# *The dynamical functional particle method for the generalized Sylvester equation*

Dmytryshyn, Andrii and Fasi,
Massimiliano and Gulliksson, Mårten

2021

MIMS EPrint: **2021.4**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

# The dynamical functional particle method for the generalized Sylvester equation*

Andrii Dmytryshyn† Massimiliano Fasi† Mårten Gulliksson†

Recent years have seen a renewal of interest in the study of generalized Sylvester equations, which have come to play a role in a number or applications. Here we consider the solution of such equations by means of the dynamical functional particle method, an iterative technique that relies on the construction and numerical integration of a damped second order dynamical system. We develop a new algorithm for the solution of a large class of these equations, a class that includes, among others, all generalized Sylvester equations with Hermitian positive definite coefficients. Our numerical experiments show that the new implementations outperform existing methods for the solution of generalized Sylvester equations, and can be faster and more accurate than the Bartels–Stewart algorithm for the solution of the Sylvester equation $AX - XB = C$, when $A$ and $B$ are well conditioned and have very different order.

**Key words.** Generalized Sylvester equation, discrete functional particle method, Sylvester equation, Lyapunov equation, Stein equation.

**AMS subject classifications.** 15A24, 65F30

## 1 Introduction

The matrix equation $AX+XB = C$, where $A \in \mathbb{C}^{m\times m}$, $B \in \mathbb{C}^{n\times n}$, and $X, C \in \mathbb{C}^{m\times n}$, is named after Joseph J. Sylvester, who was the first to investigate the homogeneous case [23]. Many similar equations appear in various branches of science and have been extensively investigated. Here we are interested in numerical algorithms for the solution of the so-called "generalized" Sylvester equation

$$\sum_{i=1}^{\ell} A_i X B_i = C, \qquad A_1, \dots, A_\ell \in \mathbb{C}^{m\times m}, \qquad B_1, \dots, B_\ell \in \mathbb{C}^{n\times n}, \qquad X, C \in \mathbb{C}^{m\times n}. \tag{1}$$

Stating conditions for the existence and uniqueness of solutions to (1) in terms of the coefficients $A_i$ and $B_i$ is a difficult problem in the general case, although well-known results are available

*Version of 24 February 2021.

†School of Science and Technology, Örebro University, Örebro, 701 82, Sweden (andrii.dmytryshyn@oru.se, massimiliano.fasi@oru.se, marten.gulliksson@oru.se).

for special cases [19, Ch. 9, 10, and 11]. As suggested by Lancaster [20], by using the Kronecker product one can recast (1) as the $mn \times mn$ linear system

$$M \operatorname{vec}(X) = \operatorname{vec}(C), \qquad M := \sum_{i=1}^{\ell} \big(B_i^T \otimes A_i\big), \tag{2}$$

where $\cdot \otimes \cdot$ and vec denote the Kronecker product and the operator that stacks the columns of a matrix into a column vector, respectively. In the following, we denote by $\operatorname{unvec}_{m,n}$ the operator that reshapes a vector of length $mn$ into an $m \times n$ matrix, so that $\operatorname{unvec}_{m,n}(\operatorname{vec}(X)) = X$ for $X \in \mathbb{C}^{m\times n}$. As it is customary, we will omit the subscript and not specify the dimensions of the reshaped matrix when these are clear from the context. The linear system (2) has a unique solution if and only if $\det M \neq 0$, that is, if the coefficient matrix $M$ is full rank. If $\det M = 0$ then (1) has infinitely many solutions if $M$ has the same rank as the augmented matrix $[M \operatorname{vec}(C)] \in \mathbb{C}^{mn\times(mn+1)}$ and no solution otherwise.

Some special cases of (1) are well known, and have been extensively studied in the literature. The most obvious example is that of a linear system with multiple right-hand sides. Here we will discuss, in particular, the following:

- the two-sided linear equation
$$AXB = C, \tag{3}$$
- the generalized inverse
$$AXA = A, \tag{4}$$
- the continuous-time Lyapunov equation
$$AX + XA^* = C, \tag{5}$$
- the discrete-time Lyapunov equation
$$AXA^* - X = C, \tag{6}$$
- the Sylvester equation
$$AX + XB = C, \tag{7}$$
- the Stein equation
$$AXB + X = C. \tag{8}$$

A couple of decades ago, the generalized equation (1) was considered to be mainly of theoretical interest [17, sect. 16.5]. In recent years, however, it has come to play an important role in a variety of applications [22] such as the numerical study of certain bilinear dynamical systems [2], [12], [16, sect. 2.2], or uncertainty quantification in PDEs with random inputs [7], [8], [21]. In many of these applications, the matrix coefficients show a special structure, and in particular are often Hermitian and positive definite or semi-definite.

