University of Southampton

FEEG3003 Individual Project

Analysis of Droplet Evaporation for Large Particle Populations Using GPUs and the OpenCL Language

by:
Andrew Kernan

Supervisor:
Prof. John Shrimpton

Word count: 9500

This report is submitted in partial fulfillment of the requirements for the MEng Aeronautics and Astronautics, Faculty of Engineering and Physical Sciences, University of Southampton

May 7, 2020

Declaration

I, Andrew Kernan declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a degree at this University;
2. Where any part of this thesis has previously been submitted for any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. None of this work has been published before submission.

Acknowledgements

I would like to thank my supervisor Professor John Shrimpton for his guidance and help during the project.

I would also like to thank Elijah Andrews for taking the time to explain and fix bugs in his code.

Contents

Abstract

The simulation of large populations of particles is computationally an expensive process. Simply running the code on CPUs does not make sense as the simulation times do not scale linearly with the number of particles. GPUs, however, have an architecture that makes them a superior compute device compared to CPUs for highly parallel operations.

This project will implement a commonly used model for heat and mass transfer, first in Python to develop and evaluate solving methods. Existing C and OpenCL code for particle motion will be modified to simplify it. With the final simulation code being added to this code once it has been verified and validated.

A number of simulations with large populations of particles will be run and PDFs for heat and mass transfer will be plotted. The effect of the Stokes number on convective heat transfer will be investigated. This provides a starting point for further statistical analysis of heat and mass transfer PDFs.

Nomenclature

Arabic Symbols

$\bar{R}$	Universial gas constant	$K\ kg\ mol$
$B_{m,eq}$	Spalding transfer number for mass	–
C	Molar concentration	mol/m^3
D	Diameter	mm
d	Diffusion coefficient	cm^2/s
f_1	Correction factor for Stokes drag	–
f_2	Correction factor for heat transfer due to evaporation	–
g_i	Acceleration due to gravity	$9.81\,m/s^2$
h	Enthalpy	J
h	Specific enthalpy	$J\ kg^{-1}$
H_M	Specific driving potential for mass transfer	–
$H_{\Delta T}$	Collection of terms contributing to non-uniform internal temperature effects	–
J	Diffusive flux density	$mol/m^2/s$
L_V	Latent heat of evaporation	J/kg
m	Mass	kg
P	Pressure	Pa
Q	Heat flux	W
q	Specific heat flux	W/kg
r	Droplet radius	m
T	Temperature	K
u_i	Carrier gas velocity	m/s
v_i	Droplet velocity	m/s
W	Molecular weight	$kg/kg\ mole$
X_i	Transient position	m
Y_G	Free stream vapour mass fraction	–
$Y_{s,eq}$	Vapour mass fraction at the droplet's surface	–

Greek Symbols

$\chi_{s,eq}$	Surface equilibrium mole fraction	–
Γ	Binary diffusion coefficient	m^2/s

λ	Thermal conductivity	$J\ m^{-1}s^{-1}K^{-1}$
μ	Viscosity	$kg/m/s$
ρ	Density	kg/m^3
τ_d	Particle time constant for Stokes flow	s
θ_1	Ratio of heat capacity for constant pressure between the gas and liquid phase	–
θ_2	Ratio of molecular weights	–

Subscripts

0	Value of property at start of simulation
B	Property boiling
C	Carrier gas property
d	Droplet property
eq	Property in equilibrium
G	Gas phase property
i	Vector component
n	Value of property at current time level
$n+1$	Value of property at next discrete time level
s	Property at droplet surface
sat	Property at saturation

Non-dimensional Numbers

Le	Lewis Number
Nu	Nusselt Number
Pr	Prandtl Number
Re	Reynolds Number
Sc	Schmidt Number
Sh	Sherwood Number

Acroynms

CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Unit
IP	Individual Project
MPI	Message Passing Interface
ODE	Ordinary Differential Equation
OpenCL	Open Computing Language

1 Introduction

This individual project builds off an individual project from two years ago titled "GPU Enabled Analysis of Agglomeration in Large Particle Populations". The goal of that project was to simulate up to 10^7 particles in a fluid including particle collisions, to observe how agglomerates form. As well as to vary the simulation properties and to do statistical analysis of the resulting agglomerate properties [1].

This individual project is titled "Analysis of Droplet Evaporation for Large Particle Populations Using GPUs and the OpenCL language". With the aim to simulate the evolution of heat and mass transfer for 10^4 particles using GPUs. The objectives required for this are:

1. Run the previous code and test it.
2. Compute coupled heat and mass transfer of a single droplet.
3. Run large scale simulations. This could lead into post-processing of the results. For example, generating probability density functions.

Objective 1 is imperative as the code produced from objective 2 will be added into the main time loop in the previous code. Objective 1 and 2 can run in parallel though; the aim is to first develop a suitable numerical model for simulating the heat and mass transfer of the particle in Python. This does not require the previous project's code, although the final code required for the large scale simulations will be required to work with the existing code. This approach reduces the risk of the project timelines slipping. Because if progress is slower than expected on one objective, more focus can be given to the other objective, allowing the project to continue to advance. Objective 3 requires both objective 1 and 2 to be completed, the post-processing of results etc... is a stretch goal for the project if time permits.

This report first introduces literature fundamental to the project and reviews the content of the literature. As well as explain the fundamental maths and physics that governs the simulation of the particles. The next section evaluates methods for implementing the models. A suitable method will be chosen and validated against experimental results in the literature.

The existing OpenCL code will then be simplified, verified and validated before the heat and mass model will be added it to it. This code will then also be verified and validated before investigating the effects the Stokes number has on convective heat transfer.

2 Literature Review

As previously mentioned this IP builds off a previous IP titled "GPU Enabled Analysis of Agglomeration in Large Particle Populations". The project developed simulation code in Python and OpenCL for simulating the collision and agglomeration of particles. This included a solution procedure for iterating the position and velocity of the particles [1]. This project uses the code produced by this previous project but removes the Discrete Element Method (DEM) model for the collisions as this is computationally expensive.

The key paper for this project with regards to simulating the heat and mass transfer processes is titled "Evaluation of equilibrium and non-equilibrium evaporation models for many-droplet gas-liquid flow simulations" by Miller, Harstad and Bellan. This paper is significant in that it details and compares eight different droplet evaporation models based on the same fundamental Lagrangian equations for droplet position, velocity, temperature and mass [2]. The eight models in the paper are denoted by Mx where x is the model number. As will be justified in Section 4 the model of choice for this project is M1 which is a form of the classical evaporation the earliest model known as the infinite conductivity model.

[2] is an unusually long paper although it presents some results that will be useful in validating the simulation code. Of note is Figure 2 with $Re = 0$ which means terms involving convection (particularly in the Nusselt and Sherwood numbers) can be neglected. Which provides a simpler test case. Moderate gas and droplet temperatures were also used again making the results simpler. There does appear to be a typo however in the caption, the starting droplet diameter D_0 was reported to be 1.1 mm contradicting the data in the plot with D^2 at $t = 0$ being 1.1 mm^2. For using this test case later in the report it is assumed $D^2 = 1.1\ mm^2$ with D_0 being $1.05mm$. This typo does not appear elsewhere in the paper, however. Although a key frustration in a paper that presents so many models is it can be hard to decipher how certain properties are evaluated. This issue is highlighted in a paper from the Center for Turbulence Research that evaluated how the definition of such properties affects the simulation sensitivity [3].

One of the most recent citations uses models M1 and M2 to simulate the evaporation of fuel particles in flame combustion. However, it used $\frac{\beta}{e^{\beta}-1}$ instead of 1 for one of the parameters f_2 [4]. Which as noted in [2] was not considered commonly used at the time that paper was published. The paper also investigates the effect of setting the Lewis number, Le to 1. It found that overall this assumption only increased the evaporation rate on the M2 model slightly. Although the effect was found to be more prominent for forced convection cases. In terms of the performance of the models the M1 model was found to more closely match experimental data for the forced convection of hexane. With $d_{p,0} = 1.76mm$, $T = 437K$ and $Re_0 = 110$. The same test for decane found that both models deviated from the experimental results but the M2 model was more accurate. For the decane case the conditions where $d_{p,0} = 2.0mm$, $T = 1000K$ and $Re_0 = 17$. The evaporation was for droplets in air. The experimental results are from [5], the same as those used in [2]. This is a good indication the experimental data used in [2] is still relevant.

The models only form half of the solution with regards to simulating populations of particles. Equally of importance is the method by which the equations are to be solved. It has already been determined from [1] the position and velocity equations are best solved numerically with an implicit method that gives higher than first order accuracy. A number of papers in the literature have used fourth order Runge-Kutta methods for solving the heat and mass equations [2] [6] [7]. Older papers such as "A Comparison of Vaporization Models in Spray Calculation" by Aggarwal et al. use a second order Runge-Kutta method

[8].

Examples of GPU simulation code in the literature seemed to be less common. Aforementioned papers have used CPU codes as the Lagrangian simulation of the droplet is normally coupled with a CFD code for the fluid. GPUs have been used to accelerate the particle Lagrangian equations, [9] provides an in-depth analysis of this. The paper highlights the Lagrangian code is accelerated by up to 14 times compared with the original CPU code. Where GPUs have been used CUDA seemed to be the language of choice [9] [10]. There have been studies using the Boltzmann Lattice Method that have used OpenCL [11] [12]. In terms of an implementation of the droplet evaporation model of the type presented here no implementations using OpenCL were found in the literature.

3 Background Material

3.1 GPU Computing

The question, why use GPUs? Naturally arises from the title of this project. CPUs in large computing clusters like the University of Southampton's Iridis Compute Cluster could be used. This would simplify the project as extra code called a kernel has to be written for the simulation to run on a GPU. The problem arises in the way CPUs share data in computing clusters.To write code for CPU parallel computing MPI has to be used. A block diagram of this is shown in Figure 3.1.

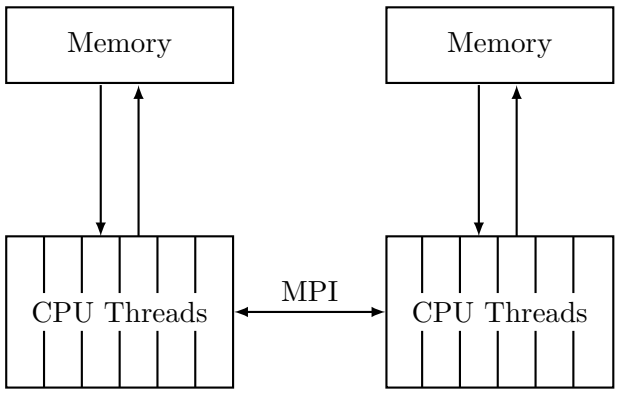


Figure 3.1: MPI block diagram.

This process of sharing data between compute nodes is what really limits CPU parallel computing. The overhead this adds would cause simulations not to scale linearly compared with using a GPU for a highly parallel task. As by comparison the GPU threads are able to access the same shared memory as shown in Figure 3.2. With regards to writing GPU kernels there are two languages to choose from. The Open Computing Language, OpenCL and the Compute Unified Device Architecture, CUDA. CUDA is a language developed by Nvidia that only runs on Nvidia GPUs, whereas OpenCL is open source and can run any GPU and indeed a wider range of compute devices. The existing code uses OpenCL kernels, to enable greater hardware compatibility, so OpenCL is the language that will be used for this project.

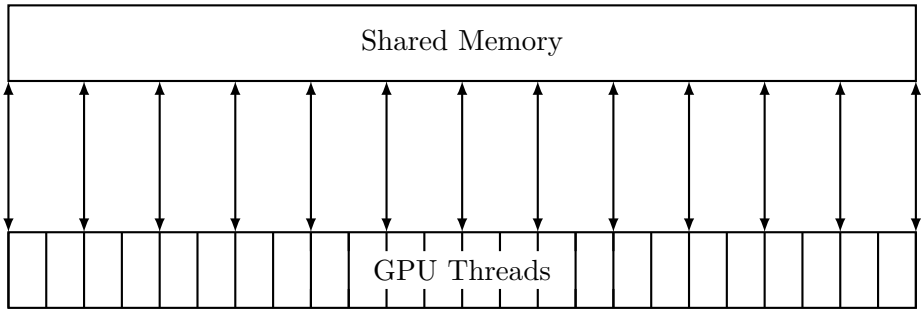


Figure 3.2: GPU memory block diagram.

3.2 Physics of Droplet Evaporation

3.2.1 Heat Transfer Processes

Fundamental to modelling droplets is an understanding of multiphase fluids. Droplet laden flows, (the subject of this report) are an example of two phase flows.

With regards to the physics of droplet evaporation there are two fundamental driving processes. These are diffusion and convection.

Diffusion is a process where there is a net movement of particles from a region of high concentration to a region of low concentration. This is driven by the concentration gradient so is affected by properties of the particles and the distance over which diffusion occurs. The carrier species of the particles will also affect the flux. Fick's Law quantifies this as:

$$J(x, y, z) = -d\left(\frac{\partial C}{\partial x}, \frac{\partial C}{\partial y}, \frac{\partial C}{\partial z}\right) \tag{3.1}$$

[13].

More useful is thermal diffusion, with the rate defined by Fourier's Law. For example, in one dimension as:

$$\dot{q}_x = -k\frac{dT}{dx} \tag{3.2}$$

With x in the direction of heat transfer, $\dot{q}_x$ the heat transfer rate per unit area, k the thermal conductivity and dT/dx the temperature gradient in the x-direction. [14]

Convection transfers heat energy in the direction of motion of the fluid. Convection can be forced (some fluid flow forcing convective transfer), or natural (the flow is created by the forces resulting from the heat transfer). Convective heat transfer for a fluid is given by:

$$\dot{q} = h\Delta T \tag{3.3}$$

where ΔT is the difference between the temperature at the surface of the fluid and the temperature at infinity: $T_s - T_\infty$. The convective heat transfer coefficient h contains variables that allow for a simple form for the convective heat transfer equation.[15]

In general an exact solution for h is only available for specific cases such as laminar flow. More common is to form an empirical relation using non-dimensional numbers that capture the physics of the problem. [16]

The general energy conservation for the droplet can be found by considering that the change in energy of the droplet is equal to the rate of energy into the droplet subtract the rate of energy leaving the droplet. Therefore, the heat flux into the droplet Q_L is the convective heat flux subtract the heat flux from vaporisation [17].

$$Q_L = Q_C - Q_V \tag{3.4}$$

Using definitions for Q_C and Q_V:

$$Q_L = \pi\lambda_G DNu(T_G - T_d) - L_V\dot{m} \tag{3.5}$$

3.2.2 Vapour Pressure

The driving force for changes to the properties in the vapour phase is the vapour pressure. Assuming the droplet is formed of a pure liquid species the vapour pressure is only a

function of temperature. For evaporation to occur the partial pressure of the liquid must be less than the vapour pressure [13].

The vapour pressure can be found by considering that for two phases to be in equilibrium their respective chemical potentials, μ_{pot} must be equal. This implies here that $\mu_{liquid} = \mu_{gas}$ and any changes must be equal $\rightarrow d\mu_{liquid} = d\mu_{gas}$. If there is a change of pressure acting on the liquid phase, i.e. $dV_{m,liquid}dP = d\mu_{liquid}$. (With $dV_{m,liquid}$ the molar volume of the liquid phase). Therefore, $dV_{m,gas}dP_{vapour} = d\mu_{gas}$, with dP_{vapour} the change in vapour pressure. Equating the changes in the liquid and gas phase, it is possible using integration and assuming the gas is perfect and the molar volume of gas is much greater than the molar volume of the liquid to derive the Clausius-Clapeyron equation [18].

$$\chi_{s,eq} = \frac{P_{atm}}{P_G} \exp\left[\frac{L_V}{\bar{R}/W_V}\left(\frac{1}{T_B} - \frac{1}{T_d}\right)\right] \tag{3.6}$$

So the equilibrium mole fraction of the vapour can be found.

3.2.3 Mass Transfer

The droplet's mass can be transferred by diffusion and convection. For a stationary droplet only diffusive transfer will occur and the Reynolds number is zero. As the liquid phase heats up a phase change process occurs at the surface of the droplet and a vapour phase forms around the droplet. This vapour phase is partially responsible for how quickly the droplet evaporates.

From the surface equilibrium mole fraction the mass fraction of the vapour $Y_{s,eq}$ at the surface of the droplet can be found by considering the equilibrium condition for the droplet surface. An equilibrium condition for the fuel vapour at the droplet surface can be found as:

$$\dot{m}_s = \pi D^2[\rho_{G,s} u Y_s] - \pi D^2 \left[\Gamma \rho_{G,s} \frac{\partial Y}{\partial r}\right]_{r=D/2} \tag{3.7}$$

with $\dot{m}_s = \rho_{G,s}\pi D^2 u$.

Mass conservation equations can also be found:

$$\frac{\partial(\rho_G r^2 u)}{\partial r} = 0 \tag{3.8}$$

With the conservation of mass at the droplet surface is given in the following equation:

$$\frac{\partial}{\partial r}(\rho_G r^2 u Y) - \frac{\partial}{\partial r}(\rho_G r^2 \Gamma \frac{\partial Y}{\partial r}) = 0 \tag{3.9}$$

The surface vapour fraction can be found from the ideal gas law and mixture definition [19].

$$Y_{s,eq} = \frac{\chi_{s,eq} W_V}{\chi_{s,eq} W_V + \chi_{g,eq} W_C} \tag{3.10a}$$

$$Y_{s,eq} = \frac{\chi_{s,eq}}{\chi_{s,eq} + (1 - \chi_{s,eq})\frac{W_C}{W_V}} \tag{3.10b}$$

$$Y_{s,eq} = \frac{\chi_{s,eq}}{\chi_{s,eq} + (1 - \chi_{s,eq})\theta_2} \tag{3.10c}$$

The ratio of the difference between the surface vapour and free stream vapour mass fractions to unity subtract the surface vapour mass fraction gives the Spalding transfer number for mass.

$$B_{M,eq} = \frac{Y_{s,eq} - Y_G}{1 - Y_{s,eq}} \tag{3.11}$$

The free stream vapour mass fraction is evaluated far from the droplet surface so this zero. Therefore the greater the surface vapour mass fraction the greater the mass transfer number.

The equations outlined here will be used in Section 4.3.5 to derive an equation for mass transfer.

3.2.4 Droplet Physical Model

The droplet consists of a a liquid phase surrounded by a vapour phase, with the droplet surrounded by a gas phase which is the carrier gas species. A diagram of the physical model of the droplet is shown in Figure 3.3.

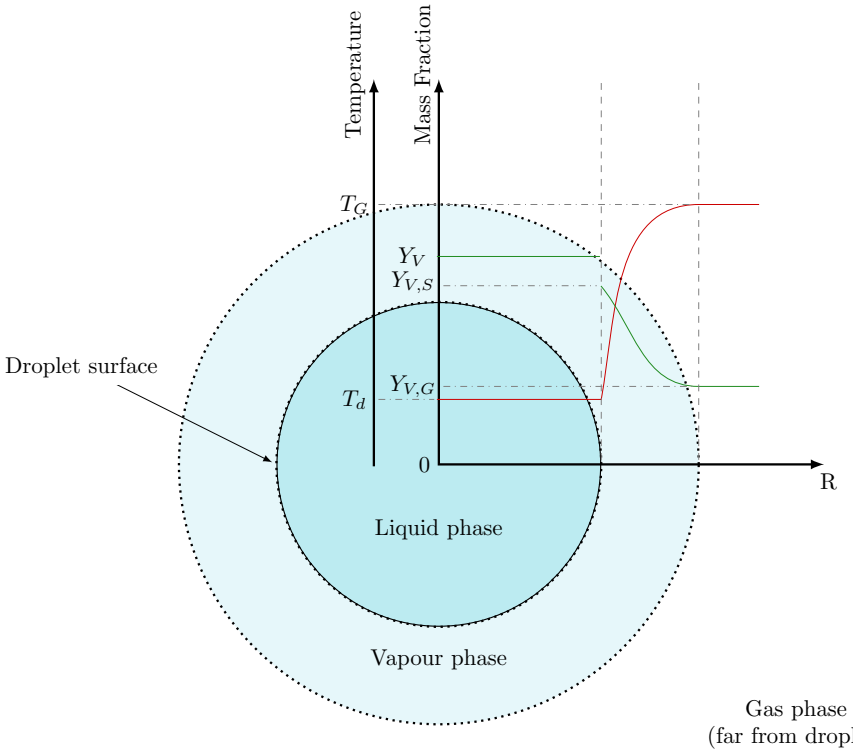


Figure 3.3: Diagram of how the droplet's and carrier gas phases with their associated properties vary with droplet radius.

The droplet temperature is assumed to be uniform within the droplet, although varies with time. In reality there is a thermal gradient across the droplet. The internal temperature profile can be modelled as part of M8 in [2]. The temperature rises across the vapour phase to the gas temperature.

Similarly within the droplet the mass fraction of droplet vapour within the droplet is uniform and is equal to 1. The mass fraction drops instantaneously at the surface, as the surface of the droplet is a mixture of droplet vapour and gas phase. The mass fraction of vapour decreases across the vapour phase until an equilibrium mixture with the gas phase phase is reached.[20].

4 Droplet Evaporation Model

4.1 Definition of Models

The 8 models in [2] are denoted by Mx where x is the model number.

What defines the models are the variables f_1, f_2, Nu, Sh $H_{\Delta T}$ and H_M. Although the main variation in the models stems from f_2, $H_{\Delta T}$ and H_M. M1 is known as the classical rapid mixing model. M2 will be ruled out for the purposes of this project as the B'_T term in the evaporation correction f_2 must be solved iteratively at each timestep. This increases the computation overhead of the simulation which may be acceptable for a single droplet but will be problematic for large population of droplets.

Model	Name	f_2	$H_{\Delta T}$	H_M
M1	Classical rapid mixing	1	0	$\ln\left[1 + B_{M,eq}\right]$
M2	Abrmazon-Sirignano	$\frac{-\dot{m}_d}{m_d B'_T}\left[\frac{3Pr_G\tau_d}{Nu}\right]$	0	$\ln\left[1 + B_{M,eq}\right]$
M3	Mass analogy Ia	1	0	$B_{m,eq}$
M7	Langmuir-Knudsen I	$\frac{\beta}{e^{\beta}-1}$	$\frac{2\beta}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)\Delta_s$	$\ln\left[1 + B_{M,neq}\right]$

Table 4.1: Detail of variables used in candidate models.

A more suitable model to provide a point of comparison in this case would be one of the mass analogy models, M3-M6. Which are derived from vapour mass fraction boundary condition at the droplet's surface. This assumes the droplet does not dissolve in the gas phase. These models use the same formulations for the surface vapour mass fraction (Y_s), Spalding transfer number for mass (B_m) and mass transfer potential ($H_{\Delta T}$). This would simplify producing code for an extra model.

Models M7 and M8, are to be ruled out as they use non-equilibrium formulations of the coefficients which would further complicate the code. The comparison between models is left as future work. With the choice of model for this project will be M1 due to its wide usage and ease of implementation.

4.2 Assumptions

The model presented here is subject to the following assumptions:

1. The droplets are formed of a single species
2. The droplet is spherical and remains spherical for the entire evaporation process
3. There is no interaction between droplets
4. Within the droplet there is
   (a) Zero temperature gradient
   (b) Zero concentration gradient
5. The droplet temperature is homogeneous

6. The properties of the carrier gas are constant and are unaffected by the process of droplets evaporating

[2][21].

4.3 Governing Equations

Each of the models are defined by the same four Lagrangian equations for the droplets:

$$\frac{dX_i}{dt} = v_i \tag{4.1}$$

$$\frac{dv_i}{dt} = \left(\frac{f_1}{\tau_d}\right)(u_i - v_i) + g_i \tag{4.2}$$

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) + \left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d} - H_{\Delta T} \tag{4.3}$$

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\left(\frac{m_d}{\tau_d}\right)H_M \tag{4.4}$$

4.3.1 Particle Timescales

τ_d is the momentum relaxation timescale for the particle. Which is the time it takes for the droplet to adjust to changes in the gas flow [22]. This provides a timescale for each particle. It is determined from:

$$\tau_d = \frac{\rho_d D^2}{18\mu_G} \tag{4.5}$$

There is an equivalent timescale for the heat transfer process. This can be derived from equation 7.2 as:

$$\tau_h = \tau_d\left(\frac{3Pr_G}{Nuf_2\theta_1}\right) \tag{4.6}$$

These two timescales are exceptionally useful for the non-dimensionalisation of results. This provides a way of evaluating how quickly processes occur in the timescale of the particle. Which is a far more useful insight into the physics taking place than particle x takes y seconds to evaporate.

4.3.2 Position

Equation 4.1 and a simpler form of equation 4.2 were used in the IP this project is based off. The differential equation for position was solved using the trapezoidal rule, as this method has second order accuracy and the result is a linear first order equation [1].

$$X_{n+1} = X_n + \frac{v_n + v_{n+1}}{2}\Delta t \tag{4.7}$$

4.3.3 Velocity

For equation 4.2, the acceleration of the droplet is defined using Stokes' Law and can be derived by considering the forces acting on the particle:

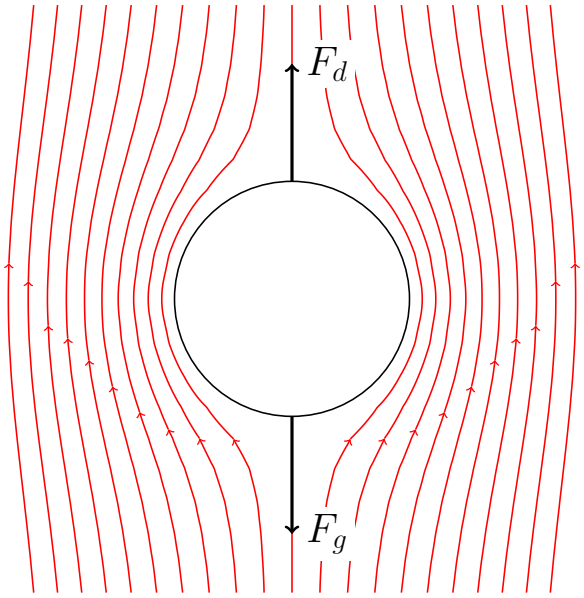


Figure 4.1: Forces acting on a particle under a Stokes flow regime.

The total force acting on the particle is:

$$F = F_g + F_d \tag{4.8}$$

This can be solved as a differential equation:

$$F = ma \tag{4.9a}$$
$$F = m\frac{du}{dt} \tag{4.9b}$$
$$\frac{du}{dt} = \frac{F(u)}{m} \tag{4.9c}$$

The force acting on the particle is the sum of the weight and drag forces. The weight force is given by:

$$F_g = mg \tag{4.10}$$

It is assumed the particle experiences a Stokes drag regime, therefore:

$$F_d = \frac{m}{\tau_d}(u_f - u) \tag{4.11}$$

So, the differential equation is:

$$\frac{du}{dt} = \frac{F(u)}{m} \tag{4.12a}$$
$$\frac{du}{dt} = \frac{F_g + F_d}{m} \tag{4.12b}$$
$$\frac{du}{dt} = \frac{mg + \frac{m}{\tau_d}(u - v)}{m} \tag{4.12c}$$
$$\frac{du}{dt} = \frac{1}{\tau_d}(u - v) + g \tag{4.12d}$$

This is for one vector component, hence the general case is:

$$\frac{dv_i}{dt} = \frac{1}{\tau_d}(u_i - v_i) + g_i \tag{4.13}$$

This change in velocity is dependent on three terms. Acceleration due to gravity g_i, and the difference between the carrier gas velocity and the velocity of the particle, $u_i - v_i$. The third term is a correction for Stokes drag that is dependent on the time constant for Stokes flow.

