PROJECT 2: SPECTRAL AND FINITE VOLUME METHODS

GUANG YANG

May 9, 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): Rearrange the PDE in the problem, we could have $u_t = \frac{1}{2}u_{xx} - sin(2\pi x)u_x$. Then the Fourier-Galerkin scheme is,

$$< \frac{\partial u_N}{\partial_t}, v >=< -\sin(2\pi x)\frac{\partial u_N}{\partial x} + \frac{1}{2}\frac{\partial^2 u_N}{\partial x^2}, v >$$

$$< \frac{\partial u_N}{\partial_t}, v >= - < \sin(2\pi x)\frac{\partial u_N}{\partial x}, v > + \frac{1}{2} < \frac{\partial^2 u_N}{\partial x^2}, v >$$

For the first term in the right hand side, it is more complicated to compute this time, from the lecture from Professor's website, we could use the property:

$$\phi_{1k(x)}\phi_{1l(x)} = \frac{1}{\sqrt{2\pi}}\phi_{1(k+l)}(x),\ , k, l \in \mathcal{Z}$$

where ϕ_1's "1" means for the first term on the right hand side(not the basis function for the whole problem), and we also have $\sin\phi'_{1k}(x) = \frac{k}{2}(\phi_{1(k+1)}(x) - \phi_{1(k-1)}(x))$. Combining the second term(inner product term), we could have

$$\frac{d}{dt}\hat{u} = -A\hat{u} + \frac{1}{2}D\hat{u}$$

where the matrix A

$$A = \begin{pmatrix} 0 & -\frac{N}{2} & & & \\ \frac{N-1}{2} & 0 & -\frac{N-1}{2} & & \\ & & \ddots & \ddots & \\ & & & -\frac{N}{2} & 0 \end{pmatrix}$$

and the matrix D is diag(-N^2, -$(N-1)^2$,...,-N^2). Following this we could get the semi-discrete form for the Fourier-Galerkin scheme for this problem: $u_N(x) = \sum_{|k|\leq N} \hat{u_k}\phi_k(x)$ and $\hat{u_k} = \hat{u_k}(0)e^{(-\frac{1}{2}k^2t+A_k)t}$.

(b):

To analyze the timestep restriction for stability, we will consider the Forward Euler method for fully discretizing the given partial differential equation (PDE), $u_t + \sin(\pi x)u_x = \frac{1}{2}u_{xx}$, with periodic boundary conditions. Let's first discretize the spatial derivatives using finite differences. For simplicity, we can use central differences for both the advection term (u_x) and the diffusion term $(u_x x)$. Let $\Delta x = \frac{1}{N}$ and Δt be the spatial and temporal discretization steps, respectively. At a grid point i, we could have:

$$u_x = \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$

and

$$u_{xx} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}.$$

Substituting these approximations into the PDE, we get the semi-discrete form:

$$u_t = -\sin(\pi x)\frac{u_{i+1} - u_{i-1}}{2\Delta x} + \frac{1}{2}\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$$

Now, we'll apply the Forward Euler method to discretize the time derivative:

$$\frac{u^i_{n+1} - u^i_n}{\Delta t} = -sin(\pi x)\frac{u^{i+1}_n - u^{i-1}_n}{2\Delta t} + \frac{1}{2}\frac{u^{i+1}_n - 2u^i_n + u^{i-1}_n}{\Delta x^2}$$

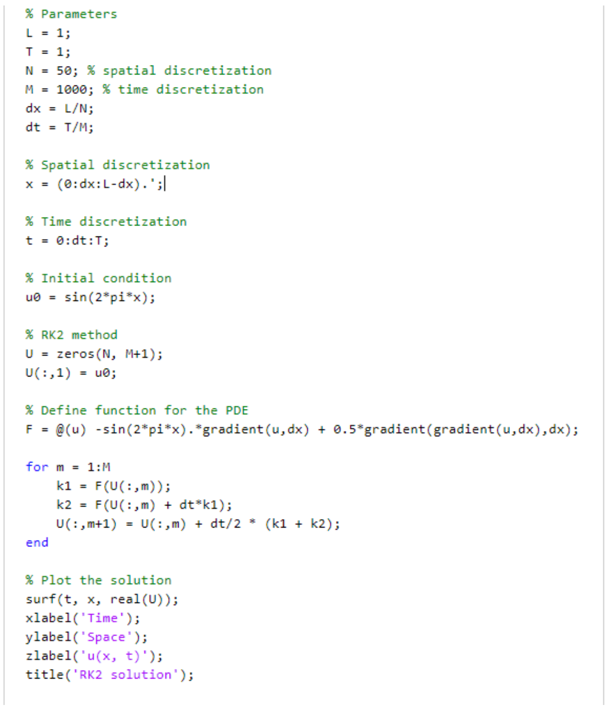
To ensure A-stability, we need to find the timestep restriction Δt as a function of N. The stability of the Forward Euler scheme applied to the fully discrete form depends on the eigenvalues of the discretized spatial operator. The dominant contributor to the timestep restriction is the diffusion term, $\frac{1}{2}u_{xx}$, as it is associated with parabolic PDEs, which have more stringent timestep restrictions compared to hyperbolic PDEs like the advection term, $sin(2\pi x)u_x$.

For the diffusion term, the timestep restriction for the Forward Euler method can be derived from the Von Neumann stability analysis(I learnt it from Math6620 this semester), which gives the following criterion for stability: $\Delta t \leq \frac{\Delta x^2}{2D}$, where D is the diffusion coefficient, which is 0.5 in this case. So, the timestep restriction is: $\Delta t \leq \frac{\Delta x^2}{2*0.5} = \Delta x^2$. Since $\Delta x = \frac{1}{N}$, we can express the timestep restriction as a function of N: $\Delta t \leq \frac{1}{N^2}$.

