Matrix-oriented numerical methods for semilinear PDEs

Valeria Simoncini

Dipartimento di Matematica
Alma Mater Studiorum - Università di Bologna
valeria.simoncini@unibo.it

Joint works with
M.C. D'Autilia & I. Sgura, Università di Lecce, Gerhard Kirsten, Università di Bologna

The differential problem

We are interested in solving

$$u_t = \ell(u) + f(u,t), \quad u = u(x,y,t) \quad \text{with } (x,y) \in \Omega \subset \mathbb{R}^2,\ t \in \mathcal{T}$$

with given initial conditions $u(x,y,0) = u_0(x,y)$ and proper b.c.

- ℓ linear in u (typically 2ˆorder diff operator in space, w/separable coeffs)
- f nonlinear function in u

Discretization: use tensor bases
(finite differences, conformal mappings, IGA, spectral methods, etc.)

♣ To simplify the presentation, Ω rectangle

The differential problem

We are interested in solving

$$u_t = \ell(u) + f(u,t), \quad u = u(x,y,t) \quad \text{with } (x,y) \in \Omega \subset \mathbb{R}^2,\ t \in \mathcal{T}$$

with given initial conditions $u(x,y,0) = u_0(x,y)$ and proper b.c.

- ℓ linear in u (typically 2^order diff operator in space, w/separable coeffs)
- f nonlinear function in u

> Discretization: use tensor bases
> (finite differences, conformal mappings, IGA, spectral methods, etc.)

♣ To simplify the presentation, Ω rectangle

The matrix differential problem. 1

$$u_t = \ell(u) + f(u,t), \quad u = u(x,y,t) \quad \text{with } (x,y) \in \Omega \subset \mathbb{R}^2,\ t \in \mathcal{T}$$

Linear operator:

$$\boxed{\ell(u) = \Delta u}$$

Standard (vector) discretization in space, $n_x \times n_y$ grid:

- $\Delta u \Rightarrow \mathcal{A}\boldsymbol{u} \qquad \mathcal{A} \in \mathbb{R}^{n_x n_y \times n_x n_y}$
- $f(u,t) \Rightarrow \boldsymbol{f}(\boldsymbol{u},t) \qquad$ ($n_x n_y$ components, evaluated component-wise)

with lexicographic ordering of the rectangle nodes

Matrix-oriented discretization in space:

- $\Delta u \Rightarrow A\boldsymbol{U} + \boldsymbol{U}B, \quad A \in \mathbb{R}^{n_x \times n_x},\ B \in \mathbb{R}^{n_y \times n_y}, (\boldsymbol{U})_{ij} \approx u(x_i, y_j)$
- $f(u,t) \Rightarrow \mathcal{F}(\boldsymbol{U},t)$ ($n_x \times n_y$, evaluated component-wise)

The matrix differential problem. 1

$$u_t = \ell(u) + f(u,t), \quad u = u(x,y,t) \quad \text{with } (x,y) \in \Omega \subset \mathbb{R}^2,\ t \in \mathcal{T}$$

Linear operator:

$$\boxed{\ell(u) = \Delta u}$$

Standard (vector) discretization in space, $n_x \times n_y$ grid:

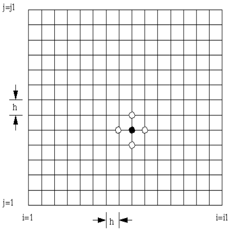
- $\Delta u \Rightarrow \mathcal{A}\boldsymbol{u} \qquad \mathcal{A} \in \mathbb{R}^{n_x n_y \times n_x n_y}$
- $f(u,t) \Rightarrow \boldsymbol{f}(\boldsymbol{u},t) \qquad$ ($n_x n_y$ components, evaluated component-wise)

with lexicographic ordering of the rectangle nodes

Matrix-oriented discretization in space:

- $\Delta u \Rightarrow A\boldsymbol{U} + \boldsymbol{U}B, \quad A \in \mathbb{R}^{n_x \times n_x},\ B \in \mathbb{R}^{n_y \times n_y}, (\boldsymbol{U})_{ij} \approx u(x_i, y_j)$
- $f(u,t) \Rightarrow \mathcal{F}(\boldsymbol{U},t)$ ($n_x \times n_y$, evaluated component-wise)

Reminder: matrix formulation of tensor discretization



Discretization: $U_{i,j} \approx u(x_i, y_j)$, with (x_i, y_j) interior nodes, so that

$$u_{xx}(x_i, y_j) \approx \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h^2} = \frac{1}{h^2}[1, -2, 1] \begin{bmatrix} U_{i-1,j} \\ U_{i,j} \\ U_{i+1,j} \end{bmatrix}$$

$$u_{yy}(x_i, y_j) \approx \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h^2} = \frac{1}{h^2}[U_{i,j-1},\ U_{i,j},\ U_{i,j+1}] \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

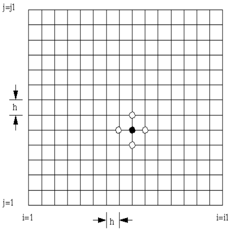
Let $T = \frac{1}{h^2}\mathrm{tridiag}(-1, \underline{2}, -1)$. Collecting all nodes together,

$$-u_{xx} \approx TU, \qquad -u_{yy} \approx UT$$

Therefore, directly from the grid,

$$-u_{xx} - u_{yy} \qquad \Rightarrow \qquad TU + UT + b.c.$$

Reminder: matrix formulation of tensor discretization



Discretization: $U_{i,j} \approx u(x_i, y_j)$, with (x_i, y_j) interior nodes, so that

$$u_{xx}(x_i, y_j) \approx \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h^2} = \frac{1}{h^2}[1, -2, 1] \begin{bmatrix} U_{i-1,j} \\ U_{i,j} \\ U_{i+1,j} \end{bmatrix}$$

$$u_{yy}(x_i, y_j) \approx \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h^2} = \frac{1}{h^2}[U_{i,j-1},\ U_{i,j},\ U_{i,j+1}] \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Let $T = \frac{1}{h^2}\mathrm{tridiag}(-1, \underline{2}, -1)$. Collecting all nodes together,

$$-u_{xx} \approx TU, \qquad -u_{yy} \approx UT$$

Therefore, directly from the grid,

$$-u_{xx} - u_{yy} \qquad \Rightarrow \qquad TU + UT + b.c.$$

The matrix differential equation

$$\dot{\boldsymbol{U}}(t) = A\boldsymbol{U}(t) + \boldsymbol{U}(t)B + \mathcal{F}(\boldsymbol{U}, t), \quad \boldsymbol{U}(0) = \boldsymbol{U}_0$$

