A frequency-independent solver for systems of linear ordinary differential equations

Tony Hu[a], James Bremer[a,*]

[a]*Department of Mathematics, University of Toronto*

Abstract

When a system of first order linear ordinary differential equations has eigenvalues of large magnitude, its solutions generally exhibit complicated behaviour, such as high-frequency oscillations, rapid growth or rapid decay. The cost of representing such solutions using standard techniques grows with the magnitudes of the eigenvalues. As a consequence, the running times of standard solvers for ordinary differential equations also grow with the size of these eigenvalues. The solutions of scalar equations with slowly-varying coefficients, however, can be represented via slowly-varying phase functions at a cost which is bounded independent of the magnitudes of the eigenvalues of the corresponding coefficient matrix. Here we couple an existing solver for scalar equations which exploits this observation with a well-known technique for transforming a system of linear ordinary differential equations into scalar form. The result is a method for solving a large class of systems of linear ordinary differential equations in time independent of the magnitudes of the eigenvalues of their coefficient matrices. We discuss the results of numerical experiments demonstrating the properties of our algorithm.

Keywords: fast algorithms, ordinary differential equations

1. Introduction

A system of n first-order linear ordinary differential equations in n unknowns

$$\boldsymbol{y}'(t) = A(t)\boldsymbol{y}(t), \tag{1}$$

where $A : [a, b] \rightarrow \mathbb{C}^{n\times n}$ is a smooth matrix-valued function, is said to be nondegenerate on $[a, b]$ provided $A(t)$ admits n distinct eigenvalues $\lambda_1(t), \ldots, \lambda_n(t)$ for each t in (a, b). The complexity of the solutions of such a system generally increases with the magnitudes of the $\lambda_1(t), \ldots, \lambda_n(t)$. This can be readily seen from standard results in asymptotic analysis. The simplest case is when all of the eigenvalues are of large magnitude throughout $[a, b]$. In this event, it is convenient to write $A(t) = \omega A_0(t)$, where ω is chosen so that the uniform

*Corresponding author

Email address: bremer@math.toronto.edu (James Bremer)

norms of the eigenvalues $\mu_1(t), \ldots, \mu_n(t)$ of $A_0(t)$ over the interval $]a, b]$ are bounded independent of ω. Then, it is well known that there exists a fundamental solution $\Psi(t)$ of (1) such that, for all $t \in [a, b]$,

$$\|\Psi(t) - \Psi_0(t)\| = \|\Psi_0(t)\| \, \mathcal{O}\left(\frac{1}{\omega}\right) \text{ as } \omega \to \infty, \tag{2}$$

where $\|\cdot\|$ is any matrix norm on $\mathbb{C}^{n\times n}$ and $\Psi_0(t)$ is of the form

$$\Psi_0(t) = Y(t) \exp \begin{pmatrix} \omega \int_0^t \mu_1(s)\, ds & & & \\ & \omega \int_0^t \mu_2(s)\, ds & & \\ & & \ddots & \\ & & & \omega \int_0^t \mu_n(s)\, ds \end{pmatrix}. \tag{3}$$

Each of the functions

$$f_j(t) = \exp\left(\omega \int_0^t \mu_j(s)\ ds\right) \tag{4}$$

is rapidly-varying in the sense that its derivatives increase quickly with ω. Indeed, for every $j = 1, \ldots, n$, positive integer i and $t \in [a, b]$, we have

$$\left|\frac{f_j^{(i)}(t)}{f_j(t)}\right| = \mathcal{O}\left(\omega^i\right) \text{ as } \omega \to \infty. \tag{5}$$

Among other things, this implies that the cost to represent each f_j to a fixed accuracy using a polynomial expansion grows linearly with ω. It then follows from (3) and (4) that every solution of (1) is rapidly varying in this sense as well. Because almost all standard ODE solvers use polynomial expansions of the solutions either implicitly or explicitly, this means that the running times of standard ODE solvers also increase linearly with ω when applied to (1). Similar results can be established in the case in which only certain eigenvalues of $A(t)$ are of large magnitude, although, of course, in such cases some solutions of (1) are rapidly varying while others are not. We refer the reader to Chapter 7 of [22] for a discussion of methods for the asymptotic approximation of solutions of systems of linear ordinary differential equations that includes a proof of the estimate (2).

Fortunately, a large class of scalar ordinary differential equations of the form

$$y^{(n)}(t) + q_{n-1}(t)y^{(n-1)}(t) + \cdots + q_1(t)y'(t) + q_0(t)y(t) = 0 \tag{6}$$

admit phase functions whose complexity depends on that of the coefficients $q_0, \ldots, q_{n-1}$ but not the magnitudes of the eigenvalues $\lambda_1(t), \ldots, \lambda_n(t)$ of the corresponding coefficient

matrix

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -q_0(t) & -q_1(t) & -q_2(t) & \cdots & -q_{n-2}(t) & -q_{n-1}(t) \end{pmatrix}. \tag{7}$$

Indeed, if the $q_0, \ldots, q_{n-1}$ are smooth and slowly varying on an interval I, and $\lambda_1(t), \ldots, \lambda_n(t)$ are distinct for all t in I, then it is possible to find smooth, slowly-varying functions $\psi_1, \ldots, \psi_n \colon I \to \mathbb{C}$ such that

$$u_j(t) = \exp\left(\psi_j(t)\right), \quad j = 1, \ldots, n, \tag{8}$$

constitute a basis in the space of solutions of (6) given on the interval I. That slowly-varying phase functions exist under mild conditions on the coefficient q_j, at least in an asymptotic sense, has long been known. In fact, this observation underlies the WKB method and almost all other techniques for the asymptotic approximation of solutions of ordinary differential equations in the high-frequency regime (see, for instance, [16], [22] and [21, 19, 20]). A careful proof of the existence of slowly-varying phase functions for second order differential equations, which can be extended to higher order scalar equations, is given in [7].

Here, we couple the algorithm of [1] with a standard technique for reducing systems of linear ordinary differential equations to scalar form. The result is a solver for a large class of linear systems of differential equations which runs in time independent of the magnitudes of the eigenvalues of (1). We refer to our method as a "frequency-independent" solver because, in most cases of interest, it is the imaginary parts of the eigenvalues of (1) which are of large magnitude. Indeed, when the coefficient matrix $A(t)$ has eigenvalues whose real parts are large, some of the vector-valued solutions of (1) have components which behave like combinations of rapidly increasing or decreasing exponential functions and most initial and boundary value problems for (1) are numerically intractable. We discuss this issue in more detail in Section 4 of this paper. It is clear from (5) that the condition number of evaluation of the f_j increases with the the magnitudes of the eigenvalues of the coefficient matrix $A(t)$ (see, for instance, Chapter 1 of [9] for a definition of the condition number of evaluation of a function). It follows from this and the asymptotic estimate (2) that the solutions of (1) will have the same property. As a consequence, we expect the accuracy of any numerical solver to decrease as the eigenvalues of $A(t)$ increase in magnitude, whether those eigenvalues have large real parts or not. Our algorithm is no exception, and the accuracy with which it computes the solutions of (1) decreases as the magnitudes of the eigenvalues of the coefficient matrix increase.

As in [5], we focus here on the case in which the system is nondegenerate on the interior

of the domain $[a, b]$ of the differential equation. However, our algorithm can easily be extended to systems which are nondegenerate on an interval $[a, b]$ except at a finite number of turning points by applying it repeatedly, on a collection of subintervals of $[a, b]$ as in [6]. Moreover, by combining our algorithm with an appropriate Levin method such as [13], inhomogenous equations of the form

$$\boldsymbol{y}'(t) = A(t)\boldsymbol{y}(t) + \boldsymbol{f}(t) \tag{9}$$

can be treated efficiently. This topic is beyond the scope of the present article, but we refer the interested reader to [18] in which a similar approach is used to efficiently solve inhomogeneous second order linear ordinary differential equations of the form

$$y''(t) + q(t)y(t) = f(t), \quad a < t < b, \tag{10}$$

in the event that the coefficient q is of large magnitude.

There is, though, one significant difficulty with our method. The transformation matrices we form to convert the system (1) to the scalar form (7) can be ill-conditioned. By the condition number of a matrix-valued function $B(t) : [a, b] \to \mathbb{C}^{n\times n}$, we mean the quantity

$$\sup_{a\le t\le b} \|B(t)\|\|B^{-1}(t)\|, \tag{11}$$

where, for the sake of concreteness, we choose $\|\cdot\|$ to be the Frobenius norm defined for a matrix C in $\mathbb{C}^{n\times n}$ with entries C_{ij} via formula

$$\|C\| = \sqrt{\sum_{i,j=1}^{n} |C_{ij}|^2}. \tag{12}$$

The condition numbers of the transformation matrices we form tend to increase with the dimension of the system under consideration. As a consequence, the obtainable accuracy of our method generally decreases as the dimension of the system increases. In the case of systems of two and three dimensions, this effect appears to be quite mild. In our experiments, we were able to achieve between 8 and 12 digits of when solving such systems (where, as expected, the accuracy decreased as the magnitude of the eigenvalues of $A(t)$ grew). However, for a larger system of four equations, the accuracy of our method varied between 5 and 9 digits. We expect this effect to become more pronounced for systems of five or more equations, and this substantially limits the applicability of our method.

Despite the challenge of constructing well-conditioned transformations which convert a system to scalar form, the algorithm of this paper applies to a large class of systems of linear ordinary differential equations of modest orders and our method appears to be the first high-accuracy solver for such problems which runs in time independent of frequency. We view this work as a step toward developing robust frequency-independent solvers for systems of ordinary differential equations, and we discuss several obvious directions for further development in the final section of this paper.

