# Notes on computational fluid dynamics

Cesar B Rocha

June 10, 2015

This are notes I've been writing for my self study as part of the class "Computational Fluid Dynamics" (MAE 290C), taught by Prof. Juan C. del Alamo in Winter 2015. I'm also including material beyond the class syllabus.

I claim no originality to the content of these notes. In particular, I'm loosely following del Alamo's class notes, and the following books

- *Spectral methods: Fundamentals in single domains* by Canuto et al.;
- *Numerical Renaissance* by Bewley;
- *Numerical Methods for Fluid Dynamics: With applications to Geophysics* by Durran.
- *Numerical Methods for Conservation Laws* by LeVeque.

## 1 Introduction

We will study methods for solving the Navier-Stokes (NS) equations

$$\partial_t \vec{u} + \vec{u} \cdot \nabla \vec{u} = -\frac{\nabla p}{\rho} + \nu \triangle \vec{u}\,, \tag{1}$$

where the laplacian is

$$\triangle \stackrel{\text{def}}{=} \nabla \cdot \nabla \tag{2}$$

The momentum equation (1) is complemented by the conservation of mass

$$\nabla \cdot \vec{u} = 0\,. \tag{3}$$

To close the system, we will also need a thermodynamic equation, which we will introduce later.

The advective term $\vec{u} \cdot \nabla \vec{u}$ gives a hyperbolic flavor to the NS equations. Its quadratic non-linearity typically prohibits analytic solutions. It gives rise to fascinating phenomena like turbulence. In contrast the linear viscous term $\nu \triangle \vec{u}$ gives a parabolic flavour the NS equations.

## 2 Temporal discretizations of the NS equations

We consider a one-dimensional linear version of (1), and drop the pressure gradient term for simplicity,

$$\partial_t u + c\,\partial_x u = \nu \partial^2_{xx} u\,, \tag{4}$$

where $c$ is a constant speed. Fourier transforming in the $x$-direction we obtain

$$\partial_t \hat{u} = \underbrace{-(\mathrm{i}c\,k + \nu k^2)}_{\stackrel{\text{def}}{=}\lambda}\,\hat{u}\,, \tag{5}$$

where $k$ is the wavenumber. (4) is an eigenproblem with eigenvalue $\lambda$. Decaying solutions have $\mathrm{Re}\,(\lambda) < 0$, and numerical schemes must represent this behavior. Figure 1 shows the eigenvalue as a function of $\nu$. A $\nu$ increases the real part of the eigenvalue becomes more negative, whereas for very small $\nu$ the eigenvalue are almost purely imaginary. This example illustrates one of the challenges for time marching numerical schemes. These schemes should have stability regions that comprise large negative real number and the imaginary axis.

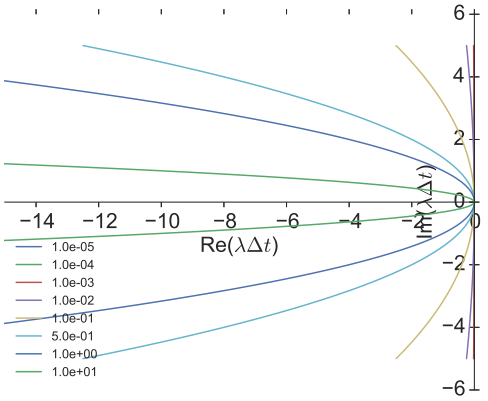


Figure 1: The eigenvalue $\lambda$ as function of the parameter $\nu$ with fixed speed $c = .5$.

One may be tempted to use implicit methods since they are unconditionally stable. However, from an implementation point of view, the implicit methods applied to the non-linear terms in (1) would lead to non-linear algebraic equations to be solved every time step, which can be very computationally costly.

For the advective terms we typically use two main schemes:

- Multi-step (e.g. Adams-Bashfort methods);
- Single-step, multiple sub-steps (e.g. Runge-Kutta methods).

Adams-Bashforth (AB) methods are more memory demanding, whereas Runge-Kutta (RK) schemes require more flops. The most efficient method depends on computer architecture. In recent computers memory, not CPU, is the main limitations, so we may use RK methods.

## 2.1 Runge-Kutta methods

Runge-Kutta (RK) are explicit, single-step, multi-sub-steps, schemes. The general form of an RK scheme for a non-linear equation

$$\partial_t u = F(u,t)\,, \tag{6}$$

is

$$u^{n+1} = u^n + \Delta t\,(b_1 k_1 + \ldots + b_M k_M) \tag{7}$$

$$\begin{aligned}
k_1 &= F(u^n, t^n)\,,\\
k_2 &= F(u^n + a_{21} k_1 \Delta t, t^n + c_2 \Delta t)\,,\\
&\vdots\\
k_M &= F(u^n + a_{21} k_1 \Delta t + \ldots + a_{M,M-1} k_{M-1} \Delta t, t^n, c_2 \Delta t + \ldots + c_M \Delta t)\,.
\end{aligned} \tag{8}$$

To systematically keep track of the coefficients, we introduce the Butcher table

Table 1: The Butcher table.

| | | | | | |
|---|---|---|---|---|---|
| $c_2$ | $a_{2,1}$ | 0 | $\dots$ | | 0 |
| $c_3$ | $a_{3,1}$ | $a_{3,2}$ | 0 | $\dots$ | $\vdots$ |
| $\vdots$ | $\vdots$ | $\ddots$ | $\ddots$ | | |
| $c_M$ | $a_{M,1}$ | $a_{M,2}$ | | $\dots$ | $a_{M,M-1}$ |

### 2.1.1 RK2, the simplest example

With M=2 we have

$$u^{n+1} = u^n + \Delta t \left(b_1 k_1 + b_2 k_2\right) , \tag{9}$$

with

$$\begin{aligned} k_1 &= F(u^n, t^n) , \\ k_2 &= F(u^n + a_{2,1} k_1 \Delta t, t^n + c_2 \Delta t) . \end{aligned} \tag{10}$$

There are four parameters $a_{2,1}$, $b_1$, $b_2$, and $c_2$. We choose these parameters to maximize the order of accuracy of the scheme. We expand $k_2$ about $k_1$

$$k_2 = k_1 + a_{21} k_1 \Delta t \, \partial_u F(u^n, t^n) + c_2 \Delta t \partial_t F + \mathcal{O}(\Delta t^2) . \tag{11}$$

Thus

$$u^{n+1} = u^n + \Delta t \left[(b_1 + b_2) F(u^n, t^n) + b_2 a_{2,1} \Delta t \, \partial_u F(u^n, t^n) + b_2 c_2 \Delta t \, \partial_t F(u^n, t^n)\right] \tag{12}$$

The Taylor expansion of $u^{n+1}$ about $u^n$ is

$$\begin{aligned} u^{n+1} = u^n + \Delta t \partial_t u|^n + \frac{\Delta t^2}{2} \partial_{tt}^n u|^n + \mathcal{O}(\Delta t^3) = \\ u^n + \Delta t \partial_t u|^n + \partial_t F(u^n, t^n) + \partial_u F(u^n, t^n) F(u^n, t^n) + \mathcal{O}(\Delta t^3) , \end{aligned} \tag{13}$$

where the equality follows from the governing equation (6). Matching terms in (13) and (12) gives

$$\begin{cases} b_1 + b_2 = 1; \\ b_2 c_2 = \frac{1}{2}; \\ b_2 a_{2,1} = \frac{1}{2}. \end{cases} \tag{14}$$

The system (14) is underdetermined: there are 4 unknowns but only 3 equations. Choosing $c_2 \equiv c$ as a free parameter, we obtain

$$a_{2,1} = c , \qquad b_2 = \frac{1}{2c} , \qquad \text{and} \qquad b_1 = 1 - \frac{1}{2c} . \tag{15}$$

The parameter $c$ can be chosen to give the best accuracy. Typically $\frac{1}{2} \le c \le 1$. The linear stability is independent of $c$. For the linear problem $F = \lambda u$. Thus

$$u^{n+1} = u^n \left(1 + \lambda \Delta t + \frac{(\lambda \Delta t)^2}{2}\right) . \tag{16}$$

Stability requires

$$\sigma = \left| 1 + \lambda \Delta t + \frac{(\lambda \Delta t)^2}{2} \right| \le 1 . \tag{17}$$

Along the real axis, the stability the stability constrain (17) is

$$1 + \lambda_r \Delta t + \frac{(\lambda_r \Delta t)^2}{2} \le 1 , \tag{18}$$

which gives

$$-2 \le \lambda_r \Delta t \le 0\,. \tag{19}$$

Along the imaginary axis, the stability region only touches $\lambda_i = 0$. Figure 2 shows the linear stability region for he RK2 scheme.

