BAMPHI (Backward-accurate Action of Matrix PHI-functions)

Franco Zivcovich
joint work with Marco Caliari[†] and Fabio Cassini[‡]

21 Juin 2021

Laboratoire Jacques–Louis Lions,
Università degli Studi di Verona[†], Università degli Studi di Trento[‡]

Table of contents

Exponential-type methods

When simulating dynamics,
more accuracy means more efforts.

This is *especially true* when the underlying differential equations are stiff.

Stiff systems are characterized by a wide range of time scales in their evolution.

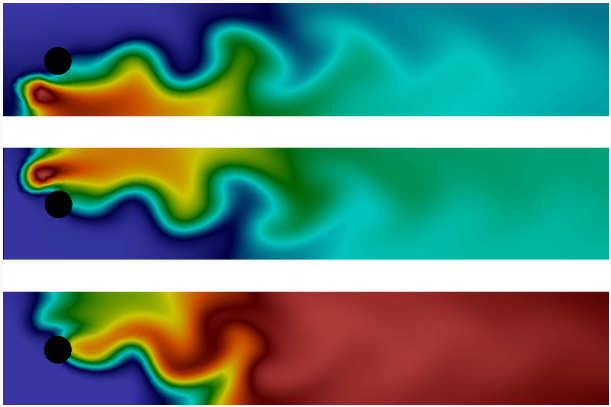


Figure 1 – Final concentrations of components in a chemical reaction described by stiff coupled systems of ADR equations.

Figure 2 – As the relative velocity of the water increases turbulence occurs in the water flow over the hull of a submarine.

Stiff equations arise in a wide range of fields including :

- *fluid dynamics*,
- *electromagnetics*,
- *acoustics*,
- *electrodynamics*,
- *molecular modeling*,
- *celestial mechanics*

. . . but also in *visual computing* for animating the dynamics of cloth, fibers, fluids, or solids, and their interaction with each other.

Figure 3 – Dynamical simulation of human hair during a head shake carried out with an exponential-type method

Write our stiff system of differential equations:

$$u'(t) = \underbrace{\mathbf{A}u(t)}_{\text{stiff guy}} + \underbrace{g(t, u(t))}_{\text{nice guy}}, \quad u(t_0) = u_0 \in \mathbb{C}^N,$$

so that the stiffness is concentrated in the linearity **A**. The exponential-type methods are usually derived from the Duhamel formula

$$u(t_n + k) = \mathrm{e}^{k\mathbf{A}} u(t_n) + \int_0^k \mathrm{e}^{(k-s)\mathbf{A}} g(t_n + s, u(t_n + s)) \mathrm{d}s$$

where the linearity **A** is treated exactly. For this reason, they are particularly suited for the integration of stiff systems of differential equations.

In particular, each exponential-type method differs from the others for how it approximates the integral in

$$u(t_n + k) = \mathrm{e}^{k\mathbf{A}} u(t_n) + \boxed{\int_0^k \mathrm{e}^{(k-s)\mathbf{A}} g(t_n + s, u(t_n + s)) \mathrm{d}s},$$

usually through the action of one or few linear combinations of φ-functions, defined as

$$\varphi_p(x) := \sum_{i=0}^{\infty} \frac{x^i}{(i+p)!} \left(= \int_0^1 \mathrm{e}^{(1-\theta)x} \frac{\theta^{p-1}}{(p-1)!} \mathrm{d}\theta, \ p > 0 \right)$$

Are the φ-functions difficult to compute ?

Linear combinations φ-functions are quite simple to compute. In fact, we can obtain

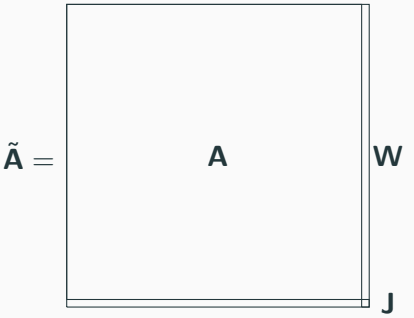
$$\mathrm{e}^{k\mathbf{A}}v_0 + \varphi_1(k\mathbf{A})v_1 + \ldots + \varphi_p(k\mathbf{A})v_p,$$

through the single - slightly larger - action of the matrix exponential:

$$\mathrm{e}^{k\tilde{\mathbf{A}}}\tilde{v},$$

where

$$\tilde{\mathbf{A}} := \begin{pmatrix} \mathbf{A} & \mathbf{W} \\ \mathbf{0} & \mathbf{J} \end{pmatrix}, \ \mathbf{J} := \begin{pmatrix} \mathbf{0} & \mathbf{I}_{p-1} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \ \mathbf{W} := \begin{pmatrix} k^{-p}v_p, k^{-p+1}v_{p-1}, \ldots, k^{-1}v_1 \end{pmatrix}, \ \tilde{v} = \begin{pmatrix} v_0 \\ e_p \end{pmatrix}.$$



BEWARE :

symmetricity (if any) is lost and the sparsity pattern worsen, if not handled correctly $\tilde{\mathbf{A}}$ may become a source of problems.

