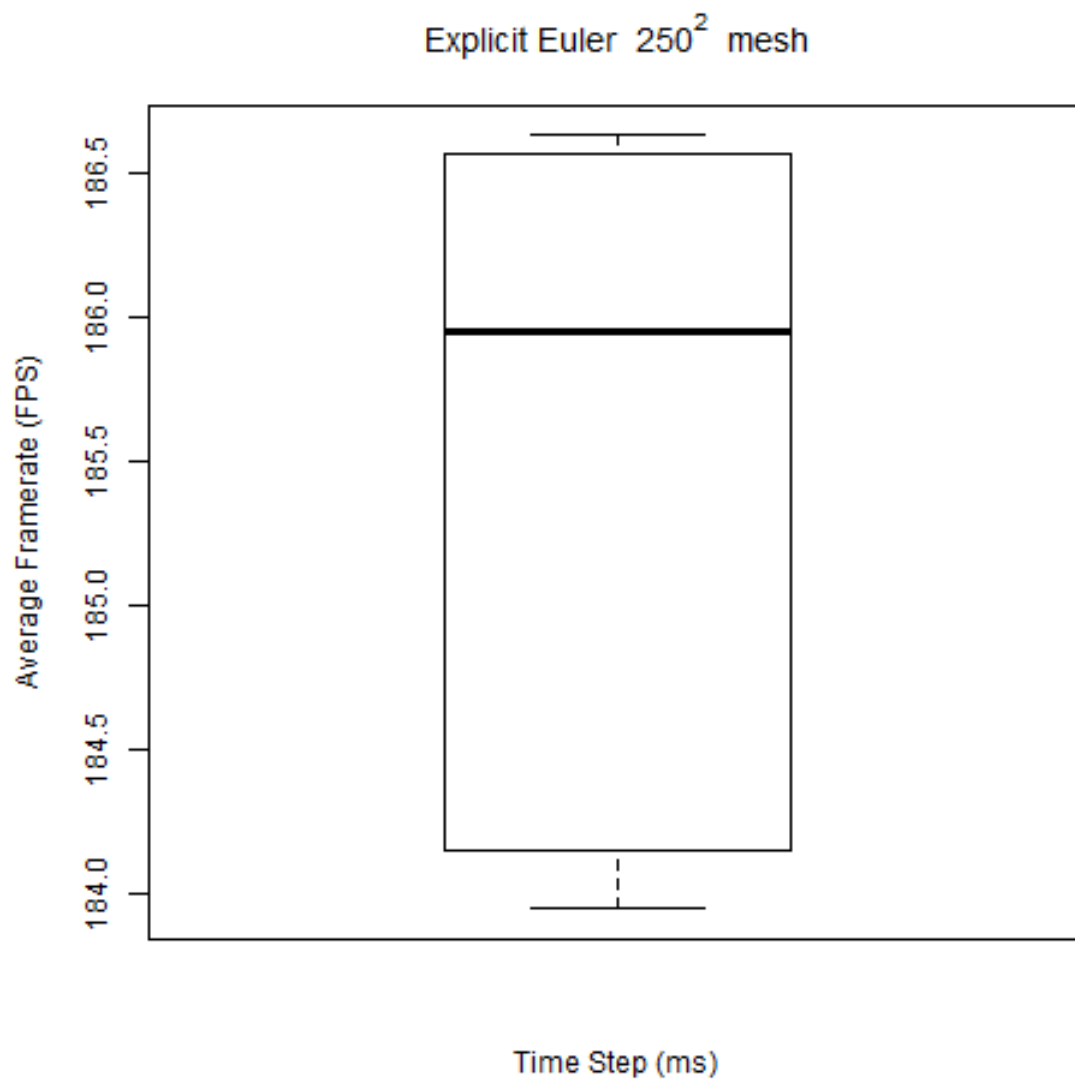
Figure C.25: Explicit Euler box plot for a 250^2 mesh (flag)
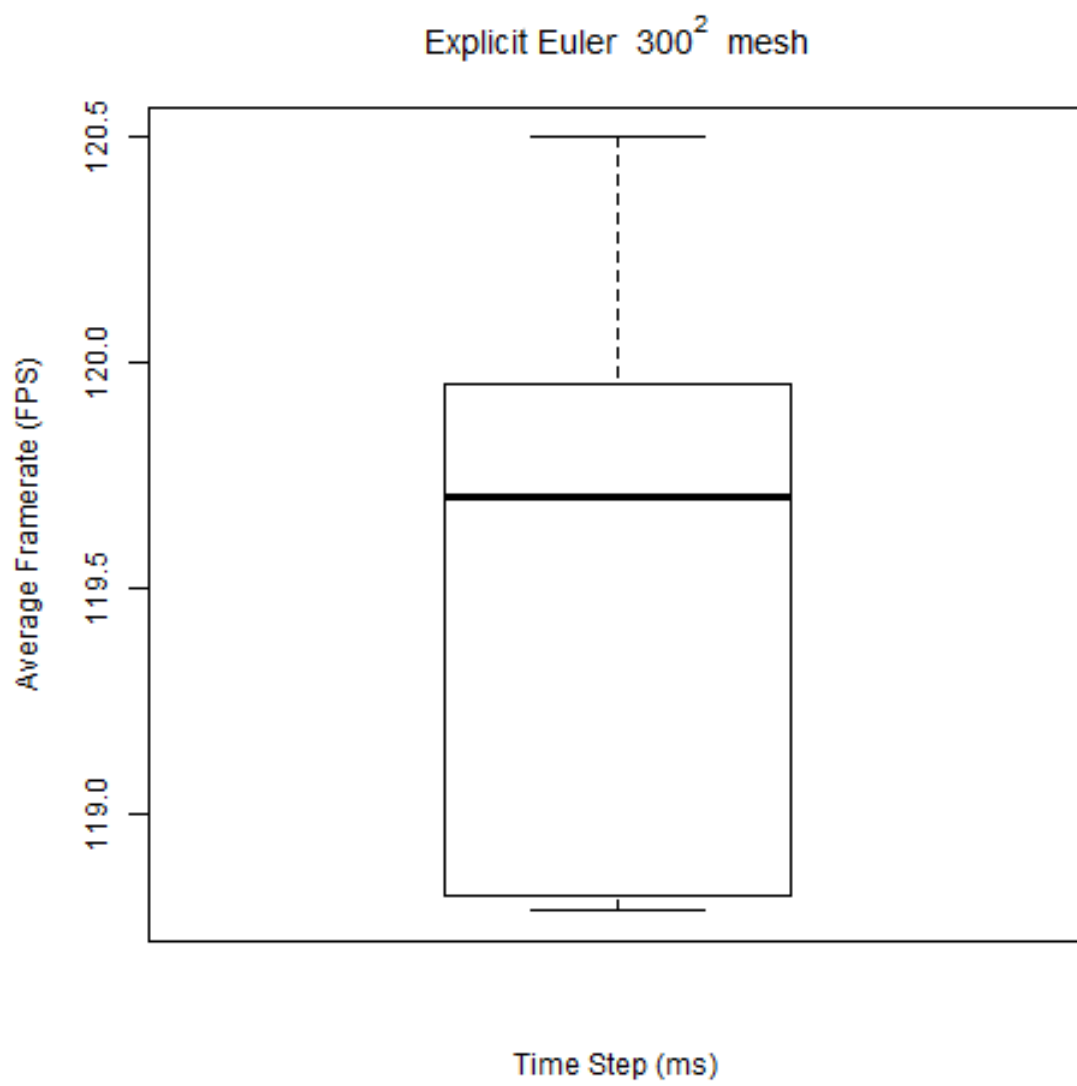
Figure C.26: Explicit Euler box plot for a 300^2 mesh (flag)

Mesh Size	Time Step (ms)	Stable/Unstable
50^2	1	Stable
50^2	5	Stable
50^2	10	Unstable
50^2	15	Unstable
50^2	20	Unstable
100^2	1	Stable
100^2	5	Stable
100^2	10	Unstable
100^2	15	Unstable
100^2	20	Unstable
150^2	1	Stable
150^2	5	Unstable
150^2	10	Unstable
150^2	15	Unstable
150^2	20	Unstable
200^2	1	Stable
200^2	5	Unstable
200^2	10	Unstable
200^2	15	Unstable
200^2	20	Unstable
250^2	1	Stable
250^2	5	Unstable
250^2	10	Unstable
250^2	15	Unstable
250^2	20	Unstable
300^2	1	Stable
300^2	5	Unstable
300^2	10	Unstable
300^2	15	Unstable
300^2	20	Unstable

Table C.4: Stability results for explicit Euler (flag)

C.2 Verlet

C.2.1 Sheet Data

Rows	Columns	Time Step	Internal force (ms)	External force (ms)	Integrate (ms)	Update (ms)	Render (ms)	FPS	Update count
50	50	1	0.1292	0.00514372	0.00850142	0.158038	0.0699893	12026	57945
50	50	1	0.141176	0.00458841	0.00830461	0.1696	0.0870659	9532.48	57256
50	50	1	0.139742	0.00520284	0.00860645	0.168434	0.080362	10312.1	57550
50	50	1	0.127906	0.00507032	0.00850687	0.15606	0.0887057	9532.38	57450
50	50	1	0.127712	0.00456654	0.00833728	0.155455	0.0686348	12232.4	57549
50	50	1	0.128189	0.00450527	0.00823626	0.156247	0.0669741	12526.5	58475
50	50	5	0.128398	0.00515085	0.00850705	0.174885	0.0693888	13815.8	11957
50	50	5	0.141781	0.00457966	0.00829447	0.185291	0.0860006	11150	11928
50	50	5	0.140044	0.00519248	0.0086085	0.184119	0.0808033	11868.5	11932
50	50	5	0.127223	0.00507196	0.00853278	0.171354	0.0834408	11510.7	11940
50	50	5	0.1273	0.00457044	0.00834477	0.171302	0.0679456	14138.1	11942
50	50	5	0.127956	0.00451481	0.00824743	0.173914	0.0661126	14496.4	11946
50	50	10	0.128282	0.0051481	0.00850746	0.196596	0.0694825	14028.7	5998
50	50	10	0.141102	0.00457583	0.00830379	0.203754	0.085207	11446.9	5989
50	50	10	0.139184	0.00518831	0.00859851	0.201983	0.0809445	12053.1	5988
50	50	10	0.127386	0.0050824	0.00853751	0.190644	0.0842738	11601.2	5987
50	50	10	0.127139	0.00457993	0.00835023	0.191212	0.0679559	14361.6	5995
50	50	10	0.127692	0.00451017	0.00824961	0.195	0.0663485	14710.6	5994
50	50	15	0.128263	0.00514679	0.00851718	0.218667	0.0694025	14123.8	4003
50	50	15	0.141721	0.00457741	0.00830542	0.223894	0.0844429	11618.2	3999
50	50	15	0.140614	0.00520937	0.00859768	0.22251	0.080732	12139	3999
50	50	15	0.127743	0.00508693	0.00855462	0.211046	0.0836584	11738.7	3998

50	50	15	0.127173	0.00457687	0.008358	0.21127	0.0678336	14462.3	4001
50	50	15	0.127686	0.00451245	0.00825431	0.216504	0.0663661	14781	4001
50	50	20	0.128454	0.00516354	0.00852637	0.240549	0.0694237	14156.5	3004
50	50	20	0.141326	0.00458932	0.00830365	0.239668	0.0879786	11177	3001
50	50	20	0.140597	0.00518829	0.00862252	0.240383	0.0820249	11997.7	3001
50	50	20	0.127907	0.00509295	0.00857623	0.229511	0.0843226	11681.3	3001
50	50	20	0.127152	0.00457609	0.00835984	0.231802	0.0678947	14487.8	3002
50	50	20	0.127837	0.0045219	0.00826082	0.237854	0.0664448	14801.8	3002
100	100	1	0.511579	0.0249554	0.0363878	0.6075	0.0703838	5807.03	58323
100	100	1	0.518305	0.0251286	0.0367537	0.615383	0.0884511	4717.1	56899
100	100	1	0.518781	0.0251611	0.0368163	0.617203	0.0831351	4893.72	57881
100	100	1	0.507824	0.0249645	0.0365096	0.603702	0.0968606	4547.27	56399
100	100	1	0.507234	0.0248978	0.0363583	0.60257	0.0690263	5974.88	58846
100	100	1	0.504087	0.0251196	0.0366892	0.600576	0.0673333	6163.13	58385
100	100	5	0.510469	0.024993	0.0364923	0.625192	0.069522	12501.5	11939
100	100	5	0.516861	0.0251417	0.0367707	0.629165	0.0877022	9885.37	11920
100	100	5	0.518653	0.025067	0.0365479	0.630508	0.0835573	10339.3	11926
100	100	5	0.506282	0.0249708	0.0365123	0.616707	0.0940467	9207.67	11904
100	100	5	0.506336	0.0249237	0.0364528	0.619096	0.0682622	12688.8	11940
100	100	5	0.503268	0.0251388	0.0367469	0.618128	0.0664842	13104.1	11947
100	100	10	0.509953	0.0250254	0.0365274	0.646762	0.0693812	13400.4	5996
100	100	10	0.516846	0.0251574	0.0367736	0.648067	0.086672	10733.3	5987
100	100	10	0.517117	0.025079	0.0368154	0.647971	0.0828408	11227.4	5987
100	100	10	0.506314	0.0249844	0.0365808	0.634072	0.0940045	9911.49	5981
100	100	10	0.506012	0.0249245	0.0364694	0.638996	0.0676062	13758.5	5994
100	100	10	0.503335	0.0251435	0.0367838	0.640193	0.0661642	14071.9	5994

100	100	15	0.464603	0.0250647	0.0366154	0.623143	0.0708815	13447.4	3995
100	100	15	0.474643	0.0251803	0.0368014	0.627239	0.0881917	10814.4	3989
100	100	15	0.476572	0.0250832	0.0368803	0.63539	0.0787872	12095.9	3992
100	100	15	0.465141	0.0250226	0.0366585	0.618011	0.0851339	11211.3	3989
100	100	15	0.463933	0.0249431	0.0365274	0.617628	0.0691007	13809.9	3992
100	100	15	0.461324	0.0251616	0.0367754	0.619761	0.067776	14074.5	3992
100	100	20	0.464782	0.0251331	0.036701	0.645431	0.0705604	13642.5	3002
100	100	20	0.474422	0.0251737	0.0368032	0.644272	0.0883516	10907.4	2998
100	100	20	0.474753	0.0252325	0.0370347	0.655087	0.0815577	11801.8	2999
100	100	20	0.465378	0.0250617	0.0366798	0.637766	0.0846656	11387.2	2998
100	100	20	0.463827	0.0249346	0.0365334	0.636855	0.0688881	13986	3000
100	100	20	0.461989	0.0251834	0.036802	0.642281	0.0674199	14291.6	3000
150	150	1	1.14689	0.0565795	0.0822402	1.35616	0.0838683	831.917	41662
150	150	1	1.15516	0.0570664	0.0829819	1.37229	0.0886234	769.083	41066
150	150	1	1.15754	0.0572862	0.0835165	1.37518	0.0826341	849.767	41154
150	150	1	1.14005	0.0566549	0.0823202	1.34923	0.135514	815.467	40407
150	150	1	1.13892	0.0564982	0.0820394	1.34782	0.0819604	870.317	41961
150	150	1	1.13204	0.0570454	0.0828546	1.34325	0.0811171	769.25	42119
150	150	5	1.14746	0.0568218	0.0823503	1.37525	0.069329	10316.8	11944
150	150	5	1.15139	0.0571201	0.083022	1.38426	0.0880775	8129.22	11930
150	150	5	1.1506	0.0572882	0.083493	1.38403	0.0839752	8481.18	11931
150	150	5	1.14153	0.0567115	0.0824293	1.36672	0.0865225	8287.47	11907
150	150	5	1.14015	0.0565765	0.082167	1.36557	0.0682649	10488.4	11940
150	150	5	1.13362	0.0570586	0.0829382	1.36248	0.0665155	10857.7	11935
150	150	10	1.06586	0.0570732	0.0824805	1.31818	0.0706959	12173.8	5972
150	150	10	1.07597	0.0572753	0.0831871	1.33098	0.0901707	9545.22	5961

150	150	10	1.07359	0.0574186	0.0836126	1.32836	0.0851829	10145.3	5964
150	150	10	1.06446	0.0568446	0.0825193	1.30732	0.0965085	8986.15	5959
150	150	10	1.06319	0.0566281	0.0823029	1.31247	0.0689893	12544.5	5969
150	150	10	1.05793	0.0570534	0.0829515	1.31088	0.0678314	12728.8	5969
150	150	15	1.06668	0.0571794	0.0825728	1.3409	0.0701581	12909	3993
150	150	15	1.07836	0.0572749	0.0832212	1.35068	0.0896625	10110.2	3987
150	150	15	1.07468	0.0572457	0.0835301	1.34589	0.0846257	10708.3	3989
150	150	15	1.06499	0.0567142	0.0824397	1.3247	0.0957884	9485.6	3985
150	150	15	1.06468	0.0568322	0.0824695	1.33521	0.0686458	13206.3	3991
150	150	15	1.05853	0.0570687	0.0829865	1.33342	0.0672801	13466.7	3991
150	150	20	1.06807	0.057407	0.0826895	1.36434	0.0701279	13212.2	2998
150	150	20	1.08063	0.0580446	0.0836151	1.37303	0.089913	10314.8	2994
150	150	20	1.07659	0.057245	0.0835468	1.36679	0.084509	10980	2994
150	150	20	1.06647	0.0574592	0.0828579	1.34556	0.0960142	9679.67	2992
150	150	20	1.06502	0.0566934	0.0823761	1.35338	0.0685551	13530.5	2996
150	150	20	1.05946	0.0571247	0.0829926	1.35706	0.0672448	13801.8	2996
200	200	1	2.05795	0.104033	0.147102	2.43893	0.118731	598.066	23458
200	200	1	2.06945	0.105885	0.148814	2.46747	0.125262	483.15	23140
200	200	1	2.06136	0.104652	0.149087	2.44493	0.112362	548	23461
200	200	1	2.04594	0.103332	0.147472	2.42569	0.203464	492.3	22820
200	200	1	2.04287	0.103011	0.146743	2.421	0.112939	587.1	23677
200	200	1	2.02934	0.103033	0.147835	2.40819	0.115846	479.983	23770
200	200	5	2.06091	0.106334	0.147903	2.46321	0.0696379	7289.5	11938
200	200	5	2.08351	0.107077	0.14917	2.49753	0.089584	5546.07	11914
200	200	5	2.10076	0.105934	0.149436	2.50074	0.0829773	5959.9	11909
200	200	5	2.04644	0.104262	0.147492	2.43999	0.0906102	5611.5	11898

200	200	5	2.05013	0.105225	0.147365	2.44702	0.0683153	7374.95	11936
200	200	5	2.03751	0.105708	0.148913	2.43691	0.0670605	7598.17	11946
200	200	10	1.93711	0.10923	0.149434	2.37324	0.0708498	10704.5	5969
200	200	10	1.95585	0.108417	0.149635	2.39824	0.0910348	8251.45	5960
200	200	10	2.04515	0.108989	0.150298	2.47946	0.0850481	8814.92	5963
200	200	10	1.92968	0.106451	0.148565	2.35077	0.0926753	8167.34	5958
200	200	10	1.93182	0.106915	0.148265	2.36089	0.0707138	10768.1	5962
200	200	10	1.92576	0.104407	0.148268	2.34831	0.070907	10703.5	5961
200	200	15	1.94056	0.110725	0.1502	2.39899	0.0777385	10755.4	3985
200	200	15	1.97114	0.109984	0.150607	2.43703	0.0907105	9201.3	3983
200	200	15	2.11456	0.105388	0.149305	2.56362	0.0852468	9679.15	3985
200	200	15	1.93048	0.106957	0.148711	2.37231	0.0922235	9101.73	3982
200	200	15	1.93752	0.109224	0.149515	2.38735	0.0776157	10792.2	3983
200	200	15	1.93138	0.110885	0.151921	2.38908	0.0778951	10749.8	3982
200	200	20	1.94996	0.112987	0.151605	2.43195	0.0811162	10785.2	2995
200	200	20	1.98992	0.112647	0.15193	2.4809	0.0906988	9613.6	2994
200	200	20	2.17486	0.111838	0.15299	2.66311	0.0851165	10130	2995
200	200	20	1.93519	0.107917	0.149148	2.39571	0.0926047	9469.25	2992
200	200	20	1.94025	0.111272	0.150273	2.41033	0.0812196	10787.3	2994
200	200	20	1.93669	0.104732	0.148779	2.39462	0.0815048	10763.6	2993
250	250	1	3.24444	0.183755	0.245967	3.91972	0.172605	273.817	14661
250	250	1	3.2648	0.190932	0.253452	3.96171	0.182132	337.8	14479
250	250	1	3.26322	0.185945	0.248296	3.94293	0.170276	396.705	14587
250	250	1	3.24748	0.186987	0.243262	3.91598	0.258285	323.317	14374
250	250	1	3.24931	0.19278	0.244606	3.92436	0.179462	395.617	14620
250	250	1	3.22365	0.182864	0.243261	3.88723	0.174011	298.05	14773

250	250	5	3.35564	0.195174	0.253748	4.07154	0.0826414	2315.43	11899
250	250	5	3.46299	0.198196	0.257336	4.18269	0.0933146	1798.17	11907
250	250	5	3.4604	0.192676	0.257915	4.16641	0.0863167	2003.73	11903
250	250	5	3.3512	0.190158	0.249597	4.04706	0.0991172	1972.33	11893
250	250	5	3.34782	0.195054	0.253988	4.05793	0.0851445	2302.5	11838
250	250	5	3.3298	0.189538	0.249261	4.02574	0.090645	2345.08	11712
250	250	10	3.07781	0.204005	0.261848	3.84527	0.0839742	7301.78	5952
250	250	10	3.40969	0.201107	0.260977	4.17353	0.0879321	6549.42	5956
250	250	10	3.4238	0.226129	0.285295	4.23245	0.0824318	6995.93	5958
250	250	10	3.0653	0.194202	0.252037	3.79169	0.0903605	6800.77	5955
250	250	10	3.06766	0.19678	0.256724	3.81467	0.0843782	7226.23	5951
250	250	10	3.06479	0.211468	0.252071	3.80706	0.0844546	7295.85	5949
250	250	15	3.07647	0.200062	0.260171	3.8513	0.100823	7356.17	3975
250	250	15	3.4327	0.229514	0.296276	4.3208	0.0969827	7326.75	3981
250	250	15	3.44039	0.225519	0.295668	4.32035	0.0969451	7326.68	3977
250	250	15	3.08028	0.198893	0.258371	3.84939	0.101153	7312.18	3975
250	250	15	3.07633	0.203865	0.260237	3.85348	0.100863	7359.05	3974
250	250	15	3.05573	0.201737	0.259624	3.82984	0.100806	7374.87	3978
250	250	20	3.08101	0.201977	0.261936	3.87726	0.109312	7356.47	2988
250	250	20	3.44324	0.235111	0.303984	4.36786	0.10639	7327.72	2986
250	250	20	3.46374	0.238511	0.283725	4.33169	0.106643	7333.4	2986
250	250	20	3.08181	0.201807	0.260989	3.87548	0.109478	7350.1	2988
250	250	20	3.0843	0.202662	0.251648	3.84631	0.109338	7372.82	2985
250	250	20	3.06746	0.206669	0.263283	3.86608	0.109072	7383.03	2985
300	300	1	5.09265	0.353192	0.428479	6.35434	0.200684	156.133	9153
300	300	1	5.09864	0.364151	0.439707	6.39023	0.208525	219.133	9092

300	300	1	5.09637	0.36176	0.437645	6.37833	0.196724	244.1	9126
300	300	1	5.08172	0.35265	0.426711	6.3363	0.284581	189.983	9062
300	300	1	5.07871	0.351981	0.427157	6.33249	0.20104	227.367	9183
300	300	1	5.02877	0.351019	0.423546	6.27495	0.200255	156.467	9266
300	300	5	4.73373	0.356865	0.434178	6.01245	0.241927	160.717	9591
300	300	5	4.74014	0.357777	0.435284	6.01972	0.252388	159.8	9564
300	300	5	4.74171	0.385936	0.43109	6.03597	0.240076	159.617	9558
300	300	5	4.71955	0.348101	0.42354	5.96383	0.323856	159.117	9541
300	300	5	4.72198	0.354781	0.430714	5.99048	0.242373	160.85	9624
300	300	5	4.68336	0.347474	0.422354	5.92614	0.248495	162.917	9714
300	300	10	4.75246	0.379215	0.457054	6.13447	0.0746482	5213.13	5949
300	300	10	5.05153	0.36537	0.441689	6.36923	0.0887881	4097.02	5942
300	300	10	5.10737	0.394411	0.4406	6.45013	0.0835249	4243.8	5944
300	300	10	4.73219	0.359368	0.431489	6.02645	0.0931049	4289.85	5939
300	300	10	4.73359	0.373336	0.453137	6.11408	0.074606	5209.32	5938
300	300	10	4.70156	0.377419	0.453479	6.08589	0.0752202	5270.07	5929
300	300	15	4.75364	0.382591	0.460101	6.15916	0.111365	5282.27	3976
300	300	15	5.14741	0.424045	0.503818	6.69568	0.104548	5255.17	3976
300	300	15	5.15011	0.444658	0.496008	6.70024	0.104626	5297.62	3974
300	300	15	4.73616	0.377929	0.45357	6.12583	0.111873	5266.52	3973
300	300	15	4.73573	0.380509	0.459316	6.14126	0.111692	5254.93	3975
300	300	15	4.69683	0.380216	0.45353	6.08808	0.112382	5278.93	3979
300	300	20	4.74923	0.382238	0.460354	6.17093	0.130532	5295.38	2984
300	300	20	5.1491	0.422996	0.505943	6.70314	0.125376	5303.73	2980
300	300	20	5.15294	0.445255	0.495819	6.71457	0.125314	5303.93	2980
300	300	20	4.73866	0.380501	0.456523	6.14327	0.130796	5297.58	2979

| 300 | 300 | 20 | 4.73891 | 0.379951 | 0.45874 | 6.14742 | 0.130888 | 5292.3 | 2982 |
| 300 | 300 | 20 | 4.70213 | 0.376521 | 0.451617 | 6.0929 | 0.131443 | 5290.63 | 2983 |

Table C.5: Raw data for Verlet (sheet)
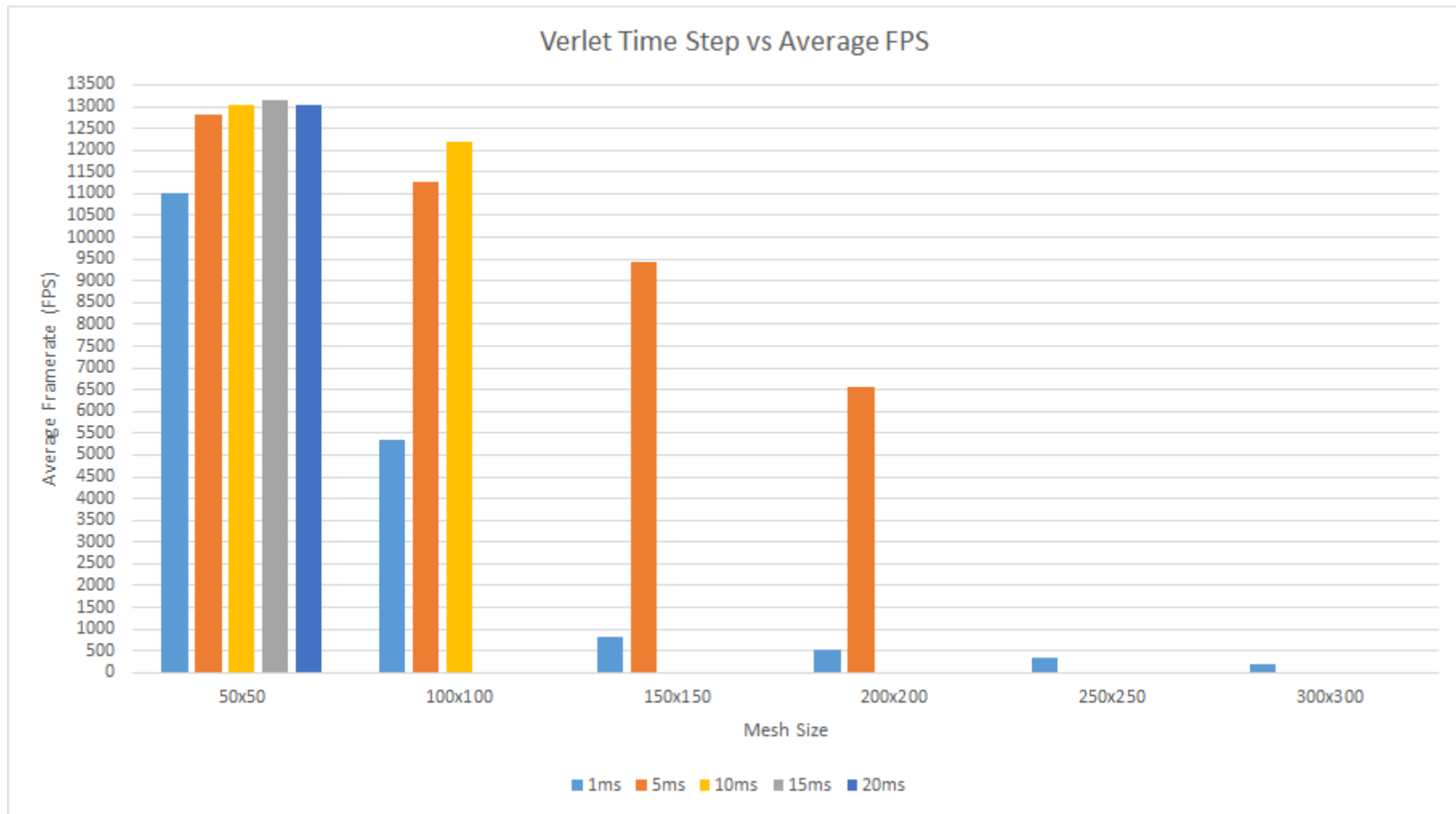
Figure C.27: Verlet time step against average FPS (sheet)
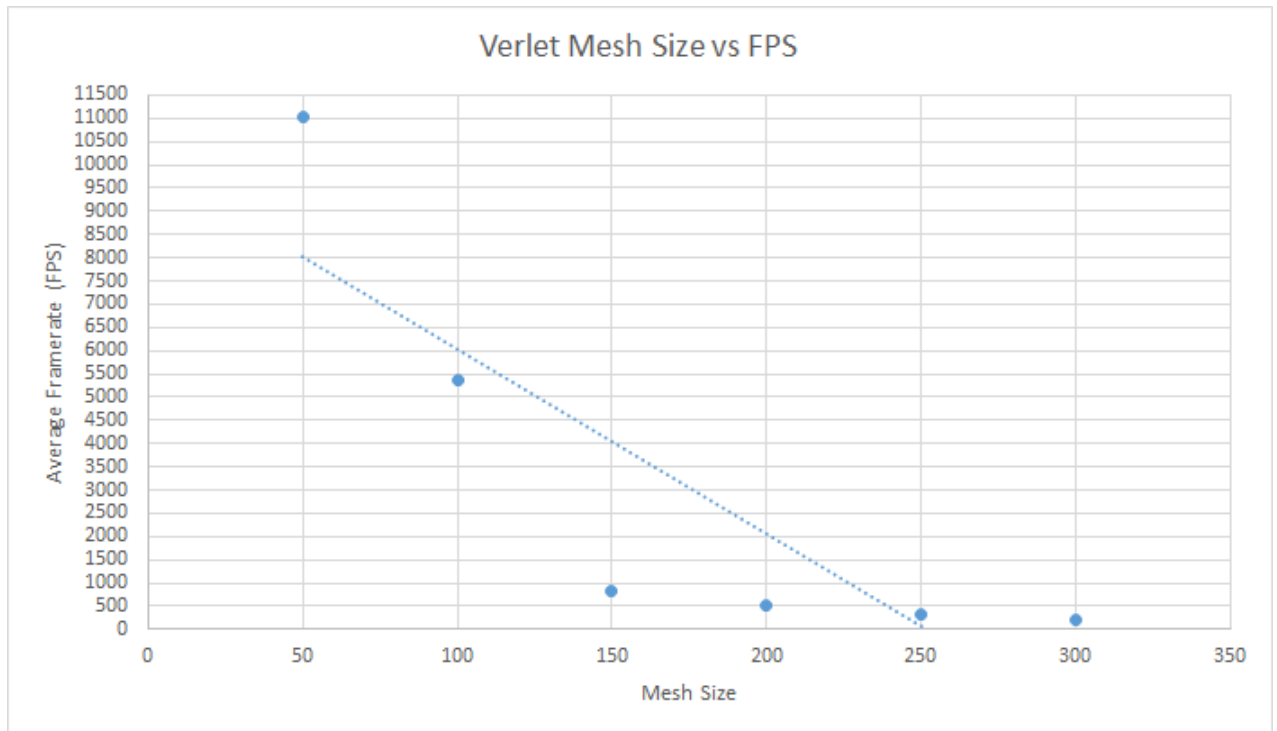
Figure C.28: Verlet mesh size against average FPS (sheet)
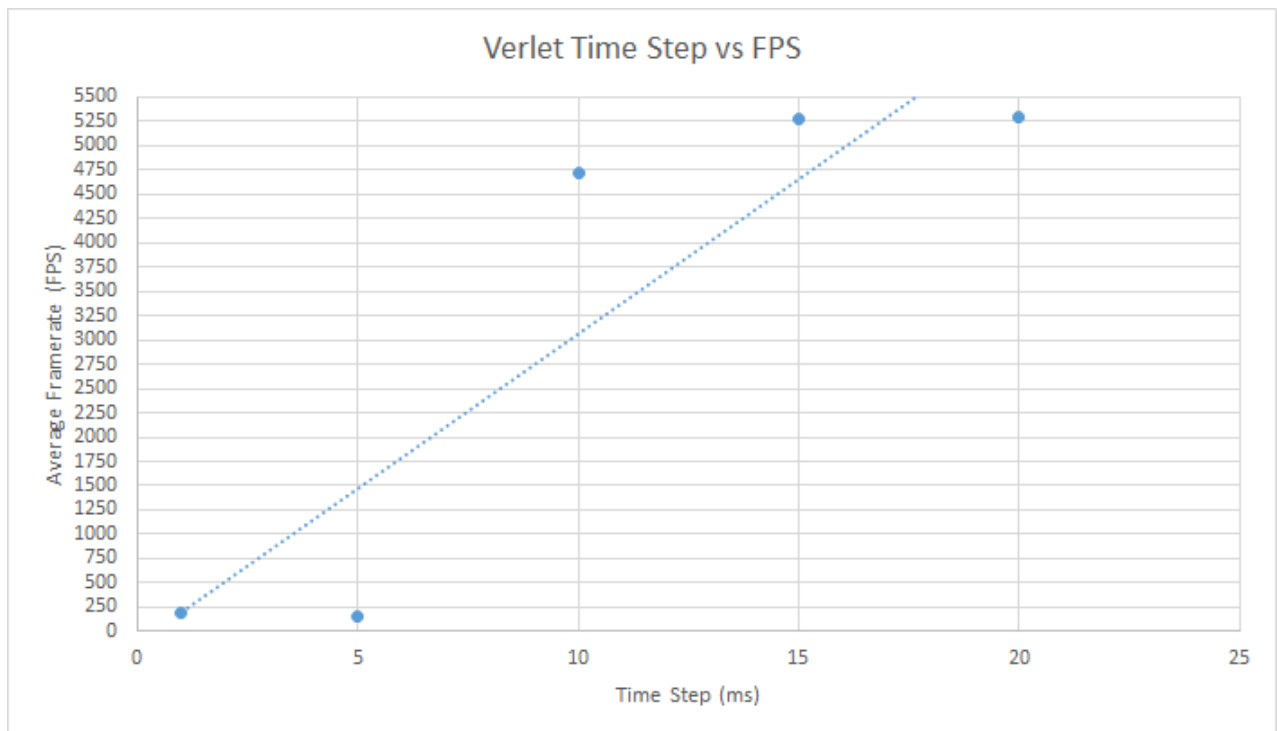
Figure C.29: Verlet time step against average FPS for a 300^2 mesh (sheet)
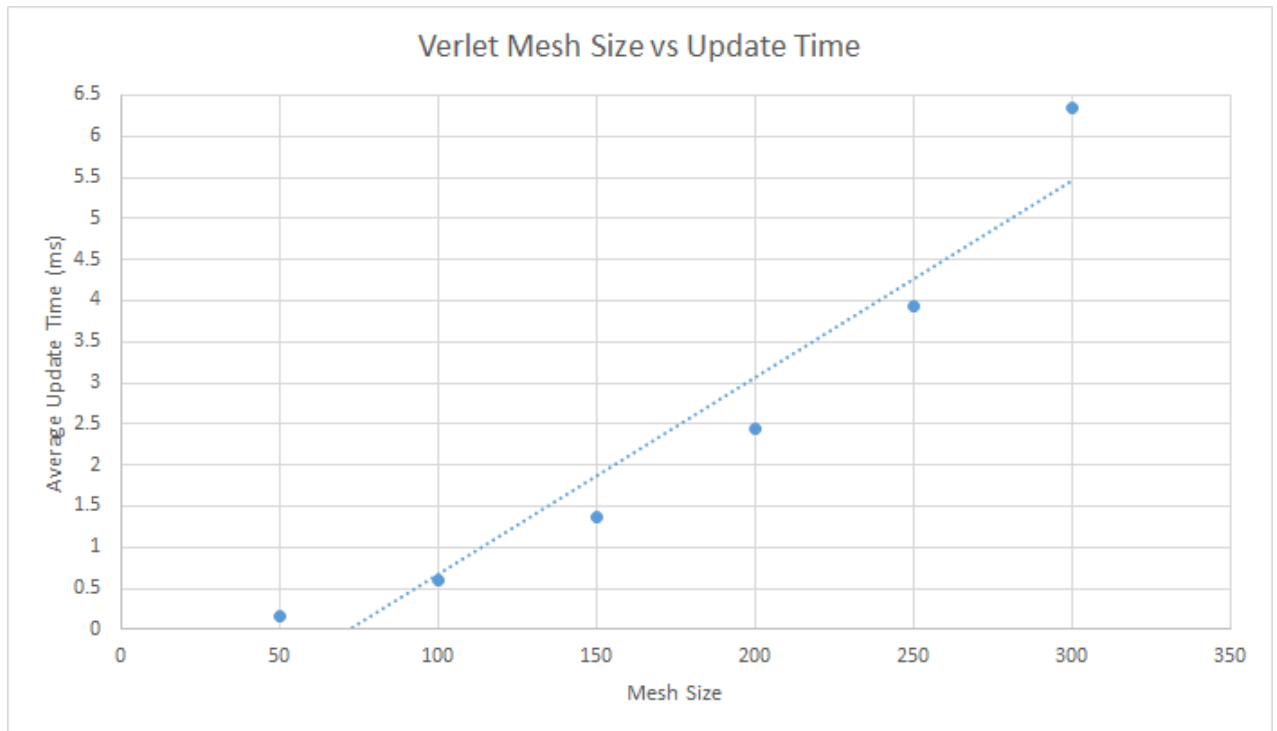
Figure C.30: Verlet mesh size against average update time (sheet)
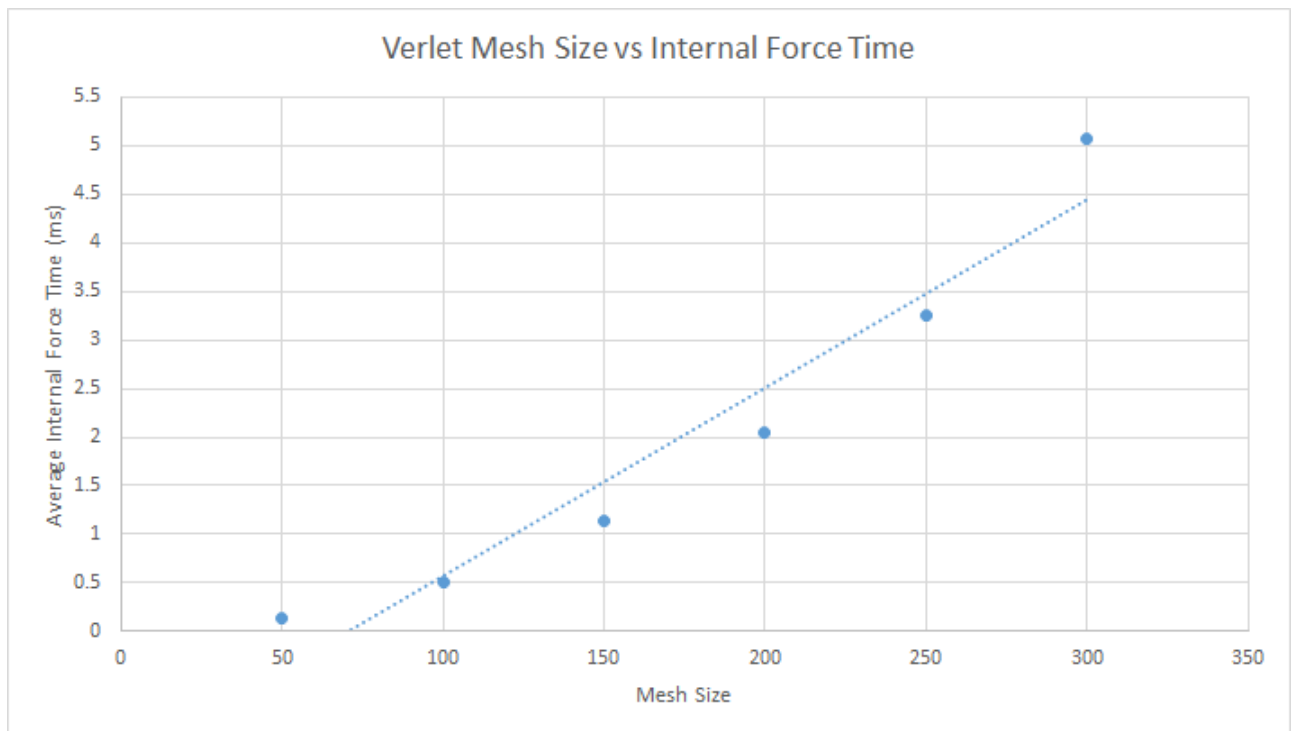
Figure C.31: Verlet mesh size against average internal force time (sheet)
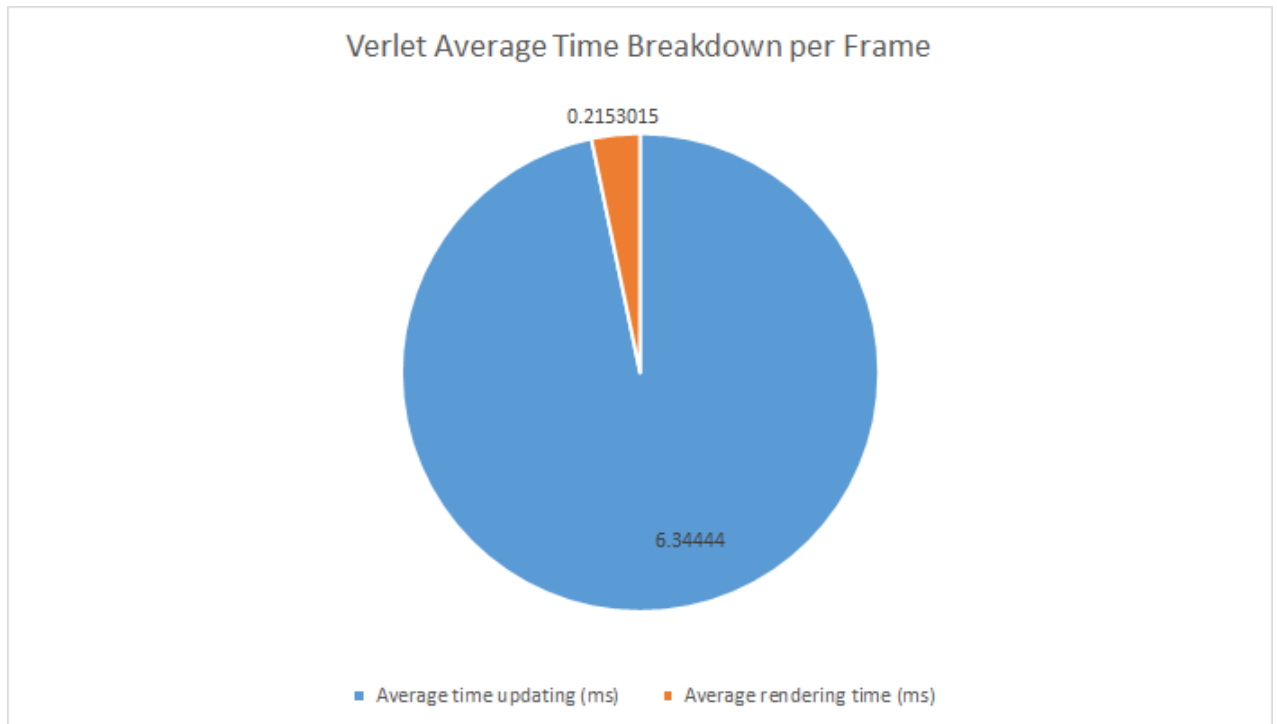
Figure C.32: Verlet frame time breakdown (sheet)
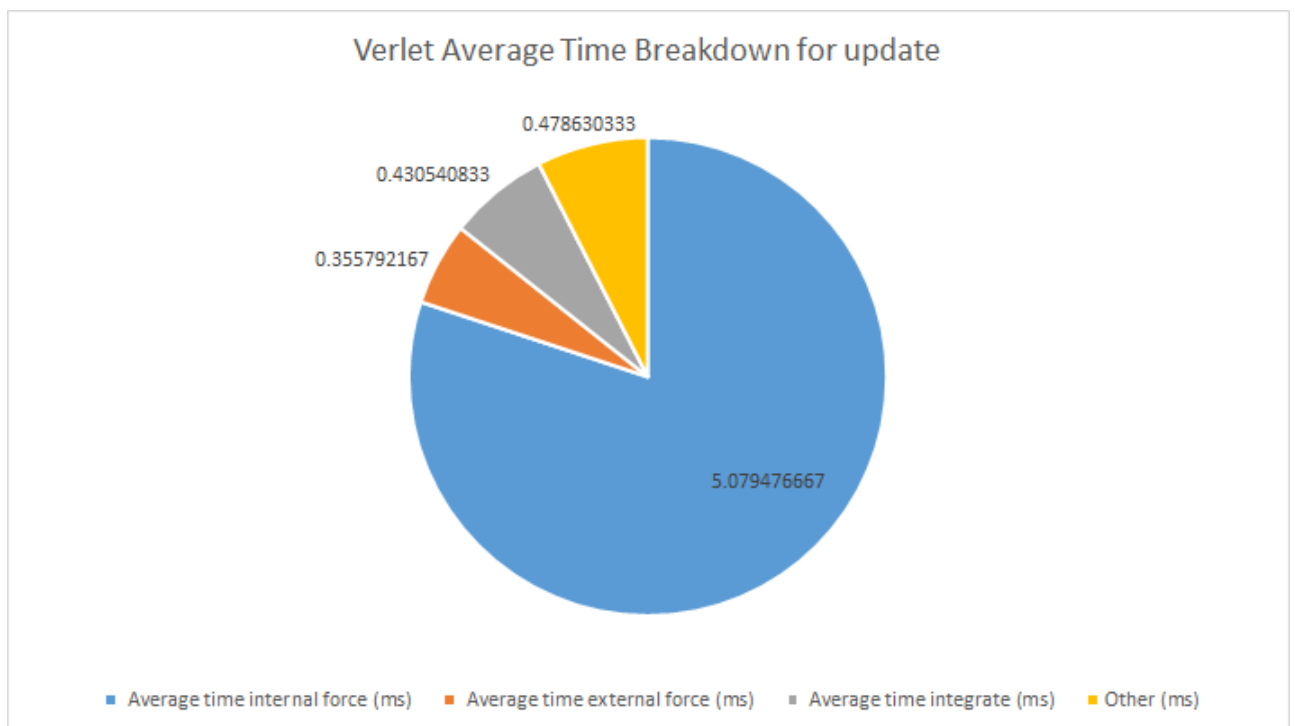
Figure C.33: Verlet update time breakdown (sheet)
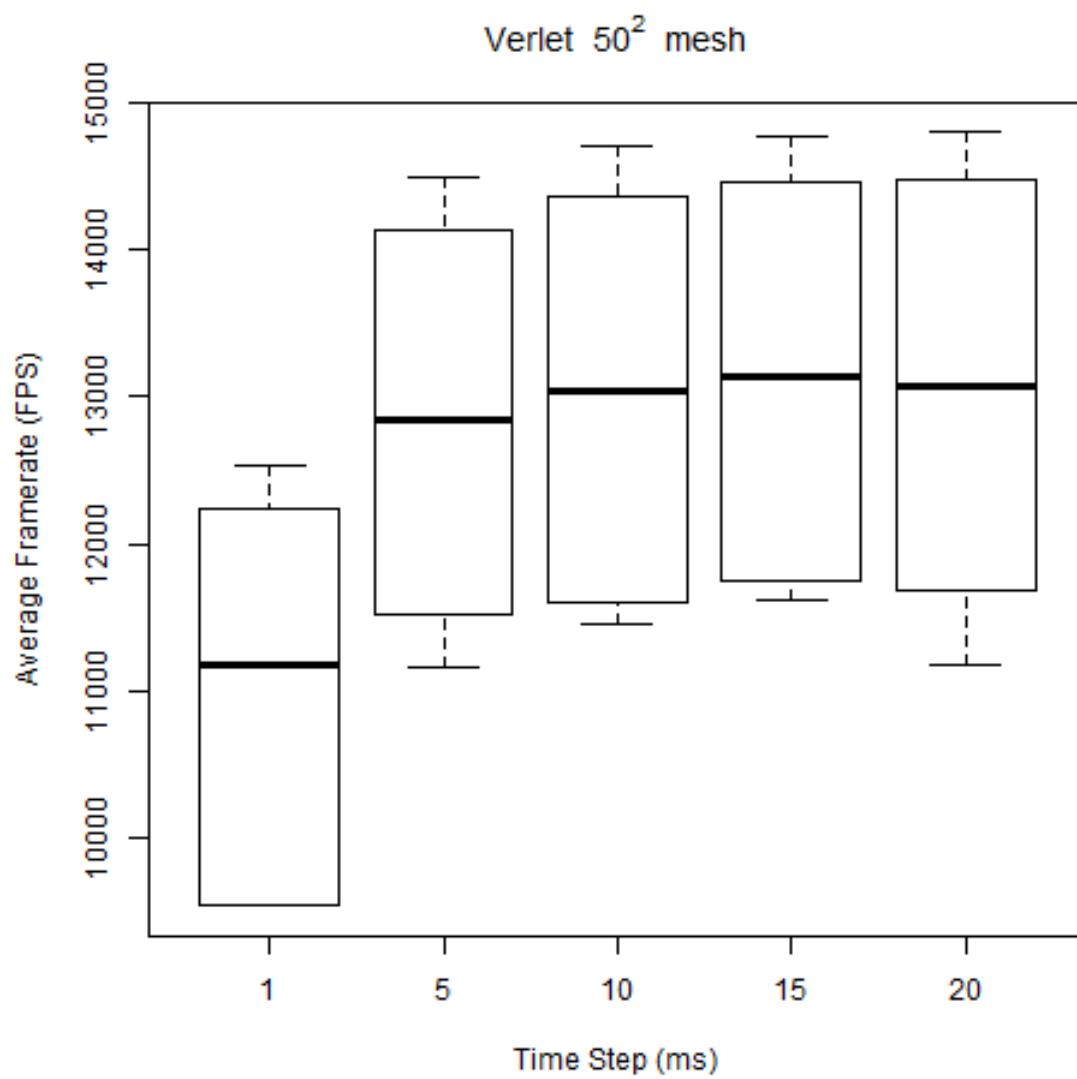
Figure C.34: Verlet box plot for a 50^2 mesh (sheet)
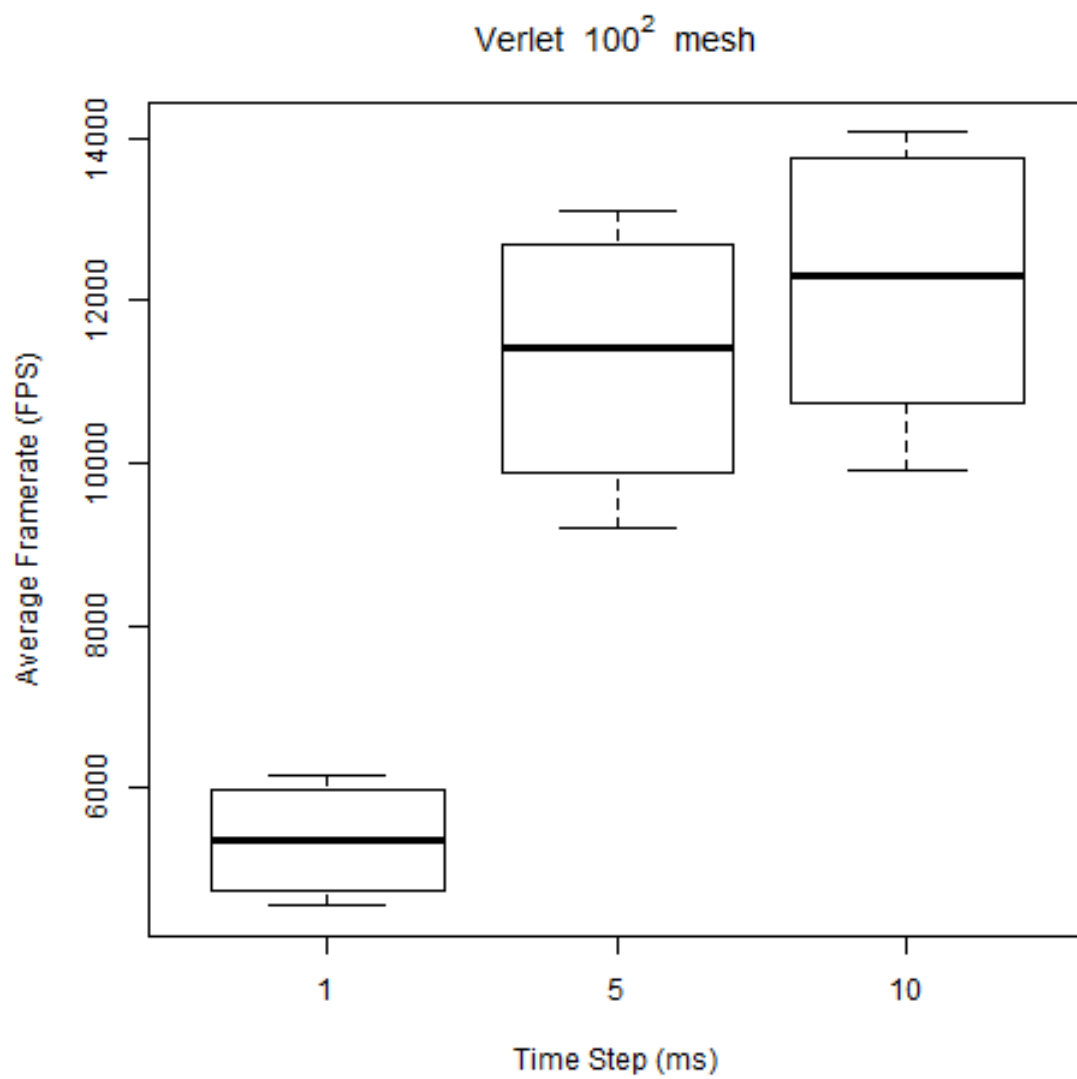
Figure C.35: Verlet box plot for a 100^2 mesh (sheet)
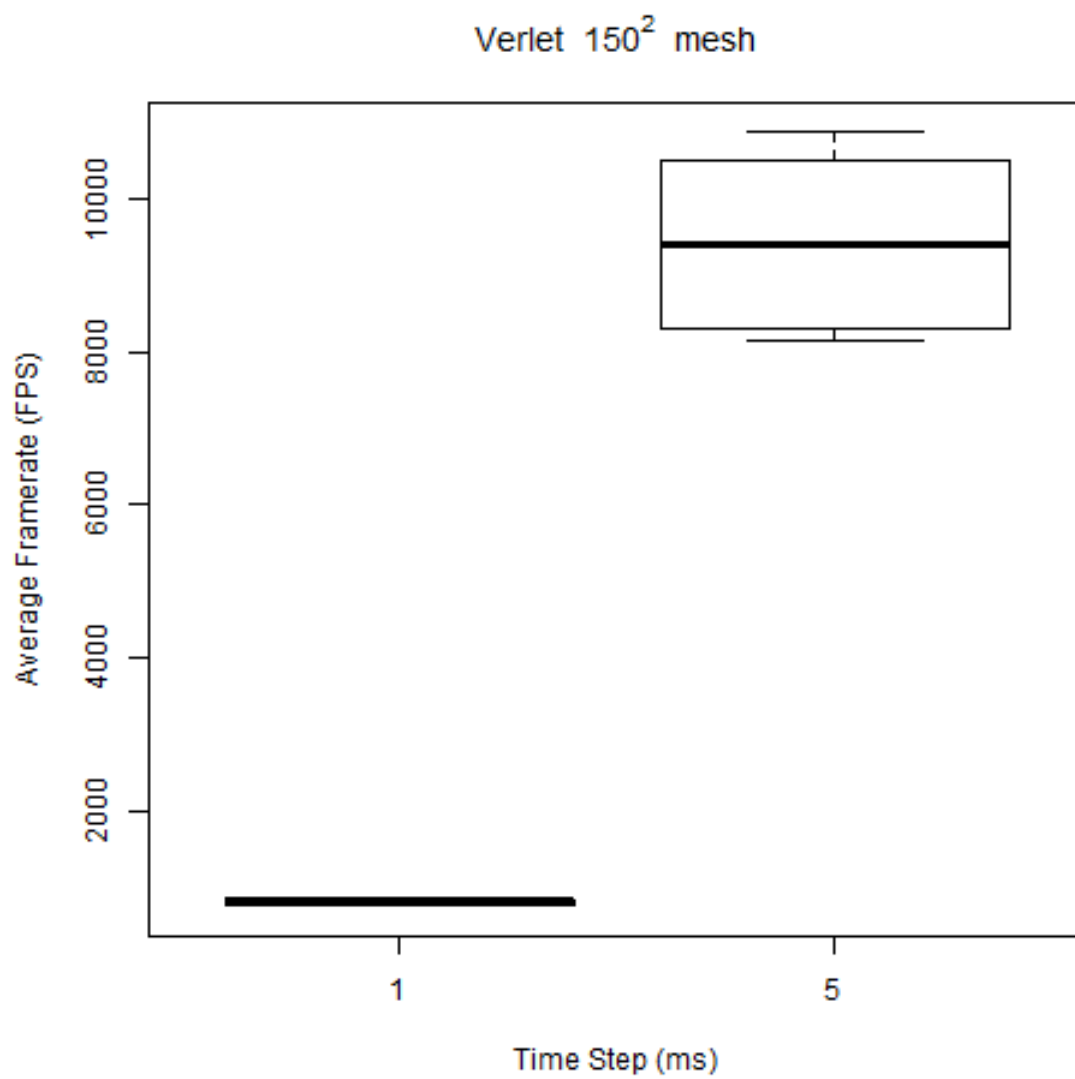
Figure C.36: Verlet box plot for a 150^2 mesh (sheet)
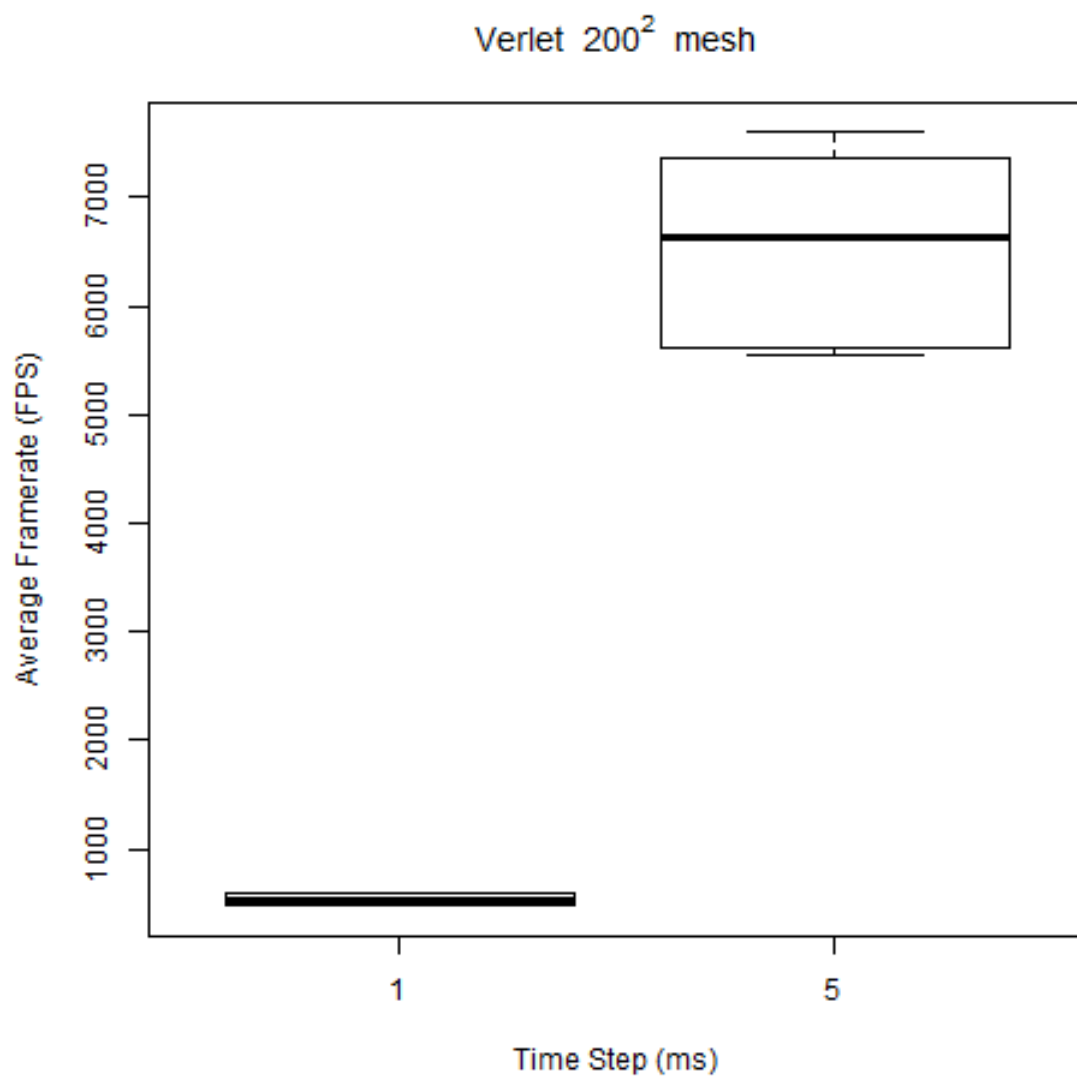
Figure C.37: Verlet box plot for a 200^2 mesh (sheet)
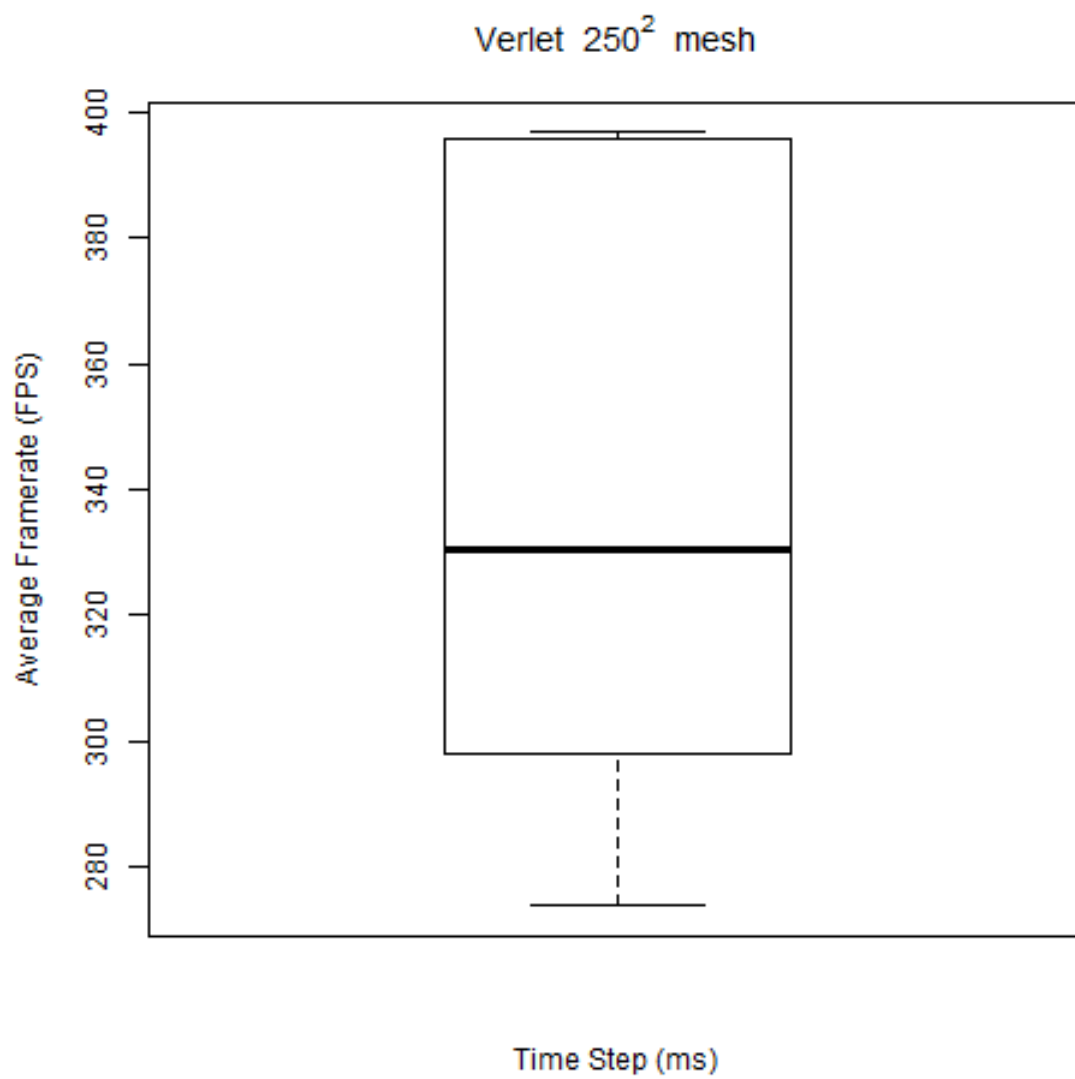
Figure C.38: Verlet box plot for a 250^2 mesh (sheet)
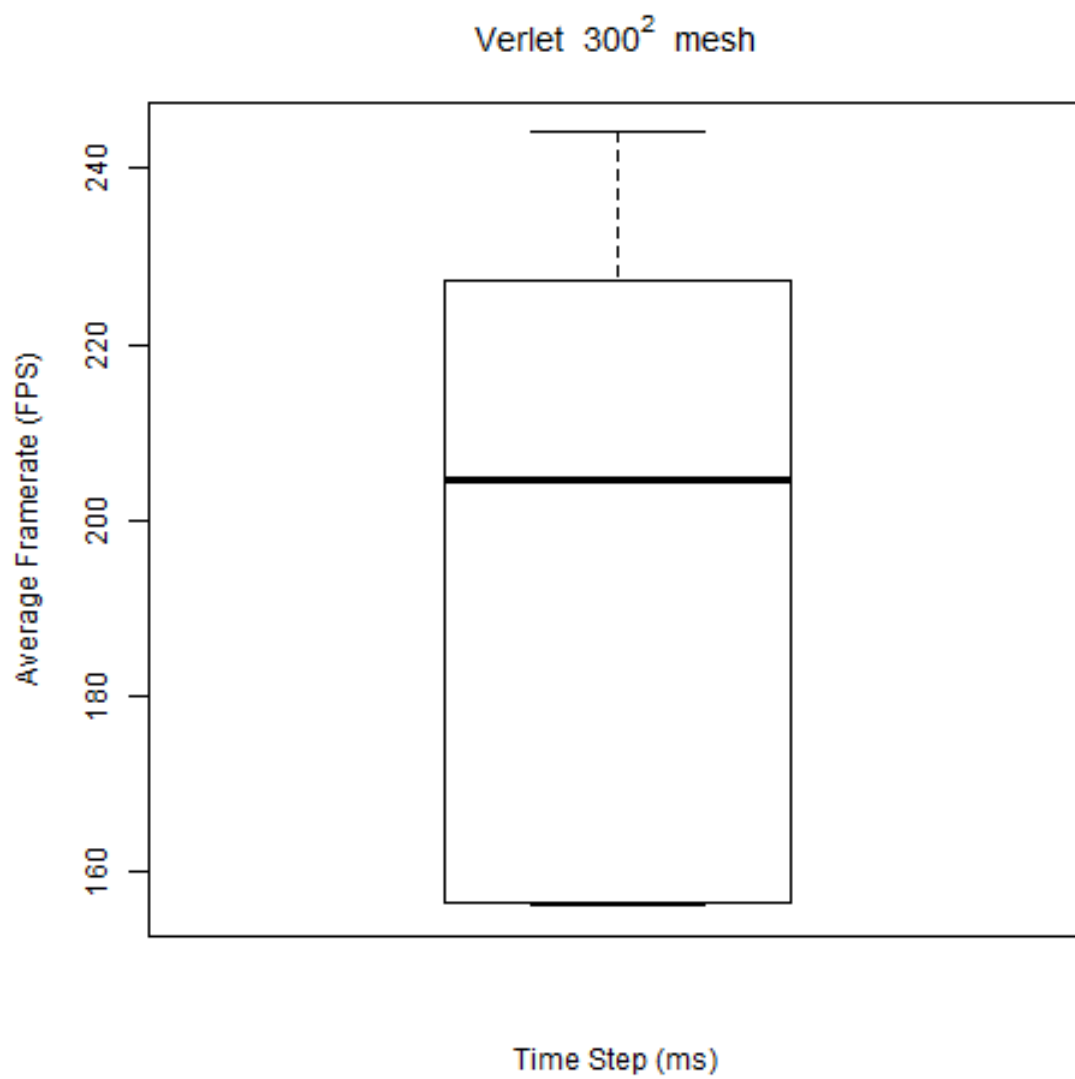
Figure C.39: Verlet box plot for a 300^2 mesh (sheet)

