Spectral solver for Navier-Stokes equations.

For incompressible flow, the Navier-Stokes have the form.

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v}\cdot\nabla)\vec{v} = -\nabla p + \nu \nabla^2 \vec{v} \quad (1)$$

$$\nabla \cdot \vec{v} = 0. \quad (2).$$

To solve them numerically with a spectral method, we bring them to the following form.

We rewrite:

$$(\vec{v}\cdot\nabla)\vec{v} = (\nabla\times\vec{v})\times\vec{v} + \frac{1}{2}\nabla(\vec{v}\cdot\vec{v}).$$

which can be verified by looking at the individual components.

(1) becomes:

$$\frac{\partial \vec{v}}{\partial t} + (\nabla\times\vec{v})\times\vec{v} = -\nabla p - \frac{1}{2}\nabla(\vec{v}\cdot\vec{v}) + \nu\nabla^2 v.$$

We define $\vec{\omega} = \nabla\times\vec{v}$ ⇝ vorticity.

$$P = p + \frac{1}{2}|\vec{v}|^2.$$

Then we obtain:

$$\frac{\partial \vec{v}}{\partial t} = -\vec{\omega}\times\vec{v} - \nabla P + \nu\nabla^2\vec{v}. \quad (3)$$

Analytically, this problem can be solved as follows:

Take the divergence of (3)

$$\nabla\cdot\frac{\partial \vec{v}}{\partial t} = -\nabla\cdot(\vec{\omega}\times\vec{v}) - \nabla\cdot\nabla P + \nu\nabla\cdot(\nabla^2\vec{v}).$$

The following terms drop out:

$$\nabla\cdot\frac{\partial \vec{v}}{\partial t} = \frac{\partial}{\partial t}(\nabla\cdot\vec{v}) = 0$$

$$\nu\nabla\cdot(\nabla^2\vec{v}) = \nu\nabla^2(\nabla\cdot\vec{v}) = 0.$$

and so we obtain: $\nabla^2 P = -\nabla\cdot(\vec{\omega}\times\vec{v})$

Putting this all together we get:

$$\vec{\omega} = \nabla \times \vec{v}$$
$$\nabla^2 P = -\nabla \cdot (\vec{\omega} \times \vec{v}).$$
$$\frac{\partial \vec{v}}{\partial t} = -\vec{\omega} \times \vec{v} - \nabla P + \nu \nabla^2 v.$$

These equations can be used as a starting point for various numerical methods.

Numerical method

Assume software that can compute:

$L_N v \longrightarrow$ Laplacian of v.
$G_N v \longrightarrow$ Gradient of v.
$C_N v \longrightarrow$ Curl of v.
$D_N v \longrightarrow$ Divergence of v.
$F_N v \longrightarrow$ Fourier transform of v.
$F_N^{-1} v \longrightarrow$ Inverse Fourier transform.

We work in fourier space.

Step 1: Want to advance velocity field with $-\vec{\omega} \times \vec{v}$ and $\nu \nabla^2 v$

a. Compute an approximation to $-\vec{\omega} \times \vec{v}$:

$$F_N \left[(F_N^{-1} C_N \hat{u}^n) \times (F_N^{-1} \hat{u}^n) \right].$$

b. Without using an integrating factor:

$$\dot{u}^n = -F_N \left[(F_N^{-1} C_N \hat{u}^n) \times (F_N^{-1} \hat{u}^n) \right] + \nu L_N u^n.$$
$$u^{n+1/2} = T(\hat{u}^n, \dot{u}^n).$$

where T is a time-stepping algorithm.
It is better to use an integrating factor.

Consider an equation of the form $\frac{\partial v}{\partial t} = R + \nu \nabla^2 v$.

In Fourier space:

$$\frac{\partial \hat{v}}{\partial t} = \hat{R} + \nu(-k^2)\hat{v} \Leftrightarrow \frac{\partial \hat{v}}{\partial t} + \nu k^2 v = \hat{R} \Leftrightarrow$$

$$\Leftrightarrow \frac{\partial \hat{v}}{\partial t} e^{\nu k^2 t} + \nu k^2 e^{\nu k^2 t} \hat{v} = \hat{R} e^{\nu k^2 t} \Leftrightarrow \frac{\partial}{\partial t}\left(\hat{v} e^{\nu k^2 t}\right) = \hat{R} e^{\nu k^2 t}.$$

We discretize using an explicit scheme: $u^{n+1} = u^n + \sum_{m=0}^{r} a_m \dot{u}^{n-m} \Delta t$

In our case:

$$(\hat{u} e^{\nu k^2 t})_{n+1} = (u e^{\nu k^2 t})_n + \sum_{m=0}^{r} a_m (\dot{u} e^{\nu k^2 t})_{n-m} \Delta t \quad , \text{ where } \dot{u}^n = \hat{R}^n \cdot \dot{u}$$

Write $t_n = n\Delta t$:

$$u^{n+1} \exp(\nu k^2 (n+1)\Delta t) = u^n \exp(\nu k^2 n \Delta t) + \Delta t \sum_{m=0}^{r} a_m \dot{u}^{n-m} \exp(\nu k^2 (n-m)\Delta t)$$

