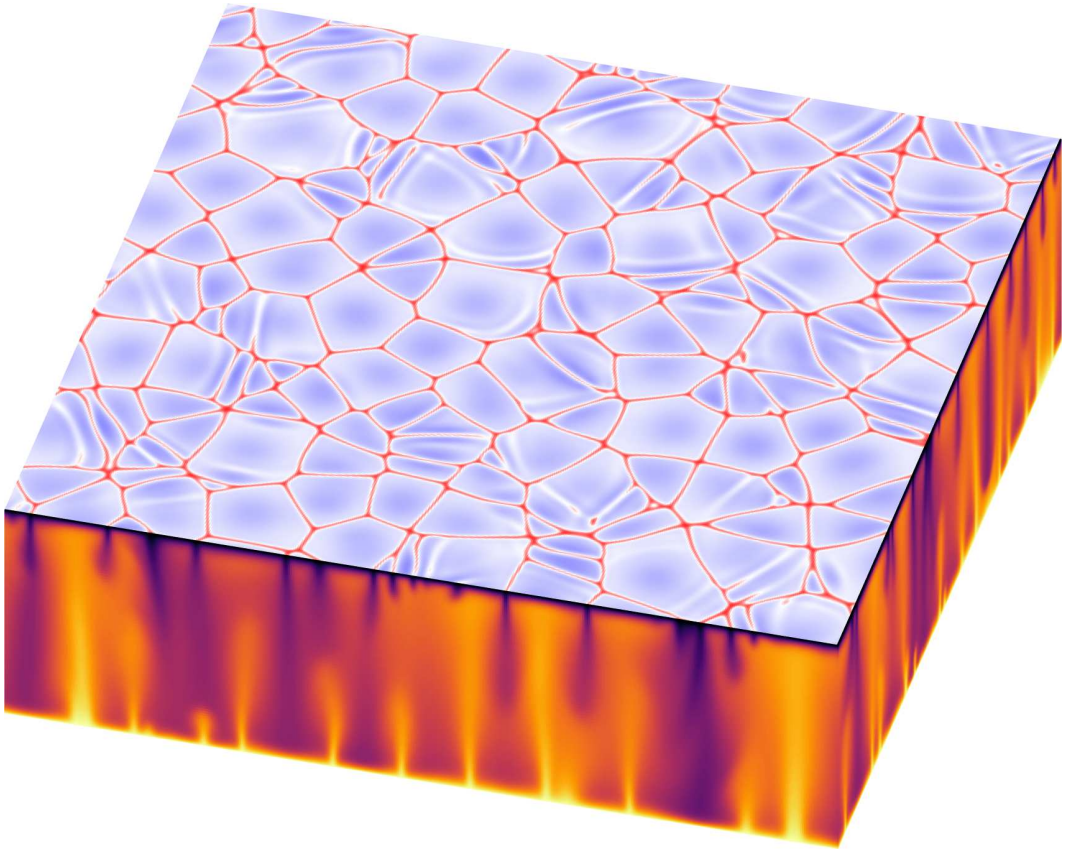
DryLa 6

Spectral element solver
for the sub-surface fluid dynamics of dry salt lakes

August 20, 2024



Developer: Cédric Beaume (University of Leeds)
User: Matthew Threadgold (PhD student, University of Leeds, 2020–2024)

Contents

1 History

This project started at the Dynamics Days Europe 2018 in Loughborough (UK), where the presentation of Dr. Lucas Goehring (Nottingham Trent University) about dry salt lake patterns caught my attention. DryLa 1 was written in the following weeks. It was a spectral solver for the sub-surface fluid dynamics of dry salt lakes that used a Fourier–Galerkin discretization in the horizontal directions and a Gauss–Lobatto–Legendre collocation method in the vertical direction.

DryLa 2 was produced for Matthew Threadgold's PhD project, funded by the Natural Environment Research Council's Doctoral Training Programme: Panorama in 2020. It was mostly an improvement in user-friendliness. Its computation speed and memory efficiency was subsequently improved, leading to DryLa 3 in early 2021.

To circumvent limitations in lake depth created by the collocation method, DryLa was entirely rewritten in 2022 to rest on a spectral element decomposition in the vertical direction. Oh, and the vertical direction was no longer called y but z. DryLa 5 was a major update made in early 2024 focusing on performance scalability and was followed, a few months later, by DryLa 6, further improving scalability and memory management.

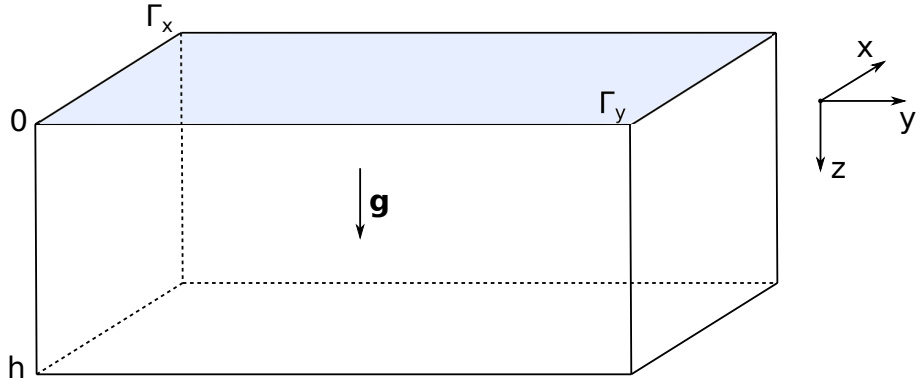


Figure 1: Sketch of the numerical domain modeling the sub-surface region of dry salt lake. The domain is a cuboid supported by the x- and y-axes in the horizontal direction and for which the z-axis is in the direction of gravity, $\mathbf{g}$. Solutions are assumed to be periodic in the x- and y-directions with periods Γ_x and Γ_y respectively. The surface (shaded) is located at $z = 0$ and the bottom of the lake is located at $z = h$.

2 Dry salt lakes equations

We consider the fluid dynamics that takes place below the salty crust of dry lakes. The domain of interest is sketched in Figure 1. It is a cuboid of height h identified by the Cartesian coordinates x and y in the horizontal directions and for which the vertical coordinate, z, increases downward. We assume that solutions are spatially periodic in the horizontal directions, with period Γ_x in x and Γ_y in y.

2.1 Primitive system

The fluid dynamics of interest is dominated by two phenomena: (i) evaporation, which we model using a through flow boundary condition in the vertical direction; and (ii) buoyancy, triggered by a large-scale, upward gradient of salinity accross the domain and modeled by Dirichlet boundary conditions for the salinity. The interior of the lake is considered to be a porous medium so that the resulting non-dimensionalized equations are:

$$\mathbf{u} = -\nabla p + Ra\, S\, \hat{\mathbf{z}}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

$$\partial_t S + (\mathbf{u} \cdot \nabla)\, S = \nabla^2 S, \tag{3}$$

where $\mathbf{u} = u\hat{\mathbf{x}} + v\hat{\mathbf{y}} + w\hat{\mathbf{z}}$ is the velocity, $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the unit vectors in the horizontal directions, $\hat{\mathbf{z}}$ is the unit vector in the descending direction, p is the pressure, Ra is the Rayleigh number, S is the salinity and t is the time. Equation (1) is the Darcy law for a fluid in a porous medium including the buoyancy force. Equation (2) is the incompressibility condition and equation (3) is the standard equation for salinity, modeling both advection and diffusion. A discussion on the non-dimensionalization of these equations is given in [4]. We solve these equations together with periodic boundary conditions in both horizontal directions and with the following boundary condition at the top of the domain:

$$S = -w = 1, \qquad \text{at } z = 0. \tag{4}$$

The boundary condition at $z = h$ is not obvious. Since dry salt lakes can extend to large depths, it might not be practical to design correspondingly deep numerical domains and, even if it were, other geological phenomena would enter the leading order modeling like horizontal groundwater flows. For these reasons, we propose two types of boundary conditions at the bottom of the domain:

$$\text{reflective :} \quad S = w + 1 = 0, \qquad \text{at } z = h, \tag{5}$$

$$\text{penetrative :} \quad \partial_z S = u = v = 0, \qquad \text{at } z = h. \tag{6}$$

2.2 Sequential system

Solving system (1)–(3) is not straightforward due to the fact that we do not possess an equation to determine the pressure. Such an equation can be classically obtained by noticing that the velocity can be eliminated from equation (1) by taking the divergence and using the incompressibility condition. The resulting system reads:

$$\nabla^2 p = Ra\, \partial_z S, \tag{7}$$

$$\mathbf{u} = -\nabla p + Ra\, S\, \hat{\mathbf{z}}, \tag{8}$$

$$\partial_t S + (\mathbf{u} \cdot \nabla)\, S = \nabla^2 S, \tag{9}$$

where the boundary conditions for the pressure are converted from the Darcy law, the boundary condition on the velocity and the incompressibility condition. The top boundary condition for this system reads:

$$S = \partial_z p - Ra = 1, \qquad \text{at } z = 0, \tag{10}$$

Lastly, the bottom boundary condition is choosen between the following:

$$\text{reflective :} \quad S = \partial_z p - 1 = 0, \qquad \text{at } z = h, \tag{11}$$

$$\text{penetrative :} \quad \partial_z S = p = 0, \qquad \text{at } z = h. \tag{12}$$

We refer to this system as *sequential*, as it allows to solve for each state variable (p, $\mathbf{u}$ and S) successively rather than system (1)–(3), in which the coupling forces to solve simultaneously for all the state variables.

3 Temporal discretization

3.1 Underlying principles

We consider the constant time-step discretization of the time:

$$t_n = n \triangle t, \tag{13}$$

where $\triangle t$ is the constant, typically small, time-step, and n is a positive integer. It is useful here to change referential to discuss function discretization. During one time-step, computations are carried out to determine the value of functions at the running time knowing its values at past times. Let us call the running time t and discretize time backwards from it: $t - \triangle t$, $t - 2\triangle t$, $t - 3\triangle t$, ... For simplicity, let us also introduce the following notation for the expression of a funtion f at time $t - n\triangle t$, where n is a positive integer: $f(t - n\triangle t) = f^{t-n\triangle t}$.

To discretize equation (9), we need to express all its terms at the running time:

$$\left(\partial_t S\right)^t + \left[\left(\mathbf{u}\cdot\nabla\right) S\right]^t = \left(\nabla^2 S\right)^t, \tag{14}$$

but we do not know the value of any of these functions at that time. We could treat the Laplacian explicitly by extrapolating its value from known times but this is a typical mistake that would cost the numerical scheme its stability. Instead, it is best kept implicit, as in the above equation. It will form part of the left-hand-side operator for this equation. The nonlinear term cannot be treated implicitly, so we need to extrapolate it from its past values. A similar observation can be made for the time-derivative, which should be expressed by means of a Taylor expansion of S.

