Calculation of Test Flow Conditions for Reflected-Shock Tunnels

Peter Jacobs, Wilson Chan, Tamara Sopek,
Fabian Zander, Kyle Damm, Nick Gibbons,
Rowan Gollan

The University of Queensland (at one time or another)

03 Dec 2020

"Classic" calculation process

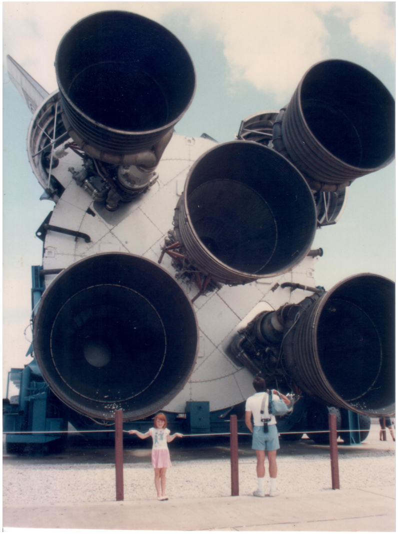
ESTC and NENZF

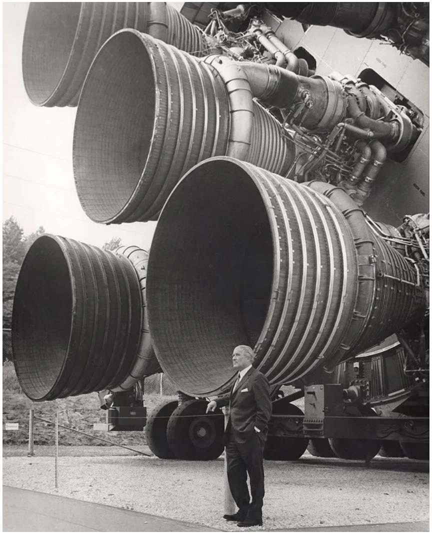
"New" codes

nenzf2d: state-to-state with axisymmetric nozzle expansion

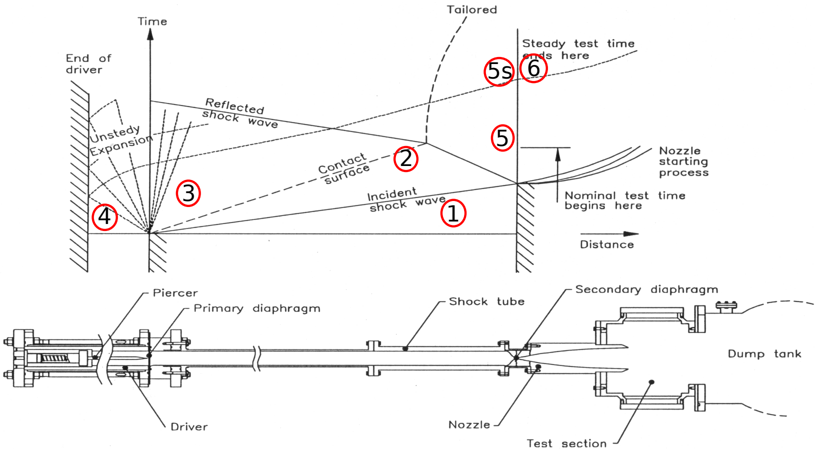
nenzf1d: state-to-state with quasi1D nozzle expansion

Motivation: nozzles with significant nonequilibrium flow.





Classic shock tunnel operation with flow states



Recipe for estimating flow conditions - 1/2

- Measure State 1 and State 4.
  Ideally, we could compute everything from this but it turns out that practice does not match theory and it takes months of supercomputer time to do this with any useful precision. So...
- Run the shot and measure V_1 and p_s (and, probably p_{Pitot}).
  Should also measure as much as we can reasonably (p_2, p_5, etc) noting that, for machines with relatively long shock tubes (like T4), there is no one-true-value for V_1.
- Compute State 2 from State 1 and V_1, assuming a normal shock moving into quiescent gas. Although the gas will react chemically, we assume that it stays in thermochemical equilibrium.

