AGNES documentation

Rishikesh Sampat

April 6, 2022

Abstract

Chapter 1

Features

1.1 Introduction

This is the official documentation of AGNES: Automatic Generation of Networks for Emission Simulation.

1.2 Features

- Read ANSYS FLUENT CFD output file.
- Clustering of cells based on pre-defined clustering criteria to form reactor centers.
- Chemical Reactor Network generated based defined cluster centers and their interconnectivity obtained form th CFD file.
- The Solver is capable of solving this CRN either with Energy turned off or on.
- The result is a distribution of temperature and species throughout the domain. The result may also be extracted as one projected on planes cut from the original domain. The Post processing also provides line plots in different locations. One can also track the reaction rates of different NOx production mechanisms.

1.3 Clustering

The breadth first search algorithm is used to search for similar , interconnected cells, to form clusters in a time efficient manner. The clustering criteria can be pre-specified and may be a combination of parameters present in the original source file. The process also keeps a record of mass flow rates of the aggregated cells and their neighbours both in the clustered as well as the original domain. This is further used while generating the CRN.

1.4 Transport equation elements

The CRN by itself is a step towards lower flow physics fidelity in order to free computer capacity to enhance chemistry calculation. Thus several flow features need to be explicitly modelled.

- Advection: this is accounted for by the mas flows between reactors. The initial values taken from the source files may not lead to a mass conserved system and hence this needs to be corrected for. This is done by using the original values to determine the distribution of ratios of outflows to different reactors from a reactor. Finally based on the boundary conditions a matrix equation is solved that gives mass flows leading to a mass balanced system.
- Diffusion: Mass diffusion is accounted for in the global solver equations. A mixture averaged species diffusion coefficient is used. This value is calculated from a Cantera gas object associated with an individual reactor. Thermal diffusion is accounted for by modelling a Cantera Wall object between reactors that allows heat transfer based on conduction.
- Turbulence: Turbulence can be included but only in terms of residence time of reactor. A future improvement would be to include a PaSR model to account for fluctuations in scalars.

1.5 Solver

The global solver has a Newton solver and a time-stepping ODE integration. The time stepping is done by the BDF method.

1.6 Software Layout

AGNES 2.0 has an Object Oriented layout. This makes it easier to introduce new modules and updates without having to re-write a large portion of the code.

Chapter 2

Solver

2.1 Conservation equation

2.2 Newton Solver

The Newton Solver calls the scipy.linalg.spsolve() function. SciPy solves systems of equations using LAPACK which is a mathematics library written in FORTRAN. For a generic matrix the documentation for scipy.linalg.solve() states that a DGESV function in LAPACK is called. This function implements the LU decomposition with partial pivoting and row interchange is used to factorize a matrix.

$$AX = B \tag{2.1}$$

In the above equation A is factorized such that A=P*L*U, where P is a permutation matrix, L is a lower triangle matrix and U is an upper triangular matrix. The resulting system is solved by forward and backward substitution to yield X.

2.3 Stopping criteria

The stopping criteria is based on an error(ϵ), rate of change of error(ϵ_ω) and total rate of change of conserved equation(g(ω))

$$\epsilon = \frac{Y_2 - Y_1}{Y_1} \tag{2.2}$$

$$\epsilon_\omega = \frac{\epsilon_2 - \epsilon_1}{\epsilon_1} \tag{2.3}$$

2.4 Mass Imbalance

To satisfy conservation of mass in a CFD simulation, the mass flowing into the cell should be equal to the mass flowing out. In practice, the cells from the CFD simulation have a certain amount of mass imbalance, i.e the total inflow and outflow mass flow rates associated to a cell have a difference equal to a certain tolerance set in the CFD solver which is accepted to be a converged solution as even the smallest difference would be equal to machine precision and not exactly zero. When several such cells are agglomerated, the mass imbalance adds up and the resultant reactor has a relatively large mass imbalance. In order to be able to solve the reactor, the mass flows need to be corrected. If each reactor is corrected individually, it would not ensure the conservation of mass throughout the network, and while creating the code, it was seen that such an approach lead to an imbalance in major species obtained at the exhaust of the combustor from the CRN when compared to the CFD solution, which should not be the case as the preliminary assumption includes the correct prediction of major species by CFD.

