Development of an improved mixing model for an SRM code

Henrik Gustafsson

Spring 2013

Abstract

Consider two gases being pumped into a container of some sort. How will they mix? How will they interact with one another? And how do you model the mixing using a computer?

One way is to first divide the two gases into smaller groups called particles and then let these particles interact with one another at random. Two of the models that is used today is CURL and Modified CURL. They work well and the results are consistent with experimental data. One drawback with them is that neither of them can handle mixing with two particles of different sizes. So here is presented a new model based on CURL and Modified CURL but with the extension of handling particles of different sizes. To test the consistency of these new models simulations are done using DARS, a product of LOGE.

Acknowledgements

I would like to thank everyone at LOGE for giving me this opportunity and my supervisor Karin Fröjd. A special thank goes out to Simon Bjerkborn for answering stupid questions, pushing me and keeping me in good spirit(s).

Contents

Abbreviations

CAD Crank Angle Degree.

MDF Mass Density Function.

PDE Partial Differential Equation.

SI Spark-Ignition.

SOI Start Of Injection.

SRM Stochastic Reactor Model.

TDC Top Dead Centre.

Nomenclature

C_Φ Proportionality constant.

F The MDF.

P Pressure $[Nm^{-2}]$.

R Ideal gas constant $\approx 8.314[Jmol^{-1}K^{-1}]$.

T Temperature $[K]$.

V Volume $[m^3]$.

Φ Vector of random variables.

Ψ Realization variable.

β CURL mixing model constant.

τ Characteristic time for mixing $[s]$.

* Superscript indicating new particle.

$_A$ Subscript indicating particle A.

$_B$ Subscript indicating particle B.

c Concentration as mass fraction [:].

h Mass specific enthalpy $[Jkg^{-1}]$.

m Mass of container $[kg]$.

n Number of particles $[mol]$.

r Size ratio model [:].

r_m Size ratio mixing $0 \leq \frac{m_B}{m_A} \leq 1$ [:].

t Time $[s]$.

Chapter 1

Introduction

The fundamental principle of a combustion engine is a piston moving up and down inside a cylinder, being powered by combustion by fuel. Diesel in a diesel engine and petrol in an SI-engine. A cycle in the engine then happens in the following pattern: the piston starts in the bottom when the volume in the cylinder is at is maximum and air is sucked in to the cylinder, the piston goes up and the pressure rises and as a consequence the temperature also goes up to comply with the ideal gas law, $PV = nRT$. In modern engines fuel is injected right before the piston reaches its maximum, called the top dead centre (TDC) and mixes with the background gas, in older engines fuel was injected together with the background gas. At high pressure and temperature, the fuel/air mixture is ignited. The resulting pressure increase forces the piston downwards, converting chemical energy into rotational movement.

The reason one would like to model this is to understand how a system of chemical kinetic would behave given an initial state. In the engine case this can be used to optimize fuel injection and the design of the cylinder and piston without building a physical model.

To simulate this using a computer the background gas and the injected fuel needs to be discretised into a set number of particles. To weave all the particles together into a continuous system the different particles has to interact with each other. This is done by mixing the particles together. The models that are used today, CURL [1] and Modified CURL [3], gives a good fit with experimental data but has one drawback. The drawback is that both of them require that all particles are of the same size. Here two new models with roots in CURL and Modified CURL will be presented that can handle particles of different sizes. The new mixing models will behave as CURL or Modified CURL when the size ratio between two mixing particles is 1. Optimally there should be no need to retune different parameters when using the new models on cases that has been tuned to fit experimental data with today's models.

The reason why one would like to have a mixing routine that can handle particles of different sizes is computational time. In the engine case the number of particles that make up the background gas could be bigger, and thus fewer, while still maintaining a high discretisation in the injected gas. The total number of particles will thus be lower and the computational costly simulations are done particle by particle and fewer particles will then give faster runtime.

Chapter 2

Theory

2.1 SRM

To model an engine in a computer, it needs to be discretized in a couple of ways. The background gas and the injected fuel needs to be split up into groups called particles. The cylinder could be divided into smaller containers, but in this model a 0-dimensional model is used and thus no discretization of the cylinder, and lastly time. An order of operations is also needed for each time step. This is called operational splitting [2] and gives a series of PDE:s that can be seen in 2.2. Those equations are then solved one by one and the system is updated between each equation. A simplified version of which operations are used and in which order they are executed are seen in the flowchart in figure (2.1). The initial step is how the system looks when the simulation starts, with background gas sucked in to the chamber with a certain pressure and temperature. Then the piston moves a small step and adjustment in volume affecting the system by the ideal gas law takes place. Then the mixing takes place. In the time steps leading up to fuel being injected, the system is rather homogeneous and the only differences is the random process of heat losses to the walls and the following chemistry. Inhomogeneities in the system changes how the system works and the discretizised particles need to interact to model the mixing effect of the gases. After fuel has been injected the fuel mixes out to the background gas in this step. After that chemical reaction occur before heat is lost to the surrounding walls. Then the piston advances one time step. All this is done using LOGE:s SRM-model and code and all implementations of code are done directly into that system.

2.2 The Curl model and the modified Curl

In a paper published in 1963, [1], Curl presents a model of how to mix two fluids in a computer simulation. The method presented in this paper is to divide the the two fluids into smaller particles that are of the same size and contains either one of the two fluids. Then randomly pick two of these particles, put them together into one and then split the resulting particle into two new particles with equal distribution of the content in the first two particles. Then this step is repeated. If m is the mass of one of the two particles and c_A is the concentration of one of the fluids in particle A and c_B is the concentration of the same fluid in particle B then the concentration of that fluid in the new particles become

$$c_A^* = c_B^* = \frac{mc_A + mc_B}{2m} = \frac{c_A + c_B}{2} \tag{2.1}$$

which is the mean of the two particles. This leads to that the number of different concentrations that the particles can have after a certain number of mixing events being exactly $2^n + 1$. An illustration of how these different concentrations occur can be seen in figure 2.3. A simulation of this using 100 particles can be

