Contents

III   Putting DNS on hold, let's try LES with a proper LES mesh first

Links for Meshing Guidelines

References:

Komen, E., Shams, A., Camilo, L., & Koren, B. (2014). Quasi-DNS capabilities of OpenFOAM for different mesh types. *Computers & Fluids*, *96*, 87–104. https://doi.org/https://doi.org/10.1016/j.compfluid.2014.02.013

Ding, P., Wang, S., & Chen, K. (2020). Numerical study on turbulent mixed convection in a vertical plane channel using hybrid RANS/LES and LES models. *Chinese Journal of Chemical Engineering*, *28*(1), 1–8. https://doi.org/https://doi.org/10.1016/j.cjche.2019.

Kasagi, N., & Nishimura, M. (1997). Direct numerical simulation of combined forced and natural turbulent convection in a vertical plane channel. *International Journal of Heat and Fluid Flow*, *18*(1), 88–99.

Pope, S. B. (2001). *Turbulent flows.* IOP Publishing.

Tu, J., Yeoh, G. H., & Liu, C. (2018). *Computational fluid dynamics: a practical approach.* Butterworth-Heinemann.

Spalart, P. R., Deck, S., Shur, M. L., Squires, K. D., Strelets, M. K., & Travin, A. (2006). A new version of detached-eddy simulation, resistant to ambiguous grid densities. *Theoretical and Computational Fluid Dynamics*, *20*(3), 181.

Grötzbach, G. 1983. Spatial resolution requirements for direct numerical simulation of Rayleigh-Bénard convection. J. Comput. Phys., 49: 241–264.

How to size a mesh properly for RANS, LES, DNS?

Part I

How to size a mesh properly for RANS, LES, DNS?

Forced convection (and normal flows):

- Firstly for free shear regions (important for DNS)

  - Kolmogorov scales

- Secondly, for BL flows

  Natural convection:

  Mixed Convection:

  First off: forced convection or forced flow (fluid mechanics only)
  If we want to resolve everything, we use DNS (no model)
  What is the length scale of the grid? Kolmogorov scales

1 What are Kolmogorov scales?

These are scales of the smallest eddies...
How do we determine this?

Consider a rough energy balance (order of magnitude analysis)

Kinetic energy (mean flow) kinetic energy in turbulence (largest eddy)

$$rate\,of\,KE\,input \sim \frac{k}{t} = \frac{\frac{1}{2}u_{mean}^2}{t}$$

What is the timescale?

$$t \sim \frac{l}{u_{mean}}$$

L is characteristic length scale of largest eddy, usually comparable to pipe length, or if flow over a sphere, sphere diameter.

$$rate\,of\,KE\,input \sim \frac{k}{t} = \frac{\frac{1}{2}u_{mean}^2}{\frac{l}{u_{mean}}}$$

$$rate\,of\,KE\,input \sim \frac{k}{t} = \frac{\frac{1}{2}u_{mean}^3}{l}$$

We also have KE dissipation,

$$\epsilon = rate\,of\,KE\,dissipation \sim \frac{k}{t}$$

We say that

$$k = \frac{1}{2}u_{smallest}^2$$

This is velocity scale of smallest eddy, defined by characteristic time and length scale.

Let's call the legnthscale η , timescale τ_η , $u_{smallest} = u_\eta$

What's the timescale we are talking about for dissipation?

Timescale here is determined by kinematic viscosity

$$\nu \left[\frac{m^2}{s} \right]$$

Now to make up for the units, we need an appropriate length scale:

$$\tau_\eta = \frac{\eta^2}{\nu}$$

Thus, this becomes our timescale.

Kinetic energy becomes

$$k = \frac{1}{2}u_\eta^2$$

$$\epsilon \sim \frac{k}{t} = \frac{u_\eta^2}{\tau_\eta}$$

$$\epsilon \sim \frac{k}{t} = \frac{u_\eta^2}{\frac{\eta^2}{\nu}}$$

$$\epsilon \sim \nu \frac{u_\eta^2}{\eta^2}$$

All right that's a lot of variables... what use is it to us? We know ν , for ϵ we can scale

Rate of energy addition = Rate of energy dissipation

$$\epsilon \sim \frac{u_{mean}^3}{l}$$

So those are based on <u>macroscopic</u> properties

So ν and ϵ can be scaled based on macroscopic properties. We can find out the velocity, length and time scales based on these.

So let's do length scale first:

$$\epsilon \sim \nu \frac{u_\eta^2}{\eta^2}$$

We make a reasonable assumption

$$Re \sim 1$$

$$Re = \frac{u_\eta \eta}{\nu}$$

This means

$$\nu \sim u_\eta \eta$$

Or

$$u_\eta \sim \frac{\nu}{\eta}$$

$$\epsilon \sim \nu \frac{\left(\frac{\nu}{\eta}\right)^2}{\eta^2}$$

$$\epsilon \sim \frac{\nu^3}{\eta^4}$$

$$\eta \sim \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}$$

8

This is known as the Kolmogorov scale...

We can do the same for velocity

$$u_\eta \sim \frac{\nu}{\eta} = \frac{\nu}{\left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}} = (\epsilon\nu)^{\frac{1}{4}}$$

$$\tau_\eta = \frac{\eta}{u_\eta} = \frac{\left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}}{(\epsilon\nu)^{\frac{1}{4}}} = \left(\frac{\nu}{\epsilon}\right)^{\frac{1}{2}}$$

Ok great, but let's make it convenient for ourselves, scale it with Reynold's number

$$\epsilon \sim \frac{u_{mean}^3}{l}$$

Let's do it for length scale first:

$$\eta \sim \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}$$

$$\eta \sim \left(\frac{\nu^3}{\frac{u_{mean}^3}{l}}\right)^{\frac{1}{4}}$$

We can say $Re = \frac{u_{mean}l}{\nu}$

So let's make the inside look more favourable...

$$\eta \sim \left(\frac{\nu^3}{\frac{u_{mean}^3 l^3}{l^4}}\right)^{\frac{1}{4}}$$

$$\eta \sim \left(\frac{\nu^3}{u_{mean}^3 l^3}\right)^{\frac{1}{4}} l$$

$$\eta \sim \left(\frac{1}{Re^3}\right)^{\frac{1}{4}} l$$

$$\frac{\eta}{l} \sim Re^{-\frac{3}{4}}$$

Do the same for the rest and we find:

$$\frac{u_\eta}{u_0} \sim Re^{-\frac{1}{4}}$$

$$\frac{\tau_\eta}{\tau_0} \sim Re^{-\frac{1}{2}}$$

9

note that we can estimate the number of cells needed based on this:

no. of cells $= \frac{V_{total}}{V_{Cell}}$

$$V_{cell} \sim \eta^3$$

$$V_{total} \sim l^3$$

$$n_{cells} \sim \frac{l^3}{\eta^3} = Re^{\frac{9}{4}}$$

Hence for DNS, in free shear region, this usually means:

$$\frac{\eta}{l} \sim Re^{-\frac{3}{4}}$$

So let's say u have a 1m ish domain,

Re=10,000

Whats $o\left(\eta\right)$?

$$\eta \sim 1m * \left(10000\right)^{-\frac{3}{4}} = 0.001m \, or \, 1mm$$

What about if Re is 2000 (remember this is transistion!)

$$\eta \sim 1m * \left(2000\right)^{-\frac{3}{4}} = 0.00334m\left(3mm\right)$$

Of course, this η depends on the domain size. . .

1.1 For scales closer to the wall, you need to pay more attention.

Good rule of thumb (Komen, Shams, Camilo, and Koren, 2014) :

In wall normal direction: $\Delta y^+ \sim 0.05 - 0.5$

In wall parallel direction: $\Delta z^+, \Delta x^+ \sim 5$

$$\Delta y^+ = \frac{\Delta y u_\tau}{\nu}$$

Now that we know the standard DNS scales, we can look to LES and RANS for reference:

Always make sure CFL is less than 1 for best results (stability)

2 LES grid sizing

Note in the PL-DDES paper for comparison in direction perpendicular to wall, $\Delta y^+ = 0.5$, equidistant for mixed convection, $\Delta x^+ = 40$, $\Delta z^+ = 20$, is typical of LES grids (Ding, Wang, and Chen, 2020).

Usually for forced flows, BL $= \delta$, $\Delta x^+ \approx \frac{1}{10}\delta$

(Spalart et al., 2006)

$\delta \rightarrow y^+ = 500$ (Tu, Yeoh, and Liu, 2018)

Shishkina, O., & Wagner, C. (2012). A numerical study of turbulent mixed convection in an enclosure with heated rectangular elements. Journal of Turbulence, (13), N22.

https://www.tandfonline.com/doi/full/10.1080/14685248.2012.677541?casa_token=3jNH8Q70aH
3AKCPlWHeKeQh13yrLYTQhy6HG-gZzr99FjKTvuij3VwOeD0kvTX13cLf9lY3Ezc8oBTgCz2e29

Looking at this, a safe estimate is that les grid size is about 10x Kolmogorov length scale.

2.1 Wall functions (RANS)

We generally have wall functions in the log law region $y^+ > 5$. Hence first mesh point should be in order of $y^+ > 5$ (Tu et al., 2018). Usually y^+ from 20 to 30 will do as a first start point, and have 8-10 mesh points in turb BL (y^+ up to 500 ish) (Tu et al., 2018).

Can check if mesh is fine enough by refining mesh and comparing plotting ν_t to ν in BL. If no change then ok, only works for RANS though.

What about in parallel directions? (Spalart et al., 2006)

Usually in parallel direction for RANS, the parallel spacing would exceed the boundary layer thickness.

And this thickness is about $y^+ = 500$

So typical spacing in Δx^+, Δz^+ direction would be 500 and up.

2.2 No Wall functions (low Re RANS)

First mesh point at $y^+ < 1$, 5-10 nodal points from first point to $y^+ = 20$. Another 20-50 points in turbulent BL (y^+ from 20 to ~ 500ish??) for total of 30-60 mesh points in BL for adequate mesh resolution without wall function.

We can see from this order of magnitude estimate:

$$\Delta y^+ \sim 50$$

Whereas for DNS $\Delta y^+ \sim 0.05 - 0.5$

So this means RANS mesh length is about 100-1000 times bigger than Kolmogorov length scale.

In free shear region for Re ~ 10000, $l \sim 1m$

$$\eta \sim l * Re^{-\frac{3}{4}}$$

$$\eta \sim 0.001m$$

$$l_{RANS} = 100 * \eta = 0.1m$$

This is just a rule of thumb. But should yield good result.

3   Natural convection

http://thermopedia.com/content/1076/

https://en.wikipedia.org/wiki/Rayleigh_number

Scheel, J. D., Emran, M. S., & Schumacher, J. (2013). Resolving the fine-scale structure in turbulent **Rayleigh–Bénard convection**. New Journal of Physics, 15(11), 113063.

Perry, R. H., & Green, D. W. (2015). Perry's chemical engineers' handbook. *Mc Graw.*

3.1   Rayleigh–Bénard convection:
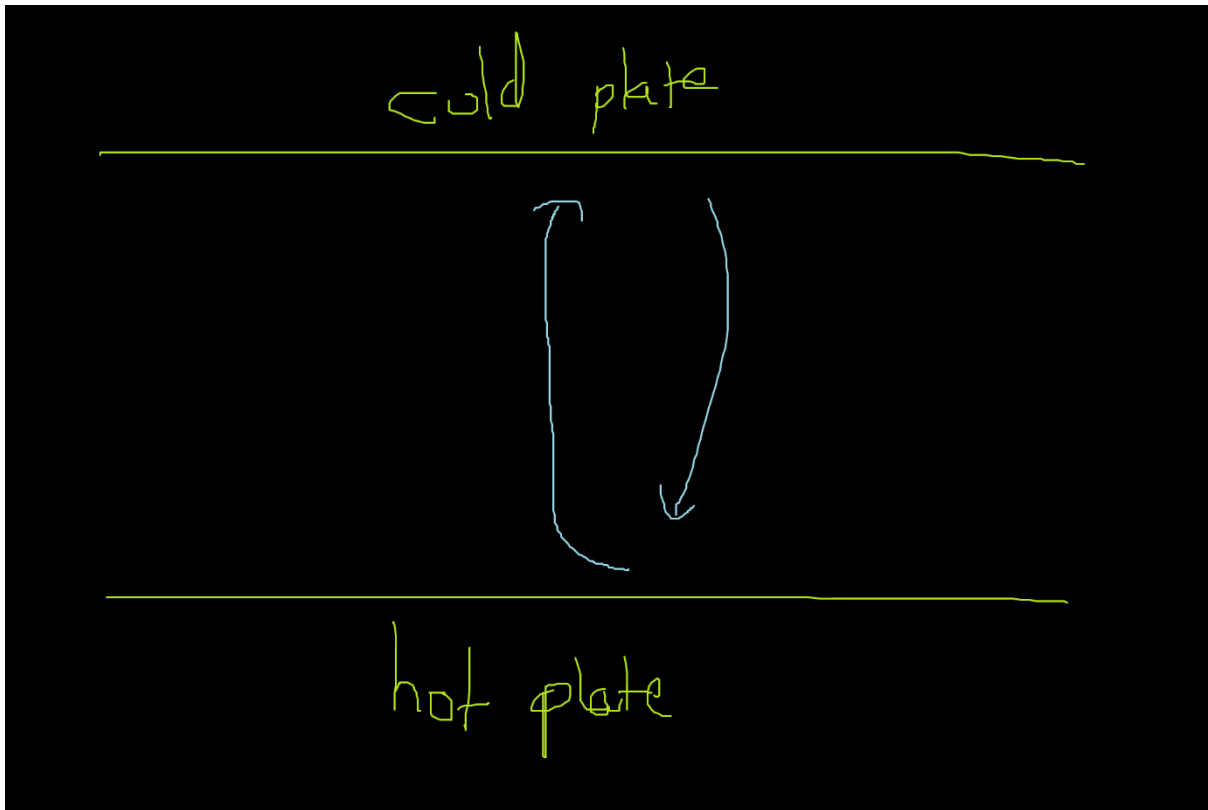
Recall for forced flows:

$$\frac{\eta}{l} \sim \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}$$
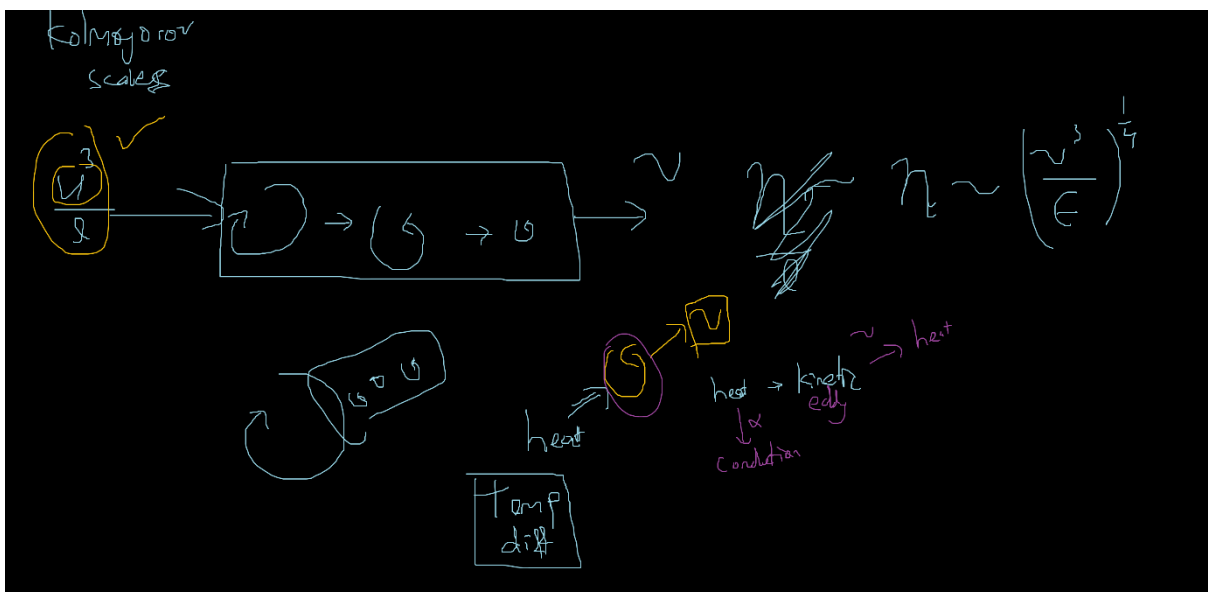
For natural convection, the story is different...

