Fourier Spectral Methods

Johane André Mutuque (johane@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Professor Nick Hale
Stellenbosch University, South Africa

23 May 2019

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

In this essay we demonstrate the use of Fourier series to numerically solve ordinary differential equations (ODE) in periodic intervals, as well as partial differential equations (PDE) on periodic rectangles. We present the collocation method and the Galerkin method using Fourier bases. The aforementioned approaches are known as Fourier spectral methods, which we demonstrate to be efficient and have high accuracy in the construction of approximate solutions of differential equations on such domains.

Keywords: Fourier spectral methods, FFT, Galerkin method, collocation method.

Resumo

Nesta dissertação demonstramos a aplicação das séries de Fourier na resolução numérica de equações diferenciais ordinárias em intervalos periódicos, bem como equações diferenciais com derivadas parciais em rectângulos periódicos. Apresentamos o método de colocação espectral e o método de Galerkin usando a base de Fourier. Por conseguinte, ficam denominados métodos espectrais de Fourier, provamos serem processos eficientes e oferecem alta precisão na construção de soluções aproximadas de equações diferenciais nos domínios referidos.

Palavras-chave: método espectral de Fourier, FFT, método de Galerkin, colocação espectral.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Johane André Mutuque, 23 May 2019

Contents

List of Figures

1. Introduction

Methods like finite difference and finite elements are based on local representations of functions using lower order polynomials. On the other side, spectral methods are based on global representations using higher order Legendre, Chebyshev polynomials or Fourier series, (Trefethen, 2000). We focus on the ones based on Forurier series, therefore called Fourier spectral methods.

According to Gottlieb and Orszag (1977) and Trefethen (2000), around 1950 Lanczos emphasised the power of Forurier series and Chebyshev polynomials on solving ordinary differential equations. Later in the 1970s Kreiss and others used spectral methods in order to solve partial differential equations of fluid dynamics. At present, they are becoming a competitor in some fields to well known finite difference and finite element. When the solution of the given problem is regular, then the spectral methods will be superior in terms of accuracy if compared to all the finite element or finite difference methods (Boyd (2001) and Mercier (1989)).

For all spectral methods the aim is approximating the solution of a differential equation by a finite summation of orthogonal bases functions. Exponential Fourier bases are used to expand functions in cases that we are solving periodic problems. We consider two Fourier spectral methods, the collocation method based on finding solution for certain (equally spaced) grid points on the considered domain and the Galerkin method, which consists on taking the approximate solution to be a truncated Fourier series. For simplicity, we focus on linear differential equations with constant coefficients but much of what we discuss is applicable to more complicated problems.

In Chapter 2 we introduce basic concepts. We look at orders of convergence and see that for C^∞, analytic and entire function the coefficients of Fourier series decay quickly as N increase. Later on, we move to trigonometric polynomial interpolation and demonstrate in Chapter 3 that this can be efficiently done using the fast Fourier transform (FFT) to find the coefficients. The FFT reduce complexity and computational time to find these coefficients.

In Chapter 4 we move to the process of solving ODEs subject to periodic boundary conditions. For the collocation method we use the interpolation of the solution to find the so called differentiation matrix (Gottlieb and Orszag, 1977). This matrix allow us to transform an ODE to a much simpler algebraic linear system that can be easily solved by Gauss elimination or some other matrix factorisation. Galerkin takes the solution as a finite sum of a Fourier series which we can get using the IFFT.

In Chapter 5 we move to PDEs. Firstly, for time dependent PDEs, we use the theory applied to ODEs to solve the IBVP, but we take the coefficients of the Fourier series to be depending on time. For this case the discretization is just applied to the variable representing displacement (space).

On periodic square domains the process of approximating solutions of BVPs, is based on looking for soluiton in the form given by Equation (5.2.1), which is the two dimensional Fourier series (Kopriva, 2009). The collocation method as seen before uses the differentiation matrix, this time transforming an PDE problem to a matrix equation that can be reduced to Sylvester equation form. The Galerkin method as previously stated for ODEs is based on truncation, and this time we find the matrix of coefficients that can done using 2D *Python* implementation of IFFT or solving matrix equations.

Finally, we conclude and present some future work ideas.

2. Fourier approximation

2.1 Fourier Series

According to Duoandikoetxea and Zuazo (2001), in the middle of the 18th century Bernoulli stated the Fourier series trying to solve the problem of a vibrating string, and the coefficients appeared in an article by Euler in 1777 (Körner, 1989). Later on, the French mathematician and physicist Jean-Baptiste Fourier (1768–1830) came out with major contributions to the study of trigonometric series, when developed a theory of solving the heat equation in a metal plate, and published his book with initial results around 1807 (Peetre, 2000). Actually, it is a very useful tool when using the method of separation of variables and related methods to solve partial differential equations.

2.1.1 Definition (Duoandikoetxea and Zuazo (2001))**.** The Fourier series of a function $f(x)$ of period $2L$ is given by

$$\frac{a_0}{2}+\sum_{n=1}^{\infty}\left[a_n\cos\left(\frac{\pi nx}{L}\right)+b_n\sin\left(\frac{\pi nx}{L}\right)\right], \tag{2.1.1}$$

with

$$a_n=\frac{1}{L}\int_a^{a+2L} f(x)\cos\left(\frac{\pi nx}{L}\right)\mathrm{d}x, \qquad (n=0,1,2,3,\cdots), \tag{2.1.2}$$

$$b_n=\frac{1}{L}\int_a^{a+2L} f(x)\sin\left(\frac{\pi nx}{L}\right)\mathrm{d}x, \qquad (n=1,2,3,\cdots). \tag{2.1.3}$$

For every point x where $f(x)$ is continuous, the function will be equal to the corresponding Fourier series if satisfies the following conditions:

- be absolutely integrable, that is
$$\int_a^{a+2L}|f(x)|\,\mathrm{d}x\leq\infty.$$
- have a finite number of discontinuities in every bounded interval, and the discontinuities must be finite.

The Fourier series can also be expressed in exponential form with complex coefficients,

$$f(x)=\sum_{-\infty}^{\infty}c_n e^{\frac{\pi nix}{L}} \quad (n=0,\pm1,\pm2,\cdots), \tag{2.1.4}$$

where the coefficients are given by

$$c_n=\frac{1}{L}\int_a^{a+2L} f(x)e^{-\frac{\pi nix}{L}}\,\mathrm{d}x, \quad (n=0,\pm1,\pm2,\cdots). \tag{2.1.5}$$

To obtain an approximation of $f(x)$, we can truncate the series of f

$$f_N(x)=\sum_{n=-N}^{N}c_n e^{\frac{\pi nix}{L}}. \tag{2.1.6}$$

We call $f_N(x)$ a trigonometric polynomial. We will see that typically the coefficients c_n decay rapidly as $|n|\to\infty$, hence $f_N(x)\approx f(x)$.

2.1.2 Orders of convergence. We consider definitions based on the asymptotic behavior of the series coefficients for large n. They may lead to wrong conclusions if one apply for small values of n (Boyd, 2001).

