Chorin's Projection Method with Spectral Collocation for 2D Incompressible Navier Stokes

August 25, 2019

The setup and time discretization with Crank Nicholson remains the same as the finite difference version of this (see other directory in repo). Here, we focus on deriving a Chebyshev pseudo-spectral estimator.

Recall the NSE (with no force function)

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p &= \Delta \mathbf{u} \\ \nabla \cdot \mathbf{u} &= 0 \\ \mathbf{u} = 0 \quad \text{on} \quad \partial\Omega \end{aligned}$$

Chorin: Step 1 Also recall that the Chorin's projection method first ignores pressure to comute an intermediate velocity field

$$\begin{aligned} \frac{\partial \mathbf{u}^*}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} &= \Delta \mathbf{u}^* \\ \mathbf{u}^* = 0 \quad \text{on} \quad \partial\Omega \end{aligned}$$

We discretize time first with Adams-Bashford and implicit Crank-Nicholson

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\triangle t} + (\mathbf{u}^{\frac{n+1}{2}} \cdot \nabla)\mathbf{u}^{\frac{n+1}{2}} = \Delta \mathbf{u}^*$$

So far things are identical to the finite difference version of Chorin's. But to solve this, we do not discretize spatial dimensions anymore: we use (pseudo)spectral methods:

We want to approximate the solution $\mathbf{u}^*$ as truncated series of Chebyshev polynomials: $\{T_k(x)\}_{k=1}^{\infty}$ where $T_k(x) = \cos(k \cos^{-1} x)$; each polynomial is restricted to $[-1, 1]$.

$$u_N^*(x) = \sum_{k=0}^{\infty} \hat{u}_k^* T_k(x) \approx \sum_{k=0}^{N} \hat{u}_k^* T_k(x)$$

where $\mathbf{u}^* = (u^*, v^*)$. We just write the derivation for the first equation for simplicity. To get the Chebyshev coefficients, we have to compute the (normalized) inner product:

$$\hat{u}_k = \frac{2}{\pi c_k} \int_{-1}^{1} u T_k w dx$$

where $c_k = \begin{Bmatrix} 2 & \text{if} & k = 0 \\ 1 & \text{if} & k \geq 1 \end{Bmatrix}$. This is hard to compute, so we need to estimate the integral. Normally, this would require a lot of points but we can get away with carefully selected ones and a Gaussian quadrature. The best points turn out to be the roots of another Chebyshev polynomial with degree one higher; these are called the Gauss-Lobatto points – they end up with denser spread near the boundaries (-1 and 1).

$$x_i = \cos \frac{\pi i}{N}, i = 0, ..., N \tag{1}$$

resulting in,

$$\hat{u}_k = \frac{2}{\bar{c}_k N} \sum_{i=0}^{N} \frac{1}{\bar{c}_i} u_i T_k(x_i), k = 0, ..., N \tag{2}$$

where $\bar{c}_k = \begin{Bmatrix} 2 & & \text{if} & k = 0 \\ 1 & \text{if} & 1 \leq k \leq N-1 \\ 2 & & \text{if} & k = N \end{Bmatrix}$. Note that to convert back and forth between spectral coefficients ($\hat{u}_k$) and the values at the collocation points ($u_N(x_i)$) is a matrix multiplication. To be explicit, let $\mathcal{T} = [cosk\pi i/N], k, i = 0, ..., N$ and $\mathcal{T}^{-1} = [2(\cos \pi i/N)/(\bar{c}_k \bar{c}_i N)]$. Then,

$$\mathcal{U}^* = \mathcal{T}\hat{\mathcal{U}}^*$$
$$\hat{\mathcal{U}}^* = \mathcal{T}\mathcal{U}^*$$

where $\mathcal{U}^* = [u^*(x_0), ..., u^*(x_N)]$ and $\hat{\mathcal{U}}^* = [\hat{u}_0, ..., \hat{u}_N]$.