Here we adapt the dynamical functional particle method [5] to the numerical solution of (1). The algorithm we develop requires the eigenvalues of the coefficient matrix $M$ in (2) to be real and have same sign. The class of equations that satisfy this requirements is of particular interest as it includes, among others, all the generalized Sylvester equations with Hermitian

positive definite coefficients, which are of great interest in applications. The general technique we develop can be tailored to the special cases (3)–(8), in order to obtain a reduction of the computational cost and therefore of the execution time.

In its basic form, the algorithm involves only matrix-matrix multiplications, and a more refined variant can exploit the availability of routines for the solution of linear systems. This simplicity translates into ease of implementation, which makes the approach very suitable for all those frameworks, such as multiprecision libraries, in which only a few linear algebra kernels are typically available. The technique developed by Bartels and Stewart for the solution of (5) and (7), for example, requires the Schur factorization of the coefficients of the matrix equation [1]. This factorization is computed by means of Francis's algorithm [26], also known as QR algorithm [24], [25], which is one of the most complex algorithms in matrix computation [11, Chap. 13], and is not always available in a multiprecision computing environment [9].

Moreover, the iterative nature of the new method allows for a finer control of the accuracy of the computed solution, as it allows the user to stop the iteration as soon as the target precision has been reached, which would not be possible if the Bartels–Stewart method were used. We remark that ours is not the only iterative method for the solution of generalized Sylvester equations of the form (1). Ding and Chen have developed several gradient-based algorithms for the solution of generalized Sylvester equations [3], which can be adapted to the case of coupled Sylvester equations [4].

The next section summarizes the main features of the dynamical functional particle method, upon which our technique for the solution of the generalized Sylvester equation (1) is built. In view of the equivalent formulation (2), particular emphasis is given to the theoretical aspects specific to the solution of linear systems. In Section 3 we describe how the method can be adapted to the solution of generalized Sylvester equations of the form (1), and discuss how the algorithm can be tailored to the solution of (3)–(8). In Section 4 we compare our implementation of the new algorithms with the built-in MATLAB function `sylvester`, and with an implementation of the gradient-based iterative algorithm of Ding and Chen [3], [4]. Finally in Section 5 we draw our conclusion and outline possible directions for future work.

## 2 The dynamical functional particle method

Edvardsson, Gulliksson, and Persson originally introduced the dynamical functional particle method as a technique for the solution of boundary value problems arising in quantum chemistry [5], but the use of this technique has been investigated for a number of applications, including constrained optimization and linear eigenvalue problems [13], and linear least squares and ill-posed linear systems [14].

Our iterative method for the solution of (1) builds upon the corresponding method for the solution of linear systems of equations [6], which we now briefly recall. Let $G \in \mathbb{C}^{n\times n}$ be a matrix with positive real eigenvalues, let $b \in \mathbb{C}^n$, and let $x : \mathbb{R}^+ \to \mathbb{C}^n$, where $\mathbb{R}^+$ denotes the closed positive real axis, be a function of the dummy time variable $t$. In order to solve the linear system $Gx = b$, consider the second order dynamical system

$$\ddot{x}(t) + \mu\,\dot{x}(t) = b - Gx(t), \qquad \mu > 0, \tag{9}$$

which can equivalently be written as the first order system

$$\begin{cases} \dot{x}(t) = v \\ \dot{v}(t) = -\mu\dot{x}(t) + b - Gx(t). \end{cases} \tag{10}$$

The system (10) can be integrated efficiently in time using the symplectic Euler algorithm [15, Chap. VI], which for $\Delta t > 0$ yields

$$\begin{cases} v_{k+1} = v_k - \mu \Delta t \, v_k \; + \Delta t (b - G x_k) \\ x_{k+1} = x_k + \Delta t \, v_{k+1}. \end{cases} \tag{11}$$

For the initial conditions, $x(0)$ is initialized to a random vector and $\dot{x}(0)$ is set to 0 for simplicity. It is easy to see that if (11) converges, then $\widetilde{x} := x(t_s)$ satisfies $G\widetilde{x} = b$ for some $t_s > 0$.

The convergence of the symplectic scheme (11) is governed by the damping coefficient $\mu > 0$ in the dissipation term and by the discretization step $\Delta t$. The optimal choice of these two parameters is [6]

$$\mu^* = \frac{2\sqrt{\lambda_{\min}(G)\lambda_{\max}(G)}}{\sqrt{\lambda_{\min}(G)} + \sqrt{\lambda_{\max}(G)}}, \qquad \Delta t^* = \frac{2}{\sqrt{\lambda_{\min}(G)} + \sqrt{\lambda_{\max}(G)}}, \tag{12}$$

where $\lambda_{\min}(G)$ and $\lambda_{\max}(G)$ denote the smallest and largest eigenvalue of the matrix $G$, respectively. This method is shown to be stable and convergent for positive definite matrices in [14, Cor. 1].