A sequence $c_n, n = 0, 1, 2, \ldots$ is said to converge *algebraically* if

$$\lim_{n\to\infty} |c_n| n^{\alpha} < \infty, \tag{2.1.7}$$

where α is the algebraic index of convergence (Boyd, 2001). From (2.1.7), equivalently α is the algebraic index of convergence if

$$c_n \sim \mathcal{O}(1/n^{\alpha}). \tag{2.1.8}$$

A sequence $c_n, n = 0, 1, 2, \ldots$ is said to converge *exponentially (spectrally)* if

$$\lim_{n\to\infty} n^{\alpha} e^{-qn^{\beta}} = 0, \quad \alpha, \beta > 0, \tag{2.1.9}$$

where β is the exponential index of convergence and

$$\beta = \lim_{n\to\infty} \frac{\log|\log|c_n||}{\log(n)}.$$

A series c_n has *supergeometric*, *geometric* or *subgeometric* rate of exponential convergence if

$$\lim_{n\to\infty} \frac{\log(|c_n|)}{n} = \begin{cases} \infty & \text{supergeometric} \\ constant & \text{geometric} \\ 0 & \text{subgeometric,} \end{cases} \tag{2.1.10}$$

or alternatively

- if $c_n \sim \mathcal{O}(e^{-qn^{\beta}})$, $\beta < 1$ then c_n has subgeometric convergence;
- if $c_n \sim \mathcal{O}(e^{-qn})$ then c_n has geometric convergence;
- if $c_n \sim \mathcal{O}(e^{-qn^{\beta}})$, $\beta \geq 1$ then c_n has supergeometric convergence.

For geometric convergence, q is said to be the *asymptotic rate of geometric convergence*.

We now move to give a graphical representation of orders of convergence for *log-linear* and *log-log* using the following sequences in their order of smoothness:

- $c_n = 1/n^2$ for algebraic,
- $c_n = e^{-1.5n^{2/3}}$ for subgeometric,
- $c_n = e^{-n}$ for geometric,
- $c_n = e^{-(n)\log n}$ for supergeometric.

In log-linear scale the coefficients converging geometrically are presented in a straight line, while in log-log scale such behaviour is expected from coefficients converging algebraically.

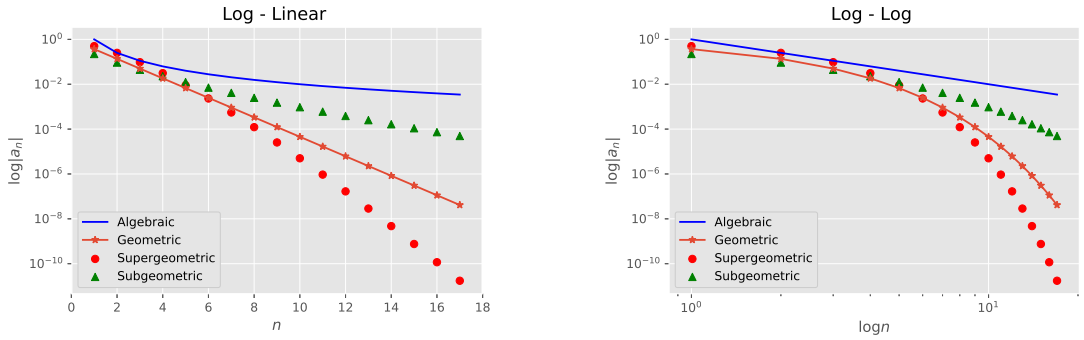

Figure 2.1: Orders of convergence.

2.1.3 Theorem (Boyd (2001)). *If $f(x), f'(x), f''(x), \cdots, f^{(k-2)}(x)$ are continuous, $f^{(k)}(x)$ is integrable and*

$$f(a) = f(a+2L), f'(a) = f'(a+2L), f''(a) = f''(a+2L), \cdots, f^{(k-2)}(a) = f^{(k-2)}(a+2L)$$

then the coefficient Fourier series defined in (2.1.4) have upper bound

$$|c_n| \leq \frac{C}{n^k} \tag{2.1.11}$$

for some sufficiently large constant C which is independent of n.

2.1.4 Theorem (Boyd (2001)). *Let $f(x) \in C^\infty(]a, a+2L[)$ and periodic. Then the coefficient of Fourier series defined in the Equation (2.1.5) is exponentially bounded, that is*

$$|c_n| \leq \frac{C}{e^{qn}} \tag{2.1.12}$$

for some sufficiently large constant C which is independent of n.

2.1.5 Numerical experiments. We present some numerical experiments, checking the order of convergence for the functions. We consider the functions $f(x) = |x|^3$, $g(x) = 1/(2+\sin(x))$ and $h(x) = e^{\sin(x)}$, which are presented in their order of smoothness. We take c_n for nonegative n, in spite of that we expect the same behaviour for negative n. Note that we compute these coefficients given by (2.1.5) using the trapezoidal rule.

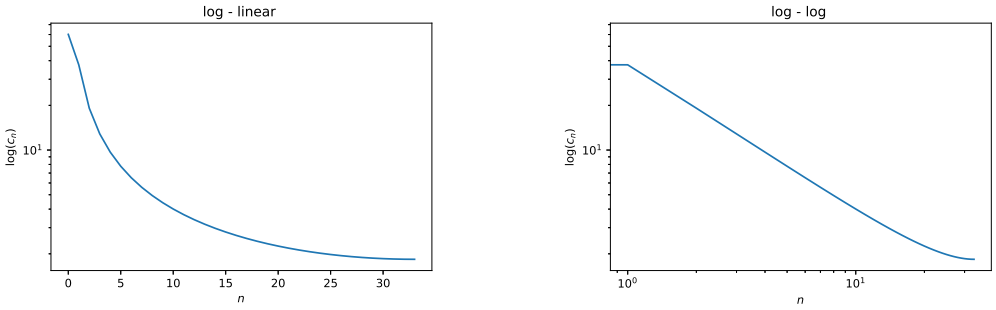

Figure 2.2: Orders of convergence of coefficients of $f(x) = |x|^3$. Algebraic convergence is observed.

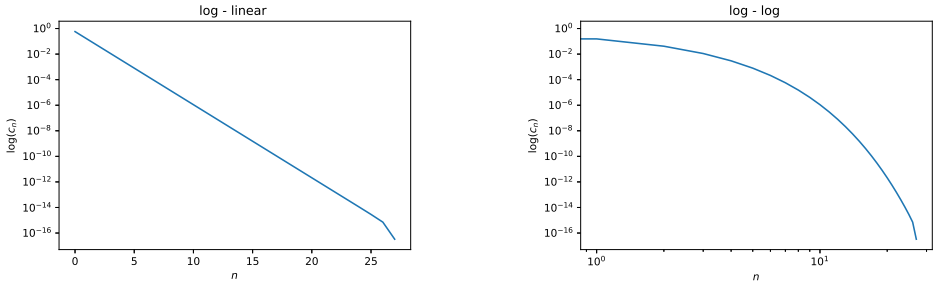

Figure 2.3: Orders of convergence of coefficients of $g(x) = 1/(2 + \sin(x))$. Geometric convergence is observed.

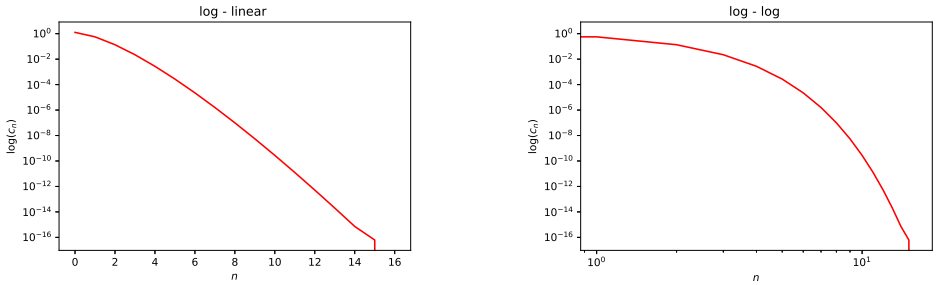

Figure 2.4: Orders of convergence of coefficients of $h(x) = e^{\sin(x)}$. Supergeometric convergence is observed.

