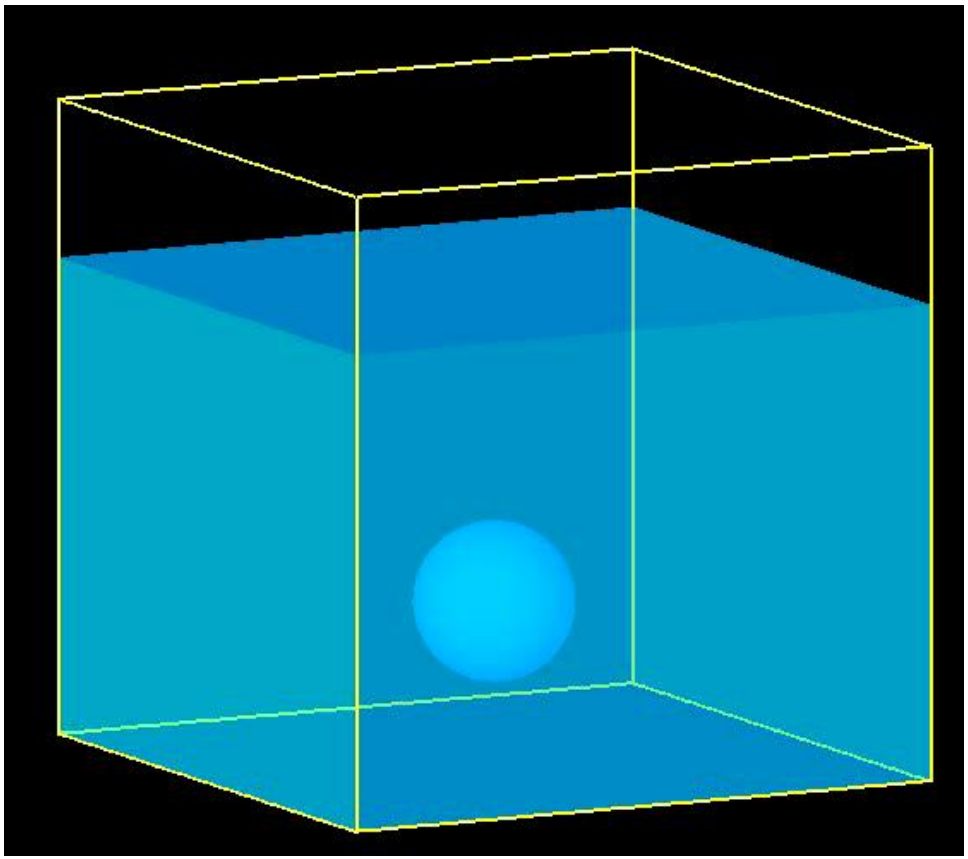
Implementation of Deflated Preconditioned Conjugate Gradient on the GPU

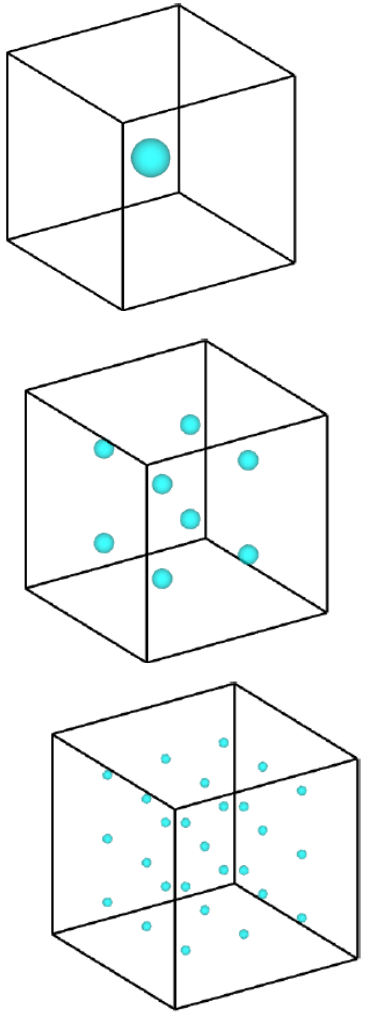
Rohit Gupta

Study Conducted under supervision of :
Prof. Dr. Ir. Kees Vuik and Ir. C.W.J. Lemmens
Delft Institute of Applied Mathematics