Mesh Size	Time Step (ms)	Stable/Unstable
50^2	1	Stable
50^2	5	Stable
50^2	10	Stable
50^2	15	Stable
50^2	20	Stable
100^2	1	Stable
100^2	5	Stable
100^2	10	Stable
100^2	15	Unstable
100^2	20	Unstable
150^2	1	Stable
150^2	5	Stable
150^2	10	Unstable
150^2	15	Unstable
150^2	20	Unstable
200^2	1	Stable
200^2	5	Stable
200^2	10	Unstable
200^2	15	Unstable
200^2	20	Unstable
250^2	1	Stable
250^2	5	Unstable
250^2	10	Unstable
250^2	15	Unstable
250^2	20	Unstable
300^2	1	Stable
300^2	5	Unstable
300^2	10	Unstable
300^2	15	Unstable
300^2	20	Unstable

Table C.6: Stability results for Verlet (sheet)

C.2.2 Flag Data

Rows	Columns	Time Step	Internal force (ms)	External force (ms)	Integrate (ms)	Update (ms)	Render (ms)	FPS	Update count
50	50	1	0.125823	0.0629402	0.00831367	0.212048	0.0698548	11362.8	57409
50	50	1	0.138965	0.0629834	0.00815317	0.225349	0.0863157	9048.38	57716
50	50	1	0.137033	0.0632484	0.00843756	0.223412	0.0815499	9614.3	57345
50	50	1	0.124867	0.0628431	0.00831579	0.210503	0.0862688	9239.28	57070
50	50	1	0.124489	0.0632578	0.00817181	0.210477	0.0687054	11539.6	58114
50	50	1	0.125031	0.0631096	0.00808651	0.211387	0.0668664	11840.5	58419
50	50	5	0.136156	0.0629894	0.00833978	0.240154	0.0695832	13617.7	11961
50	50	5	0.149571	0.0629321	0.00815858	0.251194	0.0864675	10941.4	11927
50	50	5	0.147825	0.0631911	0.00842544	0.249259	0.0822633	11512.8	11934
50	50	5	0.135177	0.062808	0.0083487	0.236583	0.0835442	11366.7	11924
50	50	5	0.135741	0.0633714	0.00822014	0.238553	0.0693647	13668.7	11939
50	50	5	0.135579	0.0631521	0.00809183	0.239642	0.0663878	14280.3	11950
50	50	10	0.151568	0.0629381	0.00833252	0.277367	0.0694576	13926.1	5997
50	50	10	0.165309	0.0629179	0.00815723	0.285248	0.0875191	11049	5988
50	50	10	0.163451	0.0632042	0.00843611	0.28343	0.0826227	11721.5	5989
50	50	10	0.151259	0.0629717	0.00838911	0.272411	0.0827372	11698.7	5988
50	50	10	0.150861	0.0632936	0.00819377	0.273361	0.0678588	14266.6	5995
50	50	10	0.151237	0.0631451	0.00809431	0.276763	0.0663634	14586.3	5995
50	50	15	0.15771	0.0630047	0.00833457	0.305429	0.0695157	14020	4004
50	50	15	0.172015	0.0629137	0.00816019	0.310209	0.0882633	11060.2	3998
50	50	15	0.170295	0.0630984	0.00840892	0.3102	0.0798043	12229.5	3999
50	50	15	0.157516	0.0629712	0.00838938	0.297615	0.083741	11659.4	3998
50	50	15	0.156904	0.0632361	0.00819757	0.299642	0.0678748	14369.7	4002
50	50	15	0.157421	0.0631227	0.00809666	0.304103	0.0662637	14721.3	4001

50	50	20	0.162517	0.0629393	0.00833937	0.331992	0.0695646	14064.1	3004
50	50	20	0.176184	0.0629635	0.0081693	0.333614	0.0875799	11183.1	3000
50	50	20	0.174672	0.0632058	0.00845386	0.333757	0.0798889	12258.7	3001
50	50	20	0.16217	0.0629868	0.00836921	0.322135	0.0827943	11829.5	3000
50	50	20	0.161732	0.0632766	0.00819717	0.324592	0.0680233	14392.9	3003
50	50	20	0.162061	0.063129	0.00811093	0.331397	0.0664513	14732.5	3002
100	100	1	0.502131	0.255795	0.0361625	0.827786	0.0715149	2857.37	57670
100	100	1	0.535949	0.256377	0.0364997	0.862635	0.0763774	2089.9	58517
100	100	1	0.519236	0.255603	0.0362794	0.845145	0.0798864	2394.45	57530
100	100	1	0.500373	0.255984	0.0362957	0.825936	0.133195	1634.6	57027
100	100	1	0.499209	0.255618	0.0361869	0.824315	0.0703461	2903.42	58028
100	100	1	0.497043	0.255703	0.0365062	0.822796	0.0685041	2901.25	58428
100	100	5	0.551322	0.256113	0.0362521	0.895754	0.0695155	11769.8	11943
100	100	5	0.560028	0.256513	0.0365546	0.902362	0.0884473	9250.92	11915
100	100	5	0.562238	0.255741	0.0365442	0.904092	0.0835818	9792.28	11913
100	100	5	0.549771	0.256269	0.0363579	0.890402	0.0940864	8730.3	11907
100	100	5	0.549145	0.255798	0.0362437	0.891204	0.0679587	12060.7	11944
100	100	5	0.546138	0.256066	0.0365251	0.890462	0.0665117	12321.5	11944
100	100	10	0.622115	0.256152	0.0363034	0.988227	0.069378	12930.2	5997
100	100	10	0.631208	0.256662	0.0365861	0.992498	0.0870982	10307.8	5986
100	100	10	0.63288	0.255814	0.0365499	0.992994	0.0827183	10852.2	5988
100	100	10	0.620757	0.25635	0.0363771	0.978663	0.0929556	9675.6	5984
100	100	10	0.619994	0.2557	0.0362687	0.981513	0.0673581	13329.3	5993
100	100	10	0.61667	0.256006	0.0365461	0.982713	0.0663504	13539.7	5993
100	100	15	0.464074	0.248545	0.0363555	0.844983	0.0698609	13439.5	4002
100	100	15	0.474867	0.249244	0.0366446	0.848807	0.0880145	10675.7	3997

100	100	15	0.474575	0.24886	0.0368687	0.855453	0.0792936	11841.4	3999
100	100	15	0.464812	0.248623	0.0364111	0.839256	0.0839649	11200.4	3998
100	100	15	0.463848	0.248148	0.0363346	0.838624	0.0682536	13768.5	4001
100	100	15	0.461208	0.248436	0.0366089	0.841554	0.0667235	14086.2	4000
100	100	20	0.464618	0.24855	0.0363839	0.867115	0.0699086	13615	3003
100	100	20	0.475033	0.24924	0.0366817	0.866195	0.0890994	10694.4	3000
100	100	20	0.474202	0.248799	0.0368251	0.894723	0.0795543	11942.1	3005
100	100	20	0.465009	0.248499	0.0364781	0.859974	0.0837567	11382.2	3001
100	100	20	0.464049	0.24807	0.0363923	0.859694	0.0682704	13951.7	3002
100	100	20	0.461716	0.248364	0.0366104	0.863545	0.0668833	14242.2	3002
150	150	1	1.12891	0.578596	0.0819239	1.85973	0.0855497	516.017	30842
150	150	1	1.14146	0.580001	0.082723	1.88121	0.0890218	509.083	30451
150	150	1	1.14218	0.579256	0.0831879	1.88197	0.0834118	510.883	30526
150	150	1	1.12493	0.579111	0.0821825	1.85649	0.174331	494.567	29542
150	150	1	1.12319	0.577864	0.0818527	1.85319	0.0819795	519.167	31003
150	150	1	1.11677	0.578788	0.0826097	1.85007	0.0824083	518.433	31046
150	150	5	1.25669	0.578823	0.0819808	2.00343	0.0694078	8656.68	11933
150	150	5	1.2634	0.580216	0.0828059	2.02014	0.088648	6741.22	11913
150	150	5	1.26432	0.579452	0.0832945	2.01665	0.0840147	7117.28	11908
150	150	5	1.25422	0.579099	0.0822429	1.99918	0.0864179	6979.05	11911
150	150	5	1.25459	0.578375	0.0820426	1.99932	0.0682185	8823.02	11936
150	150	5	1.24561	0.579064	0.0827277	1.99359	0.0666496	9033.78	11941
150	150	10	1.06472	0.562114	0.0821633	1.81768	0.0695398	11716.2	5994
150	150	10	1.07535	0.563312	0.0828693	1.83253	0.0882536	9231.33	5987
150	150	10	1.07355	0.562152	0.0833446	1.83008	0.0838303	9721.9	5985
150	150	10	1.06426	0.561865	0.0822241	1.80888	0.0930156	8784.37	5981

150	150	10	1.06252	0.561086	0.0820818	1.81128	0.0678243	12026.2	5992
150	150	10	1.05706	0.562039	0.0827275	1.81159	0.0664622	12280.8	5991
150	150	15	1.06543	0.561763	0.082293	1.8409	0.0695228	12559.3	4002
150	150	15	1.07811	0.564431	0.0830367	1.85573	0.0887389	9838.2	3997
150	150	15	1.0753	0.561932	0.0832864	1.84991	0.0839693	10410.5	3996
150	150	15	1.06487	0.561608	0.0823413	1.82764	0.0939501	9319.08	3995
150	150	15	1.06415	0.560992	0.0821068	1.8349	0.0681118	12832.1	4000
150	150	15	1.05837	0.561761	0.0827575	1.83458	0.0669024	13073.2	3999
150	150	20	1.06794	0.562082	0.0823422	1.86544	0.06958	12970.3	3002
150	150	20	1.08145	0.564474	0.0832389	1.87734	0.0890905	10132.9	2999
150	150	20	1.07609	0.562135	0.0832429	1.86855	0.084038	10752.3	2999
150	150	20	1.06674	0.561614	0.0824736	1.8473	0.0935321	9672.3	2998
150	150	20	1.06534	0.560988	0.0820788	1.85554	0.0679395	13294.3	3001
150	150	20	1.05909	0.561879	0.0827384	1.85723	0.0666499	13550.1	3002
200	200	1	2.02972	1.03565	0.146576	3.34181	0.115672	289.3	17353
200	200	1	2.04283	1.03842	0.148558	3.37186	0.125623	287	17154
200	200	1	2.03844	1.035	0.148691	3.35137	0.115832	290.05	17305
200	200	1	2.02116	1.03461	0.147395	3.33252	0.203185	284.05	16969
200	200	1	2.01972	1.03338	0.146438	3.32867	0.115309	292	17421
200	200	1	2.00503	1.0334	0.14734	3.31314	0.114511	292.5	17504
200	200	5	2.29893	1.03374	0.146514	3.61956	0.0713135	3927.05	11931
200	200	5	2.35353	1.03902	0.149103	3.69395	0.0896312	3007.52	11906
200	200	5	2.37768	1.03669	0.147929	3.70035	0.0836737	3164.75	11913
200	200	5	2.29433	1.03511	0.147138	3.61657	0.0902969	3159.77	11905
200	200	5	2.29284	1.03355	0.146389	3.61144	0.0697737	4032.5	11927
200	200	5	2.27963	1.03413	0.147798	3.59996	0.068741	4154.62	11930

200	200	10	1.93835	1.00386	0.146986	3.25775	0.0692131	9713.53	5991
200	200	10	1.96066	1.00851	0.148896	3.29382	0.0904904	7418.34	5982
200	200	10	2.03889	1.00489	0.14788	3.35692	0.083923	7899.88	5984
200	200	10	1.92736	1.00413	0.146939	3.23783	0.0915594	7376.33	5977
200	200	10	1.93923	1.00509	0.150477	3.26304	0.0697323	9634.52	5987
200	200	10	1.92084	1.00363	0.147667	3.23584	0.0671919	10047.7	5988
200	200	15	1.94197	1.00467	0.147388	3.28497	0.0723989	10741.9	3996
200	200	15	1.98393	1.00916	0.148662	3.33654	0.0902808	8580.9	3995
200	200	15	2.10371	1.00858	0.149578	3.45272	0.0840921	9124.35	3995
200	200	15	1.92877	1.00363	0.147021	3.25719	0.091685	8516.8	3993
200	200	15	1.93687	1.00463	0.147394	3.27438	0.072286	10774.2	3994
200	200	15	1.93091	1.00622	0.148205	3.2695	0.0724896	10753.6	3994
200	200	20	1.94887	1.00473	0.147114	3.30631	0.0770642	10783.1	2999
200	200	20	2.00236	1.01131	0.149478	3.38272	0.0900026	9196.7	2999
200	200	20	2.17592	1.00947	0.149722	3.54992	0.0844435	9704.63	3000
200	200	20	1.93218	1.00423	0.147361	3.27978	0.0918541	9076.1	2998
200	200	20	1.93903	1.00424	0.147246	3.29076	0.0773208	10764.2	2998
200	200	20	1.93626	1.00703	0.148393	3.29274	0.0773938	10754.7	2998
250	250	1	3.1902	1.62217	0.238002	5.28356	0.175264	183.467	10991
250	250	1	3.20741	1.6294	0.245564	5.32778	0.184091	182.183	10885
250	250	1	3.20849	1.62849	0.244668	5.31964	0.172132	182.183	10925
250	250	1	3.18631	1.62371	0.237442	5.27689	0.2543	181.417	10847
250	250	1	3.18382	1.62234	0.238226	5.27717	0.175721	184.383	11003
250	250	1	3.17329	1.62749	0.242336	5.28097	0.173509	183.767	11000
250	250	5	3.25231	1.62279	0.240496	5.35413	0.172963	192.167	10853
250	250	5	3.27211	1.62988	0.244456	5.38929	0.181144	200.883	10769

250	250	5	3.27016	1.62914	0.243388	5.38139	0.170673	183.55	10805
250	250	5	3.2497	1.62357	0.236063	5.34168	0.254937	200.467	10718
250	250	5	3.26176	1.62377	0.246699	5.37116	0.179621	216.033	10808
250	250	5	3.23906	1.6265	0.240312	5.33675	0.179887	196.65	10874
250	250	10	3.07165	1.57613	0.242923	5.1606	0.0710605	6806.37	5985
250	250	10	3.32592	1.58387	0.247442	5.43248	0.0875031	5243.97	5979
250	250	10	3.37877	1.58073	0.242973	5.46568	0.0823173	5519.73	5979
250	250	10	3.06352	1.57693	0.23904	5.13707	0.0901052	5424.85	5976
250	250	10	3.06829	1.57773	0.244427	5.15891	0.0705828	6853.35	5984
250	250	10	3.05072	1.58169	0.244036	5.14253	0.069205	7018.98	5983
250	250	15	3.07574	1.57906	0.247597	5.19949	0.0879892	7411.92	3987
250	250	15	3.42636	1.76314	0.266957	5.76097	0.0862373	7118.48	3994
250	250	15	3.44134	1.76269	0.281233	5.82356	0.0825293	7396.35	3991
250	250	15	3.07425	1.58026	0.246601	5.20177	0.088639	7354.36	3992
250	250	15	3.07346	1.57718	0.249496	5.19555	0.0880994	7408.38	3986
250	250	15	3.05065	1.57679	0.243344	5.15723	0.0883503	7419.78	3984
250	250	20	3.08424	1.58092	0.248905	5.22208	0.099466	7409.85	2995
250	250	20	3.43757	1.76622	0.282442	5.84727	0.0955931	7376.28	2997
250	250	20	3.44484	1.76135	0.280206	5.83425	0.0956375	7388.47	2993
250	250	20	3.07929	1.58211	0.247122	5.21909	0.0998437	7384.08	2994
250	250	20	3.07631	1.57844	0.248338	5.20606	0.0994661	7417.92	2995
250	250	20	3.07193	1.58028	0.2576	5.22016	0.0993508	7422.33	2994
300	300	1	4.97411	2.42526	0.425194	8.29841	0.20065	117.8	7060
300	300	1	4.97504	2.43036	0.431505	8.31215	0.212206	117.75	7039
300	300	1	4.96968	2.43586	0.425394	8.30111	0.203693	118.167	7055
300	300	1	4.96736	2.4336	0.425166	8.29507	0.280918	116.85	6996

300	300	1	4.95314	2.41985	0.417942	8.25601	0.203444	118.7	7093
300	300	1	4.91341	2.42489	0.416727	8.22112	0.205569	118.8	7120
300	300	5	4.7394	2.35999	0.423386	8.00008	0.202565	123.017	7313
300	300	5	4.74768	2.36681	0.430032	8.0201	0.212534	122.75	7287
300	300	5	4.74136	2.51809	0.427325	8.15644	0.194615	121.217	7184
300	300	5	4.71924	2.3559	0.419864	7.96003	0.283968	121.967	7277
300	300	5	4.7203	2.35472	0.420789	7.96301	0.204328	123.933	7345
300	300	5	4.68803	2.35636	0.420188	7.92753	0.203433	124.033	7378
300	300	10	4.73852	2.35408	0.428043	8.01559	0.07695	2626.63	5976
300	300	10	4.87123	2.3602	0.427376	8.14962	0.0935694	2041.9	5972
300	300	10	4.89739	2.52494	0.427274	8.33352	0.0868428	1994.49	5975
300	300	10	4.72631	2.35903	0.41475	7.97377	0.0984709	2111.62	5972
300	300	10	4.7212	2.35104	0.423466	7.98072	0.0760438	2722.48	5979
300	300	10	4.69673	2.3601	0.421956	7.96145	0.0751208	2757.72	5977
300	300	15	4.75015	2.35523	0.452973	8.11054	0.0863961	5328.5	3981
300	300	15	5.15149	2.58371	0.446148	8.70049	0.085297	4930.64	3990
300	300	15	5.14467	2.76334	0.476593	8.90348	0.0797875	5096.33	3990
300	300	15	4.73925	2.36017	0.440951	8.05825	0.0896518	5165.37	3989
300	300	15	4.73667	2.35307	0.450431	8.08759	0.086878	5323.69	3978
300	300	15	4.69635	2.35676	0.445816	8.04068	0.0875566	5321.17	3975
300	300	20	4.75576	2.35815	0.455012	8.13352	0.111471	5323.47	2986
300	300	20	5.11366	2.58129	0.497227	8.80568	0.105058	5326.8	2988
300	300	20	5.15177	2.76187	0.496081	9.01813	0.103258	5323.1	2986
300	300	20	4.75293	2.36428	0.445093	8.11289	0.111869	5318.65	2984
300	300	20	4.73618	2.35367	0.448123	8.09369	0.112055	5317.63	2983
300	300	20	4.70466	2.35685	0.440435	8.04811	0.11252	5316.95	2983

Table C.7: Raw data for Verlet (flag)
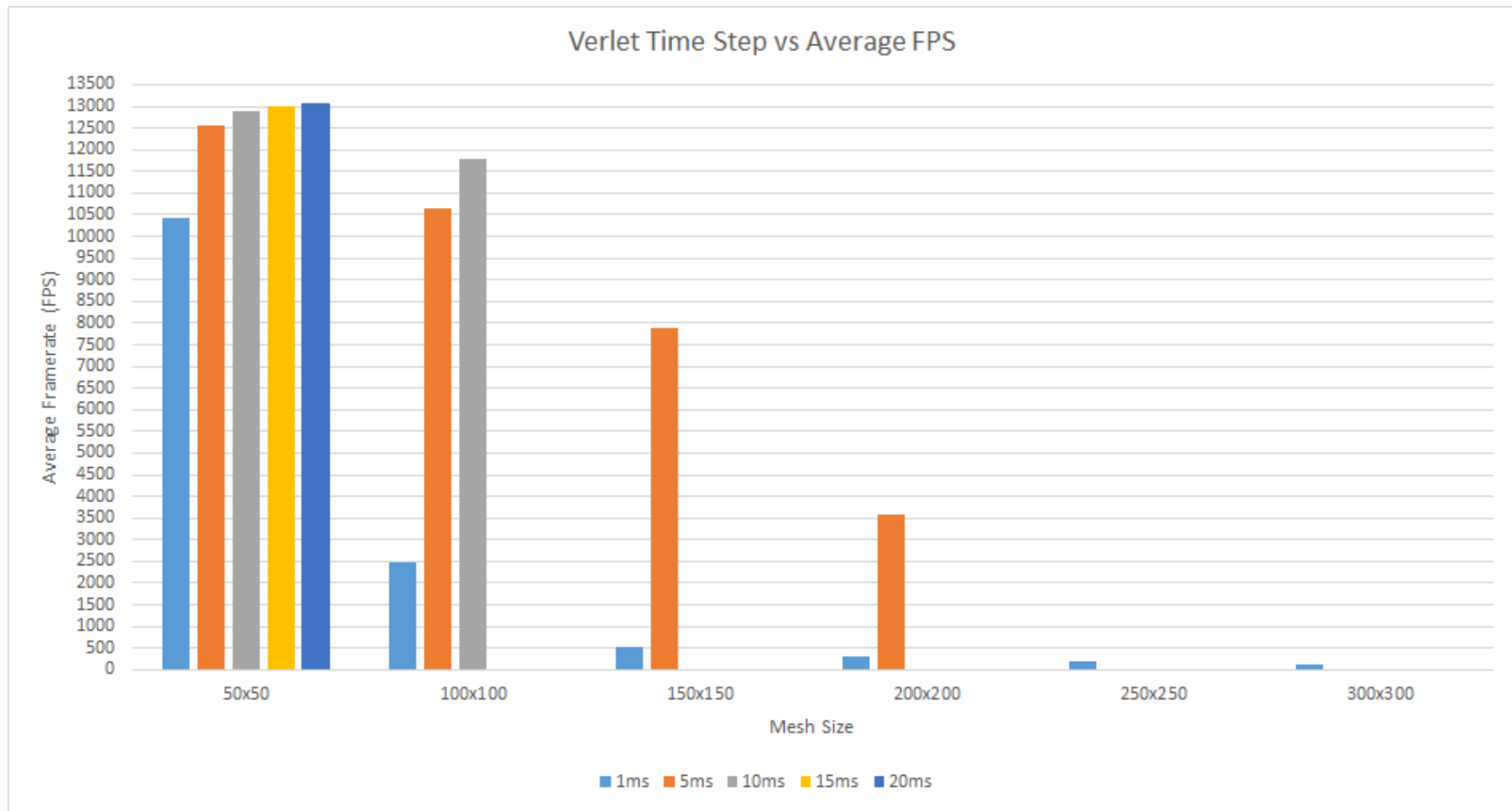
Figure C.40: Verlet mesh size against average FPS (flag)
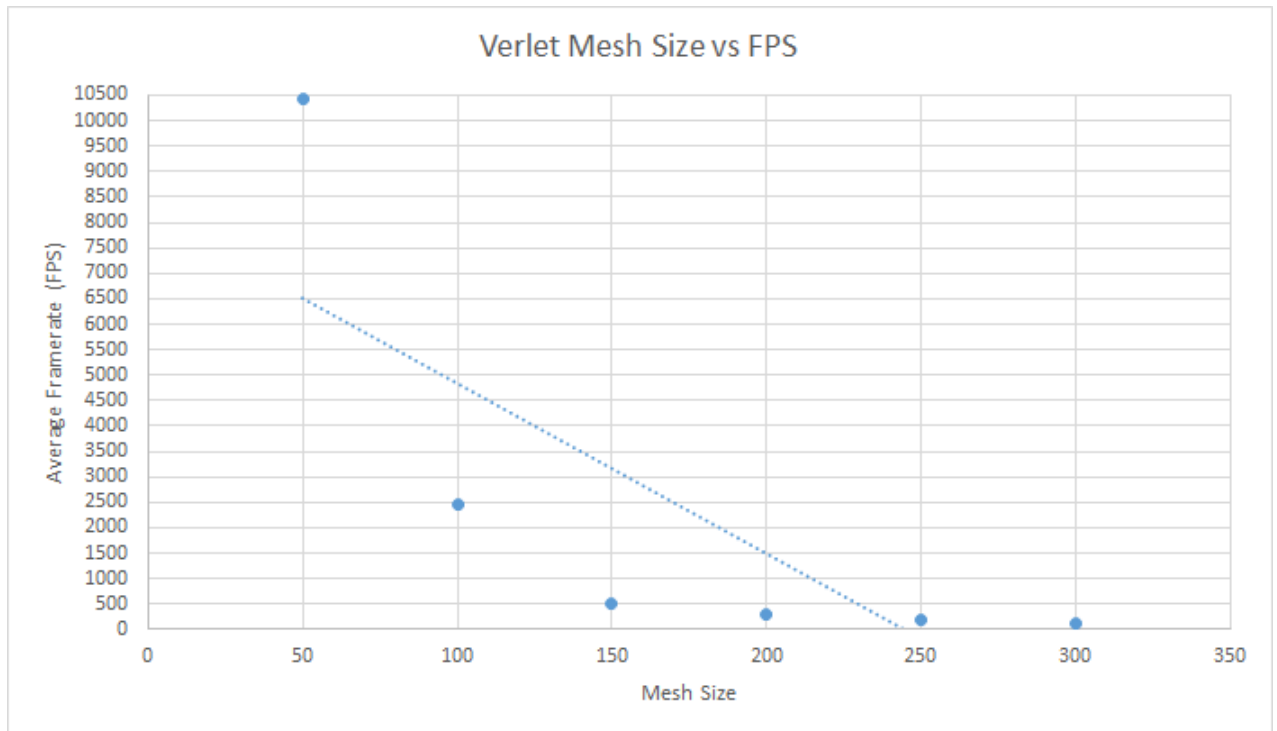
Figure C.41: Verlet mesh size against average FPS (flag)
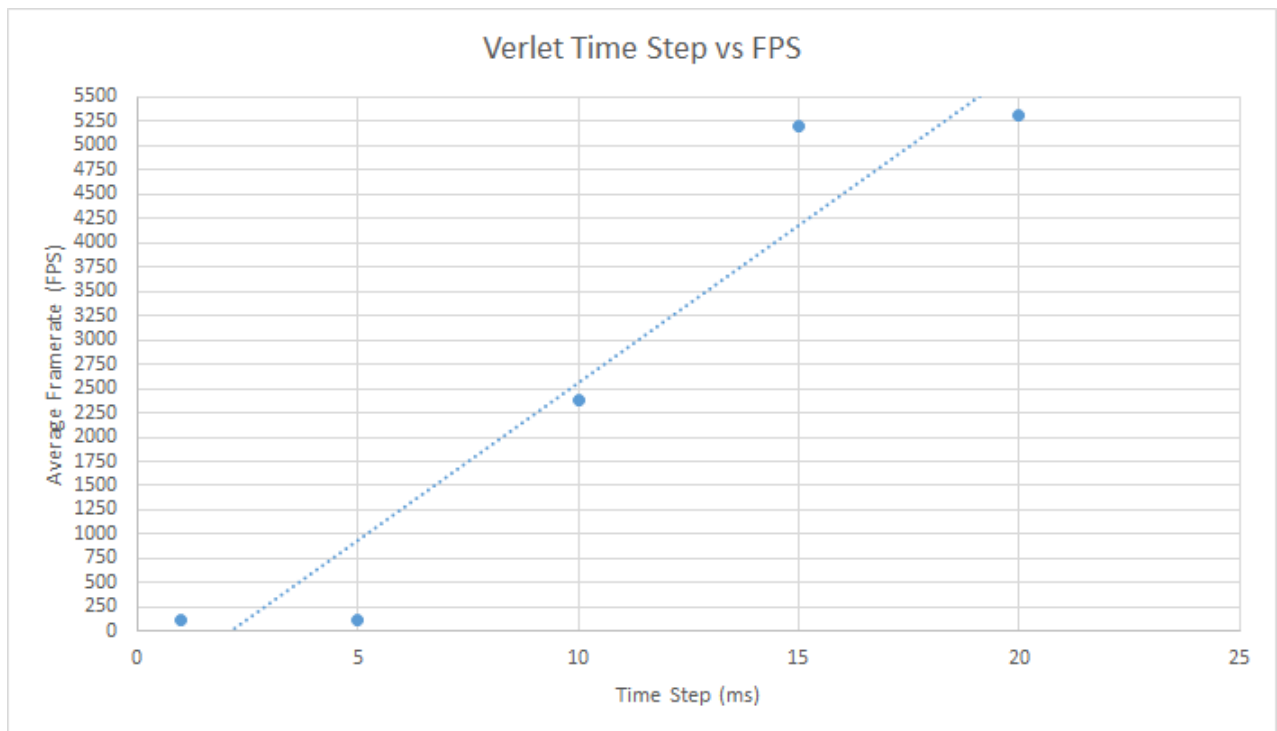
Figure C.42: Verlet time step against average FPS for a 300^2 mesh (flag)
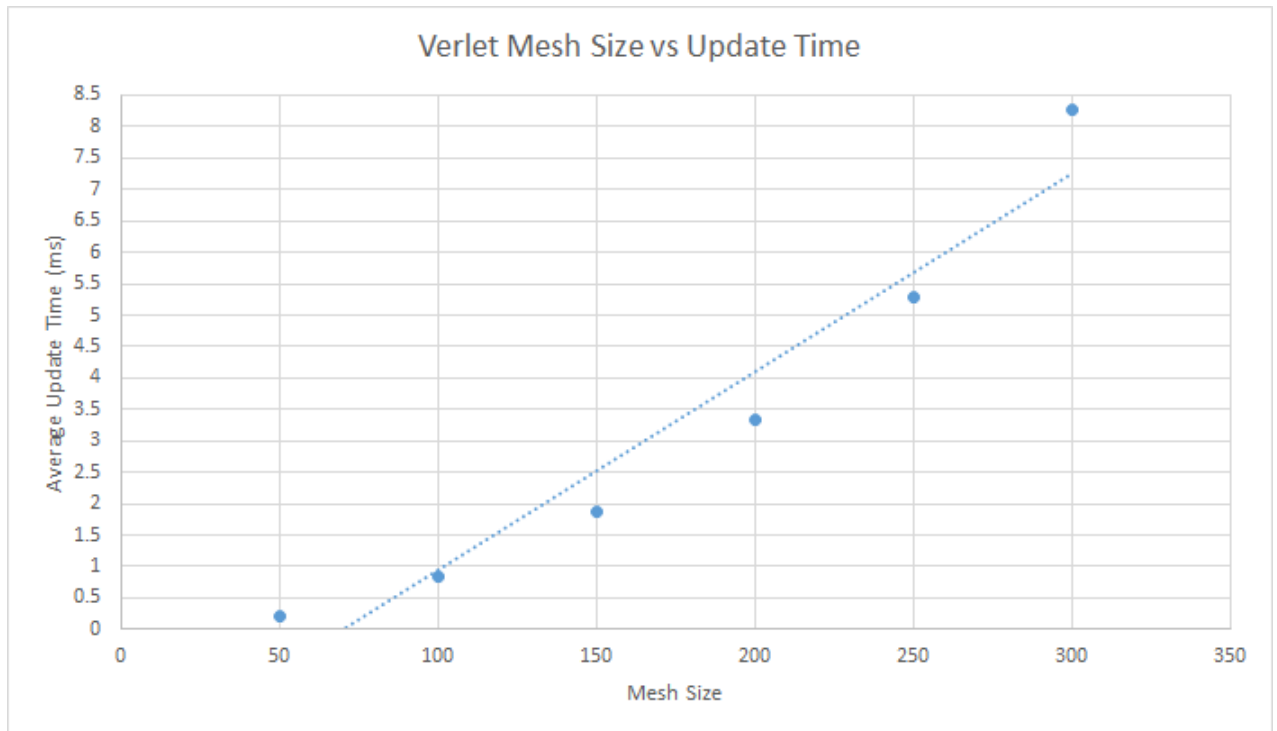
Figure C.43: Verlet mesh size against average update time (flag)
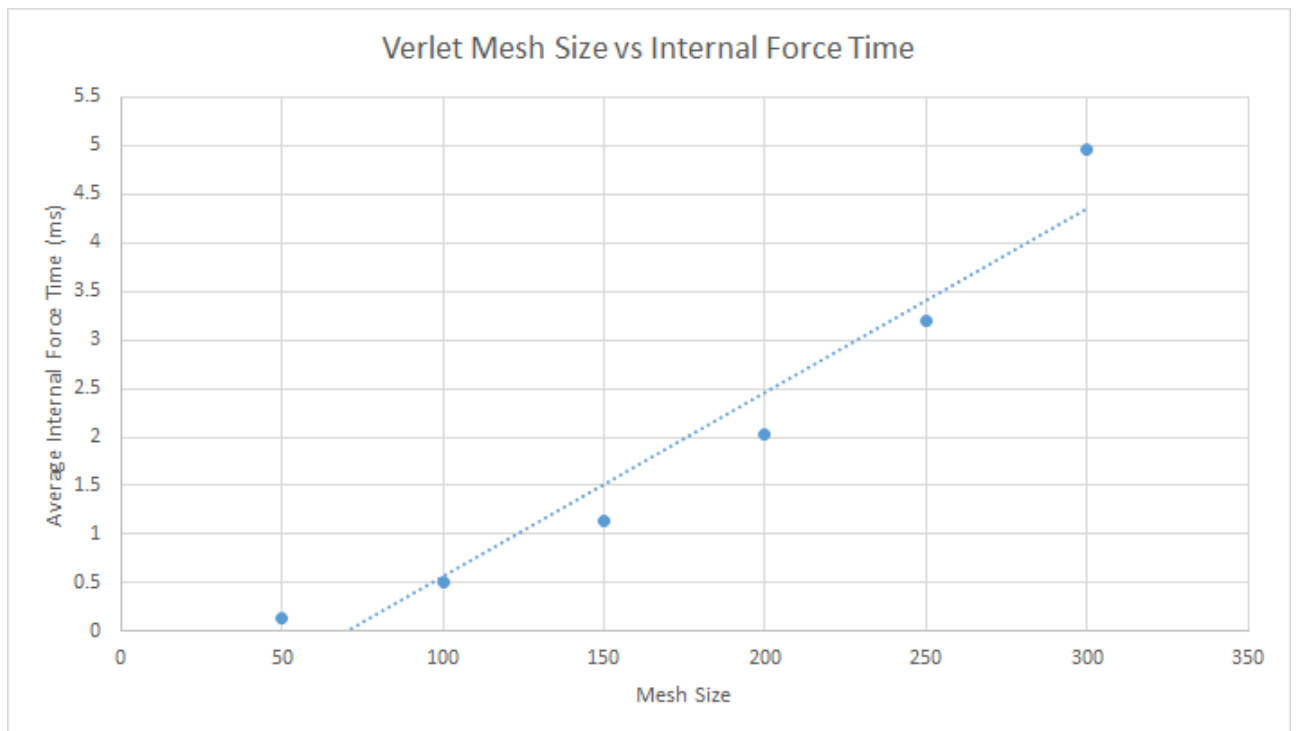
Figure C.44: Verlet mesh size against average internal force time (flag)
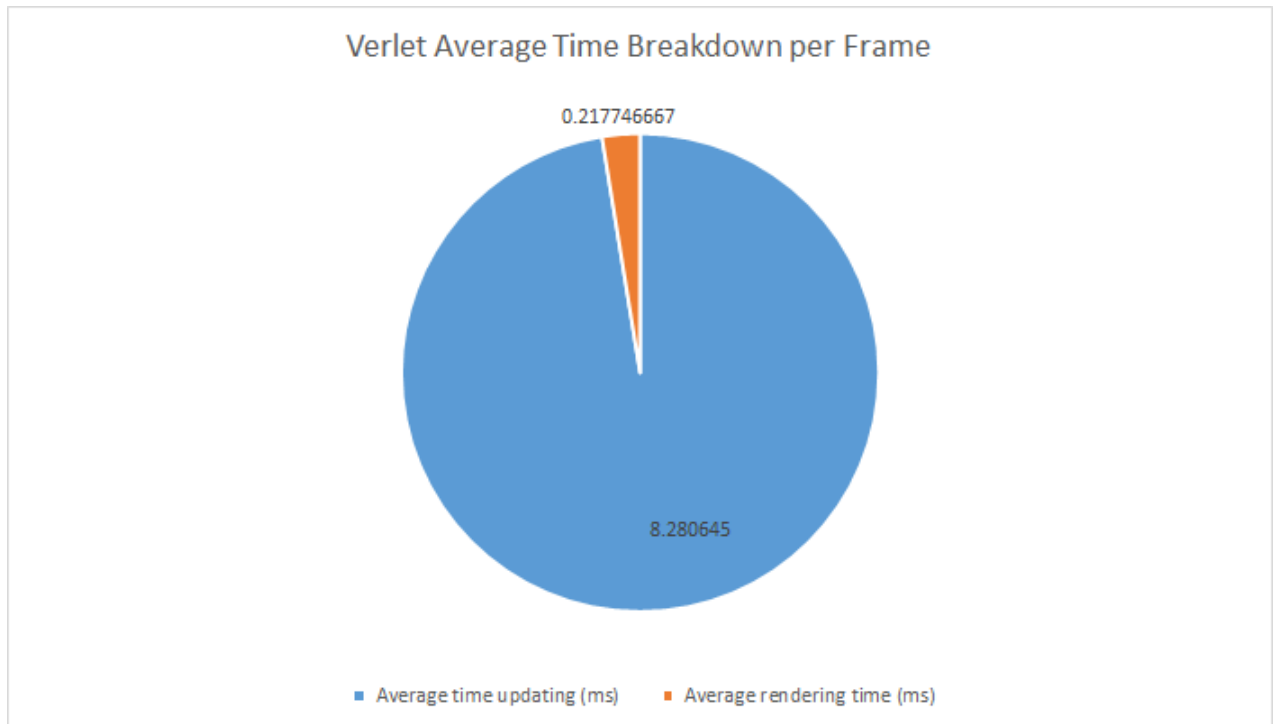
Figure C.45: Verlet frame time breakdown (flag)
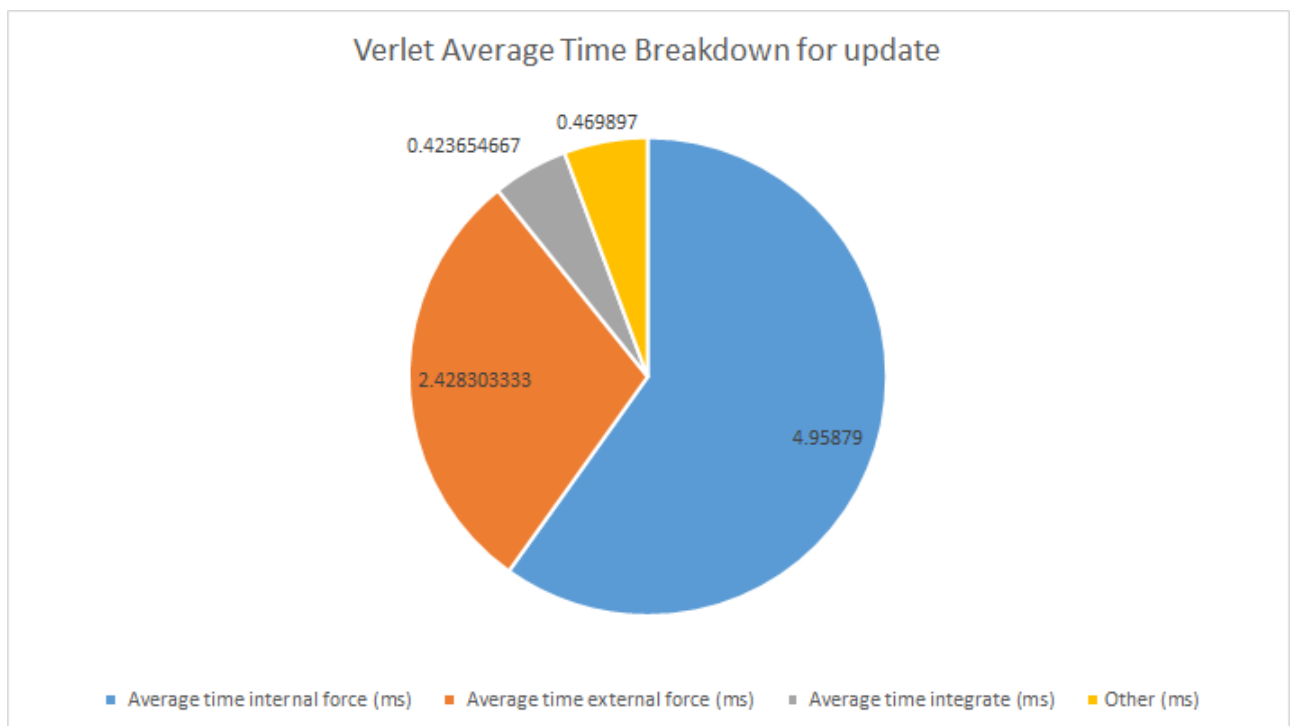
Figure C.46: Verlet update time breakdown (flag)
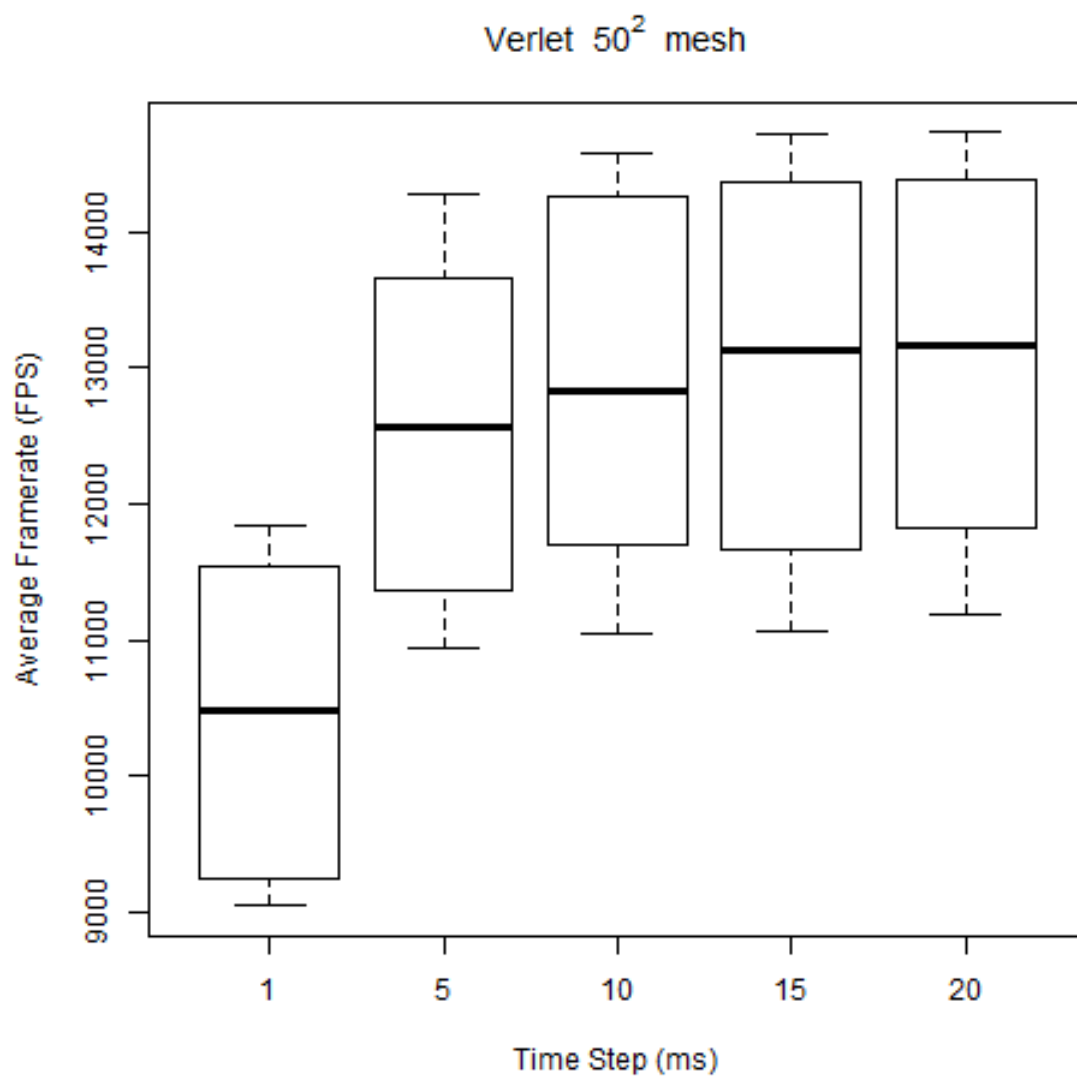
Figure C.47: Verlet box plot for a 50^2 mesh (flag)
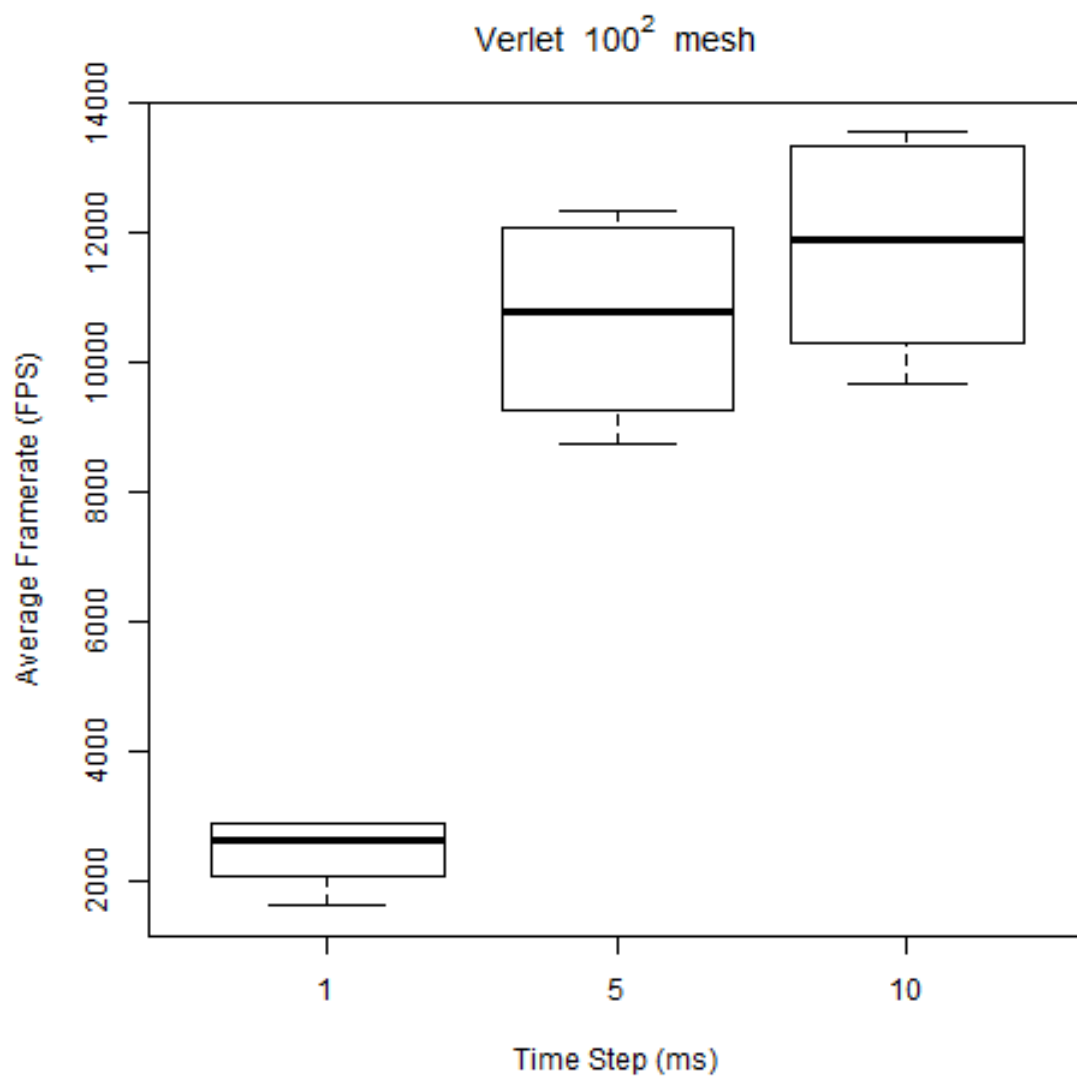
Figure C.48: Verlet box plot for a 100^2 mesh (flag)
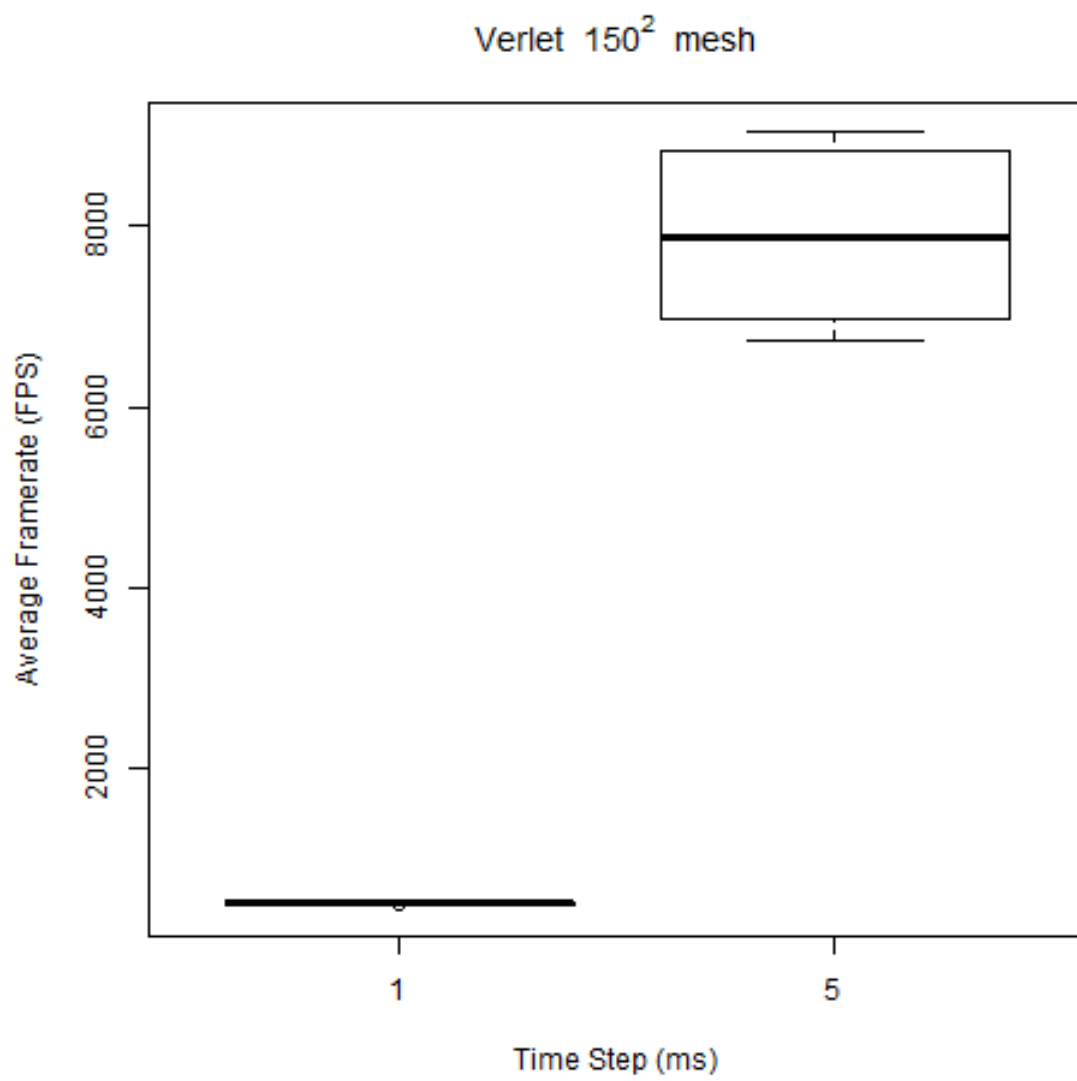
Figure C.49: Verlet box plot for a 150^2 mesh (flag)
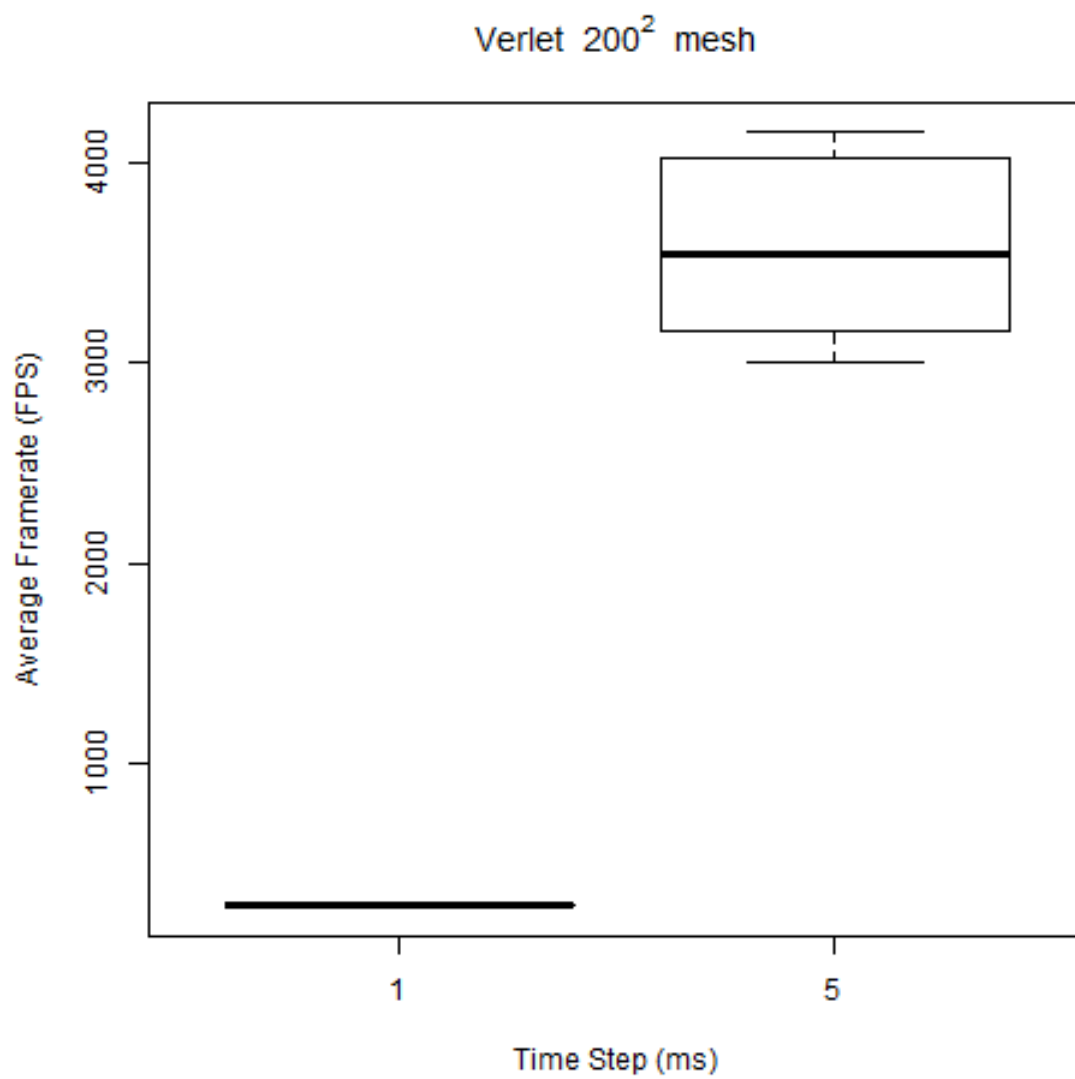
Figure C.50: Verlet box plot for a 200^2 mesh (flag)
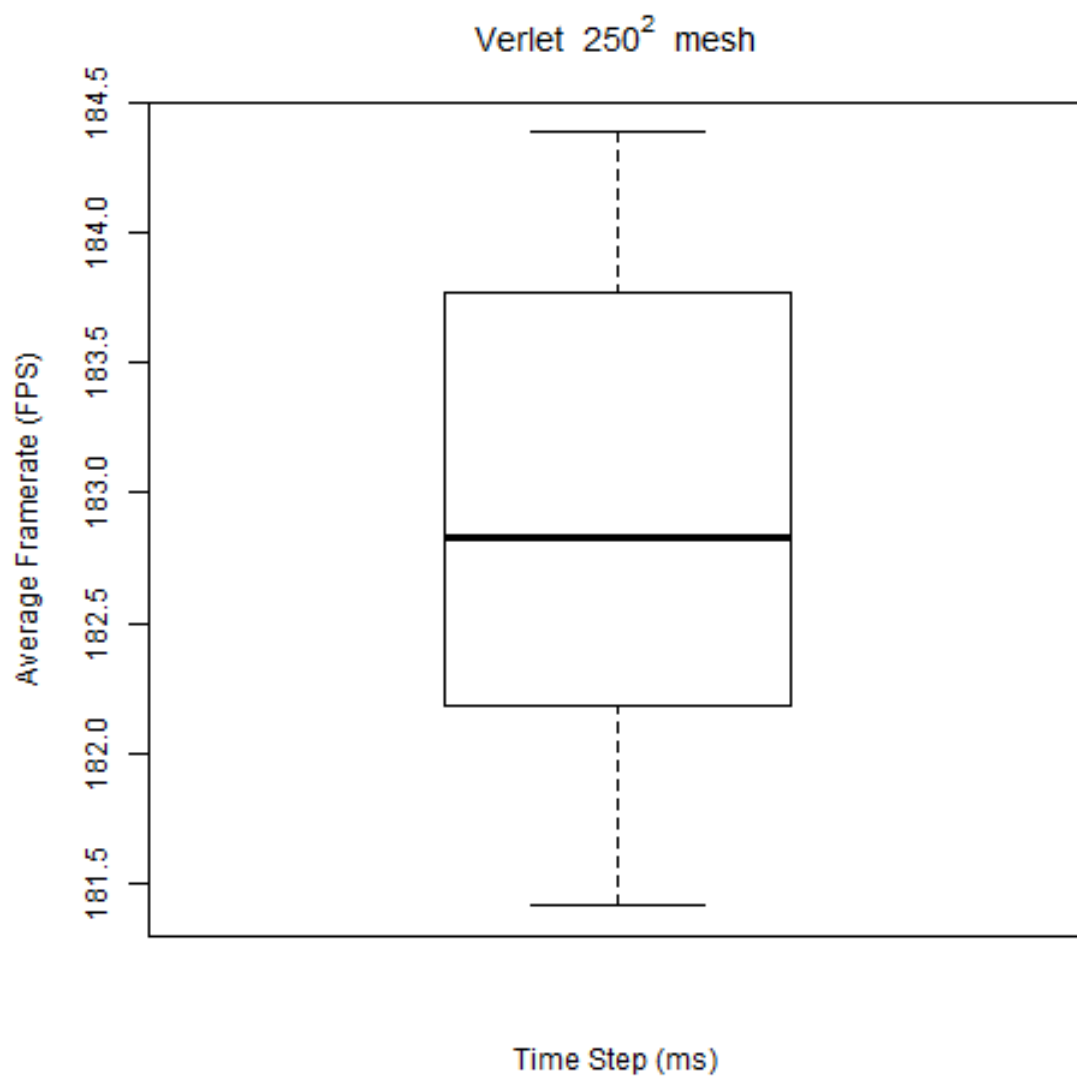
Figure C.51: Verlet box plot for a 250^2 mesh (flag)
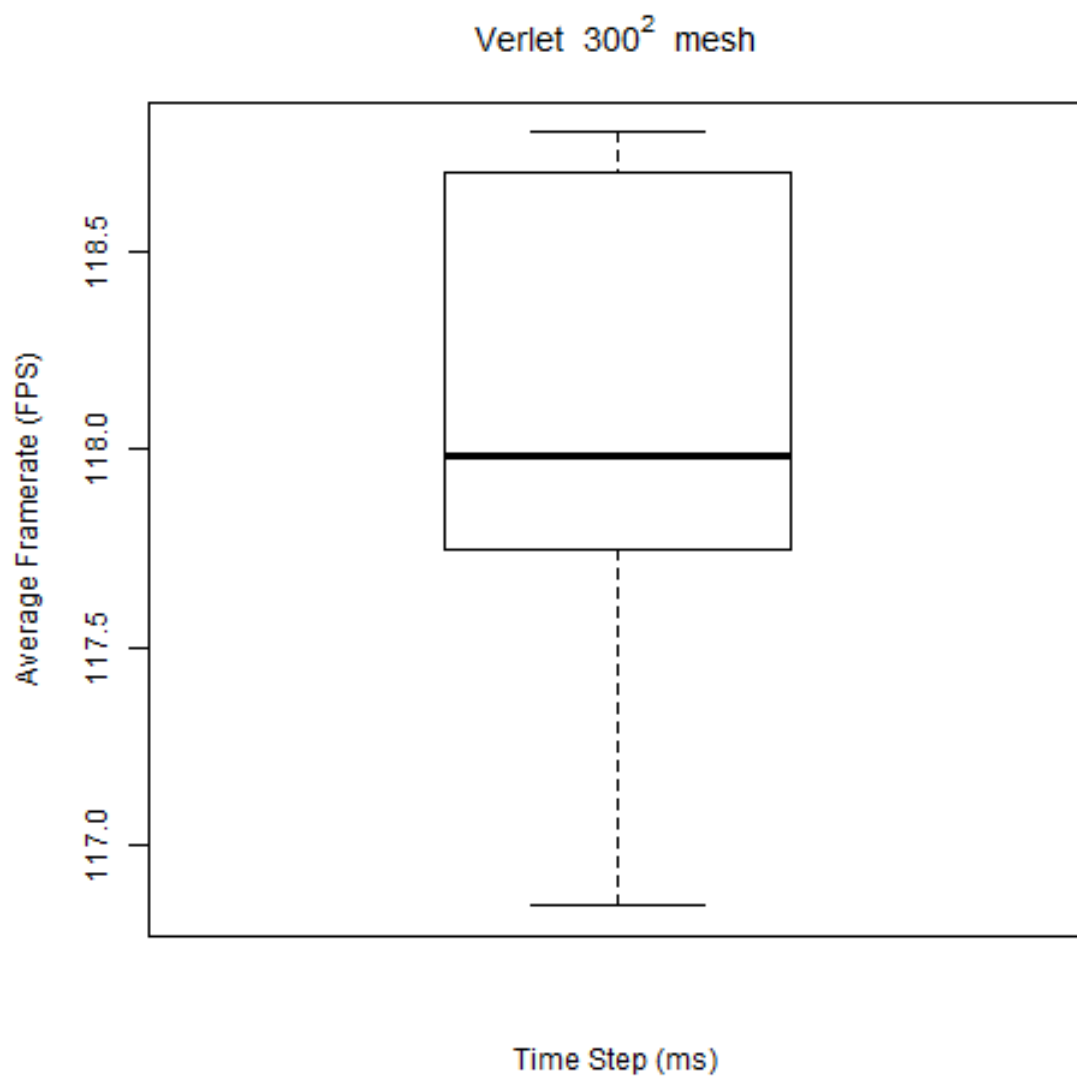
Figure C.52: Verlet box plot for a 300^2 mesh (flag)

