Modeling the open-boundary approach to pressure projection

Kraig Winters

Scripps Institution of Oceanography, University of California San Diego,La Jolla, CA 92093, USA

(Received 26 March 2021)

For open boundary problems, it is necessary to assume even extensions and use cosine expansions for the velocity field in a spectral-type algorithm. Here we explore by example a pressure-projection algorithm for a one-dimensional evolution equation intended to be a model for the approach used in **flow_solve**.

1. The model problem

We model one-dimensional flow $u(x)$ [m/s] in the closed interval $x \in [0, L]$ with boundary conditions $u(0) = u(L) = u_B(t)$, which which serve as a surrogate for Dirichlet boundary values obtained from solving a 'parent' problem over a larger domain. The governing equations are

$$u_t = \mathcal{F}_{\text{body}} + \nu u_{xx} - p_x \quad , \quad u_x = 0 \tag{1.1}$$

which describe divergent-free flow in one dimension. The 1d continuity equation implies that the only allowable flow is constant with respect to x and so, in this simple problem, the flow is a time dependent uniform flow, *i.e.*

$$u(x,t) = u_B(t) \tag{1.2}$$

Note that the viscous term is identically equal to zero for these flows. In the simplest case, when $\mathcal{F}_{\text{body}} = 0$, the flow is driven by the boundary values and the corresponding pressure field is

$$p(x,t) = -\frac{\partial u}{\partial t} x. \tag{1.3}$$

At first glance, this simple model problem seems outside the scope of using cosine expansions in x for u and p.

2. Pressure projection using cosine expansions

We denote the cosine expansion of $f(x)$ as $\hat{f}(k)$. The expansion coefficients are obtained by Fourier transforming the even extension of f about $x = L$ over the open interval $x \in [0, 2L)$. Note that these Fourier coefficients are conjugate symmetric and real. Our approach will be to compute x derivatives by taking the cosine transform, multiplying by the wavenumbers k, and inverse transforming back to physical space. In these demonstration problems, these operations are carried out by explicitly constructing even extensions from $[0, L]$ to $[0, 2L)$ and using FFTs. In **flow_solve** cosine transforms are used directly, a procedure that is more efficient with respect to memory and computation.

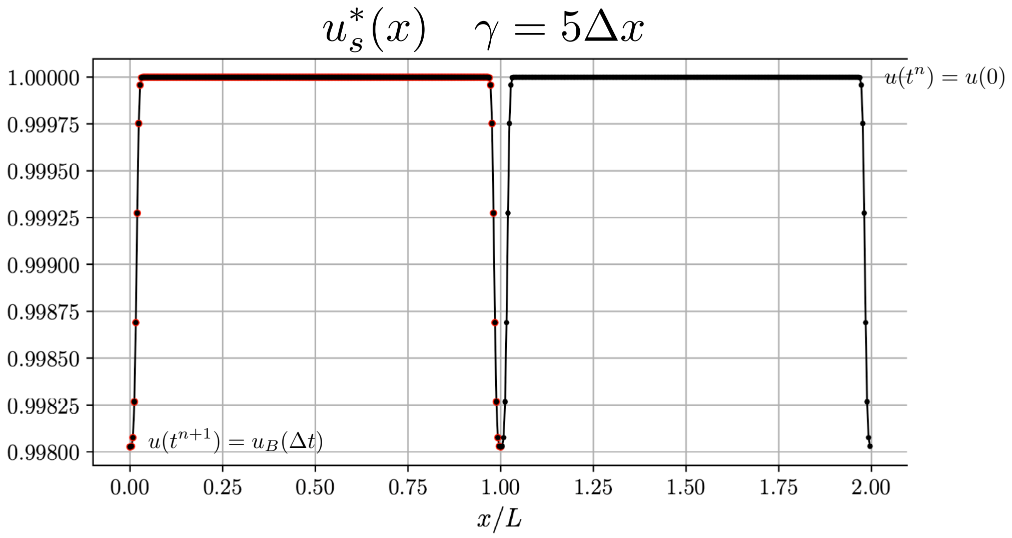

FIGURE 1. The function $u_s^*(x)$ for $n = 1$, even extended across $x = L$. The function "smoothly jumps" from the value $u(0) = 1$ to the value $u_B(\Delta t)$ at the boundaries $x = 0, L$ in such a way that the derivatives are equal to zero there. Here $u_s^*(x)$ is a well-behaved function with a narrow transition scale of width $\gamma = 5\Delta x$ where $\Delta x = L/256$

Let u^n be the solution at some discrete time value $t^n = n\Delta t$. To estimate u^{n+1} we adopt the following procedure.

