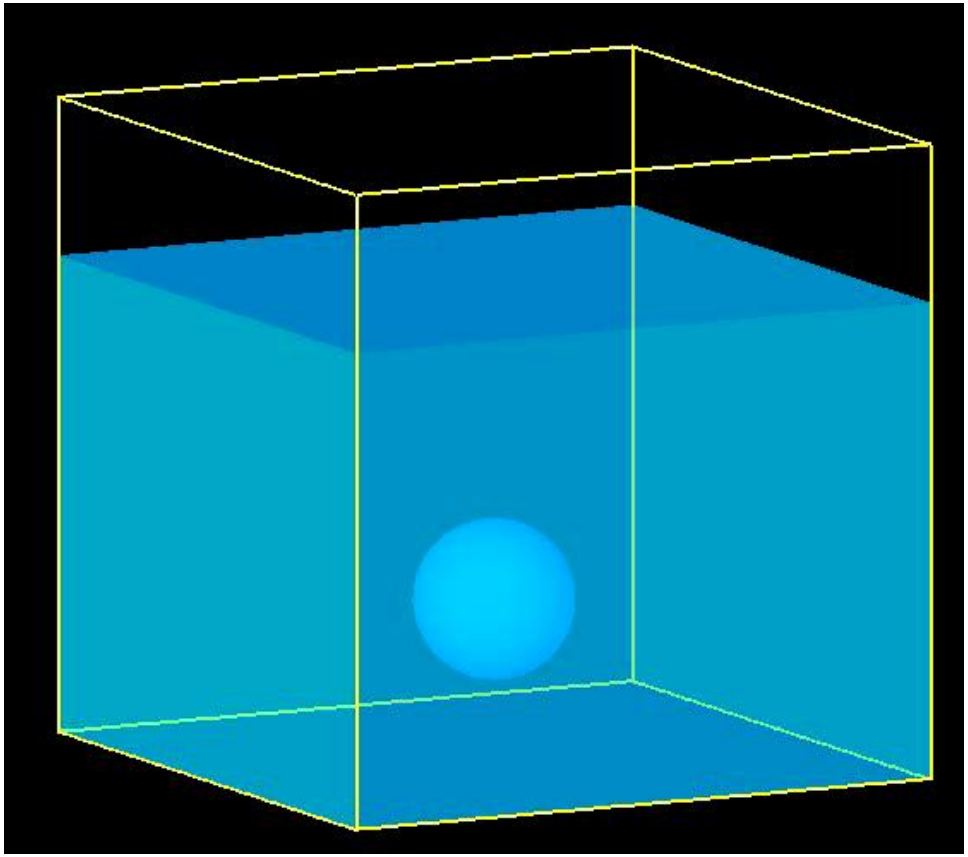
Implementation of Deflated Preconditioned Conjugate Gradient on the GPU

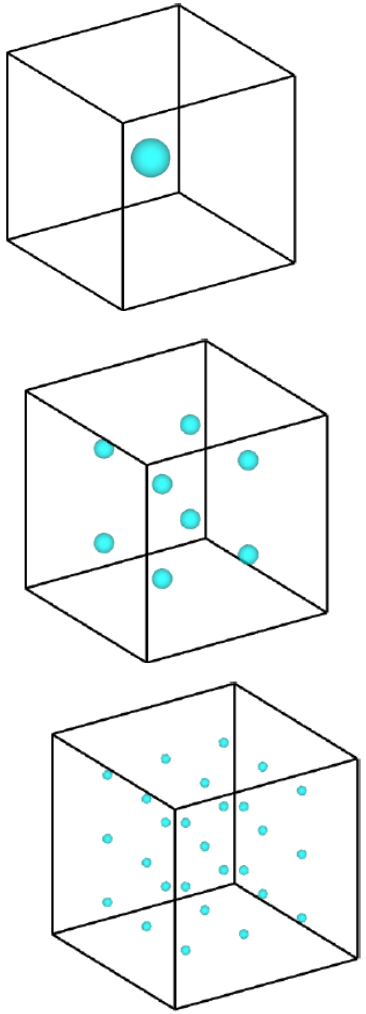
Rohit Gupta

Responsible Professor: Prof. Dr. Ir. Henk Sips
Supervisors: Prof. Dr. Ir. Kees Vuik and Ir. C.W.J. Lemmens