A formulation for f_1 is given in [2] as:

$$f_1 = \frac{1 + 0.0545Re_d + 0.1Re_d^{0.5}(1 - 0.03Re_d)}{1 + a|Re_b|^b} \tag{4.14}$$

With the Reynolds numbers defined as:

$$Re_d = \frac{\rho_G u_s D}{\mu_G} \tag{4.15a}$$

$$Re_b = \frac{\rho_G u_b D}{\mu_G} \tag{4.15b}$$

The velocities defined as:

$$u_s = |u_i - v_i| \tag{4.15c}$$

$$u_b = -\frac{\dot{m}}{\pi \rho_G D^2 u_b} \tag{4.15d}$$

And the constants a and b defined as:

$$a = 0.09 + 0.077\exp(-0.4Re_d) \tag{4.15e}$$

$$b = 0.4 + 0.77\exp(-0.04Re_d) \tag{4.15f}$$

Which is an empirical result produced from a data fit. In this format it is valid for $0 \leq Re_d \leq 100$ and $0 \leq Re_b \leq 10$. For the purpose of this research f_1 will be set to 1. This reduces the computation per timestep and still allows for comparisons to be made with data in the literature. As the models can be driven by iterating from an initial Reynolds number instead of using equation 4.2.

4.3.4 Temperature

The temperature ODE can be derived using the general energy conservation equation and application of non-dimensional numbers.

From the energy conservation equation and using the using the definition for the change in enthalpy:

$$\dot{Q} - \dot{E} = \frac{dH_d}{dt} \tag{4.16a}$$

$$Q_L = \frac{dH_d}{dt} \tag{4.16b}$$

Using the definition for specific enthalpy $h_d(dm_d/dt) + m_d(dh_d/dt)$ and the approximation $dh_d = C_{p,d}dT_d$ leads to the temperature ODE [23].

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) + \left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d} - H_{\Delta T} \tag{4.17}$$

The term $H_{\Delta T}$ is a correction factor to include effects from a non-uniform internal temperature [2].

4.3.5 Mass

Using the conservation equations from Section 3.2.3 the mass transfer ODE can be derived.

First, the continuity equation can be simplified as the model assumes quasi-steady conditions:

$$\frac{\partial(\rho u r^2)}{\partial r} = \rho u r^2 = constant \tag{4.18a}$$

$$\rho u r^2 = \frac{\dot{m}}{4\pi} \tag{4.18b}$$

The equation from the vapour phase equilibrium can also be simplified using the quasi-steady assumption:

$$\frac{\partial}{\partial r}(\rho_G r^2 u Y) - \frac{\partial}{\partial r}(\rho_G r^2 \Gamma_G \frac{\partial Y}{\partial r}) = 0 \tag{4.19a}$$

$$(\rho_G r^2 u Y) - (\rho_G r^2 \Gamma_G \frac{dY}{dr}) = 0 \tag{4.19b}$$

Combining this with the continuity equation:

$$(\rho_G r^2 u Y) - (\rho_G r^2 \Gamma_G \frac{dY}{dr}) = \frac{\dot{m}}{4\pi} \tag{4.20a}$$

$$(4\rho_G r^2 \Gamma_G \frac{dY}{dr}) = \dot{m}(Y - 1) \tag{4.20b}$$

Integration and application of the droplet surface boundary condition obtains an equation for vaporisation rate:

$$\dot{m} = 4\pi \rho_G \Gamma_G r_d \ln(1 + B_M) \tag{4.21}$$

Application of relations for the Sherwood and Schmidt numbers gives the final mass ODE [24].

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\frac{m_d}{\tau_d} H_M \tag{4.22}$$

4.3.6 Reference Properties

The well known "1/3 rule" is often used for evaluating a reference temperature T_R and a reference mass fraction Y_R [2] [3] [4] [7] [8]. The reference mass fraction is then used in mixture calculations, for example the Wilke rule to evaluate the diffusive properties. [2] shows these mixture calculations are not necessary for common experimental test cases in the literature. So, mixture calculations will not be used here.

The reference temperature is used for determining gas and vapour properties. For this instead of the "1/3 rule" an estimated value for the wet bulb temperature T_{WB} of the droplet is used. If the bulb of a thermometer is covered with a damp cloth the temperature that is read from it is the wet bulb temperature. This is lower than the dry bulb temperature as heat energy transferred from the gas is used to evaporate moisture in the cloth. In the context of droplet evaporation the wet bulb temperature can be considered the temperature at the surface of the droplet [25]. And this value can be used for the gas and vapour phase property evaluation. [2] provides a correlation fit for T_{WB} as:

$$T_{WB} = 137\left(\frac{T_B}{373.115}\right)^{0.68}\log_{10}(T_G) - 45 \tag{4.23}$$

This will be used as the reference temperature.

4.4 Model Evaluation

4.4.1 Property Evaluation

Following the method used in [2] the gas and vapour properties will be evaluated once at the start of the simulation using the reference conditions.

The process for getting the properties is as follows:

1. Set droplet physical properties:
   (a) W_V
   (b) T_B
   (c) ρ_d
   (d) C_L
2. Set fluid physical properties:
   (a) W_G
   (b) θ_2 from W_V and W_G
   (c) P_{atm}
   (d) R_{bar}
   (e) R
   (f) Y_G
3. Get reference conditions:
   (a) T_{WB} using T_B and T_G. T_{WB} is used as T_R
4. Set additional droplet physical properties:
   (a) L_V using T_R
5. Set additional fluid physical properties:
   (a) P_G using the ideal gas law with T_R and ρ_G
   (b) μ_G using T_R
   (c) Pr_G using T_R
   (d) Sc_G equal to Pr_G (assuming unity Lewis (Le) number)
6. Set additional properties:
   (a) $H_{\Delta T}$
   (b) f_2
   (c) θ_1 using $C_{p,G}$ and C_L

The droplet and fluid physical properties used can be found in Section x. ρ_G and $C_{p,G}$ are also obtained from reference data that can be found in Section C. Running the model also requires inputting the following:

1. D_0
2. T_{d0}

3. T_G
4. u_i or Re_{d0}

In the case of using Re_{d0} this value is used as Re_d which remains fixed during the simulation.

It should be noted M1 normally uses the the "1/3 rule" at each timestep. The usage of the model is in contrast to the literature but it is shown in Section 7.5 the results are still within reason.

4.4.2 Evaluation of Properties During a Timestep

For a single timestep the procedure is:

1. Calculate the mass fractions:
   (a) $\chi_{s,eq}$
   (b) $Y_{s,eq}$
   (c) $B_{M,eq}$
   (d) H_M
2. Calculate non-dimensional numbers:
   (a) Re_d
   (b) Sh
   (c) Nu
3. Get the rate of change of mass $\dot{m}_d$
4. Get the change in temperature dT_d/dt
5. Update the mass m_d
6. Update the temperature T_d

The process of getting the mass fractions can be further broken down. H_M is calculated in the case of M1 using:

$$H_M = \ln\left[1 + B_{M,eq}\right] \tag{4.24}$$

Where $B_{M,eq}$ is the Spalding transfer number for mass transfer given by:

$$B_{M,eq} = \frac{Y_{s,eq} - Y_G}{1 - Y_{s,eq}} \tag{4.25}$$

With Y_G the free stream vapour mass fraction and $Y_{s,eq}$ the vapour mass fraction at the droplet's surface:

$$Y_{s,eq} = \frac{\chi_{s,eq}}{\chi_{s,eq} + (1 - \chi_{s,eq})\theta_2} \tag{4.26}$$

$\chi_{s,eq}$ is the surface equilibrium mole fraction of the vapour and θ_2 is the ratio of molecular weights:

$$\theta_2 = \frac{W_C}{W_V} \tag{4.27}$$

The Clausius-Clapeyron equation for constant latent heat gives a relation between the saturation pressure P_{sat} and the free stream pressure P_G as:

$$\chi_{s,eq} = \frac{P_{sat}}{P_G} \tag{4.28a}$$

$$\chi_{s,eq} = \frac{P_{atm}}{P_G} \exp\left[\frac{L_V}{\bar{R}/W_V}\left(\frac{1}{T_B} - \frac{1}{T_d}\right)\right] \tag{4.28b}$$

5 Fluid Model

As noted in the Literature Review the Lagrangian droplet model is normally coupled with a CFD code. Quite complex models could be used for this, for example, for investigations into the effect of turbulence. However, the focus here is getting a droplet evaporation model to be solved on a GPU and to investigate some statistics. Complex CFD codes could be added at a later date.

5.1 Taylor-Green Vortex

A set of analytic expressions for the fluid velocity will be used. The previous IP used a Taylor-Green vortex and that is what will also be used here. It is defined as:

$$U = A\cos\left(a\left(x+\frac{\pi}{2a}\right)\right)\sin\left(a\left(y+\frac{\pi}{2a}\right)\right)\cos\left(a\left(z+\frac{\pi}{2a}\right)\right) \tag{5.1a}$$

$$V = A\sin\left(a\left(x+\frac{\pi}{2a}\right)\right)\cos\left(a\left(y+\frac{\pi}{2a}\right)\right)\sin\left(a\left(z+\frac{\pi}{2a}\right)\right) \tag{5.1b}$$

$$W = -2A\sin\left(a\left(x+\frac{\pi}{2a}\right)\right)\sin\left(a\left(y+\frac{\pi}{2a}\right)\right)\cos\left(a\left(z+\frac{\pi}{2a}\right)\right) \tag{5.1c}$$

With A defined as the flow magnitude and a defined as the vortex frequency. This flow field is visualised in Figure 5.1.

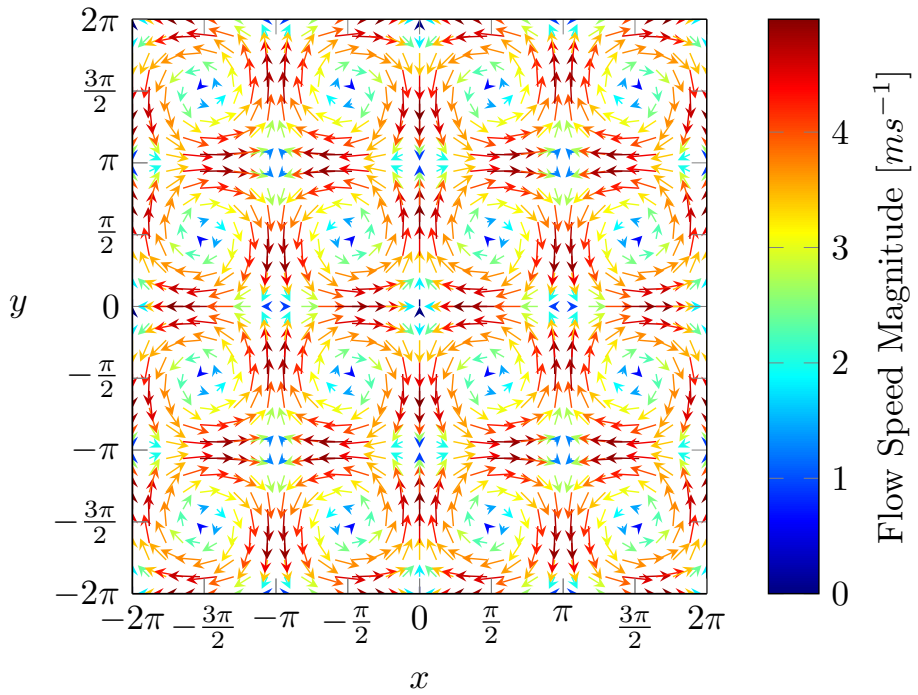


Figure 5.1: A 2D xy vector field of a Taylor-Green vortex. Using $A = 5$, $a = 1$ and a domain length of 4π.

The length of one of the individual vortices is the characteristic length l_0 used in defining the Stokes number.

$$Stk = \frac{\tau_{d0}u_0}{l_0} \tag{5.2}$$

Where:

$$\tau_{d0} = \frac{\rho_d D^2}{18\mu_G} \tag{5.3a}$$

$$u_0 = 0.7839A \tag{5.3b}$$

$$l_0 = \frac{\pi}{a} \tag{5.3c}$$

In this case $l_0 = \pi$ and $u_0 = 3.9195$ [1]. The usage of the Stokes number can be found in Section 8.2.

6 Numerical Methods

Given the governing equations are coupled and non-linear finding an analytical solution has not proved possible. It is therefore necessary to use a numerical scheme to solve the equations. The choice of scheme and its implementation are especially important when considering the solution must work for up to millions of droplets. The scheme should of course be accurate but its complexity is also a consideration. There is no point choosing a method that is very accurate but is slow because the because this will be compounded greatly across millions of particles. In addition to this the method must be stable and resistant to non-physical phenomena. (I.e. the droplet mass cannot go below zero).

Analytical solutions to the equations decoupled are available and provide a point of comparison for some of the schemes.

6.1 Timestepping Procedure

Numerical methods work by taking discrete steps forward in time. The length of the simulation is determined by the size of the timestep and the maximum number of timesteps. The timestepping procedure is shown in Figure 6.1.

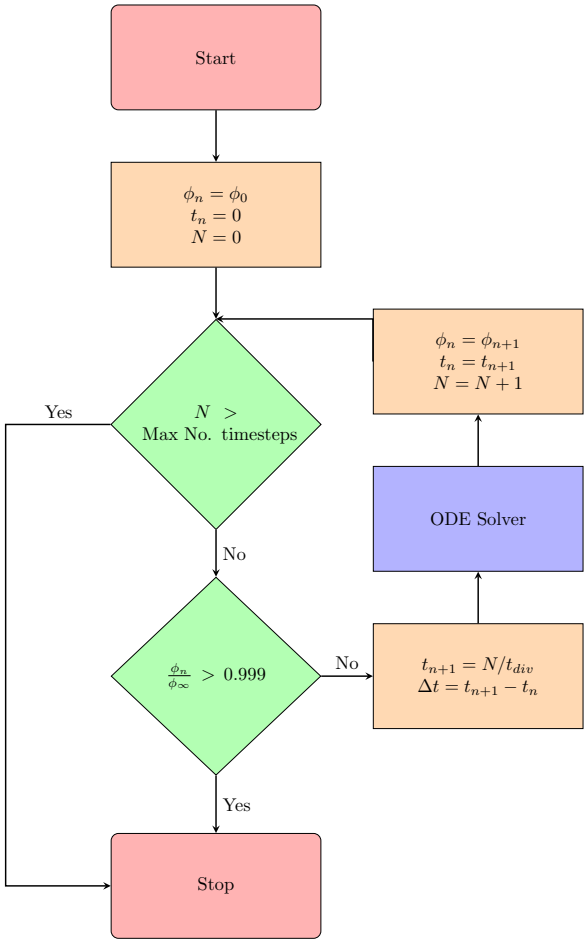


Figure 6.1: General timestepping procedure.

There are two conditions for the loop to run. The maximum number of time steps must not have been exceeded and the physical property ϕ cannot have converged to its value at infinity. This works for droplet temperature; whilst for mass the condition would be a

ratio of the current to initial mass being greater than 0.001. Limiting the maximum number of time steps is necessary to prevent an infinite recursion in the case of the simulation being unable to converge. Also by specifying the value t_{div} in addition to N the size of time step and total time to be simulated is set.

The ODE solver is a black box that returns a numerical solution to the ODE at each discrete timestep. Examples that will be tested are given in the following subsections.

6.2 Euler Method

The simplest numerical method is to step forward in time and take the value at the next time level to be the current value plus a prediction on what the change between the values will be. This can be derived from the Taylor Series:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + .. + \frac{h^{n-1}}{(n-1)!}f^{(n-1)}(x) + \frac{h^n}{n!}f^n(c) \tag{6.1}$$

Using Δt as h and a as the value at the current time level n, from the first two terms in the series:

$$f(n+1) = f(n) + \Delta t f'(n) \tag{6.2}$$

This is the forward Euler method and can be used to solve ODEs of the form:

$$\frac{d\phi}{dt} = f(t,\ \phi) \tag{6.3}$$

in increments of Δt, by assuming for a small time step the output of the differential is constant. The formula is:

$$\phi_{n+1} = \phi_n + \Delta t f(t_n,\ \phi_n) \tag{6.4}$$

This formula is used in Figure 6.1 as the ODE solver:

$$\phi_{n+1} = \phi_n + \Delta t f(t_n, \phi_n)$$

Figure 6.2: Forward Euler method solver.

The forward Euler method is easy to start with only the initial conditions being required.

The order of the error can be found by again considering the Taylor series:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) \tag{6.5a}$$

$$f(x+h) = f(x) + hf'(x) + O(h^2) \tag{6.5b}$$

The $O(h^2)$ term is the local truncation error, the error inherited at each time step. Therefore, the local error is proportional to h^2. The order of the method is however of first order. I.e the error overall is proportional to h since the number of steps is proportional to $1/h$.

6.3 Modified Euler Method

The problem with the forward Euler method is it makes one prediction about what the solution will be at the next time step and assumes this solution is correct. This method can be modified so the current and predicted value are used to estimate the gradient over the interval. This gives a more accurate numerical estimation of the gradient so the solution at the next time step should also be more accurate.

Based on this the general formula is:

$$\phi_{n+1} = \phi_n + \frac{\Delta t}{2}\left[f(t_n, \phi_n) + f(t_{n+1}, \tilde{\phi}_{n+1})\right] \tag{6.6}$$

The term $f(t_{n+1}, \phi_{n+1})$ is the prediction made using the forward Euler method. A flowchart representation of the method is shown in Figure 6.3.

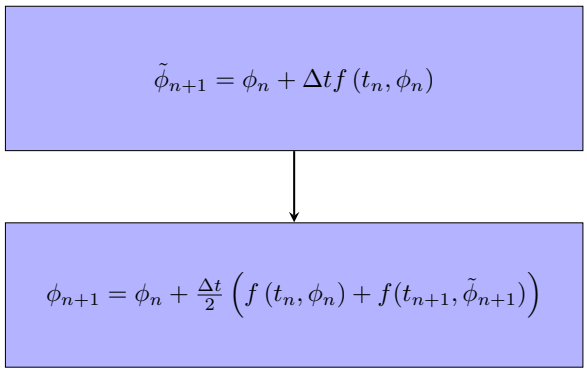


Figure 6.3: Modified Euler method solver.

6.4 Runge-Kutta Method

The modified Euler method (also known as the Heun formula) is a predictor corrector method. In fact it is an example of a more general family of predictor corrector methods known as the Runge-Kutta method. The modified Euler method is in fact a second order Runge-Kutta method:

$$k_1 = \Delta t f(t_n, \phi_n) \tag{6.7a}$$
$$k_2 = \Delta t f\left(t_n + \Delta t, \phi_n + k_1\right) \tag{6.7b}$$
$$\phi_{n+1} = y_n + \tfrac{1}{2}(k_1 + k_2) \tag{6.7c}$$

The predictor corrector process can be extended such that 4 estimates of the solution are made, which are then averaged in a single formula. This gives a fourth order accurate scheme.

$$k_1 = \Delta t f(t_n, \phi_n) \tag{6.8a}$$
$$k_2 = \Delta t f\left(t_n + \tfrac{\Delta t}{2}, \phi_n + \tfrac{k_1}{2}\right) \tag{6.8b}$$
$$k_3 = \Delta t f\left(t_n + \tfrac{\Delta t}{2}, \phi_n + \tfrac{k_2}{2}\right) \tag{6.8c}$$
$$k_4 = \Delta t f\left(t_n + \Delta t, \phi_n + k_3\right) \tag{6.8d}$$
$$\phi_{n+1} = \phi_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{6.8e}$$

More weighting is applied to the terms k_2 and k_3 as these are estimates of the midpoint of the interval. The procedure is shown in Figure 6.4.

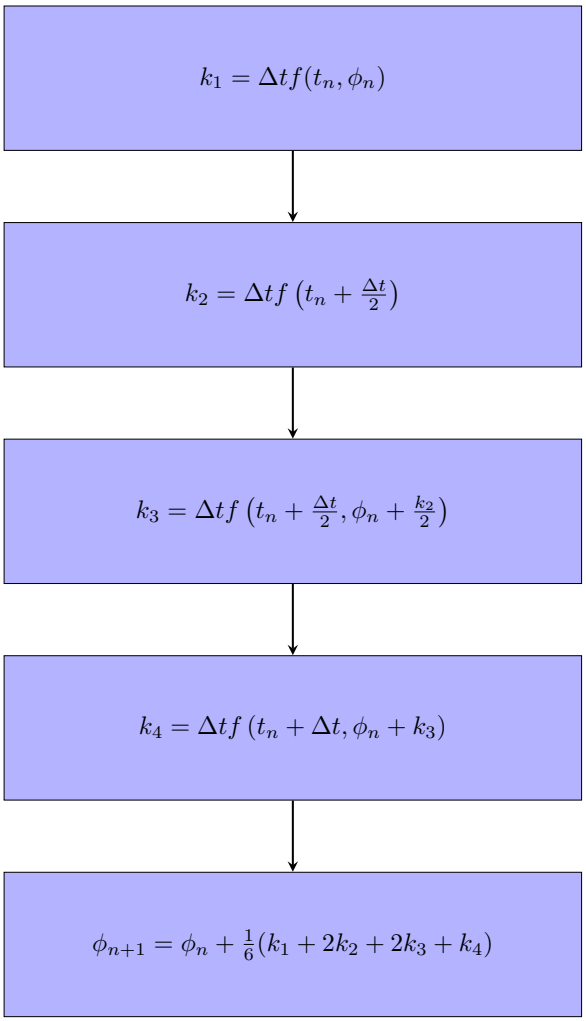


Figure 6.4: Flowchart for the Runge-Kutta method.

6.5 Backward Euler Method

The previously mentioned methods are all examples of explicit methods. So called as a function for the next value in time can be expressed explicitly in terms of known variables. The disadvantage of such methods is they extrapolate into the future; for small time steps this is fine as the local truncation error is small. But for larger time steps, (relative to the timescale of the problem) this can cause instability and the solution diverges.

An implicit method uses a function with y_{n+1} on both the left and right hand side of the equation. In general this means an iterative method must be used to solve for y_{n+1}. However, for simpler equations an explicit function can be found. The benefit of an implicit method is it is in theory more stable than an explicit method.

A first order accurate implicit method is the backward Euler method:

$$t_{n+1} = t_n + \Delta t \tag{6.9a}$$

$$y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1}) \tag{6.9b}$$

A non-iterative formula can be derived for the temperature and mass ODES. Unlike the explicit methods, the implicit method will have to be derived for each ODE. For the temperature ODE:

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) + \left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d} - H_{\Delta T} \tag{6.10}$$

This leads to:

$$t_{n+1} = \Delta t \tag{6.11a}$$

$$T_{d_{n+1}} = T_{d_n} + \Delta t \left[\frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) (T_G - T_{d_{n+1}}) + \left(\frac{L_V}{C_L} \right) \frac{\dot{m}_d}{m_d} - H_{\Delta T} \right] \tag{6.11b}$$

For the second equation rearrangement is required, the full process is shown in Section B.2 which leads to:

$$T_{d_{n+1}} = \frac{T_{d_n} + \Delta t \left[T_G \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) + \left(\frac{L_V}{C_L} \right) \frac{\dot{m}_d}{m_d} - H_{\Delta T} \right]}{\left(1 + \Delta t \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) \right)} \tag{6.12}$$

A similar process can be applied to find an equivalent expression for the mass ODE:

$$\frac{dm_d}{dt} = -\frac{Sh}{3 Sc_G} \left(\frac{m_d}{\tau_d} \right) H_M \tag{6.13}$$

With:

$$t_{n+1} = \Delta t \tag{6.14a}$$

$$m_{d_{n+1}} = m_{d_n} + \Delta t \left[-\frac{Sh}{3 Sc_G} \left(\frac{m_{d_{n+1}}}{\tau_d} \right) H_M \right] \tag{6.14b}$$

Rearranging the second equation leads to:

$$m_{d_{n+1}} = \frac{m_{d_n}}{1 + \Delta t \frac{Sh}{3 Sc_G} \left(\frac{H_M}{\tau_d} \right)} \tag{6.15}$$

(The full solution can be found in Section B.3).

The numerical methods outlined in Section 6 will be evaluated in the next section to determine their suitability.

7 Single Particle Solution Method

The ODEs for mass and heat transfer are coupled and non-linear making them challenging to solve analytically. To get the simulation running initially the ODEs for heat and mass transfer were solved separately. Different numerical methods were used to solve the ODEs to evaluate their suitability.

For the purpose of this testing the model was implemented in Python. The model is contained within a single class that contains methods for evaluating the model. This is iterated through a timestepping procedure to solve the model. The code can be found at [26].

7.1 Simulation Settings

For the uncoupled heat and mass transfer the following settings were used.

The Reynolds number was fixed at zero, which means the drop is stationary and there is only diffusive heat transfer. This fixes the value of the Nusselt and Sherwood numbers as 2. As M1 is used $f_2 = 1$ and $H_{\Delta T} = 0$.

In addition to this to decouple the temperature ODE from the mass ODE, the term $\left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d}$ was ignored. The mass transfer ODE requires no such alterations to decouple it.

Some parameters must be calculated based on empirical relations. For the droplet this includes L_V and for the gas this includes μ_G, Pr_G and Sc_G. P_G is determined using the ideal gas law. See Section C for the relations.

The full settings are listed in Table 7.1. The conditions used are from [27].

Parameter	Setting
Droplet Physical Properties	
Molecular Weight of Vapour Phase W_V	18.015 $kg/kg\ mol$
Boiling Temperature T_B	373.15 K
Density of Liquid Phase ρ_L	997 kg/m^3
Specific Heat Capacity of Liquid Phase C_L	4184 $J/(kg\ K)$
Initial Temperature T_{d0}	282 K
Initial Diameter D_0	$\sqrt{1.1}\ mm$
Initial Reynolds Number Re_0	0
Gas Properties	
Molecular Weight of Phase W_C	28.97 $kg/(kg\ mol)$
Temperature T_G	298 K
Density ρ_G	1.184 $kg\ m^{-3}$
Specific Heat for Constant Pressure $C_{P,G}$	1007 $J\ kg\ K$
Free Stream Vapour Mass Fraction Y_G	0
Atmospheric Properties	
Pressure P_{atm}	101325 Pa
Physical Constants	
Gas Constant $\bar{R}$	8314.5 $J/(K(kg\ mol))$
Universal Gas Constant R	287 $J/(kg\ K)$
Simulation Properties	
Correction factor f_2	1
Driving Potential $H_{\Delta T}$	0

Table 7.1: Simulation settings used for evaluating the numerical methods.

The timestep size used is included in the results.

7.2 Uncoupled Heat Transfer

7.2.1 Analytic Solution

Using the settings specified in Section 7.1 the temperature ODE becomes:

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) \tag{7.1}$$

Which can be solved analytically to provide a reference point for the numerical solution. See Section B.4 for the full derivation which produces the solution:

$$T_{d_{n+1}} = T_G - (T_G - T_{d_n})e^{-\left(\frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)\right)\Delta t} \tag{7.2}$$

7.2.2 Results

The implementation of equation 7.1 requires the usage of a for loop to iterate through the time steps. Importantly, this loop must check to see if T_{d_n} has exceeded T_G, if this condition is met the loop must stop as the following results will be unphysical.

For plotting data the results have been non-dimensionalised. The points in time are non-dimensionalised with the particle heat transfer time constant τ_h. the temperature

values can be non-dimensionalised with the steady state temperature by solving the uncoupled heat transfer equation for T_d when dT/dt is zero:

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) \tag{7.3a}$$

$$0 = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) \tag{7.3b}$$

$$T_d = T_G \tag{7.3c}$$

The results from a simulation run for one droplet of water evaporating in air for a time step of τ_d are shown in Figure 7.1.