Recipe for estimating flow conditions - 2/2

- Compute State 5 from State 2 assuming that the reflected shock brings the test gas to rest, as described in JD Anderson's text book. However, the pressure after the reflected shock is often nothing like that expected from an ideal computation so...
- Assume an isentropic relaxation from p_5 down to the observed equilibrium value p_s to get the *nozzle-supply condition 5s*.
- Take State 5s as the stagnation conditions and expand isentropically to sonic conditions (State 6) at the throat and further to nozzle-exit conditions. It's probably OK to assume thermochemical equilibrium down through the throat but not necessarily further into the low-density and cooler parts of the nozzle expansion so use a finite-rate thermochemistry model in this last part.

Codes for estimating flow conditions (used at UQ)

- ESTC – Equilibrium Shock Tube Conditions – McIntosh 1968
- NENZF – Nonequilibrium Nozzle Flow – Lordi *et al.* 1965
- STUBE – Nonequilibrium chemistry in tube and nozzle – Vardavas 1984
- Sharc – Axisymmetric, space-marching finite-volume – Brescianini and Morgan 1992
- Surf – Axisymmetric method of characteristics, fast chemistry, Martin Rein 1992
- STN – Shock Tube and Nozzle – Krek and Jacobs 1993
  A reimplementation of the interesting bits of ESTC extended to do the nozzle, for equilibrium air or ideal air/nitrogen.
- NENZFr – ESTC+NENZF *reloaded* 2011 Shock Tubes Workshop.
  A combination of ESTCj and an axisymmetric calculation of the nozzle expansion done with Eilmer3, with block-marching coordinated by a Python supervisory program.

ESTC: equilibrium shock tube conditions

Malcolm McIntosh (1945-2000)

- Malcolm K. McIntosh (1968) "Computer program for the numerical calculation of frozen and equilibrium conditions in shock tunnels.", Unpublished Technical Report from the Department of Physics, Australian National University.
- Good, but old-school, Fortran code.
- It is difficult to maintain and you are responsible for the thermo model. Have seen some dodgy behaviour, presumably because of incompatible polynomial pieces.
- M.K. McIntosh (1970) Computer programmes supersonic real gas dynamics. WRE Technical Note 180, Australian Defence Scientific Service, Weapons Research Establishment.

NENZF: nonequilibrium nozzle flow

- J.A. Lordi, R.E. Mates and J.R. Moselle (1966)
  Computer program for the numerical solution of nonequilibrium expansions of reacting gas mixtures. NASA CR-472
- Fast steady-state analysis produces simple (single number) values for nozzle-exit flow properties.
- Same thermo model as for ESTC; same responsibilities.
- Has trouble producing answers for low enthalpies.
- Several versions of the code floating around, even within the UQ group. It is essentially unmaintained.

