A fine-grained fully iterative ILU preconditioner for unsteady density variable Navier-Stokes equations

Monica Dessole

CERN, Geneva, Switzerland

10th June 2024

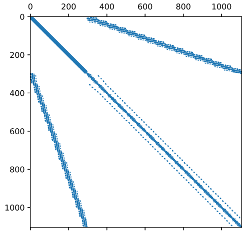
Algebraic problem

Let's focus on the solution of

$$A_n \mathbf{x}_n = \mathbf{b}_n, \quad n = 1, \dots$$

where the matrices in the sequence A_n

- have order N;
- share the same (symmetric) sparsity pattern;
- slightly differ one from another;
- focus on ILU factorizations.



Goals:

- design a general purpose preconditioning strategy;
- exploit preconditioner from previous algorithmic steps;
- efficient execution on accelerators, in particular GPUs.

Incomplete LU factorization

Incomplete LU (ILU) factorization forms a robust class of preconditioners, however they involve **two sparse triangular linear systems** at each outer Krylov iteration.
Drawbacks:

- ILU factorizations can generate significant fill-in;
- may be prone to instabilities.

Reordering techniques, such as the Reverse Cuthill-McKee (RCM) ordering, are been used to help alleviate this problem[1].
Focus:

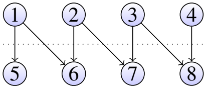
1. Improve the efficiency of triangular solver on GPUs;
2. Update LU factorization exploiting previous time steps to avoid computation from scratch.

[1] Michele Benzi. "Preconditioning Techniques for Large Linear Systems: A Survey". In: *Journal of Computational Physics* 182 (2 2002).

Parallel exact triangular solver

Taking advantage of the **sparsity**, it is possible to group together independent rows by representing the dependencies as a directed graph[2]. For example, a triangular system of size $N = 8$ can be solved in 2 algorithmic steps.

$$
\begin{pmatrix}
\times & & & & & & & \\
 & \times & & & & & & \\
 & & \times & & & & & \\
 & & & \times & & & & \\
\times & & & & \times & & & \\
\times & \times & & & & \times & & \\
 & \times & \times & & & & \times & \\
 & & \times & \times & & & & \times
\end{pmatrix}
$$

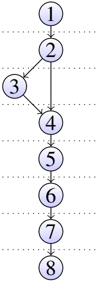


[2] Maxim Naumov. "Parallel Solution of Sparse Triangular Linear Systems in the Preconditioned Iterative Methods on the GPU". In: 2011.

Parallel exact triangular solver: RCM reordering

RCM may improve fill-in and stability, but it often erodes parallelism.

$$\begin{pmatrix}
\times & & & & & & & \\
\times & \times & & & & & & \\
 & \times & \times & & & & & \\
 & \times & \times & \times & & & & \\
 & & & \times & \times & & & \\
 & & & & \times & \times & & \\
 & & & & & \times & \times & \\
 & & & & & & \times & \times
\end{pmatrix}$$



Approximate triangular solver[3]

Consider a sparse triangular linear system $T\mathbf{x} = \mathbf{b}$, $D =$diag(T) and D_B a block diagonal approximation of T.

Jacobi(m):

$$\begin{cases} \mathbf{x}^{(k+1)} & = D^{-1}\mathbf{b} + (I - D^{-1}T)\mathbf{x}^{(k)}, \quad k = 0, \ldots, m-1 \\ \mathbf{x}^{(0)} & = D^{-1}\mathbf{b}. \end{cases} \tag{1}$$

Block-Jacobi(m):

$$\begin{cases} \mathbf{x}^{(k+1)} & = D_B^{-1}\mathbf{b} + (I - D_B^{-1}T)\mathbf{x}^{(k)}, \quad k = 0, \ldots, m-1 \\ \mathbf{x}^{(0)} & = D_B^{-1}\mathbf{b}. \end{cases} \tag{2}$$

[3] Edmond Chow et al. “Using Jacobi iterations and blocking for solving sparse triangular systems in incomplete factorization preconditioning”. In: *Journal of Parallel and Distributed Computing* (2018). ISSN: 0743-7315.

LU updates

Consider a sequence of matrices A_n, $n = 1, \ldots$, with the same sparsity pattern $\mathcal{S}$. The Iterative Thresholding Alternating LU[4] is an updating strategy.