3.2 Extrapolation of the nonlinear term

To avoid heavy notation, we introduce the following notation: $N = \left(\mathbf{u}\cdot\nabla\right) S$. We aim to approximate N^t by a linear combination of the known values: $N^{t-\triangle t}$, $N^{t-2\triangle t}$, $N^{t-3\triangle t}$, ... Let us write the Taylor expansions corresponding to these values based on the running time:

$$N^{t-\triangle t} = N^t + \frac{(-1)^1\triangle t}{1!}\frac{\partial N}{\partial t}^t + \frac{(-1)^2\triangle t^2}{2!}\frac{\partial^2 N}{\partial t^2}^t + \frac{(-1)^3\triangle t^3}{3!}\frac{\partial^3 N}{\partial t^3}^t + O(\triangle t^4), \tag{15}$$

$$N^{t-2\triangle t} = N^t + \frac{(-2)^1\triangle t}{1!}\frac{\partial N}{\partial t}^t + \frac{(-2)^2\triangle t^2}{2!}\frac{\partial^2 N}{\partial t^2}^t + \frac{(-2)^3\triangle t^3}{3!}\frac{\partial^3 N}{\partial t^3}^t + O(\triangle t^4), \tag{16}$$

$$N^{t-3\triangle t} = N^t + \frac{(-3)^1\triangle t}{1!}\frac{\partial N}{\partial t}^t + \frac{(-3)^2\triangle t^2}{2!}\frac{\partial^2 N}{\partial t^2}^t + \frac{(-3)^3\triangle t^3}{3!}\frac{\partial^3 N}{\partial t^3}^t + O(\triangle t^4). \tag{17}$$

$$\vdots$$

Obtaining a first-order scheme comes down to rearranging equation (15):

$$N^t = N^{t-\triangle t} + \triangle t\frac{\partial N}{\partial t}^t + O(\triangle t^2), \tag{18}$$

where the error is $O(\triangle t)$. To obtain a second-order scheme, we need to involve two time-steps:

$$\beta_1 N^{t-\triangle t} + \beta_2 N^{t-2\triangle t} = (\beta_1+\beta_2)N^t + \left[\frac{(-1)^1\beta_1 + (-2)^1\beta_2}{1!}\right]\triangle t\frac{\partial N}{\partial t}^t \ldots + \left[\frac{(-1)^2\beta_1 + (-2)^2\beta_2}{2!}\right]\triangle t^2\frac{\partial^2 N}{\partial t^2}^t + O(\triangle t^3), \tag{19}$$

Scheme order	β_1	β_2	β_3
First order	1	0	0
Second order	2	−1	0
Third order	3	−3	1

Table 1: Coefficients of extrapolation for the nonlinear terms as a function of the scheme order up to third order.

which imposes the following conditions on β_1 and β_2:

$$\left.\begin{array}{c} \beta_1+\beta_2=1 \\ \dfrac{(-1)^1\beta_1+(-2)^1\beta_2}{1!}=0 \end{array}\right\} \Rightarrow \left\{\begin{array}{c} \beta_1=2 \\ \beta_2=-1 \end{array}\right. . \tag{20}$$

The resulting approximation is then:

$$N^t = 2N^{t-\triangle t} - N^{t-2\triangle t} + \triangle t^2 \frac{\partial^2 N^t}{\partial t^2} + O(\triangle t^3), \tag{21}$$

where we can see that the term proportional to $\triangle t$ is absent and that the error is $O(\triangle t^2)$.

Any higher-order scheme can be obtained in a similar way, by involving as many terms as the order of the desired scheme. A ν^{th}-order extrapolation scheme takes the form:

$$N^t = \sum_{i=1}^{\nu} \beta_i N^{t-i\triangle t} + O(\triangle t^\nu), \tag{22}$$

and yields the following system for the extrapolation coefficients:

$$\sum_{i=1}^{\nu} \beta_i = 1, \tag{23}$$

$$\sum_{i=1}^{\nu} \frac{i^j}{j!}\beta_i = 0, \qquad j=1,\ldots,\nu. \tag{24}$$

The resulting coefficients, up to a scheme of third-order, are shown in table 1.

3.3 Approximation of the time-derivative

The same method is used to obtain an approximation of the time-derivative as that used to determine the coefficients of the nonlinear term extrapolation. We start by writing the salinity in its Taylor expansion form:

$$S^{t-i\triangle t} = S^t + \frac{(-i)^1\triangle t}{1!}\frac{\partial S^t}{\partial t} + \frac{(-i)^2\triangle t^2}{2!}\frac{\partial^2 S^t}{\partial t^2} + \frac{(-i)^3\triangle t^3}{3!}\frac{\partial^3 S^t}{\partial t^3} + O(\triangle t^4), \tag{25}$$

where i is a positive integer that identifies the time-step. There is a couple of differences between the ways the time-derivative and the nonlinear term are treated. The first one is that we can

Scheme order	$\triangle t\,\alpha_0$	$\triangle t\,\alpha_1$	$\triangle t\,\alpha_2$	$\triangle t\,\alpha_3$
First order	1	-1	0	0
Second order	3/2	-2	1/2	0
Third order	11/6	-3	3/2	$-1/3$

Table 2: Coefficients of extrapolation for the time-derivative as a function of the scheme order up to third order.

afford to keep the implicit term S^t in the time-derivative expansion: it will become part of the left-hand-side operator and improve the scheme stability. The expansion then follows:

$$\frac{\partial S^{\,t}}{\partial t} = \sum_{i=0}^{\nu} \alpha_i S^{t-i\triangle t} + O(\triangle t^{\nu}), \tag{26}$$

where the difference with the extrapolation in equation (22) is that the coefficient index starts at 0, allowing the implicit term to enter the expansion. The second difference is that we approximate the first derivative and not the (non-differentiated) value of the function. With these in mind, the following system for the coefficients can be written:

$$\sum_{i=0}^{\nu} \alpha_i = 0, \tag{27}$$

$$\sum_{i=0}^{\nu} i\,\alpha_i = -\frac{1}{\triangle t}, \tag{28}$$

$$\sum_{i=0}^{\nu} \frac{i^j}{j!}\,\alpha_i = 0, \qquad j = 2, \ldots, \nu. \tag{29}$$

The resulting coefficients are shown in table 2 for discretization schemes up to third-order.

3.4 Summary of the time-discretized system

Using the principles explained in the previous Sections, we can discretize system (7)–(9) into:

$$\left(\alpha_0 - \triangle t\nabla^2\right) S^t = -\sum_{i=1}^{\nu} \alpha_i S^{t-i\triangle t} - \triangle t \sum_{i=1}^{\nu} \beta_i \left[(\mathbf{u}\cdot\nabla)\,S\right]^{t-i\triangle t}, \tag{30}$$

$$\nabla^2 p^t = Ra\,\partial_z S^t, \tag{31}$$

$$\mathbf{u}^t = -\nabla p^t + Ra\,S^t\hat{\mathbf{z}}, \tag{32}$$

where $\left[(\mathbf{u}\cdot\nabla)\,S\right]^{t-i\triangle t} = \left(\mathbf{u}^{t-i\triangle t}\cdot\nabla\right) S^{t-i\triangle t}$ (and not the result of the extrapolation obtained at a previous time-step—another classic mistake) and where the equations are to be solved in this order. Equation (30) consists in combining past values of the salinity and of the nonlinear term into a right-hand-side and applying the inverse of a Helmholtz operator, $\alpha_0 - \triangle t\nabla^2$, augmented by boundary conditions:

$$S^t = 1, \qquad \text{at } z = 0, \tag{33}$$

together with either of the two bottom boundary conditions:

$$\text{reflective}: \quad S^t = 0, \qquad \text{at } z = h, \tag{34}$$

$$\text{penetrative}: \quad \partial_z S^t = 0, \qquad \text{at } z = h. \tag{35}$$

The result, S^t, is then used to compute the right-hand-side of equation (31). The pressure at the running time, p^t, is then solved for by inverting a Laplace operator, ∇^2, provided the boundary conditions:

$$\partial_z p^t = 1 + Ra, \qquad \text{at } z = 0, \tag{36}$$

with one of the following boundary conditions:

$$\text{reflective}: \quad \partial_z p^t = 1, \qquad \text{at } z = h, \tag{37}$$

$$\text{penetrative}: \quad p^t = 0, \qquad \text{at } z = h. \tag{38}$$

Lastly, p^t and S^t are used to "read" the velocity from equation (32).

Although the coefficients obtained in Sections 3.2 and 3.3 stem from simple Taylor expansions, the associated discretization scheme has been named differently by different researchers with names like Backward Differentiation Formula, Adams–Bashforth and Adams–Moulton. The first time I encountered them was in a paper by Karniadakis, Israeli & Orszag [3], where they are attractively named *stiffly-stable scheme coefficients*.

Figure 2: Equidistributed mesh for one-dimensional, periodic functions of period $\Gamma = 4$. The location of the $N = 16$ mesh-points is shown in red.

4 Spatial discretization

4.1 Fourier–Galerkin projection

4.1.1 Principles in one dimension

As we seek periodic solutions in the horizontal directions, it is natural to turn to a Fourier–Galerkin discretization. To understand such a discretization, let us consider a one-dimensional, Γ-periodic function: $f(x)$. This function is discretized on the equidistributed mesh:

$$x_j = \frac{\Gamma j}{N}, \qquad j = 0, \dots, N-1, \tag{39}$$

where N is the number of points and where the last point ($j = N$) is omitted as it is unnecessary: $f(x_N) = f(x_0)$ due to the periodicity of the function. An example of such a mesh is shown in figure 2. Note that to produce good quality plots of any function represented using this type of mesh, one needs to include the x_N data in the output.

We can express $f(x)$ as a discrete Fourier series:

$$f(x) = \sum_{n=-\infty}^{\infty} \hat{f}_n \exp\left(\imath n x \frac{2\pi}{\Gamma}\right), \tag{40}$$

where the $\hat{f}_n$ are the complex coefficients associated with the Fourier expansion and $\imath$ is the imaginary unit number. The above choice of basis implicitly takes care of the periodic boundary condition so that the expansion (40) is, in fact, a Galerkin projection. By using the finite mesh (39), we can write:

$$f(x_j) = \sum_{n=-N/2+1}^{N/2} \left[\hat{f}_n \exp\left(\imath n j \frac{2\pi}{N}\right)\right] \tag{41}$$

$$\Rightarrow f(x_j) = \hat{f}_0 + \sum_{n=1}^{N/2-1} \left[\hat{f}_n \exp\left(\imath n j \frac{2\pi}{N}\right) + \hat{f}_n^* \exp\left(-\imath n j \frac{2\pi}{N}\right)\right] + \hat{f}_{N/2}(-1)^j, \tag{42}$$

where $j = 0, \dots, N-1$ and where the asterisk denotes complex conjugation. Some simplifications were carried out: (i) the $n = 0$ and $n = N/2$ modes taking a trivial form, they have been taken out of the sum; and (ii) the replacement of the "negative" frequency coefficients by the complex conjugate of the corresponding positive frequency is a consequence of the fact that $f(x)$ is a real-valued function ($\hat{f}_{-n} = \hat{f}_n^*$). Lastly, the reason for the limited range of n is explained in Section 4.1.2. The operation that takes the function values in physical space, $f(x_j)$, and returns its values in wavenumber space, $\hat{f}_n$, is called forward Fourier transform:

$$\hat{f}_n = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) \exp\left(-\imath n j \frac{2\pi}{N}\right), \qquad n = 0, \dots, N/2, \tag{43}$$

while the reverse operation, expressed in equation (42), is called backward Fourier transform. Note that the normalization by N, here in expression (43), can either be carried out in the forward or in the backward Fourier transform. In DryLa, the above Fourier transforms are carried out efficiently by an algorithm called fast Fourier transform (FFT) and implemented via the use of the library FFTW [2].

