University of Southampton

FEEG3003 Individual Project

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

by:
Andrew Kernan

Supervisor:
Prof. John Shrimpton

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

March 20, 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 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. With the final simulation code being written in C and OpenCL. A number of simulations with large populations of particles will be run and statistics on the particles calculated.

Findings

Summary

Nomenclature

Arabic Symbols

$\bar{R}$	Universal 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_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
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
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 at current time level
$n+1$	Value at next discrete time level
s	Property at droplet surface
sat	Property at saturation

Superscripts

P	Placeholder

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
MPI	Message Passing Interface
ODE	Ordinary Differential Equation
OpenCL	Open Computing Language

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 107 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 Kerosene Particle Evaporation for Large Particle Populations Using GPUs and the OpenCL language”. With the aim to To simulate the evolution of heat and mass transfer for 10^6 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 Kerosene particle.
3. Run large scale simulations (1 million particles). 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.

1 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 evaporation is titled “Evaluation of equilibrium and non-equilibrium evaporation models for many-droplet gas-liquid flow simulations” by Miller, Harstad and Bellan. Which details eight different droplet evaporation models based on 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 X the model of choice for this project is M1 which is a form of the classical evaporation model known as the infinite conductivity model.

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 [3]. 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 [4], the same as those used in [2]. This is a good indication that although the experimental data used in [2] is still relevant.

[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 [5].

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]. Older papers such as "A Comparison of Vaporization Models in Spray Calculations" by Aggarwal et al. use a second order Runge-Kutta method [7]. An exception is a paper on experimental and computational studies of liquid aerosol evaporation that used a forward Euler integration method for the time-dependent variables.

2 Background Material

2.1 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 2.1.

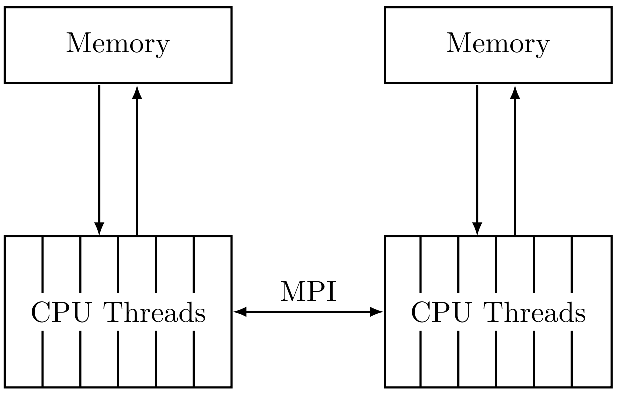


Figure 2.1: MPI block diagram.

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 2.2. With regards to writing GPU kernels there are two languages to choose from; OpenCL and 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.

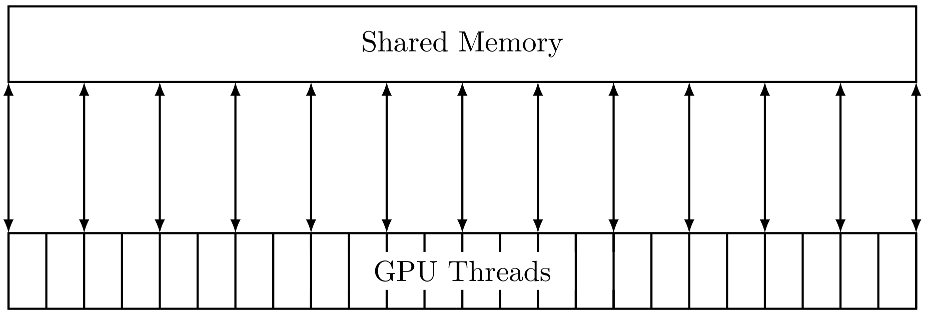


Figure 2.2: GPU memory block diagram.

2.2 Physics of Droplet Evaporation

2.2.1 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:

$$J(x, y, z) = -d\left(\frac{\partial C}{\partial x}, \frac{\partial C}{\partial y}, \frac{\partial C}{\partial z}\right) \tag{2.1}$$

[8].

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

$$\dot{q}_x = -k\frac{dT}{dx} \tag{2.2}$$

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. [9]

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:

$$\dot{q} = h\Delta T \tag{2.3}$$

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.[10]

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. [11]

2.2.2 Vapour Pressure

At the droplet's surface a number of processes take place. 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 [8].

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 >> than the molar volume of the liquid to derive the Clausius-Clapeyron equation. [12]

$$\chi_{s,eq} = \frac{P_{atm}}{P_G} \exp\left[\frac{L_V}{\bar{R}/W_V}\left(\frac{1}{T_B} - \frac{1}{T_d}\right)\right] \tag{2.4}$$

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

2.2.3 Droplet Model

For developing a simulation code to model the physics of droplets it is impractical to model the droplets as three dimensional fluid objects with the associated boundary layers, etc... Especially given the code must be robust and efficient enough to allow for a large population of particles to be simulated in a reasonable time. Therefore, each droplet is considered as a point mass and the physics of what process are taking place on the droplet are modelled.

The first assumption made in deriving a model is the droplet is spherical, with mass m_d and diameter D. With mass and diameter related by:

$$\begin{aligned} m_d &= \rho_d V_d && \text{(2.5a)} \\ &= \rho_d \left(\tfrac{4}{3}\right)\pi\left(\tfrac{D}{2}\right)^3 && \text{(2.5b)} \\ &= \tfrac{\rho_d \pi D^3}{6} && \text{(2.5c)} \end{aligned}$$

Where ρ_d is the droplet density, assumed to be constant.

Droplet diagram

The droplet consists of a a liquid phase surrounded by a vapour phase, with the droplet surrounded by a gas phase. The conditions of the gas phase are assumed to be constant and are calculated only at the start of the simulation. The processes occurring within the droplets phases can be reduced to modelling what takes place on a single boundary at the droplet's surface.

Droplet surfaces figure

3 Droplet Evaporation Model

The 8 models in [2] are denoted by Mx where x is the model number. Each of the models are defined by the same four Lagrangian equations for the droplets:

$$\frac{dX_i}{dt} = v_i \tag{3.1}$$

$$\frac{dv_i}{dt} = \left(\frac{f_1}{\tau_d}\right)(u_i - v_i) + g_i \tag{3.2}$$

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) + \left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d} - H_{\Delta T} \tag{3.3}$$

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\left(\frac{m_d}{\tau_d}\right)H_M \tag{3.4}$$

Equation 3.1 and a simpler form of equation 3.2 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:

$$X_{n+1} = x_n + \frac{v_n + v_{n+1}}{2}\Delta t \tag{3.5}$$

[1]

For equation 3.2, the acceleration of the droplet is defined using Stokes' Law which in its simplest form is given as:

equation

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. This equation can be derived by considering the forces acting on the particle:

equation

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

$$f_1 = \frac{1 + 0.0545Re_d + 0.1Re_d^{0.5}(1 - 0.03Re_d)}{1 + a|Re_b|^b} \tag{3.6}$$

Which is an empirical result produced from a data fit. It is therefore only applicable for specific conditions.

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 the classical rapid mixing model derived on the assumption that T_d is x. 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.

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 will simplify producing code for an extra model. The choice of model depends on results that are available for comparison in literature.

Models M7 and M8, are to be ruled out as they use non-equilibrium formulations of the coefficients which would further complicate the code.

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	$\ln\left[1 + B_{M,eq}\right]$

Table 3.1: Detail of variables used in candidate models

4 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 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).