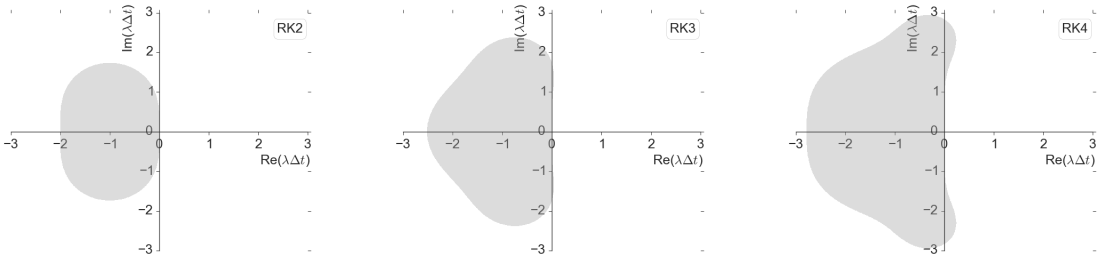


Figure 2: Stability region for Runge-Kutta schemes.

## 2.2 RK4

Derivation of higher-order RK schemes is straightforward but clumsy. A popular scheme is RK4

$$u^{n+1} = u^n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)\,, \tag{20}$$

where

$$\begin{aligned} k_1 &= F(u^n, t^n)\,, \\ k_2 &= F(u^n + \tfrac{1}{2}k_1\Delta t, t^n + \tfrac{1}{2}\Delta t)\,, \\ k_3 &= F(u^n + \tfrac{1}{2}k_2\Delta t, t^n + \tfrac{1}{2}\Delta t)\,, \\ k_4 &= F(u^n + k_3\Delta t, t^n + \Delta t)\,. \end{aligned} \tag{21}$$

This RK4 is one of my favorite time marching schemes. It is easy to implement, and has good stability properties (see figure 2).

## 2.3 A low-storage RK3 scheme

To simply notation we write

$$\partial_t u = \mathsf{L}(u) + \mathsf{H}(u)\,, \tag{22}$$

where $\mathsf{L}$ comprises the linear terms and $\mathsf{H}$ contains the nonlinear terms. The low-storage RKW3 scheme is in the form

$$u^{n+1} = u^{\star\star} + \Delta t[\mathsf{L}(\alpha_3 u^\star + \beta_3 u^{n+1})] + \gamma_3 \mathsf{H}(u^{\star\star}) + \xi_2 \mathsf{H}(u^\star)\,, \tag{23}$$

where

$$u^\star = u^n + \Delta t[\mathsf{L}(\alpha_1 u^n + \beta_1 u^\star) + \gamma_1 \mathsf{H}(u^n)]\,, \tag{24}$$

and

$$u^{\star\star} = u^\star + \Delta t[\mathsf{L}(\alpha_2 u^\star + \beta_2 u^{\star\star}) + \gamma_2 \mathsf{H}(u^\star) + \xi_1 \mathsf{H}(u^n)]\,, \tag{25}$$

with

$$\gamma_1 = \frac{8}{15}\,, \qquad \gamma_2 = \frac{5}{12}\,, \qquad \text{and} \qquad \gamma_3 = \frac{3}{4}\,, \tag{26}$$

$$\beta_1 = \frac{37}{160}\,, \qquad \beta_2 = \frac{5}{24}\,, \qquad \text{and} \qquad \beta_3 = \frac{1}{6}\,, \tag{27}$$

$$\alpha_1 = \frac{29}{96}\,, \qquad \alpha_2 = -\frac{3}{40}\,, \qquad \text{and} \qquad \alpha_3 = \frac{1}{6}\,, \tag{28}$$

and

$$\xi_1 = -\frac{17}{60}\,, \qquad \text{and} \qquad \xi_2 = -\frac{5}{12}\,. \tag{29}$$

# 3 $\theta$-schemes

For the linear viscous terms, it is customary to use implicit schemes. The $\theta$-schemes avoid rapid oscillations for large $\lambda\Delta t$, while Crank-Nicolson's second-order. These schemes have the form

$$u^{n+1} = u^n + \Delta t[(1-\theta)F^n + \theta F^{n+1}]\,. \tag{30}$$

Notice that CN is recovered with $\theta = \frac{1}{2}$, and implicit Euler is obtain with $\theta = 1$. Typically $\frac{1}{2} \le \theta \le 1$.

### Linear stability

The growth factor is

$$\sigma = \frac{1 + (1-\theta)\lambda\Delta t}{1 - \theta\lambda\Delta t}\,. \tag{31}$$

Thus

$$\lim_{\lambda\Delta t \to -\infty} \sigma = -\frac{1-\theta}{\theta} \le 1\,. \tag{32}$$

With $\theta = \frac{3}{4}$, the limit above is $-\frac{1}{3}$, a third of CN's limit.

### Order of accuracy

From the $\theta$-scheme we have

$$\frac{u^{n+1}}{u^n} = 1 + \lambda\Delta t + \theta\lambda^2\Delta t^2 + \mathcal{O}(\lambda^3\Delta t^3). \tag{33}$$

Comparing with the expansion for the exponential $\mathrm{e}^{\lambda t}$, we conclude that

$$\mathrm{Err}_\theta \sim \left(\frac{1}{2} - \theta\right)(\lambda\Delta t)^2\,. \tag{34}$$

With $\theta = \frac{3}{4}$ the error is

$$\mathrm{Err}_{3/4} \sim \frac{1}{4}(\lambda\Delta t)^2\,, \tag{35}$$

which is half of the error of the implicit Euler scheme.

# 4 Spectral methods

Spectral methods are the golden choice for numerical methods. In problems with simple geometry we express the solution as a linear combination of basis functions

$$u(x_j, t) = \sum_{n=1}^{\mathrm{N}} \hat{u}_n(t)\mathsf{b}_n(x_j)\,, \tag{36}$$

where $b_n(x)$ is the $n$'th basis function. A few desired properties for the basis are

Table 2: A comparison of $\mathrm{N}^2$ versus $5\mathrm{N}\log_2\mathrm{N}$

| N | $5\mathrm{N}\log_2\mathrm{N}$ | N |
|---|---|---|
| 8 | 120 | 64 |
| 16 | 320 | 256 |
| 32 | 800 | 1024 |
| 64 | 1920 | 4096 |
| 128 | 4480 | 16384 |
| 256 | 10240 | 65536 |

1. Orthogonality

   This property ensures accuracy. The basis formed by the set $\{b_n\}$ usually derives from a Sturm-Liouville problem on the domain of interest. The basis functions are typically orthogonal with respect to an inner product $<>$:

   $$< \mathsf{b}_n \, \mathsf{b}_m >= \alpha_{mn} \, \delta_{mn} \, , \tag{37}$$

   where $\delta$ is the Kronecker delta, and $\alpha$ is a normalization constant, usually taken to be 1. The coefficient $\hat{u}_n$ is then found by projection of the true solution on the $n$'th basis function:

   $$\hat{u_p} =< \mathsf{b}_p \, u(x,t) > \ . \tag{38}$$

2. Derivatives are known analytically

   This property allows easy computation of derivatives, e.g.

   $$\partial_x u(x_j,t) = \sum_{n=1}^{\mathrm{N}} \hat{u}_n \partial_x \mathsf{b}_n \, . \tag{39}$$

   We assume that the series for the derivative converges within the domain. This is always the case when $u(x,t)$ is periodic and sufficiently smooth within the domain. If $u(x,t)$ has cornes, discontinuities, etc differentiating term-by-term may result in a nonconvergent series. We will return to this issue later.