A better way of solving this problem is to correct the mass flows of all the reactors simultaneously. A system of equations is setup, where the independent variables are the total outflow from the reactors. The total mass flowing out from a reactor may be flowing into several neighbouring reactors. Hence, the inflow into a reactor is a fraction of the total outflow from the neighbouring reactor and this fraction is calculated from the mass flow of the original CFD solution and kept constant. The equation set up is a balance of total inflow and outflow of a reactor, for a system of N number of reactors in Equation 2.4, the matrix form of which is shown in Equation

2.5.

$$M_k - \sum_{j \neq k}^{N} \alpha_{kj} M_j = \sum f_k \tag{2.4}$$

$$\begin{bmatrix} \alpha_{11} & -\alpha_{12} & -\alpha_{13} & -\alpha_{14} & .. & .. \\ -\alpha_{21} & \alpha_{22} & -\alpha_{23} & .. & .. & .. \\ -\alpha_{31} & -\alpha_{32} & \alpha_{33} & .. & .. & .. \\ : & : & : & .. & .. & .. \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ : \\ : \end{bmatrix} = \begin{bmatrix} \sum f_1 \\ \sum f_2 \\ : \\ : \end{bmatrix} \tag{2.5}$$

where M_k is the total outflow from k^{th} reactor, M_j is the total outflow from j^{th} reactor, α is the fraction of outflow from j^{th} reactor flowing into k^{th} reactor and f_k is any other source of mass inflow. As can be seen, all mass flows, M, are defined as an outflow from a reactor. This results in a AX=B type system where A is a NxN matrix and B is a vector of source terms. In the current code, B contains the boundary inflow mass flow values. The solution to the system is found giving the total outflows as a result. The coefficients from the matrix A are used to assign the correct fractions of the total outflows to the corresponding mass flow controllers. For every mass flow rate there are two corresponding MFCs, one outflow MFC in the PSR model of the current reactor and its associated inflow MFC in the PSR model of the adjacent reactor, due to the decoupled method of solving the CRN as explained earlier.

The pseudo code for mass flow correction is mentioned in Algorithm 1. It shows the process of calculating the inlet mass flow fractions, solving the system of equations and finally correcting the mass flows in each mass flow controller.

Algorithm 1 Mass Imbalance correction

```
coeffmat=null square matrix; diagonal size=number of PSR
f: boundary condition mass flows. Inflow is positive
while Creation of MFC from r1 to r2 do
    coeffmat[r1][r1]+=mflow
    coeffmat[r2][r1]-=mflow
for p from 0 to Number of PSR-1 do
    factor=coeffmat[p][p]
    for q from 0 to Number of PSR-1 do
        coeffmat[q][p]=coeffmat[q][p]/factor
A=coeffmat
B=f
function SOLVE(A,B)
    AX=B
    return X
Corrected Mass flows=X
for All MFCs do
    mass flow mfc=abs(coeffmat[from reactor][to reactor])*X[from reactor]
```

2.4.1 Global Solver

The global solver comprises of a Newton solver and a Time stepping integrator. The governing equations were explicitly coded in. The equations are as follows:

$$g(\omega) = -C\omega + f + R(\omega) \tag{2.6}$$

where $g(\omega)$ is the net rate of production of mass of species in a reactor, C is a $N_E \times N_E$ sparse matrix accounting for convection terms, ω is a vector of mass fractions of species in all reactors, f is a vector accounting for mass flows from external environment, i.e boundary conditions and R is a vector accounting for production terms due to local reactions within the PSR. $N_E = N_C \times N_R$, where N_C is the number of species and N_R is the number of reactors. The Newton method, also known as the Newton-Raphson method is a root finding algorithm which finds a 'y' such that 'f(y)=0'. The general Newton-Raphson Equation for a single variable is

$$y_{i+1} = y_i - \frac{f_i}{f_i'} \tag{2.7}$$

where f_i is the function value at the i^{th} iteration and $f_i' = \frac{df}{dy}$ at the i^{th} iteration. The objective of the Newton Raphson method is to find the solution to f(y)=0, i.e the solution is reached when the function value, f, is zero.