## 3 Solving the generalized Sylvester equation

The dynamical functional particle method for the solution of the generalized Sylvester equation (1) can readily be formulated by using the alternative expression (2). For efficiency's sake, the coefficient matrix $M$ in (2) is never explicitly computed, and the matrix-vector product $Gx$ in (11) is performed implicitly, as we now explain.

Let $\widetilde{X}_0 \in \mathbb{C}^{mn}$ be a vector with randomly generated entries, and let $\widetilde{Y}_0 \in \mathbb{C}^{mn}$ be vector of length $mn$ with all entries set to 0. Then we can apply the symplectic Euler scheme in (11) to (2) and write

$$\begin{cases} \widetilde{R}_k = \mathrm{vec}(C) - M \, \mathrm{vec}\,(\widetilde{X}_k), \\ \widetilde{Y}_{k+1} = \widetilde{Y}_k + \Delta t \cdot (\widetilde{R}_k - \mu \widetilde{Y}_k), \\ \widetilde{X}_{k+1} = \widetilde{X}_k + \Delta t \cdot \widetilde{Y}_{k+1}, \end{cases} \tag{13}$$

where the matrix $M$ is the sum of Kronecker products defined in (2). The approximate solution to (1) at step $k'$ will be $\mathrm{unvec}_{m,n}(\widetilde{X}_{k'})$.

The iterative scheme (13) is not practical, as at each step it requires the evaluation of the matrix-vector product $M \, \mathrm{vec}(X)$. As $M$ is a matrix of order $mn$, this matrix computation requires $2m^2n^2 + o(m^2n^2)$ floating-point operations (flops), and can become unduly expensive even for moderate values of $m$ and $n$. The computation of the residual $R_k$, however, can equivalently be written

$$\widetilde{R}_k = \mathrm{vec}\left( C - \sum_{i=1}^{\ell} A_i X_k B_i \right), \tag{14}$$

so that one step of the iteration now requires only $2\ell(m^2n + mn^2) + o\big(\ell(m^2n + mn^2)\big)$ flops.

In view of this observation, we can rewrite (13) in the more natural form

$$\begin{cases} R_k = C - \sum_{i=1}^{\ell} A_i X_k B_i, \\ Y_{k+1} = Y_k + \Delta t \cdot \big(R_k - \mu Y_k\big), \\ X_{k+1} = X_k + \Delta t \cdot Y_{k+1}, \end{cases} \tag{15}$$

| Matrix equation | Coefficient $M$ | $\lambda_{\min}(M)$ | $\lambda_{\max}(M)$ |
|---|---|---|---|
| $AXB = C$ | $B^T \otimes A$ | $\lambda_{\min}(A)\lambda_{\min}(B)$ | $\lambda_{\max}(A)\lambda_{\max}(B)$ |
| $AXA = C$ | $A^T \otimes A$ | $\lambda_{\min}(A)^2$ | $\lambda_{\max}(A)^2$ |
| $AX + XA^* = C$ | $I_m \otimes A + \overline{A} \otimes I_m$ | $2\lambda_{\min}(A)$ | $2\lambda_{\max}(A)$ |
| $AXA^* - X = C$ | $\overline{A} \otimes A - I_m \otimes I_m$ | $\lambda_{\min}(A)^2 - 1$ | $\lambda_{\max}(A)^2 - 1$ |
| $AX + XB = C$ | $I_m \otimes A + B^T \otimes I_m$ | $\lambda_{\min}(A) + \lambda_{\min}(B)$ | $\lambda_{\max}(A) + \lambda_{\max}(B)$ |
| $AXB + X = C$ | $B^T \otimes A + I_m \otimes I_m$ | $\lambda_{\min}(A)\lambda_{\min}(B) + 1$ | $\lambda_{\max}(A)\lambda_{\max}(B) + 1$ |

Table 1: The spectrum of the Kronecker form of some special cases of the generalized Sylvester equation in (1).

where $X_k, Y_k, R_k \in \mathbb{C}^{m\times n}$.

In order to obtain the optimal damping and time step for this scheme we still need an efficient way of estimating the extreme eigenvalues of $M$ in (2), which are real by assumption. In the most general case, the most efficient way of estimating $\lambda_{\max}(M)$ is the power method [27, Ch. 10], an iteration that approximates an eigenvector corresponding to the largest eigenvalue in absolute value. At each step, this algorithm requires only one matrix-vector product, which can be implemented implicitly without ever forming the matrix $M$ explicitly. The power method could also be used to efficiently estimate $\lambda_{\min}(M) = \lambda_{\max}(M^{-1})$ if a routine for solving linear system having $M$ as coefficient is available. To the best of our knowledge, no such routine exists for the general matrix $M$ in (2) for $\ell$ greater than 2, but the extreme eigenvalues of $M$ can be computed efficiently for some special cases, as we now explain.