3. A computational efficient way to transform form the physical domain to the spectral domain.

   We should be able to easily and efficiently compute $u$ form $\hat{u}_n$, and vice-versa. An example of transform is

   $$\hat{u}_n(t) = \frac{1}{\mathrm{N}} \sum_{j=1}^{\mathrm{N}-1} u(x_j,t) \mathsf{b}_n(x_j) \, , \tag{40}$$

   Evaluation of the sum in above requires $\mathcal{O}(\mathrm{N})$ operations for every $n$. Thus the cost of the transform is $\mathcal{O}(\mathrm{N}^2)$. For a spectral method to be efficient we should be able to come up with algorithms that reduce this cost, and that can be implemented in parallel. For instance, the celebrated Fast Fourier Transform (FFT) reduces this cost to $\mathcal{O}(\mathrm{N}\log_2\mathrm{N})$. This results in a significant reduction of number of operations for large N (see table 3). For instance, for $\mathrm{N} = 256$ this results in a saving of $\sim 80\%$! Turbulence simulations would not be possible without the FFT.

## Fourier spectral methods

We will focus on the most used class of methods, namely the Fourier spectral methods. In Fourier methods, the set $\{\mathsf{b}_n\}$ is formed by complex exponentials $\{\exp(\mathrm{i}\kappa_n x_j)\}$ where

$$\kappa_n \stackrel{\text{def}}{=} \frac{2\pi n}{L}\,, \qquad n = -\frac{\mathrm{N}}{2}\,, -\frac{\mathrm{N}}{2}+1\,, \ldots\,, \frac{\mathrm{N}}{2}-2\,, -\frac{\mathrm{N}}{2}-1\,. \tag{41}$$

The inverse transform is defined by a truncated Fourier series

$$u(x_j) = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n\, \mathrm{e}^{\mathrm{i}\kappa_n x_j}\,, \tag{42}$$

where $\hat{u}_n$ is the $n$'th Fourier coefficient, which is generally complex-valued. Notice that the series (43) implies that $u$ is periodic with period $L$

$$u\left(-\tfrac{L}{2}\right) = u\left(\tfrac{L}{2}\right)\,. \tag{43}$$

The discrete Fourier transform (DFT) is

$$\hat{u}_n = \frac{1}{\mathrm{N}} \sum_{j=-\mathrm{N}/2}^{\mathrm{N}/2-1} u_j \mathrm{e}^{-\mathrm{i}\kappa_n x_j}\,. \tag{44}$$

We emphasize that one should sum only up to $\mathrm{N}/2-1$ since the $u$ is periodic, so that the data at $x_{\mathrm{N}/2}$ add no information to the problem. This definition is consistent with the inverse transform (43), following from the orthogonality of the discrete complex exponentials. This property is fundamental in many calculations, so we shall formally prove it. Consider the summation

$$\sum_{j=0}^{N-1} \mathrm{e}^{\mathrm{i}\left(\frac{2\pi}{\mathrm{N}}(n-s)\,j\right)}\,, \tag{45}$$

where we used $x_j = \frac{L}{\mathrm{N}} j$. The summation $S$ is a geometric series

$$s \stackrel{\text{def}}{=} 1 + r + r^2 + \ldots + r^{\mathrm{N}-1}\,, \qquad \text{with} \qquad r = \mathrm{e}^{\mathrm{i}\left(\frac{2\pi}{\mathrm{N}}(n-s)\right)}\,. \tag{46}$$

Hence

$$rs = r + r^2 + \ldots + r^{\mathrm{N}} \Longrightarrow s - rs = 1 - r^{\mathrm{N}}\,, \tag{47}$$

and therefore

$$s = \frac{1-r^{\mathrm{N}}}{1-r}\,, \qquad r \neq 1\,. \tag{48}$$

Now

$$r^{\mathrm{N}} = \mathrm{e}^{\mathrm{i}2\pi(n-s)} = 1\,, \qquad \text{if} \qquad n \neq s + m\,\mathrm{N}\,, \tag{49}$$

where $m$ is an integer. Thus

$$\sum_{j=0}^{N-1} \mathrm{e}^{\mathrm{i}\left(\frac{2\pi}{\mathrm{N}}(n-s)\,j\right)} = \begin{cases} \mathrm{N}: & n = s + m\mathrm{N}\,, \qquad m = 0 \pm 1 \pm 2 \pm 3 \ldots\,; \\ 0: & \text{otherwise}\,. \end{cases} \tag{50}$$

### Properties of the DFT

The discrete Fourier pair (43)-(44) satisfy the desired properties discussed above. This is the reason for the popularity of Fourier methods. The set $\{\exp(\mathrm{i}\kappa_n x_j)\}$ forms a complete orthogonal basis on any interval of size $L$ (e.g. $[0, L]$, $[-L/2, L/2]$, etc). Moreover, there exists the FFT algorithm performs the DFT very efficiently. In particular, the FFTW implementation in Fortran and C is very popular. A summary of important properties is

1. As mentioned above the complex exponential are orthogonal with respect to the simplest inner product
$$< \mathrm{e}^{\mathrm{i}\kappa_n x}\mathrm{e}^{\mathrm{i}\kappa_m x} > \stackrel{\text{def}}{=} \frac{1}{L}\int_0^L \mathrm{e}^{\mathrm{i}\kappa_n x}\mathrm{e}^{\mathrm{i}\kappa_m x}\mathrm{d}x = \delta_{mn}\,. \tag{51}$$

2. The function $u$ is periodic with period $L$
$$u(0) = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n \mathrm{e}^{\mathrm{i}\kappa_n \times 0} = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n \mathrm{e}^{\mathrm{i}\kappa_n \times L} = u(L) \tag{52}$$

3. The Fourier coefficients are periodic with period N
$$\hat{u}_{n+\mathrm{N}} = \frac{1}{\mathrm{N}}\sum_{j=-\mathrm{N}/2}^{\mathrm{N}/2-1} u_j \mathrm{e}^{-\mathrm{i}\,\frac{2\pi}{L}(n+\mathrm{N})\frac{jL}{\mathrm{N}}} = \frac{1}{\mathrm{N}}\sum_{j=-\mathrm{N}/2}^{\mathrm{N}/2-1} u_j \mathrm{e}^{-\mathrm{i}\,2\pi\frac{nj}{\mathrm{N}}} = \hat{u}_n\,. \tag{53}$$

4. The zeroth mode represents the average of the function. With $\kappa_0 = 0$, we have
$$\hat{u}_0 = \frac{1}{\mathrm{N}}\sum_{j=-\mathrm{N}/2}^{\mathrm{N}} u_j\,. \tag{54}$$

   Note that some implementations of the DFT do not divide by N in the definition (43). This is simply a matter of definition; one must alway check DocStrings and algorithm manuals to make sure which convention in being used.

### Truncation error, aliasing, and other potential problems

The definition of the inverse DFT assumes that $u$ has nice properties such as continuity of the function and its derivatives. If the function is discontinuous, then Gibbs oscillation might be introduce Gibbs oscillation, which amount for a 10% error near the discontinuities. The rapid Gibss oscillations can be significantly reduced by using filters.

A potential problem in spectral methods is aliasing of unresolved high-frequency modes. These modes alias back into low-frequency modes. For linear problems this is not an issue provided that the initial conditions do not contain unresolved high-frequency content. For non-linear problems, such as the NS equations, high-frequency modes are introduced every time step.