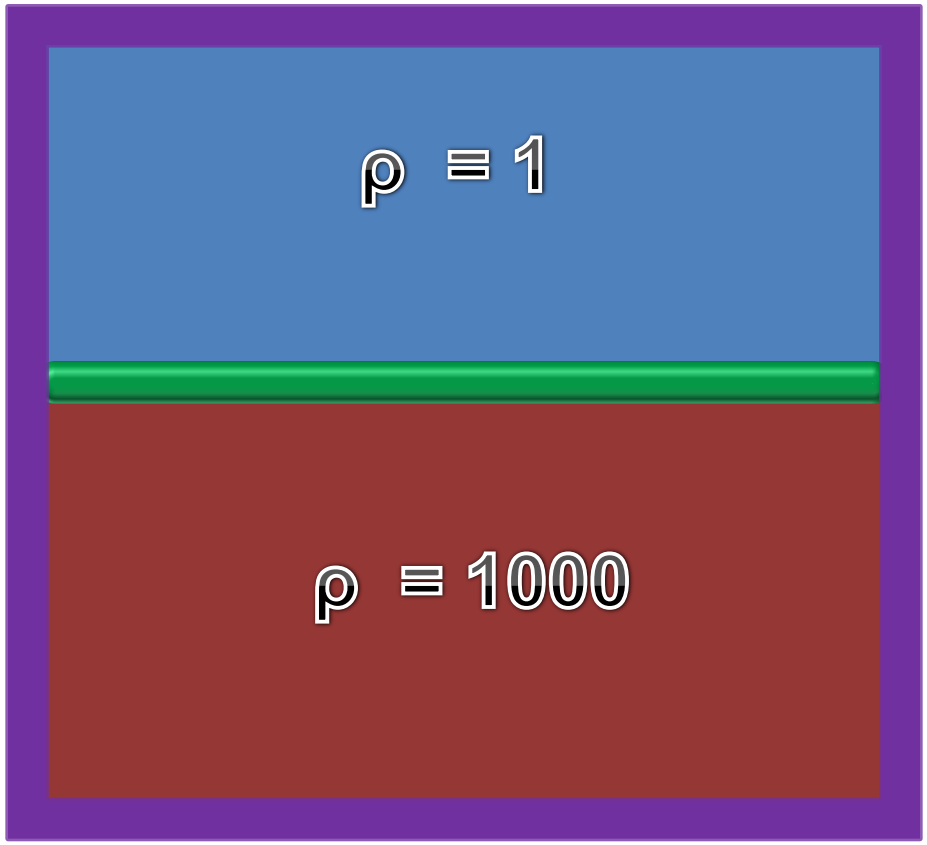
TU Delft

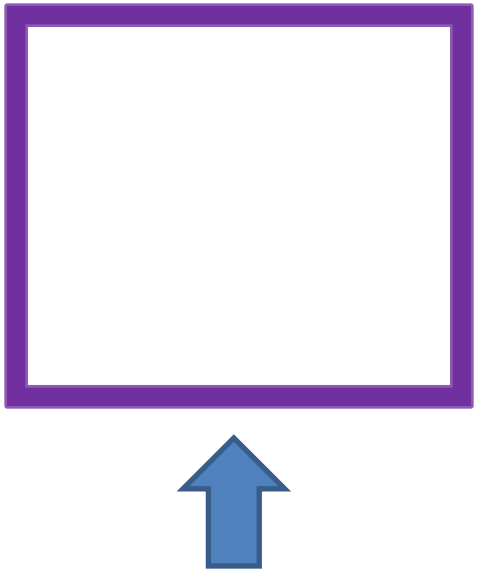
Two-Phase Fluid Flow





Model for Two Phase Computation





Boundary Conditions

$$
\begin{aligned}
-\nabla \cdot \left(\frac{1}{\rho(\mathbf{x})} \nabla p(\mathbf{x}) \right) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\
\frac{\partial}{\partial \mathbf{n}} p(\mathbf{x}) &= g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega,
\end{aligned}
$$

