TAM570: Spectral Methods for CFD
(Draft Notes, Fall 2023)

Contents

1 Scale Challenges for Thermal-Fluid Transport

In this course, we focus on the theory and development of *spectral* and multi-domain spectral methods for the simulation of incompressible fluid flow and heat transfer. We use theory and examples to illustrate why these approaches are among the most effective tools available for solution of the incompressible Navier-Stokes equations,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u}, + \mathbf{f}, \tag{1}$$

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

which governs the velocity, $\mathbf{u}$, and pressure, p, subjection to suitable boundary and initial conditions in a domain Ω. We will also apply these methods to a similar set of equations for thermal transport, which will be introduced below.

Turbulence presents a major challenge in fluid/thermal simulation because, at elevated Reynolds numbers, it leads to rapid high-wavenumber fluctuations in the velocity and pressure fields. Invariably, we are not concerned with particular details of the turbulent motion because the solutions are chaotic. They are so highly sensitive to initial conditions or other perturbations as to make precise reconstruction of any single trajectory impossible and one can hope only to recover the correct mean (time- or space-averaged) behavior, including perhaps higher-order correlations.

Several examples of turbulent flow are illustrated in Fig. 1. Figures 1 (a) and (b) show flows in Earth's atmosphere and in the Jovian atmosphere. In these images, all visible scales are quite large; the turbulent scales continue to cascade down to well below the resolution of the image. Despite the clear presence of chaotic motion at all scales, some familiar structures are evident at the largest scales of motion. In Fig. 1(a), we see three von Karman vortex streets forming in the wake of the Gaudaloupe islands. Although the background flow is fully turbulent and three-dimensional, the vortex street itself is essentially a two-dimensional flow phenomenon, analyzed by Theodore von Karman in 1911 [**?**, **?**]. (The lower street is disrupted, most likely due to the presence of multiple islands that are in line with the flow direction.) Figure 1(b) shows the most famous feature of the Jovian atmosphere, the Great Red Spot, which has persisted for hundreds of years. This phenomenon has been studied extensively by Phil Marcus and is discussed in an excellent review article [**?**]. Both of these cases illustrate the emergence of stable, persistent, large-scale structures in the presence of small-scale chaotic motion. The ability to understand and predict these larger-scale structures, which often dominate physical phenomena such as momentum and thermal transport, is one of the primary objectives of the study of turbulence.

Much smaller ranges of scales are exhibited in the turbulent flow simulations over wing sections in Fig. 1(c) and in the intake stroke of an internal combustion engine (ICE) in Fig. 1(d). In the ICE example, one finds near the top of the intake port horseshoe vortices wrapped around the valve stem that break down into hairpin vortices in the wake of the valve stem. After passing through the valve opening the flow becomes quite chaotic with no discernable pattern. In the wing-section examples we see transition from laminar flow to turbulence (induced by tripping), followed by a growth in the boundary layer thickness. The transition process includes formation of hairpin vortices that eventually break up into tangles of vortex filaments. As the Reynolds number (based on the chord length) increases from $Re = 10^5$ to 10^6 the hairpin vortices diminish in size and become more numerous.

Clearly, higher Reynolds number flows exhibit more scales of motion. In a numerical setting, this presents a challenge if one hopes to resolve all scales. At a minimum, one needs two grid points in each direction to resolve an individual eddy. In practice, five to ten points in each direction is a

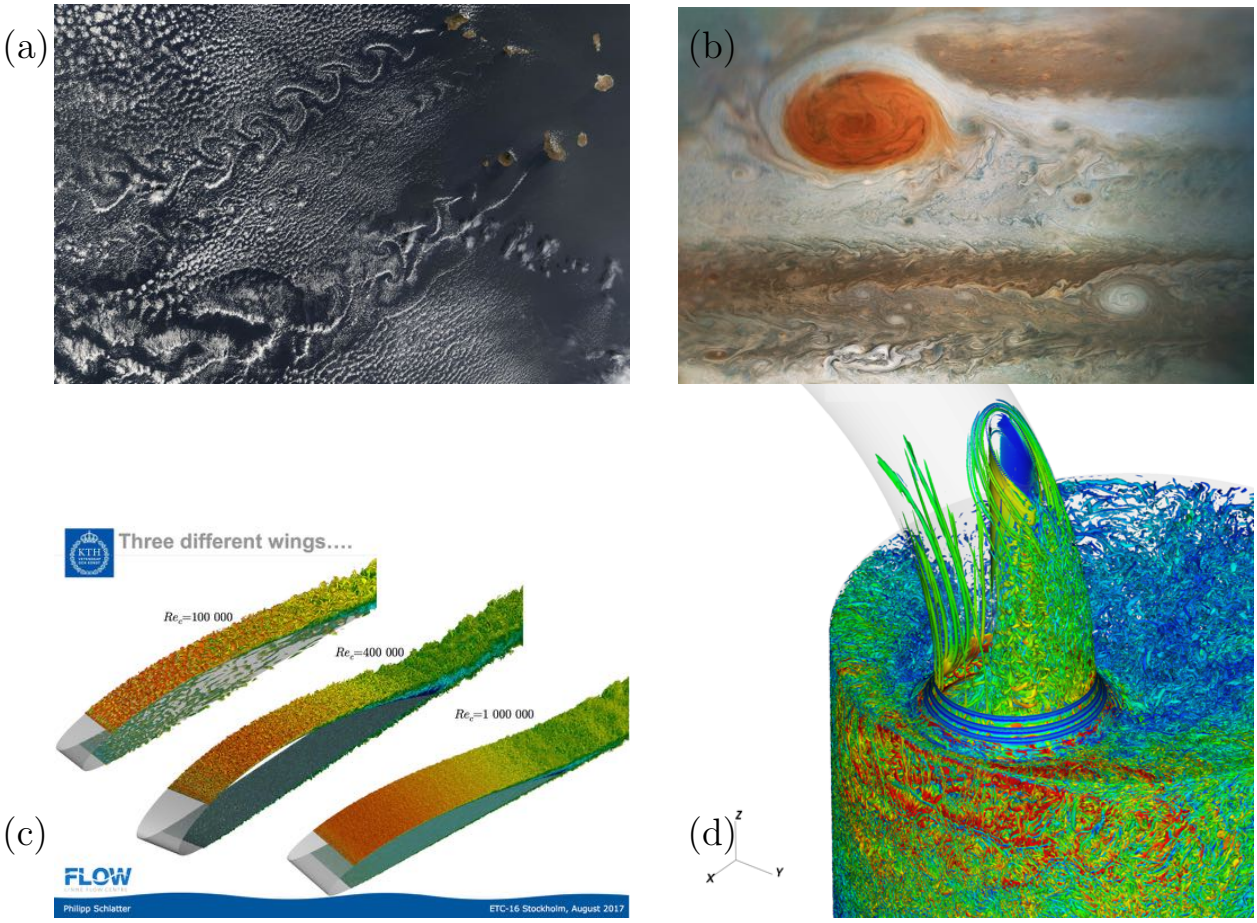


Figure 1: Examples of turbulent flow with a large range of scales: (a) von Karman vortex streets in the wake of the Gaudeloupe Islands; (b) turbulent bands and the Great Red Spot in the Jovian atmosphere; (c) turbulence on a NACA 4014 wing section (P. Schlatter group, KTH, 2017); (d) turbulence in the intake stroke of an IC engine (G. Giannakopoulos, ETHZ, 2018).

more realistic estimate of the minimal resolution requirements to resolve the smallest scale eddies and to transport them via numerical advection because accurate representation of the 1D advection operator requires at least five points per wavelength unless one is using a Fourier basis. With such an estimate in mind, we could ask if there is a way to predict the range of scales in a turbulent flow without having to first simulate it. The answer depends very much on the geometry, but a general rule of thumb is that if L represents the domain size and λ represents the size of the smallest eddy then the relationship of these sizes for a fully turbulent flow is

$$\frac{L}{\lambda} \quad \sim \quad C\, Re^{\frac{3}{4}}, \tag{3}$$

where $Re = UL/\nu$ is the Reynolds number based on a characteristic flow speed U, length scale, L, and kinematic viscosity ν. Because the total number of gridpoints, n, scales like the cube of this quantity, we have

$$n \quad \sim \quad C\, Re^{\frac{9}{4}}. \tag{4}$$

In both equations, C is a (different) flow-dependent constant.

In addition to an increase in the number of spatial points, one must typically also increase the number of time steps to simulate higher Reynolds number flows. An order-of-magnitude estimate is that the number of time steps must scale at least as

$$n_t \sim \frac{L}{\lambda} \tag{5}$$

because one must advance the small-scale structures of size λ over the length of the domain, L. Numerical stability and accuracy requirements dictate that significant features can not propagate more than one grid cell per timestep. (This is the famous Courant-Friedrichs-Lewy, or CFL, stability criterion.) Moreover, one frequently needs much longer times than indicated by the estimate (5) in order to obtain converged turbulent statistics. In any case, at a minimum, the simulation cost estimate in terms of degrees-of-freedom $\times$ number of time steps is

$$\text{work: } W \sim C n_t n = C R e^3. \tag{6}$$

An important consideration in developing a numerical method for this class of problems is that the integration times are long—one must propagate small-scale structures over the full domain length. (As we will see, at high Reynolds numbers, flow structures do not rapidly dissipate.) For most numerical methods, the discrete approximation to the advection operator does not propagate all wave numbers correctly. There is typically a dispersion error that increases linearly with time and at least linearly with wavenumber: low wavenumber structures, corresponding to long wavelengths, are propagated accurately; high wavenumber structures, corresponding to short wavelengths, are propagated inaccurately. If there is an error in the propagation speed at wavenumber k, then those waves will lag (or advance) and this error will be compounded as the simulation continues. Consequently, for *long time integration*, there is a premium on minimizing numerical dispersion and dissipation of the advection operator.

1.1 Spectral Methods

The rapid convergence of spectral methods, which exhibit exponential convergence with the approximation order, N, makes them ideally suited for simulating minimally-dissipative transport problems such as direct numerical simulation (DNS) and large eddy simulation (LES) of turbulence. In particular, for periodic domains, *Fourier* spectral methods have no numerical dispersion in the periodic direction(s) for the *resolved* wavenumbers. Note that this remarkable fact does not imply that spectral methods are error free—one still must resolve the significant waves. According to the Nyquist sampling criterion, if a simulation has N points in a given direction, the highest wavenumber that can be resolved in that direction is $k = N/2$. High-order methods to not circumvent this requirement.

In the case of Fourier methods in homogeneous domains, the principal error is thus contained in the modes that are *not* resolved. If we consider a model 1D problem (the topic of Section 4) with an exact periodic solution on $\Omega = [0, 2\pi]$ given by,

$$\tilde{u}(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k \, e^{ikx}, \tag{7}$$

then a reasonable discrete approximation would be given by its projection onto the lowest N modes. Owing to the orthogonality of the Fourier basis, $\phi_k(x) = e^{ikx}$, this projection is

$$u_N(x) = \sum_{k=-N/2}^{N/2-1} \hat{u}_k \, e^{ikx}, \tag{8}$$

and the corresponding error is

$$e_N(x) \;:=\; \tilde{u}(x) - u_N(x) \;=\; \sum_{|k|\gtrsim N/2} \hat{u}_k\, e^{ikx}, \tag{9}$$

where the summation range is taken to be the complement of that in (8). We see that the magnitude of the error depends on how rapidly the Fourier coefficients of $\tilde{u}$ decay for $|k| > N/2$. In particular, if $\tilde{u}$ is *smooth* for some $k > k_c$, we will show that the error decays exponentially fast with the approximation order, N. That is,

$$\|e_N(x)\| \;\sim\; Ce^{-\sigma N} \qquad \textit{exponential convergence}, \tag{10}$$

for some appropriate norm $\|\cdot\|$ and pair of positive constants C and σ. This rate of convergence is to be contrasted with kth-order algebraic convergence of the form

$$\|e_N(x)\| \;\sim\; CN^{-k} \qquad \textit{algebraic convergence}, \tag{11}$$

for a fixed approximation order k. (Second-order methods such as linear finite elements correspond to k=2.) If u has limited regularity, meaning that it only has s bounded derivatives on Ω, then the error behavior of the spectral method will also be algebraic, with exponent s in (11) rather than k.

The smallest scales of turbulence are dominated by viscous dissipation. Below this scale there are no turning eddies and the solution is smooth. It is this lower bound on the eddy size that determines the grid resolution requirements for DNS (where no modeling is used). In turbulence theory, this is referred to as the Kolmogorov length scale. If a simulation has a grid spacing that is smaller than the Kolmogorov scale it is said to be resolved. Note that "smaller" has different meanings for different numerical methods. A low-order method will require a much finer mesh than a high-order one because features *at the grid scale* are not propagated accurately. In 3D, a finer mesh, in each direction, leads to a signficant computational overhead.

1.2 Historical Costs

We close out this introduction with some historical data about the cost of turbulent flow simulations. In his seminal 1980 paper on spectral methods for complex domains, Steve Orszag presented solution times (in wall-clock seconds-per-step) for his highly efficient Centicube code, which was based on 3D Fourier expansions for simulation of turbulence in a box. More recently, P.K. Yeung and collaborators have published a paper on this same approach but with significantly increased resolution.

- 1980: Centicube — $128^3 \;=\; 2.1 \times 10^6$. 20 seconds/step on Cray-1 (10 MFLOPS).
- 2019: millicube — $18432^3 \;=\; 6.3 \times 10^{12}$. 14.24 seconds/step on Summit (18432 Nvidia V100s).

Note: you need *many* more steps for 18432^3 than for 128^3—roughly a factor of 144.

1.3 Fourier Orthogonality in 1D

To discuss orthogonality, we need an inner product and a norm. For the orthogonality of the Fourier bases the L^2 norm is appropriate. We say that a function $f(x)$ is in $L^2(\Omega)$ if

$$\|f\|_2 \;:=\; \left[\int_\Omega f^2\,dV\right]^{\frac{1}{2}} < \infty. \tag{12}$$

The associated inner product is

$$(f,g) \;:=\; \int_\Omega f\,g\,dV, \tag{13}$$

and two functions f and g are (L^2-) orthogonal if $(f,g)=0$. If f and g are complex, we define the inner product as

$$(f,g) \;:=\; \int_\Omega f^*\,g\,dV, \tag{14}$$

where f^* is the complex conjugate of f.

Consider $\Omega=[0:2\pi]$, $f=e^{ikx}$, and $g=e^{ilx}$ with $i:=\sqrt{-1}$ and k and l integers.

$$(f,g) \;:=\; \int_\Omega e^{-ikx}e^{ilx}\,dx \;=\; \int_\Omega e^{i(l-k)x}\,dx \;=\; \begin{cases} 2\pi, & \text{if } k=l,\\ 0, & \text{if } k\neq l. \end{cases} \tag{15}$$

Consider now the inner product of e^{ilx} with the error formula (9).

$$(e_N,e^{ilx}) \;=\; (e^{ilx},e_N) \;:=\; \int_\Omega e^{-ilx}\left[\sum_{|k|>N/2}\hat{u}_k\,e^{ikx},\right]dx \;=\; \begin{cases} 2\pi\hat{u}_l, & \text{if } |l|>N/2,\\ 0, & \text{if } |l|\leq N/2. \end{cases} \tag{16}$$

Here, for simplicity, we define the Fourier approximation space as X^N :=span$\{e^{-ixN/2},\ldots,e^{ixN/2}\}$. From (16), we see that $(e_N,v)=0$ for all $v\in X^N$. That is, the error is orthogonal to X^N.

Exercise. Suppose that X^N is a finite-dimensional space of the type defined above and that u_N is chosen such that $e_N:=\tilde{u}-u_N\perp X_N$. Show that u_N is the *closest* element in X^N to $\tilde{u}$ in the following sense:

$$u_N \;=\; \underset{v\in X^N}{\operatorname{argmin}}\left(\|v-\tilde{u}\|_2\right)^2. \tag{17}$$

Hint: Set $v=u_N-w\in X^N$ for any $w\in X^N$ and substitute into (17).

1.4 Resolution vs. Spectral Convergence

The preceding example points to a key concept throughout this course, namely generation of the *best approximation* in a given trial space through the use of *projection*. Integration is central to the application of projection operators in numerical solution of PDEs so we will use that operator here to illustrate differences between *resolution* and *convergence*. Specifically (using `trap2.m`), we show that high-order (i.e., spectral) methods have significant benefits with respect to convergence but they cannot overcome fundamental resolution requirements. Even the most accurate method must sample (or *resolve*) the solution before convergence is realized. To this end, consider the following integral,

$$\mathcal{I} \;:=\; \int_0^1 f(x)\,dx \tag{18}$$

with $f(x) := \sin(k\pi x)$. In the first instance, we use the trapezoid rule to approximate $\mathcal{I}$,

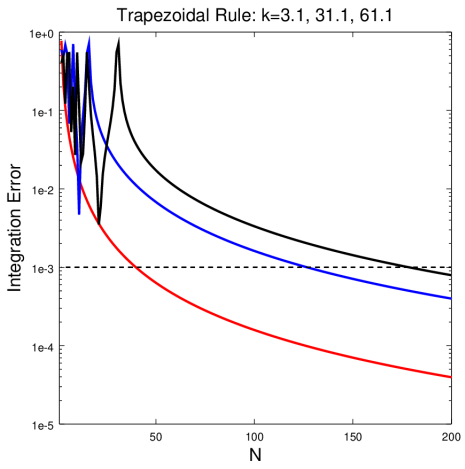
$$\mathcal{I} \;\approx\; \mathcal{I}_T \;:=\; \sum_{j=0}^{N} \sigma_j f(j\cdot h) \tag{19}$$

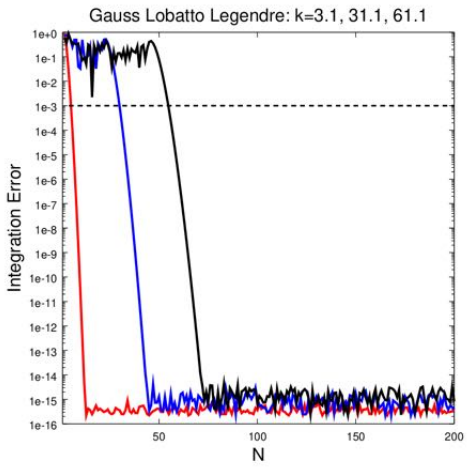
$\sigma_j = h$, $j = 1,\ldots,N-1$ and $\sigma_0 = \sigma_N = h/2$ with $h = 1/N$. As a second example, we consider Gauss-Lobatto-Legendre (GLL) quadrature,

$$\mathcal{I} \;\approx\; \mathcal{I}_{GLL} \;:=\; \sum_{j=0}^{N} w_j f(x_j), \tag{20}$$

where $x_j = (\xi_j + 1)/2$ are the GLL quadrature points, ξ_j translated and scaled from $[-1,1]$ to the interval $[0,1]$, and $w_j = \rho_j/2$ are the GLL quadrature weights, ρ_j, rescaled to $[0,1]$.

In the figures below, we plot the error for trapezoidal-based quadrature on the left and GLL quadrature on the right, as a function of N, for $k = 3.1$ (red), 31.1 (blue), and 61.1 (black). For the trapezoid case, which in this case converges as $O(N^{-2})$, we see that almost 180 points are required for an error of 10^{-3} when $k = 61.1$, whereas GLL reaches this error tolerance at $N \approx 60$. Moreover, GLL is able to reach close to machine precision ($\approx 10^{-16}$) with roughly 80 points. The trapezoid rule would require $\approx 10^8$ points to reach the same limit. For $k = 3.1$, GLL requires $N = 5$ to reach 10^{-3}, whereas the trapezoid rule needs $N = 40$. On the other hand, *neither algorithm converges* when $N < 50$ for the case $k = 61.1$, which corresponds to a highly oscillatory integrand. This observation is a manifestation of the Nyquist sampling theorem: *Every numerical scheme needs to resolve the solution before convergence (algebraic or exponential) is realized.*





1.5 Quadrature Demo: Code

The matlab script below illustrates the rapid convergence of GLL quadrature. The integral is

$$\mathcal{I} \;=\; \int_0^1 \sin(k\pi x)\,dx \;=\; \frac{1-\cos k\pi}{k\pi}, \tag{21}$$

where k needn't be an integer.

```
%-------------------------------------------------------
lw='linewidth';               %%% Plotting defs
fs='fontsize';                %%% Plotting defs
intp = 'interpreter';         %%% Plotting defs
ltx  = 'latex';               %%% Plotting defs
format compact;

% \int_0^1 sin(k pi x) dx = (1-cos(k pi ))/(k pi)

I_exact = (1-cos(k*pi))/(k*pi);

hold off
n=200+8*ceil(k); x=(0:n)/n; f=sin(k*pi*x); plot(x,f,'k-',lw,1.1);

for n=2:500;                          %  TRAPEZOIDAL RULE TEST
   x=(0:n)/n; f=sin(k*pi*x);
   I_trap(n-1) = ( sum(f(2:n)) + .5*(f(1)+f(n+1)) ) / n;
   e_trap(n-1) = abs(I_exact - I_trap(n-1));
   n_trap(n-1) = n;
   if n==20; hold on;
      plot(x,f,'r+',lw,2);
      title('Trapezoidal Rule, N=20','FontSize',16);
      xlabel('x','FontSize',16);
      ylabel('f(x)','FontSize',16);
      pause;
   end;
end;

for N=2:400;                          %  GLL TEST
   [z,w] = zwgll(N); x = .5*(z+1);
   f = sin(k*pi*x);
   I_gll(N-1)=.5*sum(w.*f);
   e_gll(N-1)=abs(I_exact-I_gll(N-1))+eps;
   n_gll(N-1)=N;
   if N==20; hold on;
      plot(x,f,'bx',lw,2);
      title('GLL Quadrature, N=20','FontSize',16);
      xlabel('x','FontSize',16);
      ylabel('f(x)','FontSize',16);
      hold off; pause;
   end;
end;

loglog(n_trap,e_trap,'ro-'); axis square;
title('Trapezoidal Rule Convergence: \int sin(k \pi x)','FontSize',16)
xlabel('N','FontSize',16); ylabel('Integration Error','FontSize',16);
legend('Composite Trapezoidal Rule','location','southwest'); pause

loglog(n_gll,e_gll,'bo-',n_trap,e_trap,'ro-'); axis square;
title('GLL and Trapezoidal Rule Convergence: \int sin(k \pi x)','FontSize',16)
xlabel('N','FontSize',16); ylabel('Integration Error','FontSize',16);
legend('Gauss-Lobatto-Legendre','Composite Trapezoidal Rule','location','southwest'); pause

semilogy(n_gll,e_gll,'bo-',n_trap,e_trap,'ro-'); axis square;
title('GLL and Trapezoidal Rule Convergence: \int sin(k \pi x)','FontSize',16)
xlabel('N','FontSize',16); ylabel('Integration Error','FontSize',16);
legend('Gauss-Lobatto-Legendre','Composite Trapezoidal Rule','location','southwest'); pause

%-------------------------------------------------------
```

Quadrature Demo: Results

The accompanying figures show the results of the matlab demo script from the preceding box. (TBD)

Q: What value of N is required for trapezoidal rule to have error $< 10^{-4}$ when $k = 3.1$?

What value of N is required for GLL?

Q: What value of N is required for trapezoidal rule to have error $< 10^{-4}$ when $k = 93.1$?

What value of N is required for GLL?

2 Navier-Stokes, Energy Transport, and Model Equations

In this course we will be primarily concerned with developing algorithms and prototype codes to solve the incompressible Navier-Stokes equations (NSE) in order to explore the accuracy (i.e., rates of convergence) and the sources of instabilities that make numerical solution of these problems challenging (even when using robust commercial codes). The NSE are

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}, + \mathbf{f}, \tag{22}$$

$$\nabla\cdot\mathbf{u} = 0. \tag{23}$$

These equations are to be satisfied in a domain Ω with appropriate conditions on the domain boundary $\partial\Omega$, which will be introduced on a case-specific basis. Here, $\mathbf{u}(\mathbf{x},t)$ is the unknown velocity field having components that we will interchangeably denote as $\mathbf{u} = [u,v,w]^T = [u_1,u_2,u_3]^T = [u_x,u_y,u_z]^T$, p is the pressure, ρ is the fluid density, and μ is the dynamic viscosity. The problem specification is completed with appropriate boundary conditions on the domain boundary, $\partial\Omega$, and initial conditions, $\mathbf{u}(\mathbf{x},t=0)$, which will be prescribed on a case-by-case basis. With a change of variables, these equations can be written in nondimensional form,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u}, + \mathbf{f}, \tag{24}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{25}$$

where $Re = UL/\nu$ is the Reynolds number based on a characteristic velocity scale, U, and length scale, L, and $\nu := \mu/\rho$ is the kinematic viscosity. Here, we have also replaced p/ρ by $p \leftarrow p/\rho$, given that ρ is constant. In this form, time is nondimensionalized by a convective time scale, $\tau := L/U$ such that a single unit of time corresponds to the time required for a particle moving with speed U to travel a distance L.

We will work with either of the forms above and will generally leave density out of the equation as it will be constant in all cases that we will consider. To clarify the interactions of some of the terms, we rewrite (22) in indicial form. Assuming $\rho \equiv 1$, we have

$$\frac{\partial u_i}{\partial t} + u_j\cdot\frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + f_i, \tag{26}$$

$$\frac{\partial u_j}{\partial x_j} = 0, \tag{27}$$

where the repeated index indicates summation over $j = 1,\ldots,d$ in $d = 2$ or 3 space dimensions. Here, we have also introduced the full stress tensor, $\partial_j u_i + \partial_i u_j$. If the kinematic viscosity, $\nu = \mu/\rho$, is constant we have

$$\frac{\partial}{\partial x_j}\nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = \nu\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{28}$$

$$= \nu\left(\frac{\partial^2 u_i}{\partial x_j\partial x_j} + \frac{\partial}{\partial x_i}\frac{\partial u_j}{\partial x_j}\right) \tag{29}$$

$$= \nu\frac{\partial^2 u_i}{\partial x_j\partial x_j}. \tag{30}$$

Here, the second term on the right of (29) is identically zero because of the incompressibility condition (27) and we recover the Laplacian form of the viscous term as shown in (24).

For the constant viscosity case, the three velocity components are coupled only through the nonlinear advection term, $u_j \partial_j u_i$, and through the divergence-free constraint, $\partial_j u_j = 0$. In addition, certain slip boundary conditions can lead to additional coupling of velocity components on boundaries that are not aligned with the principal axes, **i**, **j**, or **k**.

In addition to the incompressible Navier-Stokes equations, we are frequently interested in thermal energy transport, governed by the advection-diffusion equation,

$$\rho C_p \left(\frac{\partial T}{\partial t} \; + \; \mathbf{u} \cdot \nabla T \right) \;\; = \;\; \nabla \cdot k \nabla T \; + \; q''' \qquad in\, \Omega_T \; \times \; [0, t_f], \tag{31}$$

where T is the temperature, C_p is the specific heat, $k(\mathbf{x}, t)$ is the thermal conductivity, and $q'''(\mathbf{x}, t)$ is a volumetric source term. As with the momentum equation, the energy equation must be accompanied by initial conditions $T(\mathbf{x}, t = 0)$ and boundary conditions on $\partial\Omega_T$. Generally, the thermal domain will be equal to the fluid domain, $\Omega_T \equiv \Omega$, unless one is solving a *conjugate heat transfer* (CHT) problem that involves both fluid and solid subdomains, in which case we have $\Omega \subset \Omega_T$. In the sequel, we will assume $\Omega_T = \Omega$ unless otherwise indicated.

To guide algorithmic design decisions in the development of numerical solution methods for (22) and (31), we discuss the nature of the governing equations term-by-term, starting with the thermal transport.

2.1 Characteristics of the Advection-Diffusion Equation

The advection-diffusion equation (31) is a linear partial differential equation (PDE) in d space dimensions and time.[1] As the name implies, there are two physical phenomena, *advection* and *diffusion*, that are governing the solution behavior in (31)

- ***Advection.*** Transports stuff with speed $\mathbf{u}$: $\partial_t T \; \sim \; -\mathbf{u} \cdot \nabla T.$ *(advection_eqn_demo.m)*

- ***Diffusion.*** Diffuses stuff: $\partial_t T \; \sim \; \frac{k}{\rho C_p} \nabla^2 T.$ *(heat_demo1.m)*

For most engineering applications, the energy equation is said to be *advection* dominated, which might lead one to think that the conductivity is "small" or that velocity is "large". Unfortunately, these terms do not have the same units and therefore cannot be compared directly. To make a precise statement about which physical terms are important, we need to nondimensionalize (31). The physical behavior cannot change if (31) is expressed in units of miles in one case and nanometers in another, but the coefficients in (31) would certainly change by orders of magnitude. Because (31) is linear, we can rescale the temperature by a multiplicative constant Moreover, since T appears only in differential terms, we can add a constant to it without changing the equation. We therefore define a new variable,

$$\theta := \frac{T - T_0}{T_1 - T_0}, \tag{32}$$

which is a nondimensional temperature, shifted and scaled by reference temperatures T_0 and T_1. With a bit of rearrangement, (31) becomes

$$\rho C_p \left(\frac{\partial \theta}{\partial t} \; + \; \mathbf{u} \cdot \nabla \theta \right) \;\; = \;\; \nabla \cdot k \nabla \theta \; + \; \frac{q'''}{T_1 - T_0}. \tag{33}$$

[1] Exceptions to the linearity assumption include cases where the propertes (e.g., ρC_p or k) or forcing (q''') are temperature dependent or where the fluid force ($\mathbf{f}$) is dependent on the temperature, as is the case for buoyancy-driven flows. Unless these dependencies are extreme, the principal physics is generally captured by (31).

Note that any inhomogeneous boundary or initial data must also be transformed according to (32). Next, we collect the material properties to the right-hand side. Assuming constant properties[2], we have

$$\frac{\partial \theta}{\partial t} \; + \; \mathbf{u} \cdot \nabla \theta \;\; = \;\; \nabla \cdot \alpha \nabla \theta \; + \; \frac{q'''}{(T_1 - T_0)\rho C_p}, \tag{34}$$

where $\alpha := k/(\rho C_p)$ is the thermal diffusivity (e.g., at a reference temperature $(T_0 + T_1)/2$).

So far, we have removed the units of the *dependent* variable (temperature) from (31). The next step is to rescale the *independent* variables, namely, $\mathbf{x}$ and t. For the spatial variable, $\mathbf{x}$, one typically chooses to rescale with respect to a length scale that is directly relevant to the problem. For example, the diameter of a cylinder or of a pipe, or a channel height or (commonly) half-height, could serve as a characteristic length scale, L. Again, the precise choice is not overly important—the physics and the answers will be the same regardless of the definition of L. One generally chooses a length that is either convenient or matches standard conventions (e.g., $L = D$, the diameter, is generally preferred over the radius for flow past a cylinder or in a pipe). For time, we will invariably use a *convective* time scale, $\tau := L/U$, where U is a characteristic velocity scale such as the mean free-stream velocity in the case of external flows or the bulk (e.g., time- and space-averaged) velocity for internal flows. Obviously, this timescale will not work in the case of pure conduction (i.e., $\mathbf{u} \equiv 0$) and one must use a diffusive timescale such as $\tau = L^2/\alpha$. Defining nondimensional quantities, $\tilde{\mathbf{x}}$, $\tilde{t}$, $\tilde{\mathbf{u}}$, and $\tilde{\nabla}$ such that

$$\mathbf{x} \;\; := \;\; L\tilde{\mathbf{x}}, \qquad t \; := \; \tau \tilde{t}, \qquad \mathbf{u} \; = \; U\tilde{\mathbf{u}}, \qquad \nabla \; = \; \frac{1}{L}\frac{\partial}{\partial \tilde{x}_i} \; =: \; \frac{1}{L}\tilde{\nabla}, \tag{35}$$

we can rewrite (34) in terms of these nondimensional quantities as

$$\frac{1}{\tau}\frac{\partial \theta}{\partial \tilde{t}} \; + \; \frac{U}{L}\tilde{\mathbf{u}} \cdot \tilde{\nabla}\theta \;\; = \;\; \tilde{\nabla} \cdot \frac{\alpha}{L^2}\tilde{\nabla}\theta \; + \; \frac{q'''}{(T_1 - T_0)\rho C_p}. \tag{36}$$

Multiplying through by $\tau = L/U$ (the only dimensional term present on the left of (36)) yields

$$\frac{\partial \theta}{\partial \tilde{t}} \; + \; \tilde{\mathbf{u}} \cdot \tilde{\nabla}\theta \;\; = \;\; \tilde{\nabla} \cdot \frac{\alpha}{UL}\tilde{\nabla}\theta \; + \; \frac{Lq'''}{(T_1 - T_0)\rho C_p U} \tag{37}$$

$$= \;\; \tilde{\nabla} \cdot \frac{1}{Pe}\tilde{\nabla}\theta \; + \; \tilde{q}''', \tag{38}$$

where we have introduced the nondimensional *Peclet* number,

$$Pe \;\; := \;\; \frac{UL}{\alpha} \;\; = \;\; \frac{\rho C_p U L}{k} \tag{39}$$

and nondimensional source term

$$\tilde{q}''' \;\; := \;\; \frac{Lq'''}{(T_1 - T_0)\rho C_p U}. \tag{40}$$

In the case where k is constant, we can pull the Peclet number outside of the divergence term in (38) to arrive at the standard form for the advection-diffusion equation,

$$\frac{\partial \theta}{\partial \tilde{t}} \; + \; \tilde{\mathbf{u}} \cdot \tilde{\nabla}\theta \;\; = \;\; \frac{1}{Pe}\tilde{\nabla}^2\theta \; + \; \tilde{q}'''. \tag{41}$$

[2]If the properties are variable, we pick reference values and proceed in the same way.

To leading order, it is the Peclet number that indicates whether a system is advection dominated or diffusion dominated. Large Peclet, $Pe \gg 1$, indicates that $\rho C_p U L$ dominates the diffusive process, which scales with conductivity, k. For $Pe \lesssim 100$, conduction plays an important role. As we will see thoughout the course, conduction-dominated problems are generally easier to solve than their convection-dominated counterparts. Some illustration of the challenges of the advection-dominated limit is given by the 1D example in the accompanying boxed text.

Advection-Dominated Example in 1D

To illustrate the potential difficulty of convection-dominated problems, we consider the 1D steady-state advection-diffusion problem on $\Omega = [0, L]$,

$$-\nu \frac{d^2\tilde{u}}{dx^2} + c\frac{d\tilde{u}}{dx} = 1, \qquad \tilde{u}(0) = \tilde{u}(1) = 0. \tag{42}$$

Here, we assume that the diffusivity ν and advection speed c are constants. (Throughout the course, we will often use $\tilde{u}$ for the solution to the continuous problem that is to be compared with its numerical counterpart, u.) The solution to (42) is

$$\tilde{u}(x) = \frac{1}{c}\left(x - L\frac{e^{c(x-L)/\nu} - e^{-cL/\nu}}{1 - e^{-cL/\nu}}\right) = \frac{L}{c}\left(\xi - \frac{e^{Pe(\xi-1)} - e^{-Pe}}{1 - e^{-Pe}}\right), \tag{43}$$

where $Pe := cL/\nu$ and $\xi := x/L$. For $Pe \gg 1$, this solution exhibits a (nonoscillatory) boundary layer of thickness $\delta \sim L/Pe$ near $x = L$. As $Pe \longrightarrow \infty$, the equation is said to be *singularly perturbed* because the coefficient in front of the the second order derivative goes to zero, which nominally reduces the order of the ordinary differential equation from two to one. The difficulty is that there continue to be two boundary conditions to be satisfied and the solution must accommodate the second ("downwind") boundary condition over a distance $\delta \sim L/Pe$.

We next consider a 2nd-order centered finite difference discretization of (42) using a uniform grid with spacing $\Delta x = L/n$:

$$-\nu\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} + c\frac{u_{j+1} - u_{j-1}}{2\Delta x} = 1. \tag{44}$$

for $j = 1, \ldots, n-1$ and boundary data $u_0 = u_n = 0$. This linear 2nd-order difference equation has the closed form solution (see (12.75) in [QSS07]),

$$u_j = \frac{L}{c}\left[\frac{x_j}{L} - \frac{1 - \gamma^j}{1 - \gamma^n}\right], \qquad j = 1, \ldots, n-1, \tag{45}$$

where

$$\gamma := \frac{2 + Pe_g}{2 - Pe_g}, \qquad Pe_g := \frac{c\Delta x}{\nu}.$$

Here, we have introduced the *grid Peclet* number, Pe_g, which quantifies the ratio of advection to diffusion effects at the grid scale. Notice that γ is positive if $Pe_g < 2$ and negative if $Pe_g > 2$. As a result, the solution u_j will be oscillatory for large grid Peclet numbers.

Exercise: Write a matlab script to solve (44), starting with the code here:

```
h = L/n; n1=n-1; e = ones(n1,1);
A = -a*spdiags([e -2*e e], -1:1, n1, n1)/(h*h); % A - SPD
C = c*spdiags([-e 0*e e], -1:1, n1, n1)/(2*h); % C - skew-symm.
H = A+C; % Advection-Diffusion operator
x = [1:n1]*h; x=x';
f = 1+0*x;
u = H\f;
ub = [0; u; 0]; xb=[0; x; L]; % Extend u and x to boundaries
```

- Plot solutions to compare results from (43), (44), and (45) for several values of n and Pe.
- Under what conditions do you see oscillatory behavior in your numerical solution?
- Plot $\|\underline{u} - \underline{\tilde{u}}\|_\infty / \|\underline{\tilde{u}}\|_\infty$ vs n for $Pe = 1$, 10, 100, and 1000 (4 graphs, one figure), where $\underline{u} := [u_0, \ldots, u_n]^T$ and $\underline{\tilde{u}} := [\tilde{u}(x_0), \ldots, \tilde{u}(x_n)]^T$. Choose your axes correctly! (*Note:* use a geometric progression, e.g., n =2, 4, 8,...)
- What is your observed rate of convergence? When does the solution start to converge?

2.2 Characteristics of the Navier-Stokes Equations

The incompressible Navier-Stokes equations form a nonlinear system of partial differential equations (PDEs) in d space dimensions and time. They comprise the momentum equation (24) and the divergence-free constraint (25), which enforces the incompressibility of the flow. To obtain a nondimensional set of equations, we proceed as we did in the thermal case by introducing characteristic length, time, and velocity scales and substituting the expressions given in (35) into (22). Upon dividing the result by $\rho U/\tau = \rho U^2/L$ we arrive at

$$\frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{t}} + \tilde{\mathbf{u}} \cdot \tilde{\nabla} \tilde{\mathbf{u}} = -\tilde{\nabla} \tilde{p} + \frac{1}{Re} \tilde{\nabla}^2 \tilde{\mathbf{u}}, + \tilde{\mathbf{f}}, \tag{43}$$

$$\tilde{\nabla} \cdot \tilde{\mathbf{u}} = 0, \tag{44}$$

with corresponding initial and boundary conditions. Here, we have introduced the *Reynolds number*,

$$Re := \frac{\rho L U}{\mu} = \frac{L U}{\nu}, \tag{45}$$

and a rescaled pressure, $\tilde{p} := p/(\rho U^2)$. Naturally, the domain and time interval are also rescaled by L and τ, respectively. We will generally work with the nondimensional form and will drop the ˜s in the subsequent discussion. The Reynolds number is the all-important parameter in fluid dynamics. Large Reynolds numbers (e.g., $Re \gg 1000$) typically implies that the flow will be turbulent, whereas small Reynolds numbers (e.g., $Re < 10$) imply that the flow will be laminar (viscous-dominated).

In the momentum equations, we have the nonlinear terms, $\mathbf{u} \cdot \nabla \mathbf{u}$, which act to advect each component of the momentum (ρu_i) by the convecting field, u_j. Here of course the advecting field $\mathbf{u}$ and the solution $\mathbf{u}$ are the same, but we can view these as distinct variables. For example, a common special form of the momentum equations are the Oseen equations,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{c} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u}, + \mathbf{f}, \tag{46}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{47}$$

with $\nabla \cdot \mathbf{c} = 0$. Here $\mathbf{c}$ is a prescribed (divergence-free) advecting field and (46) is consequently *linear*. As was the case with the Peclet number in the advection-diffusion equation, a high Reynolds number, $Re \gg 1$, indicates an advection dominated problem for either the Oseen or Navier-Stokes equations. In the latter case, nonlinearity can play a significant role at large Re as the solution can break down into unsteady and potentially chaotic behavior, even when the data (forcing and boundary conditions) are time independent. Working with the Oseen equations affords significant benefits when developing numerical algorithms for the Navier-Stokes equations because it is easier to find stable exact solutions at high Reynolds numbers and, because of linearity, one can also compare against solutions based on eigenfunction expansions.

Another simplified subset of the momentum equations is provided by the Leray-regularized Navier-Stokes equations,

$$\frac{\partial \mathbf{u}}{\partial t} + < \mathbf{u} > \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u}, + \mathbf{f}, \tag{48}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{49}$$

where $< \mathbf{u} >$ represents a spatially-filtered (i.e., smoothed) advecting field. While (48) is still fully nonlinear, the small amount of regularization introduced into these equations allows forward

progress in analysis of the otherwise challenging Navier-Stokes equations (e.g., [Guermond *et al.*, 2004]).

Still simpler than any of the preceding cases are the linear (unsteady) Stokes equations, in which we drop the advection term altogether,

$$\frac{\partial \mathbf{u}}{\partial t} = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} + \mathbf{f}, \tag{50}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{51}$$

Clearly, these equations are never advection dominated—they can in fact be rescaled so that the Reynolds number is not the controlling parameter since they are really physically relevant only for $Re \ll 1$. Geophysical flow that has relatively low velocities is one class of problems that is governed by the Stokes equations. One can also have the *steady* Stokes equations, typically written as

$$-\nabla^2 \mathbf{u} + \nabla p = \mathbf{f}, \tag{52}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{53}$$

with, as always, appropriate boundary conditions prescribed on $\partial\Omega$.

For incompressible flow simulations, the pressure acts as the Lagrange multiplier that allows the momentum equations to be solved subject to the divergence-free constraint. Taking the divergence of (24) and applying (25) leads to

$$-\nabla^2 p = -\nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{f}), \tag{54}$$

which shows that the pressure satisfies an elliptic Poisson problem at every instant in time. A significant issue is that the correct boundary conditions are expressed in terms of the velocity (specifically, $\nabla \cdot \mathbf{u}|_{\partial\Omega} = 0$) and it is therefore difficult to use (54) for direct computation of the pressure. Nonetheless, some of the most efficient approaches for simulating incompressible flows are based on this approach and identification of stable approaches to resolving this issue will be an important topic in this course.

Scaling Exercises.
Estimate the following to reasonable engineering accuracy or order of magnitude. You may work with one other partner on your submission. (List your partner's name.) For each case, state the constants used and justify those values. Unless otherwise indicated, assume standard conditions (STP) when finding the constants.

1. The Reynolds number for a major-league fastball pitch.
2. The Reynolds number for a golf ball in flight.
3. The Peclet number for 1 m/s flow of liquid sodium in a pipe of diameter 2.5 cm at $T = 300C$.
4. The Reynolds number for a grain of sand falling in water.
5. The chord Reynolds number for a pigeon in flight.
6. The chord Reynolds number for a 747 at cruise conditions.
7. The Reynolds number for water flow at 1 m/s in a 1 cm pipe at $T = 300K$.
8. The Reynolds number for flow in the exhaust pipe of a typical passenger car under standard cruise conditions.
9. The Peclet number for flow in the exhaust pipe of a typical passenger car under standard cruise conditions.
10. The Reynolds number for the Couette flow that supports read/write heads in a typical computer disk drive. (*Hint:* What is the correct length scale for Couette flow?)
11. The Reynolds number in a healthy carotid artery.
12. Reynolds number for water is flowing through a 2 cm diameter tube at 1 gallon (US) per minute.
13. Reynolds number for water is flowing through a 1 cm diameter tube at 1 gallon (US) per minute.
14. Flow past a stirring stick when stirring coffee or tea.

- For 11, just use a value from available liturature and cite your source.
- Which of the above are likely to correspond to laminar flow?

3 Quick Overview of Splitting-Based Navier-Stokes Solutions

For computational efficiency, it is common to use temporal splitting to decouple the physics of the incompressible Navier-Stokes equations into independent and more tractable substeps. To illustrate this idea, we introduce here a kth-order (k =2 or 3) time-split formulation following [**?**, **?**, **?**]. In subsequent lectures we will justify each of these choices in detail. The point here is to give a global picture of the overall approach so that the readers can understand the context for choices of numerical methods as we proceed through the course.

In nondimensional form, we start with the full equations governing the velocity, $\mathbf{u}(\mathbf{x},t)$, and pressure, $p(\mathbf{x},t)$, for $\mathbf{x} \in \Omega$.

$$\frac{\partial \mathbf{u}}{\partial t} \; - \; \frac{1}{Re}\nabla^2\mathbf{u} \; + \; \nabla p \;\; = \;\; -\mathbf{u}\cdot\nabla\mathbf{u} \; + \; \mathbf{f}, \tag{55}$$

$$\nabla\cdot\mathbf{u} \;\; = \;\; 0, \tag{56}$$

with initial condition, $\mathbf{u}(\mathbf{x}, t = 0) = \mathbf{u}^0$, and boundary condition, $\mathbf{u}(\mathbf{x},t) = \mathbf{u}_b$ on that part of the domain boundary where Dirichlet conditions are applied, $\partial\Omega_D \subset \partial\Omega$. For purposes of this exposition we assume homogeneous Neumann conditions on $\partial\Omega_N := \partial\Omega\backslash\partial\Omega_D$. The terms on the left of (55)–(56) are treated implicitly, with the temporal derivatve computed at time t^m using a kth-order backward difference formulation (BDFk),

$$\left.\frac{\partial \mathbf{u}}{\partial t}\right|_{t^m} \;\; \equiv \;\; \frac{1}{\Delta t}\sum_{j=0}^{k}\beta_j\mathbf{u}^{m-j} \; + \; O(\Delta t^k), \tag{57}$$

where $\mathbf{u}^{m-j}$ denotes the velocity field at time t^{m-j}. For the case of uniform timestep size Δt, $t^m = m\Delta t$, and the BDFk coefficients are $[\beta_0,\ldots,\beta_k]$=[-1,1] for $k = 1$, [3/2, -4/2, 1/2] for $k = 2$, and [11/6, -18/6, 9/6, -2/6] for $k = 3$. The terms on the right of (55), which include nonlinear advection and any forcing (e.g., Coriolis or Boussinesq terms) are treated with kth-order extrapolation. If $\mathbf{g} := -\mathbf{u}\cdot\nabla\mathbf{u} \; + \; \mathbf{f}$, then the right-hand side of (55) is

$$\mathbf{g}|_{t^m} \;\; \equiv \;\; \sum_{j=1}^{k}\alpha_j\mathbf{g}^{m-j} \; + \; O(\Delta t^k). \tag{58}$$

For uniform Δt, we have $[\alpha_1,\ldots,\alpha_k]$=[1] for $k = 1$, [2, -1] for $k = 2$, and [3, -3, 1] for $k = 3$. For the semidiscretization in time, we drop the $O(\Delta t^k)$ residual terms and use (57)–(58) in (55). All other terms are evaluated implicitly at time t^m.

The first step in the splitting scheme is simply to sum all quantities known from previous timesteps, which amounts to constructing a tentative velocity field,

$$\hat{\mathbf{u}} \;\; := \;\; \sum_{j=1}^{k}\left(-\beta_j\mathbf{u}^{m-j} \; + \; \Delta t\alpha_j\mathbf{g}^{m-j}\right). \tag{59}$$

As this step is just data collection, no boundary conditions are imposed in (59). The remaining implict equations are linear but are still coupled because of the incompressibility constraint (56). Formally taking the divergence of the momentum equation (55) and requiring $\nabla\cdot\mathbf{u}^m = 0$ leads to a Poisson equation for the pressure,

$$-\nabla^2 p^m \;\; = \;\; -\nabla\cdot\hat{\mathbf{u}}. \tag{60}$$

Boundary conditions for (60) are found by taking the dot product of (55) with the outward-pointing unit normal on Ω, which yields

$$\nabla p^m \cdot \hat{\mathbf{n}}\Big|_{\partial\Omega} \;=\; \left(\frac{\hat{\mathbf{u}} - \beta_0 \mathbf{u}^m}{\Delta t} + \frac{1}{Re}\nabla^2 \mathbf{u}^m\right) \cdot \hat{\mathbf{n}}\Big|_{\partial\Omega}. \tag{61}$$

We remark that $\mathbf{u}^m$ is known on $\partial\Omega$ while $\nabla^2\mathbf{u}^m$ is not. As with the nonlinear term, the unknown viscous term can be treated explicitly after applying the vector identity

$$\nabla^2 \mathbf{u}^m \;=\; \nabla(\nabla \cdot \mathbf{u}^m) \;-\; \nabla \times (\nabla \times \mathbf{u}^m). \tag{62}$$

The first term on the right is set to zero because of (56), so one only needs to extrapolate the curl of the vorticity, $\nabla \times (\nabla \times \mathbf{u}^m)$, on $\partial\Omega_D$ [**?**]. Once the pressure is known from (60)–(61), we can compute each component of $\mathbf{u}^m = [u_1^m\ u_2^m\ u_3^m]$ by solving a sequence of Helmholtz problems,

$$-\frac{\Delta t}{Re}\nabla^2 u_j^m \;+\; \beta_0 u_j^m \;=\; \Delta t\left(\hat{u}_j - \frac{\partial p^m}{\partial x_j}\right), \quad j = 1, \ldots, 3, \tag{63}$$

subject to appropriate boundary conditions on each component of $\mathbf{u}^m$. In the case of mixed boundary conditions or variable viscosity, we replace (63) by a system in which the velocity components are coupled. In either case, the viscous system will be diagonally dominant for modest to high Reynolds number applications given that $\Delta t/Re \ll 1$. For these cases, Jacobi preconditioned conjugate gradient iteration is an efficient solution strategy for the fully discretized form of (63).

3.1 Chaotic Systems: Sensitivity to Initial Conditions

When determining the accuracy of a numerical code for CFD, it is common to compare against exact solutions of the Navier-Stokes equations. In this endeavor, however, one must exercise caution because exact solutions, even in the steady case, may not be stable at modest to high Reynolds numbers. Worse still, high Reynolds number solutions tend to be chaotic in time, which is one of the characteristic traits of turbulence. Small perturbations in the initial conditions, say, at the scale of round-off error in the machine representation, can rapidly lead to divergence between two solutions, even exact ones, which makes it difficult to do direct convergence studies or comparisons against "gold-standard" solutions. In this regime, one must rely either on statistical comparisons or very short-time solutions. The latter approach can be misleading, however, since short-time errors tend to reflect only the error in the initial condition rather than errors in the spatial/temporal discretization, which take some time to grow to their saturated levels.

To illustrate rapid divergence in chaotic systems, we consider the example of Frisch [**?**], who proposed the simple single-degree-of-freedom model problem,

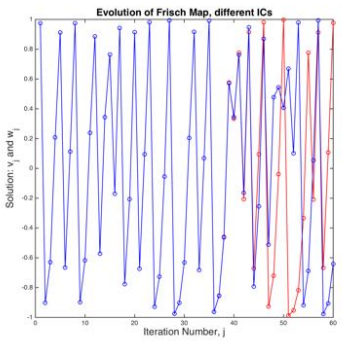
$$v_{n+1} \;=\; 1 - 2v_n^2, \qquad v_0 = v^*, \tag{64}$$

for n=0, 1, 2,... The similarity between this nonlinear map and the Navier-Stokes equations is evident if we write them next to each other. With minor rearranging we have,

$$\begin{array}{lrcrcrcl} \textit{Model Problem:} & v_{n+1} - v_n & = & -2v_n^2 & - & v_n & + & 1 \\ \textit{Navier-Stokes:} & \partial_t \mathbf{u} & = & -(\mathbf{u}\cdot\nabla\mathbf{u} + \nabla p) & + & \nu\nabla^2\mathbf{u} & + & \mathbf{f}. \end{array} \tag{65}$$

We note that $\nabla^2\mathbf{u}$ is a negative-definite operation (e.g., in Fourier space, the second derivative with respect to x leads to $-k^2\hat{\mathbf{u}}$), which is similar to the $-v_n$ term in (65). Similarly, we have the nonlinear term, coupled with the gradient of the Lagrange multiplier, p.

```
%% Frisch Map.  Usage:  n=60; frisch
u=zeros(n,1); r=zeros(n,1);
v=0.11110101010101; w=0.11110101010102;
d0=abs(v-w);
for j=1:n;
  v=1-2*v*v; u(j)=v;
  w=1-2*w*w; r(j)=w;
  s=[ j v w w-v ];
  z=sprintf('%4d %16.11f  %16.11f  %16.12f',s);
  disp(z)
end;
t=1:n; plot(t,u,'ro-',t,r,'bo-'); axis square
model = d0*2.^t; model=min(2,model);
semilogy(t,abs(r-u),'g.-',t,model,'linewidth',1.2);
axis square
```



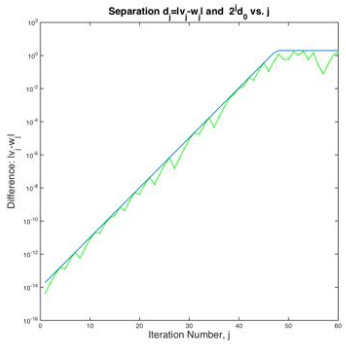


Figure 2: Evolution of the map (64) for two initial conditions, v_0 and w_0, that differ in the 14th significant digit: (left) matlab code, (center) map of v_j and w_j vs. iteration j, (right) map of $|v_j - w_j|$ vs. j.

Figure 2 illustrates the behavior of the map (64) for two different initial conditions that differ only in the 14th significant digit. This map loses one significant (binary) bit with each iteration, which means that after $46 \approx \log_2 10^{14}$ iterations the difference in the solutions will be order unity. Indeed, we observe this behavior when iterating the matlab code at the left of Fig. 2, which produces the evolution shown in the center. The growth of $|v_j - w_j|$ is shown in the semi-log plot on the right, along with the model curve, $|v_0 - w_0|2^j$. Clearly, the difference in the evolution is multiplied, on average, by a factor of 2 with each iteration.

3.1.1 Impact of Round-Off Error

We will not make extensive discussion of round-off errors in this course. Our objective in general will be to avoid situations where round-off effects have a significant impact on our solution. However, we take a moment here to review some of the basic points of concern.