### Parseval's theorem

The Parseval's relation essentially says that the total energy in physical space is equal to the total energy in Fourier space
$$\sum_{j=0}^{\mathrm{N}-1} u_j u_j^* = \mathrm{N}\sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} |\hat{u}_n|^2\,. \tag{55}$$

The proof of this relationship is simple. We introduce the inverse DFT definition (43) into (55)

$$\sum_{j=0}^{\mathrm{N}-1} u_j u_j^* = \sum_{j=0}^{\mathrm{N}-1} \left( \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n \mathrm{e}^{\mathrm{i}\kappa_n x_j} \right) \left( \sum_{m=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_m^* \mathrm{e}^{-\mathrm{i}\kappa_m x_j} \right)$$
$$= \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \sum_{m=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n \hat{u}_m^* \underbrace{\left( \sum_{j=0}^{\mathrm{N}-1} \mathrm{e}^{\mathrm{i}(\kappa_n - \kappa_m)x_j} \right)}_{=\mathrm{N}\delta_{nm}} . \tag{56}$$

**Truncation error**

To study the truncation error associated with a finite N in (43) we consider the true solution

$$u_e = \sum_{n=-\infty}^{\infty} \tilde{u}_n \mathrm{e}^{\mathrm{i}\kappa_n x_j} , \tag{57}$$

where the Fourier coefficient is

$$\tilde{u}_n = \frac{1}{L} \int_0^L u_e(x) \mathrm{e}^{\mathrm{i}\kappa_n x} . \tag{58}$$

Notice that, by aliasing, we have

$$\hat{u}_n = \tilde{u}_n + \sum_{m=-\infty}^{\infty} \tilde{u}_{n+m\mathrm{N}} . \tag{59}$$

Thus

$$u_j = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \left( \tilde{u}_n + \sum_{m=-\infty}^{\infty} \tilde{u}_{n+m\mathrm{N}} \right) \mathrm{e}^{\mathrm{i}\kappa_n x_j} , \tag{60}$$

and the $L_2$-norm of the error is

$$\frac{1}{L} \int_0^L \left| u_e(x) - \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n \, \mathrm{e}^{\mathrm{i}\kappa_n x_j} \right|^2 \mathrm{d}x = \underbrace{\sum_{|n|>\mathrm{N}/2} |\tilde{u}_n|^2}_{\text{truncation error}} + \underbrace{\sum_{-n=\mathrm{N}/2}^{\mathrm{N}/2-1} \left| \sum_{m=-\infty}^{\infty} \tilde{u}_{n+m\mathrm{N}} \right|^2}_{\text{aliasing error}} . \tag{61}$$

**Convergence of the inverse DFT**

Integrating (58) by parts we obtain

$$\tilde{u}_n = \frac{1}{L} \int_0^L u_e(x) \mathrm{e}^{\mathrm{i}\kappa_n x} dx = -\frac{1}{L} \int_0^L \frac{u_e'(x) \mathrm{e}^{\mathrm{i}\kappa_n}}{\mathrm{i}\kappa_n} \mathrm{d}x$$
$$= \frac{1}{L} \int_0^L \frac{u_e''(x) \mathrm{e}^{\mathrm{i}\kappa_n x}}{(\mathrm{i}\kappa_n)^2} \mathrm{d}x . \tag{62}$$

We can continue the process above as long as the derivate are smooth and periodic. Hence

$$|\tilde{u}_n|^2 \lesssim (\Delta x)^{-2p} , \tag{63}$$

for all $p \in \mathbb{N}$. Thus for smooth functions in physical space, spectral methods beats the most accurate finite differences scheme. Spectral accuracy is also know as exponential accuracy. A rule of thumb is that for a given data spacing $\Delta x$, a spectral scheme with $3\Delta x$ does a better job than second-order finite-differences scheme.

**Gibbs phenomenon**

Consider the step function

$$S_e(x) = \begin{cases} 1\,, & 0 \le x < \pi\,; \\ 0\,, & \pi \le x < 2\pi\,. \end{cases} \tag{64}$$

A truncated Fourier series representation is

$$S_T(x) = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2} \tilde{S}_n \mathrm{e}^{-inx}\,, \tag{65}$$

where the Fourier coefficients are

$$\tilde{S}_n = \frac{1}{2\pi}\int_0^{\pi} S_e(x)\mathrm{e}^{inx}\mathrm{d}x\,. \tag{66}$$

Thus

$$S_T(x) = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2}\left(\frac{1}{2\pi}\int_0^{\pi} S_e(\xi)\mathrm{e}^{-in\xi}\mathrm{d}\xi\right)\mathrm{e}^{inx} = \frac{1}{2\pi}\int_0^{\pi} \underbrace{S_e(\xi)}_{=1}\underbrace{\left(\sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2}\mathrm{e}^{in(x-\xi)}\right)}_{\stackrel{\mathrm{def}}{=}\mathsf{D}_\mathrm{N}}\mathrm{d}\xi\,, \tag{67}$$

where $\mathsf{D}_\mathrm{N}$ is the Dirichlet Kernel

$$\mathsf{D}_\mathrm{N}(z) = \begin{cases} \sin\left(\frac{\mathrm{N}+1}{2}z\right): & \text{if} \quad z \neq 2\pi j\,; \\ \mathrm{N}+1: & \text{if} \quad z = 2\pi j\,. \end{cases} \tag{68}$$

Figure 3 shows the Dirichlet kernel for truncation indices N from 1 through 11. As N increases the kernel distributionally approaches a delta function at $z = 0$.

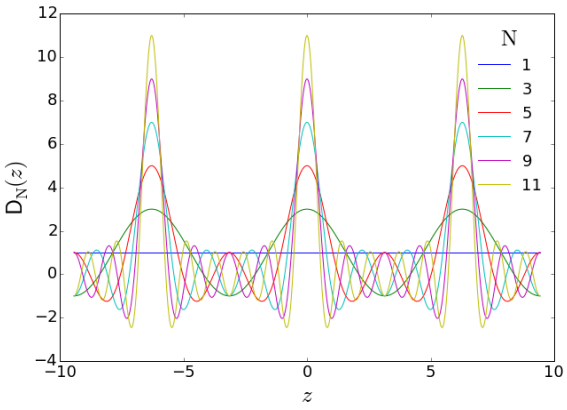


Figure 3: The Dirichlet kernel (68) for various truncation indices N.

We can rewrite the truncated series (67) by changing variables with $y \stackrel{\mathrm{def}}{=} x - \xi$

$$S_T(x) = \frac{1}{2\pi}\int_{x-\pi}^{x} \mathsf{D}_\mathrm{N}(y)\mathrm{d}y = \frac{1}{2\pi}\left[\int_0^x \mathsf{D}_\mathrm{N}(y)\mathrm{d}y + \underbrace{\int_{-\pi}^0 \mathsf{D}_\mathrm{N}(y)\mathrm{d}y}_{\approx\pi} + \underbrace{\int_{x-\pi}^{\pi} \mathsf{D}_\mathrm{N}(y)\mathrm{d}y}_{<<1}\right]. \tag{69}$$

We have

$$\int_0^x \mathsf{D}_\mathrm{N}(y)\mathrm{d}y \approx 1.08949\pi\,, \qquad \text{for} \qquad x \qquad \text{small}\,. \tag{70}$$

So that in the vicinity of the discontinuity at $x = 0$ we have

$$S_T(x) \approx 1.09\,. \tag{71}$$

That is, the truncated series $S_T$ overshoots the exact value by about 9%. This phenomenon is the Gibbs oscillation.

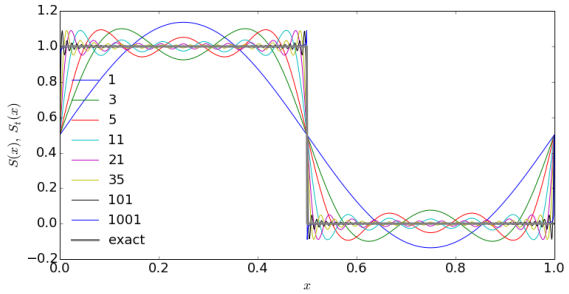


Figure 4: The representation of a step function by Fourier series under various truncation. Notice the Gibbs phenomenon near the discontinuity.