In terms of implementation the order in which the equations are solved may prove to be important. For example at a given time level should the temperature of the droplet be found before the mass? And what kind of error checking within the time step should take place?

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

4.1 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:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + .. + \frac{h^{n-1}}{(n-1)!}f^{(n-1)}(x) + \frac{h^n}{n!}f^n(c) \tag{4.1}$$

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

$$f(n+1) = f(n) + \Delta t f'(n) \tag{4.2}$$

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

$$\frac{d\phi}{dt} = f(t,\ \phi) \tag{4.3}$$

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

$$\phi_{n+1} = \phi_n + \Delta t f(t_n,\ \phi_n) \tag{4.4}$$

To produce a solution to the ODE requires stepping forward in time and calculating the solution numerically at discrete time steps. This can be achieved by using a loop structure as shown in Figure 4.1.

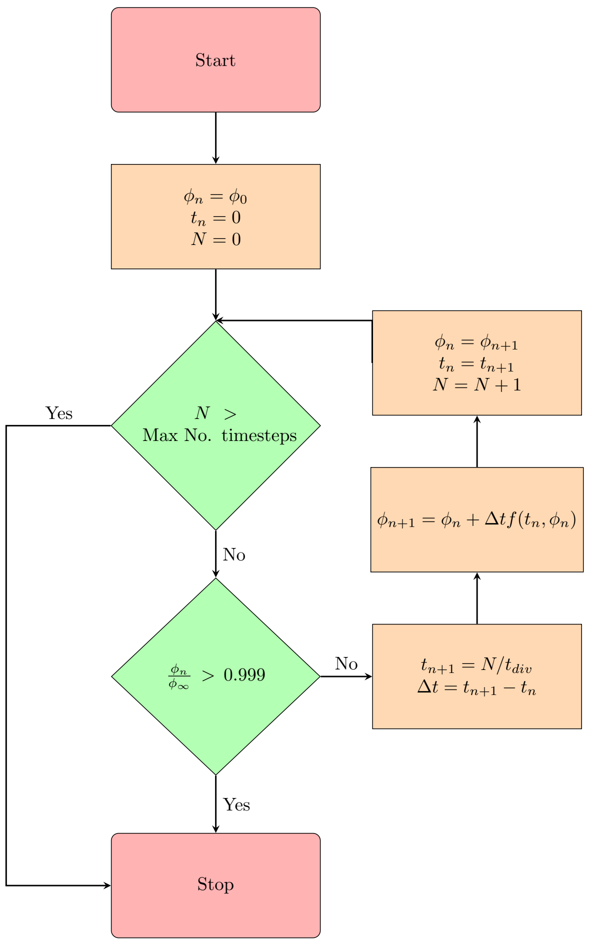


Figure 4.1: Flowchart for the time stepping process for the forward Euler method.

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 current to initial mass being greater than zero. 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 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:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) \tag{4.5a}$$

$$f(x+h) = f(x) + hf'(x) + O(h^2) \tag{4.5b}$$

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 $1/h$.

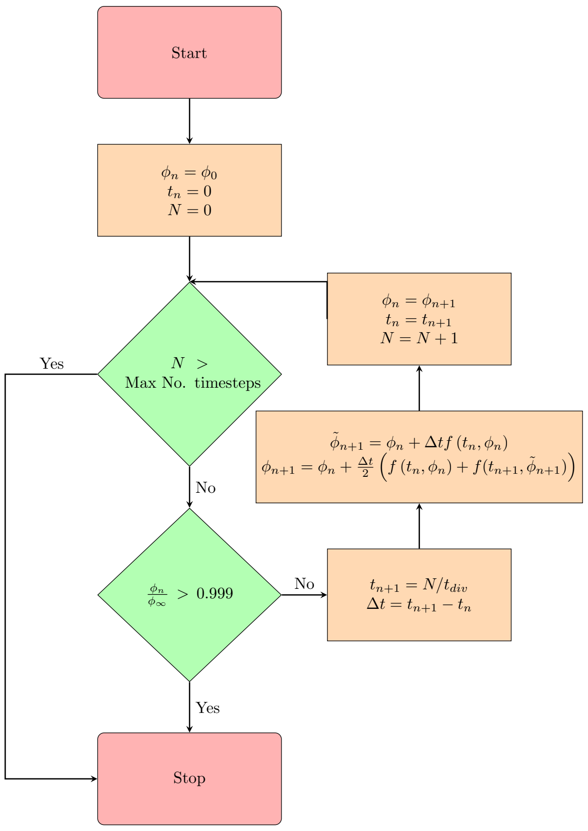
4.2 Modified Euler Method

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 that 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:

$$\phi_{n+1} = \phi_n + \frac{\Delta t}{2}\left[f(t_n, \phi_n) + f(t_{n+1}, \phi_{n+1})\right] \tag{4.6}$$

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 4.2.



Start
$\phi_n = \phi_0$
$t_n = 0$
$N = 0$
$N >$ Max No. timesteps
Yes
No
$\frac{\phi_n}{\phi_\infty} > 0.999$
No
Yes
Stop
$t_{n+1} = N/t_{div}$
$\Delta t = t_{n+1} - t_n$
$\tilde{\phi}_{n+1} = \phi_n + \Delta t f(t_n, \phi_n)$
$\phi_{n+1} = \phi_n + \frac{\Delta t}{2}\left(f(t_n, \phi_n) + f(t_{n+1}, \tilde{\phi}_{n+1})\right)$
$\phi_n = \phi_{n+1}$
$t_n = t_{n+1}$
$N = N + 1$

Figure 4.2: Flowchart for the time stepping process for the modified Euler method.

4.3 Runge-Kutta Method

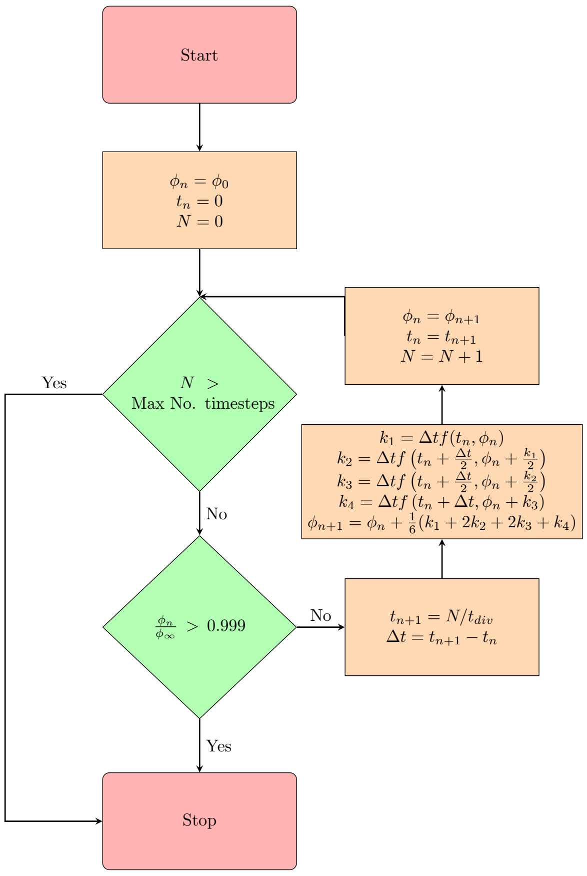
The modified Euler method (also known as the Heun formula) is a predictor corrector method. In fact it is an example of a more general family of predictor corrector methods known as the Runge-Kutta method. The modified Euler method is in fact a second order Runge-Kutta method:

$$k_1 = \Delta t f(t_n, \phi_n) \tag{4.7a}$$
$$k_2 = \Delta t f(t_n + \Delta t, \phi_n + k_1) \tag{4.7b}$$
$$\phi_{n+1} = y_n + \tfrac{1}{2}(k_1 + k_2) \tag{4.7c}$$