(1) Compute

$$u^* = u^n + \int_{t_n}^{t_{n+1}} \mathcal{F}_{\text{body}} + \nu u_{xx} \, \mathrm{d}t \tag{2.1}$$

using an explicit time stepping method. In these examples, we will use the Euler method for simplicity. Depending on the configuration, higher order Adams methods are used in **flow_solve**. We assume that u^n and u^* are smooth, continuous functions when even-extended across $x = L$, *i.e.* that their first derivatives vanish at the endpoints and their second derivatives are finite. For the illustration problem that follows, we take $\mathcal{F}_{\text{body}} = 0$ and $u_B(t) = \cos(2\pi t/T)$.

(2) Introduce the velocity boundary conditions at t^{n+1} into $u^*(x)$ as "smoothed discontinuities", *i.e.* jumps that are spread over thin transition regions of width γ. Call the result $u_s^*(x)$. For this example u_s^* transitions from $u^n = u_B(t^n)$ in the interior to $u_B(t^{n+1})$ at the boundaries.

$$u_s^*(x) = \begin{cases} (u^*(x) - u_B(0, t^{n+1}))e^{-(\frac{x}{\gamma})^4} & + u_B(0, t^{n+1}) \quad 0 \le x \le L/2, \\ (u^*(x) - u_B(L, t^{n+1}))e^{-(\frac{x-L}{\gamma})^4} & + u_B(L, t^{n+1}) \quad L/2 < x < L \end{cases} \tag{2.2}$$

Figure (1) shows $u_s^*(x)$ and its even extension. The choice of exponent in the transition function ensures that both the first and second derivatives of $u_s^*(x)$ vanish at $x = 0, L$. This allows both u_s^* and its derivative to be expanded in well-behaved cosine series in x.

(3) Compute $\frac{\partial}{\partial x} u_s^*$ using cosine expansions. The result and its even extension are shown in

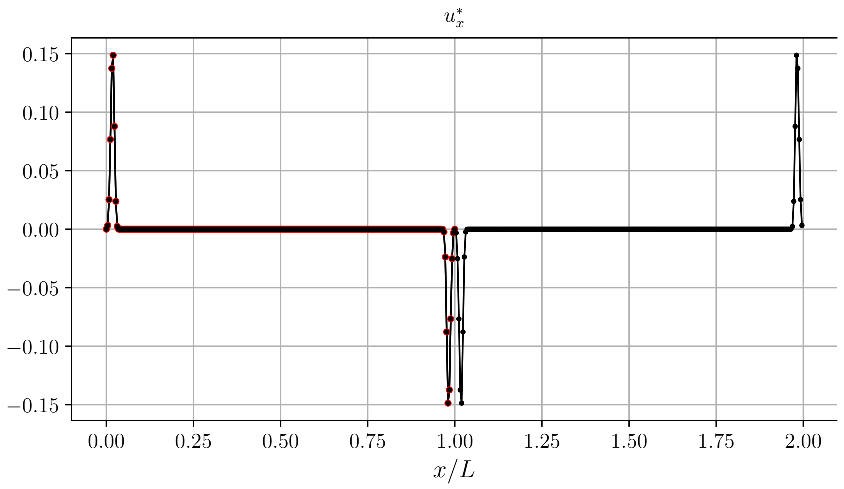

FIGURE 2. The function $\frac{\partial}{\partial x} u_s^*(x)$ for $n = 1$, even extended across $x = L$.