For the one-term and two-terms linear equations in (3)–(8) we can use the following well-know result.

**Proposition 3.1.** *[18, Theorem 4.2.12 and Exercise 19, p. 251] Let $\lambda_1, \ldots, \lambda_m$ be the eigenvalues of $A \in \mathbb{C}^{m\times m}$ and $\xi_1, \ldots, \xi_n$ be the eigenvalues of $B \in \mathbb{C}^{n\times n}$. Then for $i = 1, \ldots, n$ and $j = 1, \ldots, m$, the $mn$ eigenvalues of $A \otimes B$ and $A \otimes I_n + I_m \otimes B$ have the form $\lambda_i\xi_j$ and $\lambda_i + \xi_j$, respectively.*

Combining this result with the fact that a matrix and its transpose have the same characteristic polynomial and thus the same eigenvalues, allows us to obtain formulae for the extreme eigenvalues of $M$ which only require knowledge of the extreme eigenvalues of the coefficient matrices appearing on the left-hand side of the matrix equation. We summarize the results for the special cases of interest in Table 1.

The method described in this section inherits the convergence behavior and the stability of the algorithm for linear systems described in Section 2.

## 4 Numerical experiments

Now we compare the algorithm discussed in Section 3 with existing methods for the solution of Sylvester and generalized Sylvester equations. The results in this section were obtained by running the experiments in MATLAB 9.8.0 (R2020a) on a machine equipped with an Intel I5-5287U running at 2.9 GHz and 16 GiB of RAM. We compared the following five codes.

- `gs_kron` solves the linear system in (2) by explicitly constructing the matrix $M$ and then using the MATLAB backslash operator.

- `gs_dfpm_opt` solves (1) by using the dynamical functional particle method described in Section 3 with optimal damping and time step computed according to (12) for $G = M$. The quantities $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$ are computed by constructing the matrix $M$ explicitly, with the exception of the special cases in Table 1.

- `gs_dfpm_app` solves (1) by using the dynamical functional particle method described in Section 3 with parameters chosen according to (12) for $G = M$, but using the estimates

$$\lambda_{\min}(A) \approx \sum_{i=1}^{\ell} \lambda_{\min}(B_i^T \otimes A_i), \qquad \lambda_{\max}(A) \approx \sum_{i=1}^{\ell} \lambda_{\max}(B_i^T \otimes A_i),$$

  which do not require the explicit computation of the matrix $M$.

- `gs_gbia` solves (1) by using the gradient based iterative algorithms for solving generalized Sylvester matrix equations developed by Ding and Chen [3], [4].

- `sylvester` solves the continuous-time Lyapunov equation (5) and the Sylvester equation (7) by means of the MATLAB function `sylvester`, which implements the algorithm of Bartels and Stewart [1].

For the iterative algorithms, we set the maximum number of iterations to 50 000 and keep iterating until the 1-norm difference between two successive iterates is below $2^3u$, where $u = 2^{-53} \approx 1.11 \times 10^{-16}$ denotes the unit roundoff of binary64 floating-point arithmetic.

In our tests, the superscript notation $A^{(\eta)} \in \mathbb{R}^{m\times m}$ denotes the real non-symmetric matrix generated as

$$A^{(\eta)} = PD^{(\eta)}P^{-1}, \tag{16}$$

where the $P \in \mathbb{R}^{m\times m}$ is such that $\kappa_2(P) = 2$ and is generated using the `randsvdfast` function [10], whereas $D_\eta$ is a diagonal matrix with extreme eigenvalues $\sqrt{\eta}^{-1}$ and $\sqrt{\eta}$ and remaining diagonal elements uniformly distributed in $[\sqrt{\eta}^{-1}, \sqrt{\eta}]$. This choice of $P$ and $D^{(\eta)}$ ensures that $\kappa_2(A^{(\eta)}) \leq 4\eta$, and in practice provide a matrix $A^{(\eta)}$ such that $\kappa_2(A^{(\eta)}) \approx \eta$.

