Updating and optimizing a CFcase file for the simulation of a hypersonic flow through a 2D axisymmetric cone with LTE assumption

Sana Amri[1]

[1]Master's Student in Applied Mathematics and Scientific Computing Methods
University Paris XIII - Engineering School Sup Galilée

Internship: April - September 2018

Outline

Physical Modeling

- 2D axisymmetric blunt cone: radius 7 mm
- Chemical perspective:
  - Flow: a Neutral 5-species air $\rightarrow O, N, O_2, N_2, NO$
  - Velocity: Hypersonic (Mach:$\approx$ 6.3)

Assumptions

1- The flow is a continuum medium

- The Knusden number must be strictly inferior than 0.2 (To apply the Navier Stokes equations)

2- The dissociating gas is in chemical equilibrium

- Damkohler number : $Da^c = \frac{\tau_f}{\tau_c} \rightarrow \infty$

3- The gas mixture is in Local Thermodynamic Equilibrium

- The chimical equilibrium composition, can be computed directly for given values of:
  - pressure,
  - temperature
  - finite (fixe) or variable (by adding continuity equations to the model) elemental fractions.

Navier-Stokes Equations

By integrating on a control volume ϖ_i and after applying the divergence formula, we read:

$$\underbrace{\frac{d}{dt}\int_{\varpi_i} \boldsymbol{U} dw}_{\substack{transient \\ term}} + \underbrace{\int_{\Sigma_i} \boldsymbol{F}^c \cdot \boldsymbol{n} ds}_{\substack{convective \\ term}} = \underbrace{\int_{\Sigma_i} \boldsymbol{F}^d \cdot \boldsymbol{n} ds}_{\substack{diffusif \\ term}} + \underbrace{\int_{\varpi_i} \boldsymbol{S} dw}_{\substack{source \\ term}} \quad (1)$$

with $\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}$ conservative variables

Gouverning equations for the chemical mixture (LTE)

- **L**ocal **T**hermodynamics **E**quilibrium with **F**ixed **E**lemental **F**ractions
  We assume the elemental composition constant throughout the flow and equal to the free stream values:

$$Y_O = 0.21 \qquad Y_N = 0.79$$

  Therefore, $y_s := y_s(p, T)$, $s \in \{N_2, O__2, NO, N, O\}$

- **L**ocal **T**hermodynamics **E**quilibrium with **V**ariable **E**lemental **F**ractions

$$\begin{cases} y_s := y_s(p, T, Y_O, Y_N) \\ \frac{\partial Y_O}{\partial t} + \nabla \cdot (\rho Y_O \boldsymbol{u}) = -\nabla \cdot \boldsymbol{J}_O \\ \frac{\partial Y_N}{\partial t} + \nabla \cdot (\rho Y_N \boldsymbol{u}) = -\nabla \cdot \boldsymbol{J}_N \end{cases}$$

  J_e: mass diffusion flux of species e $\qquad$ $\boldsymbol{u}$: mass-averaged mixture velocity
  We solve simultaneously elemental continuity equations for the oxygen and the nitrogen. The missing source term in both equations translate the fact that no new elements are generated in the mixture.

Parameters

Parameters	values
Temperature	1192K
Pressure	6880 Pa
Speed of sound computed with the above parameters	674.32 (mutationpp)
Minimum Velocity	4244m/s
Mach	6.29
Isothermal wall	293K
Air mixture	N_2, O_2, NO, N, O

Numerical methods

Physical Model type	NavierStokes2D
Mutation interface versions used	Mutation2OLD - Mutation - Mutationpp
Space Method	**CellCenteredFVM**
Space Method accuracy	2^{nd} order reconstruction
Convective flux	AUSMPlus2D flux splitter
Diffusive flux	NavierStokes - Nodal extrapolation
SourceTerm	NavierStokes2DAxiST
limiter	Venkatakrishnan's limiter
Mesh	quads, unrefined, nb_elt : 10K

Previous work

1. We received an old CFcase file from Fernando MiroMiro that we updated and adapted for our case.
2. Improving the speed of the convergence in 2^{nd} order of accuracy (of the temperature residue) by testing different kind of algorithms (functions) for:
   - the CFL parameter
   - the transition in 1rst to 2nd order accuracy
   - the limiter

For more details (plots + explanations) about what has been done so far:

`https://github.com/SanaAmri/LTE_NS_Cone/blob/master/README`

Mutation interfaces in COOLFluiD - Current work (2 days ago)

Different interfaces on COOLFLuiD in order to use the Mutation libraries.

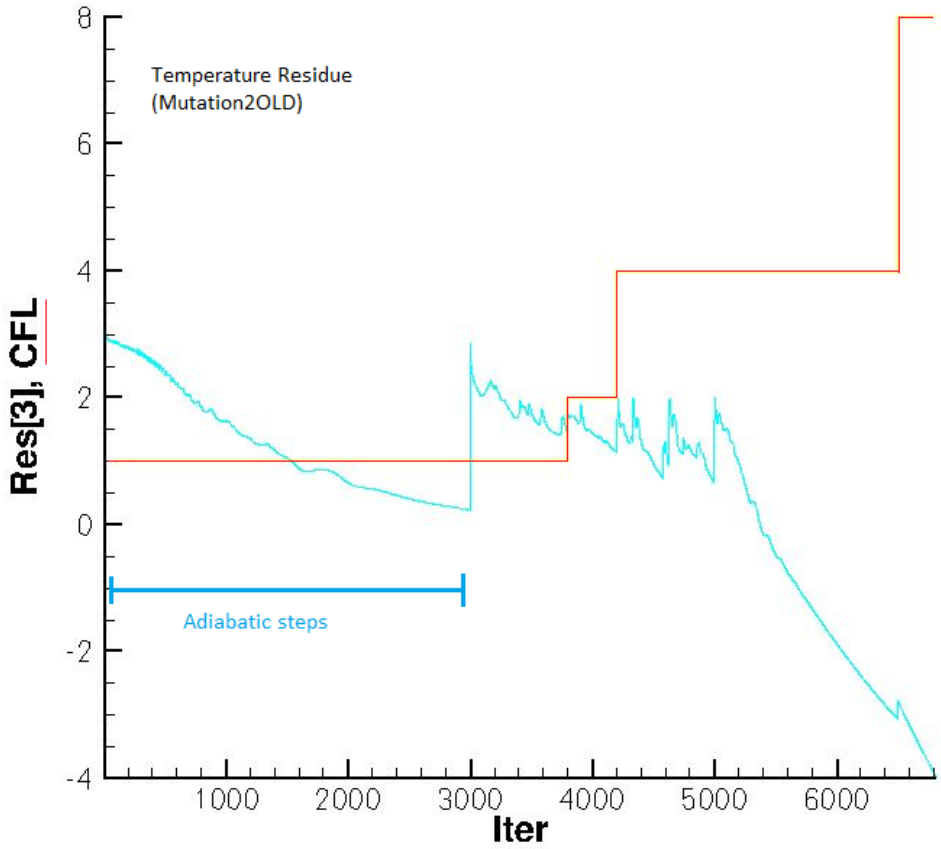
Mutation library version	**Interface name in COOLFluiD**	**Path in the COOLFluiD repository**
Mutation2 library	Mutation2OLD	plugins/Mutation2.0.0l/Mutation2OLD.*
Mutation library	Mutation (Lookup Table)	plugins/Mutationl/MutationLibrary.*
Mutation++ library	Mutationpp	plugins/Mutationppl/MutationLibrarypp.*

Until now, we were using the Mutation2OLD interface. **What is the impact on the convergence of the temperature residue (in 1rst and 2nd order accuracy) when we use the other interfaces?**

Note : For the next results obtained, we implemented a function (CFL,grad,limiter) for the convergence in 1rst order accuracy based on the results got before. For the second order accuracy we used the function 2A (see previous work)