First, we note from the preceding example that any perturbation in even the least significant bit can ultimately lead to widely separated solution trajectories. With round-off, in which each floating point operation effectively injects a random bit at the tail of the mantissa (the 52nd bit in IEEE 64-bit arithmetic), one can view every step of an algorithm as a point of noise injection. While the basic floating point operations + and * are commutative, they are not associative because of rounding of intermediate results. This issue is often manifest in parallel computing applications where results may arrive from different processors in different orders, meaning that the order and thus the results of operations might change even when running the same code with the same initial data on different numbers of processors. The key to repeatability is (*i*) to use stable algorithms and *stable discretizations* that are relatively insensitive to such perturbations and (*ii*) to recognize which questions are well-posed and in some sense repeatable.

To understand the first point, we review the conditioning of some basic operations. In numerical analysis, the condition number of a function $y = f(x)$ is an estimate of the sensitivity of an operation to perturbations in the input data. We are interested in the *relative* change in the output, Δy as a function of the *relative* change in the input, Δx. The condition number for f is

$$\text{condition number:} \quad C \;=\; \max \frac{|\Delta y/y|}{|\Delta x/x|} \;=\; \left|\frac{x}{y}\right| \left|\frac{\Delta y}{\Delta x}\right| \;\approx\; \left|\frac{x}{y}\right| \left|\frac{\Delta x f'(x)}{\Delta x}\right| \;=\; \left|\frac{x}{y}\right| \left|f'(x)\right|. \qquad (66)$$

If the condition number is order unity the operation is *well-conditioned*. If C is large, it is *ill-conditioned*. Note that C is generally dependent on the input argument, x.

Consider some basic arithmetic operations $f(x) = a * x$, $f(x) = a/x$, $f(x) = a + x$, and

$f(x) = \sqrt{x}$. We have the following condition number estimates:

$$f = a * x: \quad C = \left| \frac{x}{ax} a \right| = 1 \tag{67}$$

$$f = a/x: \quad C = \left| \frac{x}{a/x} a x^{-2} \right| = 1 \tag{68}$$

$$f = a + x: \quad C = \left| \frac{x}{a+x} \right| \tag{69}$$

$$f = \sqrt{x}: \quad C = \left| \frac{x}{x^{\frac{1}{2}}} \frac{1}{2} x^{-\frac{1}{2}} \right| = \frac{1}{2}. \tag{70}$$

Of the above, only the innocent looking $a + x$ is potentially unbounded. The rest of the operations are *benign*. In particular, from a loss-of-precision standpoint, division is no worse than multiplication. One must of course be concerned if the denominator in an operation may potentially be zero and should always inspect denominators for such potentiality and ask whether the division is absolutely necessary. (In linear algebra, division is relatively rare—factoring an $n \times n$ matrix requires $O(n^3)$ multiplications and additions, but only n divisions.) In the case of $a + x$, if a and x are approximately equal in magnitude but of opposite sign, one can expect loss of many significant digits through cancellation. The operation in this case is certainly not well conditioned.

A common place where cancellation arises in numerical analysis is in the numerical approximation of derivatives. Consider the following finite difference formula applied to $u(x_j) =: u_j$ on a grid with uniform spacing $x_j := j \cdot h$.

$$\left. \frac{du}{dx} \right|_{x_j} = \frac{u_{j+1} - u_j}{h} + c_1 h + \text{h.o.t.} \tag{71}$$

$$\left. \frac{du}{dx} \right|_{x_j} = \frac{u_{j+1} - u_{j-1}}{h} + c_2 h^2 + \text{h.o.t.} \tag{72}$$

$$\left. \frac{d^2u}{dx^2} \right|_{x_j} = \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + c_3 h^2 + \text{h.o.t.}, \tag{73}$$

where the c_*s are order-unity constants and h.o.t. refers to higher-order terms in the Taylor series expansion of u assuming that u is sufficiently differentiable in the neighborhood of x_j. In finite (floating point) arithmetic, we do not in general have exact representations of the gridpoint values $u_j = u(x_j)$. Rather, we have $u_j = u(x_j)(1 + \epsilon_j)$, where $|\epsilon_j| < \epsilon_M$ is a random variable bounded by the machine precision, $\epsilon_M \approx 10^{-16}$ for the IEEE 64-bit floating point standard. Consequently, numerically computed derivatives have error behaviors obeying

$$\frac{u_{j+1} - u_j}{h} = \left. \frac{du}{dx} \right|_{x_j} - c_1 h + \text{h.o.t.} + c_1' u_j \frac{\epsilon_M}{h} \tag{74}$$

$$\frac{u_{j+1} - u_{j-1}}{2h} = \left. \frac{du}{dx} \right|_{x_j} - c_2 h^2 + \text{h.o.t.} + c_2' u_j \frac{\epsilon_M}{h} \tag{75}$$

$$\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} = \left. \frac{d^2u}{dx^2} \right|_{x_j} - c_3 h^2 + \text{h.o.t.} + c_3' u_j \frac{\epsilon_M}{h^2}. \tag{76}$$

Here, we have introduced order-unity random variables c_*' to account for round-off. Notice that the numerical approximations to the first derivative both exhibit $O(h^{-1})$ growth in round-off error with

a small multiplier, ϵ_M, while the second-derivative operation has a more severe growth of $O(h^{-2})$ with roughly the same multiplier. In general, one finds that approximations to the kth derivative of u will exhibit round-off error of $\approx |u_j|\epsilon_M h^{-k}$. Changing to a high-order *approximation* of a derivative (e.g., as in the present case with $k = 1$) does not directly improve the round-off error. Rather, it enables the approximation to realize a reasonable numerical truncation error (here, $O(h)$ or $O(h^2)$) with a larger value of h, so that round-off effects are mitigated.

Before closing this section, we make one other remark on round-off and conditioning. A useful guideline in determining the number of correct digits when solving a linear system is that matrix factorization will yield $\approx |\log_{10} \epsilon_M| - |\log_{10} C|$ digits of accuracy, where C is the condition number of the matrix. Thus, if the condition number is 10^5, one can expect $\approx 16 - 5 = 11$ digits of accuracy when solving a given linear system. As a consequence, it is beneficial to design discretizations and algorithms that lead to well-conditioned systems. For example, methods based on orthogonal expansions are generally well conditioned (but not always rapidly convergent, unless they are eigenfunctions of singularly perturbed Sturm-Liouville problems, which is the subject of a future section.)

Round-Off Exercise

In the following exercises, consider the function $u(x) = \sin x$.

- **i.** Plot the expected *truncation* error, i.e., the difference between analytical values on the left of (72)–(73) and their finite difference formulae on the right, as a function of h, for $h = 2^{-k}$, $k = 0, 1, 2, \ldots 52$.
- **ii.** On the same graph as for Part i., plot the observed error, which is incurred when you actually implement the finite-difference formulae on the computer. To quantify the error, choose 50 uniformly-spaced points x_j on the interval $[0, \frac{\pi}{2}]$ and track the maximum pointwise error,

$$\max_j \left| \frac{u_(x_j + h) - u(x_j - h)}{2h} - \left.\frac{du}{dx}\right|_{x_j} \right| \tag{77}$$

$$\max_j \left| \frac{u(x_j + h) - 2u(x_j) + u(x_j - h)}{h^2} - \left.\frac{d^2u}{dx^2}\right|_{x_j} \right|, \tag{78}$$

  when you evaluate both the finite-difference approximation and analytical derivative on the computer. (Note: x_{j+1} is *not* $x_j + h$ in these tests.) Plot these results on the same graph as for (i) (one figure, four lines, with symbols, please).
- **iii.** What is the minimum error realized for each case (i.e., first- and second-derivative)? For what value of h do you realize this minimal error? How do the results of the computational experiments (ii) compare to the results based on analysis (i) ?
- **iv.** Write a short formula to determine the value of h where the min error is realized and the value of the realizeable error for each of the derivatives. (This is a simple balance of two terms in each of (75) and (76).)
- **v.** How would the results of (iv) change if your truncation error was $O(h^4)$? That is, what would be the h and error for the minimal-realizable error?

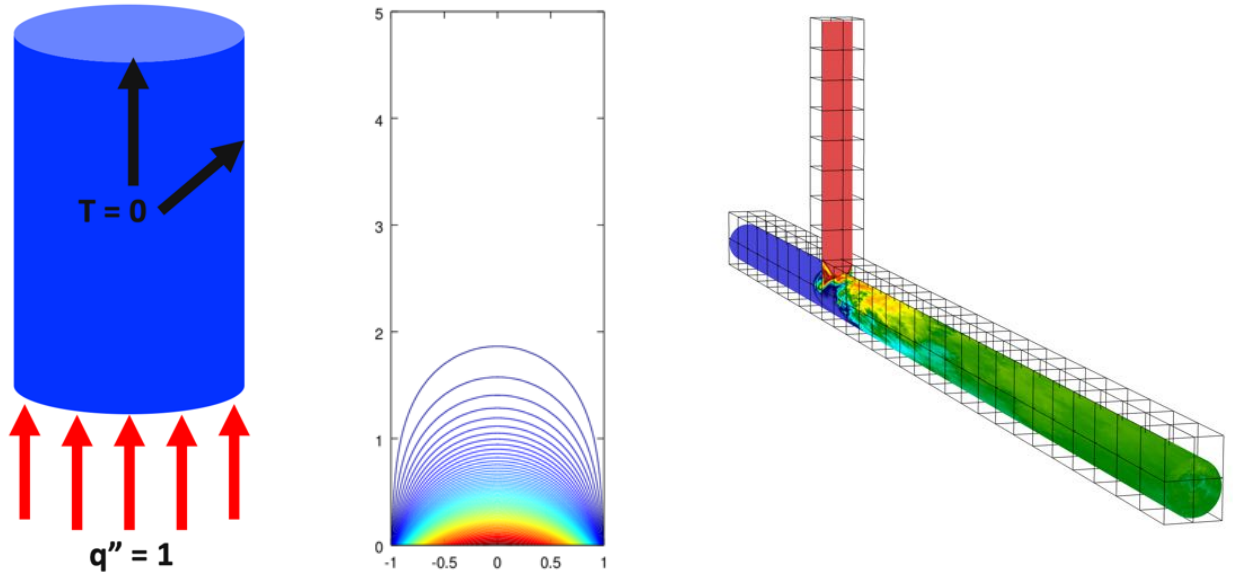


Figure 3: PDEs in simple domains, such as the cylindrical domain on the left, are often tractable using closed-form solution approaches such as separation-of-variables. The center panel illustrates the temperature distribution in a cylinder of height $5R$. More complex domains, such as the T-junction on the right, invariably require numerical methods for even the simplest PDEs.

3.2 Geometric Complexity

As noted in the previous section, one of the challenges to the solution of the Navier-Stokes equations derives from their extreme nonlinearity. In practice, however, much of the difficulty stems from the fact that the geometry is complex. Even linear problems are challenging in this respect. While we can find closed-form solutions for, say, conduction in simple domains such as the cylinder problem pictured on the left in Fig. 3, finding such solutions for the T-junction domain on the right is nearly impossible. Thus, much of engineering software development is directed towards solving problems in complex domains.

For turbulent flow simulations, spectral methods were extended to complex domains first by Orszag [SAO80] and subsequently by Patera [ATP84], who introduced the spectral element method, which was the first variational formulation for multidomain spectral methods. The use of variational projection operators (for elliptic or parabolic problems) guarantees that the solution is the *best approximation* in the space of trial functions. Thus, using orthogonal polynomial bases is not requisite—any set of $N+1$ polynomials spanning $\mathbb{P}_N$ (the space of polynomials of degree $\leq N$) would yield the same result, modulo round-off effects. It is worth noting that the obvious choice of monomial basis functions $\phi_j(x) = x^j$ leads to a condition number that grows exponentially with N—after $N > 15$ virtually all precision is lost. Orthogonal bases (in an appropriate inner product) lead to much better conditioned systems. The variational projection operators (at the heart of finite and spectral element methods) guarantee the best approximation.

3.3 Closed-Form Solution Example

Spectral methods derive their name from the fact that the solutions are combinations of the eigenfunctions of a PDE. Here, we explore the convergence behavior for such an approach by considering the solution of a classic PDE.

Consider steady-state conduction in a cylindrical domain, $\Omega = [0 : 2\pi] \times [0 : R_{cyl}] \times [0 : L]$. In cylindrical coordinates, the governing Poisson equation is

$$-\kappa \left[\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial T}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 T}{\partial \theta^2} + \frac{\partial^2 T}{\partial z^2} \right] \;=\; q'''(r,z) \;=\; 0, \qquad \text{plus BCs.} \tag{79}$$

Here, we take the volumetric source, q''' to be zero and we assume that we have a unit surface flux, $q'' = 1$ on the bottom surface of the cylinder at $z = 0$, and $T = 0$ on all other domain boundaries, as pictured in Fig. 3, left. Because the geometry, data (q'''), and boundary conditions are independent of θ we can expect that the solution is rotationally invariant and therefore the θ dependency drops out of (79). The unit-flux boundary condition corresponds to

$$-\kappa \frac{dT}{dz}\bigg|_{z=0} \;=\; 1. \tag{80}$$

Without loss of generality, we assume $\kappa = 1$ in the sequel.

To develop an analytical solution for (79) we use *separation of variables*, in which we seek a solution of the form

$$T(r,z) \;=\; T_N(r,z) \;:=\; \sum_{k=1}^{N} R_k(r) Z_k(z), \tag{81}$$

where the sum is possibly an infinite sum. For the moment, we drop the subscript k and seek a single solution of the form $T = R(r)Z(z)$. Linearity of (79) guarantees that we can sum multiple solutions as in (81) to meet the desired boundary conditions. To begin, we define

$$\begin{aligned} \mathcal{L}_r T \;&:=\; \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial T}{\partial r} \right) \;=\; \frac{1}{r} T_r + T_{rr} \;=\; \left(\frac{1}{r} R' + R'' \right) Z \\ \mathcal{L}_z T \;&:=\; \frac{\partial^2 T}{\partial z^2} \;=\; RZ'', \end{aligned} \tag{82}$$

such that (79) becomes

$$-\left(\mathcal{L}_r + \mathcal{L}_z\right) T \;=\; 0, \qquad \text{plus BCs.} \tag{83}$$

Taking the z-dependent term to the right and dividing through by RZ leads to

$$\frac{\frac{1}{r} R' + R''}{R} \;=\; -\frac{Z''}{Z} \;=\; \textit{constant} \;=\; -C. \tag{84}$$

The fact that this expression is a constant derives from the fact that the Z term is independent of r and the R term is independent of z. The value of the constant will be determined by the boundary conditions. In fact, we will have a different constant for each k in (81). Here, we have chosen the contstant to be negative if $C > 0$, in anticipation of the forthcoming discussion.

We start with solving for the Z expression. From (84), we have

$$Z'' \;=\; CZ \;\Longrightarrow\; Z \;=\; A \sinh(\alpha z) + B \cosh(\alpha z), \tag{85}$$

with $\alpha = \sqrt{C}$. (We are at liberty to choose the positive square root because the negative sign can be absorbed in the integration constants A and B.[3]) In the general case, we have

$$Z_k(z) \;=\; A_k \sinh(\alpha_k z) + B_k \cosh(\alpha_k z). \tag{86}$$

[3]Note that if we were to find that $C < 0$, the solution (85) would naturally default to the correct solution in terms of sines and cosines.

Applying the boundary condition (80) at $z = 0$ yields

$$-1 \;=\; \frac{\partial T}{\partial z} \;=\; \sum_{k=1}^{N} R_k(r) A_k \alpha_k. \tag{87}$$

It is clear that we need a set of expansion coefficients, $\beta_k := A_k \alpha_k$ such that (87) holds,

$$\sum_{k=1}^{N} \beta_k R_k(r) \;\equiv\; -1, \tag{88}$$

for $0 \le r < 1$. First, we identify the family of radial functions, R_k.

From (84), each of the R_k functions must satisfy

$$R'' + \frac{1}{r} R' + CR \;=\; 0, \tag{89}$$

with boundary conditions $R(0) < \infty$ and $R(r = 1) = 0$. Here, C is a constant that depends on k. The differential equation would be satisfied by the Bessel function $J_0(r)$ if C were equal to unity but the boundary condition at $r = 1$ would not. Fortunately, the fact that C is a free parameter allows us to meet both conditions by using a simple change of variables.

Let $r = \gamma s$ and consider

$$R(r) \;=\; R[r(s)] \;=\; \tilde{R}(s). \tag{90}$$

Using the chain rule, we have

$$R'(r) \;=\; \frac{1}{\gamma} \frac{d\tilde{R}}{ds}, \qquad R''(r) \;=\; \frac{1}{\gamma^2} \frac{d^2\tilde{R}}{ds^2}, \qquad \frac{1}{r} R'(r) \;=\; \frac{1}{\gamma^2} \frac{d\tilde{R}}{ds}. \tag{91}$$

Inserting these expressions into (89) and multiplying through by γ^2 leads to

$$\frac{d^2\tilde{R}}{ds^2} + \frac{1}{s} \frac{d\tilde{R}}{ds} + (\gamma^2 C)\tilde{R} \;=\; 0, \qquad \left.\frac{d\tilde{R}}{ds}\right|_{s=0} = 0, \qquad \tilde{R}(1/\gamma) = 0, \tag{92}$$

which is the Bessel equation solved by $\tilde{R}(s) = J_0(s)$, provided that $\gamma^2 C = 1$. We can thus consider a family of solutions of the form

$$R_k(r) \;=\; J_0(s) \;=\; J_0(r/\gamma_k). \tag{93}$$

At $r = 1$ we have $R_k = J_0(1/\gamma_k) = 0$, which implies that

$$\frac{1}{\gamma_k} \;=\; \xi_k, \qquad k = 1, \ldots, N, \tag{94}$$

where ξ_k is the kth zero of $J_0(\xi)$. Thus, our radial expansion functions are

$$R_k(r) \;=\; J_0(\xi_k r), \qquad k = 1, \ldots, N. \tag{95}$$

(Note the similarity between these functions and a more famous class of eigenfunctions, $\sin(k\pi x)$. The kth root of sin is $\eta_k = k\pi$. In both cases, the class of functions is defined by the multiplier in the argument—the kth root—rather than by changing from one function to another.)

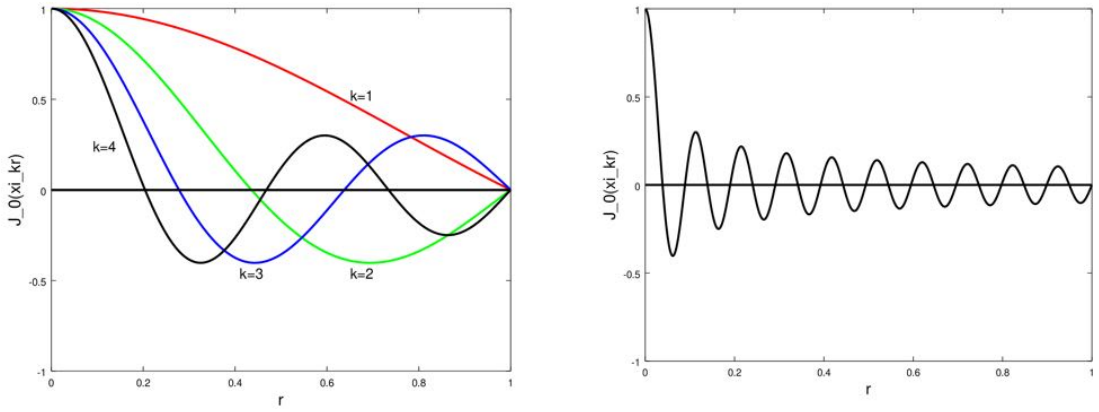


Figure 4: Scaled Bessel functions, $J_0(\xi_k r)$, $r \in [0,1]$, for $k = 1, \ldots, 4$ (left) and $k = 20$ (right), with ξ_k the kth root of $J_0(\xi)$.

To complete the construction of our solution, we now use the scaling relationship (88) that derives from the unit-flux boundary condition at $z = 0$. With the Bessel-function basis, we have

$$\sum_{k=1}^{N} \beta_k J_0(\xi_k r) \quad \equiv \quad -1. \tag{96}$$

Fortunately, $J_0(\xi_k r)$ forms an orthogonal basis under the radially-scaled inner product on $[0,1]$. That is,

$$\int_0^1 J_0(\xi_i r) J_0(\xi_j r)\, r\, dr \quad = \quad \sigma_i \delta_{ij}, \tag{97}$$

where δ_{ij} is the Kronecker delta function that is 1 when $i = j$ and 0 otherwise. If we assume that a given function $f(r)$ has the form

$$f(r) \quad = \quad \sum_{k=1}^{N} J_0(\xi_k r)\beta_k, \tag{98}$$

we can use the orthogonality relationship (97) to find the coefficients β_k by multiplying both sides of (98) by $J_0(\xi_l r)$ and integrating.

$$\int_0^1 J_0(\xi_l r) f(r)\, r\, dr \quad = \quad \sum_{k=1}^{N} \int_0^1 J_0(\xi_l r) J_0(\xi_k r) \beta_k\, r\, dr \;=\; \sigma_l \beta_l. \tag{99}$$

Solving for β_l (and switching l to k), we have

$$\beta_k \quad = \quad \frac{\int_0^1 J_0(\xi_k r) f(r)\, r\, dr}{\int_0^1 J_0(\xi_k r) J_0(\xi_k r)\, r\, dr}. \tag{100}$$

For our particular case where $f(r) \equiv -1$, we have

$$\beta_k \quad = \quad \frac{-\int_0^1 J_0(\xi_k r)\, r\, dr}{\int_0^1 J_0(\xi_k r) J_0(\xi_k r)\, r\, dr}. \tag{101}$$

We summarize the coefficients introduced so far.

$$\alpha_k \;=\; \sqrt{C_k} \;=\; \xi_k, \tag{102}$$

$$A_k \;=\; \beta_k/\xi_k, \tag{103}$$

with β_k given by (101). Finally, using the boundary condition at $z = L$, we have

$$B_k \;=\; -A_k \frac{\sinh \xi_k L}{\cosh \xi_k L}. \tag{104}$$

Note that for any argument $\zeta > 20$, $\sinh(\zeta) \approx \cosh(\zeta)$ to within machine precision in IEEE 64-bit floating point, which implies $B_k \approx -A_k$. Thus, for $L > 20$, we have

$$T(r,z) \;=\; \sum_{k=1}^{N} \left[A_k \sinh(\alpha_k z) + B_k \cosh(\alpha_k z)\right] J_0(\xi_k r) \tag{105}$$

$$\sim \; \sum_{k=1}^{N} A_k \left[\sinh(\xi_k z) - \cosh(\xi_k z)\right] J_0(\xi_k r). \tag{106}$$

$$= \; -\sum_{k=1}^{N} A_k e^{-\xi_k z} J_0(\xi_k r). \tag{107}$$

An important engineering question here is, *What is the maximum temperature at $z = 0$?*, which can be answered by evaluating (105) at $z = 0$.

$$T(r,0) \;=\; \sum_{k=1}^{N} \left[A_k \sinh(\alpha_k 0) + B_k \cosh(\alpha_k 0)\right] J_0(\xi_k r) \tag{108}$$

$$= \; \sum_{k=1}^{N} B_k J_0(\xi_k r) \tag{109}$$

$$= \; \sum_{k=1}^{N} \frac{1}{\xi_k} \, \frac{\int_0^1 J_0(\xi_k \tilde{r})\, \tilde{r}\, d\tilde{r}}{\int_0^1 J_0(\xi_k \tilde{r}) J_0(\xi_k r)\, \tilde{r}\, d\tilde{r}} \, \frac{\sinh \xi_k L}{\cosh \xi_k L} \, J_0(\xi_k r). \tag{110}$$

In the table below, we list the first few values of ξ_k, the numerator and denominator of (101), β_k, and $-A_k$. The integrals in the numerator and denominator are evaluated using Gauss-Legendre quadrature of sufficiently high order. (How high is high enough?) We see that the A_ks are alternating in sign, which is helpful for convergence near $r = 0$, but they do not rapidly converge to zero. We can expect that we might need a rather large value of N to get significant accuracy for this case. We will see later that orthogonal polynomials provide far more rapidly-convergent bases than that derived from the eigenbases [cosh, sinh, J_0]. In Fig. 4, we plot a few of the radial basis functions, $J_0(\xi_k r)$ for various values of k.

k	xi_k	num (101)	den (101)	beta_k	-A_k
1	2.404825e+00	2.158774e-01	1.347570e-01	1.601974e+00	6.661500e-01
2	5.520078e+00	-6.164130e-02	5.789006e-02	-1.064799e+00	-1.928956e-01
3	8.653727e+00	3.136824e-02	3.684317e-02	8.513991e-01	9.838525e-02
4	1.179153e+01	-1.971412e-02	2.701878e-02	-7.296452e-01	-6.187873e-02
5	1.493091e+01	1.383347e-02	2.133071e-02	6.485236e-01	4.343494e-02
6	1.807106e+01	-1.038836e-02	1.762105e-02	-5.895428e-01	-3.262358e-02
7	2.121163e+01	8.168435e-03	1.501053e-02	5.441801e-01	2.565479e-02

Heat Equation Exercise. Compute the solution to the cylindrical conduction problem given by (79) for a cylinder radius $R_{cyl} = 1$ and domain length $z = L = 20$. Spefically, look for a truncated series solution of the form (81) for several values of N (say, $N = 1, 2, \ldots, 50$).

You will need to find the roots of J_0, ξ_k, for $k = 1, \ldots, 50$. (Matlab provides besselj() as a function.) Write a secant method with initial guesses of $\xi' = \xi_{k-1} + \pi$ and $\xi'' = \xi' + 0.1$, using $\xi' = 2$ as the guess for ξ_1. Iterate until $|J_0(\xi_k)| < 10^{-14}$. How many secant iterations are required to find the roots to this tolerance for $k = 1, \ldots, 50$? (Plot iter vs. k, where iter is the number of iterations and k is the root number.)

Use a quadrature rule to evaluate the integrals in β_k (100). The Gauss-Lobatto Legendre (GLL) points and weights provided by the `zwgll` routine will be highly effective here. Be sure to use enough quadrature points, especially for the higher values of k, where $J_0(\xi_k r)$ is highly oscillatory. Demonstrate that your quadrature rule doesn't influence the solution (i.e., that all your integrals are converged). One way to do this to successively double the number of quadrature points until the change in the quadrature output is $< 10^{-15}$.

On two separate graphs, plot your solution, $T(r, z = 0)$, and the derivative, $T_z(r, z = 0)$ for $r \in [0, 1]$ and $N = 1,\ 2,\ 3, \ldots, 9$ (two graphs, nine curves each). (The z-derivative can be found by differentiating (86) and evaluating at $z = 0$.) Summarize what you observe.

What is the maximum temperature in the domain, $T_{\max}$?

Define M_N as the maximum of (81) for a given N:

$$M_N := \max_{r,z \in \Omega} T_N(r, z). \tag{111}$$

Graph $|M_N - M_{50}|$ vs N for $N = 1, \ldots, 49$ using an appropriate plot format. What is the rate of convergence of this solution? Is it exponential? Is it algebraic (i.e., $O(N^s)$)? If it is algebraic, what is the value of s? How does your plot inform this assessment?

Consider now a case where L is variable. Plot M_{50} vs. L. What is the observed behavior? What is the physical explanation for this behavior?

Overall, what is your assessment about the rate of convergence of this approach to solving this problem?

4 L^2-Projection in 1D

We return to the general approximation problem introduced earlier. As a starting point, we consider L^2-projection in 1D. Here, the objective is to find a function $u(x)$ satisfying

$$u = \underset{w \in X^N}{\operatorname{argmin}} \|w - \tilde{u}\|_2. \tag{112}$$

That is, we seek to find the element u in the finite-dimensional subspace X^N that is *closest to* $\tilde{u}$ in the 2-norm. (Later, when solving PDEs, we will choose other norms, but the principle will be the same.)

We introduce the inner product and associated norm

$$(f, g) := \int_\Omega f\, g\, dx \tag{113}$$

$$\|f\|_2 := \sqrt{(f, f)}. \tag{114}$$

We define L^2 as the space of functions f for which $\|f\|_2 < \infty$ and we assume that X^N is a finite-dimensional space satisfying $X^N \subset L^2$. For complex-valued functions, we modify (113) to use the complex conjugate f^*,

$$(f, g) := \int_\Omega f^*\, g\, dx, \tag{115}$$

but for the moment we restrict our attention to the case of real functions. The inner-product (113) is a *symmetric*, *bilinear* form, meaning,

$$(v, u) = (u, v) \qquad \text{(symmetric)} \tag{116}$$

$$(\alpha v + w, u) = \alpha(v, u) + (w, u), \qquad \text{(bilinear)} \tag{117}$$

for suitable functions u, v, $w \in L^2(\Omega)$ and constant α.

Taking $X^N = \text{span}\{l_j(x)\}$, we can write for any $u \in X^N$,

$$u(x) = \sum_{j=0}^{N} l_j(x)\hat{u}_j, \tag{118}$$

which expresses u as a function of preselected basis functions, l_j, and unknown basis coefficients, $\hat{u}_j$. Let $\Phi(u) := \|u - \tilde{u}\|^2$ be the objective function to be minimized. From the linearity of the inner product $(\cdot, \cdot)$ we have,

$$\Phi(u) := \left(\sum_{j=0}^{N} l_j\hat{u}_j - \tilde{u}, \sum_{k=0}^{N} l_k\hat{u}_k - \tilde{u} \right) \tag{119}$$

$$= \sum_{k=0}^{N}\sum_{j=0}^{N} \hat{u}_k B_{kj}\hat{u}_j - \sum_{j=0}^{N} b_j\hat{u}_j - \sum_{k=0}^{N} b_k\hat{u}_k + (\tilde{u}, \tilde{u}). \tag{120}$$

Here, we have introduced the *mass matrix*, B, with entries

$$B_{jk} := (l_j, l_k) \tag{121}$$

and the data entries

$$b_j \;:=\; (l_j, \tilde{u}). \tag{122}$$

At the global minimum, we require stationarity of Φ with respect to each unknown $\hat{u}_i$,

$$\frac{\partial \Phi}{\partial \hat{u}_i} \;=\; \frac{\partial}{\partial \hat{u}_i}\left[\sum_{k=0}^{N}\sum_{j=0}^{N}\hat{u}_k B_{kj}\hat{u}_j \;-\; \sum_{j=0}^{N} b_j\hat{u}_j \;-\; \sum_{k=0}^{N} b_k\hat{u}_k \;+\; (\tilde{u},\tilde{u})\right] \tag{123}$$

$$\;=\; 2\sum_{j=0}^{N} B_{ij}\hat{u}_j \;-\; 2b_i \;=\; 0, \qquad\qquad i = 0,\ldots,N. \tag{124}$$

In matrix form,

$$B\underline{u} \;=\; \underline{b}, \tag{125}$$

where $\underline{u} = [\hat{u}_0,\, \hat{u}_1, \ldots, \hat{u}_N]^T$ and $\underline{b} = [b_0,\, b_1, \ldots, b_N]^T$.

If the basis functions are orthogonal with respect to our chosen inner product then B is diagonal with entries

$$B_{ij} \;=\; (l_i, l_j)\delta_{ij}, \tag{126}$$

where δ_{ij} is the Kronecker delta function satisfying $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. In this orthogonal case, the projection of $\tilde{u}$ onto $X^N :=$*span*$\{l_0, \ldots, l_N\}$ takes the simple form

$$u(x) \;=\; \sum_{j=0}^{N} \frac{(l_j, \tilde{u})}{(l_j, l_j)} l_j(x) = \sum_{j=0}^{N} l_j(x)\hat{u}_j. \tag{127}$$

We see that u is an element of X^N and inherits units of $\tilde{u}$ (i.e., is proportional to $\tilde{u}$) independent of the magnitude of l_j, two properties that are expected of a projection.

An important property of projection onto *orthogonal bases* satisfying (126) is that each $\hat{u}_j$ can be computed independently,

$$\hat{u}_j \;=\; \frac{(l_j, \tilde{u})}{(l_j, l_j)}. \tag{128}$$

The significance of this property is that one realizes successive improvements to the solution through incremental updates, without the need to recompute the previous basis coefficients. Suppose we define $u_N(x) = u(x)$ as in (127). Then a new approximation will be given by

$$u_{N+1}(x) \;=\; \sum_{j=0}^{N+1} l_j(x)\hat{u}_j \;=\; u_N(x) \;+\; l_{N+1}(x)\hat{u}_{N+1}. \tag{129}$$

Because u_{N+1} is a *projection* onto a larger space, $X^{N+1} \supset X^N$, the error will satisfy $e_{N+1} \leq e_N$. If each l_j has a bounded norm then rapid decay of the coefficients $\hat{u}_{N+1}$ will indicate rapid convergence of (127).

An example of an L^2-orthogonal basis is the family of Legendre polynomials, $P_k(x) \in \mathbb{P}_k$, which satisfy

$$(P_i, P_j) \;=\; \int_{-1}^{1} P_i(x)P_j(x)\,dx \;=\; \frac{2}{2j+1}\delta_{ij}. \tag{130}$$

The value on the right of (130) derives from normalizing the polynomials such that $P_j(1) = 1$, $j = 0, 1, \ldots$

Another L^2-orthogonal set are the Fourier bases , $\phi_k = e^{ikx}$ on $\Omega := [-\pi, \pi]$, where $i := \sqrt{-1}$. Here, we use (115),

$$(\phi_j, \phi_k) = \int_{-\pi}^{\pi} e^{-ijx} e^{ikx}\, dx \tag{131}$$

$$= \int_{-\pi}^{\pi} e^{i(k-j)x}\, dx \tag{132}$$

$$= 2\pi\delta_{jk}. \tag{133}$$

Consider a Fourier expansion of the form

$$f(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k\, e^{ikx}. \tag{134}$$

The Fourier coefficients are found by taking the (Hermitian) inner product of both sides with e^{ilx},

$$\int_0^{2\pi} e^{-ilx}\, f(x)\, dx = \sum_{k=-\infty}^{\infty} \hat{f}_k \int_0^{2\pi} e^{-ilx}\, e^{ikx}\, dx. \tag{135}$$

On the right, only the $k = l$ term is nonzero, so we have (substituting k for l),

$$\hat{f}_k = \frac{\int_0^{2\pi} e^{-ikx}\, f(x)\, dx}{\int_0^{2\pi} e^{-ikx}\, e^{ikx}\, dx} = \frac{1}{2\pi} \int_0^{2\pi} e^{-ikx}\, f(x)\, dx. \tag{136}$$

Notice the important difference between (134), which is *Fourier synthesis*—generating a function as a sum of harmonics, and the inverse, (136), which is *Fourier analysis*—finding a set of Fourier coefficients given a 2π-periodic function, $f(x)$. The former simply involves sums; the latter involves integration and scaling by $1/2\pi$.

Invariably, we will use numerical quadrature to evaluate the integral in (136), which will mean that we have a *weighted sum*. For a given integrand $g(x)$, we can use an N-point rectangle rule of the form

$$\mathcal{I}_{rr} = \sum_{j=0}^{N-1} w_j\, g(x_j) \tag{137}$$

with $w_j = 2\pi/N$. (For periodic functions, this is a Gauss quadrature formula.) In this case, our Fourier coefficients are approximated as

$$\hat{f}_k = \frac{1}{2\pi} \sum_{j=0}^{N-1} e^{-ikx_j}\, f(x_j)\, w_j = \frac{1}{N} \sum_{j=0}^{N-1} e^{-ikx_j}\, f(x_j). \tag{138}$$

Now that we have replaced the continuous process of integration with a discrete, approximate, summation operation; the parallel between synthesis and analysis is now relatively clear.

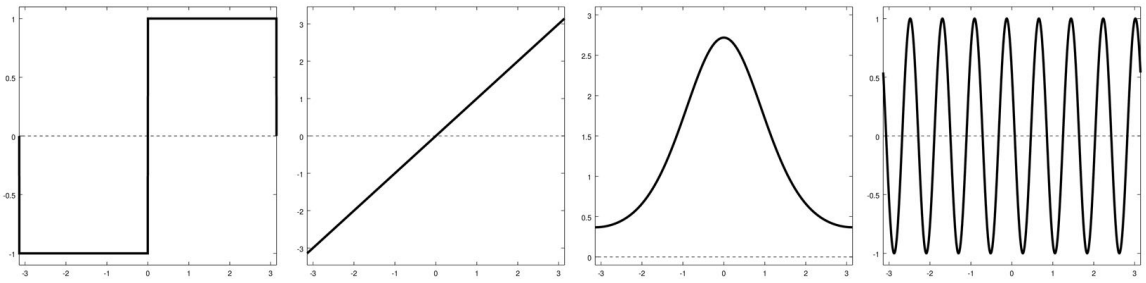


Figure 5: 2π-periodic functions on $[-\pi, \pi)$: (a) $f =$sign(x), (b) $f = x$, (c) $f = e^{\cos(x)}$, (d) $f = \cos(8x + 1)$.

We summarize the standard discrete (i.e., *computable*) Fourier relationships for a 2π-periodic function, $\tilde{u}(x)$, in the following.

$$\textit{Fourier Analysis:} \qquad \hat{u}_k = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} e^{-ikx_j}\, \tilde{u}(x_j), \quad j = 0, \ldots, N-1. \tag{139}$$

$$\textit{Fourier Synthesis:} \quad u_N(y_l) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} e^{iy_l k}\, \hat{u}_k, \quad l = 0, \ldots, M-1 \tag{140}$$

Note that the discrete transformation matrix $\hat{F}_{kj} = e^{-ikx_j}$ in (139) is square. It transforms N real values into N complex values (whose entries are complex conjugates, $\hat{u}_{-k} = \hat{u}_k^*$). For *synthesis*, we can of course have an arbitrary number of output points. If the output points, y_l are equal to the input points, x_l, then (140) is the inverse operation of (139). If, on the other hand the output points differ from the input, then the pair (139)–(140) represents an *interpolation operator*. For purposes of interpolation, the $M \times N$ synthesis matrix, $F_{lk} = e^{iy_l k}$, will generally be rectangular, with more rows (output) than columns (input). We see that, through the use of a (highly accurate!) discrete quadrature rule, we have effectively turned the projection operator into an interpolation operator. The latter are generally much easier to use and, if done with care, generate results whose accuracy differs from that of projection in a minor and acceptable way.

Note that if $\tilde{u}(x)$ has significant wave-number content in modes $k > N/2$ then those modes will be *aliased* onto the subsampled integrand in (139). Aliasing is a significant concern in the simulation of turbulent and advection-dominated flows but is easily addressed by using $M > N$ quadrature points in (138). We revisit this topic later in the context of multidimensional advection-diffusion.

4.1 Fourier Interpolation Exercise

1. Write a code to generate a *real* matrix F that performs Fourier interpolation from N uniformly-spaced points on $[a, b)$ to M uniformly-spaced points on $[a, b)$. F should be $M \times N$. Verify that $F = I$ when $M = N$. Verify that your code works for N, M even and odd.

2. Take N=10, 20, 40, 80, and $M = 10N + 10$ and plot the input points (x_j, u_j), $j = 0, \ldots, N-1$, and the output (solid black line).

3. For each of the functions, what is your maximum pointwise error as a function of N?

4. For each of the functions, plot ($|k|$,$|\hat{u}_k|$) on a loglog scale for the case N=80.

Comment on your observations, particularly regarding the relationship between the answers to questions **3** and **4**.

4.2 A Spectral BVP Example

We illustrate the use of orthogonal bases for the solution of the differential equation on $\Omega := [-\pi, \pi]$,

$$\mathcal{L}\tilde{u} \;:=\; -\frac{d^2\tilde{u}}{dx^2} \;+\; \tilde{u} \;\;=\;\; f, \tag{141}$$

subject to periodic boundary conditions $\tilde{u}(-\pi) = \tilde{u}(\pi)$.[4] We will show that the question of convergence essentially amounts to approximation properties of the chosen basis for the given data, f, and on the smoothness of f itself. Our approach closely follows the presentations in [Gottlieb & Orszag '77] and [CHQZ2].

We take as our numerical approximation the truncated Fourier series,

$$u(x) \;\;=\;\; \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{u}_k \, e^{ikx}. \tag{142}$$

Note that

$$u'(x) \;\;=\;\; \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} ik\,\hat{u}_k \, e^{ikx}, \qquad u''(x) \;=\; \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} -k^2\,\hat{u}_k \, e^{ikx}, \tag{143}$$

from which

$$\mathcal{L}u(x) \;\;=\;\; \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{u}_k \left(k^2+1\right) \, e^{ikx}. \tag{144}$$

Assuming that a series expansion for $f(x)$ exists,

$$f(x) \;\;=\;\; \sum_{j=-\infty}^{\infty} \hat{f}_k \, e^{ikx}, \tag{145}$$

an orthogonal projection (of the residual) can be found by integrating both sides of (141) with respect to weight functions $\phi_l^* = e^{-ilx}$, $l = \frac{N}{2} \ldots \frac{N}{2} - 1$,

$$\int_\Omega e^{-ilx} \mathcal{L}u \, dx \;\;=\;\; \int_\Omega e^{-ilx} f \, dx. \tag{146}$$

With (144), (145), and the orthogonality of the ϕ_ks, the contraction (146) yields

$$\hat{u}_l \left(l^2+1\right) \;\;=\;\; \hat{f}_l \;=\; \frac{1}{2\pi} \int_\Omega e^{-ilx} f \, dx, \tag{147}$$

The N-term approximation is therefore

$$u(x) \;\;=\;\; \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \frac{\hat{f}_k}{k^2+1} e^{ikx}, \tag{148}$$

[4] This problem is a specific case of a second-order low-pass filter of the form $-\delta^2 \bar{f}'' + \bar{f} = f$, which tends to suppress oscillations in the data, f, that are below the length scale δ. A general discussion of PDE-based filters of order $2m$ in 2D and 3D is presented in [Mullen & Fischer '98]. Analogous general higher-order filters for graphics applications have been developed by Gabriel Taubin and co-workers [Taubin'95;Taubin,Zhang,&Golub'96].

The corresponding error, $e_N := \tilde{u} - u$, is

$$\boxed{e_N(x) \;=\; \sum_{|k|>\frac{N}{2}} \frac{\hat{f}_k}{k^2+1} e^{ikx},} \tag{149}$$

and

$$\|e_N\|_\infty \;\leq\; \sum_{|k|>\frac{N}{2}} \frac{|\hat{f}_k|}{k^2+1}. \tag{150}$$

(Here, we take $|k| > \frac{N}{2}$ to be the complement of the indices included in the N-term expansion (142).) Equation (150) gives us the L^∞ error. We have a similar formula for the L^2 error,

$$\|e_N\|_2 \;=\; \left[\sum_{|k|>\frac{N}{2}} \left(\frac{|\hat{f}_k|}{k^2+1} \right)^2 \right]^{\frac{1}{2}}. \tag{151}$$

Remark 1. From (149) we see that e_N will tend to zero as $N \longrightarrow \infty$ even if the Fourier coefficients of f are order-unity constants bounded away from zero because of the $O(k^{-2})$ damping of the high wavenumber coefficients of u. This second-order smoothing (or regularizing) effect is an intrinsic feature of the diffusion operator.

Remark 2. More important to the rate of convergence of our spectral method, however, is the amplitude of the the Fourier coefficients of f. That is, what is the behavior of $\hat{f}_k$ as $k \longrightarrow \infty$? If $f(x)$ is smooth and 2π-periodic, these coefficients will decay rapidly, as we demonstrate momentarily. If, on the other hand, $f(x)$ has *limited* regularity (meaning a finite number of bounded derivatives on Ω), then the Fourier coefficients and, hence, the error (149) will not diminish rapidly.[5] We illustrate these behaviors with several examples.

A Square Wave

Let us first consider a square wave, $f(x) =$sign(x), on $\Omega = [-\pi, \pi]$. The Fourier coefficients are

$$\hat{f}_k \;=\; \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-ikx} f(x)\, dx \;=\; -\frac{i}{\pi} \int_0^{\pi} \sin(kx) f(x)\, dx, \tag{152}$$

where we have exploited the fact that $f(x)$ is an odd function. On the open interval $(0, \pi)$, $f \equiv 1$, so we find

$$\begin{aligned} \hat{f}_k \;=\; \frac{i}{\pi k} \cos kx \Big|_0^{\pi} \;&=\; -\frac{2i}{\pi k} \qquad \text{for } k \text{ odd,} \\ &=\; 0 \qquad\quad \text{otherwise.} \end{aligned} \tag{153}$$

Basic calculus indicates that the Fourier coefficients for the square wave decay as k^{-1} (which is not very fast). It is also useful to have a graphical understanding of why this is the case. In Fig. 6 we plot the integrand of (152) for $f = 1$ along with a shaded region that indicates the nontrivial contributions to the integral. It is clear that, for k *even*, the symmetry of $\sin kx$ leads to a null

[5] Note that one of the singular features of orthogonal expansions is that they provide closed-form error estimates, such as (149), that are not readily available in the more general case where B in (125) is full.

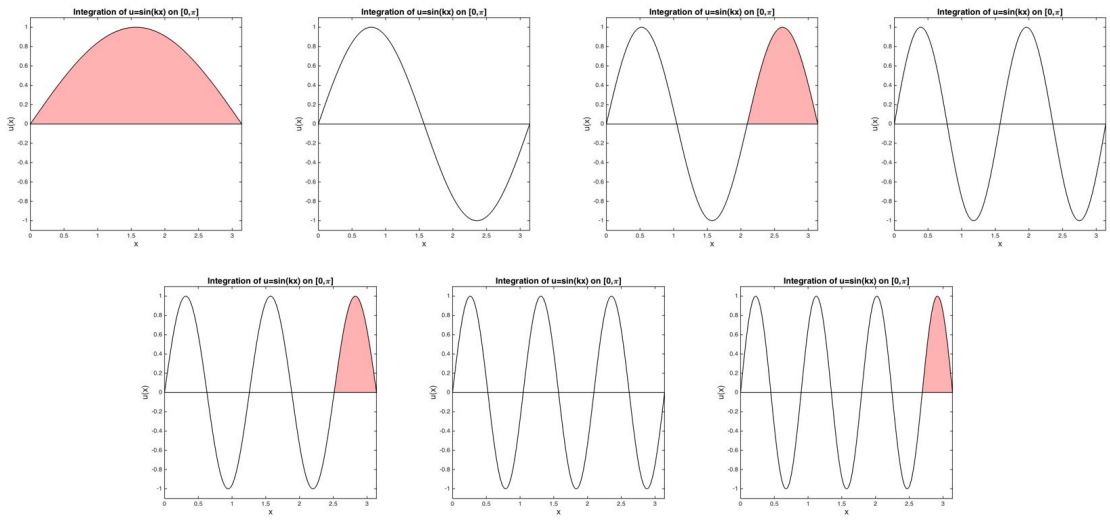


Figure 6: Integrands for $\sin kx$ on $(0, \pi)$ with shaded regions indicating nontrivial contribution to the integral.

integral. For k odd, on the other hand, we have one lobe of the sine function that contributes to the integral. Its base is diminishing in width, but the sequence decays slowly: π, $\pi/3$, $\pi/5$, and so on, which gives rise to the k^{-1} fall off in (153). Since the original integral is $\pi \times \frac{2}{\pi} = 2$, we arrive at the value given in (153).[6]

From (151), the L^2 error for the Fourier spectral appoximation to (141) when f is a square wave is

$$||e_N||_2^2 \;\sim\; C\left[\frac{1}{N^6} + \frac{1}{(N+2)^6} + \frac{1}{(N+4)^6} + \ldots\right] \;\sim\; \frac{C'}{N^5}, \tag{154}$$

from which we'd expect $||e_N||_2 = O(N^{\frac{5}{2}})$. Similarly, the L^∞ error is

$$||e_N||_\infty \;\sim\; C\left[\frac{1}{N^3} + \frac{1}{(N+2)^3} + \frac{1}{(N+4)^3} + \ldots\right] \;\sim\; \frac{C'}{N^2}. \tag{155}$$

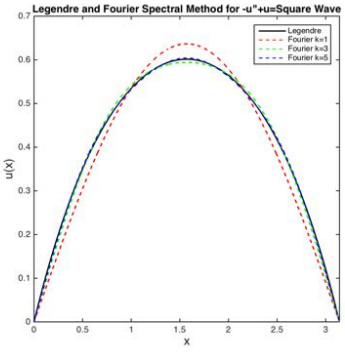
Here and in (154) the Cs are (different) constants that are independent of N.

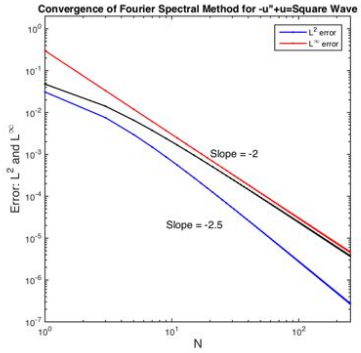
We illustrate the error behavior by comparing against a solution to (141) on $\Omega' := [0, \pi]$ with homogeneous Dirichlet conditions, which is computed using a Legendre spectral method that will be discussed shortly. We remark that the Legendre method converges to $\approx 10^{-14}$ with about 10–12 modes, but we evaluate it on a set with $M > N$ points to avoid aliasing errors when computing the error norm for the oscillatory Fourier approximation. In the accompanying boxed inset we provide the matlab script `sine_square`, along with the figures it produces. The upper figure shows the Legendre solution and first few Fourier approximations. The lower figure shows the convergence plots that confirm the rates (154) and (155).

[6]The $\frac{2}{\pi}$ is very close to the $\frac{2}{3}$ that would result if the function were a parabola having the same height and roots. The fact that $\frac{2}{\pi} < \frac{2}{3}$ tells us that the sine function is flatter than the parabola near the roots. Indeed, its second derivative is zero there.

Fourier and Legendre Examples The matlab script `square_wave.m` (available on the course webpage) implements solutions to (141).

```
%  SOLVE -d^2 u" + u = f on [0,pi] (Legendre) and [-pi,pi] (Fourier)
%
%  for f=square-wave.

format compact; format longe; hold off; clear all

N=256; M=400;  %% Over-sample for error checking
L  = pi; d=1; d2=d*d;

[Ah,Bh,Ch,Dh,z,w]=semhat(M);
Ab = (2/L)*Ah; Bb = (L/2)*Bh; Ib = eye(M+1); R = Ib(2:end-1,:);
A=R*Ab*R'; B=R*Bb*R'; H = d2*A + B;

z=L*(1+z)/2; w=L*w/2;  % Scale points and weights to [0,L]
xb=z; fb=1+0*xb;       % Set RHS for Legendre spectral soln
us= R'*(H\(R*Bb*fb));  % Solve Legendre system
plot(xb,us,'k-','linewidth',1.4); hold on;


uf=0*xb;  %% Initialize Fourier solution on GLL pts
for k=1:2:N; k1=k+1;
  sk = sin(k*xb);
% fk = 2*(w'*sk)/(2*pi); % 2 in numerator because integrate over [-pi,pi]
  fk = 2/(k*pi);         % 2 in numerator because integrate over [-pi,pi]
  uk = 2*fk./(1+d2*k*k); % 2 in numerator because +/- k mode.
  uf = uf + uk*sk;
  ef=uf-us;
  em(k1/2)=max(abs(ef));
  ef2(k1/2) = sqrt(ef'*Bb*ef/pi);
  nN(k1/2)  = k;
  if k==1; plot(xb,uf,'r--'); end;
  if k==3; plot(xb,uf,'g--'); end;
  if k==5; plot(xb,uf,'b--'); end;
end;
title('Legendre and Fourier Spectral Method for -u"+u=Square Wave','fontsize',12)
xlabel('x','fontsize',14); ylabel('u(x)','fontsize',14);
axis square; axis([0 pi 0 .7]);
legend('Legendre','Fourier k=1','Fourier k=3','Fourier k=5')
print -dpng 'legendre_fourier.png'; hold off


loglog(nN,ef2,'b.-',nN,0.3*nN.^(-2),'r.-',nN,em,'k.-','linewidth',1.3)
title('Convergence of Fourier Spectral Method for -u"+u=Square Wave','fontsize',12)
xlabel('N','fontsize',14); ylabel('Error: L^2 and L^\infty','fontsize',14);
axis square; axis([0 N 1e-7 2]); legend('L^2 error','L^\infty error')
text(9,2e-5,'Slope = -2.5','fontsize',12); text(20,1e-3,'Slope = -2','fontsize',12);
print -dpng 'fourier_sq_convergence.png'
```





4.3 A Smooth Example

In the previous section we saw that the convergence of the Fourier spectral method is only algebraic when u has limited regularity. Although u was smooth, its second derivative, $u'' = u - f$, had a jump discontinuity. We now consider the case where f and its first $m-1$ derivatives are 2π-periodic and has f has m bounded derivatives. We begin with the formula for the Fourier coefficients (147), which we integrate by parts.

$$\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} \underbrace{f}_{v}\,\underbrace{e^{-ikx}\,dx}_{du} \tag{156}$$

$$= \frac{1}{2\pi ik}\left[\int_0^{2\pi} f'\,e^{-ikx}\,dx \;-\; fe^{-ikx}\Big|_0^{2\pi}\right] \tag{157}$$

$$= \frac{1}{2\pi ik}\int_0^{2\pi} f'\,e^{-ikx}\,dx \;+\; \frac{f(0)-f(2\pi)}{2\pi ik}, \tag{158}$$

$$= \frac{1}{2\pi ik}\int_0^{2\pi} f'\,e^{-ikx}\,dxk \tag{159}$$

where, without loss of generality, we have taken the domain of integration to be $(0, 2\pi)$. Note that the boundary term in (158) vanishes because of the 2π-periodicity assumption on f. We can continue to apply integration-by-parts up to m times to yield

$$\boxed{\hat{f}_k \;=\; \frac{1}{2\pi(ik)^m}\int_0^{2\pi} f^{(m)}\, e^{-ikx}\, dx \;+\; \frac{f^{(m-1)}(0) - f^{(m-1)}(2\pi)}{2\pi(ik)^m}.} \tag{160}$$

Since $f^{(m)}$ is integrable, we have (by the Riemann-Lebesgue lemma)

$$\left|\frac{1}{2\pi(ik)^m}\int_0^{2\pi} f^{(m)}\, e^{-ikx}\, dx\right| \;\ll\; O\left(\frac{1}{k^m}\right). \tag{161}$$

Thus, the leading order contribution to $\hat{f}_k$ as $k \longrightarrow \infty$ comes from the boundary condition terms on the right of (160).

Equation (160) provides the principal error estimate for the Fourier spectral method. We make several remarks.

- If f is infinitely differentiable and 2π periodic, the Fourier coefficients converge more rapidly than any finite power of $1/N$ as $N \longrightarrow \infty$.