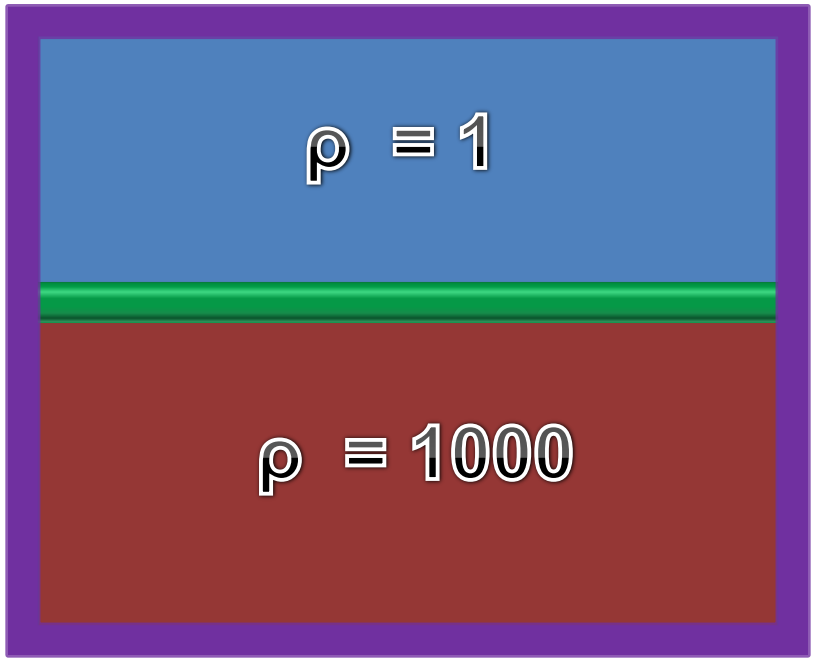
TU Delft

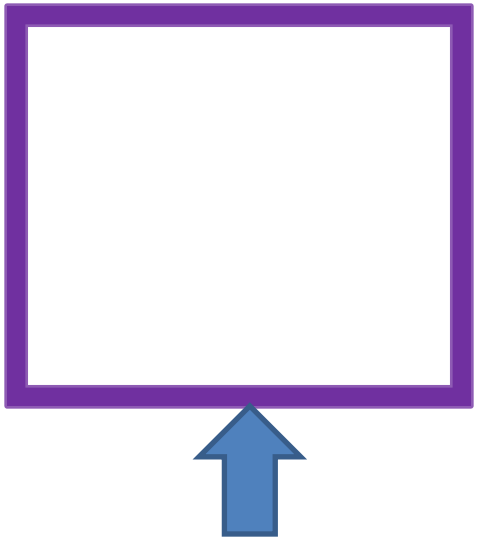
Two-Phase Fluid Flow





Model for Two Phase Computation

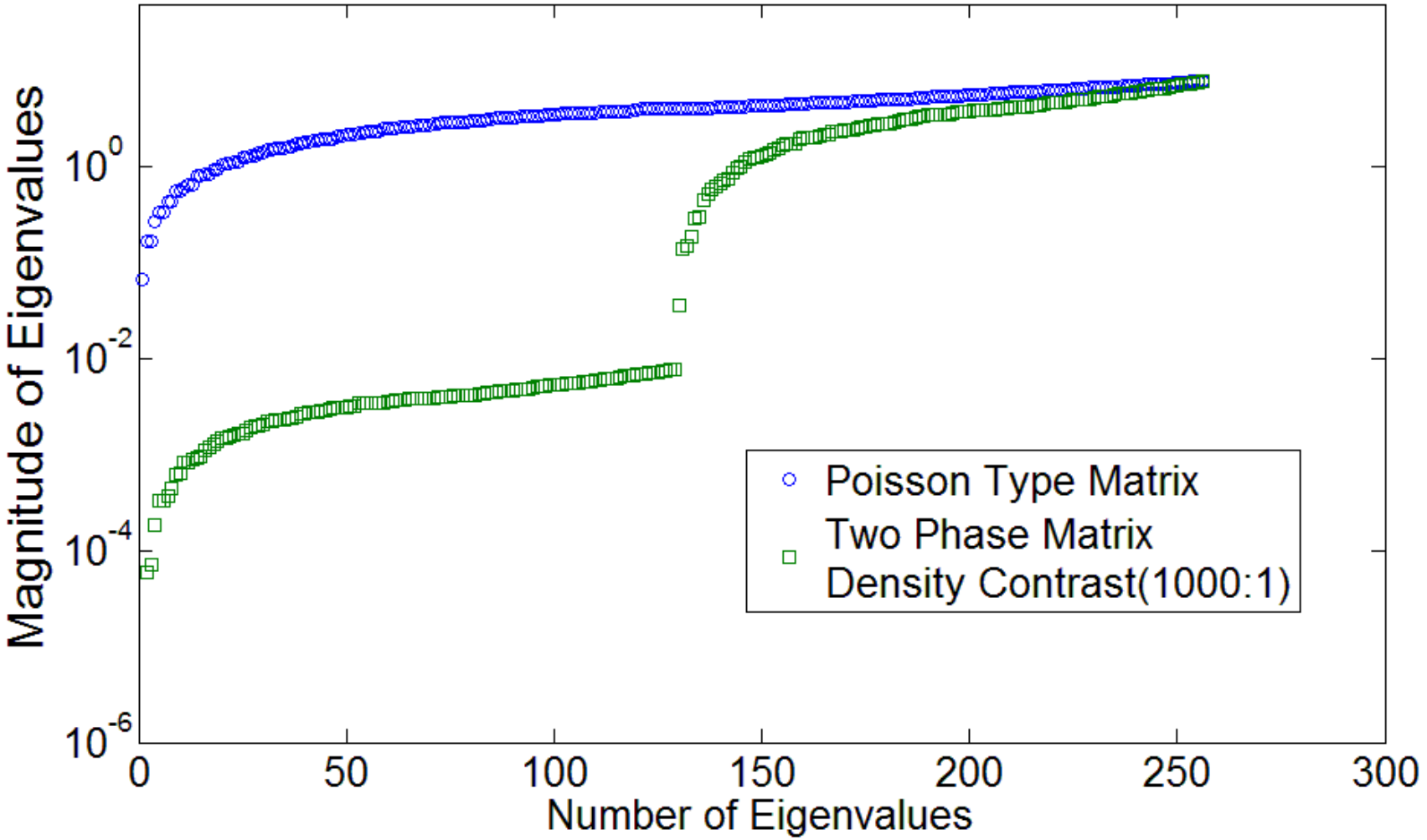




Boundary Conditions

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

TU Delft


Magnitude of Eigenvalues
10^{0}
10^{-2}
10^{-4}
10^{-6}
0
50
100
150
200
250
300
Number of Eigenvalues
Poisson Type Matrix
Two Phase Matrix
Density Contrast(1000:1)