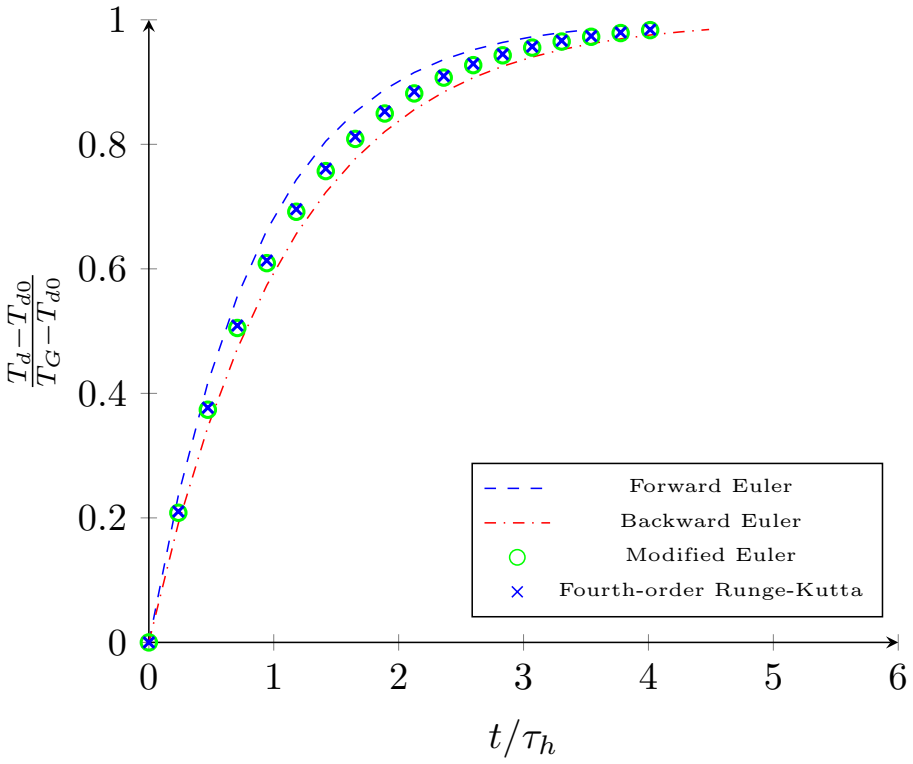


Figure 7.1: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = \tau_d$.

As expected the first order implicit and explicit methods undershoot and overshoot the analytic solution respectively. This is because an explicit method makes an estimate of what the next value will be based on the current gradient. Given the gradient of the analytic solution is always decreasing this means the explicit method is always overestimating the value at the next timestep. A similar argument can be made for an implicit method.

It can also be seen from Figure 7.1 the droplet heats up to the gas temperature within just over $4\tau_h$. As $1/e^4$ is $\approx 2\%$ and the simulation stops once the droplet temperature has reached 99.9% of the gas temperature this proves the droplet temperature increases at the correct rate.

The timestep for example be halved to $\tau/2$ as in Figure 7.2.

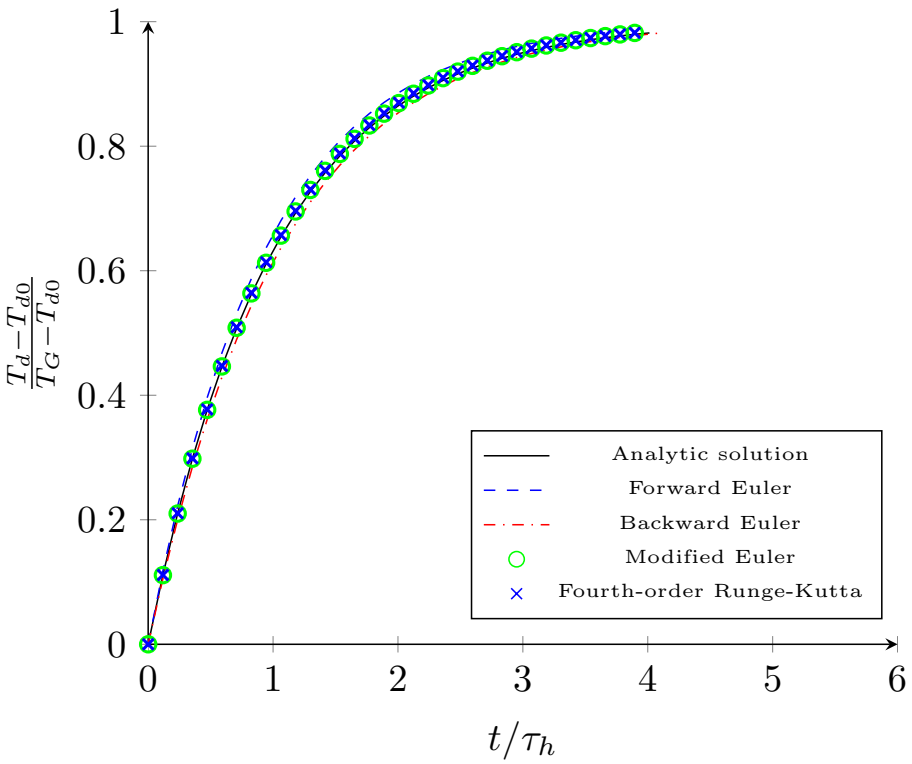


Figure 7.2: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = \tau/2$.

Figure 7.2 shows the numerical solutions are closer to the analytic solution compared with Figure 7.1. Therefore demonstrating the numerical solutions converge to the analytic solution as expected.

7.2.3 Comparison of Numerical Methods

The convergence of numerical results to the analytic solution for decreasing timesteps is highlighted by Figure 7.3.

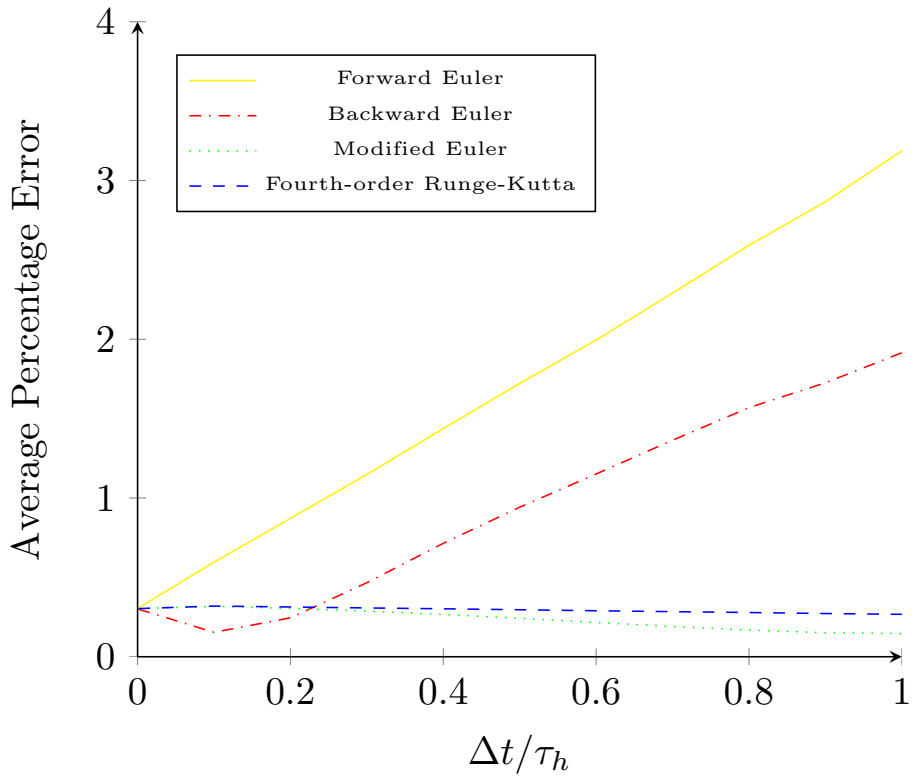


Figure 7.3: Average percentage error compared with the analytic solution for a range of timesteps and numerical methods.

For a timestep very close to zero all the numerical methods appear to have the same average percentage error. This should be zero but is actually 0.3%. This is quite an unusual result, especially when considering the average percentage error decreases with increasing timestep for the modified Euler and Runge-Kutta methods.

This turned out to be a bug in the method used for benchmarking the statistics. The error was that The simulation stops once $T_d = 0.999T_G$. Removing this stop condition and fixing the simulation lengths using the max number of timesteps produces statistics that make sense:

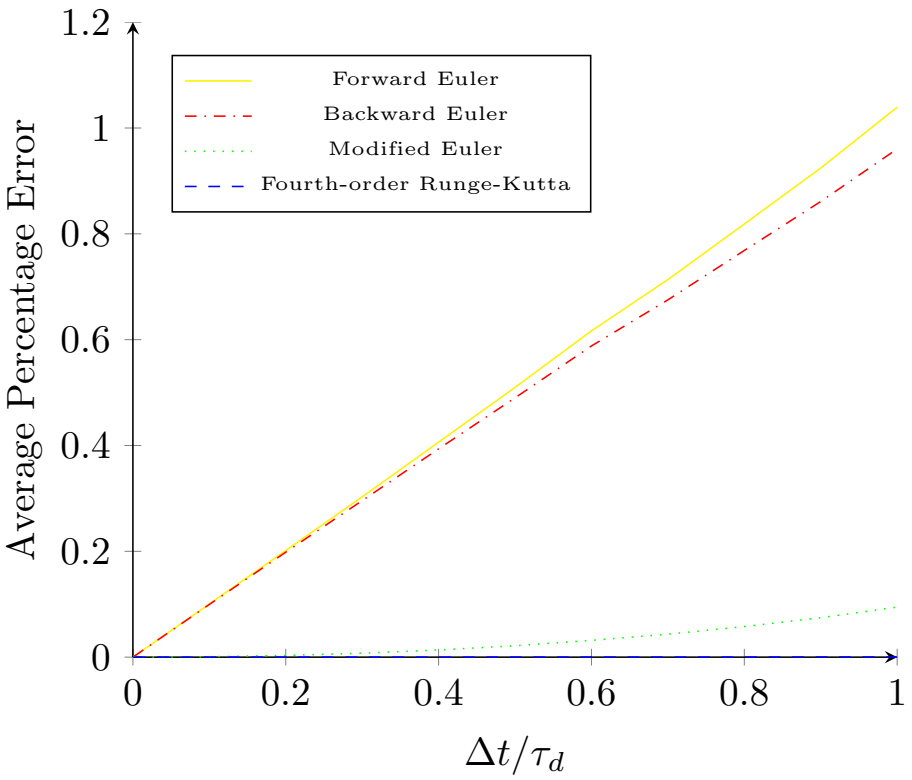


Figure 7.4: Average percentage error compared with the analytic solution for a range of timesteps and numerical methods.

The simulation duration was fixed to τ_d and the number of timesteps calculated by dividing this by the timestep size. As Figure 7.4 shows all methods converge to zero error with a timestep size of zero. The forward Euler method has perfect linearity with a R^2 value of 0.9998 as calculated by Excel. The backward Euler method has a slightly smaller error, but again has good linearity with $R^2 = 0.9996$. The modified Euler and Runge-Kutta methods now have noticeably different gradients for their error plots. It is just possible to distinguish that the modified Euler method has a quadratic error function, with $R^2 = 0.9996$. Due to how small the average percentage error of the Runge-Kutta method is this has also been plotted separately on Figure 7.5.

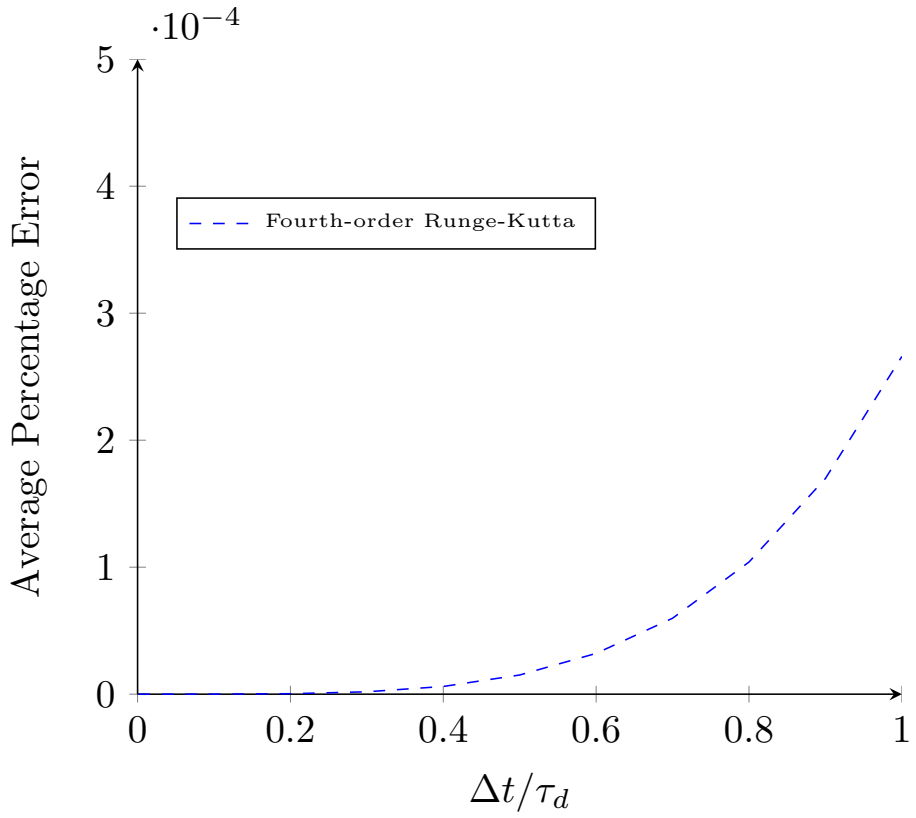


Figure 7.5: Average percentage error compared with the analytic solution for the Runge-Kutta method for a range of timesteps.

The average percentage error as a function of the timestep for the Runge-Kutta method is a quartic as expected given the method is fourth-order. The statistics are again well behaved with $R^2 = 1.000$.

R^2 values close to 1 demonstrate the numerical methods have been implemented correctly as their results are to the correct order. Moreover, as the results for all numerical methods converge to the analytic solution for decreasing timestep size this verifies they are solving the uncoupled temperature case correctly.

7.3 Uncoupled Mass Transfer

7.3.1 Analytic Solution

The mass transfer equation is:

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\frac{m_d}{\tau_d}H_M \tag{7.4}$$

Unlike the uncoupled temperature equation, equation 7.4 requires some rearrangement before it can be solved. The term τ_d contains diameter which is a function of mass so this must be taken into account. In addition to this, the literature commonly plots D^2 instead of mass; solving equation 7.4 for diameter is possible and this is outlined in Section x. However, here equation 7.4 will solved for m_d and for plotting data the mass can be converted to D^2. In either solution case a relation between diameter and mass is needed.

Using the relation between mass, density and volume:

$$m_d = \rho V_d \tag{7.5a}$$

$$m_d = \rho_d \left(\frac{4}{3}\right) \pi \left(\frac{D}{2}\right)^3 \tag{7.5b}$$

$$m_d = \frac{\rho_d \pi D^3}{6} \tag{7.5c}$$

$$D = \left(\frac{6}{\rho_d \pi}\right)^{1/3} (m_d)^{1/3} \tag{7.5d}$$

Therefore, substituting this relation into 7.4 yields:

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\frac{m_d}{\tau_d}H_M \tag{7.6a}$$

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}H_M m_d \left(\frac{18\mu_G}{\rho_d D^2}\right) \tag{7.6b}$$

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}H_M m_d \left(\frac{18\mu_G}{\rho_d}\right)\left(\left(\frac{6}{\rho_d \pi}\right)^{-2/3} (m_d)^{-2/3}\right) \tag{7.6c}$$

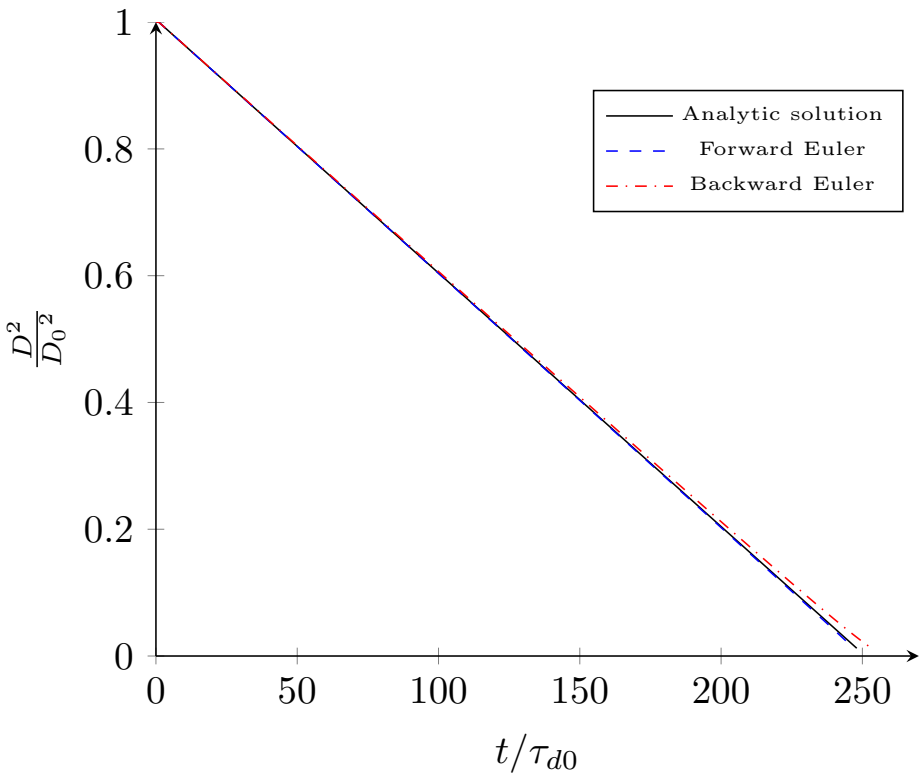
$$\frac{dm_d}{dt} = -\frac{Sh}{Sc_G}\mu_G H_M \left(\frac{6}{\rho_d}\right)^{1/3} \pi^{2/3} m_d^{1/3} \tag{7.6d}$$

The full solution as outlined in Section B.5.2 gives:

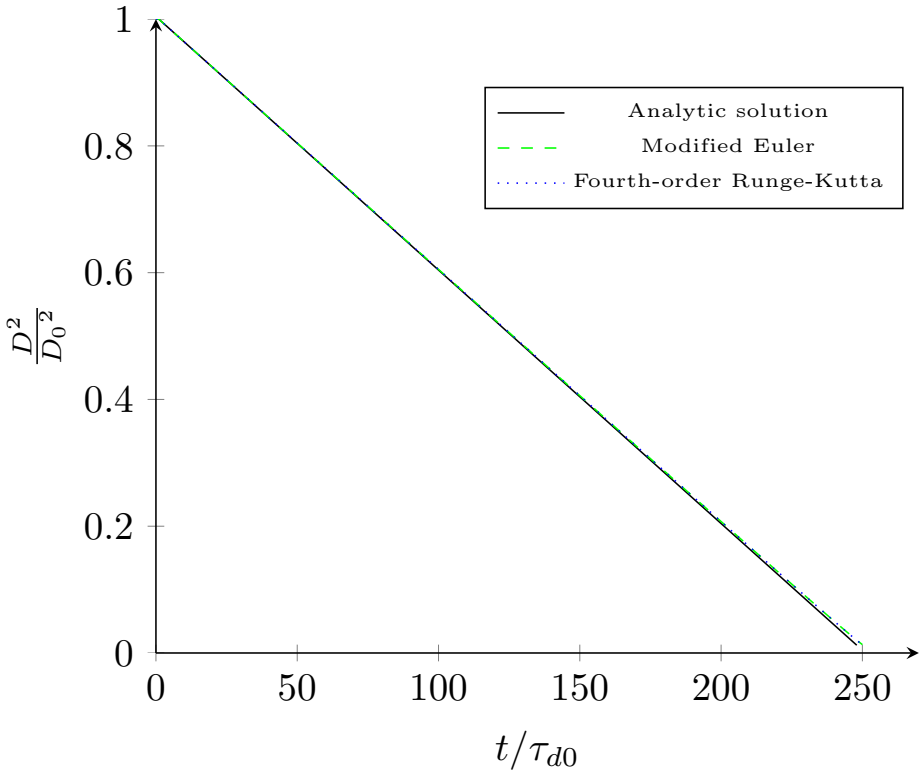
$$(m_{dn+1})^{2/3} = (m_{dn})^{2/3} - \frac{2Sh}{3Sc_G}\mu_G H_M \left(\frac{6}{\rho_d}\right)^{1/3} \pi^{2/3}\ \Delta t \tag{7.7}$$

7.3.2 Results

The mass transfer results have been plotted in Figure 7.6a for $\Delta t = \tau_d$. For the numerical solution τ_d was recalculated at the start of each timestep. At the end of each timestep the mass value is used to calculate D^2, which is non-dimensionalised with the initial diameter. The results from the modified Euler and Runge-Kutta methods have been plotted separately on Figure 7.6b.



(a)



(b)

Figure 7.6: Non-dimensionalised D^2 for a droplet sized $D_0 = \sqrt{1.1}mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = \tau_d$.

Much like was the case for the temperature solution the explicit methods overshoot the

analytic solution. The exception being the forward Euler method under-predicts the analytic solution and appears to better predict the the analytic solution. The droplet evaporation time has been non-dimensionalised with the initial droplet momentum timescale as this changes as the droplet's mass decreases.

Note that unlike the heat transfer solution the results of the mass transfer plotted as diameter squared give a linear relation. This is intrinsic to the assumption the droplet temperature remains at the wet bulb temperature. Therefore the rate of evaporation is only dependent on surface area which is a function of D^2.

7.3.3 Comparison of Numerical Methods

To confirm the numerical methods converge to the analytic solution a similar convergence plot to Figure 7.4 has been plotted in Figure 7.7. For this convergence test the simulation time was limited by the mass stop condition.

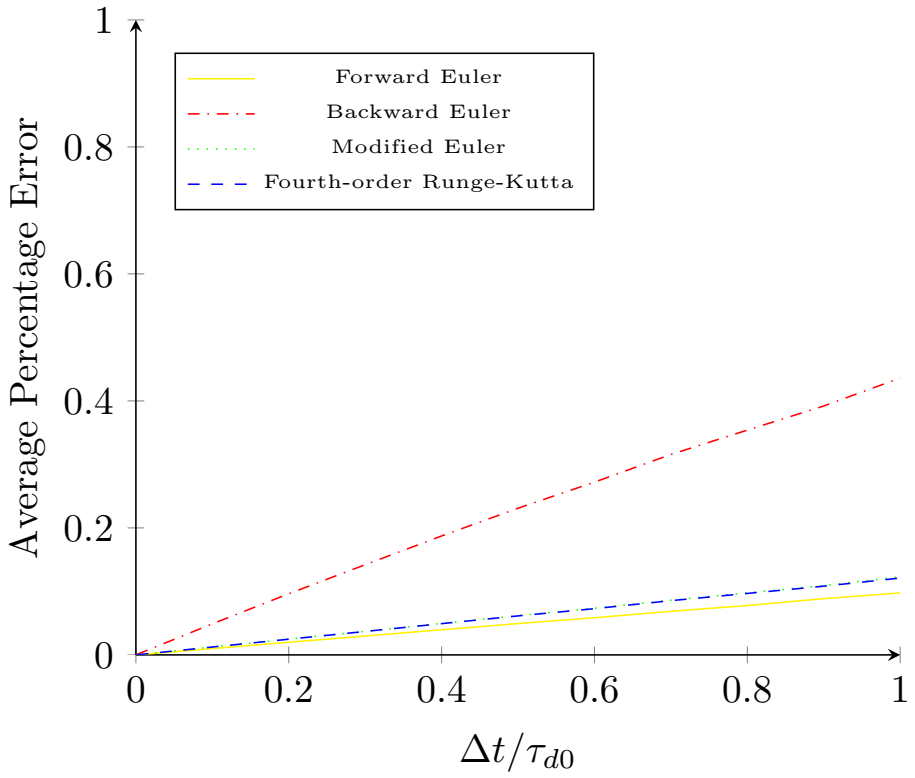


Figure 7.7: Average percentage error compared with the analytic solution for a range of numerical methods and timesteps.

As should be the case all the numerical methods converge to zero percentage error for a timestep of zero. What is unusual is the modified Euler and Runge-Kutta methods have a slightly larger error than the forward Euler method. This may be because the high order methods are multistep. At each step within the method a prediction about what the solution is, is made. For all steps the same τ_d is used. This would explain why the modified Euler and Runge-Kutta methods have such similar error functions.

This can be fixed by updating τ_d for each step which as can be seen in Figure 7.8 and 7.9 the error functions are now as expected.

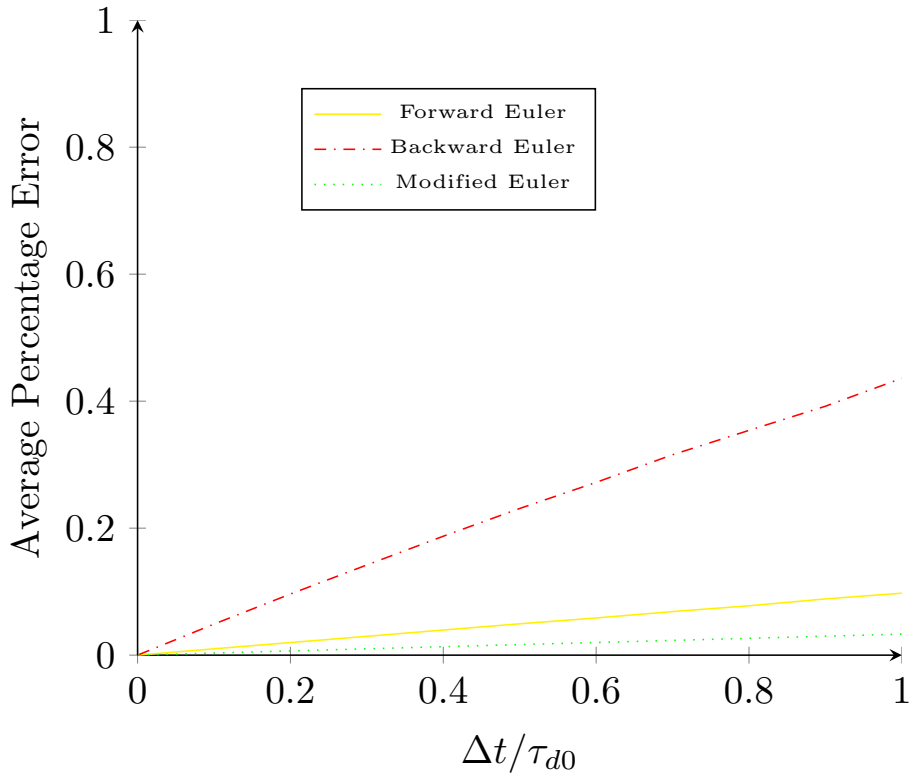


Figure 7.8: Average percentage error compared with the analytic solution for a range of numerical methods and timesteps.

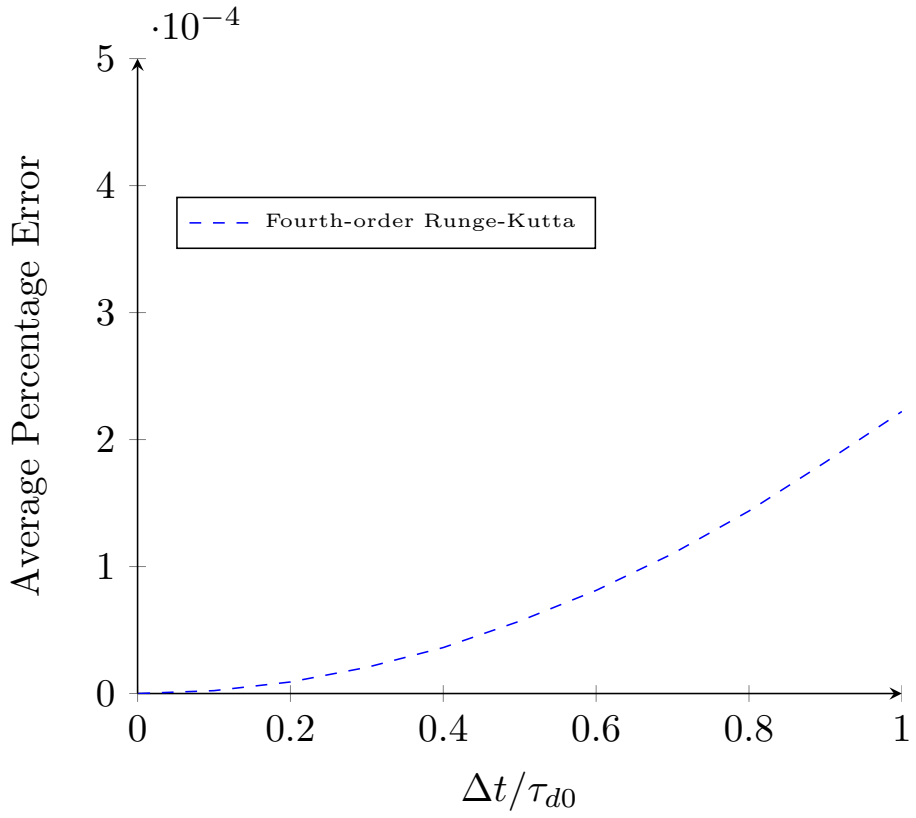


Figure 7.9: Average percentage error compared with the analytic solution for the Runge-Kutta method and a range of timesteps.

