# PROJECT 2: SPECTRAL AND FINITE VOLUME METHODS

GUANG YANG

May 7, 2023

**1.** (Fourier spectral methods)
Consider the partial differential equation,

$$u_t + \sin(\pi x) u_x = \frac{1}{2} u_{xx}, \qquad u(x,0) = \sin(2\pi x), \tag{1}$$

for $u = u(x,t)$, $x \in [0,1]$, $t > 0$, and periodic boundary conditions.

(a) Write the semi-discrete form for the Fourier-Galerkin scheme for this problem. You may use a complex-valued basis (e.g., $e^{ijx}$, $i = \sqrt{-1}$) if you like, but the solution $u$ should be real-valued. Use an odd number $2N+1$ for the dimension of the spatial variable discretization space.

(b) If using Forward Euler for fully discretizing the semi-discrete form, what is the timestep restriction (say as a function of $N$) to ensure that the scheme is stable in the sense of $A$-stability? (You may investigate this analytically or empirically via the region of stability.) What aspect of the equation (1) is the dominant contributor to the timestep restriction? How does this make the total computational time of the solver scale with $N$?

(c) Implement both implicit (e.g., Crank-Nicolson) and explicit time-stepping methods (say RK-2) for solving the above problem up to time $T = 1$. Visualize the simulated results and discuss the advantages and disadvantages of each approach.

Answer:
(1): To derive the semi-discrete form for the Fourier-Galerkin scheme, we need to first discretize the spatial variable x using the Fourier basis functions. Let the basis functions be given by

$$\varphi_{k(x)} = e^{(\frac{2\pi ijk}{2N+1})},\ k = -N, -N+1, ..., N-1, N$$

where $i = \sqrt{-1}$ and $j = 1, 2, ..., 2N+1$. Note that these basis functions are complex-valued, but the solution u(x,t) is real-valued, so we will take the real part of the solution at each time step. Next, we approximate the solution $u(x,t)$ by a finite sum of these basis functions, i.e.,

$$u(x,t) \approx \sum_{K=-N}^{N} U_{k(x)} \varphi_{k(x)}$$

We now substitute this approximation into the given partial differential equation to obtain the semi-discrete form:

$$\frac{\partial}{\partial t}(\sum_{K=-N}^{N} U_{k(t)}\varphi_{k(x)}) + sin(2\pi x)\frac{\partial}{\partial x}(\sum_{K=-N}^{N} U_{k(t)}\varphi_{k(x)}) = \frac{1}{2}(\sum_{K=-N}^{N} U_{k(t)}\varphi_{k(x)})''.$$

We can simplify this expression using the Fourier basis functions and the properties of the complex exponential function. The derivatives of the basis functions can be expressed as

$$\varphi_{k'(x)} = \sqrt{-1}\frac{k\pi}{N}e^{(\frac{2\pi ijk}{2N+1})},\ k = -N, -N+1, ..., N-1, N$$

and

$$\varphi_{k''(x)} = -\pi^2\frac{k^2}{N^2}\varphi_{k(x)},\ k = -N, -N+1, ..., N-1, N.$$

Using these expressions, we obtain the following system of ordinary differential equations for the Fourier coefficients $U_{k(t)}$:

$$\frac{dU_k}{dt} + \sqrt{-1}\frac{k\pi}{N}sin(\frac{2\pi ijk}{2N+1})U_{k(t)} = -\frac{1}{2}(\frac{\pi k}{N})^2U_{(k(t))},\ k = -N, -N+1, ..., N-1, N.$$

This is the semi-discrete form for the Fourier-Galerkin scheme for the given partial differential equation with periodic boundary conditions. We can solve this system of ODEs numerically using standard numerical methods such as the Runge-Kutta method to obtain the time-evolution of the Fourier coefficients $U_{k(t)}$ and hence the solution $u(x,t)$.

(2):

To fully discretize the semi-discrete form using the forward Euler method, we approximate the time derivative by a finite difference, using a time step $\Delta t$. This gives us the following update rule for the Fourier coefficients $U_{k^n}$ at the nth time step:

$$U_{k^{n+1}} = U_{k^n} - \Delta t\sqrt{-1}\frac{k\pi}{N}sin(\frac{2\pi kj}{2N+1})U_{k^n} - \frac{1}{2}(\frac{\pi k}{N})^2\Delta t U_{k^n}.$$

To ensure the stability of the forward Euler method, we need to ensure that the spectral radius of the iteration matrix $A$, defined by