This method is extended to a multivariate system by replacing the variables with vectors and the derivative by a Jacobian matrix,

$$\mathbf{y}_{i+1} = \mathbf{y}_i - \frac{\mathbf{f}_i}{J_i} \tag{2.8}$$

This can be re-written as

$$J_i \Delta \mathbf{y} = -\mathbf{f} \tag{2.9}$$

This is an $AX = B$ type of equation which can be easily solved using either direct or iterative solvers from the SciPy package. The aim of the Newton iterations is to obtain a ω for which $g(\omega) = 0$.

$$J_i \Delta \omega = -\mathbf{g} \tag{2.10}$$

The time integration is performed on a modified form of the governing equations. By definition,

$$m_k \frac{d\omega_{ki}}{dt} = g_{ki} \tag{2.11}$$

This equation is discretized and integrated using the BDF method.

$$\omega_{ki}^{n+1} = \omega_{ki}^{n} + \int \frac{g_{ki}^{n+1}}{m_k} dt \tag{2.12}$$

The BDF method uses implicit iteration, i.e the integral of g over the time step is calculated at n+1 iteration. For the final results, the time integrator was not used as in some small scale tests the implementation was found to be flawed and unreliable. Hence currently the software only uses the Newton's method for global iterations.

One of the limitations of Cantera's reactor network solver was its inability to handle large systems. In the global solver the Jacobian would be very large considering reactors of the order of 10^3 and each reactor has around 53 species, leading to a Jacobian of the size of around 53000×53000 which would lead back to the same memory consumption issue. Hence the Jacobian is expressed as a sparse matrix rather than a dense one and is solved using the sparse solvers in SciPy.

2.4.2 Jacobian calculation for global calculation

The Newton's method involves calculating a Jacobian matrix at the current state. This section describes how it is done. The methods adopted are from [**?**] and are used in KPP. The Jacobian for the entire reactor network can be expressed as a combination of 2 Jacobian matrices, one having a contribution due to the inter-connectivity between neighboring reactors, J_s, and the other being the sensitivity of species production within the reactor species composition, J_w.

$$J = J_s + J_w \tag{2.13}$$

Both, J_s and J_w are expected to be sparse matrices. J_s is sparse because each reactor is connected to only a few other reactors, compared to the total number of reactors, hence it has a structure similar to the one shown in Figure 2.1. Each row represents the contribution of the convection terms of a species conservation equation in the reactor, the reactor being represented as a square with dark borders lying along the diagonal of the matrix. J_w is sparse and concentrated near the diagonal. This is because only the species present within a particular reactor can directly affect the reaction rate, hence the derivatives of the species production with respect to species in other reactors is zero. Usually, a reaction mechanism has more reactions than species.

Numerical v/s analytic jacobian: The numerical way of calculating J_w would be to vary each species and for each variation, calculate the change in production rates of the other species. This would also require the evaluation of all reaction rates(325 reactions in GRI-3.0) for every change. Hence total evaluations would be 53(no. of species in GRI-3.0)*(325+3). The Jacobian may also be calculated analytically by calculating the change of reaction rates with respect to the reactants for each reaction. The reaction rates need only be calculated once. The analytic formula expresses the rate gradient as a function of the current reaction rate, the species concentration and the coefficient of the species in the reaction. As each reaction has 2 to 3 reactants, the number of evaluations are small(≈325*3). This is highly valuable in speeding up the evaluation time especially for large mechanisms, which are expected for complex fuels such as kerosene and gasoline.

The influence of the change in mass fraction of a particular species on another species within a reactor can be expressed as,

$$J_{\dot{\Omega}} = \frac{\partial \dot{\Omega}}{\partial Y} = \frac{\partial \dot{\Omega}}{\partial C} \times \frac{dC}{dY} \tag{2.14}$$

J_w is composed of many $J_{\dot{\Omega}}$ sub matrices, each one belonging a reactor in the network as shown in Figure 2.2. Equation 2.14 splits the derivative into two parts according to the chain rule. The first term relates the change