4.1.2 Danger: Aliasing

The range of n in equation (42) is better understood by examining the way modes are represented. First of all, modes with wavenumber above $n \geqslant N$ do not carry any information as our choice of mesh-grid makes them project exactly onto modes with wavenumber $m < N$, where $n \equiv m (\mathrm{mod}\, N)$:

$$\exp\left(\imath(m+N)j\frac{2\pi}{N}\right) = \exp\left(\imath mj\frac{2\pi}{N}\right). \tag{44}$$

As a result, all the information is contained in wavenumbers $0 \leqslant n < N$.

The second remark is less obvious. Let us have a look at a discretized mode of wavenumber $N/2+m$, with $m < N/2$:

$$\begin{aligned}
\exp\left[\imath\left(\frac{N}{2}+m\right)j\frac{2\pi}{N}\right] &= (-1)^j \exp\left(\imath mj\frac{2\pi}{N}\right) && (45)\\
&= \left[(-1)^j \exp\left(-\imath mj\frac{2\pi}{N}\right)\right]^* && (46)\\
&= \left\{\exp\left[\imath\left(\frac{N}{2}-m\right)j\frac{2\pi}{N}\right]\right\}^*. && (47)
\end{aligned}$$

This introduces a special wavenumber called *Nyqvist frequency*: $n = N/2$. This symmetry around this wavenumber is responsible for a phenomenon called *frequency folding*, whereby modes beyond the Nyqvist frequency fold back onto modes of lower wavenumber: their coefficients appear complex conjugated on the mode of wavenumber opposite and at the same distance from the Nyqvist frequency.

As a result from the above remarks, modes with wavenumber higher than the Nyqvist frequency are incorrectly represented. This phenomenon is called *aliasing* and is exemplified in Figure 3 for two functions. While the continuous representations of $\cos(9x)$ and $\cos(7x)$ look drastically different, their values coincide on the 16 equidistributed meshpoints between 0 and 2π (not counting in the point at 2π). This results from the fact that, given a 16 point mesh, the Nyqvist frequency is $n = 8$ and, thus, that wavenumber $n = 9$ folds back onto $n = 7$. A similar observation can be made with $\sin(9x)$, with the exception that a change of sign needs to be applied during frequency folding, owing to the complex conjugation of the coefficients, as shown above.

4.1.3 Information storage

The discretized function $f(x_j)$ contains N real values. This is the amount of information available on the collocation points. The Fourier coefficients are complex, so the storage of the same amount of information necessitates $N/2$ coefficients, corresponding to wavenumbers 0 to $N/2-1$. However, the first coefficient ($n = 0$) corresponds to the constant mode: $\cos(0x) + \imath \sin(0x) = 1$. Since f is real-valued, the imaginary part of the associated coefficient contains no information,

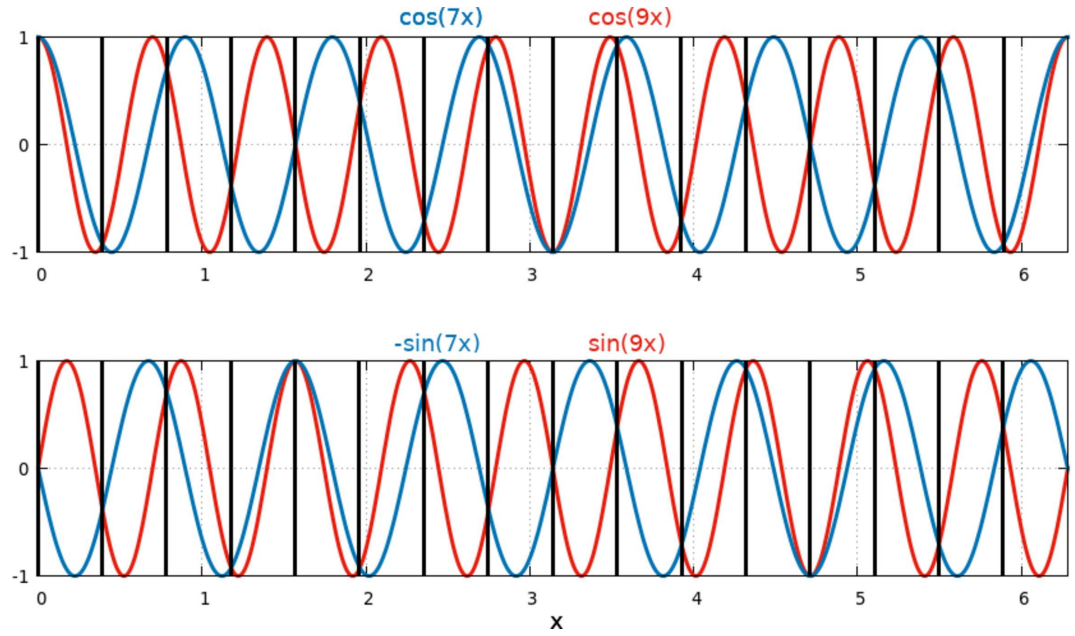


Figure 3: Illustration of the phenomenon of aliasing on a $\Gamma = 2\pi$-periodic domain in x meshed with $N = 16$ equidistributed points given by equation (39) and whose location is shown by the thick vertical lines. The Nyqvist frequency associated with this mesh is $n_{Nyq} = N/2 = 8$. The top (resp. bottom) figure shows that $\cos(9x)$ (resp. $\sin(9x)$) coincides with $\cos(7x)$ (resp. $-\sin(7x)$) onto the mesh-points.

so we need to find one more real-valued piece of information. The Nyqvist mode is the next one in order of growing wavenumber. It takes the form:

$$\exp\left(\imath\frac{N}{2}j\frac{2\pi}{N}\right) = \exp\left(\imath j\pi\right) \tag{48}$$

$$= (-1)^j. \tag{49}$$

It is real-valued and, therefore, completes the representation of the function.

In practice, both the coefficients of the constant mode and of the Nyqvist frequency modes are stored as complex numbers but their imaginary part does not enter any calculation. Fourier coefficients information storage is illustrated in Figure 4 for a function discretized over $N = 16$ mesh-points. The physical representation of the function contains 16 real-valued pieces of information. When the Fourier transform of the function is taken, a set of 9 complex coefficients are returned, corresponding to wavenumber 0 to 8. However, the imaginary part of the coefficient of the mode of wavenumber 0 and that of the coefficient of the mode of wavenumber 8 do not contain any information. The Fourier coefficient then contain $1 + 2 \times 7 + 1 = 16$ real-valued pieces of information, as much as the collocation representation of the function.

4.1.4 Prevention: De-aliasing

Functions are decomposed into a linear combination of exponentials, as in expression (42). Multiplying two discretized functions by one another yields exponential products. For example:

$$\exp\left(\imath nx\frac{2\pi}{\Gamma}\right)\exp\left(\imath mx\frac{2\pi}{\Gamma}\right) = \exp\left(\imath(n+m)x\frac{2\pi}{\Gamma}\right), \tag{50}$$

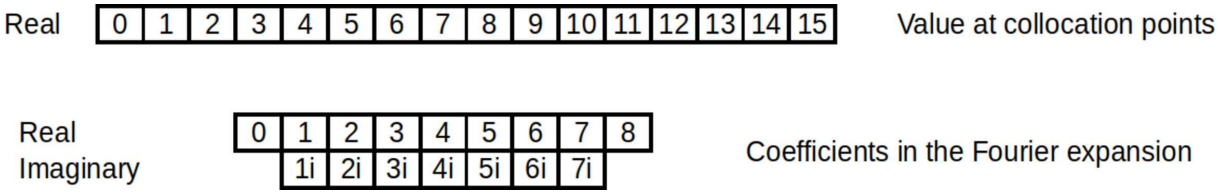


Figure 4: Top: Representation of the information contained in a real-valued, discretized function over $N = 16$ collocation points, labeled 0 to 15. The information contained in physical space is real-valued. Bottom: Representation of the information of the same function via its Fourier coefficients. All coefficients are complex but only the real and imaginary parts that contain information are represented. The numerical lable indicates the wavenumber and the suffix "i" indicates the imaginary part of the coefficient.

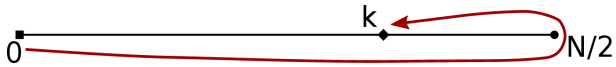


Figure 5: Graphical representation of the frequency folding phenomenon. In order of ascending wavenumber, modes project onto themselves until the Nyqvist frequency, $N/2$, then spuriously down until wavenumber 0 before going back up again in a periodic process. The idea behind dealiasing via Fourier filtering is to filter out modes with wavenumber larger than k in such a way that when the nonlinearity is applied to a mode of wavenumber k, frequency folding occurs but does not reach back to k, as shown by the dark red arrow.

where the wavenumbers n and m are integers. The result is a function whose wavenumber is the sum of the wavenumbers of the initial functions. By extension, we can easily see that a nonlinearity of order η takes energy from a mode of wavenumber k and redistributes it into a mode of wavenumber ηk. If our mesh does not possess a sufficiently large number of points to accommodate such a function, we will end up with aliasing: the energy contained in that function will spuriously be redistributed to other, lower wavenumber modes.

One way to prevent aliasing is to use a Fourier filter, which is a simple on/off low-pass filter that replaces the Fourier coefficients of modes whose wavenumber is larger than a cutoff value by 0. To calculate this cutoff value, we needs to make use of our understanding of frequency folding. We want the application of the nonlinearity onto our largest accepted wavenumber mode to fold back as close as possible but not onto itself, as illustrated in Figure 5. This can be put in equation as follows:

$$\eta k < \frac{N}{2} + \left(\frac{N}{2} - k \right), \tag{51}$$

where the left-hand-side is the wavenumber resulting from the application of the nonlinearity of order η and where the right-hand-side is the "distance" traveled on the line of wavenumbers from 0 to the folding point $(N/2)$ and back to the original wavenumber $N/2 - k$. It follows that

$$k < \frac{N}{\eta + 1} \tag{52}$$

identifies all the wavenumbers k that can remain unfiltered.

In our case, the nonlinearity is of second order, so we have $k < N/3$, filtering out about 1/3 of the modes. To take a concrete example, let us consider the case shown in Figure 4, whereby a function is evaluated on $N = 16$ mesh-points. This function is characterized numerically by 16 pieces of real-valued information, whether it is considered in its collocation or its Fourier coefficient form. Using the Fourier filter associated with the quadratic nonlinearity, we should

only retain modes associated with wavenumbers $k < N/3 \approx 5.33$. We then filter out the modes associated with wavenumbers 6, 7 and 8, which corresponds to 5 real-valued pieces of information (remember that wavenumber 8 is the Nyqvist frequency and only contain one piece of real-valued information), thereby losing 31.25% of the information initially contained by the function.