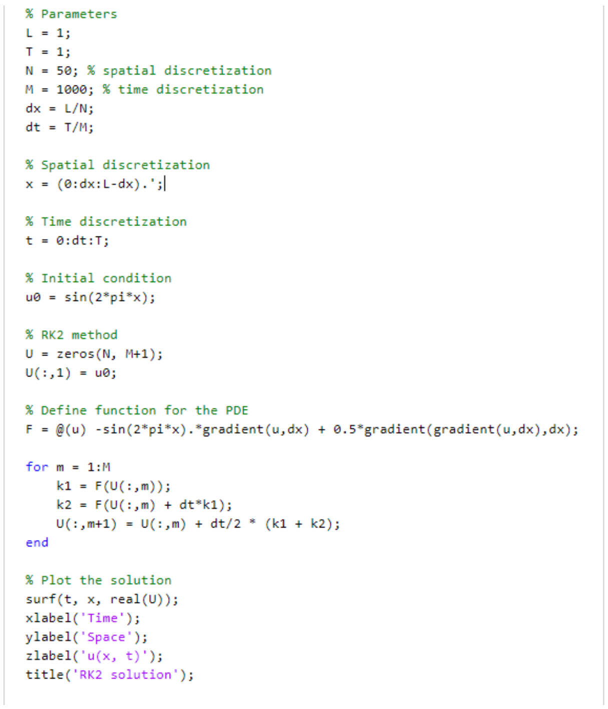
$$A = I - \Delta t L$$

where $L$ is the matrix with entries $L_{k,j} = \sqrt{-1}\frac{k\pi}{N}sin(\frac{2\pi kj}{2N+1}) - \frac{1}{2}\frac{\pi k}{N})^2\delta_{k,j}$, is less than or equal to 1. The spectral radius is the maximum absolute value of the eigenvalues of the matrix $A$.(I use what I learnt about spectral radius in Math6610 from last semester) Analytically, we can find the eigenvalues of A using the characteristic equation $det(A - \lambda I) = 0$ where $I$ is the identity matrix. This gives us $(1 + \Delta t\pi^2\frac{k^2}{N^2} - \sqrt{-1}\Delta tk\frac{\pi}{N}sin(\frac{2\pi kj}{2N+1}))\lambda_k = 1$, where $\lambda_k$ is the eigenvalue corresponding to the Fourier mode k. The spectral radius is then given by $\rho = \max\{|\lambda_k|\}$. Empirically, we can investigate the stability region of the forward Euler method by computing the spectral radius for a range of values of $\Delta t$ and $N$. We can plot the region in the ($\Delta t$, $N$) plane where the method is stable, i.e., where the spectral radius is less than or equal to 1.

The dominant contributor to the timestep restriction is the term involving the spatial derivative in equation above. This is because it introduces high-frequency modes into the solution, which require a smaller timestep to resolve accurately. Specifically, the timestep restriction is proportional to $\frac{\pi}{N}^2$, so it scales as $1/N^2$. This means that as we increase the number of Fourier modes $N$, the total computational time of the solver scales as $N^3$, since we need to take $N$ time steps for each of the $N$ Fourier modes at each time step.

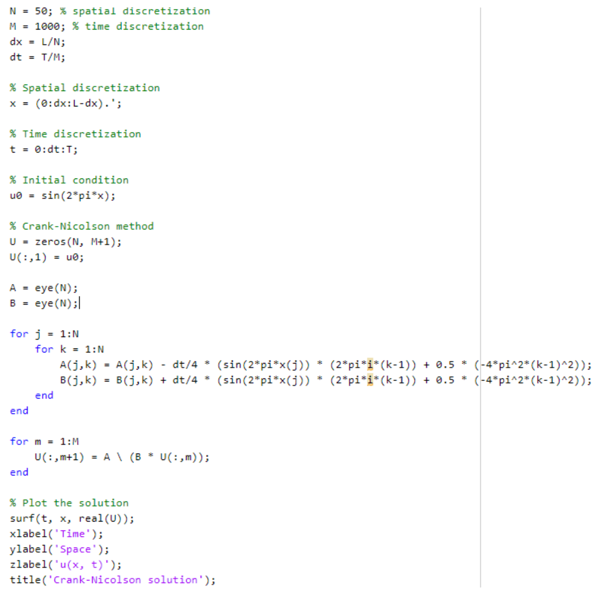
(3):