4	-1						
-1	4	-1					
	-1	4	-1				
		-1	4	-1			
			-1	4	-1		
				-1	4	-1	
					-1	4	-1
						-1	4

1.5015

-0.5005

-0.5005

2.002

-0.5005

-0.5005

2.002

-0.5005

-0.5005

2.002

-0.5005

-0.5005

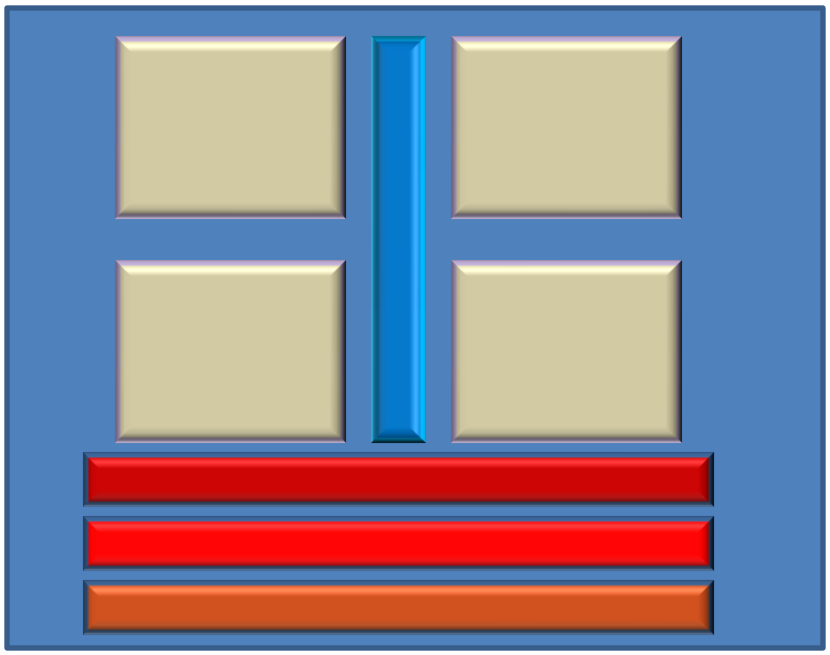
2.002

-0.5005

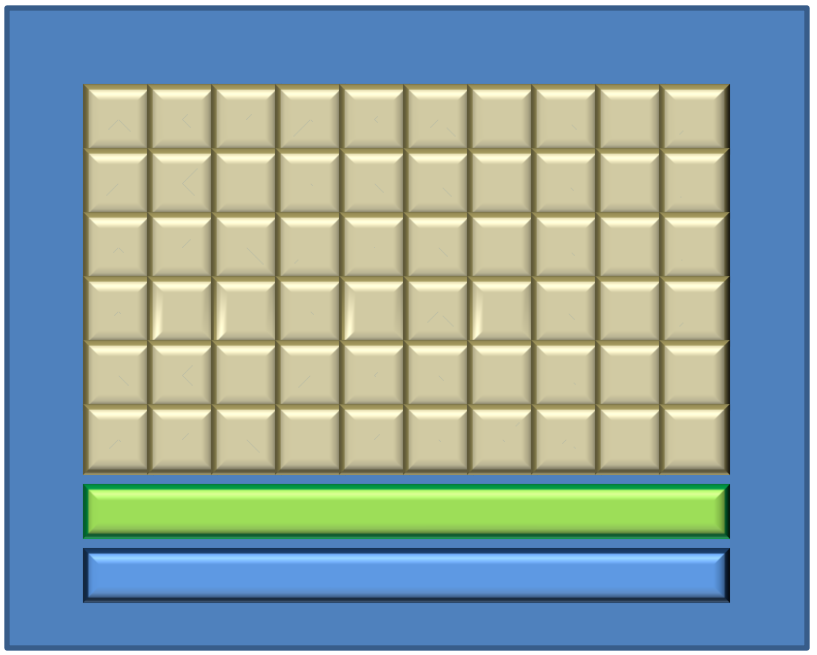