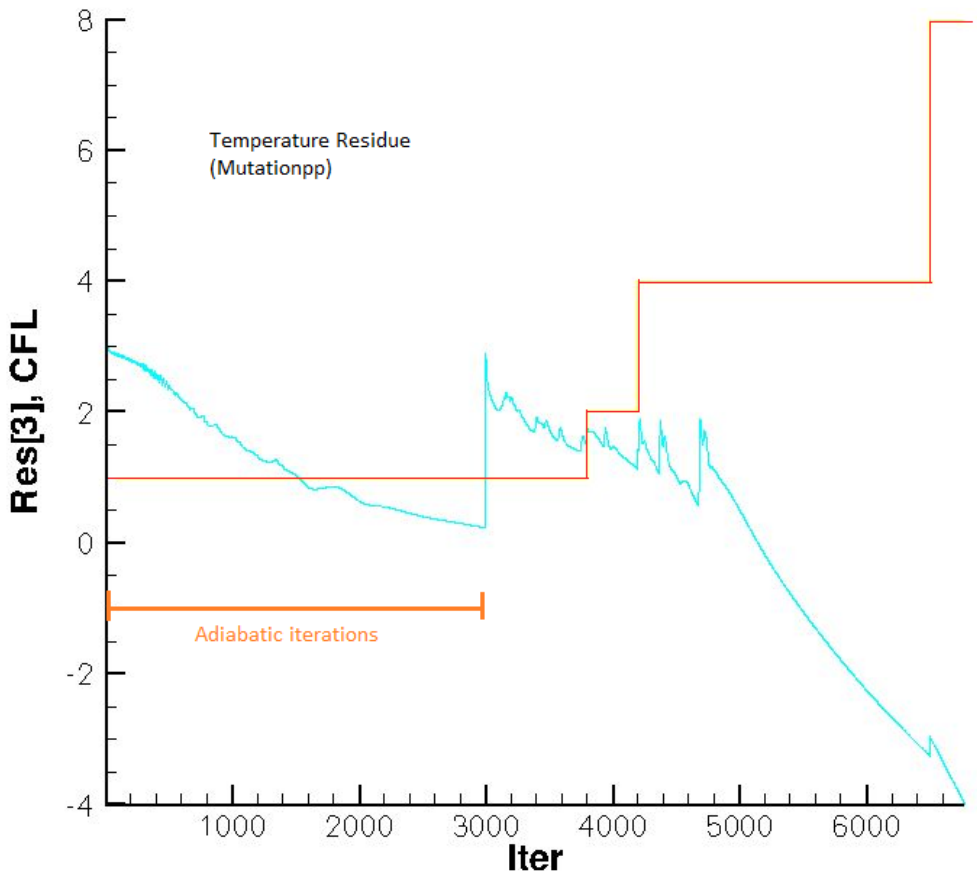
1rst order accuracy convergence - Mutation2OLD

```
Total Number Iter: 6821 Reached Residual: -4.0012 and took: 5 h 18 min 53.8307 sec
```



1rst order accuracy convergence - Mutationpp

```
Total Number Iter: 6779 Reached Residual: -4.00345 and took: 4 h 9 min 54.9361 s
```



1rst order accuracy convergence - Mutation

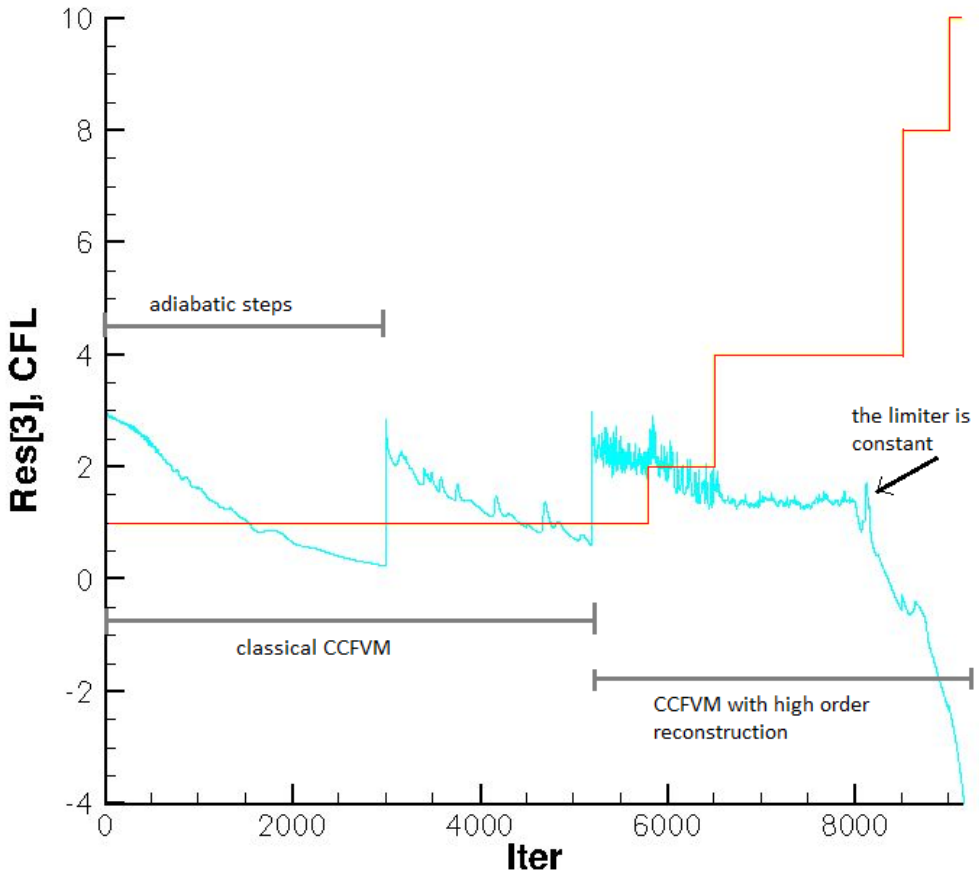
We didn't manage to converge using the Mutation interface. By changing the folowing parameters we've been able to run the case for only 19 iterations.

```
#
#       ------------
#        Mutation
#       ------------
Simulator.SubSystem.NavierStokes2D.PropertyLibrary = Mutation
Simulator.SubSystem.NavierStokes2D.Mutation.mixtureName = air5
Simulator.SubSystem.NavierStokes2D.Mutation.lookUpVars = a e h d
Simulator.SubSystem.NavierStokes2D.Mutation.Tmin = 1000.
Simulator.SubSystem.NavierStokes2D.Mutation.Tmax = 7000.
Simulator.SubSystem.NavierStokes2D.Mutation.deltaT = 10.
Simulator.SubSystem.NavierStokes2D.Mutation.Pmin = 6500.
Simulator.SubSystem.NavierStokes2D.Mutation.Pmax = 400000.
Simulator.SubSystem.NavierStokes2D.Mutation.deltaP = 10000.
#####
```

```
coolfluid-solver: /software/alternate/coolfluid/cf2/2015.11/trunk/plugins/LTE/Euler2DPuvtLTE.cxx:93: virtual void COOLFluiD::Phys
ics::LTE::Euler2DPuvtLTE::computePhysicalData(const COOLFluiD::Framework::State&, COOLFluiD::RealVector&): Assertion `pdim > 0.'
failed.
```