Like with the temperature convergence the R^2 values for the D^2 error functions can be calculated. They are as follows:

- Forward Euler - $R^2 = 0.9976$
- Backward Euler - $R^2 = 0.9999$
- Modified Euler - $R^2 = 1.000$
- Runge-Kutta - $R^2 = 1.000$

This affirms the numerical methods perform as expected.

The verification tests here confirm the importance of verifying simulation code to find bugs.

7.4 Coupled Heat and Mass Transfer

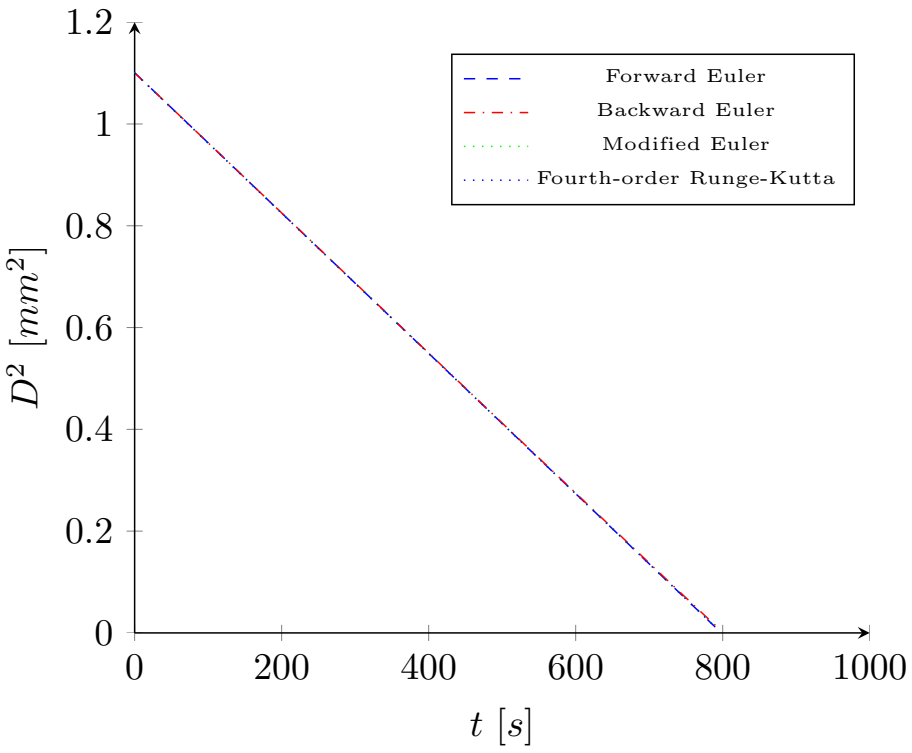
Solving the heat and mass transfer ODEs coupled is not a trivial task due to their non-linear nature. An analytic solution for the heat and mass transfer ODEs coupled has not been found so the equations can only be solved numerically.

The simulation settings used are as per Table 7.1.

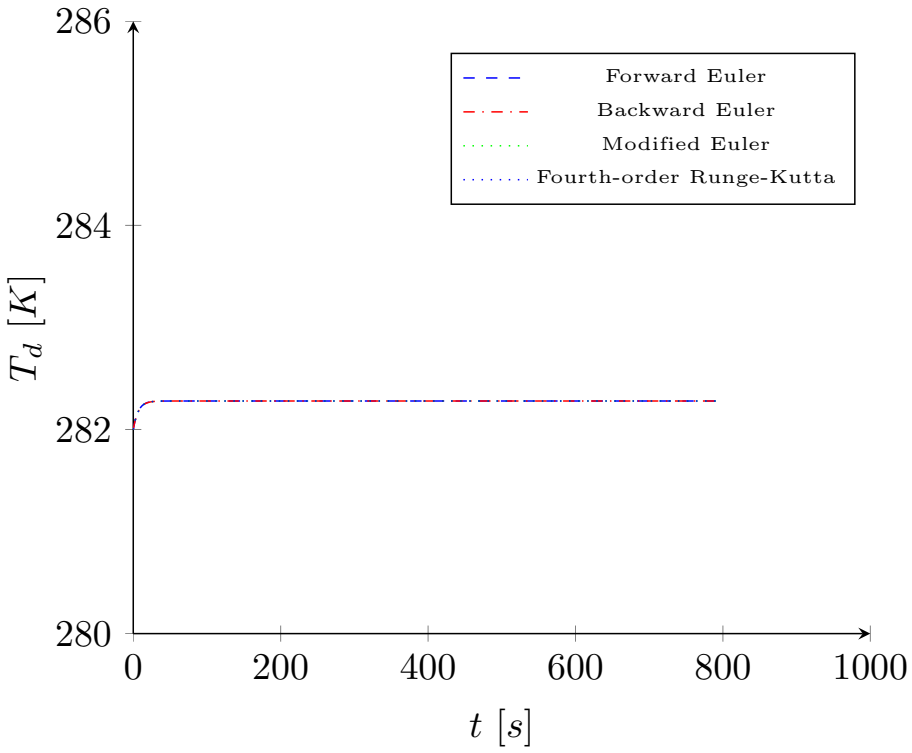
7.4.1 Results

The testing procedure is the same as for the uncoupled cases. With multiple numerical methods and timestep sizes being tested.

Figures 7.10a and 7.10b show the results for the mass and heat transfer.



(a)



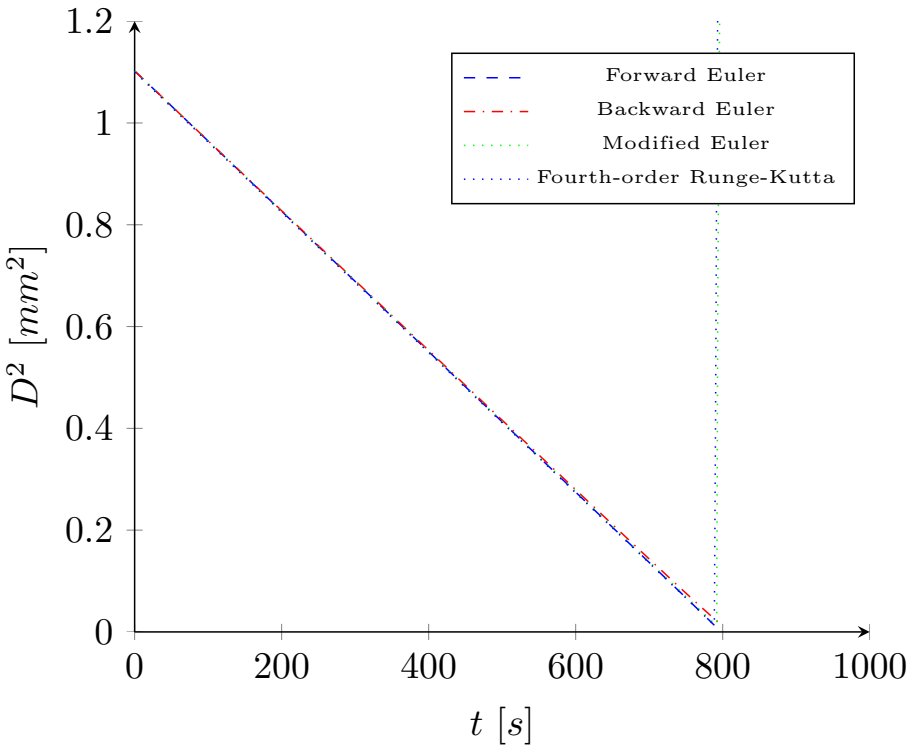
(b)

Figure 7.10: Droplet diameter squared (a) and droplet temperature (b) temporal evolution for a water droplet sized $D_0 = \sqrt{1.1}mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = \tau_d/8$.

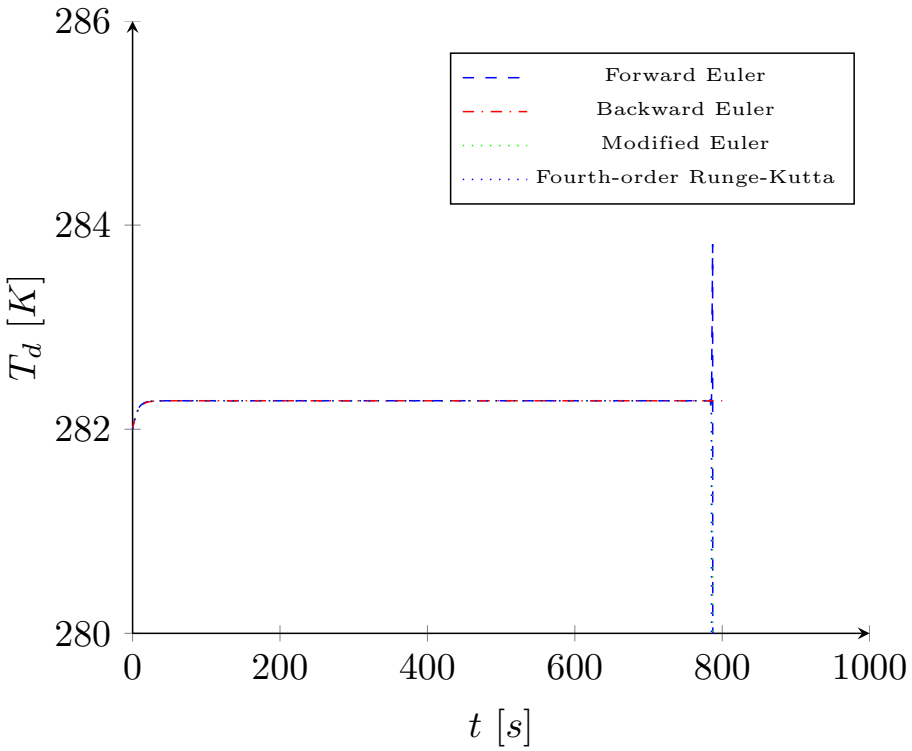
The droplet quickly reaches an equilibrium temperature of $282.27K$ and remains at this temperature as the droplet evaporates at a steady rate. There is little variation between the numerical methods.

7.4.2 Comparison of Numerical Methods

Increasing the size of the timestep makes the difference between the numerical methods more evident. Using a timestep of $\tau_d/2$ it can be seen the explicit methods diverge quite significantly at the end of the simulation in Figures 7.11a and 7.11b.



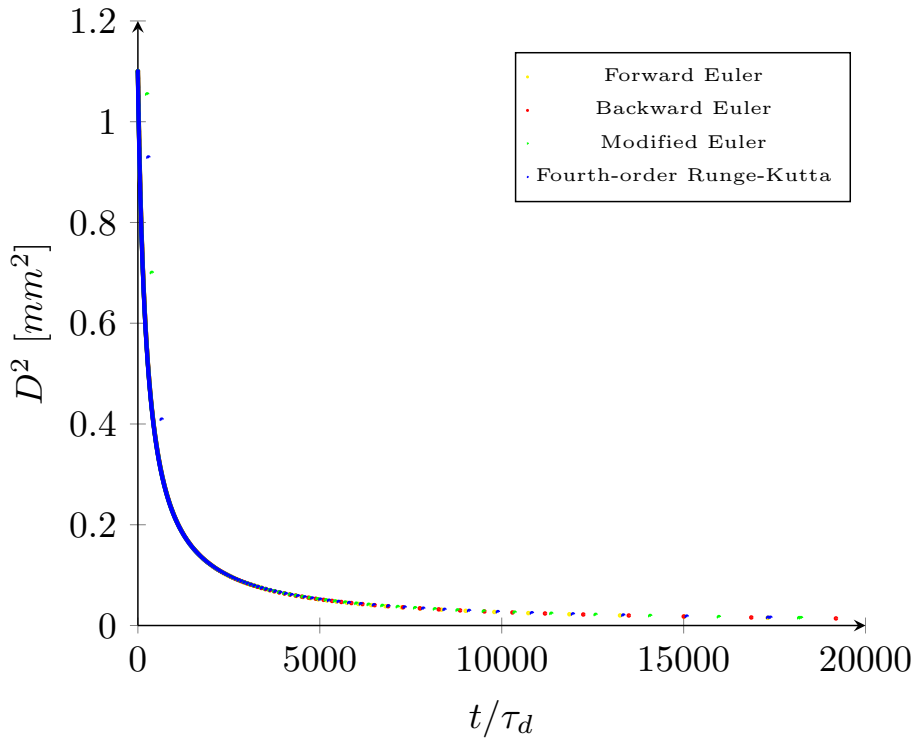
(a)



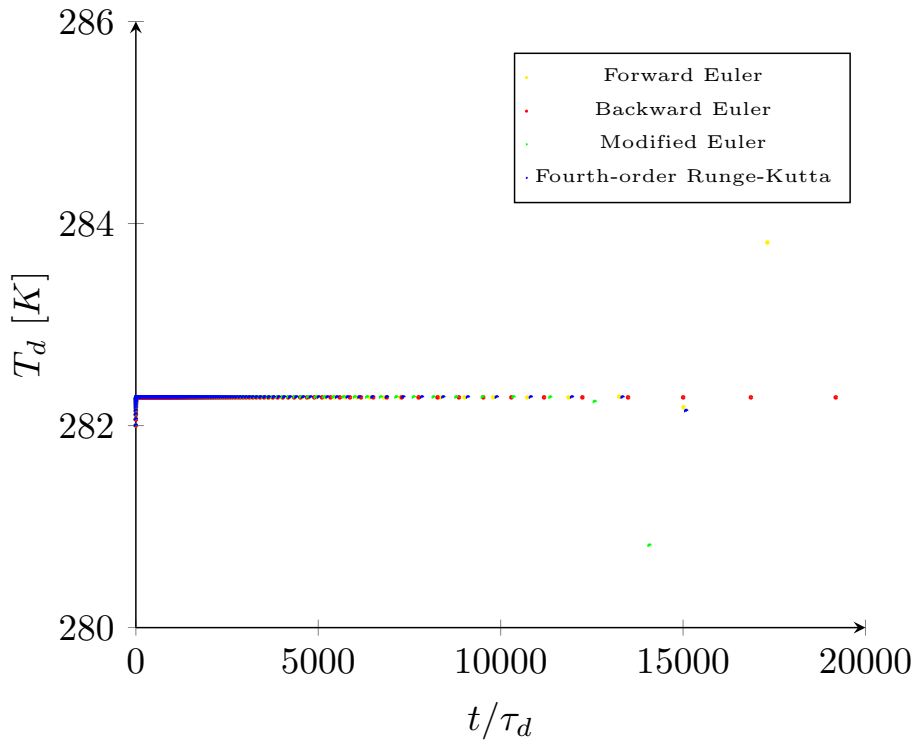
(b)

Figure 7.11: Droplet diameter squared (a) and droplet temperature (b) temporal evolution for a water droplet sized $D_0 = \sqrt{1.1}mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = \tau_d/2$.

Plotting the results non-dimensionally in Figure 7.12a and 7.12b using τ_d makes it clear what causes this:



(a)



(b)

Figure 7.12: Droplet diameter squared (a) and droplet temperature (b) temporal evolution for a water droplet sized $D_0 = \sqrt{1.1}mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = \tau_d/2$. The time axis is non-dimensionalised with τ_d.

The momentum timescale of the droplet, τ_d decreases as the droplet diameter decreases. This means the timescale of the physics increases relative to the size of the timestep as the

ODEs are a function of $1/\tau_d$. As explicit methods extrapolate based on the current value to find the next value; the error incurred in the extrapolation increases with the size of timestep. A possible solution to this problem is dynamic timestepping which involves changing the size of the timestep as a function of τ_d. For one droplet this procedure is simple enough, but for multiple droplets the timestep is common across all droplets. This means basing the timestep size on the smallest droplet timescale, which could be very inefficient. Consider a scenario where one droplet at the start of the simulation a droplet rapidly evaporates, this means a small timestep size for droplets which may be very slowly evaporating.

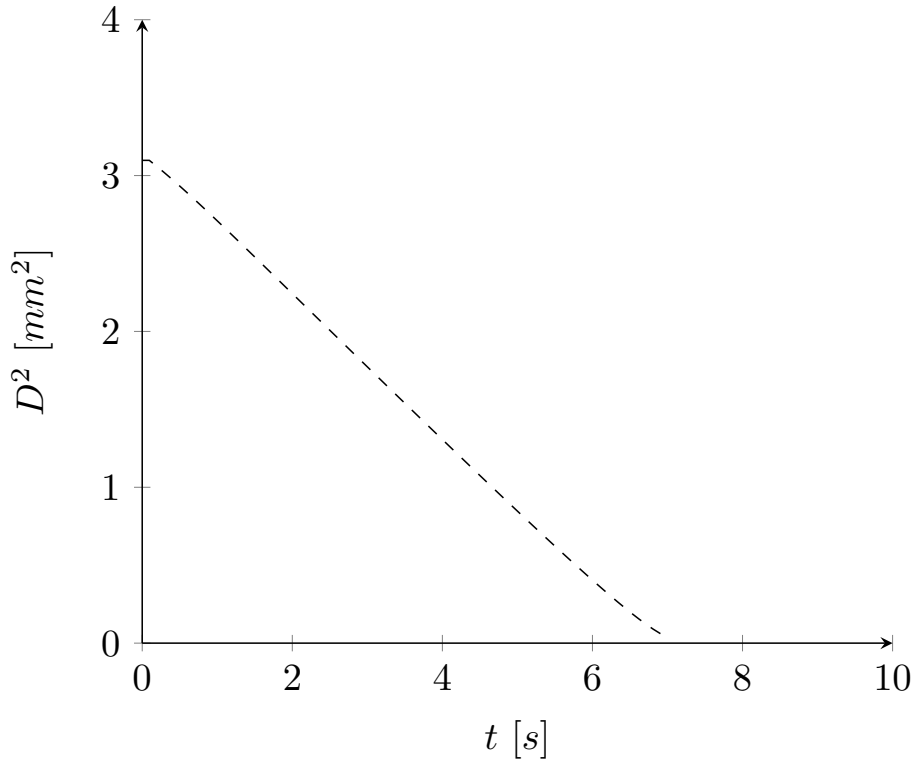
Therefore, the required solution is to use an implicit method. The backward Euler method is appropriate to use here given it does not suffer the same instability as the explicit methods. Further, it permits the usage of larger timesteps which is advantageous in reducing computation time. Higher order implicit methods are available, but at the cost of an iterative procedure at each timestep to solve the equations. Therefore the implicit Euler method will be used for the OpenCL simulation code.

7.5 Coupled Heat and Mass Transfer Validation

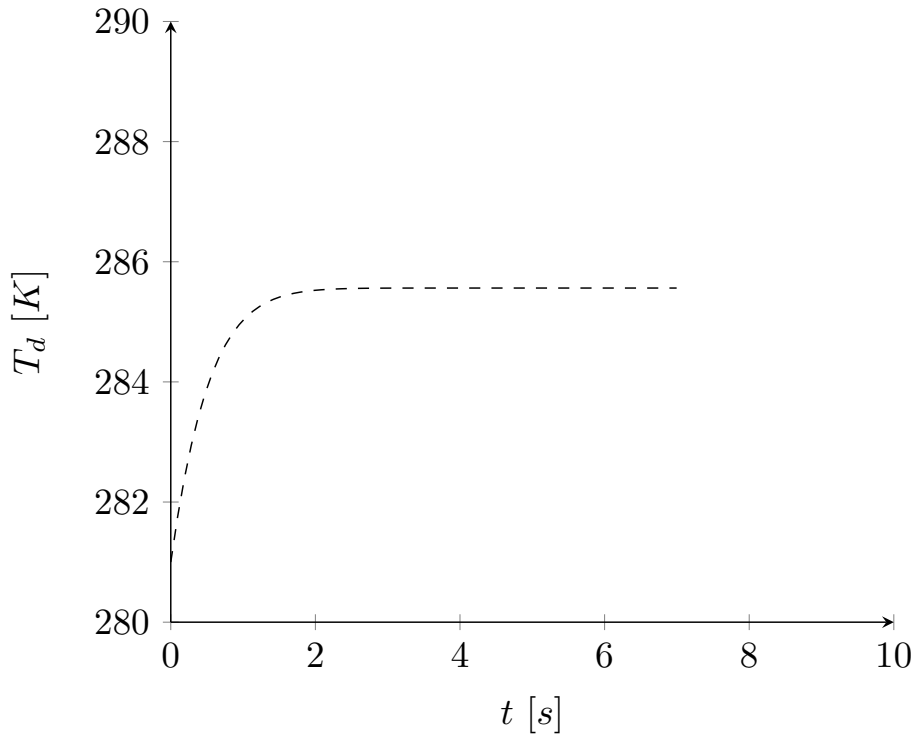
As a suitable procedure has been chosen for running the simulation the procedure can be validated using a further two common test cases found in the literature. The first is the evaporation of a hexane droplet using the conditions from [5]. These are outlined in Table 7.2.

Parameter	**Setting**
Droplet Physical Properties	
Molecular Weight of Vapour Phase W_V	86.178 $kg/kg\ mol$
Boiling Temperature T_B	447.7 K
Density of Liquid Phase ρ_L	664 kg/m^3
Specific Heat Capacity of Liquid Phase C_L	2302 $J/(kg\ K)$
Initial Temperature T_{d0}	281 K
Initial Diameter D_0	1.76 mm
Reynolds Number Re_d	110
Gas Properties	
Molecular Weight of Phase W_C	28.97 $kg/(kg\ mol)$
Temperature T_G	437 K
Density ρ_G	0.807 $kg\ m^{-3}$
Specific Heat for Constant Pressure $C_{P,G}$	1020 $J\ kg\ K$
Free Stream Vapour Mass Fraction Y_G	0
Atmospheric Properties	
Pressure P_{atm}	101325 Pa
Physical Constants	
Gas Constant $\bar{R}$	8314.5 $J/(K(kg\ mol))$
Universal Gas Constant R	287 $J/(kg\ K)$
Simulation Properties	
Correction factor f_2	1
Driving Potential $H_{\Delta T}$	0
Timestep Size Δt	$\tau_{d,0}/64$

Table 7.2: Simulation settings used for the hexane droplet simulation.



(a)



(b)

Figure 7.13: Droplet diameter squared (a) and droplet temperature (b) temporal evolution for a hexane droplet sized $D_0 = 1.76mm$ with $Re_d = 110$, $T_{d_0} = 281$, $T_G = 437$ and $\Delta t = \tau_{d0}/64$.

As with the results of M1 in the literature the model over predicts droplet evaporation rate. Nevertheless, despite the lack of usage of the "1/3 rule" the results are quite similar

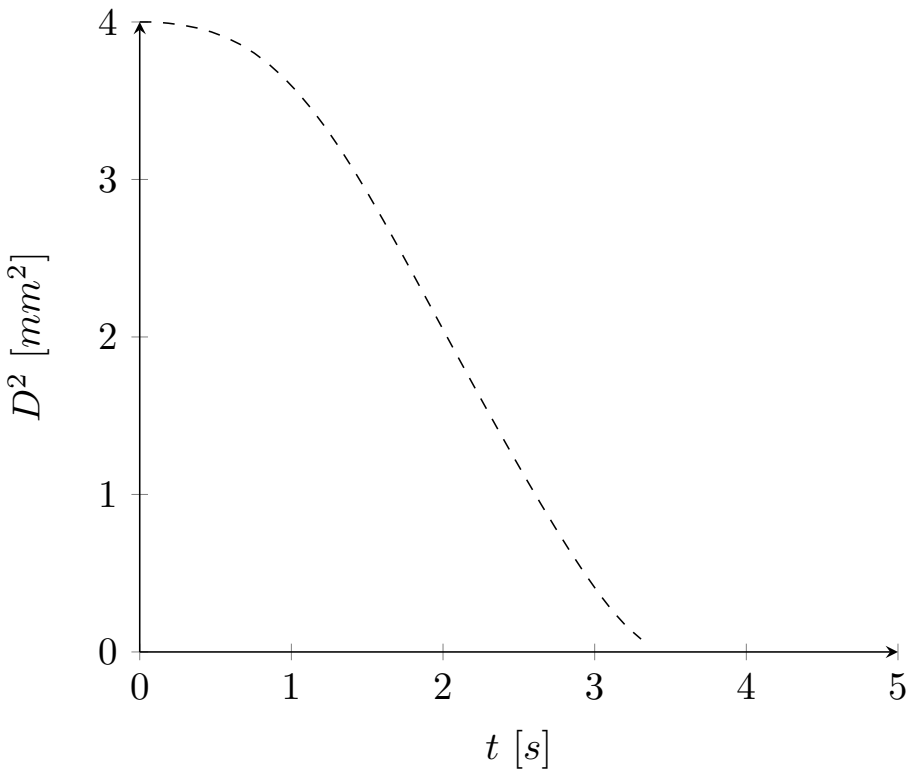
to that in [2] in terms of droplet evaporation rate. However, the model appears to reach a lower equilibrium droplet temperature.

The second validation simulation is for an evaporating decane droplet and uses the conditions from [28]. The simulation settings are outlined in Table 7.3.

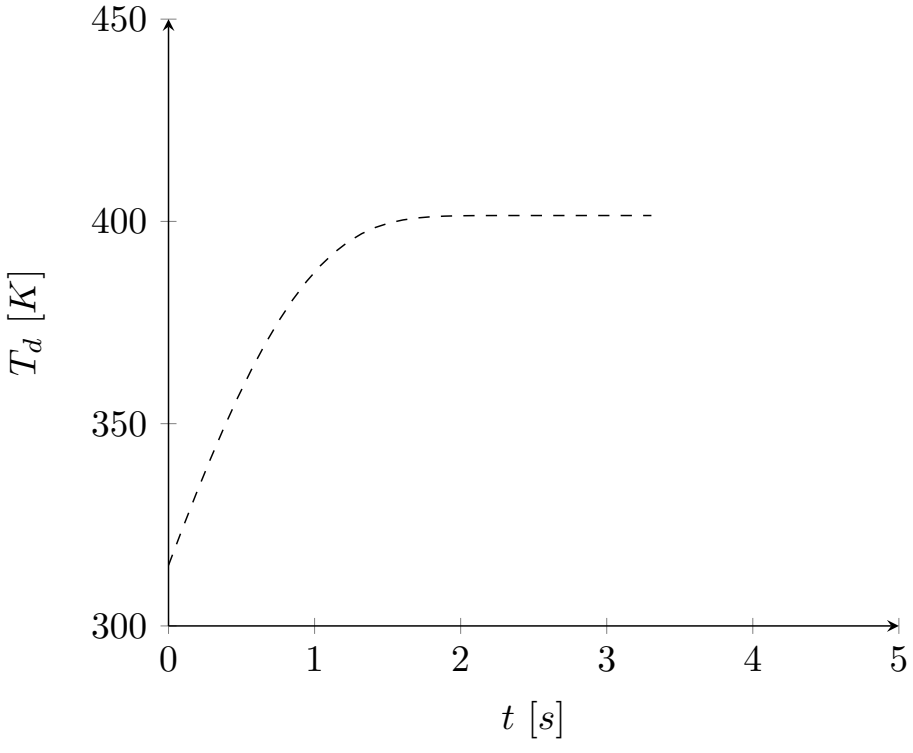
Parameter	**Setting**
Droplet Physical Properties	
Molecular Weight of Vapour Phase W_V	142 $kg/kg\ mol$
Boiling Temperature T_B	447.7 K
Density of Liquid Phase ρ_L	642 kg/m^3
Specific Heat Capacity of Liquid Phase C_L	2520.5 $J/(kg\ K)$
Initial Temperature T_{d0}	315 K
Initial Diameter D_0	2.0 mm
Reynolds Number Re_d	17
Gas Properties	
Molecular Weight of Phase W_C	28.97 $kg/(kg\ mol)$
Temperature T_G	1000 K
Density ρ_G	0.3529 $kg\ m^{-3}$
Specific Heat for Constant Pressure $C_{P,G}$	1141 $J\ kg\ K$
Free Stream Vapour Mass Fraction Y_G	0
Atmospheric Properties	
Pressure P_{atm}	101325 Pa
Physical Constants	
Gas Constant $\bar{R}$	8314.5 $J/(K(kg\ mol))$
Universal Gas Constant R	287 $J/(kg\ K)$
Simulation Properties	
Correction factor f_2	1
Driving Potential $H_{\Delta T}$	0
Timestep Size Δt	$\tau_{d,0}/64$

Table 7.3: Simulation settings used for the hexane droplet simulation.