### 4.1 The Sylvester equation

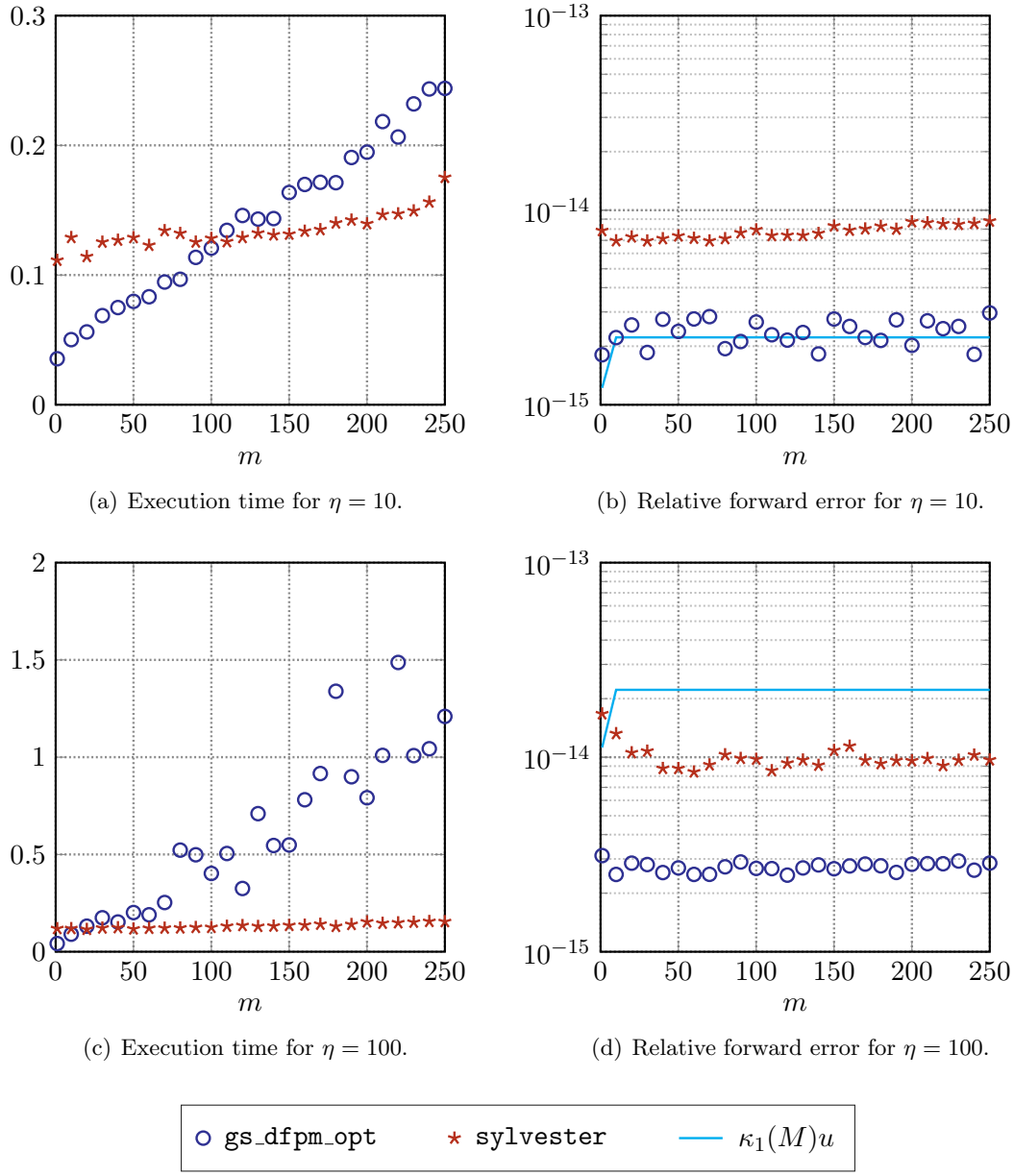
In this first experiment we consider the solution of the Sylvester equation

$$A^{(\eta)}X - XB^{(\eta)} = C, \tag{17}$$

where $A^{(\eta)} \in \mathbb{R}^{m\times m}$ and $B^{(\eta)} \in \mathbb{R}^{n\times n}$ are as in (16) and $C \in \mathbb{R}^{m\times n}$ is generated using a matrix $X \in \mathbb{R}^{m\times n}$ with entries from the Gaussian distribution.

In Figure 1 we report the execution time required by `sylvester` and `gs_dfpm_opt` to solve (17) and the forward error of the solutions the two algorithms computed. In the plots we consider two moderate values of $\eta$, namely 10 (top row) and 100 (bottom row), and we fix $n = 500$ and allow $m$ to vary. For both values of $\eta$, `gs_dfpm_opt` is more accurate than `sylvester` for all the test matrices. The forward error of `gs_dfpm_opt` is of the order of $\kappa_1(M)u$ for $\eta = 10$ and about one order of magnitude smaller than the accuracy reference for $\eta = 100$.

We now discuss the timings of the two algorithms. As `gs_dfpm_opt` is an iterative algorithm, its execution time depends not only on the total number of iterations the algorithm requires to converge, but also on the computational cost of each iteration. As discussed above, the most




(a) Execution time for $\eta = 10$.

(b) Relative forward error for $\eta = 10$.

(c) Execution time for $\eta = 100$.

(d) Relative forward error for $\eta = 100$.