Because the cost of standard solvers becomes prohibitive when the eigenvalues of (1) are large, many specialized techniques have been developed for solving systems of linear

ordinary differential equations in this regime. At the present time, the most widely-used such methods are based on Magnus expansions. Introduced in [15], Magnus expansions are certain series of the form

$$\sum_{k=1}^{\infty} \Omega_k(t) \tag{13}$$

such that $\exp\left(\sum_{k=1}^{\infty} \Omega_k(t)\right)$ locally represents a fundamental matrix for the system of differential equations (1). The first few terms for the series around $t = 0$ are given by

$$\begin{aligned}
\Omega_1(t) &= \int_0^t A(s)\ ds, \\
\Omega_2(t) &= \frac{1}{2} \int_0^t \int_0^{t_1} [A(t_1), A(t_2)]\ dt_2 dt_1 \quad \text{and} \\
\Omega_3(t) &= \frac{1}{6} \int_0^t \int_0^{t_1} \int_0^{t_2} [A(t_1), [A(t_2), A(t_3)]] + [A(t_3), [A(t_2), A(t_1)]]\ dt_3 dt_2 dt_1,
\end{aligned} \tag{14}$$

where $[A, B]$ denotes the matrix commutator $AB - BA$. The straightforward evaluation of the Ω_j is nightmarishly expensive; however, a clever technique which renders the calculations manageable is introduced in [11] and it paved the way for the development of a class of numerical solvers which represent a fundamental matrix for (1) over an interval I via a collection of truncated Magnus expansions. While the entries of the Ω_j are slowly-varying whenever the entries of $A(t)$ are slowly-varying, the radius of convergence of the series in (13) depends on the magnitude of the coefficient matrix $A(t)$, which is, in turn, related to the magnitudes of the eigenvalues of $A(t)$. Of course, this means that the number of Magnus expansions which are needed, and hence the cost of the method, depends on the magnitudes of the eigenvalues of $A(t)$. See, for instance, [10], which gives for estimates of the growth in the running time of Magnus expansion methods in the case of an equation of the form (16) as a function of the magnitude of the coefficient q.

It is only relatively recently that numerical algorithms which exploit the existence of slowly-varying phase functions have appeared. The algorithm of [5] uses a continuation scheme of sorts to construct slowly-varying ψ_1 and ψ_2 such that

$$\exp(\psi_1(t)) \quad \text{and} \quad \exp\left(\psi_2(t)\right) \tag{15}$$

form a basis in the space of solutions of a second order differential equation of the form

$$y''(t) + q(t)y(t) = 0, \qquad a \leq t \leq b, \tag{16}$$

where q is a slowly-varying real-valued function which does not change sign on the interval (a, b). The running time of this algorithm is independent of the size of the eigenvalues of the system corresponding to (16), which are given by

$$\lambda_1(t) = \sqrt{-q(t)} \quad \text{and} \quad \lambda_2(t) = -\sqrt{-q(t)}. \tag{17}$$

The condition that q does not change sign implies that (16) is nondegenerate on (a, b) and it is imposed because slowly-varying phase function need not extend across turning points

where the eigenvalues of coefficient matrix coalesce. The method of [5] can, however, be generalized to the case in which (16) is nondegenerate on $[a, b]$ except at a finite number of turning points. Indeed, if $a = \xi_1 < \xi_2 < \ldots < \xi_k = b$ is a partition of $[a, b]$ such that $\xi_2, \ldots, \xi_{k-1}$ are the roots of q in the open interval (a, b), then simply applying the method of [5] to each of the subintervals $[\xi_j, \xi_{j+1}]$, $j = 1, \ldots, k-1$, results in a collection of $2(k-1)$ slowly-varying phase functions which efficiently represent the solutions of (16). A detailed discussion of this approach, including an account of many numerical experiments performed to demonstrate its efficacy, can be found in [6].

The continuation method of [5] readily extends to the case of higher order scalar equations with slowly-varying coefficients. However, the authors have found that a strongly related algorithm introduced in [1] performs slightly better in practice. When applied to an equation of the form (6) with slowly-varying coefficients, the method of [1] produces a collection $\psi_1, \ldots, \psi_n$ of slowly-varying phase functions such that the functions (8) comprsie basis in the space of solutions of (6). The running time of this algorithm is largely independent of the magnitude of the eigenvalues of the corresponding coefficient matrix (7).

The remainder of this article is structured as follows. In Section 2, we discuss the reduction of systems of linear ordinary differential equations to scalar form. Section 3 reviews the algorithm of [1] for the construction of slowly-varying phase function which represent the solutions of scalar equations. In Section 4, we detail the algorithm of this paper. Section 5 discusses the results of numerical experiments conducted to demonstrate the properties of our algorithm. We close with a few brief remarks in Section 6.

2. Reduction of a system of ordinary differential equations to a scalar equation

It is well known that essentially any system of n linear ordinary differential equations in n unknowns can be transformed into an n^{th} order scalar equation. This is a standard result in differential Galois theory (see, for instance, [8] or [17]) which goes back at least to [14]. In this section, we discuss the well-known mechanism our algorithm uses for constructing such a transformation.

We first observe that if I is an open subinterval of $\mathbb{R}$ and $\Phi\colon I \to \mathbb{C}$ is invertible for all $t \in I$, then letting

$$\mathbf{z}(t) = \Phi(t)\mathbf{y}(t) \tag{18}$$

transforms the system

$$\mathbf{y}'(t) = A(t)\mathbf{y}(t) \tag{19}$$

into

$$\mathbf{z}'(t) = B(t)\mathbf{z}(t), \tag{20}$$

where

$$B(t) = \Phi'(t)\Phi(t)^{-1} + \Phi(t)A(t)\Phi(t)^{-1}. \tag{21}$$

Our goal is to construct $\Phi(t)$ in such a way that (21) is of the scalar form (7). Transformations of this type correspond to cyclic vectors for the transpose of $A(t)$, which are smooth maps $\mathbf{v}\colon I \to \mathbb{C}^n$ such that the matrix

$$\Phi(t) = \begin{pmatrix} \mathbf{v}(t) \\ D\left[\mathbf{v}\right](t) \\ D^2\left[\mathbf{v}\right](t) \\ \vdots \\ D^{n-1}\left[\mathbf{v}\right](t) \end{pmatrix}, \tag{22}$$

where D is the operator defined via

$$D\left[\mathbf{v}\right](t) = \mathbf{v}'(t) + (A(t))^T\mathbf{v}(t), \tag{23}$$

is invertible for all $t \in I$. Indeed, it is easy to see that when $\Phi(t)$ is of the form (22), the matrix

$$\Phi'(t) + \Phi(t)A(t) \tag{24}$$

is given by

$$\begin{pmatrix} D\left[\mathbf{v}\right](t) \\ D^2\left[\mathbf{v}\right](t) \\ \vdots \\ D^{n}\left[\mathbf{v}\right](t) \end{pmatrix}, \tag{25}$$

so that $B(t)$ is of the scalar form (7).

Because the set of cyclic vectors is dense in a certain sense, it is easy to find them. Indeed, one of the standard mechanism used by computer algebra systems to convert systems of differential equations to scalar form entails simply choosing random polynomial vector fields $\mathbf{v}(t)$ until a cyclic vector is found. We refer the reader to [8] and its references for details.

The approach we use to find cyclic vectors is similar. Our algorithm takes as input a constant vector $\mathbf{v}$ which generates the transformation $\Phi(t)$ via (22). The user is expected to make arbitrary choices of this vector until one produces a suitable transformation matrix. It is important to note that it is not sufficient for $\Phi(t)$ to be merely invertible, as it is in symbolic computations. The condition number of the transformation matrix affects the obtainable accuracy of the algorithm, and so various choices of $\mathbf{v}$ must be made until a relatively well-conditioned transformation is found. As mentioned in the introduction, it is increasingly difficult to find a suitable transformation matrix as the dimension of the system under consideration grows.

Many other mechanisms for converting systems of differential equations to scalar form have

been proposed. For instance, [4] discusses a procedure of this type which is analogous to Gaussian elimination. It applies a sequence of "elementary operations" (which are somewhat more complicated than those used in Gaussian elimination) to incrementally convert the coefficient matrix into the desired form. A partially pivoted version of this algorithm aimed at producing a numerically well-conditioned transformation matrix could almost certainly be developed. Moreover, it is clear that various well-known optimization algorithms could be adapted to the problem of finding well-conditioned transformation matrices.

In the algorithm we present here, we use the simplest possible approach — repeatedly make arbitrary choices until a reasonable outcome is obtained — and we leave investigations into improved methods for future work.

3. Phase functions for scalar equations

In this section, we discuss the construction of slowly-varying phase functions for scalar equations of the form (6) with slowly-varying coefficients. One obvious method entails solving the nonlinear $(n-1)^{st}$ order scalar equation satisfied by the derivatives $r_1, \ldots, r_n$ of the phase functions $\psi_1, \ldots, \psi_n$ which is obtained by inserting the representation

$$y(t) = \exp\left(\int r(t)\, dt\right) \tag{26}$$

into (6). By a slight abuse of terminology, we call this nonlinear equation the $(n-1)^{st}$ order Riccati equation, or the Riccati equation corresponding to (6). The general form of this equation is quite complicated, but it is relatively simple at low orders. When $n = 2$, the equation is

$$r'(t) + (r(t))^2 + q_1(t)r(t) + q_0(t) = 0; \tag{27}$$

when $n = 3$, it is

$$r''(t) + 3r'(t)r(t) + (r(t))^3 + q_2(t)r'(t) + q_2(t)(r(t))^2 + q_1(t)r(t) + q_0(t) = 0; \tag{28}$$

and, for $n = 4$, the equation is

$$\begin{aligned} r'''(t) + 4r''(t)r(t) + 3(r'(t))^2 + 6r'(t)(r(t))^2 + (r(t))^4 + q_3(t)(r(t))^3 + q_3(t)r''(t) \\ + 3q_3(t)r'(t)r(t) + q_2(t)(r(t))^2 + q_2(t)r'(t) + q_1(t)r(t) + q_0(t) = 0. \end{aligned} \tag{29}$$

Because most solutions of Riccati equation corresponding to (6) are rapidly varying when the eigenvalues $\lambda_1(t), \ldots, \lambda_n(t)$ of (7) are of large magnitude, some mechanism is needed to select slowly-varying solutions. The algorithm of [5], which applies to second order linear ordinary differential equations of the form (16) with slowly-varying coefficients, uses a continuation scheme of sorts to do so. More explicitly, it introduces a smoothly deformed version of the coefficient $\tilde{q}$ which equal to an appropriately chosen constant λ^2 on a small interval $[a_0, b_0]$ in $[a, b]$ and agrees with q outside of a neighbourhood of U of $[a_0, b_0]$. The values of two slowly-varying solutions $\widetilde{r}_1$ and $\widetilde{r}_2$ of the Riccati equation corresponding to

the deformed version

$$y''(t) + \widetilde{q}(t)y(t) = 0, \qquad a < t < b, \tag{30}$$

of (16) are known at c; indeed, we can take $\widetilde{r_1}(c) = i\lambda$ and $\widetilde{r_2}(c) = -i\lambda$. Solving the Riccati equation corresponding to (30) using these as initial values allows us to compute the values of the derivatives of two slowly-varying phase functions r_1 and r_2 for the original (16) outside the neighborhood U of $[a_0, b_0]$. We can then solve the Riccati equation for the original equation to find the values of r_1 and r_2 inside U. This technique could be easily generalized to the case of higher order scalar equations, but the authors have found the approach of [1], which is inspired by the classical Levin scheme for numerical evaluation of oscillatory integrals, to be somewhat more effective.