  In this case, if $f_N(x)$ is the L^2-projection of f onto X^N (the space spanned by $\phi_k = e^{ikx}$, $k = N/2, \ldots, N/2-1$), then

  $$\|f(x) - f_N(x)\|_\infty \;\sim\; Ce^{-\sigma N}, \tag{162}$$

  where σ is a positive real constant.

- If $\hat{f}_k$ goes to zero like $1/k^m$ and no faster, then $f^{(m-1)}$ is discontinuous. In this case,

  $$f(x) - f_N(x) \;=\; O\left(\frac{1}{N^m}\right) \tag{163}$$

  when x is fixed away from the discontinuity and,

  $$f(x) - f_N(x) \;=\; O\left(\frac{1}{N^{m-1}}\right), \tag{164}$$

  when $|x - x_o| = O(1/N)$ as $N \longrightarrow \infty$, where x_0 is a point of discontinuity of $f^{m-1}(x)$ [Gottlieb & Orszag'77].

- For the infinitely-differentiable case, the error estimate (149) for $-u_{xx} + u = f$ becomes

  $$|e_N| \;\le\; \sum_{|k|>\frac{N}{2}} \frac{|\hat{f}_k|}{k^2+1} \;\sim\; \frac{|\hat{f}_{N/2}|}{(N/2)^2+1} \;\sim\; Ce^{-\sigma N} \;\text{ as } N \longrightarrow \infty, \tag{165}$$

  where σ is a positive real constant. Note that, with such rapid decay, the leading-order term in the series expansion for the error is also a reasonable estimate of the error.

Seemingly well-behaved right-hand sides can also give problems. For example, $f = x$ is a smooth infinitely-differentiable function but it is not 2π-periodic. When used in a Fourier-spectral context, the method effectively "sees" the periodic extension of this function, which means that the data on $(0, 2\pi)$ is replicated on adjacent intervals of length 2π each (e.g., $(2\pi, 4\pi)$, $(4\pi, 6\pi)$, and so on). For $x > 0$, the periodic extension gives rise to the saw tooth function $f = mod(x, 2\pi)$, which is definitely not continuous. The corresponding rate of convergence will be low and the reconstruction will exhibit the *Gibbs phenomenon*.

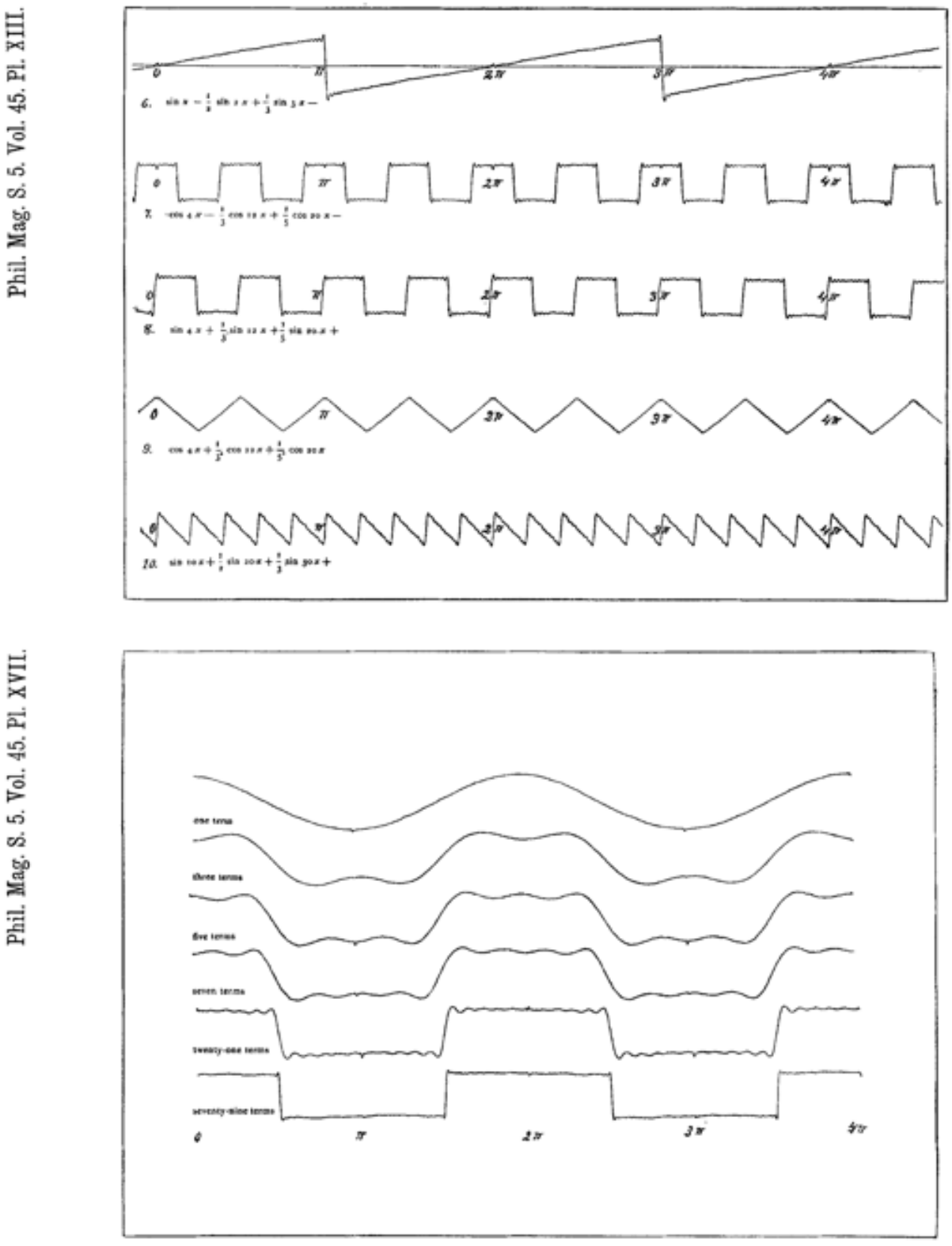


Figure 7: Plates from Michelson and Stratton's 1898 paper describing their harmonic analyzer.

4.3.1 Gibbs Phenomenon

The Gibbs phenomenon is an error in the Fourier[7] reconstruction of functions that have jump discontinuities either within the domain $\Omega = [a, b]$ or at the domain endpoints ($f(a) \neq f(b)$).

[7]Gibbs phenomena also arise in polynomial reconstruction, which can also exhibit a similar but distinct problem known as the *Runge phenomenon*.

Although it was studied in an 1848 paper by Henry Wilbraham from Trinity College in Cambridge, England, the Gibbs phenomenon wasn't known more broadly until Albert A. Michelson (winner of the 1907 Nobel Prize in physics), A.E.H. Love, Henri Poincaré, and Josiah W. Gibbs published an exchange of letters in Nature in 1898–1899. A later paper by Böcher (1906) gave the attribution to Gibbs, who had made a correct analysis of the observations. [S. Gottlieb, Comm. Comp. Phys., **9**, 2011].

Michelson had raised the question of convergence of Fourier sequences after he and University of Chicago colleague, Samuel W. Stratton,[8] had published a paper in the *London, Edinburgh, and Dublin Phil. Mag. and J. of Sci.* (1898) that described results of their mechanical harmonic analyzer. This device is able to rapidly sum sequences of the form $\sum_k a_k \cos(kx)$ or $\sum_k b_k \sin(kx)$ and evaluate the function continuously over a given interval. The pair designed two such devices, one that could sum $N = 20$ modes and would that could sum 80 modes. (UIUC has a 20 mode machine in Altgeld Hall. The American History Museum in DC has both a 20 mode and an 80 mode one.) Michelson and Stratton's 1898 paper contains many plates showing the output of their device, two of which are shown in Fig. 7. Plate XIII shows reconstructions for several classic Fourier sequences generated with up to 80 modes. Plate XVII shows the impact of the number of modes on the reconstruction of a square wave. The Gibbs phenomenon refers to the overshoot observed in the reconstruction of these discontinous examples. While is is clear from Plate XVII that the width of each overshoot diminishes with increasing N, the trend for the *height* is less clear. It is this overshoot that Michelson was aware of when he sent his initial inquiry to Nature about the convergence behavior of infinite Fourier series. Love dismissed the notion and responded with a letter that closed with the insulting remark, *[they]... would be likely to profit by reading some standard treatise dealing with the theory of infinite series, such, for example, as Hobson's "trigonometry."* Poincaré, however, came to the defense of Michelson and in 1899 Gibbs ultimately furnished analysis showing that the amplitude of the overshoot approached a limit of $\approx 18\%$ for the square wave. In a 1906 paper, Böcher coined the label *Gibbs phenomenon*, by which it has been known ever since.

4.4 Eigenfunction Expansions through Sturm-Liouville Problems

As noted in (160), a primary difficulty in Fourier spectral expansions comes from requiring that the data, $f(x)$, and all of its derivatives satisfy periodic boundary conditions. For more general conditions we use eigenfunctions, $\phi_k(x)$, of Sturm-Liouville problems,

$$\mathcal{L}\phi_k \;=\; -\frac{d}{dx}p(x)\frac{d\phi_k}{dx} \;+\; q(x)\phi_k \;\;=\;\; \lambda_k w(x)\phi_k, \tag{166}$$

on $\Omega = (-1, 1)$ with suitable boundary conditions on $\phi_k(x)$ at $x = \pm 1$. Several boundary condition choices are of interest:

$$\begin{aligned} \textit{periodic:} \quad & \phi_k(-1) \;=\; \phi_k(1) \\ \textit{Dirichlet-Dirichlet:} \quad & \phi_k(-1) \;=\; \phi_k(1) \;=\; 0 \\ \textit{Dirichlet-Neumann:} \quad & \left.\tfrac{d\phi_k}{dx}\right|_{\pm 1} \;=\; 0, \;\; \phi_k(\mp 1) \;=\; 0. \end{aligned} \tag{167}$$

[8] Stratton received his B.S. degree in 1884 in mechanical engineering from the Illinois Industrial University at Urbana (later the University of Illinois). Five years later, he was appointed head of the physics department in Urbana. He joined the University of Chicago physics department in 1892, the same year that the department was founded and Michelson was appointed head. Stratton later became the first director of the National Bureau of Standards (now NIST) and ultimately the 8th president of M.I.T.

There are other possibilities (e.g., Neumann-Neumann or Robin), but we will be primarily interested in the first two in this list. We assume in (166) that $p > 0$ on Ω (but it may vanish at the domain endpoints), $q \geq 0$ and $w > 0$.

We define the weighted inner product

$$(v, u)_w \;:=\; \int_\Omega w\, v\, u\, dx, \tag{168}$$

which satisfies $(u, u)_w > 0$ for all nontrivial u on Ω. Presently, we consider both *regular* and *singular* Sturm-Liouville problems, with specific attention to the singular case for which $p(\pm 1) = 0$ as it is this condition that will bypass the boundary requirements on $f(x)$ that otherwise limit exponential convergence.

We begin once again with the approximation problem. Let X^N =span$\{\phi_k\}$ comprise the first N eigenfunctions of (166) associated with $\lambda_1 < \lambda_2 < \cdots < \lambda_N$, and assume that they are scaled so that $(\phi_k, \phi_k)_w = 1$. The ϕ_ks are orthogonal with respect to the w-weighted inner product. To see this, we multiply both sides of (166) by ϕ_j and integrate over Ω,

$$-\int_\Omega \phi_j \left(p\phi_k'\right)' dx \;+\; \int_\Omega q\phi_j\phi_k\, dx \;\;=\;\; \lambda_k \int_\Omega \phi_j\, \phi_k\, w(x)\, dx \;=\; \lambda_k\, (\phi_j, \phi_k)_w. \tag{169}$$

Integrating the term involving $p(x)$ by parts, twice, yields

$$\lambda_k\, (\phi_j, \phi_k)_w \;\;=\;\; \int_\Omega p\phi_j'\phi_k'\, dx \;-\; \underbrace{p\phi_j\phi_k'\big|_{-1}^{1}}_{\phi_j(\pm 1) = 0} \;+\; \int_\Omega q\phi_j\phi_k\, dx \tag{170}$$

$$=\;\; -\int_\Omega \phi_k \left(p\phi_j'\right)' dx \;+\; \underbrace{p\phi_j'\phi_k\big|_{-1}^{1}}_{\phi_k(\pm 1) = 0} \;+\; \int_\Omega q\phi_j\phi_k\, dx \tag{171}$$

$$=\;\; \lambda_j \int_\Omega w(x)\phi_j\, \phi_k\, dx. \tag{172}$$

At each stage of the integration, the boundary terms either cancel each other out (for the periodic case) or vanish (because either homogeneous Neumann or Dirichlet conditions apply). From (172), we must have either $\lambda_j = \lambda_k$ or $(\phi_j, \phi_k)_w = 0$. The solutions to the linear eigenvalue problem (166) are unique—there is only one ϕ_k for a given λ_k, so it must be the case that $(\phi_j, \phi_k)_w = 0$ for $k \neq j$. For the cases that are of interest to this course, we will have $\lambda_k = O(k^2)$.

To approximate $f(x)$, consider a generalized Fourier expansion in terms of the eigenfunctions,

$$f_N(x) \;\;=\;\; \sum_{k=1}^{N} \phi_k(x)\hat{f}_k. \tag{173}$$

Using the orthogonality relationship and under the assumption that the ϕ_ks are normalized to satisfy $(\phi_i, \phi_j) = \delta_{ij}$, we have

$$\hat{f}_k \;\;=\;\; (\phi_k, f)_w \;=\; \frac{1}{\lambda_k}\left[-\int_\Omega f\left(p\phi_k'\right)' \;+\; \int_\Omega qf\phi_k\, dx\right], \tag{174}$$

where the term on the right exploits the definition (166). Integrating (174) by parts yields,

$$\hat{f}_k = \frac{1}{\lambda_k}\left[-\int_\Omega \phi_k \left(pf'\right)' + \int_\Omega qf\phi_k\,dx + pf\phi_k'\big|_{-1}^{1}\right] \tag{175}$$

$$= \frac{1}{\lambda_k}\left[\int_\Omega \phi_k\,\mathcal{L}f\,dx + pf\phi_k'\big|_{-1}^{1}\right] \tag{176}$$

$$= \frac{1}{\lambda_k}\left[\int_\Omega w\,\phi_k\,f_1\,dx + pf\phi_k'\big|_{-1}^{1}\right] \tag{177}$$

$$= \frac{1}{\lambda_k}(\phi_k, f_1)_w + \frac{1}{\lambda_k}\left[pf\phi_k'\big|_{-1}^{1}\right]. \tag{178}$$

Here, we have introduced a generalized (second) derivative of $f(x)$,

$$f_1(x) := \frac{1}{w}\mathcal{L}f = \frac{1}{w}\left[-\frac{d}{dx}p\frac{df}{dx} + qf\right]. \tag{179}$$

If $f(x)$ is sufficiently smooth, then the first term on the right of (178) will be bounded (and in fact will go to zero faster than k^{-2} as $k \longrightarrow \infty$).

As before, the challenge to realizing rapid convergence of the $\hat{f}_k$s in the cases when $f(x)$ is smooth is to make certain that the boundary term vanishes. For special cases of f (e.g., periodic functions when $p = 1$ and $\phi_k = e^{ikx}$) this will be true. For more general cases, we can bypass these overly restictive boundary condition requirements on $f(x)$ if we allow $p(\pm 1) = 0$. This is the *singular Sturm-Liouville problem.* Some common situations and the corresponding eigenfunctions are the orthogonal Legendre and Chebyshev polynomials, expressed in the following table.

Basis	p	q	w	λ_k	Recurrence/Eigenfunction	BCs on ϕ_k
Legendre	$(1-x^2)$	0	1	$k(k+1)$	$P_{k+1} = (2k+1)xP_k - kP_{k-1}$	bounded
Chebyshev	$\sqrt{1-x^2}$	0	$\frac{1}{\sqrt{1-x^2}}$	k^2	$T_{k+1} = 2xT_k - T_{k-1}$	bounded
Fourier	1	0	1	k^2	$\phi_k = e^{ikx}\ \ \Omega = (0, 2\pi)$	periodic
sine	1	0	1	k^2	$\phi_k = \sin kx\ \ \Omega = (0, \pi)$	$\phi_k = 0$

(180)

We see from (180) that $p(x)$ vanishes at the domain endpoints when the eigenfunctions are Legendre or Chebyshev polynomials, but not for the Fourier or sine expansions. The latter correspond *regular* Sturm-Liouville problems for which the boundary term in (178),

$$\frac{1}{\lambda_k}\left[pf\phi_k'\big|_{-1}^{1}\right] \tag{181}$$

will vanish only if $f(x)$ meets certain conditions. On the other hand, in the singular Sturm-Liouville case, this boundary term will vanish provided only that $f\phi_k'$ satisfies the very weak requirement of being bounded at the endpoints. Of course, both the regular and singular cases require smoothness of $f(x)$, but the latter places no strong constraints on the boundary values for f, which very much enlarges the applicable space of spectral methods.

Assuming that $f(x)$ is $2m$ differentiable on Ω and that $p(x)$ vanishes at the boundaries, one can obtain a much sharper estimate for $\hat{f}_k$ through repeated integration-by-parts on the remaining term, (ϕ_k, f_1). Let

$$f_j := \frac{1}{w}\mathcal{L}f_{j-1}, \tag{182}$$

with $f_0 := f$. After m rounds of twice-integration-by-parts, we have

$$\hat{f}_k = \frac{1}{\lambda_k^m}(\phi_k, f_m)_w + \frac{1}{\lambda_k^m}\left[pf_{m-1}\phi_k'\big|_{-1}^{1}\right] \tag{183}$$

$$= \frac{1}{\lambda_k^m}(\phi_k, f_m)_w + \ll \frac{C}{k^{2m}}. \tag{184}$$

The last inequality follows from the Reimann-Lebesgue lemma.

[Make a side box with Legendre polynomials plotted.]

Examples

Here, we look at a few approximation examples, comparing Legendre and sine/cosine function expansions.

To begin, we consider $f(x) = |x|^9$. We have that $f^{(9)} = 9!H(x)$–a multiple of the Heavyside step function–which is integrable but not differentiable. Using $m = 5$ in (183), we anticipate finding that $\hat{f}_k := (\phi_k, f)_w \sim Ck^{-10}$ as $k \longrightarrow \infty$. (A bit of closer analysis is required to get this result because f is 9 times differentiable, not 10.) We note that f is an even function and that the Legendre polynomials are odd for k odd, so only even k will have nontrivial coefficients. In this case, the integrals will be

$$\hat{f}_k = \begin{cases} 0 & k \text{ odd} \\ 2\int_0^1 x^9 \tilde{P}_k(x)\,dx \equiv \sum_{j=1}^M \frac{1}{2}\rho_j x_j^9 \tilde{P}_k(x_j), & k \text{ even}, \end{cases} \tag{185}$$

where $x_j = \frac{1}{2}(1+\xi_j)$, $j = 1, \ldots, M$, are GLL points mapped from $[-1, 1]$ to $[0, 1]$ and $\frac{1}{2}\rho_j$ are the corresponding mapped GLL weights. Moreover, we consider the set of normalized Legendre polynomials,

$$\tilde{P}_k := \sqrt{\frac{2k+1}{2}}P_k, \tag{186}$$

which satisfy $(\tilde{P}_i, \tilde{P}_j) = \delta_{ij}$. (Recall, the standard Legendre polynomials are normalized such that $P_k(1) = 1$.) In (185), the quadrature order M should be chosen high enough such that the integration is exact. For GLL quadrature, this means that $2M - 1 \geq k + 3$, or $M \geq 2 + k/2$. For more general, *non-polynomial*, $f(x)$, one would typically need to use a significantly larger value of M to ensure that the quadrature error is much smaller than the approximation error, at least for the purposes of exploring the isolated effects of approximation error. In practice, for the incompressible Navier-Stokes equations, we typically use $M = N$ for all terms save for the quadratic advection terms, $\mathbf{u} \cdot \nabla \mathbf{u}$ and $\mathbf{u} \cdot \nabla T$, for which we use $M = 3N/2$. We address this *dealiasing* strategy later in the context of multidimensional applications.

Limited Regularity Legendre Expansion Examples

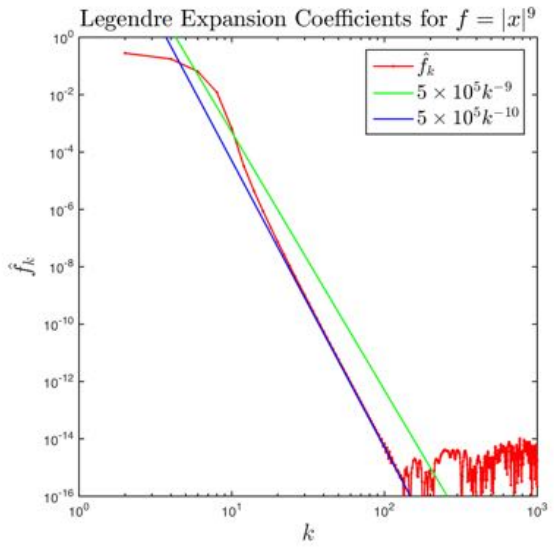
The matlab script `legendre_exp_demo.m` (available on the course webpage) computes the Legendre coefficients for $f(x) = |x|^9$ by integrating $\int_0^1 f\tilde{P}_k(x)\,dx$.

What do you observe about the *rate of convergence*?

Extend this code to compute the coefficients for $f(x) = e^x \sin \pi x$ and $f(x) = (1+5x^2)^{-1}$ (Runge's function). You will need to change your limits of integration (meaning your range and quadrature weights).

What do you observe about the rate of convergence in these cases?

```
kmax=1000;
M=2+kmax/2; [z,w]=zwgll(M); x=.5*(1+z); rho=.5*w;

f=x.^9;                 %% f(x) on [0,1]
Lk=legendre(x,kmax);    %% All Legendre polynomials up to N=kmax.
d=zeros(kmax+1,1);
for k=1:kmax+1;
%den = sqrt(2.0*Lk(:,k)'*(rho.*Lk(:,k))); Lk(:,k)=Lk(:,k)/den;
 den = sqrt(2/(2*k+1)); Lk(:,k)=Lk(:,k)/den;
end;

fk = 2*Lk'*(rho.*f);    %% 2X for contributions from x<0 and x>0.

k=0:2:kmax;             %% Take only even wavenumbers
fk=abs(fk(1:2:end));



%% PLOT HERE %%
loglog(k,fk,'r.-',k,5e5*(k.^-9),'g-',k,5e5*(k.^-10),'b-','linewidth',
title('Legendre Expansion Coefficients for $f=|x|^9$',...
      'interpreter','latex','fontsize',18);
xlabel('$k$','interpreter','latex','fontsize',18);
ylabel('${\hat f}_k$','interpreter','latex','fontsize',18)
axis([1 1000 1e-16 1]); axis square;
leg1=legend('${\hat f}_k$','$10^6k^{-9}$','$10^6k^{-10}$');
set(leg1,'Interpreter','latex'); set(leg1,'FontSize',16);
print -dpng 'fhat_x3.png'
!open fhat_x3.png
```



5 Nodal Bases for 1D Projection and Interpolation

In all of the discussions so far, we have considered *modal bases* for our approximations,

$$u_N(x) \;=\; \sum_k \phi_k(x)\hat{u}_k \;\approx\; \tilde{u}(x), \tag{187}$$

where the $\hat{u}_k$ are the basis coefficients. If the basis functions ϕ_k are orthogonal with respect to a given inner product then $u_N(x)$ will be the *projection* of $\tilde{u}$ onto $X^N = \text{span}\{\phi_k\}$ in the associated norm. This projective property is important because it implies that we have the *best fit* approximation in the space. No element in X^N can be closer to $\tilde{u}$ than $u = u_N$.

It is important to recognize that, when using a projection-based method, the best-fit solution is dependent only on the choice of norm and the choice of the appxoximation space, X^N, and *not* on the particular set of chosen basis functions (aside from possible round-off effects). Thus, for polynomial bases, we do not explicitly need to use orthogonal polynomials such as Legendre or Chebyshev. We can instead using *Lagrange interpolating polynomials*, which offer several advantages.

Consider the interval $\Omega = [-1, 1]$ with $N+1$ interpolation points x_j, $j = 0, \ldots, N$. The Lagrange polynomial interpolant of $\tilde{u}(x)$ is expressed in terms of Lagrange cardinal functions, $\phi_j(x) = l_j(x)$ as,

$$u(x) \;=\; \sum_{j=0}^{N} l_j(x)\tilde{u}(x_j) \;=\; \sum_{j=0}^{N} l_j(x)\tilde{u}_j, \tag{188}$$

with

$$l_j(x) \;\in\; \mathbb{P}_N(\Omega) \tag{189}$$

$$l_j(x_i) \;=\; \delta_{ij}. \tag{190}$$

Note that, with this definition, we have

$$u(x_i) \;=\; \sum_{j=0}^{N} l_j(x_i)\tilde{u}(x_j) \;=\; \sum_{j=0}^{N} \delta_{ij}\tilde{u}_j \;=\; \tilde{u}_i, \tag{191}$$

which is the required interpolation property,

$$u(x_j) = \tilde{u}(x_j), \qquad j = 0, \ldots, N. \tag{192}$$

The Lagrange cardinal functions, $l_j(x)$, have to satisfy (189)–(190). The following forms provide several equivalent representations

$$l_j(x) \;=\; \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_N}{x_j - x_N} \tag{193}$$

$$\;=\; \alpha_j^{-1}\,(x - x_0)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_N), \quad \alpha_j \;:=\; \prod_{i \neq j}(x_i - x_j) \tag{194}$$

$$\;=\; \alpha_j^{-1}\, s_j\, t_j, \qquad \begin{cases} s_j \;:=\; \displaystyle\prod_{i=0}^{j-1}(x - x_i) \;=\; (x - x_{j-1})\, s_{j-1} \\ \\ t_j \;:=\; \displaystyle\prod_{i=j+1}^{N}(x - x_i) \;=\; (x - x_{j+1})\, t_{j+1} \end{cases} \tag{195}$$

Note that, for a given value of $x = x^*$, the total cost to evaluate $l_j(x^*)$, $j = 0, \ldots, N$, is $\approx 7(N+1)$ operations when using (195). That is, the cost is *linear* in N, which is optimal. (This estimate assumes that the α_j are precomputed.) Naïve implementations, such as (193)–(194), yield $O(N^2)$ costs, which is often cited, incorrectly, as a reason to use barycentric formulas for Lagrange interpolants.

5.1 Properties of Lagrange Interpolants

We summarize a few properties of Lagrange interpolants that will be useful later on.

1. The polynomial of degree N satisfying (192) is *unique*, regardless of representation (e.g., Lagrange or Newton).

   pf: Consider Nth-order polynomials $w(x)$ and $u(x)$ satisfying (192). Clearly, $d = w - u \in \mathbb{P}_N$, and $d(x_j) = 0$, which implies that d has $N+1$ roots. The only Nth-order polynomial with $N+1$ roots is $d(x) \equiv 0$, which implies $w(x) \equiv u(x)$.

2. If $\tilde{u} \in \mathbb{P}_N$ then $u \equiv \tilde{u}$.

3. The sum of the Lagrange cardinal functions at any value of x is identically equal to 1.

   pf: Let $\tilde{u}(x) = 1 \in \mathbb{P}_N$, which implies $u(x) = \sum 1 \cdot l_j(x) \equiv 1$.

4. The sum of the derivatives, $l_j'(x)$, is 0. This argument follows along the lines of the preceding one.

5. $u_N'(x) \neq \tilde{u}'(x)$, in general, not even at the nodal points. However, if u_N is a good approximation to $\tilde{u}$, then u_N' will generally be a reasonable approximation to $\tilde{u}'$.

5.2 Computing Derivatives of Interpolants

Suppose we have a set of input pairs $(x_j, \tilde{u}_j)$ and wish to approximate $\tilde{u}'(x)$ at a set of output points, y_i, $i = 0, \ldots, M$. One can construct an approximation by using Nth-order polynomial interpolation,

$$u_i' \;:=\; \sum_{j=0}^{N} \underbrace{\left.\frac{dl_j}{dx}\right|_{y_i}}_{D_{ij}} \tilde{u}_j, \tag{196}$$

that is, $\underline{u}' = D\underline{\tilde{u}}$, where $\underline{u}' = [u'(y_o) \ldots u'(y_M)]$ and $\underline{\tilde{u}} = [\tilde{u}_0 \ldots \tilde{u}_N]$ are respective vectors of the output and input values. We refer to D as the *derivative* matrix.

Let us also define J as the *interpolation* matrix having entries $J_{ij} = l_j(y_i)$, and $\hat{D}$ as the square, $(N+1) \times (N+1)$, canonical derivative matrix with entries $\hat{D}_{ij} = l_j'(x_i)$. Suppose $\tilde{u} \in \mathbb{P}_N$. Then $\tilde{u}' \in \mathbb{P}_{N-1} \subset \mathbb{P}_N$, so

$$\underline{u}' \;\equiv\; J\hat{D}\underline{\tilde{u}}. \tag{197}$$

Here, we are first differentiating the polynomial of degree N and re-representing it on $N+1$ nodal points, even though it is a polynomial of degree $N-1$. This step is valid because $\mathbb{P}_{N-1} \subset \mathbb{P}_N$. Second, we interpolate the polynomial onto the chosen set of output points.

The advantage of this two-step approach to constructing $D = J\hat{D}$, which loses no information, is that it is far easier to construct $l'_j(x_i)$ than $l'_j(y_i)$ for general y_i. To see this, consider the derivative of (194).

$$\frac{dl_j}{dx} = \left.\begin{array}{ccccccc} \alpha_j^{-1}[& & (x-x_1) & (x-x_2) & \dots & (x-x_N) & \\ & +(x-x_0) & & (x-x_2) & \dots & (x-x_N) & \\ & +(x-x_0) & (x-x_1) & & \dots & (x-x_N) & \\ & \vdots & \vdots & & & \vdots & \\ & +(x-x_0) & (x-x_1) & (x-x_2) & \dots & (x-x_N) & \\ & +(x-x_0) & (x-x_1) & (x-x_2) & \dots & &] \end{array}\right\} \text{no } j \text{ term.} \tag{198}$$

Notice that the ith row in the expansion (198) is missing the ith monomial, $(x - x_i)$, because the derivative of that term is unity. If we now evaluate (198) at some $x_i \neq x_j$, then we have

$$\left.\frac{dl_j}{dx}\right|_{x_i} = \alpha_j^{-1}(x_i - x_0)(x_i - x_1)\dots(x_i - x_N) \qquad (i\text{th row: no } i \text{ term, no } j \text{ term}) \tag{199}$$

$$= \frac{1}{\alpha_j}\frac{\alpha_i}{(x_i - x_j)}. \tag{200}$$

(Notice that, as required, the units of (200) are $1/x$.)

To find the derivative of l_j at x_j, we note that the row-sum of the derivative matrix,

$$\hat{D}_{ij} := \left.\frac{dl_j}{dx}\right|_{x_i} \tag{201}$$

is zero, which is a consequence of the following properties. Let $f(x) \equiv 1$. Then,

$$f \in \mathbb{P}_N, \quad N \geq 0 \tag{202}$$

$$\hat{D}f \equiv \frac{df}{dx} \equiv 0. \tag{203}$$

Thus,

$$\hat{D}_{ii} = -\sum_{j \neq i} \hat{D}_{ij}. \tag{204}$$

5.3 Differentiation Example

The following code illustrates the use and convergence rate of the differentiation matrix.

```
fs='fontsize';
for N=1:25;
   [z,w]=zwgll(N);
   Dh = deriv_mat(z);
   f =exp(z);
   Df=Dh*f;
   ef=f-Df;
   err = max(abs(ef));

   plot(z,0*f,'k-',z,f,'kx',z,Df,'ro');
   title(['N,err: ' num2str([N err]) ],fs,20);
   xlabel('x',fs,20); ylabel('f',fs,20);
   pause(.3);

   eN(N)  = err;
   nN(N)  = N;
end;
semilogy(nN,eN,'ro');
title('Differentiation Error, GLL Points',fs,20)
xlabel('N',fs,20); ylabel('Error',fs,20);
```

5.4 Relationship to Projection

Here, we consider briefly the relationship between interpolation, *on the GLL points*, and L^2-projection. In the next section, we consider projection in the energy norm, where the GLL interpolants play a key role.

We begin by considering an alternative inner-product to the the standard L^2 one in which we replace integration with quadrature on the GLL points. We define this discrete inner product and associated norm as

$$(v,u)_N \;:=\; \sum_{k=0}^{N} \rho_k \, v(\xi_k)\, u(\xi_k) \tag{205}$$

$$\|u\|_N \;:=\; \sqrt{(u,u)_N}. \tag{206}$$

Note that the GLL quadrature weights are strictly positive,

$$\rho_k \;=\; \int_{-1}^{1} l_k(\xi)\, d\xi \;>\; 0, \tag{207}$$

which is *not* the case for Lagrange interpolants on $N+1$ uniformly-spaced point with $N > 9$. (Thus, (205) would not be a proper inner-product for the case of uniform-based interpolants.)

If $\tilde{v}$, $\tilde{u}$ are any pair of square-integrable functions on $\Omega = [-1,1]$ and v, $u \in X^N$ are their corresponding interpolants on the GLL points, then

$$(\tilde{v},\tilde{u})_N \;=\; \sum_{k=0}^{N} \rho_k \, \tilde{v}(\xi_k)\, \tilde{u}(\xi_k) \tag{208}$$

$$\;=\; \sum_{k=0}^{N} \rho_k \, v(\xi_k)\, u(\xi_k) \;=\; (v,u)_N. \tag{209}$$

The result (209) follows from the defining property of the interpolants that $u(\xi_i) = \tilde{u}(\xi_i)$. Therefore, the interpolant $u \in X^N$ is the *best-fit approximation* to $\tilde{u}$ in the discrete 2-norm (206).

6 Spectral Solution of One-Dimensional BVPs

Given our observations in the preceding section about the rate of convergence of spectral expansions (i.e., generalized Fourier expansions based on eigenfunctions of singular Sturm-Liouville problems), we turn now to the essential question of approximating solutions to boundary value problems (BVPs) in $\Omega \subset \mathbb{R}^d$.

6.1 Background

We begin with $d = 1$ and consider the model 1D Poisson problem

$$-\frac{\partial^2 \tilde{u}}{\partial x^2} \;=\; f(x), \;\; x \in [-1,1], \qquad\qquad \tilde{u}(-1) = \tilde{u}(1) = 0. \tag{210}$$

We take as an approximation, $u_N(x) \in X_0^N$,

$$u_N(x) \;=\; \sum_{j=0}^{N} \hat{u}_j \phi_j(x) \tag{211}$$

where $X^N = \mathbb{P}_N$ is the space of polynomials of degree $\leq N$ and $X_0^N = \text{span}\{\phi_0, \ldots, \phi_N\}$ is the space of functions that in X^N that vanish at $x = \pm 1$. Presently, we define the basis functions to be the Legendre *bubble functions*

$$\boxed{\phi_j(x) \;=\; P_{j+2}(x) \;-\; P_j(x)} \tag{212}$$

where P_j is the Legendre polynomial satisfying $P_j(\pm 1) = (\pm 1)^j$.

For any $u(x) \in X_0^N$, we define the residual

$$r(x;u) \;:=\; f \;-\; \left(-\frac{\partial^2 u}{\partial x^2}\right) \;=\; f + u''. \tag{213}$$

As in our earlier example, we would like this residual to be orthogonal to a finite-dimensional subspace. An important question is, *which subspace?* For second-order boundary value problems a natural choice is to make the residual orthogonal to the approximation space, which leads to the Galerkin formulation. We will see that this approach is equivalent to minimizing the error in the energy norm, which we define shortly. For now, we proceed with the orthogonal projection.

For $i = 0, \ldots, N$, we require

$$-\int_{-1}^{1} \phi_i u_N'' \, dx \;=\; \int_{-1}^{1} \phi_i f \, dx. \tag{214}$$

Inserting the expansion (211) for u_N, we have

$$-\sum_{j=0}^{N} \left(\int_{-1}^{1} \phi_i \phi_j'' \, dx\right) \hat{u}_j \;=\; \int_{-1}^{1} \phi_i f \, dx, \tag{215}$$

or $A\underline{u} = \underline{b}$, with $\underline{u} = [u_0 \, u_1 \ldots u_N]^T$ and $\underline{b}$ the vector having entries

$$b_i \;=\; \int_{-1}^{1} \phi_i f \, dx. \tag{216}$$

The *stiffness matrix*, A has entries

$$a_{ij} \;=\; -\int_{-1}^{1} \phi_i \phi_j'' \, dx. \tag{217}$$

For a multitude of reasons, not the least of which is symmetry, it is desirable to integrate this expression by parts,

$$a_{ij} \;=\; \int_{-1}^{1} \phi_i' \phi_j' \, dx \;-\; \phi_i \phi_j' \big|_{-1}^{1} \tag{218}$$

$$\;=\; \int_{-1}^{1} \phi_i' \phi_j' \, dx. \tag{219}$$

A is clearly symmetric. Moreover, for any $u \in X_0^N$ we have the energy inner-product

$$a(u,u) \;:=\; \int_{-1}^{1} u' u' \, dx \;=\; \sum_{j=0}^{N} \sum_{i=0}^{N} u_i a_{ij} u_j \;=\; \underline{u}^T A \underline{u} \;>\; 0 \;\; \forall \underline{u} \neq 0. \tag{220}$$

Notice that $u' \equiv 0$ only for the constant function but the only constant function in X_0^N is $u = 0$. Thus, the integral in (220) is positive for any nontrivial $u \in X_0^N$ and we therefore have $\underline{u}^T A \underline{u} > 0$ for all nontrivial $\underline{u}$, which implies that A is positive definite. For our particular choice of $\phi_j = P_{j+2} - P_j$, use of a Legendre polynomial identity yields

$$\phi_j' \;=\; P_{j+2}' \;-\; P_j' \;=\; (2j+3) P_{j+1}, \tag{221}$$

which implies that A is diagonal, with entries

$$a_{ij} \;=\; \delta_{ij} \, (2i+3)^2 \int_{-1}^{1} P_{j+1} P_{j+1} \, dx \;=\; \delta_{ij} \, 2(2i+3). \tag{222}$$

(See the wonderful and inexpensive book by Abramowitz and Stegun for a full list of properties of orthogonal polynomials.)

Error estimation is particularly easy given that A is diagonal, The right-hand side quantities are

$$b_i \;:=\; \int_{-1}^{1} \phi_i \, f \, dx \;=\; \int_{-1}^{1} P_{i+2} \, f \, dx \;-\; \int_{-1}^{1} P_i \, f \, dx. \tag{223}$$

As was the case with Fourier methods, the rate of convergence is controlled by the regularity of $f(x)$, save that we do not need f or u) to be periodic. Clearly, these coefficients go to zero rapidly as $i \longrightarrow \infty$, if $f(x)$ is infinitely differentiable. In this case, the leading-order error scales with the first nontrivial term in the remainder of the truncated series,

$$|e_N| \;\sim\; |\hat{u}_{N+1}| + |\hat{u}_{N+2}| \;=\; \frac{|b_{N+1}|}{a_{N+1,N+1}} + \frac{|b_{N+2}|}{a_{N+2,N+2}} \;\sim\; \frac{|\hat{f}_{N+1}|}{(2N+5)^2} + \frac{|\hat{f}_{N+2}|}{(2N+7)^2}. \tag{224}$$

Here, we take two terms in the expansion for u for those occasions when even or odd symmetry (matching the symmetries of P_k) would result in one of the pair being zero. Recall that the generalized Fourier coefficients of f,

$$\hat{f}_k \;=\; \frac{\int_{-1}^{1} P_k f(x) \, dx}{\int_{-1}^{1} P_k P_k \, dx} \;=\; \frac{2k+3}{2} \int_{-1}^{1} P_k f(x) \, dx, \tag{225}$$

will go to zero exponentially fast if f is smooth. As $N \longrightarrow \infty$, the error in this case scales as

$$|e_N| \;\sim\; |\hat{u}_{N+1}| + |\hat{u}_{N+2}| \;\leq\; |\hat{u}_N| + |\hat{u}_{N-1}|, \tag{226}$$

so we can conveniently take $|\hat{u}_N| + |\hat{u}_{N-1}|$ as a conservative error estimate.

Exercise: Plot $\|e_N\|_\infty$, $\|e_N\|_2$, and the error estimate of (226) vs. N for the case $\tilde{u} = 2 - \sqrt{3 + x^2}$.

6.2 Variational Problem Statement

In the preceding development, we elected to make the residual orthogonal to the trial space, X_0^N. More generally, we could have chosen another $(N+1)$ dimensional approximation space, Y^N. The choice $Y^N = X_0^N$ led to a symmetric positive definite system, which is certainly beneficial. Here, we explore other reasons for making this choice and, in doing so, establish an important principle that will significantly widen our development options.

In (220) we introduced the a inner-product and associated semi-norm

$$\|u\|_a \;=\; \sqrt{a(u,u)} \;:=\; \left[\int_\Omega \frac{du}{dx}\frac{du}{dx}\,dx\right]^{\frac{1}{2}}. \tag{227}$$

We further define the $\mathcal{H}^1$ norm

$$\|u\|_1 \;=\; \left[\int_\Omega \frac{du}{dx}\frac{du}{dx} \;+\; u^2\,dx\right]^{\frac{1}{2}}, \tag{228}$$

and associated space $\mathcal{H}^1 = \{v \,|\, v \in L^2,\; v' \in L^2\}$. (Note that the $\mathcal{H}^1$ definition is only sensible in the nondimensional domain $\hat{\Omega}$, but we will use it as a short-hand throughout the course.) Additionaly, we introduce the space $\mathcal{H}_0^1 := \{v \,|\, v \in \mathcal{H}^1,\; v = 0 \text{ on } \partial\Omega_D\}$, where $\partial\Omega_D$ is the subset of the domain boundary where Dirichlet boundary conditions are to be enforced. For any $u \in \mathcal{H}_0^1$, (227) is also a proper norm because it will be zero only for the constant function and the only admissible constant function in $\mathcal{H}_0^1$ is $u = 0$. In these circumstances, we will refer to (227) as the a-norm.

In our example the solution (211) came from a linear combination of trial functions, $\phi_k \in \mathcal{H}_0^1$, and the *trial space* X_0^N is the span of these basis functions, X_0^N:=span$\{\phi_k\}$, $k = 0, \ldots, N$. Note that it is not necessary for the ϕ_ks to individually satisfy the homogeneous boundary conditions—only the linear combination (211) must do so. For convenience, however, it is invariably the case that we enforce the homogeneous Dirichlet conditions directly on the basis functions. We now demonstrate that choosing Y^N (the *test space*) to be equal to X_0^N leads to a minimization problem. That is, for this particular choice, $u \in X_0^N$ will be the *Galerkin projection* of $\tilde{u}$ onto X_0^N.

We start by recasting the 1D model problem as a variational statement, *Find* $u_N \in X_0^N$ *that minimizes*

$$\Phi(u) \;:=\; \|\tilde{u} - u\|_a^2 \;=\; a(\tilde{u} - u, \tilde{u} - u), \tag{229}$$

for all $u \in X_0^N \subset \mathcal{H}_0^1$. We proceed exactly in the same way as with the L^2-projection (119). Inserting (211) into (229) and expanding all terms yields

$$\Phi(u) \;:=\; \left(\sum_{k=0}^{N} \phi_k' \hat{u}_k - \tilde{u}', \sum_{j=0}^{N} \phi_j' \hat{u}_j - \tilde{u}'\right) \tag{230}$$

$$\;=\; \sum_{k=0}^{N}\sum_{j=0}^{N} \hat{u}_k a_{kj} \hat{u}_j \;-\; \sum_{j=0}^{N} b_j \hat{u}_j \;-\; \sum_{k=0}^{N} b_k \hat{u}_k \;+\; a(\tilde{u}, \tilde{u}). \tag{231}$$

Here we have the *stiffness matrix*, A, with entries

$$a_{kj} \;:=\; (\phi_k', \phi_j') \tag{232}$$

and the data entries

$$\begin{aligned} b_i \;&:=\; (\phi_i', \tilde{u}') && (233)\\ &=\; \int_\Omega \frac{d\phi_i}{dx}\frac{d\tilde{u}}{dx}\,dx && (234)\\ &=\; -\int_\Omega \phi_i \frac{d^2\tilde{u}}{dx^2}\,dx \;-\; \underbrace{\phi_i \frac{d\tilde{u}}{dx}\Big|_{-1}^{1}}_{=0} && (235)\\ &=\; \int_\Omega \phi_i f\,dx. && (236) \end{aligned}$$

At the global minimum, we require stationarity of Φ,

$$\begin{aligned} \frac{\partial\Phi}{\partial\hat{u}_i} \;&=\; \frac{\partial}{\partial\hat{u}_i}\left[\sum_{k=0}^{N}\sum_{j=0}^{N}\hat{u}_k a_{kj}\hat{u}_j \;-\; \sum_{j=0}^{N} b_j\hat{u}_j \;-\; \sum_{k=0}^{N} b_k\hat{u}_k \;+\; (\tilde{u},\tilde{u})\right] && (237)\\ &=\; 2\sum_{j=0}^{N} a_{ij}\hat{u}_j \;-\; 2b_i \;=\; 0, \qquad\qquad i = 0,\ldots,N. && (238) \end{aligned}$$

In matrix form,

$$A\underline{u} \;=\; \underline{b}, \tag{239}$$

where $\underline{u} = [\hat{u}_0,\, \hat{u}_1, \ldots, \hat{u}_N]^T$ and $\underline{b} = [b_0,\, b_1, \ldots, b_N]^T$.

As noted earlier, basis functions $\phi_j = P_{j+2} - P_j$ are orthogonal with respect to the a inner-product such that a is diagonal with entries

$$a_{ij} \;=\; a(\phi_i,\phi_j)\delta_{ij} \;=\; 2(2j+3)\delta_{ij}, \tag{240}$$

and the projection of $\tilde{u}$ onto X_0^N has the simple form

$$u_N(x) \;=\; \sum_{j=0}^{N}\frac{a(\phi_j,\tilde{u})}{a(\phi_j,\phi_j)}\phi_j(x) = \sum_{j=0}^{N}\phi_j(x)\hat{u}_j. \tag{241}$$

We see that $u_N \in X_0^N$ and inherits units of $\tilde{u}$, as must be the case.

6.3 Principal Result

The purpose of the preceding argument was to establish that we have a broader form of stating the discrization problem. Specifically, our spectral method has now been written as a variational (or minimization) problem,

$$u_N \;=\; \underset{u\in X_0^N}{\operatorname{argmin}}\, a(\tilde{u}-u_N, \tilde{u}-u_N), \tag{242}$$

which is equivalent to the weighted residual statement
Find $u_N \in X_0^N$ *such that for all* $v \in X_0^N$,

$$a(v, u_N) \;=\; a(v, \tilde{u}) \;=\; (v, f). \tag{243}$$

Either approach is valid, but we typically use the latter as it provides a more direct route to the matrix formulation of the problem and it is also extensible to nonsymmetric systems.

The liberating feature of these statements, however, is that they simply state the *space* that is to be used for the trial and test functions—not the *form*. From (242), we have that u_N is the *best fit* approximation in X_0^N with respect to the a-norm (also known as the *energy norm*). Since Φ is a quadratic form, it has a unique minimizer in X_0^N and the representation of u_N is immaterial provided that the chosen basis functions span X_0^N. (They may even be linearly dependent–the solution will still be the unique minimizer–but this dependency would lead to a singular A matrix.)

We can thus consider the following numerical discretrization. Let

$$u(x) \;=\; \sum_{j=1}^{M-1} u_j l_j(x) \tag{244}$$

where $l_j(x) \in \mathbb{P}_M(\Omega)$ are the Lagrange cardinal basis functions and $l_j(\xi_i) = \delta_{ij}$. Here we choose the ξ_is to be the GLL points for $i = 0, \ldots, M$. Define X^N=span$\{l_k\}$, $k = 0, \ldots, M$ and X_0^N=span$\{l_k\}$, $k = 1, \ldots, M-1$. Then (243) (equivalently, (242)), leads to $A\underline{u} = \underline{b}$ with $\underline{u} = [u_1\, u_2 \ldots u_{M-1}]^T$, $\underline{b} = [b_1\, b_2 \ldots b_{M-1}]^T$, and

$$a_{ij} \;=\; \int_{-1}^{1} \frac{dl_i}{dx}\frac{dl_j}{dx}\, dx \tag{245}$$

$$b_i \;=\; \int_{-1}^{1} l_i\, f\, dx. \tag{246}$$

In the absence of round-off error, the solution (244) produced through this procedure will be *identical* to u_N from (211) provided we take $M = N+2$, as the approximation spaces are coincident in this case. Thus, if $f(x)$ is smooth, we can expect expontial convergence,

$$|\tilde{u}(x) \,-\, u(x)| \;=\; O(e^{-\sigma N}) \tag{247}$$

for positive constant σ. If f is not smooth, the convergence rate is only algebraic (i.e., $O(N^s)$ for some fixed value of s).

These results are summarized in a (a-orthogonal) projection statement in several texts. For example, equation (5.4.30) in CHQZ2 gives

$$\boxed{\|\tilde{u} - u_M\|_{H^k(\Omega)} \;\le\; CM^{k-m}|\tilde{u}|_{H^{m;M}(\Omega)},} \tag{248}$$

for all $\tilde{u} \in \mathcal{H}_0^m$ with $m \ge 1$ and $k = 0, 1$. (Note that the H^0 norm is just the L^2 norm.) In the 1D case, we have

$$\|u\|_{H^k(\Omega)} \;:=\; \left[\sum_{j=0}^{k} \int |u^{(j)}|^2\, dx\right]^{\frac{1}{2}}, \tag{249}$$

$$|u|_{H^{m;M}(\Omega)} \;:=\; \left[\sum_{j=k'}^{m} \int |u^{(j)}|^2\, dx\right]^{\frac{1}{2}}, \tag{250}$$

where $k' = \min(m, M+1)$. The statement (248) implies that if $\tilde{u}$ is m-times differentiable then the convergence in the L^2- (H^0-) norm will scale as $O(M^{-m})$ and the convergence in the H^1-norm will scale as $O(M^{1-m})$. If $\tilde{u}$ (and, hence, f) is infinitely differentiable, the error will go to zero faster than any fixed power of $1/M$. That is, we will have exponential convergence.

A somewhat cleaner statement of this result for the case $\Omega = [0,1]$ with a multidomain spectral (i.e., spectral- or p-type finite element) method is given by (12.59) in Quarteroni,Sacco & Saleri [QSS]. Let $h = 1/E$, where E is the number of uniformly-sized elements on Ω. *Assume that* $\tilde{u} \in \mathcal{H}^s(\Omega)$ *for some* $s \geq 2$. *Then*

$$\boxed{\|\tilde{u} - u_N\|_{H_0^1(\Omega)} \leq Ch^l |\tilde{u}|_{H^{l+1}(\Omega)},} \tag{251}$$

where $l = \min(N, s-1)$. *Under the same assumptions, one also has*

$$\boxed{\|\tilde{u} - u_N\|_{L^2(\Omega)} \leq Ch^{l+1} |\tilde{u}|_{H^{l+1}(\Omega)}.} \tag{252}$$

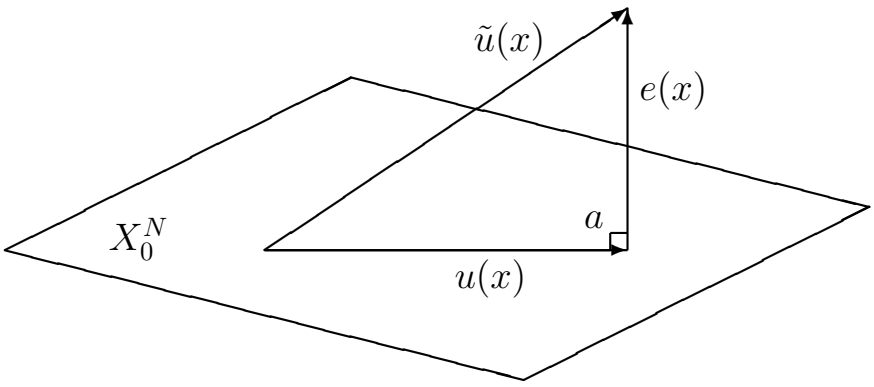
The [QSS] result shows that if $\tilde{u}$ is infinitely smooth one has convergence that is exponential with N. Otherwise, convergence is algebraic at a rate $O(h^{s-1})$ or $O(h^s)$, depending on the norm.

Coda

We have introduced in this section the basic idea using projection onto a finite-dimensional space $X_0^N \subset X^N$ to discretize a boundary-value problem and to numerically generate an approximation to the same. N is generally tied to the order of the polynomial (or trigonometric polynomial), which will be approximately the same as the dimension of the space. The precise dimension, however, will depend on the number and type of boundary conditions and also d, the number of dimensions for the spatial variable $\mathbf{x}$.

In the next section, we look at boundary-condition extensions in the 1D case. We follow this with extensions to multiple space dimensions and then to other differential equations.

6.4 Galerkin Projection Illustration



6.5 Additional Boundary Conditions in 1D

We extend our 1D model problem of the previous section by considering inhomogeneous Neumann conditions at $x = 1$,

$$-\frac{\partial^2 \tilde{u}}{\partial x^2} \;=\; f(x), \;\; x \in [-1,1], \qquad\qquad \tilde{u}(-1) = 0, \;\; \tilde{u}'(1) = g. \tag{253}$$

We define $\mathcal{H}_0^1 = \{\, v \,|\, v \in \mathcal{H}^1,\, v(-1) = 0\}$, which enforces the homogeneous Dirichlet condition only at the left endpoint. Functions $u \in X_0^N \subset \mathcal{H}_0^1$ are allowed to fluctuate at $x = 1$. We use as our basis for X_0^N the Lagrange polynomials on the $N+1$ GLL points and seek as solution $u \in X_0^N$ given by

$$u(x) \;=\; \sum_{j=1}^{N} u_j l_j(x), \tag{254}$$

where, by the presence of $u_N l_N(x)$, $u(x)$ is now allowed to vary at $x = 1$. The variational projection statement reads, *Find $u \in X_0^N$ such that for all $v \in X_0^N$,*

$$a(v,u) \;=\; a(v,\tilde{u}), \tag{255}$$

which is *exactly* the same as for the full Dirichlet case, save that we have enlarged the search space to accommodate the variation in $u(x)$ at $x = 1$. We reiterate that (255) is equivalent to the minimization problem,

$$u \;=\; \underset{v \in X_0^N}{\operatorname{argmin}} \, \|\tilde{u} - v\|_a^2, \tag{256}$$

which says that u is the element in X_0^N that is closest to $\tilde{u}$ (i.e., the *best fit*, in the energy norm). So, if $\tilde{u}$ has some particular boundary properties, as in (253), we can expect $u(x)$ to inherit these properties as the solution converges, $u(x) \longrightarrow \tilde{u}(x)$.

