Solving 1-D Wave Equation Using C++ and Open MPI

Sayem Khan

Tuesday, December 17, 2019

1 Introduction

The wave equation has very significant role in physical science and engineering. Throughout the project, 1-D wave equation is studied and the numerical solution is implemented by the C++ and Open MPI.

2 Problem Setup: 1-D Wave Equation

If $u(x,t)$ is the displacement and c is the propagation constant of the medium, then the 1-D wave equation:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} \tag{1}$$

over the spatial interval $[0, L = 1]$ and time interval $[0, t = T]$ with initial conditions:

$$u(x = x, t = 0) = \sin\left[2\pi(x - ct)\right] = \sin(2\pi x) \tag{2}$$

$$u'(x = x, t = 0) = -2\pi c \cos\left(2\pi x\right) \tag{3}$$

and the boundary conditions:

$$u(x = 0, t) = u0(t) = \sin\left[2\pi(0 - ct)\right] = sin(-2\pi ct) \tag{4}$$

$$u(x = 1, t) = u1(t) = sin[2\pi(1 - ct)] \tag{5}$$

3 Numerical Treatment

3.1 Discretizing the domain

Mesh in time:

$$0 = t_0 < t_1 < t_2 < \cdots < t_{N_t - 1} < t_{N_t} = T \tag{6}$$

Mesh in space:

$$0 = x_0 < x_1 < x_2 < \cdots < x_{N_x - 1} < t_{N_x} = L \tag{7}$$

Uniform mesh with constant mesh spacing Δt and Δx:

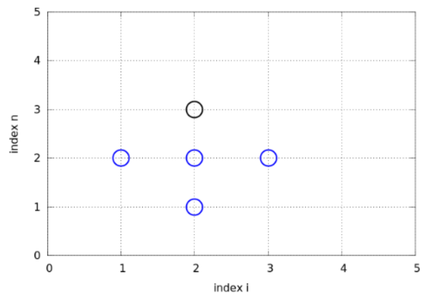

Figure 1: Stencil at interior point

$$x_i = i\Delta x,\ i = 0, ..., N_x, \qquad t_i = n\Delta t, n = 0, ..., N_t \tag{8}$$

- The numerical is a mesh function: $u_n^i \approx u(x_i, t_n)$.
- Finite difference stencil (or scheme): equation for u_i^n involving neighboring space-time points.

3.2 Fulfilling the equation at the mesh points

Let the PDE be satisfied at all interior mesh points:

$$\frac{\partial^2 u(x_i, t_n)}{\partial t^2} = c^2 \frac{\partial^2 u(x_i, t_n)}{\partial x^2} \tag{9}$$

for $i = 1, \cdots, N_x - 1$ and $n = 1, \cdots, N_t - 1$. For $n = 0$ we have the initial conditions stated in Equation 2 & Equation 3 and $i = 0$ boundary condition stated in Equation 4 & Equation 5.

3.3 Replacing derivatives by finite differences

$$\frac{\partial^2}{\partial t^2} u(x_i, t_n) \approx \frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2} = [D_t D_t u]_i^n \tag{10}$$

$$\frac{\partial^2}{\partial x^2} u(x_i, t_n) \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} = [D_x D_x u]_i^n \tag{11}$$

3.4 Algebraic version of the PDE

In Equation 1, replace derivatives by differences:

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2} = c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \tag{12}$$

In operator notation:

$$\left[D_t D_t u = c^2 D_x D_x\right]_i^n \tag{13}$$

3.5 Formulation a recursive algorithm

- Nature of the algorithm: compute u in space at $t = \Delta t, 2\Delta t, 3\Delta t, \cdots$
- Three time levels are involved in the general discrete equation: $n-1, n, n+1$
- u_i^n and u_i^{n-1} are then already computed for $i = 0, 1, , N_x$ and u_i^{n+1} is the unknown quantity.