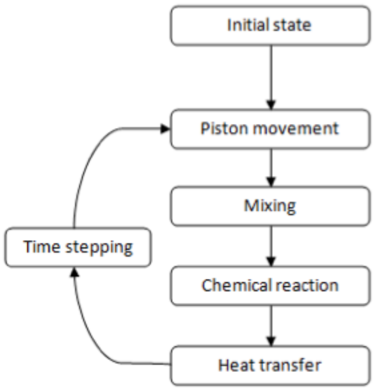


Figure 2.1: *A flowchart of the time steps in the model used, [4].*

$$\underbrace{\frac{\partial}{\partial t}F_\Phi(\Psi,t)}_{\text{Change of MDF}}$$

$$=$$

$$\underbrace{\frac{\partial}{\partial \Psi_{S+1}}\left(V\frac{1}{C_p}\frac{dp}{dt_{\delta V}}F_\Phi(\Psi,t)\right)}_{\text{Piston Movement}}$$

$$+$$

$$\underbrace{\frac{C_\Phi \beta}{\tau}\langle\int_{\delta\Phi}F_\Phi(\Psi-\delta\Psi,t)F_\Phi(\Psi+\delta\Psi)d(\delta\Psi)-F_{\Psi,t}\rangle+\frac{\partial}{\partial \Psi_{S+1}}\left(V\frac{1}{C_p}\frac{dp}{dt_{mix}}F_\Phi(\Psi,t)\right)}_{\text{Mixing}}$$

$$+$$

$$\underbrace{\left(\frac{\partial}{\partial \Psi_{S+1}}\left(\frac{1}{C_p}\sum_{i=1}^{S}h_i\frac{M_i}{\rho}\omega(\Phi)F_\Phi(\Psi,t)\right)-\sum_{i=1}^{S}\frac{\partial}{\partial \Psi_i}\left(\frac{M_i}{\rho}\omega(\Phi)F_\Phi(\Psi,t)\right)\right)+\frac{\partial}{\partial \Psi_{S+1}}\left(V\frac{1}{C_p}\frac{dp}{dt_{chem}}F_\Phi(\Psi,t)\right)}_{\text{Chemical Reaction}}$$

$$+$$

$$\underbrace{\frac{\partial}{\partial \Psi_{S+1}}\left(\frac{h_g A}{Cp}(\Psi_{S+1}-T_W)F_\Phi(\Psi,t)\right)+\frac{\partial}{\partial \Psi_{S+1}}\left(V\frac{1}{C_p}\frac{dp}{dt_{heattransfer}}F_\Phi(\Psi,t)\right)}_{\text{Heat Transfer}}$$

Figure 2.2: The PDE that are solved for each time step in the SRM code. [4]

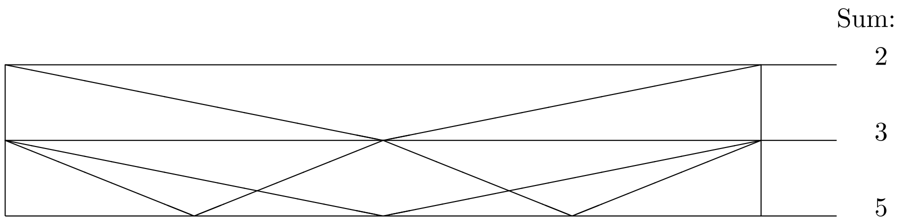


Figure 2.3: *A model showing the first possible concentrations for a particle to have after a few mixing events. It is assumed that there is at least two particles of each fluid for the purpose of this picture.*

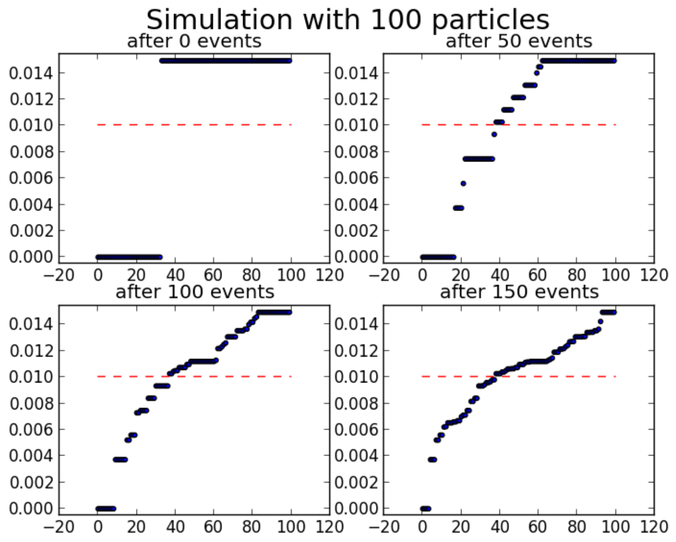


Figure 2.4: *A simulation of the Curl model using 100 particles with a 1:3 relation between the two fluids that are mixed. After 50 mixing events the different discrete states of concentration can be seen clearly and the width of each state also gives a hint about how probable it is to be in that state, in the next plot with 100 mixing events the effect is still visible but it has started to look continuous and in the last plot the effect is only visible at a hand full of points. The y-axes represent the concentration of one fluid in each particle normalized while the x-axes is the just the particle index sorted by concentration.*

seen in figure 2.4 with snapshots after some mixing events have occurred without any chemistry happening between the steps.

Two problems that exists with the standard Curl model is that there is a discrete number of concentrations possible and that every particle that has been mixed always has a twin particle that looks exactly the same. A solution to this was presented by Janicka et al in [3] were the Modified Curl was introduced to the world. The basic idea of the new model is that only a fraction, h, of the involved particles are mixed and this h is a uniformly distributed random number between 0 and 1. The equation for is mixing model then becomes

$$c_A^* = c_A + h(\Phi_c - c_A) \tag{2.2}$$

$$c_B^* = c_B + h(\Phi_c - c_B) \tag{2.3}$$

with $\Phi_c = \dfrac{mc_A + mc_B}{2m}$. Since the particles are not mixing completely the same amount of homogeneity as in CURL is not reached at each event. This needs to be compensated for when calculating how many mixing events have occurred and when there is time to do some chemistry. This compensation can be either

