Hierarchical adaptive low-rank format with applications to discretized PDEs

Stefano Massei

Joint work with D. Kressner (EPFL) and L. Robol (University of Pisa)

EINDHOVEN UNIVERSITY OF TECHNOLOGY

s.massei@tue.nl

Conference on Fast Direct Solvers, 23 October 2021

Sylvester equations coming from PDEs

We consider time-dependent PDEs of the form

$$\begin{cases} \frac{\partial u}{\partial t} = \mathcal{L}u + f(u, \nabla u) & t \in [0, T_{\max}] \\ u(x,y,0) = u_0(x,y) & (x,y) \in \Omega := [a,b] \times [c,d] \\ \text{B.C.} & (x,y) \in \partial\Omega \text{ and } t > 0 \end{cases},$$

where $\mathcal{L} = \mathcal{L}_x + \mathcal{L}_y$ is elliptic with a Kronecker sum structured discretization [1,2] $L := I \otimes A + B \otimes I$, and f is nonlinear.

Applying the IMEX Euler approach, where $\mathcal{L}$ is treated implicitly, yields [3]

$$(I - \Delta t \cdot L)u_{t+1} = u_t + \Delta t(f(u_t, \nabla u_t) + B.C.),$$

i.e., an iterative scheme where at each step we need to solve a Sylvester equation.

Challenge: Can we go large-scale?

[1] Townsend, Olver. *The automatic solution of partial differential equations using a global spectral method. Journal of Computational Physics, 2015.*
[2] Palitta, Simoncini. *Matrix-equation-based strategies for convection–diffusion equations*. BIT, 2016.
[3] D'Autilia, Sgura, Simoncini. *Matrix-oriented discretization methods for reaction-diffusion PDEs: Comparisons and applications*. Computers & Mathematics with Applications, 2020.

Main topic of the talk

We have an integration scheme that requires solving a sequence of Sylvester eqns:

$$AX_t + X_tB = C_t.$$

Ideal situation: X_t is low-rank $\forall t$ $\rightsquigarrow$ Efficient storage and computation of X_t.

- When the solution $u(\cdot, \cdot, t)$ is smooth, X_t is (numerically) low-rank,
- The presence of isolated singularities makes X_t only locally low-rank.
- Singularities that move during the time evolution $\rightsquigarrow$ time-dependent local structure.

Main topic of the talk

We have an integration scheme that requires solving a sequence of Sylvester eqns:

$$AX_t + X_t B = C_t.$$

Ideal situation: X_t is low-rank $\forall t$ $\rightsquigarrow$ Efficient storage and computation of X_t.

- When the solution $u(\cdot, \cdot, t)$ is smooth, X_t is (numerically) low-rank,
- The presence of isolated singularities makes X_t only locally low-rank.
- Singularities that move during the time evolution $\rightsquigarrow$ time-dependent local structure.

Question: Can we fully exploit local and time-dependent structures in the time integration?

Burgers equation

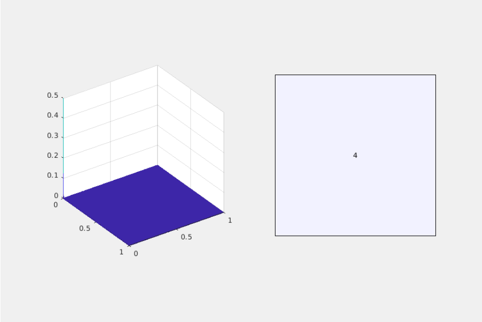
As running example, consider the 2D Burgers equation:

$$\begin{cases} \frac{\partial u}{\partial t} = 10^{-3}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - u \cdot \left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) \\ u(x, y, t) = \frac{1}{1+\exp(10^3(x+y-t)/2)} & t = 0 \text{ or } (x, y) \in \partial\Omega \end{cases}$$

Burgers equation

As running example, consider the 2D Burgers equation:

$$\begin{cases} \frac{\partial u}{\partial t} = 10^{-3}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - u \cdot \left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) & \\ u(x,y,t) = \frac{1}{1+\exp(10^3(x+y-t)/2)} & t = 0 \text{ or } (x,y) \in \partial\Omega \end{cases}$$