TU Delft

Preconditioning and Deflation

$$M^{-1}Ax = M^{-1}b$$

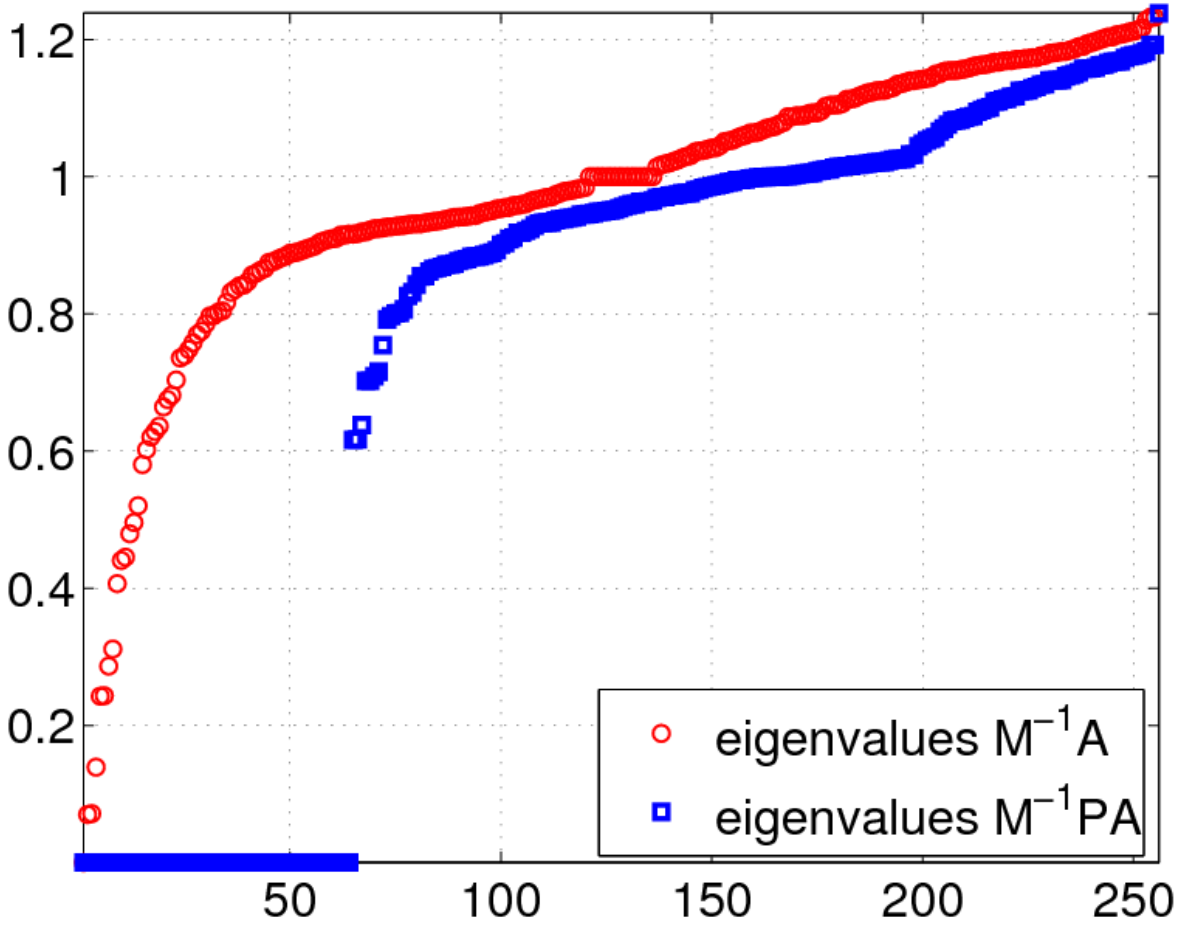
$$M^{-1}PAx = M^{-1}Pb$$

M – Preconditioning Matrix, *P – Deflation Matrix*

$$P = I - AQ; \qquad Q = ZE^{-1}Z^{T}; \qquad E = Z^{T}AZ.$$

TU Delft

Preconditioning and Deflation



TU Delft

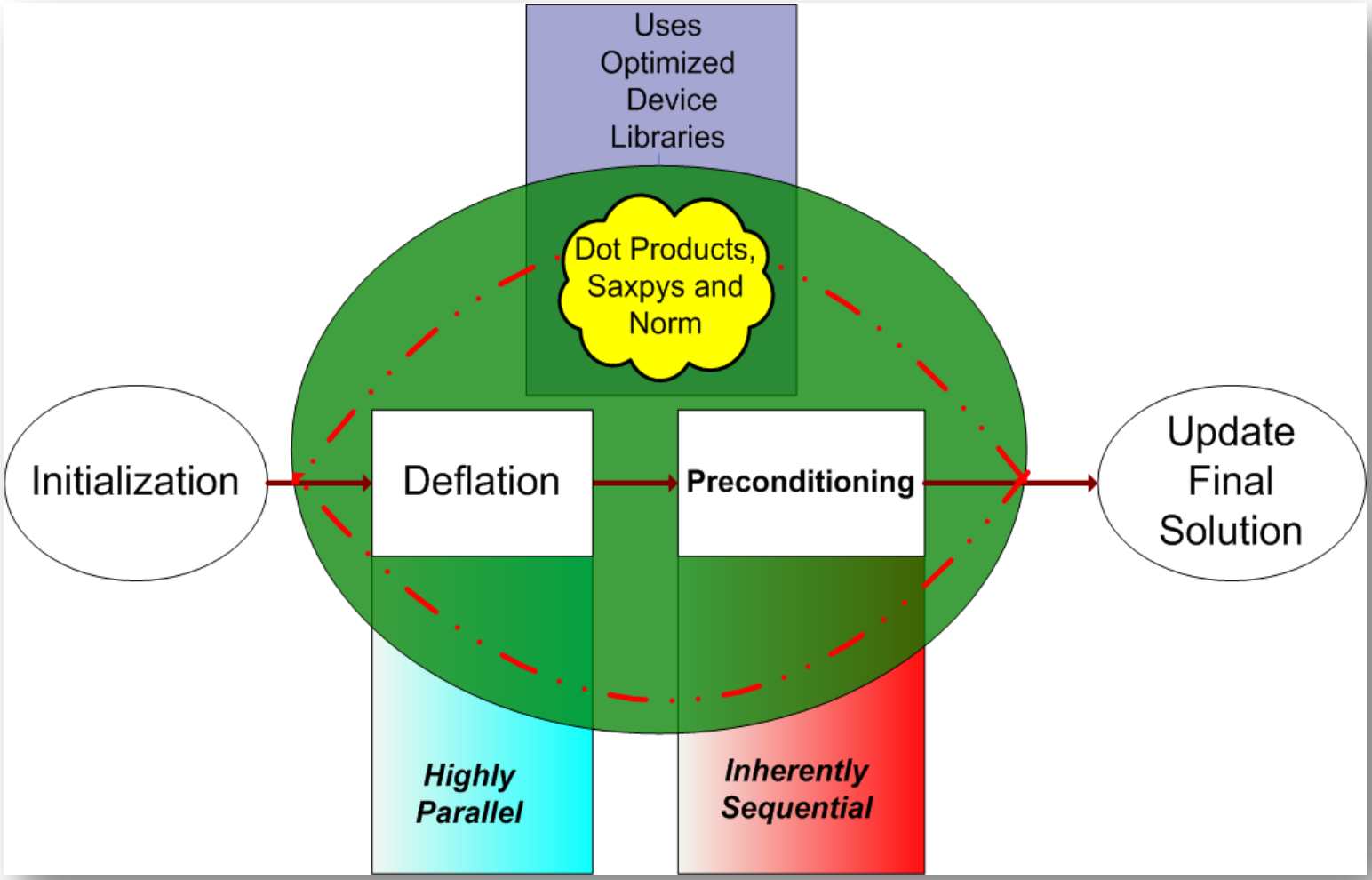
Stopping Criterion and Condition Number

$$\frac{\| b - Ax_k \|_2}{\| r_0 \|_2} < 10^{-5}$$

$$\kappa(A) := \frac{\lambda_n}{\lambda_1} \; where \lambda_n \geq \lambda_{n-1} \geq \lambda_{n-2} \geq \ldots \geq \lambda_1 \geq 0$$

TU Delft