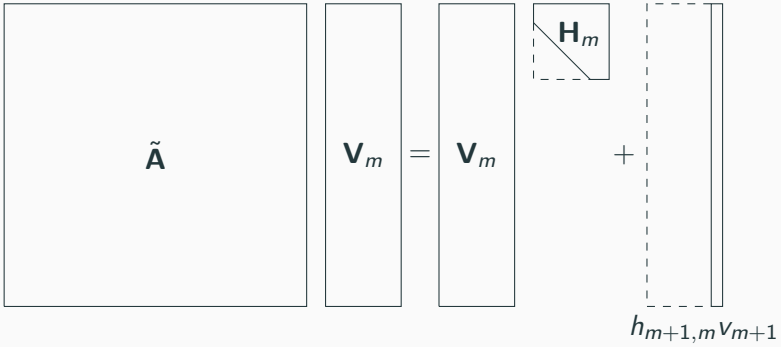
$$\tilde{\mathbf{A}} := \begin{pmatrix} \mathbf{A} & \mathbf{W} \\ \mathbf{0} & \mathbf{J} \end{pmatrix}, \ \mathbf{J} := \begin{pmatrix} \mathbf{0} & \mathbf{I}_{p-1} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \ \mathbf{W} := \left(k^{-p} v_p, k^{-p+1} v_{p-1}, \ldots, k^{-1} v_1 \right), \ \tilde{v} = \begin{pmatrix} v_0 \\ e_p \end{pmatrix}$$

Consequently, exponential-type methods boil down to computing hundreds, thousands or even millions of times the action of very few sorts of matrix exponentials only marginally changing from an integration step to another.

linear combinations of φ-functions

Computing the action of the matrix exponential

Practitioners are *fully sided* with Krylov-based methods due to their simplicity and effectiveness. The idea revolves around Arnoldi decomposition:



where $\mathbf{H}_m$ is a Hessemberg matrix and $\mathbf{V}_m^\star \mathbf{V}_m = \mathbf{I}_m$ with $m \ll N$.

Krylov approximation is obtained by forming

$$\mathrm{e}^{k\tilde{\mathbf{A}}}\tilde{v} \approx \kappa_m(k\tilde{\mathbf{A}}, \tilde{v}) := \|\tilde{v}\|_2 \mathbf{V}_m \mathrm{e}^{k\mathbf{H}_m} e_1$$

where m is chosen large enough so that the approximation is accurate. But Arnoldi decomposition

- is expensive: $O(m^2 N)$ (or $O(mN)$ if *Incomplete Orthogonalization Method* is used);
- requires huge storage space (mN elements) and frequent memory accesses;
- for large m shows stability issues.

Hence it would be best to keep m low (or to overcome Arnoldi procedure right away).

To do so, one sets $v^{(0)} := \tilde{v}$ and adoptes the following sub-stepping strategy :

$$v^{(l+1)} := \kappa_{m_{l+1}}(\tau_{l+1} k\tilde{\mathbf{A}}, v^{(l)}), \quad l = 0, 1, \ldots, s-1$$

where $\tau_1 + \ldots + \tau_s = 1$ and $m_1, \ldots, m_s$ are *reasonably small* positive integers (usually between 10 and 128).

Anyway, this means Krylov methods require to run the (IOM) Arnoldi procedure

at each sub-step (maybe tens or even hundreds)

$\times$ *of each* combination of φ-functions (usually less than ten)

$\times$ *of each* exponential integration step (maybe hundreds, thousands or even millions)

$=$ amounting to a gazillion calls of this *tiring decomposition*.