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.

$$k_1 = \Delta t f(t_n, \phi_n) \tag{4.8a}$$
$$k_2 = \Delta t f\left(t_n + \frac{\Delta t}{2}, \phi_n + \frac{k_1}{2}\right) \tag{4.8b}$$
$$k_3 = \Delta t f\left(t_n + \frac{\Delta t}{2}, \phi_n + \frac{k_2}{2}\right) \tag{4.8c}$$
$$k_4 = \Delta t f\left(t_n + \Delta t, \phi_n + k_3\right) \tag{4.8d}$$
$$\phi_{n+1} = \phi_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.8e}$$

More weighting is applied to the terms k_2 and k_3 as these are estimates of the midpoint of the interval. The time stepping process is no different than for the previous two methods:



Start
$\phi_n = \phi_0$
$t_n = 0$
$N = 0$
$N >$ Max No. timesteps
Yes
No
$\phi_n = \phi_{n+1}$
$t_n = t_{n+1}$
$N = N + 1$
$k_1 = \Delta t f(t_n, \phi_n)$
$k_2 = \Delta t f\left(t_n + \frac{\Delta t}{2}, \phi_n + \frac{k_1}{2}\right)$
$k_3 = \Delta t f\left(t_n + \frac{\Delta t}{2}, \phi_n + \frac{k_2}{2}\right)$
$k_4 = \Delta t f\left(t_n + \Delta t, \phi_n + k_3\right)$
$\phi_{n+1} = \phi_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
$\frac{\phi_n}{\phi_\infty} > 0.999$
No
$t_{n+1} = N/t_{div}$
$\Delta t = t_{n+1} - t_n$
Yes
Stop

Figure 4.3: Flowchart for the time stepping process for the Runge-Kutta method.

4.4 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 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:

$$t_{n+1} = t_n + \Delta t \tag{4.9a}$$
$$y_{n+1} = y_n + \Delta t f(t_{n+1}, y_{n+1}) \tag{4.9b}$$

Unlike the explicit methods, the implicit method will have to be derived for each ODE. For the temperature ODE:

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) + \left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d} - H_{\Delta T} \tag{4.10}$$

$$t_{n+1} = \Delta t \tag{4.11a}$$
$$T_{d_{n+1}} = T_{d_n} + \Delta t\left[\frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_{d_{n+1}}) + \left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d} - H_{\Delta T}\right] \tag{4.11b}$$

For the second equation rearrangement is required, the full process is shown in Section B.1:

$$T_{d_{n+1}} = T_{d_n} + \Delta t\left[\frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_{d_{n+1}}) + \left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d} - H_{\Delta T}\right] \tag{4.12a}$$
$$T_{d_{n+1}} = \frac{T_{d_n} + \Delta t\left[T_G \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right) + \left(\frac{L_V}{C_L}\right)\frac{\dot{m}_d}{m_d} - H_{\Delta T}\right]}{\left(1 + \Delta t \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)\right)} \tag{4.12b}$$

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

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\left(\frac{m_d}{\tau_d}\right)H_M \tag{4.13}$$

$$t_{n+1} = \Delta t \tag{4.14a}$$
$$m_{d_{n+1}} = m_{d_n} + \Delta t\left[-\frac{Sh}{3Sc_G}\left(\frac{m_{d_{n+1}}}{\tau_d}\right)H_M\right] \tag{4.14b}$$

Rearranging the second equation leads to:

$$m_{d_{n+1}} = \frac{m_{d_n}}{1 + \Delta t \frac{Sh}{3Sc_G}\left(\frac{H_M}{\tau_d}\right)} \tag{4.15a}$$

(The full solution can be found in Section B.2). These numerical methods will be evaluated in the section to determine their suitability.

5 Solution Method

The ODEs for mass and heat transfer and 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.

5.1 Uncoupled Heat Transfer

To simplify the temperature ODE, the Reynold's number was set to zero, which means the drop is stationary and there is only diffusive heat transfer. This fixes the value of the Nusselt number as constant. Furthermore, the classical rapid mixing model was used meaning $f_2 = 1$ 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. Therefore the temperature ODE becomes:

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) \tag{5.1}$$

Which can be solved analytically to provide a reference point for the numerical solution. See section B.3 for the full derivation which produces the solution:

$$T_{d_{n+1}} = T_G - (T_G - T_{d_n})e^{-\left(\frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)\right)\Delta t} \tag{5.2}$$

The numerical solution uses the Euler forward method, which can be used to solve an ODE of the form:

$$\frac{dy}{dt} = f(t,\ y) \tag{5.3}$$

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

$$y_{n+1} = y_n + \Delta t f(t_n,\ y_n) \tag{5.4}$$

With $n+1$ being the output at the end of the timestep and n being the current value. Applying this result to equation 5.1:

$$T_{d_{n+1}} = T_{d_n} + \Delta t\left(\frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d)\right) \tag{5.5}$$

The implementation of equation 5.5 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.

The full settings used for the simulation are shown in Table x.

For plotting data the results have been non-dimensionalised. The points in time are non-dimensionalised with the particle relaxation time constant τ. 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:

$$\frac{dT_d}{dt} = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) \tag{5.6a}$$

$$0 = \frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)(T_G - T_d) \tag{5.6b}$$

$$T_d = \tag{5.6c}$$

Parameter	**Setting**
Droplet Properties	
Molecular Weight of Vapour Phase W_V	18.015 $kg/kgmol$
Boiling Temperature T_B	373.15 K
Latent Heat of Evaporation L_V	
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 squared ${D_0}^2$	1.1 mm
Initial Velocity u_0	0 m/s
Initial Reynolds Number Re_0	0
Gas Properties	
Pressure P_G	101325 Pa
Temperature T_G	298 K
Density ρ_G	
Kinematic Viscosity μ_G	
Molecular Weight of Phase W_C	28.97 $kg/(kg\ mol)$
Prandtl Number Pr_G	
Schimdt Number Sh_G	
Atmospheric Properties	
Pressure P	$P = P_G$
Temperature T	$T = T_G$
Non-Dimensional Numbers	
Lewis Number	1
Sherwood Number Sh	
Nusselt Number Nu	
Physical Constants	
Gas Constant $\bar{R}$	8314.5 $J/(K(kg\ mol))$
Universal Gas Constant R	287 $J/(kg\ K)$
Specific Heat for Constant Pressure C_{PG}	
Simulation Properties	
Correction factor f_2	1
Driving Potential $H_{\Delta T}$	0
Driving Potential H_M	$\ln(1 + B_M, eq)$

The results from a simulation run for one droplet of water evaporating in air for a time step of 10s are shown in Figure 5.1.