ITALU(m)

Given $A_n \approx L_0 U_0$, compute $A_{n+1} \approx LU$

for $k = 0, ..., m-1$ **do**
- Compute $R_k = A_n - L_k U_k$
- Compute $X_U = \text{triu}(L_k^{-1} R_k)$ ▹ Direct LU solver
- Apply dropping to X_U
- $U_{k+1} = U_k + X_U$ ▹ U update
- Compute $X_L = \text{tril}(R_k U_{k+1}^{-1})$ ▹ Direct LU solver
- Apply dropping to X_L
- $L_{k+1} = L_k + X_L$ ▹ L update

[4] Caterina Calgaro, Jean-Paul Chehab, and Yousef Saad. "Incremental incomplete LU factorizations with applications". In: *Numerical Linear Algebra with Applications* 17 (5 2010).

LU updates

Consider a sequence of matrices A_n, $n = 1, \ldots$, with the same sparsity pattern $\mathcal{S}$. The Iterative Thresholding Alternating LU[4] is an updating strategy.

ITALU(m)

Given $A_n \approx L_0 U_0$, compute $A_{n+1} \approx LU$

for $k = 0, \ldots, m-1$ **do**

- Compute $R_k = A_n - L_k U_k$
- Compute $X_U = \mathrm{triu}(L_k^{-1} R_k)$ ▹ Direct LU solver
- Apply dropping to X_U
- $U_{k+1} = U_k + X_U$ ▹ U update
- Compute $X_L = \mathrm{tril}(R_k U_{k+1}^{-1})$ ▹ Direct LU solver
- Apply dropping to X_L
- $L_{k+1} = L_k + X_L$ ▹ L update

[4] Caterina Calgaro, Jean-Paul Chehab, and Yousef Saad. "Incremental incomplete LU factorizations with applications". In: *Numerical Linear Algebra with Applications* 17 (5 2010).

LU updates

Consider a sequence of matrices A_n, $n = 1, \dots,$ with the same sparsity pattern $\mathcal{S}$. The Iterative Thresholding Alternating LU[4] is an updating strategy.

ITALU(m)

Given $A_n \approx L_0 U_0$, compute $A_{n+1} \approx LU$

for $k = 0, ..., m-1$ **do**

- Compute $R_k = A_n - L_k U_k$
- Compute $X_U = \mathrm{triu}(L_k^{-1} R_k)$ ▷ Direct LU solver
- Apply dropping to X_U
- $U_{k+1} = U_k + X_U$ ▷ U update
- Compute $X_L = \mathrm{tril}(R_k U_{k+1}^{-1})$ ▷ Direct LU solver
- Apply dropping to X_L
- $L_{k+1} = L_k + X_L$ ▷ L update

[4] Caterina Calgaro, Jean-Paul Chehab, and Yousef Saad. “Incremental incomplete LU factorizations with applications”. In: *Numerical Linear Algebra with Applications* 17 (5 2010).

LU updates

Theorem

Sequences $\{L_k\}$, $\{U_k\}$ converge respectively to the exact L, U factors, in no more then N steps.

However, the correction matrices are computed as the solution of

$$\begin{aligned} L_k X_U &= R_k, \quad X_U = \text{triu}(X_U), \\ U_k X_L &= R_k, \quad X_L = \text{tril}(X_L). \end{aligned} \tag{3}$$

Inefficient on parallel machines, $2N$ sparse triangular systems **at each iteration**.

Fully iterative LU updates

Applying the Jacobi method

$$X_L^{(j)} = X_L^{(0)} + (I_N - D_k^{-1} L_k) X_L^{(j-1)} \odot S, \quad X_L^{(0)} = (D_k)^{-1} \mathrm{tril}(R_k \odot S), \tag{4}$$

where $D_k = \mathrm{diag}(U_k)$, and, since $\mathrm{diag}(U_k) = I_N$, we set

$$X_U^{(j)} = X_U^{(0)} + (I_N - U_k) X_U^{(j-1)} \odot S, \quad X_U^{(0)} = \mathrm{triu}(R_k \odot S). \tag{5}$$

Dropping: component-wise product with $S = (s_{ij})$, $s_{ij} = 1$ if $(i, j) \in \mathcal{S}$.