An interesting fact that is useful is that the approximation:

$$u_N^*(x) = \sum_{k=0}^{N} \hat{u}_k^* T_k(x) \tag{3}$$

can be viewed as a Lagrange interpolating polynomial with a set $\{x_i\}$. One can explicitly write this as

$$u_N^*(x) = \sum_{j=0}^{N} h_j(x) u^*(x_j) \tag{4}$$

where $h_j(x) = \frac{(-1)^{j+1}(1-x^2)T_N'(x)}{\bar{c}_j N^2 (x - x_j)}$ is some crazy polynomial. This is really useful because it lets us do differentiation in closed form in physical space (no need for fourier transforms). In particular, we can write the p-th derivative,

$$u_N^{*,(p)}(x_i) = \sum_{k=0}^{N} \hat{u}_k T_k^{(p)}(x_i) = \sum_{j=0}^{N} h_j^{(p)}(x_i) u_N(x_j) \tag{5}$$

Notice then that computing the derivative is just a matrix multiplication on the existing coordinate values! If we let $d_{i,j}^{(p)} = h_j^{(p)}(x_i)$, then we can get a series of formulas populating a "derivative matrix", $\mathcal{D} = [d_{i,j}^{(1)}], i, j = 0, ..., N$.

$$d_{i,j}^{(1)} = \frac{\bar{c}_i}{\bar{c}_j} \frac{(-1)^{i+j}}{(x_i - x_j)}, 0 \leq i, j \leq N, i \neq j$$
$$d_{i,i}^{(1)} = -\frac{x_i}{2(1 - x_i^2)}, 1 \leq i \leq N - 1$$
$$d_{0,0}^{(1)} = -d_{N,N}^{(1)} = \frac{2N^2 + 1}{6}$$

So, in short, $\mathcal{U}^{*,(1)} = \mathcal{D}\mathcal{U}^*$, $\mathcal{U}^{*,(2)} = \mathcal{D}^2\mathcal{U}^*$. As nice as this is,there are a few hacks we need to be careful of for numerical stability.

First, calculating $(1 - x_i^2)$ and $(x_i - x_j)$ may be hard if points are very close, so we use the following:

$$x_i - x_j = 2 \sin \frac{(j + i)\pi}{2N} \sin \frac{(j - i)\pi}{2N}$$
$$1 - x_i^2 = \sin^2 \frac{i\pi}{N}$$

Second, since this differentiation matrix is approximated, it doesn't always represent the derivative of a constant (which it should). In other words, it should be that

$$\sum_{j=0}^{N} d_{i,i}^{(1)} = 0, i = 0, ..., N \tag{6}$$

. To fix this, we should calculate the off diagonal entries later to satisfy this constraint. So...

$$d_{i,j}^{(1)} = -\sum_{j=0, j\neq i}^{N} d_{i,j}^{(1)}, i = 0, ..., N \tag{7}$$

When we want to compute $\mathcal{D}$ and $\mathcal{D}^2$, we use the following procedure:

- compute $\mathcal{D}$ with the diagonal term correction
- take square product to get a provisional $\tilde{\mathcal{D}}^2$.
- correct $\tilde{\mathcal{D}}^2$ to nsum to 1. As in first set $d_{i,j}^{(2)} = \tilde{d}_{i,j}^{(2)}$ for $j \neq i$. Then set $d_{i,i}^{(2)} = -\sum_{j=0, j\neq i}^{N} \tilde{d}_{i,j}^{(2)}, i = 0, ..., N$.

Ok, now that we know how to differentiate, the next challenge is generalizing to two dimensions. In 2D, you use two sets of Chebyshev polynomials (with degree N_x in the x-direction and degree N_y for the y-direction). Instead of a grid, you get a mesh: $\bar{\Omega}_N$ where

$$x_i = \cos \frac{\pi i}{N_x}, i = 0, ..., N_x$$
$$y_j = \cos \frac{\pi j}{N_y}, i = 0, ..., N_y$$

. Let Ω_N be the open mesh ($1...N_x - 1$ and $1...N_y - 1$). The $\bar{\Omega}_N^I$ be the mesh without the four corners. Now we write the pressure-free equation in terms of the collocation points.