4.1.5 Extension to two dimensions

The extension of the previous Sections to two-dimensional domains is relatively straightforward. For a domain of periodicity Γ_x in the x-direction and Γ_y in the y-direction, the equidistributed mesh is defined as follows:

$$x_j = \frac{\Gamma_x\, j}{N}, \qquad j = 0, \dots, N-1, \tag{53}$$

$$y_k = \frac{\Gamma_y\, k}{M}, \qquad k = 0, \dots, M-1, \tag{54}$$

where N (resp. M) is the number of points in the x-(resp. y-)direction. The discretization then becomes:

$$\begin{aligned} f(x_j, y_k) = \sum_{m=-M/2+1}^{M/2} \exp\left(\imath m k \frac{2\pi}{M}\right) \Bigg\{ \hat{f}_{0,m} \ \dots \\ + \sum_{n=1}^{N/2-1} \left[\hat{f}_{n,m} \exp\left(\imath n j \frac{2\pi}{N}\right) + \hat{f}^*_{n,m} \exp\left(-\imath n j \frac{2\pi}{N}\right) \right] + \hat{f}_{N/2,m} (-1)^j \Bigg\}, \end{aligned} \tag{55}$$

where $j = 0, \dots, N-1$ and $k = 0, \dots, M-1$. Expression (55) can be thought of as simply a Fourier transform in the y-direction applied *after* the Fourier transform in the x-direction and given in expression (42). Here, the coefficient symmetry due to the function $f(x, y)$ being real has been used on the index n and m is allowed to take positive and negative values and $m = -M/2$ has been discarded as it is the negative Nyqvist frequency associated with the y-direction and would project exactly onto the (positive) Nyqvist frequency $m = M/2$. The associated forward transform is, by extension of expression (43):

$$\hat{f}_{n,m} = \frac{1}{NM} \sum_{j=0}^{N-1} \sum_{k=0}^{M-1} f(x_j, y_k) \exp\left(-\imath n j \frac{2\pi}{N}\right) \exp\left(-\imath m k \frac{2\pi}{M}\right), \tag{56}$$

where $n = 0, \dots, N/2$ and $m = -M/2+1, \dots, M/2$.

The storage of the Fourier coefficients is a little more complicated than in one dimension. As shown in equation (55), the information is stored only for positive wavenumbers n in the x-direction but for both positive and negative wavenumbers m in the y-direction. All coefficients contain complex-valued information, except for four modes: $(n, m) = (0, 0), (N/2, 0), (0, M/2)$ and $(N/2, M/2)$, whose imaginary part does not contain any information, either because they are responsible for imaginary contributions to the physical space representation of the function, or because the associated modes project trivially onto the mesh-points. Lastly, the coefficients of the modes (n, m) and $(n, -m)$ are related by complex conjugation for $n = 0$ and $n = N/2$ so that the resulting mode combination yields a real expression in physical space. This storage scheme is illustrated in Figure 6 for a mesh-grid of $N = 16$ and $M = 16$ points. The physical form of the function (left panel) shows 256 real-valued pieces of information, one at each collocation point. In wavenumber space (right panel), the structure of the information is less trivial. Each value

y \ x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
1	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
2	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
3	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
4	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
5	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
6	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
7	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
8	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
9	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
10	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
11	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
12	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
13	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
14	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
15	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Value at collocation points

m \ n	0	1	2	3	4	5	6	7	8
0	R	C	C	C	C	C	C	C	R
1	C	C	C	C	C	C	C	C	C
2	C	C	C	C	C	C	C	C	C
3	C	C	C	C	C	C	C	C	C
4	C	C	C	C	C	C	C	C	C
5	C	C	C	C	C	C	C	C	C
6	C	C	C	C	C	C	C	C	C
7	C	C	C	C	C	C	C	C	C
8	R	C	C	C	C	C	C	C	R
-7	C	C	C	C	C	C	C	C	C
-6	C	C	C	C	C	C	C	C	C
-5	C	C	C	C	C	C	C	C	C
-4	C	C	C	C	C	C	C	C	C
-3	C	C	C	C	C	C	C	C	C
-2	C	C	C	C	C	C	C	C	C
-1	C	C	C	C	C	C	C	C	C

Coefficients in the Fourier expansion

Figure 6: Left: Representation of the information contained in a two-dimensional, real-valued, discretized function over $N = 16$ collocation points in x and $M = 16$ collocation points in y, each labelled from 0 to 15. Right: Representation of the same function in wavenumber space after the application of the Fourier transform. Wavenumbers 0 to 8 in the x-direction and -7 to 8 in the y-direction are stored in complex form but some of these coefficients do not contain as much information as they seem: while most coefficients are stored complex (denoted C), some only contain information in their real part and are highlighted in blue. The modes $(n = 0, m)$ and $(n = 0, -m)$, highlighted in pink, as well as $(n = 8, m)$ and $(n = 8, -m)$, highlighted in orange, are related by complex conjugation, implying that they contain only half as much information as their memory usage could contain.

of n between 1 and 7 included contains complex coefficients which maximize the information contained for the allocated memory: 16 complex-valued pieces of information or, equivalently 32 real-valued pieces of information. The modes for $n = 0$ or $n = 8$ contain either real-valued coefficients or complex coefficients linked to another by complex conjugation. Each of these thus only contains the equivalent of one real-valued piece of information, implying that, in total, the $n = 0$ and $n = 8$ modes contain each 16 real-valued pieces of information. The total amount of information contained in the Fourier coefficients is, in order of increasing n: $16 + 7 \times 32 + 16 = 256$, guaranteeing that the same amount of information is contained in wavenumber space as in physical space.

Lastly, a Fourier filter has to be applied in each directions to prevent aliasing. This is done straightforwardly by taking one direction at a time. For example, we consider the function represented in Figure 6. On an $N = 16$-point mesh in x, a quadratic nonlinearity will only allow modes up to $n = 5$ to lead to non-aliased calculations (see Section 4.1.4 for a full explanation). We filter out all the information contained in wavenumbers 6, 7 and 8, resulting in the loss of 80 real-valued pieces of information. As we also have $M = 16$ points in the y-direction, we will also only accept modes with wavenumbers up to $m = 5$. Of the information that survived the x-filter, we remove the information contained in the lines where $m = \pm 6$, $m = \pm 7$ and $m = 8$, or 55 more real-valued pieces of information. We have thus filtered out 135 real-valued pieces of information, or approximately 52.7% of the information that the memory could contain.

4.1.6 Example: Helmholtz equation

Let us consider the simple Helmholtz equation:

$$\left(k_1 + k_2\nabla^2\right) u(x,y) = f(x,y), \tag{57}$$

where u is the solution, f is a prescribed right-hand-side and k_1 and k_2 are constants defining the Helmholtz operator. We consider that u is periodic in x with period Γ_x and in y with period Γ_y.

Looking at the two-dimensional discrete Fourier transform:

$$u(x,y) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} \hat{u}_{i,j} \exp\left(\imath i x \frac{2\pi}{\Gamma_x}\right) \exp\left(\imath j y \frac{2\pi}{\Gamma_y}\right), \tag{58}$$

where the circumflex accent denotes the Fourier transform in the x- and y-directions, we see that:

$$\widehat{\partial_x u_i} = \imath i \frac{2\pi}{\Gamma_x}\,\widehat{u}, \tag{59}$$

$$\widehat{\partial_y u_j} = \imath j \frac{2\pi}{\Gamma_y}\,\widehat{u}, \tag{60}$$

$$\tag{61}$$

in such a way that:

$$\widehat{\nabla^2 u_{i,j}} = -4\pi^2\left(\frac{i^2}{\Gamma_x^2} + \frac{j^2}{\Gamma_y^2}\right)\widehat{u}. \tag{62}$$

As a result, using the forward Fourier transform in the x- and y-directions on equation (57), we get:

$$\left[k_1 - 4\pi^2 k_2\left(\frac{i^2}{\Gamma_x^2} + \frac{j^2}{\Gamma_y^2}\right)\right]\hat{u}_{i,j} = \hat{f}_{i,j}. \tag{63}$$

We can see that the use of the Fourier transform has decoupled the x and y components of the differential equation and the operator is now just a scalar quantity for a given couple (i,j). The fact that the operator is *diagonalized* (better seen in one dimension) by the use of the Fourier transform is a major advantage: Fourier transforming a one-dimensional field typically costs $N \times \ln(N)$ operations, where N is the number of points whereas other transforms might involve a matrix-vector multiplication, which costs $N \times N$ operations.

4.2 Gauss–Lobatto–Legendre collocation

To discretize the equations in the vertical direction, I made the choice to use a collocation method. The idea behind these methods is to write down the equations at the location of the mesh-points and to determine an approximate solution that minimizes the error there.

4.2.1 Gaussian quadrature

Collocation methods are often used in conjunction with a Gaussian quadrature, which enables to compute integrals with great accuracy. For a suitable choice of quadrature points z_k and

weights w_k, where $k = 0, \dots, N$,

$$\int_{z_{min}}^{z_{max}} \omega(z) f(z) dz = \sum_{k=0}^{N} w_k f(z_k), \tag{64}$$

where $\omega(z)$ is a weight function, is exact when f is a polynomial of degree $2N-1$ on $[z_{min}, z_{max}]$. The advantage of the Gaussian quadrature is its unusually high order (it is not "just" of order n, as one would expect).

The most basic Gaussian quadrature is a rule on integration and does not include the boundary points: $z = z_{min}$ and $z = z_{max}$. These are necessary in our case, as they will allow to apply the boundary conditions. Fortunately, these points can be included into the quadrature, in which case it is called a Gauss–Lobatto quadrature. If we further set $\omega(z) = 1$ (a good choice for functions that are not singular at the boundary), we obtain the *Gauss–Lobatto–Legendre quadrature* [1], which is the one we will be using.

The Gauss–Lobatto–Legendre quadrature is associated with the following mesh-points:

$$z_0 = z_{min}, \tag{65}$$

$$z_k = z_{min} + \frac{z_{max} - z_{min}}{2} (1 + \zeta_k), \qquad k = 1, \dots, N-1, \tag{66}$$

$$z_N = z_{max}, \tag{67}$$

where ζ_k is the k^{th} root of the first derivative of $L_N(z)$, which is the N^{th} Legendre polynomial, and where:

$$w_k = \frac{z_{max} - z_{min}}{N(N+1) L_N(z_k)^2}, \qquad k = 0, \dots, N, \tag{68}$$

are the quadrature weights. There exists a number of ways to generate the Legendre polynomials, from explicit definitions to solutions of differential equations but, perhaps, the more numerically friendly is the following recurrence relation:

$$L_0(z) = 1, \tag{69}$$

$$L_1(z) = z, \tag{70}$$

$$L_{k+1}(z) = = \frac{2k+1}{k+1} z L_k(z) - \frac{k}{k+1} L_{k-1}(z), \qquad k = 1, 2, \dots. \tag{71}$$

An example of a Gauss–Lobatto–Legendre mesh is shown in Figure 7 for a domain spanning $[0, 1]$ meshed with 17 points. We can clearly see a key characteristics of these points: they are much denser at the edges of the domain, where the point-spacing scales like N^{-2} and much more sparsely distributed around the center, where the point-spacing scales like N^{-1}. Furthermore, the weights associated with these points are larger where the point distribution is sparser to reflect the increased importance of the mesh-points in more sparsely meshed areas.