Mesh Size	Time Step (ms)	Stable/Unstable
50^2	1	Stable
50^2	5	Stable
50^2	10	Stable
50^2	15	Stable
50^2	20	Stable
100^2	1	Stable
100^2	5	Stable
100^2	10	Stable
100^2	15	Unstable
100^2	20	Unstable
150^2	1	Stable
150^2	5	Stable
150^2	10	Unstable
150^2	15	Unstable
150^2	20	Unstable
200^2	1	Stable
200^2	5	Stable
200^2	10	Unstable
200^2	15	Unstable
200^2	20	Unstable
250^2	1	Stable
250^2	5	Unstable
250^2	10	Unstable
250^2	15	Unstable
250^2	20	Unstable
300^2	1	Stable
300^2	5	Unstable
300^2	10	Unstable
300^2	15	Unstable
300^2	20	Unstable

Table C.8: Stability results for Verlet (flag)

C.3 Midpoint

C.3.1 Sheet Data

Rows	Columns	Time Step	Internal force (ms)	External force (ms)	Integrate (ms)	Update (ms)	Render (ms)	FPS	Update count
50	50	1	0.256369	0.0102026	0.283811	0.298284	0.0697182	10102.8	57746
50	50	1	0.270808	0.00922145	0.29706	0.309403	0.0872316	7949.9	58217
50	50	1	0.267236	0.00913431	0.293367	0.307639	0.0800554	8636.2	57584
50	50	1	0.256539	0.0103008	0.28431	0.297732	0.0851409	8266.4	57705
50	50	1	0.253998	0.00909905	0.279965	0.294319	0.0688496	10328.3	57385
50	50	1	0.25478	0.00902692	0.280669	0.295051	0.0668309	10533.5	58319
50	50	5	0.289113	0.0102057	0.316612	0.348711	0.0691422	13334.4	11948
50	50	5	0.305379	0.00921413	0.331593	0.361451	0.0881546	10455.5	11914
50	50	5	0.302131	0.00914068	0.328295	0.357703	0.0810981	11401.7	11924
50	50	5	0.291878	0.010302	0.319649	0.349629	0.0838594	10996.8	11919
50	50	5	0.28875	0.00911386	0.31478	0.345532	0.0681103	13560.8	11945
50	50	5	0.289421	0.00904997	0.315365	0.347299	0.0661023	13980.7	11949
50	50	10	0.249622	0.0102176	0.277155	0.331211	0.0695914	13822.3	5997
50	50	10	0.264908	0.00921108	0.29111	0.339667	0.0890508	10795	5986
50	50	10	0.261983	0.0091464	0.288134	0.337297	0.0789689	12182.3	5990
50	50	10	0.252385	0.0103119	0.28019	0.32892	0.0861578	11179.7	5988
50	50	10	0.24947	0.00911707	0.275512	0.326294	0.0677703	14194.8	5995
50	50	10	0.250274	0.00903981	0.276206	0.330044	0.0660572	14573.4	5994
50	50	15	0.24047	0.0102116	0.268017	0.34392	0.0695556	13973.7	4003
50	50	15	0.256385	0.00922726	0.282628	0.350259	0.0891525	10908.9	3997
50	50	15	0.252751	0.0091429	0.278935	0.346518	0.0800754	12149.3	3998
50	50	15	0.243107	0.0102907	0.270889	0.340096	0.08399	11587.1	3998

50	50	15	0.240382	0.00911814	0.266439	0.337216	0.0679637	14313.8	4001
50	50	15	0.241115	0.00903331	0.267052	0.342679	0.0660521	14725	4001
50	50	20	0.236703	0.0102328	0.264282	0.362055	0.0694045	14071.4	3004
50	50	20	0.252314	0.0092226	0.278543	0.365899	0.085916	11372.9	3001
50	50	20	0.249116	0.00913006	0.275284	0.360378	0.0817886	11961.1	3001
50	50	20	0.239357	0.0103108	0.267163	0.355293	0.0848871	11523.1	3001
50	50	20	0.236478	0.00910685	0.262547	0.353384	0.0680048	14373.4	3003
50	50	20	0.237224	0.0090406	0.263155	0.360033	0.066069	14794.2	3003
100	100	1	1.0702	0.0498632	1.19315	1.22573	0.0775964	817.05	46031
100	100	1	1.09838	0.0502542	1.22334	1.25609	0.0811211	884.717	44865
100	100	1	1.10324	0.0501474	1.22715	1.26007	0.0760153	805.817	44904
100	100	1	1.07964	0.0498732	1.20372	1.23633	0.17005	890.098	42659
100	100	1	1.08093	0.0498332	1.20483	1.23781	0.077722	949.533	45603
100	100	1	1.07342	0.0502146	1.19767	1.23031	0.0735147	981.583	46014
100	100	5	0.949248	0.0498628	1.07246	1.12308	0.0694247	11086.7	11945
100	100	5	0.965558	0.050318	1.0907	1.13856	0.0904344	8423.17	11918
100	100	5	0.970291	0.0502075	1.09476	1.14392	0.0836881	9148.75	11918
100	100	5	0.959469	0.0499103	1.08367	1.13048	0.0941895	8207.42	11906
100	100	5	0.958958	0.0499005	1.08314	1.13259	0.0681992	11150.8	11953
100	100	5	0.952152	0.050199	1.07641	1.12666	0.0664205	11459.9	11943
100	100	10	0.945703	0.0499634	1.06906	1.14154	0.0693529	12684.9	5996
100	100	10	0.960546	0.0503445	1.0856	1.15224	0.0888716	9919.18	5985
100	100	10	0.965386	0.0501622	1.08934	1.15671	0.0824404	10665.2	5987
100	100	10	0.954726	0.0499561	1.07906	1.14278	0.0946014	9328.17	5982
100	100	10	0.954164	0.0499374	1.07846	1.14771	0.0679753	12960.4	5994
100	100	10	0.947813	0.0502342	1.0722	1.14455	0.0660043	13351.1	5993

100	100	15	0.938993	0.0499652	1.06243	1.13849	0.099246	9320.46	3994
100	100	15	1.03148	0.0541354	1.16775	1.24802	0.107868	8476.67	3992
100	100	15	1.02698	0.0540974	1.16298	1.24248	0.103105	8896.17	3992
100	100	15	0.948814	0.0498788	1.07301	1.14808	0.107733	8551.95	3991
100	100	15	0.947822	0.049925	1.0722	1.14523	0.0995127	9277.68	3992
100	100	15	0.941695	0.0502625	1.06618	1.14239	0.094995	9724.68	3992
100	100	20	0.92401	0.0500566	1.04773	1.1634	0.0707082	13249.8	3002
100	100	20	0.939703	0.0503429	1.06504	1.17174	0.0898136	10426.4	2999
100	100	20	0.942778	0.0501986	1.067	1.18583	0.0788549	11868.5	3001
100	100	20	0.93267	0.0499423	1.05711	1.16354	0.0857615	10921.2	2999
100	100	20	0.932428	0.0499416	1.0569	1.16722	0.0694461	13488.8	3001
100	100	20	0.925876	0.0502787	1.05035	1.16534	0.0676933	13841.5	3000
150	150	1	2.25769	0.113288	2.5361	2.60689	0.0945274	491.25	22210
150	150	1	2.28566	0.114198	2.5682	2.64497	0.096877	501.767	21882
150	150	1	2.30124	0.113905	2.58193	2.65853	0.0888945	448.583	21838
150	150	1	2.27971	0.113311	2.56038	2.63255	0.164571	519.667	21450
150	150	1	2.27894	0.113244	2.55931	2.63084	0.0970039	570.767	21995
150	150	1	2.26188	0.114053	2.54276	2.61346	0.0909024	585.3	22185
150	150	5	2.14528	0.113239	2.42359	2.50734	0.0695404	7114.62	11930
150	150	5	2.16917	0.114413	2.45211	2.54318	0.0905872	5376.83	11897
150	150	5	2.17026	0.11457	2.45245	2.53992	0.0848051	5782.83	11914
150	150	5	2.16725	0.113462	2.44813	2.53026	0.0874585	5587.52	11917
150	150	5	2.16603	0.113423	2.44682	2.53029	0.0687846	7088.56	11929
150	150	5	2.15125	0.114038	2.43239	2.51618	0.0666767	7344.03	11941
150	150	10	2.13972	0.113842	2.41909	2.50952	0.133566	5606.45	5952
150	150	10	2.26621	0.121616	2.56443	2.66672	0.14075	5214	5948

150	150	10	2.27123	0.121451	2.56978	2.6711	0.134023	5457.52	5950
150	150	10	2.16121	0.113674	2.44269	2.53366	0.144935	5155.82	5947
150	150	10	2.15826	0.113419	2.43913	2.52868	0.136883	5463.17	5948
150	150	10	2.14497	0.114166	2.4262	2.5162	0.13739	5451.18	5948
150	150	15	2.11472	0.113801	2.39405	2.523	0.0702823	11742.3	3994
150	150	15	2.12273	0.114797	2.40575	2.53416	0.089778	9214.42	3989
150	150	15	2.13451	0.114197	2.41526	2.54194	0.0849444	9714.17	3989
150	150	15	2.13094	0.113885	2.4126	2.53107	0.0962311	8611.32	3987
150	150	15	2.13617	0.114323	2.41868	2.54623	0.0692462	11928.9	3992
150	150	15	2.11871	0.114073	2.39979	2.53015	0.0671547	12317.7	3991
150	150	20	2.11814	0.114504	2.39861	2.55046	0.0699921	12396.4	2999
150	150	20	2.12315	0.114925	2.40639	2.55384	0.0891297	9741.12	2996
150	150	20	2.13614	0.114867	2.4177	2.56415	0.0845522	10275.7	2996
150	150	20	2.13262	0.113853	2.41433	2.55083	0.0960785	9045.23	2995
150	150	20	2.134	0.113447	2.41493	2.56254	0.0686736	12622.9	2998
150	150	20	2.12036	0.114063	2.40148	2.55373	0.0671889	12913.7	2998
200	200	1	3.96982	0.207508	4.47147	4.60728	0.14364	334.683	12629
200	200	1	3.9908	0.211293	4.50264	4.64887	0.152468	325.1	12496
200	200	1	4.01657	0.208335	4.52232	4.65582	0.139588	252.8	12512
200	200	1	4.01087	0.207164	4.51754	4.65232	0.222742	302.867	12307
200	200	1	4.00564	0.206533	4.51027	4.64344	0.146366	374.483	12527
200	200	1	3.98056	0.20662	4.48511	4.61873	0.143916	376.733	12598
200	200	5	3.86507	0.20823	4.36778	4.50514	0.0804379	1294.38	11914
200	200	5	3.91522	0.210862	4.42605	4.57514	0.0933585	974.2	11905
200	200	5	3.96382	0.209386	4.47067	4.60702	0.0825571	1023.9	11916
200	200	5	3.89939	0.206916	4.40509	4.5406	0.113773	857.633	11907

200	200	5	3.90018	0.207565	4.40624	4.54394	0.0826005	1154.65	11926
200	200	5	3.87485	0.207494	4.38062	4.51587	0.0765359	1313.15	11931
200	200	10	3.83799	0.21155	4.34527	4.50946	0.0706746	7662.13	5981
200	200	10	3.84847	0.21437	4.3642	4.53823	0.0915744	5881.87	5972
200	200	10	3.92853	0.209372	4.43563	4.59694	0.0849318	6323.17	5975
200	200	10	3.85359	0.207313	4.35992	4.51687	0.0929543	5832.87	5969
200	200	10	3.87564	0.216137	4.39737	4.56324	0.0711765	7520.85	5977
200	200	10	3.83861	0.208715	4.34583	4.50803	0.0682323	7923.52	5979
200	200	15	3.8488	0.214166	4.36013	4.55094	0.0700082	9843.66	3991
200	200	15	3.85552	0.214477	4.37157	4.56622	0.0912421	7603.23	3987
200	200	15	3.9949	0.213479	4.50811	4.69685	0.0852598	7997.1	3988
200	200	15	3.85561	0.208766	4.36424	4.54198	0.0926648	7459.26	3985
200	200	15	3.86964	0.211734	4.38192	4.56672	0.0693654	9987.87	3990
200	200	15	3.84515	0.211087	4.35606	4.54204	0.0677259	10257.8	3990
200	200	20	3.85486	0.216638	4.37029	4.58376	0.0709106	10812.8	2997
200	200	20	3.87732	0.215818	4.3954	4.6112	0.0906122	8451.43	2995
200	200	20	4.06146	0.215656	4.57826	4.78842	0.0851261	8870.82	2996
200	200	20	3.86585	0.211708	4.37784	4.57347	0.0918968	8367.55	2994
200	200	20	3.88039	0.21705	4.40168	4.61106	0.0709696	10785.1	2995
200	200	20	3.86002	0.211816	4.37217	4.57833	0.071146	10789.1	2995
250	250	1	6.24649	0.369442	7.11179	7.36205	0.18946	217.35	7945
250	250	1	6.25858	0.375912	7.14107	7.39367	0.19564	197.917	7906
250	250	1	6.31977	0.372606	7.19058	7.43901	0.183418	138.783	7872
250	250	1	6.30585	0.368547	7.16916	7.41364	0.271003	166.933	7808
250	250	1	6.29549	0.366865	7.16033	7.40588	0.19314	227.333	7896
250	250	1	6.24589	0.360402	7.09436	7.33596	0.193244	224.25	7969

250	250	5	6.09994	0.369951	6.96478	7.21313	0.188079	135.783	8105
250	250	5	6.11453	0.386196	7.01632	7.27479	0.191026	134.383	8036
250	250	5	6.17553	0.376693	7.05484	7.30797	0.184135	134.3	8007
250	250	5	6.16235	0.373291	7.03735	7.29063	0.268269	132.467	7937
250	250	5	6.14375	0.361657	7.00158	7.24628	0.194809	135.05	8062
250	250	5	6.11061	0.362609	6.96233	7.20212	0.189372	135.967	8116
250	250	10	6.09644	0.401744	6.99046	7.24898	0.0744418	3733.72	5945
250	250	10	6.19557	0.375249	7.07912	7.34394	0.0945414	2848.12	5940
250	250	10	6.26286	0.369167	7.12699	7.38535	0.0873759	3063.13	5939
250	250	10	6.10943	0.362267	6.96365	7.21998	0.096441	2932.5	5939
250	250	10	6.09501	0.365166	6.95634	7.21807	0.0751779	3751.47	5934
250	250	10	6.10326	0.392323	6.98315	7.23893	0.0723888	3857.18	5942
250	250	15	6.09724	0.391666	6.99925	7.29469	0.0712371	7114.65	3987
250	250	15	6.48966	0.407644	7.43955	7.72671	0.0881934	5432.15	3985
250	250	15	6.54578	0.400684	7.47879	7.76263	0.0825851	5824.4	3985
250	250	15	6.13363	0.380581	7.02164	7.30358	0.0900476	5657.82	3984
250	250	15	6.12294	0.38189	7.01577	7.31103	0.070663	7154.75	3987
250	250	15	6.10394	0.383823	6.99363	7.28894	0.0698795	7268.97	3983
250	250	20	6.10359	0.393583	7.01402	7.33285	0.0859447	7318.02	2987
250	250	20	6.7764	0.421682	7.75498	8.07291	0.0870301	6789.1	2990
250	250	20	6.82637	0.439676	7.83932	8.17177	0.0817564	7202.2	2989
250	250	20	6.1471	0.384728	7.04278	7.34686	0.0892456	7044.62	2989
250	250	20	6.1299	0.38991	7.03246	7.33595	0.0858904	7324.75	2986
250	250	20	6.10592	0.400813	7.02016	7.33436	0.0857192	7345.05	2985
300	300	1	9.70445	0.709592	11.2706	11.7478	0.205074	105.883	5020
300	300	1	9.7736	0.719602	11.3666	11.8516	0.216302	98.85	4972

300	300	1	9.79357	0.720995	11.3794	11.8565	0.204179	141.767	4975
300	300	1	9.76987	0.70389	11.3272	11.7999	0.288754	160.433	4964
300	300	1	9.73216	0.695296	11.2821	11.7545	0.210314	100.833	5015
300	300	1	9.6731	0.70232	11.2239	11.6941	0.209173	93.2	5041
300	300	5	9.51305	0.702765	11.0696	11.5459	0.203355	86.0833	5106
300	300	5	9.58997	0.719051	11.1798	11.6605	0.217453	84.8167	5051
300	300	5	9.57587	0.707306	11.1362	11.6079	0.20421	85.3667	5079
300	300	5	9.5753	0.704927	11.1357	11.6094	0.289354	84.15	5042
300	300	5	9.53712	0.708278	11.1064	11.5792	0.20459	85.8333	5091
300	300	5	9.48486	0.704389	11.0389	11.5075	0.204183	86.4167	5122
300	300	10	9.39914	0.700818	10.9502	11.4269	0.20444	87.5	5157
300	300	10	9.46587	0.714276	11.0517	11.533	0.214768	86.65	5106
300	300	10	9.43056	0.704279	10.988	11.4608	0.201067	87.1	5144
300	300	10	9.45587	0.691528	10.9935	11.4627	0.287573	86.0328	5105
300	300	10	9.41937	0.704809	10.9853	11.4615	0.203248	87.2667	5142
300	300	10	9.3481	0.696923	10.886	11.3545	0.204126	88.0667	5189
300	300	15	9.42516	0.720587	11.0193	11.5215	0.0756118	3090	3979
300	300	15	9.74117	0.726951	11.3516	11.8576	0.0891838	2377.77	3977
300	300	15	9.76717	0.712732	11.3399	11.8369	0.0845786	2504.37	3978
300	300	15	9.4659	0.704866	11.0287	11.5191	0.0949021	2480.63	3979
300	300	15	9.44971	0.724839	11.0554	11.5529	0.0748849	3098.3	3978
300	300	15	9.3829	0.719587	10.9704	11.4624	0.0729243	3265.27	3978
300	300	20	9.45029	0.745279	11.0963	11.6574	0.0794586	5241.62	2977
300	300	20	10.1162	0.767107	11.8228	12.3496	0.0864112	4419.48	2982
300	300	20	10.1918	0.773618	11.8894	12.405	0.0804923	4673.62	2983
300	300	20	9.47346	0.724557	11.0768	11.5902	0.0908258	4642.93	2981

300	300	20	9.45418	0.739152	11.0936	11.6502	0.0795529	5250.18	2976
300	300	20	9.39461	0.761189	11.0576	11.6148	0.0798504	5259.42	2976