NENZF thermo – who's fiddled with *my* polynomials

```
11111111
  2.5      E+00
  0.0      E+00  -1.1735   E+01  0.0      E+00             E-
    E- 1.0      E+00-1.492823 E+01 0.0      E+00 0.0      E+00 1
  2   0.0      E+00
  3.451483 E+00  3.088332 E-04 -4.251428 E-08  2.739295 E-12
 -5.46832  E-17  3.071269 E+00  0.0      E+00             N2
    N2 2.0      E+00-4.2163   E-01 3.35324  E+03 0.0      E+00 4
  1   0.0      E+00  3   1.43685 E+05  6   1.70475 E+05  1   1.754   E+05
  3.249473 E+00  4.963449 E-04 -6.701753 E-08  4.443339 E-12
 -1.000281 E-16  5.915022 E+00  0.0      E+00             O2
    O2 2.0      E+00 1.0745   E-01 2.23897  E+03 0.0      E+00 5
  3   0.0      E+00  2   2.2037  E+04  1   3.7725  E+04  3   1.03198 E+05
  3   1.4239   E+05
  2.563282 E+00 -3.59177  E-05  7.469208 E-09 -6.747034 E-13
  2.234019 E-17  4.000939 E+00  0.0      E+00             AR
    AR 1.0      E+00 1.86557  E+00 0.0      E+00 0.0      E+00 3
  1   0.0      E+00  5   2.66307 E+05  3   2.68042 E+05
  3.008922 E+00 -3.134625 E-04  6.311813 E-08 -4.165203 E-12
  9.334886 E-17  1.303476 E+00  1.125906 E+05              N
     N 1.0      E+00 2.9868   E-01 0.0      E+00 1.125906 E+05 5
  4   0.0      E+00  6   5.4974  E+04  4   5.5125  E+04  6   8.2455  E+04
 12   2.3821   E+05
  2.594143 E+00 -5.008914 E-05  1.199502 E-08 -8.681611 E-13
  2.1481   E-17  4.600615 E+00  5.898    E+04              O
     O 1.0      E+00 4.932    E-01 0.0      E+00 5.898    E+04 6
  5   0.0      E+00  3   4.5462  E+02  1   6.4898  E+02  5   4.5368  E+04
  1   9.6615   E+04  5   2.10907 E+05
  3.756216 E+00  2.083961 E-04 -2.639548 E-08  1.690332 E-12
 -3.611523 E-17  3.611167 E+00  2.1477   E+04             NO
    NO 2.0      E+00 5.3941   E-01 2.69918  E+03 2.1477   E+04 3
  4   0.0      E+00  2   1.257   E+05  4   1.31283 E+05
  3.397385 E+00  3.749384 E-04 -6.06203  E-08  4.637506 E-12
 -1.107704 E-16  4.200563 E+00  2.3533   E+05            NO+
   NO+ 2.0      E+00 3.7861   E-01 3.37295  E+03 2.3533   E+05 6
  1   0.0      E+00  6   1.15232 E+05  3   1.68987 E+05  6   2.08732 E+05
  2   2.0959   E+05
```

First go at writing modern code for shock tunnel analysis

ESTCj: Equilibrium Shock Tube Conditions, Junior

- State-to-state gas dynamics relations rewritten in Python.
- Equilibrium thermodynamics with either equilibrium or frozen chemistry.
- Call out to CEA2 for thermodynamics of a reacting gas in chemical-equilibrium, so we have outsourced most of the responsibility for maintaining the polynomials.
- 2002 DLR report appendix

NENZFr: Nonequilibrium Nozzle Flow, Reloaded

- Built on ESTCj and Eilmer3, block-marching mode.
- Python program coordinates running of Eilmer3 on subsets of blocks.
- Equilibrium thermochemistry via LUT or finite-rate chemistry.
- Thermochemistry uses the CEA2 thermo polynomials.
- 2011 Shock Tubes Workshop presentation.

New codes for shock tunnel analysis

Built on the gas dynamics toolkit library and Eilmer4.

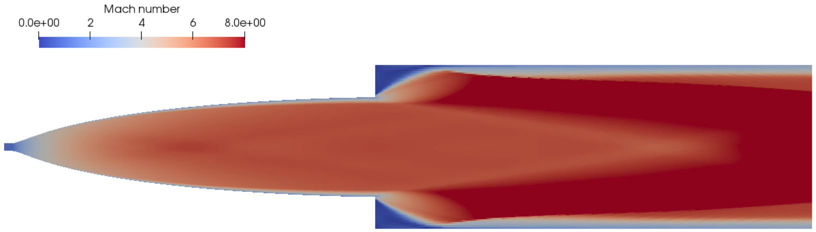
nenzf2d: Nonequilibrium nozzle flow, axisymmetric 2D

- State-to-state calculations to the sonic condition at nozzle throat.
- Eilmer4 simulation of the expanding (supersonic) part of the nozzle.
- Block-marching mode to accelerate the transient 2D solver.
- Run time on order of several hours, depending on resolution and thermochemical model.
- A better steady-state solver is coming.

nenzf1d: state-to-state with quasi1D nozzle expansion

- State-to-state calculations to the sonic condition at nozzle throat.
- Steady-state, quasi-one-dimensional calculation of the expanding part of the nozzle.
- Run time on order of a few seconds, but you need to know when to stop the expansion.