Divide with $\exp(\nu k^2 (n+1)\Delta t)$:

$$u^{n+1} = u^n \exp(-\nu k^2 \Delta t) + \Delta t \sum_{m=0}^{r} a_m \dot{u}^{n-m} \exp(\nu k^2 (-m-1)\Delta t) =$$

$$= \exp(-\nu^2 k^2 \Delta t)\left[u^n + \Delta t \sum_{m=0}^{r} a_m \dot{u}^{n-m} \exp(-\nu k^2 m \Delta t) \right]$$

The software must introduce another operator

$$I_n(c)\, u^n \longrightarrow e^{-ck^2} u^n.$$

It follows that

~~$u^{n+1} = I_N(-\nu\Delta t)\left[u^n + \Delta t \sum_{m=0}^{r} a_m \dot{u}^{n-m} \exp(\nu$~~

$$u^{n+1} = I_N(-\nu\Delta t)\left[u^n + \Delta t \sum_{m=0}^{r} a_m I_N(-\nu m \Delta t)\, \dot{u}^{n-m} \right]$$

Using integrating factor, the first step can be done:

$$\boxed{\begin{aligned} \dot{u}^n &= -F_N\left[(F_N^{-1} C_N \hat{u}^n) \times (F_N^{-1} \hat{u}^n) \right]. \\ u^{n+1} &= I_N(\nu\Delta t)\left[u^n + \Delta t \sum_{m=0}^{r} a_m I_N(\nu m \Delta t)\, \dot{u}^{n-m} \right] \end{aligned}}$$

Step 2: Time step the fields to account for the $-\nabla P$ term. We use an arbitrary Adams-Moulton scheme (implicit).

$$u^{n+1} = u^{n+1/2} - \Delta t\left(a G_N P^{n+1} + \sum_{j=0}^{r} b_j G_N P^{n-j}\right).$$

where (a, b_j, r) are the scheme constants.
P^{n+1} determined by the constraint $D_N \cdot u^{n+1} = 0$ $(\nabla \cdot \vec{v} = 0)$.

$$D_N u^{n+1} = D_N u^{n+1/2} - \Delta t\, D_N\left(a G_N P^{n+1} + \sum_{j=0}^{r} b_j G_N P^{n-j}\right) =$$

$$= D_N u^{n+1/2} - \Delta t\left(a D_N G_N P^{n+1} + \sum_{j=0}^{r} b_j D_N G_N P^{n-j}\right) = \qquad \rightsquigarrow D_N G_N = L_N.$$

$$= D_N u^{n+1/2} - \Delta t\left(a L_N P^{n+1} + \sum_{j=0}^{r} b_j L_N P^{n-j}\right) = 0 \Leftrightarrow$$

$$\Leftrightarrow a\Delta t\, L_N P^{n+1} = D_N u^{n+1/2} + \sum_{j=0}^{r} b_j L_N P^{n-j} \Leftrightarrow$$

$$\Leftrightarrow P^{n+1} = L_N^{-1}\left[\frac{1}{a\Delta t} D_N u^{n+1/2} - \frac{1}{a}\sum_{j=0}^{r} b_j L_N P^{n-j}\right] =$$

$$= \frac{1}{a\Delta t} L_N^{-1} D_N u^{n+1/2} - \frac{1}{a} L_N^{-1}\sum_{j=0}^{r} b_j L_N P^{n-j} =$$

$$= \frac{1}{a\Delta t} L_N^{-1} D_N u^{n+1/2} - \frac{1}{a}\sum_{j=0}^{r} b_j P^{n-j}$$

Using this result, our scheme becomes "explicit":

$$u^{n+1} = u^{n+1/2} - \Delta t\left(a G_N P^{n+1} + \sum_{j=0}^{p} b_j G_N P^{n-j}\right) =$$

$$= u^{n+1/2} - \Delta t\left[a G_N\left(\frac{1}{a\Delta t} L_N^{-1} D_N u^{n+1/2} - \frac{1}{a}\sum_{j=0}^{r} b_j P^{n-j}\right) + \sum_{j=0}^{pr} b_j G_N P^{n-j}\right]$$

$$= u^{n+1/2} - \Delta t\left[\frac{1}{\Delta t} G_N L_N^{-1} D_N u^{n+1/2} - \sum_{j=0}^{r} b_j G_N P^{n-j} + \sum_{j=0}^{r} b_j G_N P^{n-j}\right] =$$

$$= u^{n+1/2} - G_N L_N^{-1} D_N u^{n+1/2}. \longrightarrow \text{independent of } \Delta t, (a, b_j, r), \text{ previous timesteps!}$$

The second step is : $u^{n+1} = u^{n+1/2} - G_N L_N^{-1} D_N u^{n+1/2}$.

Algorithm (Incompressible Navier-Stokes)

1. $\dot{u}^n = -T_N F_N \left[(F_N^{-1} C_N \dot{u}^n) \times (F_N^{-1} u^n) \right]$.