Table C.9: Raw data for Midpoint (sheet)
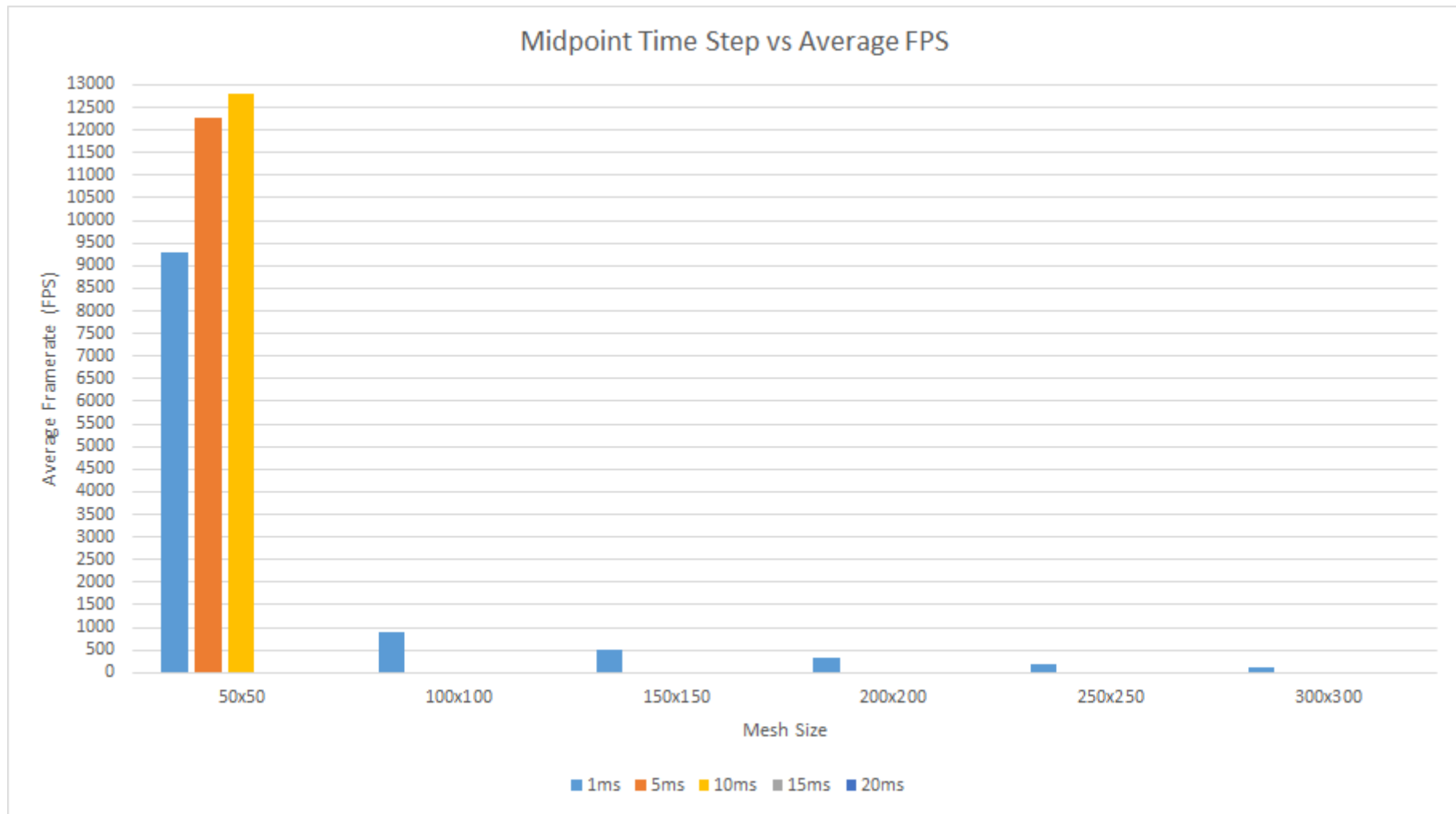
Figure C.53: Midpoint time step against average FPS (sheet)
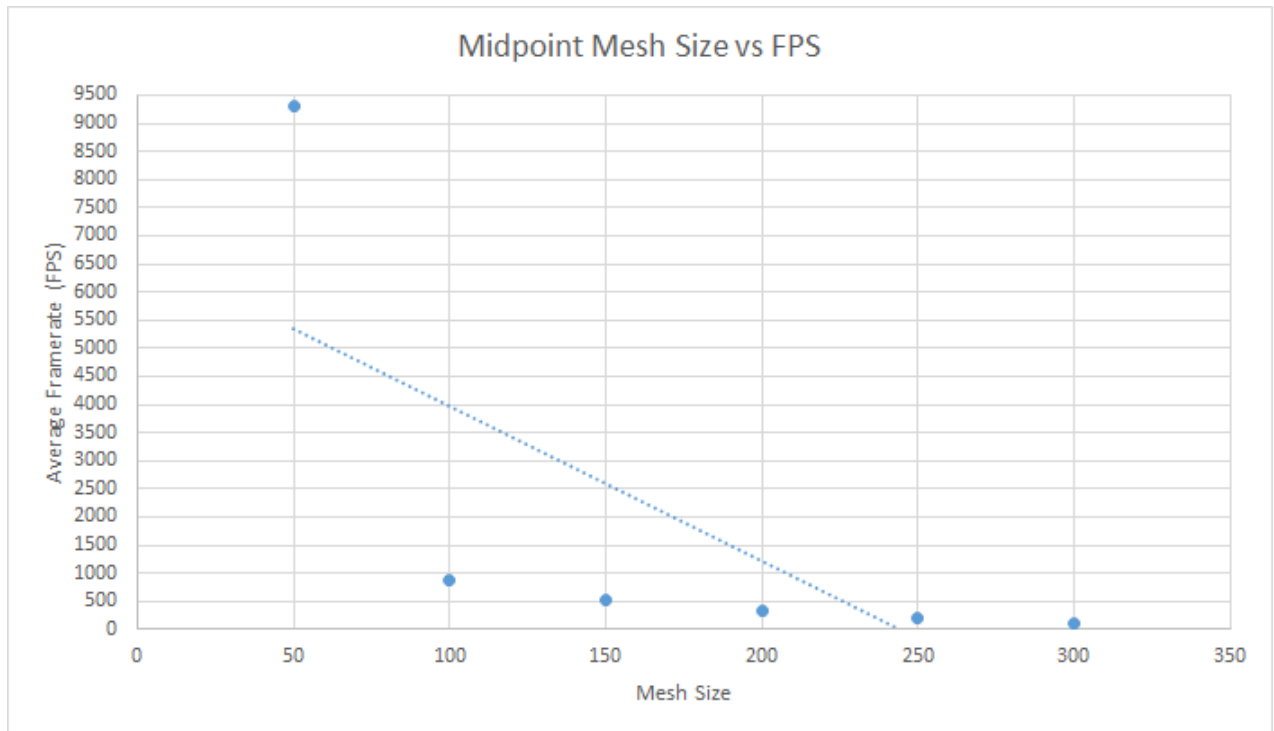
Figure C.54: Midpoint mesh size against average FPS (sheet)
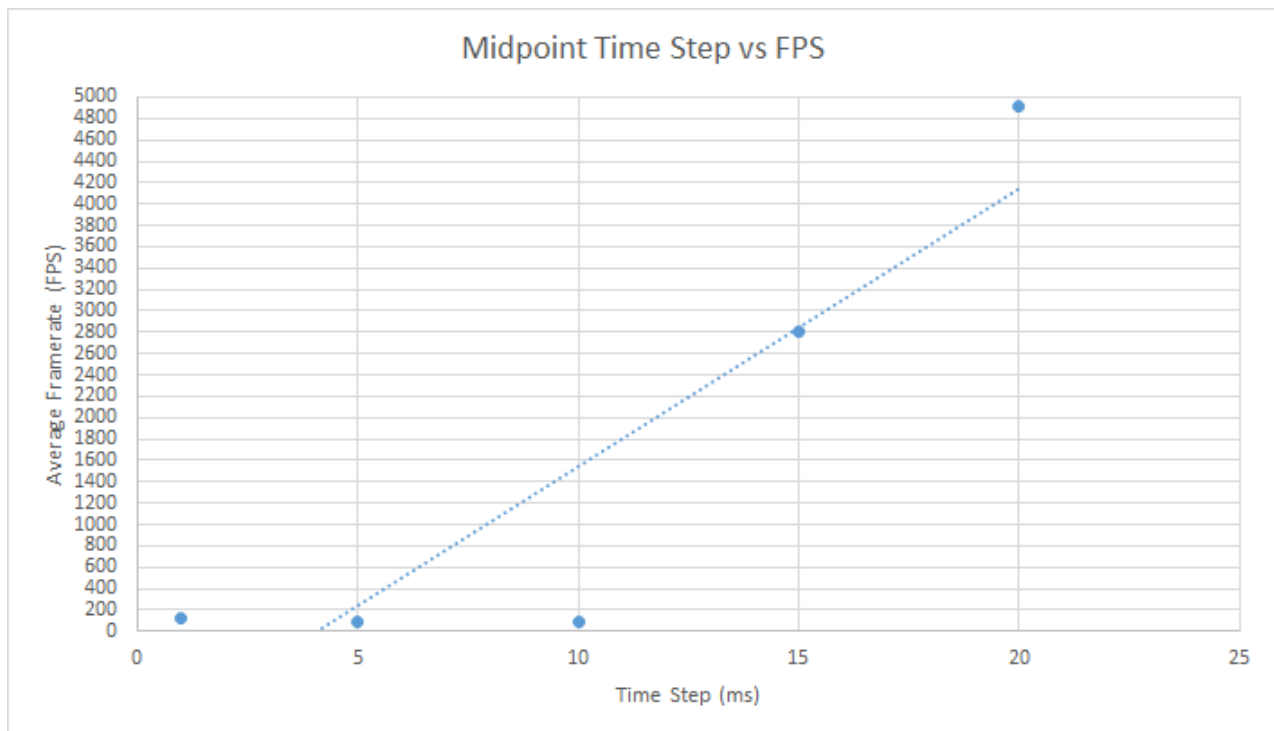
Figure C.55: Midpoint time step against average FPS for a 300^2 mesh (sheet)
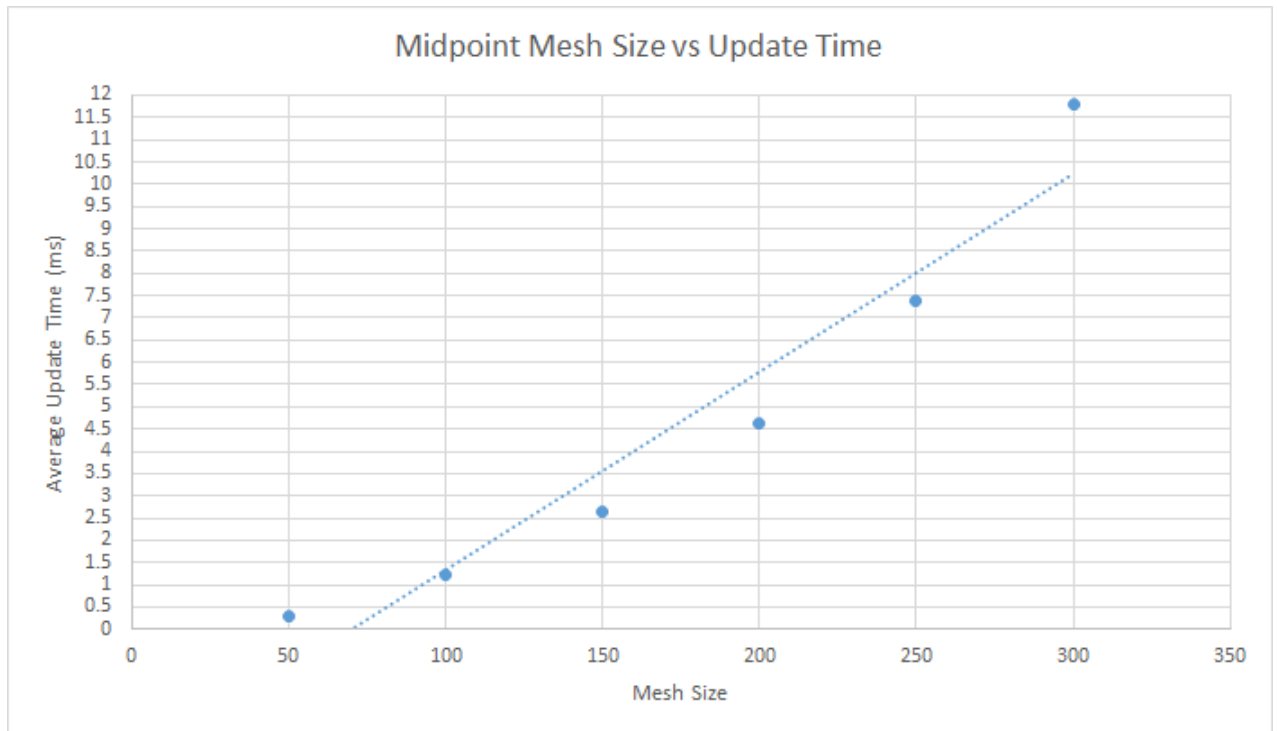
Figure C.56: Midpoint mesh size against average update time (sheet)
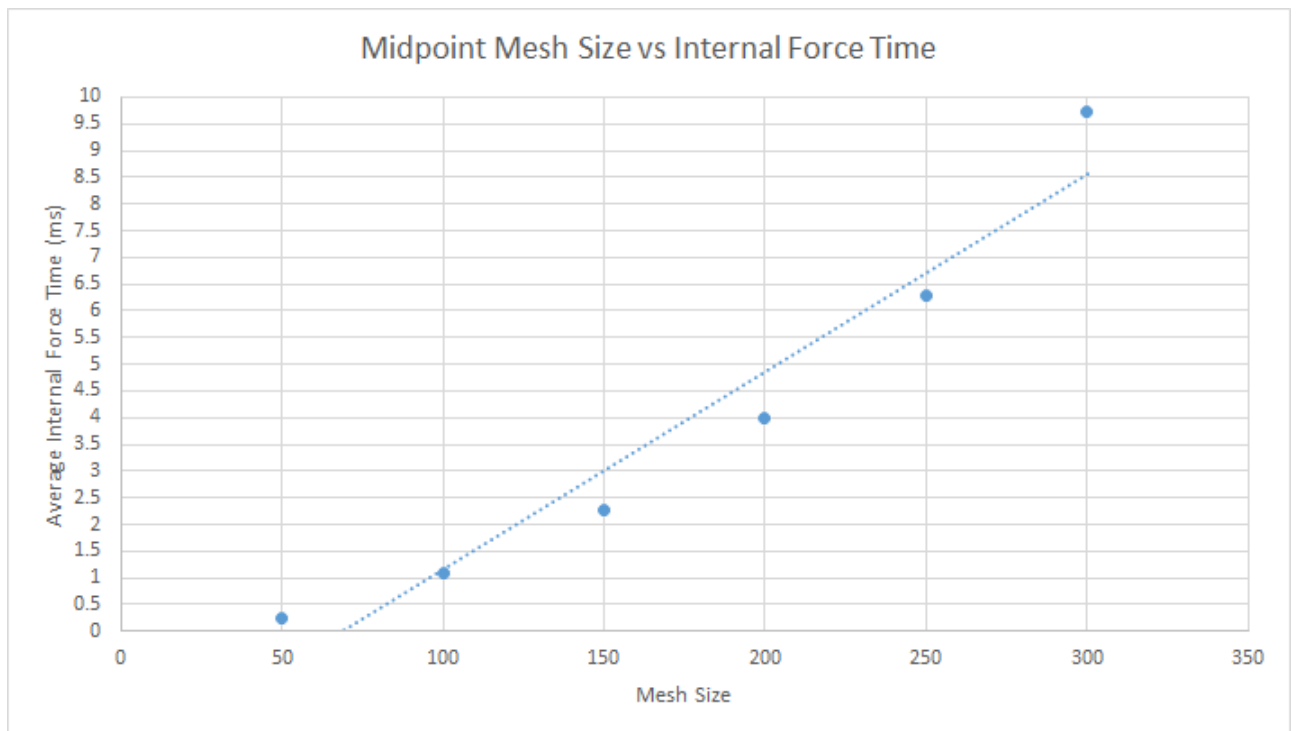
Figure C.57: Midpoint mesh size against average internal force time (sheet)
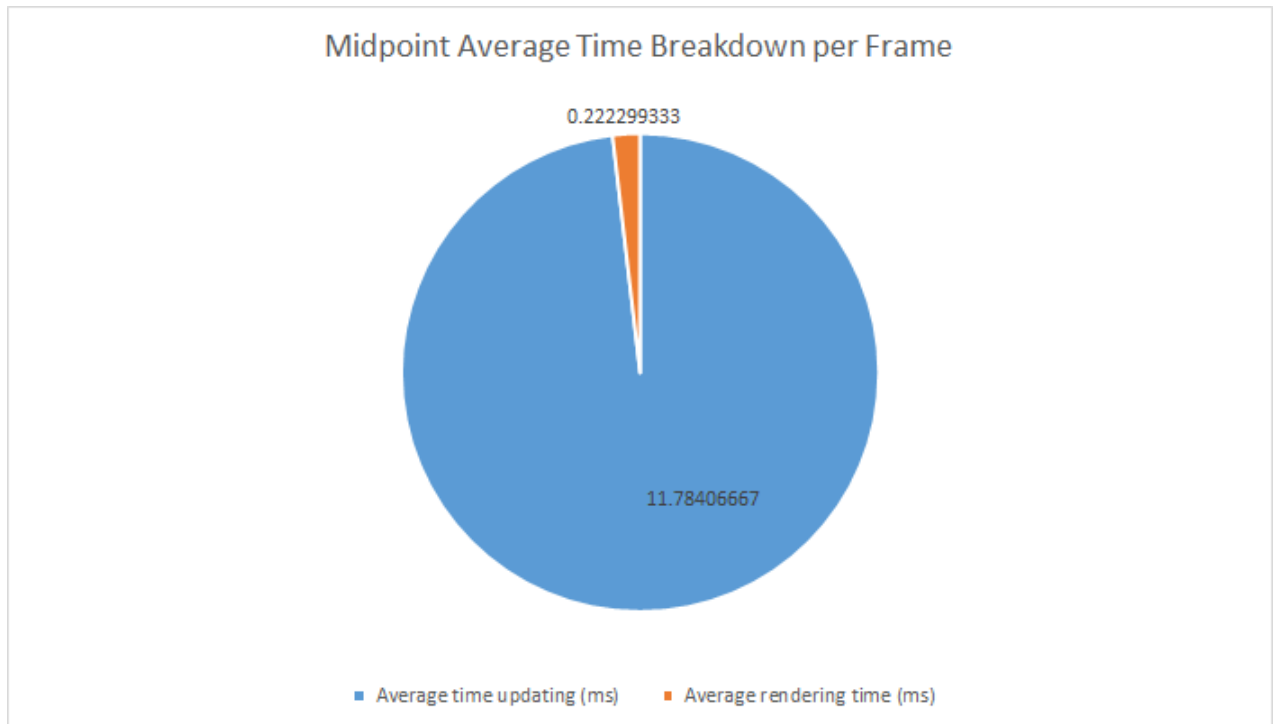
Figure C.58: Midpoint frame time breakdown (sheet)
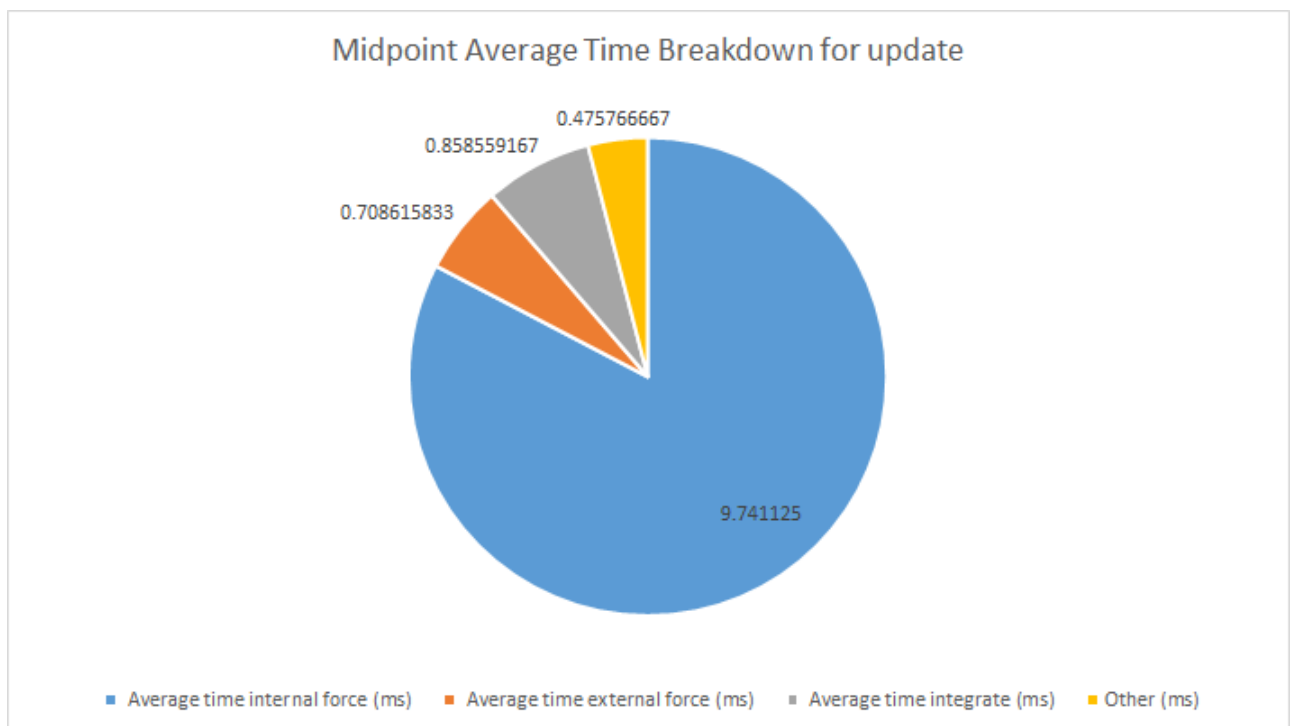
Figure C.59: Midpoint update time breakdown (sheet)
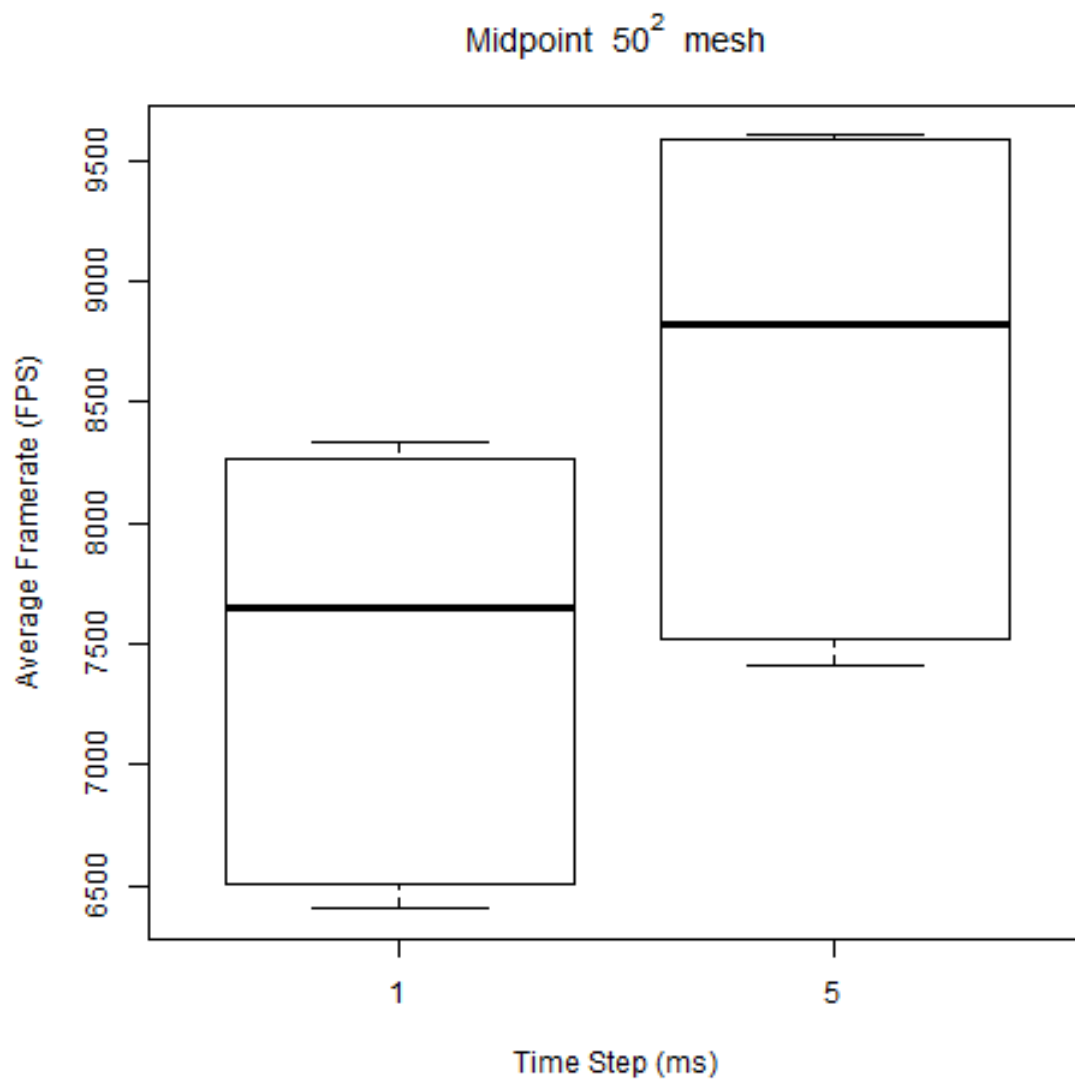
Figure C.60: Midpoint box plot for a 50^2 mesh (sheet)
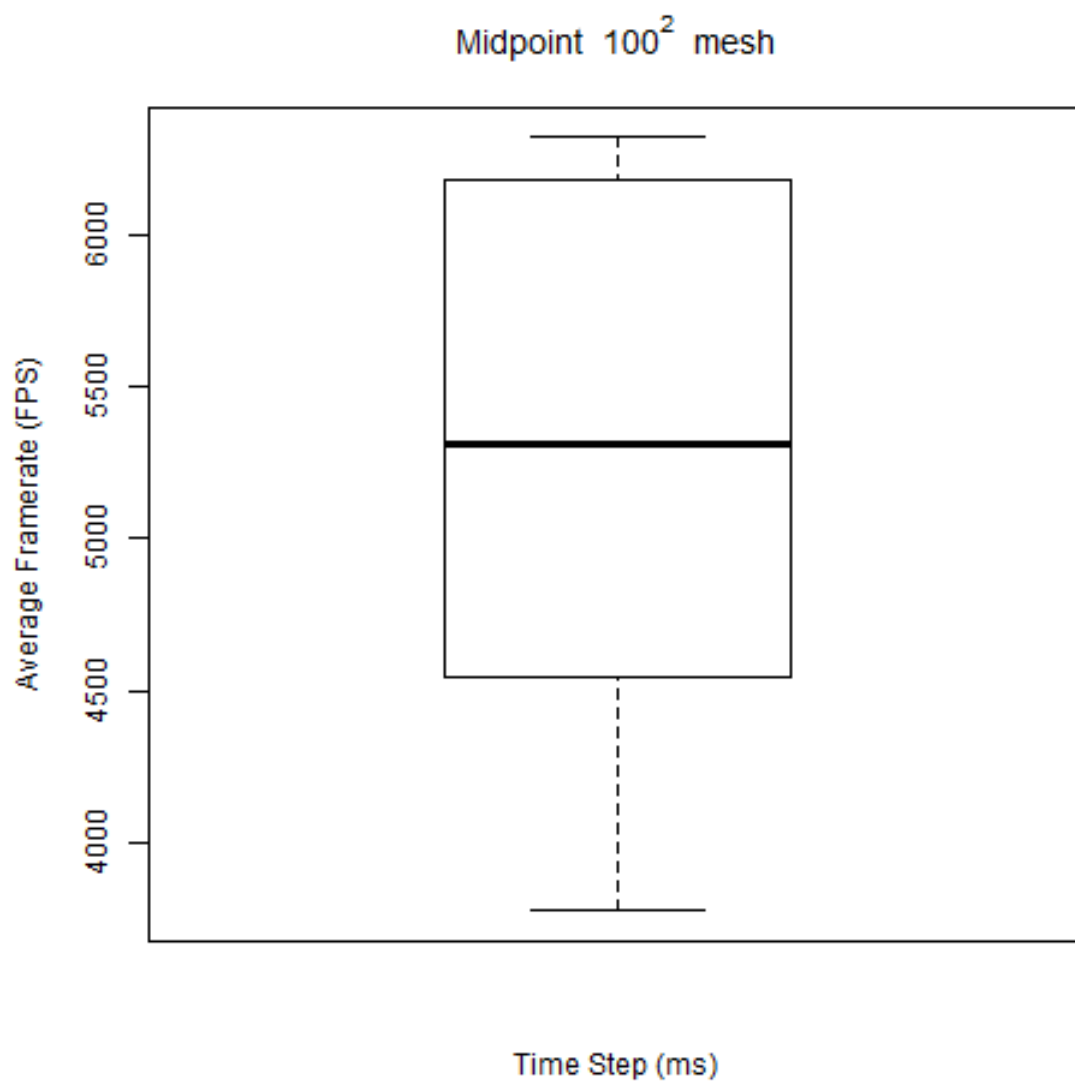
Figure C.61: Midpoint box plot for a 100^2 mesh (sheet)
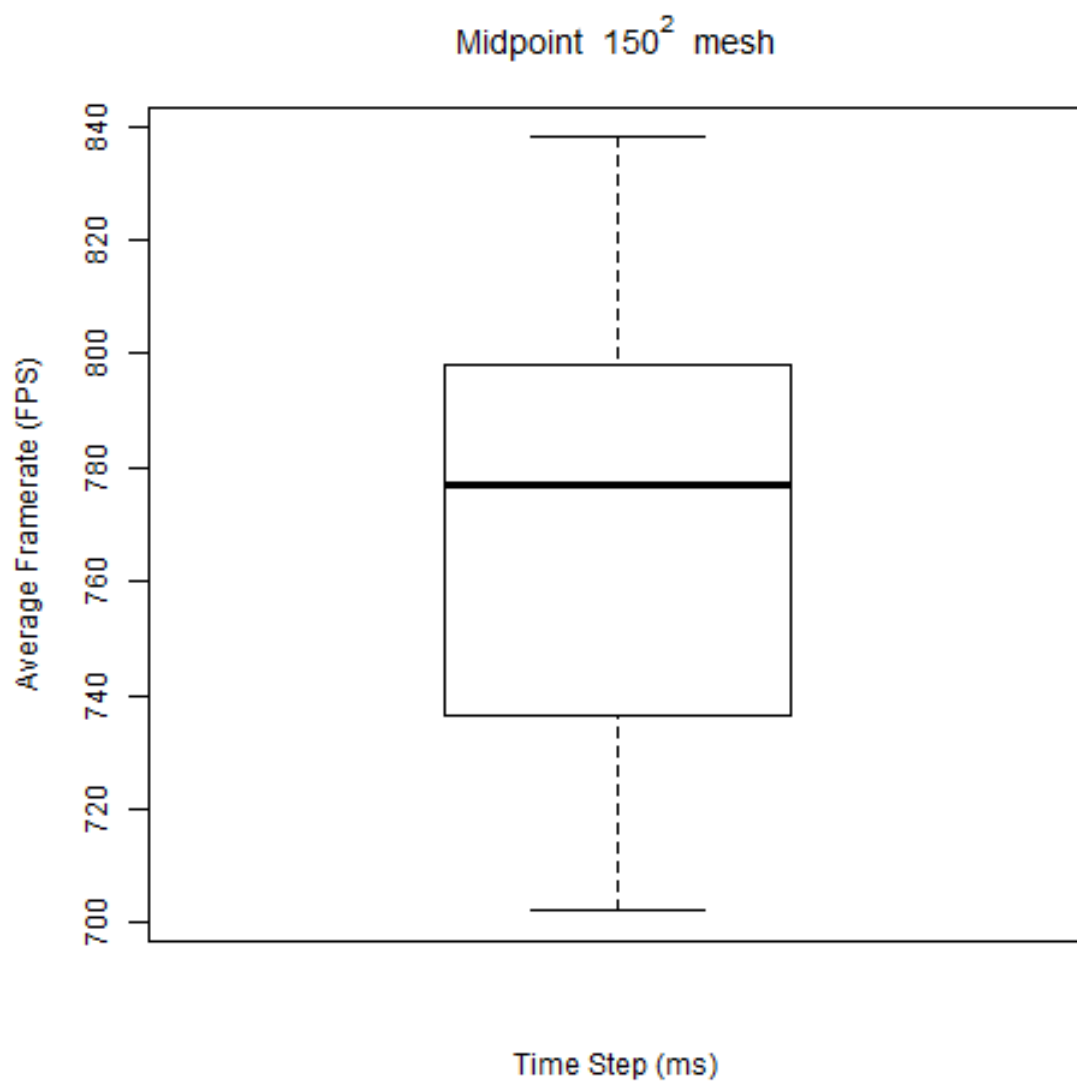
Figure C.62: Midpoint box plot for a 150^2 mesh (sheet)
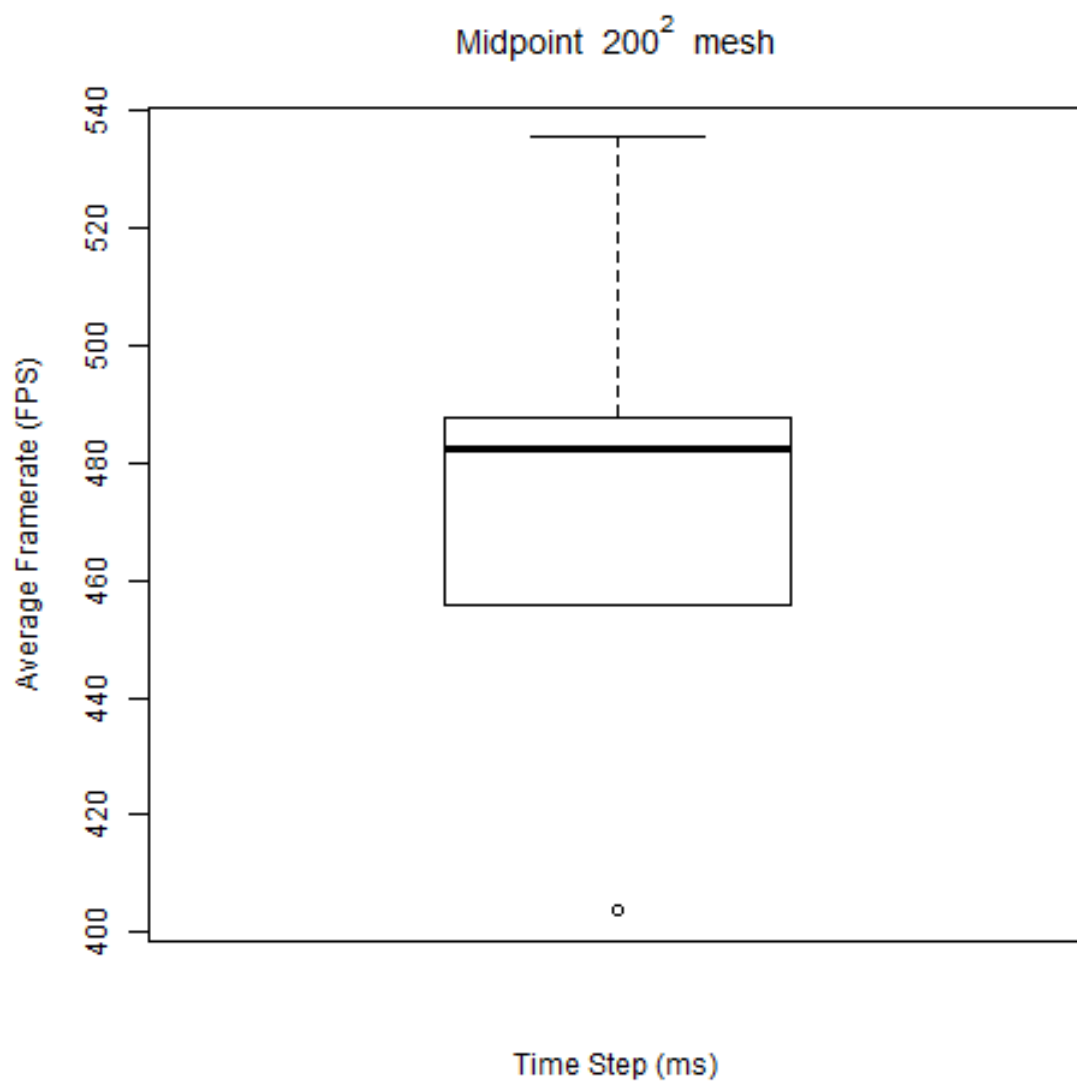
Figure C.63: Midpoint box plot for a 200^2 mesh (sheet)
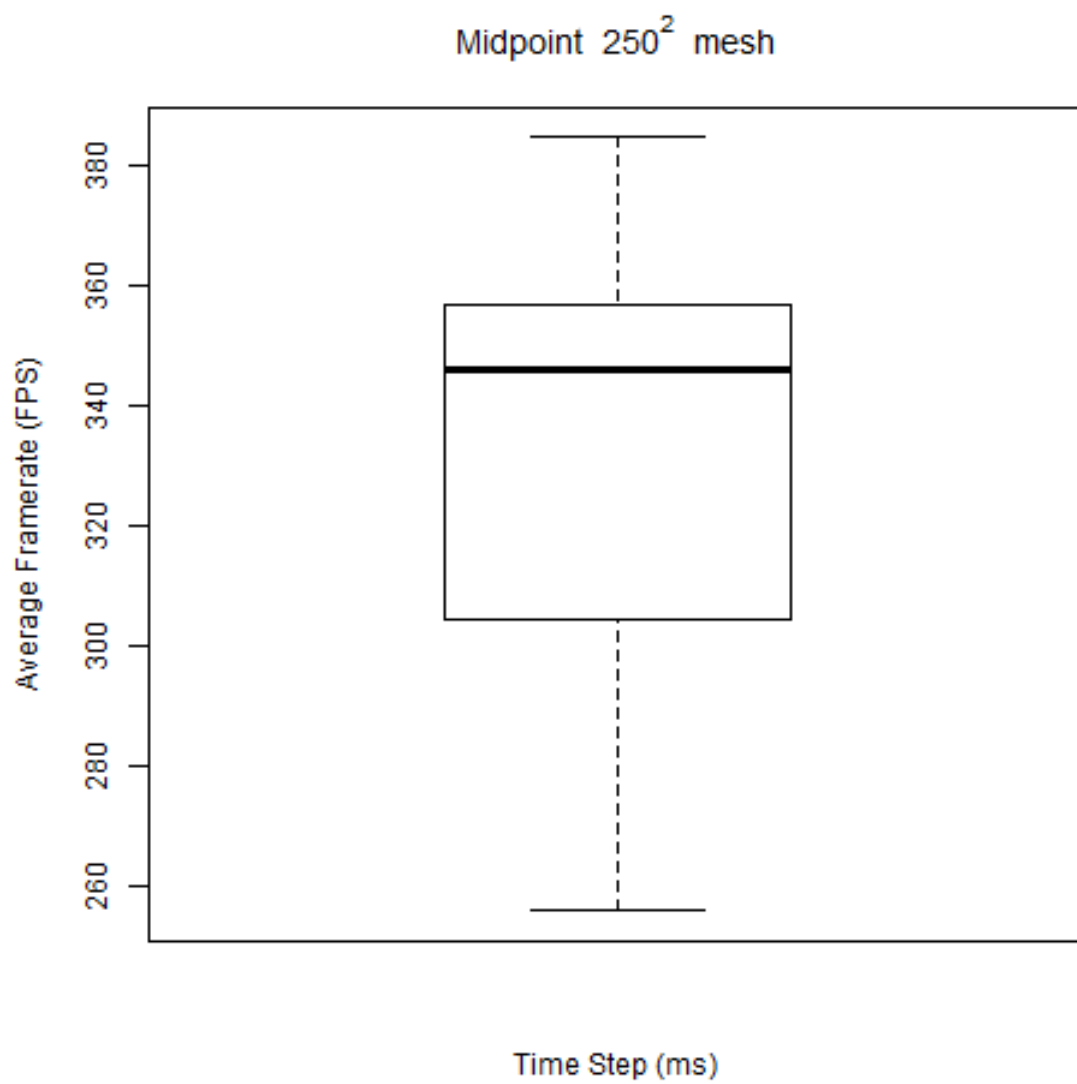
Figure C.64: Midpoint box plot for a 250^2 mesh (sheet)
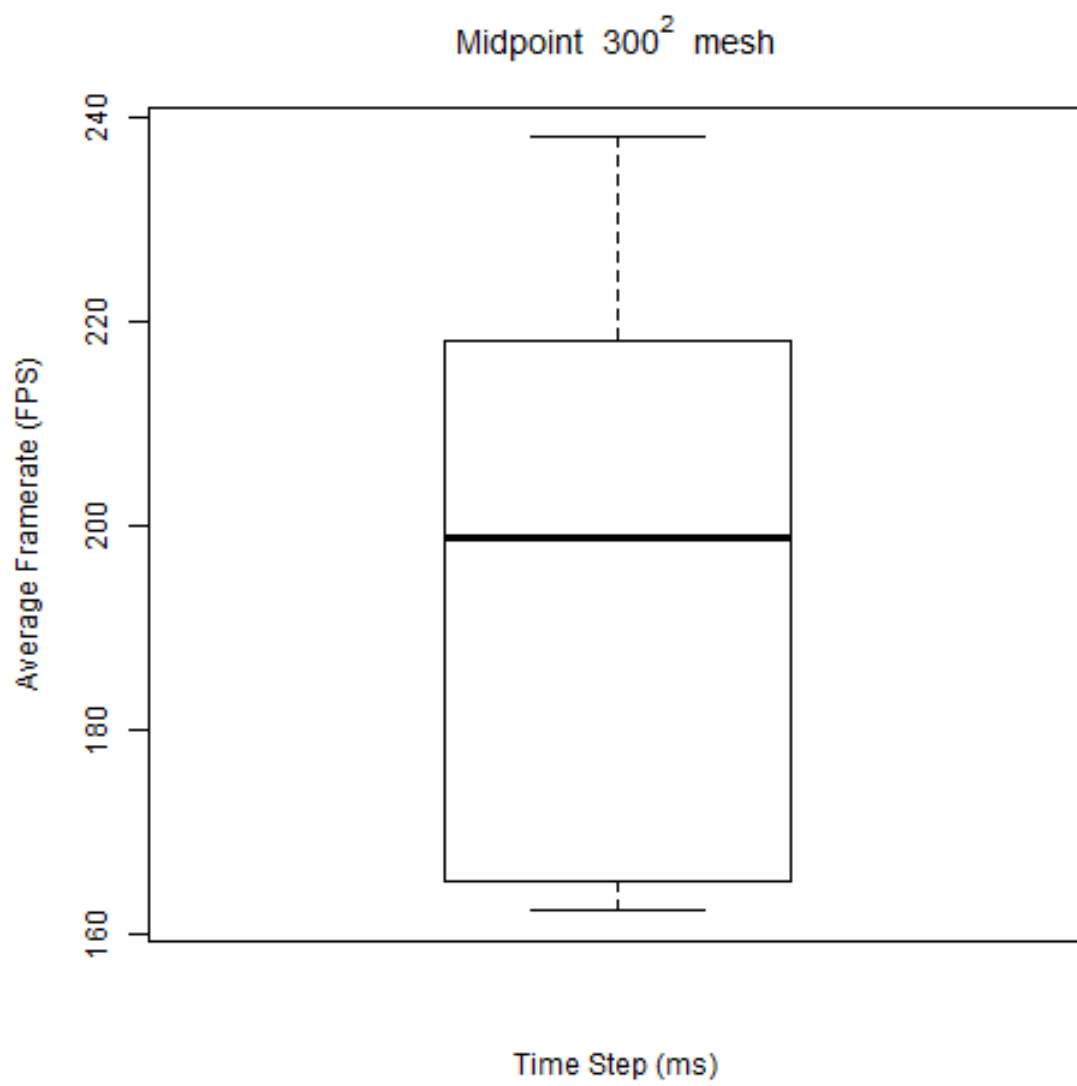
Figure C.65: Midpoint box plot for a 300^2 mesh (sheet)

Mesh Size	Time Step (ms)	Stable/Unstable
50^2	1	Stable
50^2	5	Stable
50^2	10	Stable
50^2	15	Unstable
50^2	20	Unstable
100^2	1	Stable
100^2	5	Unstable
100^2	10	Unstable
100^2	15	Unstable
100^2	20	Unstable
150^2	1	Stable
150^2	5	Unstable
150^2	10	Unstable
150^2	15	Unstable
150^2	20	Unstable
200^2	1	Stable
200^2	5	Unstable
200^2	10	Unstable
200^2	15	Unstable
200^2	20	Unstable
250^2	1	Stable
250^2	5	Unstable
250^2	10	Unstable
250^2	15	Unstable
250^2	20	Unstable
300^2	1	Stable
300^2	5	Unstable
300^2	10	Unstable
300^2	15	Unstable
300^2	20	Unstable

Table C.10: Stability results for Midpoint (sheet)

C.3.2 Flag Data

Rows	Columns	Time Step	Internal force (ms)	External force (ms)	Integrate (ms)	Update (ms)	Render (ms)	FPS	Update count
50	50	1	0.250324	0.12576	0.393067	0.406839	0.0699187	8654.07	58035
50	50	1	0.263874	0.126486	0.407091	0.418503	0.0871101	6908.1	56877
50	50	1	0.256143	0.125913	0.398748	0.410048	0.0817655	7481.2	56568
50	50	1	0.253477	0.126309	0.397033	0.408031	0.0875298	6966.63	57244
50	50	1	0.250235	0.126414	0.393131	0.406858	0.0690474	8726.75	58407
50	50	1	0.250989	0.12606	0.39365	0.407443	0.0670438	8983	58418
50	50	5	0.253896	0.125912	0.396857	0.428647	0.0694878	13100.8	11954
50	50	5	0.269715	0.126472	0.412937	0.442849	0.0884726	10268.7	11924
50	50	5	0.266289	0.126023	0.409065	0.438395	0.080931	11237.3	11926
50	50	5	0.256794	0.126393	0.400454	0.430151	0.0837233	10880.5	11930
50	50	5	0.253711	0.126533	0.396782	0.426895	0.0681945	13358.2	11945
50	50	5	0.254601	0.126179	0.397408	0.42878	0.0665359	13668.4	11945
50	50	10	0.246682	0.125885	0.389616	0.443235	0.0696099	13660.7	5997
50	50	10	0.262891	0.126426	0.406099	0.455501	0.089034	10677.6	5988
50	50	10	0.258832	0.126007	0.401587	0.449691	0.0812125	11723.1	5988
50	50	10	0.249444	0.126384	0.393103	0.442593	0.083125	11454.9	5989
50	50	10	0.246543	0.126576	0.389668	0.439829	0.0679913	13994.8	5995
50	50	10	0.247361	0.126148	0.390124	0.443295	0.0662048	14371.4	5995
50	50	15	0.241846	0.125914	0.384833	0.460402	0.0695201	13870.5	4003
50	50	15	0.257755	0.126531	0.401046	0.470253	0.0886679	10879.1	3998
50	50	15	0.254056	0.126042	0.396878	0.463588	0.0810854	11908.5	3998
50	50	15	0.244521	0.126419	0.388225	0.457024	0.0833283	11592.5	3998
50	50	15	0.24177	0.126555	0.384898	0.455203	0.0680124	14188.5	4001
50	50	15	0.24258	0.126156	0.385362	0.460127	0.0660058	14620.1	4000

50	50	20	0.240966	0.125862	0.383919	0.481191	0.0695279	13964.1	3005
50	50	20	0.257521	0.126499	0.400784	0.488197	0.0886503	10950.9	3001
50	50	20	0.253215	0.125988	0.395988	0.481855	0.0802759	12109.1	3001
50	50	20	0.243596	0.126395	0.387287	0.476366	0.0830279	11707.6	3000
50	50	20	0.240762	0.126631	0.383989	0.474923	0.0678708	14315	3003
50	50	20	0.24157	0.126126	0.384327	0.480946	0.0660511	14708.3	3002
100	100	1	1.016	0.511378	1.6001	1.63269	0.0776168	585.517	35078
100	100	1	1.04164	0.512424	1.62835	1.66108	0.0821445	575.833	34416
100	100	1	1.04713	0.511345	1.63221	1.66511	0.0761495	575.267	34455
100	100	1	1.02672	0.512233	1.61282	1.64541	0.166695	551.783	33108
100	100	1	1.02535	0.512187	1.61133	1.64433	0.0784686	583.233	34824
100	100	1	1.01853	0.511004	1.60314	1.63584	0.0740883	588.033	35086
100	100	5	0.957846	0.511552	1.54224	1.5908	0.0695267	9800.42	11931
100	100	5	0.974119	0.512928	1.56142	1.60875	0.0901581	7536.92	11904
100	100	5	0.977956	0.511686	1.563	1.61171	0.0837571	8092.35	11927
100	100	5	0.967298	0.512095	1.55326	1.59854	0.0934126	7289.7	11909
100	100	5	0.967038	0.512349	1.55328	1.60059	0.0682712	9961.98	11948
100	100	5	0.960558	0.511617	1.54588	1.59417	0.0662808	10263.2	11940
100	100	10	0.954922	0.511988	1.53996	1.61033	0.0694262	12037.5	5994
100	100	10	0.96973	0.512783	1.5568	1.62173	0.0881112	9478.32	5985
100	100	10	0.975075	0.511497	1.55982	1.62567	0.0832041	10034.1	5987
100	100	10	0.964244	0.512717	1.55086	1.61324	0.094043	8896.03	5983
100	100	10	0.96353	0.512636	1.55009	1.6173	0.0680956	12261.8	5992
100	100	10	0.957115	0.511427	1.54212	1.61213	0.0660618	12650.5	5993
100	100	15	0.952687	0.511642	1.53746	1.62994	0.0691441	12824.5	4002
100	100	15	0.968094	0.512894	1.55533	1.6401	0.0864234	10253.6	3997

100	100	15	0.972538	0.511464	1.55773	1.64167	0.0823294	10743.9	3997
100	100	15	0.961817	0.512528	1.5485	1.62908	0.0919632	9649.05	3995
100	100	15	0.9612	0.512152	1.54756	1.63485	0.0681158	12984.4	4000
100	100	15	0.954823	0.511492	1.53987	1.63168	0.0664528	13315	4000
100	100	20	0.924413	0.497228	1.49477	1.6092	0.0696846	13130.1	3004
100	100	20	0.940909	0.499056	1.51441	1.62029	0.0889613	10298.9	3000
100	100	20	0.944749	0.497741	1.51648	1.63676	0.0782533	11680.9	3003
100	100	20	0.932457	0.497447	1.50403	1.61041	0.0843202	10869.6	3000
100	100	20	0.932471	0.497874	1.50445	1.6128	0.0685199	13358.4	3002
100	100	20	0.926371	0.497178	1.49725	1.61033	0.0667117	13726.1	3002
150	150	1	2.28445	1.15755	3.60645	3.6772	0.0949903	266.133	15905
150	150	1	2.31855	1.16047	3.64701	3.72382	0.0997285	263	15692
150	150	1	2.32922	1.15685	3.6526	3.72869	0.0919263	262.583	15704
150	150	1	2.3101	1.15874	3.63587	3.70767	0.185815	258.333	15410
150	150	1	2.30691	1.15925	3.63275	3.70433	0.0978684	264.633	15780
150	150	1	2.29045	1.15862	3.61564	3.68689	0.0963423	266.067	15859
150	150	5	2.16158	1.15732	3.48362	3.56215	0.0697164	4207.15	11930
150	150	5	2.16938	1.16187	3.49811	3.58341	0.0905664	3205.87	11921
150	150	5	2.18725	1.15936	3.51372	3.59641	0.0845732	3392.43	11919
150	150	5	2.18251	1.15878	3.50842	3.58733	0.088621	3256.82	11889
150	150	5	2.18116	1.15893	3.50715	3.58622	0.0695061	4119.45	11930
150	150	5	2.16677	1.15858	3.49208	3.57089	0.0668727	4319.47	11940
150	150	10	2.15935	1.15844	3.48267	3.584	0.0692105	9171.32	5991
150	150	10	2.16246	1.1609	3.49105	3.60422	0.0886197	7184.22	5987
150	150	10	2.18226	1.1596	3.508	3.61273	0.0840853	7554.75	5983
150	150	10	2.18066	1.15928	3.50718	3.60488	0.0861333	7404.97	5983

150	150	10	2.17737	1.15909	3.50344	3.60254	0.0682913	9324.27	5990
150	150	10	2.16416	1.15868	3.48932	3.59028	0.0664829	9598.52	5990
150	150	15	2.114	1.12355	3.40255	3.52632	0.0693357	10984	4000
150	150	15	2.1204	1.12656	3.4147	3.53764	0.0888025	8581.87	3996
150	150	15	2.13469	1.12473	3.42542	3.54741	0.0839871	9080.31	3995
150	150	15	2.13228	1.12416	3.42373	3.53783	0.0942898	8086.63	3993
150	150	15	2.13254	1.12493	3.42475	3.54677	0.0681858	11159.8	3999
150	150	15	2.1191	1.12437	3.41004	3.53399	0.0668134	11403.1	3999
150	150	20	2.11701	1.1234	3.4055	3.55196	0.0690486	11858.2	3003
150	150	20	2.12346	1.12626	3.4175	3.56083	0.0884955	9259.48	2998
150	150	20	2.13554	1.12444	3.42612	3.56675	0.0834731	9811.77	2999
150	150	20	2.13264	1.12349	3.42326	3.55613	0.0932123	8796.85	2998
150	150	20	2.13217	1.12438	3.42371	3.56579	0.0682587	11993.8	3000
150	150	20	2.12117	1.12426	3.41198	3.5589	0.0662982	12356	3000
200	200	1	4.05151	2.06799	6.41331	6.54833	0.147264	150.133	8961
200	200	1	4.06667	2.07256	6.43815	6.58389	0.156276	149.267	8902
200	200	1	4.09554	2.07118	6.46302	6.59606	0.143822	148.783	8902
200	200	1	4.0864	2.06995	6.45456	6.58764	0.229686	147.417	8801
200	200	1	4.08538	2.07148	6.45485	6.58885	0.149558	149.367	8904
200	200	1	4.05605	2.07019	6.42295	6.55582	0.145172	150.2	8954
200	200	5	3.88455	2.06822	6.24677	6.38176	0.146068	168.85	9190
200	200	5	3.90015	2.07252	6.27186	6.416	0.150656	165.083	9136
200	200	5	3.92895	2.07143	6.29714	6.42992	0.139856	158.45	9132
200	200	5	3.92142	2.07124	6.291	6.4242	0.229231	161.833	9016
200	200	5	3.91707	2.07073	6.28585	6.41934	0.147539	169.433	9135
200	200	5	3.8943	2.06982	6.26121	6.39478	0.14262	172.117	9177

200	200	10	3.83501	2.00893	6.13781	6.29168	0.0701789	5322.8	5987
200	200	10	3.84496	2.01366	6.15795	6.3232	0.0907312	4096.49	5979
200	200	10	3.89995	2.0129	6.21027	6.36191	0.0844747	4336.1	5981
200	200	10	3.855	2.01001	6.16322	6.31175	0.0937857	3978.15	5977
200	200	10	3.86146	2.01099	6.17031	6.32363	0.069343	5305.82	5987
200	200	10	3.83332	2.00996	6.14005	6.29186	0.0677821	5475.53	5986
200	200	15	3.84115	2.0068	6.14243	6.32135	0.0695814	8287.48	3998
200	200	15	3.85842	2.01184	6.16987	6.35798	0.090261	6371.98	3994
200	200	15	3.9812	2.01512	6.2945	6.47197	0.0844421	6735.12	3994
200	200	15	3.85296	2.00742	6.15851	6.32799	0.0920239	6273.9	3992
200	200	15	3.86403	2.00983	6.17277	6.34796	0.068937	8349.1	3996
200	200	15	3.84572	2.00839	6.15087	6.32701	0.0675042	8546.7	3996
200	200	20	3.84799	2.00822	6.15069	6.35083	0.0694234	9794.25	3001
200	200	20	3.88522	2.01309	6.19883	6.40581	0.0903106	7506.08	2998
200	200	20	4.05448	2.01512	6.36721	6.56585	0.0842863	7956.66	2999
200	200	20	3.86113	2.00819	6.16718	6.35314	0.0917283	7422.68	2997
200	200	20	3.87451	2.01001	6.18375	6.37892	0.0687634	9872.1	3000
200	200	20	3.85674	2.00984	6.1663	6.36431	0.0674003	10082.2	2999
250	250	1	6.37679	3.2468	10.109	10.3522	0.191843	95.3	5691
250	250	1	6.37473	3.25401	10.1194	10.3593	0.200498	95.15	5682
250	250	1	6.45038	3.26262	10.1977	10.4369	0.189959	94.2833	5646
250	250	1	6.43116	3.25006	10.1627	10.3987	0.272615	93.9667	5623
250	250	1	6.43916	3.25288	10.1805	10.4185	0.193961	94.7333	5654
250	250	1	6.39613	3.25065	10.1287	10.3614	0.18879	95.25	5687
250	250	5	6.13838	3.24616	9.86791	10.1096	0.186266	98.6	5827
250	250	5	6.15145	3.25784	9.90624	10.1513	0.200022	97.8	5796

250	250	5	6.2098	3.26103	9.95885	10.2022	0.183709	97.3	5776
250	250	5	6.19345	3.24972	9.92832	10.1664	0.268627	96.4833	5749
250	250	5	6.19263	3.25187	9.93782	10.1808	0.191296	97.8833	5784
250	250	5	6.14833	3.25005	9.88017	10.1158	0.188821	98.4667	5822
250	250	10	6.05464	3.15129	9.69239	9.93718	0.18581	146.15	5925
250	250	10	6.09479	3.16101	9.74733	9.99353	0.196641	133.95	5886
250	250	10	6.10762	3.16455	9.76099	10.0034	0.186327	111.217	5886
250	250	10	6.08469	3.15107	9.72146	9.95701	0.270264	121.833	5864
250	250	10	6.07934	3.14764	9.714	9.95343	0.188571	148.65	5914
250	250	10	6.04871	3.1551	9.68465	9.91919	0.194786	145.233	5932
250	250	15	6.07923	3.15094	9.71535	9.98381	0.0714236	4723.68	3992
250	250	15	6.3608	3.1589	10.0096	10.2765	0.0881034	3610.08	3990
250	250	15	6.42904	3.16191	10.0782	10.3425	0.0827037	3777.18	3990
250	250	15	6.12294	3.15825	9.77535	10.0457	0.0905612	3678.38	3989
250	250	15	6.11026	3.15211	9.75305	10.0171	0.0711135	4713.13	3992
250	250	15	6.08041	3.15698	9.73282	10.004	0.069576	4829.47	3992
250	250	20	6.1029	3.15954	9.75895	10.0546	0.0731211	6791.73	2993
250	250	20	6.50388	3.35742	10.3936	10.6873	0.085818	5412.33	2997
250	250	20	6.59452	3.35132	10.4748	10.7572	0.0807661	5727	2996
250	250	20	6.12889	3.15943	9.77725	10.0561	0.0888105	5609.82	2995
250	250	20	6.13853	3.15477	9.7918	10.0792	0.0730449	6783.95	2993
250	250	20	6.09146	3.15858	9.73549	10.0166	0.0731645	6828.26	2993
300	300	1	9.8688	4.85278	15.5671	16.0347	0.205619	61.8	3695
300	300	1	9.95157	4.87093	15.6854	16.16	0.21955	61.25	3663
300	300	1	9.91789	4.86489	15.6282	16.0906	0.207135	61.75	3682
300	300	1	9.93443	4.86022	15.6426	16.1095	0.291313	61.35	3659

300	300	1	9.9073	4.83918	15.5899	16.0527	0.211357	61.7	3689
300	300	1	9.82978	4.8452	15.5129	15.9754	0.212438	61.9333	3707
300	300	5	9.56952	4.85648	15.2711	15.742	0.206101	63.8833	3762
300	300	5	9.63413	4.87158	15.3644	15.8389	0.215141	63.3167	3737
300	300	5	9.63564	4.86923	15.354	15.8219	0.204676	63.6333	3743
300	300	5	9.62999	4.85505	15.3306	15.7973	0.29011	62.9333	3730
300	300	5	9.59217	4.83896	15.2776	15.7433	0.208365	63.8833	3761
300	300	5	9.53383	4.84674	15.2216	15.6812	0.203704	64.05	3777
300	300	10	9.40462	4.71024	14.9688	15.444	0.205469	65.7167	3833
300	300	10	9.4633	4.72632	15.0497	15.5235	0.217018	65.3333	3811
300	300	10	9.44604	4.71616	15.0095	15.4743	0.202947	65.7667	3826
300	300	10	9.45205	4.70348	14.9963	15.4625	0.287514	64.9	3808
300	300	10	9.41038	4.69425	14.9549	15.4188	0.208002	65.7833	3839
300	300	10	9.36126	4.6954	14.8901	15.3548	0.205297	66.0333	3855
300	300	15	9.41312	4.70833	14.9775	15.4515	0.204199	91.65	3831
300	300	15	9.46025	4.72652	15.0379	15.5085	0.216049	83.65	3814
300	300	15	9.4413	4.71137	14.9881	15.449	0.204754	103.883	3832
300	300	15	9.45177	4.70725	15.0022	15.4657	0.283629	71.15	3808
300	300	15	9.42075	4.69742	14.9609	15.4269	0.206688	87.3333	3836
300	300	15	9.35615	4.69865	14.8962	15.3579	0.211088	84.35	3854
300	300	20	9.42714	4.70935	15.0032	15.5013	0.0724949	3135.83	2995
300	300	20	9.83908	4.72809	15.4329	15.9303	0.0871718	2367.48	2994
300	300	20	9.83518	4.74764	15.4299	15.9168	0.0808027	2541.42	2994
300	300	20	9.45641	4.6977	14.9999	15.487	0.0910794	2498.07	2994
300	300	20	9.43881	4.69862	14.9995	15.4827	0.0715792	3184.7	2995
300	300	20	9.38256	4.70838	14.9637	15.4556	0.0703518	3256.67	2995

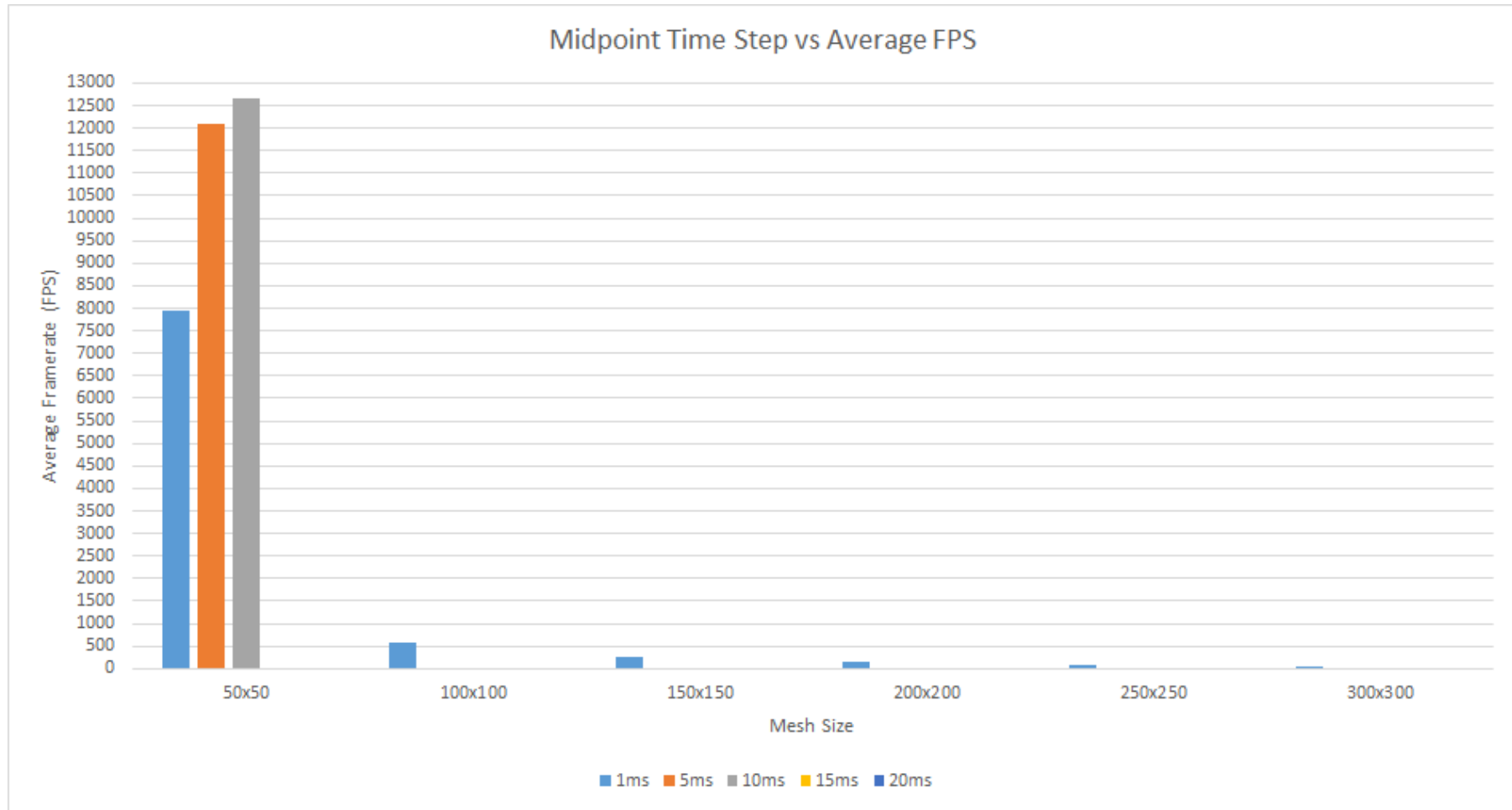Table C.11: Raw data for Midpoint (flag)

Figure C.66: Midpoint mesh size against average FPS (flag)
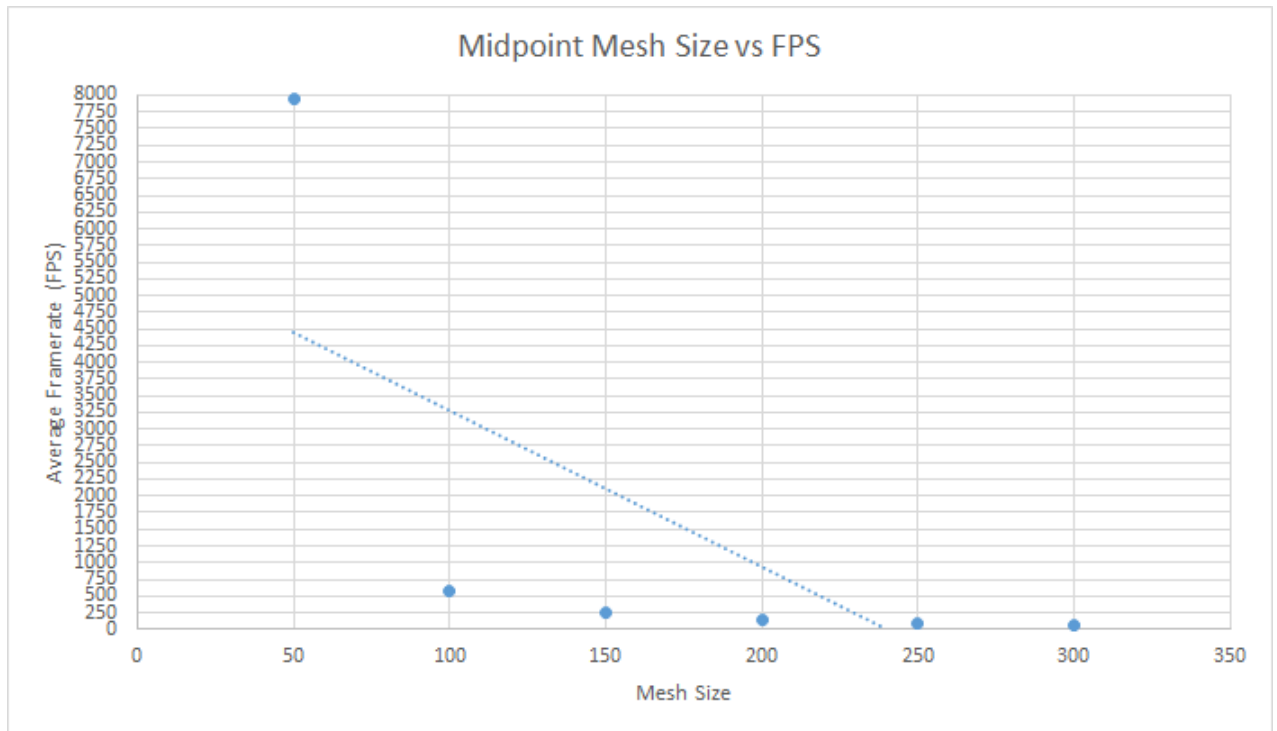
Figure C.67: Midpoint mesh size against average FPS (flag)
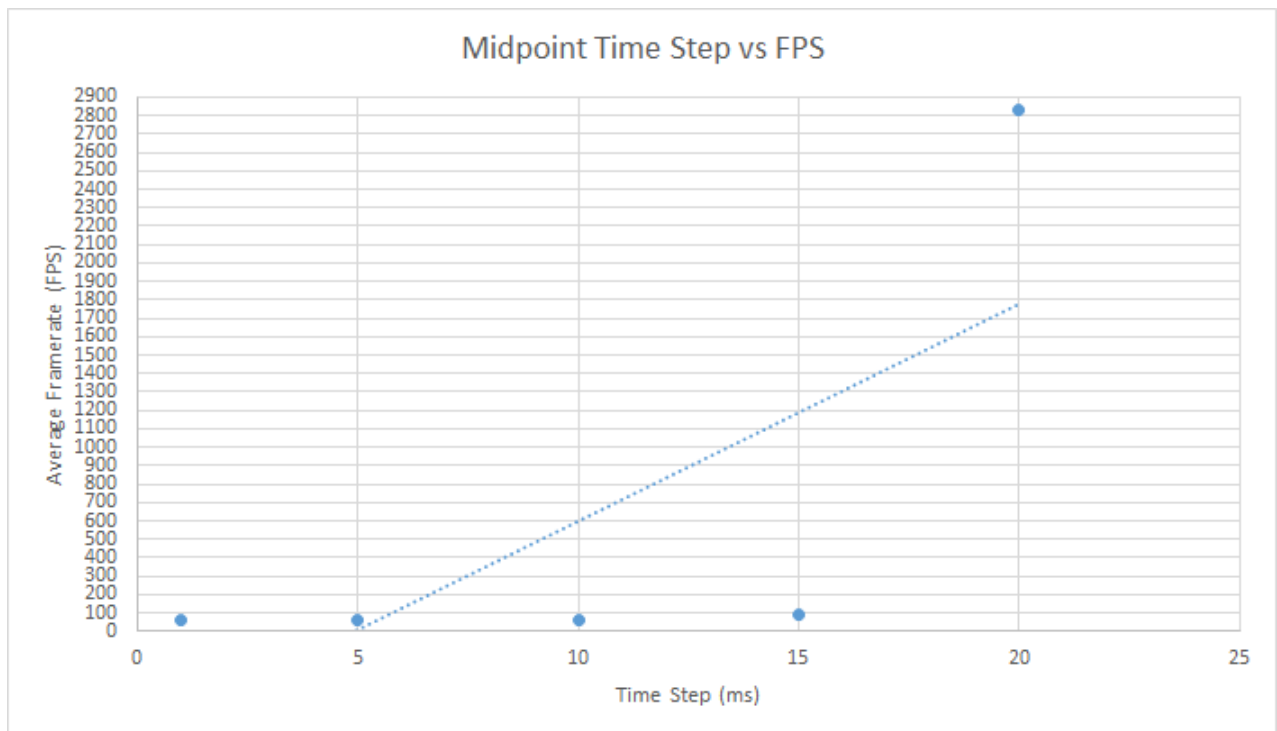
Figure C.68: Midpoint time step against average FPS for a 300^2 mesh (flag)