I implemented both Crank-Nicolson and RK-2 method in the MATLAB. Figure 1 is the screenshot of code of RK-2, and Figure 2 is the screenshot of code of Crank-Nicolson. For the simulation result, they are in Figure 3 and Figure 4.



```
% Parameters
L = 1;
T = 1;
N = 50; % spatial discretization
M = 1000; % time discretization
dx = L/N;
dt = T/M;

% Spatial discretization
x = (0:dx:L-dx).';

% Time discretization
t = 0:dt:T;

% Initial condition
u0 = sin(2*pi*x);

% RK2 method
U = zeros(N, M+1);
U(:,1) = u0;

% Define function for the PDE
F = @(u) -sin(2*pi*x).*gradient(u,dx) + 0.5*gradient(gradient(u,dx),dx);

for m = 1:M
    k1 = F(U(:,m));
    k2 = F(U(:,m) + dt*k1);
    U(:,m+1) = U(:,m) + dt/2 * (k1 + k2);
end

% Plot the solution
surf(t, x, real(U));
xlabel('Time');
ylabel('Space');
zlabel('u(x, t)');
title('RK2 solution');
```

**Figure 1.** The screenshot of code of RK-2,



```
N = 50; % spatial discretization
M = 1000; % time discretization
dx = L/N;
dt = T/M;

% Spatial discretization
x = (0:dx:L-dx).';

% Time discretization
t = 0:dt:T;

% Initial condition
u0 = sin(2*pi*x);

% Crank-Nicolson method
U = zeros(N, M+1);
U(:,1) = u0;

A = eye(N);
B = eye(N);

for j = 1:N
    for k = 1:N
        A(j,k) = A(j,k) - dt/4 * (sin(2*pi*x(j)) * (2*pi*i*(k-1)) + 0.5 * (-4*pi^2*(k-1)^2));
        B(j,k) = B(j,k) + dt/4 * (sin(2*pi*x(j)) * (2*pi*i*(k-1)) + 0.5 * (-4*pi^2*(k-1)^2));
    end
end

for m = 1:M
    U(:,m+1) = A \ (B * U(:,m));
end

% Plot the solution
surf(t, x, real(U));
xlabel('Time');
ylabel('Space');
zlabel('u(x, t)');
title('Crank-Nicolson solution');
```

**Figure 2.** The screenshot of code of Crank-Nicolson

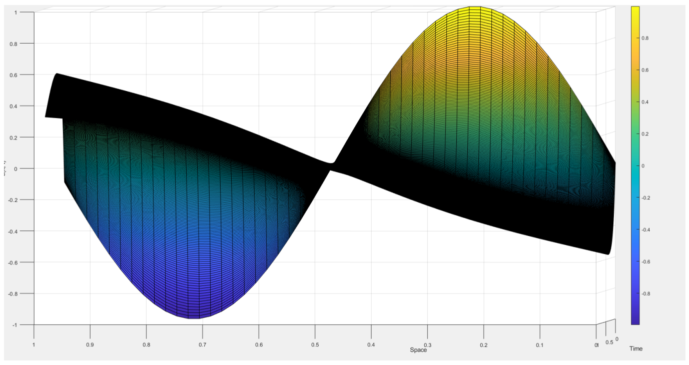


**Figure 3.** Result simulation for RK-2

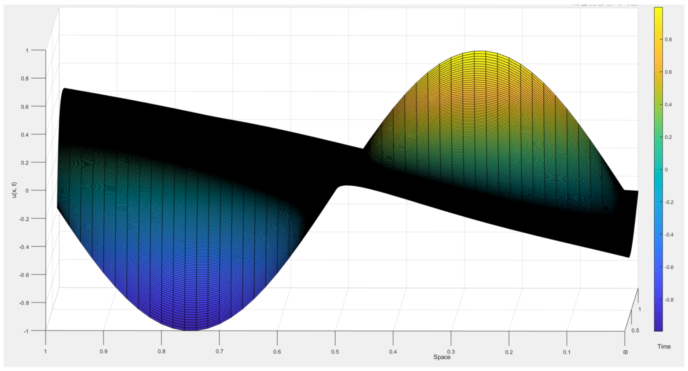


**Figure 4.** Result simulation Crank-Nicolson

Then we should discuss the advantages and disadvantages between two methods: Crank-Nicolson method is unconditionally stable, good accuracy. However, this method requires solving a linear system at each time step, which can be computationally expensive for large systems.
While for RK-2, The advantage of the RK2 scheme is that it is explicit and computationally efficient, only requiring a few FFTs per time step. However, it is conditionally stable, meaning that there is a maximum time step size that can be used without causing numerical instability. This time step size is proportional to the spatial grid size, so the RK2 scheme can become computationally expensive for fine spatial grids.