Computational strategies. Time stepping methods:

- **Small scale:** matrix-oriented IMEX methods, exponential integrators
- **Large scale:** In sequence:
  1. Order reduction procedure ($\Rightarrow$ POD-type)
  2. Feasible handling of nonlinear term $\mathcal{F}(\boldsymbol{U}, t)$ ($\Rightarrow$ matrix DEIM)
  3. Time stepping of reduced problem

The matrix differential equation

$$\dot{\boldsymbol{U}}(t) = A\boldsymbol{U}(t) + \boldsymbol{U}(t)B + \mathcal{F}(\boldsymbol{U}, t), \quad \boldsymbol{U}(0) = \boldsymbol{U}_0$$

Computational strategies. Time stepping methods:

- **Small scale:** matrix-oriented IMEX methods, exponential integrators
- **Large scale:** In sequence:
  1. Order reduction procedure ($\Rightarrow$ POD-type)
  2. Feasible handling of nonlinear term $\mathcal{F}(\boldsymbol{U}, t)$ ($\Rightarrow$ matrix DEIM)
  3. Time stepping of reduced problem

The matrix differential equation

$$\dot{\boldsymbol{U}}(t) = A\boldsymbol{U}(t) + \boldsymbol{U}(t)B + \mathcal{F}(\boldsymbol{U}, t), \quad \boldsymbol{U}(0) = \boldsymbol{U}_0$$

Computational strategies. Time stepping methods:

- **Small scale:** matrix-oriented IMEX methods, exponential integrators
- **Large scale:** In sequence:
  1. Order reduction procedure ($\Rightarrow$ POD-type)
  2. Feasible handling of nonlinear term $\mathcal{F}(\boldsymbol{U}, t)$ ($\Rightarrow$ matrix DEIM)
  3. Time stepping of reduced problem

Small scale time stepping

$$u_t = \ell(u) + f(u,t), \quad u = u(x,y,t) \quad \text{with } (x,y) \in \Omega \subset \mathbb{R}^2,\ t \in \mathcal{T}$$

- Problem is stiff
  - Use appropriate time discretizations
  - Time stepping constraints
- Possibly long time period (e.g., for pattern detection), with occurrence of transient unstable phase
- Phenomenon sets is only if domain is well represented

$$\dot{\boldsymbol{U}}(t) = A\boldsymbol{U}(t) + \boldsymbol{U}(t)B + \mathcal{F}(\boldsymbol{U},t), \quad \boldsymbol{U}(0) = \boldsymbol{U}_0$$

Time stepping Matrix-oriented methods

IMEX methods

1. *First order Euler:* $\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = h_t(\mathcal{A}\boldsymbol{u}_{n+1} + f(\boldsymbol{u}_n))$ so that

$$(I - h_t\mathcal{A})\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h_t f(\boldsymbol{u}_n), \quad n = 0, \ldots, N_t - 1$$

Matrix-oriented form: $U_{n+1} - U_n = h_t(AU_{n+1} + U_{n+1}B) + h_t F(U_n)$, so that

$$(I - h_t A)\mathrm{U}_{n+1} + \mathrm{U}_{n+1}(-h_t B) = U_n + h_t F(U_n), \quad n = 0, \ldots, N_t - 1.$$

2. *Second order SBDF*, known as IMEX 2-SBDF method

$$3\boldsymbol{u}_{n+2} - 4\boldsymbol{u}_{n+1} + \boldsymbol{u}_n = 2h_t\mathcal{A}\boldsymbol{u}_{n+2} + 2h_t(2f(\boldsymbol{u}_{n+1}) - f(\boldsymbol{u}_n)), \quad n = 0, 1, \ldots, N_t$$

Matrix-oriented form: for $n = 0, \ldots, N_t - 2$,

$$(3I - 2h_t A)\,\mathrm{U}_{n+2} + \mathrm{U}_{n+2}\,(-2h_t B) = 4U_{n+1} - U_n + 2h_t(2F(U_{n+1}) - F(U_n))$$

Time stepping Matrix-oriented methods

IMEX methods

1. *First order Euler:* $\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = h_t(\mathcal{A}\boldsymbol{u}_{n+1} + f(\boldsymbol{u}_n))$ so that

$$(I - h_t\mathcal{A})\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h_t f(\boldsymbol{u}_n), \quad n = 0, \ldots, N_t - 1$$

Matrix-oriented form: $U_{n+1} - U_n = h_t(AU_{n+1} + U_{n+1}B) + h_t F(U_n)$, so that

$$(I - h_t A)\mathrm{U}_{n+1} + \mathrm{U}_{n+1}(-h_t B) = U_n + h_t F(U_n), \quad n = 0, \ldots, N_t - 1.$$

2. *Second order SBDF*, known as IMEX 2-SBDF method

$$3\boldsymbol{u}_{n+2} - 4\boldsymbol{u}_{n+1} + \boldsymbol{u}_n = 2h_t\mathcal{A}\boldsymbol{u}_{n+2} + 2h_t(2f(\boldsymbol{u}_{n+1}) - f(\boldsymbol{u}_n)), \quad n = 0, 1, \ldots, N_t$$

Matrix-oriented form: for $n = 0, \ldots, N_t - 2$,

$$(3I - 2h_t A)\,\mathrm{U}_{n+2} + \mathrm{U}_{n+2}\,(-2h_t B) = 4U_{n+1} - U_n + 2h_t(2F(U_{n+1}) - F(U_n))$$

Time stepping Matrix-oriented methods

IMEX methods

1. *First order Euler:* $\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = h_t(\mathcal{A}\boldsymbol{u}_{n+1} + f(\boldsymbol{u}_n))$ so that

$$(I - h_t\mathcal{A})\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h_t f(\boldsymbol{u}_n), \quad n = 0, \ldots, N_t - 1$$

Matrix-oriented form: $U_{n+1} - U_n = h_t(AU_{n+1} + U_{n+1}B) + h_t F(U_n)$, so that

$$(I - h_t A)\mathrm{U}_{n+1} + \mathrm{U}_{n+1}(-h_t B) = U_n + h_t F(U_n), \quad n = 0, \ldots, N_t - 1.$$