178
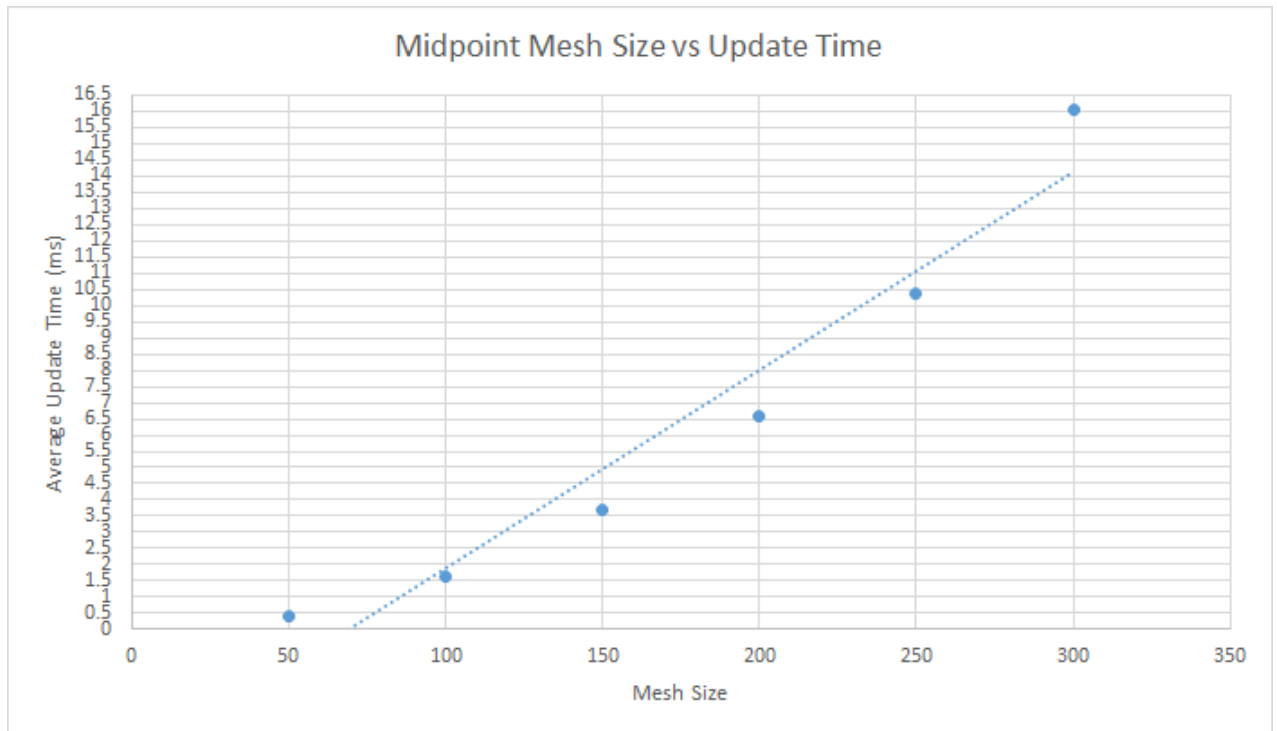
Figure C.69: Midpoint mesh size against average update time (flag)
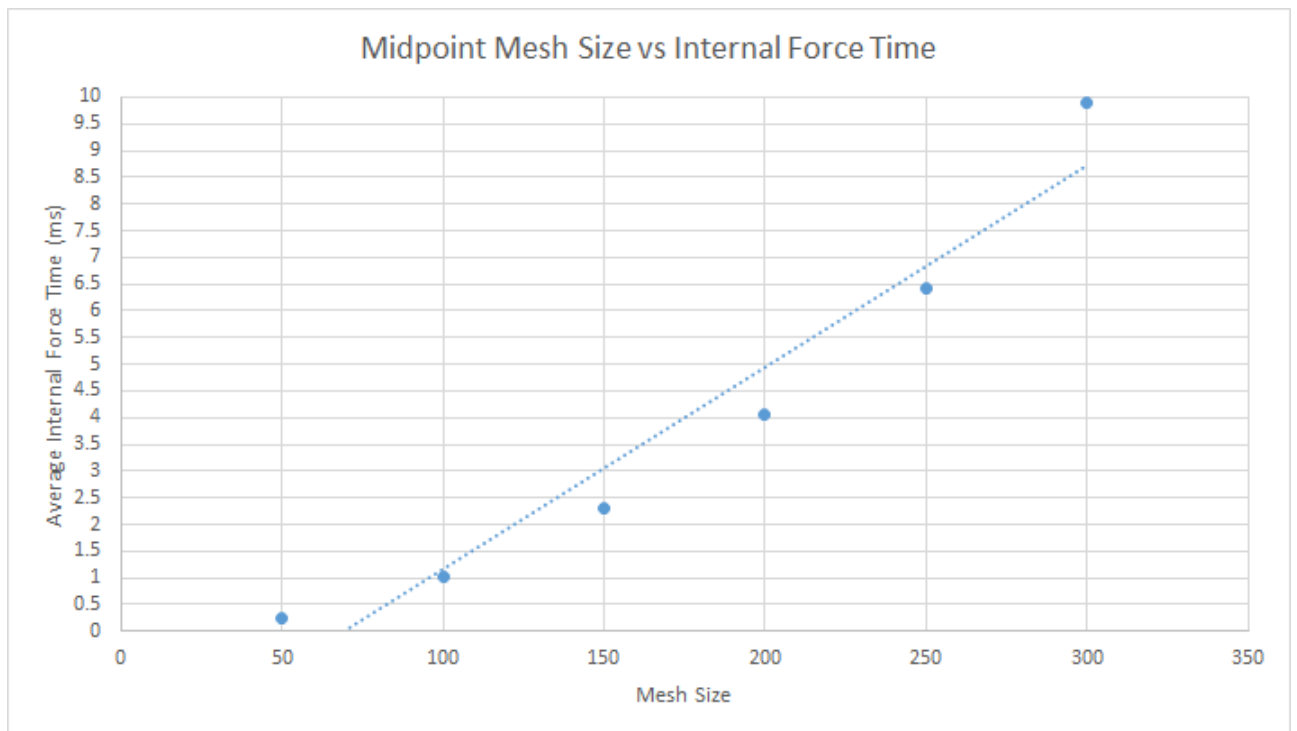
Figure C.70: Midpoint mesh size against average internal force time (flag)

179
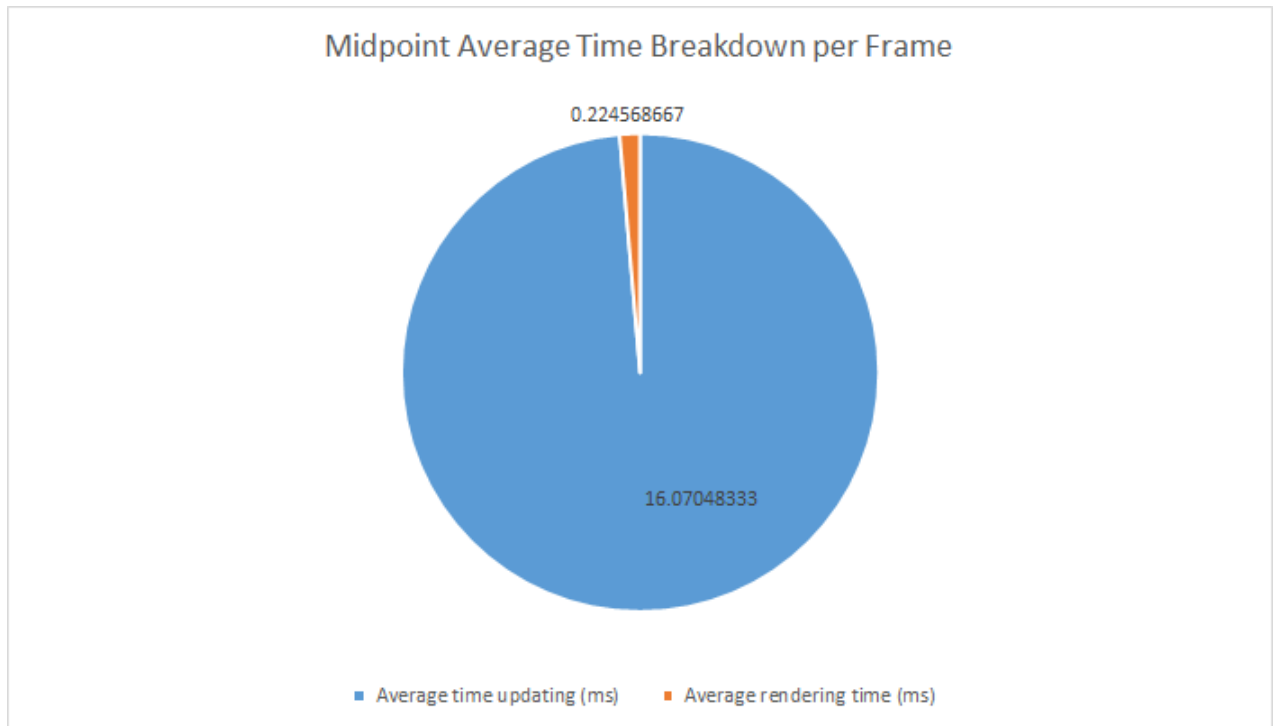
Figure C.71: Midpoint frame time breakdown (flag)
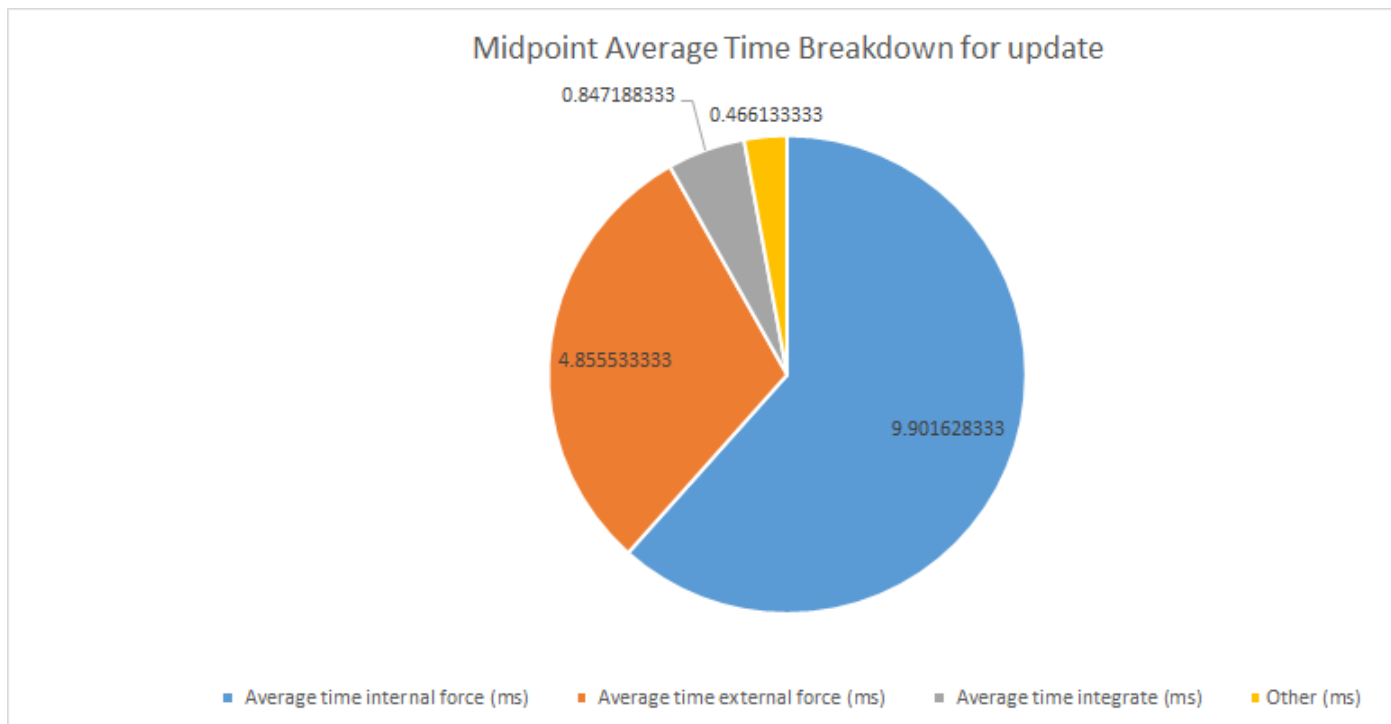
Figure C.72: Midpoint update time breakdown (flag)
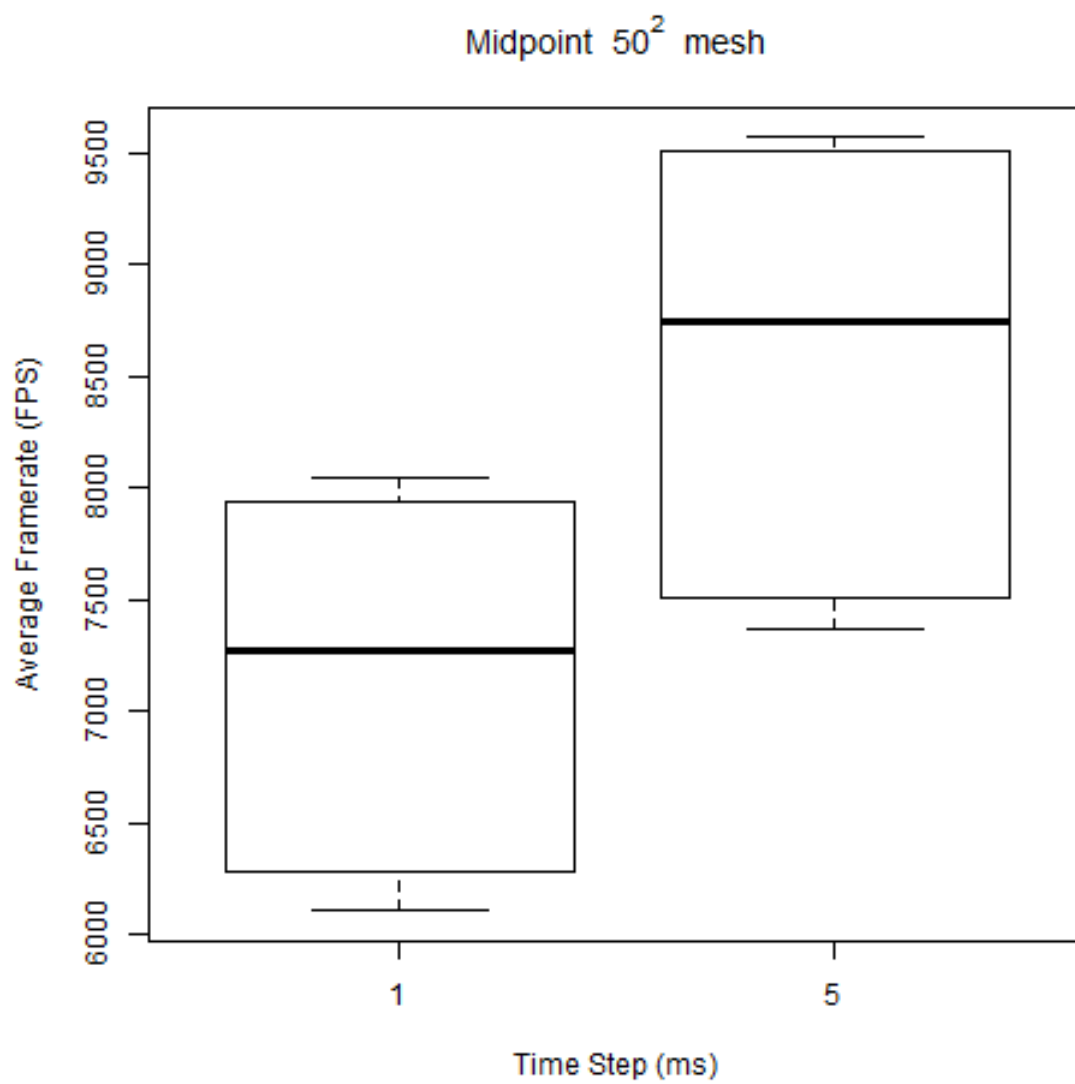
Figure C.73: Midpoint box plot for a 50^2 mesh (flag)
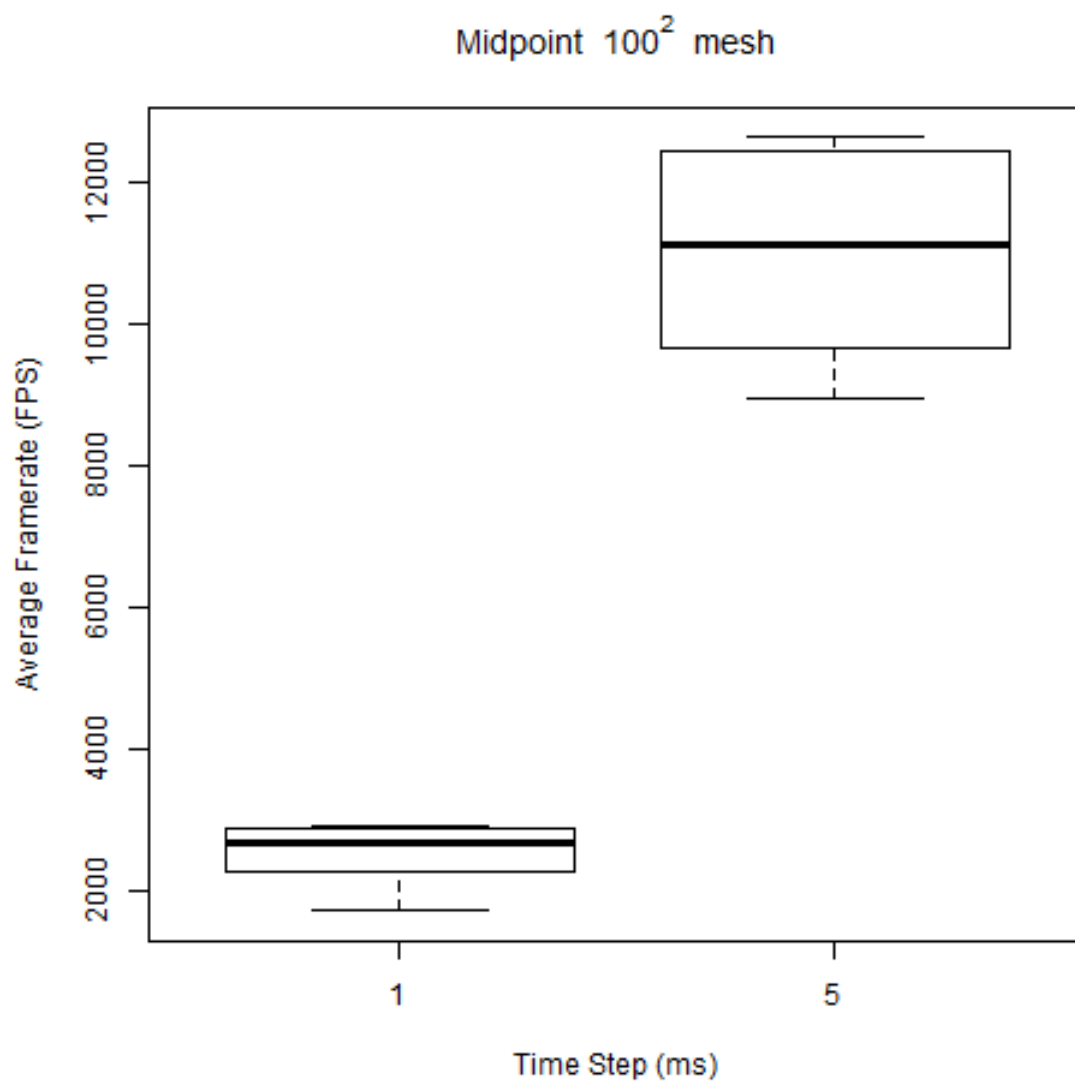
Figure C.74: Midpoint box plot for a 100^2 mesh (flag)
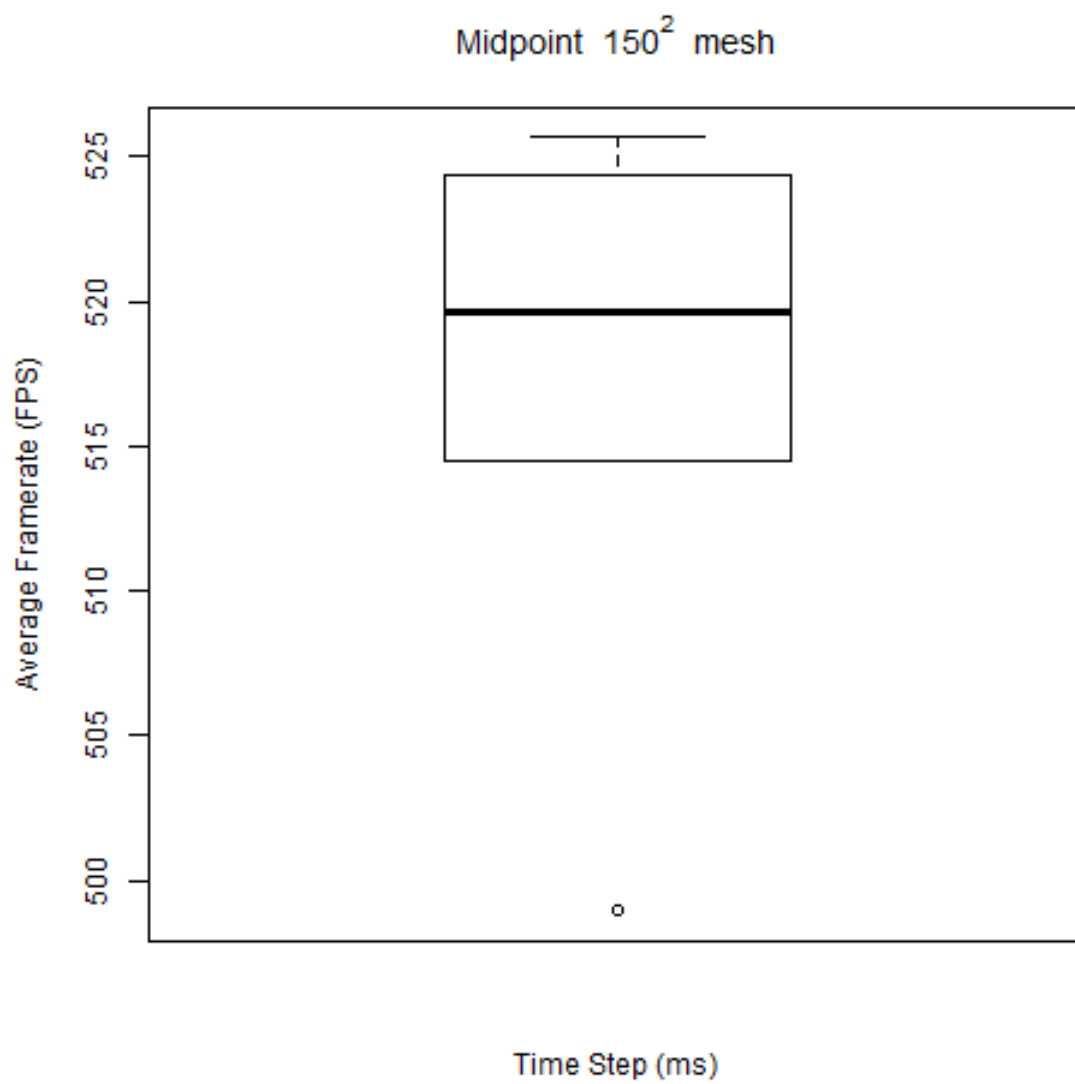
Figure C.75: Midpoint box plot for a 150^2 mesh (flag)
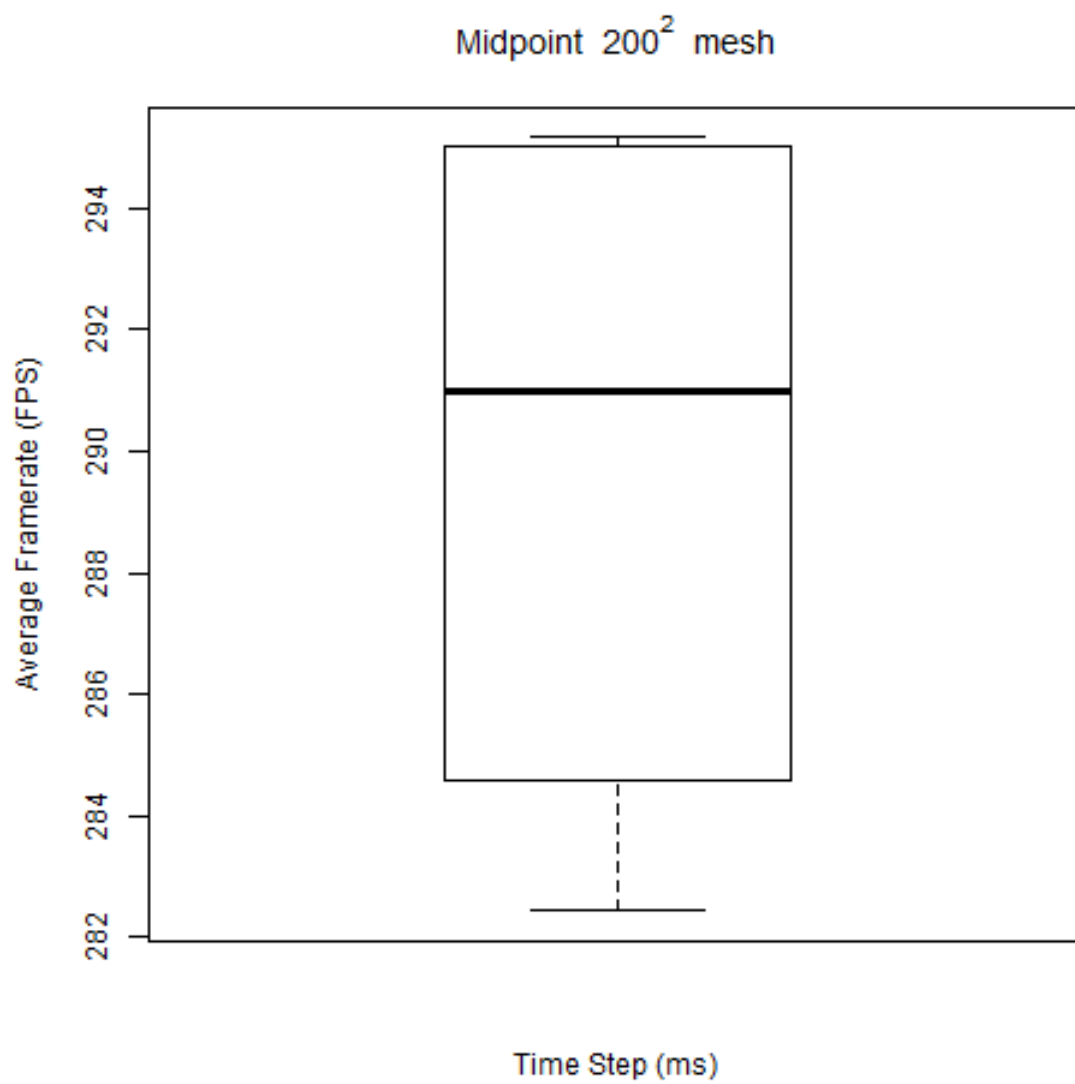
Figure C.76: Midpoint box plot for a 200^2 mesh (flag)
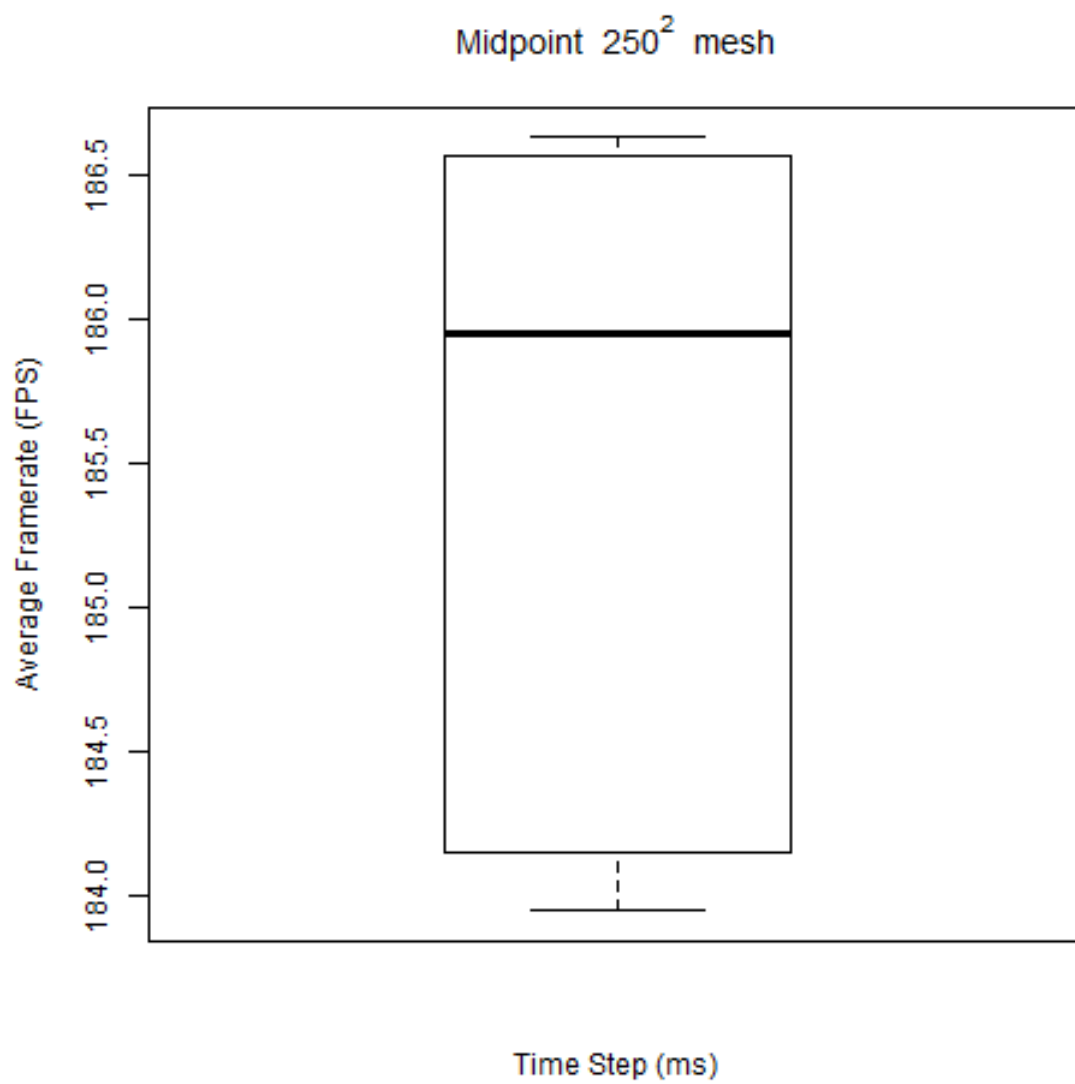
Figure C.77: Midpoint box plot for a 250^2 mesh (flag)
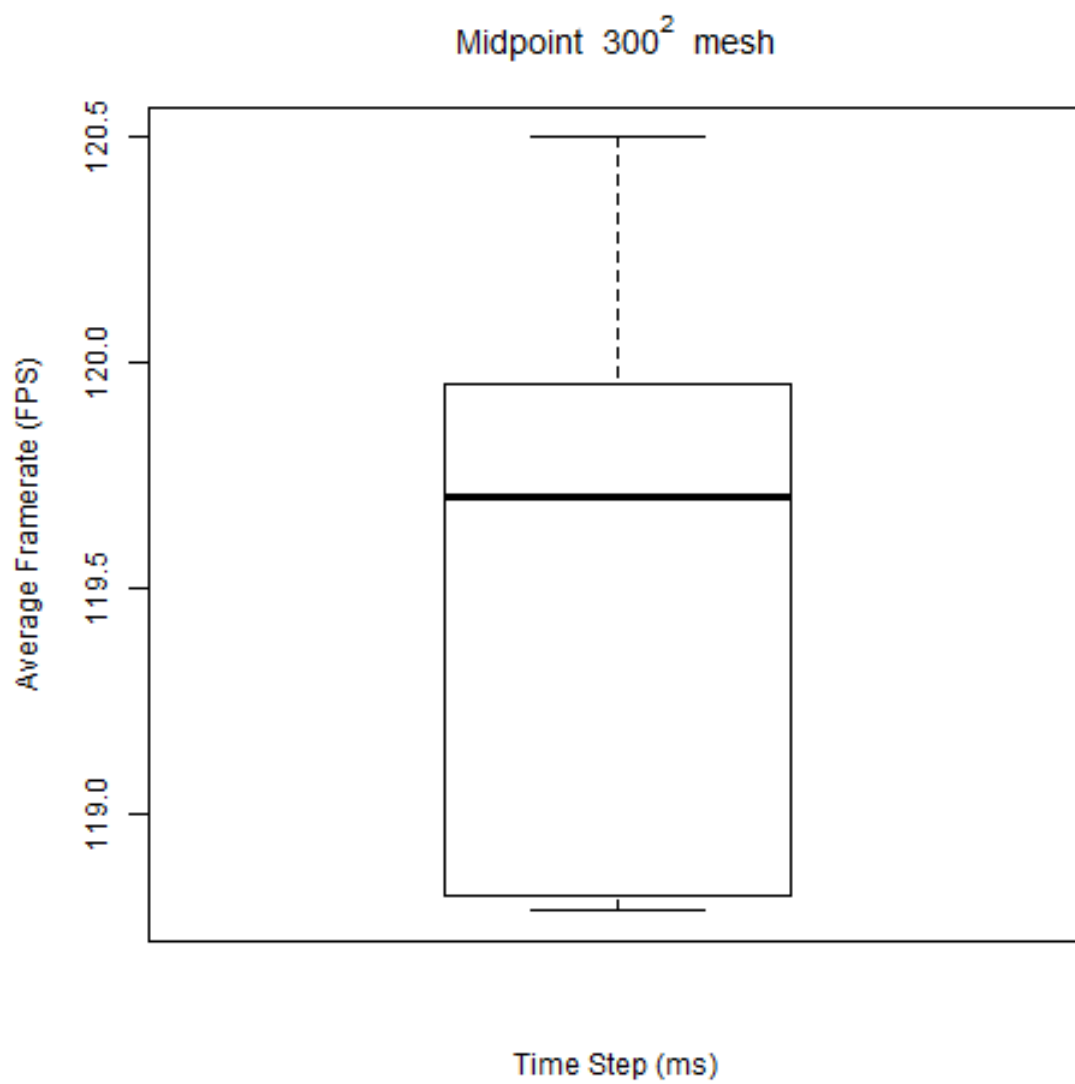
Figure C.78: Midpoint box plot for a 300^2 mesh (flag)

Mesh Size	Time Step (ms)	Stable/Unstable
50^2	1	Stable
50^2	5	Stable
50^2	10	Stable
50^2	15	Unstable
50^2	20	Unstable
100^2	1	Stable
100^2	5	Unstable
100^2	10	Unstable
100^2	15	Unstable
100^2	20	Unstable
150^2	1	Stable
150^2	5	Unstable
150^2	10	Unstable
150^2	15	Unstable
150^2	20	Unstable
200^2	1	Stable
200^2	5	Unstable
200^2	10	Unstable
200^2	15	Unstable
200^2	20	Unstable
250^2	1	Stable
250^2	5	Unstable
250^2	10	Unstable
250^2	15	Unstable
250^2	20	Unstable
300^2	1	Stable
300^2	5	Unstable
300^2	10	Unstable
300^2	15	Unstable
300^2	20	Unstable

Table C.12: Stability results for Midpoint (flag)

C.4   Fourth Order Runge-Kutta

C.4.1   Sheet Data

Rows	Columns	Time Step	Internal force (ms)	External force (ms)	Integrate (ms)	Update (ms)	Render (ms)	FPS	Update count
50	50	1	0.515794	0.0203216	0.597012	0.615732	0.070396	5746.13	57864
50	50	1	0.515431	0.0180531	0.593809	0.62018	0.0866712	4733.27	56970
50	50	1	0.516575	0.0207708	0.598574	0.618239	0.0812523	4973.83	57738
50	50	1	0.510251	0.0181419	0.588645	0.607129	0.0899682	4636.07	57601
50	50	1	0.51075	0.0182033	0.588976	0.608034	0.069348	5841.25	58563
50	50	1	0.503088	0.0205113	0.584464	0.603852	0.0674901	6057.43	58560
50	50	5	0.660857	0.0203728	0.742283	0.780792	0.0697323	11906.7	11953
50	50	5	0.666266	0.0181019	0.744818	0.787423	0.0878047	9519.53	11914
50	50	5	0.667426	0.0208131	0.749617	0.786625	0.0811447	10275.6	11925
50	50	5	0.657605	0.020347	0.739438	0.776417	0.0835344	10083.5	11914
50	50	5	0.660229	0.0182843	0.73887	0.776509	0.0686426	12200.9	11943
50	50	5	0.652813	0.0205637	0.734389	0.773186	0.0665784	12524	11943
50	50	10	0.599214	0.0203733	0.680669	0.741465	0.0693701	13274.7	5997
50	50	10	0.601846	0.0180681	0.680341	0.743905	0.0885093	10411.2	5986
50	50	10	0.603979	0.0208348	0.686204	0.743311	0.079659	11550.7	5989
50	50	10	0.595387	0.0203735	0.677283	0.734399	0.0828679	11131.7	5988
50	50	10	0.598256	0.0182795	0.676924	0.735408	0.0682147	13498.9	5994
50	50	10	0.590941	0.020565	0.672507	0.733533	0.0663334	13877.5	5994
50	50	15	0.570262	0.0203732	0.651731	0.734676	0.0694205	13623.8	4003
50	50	15	0.578528	0.018102	0.657116	0.736261	0.084438	11196.3	3998
50	50	15	0.576823	0.020857	0.65909	0.733457	0.0820544	11547.6	3998
50	50	15	0.565896	0.0203343	0.647772	0.723867	0.0837897	11309.6	3997

50	50	15	0.569374	0.018269	0.648036	0.727051	0.0683137	13852.6	4001
50	50	15	0.561846	0.0205535	0.643427	0.726061	0.0661558	14317.3	4001
50	50	20	0.451825	0.0203692	0.533444	0.638491	0.0702233	13710.6	3004
50	50	20	0.45054	0.0181082	0.529131	0.631743	0.0903233	10674.4	3001
50	50	20	0.444098	0.0208226	0.526251	0.667411	0.0846888	11348.3	3003
50	50	20	0.447476	0.0203221	0.529373	0.626126	0.0834204	11569.9	3000
50	50	20	0.451713	0.018272	0.530459	0.628866	0.0687414	14022	3002
50	50	20	0.44281	0.0205781	0.524429	0.629043	0.0670039	14386.2	3002
100	100	1	2.50595	0.099778	2.85651	2.92154	0.0824895	399.933	19973
100	100	1	2.5191	0.100518	2.87618	2.94086	0.0868948	430.6	19816
100	100	1	2.61767	0.100445	2.97473	3.0385	0.0802816	348.233	19238
100	100	1	2.49007	0.0997844	2.84133	2.90695	0.17499	344.65	19468
100	100	1	2.48969	0.0997909	2.84214	2.90672	0.0810852	457.567	20081
100	100	1	2.47592	0.100412	2.8279	2.89295	0.0774185	534.083	20199
100	100	5	2.16067	0.0997316	2.51178	2.59067	0.0696514	6898.92	11935
100	100	5	2.15328	0.100648	2.51225	2.60455	0.0900559	5293.77	11905
100	100	5	2.13908	0.100857	2.49595	2.58585	0.085315	5672.02	11910
100	100	5	2.14514	0.0997376	2.49652	2.57272	0.0911042	5351.5	11897
100	100	5	2.14565	0.0997773	2.49786	2.57665	0.0686431	7017.4	11932
100	100	5	2.1311	0.100448	2.48251	2.56191	0.0667178	7226.07	11935
100	100	10	1.85749	0.0997542	2.20972	2.31351	0.0700044	10879.5	5994
100	100	10	1.84873	0.100672	2.20852	2.32034	0.0890114	8536.23	5982
100	100	10	1.84036	0.100785	2.19648	2.30889	0.0826864	9237.18	5986
100	100	10	1.8318	0.099604	2.18314	2.28144	0.0829319	9256.65	5985
100	100	10	1.83599	0.0998241	2.18961	2.28962	0.0689629	11058.7	5990
100	100	10	1.82457	0.100448	2.1773	2.28117	0.067141	11342.9	5990

100	100	15	1.85767	0.0997777	2.21022	2.33727	0.070319	11916.8	3999
100	100	15	1.8468	0.100643	2.20705	2.33887	0.088934	9428.85	3994
100	100	15	1.97987	0.100713	2.33973	2.47328	0.0828257	10026.3	3995
100	100	15	1.83852	0.0996524	2.19073	2.30939	0.0834722	10084.6	3994
100	100	15	1.83717	0.0998415	2.18999	2.31191	0.0694307	12097	3997
100	100	15	1.82194	0.100468	2.17497	2.30163	0.0676417	12455.4	3996
100	100	20	1.8576	0.0998562	2.21136	2.36101	0.0706031	12428	3001
100	100	20	1.84746	0.100677	2.20827	2.36007	0.0892844	9837	2997
100	100	20	1.97925	0.100845	2.34017	2.49918	0.0832888	10445.1	2999
100	100	20	1.83328	0.099756	2.18636	2.32643	0.0834619	10531.5	2998
100	100	20	1.83764	0.099814	2.19162	2.33479	0.0691435	12713.7	3000
100	100	20	1.82285	0.100497	2.17666	2.3267	0.0674418	13034.7	2999
150	150	1	5.2294	0.226223	6.03911	6.19558	0.141718	245.85	9468
150	150	1	5.23797	0.228554	6.07154	6.24394	0.149719	231.517	9385
150	150	1	5.49826	0.22877	6.3325	6.49808	0.14448	175.483	9033
150	150	1	5.19719	0.226681	6.01695	6.16537	0.229536	310.517	9382
150	150	1	5.19654	0.226834	6.01064	6.16588	0.147037	256.133	9505
150	150	1	5.16129	0.228194	5.97785	6.13441	0.140601	311.917	9562
150	150	5	4.75486	0.226307	5.56895	5.72149	0.146637	170.617	10224
150	150	5	4.78225	0.227993	5.61303	5.78474	0.146933	168.6	10115
150	150	5	4.73035	0.229148	5.55182	5.71507	0.138147	171.217	10250
150	150	5	4.72375	0.226687	5.53984	5.68865	0.221468	168.767	10151
150	150	5	4.72554	0.22656	5.54151	5.69834	0.14537	171.283	10267
150	150	5	4.68903	0.22803	5.5063	5.65735	0.139897	172.45	10349
150	150	10	4.25378	0.22645	5.07025	5.25351	0.0711492	6618.03	5977
150	150	10	4.23028	0.228928	5.06875	5.27564	0.0908374	5166.6	5970

150	150	10	4.26618	0.22851	5.09398	5.29495	0.0870958	5398.75	5971
150	150	10	4.21729	0.226842	5.03543	5.21043	0.0907472	5293.32	5967
150	150	10	4.21249	0.226666	5.036	5.22254	0.0704446	6725.27	5975
150	150	10	4.18832	0.228108	5.01096	5.19256	0.0686706	6913.6	5976
150	150	15	4.26226	0.226872	5.08241	5.29176	0.0707201	9064.7	3991
150	150	15	4.22908	0.228636	5.08174	5.31485	0.0900296	7112.7	3986
150	150	15	4.25826	0.228408	5.08844	5.31821	0.0861542	7458.1	3986
150	150	15	4.21109	0.226691	5.032	5.23187	0.095577	6797.62	3986
150	150	15	4.21971	0.22716	5.0472	5.25955	0.0693648	9272.8	3990
150	150	15	4.19307	0.22824	5.01878	5.22689	0.0677875	9505.28	3989
150	150	20	4.26922	0.227442	5.09487	5.33199	0.0699951	10383.8	3000
150	150	20	4.22822	0.228596	5.07111	5.32563	0.0894519	8139.7	2996
150	150	20	4.19762	0.228874	5.03206	5.27875	0.0844106	8671.82	2996
150	150	20	4.21135	0.226643	5.03233	5.25225	0.0961427	7642.58	2994
150	150	20	4.22083	0.226896	5.04803	5.29081	0.0691662	10544.1	2998
150	150	20	4.19213	0.228215	5.01753	5.26523	0.0674473	10875.8	2998
200	200	1	8.82279	0.41502	10.419	10.7189	0.193224	121.317	5499
200	200	1	8.84244	0.419662	10.462	10.7893	0.20037	103.217	5460
200	200	1	8.72048	0.416423	10.3325	10.6365	0.189304	183.75	5543
200	200	1	8.72834	0.412385	10.3122	10.6109	0.27149	149.45	5514
200	200	1	8.74082	0.413261	10.3378	10.6385	0.194714	112.9	5539
200	200	1	8.67161	0.41059	10.2628	10.5552	0.193865	150.083	5582
200	200	5	7.73438	0.417856	9.32773	9.63212	0.186977	102.567	6110
200	200	5	7.68624	0.418372	9.29612	9.61801	0.196958	102.4	6113
200	200	5	7.66922	0.41823	9.28346	9.59105	0.185244	102.55	6137
200	200	5	7.63929	0.409469	9.22567	9.51963	0.271145	102.217	6128

200	200	5	7.64432	0.412849	9.23505	9.53445	0.189162	103.6	6170
200	200	5	7.59577	0.411795	9.18051	9.47586	0.189375	104.117	6207
200	200	10	7.7208	0.413738	9.32521	9.63503	0.109808	366.433	5976
200	200	10	7.68887	0.420539	9.30176	9.64751	0.125259	318.233	5971
200	200	10	7.63648	0.41634	9.24692	9.57824	0.110241	415.217	5974
200	200	10	7.63962	0.410398	9.22346	9.52858	0.140627	368.083	5968
200	200	10	7.64721	0.413087	9.24854	9.55905	0.104053	462.533	5974
200	200	10	7.60056	0.41395	9.21075	9.52345	0.0954432	537.8	5975
200	200	15	7.71602	0.415399	9.31811	9.67543	0.0717614	4921.35	3992
200	200	15	7.67493	0.420179	9.28028	9.67917	0.0929914	3809.78	3987
200	200	15	7.65281	0.418342	9.26075	9.65524	0.0864985	4069.35	3988
200	200	15	7.65049	0.411404	9.22167	9.58004	0.0947213	3790.22	3986
200	200	15	7.66107	0.416777	9.25407	9.62199	0.0708648	5044.82	3990
200	200	15	7.60782	0.413802	9.21748	9.57597	0.0695079	5166.45	3990
200	200	20	7.72323	0.419003	9.32312	9.72772	0.0705027	7221.52	2996
200	200	20	7.68054	0.421994	9.29705	9.72803	0.0915421	5576.02	2994
200	200	20	7.72578	0.420902	9.33963	9.7661	0.0849836	5934.64	2994
200	200	20	7.6437	0.411732	9.23214	9.60232	0.0928278	5542.33	2993
200	200	20	7.67288	0.415276	9.26333	9.66544	0.0694641	7385.8	2995
200	200	20	7.62603	0.416867	9.21228	9.61905	0.0680267	7553.57	2995
250	250	1	13.4992	0.735784	16.197	16.8077	0.205143	112.217	3527
250	250	1	13.5496	0.745211	16.2878	16.9162	0.214125	90.65	3503
250	250	1	13.357	0.737744	16.0736	16.6874	0.20041	70.9167	3553
250	250	1	13.4	0.751216	16.1041	16.7697	0.281492	110.45	3519
250	250	1	13.404	0.746044	16.1466	16.7634	0.208255	102.267	3536
250	250	1	13.2786	0.721818	15.973	16.5736	0.208181	124.533	3576

250	250	5	12.161	0.711464	14.8266	15.4356	0.202449	64.7167	3836
250	250	5	12.1691	0.752306	14.9256	15.5604	0.21107	64.05	3804
250	250	5	12.1114	0.779617	14.8574	15.5555	0.187957	64.35	3811
250	250	5	12.0921	0.736478	14.7894	15.425	0.2799	63.7833	3820
250	250	5	12.0959	0.732138	14.8003	15.4185	0.203605	64.85	3840
250	250	5	12.033	0.728511	14.7243	15.3351	0.205121	65.1167	3861
250	250	10	12.1712	0.720582	14.8401	15.4535	0.197944	65.55	3832
250	250	10	12.1422	0.72763	14.8569	15.4793	0.209838	65.3833	3823
250	250	10	12.0489	0.720688	14.7216	15.3356	0.197232	66.1167	3862
250	250	10	12.1142	0.74692	14.8108	15.4961	0.27361	64.4754	3804
250	250	10	12.1025	0.727069	14.8	15.4107	0.20731	65.7833	3841
250	250	10	12.0181	0.718989	14.6948	15.2996	0.203341	66.1333	3869
250	250	15	12.1916	0.731423	14.8834	15.5026	0.196577	65.95	3820
250	250	15	12.1569	0.751682	14.9181	15.545	0.208496	65.8	3807
250	250	15	12.0562	0.727599	14.7488	15.3665	0.196053	66.6833	3854
250	250	15	12.0927	0.727399	14.7806	15.3995	0.263659	66.1167	3829
250	250	15	12.0766	0.711102	14.7394	15.3487	0.201856	66.6167	3857
250	250	15	12.0268	0.738464	14.739	15.3649	0.200054	66.4098	3853
250	250	20	12.2125	0.752818	14.9529	15.6639	0.0731103	2936.58	2993
250	250	20	12.4555	0.779487	15.2495	16.047	0.08936	2205.05	2991
250	250	20	12.303	0.73721	15.0158	15.7554	0.0838576	2534.25	2991
250	250	20	12.0948	0.727061	14.7903	15.4857	0.0923478	2460.95	2991
250	250	20	12.1151	0.738965	14.851	15.5553	0.0723832	3047.57	2993
250	250	20	12.0503	0.733071	14.7637	15.469	0.0704485	3231.12	2993
300	300	1	20.4929	1.42199	25.2175	26.3395	0.208023	37.9508	2261
300	300	1	20.6022	1.44729	25.4157	26.5578	0.218705	37.6333	2241

300	300	1	20.2825	1.42369	25.0204	26.1388	0.207109	38.2	2278
300	300	1	20.2686	1.41428	24.999	26.1177	0.291284	37.9833	2272
300	300	1	20.3243	1.41721	25.0768	26.1964	0.215144	38.15	2272
300	300	1	20.1678	1.40602	24.8651	25.9705	0.213059	38.35	2292
300	300	5	18.8801	1.41141	23.6241	24.745	0.202863	41.1667	2405
300	300	5	18.9792	1.42221	23.7677	24.9046	0.217034	40.5	2388
300	300	5	18.7925	1.41935	23.538	24.663	0.198233	41.1	2413
300	300	5	18.7458	1.39196	23.4497	24.5642	0.29091	40.5833	2414
300	300	5	18.7449	1.40674	23.4907	24.6041	0.210588	41.1639	2418
300	300	5	18.6257	1.39725	23.2873	24.3979	0.205324	41.6667	2439
300	300	10	18.8821	1.409	23.6003	24.723	0.199509	41.8	2407
300	300	10	18.9634	1.4305	23.7616	24.9058	0.214928	41.4833	2388
300	300	10	18.7615	1.39058	23.4597	24.5755	0.201758	41.95	2421
300	300	10	18.7358	1.40322	23.4561	24.5806	0.283317	41.5667	2413
300	300	10	18.7585	1.39417	23.457	24.5717	0.2105	42.0167	2421
300	300	10	18.6192	1.4001	23.3206	24.4326	0.203443	42.25	2435
300	300	15	18.9051	1.40803	23.628	24.7537	0.202349	42.1833	2404
300	300	15	18.9587	1.45059	23.81	24.9539	0.211916	41.85	2384
300	300	15	18.7605	1.42102	23.4958	24.617	0.201578	42.3167	2417
300	300	15	18.7358	1.40361	23.4363	24.5577	0.281722	42.2833	2415
300	300	15	18.7547	1.39213	23.4659	24.5854	0.20964	42.4167	2419
300	300	15	18.6313	1.39025	23.2964	24.4058	0.204425	42.7167	2437
300	300	20	18.9221	1.40122	23.6748	24.7989	0.203579	42.5333	2399
300	300	20	18.9822	1.41558	23.7784	24.9239	0.213798	42.3167	2386
300	300	20	18.7938	1.41111	23.5036	24.6258	0.201139	42.7167	2416
300	300	20	18.7357	1.39878	23.4312	24.5493	0.279985	42.7333	2415

300	300	20	18.7401	1.40958	23.4682	24.5882	0.206195	42.8667	2419
300	300	20	18.6403	1.39761	23.3354	24.445	0.20674	43.0667	2433

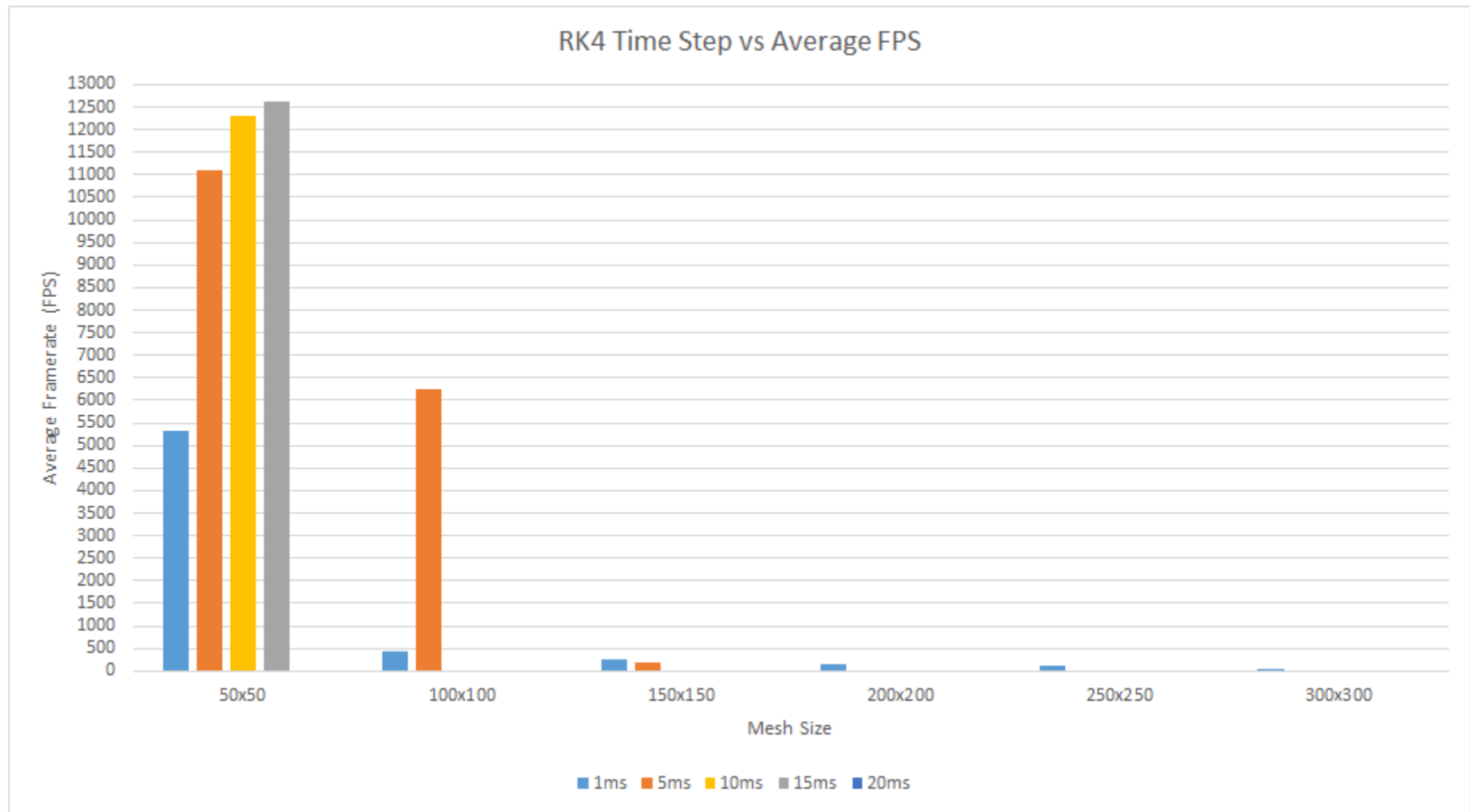Table C.13: Raw data for fourth order Runge-Kutta (sheet)