Control Flow in the Algorithm



TU Delft

Related Work

- **SpMV**
- **Conjugate Gradient & Preconditioning**
- **Precision Approaches**

Defining Speed Up

- **Many Definitions Possible**
- **Compiler Switches on CPU**
- **Gcc, Meschach, Single Core**

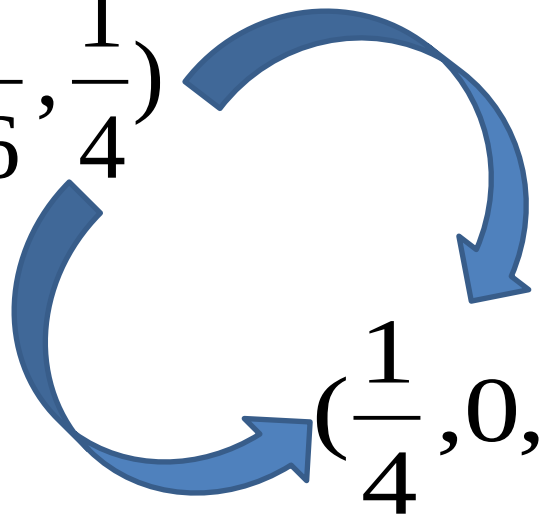
Key Optimizations

TU Delft

IP Preconditioning

$$M^{-1} = KK^T \quad \text{where} \quad K = I - LD^{-1}$$

$$\left(\frac{1}{4}, \frac{1}{16}, \frac{1}{4}, \frac{9}{8}, \frac{1}{4}, \frac{1}{16}, \frac{1}{4}\right)$$



$$\left(\frac{1}{4}, 0, \frac{1}{4}, \frac{9}{8}, \frac{1}{4}, 0, \frac{1}{4}\right)$$

