Section 2
Discrete Fourier analysis, basic linear PDE

February 20, 2018

1 Motivation

New issues arise in computing time dependent fields that are governed by partial differential equations. *Semi-discretization* is a two step approach to doing this. The first step is to "discretize in space", which means to approximate the partial differential equation by a large system of ordinary differential equations. The second step is "discretize in time", which means to solve the ordinary differential equation system by some time stepping method. The partial differential equation has the form

$$\partial_t u = Lu \tag{1}$$

where L is a linear differential operator in space (terms defined below). The semi-discrete approximation is a large system of ordinary differential equations

$$\dot{U} = AU \ . \tag{2}$$

Here, $U(t)$ is a finite dimensional approximation to the field $u(\cdot, t)$ at the same time. It is called semi-discrete because the continuous space variable x, as in $u(x, t)$ has been replaced by a discrete index j, as in $U_j(t)$. The time variable is still continuous.

The semi-discrete system (2) is then solved by discretizing in time, possibly using a Runge Kutta method. This system is likely to be *stiff* (defined below). This means that it matters what an ODE solver will do when the time step is too large to solve the ODE accurately. There is a sense in which Δt does not go to zero (explained below). Partial differential equations lead to stiff systems of ordinary differential equations.

The *discrete Fourier transform*, or *DFT*, allows us to analyze some of the semi-discrete systems (2) that arise from simple field equations (1). The DFT can be a theoretical tool that explains the ODE system (2). It also can be a practical tool that is used to define the semi-discrete approximation itself. Discretization methods that use the DFT are called *spectral methods*. They have the advantage of being *spectrally accurate* – which means very very accurate (definition below). They have the disadvantage of being more complicated, which we would put up with, and applying to problems only in certain geometries and boundary conditions (examples below). The *FFT*, *(fast Fourier transform)* is a deep and clever set of algorithms that calculate the DFT fast enough to make it practical for large scale computing.

The plan for this Section of notes is to describe the algebra of the DFT (Section 2), then explain the relation between the DFT and Fourier series (Section 3). Since "everybody knows"[1] how useful Fourier series are for differential equations, this should suggest that discrete Fourier series (the DFT) are useful for discretizations of partial differential equations. It also starts to explain the

[1] Malvin Kalos (innovator in Monte Carlo methods) started a lecture with: "The following is well known to those who know it well."

phenomenon of spectral accuracy. Section 4 describes some simple partial differential equations for time dependent fields, the linear homogeneous heat and wave equations. Section 5 derives semi-discrete approximations to these and shows that the DFT determines the eigenvalues and eigenvectors of the corresponding A in (2). This is how DFT methods got to be called spectral – they are related to the spectrum (eigenvalues) of A. Section 6 gives examples of the DFT being used to derive spectrally accurate semi-discrete approximations.

A *field* is a function of x. For example, if $u(x)$ is a temperature field, then $u(x)$ is the temperature at the point x. This would be a scalar field because there is just one number at each point. An electric field is a vector at each point in space $E(x) = (E_1(x), E_2(x), E_3(x))$ (three components for three dimensions). This terminology is often used in physics. Engineers sometimes call a field a *distributed parameter*. There is a parameter $u(x)$ "distributed" to every point x. A finite dimensional vector $u(x) \in \mathbb{R}^n$ is a *lumped parameter* because all the variation has been "lumped" into n numbers.

(Notation: we use capital letters U, V, etc., to represent discrete "functions" if a discrete index j (sequences). We use lower case letters u, v, etc., for functions of a continuos variable x. You can get a sequence by sampling a function of x, as in $U_j = u(x_j)$.)

2 The DFT, discrete Fourier transform

A sequence U_j is *periodic* with period n if, for all j,

$$U_{j+n} = U_j \ .$$

The vector space of periodic sequences with period n has dimension n. We write $U \in \mathbb{R}^n$ (for real sequences) or $U \in \mathbb{C}^n$ (for complex sequences) if U is periodic with period n. For any integer α, the *discrete Fourier mode* with *wave number* α will be written $V_\alpha \in \mathbb{C}^n$. Its entries are

$$V_{\alpha,j} = e^{2\pi i \alpha j/n} \ . \tag{3}$$

Here, $V_{\alpha,j}$ is component j of the vector $V_\alpha \in \mathbb{C}^n$. In Part 1, it was the derivative of V_α with respect to x_j.

Theorem. The modes V_α, for $\alpha = 0, \ldots, n-1$ are an orthogonal basis of $\mathbb{C}^n$.

Proof. The l^2 inner product in $\mathbb{C}^n$ is[2]

$$\langle F, G \rangle = \sum_{j=0}^{n-1} \overline{F}_j G_j \ . \tag{4}$$

We calculate that if α and β are in the range given, then

$$\langle V_\alpha, V_\alpha \rangle = n \tag{5}$$

$$\langle V_\alpha, V_\beta \rangle = 0 \quad \text{if } \alpha \neq \beta \ . \tag{6}$$

[2]Here, $\overline{z}$ is the complex conjugate of the complex number z. Other treatments may use $F_j \overline{G}_j$. They may have normalization factors of $\frac{1}{n}$ or $\frac{1}{\sqrt{n}}$.

The first formula is verified as follows:

$$\begin{aligned}\langle V_\alpha, V_\alpha \rangle &= \sum_{j=0}^{n-1} \overline{V_{\alpha,j}}\, V_{\alpha,j} \\ &= \sum_{j=0}^{n-1} \overline{e^{2\pi i \alpha j/n}}\, e^{2\pi i \alpha j/n} \\ &= \sum_{j=0}^{n-1} e^{-2\pi i \alpha j/n}\, e^{2\pi i \alpha j/n} \\ &= \sum_{j=0}^{n-1} 1 \ .\end{aligned}$$

The second formula is verified by recognizing the sum as a geometric series. If z is any complex number (except $z = 1$), then[3]

$$\sum_{j=0}^{n-1} z^j = \frac{z^n - 1}{z - 1} \ . \tag{7}$$

The left side of (6) is

$$\begin{aligned}\sum_{j=0}^{n-1} \overline{e^{2\pi i \alpha j/n}} e^{2\pi i \beta j/n} &= \sum_{j=0}^{n-1} e^{-2\pi i \alpha j/n} e^{2\pi i \beta j/n} \\ &= \sum_{j=0}^{n-1} e^{[2\pi i(\beta-\alpha)/n] j} \ .\end{aligned}$$

This is a geometric series with

$$z = e^{2\pi i(\beta-\alpha)/n} \ .$$

The geometric series formula (7) applies if $z \neq 1$. The numerator is (recall that α and β are integers)

$$z^n - 1 = e^{2\pi i(\beta-\alpha)} - 1 = 0 \ .$$

The main point is to check that $z \neq 1$. This is the same as saying that $\beta - \alpha$ is not an integer multiple of n. If $\alpha \neq \beta$ then $\beta - \alpha \neq 0$. If $0 \leq \alpha < d$ and $0 \leq \beta < n$, then $|\beta - \alpha| < n$. This proves that $\beta - \alpha$ is not a multiple of n, which finishes the proof of the theorem.

The *Kronecker delta* symbol is

$$\delta_{\beta\alpha} = \begin{cases} 1 & \text{if } \alpha = \beta \\ 0 & \text{if } \alpha \neq \beta \ . \end{cases}$$

[3]You can verify this by multiplying both sides by $z - 1$ and checking that $(z-1)(1 + z + \cdots + z^{n-1}) = z^n - 1$.

A single formula using the Kronecker delta expresses the two relations (5) and (6):

$$\langle V_\beta, V_\alpha \rangle = n\delta_{\beta\alpha} \ . \tag{8}$$

The numbers $\delta_{\beta\alpha}$ are the entries in the identify matrix. Sums involving the delta symbol simplify:

$$\sum_{\alpha=0}^{n-1} \delta_{\beta\alpha} X_\alpha = X_\beta \ .$$

This is the component-wise expression of the matrix/vector formula $IX = X$. The delta symbol is a convenient way to do some calculations. The formula (8) is always true if α or β are outside the range $0, \ldots, n-1$.

The definition (3) defines a discrete Fourier mode for any integer α, and yet a basis of $\mathbb{C}^n$ can have only n vectors. This apparent paradox illustrates *aliasing*, that the same discrete Fourier mode can have more than one label α. For a person, an "alias" is a different name he/she might use. For a Fourier mode, it is a different label $\beta \neq \alpha$ so that $V_\beta = V_\alpha$. It is "easy to check" (hints below) that $\alpha \neq \beta$ determine the same Fourier mode if and only if α and β differ by a multiple of n:

$$V_\beta = V_\alpha \quad \Longleftrightarrow \quad \beta = \alpha + kn \ \text{ for some integer } k \ . \tag{9}$$

Here's the proof. The $\Longleftarrow$ part (which is $\beta = \alpha + kn \Longrightarrow V_\alpha = V_\beta$) is easier. If j and α and k are integers, then

$$\begin{aligned} V_{\beta,j} &= e^{2\pi i(\alpha+kn)j/n} \\ &= e^{2\pi i\alpha j/n} e^{2\pi iknj/n} \\ &= e^{2\pi i\alpha j/n} \qquad (\text{because } e^{2\pi iknj/n} = e^{2\pi ikj} = 1) \\ &= V_{\alpha,j} \ . \end{aligned}$$

The other part is that if β and α are integers and $\beta \neq \alpha + kn$ for some integer k, then $V_\beta \neq V_\alpha$. The calculation above shows that if $e^{2\pi i(\beta-\alpha)/n} \neq 1$ then $\langle V_\alpha, v_\beta \rangle = 0$. Since $V_\alpha \neq 0$, this means that $V_\alpha \neq V_\beta$.

