Modelling 2D steady-state heat equation

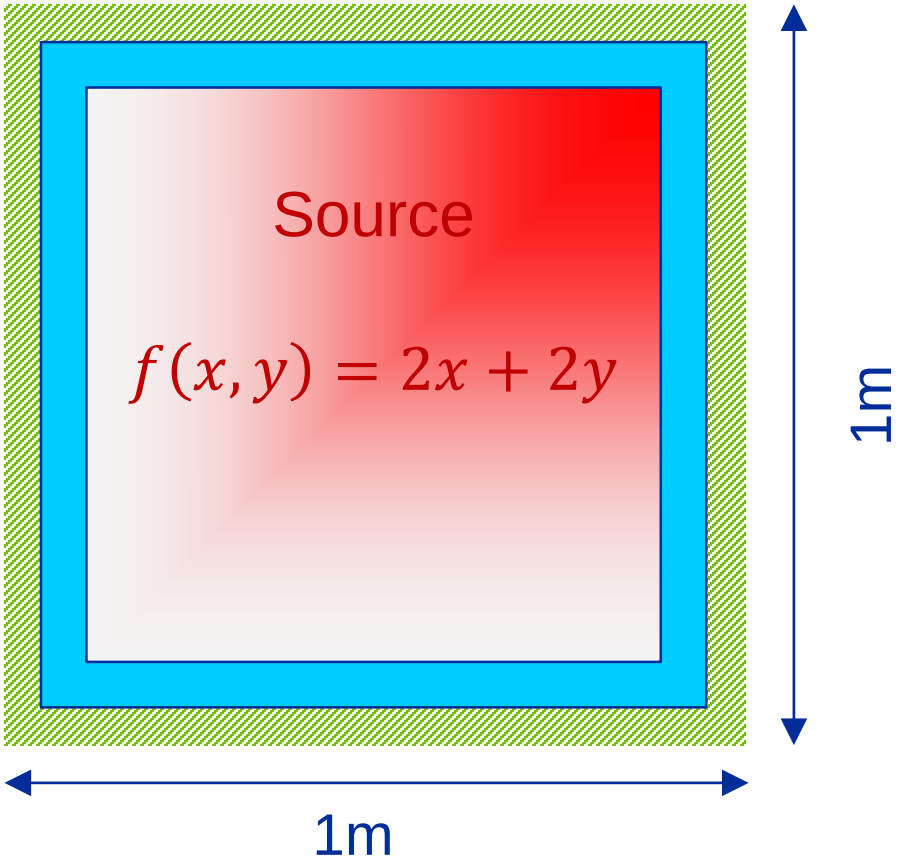
PTfS-CAM Project

Part I

PTfS 2020

Scenario – rectangular plate

Find steady state temperature distribution inside the plate!



Boundary
$T(\varphi) = 0$

Steady-state heat equation

$$-k\Delta u = f \quad \forall\,(x,y) \in \Omega$$
$$\mathrm{u}(x,y) = 0 \quad \forall\,(x,y) \in \partial\Omega$$

where $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$

Assume $k = 1$ (conductivity)

$$\Rightarrow -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f$$

Discretization

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

Use Finite Difference Method (FDM) for discretization

$$\Rightarrow -\Delta u = -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)(x,y) \approx$$