The results presented in Figure 7.14a and 7.14b show that like for the hexane case the model performs similarly to the common implementation of M1 in the literature. Again, the model over predicts the evaporation rate compared with experimental data. Although, it fairs much better compared with the experimental data when considering the droplet temperature.



(a)



(b)

Figure 7.14: Droplet diameter squared (a) and droplet temperature (b) temporal evolution for a decane droplet sized $D_0 = 2.0mm$ with $Re_d = 17$, $T_{d_0} = 315K$, $T_G = 1000K$ and $\Delta t = \tau_{d0}/64$.

8 Verification and Validation of the Existing Code

Before the heat and mass transfer methods can be added to the existing code some modifications to it had to be made. The existing code calculates the position and velocity of the particles but also includes a discrete element method (DEM) to simulate the collision of particles. The latter is computationally expensive and is not needed for the heat and mass transfer calculations. The DEM code can be "switched off" so the existing code solves just for the particle position and velocity. There are verification and validation checks that need to be done to ensure the modified code is solving the equations correctly and producing physical results.

The cut down code can be found at [29].

8.1 Analytic Test Case

One very simple check that can be made initially is to check if results from the code converge to some analytic solution. The ideal test case for this is the droplet accelerating under gravity to reach a terminal velocity. The differential equation to be solved is:

$$\frac{du}{dt} = g + \frac{1}{\tau_d}(u_f - u) \tag{8.1}$$

The full solution procedure can be found in Section B.1. This leads to:

$$u = \tau_d \left(g + \frac{u_f}{\tau_d} \right) (1 - e^{-t/\tau_d}) \tag{8.2}$$

To simplify this further the fluid velocity can be set to zero, yielding the simplified formula:

$$u = \tau_d g (1 - e^{-t/\tau_d}) \tag{8.3}$$

The settings used for the test case are shown in Table 8.1:

Parameter	**Setting**
Droplet Properties	
Density of Liquid Phase ρ_L	997 kgm^{-3}
Initial Diameter D_0	0.05 m
Initial Velocity u_0	0 ms^{-1}
Gas Properties	
Kinematic Viscosity μ_G	0.1 $kgm^{-1}s^{-1}$
Physical Constants	
Gravity g	9.81 ms^{-1}

Table 8.1: Simulation settings used for acceleration under gravity test case.

For non-dimensionalising the results τ_d has been used for time and the terminal velocity has been used u_i. See Section B.6 for a derivation of the terminal velocity formula shown in equation 8.4.

$$v_{final} = u - g\frac{\rho_d D^2}{18\mu_G} \tag{8.4}$$

Using the settings from Table 8.1 the terminal velocity is:

$$v_{final} = 0ms^{-1} - (9.81ms^{-2})\left(\frac{(997kgm^{-3})(0.05m)^2}{(18)(0.1kgm^{-1}s^{-1})}\right) \tag{8.5a}$$

$$v_{final} = -13.58\ ms^{-1} \tag{8.5b}$$

Figure 8.1 shows the analytic and numerical solutions for the gravity test case. This shows the particle accelerating to the terminal velocity. As explained earlier, as the numerical solution for the velocity uses an implicit method this under predicts the analytic solution.

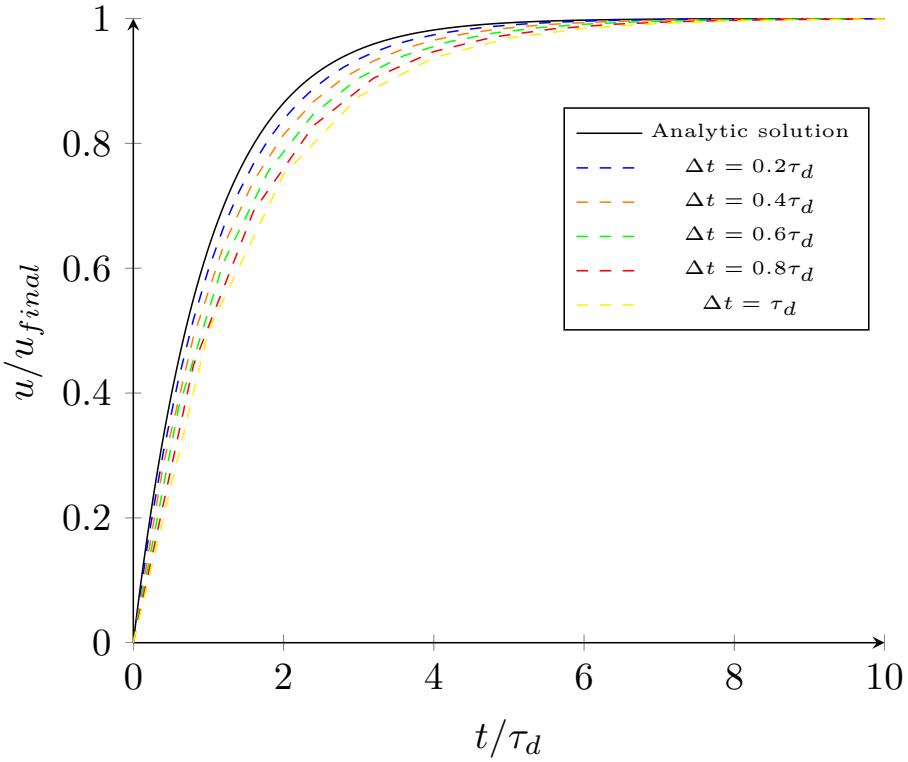


Figure 8.1: Non-dimensionalised velocity for a droplet sized $D^2 = 50mm$ with $Re_{d0} = 0$, and $\Delta t = \tau_d/2$ and $\Delta t = \tau_d/8$.

As with previous tests it is important to check the solution produced by the code converges to the analytic solution. For this test case the simulation time was set as 21 seconds which is $\sim 15\tau_d$. The results from this are shown in Figure 8.2.

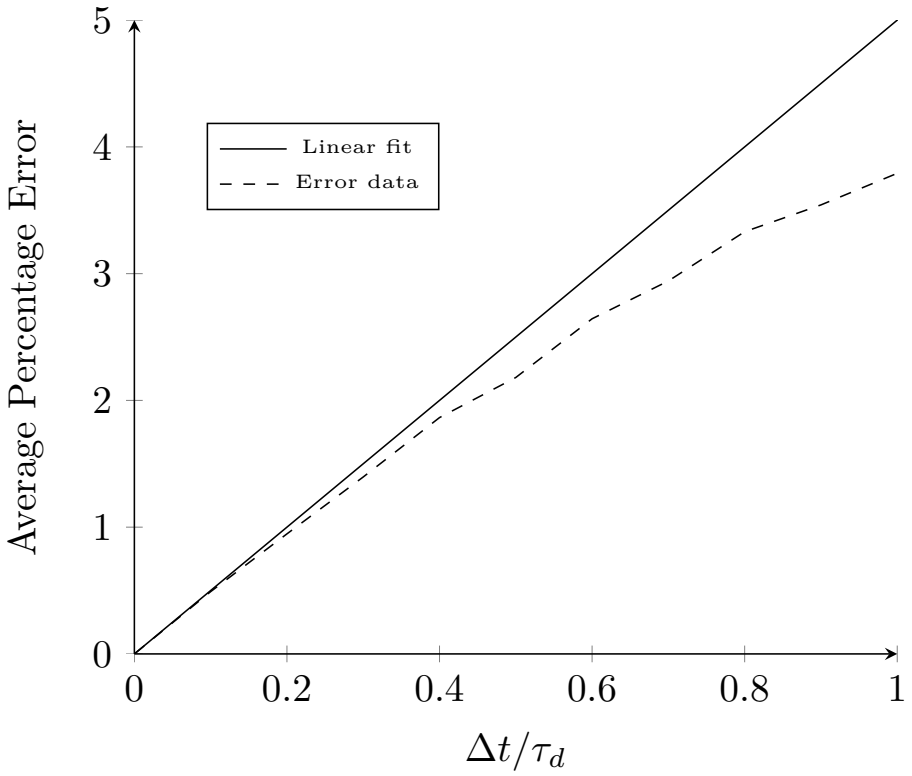


Figure 8.2: convergence of numerical results for gravity test case for a droplet sized $D^2 = 50mm$ with $Re_{d0} = 0$, and a range of timesteps.

Figure 8.2 clearly indicates convergence for the gravity test case. As well as confirming the velocity equation is solved with higher than first order accuracy as reported in [1].

8.2 Statistical Test Case

Another check that can be made is to find if the code produces statistically stationary conditions. This can be done by calculating the relative velocity. I.e. the difference between the particle velocity and the fluid velocity at different times. The particle velocity is a Lagrangian variable (only varying with time). Whilst the fluid velocity is an Eulerian variable as it varies with both space and time. To calculate the relative velocity between the particle and fluid a Lagrangian description is needed for velocity.

The averaging procedure to get the relative velocities is:

$$\bar{U}_{x,rel}(t) = \sum_{i=1}^{N} \frac{\left[U_{p_i}(t) - U_{f_i}(t)\right]}{N} \tag{8.6a}$$

$$\bar{V}_{y,rel}(t) = \sum_{i=1}^{N} \frac{\left[V_{p_i}(t) - V_{f_i}(t)\right]}{N} \tag{8.6b}$$

$$\bar{W}_{z,rel}(t) = \sum_{i=1}^{N} \frac{\left[W_{p_i}(t) - W_{f_i}(t)\right]}{N} \tag{8.6c}$$

Where for each velocity component the relative velocity is found for all particles and averaged to produce the mean velocity component. The behaviour of the mean relative velocity will be random initially but should converge to some value at a later point in time. This behaviour should also be the same for the angle between the fluid and particle velocity vectors.

The Stokes number will dictate the behaviour of the particles. If it is $<< 1$ the particles are massless and should follow the flow. If the Stokes number is $>> 1$ the particles are

essentially billiard balls and the flow field of the velocity has no bearing on the motion of the particles.

The conditions used for the simulation of a Taylor-Green vortex were:

Parameter	Setting
Droplet Properties	
Density of Liquid Phase ρ_L	2000 kg/m^3
Initial Diameter D_0	0.05 m
Gas Properties	
Kinematic Viscosity μ_G	see Table 8.3
Flow Magnitude A	5
Vortex Frequency a	3
Non-Dimensional Numbers	
Stokes Number St	see Table 8.3
Physical Constants	
Gravity g	0 ms^{-1}
Simulation Properties	
Number of Particles	100

Table 8.2: Simulation settings used for statistical test case.

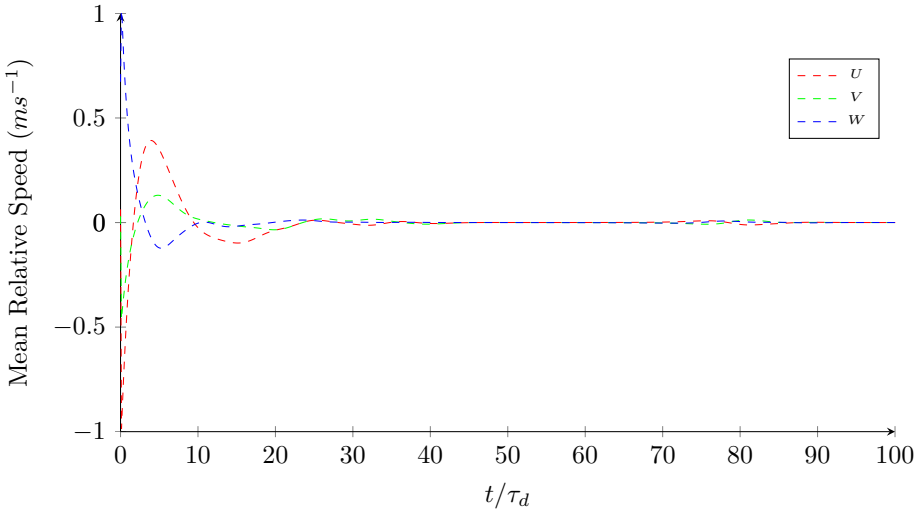
Kinematic Viscosity	Stokes Number
10.40	0.1
1.04	1.0
0.1040	10.0

Table 8.3: Settings used to get different Stokes numbers.

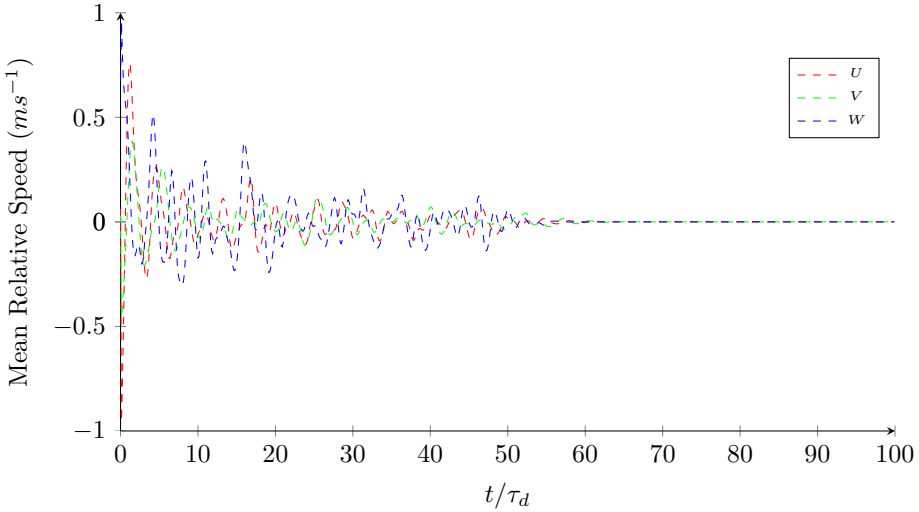
The distribution of starting conditions gives a mean particle speed of $1ms^{-1}$ and standard deviation of $0.1ms^{-1}$. With the particle directions evenly distributed.

Figure 8.3 shows the change in relative velocity between the particles and the fluid. As the Stokes number is increased the particles take longer to respond to the fluid. For a small Stokes number it can be seen that within around 20 momentum relaxation time constants the particle velocity has converged to the fluid velocity. This is further shown by Figure 8.4a where the angle between the particle and fluid velocity converges to zero.

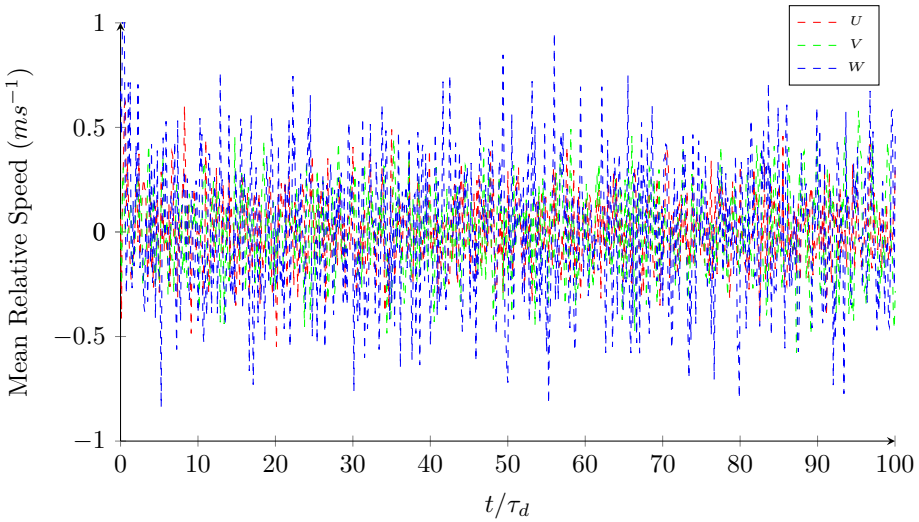
A similar behaviour is present for a Stokes number of 1. However, its takes around twice as long for the relative speeds to converge to zero. By contrast for a Stokes number of 10 the relative speeds and relative angles do not converge at all. This is to be expected as the fluid should have little bearing on the motion of the particles for this Stokes number.



(a) Stokes number of 0.1.

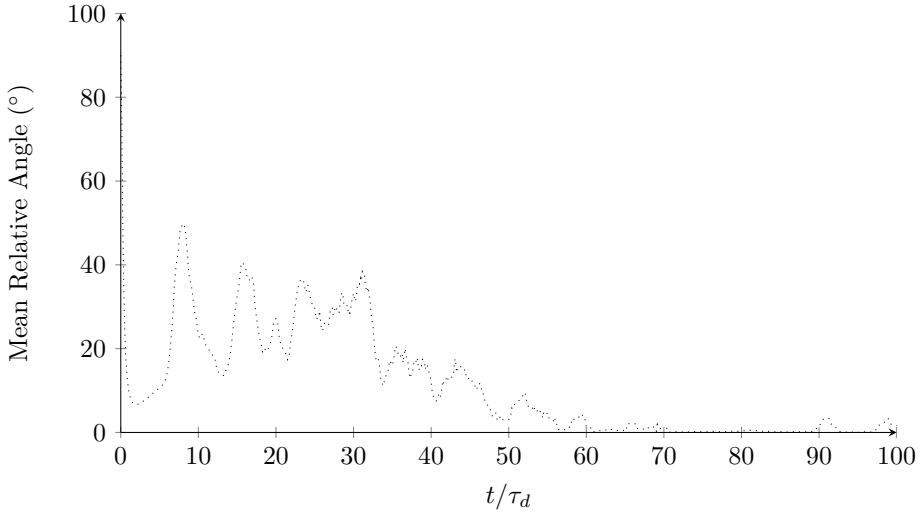


(b) Stokes number of 1.

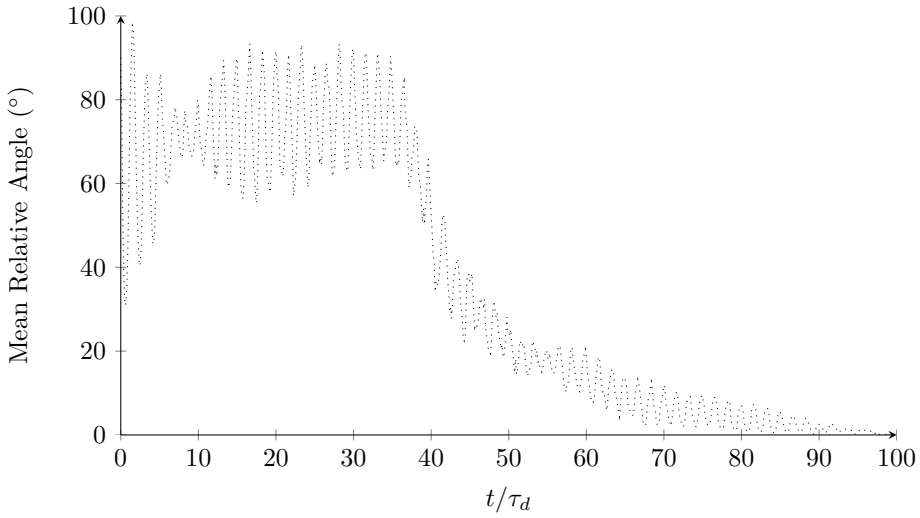


(c) Stokes number of 10.

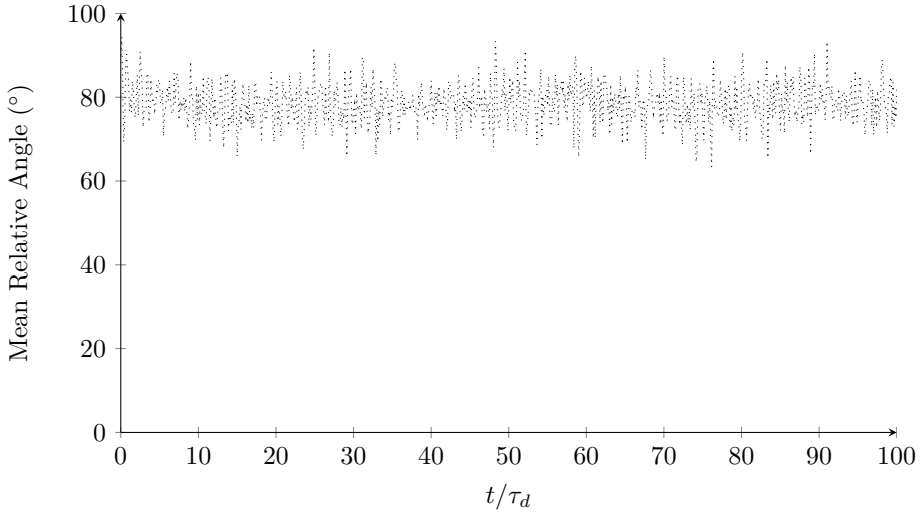
Figure 8.3: Plots of mean relative speed between the particle and fluid for each velocity component for Stokes numbers of 0.1 (a), 1.0 (b) and 10 (c).



(a) Stokes number of 0.1.



(b) Stokes number of 1.



(c) Stokes number of 10.

Figure 8.4: Plots of mean relative angle between the particle and fluid velocity vectors for Stokes numbers of 0.1 (a), 1.0 (b) and 10 (c).

8.3 Run Time Test

A final check on the existing code is to measure the run time. As the code uses first order numerical methods and runs on a GPU the run time of the code as a function of the number of particles should be linear.

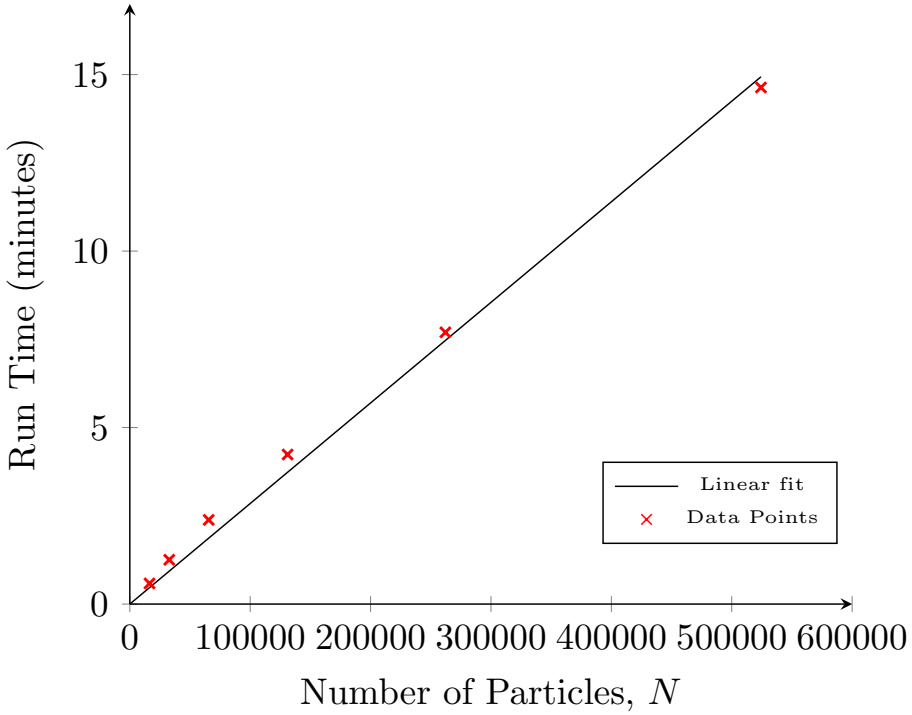


Figure 8.5: Run time of OpenCl code without data logging for different numbers of particles.

Figure 8.5 shows the results from this testing. For low numbers of particles the run time is not linear as the initialisation process is a large percentage of the overall simulation run time. However, for a larger number of particles this is no longer the case.

This set of results provides a point of comparison for when the heat and mass code is added. It should be expected that as first order methods for the heat and mass transfer equations have been used the run time of the code should still be linear.

8.4 Problems Encountered with the Existing Code

Listed in this section are bugs and problems encountered with the existing code. Bugs encountered were found by the author and fixed by E. Andrews with exception of the logstep issue. The full history of the code can be found at [30].

8.4.1 Periodic Boundary Condition

One misstep that was made in cutting down the existing code was removing the boundary condition for the particle location. The simulation domain is a cube; when a particle reaches the edge of the boundary its position is reset at the other side of the boundary as shown in Figure 8.6.

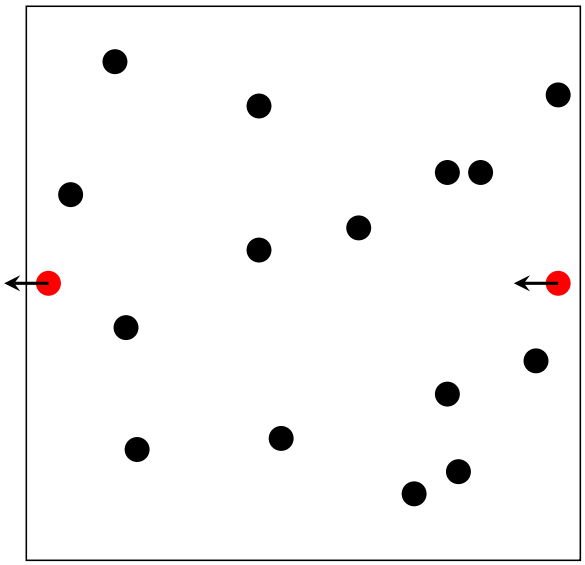


Figure 8.6: Diagram of a periodic boundary implementation.

This is a periodic boundary condition and ensures all the particles remain with the cube. Removing this boundary condition means the particles exit the domain at some velocity vector which remains unchanged as it is no longer influenced by the fluid.

8.4.2 Memory Alignment of Structures

The second issue encountered was adding a variable to the particle structure to record the fluid velocity. To ensure consistency between the host code and device code, structures passed to the device should be aligned. This means setting the order of the variables in the structure from the largest data type to the smallest. It also means for using a gcc compiler telling the compiler the required size of the structure. This is the size of the structure that fits within the smallest power of 2. For example, the size of the particle structure with the added variable can be determined from summing the size of the individual data types.

Data Type	Size (bytes)	No. of	Total Size (bytes)
cl_float3	12	5	60
cl_ulong	8	2	16
cl_float	4	4	16

Table 8.4: Data types and their respective sizes for the OpenCL language.

This gives a total size for the structure as 92 bytes. The next highest power of 2 is 128 so this is the size of the structure specified. For MSCV compilers the same value is used but the amount of padding must also be specified. This is the size specified minus the actual size. (In this case 36 bytes).

8.4.3 File Directory Checking

The code performs a number of tests during the setup of the simulation to prevent subsequent errors occurring.

