Numerical Implementation of Fourier Galerkin Method

Quanhui Zhu

November 7, 2018

1 Introduction

The Fourier-Galerkin Method is to solve the differential equations with periodic boundary conditions. We project the solutions from infinite dimensional space into finite dimensional space $\hat{B_N} = span\{e^{inx} | n \leq N/2\}$.

The numerical solution $u_h = \sum\limits_{n=-N/2}^{N/2} \hat{u}_n e^{inx_h}$, where $x_j = \frac{2\pi j}{N}, j = 0, 1, \cdots, N-1$ are the nodes of the domain and $\hat{u}_n$ are the Fourier coefficients given by

$$\hat{u}_n = \frac{1}{2\pi} \int_0^{2\pi} u(x) e^{-inx} dx \tag{1}$$

In discrete cases, it can be rewritten by

$$\hat{u}_n = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j) e^{-inx_j} \tag{2}$$

For the differential equations

$$u_t = \mathcal{L}u + \mathcal{N}u \tag{3}$$

the numerical problem using semi-implicit scheme can be rewritten by

$$\frac{u^{n+1} - u^n}{\delta t} = \mathcal{L}u^{n+1} + \mathcal{N}u^n. \tag{4}$$

Then we calculate the problem in spectral space

$$\frac{\hat{u}_k^{n+1} - \hat{u}_k^n}{\delta t} = f(k)\hat{u}_k^{n+1} + (\hat{\mathcal{N}u^n})_k. \tag{5}$$

The matrix is diagonal.

2 Matching with Matlab FFT

In matlab the ifft function is

$$X(k) = \frac{1}{2N+1} \sum_{j=1}^{2N+1} x(j) e^{\frac{2\pi i (j-1)(k-1)}{2N+1}}, \quad k = 1 : 2N+1. \tag{6}$$

For matching

$$\hat{u}_n = \frac{1}{2N+1} \sum_{j=0}^{2N} u_j e^{\frac{2\pi i j n}{2N+1}}, \quad n = -N : N, \tag{7}$$

we shift $k = n + N + 1$, thus

$$\hat{u}_n = \frac{1}{2N+1} \sum_{j=1}^{2N+1} u_{j-1} e^{\frac{2\pi i(j-1)(k-N-1}{2N+1}} = \frac{1}{2N+1} \sum_{j=1}^{2N+1} u_{j-1} e^{\frac{-2N\pi i(j-1)}{2N+1}} e^{\frac{2\pi i(j-1)(k-1}{2N+1}} \tag{8}$$

So we can obtain $\hat{u} = ifft(\{u_{j-1} e^{\frac{-2N\pi i(j-1)}{2N+1}}\}_j)$.

Similarly, we can obtain $u = \{e^{\frac{-2N\pi i(j-1)}{2N+1}}\}_j. * fft(\hat{u})$.

3 Numerical Examples

3.1 Diffusion Equation

The equation is

$$-u_{xx} = f. \tag{9}$$

Using Fourier galerkin method with periodic boundary condition, the equation can rewritten as

$$\begin{aligned} -\sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \hat{u}_n (e^{inx})_{xx} &= \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \hat{f}_n e^{inx}; \\ i.e. \qquad n^2 \hat{u}_n = \hat{f}_n, & \qquad n = -\frac{N}{2} : \frac{N}{2}. \end{aligned} \tag{10}$$

Notice that when $n = 0$, we have $\hat{f}_0 = \frac{1}{N} \sum_j f(x_j) = 0$ which is called compatibility condition. And the numerical solution is $u_h = \sum_{n=-\frac{N}{2}, n\neq 0}^{\frac{N}{2}} \hat{u}_n e^{inx} + \hat{u}_0$, where $\hat{u}_0 = \frac{1}{N} \sum_j u(x_j)$ called stability condition should be given.

Firstly we calculate $u = sin2x$ to test. $u = sin2x$ shall be exact for any N since it is a base of B_N. When $N = 10$, the residual is less than $1e - 15$. This test can show whether your codes have grammar error.

Secondly we calculate $u = \frac{3}{5-4cosx}$ and observe how the residual changes with N changing.

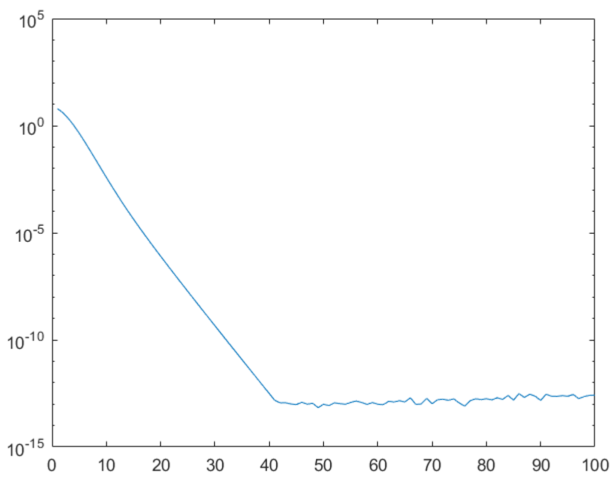

Figure 1: The x-axis is $\frac{N-1}{2}$ where N is the number of nodes and the y-axis is $ln(error)$ where the error is $||u - u_h||_{L^\infty}$.

In figure (1) it follows $||u - u_h|| = e^{-N}$ unit the error attains machine epsilon.

After these two tests finished, your program is much possible to be right and we shall use the method to solve problem now.

3.2 Allen Cahn Equation

The equation is

$$u_t = \epsilon u_{xx} + u - u^3. \tag{11}$$

Using Fourier galerkin method and semi-implicit scheme, the iteration takes the form as

$$\frac{\hat{u}_k^{n+1} - \hat{u}_k^n}{\delta t} = -\epsilon k^2 \hat{u}_k^{n+1} - (u^n - \hat{(u^n)^3})_k. \tag{12}$$

where δt depends on ϵ and h.

Here is a numerical experiment. Set $\Omega = [0, 2\pi], N = 40, \delta = \frac{1}{N^2}, T = 1, \epsilon = 0.01$ and $u(x, 0) = sin(x)$. Figure (2) shows the numerical solution.

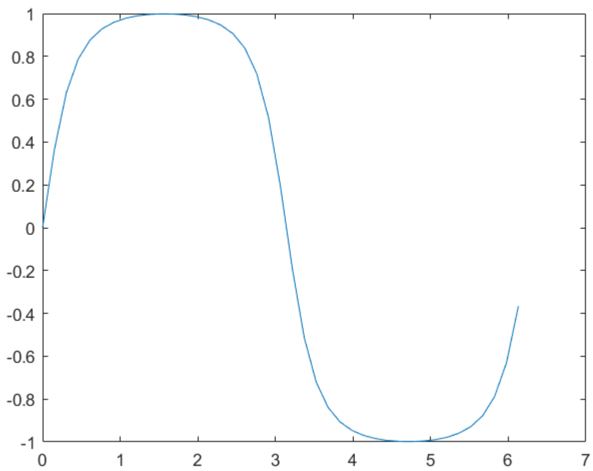

Figure 2: The Allen-Cahn equation's solution when $T = 1$ and $\Omega = [0, 2\pi], N = 40, \delta = \frac{1}{N^2}, \epsilon = 0.01, u(x, 0) = sin(x)$.