## Filters

To remove undesired Gibbs oscillations that may arise from discontinuities present in the initial conditions or developed by the equation (e.g., Burger's equation) we can use filters. The filtering process can operate in two levels

1. Level of equation

   We introduce an additional term, e.g., artificial viscosity or hyperviscosity, that will filter out highwanumber variability. For instance, in the one-dimensional Burger equation, we can add a fourth-order hyperviscosity

   $$\partial_t u + u\partial_x u = D\partial^4_{x^4} u\,. \tag{72}$$

   In Fourier space the solution to the linear hyperviscouss term is $\exp(-k^4 Dt)$, which makes clear its low-pass character.

2. Level of solution

   Once the solution is obtained, one can filter out highwavenumber oscillations using selective filters. A celebrated example is the use of the Lanczos $\sigma$-factos. In this approach, one substitute the original Fourier series by

   $$u_j = \sum_{n=-N/2}^{N/2-1} \sigma_n \hat{u}_n \mathrm{e}^{\mathrm{i}k_n x_j}\,, \tag{73}$$

   where $\sigma_n$ is a filter coefficient in the form

   $$\sigma_n = \sigma(\theta)\,, \qquad \theta \stackrel{\text{def}}{=} \frac{2\pi n}{\mathrm{N}}\,. \tag{74}$$

   The factor $\sigma(\theta)$ must satisfy the properties

(a) Reality;

(b) Symmetry about $\theta = 0$;

(c) $p - 1$ continuous differentiable for $p \geq 1$;

(d) $\sigma(\theta) = 0$, if $|\theta| \geq \pi$;
$\sigma(0) = 1$ (do not change the average);;
$\sigma^{(j)} = 0$, for $1 \leq \mathrm{J} \leq p - 1$.

Some examples of filter in the level of equation include

- Lanczos filter
  $\sigma(0) = 1$;
  $\sigma(\theta) = \frac{\sin\theta}{\theta}$.
- Raised cosine
  $\sigma(\theta) = (1 + \cos\theta)/2$.
- Sharpened raised cosine
  $\sigma(\theta) = \sigma_0^4(\theta)\left[35 - 84\sigma_0(\theta) + 70\sigma_0(\theta)^2 - 20\sigma_0(\theta)^2\right]$,
  where $\sigma_0$ is the raised cosine filter.
- Exponential filter of order $p$ (generalized viscosity)
  $\sigma(\theta) = \mathrm{e}^{-\alpha\theta^p}$,
  where $\alpha$ is an arbitrary parameter.

Figure 5 shows the step function truncated series with N = 9, and various filtered solutions.

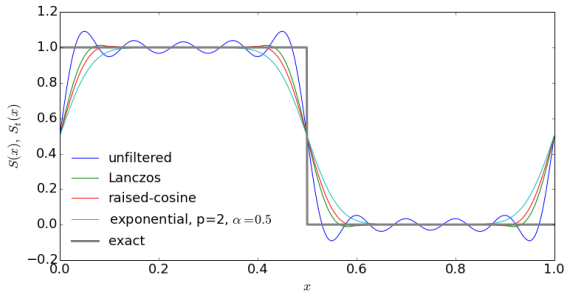


Figure 5: The representation of a step function by Fourier series with N = 9 and various filtered solutions.

### Aliasing

The calculation of Fourier transform of nonlinear terms of terms with nonconstant coefficients is very expensive. Consider the two variables

$$u_e(x) = \sum_{-\mathrm{N}/2}^{\mathrm{N}/2} \tilde{u}_n \mathrm{e}^{\mathrm{i}nx}\,, \qquad \text{and} \qquad v_e(x) = \sum_{-\mathrm{N}/2}^{\mathrm{N}/2} \tilde{v}_n \mathrm{e}^{\mathrm{i}nx}\,, \tag{75}$$

and their product

$$w_e(x) = u_e(x)v_e(x) = \sum_{-\mathrm{N}/2}^{\mathrm{N}/2} \sum_{-\mathrm{N}/2}^{\mathrm{N}/2} \tilde{u}_n \tilde{v}_m \mathrm{e}^{\mathrm{i}(n+m)x}\,, \qquad \text{and} \tag{76}$$

Hence

$$\tilde{w}_k = \frac{1}{2\pi}\int_\pi \sum_{-\mathrm{N}/2}^{\mathrm{N}/2}\sum_{-\mathrm{N}/2}^{\mathrm{N}/2} \tilde{u}_n\tilde{v}_m \mathrm{e}^{\mathrm{i}(n+m-k)x}\mathrm{d}x = \sum_{-\mathrm{N}/2}^{\mathrm{N}/2}\sum_{-\mathrm{N}/2}^{\mathrm{N}/2} \tilde{u}_n\tilde{v}_m \underbrace{\frac{1}{2\pi}\int_\pi \mathrm{e}^{\mathrm{i}(n+m-k)x}\mathrm{d}x}_{=\begin{cases}1: & \text{if} \quad m+n=k;\\ 0: & \text{otherwise.}\end{cases}} \tag{77}$$

Thus for series with N truncation index, the multiplication in physical space is equivalent to convolution in Fourier space. Computation of convolution sums requires $\mathcal{O}(\mathrm{N}^2)$ operations. The alternative is to use the so-called pseudo-spectral methods that quires $\mathcal{O}(\mathrm{N}\log_2\mathrm{N})$. In pseudo-spectral methods we compute the products in physical space, and then transform the products to Fourier space to march the system forward, then transform the variables back to physical space to compute the products, and so on. The problem with this approach is that is introduces aliasing. To see that, note that the discrete Fourier transform of (76) is

$$\begin{aligned}\hat{w}_k =& \frac{1}{\mathrm{N}}\sum_{j=-\mathrm{N}/2}^{\mathrm{N}/2-1}\left(\sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1}\sum_{m=-\mathrm{N}/2}^{\mathrm{N}/2-1}\hat{u}_n\hat{v}_m\mathrm{e}^{\mathrm{i}(m+n)x_j}\right)\mathrm{e}^{-\mathrm{i}kx_j}\\ =& \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1}\sum_{m=-\mathrm{N}/2}^{\mathrm{N}/2-1}\hat{u}_n\hat{v}_m\frac{1}{\mathrm{N}}\sum_{j=-\mathrm{N}/2}^{\mathrm{N}/2-1}\mathrm{e}^{\mathrm{i}(m+n-k)x_j}\,.\end{aligned} \tag{78}$$

Now recall that

$$\frac{1}{\mathrm{N}}\sum_{j=-\mathrm{N}/2}^{\mathrm{N}/2-1}\mathrm{e}^{\mathrm{i}(m+n-k)x_j} = \begin{cases}1: & \text{if} \quad m+n=k+pN\,,\ p=0,1,2,\ldots;\\ 0: & \text{otherwise.}\end{cases} \tag{79}$$

Thus

$$\hat{w}_k = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1}\sum_{m=-\mathrm{N}/2}^{\mathrm{N}/2-1}\hat{u}_n\hat{v}_m + \sum\sum_{m+n=k\pm\mathrm{N}}\hat{u}_n\hat{v}_m + \sum\sum_{m+n=k\pm 2\mathrm{N}}\hat{u}_n\hat{v}_m\ldots\,. \tag{80}$$

The first term on the right of (80) is the discrete Fourier transform of the product $uv$. The double sums represent aliasing terms. These terms are due to unresolved high frequency modes that project back into the lower frequency modes.

For quadratic nonlinearity we only need to consider the first term on the right of (80). The extrema alias frequencies are

$$(m+n)_{max} = \mathrm{N}-2\,, \qquad \text{and} \qquad (m+n)_{min} = -\mathrm{N}\,. \tag{81}$$

Thus we only need to consider frequencies from $-3\mathrm{N}/2$ through $3\mathrm{N}/2$.

## Dealiasing

There are two main techniques used to remove the aliased frequencies due to nonlinearities.

### Zero padding

We use discrete Fourier series for $u$, $v$, and $w$ with $\mathrm{M}\geq 3\mathrm{N}/2$, and set $\hat{u}_n = \hat{v}_m = 0$ for $m\geq \mathrm{N}/2$ before going to physical domain. We also set $\hat{w}_n$ for $n\geq \mathrm{N}/2$ after going back to Fourier space.