The modes (3) are the discrete Fourier modes. The n modes with $\alpha = 0, \ldots, n-1$ form the *Fourier basis* of $\mathbb{C}^n$. If V_α is any collection of n orthogonal (non-zero) vectors, and if U is any sequence in $\mathbb{C}^n$, then u may be represented (*expanded*) as

$$U = \sum_{\alpha=0}^{n-1} a_\alpha V_\alpha \ .$$

The expansion coefficients are given by

$$a_\alpha = \frac{\langle V_\alpha, V \rangle}{\langle V_\alpha, V_\alpha \rangle} \ .$$

When the vectors V_α are the Fourier basis, the expansion coefficients are called *Fourier coefficients*. We summarize these facts and formulas in the *Fourier representation theorem*:

Theorem. For any $U \in \mathbb{C}^n$, we have

$$U = \sum_{\alpha=0}^{n-1} \widehat{U}_\alpha V_\alpha \ , \tag{10}$$

where

$$\widehat{U}_\alpha = \frac{1}{n} \sum_{j=0}^{n-1} e^{-2\pi i \alpha j/n} U_j \ . \tag{11}$$

The Fourier representation formula (10) may be written more concretely as

$$U_j = \sum_{\alpha=0}^{n-1} \widehat{U}_\alpha \, e^{2\pi i \alpha j/n} \ . \tag{12}$$

We assumed that u_j is a periodic function of j. The Fourier modes on the right are also periodic functions of α with period n. You can see this from the sum formula defining $\widehat{U}_\alpha$, or you can see it from the fact that $V_{\alpha+n} = V_\alpha$, so $\langle V_{\alpha+n}, U\rangle = \langle V_\alpha, U\rangle$. This means that the numbers $\widehat{U}_\alpha$, being a periodic function of α, are the components of a vector $\widehat{U} \in \mathbb{C}^n$.

The *Parseval relation* gives the norm of $U \in \mathbb{C}^n$ in terms of the Fourier coefficient vector $\widehat{U}$. It is the basis of the crucial *von Neumann* stability analysis for PDE that is covered in later parts of this course. The norm is

$$\begin{aligned} \|U\|_{l^2}^2 &= \sum_{j=0}^{n-1} |U_j|^2 \\ &= \sum_{j=0}^{n-1} \overline{U}_j u_j \\ &= \langle U, U\rangle \ . \end{aligned}$$

The formula is:

Theorem. For any $U \in \mathbb{C}^n$,

$$\|U\|_{l^2} = \sqrt{n} \left\| \widehat{U} \right\|_{l^2} \ . \tag{13}$$

Proof. This is consequence of the orthogonality relations (8). Here is a verification that is direct, simple-minded, and possibly inelegant. Start with the Fourier representation (10) and calculate. A trick is to write (10) also as

$$U = \sum_{\beta=0}^{n-1} \widehat{U}_\beta V_\beta \ .$$

This allows calculations that use the Kronecker delta version of the orthogonality relation (8). First the norm is written in terms of the inner product. Then u is written as its Fourier sum. Then the coefficients $\widehat{U}_\beta$ and $\widehat{U}_\alpha$ are pulled out of the the inner product. Recall here that you take the complex conjugate of the coefficient of V_β in the complex inner product.

$$\begin{aligned}
\|U\|_{l^2}^2 &= \langle U, U \rangle \\
&= \langle \sum_{\beta=0}^{n-1} \widehat{U}_\beta V_\beta, \sum_{\alpha=0}^{n-1} \widehat{U}_\alpha V_\alpha \rangle \\
&= \sum_{\beta=0}^{n-1} \sum_{\alpha=0}^{n-1} \overline{\widehat{U}_\beta}\, \widehat{U}_\alpha \, \langle V_\beta, V_\alpha \rangle \\
&= \sum_{\beta=0}^{n-1} \sum_{\alpha=0}^{n-1} \overline{\widehat{U}_\beta}\, \widehat{U}_\alpha \, n \, \delta_{\beta\alpha} \\
&= n \sum_{\beta=0}^{n-1} \left| \widehat{U}_\beta \right|^2
\end{aligned}$$

This finishes the proof of the Parseval relation (13).

The formula (11) is the *discrete Fourier transform*, or *DFT*. The sequence $\widehat{u}_\alpha$ is the discrete Fourier transform of the sequence u_j. The Fourier representation formula (12) is also called the Fourier *inversion* formula because it reverses the Fourier transform and gives u in terms of $\widehat{u}$. The discrete Fourier transform is a linear transformation from $\mathbb{C}^n$ to $\mathbb{C}^n$. The matrix of this transformation is the $n \times n$ DFT matrix F with entries

$$F_{\alpha j} = e^{-2\pi i \alpha j / n} \ .$$

You can check that (11) is equivalent to

$$\widehat{U} = FU \ .$$

The calculations leading to the orthogonality relations (8) may be stated in matrix terms as

$$F^* F = n \, I \ .$$

Thus, F is "almost" a unitary matrix. The normalization

$$\widetilde{F} = \frac{1}{\sqrt{n}} F$$

defines a matrix that is exactly unitary: $\widetilde{F}^* \widetilde{F} = I$. If the DFT had been defined with the $\sqrt{n}$ normalization:

$$\widetilde{U} = \widetilde{F} U \ ,$$

then the Parseval relation would have been

$$\left\|\widetilde{U}\right\|_{l^2} = \|U\|_{l^2} \ .$$

The direct and inverse DFT must have a factor of n somewhere, or two factors of $\sqrt{n}$. Different conventions (F or $\widetilde{F}$) put these factors in different places.

A matrix like $\widetilde{F}$ that preserves the l^2 norm is called *unitary*. It has the property that

$$\widetilde{F}^* \widetilde{F} = \widetilde{F}\widetilde{F}^* = I \ .$$

If A is a complex valued matrix, then A^* is the *adjoint*, or *conjugate transpose* – you take the transpose (switch the indices) and the complex conjugate of the entries. The entries are

$$\left(A^*\right)_{\alpha\beta} = \overline{A}_{\beta\alpha} \ .$$

Here is the $\widetilde{F}\widetilde{F}^* = I$ equation calculated out in indices. We use the summation convention over γ and the Kronecker delta symbol to represent the entries of the identity matrix. For any square matrices A, B, and C with $C = AB$, the entries are

$$C_{\alpha\beta} = A_{\alpha\gamma} B_{\gamma\beta} \ .$$

The last step here is discrete Fourier mode orthogonality calculation we did before.

$$\begin{aligned}
\left(\widetilde{F}^*\widetilde{F}\right)^*_{\alpha\beta} &= \left(\widetilde{F}\right)^*_{\alpha\gamma} \widetilde{F}_{\gamma\beta} \\
&= \overline{\widetilde{F}}_{\gamma\alpha} \widetilde{F}_{\gamma\beta} \\
&= \frac{1}{n} \sum_{\gamma=0}^{n-1} e^{2\pi i \gamma\alpha/n} e^{-2\pi i \gamma\beta/n} \\
&= \delta_{\alpha\beta}
\end{aligned}$$

A *fundamental set* of mode numbers is a set of α values so that the corresponding modes v_α form a basis of $\mathbb{C}^n$. In view of aliasing, a set $\mathcal{S}$ is a fundamental set in this sense if for every integer β there is an $\alpha \in \mathcal{S}$ so that V_β is an alias of V_α. That means, there is an $\alpha \in \mathcal{S}$ so that $\alpha = \beta + kn$ for some integer k. Up to now, we have been using the fundamental set

$$\mathcal{S}_p = \{0, 1, \ldots, n-1\} \ . \tag{14}$$

(The "p" is for "programming".) This leads to simple formulas and simple programs. The Python FFT routines use this fundamental set.

But for analysis and for solving differential equations it can be better to use a fundamental set that is symmetric or nearly symmetric about $\alpha = 0$. If n is odd, we may write $n = 2m + 1$ and take

$$\mathcal{S}_a = \{-m, -m+1, \ldots, -1, 0, 1, \ldots, m\} \ . \tag{15}$$

(The "a" is for "analysis".) The n indices in this set are all distinct in the sense that none of them aliases to another. If n is even, we may write $n = 2m$ and take

$$\mathcal{S}_a = \{-m+1, -m+2, \ldots, 0, \ldots, m\} \ . \tag{16}$$

The discrete Fourier representation formula is one of the two

$$\left.\begin{aligned} U_j &= \sum_{\alpha \in \mathcal{S}_a} V_\alpha \widehat{U}_\alpha = \sum_{-m}^{m} v_\alpha \widehat{U}_\alpha \qquad m = 2n+1 \text{ odd} \\ U_j &= \sum_{\alpha \in \mathcal{S}_a} V_\alpha \widehat{U}_\alpha = \sum_{-m+1}^{m} V_\alpha \widehat{U}_\alpha \qquad m = 2n \text{ even} \ . \end{aligned}\right\} \tag{17}$$

3 Discrete and continuous Fourier modes

For computing, it is necessary to approximate a field u using a finite set of numbers U_j. This is similar to the *time discretization* process used in Part 1 of the course. The continuous trajectory $x(t)$ was approximated using $X_j \approx x(t_j)$. It is natural here (approximating a field) to take $U_j \approx u(x_j)$ (beware of the conflict of notation; either X_j is the approximate trajectory at the sample time t_j, or x_j is a sample point for the field). The accuracy of computing u sometimes depends on the accuracy with which the sample values U_j approximate u. Fourier series are one way to understand this issue.

The discrete Fourier transform is an operation on sequences with period n. Fourier series operate on periodic functions of x with period R. Period R means $u(x+R) = u(x)$ for all x. The continuous Fourier modes with period R and integer index α are

$$v_\alpha(x) = e^{2\pi i \alpha x/R} \ . \tag{18}$$

These are periodic, $v_\alpha(x+R) = v_\alpha(x)$. The inner product for periodic functions is

$$\langle u, v \rangle = \int_0^R \overline{u(x)}\, v(x)\, dx \ .$$

These modes are orthogonal in the sense that

$$\langle v_\alpha, v_\beta \rangle = R\, \delta_{\alpha\beta} \ .$$

Unlike the discrete formula (8), this formula holds for any integers α and β. There is no aliasing of continuous Fourier modes: $v_\alpha \neq v_\beta$ if $\alpha \neq \beta$.