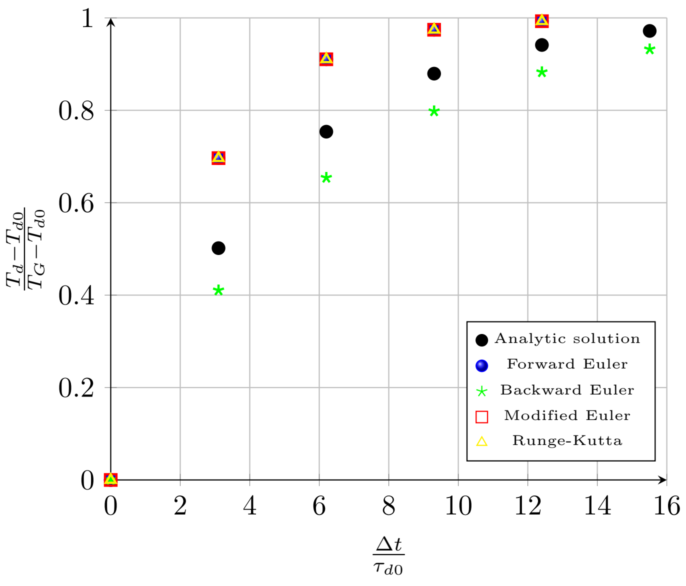


Figure 5.1: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = 10s$.

For smaller timesteps of $5s$ and $1s$:

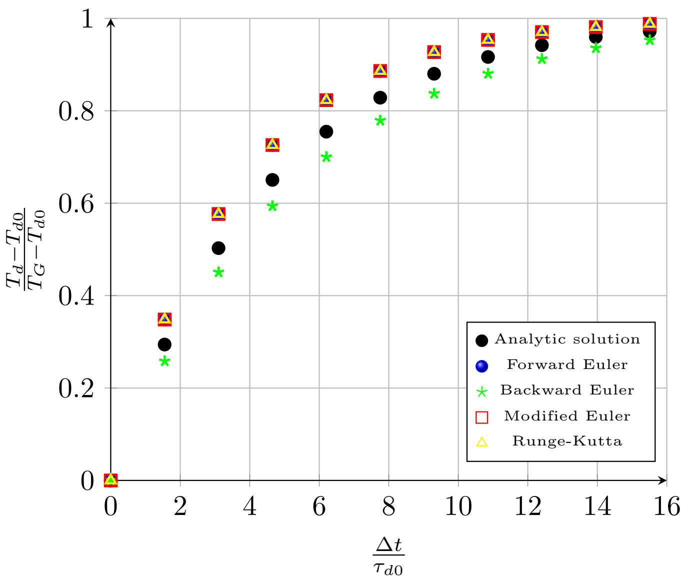


Figure 5.2: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = 5s$.

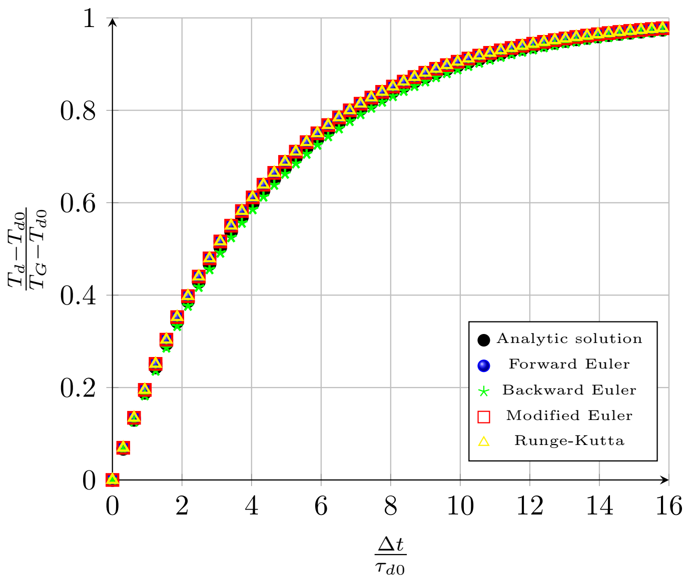


Figure 5.3: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$ and $\Delta t = 1s$.

5.2 Uncoupled Mass Transfer

The mass transfer equation is:

$$\frac{dm_d}{dt} = -\frac{Sh}{3Sc_G}\frac{m_d}{\tau_d}H_M \tag{5.7}$$

A numerical solution given by the forward Euler method is:

$$m_{d_{n+1}} = m_{d_n} + \Delta t\left(-\frac{Sh}{3Sc_G}\frac{m_{d_n}}{\tau_d}H_M\right) \tag{5.8}$$

The time stepping process in Figure 5.4 is similar to the one used for the heat transfer solution.

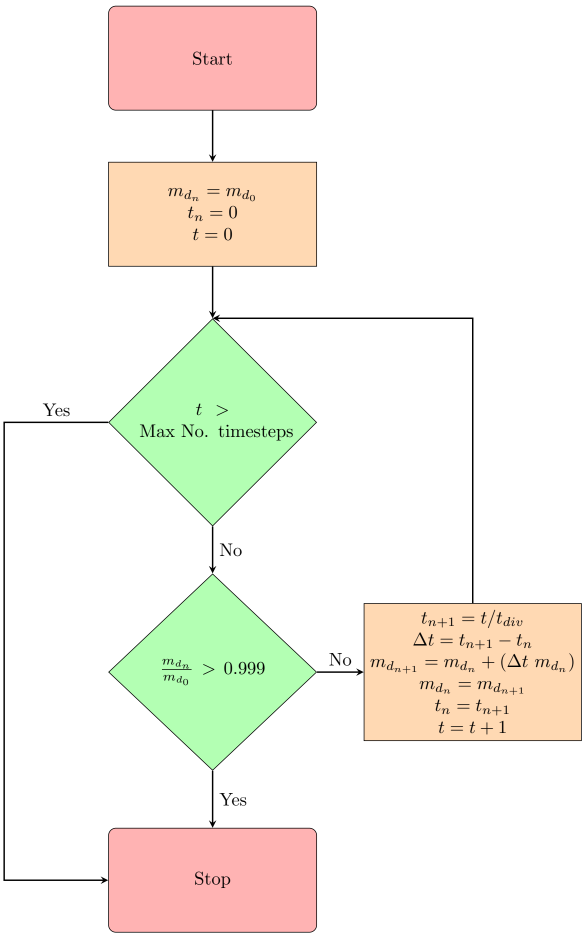


Figure 5.4: Flowchart for the heat transfer time stepping process for the forward Euler method.

Solving equation 5.7 for diameter, requires a relation between diameter D and m_d:

$$\begin{aligned} m_d =& \quad \rho_d V_d && \text{(5.9a)} \\ m_d =& \quad \rho_d\left(\tfrac{4}{3}\right)\pi\left(\tfrac{D}{2}\right)^3 && \text{(5.9b)} \\ m_d =& \quad \tfrac{\rho_d \pi D^3}{6} && \text{(5.9c)} \end{aligned}$$

So:

$$\frac{dm_d}{dt} = \frac{\rho_d \pi D^2}{2}\frac{dD}{dt} \tag{5.10}$$

These can be substituted into equation 5.7, along with $\tau_d = \rho_d D^2/(18\mu_g)$:

$$\frac{dm_d}{dt} = -\tfrac{Sh}{3Sc_G}\tfrac{m_d}{\tau_d}H_M \tag{5.11a}$$

$$\frac{\rho_d \pi D^2}{2}\frac{dD}{dt} = -\tfrac{Sh}{3Sc_G}\tfrac{18\rho_d\pi D^3\mu_g}{6\rho_d D^2}H_M \tag{5.11b}$$

$$\frac{1}{2}\left[D_{n+1}^2 - D_n^2\right] = -\tfrac{2Sh}{Sc_G\rho_d}\mu_g H_M\ \Delta t \tag{5.11c}$$

$$D_{n+1}^2 = D^2 - \tfrac{4Sh}{Sc_G\rho_d}\mu_g H_M\ \Delta t \tag{5.11d}$$

A numerical solution to the D^2 ODE can be found as:

$$D_{n+1} = D_n - \Delta t\left(\frac{2Sh}{Sc_G\rho_d}\mu_g H_M \frac{1}{D_n}\right) \tag{5.12}$$