Introduced in [12], the Levin method is based on the observation that inhomogeneous equations of the form

$$y'(t) + p_0(t)y(t) = f(t) \tag{31}$$

admit solutions whose complexity depends on that of p_0 and f, but not on the magnitude of p_0. This principle extends to the case of inhomogeneous equations of the form

$$y^{(n)}(t) + p_{n-1}(t)y^{(n-1)}(t) + \cdots + p_1(t)y'(y) + p_0(t)y(t) = f(t). \tag{32}$$

That is, such equations admit solutions whose complexity depends on that of the right-hand side f and of the coefficients $p_0, \ldots, p_{n-1}$, but not on the magnitudes of the coefficients $p_0, \ldots, p_{n-1}$. This is exploited in [1] by applying Newton's method to the Riccati equation for (6). Starting the Newton iterations with a slowly-varying initial guess ensures that each of linearized equations have slowly-varying coefficients, and so admit slowly-varying solutions. Consequently, a slowly-varying solution of the Riccati equation can be constructed via Newton's method as long as an appropriate initial guess is known. Conveniently enough, there is an obvious mechanism for generating n slowly-varying initial guesses for the solution of the $(n-1)^{st}$ order Riccati equation. In particular, the eigenvalues $\lambda_1(t), \ldots, \lambda_n(t)$ of the coefficient matrix (7), which are often used as approximations of solutions of the Riccati equation in asymptotic methods, are suitable for this purpose. A similar methodology is used in [7] in order to develop estimates on the complexity of the slowly-varying phase functions for second order equations of the form (16) in terms of a measure of the complexity of the coefficient q.

Complicating matters is the fact that the differential operator

$$D\left[y\right](t) = y^{(n)}(t) + p_{n-1}(t)y^{(n-1)}(t) + \cdots + p_1(t)y'(y) + p_0(t)y(t) \tag{33}$$

appearing on the left-hand side of (32) admits a nontrivial nullspace comprising all solutions of the homogeneous equation

$$y^{(n)}(t) + p_{n-1}(t)y^{(n-1)}(t) + \cdots + p_1(t)y'(y) + p_0(t)y(t) = 0. \tag{34}$$

This means, of course, that (32) is not uniquely solvable. But it also implies that most solutions of (32) are rapidly-varying when the coefficients $p_0, \ldots, p_{n-1}$ are of large magnitude since the homogeneous equation (34) admits rapidly-varying solutions in such cases.

It is observed in [12] that when the solutions of (34) are all rapidly-varying but (32) admits a slowly-varying solution y_0, a simple spectral collocation method can be used to compute y_0 provided some care is taken in choosing the discretization grid. In particular, if the collocation grid is sufficient to resolve the slowly-varying solution y_0, but not the solutions of (34), then the matrix discretizing (33) will be well-conditioned and inverting it yields y_0.

The article [1] describes an algorithm for constructing slowly-varying phase functions for scalar linear ordinary differential equations based on these principles. It operates in a manner very similar to the algorithm of [5] in that it first computes the values of the derivatives $r_1, r_2, \ldots, r_n$ of the desired slowly-varying phase functions $\psi_1, \ldots, \psi_n$ at a point c, and then solves the Riccati equation numerically with those values used as initial conditions in order to construct $r_1, r_2, \ldots, r_n$ over the whole interval $[a, b]$. Rather than the continuation method of [5], however, the values of the $r_1, r_2, \ldots, r_n$ at c are computed by applying the Levin approach to a single small subinterval of $[a, b]$ containing c. The Riccati equation is then solved to calculate $r_1, \ldots, r_n$ over the whole interval, and these are integrated to obtain the phase functions $\psi_1, \ldots, \psi_n$. Because most solutions of the Riccati equation are rapidly-varying and we are searching for one of a small number of slowly-varying solutions, the ordinary differential equations being solved are extremely stiff. We use a fairly standard adaptive Chebyshev method designed for stiff problems. It is described in detail in [1], but we note that essentially any solver designed to handle stiff ordinary differential equations should suffice.

The algorithm of [1] takes as input:

1. the interval $[a, b]$ over which the equation is given;
2. an external subroutine for evaluating the coefficients $q_0, \ldots, q_{n-1}$ in (6);
3. a point η on the interval $[a, b]$ and the desired values $\psi_1(\eta), \ldots, \psi_n(\eta)$ for the phase functions at that point;
4. a positive integer k which controls the order of the piecewise Chebyshev expansions used to represent phase functions;
5. a parameter ϵ which specifies the desired accuracy for the phase functions; and
6. a subinterval $[a_0, b_0]$ of $[a, b]$ over which the Levin procedure is to be applied and a point σ in that interval.

It outputs a collection of n^2 piecewise Chebyshev expansions of order $(k-1)$ representing the desired slowly-varying phase functions $\psi_1, \ldots, \psi_n$ and their derivatives of orders through $(n-1)$. By a $(k-1)^{st}$ order piecewise Chebyshev expansions on the interval $[a, b]$, we mean a sum of the form

$$\begin{aligned} &\sum_{i=1}^{m-1} \chi_{[x_{i-1}, x_i)}(t) \sum_{j=0}^{k-1} \lambda_{ij}\ T_j\left(\frac{2}{x_i - x_{i-1}} t + \frac{x_i + x_{i-1}}{x_i - x_{i-1}}\right) \\ &+ \chi_{[x_{m-1}, x_m]}(t) \sum_{j=0}^{k-1} \lambda_{mj}\ T_j\left(\frac{2}{x_m - x_{m-1}} t + \frac{x_m + x_{m-1}}{x_m - x_{m-1}}\right), \end{aligned} \tag{35}$$

where $a = x_0 < x_1 < \cdots < x_m = b$ is a partition of $[a, b]$, χ_I is the characteristic function on the interval I and T_j is the Chebyshev polynomial of degree j. The characteristic function of a half-open interval appears in the first line of (35), while the second line conains the characteristic function of a closed interval. This ensures that exactly one of the characteristic appearing functions in (35) is nonzero for each point t in $[a, b]$.

The algorithm of [1] uses the same partition of the interval $[a, b]$ for all of the piecewise Chebyshev expansions it outputs. In most cases, this increases the cost of storing the functions ψ_j and their derivatives somewhat, but it has the advantage of reducing the time required to simultaneously evaluate the $\psi_1, \ldots, \psi_n$ and all of their derivatives at a specified point t. That is because the cost of finding the subinterval containing t is actually much larger than the cost to evaluate a Chebyshev expansion of modest order on current computers.

Our choice to use piecewise Chebyshev expansions to represent the phase functions ψ_j is largely arbitrary. Many different families of Sturm-Liouville eigenfunctions, such as the Legendre polynomials or the prolate spheroidal wave functions of order 0, would serve just as well. One minor advantage of using Chebyshev expansions is that explicit formulas are available for the entries of various spectral differential and integration matrices used by the algorithm of [1], whereas these matrices must be computed via numerical procedures when bases other the Chebyshev polynomials are used.

We note that the matrix

$$\Theta(t) = \begin{pmatrix} u_1(t) & u_2(t) & \cdots & u_n(t) \\ u_1'(t) & u_2'(t) & \cdots & u_n'(t) \\ \vdots & \vdots & \ddots & \vdots \\ u_1^{(n-1)}(t) & u_2^{(n-1)}(t) & \cdots & u_n^{(n-1)}(t) \end{pmatrix}, \tag{36}$$

where the $u_j(t)$ are given by (8), is a fundamental matrix for the system (20) with $B(t)$ taken to be the coefficient matrix (7) associated with the scalar equation. In other words, the columns of $\Theta(t)$ constitute a basis in the space of solutions of the system (20).

4. Numerical Algorithm

Here, we describe our numerical algorithm for solving initial and boundary value problems for a system of linear ordinary differential equations of the form (1) over an interval $[a, b]$ under the assumption that the eigenvalues $\lambda_1(t), \ldots, \lambda_n(t)$ of the system's coefficient matrix $A(t)$ are distinct on (a, b). Our algorithm can easily be extended to the case in which the equation is nondegenerate except at a finite set of turning points in $[a, b]$ by applying it to a collection of subintervals of $[a, b]$.

Our scheme takes as input the following:

1. A positive integer n specifying the dimension of the system;
2. the interval $[a, b]$ over which the problem is given;

3. a positive integer k which controls the order of the piecewise Chebyshev expansions used to represent functions on $[a, b]$;
4. a subroutine for evaluating the elements $a_{ij}(t)$ of the matrix $A(t)$ and all of their derivatives of orders up to n at a specified point t;
5. a parameter ϵ_{disc} specifying the desired precision for discretizations used to represent the inverse transformation matrix $\Phi(t)^{-1}$ and the coefficients $q_0(t), \ldots, q_{n-1}(t)$ of the scalar equation;
6. a parameter ϵ_{phase} specifying the desired precision for the phase functions representing the solutions of the scalar equation;
7. a subinterval $[a_0, b_0]$ of $[a, b]$ over which the Levin procedure used in the construction of the slowly-varying phase functions for the scalar equation is performed; and
8. a constant vector $\mathbf{v} \in \mathbb{C}^n$.