The functions v_α are all orthogonal (see below) and linearly independent. This makes "the" vector space of periodic functions infinite dimensional. It is technical and outside the scope of this course to give a mathematically correct definition of infinite dimensional vector spaces of functions. There are many possibilities. We will use the one that, roughly speaking, is the space of functions u so that the integral (19) is defined (the discontinuities of u are not too bad) and finite (u is not too big). This vector space is called $L^2([0,R])$. The "L" is

for the French mathematician Lebesgue (pronounced "luh-beg", sort of). The 2 is for the numbers 2 in the norm formula

$$\|u\|_{L^2}^2 = \langle u, u \rangle = \int_0^R |u(x)|^2 \, dx \, . \tag{19}$$

A periodic function $u \in L^2([0, R])$ has a Fourier representation (in abstract or concrete form)

$$\left.\begin{aligned} u &= \sum_{\alpha=-\infty}^{\infty} \widehat{u}_\alpha \, v_\alpha \\ u(x) &= \sum_{\alpha=-\infty}^{\infty} \widehat{u}_\alpha \, e^{2\pi i \alpha x / R} \end{aligned}\right\} \tag{20}$$

The orthogonality relation leads to a formula for the Fourier coefficients

$$\widehat{u}_\alpha = \frac{1}{R} \langle v_\alpha, u \rangle = \frac{1}{R} \int_0^R e^{-2\pi i \alpha x / R} \, u(x) \, dx \, . \tag{21}$$

The series of coefficients $\widehat{u}_\alpha$ form the *Fourier series* corresponding to the periodic function u. The sum (20) is the Fourier series representation of u.

There is a mathematical issue with continuous Fourier series representations that does not arise with the discrete Fourier transform. The issue is *completeness*. The discrete version takes place in a vector space of dimension n. Any n non-zero orthogonal vectors form a basis. If there are n orthogonal $v_\alpha \in \mathbb{C}^n$, then the v_α are a basis, and any $u \in \mathbb{C}^n$ can be represented in terms of them. This isn't true in infinite dimensional spaces like $L^2([0, R])$. For example, if you leave out v_α for $\alpha < 0$ then you still have infinitely many v_α with $\alpha \geq 0$. But the infinitely many v_α with $\alpha \geq 0$ do not form a basis for $L^2([0, R])$. If $n = \infty$, having n orthogonal vectors does not imply that they are a basis. You also have to show that they are *complete* (form a basis). A mathematical book on Fourier series will have a proof of this. It is also possible to prove completeness of the Fourier basis v_α (for all integers α) using the completeness of the discrete Fourier basis V_α for $\alpha = 0, \ldots, n-1$.

3.1 Sampling and aliasing, interpolation and differentiation

Suppose $u(x)$ is a periodic function of period R. Suppose that x_j for $0 \leq j < n$ are evenly spaced grid points in the interval $[0, R]$. If Δx is the spacing between them, then

$$n \Delta x = R \, ,$$

so

$$\Delta x = \frac{R}{n} \, .$$

The grid point locations are

$$x_j = j \Delta x \, .$$

Let $U \in \mathbb{C}^n$ be the vector created by sampling u at the grid points. That means

$$U_j = u(x_j) \ .$$

There is a relationship between the DFT coefficients of U and the Fourier series representation of u.

We start by asking what is the sample vector corresponding to a continuous Fourier mode (18). Sampling produces the values

$$V_{\alpha,j} = e^{2\pi i \alpha x_j / R} \ .$$

If you plug in the definitions and "do the math", you find

$$V_{\alpha,j} = e^{2\pi i \alpha j / n} \ .$$

So, sampling the continuous Fourier mode number α gives the discrete mode (3) with the same α. In view of aliasing, this means that V_α and V_β, the sample values of v_α and v_β, are the same if β and α are aliases for the same discrete mode, (9).

Suppose u is any periodic function of x with Fourier series representation (20). Let $U \in \mathbb{C}^n$ be the sample values of u. Then a discrete Fourier coefficient of U is the sum of all the Fourier series coefficients of u that alias to it. We see this by writing the sampling operation applied to the Fourier series

$$\begin{aligned} U_j &= u(x_j) \\ &= \sum_{\beta=-\infty}^{\infty} \widehat{u}_\beta v_{\beta,j} \ . \end{aligned}$$

Suppose $\mathcal{S}$ is a fundamental set such as (14) or (15) or (16). Then for every β there is an $\alpha \in \mathcal{S}$ so that $\beta = \alpha + kn$ (k being an integer). By aliasing, $V_{\beta,j} = V_{\alpha,j}$. Therefore,

$$U_j = \sum_{\alpha \in \mathcal{S}} \left(\sum_{k=-\infty}^{\infty} \widehat{u}_{\alpha+kn} \right) V_{\alpha,j} \ .$$

This shows that the DFT coefficients of U are given by the *aliasing formula*.

$$\widehat{U}_\alpha = \sum_{k=-\infty}^{\infty} \widehat{u}_{\alpha+kn} \ . \tag{22}$$

This formula gives a way to understand the relationship between a function of the continuous variable x and its discrete approximation.

Choose a symmetric fundamental set (15) or (16). Suppose (we will show this often happens) that the Fourier coefficients $\widehat{u}_\beta$ are rapidly decreasing functions of β as $|\beta| \to \infty$. Then the *aliased* terms with $n \neq 0$ in the aliasing formula (22) are probably much smaller than the "un-aliased" term with $n = 0$. That is

$$\widehat{U}_\alpha \approx \widehat{u}_\alpha \quad \text{if } \alpha \in \mathcal{S}_a \ .$$

Also, the Fourier coefficients outside $\mathcal{S}_a$ are so small that

$$u(x) \approx \sum_{\alpha \in \mathcal{S}_a} \widehat{u}_\alpha e^{2\pi i \alpha x/R}$$

is a good approximation (the neglected terms $\alpha \notin \mathcal{S}_a$ are small).

Interpolation is a way of constructing a function from a collection of samples. The *interpolation* means that interpolating function is equal to the sample value at each sample point. There is linear interpolation (interpolation using a piecewise linear function), polynomial interpolation (interpolation using a polynomial), etc. *Fourier interpolation* is interpolation using a trigonometric polynomial. This trigonometric polynomial may be written

$$w_n(x) = \sum_{\alpha \in \mathcal{S}_a} \widehat{U}_\alpha e^{2\pi i \alpha x/R} \; . \tag{23}$$

This function interpolates u at the sample points x_j because of the calculation we just did:

$$w_n(x_j) = \sum_{\alpha \in \mathcal{S}_a} \widehat{U}_\alpha e^{2\pi i \alpha j/n} \; .$$

The function w_n in (23) is called a trigonometric polynomial because it is a finite sum of powers of the basic complex exponential:

$$e^{2\pi i \alpha x/R} = \left(e^{2\pi i x/R} \right)^\alpha \; .$$

These are trigonometric because

$$e^{2\pi i x/R} = \cos(2\pi x/R) + i \sin(2\pi x/R) \; .$$

Unlike normal polynomials, the sum (23) involves both positive and negative powers, $\alpha > 0$ and $\alpha < 0$.

There are different definitions of the *degree* of a trigonometric polynomial. Some people say that a sum

$$\sum_{\alpha=-m}^{m} a_\alpha e^{2\pi i \alpha x/R} \tag{24}$$

is a trigonometric polynomial of degree m. But there are $2m+1$ coefficients, so others say it has degree $2m$. This is to be consistent with ordinary polynomials, where a polynomial of degree r has $r+1$ coefficients. If n (the number of interpolation points) if even, then the sum defining w_n is not symmetric. Instead it runs from $-m+1$ to m. You can remove as much ambiguity as possible by saying that w_n is an interpolating trigonometric polynomial with n terms that is as symmetric as possible.

Fourier interpolation can be extremely accurate when u is smooth (precise definitions below). This makes computational methods based on trigonometric polynomials desirable – you can capture the desired solution accurately with a

relatively small number of degrees of freedom, n. Depending on the problem, Fourier methods might not be the best. They are not efficient for non-smooth functions (examples below). They are hard to apply in any geometry that isn't very simple.

Figure 1 shows interpolation of a very smooth function. The Gaussian is reproduced accurately using just $n = 8$ modes. Feel free to download the code `FourierInterpolation.py` and try some other examples. In the figure, the interpolating polynomial $w_n(x)$ is evaluated at many more than $n_g = 400 \gg n = 8$ points. The code does this using an FFT of size n_g. You just set the discrete Fourier coefficients beyond the original n equal to zero and use the inverse FFT.

Figure 2 shows trigonometric interpolation of a function that is continuous but not differentiable. This function is not "smooth". With $n = 20$ modes the interpolation is inaccurate. Figure 3 shows a more extreme case where u has a jump discontinuity. With $n = 40$ points, the interpolation is poor. It is poor even rather far from the discontinuity, which is called *pollution*. As with environmental pollution, the effect is felt far from the source.