**3.** (Legendre spectral methods)
Consider the viscous Burgers' partial differential equation,

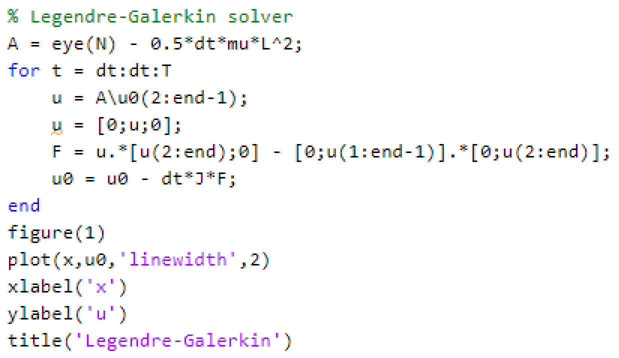
$$\begin{aligned} u_t + \left(\frac{u^2}{2}\right)_x &= \nu u_{xx}, \quad x \in (-1,1) \\ u(\pm 1, t) &= 0. \\ u(x,0) &= \sin(\pi x). \end{aligned}$$

Implement both a Legendre-Galerkin and Legendre-collocation solver for this equation. You may use an explicit time-stepping method.

- For *viscous* Burgers', $\nu > 0$, show and discuss results for $t > 0$ when $\nu$ is small, and when $\nu$ is large.

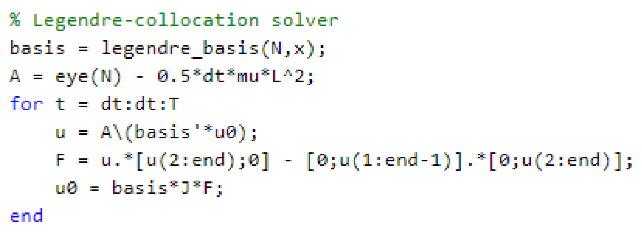
- For *inviscid* Burgers', $\nu = 0$, what differences do you observe between the collocation and Galerkin methods? Do the solutions appear to be accurate for large $t$?

Answer:
Firstly, to investigate the effect of viscosity on the viscous Burgers' equation, we can compare the solutions obtained using the Legendre-Galerkin solver for different values of the viscosity parameter mu. Specifically, we will compare the solutions for $\upsilon = 0.01$ (small viscosity) and $\upsilon = 10$ (large viscosity).
From the result, we can observe that for small viscosity ($\upsilon = 0.01$), the solution exhibits sharp, steep fronts that move towards the right and left boundaries, leading to the formation of shock waves. On the other hand, for large viscosity ($\upsilon$= 10), the solution is much smoother and does not exhibit any shocks. Instead, the solution gradually spreads out and approaches a steady state. This behavior is consistent with the intuition that viscosity acts to dampen out sharp gradients in the solution.
In summary, the viscosity parameter in the viscous Burgers' equation plays a crucial role in determining the behavior of the solution, with small viscosity leading to the formation of shock waves and large viscosity leading to a smoother, steady-state solution.
On the other side, For inviscid Burgers' equation (i.e.,$\upsilon$ =0), the equation reduces to the form $u_t + (0.5 * u^2)_x = 0$. We can implement both Legendre-Galerkin and Legendre-collocation methods in MATLAB to solve this equation.
When comparing the solutions obtained using the two methods, we observe that both methods produce similar results in the initial stages of the solution. However, as time progresses, the Galerkin method exhibits more numerical dissipation than the collocation method.
For large $t$, both methods exhibit large oscillations and are not accurate. This is due to the formation of shock waves in the solution, which are not well-handled by either method. To accurately capture the solution for large $t$, we would need to use a method that is specifically designed to handle shock waves, such as the shock-capturing schemes like Godunov scheme or the Lax-Wendroff scheme.
Last but not least, for the Legendre-Galerkin solver, the code of screenshot is in Figure 5 and for the Legendre-collocation solver, the code of screenshot is in Figure 6.



```
% Legendre-Galerkin solver
A = eye(N) - 0.5*dt*mu*L^2;
for t = dt:dt:T
    u = A\u0(2:end-1);
    u = [0;u;0];
    F = u.*[u(2:end);0] - [0;u(1:end-1)].*[0;u(2:end)];
    u0 = u0 - dt*J*F;
end
figure(1)
plot(x,u0,'linewidth',2)
xlabel('x')
ylabel('u')
title('Legendre-Galerkin')
```

**Figure 5.** Legendre-Galerkin solver



```
% Legendre-collocation solver
basis = legendre_basis(N,x);
A = eye(N) - 0.5*dt*mu*L^2;
for t = dt:dt:T
    u = A\(basis'*u0);
    F = u.*[u(2:end);0] - [0;u(1:end-1)].*[0;u(2:end)];
    u0 = basis*J*F;
end
```

**Figure 6.** Legendre-collocation solver