GPU TECHNOLOGY CONFERENCE

Tridiagonal Solvers on the GPU and Applications to Fluid Simulation

Nikolai Sakharnykh, NVIDIA
nsakharnykh@nvidia.com

NVIDIA.

Agenda

- Introduction and Problem Statement
- Governing Equations
- ADI Numerical Method
- GPU Implementation and Optimizations
- Results and Future Work

Introduction

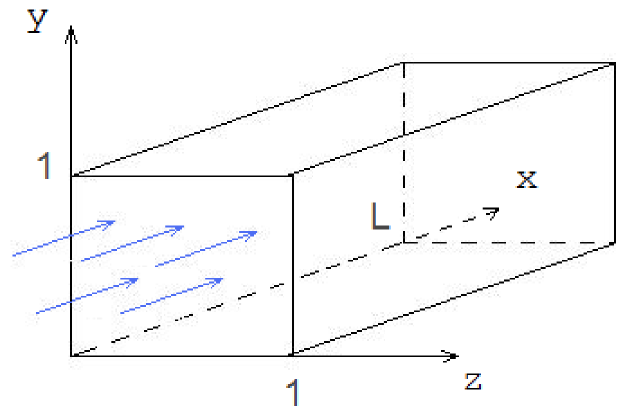
Turbulence simulation

- Direct Numerical Simulation
  - all scales of turbulence
  - expensive
- Large-Eddy Simulation
- Reynolds-Averaged Navier-Stokes

- Research at Computer Science department of Moscow State University
  – Paskonov V.M., Berezin S.B.

Problem Statement

- *Viscid incompressible* fluid in 3D domain
- Initial and boundary conditions
- Euler coordinates: velocity and temperature



Definitions

Density	$\rho = const = 1$
Velocity	$\mathbf{u} = (u, v, w)$
Temperature	T
Pressure	p

- State equation $p = \rho RT = RT$

  R - gas constant for air

Governing Equations

- Continuity equation

$$\operatorname{div}\,\mathbf{u} = 0$$

- Navier-Stokes equations
  - dimensionless form

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla T + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}$$

Re - Reynolds number

Reynolds number

- Similarity parameter
  - – the ratio of inertia forces to viscosity forces

- 3D channel:

$$\mathrm{Re} = \frac{V'L'}{\mu'}$$

V' - mean velocity

L' - length of pipe

μ' - dynamic viscosity

- High Re – turbulent flow
- Low Re – laminar flow

Governing Equations

- Energy equation
  - dimensionless form

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = -\nabla T + \frac{1}{\mathrm{Pr} \cdot \mathrm{Re}} \Delta T + \frac{\gamma - 1}{\gamma \cdot \mathrm{Re}} \Phi$$

Pr - Prandtl number

γ - heat capacity ratio

Φ - dissipative function

Numerical Method

- Alternating Direction Implicit (ADI)

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

X Y Z

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \qquad \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} \qquad \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial z^2}$$

ADI – Heat Conduction

- 3 fractional steps – X, Y, Z
- Implicit finite-difference scheme

$$\boxed{\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}} \longrightarrow \frac{u_{i,j,k}^{n+1/3} - u_{i,j,k}^{n}}{\Delta t} = \frac{u_{i+1,j,k}^{n+1/3} - 2u_{i,j,k}^{n+1/3} + u_{i-1,j,k}^{n+1/3}}{\Delta x^2}$$

$$q = 2 + \frac{\Delta x^2}{\Delta t} \qquad \begin{pmatrix} q & -1 & & & 0 \\ -1 & q & -1 & & \\ & -1 & q & -1 & \\ & & -1 & \ddots & -1 \\ 0 & & & -1 & q \end{pmatrix} \cdot \begin{pmatrix} u_{1,j,k}^{n+1/3} \\ \vdots \\ u_{i,j,k}^{n+1/3} \\ \vdots \\ u_{nx,j,k}^{n+1/3} \end{pmatrix} = \frac{\Delta x^2}{\Delta t} \begin{pmatrix} u_{1,j,k}^{n} \\ \vdots \\ u_{i,j,k}^{n} \\ \vdots \\ u_{nx,j,k}^{n} \end{pmatrix}$$

ADI – Navier-Stokes

- Equation for X velocity

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}=-\frac{\partial T}{\partial x}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)$$

X

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=-\frac{\partial T}{\partial x}+\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2}\right)$$

Y

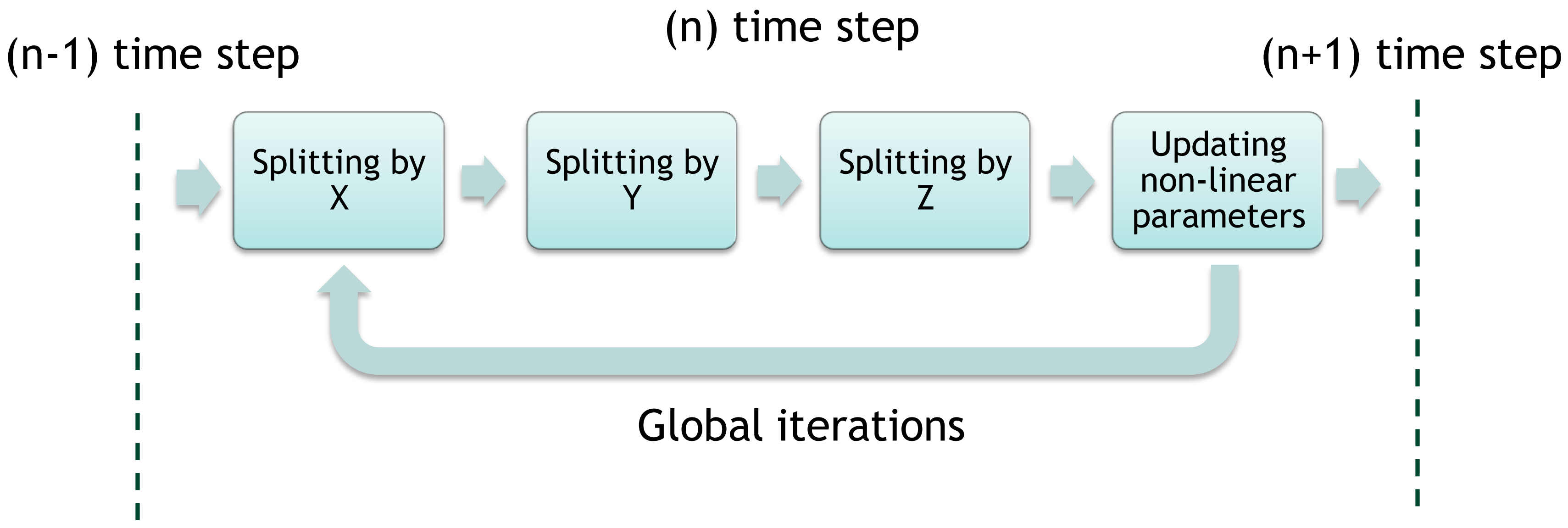
$$\frac{\partial u}{\partial t}+v\frac{\partial u}{\partial y}=\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial y^2}\right)$$