Figure 1: Left: execution time, in seconds, required by `gs_dfpm_opt` and `sylvester` to solve the matrix equation in (17) for $n = 500$. Right: relative forward error of the computed solution.

expensive operation of each iteration of `gs_dfpm_opt` is the computation of the residual, thus each iteration asymptotically requires $2(m^2n + mn^2)$ flops. Therefore, we should expect the timings of this algorithm to grow with the order of $A^{(\eta)}$ and $B^{(\eta)}$, which is confirmed by the plots in the left column of Figure 1.

The total number of iteration required by `gs_dfpm_opt` is proportional to the conditioning of the matrix $M$ in (2), which in turn roughly depend on the parameter $\eta$. This is consistent with previous findings in the literature on the dynamical functional particle method for the

solution of linear systems with positive definite coefficients. In our experiments, the algorithm required 44 and 64 iterations for the matrices in the top row and between 201 and 948 for those in the bottom row.

In our experimental setup, for $\eta = 10$ `gs_dfpm_opt` is faster than `sylvester` when the order of $A^{(\eta)}$ is at most 20% of that of $B^{(\eta)}$, a percentage that reduces to about 5% when $\eta = 100$. For larger values of $\eta$, `gs_dfpm_opt` is typically slower but still more accurate than `sylvester`. Similar results obtained for larger values of $n$ suggest that `gs_dfpm_opt` is typically marginally but consistently more accurate than `sylvester`, but is faster than the latter only when the coefficients of the matrix equation (17) are well conditioned and differ considerably in size.

## 4.2 The generalized Sylvester equation

Now we consider the solution of the more general matrix equation

$$\sum_{i=1}^{5} A_i^{(\eta)} X B_i^{(\eta)} = C. \tag{18}$$

In our experiments, the $m \times m$ matrices $A_1^{(\eta)}, \ldots, A_5^{(\eta)}$ are simultaneously diagonalizable, and so are the $m \times m$ matrices $B_1^{(\eta)}, \ldots, B_5^{(\eta)}$, although the eigenvectors of $A_i^{(\eta)}$ and $B_i^{(\eta)}$ are in general different. We resort to this technique to ensure that the matrix $M$ in (2) has positive real eigenvalues. As in the previous experiment, the matrix $C \in \mathbb{R}^{m \times m}$ is computed by using a matrix $X \in \mathbb{R}^{m \times m}$ with entries from the Gaussian distribution.

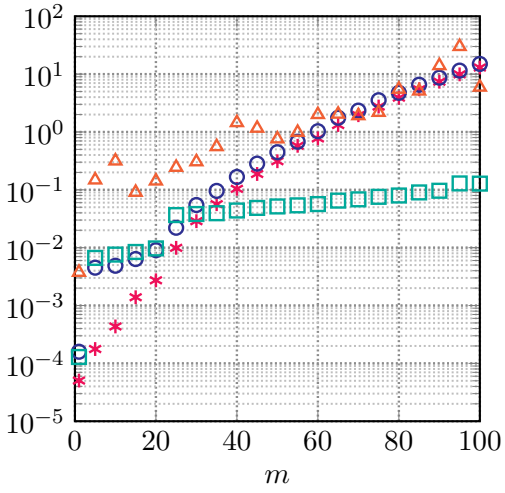
In Figure 2 we compare the performance and accuracy of `gs_kron`, `gs_dfpm_opt`, `gs_dfpm_app`, and `gs_gbia` for the solution of (18) as $m$ varies. As we are mainly interested in well-conditioned matrices, as in the previous experiment we consider the two cases $\eta = 10$ and $\eta = 100$.

Our results suggest that `gs_kron` is the most accurate of the four algorithms, and is the only one that achieves a forward error of the magnitude of $\kappa_1(M)u$. The two implementations based on the dynamical functional particle method achieve a similar level of accuracy, whereas `gs_gbia` is always the least accurate of the algorithms we test, and for three of our test matrices it fails to satisfy our stopping criterion within 50 000 iterations.

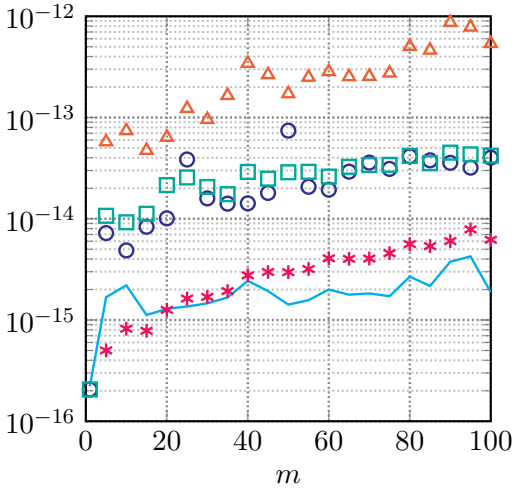
We now compare the four algorithms in terms of execution time. For matrix equations of small size, `gs_kron` is typically the most efficient algorithm, followed by `gs_dfpm_opt`, `gs_dfpm_app`, and finally `gs_gbia`. For these small matrices, the approximation of the extreme eigenvalues of $M$ in (2) is inexpensive, and the cost of the three iterative algorithms depends mostly on the number of iterations that are necessary to achieve convergence. As `gs_dfpm_opt` requires considerably fewer iterations than the other two methods, it is the fastest for $m$ below 40.