Write out $\left[D_t D_t u = c^2 D_x D_x\right]_i^n$ and solve for u_i^{n+1},

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + \alpha^2 (u_{i+1}^n - 2u_i^n + u_{i-1}^n) \tag{14}$$

Here,

$$\alpha = c\frac{\Delta t}{\Delta x}$$

is known as the Courant Number. α must satisfy the Courant–Friedrichs–Lewy (CFL) stability condition:

$$\alpha \leq 1$$

4 Organization of the program

Our goal is to use MPI in order to accelerate this computation. The method of domain decomposition is used - that is, we assume we have P MPI processes, we divide the original interval into P sub-intervals, and we expect each process to update the data associated with its sub-interval.

However, to compute the estimated solution $U(x, t+dt)$ at the next time step requires information about $U(x-dx, t)$ and $U(x+dx, t)$. When process ID tries to make these estimates, it will need one value from process $(ID-1)$, and one from process $(ID+1)$, before it can make all the updates. MPI allows the processes to communicate this information using messages.

A solution table is printed at the end of the final time. All process sends its value to master (ID = 0) process. The master process will collect all results and print in organized fashion.

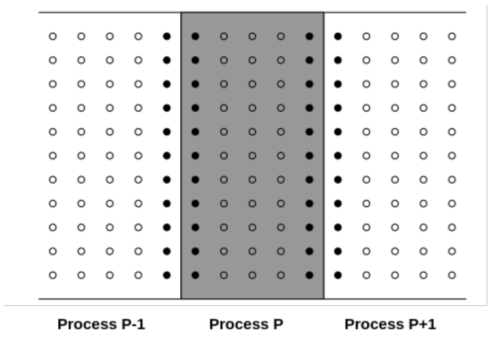

Figure 2: Exchange between the process

5 Program

```cpp
#include "update.hpp"
#include "collect.hpp"
#include "exact.hpp"
#include "dudt.hpp"

using namespace std;
/*******************************************************************************
 * The wave equation in 1-D :
 *     d^2 u/dt^2 - c^2 * d^2 u/dx^2 = 0  for 0 < x < 1, 0 < t
 * Initial Conditions:
 *     u(x=x,t=0) = g(x,t=0) = sin[2*pi*(x-c*t)] = sin(2*pi*x)
 *    u'(x=x,t=0) = h(x,t=0) = - 2*pi*c*cos[2*pi*x]
 * Boundary Condition:
 *      u(x=0,t) = u0(t) = sin(2*pi*(0-c*t)) = sin(-2*pi*c*t)
 *      u(x=1,t) = u1(t) = sin(2*pi*(1-c*t))
 * Let, the alpha = c*du/dt, for stablity: alpha <= c*dt/dx
 * Using the Finite Difference Method, we can write:
 *    U(x,t+dt) = 2 U(x,t) - U(x,t-dt)
 *                + alpha^2[U(x-dx,t)-2U(x,t)+U(x+dx,t)]
 *
 *
 * Role of the functions:
 *   1. update.hpp : for each time step calculate the u(x,t) and update the local
    array
 *                  also check the stability condition.
 *   2. collect.hpp: collect all data from the all local arrays
 *   3. exact.hpp  : calculate the exact solution and IC and BC
 *   4. dudt.hpp   : calculate the derivative of the function u(x,t)
 *
    ***************************************************************************************
    */

int main(int argc, char *argv[])
{
  double dt = 0.00125; //time step
  int i_global_hi;
  int i_global_lo;
  int id;              //the MPI process ID
  int n_global = 901; //the total number of points
  int n_local;         //number of total point per process
  int nsteps = 4000;  //the number of time steps
  int p;               //the number of MPI processes
  double *u1_local;
  double propagation_constant = 0.8;

  /******Initialize MPI*******/

  MPI_Init(&argc, &argv);

  MPI_Comm_rank(MPI_COMM_WORLD, &id);

  MPI_Comm_size(MPI_COMM_WORLD, &p);

  /*********Determine N_LOCAL***********
   *        Domain decomposition        *
   * **********************************/
  i_global_lo = (id * (n_global - 1)) / p;
  i_global_hi = ((id + 1) * (n_global - 1)) / p;
  if (0 < id)
  {
    i_global_lo = i_global_lo - 1;
  }

  n_local = i_global_hi + 1 - i_global_lo;
  //calculation
  u1_local = update(id, p, n_global, n_local, nsteps, dt, propagation_constant);
  //collect from all process
  collect(id, p, n_global, n_local, nsteps, dt, u1_local);

  MPI_Finalize();

  delete[] u1_local;

  return 0;
}
```