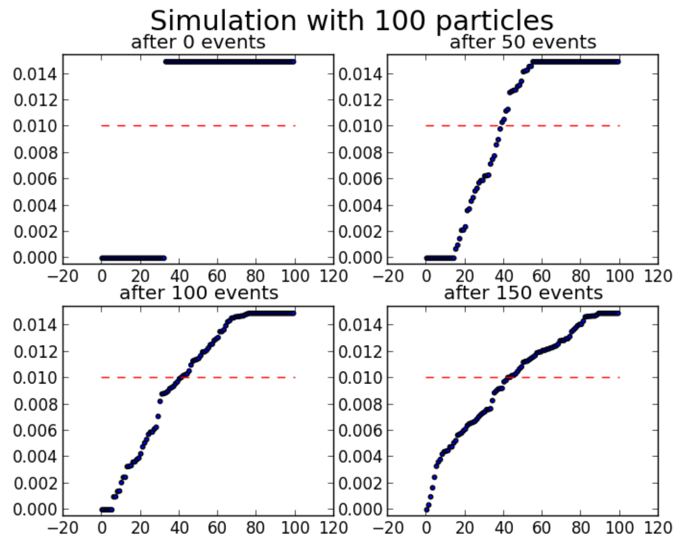


Figure 2.5: *A simulation of the Modified Curl mixing model using the same set up as in figure 2.4 without regarding that less mixing is happening in each loop and instead just counting the number of events. Here the discrete states in the CURL model are not seen due to the incomplete mixing of particles. The y-axes represent the concentration of one fluid in each particle normalized while the x-axes is the just the particle index sorted by concentration.*

to count each mixing event as a fraction equal to h of a standard Curl mixing or to double the number of mixing events, this approach follows from $E[\mathcal{U}(0,1)] = 1/2$. A simulation of modified CURL similar to the one for regular Curl can be seen in figure (2.5) with the lack of discrete levels easily spotted.

2.3 Turbulence

The amount of mixing that happens inside of the engine changes over time and specially it goes up when fuel is injected due to the increase of turbulence that follows the momentum that the injected particles has. This is modelled by the factor τ in the mixing part of 2.2. The dependence of τ regarding number of mixing event is inverse proportional, so when τ grows the number of mixing event goes down. An example of a τ-curve used later in during the simulations are seen in 2.6.

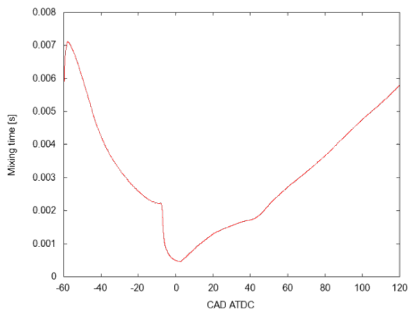


Figure 2.6: *A plot of τ against CAD. The decrease in tau at $-8CAD$ corresponds to the start of injection of fuel into the cylinder. Since the mixing of the gas prior to fuel injection is not of great importance, the tau value at $CAD < -8$ has not been tuned.*

Chapter 3

A weighted model

As stated in the introduction the new weighted model should behave as either CURL or Modified CURL when the ratio, (defined as $r_m = \dfrac{m_B}{m_A}$ with m_A, m_B being the weight of the mixing particles, A and B with $m_A \geq m_b$ holds), goes to one. For example if a simulation injects fuel that is discretized to particles that are smaller than those of the background gas, then CURL or Modified CURL will happen in the cases when two background particles mixes or when two of the injected fuel particles mixes. As a consequence if the ratio is set to 1 when starting the simulation it will be the same as running today's versions of CURL and Modified CURL but with some more calculations that becomes one. So it is not recommended to do so because of the adding of computation needed and round off errors that might occur.

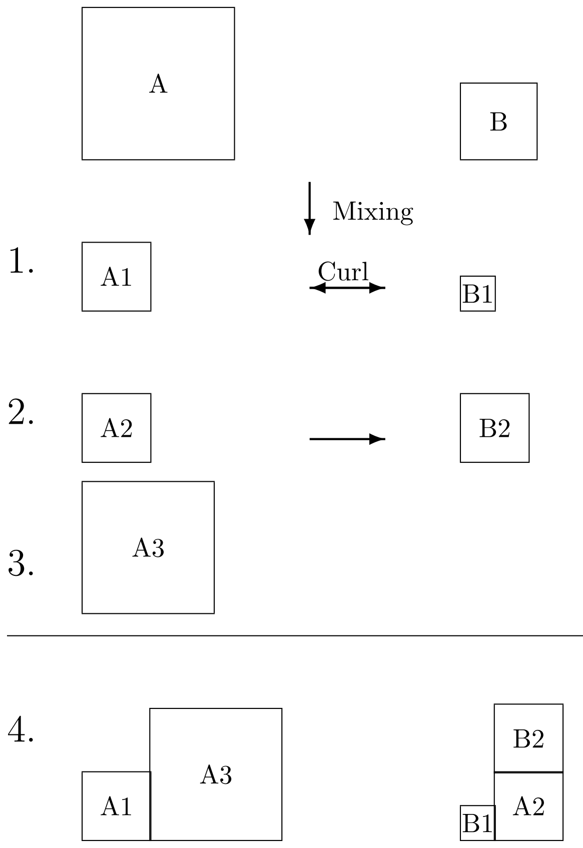
The other extreme, when the smaller particle becomes infinitesimal, or the larger particle goes to infinity, this model goes to that the smaller particle doubles in size by taking a portion of it self from the bigger one. This leads to that the bigger one having the same concentration as before and the smaller one being as in the CURL model before it splits up into two. The idea behind this comes from thermodynamics and the idea of a heat reservoir that is so so relatively large that it is unaffected by contact with the smaller body.

In between these two extremes interpolation happens and to understand it is easiest to split up the process into three parts. The first is the CURL bit, and that will consist of a fraction equal to r_m of the two particles, the second is the moving from particle A to particle B and that is equal to the rest of particle B, or $(1 - r_m)$ then lastly there is a portion of A left that will be left untouched. A flowchart with the steps and pictures is located below in figure 3.1 with the resulting equations at each step and a visualization of the particles masses split into a number of parts.

The modified CURL version of this model follows the main theory of the modified CURL and simply just letting a fraction of each particle go into the mixing. A simulation of these two new models that have been described here can be seen in figure 2.5 and 3.3

For these models to be consistent with other models there is need for a way to measure the impact of each mixing event. The difference between the two is the same as between CURL and Modified CURL, that is the weight of each mixing event in the new weighted Modified CURL is a fraction equal to h of the weight that it would have been if the new weighted CURL had been used instead. The weight of each event in weighted CURL is set to be the mass of the total except the non-mixing part $A3$ in figure 3.1.