The *spectral derivative*, or the *Fourier derivative* is the derivative of the interpolating polynomial. Suppose $U \in \mathbb{C}^n$ is a discrete vector, which may be

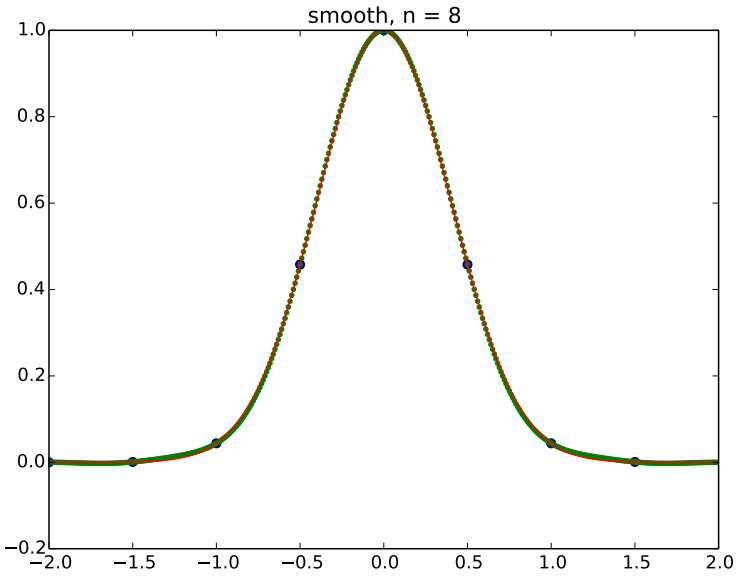

Figure 1: Fourier interpolation of a smooth function. The exact function is the solid line, the interpolant is small dots, interpolating points are large dots. The Python source is in `codes/FourierInterpolation.py`.

thought of as the samples of a function $u(x)$. Let $w(x)$ be the trigonometric interpolating polynomial. Define

$$V_j = \partial_x w(x_j) \ , \qquad \text{for } j = 0, \ldots, n-1. \tag{25}$$

This is the spectral derivative of U. The vector $V \in \mathbb{C}^n$ is the spectral derivative of U. The spectral differentiation operator is a linear map from $\mathbb{C}^n$ to $\mathbb{C}^n$. There it can be represented by an $n \times n$ complex matrix, D:

$$V = DU \ . \tag{26}$$

The matrix D is *dense* in the sense that $D_{jk} \neq 0$ for most (j,k); most of the entries are not zero. It is not hard to write a formula for D_{jk} that would allow you to compute the matrix vector product (26) directly. However (details below), it is more efficient to do the operation in *Fourier space*. That is,

$$U \overset{\text{FFT}}{\longrightarrow} \widehat{U} \overset{\text{diag mult}}{\longrightarrow} \widehat{D}\widehat{U} \overset{\text{inverse FFT}}{\longrightarrow} DU$$

The diagonal multiplication step is $\widehat{U}_\alpha \longrightarrow k_\alpha \widehat{U}_\alpha$. This means that in the Fourier domain, D is a diagonal matrix. The diagonal entry is ik_α, and is given by (28)

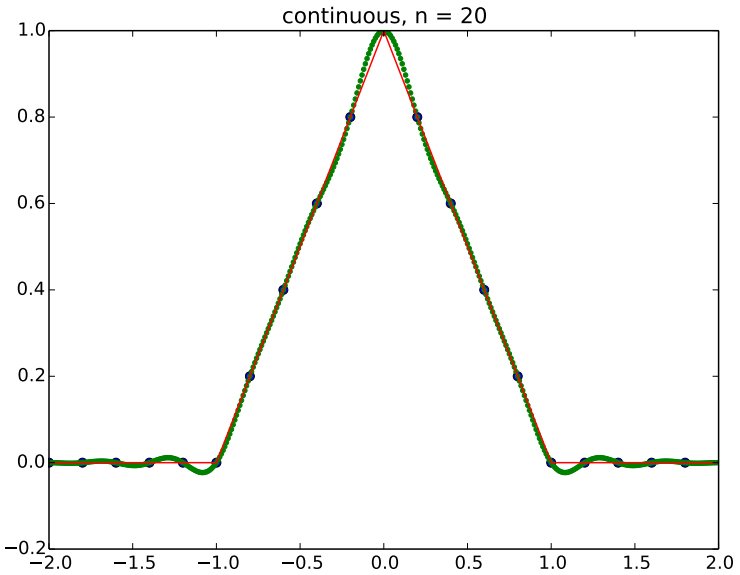

Figure 2: Fourier interpolation of a continuous but not smooth function. The exact function is the solid line, the interpolant is small dots, interpolating points are large dots. The Python source is in `codes/FourierInterpolation.py`.

below. The Fourier differentiation formula is (30). The cost of computing DU is the cost of two FFT operations, which is about $2 \times 2\log_2(n)$. The diagonal multiplication step is n multiplications, which is less. The matrix D is skew hermitian, which is

$$D^* = -D \; .$$

This is because its eigenvalues are ik_α which are pure imaginary and its eigenvectors V_α are orthogonal.

3.2 Wave number, feature size, resolution, artifacts

The function

$$w(x) = e^{ikx} \tag{27}$$

is a *plane wave* with *wave number* k. In higher dimensions, the corresponding formula has *wave fronts* that are planar. The modes $v_\alpha(x)$ are plane waves with wave number

$$k_\alpha = \frac{2\pi\alpha}{R} \; . \tag{28}$$

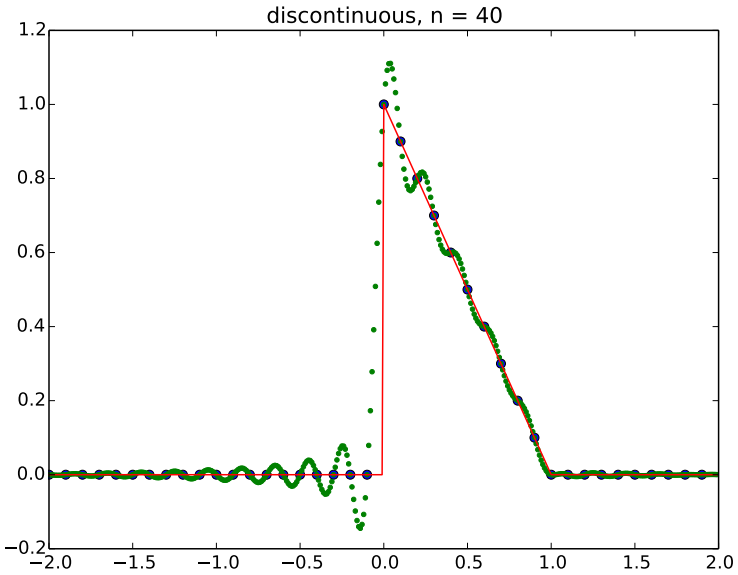

Figure 3: Fourier interpolation of a smooth function. The exact function is the solid line, the interpolant is small dots, interpolating points are large dots. The Python source is in `codes/FourierInterpolation.py`.

A plane wave is periodic with fundamental period

$$R_k = \frac{2\pi}{k} \ . \tag{29}$$

The family (18) consists of all plane waves that are periodic with period R. The wave number k has units

$$[k] = \frac{1}{\text{Length}} \ .$$

This allows the exponent in (27) to be dimensionless, and it is consistent with the period (29) being a length. Large wave number k corresponds to short wave length R_k.

The number α in (18) is a dimensionless integer that indexes the Fourier modes. For example, index $\alpha = 1$ gives the first non-constant mode in the positive wave number direction. The wave number corresponding to this mode depends on the size of the "box" (interval, period), R.

A crucial question for practical computation is how many wave numbers, what size n, is needed for a given computation. This can be determined by the minimum feature size in the solution. The notions of feature and feature size are vague but crucial. For example, a Gaussian "feature" might be a function of the form $e^{-(x/l)^2}$. The size of this feature is l (or $l/2$ or something). The function $x/\sqrt{l^2 + x^2}$ has a feature size l because it "transitions" from nearly -1 to nearly 1 over an interval of length l (or $2l$?).

You "resolve" a feature by including enough Fourier modes so that the feature can be clearly seen in a trigonometric polynomial of that degree. In order to resolve a feature, you need wave numbers appropriate for that feature size, which means $k \sim 1/l$. This is a physical requirement that does not depend on the size of the box R. We find the mode number needed to resolve a feature of size l by setting $R_k = l$ in (29) and solving for α. The result is

$$\alpha = \frac{R}{l} \ .$$

(This is a dimensionless ratio of two lengths.) The α needed is proportional to the size of the box and inversely proportional to the size of the feature. It is expensive (large n) to resolve a small feature in a big box. How do you estimate the n (number of modes in all) needed for a given calculation? It is at least 2α, because of positive and negative wave numbers. If you have a 20cm box and a feature of size .5cm, then you probably need at least $n = 2 \cdot 20/.5 = 80$ modes. This is likely to be an under-estimate because it just barely resolves the feature. To calculate at all accurately, you probably need a multiple of this, maybe a factor of 2 or 4 or more.

The DFT makes this resolution discussion is easy to understand. The DFT is a one to one correspondence between n terms in a Fourier series and the values of a function at n evenly spaced points. If the box has size R, the distance between points is R/n. This seems to be the smallest feature size that can be represented in this way. A feature is *resolved* or *under-resolved* depending on whether there are enough modes to resolve it.

Figures 4 and 5 illustrate resolution and under-resolution. With $n = 20$ modes, the feature is too small to be represented on the grid x_j and the trigonometric interpolant looks bad. With $n = 40$ modes, the interpolant is an accurate approximation to the function. Figure 4 illustrates the typical behavior of under-resolved features – overshoots, oscillations, and pollution. The actual function is always positive, but the interpolant overshoots zero to a value a little just below $-.1$. Overshoots are sometimes called *Gibbs' phenomenon*. The interpolant overshoots in an oscillatory way, going negative repeatedly. The exact u has no oscillations. The oscillations persist far from the feature, which is called *pollution*. Error from the under-resolved feature pollutes the whole domain. An *artifact* is something in the numerical calculation (the interpolant in this case) that is not in the desired solution. Under-resolved Fourier approximations lead to serious artifacts.

A sharp feature like a discontinuity in u (Figure 3) or a discontinuity in $\partial_x u$ (Figure 2) may be thought of an infinitely small feature. This makes it impossible to resolve in the way just described. Such a sharp feature will give artifacts no matter how many modes are used. The Gibbs phenomenon overshoot in Figure 3 will not go away as $n \to \infty$. The artifacts seem less severe for a discontinuity in the derivative, Figure 2. But they are still visible. Fourier interpolation of the continuous but not differentiable function converges but is not very accurate.