Blue blocks: Full rank submatrices, **Grey blocks**: Low-rank submatrices

The time marching scheme

procedure BURGERS_IMEX(n, Δt, $T_{\max}$)
  $A \leftarrow \frac{1}{2} I - \Delta t A_n$
  $(X_0)_{ij} \leftarrow u(x_i, y_j, 0)$
  for $t = 0, 1, \ldots, T_{\max}$ **do**
    $F \leftarrow X_t \circ \left[D_n X_t + X_t D_n^T\right]$
    $C_t \leftarrow X_t + \Delta t \cdot F +$ low-rank
    Solve $A X_{t+1} + X_{t+1} A = C_t$
  end for
end procedure

$$
\begin{aligned}
A_n &= n \times n \text{ discretized second derivative operator,} \\
D_n &= n \times n \text{ discretized first derivative operator.}
\end{aligned}
$$

The time marching scheme

```
procedure BURGERS_IMEX(n, Δt, T_max)
    A ← (1/2) I − Δt A_n
    (X_0)_ij ← u(x_i, y_j, 0)
    for t = 0, 1, . . . , T_max do
        F ← X_t ∘ [D_n X_t + X_t D_n^T]
        C_t ← X_t + Δt · F + low-rank
        Solve A X_{t+1} + X_{t+1} A = C_t
    end for
end procedure
```

To do:

- Construct a structured representation of X_0
- Construct a structured representation of the rhs C_t
- Efficiently solve the matrix equation

Finding a low-rank block representation

Given $M \in \mathbb{R}^{n\times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$$\texttt{LRA}(M, \epsilon, \mathrm{maxrank}) \begin{cases} \widetilde{M} \text{ such that } \|M - \widetilde{M}\| \leq \epsilon,\ \mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}\ ✓ \\ \text{failure}\ ✗ \end{cases}$$

Example: $\texttt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

Finding a low-rank block representation

Given $M \in \mathbb{R}^{n \times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ →
- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

Example: $\texttt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

$M =$ [LRA]

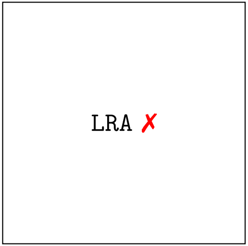
Finding a low-rank block representation

Given $M \in \mathbb{R}^{n \times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ →
- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

Example: $\texttt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

$M =$



Finding a low-rank block representation

Given $M \in \mathbb{R}^{n \times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ →

- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

Example: $\texttt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

$M =$

LRA	LRA
LRA	LRA

Finding a low-rank block representation

Given $M \in \mathbb{R}^{n \times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ →

- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

Example: $\texttt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

$M =$

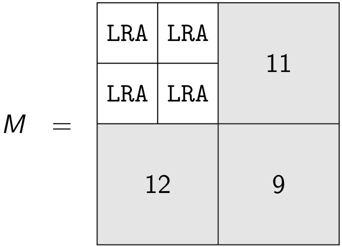
LRA ✗	LRA ✓
LRA ✓	LRA ✓

Finding a low-rank block representation

Given $M \in \mathbb{R}^{n\times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\mathtt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\mathtt{LRA}(M, \epsilon, \mathrm{maxrank})$ →
- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

Example: $\mathtt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

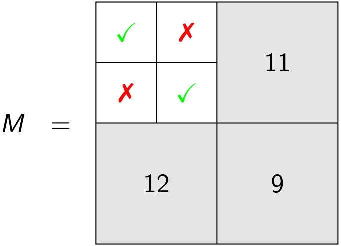


Finding a low-rank block representation

Given $M \in \mathbb{R}^{n \times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\mathtt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\mathtt{LRA}(M, \epsilon, \mathrm{maxrank})$ →
- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

Example: $\mathtt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.