Regarding the probability for particles to be selected at each mixing event is not uniform over all particles but rather it is weighted with the size of each particle. This is to ensure that the fuel from the injected particles are distributed into the system with the same speed as in today's models. The only downside with this is that a mixing of two background particles is as heavily weighted as r mixing events between two injected particles and $\frac{r^2-r+2}{2}$ time as heavy as a mixing between a background particle and an injected particle. Here r is the size ratio that is set by the user for the simulation and is the ratio between the background particles and the injected particles. Since the probability to chose either a background particle or an injected one is the same as in CURL and Modified CURL it might become a problem that the mixing events between two background gas particles are weighted too much. This might just be a small problem



$$c^*_{A1} = c^*_{B1} = \frac{m_A r_m c_A + m_B r_m c_B}{m_A r_m + m_B r}$$

$$m^*_{A1} = m_A r_m,\ m^*_{B1} = m_B r_m$$

$$c^*_{B2} = (\frac{c_A + c_B}{2})$$

$$m^*_{A2} = m^*_{B2} = m_B(1 - r_m)$$

$$c^*_{A3} = c_A,\ m^*_{A3} = m_A r_m - m_B(1 - r_m)$$

$$c^*_A = \frac{c^*_{A1} m^*_{A1} + c^*_{A3} m^*_{A3}}{m^*_{A1} + m^*_{A3}}$$

$$c^*_B = \frac{c^*_{B1} m_B r_m + 2c^*_{B2} m_B(1 - r_m)}{m_B(2 - r_m)}$$

$$m^*_A = m_A - m_B r_m,\ m^*_B = m_B(2 - r_m)$$

Figure 3.1: *A scheme over how the new mixing model works with the splitting and combining to two new containers. If the ratio, r_m is 1, that is that the containers are of the same size, only step one is left and the mixing becomes a regular Curl-mixing. Bellow follows an explanation to each step.*

1. A fraction of each container, A1 and B1, equal to the ratio, r is mixed using the Curl method weighted with the size of the fraction and no mass is moved between the two containers, just concentration.
2. A piece with the same size as the remaining fraction of the smaller container, A2 and B2, is moved over from the bigger. Since they have the same size the concentration of them together is going to be the mean of the two particles.
3. What remains of bigger container, A3, is left untouched.
4. The pieces are then put together and a new mass fraction is calculated for the two containers.

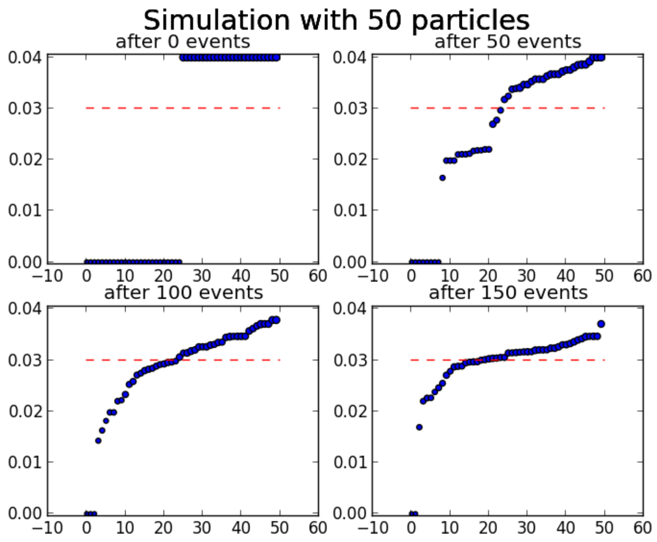


Figure 3.2: *The same plot as in figure 2.4 and 2.5 but using the new weighted model instead. Here there is just 50 particles to counter that each small particle is only half the size of the bigger particles. The probability to choose a smaller particle or choosing a bigger particle is, with these changes in size and number, the same as in the simulations with CURL and modified CURL. That is the total normalized mass of the smaller particles is the same in this case as in the old case. The size of each point corresponds to the weight of that particle. There exists some discrete states as in figure 2.4 but since the mixing particles do not end up the same this feature is not as prominent in this case. As in the other plots the x-axis is the particle index and on the y-axes the corresponding concentration.*

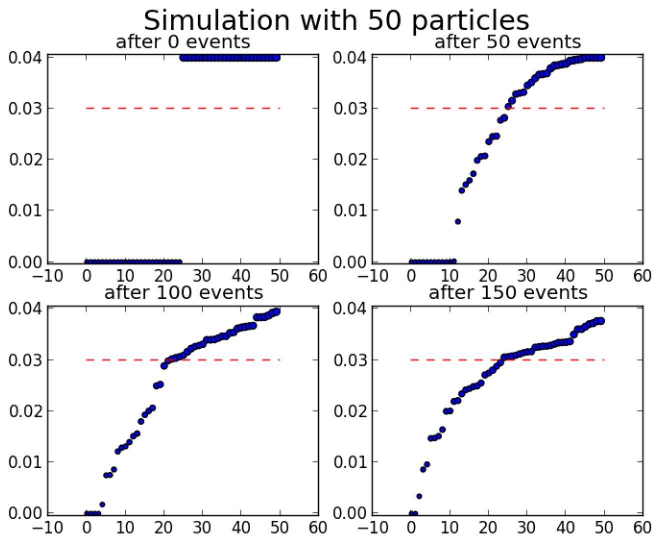


Figure 3.3: *The same set up as in 3.2 but with the new modified CURL instead. Here, as one would expect, there are no discrete lines. The axes are the same as in the other three plots.*

and can be fixed by increasing the turbulence and thus the amount of mixing taking place.

Chapter 4

Results & Outlook

4.1 Performance

The first simulations that were made indicated that the two new models worked and it was possible to go down to a size ratio of four while still giving similar results as the standard mixing model. But these simulations were based on confidential data and thus cannot be presented here. The reason that those simulations worked so good was because they were very insensitive to discretization. As a result of that once could go down to rather few particles in today's CURL and still get a decent result. Because of that a new test case were set up and sadly the new case gave not nearly as good results as the ones before.