Since f has lower order of smoothness that is $f \in C^2$, forced by $|x|$, on Figure 2.2 we get algebraic order of convergence of its coefficients, also we get a straight line as expected for the $log - log$ scale.

For the analytic function g we get geometric convergence, its coefficients decay more rapidly than the ones of f.

And finally,on Figure 2.4 we have supergeometric order of convergence of Fourier coefficients of the entire function h. Moreover, we see that $\log |c_n|$ for h with $n = 15$ decay to 10^{-16}, in contrast g needs $n > 30$ to reach that level.

The numerical experiments above emphasise that the smoothness of a function has a direct influence in the rapid decay of its Fourier coefficients. As seen we get exponential decay rate of coefficients when we have analytical and entire functions, exactly geometric and supergeometric convergence respectively.

2.1.6 Differentiation of the Fourier series. The key property of Fourier series in their usefulness for solving differential equations is given below. Which is done by differentiating the series term by term.

2.1.7 Theorem (Anhaouy (1997)). *Let f be the function represented in (2.1.4) such that is continuous for $a \leq x \leq a + 2L$, $f(a) = f(a + 2L)$ then is differentiable at each point $a < x_0 < a + 2L$ and is given by*

$$f'(x_0) = \frac{\pi i}{L} \sum_{-\infty}^{\infty} n c_n e^{\frac{\pi n i x_0}{L}}, \quad (n = 0, \pm 1, \pm 2, \cdots). \tag{2.1.13}$$

This theorem play an important role, allowing us to define the $j-$th derivative of f at point x_0 as

$$f^{(j)}(x_0) = \left(\frac{\pi i}{L}\right)^j \sum_{-\infty}^{\infty} n^j c_n e^{\frac{\pi n i x_0}{L}}.$$

2.2 Trigonometric polynomial interpolation

2.2.1 Definition. Given that $f(x)$ has period $2L$, we can approximate $f(x)$ by a *trigonometric polynomial* of degree $2N+1$ that requires

$$f(x_n) = p_n(x_n), \quad a = x_0 < x_1 < x_2 < \cdots < x_{2N} = a + 2L, \tag{2.2.1}$$

as:

$$p_N(x) = \sum_{n=-N}^{N} \hat{c}_n e^{\frac{\pi n i x}{L}}, \tag{2.2.2}$$

where $\hat{c}_n$ is given by (3.2.1).

The polynomial $p_N(x)$ is then called the trigonometric polynomial interpolant.

We take the x_j points to be evenly spaced, that means $x_j = a + j\frac{2L}{2N}$, $j = 0, 1, \cdots, 2N$, and the coefficients $\hat{c}_n$ are defined as

$$\hat{c}_n = \frac{1}{2N} \sum_{j=0}^{2N} p_N(x_j) e^{-\frac{\pi n i x_j}{L}}.$$

Note that the interpolation condition guarantee that we can swap p_N by f only on the interpolation points. The formula above can be related to the Discrete Fourier transform which we will formally present in Subsection 3.2. Notice that $\hat{c}_n$ is exactly the trapezoidal rule applied to approximate (2.1.5)

2.2.2 Trigonometric interpolation convergence. According to Anhaouy (1997), as the numbers of terms included of points of interpolation increase it will reduce the error, that is

$$p_N(x) \to f(x), \quad \text{as } N \to \infty.$$

It is important to know the maximum error that we get by this approach, we use $L_\infty([a, a+2L])$ norm to compute it.

2.2.3 Theorem (Boyd (2001))**.** *Let $p_N(x)$ be the trigonometric interpolat of a function $f(x)$ at $2N+1$ points. Then*

$$\| f(x) - p_N(x) \|_\infty \leq \sum_{|n|>N} |c_n|. \tag{2.2.3}$$

The theorem above establish that the error is bounded by the summation of the absolute value of all coefficients that were not included in the interpolation.

Now we can be more precise in estimating the error $f(x) - p_N(x)$, using the Theorems 2.1.3 and 2.2.3 we get the following corollary.

2.2.4 Corollary. Let k be a positive integer. If $f(x) \in C^k([a, a+2L])$ and $f^{(k')}(a) = f^{(k')}(a+2L)$, $0 \leq k' \leq k-1$, then

$$|f(x) - p_N(x)| \leq \mathcal{O}\left(\frac{1}{N^k}\right). \tag{2.2.4}$$

Moreover, and if $f(x) \in C^\infty(]a, a+2L[)$,then the Theorem (2.1.4) applies to get

$$|f(x) - p_N(x)| \leq \mathcal{O}\left(\frac{1}{e^{qN}}\right). \tag{2.2.5}$$

This means that the trigonometric polynomial interpolation error only depends on how smooth is the function to be approximated.

2.2.5 Numerical experiment. We look on the trigonometric polynomial interpolation of the function $f(x) = e^{\sin(x)}$. We consider 4 and 16 grid points for interpolation.

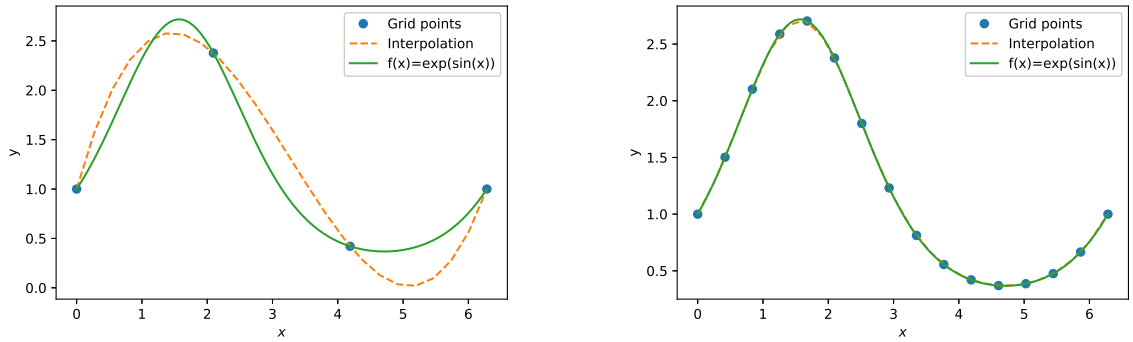

Figure 2.5: Trigonometric polynomial interpolation of the function $f(x) = e^{\sin(x)}$, with 4 grid points (above) and 16 grid points (below).

We can see that the interpolation with 16 points is very close to the function that we are interpolating, hinting at exponential rate of convergence.

So, let us move to the plot that present $log|error|$ versus N, we have that N is the number of interpolation points.

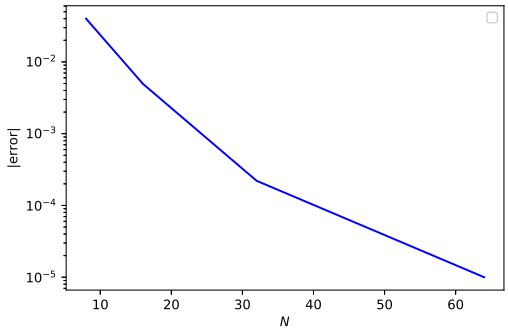

Figure 2.6: Trigonometric polynomial interpolation error of the function $f(x) = e^{\sin(x)}$. Subgeometric convergence is observed, this is a expected result from the Corollary 2.2.4.

As expected, on the Figure 2.6 we see that there is exponential decay rate, exactly subgeometric convergence.

Numerical results on other functions such as discussed in Section 2.1.5 yielded similar results, this validate the Corollary 2.2.4.