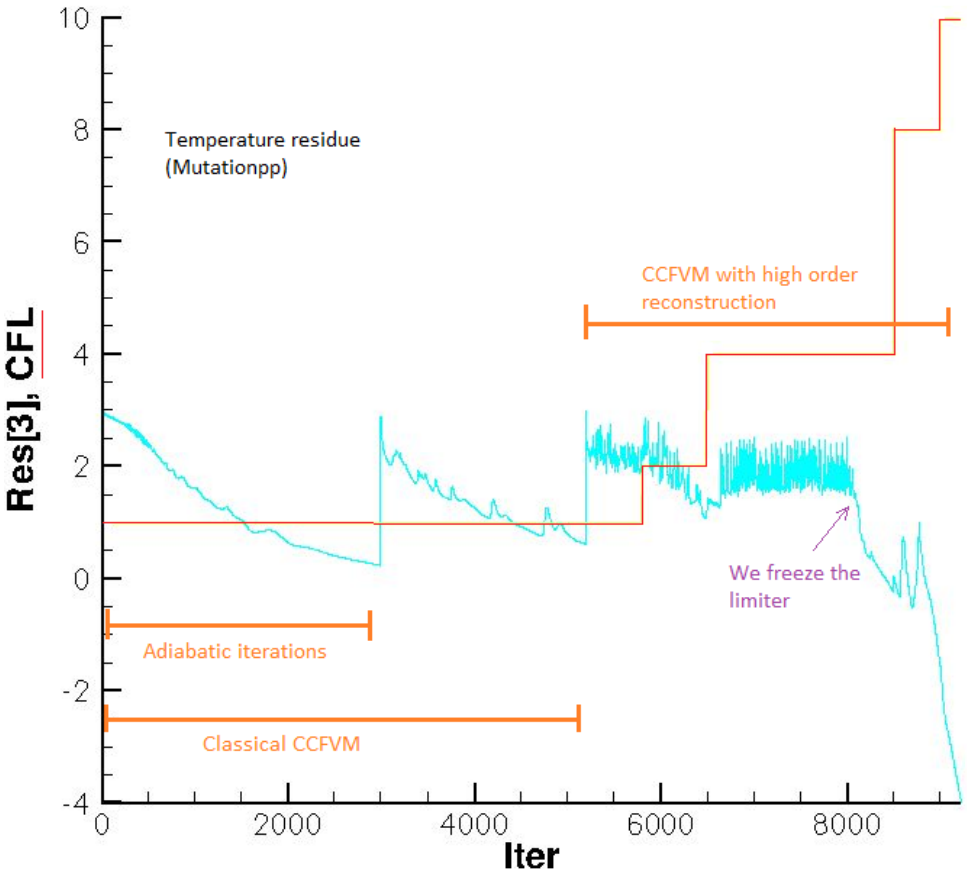
2nd order accuracy convergence - Mutation2OLD

```
Total Number Iter: 9170 Reached Residual: -4.00844 and took: 7 h 22 min 59.3689 sec
```



2nd order accuracy convergence - Mutationpp

```
Total Number Iter: 9232 Reached Residual: -4.00377 and took: 5 h 48 min 21.1605 sec
```



Recap

Interface in COOLFluiD	Mutation2OLD	Mutationpp	Mutation
Temp Residue cv 1rst order	cv ok - 6821 iters - 5h19	cv ok - 6779 iters - 4h10	no cv - reach 19 iters max
Temp Residue cv 2nd order	cv ok - 9170 iters - 7h23	cv ok - 9232 iters - 5h48	

Lookup table in Mutationpp ?

AIM : Improve the convergence using the Mutationpp interface in COOLFluiD on which we will add a Lookup table to minimize the time of the CFcase file execution. We need to tabulate :

- The total energy: e
- The total enthalpy: h
- The sound speed: a
- The density: d

Lookup table already implemented in the Mutation interface

In plugins/MutationI/MutationLibrary.*

```
44
45    typedef Common::LookupTable2D<CFdouble, CFdouble, CFdouble> LkpTable;
46
```

```
2495    // compute the arrays containing all the keys for T and P
2496    vector<CFdouble> vT(nbT);
2497    vector<CFdouble> vP(nbP);
2498    for(CFuint i = 0; i < nbT; ++i) {
2499      vT[i] = _Tmin + i*_deltaT;
2500    }
2501    for(CFuint j = 0; j < nbP; ++j) {
2502      vP[j] =_pmin + j*_deltaP;
2503    }
2504
```

```
638    /// table of lookup tables
639    LkpTable _lookUpTables;
```

Lookup table: Mutation -> Mutationpp

```
1412  void MutationLibrary::setComposition(CFdouble& temp,
1413                                       CFdouble& pressure,
1414                                       RealVector* x)
1415  {
1416    if (!_useLookUpTable) {
1417      // initialization of the molar fractions
1418      for(int i = 0; i < _NS; ++i) {
1419        Xini[i] = 1.0;
1420      }
1421
1422      // compute the molar fractions corresponding to the given temperature and
1423      // pressure for the mixture Xn
1424      FORTRAN_NAME(composition)(WR1,&LWR1,WI,&LWI,&temp,&pressure,Xn,Xini,X);
1425
1426      if (x != CFNULL) {
1427        for(int i = 0; i < _NS; ++i) {
1428          (*x)[i] = static_cast<CFreal>(X[i]);
1429        }
1430      }
1431
1432      // set mass fractions which will be used later
1433      CFreal massTot = 0.;
1434      for (int is = 0; is < _NS; ++is) {
1435        if (X[is] > 1.00000000001) {
1436          cout << "X[" << is << "] = " << X[is] << endl;
1437          abort();
1438        }
1439        const CFreal mm = X[is]*MOLARMASSP[is];
1440        massTot += mm;
1441        Y[is] = mm;
1442      }
1443
1444      for (int is = 0; is < _NS; ++is) {
1445        Y[is] /= massTot;
1446      }
1447    }
1448  }
```

```
296  void MutationLibrarypp::setComposition(CFdouble& temp,
297                                         CFdouble& pressure,
298                                         RealVector* x)
299  {
300    if (temp < 100.) {temp = 100.;}
301
302    m_gasMixtureEquil->setState(&pressure, &temp, 1);
303    const double* xm = m_gasMixtureEquil->X();
304
305    if (x != CFNULL) {
306      for(CFint i = 0; i < _NS; ++i) {
307        (*x)[i] = xm[i];
308      }
309    }
310
311    m_gasMixtureEquil->convert<X_TO_Y>(xm, &m_y[0]);
312
313    CFLog(DEBUG_MAX, "Mutation::setComposition() => m_y = " << m_y << "\n");
314  }
```

Lookup table in Mutation -> Mutationpp

```
1392 CFdouble MutationLibrary::soundSpeed(CFdouble& temp,
1393                                      CFdouble& pressure)
1394 {
1395   if (!_useLookUpTable) {
1396     // compute the ratio of the mixture frozen specific heat in thermal equilibrium  c
1397     CFdouble gamma = 0.0;
1398     CFdouble drhodp = 0.0;
1399     CFdouble rho = density(temp, pressure, CFNULL);
1400     CFdouble eps = 0.1;
1401     FORTRAN_NAME(equigamma)(WR1, &LWR1, WI, &LWI, &temp, &pressure,
1402                             &rho, Xn, X, &eps, &gamma, &drhodp);
1403     return sqrt(gamma/drhodp);
1404   }
1405   else {
1406     return _lookUpTables.get(temp, pressure, _nameToIdxVar.find("a"));
1407   }
1408 }
```

```
288 CFdouble MutationLibrarypp::soundSpeed(CFdouble& temp, CFdouble& pressure)
289 {
290   m_gasMixture->setState(&pressure, &temp, 1);
291   return m_gasMixture->equilibriumSoundSpeed();
292 }
```