Figure C.79: Fourth Order Runge-Kutta time step against average FPS (sheet)
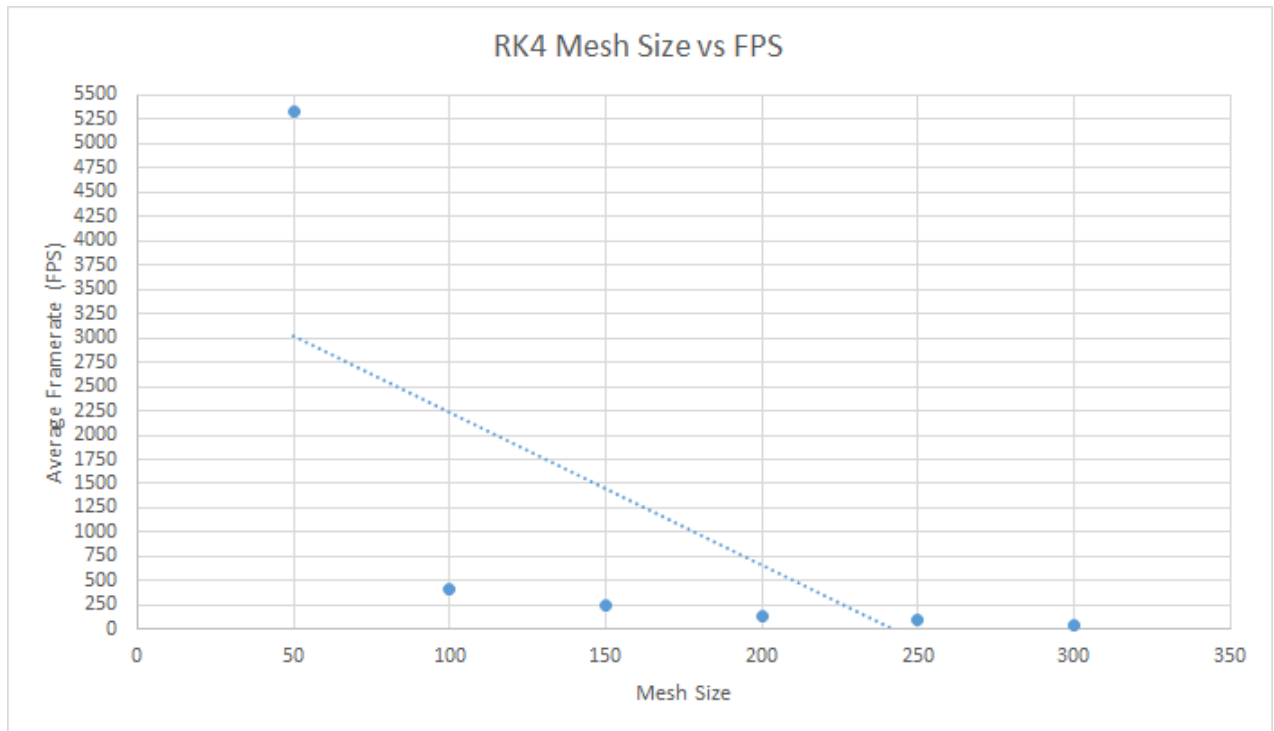
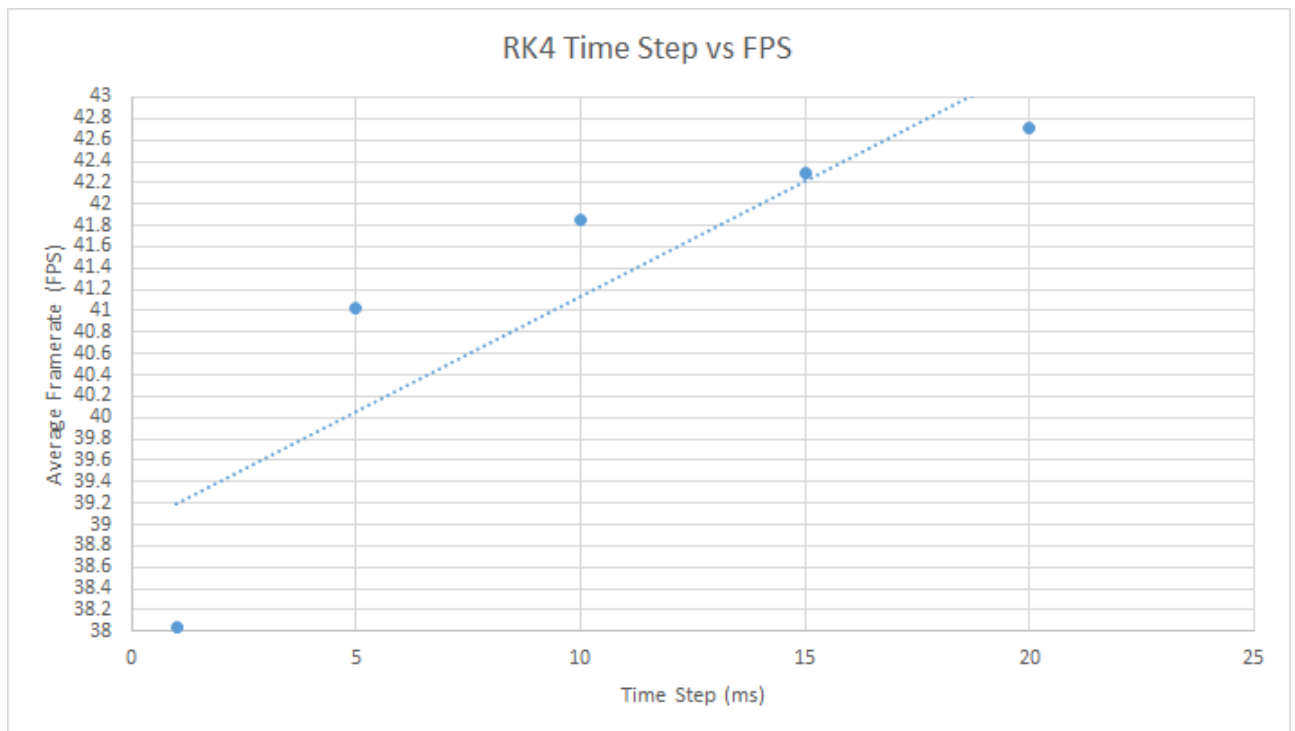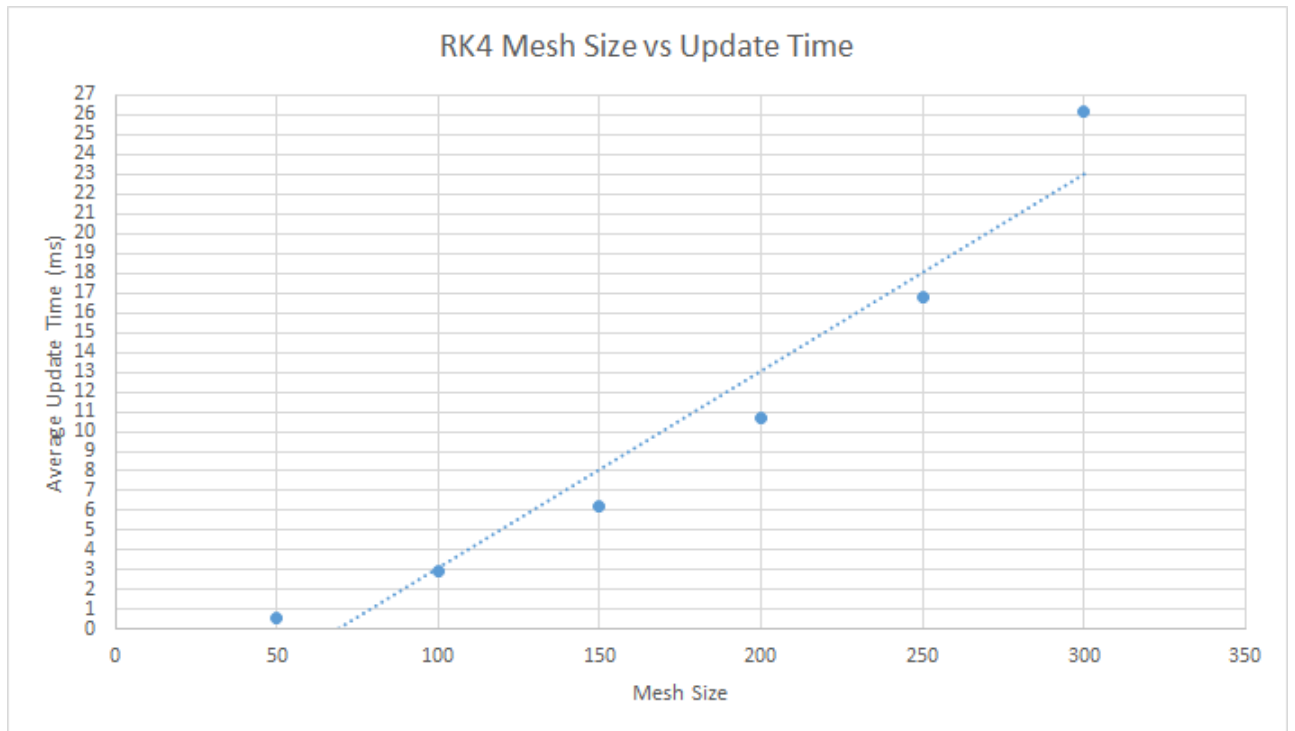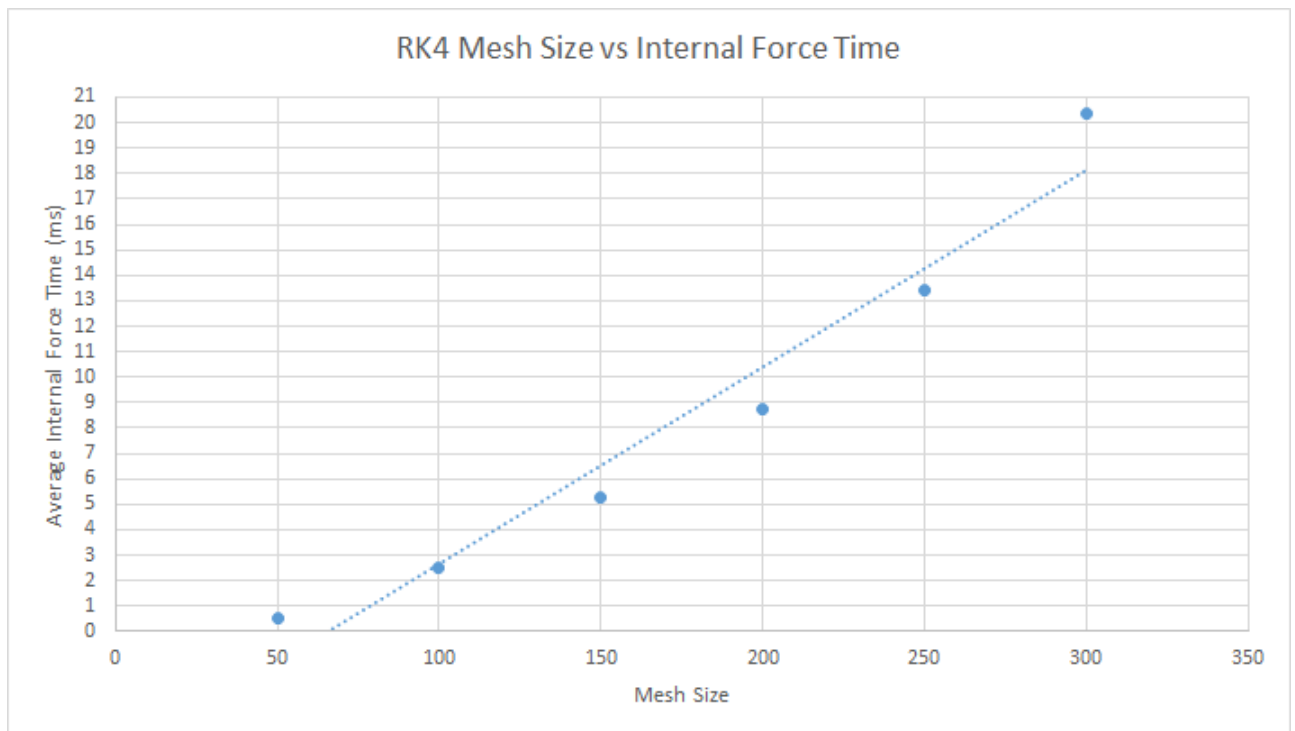Figure C.80: Fourth Order Runge-Kutta mesh size against average FPS (sheet)

Figure C.81: Fourth Order Runge-Kutta time step against average FPS for a 300^2 mesh (sheet)

Figure C.82: Fourth Order Runge-Kutta mesh size against average update time (sheet)

Figure C.83: Fourth Order Runge-Kutta mesh size against average internal force time (sheet)
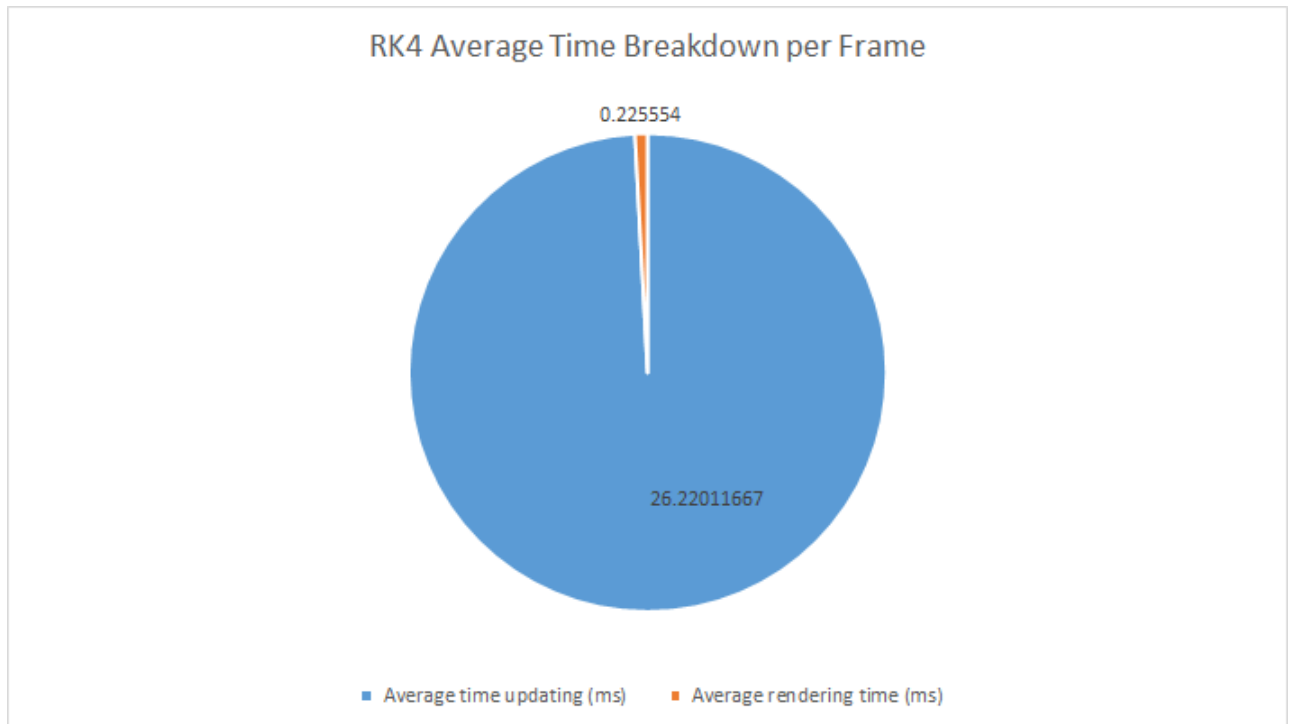
Figure C.84: Fourth Order Runge-Kutta frame time breakdown (sheet)
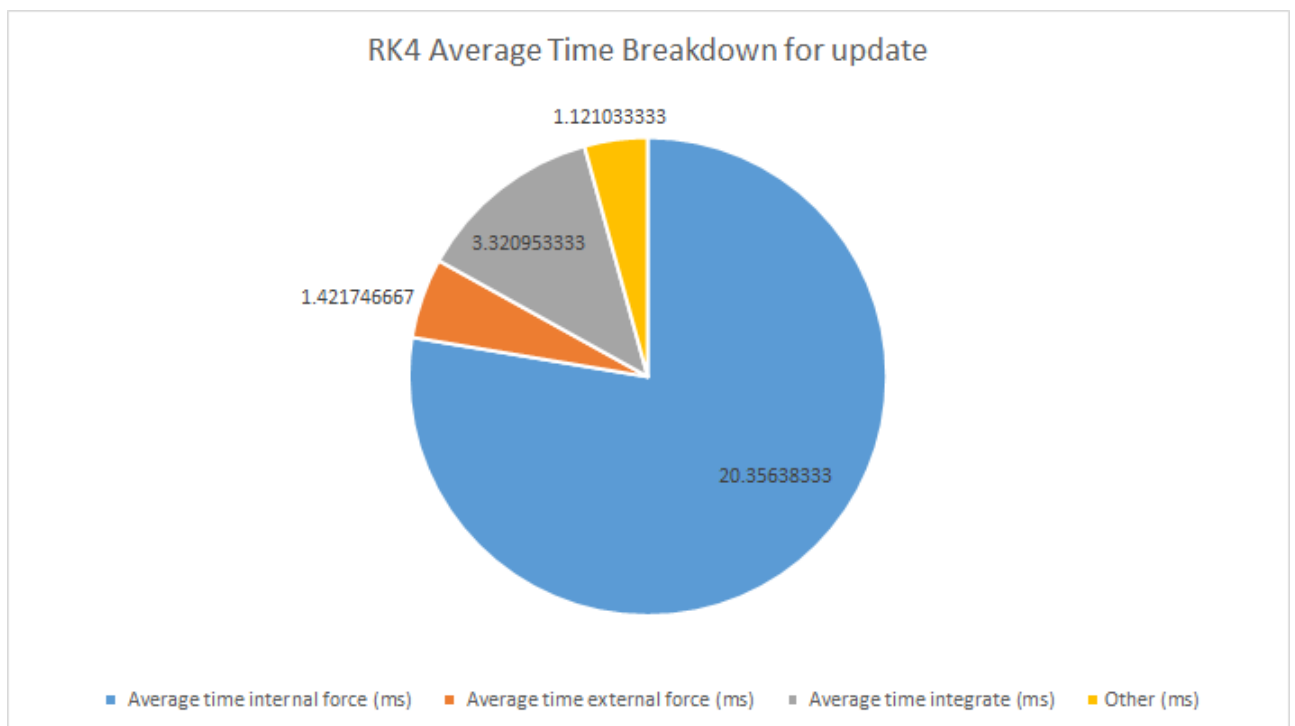
Figure C.85: Fourth Order Runge-Kutta update time breakdown (sheet)
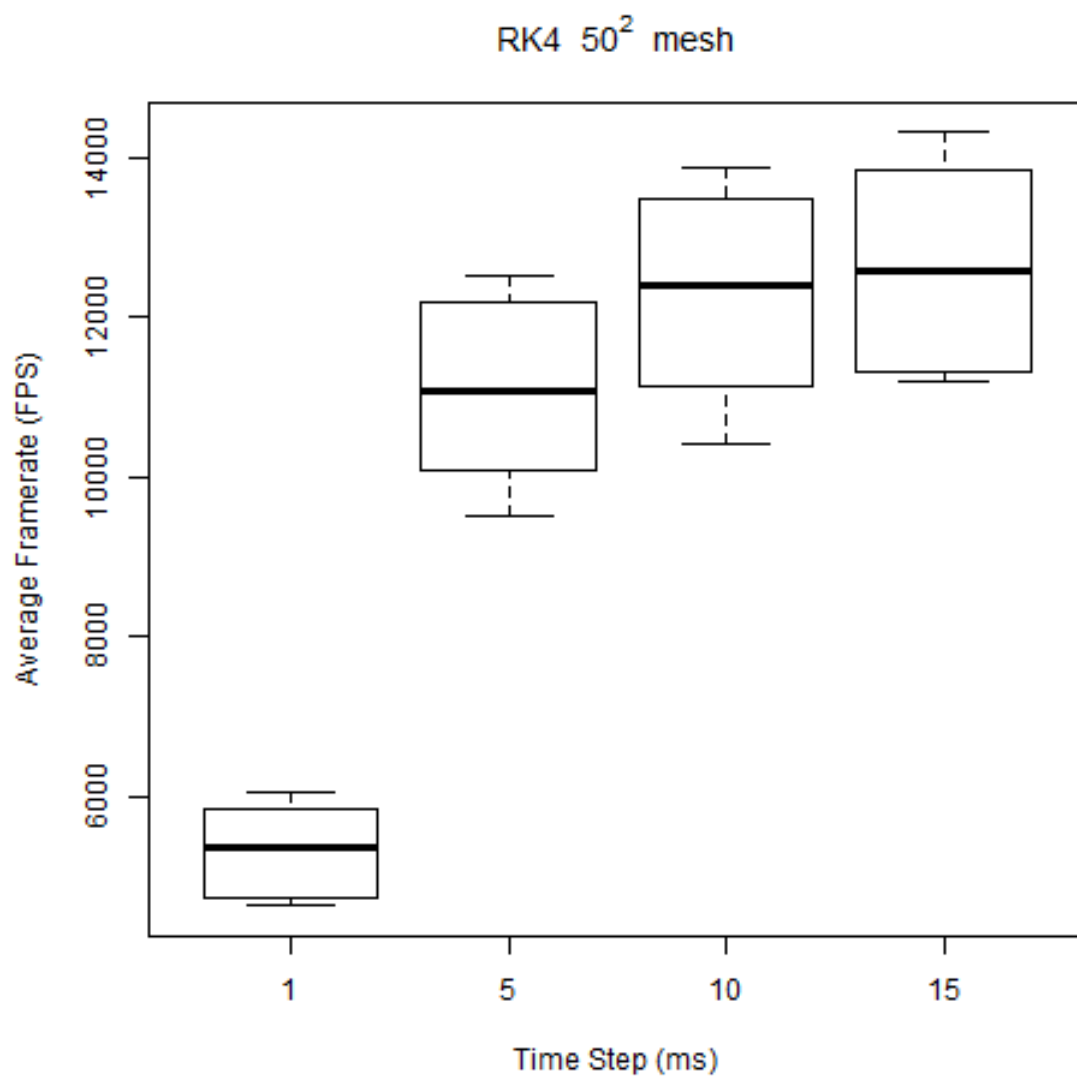
Figure C.86: Fourth order Runge-Kutta box plot for a 50^2 mesh (sheet)
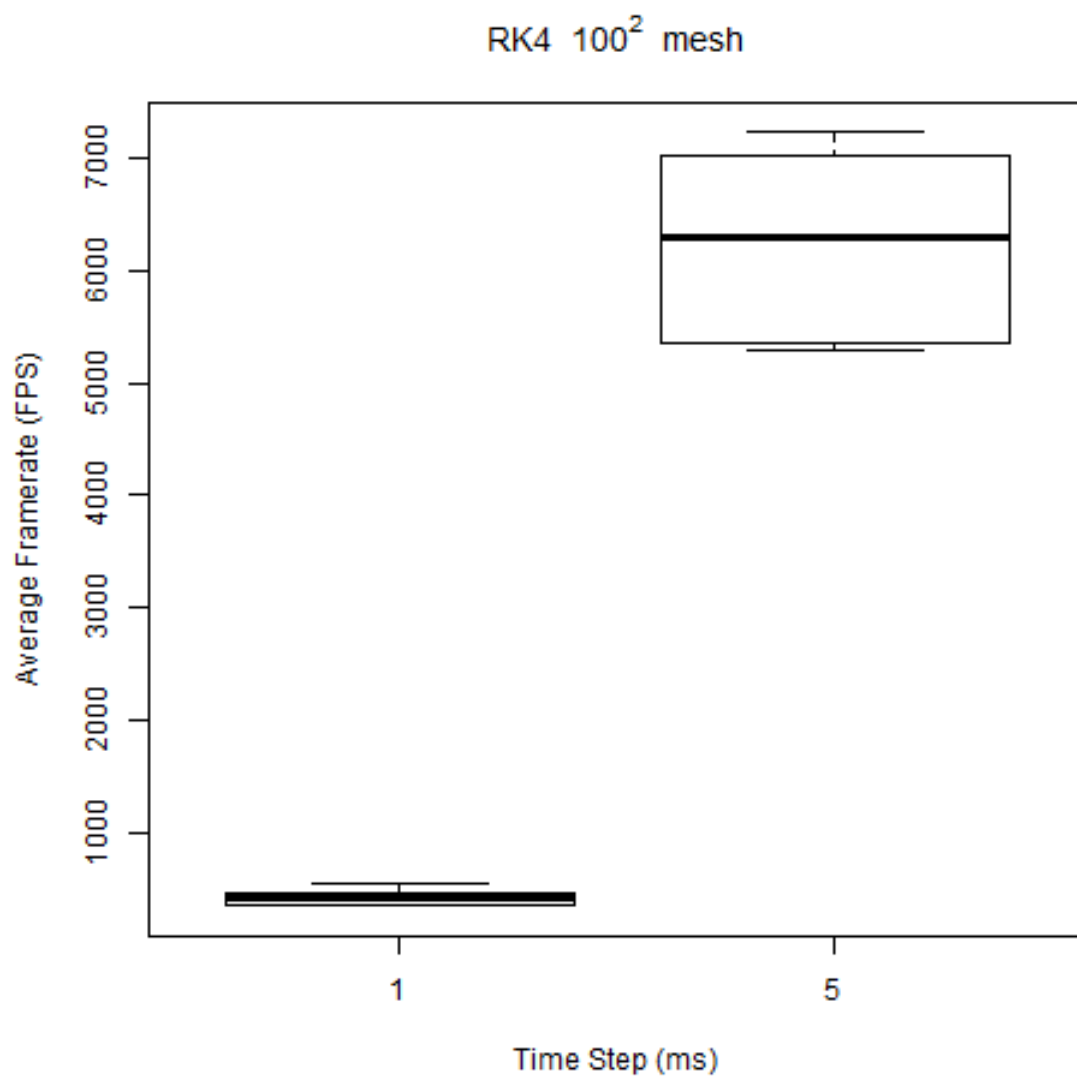
Figure C.87: Fourth order Runge-Kutta box plot for a 100^2 mesh (sheet)
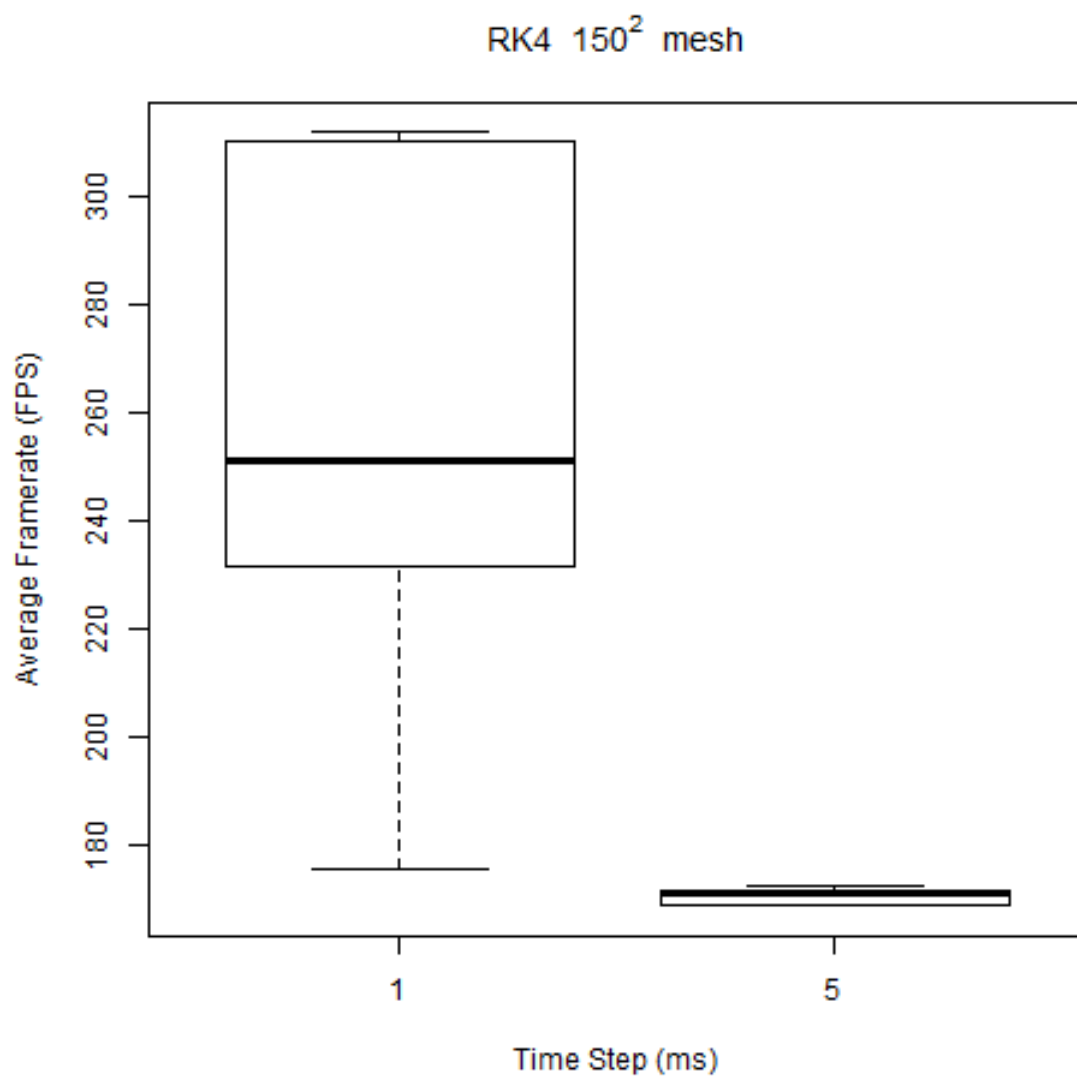
Figure C.88: Fourth order Runge-Kutta box plot for a 150^2 mesh (sheet)
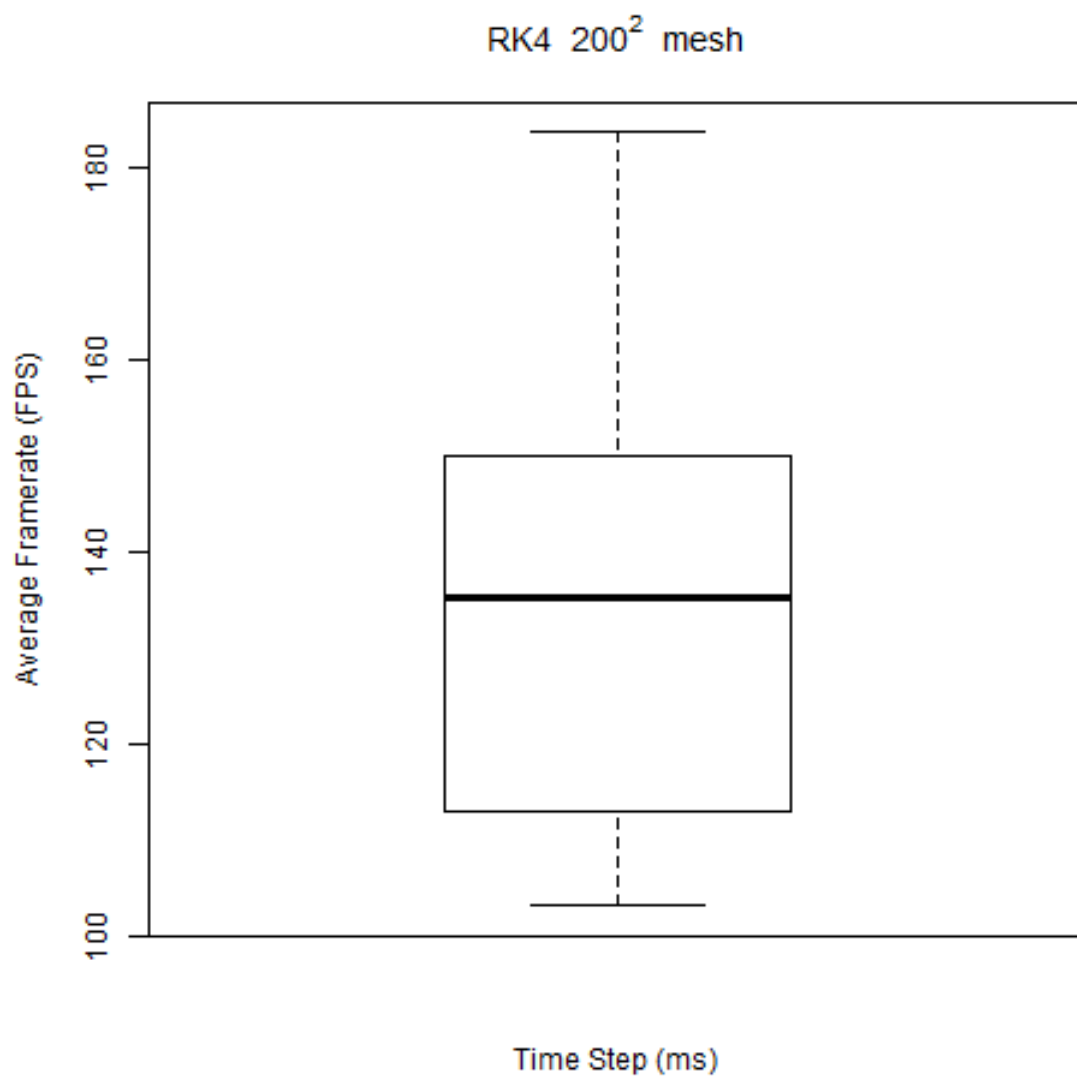
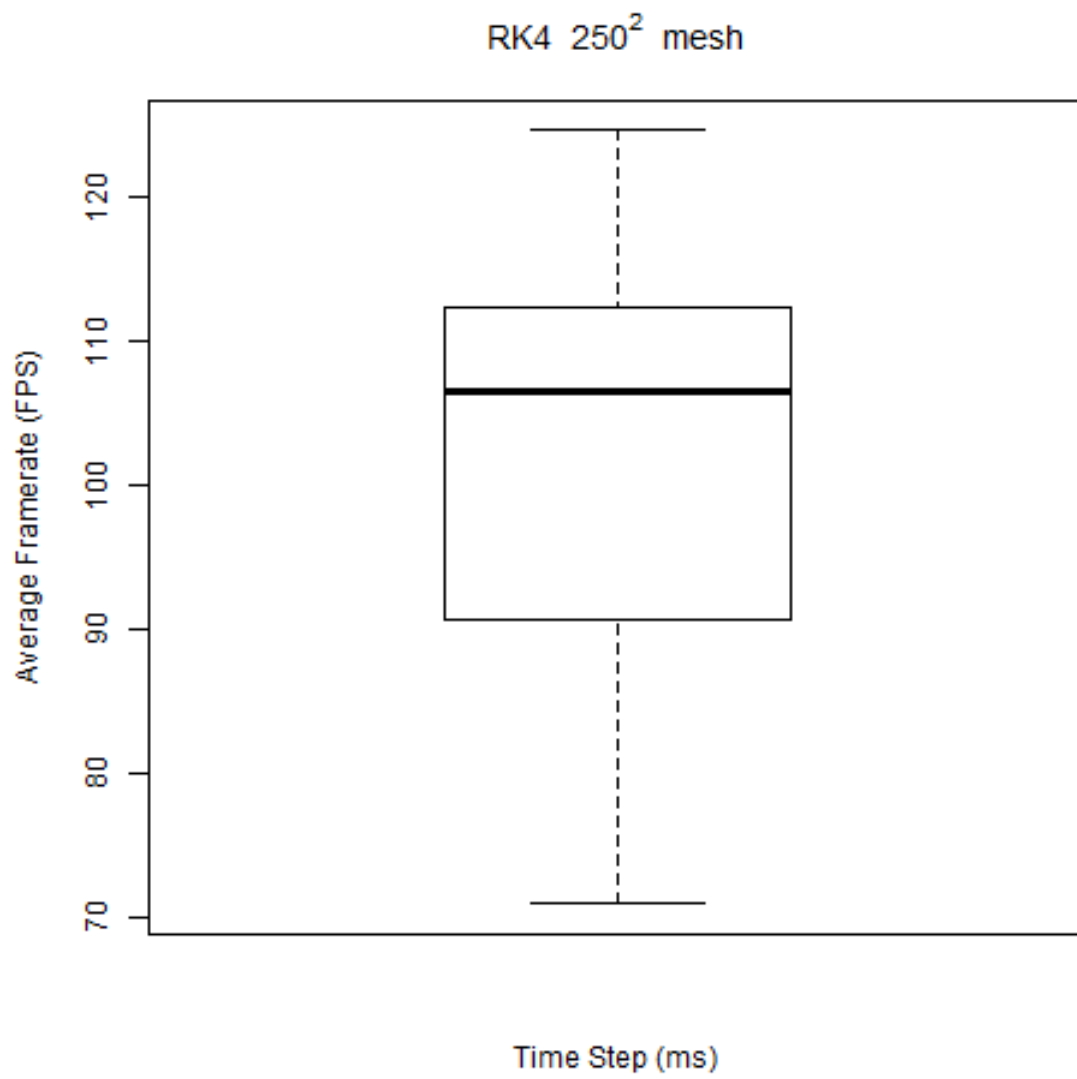Figure C.89: Fourth order Runge-Kutta box plot for a 200^2 mesh (sheet)

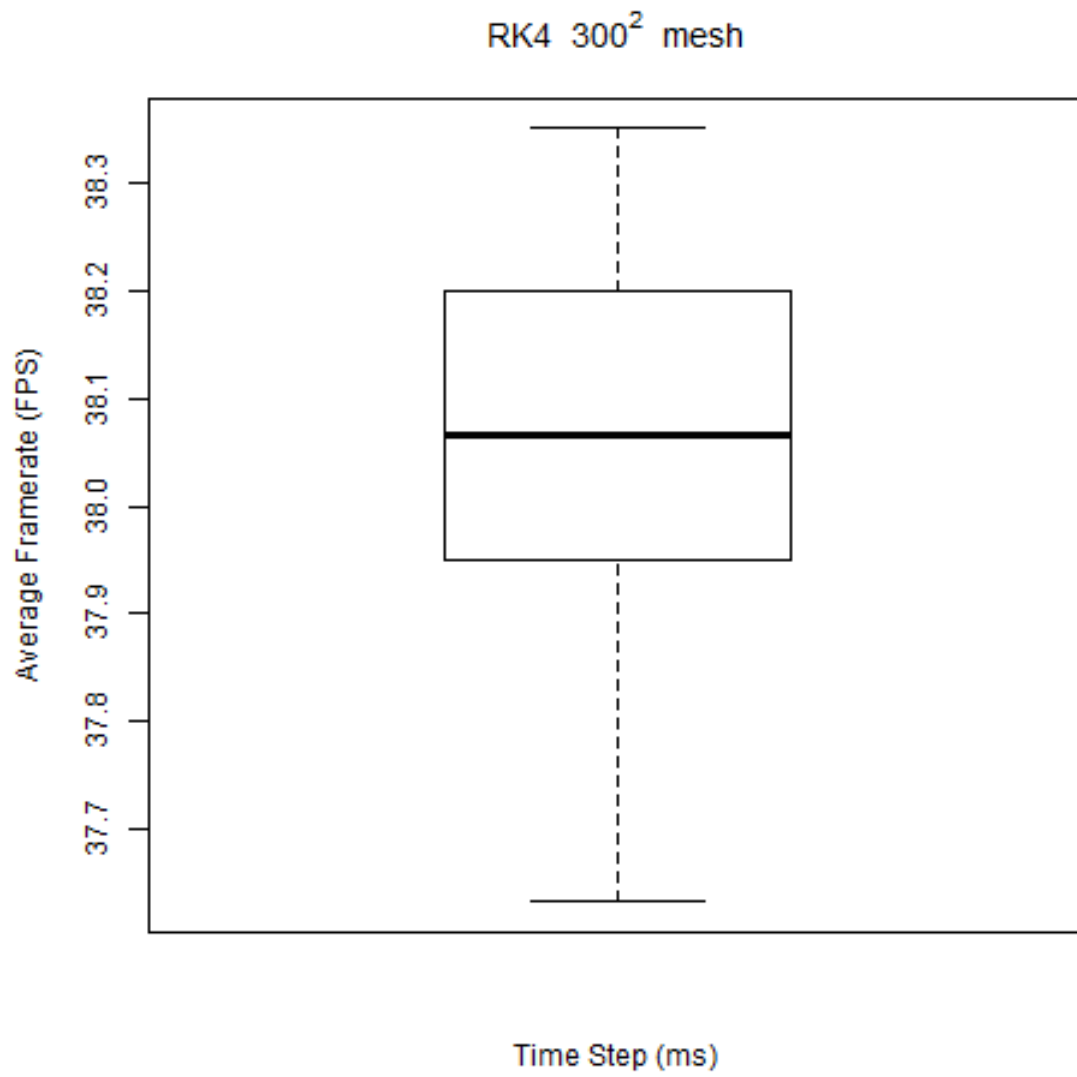Figure C.90: Fourth order Runge-Kutta box plot for a 250^2 mesh (sheet)

Figure C.91: Fourth order Runge-Kutta box plot for a 300^2 mesh (sheet)

Mesh Size	Time Step (ms)	Stable/Unstable
50^2	1	Stable
50^2	5	Stable
50^2	10	Stable
50^2	15	Stable
50^2	20	Unstable
100^2	1	Stable
100^2	5	Stable
100^2	10	Unstable
100^2	15	Unstable
100^2	20	Unstable
150^2	1	Stable
150^2	5	Stable
150^2	10	Unstable
150^2	15	Unstable
150^2	20	Unstable
200^2	1	Stable
200^2	5	Unstable
200^2	10	Unstable
200^2	15	Unstable
200^2	20	Unstable
250^2	1	Stable
250^2	5	Unstable
250^2	10	Unstable
250^2	15	Unstable
250^2	20	Unstable
300^2	1	Stable
300^2	5	Unstable
300^2	10	Unstable
300^2	15	Unstable
300^2	20	Unstable

Table C.14: Stability results for fourth order Runge-Kutta (sheet)

C.4.2 Flag Data

Rows	Columns	Time Step	Internal force (ms)	External force (ms)	Integrate (ms)	Update (ms)	Render (ms)	FPS	Update count
50	50	1	0.571771	0.251376	0.883202	0.900551	0.0725067	1863.47	57868
50	50	1	0.581211	0.259255	0.907117	0.925019	0.0773659	1350.85	58179
50	50	1	0.578841	0.261286	0.906357	0.923732	0.0729485	1454.28	58218
50	50	1	0.567085	0.252367	0.879916	0.897492	0.15403	1060.62	55940
50	50	1	0.571243	0.253565	0.884344	0.902228	0.0725854	1806.15	57870
50	50	1	0.563799	0.252538	0.876535	0.894639	0.0690122	2017.22	57938
50	50	5	0.650669	0.251691	0.962637	1.00026	0.0697206	11434.2	11945
50	50	5	0.655426	0.251613	0.966756	1.00929	0.08698	9154.72	11914
50	50	5	0.653885	0.253861	0.968534	1.00686	0.0825761	9666.65	11918
50	50	5	0.646844	0.252482	0.959963	0.995821	0.0851802	9383.1	11921
50	50	5	0.649321	0.253719	0.962723	0.999307	0.0686408	11626.6	11938
50	50	5	0.642042	0.252761	0.955182	0.993142	0.0665259	12021.9	11944
50	50	10	0.595832	0.25166	0.907828	0.967886	0.0694853	12937.9	5996
50	50	10	0.599614	0.251615	0.910963	0.973813	0.0867777	10367.4	5987
50	50	10	0.597201	0.254117	0.912089	0.971752	0.0826191	10900.6	5988
50	50	10	0.592005	0.252751	0.90549	0.961601	0.0823607	10935.5	5989
50	50	10	0.594775	0.253738	0.908258	0.96573	0.0683925	13155.4	5993
50	50	10	0.587358	0.252763	0.900492	0.960854	0.0661854	13599.8	5993
50	50	15	0.572208	0.25186	0.884423	0.966234	0.0691722	13458.4	4003
50	50	15	0.574637	0.251567	0.885939	0.96771	0.0869191	10721.4	3997
50	50	15	0.574877	0.253966	0.889623	0.967158	0.0826747	11274.3	3998
50	50	15	0.567375	0.252687	0.880864	0.956318	0.0831726	11219.1	3998
50	50	15	0.570649	0.253799	0.884197	0.96208	0.0682751	13643.2	4001
50	50	15	0.563227	0.252781	0.876425	0.9583	0.0660752	14105.8	4000

50	50	20	0.453441	0.244872	0.758802	0.863366	0.0695578	13685.3	3004
50	50	20	0.449974	0.244515	0.754259	1.32957	0.0844739	10679.5	3053
50	50	20	0.480609	0.246844	0.788885	1.23961	0.0819224	11255	3024
50	50	20	0.448672	0.246013	0.755515	0.85266	0.0820871	11606.2	3001
50	50	20	0.45278	0.246897	0.759487	0.857467	0.0681264	13981.5	3003
50	50	20	0.444082	0.246071	0.750532	0.854901	0.0662953	14375.1	3002
100	100	1	2.51524	1.02384	3.78868	3.85251	0.0828854	254.633	15246
100	100	1	2.52332	1.02474	3.80474	3.87358	0.0877292	253.267	15146
100	100	1	2.62118	1.02944	3.90318	3.97106	0.0820599	247	14803
100	100	1	2.49725	1.02442	3.77193	3.83478	0.170008	249.883	14982
100	100	1	2.49688	1.02384	3.77198	3.83481	0.0832544	256.033	15313
100	100	1	2.48054	1.02294	3.75279	3.81983	0.0789025	257.7	15389
100	100	5	2.18495	1.02244	3.45718	3.5318	0.0699508	4265.18	11925
100	100	5	2.19799	1.02452	3.4791	3.56592	0.0892771	3293.18	11904
100	100	5	2.17786	1.02538	3.455	3.53974	0.0852044	3495.77	11902
100	100	5	2.16655	1.02239	3.43843	3.51082	0.091836	3308.37	11897
100	100	5	2.17038	1.02348	3.44344	3.51686	0.0690842	4369.67	11927
100	100	5	2.15498	1.02303	3.42752	3.50366	0.0670877	4552.35	11929
100	100	10	1.85696	0.992772	3.10064	3.20027	0.0697272	9729.55	5993
100	100	10	1.84725	0.995073	3.09986	3.20818	0.0874766	7761.38	5984
100	100	10	1.83589	0.997708	3.08654	3.20252	0.0788184	8605.88	5988
100	100	10	1.8388	0.99311	3.08199	3.17726	0.0838125	8132.58	5984
100	100	10	1.83571	0.993736	3.08041	3.17734	0.0686622	9922.2	5990
100	100	10	1.82332	0.993802	3.06726	3.16672	0.0667683	10206	5990
100	100	15	1.85703	0.991961	3.10046	3.22263	0.0698257	11206.7	4000
100	100	15	1.8451	0.993763	3.09689	3.22501	0.0882915	8862	3996

100	100	15	1.97845	1.00923	3.24582	3.38011	0.0802625	9626.68	3997
100	100	15	1.83281	0.992577	3.07605	3.19102	0.0832723	9431.7	3996
100	100	15	1.8376	0.993224	3.08301	3.19871	0.0689588	11362.2	3998
100	100	15	1.82344	0.993121	3.06722	3.18979	0.0669918	11712.6	3999
100	100	20	1.85718	0.991711	3.10135	3.24685	0.0697008	11961.7	3003
100	100	20	1.84625	0.993661	3.09814	3.24549	0.0881658	9468.25	2999
100	100	20	1.97534	1.009	3.24187	3.40386	0.0792637	10434.1	3001
100	100	20	1.83547	0.993621	3.08071	3.21793	0.0832971	10050.6	3000
100	100	20	1.83688	0.993327	3.08236	3.22155	0.0687674	12146.7	3001
100	100	20	1.82502	0.993324	3.06954	3.21539	0.0668502	12506.5	3002
150	150	1	5.32873	2.31429	8.2277	8.38344	0.145265	117.7	7035
150	150	1	5.33456	2.3192	8.25767	8.4291	0.159227	116.9	6986
150	150	1	5.3302	2.31984	8.2433	8.40599	0.141182	117.233	7020
150	150	1	5.29687	2.3174	8.19887	8.34928	0.226954	117.333	6996
150	150	1	5.29669	2.31695	8.19865	8.35395	0.152328	118.033	7054
150	150	1	5.25894	2.31896	8.1602	8.31035	0.154677	118.783	7088
150	150	5	4.74976	2.31543	7.64885	7.80049	0.139953	127	7556
150	150	5	4.77334	2.31472	7.68983	7.8602	0.151356	125.867	7489
150	150	5	4.86046	2.32766	7.77856	7.94059	0.146987	124.383	7418
150	150	5	4.71531	2.31823	7.61866	7.76784	0.232761	126.115	7499
150	150	5	4.71705	2.31702	7.62419	7.78322	0.147849	127.2	7565
150	150	5	4.6815	2.31849	7.59158	7.73986	0.148458	128.017	7606
150	150	10	4.25149	2.24382	7.08144	7.25168	0.0710398	3915.08	5983
150	150	10	4.22707	2.24997	7.08823	7.28494	0.0907731	3041.2	5975
150	150	10	4.26095	2.2561	7.11275	7.30897	0.086651	3143.85	5976
150	150	10	4.21894	2.24953	7.05627	7.22048	0.0931389	3012.48	5976

150	150	10	4.21713	2.24774	7.05923	7.23494	0.0704596	3972.7	5984
150	150	10	4.19026	2.24906	7.03114	7.19995	0.0687986	4131.62	5984
150	150	15	4.26467	2.24806	7.1084	7.30629	0.0702611	7305.73	3996
150	150	15	4.23149	2.25159	7.09409	7.3166	0.0895326	5731.35	3992
150	150	15	4.20479	2.25835	7.06413	7.28264	0.0861343	5976.98	3994
150	150	15	4.2195	2.2499	7.05993	7.24848	0.0917077	5634.85	3992
150	150	15	4.21886	2.24866	7.06301	7.26824	0.0692076	7457.45	3996
150	150	15	4.19042	2.25039	7.04186	7.23936	0.0672938	7671.28	3996
150	150	20	4.2647	2.24805	7.10661	7.33187	0.0698266	9034.05	3001
150	150	20	4.22838	2.25042	7.09143	7.33594	0.0886503	7140.38	2999
150	150	20	4.19666	2.2521	7.05164	7.29586	0.0840698	7546.68	2999
150	150	20	4.21661	2.2498	7.06147	7.27401	0.0899293	7063.1	2997
150	150	20	4.22215	2.24814	7.06875	7.29913	0.068718	9206.03	3000
150	150	20	4.19454	2.25	7.04683	7.27365	0.0667826	9498.53	2999
200	200	1	9.04429	4.14262	14.366	14.6662	0.195791	67.5	4037
200	200	1	9.03012	4.14503	14.3727	14.6975	0.198242	67.2833	4028
200	200	1	8.93784	4.14711	14.272	14.5762	0.191042	68.1667	4063
200	200	1	8.94997	4.14494	14.2786	14.5758	0.278541	67.6167	4040
200	200	1	8.97067	4.14349	14.2934	14.5938	0.196041	67.8167	4057
200	200	1	8.88948	4.13988	14.2131	14.5068	0.194459	68.3167	4082
200	200	5	7.72556	4.02211	12.935	13.2359	0.190853	75.4833	4468
200	200	5	7.68745	4.02228	12.8977	13.2227	0.198699	75.2333	4470
200	200	5	7.6677	4.01907	12.8698	13.1736	0.186642	75.95	4491
200	200	5	7.64838	4.02464	12.8467	13.1464	0.271996	75.2	4471
200	200	5	7.65186	4.02255	12.8512	13.1552	0.191357	75.8833	4495
200	200	5	7.6112	4.02791	12.8179	13.1172	0.189296	76.2167	4509

200	200	10	7.71844	4.01717	12.9073	13.2077	0.189704	77.3	4478
200	200	10	7.68597	4.01881	12.8987	13.2214	0.195574	77.15	4471
200	200	10	7.60874	4.02431	12.8162	13.1365	0.189747	81.2	4502
200	200	10	7.63461	4.02077	12.8361	13.1338	0.23272	79.1833	4488
200	200	10	7.65833	4.02078	12.8885	13.1955	0.191367	77.8333	4480
200	200	10	7.59783	4.01819	12.8008	13.1013	0.185876	81.4667	4514
200	200	15	7.70561	4.0084	12.9013	13.2336	0.0768221	1577.48	3992
200	200	15	7.66562	4.02022	12.8797	13.2536	0.0979315	1227.87	3988
200	200	15	7.61675	4.02325	12.8236	13.184	0.0918009	1384.03	3991
200	200	15	7.63931	4.02157	12.8273	13.147	0.0993526	1298.9	3988
200	200	15	7.64752	4.01663	12.8315	13.1593	0.0749192	1680.62	3991
200	200	15	7.59516	4.01576	12.7859	13.1096	0.0731029	1779.92	3992
200	200	20	7.72269	4.00977	12.9012	13.287	0.0703865	4785.65	2998
200	200	20	7.70685	4.02319	12.9169	13.3438	0.0913243	3658.65	2996
200	200	20	7.70274	4.02319	12.905	13.2996	0.0852354	3932.4	2997
200	200	20	7.64112	4.01943	12.8276	13.1802	0.095359	3607.73	2995
200	200	20	7.65702	4.018	12.8563	13.224	0.0702229	4840.32	2997
200	200	20	7.61354	4.0169	12.8062	13.1743	0.0684039	5014.43	2998
250	250	1	13.8113	6.48425	22.2236	22.8372	0.205889	43.7333	2604
250	250	1	13.8107	6.50372	22.2785	22.911	0.212838	43.5167	2595
250	250	1	13.6762	6.50921	22.1612	22.7806	0.203281	43.7	2611
250	250	1	13.6883	6.49932	22.1271	22.7386	0.292989	43.7333	2606
250	250	1	13.8256	6.52104	22.3244	23.0168	0.201699	43.3667	2585
250	250	1	13.6162	6.50705	22.0798	22.6914	0.206495	44.0333	2621
250	250	5	12.1827	6.28804	20.4296	21.0486	0.19958	48.3833	2824
250	250	5	12.1532	6.31154	20.4455	21.0766	0.211114	48.05	2818

250	250	5	12.0428	6.31354	20.3123	20.926	0.200144	48.3333	2840
250	250	5	12.0748	6.29941	20.2926	20.9133	0.286103	47.8333	2830
250	250	5	12.0989	6.30555	20.3677	20.9829	0.208464	48.4833	2831
250	250	5	12.0083	6.30718	20.2393	20.8395	0.206249	48.8833	2851
250	250	10	12.1918	6.29181	20.4442	21.0834	0.194608	49.1167	2819
250	250	10	12.1586	6.30776	20.4543	21.0953	0.20445	48.9667	2816
250	250	10	12.0747	6.31568	20.3654	20.9996	0.194581	49.2167	2830
250	250	10	12.1149	6.32271	20.3985	21.0233	0.282732	48.3667	2815
250	250	10	12.0935	6.31307	20.3777	20.9984	0.206543	49.2833	2829
250	250	10	12.0217	6.3115	20.291	20.8981	0.205924	49.5667	2842
250	250	15	12.1766	6.29064	20.4171	21.034	0.200347	49.8333	2825
250	250	15	12.1374	6.30373	20.4321	21.06	0.207189	49.7333	2820
250	250	15	12.0528	6.31336	20.3229	20.9387	0.196932	49.95	2838
250	250	15	12.0593	6.29643	20.2895	20.9024	0.281628	49.5	2831
250	250	15	12.0922	6.30311	20.3586	20.9743	0.204034	49.9667	2832
250	250	15	12.0178	6.3139	20.268	20.8739	0.202022	49.9167	2846
250	250	20	12.1569	6.2859	20.3814	20.9938	0.194493	99.3167	2830
250	250	20	12.2354	6.31477	20.5367	21.1642	0.209458	80.0167	2806
250	250	20	12.0263	6.30747	20.2901	20.9221	0.200115	68.5667	2841
250	250	20	12.0705	6.30283	20.3175	20.9343	0.289417	57.3	2825
250	250	20	12.1527	6.31752	20.4508	21.1425	0.192738	94.7	2811
250	250	20	12.0109	6.31002	20.2557	20.86	0.201159	57.35	2847
300	300	1	21.0396	9.69548	34.0725	35.1971	0.206906	28.4167	1695
300	300	1	21.1679	9.72781	34.2965	35.4462	0.218213	28.3167	1683
300	300	1	20.8155	9.72803	33.8733	34.9985	0.209702	28.6333	1705
300	300	1	20.7941	9.70746	33.8273	34.9533	0.295281	28.5	1703