- In the smallest eddies, have two dissipation mechanisms now:

    ○ α thermal conduction

    ○ ν viscous dissipation

    ○ No eddy diffusivity or eddy heat flux to dissipate smallest eddies

This also means there are two types scales now involved, the Kolmogorov scale (ν based dissipation), and the Batchelor scale (α based dissipation).

$$\eta_B = \frac{\eta_{kolmogorov}}{\sqrt{Pr}}$$

For large Pr, batchelor scale dominates.(Scheel, Emran, and Schumacher, 2013)

3.1.1 Kolmogorov Scale(Scheel et al., 2013)

$$\frac{\eta}{l} \sim \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}$$

Skipping derivation, but we nondimensionalize ϵ to get:

Also apply characteristic length l=H (cell height)

$$\frac{\eta_k}{H} = \frac{Pr^{\frac{3}{8}}}{Ra^{\frac{3}{8}}} \frac{1}{\widetilde{\epsilon}^{\frac{1}{4}}}$$

Note: $\widetilde{\epsilon}^{\frac{1}{4}}$ is dimensionless

On average,

$$\widetilde{\langle\epsilon\rangle} = \frac{Nu}{\sqrt{RaPr}}$$

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{(Nu-1)\,Ra}\right)^{\frac{1}{4}}$$

Evaluate dimensionless numbers (fluid properties) at $T_{film} = \frac{T_S + T_\infty}{2}$

3.1.2 Batchelor scale (Scheel et al., 2013):

$$\eta_B = \frac{\eta_k}{\sqrt{Pr}}$$

$$\frac{\eta_B}{H} = \frac{Pr^{\frac{1}{8}}}{Ra^{\frac{3}{8}}} \frac{1}{\tilde{\epsilon}^{\frac{1}{4}}}$$

$$\frac{\eta_B}{H} = \left(\frac{1}{(Nu - 1) Ra} \right)^{\frac{1}{4}}$$

As to which is smaller depends on Pr

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{(Nu - 1) Ra} \right)^{\frac{1}{4}} \; ; \; \frac{\eta_B}{H} = \left(\frac{1}{(Nu - 1) Ra} \right)^{\frac{1}{4}}$$

For large Pr, batchelor scale is smaller.

For small Pr, Kolmogorov scale is smaller.

Pre-requisite: need to know Nu, Pr and Ra in order to know the length scale size. . .

Also note, these scales are for natural convection, which is <u>near the wall</u> anyhow. Estimates with wall function aren't as necessary.

A rough scaling for Nu : like Re, it depends on the situation.

Eg. Flat plate flow, high Pr (Bejan, 2013)

$$Nu \sim Ra_H^{\frac{1}{3}}$$

For Rayleigh Benard convection (Perry and Green, 2015)

$$Nu = Ra_H^{\frac{1}{3}}$$

4   Mixed convection (free shear)

Just an estimate:

Three scales we discussed:

$$\frac{\eta}{l} \sim Re^{-\frac{3}{4}}$$

$$\frac{\eta_B}{H} = \left(\frac{1}{(Nu - 1) Ra} \right)^{\frac{1}{4}}$$

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{(Nu - 1)\, Ra} \right)^{\frac{1}{4}}$$

You could pick the smallest of the three to start. . .

(25 June 2020) Note that Batchelor scales are also important in forced convection, forgot to add this. $\eta_B = \frac{\eta_k}{\sqrt{Pr}}$ Please take note.

In literature, most mesh sizes are listed in terms of wall scale

$$Ri = \frac{Gr}{Re^2}$$

In literature:

Base your grid spacing on wall spacing estimates (done in literature) (You, Yoo, and Choi, 2003)

Eg. For a tube:

$$0.17 \leq \Delta r^+ \leq 5.1$$

$$\Delta \phi^+ \approx 8.85$$

$$7 \leq \Delta x^+ \leq 10.5$$

These are basically r, θ, z coordinates.

But it gives you an idea of the grid spacing needed!

5 Friction Velocity Estimates for Forced and Natural Flows

U need wall shear stress to predict friction velocity:

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}}$$

$$y^+ = \frac{y u_\tau}{\nu}$$

Standard pipe correlations (moody chart predicting Fanning's friction factor)

$$f = \frac{\tau_{wall}}{\frac{1}{2}\rho u^2_{mean}}$$

From Churchill (in perry's chemical engineering handbook) (Perry and Green, 2015)

$$\frac{1}{\sqrt{f}} = -4\log_{10}\left(0.27\frac{\epsilon}{D} + \left(\frac{7}{Re_{mean}}\right)^{0.9}\right)$$

For smooth pipes, surface roughness =0 (for most DNS we ignore this anyhow)

Check whether it's $\ln x$ or $\log_{10} x$?

Quick calculation:

$$Re_{mean} = 5,000$$

$$Re = \frac{uD}{\nu}$$

From graph: $f \approx 0.0091$

Excel calculated value

$$\frac{1}{\sqrt{f}} = -4\log_{10}\left(\left(\frac{7}{5000}\right)^{0.9}\right) = 10.2739$$

$$f = \left(\frac{1}{10.2739}\right)^2 = 0.00947384$$

Good match!

Can we correlate Re_{mean} to u_τ for smooth pipes?

Absolutely!!

$$f = \frac{\tau_{wall}}{\frac{1}{2}\rho u^2_{mean}}$$

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}}$$

$$u_\tau = \sqrt{\frac{\frac{1}{2}\rho u^2_{mean}f}{\rho}}$$

17

$$u_\tau = \sqrt{\frac{1}{2} u_{mean}^2 f}$$

$$u_\tau = u_{mean} \sqrt{\frac{1}{2} f}$$

Noting:

$$\frac{1}{\sqrt{f}} = -4 \log_{10} \left(0.27 \frac{\epsilon}{D} + \left(\frac{7}{Re_{mean}} \right)^{0.9} \right)$$

Tada!!

$$u_\tau = u_{mean} \sqrt{\frac{1}{2}} * \frac{1}{-4 \log_{10} \left(0.27 \frac{\epsilon}{D} + \left(\frac{7}{Re_{mean}} \right)^{0.9} \right)}$$

And if we want to correlate Re_τ in terms of Re_{mean}

Provided length scale is the same, (which is should be)

$$\frac{u_\tau L}{\nu} = \frac{u_{mean} L}{\nu} \sqrt{\frac{1}{2}} * \frac{1}{-4 \log_{10} \left(0.27 \frac{\epsilon}{D} + \left(\frac{7}{Re_{mean}} \right)^{0.9} \right)}$$

If L=D (pipe diameter)

Then we define:

$$Re_\tau = \frac{u_\tau D}{\nu}$$

Tada!!

$$Re_\tau = Re_{mean} \sqrt{\frac{1}{2}} * \frac{1}{-4 \log_{10} \left(0.27 \frac{\epsilon}{D} + \left(\frac{7}{Re_{mean}} \right)^{0.9} \right)}$$

Okay if we don't want to remember logarithm and everything. . . and simplify for smooth pipes:

$$Re_\tau = Re_{mean} \sqrt{\frac{1}{2}} * \frac{1}{-4 \log_{10} \left(\left(\frac{7}{Re_{mean}} \right)^{0.9} \right)}$$

18

Plot in excel. . .

Again you have to be careful what length scale you are talking about before u use this formula!!!
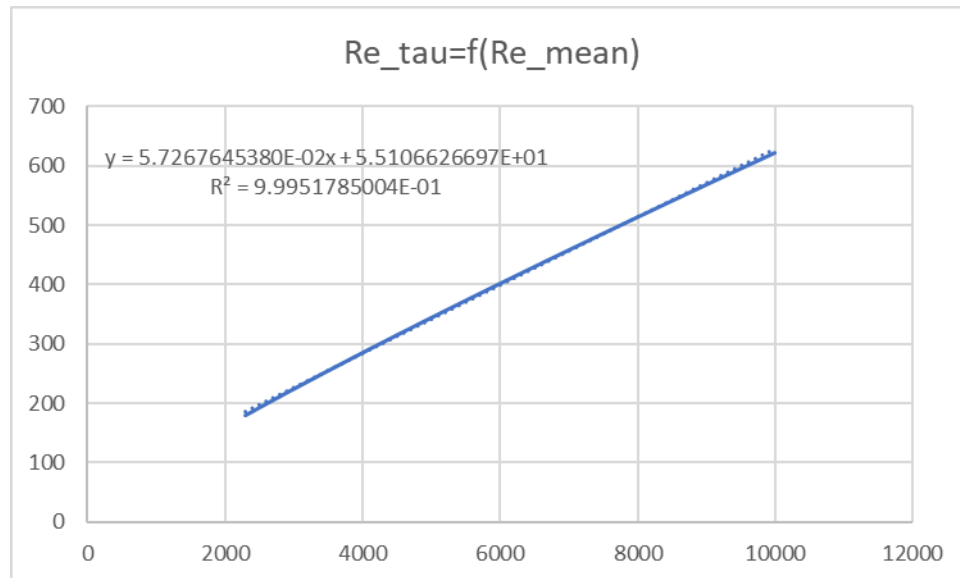
$$Re_\tau = 5.72676 * 10^{-2} * Re_{mean} + 5.51066$$

(5 d.p)

This is a linear graph for smooth pipes, so u can get Re_τ and u_τ estimate for DNS data. . .

So what is x^+?

$$x^+ = \frac{xu_\tau}{\nu}$$

But

$$Re_\tau = \frac{u_\tau L}{\nu}$$

We can write in this manner:

$$x^+ = x * \frac{Re_\tau}{L}$$

Very simple formula if you know your Re_τ beforehand.

So let's say we have $Re_\tau = 180$ (corresponds to $Re_{mean} \approx 2300$)

What is the how small should your DNS mesh be for 0.1m diameter pipe?

$\Delta x^+ \approx 0.05\,(smallest\,DNS\,mesh)$

$$x = \frac{1}{\frac{Re_\tau}{L}} * x^+$$

$$x = \frac{1}{\frac{180}{0.1}} * 0.05 = 2.777 * 10^{-5}m = 0.02777mm$$

We are talking in microns here!

So small. . .

And how thick is the VSL region?

$$x^+ \approx 5$$

$$x = \frac{1}{\frac{180}{0.1}} * 5 = 2.777 * 10^{-3}m \approx 2.777mm$$

That's quite thin!

And the BL thickness estimate? x^+ or $y^+ = 500$

$$x = \frac{1}{\frac{180}{0.1}} * 500 = 2.777 * 10^{-1}m \approx 27.8cm$$

That's where we get it!

How does Δx^+ compare to Kolmogorov scale? $Re_{mean} \approx 2300$

$$\frac{\eta}{D} \sim \frac{1}{Re_{mean}^{\frac{3}{4}}} = 0.003010954$$

$$\eta \sim \frac{1}{Re_{mean}^{\frac{3}{4}}} * 0.1m = 0.003010954 * 0.1m = 0.3mm$$

Kolmogorov lengthscale about 10x smaller (1 order of magnitude smaller) than VSL region!

It's about 10x bigger than the smallest DNS mesh size.

In other words, kolmmogorov lengthscale is about

$$\eta \approx \Delta x^+ * \frac{L}{Re_\tau}$$

$$\Delta x^+ \approx 1.85$$

This is for this case only, or in general <u>low Re DNS with smooth pipe</u>, but we can see the rough relationship here. . .

For mixed convection of low Richardson number, this rule should still apply.

6 Near Wall Smallest Scale Estimates for Natural Convection

Hardcore estimate (Yuan, Moser, and Suter, 1993):

	Forced Convection	Natural Convection
Distance from wall	$y^+ \equiv \frac{y u_\tau}{\nu}$	$y^* \equiv \frac{y u_q}{\alpha}$
Velocity	$u^+ \equiv \frac{u}{u_\tau}$	$u^{**} \equiv \frac{u_q^3 u}{u_\tau^4}$
Temperature	$T^+ \equiv \frac{(T_{wall}-T)u_\tau}{\left(\frac{q_{wall}}{\rho c_P}\right)}$	$T^*_{natl} \equiv \frac{(T_{wall}-T)}{T_q}$
Reference dimensionless quantities	$u_\tau \equiv \sqrt{\frac{\tau_{wall}}{\rho}}$	$u_q \equiv \left(\frac{g\beta\alpha q_{wall}}{\rho c_P}\right)^{\frac{1}{4}}$ $T_q \equiv \left(\frac{q_{wall}^3}{g\beta\alpha(\rho c_p)^3}\right)^{\frac{1}{4}}$

$$u_q \equiv \left(\frac{g\beta\alpha q_{wall}}{\rho c_P}\right)^{\frac{1}{4}}$$

Temperature length scales

$$y^* \equiv \frac{y u_q}{\alpha}$$

Velocity length scales

$$y_1^{**} = \frac{y u_q^3}{\alpha u_\tau^2} = y^* \frac{u_q^2}{u_\tau^2}$$

Problem: Estimating u_τ is a pain (correlations are sparse if anything – most people don't care much for wall shear stress in natural convection applications)

6.1 Ballpark Estimate:

- 10x smaller than smallest of Kolmogorov/Batchelor scales

- $\frac{\eta}{l} \sim Re^{-\frac{3}{4}}$

- $\frac{\eta_B}{H} = \left(\frac{1}{(Nu-1)Ra} \right)^{\frac{1}{4}}$

- $\frac{\eta_k}{H} = \left(\frac{Pr^2}{(Nu-1)Ra} \right)^{\frac{1}{4}}$

Lower Bound estimate convert natural convection into mixed convection (cheating):

6.2 Method 1: Pretend the situation is a mixed convection, set Richardson number to 0.1

- Ie $\frac{Gr}{Re^2} = 0.1$

- $Re = \sqrt{10Gr}$

- $\sqrt{10Gr} = \frac{u_{buoyancy}L}{\nu}$

6.3 Method 2: use buoyancy Velocity to estimate C_f

- Ie $Gr = Re^2$

- $\sqrt{Gr} = \frac{u_{buoyancy}L}{\nu}$

Eg. Square cavity
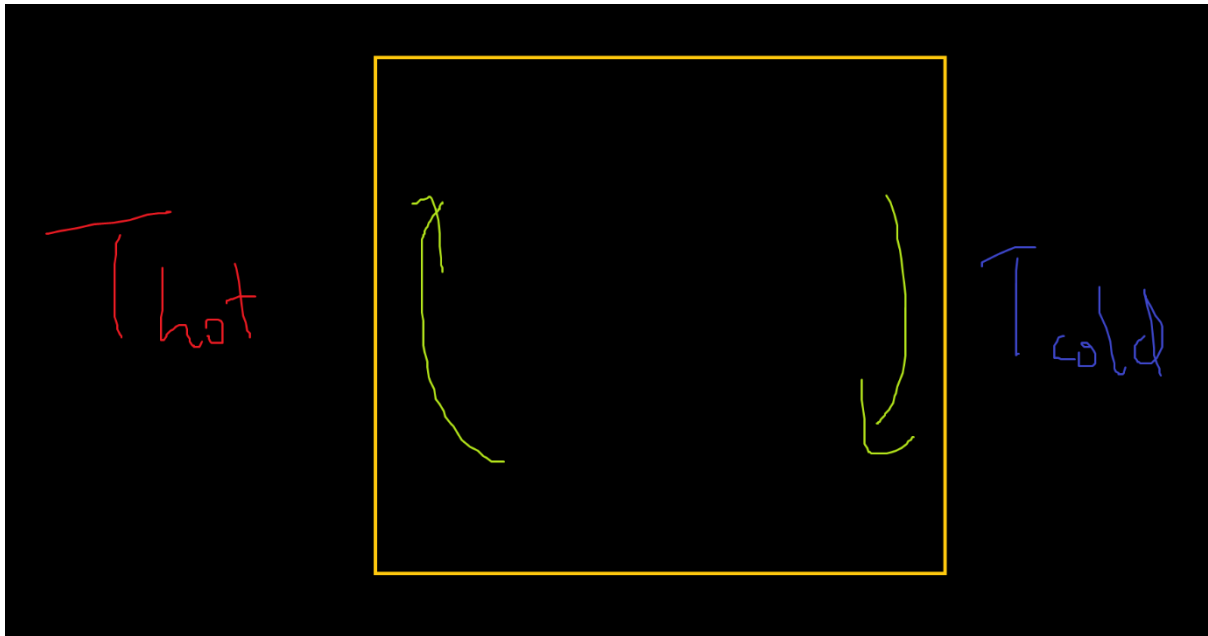
How to estimate?