Z

$$\frac{\partial u}{\partial t}+w\frac{\partial u}{\partial z}=\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial z^2}\right)$$

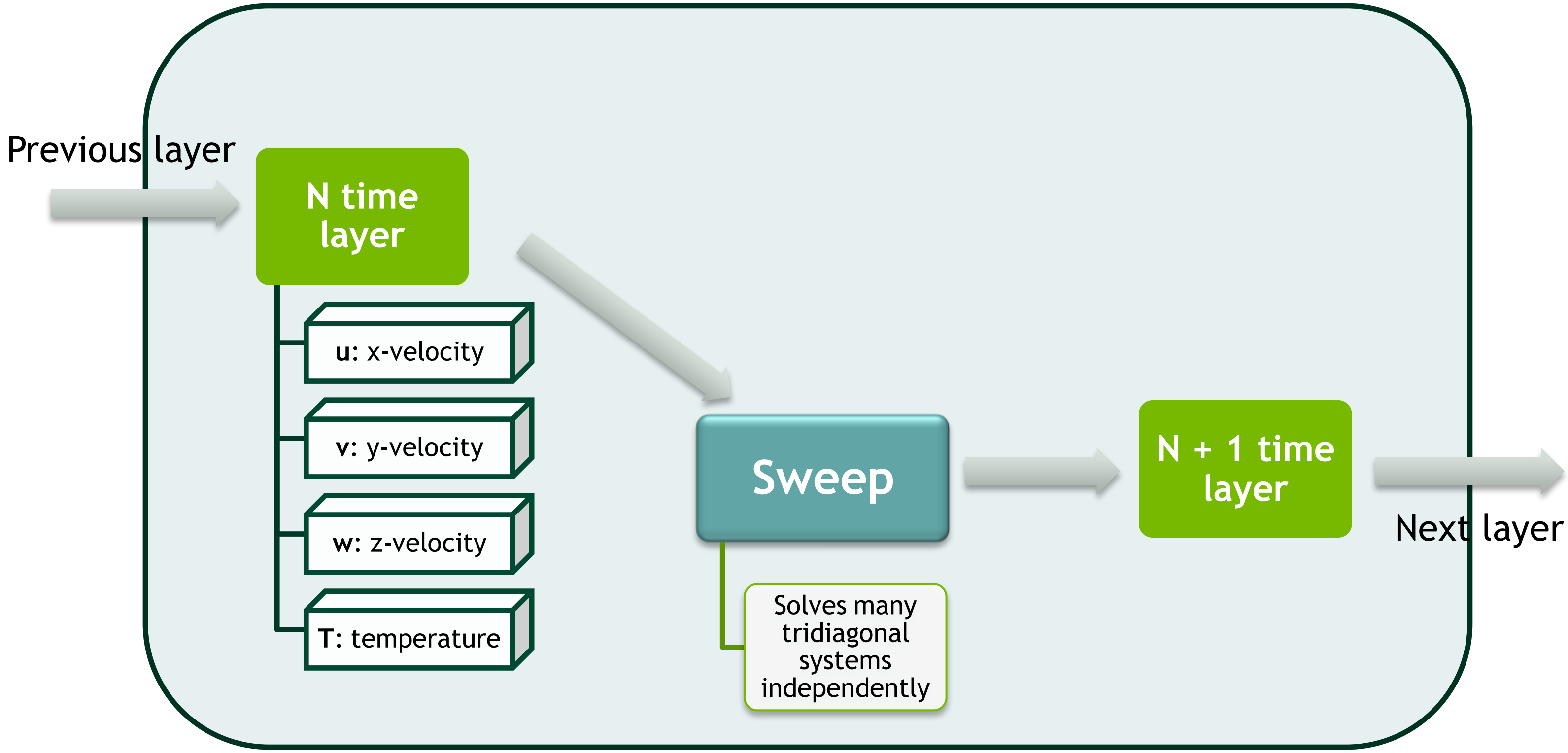
  – need iterations for non-linear PDEs

ADI - Time Step



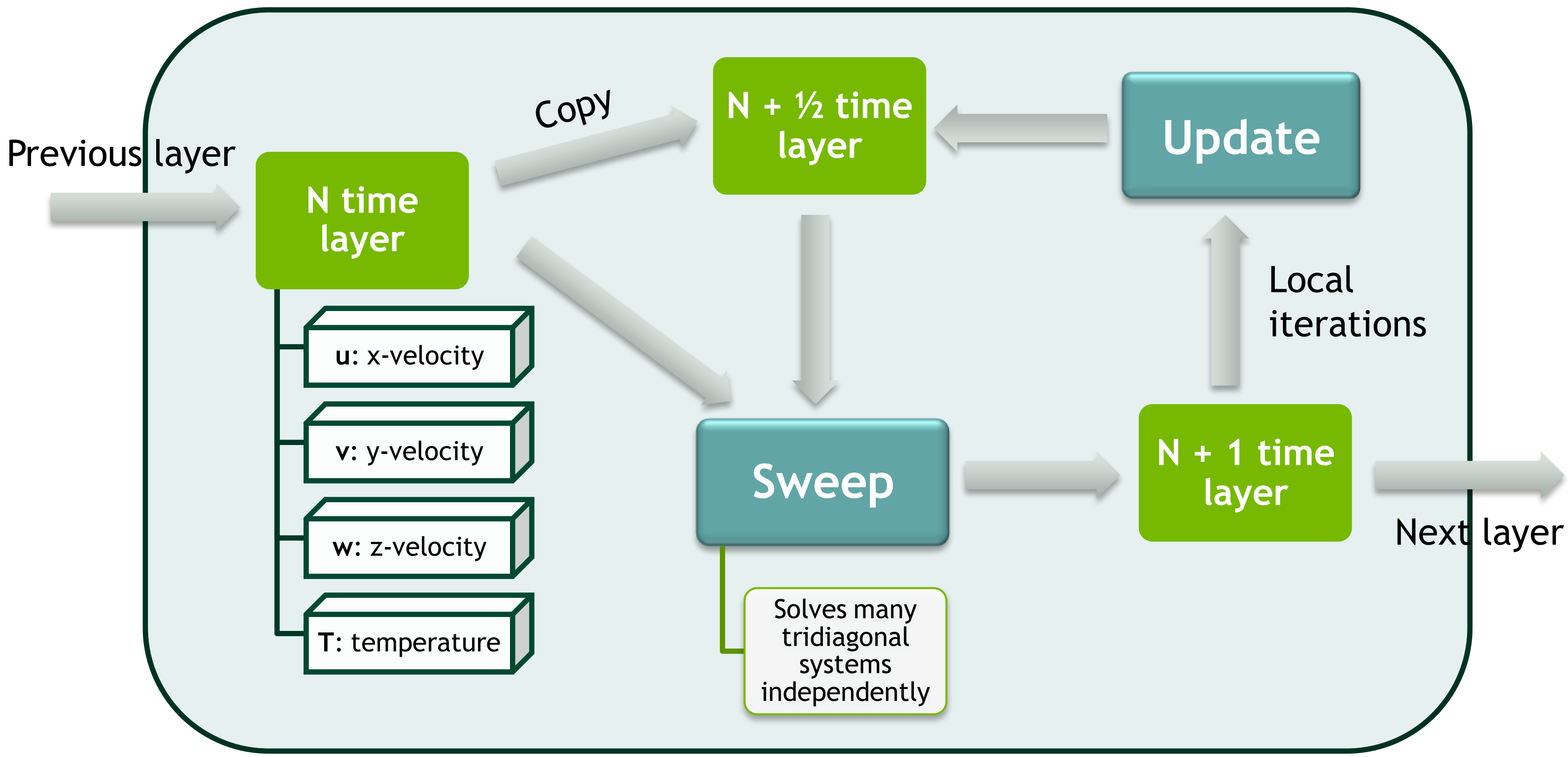
ADI - Fractional Time Step

- Linear PDEs



ADI - Fractional Time Step

- Non-Linear PDEs



Main Stages of the Algorithm

- Solve a lot of independent tridiagonal systems
  - Computationally intensive
  - Easy to parallelize
- Subtasks:
  - Evaluate dissipation term
  - Update non-linear parameters

Tridiagonal Solvers Overview