There is mainly three things that are off with these results as seen in figures 4.3 and 4.4 compared to figures 4.1 and 4.2; firstly the big difference in maximum pressure and temperature, secondly the difference in CAD when the engine ignites and lastly there is a dip in temperature when the fuel is injected. The last one is easiest spotted with 50 particles in figure 4.4. The last two is somewhat related since the rather small dip temperature might be just enough for the fuel not to ignite when it is supposed to. Instead it needs more fuel or a higher pressure to achieve the sweet spot where it ignites. These problem can also be linked to the first one since the local extreme value happens at a later CAD the piston has moved away and the volume is larger and to comply with the ideal gas law the two parameters, pressure and temperature, becomes smaller.

Regarding computation time and to get numbers that are comparable to each other there is a small problem with modern cpus and that is called hyperthreading and turbo. These together leads to that if the computer uses all of the logical cores it is slower than if it just uses the physical cores. To minimize the problem caused by this when timing the simulations a maximum of simulations equal to the number of physical cores were run simultaneously, otherwise a speed up could have happened when the faster simulations were completed and the cpu went in to turbo mode. So with this in mind when running the simulations the result of gain in computation time can be seen in figures 4.5 and 4.6. The benefit of going down in number of particles is big, it is almost linear with the number of particles so to have a good model that runs on fewer particles would significantly lower the computation time. There is a bigger variance in the case with 100 particles in today's model but that might just be a runtime interruption because when looking at the raw, it was found that all the slowest cycles were consecutive

4.2 Tuning

One possible cause of the problem with the engine igniting to late could be that either to much or to little mixing is happening. This arises from that particle A is assumed to be homogeneous after mixing and that the internal mixing between $A1$ and $A3$ leading to the resulting A is not accounted for. That is to say that the calculation of the weight of each mixing event is done wrongly. In figure 4.8 the problem is illustrated. So to fix this one could add some fraction of $A3$ to the weight of each mixing event, but to see if this would give the appropriate result a change in the number of mixing events that are happening at each time step

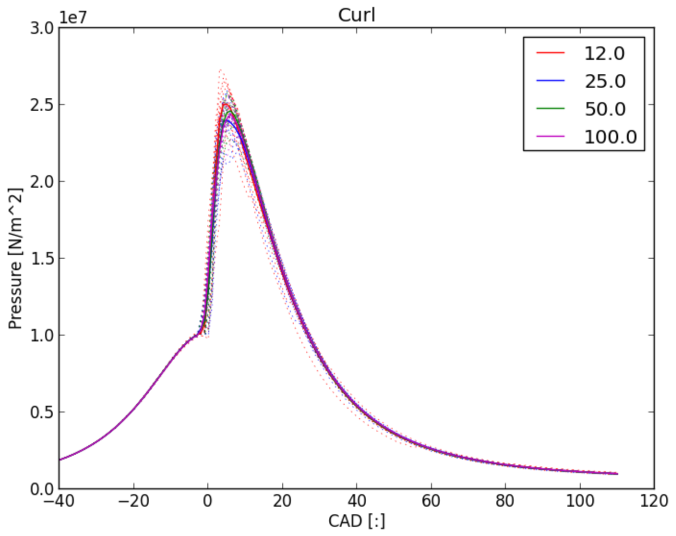


Figure 4.1: *A simulation of the new test case using CURL. Here the pressure is plotted as a function of time steps in CAD. Each colour is a different number of particles and the dotted lines are one motor cycle while the line is the average of multiple runs, here this is over 11 motor cycles. It looks good for 100 and for 50 particles but when the number of particles are pushed down to 25 and 12 the result becomes worse, which was suspected. The peak in maximum pressure happens a little earlier and the pressure has a lower maximum in the case with 25 particles and is higher with 12 particles. As will be seen in figure (should there be a reference here?) the variance in between runs are also higher when the number of particles drop.*

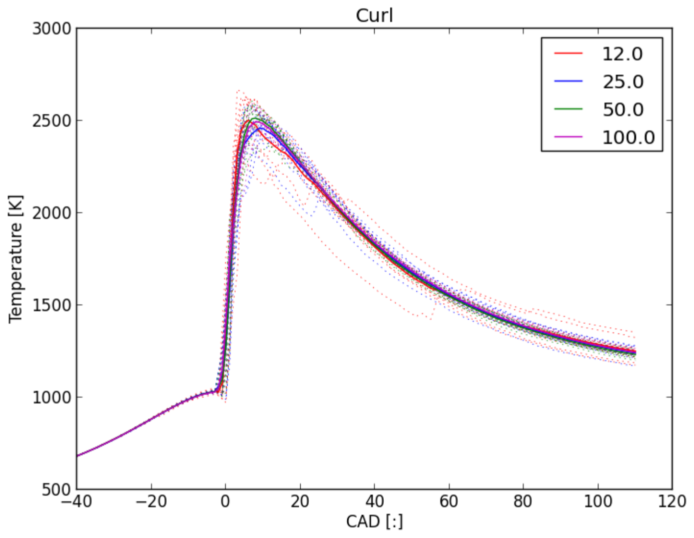


Figure 4.2: *The corresponding temperature curve to figure 4.1.*

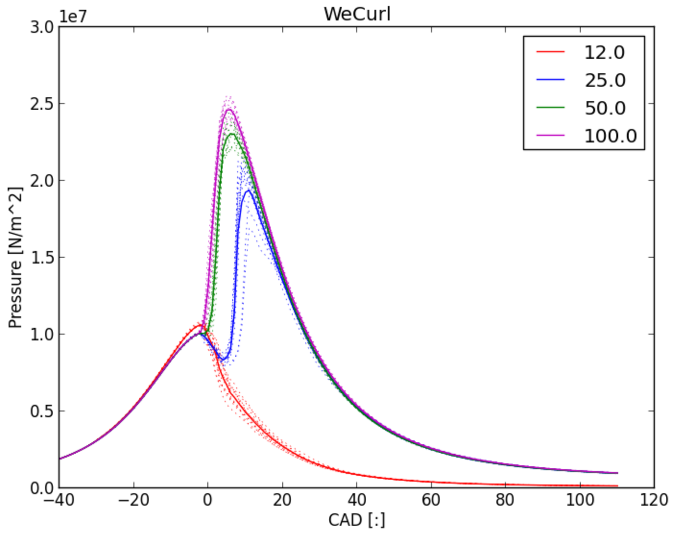


Figure 4.3: *This is the same test case as in figure 4.1 but using the new presented mixing model instead. For each halving of the number of particles the size ratio have been doubled to maintain the same size of the injected particles. The case with 100 particles is set to be the standard with a size ratio of 1 and from that follows that it in practice uses CURL as a mixing model. It is easy to see that none of the others are working well. Both the cases with 50 and 25 particles ignite late and as a result they do not reach as high in pressure. For the case with 12 particles the simulation does not ignite at all. Although there is a rise in pressure before the others and that can be traced back to that some outgoing gas is pumped back into the system and there is some fuel left that can burn.*