Let's examine how the inhomogeneous Neumann condition manifests itself in the context of our Galerkin formulation. Starting with (255), we integrate the analytical solution by parts, and then substitute the defining relation for $-u''$ given by (253),

$$\begin{aligned} a(v,u) \;&=\; a(v,\tilde{u}) && (257)\\ &=\; \int_\Omega v'\tilde{u}'\,dx && (258)\\ &=\; -\int_\Omega v\tilde{u}''\,dx \,+\, v\tilde{u}'\big|_{-1}^{1} && (259)\\ &=\; \int_\Omega vf\,dx \,+\, v(1)g. && (260) \end{aligned}$$

Notice that $v(-1) = 0$ for all $v \in X_0^N$, so the boundary term in (259) equates to $v(1)g$.

Inserting (254) into (260) and taking $v(x) = l_i(x)$ for $i = 1, \ldots, N$ yields N equations in N unknowns,

$$\sum_{j=1}^{N} a_{ij} u_j \;=\; b_i, \tag{261}$$

with

$$a_{ij} = a(l_i, l_j) \tag{262}$$

$$b_i = \begin{cases} (l_i, f), & i = 1, \ldots, N-1 \\ (l_N, f) + g, & i = N \end{cases} . \tag{263}$$

The last equation follows from (260) because only l_N has a nontrival value at $x = 1$.

For Robin boundary conditions of the form $u' = \beta(u_\infty - u)$, one proceeds in much the same way as above save that we replace $\tilde{u}'$ in (259) with $\beta(u_\infty - u)$, which gives rise to a modification of both the right-hand-side of (261) and to the last diagonal term of A. Specifically, we end up with $a_{NN} = a(l_N, l_N) + \beta$ and $b_N = (l_N, f) + \beta u_\infty$. Notice that we must have $\beta \geq 0$ to ensure that A remains positive definite.

6.6 Restriction Operators for Dirichlet/Neumann Conditions

To simplify our notation (and coding!) for a variety of boundary conditions, we introduce some important notation that allows us to easily switch from Dirichlet to Neumann or Robin conditions and to readily impose *inhomogeneous* Dirichlet conditions.

To begin, we define a function $\bar{u} \in X^N$ as

$$\bar{u}(x) = \sum_{j=0}^{N} u_j l_j(x) \tag{264}$$

and the restriction of this function to X_0^N as,

$$u(x) = \sum_{j=1}^{N} u_j l_j(x), \tag{265}$$

for the particular case where Dirichlet conditions are applied only on the left (i.e., as prescribed in (253)). If we have Dirichlet conditions on both ends, we will define

$$u(x) = \sum_{j=1}^{N-1} u_j l_j(x). \tag{266}$$

We see already that this notation is cumbersome because the index ranges and the number of degrees of freedom depend on the precise nature of the boundary conditions whereas the mathematical statement (255) does not. To streamline the notation, we introduce a *restriction matrix*, R, which extracts only the interior (or active) degrees of freedom from a coefficient vector. In essence, we wish to generate a rectangular matrix R such that

$$\underline{u} = R\underline{\bar{u}}, \tag{267}$$

where $\underline{\bar{u}} = [u_0\, u_1\, \ldots\, u_N]^T$ and $\underline{u} = [u_1\, u_2\, \ldots\, u_N]^T$ (say, for the case of a single Dirichlet condition on the left). In this case,

$$R = [\,\underline{0}\;\; I\,], \tag{268}$$

where I is $N \times N$ and R is $N \times (N+1)$. Note that

$$R^T \;=\; \left[\begin{array}{c} \underline{0}^T \\ I \end{array}\right] \tag{269}$$

applied to a vector $\underline{u}$ *prolongates* (i.e., extends) by zero any set of interior basis coefficients $\underline{u}$ to include zeros on the boundary. We can denote the operation as $\bar{\underline{u}} = R^T \underline{u}$, which is illustrated by the following,

$$\bar{\underline{u}} \;=\; \left(\begin{array}{c} 0 \\ \bar{u}_1 \\ \vdots \\ \bar{u}_N \end{array}\right) = \left[\begin{array}{cc} & \underline{0}^T \\ & \\ I_N & \\ & \end{array}\right] \left(\begin{array}{c} u_1 \\ \vdots \\ u_N \end{array}\right), \tag{270}$$

with $\bar{u}_j = u_j$, $j = 1, \ldots, N$. Note that, as always, the output of a matrix-vector product is a linear combination of the columns of the matrix,

$$\underline{v} \;=\; A\underline{u} \;\;\longrightarrow\;\; \underline{v} = \sum_j \underline{a}_j \, u_j, \tag{271}$$

where $\underline{a}_j$ is defined to the jth column of A. Thus, from (270) we can only expect that $\bar{u}_0 \equiv 0$ given that the elements of the first row of R^T are all zero.

Masking. As an example, suppose we have a vector $\underline{u}_b$ that includes a nontrivial value u_0 as its first entry. If we apply the restriction operator, R, followed by the prolongation operator, R^T, the net result is a vector that is as long as the original but now has 0 in place of u_0. Such an operation is referred to as a *mask*. It zeros out boundary values in a vector of nodal basis coefficients. The operation is often written as $\underline{u}_0 = R^T R \underline{u}_b = \mathcal{M}\underline{u}_b$. The mask matrix, $\mathcal{M}$, is essentially the identity save for zeros wherever Dirichlet conditions are applied.

We make the following observation. Suppose that $\bar{\underline{u}} = R^T\underline{u}$ and $\bar{\underline{v}} = R^T\underline{v}$, and that $\bar{u}(x)$ is given by (264) with a similar formula for $\bar{v}(x)$. Then, despite having the representation of functions in X^N (i.e., *all* $N+1$ of their coefficients are defined), $\bar{u}$ and $\bar{v}$ are in X_0^N and $a(\bar{v}, \bar{u}) \equiv a(v, u)$.

6.7 Inhomogeneous Dirichlet Conditions

The restriction matrix notation is particularly useful in the context of inhomogeneous Dirichlet conditions. Consider the d-dimensional Poisson example,

$$\begin{aligned} -\nabla^2 \tilde{u} \;&=\; f(\mathbf{x}), \qquad \tilde{u}(\mathbf{x}) = \tilde{u}_b(\mathbf{x}) \text{ on } \partial\Omega_D, \\ &\qquad\qquad \nabla\tilde{u}(\mathbf{x}) \cdot \hat{\mathbf{n}} = 0 \text{ on } \partial\Omega_N, \end{aligned} \tag{272}$$

where $\hat{\mathbf{n}}$ is the outward pointing unit normal vector on $\partial\Omega$.

As before, let $X^N = \text{span}\{\phi_j(\mathbf{x})\}$ be the space of *all* of our chosen basis functions, even those that do not vanish on $\partial\Omega_D$, and let

$$X_0^N = \{v \in X^N \mid v = 0 \text{ on } \partial\Omega_D\} \tag{273}$$

be our usual restricted approximation space comprising only those basis functions that vanish on $\partial\Omega_D$. We now introduce a new space,

$$X_b^N \;:=\; \{\bar{v} \in X^N \mid \bar{v} = \tilde{u}_b \text{ on } \partial\Omega_D\}. \tag{274}$$

This space is not closed under addition since, if $\bar{v} \in X_b^N$ and $\bar{u} \in X_b^N$, we would have $(\bar{v} + \bar{u})|_{\partial\Omega_D} = 2\tilde{u}_b$. Nonetheless, we can still seek a solution $\bar{u} \in X_b^N$ that is as close as possible to $\tilde{u}$. The minimization statement reads, *Find* $\bar{u} \in X_b^N$ *such that*

$$\|\tilde{u} - \bar{u}\|_a \;\leq\; \|\tilde{u} - \bar{w}\|_a \;\forall\; \bar{w} \;\in\; X_b^N. \tag{275}$$

Here, we have suggestively used the notation $\bar{u}$ to imply that our discrete solution includes the values of $u(\mathbf{x})$ that are nonzero on $\partial\Omega_D$.

For simplicity, it is desirable to recast the minimization statement (275) into an equivalent orthogonalization statement. Let v be defined as follows,

$$\bar{w} \;:=\; \bar{u} \,-\, \epsilon v \;\longrightarrow\; \tilde{u} - \bar{w} \,=\, \tilde{u} - \bar{u} + \epsilon v \,=\, e_N + \epsilon v, \tag{276}$$

where ϵ is a nontrivial scalar. Notice, crucially, that $v(\mathbf{x}) = \epsilon^{-1}(\bar{u} \,-\, \bar{w}) \;\in\; X_0^N$.

Returning to our minimization statement (275), we have

$$\|\tilde{u} - \bar{u}\|_a^2 \;=\; \|e_N\|_a^2 \;\leq\; \|e_N + \epsilon v\|_a^2 \;=\; \|e_N\|_a^2 \;+\; 2(v, e_N)_a + \epsilon^2 \|v\|_a^2 \tag{277}$$

The inequality must hold for all $v \;\in\; X_0^N$, which implies that, while $\bar{u} \in X_b^N$ (the trial space), our *test space* is X_0^N. So, minimization in X_b^N requires that the error be a-orthogonal to X_0^N

$$(v, e_N)_a \;=\; 0 \;\Longleftrightarrow\; (v, \bar{u})_a \,=\, (v, \tilde{u})_a \;\forall\; v \;\in\; X_0^N. \tag{278}$$

The search for $\bar{u}$ is greatly simplified by decomposing it into a homogeneous part, $\bar{u}_0 \in X_0^N$ and a known inhomogeneous part, $\bar{u}_b \in X_b^N$. When using nodal bases for X_N, we generally take $\bar{u}_b$ to be the interpolant that satisfies the inhomogenous Dirichlet conditions,

$$\bar{u}_b(\mathbf{x})|_\Omega \;:=\; \sum_{\mathbf{x}_j \in \partial\Omega_D} \tilde{u}(\mathbf{x}_j)\phi_j(\mathbf{x}). \tag{279}$$

(Note that (279) introduces a potential error on $\partial\Omega_D$ if the boundary data is not well represented in X_b^N, but this error is typically small.) Inserting the expansion $\bar{u} = \bar{u}_0 + \bar{u}_b$ into the Galerkin form leads to, *For all* $v \in X_0^N$,

$$(v, \bar{u}_0 + \bar{u}_b)_a \;=\; (v, \tilde{u})_a \,=\, (v, f) \tag{280}$$

$$(v, \bar{u}_0)_a \;=\; (v, f) \,-\, (v, \bar{u}_b)_a. \tag{281}$$

At this point, we can consider evaluation of (281) for the 1D case where $\phi_j(\mathbf{x}) = l_j(x)$, $j = 0, \ldots, N$. Let's suppose the problem in strong form is

$$-\tilde{u}'' \;=\; f, \quad \tilde{u}(-1) = \tilde{u}_0, \quad \tilde{u}'(1) = 0. \tag{282}$$

Let

$$\bar{v}(x) \;=\; \sum_{j=0}^{N} l_j(x)\, \bar{v}_j \tag{283}$$

$$\bar{u}_0(x) \;=\; \sum_{j=0}^{N} l_j(x)\, \bar{u}_{0,j} \tag{284}$$

$$u_0(x) \;=\; \sum_{j=1}^{N} l_j(x)\, u_{0,j} \tag{285}$$

$$\bar{u}_b(x) \;=\; \sum_{j=0}^{N} l_j(x)\, \bar{u}_{b,j}, \tag{286}$$

with $\bar{v}_0 = 0$, $\bar{u}_{0,0} = 0$ and $\bar{u}_{b,0} = \tilde{u}_0$. Note that we assume $u_0(x) \equiv \bar{u}_0(x)$, with the only distinction being whether we include the trivial boundary data representation or not. Define $\underline{\bar{v}} = R^T \underline{v}$ and $\underline{\bar{u}}_0 = R^T \underline{u}_0$, with R given by (268) Define $\bar{A}$ as the matrix with entries

$$\bar{A}_{ij} \;=\; a(\phi_i, \phi_j) \;=\; \int_{-1}^{1} \frac{dl_i}{dx}\frac{dl_j}{dx}\,dx, \quad i,j \;\in\; \{0,\ldots,N\}^2 \tag{287}$$

Then,

$$a(v,u_0) \;=\; a(\bar{v},\bar{u}_0) \;=\; \underline{\bar{v}}^T \bar{A}\underline{\bar{u}}_0 \;=\; (R^T\underline{v})^T\bar{A}(R^T\underline{u}_0) \;=\; \underline{v}^T \underbrace{R\bar{A}R^T}_{A}\underline{u}_0 \tag{288}$$

We typically refer to $\bar{A}$ as the *Neumann operator* since it corresponds to the discrete semi-positive-definite Laplacian operator that would arise if one had all Neumann conditions (i.e., $R = I$). A, on the other hand is SPD and invertible.

On the right hand side of (281) we proceed in a similar fashion,

$$(v,\bar{u}_b)_a \;=\; a(\bar{v},\bar{u}_b) \;=\; \underline{\bar{v}}^T\bar{A}\underline{\bar{u}}_b \;=\; (R^T\underline{v})^T\bar{A}\underline{\bar{u}}_b \;=\; \underline{v}^T R\bar{A}\underline{\bar{u}}_b. \tag{289}$$

Here, there is no application of R^T to $\underline{\bar{u}}_b$ as it is already defined on the boundary. The *restriction*, however, is applied, as that term comes from $v \in X_0^N$—we test only against functions in X_0^N.

Combining all of these ideas, we have the following,

$$\underline{\bar{u}} \;=\; R^T\underline{u}_0 \;+\; \underline{\bar{u}}_b \tag{290}$$

$$\underline{u}_0 \;=\; A^{-1}R\left[\bar{B}\underline{\bar{f}} \;-\; \bar{A}\underline{\bar{u}}_b\right]. \tag{291}$$

It is interesting to observe what happens for the case $f \equiv 0$, which implies that $\tilde{u}$ is the solution to Laplace's equation with nontrivial boundary data. In this case, we have

$$\underline{\bar{u}} \;=\; \underline{\bar{u}}_b \;+\; R^T\underline{u}_0 \tag{292}$$

$$\;=\; \underline{\bar{u}}_b \;-\; R^T A^{-1}R\bar{A}\underline{\bar{u}}_b \tag{293}$$

$$\;=\; \left(I \;-\; R^T(R\bar{A}R^T)^{-1}R\bar{A}\right)\underline{\bar{u}}_b, \tag{294}$$

which is the (energy-miminizing) projection of $\underline{\bar{u}}_b$ onto X^N. The corresponding continuous solution, $\bar{u}(\mathbf{x})$ is sometimes referred to as the *harmonic extension* of the prescribed boundary data, $\tilde{u}_b(\mathbf{x})$.

6.8 Periodic Boundary Conditions

One can view the prolongation matrix of the preceding section as a map from the set of uknowns, $\underline{u}$ to a continuous function $\bar{u}(x) \in X_0^N$, where $\bar{u}$ formally has $N+1$ basis coefficients,

$$\bar{u}(x) \;=\; \sum_{j=0}^{N} l_j(x)\bar{u}_j. \tag{295}$$

Because $\underline{\bar{u}} = R^T\underline{u}$, we know that $u_j = 0$ wherever homogeneous Dirichlet conditions are applied.

In a similar fashion, we can consider the finite-dimensional space of *periodic* functions,

$$X_p^N \;=\; \{v \in X^N \,|\, v(-1) = v(1)\,\}. \tag{296}$$

With our Lagrangian basis functions, $X^N = \text{span}\{l_0, \ldots, l_N\}$, (296) implies that $u_0 = u_N$ for any $u(x) \in X_p^N$. As we did with the Dirichlet case, we could use R^T as a discrete map from the unknowns (the u_js) into the desired approximation space. For future notational reasons, however, we choose to represent this map with Q such that $\underline{\bar{u}} = Q\underline{u}$. Given the set of unknown basis coefficients, $\underline{u} = [u_1\ u_2\ \ldots\ u_N]^T$, we generate the full set $\underline{\bar{u}}$ as follows.

$$\underline{\bar{u}} \;=\; Q\underline{u} \;=\; \begin{bmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & \cdots & \\ & & & 1 \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} \;=\; \begin{pmatrix} u_N \\ u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}. \tag{297}$$

Proceeding as before, our projection statement is, *Find* $\bar{u} \in X_p^N$ *such that, for all* $\bar{v} \in X_p^N$,

$$a(v, \bar{u}) \;=\; a(v, \tilde{u}) \;=\; (v, f) \tag{298}$$

The fully discrete form of this statement is, *For all* $\underline{v} \in \mathbb{R}^N$,

$$a(v, \bar{u}) \;=\; (Q\underline{v})^T \bar{A} (Q\underline{u}) \tag{299}$$

$$\;=\; \underline{v}^T \underbrace{Q^T \bar{A} Q}_{A} \underline{u} \;=\; \underline{v}^T Q^T \bar{B} \underline{\bar{f}}. \tag{300}$$

Here, we have not insisted that $f(x)$ be periodic, so we do not force $\underline{\bar{f}} = Q\underline{f}$. We refer to $A = Q^T \bar{A} Q$ as the *assembled* system matrix. It arises from restricting our attention to test and trial functions that are periodic. Q plays a very similar role to R^T of the preceding section in that it injects the function contolled by the coefficients in $\underline{u}$ into the proper search space. However, it also changes the matrix topology, in the same way that matrix assembly does in the multidomain (i.e., multielement) case, as we shall see later. It's a bit ambiguous here whether this Q should be viewed as a boundary operator (and hence labeled "R^T") or as an assembly operator. Because of the implications related to the matrix topology we have opted for the latter interpretation.

6.9 Robin Boundary Conditions

Robin boundary conditions have the form

$$\left[\alpha \frac{du}{dx} + \beta u\right]_{x=1} \;=\; \gamma \tag{301}$$

In heat transfer, these boundary conditions are often associated with Newton's Law of Cooling, which says that the rate of heat loss at the boundary of a solid is proportional to $T_w - T_\infty$, where T_w is the surface temperature and T_∞ is the ambient temperature (in air, say). That is,

$$-k \nabla T \cdot \hat{\mathbf{n}} \;=\; h_c \left(T_w - T_\infty\right), \tag{302}$$

where h_c is a positve heat transfer coefficient (with appropriate units). Notice that if $T_w > T_\infty$ then the object is losing heat, meaning that its interior is warmer than its surface temperature. In this case $\nabla T \cdot \hat{\mathbf{n}} < 0$, where $\hat{\mathbf{n}}$ is the outward pointing unit normal on the domain surface. Notice that, if $h_c < 0$, we would have a run-away condition; as $T_w - T_\infty$ increases, more energy would be pumped into the object, which would make T_w increase further, etc. The implication is that α and β in (301) cannot be of arbitrary sign. This restriction is also evident in the matrix formulation of the Robin boundary condition.

For now, we leave this derivation as an exercise.

7 Numerical Quadrature

It is common in spectral and finite elements to use numerical quadrature to evaluate all integrals of the types encountered in (261). With nodal basis functions on the GLL points, one has a choice of either evaluating the quadrature directly on the GLL points or evaluating the quadrature on a finer set of points. In the latter case, one usually interpolates to Gauss-Legendre quadrature points as these are more accurate for a given number of function evaluations. As a reminder, the Gauss Legendre (GL) quadrature points are the M roots η_k of the Mth-order Legendre polynomial, $P_M(x)$, all of which are in the interval $(-1, 1)$. The corresponding weights are

$$w_k \;=\; \frac{2}{(1-\eta_k^2)\,(P_M'(\eta_k))^2}. \tag{303}$$

M-point GL quadrature is exact for all polynomials in $\mathbb{P}_{2M-1}$. The Gauss Lobatto Legendre (GLL) quadrature points are the $N+1$ roots ξ_k of $(1-x^2)P_N'(x)$, which are in the closed interval $[-1, 1]$. The weights are

$$\rho_k \;=\; \begin{cases} \frac{2}{N(N+1)} & \xi_k \;=\; \pm 1, \\ \frac{2}{N(N+1)\,(P_N'(\xi_k))^2}, & \text{otherwise.} \end{cases} \tag{304}$$

The $(N+1)$ GLL quadrature rule is exact for all polynomials in $\mathbb{P}_{2N-1}$. By interpolating from $(N+1)$ GLL points to $(N+1)$ GL points the order of the quadrature rule increases from $2N-1$ to $2N+1$.

Accurate quadrature is a significant concern for finite element methods (FEMs), where the approximation order is relatively low even for the p-type finite element method, where p refers to the polynomial order ($p = N$ in the case of spectral methods). For $p \leq 4$, which is already very high for the FEM because of their $O(p^6)$ storage and operation costs, it is important to increase the accuracy of the quadrature. For spectral and spectral element methods, the general principle is that one already has a trunctation error arising from the finite N-term expansion and exact quadrature is therefore not necessary. As long as the contribution of each is of the same order, it is fine to use approximate quadrature. There are, however, occasions where exact quadrature is desirable for stability and (less often) accuracy, particularly for advection-dominated problems [Malm et al., 2013]. We will discuss this case in a later section. Presently, we turn to practical application of quadrature in generating our system matrices and data that appear in our 1D model problem.

Let us now define the matrix $\bar{A}$ having entries

$$a_{ij} \;=\; a(l_i, l_j) \;=\; \int_{-1}^{1} \frac{dl_i}{dx}\frac{dl_j}{dx}\,dx, \quad i,j \,\in\, \{0,\ldots,N\}^2. \tag{305}$$

We think of matrices and vectors having an overbar as being associated with the closure of Ω—they account for values on the domain boundary, $\partial\Omega$, as well as those in the interior. Suppose that $\underline{\bar{v}} = R^T\underline{v}$ and $\underline{\bar{u}} = R^T\underline{u}$ represent extended basis coefficients for $u, v \in X_0^N$. Then

$$a(v,u) \;=\; \sum_{j=0}^{N}\sum_{i=0}^{N} v_i\, a_{ij} u_j \;=\; \underline{\bar{v}}^T\bar{A}\underline{\bar{u}} \;=\; (R^T\underline{v})^T\bar{A}R^T\underline{u} \;=\; \underline{v}^T(R\bar{A}R^T)\underline{u} \;=\; \underline{v}^T A\underline{u}. \tag{306}$$

We conclude that our 1D stiffness matrix is given by $A = R\bar{A}R^T$. In fact, this form will hold in all space dimensions and for problems of arbitrary complexity. The simplification is significant. We

simply construct the stiffness matrix (or any other differential operator expressed in weak form) and then apply the restriction matrix and its transpose to enforce the conditions that the test and trial functions live in $X_0^N \subset X^N$. The decision of which boundary conditions to impose and how these should be implemented is deferred to the last step in the problem setup and execution.

Rule. *Evaluate all quadratures and all matrix entries for all elements of the approximation space X^N. Make the restriction to X_0^N algebraically through proper application of R and R^T only as the final step to solving the system.*

7.1 Evaluation of Coefficients via Quadrature

We now turn to the task of forming the spectral coefficients a_{ij} in (262). For $\bar{A}$, we require for $i = 0, \ldots, N$, $j = 0, \ldots, N$,

$$a_{ij} := \int_{-1}^{1} \frac{dl_i}{dx} \frac{dl_j}{dx}\, dx \tag{307}$$

$$\equiv \sum_{k=0}^{N} \rho_k \left.\frac{dl_i}{dx}\right|_{\xi_k} \left.\frac{dl_j}{dx}\right|_{\xi_k} \tag{308}$$

$$\equiv \sum_{k=0}^{N} (\hat{D}^T)_{ik} \hat{B}_{kk} \hat{D}_{kj}. \tag{309}$$

Here, we have introduced the derivative matrix, $\hat{D}$ and (diagonal) mass matrix $\hat{B}$:

$$\hat{D}_{ij} = \left.\frac{dl_j}{dx}\right|_{\xi_i} \tag{310}$$

$$\hat{B}_{ij} = \rho_i \delta_{ij}. \tag{311}$$

We have equivalence in (308) because the GLL rule is exact for all polynomial integrands of degree $\leq 2N-1$ and this integrand is of degree $2N-2$. Our 1D stiffness matrix on $\hat{\Omega} = [-1, 1]$ thus takes the particularly simple form:

$$\bar{A} = \hat{D}^T \hat{B} \hat{D} \tag{312}$$

$$A = R \bar{A} R^T. \tag{313}$$

For the right-hand side of (261) we have several choices. The most straightforward option is to use the $(N+1)$-point GLL quadrature on $f(x)$ to yield,

$$b_i = \rho_i f(\xi_i), \tag{314}$$

If we take $i = 0, \ldots, N$ in (314), we can get the proper restriction of $\underline{b}$ with our restriction matrix R,

$$\underline{\bar{b}} = \hat{B} \underline{\bar{f}} \tag{315}$$

$$\underline{b} = R \underline{\bar{b}} = R \hat{B} \underline{\bar{f}}. \tag{316}$$

With the preceding notations, the solution to (253) is

$$\underline{\bar{u}} = R^T A^{-1} R \left(\hat{B} \underline{\bar{f}} + g \underline{\hat{e}}_N \right) \tag{317}$$

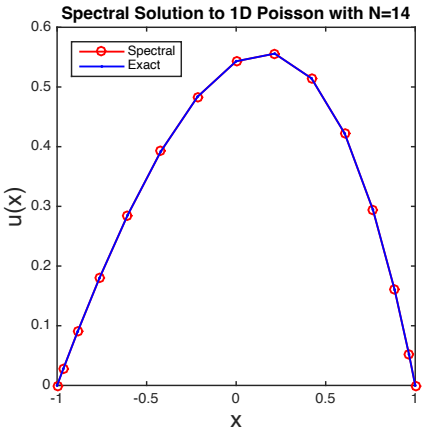
$$= R^T \left(R \bar{A} R^T \right)^{-1} R \left(\hat{B} \underline{\bar{f}} + g \underline{\hat{e}}_N \right), \tag{318}$$

where $\underline{\hat{e}}_N$ is the last column of the identity matrix. This solution is zero on the Dirichlet boundaries and will converge—*exponentially fast for smooth solutions*—to the correct Neumann condition at $x = 1$.

Legendre Spectral Example The following short script implements the Legendre spectral method.

Extend this to an interesting RHS and plot the rate of convergence.

Extend this to the advection-diffusion problem from earlier in the course and plot rates of convergence.




```
[z,w]=zwgll(N);
R=speye(N+1); R=R(2:end-1,:); % Dirichlet-Dirichlet
Dh=deriv_mat(z);
Bh=diag(w);
Ah=Dh'*Bh*Dh;Ah=0.5*(Ah+Ah'); % Enforce exact symmetry
A = R*Ah*R';
f = exp(z);
b = R*Bh*f;
u = A\b; ub= [0; u; 0];
a=exp(-1); b=exp(1); ue=(a+(b-a)*.5*(1+z))-exp(z);
err=max(abs(ue-ub)); format shorte; disp([N err])
plot(z,ub,'ro-',z,ue,'b.-','linewidth',1.3); axis square
title('Spectral Solution to 1D Poisson with N=14','fontsize',12)
xlabel('x','fontsize',16); ylabel('u(x)','fontsize',16);
legend('Spectral','Exact','location','northwest'); pdfplot;
```

7.2 Legendre Spectral Exercises

1. Use the routines `zwgll()`, `zwuni()`, and `interp_mat()` to explore the Gibbs and Runge phenomena for polynomial interpolation by plotting the interpolant for the following functions with either GLL or uniform points on [-1,1]. For each function, use $N = 5 : 5 : 60$ and plot the function and its interpolant several values of N in this interval. Plot the error vs. N on a separate semilogy scale.

$$f(x) = e^x, \tag{319}$$

$$f(x) = \text{sign}(x), \quad f(0) = 0 \tag{320}$$

$$f(x) = \frac{1}{1 + 25x^2} \tag{321}$$

   What conclusions can you make about these choices of nodal points? Which cases exhibit Gibbs phenomena? Which cases exhibit the Runge phenomenon?

2. Use the (nodal) Legendre weighted residual method to solve

$$-\tilde{u}'' + \tilde{u} = \left[\left(\frac{\pi}{4}\right)^2 + 1\right] \sin(\pi(x+1)/4), \quad \tilde{u}(-1) = 0, \tilde{u}'(1) = 0. \tag{322}$$

   Demonstrate that you get spectral convergence by plotting the maximum pointwise error, $\epsilon_N := \|\underline{\tilde{u}} - u u_N\|_\infty$, as a function of N on a semilogy plot.

3. Solve, by hand,

$$-\tilde{u}'' = 1, \quad u(0) = 0, u'(1) = 0. \tag{323}$$

   using the nodal Legendre WRT with $N = 1$.

   - How many unknowns are there?
   - What is the A matrix?
   - What is the RHS?
   - What is the error at $x = 1$?

4. Revisit the advection example,

$$-\nu\tilde{u}'' + c\tilde{u}' = 1; \quad \tilde{u}(0) = \tilde{u}(L) = 0, \tag{324}$$

   with $c = 1$, $L = 1$, and $\nu = .01$, $.005$, and $.001$.

   - How does the restriction matrix R change in this case compared to the earlier Dirichlet-Neumann case?
   - On a *loglog* graph, plot the maximum pointwise error, ϵ_N, as a function of N for $N = 5 : 5 : 150$ for each of the three values of ν (one figure, three plots, different colors).
   - On the same graph, plot the grid Peclet number,

$$Pe_g := \frac{ch}{\nu}, \tag{325}$$

     where $h := \min_j \Delta x_j$ is the minimum grid spacing between the nodal points on [0,1]. (Be sure to compute the endpoints... in fact, in this case, $h = (z_2 - z_1)/2$, where $\underline{z} = [z_1\ z_2\ z_{N+1}]^T$ is the vector of points returned by the `zwgll()` call.
   - What correlation can you make between Pe_g and ϵ_N?

8 Projection and Interpolation in d-Space Dimensions

Here we consider some basic approximation problems of the form,

Given data about $\tilde{u}(\mathbf{x})$*, find* $u(\mathbf{x}) \in X^N$ *such that* $u \approx \tilde{u}$*,*

where X^N is a finite-dimensional approximation (or trial) space. The nature of the data and approximation relationship will be made precise shortly, but we could argue that this question is at the heart of the entire course and indeed the entire field of CFD. Presently, we restrict our attention to the questions of projection and interpolation in tensor products of the unit interval, $\hat{\Omega} := [-1, 1]^d$, and invertible images thereof, for $d = 1$, 2, and 3. Functions in the finite-dimensional trial space X^N are represented by a set of *basis functions*, to which we associate basis coefficients, $\underline{u} = \{u_j\}$, that define our approximation, $u(\mathbf{x})$.

8.1 3D Examples

To see the power of the tensor-product interpolation concept, we begin with an example where it is most useful, namely interpolation in 3D. Throughout the text, we will be working with a *computable* tensor-product polynomial $u_N = u(\mathbf{x})$ that approximates a (usually unknown) function, $\tilde{u}$. Wherever possible, we will drop the subscript N for the sake of clarity. Also, in anticipation of mapping the unit cube to more general domains $\mathbf{x} \in \Omega$, we will generally use $\mathbf{r} \in \hat{\Omega}$ as our independent variable, with $\mathbf{r} = [r, s, t]$ being the most common representation.[9]

On $\hat{\Omega} := [-1, 1]^3$, we write the interpolant as

$$u(\mathbf{r}) \;:=\; u(r,s,t) \;=\; \sum_{k=0}^{N}\sum_{j=0}^{N}\sum_{i=0}^{N} l_k(t) l_j(s) l_i(r) u_{ijk}. \tag{326}$$

Here, the l_is are Lagrange polynomial cardinal functions,

- $l_i(r) \in \mathbb{P}_N(r)$
- $l_i(r_j) = \delta_{ij}$,

where $\mathbb{P}_N$ is the space of all polynomials of degree $\leq N$ and δ_{ij} is the Kronecker delta function, which is equal to one when $i = j$ and zero when $i \neq j$. The set of points ξ_j, $i = 0, \ldots, N$ are the *interpolation nodes*, which have a significant influence on the quality of the approximation (326). Throughout the course, we will assume unless otherwise indicated that we are using the same nodal point distribution in each direction. (For multiple reasons, the ξ_is will invariably be chosen to be the Gauss-Lobatto-Legendre, or GLL, quadrature points.)

Let $\tilde{\mathbf{r}}_{lmn} := [\xi_l, \xi_m, \xi_n]$ denote the lmn'th nodal point in $\hat{\Omega}$. Because of the Kronecker-delta property, we have

$$u(\tilde{\mathbf{r}}_{lmn}) \;=\; u(\xi_l, \xi_m, \xi_n) \;=\; \sum_{k=0}^{N}\sum_{j=0}^{N}\sum_{i=0}^{N} l_k(\xi_n) l_j(\xi_m) l_i(\xi_l) u_{ijk} \tag{327}$$

$$= \; u_{lmn}, \tag{328}$$

[9] Later, t will also be the independent variable for time, but there will be no occasion for confusion as each usage will be clear from the context.

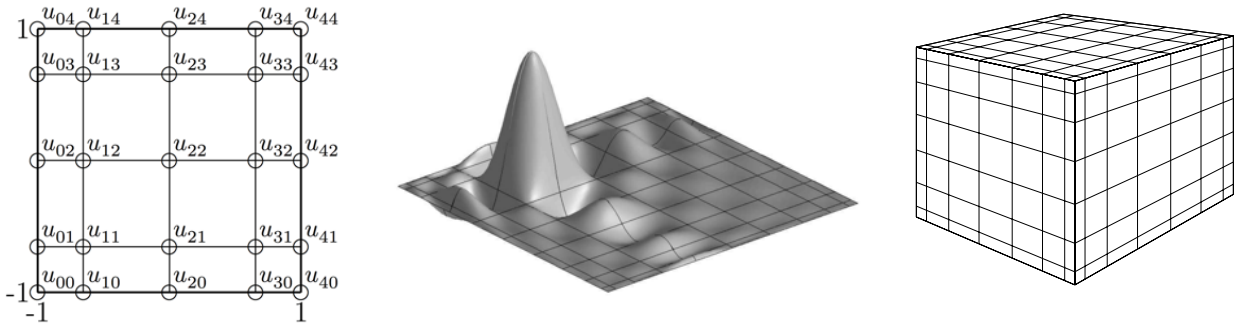


Figure 8: GLL point distributions in 2D and 3D: (left) enumerated degrees-of-freedom for a 2D element of order N=4, (center) a Lagrange cardinal basis function for a 2D element with N=10, (right) GLL point distribution for N=7.

and $u(\mathbf{r})$ will be the interpolant of $\tilde{u}(\mathbf{r})$ at $\tilde{\mathbf{r}}_{lmn}$ provided that we define the basis coefficients to be

$$u_{lmn} \quad := \quad \tilde{u}(\mathbf{r}_{lmn}). \tag{329}$$

Formula (326) is a central pillar of spectral methods. Its simplicity belies the power that it provides and we comment further on this remark at the close of this example.

For the moment, let's observe that (326) gives us a complete and computable representation of $u(\mathbf{r})$ on $\hat{\Omega}$. Let $\underline{u} := \{u_{ijk}\}$ be the set of $(N+1)^3$ known nodal values over the index range $i,j,k \in [0,\ldots,N]^3$ and $\underline{\mathbf{r}}$ similarly be the set of nodal points. For clarity, we will choose the points to be tensor-product distributions of the Gauss-Lobatto-Legendre points, which are the $N+1$ roots of $(1-r^2)P'_N(r)$ on $[-1,1]$, where P_N is the Nth-order Legendre polynomial. We illustrate 2D and 3D distributions of these points in Fig. 8, along with one of the 2D basis functions, $l_i(r)l_j(s)$, in Fig. 8 (center).

8.2 Single Point Interpolation

If we wish to evaluate $u^* := u(\mathbf{r}^*)$, where $\mathbf{r}^* \notin \{\mathbf{r}_{ijk}\}$, then we must evaluate the sum

$$u(\mathbf{r}^*) \quad = \quad \sum_{k=0}^{N}\sum_{j=0}^{N}\sum_{i=0}^{N} l_k(t^*)l_j(s^*)l_i(r^*)u_{ijk} \tag{330}$$

$$\quad = \quad \sum_{k=0}^{N} l_k(t^*)\left[\sum_{j=0}^{N} l_j(s^*)\left(\sum_{i=0}^{N} l_i(r^*)u_{ijk}\right)\right], \tag{331}$$

which is effected through the following sequence,

$$v_{jk} := \sum_{i=0}^{N} l_i(r^*) u_{ijk} \qquad 2(N+1)^3 \tag{332}$$

$$w_k := \sum_{j=0}^{N} l_j(s^*) v_{jk} \qquad 2(N+1)^2 \tag{333}$$

$$u^* := \sum_{k=0}^{N} l_k(t^*) w_k \qquad 2(N+1). \tag{334}$$

Here, we have enumerated at the right the number of operations required to evaluate u^*. (Each add or multiply is considered an operation.) We see that, for high order, the cost of general, single-point, interpolation is $\approx 2(N+1)^3$. Not included in that estimate is the cost of evaluating $l_i(r^*)$. As established in (195) this cost is only $O(N)$ for the entire set, $i = 0, \ldots, N$, which is subdominant to the $O(N^3)$ cost in (332).

8.3 Interpolation to a Tensor Array

In the preceding example, we established that the interpolation cost was $O(N^3)$ to evaluate $u(\mathbf{r})$ at a single point $\mathbf{r}^*$. Consider now the commonly required operation of evaluating u at a *structured array* of points, $\underline{\mathbf{r}}^* = [r^*_{\hat{\imath}}, s^*_{\hat{\jmath}}, t^*_{\hat{k}}]$, $\hat{\imath}, \hat{\jmath}, \hat{k} \in [0, \ldots, M]^3$. For this operation, we have $(N+1)^3$ inputs, u_{ijk}, and $(M+1)^3$ outputs, so we might anticipate a cost of $O(M^3 N^3)$, or $O(N^6)$ if $M \approx N$. The tensor-product structure of the inputs and outputs, however, leads to a significant reduction of cost to only $O(N^4)$. This is the classic spectral complexity estimate that we will anticipate for application of high-order methods in 3D throughout this course. Any higher complexity estimates (e.g., $O(N^5)$) should be judged as *wrong* or questioned at the most fundamental levels. It is this tensor-product structure which led to a revolution in high-order methods for complex geometries, starting with the seminal paper of Orszag in 1980 [**?**].

Clearly, if we proceed with $(M+1)^3$ evaluations of (332) we will have a cost of $2(M+1)^3(N+1)^3$, which is much too high. The secret to fast evaluation is to recognize that we do not have $(M+1)^3$ different values of $r^*_{\hat{\imath}}$, $s^*_{\hat{\jmath}}$, and $t^*_{\hat{k}}$. Rather, *we have only* $(M+1)$ *of each.* Successively substituting these into (332) leads to an expression of the form

$$v_{ijk} := \sum_{i=0}^{N} l_i(r^*_{\hat{\imath}}) u_{ijk} \qquad 2(M+1)(N+1)^3 \tag{335}$$

$$w_{i\hat{\jmath}k} := \sum_{j=0}^{N} l_i(s^*_{\hat{\jmath}}) v_{ijk} \qquad 2(M+1)^2(N+1)^2 \tag{336}$$

$$u^*_{\hat{\imath}\hat{\jmath}\hat{k}} := \sum_{k=0}^{N} l_k(t^*_{\hat{k}}) w_{i\hat{\jmath}k} \qquad 2(M+1)^3(N+1). \tag{337}$$

The operational complexities at the right of (335)–(337) reflect the $2(N+1)$ operations required for each of the *outputs*, v_{ijk}, $w_{i\hat{\jmath}k}$, and $u^*_{\hat{\imath}\hat{\jmath}\hat{k}}$.

We define interpolation matrices $\tilde{J}_r$, $\tilde{J}_s$, and $\tilde{J}_t$ with entries

$$\tilde{J}_{r,\hat{\imath}i} := l_i(r^*_{\hat{\imath}}), \tag{338}$$

$$\tilde{J}_{s,\hat{\imath}i} := l_i(s^*_{\hat{\imath}}), \tag{339}$$

$$\tilde{J}_{t,\hat{\imath}i} := l_i(t^*_{\hat{\imath}}). \tag{340}$$

or simply $\tilde{J}_{\hat{\imath}i} = l_i(r^*_{\hat{\imath}})$ if the output point sets are the same in each direction, $r^*_{\hat{\imath}} = s^*_{\hat{\imath}} = t^*_{\hat{\imath}}$. The cost of generating $\tilde{J}$ is $O(MN)$ ($O(MN)$ matrix entries, each requiring $O(1)$ operations), which is subdominant to the cost of (335)–(337) and very often amortized over thousands of applications when running a code. In matrix form, the interpolation can be expressed as

$$\underline{u}^* = J\underline{u} = (\tilde{J}_t \otimes \tilde{J}_s \otimes \tilde{J}_r)\underline{u}, \tag{341}$$

where J is the *Kronecker*- or *tensor*-product matrix $(\tilde{J}_t \otimes \tilde{J}_s \otimes \tilde{J}_r)$. We describe properties and applications of tensor-product matrices in Section 9. We will use them as a central tool for spectral methods.

Just one example of the importance of tensor-product forms is to recognize that J is in general *full*, with $O(N^6)$ entries, while the total storage for the $\tilde{J}$ trio is only $3(N+1)^2$ (if $M = N$, say). The leading-order data movement, which is a central component of performance overhead in modern high-performance computers, is thus equal to the size of the data, $\underline{u}$, that is, $O(n)$, where $n = (N+1)^3$. For $d = 2$, the storage complexity is higher because each $\tilde{J}$ operator matrix is as large as $\underline{u}$. For $d = 1$, the situation is far worse. The operators are of size n^2, since $n = N + 1$, which makes the computational complexity of high-order methods far worse in 1D because the $\tilde{J}$ operators are *full* rather than sparse (e.g., as would be the case for piecewise linear interpolation).

Another important observation is to recognize that (341) is a *tensor contraction*, which can be implemented as a sequence of fast matrix-matrix products [DFM02, chap08].These BLAS3 operations offer some of the most favorable computational complexity estimates in scientific computing with an $O(N)$ computational intensity (the ratio of floating point operations, *flops*, to bytes transferred from memory). The great importance of tensor-product oriented operations has been recognized by hardware vendors, who have started to build tensor-processing-units (TPUs) for machine learning. Unfortunately, TPUs are most often offered only with support for 16-bit arithmetic, rather than the 64-bit required for most scientific computing.

8.4 Exercises

1. Consider a 2D tensor product interpolant,

$$u(r,s) = \sum_{j=0}^{N}\sum_{i=0}^{N} u_{ij} l_i(r) l_j(s), \tag{342}$$

   where the l_is are cardinal Lagrange functions based on the GLL points.

   Plot $u(r,s)$ for $u_{ij} = \delta_{i0}\delta_{j0}$.

   Plot $u(r,s)$ for $u_{ij} = \delta_{i2}\delta_{j0}$.

   Plot $u(r,s)$ for $u_{ij} = \delta_{i2}\delta_{j3}$.

   Plot $u(r,s)$ for $u_{ij} = \delta_{i2}\delta_{j3}$. Take $N = 7$ in each case.

2. Consider the following expansion for coordinates $\mathbf{x} = (x, y)$,

$$\mathbf{x}(r,s) = \sum_{j=0}^{M}\sum_{i=0}^{M} \mathbf{x}^M_{ij} l^M_i(r) l^M_j(s), \tag{343}$$

where the l_i^Ms are cardinal Lagrange functions based on $M+1$ GLL points. Taking $M=1$ corresponds to a bilinear expansion for $\mathbf{x}$. Let $J_c := \tilde{J}_c \otimes \tilde{J}_c$ denote the interpolant from the order M GLL points to the order N points and, for $M=1$, consider

$$\mathbf{x}_{00}^1 = (0,0) \qquad \mathbf{x}_{01}^1 = (0,1)$$
$$\mathbf{x}_{10}^1 = (1,0) \qquad \mathbf{x}_{11}^1 = (2,2).$$

The order-N coordinates corresponding to the linear inputs $\underline{\mathbf{x}}^1 = (\underline{x}^1, \underline{y}^1)$ are obtained as $\underline{x} = J_c\, \underline{x}^c$ and $\underline{y} = J_c\, \underline{y}^c$. Let Ω be defined as the mapping of $\hat{\Omega}$ to the range of (343) and plot $u(r,s)$ on this domain for $u_{ij} = \delta_{i0}\delta_{j0}$ and $N=7$.

3. Consider the equilateral triangle with vertices $\mathbf{x}_j = (\cos\theta_j, \sin\theta_j)$, $\theta_j = 2j\pi/3$, $j = 0$, 1, and 2. We will partition this domain into three congruent quadrilaterals (quads) that share a common vertex at $\mathbf{x} = (0,0)$. The edges of the quads are defined by extending a three line segments from the origin to the midpoint of the edges of the triangle. Each quad has one vertex corresponding to one of the vertices of the triangle and two vertices corresponding to distinct triangle-midside points.

   Construct interpolants on these quads as in the preceding exercise (take $N=9$). On a single graph, make a surface plot of the 3 basis functions (one in each quad) that are 1 at the origin and zero at the other GLL points.

4. Using numerical quadrature, what is the *total area* covered by the three quadrilaterals (i.e., the equilateral triangle) in the preceding example?
   Hint: On each quadrilateral, Ω^e, $e = 1, \ldots, 3$, we have

$$\int_{\Omega^e} f(\mathbf{x})\, d\mathbf{x} \;=\; \int_1^{-1}\int_1^{-1} f(\mathbf{x}(\mathbf{r}))\, \mathcal{J}(r,s)\, dr\, ds. \tag{344}$$

   Here, the *Jacobian*, $\mathcal{J}r,s$ is the scale factor that maps the infitesimal area $dr \times ds$ to the corresponding area in the x-y plane,

$$\mathcal{J}(r,s) \;=\; \frac{d\mathbf{x}}{dr} \times \frac{d\mathbf{x}}{ds}, \tag{345}$$

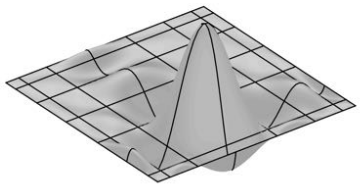
   as illustrated in Fig. 14.

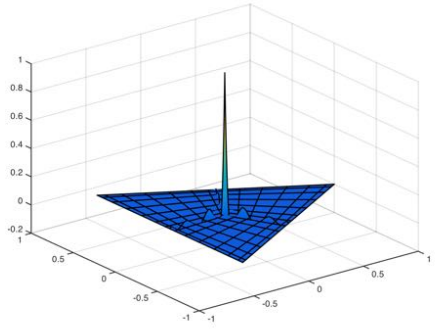
Tensor Product Examples The matlab script `quick_surf.m` (available on the course webpage) implements some of the ideas described in the exercises above. Extend these by

- plotting one or two of the basis functions for $N = 9$ on a *uniform* grid,
- plotting $\frac{\partial u}{\partial r}$ for $u = l_2(r)l_3(s)$ with $N = 8$.

quick_surf.m

```
format compact; format shorte; %close all

N=7;      %% Polynomial Order
M=40+3*N; %% Target. Increase M>>N for smoother output.

%[z,w]=zwglc(N);  % Chebyshev
 [z,w]=zwuni(N);  % Uniform
 [z,w]=zwgll(N);  % GLL

[zm,wm] = zwgll(M); J=interp_mat(zm,z); % Fine points on GLL
[R,S] = ndgrid(z,z);                    % Tensor of points on Omega-hat

F=0*R; i=4; j=3;  F(i,j) = 1;           % Plot phi_i,j

FF=J*F*J';X=J*R*J';Y=J*S*J';
hold off; surfl(X,Y,FF,[1 2 3],[.35 .10 .1 01]);   %%% Lighted grayscale surface
colormap('gray'); shading interp; axis off; hold on;
LX=J*F;  RX=J*R;  RY=J*S;  plot3(RX ,RY ,LX ,'k-','linewidth',1.3)
LX=F*J'; RX=R*J'; RY=S*J'; plot3(RX','RY','LX','k-','linewidth',1.3)
axis equal


% Generate deformed geometry on coarse 2x2 grid

 figure
 for k=0:2;
   theta=(2*k*pi)/3 + 2*pi*(0:3)/3; x=cos(theta); y=sin(theta);
   xmid=zeros(3,1);ymid=xmid;
   xmid(:)=0.5*(x(1:3)+x(2:end)); Xc=[ 0 xmid(3); xmid(1) x(1) ]';
   ymid(:)=0.5*(y(1:3)+y(2:end)); Yc=[ 0 ymid(3); ymid(1) y(1) ]';
   Nc=1; [zc,wc]=zwgll(Nc);
   Jc=interp_mat(z,zc); X=Jc*Xc*Jc'; Y=Jc*Yc*Jc';

   i=1; j=1; F=0*R;
   if k==0;   F(i,j) = 1; end;      % Plot phi_i,j
   if i*j==1; F(i,j) = 1; end;      % Plot phi_i,j
   FF=J*F*J';XF=J*X*J';YF=J*Y*J'; surf(XF,YF,FF,'EdgeColor','none'); hold on
   LX=J*F;  RX=J*X;  RY=J*Y;  plot3(RX ,RY ,LX ,'k-','linewidth',1.3)
   LX=F*J'; RX=X*J'; RY=Y*J'; plot3(RX','RY','LX','k-','linewidth',1.3)
 end;
```





8.5 Differentiation to a Tensor Array

Suppose now we wish to compute the derivative of $u(\mathbf{r})$ given by (326) at a structured set of outputs as in the last example. Specifically, consider

$$\left.\frac{\partial u}{\partial r}\right|_{r_i^*,s_j^*,t_k^*} = \sum_{k=0}^{N}\sum_{j=0}^{N}\sum_{i=0}^{N} l_k(t_{\hat{k}}^*)l_j(s_{\hat{j}}^*)\left.\frac{dl_i}{dr}\right|_{r_i^*} u_{ijk} \tag{346}$$

$$= \sum_{k=0}^{N}\sum_{j=0}^{N}\sum_{i=0}^{N} \tilde{J}_{t,k\hat{k}}\tilde{J}_{s,j\hat{j}}\tilde{D}_{r,i\hat{i}}u_{ijk}, \tag{347}$$

where $\tilde{D}_r$ is the derivative matrix having entries

$$\tilde{D}_{r,ij} := \left.\frac{\partial l_j}{\partial r}\right|_{r_i^*}. \tag{348}$$

We will show how to efficiently evaluate $\tilde{J}$ and $\tilde{D}$ shortly, but we note that we can express then general derivative matrix as

$$\tilde{D} = \tilde{J}\,\hat{D}, \tag{349}$$

where $\hat{D}$ is the (square) matrix of derivatives evaluated at the nodal points,

$$\hat{D}_{ij} \;:=\; \left.\frac{dl_j}{dr}\right|_{\xi_j}. \tag{350}$$

The formula (349) holds because the derivative of a polynomial of degree N (effected by $\hat{D}$) remains in $\mathbb{P}_N$ and the Nth-order interpolant of u_x (effected by $\tilde{J}$) is equivalent to u_x for any $u_x \in \mathbb{P}_N$.

If we again assume identical output point distributions in r, s and t, then we can write

$$\underline{u}^*_r \;=\; D_r\underline{u} \;=\; (\tilde{J}_t \otimes \tilde{J}_s \otimes \tilde{D}_r)\underline{u}, \tag{351}$$

where $\underline{u}_r$ is the set of output values of $\partial_r u$ at $\underline{\mathbf{r}}^*$. By symmetry, we immediately deduce the formulas to compute the other components of the gradient of u,

$$\underline{u}^*_s \;=\; D_s\underline{u} \;=\; (\tilde{J}_t \otimes \tilde{D}_s \otimes \tilde{J}_r)\underline{u} \tag{352}$$

$$\underline{u}^*_t \;=\; D_t\underline{u} \;=\; (\tilde{D}_t \otimes \tilde{J}_s \otimes \tilde{J}_r)\underline{u}. \tag{353}$$

Note that if $\underline{\mathbf{r}}^* = \underline{\mathbf{r}}$ (i.e., the output points are the same as the input points), then $\tilde{J} = \hat{I}$, the identity matrix of order $N+1$, and $\tilde{D} = \hat{D}$, which yields

$$\underline{u}^*_r \;=\; (\hat{I} \otimes \hat{I} \otimes \hat{D})\underline{u} \tag{354}$$

$$\underline{u}^*_s \;=\; (\hat{I} \otimes \hat{D} \otimes \hat{I})\underline{u} \tag{355}$$

$$\underline{u}^*_t \;=\; (\hat{D} \otimes \hat{I} \otimes \hat{I})\underline{u}. \tag{356}$$

The form (354) leads to a savings of $\approx 3\times$ in operation count over (351)–(353).

From the preceding arguments, in it straightforward to deduce that the derivatives for the two-dimensional case can be computed as

$$\underline{u}^*_r \;=\; D_r\underline{u} \;=\; (\tilde{J}_s \otimes \tilde{D}_r)\underline{u} \tag{357}$$

$$\underline{u}^*_s \;=\; D_s\underline{u} \;=\; (\tilde{D}_s \otimes \tilde{J}_r)\underline{u}, \tag{358}$$

for general input-output pairs, and as

$$\underline{u}^*_r \;=\; (\hat{I} \otimes \hat{D})\underline{u} \tag{359}$$

$$\underline{u}^*_s \;=\; (\hat{D} \otimes \hat{I})\underline{u}, \tag{360}$$

for the nodal-to-nodal map case.

When deriving algorithms, we invariably use the matrix-vector product forms of operator evaluations, such as (351)–(359). From an implementation (i.e., performance) standpoint, however, it is important to recognize that the application of tensor-product based matrices can be cast as *matrix-matrix* products. Recalling the original contraction operation (346), we can rewrite (357) as

$$U_r \;=\; \tilde{D}_r\, U\, \tilde{J}'_s \tag{361}$$

$$U_s \;=\; \tilde{J}_r\, U\, \tilde{D}'_s, \tag{362}$$

where we view U as a matrix of entries u_{ij} and U_r and U_s as matrices that hold the respective derivatives, $u_{r,kl}$ and $u_{s,kl}$. The form (361) is a fast BLAS3 operation requiring only $2(N+1)^2$ memory references and $2(N+1)^3$ operations.