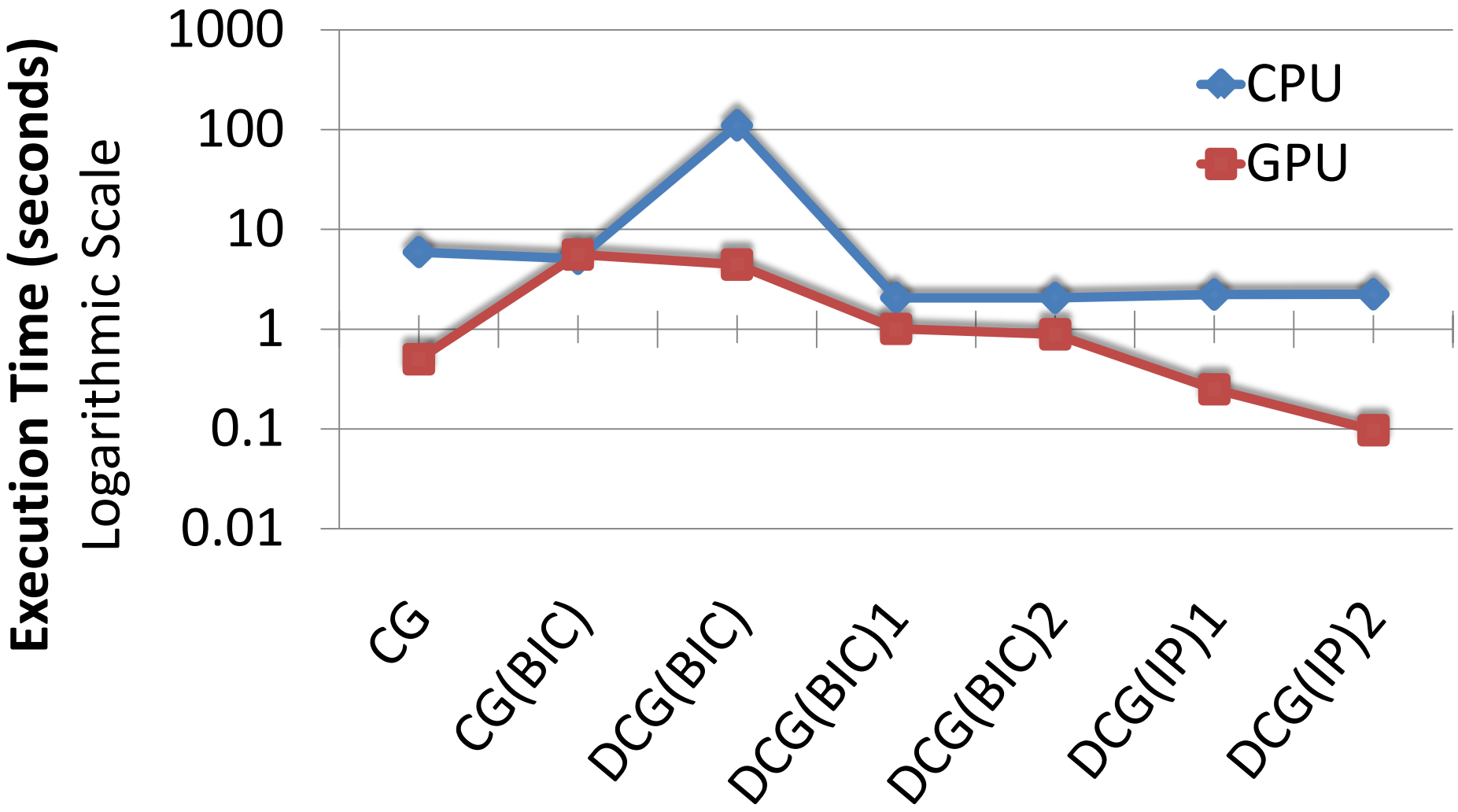
Code Versions

D CG (IP) (BIC) 1 2

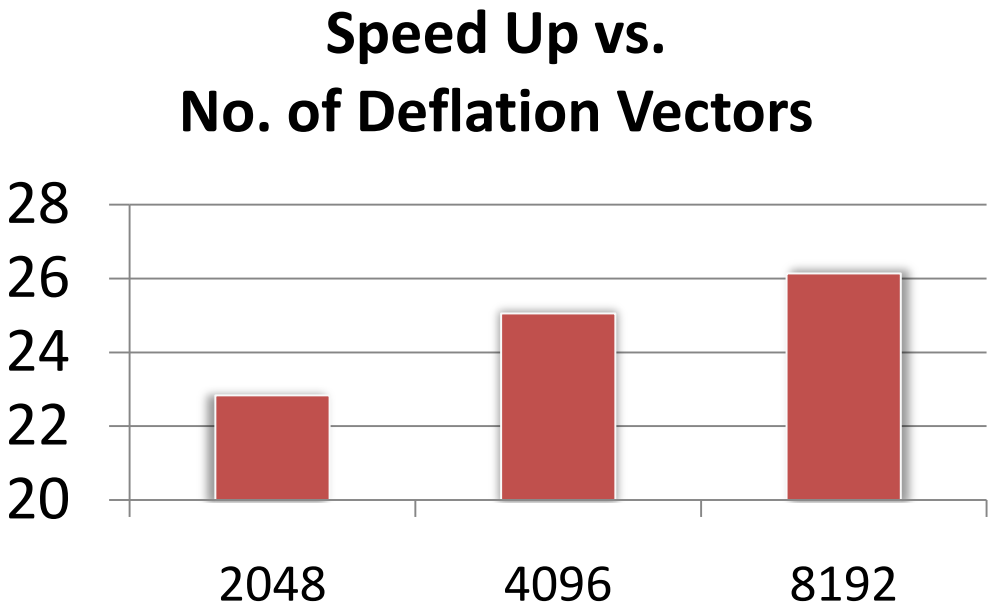
1. Optimized Storage of AZ matrix for deflation
2. Optimized BLAS using MAGMA and combined some kernels in the iteration