Lookup table in Mutation -> Mutationpp

```
1659  CFdouble MutationLibrary::density(CFdouble& temp,
1660                                    CFdouble& pressure,
1661                                    CFreal* tVec)
1662  {
1663    //  CF_DEBUG_POINT;
1664    if (!_useLookUpTable) {
1665      CFdouble ND = 0.0;
1666      CFdouble rho = 0.0;
1667
1668      CFdouble Te = getTe(temp,tVec);
1669      FORTRAN_NAME(numberd)(WR1,&LWR1,&pressure,&temp,&Te,X,&ND);
1670      FORTRAN_NAME(density)(WR1,&LWR1,X,&ND,&rho);
1671      return rho;
1672    }
1673    return _lookUpTables.get(temp, pressure, _nameToIdxVar.find("d"));
1674  }
```

```
352  CFdouble MutationLibrarypp::density(CFdouble& temp,
353                                      CFdouble& pressure,
354                                      CFreal* tVec)
355  {
356    if (m_smType == LTE) {m_gasMixture->setState(&pressure, &temp, 1);}
357    return m_gasMixture->density();
358  }
```

Lookup table in Mutation -> Mutationpp

```
1723 CFdouble MutationLibrary::energy(CFdouble& temp,
1724                                  CFdouble& pressure)
1725 {
1726   if (!_useLookUpTable) {
1727     FORTRAN_NAME(energy)(WR1,&LWR1,WI,&LWI,&temp,&temp,&temp,&temp,&pressure,
1728                          _ETOTAL,_ETRANS,_EELECT,_EROT,_EVIBR,_EFORM);
1729
1730     CFdouble ND = 0.0;
1731     CFdouble rho = 0.0;
1732     CFdouble MMass = 0.0;
1733
1734     // store the density
1735     FORTRAN_NAME(numberd)(WR1,&LWR1,&pressure,&temp,&temp,X,&ND);
1736     FORTRAN_NAME(density)(WR1,&LWR1,X,&ND,&rho);
1737     FORTRAN_NAME(molarmass)(WR1,&LWR1,&rho,&ND,&MMass);
1738
1739     // sum up all the internal energies for each species
1740     CFdouble intEnergy = 0.0;
1741     for(int i = 0; i < _NS; ++i) {
1742       intEnergy += X[i]*_ETOTAL[i];
1743     }
1744     return intEnergy /= MMass;
1745   }
1746   else {
1747     return _lookUpTables.get(temp, pressure, _nameToIdxVar.find("e"));
1748   }
1749 }
```

```
392 CFdouble MutationLibrarypp::energy(CFdouble& temp,
393                                     CFdouble& pressure)
394
395 {
396   m_gasMixture->setState(&pressure, &temp, 1);
397   return m_gasMixture->mixtureEnergyMass()- m_H0;
398 }
```

Lookup table in Mutation -> Mutationpp

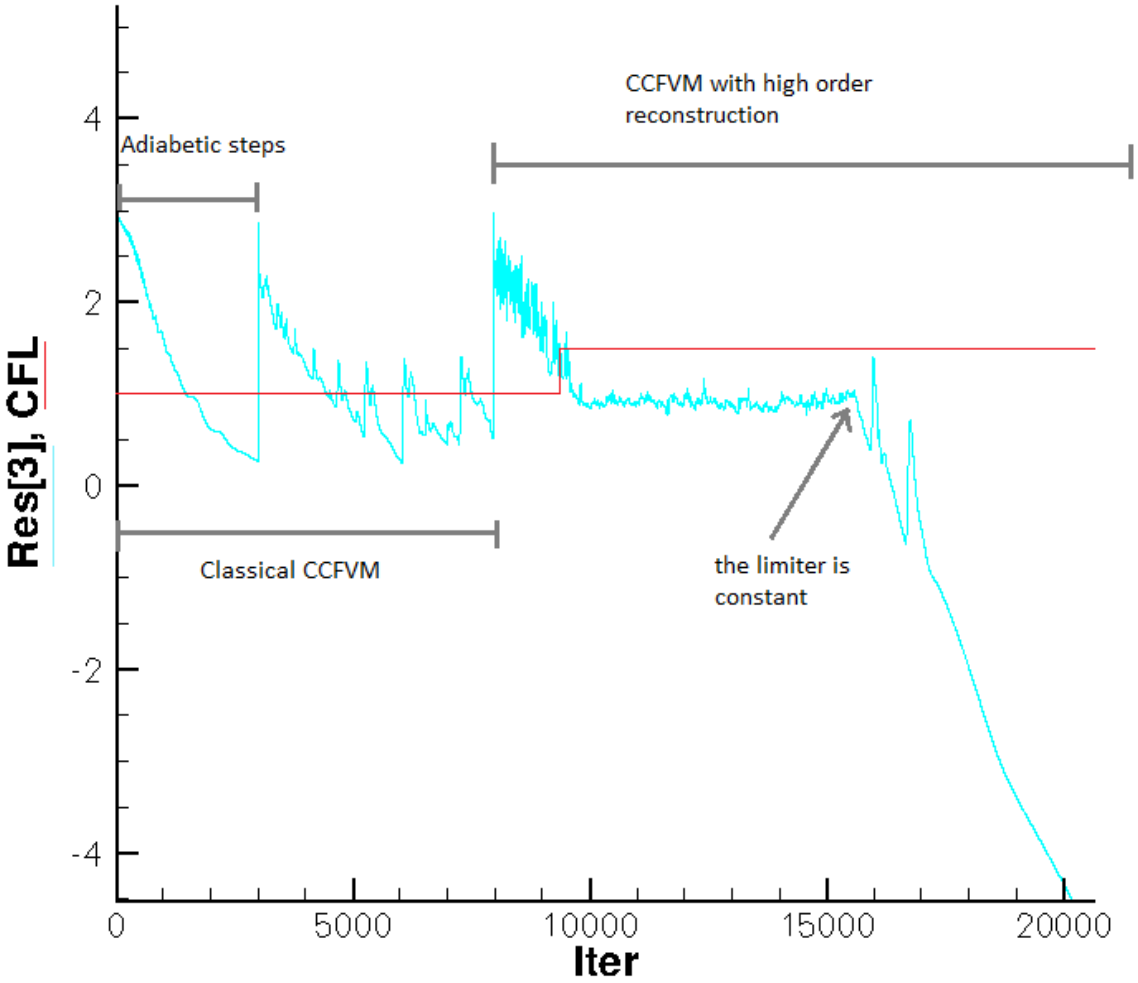
```
1753  CFdouble MutationLibrary::enthalpy(CFdouble& temp,
1754                                     CFdouble& pressure)
1755  {
1756    if (!_useLookUpTable) {
1757      FORTRAN_NAME(enthalpy)(WR1,&LWR1,WI,&LWI,&temp,&temp,&temp,&temp,&pressure,
1758                             _HTOTAL,_HTRANS,_HELECT,_HROT,_HVIBR,_HFORM);
1759
1760      CFdouble ND = 0.0;
1761      CFdouble rho = 0.0;
1762      CFdouble MMass = 0.0;
1763
1764      // store the density
1765      FORTRAN_NAME(numberd)(WR1,&LWR1,&pressure,&temp,&temp,X,&ND);
1766      FORTRAN_NAME(density)(WR1,&LWR1,X,&ND,&rho);
1767      FORTRAN_NAME(molarmass)(WR1,&LWR1,&rho,&ND,&MMass);
1768
1769      // sum up all the internal energies for each species
1770      CFdouble h = 0.0;
1771      for(int i = 0; i < _NS; ++i) {
1772        h += X[i]*_HTOTAL[i];
1773      }
1774      return h /= MMass;
1775    }
1776    else {
1777      return _lookUpTables.get(temp, pressure, _nameToIdxVar.find("e"));
1778    }
1779  }
```

```
402  CFdouble MutationLibrarypp::enthalpy(CFdouble& temp,
403                                       CFdouble& pressure)
404  {
405    m_gasMixture->setState(&pressure, &temp, 1);
406    return m_gasMixture->mixtureHMass() - m_H0;
407  }
```

APPENDIX

Improving the speed of the convergence in 2^{nd} order of accuracy

F I R S T A T T E M P T: Patterns in the convergence for the temperature residue.