Notice that the worst case scenario is $n=m=-\mathrm{N}/2$:

$$-\frac{\mathrm{N}}{2}-\frac{\mathrm{N}}{2} = \frac{\mathrm{N}}{2}\pm\mathrm{M} \Rightarrow \mathrm{M} = \frac{3}{2}\mathrm{N}\,. \tag{82}$$

The computational cost of zero padding is $\mathcal{O}(3/2)^n$, where $n$ is the number of dimensions. Thus zero padding can be quite expensive, particularly in two- or three-dimensions.

**Phase shift**

Consider the product of $u$ and $v$ computed at the original grid

$$w(x) = u(x)v(x)\,, \tag{83}$$

and the products computed in a grid shifted by $\xi$

$$w(x + \Delta x) = u(x + \xi)v(x + \xi)\,. \tag{84}$$

The DFT of (83) is

$$\hat{w}_{o,k} = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \sum_{m=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n \hat{v}_m + \sum\sum_{m+n=k\pm\mathrm{N}} \hat{u}_n \hat{v}_m\,, \tag{85}$$

whereas the DFT of (84) is

$$\hat{w}_{\xi,k} = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \sum_{m=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n \hat{v}_m + \mathrm{e}^{\pm \mathrm{i}\mathrm{N}\xi} \sum\sum_{m+n=k\pm\mathrm{N}} \hat{u}_n \hat{v}_m\,. \tag{86}$$

The idea is to combine (85) and (86) in such a way as to eliminate the aliasing terms

$$\hat{w}_k = \frac{\hat{w}_{o,k} + \hat{w}_{\xi,k}}{2} = \hat{w}_{\xi,k} = \sum_{n=-\mathrm{N}/2}^{\mathrm{N}/2-1} \sum_{m=-\mathrm{N}/2}^{\mathrm{N}/2-1} \hat{u}_n \hat{v}_m + \frac{1 + \mathrm{e}^{\pm \mathrm{i}\mathrm{N}\xi}}{2} \sum\sum_{m+n=k\pm\mathrm{N}} \hat{u}_n \hat{v}_m\,. \tag{87}$$

Hence

$$\mathrm{e}^{\pm \mathrm{i}\mathrm{N}\xi} = -1 \Rightarrow \xi\mathrm{N} = \pm\pi \Rightarrow \xi = \frac{\pi}{\mathrm{N}} = \frac{\Delta x}{2}\,. \tag{88}$$

Thus the algorithm for phase shifting is

1. Perform the pseudo-spectral without shift;
2. Shift the Fourier coefficients $\hat{u}_k$ and $\hat{v}_k$ in Fourier space by multiplying by $\exp(\mathrm{i}\mathrm{N}\Delta x/2)$. Then perform the pseudo-spectral calculation for this shifted coefficients;
3. Take the average of (1) and (2), and march the system forward.

The cost of the phase shifting is $\mathrm{N}\log_2\mathrm{N}$, and therefore keeps the same order of operations as the aliased calculation. It may be advantageous if N is too large, since is does not require more memory than the original problem.

**Zero padding vs. phase shifting in one dimension**

It is informative to compare the relative cost of the two dealiasing methods described above. Suppose the original grid has N elements. Hence zero padding requires computation of DFT on $\mathrm{M} = \frac{3}{2}\mathrm{N}$ grids. Thus one cycle of the pseudo-spectral calculation requires $15\mathrm{M}\log_2\mathrm{M} = \frac{45}{2}\mathrm{N}\log_2\mathrm{N}$ operations. The phase-shifting dealiasing method requires DFT computations on N grid but the pseudo-spectral cycle must be repeated twice (in one dimension). Thus the total number operations required for dealiasing using phase shifting is $30\mathrm{N}\log_2\mathrm{N}$. Thus in one dimension phase shifting requires $\frac{12}{5}$ more operations than zero padding. The latter, however, requires more memory. Thus, the question of the relative efficiency of these two methods may depend on the computer one wants to perform the calculations.

Table 3: Number of aliasing terms for quadratic nonlinearities in pseudo-spectral methods from 1 through three dimensions.

| | 1D | 2D | 3D |
|---|---|---|---|
| single | 1 | 2 | 3 |
| double | 0 | 1 | 3 |
| triple | 0 | 0 | 1 |

### Dealiasing in higher dimensions

Real world problems are typically two- of three-dimensional. There is a qualitative difference between aliasing in one and two dimensions because for dimensions larger than one various combinations between wavenumbers produce aliasing. For instance if the original signal is

$$s = \sin x + \cos y \,, \tag{89}$$

then a quadratic product produces the terms

$$\sin^2 x = \frac{1 - \cos 2x}{2} \,, \tag{90}$$

$$\cos^2 y = \frac{1 + \cos 2y}{2} \,, \tag{91}$$

and also the cross-terms

$$2 \sin x \cos y = \sin 2x \,. \tag{92}$$

Thus aliasing in higher dimensions is richer.

Consider the representation of the discrete Fourier transform of $\hat{u}_{\vec{n}}$

$$\hat{u}_{\vec{n}} = \sum_{j_1=0}^{\mathrm{N}_1-1} \sum_{j_2=0}^{\mathrm{N}_2-1} u_{j_1,j_2} \exp -\mathrm{i}[k_{n_1} x_{j_1} + k_{n_2} x_{j_2}] \equiv \sum_{\vec{j}=0}^{\vec{N}-1} u_{\vec{j}} \, \mathrm{e}^{\mathrm{i} k_{\vec{n}} \cdot \vec{x}_{\vec{j}}} \,, \tag{93}$$

where

$$\vec{n} = (n_1 \,, n_2) \,, \qquad \vec{j} = (j_1 \,, j_2) \,, \qquad \text{and} \qquad \vec{x} = x_{j_1} \vec{e}_1 + x_{j_2} \vec{e}_2 \,, \tag{94}$$

with $\vec{e}_1$ and $\vec{e}_2$ the unit vector of the two-dimensional Cartesian grid.

Analogously to the one-dimensional case, the aliasing error is obtained by analyzing the DFT of the product of two variables $u$ and $v$. We obtain

$$\sum_{\vec{m}+\vec{n}=\vec{p}} \tilde{u}_{\vec{m}} \tilde{v}_{\vec{n}} = \underbrace{\sum_{\vec{m}+\vec{n}=\vec{p}} \hat{u}_{\vec{m}} \hat{v}_{\vec{n}}}_{\text{Truncated series}} + \underbrace{\sum_{\vec{m}+\vec{n}=\vec{p}\pm N_1\vec{e}_1} \hat{u}_{\vec{m}} \hat{v}_{\vec{n}} + \sum_{\vec{m}+\vec{n}=\vec{p}\pm N_2\vec{e}_2} \hat{u}_{\vec{m}} \hat{v}_{\vec{n}}}_{\text{Single alias errors}} + \underbrace{\sum_{\vec{m}+\vec{n}=\vec{p}\pm N_1\vec{e}_1\pm N_2\vec{e}_2} \hat{u}_{\vec{m}} \hat{v}_{\vec{n}}}_{\text{Double alias error}} \,. \tag{95}$$

Table **??** shows the number of aliasing term for pseudo-spectral problems in 1 through 3 dimensions.

### Zero padding in two dimensions

The zero padding approach discussed above in one dimension easily generalizes to higher dimensions. That is, we simply apply the 3/2 rule to each dimension. The memory cost increase is $\left(\frac{3}{2}\right)^p$ where $p$ is the number of dimensions. Thus in higher dimensions zero padding may significantly slow down the code.

Table 4: Phase shift operations to zero out aliasing terms.

| $\xi_1$ | 0 | $\Delta x/2$ | 0 | $\Delta x/2$ |
|---|---|---|---|---|
| $\xi_2$ | 0 | 0 | $\Delta y/2$ | $\Delta y/2$ |
| $\mathrm{AE}_{10}$ | 1 | -1 | 1 | -1 |
| $\mathrm{AE}_{01}$ | 1 | 1 | -1 | -1 |
| $\mathrm{AE}_{11}$ | 1 | -1 | -1 | 1 |

**Phase shifting in two dimensions**

Phase shifting becomes trickier in higher dimensions. While phase shifting in one direction, it introduces error in other directions. Thus a combination of phase shifts are necessary to cancel all errors. Table 4 shows the combination of shifts that zeros out the aliasing terms. In two dimensions, phase shifts requires four evaluations of the pseudo-spectral cycle.