Let's do it for airflow: simple example (Ampofo and Karayiannis, 2003)

$0.75 * 0.75 * 1.5 \ (\text{m}^3)$ cavity

$$T_{hot} = 50°C$$

$$T_{cold} = 10°C$$

$$Ra = 1.58 * 10^9$$

$$Ra = \frac{g\beta \left(T_{hot} - T_{cold}\right) H^3}{\nu^2} Pr$$

$$Pr = 0.71$$

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{(Nu - 1)\, Ra}\right)^{\frac{1}{4}}$$

Let's estimate the Kolmogorov scale:

$$Nu \sim Ra_H^{\frac{1}{3}}$$

$$Nu = 0.15 * Ra_H^{\frac{1}{3}}$$

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{\left(Ra_H^{\frac{1}{3}} - 1 \right) Ra} \right)^{\frac{1}{4}} = \left(\frac{0.71^2}{\left((1.58 * 10^9)^{\frac{1}{3}} - 1 \right) 1.58 * 10^9} \right)^{\frac{1}{4}}$$

Since $Ra_H^{\frac{1}{3}} \gg 1$

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{\left(Ra_H^{\frac{1}{3}} - 1 \right) Ra} \right)^{\frac{1}{4}} = \left(\frac{0.71^2}{(1.58 * 10^9)^{\frac{4}{3}}} \right)^{\frac{1}{4}} \approx 0.000723453$$

If $H = 0.75$

$$\eta_K = 0.00054259m \approx 0.5mm$$

What about the Batchelor Scale?

$$\eta_B = \frac{\eta_K}{\sqrt{Pr}} = 0.000643935m \approx 0.6mm$$

Here, Kolmogorov scales are smaller

Using minimum mesh size $=1/10$ of Kolmogorov/Batchelor scale,

$$smallest \ \Delta x \approx 0.05mm \ (wallnormal)$$

6.3.1   Worked Examples of Lower Bound Estimates:

Method 1: Pretend the situation is a mixed convection, set Richardson number to 0.1

- Ie $\frac{Gr}{Re^2} = 0.1$

- $Re = \sqrt{10Gr}$

- $\sqrt{10Gr} = \frac{u_{buoyancy} L}{\nu}$

Worked Example Method 2: use buoyancy Velocity to estimate C_f

- Ie $Gr = Re^2$

- $\sqrt{Gr} = \frac{u_{buoyancy} L}{\nu}$

Schultz Grunow Correlation (Bejan, 2013): $C_{f,x} = 0.37 * \left\{ \log_{10} \left(\frac{U_\infty x}{\nu} \right) \right\}^{-2.584}$

Id say substitute $Re_x = \frac{U_\infty x}{\nu}$ as $Re_x = \sqrt{Gr}$ (or $\sqrt{10 \, Gr}$)

Compress the formula:

$$C_{f,x} = 0.37 * \left\{ \log_{10} \left(\sqrt{10Gr} \right) \right\}^{-2.584}$$

$$C_{f,x} = 0.37 * \left\{ \log_{10} \left(\sqrt{Gr} \right) \right\}^{-2.584}$$

Estimate for skin friction...

We derived in the last video:

$$u_\tau = u_{mean} \sqrt{\frac{1}{2}f}$$

Here, if the $L = H$ for both sides, multiply, by $\frac{H}{\nu}$

$$Re_\tau = Re_{mean} \sqrt{\frac{1}{2}f}$$

Substitute:

$$Re_\tau = \sqrt{10Gr} \sqrt{\frac{1}{2}0.37 * \left\{ \log_{10} \left(\sqrt{10Gr} \right) \right\}^{-2.584}}$$

$$Re_\tau = \sqrt{Gr} \sqrt{\frac{1}{2}0.37 * \left\{ \log_{10} \left(\sqrt{Gr} \right) \right\}^{-2.584}}$$

Here,

$$Ra = 1.58 * 10^9$$

But Pr is $o(1)$ so $o(Ra) = o(Gr)$

If im just estimating order of magnitude, just substitute Ra in, otherwise of course, divide by Pr

This is quick and fast estimate

$$Re_\tau = \sqrt{1.58 * 10^{10}} \sqrt{\frac{1}{2}0.37 * \left\{ \log_{10} \left(\sqrt{1.58 * 10^{10}} \right) \right\}^{-2.584}}$$

$$Re_\tau = 6588$$

$$Re_\tau = \sqrt{1.58 * 10^9} \sqrt{\frac{1}{2} 0.37 * \left\{ \log_{10} \left(\sqrt{1.58 * 10^9} \right) \right\}^{-2.584}}$$

$$Re_\tau = 2380$$

Looks like Re_τ is quite high! But this is ultraconservative anyhow

Now let's estimate our length scale:

$$x = \Delta x^+ * \frac{L}{Re_\tau} = 0.05 * \frac{0.75}{6588} = 5.69 * 10^{-6} m \approx 0.006mm$$

(absolute smallest length scale, no smaller)

$$x = \Delta x^+ * \frac{L}{Re_\tau} = 0.05 * \frac{0.75}{2380} = 1.57 * 10^{-5} m \approx 0.02mm$$

We can see this estimate is on the same order of magnitude as 10x below Kolmogorov/Batchelor scale. Pretty good!

$$\eta_K = 0.00054259m \approx 0.5mm$$

$$\eta_B = \frac{\eta_K}{\sqrt{Pr}} = 0.000643935m \approx 0.6mm$$

Now let's compare these with the hardcore estimates, and similar techniques see these scales:

Temperature length scales

$$y^* \equiv \frac{y u_q}{\alpha}$$

Velocity length scales

$$y_1^{**} = \frac{y u_q^3}{\alpha u_\tau^2} = y^* \frac{u_q^2}{u_\tau^2}$$

$$u_q \equiv \left(\frac{g \beta \alpha q_{wall}}{\rho c_P} \right)^{\frac{1}{4}}$$

Let's try to make look more like dimensionless numbers:

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(\frac{g \beta \alpha q_{wall}}{\rho c_P} \right)^{\frac{1}{4}}$$

$$Ra_H = \frac{g\beta \left(T_H - T_C\right) H^3}{\nu\alpha}$$

$$g\beta = \nu\alpha \frac{Ra_H}{\left(T_H - T_C\right) H^3}$$

Substitute,

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(\frac{\nu\alpha \frac{Ra_H}{(T_H-T_C)H^3} \alpha q_{wall}}{\rho c_P} \right)^{\frac{1}{4}}$$

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(\frac{\nu\alpha^2 Ra_H q_{wall}}{\rho c_P H^3 \left(T_H - T_C\right)} \right)^{\frac{1}{4}}$$

What's Nu?

$$Nu_{local} = \frac{hL}{k}$$

L=H (cavity height)

$$h = \frac{q}{T_h - T_c}$$

$$Nu_{local} = \frac{q}{T_h - T_c} \frac{H}{k}$$

$$\frac{q}{T_h - T_c} = \frac{Nu_{local} k}{H}$$

Substitute!

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(\frac{\nu\alpha^2 Ra_H q_{wall}}{\rho c_P H^3 \left(T_H - T_C\right)} \right)^{\frac{1}{4}}$$

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(\frac{\nu\alpha^2 Ra_H \frac{Nu_{local}k}{H}}{\rho c_P H^3} \right)^{\frac{1}{4}}$$

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(\frac{\nu\alpha^2 Ra_H Nu_{local} k}{\rho c_P H^4} \right)^{\frac{1}{4}}$$

Substitute $\alpha = \frac{k}{\rho c_p}$

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(\frac{\nu \alpha^3 Ra_H Nu_{local}}{H^4} \right)^{\frac{1}{4}}$$

And finally, we have Pr

$$\nu = \alpha Pr$$

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(\frac{\alpha^4 Pr \, Ra_H Nu_{local}}{H^4} \right)^{\frac{1}{4}}$$

$$\frac{u_q}{\alpha} \equiv \frac{1}{\alpha} \left(Pr \, Ra_H Nu_{local} \right)^{\frac{1}{4}} \frac{\alpha}{H}$$

$$\frac{u_q}{\alpha} \equiv \frac{\left(Pr \, Ra_H Nu_{local} \right)^{\frac{1}{4}}}{H}$$

Or

$$u_q = \frac{\alpha}{H} \left(Pr \, Ra_H Nu_{local} \right)^{\frac{1}{4}}$$

This is much more palatable!

Let's check out the thermal length scale:

$$y^* \equiv \frac{y u_q}{\alpha}$$

$$y^* \equiv y \frac{\left(Pr \, Ra_H Nu_{local} \right)^{\frac{1}{4}}}{H}$$

$$y = y^* \frac{H}{\left(Pr \, Ra_H Nu_{local} \right)^{\frac{1}{4}}}$$

Assuming we want $y^* \approx 0.05$

$$y = 0.05 \frac{H}{\left(Pr \, Ra_H Nu_{local} \right)^{\frac{1}{4}}}$$

Substitute $Nu \sim Ra_H^{\frac{1}{3}}$

$$y = 0.05 \frac{H}{\left(Pr \, Ra_H^{\frac{4}{3}} \right)^{\frac{1}{4}}}$$

28

$$Ra = 1.58 * 10^9$$

$$Pr = 0.71$$

$$H = 0.75m$$

$$y_{thermal} = 0.05 \frac{0.75}{\left(0.71 \ (1.58e9)^{\frac{4}{3}}\right)^{\frac{1}{4}}} = 3.51 * 10^{-5}m \approx 0.035mm!$$

About 10x below Batchelor scale too

Let's test the momentum BL scales,

$$y_1^{**} = \frac{y u_q^3}{\alpha u_\tau^2} = y^* \frac{u_q^2}{u_\tau^2}$$

$$\Delta y^* = \frac{y u_q}{\alpha} \frac{u_q^2}{u_\tau^2}$$

Let's make life easier and formulate $\left(\frac{u_q}{u_\tau}\right)^2$ in terms of dimensionless numbers

$$Re_\tau = \frac{u_\tau L}{\nu}$$

$$u_\tau = \frac{Re_\tau \nu}{L}$$

And for u_q ,

$$u_q = \frac{\alpha}{H} \left(\text{Pr} \, Ra_H Nu_{local}\right)^{\frac{1}{4}}$$

$$\left(\frac{u_q}{u_\tau}\right)^2 = \frac{\left(\frac{\alpha}{H} \left(\text{Pr} \, Ra_H Nu_{local}\right)^{\frac{1}{4}}\right)^2}{\left(\frac{Re_\tau \nu}{L}\right)^2}$$

We note that length scales can cancel out

$$\left(\frac{u_q}{u_\tau}\right)^2 = \frac{\left(\alpha \left(\text{Pr} \, Ra_H Nu_{local}\right)^{\frac{1}{4}}\right)^2}{\left(Re_\tau \nu\right)^2}$$

And Prandtl Number appears in denominator

$$\left(\frac{u_q}{u_\tau}\right)^2 = \frac{\left((Pr\ Ra_H Nu_{local})^{\frac{1}{4}}\right)^2}{(Re_\tau Pr)^2}$$

Now life is easier!

Substitute all in:

$$\Delta y^* = \frac{yu_q}{\alpha}\frac{u_q^2}{u_\tau^2}$$

$$\Delta y^* = y\frac{(Pr\ Ra_H Nu_{local})^{\frac{1}{4}}}{H}\frac{\left((Pr\ Ra_H Nu_{local})^{\frac{1}{4}}\right)^2}{(Re_\tau Pr)^2}$$

$$\Delta y^* = \frac{y}{H}\frac{(Pr\ Ra_H Nu_{local})^{\frac{3}{4}}}{(Re_\tau Pr)^2}$$

Now let's crunch some numbers:

$$Nu_{local} \sim Ra_H^{\frac{1}{3}}$$

$$\Delta y^* = \frac{y}{H}\frac{\left(Pr\ Ra_H Ra_H^{\frac{1}{3}}\right)^{\frac{3}{4}}}{(Re_\tau Pr)^2}$$

$$\Delta y^* = \frac{y}{H}\frac{(Pr\ )^{\frac{3}{4}}\ Ra_H}{(Re_\tau Pr)^2}$$

$$y = \Delta y^* H\left(\frac{(Re_\tau Pr)^2}{(Pr\ )^{\frac{3}{4}}\ Ra_H}\right)$$

Substitute:

$$\Delta y^* \approx 0.05$$

$$Re_\tau = 2380\ or\ 6588$$

$$\Delta y = 0.05 * 0.75\left(\frac{(2380 * 0.71)^2}{(0.71)^{\frac{3}{4}}\ (1.58e9)}\right)$$

30

$$\Delta y = 8.77E - 05 \approx 0.08mm$$

Looks like our rule of thumb really works! (as a lower bound)

Just use 10x smaller than Kolmogorov scale for near wall region.