-0.5005

2.002

GPU Architecture



CPU



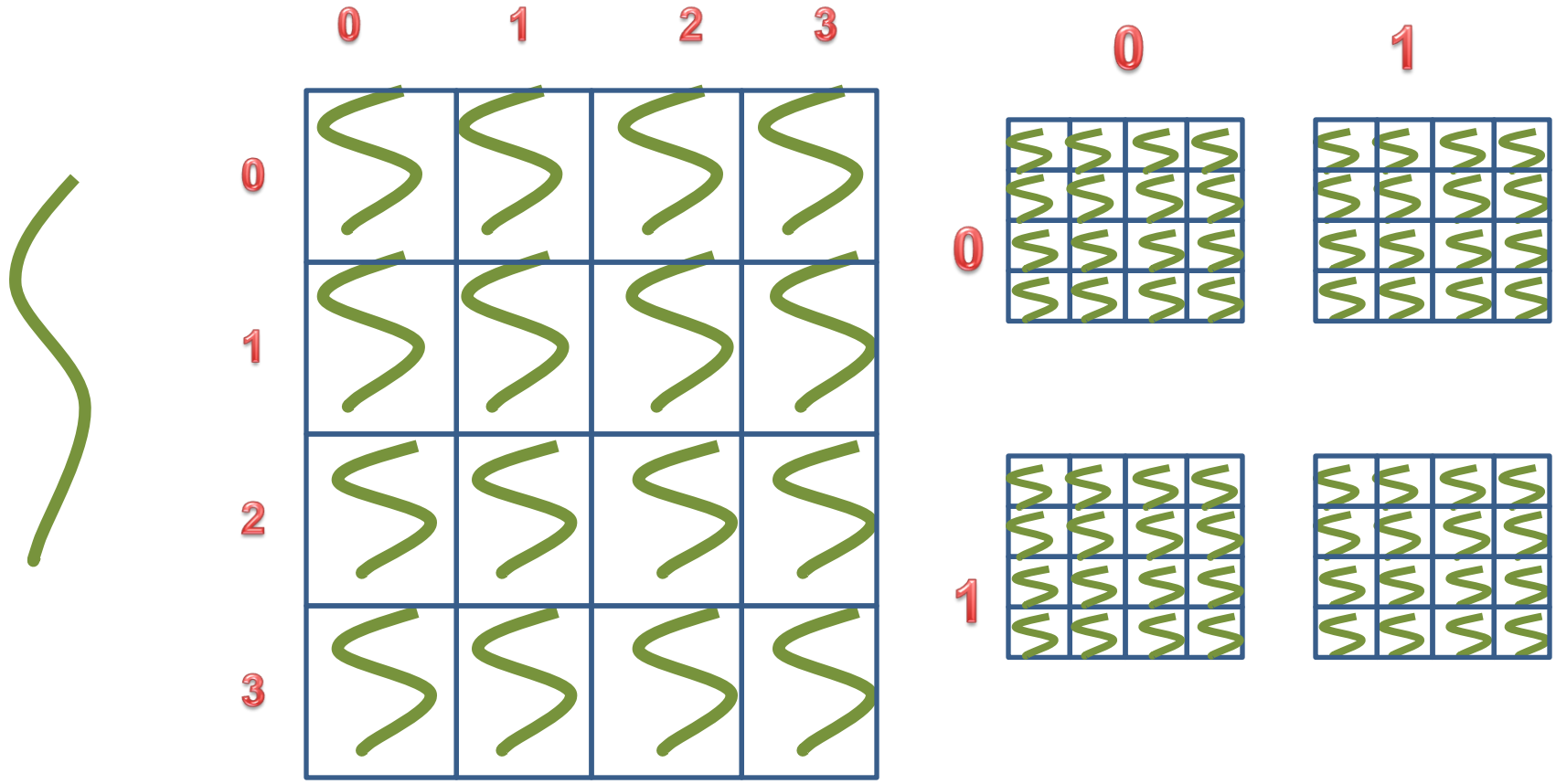
GPU

TU Delft

Memory Bandwidth

TU Delft

Launch Configuration



TU Delft

Key Optimizations

- **Coalescing**
- **Minimum Memory Transfers**
- **Minimum Divergence**
- **Caching**