3.3 Decay of Fourier coefficients

A function is *smooth* to the extent that its derivatives exist and are not crazy. The degree of smoothness is determined by how many derivatives make sense. A very smooth function has a lot of derivatives. *Analyticity* is the most extreme form of smoothness. A function $u(x)$ is analytic if all its derivatives exist and if there is a C and a $\rho > 0$ so that

$$|\partial_x^n u(x)| \leq \frac{Cn!}{\rho^n} \; .$$

The right side is the formula related to radius of convergence of Taylor series in basic calculus. A function is analytic of its Taylor series has a positive radius of convergence at every point. The solutions to differential equations often are smooth or analytic.

It is a basic fact in scientific computing that the Fourier representation (20) is efficient for smooth functions. *Efficient* means that you get an accurate approximation using a small number of terms in the sum. Fourier approximation is efficient for smooth functions because the Fourier coefficients go to zero rapidly. If u has n derivatives, then (we will see this)

$$|\widehat{u}_\alpha| \leq Cn^{-\alpha} \; .$$

The more derivatives, the more rapidly $\widehat{u}_\alpha$ converges to zero. If u is analytic, then there is an $a < 1$ with

$$|\widehat{u}_\alpha| \leq Ca^{-|\alpha|} \; .$$

That is, the Fourier coefficients converge to zero exponentially.

There are many ways to look at derivatives using Fourier series. You can write the Fourier series using the wave numbers k_α of (28) as

$$u(x) = \sum_\alpha \widehat{u}_\alpha e^{ik_\alpha x} \ .$$

The derivative is

$$\partial_x u(x) = \sum_\alpha ik_\alpha \widehat{u}_\alpha e^{ik_\alpha x} \ . \tag{30}$$

This formula says that you get the derivative by multiplying the Fourier coefficient by ik_α. The Fourier coefficients of $\partial_x u$ are $ik_\alpha \widehat{u}_\alpha$. The Fourier differentiation formula can be applied repeatedly, do $\partial_x^2 u$ brings down $(ik_\alpha)^2$, and so

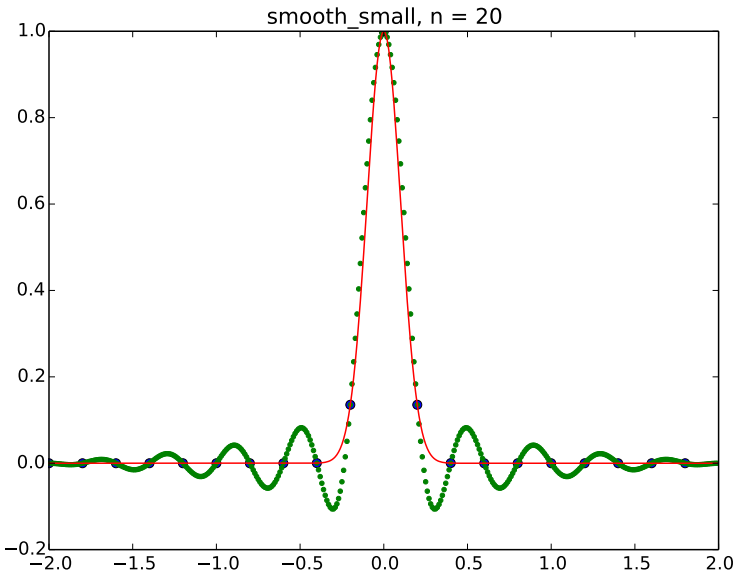

Figure 4: Fourier under-resolved interpolation of a smooth function. The exact function is the solid line, the interpolant is small dots, interpolating points are large dots. The grid is too coarse to resolve the feature. Instead we see overshoots. The Python source is in `codes/FourierInterpolation.py`.

on:

$$
\begin{aligned}
u &\iff \widehat{u}_\alpha \\
\partial_x u &\iff i k_\alpha \widehat{u}_\alpha = \frac{2\pi i \alpha}{R} \widehat{u}_\alpha \\
\partial_x^n u &\iff i^n k_\alpha^n \, \widehat{u}_\alpha = i^n \left(\frac{2\pi}{R} \right)^n \alpha^n \, \widehat{u}_\alpha
\end{aligned}
\tag{31}
$$

Look at (31). The factor i^n is something like a sign in that $|i^n| = 1$. The factor k^n indicates that differentiation increases the high k Fourier coefficients more. It also makes the $k = 0$ (or $\alpha = 0$) Fourier coefficient equal to zero. The factor $(2\pi/R)^n$ increases or decreases with n (depending on R) but is independent of α.

The spectral derivative of a grid function is defined as follows. You have the numbers U_j that are represent the values of some approximation on grid points x_j. The grid points are uniformly spaced on an interval of length R, $x_j = j\Delta x$, $\Delta x = R/n$. The discrete Fourier coefficients are $\widehat{U}_\alpha$ and the trigonometric interpolant is $w(x)$. It is important to use the symmetric fundamental set $\mathcal{S}_a =$

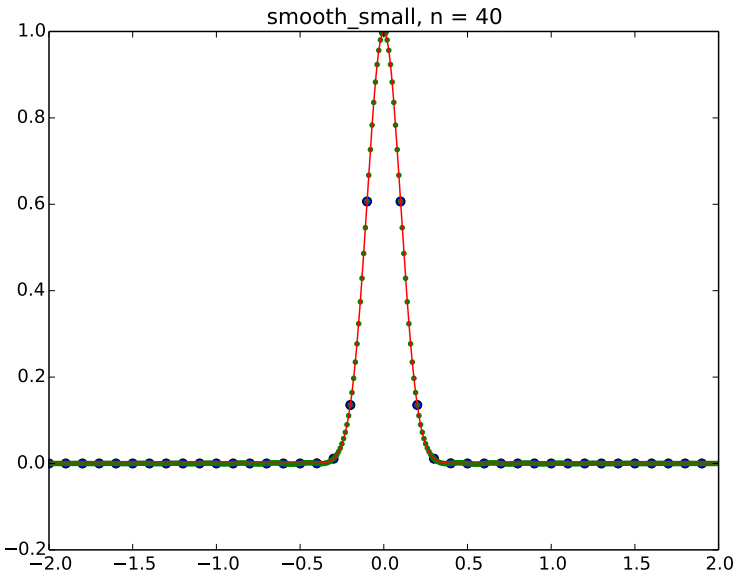

Figure 5: Fourier interpolation of a smooth function, well resolved. The exact function is the solid line, the interpolant is small dots, interpolating points are large dots. The interpolating function is accurate. The Python source is in `codes/FourierInterpolation.py`.

$\{-m+1, \ldots, m\}$, with for $n = 2m$ nodes. The *spectral derivative* operator is a map $D : \mathbb{R}^n \to \mathbb{R}^n$ defined by

$$(DU)_j = \partial_x w(x_j) \ . \tag{32}$$

To compute the spectral derivative:

1. calculate the discrete Fourier coefficients $\widehat{U} = \mathrm{DFT}(U)$.
   You must check what version of the DFT the FFT software calculates. The Python `numpy.fft` package computes

   $$Y_\alpha = \sum_{j=0}^{n-1} e^{-2\pi i \alpha j/n} U_j \ .$$

   for $\alpha \in \mathcal{S}_p$, which is $\alpha \in \{0, \ldots, n-1\}$. To get the coefficients in our symmetric fundamental set, you must normalize by dividing by n and shift. The posted code `FourierInterpolation.py` does this.

2. differentiate in Fourier space $\widehat{V}_\alpha = i k_\alpha \widehat{U}_\alpha$.
   The wave numbers are related to α by (28). In the code `FourierInterpolation.py`, the array holding $\widehat{U}$ is centered but, but Python arrays start from index zero. You have to get this right to calculate k_α correctly.

3. calculate the inverse discrete Fourier transform $DU = \mathrm{IDFT}(\widehat{V})$. Be careful of conventions. The Python inverse discrete Fourier transform code `numpy.fft.ifft` has the right factor of n convention, but it returns a complex sequence. In exact arithmetic the real parts all would be zero (why?). In Python, you need to *cast* the complex array returned by `ifft` to desired real array by taking the real part.

Smooth means that many derivatives of u exist and are not infinite. For example, suppose that the p^{th} derivative of u is bounded:

$$\max_x \left| u^{(p)}(x) \right| = A_p < \infty \ . \tag{33}$$

This (we will see) implies that the Fourier coefficients go to zero quickly as $\xi \to \infty$. The rapid decay of Fourier coefficients implies that "most" of the Fourier sum (20) is given by a the few terms with small ξ.

The Fourier integrals (20) are small for large ξ because of *cancellation*. The integrand is not small, but the integral is small because the positive and negative parts almost exactly cancel. You can see this by integrating $e^{-ikx}u(x)$ over one period of the plane wave. That means, you integrate from x_0 to $x_0 + 2\pi/k$. For large k, this is a small interval. If $u'(x)$ is not large, then u does not change much over this interval. Suppose that

$$|u(x) - u(x_0)| \leq \epsilon \quad \text{if } x_0 \leq x \leq x_0 + \frac{2\pi}{k} \ .$$

Integrating the constant over one period gives zero. That's the cancellation:

$$\int_{x_0}^{x_0+\frac{2\pi}{k}} e^{-ikx} u_0 \, dx = 0 \; .$$

Therefore integrating with u over one period gives something small (remember that $\left|e^{-ikx}\right| = 1$ always)

$$\begin{aligned}
\left| \int_{x_0}^{x_0+\frac{2\pi}{k}} e^{-ikx} u(x) \, dx \right| &= \left| \int_{x_0}^{x_0+\frac{2\pi}{k}} e^{-ikx} u_0 \, dx + \int_{x_0}^{x_0+\frac{2\pi}{k}} e^{-ikx} \left(u(x) - u_0\right) \, dx \right| \\
&= \left| \int_{x_0}^{x_0+\frac{2\pi}{k}} e^{-ikx} \left(u(x) - u_0\right) \, dx \right| \\
&\leq \int_{x_0}^{x_0+\frac{2\pi}{k}} \left|u(x) - u_0\right| \, dx \\
&\leq \int_{x_0}^{x_0+\frac{2\pi}{k}} \epsilon \, dx \\
&= \frac{2\pi}{k} \, \epsilon \; .
\end{aligned}$$

The integral over a period of the plane wave is less than the length of the period ($2\pi/k$) multiplied by the "oscillation" of u over that interval (the amount u varies over the interval).