Figure (2). The even extension is continuous and has derivatives that vanish at $x = 0, L$. For this example problem, where u^* is only nonzero due to the jump in boundary values, $\frac{\partial}{\partial x} u_s^*$ is a resolvable approximation to the derivative of a delta function located at the boundaries.

(4) Given the definition of $u^*(x)$ (2.1), it follows that the corrected velocity

$$u^{n+1} = u^* - \Delta t\, \phi_x$$

provided that $\phi_x = \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} p_x \, \mathrm{d}t$. Using the smoothed field u_s^* in place of u^* in this equation and differentiating with respect to x yields

$$\phi_{xx} = \frac{-1}{\Delta t} \frac{\partial}{\partial x} u_s^*.$$

Because u_s^* already satisfies the desired boundary conditions for $u(t^{n+1})$ at $x = 0, L$, the boundary conditions on ϕ are simply $\phi_x = 0$ at $x = 0, L$. We solve for ϕ by taking the cosine transform of $\frac{\partial}{\partial x} u_s^*$, multiplying by $\frac{-1}{\Delta t\, k^2}$, and inverse transforming. The resulting function $\phi(x)$ is shown in Figure(3).

(5) Compute ϕ_x using cosine expansions. The updated velocity is then $u^{n+1} = u_s^* - \Delta t\, \phi_x$. Thus, we have constructed u^{n+1} that satisfies the equation of motion, the divergence-free condition, and the required boundary conditions. The pressure gradient ϕ_x and the computed solution $u^{n+1}(x)$ are shown in Figures 4 and 5 respectively. The mean error $|u(t^{n+1}, x) - u_B(t^{n+1})|$ is 9.5×10^{-8}.

After one time step, we have a very good approximate solution for u in a boundary driven problem using a numerical method based on implicit satisfaction of homogeneous Neumann conditions for all dependent variables. This is achieved via a discrete version of exchanging inhomogeneity in the boundary conditions for inhomogeneity in the source

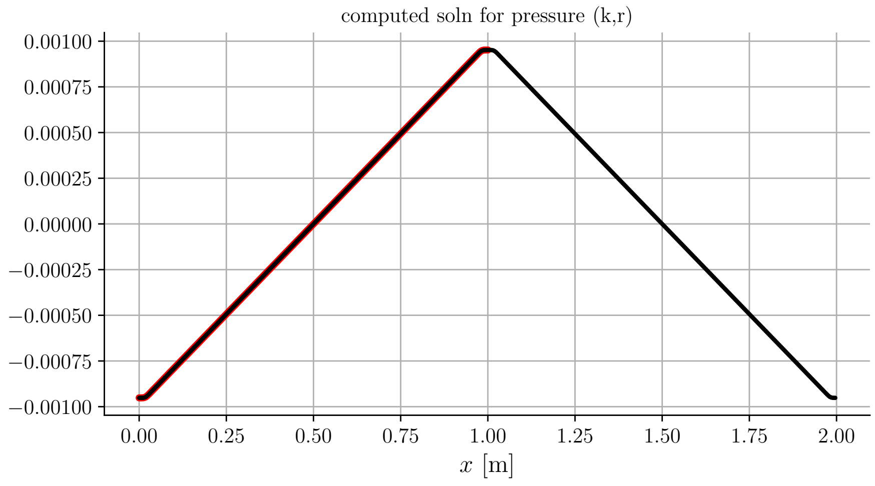

FIGURE 3. The function $\phi(x)$ for $n = 1$, even extended across $x = L$. The pressure field is linear in x throughout the interior of the domain but transitions to a vanishing slope within γ transition regions near the boundaries. The function is well behaved, has vanishing derivatives at the boundaries and has continuous second derivatives. The pointwise residual $\phi_{xx} + \frac{-1}{\Delta t}\frac{\partial}{\partial x}u_s^*$, normalized by the maximum magnitude of $\phi_x x$, is everywhere less than 10^{-12}.