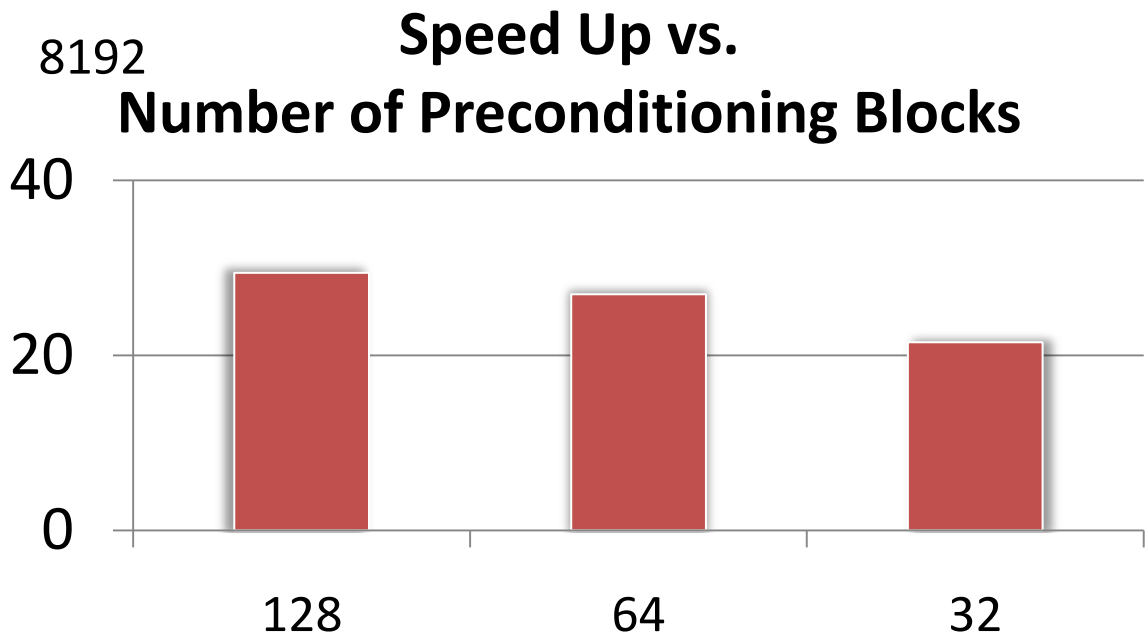
TU Delft

Execution Times across Code Versions



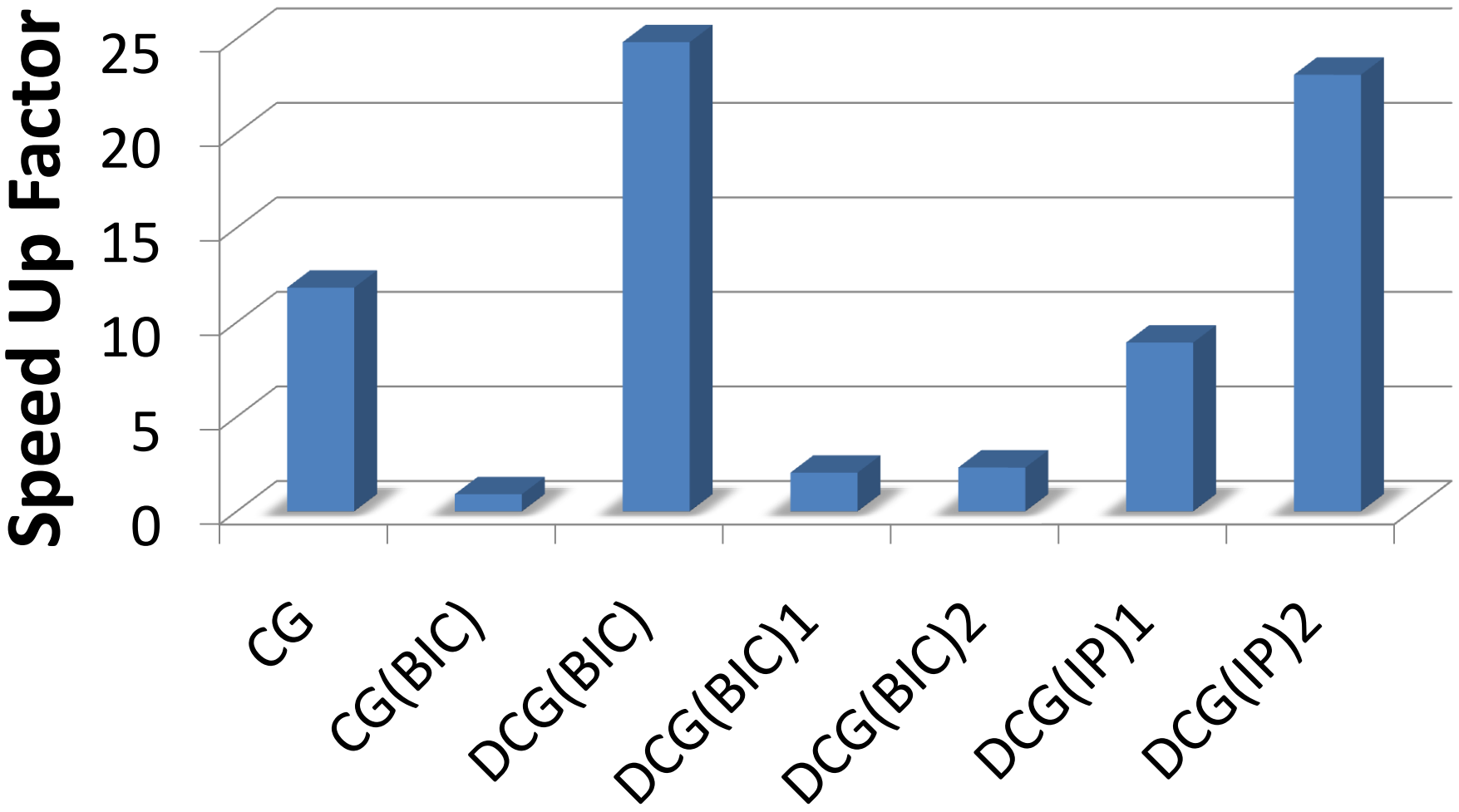
TU Delft


Speed Up vs.
No. of Deflation Vectors
28
26
24
22
20
2048
4096
8192


Speed Up vs.
Number of Preconditioning Blocks
40
20
0
128
64