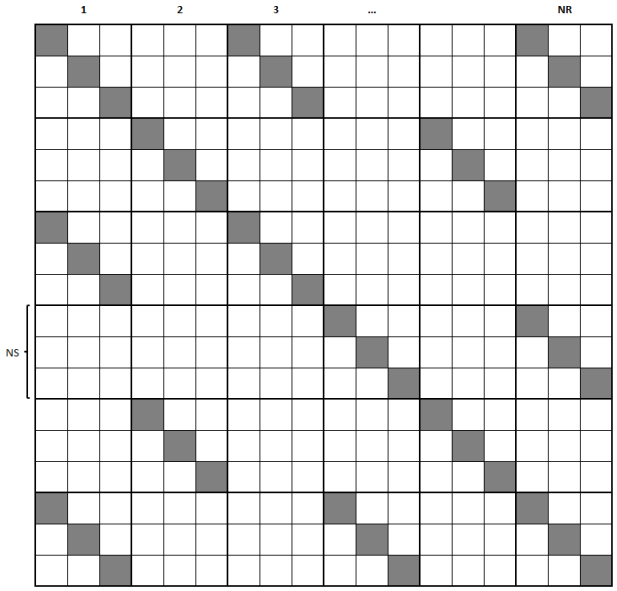

Figure 2.1: Sample structure of Jacobian due to reactor inter-connectivity(J_s), where NS is the number of species and NR is the number of reactors.

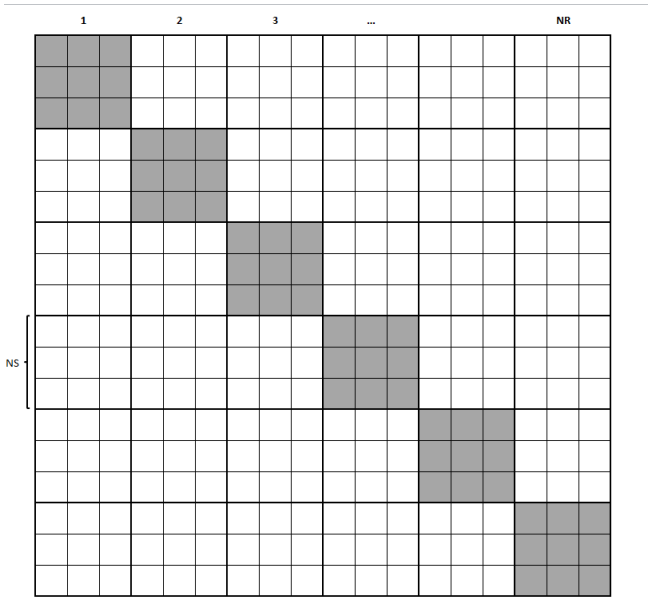

Figure 2.2: Sample structure of Jacobian due to reactions(J_w), where NS is the number of species and NR is the number of reactors.

in species production rates to change in concentrations of all the species and the second term relates change in all the species concentrations to the change in that species' mass fractions.

$$\frac{\partial \dot{\Omega}}{\partial C} = A, \qquad \frac{\partial C}{\partial Y} = \Gamma \tag{2.15}$$

A chemical reaction can be generally expressed as:

$$\nu'_1 C_1 + \nu'_2 C_2 + ...\nu'_{n1} C_{n1} \rightleftharpoons \nu"_1 C_1 + \nu"_2 C_2 + ...\nu"_{n2} C_{n2} \tag{2.16}$$

The net rate of progress of reaction expressed in units of $\frac{kmol}{m^3 - s}$ is shown in Equation 2.17,

$$\begin{aligned} r &= r_f - r_b \\ r &= k' \prod_{i=1}^{n1} c_i^{\nu'_i} - k" \prod_{j=1}^{n2} c_j^{\nu"_j} \end{aligned} \tag{2.17}$$

where k is the rate constant given by Equation 2.18.

$$k = A(T)e^{\frac{E_a}{RT}} \tag{2.18}$$

The net rate of increase of species, $\dot{\Omega}$, involved can be obtained by multiplying 'r' by the stoichiometric coefficient, a constant α, and the volume of the reactor, where $\dot{\Omega}$ has the units of $\frac{kg}{s}$,

$$\begin{aligned} \dot{\Omega}_{ii,r} &= \alpha_i \times \nu_i \times r_i \times MW \times V_r \\ \alpha &= \begin{cases} -1 & \text{if i is a reactant species} \\ 1 & \text{if i is a product species} \end{cases} \end{aligned} \tag{2.19}$$

The derivative of 'r' with respect to a reactant species is of the form,

$$\begin{aligned} \frac{\partial r}{\partial C_\theta} &= \nu'_\theta k' c_\theta^{\nu'_\theta - 1} \prod_{i=1, i\neq\theta}^{n1} c_i^{\nu'_i} = \frac{\nu'_\theta}{c_\theta} k' \prod_{i=1}^{n1} c_i^{\nu'_i} \\ &= \frac{\nu'_\theta}{c_\theta} r_f \end{aligned} \tag{2.20}$$