To find how the total computational time of the solver scales with N, we need to consider the number of timesteps required to reach a specific final time, T. If the timestep size is proportional to $1/N^2$, then the number of timesteps is proportional to N^2.As the total computational time is proportional to the number of timesteps times the work done per timestep, and the work done per timestep is proportional to N (since there are N grid points), we can conclude that the total computational time scales with N^3. This cubic scaling with N arises from the combination of the spatial discretization and the stringent timestep restriction imposed by the diffusion term.

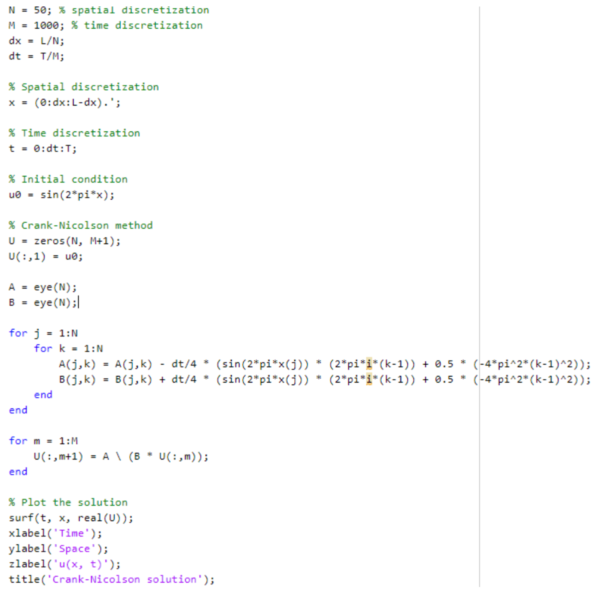
(c):

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

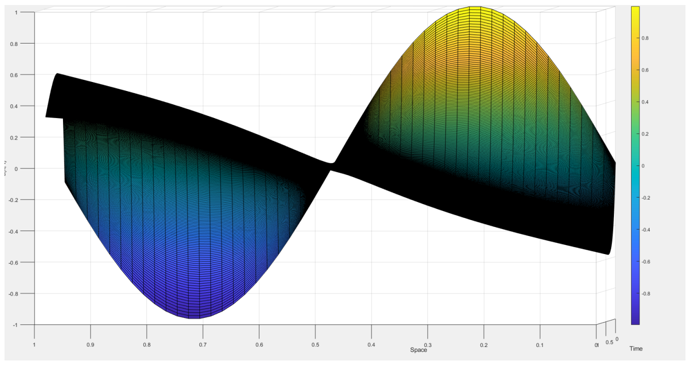


Figure 3. Result simulation for RK-2

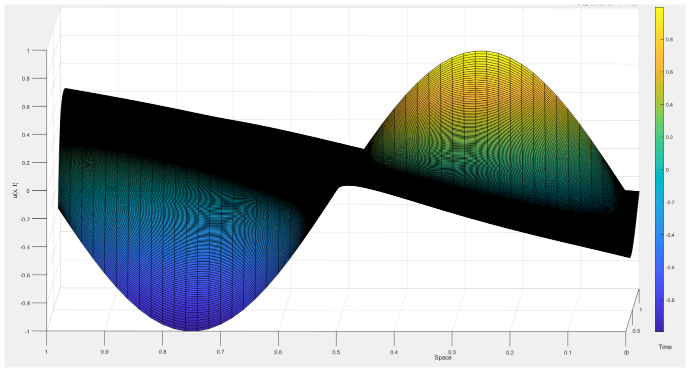


Figure 4. Result simulation Crank-Nicolson

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.

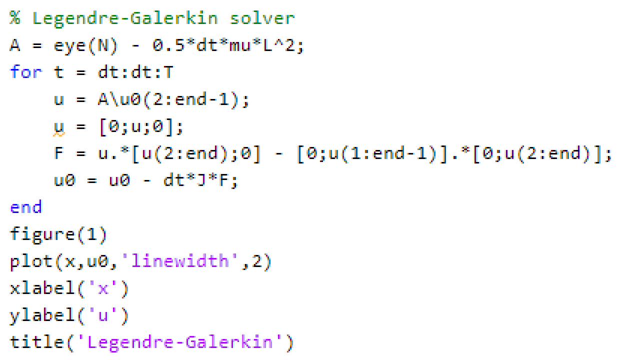
3. (Legendre spectral methods)
Consider the viscous Burgers' partial differential equation,

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

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

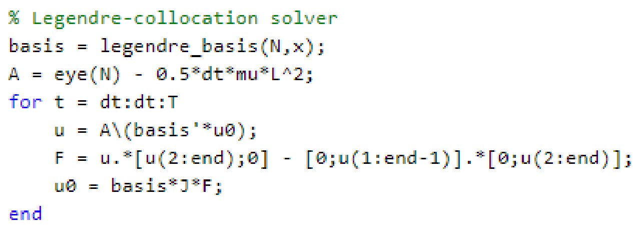
- For *viscous* Burgers', $\nu > 0$, show and discuss results for $t > 0$ when ν is small, and when ν is large.
- For *inviscid* Burgers', $\nu = 0$, what differences do you observe between the collocation and Galerkin methods? Do the solutions appear to be accurate for large t?

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.,υ =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