Remark. In spectral and spectral element methods, operations like (354)–(356) constitute the bulk ($> 90\%$) of the floating point operations. Implementing them in a fast way is paramount to high performance and to realizing more complex physics or higher Reynolds numbers (by virtue of realizing a larger value of n for the same wall-clock time). Fast routines for these operations, typically based on BLAS3-type operations, are thus a central concern in high-order code development.

8.6 Summary of d-Dimensional Operators

Here we provide a summary overview of d-dimensional interpolation/integration/differentiation for d=1, 2, 3. In all of the examples, we assume that u, $v \in X^N$, which is the basic interpolation space in $\hat{\Omega} := [-1,1]^d$ with no boundary condition restrictions. X^N is spanned by tensor-product polynomials of degree N and the chosen basis functions in each direction are Lagrange polynomials based on the GLL points, ξ_j, $j = 0, \ldots, N$, on $[-1,1]$.

8.6.1 Interpolation Summary

$$\text{1D: } u(x) = \sum_{j=0}^{N} l_j(x)\, u_j \tag{363}$$

$$\text{2D: } u(\mathbf{x}) = \sum_{j=0}^{N}\sum_{i=0}^{N} l_j(y)\, l_i(x) u_{ij} \tag{364}$$

$$\text{3D: } u(\mathbf{x}) = \sum_{k=0}^{N}\sum_{j=0}^{N}\sum_{i=0}^{N} l_k(z) l_j(x)\, l_i(x) u_{ijk} \tag{365}$$

8.6.2 Quadrature Summary

In variational-based numerical methods, we frequently need to evaluate the L^2 inner-product,

$$(v,u) := \int_{\Omega} v(\mathbf{x})\, u(\mathbf{x})\, dV. \tag{366}$$

We define the discrete GLL equivalent to (366) as

$$\text{1D: } (v,u)_N = \sum_{k=0}^{N} v_k \rho_k u_k = \underline{v}^T \bar{B} \underline{u} \tag{367}$$

$$\text{2D: } (v,u)_N = \sum_{l=0}^{N}\sum_{k=0}^{N} v_{kl} \rho_k \rho_l u_{kl} = \underline{v}^T \bar{B} \underline{u} \tag{368}$$

$$\text{3D: } (v,u)_N = \sum_{m=0}^{N}\sum_{l=0}^{N}\sum_{k=0}^{N} v_{klm} \rho_k \rho_l \rho_m u_{klm} = \underline{v}^T \bar{B} \underline{u}, \tag{369}$$

where the mass matrix in each case is defined as a tensor-product of the 1D mass matrices comprising the GLL quadrature weights. With $\hat{B} :=$diag(ρ_k), the d-dimensional mass matrices are

$$\text{1D: } \bar{B} = \hat{B} \tag{370}$$

$$\text{2D: } \bar{B} = \hat{B} \otimes \hat{B} \tag{371}$$

$$\text{3D: } \bar{B} = \hat{B} \otimes \hat{B} \otimes \hat{B}, \tag{372}$$

which are diagonal. As an example, the diagonal mass matrix in 2D is

$$\bar{B} \;:=\; \hat{B}\otimes\hat{B} \;=\; \begin{bmatrix} \rho_0\rho_0 & & & \\ & \rho_1\rho_0 & & \\ & & \ddots & \\ & & & \rho_N\rho_N \end{bmatrix}. \tag{373}$$

For $v, u \in X^N$, the integrand in (366) is a polynomial of degree $2N$. Gauss-Lobatto-Legendre quadrature on $(N+1)$ points is exact for integrands in $\mathbb{P}_{2N-1}$. *Gauss-Legendre* quadrature on $(N+1)$ points, however, is exact for integrands in $\mathbb{P}_{2N+1}$. Let η_k, $k = 0, \ldots, N$ be the GL points on [-1,1] and ω_k be the corresponding quadrature weights. Define the interpolation matrix that maps from the GLL to the GL points,

$$\tilde{J}_{kj} \;:=\; l_j(\eta^k), \tag{374}$$

and the corresponding mass matrix B^M :=diag(ω_k). If $\underline{\tilde{u}} := \tilde{J}\underline{u}$ and $\underline{\tilde{v}} := \tilde{J}\underline{v}$, the full (or consistent) L^2 inner product in 1D is

$$(v,u) \;=\; \underline{\tilde{v}}^T B^M \underline{\tilde{u}} \;=\; \underline{v}^T \tilde{B}\underline{u}, \tag{375}$$

where $\tilde{B} := \tilde{J}^T B^M \tilde{J}$. In 2D and 3D we also have

$$(v,u) \;=\; \underline{v}^T \tilde{B}\underline{u}, \tag{376}$$

with the definitions

$$\text{1D:}\;\; \bar{B} \;=\; \tilde{B} \tag{377}$$

$$\text{2D:}\;\; \bar{B} \;=\; \tilde{B}\otimes\tilde{B} \tag{378}$$

$$\text{3D:}\;\; \bar{B} \;=\; \tilde{B}\otimes\tilde{B}\otimes\tilde{B}, \tag{379}$$

which are *full*. Of course, in applying $\bar{B}$ we would use the tensor product form rather than explicitly forming the matrix. In the case of the diagonal matrix, however, it is often more convenient to just form it explicitly, particularly when there is a nontrivial Jacobian to consider in the context of deformed domains. In this latter case, the diagonal property of the matrix holds, whereas the tensor-product form does not. (In this case, one has to map the Mth-order quadrature points and evaluate the product of the integrand, the Jacobian, and the weights on the elevated mesh.) For sufficiently large N (e.g., $N > 4$), GLL quadrature (366) is often adequate as the quadrature error is on par with the approximation error. A notable exception is in evaluation of the advection operator, where full quadrature is required to preserve skew-symmetry and the imaginary character of the eigenvalues.

8.6.3 Differentiation Summary

Here we consider differentiation that maps u from the GLL points to its derivative on the same points.

$$\text{1D: } \underline{u}_x = \hat{D}\underline{u} \tag{380}$$

$$\text{2D: } \underline{u}_x = (\hat{I} \otimes \hat{D})\underline{u} = \hat{D}U \tag{381}$$

$$\underline{u}_y = (\hat{D} \otimes \hat{I})\underline{u} = U\hat{D}^T \tag{382}$$

$$\text{3D: } \underline{u}_x = (\hat{I} \otimes \hat{I} \otimes \hat{D})\underline{u} \tag{383}$$

$$\underline{u}_y = (\hat{I} \otimes \hat{D} \otimes \hat{I})\underline{u} \tag{384}$$

$$\underline{u}_z = (\hat{D} \otimes \hat{I} \otimes \hat{I})\underline{u} \tag{385}$$

8.7 2D Poisson on $\hat{\Omega}$

Here we consider the canonical Poission problem,

$$-\nabla^2\tilde{u} = -\left(\frac{\partial^2\tilde{u}}{\partial x^2} + \frac{\partial^2\tilde{u}}{\partial y^2}\right) = f \quad \text{in } \Omega = \hat{\Omega} = [-1,1]^2. \tag{386}$$

We will take as boundary conditions,

$$\tilde{u}(-1,y) = \tilde{u}(x,-1) = 0 \tag{387}$$

$$\nabla\tilde{u}\cdot\hat{\mathbf{n}} = 0 \text{ for } x = 1 \text{ or } y = 1. \tag{388}$$

Our discrete formulation is given by the Galerkin projection statement, *Find* $u \in X_0^N$ *such that, for all* $v \in X_0^N$*,*

$$a(v,u) = a(v,\tilde{u}) := \int_\Omega \nabla v\cdot\nabla\tilde{u}\, dV = (v,f). \tag{389}$$

The last equality is found by integrating $a(v,\tilde{u})$ by parts and using (386) to substitute f for $-\nabla^2\tilde{u}$, along with the boundary conditions (387)–(388). Here, X_0^N is the set of functions within our tensor-product Lagrange polynomial space that also vanish on $\partial\Omega_D$.

We will use the same expression for $v(\mathbf{x})$ and $u(\mathbf{x})$. For $u \in X_0^N$, consider

$$u(x,y) = \sum_{j=1}^{N}\sum_{i=1}^{N} l_i(x)l_j(y)\, u_{ij} \tag{390}$$

$$= \sum_{j=0}^{N}\sum_{i=0}^{N} l_i(x)l_j(y)\, \bar{u}_{ij}, \tag{391}$$

where $\bar{u}_{ij} = 0$ for i or $j = 0$, and $\bar{u}_{ij} = u_{ij}$, otherwise.

We introduce a bit of tensor-product notation at this point (but more is given later). Consider the unknown coefficients as an *array*,

$$U \;=\; \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1N} \\ u_{21} & u_{22} & \cdots & u_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NN} \end{bmatrix}. \tag{392}$$

For a given value of j, u_{ij} represents a *row* of data in the computational mesh, corresponding to x_i, $i = 1, \ldots, N$, with fixed of $y = y_j$. Let R_x^T denote the prolongation matrix

$$R_x^T \;=\; \begin{bmatrix} \underline{0}^T \\ I \end{bmatrix}, \tag{393}$$

which extends by zero any N-vector to have a "0" in the first position (corresponding to a homogeneous Dirichlet value at $x = -1$). Applying R_x^T to U, we have

$$R_x^T U \;=\; \begin{bmatrix} u_{01} & u_{02} & \cdots & u_{0N} \\ u_{11} & u_{12} & \cdots & u_{1N} \\ u_{21} & u_{22} & \cdots & u_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NN} \end{bmatrix}, \tag{394}$$

with $u_{0j} \equiv 0$. In a similar fashion, we can define

$$R_y^T \;=\; \begin{bmatrix} \underline{0}^T \\ I \end{bmatrix}. \tag{395}$$

If we *post-multiply* $R_x^T U$ by R_y, we obtain

$$\bar{U} \;:=\; R_x^T U R_y \;=\; \begin{bmatrix} u_{00} & u_{01} & \cdots & u_{0N} \\ u_{10} & u_{11} & \cdots & u_{1N} \\ u_{20} & u_{21} & \cdots & u_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N0} & u_{N1} & \cdots & u_{NN} \end{bmatrix}. \tag{396}$$

with $u_{i0} \equiv 0$. Thus, we prolongate our unknown field, U to $\bar{U}$ (which has zeros on the lower and left boundaries), as

$$\bar{U} \;=\; R_x^T U R_y. \tag{397}$$

As we show in (443), the operation (397) is equivalent to the matrix-vector product,

$$\underline{\bar{u}} \;=\; R^T \underline{u}, \tag{398}$$

with R^T formed as the *Kronecker* (or *tensor*) *product*,

$$R^T \;=\; R_y^T \otimes R_x^T. \tag{399}$$

The equivalence between (397) and (398) depends critically on the assumption that $\bar{\underline{u}}$ and $\underline{u}$ are *lexicographically ordered*,

$$\bar{\underline{u}} \;=\; \begin{pmatrix} u_{00} \\ u_{10} \\ \vdots \\ u_{N0} \\ u_{01} \\ u_{21} \\ \vdots \\ u_{N1} \\ \vdots \\ u_{NN} \end{pmatrix}, \qquad \underline{u} \;=\; \begin{pmatrix} u_{11} \\ u_{21} \\ \vdots \\ u_{N1} \\ u_{12} \\ u_{22} \\ \vdots \\ u_{N2} \\ \vdots \\ u_{NN} \end{pmatrix}. \tag{400}$$

That is, the x (i) index advances most rapidly as one moves down the vector (or *across* the grid).

Generation of system matrices for (389) amounts to evaluating several integrals. We start with the one involving the data $f(\mathbf{x})$. We use expressions for v that are the equivalents of () and (398). In addition, we will approximate the L^2 inner-product, (v, f) by a discrete quadrature-based approximation,

$$(v, f) \;\approx\; (v, f)_N \;:=\; \sum_{k=0}^{N}\sum_{l=0}^{N} v(\xi_k, \xi_l)\, f(\xi_k, \xi_l)\, \rho_k\, \rho_l, \tag{401}$$

where (ξ_k, ξ_l) are the GLL nodal points on $\hat{\Omega}$ and $\rho_k\rho_l$ are the corresponding quadrature weights. Because

$$v \;=\; \sum_{p=0}^{N}\sum_{q=0}^{N} l_p(x)\, l_q(y)\, v_{pq} \tag{402}$$

and the Lagrange basis functions satisfy $l_p(\xi_k) = \delta_{pk}$, we have $v(\xi_k, \xi_l) = v_{kl}$. Similarly, we define $f_{kl} := f(\xi_k, \xi_l)$. In both cases, the range is $k, l \in \{0, \ldots, N\}^2$.

Note that (v, f) is a *number*, whether represented as a continuous inner product or its discrete equivalent. For the latter, we have

$$(v, f)_N \;=\; \bar{\underline{v}}^T \bar{B} \bar{\underline{f}}. \tag{403}$$

Recall that $\bar{\underline{v}} = R^T \underline{v}$ (because $v \in X_0^N$), so the right-hand-side of our variational form is

$$(v, f)_N \;=\; \underline{v}^T R \bar{B} \bar{\underline{f}}. \tag{404}$$

We see that we *restrict* the rhs data through the application of R to $\bar{B}\bar{\underline{f}}$. Notice that we do not *a priori* restrict f—it is allowed to be in L^2 and need not vanish on $\partial\Omega_D$. v, on the other hand, must vanish on $\partial\Omega_D$, which is why R is present in (404).

The next set of integrals to be evaluated are associated with the lhs of (389). Specifically,

$$a(v, u) \;=\; \int_{-1}^{1}\int_{-1}^{1} \underbrace{\frac{\partial v}{\partial x}\frac{\partial u}{\partial x}}_{\mathcal{I}_x} + \underbrace{\frac{\partial v}{\partial y}\frac{\partial u}{\partial y}}_{\mathcal{I}_y}\, dx\, dy. \tag{405}$$

As in the rhs, we replace the integrals with GLL quadrature. Starting with $\mathcal{I}_x$, we have

$$\mathcal{I}_x \approx \sum_{l=0}^{N}\sum_{k=0}^{N} \left.\frac{\partial v}{\partial x}\right|_{\xi_k,\xi_l} \left.\frac{\partial u}{\partial x}\right|_{\xi_k,\xi_l} \rho_k\,\rho_l. \tag{406}$$

Using the prolongated representation of u (meaning we include *all* nodal points), we have

$$u(x,y) = \sum_{j=0}^{N}\sum_{i=0}^{N} l_i(x)\,l_j(y)\,u_{ij} \tag{407}$$

$$= \sum_{j=0}^{N} l_j(y) \sum_{i=0}^{N} l_i(x)\,u_{ij} \tag{408}$$

$$u_x := \frac{\partial u}{\partial x} = \sum_{j=0}^{N} l_j(y) \sum_{i=0}^{N} \frac{dl_i}{dx}\,u_{ij}. \tag{409}$$

8.7.1 Evaluation at alternative quadrature points.

It is instructive to consider evaluation of (409) on a tensor-product of alternative quadrature points (e.g., GL points, $\eta_1,\ldots,\eta_M$). Let $\tilde{J}$ be the interpolation matrix and $\tilde{D}$ the derivative matrix evaluated at these points,

$$\tilde{J}_{ki} = l_i(\eta_k) \tag{410}$$

$$\tilde{D}_{lj} = l_j(\eta_l). \tag{411}$$

Then,

$$\left.\frac{\partial u}{\partial x}\right|_{\eta_k\eta_l} = \sum_{j=0}^{N} l_j(\eta_l) \sum_{i=0}^{N} \frac{dl_i}{d\eta_k}\,u_{ij} \tag{412}$$

$$= \sum_{j=0}^{N} \tilde{J}_{lj} \sum_{i=0}^{N} \tilde{D}_{ki} u_{ij}. \tag{413}$$

If we view the output $u_x(\eta_k,\eta_l)$ as a matrix U_x with entries $U_{x,kl}$ we have

$$U_x = \tilde{D}\,U\,\tilde{J}^T. \tag{414}$$

In vector form, the derivative evaluated on a tensor-product of alternative quadrature points is

$$\underline{u}_x = (\tilde{J}\otimes\tilde{D})\underline{u}. \tag{415}$$

Note that $\tilde{J}$ and $\tilde{D}$ are full $M\times(N+1)$ matrices. The cost of evaluating (414) is $2M^2(N+1)+2M(N+1)^2$, or $4(N+1)^2$ if $M=N+1$ (e.g., if mapping from $(N+1)$ GLL points, to $(N+1)$ GL points, which would improved the order of quadrature by degree 2).

8.7.2 Evaluation at the nodal points.

Note that if one chooses to simply use the $(N+1)$ GLL points (i.e., the nodal points for the underlying Lagrange cardinal functions), then $\tilde{J}\equiv\hat{I}$, the order $N+1$ identity matrix, and $\tilde{D}=\hat{D}$, the standard derivative matrix on the nodal points. The matrix form simplifies in this case to

$$U_x = \hat{D}\,U, \tag{416}$$

with a two-fold reduction in cost compared to (414). We express (414) in vector form as

$$\underline{u}_x \quad = \quad (\hat{I} \otimes \hat{D})\underline{u}, \tag{417}$$

but it is always evaluated as the matrix-matrix product of (414), which is a fast BLAS3 operation requiring only $2(N+1)^2$ memory references and $2(N+1)^3$ operations.

8.7.3 .

8.8 Non-Tensor-Product Bases

It's important to recognize that (326) is a subset of a more general representation,

$$u(\mathbf{r}) \quad = \quad \sum_{i=1}^{n} \phi_i(\mathbf{r}) u_i, \tag{418}$$

where the ϕ_is are Lagrangian cardinal functions in $\mathbb{R}^3$ satisfying $\phi_i(\mathbf{r}_j) = \delta_{ij}$ at nodal points $\mathbf{r}_j$. This form is used, for example, in the case of p-type finite elements on simplices, pyramids, etc. We will use it often—*for notational purposes only*—because of its relative simplicity. Without further information, however, one cannot assume a tensor form for (418). In this case, the interpolation matrix, J is full, with entries

$$J_{ij} \quad := \quad \phi_j(\mathbf{r}_i^*). \tag{419}$$

The derivative matrices are also full. Operator evaluation and storage costs are thus $O(N^6)$, which is prohibitive and is the fundamental reason why p-type finite elements are rarely used for $N = p > 4$. Without the basic tensor-product assumption (326), spectral methods are generally not viable.[10]

9 Kronecker Products

For two matrices A and B, their Kronecker (or tensor) product $C = A \otimes B$ is defined as the block matrix

$$C \quad := \quad \begin{pmatrix} a_{11}B & a_{12}B & \cdots & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & \cdots & a_{2n}B \\ \vdots & \vdots & & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & \cdots & a_{mn}B \end{pmatrix}. \tag{420}$$

Kronecker products have several useful properties in the the solution of PDEs and (more recently) in machine learning, which is driving the development of specialized software and hardware for Kronecker product application and manipulation. (Unfortunately, much of this development is targeting 16-bit operations, which is not sufficient for most scientific computing applications.) There are two main properties of interest, the matrix-product rule and matrix-vector products.

We remark that if A and B are each $N \times N$ matrices, then C is a much larger, $N^2 \times N^2$, matrix with N^4 nonzeros in the case when A and B are full. We also note that $I \otimes B$ and $A \otimes I$ have a

[10] We note, however, that there has been progress on high-order straightsided and curved tetrahedra through means outlined in [Moxey *et al.*,2020].

special structure.

$$I \otimes B \;=\; \begin{pmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{pmatrix}, \tag{421}$$

$$A \otimes I \;=\; \begin{pmatrix} a_{11}I & a_{12}I & \cdots & a_{1n}I \\ a_{21}I & a_{22}I & \cdots & a_{2n}I \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}I & a_{m2}I & \cdots & a_{mn}I \end{pmatrix}. \tag{422}$$

Thus, $I \otimes B$ is block-diagonal, whereas $A \otimes I$ comprises diagonal blocks.

The most important property of Kronecker products for the purposes of PDEs is the matrix-product rule. Suppose we have C and F satisfying

$$C \;=\; A \otimes B, \qquad F \;=\; D \otimes E.$$

Then, straightforward application of (420) reveals that the matrix product CF is given by

$$CF \;=\; (A \otimes B)(D \otimes E) \;=\; AD \otimes BE, \tag{423}$$

under the assumption that the dimensions of (A,D) and (B,E) are such that the products AD and BE make sense. (Note that $C = (A \otimes I)(I \otimes B)$ makes perfect sense from this formula. Post-multiplication of $(A \otimes I)$ by the block-diagonal matrix (421) scales each column of (422) by B, yielding the original definition of C given by (420).)

The product rule (423) leads to several notable properties. To simplify the exposition, we'll assume that A and B are $N \times N$ matrices unless otherwise noted.

- **Inverse.** Applying (423) to $(A \otimes B)$ times $(A^{-1} \otimes B^{-1})$ yields

$$(A \otimes B)(A^{-1} \otimes B^{-1}) \;=\; (AA^{-1} \otimes BB^{-1} \;=\; I_N \otimes I_N \;=\; I_{N^2}. \tag{424}$$

  The inverse of $C = A \otimes B$ is $C^{-1} = A^{-1} \otimes B^{-1}$. Thus, the inverse of the $N^2 \times N^2$ tensor-product matrix C can be found by inverting (or, more typically, factoring) two much smaller $N \times N$ matrices.

- **Eigenvalues.** Let $C = B_y \otimes B_x$. Suppose that there exists a diagonal matrix of eigenvalues, Λ_x, and associated matrix of eigenvectors, S_x, such that $S_x^{-1} B_x S_x = \Lambda_x$, with a similar decomposition for B_y. Then

$$C \;=\; B_y \otimes B_x \;=\; (S_y \Lambda_y S_y^{-1}) \otimes (S_x \Lambda_x S_x^{-1}) \tag{425}$$

$$\;=\; \underbrace{(S_y \otimes S_x)}_{S}\, \underbrace{(\Lambda_y \otimes \Lambda_x)}_{\Lambda}\, \underbrace{(S_y^{-1} \otimes S_x^{-1})}_{S^{-1}}. \tag{426}$$

  Thus, the $N^2 \times N^2$ eigenvalue problem $CS = S\Lambda$ is solved by solving two small $(N \times N)$ eigenvalue problems, which lead to $S = S_y \otimes S_x$ and $\Lambda = \Lambda_y \otimes \Lambda_x$.

- **Fast Diagonalization Method.** Let A be given by the *separable form*,

$$A \;=\; B_y \otimes A_x \;+\; A_y \otimes B_x. \tag{427}$$

If A_x is symmetric and B_x is symmetric positive definite then there exists a diagonal matrix of eigenvalues, Λ_x=diag(λ_j), and associated matrix of eigenvectors, $S_x = [\underline{s}_1 \;\underline{s}_2 \; dots \underline{s}_N]$, satisfying the *generalized* eigenvalue problem,

$$A_x s_j \;=\; \lambda_j B_x s_j, \tag{428}$$

with the eigenvectors normalized to satisfy $s_i^T B_x s_j = \delta_{ij}$. In matrix form,

$$S_x^T A_x S_x \;=\; \Lambda_x \quad\Longleftrightarrow\quad A_x \;=\; S_x^{-T} \Lambda_x S_x^{-1} \tag{429}$$

$$S_x^T B_x S_x \;=\; I_x \quad\Longleftrightarrow\quad B_x \;=\; S_x^{-T} S_x^{-1} \tag{430}$$

Assuming there exists a similar decomposition for y, then

$$A \;=\; B_y \otimes A_x \;+\; A_y \otimes B_x \tag{431}$$

$$\;=\; S_y^{-T} S_y^{-1} \otimes S_x^{-T} \Lambda_x S_x^{-1} \;+\; S_y^{-T} \Lambda_y S_y^{-1} \otimes S_x^{-T} S_x^{-1} \tag{432}$$

$$\;=\; (S_y^{-T} \otimes S_x^{-T}) \underbrace{(I_y \otimes \Lambda_x \;+\; \Lambda_y \otimes I_x)}_{D} (S_y^{-1} \otimes S_x^{-1}). \tag{433}$$

The central expression is diagonal, which makes it particularly simple to solve a system of the form $A\underline{u} = \underline{f}$. Specifically,

$$\underline{u} \;=\; \underbrace{(S_y \otimes S_x)}_{\mathrm{DFT}^{-1}} \underbrace{(I_y \otimes \Lambda_x \;+\; \Lambda_y \otimes I_x)^{-1}}_{\text{inverse of eigenvalues}} \underbrace{(S_y^T \otimes S_x^T)\, \underline{f}}_{\mathrm{DFT}} \tag{434}$$

Here, we have indicated the role played by each term. Working from right to left, the first step is to compute $\hat{\underline{f}} = (S_y^T \otimes S_x^T)\underline{f}$, which is effectively a generalized discrete Fourier transform (DFT). The next step is to divide by the sum of the eigenvalues to get $\hat{\underline{u}} = D^{-1}\hat{\underline{f}}$. The final step is to recombine the eigenvectors to obtain the desired solution $\underline{u} = (S_y \otimes S_x)\hat{\underline{u}}$.

The above *fast diagonalization method* (FDM) was introduced by Lynch, Rice, and Thomas in 1964 in the context of finite differences. It readily extends to the 3D constant coefficient (or separable) case of

$$A \;=\; B_z \otimes B_y \otimes A_x \;+\; B_z \otimes A_y \otimes B_x \;+\; A_z \otimes B_y \otimes B_x, \tag{435}$$

with the following solution,

$$\underline{u} \;=\; (S_z \otimes S_y \otimes S_x)\, D^{-1}\, (S_z^T \otimes S_y^T \otimes S_x^T)\, \underline{f}, \tag{436}$$

where D is the diagonal matrix comprising the one-dimensional eigenvalues,

$$D \;:=\; I_z \otimes I_y \otimes \Lambda_x \;+\; I_z \otimes \Lambda_y \otimes I_x \;+\; \Lambda_z \otimes I_y \otimes I_x. \tag{437}$$

Assuming that the eigenvalues are enumerated in x as $[\lambda_{x,1} \;\ldots\; \lambda_{x,1}]$ and similarly in y and z, then the diagonal entries of D would be of the form

$$D \;=\; \begin{bmatrix} \lambda_{x,1} + \lambda_{y,1} + \lambda_{z,1} & & & \\ & \lambda_{x,2} + \lambda_{y,1} + \lambda_{z,1} & & \\ & & \ddots & \\ & & & \lambda_{x,n_x} + \lambda_{y,n_y} + \lambda_{z,n_z} \end{bmatrix}. \tag{438}$$

- **Matrix-Vector Product.** Suppose $S = S_y \otimes S_x$ and we wish to evaluate the matrix-vector product $\underline{w} = S\underline{u}$, where $\underline{u}$ is assumed to have a natural *dual-subscript* ordering, $\{u_{ij}\}$, as in Fig. 8. If we evaluate $\underline{w}$ by first forming S, then the storage and work rises sharply from $O(N^2)$ to $O(N^4)$. Instead, we evaluate the product in *factored form*,

$$\underline{w} \;=\; (S_y \otimes I_x)\,(I_y \otimes S_x)\underline{u}. \tag{439}$$

The first evaluation generates the vector $\underline{v} := (I_y \otimes S_x)\underline{u}$, which has entries

$$v_{ij} \;=\; \sum_{p=1}^{n_x} (S_x)_{ip} u_{pj}, \quad i \in [1,\ldots,n_x],\; j \in [1,\ldots,n_y]. \tag{440}$$

Assuming S_x is full, the cost of computing $\underline{v}$ is $2n_x$ for each of the $n_y n_x$ entries, or $O(N^3)$ for $N \sim n_x = n_y$.

The next step is evaluation of $\underline{w} = (S_y \otimes I_x)\underline{v}$.

$$w_{ij} \;=\; \sum_{q=1}^{n_y} (S_y)_{jq} v_{iq}, \tag{441}$$

which has cost $2n_y^2 n_x$.

Note that *significant performance gains* (up to an order-of-magnitude) can be realized by recognizing that the doubly-indexed vectors, $\underline{w}$, $\underline{v}$, and $\underline{u}$ can be viewed as corresponding $n_y \times n_x$ matrices, W, V, and U. In this case, the *matrix-vector product* evaluation (439) can be expressed in *matrix-matrix* product form:

$$W = S_x U S_y^T. \tag{442}$$

Or, in general, for any $\underline{v} = (A \otimes B)\underline{u}$, we have

$$V \;=\; B\,U\,A^T. \tag{443}$$

Matrix-matrix products are some of the most efficient operations in numerical computation because they require only $O(N^2)$ memory references for $O(N^3)$ operations, so the form (442) is generally very fast if one invokes well-written utilities. (See, e.g., DFM02, Chapter 8. Matlab also provides fast matrix-matrix products.)

- Matrix-vector products involving third-order tensors of the form $A = A_z \otimes A_y \otimes A_x$ can be evaluated with similar efficiencies. In particular, $\underline{z} = A\underline{u}$ would be evaluated as

$$v_{ijk} \;=\; \sum_{p=1}^{n_x} (A_x)_{ip} u_{pjk} \qquad \underline{v} \;=\; (I_z \otimes I_y \otimes A_x)\underline{u} \tag{444}$$

$$w_{ijk} \;=\; \sum_{q=1}^{n_y} (A_y)_{jq} v_{iqk} \qquad \underline{w} \;=\; (I_z \otimes A_y \otimes I_x)\underline{v} \tag{445}$$

$$z_{ijk} \;=\; \sum_{r=1}^{n_z} (A_z)_{kr} v_{ijr} \qquad \underline{z} \;=\; (A_z \otimes I_y \otimes I_x)\underline{w}, \tag{446}$$

which has a total operation count of $O(N^4)$ and storage count of $O(N^3)$ for the input and output data. It is also possible, with minor effort, to recast (444)–(446) in terms of fast matrix-matrix products. (Recall, in 3D, $N^3 \approx n$.)

10 Time Stepping

Unsteady phenomena pervade science and engineering applications. Turbines, piston engines, blood flow, and wave motion are just a few of the systems that are intrinsically time-dependent. We also have systems where unsteadiness emerges even if the boundary conditions are time-*independent*. Turbulence is a classic example—the boundary conditions and forcing can be steady but instabilities in the (nonlinear) balance of forces can result in unsteady and even chaotic behavior. In the following sections, we'll examine the temporal behavior of some partial differential equations and their discrete counterparts in order to understand the behavior of time advancement strategies.

To illustrate some of the basic issues, we begin with the 1-D heat equation,

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}, \tag{447}$$

with initial and boundary conditions,

$$\text{I.C.} \qquad u(x, t=0) = u^0(x), \tag{448}$$

$$\text{B.C.} \qquad u(x=0,t) = u(x=1,t) = 0. \tag{449}$$

Using an m-point finite difference discretization in space, the (semi-)discrete analog of (447) is

$$\frac{d\underline{u}}{dt} = -\nu A\underline{u}, \qquad \underline{u}(t=0) = \underline{u}^0, \tag{450}$$

with $\underline{u} = (u_1, u_2, \ldots, u_m)^T$ being the vector of unknowns on a uniformly spaced grid, $x_j = j\,h$, $h = 1/(m+1)$, and $A\underline{u}$ representing the standard 2nd-order finite difference approximation. In expanded form, (450) reads

$$\begin{pmatrix} \frac{du_1}{dt} \\ \frac{du_2}{dt} \\ \vdots \\ \frac{du_m}{dt} \end{pmatrix} = -\frac{\nu}{\Delta x^2} \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & \\ & & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{pmatrix}. \tag{451}$$

We see that each value of $\frac{d}{dt}u_j(t)$ is dependent on its own current value and that of its neighbors,

$$\frac{d}{dt}u_j(t) = -\frac{\nu}{\Delta x^2}\left(u_{j+1} - 2u_j + u_{j-1}\right). \tag{452}$$

Equations of the form (450)–(452) are referred to as a *system* of ordinary differential equations (ODEs). In this case, the ODEs are *initial value problems* because we have one initial condition which we use as the starting point to oberseve the evolution of $\underline{u}(t)$. (*The boundary conditions have been built into the discretization.*)

In this section we will develop time marching methods that allow us to compute approximations to $\underline{u}(t)$ at specific time points, t^0, $t^1, \ldots t^n, \ldots$. Typically, we will consider a uniform timestep size Δt such that $t^n = n\Delta t$. The simplest timestepping formula is given by the Euler forward (EF) scheme, which is based on a first-order approximation at time t^{n-1},

$$\left.\frac{d\underline{u}}{dt}\right|_{t^{n-1}} = \frac{\underline{u}^n - \underline{u}^{n-1}}{\Delta t} + O(\Delta t) = -\nu A\underline{u}|_{t^{n-1}}. \tag{453}$$

Dropping the $O(\Delta t)$ residual and rearranging, one obtains the EF update,

$$\underline{u}^n = \underline{u}^{n-1} - \Delta t \nu A \underline{u}^{n-1}. \tag{454}$$

We make several remarks concerning (454). First, EF can also be viewed as an approximation to the *integral* associated with (450),

$$\underline{u}^n = \underline{u}^{n-1} - \int_{t^{n-1}}^{t^n} \nu A \underline{u}(t)\, dt \tag{455}$$

$$\approx \underline{u}^{n-1} - \Delta t \nu A \underline{u}|_{t^{n-1}}\,, \tag{456}$$

where the integrand has been approximated as a *constant* (a polynomial of degree 0) over the interval $[t^{n-1}, t^n]$. Approximation of integrals (by higher-order polynomials) is a common approach to generating timesteppers. Runge-Kutta methods, the trapezoidal (Crank-Nicolson) rule, and Adams-Bashforth methods are all based on this approach. Second, (454) reveals the classic form that $\underline{u}^n$ is nothing other than the old value, $\underline{u}^{n-1}$, plus an $O(\Delta t)$ correction. In the limit that $\Delta t \longrightarrow 0$, one should expect that $\underline{u}^n$ is always a small perturbation from $\underline{u}^{n-1}$ and any code attempting to implement Euler forard (or any other) timestepping should demonstrate such a behavior. (A counter-example, however, is if the system has a constraint to be satisfied at each timestep, as is the case with the incompressible Navier-Stokes equations.) Third, recall that A is symmetric positive definite, with positive real eigenvalues. We thus expect (450) and (454) to lead to a decay in the amplitude of the basis coefficients (for sufficiently long times).

The following Matlab code illustrates application of EF for the heat equation (447), (450). The central kernel is the matrix-vector product, $\underline{u} = G\underline{u}$, which repeatedly applies the Euler forward evolution matrix, $G := I - \nu \Delta t A$. The principal attraction of Euler forward and other explicit methods is that G is *sparse*, so the cost is $O(m)$ per iteration for m gridpoints in any number of space dimensions. That is, there is no fill associated with matrix factorization.

```
%
% HEAT EQUATION, u_t = nu u_xx, USING m-POINT FINITE DIFFERENCE AND E.F.
%

for m=30:31; % Usage:  m=20; heat_ef
  dt = 0.010; T = 1.0; nstep = ceil(T/dt); dt=T/nstep;

  dx = 1./(m+1); x = dx*(1:m)'; xb = [0; x; 1];
  e = ones(m,1); A = spdiags([-e 2*e -e],-1:1, m,m);
  A = A/(dx*dx);

  nu = .05;

  I = speye(m);
  G = I - dt*nu*A;    % Euler forward evolution matrix

  v=-dt*nu*eig(full(A));

  u0=sin(pi*x).*(1+sin(pi*x)+0.7*sin(pi./(1.1-x))); u=u0;


  % PLOT AXIS and Initial Condition:
  ub=[0;u;0]; hold off; plot(xb,ub,'k-',[0 1],[0 0],'k-'); hold on;
  axis([0 1 -.1 2.5]); axis square;
  xlabel('-- x --','FontSize',14); ylabel('u(x,t)','FontSize',14);

  for k=1:nstep;  time=k*dt; if mod(k,20)==0; ub=[0;u;0]; plot(xb,ub,'r-'); ;end;

    u  = G*u;  % Euler Forward

  end;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%      PLOT RESULTS      %%%%%%%%%%%%%%%%%%%%%%%%%%%%

  if m==30;
   title('u_t=\nu u_{xx} on [0,1], \nu=.05, EF w/ finite difference, m=30','FontSize',12);
   print -depsc 'heat_m30.eps'; end;
  if m==31;
   title('u_t=\nu u_{xx} on [0,1], \nu=.05, EF w/ finite difference, m=31','FontSize',12);
   print -depsc 'heat_m31.eps'; end;

  th=0:1000; th=2*pi*th'/1000; x=cos(th)-1; y=sin(th);
  hold off
  if m==30;   %%%% PLOT EIGENVALUES and STABILITY REGION %%%%
     plot(x,y,'k-',real(v),imag(v),'rx',[-3 1],[0 0],'k-',[0 0],[-2  2],'k-')
     axis square; xlabel('Re( \lambda \Deltat )','FontSize',14);
                  ylabel('Im( \lambda \Deltat )','FontSize',14);
     title('\lambda \Deltat for A_{30} and EF Stability Region','FontSize',12)
     text(-1.3,0.2,'Stable','FontSize',18); text(-2.3,1.3,'Unstable','FontSize',18);
     print -depsc 'eig_m30.eps'; end;
  if m==31;
     plot(x,y,'k-',real(v),imag(v),'rx',[-3 1],[0 0],'k-',[0 0],[-2  2],'k-')
     axis square; xlabel('Re( \lambda \Deltat )','FontSize',14);
                  ylabel('Im( \lambda \Deltat )','FontSize',14);
     title('\lambda \Deltat for A_{31} and EF Stability Region','FontSize',12)
     text(-1.3,0.2,'Stable','FontSize',18); text(-2.3,1.3,'Unstable','FontSize',18);
     print -depsc 'eig_m31.eps'; end;
end;
```

The output of this code is illustrated in Fig. 9, which shows the solution at $t = 0$, .05, .10, .15, ... on $t = [0, 1]$ for m=30 (left) and 31 (right). For m=30, the solution behaves as expected. The homogeneous unsteady heat (or *diffusion*) equation acts like a smoother. Portions of the solution with negative curvature ($u_{xx} < 0$) move down; portions with positive curvature move up. The sharper the curvature, the more rapid the motion. At the end, only one smooth slowly decaying mode remains. For m=31, however, we observe a "2-Δx" wave that is growing with time. This mode, the highest frequency wave that can be represented on the grid, exhibits exponential growth. If the computation were continued the solution would rapidly grow to a magnitude beyond the maximum value representable in 64-bit floating point arithmetic.

To summarize, Euler forward is attractive because it is straightforward and easy to code. It is also inexpensive to apply because G is *sparse* (in any space dimension). However, as might be expected, it is only first-order accurate in time and it also exhibits an instability that is sensitive to the underlying discretization parameters, Δx and Δt. In the following sections we explore alternative timestepping strategies and seek to understand the stability and accuracy considerations. We begin with the question of stability in the context of Euler forward.

10.1 Modal Stability Analysis

In the preceding example, both solutions evolved to a single dominant mode, which is associated with the largest (in modulus) eigenvalue of G. To see this, consider the following modal decomposition of the initial condition,

$$\underline{u}^0 = \sum_{k=1}^{m} \hat{u}_k^0 \, \underline{s}_k, \tag{457}$$

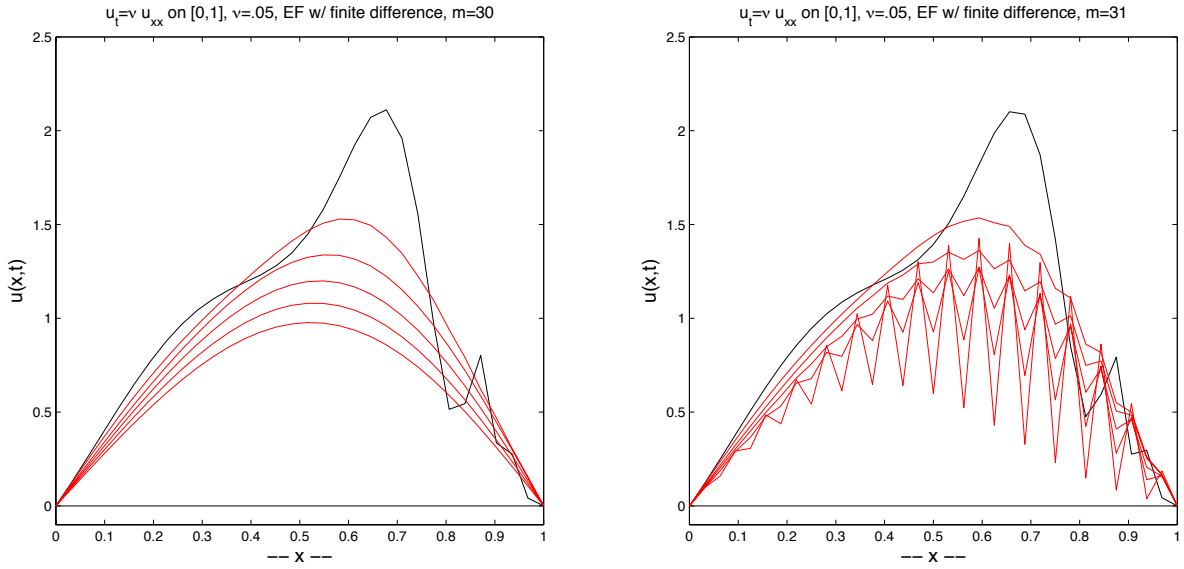


Figure 9: Initial condition (black) and solution (red) of (450) with ν=.05 using Euler forward timestepping with Δt=.01 and 2nd-order finite differences in space with grid spacing $h = 1/(m+1)$ for (right) m=30 and (left) m=31.

where $\underline{s}_k$ is the kth eigenvector of G and assumed to satisfy $G\underline{s}_k = \mu_k \underline{s}_k$ with associated eigenvalue μ_k. Assume that μ_m is the eigenvalue of maximum modulus (i.e., $|\mu_m| > |\mu_k|$, $k \neq m$). Because G is symmetric, the eigenvalues are real and there exist m orthogonal eigenvectors such that $\underline{s}_i^T \underline{s}_j = 0$ for $i \neq j$. As such, any vector in $\mathbb{R}^m$ admits a decomposition of the form (457). We refer to the set of coefficients $\{\hat{u}_k^0\}$ as the *spectrum* of the initial condition $\underline{u}^0$. Here, we assume that all of the coefficients $\hat{u}_k^0$ are nonzero. Even in cases where some coefficients are initially zero, the effect of round-off in the timestepping process tends to inject noise such that the solution invariably has a nontrivial contribution from each eigenmode.

After n rounds of the Euler forward iteration, we find

$$\begin{aligned}
\underline{u}^n &= G^n \underline{u}^0 && (458)\\
&= G^n \left(\sum_{k=1}^{m} \hat{u}_k^0 \, \underline{s}_k \right) && (459)\\
&= \left(\sum_{k=1}^{m} \hat{u}_k^0 \, G^n \underline{s}_k \right) && (460)\\
&= \left(\sum_{k=1}^{m} \hat{u}_k^0 \, \mu_k^n \, \underline{s}_k \right) && (461)\\
&= \mu_m^n \left[\hat{u}_m^0 \, \underline{s}_m + \sum_{k=1}^{m-1} \hat{u}_k^0 \left(\frac{\mu_k}{\mu_m} \right)^n \underline{s}_k \right]. && (462)
\end{aligned}$$

The final expression points to the behavior observed after several timesteps,

$$\underline{u}^n \sim \mu_m^n \hat{u}_m^0 \underline{s}_m. \qquad (463)$$

This asymptotic behavior emerges because $\left| \frac{\mu_k}{\mu_m} \right| < 1$ for $k \neq m$ and the remainder in (462) goes to zero as $n \longrightarrow \infty$. From (463), it is clear that we will have a growing, *unstable*, solution if $|\mu_m| > 1$ and a decaying, *stable*, solution if $|\mu_m| < 1$. If $|\mu_m| = 1$ we say the solution is *neutrally stable*.

The growth factors $\mu_k = \mu_k(G)$ derive from two components, the eigenvalues of A and the design of our timestepper. It is convenient to decouple these contributions by considering a generic system of ODEs,

$$\frac{d\underline{u}}{dt} = L\underline{u}, \qquad \underline{u}(t=0) = \underline{u}^0. \tag{464}$$

(In our present example, $L = -\nu A$.) Euler forward timestepping yields

$$\underline{u}^n = G\underline{u}^{n-1} = (I + \Delta t L)\,\underline{u}^{n-1}. \tag{465}$$

If $\lambda = \lambda(L)$ is an eigenvalue of L then $\mu = (1+\Delta t\lambda)$ is an eigenvalue of G. The stability requirement $|\mu| \leq 1$ translates into a constraint on λ, namely,

$$|1 + \lambda\Delta t| \leq 1. \tag{466}$$

In the complex $\lambda\Delta t$ plane, (466) constitutes a disk of radius 1 centered at (-1,0i). Values of $\lambda\Delta t$ within this disk correspond to stable modes. Values outside correspond to unstable modes. For a scheme to be stable, (466) must be satisfied for all eigenvalues λ_k.

In Fig. 10, we plot the Euler Forward stability region given by (466), along with the scaled eigenvalues, $\Delta t\lambda(L)$, for the m=30 and 31 point discretizations corresponding to the stable and unstable results of Fig. 9. We see that some of the eigenvalues for m=31 are indeed outside the stability region and therefore expect the case for $m = 31$ to blow up. (We need *only one* eigenvalue outside the stability region to have blow-up.) In the next section, we analyze the spectrum of $L = -\nu A$ to understand how these eigenvalues relate to the finite difference discretzation for u_{xx}.

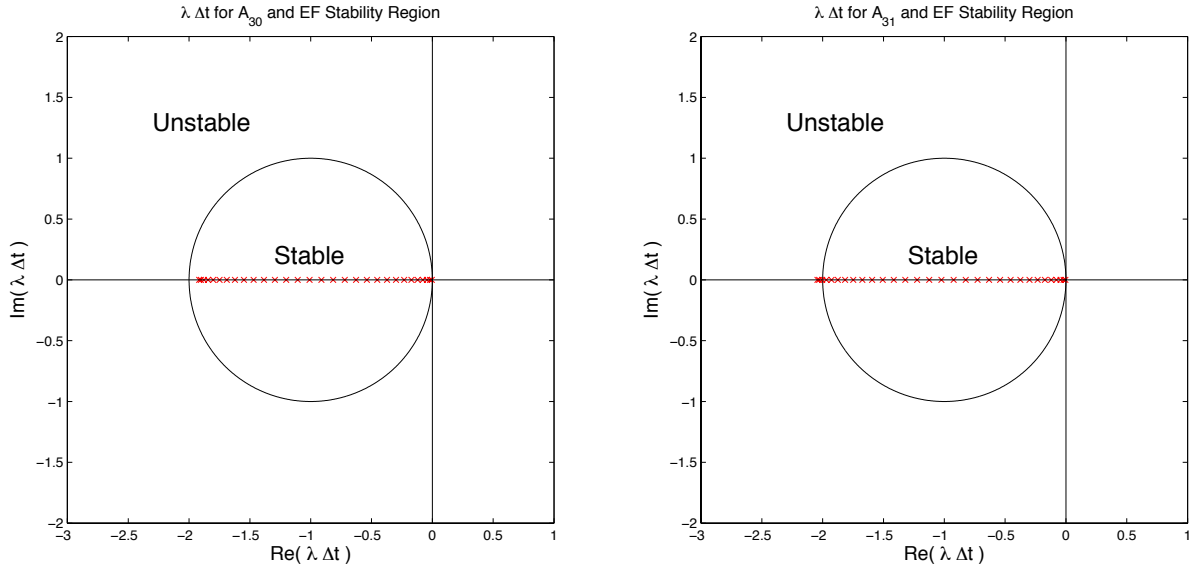


Figure 10: Stability region for Euler forward timestepping and the scaled eigenvalues, $\Delta t\lambda$ for $L = -\nu A$ with $\nu = .05$ and A the m-point 2nd-order finite difference approximation to $-u_{xx}$: (left) m=30, (right) m=31.

We refer to $(1 + \lambda\Delta t)$ as the growth factor, G, associated with Euler forward timestepping. It is the multiplier that results if we apply Euler forward to the scalar ODE,

$$\frac{du}{dt} = \lambda\, u, \qquad u(t=0) = u^0. \tag{467}$$

Euler forward yields

$$u^n = Gu^{n-1}, \qquad G := (1 + \lambda\Delta t), \tag{468}$$

and stability requires $|G(\lambda\Delta t)| \leq 1$, which is a generalization of the system requirement, $|G(\lambda_k\Delta t)| \leq 1$, for $k = 1, \ldots, m$. It is easier to work with scalar systems (467), so this is the preferred approach in analyzing timesteppers.

10.2 Interaction of Eigenvalues and Timesteppers

We will close this introductory section with another example. Before doing so, however, we remark on the general procedure and structure of timesteppers for partial differential equations. The eigenvalues, λ_k are determined by the physics of the problem and by the spatial discretization. The growth factor, $G = G(\lambda\Delta t)$, is determined by the particular timestepper.

Regarding the eigenvalues resulting from the physics, we return to our original heat equation (447). The associated continuous eigenvalue problem is

$$\mathcal{L}\tilde{u}(x) = \tilde{\lambda}\tilde{u}(x), \qquad \tilde{u}(0) = \tilde{u}(1) = 0, \tag{469}$$

where

$$\mathcal{L}\tilde{u}(x) := \nu\frac{d^2\tilde{u}}{dx^2}. \tag{470}$$

The solutions to (469) are $\tilde{u} = \sin k\pi x$ and the eigenvalues are $\tilde{\lambda}_k = -\nu k^2\pi^2$, for $k = 1, 2, \ldots, \infty$.

The discrete counterpart to (469) is

$$L\underline{u} = \lambda\underline{u}, \qquad L := -\nu A. \tag{471}$$

Remarkably, the eigenvectors for the finite difference (and linear finite element) method with uniform grid spacing, Δx, are the same as their continuous counterparts. That is, they are sine functions with wavenumber k. If we denote the kth eigenvector of L by $\underline{z}_k$, then its jth component is

$$(\underline{z}_k)_j = \sin k\pi x_j, \qquad x_j = j\Delta x = j/(m+1). \tag{472}$$

To find the associated eigenvalues, we apply L to $\underline{z}_k$. Let $\underline{u} = \underline{z}_k$ and $\underline{w} = L\underline{u}$. Then

$$w_j = \nu\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} \tag{473}$$

$$= \frac{\nu}{\Delta x^2}\left[\sin(k\pi x_{j+1}) - 2\sin(k\pi x_j) + \sin(k\pi x_{j-1})\right]. \tag{474}$$

At this point, we invoke a trigonometric identity,

$$\sin(a+b) + \sin(a-b) = 2\sin(a)\cos(b), \tag{475}$$

and note that $x_{j\pm1} = x_j \pm \Delta x$. Using these two results (473) becomes,

$$w_j = \frac{\nu}{\Delta x^2}\left[2\sin(k\pi x_j)\cos(k\pi\Delta x) - 2\sin(k\pi x_j)\right] \tag{476}$$

$$= -\frac{2\nu}{\Delta x^2}\left[1 - \cos(k\pi\Delta x)\right]\sin(k\pi x_j) \tag{477}$$

$$= -\frac{2\nu}{\Delta x^2}\left[1 - \cos(k\pi\Delta x)\right]u_j \tag{478}$$

$$= \lambda_k\, u_j. \tag{479}$$

We have thus established that the eigenvalues of L for the heat equation are

$$\lambda_k = -\frac{2\nu}{\Delta x^2}\left[1-\cos(k\pi\Delta x)\right]. \tag{480}$$

Like their continuous counterpart, they are all negative, which implies that all modes should decay. We are typically interested in the extreme ends of the spectrum, that is, the smallest and largest (in magnitude) eigenvalues. For small values of $k\Delta x$, we have $\lambda_k = \tilde{\lambda}_k + O(k^2\Delta x^2)$ as can be seen from a Taylor series expansion for (480).

$$\lambda_k = -\frac{2\nu}{\Delta x^2}\left[1-\left(1-\frac{1}{2!}k^2\pi^2\Delta x^2+\frac{1}{4!}k^4\pi^4\Delta x^4+\text{h.o.t.}\right)\right] \tag{481}$$

$$= -\nu k^2\pi^2\left(1-\frac{1}{12}\pi^2(k\Delta x)^2+\text{h.o.t.}\right). \tag{482}$$

The smaller eigenvalues (k small) are quite close to their physical counterparts, $-\nu k^2\pi^2$. On the other hand, as $k \longrightarrow m$, we have from (480) $\lambda_k \longrightarrow -4\nu/\Delta x^2 = -4\nu(m+1)^2$, wherase the mth eigenvalue for the continuous problem is $\tilde{\lambda}_m = -\nu\pi^2 m^2$. The ratio $\tilde{\lambda}_m/\lambda_m$ is thus approximately $\pi^2/4$ as $m \longrightarrow \infty$.

In summary, for low wavenumbers k, we have $\lambda_k \sim \tilde{\lambda}_k = -\nu k^2\pi^2$. For large wavenumbers, we have $\lambda_k \sim -4\nu/\Delta x^2$. It is the larger wavenumbers that give rise to instabilities with explicit timesteppers such as Euler forward. Specifically, a sufficient condition for stability for EF is

$$1 > |1+\lambda\Delta t| \tag{483}$$

$$> \left|1-\nu\frac{4\Delta t}{\Delta x^2}\right|, \tag{484}$$

which implies, for EF in time, central difference in space,

$$\frac{2\nu\Delta t}{\Delta x^2} = 1, \tag{485}$$

or

$$\boxed{\Delta t = \frac{\Delta x^2}{2\nu}.} \tag{486}$$

10.3 Implicit Methods

We can see from (486) that, as we refine the mesh in space, we need finer and finer resolution in time, *simply for stability reasons.* This constraint is a characteristic of explicit timesteppers, particularly for high-order (e.g., 2nd-order) differential operators, where $\Delta t \sim C\Delta x^2$. This same result carries over to spectral methods, save that for polynomial-based methods, $\Delta x_{\min} \sim C'N^{-1}$, so $\Delta t \sim CN^{-4}$, which follows from the fact that $\lambda_{\max} \sim CN^4$ for the Legendre spectral discretization of the Laplace operator. Moreover, in the case of the heat equation, the modes that are forcing the use of a small timestep are generally uninteresting; It is the high-frequency modes, which are under-resolved on the grid, that very rapidly decay. This stability constraint on timestep size can be mitigated by using *implicit methods*, in which terms on the right-hand side of the evolution equation (450) are evaluated at the new time, t^n. Once again, we use a first-order Euler method to illustrate the basic principles.