4.2.2 Differential operator

Let us consider the simple Helmholtz equation:

$$\left(k_1 + k_2 \partial_z^2\right) u(z) = f(z), \tag{72}$$

where u is the solution, f is a given right-hand-side and k_1 and k_2 parameterize the Helmholtz operator. Let us also prepare the collocation method by expanding functions in terms of Lagrange polynomials:

$$u(z) = \sum_{k=0}^{N} u_k \psi_k(z), \tag{73}$$

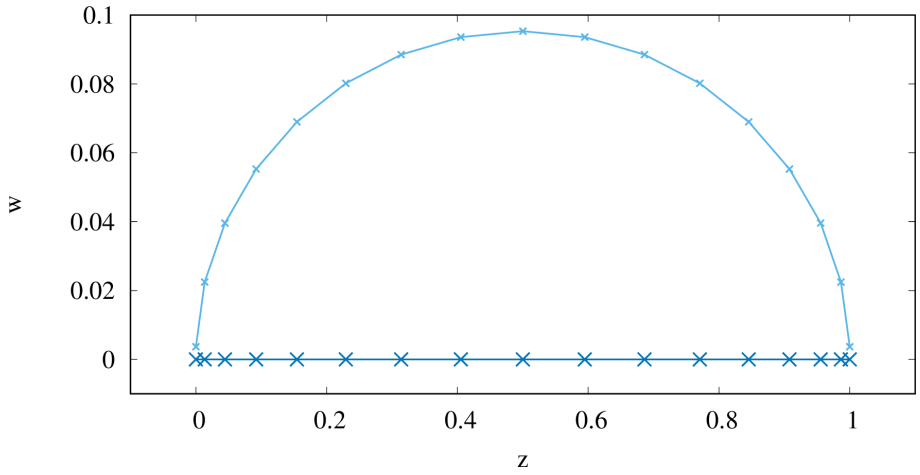


Figure 7: Representation of a Gauss–Lobatto–Legendre discretization of the interval $[0, 1]$ using 17 points. The location of the points is shown along the horizontal axis by the large, dark-blue crosses while the weight w associated with these points is shown by the light-blue curve.

where $\psi_k(z)$ is the k^{th} Lagrange polynomial associated with the Gauss–Lobatto–Legendre mesh-grid defined in equations (65)–(67):

$$\psi_k(z) = \prod_{\substack{l=0 \\ l \neq k}}^{N} \frac{z - z_l}{z_k - z_l}. \tag{74}$$

The advantage of using Lagrange polynomials is that they possess the Kronecker property: $\psi_k(z_m) = \delta_{km}$, where δ is the Kronecker. This property is in line with the spirit of collocation methods, since it implies that $u(z_k) = u_k$.

To recast equation (72) into an integral and use the Gauss–Lobatto–Legendre quadrature explained in the Section 4.2.1, we first multiply it by a set of test functions which, for simplicity, we take to be the Lagrange polynomials. This preliminary step has two purposes: it allows to obtain as many equations as there are mesh-points (one per Lagrange polynomial), and it provides an way to implement boundary conditions explicitly, as we shall see. The choice of Lagrange polynomial, here, is, again, motivated by the Kronecker property. Equation (72) becomes:

$$\int_{z_{min}}^{z_{max}} k_1\, u(z)\, \psi_l(z)\, dz + \int_{z_{min}}^{z_{max}} k_2\, \psi_l(z)\, \partial_z^2 u(z)\, dz = \int_{z_{min}}^{z_{max}} f(z)\, \psi_l(z)\, dz, \qquad l = 0, \ldots, N, \tag{75}$$

which, upon integration by part, generates a boundary term, which is nothing else but the Neumann boundary condition:

$$\begin{aligned} \int_{z_{min}}^{z_{max}} k_1\, u(z)\, \psi_l(z)\, dz + \Big[k_2\, \psi_l(z)\, \partial_z u(z) \Big]_{z_{min}}^{z_{max}} \quad - \quad & \int_{z_{min}}^{z_{max}} k_2\, \partial_z \psi_l(z)\, \partial_z u(z) dz \\ = \quad & \int_{z_{min}}^{z_{max}} f(z)\, \psi_l(z)\, dz, \quad l = 0, \ldots, N. \end{aligned} \tag{76}$$

Now, we discretize the functions u and f according to equation (73) and without touching the

boundary term:

$$\begin{aligned}\int_{z_{min}}^{z_{max}} \sum_{k=0}^{N} k_1\, u_k\, \psi_k(z)\, \psi_l(z)\, dz \quad &+ \quad \Big[k_2\, \psi_l(z)\, \partial_z u(z)\Big]_{z_{min}}^{z_{max}} \dots \\ &- \quad \int_{z_{min}}^{z_{max}} \sum_{k=0}^{N} k_2\, u_k\, \partial_z\psi_l(z)\, \partial_z\psi_k(z)\, dz \dots \\ &= \quad \int_{z_{min}}^{z_{max}} \sum_{k=0}^{N} f_k\, \psi_k(z)\, \psi_l(z)\, dz, \qquad l = 0, \dots, N, \end{aligned} \tag{77}$$

where we have made use of the fact that the coefficients u_k and f_k do not depend on z. We can now express the above equation at the Gauss–Lobatto–Legendre mesh-points and apply the quadrature rule for all integrals:

$$\begin{aligned}\sum_{j=0}^{N} \left(w_j \sum_{k=0}^{N} k_1\, u_k\, \psi_k(z_j)\, \psi_l(z_j) \right) \quad &+ \quad \Big[k_2\, \psi_l(z_j)\, \partial_z u(z_j)\Big]_{j=0}^{j=N} \dots \\ &- \quad \sum_{j=0}^{N} \left(w_j \sum_{k=0}^{N} k_2\, u_k\, \partial_z\psi_l(z_j)\, \partial_z\psi_k(z_j) \right) \dots \\ &= \quad \sum_{j=0}^{N} \left(w_j \sum_{k=0}^{N} f_k\, \psi_k(z_j)\, \psi_l(z_j) \right), \qquad l = 0, \dots, N, \end{aligned} \tag{78}$$

which, thanks to the Kronecker property of the Lagrange polynomials, simplifies into:

$$\begin{aligned}\sum_{j=0}^{N} \sum_{k=0}^{N} k_1\, w_j\, u_k\, \delta_{jk}\, \delta_{jl} \quad &+ \quad k_2\, \delta_{Nl}\, \partial_z u(z_N) - k_2\, \delta_{0l}\, \partial_z u(z_0) \dots \\ &- \quad \sum_{j=0}^{N} \sum_{k=0}^{N} k_2\, w_j\, u_k\, \partial_z\psi_l(z_j)\, \partial_z\psi_k(z_j) \dots \\ &= \quad \sum_{j=0}^{N} \sum_{k=0}^{N} w_j\, f_k\, \delta_{jk}\, \delta_{jl}, \qquad l = 0, \dots, N. \end{aligned} \tag{79}$$

This expression is much simpler than it seems. First of all, the first double sum involves two Kronecker operators and reduces down to a single term, much like the right-hand-side. Furthermore, we note that the boundary terms are only present for equations $l = 0$ and $l = N$. Given that the boundary conditions are known, it is convenient to send the related terms to the right-hand-side. The resulting system reads:

$$\begin{aligned} k_1 u_l \quad - \quad \sum_{j=0}^{N} \sum_{k=0}^{N} k_2\, \frac{w_j}{w_l} u_k\, \partial_z\psi_l(z_j)\, \partial_z\psi_k(z_j) = f_l \dots \\ + \frac{k_2}{w_l} \partial_z u(z_0) \delta_{0l} - \frac{k_2}{w_l} \partial_z u(z_N) \delta_{Nl}, \qquad l = 0, \dots, N, \end{aligned} \tag{80}$$

where we have divided by w_l.

Equation (80) features the derivative of the Lagrange polynomials at the Gauss–Lobatto–

Legendre mesh-points. Fortunately, it can easily be computed:

$$D_{kl} = \begin{cases} \dfrac{2L_N(z_k)}{(z_{max}-z_{min})(z_k-z_l)L_N(z_l)} & k \neq l \\ \dfrac{N(N+1)}{2(z_{max}-z_{min})} & k = l = 0 \\ -\dfrac{N(N+1)}{2(z_{max}-z_{min})} & k = l = N \\ 0 & \text{otherwise} \end{cases}, \tag{81}$$

where $D_{kl} = \partial_z \psi_l(z_k)$ in such a way that

$$\partial_z u(z_k) = \sum_{l=0}^{N} D_{kl} u_l, \tag{82}$$

is spectrally accurate. Equation (80) can then be written in matrix form:

$$\mathcal{M}_{lk} u_k = f_l + \mathcal{B}_l, \tag{83}$$

where

$$\mathcal{M}_{lk} = k_1 - k_2 \sum_{j=0}^{N} \frac{w_j}{w_l} D_{jl} D_{jk}, \tag{84}$$

$$\mathcal{B}_l = \frac{k_2}{w_l} \partial_z u(z_0) \delta_{0l} - \frac{k_2}{w_l} \partial_z u(z_N) \delta_{Nl}, \tag{85}$$

for $l = 0, \ldots, N$ and $k = 0, \ldots, N$.

In the case we want to impose Dirichlet boundary conditions (u_0 and u_N are constants), we do no longer need equations $l = 0$ and $l = N$, which removes the contribution of the Neumann boundary condition term $\mathcal{B}_l$ from the right-hand-side. The left-hand-side of the remaining equations are split into a boundary contribution and the contribution of the inner part of the domain. Since the boundary contributions are known, they are sent to the right-hand-side and the system reduces to:

$$\mathcal{M}_{lk} u_k = f_l - \mathcal{M}_{l0} u_0 - \mathcal{M}_{lN} u_N, \tag{86}$$

for $l = 1, \ldots, N-1$ and $k = 1, \ldots, N-1$.

4.3 Spectral element domain decomposition

The discretization detailed in Section 4.2 is ideal for domains are relatively small extent in the vertical direction. However, the lake might display a surface solutal boundary layer of small extent compared to the vertical diffusive scale. The problem caused by this scale difference cannot be entirely addressed by the simple collocation method: increasing the number of points would result in a larger meshing differences between the edges and the center of the domain, which can undermine numerical stability and yield wasteful computations (the bottom of the lake, where the dynamics is mostly diffusive, would end up massively over-meshed).

To overcome this difficulty, we decompose the domain in elements in the vertical direction, each element being discretized using the Gauss–Lobatto–Legendre discretization of Section 4.2. An

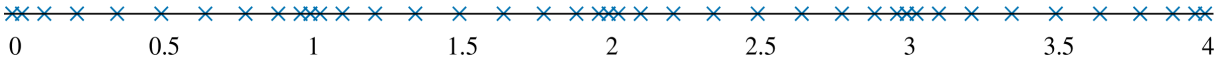


Figure 8: Representation of an Gauss–Lobatto–Legendre element discretization of the interval $[0, 4]$ using 4 identical elements of 11 points each.

example of such a spectral element mesh-grid on $[0, 4]$ is shown in Figure 8 for 4 elements discretized by 11 Gauss–Lobatto–Legendre points each. The structure of each element is the same, which means that the weights w and the differentiation matrix D are the same for each element. One key subtelty of this discretization is that the interface between two elements is discretized using two mesh-points: the last point of the first element and the first point of the next one.