To output the results of the simulation the code writes a text file at some user defined “log step”. For the logging to work the program needs to know an accessible logging directory

exists. It must therefore check this is the case.

When compiled with MSVC the code performs the check using `GetFileAttributes` as in Figure 8.7.

```
if (GetFileAttributes(dir) == INVALID_FILE_ATTRIBUTES) {
    return FALSE;
} else {
    return TRUE;
}
```

Figure 8.7: Check to see if logging directory exists with `GetFileAttributes`.

When compiled with gcc the code checks if the directory exists by trying to open and close the directory as shown in Figure 8.8.

```
DIR *dir_obj = opendir(dir);
if (dir_obj) {
    /* Directory exists. */
    closedir(dir_obj);
    return TRUE;
} else if (ENOENT == errno) {
    /* Directory does not exist. */
    return FALSE;
} else {
    /* opendir() failed for some other reason. */
    return FALSE;
}
```

Figure 8.8: Check to see if logging directory exists.

Originally the code used the `stat` structure to perform this check which only worked correctly on some Windows computers.

8.4.4 Initialisation of Particle Speeds

The code uses a Box-Muller transformation to turn uniformly distributed random numbers into a normal distribution for initialisation of particle speeds.

The process is as follows: Generate two random numbers u and v between 0 and 1 using `rand` and `RAND_MAX` as per Figure 8.9.

```
u = (float) rand() / (float) (RAND_MAX);
v = (float) rand() / (float) (RAND_MAX);
```

Figure 8.9: Creation of u and v.

Next, the normal distribution parameters (mean μ and standard deviation σ) must be specified. Then the normal distribution of speeds using the Box-Muller transform are created using:

$$speed = \sigma\sqrt{-2\ln(u)}\cos(2\pi v) + \mu \tag{8.7}$$

This formula is problematic in that it contains a log. If u is set to zero then $\ln(u)$ will tend to $-\infty$. This only occurred for two particles in a population of $100,000$ particles. This

does not affect the other particles but prevented Paraview from being able to read the text files produced by the code. The fix was to add 1 to the random numbers as in Figure 8.10 so that u and v can not be zero.

```
u = (float) (rand() + 1) / (float) (RAND_MAX + 1);
v = (float) (rand() + 1) / (float) (RAND_MAX + 1);
```

Figure 8.10: Ensuring u and v can not be zero.

8.4.5 Logsteps Being Skipped

The Python implementation of the code logs data at the end of every timestep. This solution is fine for one particle but not is not suitable when considering thousands of particles as the amount of data generated would be very large. The OpenCL code includes a logstep which determines how frequently data is logged. This means small timesteps can be used to increase accuracy and logsteps can be set larger than the timestep to reduce the amount of data generated.

The code responsible for this is contained within the main time loop of the simulation as presented in Figure 8.11.

```
if (LOG_DATA && time - last_write >= log_step) {
    ret = particlesToHost(queue, gparticles, &hparticles, NUMPART);
    printf("Logging at time: %f\n", time);

    if (!writeParticles(hparticles, time, prefix, log_dir, NUMPART, log_vel)) {
        return 1;
    }

    last_write = time;
}
```

Figure 8.11: Determining if data is logged during the timestep.

The boolean `LOG_DATA` is used to determine if data is to be logged or not. The second part of the condition is the current time subtract the time data was last written must be greater than or equal to the log step. For some values the second part of the if statement would not return true even when this should have been the case. (This is most likely a rounding error). Putting a "fudge factor" of 0.99 on the logstep as shown in Figure 8.12 fixed this problem.

```
if (LOG_DATA && time - last_write >= 0.99*log_step) {
```

Figure 8.12: Data logging fix.

9 OpenCL Implementation

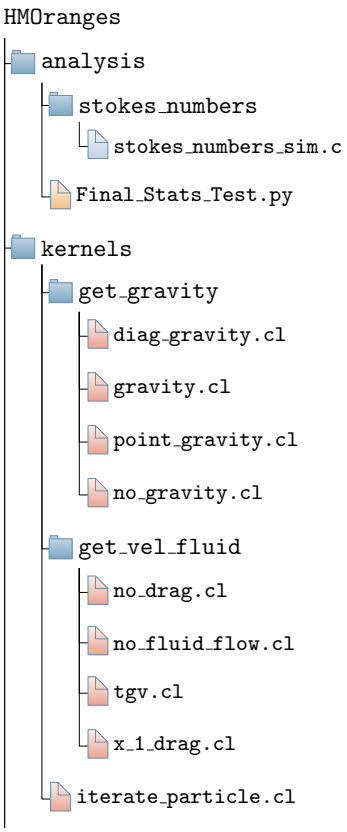
The initial testing of numerical methods and the model were performed in Python as this language allows for prototype code to be quickly produced due to its simplicity and feature rich libraries such as NumPy. Python also features capability for creating GPU code through the PyOpenCL library [31]. However, the existing code uses C and OpenCL to build executable files for running simulations.

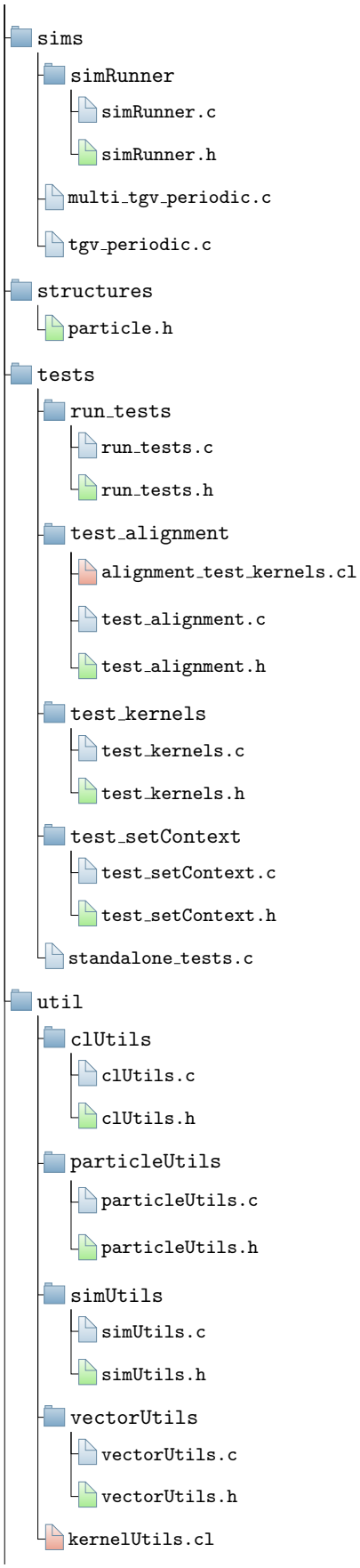
9.1 Additions made to the Code

The implementation of the heat and mass model was through adding the required code to the cut down code that was tested in Section 8.

9.1.1 Source Code Structure

For the sake of clarity the existing source code structure is shown in Figure D.1 which can be found in Section D. The new source code structure with DEM functionality removed and the heat and mass model included is shown in Figure 9.1.




sims
simRunner
simRunner.c
simRunner.h
multi_tgv_periodic.c
tgv_periodic.c
structures
particle.h
tests
run_tests
run_tests.c
run_tests.h
test_alignment
alignment_test_kernels.cl
test_alignment.c
test_alignment.h
test_kernels
test_kernels.c
test_kernels.h
test_setContext
test_setContext.c
test_setContext.h
standalone_tests.c
util
clUtils
clUtils.c
clUtils.h
particleUtils
particleUtils.c
particleUtils.h
simUtils
simUtils.c
simUtils.h
vectorUtils
vectorUtils.c
vectorUtils.h
kernelUtils.cl

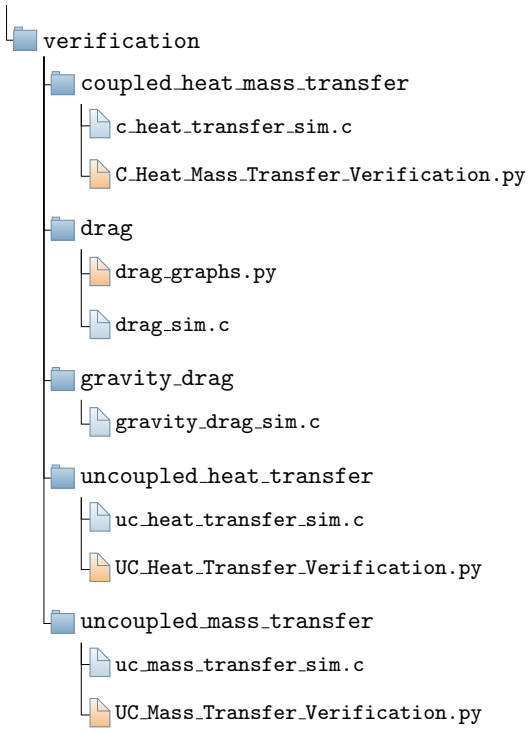


Figure 9.1: File structure of the HMOranges source code.

9.1.2 Variables

As already mentioned the DEM code had to be stripped out. Once this was completed and results from the code verified and validated the heat and mass transfer methods could be added. This involved adding the following set of new variables to the particle structure:

- T_d
- W_V
- T_B
- L_V
- C_L
- P_atm
- R_bar
- R
- W_G
- theta_1
- theta_2
- Y_G
- Pr_G
- Sc_G
- f_2
- P_G
- T_G
- H_deltaT
- m_d
- Reynolds_d
- initial_mass
- rho_G

In addition to this code was added to the `createNormalDistVelocities` function that initialises the particles so the extra required variables are set to the required initial conditions. Finally, the equations to solve the heat and mass transfer were added to the iterate particle kernel. This included adding a stop condition for updating the particle properties. Namely: position, speed, temperature and mass. When the current particle mass reaches 1% of its starting mass, all properties for the particle are set to zero. At each further timestep the value of the particle properties is logged as zero. This makes later data analysis easier as particles are identified by a row of data in the text file. So the number of rows has to remain the same. This is especially important for utilising Paraview which identifies particles by row.

9.1.3 Heat and Mass Transfer Model

The heat and mass transfer model as outlined in Section 4 was implemented slightly differently to Python. The OpenCL code uses a single kernel to iterate the particles and this contains the functionality to evaluate the model.

9.2 Verification and Validation

Similar to the Python code the OpenCl code must undergo verification and validation tests to ensure the equations are being solved correctly.

For the verification tests in Sections 9.2.2 and 9.2.3 the settings used are as per Table 7.1.

9.2.1 Uncoupled Heat Transfer

Figure 9.2 shows the uncoupled heat transfer for a range of timesteps with the numerical solution converging to the analytic solution with decreasing timestep. As an implicit method has been used the numerical solution under predicts the analytic solution and this can also be clearly seen.

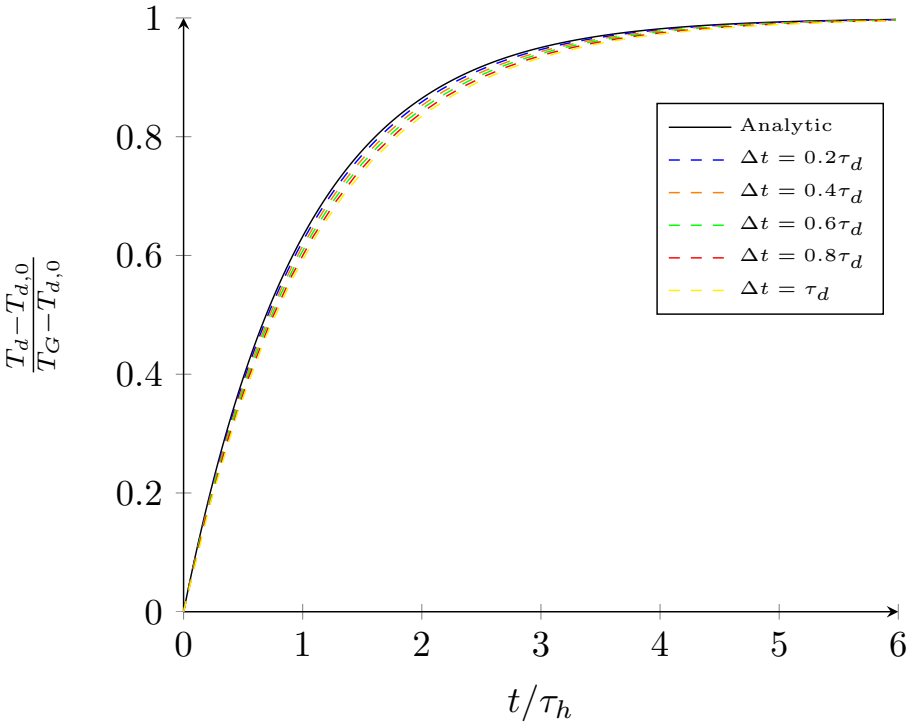


Figure 9.2: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$.

9.2.2 Uncoupled Mass Transfer

For the mass transfer non-dimensionalised droplet diameter squared has been plotted in Figure 9.3. Like the uncoupled heat transfer plot it can be seen the numerical solution converges to the analytic solution for decreasing timestep size.

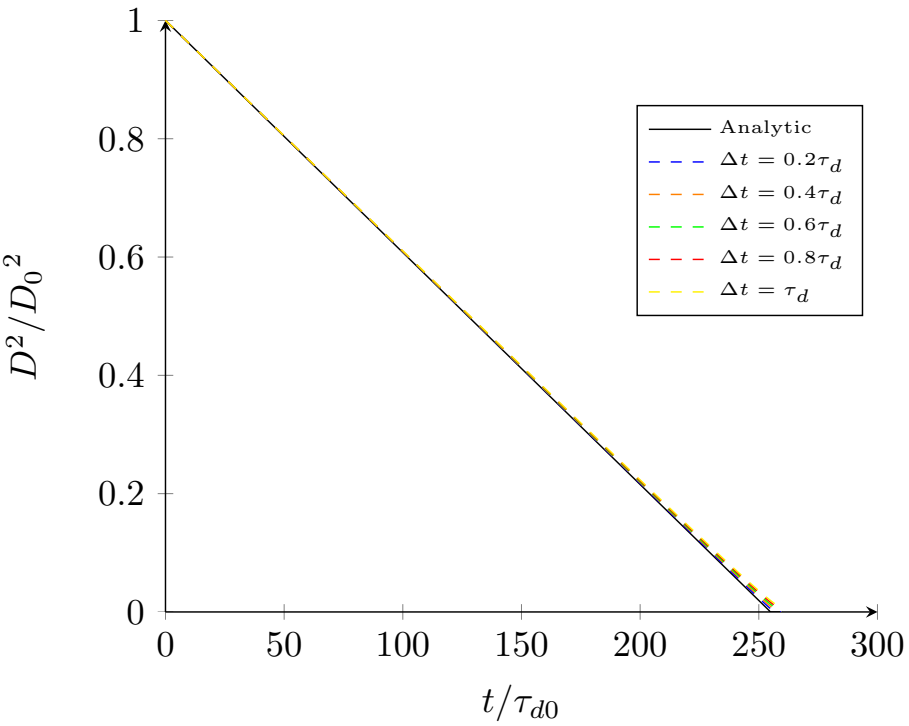
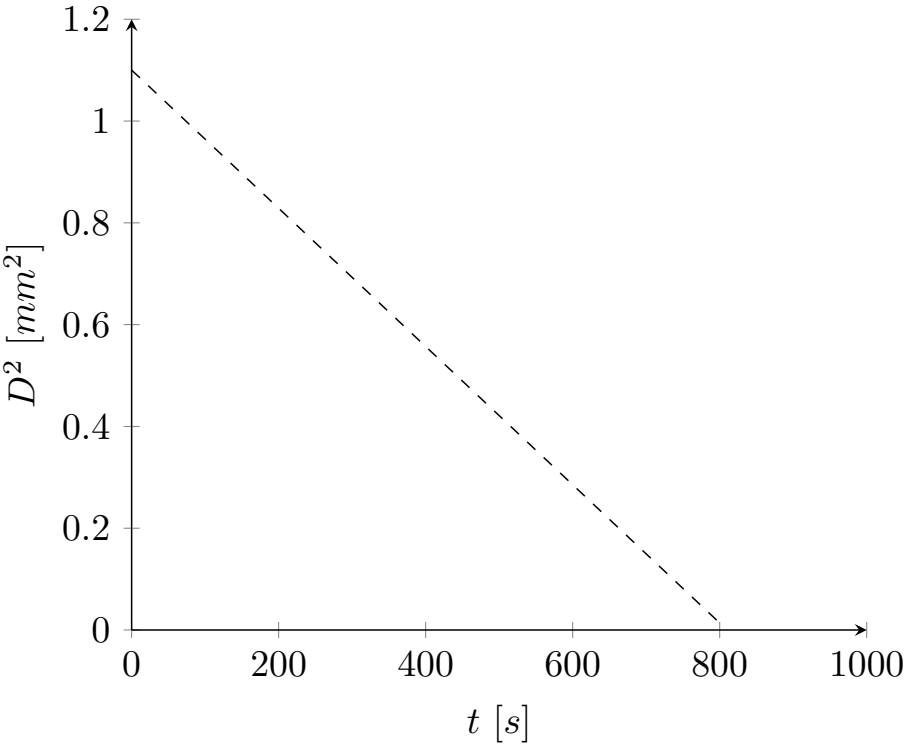


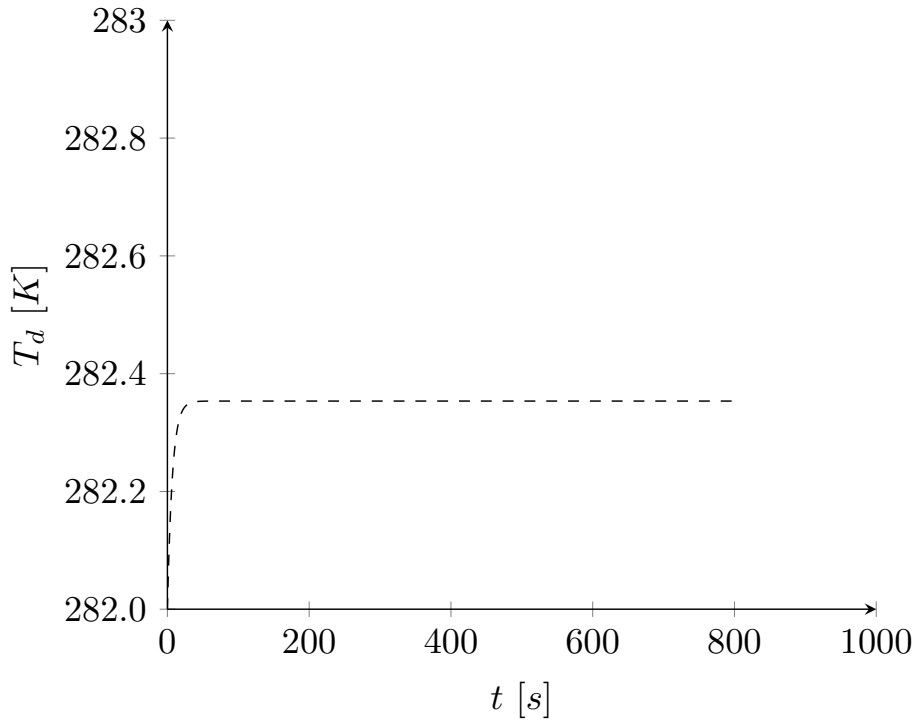
Figure 9.3: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$.

9.2.3 Coupled Heat and Mass Transfer

The coupled heat and mass results are plotted in Figure 9.4a and Figure 9.4b. The settings used can be found in Tables 7.1, 7.2 and 7.3. As with Section 7.5 these show good agreement with the results in the literature.



(a)



(b)

Figure 9.4: Droplet diameter squared (a) and droplet temperature (b) temporal evolution for a water droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$, $Y_G = 0$ and $\Delta t = \tau_{d0}/64$.

As with Section 7.5 additional results for hexane and decane were produced. These results are included in the Section E of the Appendix. The results presented there are in agreement with those from the Python code.

10 Results

10.1 Probability Density Functions of the Temporal Evolution of Temperature and Mass

The code is setup so the initial temperature and mass is the same for all the droplets. In the case of no convective heat transfer it would be expected that all droplets evaporate at the same rate and have the same increase in temperature. However, placing the droplets in a Taylor-Green vortex means the droplets will experience different fluid velocities. Therefore there will be a range of convective heat transfer across the droplets.

This effect can be observed by plotting how the probability density function (PDF) evolves over time. The spread of droplet temperature can be plotted by using the deviation $T_d{}'$:

$$T_d{}' = T_d - \bar{T_d} \tag{10.1}$$

This process can be used at each timestep to produce a 3D PDF to show how the deviation of droplet temperatures changes.

Parameter	Setting
Droplet Properties	
Molecular Weight of Vapour Phase W_V	18.015 $kg/kg\ mol$
Boiling Temperature T_B	373.15 K
Density of Liquid Phase ρ_L	997 kg/m^3
Specific Heat Capacity of Liquid Phase C_L	4184 $J/(kg\ K)$
Initial Temperature T_{d0}	282 K
Initial Diameter D_0	$\sqrt{1.1}\ mm$
Gas Properties	
Molecular Weight of Phase W_C	28.97 $kg/(kg\ mol)$
Temperature T_G	298 K
Density ρ_G	1.184 $kg\ m^{-3}$
Specific Heat for Constant Pressure $C_{P,G}$	1007 $J\ kg\ K$
Free Stream Vapour Mass Fraction Y_G	0
Atmospheric Properties	
Pressure P_{atm}	101325 Pa
Fluid Conditions	
Flow Magnitude A	0.05
Vortex Frequency a	3
Physical Constants	
Gravity g	0 ms^{-1}
Gas Constant $\bar{R}$	8314.5 $J/(K(kg\ mol))$
Universal Gas Constant R	287 $J/(kg\ K)$
Simulation Settings	
Correction factor f_2	1
Driving Potential $H_{\Delta T}$	0
Number of Droplets	10000
Simulation Length	$200\tau_{d,0}$
Timestep Size Δt	$\tau_{d,0}/8$

Table 10.1: Simulation settings used for creating the heat and mass PDFs.

The simulation was run using the settings in Table 10.1. Note there is no gravitational acceleration, so only the effects of the flow are considered. Including a weight force would

induce an effect known as enhanced settling. All further necessary values are calculated as per Sections 4.4.1 and 4.4.2. The corresponding PDF for the droplet temperature evolution is shown in Figure 10.1.

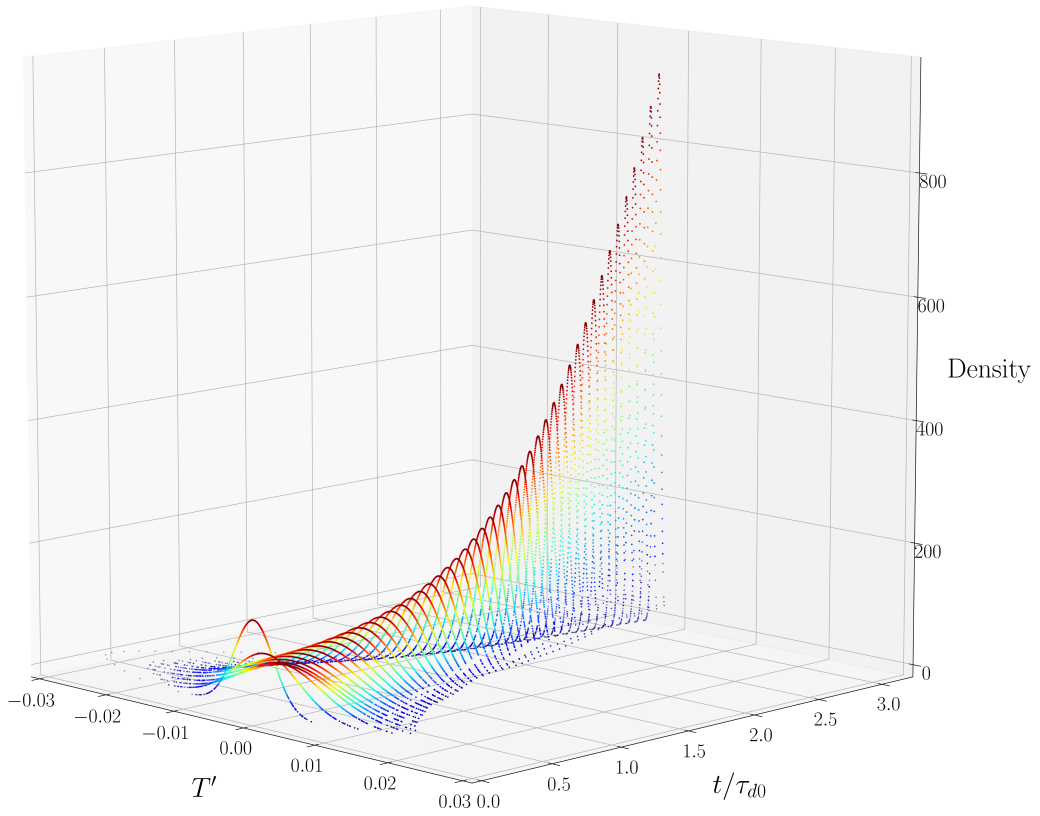


Figure 10.1: PDFs of the droplet temperature deviation for the first 50 timesteps of the simulation.

The initial conditions are not shown in this plot but would be a Dirac delta function centered around zero on the x-axis with a density of 10000. For the next 5 timesteps the deviation of droplet temperature increases as the droplets spread out into the fluid. The deviation then begins to decrease as the mean droplet temperature reaches the steady state temperature.

A similar PDF can be plotted for droplet diameter, much like the temperature deviation the droplet diameter normalised with initial diameter will deviate as the droplets evaporate at different rates. Furthermore, unlike the Figure 10.1 the mean value of the normalised D^2 PDF decreases as a result of the droplets losing mass. This is is shown in Figure 10.2.