2. *Second order SBDF*, known as IMEX 2-SBDF method

$$3\boldsymbol{u}_{n+2} - 4\boldsymbol{u}_{n+1} + \boldsymbol{u}_n = 2h_t\mathcal{A}\boldsymbol{u}_{n+2} + 2h_t(2f(\boldsymbol{u}_{n+1}) - f(\boldsymbol{u}_n)), \quad n = 0, 1, \ldots, N_t$$

Matrix-oriented form: for $n = 0, \ldots, N_t - 2$,

$$(3I - 2h_t A)\,\mathrm{U}_{n+2} + \mathrm{U}_{n+2}\,(-2h_t B) = 4U_{n+1} - U_n + 2h_t(2F(U_{n+1}) - F(U_n))$$

Time stepping Matrix-oriented methods

IMEX methods

1. *First order Euler:* $\boldsymbol{u}_{n+1} - \boldsymbol{u}_n = h_t(\mathcal{A}\boldsymbol{u}_{n+1} + f(\boldsymbol{u}_n))$ so that

$$(I - h_t\mathcal{A})\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h_t f(\boldsymbol{u}_n), \quad n = 0, \ldots, N_t - 1$$

Matrix-oriented form: $U_{n+1} - U_n = h_t(AU_{n+1} + U_{n+1}B) + h_t F(U_n)$, so that

$$(I - h_t A)\mathrm{U}_{n+1} + \mathrm{U}_{n+1}(-h_t B) = U_n + h_t F(U_n), \quad n = 0, \ldots, N_t - 1.$$

2. *Second order SBDF*, known as IMEX 2-SBDF method

$$3\boldsymbol{u}_{n+2} - 4\boldsymbol{u}_{n+1} + \boldsymbol{u}_n = 2h_t\mathcal{A}\boldsymbol{u}_{n+2} + 2h_t(2f(\boldsymbol{u}_{n+1}) - f(\boldsymbol{u}_n)), \quad n = 0, 1, \ldots, N_t$$

Matrix-oriented form: for $n = 0, \ldots, N_t - 2$,

$$(3I - 2h_t A)\,\mathrm{U}_{n+2} + \mathrm{U}_{n+2}\,(-2h_t B) = 4U_{n+1} - U_n + 2h_t(2F(U_{n+1}) - F(U_n))$$

Time stepping Matrix-oriented methods

Exponential integrator

Exponential first order Euler method:

$$\boldsymbol{u}_{n+1} = e^{h_t\mathcal{A}}\boldsymbol{u}_n + h_t\varphi_1(h_t\mathcal{A})f(\boldsymbol{u}_n)$$

$e^{h_t\mathcal{A}}$: matrix exponential, $\varphi_1(z) = (e^z - 1)/z$ first "phi" function
That is,

$$\boldsymbol{u}_{n+1} = e^{h_t\mathcal{A}}\boldsymbol{u}_n + h_t\boldsymbol{v}_n, \quad \text{where } \mathcal{A}\boldsymbol{v}_n = e^{h_t\mathcal{A}}f(\boldsymbol{u}_n) - f(\boldsymbol{u}_n) \qquad n = 0, \ldots, N_t - 1.$$

Matrix-oriented form: since $e^{h_t\mathcal{A}}\boldsymbol{u} = \left(e^{h_tB^\top} \otimes e^{h_tA}\right)\boldsymbol{u} = \mathrm{vec}(e^{h_tA}\boldsymbol{U}e^{h_tB})$

1. Compute $E_1 = e^{h_tA}$, $E_2 = e^{h_tB^\top}$
2. For each n

$$\begin{aligned} \text{Solve} \qquad & AV_n + V_nB = E_1F(\boldsymbol{U}_n)E_2^\top - F(\boldsymbol{U}_n) \\ \text{Compute} \qquad & \boldsymbol{U}_{n+1} = E_1\boldsymbol{U}_nE_2^\top + h_tV_n \end{aligned}$$

Time stepping Matrix-oriented methods

Exponential integrator

Exponential first order Euler method:

$$\boldsymbol{u}_{n+1} = e^{h_t \mathcal{A}} \boldsymbol{u}_n + h_t \varphi_1(h_t \mathcal{A}) f(\boldsymbol{u}_n)$$

$e^{h_t \mathcal{A}}$: matrix exponential, $\varphi_1(z) = (e^z - 1)/z$ first "phi" function
That is,

$$\boldsymbol{u}_{n+1} = e^{h_t \mathcal{A}} \boldsymbol{u}_n + h_t \boldsymbol{v}_n, \quad \text{where } \mathcal{A} \boldsymbol{v}_n = e^{h_t \mathcal{A}} f(\boldsymbol{u}_n) - f(\boldsymbol{u}_n) \qquad n = 0, \ldots, N_t - 1.$$

Matrix-oriented form: since $e^{h_t \mathcal{A}} \boldsymbol{u} = \left(e^{h_t B^\top} \otimes e^{h_t A} \right) \boldsymbol{u} = \mathrm{vec}(e^{h_t A} \boldsymbol{U} e^{h_t B})$

1. Compute $E_1 = e^{h_t A}$, $E_2 = e^{h_t B^\top}$
2. For each n

$$\begin{array}{rc} \text{Solve} & A V_n + V_n B = E_1 F(\boldsymbol{U}_n) E_2^\top - F(\boldsymbol{U}_n) \\ \text{Compute} & \boldsymbol{U}_{n+1} = E_1 \boldsymbol{U}_n E_2^\top + h_t V_n \end{array}$$

Time stepping Matrix-oriented methods

Computational issues:

- Dimensions of A, B very modest
- A, B quasi-symmetric (non-symmetry due to bc's)
- A, B do not depend on time step

♣ Matrix-oriented form all in spectral space (after eigenvector transformation)

Numerical properties:

Structural properties are preserved

Time stepping Matrix-oriented methods

Computational issues:

- Dimensions of A, B very modest
- A, B quasi-symmetric (non-symmetry due to bc's)
- A, B do not depend on time step

♣ Matrix-oriented form all in spectral space (after eigenvector transformation)

Numerical properties:

