COMMUN. MATH. SCI.
Vol. 5, No. 3, pp. 723–741

AUTOMATIC COORDINATE TRANSFORMATION FOR TWO-POINT BOUNDARY VALUE PROBLEMS*

AADITYA V. RANGAN†

Abstract. When one is attempting to build a general boundary value problem solver which uses Green's functions, it is often desirable to automatically transform inhomogeneous boundary value problems into homogeneous boundary value problems. If the sum of the boundary matrices is singular, a change of coordinates is required which can transform the inhomogeneous boundary value problem into a homogeneous boundary value problem. This change of coordinates is an invertible C^1 path in matrix space with endpoints that can multiply the boundary matrices to produce a nonsingular sum. Here we propose a simple automatic coordinate transformation that performs this task. We prove that this algorithm constructs a coordinate transformation whenever such a transformation exists, and provide numerical examples illustrating the practicability of the method.

Key words. linear two-point boundary value problems, spectral integral methods, boundary matrices

AMS subject classifications. 65N99

1. Introduction

Boundary value problems often arise from studies in science and engineering [1, 6, 7, 8, 10]. For example, partial differential equations can be reduced to sequences of boundary value problems via Rothe's method [2]. Many of these naturally occurring boundary value problems are stiff, and require specialized numerical methods to solve.

The integral methods of [9, 14] are particularly useful for solving stiff 1-dimensional two-point boundary value problems, since they can adaptively discretize space as necessary in an asymptotically optimal manner. These integral methods can be applied to find the vector-valued solution $\Phi(x):[-1,+1]\to R^n$ of a linear two-point boundary value problem of the form

$$\begin{aligned}&\partial_x\Phi(x)+p(x)\Phi(x)=f(x),\\&A\Phi(-1)+C\Phi(+1)=0,\end{aligned}\tag{1.1}$$

with matrix-valued coefficient function $p(x):[-1,+1]\to R^{n\times n}$, vector-valued forcing function $f(x):[-1,1]\to R^n$, and boundary matrices $A,C\in R^{n\times n}$. This integral method uses the fact that the solution to the linear homogeneous boundary value problem Equation (1.1) can be written using the Green's function [5]. The Green's function $G(x,y)$ is the solution of Equation (1.1) with the right hand side $f=\delta(x-y)$, and can be used to generate the solution of Equation (1.1) for arbitrary $f(x)$ by simple linear superposition,

$$\Phi(x)=\int_{-1}^{1}G(x,y)f(y)dy.\tag{1.2}$$

In order for this Green's function formulation to be applicable, the original linear two-point boundary value problem (Equation (1.1)) must have homogeneous boundary conditions (i.e., $A\Phi(-1)+C\Phi(+1)=0$).

*Received: April 30, 2007; accepted (in revised version): July 24, 2007. Communicated by Peter Smereka.

†251 Mercer Street, New York, NY 10012-1185, USA. http://cims.nyu.edu/∼rangan

If we want to solve a linear two-point boundary value problem with inhomogeneous boundary conditions

$$\begin{aligned} &\partial_x \Phi(x) + p(x)\Phi(x) = f(x), \\ &A\Phi(-1) + C\Phi(+1) = \gamma, \end{aligned} \tag{1.3}$$

with nonzero boundary vector $\gamma \in R^n$, then we cannot directly apply the integral method [14]. When confronted with inhomogeneous boundary conditions, one standard strategy is to transform the original problem Equation (1.3) into a boundary value problem with homogeneous boundary conditions [9]. The way this is accomplished is to define

$$\begin{aligned} \hat{\gamma} &= (A+C)^{-1}\gamma, \\ \hat{\Phi}(x) &= \Phi(x) - \hat{\gamma}, \\ \hat{f}(x) &= f(x) - p(x)\hat{\gamma} \end{aligned} \tag{1.4}$$

and rewrite Equation (1.3) as

$$\begin{aligned} &\partial_x \hat{\Phi}(x) + p(x)\hat{\Phi}(x) = \hat{f}(x), \\ &A\hat{\Phi}(-1) + C\hat{\Phi}(+1) = 0. \end{aligned} \tag{1.5}$$

However, there are many two-point boundary value problems where the matrix sum $A+C$ is singular, and the simple transformation Equation (1.4) cannot be applied [5]. One common situation where this can occur is when components of a differential equation have periodic boundary conditions (i.e., in the simplest case of fully periodic boundary conditions, $A = -C = I_{n\times n}$). Another common situation is when a second order system of differential equations with, say, Dirichlet boundary conditions is reduced to a first order system. For example, the simple second order equation $\partial_{xx} y = L \cdot y$ with boundary conditions $y(-1) = \gamma^{-1}, y(1) = \gamma^{+1}$ reduces to the first order system

$$\partial_x \begin{bmatrix} y \\ \partial_x y \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ L & 0 \end{bmatrix} \cdot \begin{bmatrix} y \\ \partial_x y \end{bmatrix},$$

with boundary conditions

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} y(-1) \\ \partial_x y(-1) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} y(-1) \\ \partial_x y(-1) \end{bmatrix} = \begin{bmatrix} \gamma^{-1} \\ \gamma^{+1} \end{bmatrix}.$$

It is also possible for these two situations to be combined, as occurs within a large second (or higher) order system where some components have periodic boundary conditions.

In addition, there are several cases where the condition number of the matrix sum $A+C$ is very high, even though the matrix is not exactly singular. This can result in poorly conditioned Green's function operators (which often rely heavily on multiplication by $(A+C)^{-1}$ [14]), and consequent loss of numerical precision. One such example can be found in systems of kinetic equations for neuronal network dynamics [3, 12], which govern the evolution of a single particle (i.e., single neuron) probability density function. In this case, the sum of boundary matrices $A+C$ can be arbitrarily ill-conditioned, depending on the neuron's 'firing-rate' (i.e., level of neuronal activity). To complicate matters, the firing rate of the system itself is a

sensitive functional of the probability distribution, and can change over time. The accurate application of integral equation methods to this problem depends critically on the accurate resolution of the firing rate and probability distribution [13]. Thus, situations where the boundary matrix $A+C$ is nearly singular must be treated with care.

Nevertheless, even when the simple change of variables above Equation (1.4) cannot be directly applied, it may still be possible to transform problems of type (1.3) with singular D into problems of type (1.1) as follows.

If γ is nonzero and $D=A+C$ is singular, we can attempt to construct an invertible and differentiable coordinate transformation

$$\mathcal{T}(x):[-1,+1]\rightarrow R^{n\times n} \tag{1.6}$$

such that

$$\hat{D}=A\mathcal{T}(-1)+C\mathcal{T}(+1) \tag{1.7}$$

is nonsingular. Given such a transformation $\mathcal{T}$, we can transform the problem

$$\begin{aligned}\hat{\gamma}&=\hat{D}^{-1}\gamma,\\ \hat{p}&=\mathcal{T}(x)^{-1}(\partial_x\mathcal{T}(x)+p(x)\mathcal{T}(x)),\\ \hat{f}(x)&=\mathcal{T}(x)^{-1}f(x)-\hat{p}\hat{\gamma},\\ \hat{\Phi}(x)&=\mathcal{T}(x)^{-1}\Phi(x)-\hat{\gamma},\end{aligned} \tag{1.8}$$

so that $\hat{\Phi}(x)$ satisfies the boundary value problem:

$$\begin{aligned}\partial_x\hat{\Phi}(x)+\hat{p}(x)\hat{\Phi}(x)&=\hat{f}(x),\\ A\mathcal{T}(-1)\hat{\Phi}(-1)+C\mathcal{T}(+1)\hat{\Phi}(+1)&=0,\end{aligned} \tag{1.9}$$

with homogeneous boundary conditions.