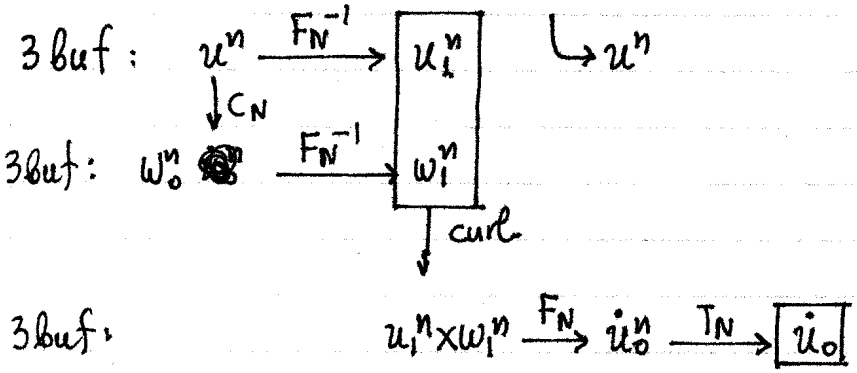
2. $u^{n+1/2} = I_N(\nu\Delta t) \left[u^n + \Delta t \sum_{m=0}^{r} a_m I_N(\nu m \Delta t)\, \dot{u}^{n-m} \right]$

3. $u^{n+1} = u^{n+1/2} - G_N L_N^{-1} D_N u^{n+1/2}$.

where T_N is a truncation filter, applied for anti-aliasing.

Implementation Details.

① The first step requires 9 memory buffers.



② The second step is just a summation but note that

$$I_N(a+b) = I_N(a) I_N(b).$$

We exploit this as follows:

Rotate buffers holding $\dot{u}^{n-m}$: $(\dot{u}^0, \dot{u}^1, \ldots, \dot{u}^r)$.
for $m = r, \ldots, 1$: $\dot{u}^m = I_N(\nu\Delta t)\, \dot{u}^{m-1}$
Compute new $\dot{u}^0$ (step 1).
for $m = 0, \ldots, r$: $u^n = u^n + \Delta t\, a_m \dot{u}^m$
$u^n = I_N(\nu\Delta t)\, u^n$

This pseudocode has computed $u^{n+1/2}$ from u^n.

③ The third step can be computed efficiently, component by component. OTOH we can grab the vorticity memory buffer and use it to store: $G_N L_N^{-1} D_N u^{n+1/2}$.
Let c_1 be the cube for one of the vorticity components.

```
c1 = DN u^{n+1/2}
c1 = LN^{-1} c1        !! minus
for d = 1,2,3          # for each dimension.
   c2 = c1
   c2 = (GN)_d c2
   (u^{n+1})_d = (u^n)_d + c2
endfor.
```

We reuse cubes c_1, c_2 that hold the vorticity.

Number of cubes wanted:
$(u^n)_1, (u^n)_2, (u^n)_3, (w^n)_1, (w^n)_2, (w^n)_3,$
iu^0, iu^1, iu^2, tmp.
16 cubes.
For an N^3 problem: $16N^3 \cdot 8$ bytes.
For $N = 2^n$: $M = 2^4 2^3 2^{3n} = 2^{3n+7}$ bytes.

$\Rightarrow$

$N = 32 \Rightarrow n = 5 \Rightarrow M = 2^{21} = 2\text{MB}$
$N = 64 \Rightarrow n = 6 \Rightarrow M = 2^{18+7} = 2^{25} = 33\text{MB}$
$N = 128 \Rightarrow n = 7 \Rightarrow M = 2^{21+7} = 268\text{MB}$
$N = 256 \Rightarrow n = 8 \Rightarrow M = 2^{24+7} = 2\text{GB}$
$N = 512 \Rightarrow n = 9 \Rightarrow M = 2^{27+7} = 2^{34} = 17\text{GB}.$

Algorithm

Part 1

Rotate buffers holding $\dot{u}^{n-m}$: $(\dot{u}^0, \dot{u}^1, \dots, \dot{u}^r)$
for $m = r, \dots, 1$: $\dot{u}^m = I_N(v\Delta t)\,\dot{u}^{m-1}$
Compute the new $\dot{u}^0$
$u^n \longrightarrow$ tmp , copy.
$\omega^n = F_N^{-1}\,\omega^n$
$u^n = F_N^{-1} u^n$
$\dot{u}^0 = u^n \times \omega^n$
$\dot{u}^0 = T_N F_N\,\dot{u}^0 \longrightarrow$ this is our new $\dot{u}^0$

Part 2

tmp $\longrightarrow u^n$ (recover u^n).
Compute $u^{n+1/2}$
for $m = 0, \dots, r$: $u^n = u^n + \Delta t\, a_m \dot{u}^m$
$u^n = I_N(v\Delta t)\,u^n \longrightarrow$ now u^n has $u^{n+1/2}$
Compute u^{n+1}
c_1 uses the memory of $(\omega^n)_1$ } one-dim cubes.
c_2 uses the memory of $(\omega^n)_2$ }
$c_1 = D_N u^n$
$c_1 = -L_N^{-1} c_1$
for $d = 1, 2, 3$
  $c_2 = c_1$
  $c_2 = (G_N)_d\, c_2$
  $(u^n)_d = (u^n)_d - c_2$
endfor. $\longrightarrow$ now u^n is u^{n+1}
$\omega^n = C_N u^n$. # compute the new vorticity.

It is best to split this up into two sub-algorithms:
$(u^n, \omega^n) \longrightarrow (u^{n+1/2}, F_N^{-1}\omega^n)$.
$(u^{n+1/2}, F_N^{-1}\omega^n) \longrightarrow (u^{n+1}, \omega^{n+1})$.

1. Initially both u^n, ω^n are given in Fourier space
2. $u^{n+1/2}$ in Fourier space, ω^n in <u>real space</u> !!!

- ~~Any processing that requires vorticity in real space can be done after the half-step.~~
- Algorithm is split into two parts. Any part that requires u^n, w^n in real space can be done after the first halfstep.