nenzf2d: Nonequilibrium nozzle flow, 2D axisymmetric



- It is *just* a normal simulation for Eilmer4.
- State-to-state analysis of the equilibrium gas to the sonic throat condition is handled with a library for shocked and isentropic flow relations; called from within the input script.
- Supersonic (expansion) part of the nozzle is handled as the main simulation task.
- Computation time just for nozzle expansion is 2.5 hours on 4 MPI tasks for LUT gas model; nearly 4.5 hours for finite-rate chemistry.

nenzf2d: state-to-state calculation

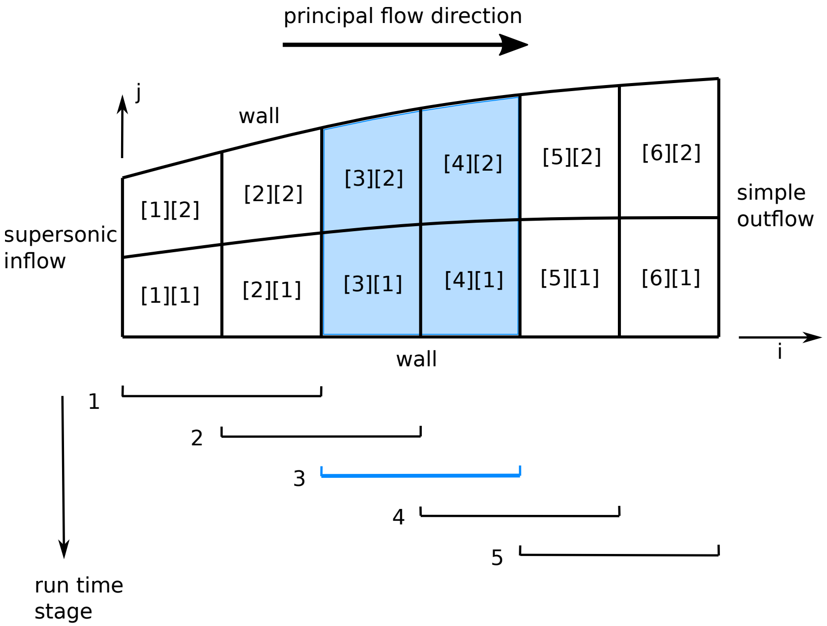
Part of the Eilmer4 input script replaces ESTCj.

```
-- Get flow conditions at the nozzle throat by performing ESTCj-like calculations,
-- using the CEA-backed gas model.
print("shock-tube fill conditions")
gm = GasModel:new{'cea-air13species-gas-model.lua'}
state1 = GasState:new{gm}
state1.p = 200.0e3; state1.T = 300.0
gm:updateThermoFromPT(state1); gm:updateTransCoeffs(state1)
--
print("normal shock, given shock speed")
Vs = 1678.62
state2, V2, Vg = gasflow.normal_shock(state1, Vs)
gm:updateThermoFromPT(state2); gm:updateTransCoeffs(state2)
--
state5, Vr = gasflow.reflected_shock(state2, Vg)
gm:updateThermoFromPT(state5); gm:updateTransCoeffs(state5)
--
print("Expand from stagnation (with ratio of pressure to match observation)")
state5s, V5s = gasflow.expand_from_stagnation(state5, 19.3253e6/state5.p)
gm:updateThermoFromPT(state5s); gm:updateTransCoeffs(state5s)
print("state5s:"); printValues(state5s)
print("(h5s-h1)=", gm:enthalpy(state5s) - gm:enthalpy(state1))
--
state6, V6 = gasflow.expand_to_mach(state5s, 1.0001)
gm:updateThermoFromPT(state6); gm:updateTransCoeffs(state6)
```

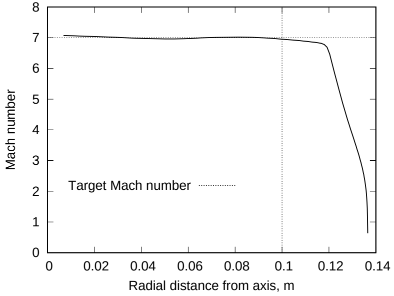
nenzf2d: expanding (supersonic) part of nozzle

