Math 4997-3

Lecture 12: One-dimensional heat equation

https://www.cct.lsu.edu/~pdiehl/teaching/2019/4977/

Reminder

Lecture 12

What you should know from last lecture

- What is HPX
- Asynchronous programming using HPX
- Shared memory parallelism using HPX

Heat equation

Heat equation

Statement of the heat equation

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)$$

where alpha is the diffusivity of the material.

Compact form

$$\dot{u} = \alpha \nabla u$$

The heat equation computes the flow of heat in a homogeneous and isotropic medium.

More details [1].

Easiest case

1D heat equation

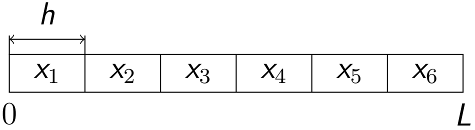
$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \quad 0 \leq x \leq L, t > 0$$

Boundary conditions

The solution of the heat equation requires boundary conditions

- $u(0, t) = u_0$
- $u(L, t) = u_L$
- $u(x, 0) = f_0(x)$

Discretization



Discrete mesh

$$x_i = (i-1)h, \quad i = 1, 2, \ldots, N$$

where N is the total number of nodes and h is given by $h = {}^{L}/_{N-1}$.

Finite difference method

Approximation of the first derivative

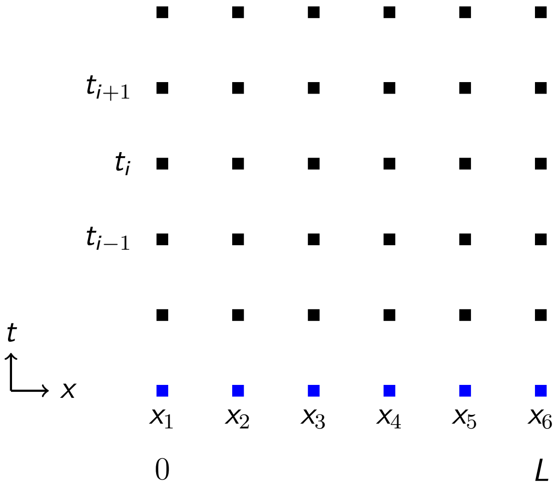
$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1} - u_i}{2h}$$

Approximation of the second derivative

$$\frac{\partial u}{\partial x^2} \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{2h}$$

Note that a central difference scheme is applied. More details [3, 2].

Discretization in space and time



Serial implementation

Time measurement and system information

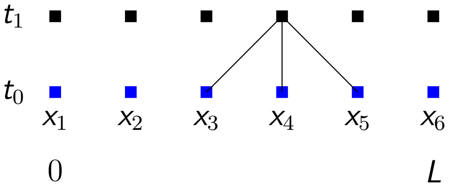
Time measurement

```
std::uint64_t t
        = hpx::util::high_resolution_clock::now();
// Do work
std::uint64_t elapsed
        = hpx::util::high_resolution_clock::now() - t;
```

Accessing system information

```
std::uint64_t const os_thread_count
        = hpx::get_os_thread_count();


std::cout << "Computation took " << elapsed
        << " on " << os_thread_count << " threads"
        << std::endl;
```

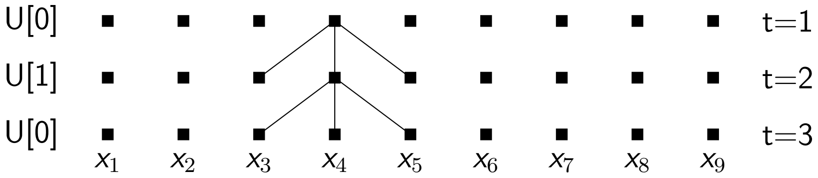
Discretization scheme



Approximation of the heat equation

```
static double heat(double left,
                   double middle,
                   double right)
{
    return middle +
           (k*dt/(dx*dx)) * (left - 2*middle + right);
}
```

Swapping the data



Swapping function

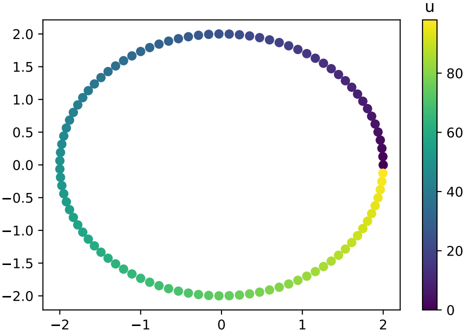
```
space do_work(std::size_t nx, std::size_t nt)
{
    // U[t][i] is the state of position i at time t.
    std::vector<space> U(2);
    for (space& s : U)
        s.resize(nx);

    // Return the solution at time-step 'nt'.
    return U[nt % 2];
}
```

Do the actual work

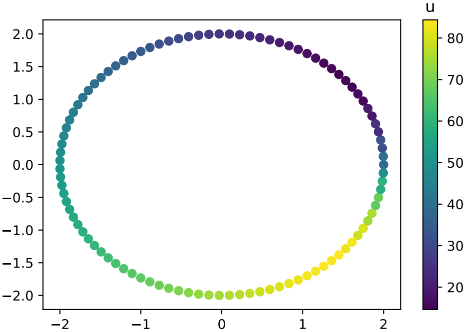
```cpp
// Actual time step loop
for (std::size_t t = 0; t != nt; ++t)
{
    space const& current = U[t % 2];
    space& next = U[(t + 1) % 2];

    next[0] =
        heat(current[nx-1], current[0], current[1]);

    for (std::size_t i = 1; i != nx-1; ++i)
        next[i] =
         heat(current[i-1], current[i], current[i+1]);

    next[nx-1] =
        heat(current[nx-2], current[nx-1], current[0]);
}
```

Initial conditions

$$u(x, 0) = f(i, 0), \text{ with } f(0, i) = i \text{ for } i = 1, 2, \ldots, N$$



Solution



Parameters

- heat transfer coefficient $k = 0.5$
- time step size $dt = 1.;$
- grid spacing $h = 1.;$
- time steps $nt = 45;$

Summary

Summary

After this lecture, you should know

- One-dimensional heat equation
- Serial implementation

References

References I

John Rozier Cannon.
The one-dimensional heat equation.
Number 23. Cambridge University Press, 1984.

Randall J LeVeque.
Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems, volume 98.
Siam, 2007.

John C Strikwerda.
Finite difference schemes and partial differential equations, volume 88.
Siam, 2004.