Structural properties are preserved

A numerical example of system of RD-PDEs

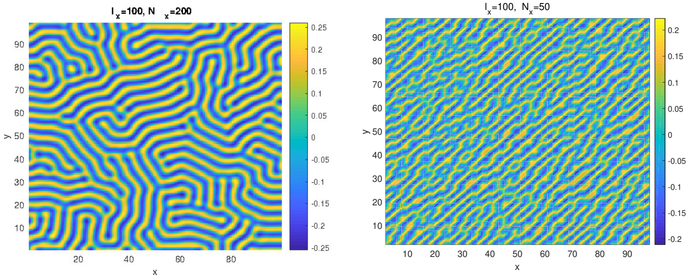
$$\begin{cases} u_t = \ell_1(u) + f_1(u,v), \\ v_t = \ell_2(v) + f_2(u,v), \quad \text{with} \quad (x,y) \in \Omega \subset \mathbb{R}^2, \quad t \in]0,T] \end{cases}$$

Model describing an electrodeposition process for metal growth

$f_1(u,v) = \rho\left(A_1(1-v)u - A_2\, u^3 - B(v-\alpha)\right)$

$f_2(u,v) = \rho\left(C(1+k_2 u)(1-v)[1-\gamma(1-v)] - Dv(1+k_3 u)(1+\gamma v)\right)$

Turing pattern



Joint work with M.C. D'Autilia & I. Sgura, Università di Lecce

A numerical example of system of RD-PDEs

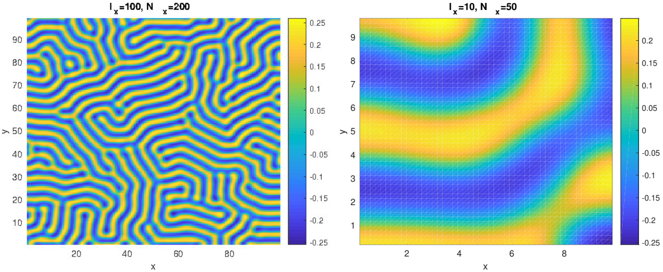
$$\begin{cases} u_t = \ell_1(u) + f_1(u,v), \\ v_t = \ell_2(v) + f_2(u,v), \quad \text{with} \quad (x,y) \in \Omega \subset \mathbb{R}^2, \quad t \in]0, T] \end{cases}$$

Model describing an electrodeposition process for metal growth

$f_1(u,v) = \rho\left(A_1(1-v)u - A_2\, u^3 - B(v-\alpha)\right)$

$f_2(u,v) = \rho\left(C(1+k_2 u)(1-v)[1-\gamma(1-v)] - Dv(1+k_3 u)(1+\gamma v)\right)$

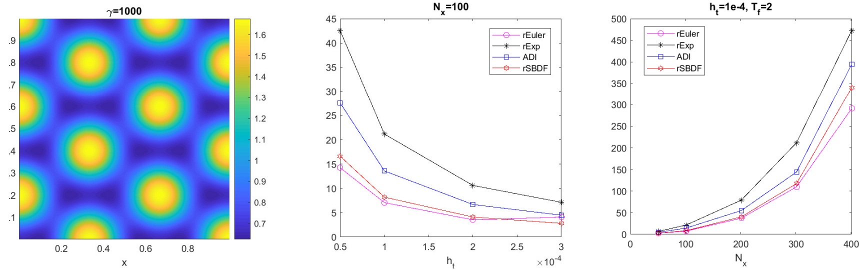
Turing pattern



Joint work with M.C. D'Autilia & I. Sgura, Università di Lecce

Schnackenberg model

$$f_1(u,v) = \gamma(a - u + u^2 v), \;\; f_2(u,v) = \gamma(b - u^2 v)$$



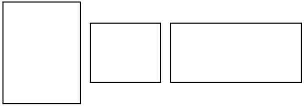
Left plot: Turing pattern solution for $\gamma = 1000$ ($N_x = 400$)
Center plot: CPU times (sec), $N_x = 100$ variation of h_t
Right plot: CPU times (sec), $h_t = 10^{-4}$, increasing values of $N_x = 50, 100, 200, 300, 400$

Large scale time stepping

$$\dot{\boldsymbol{U}}(t) = A\boldsymbol{U}(t) + \boldsymbol{U}(t)B + \mathcal{F}(\boldsymbol{U}, t), \quad \boldsymbol{U}(0) = \boldsymbol{U}_0$$

Approximation strategy: $\boldsymbol{U} \in \mathbb{R}^{n_x \times n_y}$,

$$\boldsymbol{U} \approx \boldsymbol{V}_{\ell,U} \boldsymbol{Y}_k(t) \boldsymbol{W}_{r,U}^\top =$$



$$, \quad t \in [0, T_f]$$

♣ $\boldsymbol{V}_{\ell,U} \in \mathbb{R}^{n_x \times k_1}$, $\boldsymbol{W}_{r,U} \in \mathbb{R}^{n_y \times k_2}$ matrices[1] to be determined, independent of time

♣ Function $\boldsymbol{Y}_k(t)$ numerical solution to *reduced* semilinear problem:

$$\dot{\boldsymbol{Y}}_k(t) = A_k \boldsymbol{Y}_k(t) + \boldsymbol{Y}_k(t) B_k + \widehat{\mathcal{F}_k(\boldsymbol{Y}_k, t)}$$

$$\boldsymbol{Y}_k(0) = \boldsymbol{Y}_k^{(0)} := \boldsymbol{V}_{\ell,U}^\top \boldsymbol{U}_0 \boldsymbol{W}_{r,U}$$

with $A_k = \boldsymbol{V}_{\ell,U}^\top A \boldsymbol{V}_{\ell,U}$, $B_k = \boldsymbol{W}_{r,U}^\top B \boldsymbol{W}_{r,U}$,

♣ $\widehat{\mathcal{F}_k(\boldsymbol{Y}_k, t)}$ is a matrix-oriented DEIM approximation to

$$\mathcal{F}_k(\boldsymbol{Y}_k, t) = \boldsymbol{V}_{\ell,U}^\top \mathcal{F}(\boldsymbol{V}_{\ell,U} \boldsymbol{Y}_k \boldsymbol{W}_{r,U}^\top, t) \boldsymbol{W}_{r,U}.$$

[1] $k_1, k_2 \ll n$ and we let $k = (k_1, k_2)$

Large scale time stepping

$$\dot{\boldsymbol{U}}(t) = A\boldsymbol{U}(t) + \boldsymbol{U}(t)B + \mathcal{F}(\boldsymbol{U}, t), \quad \boldsymbol{U}(0) = \boldsymbol{U}_0$$