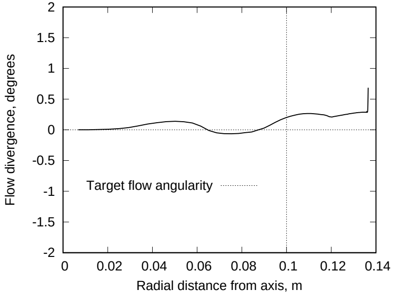
```
-- then, populate table containing Bezier points.
bez_points = {}
for npoints=1,#bezx do
   bez_points[#bez_points+1] = Vector3:new{x=bezx[npoints], y=bezy[npoints]}
end
-- and define some critical geometrical points of this nozzle.
L_thr = 0.02                 -- axial length of throat
x_start = bez_points[1].x    -- start of supersonic expansion
y_throat =  bez_points[1].y  -- throat radius

-- The throat-region is the constant-area section up to the
-- start of the contoured expansion.
throat_region = CoonsPatch:new{
   p00=Vector3:new{x=-L_thr, y=0.0},
   p10=Vector3:new{x=x_start, y=0.0},
   p11=Vector3:new{x=x_start, y=y_throat},
   p01=Vector3:new{x=-L_thr, y=y_throat}}
-- Supersonic expansion is fully defined by its north edge
nozzle_profile =
   ArcLengthParameterizedPath:new{underlying_path=Bezier:new{points=bez_points}}
exp_region = NozzleExpansionPatch:new{north=nozzle_profile}
-- Define structured grids for both regions.
print "Building grid."
x_cf_throat = RobertsFunction:new{end0=true, end1=true, beta=1.53}
x_cf = RobertsFunction:new{end0=true, end1=false, beta=1.1} --1.1
y_cf = RobertsFunction:new{end0=false, end1=true, beta=1.01}
throat_grid = StructuredGrid:new{psurface=throat_region, niv=31, njv=81,
                                 cfList={west=y_cf, east=y_cf,
                                         south=x_cf_throat, north=x_cf_throat}}
exp_grid = StructuredGrid:new{psurface=exp_region, niv=601, njv=81,
                              cfList={west=y_cf, east=y_cf,
                                      south=x_cf, north=x_cf}}
```

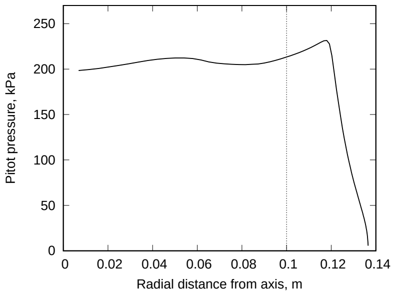
Eilmer4 block marching

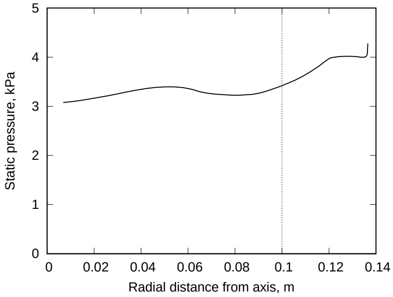


nenzf2d: nozzle-exit profile (equilibrium, LUT gas)

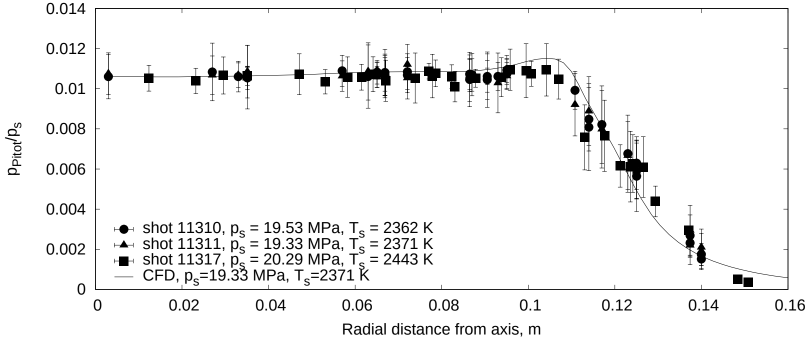




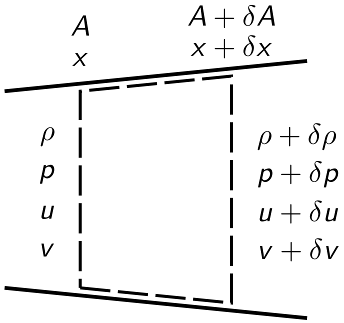




nenzf2d: nozzle-exit profile compared with measurements



Nonequilibrium nozzle flow, quasi-one-dimensional, steady



- Conservation of mass can be written for the control volume.