SITALU(m, j) (Scalable ITALU)

Given $A_n \approx L_0 U_0$, compute $A_{n+1} \approx LU$

for $k = 0, \ldots, m-1$ **do**
  Compute $R_k = (A_n - L_k U_k)$
  Compute $X_L = X_L^{(j)}$ using (4) ▷ Iterative LU solver
  $L_{k+1} = L_k + X_L$ ▷ L update
  Compute $X_U = X_U^{(j)}$ using (5) ▷ Iterative LU solver
  $U_{k+1} = U_k + X_U$ ▷ U update

Navier-Stokes equations with variable density

Let $\Omega \subset \mathbb{R}^2$, the density dependent Navier-Stokes system on $(0, T] \times \Omega$ is

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{6}$$

$$\partial_t(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) - \mu \Delta \mathbf{u} + \nabla p = \rho \mathbf{f}, \tag{7}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{8}$$

where the unknowns are

- $\mathbf{u} = \mathbf{u}(t, \mathbf{x})$ is the velocity vector field,
- $p = p(t, \mathbf{x})$ is the pressure field,
- $\rho = \rho(t, \mathbf{x})$ is the density field.

Rayleigh-Taylor instability[5]

Play/Pause

A heavy fluid fluid (density $\rho_{\max}$, in yellow) is superposed to a light fluid (density $\rho_{\min}$, in blue) under the action of a gravitational field.

The numerical difficulty essentially depends on:

- the Reynolds number Re;
- the Atwood number

$$At = \frac{\rho_{\max} - \rho_{\min}}{\rho_{\max} + \rho_{\min}}.$$

[5] M. Dessole and F. Marcuzzi. "Fully iterative ILU preconditioning of the unsteady Navier–Stokes equations for GPGPU". In: *Computers & Mathematics with Applications* 77.4 (2019), pp. 907–927.

The numerical scheme

Use Strang time splitting[6] and solve with a second order hybrid FE-FV scheme[7]:

- $\mathbb{P}^1$ Finite Volume approximation of the transport equation

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0.$$

- $\mathbb{P}^2 - \mathbb{P}^1$ Finite Element approximation of NS

$$\partial_t(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) - \mu \Delta \mathbf{u} + \nabla p = \rho \mathbf{f},$$
$$\nabla \cdot \mathbf{u} = 0.$$

[6] Gilbert Strang. “On the Construction and Comparison of Difference Schemes”. In: *SIAM Journal on Numerical Analysis* 5 (3 Sept. 1968).

[7] Caterina Calgaro, Emmanuel Creusè, and Thierry Goudon. “An hybrid finite volume–finite element method for variable density incompressible flows”. In: *Journal of Computational Physics* 227 (9 2008).

NS: algebraic problem

Using a projection method, at each time step the linear problem is:

$$A_{n+1} U_i^{n+1} = F_{u_i}^{n+1}, \quad i = 1, 2, \tag{9}$$

$$L_p \Phi_{n+1} = F_{\Phi}^{n+1}, \tag{10}$$

$$M_p P_{n+1} = F_p^{n+1} \tag{11}$$

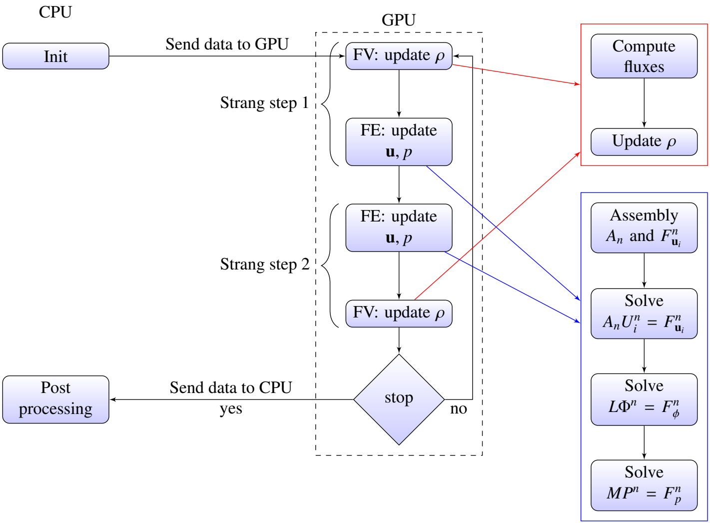
where L_p, M_p are the stiffness and mass $\mathbb{P}^1$-matrices and

$$A_{n+1} = \frac{3}{2\Delta t} M_u(\rho^*) + \frac{1}{Re} L_u + NL(\bar{\mathbf{u}}^{n+1}, \rho^*). \tag{12}$$

where

- M_u is the mass matrix, depending on the current density fiend ρ^*;
- L_u is the stiffness matrix
- NL is the nonlinear term matrix, evaluated on a second order extrapolation of the velocity field $\bar{\mathbf{u}}^{n+1}$ and the current density field ρ^*.