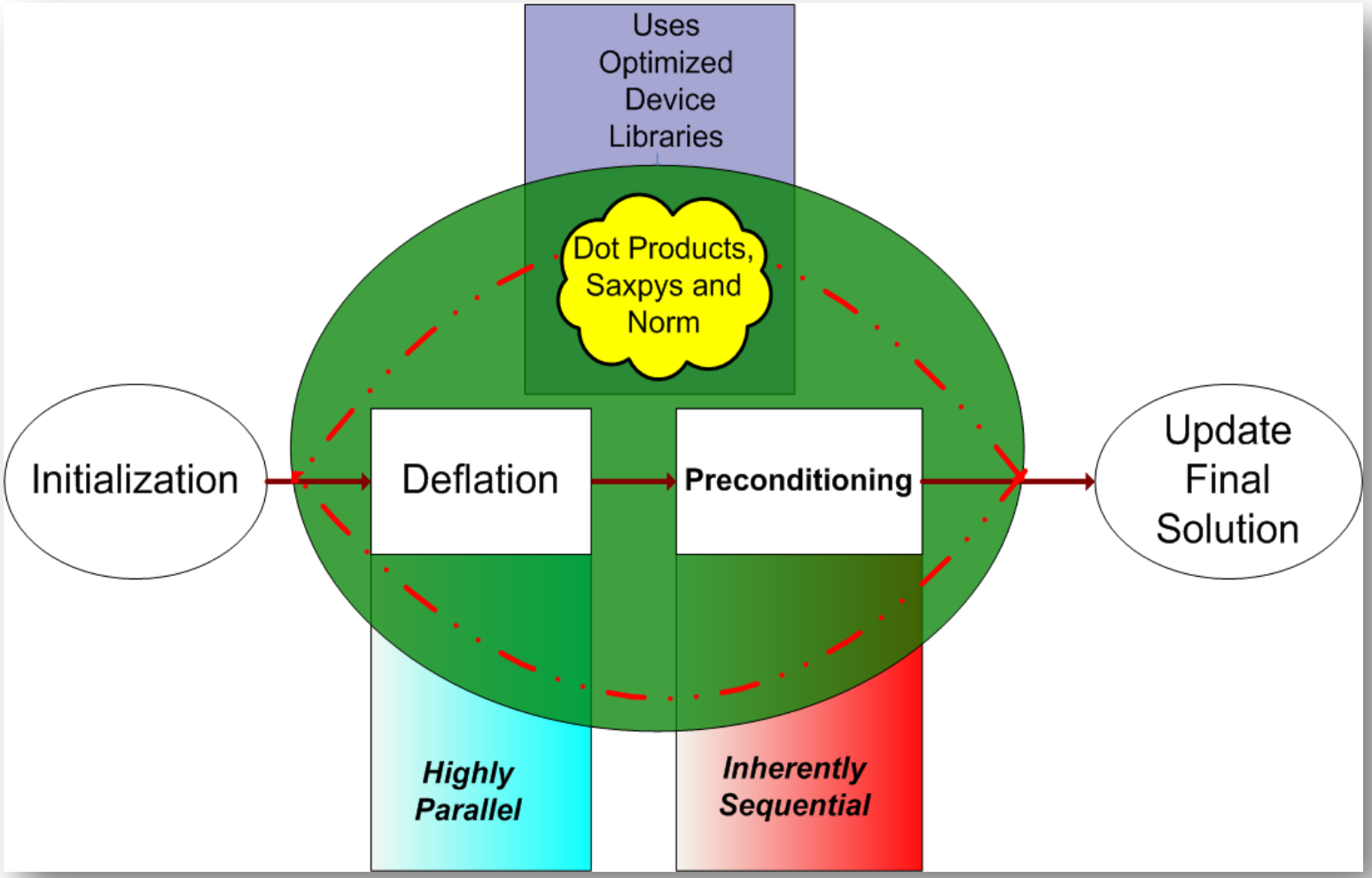
TU Delft

Related Work

- **SpMV Kernel**
- **Preconditioning Kernels**
- **Deflation Kernels**

TU Delft

Two-Phase Fluid Flow



Implementation GPU

- **SpMV Kernel**
- **Preconditioning Kernels**
- **Deflation Kernels**

TU Delft

Implementation CPU

- **Meschach/ gotoBLAS**
- **Compiler Options**
- **Same Data Structures**

TU Delft

Flavors

- **Conjugate Gradient (CG)**
- **CG with Preconditioning**
- **CG with Preconditioning and Deflation**

TU Delft

Preconditioning

- **Diagonal**
- **Incomplete Cholesky**
- **Incomplete Poisson**

TU Delft

Deflation

- **Important Operations**
- **Subdomains**
- **Deflation Kernels**

TU Delft

Methodology

- **Step-by-Step Optimizations**
- **Profiler for Advice**
- **Modularity and Readability**

TU Delft

Conjugate Gradient - Vanilla

- **DIA Format is best**
- **SpmV dominate GPU execution**
- $10^{-3} < \frac{\| X_{exact} - X_k \|_2}{\| X_k \|_2} < 10^{-5}$

TU Delft

Preconditioning
Block Incomplete Cholesky

- **Preconditioning dominates execution**
- **More Blocks – Better SpeedUp**
- **Shared Memory Use restricted**

TU Delft

Deflation

- **Speed Up recovers**
- **Preconditioning gets company in** $AZ \times E^{-1}b$
- **More Deflation Vectors – More Speed Up**

TU Delft

Deflated Preconditioned Conjugate Gradient

- **Storage of AZ optimized**
- **Speed Up Suffers – CPU performs better**
- **Optimizations on CPU**

TU Delft

Deflated Preconditioned Conjugate Gradient

- **Incomplete Poisson Preconditioning**
- **Speed Up favoring move**
- $\frac{\| X_{exact} - X_k \|_2}{\| X_k \|_2}$ IP $\simeq$ $\frac{\| X_{exact} - X_k \|_2}{\| X_k \|_2}$ BIC

TU Delft

Deflated Preconditioned Conjugate Gradient

- **Matrix Vector Multiplication** $E^{-1}b$ **optimized**
- **Kernels utilize 85% of bandwidth**
- **Clubbing Kernels – Reducing Memory Traffic**