$$\rho\, v\, A = (\rho + \delta\rho)(v + \delta v)(A + \delta A)$$

Discard higher-order terms to get a linear constraint equation.

$$0 = \rho\, A\, \delta v + v\, A\, \delta\rho + \rho\, v\, \delta A$$

Nonequilibrium nozzle flow, quasi-one-dimensional, steady

- Conservation of momentum

$$\begin{aligned}\rho\, v^2\, A + p\, A + (p + \delta p/2)\, \delta A &= (\rho + \delta\rho)(v + \delta v)^2 (A + \delta A) + \\ &\quad (p + \delta p)(A + \delta A)\end{aligned}$$

reduces to the linear equation

$$0 = \rho\, v\, \delta v + \delta p$$

- Conservation of energy

$$\begin{aligned}\rho\, v\, A\, E + p\, A\, v &= (\rho + \delta\rho)(v + \delta v)(A + \delta A)(E + \delta E) + \\ &\quad (p + \delta p)(A + \delta A)(v + \delta v)\end{aligned}$$

reduces to the linear equation

$$0 = v\, E\, A\, \delta\rho + (\rho\, E + p)\, A\, \delta v + \rho\, v\, A\, \delta u + (\rho\, E + p)\, v\, \delta A$$

where $E = u + \frac{1}{2} v^2$ and $\delta E = \delta u + v\, \delta v$

Nonequilibrium nozzle flow, quasi-one-dimensional, steady

- If the flow quantities at x are given, we now have three linear equations in the four unknown quantities $\delta\rho$, δp, δu and δv. We close the system with the equation of state $p = f(\rho, u)$ to get the fourth linear equation

$$0 = \left.\frac{\partial f}{\partial \rho}\right|_u \delta\rho + \left.\frac{\partial f}{\partial u}\right|_\rho \delta u - \delta p \quad .$$

- Now, consider the overall change for internal energy and pressure to be partly chemical change and partly gas-dynamic accommodation

$$\delta u = \delta u_{chem} + \delta u_{gda} \quad , \delta p = \delta p_{chem} + \delta p_{gda}$$

  and let the finite-rate chemical update occur over a time-step δt, with $\delta x = v\,\delta t$, to get δp_{chem} and δu_{chem}.

Nonequilibrium nozzle flow, quasi-one-dimensional, steady

- The linear constraint equations, in matrix form, are

$$\begin{bmatrix} v\,A & \rho\,A & 0 & 0 \\ 0 & \rho v & 1 & 0 \\ v\,E\,A & \rho(E+p)A & 0 & \rho\,v\,A \\ \frac{\partial f}{\partial \rho} & 0 & -1 & \frac{\partial f}{\partial u} \end{bmatrix} \cdot \begin{bmatrix} \delta\rho \\ \delta v \\ \delta p_{gda} \\ \delta u_{gda} \end{bmatrix}$$

$$= \begin{bmatrix} -\rho\,v\,\delta A \\ -\delta p_{chem} \\ -\rho\,v\,A\,\delta u_{chem} - v\,(\rho\,E + p)\,\delta A \\ 0 \end{bmatrix}$$

- If the chemistry update is behaving well, $\delta u_{chem} = 0$ for the isolated reactor model.
- Solve the linear system to get the increments and add them to the state variables. Rinse and repeat.