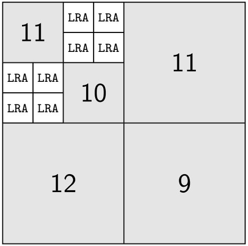
Finding a low-rank block representation

Given $M \in \mathbb{R}^{n \times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\mathtt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\mathtt{LRA}(M, \epsilon, \mathrm{maxrank})$ →

- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

Example: $\mathtt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

$M =$ 

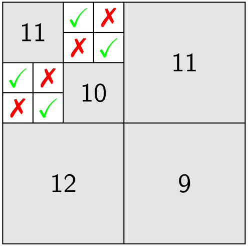
Finding a low-rank block representation

Given $M \in \mathbb{R}^{n \times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\texttt{LRA}(M, \epsilon, \mathrm{maxrank})$ →

- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

Example: $\texttt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

$M =$ 

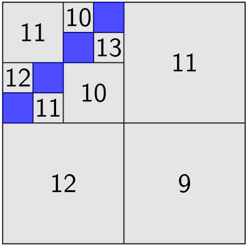
Finding a low-rank block representation

Given $M \in \mathbb{R}^{n \times n}, \epsilon > 0$, $\mathrm{maxrank} \in \mathbb{N}$ and let $\mathtt{LRA}(M, \epsilon, \mathrm{maxrank})$ be such that

$\mathtt{LRA}(M, \epsilon, \mathrm{maxrank})$ →
- $\widetilde{M}$ such that $\|M - \widetilde{M}\| \leq \epsilon$, $\mathrm{rank}(\widetilde{M}) \leq \mathrm{maxrank}$ ✓
- failure ✗

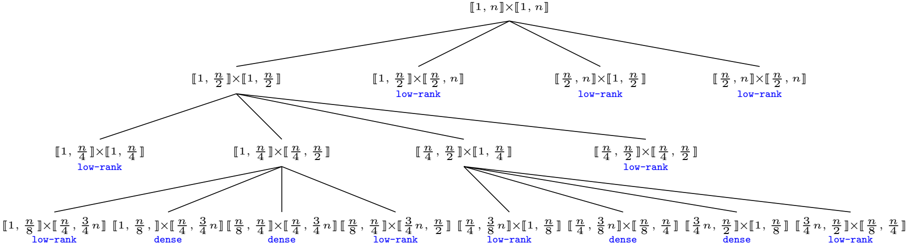
Example: $\mathtt{LRA} = \mathrm{maxrank}$ iterations of the *Adaptive Cross Approximation* algorithm.

$M =$



Hierarchically Adaptive Low-Rank matrices (HALR)

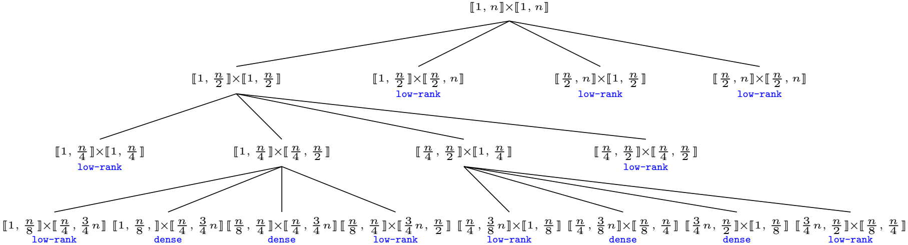
We can associate with M the quad-tree cluster $\mathcal{T}$ of the form:



Def: $M \in \mathbb{R}^{n\times n}$ is $(\mathcal{T}, r)$-HALR if its submatrices corresponding to the `low-rank` leaves of $\mathcal{T}$ have rank $\leq r$.

Hierarchically Adaptive Low-Rank matrices (HALR)

We can associate with M the quad-tree cluster $\mathcal{T}$ of the form:



Def: $M \in \mathbb{R}^{n \times n}$ is $(\mathcal{T}, r)$-HALR if its submatrices corresponding to the `low-rank` leaves of $\mathcal{T}$ have rank $\leq r$.