4.3.1 Operator structure

To understand the operator generated by the spectral element method, let us go back to a simple Helmholtz problem to make some preliminary observations. Firstly, the elements are all identical and are discretized using Gauss–Lobatto–Legendre points, so they yield a block-diagonal operator matrix, each block corresponding to the element operator as defined in equation (83). As a first, naive step, we can write the resulting discretized Helmholtz problem as:

$$\left(\begin{array}{c|c|c|c|c} \mathcal{M}_1 & 0 & 0 & \dots & \\ \hline 0 & \mathcal{M}_2 & 0 & \dots & \\ \hline 0 & 0 & \mathcal{M}_3 & & \\ \hline \vdots & \vdots & & \ddots & \\ \hline & & & & \mathcal{M}_{\mathcal{N}} \end{array}\right) \left(\begin{array}{c} u_1^{top} \\ u_1^R \\ u_1^{btm} \\ \hline u_2^{top} \\ u_2^R \\ u_2^{btm} \\ \hline u_3^{top} \\ u_3^R \\ u_3^{btm} \\ \hline \vdots \\ \hline u_{\mathcal{N}}^{top} \\ u_{\mathcal{N}}^R \\ u_{\mathcal{N}}^{btm} \end{array}\right) = \left(\begin{array}{c} f_1^{top} \\ f_1^R \\ f_1^{btm} \\ \hline f_2^{top} \\ f_2^R \\ f_2^{btm} \\ \hline f_3^{top} \\ f_3^R \\ f_3^{btm} \\ \hline \vdots \\ \hline f_{\mathcal{N}}^{top} \\ f_{\mathcal{N}}^R \\ f_{\mathcal{N}}^{btm} \end{array}\right) + \left(\begin{array}{c} \mathcal{B}^{top} \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ \hline \vdots \\ \hline 0 \\ 0 \\ \mathcal{B}^{btm} \end{array}\right), \tag{87}$$

where we have assumed a discretization with $\mathcal{N}$ elements. In writing equation (87), we have included some new notation: subscripts indicate the element number, so that the matrix $\mathcal{M}_l$ represents the element operator for element l; we have decomposed the vectors u and f within each element into a reduced part, denoted with superscript R and which corresponds to all the values of the vector except the first and last, and into a top (resp. bottom) part, denoted with superscript *top* (resp. *btm*) and which corresponds to the first (resp. last) value of the vector, located at the top (resp. bottom) of the element. The boundary condition only enter in the top and bottom equation, as indicated by the last vectorial term in equation (87). This equation is not entirely rigorous: it overlooks the fact that the end point of an element coincides with the first point of the next. To correct equation (87), we note that $f_l^{btm} = f_{l+1}^{top}$ and

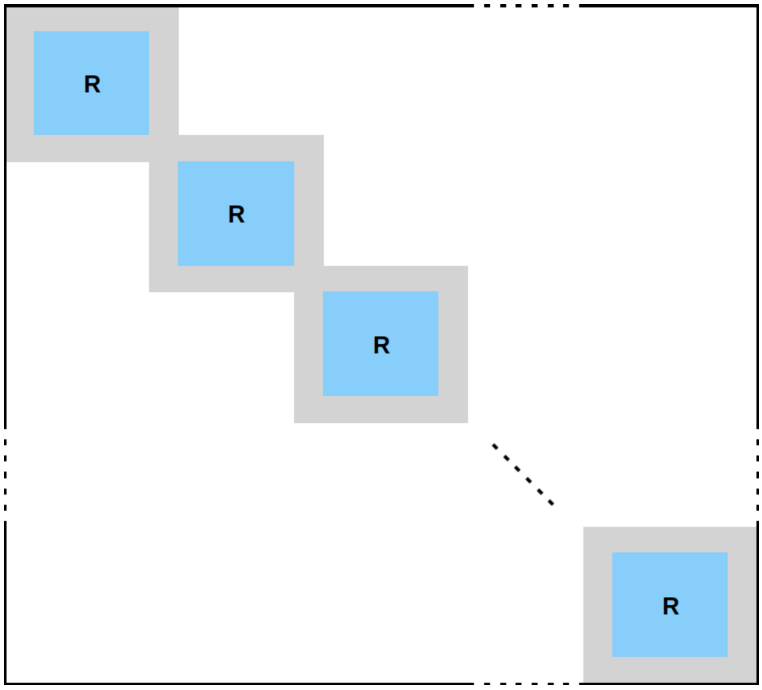


Figure 9: Representation of the Helmholtz operator matrix sparsity resulting from a spectral element collocation method. The matrix is structured into blocks on its diagonal. The blocks are structured into an inner part, shown in blue and marked R for "reduced" and an interface part, represented in gray, which couples neighboring elements. The white region is filled with 0.

$u_l^{btm} = u_{l+1}^{top}$, since these terms correspond to the evaluation of the right-hand-side and the solution at the same location. To take this fact into account, we average these values into new variables $u_l^I = (u_l^{btm} + u_{l+1}^{top})/2$ and $f_l^I = (f_l^{btm} + f_{l+1}^{top})/2$, which correspond to the value of u and f at the l^{th} interfacial point. This coupling occurs at all the $\mathcal{N}-1$ interfaces and alters the structure of the operator matrix from equation (87) by coupling the diagonal blocks through their first and last rows and columns, as shown in figure 9.

To make further progress, I find it easier to approach the full operator matrix from the element operator matrix's perspective. Let us consider the element operator matrix for element l, $\mathcal{M}_l$, as defined in equation (83). This matrix can be decomposed into blocks:

$$\mathcal{M}_l = \left(\begin{array}{c|c|c} \mathcal{M}_l^{\nwarrow} & \mathcal{M}_l^{\uparrow} & \mathcal{M}_l^{\nearrow} \\ \hline \mathcal{M}_l^{\leftarrow} & \mathcal{M}_l^{RR} & \mathcal{M}_l^{\rightarrow} \\ \hline \mathcal{M}_l^{\swarrow} & \mathcal{M}_l^{\downarrow} & \mathcal{M}_l^{\searrow} \end{array}\right), \tag{88}$$

where:

- $\mathcal{M}_l^{RR}$ has dimension $N-1 \times N-1$ and corresponds to the terms of the equations written inside an element and involving the points inside that element,
- $\mathcal{M}_l^{\nwarrow}$ has dimension 1×1 and corresponds to the term of the equation written on the top interface involving the top interfacial points,

- $\mathcal{M}_l^{\searrow}$ has dimension 1×1 and corresponds to the term of the equation written on the bottom interface involving the bottom interfacial points,
- $\mathcal{M}_l^{\nearrow}$ has dimension 1×1 and corresponds to the term of the equation written on the top interface involving the bottom interfacial points,
- $\mathcal{M}_l^{\swarrow}$ has dimension 1×1 and corresponds to the term of the equation written on the bottom interface involving the top interfacial points,
- $\mathcal{M}_l^{\uparrow}$ has dimension $1 \times N-1$ and corresponds to the terms of the equation written on the top interface that involve the points inside the element,
- $\mathcal{M}_l^{\downarrow}$ has dimension $1 \times N-1$ and corresponds to the terms of the equation written on the bottom interface that involve the points inside the element,
- $\mathcal{M}_l^{\leftarrow}$ has dimension $N-1 \times 1$ and corresponds to the terms of the equations written on the reduced elements involving the top interfacial points.
- $\mathcal{M}_l^{\leftarrow}$ has dimension $N-1 \times 1$ and corresponds to the terms of the equations written on the reduced elements involving the bottom interfacial points.

We can use the blocks defined above to re-write the operator matrix from equation (87) in its coupled form, as graphically represented in Figure 9:

$$\begin{pmatrix}
\ddots & & & & & & \\
& \frac{\mathcal{M}_{l-1}^{\searrow} + \mathcal{M}_l^{\nwarrow}}{2} & \frac{\mathcal{M}_l^{\uparrow}}{2} & \frac{\mathcal{M}_l^{\nearrow}}{2} & & & \\
& \mathcal{M}_l^{\leftarrow} & \mathcal{M}_l^{RR} & \mathcal{M}_l^{\rightarrow} & & & \\
& \frac{\mathcal{M}_l^{\swarrow}}{2} & \frac{\mathcal{M}_l^{\downarrow}}{2} & \frac{\mathcal{M}_l^{\searrow} + \mathcal{M}_{l+1}^{\nwarrow}}{2} & \frac{\mathcal{M}_{l+1}^{\uparrow}}{2} & \frac{\mathcal{M}_{l+1}^{\nearrow}}{2} & \\
& & & \mathcal{M}_{l+1}^{\leftarrow} & \mathcal{M}_{l+1}^{RR} & \mathcal{M}_{l+1}^{\rightarrow} & \\
& & & \frac{\mathcal{M}_{l+1}^{\swarrow}}{2} & \frac{\mathcal{M}_{l+1}^{\downarrow}}{2} & \frac{\mathcal{M}_{l+1}^{\searrow} + \mathcal{M}_{l+2}^{\nwarrow}}{2} & \\
& & & & & & \ddots
\end{pmatrix}, \tag{89}$$

for $2 \leqslant l \leqslant \mathcal{N} - 2$. The first and last block lines of the matrix are trivially written by taking the corresponding terms from equation (88) without dividing them by two, as they do not correspond to interactions between elements but rather to boundary equations.

The above description holds for Neumann boundary conditions. In the case of Dirichlet boundary condition, a similar treatment as in the mono-element discretization in Section 4.2 is applied: the first and last line of the matrix are removed and the contributions of the boundary points to the left-hand-side are sent to the right-hand-side as in equation (86). This only impacts the first and last elements.

4.3.2 Schur decomposition

The system resulting from the spectral element collocation method (see equation (89) and Figure 9) is a large-dimensional coupled system, which is impractical to solve without any algebraic trick. Fortunately, the coupling between elements is only done via the interface and is, thus, of low order. We can take advantage of it by re-ordering the terms:

$$\begin{pmatrix} u_{top} \\ u_1^R \\ u_1^I \\ u_2^R \\ u_2^I \\ u_3^R \\ u_3^I \\ \vdots \\ u_{\mathcal{N}-1}^I \\ u_{\mathcal{N}}^R \\ u^{btm} \end{pmatrix} \longrightarrow \left(\begin{array}{c} u_1^R \\ u_2^R \\ u_3^R \\ \vdots \\ u_{\mathcal{N}}^R \\ \hline u_{top} \\ u_1^I \\ u_2^I \\ u_3^I \\ \vdots \\ u_{\mathcal{N}-1}^I \\ u^{btm} \end{array}\right), \tag{90}$$

where the values of u inside an element, denoted by the superscript R, are now at the top of the solution vector and follow in their original order. The remaining values are those at the interfacial points, denoted by the superscript I, and at the boundaries. These are sent to the end of the solution vector in the order in which they originally appeared. This reordering allows to rewrite equation (87) in the following way:

$$\begin{pmatrix} \mathcal{M}_1^{RR} & 0 & 0 & \dots & 0 & \mathcal{M}_1^{RI} \\ 0 & \mathcal{M}_2^{RR} & 0 & \dots & 0 & \mathcal{M}_2^{RI} \\ 0 & 0 & \mathcal{M}_3^{RR} & & & \mathcal{M}_3^{RI} \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & & & \mathcal{M}_{\mathcal{N}}^{RR} & \mathcal{M}_{\mathcal{N}}^{RI} \\ \mathcal{M}_1^{IR} & \mathcal{M}_2^{IR} & \mathcal{M}_3^{IR} & \dots & \mathcal{M}_{\mathcal{N}}^{IR} & \mathcal{M}^{II} \end{pmatrix} \begin{pmatrix} u_1^R \\ u_2^R \\ u_3^R \\ \vdots \\ u_{\mathcal{N}}^R \\ u^I \end{pmatrix} = \begin{pmatrix} f_1^R \\ f_2^R \\ f_3^R \\ \vdots \\ f_{\mathcal{N}}^R \\ f^I \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \mathcal{B}^I \end{pmatrix}, \tag{91}$$

where the $\mathcal{M}_l^{RR}$ are defined in equation (88), u^I (resp. f^I) corresponds to the vector constituted of the boundary and interfacial values of u (resp. f) and ordered in the same way as the values located below the horizontal line in equation (90) and $\mathcal{B}^I$ is a sparse vector with only two non-zero values: $\mathcal{B}^{top}$ as its first entry and $\mathcal{B}^{btm}$ as its last entry. The $\mathcal{N}$ matrices $\mathcal{M}_l^{RI}$ are sparse and of dimension $(N-1) \times (\mathcal{N}+1)$. Their only non-zero entries are their l^{th} and $(l+1)^{\text{st}}$ columns, which are, respectively, $\mathcal{M}_l^{\leftarrow}$ and $\mathcal{M}_l^{\rightarrow}$. The $\mathcal{N}$ matrices $\mathcal{M}_l^{IR}$ are also sparse and of dimension $(\mathcal{N}+1) \times (N-1)$. Their only non-zero entries are their l^{th} and $(l+1)^{\text{st}}$ rows, which