A rearrangement of equation 5.9 gives the droplet diameter as:

$$m_d = \tfrac{\rho_d\pi D^3}{6} \tag{5.13a}$$

$$D = \left(\tfrac{6}{\rho_d\pi}\right)^{1/3}(m_d)^{1/3} \tag{5.13b}$$

Therefore:

$$\frac{dm_d}{dt} = -\tfrac{Sh}{3Sc_G}\tfrac{m_d}{\tau_d}H_M \tag{5.14a}$$

$$\frac{dm_d}{dt} = -\tfrac{Sh}{3Sc_G}H_M m_d\left(\tfrac{18\mu_G}{\rho_d D^2}\right) \tag{5.14b}$$

$$\frac{dm_d}{dt} = -\tfrac{Sh}{3Sc_G}H_M m_d\left(\tfrac{18\mu_G}{\rho_d}\right)\left(\left(\tfrac{6}{\rho_d\pi}\right)^{-2/3}(m_d)^{-2/3}\right) \tag{5.14c}$$

$$(m_{dn+1})^{2/3} = (m_{dn})^{2/3} - \tfrac{2Sh}{3Sc_G}\mu_G H_M\left(\tfrac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\ \Delta t \tag{5.14d}$$

The full integration steps for the mass and diameter equations are given in Section B.4.

Calculating H_M:

$$H_M = \ln\left[1 + B_{M,eq}\right] \tag{5.15}$$

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

$$B_{M,eq} = \frac{Y_{s,eq} - Y_G}{1 - Y_{s,eq}} \tag{5.16}$$

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

$$Y_{s,eq} = \frac{\chi_{s,eq}}{\chi_{s,eq} + (1-\chi_{s,eq})\theta_2} \tag{5.17}$$

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

$$\theta_2 = \frac{W_C}{W_V} \tag{5.18}$$

For water evaporating in air $W_V = 18.015\ kg/(kg\ mole)$ is the molecular weight of water vapour and $W_C = 28.97\ kg/(kg\ mole)$ is the molecular weight of air. So $\theta_2 = 1.61$.

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:

$$\chi_{s,eq} = \frac{P_{sat}}{P_G} \tag{5.19a}$$

$$\chi_{s,eq} = \frac{P_{atm}}{P_G} \exp\left[\frac{L_V}{\bar{R}/W_V}\left(\frac{1}{T_B} - \frac{1}{T_d}\right)\right] \tag{5.19b}$$

Where:

- $P_{atm} = 101325\ Pa$
- $P_G = \rho_G R T_G = (1.225\ kg/m^3)(287\ J/kg\ k)(298\ K) = 104769\ Pa$
- $\bar{R} = 8.3145\ J/(K\ mole) = 8314.5\ J/(K\ kg\ mol)$
- $L_V = 2.257 \times 10^6 + 2.595 \times 10^3(373.15 - T)\ J/kg$ Latent heat of evaporation for water
- $T_B = 373.15\ K$ boiling temperature of water

The conditions used:

- Droplet species - water
- Gas species - air
- $m_{d,0} = 0.001\ kg$
- $D_{d,0} = 1.24\ mm$
- $T_G = 365\ K$
- $T_{d,0} = 350\ K$
- $Re_d = 0$
- $L_V = 2.45 \times 10^6\ J/kg$ using $T = T_G$

This gives $\chi_{s,eq}$ as:

$$\chi_{s,eq} = \frac{101325\ Pa}{104769\ Pa} \exp\left[\frac{2.45\times10^6\ J/kg}{8314.5\ J/(K\ kg\ mol)/18.015\ kg/(kg\ mole)}\left(\frac{1}{373.15\ K} - \frac{1}{365\ K}\right)\right] \tag{5.20a}$$

$$\chi_{s,eq} = 0.967 \exp\left[(5308\ K)(-5.98 \times 10^{-5}\ /K)\right] \tag{5.20b}$$

$$\chi_{s,eq} = 0.967 \exp[-0.318] \tag{5.20c}$$

$$\chi_{s,eq} = 0.704 \tag{5.20d}$$

Therefore $Y_{s,eq}$ is:

$$Y_{s,eq} = \frac{\chi_{s,eq}}{\chi_{s,eq} + (1-\chi_{s,eq})\theta_2} \tag{5.21a}$$

$$Y_{s,eq} = \frac{0.704}{0.704 + (1-0.704)1.61} \tag{5.21b}$$

$$Y_{s,eq} = 0.596 \tag{5.21c}$$

And:

$$B_{M,eq} = \frac{Y_{s,eq} - Y_G}{1 - Y_{s,eq}} \tag{5.22a}$$

$$B_{M,eq} = \frac{0.596 - 0}{1 - 0.596} \tag{5.22b}$$

$$B_{M,eq} = 1.48 \tag{5.22c}$$

Finally:

$$H_M = \ln\left[1 + B_{M,eq}\right] \tag{5.23a}$$

$$H_M = \ln\left[1 + 1.48\right] \tag{5.23b}$$

$$H_M = 0.908 \tag{5.23c}$$

The mass transfer solution has been plotted in Figure 5.5 for $\Delta t = 5\ s$. For the numerical solution D and τ_d were recalculated at the start of each timestep. There is also a plot of D^2 in Figure 5.6

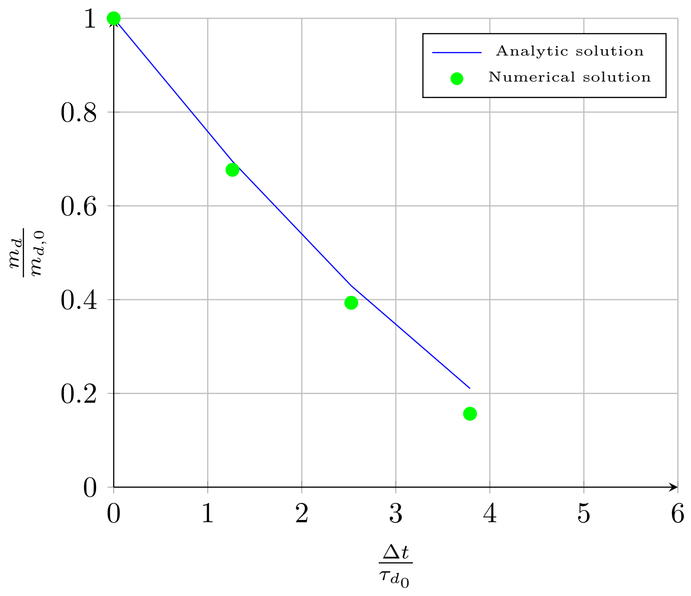


Figure 5.5: m_d for a droplet sized $D = 1.24mm$ with $Re_d = 0$, $T_{d_0} = 350K$, $T_G = 365K$, $Y_G = 0$ and $\Delta t = 5s$.