Here, the implicit analog to (464)–(465) is the Euler backward (EB) method,

$$\left.\frac{d\underline{u}}{dt}\right|_{t^n} = \frac{\underline{u}^n-\underline{u}^{n-1}}{\Delta t}+O(\Delta t) = L\underline{u}^n. \tag{487}$$

Dropping the $O(\Delta t)$ residual and rearranging, one obtains the EB update,

$$(I - \Delta t L)\,\underline{u}^n = \underline{u}^{n-1}. \tag{488}$$

EB requires solving a system at each timestep, which requires either matrix factorization or the use of iterative methods. In higher space dimensions, iterative methods with $O(m)$ or $O(m \log m)$ computational complexity can be realized through the use of multigrid preconditioning combined with Krylov subspace projection schemes such at conjugate gradient iteration. In any case, the work per step for an implicit scheme is generally greater than that for an explicit scheme but, as we now show, the reward is avoidance of an unduly small timestep resulting from stability considerations.

Determination of Stability for EB

Our stability analysis starts with EB applied to the scalar model problem (467),

$$(1 - \Delta t\lambda)\,u^n = u^{n-1}, \tag{489}$$

from which

$$u^n \;=\; \frac{1}{1-\lambda\Delta t}\,u^{n-1} \;=\; G\,u^{n-1}. \tag{490}$$

Here, the growth factor $G := (1-\lambda\Delta t)^{-1}$ is a rational polynomial in $\lambda\Delta t$. We will have $|G| < 1$ for all points in the $\lambda\Delta t$-plane that are outside the disk of radius 1 that is centered at $(1, 0i)$. For the heat equation, all eigenvalues of L are on the negative real axis and the denominator in (490) is thus > 1, which implies that Euler backward for this problem is *stable*.

10.4 Higher-Order Timesteppers

EF and EB are first-order one-step methods. For the scalar model problem (467) we have

$$\text{EF:} \qquad u^n = (1+\lambda\Delta t)u^{n-1}, \tag{491}$$

$$\text{EB:} \qquad u^n = \frac{1}{1-\lambda\Delta t}u^{n-1}. \tag{492}$$

Both EF and EB yield function updates of the form $u^n = Gu^{n-1}$, where G is either a polynomial or a rational polynomial in $\lambda\Delta t$. We note that the exact solution can also be cast in this form

$$\tilde{u}^n \;=\; e^{\lambda\Delta t}\tilde{u}^{n-1} \tag{493}$$

$$\;=\; \left(1 + \lambda\Delta t + \frac{(\lambda\Delta t)^2}{2!} + \frac{(\lambda\Delta t)^3}{3!} + \ldots\right)\tilde{u}^{n-1}. \tag{494}$$

We can see that there is an $O(\Delta t^2)$ discrepancy between (491) and (493), which we refer to as the *local truncation error* (LTE). The LTE indicates the magnitude in the error as we move from t^{n-1} to t^n, for n=1,2,.... If T is the desired final time for integration of (467) and we require $n_{final} = T/\Delta t$ steps to reach that point, then we accumulate the LTE on each step, such that our final error is

$$\text{GTE} \;=\; n_{final}\text{LTE} \tag{495}$$

$$\;=\; n_{final}O(\Delta t^2) \tag{496}$$

$$\;=\; O(n_{final}\Delta t^2) \tag{497}$$

$$\;=\; O(\Delta t T) \tag{498}$$

$$\;=\; O(\Delta t)T. \tag{499}$$

We refer to GTE as the *global truncation error*. It is the error in the final result at time T. Note that, if we reduce Δt, we must increase $n_{final} = T/\Delta t$ so that we compare errors at the same final time T. Invariably, $LTE = \Delta t GTE$. Moreover, GTE scales as T. Longer time integration thus implies a need for a smaller truncation error. In the case of EB and EF the GTE is $O(\Delta t)$, that is, the methods are first order in time. We would expect therefore that reducing Δt by a factor of 2 (and doubling the number of timesteps) would reduce the temporal error at time T by a factor of 2.

To generate higher-order methods one has several choices. Among these, we will discuss high-order backward-difference formulae, Adams-Bashforth methods, and Runge-Kutta schemes. The main properties of interest for each method are, *stability*, *accuracy*, *cost*, and *data dependencies*.

10.5 Backward-Difference Approximations

We start with backward-difference formulae of order k (BDFk). These methods offer flexibility in terms of order and stability. Although classic BDFk is implicit, we show how they can be combined with extrapolation to develop explicit or even semi-implicit methods.

The idea behind BDFk is to approximate $\frac{du}{dt}$ at time the current timestep, t^n, with a finite difference formula based on the unknown value, u^n, and known past values $u^{n-1}, u^{n-2}, \ldots, u^{n-k}$. One way to generate the finite difference formula is to fit an interpolating polynomial of degree k through the solution $u(t)$ at time points t^n, $t^{n-1}, \ldots, t^{n-k}$ and evaluate the derivative of this polynomial at the current timestep level, t^n. The situation is as pictured in Fig. 11. For *uniform* Δt, the formulas for $k = 1$, 2, and 3 are

$$\text{BDF1:} \quad \tfrac{\partial u}{\partial t}\big|_{t^n} = \frac{u^n - u^{n-1}}{\Delta t} + O(\Delta t) \tag{500}$$

$$\text{BDF2:} \quad \tfrac{\partial u}{\partial t}\big|_{t^n} = \frac{3u^n - 4u^{n-1} + u^{n-2}}{2\Delta t} + O(\Delta t^2) \tag{501}$$

$$\text{BDF3:} \quad \tfrac{\partial u}{\partial t}\big|_{t^n} = \frac{11u^n - 18u^{n-1} + 9u^{n-2} - 2u^{n-3}}{6\Delta t} + O(\Delta t^3). \tag{502}$$

The right hand side of the ODE can either be evaluated directly at time t^n, in which case the method is implicit, or by using kth-order extrapolation.

To illustrate the procedure, we consider the case k=2 using the model problem

$$\frac{du}{dt} = g(u,t) + f(u,t). \tag{503}$$

To begin, evaluate each term at time t^n,

$$\left.\frac{du}{dt}\right|_{t^n} = g(u,t)|_{t^n} + f(u,t)|_{t^n}\,. \tag{504}$$

and choose a method of approximation for each. For the time derivative, we use the BDF2 formula (501). If g is nonlinear and governing relatively slowly evolving behavior, it might be most conveniently evaluated explicitly using 2nd-order extrapolation. Conversely, if f is governing fast behavior, one might need to treat it implicitly. Such an approach gives rise to the following semi-implict scheme,

$$\frac{3u^n - 4u^{n-1} + u^{n-2}}{2\Delta t} + O(\Delta t^2) = \left(2g^{n-1} - g^{n-2} + O(\Delta t^2)\right) + f(u^n, t^n), \tag{505}$$

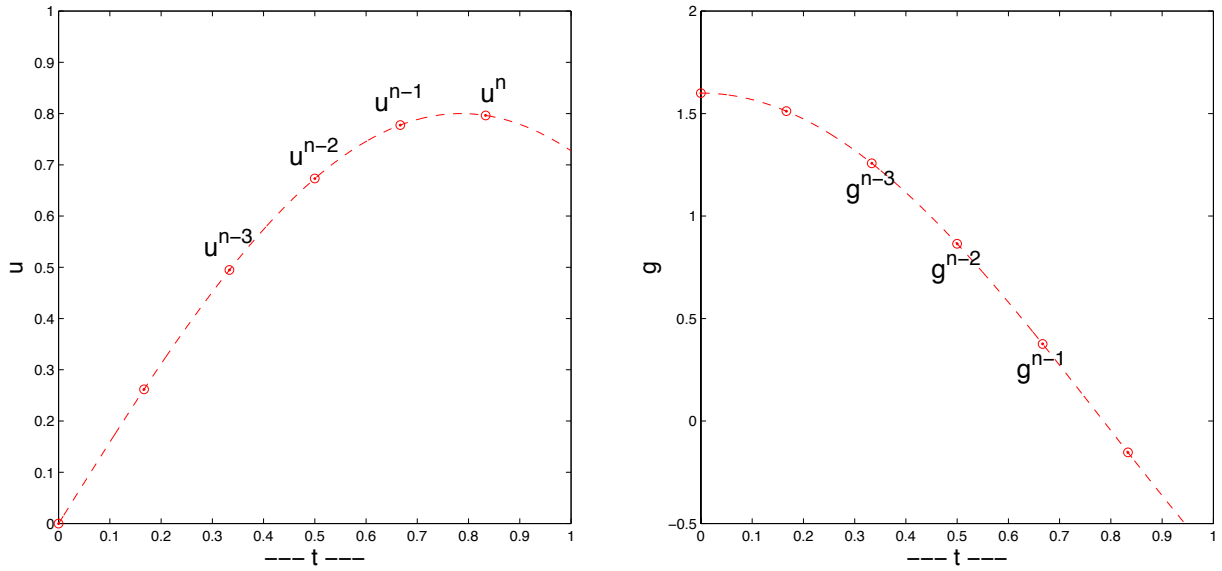


Figure 11: Time points for $u(t)$ and $g(t)$ used in BDFk timestepping.

which is equivalent to (503) because the (global) truncation errors are included. Dropping the these error terms and rearranging yields the semi-implicit BDF/EXT2 update:

$$\frac{3}{2}u^n - \Delta t f(u^n, t^n) \;=\; \frac{1}{2}\left(4u^{n-1} - u^{n-2}\right) + \Delta t\left(2g^{n-1} - g^{n-2}\right). \tag{506}$$

A common situation where semi-implicit schemes are needed is the convection-diffusion equation,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} \;=\; \nu\frac{\partial^2 u}{\partial x^2}, \tag{507}$$

which, when equipped with appropriate initial conditions and boundary conditions takes the form

$$\frac{du}{dt} \;=\; -cD\underline{u} - \nu A\underline{u}, \qquad \underline{u}(t=0) = \underline{u}^0. \tag{508}$$

Here, we assume a spatial discretization based on, say, one dimensional finite difference approximations for which the diffusive eigenvalues scale like $-\nu/\Delta x^2$. As we shall see shortly, we can expect that the convective term has eigenvalues scaling as $\hat{\imath}c/\Delta x$. In the limit $\Delta x \longrightarrow 0$ it is clear that the diffusive processes are very fast and best treated implicitly for stability. If we use second-order extrapolation for convection, we have the following semi-implicit scheme

$$\left(\frac{3}{2}I + \nu\Delta t A\right)\underline{u}^n \;=\; \frac{1}{2}\left(4\underline{u}^{n-1} - \underline{u}^{n-2}\right) - \Delta t\left(2C\underline{u}^{n-1} - C\underline{u}^{n-2}\right), \tag{509}$$

where we have introduced the convection operator $C = cD$, with $D\underline{u}$ the approximation to the first derivative of $u(x)$.

The amount of work to advance the higher order implicit timesteppers is roughly the same as that for EB. One important difference, however, is that BDFk and BDF/EXTk require previously known solutions that are not available at time t^0. A common approach to starting kth-order multistep methods of this type is to bootstrap the solution using order l, $l = 1, \ldots, k-1$ for steps 1 through $k-1$, and a kth-order method from that point on. Such an approach is feasible if one does not care about the initial conditions (e.g., as is the case with turbulent flows, where initial

conditions are invariably synthetic). However, if the final trajectory is strongly dependent on the initial conditions then a higher-order approach must be used for the first few steps. Richardson extrapolation would be an excellent candidate in this respect. (We describe Richardson extrapolation in the Appendix, where it is set in a more general context.)

10.5.1 Stability of BDFk and BDF/EXTk

As before, to study the stability of the timesteppers we consider their application to the model problem $u_t = \lambda u$. We begin with BDF2, which has the implicit update

$$\frac{3u^n - 4u^{n-1} + u^{n-2}}{2\Delta t} = \lambda u^n. \tag{510}$$

Rearranging we have

$$u^n = \frac{2}{3 - 2\lambda\Delta t}\left(4u^{n-1} - u^{n-2}\right) \tag{511}$$

$$= Gu^{n-1}. \tag{512}$$

Unlike the one-step EF and EB methods, it is not immediately transparent from (511) how the growth factor G relates to the discretization parameter $\lambda\Delta t$. Fortunately, for linear homogeneous difference equations of the form (510) one can assume a characteristic solution of the form

$$u^n = Gu^{n-1}, \tag{513}$$

just as one seeks solutions of the form $u = e^{st}$ in the case of homogeneous ODEs with constant coefficients. G will in general be a complex number. For neutral stability, we are interested in the values of $\lambda\Delta t$ such that $|G| \equiv 1$, which will be satisfied by $G = e^{i\theta}$ for $i := \sqrt{-1}$.

Starting with (510) and solving for $\lambda\Delta t$, we have

$$\lambda\Delta t = \frac{3u^n - 4u^{n-1} + u^{n-2}}{2u^n}. \tag{514}$$

With $u^n = Gu^{n-1} = G^n u^0$, we have

$$\lambda\Delta t = \frac{3G^n - 4G^{n-1} + G^{n-2}}{2G^n} \tag{515}$$

$$= \frac{3 - 4G^{-1} + G^{-2}}{2} \tag{516}$$

$$= \frac{3 - 4e^{-i\theta} + e^{-2i\theta}}{2}, \tag{517}$$

where we consider $\theta \in [0, 2\pi]$. The corresponding stability curves for BDF1 (EB) and BDF3 are, respectively,

$$\lambda\Delta t|_{\text{BDF1}} = 1 - e^{-i\theta}, \tag{518}$$

$$\lambda\Delta t|_{\text{BDF3}} = \frac{11 - 18e^{-i\theta} + 9e^{-2i\theta} - 2e^{-3i\theta}}{6}. \tag{519}$$

The BDFk neutral-stability curves are shown in Fig. 12 (left), followed by the generating matlab code that implements (518). BDF1 is seen to be the mirror of the EF stability curve, crossing the positive $Re(\lambda\Delta t)$ curve at $+2$. The *unstable* region grows with increasing order k and, for the third-order case, is seen to actually cross into the negative half of the $\lambda\Delta t$ plane, indicating that BDF3 would not be a good choice for a system possessing imaginary eigenvalues.

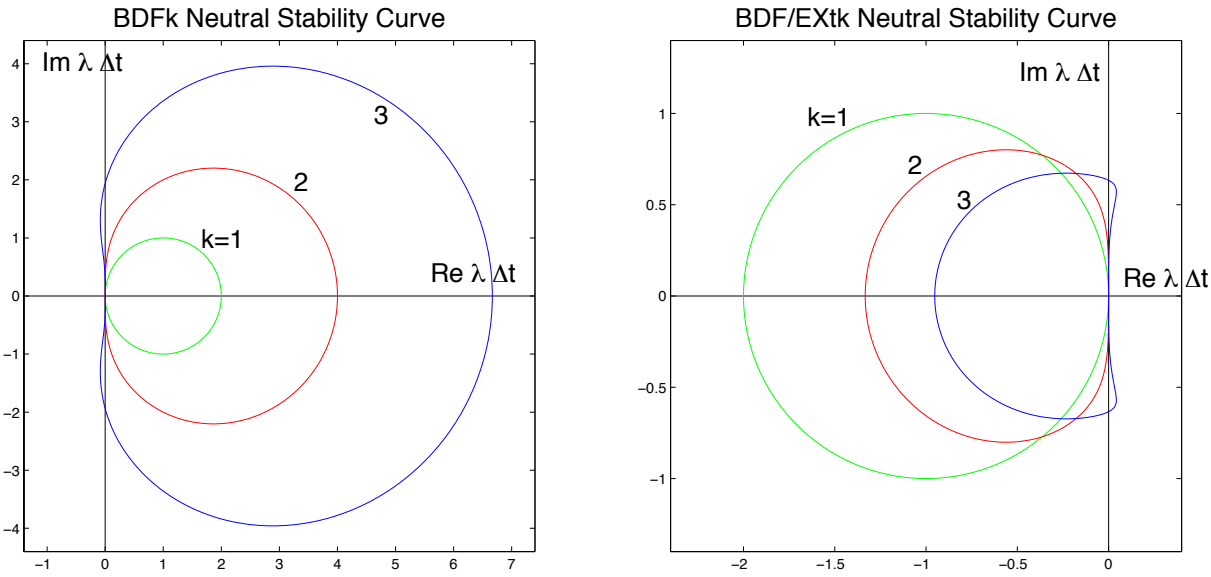


Figure 12: Stability regions for (left) BDFk and (right) BDF/EXTk, k=1,2, and 3.

```
i=sqrt(-1.); th=0:1000; th=2*pi*th'/1000;

em1=exp(-1*i*th); em2=exp(-2*i*th); em3=exp(-3*i*th); e0=1. + 0*em1;
ep1=exp( 1*i*th); ep2=exp( 2*i*th); ep3=exp( 3*i*th);

hold off
xm=-1.4;xM=7.4;ym=-4.4;yM=4.4;xa=[xm xM];y0=0*xa;ya=[ym yM];x0=0*ya;
plot(xa,y0,'k-',x0,ya,'k-'); axis equal; axis([xm xM ym yM]); hold on;
bdf1=(1-1*em1)/1;                              plot(bdf1,'g-');
bdf2=(3-4*em1+em2)/2;                          plot(bdf2,'r-');
bdf3=(11-18*em1+9*em2-2*em3)/6;                plot(bdf3,'b-');
text( 1.65,0.98,'k=1','FontSize',18);
text( 3.25,1.97,'2','FontSize',18); text( 4.63,3.13,'3','FontSize',18);
text(-1.09,4.03,'Im \lambda \Deltat','FontSize',18);
text( 5.60,0.40,'Re \lambda \Deltat','FontSize',18);
title('BDFk Neutral Stability Curve','FontSize',18);print -depsc bdfk.eps

hold off
xm=-2.4;xM=0.4;ym=-1.4;yM=1.4;xa=[xm xM];y0=0*xa;ya=[ym yM];x0=0*ya;
plot(xa,y0,'k-',x0,ya,'k-'); axis equal; axis([xm xM ym yM]); hold on;
bdfext1= bdf1./(em1);                          plot(bdfext1,'g-');
bdfext2= bdf2./(2*em1-em2);                    plot(bdfext2,'r-');
bdfext3= bdf3./(3*em1-3*em2+em3);              plot(bdfext3,'b-');
title('BDF/EXtk Neutral Stability Curve','FontSize',18);
text(-1.65,0.98,'k=1','FontSize',18);
text(-1.10,0.72,'2','FontSize',18); text(-0.83,0.53,'3','FontSize',18);
text(-0.49,1.23,'Im \lambda \Deltat','FontSize',18);
text( 0.08,0.10,'Re \lambda \Deltat','FontSize',18);
print -depsc bdfextk.eps
```

Stability for BDF with explicit extrapolation, BDF/EXTk is established as for standard BDFk. We illustrate the procedure for k=2 starting with $u_t = \lambda u$,

$$\frac{3u^n - 4u^{n-1} + u^{n-2}}{2\Delta t} \;=\; \lambda\left(2u^{n-1} - u^{n-2}\right). \tag{520}$$

Solving for $\lambda\Delta t$ and substituting $u^n = G^n u^0$ we have

$$\lambda\Delta t \;=\; \frac{3G^n - 4G^{n-1} + G^{n-2}}{2(2G^{n-1} - G^{n-2})} \tag{521}$$

$$\;=\; \frac{3 - 4G^{-1} + G^{-2}}{4G^{-1} - 2G^{-2}} \tag{522}$$

$$\;=\; \frac{3 - 4e^{-i\theta} + e^{-2i\theta}}{4e^{-i\theta} - 2e^{-2i\theta}}. \tag{523}$$

Similar formulae result for k=1 and 3 and the results are plotted in Fig. 12 (right).

Remark 1. We note that BDF/EXT1 corresponds to Euler Forward.

Remark 2. The BDF/EXT3 stability region encompasses part of the imaginary axis in the $\lambda\Delta t$ plane, which makes it attractive for systems, such as convection, that have purely or predominantly imaginary eigenvalues. Generally speaking, explicit schemes (Adams-Bashforth, Runge-Kutta, etc.) are required to be *3rd-order or higher* to have this property.

10.6 Adams-Bashforth Methods

In contrast to BDFk/EXTk methods, which are based on approximating the time derivative and data at t^n using finite differences and/or extrapolation, Adams-Bashforth (ABk) methods are derived by approximating the *integral* of the data on the interval $[t^{n-1}, t^n]$. If the integration error on this interval (corresponding to the *LTE*) is $O(\Delta t^{k+1})$, then the global truncation error, will be *GTE*=$O(\Delta t^k)$. Two classes of methods are based on this idea, *Adams-Bashforth* methods, which are explicit, and *Adams-Moulton*. Our focus will be on Adams-Bashforth. (Adams-Moulton methods are not L-stable and thus not of interest when compared to BDFk.)

Adams-Bashforth methods are a somewhat simpler alternative to BDFk/EXTk. To start, we assume that $\underline{u}^{n-j}$ and $\underline{f}^{n-1} = \underline{f}(\underline{u}^{n-j}, t^{n-j})$ are known for $j = 1, \ldots, k$, with k being the order of the ABk scheme. We generate a scheme by approximating the integral in the exact time advancement formula,

$$\underline{u}^n \;=\; \underline{u}^{n-1} \;+\; \int_{t^{n-1}}^{t^n} \underline{f}(t, \underline{u})\, dt.$$

For each order k, $\underline{f}$ is approximated by a polynomial of degree $k-1$ that passes through $[t^{n-j}, \underline{f}^{n-j}]$, $j = 1, \ldots, k$. The approximate integral (quadrature) is linear in the data values, $\underline{f}^{n-j}$, meaning that result is simply the sum of some predetermined weights, dependent on the t^{n-j}s, times the data:

$$\int_{t^{n-1}}^{t^n} f(t, \underline{u})\, dt \;=\; \sum_{j=1}^{k} w_j f^{n-j} \;+\; O(\Delta t^{k+1}).$$

For the particular case of uniform stepsize, Δt, the formulas and local truncation errors for various

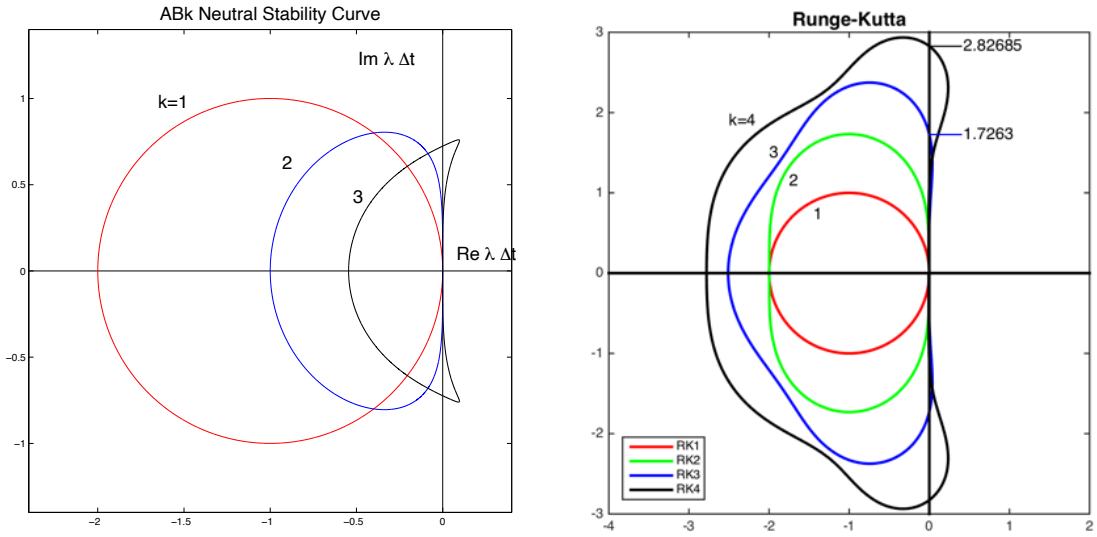


Figure 13: Stability regions for (left) ABk and (right) RKk.

orders k are listed below.

$$\text{AB1:} \qquad \int_{t^{n-1}}^{t^n} f(t,\underline{u})\,dt \;=\; \Delta t f^{n-1} \;+\; O(\Delta t^2)$$

$$\text{AB2:} \qquad \int_{t^{n-1}}^{t^n} \underline{f}(t,\underline{u})\,dt \;=\; \Delta t \underline{f}^{n-1} \;+\; \frac{\Delta t}{2}\left[\frac{\underline{f}^{n-1}-\underline{f}^{n-2}}{\Delta t}\right] \;+\; O(\Delta t^3)$$

$$= \Delta t\left(\frac{3}{2}\underline{f}^{n-1} - \frac{1}{2}\underline{f}^{n-2}\right) \;+\; O(\Delta t^3)$$

$$\text{AB3:} \qquad \int_{t^{n-1}}^{t^n} \underline{f}(t,\underline{u})\,dt \;=\; \Delta t\left(\frac{23}{12}\underline{f}^{n-1} - \frac{16}{12}\underline{f}^{n-2} + \frac{5}{12}\underline{f}^{n-3}\right) \;+\; O(\Delta t^4)$$

The corresponding stability regions for ABk are shown in Fig. 13(left). Note that AB1 corresponds to Euler forward and that, of the three regions, only AB3 encompasses part of the imaginary axis, $\mathcal{I}m(\lambda\Delta t) \in [-0.7236 : 0.7236]$. Also note that for diffusion problems (i.e., $\mathcal{R}e(\lambda(L)) < 0$), AB3's region of stability is smaller than Euler forward.

10.7 Richardson Extrapolation

The basic idea behind Richardson extrapolation is to eliminate the leading-order truncation error in any given approximation by combining approximations computed at two different scales. Let $\tilde{u}$ be an exact value that one is trying to compute and let u_h denote a kth-order approximation to $\tilde{u}$. Specifically,

$$u_h \;=\; \tilde{u} + O(h^k), \tag{524}$$

$$\;=\; \tilde{u} + c_k h^k + O(h^l), \;\; l > k, \tag{525}$$

where c_k is an constant that is independent of h. (This is the precise meaning of $O(h^k)$). It is possible to eliminate the leading-order h^k term without knowing c_k by evaluating (524) at another

value of the discretization parameter, say, $2h$. (The choice $h/2$ is also common.) From (524) we have,

$$u_{2h} = \tilde{u} + c_k(2h)^k + O(h^l) \quad (526)$$

$$= \tilde{u} + c_k 2^k h^k + O(h^l). \quad (527)$$

Taking 2^k times (524) minus (526) yields

$$2^k u_h - u_{2h} = (2^k - 1)\tilde{u} + O(h^l). \quad (528)$$

Rescaling by $(2^k - 1)^{-1}$ gives a a new expression for $\tilde{u}$ that is $O(h^l)$ accurate,

$$\frac{2^k u_h - u_{2h}}{2^k - 1} = \tilde{u} + O(h^l), \quad (529)$$

If the order of the next largest error term is known Richardson iteration can be repeated by evaluating the above expression for, say, $(h, 2h)$ and $(2h, 4h)$ to eliminate the $O(h^l)$ term. Again, we do not need to know c_l nor how the coefficient in front of h^l is modified by prior extrapolation steps.

We illustrate the procedure by considering numerical integration of $\sin \pi x$ on $[0, 1]$ using the trapezoidal rule with $m + 1$ points, $m = 2$, 4, 8, and 16. We take $h = 1/m$. The following matlab code implements two rounds of Richardson, successively eliminating the $O(h^2)$ and $O(h^4)$ error terms to give a final convergence rate of $O(h^6)$.

```
%% Richardson extrapolation applied to I := \int sin(pi x) dx

exact = 2/pi; km=4;
E1=zeros(1,km); E2=zeros(1,km-1); E3=zeros(1,km-2);

for k=1:km; N=2^k; L=pi;
  h=L/N;
  x=h*[0:N'];
  y=sin(x);
  w=h*ones(N+1,1); w(1)=h/2; w(end)=h/2;
  I(k)=w'*y;                      % I(k) is trapezoidal rule
  E1(k)=exact-I(k);               % and is O(h^2) accurate
end;

for j=1:km-1;
   I2(j)=(4*I(j+1)-I(j))/3;       % I2 is first round of Richardson
   E2(j)=exact-I2(j);             % Should be O(h^4)
end;

for j=1:km-2;
   I3(j)=(16*I2(j+1)-I2(j))/15;  % I3 is second round of Richardson
   E3(j)=exact-I3(j);             % Should be O(h^6)
end;

format shorte
I1
I2
I3
E1
E2
E3
```

The results of this code are shown below.

```
h  =     0.5          0.25         0.125        0.0625
-------------------------------------------------------
I  =  5.0000e-01   6.0355e-01   6.2842e-01   6.3457e-01
I2 =  6.3807e-01   6.3671e-01   6.3663e-01
I3 =  6.3661e-01   6.3662e-01
-------------------------------------------------------
E1 =  1.3662e-01   3.3066e-02   8.2023e-03   2.0466e-03
E2 = -1.4514e-03  -8.5679e-05  -5.2811e-06
E3 =  5.3696e-06   7.8796e-08
```

One can clearly see that the error for unextrapolated trapezoidal rule is scaling as $O(h^2)$—a factor of four reduction with each two-fold reduction in h. The error in the final row is seen to be reduced by approximately $2^6 = 64$ with a two-fold reduction in h.

A crucial point in the application of Richardson extrapolation is that we do not necessarily reformulate the integration rule. Rather, we use the *same* rule (and thus, same code) with different values of the discretization parameter h. We then take a combination of the *outputs* (here, the `I`'s) from the integrator to yield an improved estimate of the integral.

Appendix II: Determining Rate of Convergence

Application of Richardson extrapolation formally requires knowledge of the rate of convergence, k, for the sequence in question. What should you do when you don't know the answer and also don't know k? There are two convergence cases to consider. The first is algrebraic convergence of the form we've seen with finite differences in space in time, where the error for N gridpoints (or time points) with a kth-order method is $O(h^k)$, where $h = L/N$ (say). The second is exponential convergence, which is realized for spectral methods, where the error is $O(\sigma^N)$, with $0 < \sigma < 1$. We begin with the algebraic case.

Algebraic Case: error is a power of h as $h \longrightarrow 0$. In the expression below, we have three unknowns on the right-hand side of each equation: $\tilde{u}$, c, and k, plus the unknown higher-order terms (h.o.t.). On the left, we have known (i.e., computable) numbers, u_h, u_{2h}, and u_{4h}.[11]

$$\underline{\text{Computable}} \qquad \underline{\text{Unknown}} \tag{530}$$

$$u_h \quad = \quad \tilde{u} \; + \; ch^k \; + \; \text{h.o.t.} \tag{531}$$

$$u_{2h} \quad = \quad \tilde{u} \; + \; c(2h)^k \; + \; \text{h.o.t.} \tag{532}$$

$$u_{4h} \quad = \quad \tilde{u} \; + \; c(4h)^k \; + \; \text{h.o.t.} \tag{533}$$

With this sequence, we compute the differences of our known quantities to eliminate the unknown analytical solution, $\tilde{u}$ on the right.

- Compute the differences:

$$D_h := u_{2h} \; - \; u_h \quad = \quad c\left(2^k - 1\right) h^k \; + \; \text{h.o.t.} \tag{534}$$

$$D_{2h} := u_{4h} \; - \; u_{2h} \quad = \quad c\left(4^k - 2^k\right) h^k \; + \; \text{h.o.t.} \tag{535}$$

$$= \quad c\, 2^k \left(2^k - 1\right) h^k \; + \; \text{h.o.t.} \tag{536}$$

With the sequence of differences, we compute the ratios to get rid of the contstant c.

- Take the ratio:

$$R_h := \frac{D_h}{D_{2h}} \quad = \quad \frac{c\left(2^k - 1\right) h^k \; + \; \text{h.o.t.}}{c\, 2^k \left(2^k - 1\right) h^k \; + \; \text{h.o.t.}} \tag{537}$$

$$= \quad \frac{1}{2^k} \; + \; \text{h.o.t.} \; \sim \; \frac{1}{2^k}. \tag{538}$$

As $h \longrightarrow 0$, the ratio will approach 2^{-k} (e.g., $\frac{1}{8}$ for $k = 3$) and we compute the rate of convergence by taking the base-2 logarithm of the ratio.

- Order of convergence: $\quad k \sim -\log_2 |R_h|, \;$ as $h \longrightarrow 0$.

We reiterate that this process indicates that the leading-order error is $O(h^k) \sim ch^k$ for some unknown constant, c. One could subsequently estimate c from the expression for the difference, D_h, and then derive an improved estimate, $\tilde{u} \approx u_h - ch^k$. In fact, this is precisely the principle behind Richardson extrapolation. With a sequence of such approximations (for h, $2h$, and $4h$), it's possible to repeat the process to eliminate the next error term. It is necessary, however, to start with four initial approximations (e.g., by augmenting the original solution set with u_{8h}).

Exponential Case: In the exponential case, the ratio R_h will tend towards 0 rather than a constant. In the event of this happy circumstance, extrapolation is not possible—the method is extracting all available information from the N data points that are provided when generating u_h—and one should simply accept a sufficiently converged result.

[11] Typically, the u_{jh} values are cheaper to compute than the u_h values for $j > 1$. Also, one can derive Richardson formulas with a computed sequence $[u_h, u_{2h}, u_{3h}]$ instead of $[u_h, u_{2h}, u_{4h}]$.

11 Multidimensional Advection-Diffusion

Here, we introduce several concepts in the context of the *unsteady advection-diffusion equation* in d space dimensions,

$$\frac{\partial u}{\partial t} + \mathbf{c}\cdot\nabla u \;=\; \nu\nabla^2 u + f, \qquad u|_{\partial\Omega} = 0. \tag{539}$$

For the moment, we consider $\Omega = \hat{\Omega} := [-1,1]^d$, but the majority of our exposition is not resticted to this condition. Furthermore, in anticipation of solving the incompressible Navier-Stokes equations, we consider only divergence-free advecting fields, $\nabla\cdot\mathbf{c} \equiv 0$.

11.1 Weighted Residual Formulation

We begin with the semidiscretization of (539), in which we restrict our search for solution to the trial space X_0^N=span$\{\phi_j\}$, $j = 1,\ldots,n$. We assume that each $\phi_j(\mathbf{x})$ vanishes on $\partial\Omega$ and we also consider test functions $v \in X_0^N$. To apply the *weighted residual technique* (WRT), we seek $u \in X_0^N$ such that, for every $v \in X_0^N$,

$$\int_\Omega v\left[\frac{\partial u}{\partial t} + \mathbf{c}\cdot\nabla u \;-\; \nu\nabla^2 u \;-\; f\right] dV \;=\; 0. \tag{540}$$

Equivalently,

$$(v, u_t) \;=\; -\nu\, a(v,u) \;-\; c(v,u) \;+\; (v,f), \tag{541}$$

where, in addition to our L^2 inner product $(v,u) = \int vu\, dV$ and a inner product, $a(v,u) := \int \nabla v\cdot\nabla u\, dV \equiv (\nabla v, \nabla u)$, we have the advection bilinear form,

$$c(v,u) \;:=\; (v, \mathbf{c}\cdot\nabla u) \;=\; \int_\Omega v\,\mathbf{c}\cdot\nabla u\, dV. \tag{542}$$

In the sequel, we will show how to efficiently evaluate each of these terms in the unit d-dimensional box where tensor-product operators can be applied. Prior to that, we examine some of the properties of (539) and its associated weak form (541).

11.1.1 Energy Conservation

With the WRT for advection-diffusion (541), we no longer have the energy minimization property that we had in the case of the elliptic (and symmetric) diffusion problem, so we do not have a proper best-fit property. However, following our earlier spectral approximation exercise, we can expect to have spectral convergence provided that the exact solution has sufficient regularity and that we take N large enough to ensure that boundary layers are well resolved in cases where advection dominates (i.e., $Pe := L|\mathbf{c}|/\nu \gg 1$, where L is a measure of the domain size).

One property that is ensured with the weak formulation is *energy conservation*, as we now demonstrate. Consider the case where $f(\mathbf{x}) \equiv 0$ and $u^0(\mathbf{x}) \neq 0$. Since (541) holds for all $v \in X_0^N$ it certainly holds for $v = u$. Now consider each of the bilinear forms in (541). The term on the left reads,

$$(u, u_t) \;=\; \int_\Omega u\frac{\partial u}{\partial t}\, dV \;=\; \frac{1}{2}\int_\Omega \frac{\partial u^2}{\partial t}\, dV \;=\; \frac{1}{2}\frac{d}{dt}\int_\Omega u^2\, dV \;=\; \frac{1}{2}\frac{d}{dt}\|u\|^2, \tag{543}$$

which indicates the rate of decay of $\frac{1}{2}\int u^2\,dV$.[12] The first term on the right reads,

$$-\nu a(u,u) \;=\; -\nu\int_\Omega \nabla u\cdot\nabla u\,dV \;<\; 0 \quad \forall\, u\neq 0\in X_0^N. \tag{544}$$

Clearly, in the absence of advection, we have that energy decays through the process of diffusion. The solution tends toward zero by virtue of diffusion coupled with the homogeneous Dirichlet conditions on $\partial\Omega$. Finally, we evaluate (542) for $v=u$. Prior to doing so, however, it is instructive to integrate by parts with the distinct arguments.

$$c(v,u) \;:=\; \int_\Omega v\,\mathbf{c}\cdot\nabla u\,dV \tag{545}$$

$$\;=\; -\int_\Omega u\nabla\cdot(\mathbf{c}v)\;dV \;+\; \underbrace{\int_{\partial\Omega} v\,u\,\mathbf{c}\cdot\hat{\mathbf{n}}\,dS}_{=0 \text{ because } v|_{\partial\Omega}=0} \tag{546}$$

$$\;=\; -\int_\Omega u\,v\,\nabla\cdot\mathbf{c}\,dV - \int_\Omega u\mathbf{c}\cdot\nabla v\,dV \tag{547}$$

$$\;=\; -\int_\Omega u\mathbf{c}\cdot\nabla v\,dV \tag{548}$$

$$\;=\; -c(u,v). \tag{549}$$

In (547), we exploit the fact that $\nabla\cdot\mathbf{c}=0$ to reach the important conclusion that, under appropriate boundary conditions (here, $v=0$ on $\partial\Omega$), the advective bilinear form is *skew symmetric*, $c(v,u) = -c(u,v)$. Equivalently, $c(v,u)+c(u,v)=0$, from which we conclude that $c(u,u)\equiv 0$ for all $u\in X_0^N$. Thus, taking $v=u$ in (541) leads to the important result

$$\frac{1}{2}\frac{d}{dt}\int_\Omega u^2\,dV \;=\; -\nu a(u,u) \;<\; 0. \tag{550}$$

The solution will therefore decay.

In the absence of advection, the anticipated rate of decay will be

$$|u| \;\sim\; \max_k\; C_k e^{-\nu t\lambda_k}, \tag{551}$$

where $\lambda_k\sim C'(k/L)^2$, with L a characteristic domain length and C' and C_k constants. Clearly, the $k=1$ mode, corresponding to the minimum eigenvalue of $a(\cdot,\cdot)$, will be the most slowly decaying mode and will dominate the solution after a short time. With advection, however, we cannot *a priori* assume solution expansions in terms of eigenfunctions of a only; we must also consider the influence of advection, which can drive smooth, low-wavenumber-in-a, solutions to have high wavenumber components. This action is precisely what *stirring* does to a solution. Inhomogeneous fluid/thermal distributions induced by diffusion are stretched into high wavenumber distributions, which then quickly decay. We will soon consider an example of this process where we compare the rate of decay of a stirred field to that of a pure diffusion problem.

11.2 Spectral Expansions in $\hat{\Omega}$

Here we develop fast evaluation approaches for each of the bilinear forms in (541) for the unit box in $\mathbb{R}^2$ and $\mathbb{R}^3$. We begin with the advection matrix, which in some ways is the most complex because of the need to support variable coefficients (i.e., $\mathbf{c}(\mathbf{x})$).

[12] This quantity is sometimes referred to as "energy" in the case where the scalar u is replaced by the vector velocity field and $\frac{1}{2}u^2$ is replaced by the kinetic energy $\frac{1}{2}\rho\mathbf{u}\cdot\mathbf{u}$.

11.2.1 Advection Matrix in 2D

Here we evaluate the skew-symmetric advection term. Let the vector field $\mathbf{c} = [c_x,\, c_y]^T$ denote the advecting velocity, which satisfies $\nabla \cdot \mathbf{c} = 0$. The bilinear form associated with the WRT is

$$c(v,u) := \int_\Omega v\mathbf{c}\cdot\nabla u\, dV \tag{552}$$

$$= \int_\Omega v\left[c_x\frac{\partial u}{\partial x} + c_y\frac{\partial u}{\partial y}\right] dV. \tag{553}$$

As in the case of the Laplace operator, we break this term into two integrals. The first of these is

$$\mathcal{I}_x := \int_\Omega v c_x\frac{\partial u}{\partial x}\, dV \tag{554}$$

$$= \sum_{ij}\sum_{pq} v_{ij}\left[\int_{-1}^{1}\int_{-1}^{1}(l_i(r)l_j(s))\; c_x(r,s)\;\left(\frac{dl_p}{dr}l_q(s)\right) dr\, ds\right] u_{pq}. \tag{555}$$

Here (and in other *variable coefficient* cases), we have no separability because $c_x(r,s)$ is of unknown form. To make progress, we need to introduce a quadrature rule, as is used in practice.

Let's assume that we wish to use an M-point GL quadrature rule where, typically, $M > N$. We rewrite $\mathcal{I}_x$ as

$$\mathcal{I}_x = \sum_{ij}\sum_{pq} v_{ij}\left[\sum_{kl}\rho_k^M\rho_l^M\;(l_i(\eta_k)l_j(\eta_l))\; c_x(\eta_k,\eta_l)\;\left(\left.\frac{dl_p}{dr}\right|_{\eta_k} l_q(\eta_l)\right)\right] u_{pq}, \tag{556}$$

where the quadrature points are η_k, $k = 1,\ldots,M$, and the associated weights are ρ_k^M. We introduce the following matrices.

$$\hat{B}^M := \mathrm{diag}(\rho_k^M) \tag{557}$$

$$\hat{J}_{lq} := l_q(\eta_l) \tag{558}$$

$$\tilde{D}_{kp} := \left.\frac{dl_p}{dr}\right|_{\eta_k} = \left(\hat{J}\hat{D}\right)_{kp} \tag{559}$$

$$B^M := \hat{B}^M \otimes \hat{B}^M. \tag{560}$$

Note that B^M is diagonal, by construction. We also introduce the diagonal matrix C_x^M having entries $c_x(\eta_k,\eta_l)$ at locations corresponding to $\rho_k^M\rho_l^M$ in B^M. For coordinates (η_k,η_l), the ith diagonal entry of C_x^M will hold $c_x(\eta_k,\eta_l)$, where $i = k + M(l-1)$. We illustrate the definition of C_x^M in the boxed example below.

With the preceding definitions, the first contribution to two-dimensional advection can be expressed in matrix form as

$$\mathcal{I}_x = \underline{\bar{v}}^T\left[\left(\hat{J}^T\otimes\hat{J}^T\right)B^M C_x^M\left(\hat{J}\otimes\tilde{D}\right)\right]\underline{\bar{u}}. \tag{561}$$

The contribution from the y-velocity component will be

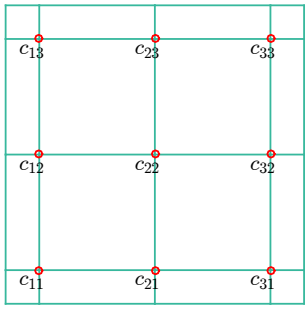
$$\mathcal{I}_y = \underline{\bar{v}}^T\left[\left(\hat{J}^T\otimes\hat{J}^T\right)B^M C_y^M\left(\tilde{D}\otimes\hat{J}\right)\right]\underline{\bar{u}}. \tag{562}$$

Notice that each matrix in (561) and (562), save for C_x^M and C_y^M, is a tensor product. For *fast operator evaluation*, they should be left in this form. Fortunately, C_x^M and C_y^M are diagonal, so they permit evaluation in only M^d operations and storage. One can also store B^M as a diagonal matrix or, as is commonly done, combine it with the C^M matrices.

2D Quadrature Example

Quadrature evaluation with the variable coefficient $c_x(r,s)$ in the inner-product (552) can be viewed as pointwise collocation of the velocity component with the quadrature weights, which are entries in the diagonal matrix B^M. As such, we can also view these pointwise velocity values as entries in a diagonal matrix C_x (or C_x^M, to indicate that the evaluation is on the nodes of the M-point GL formula). The figures below illustrate the correspondence between the GL points and the lexicographically-ordered locations of the entries in B^M and C_x.

$$\underbrace{\begin{bmatrix} \rho_1^M\rho_1^M & & & \\ & \rho_1^M\rho_2^M & & \\ & & \ddots & \\ & & & \rho_M^M\rho_M^M \end{bmatrix}}_{B^M = \hat{B}^M \otimes \hat{B}^M} \qquad \underbrace{\begin{bmatrix} c_{11} & & & \\ & c_{21} & & \\ & & \ddots & \\ & & & c_{MM} \end{bmatrix}}_{C_x^M}$$



3×3 GL grid

11.2.2 Extension to 3D

With the use of tensor-products, the extension to 3D is relatively straightforward. Our extended vector of basis coefficients takes the form

$$\underline{\bar{u}} \;=\; [u_{ijk}]^T\,,$$

where the index range is $i,j,k \in \{0,\ldots,N\}^3$ in the uniform case, or $i \in \{0,\ldots,N_x\}$, $j \in \{0,\ldots,N_y\}$, and $k \in \{0,\ldots,N_z\}$ in a more general expansion. (We will use the uniform case throughout to simplify the exposition.)

For the scalar advection term, (561) and (562) are replaced by three integral quantities,

$$\mathcal{I}_x \;=\; \underline{\bar{v}}^T \left[\left(\hat{J}^T \otimes \hat{J}^T \otimes \hat{J}^T \right) B^M C_x^M \left(\hat{J} \otimes \hat{J} \otimes \tilde{D} \right) \right] \underline{\bar{u}}, \tag{563}$$

$$\mathcal{I}_y \;=\; \underline{\bar{v}}^T \left[\left(\hat{J}^T \otimes \hat{J}^T \otimes \hat{J}^T \right) B^M C_y^M \left(\hat{J} \otimes \tilde{D} \otimes \hat{J} \right) \right] \underline{\bar{u}}, \tag{564}$$

$$\mathcal{I}_z \;=\; \underline{\bar{v}}^T \left[\left(\hat{J}^T \otimes \hat{J}^T \otimes \hat{J}^T \right) B^M C_z^M \left(\tilde{D} \otimes \hat{J} \otimes \hat{J} \right) \right] \underline{\bar{u}}. \tag{565}$$

Here, $B^M = \hat{B}^M \otimes \hat{B}^M \otimes \hat{B}^M$ and C_x, C_y, and C_z are diagonal matrices with entries corresponding to the velocity components of $\mathbf{c} = [c_x\ c_y\ c_z]^T$ evaluated at the GL points, (η_l, η_m, η_n).

Because of the contraction implied by the "dot" in $\mathbf{c} \cdot \nabla u$, we ultimately want the sum, $\mathcal{I} = \mathcal{I}_x + \mathcal{I}_y + \mathcal{I}_z$, which implies that we can realize considerable savings in evaluation of the bilinear

form by factoring out the common $\hat{J}^T$ terms, leading to

$$\mathcal{I} \;=\; \underline{\bar{v}}^T J^T B^M \left[C_x^M \left(\hat{J} \otimes \hat{J} \otimes \tilde{D} \right) + C_y^M \left(\hat{J} \otimes \tilde{D} \otimes \hat{J} \right) + C_z^M \left(\tilde{D} \otimes \hat{J} \otimes \hat{J} \right) \right] \underline{\bar{u}}. \tag{566}$$

Here, $J := \hat{J} \otimes \hat{J} \otimes \hat{J}$ is the three-dimensional interpolation matrix *to* the quadrature points. As always, we never form J because that would result in $O(N^6)$ storage and work, whereas the factored form requires only $O(N^3)$ storage (for $\underline{u}$) and $O(N^4)$ operations for the tensor contraction. We further remark that that J^T is *not* an interpolation to the nodes—it derives from the act of interpolating the test functions to the quadrature points.

A still more compact form of (566) can be realized by introducing the *gradient* operators,

$$\tilde{D}_x \;:=\; \hat{J} \otimes \hat{J} \otimes \tilde{D} \tag{567}$$

$$\tilde{D}_y \;:=\; \hat{J} \otimes \tilde{D} \otimes \hat{J} \tag{568}$$

$$\tilde{D}_z \;:=\; \tilde{D} \otimes \hat{J} \otimes \hat{J}, \tag{569}$$

which serve to interpolate the derivatives of u onto $\{\eta_k\}$. With this notation, we have

$$\mathcal{I} \;=\; \underline{\bar{v}}^T J^T B^M \left[C_x^M \tilde{D}_x \,+\, C_y^M \tilde{D}_y \,+\, C_z^M \tilde{D}_z \right] \underline{\bar{u}}. \tag{570}$$

The gradient operators can alternatively be evaluated as $D_x J$, $D_y J$, $D_z J$, where $D_x = I_M \otimes I_M \otimes D_M$, with similar expressions for D_y and D_z. Here, I_M and D_M are square matrices operating on the order M quadrature points. In this case (570) becomes

$$\mathcal{I} \;=\; \underline{\bar{v}}^T J^T B^M \left[C_x^M D_x \,+\, C_y^M D_y \,+\, C_z^M D_z \right] J \underline{\bar{u}}, \tag{571}$$

which has a more appealing symmetry because it clearly reveals the role of J in mapping u and v to the quadrature points.

11.2.3 Fast Operator Evaluation for Advection

From (561)–(562) and (571), the advection operators in $\hat{\Omega}$ are

$$C \;=\; \left(\hat{J}^T \otimes \hat{J}^T \right) B^M \left[C_x^M \left(\hat{J} \otimes \tilde{D} \right) \,+\, C_y^M \left(\tilde{D} \otimes \hat{J} \right) \right] \tag{572}$$

in 2D and

$$C = \left(\hat{J}^T \otimes \hat{J}^T \otimes \hat{J}^T \right) B^M \left[C_x^M \left(\hat{J} \otimes \hat{J} \otimes \tilde{D} \right) \,+\, C_y^M \left(\hat{J} \otimes \tilde{D} \otimes \hat{J} \right) \,+\, C_z^M \left(\tilde{D} \otimes \hat{J} \otimes \hat{J} \right) \right] \tag{573}$$

in 3D. Of course, these matrices are never formed as they are typically full because of the nonseparability of the advecting velocity fields. Fast evaluation relies on treating the diagonal matrices B^M, C_x^M, C_y^M, and C_z^M as scalar fields that have one entry for each of the M^d quadrature points, d=2 or 3. Consider the specific case of $d = 3$ and let $\underline{\tilde{b}}$, $\underline{\tilde{c}}_x$, $\underline{\tilde{c}}_y$, and $\underline{\tilde{c}}_z$ be vectors of length M^3 corresponding to quadrature points (η_i, η_j, η_k). The vector $\underline{\tilde{b}}$ will have entries

$$\tilde{b}_{ijk} \;=\; [\; \rho_0^M \rho_0^M \rho_0^M \;\; \rho_1^M \rho_0^M \rho_0^M \;\; \ldots \;\; \rho_M^M \rho_M^M \rho_M^M \;]^T.$$

The advecting field, $\mathbf{c}^M = [\; \tilde{c}_x \, \tilde{c}_y \, \tilde{c}_z \;]^T$, is evaluated at the quadrature points such that

$$[\underline{\tilde{c}}_x]_{ijk} = \tilde{c}_x(\eta_i, \eta_j, \eta_k),$$

with similar expressions for $\tilde{c}_y$ and $\tilde{c}_z$. If the advecting velocity is time-invariant or is used to advect more than one field (as in the case of Navier-Stokes or Navier-Stokes plus thermal transport), it is advantageous to pre-multiply the velocity components by $\underline{\tilde{b}}$. We define $\underline{\tilde{d}}_x := \underline{\tilde{b}} \circ \underline{\tilde{c}}_x$ as the scalar field having entries

$$[\underline{\tilde{d}}_x]_{ijk} \;\; := \;\; \tilde{b}_{ijk}[\underline{\tilde{c}}_x]_{ijk}. \tag{574}$$

Here, the "$\circ$" symbol implies pointwise multiplication of the operands, as indicated in (574). There is no summation. In matlab, this operation would be, say, `dtx=bt.*ctx`. It is the ".*" operation that implies pointwise multiplication (i.e., collocation) in matlab. An efficient matlab implementation of the matrix-vector product for (573) would be

```
% Apply dealiased advection operator in 3D: w = C*u
ux=fast_tens3d_eval(u,Jh,Jh,Dt); % ux = (Jh x Jh x Dt) u
uy=fast_tens3d_eval(u,Jh,Dt,Jh); % etc.
uz=fast_tens3d_eval(u,Dt,Jh,Jh); % etc.
ux=dx.*ux+dy.*uy+dz.*uz;         % dx = B*Cx, etc.
w =fast_tens3d_eval(ux,Jh',Jh',Jh');
```

In 2D, the fast tensor contraction can be expressed in a single line, yielding

```
% Apply dealiased advection operator in 2D: w = C*u
ux=Dt*u*Jh';              % ux = (Jh x Dt) u
uy=Jh*u*Dt';              % uy = (Dt x Jh) u
ux=dx.*ux+dy.*uy;         % dx = B*Cx, etc.
w =Jh'*ux*Jh;             % w  = J'*BM*(C.grad u)
```

In these expressions, we assume that `dx`, etc. have been pre-collocated with the fine-grid mass matrix entries, $\tilde{b}$.

Exercise. If $M = \gamma N$, what is the computational complexity (in number of flops) for (570) and (571) as a function of N and γ? For the particular case of $\gamma = 3/2$, which approach is fastest?

11.2.4 Choice of Quadrature Order, M

If $\mathbf{c}$ is a polynomial of degree N (as would be the case for the Navier-Stokes equations where $\mathbf{c} = \mathbf{u}$ is the solution in X_0^N), then the integrand for $c(v, u)$ is a polynomial of degree $3N$. Since M-point GL quadrature is exact for all integrands of degree $\leq 2M-1$ we can guarantee that the quadrature is exact if we take $2M - 1 \geq 3N$ or $M \geq (3N + 1)/2$. In spectral methods, this is known as the 3/2-rule for *dealiasing*. Dealiasing simply means that we are avoiding subsampling (i.e., aliasing) of the integrand ($\mathbf{c} \cdot \nabla u$ times the test function) by evaluating each term on a sufficiently dense point set to guarantee that we honor the integration in the variational form.