Similarly, the derivative with respect to a product species is,

$$\begin{aligned} \frac{\partial r}{\partial C_\theta} &= -\nu"_\theta k" c_\theta^{\nu"_\theta - 1} \prod_{i=1, i\neq\theta}^{n2} c_i^{\nu"_i} = -\frac{\nu"_\theta}{c_\theta} k" \prod_{i=1}^{n2} c_i^{\nu"_i} \\ &= -\frac{\nu"_\theta}{c_\theta} r_b \end{aligned} \tag{2.21}$$

'A', in Equation 2.15, is a matrix that can be assumed to be a sum of the changes due to the different reactions(1 to I) occurring, as shown in Equation 2.22, where S_i is the contribution of coefficients related to the i^{th} reaction.

$$A = \frac{\partial \dot{\Omega}}{\partial C} = \sum_{i=1}^{I} S_i \frac{\partial r_i}{\partial C} \tag{2.22}$$

Thus, A has a unit of $\frac{kg - m^3}{kmol - s}$

$$\begin{aligned} C &= \frac{n}{V} \\ C &= \frac{n}{V}\frac{N}{N}\frac{MW_n}{MW_n}\frac{MW_N}{MW_N} = \frac{1}{V}\frac{n \times MW_n}{N \times MW_N}\frac{N \times MW_N}{MW_n} = \frac{1}{V}\frac{m_n}{m_N}\frac{m_N}{MW_n} \\ \therefore C &= \frac{Y_n}{V}\frac{m_N}{MW_n} \end{aligned} \tag{2.23}$$

Differentiating Equation 2.23 by mass fraction of the same species yields,

$$\frac{\partial C}{\partial Y} = \frac{m_N}{V \times MW_n} = \Gamma \tag{2.24}$$

The unit of Γ is $\frac{kg}{m^3 \frac{kg}{kmol}} = \frac{kmol}{m^3}$. This brings the overall Jacobian, $J = A\Gamma$ to the unit of $\frac{kg}{s}$.

The advantage of the analytic Jacobian is that exponential evaluation of the rate constants needs to be done only once at the beginning of the iterations, as the temperatures are kept constant while solving the CRN(isothermal reactors), whereas in the numerical Jacobian, the rate constants need to be re-evaluated for change in every species. This makes it a faster process and the gains are more on a larger mechanism.

Js is the Jacobian due to transport from neighbouring reactors. It represents the rate of change of species mass in a particular reactor with respect to the change in mass fraction of the same species in that or other reactors. Consider the species conservation equation,

$$C_\omega = m\frac{dY_k}{dt} = \sum_{in} \dot{m}_{in}(Y_{k,in} - Y_k) + V\dot{\omega}_k MW_k = S + W \tag{2.25}$$

Differentiating the term for k^{th} species in i^{th} reactor with respect to to Y_{kj}, where j refers to neighboring inlet reactor index,

$$\frac{dC_{ik\omega S}}{Y_{ikj}} = \dot{m}_{in} = \alpha_{ij}\dot{m}_j \tag{2.26}$$

The general convention used in the representation of the governing equations is that all connecting mass flows are specified as those going out of a reactor, so all inlet mass flows are in fact outlet flows from neighboring reactors. α_{ij} is obtained from the coefficient matrix from the mass balance equation.

2.4.3 Convergence Criteria

The convergence is determined based on two residual conditions. The first one ϵ_s is based on the species mass fraction defined as

$$\epsilon_s = \max\Big(\frac{Y_k - Y_{k_{old}}}{\max(Y_{k_{old}})}\Big) \tag{2.27}$$

The second residual is ϵ_r, which is defined as the maximum rate of production of a species among all species present in all reactors. Theoretically, at steady state, rate of production/consumption of all species should be zero and the residual related to change in mass fraction should also be zero as once steady state is achieved there should not be a change between iterations.