Single node GPU-offloading



Experimental framework

The linear systems are solved with GMRES(30).

Abbr.	LU update	LU solver
ILU(0)	No	Direct exact solver
SITALU(m, j)	Yes	Direct exact solver
SITALU(m, j)+Jacobi(k)	Yes	Jacobi solver
SITALU(m, j)+block-Jacobi(k)	Yes	block-Jacobi solver

On what follows $j = 1$. Initialization is performed with ILU(0)[8].
Comparison metrics:

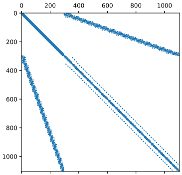
- the number of iterations for convergence (maxit$= 3000$);
- runtime of the solve phase (preconditioner computation/updating+GMRES).

Hardware: **gpu01**, NVidia GeForce GTX1060 GPU with 2560 CUDA cores.

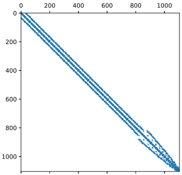
[8] E. Chow and A. Patel. "Fine-Grained Parallel Incomplete LU Factorization". In: *SIAM Journal on Scientific Computing* 37.2 (2015), pp. C169–C193.

A closer look to block-Jacobi

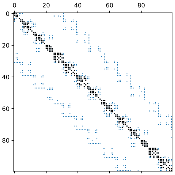
Fine-grained block approximation:



(a) Lexicographic ordering.



(b) RCM ordering.



(c) Diagonal blocks, close up.

Trade-off: better parallelism, less efficiency.

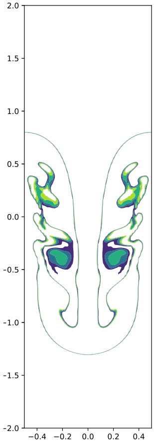
Interplay between Reynolds and Atwood numbers

Rayleigh-Taylor instability is investigated in different settings, i.e. different values of the Reynolds number *Re* and the Atwood number *At*.

- Low, moderate and high Reynolds values, i.e. $Re = 10^{-2}, 1000, 20000$.
- Moderate, high and very high Atwood values, i.e. $At = 0.5, 0.9, 0.98$. Recall that

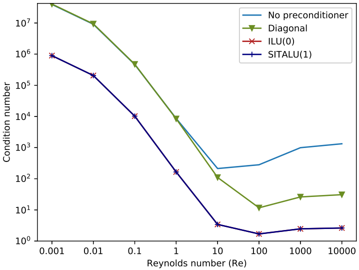
$$At = \frac{\rho_{\max} - \rho_{\min}}{\rho_{\max} + \rho_{\min}}.$$

For simplicity, we indicate also the density ratio $\rho_{\max}/\rho_{\min} = 3, 19, 100$.

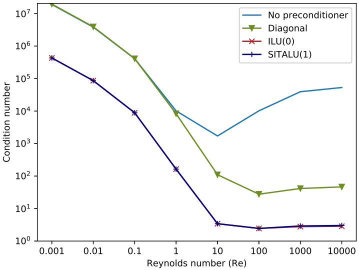


Condition number

Finally, the SITALU(1) preconditioner performs as well as ILU(0), meaning that it achieves its optimal behaviour.



(a) Density ratio 3.



(b) Density ratio 19.

Figure: Condition number in function of the Reynolds number at different density ratio.