We note that while our algorithm requires as input a subroutine for evaluating the derivatives of the entries of the coefficient matrix $A(t)$ of orders up to n, it is possible for the user-supplied routine to compute these derivatives numerically given only the values of the coefficient matrix $A(t)$.

The output of our algorithm consists of two collections of piecewise Chebyshev expansions. The first collection consists of n^2 piecewise Chebyshev expansions of order $(k-1)$, each of which represents one entry of the inverse $\Phi(t)^{-1}$ of the transformation matrix $\Phi(t)$ defined via (22) with $\mathbf{v}$ the vector supplied by the user. The second collection of expansions comprises n^2 piecewise Chebyshev expansions of order $(k-1)$ representing n slowly-varying phase functions $\psi_1(t), \psi_2(t), \ldots, \psi_n(t)$ for the scalar equation (6) corresponding to the coefficient matrix $B(t)$ given by the formula (21), as well as the derivatives up to order $(n-1)$ of these phase functions. The matrix (36), where the u_j are given by (8), is a fundamental matrix for the scalar system corresponding to (6), and

$$\Psi(t) = \Phi(t)^{-1}\Theta(t) \tag{37}$$

is a fundamental matrix for the original system (1). Our decision to use expansions in Chebyshev polynomials to represent the entries of $\Phi(t)^{-1}$ and $\Theta(t)$ is largely arbitrary. As with the algorithm [1], many other choices of orthonormal basis would serve just as well.

It is important to note that we do not compute piecewise Chebyshev expansions representing the entries of the fundamental matrix $\Psi(t)$. These are rapidly-varying functions which would be extremely expensive to represent in such a fashion. Instead, we compute peicewise Chebyshev expansions of the slowly-varying phase functions and of the slowly-varying entries of the transformation matrix and use these representations to evaluate $\Psi(t)$ as needed via the following procedure. First, we use the piecewise Chebyshev expansions representing the phase functions $\psi_1, \ldots, \psi_n$ and their derivatives to evaluate them at the point t. We then use these values and the formula (8) to evalate the entries of $\Theta(t)$. Next, we form the matrix $\Phi(t)^{-1}$ using the piecewise Chebyshev expansions of its entries computed by our algorithm. Finally, we evaluate $\Psi(t)$ by taking the product of $\Phi(t)^{-1}$ and $\Theta(t)$. Although this sounds cumbersome, it is reasonably efficient because

of the speed with which Chebyshev expansions can be evaluated and the relatively small dimensionality of the systems we consider.

Once the fundamental matrix (37) has been formed, a large class of initial and boundary value problems for (1) can be readily solved. For instance, there is a unique solution of (1) which satisfies the conditions

$$C_1\boldsymbol{y}(t_1) + C_2\boldsymbol{y}(t_2) = \boldsymbol{\eta}, \tag{38}$$

where t_1 and t_2 are points in $[a, b]$, if and only if the matrix

$$Q = C_1\Psi(t_1) + C_2\Psi(t_2) \tag{39}$$

is invertible and, in this event, it is given by $\boldsymbol{y}(t) = \Psi(t)Q^{-1}\boldsymbol{\eta}$. As mentioned in the introduction, when the coefficient matrix $A(t)$ has one or more eigenvalues whose real parts are of large magnitude, many initial and boundary value problems for (1) are numerically intractable. This phenomenon is best illustrated with a simple example. We observe that

$$\Psi(t) = \begin{pmatrix} \exp(t) & \exp(\omega t) \\ 0 & \exp(\omega t) \end{pmatrix} \tag{40}$$

is a fundamental matrix for (1) when

$$A(t) = \begin{pmatrix} \omega & (\omega - 1) \\ 0 & 1 \end{pmatrix}. \tag{41}$$

The eigenvalues of $A(t)$ are 1 and ω, and the unique solution of (38) which satisfies the initial condition $\boldsymbol{y}(0) = (\,1\;0\,)^T$ is $\boldsymbol{y}(t) = \Psi(t)\boldsymbol{\eta}$, where $\boldsymbol{\eta} = (\,1\;0\,)^T$. However, if $\boldsymbol{\eta}$ is calculated numerically, then errors are inevitable and the obtained solution is of the form

$$\Psi(t)\begin{pmatrix} 1+\epsilon_1 \\ \epsilon_2 \end{pmatrix} = (1+\epsilon_1)\begin{pmatrix} \exp(t) \\ 0 \end{pmatrix} + \epsilon_2\begin{pmatrix} \exp(\omega t) \\ \exp(\omega t) \end{pmatrix}, \tag{42}$$

where ϵ_1 and ϵ_2 are small perturbations of the exact coefficients introduced by whatever numerical procedure is used to approximate $\boldsymbol{\eta}$. When ω is large and positive, (42) will obviously deviate substantially from the exact solution of the initial value problem, even if ϵ_1 and ϵ_2 are on the order of machine zero, which is the best that can be hoped for. When ω is large and negative instead, we choose the initial condition to be $\boldsymbol{y}(0) = (\,1\;1\,)^T$. In this case the exact solution is $\Psi(t)(\,0\;1\,)^T$, while the obtained solution is of the form

$$\Psi(t)\begin{pmatrix} \epsilon_1 \\ 1+\epsilon_2 \end{pmatrix} = \epsilon_1\begin{pmatrix} \exp(t) \\ 0 \end{pmatrix} + (1+\epsilon_2)\begin{pmatrix} \exp(\omega t) \\ \exp(\omega t) \end{pmatrix}. \tag{43}$$

Clearly, the relative difference between the two will be extremely large even if ϵ_1 and ϵ_2 are on the order of machine zero. The fundamental issue is that different entries of the

matrix (40) grow at vastly different rates. Standard asymptotic estimates for solutions of systems of ordinary differential equations, such as those in Chapter 7 of [22], show that the same will generally be true when $A(t)$ has eigenvalues whose have real parts are of large magnitude.

The first step of our algorithm consists of adaptively discretizing $\Phi(t)^{-1}$ and the coefficients $q_0(t), \ldots, q_{n-1}(t)$ of the reduced scalar equation. To do so, our algorithm maintains two lists of subintervals of $[a, b]$: one consisting of "accepted subintervals" and the other of subintervals which have yet to be processed. A subinterval is accepted if the entries of $\Phi(t)^{-1}$ and the functions $q_0(t), \ldots, q_{n-1}(t)$ are deemed to be adequately represented by a $(k-1)^{st}$ order Chebyshev expansion on that subinterval. Initially, the list of accepted subintervals is empty and the list of subintervals to process contains the single interval $[a, b]$. We then proceed as follows until the list of subintervals to process is empty:

1. Remove a subinterval $[c, d]$ from the list of intervals to process.
2. Construct the k-point extremal Chebyshev grid $t_1, \ldots, t_k$ on the interval $[c, d]$. The nodes are given by the formula
$$t_j = \frac{d-c}{2} \cos\left(\pi \frac{k-j}{k-1}\right) + \frac{d+c}{2}. \tag{44}$$
3. Evaluate $\Phi(t)$ at each of the nodes $t_1, \ldots, t_k$.
4. For each $j = 1, \ldots, k$, compute a singular value decomposition of the matrix $\Phi(t_j)$ and use it to form the inverse $\Phi(t_j)^{-1}$.
5. For each $j = 1, \ldots, k$, use Formula (21) to compute the matrix $B(t_j)$.
6. Construct $(k-1)^{st}$ order Chebyshev expansions representing each entry of $\Phi(t)$ and the functions $q_0(t), \ldots, q_{n-1}(t)$, appearing in the final row of $B(t)$.
7. For each of the expansions formed in the previous step, which are of the form
$$\sum_{j=0}^{k-1} \alpha_j \; T_j\left(\frac{2}{d-c}t + \frac{d+c}{d-c}\right), \tag{45}$$
we compute the "goodness of fit" metric
$$\frac{\sum_{j=k-3}^{k} |\alpha_j|^2}{\sum_{j=0}^{k-1} |\alpha_j|^2}. \tag{46}$$
8. If every one of the goodness of fit metrics computed in the previous step is less than ϵ_{disc}^2 then we move the interval $[c, d]$ into the list of accepted intervals. Otherwise, we put the intervals
$$\left[c, \frac{c+d}{2}\right], \quad \text{and} \quad \left[\frac{c+d}{2}, d\right] \tag{47}$$
into the list of intervals to process.

Upon termination of this first step, we have piecewise Chebyshev expansions of order $(k-1)$ representing $\Phi(t)^{-1}$ and the coefficients $q_0(t), \ldots, q_{n-1}(t)$ of the reduced scalar

equation. The same partition of $[a, b]$ is used for each piecewise expansion and it is determined by the list of accepted intervals. We note that the matrix $\Phi(t)$ depends on the entries $a_{ij}(t)$ of the coefficient matrix $A(t)$ and their derivatives of orders up to $(n-1)$, while $\Phi'(t)$ depends on the $a_{ij}(t)$ and their derivatives through order n.

In the second step of the algorithm, we use the method of [1], which is discussed in Section 3 of this article, to construct piecewise Chebyshev expansions of order $(k-1)$ representing slowly-varying phase functions $\psi_1(t), \ldots, \psi_n(t)$ for the scalar equation (6) and their derivatives of orders up to $(n-1)$. The precision parameter ϵ_{phase} is passed to this algorithm, as is the integer k controlling the order of the Chebyshev expansions used to represent function and the interval $[a_0, b_0]$ over which the Levin procedure is to be applied. For the sake of simplicity, we choose the constants of integration for the phase functions via the requirements

$$\psi_1(a) = \psi_2(a) = \cdots = \psi_n(a) = 0. \tag{48}$$

The coefficients $q_0(t), \ldots, q_{n-1}(t)$ are evaluated using the piecewise Chebyshev expansions formed in the first step of the procedure.