Now consider a Fourier mode (18) with large α and wave vector $k = 2\pi\alpha/R$ also large. The Fourier integral (20) is the sum of ξ integrals over individual periods of $e^{-2\pi i\alpha x/R}$. Even if the oscillation of u is large, the oscillation over each small period is small. The sum of the lengths of these periods is R. If you put this information together, you see that the integral should be small.

This argument is intuitive, but clumsy to carry out in detail. There is a more "efficient" version of this argument that gets better answers with less calculation, but also possibly with less intuition. The trick is to integrate by parts in the Fourier integral (21) assuming that $\alpha \neq 0$. We use the plane wave differentiation formula in the form

$$\frac{-R}{2\pi i\alpha} e^{-2\pi i\alpha x/R} = \partial_x e^{-2\pi i\alpha x/R} \; .$$

There are no "boundary terms" in the integration by parts because the values

at $x = 0$ and $x = R$ are equal because u and v_ξ are periodic.

$$\begin{aligned}
\widehat{u}_\alpha &= \int_0^R e^{-2\pi i\alpha x/R}\, u(x)\, dx \\
&= \frac{-R}{2\pi i\alpha} \int_0^R \left(\frac{-2\pi i\alpha}{R} e^{-2\pi i\alpha x/R} \right) u(x)\, dx \\
&= \frac{-R}{2\pi i\alpha} \int_0^R \left(\partial_x e^{-2\pi i\alpha x/R} \right) u(x)\, dx \\
&= \frac{R}{2\pi i\alpha} \int_0^R e^{-2\pi i\alpha x/R} \partial_x u(x)\, dx \ .
\end{aligned}$$

This integration by parts can be repeated p times. Each integration by parts gives another factor of $R/2\pi i\xi$. The result is

$$\widehat{u}_\alpha = \left(\frac{R}{2\pi i} \right)^p \frac{1}{\alpha^p} \int_0^R e^{-2\pi i\alpha x/R} \left(\partial_x^p u(x) \right)\, dx \ .$$

This leads to the inequality

$$|\widehat{u}_\alpha| \le C_p\, |\alpha|^{-p} \ ,$$

where

$$C_p = \frac{A_p R^{p-1}}{(2\pi)^p} \ .$$

4 Time dependent fields

In this context, a *field* is a function of a "spatial variable" $x \in \mathbb{R}^d$. For example, if $u(x)$ is the temperature at point x, then the function u would be the temperature field. If $E(x) = (E_1(x), E_2(x), E_3(x))$, then E could be the electric field. A field with just one component is a *scalar* field. The electric field is a vector field. Some engineers call a field a *distributed parameter*. The parameter (temperature or electric field), has different values at different points. A *lumped* parameter can be a finite dimensional approximation to a distributed parameter. For example, a lumped parameter approximation to a temperature field could be a list of n temperature readings at n measurement points.

A *time dependent* field is one that changes with with time. For example, $u(x,t)$ could be the temperature at location x at time t. We write $u(\cdot,t)$ to represent the whole field at time t. For example, $v(\cdot) = u(\cdot,t_0)$ means that $v(x) = u(x,t_0)$ for all x. The *evolution* of a time dependent field is the change of the field with time. This could be modeled by an *evolution equation* of the form

$$\partial_t u(\cdot,t) = F(u(\cdot,t)) \ .$$

The "function" F takes a field as an argument and returns another field. Functions like that (functions of functions) are often called *operators*. An operator can be linear or nonlinear.

Suppose F is an operator and $v(\cdot) = F(u(\cdot))$. Then F is *local* if $v(x)$ depends only on $u(x)$ and some derivatives of u at x. For example, $v(x) = u(x)^2$ is a local and nonlinear operator. It also might be written $F(u)(x) = u(x)^2$, as $F(u)$ is a function (also written v) and we can evaluate the function at x. The *Laplace operator*, or *Laplacian*, is the local linear differential operator

$$\triangle u(x) = \sum_{k=1}^{d} \partial_{x_k}^2 u(x) \ .$$

The *inverse Laplace operator* in $3D$ is

$$u = \triangle^{-1} v \iff u(x) = \frac{1}{4\pi} \int_{\mathbb{R}^3} \frac{1}{|x-y|} \, v(y) \, dy \ .$$

This is a non-local integral operator. It is non-local because $u(x)$ depends on v at every other point, not just near x. It is integral because it involves an integral. It is the inverse Laplace operator because, if u is a suitable function of $x = (x_1, x_2, x_3) \in \mathbb{R}^3$, and if

$$v(x) = \frac{\partial^2 u(x)}{\partial x_1^2} + \frac{\partial^2 u(x)}{\partial x_2^2} + \frac{\partial^2 u(x)}{\partial x_3^2} \ ,$$

then

$$u(x_1, x_2, x_3) = \frac{1}{4\pi} \int \int \int \frac{v(y_1, y_2, y_3)}{\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^3}} \, dy_1 dy_2 dy_3 \ .$$

We are concerned with time dependent fields that satisfy local evolution equations, which are partial differential equations. Such equations are sometimes called "time dependent equations", but it isn't the equation that is changing, it's the field. The initial value problem for such a partial differential equation consists of the evolution equation with a local operator and the initial values. There are two "time dependent" partial differential equations that illustrate features of many others. One is the *heat equation* (also called the *diffusion equation*)

$$\partial_t u = D \triangle u \ . \tag{34}$$

The other is the *wave equation* (or *scalar* wave equation)

$$\partial_t^2 u = c^2 \triangle u \ . \tag{35}$$

The physical meanings of these equations are explained in the *Feynman Lectures on Physics*, (**bold** blue text should be clickable links) `Volume 2, Lecture 3` (the heat equation), and `Volume 1, Lecture 47` (the wave equation). Please read these lectures. You can buy them cheaply, or read them online at the `Feynman lectures web site`, which is `http://www.feynmanlectures.caltech.edu/` .

4.1 Simple diffusion in one dimension

The diffusion equation (34) in one dimension models diffusion of something (heat or the concentration of a chemical) along a line. Let $x \in \mathbb{R}$ be position along a line and let $u(x,t)$ represent the concentration at location x at time t. As the stuff diffuses, $u(x,t)$ changes according to the *diffusion equation*

$$\partial_t u = D\,\partial_x^2 u\ . \tag{36}$$

The coefficient D is the *diffusion coefficient* (not the spectral differentiation operator). If x has units of length (L) and t has units of time (T), then D has units of L^2/T. The initial value problem is to determine $u(x,t)$ for $t > 0$ when the initial conditions $u(x,0)$ are known.

A semi-discrete approximation of the one dimensional diffusion equation (36) involves a *discretized*, or "lumped", approximation vector $U(t) \in \mathbb{R}^n$. The field $u(\cdot,t)$ may be approximated through *sampling*. Let Δx be a *grid spacing*, or *space step*, and define *grid points*, or *sample points*

$$x_j = j\Delta x\ .$$

The set of grid points forms the *grid*, or *lattice*. The field values at the sample points are the sample values $u_j(t) = u(x_j,t)$. We suppose $U(t)$ is an approximation to u_j:

$$U_j(t) \approx u(x_j,t)\ .$$

The operator in the evolution equation (36) is the linear differential operator

$$L = D\partial_x^2\ .$$

To create A in the semi-discrete approximation (2), we seek a discrete approximation of L that acts on U to produce a discrete approximation to $D\partial_x^2 u(x_j,t)$.

A *finite difference method* replaces the differential operator ∂_x^2 with a finite difference approximation:

$$\partial_x^2 f(x) = \frac{1}{\Delta x^2}\left[f(x-\Delta x) - 2f(x) + f(x+\Delta x)\right] + O(\Delta x^2)\ .$$

The $O(\Delta x^2)$ size of the error is true if f has four continuous derivatives. We can apply this at each t to the time dependent field $u(\cdot,t)$ to get

$$\dot{u}_j = \frac{D}{\Delta x^2}\left[u_{j+1} - 2u_j + u_{j-1}\right] + O(\Delta x^2)\ . \tag{37}$$

This suggests the semi-discrete approximation

$$\dot{U}_j = \frac{D}{\Delta x^2}\left(U_{j+1} - 2U_j + U_{j-1}\right)\ . \tag{38}$$

The semi-discrete evolutions equations (38) do not specify the semi-discrete problem exactly. There can be only finitely many U values in the computer, so

we must restrict the range of j using *boundary conditions*. Boundary conditions are a part of the physical problem description too. We must say what happens at the ends of the x interval. Boundary conditions, particularly discretizing them, can be more challenging than the semi-discrete equations that apply in the interior. Therefore, we need a way to make U finite without putting in boundary conditions.