This problem is known in the community from a long time. An attempt to tackle it traces back to `expmv`, which is based on a Taylor interpolation

$$\mathrm{e}^{k\tilde{\mathbf{A}}}\tilde{v} \approx T_m(k\tilde{\mathbf{A}})\tilde{v} := \sum_{i=0}^{m} \frac{(k\tilde{\mathbf{A}})^i}{i!}\tilde{v}$$

and it comes, for stability reasons, with a sub-stepping strategy too

$$v^{(l+1)} := T_m(s^{-1}k\tilde{\mathbf{A}})v^{(l)}, \quad l = 0, 1, \ldots, s-1$$

where $v^{(0)} := \tilde{v}$.

Now, Taylor interpolation is usually deprecated [1] and referred to as a bad idea for this kind of task, in fact, $\sigma(\tilde{\mathbf{A}})$ is usually scattered and interpolating at the origin may cause:

- a disproportionate amount of matrix-vector products to perform;
- numerical instabilities.

Yet, `expmv` succeded to prove a point compared to Krylov methods, in fact:

- performing Taylor interpolation only requires storing two vectors;
- any Taylor iteration only requires one matrix-vector product.

1. Author's impression formed by talking with some (but not every) practitioners.

What about `bamphi` ?

In a way, `bamphi` is similar to `expmv`, in fact, it is based on a Newton interpolation

$$\mathrm{e}^{k\tilde{\mathbf{A}}}\tilde{v} \approx p_m(k\tilde{\mathbf{A}})\tilde{v} := \sum_{i=0}^{m} d_i \prod_{j=0}^{i-1}(k\tilde{\mathbf{A}} - kx_j\mathbf{I})\tilde{v}$$

which is a mere generalization of Taylor interpolation. As such,

- performing Newton interpolation only requires storing two vectors;
- any Newton iteration only requires one matrix-vector product.

And it comes too, for stability reasons, with a sub-stepping strategy

$$v^{(l+1)} := p_m(s^{-1}k\tilde{\mathbf{A}})v^{(l)}, \quad l = 0, 1, \ldots, s-1$$

where $v^{(0)} := \tilde{v}$.

Now, we need a solid interpolation set lying close to the eigenvalues of $\tilde{\mathbf{A}}$ to overcome `expmv`'s weaknesses. Theorem from [1] says:

"The approximation $\kappa_m(\mathbf{X}, v)$ is mathematically equivalent to $p_m(\mathbf{X})v$ provided $p_m(\,\cdot\,)$ is the polynomial interpolating e^x at the Ritz's values, i.e., $\sigma(\mathbf{H}_m)$".

Hence the idea is to interpolate right at the Ritz's values so that we can emulate Krylov methods *without actually performing* a Krylov method.

In fact, what we are going to do is the following :

1. RUN the IOM Arnoldi decomposition [2] of $\mathbf{A}$ once and compute the set $\sigma(k\mathbf{H}_m)$;
2. COMPUTE the linear combinations of φ-functions $\mathrm{e}^{k\tilde{\mathbf{A}}}\tilde{v}$ interpolating at

$$\sigma(k\mathbf{H}_m) \cup \underbrace{\{0, 0, \ldots, 0\}}_{p \text{ times}};$$

3. STEP AHEAD : IF $\mathbf{A}$ changed since the last integration step *go to 1*, ELSE *go to 2*.

2. we don't even need to store the huge and full matrix $\mathbf{V}_m$, we just need $\mathbf{H}_m$.

This means that `bamphi` requires to run the (IOM) Arnoldi procedure

once and for all at the first call of the routine

= amounting to one call

if the matrix **A** doesn't change from timestep to timestep or

at each exponential integration step (maybe hundreds, thousands or even millions)

= amounting to several (but not extremely many) calls

if the matrix **A** unfortunately does.

Numerical evidence

Routines : for the numerical tests, we compare the two following MATLAB routines:

- `kiops`: is the *state-of-the-art* routine when it comes to Krylov method, it employs IOM Arnoldi decomposition and it is *widely used* for its *strength and simplicity*;
- `bamphi`: is the routine based on Newton interpolation at Ritz's values that we described;

Test 1 : consider the 2-dimensional Advection-Diffusion-Reaction equation

$$\begin{cases} u_t = \varepsilon \Delta u - \alpha(u_x + u_y) + \gamma u(u - \frac{1}{2})(1 - u) \\ u_0 = 256x^2y^2(1-x)^2(1-y)^2 + \frac{3}{10} \end{cases}$$

with $\Omega = [0,1]^2$, $t \in [0, \frac{1}{10}]$, and homogeneous Neumann conditions are set. We employ second-order finite differences discretization with $N_x = 500$ points for each dimension. We take $\varepsilon = \frac{1}{100}$, $\gamma = 100$ and α so that the problem has

- Peclet number equal to 0, see figure (4);
- Peclet number equal to 0.5, see figure (5);
- Peclet number equal to 1, see figure (6);

for time marching we use the *Runge–Kutta exp. integrators* `exprk4s6` and `exprk5s10`.