3. Fourier Transform and Fast Fourier Transform

In this section we discuss the Fourier transform (FT), the Discrete Fourier Transform (DFT) and the fast Fourier transform (FFT) which will help us to decrease the complexity of computing the values of DFT, from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log(N))$ when finding the coefficients for the Fourier approximation or the values of an interpolant [(Boyd, 2001) and (Trefethen, 2000)].

3.1 The Fourier Transform

3.1.1 Definition. The Fourier Transform (FT) and the Inverse Fourier Transform (IFT) are respectively defined as

$$\mathcal{F}(f)(x) = F(x) = \int_{-\infty}^{\infty} f(t)e^{\frac{-\pi ixt}{L}}\,\mathrm{d}t, \tag{3.1.1}$$

and

$$\mathcal{F}^{-1}(F)(t) = f(t) = \frac{1}{2L}\int_{-\infty}^{\infty} F(x)e^{\frac{\pi ixt}{L}}\,\mathrm{d}x. \tag{3.1.2}$$

The existence of FT in Equation (3.1.1) is subject to the condition that $f(t)$ is integrable, that is

$$\int_{-\infty}^{\infty} |f(t)|\,\mathrm{d}t < \infty.$$

3.1.2 Properties. Now we highlight some properties of FT according to Brigham (1974).

1. **Linearity:** If $f(t)$ and $g(t)$ have the FT $F(x)$ and $G(x)$ respectively, then $[f+g](t)$ has the FT $[F+G](x)$, or simply
$$\mathcal{F}(f+g)(x) = \mathcal{F}(f)(x) + \mathcal{F}(g)(x).$$

2. **Time scaling:** If $f(t)$ has FT $\mathcal{F}(f)(x)$, then the FT of $f(kt)$ $(k \neq 0)$ is defined as
$$|k|\mathcal{F}(f(kt))(x) = \mathcal{F}(f)\left(\frac{x}{k}\right).$$

3. **Time shifting:** If $f(t)$ has FT $\mathcal{F}(f)(x)$, then the FT of $(t-t_0)$ becomes
$$\mathcal{F}(f(t-t_0))(x) = e^{\frac{-\pi ixt_0}{L}}\mathcal{F}(f)(x),$$
where L is half of the period of f.

4. **Differentiation:** If $f(t)$ is a continuous differentiable function, and both f and its derivative f' are integrable. Then the FT of the derivative is given by
$$F'(x) = 2\pi ixF(x),$$
and more generally the $n-$th derivative of FT becomes
$$F^{(n)}(x) = (2\pi ix)^n F(x). \tag{3.1.3}$$

3.2 Discrete Fourier Transform

3.2.1 Definition (Brigham (1974)). The Discrete Fourier Transform (DFT) is defined as

$$X_j = \sum_{k=0}^{N-1} x_k e^{-ik\left(\frac{2\pi}{N}j\right)}, \quad j = 0, 1, \cdots, N-1 \tag{3.2.1}$$

and the Inverse Discrete Fourier Transform (IDFT) by

$$x_k = \frac{1}{N} \sum_{j=0}^{N-1} X_j e^{ik\left(\frac{2\pi}{N}j\right)}, \quad k = 0, 1, \cdots, N-1. \tag{3.2.2}$$

We can look at DFT as a transformation of a complex vector to another complex vector. This means that coefficients of the trigonometric polynomial interpolation $\hat{c}_n$ can be found using the DFT in (3.2.1) divided by the number of grid points considered, and the values of $p_N(x_j)$ multiplying the IDFT by N.

3.3 The Fast Fourier Transform

Now, if $w = e^{\frac{-2\pi i}{N}}$ then we can write the DFT formula in (3.2.1) as a power series in w:

$$X_j = \sum_{k=0}^{N-1} x_k w^{kj}, \quad j = 0, 1, \cdots, N-1, \tag{3.3.1}$$

moreover, (3.3.1) can easily be represented in matrix form

$$\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{N-1} \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & \dots & w^0 \\ w^0 & w^1 & \dots & w^{N-1} \\ w^0 & w^2 & \dots & w^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ w^0 & w^{N-1} & \dots & w^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix}. \tag{3.3.2}$$

We write (3.3.2) as

$$\boldsymbol{X} = \boldsymbol{W}\boldsymbol{x}.$$

Since in general $\boldsymbol{W}$ and $\boldsymbol{x}$ are complex, this implies that to obtain the DFT we are required to compute N^2 complex multiplications and $N(N-1)$ complex aditions. The Fast Fourier Transform owes its success to the fact that reduces the number of operations required to obtain the DFT in (3.3.2).

The identity $w^{kj} = w^{kj \mod N}$ is very useful in this process.

There are many FFT algorithms, we choose Cooley-Tukey formulation presented by Brigham (1974). This approach consider the number of points x_k to be in form $N = 2^r$, where r is an integer, and we implement it in *Python 3* using *numpy.fft* and *numpy.ifft* for computing DFT and IDFT respectively.

We note that the FFT can still be computed when N is not in the form of a power of two.

3.3.1 Numerical experiment. We now move to do a numerical experiment based on random vectors to compute their DFT, which give us the following situation.

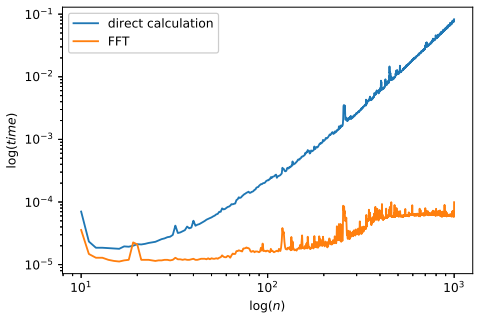

Figure 3.1: Direct computation of DFT versus FFT algorithm.

It is obvious that we save a lot of time implementing the DFT via FFT algorithm, this evident for cases where the dimension of the vector is very big.

4. Fourier Spectral Methods on ODEs

For problems that have smooth or piecewise smooth solutions, spectral methods lead to fast convergence (exponential expected), and they are used jointly with fast and stable algorithms like the FFT (Driscoll and Hale, 2015).

Consider the following second order ODE

$$\beta u(x) - u''(x) = f(x), \quad x \in [0, 2L) \tag{4.0.1}$$

where $f(x)$ is periodic and u is subject to periodic boundary conditions.

Our aim is to solve this ODE using Fourier methods.

4.1 Collocation method

First we divide the given interval by considering N equally spaced grid points, that is $\{x_k : x_k = 2Lk/N,\ k = 0, 1, \cdots, N-1\}$, and we use it to get the differentiation matrix and later on the approximate solution of the Equation (4.0.1).

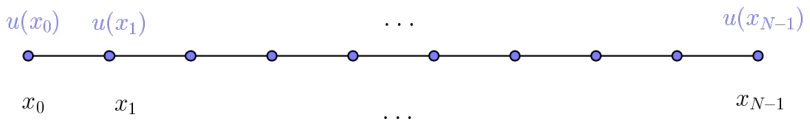

Figure 4.1: Schematic configuration of points.

4.1.1 Differentiation matrices. We now turn to spectral differentiation on a bounded, periodic grid. On Figure 4.1, the points are equally spaced and the difference between them is $h = 2L/N$. We look for a way to get the derivatives of the interpolation.

