LECTURE 5

Algorithms. Optimization and stability.

- Analysis of the quickest algorithm for Toronto Census
- Why (sometimes) we see deviations from expected scaling law of error of integration scheme with the number of points (catching up with lecture 4)
- PDEs discretization limitations (timestep in heat diffusion)
- non-negativity constraint
- von Neumann analysis of algorithms for PDEs

Tutorial 4

- Assignment 2: detailed discussion of solutions
- Learn C by programming Kruskal sequences

Literature: **see links on our course home page, the references page, and the coding page available from the course page.**

Census algorithm: problem description

N is the constant number of ~~processors~~ people involved, and we try to engage them as concurrently as possible. The goal of the census is deriving averages of some data every inhabitant possesses.

Two time constants are given: $\boldsymbol{\tau}$ and **t.** They stand for single communication time ofincoming data and single computation time using the freshly obtained data. We can start thinking of $\boldsymbol{\tau}$ = 10 s and **t** = 10 s, but later consider other ratios.

Comm and comp are serial operations, a person cannot perform multitasking while talking on a phone to obtain data, or punching numbers on a calculator.

We have earlier (in L3) figured out that concurrency of data processing requires some kind of tree of connections, but the number of branches from each stem (k) is not known. Perhaps k=2 is best, i.e. at each level of the census each person tasked with processing data will combine own data with data coming from k-1 = 1 other person.

The number of people working on the census will go down by a factor of k in every level. Therefore, the number of levels is a ceiling (rounding up to integer) of the $\log_k N$. For example, if there are 10 people, who combine the data pairwise (k=2), more than 3 levels (4 levels) are needed.

To do more quantitative evaluation of various k-schemes, we need to write a formula for total wall clock time of the census.

Since each person needs to receive via voice call k-1 data packages, that will take time (k-1) $\boldsymbol{\tau}$ seconds. Computing will take each person k **t** seconds.

Census algorithm

The total wall-clock time of census is thus

$T = [\log_k N]_{ceiling} [(k\text{-}1)\,\boldsymbol{\tau} + k\,\mathbf{t}\,]$

For k=2,3,4 we have $[\log_k N]_{ceiling}$ = 20,13, 10 levels.

Evaluating the rest of the expression, we saw that the total time is minimized by the binary tree (k=2) if

The ratio of comm:comp = 1. The same happens if that ratio is > 1, i.e. if the communication is slower than computation (per one n bytes of data transmitted, all arithmetic operations that process that amount of data take less time than transmission).

But the result changes when $\boldsymbol{\tau} \gg \mathbf{t}$, the optimum **k = 3** then, although the T(k=2) is very close behind, and other trees are not too bad. We saw this when we tried $\boldsymbol{\tau} = 0.1\,\mathbf{t}$.

Such an example is similar to a long computation time, relative to a faster inter-node communication. This happens when a cluster node is doing calculations about some assigned volume of space, while sharing with neighbors only the surface data of that volume, or some other limited set of data.

Census algorithm: digression

The duration of census: $T(k) = [\log_k N]_{ceiling} [(k-1)\, \boldsymbol{\tau} + k\, \mathbf{t}\,]$

can be semi-analytically minimized. This requires assuming that k is a continuous variable, which allows minimum finding by calculus. Also, the rounding up of log function to integer is omitted. So – some shortcuts and assumptions are made in the hope that a more general, simple result can be written down without computing a table of results for every choice of time scales. Let's see what we got along that path.

Requiring that dT/dk = 0, we found that the best k satisfies a very tough equation

$$\ln k = 1 - (1/k)\ \boldsymbol{\tau} / (\boldsymbol{\tau} + \mathbf{t}).$$

No analytical solutions! Graphical, yes. We can solve it numerically, either by bisection, or iteratively.

We didn't do it, but it turns our after checking two possible iterative expressions that this iteration

$$k^{(next)} = \exp[1 - (1+\mathbf{t}/\boldsymbol{\tau})^{-1}/k^{(previous)}\ ]$$

is non-divergent and gives reasonable values after rounding, if one starts from $k^{(0)} = 2$.

But you should be warned that:

(i) Assumptions done in analytical work can return a different rounded-off optimal integer k than a simple evaluation of T(k), k=2,3,4,… and

(ii) We did not get a nice analytical solution after all. Numerical iteration or bisection is still needed (notice: a binary tree there), so all the hopes did not really pan out.

Conclusion: in this problem, don't do it.

PDEs

- On a computer, PDEs use stencils (see L4 for the intro to 2nd derivative and Laplacian stencil)
- Depending on the stencil, there may be limitations on discretization steps in time and/or space.
- The implicit schemes often avoid these limitations, but take time to solve implicit equation set in every timestep. (Implicit: when an unknown is found both on the left and right sides of equation.)
- Explicit schemes are fast but have timestep and/or spatial discretization limits
- Courant-Friedrichs-Levy is an example for wave-type and hydrodynamics equations
- Diffusion equation has a maximum limit on dt as well
- Sometimes, like in the heat diffusion problem, we can write the numerical evolution equation and guess what the limiting condition is (in that problem, it boils down to requiring that there is no chance for negative values of temperature, of concentration, or light intensity, to appear)

Von Neumann stability analysis of PDE algorithms:

diffusion equation.