**Hybrid dealiasing schemes**

The idea of the hybrid dealiasing method is to combine a polyhedral truncation with phase shift. The polyhedral truncation is an incomplete zero padding scheme. In particular, one realizes that the corner of the wavenumber domain are insignificant when it comes to studying flows with isotropic statistics. Thus only $|k_{\mathrm{N}_1} \pm k_{\mathrm{N}_2}| < \frac{3}{2}$ is kept; the other wavenumbers are zeroed out. Similarly, in three dimensions we use the spherical truncation $|k^2_{\mathrm{N}_1} + k^2_{\mathrm{N}_2} + k^2_{\mathrm{N}_3}| < \frac{4}{9}\mathrm{N}^2$. This truncation removes double and triple aliasing terms (and somewhat more mores). After this truncation, one only needs to three evaluations of the pseudo-spectral cycle: one in the original grid and two with shifts in each direction to cancel out the single aliasing errors. Similarly, only 4 evaluations are needed in three dimensions.

## Some practical aspects of FFT routines in Fortran

We use FFT in the West (FFTW; for details see www.fftw.org/documentation). The application of the one dimensional transform is straight forward and follows a common intuition. In higher dimensions, however, the output is given in nonstandard form. In particular, we note that, for a real signal we have

$$\begin{aligned} \mathrm{2D} &\rightarrow \mathrm{CFT}\left[\mathrm{CRFT}(u)\right] ; \\ \mathrm{3D} &\rightarrow \mathrm{CFT}\left\{\mathrm{CFT}\left[\mathrm{CRFT}(u)\right]\right\} , \end{aligned} \tag{96}$$

where CFT represents the complex Fourier transform and CRFT represents the complex real Fourier transform. The latter takes advantage of the Hermitian symmetric of the Fourier coefficients to perform the calculation faster and requires less storage.

The are two ways of calling the FFTW routine:

- **Out-of-place**

  The input array is not destroyed. That is, the routine returns an output array different than the input array. For a one dimensional real input with N points, the output has $\frac{\mathrm{N}+1}{2}$ non-redundant elements.

- **In-place**

  The routine overwrites the input with the output, using the same memory allocation. Because the input is real, care has to be taken to interpret the output. In one dimension, if the input goes is distributed as $0, 1, \ldots, \mathrm{N}-2, \mathrm{N}-1$. Then the output is $\mathrm{R}_0$, $\mathrm{R}_1$, ..., $\mathrm{R}_{N-1}$, $\mathrm{I}_{N/2-1}$, ..., $\mathrm{I}_2$, $\mathrm{I}_1$ ,where R stands for the real part and I for the imaginary part of the Fourier coefficients.

Things get more complicated in higher dimensions. So, please consult the manual for an accurate description.

# 5 Working with incompressible equations

The incompressible Navier-Stokes equations are

$$\partial_t \vec{u} + \vec{u} \cdot \nabla \vec{u} = -\frac{\nabla p}{\rho} + \nu \triangle \vec{u}\,, \tag{97}$$

and

$$\nabla \cdot \vec{u} = 0\,. \tag{98}$$

## 5.1 Fractional step method

In this approach, we split every time integration step in two substeps. In the first substep, we perfom a preliminary estimation of the velocity $bu^*$ neglecting the pressure term altogether. Using an implicit Euler scheme for the linear terms and explicit Euler for the nonlinear terms, we obtain

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} - \nu \triangle \boldsymbol{u}^* = \mathbf{N}(\boldsymbol{u}^n)\,, \qquad \text{on the volume} \qquad \Omega\,, \tag{99}$$

with non slip boundary conditions

$$\boldsymbol{u}^* = 0\,, \qquad \text{on the surface} \qquad \partial\Omega\,. \tag{100}$$

Notice that these boundary conditions are necessary in this substeps because it contains the higher derivative viscous terms. The second substep, the projection step, deals with the pressure term

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} + \nabla p^{n+1} = 0\,, \tag{101}$$

where $p$ is the dynamic pressure, and imposes the nondivergence constraint

$$\nabla \cdot \boldsymbol{u}^{n+1} = 0\,, \tag{102}$$

and no-normal flow boundary conditions

$$\boldsymbol{u}^{n+1} \cdot \hat{\mathbf{n}}\,, \qquad \text{on the surface} \qquad \partial\Omega\,. \tag{103}$$

Taking the divergence of the second equation (101) we obtain an elliptic problem for the pressure

$$\triangle p^{n+1} = \frac{\nabla \cdot \boldsymbol{u}^*}{\Delta t}\,, \tag{104}$$

Using the divergence theorem, we obtain

$$\frac{\boldsymbol{u}^{n+1} \cdot \hat{\mathbf{n}}}{\Delta t} + \frac{\partial \phi}{\partial \mathbf{n}} = 0\,, \qquad \text{on the surface} \qquad \partial\Omega\,. \tag{105}$$

That is, we impose no-slip on $\boldsymbol{u}^*$ and no-normal flow on $\boldsymbol{u}^*$ and $\boldsymbol{u}^{n+1}$. There is an $\mathcal{O}(\Delta t)$ error in the no-slip boundary condition on $\boldsymbol{u}^{n+1}$. The final single time-step marching scheme has the form

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \nabla p^{n+1} \Delta t\,. \tag{106}$$

### 5.2 Pressure correction methods

This scheme is also based on two substeps. In the first substep we solve the problem implicitly for pressure

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} - \nu \triangle \boldsymbol{u}^* = \mathbf{N}(\boldsymbol{u}^n) - \nabla p^n \,, \tag{107}$$

with no-slip boundary conditions

$$\boldsymbol{u}^* = 0 \,, \qquad \text{on the surface} \qquad \partial\Omega \,. \tag{108}$$

The second substep then makes a pressure correction

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} + \nabla \cdot \phi = 0 \,, \tag{109}$$

by imposing the nondivengece constraint on $\boldsymbol{u}^{n+1}$, which implies

$$\triangle \phi = \frac{\nabla \cdot \boldsymbol{u}^*}{\Delta t} \,, \qquad \text{on the volume} \qquad \Omega \,, \tag{110}$$

subject to no-slip boundary conditions i.e.,

$$\frac{\partial \phi}{\partial n} = 0 \,, \qquad \text{on the surface} \qquad \partial\Omega \,. \tag{111}$$

Thus the corrected pressure is

$$p^{n+1} = p^n + \phi - \nu \nabla \cdot \boldsymbol{u}^* \,. \tag{112}$$

For this scheme, the no-slip boundary conditions are affected by a $\Delta t^2$ error term.