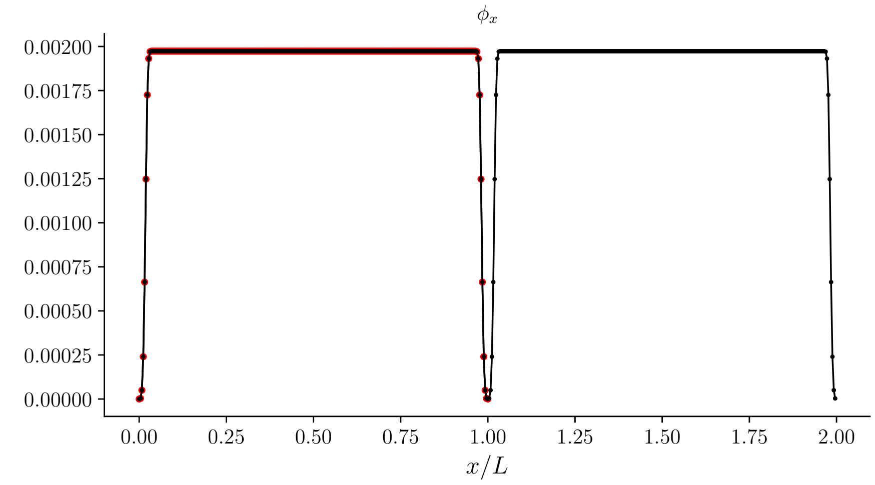

FIGURE 4. The function $\phi_x(x)$ for $n = 1$, even extended across $x = L$. The gradient is approximately constant in the interior but zero at the boundaries.

term in the elliptic equation for pressure. While the errors in the velocity field are small in magnitude, they have small scale components and we have the possibility of discontinuities in derivatives higher than $n = 2$. For all time steps after the first, the viscous term will be nonzero as viscosity will act on the small scale errors in the solution. While this seems like a good thing in that small scales are continuously damped, continuity issues may give rise to the continuous generation of these errors. Stability of the time integration will depend on the maintanence of smoothness at small scales. To enhance smoothness, we let $\nu = 0.01\Delta x^2/\Delta t$ be nonzero and we use a Gaussian filter when dif-

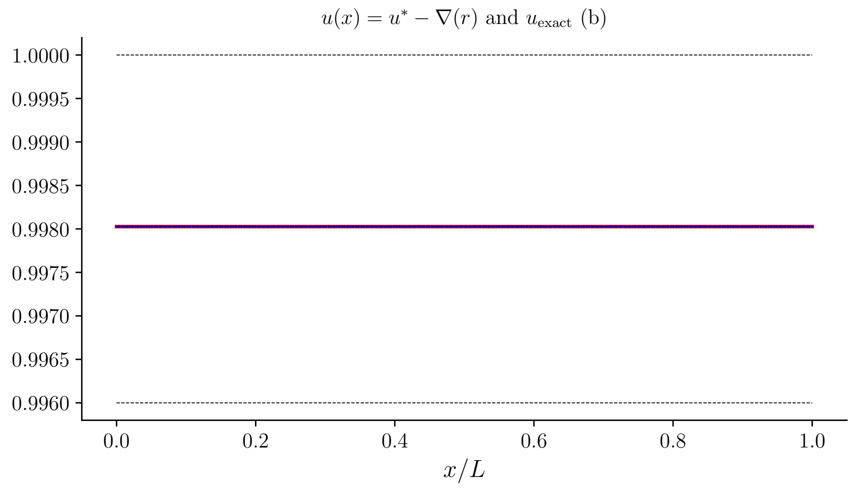

FIGURE 5. The function $u(t^{n+1}, x) = u_s^* - \Delta t \phi_x$ for $n = 1$, even extended across $x = L$. The velocity is approximately constant and matches the boundary value $u_B()t^{n+1}$.