- Simplified Gauss elimination
  - Also known as Thomas algorithm, Sweep
  - The fastest serial approach
- Cyclic Reduction methods
  - Attend Yao Zhang's talk "Fast Tridiagonal Solvers" afterwards!

Sweep algorithm

- Memory requirements
  - one additional array of size N
- Forward elimination step
- Backward substitution step
- Complexity: O(N)

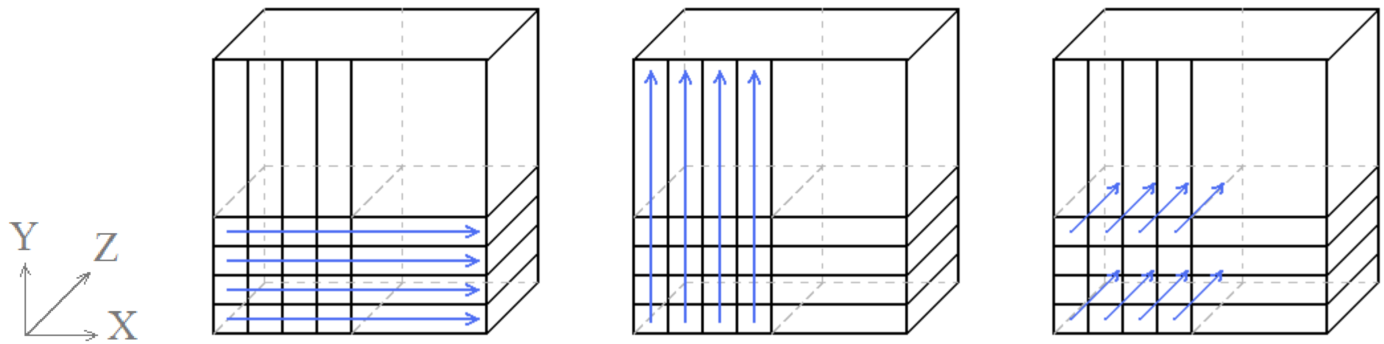
GPU Implementation

- All data arrays are stored on GPU
- Several 3D time-layers
  - overall 1GB for 192x192x192 grid in DP

- Main kernels
  - Sweep
  - Dissipative function evaluation
  - Non-linear update