What we can do with the HALR format:

- Matrix operations between HALR matrices with different partitionings
- Complexity is log/log^2-proportional to the storage cost of the outcome
- Adjust/Refine the cluster (in the spirit of the construction algorithm)

Solving Sylvester equations

We have well established techniques for solving $\boxed{AX + XB = C}$ when:

- C is dense $\rightsquigarrow$ Bartels & Stewart [4] or Hessenberg-Schur [5] algorithms
- C is low-rank $\rightsquigarrow$ Krylov projection methods [6,7] or ADI [8,9]

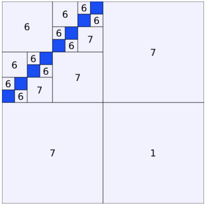
[4] Bartels, Stewart. *Algorithm 432: The solution of the matrix equation $AX - XB = C$*, Commun. ACM, 1972.
[5] Golub, Nash, Van Loan. *Hessenberg–Schur method for the problem $AX + XB = C$*, IEEE Trans. Automat. Control, 1979.
[6] Hu, Reichel. *Krylov-subspace methods for the Sylvester equation*, Libear Algebra Appl., 1992.
[7] Simoncini. *A new iterative method for solving large-scale Lyapunov matrix equations*, SISC, 2007.
[8] Wachpress. *Solution of Lyapunov equations by ADI iteration*, Comput. Math. Appl., 1991.
[9] Benner, Li, Truhar. *On the ADI method for Sylvester equations*, J. Comp. and App. Math., 2009.

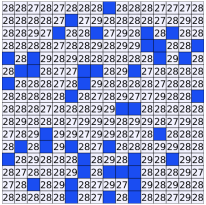
Solving Sylvester equations

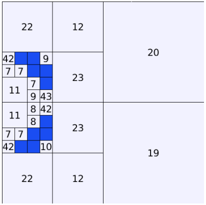
We have well established techniques for solving $\boxed{AX + XB = C}$ when:

- C is dense $\rightsquigarrow$ Bartels & Stewart [4] or Hessenberg-Schur [5] algorithms
- C is low-rank $\rightsquigarrow$ Krylov projection methods [6,7] or ADI [8,9]

We want to develop an algorithm to deal with the case $C \in$ HALR:







[4] Bartels, Stewart. *Algorithm 432: The solution of the matrix equation $AX - XB = C$*, Commun. ACM, 1972.
[5] Golub, Nash, Van Loan. *Hessenberg–Schur method for the problem $AX + XB = C$*, IEEE Trans. Automat. Control, 1979.
[6] Hu, Reichel. *Krylov-subspace methods for the Sylvester equation*, Libear Algebra Appl., 1992.
[7] Simoncini. *A new iterative method for solving large-scale Lyapunov matrix equations*, SISC, 2007.
[8] Wachpress. *Solution of Lyapunov equations by ADI iteration*, Comput. Math. Appl., 1991.
[9] Benner, Li, Truhar. *On the ADI method for Sylvester equations*, J. Comp. and App. Math., 2009.

Hierarchical low-rank structure in A, B

The matrices A and B are usually banded or have low-rank off-diagonal blocks.

From now on we assume that A and B can be block partitioned as:

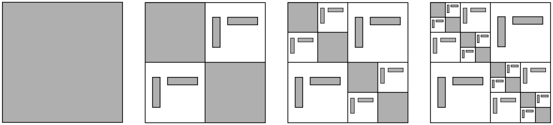
$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}}_{\textit{Block structured}} + \underbrace{\begin{bmatrix} & A_{12} \\ A_{21} & \end{bmatrix}}_{\text{low-rank}}.$$

The matrices A and B are usually banded or have low-rank off-diagonal blocks.

From now on we assume that A and B can be block partitioned as:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}}_{\textit{Block structured}} + \underbrace{\begin{bmatrix} & A_{12} \\ A_{21} & \end{bmatrix}}_{\text{low-rank}}.$$

Simple idea: store low-rank blocks as outer products, and diagonal ones recursively ($\mathcal{H}$-matrices, HODLR) [10].



[10] Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*, Springer Series in Computational Mathematics, 2015.

Sylvester equations with $A, B \in$ HODLR and $C \in$ HALR

Idea: HODLR matrices can be block-diagonalized via low-rank modifications.

Splitting A and B into their block diagonal and antidiagonal parts, leads to:

- Solve the equation

$$\begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix} X_0 + X_0 \begin{bmatrix} B_{11} & \\ & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix},$$

- Update X_0 by solving [11]

$$A\ \delta X + \delta X\ B = \underbrace{- \begin{bmatrix} & A_{12} \\ A_{21} & \end{bmatrix} X_0 - X_0 \begin{bmatrix} & B_{12} \\ B_{21} & \end{bmatrix}}_{\text{low-rank}}.$$

The first equation can be decomposed in 4 equations with HODLR coefficients of dimension $\frac{n}{2}$. This leads to a divide-and-conquer scheme.