Solving incompressible combustion

① Fast chemistry approximation:

$$\frac{\partial z}{\partial t} = D\nabla^2 z - (u\cdot\nabla)z$$

$$Y_F = Y_{F,\emptyset}\,\frac{z - z_{st}}{1 - z_{st}}\,H(z - z_{st})$$

$$Y_O = Y_{O,\emptyset}\,\frac{z - z_{st}}{z_{st}}\,H(z_{st} - z)$$

where $z_{st} = \dfrac{Y_{O,\emptyset}}{r Y_{F,\emptyset} + Y_{O,\emptyset}}$ and $H(z) = \begin{cases} 1 & , z > 0 \\ 0 & , z < 0 \end{cases}$

Need to timestep z numerically. This can be done by:

1. $\dot{z}^n = -T_N F_N\left[(F_N^{-1}u^n)\cdot(F_N^{-1}G_N z^n)\right]$ convective form
   or $\dot{z}^n = -D_N T_N F_N\left[(F_N^{-1}u^n)(F_N^{-1}z^n)\right]$ conservative form.
2. $z^{n+1} = I_N(D\Delta t)\left[z^n + \Delta t \sum_{m=0}^{r} a_m I_N(mD\Delta t)\,\dot{z}^{n-m}\right]$

where we use the convective/conservative form alternating to eliminate aliasing. Note that since $p=1$, we may switch between these two forms according to:

$$\frac{\partial z}{\partial t} + \nabla\cdot(zu) = \frac{\partial z}{\partial t} + (u\cdot\nabla)z.$$

Since our algorithm requires the velocity field u in real space ~~we work out~~ we combine it with our fluids algorithm as follows:

Algorithm

1. $(u^n, w^n) \longrightarrow (F_N^{-1}u^n, F_N^{-1}w^n)$ (Part 1)
2. if (convective) $\dot{z}_n = -T_N D_N\left[(F_N^{-1}u^n)\cdot(F_N^{-1}G_N z)\right]$
   else $\dot{z}_n = -D_N T_N F_N\left[F_N^{-1}u^n)(F_N^{-1}z)\right]$
3. $(F_N^{-1}u^n, F_N^{-1}w^n) \longrightarrow (u^{n+1}, w^{n+1})$ (Part 2).
4. $z^{n+1} = I_N(D\Delta t)\left[z^n + \Delta t \sum_{m=0}^{r} a_m I_N(m\Delta t)\,\dot{z}^{n-m}\right]$

Now in more detail:

Algorithm 1: (Evaluation of $\dot{z}^{(n)}$) - step 2

```
# Rotate buffers holding ż^{n-m} : (ż^0, ż^1, ..., ż^r)
for m = r, ..., 1 : ż^m = I_N(DΔt) ż^{m-1}
# Compute the new ż^0
if (convective)
   u^n is given in real space
   ż^0 = 0  <-->  (*)  or  Ẏ_F = ẇ_F
   for d = 1,2,3        # loop over 3d to compute dot product
      tmp = (G_N)_d z^n
      tmp = F_N^{-1} tmp
      ż^0 = ż^0 + [F_N^{-1} u^n]_d tmp
   end for
   ż^0 = - T_N F_N ż^0
else if (conservative)
    u^n is given in real space
    tmp = F_N^{-1} z^n
    ż^n = 0
    for d = 1,2,3
       tmp2 = (F_N^{-1} u) tmp
       tmp2 = (G_N)_d T_N F_N tmp2
       ż^n = ż^n - tmp2
    endfor
endif.
end.
```

Remark: Two temporary cubes are required: tmp, tmp2. We may consider them available for other uses then as the need ~~arise~~ arises. For instance in the center-manifold computation they can be used to compute Y_F and Y_0.

Algorithm 2 : (advancing to Z^{n+1}). - step 4.

Compute Z^{n+1}
for $m = 0, \dots, r$: $Z^n = Z^n + \Delta t\, a_m \dot{Z}^m$
$Z^n = I_N(D\Delta t)\, Z^n$
$\longrightarrow$ now Z^n is Z^{n+1}.

Remark : As we will show, this very code can be used to time-step the equation for Y_F, except for a little modification in algorithm 1. It is best that one writes a generic solver to handle equations of the form:

$$\frac{\partial \xi}{\partial t} = D\nabla^2 \xi - (u \cdot \nabla)\xi + A$$

where A is arbitrary. and then specialize it for Y_F and Z.

(2) General chemistry solution.

$$\frac{\partial Y_F}{\partial t} = D\nabla^2 Y_F - (u\cdot\nabla) Y_F \blacklozenge - A Y_F Y_O \qquad (1)$$

$$\frac{\partial Y_O}{\partial t} = D\nabla^2 Y_O - (u\cdot\nabla) Y_O - rA Y_F Y_O. \qquad (2).$$

To reduce the stiffness we cast the problem in the (Y_F, Z) phase plane where $Z = \dfrac{rY_F - Y_O + Y_{O,\infty}}{rY_{F,\infty} + Y_{O,\infty}}$

Z follows a non-stiff equation:

$$\frac{\partial Z}{\partial t} = D\nabla^2 Z - (u\cdot\nabla) Z$$

and Y_O can be obtained from Y_F and Z.