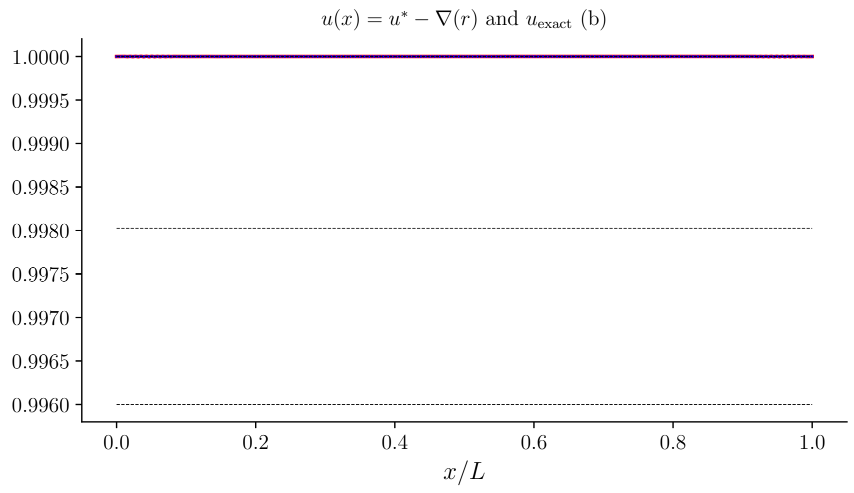

FIGURE 6. The computed solution $u(T, x) = u_s^* - \Delta t \phi_x$ after 100 time steps, even extended across $x = L$. The velocity is nearly constant and approximately matches the boundary value $u_B(T) = 1$. It is a little noisy at small scales, but not significantly worse than after a single time step, *eg.* the mean magnitude of the difference between the computed and exact solutions is 4.5×10^{-7}.

ferentiating in wavenumber space to taper the coefficients corresponding to the highest 10% of the discrete wavenumbers.

3. Time stepping

For $\Delta t = T/100$, we now integrate for one period of the boundary forcing. The exact solution is again $u = 1$ and the computed solution is shown in Figure 6 along with the computed pressure gradient ϕ_x in Figure 7.

We now check the spectral behavior of the functions used to construct the solution after

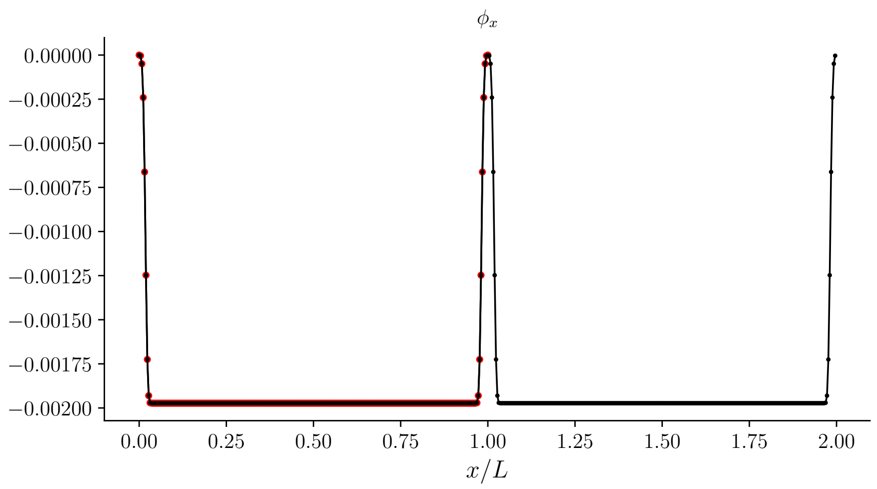

FIGURE 7. The function $\phi_x(x)$ at step 100, even extended across $x = L$. The gradient ϕ_x is essentially constant in the interior but vanishes at the boundaries. ϕ_x has a well behaved expansion in a cosine series.