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\triangle t} + \frac{3}{2}(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n - \frac{1}{2}(\mathbf{u}^{n-1} \cdot \nabla)\mathbf{u}^{n-1} - \Delta(\frac{\mathbf{u}^* + \mathbf{u}^n}{2}) = 0$$
$$\mathbf{u}^* = 0$$

Rearrange to

$$\frac{\mathbf{u}^*}{\triangle t} - \frac{\Delta \mathbf{u}^*}{2} = \frac{\mathbf{u}^n}{\triangle t} - \frac{3}{2}(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \frac{1}{2}(\mathbf{u}^{n-1} \cdot \nabla)\mathbf{u}^{n-1} + \frac{\Delta \mathbf{u}^n}{2}$$
$$2\mathbf{u}^* - \triangle t \Delta \mathbf{u}^* = 2\mathbf{u}^n - 3 \triangle t(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \triangle t(\mathbf{u}^{n-1} \cdot \nabla)\mathbf{u}^{n-1} + \triangle t \Delta \mathbf{u}^n$$
$$2\mathbf{u}^* - \triangle t \Delta \mathbf{u}^* = \mathbf{F}$$

where $\mathbf{F} = [f, g]$ represents all the known variables.

We can write out the dimensions,

$$2u^* - \triangle t \partial_{xx} u^* - \triangle t \partial_{yy} u^* = f$$
$$2v^* - \triangle t \partial_{xx} v^* - \triangle t \partial_{yy} v^* = h$$

We can actually consider general robin boundary conditions (not just dirichlet)... we just write out the first dimension for now:

$$\alpha_- u^*(-1) + \beta_- u^{*,(1)}(-1) = b_-$$
$$\alpha_+ u^*(1) + \beta_+ u^{*,(1)}(1) = b_+$$

Then i can write this in "matrix form" with respect to the mesh, $\bar{\Omega}$,

$$2\mathcal{U}^* - \triangle t \mathcal{D}_x \mathcal{U}^* - \triangle t \mathcal{U}^* \mathcal{D}_y^T = \mathcal{F} \tag{8}$$

where $\mathcal{U}^* = [u_N(x_i, y_j)]$ for $i = 1, ..., N_x - 1$ and $j = 1, ..., N_y - 1$. $\mathcal{D}_x = [d_{i,j}]$ where

$$d_{i,j} = d_{i,j}^{(2)} + \frac{1}{e}(b_{0,j} d_{i,0}^{(2)} + b_{N,j} d_{i,N}^{(2)})$$
$$e = c_{0,+} c_{N,-} - c_{0,-} c_{N+}$$
$$c_{0,-} = -\beta_+ d_{0,N}^{(1)}$$
$$c_{0,+} = \alpha_- + \beta_- d_{N,N}^{(1)}$$
$$c_{N,+} = -\beta_- d_{N,0}^{(1)}$$
$$c_{N,-} = \alpha_+ + \beta_+ d_{0,0}^{(1)}$$
$$b_{0,j} = -c_{0,+} \beta_+ d_{0,j}^{(1)} - c_{0,-} \beta_- d_{N,j}^{(1)}, j = 1...N-1$$
$$b_{N,j} = -c_{N,-} \beta_- d_{N,j}^{(1)} - c_{N,+} \beta_+ d_{N,j}^{(1)}, j = 1...N-1$$

Note that $\mathcal{D}_x$ builds in the boundary conditions. We can similarly design $\mathcal{D}_y$. Note that for v^* we would need completely new differential matrices for its initial conditions.