nenzf1d: input script (t4m7-air.yaml in examples)

```
title: "T4 shot 11311 with Mach 7 nozzle."    # Any string will do.

species: ['N2', 'O2', 'N', 'O', 'NO']         # List
molef: {'N2': 0.79, 'O2': 0.21}               # Map of nonzero values will suffice.
# Gas model and reactions files need to be consistent with the species above.
# Gas model 1 is usually a CEAGas model file.
# Gas model 2 is a thermally-perfect gas model for the finite-rate chemistry.
gas-model-1: cea-air5species-gas-model.lua
gas-model-2: air-5sp-1T.lua
reactions: air-5sp-1T-reactions.lua

# Observed parameter values for shock-tube operation from Table 1 in Appendix A.
T1: 300         # K
p1: 200.0e3     # Pa
Vs: 1679.0      # m/s
pe: 19.33e6     # Pa
ar: 169.2       # Mach 6 nozzle
pp_ps: 0.0105   # From Figure 8.

C: 0.96         # pPitot/(rho*v^2), a value obtained from sphere simulations.

# Define the expanding part of the nozzle as a schedule of diameters with position.
xi: [0.0000, 5.126e-3, 1.021e-2, 2.008e-2, 5.023e-2, 0.1003, 0.2004, 0.4006,
     0.6000, 0.8012, 1.0000]
di: [0.0210, 0.0220,   0.0243,   0.0303,   0.0518,   0.0855, 0.1359, 0.2005,
     0.2389, 0.2626, 0.2732]
```

nenzf1d: output

```
Begin part B: supersonic expansion
Throat condition:
  velocity    0.882075 km/s
  sound-speed 0.881194 km/s
  (v-V6)/V6   0.00943504
  pressure    10637.4 kPa
  density     17.742 kg/m^3
  temperature 2080.35 K
  massf[N2]   0.762544
  massf[O2]   0.227701
  massf[N]    0
  massf[O]    2.94832e-05
  massf[NO]   0.00972495
```

```
Exit condition:
  x           0.946781 m
  area-ratio  165.75
  velocity    2.24019 km/s
  Mach        7.08211
  p_pitot     202.964 kPa
  pressure    3.01241 kPa
  density     0.0421286 kg/m^3
  temperature 248.108 K
  massf[N2]   0.762544
  massf[O2]   0.227703
  massf[N]    0
  massf[O]    2.80863e-05
  massf[NO]   0.00972495
Expansion error-indicators:
  relerr-mass 0.000440757
  relerr-H    4.17032e-06
```

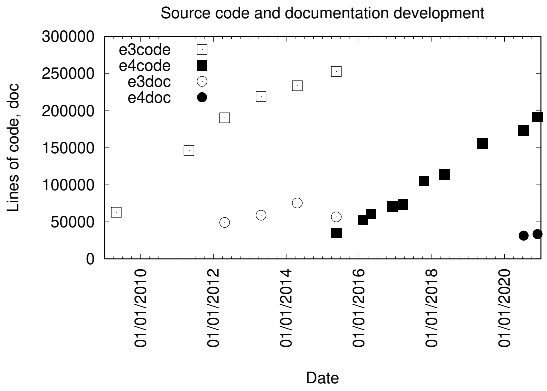
nenzf1d compared with nenzf2d

Compare exit condition 11311

Quantity	units	nenzf1d	e4-avg	e4-mass-avg
vel.x	km/s	2.24	2.237	2.236
Mach		7.08	7	7
p_Pitot	kPa	203	204	205
pressure	kPa	3.01	3.21	3.23
density	kg/m^3	0.0421	0.044	0.0443
T	K	248.1	252.7	253.3
massf-N2		0.7625	0.7625	0.7625
massf-O2		0.2277	0.2277	0.2277
massf-N		0	0	0
massf-O		2.81E-05	2.89E-05	2.89E-05
massf-NO		0.00972	0.00971	0.00971

- Species mass fractions, Mach number, temperature and velocity are very good; density and pressure are fairly good.
- Remember that, for 1D analysis, you need to know how your nozzle's boundary layers behave. Here, this is encoded as the Pitot pressure to supply-pressure ratio.

Development progress of the code. Are we there yet?



To quote Bill Gates:

- "Measuring programming progress by lines of code is like measuring aircraft building progress by weight."
- "There are no significant bugs in our released software that any significant number of users want fixed."