If the dimension n is very large, or one wishes to solve many two-point boundary value problems, it becomes cumbersome to construct $\mathcal{T}(x)$ by hand, and it becomes desirable to have a method for constructing $\mathcal{T}(x)$. In addition, when designing an automatic two-point boundary value problem solver (such as a module which can apply the integral solvers of [9]), it is desirable to automatically generate the coordinate transformation $\mathcal{T}(x)$, as well as its derivative and inverse (note the terms $\mathcal{T}(x)^{-1}$ and $\partial_x\mathcal{T}(x)$ in Equation (1.8)). For the integral solvers of [9], space is discretized adaptively, therefore, it also becomes necessary to evaluate $\mathcal{T}(x),\partial_x\mathcal{T}(x)$ and $\mathcal{T}(x)^{-1}$ for arbitrary x within the domain.

In this paper, we provide a simple algorithm which, given the boundary matrices A and C, generates a coordinate path $\mathcal{T}(x)$ whenever such a path exists. Our method uses scaling matrices, reflection matrices and permutation matrices to provide analytical formulas for $\mathcal{T}(x)$, $\partial_x\mathcal{T}(x)$ and $\mathcal{T}^{-1}(x)$, which can then be evaluated for any x. Our method performs very well in practice, and when tested on pairs of random matrices with singular sums, produces well-conditioned transformations $\mathcal{T}(x),\partial_x\mathcal{T}(x),\mathcal{T}^{-1}(x)$.

The paper is organized as follows. In Section 2, we give an example which describes and illustrates the main features of our method. In Section 3, we detail the construction of the algorithm and provide a proof that this method constructs a path whenever a path exists. In Section 4, we provide numerical examples which show that this method is practicable and produces well-conditioned coordinate transformations. Finally, in Section 5, we present psuedocode for our algorithm.

2. Automated coordinate change: overview

For the remainder of this paper, we assume that matrices A and C are given, such that the condition number of $D=(A+C)$ is large (or infinite). We focus on the issue of constructing a continuous invertible $\mathcal{T}(x)$ such that $\hat{D}=A\mathcal{T}(-1)+C\mathcal{T}(+1)$ is invertible with a low condition number. The apparatus we construct revolves around two basic ideas.

Consider, first, the case

$$A=1, \quad C=-1. \tag{2.1}$$

Here we only have one dimension to work with. There is no room to 'rotate'. Therefore, naturally, our transformation $\mathcal{T}$ must scale. That is, $\mathcal{T}(-1)$ must be different from $\mathcal{T}(+1)$. (As discussed above $\mathcal{T}(x)$ has to be continuous and invertible for all x, therefore $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$ must be of the same sign). So here, any simple scaling will do, such as $\mathcal{T}(x)=1+(\epsilon-1)(x+1)/2$.

Now consider the case

$$A=\begin{bmatrix}1&0\\0&0\end{bmatrix}, \quad C=\begin{bmatrix}0&0\\1&0\end{bmatrix}. \tag{2.2}$$

Here we have two dimensions. No simple scaling will work. In fact, since we want $\hat{D}$ to be invertible, (and $\mathcal{T}$ to be invertible), we have no choice but to require that

$$\hat{D}\mathcal{T}(-1)^{-1}=A+C(\mathcal{T}(+1)\mathcal{T}(-1)^{-1}) \tag{2.3}$$

must be invertible. Therefore, $\mathcal{T}(+1)\mathcal{T}(-1)^{-1}$ must permute the columns of C. Without loss of generality assume that $\mathcal{T}(-1)=I$. This means that a good choice for $\mathcal{T}(+1)$ is

$$\mathcal{T}(+1)=\begin{bmatrix}0&1\\-1&0\end{bmatrix}. \tag{2.4}$$

Note that we require that the sign of $\det(\mathcal{T}(+1))$ match the sign of $\det(\mathcal{T}(-1))$. (This follows from continuity and invertibility of $\mathcal{T}(x)$ for all x).

So our algorithm for constructing $\mathcal{T}$ must accommodate for both scaling and permutation. We also need to ensure that the determinant of $\mathcal{T}$ be nonzero (i.e., possess the same sign) over the interval $[-1,+1]$. Finally, we would like to keep the condition number of $\mathcal{T}$ low over the interval $[-1,+1]$. (We will see that this last condition can conflict with our desire to keep the condition number of $\hat{D}$ low).

There is a simple necessary condition for $\mathcal{T}$ to exist. No matter what $\mathcal{T}$ is, the column space of $\hat{D}$ is just the column space of D, which is a subset of the column space of the augmented matrix $[A|C]$. Therefore, in order for $\hat{D}$ to be invertible (that is, in order for the column space of $\hat{D}$ to be all of R^n), we must have that the columns of $[A|C]$ span R^n. It turns out (as we will show later), that this simple necessary condition also turns out to be sufficient.

The basic idea behind our algorithm is this: $[A|C]$ has $2n$ columns, but only n of them are needed for the column space to be R^n. We find these n relevant 'dominant' columns. Some of them will come from A, and some will come from C. Sometimes (as in the case of our first example), all (or most) of them will come from one of the boundary matrices. In this case we apply a scaling that will mitigate the effect of the other irrelevant 'non-dominant' columns from the other boundary matrix. Sometimes (as in the case of the second example), some of the dominant columns will come from

one boundary matrix, and some will come from the other boundary matrix. In this case we need to permute the relevant dominant columns into the correct positions so that the sum $\hat{D}$ has full rank. Most of the time both of these effects happen simultaneously. We end up choosing a $\mathcal{T}$ that permutes the dominant columns into the correct positions, and scales the non-dominant columns so that they do not affect the rank of $\hat{D}$.

2.1. Example. Here is an example which illustrates most of the relevant ideas in the algorithm. Let

$$A = \begin{bmatrix} 0 & 1 & 1 & -2 \\ 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & -2 & -1 & 2 \\ 1 & 0 & 1 & 3 \\ 2 & 1 & 0 & 4 \\ 0 & 2 & 2 & 6 \end{bmatrix}, \tag{2.5}$$

so that $A+C$ is given by

$$D = A+C = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 3 & -1 & 1 & 3 \\ 2 & 1 & 0 & 3 \\ 1 & 2 & 2 & 5 \end{bmatrix}, \tag{2.6}$$

which is singular (i.e., the vector $[1,1,1,-1]^{\mathrm{T}}$ is in the null space of D).

The first step in our algorithm is to find the dominant columns of $[A|C]$. This can be carried out either by taking a QRP factorization of $[A|C]$, or a PLU factorization of $[A|C]^{\mathrm{T}}$. We choose the PLU factorization (with partial pivoting) since the pivoting for a QRP factorization is not as easy. This PLU factorization produces matrices P, a $2n \times 2n$ permutation matrix, L, a $2n \times n$ lower triangular matrix, and U, an $n \times n$ upper triangular matrix such that $P[A|C]^{\mathrm{T}} = LU$:

$$P = \left[\begin{array}{cccccccc} 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ \hline 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&1&0&0&0&0&0&0 \end{array}\right], \quad \begin{bmatrix} A^{\mathrm{T}} \\ C^{\mathrm{T}} \end{bmatrix} = \left[\begin{array}{cccc} 0 & 2 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -2 & 0 & -1 & -1 \\ \hline 1 & 1 & 2 & 0 \\ -2 & 0 & 1 & 2 \\ -1 & 1 & 0 & 2 \\ 2 & 3 & 4 & 6 \end{array}\right]$$

$$L = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 2/3 & 1 & 0 \\ -1/2 & 1/3 & -1/4 & 1 \\ \hline -1/2 & 0 & 1/4 & -30/99 \\ 1 & 0 & -1 & -24/99 \\ 1/2 & 1/3 & 1/4 & -51/99 \\ -1/2 & -1/3 & -1/4 & -21/99 \end{array}\right], \quad U = \begin{bmatrix} -2 & 0 & -1 & -1 \\ 0 & 3 & 3 & 5 \\ 0 & 0 & -2 & -7/3 \\ 0 & 0 & 0 & -11/4 \end{bmatrix}. \tag{2.7}$$

First, we look at the diagonal entries of U. Since they are all nonzero, we know that the matrix $[A|C]$ has full rank. Now we can move on to choosing $\mathcal{T}$. The only relevant part of this factorization (other than invertibility of U) is the matrix P. The matrix

P corresponds to a reordering of the rows of $[A|C]^{\mathrm{T}}$, that is, to the columns of $[A|C]$. The first n rows of P correspond to the dominant columns of $[A|C]$. The last n rows of P correspond to the non-dominant columns of $[A|C]$. Now we can read off from the top half of P exactly which columns of $[A|C]$ went into making the final upper triangular matrix U, and which of them were shuffled down to the bottom (and later forgotten). From Equation (2.7) we see that the dominant columns are (in the order of the PLU factorization): (i) the fourth column of A, (ii) the fourth column of C, (iii) the first column of A, (iv) the first column of C. Two of these come from A, and the other two come from C.