5. Numerical Experiments

In this section, we present the results of numerical experiments conducted to illustrate the properties of the algorithms of this paper. The code for these experiments was written in Fortran and compiled with version 13.1.1 of the GNU Fortran compiler. They were performed on a desktop computer equipped with an AMD 9950X processor and 64GB of memory. Although this processor has 16 cores, only one was utilized in our experiments.

The algorithm of [1], which we use as a component of the algorithm here, calls for computing the eigenvalues of companion matrices. Standard eigensolvers lose significant accuracy when applied to many matrices of this type. Accordingly, we used the backward stable and highly accurate method described in [3, 2] to perform these calculations.

In each of the experiments described below, we considered a system of linear ordinary differential equations whose coefficient matrix depends on a parameter ω which controls the magnitudes of the eigenvalues of the coefficient matrix for the problem. For each $\omega = 2^8, 2^9, \ldots, 2^{20}$, we solved an initial or boundary value problem for the system over the finite interval $[-1, 1]$ using the method of this paper. The parameter k controlling the order of Chebyshev expansions used by our algorithm was always taken to be 30, while the parameters ϵ_{disc} and ϵ_{phase} which specify the desired precision for the discretization of the transformed system and for the solution of the scalar equation varied from experiment to experiment. Likewise, different choices of the interval $[a_0, b_0]$ on which the Levin procedure was performed and the vector $\mathbf{v}$ used to generate the transformation matrix Φ were made for each experiment.

The accuracy of each solution $\mathbf{y}$ obtained by our algorithm was measured by comparison with a reference solution $\mathbf{z}$ computed using a standard solver. To be more explicit, the

error in each solution was measured via the quantity

$$\max_{1\leq i\leq m} \frac{\|\mathbf{y}(t_i) - \mathbf{z}(t_i)\|_2}{\|\mathbf{z}(t_i)\|_2}, \tag{49}$$

where $m = 10,000$ and $t_1, \ldots, t_m$ are the equispaced points in the interval $[-1, 1]$ given by the formula $t_i = -1 + 2\,(i-1)/(m-1)$. The time taken by our method was measured by repeating the calculation 100 times and averaging the results. For each experiment, we provide a figure containing four plots which report the following quantities, all of which are functions of the parameter ω:

1. the time required by our method;
2. the maximum observed value of (49);
3. the number of m of subintervals in the piecewise discretization schemes used to represent the phase functions and the number l of subintervals in the piecewise discretization scheme used to the entries of the transformation matrix $\Phi(t)^{-1}$; and
4. an estimate of the condition number of the transformation matrix $\Phi(t)^{-1}$, as defined via (11).

Because the same piecewise discretization scheme is used for all of the phase functions and their derivatives, the total number of Chebyshev coefficients used to represent them is simply kn^2m. Likewise, the total number of Chebyshev coefficients used to represent the transformation matrix $\Phi(t)^{-1}$ is kn^2l. We estimated the condition numbers of each of the transformation matrices $\Phi(t)^{-1}$ by calculating the quantity $\|\Phi^{-1}(t)\|\|\Phi(t)\|$ at $1,000$ equispaced nodes in $[a, b]$ and selecting the maximum of the obtained values.

We note that, as discussed in the introduction, the condition number of evaluation of the solutions of a system of the form (1) increases with magnitudes of the eigenvalues of $A(t)$. Accordingly, the accuracy of any numerical method will deteriorates as the magnitudes of the eigenvalues of $A(t)$ increase. Our algorithm is no exception, and we see this effect clearly in the results of our experiments.

5.1. An initial value problem for a system of two equations

In our first experiment, we solved the system of differential equations (1), where

$$A(t) \begin{pmatrix} 1+t^2 & \frac{1}{1+t^4} \\ \frac{-\omega}{1+t^2} & \frac{-i\omega(2+t)}{5+t} \end{pmatrix}, \tag{50}$$

over the interval $[-1, 1]$ subject to the condition $\mathbf{y}(0) = (\,1\;1\,)^T$. Various computer algebra systems can express the eigenvalues $\lambda_1(t), \lambda_2(t)$ of the coefficient matrix appearing in (50) in terms of elementary functions; however, these formulas are too complicated to reproduce here. To give a sense of the magnitudes of $\lambda_1(t), \lambda_2(t)$, we note that

$$\begin{aligned} \lambda_1(0) &= \frac{-2i\omega}{5} - \frac{5i}{2} + \mathcal{O}\left(\frac{1}{\omega}\right) \quad \text{and} \\ \lambda_2(0) &= 1 + \frac{5i}{2} + \mathcal{O}\left(\frac{1}{\omega}\right). \end{aligned} \tag{51}$$

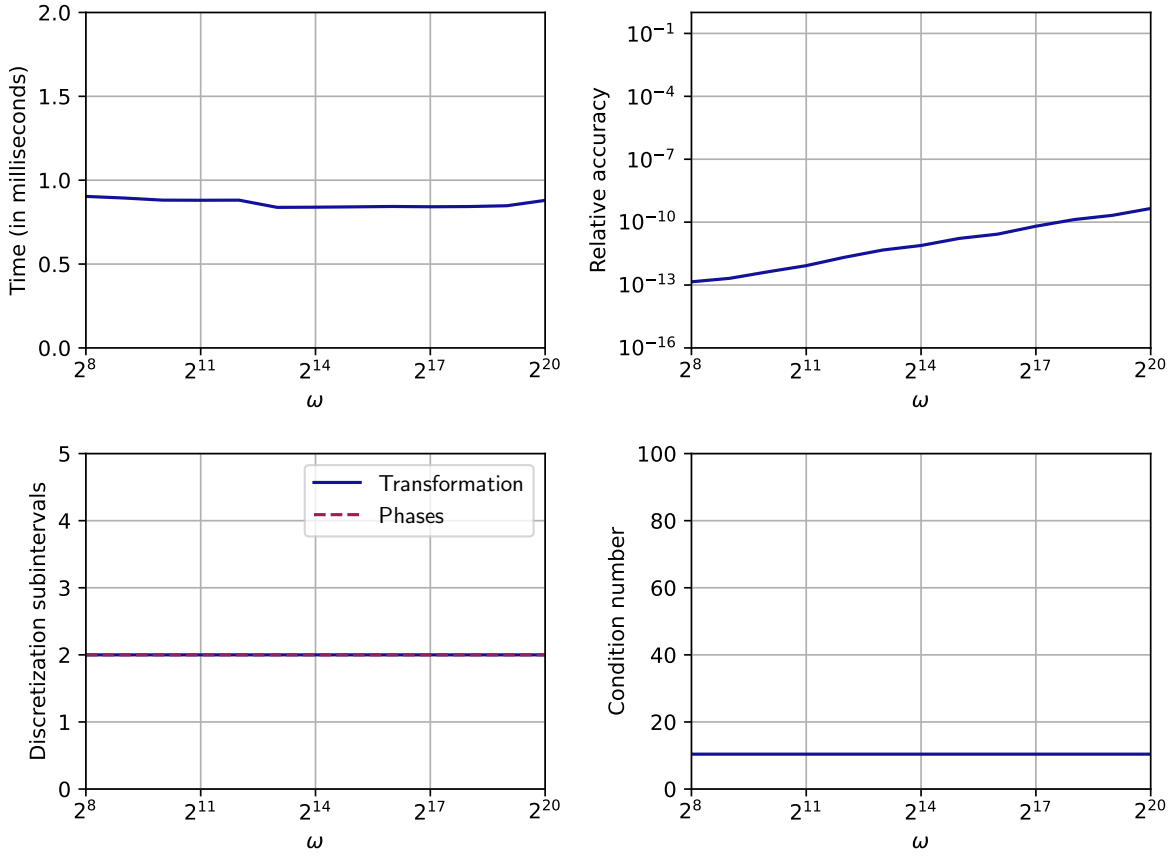

Figure 1: The results of the experiment of Subsection 5.1.

The Levin procedure was performed on the interval $[-0.5, 0.0]$ and the accuracy parameters were taken to be $\epsilon_{\text{disc}} = 1.0 \times 10^{-12}$ and $\epsilon_{\text{phase}} = 1.0 \times 10^{-12}$. The vector $\mathbf{v}$ which generates the transformation matrix Φ was chosen to be $\mathbf{v} = (\,1\;0\,)^T$.

The results of this experiment are reported in Figure 1. For every value ω considered, 480 Chebyshev coefficients were needed to represent the transformation matrix and phase functions. There was surprisingly little variation in the time required by our method given its complexity and the number of adaptive subprocedures it relies on. Indeed, approximately 0.8 milliseconds were required for all values of ω considered. Finallly, we note that, as ω increased from 2^8 to 2^{20}, the relative error increased in an almost linear fashion from 1.42×10^{-13} to 4.47×10^{-10}.