Improving the speed of the convergence in 2^{nd} order of accuracy

Improving the speed of the convergence in 2^{nd} order of accuracy (of the temperature residue) by testing different kind of algorithms (functions) for:

- the CFL parameter
- the transition in 1rst to 2nd order accuracy
- the limiter (when to freeze it during the convergence)

RESULTS: After the different tests (next slides), the residue of the temperature converge after only 9000 iterations (instead of 20K) in second order accuracy.

Improving the speed of the convergence in 2^{nd} order of accuracy

Two strategies for increasing the speed of the convergence of the temperature residue :

1. Compute from scratch the case with the classical CCFVM ($\Rightarrow O(h)$) and try to tend to the criteria of convergence (10^{-4} for the temperature residue) without reaching it. Then , make the transition in second order accuracy $O(h^2)$ by activating the high order reconstruction (gradient=1.0). That way, we expect that the convergence in 2^{nd} order accuracy will go faster because the residue will be very close to zero during the transition. The configurations for the CFL and the limiter are based on the results obained. **[See test functions 1A,1B,1C and 1D - next slides]**
2. Start from scratch to run with the classical CCFVM, and make the transition in second order accuracy right after the adiabatic steps. Then adapt the CFL and the limiter parameters based on the results obained. **[See test functions 2A and 2B - next slides]**

Function test 1A: Algorithm

Function test 1A

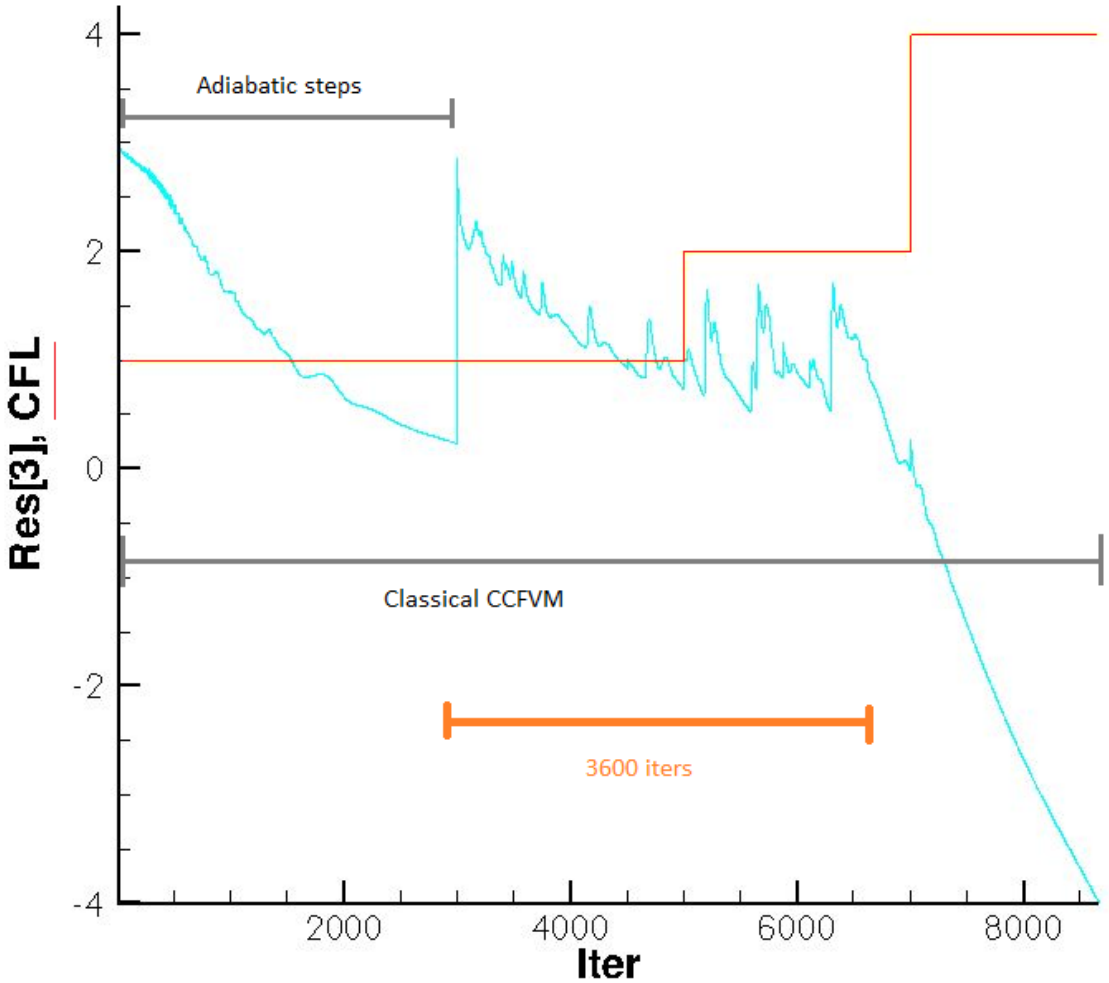
1. Gradient= 0. $\rightarrow$ Classic CCFVM, 1^{rst} order accuracy
2. **WHILE** temperature residue $> 10^{-4}$
3. **IF**(iter<5000)
4. CFL=1.0
5. **ELSE IF**(5000$\leq$iter<7000)
6. CFL=2.0
7. **ELSE IF**(7000$\leq$iter<10000)
8. CFL=4.0
9. **ELSE IF**(10000$\leq$iter<12000)
10. CFL=1.0
11. **ELSE IF**(12000$\leq$iter<16000)
12. CFL=2.0
13. **IF**(iter=12000)
14. Gradient=1. $\rightarrow$ Transition to second order accuracy
15. **END IF**
16. **ELSE** (16000$\leq$iter)
17. CFL=4.0
18. **IF**(iter=18000)
19. $\rightarrow$ Freeze the factor limiter from the piecewise
20. polynomials approximation of each cells
21. **END IF**
22. **END IF**
23. **END WHILE**

Function test 1A: Expectations

G O A L S:

- Reduce the oscillating part after the end of the 3000 adiabatic iterations (These steps are usefull to detache the shock from the boundaries).
- Increase the order of accuracy of the scheme when the residue is close to 10^{-4} (convergence value) without reaching it. That way, we expect that the convergence in 2^{nd} order accuracy will go faster because the residue will be very close to zero during the transition.

Function test 1A: Results



Function test 1A: Observations

O B S E R V A T I O N S:

- The residue converged (iter=8674, $\simeq$13h) before the attempt to reach a second order accuracy (gradient=1. at iter 12000 in the previous algorithm)
- We reduced (relatively to the **first try**) the oscillating part after the adiabatic steps (5000 $\rightarrow$ 3600)