$rY_F - Y_O + Y_{O,\infty} = Z(rY_{F,\infty} + Y_{O,\infty}) \Leftrightarrow$

$\Leftrightarrow Y_O = rY_F + Y_{O,\infty} - Z(rY_{F,\infty} + Y_{O,\infty})$

Recall that we defined $Z_{st} = \dfrac{Y_{O,\infty}}{rY_{F,\infty} + Y_{O,\infty}}$

$$\left(\text{Then } 1 - Z_{st} = \frac{rY_{F,\infty} + Y_{O,\infty} - Y_{O,\infty}}{rY_{F,\infty} + Y_{O,\infty}} = \frac{rY_{F,\infty}}{rY_{F,\infty} + Y_{O,\infty}}\right).$$

$$Y_O = rY_F + (rY_{F,\infty} + Y_{O,\infty}) Z_{st} - (rY_{F,\infty} + Y_{O,\infty}) Z =$$
$$= rY_F + (rY_{F,\infty} + Y_{O,\infty})(Z_{st} - Z) =$$
$$= rY_F + Y_{O,\infty}\frac{Z_{st} - Z}{Z_{st}}$$

Putting this altogether the problem is:

$\dfrac{\partial Y_F}{\partial t} = D\nabla^2 Y_F - (u\cdot\nabla) Y_F + \dot{w}_F$
$\dfrac{\partial Z}{\partial t} = D\nabla^2 Z - (u\cdot\nabla) Z$
$\dot{w}_F = -A Y_F \left(rY_F + Y_{O,\infty}\dfrac{Z_{st} - Z}{Z_{st}}\right).$
$Y_O = rY_F + Y_{O,\infty}\dfrac{Z_{st} - Z}{Z_{st}}$

The overall algorithm for time-stepping this problem is:

Algorithm

1. $(u^n, w^n) \longrightarrow (F_N^{-1}u^n, F_N^{-1}w^n)$
2. Compute $\dot{z}^0$ (algorithm explained)
3. Set $\dot{Y}_F^0 = \dot{w}_F$ (and supply the rest of the terms by the same algorithm as in 2)
4. Supply the rest of the terms by the same algorithm as in 2. Obtain full $\dot{Y}_F^0$.
5. $(F_N^{-1}u^n, F_N^{-1}w^n) \longrightarrow (u^{n+1}, w^{n+1})$
6. $$z^{n+1} = I_N(D\Delta t)\left[z^n + \Delta t \sum_{m=0}^{r} a_m I_N(m\Delta t)\dot{z}^{n-m} \right]$$
7. $$Y_F^{n+1} = I_N(D\Delta t)\left[Y_F^n + \Delta t \sum_{m=0}^{r} a_m I_N(m\Delta t)\dot{z}^{n-m} \right]$$

Remark: After step 3, $\dot{w}_F$ is available in real space or fourier space, as one chooses.

All the steps have been detailed out except for step 3 which follows:

$$\dot{w}_F = T_N F_N \left\{ -A(F_N^{-1}Y_F^n)\left[r F_N^{-1}Y_F^n + Y_{0,\sigma}\frac{z_{st} - F_N^{-1}z^n}{z_{st}} \right] \right\} =$$

$$= -A T_N F_N \left\{ (F_N^{-1}Y_F)\left[r F_N^{-1}Y_F^n + Y_{0,\sigma}\frac{z_{st} - F_N^{-1}z^n}{z_{st}} \right] \right\}$$

Characteristic length scales.

Let $u(k_1, k_2, k_3)$ be the velocity field in Fourier space.
Let $B(k) = \{(k_1, k_2, k_3) : \sqrt{k_1^2 + k_2^2 + k_3^2} = k\}$ be a ball.
The energy spectrum is given by:

$$E(k) = \frac{1}{4\pi k^2} \int_{B(\vec{k})} u(\vec{k}) u^*(\vec{k}) \, d\vec{k}$$

The three characteristic length scales are then computed by

a) Integral scale (large eddies)

$$\ell = \frac{3\pi}{4} \frac{\int_0^{+\infty} \frac{E(k)}{k} \, dk}{\int_0^{+\infty} E(k) \, dk}.$$

b) Taylor scale.

$$\lambda^2 = \frac{15 \nu u^2}{\varepsilon}$$

where ν = viscosity.
u = root mean square velocity.
ε = energy dissipation.

$$\varepsilon = 2\nu \int_0^{+\infty} k^2 E(k) \, dk.$$

c) Kolmogorov scale $\eta_K = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}$.

For a discretized velocity field, things are more tricky.
Recall that wavenumbers are given by $k = \frac{\pi j}{d}$
where $j \in [-N/2, N/2]$.

d) Reynolds number: $Re = \frac{u_{rms} \lambda}{\nu}$

The set of all wavenumber vectors is given by:

$$S = \left\{ \left(\frac{n\alpha}{d}, \frac{n\beta}{d}, \frac{n\gamma}{d} \right) \mid (\alpha, \beta, \gamma) \in \left\{ -\frac{N}{2}, \dots, \frac{N}{2} \right\} \right\}$$

$2d$ = edge-length of cube

For given (α, β, γ): $\quad k = \dfrac{n\sqrt{\alpha^2+\beta^2+\gamma^2}}{d}$

The max value of k is for $\alpha = \beta = \gamma = N/2$ $\quad k_{max} = \dfrac{nN\sqrt{3}}{2d}$