Sweep on the GPU

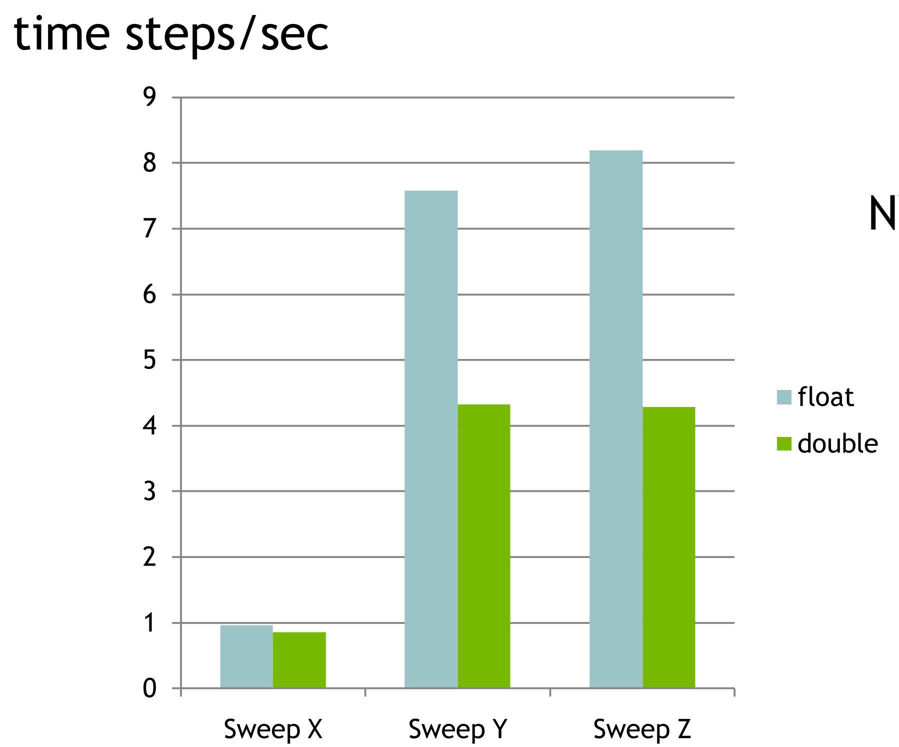
- One thread solves one system
  - N^2 systems on each fractional step



Splitting by X　　Splitting by Y　　Splitting by Z

- Each thread operates with 1D slice in corresponding direction

Sweep - performance

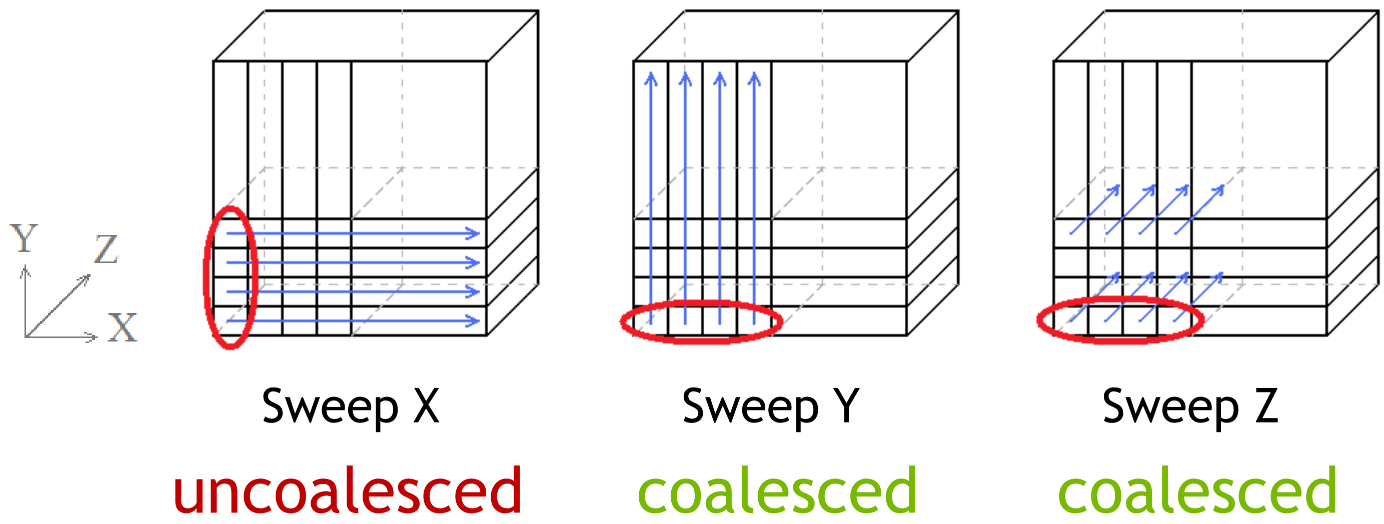


NVIDIA Tesla C1060

- X splitting is much slower than Y/Z ones

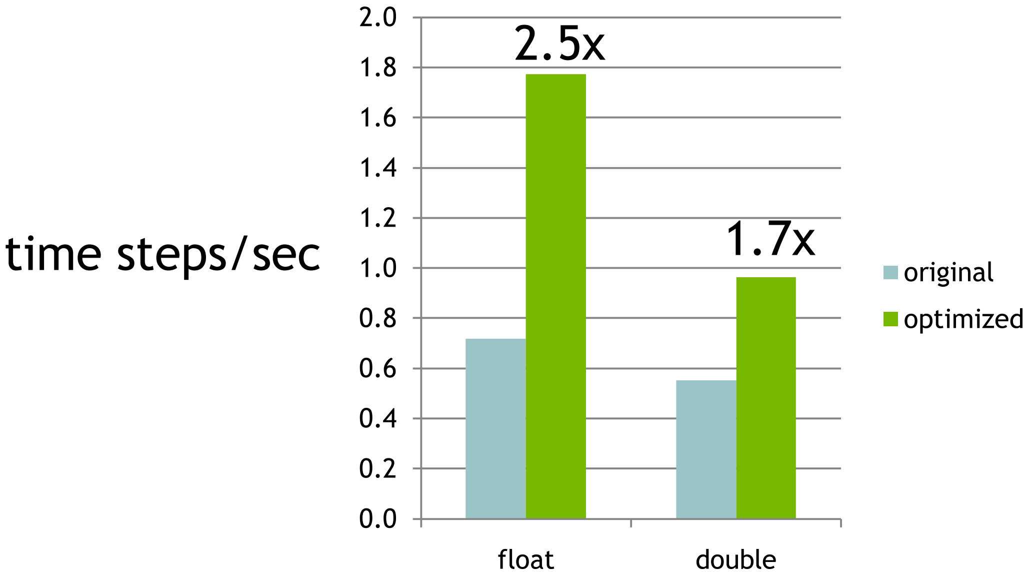
Sweep - going into details

- Memory bound – need to optimize access to the memory



Sweep - optimization

- Solution for X-splitting
  - Reorder data arrays and run Y-splitting
  - Need few additional 3D matrix transposes