To interpolate $u(x)$ we can use the results given in Equation (2.2.2), or use the so called band-limited interpolant (Trefethen, 2000). Moreover, Gottlieb and Orszag (1977) establish that the interpolation of $u(x)$, $u_N(x)$, can be reduced to:

$$u_N(x) = \sum_{j=0}^{N-1} u(x_j) g_j(x), \tag{4.1.1}$$

where

$$g_j(\boldsymbol{x}) = \frac{1}{2N} \sin\left[(2N-1)\frac{x - x_j}{2}\right] \cot\left[\frac{x - x_j}{2}\right], \quad \boldsymbol{x} = [x_0, x_1, \ldots, x_{N-1}]^T.$$

Since $u(x_j)$ are considered constants we that

$$u'_N(x_k) = \sum_{j=0}^{N-1} u(x_j) g'_j(x_k), \tag{4.1.2}$$

where

$$g_j'(x_k) = d_{k,j} = \begin{cases} \frac{(-1)^{j+k}}{2} \cot\left[\frac{x_k - x_j}{2}\right] & , j \neq k \\ 0 & , j = k \end{cases}. \tag{4.1.3}$$

Hence

$$u_N'(\boldsymbol{x}) = D' u_N(\boldsymbol{x}),$$

where D' is the differentiation matrix composed by elements $d_{k,j}$.

The matrix D' has dimension $N \times N$, traceless and is antisymmetric.

More generally, we use the same idea to get the n-th order differentiation matrix, $D^{(n)}$, that is

$$d_{k,j}^{(n)} = g_j^{(n)}(\boldsymbol{x}). \tag{4.1.4}$$

It is evident that in the process of solving ODE in the form of the Equation (4.0.1) the second order differentiation matrix is needed. $D^{(2)}$ can be formed by $D' \times D'$, but this is inefficient and can lead to inaccuracies. Instead, we observe that $D^{(2)}$ is composed by elements defined as

$$g_j''(x_k) = d_{k,j}'' = \begin{cases} \frac{(-1)^{j+k}}{2} \csc^2\left(\frac{x_k - x_j}{2}\right) & , j \neq k \\ -\frac{2N^2+1}{6} & , j = k \end{cases}, \tag{4.1.5}$$

which is basically the implementation of (4.1.4).

Notice that the differentiation matrix $D^{(2n)}$ is symmetric and $D^{(2n+1)}$ is antisymmetric.

At this point we see that the complexity of getting the derivative of an interpolation is of order $\mathcal{O}(N^2)$, but Trefethen (2000) and Gottlieb and Orszag (1977) demonstrate that the FT property given by Equation 3.1.3 and (2.1.13) are useful, since it allow us to use FFT on determining the derivative. Therefore the complexity is reduced to $\mathcal{O}(N \log(N))$.

4.1.2 Example. The spectral derivative of $f(x) = e^{\sin(x)}$ using the differentiation matrix is given by the following figure.

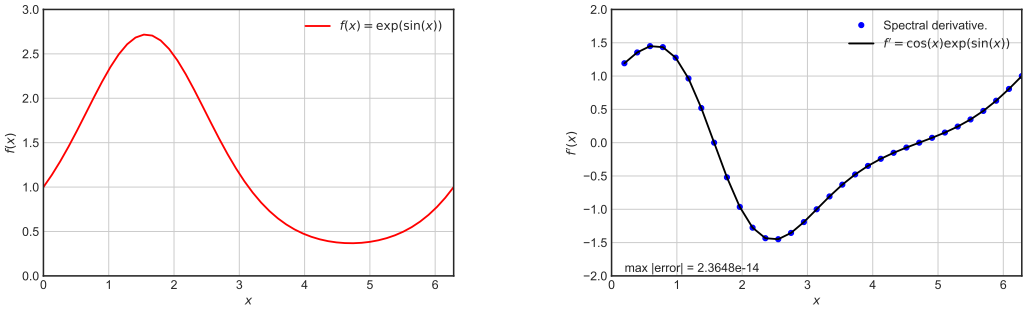

Figure 4.2: The function $f(x) = e^{\sin(x)}$ and its spectral derivative with $N = 32$ grid points.

In this numerical experiment we use $L_\infty([0, 2\pi])$ norm to get the error. Which is the maximum absolute value of the difference betwen the spectral derivative and the real derivative of the function considered.

Comments: Trefethen (2000) determined the derivative of this function with $N = 24$ grid points to get the $\max |error| = 9.6878e - 13$, in our experiment with $N = 32$ grid points we get the outcome

$\max|error| = 2.3648e-14$. This support the idea that the error is decreasing as we increase the number of grid points.

The second order derivative of the entire function above is presented in the figure below for $N = 24$ and $N = 36$.

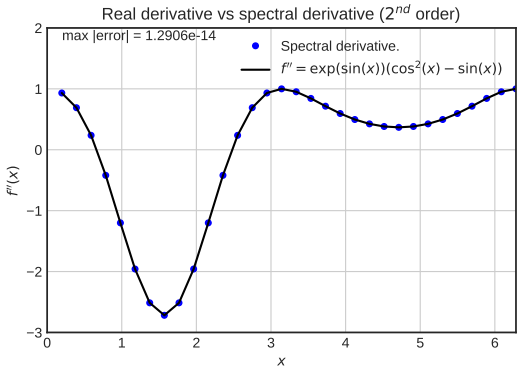

Figure 4.3: Real derivative vs spectral derivative (2^{nd} order) of $f(x) = e^{\sin(x)}$ for $N = 24$ (left) and $N = 36$ (right).

Comments: On Figure 4.3, we show that in fact the spectral differentiation has high accuracy since the difference between the dots and the continous line.

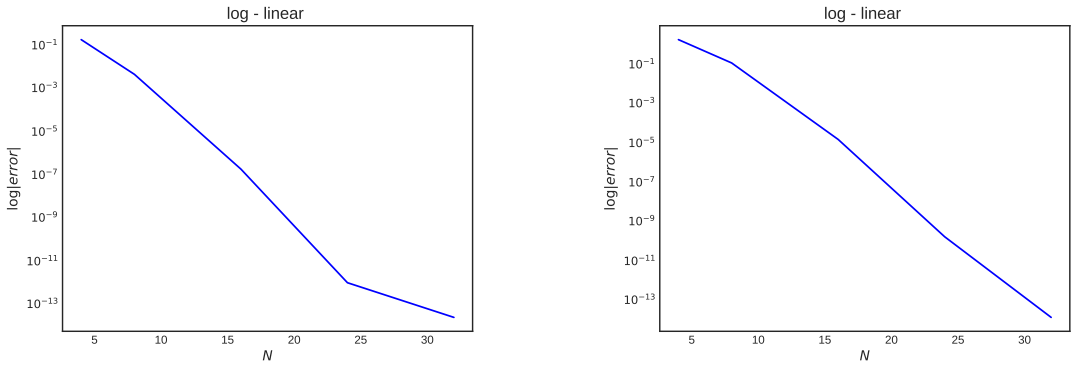

Figure 4.4: Convergence of the spectral derivative of $f(x) = e^{\sin(x)}$, first order and second order respectively.

From the plots above, we verify that for the entire function $f(x) = e^{\sin(x)}$ we get supergeometric convergence.

Notice that using the differentiation matrix we consider all grid points to get the derivative in a single point, this is the feature that make this method accuracy high. Finite difference methods just consider points in the neighbourhood to find the derivative.

4.1.3 Implementing the collocation. We approximate the Equation (4.0.1) on the grid points by using the second order differentiation matrix for $u''(x_k)$, getting

$$\beta u(x_j) - D^{(2)}_{jk} u(x_k) = f(x_j), \quad \forall j = 0, 1, \cdots, N-1.$$

Let $\boldsymbol{u} = (u(x_0), u(x_1), \cdots, u(x_{N-1}))^T$, and $\boldsymbol{f} = (f(x_0), f(x_1), \cdots, f(x_{N-1}))^T$.