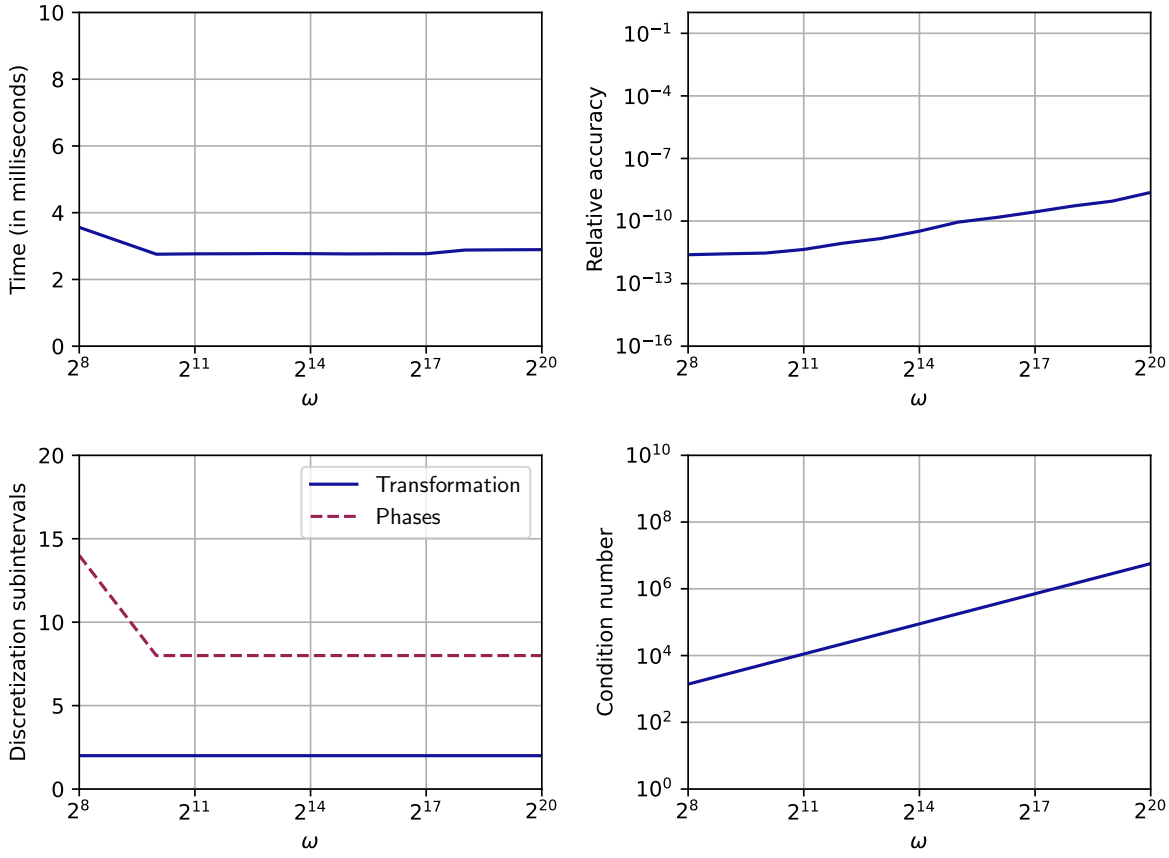

Figure 2: The results of the experiment of Subsection 5.2.

5.2. A boundary value problem for a system of two equations

In this experiment, we solved the system (1), where

$$A(t) = \begin{pmatrix} i\omega\,\frac{2+\sin^2(6t)}{1+t^2} & -\frac{\omega}{1+t^2} \\ i+\omega\exp(t) & i\omega\exp(t) \end{pmatrix}, \tag{52}$$

over the interval $[-1,1]$ subject to the condition

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \boldsymbol{y}(-1) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \boldsymbol{y}(1) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{53}$$

As with the experiment described in the preceding section, while explicit formulas for $\lambda_1(t), \lambda_2(t)$ are available, they are too complicated to reproduce here. However, the fol-

lowing formulas give a sense of their magnitudes:

$$\begin{aligned}\lambda_1(0) &= \frac{3i}{2}\omega - \frac{i}{2}\sqrt{\omega(4i+5\omega)} \sim \frac{i}{2}\left(3+\sqrt{5}\right)\omega - \frac{1}{\sqrt{5}} + \mathcal{O}\left(\frac{1}{\sqrt{\omega}}\right) \quad \text{and} \\ \lambda_2(0) &= \frac{3i}{2}\omega + \frac{i}{2}\sqrt{\omega(4i+5\omega)} \sim \frac{i}{2}\left(3-\sqrt{5}\right)\omega + \frac{1}{\sqrt{5}} + \mathcal{O}\left(\frac{1}{\sqrt{\omega}}\right).\end{aligned} \tag{54}$$

The Levin procedure was performed on the interval $[-0.5, 0.0]$ and we took the accuracy parameters to be $\epsilon_{\text{disc}} = 1.0 \times 10^{-12}$ and $\epsilon_{\text{phase}} = 1.0 \times 10^{-12}$. The vector $\boldsymbol{v}$ which generates the transformation matrix Φ was $\boldsymbol{v} = (\,0\;1\,)^T$. Figure 2 gives the results of this experiment.

For $\omega = 2^8$, 1,920 Chebyshev coefficients were used to represent the phase functions and transformation matrix, while 1,560 coefficients were needed when $\omega = 2^9$. For all other values of ω, 1,200 Chebyshev coefficients sufficed. It is unsurprising that the number of coefficients decreased with ω, because the complexity of phase functions generally decreases with increasing frequency. The time required by our algorithm was a bit higher when $\omega = 2^8$ and $\omega = 2^9$, but, again, this is a consequence of the slightly higher complexity of the phase functions for those values of ω. The obtained relative error in the solutions increased from approximately 2.305×10^{-12} when $\omega = 2^8$ to around 2.284×10^{-9} when $\omega = 2^{20}$.

The condition number of the transformation matrix was roughly 1.392×10^3 when $\omega = 2^8$ and it increased in an almost linear fashion to 5.701×10^7 by the time $\omega = 2^{20}$. Although 5.701×10^7 is quite a large condition number, from (54) we see tht the condition number of evaluation of the solutions behaves as $\mathcal{O}(\omega)$. In particular, when $\omega = 2^{20}$, the condition number of the transformation matrix is about an order of magnitude larger than the condition number of evaluation of the solutions, so it seems likely that only a modest amount of accuracy was lost to the ill-conditioning of the transformation matrix.

5.3. An initial value problem for a system of three equations

In this experiment, we solved the system of differential equations (1), where

$$A(t) = \begin{pmatrix} -i\omega\left(3t^2 + e^{t-12t^2} + 3e^t + e^t\cos(17t) + 3\right) & -i\omega\left(e^{-12t^2} + \cos(17t) + 3\right) \\ -i\omega e^t\left(-3t^2 + e^{-12t^2} - 2\right) & -i\omega\left(e^{-12t^2} + 1\right) \\ -i\omega e^t\left(e^{-12t^2} + \cos(17t) + 3\right) & -i\omega\left(e^{-12t^2} + \cos(17t) + 3\right) \end{pmatrix}$$

$$\begin{pmatrix} i\omega\left(3t^2 + e^{t-12t^2} + 3e^t + (e^t+1)\cos(17t) + 5\right) \\ i\omega e^t\left(-3t^2 + e^{-12t^2} - 2\right) \\ i\omega\left(e^{t-12t^2} + 3e^t + (e^t+1)\cos(17t) + 2\right) \end{pmatrix},$$

over the interval $[-1, 1]$ subject to the condition $\boldsymbol{y}(-1) = (\,1\;\;0\;\;-1\,)^T$. The eigenvalues of $A(t)$ are given by the following formulas:

$$\begin{aligned} \lambda_1(t) &= i\omega\,(2 + \cos(12t))\,, \\ \lambda_2(t) &= -3i\omega(1 + t^2) \quad \text{and} \\ \lambda_3(t) &= -i\omega\left(1 + \exp\left(-12t^2\right)\right). \end{aligned} \tag{55}$$

The Levin procedure was performed on the interval $[-0.25, 0.0]$ and the accuracy parameters were taken to be $\epsilon_{\text{disc}} = 1.0 \times 10^{-12}$ and $\epsilon_{\text{phase}} = 1.0 \times 10^{-12}$. The initial vector for constructing the transformation matrix was chosen to be $\boldsymbol{v} = (\,1\;0\;0\,)^T$. Figure 3 gives the results of this experiment. For all values of ω considered, 3,780 Chebyshev coefficients were used to represen the transformation matrix. The number of Chebyshev coefficients required to represent the phase functions decreased from 1,620 to 540 as ω increased from 2^8 to 2^{20}. The time taken by our algorithm also decreased slightly with increasing ω. The condition number of the transformation matrix increased from 5.097×10^3 (when $\omega = 2^8$) to 2.088×10^7 (when $\omega = 2^{20}$). Again, because the eigenvalues are on the order of ω, the

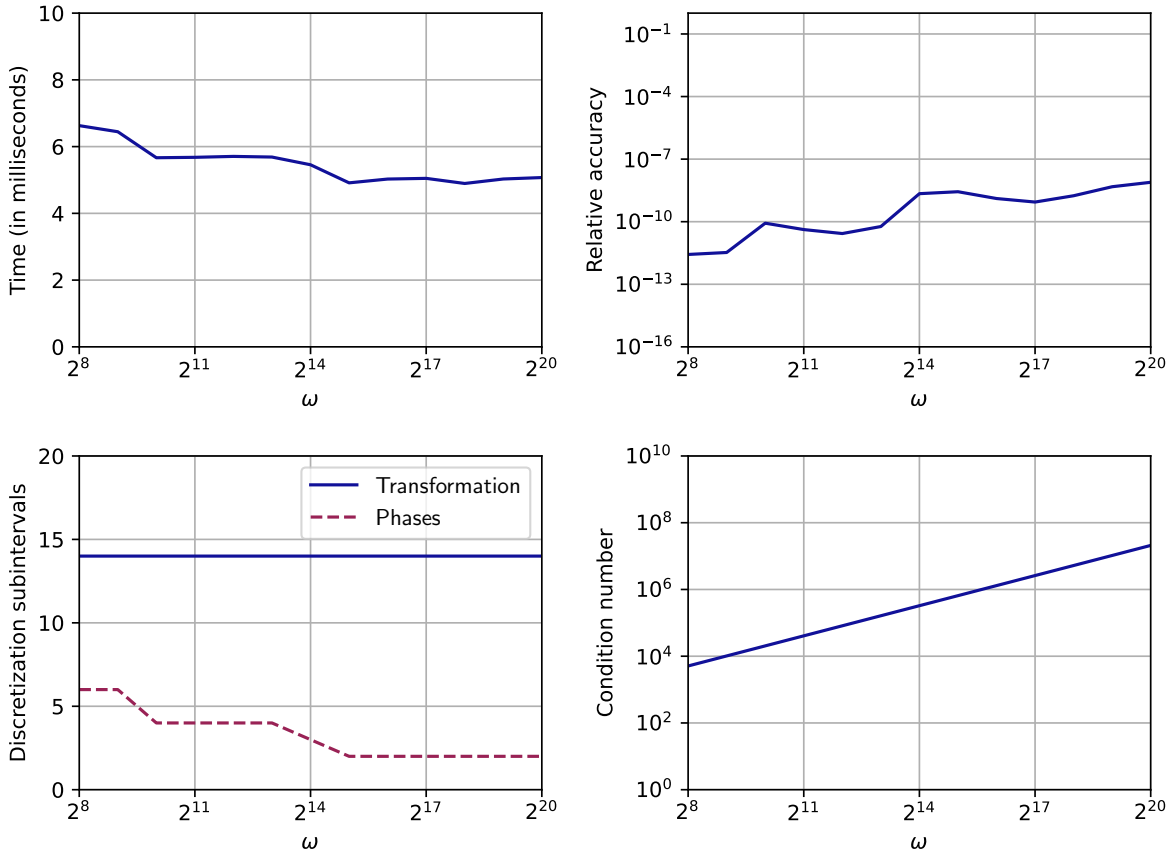

Figure 3: The results of the experiment of Subsection 5.3.

condition number of the transformation matrix was only about an order of magnitude larger than the condition number of evaluation of the solutions when $\omega = 2^{20}$. There was somewhat more variability in the obtained accuracy than in previous experiments; however, relatively high accuracy was still achieved in all cases and it is unclear if the variation is due to intrinsic properties of the problem (i.e., variation in the condition number) or the properties of our algorithm.