The second step of the algorithm involves setting up the preliminary skeletons for $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$. We know that we will want $\mathcal{T}(-1)$ to emphasize the fourth and first columns of A. We also know that we will want $\mathcal{T}(+1)$ to emphasize the fourth and first columns of C. Moreover, we want the sum $A\mathcal{T}(-1)+C\mathcal{T}(+1)$ to be invertible. This leads us to an initial choice of

$$\mathcal{T}(-1)\approx\begin{bmatrix}0&1&0&0\\0&0&0&0\\0&0&0&0\\1&0&0&0\end{bmatrix},\qquad \mathcal{T}(+1)\approx\begin{bmatrix}0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&1&0\end{bmatrix}. \tag{2.8}$$

This immediately implies that the first two columns of $A\mathcal{T}(-1)$ will be the fourth and first columns of A (respectively). Similarly, the last two columns of $C\mathcal{T}(-1)$ will be the fourth and first columns of C (respectively). So if we add $A\mathcal{T}(-1)$ to $C\mathcal{T}(+1)$ we obtain exactly the selection of columns of $[A|C]$ we need for a full column space. Now we also need $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$ to be invertible. As they stand, they are singular, since they are essentially permutation matrices with some columns missing. An easy way to make them invertible is to just 'complete' the permutation structure of $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$ by putting in the missing unit vectors. However, in order to maintain the invertibility of $\hat{D}$, we cannot throw in the missing unit vectors at full strength. Rather, we complete the permutation structure of $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$ with unit vectors of magnitude ϵ. That is, without fixing ϵ we set

$$\mathcal{T}(-1)\approx\begin{bmatrix}0&1&0&0\\0&0&0&\epsilon\\0&0&\epsilon&0\\1&0&0&0\end{bmatrix},\qquad \mathcal{T}(+1)\approx\begin{bmatrix}0&0&0&1\\\epsilon&0&0&0\\0&\epsilon&0&0\\0&0&1&0\end{bmatrix}. \tag{2.9}$$

Here note that if we choose $\epsilon=0$, then we have the case of Equation (2.8), and $\hat{D}$ is invertible, so $\det(\hat{D})\neq 0$. Therefore, since det is a continuous function of matrix entries, there must be some nonzero positive ϵ such that $\hat{D}$ specified by Equation (2.9) is nonsingular. (Note that we have not set ϵ yet, we are just filling in the permutation structure of $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$).

Now, for the third step of the algorithm, we need to ensure that $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$ are compatible. Recall that we need $\mathcal{T}(x)$ to be invertible for all x. Therefore $\det(\mathcal{T}(-1))$ must match $\det(\mathcal{T}(+1))$ in sign. However, in Equation (2.9), we see that the sign of $\mathcal{T}(-1)$ is positive (it is an even permutation). The sign of $\mathcal{T}(+1)$ is negative (it is an odd permutation). We can easily fix this simply by negating one of the components of $\mathcal{T}(+1)$. It does not matter which one we negate, as can be seen

by taking the $\lim_{\epsilon \to 0^+}$. This sets

$$\mathcal{T}(-1) \approx \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \epsilon \\ 0 & 0 & \epsilon & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathcal{T}(+1) \approx \begin{bmatrix} 0 & 0 & 0 & -1 \\ \epsilon & 0 & 0 & 0 \\ 0 & \epsilon & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.10}$$

At this point, we can choose ϵ if we wish. We are guaranteed that an ϵ exists, but in general, it is difficult to find the maximum possible ϵ. Therefore, we just try different values for ϵ until we find one that works. This involves plugging in an ϵ, and then checking the condition number of the corresponding $\hat{D}$. If the condition number is too high, then we halve our value for ϵ and try again. For this we need a fast condition number estimator [4]. We use Haag's 1-norm condition number estimator (which requires one PLU factorization of $\hat{D}$ as well as $O(n^2)$ additional work). In this particular case, $\epsilon = \frac{1}{2}$ works fine. This fixes

$$\mathcal{T}(-1) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathcal{T}(+1) = \begin{bmatrix} 0 & 0 & 0 & -1 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.11}$$

(In practice, it is observed that ϵ almost always ends up within the interval $[\frac{1}{8}, \frac{1}{2}]$).

The fourth step of the algorithm involves finding a way to link $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$ with a continuous nonsingular transformation $\mathcal{T}(x)$. We do this as follows. First define functions

$$\begin{aligned} \gamma(x) &= \epsilon + (x+1)(1-\epsilon)/2, \\ c(x) &= \left[\cos\left(\frac{\pi(x+1)}{4}\right)\right]^{2/4}, \\ s(x) &= \left[\sin\left(\frac{\pi(x+1)}{4}\right)\right]^{2/4}. \end{aligned} \tag{2.12}$$

These are constructed so that $\gamma(x)$ is a linear scaling from ϵ to 1. Similarly, $\gamma(-x)$ scales from 1 to ϵ. And $c(x)$ goes from 1 to 0, while $s(x)$ moves from 0 to 1. (The particular choice of $c(x)$ and $s(x)$ stem from our desire to make the analytical inverse $\mathcal{T}(x)^{-1}$ easy to write down, as we will discuss in more detail later). Now express

$$\mathcal{T}(-1) = \begin{bmatrix} 0 & c(-1) & 0 & 0 \\ 0 & 0 & 0 & c(-1) \\ 0 & 0 & c(-1) & 0 \\ c(-1) & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \gamma(-1) & 0 \\ 0 & 0 & 0 & \gamma(-1) \end{bmatrix}$$

$$\mathcal{T}(+1) = \begin{bmatrix} 0 & 0 & 0 & -s(+1) \\ s(+1) & 0 & 0 & 0 \\ 0 & s(+1) & 0 & 0 \\ 0 & 0 & s(+1) & 0 \end{bmatrix} \cdot \begin{bmatrix} \gamma(-1) & 0 & 0 & 0 \\ 0 & \gamma(-1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

which can be immediately generalized to

$$\mathcal{T}(x) = \begin{bmatrix} 0 & c(x) & 0 & -s(x) \\ s(x) & 0 & 0 & c(x) \\ 0 & s(x) & c(x) & 0 \\ c(x) & 0 & s(x) & 0 \end{bmatrix} \cdot \begin{bmatrix} \gamma(-x) & 0 & 0 & 0 \\ 0 & \gamma(-x) & 0 & 0 \\ 0 & 0 & \gamma(+x) & 0 \\ 0 & 0 & 0 & \gamma(+x) \end{bmatrix}, \tag{2.13}$$

or, more succinctly,

$$\begin{aligned}\mathcal{T}(-1) &= c(-1)P^{\eta_A}\cdot S^{2,\epsilon,1},\\ \mathcal{T}(+1) &= s(+1)P^{\eta_C}\cdot R^{\{4\}}\cdot S^{2,1,\epsilon},\\ \mathcal{T}(x) &= \left[c(x)P^{\eta_A}+s(x)P^{\eta_C}\cdot R^{\{4\}}\right]\cdot S^{2,\gamma(-x),\gamma(+x)},\end{aligned} \tag{2.14}$$

where P^{η_A}, P^{η_C} are permutation matrices, $S^{2,\gamma(-x),\gamma(+x)}$ is a diagonal scaling matrix, and $R^{\{4\}}$ is a diagonal reflection matrix which negates only the fourth column of P^{η_C}. We will precisely define this notation later (see Equation (3.3), Equation (3.4) and Equation (3.7)).