Approximation strategy: $\boldsymbol{U} \in \mathbb{R}^{n_x \times n_y}$,

$$\boldsymbol{U} \approx \boldsymbol{V}_{\ell,U} \boldsymbol{Y}_k(t) \boldsymbol{W}_{r,U}^\top = \square\;\square\;\square, \quad t \in [0, T_f]$$

♣ $\boldsymbol{V}_{\ell,U} \in \mathbb{R}^{n_x \times k_1}$, $\boldsymbol{W}_{r,U} \in \mathbb{R}^{n_y \times k_2}$ matrices[1] to be determined, independent of time

♣ Function $\boldsymbol{Y}_k(t)$ numerical solution to *reduced* semilinear problem:

$$\dot{\boldsymbol{Y}}_k(t) = A_k \boldsymbol{Y}_k(t) + \boldsymbol{Y}_k(t) B_k + \widehat{\mathcal{F}_k(\boldsymbol{Y}_k, t)}$$
$$\boldsymbol{Y}_k(0) = \boldsymbol{Y}_k^{(0)} := \boldsymbol{V}_{\ell,U}^\top \boldsymbol{U}_0 \boldsymbol{W}_{r,U}$$

with $A_k = \boldsymbol{V}_{\ell,U}^\top A \boldsymbol{V}_{\ell,U}$, $B_k = \boldsymbol{W}_{r,U}^\top B \boldsymbol{W}_{r,U}$,

♣ $\widehat{\mathcal{F}_k(\boldsymbol{Y}_k, t)}$ is a matrix-oriented DEIM approximation to

$$\mathcal{F}_k(\boldsymbol{Y}_k, t) = \boldsymbol{V}_{\ell,U}^\top \mathcal{F}(\boldsymbol{V}_{\ell,U} \boldsymbol{Y}_k \boldsymbol{W}_{r,U}^\top, t) \boldsymbol{W}_{r,U}.$$

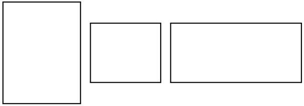
[1] $k_1, k_2 \ll n$ and we let $k = (k_1, k_2)$

Large scale time stepping

$$\dot{\boldsymbol{U}}(t) = A\boldsymbol{U}(t) + \boldsymbol{U}(t)B + \mathcal{F}(\boldsymbol{U}, t), \quad \boldsymbol{U}(0) = \boldsymbol{U}_0$$

Approximation strategy: $\boldsymbol{U} \in \mathbb{R}^{n_x \times n_y}$,

$$\boldsymbol{U} \approx \boldsymbol{V}_{\ell,U} \boldsymbol{Y}_k(t) \boldsymbol{W}_{r,U}^\top =$$



$$, \quad t \in [0, T_f]$$

♣ $\boldsymbol{V}_{\ell,U} \in \mathbb{R}^{n_x \times k_1}$, $\boldsymbol{W}_{r,U} \in \mathbb{R}^{n_y \times k_2}$ matrices[1] to be determined, independent of time

♣ Function $\boldsymbol{Y}_k(t)$ numerical solution to *reduced* semilinear problem:

$$\dot{\boldsymbol{Y}}_k(t) = A_k \boldsymbol{Y}_k(t) + \boldsymbol{Y}_k(t) B_k + \widehat{\mathcal{F}_k(\boldsymbol{Y}_k, t)}$$
$$\boldsymbol{Y}_k(0) = \boldsymbol{Y}_k^{(0)} := \boldsymbol{V}_{\ell,U}^\top \boldsymbol{U}_0 \boldsymbol{W}_{r,U}$$

with $A_k = \boldsymbol{V}_{\ell,U}^\top A \boldsymbol{V}_{\ell,U}$, $B_k = \boldsymbol{W}_{r,U}^\top B \boldsymbol{W}_{r,U}$,

♣ $\widehat{\mathcal{F}_k(\boldsymbol{Y}_k, t)}$ is a matrix-oriented DEIM approximation to

$$\mathcal{F}_k(\boldsymbol{Y}_k, t) = \boldsymbol{V}_{\ell,U}^\top \mathcal{F}(\boldsymbol{V}_{\ell,U} \boldsymbol{Y}_k \boldsymbol{W}_{r,U}^\top, t) \boldsymbol{W}_{r,U}.$$

[1] $k_1, k_2 \ll n$ and we let $k = (k_1, k_2)$

Large scale time stepping: computational steps

- Determine $\boldsymbol{V}_{\ell,U} \in \mathbb{R}^{n_x \times k_1}$, $\boldsymbol{W}_{r,U} \in \mathbb{R}^{n_y \times k_2}$ via new two-sided matrix POD
  (*literature: parameter-based affine formulations/approximations (MDEIM, Manzoni etal), Jacobian approximation (Stefanescu etal 2017),...*)
- Determine $\widehat{\mathcal{F}_k(\boldsymbol{Y}_k, t)}$ via new matrix-oriented DEIM
- Matrix-oriented time stepping for $\boldsymbol{Y}_k(t)$ (small scale)

for general vector treatment, Benner, Gugercin, Willcox, SIREV 2015

♣ Matrix formulation preserves structure, e.g., symmetry

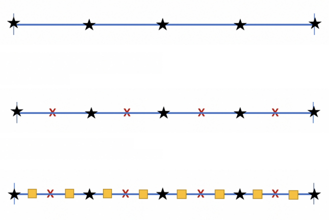
Two-sided matrix POD

POD: select an approximation basis using solution "snapshots" $\{\boldsymbol{U}(t_i)\}_{i=1}^{n_{\max}}$

Matrix-oriented POD: select two bases

Snapshot dynamic selection procedure

Refinements $\mathcal{I}_j$, $j = 1, \ldots, 3$ of time interval



♣ In fact: Snapshots computed on the fly (SI Euler) while the time interval is spanned

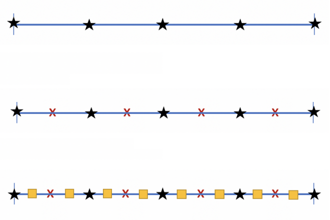
Two-sided matrix POD

POD: select an approximation basis using solution "snapshots" $\{\boldsymbol{U}(t_i)\}_{i=1}^{n_{\max}}$

Matrix-oriented POD: select two bases

Snapshot dynamic selection procedure

Refinements $\mathcal{I}_j$, $j = 1, \ldots, 3$ of time interval



♣ In fact: Snapshots computed on the fly (SI Euler) while the time interval is spanned