Now we need to solve Equation 8. We do this with matrix diagonalization. Pay upfront cost to diagonalize the differential matrices:

$$\mathcal{D}_x = \mathcal{P}\Lambda_x\mathcal{P}^{-1}$$
$$\mathcal{D}_y = \mathcal{Q}\Lambda_y\mathcal{Q}^{-1}$$

Then for each iteration, do the following. First left multiply everything by $\mathcal{P}^{-1}$.

$$2\tilde{\mathcal{U}}^* - \triangle t\mathcal{P}^{-1}\mathcal{D}_x\mathcal{P}\tilde{\mathcal{U}}^* - \triangle t\tilde{\mathcal{U}}^*\mathcal{D}_y^T = \tilde{\mathcal{F}}$$
$$2\tilde{\mathcal{U}}^* - \triangle t\Lambda_x\tilde{\mathcal{U}}^* - \triangle t\tilde{\mathcal{U}}^*\mathcal{D}_y^T = \tilde{\mathcal{F}}$$

where $\tilde{\mathcal{U}}^* = \mathcal{P}^{-1}\mathcal{U}^*$ and $\tilde{\mathcal{F}} = \mathcal{P}^{-1}\mathcal{F}$. Then, right multiply everything by $(\mathcal{Q}^T)^{-1}$.

$$2\hat{\mathcal{U}}^* - \triangle t\Lambda_x\hat{\mathcal{U}}^* - \triangle t\hat{\mathcal{U}}^*\mathcal{Q}^T\mathcal{D}_y^T(\mathcal{Q}^T)^{-1} = \hat{\mathcal{F}}$$
$$2\hat{\mathcal{U}}^* - \triangle t\Lambda_x\hat{\mathcal{U}}^* - \triangle t\hat{\mathcal{U}}^*\Lambda_y = \hat{\mathcal{F}}$$

where $\hat{\mathcal{U}}^* = \mathcal{U}^*(\mathcal{Q}^T)^{-1}$ and $\hat{\mathcal{F}} = \mathcal{F}(\mathcal{Q}^T)^{-1}$. We can then compute

$$\hat{u}^*_{i,j} = \frac{\hat{f}_{i,j}}{2 - \triangle t\lambda_{x,i} - \triangle t\lambda_{y,j}}, i = 1...N_x - 1, j = 1...N_y - 1 \tag{9}$$

If $\lambda_{x,i} = \lambda_{y,j} = 0$, replace $\hat{u}^*_{i,j} = 0$. Now that we have computed $\hat{\mathcal{U}}^* = [\hat{u}^*_{i,j}]$, we then $\tilde{\mathcal{U}} = \hat{\mathcal{U}}^*\mathcal{Q}^T$ and calculate $\mathcal{U}^* = \mathcal{P}\hat{\mathcal{U}}^*$. Remember that $\mathcal{U}$ is size $N_x - 2 \times N_y - 2$. Now we need to compute boundary values. This is done by representing all derivatives in matrix form and solving the linear equation. If we do the same for the second dimension of velocity, we now have the solution for $\mathbf{u}^*$.

Chorin: Step 2 We need to correct the intermediate velocity field. There are two ways to this specific to spectral methods: we can either parameterize pressure with a polynomial of the same degree as velocity (and assume a Neumann boundary condition) or a polynomial with two degrees lower.

Recall the remaining component of Chorin's:

$$\partial_t\mathbf{u} = -\frac{1}{\rho}\nabla p \tag{10}$$

which can be rewritten as

$$\frac{\rho}{\triangle t}(\mathbf{u}^{n+1} - \mathbf{u}^*) + \nabla p^{n+1} = 0$$
$$\nabla \cdot \mathbf{u}^{n+1} = 0 \quad \text{in} \quad \Omega$$
$$\mathbf{u}_N^{n+1} \cdot \mathbf{n} = 0$$

There are two ways to do this that involve different assumptions and polynomials of different degrees. We pick the one that assumes NO boundary conditions on pressure.