300	300	1	20.872	9.68895	33.9235	35.0516	0.214681	28.65	1702
300	300	1	20.6843	9.72679	33.7073	34.817	0.216158	28.7333	1713
300	300	5	18.9079	9.39972	31.649	32.7774	0.201525	31.3833	1819
300	300	5	18.9823	9.43072	31.8067	32.95	0.218054	31.2131	1809
300	300	5	18.699	9.43517	31.4668	32.5955	0.204379	31.5833	1829
300	300	5	18.7411	9.40517	31.487	32.6176	0.294934	30.9	1823
300	300	5	18.7584	9.39473	31.4669	32.5856	0.209602	31.5667	1830
300	300	5	18.6687	9.42183	31.4097	32.521	0.206836	31.7333	1833
300	300	10	18.9198	9.41136	31.7027	32.841	0.199314	32	1816
300	300	10	18.9604	9.4225	31.786	32.9338	0.216526	31.8833	1810
300	300	10	18.7821	9.40741	31.5323	32.663	0.198335	32.25	1826
300	300	10	18.7386	9.4031	31.4701	32.5983	0.287537	31.9167	1824
300	300	10	18.7628	9.40481	31.496	32.6121	0.211306	32.2	1828
300	300	10	18.6337	9.42562	31.3641	32.4763	0.204934	32.3833	1836
300	300	15	18.889	9.39206	31.6324	32.7671	0.198694	32.5333	1820
300	300	15	18.9689	9.43062	31.7876	32.933	0.216251	32.3333	1810
300	300	15	18.781	9.40528	31.5113	32.6393	0.201581	32.7333	1827
300	300	15	18.7244	9.41007	31.4814	32.6048	0.284369	32.6333	1824
300	300	15	18.7574	9.40478	31.5032	32.6209	0.210927	32.65	1827
300	300	15	18.6351	9.41903	31.3542	32.4663	0.205302	32.85	1836
300	300	20	18.9017	9.40319	31.6633	32.7934	0.20325	32.9833	1818
300	300	20	18.986	9.44324	31.8444	32.9941	0.211953	32.75	1806
300	300	20	18.7668	9.41047	31.5028	32.6283	0.200612	33.2	1827
300	300	20	18.7264	9.40908	31.4822	32.607	0.280948	33.0833	1824
300	300	20	18.7629	9.4044	31.4872	32.6122	0.208597	33.1167	1828
300	300	20	18.6452	9.43269	31.3941	32.5078	0.208997	33.2667	1833

Table C.15: Raw data for fourth order Runge-Kutta (flag)
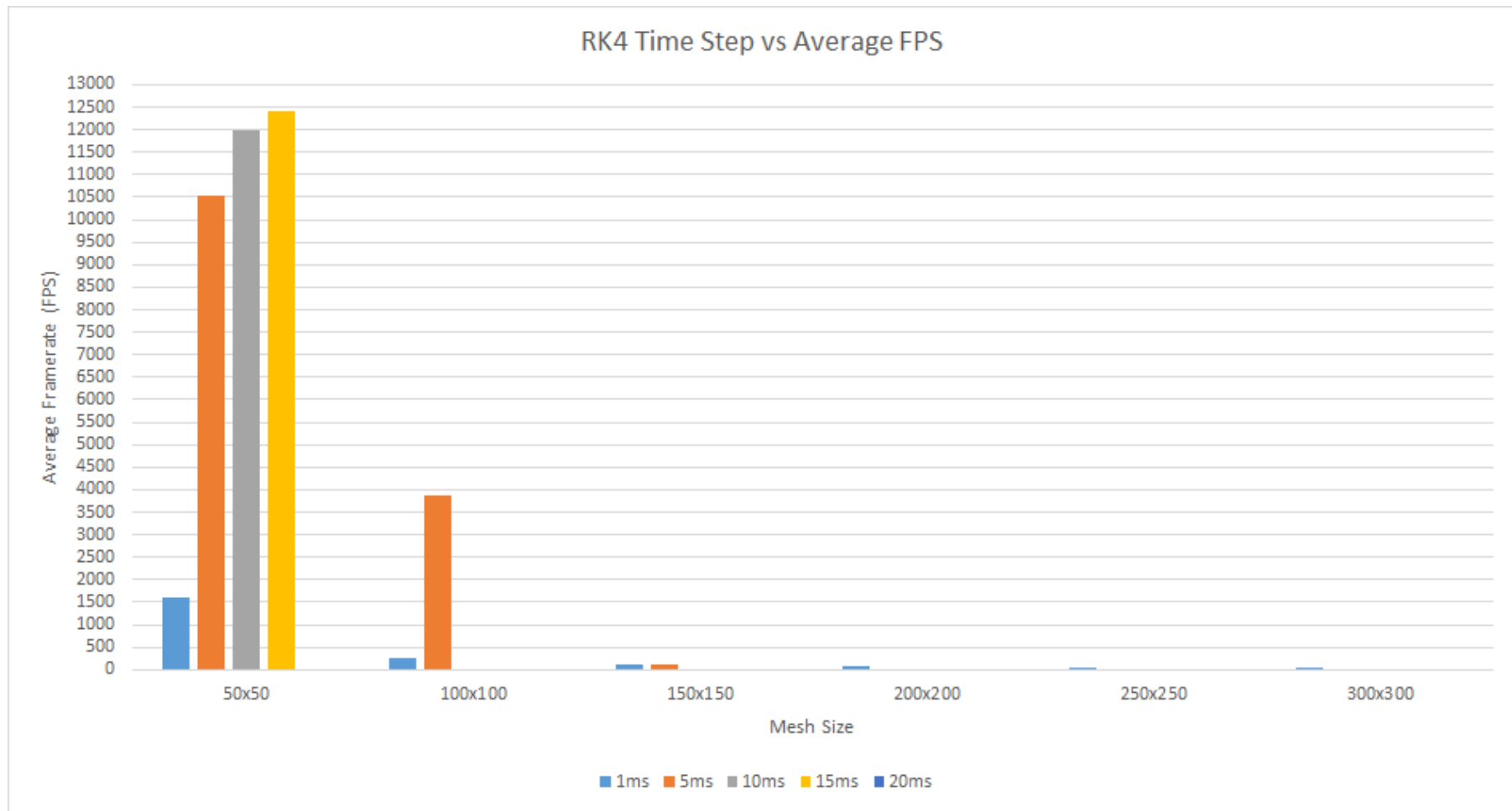
Figure C.92: Fourth Order Runge-Kutta mesh size against average FPS (flag)
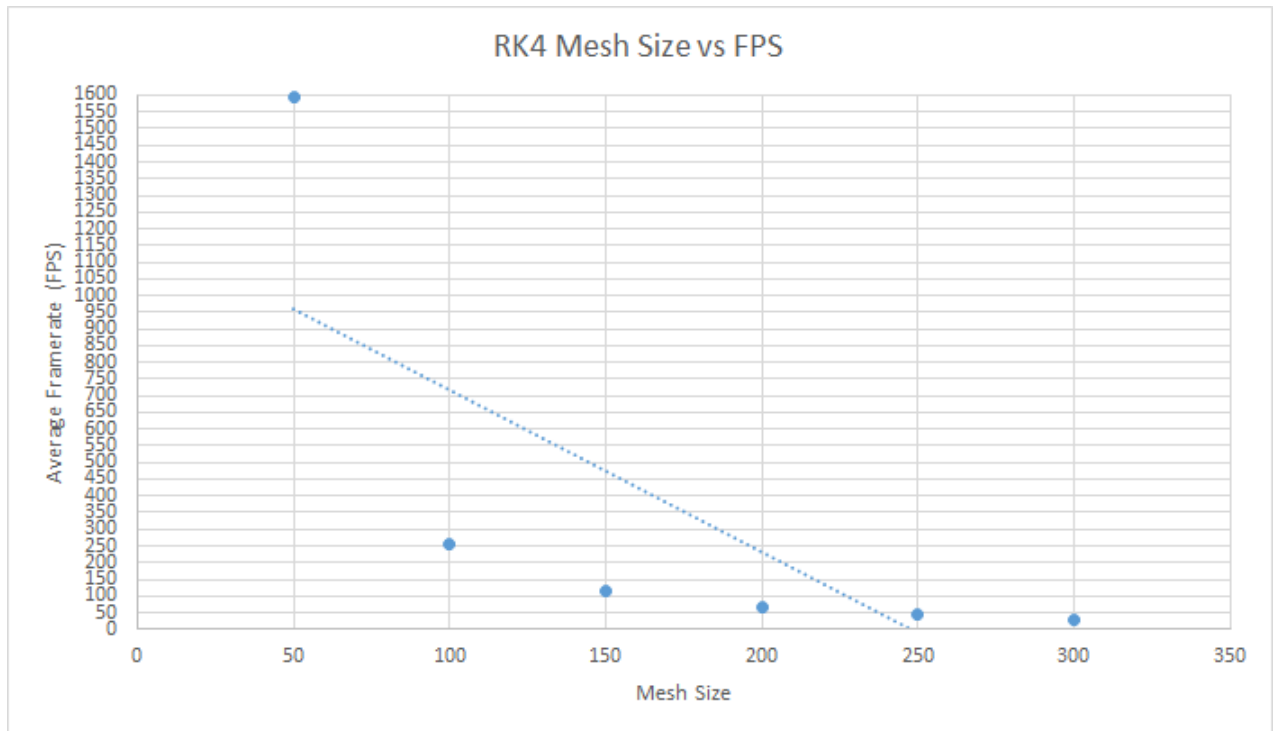
Figure C.93: Fourth Order Runge-Kutta mesh size against average FPS (flag)
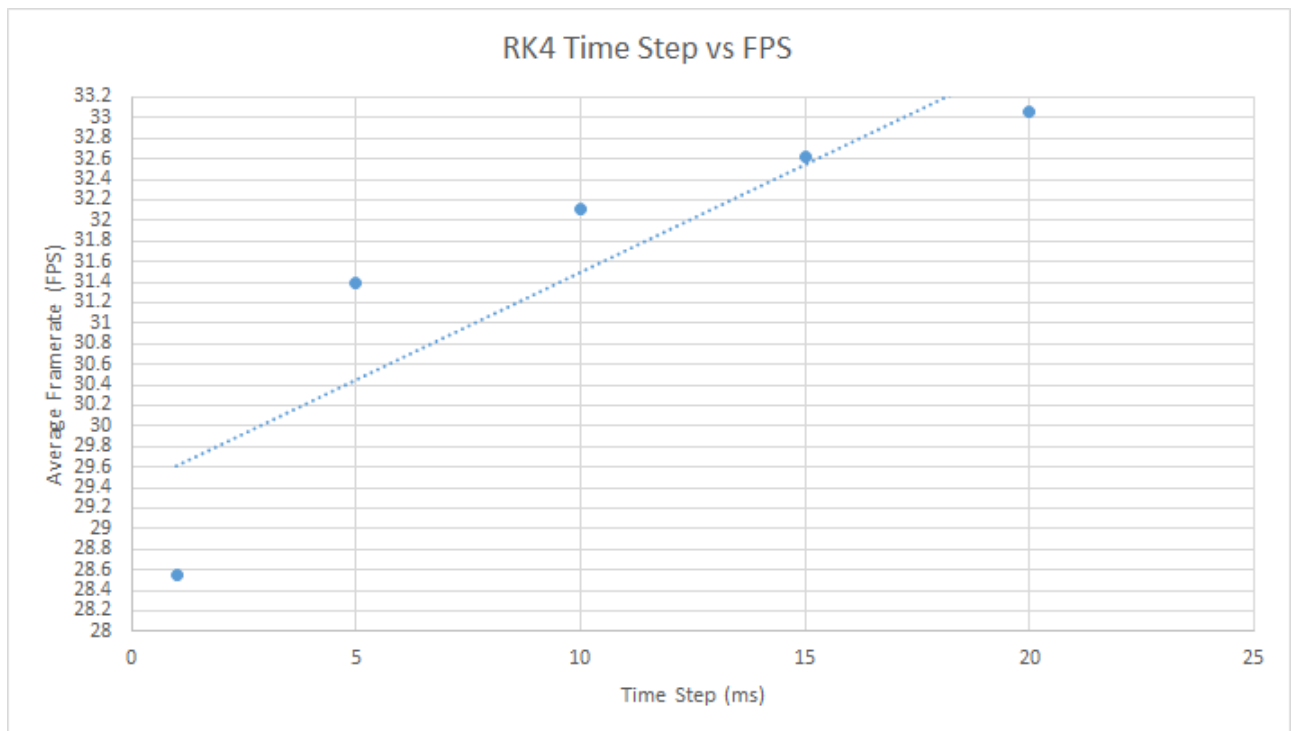
Figure C.94: Fourth Order Runge-Kutta time step against average FPS for a 300^2 mesh (flag)
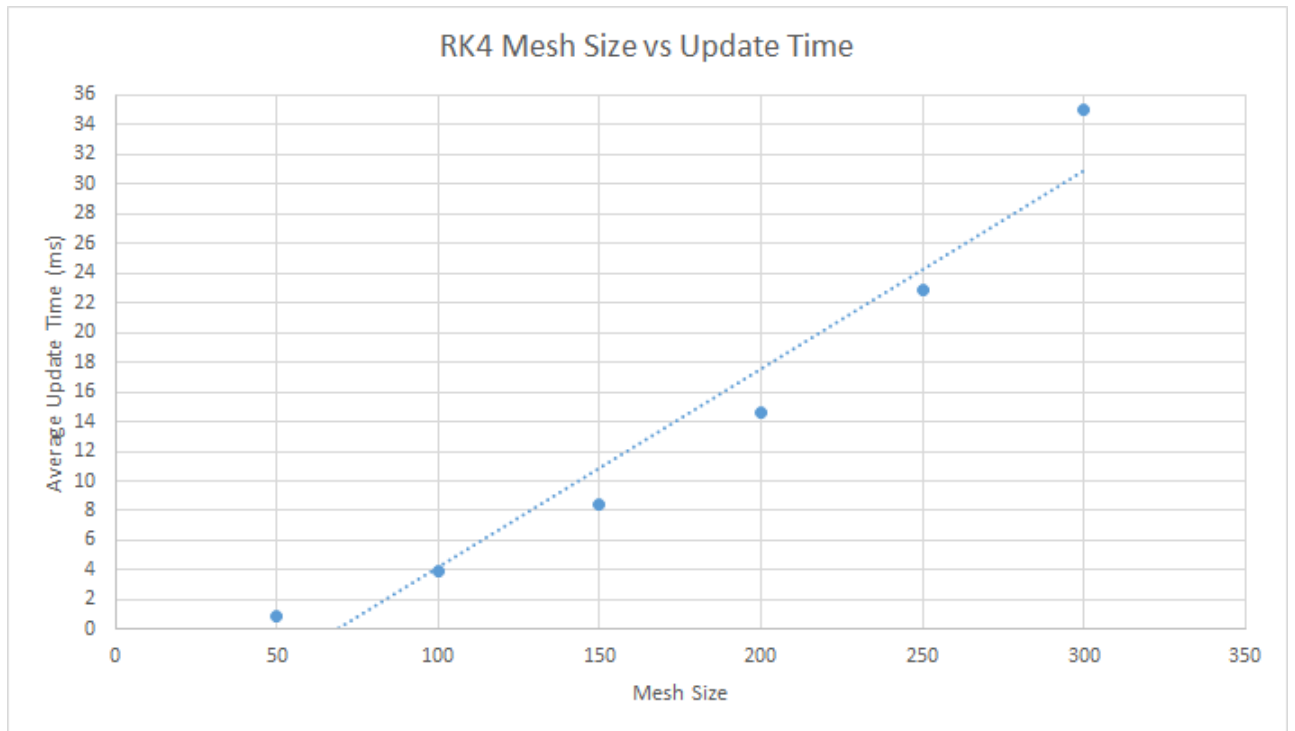
Figure C.95: Fourth Order Runge-Kutta mesh size against average update time (flag)
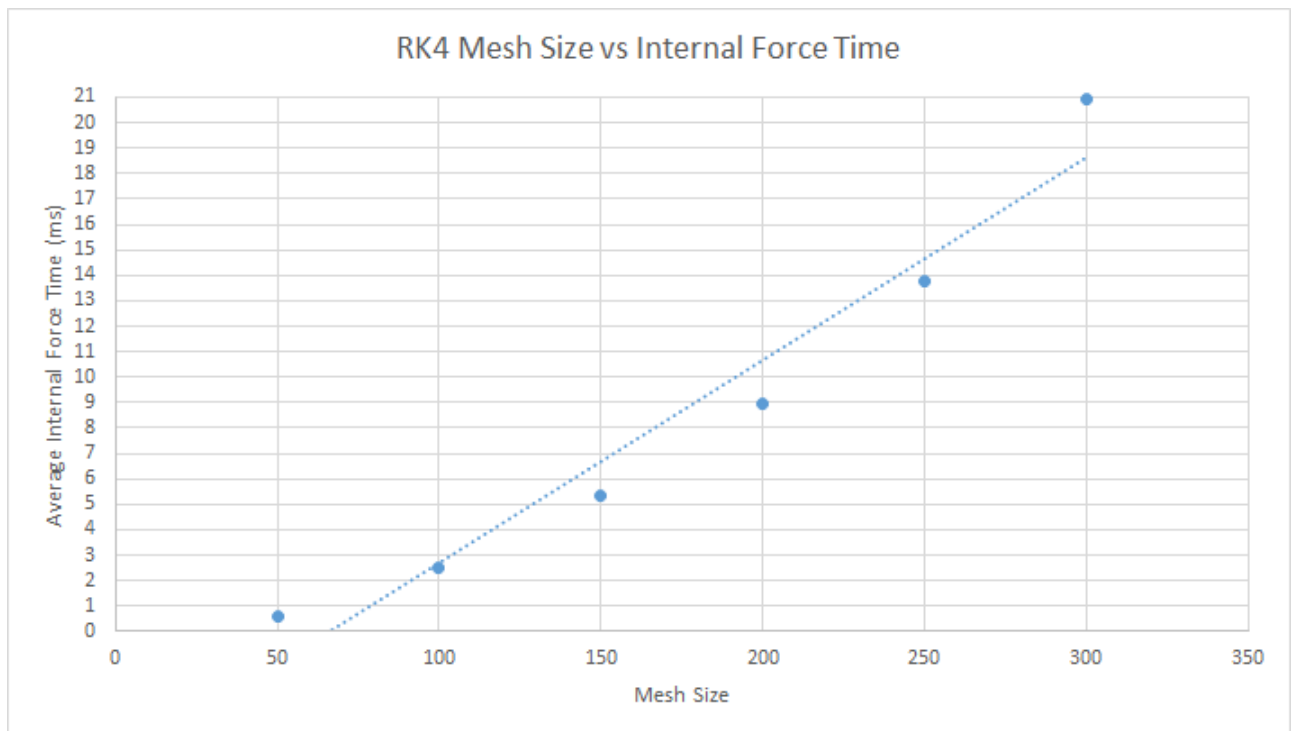
Figure C.96: Fourth Order Runge-Kutta mesh size against average internal force time (flag)
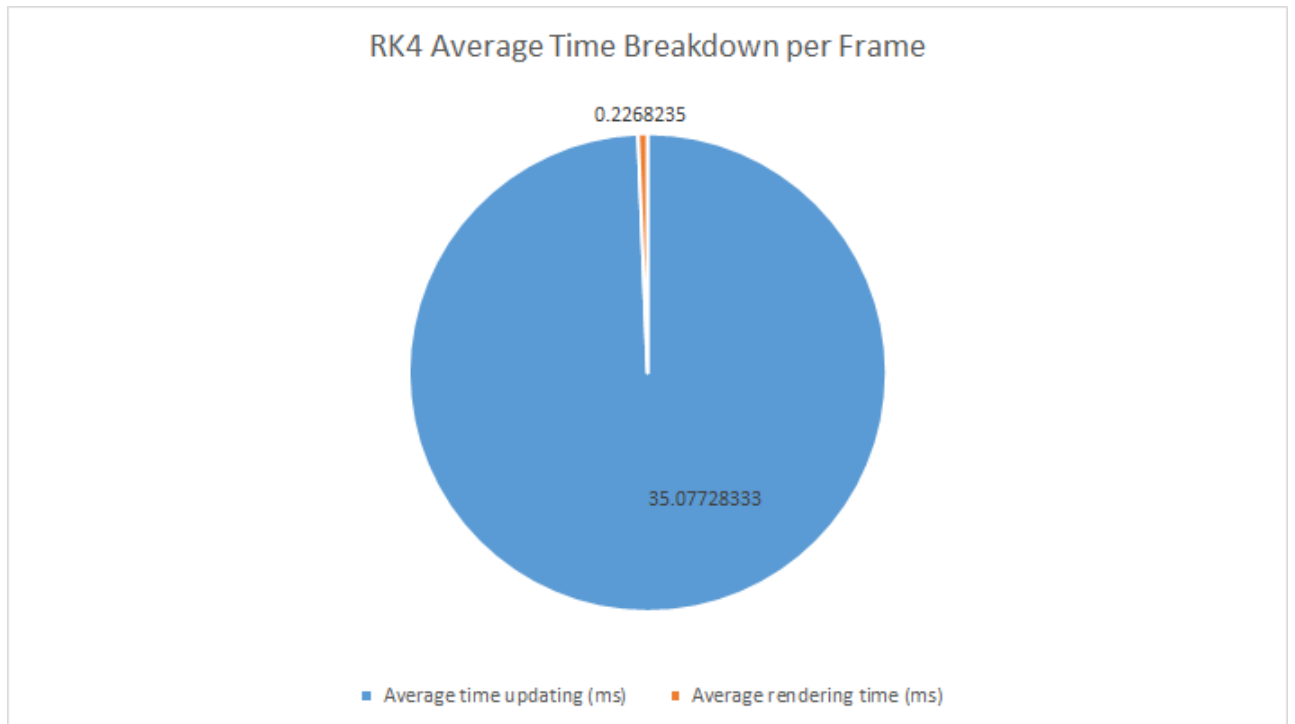
Figure C.97: Fourth Order Runge-Kutta frame time breakdown (flag)
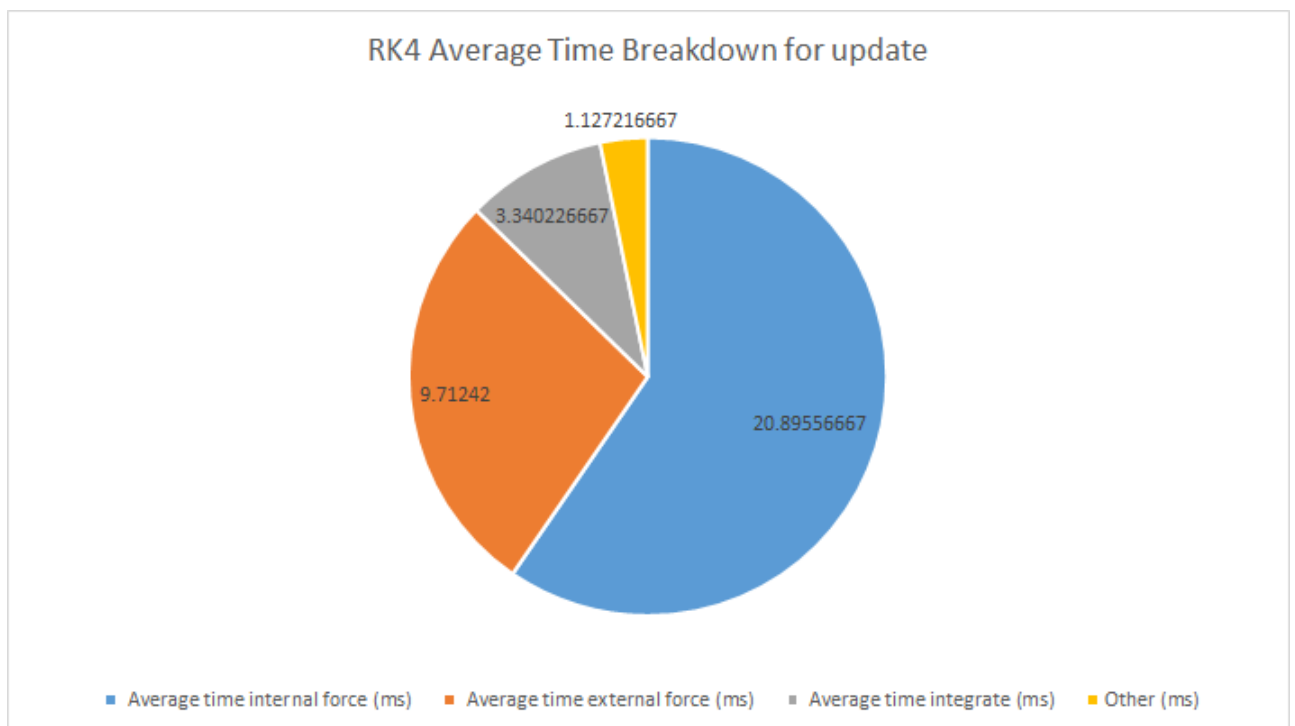
Figure C.98: Fourth Order Runge-Kutta update time breakdown (flag)
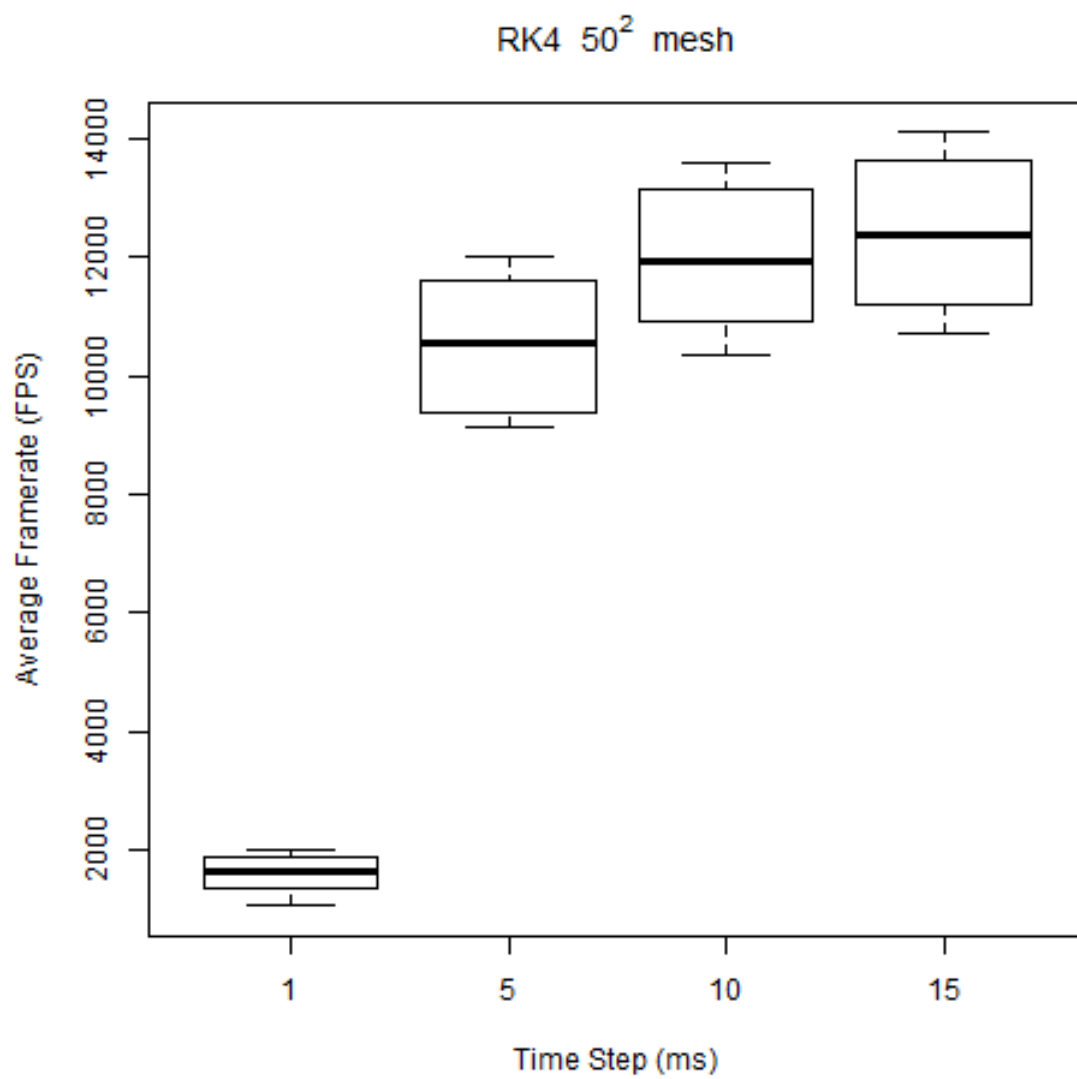
Figure C.99: Fourth order Runge-Kutta box plot for a 50^2 mesh (flag)
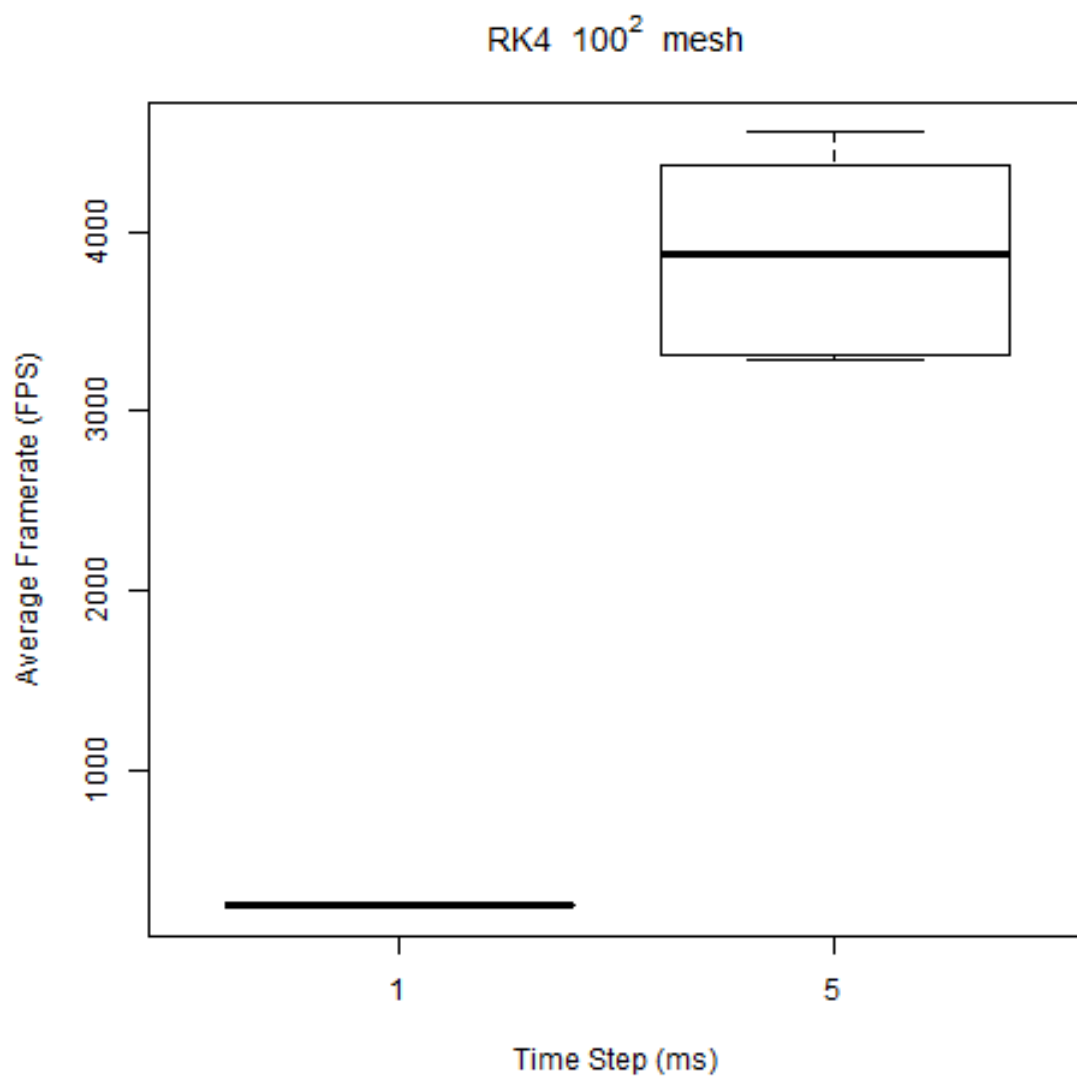
Figure C.100: Fourth order Runge-Kutta box plot for a 100^2 mesh (flag)
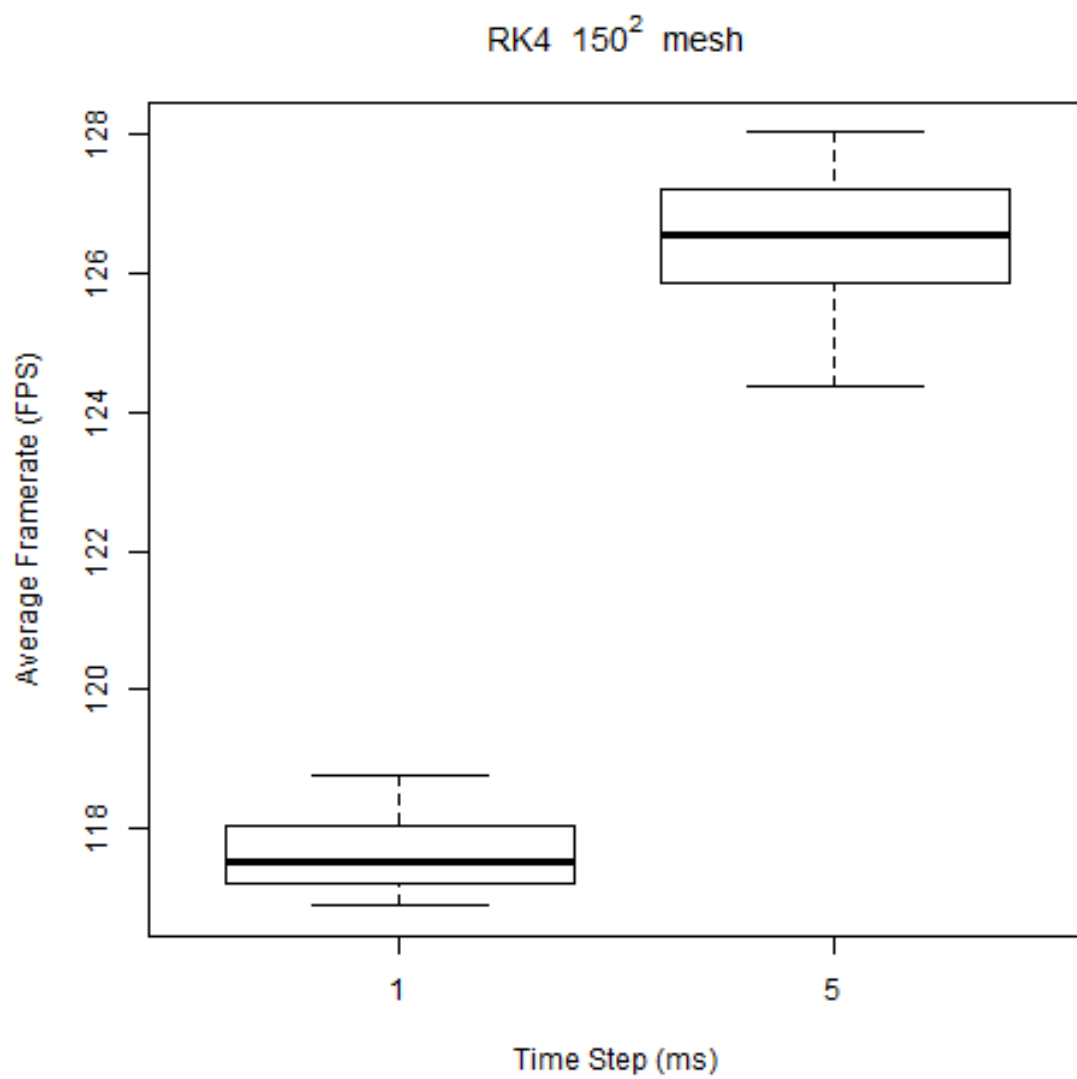
Figure C.101: Fourth order Runge-Kutta box plot for a 150^2 mesh (flag)
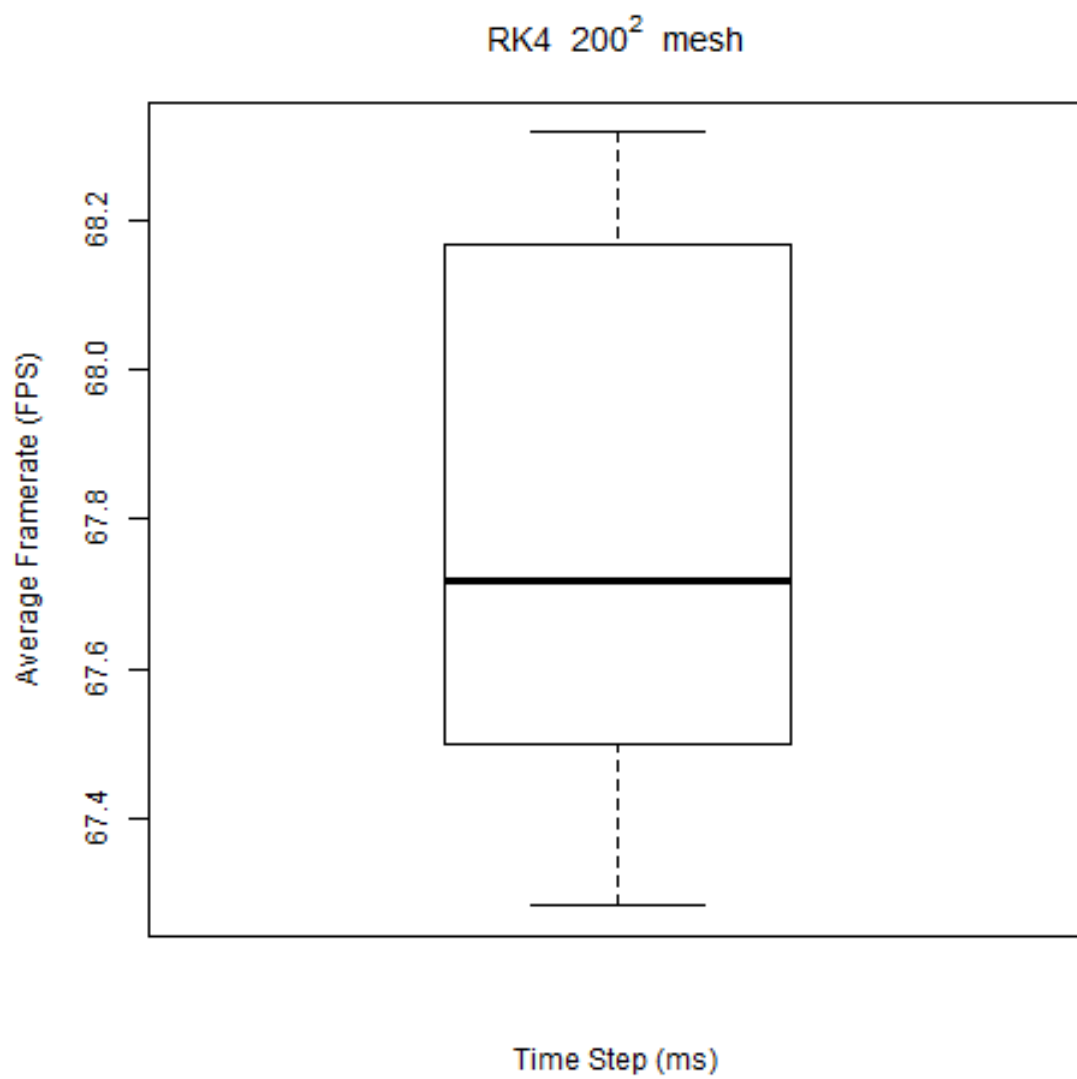
Figure C.102: Fourth order Runge-Kutta box plot for a 200^2 mesh (flag)
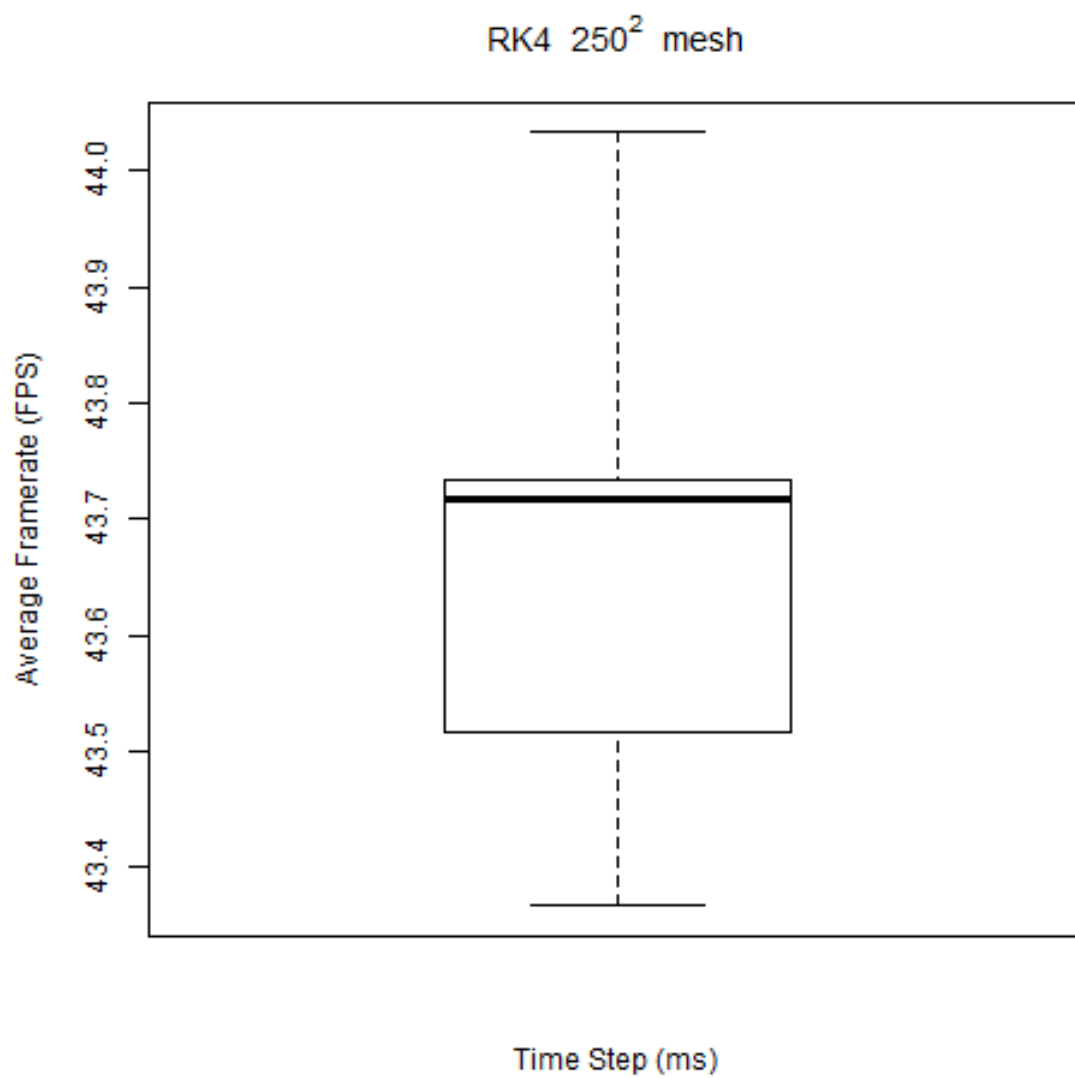
Figure C.103: Fourth order Runge-Kutta box plot for a 250^2 mesh (flag)

RK4 300^2 mesh

Average Framerate (FPS)
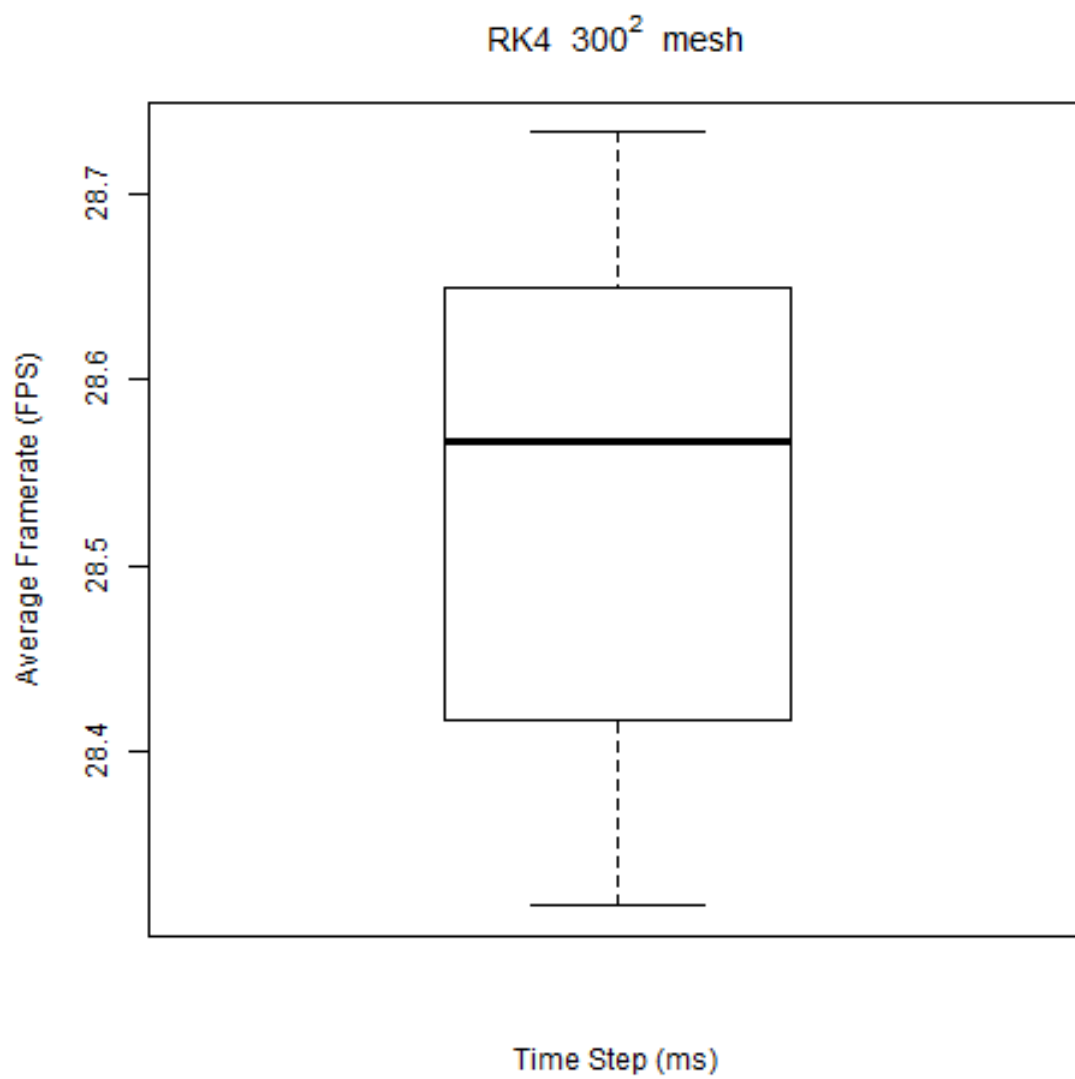
Time Step (ms)

Figure C.104: Fourth order Runge-Kutta box plot for a 300^2 mesh (flag)

Mesh Size	Time Step (ms)	Stable/Unstable
50^2	1	Stable
50^2	5	Stable
50^2	10	Stable
50^2	15	Stable
50^2	20	Unstable
100^2	1	Stable
100^2	5	Stable
100^2	10	Unstable
100^2	15	Unstable
100^2	20	Unstable
150^2	1	Stable
150^2	5	Stable
150^2	10	Unstable
150^2	15	Unstable
150^2	20	Unstable
200^2	1	Stable
200^2	5	Unstable
200^2	10	Unstable
200^2	15	Unstable
200^2	20	Unstable
250^2	1	Stable
250^2	5	Unstable
250^2	10	Unstable
250^2	15	Unstable
250^2	20	Unstable
300^2	1	Stable
300^2	5	Unstable
300^2	10	Unstable
300^2	15	Unstable
300^2	20	Unstable

Table C.16: Stability results for fourth order Runge-Kutta (flag)