5.4. A boundary value problem for a system of three equations

In this experiment, we solved the system of differential equations (1), where

$$A(t) = \begin{pmatrix} \frac{4(1+4i\omega)e^{3t^2}+8i\omega e^{3t^2}\sin(3t)-i\omega t\log\left(t+\frac{1001}{1000}\right)}{4e^{3t^2}-t} & 0 & \frac{2ie^{t^2}t\left(\omega\log\left(t+\frac{1001}{1000}\right)-2\omega\sin(3t)-4\omega+i\right)}{4e^{3t^2}-t} \\ \frac{2e^{2t^2}t\left(-8i\omega e^{t^2}+2i\omega\sin(3t)+\log(\omega)\sin(t)+4i\omega+1\right)}{4e^{3t^2}-t} & -\log(\omega)\sin(t)+8i\omega e^{t^2} & \frac{it^2\left(8\omega e^{t^2}-2\omega\sin(3t)+i\log(\omega)\sin(t)-4\omega+i\right)}{4e^{3t^2}-t} \\ \frac{2e^{2t^2}\left(-i\omega\log\left(t+\frac{1001}{1000}\right)+2i\omega\sin(3t)+4i\omega+1\right)}{4e^{3t^2}-t} & 0 & \frac{i\left(4\omega e^{3t^2}\log\left(t+\frac{1001}{1000}\right)+(-4\omega+i)t-2\omega t\sin(3t)\right)}{4e^{3t^2}-t} \end{pmatrix}, \tag{56}$$

over the interval $[-1, 1]$ subject to the condition

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \boldsymbol{y}(-1) + \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{pmatrix} \boldsymbol{y}(1) = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}. \tag{57}$$

The eigenvalues of $A(t)$ are given by the following formulas:

$$\begin{aligned} \lambda_1(t) &= 1 + 2i\omega\left(2 + \sin(3t)\right), \\ \lambda_2(t) &= -\log(\omega)\sin(t) + 8i\omega\exp(t^2) \quad \text{and} \\ \lambda_3(t) &= i\omega\log\left(1 + 10^{-3} + t\right). \end{aligned} \tag{58}$$

The Levin procedure was performed on the interval $[-0.1, 0.0]$ and the accuracy parameters $\epsilon_{\text{disc}} = 1.0 \times 10^{-12}$ and $\epsilon_{\text{phase}} = 1.0 \times 10^{-12}$. We used $\boldsymbol{v} = (\,1\ 1\ 1\,)^T$. as the initial vector for constructing the transformation matrix Φ. Figure 4 gives the results of this experiment. For each value of ω considered, 2,970 Chebyshev coefficients were used to represent the transformation matrix and 2,700 were used to represent the phase functions, for a total of 5,6700 coefficients. The running time of the algorithm trended downward as ω increases, falling from approximately 14 milliseconds when $\omega = 2^8$ to a bit over 10 milliseconds when $\omega = 2^{20}$.

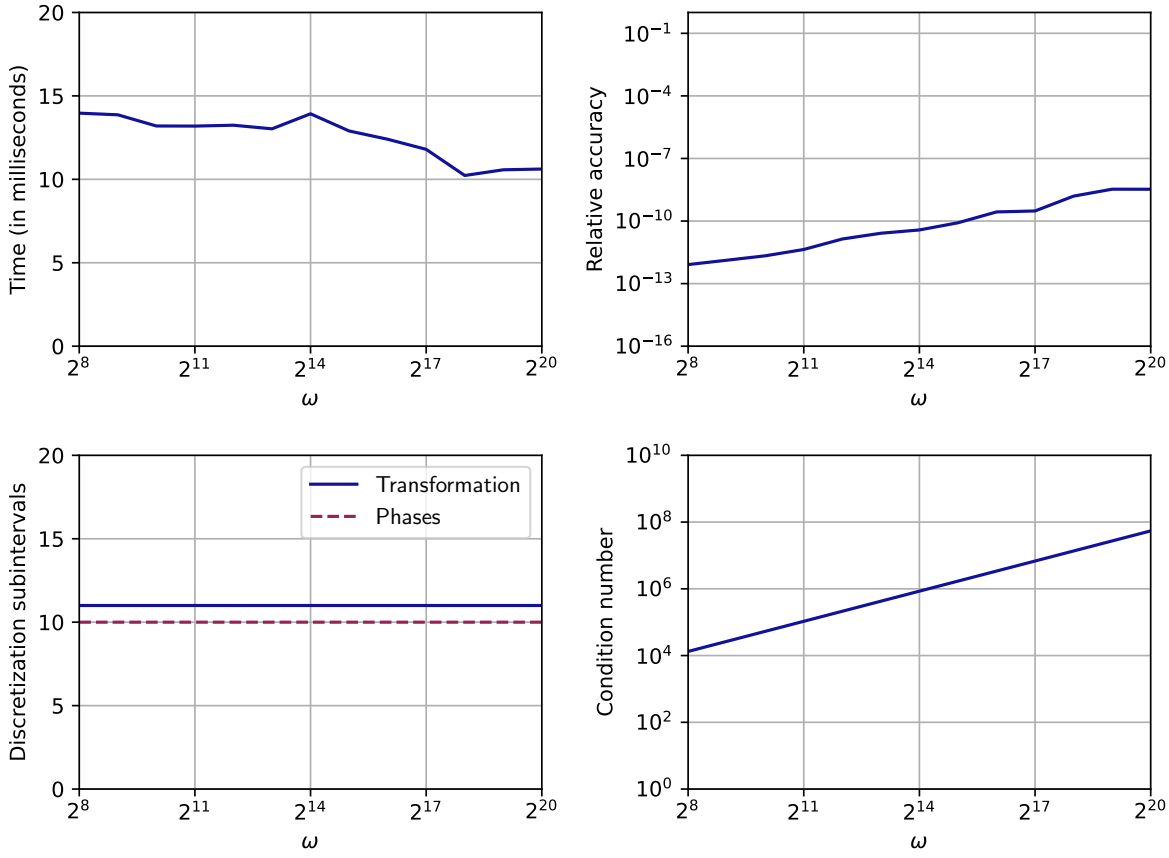

Figure 4: The results of the experiment of Subsection 5.4.

5.5. An initial value problem for a system of four equations

In this experiment, we solved the system of differential equations (1), where coefficient matrix $A(t)$ is given by

$$\begin{pmatrix} \frac{\cos(t)\left(\log(t+2)-i\sqrt{\omega}\right)-8i\omega e^{2t^2}}{4e^{t^2}+\cos(t)} & 0 & -\frac{ie^{t^2}\cos(t)\left(2\omega e^{t^2}-\sqrt{\omega}-i\log(t+2)\right)}{(t^2+1)\left(4e^{t^2}+\cos(t)\right)} \\ 0 & \frac{-i\omega(t-8)+t\left(2e^t-i\omega\right)\sin(t)+2e^t t}{4t^2-2t-2t\sin(t)+4} & 0 \\ -\frac{4i\left(t^2+1\right)\left(2\omega e^{t^2}-\sqrt{\omega}-i\log(t+2)\right)}{4e^{t^2}+\cos(t)} & 0 & \frac{2e^{t^2}\left(-i\omega\cos(t)-2i\sqrt{\omega}+2\log(t+2)\right)}{4e^{t^2}+\cos(t)} \\ 0 & \frac{t\left(2e^t\left(t^2+1\right)-i\omega\left(t^2-3\right)\right)}{2(t^2+1)(2t^2-t-t\sin(t)+2)} & 0 \end{pmatrix}$$

$$\left.\begin{array}{c} 0 \\ -\frac{\left(2e^t\left(t^2+1\right)-i\omega\left(t^2-3\right)\right)(\sin(t)+1)}{2t^2-t-t\sin(t)+2} \\ 0 \\ \frac{i\omega\left(t^4+2t^2-2t+1\right)-2i\omega t\sin(t)-2e^t\left(t^2+1\right)^2}{(t^2+1)(2t^2-t-t\sin(t)+2)} \end{array}\right),$$

over the interval $[-1, 1]$ subject to the condition $\boldsymbol{y}(0) = (\,1\ -1\ 1\ -1\,)^T$ The eigenvalues of $A(t)$ are:

$$\begin{aligned} \lambda_1(t) &= -2i\omega\exp(t^2), \\ \lambda_2(t) &= \frac{2i\omega}{1+t^2}, \\ \lambda_3(t) &= \log(2+t) - i\sqrt{\omega} \ \text{ and} \\ \lambda_4(t) &= -\exp(t) + \frac{i\omega}{2}. \end{aligned} \tag{59}$$

The Levin procedure was performed on the interval $[-0.25, 0.00]$, and the accuracy parameters were taken to be $\epsilon_{\text{disc}} = 1.0 \times 10^{-10}$ and $\epsilon_{\text{phase}} = 1.0 \times 10^{-10}$. The initial vector for constructing the transformation matrix was taken to be $\boldsymbol{v} = (\,0\ 1\ 1\ 0\,)^T$. Because the condition number of the transformations Φ our algorithm forms tend to increase with the order of the system, the accuracy achievable by the algorithm of this paper generally decreases with order. This is why the accuracy parameters needed to be lowered somewhat for this experiment. Figure 5 gives the results of this experiment.