$\mathbb{P}_N - \mathbb{P}_{N-2}$ **projection method** Let $\mathbb{P}_N$ be the space of polynomials with degree N_x in x and N_y in y – we used this to approximate velocity: $\mathbf{u}_N^* = (u_N^*, v_N^*)$ and $\mathbf{u}_N^{n+1} = (u_N^{n+1}, v_N^{n+1})$. We then approximate pressure with a polynomial of degree two less in x and y i.e. $p^{n+1} \approx p_{N-2}^{n+1} \in \mathbb{P}_{N-2}$. But! we still evaluate at the same Gauss-Lobatto points as for velocity. So we have to define a Lagrange polynomial of degree $N_x - 2$ with roots of a $N_x + 1$ Chebyshev polynomial... more special formulas!

There is a relation between the Lagrange polynomial used here, $\hat{h}_j$ with the one used for velocity, h_j:

$$\hat{h}_j(x) = \frac{1 - x_j^2}{1 - x^2} h_j(x) \tag{11}$$

. This gives us definitions for the differential matrices:

$$\hat{d}_{i,j}^{(1)} = \frac{(-1)^{j+1}(1 - x_j^2)}{(1 - x_i^2)(x_i - x_j)}, i, j = 1...N-1, i \neq j$$
$$\hat{d}_{i,i}^{(1)} = \frac{3x_i}{2(1 - x_i^2)}, i = 1...N-1$$

We emphasize these are unique to the $N - 2$ class of polynomials and should only be used for pressure. The system can then we written as:

$$\frac{\rho}{\triangle t}\mathcal{U} + \hat{\mathcal{D}}_x \mathcal{P} = \frac{\rho}{\triangle t}\mathcal{U}^*$$
$$\frac{\rho}{\triangle t}\mathcal{V} + \mathcal{P}\hat{\mathcal{D}}_y^T = \frac{\rho}{\triangle t}\mathcal{V}^*$$
$$\mathcal{D}_x \mathcal{U} + \mathcal{V}\mathcal{D}_y^T = \mathcal{S}$$

where $\mathcal{U} = [u_N^{n+1(x_i,y_j)}]$, $\mathcal{V} = [v_N^{n+1}(x_i, y_j)]$, and $\mathcal{P} = [p_{N-2}^{n+1}(x_i, y_j)]$ for $i = 1...N_x - 1$ and $j = 1...N_y - 1$. Note which differentiation matrices have hats. $\mathcal{S} = -(\bar{\mathcal{D}}_x \mathcal{U}_\Gamma + \mathcal{V}_\Gamma \bar{\mathcal{D}}_y^T)$ contain info about the boundary conditions of velocity and $\bar{\mathcal{D}}_x = [d_{i,j}^{(1)}]$ for $i = 1...N_x - 1, j = 0, N_x$ and $\bar{\mathcal{D}}_y = [d_{i,j}^{(1)}]$ for $i = 1...N_y - 1, j = 0, N_y$ and $\mathcal{U}_\Gamma = [u_\Gamma^{n+1}(x_i, y_j)]$ for $i = 0, N_x$ and $j = 1...N_y - 1$ and $\mathcal{V}_\Gamma = [v_\Gamma^{n+1}(x_i, y_j)]$ for $i = 1...N_x - 1$, $j = 0, N_y$.

Then the solution (after eliminating $\mathcal{U}$ and $\mathcal{V}$):

$$\mathcal{D}_x \hat{\mathcal{D}}_x \mathcal{Q} + \mathcal{Q}(\mathcal{D}_y \hat{\mathcal{D}}_y)^T = -\frac{\rho}{\triangle t}(\mathcal{S} - \mathcal{D}_x \mathcal{U}^* - \mathcal{V}^* \mathcal{D}_y^T) \tag{12}$$

We can solve this by matrix-diagonalization.