Detailed Explanation of the Code for Solving the PDE

The provided code solves the following partial differential equation (PDE):

$$u_{xx} - a(x)u = f,$$

where $u(x)$ is the unknown function, $a(x)$ is a known coefficient, and f is a source term. The solution is obtained using the Newton-GMRES method in a large-scale periodic domain with GPU acceleration. This section provides an in-depth breakdown of each step in the code and the mathematical reasoning behind them.

Step 1: Setting up the Problem

We begin by setting up the spatial grid and defining the parameters:

- `par.n` = 8192: This is the number of discretization points, chosen to provide a fine resolution for the solution. A higher value improves accuracy but requires more computational resources.
- $M = 50$: This is a scaling factor for the domain size. The physical domain extends from $-L$ to L, where $L = M \times \pi$, i.e., $L = 50\pi$, giving a large domain for the problem.
- `par.x`: The spatial grid x is generated using MATLAB's `linspace`, spanning the interval $[-L, L]$, and then adjusted to ensure periodic boundary conditions. The grid consists of n points, each separated by a distance `par.dx`.
- `par.dx`: The spatial step size is given by `par.dx` $= \frac{2L}{n}$, which determines the resolution of the discretization.

These parameters set up the numerical discretization for solving the PDE on a periodic domain.

Step 2: Defining the Known Solution and Coefficients

- $u(x) = \cos(x)$: This is the known solution $u(x)$ of the PDE. In this case, we are assuming a cosine function as the true solution. This choice allows us to evaluate the accuracy of the computed solution.
- $a(x) = 1 + 0.5\cos(x)$: This is a known coefficient function $a(x)$, which is part of the PDE. The choice of this form ensures that the problem is non-trivial and requires the solution to adapt to the varying coefficient.

Both $u(x)$ and $a(x)$ are moved to the GPU using `gpuArray` to enable fast computation on the GPU.

Step 3: Fourier Space Setup

To solve the PDE efficiently, we move to Fourier space. The second derivative u_{xx} is simpler to compute in Fourier space because it corresponds to multiplication by $-k^2$, where k is the wave number.

- k: The wave numbers k are defined as the discrete Fourier modes scaled by $\frac{1}{M}$ (to normalize over the domain). The wave numbers range from 0 to $\frac{n}{2}$ and then from $-\frac{n}{2}+1$ to -1, ensuring a full Fourier representation.
- k^2: The square of the wave numbers, used to compute the second derivative u_{xx} in Fourier space. In the Fourier transform, the second derivative of a function corresponds to multiplying its Fourier transform by $-k^2$, which simplifies the computation of u_{xx}.

Step 4: Defining the Residual Function $R(f)$

The key to solving the PDE is defining the residual function. The equation we wish to solve is:

$$u_{xx} - a(x)u = f.$$

Using Fourier methods, the second derivative of u in Fourier space is $-k^2\hat{u}(k)$, where $\hat{u}(k)$ is the Fourier transform of $u(x)$. Rearranging the PDE, we obtain:

$$f = u_{xx} - a(x)u = \mathcal{F}^{-1}\left(-k^2\mathcal{F}(u)\right) + a(x)u.$$

Thus, the residual function $R(f)$ is defined as:

$$R(f) = f - \Re\left(\mathcal{F}^{-1}\left(-k^2\mathcal{F}(u)\right)\right) + a(x)u(x),$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transform and its inverse, respectively. The term $\Re$ represents the real part because the inverse Fourier transform returns a complex value, but we are only interested in the real part of the solution.

This residual is minimized during the Newton iterations to iteratively improve the solution for f.

Step 5: Preconditioning

The preconditioner is used to improve the convergence of the GMRES solver. In this case, the preconditioner is a simple one that scales the Fourier transform of the update *df* by $\frac{1}{1+|k^2|}$, which is applied to the update *df* in Fourier space:

$$\texttt{pc}(df) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(df)}{1+|k^2|}\right).$$

This preconditioning step helps to smooth out the solution and improve the convergence rate of the GMRES solver by damping high-frequency components that could otherwise cause slow convergence.

Step 6: Newton-GMRES Iterations

The Newton method is used to iteratively solve for f by minimizing the residual function. The basic idea is to iteratively update the guess for f by solving a linearized system around the current guess.

In each Newton iteration:

- The residual $R(f)$ is computed, which measures how far the current guess is from satisfying the PDE.
- The norm of the residual is computed and compared to a tolerance `tol`. If the residual norm is sufficiently small, the algorithm stops, and the solution is considered converged.
- If the residual is not small enough, GMRES is used to solve for the update df that minimizes the residual. GMRES is a Krylov subspace method that solves the linear system efficiently, especially for large problems. The preconditioner `pc` is applied during the GMRES solve to accelerate convergence.
- The solution f is updated by adding the computed update df.

The Newton loop continues until the residual is below the specified tolerance or the maximum number of iterations is reached.

Step 7: Result and Visualization

Once the solution $f(x)$ converges, the result is transferred back from the GPU to the CPU using `gather(f)`. The solution is then plotted using the MATLAB `plot` function to visualize the computed solution $f(x)$ on the spatial grid x.

- The x-axis represents the spatial domain x, and the y-axis represents the computed function $f(x)$.
- The plot is titled "Computed $f(x)$ for Large-Scale Problem (GPU Accelerated)".

Summary

This code solves the PDE $u_{xx} - a(x)u = f$ in a large periodic domain using GPU acceleration and the Newton-GMRES method. The solution involves discretizing the problem in space, transforming it to Fourier space for efficient computation of the second derivative, and solving the resulting nonlinear equation iteratively using GMRES. The use of GPU acceleration speeds up the computation significantly for large-scale problems, and the preconditioner improves the convergence of the iterative solver.