This procedure always works (see Lemma (3.3)), and we can in fact prove that

$$\det(\mathcal{T}(x)) = \pm\left[\Pi_{j=1}^{k}(c(x)^{\alpha_j}+s(x)^{\alpha_j})\right]\gamma(+x)^{n-p}\gamma(-x)^{p} \tag{2.15}$$

where

1. n is the dimension of the system (in the above case $n=4$)
2. $\{\alpha_1,\ldots,\alpha_k\}$ is some partition of n (above we have $k=1$, and $\alpha_1=4$)
3. p is the number of dominant columns coming from A (above we have $p=2$).

Thus, if we choose $c(x)$ and $s(x)$ correctly, $\mathcal{T}(x)$ will always be invertible. (In practice we usually choose c and s to be cosines or sines). Moreover, we can define

$$\tilde{P} = [P^{\eta_A^{-1}}\cdot P^{\eta_C}\cdot R^{\{4\}}] = P^{\eta_A^{-1}\cdot\eta_C}\cdot R^{\{4\}}, \tag{2.16}$$

and notice the following geometric series

$$\begin{aligned}\left(c^3P^{\eta_A^{-1}}-c^2s\tilde{P}\cdot P^{\eta_A^{-1}}+cs^2\tilde{P}^2\cdot P^{\eta_A^{-1}}-s^3\tilde{P}^3\cdot P^{\eta_A^{-1}}\right)\cdot\\ (cP^{\eta_A}+sP^{\eta_C}\cdot R^{\{4\}}) = (c^4I+s^4I) = I.\end{aligned} \tag{2.17}$$

This allows us to immediately write down the inverse of $\mathcal{T}(x)$ in Equation (2.13) as

$$\mathcal{T}^{-1} = \begin{bmatrix}\frac{1}{\gamma(-x)} & 0 & 0 & 0\\ 0 & \frac{1}{\gamma(-x)} & 0 & 0\\ 0 & 0 & \frac{1}{\gamma(+x)} & 0\\ 0 & 0 & 0 & \frac{1}{\gamma(+x)}\end{bmatrix}\cdot\begin{bmatrix}cs^2 & s^3 & -c^2s & c^3\\ c^3 & c^2s & s^3 & -cs^2\\ -c^2s & -cs^2 & c^3 & s^3\\ -s^3 & c^3 & cs^2 & -c^2s\end{bmatrix}. \tag{2.18}$$

It is the geometric series Equation (2.17) and the resulting inverse Equation (2.18) which motivate our original choice for $c(x)$ and $s(x)$ (see Equation (2.12)). If the composite permutation $P^{\eta_A^{-1}\eta_C}$ had been a product of disjoint 2-cycles (instead of a single 4-cycle), we would have chosen $c(x)$ and $s(x)$ differently (see Lem. 3.3 for more details).

It should be noted that sometimes there is a conflict between optimization of the condition number of $\hat{D}$ and optimization of the condition number of $\mathcal{T}$. In certain scenarios (such as Equation (2.2)), the condition number of $\hat{D}$ is optimized by choosing ϵ very small. However, this results in a poorly conditioned scaling matrices $S(x)$, and hence a poorly conditioned $\mathcal{T}(x)$. In these cases it is usually best to compromise, and choose ϵ so that the condition number of both $\hat{D}$ and $S(x)$ are reasonably low.

3. Automated coordinate change: detailed explanation

Our algorithm for constructing $\hat{D}$ and $\mathcal{T}$ involves permutations, reflections and scalings. The algorithm is structured such that the determinant of $\mathcal{T}$ is nonzero over the interval $[-1,+1]$, and the inverse $\mathcal{T}(x)^{-1}$ can be easily constructed. As partially illustrated by the example of Section 2.1, the basic ideas behind our algorithm are:

1. Perform a PLU decomposition of the augmented matrix $[A|C]^{\mathrm{T}}$ to find the n relevant 'dominant' columns. (Compare with Equations (2.5)−−(2.7).)
2. Construct the corresponding permutation matrices, which shuffle the relevant 'dominant' columns into the correct positions. (Compare with Equations (2.8)−−(2.9).)
3. By inspecting the structure of the permutation matrices, design appropriate reflection matrices which ensure that the derivative of $\mathcal{T}(x)$ is never 0. (Compare with Equation (2.10).)
4. Fix the correct scalar paths and scaling matrices so that $\mathcal{T}$ will be easy to invert. (Compare with eqs. (2.12)−−(2.13).)
5. Put all the ingredients together and define $\mathcal{T}(x)$. (Compare with Equation (2.14)). If required, we can write down the analytical inverse of $\mathcal{T}(x)$. (Compare with Equation (2.17).)

Here we will further detail each step and along the way provide a proof that this algorithm works whenever the matrix $[A|C]$ has full rank (i.e., whenever a valid transformation exists).

3.1. Step 1: determine the dominant columns. First we perform a PLU decomposition:

$$\begin{aligned} P\cdot[A|C]^{\mathrm{T}} &= L\cdot U, \\ P &= 2n\times 2n \quad \text{a permutation matrix,} \\ L &= 2n\times n \quad \text{a lower triangular matrix,} \\ U &= n\times n \quad \text{a upper triangular matrix.} \end{aligned} \tag{3.1}$$

We inspect the first n rows of P. The nonzero entries correspond to the n dominant columns of $[A|C]$. We record these column indices and divide them into two groups:

1. $i_1^A,\ldots,i_p^A$ correspond to the p dominant columns of A;
2. $i_{p+1}^C,\ldots,i_n^C$ correspond to the $n-p$ dominant columns of C.

We then record the column indices of the irrelevant columns of A and C:

1. $i_{p+1}^A,\ldots,i_n^A$ correspond to the $n-p$ irrelevant columns of A;
2. $i_1^C,\ldots,i_p^C$ correspond to the p irrelevant columns of C.

3.2. Step 2: construct permutation matrices. We define permutations

$$\eta_A(j)=i_j^A \;\; \eta_C(j)=i_j^C \tag{3.2}$$

and permutation matrices

$$P_{ij}^{\eta_A}=\delta_{i,\eta_A(j)} \;\; P_{ij}^{\eta_C}=\delta_{i,\eta_C(j)}. \tag{3.3}$$

The permutation matrix P^{η_A} shuffles the p dominant columns of A to the front, and P^{η_C} shuffles the $n-p$ dominant columns of C to the back. The first p columns of

AP^{η_A} and the last $n-p$ columns of CP^{η_C} can be combined to make an invertible matrix.

Now we concentrate on scaling down the effect of the irrelevant columns of $[A|C]$. We define scaling matrices

$$S_{ij}^{p,\mu,\nu} = \begin{cases} \mu\delta_{ij} & j \le p \\ \nu\delta_{ij} & j > p. \end{cases} \tag{3.4}$$

Now $AS^{p,\mu,\nu}$ is the matrix A with the first p columns multiplied by μ and the remaining $n-p$ columns multiplied by ν. With our notation, the nonzero columns of $AP^{\eta_A}S^{p,1,0}$ are simply the p dominant columns of A. Similarly, the nonzero columns of $CP^{\eta_C}S^{p,0,1}$ are the $n-p$ dominant columns of C. The sum

$$AP^{\eta_A}S^{p,1,0} + CP^{\eta_C}S^{p,0,1} \tag{3.5}$$

is invertible by construction. Unfortunately, we cannot simply choose $\mathcal{T}(-1) = P^{\eta_A}S^{p,1,0}$ and $\mathcal{T}(+1) = P^{\eta_C}S^{p,0,1}$, since $S^{p,1,0}, S^{p,0,1}$ are not invertible. Nevertheless, the sum

$$AP^{\eta_A}S^{p,1,\epsilon} + CP^{\eta_C}S^{p,\epsilon,1} \tag{3.6}$$

is invertible for some $\epsilon > 0$ since the determinant is a continuous function of the matrix coefficients. We do not specify ϵ just yet, but later we will find ϵ numerically simply by trying different values until we find one that works. In practice we almost always find $\epsilon \in [1/8, 1/2]$, and so this step is neither time-consuming nor conducive to ill-conditioned coordinate-transformations. The structure of Equation (3.6) suggests a choice of $\mathcal{T}(-1) = P^{\eta_A}S^{p,1,\epsilon}$ and $\mathcal{T}(+1) = P^{\eta_C}S^{p,\epsilon,1}$. However, if we make this choice, we may not be able to construct an invertible matrix path $\mathcal{T}(x)$ between these two boundary values. In particular, the determinants of $P^{\eta_A}S^{p,1,\epsilon}$ and $P^{\eta_C}S^{p,\epsilon,1}$ must have the same sign.