32

TU Delft

Speed Up Across Code Versions



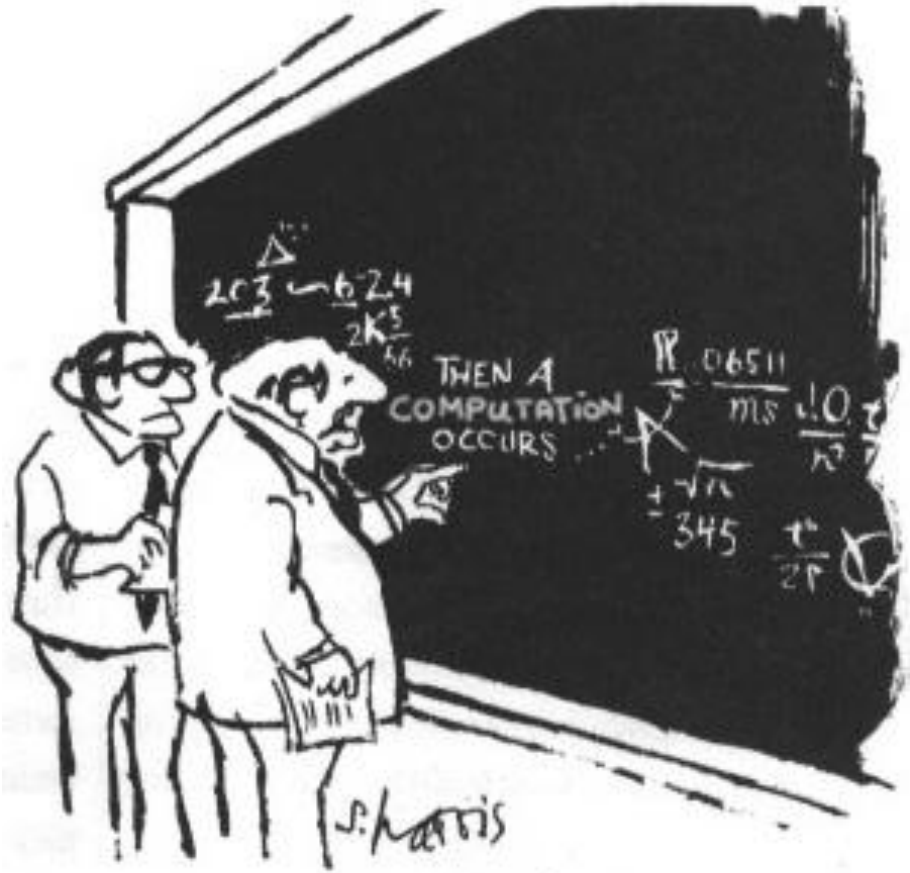
TU Delft

Analysis

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

TU Delft

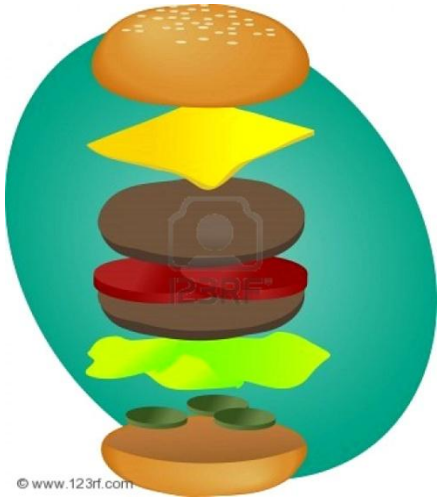
Scalability

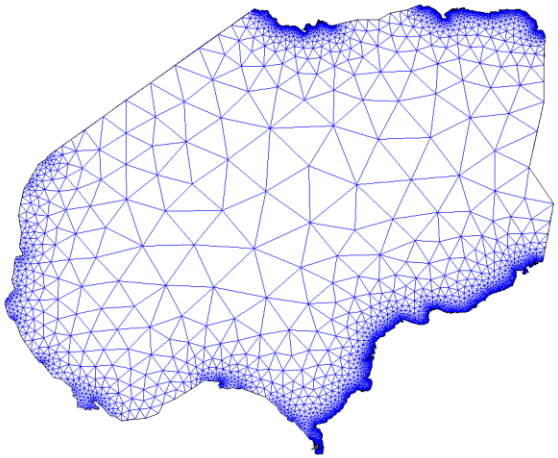