Then we have

$$\beta \boldsymbol{u} - D^{(2)}\boldsymbol{u} = \boldsymbol{f} \Rightarrow (\beta I_N - D^{(2)})\boldsymbol{u} = \boldsymbol{f}, \tag{4.1.6}$$

where I_N is the identity matrix of dimension N. In Equation (4.1.6), we have a linear system that is dense. We can solve it by using matrix factorization or Gauss elimination method.

4.1.4 Example. We use the collocation method to solve the ODE given by

$$u'' - u = e^{\sin x}, \qquad x \in [0, 2\pi]. \tag{4.1.7}$$

We solve the problem above using Python, and we impose the following periodic boundary conditions.

The analytic solution of this problem is unknown, then we take $N = 300$ to be the "approximate" exact solution.

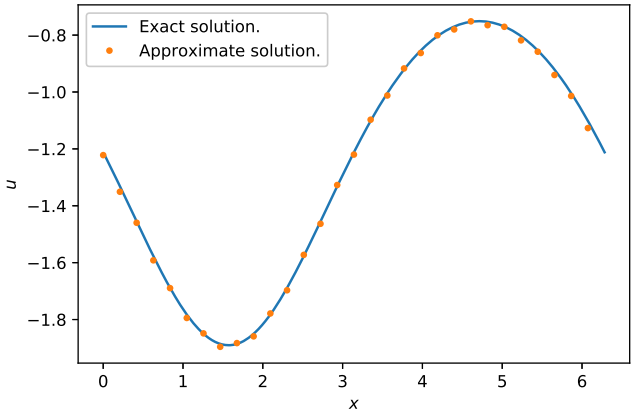

Figure 4.5: The approximate solution using the spectral collocation method with $N = 32$ grid points.

We see that with 32 grid points we a fair approximation of the exact solution. Notice that our problem (4.1.7) does not have analytic solution, therefore we take $N = 300$ to be the exact solution.

Now we present the error of approximating the solution of the Boundary Value Problem using the collocation method.

On Figure 4.6 we have got a geometric convergence, meaning that as we increase N the error of approximation of (4.1.7) decay exponentially.

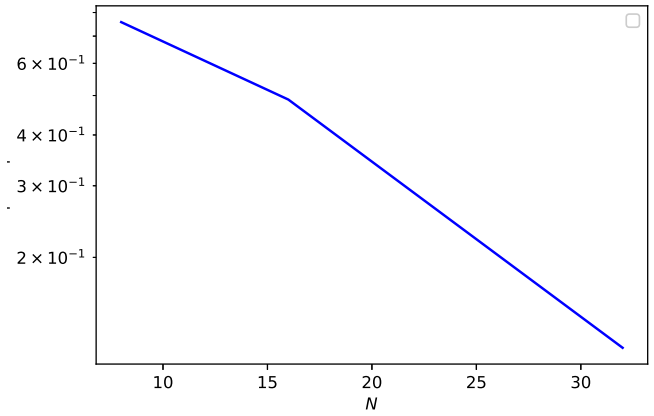

Figure 4.6: Error of computing the approximate solution of (4.1.7) using the spectral collocation method with N terms.

4.2 Galerkin method

We look for the solution of (4.0.1) in the form $u_N = \sum_{j=-N/2}^{N/2} v_j e^{ijx}$. Taking the inner product with e^{ikx}, we get

$$\beta v_j \delta_{jk} + v_j j^2 \delta_{jk} = f_k, \quad f_j = \langle f, e^{ijx} \rangle, \quad k = -N/2, \cdots, 0, \cdots, N/2$$

i.e.

$$\beta v_k + k^2 v_k = f_k$$

which give us

$$v_k = \frac{f_k}{\beta + k^2} \tag{4.2.1}$$

therefore the solution is given by

$$u_N(x) = \sum_{j=-N/2}^{N/2} v_j e^{ijx} = \sum_{j=-N/2}^{N/2} \frac{f_k}{\beta + k^2} e^{ijx}$$

In summary, the procedure of Garlekin method is based in the following steps:

1. use FFT to calculate inner product to get the f_j;
2. calculate the coefficients v_j using the formula (4.2.1);
3. calculate the approximate solution u_N on the grid, using the IFFT.

4.2.1 Remark. As the algorithm above suggests, it consists in proposing the solution as a trigonometric polynomial which minimizes the error.

4.2.2 Example. We use the Galerkin method to solve the ODE given by Equation (4.1.7). The implementation in *Python* gives us the following solution.

As seen before, the problem that we are solving does not have an analytic solution, again we take the approximation for $N = 300$ as the exact solution.

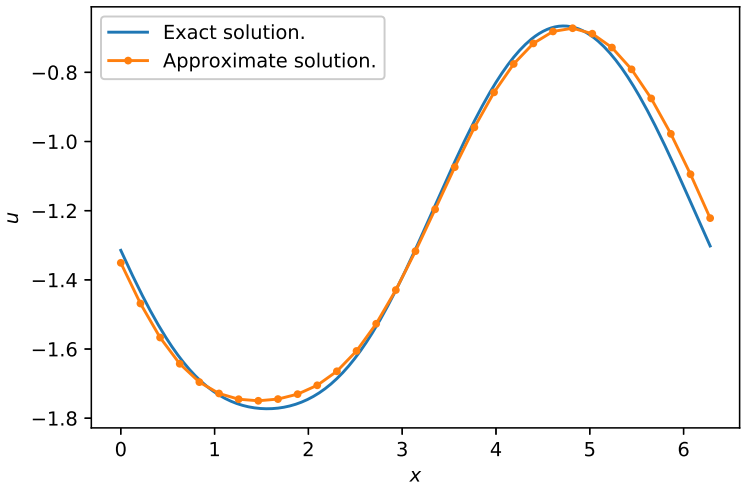

Figure 4.7: The approximate solution of (4.1.7) using the Galerkin method with $N = 32$ grid points.

The error of approximating the solution of the ODE 4.1.7, using the Galerkin method is given by the following plot.

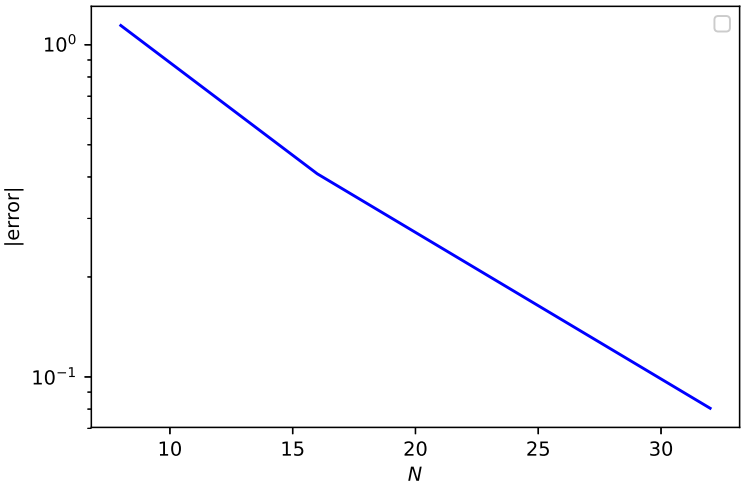

Figure 4.8: Error of computing the approximate solution of (4.1.7) using the Galerkin method with N terms.

In this method, the asymptotic behaviour is almost the same as the one from spectral collocation i.e, we get spectral convergence (geometric).

5. Fourier Spectral Methods on PDEs

In this chapter, first we present the Fourier spectral for time dependent PDEs considering $u(x,t)$, and later time independent PDEs are discussed on square domains, $u(x,y)$.