ϵ_s should be less than 1e-09 or ϵ_r should be less than 1e-15 for convergence to be achieved. ϵ_s is a relative quantity an hence the lower threshold for convergence. The denominator in this residual is the maximum of all the mass fractions, else the criteria is too strict and takes significantly longer to converge without much benefit in absolute quantities. In this project, minor species pollutants were intended to be studied and are generally expected to be present at levels of the order of parts per million, i.e mole fractions of the order of 1e-06. But at the same time the maximum mass fractions of certain other species such as CO2 can be expected to be of the order of 0.1. If instead of maximum mass fraction, the mass fraction of that particular species is used, the maximum allowable change in mass fraction for convergence to occur is 1e-15(1e-06 $\times$ 1e-09) On the other hand if maximum mass fraction is used, convergence can occur for a maximum difference of 1e-10. This is 1e-04 times less than the expected pollutant levels, which is an acceptable level of accuracy to call it a steady state. Of course the former case is more mathematically correct, but is an unnecessary constraint. The threshold values for convergence were taken based on the default values used for relative and absolute tolerance in Cantera.

Another important consideration for convergence is the reactor model used by the local solver. It was stated earlier that a combination of reservoirs, ideal gas reactors, mass flow controllers and valves is used to represent a PSR. The valves allow for a variable mass flow exchange between reactors, while the mass flow controller represents the fixed exchange. Once the residuals of species change, ϵ_s drop below 1e-06 continuously for 2 iterations, or the maximum mass flow through the valves relative to the fixed balance mass flow into the respective reactor is less than 1e-06, the model is switched to a fixed mass flow one by setting the valve coefficients to zero, effectively making the valves inactive. The aim of the CRN is not to obtain a perfect pressure distribution, but one that has a closer pressure condition to the constant pressure conditions. In the tests conducted on the case of an atmospheric combustion chamber, it was found that such an approach ensured the reactors were close to atmospheric pressure and reduced the convergence time. If there is an active valve present, the only way there is zero mass flow through it is if either the pressure is exactly equal on either side, which is difficult to achieve numerically, or the pressure difference is negative. Even a slight pressure difference will result in some mass flow through the valve and if the final solution converges with the presence of mass flows through valves, that would be physically inconsistent as the actual mass flows obtained from the CFD solver are modeled in the mass flow controllers and any other flows are in excess to that.

Chapter 3

Code

3.1 MassImbalance

The data of massflow rates from CFD would consist of some error, of the order of machine precision per cell, with respect to mass balance of the entire system. When clustering of cells is done to create reactors, this error also adds up and hence if the mass flow rates between reactors is used directly in absolute terms, mass balance of the system cannot be maintained. Hence the relative values are used and the absolute value is determined based on the relative mass flows and boundary conditions. This class extracts the mass flow rates between reactors and assimilates into a matrix equation consisting of a matrix of coefficients and an RHS vector that contains the boundary conditions of the system. The mass_imbalance() method is responsible for the core of this operation. The assembly of the matrix, coeffmat, is done outside of this class and it is passed as a parameter to the method of this class. The diagonal consists of the total outlet massflow and the elements of the row represent inflows from neighbouring reactors. Each row is normalised by it's respective diagonal element to obtain the ratios. Absolute values of the inlet massflows are specified in the RHS vector, whereas the outlet massflow from the system is determined by solving the system of equations.

3.2 CRNSolver

The global solver is called via the globalsolver() and globalodeint() methods. The first one implements a Newton solver and the second one implements a time-integrator. The time integration is done by the BDF method imported from scipy.integrate. This method can accept sparse matrices as Jacobian.

3.3 Data structures

These are data structures as used in ChemicalReactorNetwork.py .

- zone(for face list) = {zone id : [13,first face id, last face id, zone type]... }
- zone(for cell list) = {zone id : [12,first cell id, last cell id]... }
- data_std(data wrt original cfd faces) = {zone id :{property id : [value face0, value face1,..] }... }
- graph = {reactor number: {'adjacent': [r0, r1...], 'faces': [face0, face1..], 'cells':[cell0, cell1...], },..}
- data(data for each reactor) = {zone id : {header index : [val reactor0, val reactor1,...]} ...}
- facearea = { face id : [0, 0 , [LHS cell, RHS cell], area] ...}
- PSR : {reactor number: reactor object, ...}
- data_num = 15
- coeffmat = csc_matrix(): composed of list to express sparse matrix

Zone ids:

- 2 = internal cell
- 20/10 = mass flow inlet/velocity inlet

- 12 = periodic
- 8 = shadow periodic
- 3 = wall
- 36 = outlet

Data accessed from data_std(original dataset) during CRN formation and calculation:

- mass flux transferred through a face(18 is the property id for mass flux)
  mflux = data_std[z][18][face - zone[z][1]]

3.4 Startup

Chapter 4

PIV-CRN

The PIV-CRN processes is handled slightly differently. As we start from a domain of velocity field and spatial coordinates, the mas fluxes are not known. However the inlet mass flux may be defined based on the velocities as the inlet is considered to be of a known composition and gas state. The inter cell velocity flows are also known. Hence the mass flux distribution needs to be updated after every iteration based on changes in density.