3.3. Step 3: design reflection matrices. To ensure that $\det(\mathcal{T}(-1))$ and $\det(\mathcal{T}(+1))$ have the same sign, we design reflection matrices:

$$R_{ij}^{\mathcal{L}} = \begin{cases} \delta_{ij} & j \notin \mathcal{L} \\ -\delta_{ij} & j \in \mathcal{L}, \end{cases} \tag{3.7}$$

where $\mathcal{L}$ can be any set of indices. The matrix $AR^{\mathcal{L}}$ is just the matrix A where the columns referred to by $\mathcal{L}$ have been negated. We form the matrices

$$\mathcal{T}(-1) = P^{\eta_A}S^{p,1,\epsilon},\ \mathcal{T}(+1) = P^{\eta_C}S^{p,\epsilon,1}R^{\mathcal{L}}, \tag{3.8}$$

where $\mathcal{L}$ has not yet been chosen. It is well known that GL_n has two path-connected components, namely, matrices with positive determinant and matrices with negative determinant [15]. We can easily choose $\mathcal{L}$ (either empty or a singleton set) to ensure that there *exists* a path between $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$. However, we will choose $\mathcal{L}$ in a way that makes the path $\mathcal{T}(x)$ easy to invert.

To start we construct the scalar paths

$$\begin{aligned} c(x,m) &= \cos(\pi(x+1)/4)^{2/m}, \\ s(x,m) &= \sin(\pi(x+1)/4)^{2/m}, \\ \gamma(x) &= \epsilon + (x+1)(1-\epsilon)/2, \end{aligned} \tag{3.9}$$

and the matrix paths

$$\begin{aligned} c(x) &= \mathrm{diag}(c(x,m_1),\ldots,c(x,m_n)),\\ s(x) &= \mathrm{diag}(s(x,m_1),\ldots,s(x,m_n)),\\ P(x) &= P^{\eta_A} c(x) + P^{\eta_C} R^{\mathcal{L}} s(x),\\ S(x) &= S^{p,\gamma(-x),\gamma(+x)},\\ \mathcal{T}(x) &= P(x)S(x). \end{aligned} \tag{3.10}$$

Here $c(x,m), s(x,m)$ are positive continuous paths linking 1 to 0, and $\gamma(x)$ is a positive continuous path linking ϵ to 1. The matrix paths $c(x), s(x)$ simply link I to 0 with different paths for different indices. At this point $\epsilon, \mathcal{L}$ and m_i are not chosen, but we will fix all of these parameters later. We will eventually choose ϵ and $\mathcal{L}$ so that $\hat{D}$ is invertible, and we will choose the $m_1,\ldots,m_n$ to make $\mathcal{T}(x)$ easy to invert. With this current representation for $\mathcal{T}$ we have

$$\hat{D} = A\mathcal{T}(-1) + C\mathcal{T}(+1) = AP^{\eta_A} S^{p,1,\epsilon} + CP^{\eta_C} S^{p,\epsilon,1} R^{\mathcal{L}}. \tag{3.11}$$

Note that $\hat{D}$ will be invertible for any choice of $\mathcal{L}$. This is the invertible boundary matrix we want. Since $\det(S(x)) \geq \epsilon^n$ for every x, $\mathcal{T}(x)$ will be invertible if and only if $P(x)$ is invertible.

In order to determine the underlying permutation structure of $P(x)$, we apply a preliminary transformation

$$(P^{\eta_A})^{-1} P(x) = c(x) + (P^{\eta_A})^{-1} P^{\eta_C} R^{\mathcal{L}} s(x). \tag{3.12}$$

Noting that

$$P^{\eta_2} P^{\eta_1} = P^{\eta_2 \cdot \eta_1}, \tag{3.13}$$

we write

$$\hat{P}(x) = (P^{\eta_A})^{-1} P(x) = c(x) + P^{\hat{\eta}} R^{\mathcal{L}} s(x), \tag{3.14}$$

where we have defined

$$\hat{\eta} = \eta_A^{-1} \eta_C. \tag{3.15}$$

We need to show that $\hat{P}(x)$ is invertible. To do this we decompose $\hat{\eta}$ into k disjoint cycles (cyclic permutations) [11],

$$\hat{\eta} = \alpha_k \cdot \alpha_{k-1} \cdots \alpha_1, \tag{3.16}$$

where each α_i has length l_i. We associate to each α_i the set of indices (coordinates) β_i that it permutes. The β_i form a partition of the set $\{1,\ldots,n\}$.

Note that if we write $\hat{\eta}$ as the product of k disjoint cycles $\hat{\eta} = \prod_{i=k}^{1} \alpha_i$ each of length l_i, then $P^{\hat{\eta}} = P^{\alpha_k} \cdots P^{\alpha_1}$ is a matrix with k blocks, each of size $l_i \times l_i$. If we associate with each α_i its set β_i of permuted indices, then the blocks of P^{η} are those associated with the coordinate groups β_j. This block matrix becomes visually obvious if we reorder coordinates such that the indices in each cycle α_{j+1} are larger than the indices in the previous cycle α_j. (Alternatively, we could require the entries of β_{j+1} to be larger than the entries of β_j).

Armed with this cycle decomposition of $\hat{\eta}$, we write:

$$\hat{P} = c(x) + P^{\alpha_k} \cdots P^{\alpha_1} R^{\mathcal{L}} s(x). \tag{3.17}$$

It is convenient to define the restriction

$$[P\big|_{\beta}]_{ij} = \begin{cases} P_{ij} & i,j \in \beta \\ \delta_{ij} & \text{otherwise.} \end{cases} \tag{3.18}$$

Note that

$$P^{\alpha_i}\big|_{\beta_i} = P^{\alpha_i} \tag{3.19}$$

and

$$R^{\mathcal{L}}\big|_{\beta_i} = R^{\mathcal{L} \bigcap \beta_i}. \tag{3.20}$$

Note also that since the β_i form a partition of $\{1,\ldots,n\}$, the product $\prod_{i=k}^{1} A\big|_{\beta_i}$ is simply the matrix A restricted to the blocks β_i (with zeros elsewhere). Using Equation (3.18) we rewrite Equation (3.17) as

$$\begin{aligned} \hat{P} &= \prod_{i=k}^{1} c(x)\big|_{\beta_i} + \prod_{i=k}^{1} (P^{\alpha_i} R^{\mathcal{L}} s(x))\big|_{\beta_i} \\ &= \prod_{i=k}^{1} \left(c(x) + P^{\alpha_i} R^{\mathcal{L}} s(x)\right)\big|_{\beta_i} \\ &= \prod_{i=k}^{1} \left(c(x)\big|_{\beta_i} + P^{\alpha_i} R^{\mathcal{L} \bigcap \beta_i} s(x)\big|_{\beta_i}\right). \end{aligned} \tag{3.21}$$

Essentially, the matrix path $\hat{P}(x)$ is equivalent to a product of simpler matrix paths which are each built from disjoint cyclic permutations. So the determinant of $\hat{P}$ is given by the product of the determinant of the coordinate blocks

$$\det(\hat{P}) = \prod_{i=k}^{1} \det\left(c(x)\big|_{\beta_i} + P^{\alpha_i} R^{\mathcal{L} \bigcap \beta_i} s(x)\big|_{\beta_i}\right). \tag{3.22}$$

To prove that each of these coordinate blocks has nonzero determinant we use the following lemma:

LEMMA 3.1. *If α is a single cycle of length n operating on every coordinate, and $c(x) = c(x,n)I$ and $s(x) = s(x,n)I$, then*

$$\det(c(x) + P^{\alpha} R^{\mathcal{L}} s(x)) = c^n(x,n) + (-1)^{n-1} \det(R^{\mathcal{L}}) s^n(s,n).$$