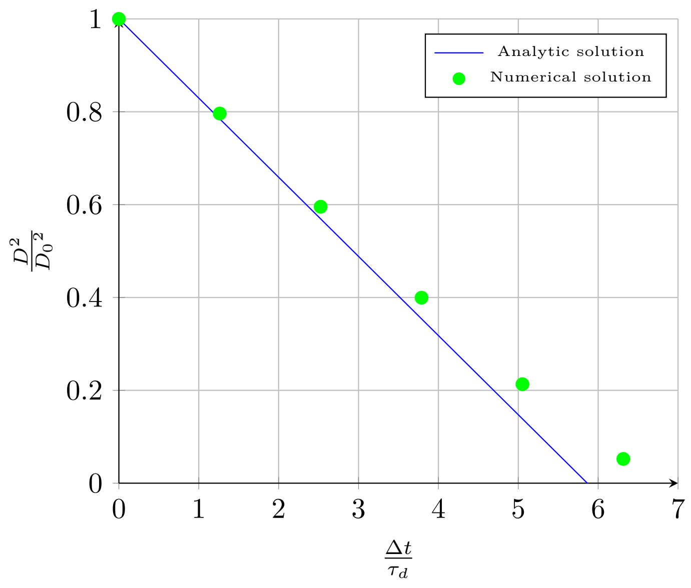


Figure 5.6: D^2 for a droplet sized $D = 1.24mm$ with $Re_d = 0$, $T_{d_0} = 350K$, $T_G = 365K$, $Y_G = 0$ and $\Delta t = 5s$.

5.3 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 ODEs coupled has not been found so the equations can only be solved numerically.

The following figures use the data from Miller et al. corresponding to Figure 2 in that paper. Figures 5.7 and 5.8 show the results for a forward Euler method and Figures 5.9

and 5.10 show the results for a fourth order Runge-Kutta method. For both methods a timestep of $0.01s$ was used and the final mass was limited to $0.000000001\ kg$.

Figures 5.11 and 5.12 shows the effect of increasing the timestep Δt.

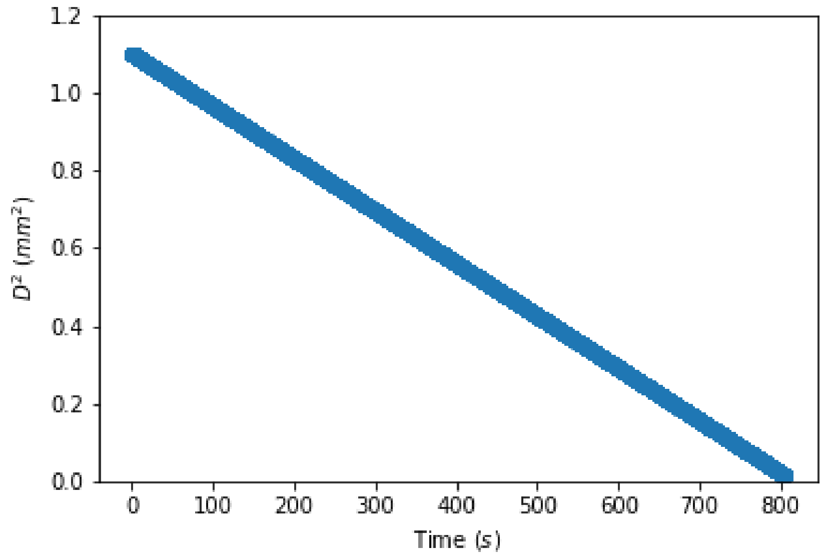


Figure 5.7: D^2 for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$, $Y_G = 0$ and $\Delta t = 0.01s$. Using a forward Euler method.

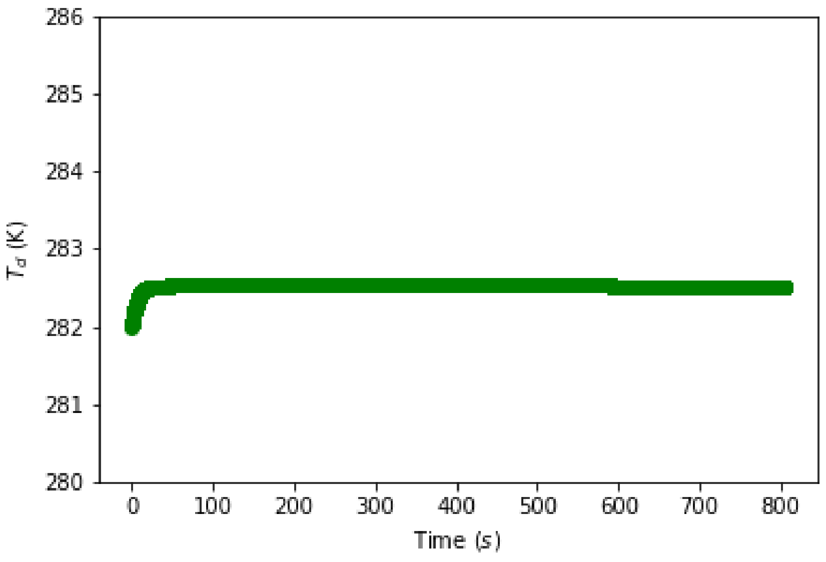


Figure 5.8: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$, $Y_G = 0$ and $\Delta t = 0.01s$. Using a forward Euler method.

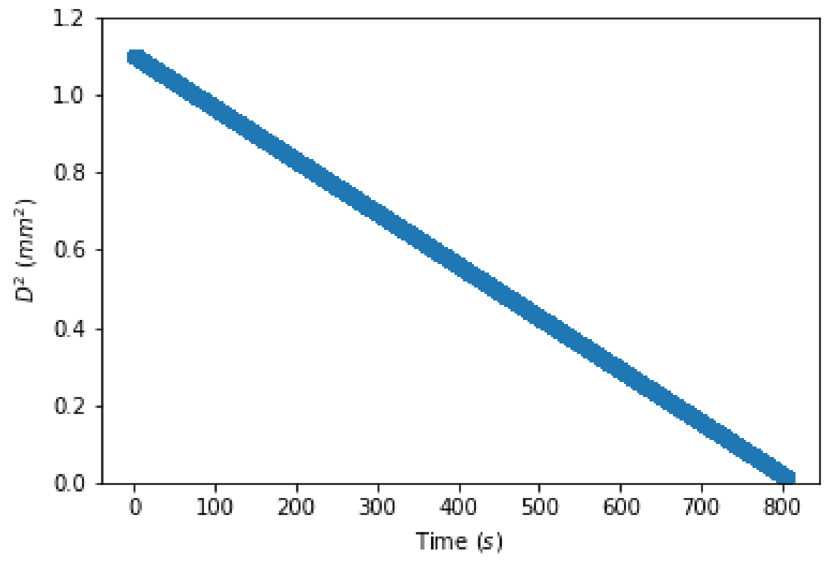


Figure 5.9: D^2 for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$, $Y_G = 0$ and $\Delta t = 0.01s$. Using a Runge-Kutta method.