Two-sided matrix POD

Given snapshots $\{\boldsymbol{U}(t_i)\}_{i=1}^{n_{\max}}$ and three refinements $\mathcal{I}_j$, $j = 1, \ldots, 3$ of time interval

for each $j = 1, \ldots, 3$,

for each $t_i \in \mathcal{I}_j$ such that $\boldsymbol{U}_i$ is to be included

- Compute $[\boldsymbol{V}_i, \boldsymbol{\Sigma}_i, \boldsymbol{W}_i] = \texttt{svds}\,(\boldsymbol{U}_i, \kappa)$
- Append $\widetilde{\boldsymbol{V}}_i \leftarrow (\widetilde{\boldsymbol{V}}_{i-1}, \boldsymbol{V}_i)$, $\widehat{\boldsymbol{W}}_i \leftarrow (\widehat{\boldsymbol{W}}_{i-1}, \boldsymbol{W}_i)$, $\widetilde{\boldsymbol{\Sigma}}_i \leftarrow \texttt{blkdiag}(\widetilde{\boldsymbol{\Sigma}}_{i-1}, \boldsymbol{\Sigma}_i)$
- Decreasingly order the entries of (diagonal) $\widetilde{\boldsymbol{\Sigma}}_i$ and keep the first κ
- Order $\widetilde{\boldsymbol{V}}_i$ and $\widehat{\boldsymbol{W}}_i$ accordingly and keep the first κ vectors of each

Check if next refinement is needed

Matrix-oriented DEIM approximation of nonlinear function

Matrix-oriented POD: Given snapshots $\{\mathcal{F}(t_j)\}_{j=1}^{n_s}$ use dynamic selection to generate $\boldsymbol{V}_{\ell,\mathcal{F}} \in \mathbb{R}^{n\times p_1}$ such that

$$\mathcal{F}(t) \approx \boldsymbol{V}_{\ell,\mathcal{F}} \boldsymbol{C}(t) \boldsymbol{W}_{r,\mathcal{F}}^\top$$

with $\boldsymbol{C}(t)$ to be determined.

DEIM strategy, in a two-sided context (2S-DEIM):

$$\boldsymbol{V}_{\ell,\mathcal{F}} \boldsymbol{C}(t) \boldsymbol{W}_{r,\mathcal{F}}^\top = \mathcal{F}(t)$$

1. Select independent rows of $\boldsymbol{V}_{\ell,\mathcal{F}}$ and $\boldsymbol{W}_{r,\mathcal{F}}$ (via reduction indices)

$$\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,\mathcal{F}} \boldsymbol{C}(t) \boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{P}_{r,\mathcal{F}} = \boldsymbol{P}_{\ell,\mathcal{F}}^\top \mathcal{F}(t) \boldsymbol{P}_{r,\mathcal{F}},$$

Solve for $\boldsymbol{C}(t)$

2. Project onto U-spaces: (element-wise eval of $\mathcal{F}$)

$$\begin{aligned}\mathcal{F}_k(\boldsymbol{Y}_k,t) &\approx \boldsymbol{V}_{\ell,U}^\top \boldsymbol{V}_{\ell,\mathcal{F}}(\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,\mathcal{F}})^{-1} \boldsymbol{P}_{\ell,\mathcal{F}}^\top \mathcal{F}(\boldsymbol{V}_{\ell,U}\boldsymbol{Y}_k(t)\boldsymbol{W}_{r,U}^\top,t)\boldsymbol{P}_{r,\mathcal{F}}(\boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{P}_{r,\mathcal{F}})^{-1}\boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{W}_{r,U} \\ &= \boldsymbol{V}_{\ell,U}^\top \boldsymbol{V}_{\ell,\mathcal{F}}(\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,\mathcal{F}})^{-1} \mathcal{F}(\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,U}\boldsymbol{Y}_k(t)\boldsymbol{W}_{r,U}^\top \boldsymbol{P}_{r,\mathcal{F}},t)(\boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{P}_{r,\mathcal{F}})^{-1}\boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{W}_{r,U} \\ &=: \widehat{\mathcal{F}_k(\boldsymbol{Y}_k,t)}\end{aligned}$$

Matrix-oriented DEIM approximation of nonlinear function

Matrix-oriented POD: Given snapshots $\{\mathcal{F}(t_j)\}_{j=1}^{n_s}$ use dynamic selection to generate $\boldsymbol{V}_{\ell,\mathcal{F}} \in \mathbb{R}^{n \times p_1}$ such that

$$\mathcal{F}(t) \approx \boldsymbol{V}_{\ell,\mathcal{F}} \boldsymbol{C}(t) \boldsymbol{W}_{r,\mathcal{F}}^\top$$

with $\boldsymbol{C}(t)$ to be determined.

DEIM strategy, in a two-sided context (2S-DEIM):

$$\boldsymbol{V}_{\ell,\mathcal{F}} \boldsymbol{C}(t) \boldsymbol{W}_{r,\mathcal{F}}^\top = \mathcal{F}(t)$$

1. Select independent rows of $\boldsymbol{V}_{\ell,\mathcal{F}}$ and $\boldsymbol{W}_{r,\mathcal{F}}$ (via reduction indices)

$$\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,\mathcal{F}} \boldsymbol{C}(t) \boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{P}_{r,\mathcal{F}} = \boldsymbol{P}_{\ell,\mathcal{F}}^\top \mathcal{F}(t) \boldsymbol{P}_{r,\mathcal{F}},$$

Solve for $\boldsymbol{C}(t)$

2. Project onto U-spaces: (element-wise eval of $\mathcal{F}$)

$$\begin{aligned}\mathcal{F}_k(\boldsymbol{Y}_k, t) &\approx \boldsymbol{V}_{\ell,U}^\top \boldsymbol{V}_{\ell,\mathcal{F}} (\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,\mathcal{F}})^{-1} \boldsymbol{P}_{\ell,\mathcal{F}}^\top \mathcal{F}(\boldsymbol{V}_{\ell,U} \boldsymbol{Y}_k(t) \boldsymbol{W}_{r,U}^\top, t) \boldsymbol{P}_{r,\mathcal{F}} (\boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{P}_{r,\mathcal{F}})^{-1} \boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{W}_{r,U} \\ &= \boldsymbol{V}_{\ell,U}^\top \boldsymbol{V}_{\ell,\mathcal{F}} (\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,\mathcal{F}})^{-1} \mathcal{F}(\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,U} \boldsymbol{Y}_k(t) \boldsymbol{W}_{r,U}^\top \boldsymbol{P}_{r,\mathcal{F}}, t) (\boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{P}_{r,\mathcal{F}})^{-1} \boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{W}_{r,U} \\ &=: \widehat{\mathcal{F}_k(\boldsymbol{Y}_k, t)}\end{aligned}$$