## 6 Pressure correction with RKW3

Following the notation used above for the RKW3 scheme, the first substep of the pressure correction method has the form

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\gamma_1 \mathsf{H}^n + \frac{\alpha_1}{\mathrm{Re}} \mathsf{L}\boldsymbol{u}^* - (\alpha_1 + \beta_1) \mathsf{G} p^n \,, \tag{113}$$

$$\frac{\boldsymbol{u}^2 - \boldsymbol{u}^*}{\Delta t} = -(\alpha_1 \beta_1) \mathsf{G}\phi \,, \tag{114}$$

$$\mathsf{D} \cdot \boldsymbol{u}^* = 0 \Longrightarrow \mathsf{L}\phi^1 = \frac{\mathsf{D}\boldsymbol{u}^*}{(\alpha_1 + \beta_1)\Delta t} \,, \qquad \text{on the volume} \qquad \Omega \,, \tag{115}$$

and therefore

$$\partial_n \phi' = 0 \,, \qquad \text{on the surface} \qquad \partial\Omega \,. \tag{116}$$

The pressure correction has the form

$$p^2 = p^n + \phi^1 - \frac{1}{\mathrm{Re}} \frac{\beta_1}{\alpha_1 + \beta_1} \mathsf{D} \cdot \boldsymbol{u}^* \,. \tag{117}$$

The second substep is

$$\frac{\boldsymbol{u}^{**} - \boldsymbol{u}^2}{\Delta t} - \gamma_2 \mathsf{H}^2 - \xi_2 \mathsf{H}^n + \frac{\alpha_2}{\mathrm{Re}} \mathsf{L}\boldsymbol{u}^2 + \frac{\beta_2}{\mathrm{Re}} \mathsf{L}\boldsymbol{u}^* * - (\alpha_2 + \beta_2) \mathsf{G} p^2 \,, \tag{118}$$

$$\frac{\boldsymbol{u}^3 - \boldsymbol{u}^{**}}{\Delta t} = -(\alpha_2 + \beta_2) \mathsf{G} p^2 \,, \tag{119}$$

$$\mathsf{D}\cdot\boldsymbol{u}^3=0\Longrightarrow\mathsf{L}\phi^2=\frac{\mathsf{D}u^{**}}{(\alpha_2+\beta_2)\Delta t}\,. \tag{120}$$

The pressure correction in this substep is

$$p^3=p^2+\phi^2-\frac{\beta_2}{(\alpha_2+\beta_2)\mathrm{Re}}\mathsf{D}\cdot\boldsymbol{u}^{**}\,. \tag{121}$$

The third substep is

$$\frac{\boldsymbol{u}^{***}-\boldsymbol{u}^3}{\Delta t}-\gamma_3\mathsf{H}^3-\xi_3\mathsf{H}^2+\frac{\alpha_3}{\mathrm{Re}}\mathsf{L}\boldsymbol{u}^3+\frac{\beta_3}{\mathrm{Re}}\mathsf{L}\boldsymbol{u}^{***}-(\alpha_3+\beta_3)\mathsf{G}p^3\,, \tag{122}$$

$$\frac{\boldsymbol{u}^{n+1}-\boldsymbol{u}^{***}}{\Delta t}=-(\alpha_3+\beta_3)\mathsf{G}\phi^3\,, \tag{123}$$

$$\mathsf{D}\cdot\boldsymbol{u}^3=0\Longrightarrow\mathsf{L}\phi^3=\frac{\mathsf{D}u^{***}}{(\alpha_3+\beta_3)\Delta t}\,. \tag{124}$$

Finally, the pressure at step $n+1$ is

$$p^{n+1}=p^3+\phi^3-\frac{1}{\mathrm{Re}}\frac{\beta_3}{\alpha_3+\beta_3}\mathsf{D}\boldsymbol{u}^{***}\,. \tag{125}$$

# 7 Solving large algebraic systems

As we have seen, working with incompressible Navier-Stokes equations give rise to a Poisson problem to be solve every time step. The discrete version of this problem is a large system of algebraic equations, whose solution is computationally costly. In this section, we explore some of the main iterative approaches to solve such systems.

## 7.1 Relaxation

The problem to solve is

$$\triangle\phi=f\,, \tag{126}$$

where f is a given function of space. In the relaxation method, we add time dependence to the problem

$$\partial_t\phi=\triangle\phi-f\,, \tag{127}$$

and look for the steady solution. Because we are not interested in the transients, we simply step this system forward using an implicit scheme, given an initial guess. Because the true solution is steady, this is equivalent of solving for the residual due to the initial guess

$$\partial_t r=\triangle r\,. \tag{128}$$

In Fourier space we have

$$\hat{r}_k=\hat{r}\mathrm{e}^{-k^2t}\,. \tag{129}$$

Clearly, high wavenumbers $k$ converge faster that low wavenumbers, since the residual of the former decays faster.

### 7.2 Alternate direction implicit

For two-dimensional or three-dimensional problems, we generally simplify the problem by making an operator split

$$[\mathsf{I} - \Delta t(\mathsf{L}_x + \mathsf{L}_y)] = [\mathsf{I} - \Delta t\mathsf{L}_x][\mathsf{I} - \Delta t\mathsf{L}_y] + 2\mathsf{L}_x\mathsf{L}_y\Delta t^2\,, \tag{130}$$

where $\mathsf{L}_x$ and $\mathsf{L}_y$ are the discrete Laplacian operator in each direction. The split above is accurate to second-order in time, which is the accuracy of a Crank-Nicolson scheme. We then solve the problem in two steps

$$[\mathsf{I} - \Delta t\mathsf{L}_x]r^* = r^n\,, \tag{131}$$

and

$$[\mathsf{I} - \Delta t\mathsf{L}_y]r^{n+1} = r^*\,. \tag{132}$$

These two step are much easier to solve than the original problem because they're series of tri-diagonal systems. This ADI scheme can be embedded in the Crank-Nicolson scheme

$$[\mathsf{I} - \tfrac{\Delta t}{2}\mathsf{L}_x]r^* = [\mathsf{I} + \tfrac{\Delta t}{2}\mathsf{L}_y]r^n \tag{133}$$

$$[\mathsf{I} - \tfrac{\Delta t}{2}\mathsf{L}_y]r^{n+1} = [\mathsf{I} + \tfrac{\Delta t}{2}\mathsf{L}_x]r^*\,. \tag{134}$$

Notice that the growth factor at each sub-steps is

$$\sigma_k = \frac{1 - k^2\frac{\Delta t}{2}}{1 + k^2\frac{\Delta t}{2}}\,. \tag{135}$$

Figure (6) shows the absolute values of the growth factor $|\sigma_k|$ as a function of time step for various values of $k\Delta x$. As can be seen, different wavenumbers have distinct optimum time steps. The optimum time step for low wavenumbers is typically very large; the contrary is true for very small wavenumbers:

$$\delta_{max} = \Delta t\left(\frac{\pi}{\Delta x}\right)^2 = \Delta t k_{max}^2\,, \tag{136}$$

$$\delta_{min} = \Delta t\left(\frac{\pi}{\Delta x}\right)^2 = \Delta t k_{min}^2 = \frac{\delta_{max}}{\mathrm{N}^2}\,, \tag{137}$$

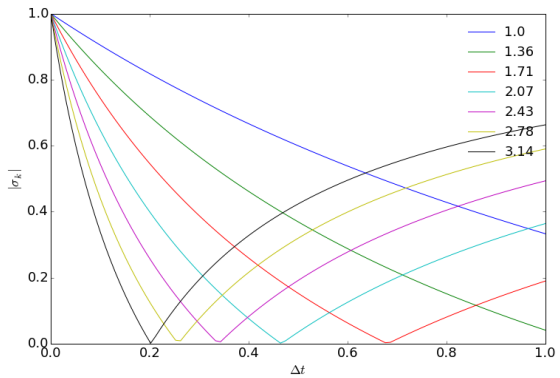


Figure 6: The growth factor of the ADI-CN method for various values of $k\Delta x$.

(SOME SUTFF AND DON'T UNDERSTAND WHERE IT COMES FROM. FOR INSTANCE HOW TO OBTAIN THIS DELTA -8, ETC)

The optimum $\Delta t$ is then

$$\Delta t_{\rm opt} = 2\frac{\Delta x^2}{\pi^2}\mathrm{N} = 2\frac{L_x \Delta x}{\pi^2}\,, \tag{138}$$

which is the geometric mean between the extrema. An estimative of the number of iterations required for convergence within machine precision is

$$\left|\frac{r^{n+1} - r^n}{r^0}\right| = \text{Rel. err.}\,. \tag{139}$$

Thus

$$\log \text{Rel. err.} = \log \text{tol.} = n \log(\epsilon - 1)\,, \tag{140}$$

where

$$\epsilon = 1 - 2\frac{\delta_{max}}{\mathrm{N}^2} = 1 - \frac{4}{\mathrm{N}}\,. \tag{141}$$

We can always sweep randomly through the $\Delta t$ space. In this case, it can be shown that we obtain a faster convergence ($\mathrm{N}^{1/2}$).