Proof. We use the following definition of determinant: Given an $n \times n$ matrix M, let $\varepsilon(\zeta)$ be the fully alternating antisymmetric n-tensor:

$$\varepsilon(\zeta) = \begin{cases} 0 & \zeta \text{ not an } n\text{-cycle} \\ \text{parity of } \zeta & \zeta \text{ is an } n\text{-cycle.} \end{cases} \tag{3.23}$$

The determinant of M is given by

$$\begin{aligned}\det(M) &= \sum_{\zeta\in S_n} \varepsilon(\zeta) M_{1,\zeta(1)} M_{2,\zeta(2)} \cdots M_{n,\zeta(n)} \\ &= \sum_{\zeta\in S_n} \varepsilon(\zeta) \Pi_{j=1}^{j=n} M_{j,\zeta(j)}. \end{aligned} \tag{3.24}$$

Now when considering terms of

$$M = c(x) + P^{\alpha} R^{\mathcal{L}} s(x)$$

we have the following possibilities for $\Pi_{j=1}^{j=n} M_{j,\zeta(j)}$ and $\varepsilon\zeta$:

1. The single term $c^n(x,n)$ corresponds to $\zeta(j)=j$, and so $\varepsilon(\zeta)=1$.
2. The single term $\det(R^{\mathcal{L}}) s^n(x,n)$ corresponds to $\zeta(j)=\alpha(j)$ and so $\varepsilon(\zeta)=$ parity(α).
3. The cross terms $\pm c^m(x,n) s^{n-m}(x,n)$ for $0<m<n$ include some $s(x,n)$-terms and some $c(x,n)$-terms from M.

We examine this last case. Since α is a full n-cycle, we know that each row and each column of M has one $s(x,n)$-term and one $c(x,n)$-term. We also know that the $c(x,n)$-terms are on the diagonal. Therefore, any term of the form $c^m(x,n)s^{n-m}(x,n)$ must take m $c(x,n)$-terms from the diagonal of M. The columns containing the m $c(x,n)$-terms must also contain m $s(x,n)$-terms, since α is a permutation. Similarly, the rows containing the m $c(x,n)$-terms must contain m $s(x,n)$-terms.

If there were only m $s(x,n)$-terms in those particular m rows and columns of M altogether, then they would all have to be in the $m\times m$ subblock of M which contains the m $c(x,n)$-terms. However, since α is an n-cycle, no $m\times m$ subblock of P^{α} can have m $s(x,n)$-terms in it. So there must be at least $(m+1)$ $s(x,n)$-terms in those m particular rows and columns of M. That leaves only $(n-m-1)$ $s(x,n)$-terms in the remaining $(n-m)\times(n-m)$ block submatrix of M.

In order for $\varepsilon(\zeta)$ to be nonzero, the $(n-m)$ $s(x,n)$-terms must have come from the remaining $(n-m)\times(n-m)$ block submatrix of M. This is not possible, and so $\varepsilon(\zeta)$ must be zero. So

$$\begin{aligned}\det(M) &= c^n(x,n) + \det(R^{\mathcal{L}})\text{parity}(\alpha) s^n(x,n) \\ &= c^n(x,n) + (-1)^{n-1}\det(R^{\mathcal{L}}) s^n(x,n).\end{aligned}$$

□

Under the assumptions of the lemma, we can easily choose $\mathcal{L}$ so that M is invertible. Namely, we start with an empty $\mathcal{L}$, and refer back to Equation (3.16). Then, we simply inspect the parity of each element of the disjoint cycle decomposition $\alpha_k\cdots\alpha_1=\hat{\eta}$. For each cycle α_i with negative parity, we add an element of β_i to $\mathcal{L}$. This will guarantee that the analog of the determinant in Lem. 3.1 will be nonzero.

So, to recapitulate, at this point we have a coordinate transformation $\mathcal{T}(x)$ given by

$$\begin{aligned} c(x) &= \text{diag}(c(x,m_1),\ldots,c(x,m_n)), \\ s(x) &= \text{diag}(s(x,m_1),\ldots,s(x,m_n)), \\ P(x) &= P^{\eta_A} c(x) + P^{\eta_C} R^{\mathcal{L}} s(x), \\ S(x) &= S^{p,\gamma(-x),\gamma(+x)}, \\ \mathcal{T}(x) &= P(x)S(x), \end{aligned} \tag{3.25}$$

where $\mathcal{L}$ is fixed so that, for any set of positive m_i and sufficiently small ϵ, the matrix $\mathcal{T}(x)$ will be invertible (compare with Equation (3.10)).

3.4. Step 4: write down the inverse. The next step is to fix the m_i so that the inverse $\mathcal{T}(x)^{-1}$ is easy to write down.

LEMMA 3.2. *Assume α is a single cycle of length n operating on every coordinate. Also assume that $c(x) = c(x,n)I$ and $s(x) = s(x,n)I$. If α has positive parity (n is odd), then let $\mathcal{L}$ be empty. If α has negative parity (n is even), then let $\mathcal{L}$ be a singleton set of any index permuted by α. Then*

$$\begin{aligned} V^\alpha &= (c(x,n)I + s(x,n)P^\alpha R^{\mathcal{L}})^{-1} \\ &= \sum_{i=0}^{n-1} (-1)^i c(x,n)^{n-1-i} s(x,n)^i (P^\alpha R^{\mathcal{L}})^i, \end{aligned} \tag{3.26}$$

and

$$\|V\|_1 = \|V\|_\infty = \sum_{i=0}^{n-1} c(x,n)^{n-1-i} s(x,n)^i < n. \tag{3.27}$$

Proof. Assume first that α has positive parity. Then n is odd, and $R^{\mathcal{L}} = I$. We have

$$(c(x,n)I + s(x,n)P^\alpha R^{\mathcal{L}})V^\alpha = c(x,n)^n I + s(x,n)^n (P^\alpha)^n. \tag{3.28}$$

Since α is an n-cycle, $(P^\alpha)^n = I$ and we have

$$(c(x,n)I + s(x,n)P^\alpha R^{\mathcal{L}})V^\alpha = c(x,n)^n I + s(x,n)^n I = I. \tag{3.29}$$

Now if α has negative parity, then n is even. Without loss of generality, assume that $\mathcal{L} = \{1\}$ and $R^{\mathcal{L}}$ reflects the first column of the preceding matrix. In this case

$$(c(x,n)I + s(x,n)P^\alpha R^{\mathcal{L}})V^\alpha = c(x,n)^n I - s(x,n)^n (P^\alpha R^{\mathcal{L}})^n. \tag{3.30}$$

The term $(P^\alpha R^{\mathcal{L}})^n$ is formed by starting with the identity matrix I, and then performing n permutation-reflections. Each permutation-reflection shuffles the columns around via the n-cycle α, and then negates the first resultant column. After n of these permutation-reflections every original column has been in each column position, and has been negated exactly once (since α is an n-cycle). Therefore $(P^\alpha R^{\mathcal{L}})^n = -I$, and we have

$$(c(x,n)I + s(x,n)P^\alpha R^{\mathcal{L}})V^\alpha = c(x,n)^n I - s(x,n)^n (-I) = I. \tag{3.31}$$

Using similar reasoning, we can deduce the structure of V^α. Each term

$$c(x,n)^{n-1-i} s(x,n)^i (P^\alpha R^{\mathcal{L}})^i$$

is composed of i permutation-reflections as described above. The matrix V is formed by summing up the results of $n-1$ successive permutation-reflections. Therefore, each column of V has n unique nonzero entries, with magnitudes

$$c(x,n)^{n-1} s(x,n)^0, \ldots, c(x,n)^0 s(x,n)^{n-1}.$$

The same result holds for the rows of V. Thus we can bound the ∞-norm condition number of $c(x,n)I+s(x,n)P^{\alpha}R^{\mathcal{L}}$ by $2n$. □

Lem. 3.2 immediately suggests a method for choosing the set $\{m_1,\ldots,m_n\}$. Referring back to the disjoint cyclic decomposition $\alpha_k\cdots\alpha_1=\eta$ (see Equation (3.16)), we can simply set m_i to be the length l_j of the cyclic permutation α_j which acts on the index $i\in\beta_j$. Then, the construction of $c(x)$ and $s(x)$ via Equation (3.10) will allow for the manipulations of Lem. 3.2, and the inverse $\mathcal{T}(x)^{-1}$ will be easy to write down.