Matrix-oriented DEIM approximation of nonlinear function

Matrix-oriented POD: Given snapshots $\{\mathcal{F}(t_j)\}_{j=1}^{n_s}$ use dynamic selection to generate $\boldsymbol{V}_{\ell,\mathcal{F}} \in \mathbb{R}^{n\times p_1}$ such that

$$\mathcal{F}(t) \approx \boldsymbol{V}_{\ell,\mathcal{F}}\boldsymbol{C}(t)\boldsymbol{W}_{r,\mathcal{F}}^\top$$

with $\boldsymbol{C}(t)$ to be determined.

DEIM strategy, in a two-sided context (2S-DEIM):

$$\boldsymbol{V}_{\ell,\mathcal{F}}\boldsymbol{C}(t)\boldsymbol{W}_{r,\mathcal{F}}^\top = \mathcal{F}(t)$$

1. Select independent rows of $\boldsymbol{V}_{\ell,\mathcal{F}}$ and $\boldsymbol{W}_{r,\mathcal{F}}$ (via reduction indices)

$$\boldsymbol{P}_{\ell,\mathcal{F}}^\top\boldsymbol{V}_{\ell,\mathcal{F}}\boldsymbol{C}(t)\boldsymbol{W}_{r,\mathcal{F}}^\top\boldsymbol{P}_{r,\mathcal{F}} = \boldsymbol{P}_{\ell,\mathcal{F}}^\top\mathcal{F}(t)\boldsymbol{P}_{r,\mathcal{F}},$$

Solve for $\boldsymbol{C}(t)$

2. Project onto U-spaces: (element-wise eval of $\mathcal{F}$)

$$\begin{aligned}\mathcal{F}_k(\boldsymbol{Y}_k,t) &\approx \boldsymbol{V}_{\ell,U}^\top\boldsymbol{V}_{\ell,\mathcal{F}}(\boldsymbol{P}_{\ell,\mathcal{F}}^\top\boldsymbol{V}_{\ell,\mathcal{F}})^{-1}\boldsymbol{P}_{\ell,\mathcal{F}}^\top\mathcal{F}(\boldsymbol{V}_{\ell,U}\boldsymbol{Y}_k(t)\boldsymbol{W}_{r,U}^\top,t)\boldsymbol{P}_{r,\mathcal{F}}(\boldsymbol{W}_{r,\mathcal{F}}^\top\boldsymbol{P}_{r,\mathcal{F}})^{-1}\boldsymbol{W}_{r,\mathcal{F}}^\top\boldsymbol{W}_{r,U}\\ &= \boldsymbol{V}_{\ell,U}^\top\boldsymbol{V}_{\ell,\mathcal{F}}(\boldsymbol{P}_{\ell,\mathcal{F}}^\top\boldsymbol{V}_{\ell,\mathcal{F}})^{-1}\mathcal{F}(\boldsymbol{P}_{\ell,\mathcal{F}}^\top\boldsymbol{V}_{\ell,U}\boldsymbol{Y}_k(t)\boldsymbol{W}_{r,U}^\top\boldsymbol{P}_{r,\mathcal{F}},t)(\boldsymbol{W}_{r,\mathcal{F}}^\top\boldsymbol{P}_{r,\mathcal{F}})^{-1}\boldsymbol{W}_{r,\mathcal{F}}^\top\boldsymbol{W}_{r,U}\\ &=: \widehat{\mathcal{F}_k(\boldsymbol{Y}_k,t)}\end{aligned}$$

Algorithm 2S-POD-DEIM

INPUT: n_{max}, κ, and τ, $n_\mathfrak{t}$, $\{\mathfrak{t}_i\}$, $i = 0, \ldots, n_\mathfrak{t}$

OUTPUT: $\boldsymbol{V}_{\ell,U}$, $\boldsymbol{W}_{r,U}$ and $\boldsymbol{Y}_k^{(i)}$, $i = 0, \ldots, n_\mathfrak{t}$ (for $\boldsymbol{V}_{\ell,U}\boldsymbol{Y}_k^{(i)}\boldsymbol{W}_{r,U}^\top \approx \boldsymbol{U}(\mathfrak{t}_i)$)

Offline:

1. Determine $\boldsymbol{V}_{\ell,U}, \boldsymbol{W}_{r,U}$ for $\{\boldsymbol{U}\}_{i=1}^{n_{\max}}$ and $\boldsymbol{V}_{\ell,\mathcal{F}}, \boldsymbol{W}_{r,\mathcal{F}}$ for $\{\mathcal{F}\}_{i=1}^{n_{\max}}$ via dynamic procedure
2. Compute $\boldsymbol{Y}_k^{(0)} = \boldsymbol{V}_{\ell,U}^\top \boldsymbol{U}_0 \boldsymbol{W}_{r,U}$, $A_k = \boldsymbol{V}_{\ell,U}^\top \boldsymbol{A}\boldsymbol{V}_{\ell,U}$, $B_k = \boldsymbol{W}_{r,U}^\top \boldsymbol{B}\boldsymbol{W}_{r,U}$
3. Determine $\boldsymbol{P}_{\ell,\mathcal{F}}, \boldsymbol{P}_{r,\mathcal{F}}$ using 2S-DEIM
4. Compute $\boldsymbol{V}_{\ell,U}^\top \boldsymbol{V}_{\ell,\mathcal{F}}(\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,\mathcal{F}})^{-1}$, $(\boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{P}_{r,\mathcal{F}})^{-1}\boldsymbol{W}_{r,\mathcal{F}}^\top \boldsymbol{W}_{r,U}$, $\boldsymbol{P}_{\ell,\mathcal{F}}^\top \boldsymbol{V}_{\ell,U}$ and $\boldsymbol{W}_{r,U}^\top \boldsymbol{P}_{r,\mathcal{F}}$

Online:

For each $i = 1, \ldots, n_\mathfrak{t}$

(i) Evaluate $\mathfrak{f}(\boldsymbol{Y}_k^{(i-1)}) := \widetilde{\mathcal{F}_k(\boldsymbol{Y}_k^{(i-1)}, \mathfrak{t}_{i-1})}$