We note that, in our implementations, `gs_dfpm_opt` cannot achieve an execution time lower than that of `gs_kron`. In fact, estimating $\lambda_{\min}(M)$ requires the solution of at least one—but typically several—linear systems with coefficient matrix $M$. Therefore the computation that `gs_kron` performs is, in a sense, just a pre-processing step for `gs_dfpm_opt`. In order for this method to be competitive, an alternative technique for estimating the smallest eigenvalue of $M$ is necessary. We do not investigate this further, as the choice of said technique is likely to depend on the problem being solved and on a priori knowledge of the properties of the coefficient matrices.

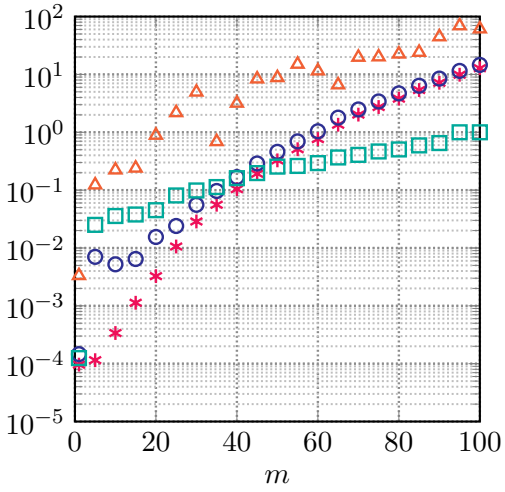
As the size of the matrix coefficients of the equations grows, estimating the extreme eigenvalues of $M$ becomes more expensive, and the lower number of iterations that the optimal



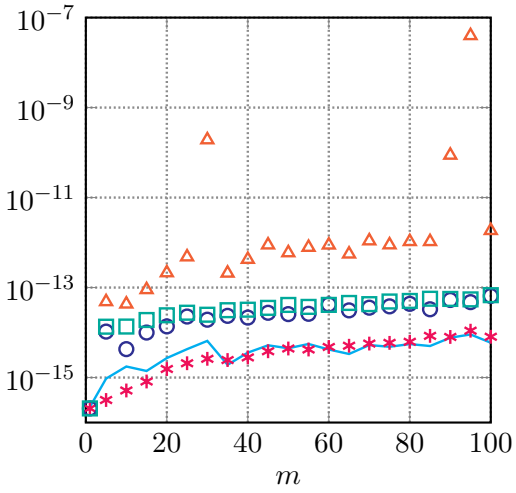

(a) Execution time for $\eta = 10$.




(b) Relative forward error for $\eta = 10$.




(c) Execution time for $\eta = 100$.




(d) Relative forward error for $\eta = 100$.

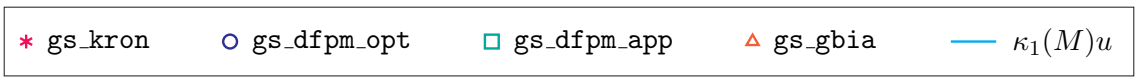



Figure 2: Left: execution time, in seconds, required by `gs_kron`, `gs_dfpm_opt`, `gs_dfpm_app` and `gs_gbia` to solve the matrix equation in (18) $m = n$ between 1 and 100. Right: relative forward error of the computed solution.

choice of parameters produces is not sufficient to offset the time spent estimating $\lambda_{\min}(M)$. Thus for larger matrices the crude choice of parameters of `gs_dfpm_app` pays off, leading to an execution time two order of magnitudes smaller for $m$ as small as 100.

## 5 Conclusion

We have developed a family of algorithms for the solution of a class of generalized Sylvester equations in the form (1), and we have explained how these methods can be tailored to tackle some special cases of particular importance in applications. The new techniques build upon the dynamical functional particle method for the solution of linear systems, and exploit the equivalence between the two formulations (1) and (2).

Numerical results show that our implementations are typically capable of outperforming existing methods for the solution of (1) in terms of both accuracy and execution speed. In order to show the potential of our new techniques, we compared the algorithm for the Sylvester equation (7) to the built-in MATLAB function `sylvester`. We found that if the matrix coefficients on the left-hand side are sufficiently well conditioned and have very different size, then our implementation of the discrete functional particle method can outperform `sylvester` in terms of both accuracy and speed.