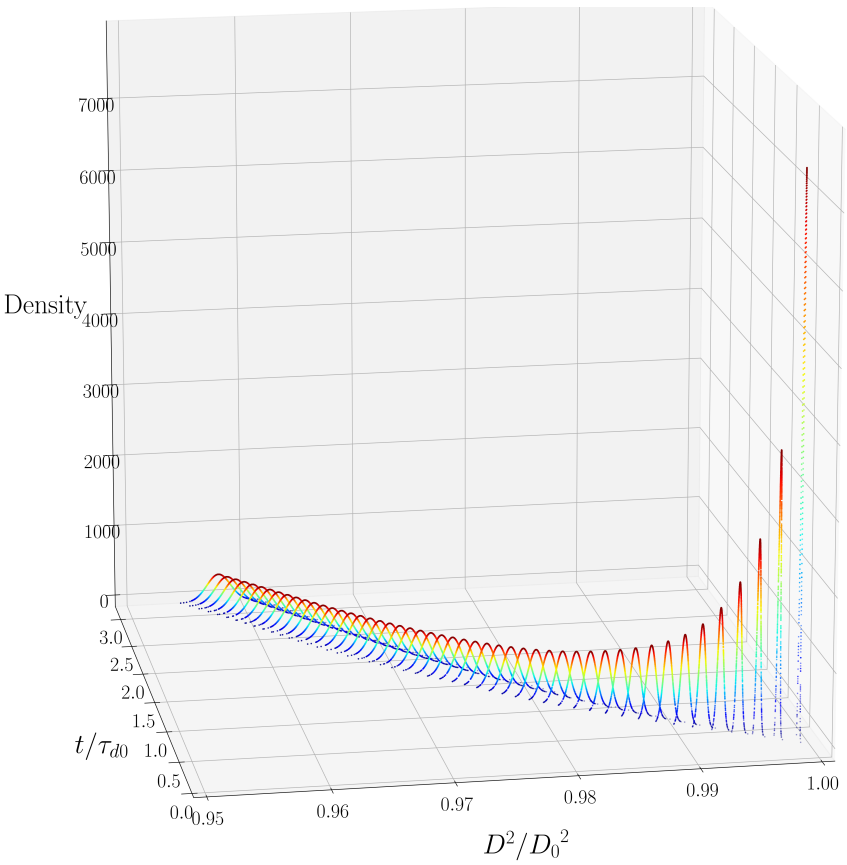
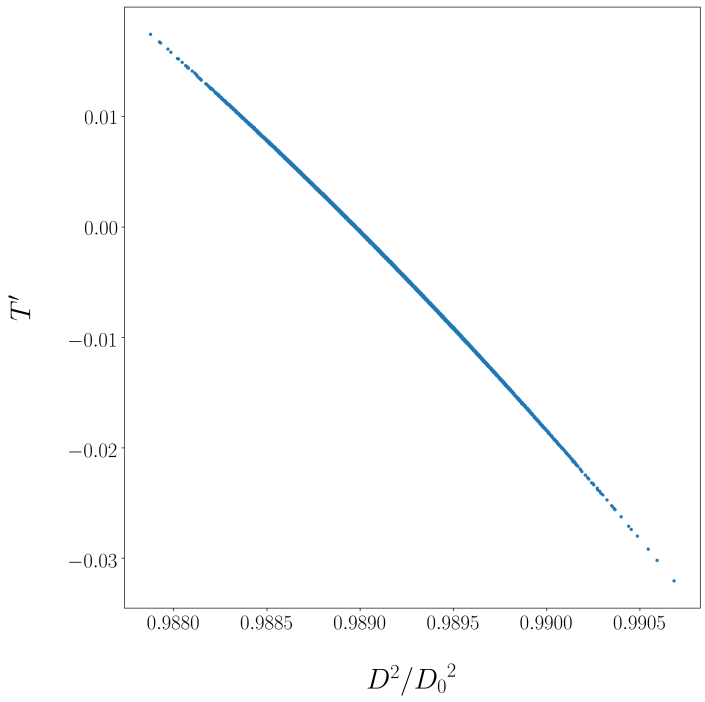
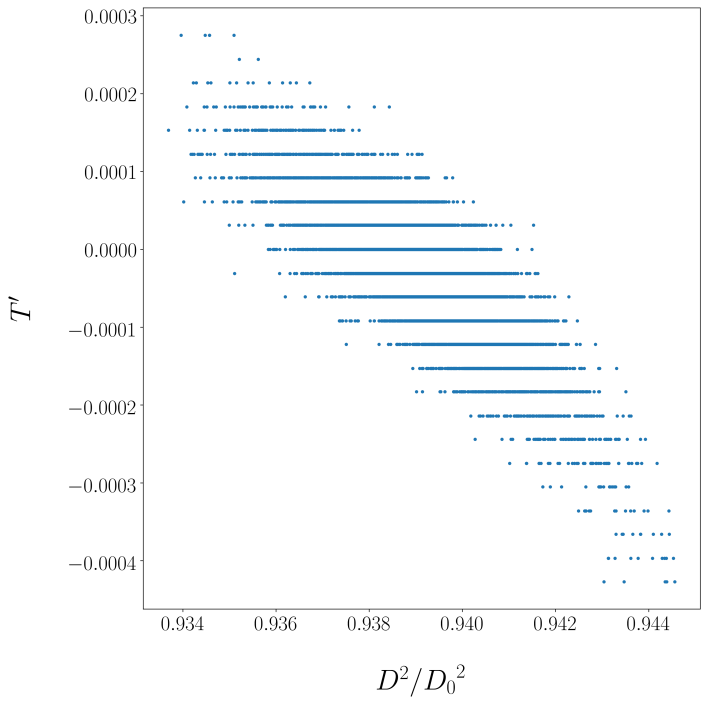


Figure 10.2: PDFs of the droplet $D^2/{D_0}^2$ for the first 50 timesteps of the simulation.

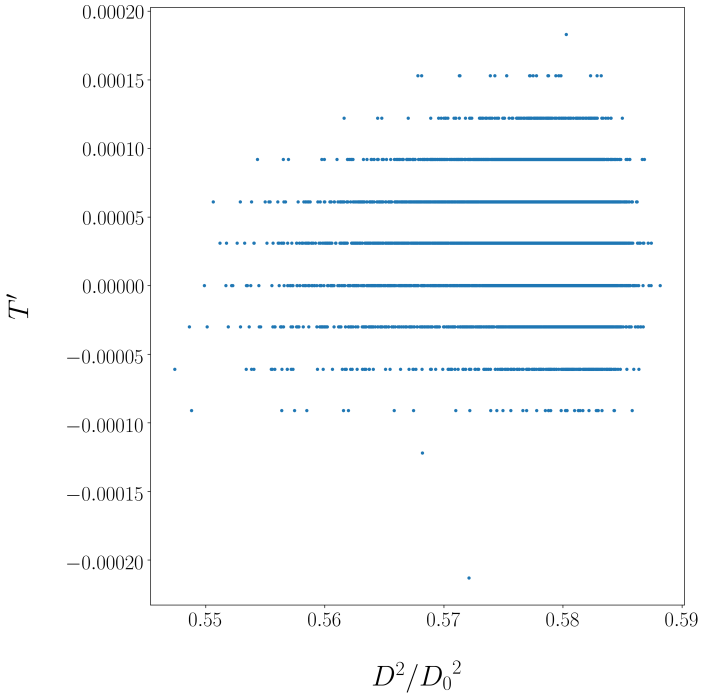
The relation between droplet mass deviation and droplet temperature deviation is shown in Figure 10.3. For Figure 10.3(a) It can be seen that near the start of the simulation the relation between $T_d{}'$ and $D^2/{D_0}^2$ is almost proportional . However, further into the simulation the deviation increases as can be seen in Figure 10.3(b). Although once the droplets reach an equilibrium temperature the spread of $T_d{}'$ while the deviation in $D^2/{D_0}^2$ has continued to increase as in Figure 10.3(c).


0.01
0.00
−0.01
−0.02
−0.03
T'
0.9880
0.9885
0.9890
0.9895
0.9900
0.9905
$D^2/{D_0}^2$

(a)


0.0003
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
T'
0.934
0.936
0.938
0.940
0.942
0.944
$D^2/{D_0}^2$

(b)



(c)

Figure 10.3: $D^2/{D_0}^2$ against ${T_d}'$ at $1\tau_{d,0}$ (a), $10\tau_{d,0}$ (b) and $100\tau_{d,0}$ (c) into the simulation.

10.2 Effect of Stokes Number on Heat Transfer

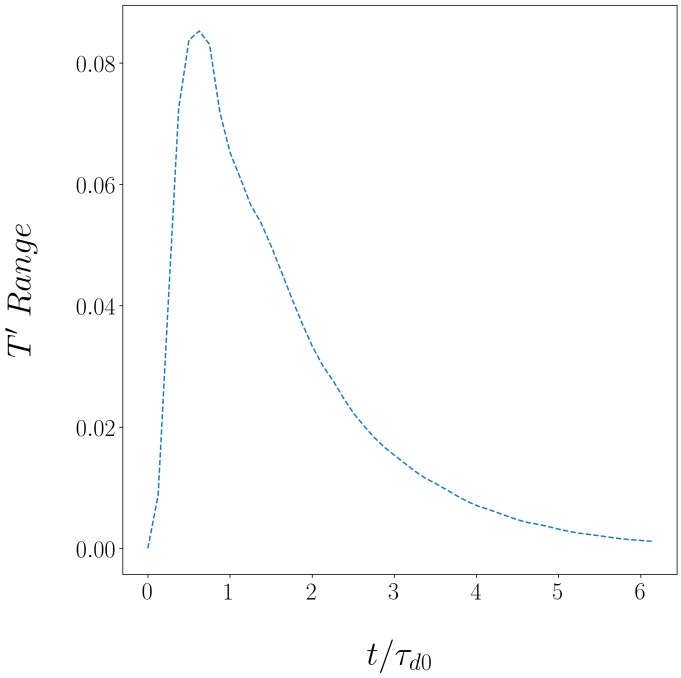
The Stokes number will have some bearing on the heat and mass transfer of the droplets. This is because this will change how much the droplets follow the gas flow and therefore how much convective heat transfer there is.

For the simulation the settings in Table 10.1 were used. However, instead of calculating μ_G using T_R suitable values were chosen to give the required Stokes numbers. These values are given in Table 10.2.

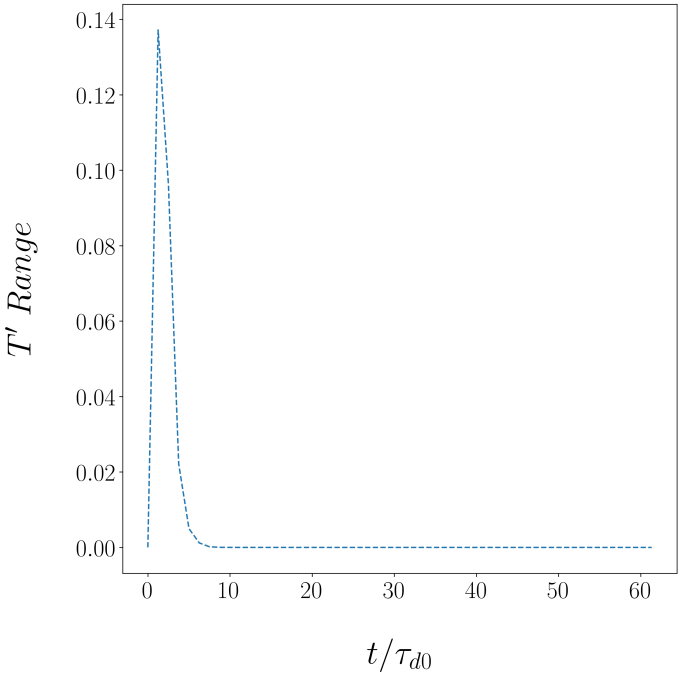
Kinematic Viscosity	**Stokes Number**
2.2804×10^{-5}	0.1
2.2804×10^{-6}	1.0
2.2804×10^{-7}	10.0

Table 10.2: Settings used to get different Stokes numbers.

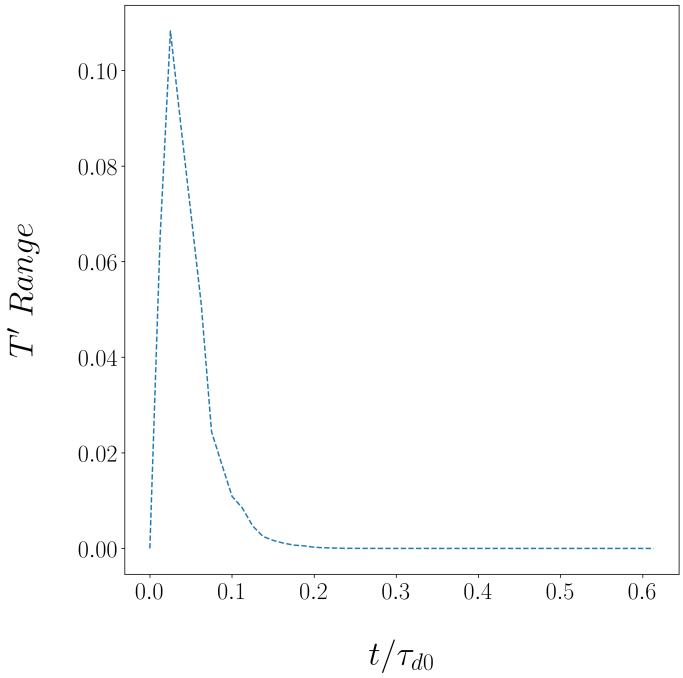
As mentioned previously the effect of the flow is to increase the deviation of droplet temperature and normalised droplet diameter. To compare results for different Stokes numbers the range between the maximum and minimum deviation for each timestep have been plotted in 10.4.


0.08
0.06
0.04
0.02
0.00
T' Range
0
1
2
3
4
5
6
t/τ_{d0}

(a)


0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
T' Range
0
10
20
30
40
50
60
t/τ_{d0}

(b)



(c)

Figure 10.4: T'' range for Stokes numbers of 0.1 (a), 1.0 (b) and 10 (c)

Increasing the Stokes number has the effect of decreasing the range between the minimum and maximum T'. Although this appears less pronounced for Stokes numbers between 0.1 and. This effect should be expected as for increasing Stokes numbers the less effect the fluid flow has on the droplet motion. Therefore, the difference in convective heat transfer between droplets will be less.

11 Conclusion

11.1 Summary of Results

An implementation of a common heat and mass transfer model has been produced in Python. Various numerical methods were evaluated in Python and a suitable method was found. The existing OpenCL code was modified to remove the DEM model. This code was successfully verified and validated. In addition, bugs in the code where found and these were fixed.

The heat and mass model was then implemented into the OpenCl code and this was also verified and validated. The code was then used to plot heat and mass transfer PDFs to demonstrate the effects of the flow on the convective heat transfer process. It was further found increasing the Stokes number had the effect of decreasing the range in convective heat transfer experienced by the droplets.

Therefore, this project has met the objectives detailed in Section 1.

11.2 Future Work

11.2.1 Additional Models and Droplet Species

Firstly, only one model has been implemented. The mass analogy models would make a good starting point for adding further models as this would require minimal alteration to the existing code. This could be taken further in making the code modular enough to support a user being able to easily choose a given model for running a simulation. Further to this a wider range of droplet species could be tested, a challenging test case would for example be common rocket engine propellants.

11.2.2 Re-ordering of Code to Simplify Running Simulations

Currently the code must be re-built every time the simulation parameters are changed. A more elegant approach would be to store simulation parameters in a text file which is loaded at runtime. This makes keeping track of simulation parameters easier (they are currently spread out across at least three source code files) and means the executable does not have to be re-built every time the parameters are changed.

11.2.3 Quantitative Analysis and Optimisation of the Code

The code has a whole provides an opportunity for a study on optimisation. One limiting factor is data logging is exceptionally time consuming when compared to running a simulation without data logging. Steps such as finding a faster alternative to C's `fprintf` and perhaps writing in binary instead of to a text file may provide the required performance improvement. This could be taken further, the results from the simulation could be stored in memory and the statistics calculated using the GPU. The benefit of this is twofold. The processing of data for very large datasets will be quicker than with using Python. Secondly, a reduced amount of information has to be written reducing the time it takes to log data.

Appendix

A Method Statement

As this project involves no practical experimentation and only concerns desk-based research there is not a need to submit a risk assessment.

B Derivations

B.1 Velocity of a Particle under Weight and Stokes Drag Forces

The derived ODE from Section 8.1 is:

$$\frac{du}{dt} = g + \frac{1}{\tau_d}(u_f - u) \tag{B.1a}$$

$$\frac{du}{dt} + \frac{u}{\tau_d} = g + \frac{u_f}{\tau_d} \tag{B.1b}$$

This is an equation of the form $du/dt + Pu = Q$, where P and Q are functions of t. Which can be solved with an integrating factor.

Integrating factor:

$$IF = e^{\int P dt} \tag{B.2a}$$

$$= e^{\int 1/\tau_d dt} \tag{B.2b}$$

$$= e^{t/\tau_d} \tag{B.2c}$$

Therefore:

$$e^{t/\tau_d}\frac{du}{dt} + e^{t/\tau_d}\frac{u}{\tau_d} = e^{t/\tau_d}\left(g + \frac{u_f}{\tau_d}\right) \tag{B.3a}$$

$$\frac{d}{dt}\left(e^{t/\tau_d}u\right) = e^{-t/\tau_d}\left(g + \frac{u_f}{\tau_d}\right) \tag{B.3b}$$

$$e^{t/\tau_d}u = \int e^{t/\tau_d}\left(g - \frac{u_f}{\tau_d}\right) dt \tag{B.3c}$$

$$e^{t/\tau_d}u = \tau_d e^{t/\tau_d}\left(g + \frac{u_f}{\tau_d}\right) + C \tag{B.3d}$$

Solve for the integration constant C with the boundary condition $u = 0$ when $t = 0$:

$$0 = \tau_d\left(g + \frac{u_f}{\tau_d}\right) + C \tag{B.4a}$$

$$C = -\tau_d\left(g + \frac{u_f}{\tau_d}\right) \tag{B.4b}$$

So that:

$$e^{t/\tau_d}u = \tau_d e^{t/\tau_d}\left(g + \frac{u_f}{\tau_d}\right) - \tau_d\left(g + \frac{u_f}{\tau_d}\right) \tag{B.5a}$$

$$u = \tau_d\left(g + \frac{u_f}{\tau_d}\right) - \tau_d\left(g + \frac{u_f}{\tau_d}\right)e^{-t/\tau_d} \tag{B.5b}$$

$$u = \tau_d\left(g + \frac{u_f}{\tau_d}\right)(1 - e^{-t/\tau_d}) \tag{B.5c}$$

B.2 Backward Euler Method for Temperature ODE

Rearrangement of the backward Euler method for the temperature ODE:

$$T_{d_{n+1}} = T_{d_n} + \Delta t \left[\frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) (T_G - T_{d_{n+1}}) + \left(\frac{L_V}{C_L} \right) \frac{\dot{m}_d}{m_d} - H_{\Delta T} \right] \tag{B.6a}$$

$$\frac{T_{d_{n+1}}}{\Delta t} = \frac{T_{d_n}}{\Delta t} + \left[T_G \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) - T_{d_{n+1}} \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) + \left(\frac{L_V}{C_L} \right) \frac{\dot{m}_d}{m_d} - H_{\Delta T} \right] \tag{B.6b}$$

$$\frac{T_{d_{n+1}}}{\Delta t} + T_{d_{n+1}} \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) = \frac{T_{d_n}}{\Delta t} + \left[T_G \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) + \left(\frac{L_V}{C_L} \right) \frac{\dot{m}_d}{m_d} - H_{\Delta T} \right] \tag{B.6c}$$

$$T_{d_{n+1}} \left[\frac{1}{\Delta t} + \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) \right] = \frac{T_{d_n}}{\Delta t} + \left[T_G \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) + \left(\frac{L_V}{C_L} \right) \frac{\dot{m}_d}{m_d} - H_{\Delta T} \right] \tag{B.6d}$$

$$T_{d_{n+1}} = \frac{\frac{T_{d_n}}{\Delta t} + \left[T_G \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) + \left(\frac{L_V}{C_L} \right) \frac{\dot{m}_d}{m_d} - H_{\Delta T} \right]}{\left(\frac{1}{\Delta t} + \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) \right)} \tag{B.6e}$$

$$T_{d_{n+1}} = \frac{T_{d_n} + \Delta t \left[T_G \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) + \left(\frac{L_V}{C_L} \right) \frac{\dot{m}_d}{m_d} - H_{\Delta T} \right]}{\left(1 + \Delta t \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) \right)} \tag{B.6f}$$

B.3 Backward Euler Method for Mass ODE

Rearrangement of the backward Euler method for the mass ODE:

$$m_{d_{n+1}} = m_{d_n} + \Delta t \left[-\frac{Sh}{3 Sc_G} \left(\frac{m_{d_{n+1}}}{\tau_d} \right) H_M \right] \tag{B.7a}$$

$$\frac{m_{d_{n+1}}}{\Delta t} = \frac{m_{d_n}}{\Delta t} + \left[-\frac{Sh}{3 Sc_G} \left(\frac{m_{d_{n+1}}}{\tau_d} \right) H_M \right] \tag{B.7b}$$

$$\frac{m_{d_{n+1}}}{\Delta t} + \frac{Sh}{3 Sc_G} \left(\frac{m_{d_{n+1}}}{\tau_d} \right) H_M = \frac{m_{d_n}}{\Delta t} \tag{B.7c}$$

$$m_{d_{n+1}} \left[\frac{1}{\Delta t} + \frac{Sh}{3 Sc_G} \left(\frac{H_M}{\tau_d} \right) \right] = \frac{m_{d_n}}{\Delta t} \tag{B.7d}$$

$$m_{d_{n+1}} = \frac{\frac{m_{d_n}}{\Delta t}}{\frac{1}{\Delta t} + \frac{Sh}{3 Sc_G} \left(\frac{H_M}{\tau_d} \right)} \tag{B.7e}$$

$$m_{d_{n+1}} = \frac{m_{d_n}}{1 + \Delta t \frac{Sh}{3 Sc_G} \left(\frac{H_M}{\tau_d} \right)} \tag{B.7f}$$

B.4 Analytic Solution for Uncoupled Heat Transfer

The uncoupled heat transfer ODE is:

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) (T_G - T_d) \tag{B.8}$$

To make the solution steps clearer, define the constants:

$$A = \frac{f_2 Nu}{3 Pr_G} \left(\frac{\theta_1}{\tau_d} \right) \tag{B.9a}$$

$$B = T_G \tag{B.9b}$$

The ODE can then be solved for $T_{d_{n+1}}$ for a given increment in time Δt from a state where $T_d = T_{d_n}$:

$$\frac{dT_d}{dt} = A(B - T_d) \tag{B.10a}$$

$$\int_{T_{d_n}}^{T_{d_{n+1}}} \frac{dT_d}{(B - T_d)} = \int_0^{\Delta t} A \; dt \tag{B.10b}$$

$$[\ln(B - T_d)]_{T_{d_n}}^{T_{d_{n+1}}} = [A \; t]_0^{\Delta t} \tag{B.10c}$$

$$-\ln(B - T_{d_{n+1}}) + \ln(B - T_{d_n}) = A\Delta t \tag{B.10d}$$

$$\ln\left(\frac{B - T_{d_{n+1}}}{B - T_{d_n}}\right) = -A\Delta t \tag{B.10e}$$

$$\frac{B - T_{d_{n+1}}}{B - T_{d_n}} = e^{-A\Delta t} \tag{B.10f}$$

$$B - T_{d_{n+1}} = (B - T_{d_n})e^{-A\Delta t} \tag{B.10g}$$

$$T_{d_{n+1}} = B - (B - T_{d_n})e^{-A\Delta t} \tag{B.10h}$$

Hence, the final analytic solution is:

$$T_{d_{n+1}} = T_G - (T_G - T_{d_n})e^{-\left(\frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)\right)\Delta t} \tag{B.11}$$

B.5 Analytic Solution for Uncoupled Mass Transfer

The uncoupled mass transfer ODE is:

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\frac{m_d}{\tau_d}H_M \tag{B.12}$$

B.5.1 Solution for Diameter

Solving equation B.12 for diameter requires a relation between D and m_d:

$$m_d = \rho_d V_d \tag{B.13a}$$

$$m_d = \rho_d \left(\frac{4}{3}\right)\pi\left(\frac{D}{2}\right)^3 \tag{B.13b}$$

$$m_d = \frac{\rho_d \pi D^3}{6} \tag{B.13c}$$

So:

$$\frac{dm_d}{dt} = \frac{\rho_d \pi D^2}{2}\frac{dD}{dt} \tag{B.14}$$

These can be substituted into equation B.12, along with $\tau_d = \rho_d D^2/(18\mu_g)$:

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\frac{m_d}{\tau_d}H_M \tag{B.15a}$$

$$\frac{\rho_d \pi D^2}{2}\frac{dD}{dt} = -\frac{Sh}{3Sc_G}\frac{18\rho_d \pi D^3 \mu_g}{6\rho_d D^2}H_M \tag{B.15b}$$

$$\frac{1}{2}\left[D_{n+1}^2 - D_n^2\right] = -\frac{2Sh}{Sc_G \rho_d}\mu_g H_M \; \Delta t \tag{B.15c}$$

$$D_{n+1}^2 = D_n^2 - \frac{4Sh}{Sc_G \rho_d}\mu_g H_M \; \Delta t \tag{B.15d}$$

The analytic solution for the uncoupled mass transfer ODE for D^2:

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\frac{m_d}{\tau_d}H_M \tag{B.16a}$$

$$\frac{\rho_d \pi D^2}{2}\frac{dD}{dt} = -\frac{Sh}{3Sc_G}\frac{18\rho_d \pi D^3 \mu_g}{6\rho_d D^2}H_M \tag{B.16b}$$

$$D\frac{dD}{dt} = -\frac{2Sh}{Sc_G\rho_d}\mu_g H_M \tag{B.16c}$$

$$\int_{D_n^2}^{D_{n+1}^2} D\ dD = -\frac{2Sh}{Sc_G\rho_d}\mu_g H_M \int_0^{\Delta t}\ dt \tag{B.16d}$$

$$\left[\frac{D^2}{2}\right]_{D_n^2}^{D_{n+1}^2} = -\frac{2Sh}{Sc_G\rho_d}\mu_g H_M \left[t\right]_0^{\Delta t} \tag{B.16e}$$

$$\frac{1}{2}\left[D_{n+1}^2 - D_n^2\right] = -\frac{2Sh}{Sc_G\rho_d}\mu_g H_M\ \Delta t \tag{B.16f}$$

$$D_{n+1}^2 = D_n^2 - \frac{4Sh}{Sc_G\rho_d}\mu_g H_M\ \Delta t \tag{B.16g}$$

B.5.2 Solution for Mass

The result from Section 7.3 can be solved for mass:

$$\frac{dm_d}{dt} = -\frac{Sh}{Sc_G}\mu_G H_M \left(\frac{6}{\rho_d}\right)^{1/3}\pi^{2/3} m_d^{1/3} \tag{B.17a}$$

$$\int_{m_{dn}}^{m_{dn+1}} \frac{dm_d}{(m_d)^{1/3}} = -\frac{Sh}{Sc_G}\mu_G H_M \left(\frac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\int_0^{\Delta t}\ dt \tag{B.17b}$$

$$\left[\frac{3}{2}(m_d)^{2/3}\right]_{m_{dn}}^{m_{dn+1}} = -\frac{Sh}{Sc_G}\mu_G H_M \left(\frac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\left[t\right]_0^{\Delta t} \tag{B.17c}$$

$$\frac{3}{2}\left[(m_{dn+1})^{2/3} - (m_{dn})^{2/3}\right] = -\frac{Sh}{Sc_G}\mu_G H_M \left(\frac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\ \Delta t \tag{B.17d}$$

$$(m_{dn+1})^{2/3} = (m_{dn})^{2/3} - \frac{2Sh}{3Sc_G}\mu_G H_M \left(\frac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\ \Delta t \tag{B.17e}$$

B.6 Terminal Velocity

The particle reaches a terminal velocity when the weight force balances the drag force:

$$F_g = F_d \tag{B.18}$$

Using the definitions for F_g and F_d from Section 4.3.3:

$$mg = \frac{m}{\tau_d}(u - v) \tag{B.19a}$$

$$g = \frac{1}{\tau_d}(u - v) \tag{B.19b}$$

$$v = u - g\tau_d \tag{B.19c}$$

$$v = u - g\frac{\rho_d D^2}{18\mu_G} \tag{B.19d}$$

C Physical Data

Note that T_R is the reference temperature used when evaluating a property. The physical data presented here is as per [2]. The original source of physical data is also referenced.

C.1 Air

Data originally from [32].

$$W_C = 28.97\ kg\ (kg\ mole)^{-1} \tag{C.1a}$$
$$\mu_C = 6.19 \times 10^{-6} + 4.604 \times 10^{-8} T_R - 1.051 \times 10^{-11} {T_R}^2\ kg\ m^{-1} s^{-1} \tag{C.1b}$$
$$\lambda_C = 3.227 \times 10^{-3} + 8.3894 \times 10^{-5} T_R - 1.9858 \times 10^{-8} {T_R}^2\ J\ m^{-1} s^{-1} K^{-1} \tag{C.1c}$$
$$Pr_C = 0.815 - 4.958 \times 10^{-4} T_R + 4.514 \times 10^{-7} {T_R}^2\ (T_R \leq 600\ K) \tag{C.1d}$$
$$Pr_C = 0.647 - 5.5 \times 10^{-5} T_R\ (T_R > 600\ K) \tag{C.1e}$$

The gas pressure is calculated from the reference temperature and the gas density using the ideal gas law. This requires knowing the gas density, the reference data for this is tabulated in Table C.1 and referenced from [33].