The PIV data is exported as a .dat file from DaVis. The file contains 5 main columns of X-coordinate, Y-coordinate, X-velocity, Y-velocity and Validity. The "Validity" takes a value of 0 or 1 for invalid and valid vectors respectively. This can be used to remove masked regions. Currently this data is converted into a matrix and further converted into graphs and data dictionaries that are readable by AGNES. This is largely based on the output formats of casefile.py and datafile.py that read ANSYS Fluent files.

The data is assumed to be arranged in uniformly spaced and sized square cells forming a rectangular region. The boundaries are assumed to be along the four edges.

During the code set up it was noticed that the coefficient matrix that ensure mass balance was set up in a dense format in AGNES. This was restricting the maximum cells acceptable in the CRN. This is changed to be in a sparse format as a typical PIV domain of the combustor contains about 500×500 vectors (250000) which is way more than the 10,000 limit.

4.1 Derivation of governing equations

Reynolds Transport Theorem may be used to determine the governing equations of the reactor network consisting of ideal perfectly stirred reactors. RTT relates conservation equations of a system to a control volume. A system by definition is a arbitrary mass of fixed quantity. This means, the fluxes at the edge of the system are zero. The material derivative is the rate of change of a property of the system. RTT relates the change in a control volume and the fluxes across the volume's surfaces.

In the PIV-CRN process the velocity field is measured and assumed to remain constant during the computation. Assuming steady state conditions, the mass conservation equation can be directly solved to obtain the density field. This density field may be kept fixed during the rest of the computation. Now, only the conservation of species mass and conservation of energy equations need to be solved to evaluate the species mass fractions and temperature.

$$g(\rho) = \frac{\partial \rho V}{\partial t} = -\sum \rho(\overrightarrow{u}.\overrightarrow{n})dA \tag{4.1}$$

$$g(T) = \frac{\partial (mC_vT)}{\partial t} = -\sum \rho_{n,k}C_pT_{n,k}(\overrightarrow{u}_{n,k}.\overrightarrow{n})dA - \sum_{j=1}^{N_{sides}} K\nabla T dA - \sum_{i=1}^{N_{species}} \omega_i V h_{fi} \tag{4.2}$$

$$g(Y_i) = \frac{\partial (mY_i)}{\partial t} = -\sum \rho_{n,k}Y_i(\overrightarrow{u}_{n,k}.\overrightarrow{n})dA - \sum_{j=1}^{N_{sides}} D\nabla Y_i dA - \omega_i V MW_i \tag{4.3}$$

4.2 Jacobian calculation

4.2.1 Energy conservation equation

$$\frac{\partial g}{\partial Y_i} = -\sum_{j=1}^{N} \frac{\partial \omega_j}{\partial Y_i} V MW_j h^o_{fi} \tag{4.4}$$

$$\frac{\partial g}{\partial T_k} = -\rho_{n,k} C_p (\overrightarrow{u}_{n,k}.\overrightarrow{n}) dA - \frac{K}{\delta x}\frac{\partial \delta T}{\partial T_k} dA - \sum_{i=1}^{N} \frac{\partial \omega_i}{\partial T_k} VMW_i h_{fi}^o \tag{4.5}$$

4.2.2 Conservation of species mass

$$\frac{\partial g}{\partial Y_{j,k}} = -\rho_{n,k} (\overrightarrow{u}_{n,k}.\overrightarrow{n}) dA - \rho_{n,k}\frac{D}{\delta x}\frac{\partial \delta Y_i}{\partial Y_{j,k}} dA - VMW_i \frac{\partial \omega_i}{\partial Y_j} \tag{4.6}$$

$$\frac{\partial g}{\partial T_k} = -VMW_i \frac{\partial \omega_i}{\partial T_k} \tag{4.7}$$

4.2.3 Source terms derivatives

$\frac{\partial \omega_i}{\partial Y_j}$ is calculated in an analytical manner as shown in Section 2.4.2.