5.1 Time dependent PDEs

Let $u(x,t)$ be periodic in x and consider the PDE

$$\frac{\partial u}{\partial t} = S(u) = F\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2} \cdots\right), \tag{5.1.1}$$

where $S(u)$ is a linear operator. Equally important is the fact that the boundary conditions are periodic like u. An initial value $u(x,o)$ can be considered, leading us to an initial value boundary problem (IVBP) that is periodic.

Even though u has got two variables, it still possible to solve the IVBP above using the methods discussed for ODEs in the previous chapter, for instance using the collocation method and the Galerkin method. We take the basis to depend only on x and the coefficients to depend on time, i.e.,

$$u(x,t) = \sum_{k=-\infty}^{\infty} c_k(t) e^{-ikx}.$$

5.1.1 Collocation method. As might be expected, we approximate the solution of the Equation 5.1.1 on the grid points on Figure 4.1. The differentiation matrix is used for derivatives in x, however not applied for t. Consequently, the Equation 5.1.1 becomes a semi-discrete problem

$$\frac{\partial}{\partial t} u(x_j,t) = S(u(x_j,t)) = F\left(u(x_j,t), D'u(x_j,t), D^{(2)}u(x_j,t) \cdots\right). \tag{5.1.2}$$

After the transformation above on the grid points, since we know that

$$\frac{\partial}{\partial t} u(x_j,t) = \lim_{\Delta t \to 0} \frac{u(x_j, t+\Delta t) - u(x_j,t)}{\Delta t}$$

is useful to take the approximation

$$\frac{\partial}{\partial t} u(x_j,t) \approx \frac{u(x_j, t+\Delta t) - u(x_j,t)}{\Delta t}. \tag{5.1.3}$$

This will help us to discretise in time and use methods like Euler, Runge-Kutta or exponential integrators.

5.1.2 Example. We consider the linear Heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad x \in [0, 2\pi], \tag{5.1.4}$$

with periodic boundary conditions, and the initial condition

$$u(x,0) = e^{\sin(x)}.$$

For instance using the collocation method we transform (5.1.2) into a system of ordinary differential equations

$$\frac{\partial}{\partial t}u(x_j,t) = D^{(2)}u(x_j,t),$$

for some fixed t.

That can be implemented using, for example, *scipy.integrate.solve_ivp* in *Python*.

5.1.3 Galerkin method. We choose the solution of the Equation (5.1.1) in the form

$$u_N(x,t) = \sum_{k=-N/2}^{N/2} c_k(t)e^{-ikx}.$$

Afterwards we just have to find the coefficients c_k, which can quickly be done by using the FFT at each value of t, (Canuto et al. (2006) and Kopriva (2009)).

If we plug in $u_N(x,t)$ into Equation 5.1.1 and apply the Fourier transform we get

$$c_k'(t) \equiv \frac{1}{2\pi}\int_0^{2\pi} \frac{\partial u_N(x,t)}{\partial t}e^{ilx}\mathrm{d}x = \frac{1}{2\pi}\int_0^{2\pi} S(u_N(x,t))e^{ilx}\mathrm{d}x. \tag{5.1.5}$$

From the ODEs in (5.1.5) we get the coefficients, to this end we compose the trigonometric polynomial approximating the solution of (5.1.1).

5.2 PDEs on a square domain

In this section, we look on describing the Fourier spectral methods on the square domains, $(x,y) \in [a,a+2L]\times[a,a+2L]$. This is a good start that can be easily extended to more general rectangular domains. This approach turns out to be useful to study mathematical physics problems that are modeled by PDEs such as the Poisson equation, the advection-difusion equation,as well as systems of conservation laws (Kopriva, 2009).

The two space dimensions Fourier series of a periodic function $u(x,y)$ is defined by

$$u(x,y) = \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} c_{nm}e^{-inx}e^{-imy}, \tag{5.2.1}$$

where

$$c_{nm} = \frac{1}{(2L)^2}\int_a^{a+2L}\int_a^{a+2L} u(x,y)e^{i(nx+my)}\mathrm{d}x\mathrm{d}y. \tag{5.2.2}$$

Using the same argument as used in the second chapter of this essay, we can get the truncated Fourier series u_{NM}. The same applies for the Fourier interpolation in two space dimensions.

Note that the periodicity of $u(x,y)$ implies that

$$u(x,y) = u(x+2L,y) = u(x,y+2L).$$

Consider the PDE

$$\nabla^2 u = u_{xx} + u_{yy} = f(x,y), \qquad (x,y)\in(0,2\pi)\times(0,2\pi), \tag{5.2.3}$$

under periodic boundary conditions. These boundary conditions may not seem relevant in many physical situations, but they are in fact used when modellers do not know what kind of boundary conditions they really should be using.

Next, we are going to consider the approximate solutions of (5.2.3), first using the Fourier collocation method and later on the Galerkin method with Fourier basis in two dimensions.

5.2.1 Collocation method. We know that the collocation method has a starting point on grid of points (x_j, y_k). We consider the x_j's equally spaced as well as the y_k's, situation shown in the following figure.

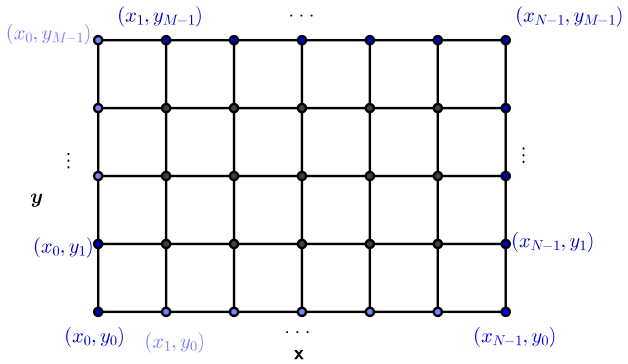

Figure 5.1: Schematic configuration of points in two space dimensions.

For simplicity, we take a square domain, that is a $M = N$ on Figure 5.1.

To derive the approximate solution of the Equation 5.2.3, we need to get the spectral derivative. The differentiation matrices defined for ODEs are the ones used for PDEs as well. This leads to

$$u_{xx}(x_j, y_k) = \sum_{l=0}^{N-1} D_{jl}^{(2),x} u(x_j, y_k),$$

and

$$u_{yy}(x_j, y_k) = \sum_{l=0}^{N-1} D_{kl}^{(2),y} u(x_j, y_k),$$

where $D_{kl}^{(2)}$ is the second order spectral differentiation matrix.

Finally, the Poisson equation has its approximate solution recovered from the matrix equation

$$\sum_{l=0}^{N-1} D_{jl}^{(2),x} u(x_j, y_k) + \sum_{l=0}^{N-1} D_{kl}^{(2),y} u(x_j, y_k) = f(x_j, y_k) \qquad j, k = 0, 1, 2, \cdots, N-1. \tag{5.2.4}$$

The solution of (5.2.4), is computed using the *Python* command *numpy.linalg.solve-sylvester* because the aforementioned can be transformed to

$$\boldsymbol{U}\boldsymbol{D}^{(2)} + \boldsymbol{D}^{(2)}\boldsymbol{U} = F,$$

which takes the Sylvester equation form.

5.2.2 Example. Now we move on to *python* implementation of the aforementioned method for the Poisson equation given in (5.2.3) with $f(x, y) = 1$, under periodic boundary conditions.