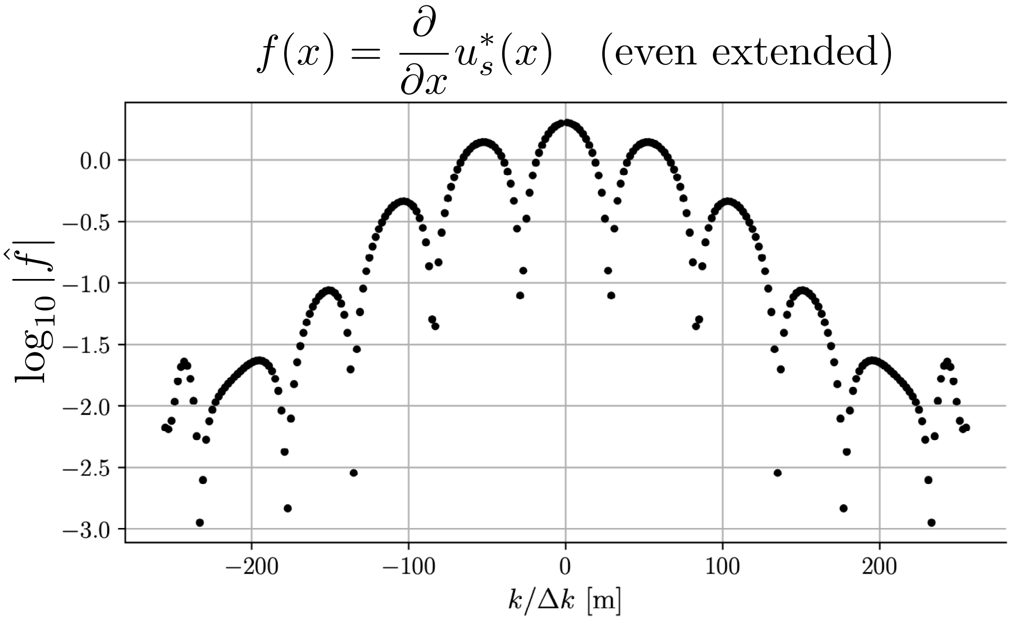

FIGURE 8. Fourier transform of the even-extended source term in the Poisson equation for pressure. Odd coefficients are equal to zero and not shown.

integrating 100 steps. Shown are the Fourier transforms of the underlying functions after even extension from $[0, L]$ to $[0, 2L)$. The cosine coefficients are twice the odd coefficients for $k \geq 0$. The transformed Poisson solution $\hat{\phi}(k)$ is simply a multiplication of $\hat{f}(k)$ by $1/k^2$, while the calculation of $\hat{\phi_x}$ is just a multiplication of this result by k. The spectral shape of these functions are shown in Figure 9.

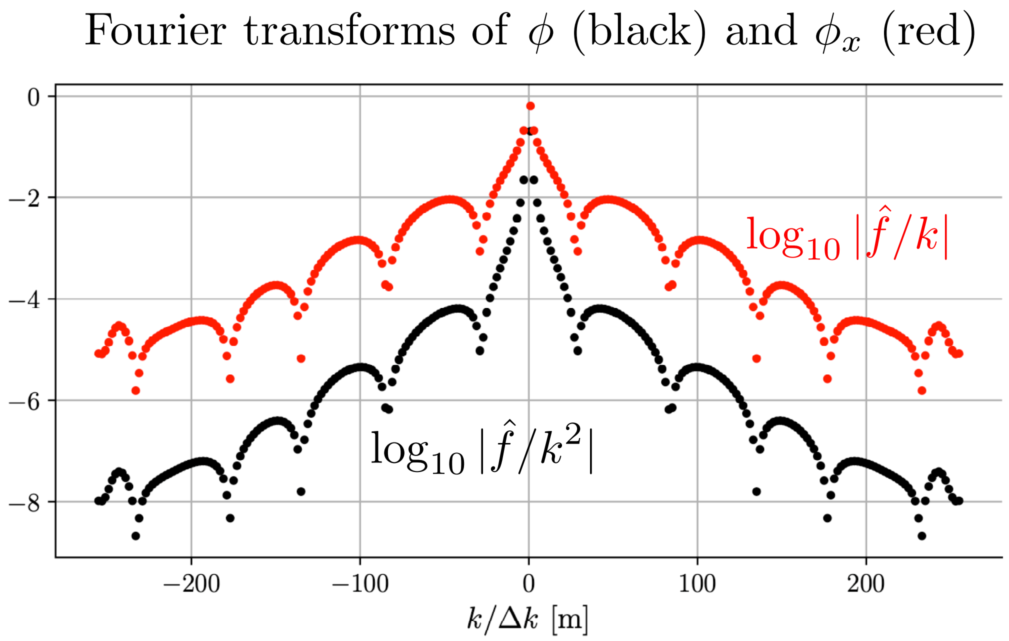

FIGURE 9. Fourier transform of the even-extended functions ϕ and ϕ_x. Odd coefficients are equal to zero and not shown.