C.5   All Integrators
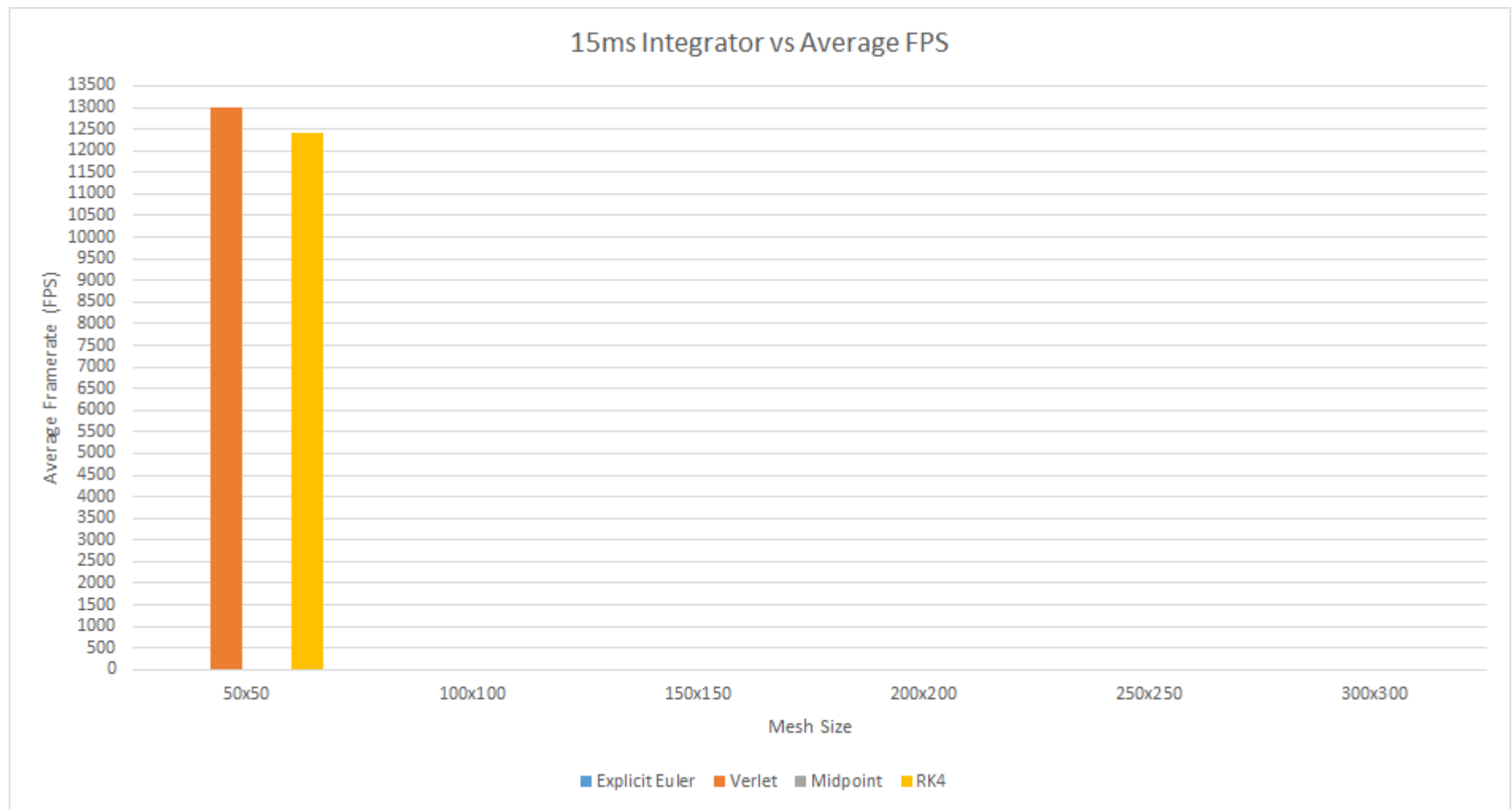
C.5.1   Sheet Data
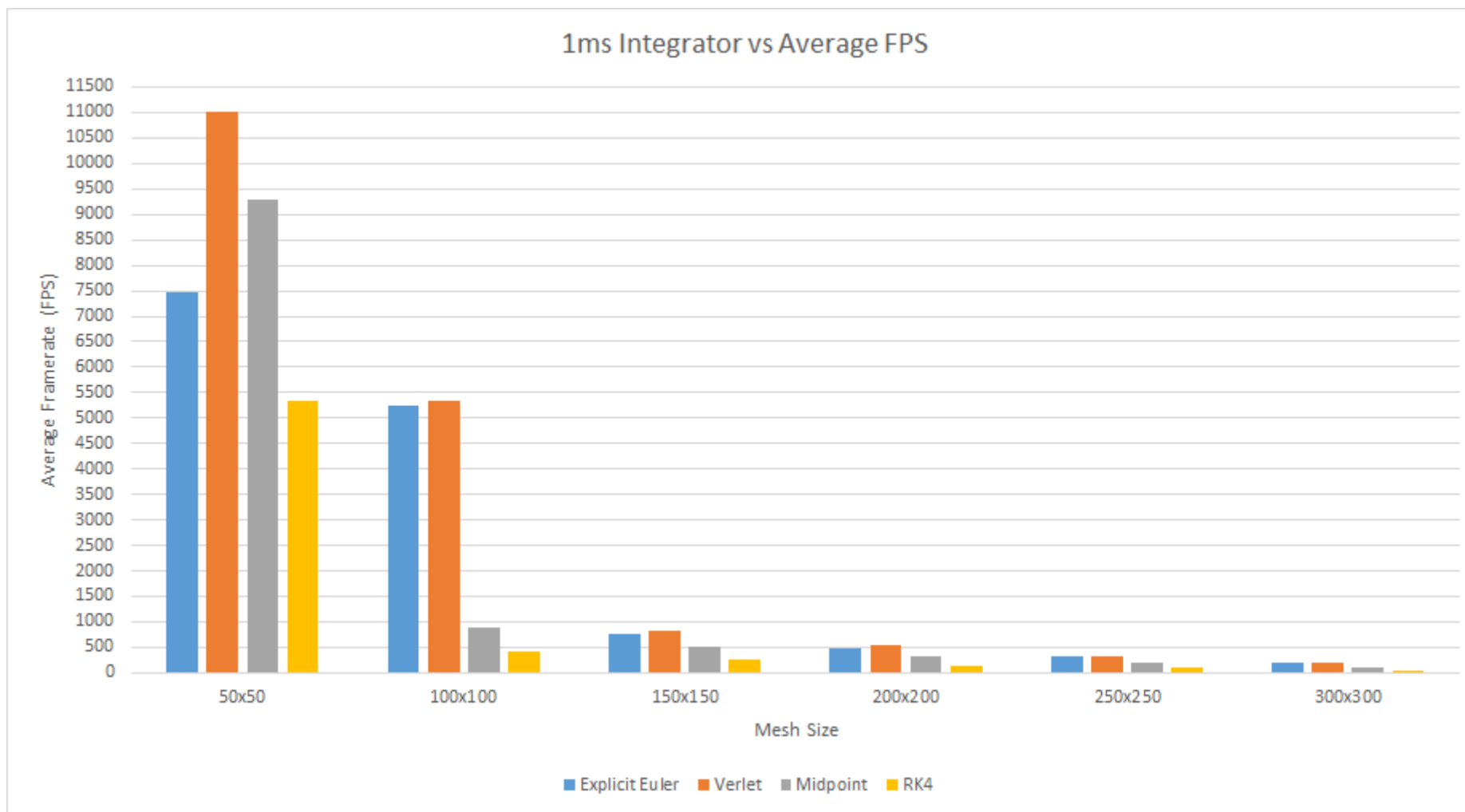
Figure C.105: Mesh size against average FPS for all integrators with 1ms time step (sheet)
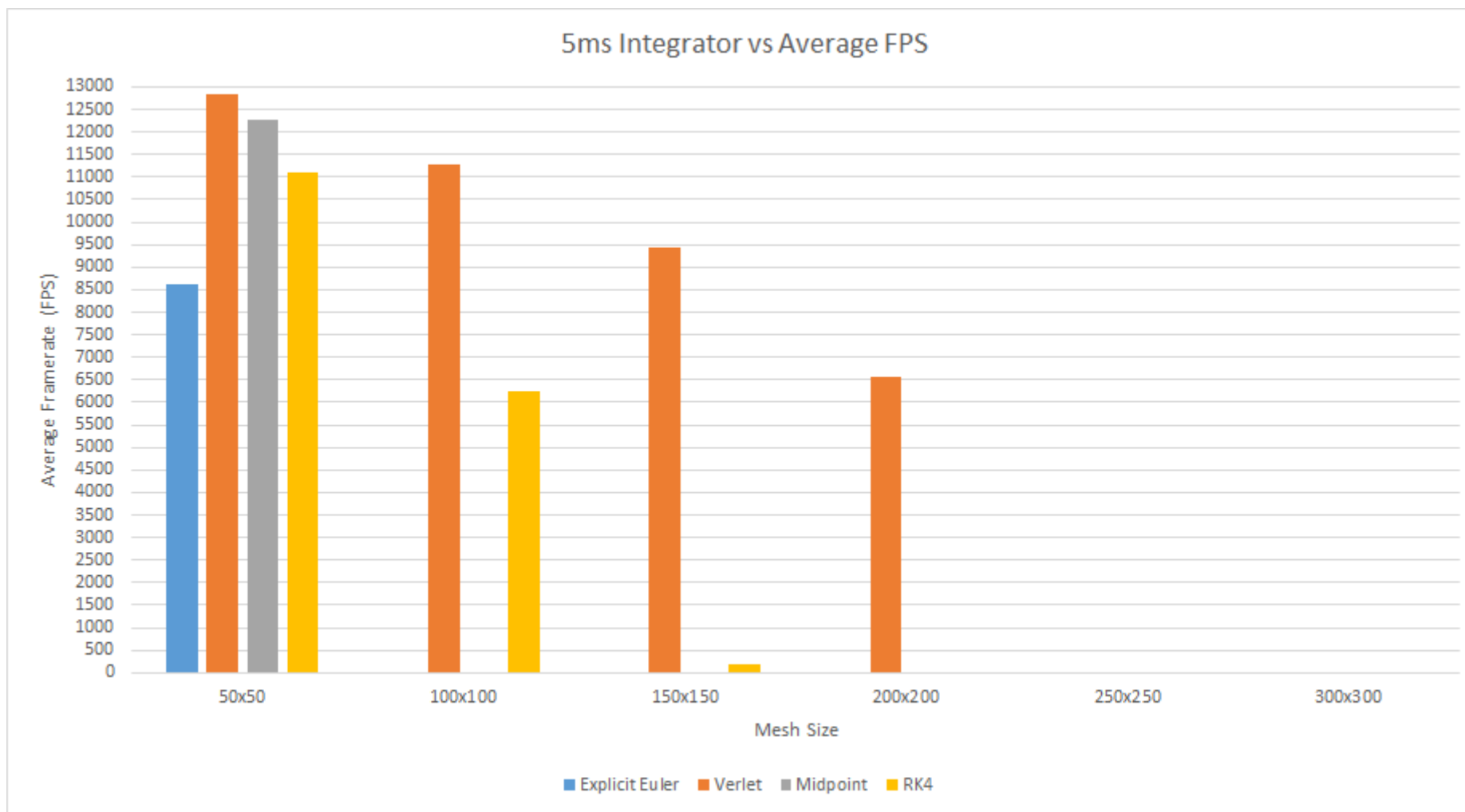
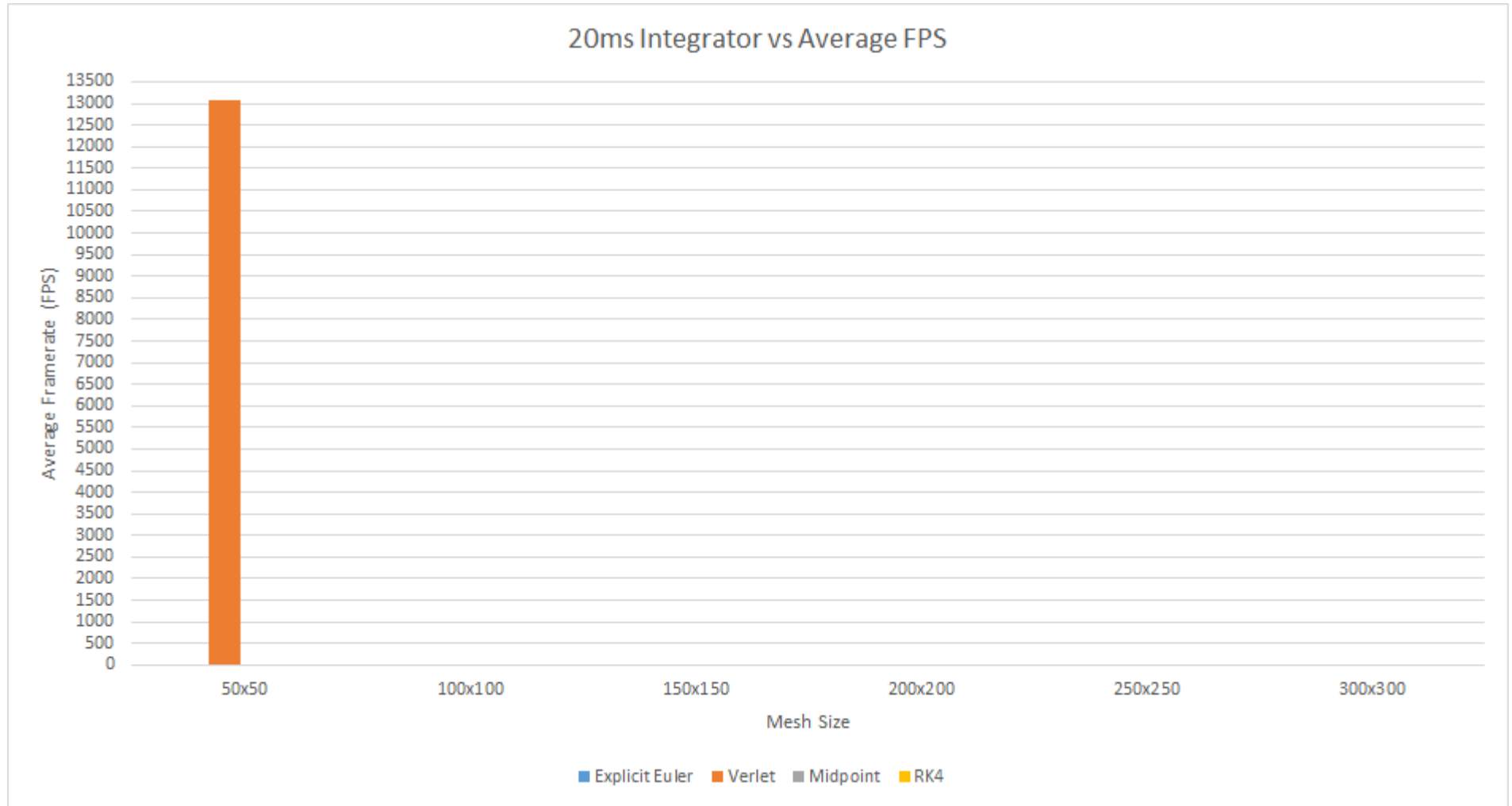Figure C.106: Mesh size against average FPS for all integrators with 5ms time step (sheet)
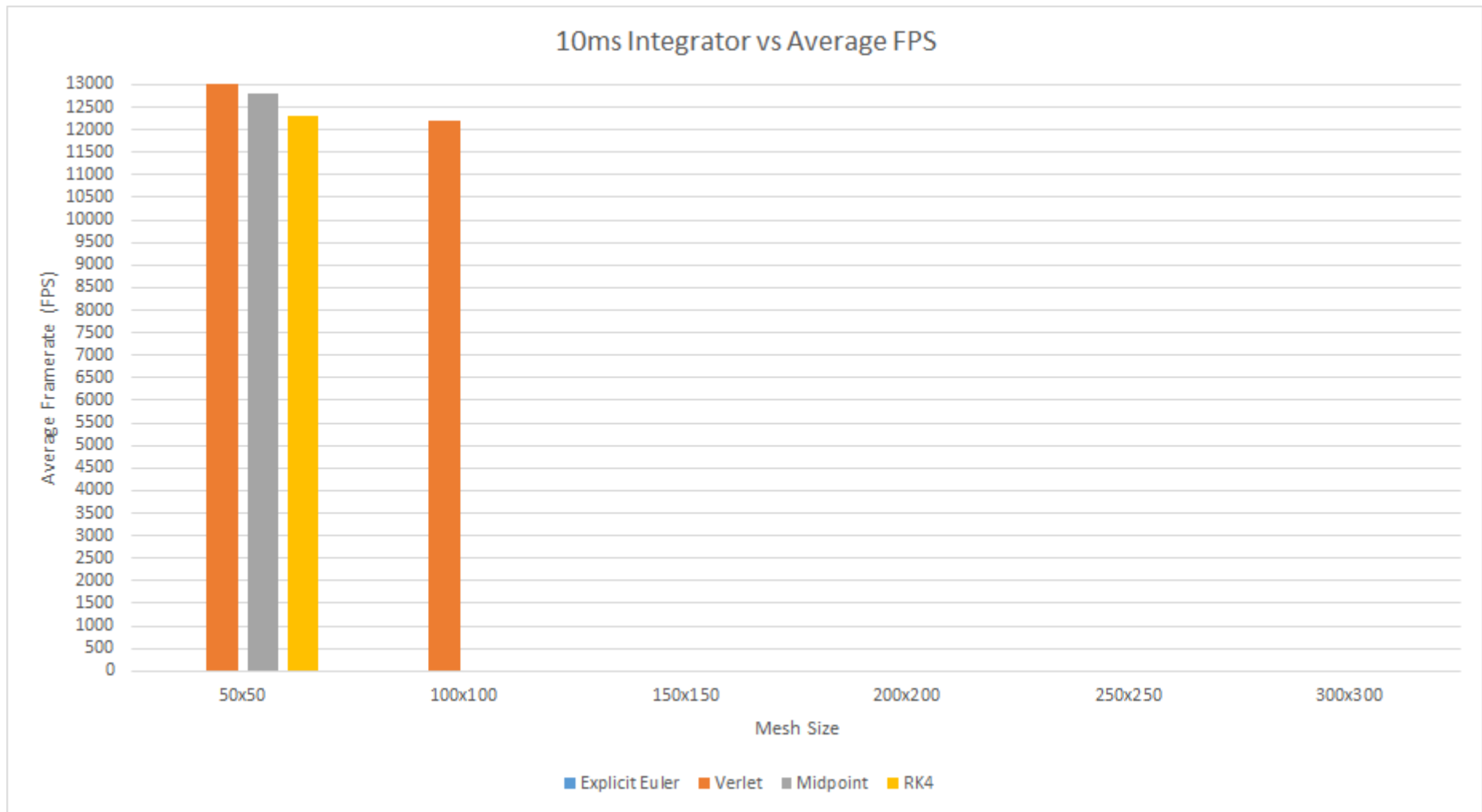
Figure C.107: Mesh size against average FPS for all integrators with 10ms time step (sheet)
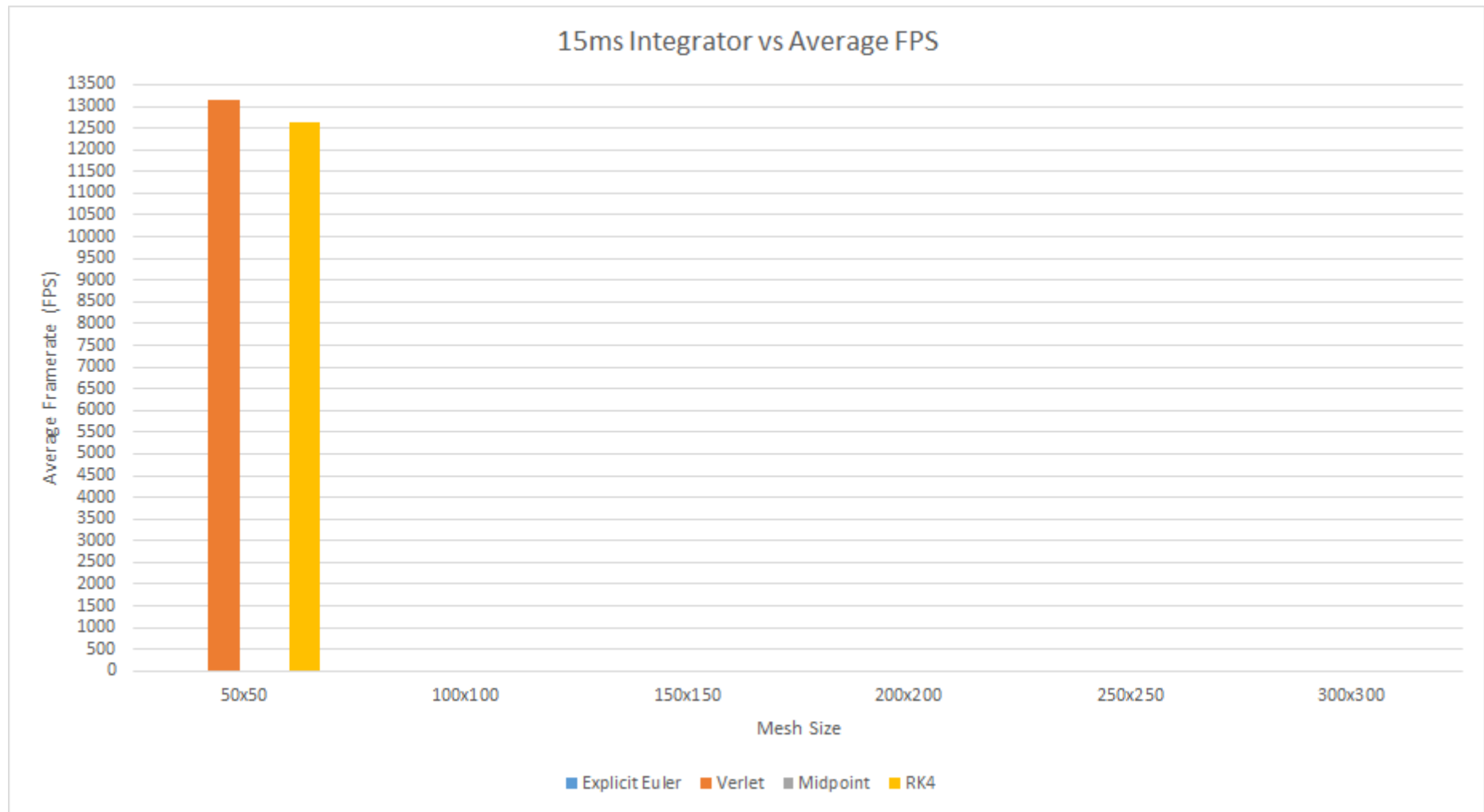
Figure C.108: Mesh size against average FPS for all integrators with 15ms time step (sheet)
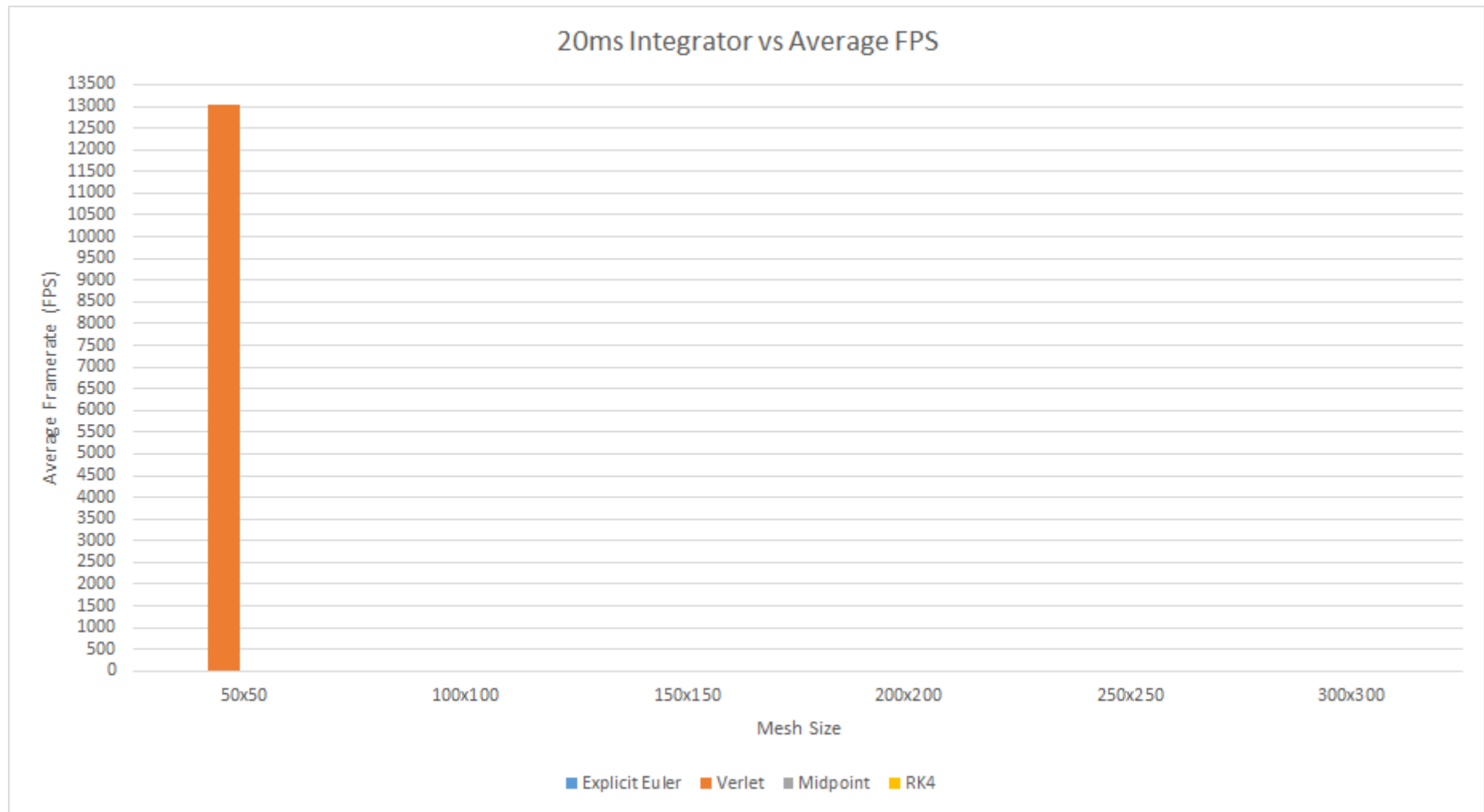
Figure C.109: Mesh size against average FPS for all integrators with 20ms time step (sheet)
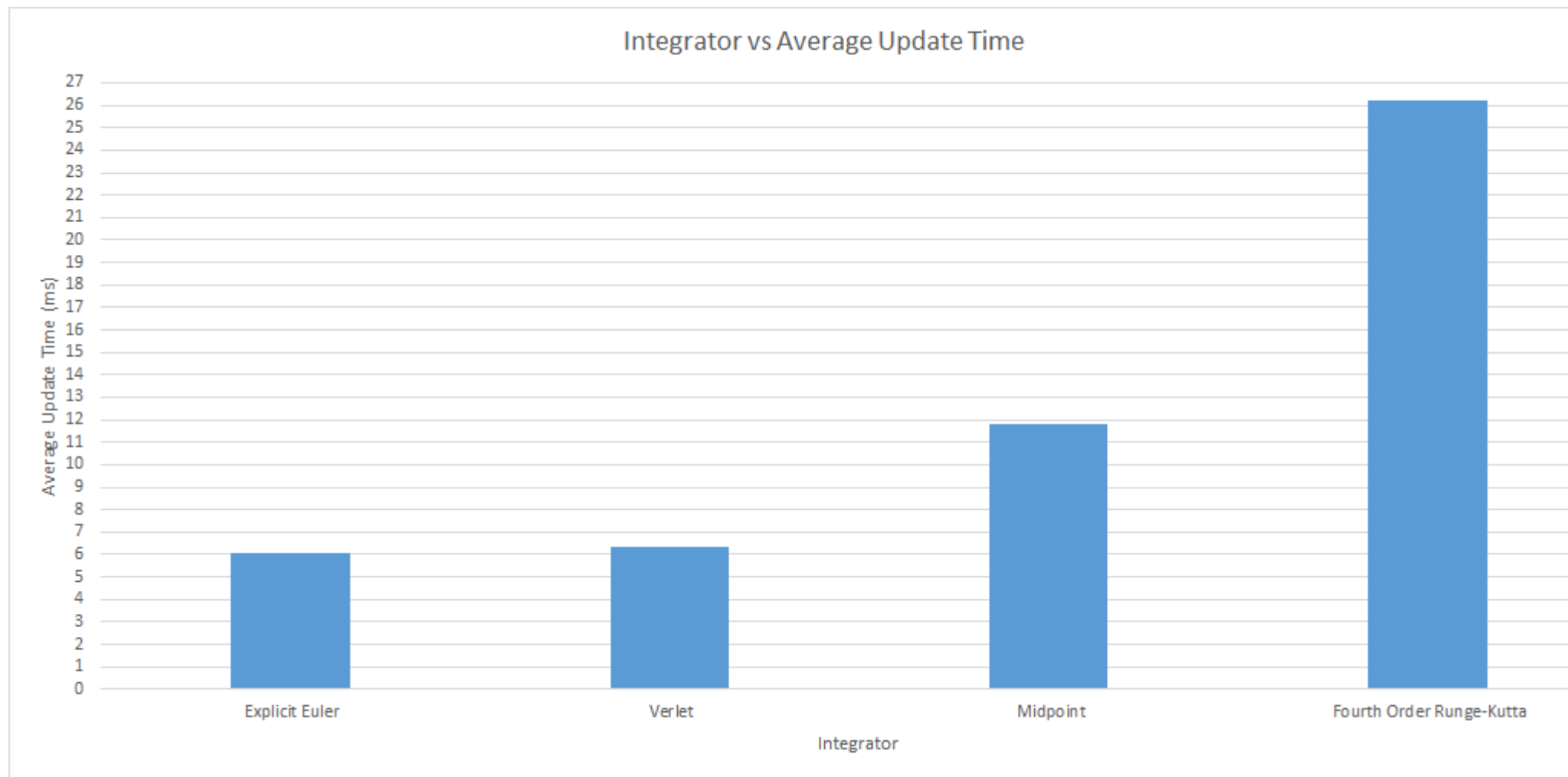
Figure C.110: Average update time for each integrator (sheet)

C.5.2 Flag Data
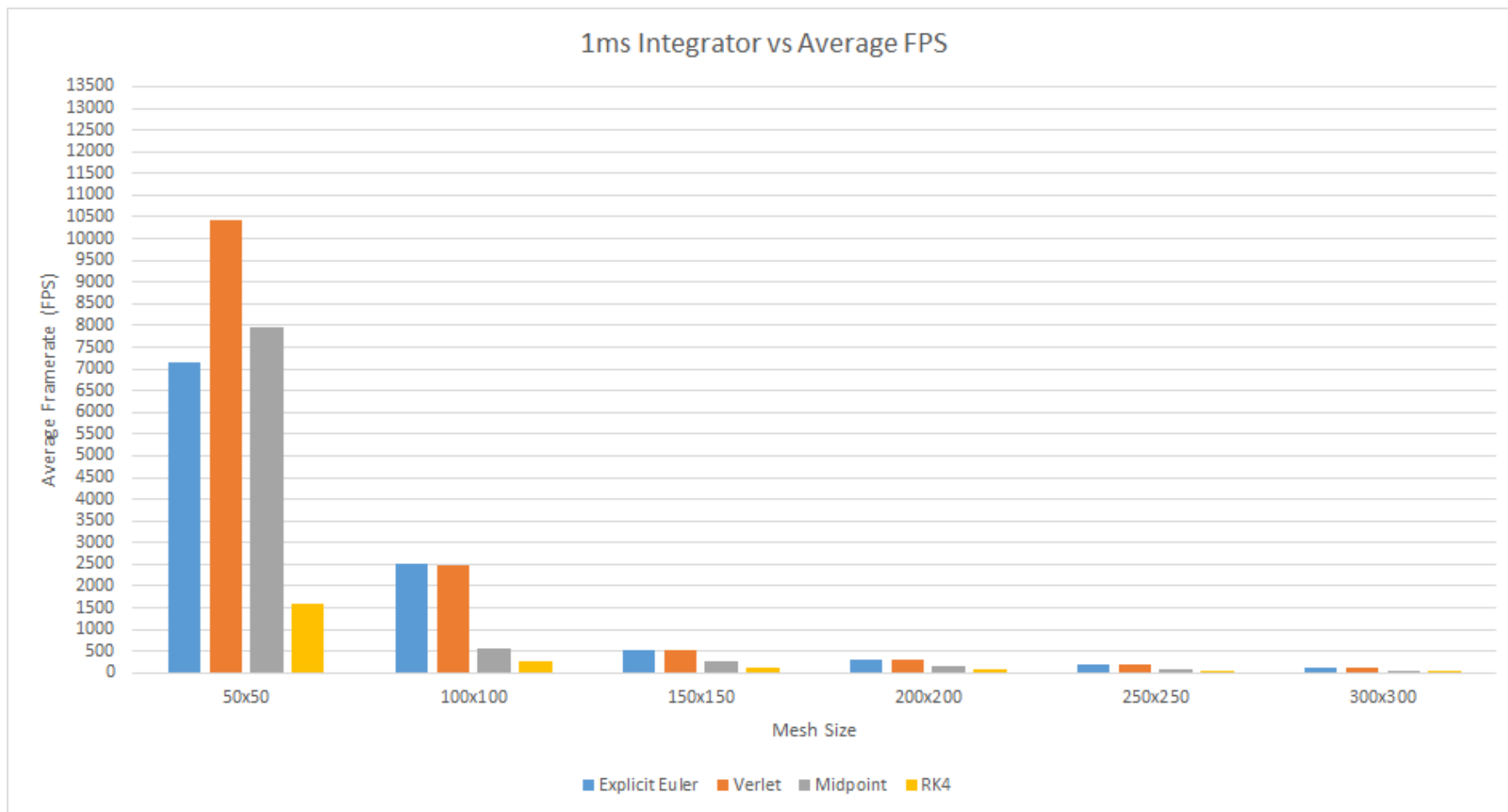
Figure C.111: Mesh size against average FPS for all integrators with 1ms time step (flag)
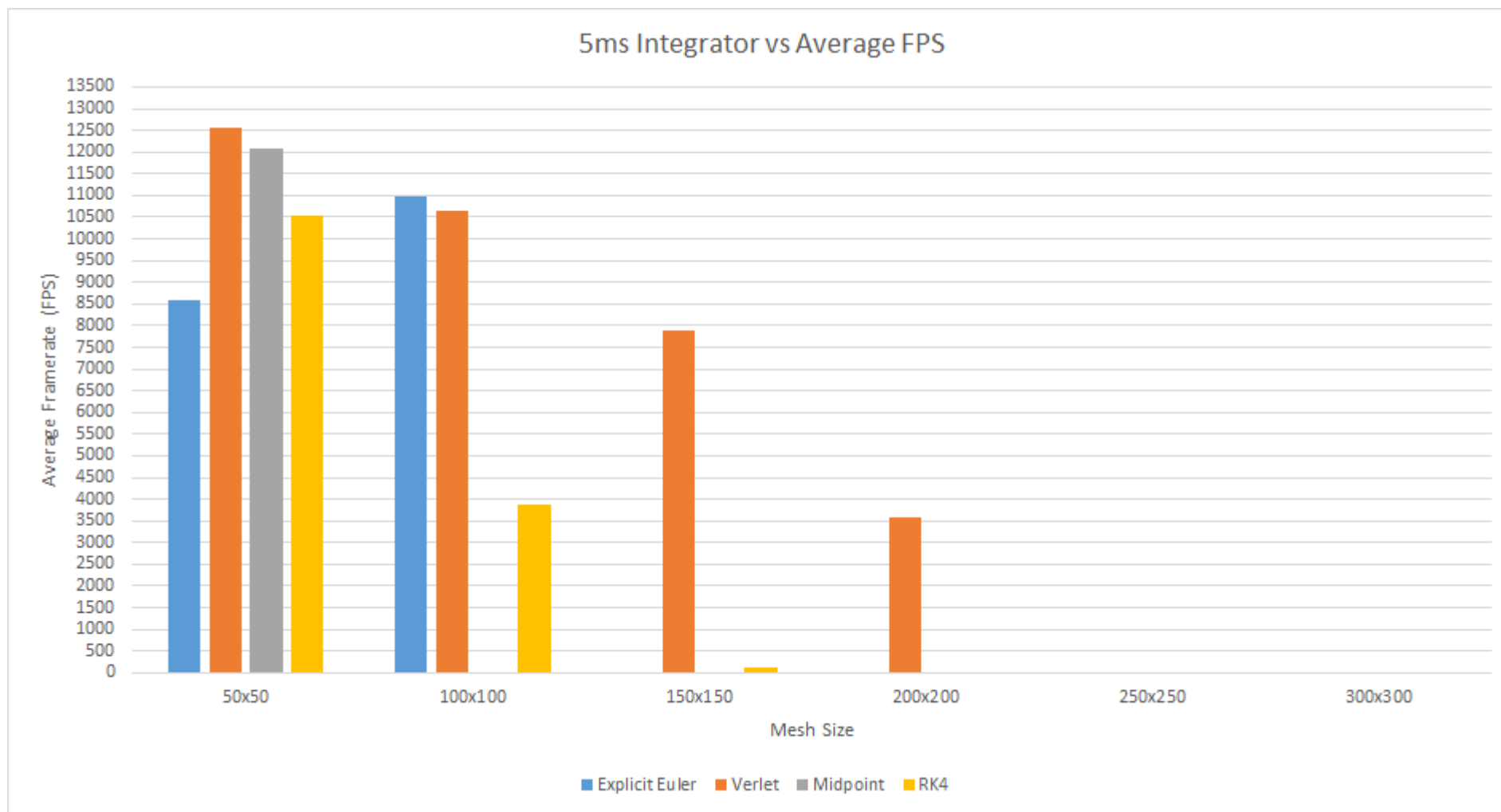
Figure C.112: Mesh size against average FPS for all integrators with 5ms time step (flag)
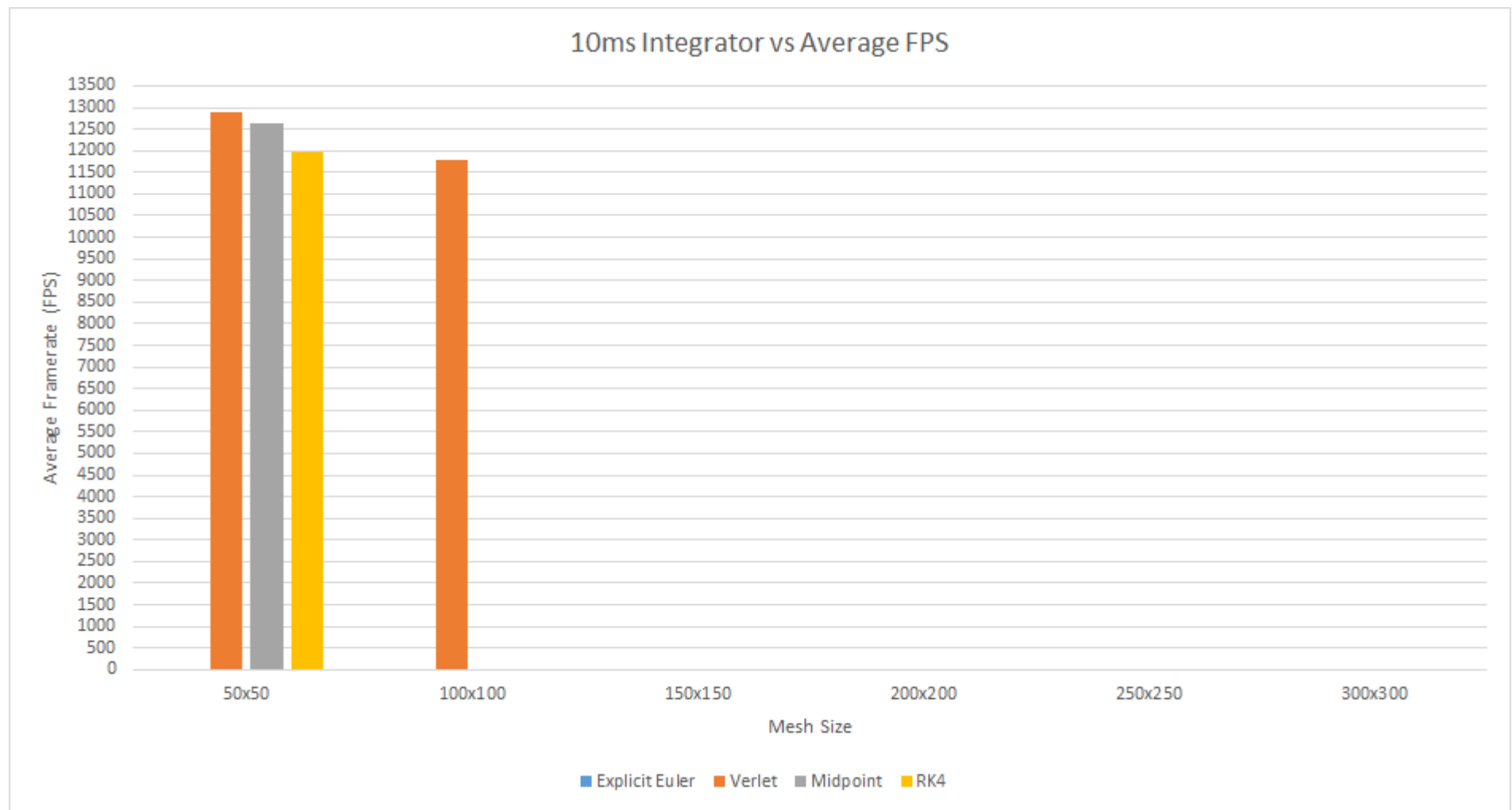
Figure C.113: Mesh size against average FPS for all integrators with 10ms time step (flag)

Figure C.114: Mesh size against average FPS for all integrators with 15ms time step (flag)

Figure C.115: Mesh size against average FPS for all integrators with 20ms time step (flag)
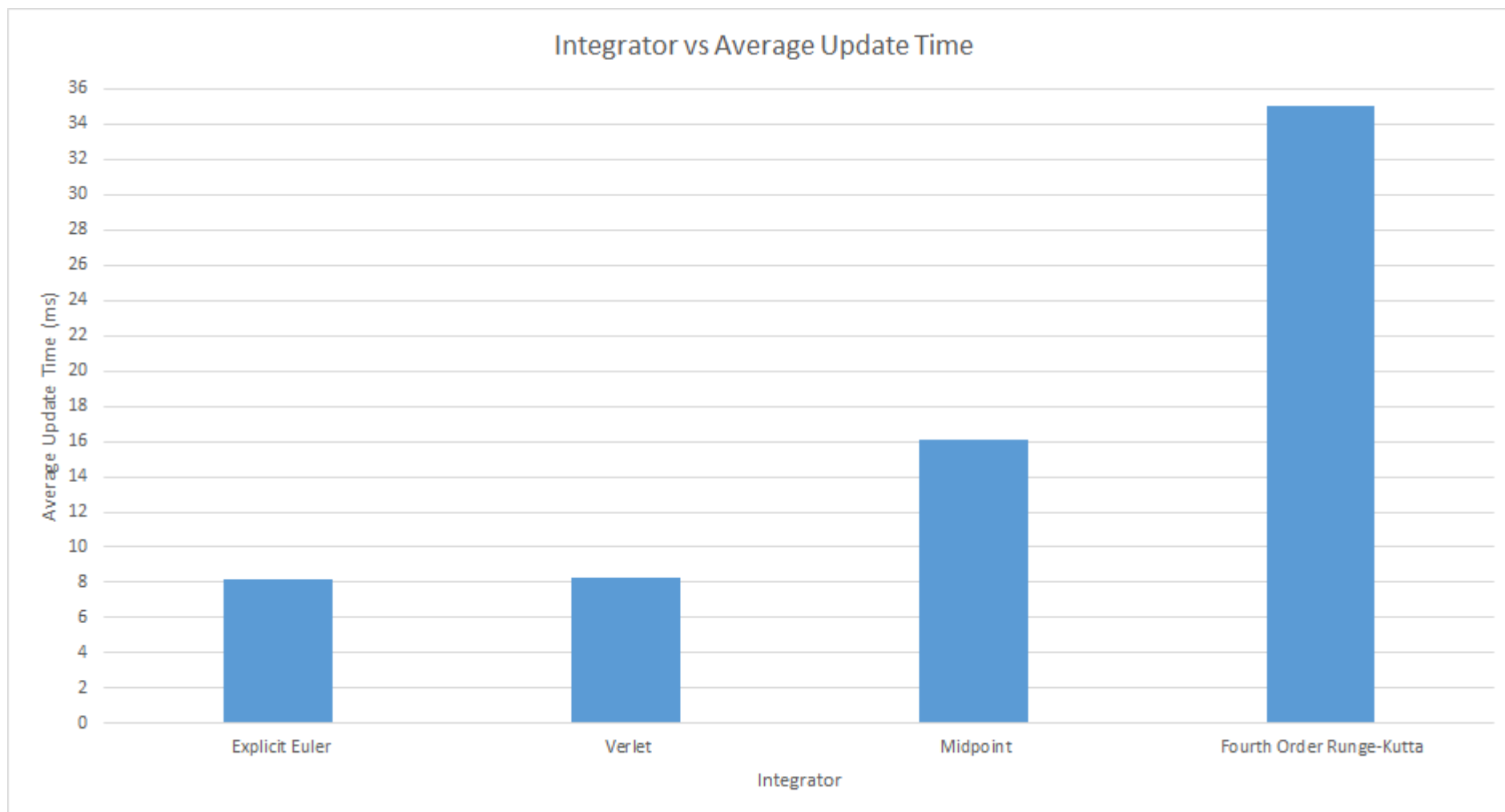
Figure C.116: Average update time for each integrator (flag)

References

Baraff, David and Andrew Witkin (1998). "Large Steps in Cloth Simulation." In: *SIGGRAPH*, pp. 43–54.

Bartels, Pieterjan (2014). *Simulation of Tearable Cloth*. URL: `https://nccastaff.bournemouth.ac.uk/jmacey/CGITech/reports2015/PBartels_PieterjanBartels_CGITECH_final.pdf` (visited on 03/10/2016).

Choi, Kwang-Jin and Hyeong-Seok Ko (2002). "Stable but Responsive Cloth." In: *SIGGRAPH*, pp. 604–611.

Enqvist, Henrik (2010). *The Secrets Of Cloth Simulation In Alan Wake*. URL: `http://www.gamasutra.com/view/feature/132771/the_secrets_of_cloth_simulation_in_.php` (visited on 03/03/2016).

Jakobsen, Thomas (2005). *Advanced Character Physics*. URL: `http://www.gotoandplay.it/_articles/2005/08/advCharPhysics.php` (visited on 03/03/2016).

Kang, Young-Min, Jeong-Hyeon Choi, and Hwan-Gue Cho (2000). "Fast and Stable Animation of Cloth with an Approximated Implicit Method." In: *Computer Graphics International*. Geneva.

Kim, Tae-Yong (2011). *Character Clothing in PhysX-3*.

Lander, Jeff (2000b). *Devil in the Blue Faceted Dress: Real Time Cloth Animation*. URL: `http://www.gamasutra.com/view/feature/131851/devil_in_the_blue_faceted_dress_.php` (visited on 03/03/2016).

Magnenat-Thalmann, Nadia and D. Thalmann (2004). *Handbook of Virtual Humans*. 1st. Chichester: Wiley.

Mesit, Jaruwan, Ratan Guha, and Shafaq Chaudhry (2007). "3D soft body simulation using mass-spring system with internal pressure force and simplified implicit integration." In: *Journal of Computers*.

Mongus, D. et al. (2012). "A hybrid evolutionary algorithm for tuning a cloth-simulation model." In: *Applied Soft Computing Journal*.

Mosegaard, Jesper (2009). *Mosegaards Cloth Simulation Coding Tutorial*. URL: `http://cg.alexandra.dk/?p=147` (visited on 07/05/2016).

Müller, Matthias et al. (2006). "Position Based Dynamics." In: *VRIPHYS*, pp. 71–80.

Naveen (2015). *What is Waterfall Model in software testing and what are advantages and disadvantages of Waterfall Model*. URL: `http://testingfreak.com/waterfall-model-software-testing-advantages-disadvantages-waterfall-model/` (visited on 06/15/2016).

Ng, Hing N. and Richard L. Grimsdale (1996). "Computer Graphics Techniques for Modeling Cloth." In: *IEEE Computer Graphics & Applications* 16, pp. 28–42.

Parent, Rick (2012). *Computer Animation Algorithms and Techniques*. Third. Waltham: Elsevier.

Provot, Xavier (1995). "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour." In: *Graphics Interface'95*, pp. 141–155.

Tang, Min et al. (2013). "A GPU-based streaming algorithm for high-resolution cloth simulation." In: *Computer Graphics Forum*.

Vassilev, T, B Spanlang, and Y Chrysanthou (2001). "Fast Cloth Animation on Walking Avatars." In: *Eurographics*.

Volino, Pascal, Frederic Cordier, and Nadia Magnenat-Thalmann (2005). "From early virtual garment simulation to interactive fashion design." In: *Computer-Aided Design* 37.6, pp. 593–608.

Volino, Pascal and Nadia Magnenat-Thalmann (2001). "Comparing Efficiency of Integration Methods." In: *Conference on Computer Graphics International*.

Wacker, Markus, Bernhard Thomaszewski, and Michael Keckeisen (2005). "Physical Models, Numerical Solvers for Cloth Animation and Virtual Cloth Design." In: *Eurographics*.

Wang, Jianchun, Xinrong Hu, and Yi Zhuang (2009). "The dynamic cloth simulation performance analysis based on the improved spring-mass model." In: *International Conference on Wireless Networks and Information Systems, WNIS 2009*, pp. 282 –285.

Xinrong, Hu et al. (2009). "Review of cloth modeling." In: *2009 Second ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2009*.

Yalçin, M Adil and Cansin Yildiz (2009). *Techniques for Animating Cloth*. URL: `http://www.cs.bilkent.edu.tr/~cansin/projects/cs567-animation/cloth/cloth-paper.pdf` (visited on 05/25/2016).

Zeller, Cyril (2005). "Cloth Simulation On The GPU." In: *SIGGRAPH*.

Zhang, Dongliang and Matthew M F Yuen (2001). "Cloth simulation using multilevel meshes." In: *Computers and Graphics (Pergamon)*.

Zink, Nico and Alexandre Hardy (2007). "Cloth Simulation and Collision Detection using Geometry Images." In: *AFRIGRAPH*. New York: ACM, pp. 187–195.

Bibliography

Adriansnetlis (2016). *How does Runge-Kutta 4 work in games*. URL: `https://www.gamedev.net/topic/679116-how-does-runge-kutta-4-work-in-games/` (visited on 06/30/2016).

Akiya, Naoko, Scott Bury, and John M. Wassick (2011). "A General Model for Soft Body Simulation in Motion." In: *Winter Simulation Conference*. 2010, pp. 2194–2205.

Baraff, David and Andrew Witkin (1998). "Large Steps in Cloth Simulation." In: *SIGGRAPH*, pp. 43–54.

Bartels, Pieterjan (2014). *Simulation of Tearable Cloth*. URL: `https://nccastaff.bournemouth.ac.uk/jmacey/CGITech/reports2015/PBartels_PieterjanBartels_CGITECH_final.pdf` (visited on 03/10/2016).

Bender, Jan, Daniel Weber, and Raphael Diziol (2013). "Fast and stable cloth simulation based on multi-resolution shape matching." In: *Computers and Graphics (Pergamon)*.

Bhardwaj, Manas (2013). *A beginner's guide to various Software development methodologies*. URL: `http://manasbhardwaj.net/a-beginners-guide-to-various-software-development-methodologies/` (visited on 06/15/2016).

Bourg, David M. and Bryan Bywalec (2013). *Physics for Game Developers*. Second. Sebastopol: O'Reilly. ISBN: ISBN 9781449361051.

Boxerman, Eddy (2003). "Speeding Up Cloth Simulation." Master's Thesis. The University of British Columbia.

Bridson, Robert, Ronald Fedkiw, and John Anderson (2002). "Robust Treatment of Collisions, Contact and Friction for Cloth Animation." In: *SIGGRAPH*, pp. 594–603.

Choi, Kwang-Jin and Hyeong-Seok Ko (2002). "Stable but Responsive Cloth." In: *SIGGRAPH*, pp. 604–611.

— (2005). "Research problems in clothing simulation." In: *Computer Aided Design* 37, pp. 585–592.

Conger, David (2004). *Physics Modeling for Game Programmers*. First. Boston: Course Technology / Cengage Learning.

De Farias, Thiago S M C et al. (2008). "A high performance massively parallel approach for real time deformable body physics simulation." In: *Proceedings - Symposium on Computer Architecture and High Performance Computing*.

Desbrun, Mathieu, Peter Schröder, and Alan Barr (1999). "Interactive Animation of Structured Deformable Objects." In: *Graphics Interface*. San Francisco: Morgan Kaufmann Publishers Inc., pp. 1–8.

Endre, Simó (2011). *Cloth simulation with RK4 numerical integration*. URL: `http://www.esimov.com/2011/07/cloth-simulation-with-rk4-numerical-integration#.V4YzVjVpuvY` (visited on 07/13/2016).

Enqvist, Henrik (2010). *The Secrets Of Cloth Simulation In Alan Wake*. URL: `http://www.gamasutra.com/view/feature/132771/the_secrets_of_cloth_simulation_in_.php` (visited on 03/03/2016).

Ertekin, Burak (2014). *Cloth Simulation*. URL: `https://nccastaff.bournemouth.ac.uk/jmacey/CGITech/reports2015/BErtekin_burakertekin-cgitech.pdf` (visited on 03/10/2016).

Fielder, Glenn (2006). *Integration Basics*. URL: `http://gafferongames.com/game-physics/integration-basics/` (visited on 06/30/2016).

Garre, Carlos and Alvaro Pérez (2008). *A simple Mass-Spring system for Character Animation*. URL: `http://www.gmrv.es/~cgarre/APO_MassSpring_Jun08.pdf` (visited on 05/25/2016).

Gibson, Sarah F. F. and Brian Mirtich (1997). *A Survey of Deformable Modeling in Computer Graphics*. Tech. rep. Cambridge: MERL.

Harada, Takahiro (2006). "Real-time Cloth Simulation Interacting with Deforming High-Resolution Models." In: *SIGGRAPH*.

— (2008). "Real-Time Rigid Body Simulation on GPUs." In: *GPU Gems 3*. Pearson Education Inc. Chap. 29. URL: `http://http.developer.nvidia.com/GPUGems3/gpugems3_ch29.html`.

Jakobsen, Thomas (2005). *Advanced Character Physics*. URL: `http://www.gotoandplay.it/_articles/2005/08/advCharPhysics.php` (visited on 03/03/2016).

Kang, Young-Min, Jeong-Hyeon Choi, and Hwan-Gue Cho (2000). "Fast and Stable Animation of Cloth with an Approximated Implicit Method." In: *Computer Graphics International*. Geneva.

Kim, Tae-Yong (2011). *Character Clothing in PhysX-3*.

Lander, Jeff (2000a). *Collision Response: Bouncy, Trouncy, Fun*. URL: `http://www.gamasutra.com/view/feature/3427/collision_response_bouncy_.php` (visited on 03/24/2016).

— (2000b). *Devil in the Blue Faceted Dress: Real Time Cloth Animation*. URL: `http://www.gamasutra.com/view/feature/131851/devil_in_the_blue_faceted_dress_.php` (visited on 03/03/2016).

Lovrovic, Bojan (2014). *Real Time Cloth Simulation with B-spline Surfaces - Graphics Programming and Theory - Articles - Articles - GameDev.net*. URL: `http://www.gamedev.net/page/resources/_/technical/graphics-programming-and-theory/real-time-cloth-simulation-with-b-spline-surfaces-r3814` (visited on 03/03/2016).

Magnenat-Thalmann, Nadia and D. Thalmann (2004). *Handbook of Virtual Humans*. 1st. Chichester: Wiley.

Mesit, Jaruwan, Ratan Guha, and Shafaq Chaudhry (2007). "3D soft body simulation using mass-spring system with internal pressure force and simplified implicit integration." In: *Journal of Computers*.

Mesit, Jaruwan and Ratan K. Guha (2008). "Soft Body Simulation with Leaking Effect." In: *2008 Second Asia International Conference on Modelling & Simulation (AMS)*. IEEE, pp. 390–395.

Miguel, E et al. (2012). "Data-Driven Estimation of Cloth Simulation Models." In: *Computer Graphics Forum* 31.2, pp. 519–528.

Mongus, D. et al. (2012). "A hybrid evolutionary algorithm for tuning a cloth-simulation model." In: *Applied Soft Computing Journal*.

Mosegaard, Jesper (2009). *Mosegaards Cloth Simulation Coding Tutorial*. URL: `http://cg.alexandra.dk/?p=147` (visited on 07/05/2016).

Müller, Matthias et al. (2006). "Position Based Dynamics." In: *VRIPHYS*, pp. 71–80.

Naveen (2015). *What is Waterfall Model in software testing and what are advantages and disadvantages of Waterfall Model*. URL: `http://testingfreak.com/waterfall-model-software-testing-advantages-disadvantages-waterfall-model/` (visited on 06/15/2016).

Ng, Hing N. and Richard L. Grimsdale (1996). "Computer Graphics Techniques for Modeling Cloth." In: *IEEE Computer Graphics & Applications* 16, pp. 28–42.

NVidia (2007). *Cloth Simulation*. URL: `http://developer.download.nvidia.com/SDK/10/direct3d/Source/Cloth/doc/Cloth.pdf` (visited on 05/25/2016).

O 'connor, Corey and Keith Stevens (2003). *Modeling Cloth Using Mass Spring Systems*. (Visited on 03/10/2016).

Ozgen, Oktar et al. (2010). "Underwater Cloth Simulation with Fractional Derivatives." In: *ACM Trans. Graph* 29.23.

Parent, Rick (2012). *Computer Animation Algorithms and Techniques*. Third. Waltham: Elsevier.

Plath, Jan (2000). "Realistic modelling of textiles using interacting particle systems." In: *Computers & Graphics* 24, pp. 897–905.

Provot, Xavier (1995). "Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour." In: *Graphics Interface'95*, pp. 141–155.

RubyNL (2012). *Rope/cloth simulation problems*. URL: `https://www.gamedev.net/topic/626870-ropecloth-simulation-problems/` (visited on 03/03/2016).

Santos Souza, Marco, Aldo Von Wangenheim, and Eros Comunello (2014). "Fast Simulation of Cloth Tearing." In: *SBC Journal on Interactive Systems* 5.1, pp. 44–48.

Schmitt, N et al. (2013). "Multilevel Cloth Simulation using GPU Surface Sampling." In: *Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS*.

Scrum Alliance. *Learn About Scrum*. URL: `https://www.scrumalliance.org/why-scrum` (visited on 06/15/2016).

Selle, Andrew et al. (2009). "Robust High-Resolution Cloth Using Parallelism, History-Based Collisions and Accurate Friction." In: *IEEE Transactions on Visualization and Computer Graphics* 15, pp. 339–350.

Shadx (2006). *Runge Kutta 4 For cloth simulation*. URL: `https://www.gamedev.net/topic/368698-runge-kutta-4-for-cloth-simulation/` (visited on 06/30/2016).

Tang, Min et al. (2013). "A GPU-based streaming algorithm for high-resolution cloth simulation." In: *Computer Graphics Forum*.

Vassilev, T, B Spanlang, and Y Chrysanthou (2001). "Fast Cloth Animation on Walking Avatars." In: *Eurographics*.

Vassilev, Tzvetomir and Bernhard Spanlang (2002). "A Mass-Spring Model For Real Time Deformable Solids." In: *East-West Vision*.

Vassilev, Tzvetomir Ivanov (2010). "Comparison of Several Parallel API for Cloth Modelling On Modern GPUs." In: *CompSysTech*, pp. 131–136.

— (2011). "Comparison of Parallel Algorithms for Modelling Mass-springs Systems with Several APIs on Modern GPUs." In: *CompSysTech*, pp. 204–209.

Volino, Pascal, Frederic Cordier, and Nadia Magnenat-Thalmann (2005). "From early virtual garment simulation to interactive fashion design." In: *Computer-Aided Design* 37.6, pp. 593–608.

Volino, Pascal, Martin Courchesne, and Nadia Magnenat-Thalmann (1995). "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects." In: *SIGGRAPH*, pp. 137–144.

Volino, Pascal and Nadia Magnenat-Thalmann (1997a). "Developing Simulation Techniques for an Interactive Clothing System." In: *International Conference on Virtual Systems and MultiMedia*. IEEE, pp. 109–118.

— (1997b). "Interactive Cloth Simulation: Problems and Solutions." In: *JWS97-B*. Geneva.

— (2001). "Comparing Efficiency of Integration Methods." In: *Conference on Computer Graphics International*.

Wacker, Markus, Bernhard Thomaszewski, and Michael Keckeisen (2005). "Physical Models, Numerical Solvers for Cloth Animation and Virtual Cloth Design." In: *Eurographics*.

Wang, Jianchun, Xinrong Hu, and Yi Zhuang (2009). "The dynamic cloth simulation performance analysis based on the improved spring-mass model." In: *International Conference on Wireless Networks and Information Systems, WNIS 2009*, pp. 282 –285.

Wang, Xiuzhong and Venkat Devarajan (2004). "2D Structured Mass-spring System Parameter Optimization based on Axisymmetric Bending for Rigid Cloth Simulation." In: *SIGGRAPH*, pp. 317–323.

Xinrong, Hu et al. (2009). "Review of cloth modeling." In: *2009 Second ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2009*.

Yalçin, M Adil and Cansin Yildiz (2009). *Techniques for Animating Cloth*. URL: `http://www.cs.bilkent.edu.tr/~cansin/projects/cs567-animation/cloth/cloth-paper.pdf` (visited on 05/25/2016).

Zeller, Cyril (2005). "Cloth Simulation On The GPU." In: *SIGGRAPH*.

Zhang, Dongliang and Matthew M F Yuen (2001). "Cloth simulation using multilevel meshes." In: *Computers and Graphics (Pergamon)*.

Zhenfang, Cao and He Bing (2012). "Research of Fast Cloth Simulation Based on Mass- Spring Model." In: *National Conference on Information Technology and Computer Science*, pp. 323–327.

Zink, Nico and Alexandre Hardy (2007). "Cloth Simulation and Collision Detection using Geometry Images." In: *AFRIGRAPH*. New York: ACM, pp. 187–195.