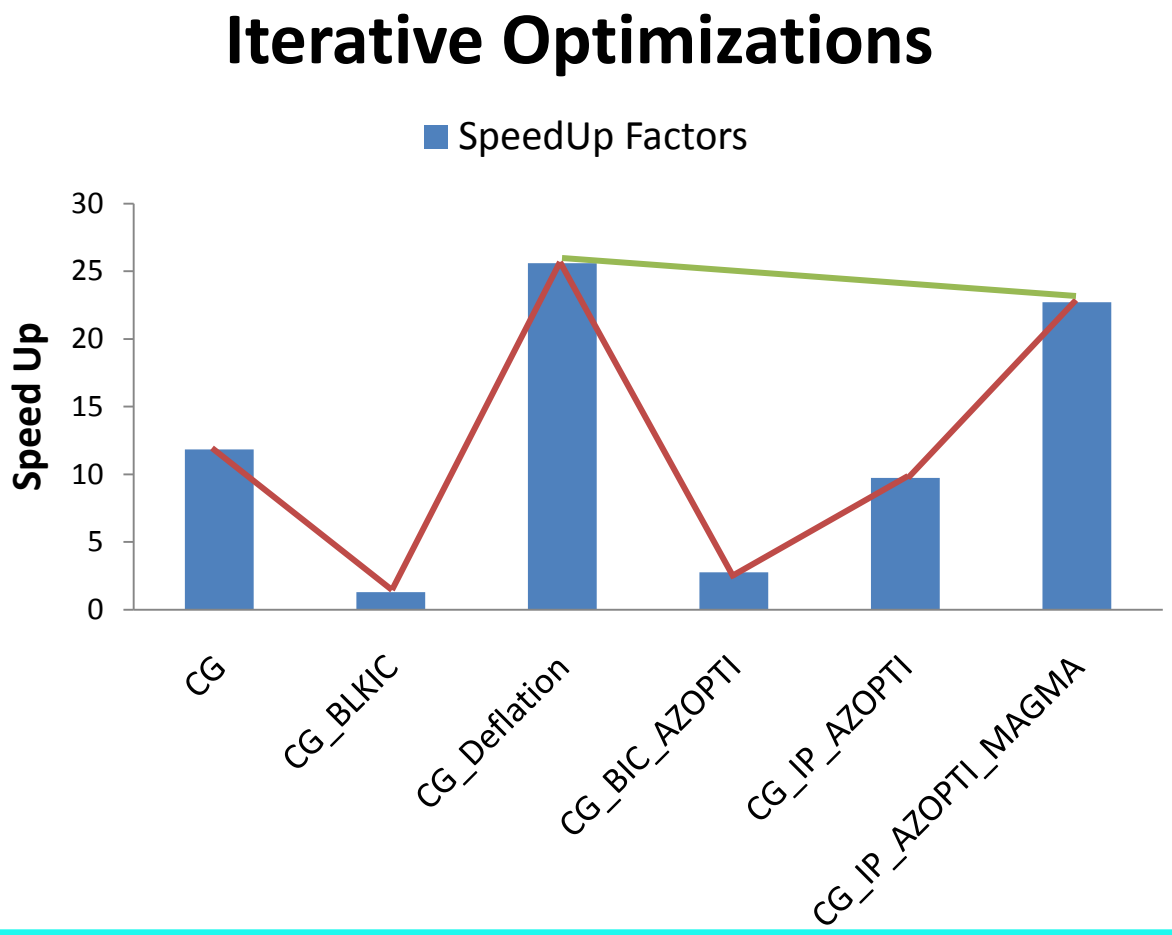
TU Delft

Two Phase Matrix

- **Density Contrast 1000:1**
- $0 < \frac{\| X_{exact} - X_k \|_2}{\| X_k \|_2} < 10^{-1}$
- **False Stop**

TU Delft

Performance Timeline



Analysis

- **Very close to possible peak performance**
- **Bandwidth bound Kernels**
- **Platform Utilization – Startling Facts**

TU Delft

Future Work

- **Multi-GPU Multi-CPU**
- **3D Domains, More Interfaces, Mixed Precision**
- **Different Preconditioners, Grid Types.**