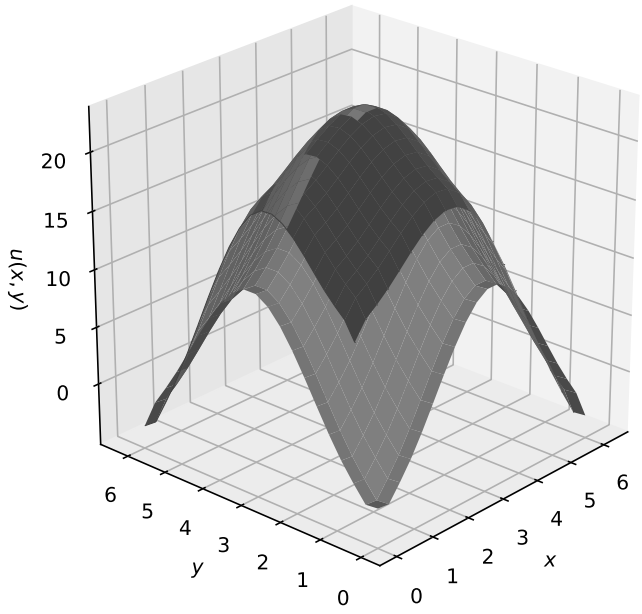

Figure 5.2: Approximate solution for the Poisson equation with $f(x, y) = 1$ under periodic boundary conditions, implemented using the spectral collocation for 24×24 grid points.

On Figure 5.2 we easily see that we a situation of periodic boundary conditions, since $u(0, y) \approx u(2\pi, y)$ and $u(x, 0) \approx u(x, 2\pi)$. For the fact we fix the conditions not explicity it may lead to a ill-posed problem because we might get different solutions for different values of N.

5.2.3 Galerkin method. Like we have done for the two previous cases, we look for solution of the Equation 5.2.3 in the form of a truncated Fourier series. Hence, we look for

$$u(x, y) \approx u_N(x, y) = \sum_{n=-N}^{N} \sum_{m=-N}^{N} c_{nm} e^{-i(nx+my)}, \tag{5.2.5}$$

where c_{nm} is defined by (5.2.2) with $a = 0$ and $L = \pi$.

We can also consider the truncated Fourier of $f(x, y)$, the right hand side of the Poisson equation, that is

$$f(x, y) \approx f_N(x, y) = \sum_{n=-N}^{N} \sum_{m=-N}^{N} b_{nm} e^{-i(nx+my)}. \tag{5.2.6}$$

Plugging in into Poisson equation the two truncated series given in (5.2.5) and (5.2.6), we get

$$\sum_{n=-N}^{N}\sum_{m=-N}^{N} c_{nm}(-n^2-m^2)e^{-i(nx+my)} = \sum_{n=-N}^{N}\sum_{m=-N}^{N} b_{nm}e^{-i(nx+my)},$$

we assume that $u(x,y)$ has continuous second derivative, and it allow us to exchange partial differentiation with an infinite sum. Now, since we have equality of two truncated Fourier series, therefore we must equate the coefficients to get

$$c_{nm} = -\frac{b_{nm}}{n^2+m^2}. \tag{5.2.7}$$

Finally, the approximate solution of the Poisson equation 5.2.3 is given by

$$u(x,y) \approx u_N(x,y) = \sum_{n=-N}^{N}\sum_{m=-N}^{N} -\frac{b_{nm}}{n^2+m^2}e^{-i(nx+my)}.$$

For implementation in *Python (numpy.fft2)*, we reduce the complexity of the problem by using the following algorithm:

1. use the FFT to compute b_{nm};
2. compute the coefficients c_{nm} using (5.2.7);
3. use the IFFT to get $u_N(x,y)$.

6. Conclusion and Future work

6.1 Conclusion

We investigated spectral methods based on Fourier series to solve ODEs and PDEs with constant coefficients. When using the Fourier collocation, the spectral differentiation turn out to be an accurate way to estimate derivatives because it uses all grid points to calculate a derivative in a single point instead of just considering the neighbours. For the Galerkin method supported by trigonometric polynomials we implemented it using FFT to get the coefficients as well as the value on certain points. We noted that the approximated solution converges quickly if the function is smooth. This happen because smooth function's coefficients decay exponentially. For PDEs we discussed the heat equation for the time dependent problem (IBVP) and the Poisson equation for the 2D BVP.

The numerical experiments were all implemented in *Python* using the libraries *NumPy* and *SciPy*. This implementation was quite challenging since there are no many biult-in functions like *MATLAB* has.

6.2 Future work

For future work, we are interested to use Fourier methods to solve more complicated PDEs some of them with real applications such as the KdV equation and Burger's equation which introduce non-linearities and non-constant coefficients. We are also interested on solving PDEs on the surface of a sphere using Double Fourier sphere method as Townsend et al. (2016) suggests.

Acknowledgements

First, I thank God Almighty for giving me the strength, knowledge, ability and opportunity to undertake the research study that lead to this essay.

Second, I take pride to acknowledge the insightful guidance of my supervisor Professor Nick Hale, for his heartfelt support and guidance at all times, as well as inspiration and suggestions in my quest for knowledge. My thanks go also to Taboka, my tutor who assisted me in this project.

I express my sincere thanks to AIMS staff for giving me this great opportunity to learn more about mathematical sciences, from people of all over the world. Thanks to the wonderful family that I found at AIMS especially Mbuso, Sebe and Thabani. We stayed together, shared ideas and make each other grow.

I also thank my siblings Ofélia, Sidy, Yola and Clédson, my nephew Enzo as well as my bride Suzy for their support during my studies.

Finally, I dedicate this essay to my parents André Mutuque and Celina Cossa.

References

Anhaouy, P. Fourier spectral methods for solving the Korteweg-De Vries equation. 1997.

Ben-Yu, G. *Spectral methods and their applications*. World Scientific, 1998.

Boyd, J. P. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

Brigham, E. O. *The fast Fourier transform*. Englewood Cliffs, N. J., Prentice-Hall, Inc., 1974.

Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. *Spectral methods*. Springer, 2006.

Chui, C. K. and Jiang, Q. Applied mathematics. *Data Compression. Spectral Methods, Fourier Analysis, Wavelets and Applications*, 2013.

Driscoll, T. A. and Hale, N. Rectangular spectral collocation. *IMA Journal of Numerical Analysis*, 36 (1):108–132, 2015.

Driscoll, T. A., Hale, N., and Trefethen, L. N. Chebfun guide, 2014.

Duoandikoetxea, J. and Zuazo, J. D. *Fourier analysis*, volume 29. American Mathematical Soc., 2001.

Dutykh, D. A brief introduction to pseudo-spectral methods: application to diffusion problems. *arXiv preprint arXiv:1606.05432*, 2016.

Gottlieb, D. and Orszag, S. A. *Numerical analysis of spectral methods: theory and applications*, volume 26. SIAM, 1977.

Hesthaven, J. S., Gottlieb, S., and Gottlieb, D. *Spectral methods for time-dependent problems*, volume 21. Cambridge University Press, 2007.

Kopriva, D. A. *Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers*. Springer Science & Business Media, 2009.

Körner, T. W. *Fourier analysis*. Cambridge university press, 1989.

Mercier, B. *An introduction to the numerical analysis of spectral methods*. Springer, 1989.

Peetre, J. On fourier's discovery of fourier series and fourier integrals. *preprint*, 2000.

Pozrikidis, C. *Introduction to finite and spectral element methods using MATLAB*. CRC Press, 2005.

Townsend, A., Wilber, H., and Wright, G. B. Computing with functions in spherical and polar geometries i. the sphere. *SIAM Journal on Scientific Computing*, 38(4):C403–C425, 2016.

Trefethen, L. N. Finite difference and spectral methods for ordinary and partial differential equations. *Unpublished text available at http://www.comlab.ox.ac.uk/nick.trefethen/pdetext.html*.

Trefethen, L. N. *Spectral methods in MATLAB*, volume 10. SIAM, 2000.

Venkatraman, R. Lecture notes: The Galerkin method. *arXiv preprint arXiv:1112.1176*, 2011.