Code analysis

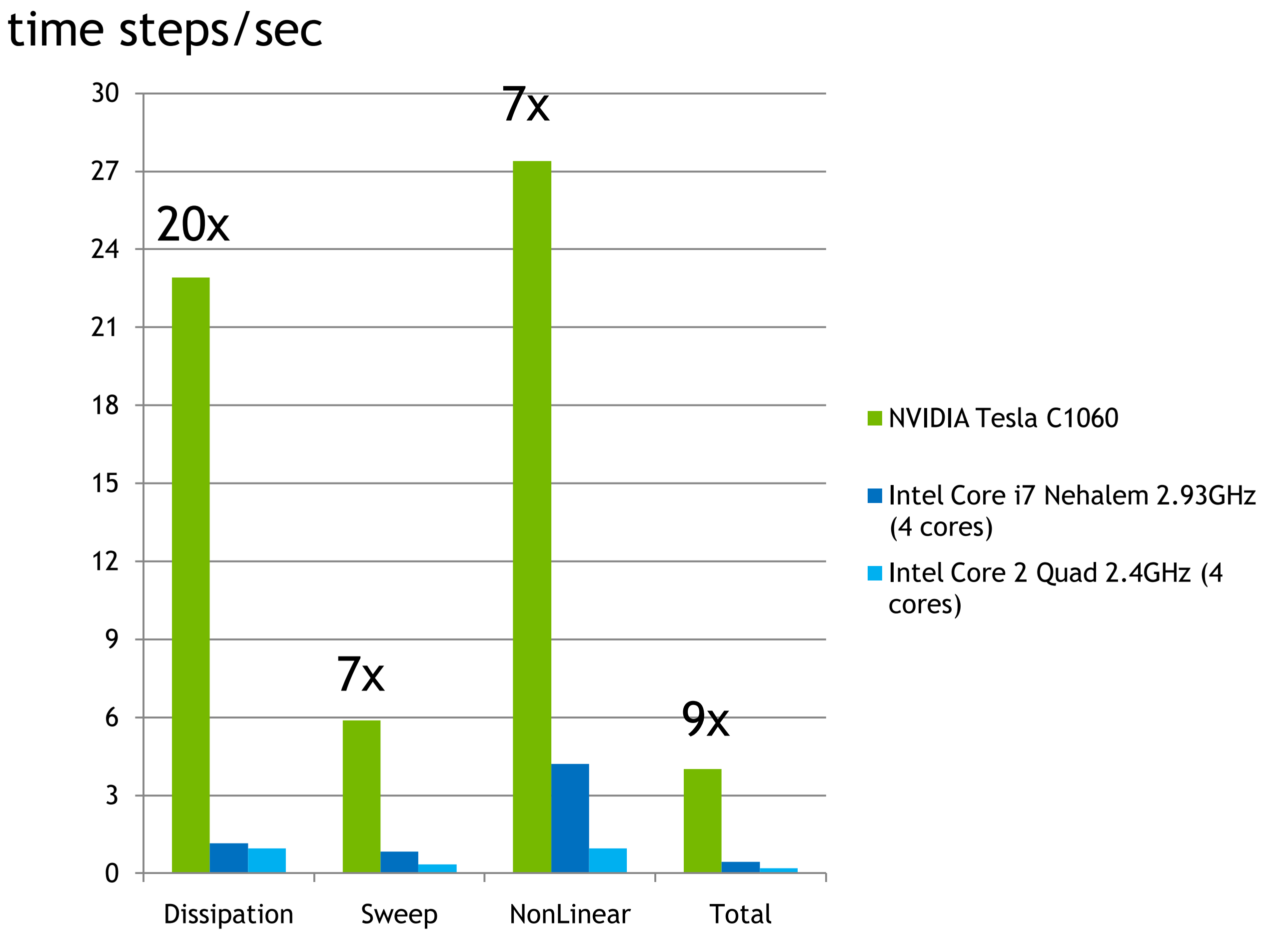
- GPU version is based on the CPU code

```
// boundary conditions
switch (dir)
{
	case X: case X_as_Y: bc_x0(…); break;
	case Y: bc_y0(…); break;
	case Z: bc_z0(…); break;
}
a[1] = - c1 / c2;
u_next[base_idx] = f_i / c2;

// forward trace of sweep
int idx = base_idx;
int idx_prev;
for (int k = 1; k < n; k++)
{
	idx_prev = idx;
	idx += p.stride;

	double c = v_temp[idx];
	c1 = p.m_c13 * c - p.h;
	c2 = p.m_c2;
	c3 = - p.m_c13 * c - p.h;

	double q = (c3 * a[k] + c2);
	double t = 1 / q;
	a[k+1] = - c1 * t;
	u_next[idx] = (f[idx] - c3 * u_next[idx_prev]) * t;
}
```