For all values of ω except 2^{20}, our solver took under 10 milliseconds and the transformation matrix and phase functions were represented using 1,920 Chebyshev coefficients. When $\omega = 2^{20}$, 4,320 Chebyshev coefficients were needed and the running time increased to around 16 milliseconds. The relative accuracy of the obtained solution increased from around 8.717×10^{-10} when $\omega = 2^8$ to approximately 7.859×10^{-7} when $\omega = 2^{20}$. The condition number of the transformation matrix increased from 2.500×10^4 when $\omega = 2^8$ to 1.023×10^8 when $\omega = 2^{20}$. The condition numbers of the transformation matrix are somewhat larger than in previous experiments because this experiment concerns a system of four equations. The obtained accuracy is, nonetheless, reasonable high. It is not entirely clear why the cost to represent the phase functions, and also the algorithm's running time, increased substantially when ω was taken to be 2^{20}, but the performance of the algorithm is still acceptable in this case.

6. Conclusions

We have introduced a numerical method for solving a large class of systems of ordinary differential equations of modest dimensions in time independent of the magnitudes of the eigenvalues of the system's coefficient matrix. It operates by transforming the system into a scalar equation via a standard approach, and then calculating slowly-varying phase functions which represent a basis in the space of solutions of the scalar equation.

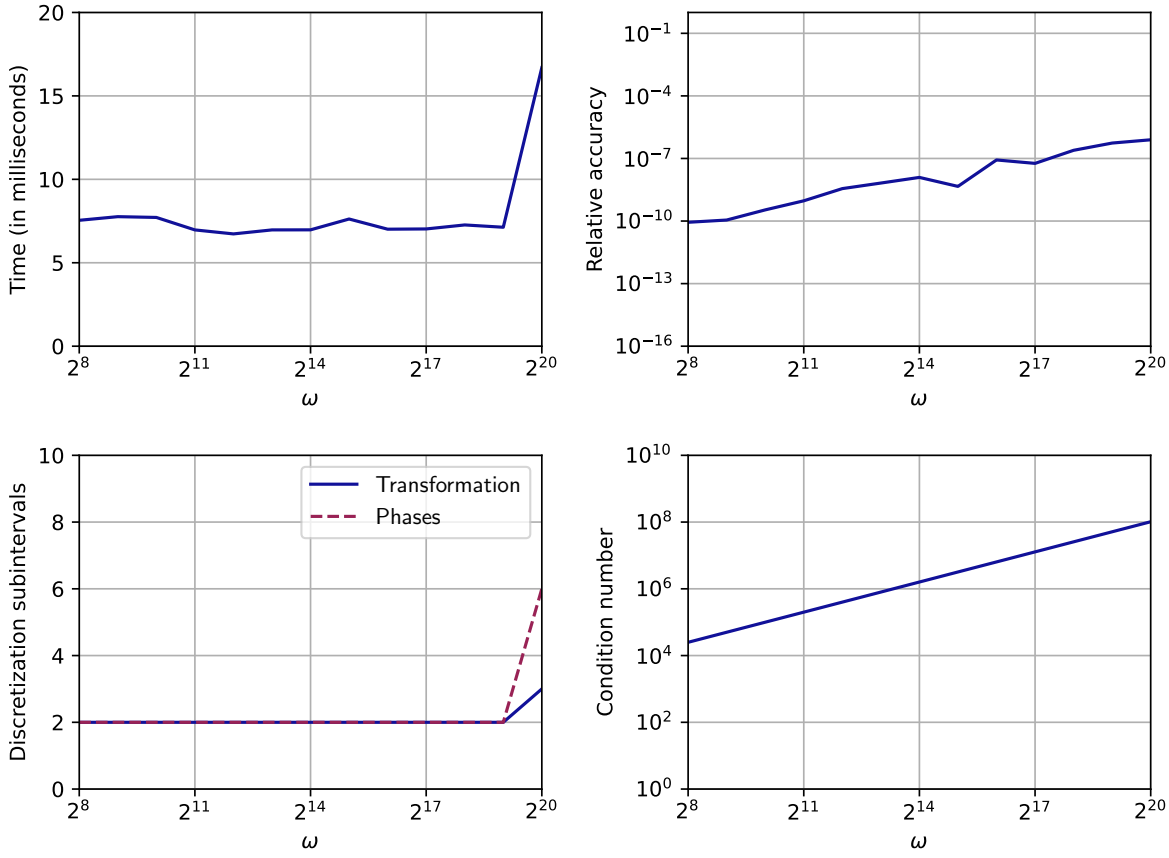

Figure 5: The results of the experiment of Subsection 5.5.

Our algorithm is limited to systems of modest dimensions because the transformation matrices we use become increasingly ill-conditioned as the dimension of the system grows. The set of transformations which take a given system to scalar form, however, is well-known to be large in various senses and it is likely that well-conditioned transformations exist. The most obvious way to overcome the difficulties of our current approach would be to develop an optimization algorithm to efficiently search through the set of possible transformations for a well-conditioned one. The development of a numerical version of the elimination algorithm of [4] (which we briefly discuss in Section 2) is another possible approach to constructing a stable transformation matrix. Alternatively, one could simply insert the representation (36) into the system (1) and derive a system of differential equations satisfied by the phase functions $\psi_1, \ldots, \psi_n$ and the entries of the transformation matrix $\Phi(t)$ which could be solved numerically. The authors are actively investigating these and other possible methods for improving the algorithm of this paper.

Despite its limitations, the method shows that the solutions of a large class of systems of differential equations can be represented more efficiently than was previously believed to be possible. Moreover, we view it as a promising first step toward the development of a

class of efficient generaly purpose "frequency-independent" solvers for systems of ordinary differential equations with slowly-varying coefficient matrices.

7. Acknowledgments

JB was supported in part by NSERC Discovery grant RGPIN-2021-02613. The authors are grateful to Kirill Serkh for many helpful discussions regarding this work. The authors would also like to thank the anonymous reviewers of this work for their many helpful comments.

8. Data availability statement

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

9. Competing Interests Declaration

The authors confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

[1] Aubry, M., and Bremer, J. A solver for linear scalar ordinary differential equations whose running time is bounded independent of frequency. *arXiv 2311.08578* (2023).

[2] Aurentz, J., Mach, T., Robol, L., Vanderbril, R., and Watkins, D. S. Fast and backward stable computation of roots of polynomials, part II: Backward error analysis; companion matrix and companion pencil. *SIAM Journal on Matrix Analysis and Applications 39* (2018), 1245–1269.

[3] Aurentz, J., Mach, T., Vanderbril, R., and Watkins, D. S. Fast and backward stable computation of roots of polynomials. *SIAM Journal on Matrix Analysis and Applications 36* (2015), 942–973.

[4] Barkatou, M. A. An algorithm for computing a companion block diagonal form for a system of linear differential equations. *Applicable Algebra in Engineering, Communication and Computing 4* (1993), 185–195.

[5] Bremer, J. On the numerical solution of second order differential equations in the high-frequency regime. *Applied and Computational Harmonic Analysis 44* (2018), 312–349.

[6] Bremer, J. Phase function methods for second order linear ordinary differential equations with turning points. *Applied and Computational Harmonic Analysis 65* (2023), 137–169.

[7] Bremer, J., and Rokhlin, V. Improved estimates for nonoscillatory phase functions. *Discrete and Continuous Dynamical Systems, Series A 36* (2016), 4101–4131.

[8] Churchill, R., and Kovacic, J. Cyclic vectors. In *Differential Algebra and Related Topics*, L. Guo, W. F. Keigher, P. J. Cassidy, and W. Y. Sit, Eds. World Scientific, 2002, pp. 191–217.

[9] Higham, N. J. *Accuracy and Stability of Numerical Algorithms*, second ed. SIAM, 2002.

[10] Iserles, A. On the global error of discretization methods for highly-oscillatory ordinary differential equations. *BIT 32* (2002), 561–599.

[11] Iserles, A., and Nørsett, S. P. On the solution of linear differential equations in Lie groups. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences 357*, 1754 (1999), 983–1019.

[12] Levin, D. Procedures for computing one- and two-dimensional integrals of functions with rapid irregular oscillations. *Mathematics of Computation 38* (1982), 531–5538.

[13] Levin, D. Fast integration of rapidly oscillatory functions. *Journal of Computational and Applied Mathematics 67* (1996), 95–101.

[14] Loewy, A. *Über einen Fundamentalsatz für Matrizen oder Lineare Homogene Differentialsysteme*. Sitzungsberichte der Heidelberger Akademie der Wissenschaften, Mathematisch-naturwissenschaftliche Klasse, 1918.

[15] Magnus, W. On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics 7* (1954), 649–673.

[16] Miller, P. D. *Applied Asymptotic Analysis*. American Mathematical Society, Providence, Rhode Island, 2006.

[17] Put, M., and Singer, M. *Galois Theory of Linear Differential Equations*. Spinger Berlin, Heidelberg, 2003.

[18] Serkh, K., and Bremer, J. Phase function methods for second order inhomogeneous linear ordinary differential equations. *Journal of Scientific Computing 98* (2023).

[19] Spigler, R. Asymptotic-numerical approximations for highly oscillatory second-order differential equations by the phase function method. *Journal of Mathematical Analysis and Applications 463* (2018), 318–344.

[20] Spigler, R., and Vianello, M. A numerical method for evaluating the zeros of solutions of second-order linear differential equations. *Mathematics of Computation 55* (1990), 591–612.

[21] Spigler, R., and Vianello, M. The phase function method to solve second-order asymptotically polynomial differential equations. *Numerische Mathematik 121* (2012), 565–586.

[22] Wasow, W. *Asymptotic expansions for ordinary differential equations*. Dover, 1965.