Now for experimental data (Ampofo and Karayiannis, 2003)

So we'll need u_q and u_τ for this case

$$u_q = \frac{\alpha}{H} \left(Pr \, Ra_H Nu_{local} \right)^{\frac{1}{4}}$$

$$\max Nu = 138$$

Note

$$0.15 Ra_H^{\frac{1}{3}} = 0.15 * (1.58e9)^{\frac{1}{3}} = 0.15 * 1164 = 175$$

Close enough for this correlation! Though I've been using $Ra_H^{\frac{1}{3}} \approx Nu$ so that may give me overestimated Nu. (Batchelor scales are too small, overconservative)

Maximum wall shear stress:

$$\tau_{wall} = 1.72 * 10^{-3} \frac{N}{m^2}$$

Air density estimate (for upperbound u_τ), is small ρ

https://www.engineeringtoolbox.com/air-density-specific-weight-d_600.html

$$\rho \approx = 1.23 \frac{kg}{m^3}$$

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}} = \sqrt{\frac{1.72 * 10^{-3} \frac{N}{m^2}}{1.23 \frac{kg}{m^3}}} = 0.0373 \frac{m}{s}$$

Kinematic viscosity of air

https://www.engineeringtoolbox.com/air-absolute-kinematic-viscosity-d_601.html

$$Re_\tau \approx 0.0373 * \frac{0.75}{20 * 10^{-6} \frac{m^2}{s}} = 1402.306462$$

Looks like this method:

$$Re_\tau = \sqrt{Gr}\sqrt{\frac{1}{2}0.37 * \left\{\log_{10}\left(\sqrt{Gr}\right)\right\}^{-2.584}}$$

Is not too bad!

Now let's determine our scales

Temperature scales:

$$y^* \equiv y\frac{(Pr\ Ra_H Nu_{local})^{\frac{1}{4}}}{H}$$

$$y^* \equiv y\frac{(0.71 * 1.58e9 * 138)^{\frac{1}{4}}}{0.75}$$

$$y^* \equiv y * 836$$

For y^* spacing of 0.05,

$$y = \frac{0.05}{836} = 6e - 5 = 0.06mm$$

This is about 10x smaller than Batchelor scale!

How about the velocity scales?

$$\Delta y^* = \frac{y}{H}\frac{(Pr\ Ra_H Nu_{local})^{\frac{3}{4}}}{(Re_\tau Pr)^2}$$

$$\Delta y^* = \frac{y}{0.75}\frac{(0.71 * 1.58e9 * 138)^{\frac{3}{4}}}{(1400 * 0.71)^2}$$

$$\Delta y^* = \frac{y}{0.75}249$$

$$y = \frac{0.75}{249} * \Delta y^*$$

For Δy^* spacing of 0.05

$$y = \frac{0.75}{249} * \Delta y^* = 0.00015m = 0.15mm$$

This Is about same order of magnitude as Kolmogorov and batchelor scale

$$\eta_K = 0.00054259m \approx 0.5mm$$

$$\eta_B = \frac{\eta_K}{\sqrt{Pr}} = 0.000643935m \approx 0.6mm$$

Conclusion

Smallest mesh size is about 10x smaller than Kolmogorov/Batchelor scale for natural convection and forced convection at low level turbulence (just after transition and not 100x bigger).

This should give you good estimate for mesh size in natural or mixed convection

Mixed convection can check Richardson number and see if you want to use the natural convection Kolmogorov scale or forced convection Kolmogorov scale

Part II

Quasi Direct Numerical Simulation and other Models – Practice Case in OpenFOAM

Kasagi, N., & Nishimura, M. (1997). Direct numerical simulation of combined forced and natural turbulent convection in a vertical plane channel. International Journal of Heat and Fluid Flow, 18(1), 88–99.

Komen, E., Shams, A., Camilo, L., & Koren, B. (2014). Quasi-DNS capabilities of Open-FOAM for different mesh types. Computers & Fluids, 96, 87–104. https://doi.org/https://doi.org/10.10

How is real DNS done? In summary, take fourier transform of navier stokes equations, and approximate it with Chebyshev Polynomials (Kasagi & Nishimura, 1997).

Unfortunately OpenFOAM doesn't do too well with this since it uses the finite volume method (FVM), but we can simulate turbulence anyhow by using a ultra fine mesh version of FVM: Quasi-DNS (Komen et al., 2014)

Let's try this for a practice case in OpenFOAM...

Suppose you have a mixed convection flow in a planar channel...

- T_{avg}=70C, T_{hot}=100C, T_{cold}=40C

- Re=3200 downward flow

- Dowtherm A oil

- Channel spacing = 1.5 cm

- Spanwise = 4.5 cm (3x channel spacing)

- Streamwise = 12 cm (8x channel spacing)

- Periodic BC

7 Smallest Scale Estimates

Mission: run simulation from initial conditions for 65 flow through time (FTT), get statistics for velocity and temperature (raw data, mean and rms) time averaged over 35 FTT. Also get time averaged friction coefficient and Nusselt Number over 35 FTT.

What's our Kolmogorov scale?

$$\frac{\eta}{l} \sim \frac{1}{3200^{0.75}} = 0.002350377$$

What's our natural convection based Batchelor and Kolmogorov scale?

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{(Nu-1)\,Ra}\right)^{\frac{1}{4}} ; \; \frac{\eta_B}{H} = \left(\frac{1}{(Nu-1)\,Ra}\right)^{\frac{1}{4}}$$

We use $Nu \sim 0.15 Ra^{\frac{1}{3}}$ (Bejan, 2013) for rayleigh benard cell. . .

We need our Grashof Number to find our Ra

$$Gr = \frac{g\beta\Delta T L^3}{\nu^2}$$

For dowtherm A at ~ 70C,

$$\nu = 1.44356E - 06\frac{m^2}{s}$$

$$\beta = 0.000802633$$

$$L = \frac{0.015}{2}m\,(channel\,half\,height)$$

$$g = 9.81\frac{m}{s^2}$$

34

$$\Delta T = 100 - 40 = 60K$$

$$Gr = \frac{g\beta\Delta T L^3}{\nu^2} = \frac{9.81 * 0.000802633 * 60K * \left(\frac{0.015}{2}\right)^3}{(1.44356E - 06)^2} = 95567.17253$$

Note:

$$Ri = \frac{9.55e4}{3200^2} \approx 0.0092$$

At transition Re $\approx 2300 - 2350$

$$Ri = \frac{9.55e4}{2350^2} = 0.0172$$

Now for Ra,

$$Pr = 19$$

$$Ra = 95567.17253 * 19 = 1815776.278$$

$$Nu = 0.15 * 1815776.278^{\frac{1}{3}} = 18.3$$

So we can make Batchelor Scale estimates

$$\frac{\eta_B}{H} = \left(\frac{1}{(Nu - 1) Ra}\right)^{\frac{1}{4}} = \left(\frac{1}{(18.3 - 1) \, 1815776}\right)^{\frac{1}{4}} = 0.013357476$$

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{(Nu - 1) Ra}\right)^{\frac{1}{4}} = \frac{\eta_B}{H} * \sqrt{Pr} = 0.013357476 * \sqrt{19} = 0.0582$$

Looks like the Kolmogorov scales based on Re are the smallest, we use that to size our mesh.

$$\frac{\eta}{l} \sim \frac{1}{3200^{0.75}} = 0.002350377$$

How many cells do we need if we were to size everything to Kolmogorov scale?

$$\frac{l}{\eta} = \frac{1}{0.002350377} \approx 426 cells$$

8 Mesh Grading

8.1 Wall normal direction

Smallest cell size should be 1/10 kolmogorov scale near the wall... (ultra conservative)

Largest cell size should be 3x Kolmogorov scale max.

We take smallest cell size to be 0.5 kolmogrov scale, max cell size is 2 kolmogorov scales. Quite decent... (Komen et al., 2014)

How did we determine?

We take, based on experimentation...

$$kolmogorovscale \sim \Delta x^+ = 1.83$$

For OpenFOAM, (Komen et al., 2014), wall parallel directions: $\Delta x^+ = 9$, $\Delta z^+ = 4.5$ $\Delta y^+ = 0.8 - 4.5$,

Meaning to say, in y direction, $\frac{1}{2}$ Kolmogorov scale is sufficient to resolve near wall, 2x Kolmogorov scale far from wall.

Split domain in half. Largest cell is 4x of smallest cell.

8.2 Streamwise direction

Cell size $\sim$ 4x Kolmogorov scale (Komen et al., 2014)

$$\frac{l}{\eta} = \frac{1 * 8}{0.002350377 * 4} \approx 851 cells$$

8.3 Spanwise direction

Cell size $\sim$ 2x Kolmogorov scale (Komen et al., 2014)

$$\frac{l}{\eta} = \frac{1 * 3}{0.002350377 * 2} \approx 639 cells$$

Total no. of cells:

$$2.32E + 08 \approx 232 million$$

That's a LOT, note: LES simulations can be on order of 0.3 million cells.

9 TimeStep Estimate

What's the velocity?

$$Re = \frac{u_{mean}H_{channel}}{\nu}$$

$$u_{mean} = \frac{\nu Re}{H_{channel}} = \frac{1.44356E - 06\frac{m^2}{s} * 3200}{0.015m} = 0.307959467\frac{m}{s}$$

What's the smallest cell size streamwise?

$$4 * kolmogrov\ scale = 4 * 0.002350377$$

Smallest cell size

$$\Delta x = 4 * 0.002350377 * 0.015m = 0.000141023m = 1.41 * 10^{-4}\ m$$

Timestep (Co=1)

$$\Delta t = 1.41 * 10^{-4} * \frac{1}{0.307959467} = 4.58E - 04s$$

For Co ≈ 0.5

$$\Delta t = 2.29 * 10^{-4}s$$

Or $\Delta t = 2 * 10^{-4}s$

This should be sufficient.

9.1 Data Collection

Periodic, once per FTT over 35 FTT.

What is the FTT duration?

$$FTT = \frac{0.12\ m}{0.307\frac{m}{s}} = 0.39s$$

So one FTT is about 0.4s

65 FTT is about 26s simulation time

35 FTT is about 14s simulation time

Collect data every 0.4s after 26s until t=40s

Since we have 426 cells in y direction, we can collect about 400 data points, for u, v, w and T over 16s.

Next, we also want to collect Cf and Nu data,

However, openFoam utilities measure only τ_{wall} and h, so I'd like to measure τ_{wall} and h for both hot and cold sides every 0.4s after 24s.

Look's like we are ready to setup!

9.2   Case Setup

Due to the block structure we want,

We may want to take blockMesh from cavityclipped or some other case! Eg. In my OpenFOAM tutorials, I had a modified an icoFoam case for BL flow

- Better yet, look for channel395 in LES tutorial, it is a horizontal channel (we want vertical channel)

The blockmesh file is quite good because it separates the flow domain into two blocks.

But we want to use buoyantPimpleFoam to run our simulation, so we can use a cavity case in buoyantPimpleFoam to give the rough case structure

```
theodore_ong@LAPTOP-FK3MEHTI:/opt/OpenFOAM/OpenFOAM-v1912/tutorials/heatTransfer/buoyantSimpleFoa
m/buoyantCavity$ l
0/  Allclean*  Allrun*  README  constant/  system/  validation/
```

in 0 directory, we can remove everything except T U, p and P_RGH, p_rgh is pressure minus hydrostatic pressure

```
theodore_ong@LAPTOP-FK3MEHTI:/opt/OpenFOAM/OpenFOAM-v1912/tutorials/heatTransfer/buoyantSimpleFoa
m/buoyantCavity/0$ l
T  U  alphat  epsilon  k  nut  omega  p  p_rgh
```

Detailed BCs can do it later. . .

In constant directory, we can use thermophysical properties from dowtherm A (from heat transfer)

```
theodore_ong@LAPTOP-FK3MEHTI:/opt/OpenFOAM/OpenFOAM-v1912/tutorials/heatTransfer/buoyantSimpleFoa
m/buoyantCavity/constant$ l
g  thermophysicalProperties  turbulenceProperties
```

Turbulence properties laminar (not truly laminar but it just means I don't use any model here)

9.2.1 BCs, fvSchemes, fvSolution

For temp BCs, we have fixed temperature BCs and cyclic on all sides relatively simple

For velocity BCs, we need to fix the mean flow and make it periodic

For Pressure BCs, we need to have a fixed pressure drop for fixed flow (one is calculated, one is fixed flux pressure drop)

fvSchemes

- As non dissipative as possible, central difference 2nd order schemes (Gauss linear) would be best, fingers crossed on stability!

fvSolution

- Not quite sure yet

- How do we iterate discretized equations

- What is P, pfinal

    ◦ https://cfd.direct/openfoam/user-guide/v6-fvsolution/
    ◦ Remember nCorrectors? Eg. Solving the pressure corrector in piso algorithm 4 times

    ◦ First 3 times use the settings for p (usually error bigger here, so doesn't converge as fast)
    ◦ Final time use settings for pfinal (smaller error here, so can be more stringent with tolerance)

GAMG or PCG? And what are preconditioners?

- A good report here:

- http://www.tfd.chalmers.se/~ hani/kurser/OS_CFD_2008/TimBehrens/tibeh-report-fin.pdf

- We are solving systems of equations in matrices

- Eg $Ax = b$

    ◦ Preconditioner gives

* $M^{-1}Ax = M^{-1}b$
* This supposedly helps solve the matrix with less iterations or more stably

○ Smoothers

* Remember how preconditioners reduce no. of iterations?
* However as mesh grows # iterations also grows
* We can reduce this dependency with smoothers

○ In general matrices are asymmetric

* DILU is good for this (diagonal based incomplete LU decomposition)
    † Velocity equations are complex
* DIC class is good for symmetric matrices
    † Pressure equations are more symmetrical

Solvers

- Now after preconditioning and smoothing we can think about solvers

- Detailed background here:

  ○ https://www.youtube.com/watch?v=BApG0hasPf4& list=PLbMVogVj5nJS4AsBvnp9bCw0
  ○ https://www.foamacademy.com/wp-content/uploads/2018/03/OF2018matrix.pdf
  ○ Krylov Subspace Methods

    * https://www.youtube.com/watch?v=B_eSPrYuIuU

- GAMG

  ○ Multigrid solver coarse mesh solve then map onto fine mesh (usually faster)

- Conjugate solvers (or some variation)

  ○ Exploits the conjugate vectors (not explaining here)

    * What are conjugate vectors?
        † Suppose we have a matrix A, and two vectors $\overrightarrow{v}$ and $\overrightarrow{w}$
        † These are conjugate vectors of A if: $\overrightarrow{v}^T A \overrightarrow{w} = 0$

- Gauss seidel (quite basic)

Examples of fvSolution in OpenFOAM tutorials

For dnsFoam boxTurb16, the following solvers are used:

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver              PCG;
        preconditioner  DIC;
        tolerance        1e-06;
        relTol           0;
    }

    U
    {
        solver              smoothSolver;
        smoother         symGaussSeidel;
        tolerance        1e-05;
        relTol           0;
    }
}

PISO
{
    nCorrectors        2;
    nNonOrthogonalCorrectors 0;
}

// ************************************************************************* //
```

In fireFoam, (quite challenging and non disspative)

```
p_rgh
{
    solver              GAMG;
    tolerance           1e-7;
    relTol              0.01;
    smoother            GaussSeidel;
    cacheAgglomeration  true;
    nCellsInCoarsestLevel   10;
    agglomerator    faceAreaPair;
    mergeLevels     1;
};

p_rghFinal
{
    $p_rgh;
    tolerance           1e-7;
    relTol              0;
};

"(U|Yi|h|k)"
{
    solver          smoothSolver;
    smoother        GaussSeidel;
    tolerance       1e-07;
    relTol          0.1;
}
 "(U|Yi|h|k)Final"
{
    $U;
    tolerance       1e-7;
    relTol          0;
};