TU Delft

Conclusions

TU Delft

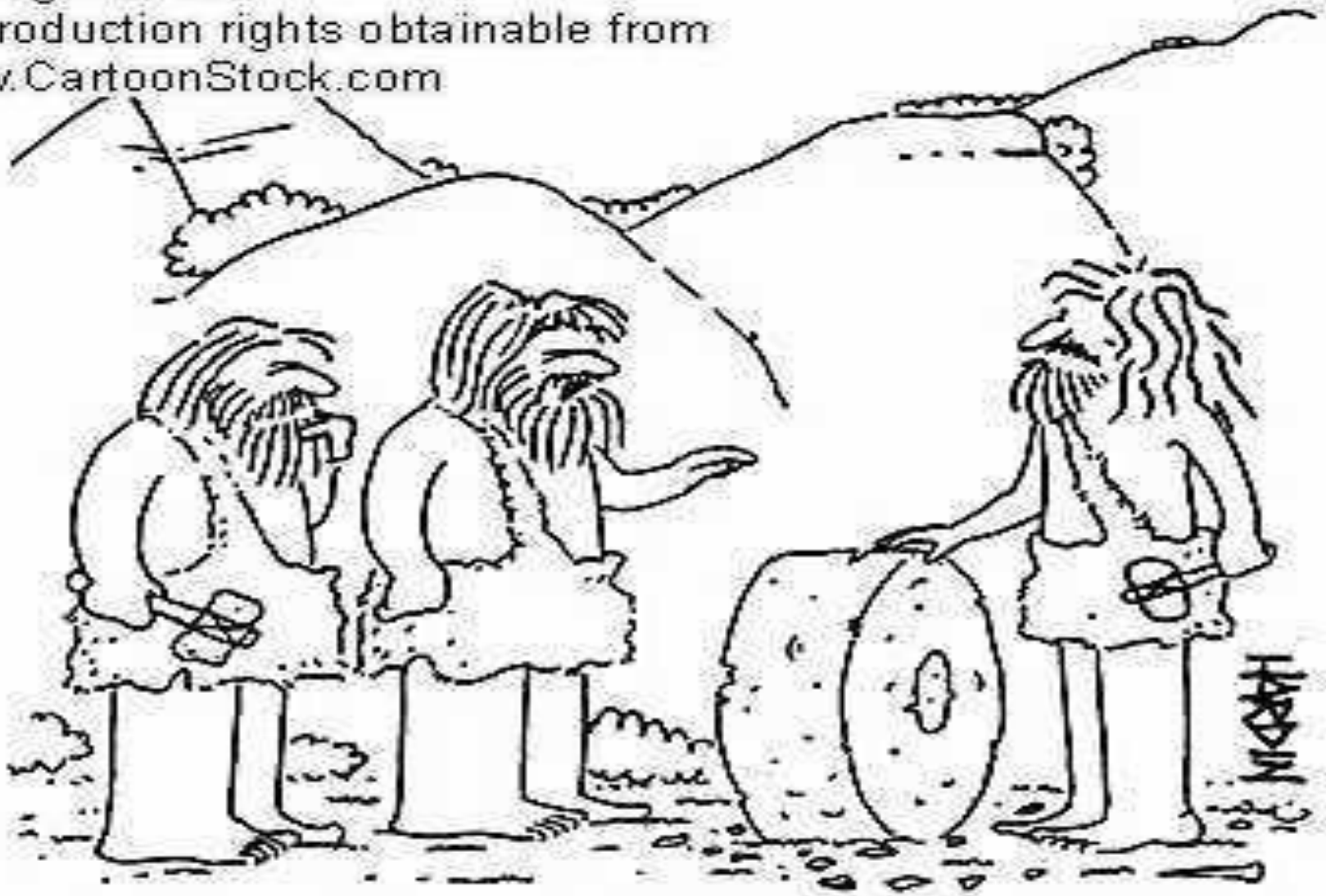
Conclusions

- **Deflation suits the many core platform**
- **Two Phase accuracy suffers**
- **Deflation with IP Preconditioning**

TU Delft


TU Delft

"IMPLEMENTING THESE CHANGES WON'T BE EASY. WE'RE PRETTY SET IN DOING THINGS THE WRONG WAY."



"This 'Wheel' thing of yours—Does it have to be round or will any shape do?"