As the dynamical functional particle method has been successfully applied to nonlinear optimization problems, it is natural to ask whether similar techniques for the solution of nonlinear matrix equations can be derived. This nontrivial problem will be the subject of future work.

## Acknowledgements

The work of Massimiliano Fasi was supported by the Wenner-Gren Foundations [grant UPD2019-0067].

## References

[1] Richard H. Bartels and George W. Stewart, *Algorithm 432: Solution of the matrix equation $AX + XB = C$*, Comm. ACM, 15 (1972), pp. 820–826.

[2] Peter Benner and Tobias Damm, *Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems*, SIAM J. Control Optim., 49 (2011), p. 686–711.

[3] Feng Ding and Tongwen Chen, *Gradient based iterative algorithms for solving a class of matrix equations*, IEEE Trans. Automat. Control, 50 (2005), p. 1216–1221.

[4] Feng Ding and Tongwen Chen, *On iterative solutions of general coupled matrix equations*, SIAM J. Control Optim., 44 (2006), p. 2269–2284.

[5] Sverker Edvardsson, Mårten Gulliksson, and Johan Persson, *The dynamical functional particle method: An approach for boundary value problems*, J. Appl. Mech., 79 (2012).

[6] Sverker Edvardsson, Magnus Neuman, Per Edström, and Håkan Olin, *Solving equations through particle dynamics*, Comput. Phys. Comm., 197 (2015), p. 169–181.

[7] Howard C. Elman, Darran G. Furnival, and Catherine E. Powell, *$H(div)$ preconditioning for a mixed finite element formulation of the diffusion problem with random data*, Math. Comp., 79 (2009), p. 733–760.

[8] Oliver G. Ernst, Catherine E. Powell, David J. Silvester, and Elisabeth Ullmann, *Efficient solvers for a linear stochastic Galerkin mixed formulation of diffusion problems with random data*, SIAM J. Sci. Comput., 31 (2009), p. 1424–1447.

[9] Massimiliano Fasi and Nicholas J. Higham, *Multiprecision algorithms for computing the matrix logarithm*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 472–491.

[10] ——, *Generating extreme-scale matrices with specified singular values or condition numbers*, MIMS EPrint 2020.8, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Mar. 2020. Revised October 2020. To appear in SIAM J. Sci. Comput.

[11] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, USA, 4th ed., 2013.

[12] W. Steven Gray and Joseph Mesko, *Energy functions and algebraic Gramians for bilinear systems*, IFAC P. Vol., 31 (1998), p. 101–106.

[13] Mårten Gulliksson, *The discrete dynamical functional particle method for solving constrained optimization problems*, Dolomites Res. Notes Approx., 10 (2017), p. 6–12.

[14] Mårten Gulliksson, Magnus Ögren, Anna Oleynik, and Ye Zhang, *Damped dynamical systems for solving equations and optimization problems*, Handbook of the Mathematics of the Arts and Sciences, (2018), p. 1–44.

[15] Ernst Hairer, Gerhard Wanner, and Christian Lubich, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer Series in Computational Mathematics, Springer-Verlag, 2006.

[16] Carsten Hartmann, Boris Schäfer-Bung, and Anastasia Thöns-Zueva, *Balanced averaging of bilinear systems with applications to stochastic control*, SIAM J. Control Optim., 51 (2013), p. 2356–2378.

[17] Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second ed., 2002.

[18] Roger A. Horn and Charles R. Johnson, *Topics in Matrix Analysis*, 1991.

[19] Mihail Konstantinov, Da-Wei Gu, Volker Mehrmann, and Petko Petkov, *Perturbation Theory for Matrix Equations*, vol. 9 of Studies in Computational Mathematics, Elsevier, Amsterdam, The Netherlands, 2003.

[20] Peter Lancaster, *Explicit solutions of linear matrix equations*, SIAM Rev., 12 (1970), p. 544–566.

[21] Catherine E. Powell, David Silvester, and Valeria Simoncini, *An efficient reduced basis solver for stochastic Galerkin matrix equations*, SIAM J. Sci. Comput., 39 (2017), p. A141–A163.

[22] Valeria Simoncini, *Computational methods for linear matrix equations*, SIAM Rev., 58 (2016), p. 377–441.

[23] James Joseph Sylvester, *Sur l'equations en matrices $px = xq$*, C. R. Acad. Sci. Paris, 99 (1884), pp. 67–71.

[24] David Scott Watkins, *Understanding the QR algorithm*, SIAM Rev., 24 (1982), pp. 427–440.

[25] ——, *The QR algorithm revisited*, SIAM Rev., 50 (2008), pp. 133–145.

[26] ——, *Francis's algorithm*, Amer. Math. Monthly, 118 (2011), p. 387.

[27] James Hardy Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965.