[11] Kressner, Massei, Robol. *Low-rank updates and a divide-and-conquer algorithm for linear matrix equations*, SISC, 2019.

Sylvester equations with $A, B \in$ HODLR and $C \in$ HALR (cont'd)

procedure D&C_Sylv(A, B, C)
    if A, B are small matrices **then return** Bartels&Stewart(A, B, C)
    end if
    if $C = C_L C_R^*$ is low-rank **then return** low_rank_Sylv(A, B, C_L, C_R)
    end if
    Decompose

$$A = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} + \delta A, \quad B = \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix} + \delta B, \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

    $X_{11} \leftarrow \text{D\&C\_Sylv}(A_{11}, B_{11}, C_{11}), \quad X_{12} \leftarrow \text{D\&C\_Sylv}(A_{11}, B_{22}, C_{12})$
    $X_{21} \leftarrow \text{D\&C\_Sylv}(A_{22}, B_{11}, C_{21}), \quad X_{22} \leftarrow \text{D\&C\_Sylv}(A_{22}, B_{22}, C_{22})$
    $X_0 \leftarrow \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$
    Compute C_L and C_R such that $C_L C_R^* = -\delta A X_0 - X_0 \delta B$
    $\delta X \leftarrow$ low_rank_Sylv(A, B, C_L, C_R)
    return $X_0 + \delta X$
end procedure

Complexity and solution structure of D&C

$$\boxed{AX + XB = C}$$

Assumptions:

- C has low-rank blocks of rank $\leq r$; the storage cost of C is $\mathcal{O}(S)$
- A and B are HODLR matrices with HODLR rank $\leq k$
- Bartels&Stewart is applied only on matrices of size $\leq n_{\min}$
- Solving equations with low-rank RHS costs $\mathcal{O}(k^2 n \log^2(n))$

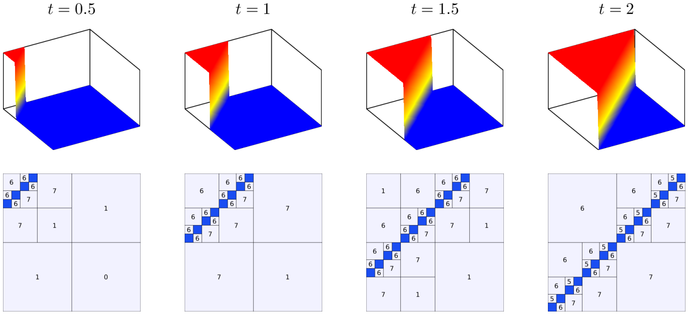
Theorem

The solution X has the same HALR structure of C with ranks $\mathcal{O}(r + k\log(n))$ and the D&C method costs $\mathcal{O}(S \cdot k^2 \log^2(n))$.

Remark: The estimate $\mathcal{O}(r + k\log(n))$ for the ranks in X is typically pessimistic.

Numerical results: Burgers equation

$$
\begin{cases}
\dfrac{\partial u}{\partial t} = 10^{-3}\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right) - u \cdot \left(\dfrac{\partial u}{\partial x} + \dfrac{\partial u}{\partial y}\right) & \\
u(x,y,t) = \frac{1}{1+\exp(10^3(x+y-t)/2)} & t = 0 \text{ or } (x,y) \in \partial\Omega
\end{cases}
$$



Numerical results: Burgers equation (cont'd)