Now divide the interval $[0, k_{max}]$ to N_s pieces: $I_n = \left[\dfrac{(n-1/2)}{N_s} k_{max}, \dfrac{(n+1/2}{N_s} \right.$
and define the corresponding
"wavenumber-bands" by:

$$B(n; N_s) = \{ \vec{k} \in S : k \in I_n \}.$$

Note that: for $k = (n\alpha/d, n\beta/d, n\gamma/d)$

$$\vec{k} \in B(n;N) \Leftrightarrow k \in I_n \Leftrightarrow \frac{n-1/2}{N_s} k_{max} \leq \frac{n\sqrt{\alpha^2+\beta^2+\gamma^2}}{d} \leq \frac{n+1/2}{N_s} k_{max}$$

Since $\dfrac{k_{max}\, d}{n} = \dfrac{nN\sqrt{3}}{2d} \dfrac{d}{n} = \dfrac{N\sqrt{3}}{2}$:

$$\frac{n-1/2}{N_s} \frac{N\sqrt{3}}{2} \leq \sqrt{\alpha^2+\beta^2+\gamma^2} \leq \frac{n+1/2}{N_s} \frac{N\sqrt{3}}{2}.$$

So we find that

$$B(n; N_s) = \left\{ \left(\frac{n\alpha}{d}, \frac{n\beta}{d}, \frac{n\gamma}{d} \right) \mid \frac{n-1/2}{N_s} \frac{N\sqrt{3}}{2} \leq \sqrt{\alpha^2+\beta^2+\gamma^2} \leq \frac{n+1/2}{N_s} \frac{N\sqrt{3}}{2} \right\}$$

- Discrete energy spectrum $\rightarrow e(n)$ for $n \in \{0, 1, \dots, N_s\}$.

$$e(n) = \sum_{\vec{k} \in B(n;N_s)} u(\vec{k}) u^*(\vec{k})$$

The root-mean-square velocity is:

$$u_{rms} = \frac{1}{N^{3/2}} \left(\sum_{\vec{k} \in S} u(\vec{k}) u^*(k) \right)^{1/2}.$$

a) Integral length scale approximation.

Let $k_n = \frac{n}{N_s} k_{max} = n \frac{\pi\sqrt{3}}{2d}$. Now the length scales can be comp as follows:

$$l = \frac{3\pi}{4} \frac{\sum_{n=0}^{N_s} e(n)/k_n}{\sum_{n=0}^{N_s} e(n)}.$$

b) Taylor scale approximation.

$$\lambda^2 = \frac{15 \nu u_{rms}^2}{\varepsilon}$$

where ν = viscosity. and

u_{rms} = root-mean square velocity.

$$\varepsilon = 2\nu \sum_{n=0}^{N_s} k_n^2 e(n). = \text{energy dissipation.}$$

c) Kolmogorov microscale:

$$\eta_K = \left(\frac{\nu}{\varepsilon}\right)^{1/4}.$$

Length scales are useful ~> they tell us whether the current state of the velocity field is well-resolved.

Define $\eta_K k_{max} \alpha \geq B = O(1)$.

$l/d \leq A = O(1)$

Velocity ~~field~~ field initialization

Want ~~int~~ initial velocity field to have fully developed turbulence. One requirement is that

$$E(k) = a(k/c)^4 e^{-2(k/c)^2}.$$

Want to choose a, c such that the corresponding field is well-re
We also want to obtain the field.

- Choosing a, c

Let's compute the length scales: Simple integration yields

$$\int_0^{+\infty} E(k)\,dk = \frac{3}{32} ac\sqrt{\frac{\pi}{2}}, \quad \int_0^{+\infty} k^2 E(k)\,dk = \frac{15}{128} ac^3\sqrt{\frac{\pi}{2}}$$

and $$\int_0^{+\infty} \frac{E(k)}{k} = \frac{a}{8}$$

It follows that:

$$l = \frac{3\pi}{4} \frac{\int_0^{+\infty} (E(k)/k)\,dk}{\int_0^{+\infty} E(k)\,dk} = \frac{3\pi}{4} \frac{a/8}{\frac{3}{32} ac\sqrt{\frac{\pi}{2}}} = \frac{\sqrt{2\pi}}{c}$$

$$\varepsilon = 2\nu \int_0^{+\infty} k^2 E(k)\,dk = \frac{15}{64} ac^3 \nu \sqrt{\frac{\pi}{2}} \Rightarrow$$

$$\Rightarrow \eta_K = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4} = \left[\frac{\nu^3}{\frac{15}{64} ac^3 \nu \sqrt{\frac{\pi}{2}}}\right]^{1/4} = \left[\frac{64}{15} \frac{\nu^2}{ac^3} \sqrt{\frac{2}{\pi}}\right]^{1/4}$$

The field will be well-resolved if the following conditions hold:

$$\eta_K k_{max} \geq B.$$
$$l\,d \leq A$$

where $k_{max} = \frac{\pi N\sqrt{3}}{2d}$

$l/d \leq A \Leftrightarrow \frac{\sqrt{2\pi}}{cd}\, d \leq A \Leftrightarrow c \geq \frac{\sqrt{2\pi}}{dA}$. $\rightsquigarrow$ Choose $c = \frac{\sqrt{2\pi}}{Ad}$.