Ii
{
    solver                  GAMG;
    tolerance               1e-4;
    relTol                  0;
    smoother                symGaussSeidel;
    cacheAgglomeration      true;
    nCellsInCoarsestLevel   10;
    agglomerator            faceAreaPair;
    mergeLevels             1;
    maxIter                 1;
}
```

Pressure, velocity, radiation all use gauss seidel or variants of such.

More on linear equation solvers

https://www.openfoam.com/documentation/guides/latest/doc/guide-solvers.html# sec-solvers-intro

And how are fvSchemes setup?

I'll try to make it as non dissipative as possible, eg. Linear or cubic interpolation, and use a stable solver, PCG for pressure and smooth solver for velocity, temperature.

Periodic BCs

How are cyclic BCs done in OpenFOAM?

https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-coupled-cyclic.html

Consider the FlamePropagation with Obstacles case in PDRFoam under combustion in tutorials:

First we need to define which two patches are linked in blockMeshDict

And in each BC, eg. In the changeDictionaryDict

Define it as cyclic.

Another good place to look is LES channel395 tutorial

https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-coupled-cyclic.html

One question though, for pressure, we need fixed flux, though hydrostatic pressure will go through the roof because it's "infinite length"

How is this solved?

For one, I can fix it at 6m depth worth of hydrostatic pressure (or settle for atmospheric!)

I'll have to experiment with this on a coarse mesh first then go to fine mesh.

The other thing is this,

The total pressure field is all calculated, meaning to say that the hydrostatic pressure at the bottom is not added to the top, but rather the p_rgh (pressure without hydrostatic component) is added and cyclic.

https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-coupled-jump-fan.html

fanPressure from TJunction will be most helpful

Ohh!!!

No need to go through the trouble of fan and everything, this is easier:

https://www.openfoam.com/documentation/guides/latest/doc/guide-fvoptions-sources-mean-velocity-force.html

meanVelocityForce enables can fix the velocity

- We want mean velocity to be: 0.307959467 m/s

When I run the thing, im shown that cyclic conditions cannot use calculated.

Where can I go to obtain graphical data generation templates?

Firstly postChannel from channel395 is an interesting way to start data collection. . .

- But I'll probably need to store obscene amounts of data on the computer, I need graphs created at certain timesteps. In fact, every 0.4s I want a graphical chart of data.

- Plus I want heat transfer c

One start point is here:

https://cfd.direct/openfoam/user-guide/v6-post-processing-cli/# x31-2450006.2.1.9

singleGraph option is good, we can look in laminar planarPoiseuille

Also my hotSphere in oil case. . . I run functionObjects during runtime

I wish to purgewrite the data off, but preserve maybe 6 timesteps tops, thankfully data is listen in postprocessing on the fly.

Looks like that's it. . . I'll need to try it out.

functionObjects OpenFoam Examples

Surface field values (see hot sphere in oil case)

- Wall Shear stresses

- Average Heat transfer Coefficients

Sets (see buoyantCavity Case)

https://www.openfoam.com/documentation/guides/latest/doc/guide-fos-sampling-sets.html

https://develop.openfoam.com/Development/openfoam/-/blob/master/tutorials/heatTransfer/buoyan

After doing a RANS run coarse mesh (10 by 10 by 10), realise it is more efficient to run it to convergence and map this temperature and velocity field to dns.

So I run RANS for 100s simulated time, map fields and run quasiDNS 40s simulated time

Can we set up a run file for this?

From mapfields cavity, case, we get a rather interesting way. . .

10   First Run of QuasiDNS

With this cell level, of refinement, the computer froze trying to run blockMesh.

And like 5 mins later, the blockMesh still isn't complete. This doesn't bode well for actual DNS.

$$kolmogorovscale \sim \Delta x^+ = 1.83$$

For OpenFOAM, (Komen et al., 2014), wall parallel directions: $\Delta x^+ = 9$, $\Delta z^+ = 4.5$ $\Delta y^+ = 0.8 - 4.5$,

Meaning to say, in y direction, $\frac{1}{2}$ Kolmogorov scale is sufficient to resolve near wall, 2x Kolmogorov scale far from wall.

Split domain in half. Largest cell is 4x of smallest cell.

Streamwise direction

Cell size $\sim$ 4x Kolmogorov scale (Komen et al., 2014)

$$\frac{l}{\eta} = \frac{1*8}{0.002350377 * 4} \approx 851 cells$$

Spanwise direction

Cell size $\sim$ 2x Kolmogorov scale (Komen et al., 2014)

$$\frac{l}{\eta} = \frac{1*3}{0.002350377 * 2} \approx 639 cells$$

Total no. of cells:

$$2.32E + 08 \approx 232 million$$

This is 426 in y direction, 852 streamwise, 639 spanwise.

That's a LOT, note: LES simulations can be on order of 0.3 million cells.

11 2nd Run of QuasiDNS - get blockMesh to generate less cells

Need to reduce cells to about 1-2 million. Meaning 10x less cells in x and y direction. (z I cannot compromise too much)

Means streamwise, we get $\sim$ cell size of 40x Kolmogorov scale streamwise and 20x Kolmogorov scale in spanwise, and then we get 1x-4x Kolmogorov in wall normal direction.

Use kOmegaSST-IDDES since that has proven to be useful and it's convenient (kOmegaSST fields are set well)

Note that about 80s later, the average heat trf coefficient settles down.

Good rule of thumb: 1 million cells 1 GB ram required for mesh

Alternative 2 would be to lessen the domain spanwise and streamwise by 10x. This preserves mesh grading but may neglect large turbulence structure formation. (I'm suppressing large eddy formation)

Hard limit on computer is about 2-4 million cells.

11.1   Quantify y^+ so that i can coarsen the mesh

Anyhow, wall shear stress is $\tau_{wall} \approx 4.14 * 10^{-1}$ according to RANS calcs (Re = 3200)

Friction velocity is:

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}} = \sqrt{\frac{0.414}{1019}} = 0.020156405 \frac{m}{s}$$

Kolmogorov scale is about $7 * 10^{-6} m$

$$\frac{\eta}{l} \sim \frac{1}{3200^{0.75}} = 0.002350377$$

Based on 1.5cm length,

$$\eta = 3.53 * 10^{-5} \approx 35 microns$$

This corresponds to:

$$y^+ = \frac{y\,[m]\,u_\tau\,\left[\frac{m}{s}\right]}{\nu\,\left[\frac{m^2}{s}\right]} = \frac{3.53 * 10^{-5} * 0.02}{1.44 * 10^{-6}} = 4.88E - 01 \approx 0.5$$

1 kolmogorov scale is approx. 0.5 y^+ !

My smallest mesh size is about 0.25 y^+ (1/2 kolmogorov scale)

At the right side,

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}} = \sqrt{\frac{0.464}{1019}} = 0.0214 \frac{m}{s}$$

Doesn't actually change the y^+ too much!

$$y^+ = \frac{y\,[m]\,u_\tau\,\left[\frac{m}{s}\right]}{\nu\,\left[\frac{m^2}{s}\right]} = \frac{3.53 * 10^{-5} * 0.0214}{1.44 * 10^{-6}} = 5.23E - 01 \approx 0.523$$

1 kolmogorov scale is enough to be the smallest which means I can halve my cell requirements in y direction.

This is 213 in y direction, 852 streamwise, 639 spanwise.

Likewise, in streamwise direction, I'm having 4 kolmogorov scales

$$\Delta x^+ \approx 2.1$$

I can decrease cell count by 4x and still be within guidelines

$$\Delta x^+ \approx 8.4 \, (streamwise)$$

This is 213 in y direction, 213 streamwise, 639 spanwise.

Finally, spanwise, im also too strict, being 3 kolmogorov scales,

$$\Delta z^+ \approx 1.5$$

I can decrease it 3x

$$\Delta z^+ \approx 4.5 \, (spanwise)$$

This is 213 in y direction, 213 streamwise, 213 spanwise.

This makes for about 9.66 million cells, way more doable (but I'm really pushing it)

Running blockMesh for this, about 95% of free ram was used, computer went hanging for 15 mins

12 The 3rd run, reduce no. of cells due to RAM limit of 8GB

Should push it down as far as possible, try Re=2350 mixed convection.

$$213 * \frac{2350}{3200} = 157$$

$$157^3 = 3827300 cells \approx 3.9 million cells$$

That is 3.9 million cells. That's about the best we can manage. . .

For LES type:

$157 * 40 * 40$

Or $158/2 = 79$ cells

For RANS type

$157 * 4 * 4$

Velocity is now

$$0.307959467 * \frac{2350}{3200} \approx 0.22618 \frac{m}{s}$$

Wall shear stress is now (in RANS mode) estimated

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}} = \sqrt{\frac{0.27}{1019}} = 0.0162777 \frac{m}{s}$$

New Kolmogorov scale is:

$$\frac{\eta}{l} \sim \frac{1}{2350^{0.75}} = 0.002962$$

Note that this is still smaller than the Kolmogorov scale of natural convection:

$$\frac{\eta_k}{H} = \left(\frac{Pr^2}{(Nu-1)\,Ra}\right)^{\frac{1}{4}} = \frac{\eta_B}{H} * \sqrt{Pr} = 0.013357476 * \sqrt{19} = 0.0582$$

So we'll use the forced convection kolmogrov scale as reference.

$$\eta = 0.015m * 0.002962 = 4.444 * 10^{-5} \approx 44\mu m$$

In terms of y^+ units, Kolmogorov scale is:

$$y^+ = \frac{y\,[m]\,u_\tau\left[\frac{m}{s}\right]}{\nu\left[\frac{m^2}{s}\right]} = \frac{4.44 * 10^{-5} * 0.0162777}{1.44 * 10^{-6}} = 5.02368E - 01 \approx 0.503$$

Looks good! Kolmogorov scale $=1/2\ y^+$ ish

Can live with this in fact wall shear stress decreases as the simulation stablilises.

$$Re_\tau = \frac{0.0162777 * 0.015}{1.44 * 10^{-6}} \approx 169$$

(low level turbulence)

Note that I use a Prt of 0.85 throughout in all these. Though for RANS, probably used Prt =0.85 near wall, 1 in bulk fluid.

Okay if you copy and paste the files over, run mapFields for t=100s.

Source file shows coarse mesh

Target file must show fine mesh (but now it doesn't do that yet.)

So I need to run RANS in the other folder and then map the fields over!

For IDDES,

Estimated u_τ

Is at

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}} = \sqrt{\frac{0.29}{1019}} = 0.0169 \frac{m}{s}$$

$$y^+ = \frac{y\,[m]\,u_\tau\,\left[\frac{m}{s}\right]}{\nu\,\left[\frac{m^2}{s}\right]} = \frac{4.44 * 10^{-5} * 0.0169 \frac{m}{s}}{1.44 * 10^{-6}} = 5.201E - 01 \approx 0.52$$

$$\frac{\eta}{l} \sim \frac{1}{(2350^{0.75})}$$

$$\frac{l}{\eta} \sim 337.52$$

Divide by two

And we need $\sim$ 169 cells in each direction.

$$169^3 \approx 4.8268 million cells$$

Still acceptable

In streamwise direction we have about 16 nodes

Distance is:

$$\Delta x^+_{IDDESSize} = \frac{\frac{0.12}{16}m * 0.0169 \frac{m}{s}}{1.44 * 10^{-6}} = 87.86$$

This is not quite LES size mesh... not LES, need it about $\Delta x^+ \approx 40$,

Good for mesh refinement study anyhow...

$$u_\tau = \sqrt{\frac{\tau_{wall}}{\rho}} = \sqrt{\frac{0.31}{1019}} = 0.0175 \frac{m}{s}$$

$$\Delta y^+ \text{ (158 cells mesh)} = \frac{\frac{0.015m}{158} * 0.0175 \frac{m}{s}}{1.44 * 10^{-6}} = 1.154$$

Smallest cell size:

$$\Delta y^+_{smallest} = \frac{1.154}{2} = 0.5768$$

In x direction (streamwise)

$$\Delta z^+_{159-cell} = \frac{\frac{0.12}{159}m * 0.0175\frac{m}{s}}{1.44 * 10^{-6}} = 9.171\,(streamwise)$$

Recommended size (streamwise) $\Delta z^+_{stream} = 9$

Spanwise,

$$\Delta x^+_{159-cell} = \frac{\frac{0.045}{159}m * 0.0175\frac{m}{s}}{1.44 * 10^{-6}} = 3.057\,(spanwise)$$

So basically, increase mesh refinement 3x in x and z direction to get proper LES length scale

Reduce by 10x ish (to 157 cells) to get DNS length scale

Oops I used 157 mesh not 159

In x direction (streamwise)

$$\Delta z^+_{159-cell} = \frac{\frac{0.12}{157}m * 0.0175\frac{m}{s}}{1.44 * 10^{-6}} = 9.288\,(streamwise)$$

Recommended size (streamwise) $\Delta z^+_{stream} = 9$

Spanwise,

$$\Delta x^+_{159-cell} = \frac{\frac{0.045}{159}m * 0.0175\frac{m}{s}}{1.44 * 10^{-6}} = 3.096\,(spanwise)$$

Anyhow testing DNS mesh size for blockMesh. . .

Really Pushes the computer to the limit. That's about all it can manage.

Huge Ram requirements. But it's a DNS mesh. . . what do you expect :)

Note file size for the polymesh also!!

(ls -lh command)

380 Mb just for one polymesh file.

Takes about 1-3 minutes to blockMesh and mapFields

Likely I'll need to make purgewrite 1 to delete all other files besides last written timestep.

Quite an insane memory requirement.

Running pimpleFoam on this mesh froze my computer...

So I would like to take advantage of mesh in x direction, decrease it to 125 cells, and increase z cells to 162

Total mesh number is:

$$\#cells = 125 * 158 * 162 = 3199500 cells \approx 3.2 million$$

Looks like my hardware can only take 3 million is cells.

Secondly we have instability growth here...

System is unstable here...

Let's solve one thing at a time.

13 The 4th run, and several debugs

13.1 First let's find the upper limit of blockMesh

...

- Max memory $\sim$ 3.2 GB used, and it's much faster, felt like 1/3 of time

- Looks good! Now let's see buoyantPimpleFoam, did it freeze the computer? Looks okay, just bearable.
  Yup so 3.2 million cells is the upper limit. NO MORE THAN THIS.

Takes about 1 minute to compute one time step.

13.2 Let's see the upper limit on time

So how long will it take if this is stable?

$$totaltime = 100s\,(sim) * \frac{1\,\min\,(real)}{2e-4s\,(sim)} = 500,000 min = 8334 hrs = 347 days\,(1 year)$$

With this speed, 40s 140 days (5 months) 1 entire semester!

What about write time?

0.4s

$$totaltime = 0.4s\,(sim) * \frac{1\min(real)}{2e-4s\,(sim)} = 2000min = 33.33hrs$$

I'd like it once per half hour..

Perhaps 0.004s is better, that should be the write time.

13.3 Coarse Mesh IDDES and RANS buoyantPimpleFoam can resolve, but not fine mesh DNS with buoyantPimpleFoam

Now that calculation is long but manageable, I realise that another issue crops up when I do fine mesh buoyantPimpleFoam doesn't really resolve fine mesh instabilities well, only coarse mesh LES works and RANS too, any smaller and the solver becomes unstable

Suppose the time issue is ok, then how about the instability issue?

13.4 Do LES fine mesh introduce some viscosity to control oscillation, may help control turbulence,

Given of course that LES is meant to tend to DNS if the mesh is ultra small. (Kolmogorov scale)

(characteristic lengthscale tends to 0 anyhow) so this should be ok

Second is to change controlDict to use constant timestep?

Suspicion: starting timestep too small.

Anyhow Co seems to drop below 1 (oscillating)

Okay instabilities growing. . .

13.5 Try introducing pimple Loop nOutercorrectors to stabilise the system

https://www.researchgate.net/post/Why_do_calculations_blow_up_when_using_finer_grids_k-omega_SST

2 ways to resolve numerical stability in fine mesh according to the post:

Increase nOuterCorrectors in Pimple loop to 3

- Linear upwind scheme used

- Increase no. of cells
- Upwind scheme more dispersive, so need to increase no. of cells for DNS

I'll try the pimple loop first.

This was kOmegaSSTIDDES with DNS sized mesh (based on Kolmogorov scale)

Okay this didn't work...

Probably the viscosity was too small to even make a difference due to small mesh size

13.6 I'm going to try eliminating solvers as a problem, probably Gauss Siedel solver is more stable than PCBiCG Solver

I'm going to try smooth solver Gauss seidel in fvSOlutions as per dnsFoam boxTurb16

Solver doesn't work!

13.7 Now if the timestep is too small, the solver becomes unstable for some reason, so I don't adjust timestep to be smaller, but fix it instead

Next I can try fixing the timestep

Use runtime instead

I'll write data every 20 min

https://cfd.direct/openfoam/user-guide/v6-controldict/

With parallel processing, I get 1.5 min per timestep..

- But the memory usage is a lot less now, 3.4 GB as compared to 4ish GB

Anyhow instability developed, pimpleFoam not able to run.

13.8 Trying out GAMG matrix solver to see if fvSolution is the problem

I'm going to try GAMG for U to see if this works to stabilise things...

Doesn't seem to,

Let me use a bigger timestep instead!

$$Co = \frac{v\Delta t}{l_{cell}} = \frac{0.22618 * \Delta t}{\frac{0.074074}{100}} = 3.053 * 10^2 \Delta t$$

53

For Co=0.5

$$\Delta t = \frac{0.5}{3.053 * 10^2} = 0.001677s \approx 0.0015s$$

13.8.1 i suspect having a varying timestep that was adjusting the timestep too small was an issue, and tried eliminating that

Maybe it's too small a timestep causing the error!

Changing timestep to 1.5e-3

Going back to symmetric Gauss seidel smoother smooth solver this was faster and yielded the same results

So regardless of solver it looks the same!

And smooth solver gauss seidel solver takes less memory than GAMG as well! (3.4 GB vs 4 GB when running buoyantPimpleFoam in serial)

It turns out unstable and the max iterations for k and omega are reached.

- I'll remove k and omega from calculation

13.9 k and omega reached max iterations in last calculation, i turned off the turbulence model to see if that could help with stabiltiy

Okay this still results in instability.

Courant number blows up to 18.

13.10 Perhaps my discretisation schemes in fvSchemes is the problem, though I don't really want to mess with it since it gives me less accuracy

However, in less resolved meshes, it seems to work I posit that dissipation adds to stability. Therefore, I'll try to use dissipation schemes (upwind) for conduction.

Let me try Gauss Upwind for div(phi,h) these are convective terms.

Ok that doesn't work. . .

13.11 I don't know what to do, but i want to look online to see if same problems are encountered and if there are solutions

Are there any people doing dns that have done it online?

https://bugs.openfoam.org/view.php?id=1894

This is one,

This is his code:

```
ddtSchemes
{
    default        CrankNicolson 0.9;//backward;//CrankNicolson 1.0;//
}

gradSchemes
{
    default        Gauss linear;
}

divSchemes
{
    default        none;
    div(phi,U)     Gauss linear;//midPoint;//
    div(phi,h)     Gauss QUICK phi;//
    div(phi,e)     Gauss linear;
    div(phi,K)     Gauss linear;
  div(phi,epsilon) Gauss linear;
    div(phi,R)     Gauss linear;
    div(phi,Ekp)   Gauss linear;
    div(R)         Gauss linear;
    div((muEff* dev2(T(grad(U))))) Gauss linear;// corrected;
}
```

https://www.openfoam.com/documentation/user-guide/fvSchemes.php

according to this, gauss quick phi is one way to do it...

next I want to eliminate a bad initial condition where the initial condition is so far from actual solution that it causes instability, I'd want to have an initial condition with some semblance of turbulence to allow DNS to catch up so to speak

Another note: when I reduce mesh size, for the first few seconds, wall shear stress drops dramatically.

- 0.261 initially for IDDES case 1: mesh size $\Delta z^+ (streamwise) = 89$

- Then for first timesteps, the average wall shear stress was 1.733e-1 (0.17)

- Only went back to 0.245 a few seconds later

- This lack of wall shear stress probably due to modelled stress depletion (suddenly ν_t disappears and now turbulence is expected to be resolved) may be a factor in causing such a thing.

Thus the initial velocity profiles for less resolved cases may cause certain instabilities esp near wall?

Not really sure. . .

13.12   Perhaps mapping fields from a coarse solution caused it to blow up?

But if I DNS from scratch will this issue re-alppear?

Anyway IDDES finer mesh solver blew up as well

13.13   maybe dpdt term in buoyantPimpleFoam causes problems

So trying the above, i thought something else must be the issue, I wondered if other people were facing the same issue as i

https://www.cfd-online.com/Forums/openfoam-solving/220924-highly-unstable-buoyantpimplefoam-simulation.html

One former here suggested dpdt off. . .

We can try that.

Courant number still grows. . .

Anyhow to run in parallel, use:

Mpirun -np 4 buoyantPimpleFoam -parallel &

13.14 Looks like many people online face the same issue!

https://www.cfd-online.com/Forums/openfoam-solving/220924-highly-unstable-buoyantpimplefoam-simulation.html

Pimplefoam unstable at small courant numbers, not the first time!

https://www.cfd-online.com/Forums/openfoam-solving/172144-pimplefoam-rhopimplefoam-unstable-small-courant-numbers-time-steps.html

"Folks - I have not reviewed your cases in detail, but let me offer the following observation to see if it helps.

Reducing the time step significantly will reduce your temporal differencing errors ... if this is first order temporal differencing then you are removing some of the numerical stabilising (damping) that is present in your high Co runs. That is probably why your low Co runs are much more lively.

You see the same effect when refining mesh spacing - what runs nice and stably with a coarse mesh can go haywire as you refine the mesh and remove the numerical diffusion that is stabilising the solution." – Tobermory

13.15 Thermal Diffusion timescales not resolved, thus instability? let's test

One other possibility is that my time scales are not small enough to resolve heat transfer, usually this case of instability comes with heat transfer because we have negative temperatures

I'm going to try the crank Nicolson time scheme now, cos that seems to be the most logical (two other folks with working simulations use this. . .)

Still blew up:

One more timescale to consider:

https://en.wikipedia.org/wiki/Von_Neumann_stability_analysis

$$\frac{\alpha \Delta t}{\Delta x^2} \leq 0.5$$

Larger alpha is more conservative

$$Pr = \frac{\nu}{\alpha}$$

$$\alpha = \frac{\nu}{Pr} = \frac{1.443e-6}{15} = 9.62e-8$$

$$\frac{\alpha \Delta t}{\Delta x^2} \leq 0.5$$

Kolmogorov lengthscale $\eta = \frac{0.015m}{157} =$

$$9.62e - 8 \left(\frac{m^2}{s} \right) * \frac{2e - 4s}{95\mu m^2} = 0.0021077$$

So tis criteria is met. Nothing to do with this.

Part III

Putting DNS on hold, let's try LES with a proper LES mesh first

Okay, so DNS isn't working, I'll try something simpler first: IDDES with LES type mesh

14 IDDES Case 2: Proper LES mesh, Run 1

$\Delta x^+ \approx 40$

I'm also trying crank Nicolson for IDDES case 2, with LES type mesh. Initial results seem stable.

But after 0.15s simulation time, the simulation blows up.

- Interesting thing, the courant number has growing oscillations about Co=0.5, indicative of some oscillation.

14.0.1 More Reads online show that buoyantPimpleFoam Instability is a common issue

Next interesting thing, when I search buoyantPimpleFoam velocity oscillations, i see a few people's master's thesis and they experience the same problem.

- In fact they could not finish their thesis when they tried various things, fvSchemes, relaxation factors, BCs etc.

- Common problem, im not alone

- Eg. https://fenix.tecnico.ulisoba.pt/downloadFile/1689244997257061/tese.pdf

- www5.in.tum.de/pub/Cheung2016_Thesis.pdf

- Some ignore buoyancy altogether

14.0.2 Batchelor Scales apply to Forced convection! not just natural convection

As I read some things (Steven Van Haren thesis)

I realise batchelor scales also apply to Forced convection!

- $\eta_B = \frac{\eta}{\sqrt{Pr}}$ (dowtherm type $Pr > 1$), or $\eta_B = \frac{\eta}{Pr^{\frac{3}{4}}}$ ($for Pr \leq 1$) (Haren, 2011)

  - G.K. Batchelor. Small-scale variation of convected quantities like temperature in turbulent fluid. Fluid Mech, 5:113–133, 1958.
  - P.A. Davidson. Turbulence. Oxford University Press, 2004

- https://repository.tudelft.nl/islandora/object/uuid:67186c81-cac5-41b6-a21f-8e3670cafead

Oh bother..

This means that I need way finer, way way finer mesh.

14.0.3 Do i need a finer mesh for Batchelor scales? and how much finer?

But some mixed convection DNS use the mesh stated above. . .

Nevertheless, the DNS done is usually for air, with Pr $\approx$ 0.71. Dowtherm has high Pr, may still need finer mesh to resolve Batchelor scales.

15 IDDES Run 2, suspect that system is unable to cope going from start state to steady state, change in temperature too steep

Is initial condition still the problem? Ie is initial condition so far from actual solution that the solver blows up?

One way to circumvent is using a slightly bigger mesh, and use the well conditioned flow field to go down to DNS.

This is 36 x 158 x 36

This ensures max $\Delta x^+ \approx 39.11$

Still blows up.

16 IDDES Run 3, suspect Batchelor scales are not properly resolved in IDDES, thus refine mesh and lower time to account for that

<u>Batchelor scale hypothesis</u>

I posit that when I reduce the size of the mesh, ν_{sgs} drops dramatically (because of smaller mesh size) so that resolved turbulence can make up for the loss of modelled viscosity.

However, on the heat transfer end, α_{sgs} also drops due to ν_{sgs} decreasing. But the Batchelor scales may be smaller than Kolmogorov scales. I calculated batchelor scales for natural convection, thinking it doesn't apply to forced convection.

But batchelor scales also apply to forced convection! It is just that when air is used, the Reynold's analogy can hold well as $Pr \rightarrow 1$.

$\eta_B = \frac{\eta}{\sqrt{Pr}}\left(dowthermtype Pr > 1\right), or \eta_B = \frac{\eta}{Pr^{\frac{3}{4}}}$ (Haren, 2011)

Therefore, when I decrease mesh size and ν_{sgs} , the resolved turbulence increase $\overline{u'v'}$ can make up for the loss in α_{sgs} and hopefully resolved turbulent heat flux $\overline{u'T'}$ can also make up for the loss in α_{sgs} .

However, when we have high Prandtl number fluids such as water and dowtherm A, forced convection batchelor scales become important. Most of the instability issues arise from water which has Pr of 5-7.

$$\eta_B = \frac{\eta}{\sqrt{7}} = 0.377\eta$$

Thus reducing mesh size to Kolmogorov scale in IDDES will resolve most fluid motion, but not resolve the Batchelor scales.

Van Haren's thesis uses $Pr = 1$. This allows for the reynold's analogy to be used.

However, for $Pr = 7$, Batchelor scales will come into play. Does it mean we refine in 3D? No. He writes that increasing fidelity in wall normal direction is sufficient.

These two papers were cited:

Y. Na and T.J. Hanratty. Limiting behavior of turbulent scalar transport close to a wall. Int. Journal of Heat and Mass transfer, 43:1749–1758, 2000.

R Bergant. Turbulent Heat Transfer near the Flat Wall for High Prandtl Numbers. PhD thesis, University of Ljubljana, Faculty of Mathematics and Physics, 2005.

So there are two ways of approaching this:

1) I do the IDDES simulation with air as a fluid, but I'll have to change grashof numbers etc

2) I refine the y normal direction according to batchelor scale, not Kolmogorov and work from there.

16.0.1 Smallest scales of Batchelor scale and its impact on mesh size

For this I'll need a mesh of:

$$\frac{\eta_B}{\eta} = \frac{1}{\sqrt{Pr}}$$

For us, highest Pr number is $\sim$ 28-30

$$\frac{\eta_B}{\eta} = \frac{1}{\sqrt{30}} = 0.182$$

In other words

$$\frac{\eta}{\eta_B} \approx 5.5$$

So I'll need to increase mesh refinement in y normal direction 5 times to capture Batchelor scale.

So mesh refinement is up to 395.

```
blocks
(
    hex (0 1 3 2 6 7 9 8) (36 395 36) simpleGrading (1 4 1)
    hex (2 3 5 4 8 9 11 10) (36 395 36) simpleGrading (1 0.25 1)
    // how simple grading works: in positive y direction
    // as y increases, cell gets bigger according to expansion ratio
    // such that largest cell at Ymax is 4x smallest cell at Ymin

);
```

This should capture properly all batchelor scales.

16.0.1.1 I probably need a better PC
However, doing this will ensure that I cannot even do DNS on this computer unless I reduce the size of the domain by 5 times also.

- This will be interesting to test if available, a smaller domain DNS for air, compare to Kim and Menon for heat flux, wall normal profile. Nevertheless, it WILL take long.

- 3.2 million cells x5 $\approx 16 million cells$

- I will need a 32 GB ram machine to calculate this comfortably

16.0.1.2 Lowering Timescales to resolve Batchelor Scales

Likewise I need to make sure the speed of the turbulent heat flux does not exceed what is required similar to how Co is being done...

The heuristic is that if the batchelor scale is 5.5x lower than the Kolmogorov scale, the batchelor timescale should be 5.5x lower than the Kolmogorov timescale.

Ie the courant number should be $= \frac{0.5}{5.5} = 0.09$

So I'll set that in the controlDict.

Now it's running.

One thing though, this IDDES simulation is about 1 million cells

Name	Status	ᵛ 100% CPU	94% Memory	4% Disk	0% Network
buoyantPimpleFoam		14.7%	446.4 MB	0 MB/s	0 Mbps
buoyantPimpleFoam		14.7%	466.0 MB	0 MB/s	0 Mbps
buoyantPimpleFoam		14.3%	460.0 MB	0 MB/s	0 Mbps
buoyantPimpleFoam		14.2%	459.0 MB	0 MB/s	0 Mbps

Needs this much RAM 1.6 GB.

So 1 million cells to about 1.6 GB ram is to be expected

16.0.1.3 Can i change how we decomposePar our domain for a Speed Tweak? (not quite)

One small tweak with parallel processing, I will split in x and z direction instead of y direction.

Original:

```
numberOfSubdomains  4;

method          simple;

coeffs
{
    n           (1 2 2);
}
```

For DNS test 3.2 million cells, 1 parallel timestep calculation with y and z split in 2 and 2 costs ~ 60s.

I can change the setting to this