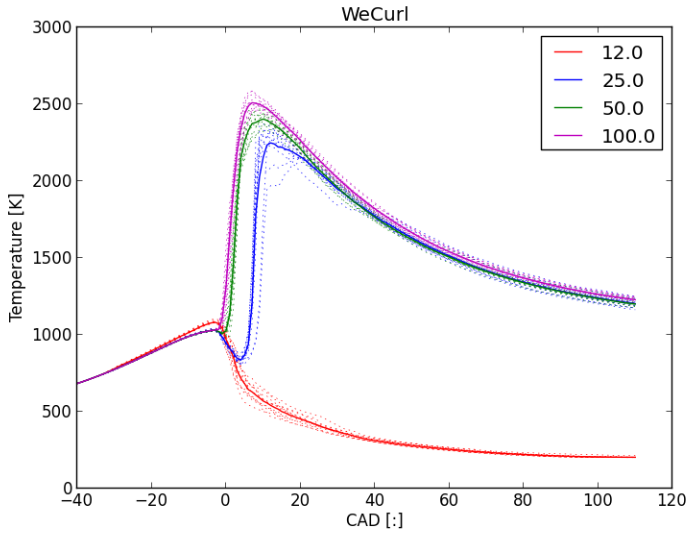


Figure 4.4: *The corresponding temperature curve to figure 4.3.*

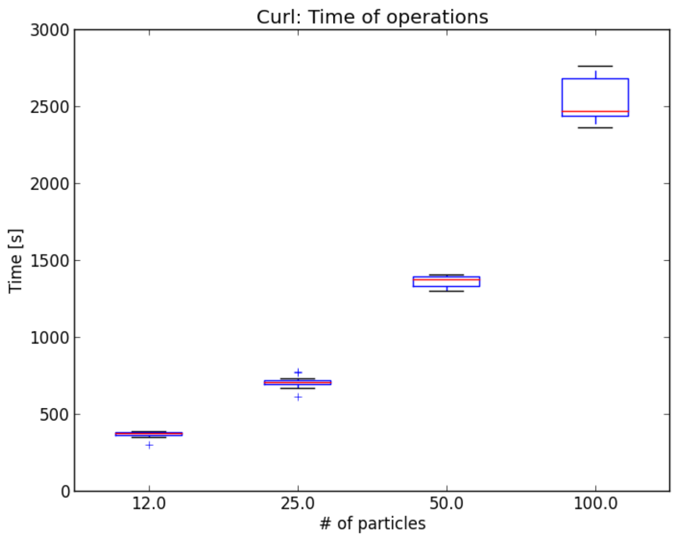


Figure 4.5: *The execution time for one motor cycle running with today's CURL model.*

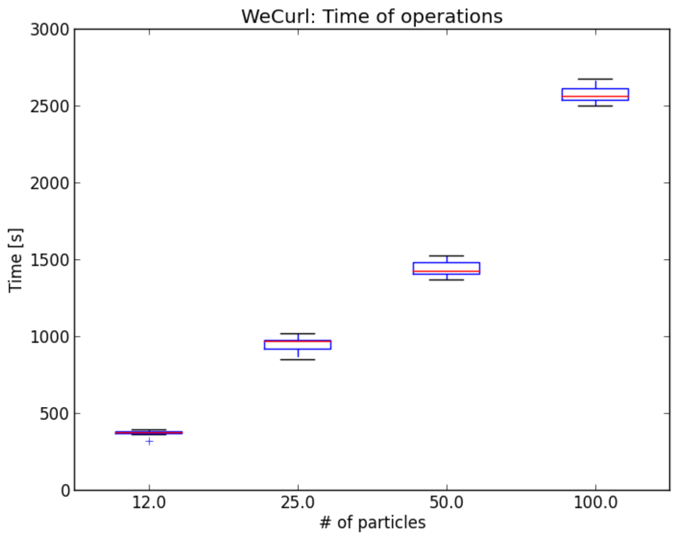


Figure 4.6: *Execution time for one motor cycle running the weighted CURL model. Since the number of computations that are added in the new model are rather few and all of them are simple manipulations with scalars, the result is the same as in figure 4.5.*

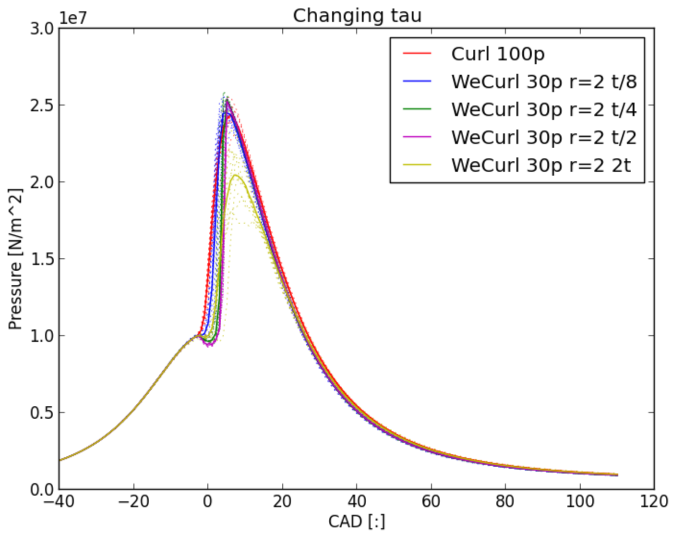


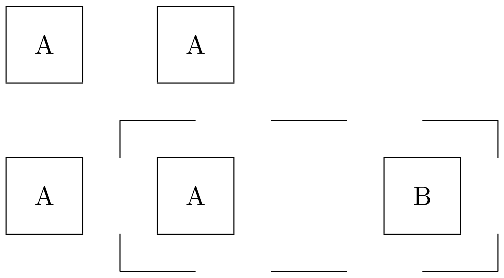
Figure 4.7: *Pressure curve of simulations of Weighted CURL with different τ curves. t/n indicates that each value of τ has been divided by n.*

and not have to fiddle with the code. The result of such a change in numbers of events are seen in figure 4.7 were the number of events have been changed in both directions. This gives an over all improvement but at the cost of a lot more mixing taking place. The time of ignition is still not perfect and neither is the overall fit. But with a huge increase in mixing taking place one could get a working model that would require tuning, which is not what is wanted.