Function test 1B: Algorithm

Function test 1B

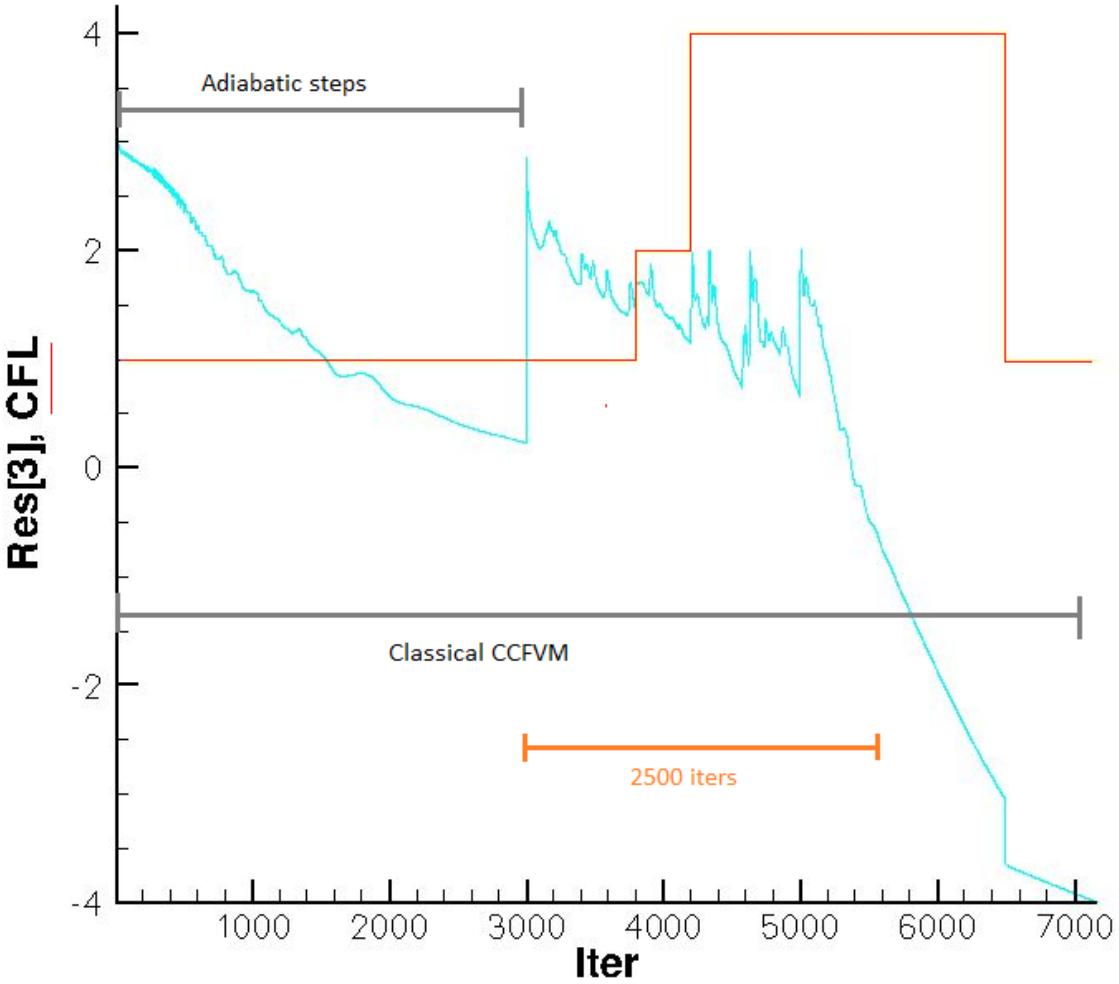
1. Gradient= 0. $\rightarrow$ Classic CCFVM, 1^{rst} order accuracy
2. **WHILE** temperature residue $> 10^{-4}$
3. **IF**(iter$<$3800)
4. CFL=1.0
5. **ELSE IF**(3800$\leq$iter$<$4200)
6. CFL=2.0
7. **ELSE IF**(4200$\leq$iter$<$6500)
8. CFL=4.0
9. **ELSE IF**(6500$\leq$iter$<$7000)
10. CFL=1.0
11. **ELSE IF**(7000$\leq$iter$<$9000)
12. CFL=2.0
13. **IF**(iter=8500)
14. Gradient=1. $\rightarrow$ Transition to second order accuracy
15. **END IF**
16. **ELSE** (9000$\leq$iter)
17. CFL=4.0
18. **IF**(iter=12000)
19. $\rightarrow$ Freeze the factor limiter from the piecewise
20. polynomials approximation of each cells
21. **END IF**
22. **END IF**
23. **END WHILE**

Function test 1B: Expectations

G O A L S:

- Reduce (relatively to function 1A) the oscillating part after the end of the 3000 adiabatic iterations.
- Change to second order accuracy sooner than the function 1A to avoid the complete convergence in first order accuracy.

Function test 1B: Results



function test 1B: Observations

O B S E R V A T I O N S:

- Again, the residue converged (iter=7176, $\simeq$11,5h) before the attempt to reach a second order accuracy (gradient=1. at iter 8000 in the algorithm 1B).
- As expected, we reduced (relatively to the function 1A) the oscillating part after the adiabatic steps (3600iters $\rightarrow$ 2500iters).

function test 1C: Algorithm

Function test 1C

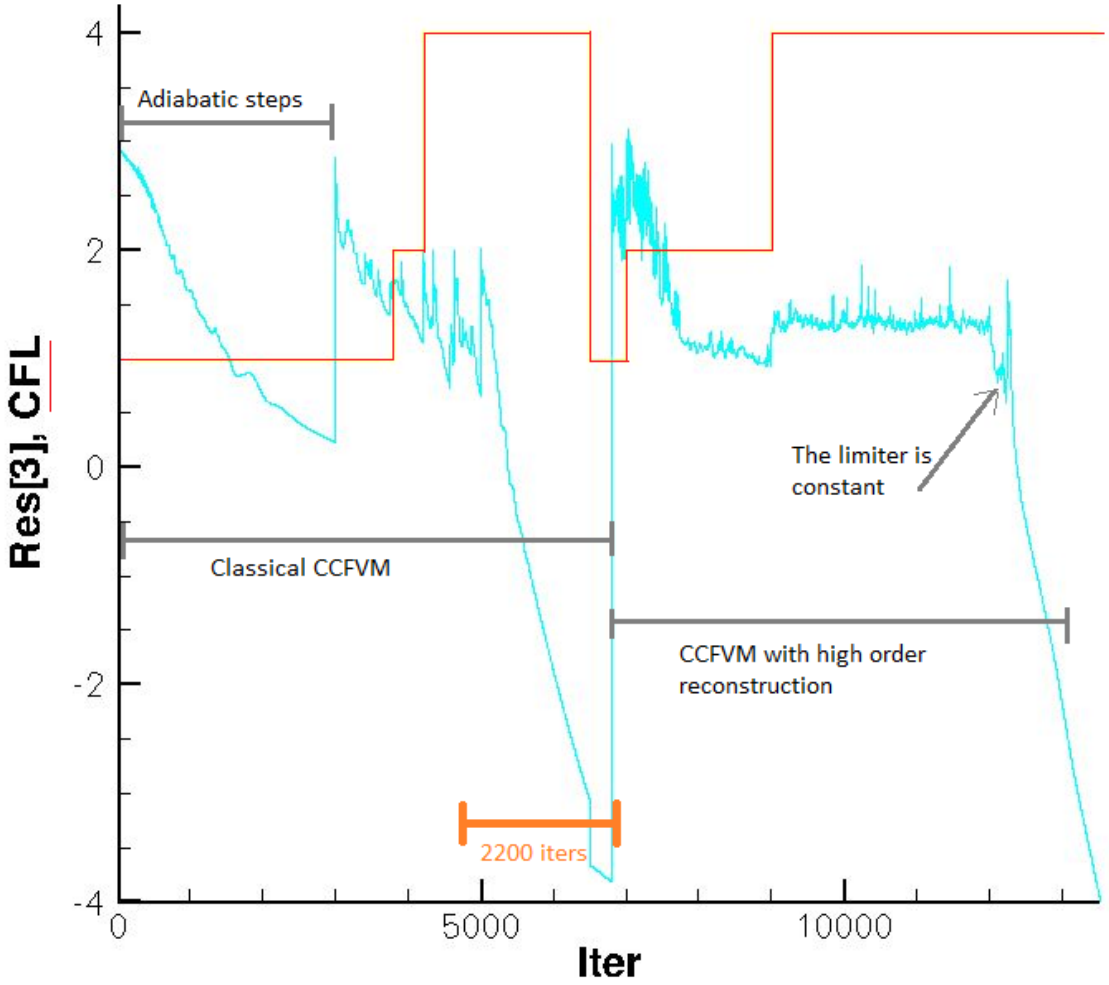
1. Gradient= 0. $\rightarrow$ Classic CCFVM, 1^{rst} order accuracy
2. **WHILE** temperature residue $> 10^{-4}$
3. **IF**(iter$<$3800)
4. CFL=1.0
5. **ELSE IF**(3800$\leq$iter$<$4200)
6. CFL=2.0
7. **ELSE IF**(4200$\leq$iter$<$6500)
8. CFL=4.0
9. **ELSE IF**(6500$\leq$iter$<$7000)
10. CFL=1.0
11. **IF**(iter=6800)
12. Gradient=1. $\rightarrow$ Transition to second order accuracy
13. **END IF**
14. **ELSE IF**(7000$\leq$iter$<$9000)
15. CFL=2.0
16. **ELSE** (9000$\leq$iter)
17. CFL=4.0
18. **IF**(iter=12000)
19. $\rightarrow$ Freeze the factor limiter from the piecewise
20. polynomials approximation of each cells
21. **END IF**
22. **END IF**
23. **END WHILE**

function test 1C: Expectations

G O A L S:

- Make the transition to a second order accuracy before the residue reaches 10^{-4} .

function test 1C: Results



function test 1C: Observations

O B S E R V A T I O N S:

- We manage to make the transition in a higher order accuracy, but we have a big gap that we didn't expect.
- In second order, we can see that the limiter has a big impact on the convergence of the temperature residue.

```
Total Number Iter: 13537 Reached Residual: -4.00207 and took: 13 h 54 min 45.0284 sec
```

function test 1D: Algorithm

Function test 1D

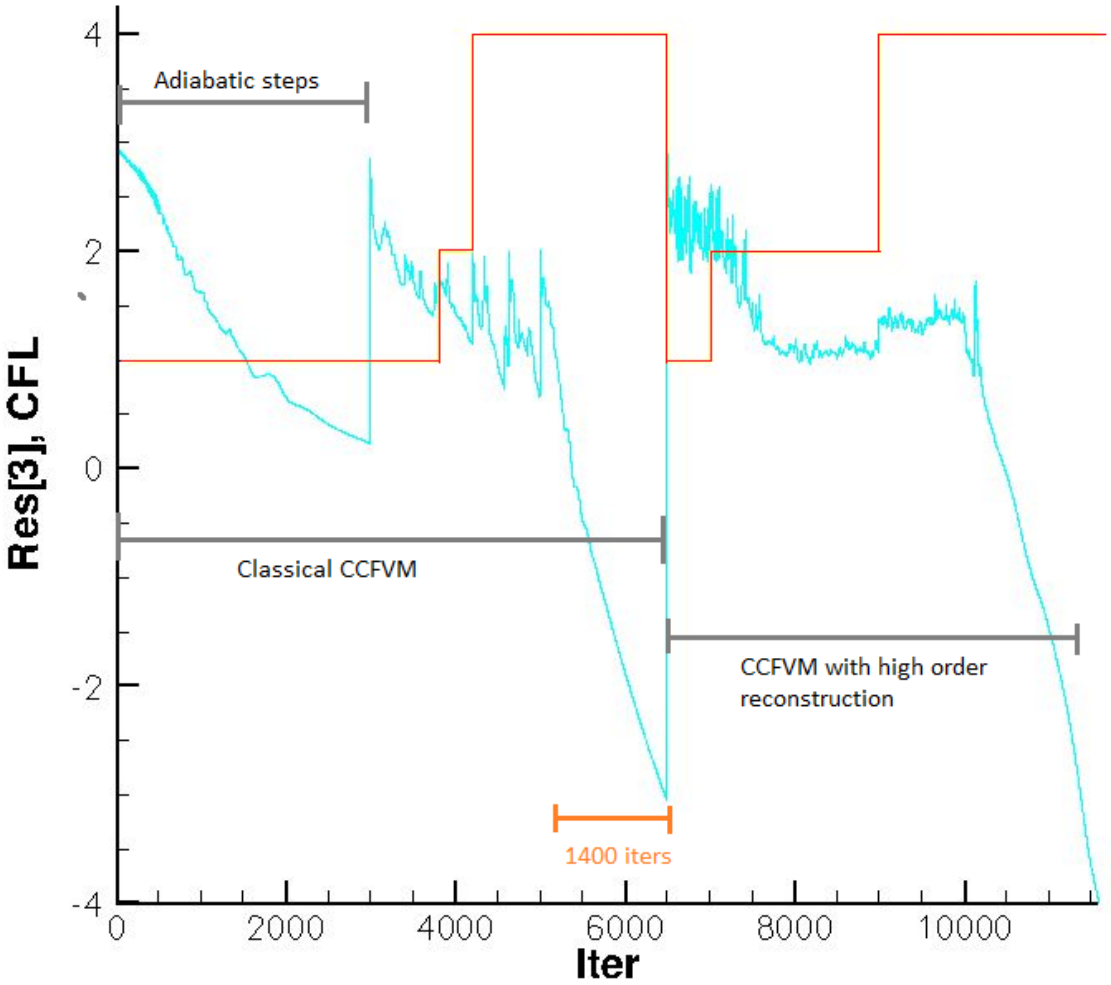
1. Gradient= 0. $\rightarrow$ Classic CCFVM, 1^{rst} order accuracy
2. **WHILE** temperature residue $> 10^{-4}$
3. **IF**(iter$<$3800)
4. CFL=1.0
5. **ELSE IF**(3800$\leq$iter$<$4200)
6. CFL=2.0
7. **ELSE IF**(4200$\leq$iter$<$6500)
8. CFL=4.0
9. **ELSE IF**(6500$\leq$iter$<$7000)
10. CFL=1.0
11. **IF**(iter=6500)
12. Gradient=1. $\rightarrow$ Transition to second order accuracy
13. **END IF**
14. **ELSE IF**(7000$\leq$iter$<$9000)
15. CFL=2.0
16. **ELSE** (9000$\leq$iter)
17. CFL=4.0
18. **IF**(iter=10000)
19. $\rightarrow$ Freeze the factor limiter from the piecewise
20. polynomials approximation of each cells
21. **END IF**
22. **END IF**
23. **END WHILE**

function test 1D: Expectations

G O A L S:

- We would like to minimize (relatively to the function 1C), the gap during the transition part. The trick is to change the gradient to 1. **and** the CFL to 1 at the same iteration.

function test 1D: Results



function test 1D: Observations

O B S E R V A T I O N S:

- We manage to reduce the transition to a higher order accuracy, but the gap is still too high and add many unnecessary steps of convergence.
- In the first attempt, where we configurated the parameter manually (interfile), we changed the gradient to 1. without waiting for the convergence in first order of accuracy. This strategy avoided the discontinuity that we obtained here. That is why I changed the strategy of convergence for the next test functions.

```
Total Number Iter: 11592 Reached Residual: -4.00108 and took: 12 h 28 min 38.6879 sec
```

Function test 2A: Algorithm

Function test 2A : (we don't wait to converge in first order accuracy)