(ii) Matrix exponential integrator: solve the matrix equation

$$A_k \Phi + \Phi B_k = e^{hA_k}\mathfrak{f}(\boldsymbol{Y}_k^{(i-1)})e^{hB_k} - \mathfrak{f}(\boldsymbol{Y}_k^{(i-1)})$$

and compute

$$\boldsymbol{Y}_k^{(i)} = e^{hA_k}\boldsymbol{Y}_k^{(i-1)}e^{hB_k} + h\Phi^{(i-1)}$$

A numerical example, the 2D Allen-Cahn equation

$$u_t = \epsilon_1 \Delta u - \frac{1}{\epsilon_2^2}\left(u^3 - u\right), \quad \Omega = [a,b]\times[a,b], \quad t \in [0, T_f], \quad u(x,y,0) = u_0$$

EXAMPLE AC1 ([Song, Jian, Li, 2016])

$$\epsilon_1 = 10^{-2}, \quad \epsilon_2 = 1, \quad a = 0, \quad b = 2\pi, \quad T_f = 5$$

$u_0 = 0.05 \sin x \cos y$ and zero Dirichlet b.c.

EXAMPLE AC2 ([Evans, Spruck, 1991, Ju, Zhang, Zhu, Du, 2015])

$$\epsilon_1 = 1,\ \epsilon_2 \in \{0.01, 0.02, 0.04\}, \quad a = -0.5, \quad b = 0.5, \quad T_f = 0.075$$

$u_0 = \tanh\left(\frac{0.4 - \sqrt{x^2+y^2}}{\sqrt{2}\epsilon_2}\right)$ and periodic b.c.

Problem dimension: $n_x = n_y \equiv n = 1000$

Numerical results. 1

PB.	n_{max}/κ	Ξ	ALGORITHM	$\mathcal{I}$ REFIN	n_s	ν_ℓ/ν_r
AC 1	40/50	U	DYNAMIC	1	8	9/2
			VECTOR	2	9	9
		$\mathcal{F}$	DYNAMIC	1	7	10/3
			VECTOR	2	9	9
AC 2 $\epsilon_2 = 0.04$	400/50	U	DYNAMIC	1	2	15/15
			VECTOR	2	25	25
		$\mathcal{F}$	DYNAMIC	1	3	27/27
			VECTOR	2	40	40
AC 2 $\epsilon_2 = 0.02$	1200/70	U	DYNAMIC	1	3	30/30
			VECTOR	1	28	28
		$\mathcal{F}$	DYNAMIC	1	4	39/39
			VECTOR	2	53	53
AC 2 $\epsilon_2 = 0.01$	5000/150	U	DYNAMIC	1	3	62/62
			VECTOR	1	43	43
		$\mathcal{F}$	DYNAMIC	1	5	73/73
			VECTOR	2	92	92

n_{max}: max # snapshots κ: max allowed POD dim

n_s: employed # snapshots ν_ℓ, ν_r: dim two POD bases

Numerical results. 2

		OFFLINE			ONLINE		
PB.	METHOD	BASIS TIME	DEIM TIME	MEMORY	TIME (n_t)	MEMORY	REL. ERROR
AC 1	DYNAMIC	1.8	0.001	$200n$	0.009 (300)	$24n$	$1 \cdot 10^{-4}$
	VECTOR	0.6	0.228	$18n^2$	0.010 (300)	$18n^2$	$1 \cdot 10^{-4}$
AC 2 0.04	DYNAMIC	0.8	0.005	$200n$	0.010 (300)	$84n$	$3 \cdot 10^{-4}$
	VECTOR	8.4	3.745	$65n^2$	0.020 (300)	$65n^2$	$2 \cdot 10^{-4}$
AC 2 0.02	DYNAMIC	1.8	0.004	$280n$	0.140 (1000)	$138n$	$2 \cdot 10^{-4}$
	VECTOR	14.6	5.273	$81n^2$	0.120 (1000)	$81n^2$	$3 \cdot 10^{-5}$
AC 2 0.01	DYNAMIC	5.3	0.008	$600n$	0.820 (2000)	$270n$	$5 \cdot 10^{-4}$
	VECTOR	46.2	13.820	$135n^2$	0.420 (2000)	$135n^2$	$2 \cdot 10^{-4}$

Conclusions and outlook

- Two-sided matrix-oriented approximation $\boldsymbol{V}_{\ell,U}\boldsymbol{Y}_k(t)\boldsymbol{W}_{r,U}^\top$ is a feasible and effective technique (memory and CPU time saving, structure aware)
- Matrix approach unables combining POD-DEIM with robust exponential integrators
- 3D (tensor) version already available (G. Kirsten, arXiv 2103.04343 (2021))
- Multiparameter version can be foreseen

REFERENCES

- Maria Chiara D'Autilia, Ivonne Sgura and V. Simoncini
Matrix-oriented discretization methods for reaction-diffusion PDEs: comparisons and applications
Computers and Mathematics with Applications, v. 79 (1), 2020, pages 2067-2085.

- Gerhard Kirsten and V. Simoncini
A matrix-oriented POD-DEIM algorithm applied to semilinear matrix differential equations
pp. 1-25, Dipartimento di Matematica, Universita' di Bologna, June 2020. arXiv preprint n. 2006.13289

Conclusions and outlook

- Two-sided matrix-oriented approximation $\boldsymbol{V}_{\ell,U}\boldsymbol{Y}_k(t)\boldsymbol{W}_{r,U}^\top$ is a feasible and effective technique (memory and CPU time saving, structure aware)
- Matrix approach unables combining POD-DEIM with robust exponential integrators
- 3D (tensor) version already available (G. Kirsten, arXiv 2103.04343 (2021))
- Multiparameter version can be foreseen

REFERENCES

- Maria Chiara D'Autilia, Ivonne Sgura and V. Simoncini
Matrix-oriented discretization methods for reaction-diffusion PDEs: comparisons and applications
Computers and Mathematics with Applications, v. 79 (1), 2020, pages 2067-2085.

- Gerhard Kirsten and V. Simoncini
A matrix-oriented POD-DEIM algorithm applied to semilinear matrix differential equations
pp. 1-25, Dipartimento di Matematica, Universita' di Bologna, June 2020. arXiv preprint n. 2006.13289