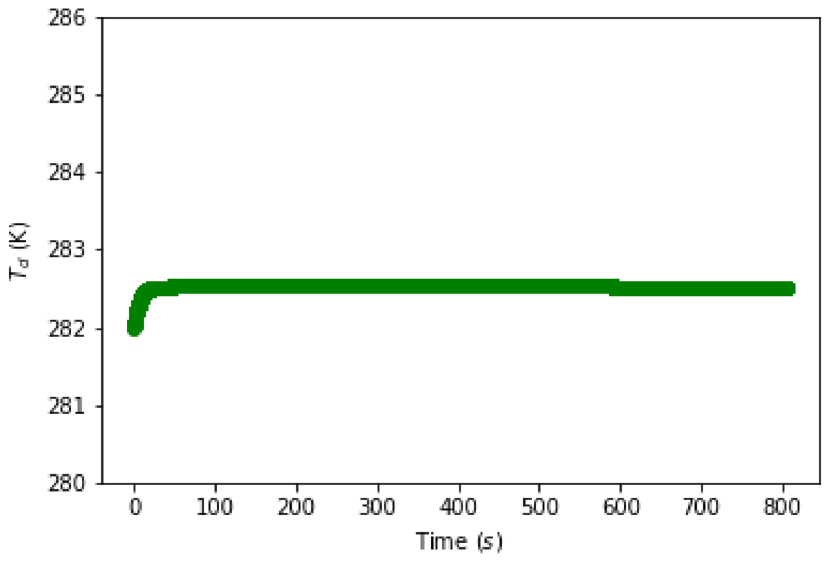


Figure 5.10: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$, $Y_G = 0$ for $\Delta t = 0.01s$. Using a Runge-Kutta method.

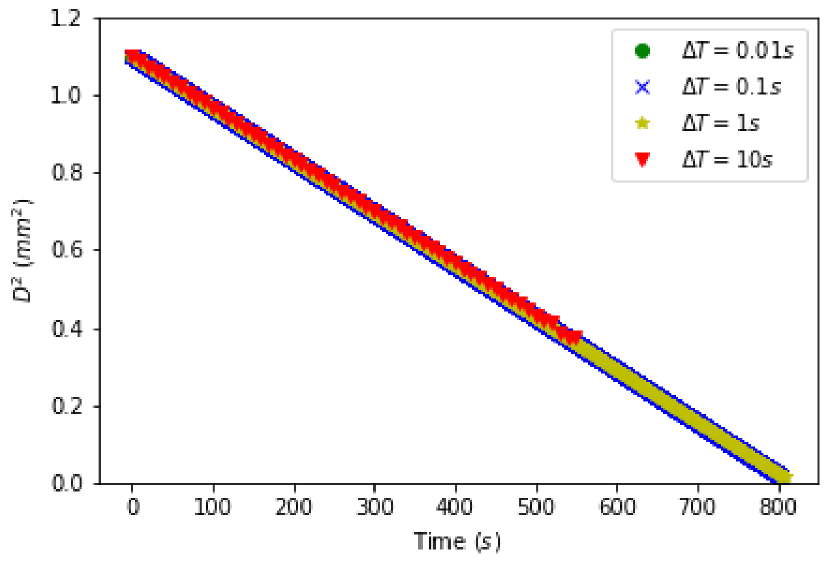


Figure 5.11: D^2 for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$, $Y_G = 0$ for different Δt. Using a Runge-Kutta method.

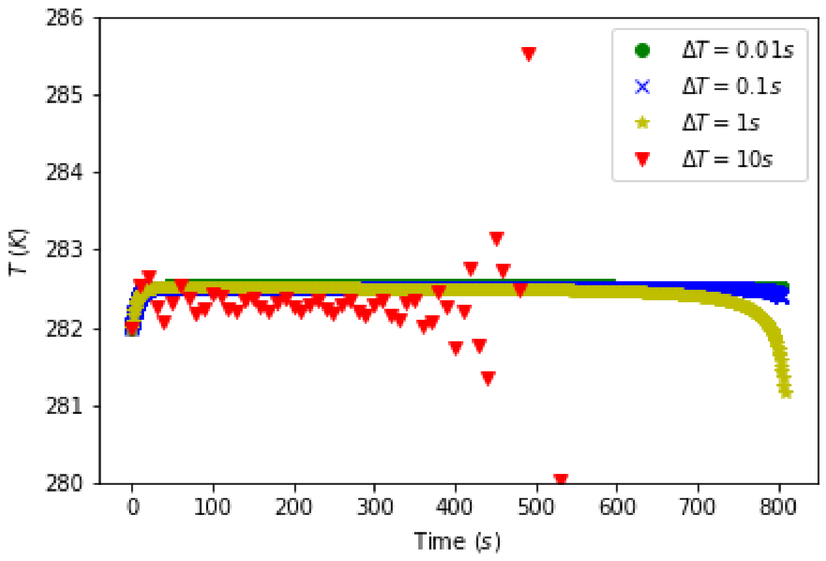


Figure 5.12: T_d for a droplet sized $D^2 = 1.1mm$ with $Re_d = 0$, $T_{d_0} = 282K$, $T_G = 298K$, $Y_G = 0$ and different Δt. Using a Runge-Kutta method.

As can be seen for larger timesteps the explicit methods diverge. The point of divergence occurs earlier into the simulation run as the timestep size is increased. Plotting the results non-dimensionally makes it clear what causes this:

Figure.

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 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, an appropriate solution is to use an implicit method. The backwards 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. Other 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 GPU simulation code.

6 Verification and Validation of the Existing Code

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 some checks that need to be done to ensure the code is solving the equations correctly and producing physical results.

The 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:

$$\bar{U}_{x,rel}(t) = \sum_{i=1}^{N} \frac{[U_{p_i}(t) - U_{f_i}(t)]}{N} \tag{6.1a}$$

$$\bar{V}_{y,rel}(t) = \sum_{i=1}^{N} \frac{[V_{p_i}(t) - V_{f_i}(t)]}{N} \tag{6.1b}$$

$$\bar{W}_{z,rel}(t) = \sum_{i=1}^{N} \frac{[W_{p_i}(t) - W_{f_i}(t)]}{N} \tag{6.1c}$$

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 be the same for the other statistical moments.

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 default conditions the code uses for the simulation of a Taylor-Green vortex are:

- Number of particles: 10000
- Particle density: 2000 kg/m^3
- Particle diameter: 0.05 m
- Fluid viscosity: 0.0193 Ns/m^2 (according to the report 1.040 should be used for a Stokes number of 1)

The distribution of starting conditions gives a mean particle speed of $1m/s$ and standard deviation of $0.1m/s$. With the particle directions evenly distributed.

For the sake of clarity here is the structure of the code without the DEM:

Figure.

A couple of missteps were made in the process of cutting down the code. The first was removing the boundary condition for the particle location. The simulation domain is cube, when a particle reaches the end of the cube its position is reset at the other side of the box as shown in Figure x.

Figure x

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.

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:

Table

$$Size = x \tag{6.2a}$$

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.

7 OpenCL Implementation

The initial testing of numerical methods and the model were performed in Python to 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 [13]. However, Python is not performance oriented [citation needed] and the existing code uses C and OpenCL to build executable files for running simulations.

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

- variable

In addition to this code was added to the function that initialises the particles so the extra required variables are set at 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.

8 Results

9 Conclusion

9.1 Future Work

There are a number of possible avenues of future work that could be taken based on this project.

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 alterion 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, for example common rocket engine propellants.

This leads into another point of future work. Currently the code must be re-built everytime 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.

In addition 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 a text file may provide the required performance improvement. Although the choice of numerical scheme has been justified on the basis of simplicity and performance it would be interesting to evaluate higher order implicit methods quantitatively. As fourth order schemes are quite common in the literature.