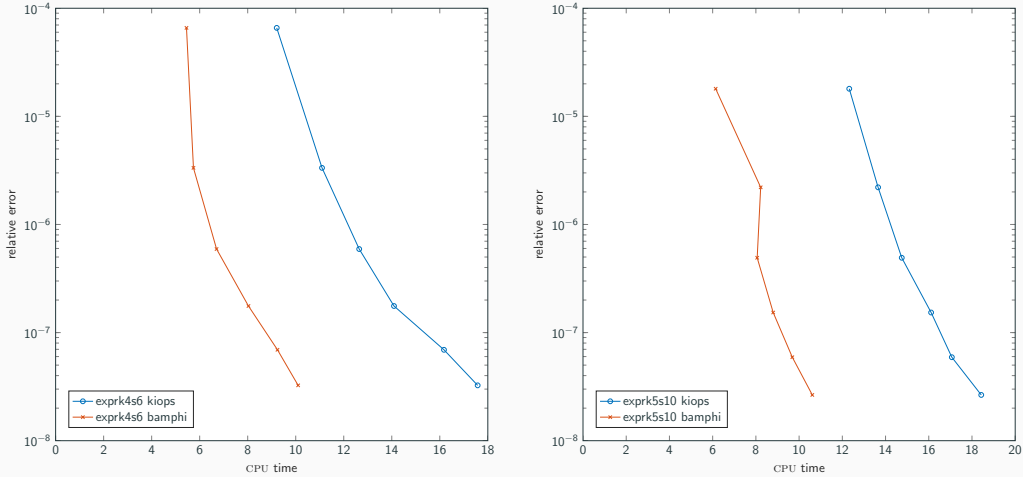


Figure 4 – ADR, Peclet number 0, `exprk4s6`, `exprk5s10`

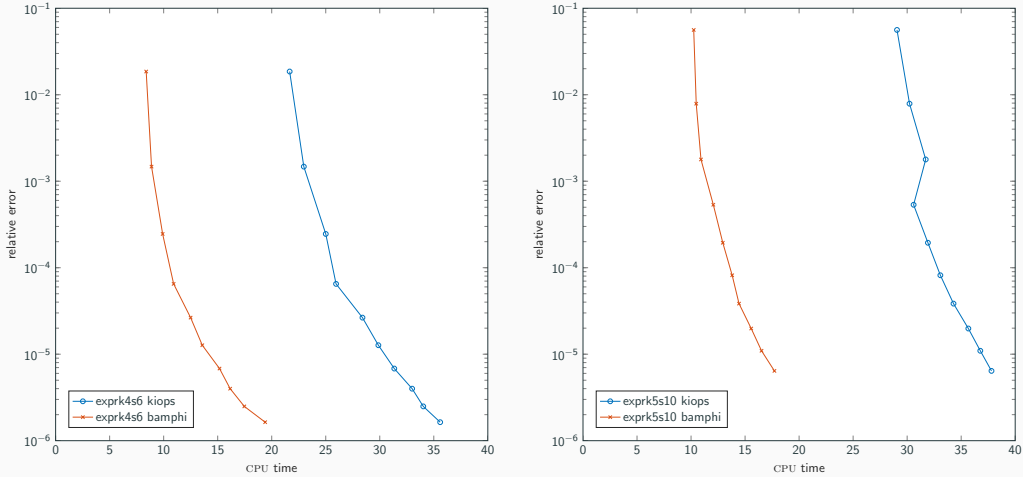


Figure 5 – ADR, Peclet number 0.5, `exprk4s6`, `exprk5s10`

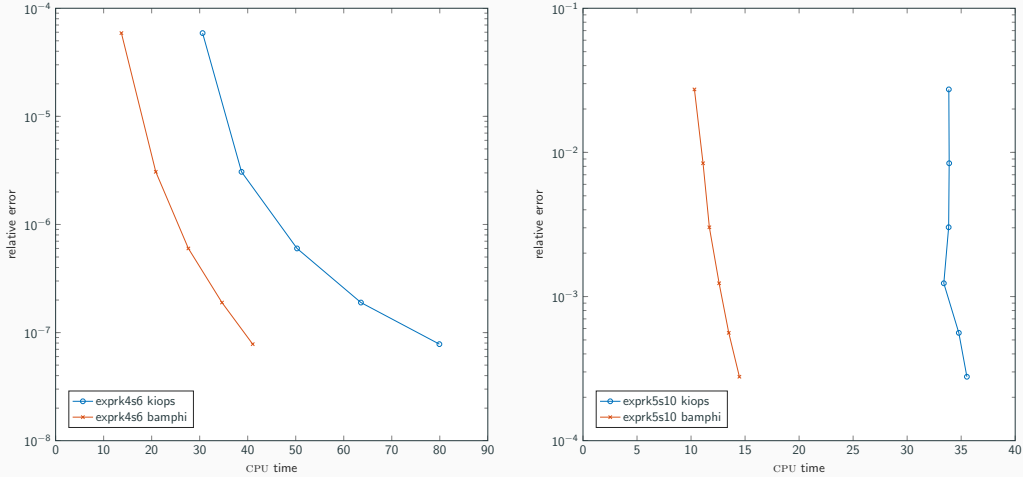


Figure 6 – ADR, Peclet number 1, `exprk4s6`, `exprk5s10`

Test 2 : consider the 1-dimensional Cubic Schrödinger equation

$$\mathrm{i}u_t = -\Delta u + |u|^2 u$$

with $\Omega = [-\pi, \pi]$, $t \in [0, 1]$, $u_0 \in H_0^{3/2}(\Omega)$ and homogeneous Dirichlet conditions. We employ second-order finite differences discretization with $N_x = 500$ points for each dimension.

For time marching we use the *Low-Regularity exp. type integrators* `explr1s2` and `explr2s4`.

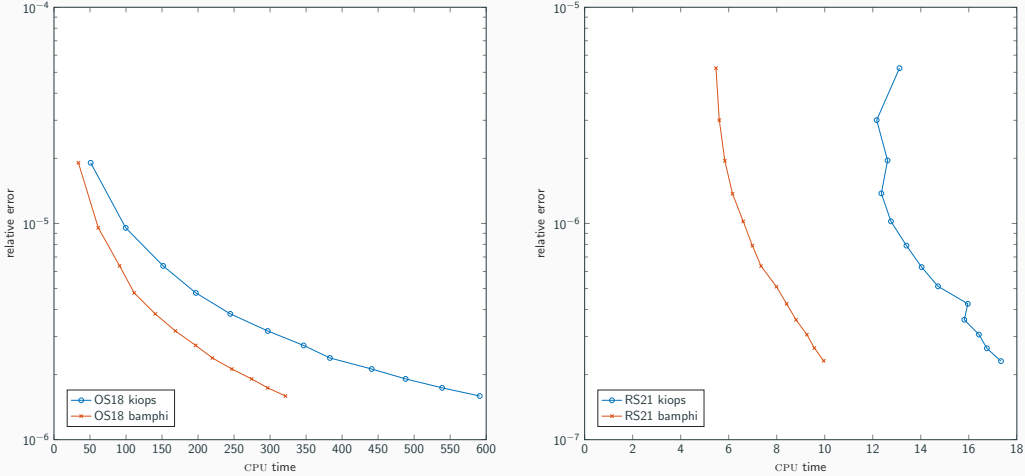


Figure 7 – Cubic Schrödinger equation, `explr1s2`, `explr2s4`

References

- A. H. Al-Mohy and N. J. Higham, *Computing the action of the matrix exponential with an application to exponential integrators*, SIAM J. Sci. Comput. 33 (2) (2011) 488–511.
- M. Caliari, P. Kandolf, A. Ostermann and S. Rainer, *The Leja method revisited : backward error analysis for the matrix exponential*, SIAM J. Sci. Comput. 38 (3) (2016) A1639–A1661.
- S. Gaudreault, G. Rainwater and M. Tokman, *KIOPS : A fast adaptive Krylov subspace solver for exponential integrators*, J. Comput. Phys., 372 (2018), 236-–255,

References

- M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica, 19 (2010), 209-286.
- J. Niesen and W.M. Wright, *A Krylov Subspace Algorithm for Evaluating the phi-Functions appearing in Exponential Integrators*, ACM Trans. Math. Software, 38(3) (2012), Article 22.
- D. Michels, V.T. Luan and M. Tokman, *A Stiffly Accurate Integrator for Elastodynamic Problems*, ACM Trans. on Graphics, Vol. 36, No. 4, Article 116, (2017).

References

 Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM. 29 (1) (1992), 209-228.