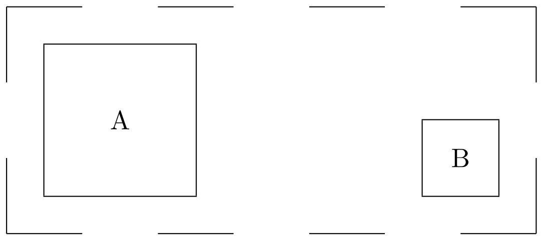
The time of ignition is just off by a few CAD so the problem could be that the time step is too large, it is set to 1 CAD in the simulations so a smaller time step might give an earlier ignition time. The result of testing this with a time step of 0.2 is seen in figure 4.9. Although this gives a better result it does not give as much of an improvement that is needed and it is a tuning factor that should not be needed in the best case, even tough shorter time step often gives a faster simulation. So this way of solving the problem can be ruled out.

The problem is still there and at the moment the model is not working perfectly but an idea that would probably fix things has to do with discretion and that when mixing both the bigger particle and the smaller particle misses the sweet spot of ignition. When a particle that has ignited is mixed with a background particle the resulting bigger particle is to big with to little fuel so it burns out.

There is still some hope left that the Weighted Modified CURL version will perform better. The result of this is seen in figure 4.10 and is to be compared with figure 4.11 that is the same simulation but with today's modified CURL. Since going down to 12 particles and a size ratio of 8 was far off to even ignite that simulation is ignored in these tests. As seen in the figures it performs much better for 50 particles with a size ratio of 2 but the problem with late ignition is still there. For a size ratio of 4 and with 25 particles also performs better but neither of the two cases are good enough. So to conclude the Weighted Modified CURL with its not perfect mixing works better than Weighted CURL but none of them works perfectly.



The mixing that would happen if the containers were of the same size



Mixing that is happening with different sized particles.

Figure 4.8: *This is an illustration of the mixing that happens if the size ratio is set to 4. In the top is how it would be in today's model, that is that one of particle B mixes with one of particle A. Beneath is how it happens in the weighted model, one particle B mixes with a four times larger particle A and both are set to be internally homogeneous after the mixing has occurred. But the mixing that happens inside of the big A is unaccounted for and might be as large as 3 mixing events of today's model.*

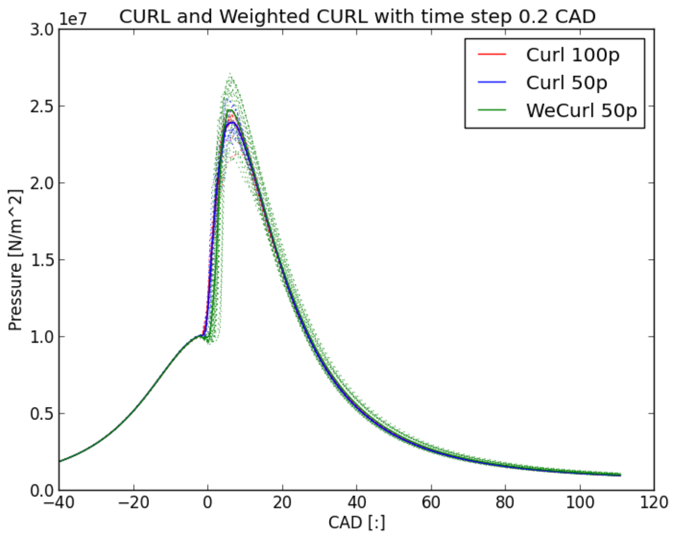


Figure 4.9: *A simulation of CURL and Weighted CURL with a reduced time step from* $1CAD$ *to* $0.2CAD$. *As seen the problem with time of ignition is still there, but there is an overall better result to fit.*

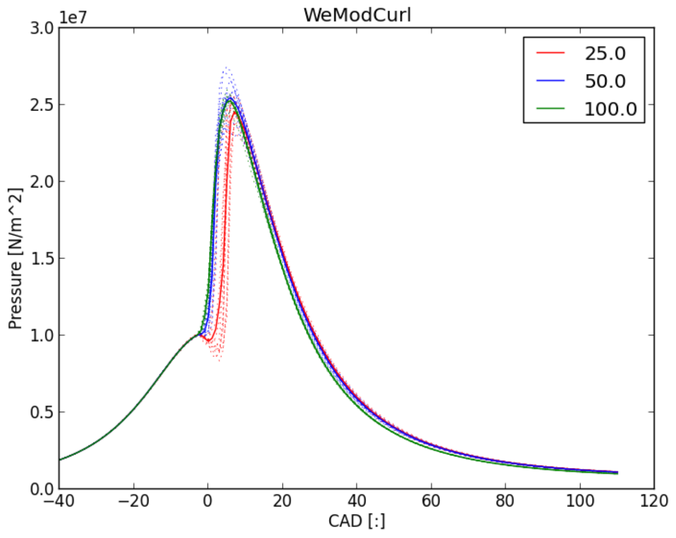


Figure 4.10: *The pressure curve of a simulation of Weighted Modified CURL for different particle sizes.*

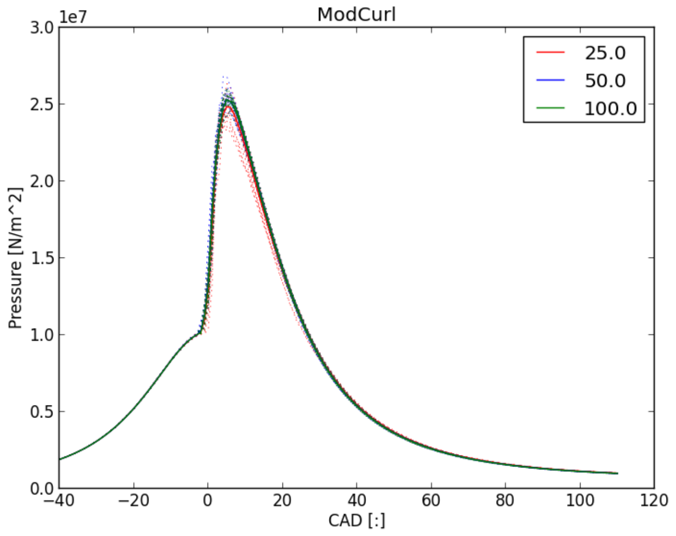


Figure 4.11: *The pressure curve of a simulation of Modified CURL for different particle sizes.*