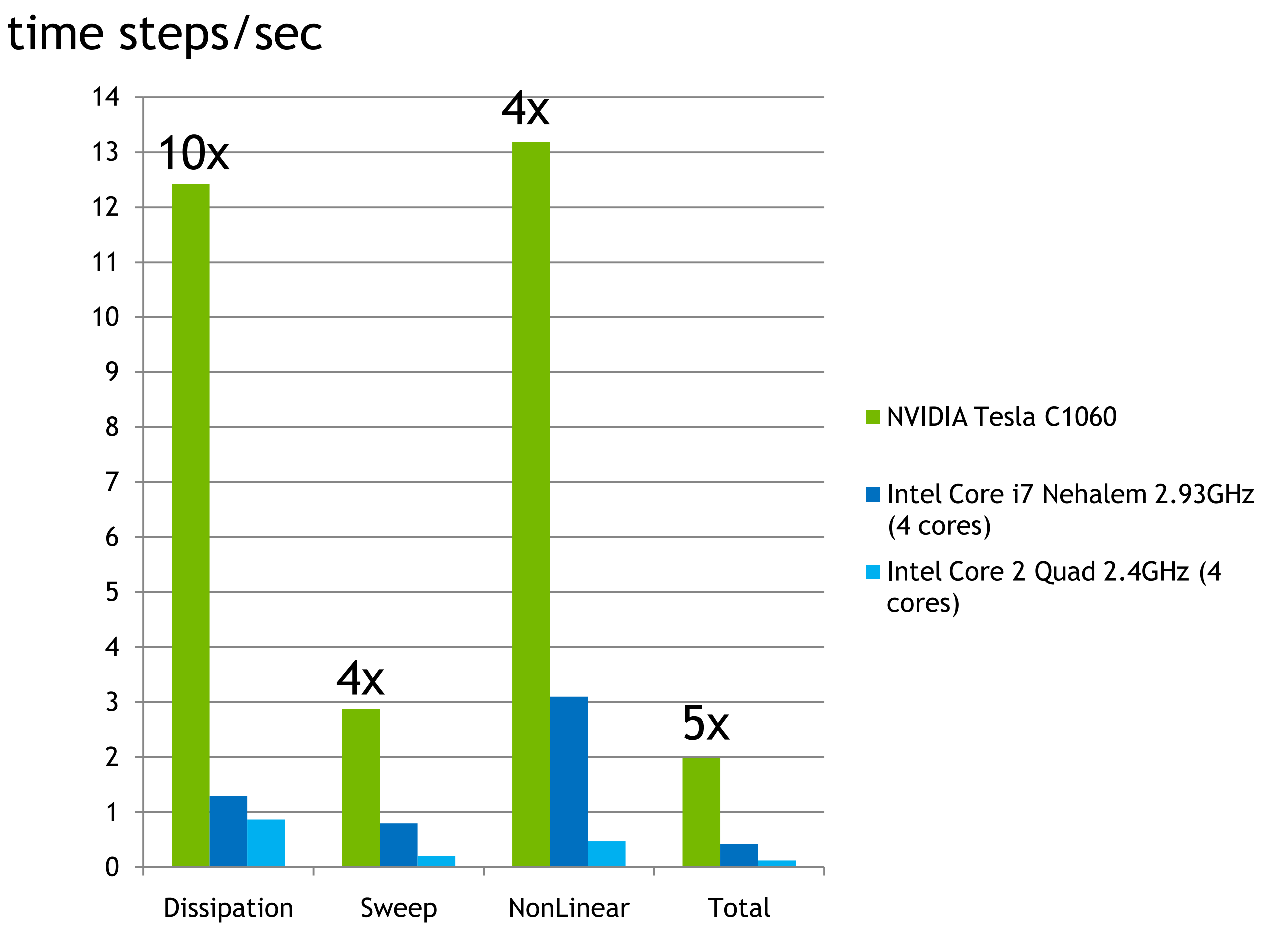
Performance Comparison

- Test data
  - Grid size of 128/192
  - 8 non-linear iterations (2 inner x 4 outer)
- Hardware
  - NVIDIA Tesla C1060
  - Intel Core2 Quad (4 threads)
  - Intel Core i7 Nehalem (8 threads)