Density ratio 3 ($At = 0.5$), $Re = 20000$

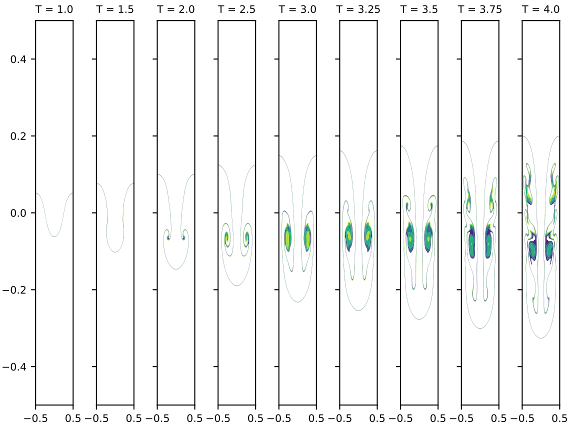
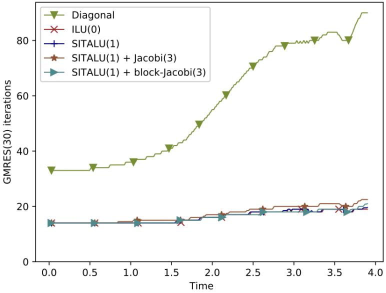
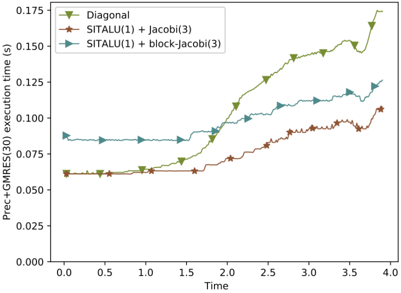


Figure: Evolution of the interface in time, $N = 88981$

Density ratio 3 ($At = 0.5$), $Re = 20000$



(a) Iteration number



(b) Execution times

Figure: N=88981

Density ratio 3 ($At = 0.5$), $Re = 20000$

Preconditioner	k	Avg. iter. number	Avg. sol. Time (s)
ILU(0)	2	17.0	1.1701
	3	17.0	1.1727
	4	17.1	1.1789
SITALU(1)	2	17.1	0.3850
	3	17.2	0.3890
	4	17.5	0.3952
SITALU(1)+Jacobi(3)	2	18.3	0.0770
	3	18.4	0.0769
	4	18.7	0.0776

Table: Average values, the preconditioner is reused for k iterations.

Density ratio 100 ($At = 0.98$), $Re = 10^{-2}$

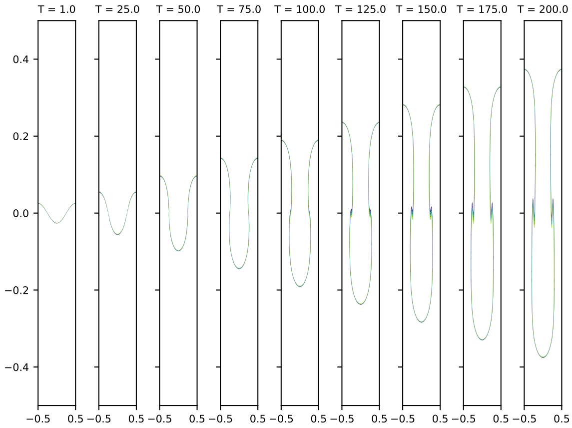
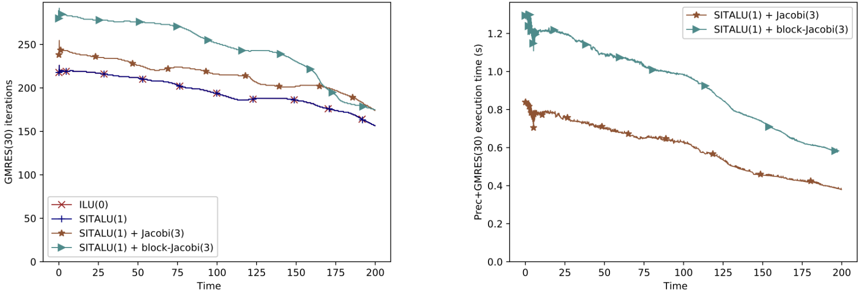


Figure: Evolution of the interface in time, $N = 29341$

Density ratio 100 ($At = 0.98$), $Re = 10^{-2}$



(a) Iteration number

(b) Execution times

Figure: $N = 29341$

Density ratio 100 ($At = 0.98$), $Re = 10^{-2}$

Preconditioner	k	Avg. iter. number	Avg. sol. Time (s)
ILU(0)	2	216.30	6.1322
	3	216.33	6.1339
	4	216.38	6.1344
SITALU(1)	2	216.30	3.3840
	3	216.33	3.3849
	4	216.38	3.3949
SITALU(1)+Jacobi(3)	2	239.64	0.6730
	3	239.65	0.6733
	4	239.40	0.6739

Table: Average values, the preconditioner is reused for k iterations.