In the case of the constant-coefficient diffusion operator, $a(\cdot, \cdot)$, we have exact integration in 1D. For $d > 1$, however, integration with respect to the undifferentiated direction requires quadrature of a polynomial of degree $2N$, which is not exact with the $(N+1)$-point GLL quadrature rule. For the diffusion operator, this induces an exponentially-small error, but the operator remains symmetric positive definite and stable.

By contrast, failure to honor the integration of the advection operator can *completely change* the character of of the operator. Specifically, from (545), we have

$$C_{ij} \;:=\; c(l_i, l_j) \;=\; -c(l_j, l_i) \;=\; -C_{ji}, \tag{575}$$

or $C = -C^T$, which means that C is skew symmetric and therefore has imaginary eigenvalues. Because the mass matrix is SPD, we also have imaginary eigenvalues for the evolution matrix, $-B^{-1}C$, associated with the pure advection problem,

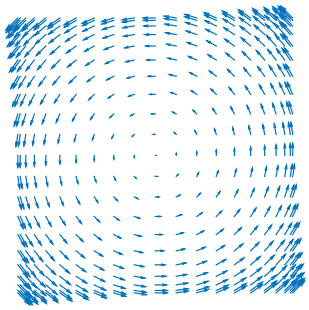
$$\frac{\partial u}{\partial t} \;=\; -\mathbf{c} \cdot \nabla u \quad\Longleftrightarrow\quad \frac{d\underline{u}}{dt} \;=\; -B^{-1}C\underline{u}, \tag{576}$$

assuming that the PDE is equipped with a suitable set of boundary conditions (e.g., Dirichlet, periodic, or vanishing flux on $\partial\Omega$). We can therefore expect, as in the continuous case, that advection will not result in energy growth, assuming that we use a suitably small step size with a timestepper whose stability region encompasses part of the imaginary axis.

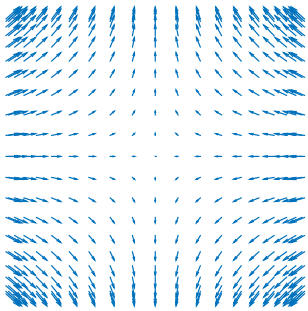
The establishment of $c(v, u) = -c(u, v)$ hinges crucially on integration by parts. If we replace $c(v, u)$ with $c_M(v, u)$, where M represents Mth-order quadrature, we cannot guarantee that $c_M(v, u) = -c_M(u, v)$ unless the integration is *exact*. If we lose skew symmetry, we cannot guarantee that the eigenvalues are imaginary. If the eigenvalues have a real part then there will invariably be both positive and negative real parts, which means that there will be eigenvalues outside of the stability region for any viable (implicit or explicit) timestepping scheme. Unless there is sufficient diffusion (i.e., the Peclet number is relatively small), instability will follow.

Dealiasing Example[a]

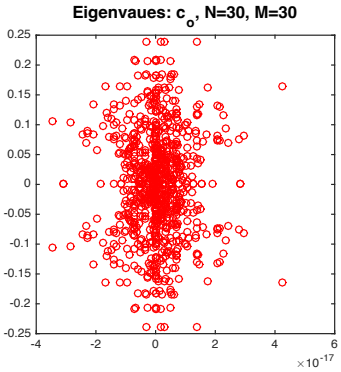
Here, we look at the eigenvalues for two velocity fields $\mathbf{c}_o = [-y, x]^T$ and $\mathbf{c}_+ = [-x, y]^T$, which satisfy $\mathbf{c} \in \mathbb{P}_1$. Both cases will yield skew-symmetric advection matrices $C = -C^T$ if we take $M = N + 1$ using either GL or GLL quadrature. It is also straightforward to show that C for the case of plane-rotation, $\mathbf{c}_o$, will be skew symmetric with $M = N$ and GLL quadrature. For $\mathbf{c}_+$, which corresponds to a straining field, C will not be skew symmetric. This case is unstable.

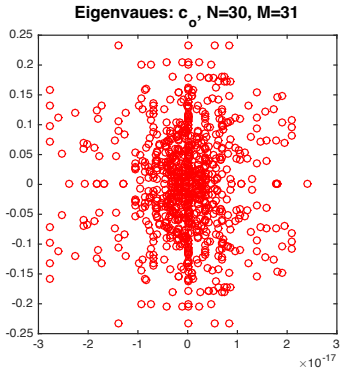


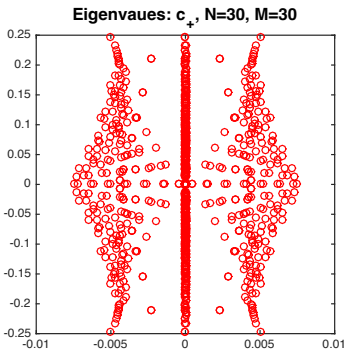
$\mathbf{c}_o = [-y, x]$

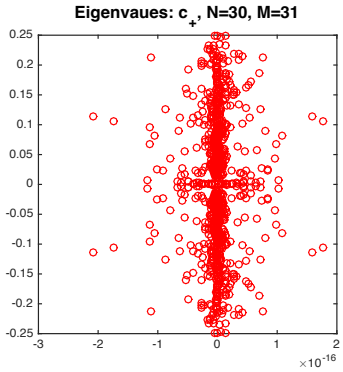


$\mathbf{c}_+ = [-x, y]$



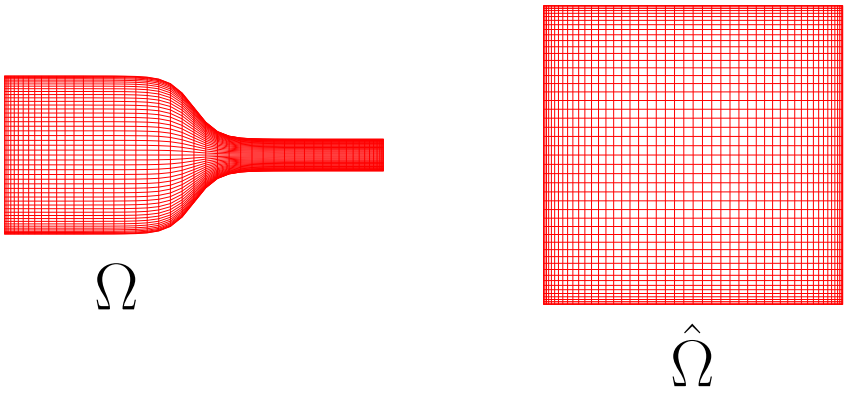






[a]This topic was explored in [Malm *et al.*,2012].

12 Deformed Geometries



If Ω is deformed, then the geometry is typically represented *isoparametrically.*[13] For example, in 2D we have

$$\mathbf{x}(r,s) \;=\; \sum_{j=0}^{N}\sum_{i=0}^{N} \mathbf{x}_{ij}\, l_i(r)\, l_j(s). \tag{577}$$

The derivative in x-y space (i.e., in Ω) is computed using the *chain rule*,

$$[\underline{w}]_{pq} \;=\; \left.\frac{\partial u}{\partial x}\right|_{r_p s_q} \;=\; \left(\frac{\partial u}{\partial r}\,\frac{\partial r}{\partial x} \;+\; \frac{\partial u}{\partial s}\,\frac{\partial s}{\partial x}\right)_{r_p s_q} \tag{578}$$

$$\;=\; (r_x)_{pq}(u_r)_{pq} \;+\; (s_x)_{pq}(u_s)_{pq}. \tag{579}$$

The precise form of the corresponding operators depends on whether we want the output on the Lagrangian interpolating nodes or on a different set of points (e.g., on quadrature points). In the general case, we have

$$\underline{w} \;=\; [r_x] D_r \underline{u} \;+\; [s_x] D_s \underline{u}, \tag{580}$$

where, assuming the same point distributions in r and s, the r-s gradient operators are $D_r = \hat{J} \otimes \tilde{D}$ and $D_s = \tilde{D} \otimes \hat{J}$. Here, $\tilde{D} := \hat{J}\hat{D}$; $\hat{J}$ is the interpolant from the basis nodal points to the quadrature points; and $\hat{D}$ is the usual square differentiation matrix on the nodal points. If we are mapping the input to the *same* Lagrangian nodal points, then $\hat{J} = \hat{I}$ and $\tilde{D} = \hat{D}$. Naturally, the metric terms must be evaluated on the output points. The metrics $[r_x]$ and $[s_x]$ are *diagonal* matrices of size M^2, assuming that we have M outputs (quadrature points) in each direction. In practice, all the metric terms are stored on the target GLL or GL points applied as pointwise collocation.

The full gradient (*vector field*) is given by,

$$\underline{\mathbf{w}} \;=\; \nabla \underline{u} \;=\; \underbrace{\begin{bmatrix} [r_x] & [s_x] \\ [r_y] & [s_y] \end{bmatrix}}_{\mathbf{R_x}} \begin{bmatrix} D_r \\ D_s \end{bmatrix} \underline{u} \tag{581}$$

$$\;=\; \mathbf{R_x} \mathbf{D}_r\, \underline{u}. \tag{582}$$

[13] Isoparamtric means that the geometry uses the same approximation basis as the solution.

Here, we use **bold font** to indicate a *vector field* or a matrix that operates on and/or produces a vector field. Matlab code that implements gradient evaluation in 2D is given below.

Gradient in Deformed Geometry

```
ur = Dh*u; us = u*Dh';
ux = ur.*rx + us.*sx;
uy = ur.*ry + us.*sy;
```

12.1 Metrics

To evaluate the gradient of a field in Ω, we require inverse metrics of the form

$$\left.\frac{\partial r}{\partial x}\right|_{r_p s_q} \tag{583}$$

From (577), we can readily compute

$$\left.\frac{\partial \mathbf{x}}{\partial r}\right|_{pq} \;=\; \sum_{ij} \mathbf{x}_{ij} \left.\frac{dl_i}{dr}\right|_p l_j(s_q) \;=\; D_r \underline{\mathbf{x}} \;=\; (\hat{I} \otimes \hat{D})\underline{\mathbf{x}} \tag{584}$$

$$\left.\frac{\partial \mathbf{x}}{\partial s}\right|_{pq} \;=\; \sum_{ij} \mathbf{x}_{ij}\, l_i(r_p) \left.\frac{dl_j}{ds}\right|_q \;=\; D_s \underline{\mathbf{x}} \;=\; (\hat{D} \otimes \hat{I})\underline{\mathbf{x}}. \tag{585}$$

Recalling the Chain Rule,

$$\frac{\partial u}{\partial x} \;=\; \frac{\partial u}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial u}{\partial s}\frac{\partial s}{\partial x}, \tag{586}$$

and successively setting $u = x$ and $u = y$ yields

$$1 \;=\; \frac{\partial x}{\partial x} \;=\; \frac{\partial x}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial x}{\partial s}\frac{\partial s}{\partial x} \tag{587}$$

$$0 \;=\; \frac{\partial y}{\partial x} \;=\; \frac{\partial y}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial y}{\partial s}\frac{\partial s}{\partial x} \tag{588}$$

$$0 \;=\; \frac{\partial x}{\partial y} \;=\; \frac{\partial x}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial x}{\partial s}\frac{\partial s}{\partial y} \tag{589}$$

$$1 \;=\; \frac{\partial y}{\partial y} \;=\; \frac{\partial y}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial y}{\partial s}\frac{\partial s}{\partial y}. \tag{590}$$

This identity must hold at *each point*, (r_p, s_q). In matrix form we have,

$$\left[\begin{array}{cc} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{array}\right]_{r_p,s_q} \left[\begin{array}{cc} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\ \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} \end{array}\right]_{r_p,s_q} = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right]. \tag{591}$$

Thus we find the inverse metrics, $\frac{\partial r_i}{\partial x_j}$, by inverting the 2×2 matrix of *computable* derivatives $\frac{\partial x_i}{\partial r_j}$, at each gridpoint (r_p, s_q).

It is instructive to derive a closed-form expression for the metrics $\frac{\partial r_i}{\partial x_j}$. Recall Cramer's rule for the inverse of a 2×2 matrix,

$$\left[\begin{array}{cc} a & b \\ c & d \end{array}\right]^{-1} \;=\; \frac{1}{\mathcal{J}}\left[\begin{array}{cc} d & -b \\ -c & a \end{array}\right], \tag{592}$$

where the Jacobian $\mathcal{J} = ad - bc$. Inserting the metric terms from (591), yields

$$\begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\ \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} \end{bmatrix} = \frac{1}{\mathcal{J}} \begin{bmatrix} \frac{\partial y}{\partial s} & -\frac{\partial x}{\partial s} \\ -\frac{\partial y}{\partial r} & \frac{\partial x}{\partial r} \end{bmatrix}, \tag{593}$$

with $\mathcal{J} = x_r y_s - x_s y_r$, which is precisely the same definition of the Jacobian (598) introduced below in the discussion of integration rules. If the geometry is a polynomial of degree G then the Jacobian will be a polynomial of degree $2G$. The metric terms such as $r_x = y_s/\mathcal{J}$ will be *rational* polynomials in r and s, but the metric terms times the Jacobian will be polynomials of degree G. For example, if Ω is an arbitrary quadilateral then x and y are linear functions of r and s, which corresponds to G=1, and the Jacobian will be a polynomial of degree at most 2.

The metric terms in the three-dimensional case have a similar form. From the chain rule,

$$\begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial r} & \frac{\partial z}{\partial s} & \frac{\partial z}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial z} \\ \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} & \frac{\partial s}{\partial z} \\ \frac{\partial t}{\partial x} & \frac{\partial t}{\partial y} & \frac{\partial t}{\partial z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{594}$$

Although a bit more tedious, Cramer's rule can also be applied to the 3D case. The (corrected!) 3×3 inverse formula is given by

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^{-1} = \frac{1}{\mathcal{J}} \begin{bmatrix} (ei - fh) & (ch - bi) & (bf - ec) \\ (fg - di) & (ai - cg) & (dc - af) \\ (dh - eg) & (bg - ah) & (ae - bd) \end{bmatrix}, \tag{595}$$

with $\mathcal{J} = a(ei - fh) - b(di - fg) + c(dh - eg)$. As in the 2D case, we proceed by inserting the metric terms (594) into (595),

$$\begin{bmatrix} r_x & r_y & r_z \\ s_x & s_y & s_z \\ t_x & t_y & t_z \end{bmatrix} = \frac{1}{\mathcal{J}} \begin{bmatrix} (y_s z_t - y_t z_s) & (x_t z_s - x_s z_t) & (x_s y_t - y_s x_t) \\ (y_t z_r - y_r z_t) & (x_r z_t - x_t z_r) & (y_r x_t - x_r y_t) \\ (y_r z_s - y_s z_r) & (x_s z_r - x_r z_s) & (x_r y_s - x_s y_r) \end{bmatrix}. \tag{596}$$

Here, the Jacobian is the determinant of $\left[\frac{\partial x_i}{\partial r_j}\right]$. As in the 2D case, the metric terms $\frac{\partial r_i}{\partial x_j}$ are rational polynomials in (r, s, t), but the metrics times the Jacobians are polynomials of degree $2G$. The Jacobian is a polynomial of degree $3G$.

12.2 Integration in Ω

In addition to the derivatives, we evaluate all integrals over Ω as corresponding integrals over $\hat{\Omega}$,

$$\int_{\Omega} f \, dA = \int_{\hat{\Omega}} f \, \mathcal{J} \, dr \, ds. \tag{597}$$

Here, $\mathcal{J}(r, s)$ is the *Jacobian*. It is a scalar field corresponding to the amount of area in Ω that is associated with a unit area in $\hat{\Omega}$. An example is shown in the figure below.

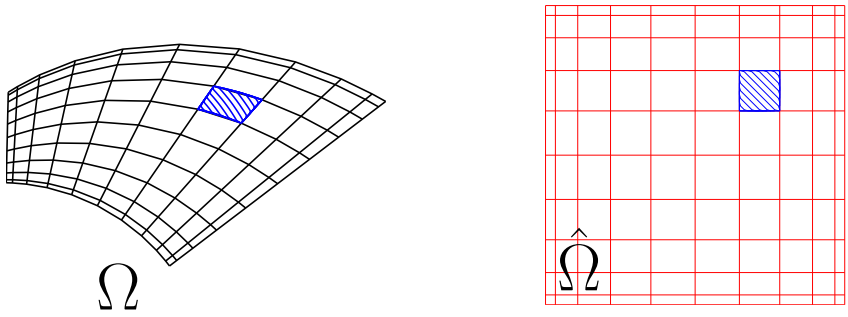


Figure 14: Example of mapped domain (left) from reference element, $\hat{\Omega} = [-1, 1]^2$ (right).

As the cells become infitesimally small, the blue area is given by the *cross product*,

$$\begin{aligned} dA &= \frac{\partial \mathbf{x}}{\partial r} dr \times \frac{\partial \mathbf{x}}{\partial s} ds = \mathcal{J}\, dr\, ds \\ &= \left[\frac{\partial x}{\partial r}\frac{\partial y}{\partial s} - \frac{\partial x}{\partial s}\frac{\partial y}{\partial r} \right] dr\, ds \\ &= \left| \begin{array}{cc} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{array} \right| dr\, ds. \end{aligned} \tag{598}$$

That is, $\mathcal{J}_{pq}$ is the determinant of the 2×2 metric tensor at each quadrature point (r_p, s_q). If we accept some (exponentially small) quadrature error, we can have a diagonal mass matrix by evaluating $\mathcal{J}$ at the GLL points,

$$B = \mathrm{diag}(\mathcal{J}_{pq}\rho_p\rho_q) = \mathcal{J}(\hat{B} \otimes \hat{B}), \tag{599}$$

Here, we treat $\mathcal{J}$ as a diagonal matrix, following our standard practice with GLL quadrature. For M-point quadrature rules, (599) will be replaced by a full mass matrix,

$$\tilde{B} = J^T \left[\mathrm{diag}(\mathcal{J}^M_{pq}\rho^M_p\rho^M_q) \right] J = (\hat{J}^T \otimes \hat{J}^T)\mathcal{J}^M(B^M \otimes B^M)(\hat{J} \otimes \hat{J}). \tag{600}$$

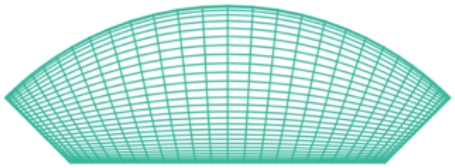
Here, B^M=diag(ρ^M_j) and $\hat{J}$ is the interpolation matrix from the N+1 GLL points to the M quadrature points, and $\mathcal{J}^M$ is the Jacobian interpolated to the $M \times M$ arrary of quadrature points.

In 3D we have a similar form for the Jacobian as the volumetric multiplier of $dr\, ds\, dt$. Here, we take the dot product of the variation in t with the cross product introduced in (598),

$$\mathcal{J} = \frac{\partial \mathbf{x}}{\partial t} \cdot \left(\frac{\partial \mathbf{x}}{\partial r} \times \frac{\partial \mathbf{x}}{\partial s} \right). \tag{601}$$

By symmetry, we can obtain several equivalent representations through permutations of r, s, and t in (601).

Mapped-Domain Example
Use (599) to compute the area of the domains shown. The one on the right represents one-quarter of a circular disk of radius $r = 1$ that has a concentric 1×1 square hole. The one on the left is the same domain save for the circular boundary. **Plot** the relative error as a function of N for N=2, 3,. . . ,12.



12.3 Generation of $\mathbf{x}_{ij}$

A requirement of the isoparametric mapping (577) is to have a set of nodal points $\mathbf{x}_{ij}$ that constitute the basis coefficients for the geometric deformation. Generally, one only has boundary information describing the geometry and there is no *a priori* point distribution for nodes interior to Ω. While there is some leeway in how the points are distributed, it's generally best if the distribution is smooth and representable as a low-order polynomial in (r, s) whenever possible. In essence, we require a *blending function* that smoothly propagates boundary data into the domain interior. The *Gordon Hall mapping* [Gordon & Hall '76] (a.k.a. transfinite interpolation) is a relatively simple and effective approach to meeting these goals.

To illustrate Gordon-Hall (GH), consider a 2D case where we know the nodal point distribution on the domain edges, $\{\mathbf{x}_{0,j},\, \mathbf{x}_{N,j},\, \mathbf{x}_{i,0},\, \mathbf{x}_{i,N}\}$, for $i, j \in \{0, \ldots, N\}$. We begin by generating a bilinear map that involves only the vertex nodes. Let l_i denote the usual Nth-order Lagrange cardinal functions on (say) the GLL points and l_i^1, $i = 0, 1$, denote the 1st-order Lagrange cardinal points on $[-1, 1]$. The vertex-only map is given by

$$\hat{\mathbf{x}}(r, s) \;=\; \mathbf{x}_{00}\, l_0^1(r) l_0^1(s) \,+\, \mathbf{x}_{N0}\, l_1^1(r) l_0^1(s) \,+\, \mathbf{x}_{0N}\, l_0^1(r) l_1^1(s) \,+\, \mathbf{x}_{NN}\, l_1^1(r) l_1^1(s). \tag{602}$$

Note that $\hat{\mathbf{x}} \in \mathbb{P}_1(r) \times \mathbb{P}_1(s)$ is *bilinear*. If Ω has straight sides then we use (602) to evaluate $\mathbf{x}_{ij} := \hat{\mathbf{x}}(\xi_i, \xi_j)$ and the map is complete. We have a geometry that is precisely represented by a polynomial of degree $G = 1$.

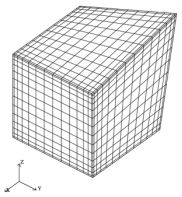
If Ω has curved sides, we define a new edge function, $\delta(r, s) = \mathbf{x} - \hat{\mathbf{x}}$, for which $\{\delta_{0,j},\, \delta_{N,j},\, \delta_{i,0},\, \delta_{i,N}\}$, are readily computable. We next extend each of these edge perturbations into the domain interior, (that is, into the interior of $\hat{\Omega}$). The GH blending formula is,

$$\begin{aligned} \mathbf{x}(r, s) \;&=\; \hat{\mathbf{x}}(r, s) \\ &+\; \sum_{i=0}^{N} \left[\delta_{i0} l_i(r) l_0^1(s) \,+\, \delta_{iN} l_i(r) l_1^1(s)\right] \\ &+\; \sum_{j=0}^{N} \left[\delta_{0j} l_0^1(r) l_j(s) \,+\, \delta_{Nj} l_1^1(r) l_j(s)\right]. \end{aligned} \tag{603}$$

We see that (603) yields a tensor-product polynomial that is a combination of linear polynomials in one direction with Nth-order polynomials in the other. The Nth-order polynomials capture the fluctuation of δ along an edge of $\hat{\Omega}$. The linear polynomials map this perturbation to zero as r or s approaches the opposite edge of $\hat{\Omega}$.

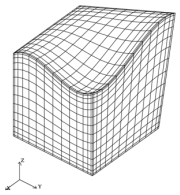
To apply GH in 3D we again use a vertex blending function, but we now need twelve edge-blending functions and six face-blending functions. As in the 2D case, we assume that the boundary

$$\mathbf{v}_{ijk} = \sum_{\hat{\imath}\hat{\jmath}\hat{k}} l^1_{\hat{\imath}}(\xi^N_i) l^1_{\hat{\jmath}}(\xi^N_j) l^1_{\hat{k}}(\xi^N_k) \mathbf{x}_{\hat{\imath}N,\hat{\jmath}N,\hat{k}N}$$



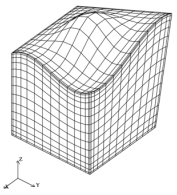
$\mathbf{v}$

$$\begin{aligned}\mathbf{e}_{ijk} = \mathbf{v}_{ijk} &+ \sum_{\hat{\jmath}\hat{k}} l^1_{\hat{\jmath}}(\xi^N_j) l^1_{\hat{k}}(\xi^N_k)(\mathbf{x}_{i,\hat{\jmath}N,\hat{k}N} - \mathbf{v}_{i,\hat{\jmath}N,\hat{k}N}) \\ &+ \sum_{\hat{\imath}\hat{k}} l^1_{\hat{\imath}}(\xi^N_i) l^1_{\hat{k}}(\xi^N_k)(\mathbf{x}_{\hat{\imath}N,j,\hat{k}N} - \mathbf{v}_{\hat{\imath}N,j,\hat{k}N}) \\ &+ \sum_{\hat{\imath}\,\hat{\jmath}} l^1_{\hat{\imath}}(\xi^N_i) l^1_{\hat{\jmath}}(\xi^N_j)(\mathbf{x}_{\hat{\imath}N,\hat{\jmath}N,k} - \mathbf{v}_{\hat{\imath}N,\hat{\jmath}N,k})\end{aligned}$$



$\mathbf{e}$

$$\begin{aligned}\mathbf{f}_{ijk} = \mathbf{e}_{ijk} &+ \sum_{\hat{\imath}} l^1_{\hat{\imath}}(\xi^N_i)(\mathbf{x}_{\hat{\imath}N,j,k} - \mathbf{e}_{\hat{\imath}N,j,k}) \\ &+ \sum_{\hat{\jmath}} l^1_{\hat{\jmath}}(\xi^N_j)(\mathbf{x}_{i,\hat{\jmath}N,k} - \mathbf{e}_{i,\hat{\jmath}N,k}) \\ &+ \sum_{\hat{k}} l^1_{\hat{k}}(\xi^N_k)(\mathbf{x}_{i,j,\hat{k}N} - \mathbf{e}_{i,j,\hat{k}N})\end{aligned}$$



$\mathbf{f}$

Figure 15: Application of the Gordon-Hall algorithm in $\mathbb{R}^3$. Subscripts i, j, and k range from 0 to N. Summations involving $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ range from 0 to 1, accounting for contributions from opposing faces in each of the three coordinate directions. The outputs of each of the three phases, vertex-, edge-, and face-extension, are indicated be $\mathbf{v}$, $\mathbf{e}$, and $\mathbf{f}$, respectively. The final geometry is given by $\mathbf{x} := \mathbf{f}$ and is defined at all interior and surface points.

points are known. The vertex map is

$$\begin{aligned}\hat{\mathbf{x}}(r,s,t) = &\ \mathbf{x}_{000}\ l^1_0(r)\,l^1_0(s)\,l^1_0(t) + \mathbf{x}_{N00}\ l^1_1(r)\,l^1_0(s)\,l^1_0(t) \\ &+ \mathbf{x}_{0N0}\ l^1_0(r)\,l^1_1(s)\,l^1_0(t) + \mathbf{x}_{NN0}\ l^1_1(r)\,l^1_1(s)\,l^1_0(t) \\ &+ \mathbf{x}_{000}\ l^1_0(r)\,l^1_0(s)\,l^1_1(t) + \mathbf{x}_{N00}\ l^1_1(r)\,l^1_0(s)\,l^1_1(t) \\ &+ \mathbf{x}_{0N0}\ l^1_0(r)\,l^1_1(s)\,l^1_1(t) + \mathbf{x}_{NN0}\ l^1_1(r)\,l^1_1(s)\,l^1_1(t)\end{aligned} . \tag{604}$$

In this case, $\hat{\mathbf{x}}$ is *trilinear*. If we define $\underline{\mathbf{x}}^1 = [\mathbf{x}_{000}\,\mathbf{x}_{N00}\ldots\mathbf{x}_{NNN}]^T$, and $J^1 = \hat{J}^1 \otimes \hat{J}^1 \otimes \hat{J}^1$, where $\hat{J}^1$ as the $(N+1)\times 2$ interpolation matrix from the vertices ± 1 to the GLL points, then we can evaluate (604) at the GLL points with the simple expression $\underline{\mathbf{x}} = J^1\underline{\mathbf{x}}^1 = (\hat{J}^1 \otimes \hat{J}^1 \otimes \hat{J}^1)\underline{\mathbf{x}}^1$, which is simpler and less error-prone than trying to evaluate the polynomials in (604) directly. The full GH mapping for the 3D case is illustrated in Fig. 15.

12.4 Bilinear Form in Deformed Coordinates

Equipped with the tools required for integration and differentiation, we can now turn to the problem of solving PDEs in deformed geometries. For the Poisson problem, we have the usual energy inner product,

$$a(v,u) = \int_\Omega \nabla v \cdot \nabla u \, dA \tag{605}$$

$$= [\mathbf{D}\underline{v}]^T\ (\mathbf{B})\ [\mathbf{D}\underline{u}] \tag{606}$$

$$= [\mathbf{R}_\mathbf{x}\mathbf{D}_r\underline{v}]^T\ (\mathbf{B})\ [\mathbf{R}_\mathbf{x}\mathbf{D}_r\underline{u}]\,. \tag{607}$$

If we are using an M-point quadrature rule in each direction, the expanded form of (607) in two dimensions is

$$a(v,u) \;=\; \underline{v}^T \begin{pmatrix} D_r \\ D_s \end{pmatrix}^T \underbrace{\begin{bmatrix} [r_x] & [s_x] \\ [r_y] & [s_y] \end{bmatrix}^T \begin{pmatrix} B^M & 0 \\ 0 & B^M \end{pmatrix} \begin{bmatrix} [r_x] & [s_x] \\ [r_y] & [s_y] \end{bmatrix}}_{\textit{diagonal, evaluated on quadrature points}} \begin{pmatrix} D_r \\ D_s \end{pmatrix} \underline{u}. \tag{608}$$

$$= \; \underline{v}^T \mathbf{D}^T \mathbf{G} \mathbf{D} \underline{u} \;=\; \underline{v}^T \bar{A} \underline{u}. \tag{609}$$

Here, $B^M = \mathcal{J}^M(\hat{B}^M \otimes \hat{B}^M)$ is the diagonal mass matrix for the deformed geometry, which includes the Jacobian, $\mathcal{J}^M$, evaluated on M^d GL quadrature points and the tensor-product of the 1D GL mass matrices, $\hat{B}^M$. In d space dimensions, we define the *geometric factors*, G_{ij}, having the form

$$G_{ij} \;:=\; B^M \left(\sum_{k=1}^{d} \left[\frac{\partial r_i}{\partial x_k} \right] \left[\frac{\partial r_j}{\partial x_k} \right] \right) \quad i,j \in \{1,\ldots,d\}^2. \tag{610}$$

Here, the bracketed terms are evaluated on the quadrature points. For example, with $d = 2$ we have

$$\mathbf{G} \;=\; \begin{bmatrix} G_{rr} & G_{rs} \\ G_{sr} & G_{ss} \end{bmatrix} \;=\; \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}. \tag{611}$$

$$\tag{612}$$

Note: $G_{ij} = G_{ji}$. *Each* G_{ij} is a *diagonal matrix* with M^d entries.

Note that $\underline{u}$ is known on $(N+1)^d$ points, while the interior terms are known on M^d quadrature points. The key here is that the interpolation from the *nodal points* to the *quadrature points* is embedded in the derivative matrices. For example, in 3D, we would have

$$D_r \;=\; (\hat{J} \otimes \hat{J} \otimes \hat{J})(I \otimes I \otimes \hat{D}) \tag{613}$$

$$D_s \;=\; (\hat{J} \otimes \hat{J} \otimes \hat{J})(I \otimes \hat{D} \otimes I) \tag{614}$$

$$D_t \;=\; (\hat{J} \otimes \hat{J} \otimes \hat{J})(\hat{D} \otimes I \otimes I), \tag{615}$$

where $\hat{J}$ is the GLL to M-point GL interpolation operator and $\hat{D}$ is our standard derivative operator on the $(N+1)$ GLL nodal points. Invariably, the most efficient quadrature choice is to use $M = (N+1)$ and to simply evaluate G_{ij} and the metrics at the GLL nodal points. This approach dispenses with the interpolation, which is expensive, and yields the following derivative operators,

$$D_r \;=\; (I \otimes I \otimes \hat{D}) \tag{616}$$

$$D_s \;=\; (I \otimes \hat{D} \otimes I) \tag{617}$$

$$D_t \;=\; (\hat{D} \otimes I \otimes I). \tag{618}$$

A-Matrix Complexity Exercises.

1. Suppose $N = 7$ and $M = 10$. Estimate the operation count to apply the operator implied in (609) in 2D and (most importantly) in 3D. Note–*we use* (609), *not* (608).

2. Suppose we use (N+1)-point GLL quadrature instead of M-point GL quadrature for (609). Estimate the operation count in 2D and in 3D.

3. To leading-order, what are the number of memory references in the 3D cases, including the cost of fetching $\underline{u}$ from memory?

4. Suppose $\Omega = \hat{\Omega}$. Show that $G_{ij} = 0$ for $i \neq j$.

5. Suppose that the mapping from $\hat{\Omega}$ to Ω is

$$x = \alpha r + \beta s \tag{619}$$
$$y = -\beta r + \alpha s, \tag{620}$$

where $\alpha = \cos\theta$ and $\beta = \sin\theta$ for some arbitrary angle θ. Show that off diagonal terms, G_{ij} are zero for this rotated-geometry case, as they must be. (Why must this be the case?)

6. Can the FDM be used for the preceding example?

7. When using nodal-based quadrature for the (forward) *matrix-vector* product, $\underline{w} = A\underline{u}$, it is actually possible to cut the leading-order work term in half for both 2D and 3D in the case where the geometry is an affine map (i.e., a simple stretching/translation in each direction).
Consider the the 2D case with $\Omega = [0 : L_x] \times [0 : L_y]$. What is $\mathcal{J}$ in this case? Provide an expression similar to (608) or (609) that yields half of the leading order work as (609) when evaluating $\underline{w} = A\underline{u}$.

1D Projection Exercise (Review).

1. Use L^2-projection of $f(x) = \sin 2k\pi x$ onto $\mathbb{P}_N[0 : 1]$ to show that, asympotically, the number of points per wavelength,

$$\text{PPW} := N/k, \tag{621}$$

required to resolve k waves on the unit interval is PPW $\sim \pi$ as $k \longrightarrow \infty$.

To do this, plot maximum pointwise error on a semilogy scale as a function of N/k as $k \longrightarrow \infty$. (There should be a family of graphs, say, for $k = 1, 2, 4, 8, \ldots$)

At what value of N/k do you observe a cross-over point?

Explain how you computed your L^2 projection.

2. For N even, what is the asymptotic value of $N\rho_{\frac{N}{2}}$, where ρ_j is the the jth of $N + 1$ GLL quadrature points on [-1,1]?

12.5 Solving Poisson in Deformed Coordinates

A priori, the discrete energy inner product will be symmetric positive definite for any $u \in X_0^N$ and thus we can expect that

$$A \;=\; R\bar{A}R^T = R\left[\mathbf{D}^T\mathbf{G}\mathbf{D}\right]R^T \tag{622}$$

will be SPD even in the case of deformed geometries. Unfortunately, the deformed case is generally *not* separable if off-diagonal entries G_{ij}, $i \neq j$, are nonzero. However, it is possible to use a separable approximation to A as a preconditioner. Since both the operator and its preconditioner are SPD, we can use preconditioned conjugate gradient to solve this problem. Generally, this approach will lead to iteration counts that are bounded independent of N.

13 Fourier Bases

We consider now the use of Fourier bases to solve boundary value problems in d space dimensions. Because of their limited applicability in general domains, these will generally be combined with polynomial bases in one or more space dimensions, depending on the boundary conditions. Our interest in Fourier bases will thus be confined to problems that are periodic in one or more dimensions. We further assume that the domain is either translationally invariant or smoothly varying in the periodic direction. Despite their limitations, Fourier bases offer some significant advantages over other spectral bases, including:

- $O(N \log N)$ operator application instead of $O(N^2)$.
- No numerical dispersion—all resolved waves are propagated at the correct speed.
- Uniform point distribution and CFL $\sim O(N^{-1})$ instead of $O(N^{-2})$.

For these reasons, they are often used for targeted applications.

To illustrate the application of Fourier spectral methods, we consider the 1D Burgers equation,

$$\frac{\partial \tilde{u}}{\partial t} \;+\; \tilde{u}\frac{\partial \tilde{u}}{\partial x} \;\;=\;\; \nu \frac{\partial^2 \tilde{u}}{\partial x^2} \;+\; q \tag{623}$$

subject to periodic boundary conditions, $\tilde{u}(0,t) = \tilde{u}(2\pi,t)$ and initial condition $u(x,t=0) = u^0(x)$. We have included $q(x,t)$ to illustrate how data is incorporated into the solution process, but it is typically zero for most Burgers model problems.

We follow the spectral-Galerkin approach to solving this problem by seeking a trial solution $u(x,t) \in X_p^N$, where $X_p^N = \{\phi_1, \phi_2, \ldots, \phi_N\}$ is the space of 2π-periodic functions with $\phi_k(0) = \phi_k(2\pi)$. The solution will be written as

$$u(x,t) \;\;=\;\; \sum_{l=1}^{N} \phi_l(x)\, \hat{u}_l(t), \tag{624}$$

As usual, we require the residual of (623) to be orthogonal to X_p^N. The specific problem statement reads, *Find $u(x,t) \in X_p^N$ such that for all $v \in X_p^N$,*

$$\left(v, \frac{\partial u}{\partial t}\right) \;+\; \left(v, c\frac{\partial u}{\partial x}\right) \;\;=\;\; \nu\left(v, \frac{\partial^2 u}{\partial x^2}\right) \;+\; \Big(v, f\Big), \tag{625}$$

where $(f,g) := \int_\Omega fg\, dx$ is the standard L^2 inner product on $\Omega = [0, 2\pi]$.

In (625) we recognize that the advecting field $c(x,t)$ is actually the solution, $c := u$. We have temporarily relabled it for clarity. We also remark that it is common to express the nonlinear term in conservation form

$$u\frac{\partial u}{\partial x} \;\;=\;\; \frac{1}{2}\frac{\partial u^2}{\partial x} \tag{626}$$

While mathematically equivalent, differences can result when one approximates the derivatives in (626) using either finite difference or finite volume methods. In the WRT, however, the approximation has already been imposed in the choice of the finite-dimensional trial/test space, X_p^N. No

further approximations are required, and either form in (626) will be equivalent under the assumption that the derivatives and integrals involving the ϕ_ks are computed exactly. Differences may arise if shortcuts (e.g., quadrature) are implemented without due care. We will, however, strive to avoid these differences as we will use fully dealiased quadrature for the advection operator in order to ensure stability of the overall system.

Implemenation of (625) requires the evaluation of four integrals. Expanding the test function as $v(x) = \sum_k \phi_k(x)\hat{v}_k$ leads to

$$\left(v, \frac{\partial u}{\partial t}\right) = \sum_{k=1}^{N}\sum_{l=1}^{N} \hat{v}_k \left(\phi_k, \phi_l\right) \frac{d\hat{u}_l}{dt}, \tag{627}$$

$$\left(v, c\frac{\partial u}{\partial x}\right) = \sum_{k=1}^{N}\sum_{l=1}^{N} \hat{v}_k \left(\phi_k, c\frac{d\phi_l}{dx}\right) \hat{u}_l \tag{628}$$

$$\left(v, c\frac{\partial^2 u}{\partial x^2}\right) = \sum_{k=1}^{N}\sum_{l=1}^{N} \hat{v}_k \left(\phi_k, \frac{d^2\phi_l}{dx^2}\right) \hat{u}_l \tag{629}$$

$$\left(v, q\right) = \sum_{k=1}^{N} \hat{v}_k \left(\phi_k, q\right) \approx \sum_{k=1}^{N}\sum_{l=1}^{N} \hat{v}_k \left(\phi_k, \phi_l\right) \hat{q}_l. \tag{630}$$

We note that, for sufficiently large N, the approximation in the last integral will be very good if f is smooth and 2π-periodic in x because the convergence will be exponential in this case. Assuming that f is independent of u (i.e., that it is just data), then approximating (630) by expressing f as an element of X_p^N will have no adverse impact on stability. (If f is not 2π-periodic, then it is not smooth in the context of this analysis.)

Denoting $\underline{\hat{v}} = [\hat{v}_1 \dots \hat{v}_N]^T$, $\underline{\hat{u}} = [\hat{u}_1 \dots \hat{u}_N]^T$, $\underline{\hat{q}} = [\hat{q}_1 \dots \hat{q}_N]^T$, and

$$A_{kl} = \left(\frac{d\phi_k}{dx}, \frac{d\phi_l}{dx}\right) \tag{631}$$

$$B_{kl} = \left(\phi_k, \phi_l\right) \tag{632}$$

$$C_{kl} = \left(\phi_k, c\frac{d\phi_l}{dx}\right), \tag{633}$$

we can express (625) as an equivalent linear algebra statement, *Find* $\underline{\hat{u}}(t)$ *such that for all* $\underline{\hat{v}} \in \mathbb{R}^N$,

$$\underline{\hat{v}}^T B \frac{d\underline{\hat{u}}}{dt} = -\underline{\hat{v}}^T \left(A + C\right) \underline{\hat{u}} + \underline{\hat{v}}^T B \underline{\hat{q}}, \tag{634}$$

or, equivalently,

$$B \frac{d\underline{\hat{u}}}{dt} = -\left(A + C\right) \underline{\hat{u}} + B\underline{\hat{q}}. \tag{635}$$

In (631) we defined A to be the negative of the integral, so that A itself will be positive-definite in keeping with our standard notation.

13.1 Choice of Bases

The preceding derivation of the WRT for Burgers equation was relatively generic. To make further progress towards a functional code we need to make specific choices for the ϕ_ks.

One common option is to use the complex exponentials. With this approach the numerical solution is represented by one of the truncated series,

$$u(x,t) \;=\; \sum_{k=-N/2}^{N/2-1} e^{ikx}\hat{u}_k(t) \qquad\qquad N \text{ even}, \tag{636}$$

or

$$u(x,t) \;=\; \sum_{k=\frac{1-N}{2}}^{\frac{N-1}{2}} e^{ikx}\hat{u}_k(t) \qquad\qquad N \text{ odd}. \tag{637}$$

The expressions (636)–(637) involve complex basis functions and complex basis coefficients. If $u(x,t) \in \mathbb{R}$, we must have the conjugacy condition,

$$\hat{u}_k \;=\; \hat{u}^*_{-k}, \tag{638}$$

which means that working with N complex numbers in this case leads to a duplication of effort. Moreover, this choice requires that all operations be implemented in complex arithmetic, which is generally not as fast as using highly tuned routines available to support real arithmetic.

An alternative to complex bases is to work directly with sines and cosines. Specifically, consider an expansion of the form

$$u(x,t) \;=\; \sum_{k=1}^{N} \phi_k(x)\hat{u}_k(t), \tag{639}$$

with (real-valued) bases,

$$\begin{aligned} X_p^N \;&=\; \text{span}\{\phi_1,\, \phi_2,\, \ldots\, \phi_N\} \\ &=\; \text{span}\{1,\, \cos x,\, \sin x,\, \cos 2x,\, \sin 2x,\, \ldots\, \cos \tfrac{N-1}{2}x,\, \sin \tfrac{N-1}{2}x\}. \end{aligned} \tag{640}$$

In this case, if N is even, we drop the last sine function in the basis set and replace the last cosine function by $\cos \frac{N}{2}x$. Somewhat curiously, if N=4, we have a set of four basis functions,

$$X_p^N \;=\; \text{span}\{1,\, \cos x,\, \sin x,\, \cos 2x\}, \tag{641}$$

which exhibits a preponderence of cosines corresponding to $k = 0$, 1, and 2, and only one sine function. This apparent asymmetry stems from the facts that $\sin kx \equiv 0$ for $k = 0$ and that $\sin \frac{N}{2}x$ vanishes on the uniform point distribution $x_j = jh$ with $h = 2\pi/N$, $j = 0, \ldots, N-1$. In the following discussion we will assume that N is odd. The case of N even, however, presents no particular difficulty.

The advantage of (639)–(640) is that the functions and basis coefficients are real-valued, so there is no need for complex arithmetic. A slight disadvantage is that the odd-numbered derivative matrices will be bidiagonal rather than diagonal. The even-numbered ones, however, will be diagonal and these are the ones that we generally care about when solving elliptic or parabolic PDEs.

13.2 Differentiation of the Sine Bases

To implement our PDE solver we will need to be able to differentiate functions in the trial space. Let $u \in X_p^N$ and

$$w \;=\; \sum_{k=1}^{N} \phi_k(x)\hat{w}_k \;:=\; \frac{\partial u}{\partial x}. \tag{642}$$

We'd like to identify the coefficients for $w(x) \in X_p^N$. With the cosine/sine basis and N odd we have,

$$u(x) \;=\; \hat{u}_1 \;+\; \sum_{k=1}^{\frac{N-1}{2}} \Big(\hat{u}_{2k} \cos kx \;+\; \hat{u}_{2k+1} \sin kx \Big) \tag{643}$$

$$\frac{du}{dx} \;=\; 0 \;+\; \sum_{k=1}^{\frac{N-1}{2}} \Big(-\hat{u}_{2k} k \sin kx \;+\; \hat{u}_{2k+1} k \cos kx \Big) \tag{644}$$

$$\;=\; 0 \;+\; \sum_{k=1}^{\frac{N-1}{2}} \Big(\hat{u}_{2k+1} k \cos kx \;-\; \hat{u}_{2k} k \sin kx \Big)$$

$$\;=\; 0 \;+\; \sum_{k=1}^{\frac{N-1}{2}} \Big(\hat{w}_{2k} \cos kx \;+\; \hat{w}_{2k+1} \sin kx \Big).$$

Consequently,

$$\hat{w}_1 \;=\; 0, \;\; \hat{w}_{2k} \;=\; k \hat{u}_{2k+1}, \;\; \hat{w}_{2k+1} \;=\; -k \hat{u}_{2k}. \tag{645}$$

In matrix form,

$$\underline{\hat{w}} \;=\; \left[\begin{array}{c|ccccccc} 0 & & & & & & & \\ \hline & 0 & 1 & & & & & \\ & -1 & 0 & & & & & \\ & & & 0 & 2 & & & \\ & & & -2 & 0 & & & \\ & & & & & \ddots & & \\ & & & & & & 0 & \frac{N-1}{2} \\ & & & & & & \frac{1-N}{2} & 0 \end{array}\right] \left(\begin{array}{c} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \\ \vdots \\ \\ \vdots \\ \hat{u}_N \end{array} \right) \;=\; \hat{D}_F \, \underline{\hat{u}}. \tag{646}$$

Clearly, the cost of differentiation in the cosine/sine basis is linear in N. For the case of N even, we remark that $\frac{du}{dx} \notin X_p^N$ because the trailing cosine term becomes a multiple of $\sin \frac{N}{2} x$, which is not in the space. This typically presents no difficulty—we simply set the last coefficient to zero. For N odd, we can write our Fourier differentiation matrix in the following compact form

$$\hat{D}_F \;=\; \left[\begin{array}{c|c} 0 & \\ \hline & \left(\begin{array}{c} \\ K \otimes S \\ \\ \end{array} \right) \end{array}\right], \tag{647}$$

with K=diag(1, 2,. . . , $\frac{N-1}{2}$) and

$$S \;=\; \left[\begin{array}{cc} 0 & 1 \\ -1 & 0 \end{array} \right]. \tag{648}$$

For the second derivative, we have $z = \frac{d^2u}{dx} \in X_p^N$ and we have a diagonal second-derivative matrix, $\hat{\hat{D}}_F$=diag(0 -1 -1 -2^2 -2^2 . . . -$\big(\frac{N-1}{2}\big)^2$). If N is even, we replace the last entry with $-\big(\frac{N}{2}\big)^2$. For N odd, we have $\hat{\hat{D}}_F = \hat{D}_F^2$. From (647) we see that the lower right block entry is $(K \otimes S)(K \otimes S) = K^2 \otimes S^2 = -K^2$. For N even, $\hat{D}_F^2$ is identical to $\hat{\hat{D}}_F$ save that the last diagonal entry is 0. It is not uncommon to use $\hat{D}_F^2$ for the second derivative.

13.3 Integration of the Sine Bases

The functions $\{\cos kx,\, \sin kx\}$, k=0, 1, ..., form an orthogonal basis on Ω. Specifically,

$$\Big(\phi_i, \phi_j\Big) = \tilde{\rho}_j\, \delta_{ij}, \quad \text{with} \quad \tilde{\rho}_j \;:=\; \begin{cases} 2\pi, & j = 1 \\ \pi, & j > 1 \end{cases} . \tag{649}$$

Because of the FFT, we will also want to consider the discrete inner product based on the N-point trapezoidal rule,

$$\Big(\phi_i, \phi_j\Big)_N := \sum_{l=1}^{N} h\phi_i(x_l)\phi_j(x_l) \;=\; \hat{\rho}_j\, \delta_{ij}, \tag{650}$$

where $h = \frac{2\pi}{N}$. Here, for $\phi_i,\, \phi_j \in X_p^N$, we have

$$\hat{\rho}_j \;=\; \begin{cases} 2\pi, & j = 1 \\ \pi, & 1 < j < N \\ \pi, & j = N \text{ if } N \text{ is odd}, \\ 2\pi, & j = N \text{ if } N \text{ is even}. \end{cases} \tag{651}$$

If $i + j > N$ there will be *aliasing* and the orthogonality (651) will not hold in general.

Remark. *For periodic functions, the N-point trapezoidal rule is a Gauss rule with quadrature weight h.* It is exact for all (ϕ_i, ϕ_j) pairs in X_p^N if N is odd, and is wrong (by a factor of 2) for (ϕ_N, ϕ_N) if N is even. It exhibits exponential convergence for smooth periodic functions because the Fourier basis coefficients decay exponentially fast in this case. (Recall (160).)

Useful Product Identities

The following identities show that products in (650) become elements of X^{2N}, which are still sines and cosines that integrate to zero unless the arguments are identical.

$$\sin\alpha\cos\beta \;=\; \frac{\sin(\alpha+\beta) \,+\, \sin(\alpha-\beta)}{2} \tag{652}$$

$$\cos\alpha\cos\beta \;=\; \frac{\cos(\alpha+\beta) \,+\, \cos(\alpha-\beta)}{2} \tag{653}$$

$$\sin\alpha\sin\beta \;=\; \frac{\cos(\alpha-\beta) \,-\, \cos(\alpha-\beta)}{2} \tag{654}$$

13.4 Interpolation of the Sine Bases and FFTs

For multiple reasons, we often want to evaluate (639) at a discrete set of points x_j. For example, one may want to plot data or, more frequently within a code, one may want to evaluate an integral using the trapezoidal rule. As we shall see, a significant advantage of the Fourier bases is that interpolation to N uniformly distributed points can be realized in $\approx 5N\log_2 N$ operations.

For an N term expansion we have

$$u(x_j) \;=\; \sum_{k=1}^{N} \phi_k(x_j)\hat{u}_k. \tag{655}$$

There are two cases to consider,

$$1.\ x_j \;=\; jh, \quad h \;=\; \frac{2\pi}{N},\ \ j = 0, \ldots, N-1, \tag{656}$$

$$2.\ x_j \;=\; j\delta, \quad \delta \;=\; \frac{2\pi}{M},\ \ j = 0, \ldots, M-1,\ \ M \;>\; N \ \text{(say)}. \tag{657}$$

As is typically done, Case 2 can be recast as Case 1 by setting $\hat{u}_k = 0$, $k = N+1, \ldots, M$, and then evaluating

$$u(x_j) \;=\; \sum_{k=1}^{M} \phi_k(x_j)\hat{u}_k, \quad j = 0, \ldots, M-1, \quad x_j = 2\pi j/M. \tag{658}$$

To cast this into matrix form, let P be the prolongation matrix that extends $\underline{\hat{u}}$ by $(M-N)$ zeros, Z be the $M \times M$ interpolation matrix with entries $Z^M_{jk} = \phi_k(x_j)$, and $J := ZP$. We then have

$$\underline{u}^M \;=\; J\underline{\hat{u}} \;=\; ZP\underline{\hat{u}}, \tag{659}$$

where $\underline{u}^M$ is the set of interpolated nodal values $[\underline{u}^M]_j = u(jh^M)$, $j = 0, \ldots, M-1$ with spacing on the fine grid, $h^M := 2\pi/M$.

As it stands, Z is a full $M \times M$ matrix, such that (659) requires $2M$ operations. We can recast this into the "standard" form[14] for the DFT (e.g.,(2.1.25) and (2.1.27) in [CHQZ2]) by using the identities,

$$\sin kx \;=\; \frac{1}{2i}\Big(e^{ikx} \,-\, e^{-ikx}\Big) \tag{660}$$

$$\cos kx \;=\; \frac{1}{2}\Big(e^{ikx} \,+\, e^{-ikx}\Big), \tag{661}$$

where, as usual with Fourier bases, $i := \sqrt{-1}$. We'd like to express (655) as a sum of complex exponentials. Because of our particular numbering a direct translation is not straightforward, but we can derive the transformation by considering a simple case. With $N = 3$ we have

$$\begin{aligned} u(x_j) \;&=\; e^{i0x}\hat{u}_1 \,+\, \frac{1}{2}\Big(e^{ix} \,+\, e^{-ix}\Big)\hat{u}_2 \,+\, \frac{1}{2i}\Big(e^{ix} \,-\, e^{-ix}\Big)\hat{u}_3 \\ &=\; \frac{1}{2}(\hat{u}_2 + i\hat{u}_3)e^{-ix} \,+\, \hat{u}_1 e^{i0x} \,+\, \frac{1}{2}(\hat{u}_2 - i\hat{u}_3)e^{ix} \\ &=\; \hat{a}_{-1}e^{-ix} \,+\, \hat{a}_0 e^{i0x} \,+\, \hat{a}_1 e^{ix}. \end{aligned} \tag{662}$$

By inspection we deduce, for $k > 0$,

$$\hat{a}_k \;=\; \frac{1}{2}(\hat{u}_{2k} \,-\, i\hat{u}_{2k+1}), \quad \hat{a}_{-k} \;=\; \hat{a}^*_k, \quad \hat{a}_0 \;=\; \hat{u}_1. \tag{663}$$

Our N-point interpolation formula (N odd) is given by

$$u(x_j) \;=\; \sum_{k \,=\, \frac{1-N}{2}}^{\frac{N-1}{2}} e^{ikj2\pi/N}\, \hat{a}_k \tag{664}$$

$$=\; \sum_{k \,=\, \frac{1-N}{2}}^{\frac{N-1}{2}} \gamma^{kj}\, \hat{a}_k, \tag{665}$$

[14] In reality, there is no standard form; compare [CHQZ2] (2.1.25) and (2.1.27) with the matlab definitions of the DFT in the boxed section below.

where $\gamma = e^{2\pi i/N}$ is the Nth-root of unity. For the M-point interpolation formula we simply substitute M for each N in (664).