Now $n_K = \left[\frac{64}{15}\frac{V^2}{ac^3}\sqrt{\frac{2}{\pi}}\right]^{1/4}$ ⇝ must substitute c.

and $n_K k_{max} = \left[\frac{64}{15}\cdot\frac{V^2}{ac^3}\sqrt{\frac{2}{\pi}}\right]^{1/4} k_{max} = B \Leftrightarrow$

$\Leftrightarrow \frac{64}{15}\frac{V^2}{ac^3}\sqrt{\frac{2}{\pi}}\cdot k_{max}^4 = B^4 \Leftrightarrow a = \frac{64}{15}\frac{V^2}{B^4c^3}k_{max}^4\sqrt{\frac{2}{\pi}}$.

• Velocity field initialization.

Given a, c we know $E(k)$. Compute $e(n)$ for $n \in \{0,1,\dots,N\}$ by

$$e(n) = \int_{I_n} E(k)\,dk$$

where the integral is computed with a simple integration method.

and. $$I_n = \left[\frac{(n-1/2)}{N_s}k_{max}, \frac{(n+1/2)}{N_s}k_{max}\right], \quad n \in \{1,\dots,N_s-1\}$$

$$I_0 = \left[0, \frac{k_{max}}{2N}\right], \quad I_{N_s} = \left[\frac{(N_s-1/2)}{N_s}k_{max}, k_{max}\right]$$

Now we want a velocity field $\vec{v}$ such that $\nabla\cdot\vec{v} = 0$. If we define $\vec{v}$ in terms of Λ by $\vec{v} = \nabla\times\vec{\Lambda}$ then $\nabla\cdot\vec{v} = \nabla\cdot(\nabla\times\vec{\Lambda}) = 0$.

~~If the components of Λ are independent and isotropic (i.e $\langle\Lambda_1^2\rangle_K = \langle\Lambda_2^2\rangle_K = \langle\Lambda_3^2\rangle_K = \sigma_\Lambda^2$) then~~

$$\sum_{\vec{k}\in B(n;N_s)} u(\vec{k})u^*(\vec{k}) = 2k_n^2\sigma_\Lambda^2(n)|B(n;N_s)|.$$

Suppose that the $\Lambda(\vec{k})$ components with $\vec{k}\in B(n;N_s)$ are distributed with mean 0 and variance $\sigma_\Lambda(n)$. Then the energy spectrum will be:

$$e(n) = 2k_n^2\sigma_\Lambda^2(n)|B(n;N_s)|$$

But $e(n)$, k_n, $|B(n;N_s)|$ are known! So:

$$\boxed{\sigma_\Lambda(n) = \frac{e(n)}{k_n^2 |B(n;N_s)|}}$$

Note that in fourier space the components of Λ are complex.
so $\sigma_\Lambda^2(n) = \langle [Re\,\Lambda]^2 + [Im\,\Lambda]^2 \rangle_{\kappa \in B(n;N_s)}$.
→ the real and imaginary parts must be distributed by $\sigma_\Lambda(n)/2$ so that the whole is distributed by $\sigma_\Lambda(n)$.
WITH AN EXCEPTION!!!

!!! Caution: Since Λ is a real field we only store $0 \leq k_1 < k_{max}$ instead of $-k_{max} \leq k_1 \leq k_{max}$.

$$u(k_1, y, z) = u^*(-k_1, y, z) \Rightarrow$$

$$\Rightarrow u(k_1, k_2, k_3) = u^*(-k_1, -k_2, -k_3).$$

so for $k_2 = k_3 = 0$: $u(k_1, 0, 0) = u^*(-k_1, 0, 0)$.
Since we do not store $k_1 < 0$ this is implied.
However, for $k_1 = k_3 = 0$: $u(0, k_2, k_3) = u^*(0, -k_2, -k_3)$
is not implied because both $u(0, k_2, k_3)$ and $u^*(0, -k_2, -k_3)$ are stored. To deal with this issue:
Force $u(\vec{k})$ with $k_1 = 0$ to be real → real part distribute u $\sigma_\Lambda(n)$.
Then Force $u(0, -k_2, -k_3) = u(0, k_2, k_3)$, $\forall k_2 \geq 0$, $\forall k_3 \geq 0$.

Once the velocity field is constructed, we timestep it until the energy spectrum has developped sufficiently for self-similar decay. During self similar decay, the energy spectrum varies according to the following relation:

$$E(k) = u_{rms}^2 \, l \, f(kl).$$

where f is a universal function. ~~We know that~~
The only way to find f is to timestep our field until we reach self-similar decay. ~~Then we use f to create a new velocity field with $E(k)$ that matches f.~~

When that happens f is given by:

$$E(k) = u_{rms}^2 \, l f(kl) \Rightarrow f(kl) = \frac{E(k)}{u_{rms}^2 l} \Rightarrow f(x) = \frac{E(k/l)}{u_{rms}^2 l}$$

The field is ready to be used, BUT we may want to rescale it so that it fits our resolution better. Given f we want a new field of the form:

$$E'(k) = u'^2_{rms} \, l' f(kl').$$

We choose l' such that

$$l'/d = A \Rightarrow l' = Ad.$$

Now we must choose u'_{rms} such that

$$\eta'_k \, k_{max} = B.$$

and this is a bit of work to derive.
Answer: To force a desired η'_k, choose:

$$u'^2 = u^2 \left(\frac{l'}{l}\right)^2 \left(\frac{\eta_k}{\eta_{k'}}\right)^4.$$