C.2 Water

Data originally from [32].

$$W_V = 18.015\ kg\ (kg\ mole)^{-1} \tag{C.2a}$$
$$T_B = 373.15\ K \tag{C.2b}$$
$$C_{p,V} = 8137 - 37.34 T_R + 0.07482 {T_R}^2 - 4.956 \times 10^{-5} {T_R}^3\ J\ kg^{-1} K^{-1} \tag{C.2c}$$
$$\mu_V = 4.07 \times 10^{-8} T_R - 3.077 \times 10^{-6}\ kg\ m^{-1} s^{-1} \tag{C.2d}$$
$$L_V = 2.257 \times 10^6 + 2.595 \times 10^3 (371.15 - T_R)\ J\ Kg^{-1} \tag{C.2e}$$
$$\rho_L = 997\ kg\ m^{-3} \tag{C.2f}$$
$$C_L = 4148\ J\ kg^{-1} K^{-1} \tag{C.2g}$$
$$\lambda_L = 0.6531\ J\ m^{-1} s^{-1} K^{-1} \tag{C.2h}$$

C.3 Hexane

Data originally from [34].

$$W_V = 86.178\ kg\ (kg\ mole)^{-1} \tag{C.3a}$$
$$T_B = 344.6\ K \tag{C.3b}$$
$$C_{p,V} = -51.31 + 6.767 T_R - 3.626 \times 10^{-3} {T_R}^2\ J\ kg^{-1} K^{-1} \tag{C.3c}$$
$$\mu_V = 5.592 \times 10^{-6} + 5.622 \times 10^{-9} T_R\ kg\ m^{-1} s^{-1} \tag{C.3d}$$
$$L_V = 5.14787 \times 10^5 (1 - T_R/512)^{0.3861}\ J\ Kg^{-1} \tag{C.3e}$$
$$\rho_L = 664\ kg\ m^{-3} \tag{C.3f}$$
$$C_L = 2302\ J\ kg^{-1} K^{-1} \tag{C.3g}$$
$$\lambda_L = 0.1046\ J\ m^{-1} s^{-1} K^{-1} \tag{C.3h}$$

Temperature T $(°C)$	Density ρ $(kg\ m^{-3})$	Specific Heat $C_{p,g}$ $(J\ kg\ K)$
-150	2.866	983
-100	2.038	966
-50	1.582	999
-40	1.514	1002
-30	1.451	1004
-20	1.394	1005
-10	1.341	1006
0	1.292	1006
5	1.269	1006
10	1.246	1006
15	1.225	1007
20	1.204	1007
25	1.184	1007
30	1.164	1007
35	1.145	1007
40	1.127	1007
45	1.109	1007
50	1.092	1007
60	1.059	1007
70	1.028	1007
80	0.9994	1008
90	0.9718	1008
100	0.9458	1009
120	0.8977	1011
140	0.8542	1013
160	0.8148	1016
180	0.7788	1019
200	0.7459	1023
250	0.6746	1033
300	0.6158	1044
350	0.5664	1056
400	0.5243	1069
450	0.4880	1081
500	0.4565	1093
600	0.4042	1115
700	0.3627	1135
800	0.3289	1153
900	0.3008	1169
1000	0.2772	1184
1500	0.1990	1234
2000	0.1553	1264

Table C.1: Air properties at 1 atm pressure.

C.4 Decane

Data originally from [35]. Where ($T^* = T_R/1000$).

$$W_V = 142\ kg\ (kg\ mole)^{-1} \tag{C.4a}$$

$$T_B = 447.7\ K \tag{C.4b}$$

$$C_{p,V} = 106.6 + 5765T^* - 1675T^{*2} + 473.1T^{*3}\ J\ kg^{-1}K^{-1} \quad \text{for } T^* \leq 0.8 \tag{C.4c}$$

$$C_{p,V} = 411.1 + 5460T^* - 2483T^{*2} + 422.9T^{*3}\ J\ kg^{-1}K^{-1} \quad \text{for } T^* > 0.8 \tag{C.4d}$$

$$\mu_V = 5.64 \times 10^{-6} + 1.75 \times 10^{-8}(T_R - 300)\ kg\ m^{-1}s^{-1} \tag{C.4e}$$

$$\lambda_V = 1.214 \times 10^{-2}(T/300)^{1.8}\ J\ m^{-1}s^{-1}K^{-1} \tag{C.4f}$$

$$\Gamma_V = 5.46 \times 10^{-6}(T/300)^{1.583}\ P^{-1}m^2\ s^{-1} \tag{C.4g}$$

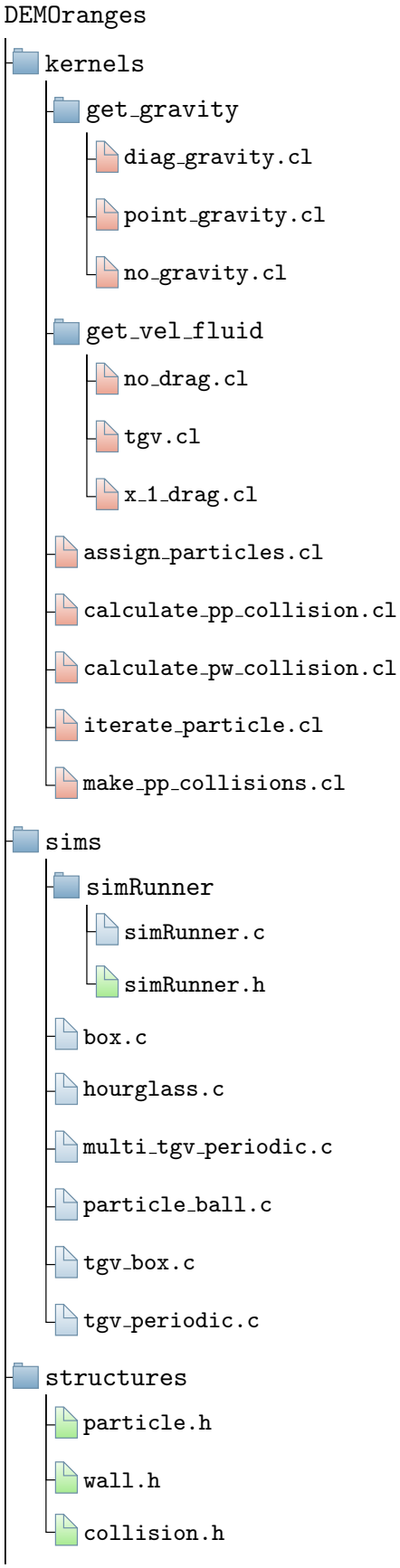
$$L_V = 3.958 \times 10^4(619 - T_R)^{0.38}\ J\ Kg^{-1} \tag{C.4h}$$

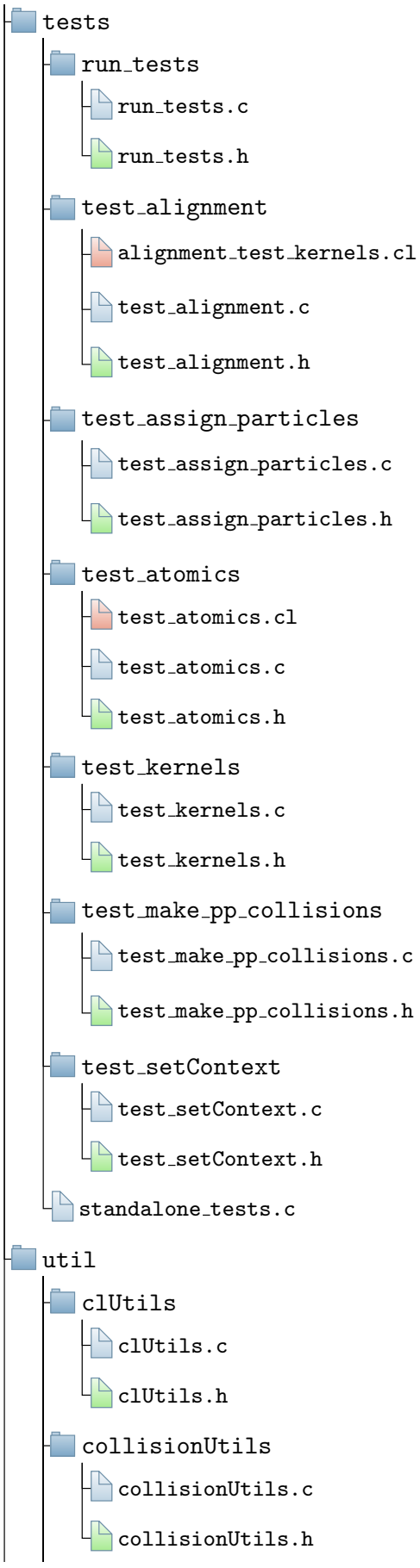
$$\rho_L = 642\ kg\ m^{-3} \tag{C.4i}$$

$$C_L = 2520.5\ J\ kg^{-1}K^{-1} \tag{C.4j}$$

$$\lambda_L = 0.1055\ J\ m^{-1}s^{-1}K^{-1} \tag{C.4k}$$

D Existing Source Code Structure




tests
run_tests
run_tests.c
run_tests.h
test_alignment
alignment_test_kernels.cl
test_alignment.c
test_alignment.h
test_assign_particles
test_assign_particles.c
test_assign_particles.h
test_atomics
test_atomics.cl
test_atomics.c
test_atomics.h
test_kernels
test_kernels.c
test_kernels.h
test_make_pp_collisions
test_make_pp_collisions.c
test_make_pp_collisions.h
test_setContext
test_setContext.c
test_setContext.h
standalone_tests.c
util
clUtils
clUtils.c
clUtils.h
collisionUtils
collisionUtils.c
collisionUtils.h

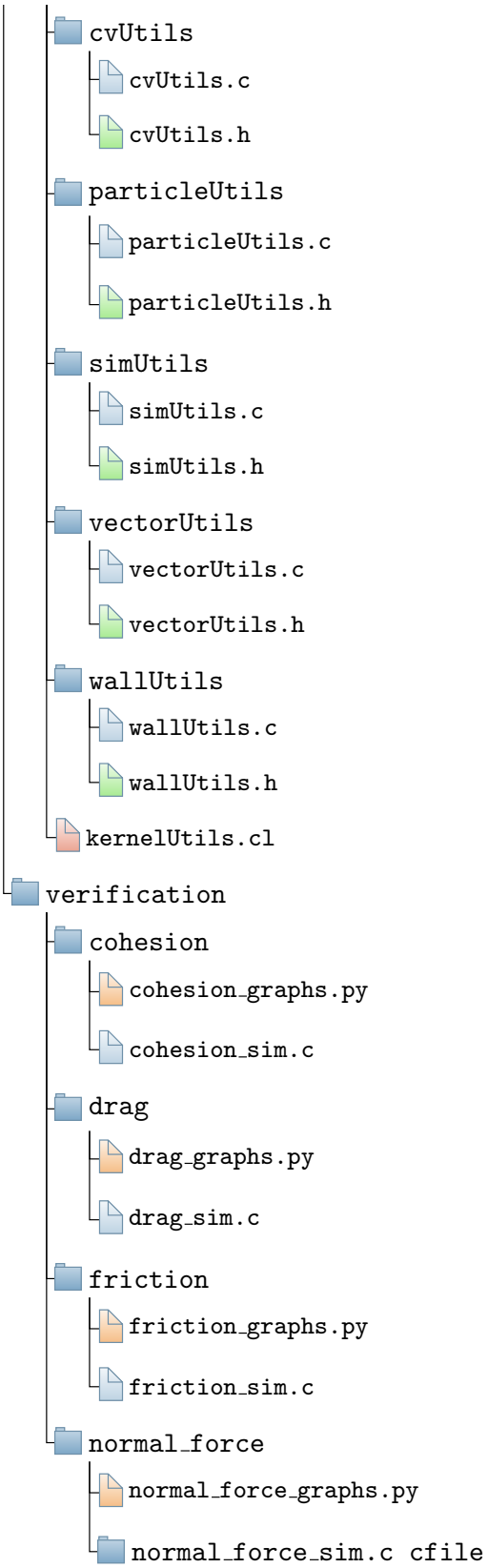
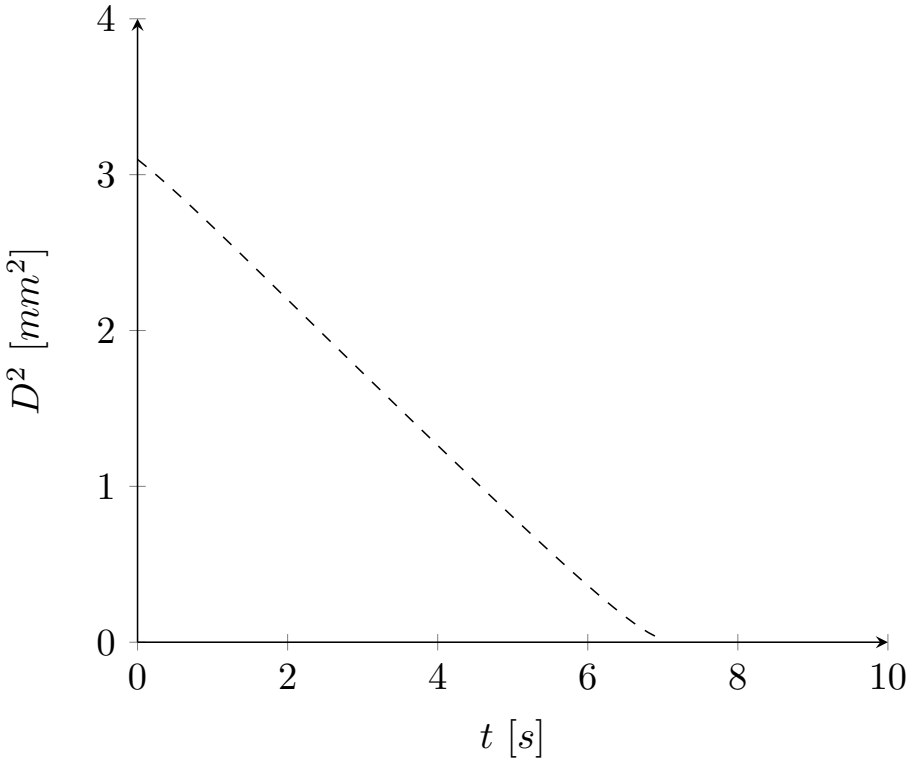


Figure D.1: File structure of the DEMOranges source code.

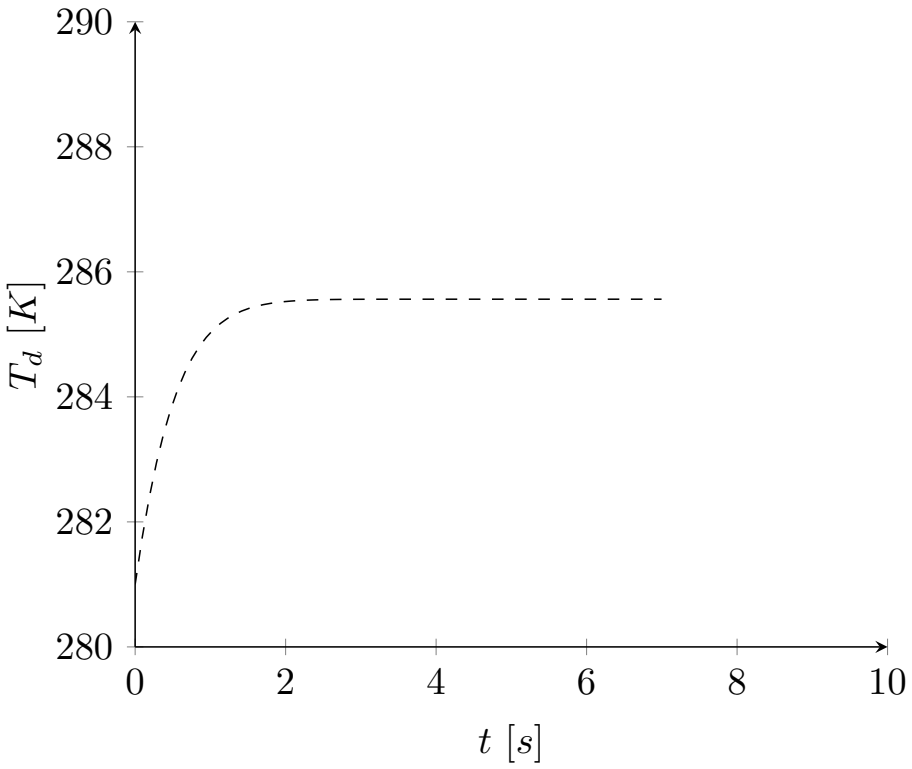
E Additional OpenCL Validation Results

Included in this section are additional validation results for the OpenCL code. This includes hexane (Figure E.1a and E.1b) and decane (E.2a and E.2b) droplets. The settings used can be found in Table 7.2 and Table 7.3 respectively.

E.1 Hexane Droplet



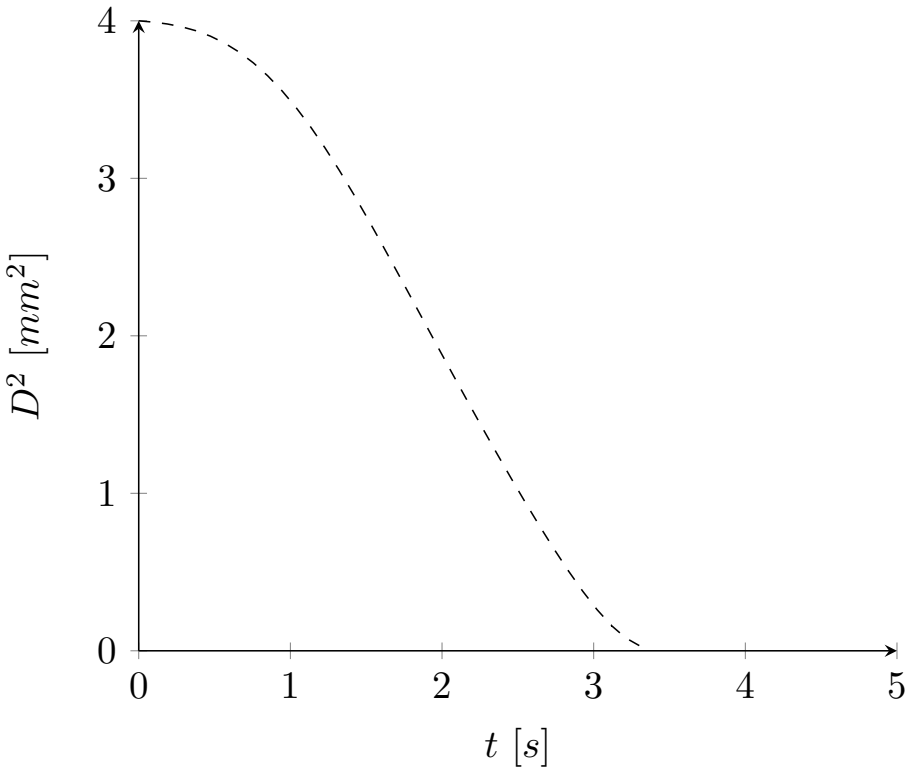
(a)



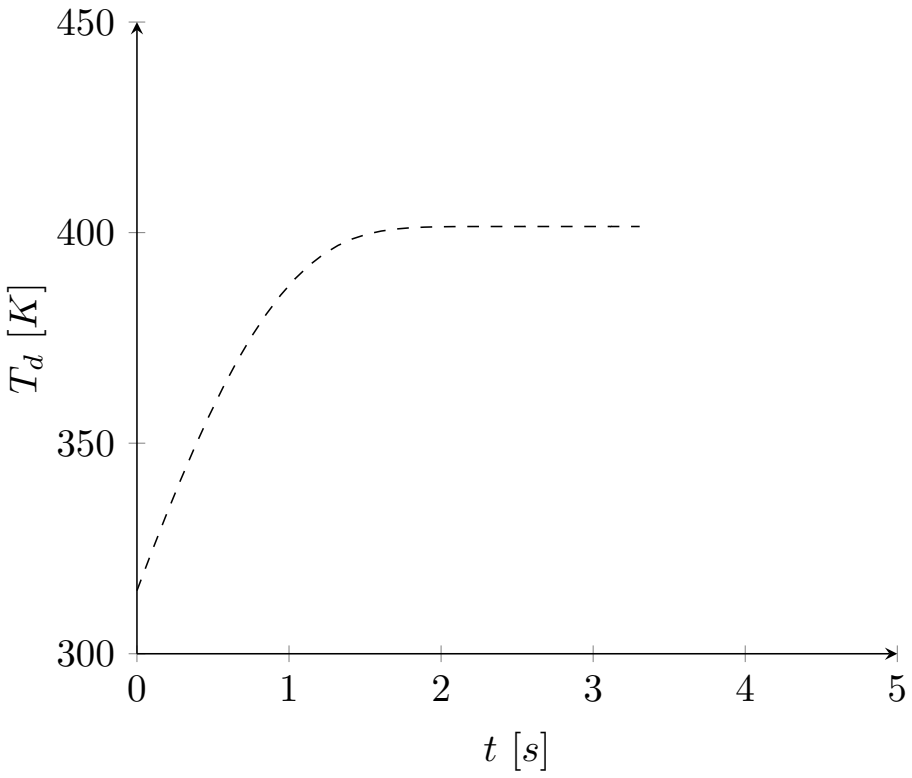
(b)

Figure E.1: Droplet diameter squared (a) and droplet temperature (b) temporal evolution for a hexane droplet sized $D_0 = 1.76mm$ with $Re_d = 110$, $T_{d_0} = 281K$, $T_G = 437K$ and $\Delta t = \tau_{d0}/64$.

E.2 Decane Droplet



(a)



(b)

Figure E.2: Droplet diameter squared (a) and droplet temperature (b) temporal evolution for a decane droplet sized $D_0 = 2.0mm$ with $Re_d = 17$, $T_{d_0} = 315K$, $T_G = 1000K$ and $\Delta t = \tau_{d0}/64$.

References

[1] E. Andrews, "GPU Enabled Analysis of Agglomeration in Large Particle Populations," 2018.

[2] R. S. Miller, K. Harstad, and J. Bellan, "Evaluation of equilibrium and non-equilibrium evaporation models for many-droplet gas-liquid flow simulations," *International Journal of Multiphase Flow*, vol. 24, pp. 1025–1055, 1998.

[3] E. K. Shashank, E. Knudsen, and H. Pitsch, "Spray Evaporation Model Sensitivities," *Annual Research Briefs of the CTR*, pp. 213–224, 2011.

[4] F. L. Sacomano Filho, G. C. Krieger Filho, J. A. van Oijen, A. Sadiki, and J. Janicka, "A novel strategy to accurately represent the carrier gas properties of droplets evaporating in a combustion environment," *International Journal of Heat and Mass Transfer*, 2019.

[5] C. Downing, "The evaporation of drops of pure liquids at elevated temperatures: Rates of evaporation and wet-bulb temperatures," *American Institute of Chemical Engineers*, vol. 12, pp. 760–766, 1966.

[6] H. Salman and M. Soteriou, "Lagrangian simulation of evaporating droplet sprays," *Physics of Fluids - PHYS FLUIDS*, vol. 16, pp. 4601–4622, 12 2004.

[7] D. Kolaitis and M. Founti, "A Comparative Study of Numerical Models for Eulerian–Lagrangian Simulations of Turbulent Evaporating Sprays," *International Journal of Heat and Fluid Flow*, vol. 27, pp. 424–435, 06 2006.

[8] S. Aggarwal, S. William, and A. Tong, "A Comparison of Vaporization Models in Spray Calculations," *AIAA Journal*, vol. 22, pp. 1448–1457, 09 1984.

[9] J. Sweet, D. Richter, and D. Thain, "GPU acceleration of Eulerian-Lagrangian particle-laden turbulent flow simulations," *International Journal of Multiphase Flow*, vol. 99, pp. 437–445, 11 2017.

[10] Q. Zhang, C. Zhu, and M. Zhu, "Three-Dimensional Numerical Simulation of Droplet Evaporation Using the Lattice Boltzmann Method Based on GPU-CUDA Accelerated Algorithm," *Communications in Computational Physics*, vol. 23, pp. 1150–1166, 01 2018.

[11] C. Obrecht, B. Tourancheau, and F. Kuznik, "Performance Evaluation of an OpenCL Implementation of the Lattice Boltzmann Method on the Intel Xeon Phi," *Parallel Processing Letters*, vol. 25, 09 2015.

[12] W. Verdier, P. Kestener, and A. Cartalade, "Performance portability of lattice boltzmann methods for two-phase flows with phase change," unpublished results.

[13] W. Nazaroff and L. Alvarez-Cohen, *Environmental Engineering Science*. John Wiley and Sons, Inc., 2001.

[14] R. Deiterding, "Lecture notes on conduction heat transfer," November 2019.

[15] J. Shrimpton, *An Introduction to Engineering Thermofluids*. Cuesta Ltd, 2015.

[16] R. Deiterding, "Lecture notes on convective heat transfer," November 2019.

[17] V. Deprédurand, G. Castanet, and F. Lemoine, "Heat and mass transfer in evaporating droplets in interaction: Influence of the fuel," *International Journal of Heat and Mass Transfer*, vol. 53, p. 3495–3502, 2010.

[18] P. W. Atkins and J. D. Paula, *Physical chemistry.* Freeman, 2010.

[19] N. Ladommatos and D. W. Rose, "A model of droplet thermodynamic and dynamic behaviour in the port of a port-injected engine," *SAE Transactions*, vol. 105, pp. 560–573, 1996. [Online]. Available: http://www.jstor.org/stable/44736299

[20] A. P. Pinheiro and J. M. Vedovoto, "Evaluation of Droplet Evaporation Models and the Incorporation of Natural Convection Effects," *Flow, Turbulence and Combustion*, vol. 102, no. 3, p. 537–558, Jan 2018.

[21] A. Frohn, *Dynamics of Droplets.* Springer, 2000.

[22] H. J. Holterman, "Kinetics and evaporation of water drops in air," *IMAG Report*, 2003.

[23] M. Protheroe, "An Investigation of Droplet Evaporation Characteristics in an Ultrasound Environment," Ph.D. dissertation, Auckland University of Technology, 2014.

[24] W. A. Sirignano, *Fluid Dynamics and Transport of Droplets and Sprays.* Cambridge University Press, 1999.

[25] V. Lukashov, "On the Determination of the Surface Temperature of an Evaporating Liquid," *Theoretical Foundations of Chemical Engineering*, vol. 37, pp. 325–329, 07 2003.

[26] A. Kernan. (2020) HM-Droplet-Python. [Online]. Available: https://github.com/TheOfficialDarthVader/HM-Droplet-Python

[27] W. E. Ranz and W. R. Marhsall, "Evaporation from drops: II," *Chemical Engineering Progress*, pp. 173–180, 1952.

[28] S. C. Wong and A. C. Lin, "Internal temperature distributions of droplets vaporizing in high-temperature convective flows," *Journal of Fluid Mechanics*, vol. 237, pp. 671 – 687, 04 1992.

[29] A. Kernan. (2020) Oranges. [Online]. Available: https://github.com/TheOfficialDarthVader/Oranges-master

[30] E. Andrews. (2020) DEMOranges Commits. [Online]. Available: https://github.com/Xorgon/DEMOranges/commits/master

[31] pypi. (2019) pyopencl 2019.1.2. [Online]. Available: https://pypi.org/project/pyopencl/

[32] G. M. Harpole, "Droplet Evaporation in High Temperature Environments," *Journal of Heat Transfer*, vol. 103, no. 1, pp. 86–91, 02 1981. [Online]. Available: https://doi.org/10.1115/1.3244437

[33] Y. A. Çengel, *Fundamentals of thermal-fluid sciences*, 3rd ed. McGraw-Hill, 2008.

[34] R. C. Reid, B. E. Poling, and J. M. Prausnitz, *The Properties of Gases and Liquids.* McGraw-Hill, 1987.

[35] B. Abramzon and W. A. Sirignano, "Droplet vaporization model for spray combustion calculations," *International Journal of Heat and Mass Transfer*, vol. 32, no. 9, p. 1605–1618, 1989.