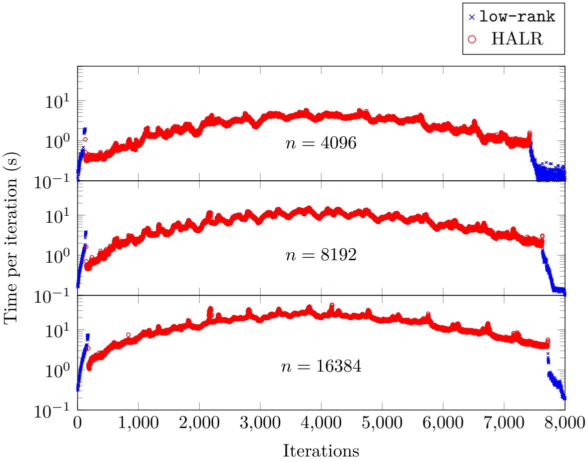
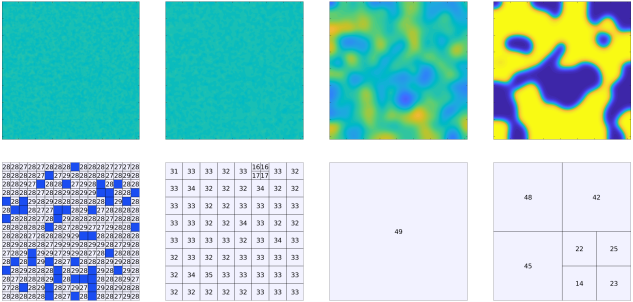


Figure: 8000 Iteration timings of the Euler-IMEX scheme on Burgers equation for different spatial discretization steps and $\mathrm{maxrank} = 50$.

Allen-Cahn equation

$$\begin{cases} \frac{\partial u}{\partial t} - 5 \cdot 10^{-5} \Delta u = u(u - \frac{1}{2})(1 - u) \\ u(x, y, 0) = \frac{1}{2} + \frac{1}{2}\texttt{randn} \\ \frac{\partial u}{\partial \vec{n}} = 0 & (x, y) \in \partial\Omega \end{cases}$$

$t = 0.3$ $\quad$ $t = 0.5$ $\quad$ $t = 18.0$ $\quad$ $t = 35.0$



Allen-Cahn equation (cont'd)

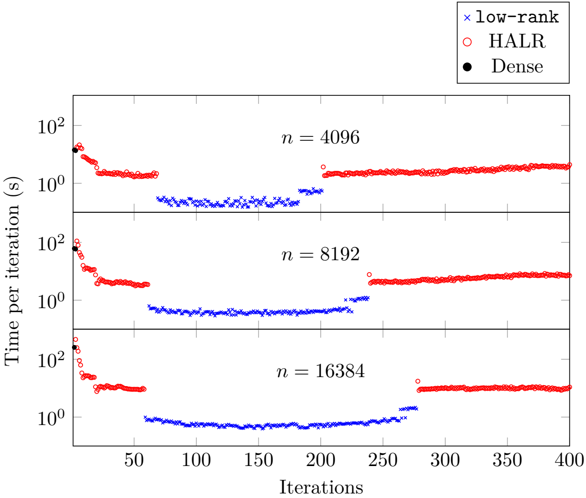


Figure: 400 Iteration timings of the Euler-IMEX scheme on Allen-Cahn equation for different spatial discretization steps and $\mathrm{maxrank} = 100$

Comparison with FFT-based solvers

HALR-based algorithms

n	Burgers T_{tot} (s)	Burgers Avg. T_{lyap} (s)	Allen-Cahn T_{tot} (s)	Allen-Cahn Avg. T_{lyap} (s)
4096	22334.0	1.32	505.2	0.81
8192	**57096.9**	4.01	1147.4	1.82
16384	**119130.4**	9.55	**2336.8**	3.32

FFT-based algorithms

n	Burgers T_{tot} (s)	Burgers Avg. T_{lyap} (s)	Allen-Cahn T_{tot} (s)	Allen-Cahn Avg. T_{lyap} (s)
4096	**18094**	2.26	**174.97**	0.44
8192	70541	8.82	**847.3**	2.12
16384	295507	36.94	2967	7.42

Conclusions & outlook

Take away messages:

- Exploiting local and time-dependent structures can make the difference.
- Sylvester equations with HODLR coefficients A, B can be solved with a complexity $\log^2$-proportional to the storage cost for the RHS.

What's next?

- Can we deal with 3D problems? Which tensorial format is the most suitable?

Full story:

- S.M., L. Robol, D. Kressner. *Hierarchical adaptive low-rank format with applications to discretized PDEs*, arXiv 2021.