4.3 Outlook

So what needs to be changed to make it work? One possible solution is to some degree separate $A3$ from the resulting particle A. That is from two particles that goes in to the mixing three comes out, we'll not always get a new particle but a larger fraction of A_3 goes to form a new particle with other fractions of A_3 from other mixing events and when they become sufficiently large, maybe in the size of the background gas before fuel was injected, they pop out as a new particle and the coming fractions of A_3 would form a second particle and so forth. This will lead to some mixing taking place inside of the creation of new particles but that factor should be the same in all cases and can thus be tuned in once. The great advantage with doing this is that the resolution is better and the particles that have the right fuel to air ratio to burn is larger. The disadvantage is that it might be slower since the number of particles will grow, but the speed up from before the fuel is injected might be large enough to still give a significant gain in overall computing time. This fix has not yet to be implemented but is the way to go to end up with a model that is as consistent with other models as possible.

Bibliography

[1] Curl, R. L. Dispersed phase mixing: I. theory and effects in simple reactors. *AIChE Journal 9*, 2 (1963), 175–181.

[2] Hundsdorfer, W., and Verwer, J. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations.* Springer Series in Computational Mathematics. Springer, 2003.

[3] Janicka, J., Kolbe, W., and Kollmann, W. Closure of the transport equation for the probability density function of turbulent scalar fields. *Journal of Non-Equilibrium Thermodynamics 4* (1979), 47–66.

[4] LOGE. Manual - book 3, engine in-cylinder reactor models, 2012.

Appendix A

Python code

Here follows at simple python code that implements the four mixing models that was mentioned in the text. This piece of code, with some modification for plotting, was used to make the plots in figure 2.4, 2.5, 3.2 and 3.3.

```
import numpy as np

% Define a function for each of the four mixing models all taking the input in the same way
% and returning it the same way.
def curl(x1,x2):
    return [[(x1[0]*x1[1] + x2[0]*x2[1])/(x1[1] + x2[1]),x1[1]],
             [(x1[0]*x1[1] + x2[0]*x2[1])/(x1[1] + x2[1]),x2[1]]]

def modcurl(x1,x2):
    a = random()
    return [[x1[0] + a*((x1[0]*x1[1] + x2[0]*x2[1])/(x1[1] + x2[1]) - x1[0]),x1[1]] ,
             [x2[0] + a*((x1[0]*x1[1] + x2[0]*x2[1])/(x1[1] + x2[1]) - x2[0]), x2[1]]]

def weicurl(x1,x2):
    r = x2[1] / x1[1]
    c_curl = (x1[0]*x1[1] + x2[0]*x2[1])/(x1[1]+x2[1])
    phi_1 = (c_curl*x1[1]*r + x1[0]*(x1[1]*(1-r)-x2[1]*(1-r))) / (x1[1]-(1-r)*x2[1])
    phi_2 = (c_curl*x2[1]*r + x2[1]*(x1[0]-x2[0])*(1-r)) / (x1[1]+(1-r)*x2[1])
    return [[phi_1, x1[1] - (1-r)*x2[1] ] ,
             [phi_2, x2[1] + (1-r)*x2[1]]]

def weimodcurl(x1,x2):
    a = random()
    r = x2[1] / x1[1]
    c_curl = (x1[0]*x1[1] + x2[0]*x2[1])/(x1[1]+x2[1])
    phi_1 = (c_curl*x1[1]*r + x1[0]*(x1[1]*(1-r)-x2[1]*(1-r))) / (x1[1]-(1-r)*x2[1])
    phi_2 = (c_curl*x2[1]*r + x2[1]*(x1[0]-x2[0])*(1-r)) / (x1[1]+(1-r)*x2[1])
    return [[ x1[0] + a*(phi_1-x1[0]), x1[1] - x2[1]*(1-r)*a ],
             [x2[0] + a*(phi_2 -x2[0]), x2[1] + x2[1]*(1-r)*a ]]

% This function together with the next picks two indices at random using the size of the
% container as weight and checking so that the two indices are not the same
def drawone(X):
    a = random()
    for i in xrange(size(X)//2):
        if a < sum(X[1,:i]):
            break
    return i

def draw(X):
    i = drawone(X)
    j = drawone(X)
    while i==j:
        j = drawone(X)
    if i>j:
        return i,j
    else:
        return j,i

% The main function that set up matrix with the different containers and normalize the weight.
% After that is done it loops over the matrix a fixed number of times and in each loop it
% picks two containers using the draw function and then runs them through the one of the
% mixing models.
def mix(mixing, npa=100, sizeratio=2, npratio=5, last=150):
    """
    mixing: 1 - Curl
            2 - Modified Curl
            3 - Weighted Curl
            4 - Weighted Modified Curl
    """

    if mixing== 1:
        mixa = curl
    elif mixing == 2:
        mixa = modcurl
    elif mixing == 3:
        mixa = weicurl
    elif mixing == 4:
        mixa = weimodcurl
    else:
        print ""
        return
    X = np.zeros([2, npa])
    X[0,npa // npratio:] = 1
    X[0,:]=X[0,:]/sum(X[0,:])

    if mixing <= 2:
        X[1,:] = 1
        X[1,:] = X[1,:]/sum(X[1,:])
    else:
        X[1,npa//npratio:]=1
        X[1,:npa//npratio]=1/sizeratio
        X[1,:] = X[1,:]/sum(X[1,:])
```

```
for k in xrange(last+1):
    i,j = draw(X)
    if X[1,i] > X[1,j]:
        [X[:,i], X[:,j]] = mixa(X[:,i], X[:,j])
    else:
        [X[:,j], X[:,i]] = mixa(X[:,j], X[:,i])
return X
```