This can be done by supposing that the field $u(\cdot, t)$ is *periodic* in x with period R. That is

$$u(x + R, t) = u(x, t) \ .$$

Boundary conditions (along with the diffusion equation and initial conditions) complete the formulation of the initial value problem. For now, we will assume that the initial conditions are periodic with period R. For all x,

$$u(x, 0) = u(x + R, 0) \ .$$

This implies (since the solution is unique) that the solution also is periodic with period R. The finite difference approximation sequence $u_j(t)$ will be periodic if exactly d grid points fit in one period. This means that

$$n \, \Delta x = R \ , \ \ \Delta x = \frac{R}{n} \ . \tag{39}$$

The matrix A corresponding to the right side of (37) has diagonals

$$A_{jj} = \frac{-2D}{\Delta x^2} \ .$$

The off diagonals are

$$A_{j,j+1} = A_{j,j-1} = \frac{D}{\Delta x^2} \ .$$

But in addition to its tri-diagonal entries, A has non-zeros in the top right and bottom right:

$$A_{0,d-1} = A_{d-1,0} = \frac{D}{\Delta x^2} \ .$$

These "wraparound" entries arise from periodicity. For example, the $j = 0$ differential equation (37) is

$$\dot{u}_0 = \frac{D}{\Delta x^2} \left[u_1 - 2u_0 + u_{-1} \right] = \frac{D}{\Delta x^2} \left[u_1 - 2u_0 + u_{d-1} \right] \ .$$

The matrix A is

$$A = \begin{pmatrix} \frac{-2D}{\Delta x^2} & \frac{D}{\Delta x^2} & 0 & \cdots & 0 & \frac{D}{\Delta x^2} \\ \frac{D}{\Delta x^2} & \frac{-2D}{\Delta x^2} & \frac{D}{\Delta x^2} & 0 & \cdots & 0 \\ 0 & \frac{D}{\Delta x^2} & \frac{-2D}{\Delta x^2} & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & & 0 \\ 0 & & \ddots & & \frac{-2D}{\Delta x^2} & \frac{D}{\Delta x^2} \\ \frac{D}{\Delta x^2} & 0 & \cdots & 0 & \frac{D}{\Delta x^2} & \frac{-2D}{\Delta x^2} \end{pmatrix} \ . \tag{40}$$

The formula (37) is another convenient way to express the action of A. If $f \in \mathbb{C}^d$ with components $f_0, \ldots, f_{d-1}$ and $g = Af$, then for $j = 0, 1, \ldots, d-1$,

$$g_j = \frac{D}{\Delta x^2} \left[f_{j+1} - 2f_j + f_{j-1} \right] \ , \text{ where } \ f_{-1} = f_{d-1} \ , \text{ and } \ f_d = f_0 \ .$$

The eigenvalues and eigenvectors of A tell us a lot about the behavior of solutions (see notes, part 1). Discrete Fourier modes are the eigenvectors of A. The corresponding eigenvalues are found by calculation. These calculations, called *symbol calculations*, are used constantly to derive and understand finite difference schemes for partial differential equations. Suppose $v_\alpha \in \mathbb{C}^d$ is a Fourier mode given by (3). We write component j of Av_α as $(Av_\alpha)_j$. The crucial step in the symbol calculation is finding the common exponential factor on the right.

$$\begin{aligned}
(Av_\alpha)_j &= \frac{D}{\Delta x^2} \left[v_{\alpha,j+1} - 2v_{\alpha,j} + v_{\alpha,j-1} \right] \\
&= \frac{D}{\Delta x^2} \left[e^{2\pi i \alpha (j+1)/d} - 2e^{2\pi i \alpha j/d} + e^{2\pi i \alpha (j-1)/d} \right] \\
&= \frac{D}{\Delta x^2} \left[e^{2\pi i \alpha j/d} e^{2\pi i \alpha/d} - 2e^{2\pi i \alpha j/d} + e^{2\pi i \alpha j/d} e^{-2\pi i \alpha/d} \right] \\
&= \frac{D}{\Delta x^2} \left[e^{2\pi i \alpha/d} - 2 + e^{-2\pi i \alpha/d} \right] e^{2\pi i \alpha j/d}
\end{aligned}$$

The result is

$$(Av_\alpha)_j = m(\alpha) e^{2\pi i \alpha j/d} \ , \tag{41}$$

where the *symbol* of the difference operator A is

$$m(\alpha) = \frac{D}{\Delta x^2} \left[e^{2\pi i \alpha/d} - 2 + e^{-2\pi i \alpha/d} \right] \ .$$

This is simplified by writing

$$\theta = \frac{2\pi\alpha}{d}$$

and using a trig identity. Then (with some abuse of notation)

$$m(\theta) = \frac{-2D}{\Delta x^2} \left[1 - \cos(\theta) \right] \ . \tag{42}$$

The eigenvalues of A are

$$\lambda_\alpha = m(\theta_\alpha) \ , \ \ \alpha = 0, \ldots, d-1 \ . \tag{43}$$

We will return several times to these formulas. Some important properties if these eigenvalues that we will return to constantly:

1. They are all real. This is a consequence of the fact that the matrix A in (40) is symmetric.

2. They are all non-positive, which is equivalent to the fact that A is negative semi-definite. This is a consequence of the formula (which we will use in more detail later)

$$\langle U, AU \rangle = -D\Delta x \sum_{0}^{n-1} \frac{(U_{j+1} - U_j)^2}{\Delta x^2} \ .$$

The right side is clearly non-positive, and equal to zero only if U is constant. It is a discrete approximation to

$$-D \int_0^R \left(\partial_x u(x)\right)^2 \, dx \ .$$

3. They have a wide range of values. The first non-zero eigenvalue comes from $\alpha = \pm 1$ and is (check this)

$$\lambda_{\pm 1} = \frac{-D\,(2\pi)^2}{R^2} + O(\Delta x^2) \ .$$

The point is that this has a finite limit as $n \to \infty$ and $\Delta x \to 0$. The "largest" (most negative) eigenvalue is from the other end of the "spectrum" with $\theta \approx \pi/2$, which gives

$$\lambda_{\max} \approx \frac{-4D}{\Delta x^2} \ .$$

Thus A has a wide range of eigenvalues, which corresponds to a wide range of time scales in the solution of the semi-discrete problem (2). If we want to compute the solution correctly, this would take

$$\Delta t \ll \frac{1}{|\lambda_{\max}|} = \frac{\Delta x^2}{4D} \ . \tag{44}$$

This makes Δt very small when Δx is small.

The formula (44) is our first instance of a *time step constraint* in solving a partial differential equation. If we discretize in space and time, there is likely to be a relation between Δx and Δt, or at least a constraint on the relation. For the diffusion equation this seems to require very small time steps. In three dimensions, the number of grid points is proportional to Δx^{-3}. If the time step is proportional to Δx^2, then the number of time steps to reach a specific time is proportional to Δx^{-2}. The overall computation requires you to visit each grid point once per time step, for a total of

$$\Delta x^{-3} \times \Delta x^{-2} = \Delta x^{-5}$$

operations for the complete simulation. If you decide you need more resolution and replace do it again with grid spacing $\Delta x/2$, this computation is longer by a factor of $2^5 = 32$.

The requirement $\Delta t << \Delta x^2$ is usually not necessary for accuracy even though it seems to be. This is because the problem is usually *stiff*. The modes corresponding to eigenvalues $\lambda_{\max}$ have so little energy that they do not need to be computed accurately. A problem is *stiff* if you would be happy to solve it with time steps much larger than the fastest time scale in the problem. This makes sense only if the fast modes have little "energy". We have seen that if u is analytic, then the Fourier coefficients decay exponentially (actually, this is the definition of "analytic"). Problems with time dependent fields are likely to be stiff if the field being computed is analytic.

4.2 The wave equation in one dimension

The wave equation (35) has similarities and important differences from the heat equation. It is natural to semi-discretize the wave equation in one dimension in a manner similar to (38)

$$\ddot{U}_j = \frac{c^2}{\Delta x^2} \left(U_{j+1} - 2U_j + U_{j-1} \right) \ . \tag{45}$$

This has modes of the form

$$U(t) = V_\alpha e^{i\omega_\alpha t} \ .$$

You can calculate the ω_α explicitly to verify the following:

1. The ω_α are all real.
2. They have a wide range of scales, from $\omega_{\pm 1} = O(1)$ to

$$\omega_{\max} = O\big(\Delta x^{-1}\big) \ .$$

This means that an accurate solution of all modes would require

$$\Delta t \ll \Delta x \ .$$

This is less severe than for the diffusion equation, but still severe. If we write (45) as a first order system, the eigenvalues are of the form $\pm i\omega_\alpha$. For the wave equation, the eigenvalues have a wide range along the imaginary axis.

5 Exercises

1. (*You may have to review some linear algebra, vector norms and condition number of matrices, for this problem.*) Vector and matrix norms matter, particularly in high and infinite dimensions. Three important norms for

sequences $U \in \mathbb{R}^n$ or $U \in \mathbb{C}^n$ are

$$\begin{aligned} \|U\|_{l^1} &= \sum_{j=0}^{n-1} |U_j| \\ \|U\|_{l^2} &= \left(\sum_{j=0}^{n-1} |U_j|^2 \right)^{1/2} \\ \|U\|_{l^\infty} &= \max_j |U_j| \end{aligned}$$

If $\|U\|_a$ and $\|U\|_b$ are any two norms, then the condition number of norm a with respect to norm b is

$$\kappa(a,b) = \frac{\max\limits_{U \neq 0} \dfrac{\|U\|_a}{\|U\|_b}}{\min\limits_{U \neq 0} \dfrac{\|U\|_a}{\|U\|_b}} \quad . \tag{46}$$

If $\kappa(a,b)$ is large, then knowing something in $\|U\|_a$ may not tell you much about $\|U\|_b$. There are many examples of this in numerical differential equations.

(a) Suppose M is an $n \times n$ positive definite symmetric matrix. Define the "M norm" as

$$\|U\|_M = (\langle U, MU \rangle)^{1/2} \quad .$$

Let $\kappa(M)$ be the "traditional" condition number

$$\kappa(M) = \frac{\lambda_{\max}(M)}{\lambda_{\min}(M)} \; . \tag{47}$$

Show that

$$\kappa(M, 2) = \kappa(M) \; .$$

On the left is the condition number of the M norm with respect to the 2 norm (46). On right is the condition number of M (47).

(b) Show that $\kappa(a,b) \geq 1$ for any pair of vector norms.

(c) Show that if $\|\cdot\|_a$ is a *scaling* of $\|\cdot\|_b$ then $\kappa(a,b) = 1$. We say that $\|\cdot\|_a$ is a *scaling* of $\|\cdot\|_b$ by s if $\|U\|_a = s \|U\|_b$ for all U.