$$\frac{1}{h^2}(4\,u(x,y) - u(x-h,y) - u(x+h,y) - u(x,y-h) - u(x,y+h))$$

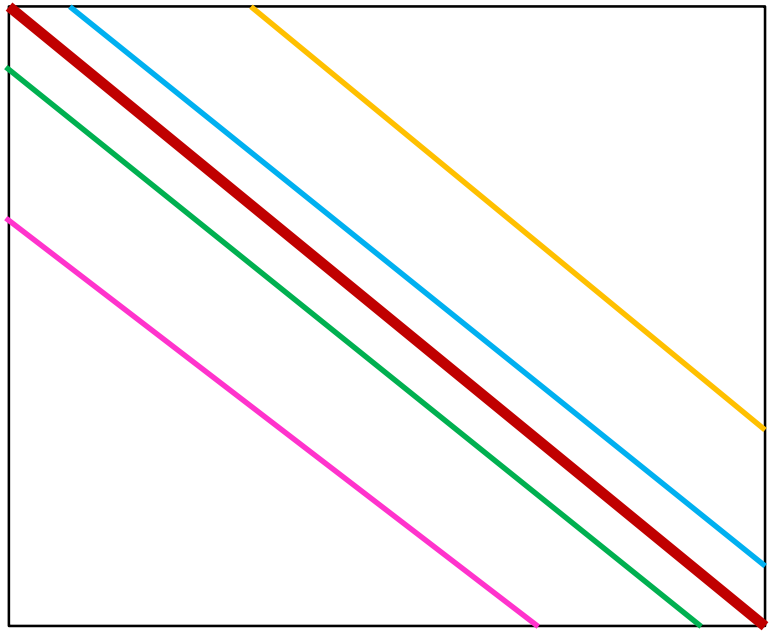
Linear system

$$\Rightarrow -\Delta u = f \approx \frac{1}{h^2}(4\,u(x,y) - 1\,u(x-h,y) - 1\,u(x+h,y) - 1\,u(x,y-h) - 1\,u(x,y+h))$$

$$\Rightarrow A\,u = f$$

Similar to 2d-5pt stencil

$A =$



$u =$

u(0,0)
u(0,h)
u(0,2h)
.
.
.
u(h,0)
u(h,h)
u(h,2h)
.
.
.

$f =$

f(0,0)
f(0,h)
f(0,2h)
.
.
.
f(h,0)
f(h,h)
f(h,2h)
.
.
.

2d-5pt Stencil

```
for j=1,jmax
  for k=1,kmax
    y[j,k] = (1/h²) * ( 4*x[j,k] – x[j-1,k] &
                  - x[j+1,k] – x[j,k-1] – x[j,k+1] )
  enddo
enddo
```

Solving the linear system

Solve for u : $A\,u = f$

1. Use Conjugate Gradient (CG)

2. Use Preconditioned Conjugate Gradient (PCG) with symmetric Gauss-Seidel preconditioning

Your TASKS

1. Copy the code from ~unrz002h/ptfs_cam_2020 :
   `cp -r ~unrz002h/ptfs_cam_2020 <target destination>`

2. Build the code using the given Makefile, i.e., just type
   `CXX=icpc make`

3. To switch on LIKWID measurement set the LIKWID flag to ‘on’, i.e.,
   `LIKWID=on CXX=icpc make`

4. Check for code correctness using the following commands:
   `./test`

5. To run the actual code us the following commands:
   `./perf  num_grids_y  num_grids_x`

6. If all tests pass, **parallelize building blocks using OpenMP**. Always observe correctness!

7. Are there any possible optimizations that you could do in the CG and PCG solver implemented in **`SolverClass::(P)CG`** (Solver.cpp)? If so, implement them!

Remarks regarding implementation
PCG example

```
r = b – A x
res_norm = <r,r>
z = P r0
α0 = <r,z>
p = z
```

Matrix-free implementation,
i.e., stencil updates

$P = (D + U)^{-1} D (D + L)^{-1}$

Think of solving this efficiently. See
`PDE::GS_precon` defined in PDE.cpp

```
while( (iter<niter) && (res_norm > tol*tol) )
      v  = A p
      λ  = α0 / <v,p>
      x  = x + λ p
      r  = r - λ v
      z  = Pr
      α1 = <r,z>
      p  = z + (α1/α0) p
      α0 = α1
      ++ iter
```

Things to take care of

- Think to use loop fusion wherever necessary.
- For debugging please compile code as
  `CXX=icpc make EXTRA_FLAGS=-DDEBUG`
- Sometimes it's useful for debugging to visualize your arrays. Use the function **`writeGnuplotFile`** and plot using **`splot`** in gnuplot if needed.
- Take particular care with parallelizing the Gauss-Seidel preconditioner (refer next exercise class).
- Use EMMY (Ivy Bridge) for getting your performance results.
- Check if the measurements are reproducible. (Think of pinning, scheduling, and clock frequency).
- Request a dedicated node for measuring performance.