Derivation:

$$\eta'_k = \left(\frac{\nu^3}{\varepsilon'}\right)^{1/4} \Leftrightarrow \eta'^4_k = \frac{\nu^3}{\varepsilon} \Leftrightarrow \varepsilon' = \frac{\nu^3}{\eta'^4_k}, \quad \text{also} \quad \varepsilon = \frac{\nu^3}{\eta_k^4}$$

It remains to relate ε' with u_{rms}.

$$\varepsilon' = \int_0$$

$$\varepsilon' = 2\nu \int_0^{+\infty} k^2 E'(k)\,dk = 2\nu \int_0^{+\infty} u'^2_{rms} \, l' f(kl')\,dk =$$

$$= 2\nu \int_0^{+\infty} u'^2_{rms} \, l' \frac{E(kl'/l)}{u^2_{rms} l}\,dk =$$

$$\varepsilon' = 2\nu \int_0^{+\infty} k^2 E'(k)\,dk = 2\nu \int_0^{+\infty} k^2 u'^2_{rms}\, l' f(kl')\,dk =$$

$$= 2\nu \int_0^{+\infty} k^2 u'^2_{rms}\, l' \frac{E(kl'/l)}{u^2_{rms} l}\,dk = 2\nu \frac{u'^2_{rms} l'}{u^2_{rms} l} \int_0^{+\infty} k^2 E(kl'/l)\,dk =$$

$$= 2\nu \frac{u'^2_{rms} l}{u^2_{rms} l'} \int_0^{+\infty} (kl'/l)^2 E(kl'/l)\, d(kl'/l) =$$

$$= 2\nu \frac{u'^2_{rms} l}{u^2_{rms} l'} \int_0^{+\infty} k' E(k')\,dk' = \frac{u'^2_{rms} l}{u^2_{rms} l'}\,\varepsilon. = \frac{u'^2_{rms} l}{u^2_{rms} l'} \frac{\nu^3}{\eta_k^4}$$

It follows that:

$$\varepsilon' = \frac{\nu^3}{\eta_k'^4} \Leftrightarrow \frac{u'^2_{rms} l}{u^2_{rms} l'} \frac{\nu^3}{\eta_k^4} = \frac{\nu^3}{\eta_k'^4} \Leftrightarrow$$

$$\Leftrightarrow u'^2_{rms} = u^2_{rms} \left(\frac{l'}{l}\right)^2 \left(\frac{\eta_k}{\eta_k'}\right)^4 . \quad \square.$$

with l', u' known we rescale the spectrum. $\rightsquigarrow E'(k)$.
Now we construct a brand new field with the more "correct" spectrum. We can repeat this process as many times as we want. Usually once is enough.

- Conditions for self-similar decay.

How do we know that the timestepped solution has reached the point of self-similar decay?
Condition 1 $\rightsquigarrow$ All the velocity length scales are increasing.
Condition 2 involves a quantity called velocity derivative skewness
Let $u = (u_1, u_2, u_3)$ be the velocity field.
The velocity derivative skewness is defined by:

$$s(t) = -\frac{\langle (\partial u_1/\partial x)^3 \rangle |_t}{\langle (\partial u_1/\partial x)^2 \rangle^{3/2} |_t}$$

$s(t)$ measures the rate with which energy flows from the low energies to the high energies.
Self similar decay $\rightsquigarrow$ AFTER $s(t)$ peaks. !!!

Scalar field initialization.

Once the $\vec{u}$ field is initialized we must initialize the fields Z and Y_F. For premixed scalars $Y_F = 0 \vee Y_F = 1$ and $Z = 0 \vee Z = 1$ for each point. $\rightsquigarrow$ infinitely steep gradients.
Therefore initialization must be more involved.
For generality sake, let $S(x,y,z)$ be the scalar field
Let $\hat{S}(\vec{k}) = \hat{S}(k_1, k_2, k_3)$ be in Fourier space.
Want blobs of size $1/k_s$.

a) Let

$$S(\vec{k}) = \left[\frac{m(k)}{4\pi k^2}\right]^{1/2} e^{2\pi i \vartheta(\vec{k})}$$

where $m(k) = \begin{cases} 1 & , \ k_s - 1/2 \leq k \leq k_s + 1/2. \\ 0 & , \ \text{otherwise} \end{cases}$

$\vartheta(\vec{k}) =$ uniformly distributed numbers from 0 to 1.

b) $\hat{S}(\vec{k}) \rightsquigarrow S(\vec{x})$
Sort the values of $S(x)$. Lower half set to zero.
Upper half set to 1.

c) Now iterate as follows:

$$S(\vec{x}) \longrightarrow S(k) \longrightarrow S(k)F(k) \longrightarrow S_1(\vec{x})$$

where $F(k) = \begin{cases} 1 & k \leq k_c \\ (k_c/k)^2 & k_c < k < k_{max} \\ 0 & k > k_{max}. \end{cases}$

k_c controls how much premixed we want our fields to be

d) For all points with $S(x) > 1 \Rightarrow$ set $S(x) = 1$.
$S(x) < 0 \Rightarrow$ set $S(x) = 0$.
Then apply filter in c. Repeat until $0 \leq S(x) \leq 1$ everywhere