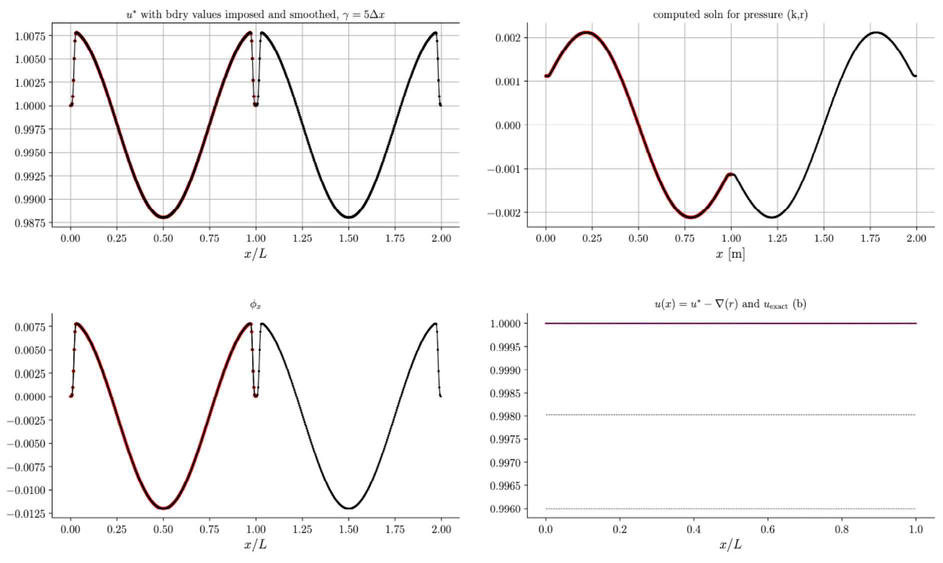

FIGURE 10. Various fields after 100 time steps for nonzero body forcing.

4. Removal of divergence resulting from body forces

To this point, we have focused on the behavior of the solution procedure in response to the imposition of velocity boundary conditions. Essentially, we have absorbed the inhomogeneous boundary information into a modified source term for the pressure Poisson equation while retaining homogeneous Neumann conditions. We now show that this procedure also removes any interior divergence that appears in u_* by virtue of the body force. In this simple one-dimensional example, any non-constant component of $\mathcal{F}_{\rm body}(x)$ will need to be removed in the pressure projection step. To illustrate, we take

$$\mathcal{F}_{\rm body}(x) = \cos(2\pi x/L) \tag{4.1}$$

a generic term that will introduce divergence but is itself an even function about $x = 0, L$.

The exact solution for velocity remains the same:u is constant in x and matches the imposed, time dependent boundary values. The pressure field, however, is different. The pressure correction at each time step must now accelerate the flow to match the updated boundary values while at the same time removing the divergence induced by the body forcing. Figure 10 shows that this introduces no further difficulties. The mean magnitude of the difference between the computed and exact solutions is 4.4×10^{-7}, essentially unchanged.

5. Indirect imposition of boundary conditions by nudging

Here we consider the same physical problem with solution $u = u_B(t)$ for all x. The solution will be approximated by adding a forcing term to the equation for u that nudges the computed solution toward the boundary values over a time scale on the order of Δt. The additional forcing term takes the form

$$\mathcal{F}_{\rm nudge} = \frac{-1}{\tau} \left\{ \mathcal{W}_0(x)\ [u(x,t) - u_B(0,t)] + \mathcal{W}_L(x)\ [u(x,t) - u_B(L,t)] \right\} \tag{5.1}$$