Density ratio 19 ($At = 0.9$), $Re = 1000$

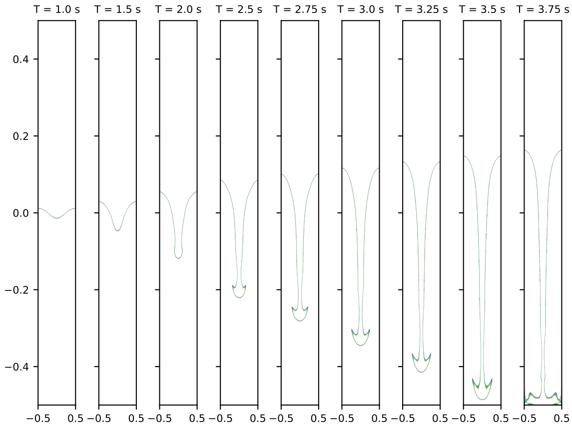
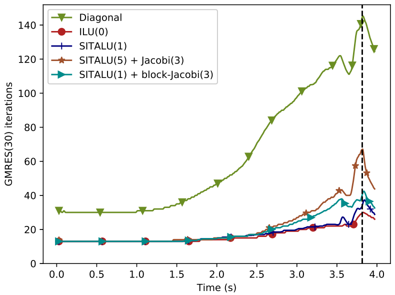
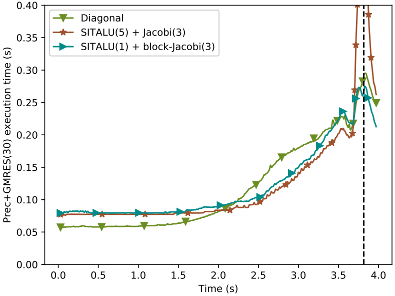


Figure: Evolution of the interface in time, $N = 88981$

Density ratio 19 ($At = 0.9$), $Re = 1000$



(a) Iteration number



(b) Execution times

Figure: $N = 88981$

Conclusions

- A **fully iterative ILU preconditioner** performs 80% better on a GPU.
- SITALU algorithm turns out to **efficient and inexpensive**, 2 to 3 times faster then recomputing the preconditioner from scratch.
- Preconditioner reuse has no global benefit to the execution times, and sometimes it causes the preconditioner to fail.

Further directions:

- The **number of sweeps could be adjusted dynamically**: the number of Jacobi sweeps *m* could be tuned on the residual norm reduction.
- Generalization to other preconditioning techniques should be investigated.

Thank you for listening!

References I

[1] Michele Benzi. "Preconditioning Techniques for Large Linear Systems: A Survey". In: *Journal of Computational Physics* 182 (2 2002).

[2] Caterina Calgaro, Jean-Paul Chehab, and Yousef Saad. "Incremental incomplete LU factorizations with applications". In: *Numerical Linear Algebra with Applications* 17 (5 2010).

[3] Caterina Calgaro, Emmanuel Creusè, and Thierry Goudon. "An hybrid finite volume–finite element method for variable density incompressible flows". In: *Journal of Computational Physics* 227 (9 2008).

[4] E. Chow and A. Patel. "Fine-Grained Parallel Incomplete LU Factorization". In: *SIAM Journal on Scientific Computing* 37.2 (2015), pp. C169–C193.

[5] Edmond Chow et al. "Using Jacobi iterations and blocking for solving sparse triangular systems in incomplete factorization preconditioning". In: *Journal of Parallel and Distributed Computing* (2018). ISSN: 0743-7315.

References II

[6] M. Dessole and F. Marcuzzi. "Fully iterative ILU preconditioning of the unsteady Navier–Stokes equations for GPGPU". In: *Computers & Mathematics with Applications* 77.4 (2019), pp. 907–927.

[7] Maxim Naumov. "Parallel Solution of Sparse Triangular Linear Systems in the Preconditioned Iterative Methods on the GPU". In: 2011.

[8] Gilbert Strang. "On the Construction and Comparison of Difference Schemes". In: *SIAM Journal on Numerical Analysis* 5 (3 Sept. 1968).