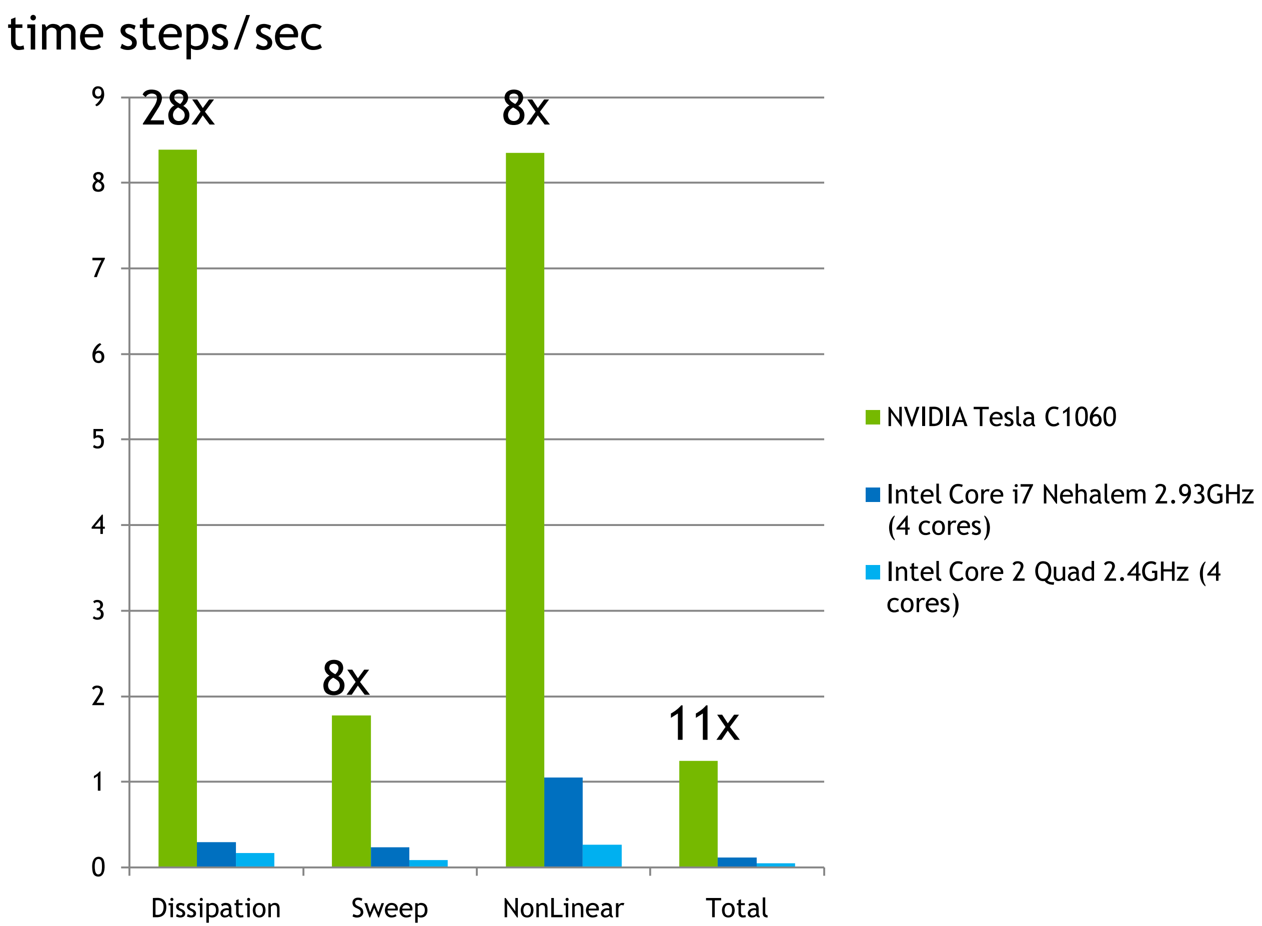
Performance - 128 - float



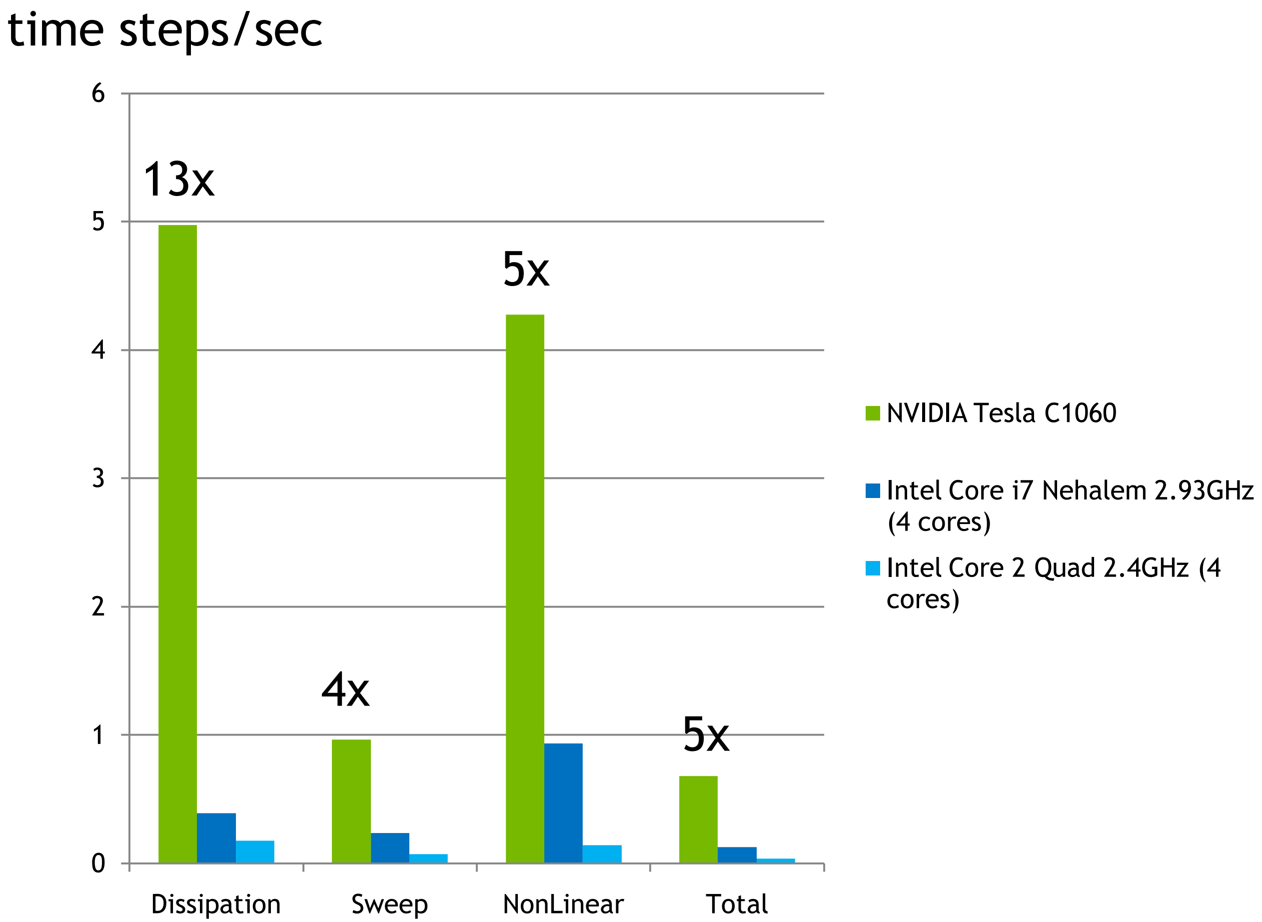
Performance - 128 - double



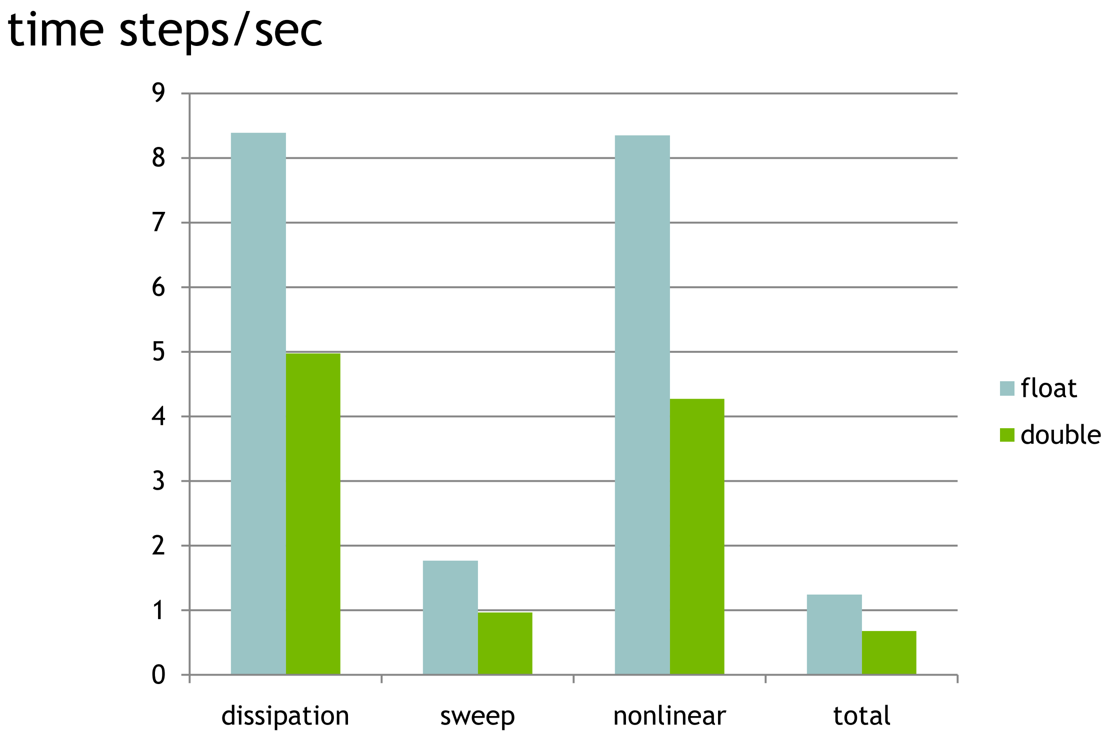
Performance - 192 - float



Performance - 192 - double



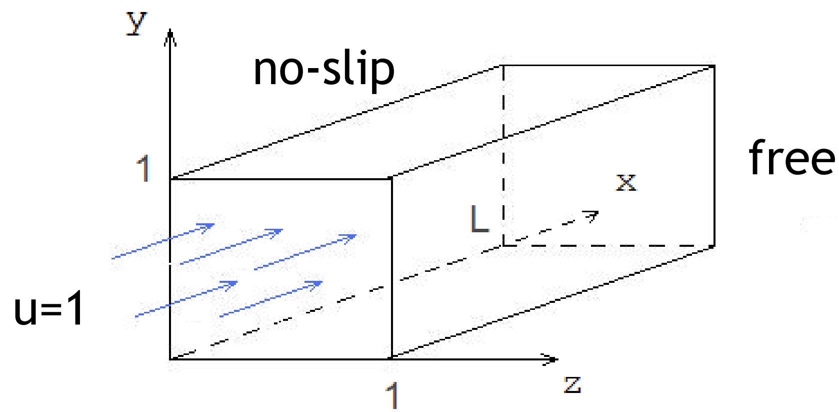
GPU performance - SP/DP



- In double precision GPU is only 2x slower than in single precision

Visual results

- Boundary conditions

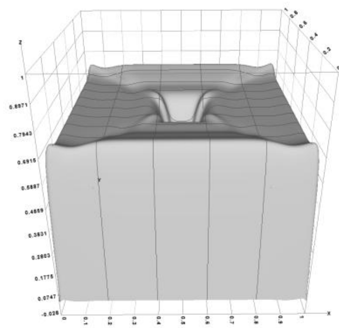


  - Constant flow at start: $u = 1, v = w = 0$
  - No-slip on sides: $u = v = w = 0$
  - Free at far end: $\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 v}{\partial x^2} = \frac{\partial^2 w}{\partial x^2} = 0$

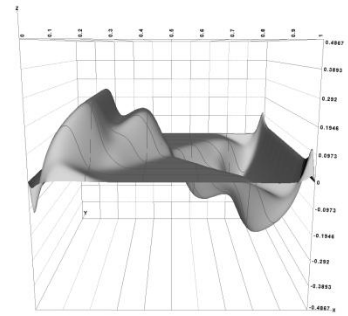
GPU TECHNOLOGY CONFERENCE

Visual Results - X-slice

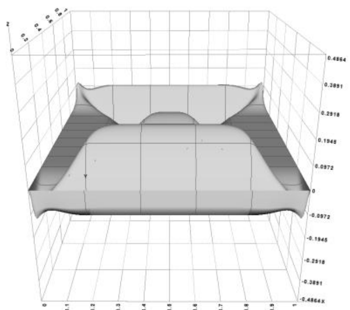
u



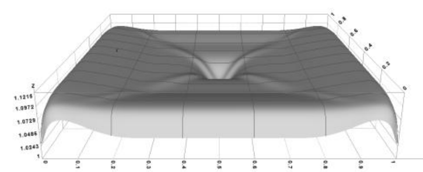
v



w



T



x = 0,9
t = 6

Re = 1000

NVIDIA.

Future Work

- Effective multi-GPU usage
  - Distributed memory systems

- Performance improvements

- High resolution grids, high Reynolds numbers

Conclusion

- High performance and efficiency of GPUs in complex 3D fluid simulation
- CUDA is an easy-to-use tool for GPU compute programming
- GPU enables new possibilities for researching

Questions?

Thank you!

- Keywords:
  - ADI, Tridiagonal Solvers, DNS, Turbulence

- E-mail: nsakharnykh@nvidia.com

References

- Paskonov V.M., Berezin S.B., Korukhova E.S. (2007) “A dynamic visualization system for multiprocessor computers with common memory and its application for numerical modeling of the turbulent flows of viscous fluids”, Moscow University Computational Mathematics and Cybernetics

- *ADI method* - Douglas Jr., Jim (1962), "Alternating direction methods for three space variables", *Numerische Mathematik* **4**: 41–63

Dissipative Function

$$\Phi = \Phi_x + \Phi_y + \Phi_z$$

$$\Phi_x = 2\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2 + \frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + \frac{\partial u}{\partial z}\frac{\partial w}{\partial x}$$

$$\Phi_y = \left(\frac{\partial u}{\partial y}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial y}\right)^2 + \frac{\partial v}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial z}\frac{\partial w}{\partial y}$$

$$\Phi_z = \left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial z}\right)^2 + 2\left(\frac{\partial w}{\partial z}\right)^2 + \frac{\partial w}{\partial x}\frac{\partial u}{\partial z} + \frac{\partial w}{\partial y}\frac{\partial v}{\partial z}$$