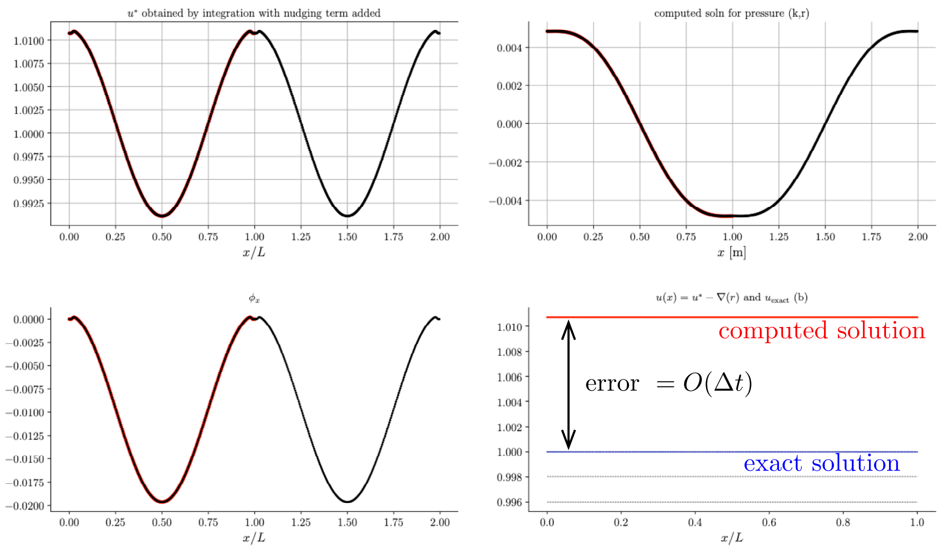

FIGURE 11. Various fields after 100 time steps for nonzero body forcing.

where $\tau = O(\Delta t)$ and the near-boundary windowing functions are given by

$$\mathcal{W}_0(x) = e^{-(\frac{(x-0)}{\sigma})**4} \quad \text{and} \quad \mathcal{W}_L(x) = e^{-(\frac{(x-L)}{\sigma})**4}. \tag{5.2}$$

In practice, τ has to be greater than about $3\Delta t$ in order to maintain stability of the discrete time integration.

With this approach

$$u^* = u^n + \int_{t_n}^{t_{n+1}} \mathcal{F}_{\text{body}} + \mathcal{F}_{\text{nudge}} + \nu u_{xx}\,\mathrm{d}t \tag{5.3}$$

and there is no need to further adjust the values near the boundaries nor do any smoothing. Smoothness of u^* near the endpoints is controlled by the vanishing derivatives of the windowing functions at the boundaries. Time-varying boundary conditions will never be satisfied exactly but the magnitude of the boundary error scales like τ/T where T is the time scale over which the boundary values vary. For nested applications where the required time step Δt is much smaller than that used in the outer simulation that produces the boundary values, this error is likely to be negligible.

Figure 11 shows the result of using this procedure with $\tau = 3\Delta t$ and $\sigma = 5\Delta x$. This approach has both advantages and disadvantages compared with the direct insertion method of the previous section. The smoothness of $u^*(x)$ near the boundaries produces a solution that requires less viscous damping and less aggressive wavenumber filtering to maintain spatially smooth computed solutions. This is a very nice feature because, in practice, we often wish to use higher order dissipation operators and these may be sensitive to small scale noise. The solution shown in Figure 11 was computed with $\nu = 0$ and a wavenumber filtering fraction of 0.05. On the other hand, the solution is only within about $\Delta t = .01$ of the exact solution $u = 1$. For the number of discrete time steps $nt = 10^2, 10^3, 10^4$, the error after integrating over one period T is .01, .003 and .0003 respectively.

Investigated but not explicitly shown here:

- an exponent of 4 in the windowing functions works better than 2

- some wavenumber filtering is beneficial, though less is needed than for the previous approach
- wider windowing functions (*eg.* $\sigma = 10\Delta x$) produce better spectral behavior and smoother solutions (by eye, they all look smooth though)
- $\nu = 0$ and a filtering fraction of 0.05 results in a slightly noisier solution after 100 steps using the direct insertion approach but the mean absolute error is only 1×10^{-6}, which is much much smaller than the 0.01 error in the nudging approach.