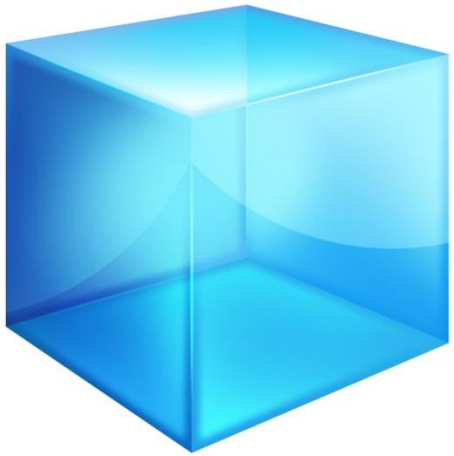
" good call using a computer here in step two."

TU Delft

Future Work





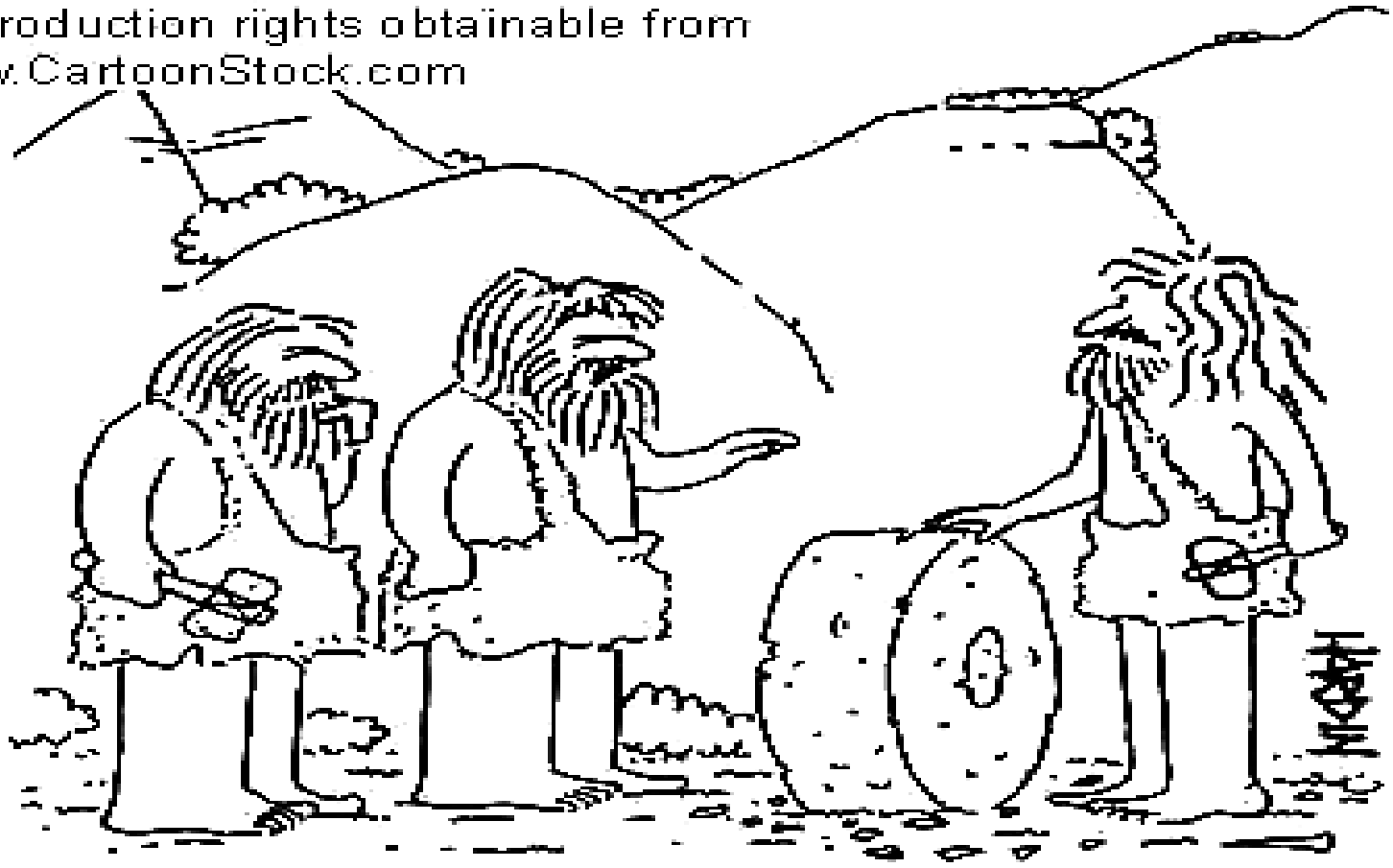


TU Delft

Conclusions

Deflation with Preconditioning

TU Delft



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