While the matrix with entries γ raised to the kj power is completely full, it is possible to factor it into a sequence of sparse matrix-vector products such that the computational complexity is reduced from $2N^2$ (N^2 complex-valued multiply-adds) to $\approx 5N \log_2 N$ operations. We denote this matrix as F^{-1}, which is the inverse of the discrete Fourier transform (DFT), which is invariably implemented as in the inverse *fast Fourier transform* (FFT). Associated with this is the forward FFT, $F = \frac{1}{N}\gamma^{-jk}$. Unfortunately, there is no standard definition of the FFT and users must familiarize themselves with the expected input/output behavior of each FFT implementation prior to using it. It pays therefore, to write a simulation code with a particular format in mind and then use a wrapper to implement the change of variables via the FFT that is available. That is what we provide in the *real* FFT script, `rfft.m`, which takes data at N points, $x_j = jh$, $j = 0, \ldots, N-1$, $h = 2\pi/N$ and returns the basis coefficients associated with the sine/cosine basis of (639)–(640). Similarly, the inverse, `irfft.m`, performs the synthesis, which takes the $\underline{h}_k$ basis coefficents and evaluates (639) at points x_j. In anticipation of solving PDEs, both of these scripts support matrix inputs, which means they can apply the FFT to multiple columns at a time. As in the matrix-matrix case used for the Legendre spectral method, significant performance gains are realized by applying these transformations in block form (i.e., as tensor contractions).

Real-to-Real FFT and Inverse

Below is the matlab interface to the FFT and its inverse. It does not match the form (664) nor the form given in [CHQZ2] (2.1.25) and (2.1.27).

```
FFT Discrete Fourier transform.
   FFT(X) is the discrete Fourier transform (DFT) of vector X.  For
   matrices, the FFT operation is applied to each column. For N-D
   arrays, the FFT operation operates on the first non-singleton
   dimension.

   FFT(X,N) is the N-point FFT, padded with zeros if X has less
   than N points and truncated if it has more.

   FFT(X,[],DIM) or FFT(X,N,DIM) applies the FFT operation across the
   dimension DIM.

   For length N input vector x, the DFT is a length N vector X,
   with elements
                    N
      X(k) =       sum  x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
                   n=1
   The inverse DFT (computed by IFFT) is given by
                    N
      x(n) = (1/N) sum  X(k)*exp( j*2*pi*(k-1)*(n-1)/N), 1 <= n <= N.
                   k=1

   See also FFT2, FFTN, FFTSHIFT, FFTW, IFFT, IFFT2, IFFTN.
```

Exercise
Use the real-to-real FFT routines provided on the Relate page, `rfft.m` and `irfft.m` to answer the following questions.

- Write a script `uM=interp_f(M,u)` that will perform the Fourier interpolation of an N-point set of function values, $u_j = u(x_j)$, $j = 0 : N-1$, to an M-point set of output values.

- Use this script to interpolate a square wave $u = sign(x - \pi)$, $0 < x < 2\pi$ with N=20 and $M = 10N$ and plot the result.

- Use this script to interpolate a square wave $u = sign(x - \pi)$, $0 < x < 2\pi$ with N=80 and $M = 10N$ and plot the result.

- Use this script to interpolate a square wave $u = sign(x - \pi)$, $0 < x < 2\pi$ with N=200 and $M = 10N$ and plot the result.

- What is $\max_j uM(j)$? Does it approach a limit as $N \longrightarrow \infty$?
- Repeat the preceding exercises with $u(x) = (x - \pi)/\pi$.

- What is $\max_j uM(j)$?

- Use the given real FFT functions to plot $\cos 3x$ on $[0, 2\pi]$ with at least 80 interpolation points.

13.5 Evaluation of the Advection Term

$$\underline{u}^M \quad = \quad J\underline{\hat{u}} = Z^M P^M. \tag{666}$$

We'll revisit this topic later, with a slight improvement in notation. The key will be to introduce a new operator, `trfft.m`, which will apply the transpose of the *hat-to-physical* space operator, rather than the inverse. It is the transpose that is needed for the variational formulations.

Also, timings indicate that the matrix-matrix product approach is faster than the FFT based approach for $N < \approx 200$, so we may as well just use our real-space (i.e., nodal) variables and apply matrices in this space, which allows us to readily switch from Legendre (i.e., Lagrange on GLL points) bases to Fourier bases.

14 Stokes and Navier-Stokes

See the `var_stokes.pdf` in the course notes page.

15 Complex Geometries

We have two mechanisms by which we can stretch our domain complexity beyond the tensor-product box. The first is through geometric deformation, which we've already covered. As with Poisson, for the Navier-Stokes equations deformation amounts simply to applying the chain rule to transform derivatives from $\hat{\Omega}$ to Ω and using quadrature with an appropriate Jacobian to evaluate the bilinear forms associated with the WRT. We will revisit deformed geometries in this context later in the course.

The second approach is to support multiple subdomains, each of which is a (possibly deformed) tensor-product box. This is the motivation behind the *spectral element method* (SEM) [Patera, JCP84], which is a (fast) subset of the finite element method (FEM). In this case, we retain our standard exponentially-convergent approximation properties and, hence, exponential convergence in the numerical approximation to PDEs having sufficiently regular solutions. In addition, the SEM retains the performance advantages of locally-structured tensor-product forms, which is what allows it to be applied with $N > 4$. (The $O(N^6)$ overhead restricts $N \leq 4$ for the standard p-type FEM.)

15.1 Multi-Domain Spectral Methods

Exercise
Consider the unsteady 1D advection problem with periodic boundary conditions on [0,1] and unit advecting velocity $c = 1$. Discretize this problem in space with E elements of order N (a total of $n = EN$ dofs) and RK4 in time. Keep your dofs in *global* representation and write a function that returns the action of the advection operator, C upon an input vector $\underline{u}$.

For a uniform partition of $E = 256/N$, plot $\rho \Delta x_{\min}/c$ vs. N for $N = 1$, 2 ,4, 8, 16, ..., 256. Here ρ is the spectral radius of the evolution operator $B^{-1}C$, where B is the full mass matrix. Carry out this exercise for both the diagonal mass matrix and the standard mass matrix, $B_{ij} = \int \phi_i \phi_j \, \Delta x$. Plot both curves on the same figure.

Using a steep Gaussian or similar curve (I like $[\sin(\pi x)]^{80}$) as an initial condition, run your code for time $t = 0$ to $t = 5$ (five times around) and compute the max point-wise error. Plot this error vs. N for both the diagonal and consistent mass-matrix cases.

A significant advantage of the SEM over the pure spectral method is that we can increase resolution and avoid severe CFL-limited constraints on Δt by increasing the number of subdomains (elements) for fixed (high) N, rather than simply increasing N. This added flexibility is reflected in two discretization parameters, E, the number of elements, and N their order. (One can also have N be different in each subdomain.)

In the following exercise, we consider the 1D advection problem with periodic boundary conditinos,

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \qquad u(0,t) = u(1,t), \tag{667}$$

and initial condition $u^0(x)$. We are interested in determining the maximum timestep size, Δt that would be allowed for stability as a function of (E, N). We also would like to understand the accuracy of this method compared to, say, 2nd-order finite differences.

To answer the question about maximum timestep size, consider the following. Our WRT will give rise to a system of the form,

$$B\frac{d\underline{u}}{dt} \;=\; -C\underline{u}, \tag{668}$$

and stability is governed by the eigenvalues λ_k of the evolution operator, $B^{-1}C$. Specifically, we require,

$$\max_k |\lambda_k|\,\Delta t \;<\; C, \tag{669}$$

where $C \approx 1$ is determined by the stability region of our chosen timestepper. We typically approximate our timestep bound by the CFL constraint

$$\mathrm{CFL} \;:=\; \max_i \left|\frac{c_i}{\Delta x_i}\right| \Delta t \;<\; sC, \tag{670}$$

where s is an order-unity constant that we need to find. We have

$$\frac{\mathrm{CFL}}{s} \;<\; C \;=\; \max_k |\lambda_k \Delta t|\,. \tag{671}$$

Therefore

$$\frac{\mathrm{CFL}}{s} \;=\; \max_k |\lambda_k \Delta t|\,. \tag{672}$$

$$s \;=\; \frac{\mathrm{CFL}}{\max_k |\lambda_k \Delta t|} \;=\; \frac{\max_i \left|\frac{c_i}{\Delta x_i}\right|}{\max_k |\lambda_k|}. \tag{673}$$

16 Navier-Stokes Splitting Strategies

Here we introduce a 3rd-order time-split formulation of the Navier-Stokes equations that is appropriate for high-order spectral element methods. We begin with a bit of background on a highly flexible and efficient semi-implicit timestepper.

16.1 BDFk/EXTk Timestepping

Before starting on the derivation of the $\mathbb{P}_N - \mathbb{P}_N$ splitting formulation we introduce a kth-order semi-implicit timestepping scheme based on backward differentation coupled with extrapolation, which we denote as BDFk/EXTk. Consider the unsteady advection diffusion equation,

$$\frac{\partial u}{\partial t} - \nu\nabla^2 u \;=\; -\mathbf{c}\cdot\nabla u + f \;+\; \text{BCs/ICs.} \tag{674}$$

Evaluate this equation at time t^n,

$$\left.\frac{\partial u}{\partial t}\right|_{t^n} - \left.\nu\nabla^2 u\right|_{t^n} \;=\; (-\mathbf{c}\cdot\nabla u + f)_{t^n} \;=\; g|_{t^n}\,. \tag{675}$$

We use a backward difference formula (BDF) for the time derivative at t^n and extrapolation for the right-hand side terms, which we denote as g. The formulas for $k = 1$, 2, and 3 and uniform Δt are

$$\begin{array}{lll}
k=1 & \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{u^n - u^{n-1}}{\Delta t} + O(\Delta t) & g|_{t^n} = g^{n-1} + O(\Delta t) \\
k=2 & \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{3u^n - 4u^{n-1} + u^{n-2}}{2\Delta t} + O(\Delta t^2) & g|_{t^n} = 2g^{n-1} - g^{n-2} + O(\Delta t^2) \\
k=3 & \left.\frac{\partial u}{\partial t}\right|_{t^n} = \frac{11u^n - 18u^{n-1} + 9u^{n-2} - 2u^{n-3}}{6\Delta t} + O(\Delta t^3) & g|_{t^n} = 3g^{n-1} - 3g^{n-2} + g^{n-3} + O(\Delta t^3)
\end{array}$$

Combining these $O(\Delta t^k)$ approximations with implicit treatment of the viscous term, $-\nu\nabla^2 u^n$, leads to

$$\beta_0 u^n - \Delta t\nu\nabla^2 u^n \;=\; -\sum_{j=1}^{k}\left(\beta_j u^{n-j} + \Delta t\alpha_j\left(\mathbf{c}^{n-j}\cdot\nabla u^{n-j} - f^{n-j}\right)\right), \tag{676}$$

where β_j and α_j are the respective BDFk/EXTk coefficients given in Table 1, which also shows the stability regions for the explicit part (e.g., if $\nu = 0$). Note that the stability region for $k = 3$ encompasses part of the imaginary axis, which is important for the advection operator.

To introduce the splitting idea, consider the split representation of (676),

$$\hat{u} \;=\; -\sum_{j=1}^{k}\left(\beta_j u^{n-j} + \Delta t\alpha_j\left(\mathbf{c}^{n-j}\cdot\nabla u^{n-j} - f^{n-j}\right)\right), \tag{677}$$

$$\beta_0 u^n - \Delta t\nu\nabla^2 u^n \;=\; \hat{u}. \tag{678}$$

Formally, we've done nothing other than to introduce a new variable $\hat{u}$ which holds the right-hand side data of (676). However, $\hat{u}$ has physical significance. If there were no diffusion term, then $\hat{u}$ would be the solution, save for the factor of β_0: $u^n = \beta_0^{-1}\hat{u}$. Moreover, note that no boundary conditions are imposed when evaluating $\hat{u}$—those are formally applied when we solve (678). In the case of semi-implicit advection-diffusion, there is no difficulty in carrying through with this splitting scheme. For Navier-Stokes, however, the boundary conditions couple the pressure and velocity components, which makes the system more difficult to solve. In the next section we combine BDFk/EXTk with splitting to develop and efficient 3rd-order accurate splitting in which each of these terms are decoupled.

k	β_0	β_1	β_2	β_3	α_1	α_2	α_3
1	1	-1	0	0	1	0	0
2	$\frac{3}{2}$	$-\frac{4}{2}$	$\frac{1}{2}$	0	2	-1	0
3	$\frac{11}{6}$	$-\frac{18}{6}$	$\frac{9}{6}$	$-\frac{2}{6}$	3	-3	1

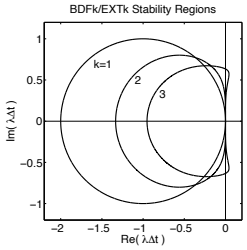


Table 1: BDFk/EXTk coefficients for uniform Δt and stability regions for explicit part.

16.2 Splitting for Navier-Stokes

The unsteady incompressible Navier-Stokes and even the linear unsteady Stokes equations correspond to set of time- and space-dependent PDEs involving d equations for the momentum that are coupled through the divergence-free constraint. This constraint is manifest through the addition of the pressure, which is the effective Lagrange multiplier that allows the constraint to be satisfied. With $\nu := 1/Re$, the equations in nondimensional form read,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \nu \nabla^2 \mathbf{u} - \nabla p + \mathbf{f} \tag{679}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{680}$$

A major challenge in advancing these equations is the tight coupling of the velocity components through divergence-free constraint (680), which must be satisfied at all times. In these equations, the sound speed is effectively infinite, which means that the pressure must be computed implicitly as it is associated with the (infinitely) fastest time scale in the problem. If we have $\approx n$ velocity grid points and $\approx n$ pressure points, then we have a system of size $4n$ unknowns to be solved at every timestep. Moreover, this system is not a well-behaved SPD system like the Poisson equation. It is typically an indefinite saddle problem, which makes it significantly more challenging to solve.

Our point of departure is to discretize (679) using BDFk/EXTk, which leads to

$$\beta_0 \mathbf{u}^n - \Delta t \nu \nabla^2 \mathbf{u}^n + \Delta t \nabla p^n = -\sum_{j=1}^{k} \beta_j \mathbf{u}^{n-j} + \Delta t \sum_{j=1}^{k} \alpha_j \left(\mathbf{f} - \mathbf{u} \cdot \nabla \mathbf{u} \right)^{n-j} \tag{681}$$

$$\nabla \cdot \mathbf{u}^n = 0. \tag{682}$$

Here, β_j and α_j are the respective coefficients for the kth-order backward difference and extrapolation terms. With this scheme, the pressure is evaluated precisely at time t^n and is chosen such that (682) is satisfied.

To decouple the momentum and pressure equations, we begin with the splitting formulation introduced above,

$$\hat{\mathbf{u}} = -\sum_{j=1}^{k} \beta_j \mathbf{u}^{n-j} + \Delta t \sum_{j=1}^{k} \alpha_j \left(\mathbf{f} - \mathbf{u} \cdot \nabla \mathbf{u} \right)^{n-j} \tag{683}$$

$$\beta_0 \mathbf{u}^n - \Delta t \nu \nabla^2 \mathbf{u}^n + \Delta t \nabla p^n = \hat{\mathbf{u}} \tag{684}$$

$$\nabla \cdot \mathbf{u}^n = 0. \tag{685}$$

As before, $\hat{\mathbf{u}}$ is a tentative solution that incorporates the inertia and the nonlinear advection terms. It does not, however, have the correct boundary values for $\mathbf{u}^n$ nor does it satisfy the divergence-free constraint. For the latter, we can derive an elliptic equation for the pressure by taking the divergence of (684) and using (685) to yield

$$-\nabla^2 p^n \quad = \quad -\frac{1}{\Delta t}\hat{\mathbf{u}}. \tag{686}$$

The boundary conditions for (686) can be found by dotting (684) with the outward-pointing unit normal on $\partial\Omega$,

$$\nabla p^n \cdot \hat{\mathbf{n}}\,|_{\partial\Omega} \quad = \quad \frac{1}{\Delta t}\left(\hat{\mathbf{u}} - \beta_0 \mathbf{u}^n + \Delta t \nu \nabla^2 \mathbf{u}^n\right) \cdot \hat{\mathbf{n}}\Big|_{\partial\Omega}. \tag{687}$$

All terms on the right of (687) are known, save for the viscous term. This term can be recovered by extrapolating in time, as we do with the nonlinear terms. It is beneficial, however, to first apply the following vector identity: *For any sufficiently differentiable vector field,* $\mathbf{w}$*, we have*

$$\nabla^2 \mathbf{w} \quad = \quad \nabla(\nabla \cdot \mathbf{w}) \; - \; \nabla \times (\nabla \times \mathbf{w}). \tag{688}$$

In the present case, $\nabla \cdot \mathbf{u} = 0$, so we can replace $\nabla^2 \mathbf{u}$ in (687) to yield

$$\nabla p^n \cdot \hat{\mathbf{n}}\,|_{\partial\Omega} \quad = \quad \frac{1}{\Delta t}\left(\hat{\mathbf{u}} - \beta_0 \mathbf{u}^n - \Delta t \nu \nabla \times \nabla \times \tilde{\mathbf{u}}^n\right) \cdot \hat{\mathbf{n}}\Big|_{\partial\Omega}. \tag{689}$$

Here, we take $\tilde{\mathbf{u}}^n := \sum_{j=1}^{k} \alpha_j \mathbf{u}^{n-j}$ to be the kth-order extrapolant of $\mathbf{u}^n$. Once we have p^n, we take the final splitting step,

$$\hat{\hat{\mathbf{u}}} \quad = \quad \hat{\mathbf{u}} \; - \; \Delta t \nabla p^n \tag{690}$$

$$\left(-\beta_0 \mathbf{u}^n + \Delta t \nu \nabla^2 \mathbf{u}^n\right) \quad = \quad \hat{\hat{\mathbf{u}}}\,, \tag{691}$$

with standard velocity boundary conditions on $\mathbf{u}^n$. (See Eq. (3.59) in Moser, 2022, for treatment of the curl-curl boundary terms in the pressure equation.)

17 Useful Inequalities

In this section, we present a few of the basic inequalities that are fundamental analysis tools for variational methods such as the FEM.

17.1 Bilinear Forms

Consider a map (u, v) that takes elements u, and v of a vector space V into the reals. We say that (u, v) is bilinear if, for all u, v,$w \in V$, and real numbers α,β,

$$\begin{aligned} (\alpha u, v) &= \alpha(u, v) = (u, \alpha v) && (692)\\ (u, v + w) &= (u, v) + (u, w) && (693)\\ (u + w, v) &= (u, v) + (w, v). && (694) \end{aligned}$$

We call the bilinear form *symmetric* if, for all u, $v \in V$,

$$(u, v) = (v, u), \tag{695}$$

and *positive definite* if, for all $u \in V$, $u \neq 0$,

$$(u, u) > 0. \tag{696}$$

Clearly, if $u = 0$ then $(u, u) = 0$ by (692).

Examples.

i. $V = \mathbb{R}^n$ and

$$(\underline{u}, \underline{v}) := \sum_{i=1}^{n} v_i u_i. \tag{697}$$

ii. $V = L^2(\Omega)$ (the space of functions $v(\mathbf{x})$ on $\Omega \subset \mathbb{R}^d$, $d \geq 1$ such that $\int_\Omega v^2 d\mathbf{x} < \infty$), and

$$(v, u) := \int_\Omega v(\mathbf{x}) u(\mathbf{x})\, d\mathbf{x}. \tag{698}$$

iii. $V = \mathcal{H}^1(\Omega)$ ($:= \{ v |\, v \in L^2(\Omega)\}$ and $\int_\Omega \nabla v \cdot \nabla v d\mathbf{x} < \infty\}$), and

$$a(v, u) := \int_\Omega \nabla v \cdot \nabla u d\mathbf{x}. \tag{699}$$

iv. $V = \mathcal{H}^1(\Omega)$ and

$$c(v, u) := \int_\Omega v(\mathbf{x})\, \mathbf{c} \cdot \nabla u(\mathbf{x})\, d\mathbf{x}. \tag{700}$$

Note that *i–iii* are symmetric, whereas *iv* is not. Note also that *i* and *ii* are positive-definite, whereas *iii* is positive semi-definite, meaning $a(u, u) \geq 0$ for any $u \in \mathcal{H}^1$.

If the bilinear form is symmetric positive definite (SPD) then we have an associated norm,

$$\|u\| := \sqrt{(u, u)}. \tag{701}$$

17.2 Cauchy-Schwarz

For the SPD case we have the *Cauchy-Schwarz inequality*,

$$|(v,u)| \;\leq\; \|v\|\,\|u\|. \tag{702}$$

In the case of $\underline{u},\ \underline{v} \in \mathbb{R}^n$, Cauchy-Schwarz (CS) is a direct manifestation of a familiar geometric identity,

$$(\underline{u},\underline{v}) \;=\; \underline{u}^T\underline{v} \;=\; \|\underline{u}\|\,\|\underline{v}\|\,\cos\theta, \tag{703}$$

where θ is the angle between the two vectors $\underline{u}$ and $\underline{v}$.

The more general case is established by first noting that (702) holds trivially if $v \equiv 0$. For $v \neq 0$, consider the scalar

$$\lambda \;=\; \frac{(u,v)}{(v,v)} \;=\; \frac{(u,v)}{\|v\|^2}, \tag{704}$$

and the norm,

$$0 \;\leq\; \|u - \lambda v\|^2 \tag{705}$$

$$=\; (u,u) - 2\lambda(u,v) + \lambda^2(v,v). \tag{706}$$

Substituting (704) for λ and rearranging yields,

$$(u,v)^2 \;\leq\; (u,u)\,(v,v) \;=\; \|u\|^2\,\|v\|^2 \tag{707}$$

or

$$|(u,v)| \;\leq\; \|u\|\,\|v\|. \tag{708}$$

It may seem, rightly, that the algebraic steps (704)–(708) were pulled out of thin air like a magician's trick. A more intuitive understanding can be had by considering the rank-2 vector space $[u,v] \subset V$, $v \neq 0$, and computing the projection of u onto v. The solution to this problem is

$$\bar{u} = \frac{(u,v)}{(v,v)}v \;=\; \mathcal{P}u, \tag{709}$$

where $\mathcal{P}$ is the projection operator, as we now establish. First, let us look at some properties of $\bar{u}$. It is a multiple of v, as it should be. Its length scales with u, independent of $\|v\|$, which can be seen by considering the projection onto αv for $\alpha > 0$,

$$\bar{w} \;=\; \frac{(u,\alpha v)}{(\alpha v,\alpha v)}\alpha v \tag{710}$$

$$=\; \frac{(u,v)}{(v,v)}v \tag{711}$$

$$=\; \bar{u}. \tag{712}$$

Moreover, $\mathcal{P} = \mathcal{P}^2$ is idempotent, which is to say that the projection of a projection is equal to the original projection, $\mathcal{P}\bar{u} = \bar{u}$, as is readily shown by taking $\alpha = \frac{(u,v)}{(v,v)}$ in (710). Finally, it easy to show that (709) satisfies an important projective property, namely, that it minimizes the norm associated

with the $(\cdot,\cdot)$ inner product of $u - w$ among all $w = \alpha v$. To see this, consider $w = \bar{u} - \epsilon v \in \mathcal{R}(v)$, with $\epsilon \in \mathbb{R}$, $v \in V$, If $\bar{u}$ is the minimizer, then we have, for all $\epsilon \in \mathbb{R}$,

$$\begin{aligned} \|u - \bar{u}\|^2 &\leq \|u - \bar{u} + \epsilon v\|^2 && (713)\\ &= \|u - \bar{u}\|^2 + 2\epsilon(v, u - \bar{u}) + \epsilon^2\|v\|^2, && (714)\end{aligned}$$

which can be true only if and only if

$$(v, u - \bar{u}) = 0 \quad (715)$$

for all $v \in V$. Setting $\bar{u} = \alpha\, v$ in (715) and solving for α, we have

$$(v, u) = (v, \bar{u}) = (v, \alpha\, v) = \alpha\,(v, v), \quad (716)$$

or $\alpha = \frac{(v,u)}{(v,v)}$, as stated in (709).

Defining $\mathcal{I}$ as the identity operator on V, we can write $u = \mathcal{P}u + (\mathcal{I} - \mathcal{P})u$, which represents the decomposition of u into its projection onto v and the orthogonal complement, as illustrated in Fig. 16. Note that $\mathcal{P}(\mathcal{I} - \mathcal{P})u = (\mathcal{P} - \mathcal{P})u = 0$ because $\mathcal{P}$ is idempotent. From this property, we have the Pythagorean theorem,

$$\|u\|^2 = \|\mathcal{P}u\|^2 + \|(\mathcal{I} - \mathcal{P})u\|^2, \quad (717)$$

and Cauchy-Schwarz follows,

$$\begin{aligned} \|u\|^2 &\geq \|\mathcal{P}u\|^2 && (718)\\ &= \left(\frac{(u, v)}{\|v\|^2}\right)^2 \|v\|^2 && (719)\\ &= \frac{(u, v)^2}{\|v\|^2}, && (720)\end{aligned}$$

with the final result

$$\|v\|\|u\| \geq |(u, v)|. \quad (721)$$

Notice that equality holds if and only if u is a multiple of some $v \in V$ because, in this case, $(\mathcal{I} - \mathcal{P})u = u - \mathcal{P}u = u - u$. That is, the projection of $u = \alpha v$ onto V is $\alpha v = u$.

Exercises.

v. Let $\Omega = [-1, 1]$, $v = 1 - x^2$, and $u = \cos\frac{\pi}{2}x$. Show that Cauchy-Schwarz holds for the L^2 inner product of Example (*ii*).

vi. For the domain and inner product of Exercise (*v*) can you find a polynomial $v \in \mathbb{P}_3$ (the space of all polynomials of degree ≤ 3 on Ω) such that $(v, u) = 0$ for $u = \cos\frac{\pi}{2}x$?

vi. For the domain and inner product of Exercise (*v*) can you find a polynomial $v \in \mathbb{P}_9$ such that equality holds for Cauchy-Schwarz?

17.3 Poincaré Inequality

The Poincaré inequality is important for establishing stability (a.k.a. coercivity) for numerical methods. Here, we consider the $\mathcal{H}^1$ semi-norm defined as

$$|u|_a := \sqrt{a(u, u)} = \left[\int_\Omega \nabla u \cdot \nabla u \, d\mathbf{x}\right]^{\frac{1}{2}}. \quad (722)$$

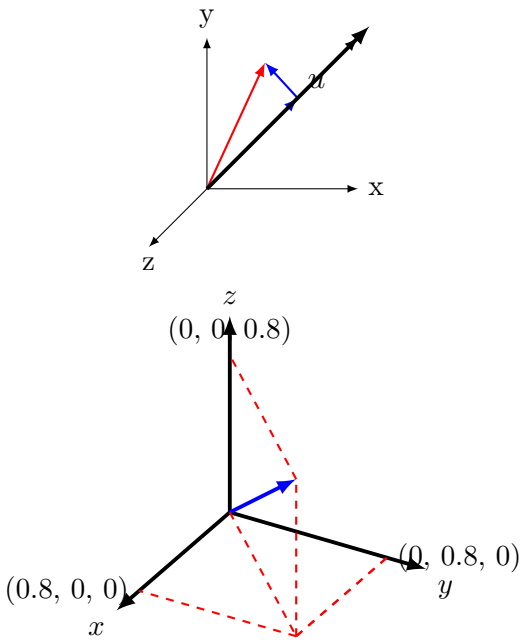


Figure 16: Projection of u onto v (under construction).

If $u(\mathbf{x})$ =constant on Ω then $|u|_a = 0$. Otherwise, for any nonconstant continuous function ($u \in C^0$) we have $|u|_a > 0$. In addition to $\mathcal{H}^1$, we will need $\mathcal{H}^1_0 \subset \mathcal{H}^1$, which is the space of functions in $\mathcal{H}^1$ that vanish on a nontrivial subset of the boundary of Ω, $\partial\Omega_D \subset \partial\Omega \subset \Omega$. Here, $\partial\Omega_D$ is assumed to be the Dirichlet portion of the domain boundary, $\partial\Omega$. Note that the only constant in $\mathcal{H}^1_0$ is $u = 0$, which implies that $|u|_a$ is a norm on $\mathcal{H}^1_0$.

The Poincaré inequality states that there exists a constant $\alpha > 0$ (which is a function of Ω) such that for all $u \in \mathcal{H}^1_0$,

$$\alpha\,(u,u) \quad \leq \quad a(u,u) = \int_\Omega \nabla u \cdot \nabla u \, d\mathbf{x}. \tag{723}$$

By dimensional analysis, $\alpha \propto L^{-2}$, where L is a characteristic length scale (e.g., the diameter) of Ω.

We'll look at two examples for establishing an estimate of α in the 1D (d=1) case with $\Omega = [0, L]$ and $\partial\Omega = [0]$ (meaning $u(0) = 0$ for all $u \in \mathcal{H}^1_0$). Following Strang and Fix [SF73], we begin with

$$u(x) \;=\; \int_0^x 1 \cdot u'(\xi)\, d\xi \;\leq\; \left[\int_0^x 1^2\, d\xi\right]^{\frac{1}{2}} \left[\int_0^x [u'(\xi)]^2\, d\xi\right]^{\frac{1}{2}} \;=\; \sqrt{x}\left[\int_0^x [u'(\xi)]^2\, d\xi\right]^{\frac{1}{2}}, \tag{724}$$

which is a direct application of the Cauchy-Schwarz inequality. Squaring both sides and integrating with respect to x we have

$$\begin{aligned}
\|u\|^2 \;=\; \int_0^L u(x)^2\, dx \quad &\leq \quad \int_0^L x \cdot \left(\int_0^x [u'(\xi)]^2\, d\xi\right) dx && (725)\\
&\leq \quad \int_0^L M_1 \cdot M_2\, dx && (726)\\
&\leq \quad L\; M_1 \cdot M_2 && (727)\\
&= \quad L^2 \int_0^L (u')^2\, d\xi && (728)\\
&= \quad L^2\, |u|_a^2. && (729)
\end{aligned}$$

Here, $M_1 = \max_\Omega x = L$ and $M_2 = \max_\Omega \int_0^x (u')^2\, d\xi = |u|_a^2$ are constants that maximize the integrand in (725).

Examples. Let's verify the estimated Poincaré constant $\alpha = L^{-2}$ for a few specific cases of $u(x)$. (Equation (723) must of course hold for all $u(x) \in \mathcal{H}_0^1$.) As a first example, consider $u(x) = x^m$, $m \geq 1$.

$$\frac{1}{L^2}\int_0^L u^2\, dx = \frac{L^{2m+1}}{2m-1} \tag{730}$$

$$\int_0^L [u']^2\, dx = \frac{2m}{2m-1} L^{2m-1} \tag{731}$$

$$> \frac{1}{2m+1} L^{2m-1}. \tag{732}$$

As another example, consider $u = \sin x$, $L = \frac{\pi}{2}$.

$$\frac{1}{L^2}\int_0^L u^2\, dx = \frac{1}{\pi} \tag{733}$$

$$\int_0^L u'^2\, dx = \frac{\pi}{4} > \frac{1}{\pi}. \tag{734}$$

Finally, one might wonder, *What is the largest value of α that can be found for this 1D case on $[0, L]$?* That is, we want to find $\alpha_{\max}$ such that, for all $v \in \mathcal{H}_0^1$,

$$\frac{a(v,v)}{(v,v)} \geq \min_{u \in \mathcal{H}_0^1} \frac{a(u,u)}{(u,u)} =: \alpha_{\max}. \tag{735}$$

In this case, $\alpha_{\max}$ corresponds to the minimum eigenvalue $\tilde{\lambda}$ satisfying

$$\mathcal{L}u = \tilde{\lambda} u, \tag{736}$$

with $\mathcal{L}u = -u''$, $u(0) = 0$, $u'(L) = 0$, as we illustrate later in the context of variational methods such as the FEM.

Let us recall that for a finite-dimensional system, $A\underline{u} = \underline{f}$, with $A \in \mathbb{R}^{n\times n}$ being SPD, we have n eigenpairs, $(\lambda_j, \underline{z}_j)$ satisfying

$$A\underline{z}_j = \lambda_j \underline{z}_j, \quad j = 1, \ldots, n, \tag{737}$$

with $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Because A is symmetric, there exists a set of orthonormal eigenvectors satisfying $(\underline{z}_i, \underline{z}_j) := \underline{z}_i^T \underline{z}_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta. Let $Z = [\underline{z}_1\ \underline{z}_2\ \ldots\ \underline{z}_n]$. Then, for any $\underline{y} \in \mathbb{R}^n$, we have $\underline{y} = Z\underline{\hat{y}} \iff \underline{\hat{y}} = Z^{-1}\underline{y} = Z^T\underline{y}$. It is easy to show that, for $\underline{y} \in \mathbb{R}^n$,

$$\min_{\underline{y}} \frac{\underline{y}^T A \underline{y}}{\underline{y}^T \underline{y}} = \lambda_1 \tag{738}$$

$$\max_{\underline{y}} \frac{\underline{y}^T A \underline{y}}{\underline{y}^T \underline{y}} = \lambda_n \tag{739}$$

For the particular case of the 1D finite difference approximation to $\mathcal{L}$, we have with $h := L/n$,

$$\lambda_1 = \left(\frac{\pi}{2}\right)^2 + O(h^2) \sim \left(\frac{\pi}{2}\right)^2 \tag{740}$$

$$\lambda_n = O(n^2). \tag{741}$$

Clearly, λ_n is not bounded unless n is fixed. By contrast, λ_1 is bounded as $n \longrightarrow \infty$.

The lower bound (738) holds true also for the continuous eigenvalue problem (736). That is, for $u \in \mathcal{H}_0^1$,

$$\min_u \frac{a(u,u)}{(u,u)} \quad = \quad \tilde{\lambda}_1 \; = \; \left(\frac{\pi}{2}\right)^2, \tag{742}$$

which implies that, for all $u \in \mathcal{H}_0^1$,

$$\alpha\,(u,u) \quad \leq \quad \frac{a(u,u)}{\tilde{\lambda}_1}, \tag{743}$$

where $\tilde{\lambda}_1$ is the minimum eigenvalue of the continuous problem. Thus, $\alpha = \tilde{\lambda}_1$ is the maximum possible constant in the Poincaré inequality for the Laplace operator (736). Because of (741), it is also clear that, in the continuous case, we cannot find a constant C_2 such that

$$a(u,u) \leq C_2(u,u), \tag{744}$$

for all $u \in V$. (That is, there is no maximum eigenvalue for the continuous problem.) On the other hand, for the finite-dimensional case (with n *fixed*), we can choose $C_2 = \lambda_n$, for which (744) *does hold.* This observation is a manifestation of the equivalence of norms, which says that for any two norms $\|\cdot\|_\circ$ and $\|\cdot\|_*$ on a finite dimensional vector space V, there exist constants c and C (dependent on n, the dimension of V), such that

$$c\|v\|_\circ \quad \leq \quad \|v\|_* \; \leq \; C\|v\|_\circ \tag{745}$$

for all $v \in V$. Such a result does not generally hold in the infinite-dimensional case, and norm-equivalence must be established on a case-by-case basis.

18 Error-Estimator Notes for Reduced Basis Approximations

We consider a reduced-basis (RB) approximation to solution of the Poisson equation with a single (fixed) parameter, μ, given here in weak form: *Find* $\tilde{u} \in \mathcal{H}_0^1(\Omega)$ *such that, for all* $v \in \mathcal{H}_0^1$,

$$a(v,\tilde{u}) \;:=\; \mu \int_\Omega \nabla v \cdot \nabla \tilde{u}\, dV \;=\; \int_\Omega v\, f\, dV. \tag{746}$$

We assume that $\mu > 0$, $f \in \mathcal{L}^2$, and that Ω has sufficient regularity that all the integrals are well defined.

Next, we assume that we have (say) a spectral element solution $u(\mathbf{x}) \approx \tilde{u}(\mathbf{x})$ that lives in a finite-dimensional subspace, $X_0^{\tilde{N}} \subset \mathcal{H}_0^1$ that is close enough to $\tilde{u}$ to be considered our *truth* solution. We can evaluate certain quantities (e.g., norms and inner products) with respect to u and $X_0^{\tilde{N}}$ in the off-line phase, but only at considerable cost, $O(\tilde{N})$, where $\tilde{N} = 10^6$–10^9.

To be explicit, assume that we have a nodal basis, $\{\phi_j(\mathbf{x})\}$ for the fine space $X_0^{\tilde{N}}$ satisfying $\phi_j(\mathbf{x}_i) = \delta_{ij}$ on nodal points $\mathbf{x}_i$, $i = 1,\ldots,\tilde{N}$. We'll also include an extended space, $X^{\tilde{N}}$ =span$\{\phi_j(\mathbf{x})\}$ for $j = 1,\ldots,\tilde{N}_E$ that includes points on the boundary and will approximate $f(\mathbf{x})$ by its interpolant in this basis. We define the fine-scale stiffness and mass matrices,

$$A_{ij} \;:=\; \mu \int_\Omega \nabla\phi_i \cdot \nabla\phi_j\, dV \tag{747}$$

$$M_{ij} \;:=\; \int_\Omega \phi_i\, \phi_j\, dV, \tag{748}$$

$$\bar{M}_{ij} \;:=\; \int_\Omega \phi_i\, \phi_j\, dV, \quad i = 1,\ldots,\tilde{N},\;\; j = 1,\ldots,\tilde{N}_E. \tag{749}$$

The truth solution is

$$u(\mathbf{x}) \;=\; \sum_{j=1}^{\tilde{N}} \phi_j(\mathbf{x}) u_j \tag{750}$$

with the basis coefficient vector $\underline{u} = [u_1\, u_2\, \ldots\, u_{\tilde{N}}]^T$ satisfying

$$A\underline{u} = \bar{M}\underline{\bar{f}}. \tag{751}$$

Here, $\underline{\bar{f}}$ is set of basis coefficients, $f(\mathbf{x}_j)$ $j = 1,\ldots,\tilde{N}_E$.

Now, we consider a coarse (RB) solution, $u_c(\mathbf{x}) \in X_0^N \subset X_0^{\tilde{N}}$ with basis functions $\xi_n(\mathbf{x})$, $n = 1,\ldots,N$ and $N \ll \tilde{N}$. The coarse-grid problem is *Find* $u_c \in X_0^N \subset \mathcal{H}_0^1(\Omega)$ *such that, for all* $v \in X_0^N$,

$$a(v,u_c) \;:=\; \int_\Omega \nabla v \cdot \nabla u_c\, dV \;=\; \int_\Omega v\, f\, dV \;=\; (v,f). \tag{752}$$

Let $B = [\underline{b}_1\, \underline{b}_2\, \ldots\, \underline{b}_N]$ be an $\tilde{N} \times N$ matrix whose columns represent the coarse-space basis functions evaluated on the fine-space gridpoints,

$$B_{ij} \;=\; \xi_j(\mathbf{x}_i). \tag{753}$$

Let $\hat{\underline{u}}_c \in \mathbb{R}^N$ be the vector of unknown coarse-space basis coefficients and $\underline{u}_c = B\hat{\underline{u}}_c$ be the discrete nodal values corresponding to $u_c(\mathbf{x}) \in X_0^N$. Computation of $\underline{u}_c$ is a straightforward Galerkin projection,

$$\underline{u}_c \;=\; BA_c^{-1}B^T\bar{M}\bar{\underline{f}}, \tag{754}$$

with $A_c := B^TAB$.

The continuous and equivalent discrete representations of the error in $X_0^{\tilde{N}}$ are

$$e_c(\mathbf{x}) \;:=\; u(\mathbf{x}) \;-\; u_c(\mathbf{x}) \tag{755}$$

$$\underline{e}_c \;:=\; \underline{u} \;-\; \underline{u}_c. \tag{756}$$

Here, we are defining the error with respect to the truth solution, $u(\mathbf{x})$. In the following, we seek to estimate the norm of this error without explicit computation of u.

To simplify the exposition, we will assume that $f(\mathbf{x})$ vanishes on the domain boundary, $\partial\Omega$. This will allow us to work with vectors $\underline{f}$ that are in $\mathbb{R}^{\tilde{N}}$, rather than having to introduce prolongation matrices that bring data to the domain boundaries. We note, however, that such prolongation operators and careful tracking of domain-boundary values is essential for many quantities of interest (e.g., the Nusselt number in convective heat transfer). With this simplification, we recast (751) as $A\underline{u} = M\underline{f}$. With a similar restriction for the reduced problem we have the following equations for the residuals,

$$\underline{r}_c \;=\; \underline{f} - M^{-1}A\underline{u}_c \;\neq\; 0 \tag{757}$$

$$\underline{0} \;=\; \underline{f} - M^{-1}A\underline{u}. \tag{758}$$

Taking the difference of these two, we have

$$\underline{r}_c \;=\; M^{-1}A\,(\underline{u} \;-\; \underline{u}_c) \tag{759}$$

$$\;=\; M^{-1}A\underline{e}_c \tag{760}$$

Note that $\underline{r}_c$ is computable (without inversion of A) from (757). We next consider the A- (or energy-) norm of e_c,

$$\|e_c\|_a^2 \;=\; a(e_c,e_c) \;=\; \underline{e}_c^TA\underline{e}_c \;=\; \underline{r}_c^TM^TA^{-1}M\underline{r}_c \;=\; \underline{r}_c^TM^T\underline{e}_c \;=\; (r_c,e_c). \tag{761}$$

We'd like to estimate this quantity without inverting A.

We can establish an error bound based on the residual starting with Poincaré and Cauchy-Schwarz,

$$\alpha\,\|e_c\|_2^2 \;\leq\; a(e_c,e_c) \;=\; (r_c,e_c) \;\leq\; \|e_c\|_2\,\|r_c\|_2. \tag{762}$$

$$\|e_c\|_2 \;\leq\; \frac{1}{\alpha}\|r_c\|_2 \tag{763}$$

$$\;=\; \frac{1}{\alpha}\left[\underline{r}_c^TM\underline{r}_c\right]^{\frac{1}{2}} \tag{764}$$

$$\;=\; \frac{1}{\alpha}\left[\left(\underline{f} - M^{-1}AB\underline{u}_c\right)^T M\left(\underline{f} - M^{-1}AB\underline{u}_c\right)\right]^{\frac{1}{2}} \tag{765}$$

$$\;=\; \frac{1}{\alpha}\left[\bar{\underline{u}}_c^TG\bar{\underline{u}}_c\right]^{\frac{1}{2}}. \tag{766}$$

Here, we define $G = R^T M R$, with

$$R := \left[\underline{f}\ \underline{q}_1\ \underline{q}_2 \dots \underline{q}_N\right] \tag{767}$$

$$\underline{q}_j := M^{-1} A \underline{b}_j \tag{768}$$

$$\underline{\bar{u}}_c := [1\ -u_{c,1}\ -u_{c,2}\ \dots\ -u_{c,N}]^T . \tag{769}$$

18.1 Tighter Bounds in the Energy Norm

A weakness of (762) is that the 2-norm of r_c is not necessarily bound (e.g, if f is the Dirac-delta function). We can get a sharper estimate if we seek to bound e_c in the $|\cdot|_a$ (or $A-$) norm. We'll illustrate the point with discrete inner products, where convenient, exploiting the fact that there exists an SPD matrix $A^{\frac{1}{2}}$ given that A is SPD. From (762), we have

$$|e_c|_a^2 \;=\; a(e_c,e_c) \;=\; (e_c,r_c) \;=\; \underbrace{\underline{e}_c^T A^{\frac{1}{2}}}_{\underline{v}^T}\underbrace{A^{-\frac{1}{2}}\underline{r}_c}_{\underline{w}} \;=\; \|e_c\|_A\,\|r_c\|_{A^{-1}} \;=\; |e_c|_a\,\|r_c\|_{A^{-1}}, \quad (770)$$

where we have used Cauchy-Schwarz (here, with equality) to split the norms. Dividing both sides by $|e_c|_a$ yields the desired result

$$|e_c|_a \;=\; \|r_c\|_{A^{-1}}, \quad (771)$$

which in this case is somewhat obvious. In the next section, we will use the same approach to generate bounds for a problem that requires (accessible) estimates, where we will generally have an *inequality*.

19 *A Posteriori* Error Bounds for pMOR

Here, we are interested in estimating the error for a solution that is computed (inexpensively, i.e., online) using a parameterized ROM. For purposes of illustration, we consider the following generalized Poisson problem, *Find* $u \in \mathbb{W}(\Omega)$ *such that, for all* $v \in \mathbb{W}$*,*

$$\int_\Omega (1+\mu\psi(\mathbf{x}))\,\nabla v\cdot\nabla u\,dV \;=\; \int_\Omega v\,f\,dV. \quad (772)$$

Here, $\psi(\mathbf{x}) \in (0,1]$ on Ω is a positive weight function and $\mu \in (-1,\infty) =: \mathcal{D}$ is the free parameter. Note that this configuration is a superset of the case where one has a jump in diffusivity between two subsets of Ω. The trial/test space $\mathbb{W} \subset \mathcal{H}_0^1$ is intentionally ambiguous in (772) as the same statement applies for the continuous problem, the truth (FOM) problem, and the reduced-order model.

For the FOM case, one proceeds in the usual way of forming the $\mathcal{N}\times\mathcal{N}$ stiffness matrix A_μ with

$$A_{\mu,ij} \;=\; \int_\Omega (1+\mu\psi(\mathbf{x}))\,\nabla\phi_i\cdot\nabla\phi_j\,dV \quad (773)$$

and solves the system $A_\mu\underline{u} = M\underline{f}$ for the fine-scale solution $\underline{u}(\mu)$.

For pMOR, we wish to generate solutions, $\underline{u}_c(\mu)$, *and* error estimates, ϵ_μ, for a large number ($\mathcal{M} \gg 1$) of values of μ without incurring $O(\mathcal{N})$ work for each case. To do so, we split the A_μ operator into two terms,[15] $A_\mu = A_0 + \mu A_1$, with

$$A_{0,ij} \;=\; \int_\Omega \nabla\phi_i\cdot\nabla\phi_j\,dV \quad (774)$$

$$A_{1,ij} \;=\; \int_\Omega \psi(\mathbf{x})\nabla\phi_i\cdot\nabla\phi_j\,dV. \quad (775)$$

[15] We remark that the existence of such an affine decomposition of the parameter dependence is a central element of pMOR with certifiable error estimators, but it is not absolutely essential. Other approaches include the empirical interpolation method (EIM) of Barrault *et al.* [EIM04].

We then form the $N \times N$ coarse-grid matrices

$$A_{c,0} = B^T A_0 B \tag{776}$$
$$A_{c,1} = B^T A_1 B, \tag{777}$$

where we remark that $A_{c,*}$ ($*$=0 or 1) can be formed without explicit formation of A_* provided only that one has the ability to effect the matrix-vector product $A_* \underline{b}_j$ for each column of the basis vectors, $j = 1, \ldots, N$. For any value of $\mu \in \mathcal{D}$, the RB solution is given by

$$\underline{u}_c(\mu) = B A_c^{-1} B^T M \underline{f}, \tag{778}$$

with $A_c(\mu) := A_{c,0} + \mu A_{c,1}$. The (computable) residual is

$$\underline{r}_c(\mu) = M\underline{f} - A_\mu \underline{u}_c, \tag{779}$$

which we use to estimate a bound on $e_c(\mu) := u(\mu) - u_c(\mu)$.

We start with the A-norm and seek a coercivity constant $\alpha_{LB}(\mu)$ such that, for all $v \in X^{\mathcal{N}}$,

$$\alpha_{LB}(\mu)|v|_a^2 = \alpha_{LB}(\mu)\, a(v,v) \le a_\mu(v,v), \tag{780}$$

with $a(v,v) = \underline{v}^T A_0 \underline{v}$ and $a_\mu(v,v) = \underline{v}^T A_\mu \underline{v} = \underline{v}^T A_0 \underline{v} + \mu \underline{v}^T A_1 \underline{v}$. Dividing (780) by $a(v,v)$, the lower-bound expression simplifies to

$$\alpha_{LB}(\mu) \le 1 + \mu \frac{\int_\Omega \psi \nabla v \cdot \nabla v \, dV}{\int_\Omega \nabla v \cdot \nabla v \, dV} \tag{781}$$

$$\le 1 + \mu \frac{\int_\Omega \psi\, q \, dV}{\int_\Omega q \, dV}, \quad q(\mathbf{x}) > 0. \tag{782}$$

Under the assumption that $0 < \psi(\mathbf{x}) < 1$, there are two cases to consider,

$$\mu \in (-1, 0], \qquad \alpha_{LB} = 1 + \mu \max_{\mathbf{x}} \psi(\mathbf{x}) \tag{783}$$

$$\mu > 0, \qquad \alpha_{LB} = 1 + \mu \min_{\mathbf{x}} \psi(\mathbf{x}). \tag{784}$$

From (780), we have

$$a(e_c, e_c) \le \frac{1}{\alpha_{LB}(\mu)} a_\mu(e_c, e_c) \tag{785}$$

$$\le \frac{1}{\alpha_{LB}(\mu)} \underbrace{\underline{e}_c^T A_0^{\frac{1}{2}}}_{\underline{v}^T} \underbrace{A_0^{-\frac{1}{2}} \underline{r}_c}_{\underline{w}}. \tag{786}$$

Applying Cauchy-Schwarz yields,

$$|e_c|_a \le \frac{1}{\alpha_{LB}(\mu)} \|\underline{r}_c\|_{A_0^{-1}}, \tag{787}$$

which is the desired bound.

We now show that $\|\underline{r}_c(\mu)\|_{A_0^{-1}}$ can be computed in $O(N^2)$ operations for any $\mu \in \mathcal{D}$. In a preprocessing (offline) phase, compute

$$R := \left[M\underline{f} \;\; A_0 B \;\; A_1 B \right] \tag{788}$$

$$(A_0^{-1} R) := \left[A_0^{-1} M \underline{f} \;\; B \;\; A_0^{-1} A_1 B \right] \tag{789}$$

$$G := R^T A_0^{-1} R, \tag{790}$$

which requires solving $N+1$ systems in A_0. As in the previous section, we have

$$\|\underline{r}_c\|^2 \;=\; \bar{\underline{u}}_c^T \left(R^T A_0^{-1} R\right) \bar{\underline{u}}_c. \tag{791}$$

The magic happenc because $\bar{\underline{u}}_c$ is of *low rank*,

$$\bar{\underline{u}}_c \;:=\; \left[1,\, -\hat{\underline{u}}_c^T,\, -\mu \hat{\underline{u}}_c^T\right]^T, \tag{792}$$

with $\hat{\underline{u}}_c \in \mathbb{R}^N$ the vector of coarse-space basis coefficients. Thus, in the on-line phase, one can compute the error bound,

$$\epsilon_a \;=\; \frac{\|\underline{r}_c(\mu)\|_{A_0^{-1}}}{\alpha_{LB}(\mu)} \tag{793}$$

$$\;=\; \frac{\left[\bar{\underline{u}}_c^T G \bar{\underline{u}}_c\right]^{\frac{1}{2}}}{\alpha_{LB}(\mu)} \tag{794}$$

Note that this bound depends on μ directly through the μ dependence in (792) and $\alpha_{LB}(\mu)$, and indirectly through the dependence of u_c on μ. G, is μ-independent. Its size scales, roughly, as N times the number of free parameters. (In this example we have only one, μ.)

19.1 *A Posteriori* Error Bounds Example

We consider (772) with $\Omega = [0,1]^2$, unit source ($f \equiv 1$), homogeneous Dirichlet conditions on $\partial\Omega$, and

$$\psi(\mathbf{x}) \;=\; \left[\cos(\pi x)\, \cos(\pi y)\right]^8. \tag{795}$$

We take as the coarse-space basis,

$$\phi_i(\mathbf{x}) \;=\; \sin(k\pi x)\, \sin(l\pi y), \quad k,l \in \{1,\ldots,N_b\}^2, \tag{796}$$

with $i = k + (l-1)N_b$. We choose $N_b = 3$, which implies a coarse-space dimension of $N = 9$. The results are completely driven from a Nek5000 user file.

The coarse-space and truth solutions are plotted in Fig. 17 along with the error for $\mu = -0.8$. Also shown is the behavior of the error and the error estimate as a function of μ.

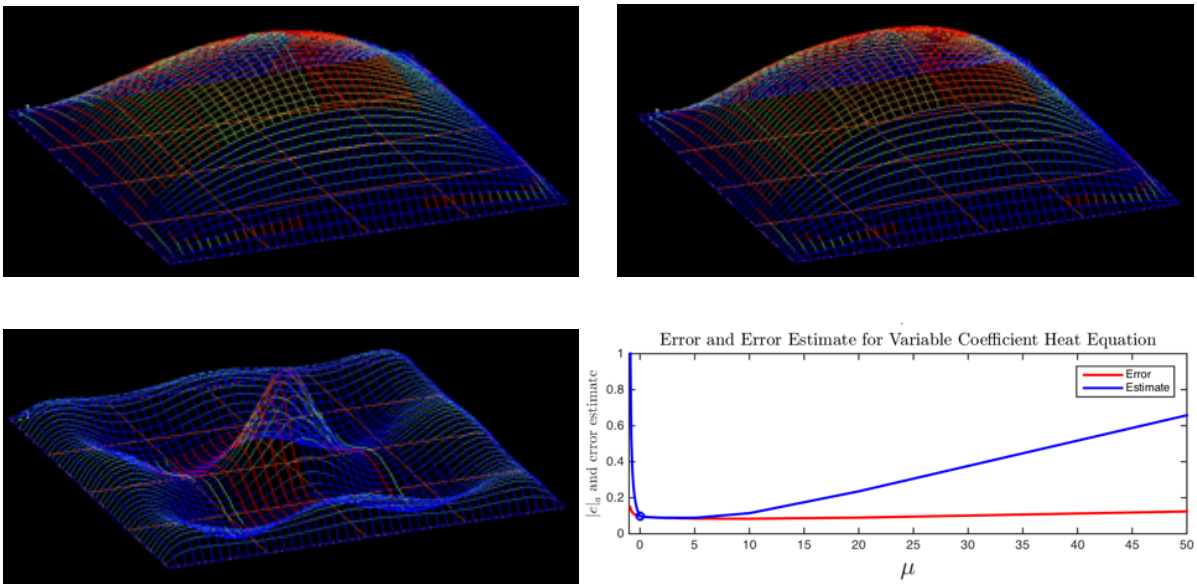


Figure 17: ROM (top left) and FOM truth solution (top right) for test problem defined by (772) and (795) with $\mu = -.8$. The truth solution is solved using a 4×4 array of spectral elements of order $p = 11$. The ROM error is shown lower left. The errors and error estimates (793) for $\mu \in [-.9 : 50]$ are plotted in the lower right.