$$\omega_j = \sum_{i=1}^{I} S_{ij} r_i \tag{4.8}$$

,where $S_{i,j}$ is the contribution of coefficients related to the i^{th} reaction on formation of j^{th} species.

$$\frac{\partial \omega_j}{\partial T} = \sum_{i=1}^{I} S_{ij} \frac{\partial r_i}{\partial T} \tag{4.9}$$

if,

$$k = AT^{\beta} e^{-\frac{E_a}{RT}} \tag{4.10}$$

then,

$$\begin{aligned} \frac{\partial r_i}{\partial T} &= \frac{\partial k'}{\partial T}\prod_{i=1}^{N_1} c_i^{\nu_i'} - \frac{\partial k"}{\partial T}\prod_{j=1}^{N_2} c_j^{\nu_j"} \\ \frac{\partial r_i}{\partial T} &= \Big[\frac{\beta_1}{T} + \frac{E_{a1}}{RT^2}\Big] k' \prod_{i=1}^{N_1} c_i^{\nu_i'} - \Big[\frac{\beta_2}{T} + \frac{E_{a2}}{RT^2}\Big] k" \prod_{j=1}^{N_2} c_j^{\nu_j"} \end{aligned} \tag{4.11}$$

4.3 Calculating reverse reaction coefficients

- The equilibrium constant is the ratio of the forward reaction rate coefficient and the backward reaction rate coefficient. This value changes with temperature due to the change in mole fractions of species.

$$K = \frac{\prod_j X_j^{\nu_j}}{\prod_i X_i^{\nu_i}} \tag{4.12}$$

- Next, for a given temperature, the backward reaction coefficient may be written as:

$$k_r = \frac{k_f}{K} = A_r T^{\beta_r} e^{\frac{-E_a}{RT}} \tag{4.13}$$

  Assuming $\beta = 0$,

$$T \ln \frac{k_f}{K} = T \ln A_r - \frac{E_{ar}}{R} \tag{4.14}$$

  Thus, straight line can be fit to obtain the values of A and E_a.

4.4 Mesh Generation

The PIV data is exported as a .dat file from DaVis. The file contains 5 main columns of X-coordinate, Y-coordinate, X-velocity, Y-velocity and Validity. The "Validity" takes a value of 0 or 1 for invalid and valid vectors respectively. This can be used to remove masked regions. Currently this data is converted into a matrix and further converted into graphs and data dictionaries that are readable by AGNES. This is largely based on the output formats of casefile.py and datafile.py that read ANSYS Fluent files.

The data is assumed to be arranged in uniformly spaced and sized square cells forming a rectangular region. The boundaries are assumed to be along the four edges.

4.5 Massbalance

PIV measurements are used to determine the velocity field. The velocities by measurement are cell c3entered, hence to obtain velocity flux between cells, the values are interpolated to obtain velocity at connecting faces. Assuming that a cell represents a 0D reactor, all outflow streams from the reactor to its neighbours will have the same composition. Thus the massflow through each of these outflow streams can be expressed as the product of a common density and the respective velocity fluxes. By extension of this, the ratios of massflows in the outflows from each cell are a ratio of their velocity fluxes. This principle is used to generate a coefficient matrix as in Eqn 2.5. The coefficient matrix is constructed in the GenerateMFC.py by direct substitution of fluxes in coeffmat, which is a sparse matrix stored in CSC format. Then in MassBalance.py, the diagonal elements are accessed and serve as the a measure of the total outflow from a cell. This is used to normalise the respective column. The boundary conditions are stored in 'rhsvect'. This is constructed by taking the inlet cells' velocity and multiplying it by the inlet gas density. For a "main inlet", the gas composition is the fuel/air mixture supplied at a given temperature and pressure. For all other "secondary inlets" identified as a result of the measured velocity in the domain, the composition is taken as burnt products of this inlet composition. The system of AX=B type of equations is solved to obtain X which is a vector of massflow rates exiting the respective reactors. The density of each reactor is calculated by dividing the massflow rates obtained, by the total velocity flux exiting each reactor obtained from the original diagonal elements of coeffmat, earlier. This density is kept constant during the rest of the calculation. An artificial limit of density of 1.5 kg/m3 is enforced to prevent outlier vectors due to noise from reflections in the PIV measurements from creating reactors with very high densities.

Bibliography