To be more precise, we can simultaneously choose $\mathcal{L}$ and the m_i and write down the inverse $\mathcal{T}(x)^{-1}$ as follows:

LEMMA 3.3. *Assume $\hat{\eta}=\alpha_k\cdots\alpha_1$ is a disjoint cycle decomposition with corresponding coordinate blocks $\{\beta_k,\ldots,\beta_1\}$ of respective lengths $\{l_k,\ldots,l_1\}$. Choose m_i to be the length l_j of the cycle α_j that acts on index i. Construct $c(x),s(x)$ as in Equation (3.10). Initialize $\mathcal{L}$ to be the empty set. For each cycle α_i with negative parity, append an index permuted by α_i to $\mathcal{L}$. Now we have*

$$\begin{aligned}\det(\hat{P}(x)) &= \pm\prod_{i=k}^{1} c^{l_i}(x,l_i)+\det(R^{\mathcal{L}\bigcap\beta_i})\mathrm{parity}(\alpha_{\mathrm{i}})\mathrm{s}^{\mathrm{l_i}}(\mathrm{x},\mathrm{l_i})\\ &= \pm 1.\end{aligned} \tag{3.32}$$

We can write $\hat{P}(x)^{-1}$ as

$$\begin{aligned}V^{\alpha_i}(x) &= \left(\sum_{j=0}^{l_i-1}(-1)^j c^{l_i-1-j}(x,l_i)s^j(x,l_i)(P^{\alpha_i}R^{\mathcal{L}\bigcap\beta_i})^j\right)\Big|_{\beta_i},\\ \hat{P}(x)^{-1} &= \prod_{i=1}^{k} V^{\alpha_i}(x).\end{aligned} \tag{3.33}$$

Moreover, we can bound the ∞-norm condition number of $\hat{P}(x)$

$$\kappa_{\infty}(\hat{P}(x))<2n. \tag{3.34}$$

This establishes the invertibility of $\hat{P}(x)$, $P(x)$ and $\mathcal{T}(x)$. Using Equation (3.10) and Equation (3.33), we can construct

$$\begin{aligned}\mathcal{T}(x)^{-1} &= S(x)^{-1}P(x)^{-1}\\ &= S^{p,1/\gamma(-x),1/\gamma(+x)}\hat{P}(x)^{-1}(P^{\eta_A})^{-1}\\ &= S^{p,1/\gamma(-x),1/\gamma(+x)}(\prod_{i=1}^{k}V^{\alpha_i}(x))(P^{\eta_A})^{-1}.\end{aligned} \tag{3.35}$$

Using Equation (3.34), we can bound the ∞-norm condition number of $\mathcal{T}(x)$.

$$\begin{aligned}\|\mathcal{T}(x)^{-1}\|_{\infty} &< 2\epsilon^{-n}(2n),\\ \kappa_{\infty}(\mathcal{T}(x)) &< 8n\epsilon^{-n}.\end{aligned} \tag{3.36}$$

3.5. Step 5: Construct $\mathcal{T}(x)$. Finally, we have fixed $\mathcal{L}$ and the m_i, and are able to construct $\mathcal{T}(x)$ (see Equation (3.25)) and $\mathcal{T}(x)^{-1}$ (see Equation (3.35)). The final remaining task is to fix the parameter ϵ which has been floating throughout this procedure. To do so we simply set $\epsilon = 1/2$ and construct

$$\hat{D} = A\mathcal{T}(-1) + C\mathcal{T}(+1). \tag{3.37}$$

We can check the condition number of $\hat{D}$ using any standard tool (such as Haag's condition number estimator [4]). If $\hat{D}$ is nearly singular, we go back and halve ϵ, check the condition number of $\hat{D}$ again, and repeat. This fixes all the parameters, and finalizes the coordinate transformation.

Note that we can easily make our algorithm more flexible. For example:

1. There is no reason to choose only one ϵ in Equation (2.9), or to set the other entries to 1. We could choose a different ϵ for each column of $\mathcal{T}(-1)$ or $\mathcal{T}(+1)$, choosing some values near 1, and other values as small as necessary. This would result in a different scaling for each column, but these scalings could be chosen to minimize the condition number of $\hat{D}$, or to minimize the condition number of $\mathcal{T}$.
2. In step 2, we can also choose the signs of the columns of $\mathcal{T}(-1)$ and $\mathcal{T}(+1)$ to minimize the condition number of $\hat{D}$. (Ideally, we do not want the angle between any two columns of $\hat{D}$ to be too small). We still have to ensure that the determinants at both endpoints match in sign, but other than than that, we have freedom to choose.
3. In Equation (2.12) we do not have to scale linearly, or choose sines and cosines. We can always choose functions that are flatter in certain regions and steeper in others. This allows us to somewhat compensate (and have a relatively constant $\mathcal{T}$) in regions where $\hat{f}$ and $\hat{p}$ are changing quickly.

In addition, note that, with the analytical apparatus constructed above, we can easily construct an alternative coordinate transformation as follows. First we define the simple linking matrices $P^{\tau}_{[x_1,x_2]}(x)$ which connect transpositions over a small interval.

$$\begin{gathered}
\tau \in S_n \text{ is a transposition,} \\
[x_1, x_2] \text{ is some interval,} \\
l_\tau = \text{ one of the indices swapped by } \tau \quad , \\
P^{\tau}_{[x_1,x_2]}(x) = c\left(\left(x - \frac{x_1+x_2}{2}\right)\frac{x_2-x_1}{2}\right) I_{n\times n} + s\left(\left(x - \frac{x_1+x_2}{2}\right)\frac{x_2-x_1}{2}\right) P^{\tau} R^{\{l_\tau\}}.
\end{gathered} \tag{3.38}$$

So $P^{\tau}_{[x_1,x_2]}(x_1) = I$ is the identity matrix, and $P^{\tau}_{[x_1,x_2]}(x_2) = P^{\tau}R^{\{l_\tau\}}$ is the identity matrix with two rows swapped (those specified by τ) and one negated. $P^{\tau}_{[x_1,x_2]}(x)$ merely provides a nonsingular path between these two endpoints.

Now we can decompose $\eta_A^{-1}\eta_C = \tau_k \cdots \tau_1$ as the product of k transpositions, and let l_j be one of the indices swapped by τ_j. We also choose $k+1$ points

$$-1 = x_0 < x_1 < \ldots < x_{k-1} < x_k = 1$$

TABLE 4.1. *Results for 6 by 6 boundary matrices, 500 trials each*

Rank of A and C	Rank of D	$\bar{\kappa}(\hat{D})$	$\kappa_{\max}(\hat{D})$	$\bar{\kappa}(\mathcal{T}(x))$	$\kappa_{\max}(\mathcal{T}(x))$	$\bar{\kappa}(\mathcal{T}'(x))$	$\kappa_{\max}(\mathcal{T}'(x))$
6	4	4×10^1	2×10^2	8×10^0	2×10^1	1×10^2	6×10^4
6	2	7×10^1	3×10^3	8×10^0	2×10^1	2×10^2	1×10^4
6	0	4×10^3	9×10^5	6×10^0	6×10^0	8×10^0	9×10^0
5	4	5×10^1	1×10^3	8×10^0	2×10^1	1×10^2	5×10^3
5	2	3×10^2	2×10^4	8×10^0	3×10^1	1×10^2	7×10^3
4	4	2×10^2	6×10^3	8×10^0	2×10^1	2×10^2	5×10^3

inside the interval $[-1,1]$. Now we can write

$$P(x)=\left\{\begin{array}{r} P^{\eta_A}P^{\tau_1}_{[x_0,x_1]}(x)\ x\in[x_0,x_1] \\ P^{\eta_A}P^{\tau_1}R^{\{l_1\}}P^{\tau_2}_{[x_1,x_2]}(x)\ x\in[x_1,x_2] \\ P^{\eta_A}P^{\tau_2\tau_1}R^{\{l_1,l_2\}}P^{\tau_3}_{[x_2,x_3]}(x)\ x\in[x_2,x_3] \\ \vdots\vdots \\ P^{\eta_A}P^{\tau_k\cdots\tau_1}R^{\{l_1,\ldots,l_k\}}P^{\tau_k}_{[x_{k-1},x_k]}(x)\ x\in[x_{k-1},x_k] \end{array}\right\}. \tag{3.39}$$

Now $P(x)$ goes through k steps, swapping two columns with each step. Note that with the construction of Equation (3.39), it is convenient to choose c and s such that $c+s=1$. This means that each $P^{\tau}_{[x_1,x_2]}$ acts only on the columns swapped by τ, and does not scale any other columns. (Note that $c(x)=\cos^2((x+1)\pi/4)$ and $s(x)=\sin^2((x+1)\pi/4)$ will do the job).