Listing 1: **wave.cpp**: the main program for calculation

```
#!/bin/bash

make

file_1="wave.dat"

if [ -f $file_1 ] ; then
    rm $file_1
fi
c=0.15
numProcess=4
mpirun -np $numProcess wave $c >> $file_1
python plot.py
```

Listing 2: **driver.sh**: driver program.

6 Results

The program has been run for different propagation constants:

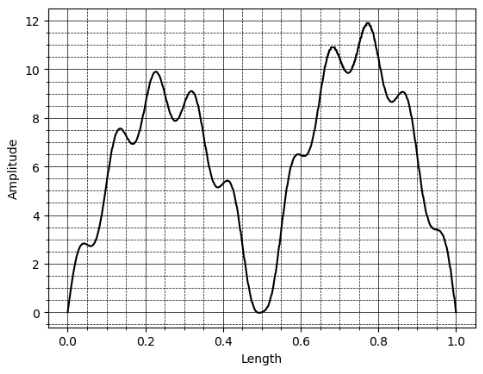

Figure 3: Result of the simulation with propagation constant $c = 0.1$

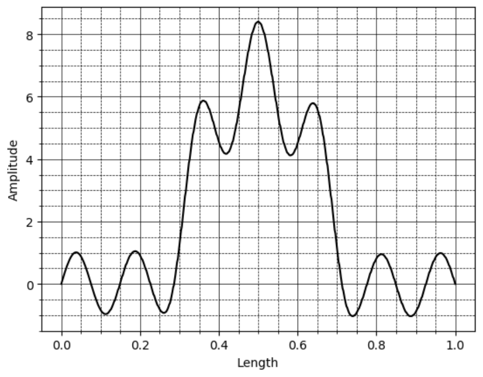

Figure 4: Result of the simulation with propagation constant $c = 0.15$

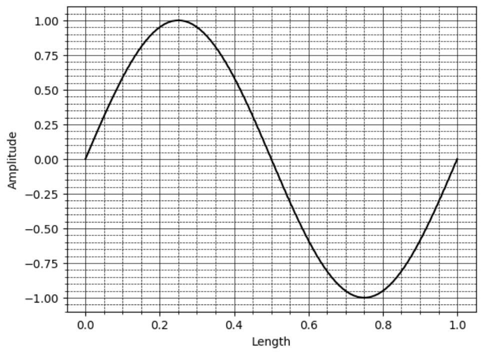

Figure 5: Result of the simulation with propagation constant $c = 0.15$

7 Limitation of the program and Future scope

Due to CFL stability condition, the program can't be run using arbitrary discreet point of x and t. All through the program time-step, $dt = 0.00125$ s is used and there is scope for user defined Initial and Boundary conditions. For these reason the program has not any universality. In future, this problem could be studied and solved. Also, it could be extended for 3D wave equation, which is much more relevant for practical problem.

Resource and study materials used for the project

a) Joe Pitt-Francis, Jonathan Whiteley - Guide to Scientific Computing in C++, Second Edition, Springer, 2012

b) Hans Petter Langtangen, Svein Linge - Finite Difference Computing with PDEs: A Modern Software Approach, Springer, 2017

c) Borut Robič, Patricio Bulič, and Roman Trobeč - Introduction to Parallel Computing From Algorithms to Programming on State-of-the-Art Platforms, Springer, 2018

d) Parallel Algorithms for Solving Partial Differential Equations, Stracy Pschenica, Department of Aerospace Engineering, Iowa State University, The Journal of the Iowa Academy of Science: JIAS, Volume 101, 1994