```
numberOfSubdomains   4;

method               simple;

coeffs
{
    n                (2 1 2);
}
```

The area of cells talking to each other so to speak diminishes. . . (I hope)

Okay regardless how its split here, computation time is roughly the same.

Back to main topic of Batchelor scales

My Alternative is to convert my current case into an air case with same temperature bounds. Though likely that will change the Grashof and Rayleigh number quite a bit.

My case is blowing up. If it's truly a batchelor scale issue, I can try decreasing the turbulent Prandtl number to increase the turbulent conduction. That's one way to deal with it.

Turbulent Prandtl number decrease to 1e-5.

This significantly increases α_t which should simulate the extra large $\overline{v'T'}$ interactions as compared to $\overline{u'v'}$

Okay im still getting a blowup here. . . I wonder why.

But if $\nu_{sgs} \to 0$, doesn't matter how big α_t will be, there will still be big convection fluctuations generated. Very hard to model.

One thing I catch however, is that ubar is positive (flow is moving up!)

Ubar should be negative, and that is same with downcomer.

However, when case blows up, uncorrected ubar goes from 0.224 to about -1e7. Which means there is a natural convection downwards, that's the only thing that can cause this.

So definitely problem is with natural convection.

- If the problem is only with natural convection then id say take out the temperature differences.

- However, it seems that pimpleFoam has a problem with small mesh sizes in general. Not just buoyantPimpleFoam. So doubtfully it's just natural convection.

17 IDDES Run 4, Trying things out with air as the fluid to make Kolmogorov scales the smallest Relevant Scales

Imitation with air takes out the Batchelor Scale problem, plus there are profiles to compare to

Let me simulate with air and see if anything changes.

I'll use the same Reynold's number.

Looks like Openfoam is quite ill-equipped to simulate high Pr fluids in turbulent motion! (I don't really know, pimplefoam also has some problems in general)

Let's try imitating one case for air, then call it a day don't think that's feasible but can try first at least

The Pr is 0.705, $\mu = 1.805 * 10^{-5}$

At 70C, air kinematic viscosity is $\approx 2e - 5$

https://www.engineersedge.com/physics/viscosity_of_air_dynamic_and_kinematic_14483.htm# :∼ :text=The% 20viscosity% 20of% 20air% 20depends,2% 20% 2Fs% 20or% 2014.8% 20cSt.

To keep Re=2350

$$2350 = \frac{u\,(0.015)}{2e - 5}$$

$$u_{mean} = 2350 * \frac{2e - 5}{0.015} = 3.133\frac{m}{s}$$

My alternative is to make the width bigger, though that will increase the grashof number

http://www.thermopedia.com/content/1191/

(im testing solver stability first rather than compare cases)

$$\beta_{IdealGas} = \frac{1}{T\,(K)}$$

$$Gr = \frac{g\beta\Delta T L^3}{\nu^2} = \frac{9.81\left(\frac{\Delta T}{T}\right)\left(\frac{0.015^3}{2}\right)}{2e - 5}$$

$$Gr = \frac{g\beta\Delta T L^3}{\nu^2} = \frac{9.81\left(\frac{40}{70}\right)\left(\frac{0.015}{2}\right)^3}{(2e - 5)^2} = 5912$$

Let me increase L to match Gr

The number to match is 95567

$$Gr = \frac{g\beta\Delta T L^3}{\nu^2} = \frac{9.81\left(\frac{40}{70}\right)\left(\frac{H}{2}\right)^3}{(2e-5)^2} = 95567$$

Only thing I want to match here is Kolmogorov scale resolution, other quantities don't have to match

Using solver

$$\frac{H}{2} = 0.01896$$

$$H = 0.03793m$$

Therefore, u_{mean} needs to be:

$$u_{mean} = 2350 * \frac{2e-5}{0.03793} = 1.23924\frac{m}{s}$$

This way I match Re and Gr

An easy hack is to change the length scale in blockMeshDictIDDES

$$scale = 0.01 * \frac{0.03793}{0.015} = 0.02528$$

17.0.1 I noticed that i forgot to turn on my LES model, but even turning it back on don't matter

What about my LES model??

One thing I noted, my LES model was turned off! For the previous runs

Let me test the LES mesh with LES model on before I do something else eg. air.

Setting blockMesh back to $36 * 158 * 36$

And courant number back to 0.5.

Anyhow looks like model blowing up again.

17.0.2 Is the issue again that a bad initial field was given?

Regardless if I start from T=200 or T=0, blows up!

17.0.3   The air test has failed, so insufficiently resolving scales is not the issue

But now let me try the alpha stability test again. That failed.

Note that bounding omega has exponentially increased: max 7.19e28

17.0.4   I suspect again it may be a case where the initial field is too far from steady state, thus the solver blows up

17.0.4.1   maybe boxTurb can help us generate good initial fields   boxTurb for initial field???

Anyhow, if turbulence doesn't seem to be doing anything, maybe I'll try boxTurb to start a "good" initial field??

17.1   I suspect that pimpleFoam in general isn't too well suited for resolving turbulence, LES or DNS, but it runs ok with RANS type!

So how unstable is pimpleFoam and how can we get it to stabilise?

I don't really know anymore... but we can look at some cases with fine meshes where pimpleFoam doesn't blow up

17.1.1   There may be some hope with pimpleFoam, ie take heat transfer out of the equation

Three cases with LES fine mesh:

https://www.openfoam.com/documentation/guides/latest/doc/verification-validation-turbulent-periodic-hill.html

https://www.openfoam.com/documentation/guides/latest/doc/verification-validation-turbulent-plane-channel-flow.html

https://www.openfoam.com/documentation/guides/latest/doc/verification-validation-turbulent-surface-mounted-cube.html

Apparently channel flow is possible with mesh...

$$\Delta x^+ = 49.6, \ \Delta z^+ \approx 15.1, \ \Delta y^+ \approx 1.1$$

Now this is LES for sure. I'd of course like to see if Δx^+ can drop below 40... this is critical.

Note that this channel flow has about 1.8-2.4 million cells, which is pretty high.

And they do with 8 processors... wow.

To run:

Mpirun -np 8 pimpleFoam -parallel &

Interesting, more than 2 pimple iterations are used.

- Outercorrectors used 3 times

- nCorrectors used 1 time

Courant number is about 1.3-1.4

If that floats the boat, that floats the boat, it's not 5 or 10...

Note this also, the pimple solver uses only the final settings so to speak... (in the fvSolutions),

I'm not using the full capability!

17.2 pimpleFoam Plane Channel Flow test Results, LES mesh size $\Delta x^+ = 40$ with kEqn in pimpleFoam is ok

Plane channel flow in the OpenFoam tutorials was modified to give a smaller mesh. Such that Δx^+ can drop below 40. (I made 625 grid points in blockMesh rather than 500)

However, the BCs are different here and so is the solver.

The solver here is pimpleFoam

- no cyclic conditions are used for inlet/outlet, instead a turbulentDFSEM intlet is used to artificially generate eddies

- a fixed outlet pressure is also used

- https://www.openfoam.com/documentation/guides/latest/doc/verification-validation-turbulent-plane-channel-flow.html

The simulation run is <u>very</u> stable, runs for 30-65s without any flaws.

- Started 0-31s with timestep 0.004, CFL$\sim$ 1.2 (constant)

- From 31s-63.5s, set maxCo=0.5

  - Max CFL will fluctuate periodically, which should be expected for resolved eddies

  - Timestep will adjust to make max Co set at 0.5

- Oscillations are manageable, solver does not blow up after this set time (don't know if it will in long run, ie 100s of seconds, but I doubt we need to measure time for that long)

  Solver has completed running after about 2 weeks at 8 processes.

  Oscillations manageable, not solver blow up, stable performance with adjustable timestep. So it goes to show with the right settings, resolution of turbulence is still possible.

18 IDDES Run 5: IsoThermal BuoyantPimpleFoam

18.0.1 i suspect heat transfer may cause the instability issue, thus i'm trying buoyantPimpleFoam but making the entire case isothermal

Can similar stability be done for buoyantPimpleFoam?

I suppose so, I want to eliminate natural convection as cause of instability so I'm just switching to isothermal BCs for buoyantPimpleFoam, to test if the solver itself has capability to handle the small mesh.

However, I ran buoyantPimpleFoam copying IDDESCase2 and changing temp BCs to isothermal, and the solver crashed due to exponential CFL number.

I then tried using a RANS type mesh and ran the same thing, again it crashed.

I wonder what mistake I made there... did I change fvSchemes or some other thing to cause instability?

I'm re-running the RANSCase1 which worked before as a test.

In fact IDDESCase1 also worked which I can then use for a template for IDDESIsothermal (which is IDDES model used with isothermal BCs).

The RANS case is quite stable. So I'm quite satisfied.

I hope the isothermal BC is not causing system crash.

To organise which cases are stable and which are unstable,

I should rename RANSCase1 and IDDESCase1 as stable cases, unproven cases are called test cases.

19 IDDES Run 6: Removing cyclic BCs as a cause of instability

One other reason for instability is inappropriate BCs.

Instability has many reasons, a smaller mesh size in openfoam tends to be more numeri-

cally unstable as numerical errors may make up a larger percentage of the simulated data. (conjecture)

Any simulation with stiff type BCs can cause this to happen. There must be a way to dampen these data out

- One is via larger mesh size

- Second is via stable BCs

Taking large mesh size away as a way to dampen the errors may leave the error dampening mechanisms insufficient.

Probably need more error dampening mechanisms:

19.0.1 What constitutes Stable BCs?

- eg pressure must be fixed at one end, rather than left as a cyclic patch

    ○ better yet have a fixed pressure drop and zerogradient velocity

- velocity should be fixed at one end and zerogradient at the other

- Refer to this table:

    ○ https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-common-combinations.html

    ○ https://cfd.direct/openfoam/user-guide/v7-boundaries/

    ○ A larger library of stable tutorial cases!

- https://cfd.fossee.in/case-study-project/case-study-run/20

So if I take away the cyclic BCs, I can instead have a fixed temperature inlet BC, fixed pressure inlet BC, and sort of tune it to get a desired velocity and Re.

This way instead of simulating an infinitely long inlet, I simulate the entrance region, which could be easier to validate.

I posit that if the entrance heat trf coeffs and friction can be matched (DNS-LES-RANS), it is at least a second best guestimate for the rest of the flow (infinite periodic region).

Alternative is this, if stable and the entrance U, T and p fields are developed in LES, then I map these to LES in periodic BC with help of change dictionary. (I can have a entrance region case which is stable and copy into the file)

19.0.1.1 RANS Case 1 and IDDES Case 1 seems to work well with cyclic BCs! What would I wish to work? IDDESCase2

How? We'll want to test IDDES Case 2 type mesh with New BCs first (343k entry temperature, no periodic BCs, zerogradient Exit), what should velocity be? Turbulent DFSEM inlet and zerogradient outlet with correct average velocity...

Apparently fixed mass/vol flowrate and fixed static pressure is very stable in the outlet.

https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-common-combinations.html

19.0.2 Possible BCs to use

The velocity BC can be set as fixed flowrate with ability to extrapolate velocity profile downstream.

https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-inlet-flow-rate-inlet.html

DFSEM inlet can also be used for turbulence generation. Necessary? Not sure... but definitely more physical.

To see if its stable.

Usage

The condition is specified in the field file for a volumetric flow rate using:

```
<patchName>
{
    type              flowRateInletVelocity;
    volumetricFlowRate <Function1>;
    value             <field value>;
}
```

Field file for a mass flow rate:

```
<patchName>
{
    type              flowRateInletVelocity;
    massFlowRate      <Function1>;
    value             <field value>;
}
```

Hopefully this will be stable...

Velocity of 0.22618 m/s gives Re$\sim$ 2350. So I'll put this into my BCs.

19.0.3 Steps taken to change BCs

Here are the steps taken:

19.0.3.1 BlockMesh Changed inletoutlet into patch in blockMeshDictIDDES

19.0.3.1.1 Rename the patches because I got my inlet/outlet and side periodic patches mixed up
Seems I've been simulating the wrong thing all along LOL

Nevertheless it shows that it can work for smaller cases

I'd like to see it work for larger cases though

Next time place the x and y and z coordinates of the planes in blockmesh for easier refernce

But anyhow for cyclic conditions, it doesn't really matter how I name the patches, only wall is impt as well as temperature of plate physically still simulating correct thing but wrong patch names that's all

19.0.3.2 changeDictionary
Change BCs using changeDictionary

19.0.3.2.1 Velocity inlet and outlet
For velocity, inletoutlet and turbulentDFSEM is used

19.0.3.2.2 Pressire BCs
Pressure

1) Cyclic at cyclic BCs

2) Calculated otherwise

P_rgh

1) Inlet zerogradient (or for this case, since its compressible we use fixedfluxpressure)

   a) https://cfd.direct/openfoam/user-guide/v6-boundaries/

2) Outlet will be fixed value

19.0.3.2.3 Temperature BCs
For temperature, inlet is 343K, outlet is also fixed to 343K

19.0.3.2.4 Turbulence Quantities BCs
For k & omega

1) https://www.researchgate.net/post/How_do_we_give_boundary_conditions_in_k_omega_SST_m

2) Can expect low level or turbulence to be modelled,

   a) For 5% intensity, I can expect 20% to be modelled for typical LES

   b) Hence turbulent intensity can set to 1%

   c) In the turbulentDFSEM case, a miniscule value is used for k (1e-5)

(i) https://develop.openfoam.com/Development/openfoam/-/blob/master/tutorials/incom

(ii) Outlet is inletOutlet (zerogradient for outflow, fixedvalue for reverse flow) with $ internalField as value

d) For omega I can set it to count on mixing length $\sim$ l= 0.038* d_h

   (i) Or 0.038 channel width

   (ii) https://www.cfd-online.com/Forums/openfoam-solving/185059-k-omega-inlet-k-omega-sst.html

   (iii) I used 7% channel width for simplicity

   (iv) Outlet is inletOutlet, fixed reverse flow revalue and initial value with $ internalField as value

e) For nut

   (i) I set it to zerogradient/nutUWallFunction at walls

   (ii) Cyclic at cyclic boundaries

   (iii) Calculated with initial value of 1e-8 (similar to the channel395 DFSEM values)

f) For epsilon (if I ever use it)

   (i) https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-inlet-turbulent-epsilon-turbulent-mixing-length-dissipation-rate.html

   (ii) For inlet outlet turbulent mixing length dissipation rate (0.07 times channel width)

g) Alphat

   (i) Wallfunctions with Prt of 0.85

   (ii) Initial condition of 0, (doesn't seem physical however)

     (1) But usually code will calculate it using a Prt

   (iii) Inlet and outlet done using calculated

   (iv) https://develop.openfoam.com/Development/openfoam/-/blob/master/tutorials/heatT

19.0.4 Getting TurbulentDFSEMInlet to Work Right

debug log

round 1 says that there is some error, likely syntax or something, or missing tensor value,

I suspect turbulent DFSEM doesn't have an initial field to start, which is why is fails.

I tested the code by commenting out portions of the code one at a time. Sure enough it's in the turbulentDFSEM part

I notice the header file of for DFSEM case channel395DFSEM says location 1 . . . why?

Oh I found the bug, we need patch values for R, U and L, the Reynolds stress tensor

https://www.openfoam.com/releases/openfoam-v1606+/boundary-conditions.php

these should be in the constant directory.

Okay. So based on these tensors we simulate turbulence.

Useful article:

http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2012/NinaGallJorgensen/ReportReviewed.pdf

In general timevaryingMappedFixedValue should be able to work as well, almost like a semi periodic BC.

In pisoFoam pitzDailyMapped there is such a BC.

It is also known as mapped in the ESI release of OpenFoam.

Anyhow to obtain boundary data, probably need to use some postprocessing to obtain the relevant length scales, U fields, reynold's stress tensor and points file.

How to generate these fields?

Reynold's stress field can be obtained using

buoyantPimpleFoam -postProcess -func R

https://cfd.direct/openfoam/user-guide/v6-post-processing-cli/

the other way is using boundaryData. . .

https://www.openfoam.com/releases/openfoam-v1606+/boundary-conditions.php

so basically its sampling using functionObjects and releasing it as boundaryData format. . .

https://www.openfoam.com/documentation/user-guide/userse21.php

Okay. . . can do after lunch.

I can specify some random values first for turbulentDFSEM.

Just want to test stability.

Using this format. . .