PDEs Example u - spreading quantity $u(x,y,z,t)$ (in 3D)

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

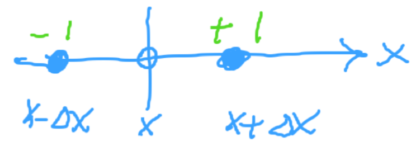
(Δ: Laplacian)

diff. coeff.: D, ν (alt. names)

$[D] = \frac{m^2}{s}$

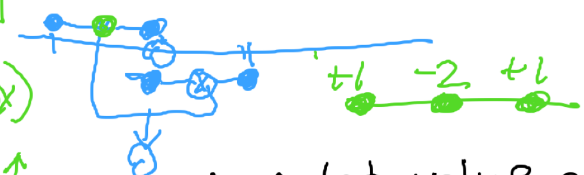
Spatial differentiation

ex. $\frac{\partial u}{\partial x} \approx \frac{u(x+\Delta x) - u(x-\Delta x)}{2\Delta x}$



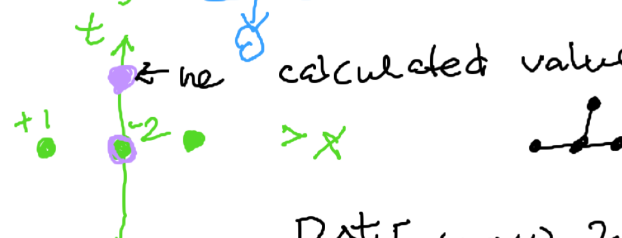
$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{\Delta x}\left[\frac{u(x+\Delta x)-u(x)}{\Delta x} - \frac{u(x)-u(x-\Delta x)}{\Delta x}\right]$$

$$= \frac{1}{(\Delta x)^2}\left(u(x+\Delta x) - 2u(x) + u(x-\Delta x)\right)$$



stencil for heat eq. (1-D)

$$\frac{\partial u}{\partial t} \approx \frac{u(t+\Delta t) - u(t)}{\Delta t}$$



$$u(x,\dots,t+\Delta t) = \frac{D\Delta t}{(\Delta x)^2}\left[u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)\right] + u(x,t)$$

(new) (old) (old)

$$u(x,t+\Delta t) = u(x) + r\left(u(x+\Delta x) - 2u(x) + u(x-\Delta x)\right)$$ where $r := \frac{D\Delta t}{(\Delta x)^2}$

If parameter r is too large, the algorithm is violently unstable:
It grows (out of noise) spurious numerical oscillations with the shortest wavelength representable on a grid. They are known as even-odd decoupling. To study the error propagation on a grid, we study the behavior of harmonic (sinusoidal) waves. This yields a condition parameter r must obey.

Von Neumann stability analysis of PDE algorithms:

diffusion equation (done below),

wave equation (exercise for you)

You should apply the von Neumann stability analysis to the wave equation, and derive the so-called CFL, or Courant-Fredrichs –Levi criterion, limiting either dx or dt, if the other value is known. Since c=soundspeed, dimensional analysis already suggests the form of the answer. But the unknown coefficient in the box is important in practice:

$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u$ wave equation

new Q: $\Delta t \leq$?

$\Delta t \leq \square \frac{\Delta x}{c}$

Von Neumann stability anal. for PDEs

$u = u^{num.} + \text{error } \epsilon$

For linear eqs. ϵ: $\partial_t \epsilon = D \nabla^2 \epsilon$ (idea: Fourier)

discretize u_j^n ← n: timestep, j: spacial location; ϵ_j^n:

$$\epsilon_j^{n+1} = \epsilon_j^n + r\left(\epsilon_{j+1}^n - 2\epsilon_j^n + \epsilon_{j-1}^n\right)$$

$$\epsilon_j^n \sim E(t)\, e^{ikx}$$

e^{ikx} (wave along x); $e^{ikx} = \cos(kx) + i \sin(kx)$

Plug that into PDE

$$E(t+\Delta t) e^{ikx} = E(t) e^{ikx} + r\left(E(t) e^{ik(x+\Delta x)} - 2E(t) e^{ikx} + E(t) e^{ik(x-\Delta x)}\right) \quad \leq E(t)$$

Stability condition

$$|E(t+\Delta t)| \leq |E(t)| \iff \left|1 + r\left(e^{ik\Delta x} + e^{-ik\Delta x} - 2\right)\right| < 1$$

Trig: $\sin\frac{\theta}{2} = \frac{e^{i\theta/2} - e^{-i\theta/2}}{2i} \Rightarrow \sin^2\frac{\theta}{2} = \left(-\frac{1}{4}\right)\left(e^{i\theta} + e^{-i\theta} - 2\right)$, $\theta = k\Delta x$

Stab. crit.: $\left|1 - 4rn \sin^2 \frac{k\Delta x}{2}\right| < 1 \Rightarrow r = \frac{D\Delta t}{(\Delta x)^{2n}} < \frac{1}{2n}$ ($\sin^2$: 0 … $\frac{1}{2}$)

$\Delta t < \frac{1}{2} \cdot \frac{(\Delta x)^2}{D}$ (1-D)

$n = \#\dim$