are, respectively, $\mathcal{M}_l^{\uparrow}/2$ and $\mathcal{M}_l^{\downarrow}/2$. Lastly, the matrix $\mathcal{M}^{II}$ is the tridiagonal matrix:

$$\mathcal{M}^{II} = \begin{pmatrix} \mathcal{M}_1^{\nwarrow} & \mathcal{M}_1^{\nearrow} & 0 & 0 & \dots & 0 & 0 \\ \frac{\mathcal{M}_1^{\swarrow}}{2} & \frac{\mathcal{M}_1^{\searrow} + \mathcal{M}_2^{\nwarrow}}{2} & \frac{\mathcal{M}_2^{\nearrow}}{2} & 0 & \dots & 0 & 0 \\ 0 & \frac{\mathcal{M}_2^{\swarrow}}{2} & \frac{\mathcal{M}_2^{\searrow} + \mathcal{M}_3^{\nwarrow}}{2} & \frac{\mathcal{M}_3^{\nearrow}}{2} & & & 0 \\ 0 & 0 & \frac{\mathcal{M}_3^{\swarrow}}{2} & \frac{\mathcal{M}_3^{\searrow} + \mathcal{M}_4^{\nwarrow}}{2} & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & & 0 \\ 0 & 0 & & & & \frac{\mathcal{M}_{\mathcal{N}-1}^{\searrow} + \mathcal{M}_{\mathcal{N}}^{\nwarrow}}{2} & \frac{\mathcal{M}_{\mathcal{N}}^{\nearrow}}{2} \\ 0 & 0 & 0 & \dots & 0 & \mathcal{M}_{\mathcal{N}}^{\swarrow} & \mathcal{M}_{\mathcal{N}}^{\searrow} \end{pmatrix}. \tag{92}$$

Adapting the above to Dirichlet boundary conditions is straightforward. Since the boundary conditions have been moved to the interface part of the field during the Schur decomposition, we need to remove the first and last rows of u^I. The resulting changes impact the $\mathcal{M}_l^{IR}$ by removing their first and last rows, the $\mathcal{M}_l^{RI}$ by removing their first and last columns and M^{II} by removing its first and last rows and first and last columns. The right-hand-side does not longer feature $\mathcal{B}^I$, which corresponded to the contribution of the Neumann boundary condition, but now should include the penalty for the Dirichlet boundary conditions (see equation (86)) on the first and last elements.

4.3.3 Solution method

The aim of the Schur decomposition was to take advantage of the the structure of the Helmholtz operator generated by the spectral element collocation method. In particular, we can see from equation (91) that the operator couples elements with each other via their interface so that knowing the solution at the interface decouples the elements. To use this, let us first write the system in its index form:

$$\mathcal{M}_n^{RR}\, u_n^R \,+\, \mathcal{M}_n^{RI}\, u^I \quad = f_n^R, \qquad n = 1, \dots, \mathcal{N}, \tag{93}$$

$$\sum_{n=1}^{\mathcal{N}} \mathcal{M}_n^{IR}\, u_n^R \,+\, \mathcal{M}^{II}\, u^I \quad = f^I + \mathcal{B}^I. \tag{94}$$

We can now express u inside each elements as a function of its value at the interfaces:

$$\mathcal{M}_n^{RR} u_n^R = f_n^R - \mathcal{M}_n^{RI} u^I, \qquad n = 1, \dots, \mathcal{N}, \tag{95}$$

which can then be used to express the value of u inside the elements in equation (94):

$$\left[\mathcal{M}^{II} - \sum_{n=1}^{\mathcal{N}} \mathcal{M}_n^{IR} \left(\mathcal{M}_n^{RR}\right)^{-1} \mathcal{M}_n^{RI}\right] u^I = f^I + \mathcal{B}^I - \sum_{n=1}^{\mathcal{N}} \mathcal{M}_n^{IR} \left(\mathcal{M}_n^{RR}\right)^{-1} f_n^R. \tag{96}$$

This equation allows to determine the solution at the interfacial points, which can then be used to decouple the $\mathcal{N}$ equations (95) and obtain the value of the solution inside each element.

To summarize, the problem is solved sequentially by first solving for the solution at the interfaces:

$$u^I = \left(\mathcal{S}^L\right)^{-1} \left[f^I + \mathcal{B}^I - \sum_{n=1}^{\mathcal{N}} \mathcal{S}_n^R f_n^R \right], \tag{97}$$

where $\mathcal{S}^L = \mathcal{M}^{II} - \sum_{n=1}^{\mathcal{N}} \mathcal{M}_n^{IR} \left(\mathcal{M}_n^{RR}\right)^{-1} \mathcal{M}_n^{RI}$ and $\mathcal{S}_n^R = \mathcal{M}_n^{IR} \left(\mathcal{M}_n^{RR}\right)^{-1}$, for $n = 1, \ldots, \mathcal{N}$, are precomputed matrices. The solution inside the elements is then simply computed using:

$$u_n^R = \left(\mathcal{M}_n^{RR}\right)^{-1} \left[f_n^R - \mathcal{M}_n^{RI} u^I \right], \qquad n = 1, \ldots, \mathcal{N}, \tag{98}$$

with the values of u^I just obtained. We note that these equations only require the following matrices to be stored and carried over during the computation: the $\mathcal{M}_n^{RR}$, the $\mathcal{M}_n^{RI}$, $\mathcal{S}^L$ and the $\mathcal{S}_n^R$. These matrices are computed in a preliminary step.

5 Dry salt lake solver

We recall that the dry salt lake is represented using a cuboid of extent h in the vertical direction z and of extent Γ_x and Γ_y in the horizontal directions x and y respectively. On this domain, we solve the following equations:

$$\nabla^2 p = Ra\,\partial_z S, \tag{99}$$

$$\mathbf{u} = -\nabla p + Ra\,S\,\hat{\mathbf{z}}, \tag{100}$$

$$\partial_t S + (\mathbf{u}\cdot\nabla)\,S = \nabla^2 S, \tag{101}$$

where p is the pressure, Ra is the Rayleigh number, S is the salinity, $\mathbf{u}$ is the velocity field, $\hat{\mathbf{z}}$ is the descending unit vector and t is the time. These equations are accompanied with the following boundary condition in the vertical direction:

$$S = \partial_z p - Ra = 1, \qquad \text{at } z = 0, \tag{102}$$

together with one of the two bottom boundary conditions:

$$\text{reflective :} \qquad S = \partial_z p - 1 = 0, \qquad \text{at } z = h, \tag{103}$$

$$\text{penetrative :} \qquad \partial_z S = p = 0, \qquad \text{at } z = h, \tag{104}$$

and with periodic boundary conditions in both horizontal directions.

The domain is discretized using a Cartesian grid, with equidistributed points in the horizontal directions:

$$x_i = \frac{i\,\Gamma_x}{N_x}, \qquad i = 0,\dots,N_x-1, \tag{105}$$

$$y_j = \frac{j\,\Gamma_y}{N_y}, \qquad j = 0,\dots,N_y-1, \tag{106}$$

where N_x and N_y are the number of points in the x and y directions respectively; and with a Gauss–Lobatto–Legendre element discretization in the vertical direction z:

$$z_{k,l} = \frac{(l-1)\,h}{N_e} + \tilde{z}_k, \qquad l = 1,\dots,N_e, \tag{107}$$

where N_e is the number of elements and $\tilde{z}_k$ is the k^{th} Gauss–Lobatto–Legendre point of the interval $[0; h/N_e]$, where $k = 0,\dots,N_z$ and N_z is the number of discretization intervals per element. Basic information on Gauss–Lobatto–Legendre discretization is presented in Section 4.2.

The horizontal directions are treated using the Fourier–Galerkin formalism developed in Section 4.1, while the vertical direction is handled using the spectral element method described in Section 4.3. We discretize the salinity as follows:

$$\begin{aligned} S(x_i,y_j,z_{k,l}) = \sum_{r=1}^{N_e}\sum_{q=0}^{N_z}\sum_{n=-N_y/2+1}^{N_y/2} \psi_{q,r}(z_{k,l})\exp\left(\imath n j\frac{2\pi}{N_y}\right)\Bigg\{ \breve{S}_{0,n,q,r}\ \dots \\ + \sum_{m=1}^{N_x/2-1}\left[\breve{S}_{m,n,q,r}\exp\left(\imath m i\frac{2\pi}{N_x}\right) + \breve{S}^*_{m,n,q,r}\exp\left(-\imath m i\frac{2\pi}{N_x}\right)\right]\dots \\ + \breve{S}_{N_x/2,n,q,r}(-1)^i\Bigg\}, \end{aligned} \tag{108}$$

where $\imath$ is the imaginary unit number, $i = 0, \dots, N_x - 1$, $j = 0, \dots, N_y - 1$, $k = 0, \dots, N_z$ and $l = 1, \dots, N_e$. The coefficients in "spectral" space, $\breve{S}_{m,n,q,r}$ are complex, owing to the use of the exponential Fourier basis to treat the x- and y-directions. The interpolation polynomials in the z-direction are the element-specific Lagrange polynomials:

$$\psi_{q,r}(z) = \begin{cases} \prod_{\substack{\alpha=0 \\ l \neq q}}^{N_z} \dfrac{z - z_{\alpha,r}}{z_{q,r} - z_{\alpha,r}} & \text{if} \quad \dfrac{(r-1)h}{N_e} \leqslant z \leqslant \dfrac{rh}{N_e} \\ = 0 & \text{otherwise} \end{cases}, \tag{109}$$

for $q = 0, \dots, N_z$ and $r = 1, \dots, N_e$. These polynomial possess the extended Kronecker property $\psi_{q,r}(z_{k,l}) = \delta_{kq}\delta_{lr}$. The other state variables are discretized similarly.

We first Fourier transform the equations in both horizontal directions. This operation turns each three-dimensional partial differential equation into a set of $N_x/2 + 1 \times N_y$ decoupled one-dimensional partial differential equation on the vertical direction, one partial differential equation for each horizontal wavenumber. For $m = 1, \dots, N_x/2 + 1$ and $n = -N_y/2 + 1, \dots, N_y/2$, these equations are:

$$\left(\partial_z^2 - k_{m,n}\right) \hat{p}_{m,n} = Ra\, \partial_z \hat{S}_{m,n}, \tag{110}$$

$$\hat{\mathbf{u}}_{m,n} = -\hat{\mathbf{f}_1}_{m,n} + Ra\, \hat{S}_{m,n}\, \hat{\mathbf{z}}, \tag{111}$$

$$\partial_t \hat{S}_{m,n} + \hat{f}_{2m,n} = \left(\partial_z^2 - k_{m,n}\right) \hat{S}_{m,n}, \tag{112}$$

where

$$k_{m,n} = 4\pi^2 \left(\frac{m^2}{\Gamma_x^2} + \frac{n^2}{\Gamma_y^2} \right), \tag{113}$$

and where the use of a circumflex accent denotes that the quantity has been Fourier transformed in x and y, $\hat{\mathbf{f}_1}$ (resp. $\hat{f}_2$) is the Fourier transform of ∇p (resp. $(\mathbf{u} \cdot \nabla) S$).