```
inlet
{
type turbulentDFSEMInlet;
delta 2;
R uniform (xx xy xz yy yz zz);
U uniform (x y z);
L uniform x;
value (0 0 0);
}
```

Oh boy it works! (the BC does anyway)

Okay testing on big mesh, no exceptionally large Courant numbers. . .

($\sim$ 200,000 cells simulation)

Doesn't seem to blow up yet though. (t=0.07s)

Looks like non periodic does make the simulation stable. So correct BC setting is impt. Esp for LES where stabilising effect of large timestep is removed. Likewise looks like DNS requires the same things to work.

Running parallel on this mesh. . .

So looks good! Solver is stable at these times. (t=2.31 s), (t=5.21s)

How then do I get data

I even left the fvOptions for mean flow on by accident and that still doesn't affect stability much!

I can use the turbulenceFields function to calculate R and L.

So I can get a snapshot of U, R and L at some specified points. Great! I can then map these over to boundary field data (just remember that the patch names may be different cos I named them wrongly)

Oh man, I also forgot, i made the flow in the wrong direction! Got to run the RANS and IDDES case again. . .

Let's mend the git repository and pull changes in.

Btw, for large files more than 100MB or more, one can actually decomposePar to make them become less than 100 MB then it makes it easy to upload on github.

- I first uploaded the pimpleFoam files directly to github (turbulentDFSEMInlet files after running stably for 85s and cleaning it out)

- I pushed the commits from the IDDES branch and merged it into the master branch, thus resolving the problem. . .

19.0.5   Generating R, L and U boundaryData for turbulentDFSEMInlet

The function Object branch

Now I want to include function objects, so i'm making a branch for function objects.

https://www.openfoam.com/documentation/guides/latest/api/classFoam_1_1functionObjects_1_1turb

Firstly I'll want to generate turbulence fields

And I'll also want to write the coordinates into a points file so that the boundaryData format.

For easy reference, this may help.

https://www.openfoam.com/documentation/user-guide/userse21.php

I already have a sets function:

```
temp_and_velocity
{
    type sets;
    libs            ("libsampling.so");

    interpolationScheme cellPointFace;

    setFormat raw;
    executeControl  writeTime;
    writeControl    writeTime;
    sets
    (

    faceLineIntersection // samples data across a line at cell faces
    {
        type            face;
        axis            y; // i use y for easy plotting, but u can use xyz to indicate you want a
ll 3 coordinates
        start           (0 -0.0075 0.06);
        end             (0 0.0075 0.06);
    }

    uniformPoints // samples data uniformly across a line based on no. of points
    {
        type            uniform;
        axis            distance;
        start           (0 -0.0075 0.06);
        end             (0 0.0075 0.06);
        nPoints         300;
    }

    );


    fields
    (
    T
    U
    );
}
```

A similar thing should be used to write boundaryData.

I can include this in controlDict and runFunctionObjects.

- Let me just try doing boundaryData of velocity.

- https://www.openfoam.com/documentation/guides/latest/doc/guide-fos-sampling-surfaces.html

- this is the correct function object for sampling surface data.

- An example of use is found here:

- https://develop.openfoam.com/Development/openfoam/-/blob/master/tutorials/incompressible/

use buoyantPimpleFoam -postProcess

to start the postprocess functions.

So after some debugging, I get the correct syntax and this works, T and U are on the patches, but somehow the temperature is uniform on the patch (strange!).

At least the code syntax is right.

Argh so annoying, my changes were undone..

```
theodore_ong@LAPTOP-FK3MEHTI:~/GitHub_OpenFoam_workspace/turbulenceModelling/OpenFOAM_Examples/Qu
asi_DNS_Dowtherm_Mixed_Conv$ git push
Username for 'https://github.com': theodoreOnzGit
Password for 'https://theodoreOnzGit@github.com':
Counting objects: 14, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (14/14), 1.38 KiB | 1.38 MiB/s, done.
Total 14 (delta 12), reused 0 (delta 0)
error: RPC failed; curl 56 GnuTLS recv error (-12): A TLS fatal alert has been received.
fatal: The remote end hung up unexpectedly
fatal: The remote end hung up unexpectedly
Everything up-to-date
```

19.0.5.1 Windows Subsystem for Linux 2 Github bug in Jun 2020 And I cannot push my changes up to github... I wonder why.

There is a possibility the remote hangup is due to a WSL2 error.

I thought it was due to size and so I uploaded leaner versions of the file to a new repo (turbulenceModelling2)

Even when I ran ./Allclean on pretty much everything there, the same error came up.

I'm trying this on linuxMint to see if it gives me the same issue.

linuxMint gives me an upload even with large files and after cleaning up all the commit history IT'S OK.

I'm suspectin it's a WSL 2 problem, so bye bye WSL 2..

WSL works perfectly for such issues, no problem there.

19.0.5.2 In case my Boundary Conditions weren't Correct... Resetting BCs for the original case?

I didn't do the names of the blockMesh right, I thought that means that the flow will be like such:
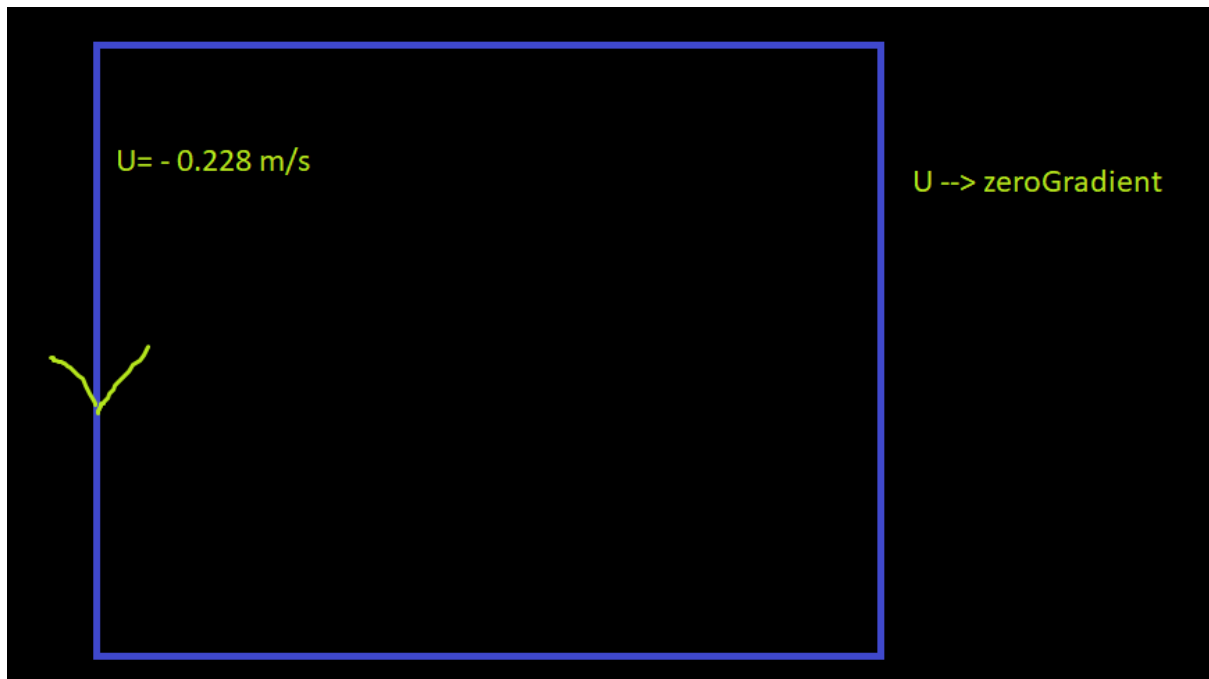
But this is not the case.

How I name the BCs has marginal impact on the flow since I use a forcing function for the flow... and the conditions are cyclic anyhow. The only impact is this: that the mesh sizing spanwise and streamwise will be different.

Shouldn't be an issue now though...

So i have found a way to place boundary data in surface format from existing fields, eg. T U on my github.

Repo is here: https://github.com/theodoreOnzGit/turbulenceModelling.git.

Now I actually want to have my data as including turbulent quantities for the inlet fields. Now, this will actually calculate all relevant fields, eg. k, epsilon, omega, R (Reynold's stress tensor), L (mixing length), I (turbulence intensity), nuTilda (spalart allmaras variable). This enables my field from k-omega SST IDDES to be applied also to SA-IDDES fields. That's pretty good!

In addition i can map alphat as well, which should prove useful. For heat transfer, but maybe that part I'll do later. Thankfully the entry is well documented!

19.0.5.3 Interpolation causes issues with TurbulentDFSEMInlet

Now i'm getting an error: my fields read right but when some values of reynold's stress tensor are less than 0 (indicative of backscatter).

What i notice though about the fields is that the values close to 0 are often minute (10^{-6}) whereas the positive values are on the order 10^{-4}. The idea is to replace all the negative values with 0. Using the sed fucntion sed

-i "s/-*/0/g" <filename>

to replace all the negative values. Unfortunately, the values themselves are in scientific format eg. e-07. So i'll need to instead use

sed -i "s/-1.*/0/g" <filename>

in order to replace all the relevant data points.

Furthermore, sed actually replaces following the –1.* with 0, including all following numbers!

Doesn't look like sed makes it easy!

Another way to approach: use the RANS mesh case. (unfortunately that too has negative numbers)

So got to use sed

The way is this:

URL reference: https://www.unix.com/unix-for-dummies-questions-and-answers/246213-sed-awk-find-negative-numbers-replace-1-a.html

sed "s/-[0-9].[0-9]*/0/g" <filename>

unfortunately that also replaces all the powers with e-09 to e0. Placing one more specifier works!

sed "s/-[0-9].[0-9][0-9]*/0/g" <filename>

Ayio, again sed does not like R_xx=0, it must be >0.

Also to have better control of values replaced by spacebar, one can use

sed -i 's/0 /1e-09 /g' <filename>

to do replace specific value

Looks like i mseed it up but, then again i look at the R file, only one value in R_XX is negative, may as well change that.

Looks like that solved it! Must have been some interpolation error. Only face 76 seems to be an issue

Can i interpolate such that it doesn't give me this issue?

Or i can make interpolate false!

That should debug things.

After debugging, interpolate false works, the fields can now be read. But issue is that again i have the IOStream error....

ima try to leave out the nCellsPerEddy case...

Doesn't work... ugh.

Can check turbulentDFSEMInlet Source code to see the correct input mode:

https://www.openfoam.com/documentation/guides/latest/api/classFoam_1_1turbulentDFSEMInletFv

im still not really sure, but let erase the first line: value 1264 from points, and that should be ok. (doesn't really make a diff whether u have it or not)

Done this and still gives the error (error in IOstream)

Looks like the points on the right side BC were missing!

now everything works well. the simulation is running without stability issues

I also want to map pressure and temperature fields as well as alphat type fields and nut type fields if possible.

19.0.6 Results of IDDES Run 6

syntax is ok but courant Number still blows up =(

Probably does so on injected turbulence.

The other time i did the entrance BC, it was stable probably because turbulence structures were not well formed yet.

However, it was stable till 0.1595s.

My other option is to increase the corrections in my pimple loop. (ie give it more outer-correctors)

Okay even that crashed, but it crashed much faster though! (within one calculation it crashed)

This may give some interesting insight.

(but i'll leave it for later, in another section)

The trend is a steadily increasing maxCo, for which the timestep adjusts to compensate to make maxCo about 0.5. At timestep = 7e-05 timestep decreases steadily. At about timestep adjusted to 5.5e-05, the iterations get more and more long to calculate, lag is noticeable.

At timestep below 5e-05, oscillations increase rapidly. Solver instability happens. Why this number though and not other numbers...

The mean Co also starts to jump exponentially at timestep below 5e-05.

Can look here for examples:

https://cfd.fossee.in/case-study-project/case-study-run/20

20 IDDES Run 7: Switching off Natural Convection

This is just conjecture, but since the pimpleFoam case works, i suspect buoyantPimple-Foam will work without gravity.

Funny though, even without gravity the solver is facing the same issues.

Apparently it's the maxCo which is driving the timestep ultrasmall to keep maxCo at 0.5, the mean Co actually remains relatively small 0.016 ish. So at around 0.159727s again, it fails. Even without natural convection. So it's a problem then with resolving the forced convection bit more like. Natural convection does not play too much of a role in these instabilities.

21 IDDES Run 8: (not done)

channel395DFSEMInlet case uses kEqn, i use kOmegaSSTIDDES. While kEqn resolves turb BL, kOmegaSSTIDDES will have a RANS region gradually transitioning to LES region.

Could the turbulence model be the issue? I suspect not, even with no turbulence model, the buoyantPimpleFoam Solver Crashes.

I suppose i can switch the model to kEqn and Spalart Allmaras IDDES and see what happens... But i'm taking off from turbulence modelling for now.

21.1 IDDES Run 9: switching from buoyantPimpleFoam to buoyantBousinesqPimpleFoam

Part IV

Bibliography and Citation

References

Komen, E., Shams, A., Camilo, L., & Koren, B. (2014). Quasi-DNS capabilities of Open-FOAM for different mesh types. *Computers & Fluids*, *96*, 87–104. doi:https://doi.org/10.1016/j.compfluid.2014.02.013

Ding, P., Wang, S., & Chen, K. (2020). Numerical study on turbulent mixed convection in a vertical plane channel using hybrid RANS/LES and LES models. *Chinese Journal of Chemical Engineering*, *28*(1), 1–8. doi:https://doi.org/10.1016/j.cjche.2019.04.007

Spalart, P. R., Deck, S., Shur, M. L., Squires, K. D., Strelets, M. K., & Travin, A. (2006). A new version of detached-eddy simulation, resistant to ambiguous grid densities. *Theoretical and computational fluid dynamics*, *20*(3), 181.

Tu, J., Yeoh, G. H., & Liu, C. (2018). *Computational fluid dynamics: a practical approach*. Butterworth-Heinemann.

Scheel, J. D., Emran, M. S., & Schumacher, J. (2013). Resolving the fine-scale structure in turbulent Rayleigh–Bénard convection. *New Journal of Physics*, *15*(11), 113063.

Bejan, A. (2013). *Convection heat transfer*. John wiley & sons.

Perry, R. H. & Green, D. W. (2015). Perry's chemical engineers' handbook. *Mc Graw*.

You, J., Yoo, J. Y., & Choi, H. (2003). Direct numerical simulation of heated vertical air flows in fully developed turbulent mixed convection. *International Journal of Heat and Mass Transfer*, *46*(9), 1613–1627. doi:https://doi.org/10.1016/S0017-9310(02)00442-8

Yuan, X., Moser, A., & Suter, P. (1993). Wall functions for numerical simulation of turbulent natural convection along vertical plates. *International journal of heat and mass transfer*, *36*(18), 4477–4485.

Ampofo, F. & Karayiannis, T. G. (2003). Experimental benchmark data for turbulent natural convection in an air filled square cavity. *International Journal of Heat and Mass Transfer*, *46*(19), 3551–3572. doi:https://doi.org/10.1016/S0017-9310(03)00147-9

Haren, S. W. (2011). Testing DNS Capability of Open FOAM and STAR-CCM+. Master Thesis, Delft University of Technology, Delft.