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 Backward Euler Method for Temperature ODE

Rearrangement of the backward Euler method for the temperature ODE:

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

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

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

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

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

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

B.2 Backward Euler Method for Mass ODE

Rearrangement of the backward Euler method for the mass ODE:

$$m_{d_{n+1}} = m_{d_n} + \Delta t\left[-\frac{Sh}{3Sc_G}\left(\frac{m_{d_{n+1}}}{\tau_d}\right)H_M\right] \tag{B.2a}$$

$$\frac{m_{d_{n+1}}}{\Delta t} = \frac{m_{d_n}}{\Delta t} + \left[-\frac{Sh}{3Sc_G}\left(\frac{m_{d_{n+1}}}{\tau_d}\right)H_M\right] \tag{B.2b}$$

$$\frac{m_{d_{n+1}}}{\Delta t} + \frac{Sh}{3Sc_G}\left(\frac{m_{d_{n+1}}}{\tau_d}\right)H_M = \frac{m_{d_n}}{\Delta t} \tag{B.2c}$$

$$m_{d_{n+1}}\left[\frac{1}{\Delta t} + \frac{Sh}{3Sc_G}\left(\frac{H_M}{\tau_d}\right)\right] = \frac{m_{d_n}}{\Delta t} \tag{B.2d}$$

$$m_{d_{n+1}} = \frac{\frac{m_{d_n}}{\Delta t}}{\frac{1}{\Delta t} + \frac{Sh}{3Sc_G}\left(\frac{H_M}{\tau_d}\right)} \tag{B.2e}$$

$$m_{d_{n+1}} = \frac{m_{d_n}}{1 + \Delta t\frac{Sh}{3Sc_G}\left(\frac{H_M}{\tau_d}\right)} \tag{B.2f}$$

B.3 Uncoupled Heat Transfer

Analytic solution of the uncoupled heat transfer ODE:

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

To make the solution steps clearer, define the constants:

$$A = \tfrac{f_2 Nu}{3Pr_G}\left(\tfrac{\theta_1}{\tau_d}\right) \tag{B.4a}$$
$$B = T_G \tag{B.4b}$$

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

$$\frac{dT_d}{dt} = A(B - T_d) \tag{B.5a}$$
$$\int_{T_{d_n}}^{T_{d_{n+1}}} \frac{dT_d}{(B - T_d)} = \textstyle\int_0^{\Delta t} A\ dt \tag{B.5b}$$
$$[\ln(B - T_d)]_{T_{d_n}}^{T_{d_{n+1}}} = [A\ t]_0^{\Delta t} \tag{B.5c}$$
$$-\ln(B - T_{d_{n+1}}) + \ln(B - T_{d_n}) = A\Delta t \tag{B.5d}$$
$$\ln\left(\frac{B - T_{d_{n+1}}}{B - T_{d_n}}\right) = -A\Delta t \tag{B.5e}$$
$$\frac{B - T_{d_{n+1}}}{B - T_{d_n}} = e^{-A\Delta t} \tag{B.5f}$$
$$B - T_{d_{n+1}} = (B - T_{d_n})e^{-A\Delta t} \tag{B.5g}$$
$$T_{d_{n+1}} = B - (B - T_{d_n})e^{-A\Delta t} \tag{B.5h}$$

Hence, the final analytical solution is:

$$T_{d_{n+1}} = T_G - (T_G - T_{d_n})e^{-\left(\frac{f_2 Nu}{3Pr_G}\left(\frac{\theta_1}{\tau_d}\right)\right)\Delta t} \tag{B.6}$$

B.4 Uncoupled Mass Transfer

The analytic solution for the uncoupled mass transfer ODE:

$$\frac{dm_d}{dt} = -\tfrac{Sh}{3Sc_G}\tfrac{m_d}{\tau_d}H_M \tag{B.7a}$$
$$\frac{\rho_d \pi D^2}{2}\frac{dD}{dt} = -\tfrac{Sh}{3Sc_G}\tfrac{18\rho_d \pi D^3 \mu_g}{6\rho_d D^2}H_M \tag{B.7b}$$
$$D\frac{dD}{dt} = -\tfrac{2Sh}{Sc_G \rho_d}\mu_g H_M \tag{B.7c}$$
$$\int_{D_n^2}^{D_{n+1}^2} D\ dD = -\tfrac{2Sh}{Sc_G \rho_d}\mu_g H_M \textstyle\int_0^{\Delta t}\ dt \tag{B.7d}$$
$$\left[\frac{D^2}{2}\right]_{D_n^2}^{D_{n+1}^2} = -\tfrac{2Sh}{Sc_G \rho_d}\mu_g H_M\ [t]_0^{\Delta t} \tag{B.7e}$$
$$\frac{1}{2}\left[D_{n+1}^2 - D_n^2\right] = -\tfrac{2Sh}{Sc_G \rho_d}\mu_g H_M\ \Delta t \tag{B.7f}$$
$$D_{n+1}^2 = D^2 - \tfrac{4Sh}{Sc_G \rho_d}\mu_g H_M\ \Delta t \tag{B.7g}$$

Solved for mass:

$$\frac{dm_d}{dt} = -\tfrac{Sh}{3Sc_G}\tfrac{m_d}{\tau_d}H_M \tag{B.8a}$$

$$\frac{dm_d}{dt} = -\tfrac{Sh}{3Sc_G}H_M m_d^{1/3}(3\mu_G)\left(\left(\tfrac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\right) \tag{B.8b}$$

$$\frac{dm_d}{dt} = -\tfrac{Sh}{Sc_G}\mu_G H_M\left(\tfrac{6}{\rho_d}\right)^{1/3}\pi^{2/3}m_d^{1/3} \tag{B.8c}$$

$$\int_{m_{dn}}^{m_{dn+1}}\frac{dm_d}{(m_d)^{1/3}} = -\tfrac{Sh}{Sc_G}\mu_G H_M\left(\tfrac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\textstyle\int_0^{\Delta t}\ dt \tag{B.8d}$$

$$\left[\frac{3}{2}(m_d)^{2/3}\right]_{m_{dn}}^{m_{dn+1}} = -\tfrac{Sh}{Sc_G}\mu_G H_M\left(\tfrac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\left[t\right]_0^{\Delta t} \tag{B.8e}$$

$$\frac{3}{2}\left[(m_{dn+1})^{2/3}-(m_{dn})^{2/3}\right] = -\tfrac{Sh}{Sc_G}\mu_G H_M\left(\tfrac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\ \Delta t \tag{B.8f}$$

$$(m_{dn+1})^{2/3} = (m_{dn})^{2/3}-\tfrac{2Sh}{3Sc_G}\mu_G H_M\left(\tfrac{6}{\rho_d}\right)^{1/3}\pi^{2/3}\ \Delta t \tag{B.8g}$$

As:

$$\tau_d = \frac{\rho_d D^2}{18\mu_g} \tag{B.9}$$

and:

$$D = \left(\frac{6}{\rho_d\pi}\right)^{1/3}(m_d)^{1/3} \tag{B.10}$$

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,” vol. 24, pp. 1025–1055, 1998.

[3] 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.

[4] C. Downingm, “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.

[5] E. K. Shashank, E. Knudsen, and H. Pitsch, “Spray evaporation model sensitivities,” *Annual Research Briefs of the CTR*, pp. 213–224, 2011.

[6] 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.

[7] S. Aggarwal, S. William, and A. Tong, “A comparison of vaporization models in spray calculations,” *AIAA Journal*, vol. 22, pp. 1448–1457, 09 1984.

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

[9] R. Deiterding, “Lecture notes on conduction heat transfer,” November 2019.

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

[11] R. Deiterding, “Lecture notes on convective heat transfer,” November 2019.

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

[13] pypi. pyopencl 2019.1.2.