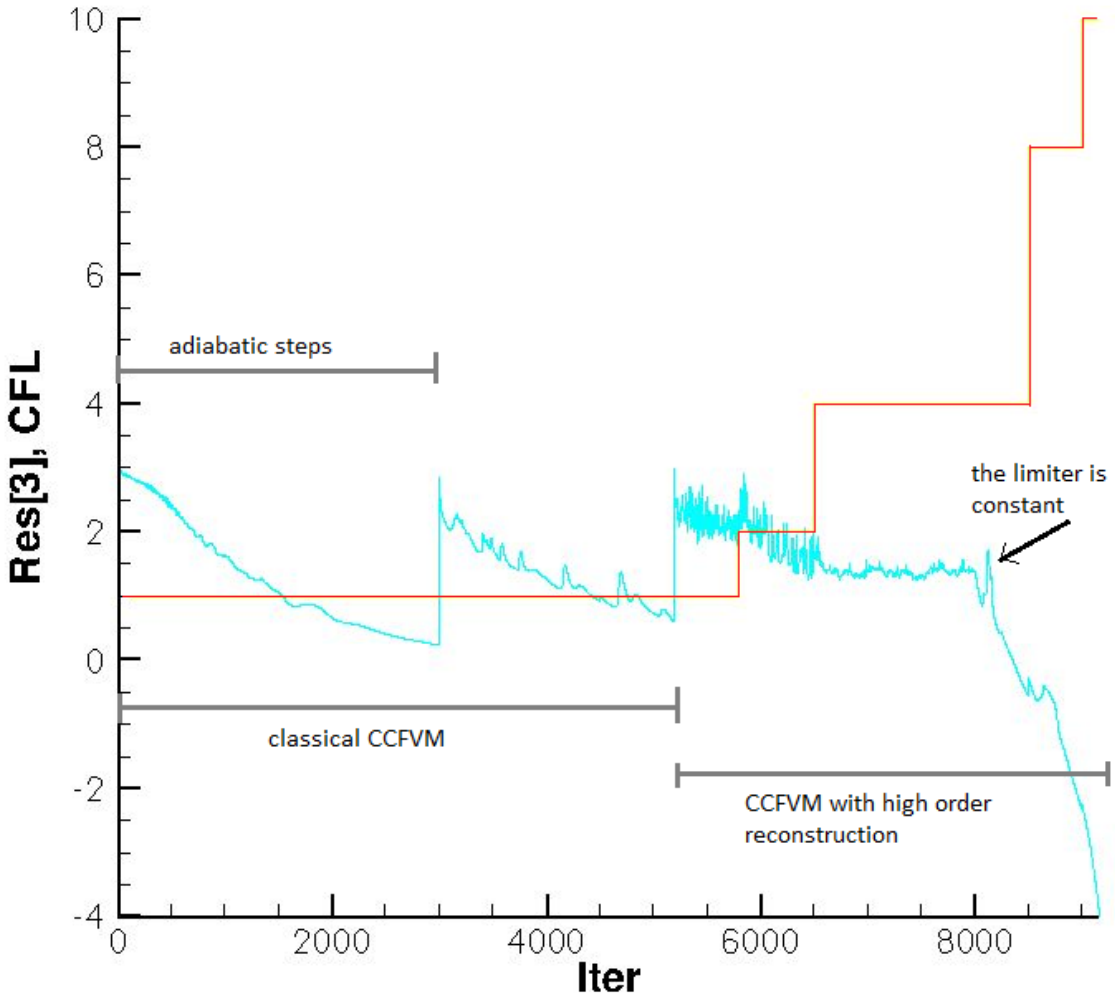
1. Gradient= 0. $\rightarrow$ Classic CCFVM, 1^{rst} order accuracy
2. **WHILE** temperature residue $> 10^{-4}$
3. **IF**(iter<5800)
4. CFL=1.0
5. **IF**(iter=5200)
6. Gradient=1. $\rightarrow$ Transition to second order accuracy
7. **END IF**
8. **ELSE IF**(5800$\leq$iter<6500)
9. CFL=2.0
10. **ELSE IF**(6500$\leq$iter<8500)
11. CFL=4.0
18. **IF**(iter=8000)
19. $\rightarrow$ Freeze the factor limiter from the piecewise
20. polynomials approximation of each cells
21. **END IF**
12. **ELSE** (8500$\leq$iter<9000)
13. CFL=8.0
16. **ELSE** (9000$\leq$iter)
17. CFL=10.0
22. **END IF**
23. **END WHILE**

Function test 2A: Expectations

G O A L S:

- Based on the observations made in the first attempt, we would like to reduce the oscillating part after the 3000 iteration.
- Also, when we reach the second order accuracy, we would like to freeze the limiter sooner to reduce the oscillating part and reach the downward slope faster.

Function test 2A: Results



Function test 2A: Observations

O B S E R V A T I O N S:

- The expectations are reached, but we can reduce even more the oscillating parts.

```
Total Number Iter: 9170 Reached Residual: -4.00844 and took: 10 h 22 min 30.3091
 sec
```

Function test 2B: Algorithm

Function test 2B :

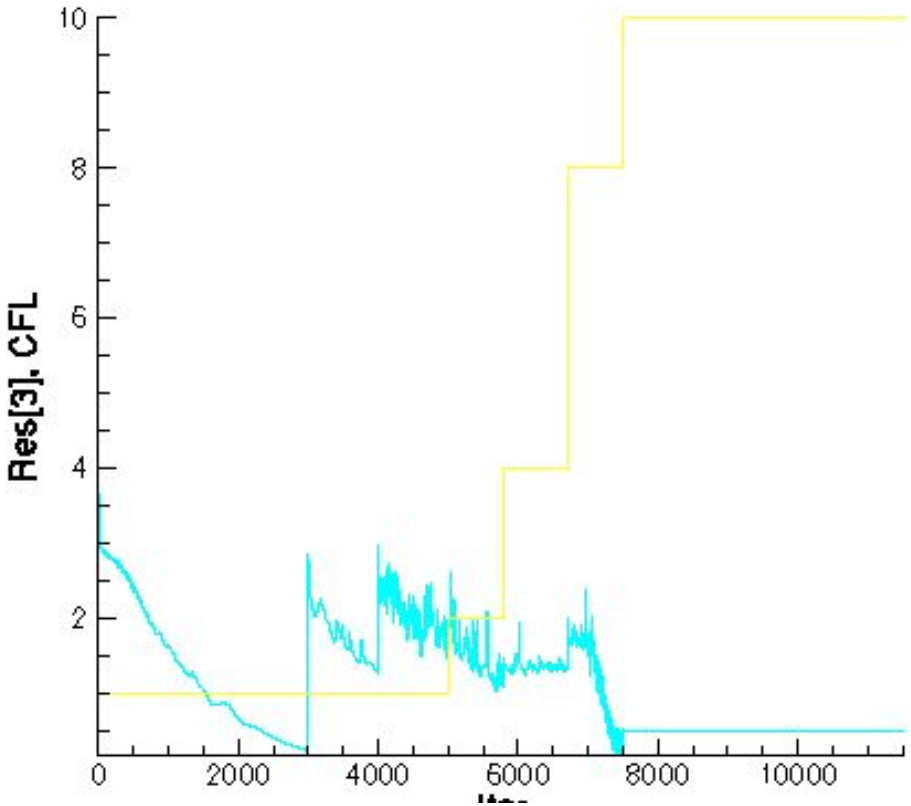
1. Gradient= 0. $\rightarrow$ Classic CCFVM, 1^{rst} order accuracy
2. **WHILE** temperature residue $> 10^{-4}$
3. **IF**(iter$<$5000)
4. CFL=1.0
5. **IF**(iter=4000)
6. Gradient=1. $\rightarrow$ Transition to second order accuracy
7. **END IF**
8. **ELSE IF**(5000$\leq$iter$<$5800)
9. CFL=2.0
10. **ELSE IF**(5800$\leq$iter$<$6700)
11. CFL=4.0
18. **IF**(iter=7000)
19. $\rightarrow$ Freeze the factor limiter from the piecewise
20. polynomials approximation of each cells
21. **END IF**
12. **ELSE IF**(6700$\leq$iter$<$7500)
13. CFL=8.0
16. **ELSE** (7500$\leq$iter)
17. CFL=10.0
22. **END IF**
23. **END WHILE**

Function test 2B: Expectations

G O A L S:

- Optimizing the previous results obtained using the function 2A.

Function test 2B: Results



Function test 2B: Observations

O B S E R V A T I O N S:

- When we tried to reduce the oscillating parts the residue tend to stay constant. So the best result that we have is the function 2A.