(d) Show that $\kappa(a,b) = \kappa(b,a)$. *Hint*: Suppose U_1 maximizes $\|U\|_a / \|U\|_b$, then U_1 minimizes $\|U\|_b / \|U\|_a$, why? Also,

$$\kappa(a,b) = \max_{\|U\|_b = 1} \|U\|_a \quad \text{(why?)} \; .$$

(e) Find the condition numbers of the three norms above (l^1, l^2, and l^∞) relative to each of the others. The important thing here is now these condition numbers scale with n.

2. This series of steps gives a convergence proof for a semi-discrete approximation. It is a consistency/stability argument, but such arguments are more technical when applied to time dependent field problems. Consider a semi-discrete approximation of the diffusion equation

$$\dot{U}_j = \frac{D}{\Delta x^2} \left(U_{j+1} - 2U_j + U_{j-1} \right) \ . \tag{48}$$

Suppose that U is real, which means that $U(t) \in \mathbb{R}^n$.

(a) Show that (48) satisfies the *discrete maximum principle*

$$\|U(t)\|_{l^\infty} \leq \|U(0\|_{l^\infty} \ . \tag{49}$$

Hint: Start by showing that the semi-discrete equations imply

$$U_j(t) = \max_k U_k(t) \geq 0 \implies \dot{U}_j(t) \leq 0 \ .$$

You need a similar fact for the min in case $\|U(t)\|_{l^\infty} = |\min U_k(t)|$. This proves that the semi-discrete approximation (48) is stable in the l^∞ norm.

(b) We write $S(t)$ for the solution "operator" (an $n \times n$ matrix in this case) of the semi-discrete approximation (2). This satisfies $S(0) = I$ and $\dot{S} = AS$. Suppose E satisfies the *inhomogeneous* equation

$$\dot{E} = AE + R(t) \ .$$

Show that

$$E(t) = S(t)U(0) + \int_0^t S(t - t_1)R(t_1)\, dt_1 \ . \tag{50}$$

We will call this *Duhamel's principle*, but it has other names.

(c) Show that the discrete maximum principle implies that for all $t \geq 0$,

$$\|S(t)\|_{l^\infty} \leq 1 \ .$$

Use this to conclude that if U satisfies the inhomogeneous equation, then

$$\|E(t)\|_{l^\infty} \leq \|E(0)\|_{l^\infty} + t \max_{0 \leq t_1 \leq t} \|R(t_1)\|_{l^\infty} \ . \tag{51}$$

You may assume that if $V(t)$ is "any" vector function of t (any measurable function, or continuous function) and if $\|\cdot\|$ is any vector norm, then there is an "obvious" integral version of the triangle inequality:

$$\left\| \int_0^t V(t_1)\, dt_1 \right\| \leq \int_0^t \|V(t_1)\| \ dt_1 \ .$$

(d) Suppose that $u(x,t)$ satisfies the diffusion equation $\partial_t u = D\partial_x^2 u$, and that $\left|\partial_x^4 u(x,t_1)\right| \leq C$ for all x and $0 \leq t_1 \leq t$. Define the error vector $E(t) \in \mathbb{R}^n$ as

$$E_j(t) = U_j(t) - u(x_j,t) \; .$$

Suppose that $U_j(0) = u(x_j,0)$ (the initial data for U are correct). Show that the semi-discrete approximation is second order accurate in the sense that

$$\max_j \max_{t \leq T} |U_j(t) - u(x_j,t)| \leq CT\Delta x^2 \; . \tag{52}$$

3. The *Kuramoto Sivashinsky* equation (called *KS*), in one dimension, is for a function $u(x,t)$ with $x \in \mathbb{R}$ (i.e., one dimensional) and $t \geq 0$. We will study solutions that are periodic in space (in x) with a period R that is large but finite. The equation models the evolution of certain nearly planar flames that are unstable by linear analysis. A remarkable feature is that this linear instability *saturates*, so that perturbations grow to a certain size then stop growing. Although the solution stops growing, it continues to evolve in a chaotic manner. The solution is chaotic both in time (as the Lorenz system is) and in space, which is new. Systems with spatial and temporal chaos are often called *turbulent*, by analogy with turbulent water flows that have a lot of "structure" (the image at a given time is a complicated function of x). This system has been used as a model of other turbulent systems.

$$\partial_t u + \tfrac{1}{2}\partial_x u^2 = -\partial_x^2 u - \partial_x^4 u \; . \tag{53}$$

A simplified system that looks similar but has simpler behavior is the *Burgers Sivashinksy* equation (or *BS*).

$$\partial_t u + \tfrac{1}{2}\partial_x u^2 = u + \partial_x^2 u \; . \tag{54}$$

For both equations, we assume that

$$\int_0^R u(x,t)\, dx = 0 \; . \tag{55}$$

You can check that this is consistent with both evolution equations, which means that if it is satisfied at time $t = 0$, then it is satisfied at later times $t > 0$. This exercise asks you to solve the initial value problem for the KS and BS equations and explore the behavior numerically. The numerical solution will use a *spectral method* based on the DFT/FFT.

The steps below are sometimes vague. This forces you to decide what makes sense. You must find good test problems and decide how to present results. Please include plots and movies when they are helpful. You will need to pay constant attention to the issue of real and complex arithmetic. The code will be build in a sequence of steps, so you don't need separate codes for the individual parts.

Write in Python using vector instructions as much as possible to make the code run faster. The posted code `FourierInterpolation.py` gives hints how to use the Python FFT routines.

(a) The term $\frac{1}{2}\partial_x u^2$ is the *advection* term. Linearizing the equation means leaving this out, leaving a linear equation. Show that for KS and for BS the linearized equations have many modes that are linearly unstable if R is large. For a linear evolution equation $\partial_t = Au$, an *unstable mode* is an eigenfunction v with $Av = \lambda v$ and $\mathrm{Re}(\lambda) > 0$ (the right half plane). "Many" unstable modes means many linearly independent modes. It happens that each eigenspace is one dimensional (for real v), so "many unstable modes" also means many eigenvalues in the right half plane. For this problem, if $w = e^{2\pi i \alpha x/R}$ is an eigenfunction with real eigenvalue (it is, check), then $v(x) = \mathrm{Re}(w(x))$ is a real eigenfunction with the same eigenvalue (check). Show that (for large enough R), most solutions of the linearized equations grow exponentially in time.

(b) Suppose $U \in \mathbb{R}^n$ is a discrete periodic sequence. Then $V \in \mathbb{R}^n$ is the *spectral derivative* if

$$V_j = \partial_x w(x_j) \ ,$$

where $\Delta x = R/n$, $x_j = j\Delta x$, and $w(x)$ is the trigonometric polynomial that interpolates U at the sample points x_j. We are interested in the real spectral derivative of a real sequence U (you might have to take the real part at some points). Write Python code to evaluate the spectral derivative of a periodic sequence. Check that this is exact for $U_j = \cos(2\pi\alpha x_j/R)$ (to within rounding error). Check that it has *spectral accuracy* for some other periodic and analytic functions where it is not exact. Modify your code to compute second and fourth spectral derivatives. Check that these are correct.

(c) Write a *method of lines* code for BS (that is easy to change to KS). This code should consist of a Runge Kutta time stepper applied to a semi-discrete approximation $\dot{U} = F(U)$. Choose data with mean zero, so $\sum_j U_j(t) = 0$. If you do the spectral differentiation right, if this is true at time 0, then it remains exactly true (in exact arithmetic) for $t > 0$. The function F should consist of spectral derivatives for all partial derivatives, and U_j^2 for u^2. It is best to a higher order accurate Runge Kutta code, either the third order one from the last homework or "the" four stage fourth order method (find it on wikipedia). If you choose initial data that has period R but not any smaller period, then the solution should converge as $t \to \infty$ to a steady state. Make a movie of this convergence. Check that the solution is accurate to plotting accuracy by running with various values of Δt and Δx. You will find that you need to reduce Δt when you reduce Δx. Make plots of the discrete Fourier coefficients to see that the solution is analytic. One way to do this (possibly not the best

way?) is to make a plot of

$$\log\left(\left|\widehat{U}_\alpha\right|\right)$$

and see whether this decreases linearly as α increases, for large α. The steady states you get for large R should look like the one pictured on page 300 of "Stability of the Kuramoto-Sivashinsky and Related Systems" (Communications of Pure and Applied Math, vol. 47, pp 293-306). Experiment with n (the number of modes) as a function of R that you need to resolve (compute accurately) the solution. Explain the result in terms of the smallest feature size in the solution for large R.

(d) Apply your code to the KS equation. For large R, if the initial data has no symmetry beyond having period R (is not even or odd, does not have a period smaller than R), then the solution should just be chaotic in space and time (turbulent). Experiment to see what n you need as a function of R. You should find that the BS equation requires more modes than the KS equation for large R. See if you can explain this by looking at the computed solutions.

(e) (*the science of this problem*). Solutions of KS for large R have features in common with turbulence in high Reynolds number fluids. In particular, there is an *inertial range* and a *dissipation range* in the Fourier transform of the solution. The inertial range is wave numbers below the *Kolmogorov scale*, which means $|\alpha| < \alpha_K$ in our notation (since short lengths correspond to large wave numbers). Here, $\widehat{U}_\alpha$ has a simple behavior and does not decay exponentially. The dissipation range is $|\alpha| > \alpha_K$, where the Fourier coefficients decay exponentially as they should for an analytic function. The Wikipedia page `https://en.wikipedia.org/wiki/Energy_cascade` has more. This is an active area of research around the world and at the Courant Institute. Make plots of the Fourier coefficients ($\left|\widehat{U}_\alpha\right|$)for large R. Look for a transition from one kind of decay (or lack of decay) to exponential decay. It is not a precise transition, but can be seen in plots. How does the transition point depend on R? You may have to do some big runs to see this clearly.