We now apply a Taylor expansion-based temporal discretization for the time-derivative and the nonlinear terms, treating the diffusion term implicitly:

$$\left[\alpha_0 - \triangle t \left(\partial_z^2 - k_{m,n}\right)\right] \hat{S}^t_{m,n} = -\sum_{i=1}^{\nu} \alpha_i \hat{S}^{t-i\triangle t}_{m,n} - \triangle t \sum_{i=1}^{\nu} \beta_i \left(\hat{f}_{2m,n}\right)^{t-i\triangle t}, \tag{114}$$

$$\left(\partial_z^2 - k_{m,n}\right) \hat{p}^t_{m,n} = Ra\, \partial_z \hat{S}^t_{m,n}, \tag{115}$$

$$\hat{\mathbf{u}}^t_{m,n} = -\hat{\mathbf{f}_1}^{\,t}_{m,n} + Ra\, \hat{S}^t_{m,n} \hat{\mathbf{z}}, \tag{116}$$

where $\triangle t$ is the time-step, the α_i, $i = 0, \dots, \nu$ (resp. β_i, $i = 1, \dots, \nu$) are the coefficients of expansion for the time-derivative (resp. nonlinear term) for the scheme described in Section 3 at order ν and where the superscripts involving t indicate the time at which the term is taken.

The discretized equations are solved in a sequential manner, as follows. Firstly, the salinity is solved for via the equation:

$$\mathcal{H}^S_{m,n} \hat{S}^t_{m,n} = -\sum_{i=1}^{\nu} \alpha_i \hat{S}^{t-i\triangle t}_{m,n} - \triangle t \sum_{i=1}^{\nu} \beta_i \left(\hat{f}_{2m,n}\right)^{t-i\triangle t}, \tag{117}$$

for each horizontal wavenumber couple (m, n), where $m = 1, \dots, N_x/2 + 1$ and $n = -N_y/2 + 1, \dots, N_y/2$. This is achieved by inverting the salinity Helmholtz operator $\mathcal{H}^S_{m,n} = \alpha_0 + \triangle t\, k_{m,n} -$

$\triangle t\, \partial_z^2$ together with the Dirichlet boundary condition at the top:

$$S^t = 1, \qquad \text{at } z = 0, \tag{118}$$

and either a Dirichlet or a Neumann boundary condition at the bottom:

$$\text{reflective :} \quad S^t = 0, \qquad \text{at } z = h, \tag{119}$$
$$\text{penetrative :} \quad \partial_z S^t = 0, \qquad \text{at } z = h, \tag{120}$$

according to the Schur decomposition method explained in Section 4.3. Secondly, knowing the salinity at the running time t, we invert another Helmholtz problem, this time for the pressure:

$$\mathcal{H}^p_{m,n} \hat{p}^t_{m,n} = Ra\, \partial_z \hat{S}^t_{m,n}, \tag{121}$$

where $\mathcal{H}^p_{m,n} = -k_{m,n} + \partial_z^2$, together with the Neumann boundary condition at the top:

$$\partial_z p^t = 1 + Ra, \qquad \text{at } z = 0, \tag{122}$$

and either a Neumann or a Dirichlet boundary condition at the bottom:

$$\text{reflective :} \quad \partial_z p^t = 1, \qquad \text{at } z = h, \tag{123}$$
$$\text{penetrative :} \quad p^t = 0, \qquad \text{at } z = h. \tag{124}$$

The way this second Helmholtz problem inversion is carried out is also explained in Section 4.3. Lastly, the velocity field is "read" by computing the right-hand-side of:

$$\hat{\mathbf{u}}^t_{m,n} = -\hat{\mathbf{f}}_{\mathbf{1}}{}^t_{m,n} + Ra\, \hat{S}^t_{m,n} \hat{\mathbf{z}}, \tag{125}$$

where $\hat{\mathbf{f}}_{\mathbf{1}}{}^t_{m,n}$ is the gradient of the pressure just computed.

6 Performance

We can predict the theoretical performance of DryLa by assuming it is limited by algebraic operations, leaving aside any lower level operations like memory fetching and writing. While the initialization of the code can be costly due to the number of calculations required to compute $\mathcal{M}_n^{RR}$, $\mathcal{M}_n^{RI}$, $\mathcal{S}^L$ and $\mathcal{S}_n^R$ for both the salinity and the pressure equations (see Section 4.3.3), it is only carried out a small number of times compared to the operations involved in the time-stepping loop. The steps involved by time-stepping are:

- Evaluation of the right-hand-side for the salinity/pressure equation: $O(N_x N_y N_z N_e)$ operations.
- Solution of the salinity/pressure equation: see below.
- Computation of the horizontal derivatives for the salinity/pressure: taking advantage of the availability of the spectral representation of the salinity/pressure, the derivative is computed in spectral space and involves backward Fourier transforms. This is carried out in $O(N_x N_y N_z N_e \ln(N_x N_y))$ operations.
- Computation of the vertical derivative for the salinity/pressure: $O(N_x N_y N_z^2 N_e)$ operations.
- Evaluation of the velocity: $O(N_x N_y N_z N_e)$ operations.

Computation of the salinity and of the pressure relies on a spectral element method and is explained in Section (4.3.3). It involves the following steps:

- Splitting/reassembling the fields into interfacial and element values: $O(N_x N_y N_z N_e)$ operations.
- Implementation of the vertical boundary conditions: $O(N_x N_y)$ operations.
- Horizontal Fourier transforms: $O(N_x N_y N_z N_e \ln(N_x N_y))$ operations.
- Computation of the right-hand-side for the interfacial (resp. element) problem: taking advantage of the fact that $\mathcal{S}_n^R$ (resp. $\mathcal{M}_n^{RI}$) only has two non-zero rows (resp. columns), this step costs $O(N_x N_y N_z N_e)$ operations.
- Interface solution computation: taking advantage of the fact that $\mathcal{S}^L$ is tri-diagonal, the interface problem is solved using an LU decomposition and costs $O(N_x N_y N_e)$ operations.
- Element solution computation: this step is dominated by the spectral transforms in the vertical direction and costs $O(N_x N_y N_z^2 N_e)$ operations.

The limiting operations are consequently the horizontal Fourier transforms, which scale like $N_x N_y \ln(N_x N_y)$, and the vertical derivatives and the element solution computation, which scale like N_z^2. All operations scale, at most, like N_e.

To assess the actual performance of DryLa, we proceeded in the following way. A baseline spatial discretization was set: $N_x = N_y = 32$, $N_z = 20$ and $N_e = 10$. Each simulation used the second-order time-discretization scheme and was run on 8 threads for 4 hours on ARC4, the high-performance computing facility of the University of Leeds. Some of the larger simulations

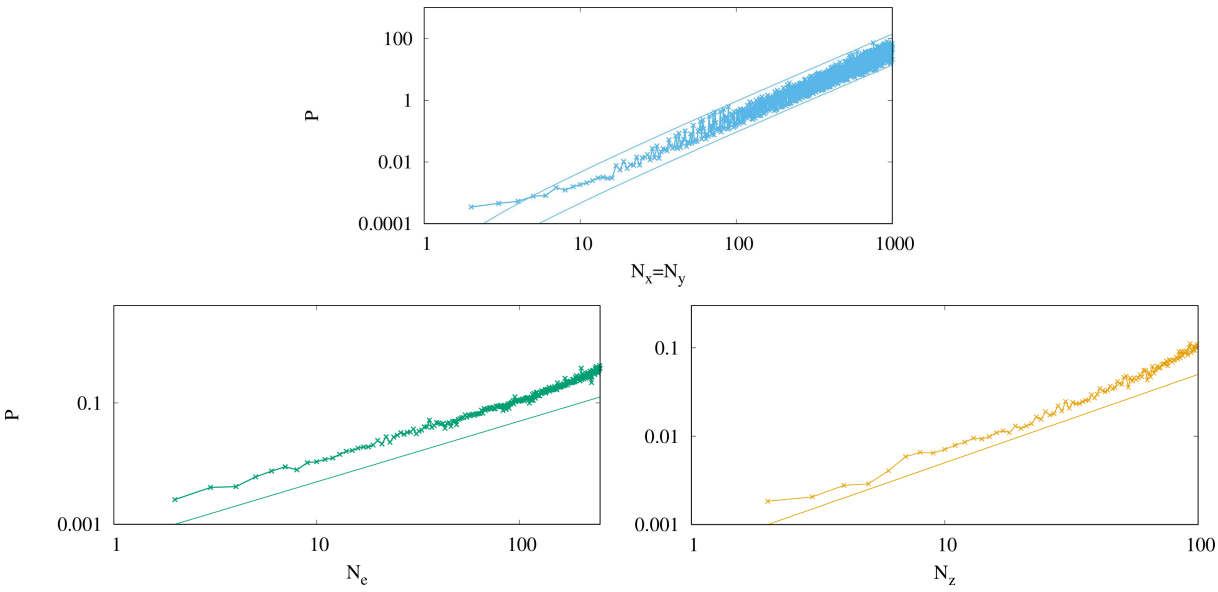


Figure 10: Performance of DryLa represented by the inverse of the number of time-steps calculated P as a function of the changed discretization parameter. The top panel shows P as a function of the horizontal discretization parameter $N_x = N_y$ together with curves scaling like $N_x^2 \ln(N_x^2)$ (version 5.6). The bottom left panel shows P as a function of the number of spectral elements in the vertical direction N_e as well as curves scaling like N_e (version 6.2). The bottom right panel shows P as a function of the number of points per element in the vertical direction, N_z, next to a curve scaling like N_z (version 5.6). Details of the simulation are given in the text.

had larger memory requirements; these were satisfied as needed by requesting additional memory and did not incur any measurable impact on performance. In order to test the scalability of the solver only, all writing output was disabled except for the iteration count. The results were collected in the form of the number of iterations completed within the allocated time and represented as the inverse of this number, hereafter referred to as P.

The top panel of Figure 10 shows P as a function of N_x with the constraint $N_x = N_y$ in order to test the performance scaling with the number of points in the horizontal direction without any direction bias. As N_x increases, the cost per iteration increases in proportion to $N_x^2 \ln(N_x^2)$, indicating that the horizontal Fourier transforms are indeed the dominant contribution to the computing cost. The lower panels of Figure 10 show how the performance of DryLa scales with the vertical discretization parameters. The performance scales linearly with N_e and with N_z until N_z is sufficiently large, where the cost increases faster than linearly against N_z. It is logical to presume that, ultimately, performance will scale like the theoretical quadratic law in N_z, however, this was not observed for the values of N_z tested. These should not, however, exceed $O(10)$ values[1] to avoid the deterioration of the solution due to the poor conditioning of the differentiation matrix. In practice, it is often found that values around $N_z = 20$ represent a safe default choice.

[1] some references go as far as suggesting $N_z \leqslant 64$.

References

[1] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. Spectral methods in fluid dynamics. *Springer*, 1988.

[2] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93, 2005.

[3] G. E. Karniadakis, M. Israeli, and S. A. Orszag. High-order splitting methods for the incompressible Navier–Stokes equations. *J. Comput. Phys.*, 97, 1991.

[4] J. Lasser, M. Ersnt, and L. Goehring. Stability and dynamics of convection in dry salt lakes. *J. Fluid Mech.*, 917, 2021.