Construction (3.39) is useful because it is especially easy to invert. Unfortunately, the $\mathcal{T}$ constructed is not necessarily as smooth as c and s. (In the case of $c=\cos^2$,$s=\sin^2$, $\mathcal{T}$ is only C^1). Here however, we can choose c and s such that $\mathcal{T}$ is as smooth as we desire.

4. Numerical examples

We test the performance of this method by constructing several boundary matrices A and C such that their sum D is singular. We construct a coordinate transformation $\mathcal{T}(x)$ on the interval $[-1,1]$ for each pair of boundary matrices. We measure the average condition number and maximum condition number of $\hat{D}=A\mathcal{T}(-1)+C\mathcal{T}(+1)$, $\mathcal{T}(x)$ and $\mathcal{T}'(x)$ over several trials. The process used to construct singular boundary matrices is as follows:

1. Choose the dimension n of the boundary matrices.
2. Choose the rank $j\leq n$ of A and C.
3. Choose an even number d for the rank of $D=A+C$.
4. Let $k=j-\frac{d}{2}$.
5. Construct $2j-k$ rank-1 outer products $u_i\cdot v_i^{\mathrm{T}}$ at random.
6. Set $A=\sum_{i=1}^{j}u_i\cdot v_i^{\mathrm{T}}$.
7. Set $C=\sum_{i=j-k+1}^{j}-u_i\cdot v_i^{\mathrm{T}}+\sum_{i=j+1}^{2j-k}u_i\cdot v_i^{\mathrm{T}}$.
8. Now in general, $D=A+C$ will have rank $2(j-k)=d$.

In general, $[A|C]$ will have full rank if $2j-k\geq n$. This is equivalent to requiring $2j+d\geq 2n$. Table (4.1) contains the results for $n=6$ and values of j ranging from 6 to 4. Each test was carried out using 500 trials. The condition numbers of $\hat{D}$,$\mathcal{T}(x)$ and $\mathcal{T}'(x)$ are all fairly low, even in the worst cases. The condition number of $\hat{D}$ is

highest when D is singular. However, in this case the condition numbers of $\mathcal{T}$ and $\mathcal{T}'$ are at their lowest.

5. Algorithm Psuedocode

Assume that we are given matrices $A \in R^{n\times n}$ and $C \in R^{n\times n}$ such that their sum D is singular. We focus on the problem of constructing a differentiable path $\mathcal{T}:[-1,1]\to R^{n\times n}$ such that $\hat{D} = A\mathcal{T}(-1) + C\mathcal{T}(+1)$ is nonsingular and $\mathcal{T}(x)$ is invertible for all x. Note that the column space of $\hat{D}$ lies within the column space of $[A|C]$. Therefore $[A|C]$ must have full rank in order for $\mathcal{T}(x)$ to exist. This simple necessary condition for the existence of $\mathcal{T}$ proves to be sufficient. The apparatus we construct revolves around permutation, reflection and scaling matrices:

ALGORITHM 5.1. *Build* $\mathcal{T}$

1. **Determine the dominant columns of [A—C]** *(See Section 3.1).*
   - *Read A and C as input.*
   - *Perform a PLU decomposition $P[A|C]^T = LU$. Inspect the diagonal entries of U, and ascertain that $[A|C]$ has full rank.*
   - *Inspect the upper $n\times 2n$ submatrix of P. There will be n nonzero entries. Divide the column indices of these entries into two sets. Set 1 of length p is the set of column indices between 1 and n. Set 2 of length $n-p$ is the set of column indices between $n+1$ and $2n$. Subtract n from each element of Set 2.*
2. **Construct the permutation matrices** *(See Section 3.2).*
   - *Create a permutation matrix P^{η_A} such that the row indices of the first p columns of P^{η_A} are each elements of Set 1.*
   - *Create a permutation matrix P^{η_C} such that the row indices of the last $n-p$ columns of P^{η_C} are each elements of Set 2.*
3. **Construct the reflection matrices** *(See Section 3.3 and Section 3.4).*
   - *Inspect the permutation $(P^{\eta_A})^{-1}P^{\eta_C}$.*
     Initialize $\mathcal{L} = \{\}$ to be the empty set.
     Initialize $\{m_1,\ldots,m_n\}$ to be 0.
     Decompose this permutation into k disjoint cycles
     $(P^{\eta_A})^{-1}P^{\eta_C} = \alpha_k\cdots\alpha_1$,
     each of length l_i acting on coordinate sets β_i.
     for $i = 1,\ldots,k$
     if α_i has negative parity, append an index permuted by α_i to $\mathcal{L}$.
     for each index j acted on by α_i, set $m_j = l_i$.
     end for
     Define $R^{\mathcal{L}}$ to be a diagonal matrix with entries of -1 in the columns listed in $\mathcal{L}$, and entries of 1 in the other columns.
4. **Construct $\mathcal{T}$ and fix ϵ** *(See Section 3.5).*
   - *Define $\gamma(x) = \epsilon + (1+x)(1-\epsilon)/2$. We will choose ϵ later.*
   - *Define $c(x,m) = \cos(\pi(1+x)/4)^{2/m}$, $s(x,m) = \sin(\pi(1+x)/4)^{2/m}$.*
   - *Define $c(x) = \mathrm{diag}(c(x,m_1),\ldots,c(x,m_n))$.*
   - *Define $s(x) = \mathrm{diag}(s(x,m_1),\ldots,s(x,m_n))$.*
   - *Define $S^{p,\mu,\nu}$ to be a diagonal matrix with μ for the first p entries and ν in the final $n-p$ entries.*
   - *Define $\mathcal{T}(x) = (P^{\eta_A}c(x) + P^{\eta_C}R^{\mathcal{L}}s(x))S^{p,\gamma(x),\gamma(-x)}$.*
   - *Initialize $\epsilon = 1/2$*
     while $\{AP^{\eta_A}S^{p,1,\epsilon} + CP^{\eta_C}S^{p,\epsilon,1}$ is nearly singular$\}$

$\epsilon = \epsilon/2$

end while

- *Now* $\mathcal{T}(x)$ *is a differentiable invertible matrix path from* A *to* C.

5. **If required, construct** $\mathcal{T}^{-1}$ *(See Section 3.4).*
   - *Define* $P^{\alpha_i}_{jk} = \delta_{j,\alpha_i(k)}$.
   - *Define* $V^{\alpha_i}(x) = \sum_{j=0}^{l_i-1}(-1)^j c^{l_i-1-j}(x,l_i)s^j(x,l_i)(P^{\alpha_i}R^{\mathcal{L}\bigcap\beta_i})^j$
   - *Set* $V^{\alpha_i}_{jk}(x) = \delta_{jk}$ *for pairs* j,k *not both within* β_i.
   - $\mathcal{T}(x)^{-1}$ *can be evaluated analytically via* $S^{p,1/\gamma(x),1/\gamma(-x)}(\prod_{i=1}^{k}V^{\alpha_i}(x))(P^{\eta_A})^{-1}$.
   - *The condition number* $\kappa_\infty(\mathcal{T}(x))$ *is bounded by* $8n\epsilon^{-n}$.

REFERENCES

[1] U.M. Ascher, R.M.M. Mattheij and R.D. Russel, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, SIAM, 1995.

[2] F.A. Bornemann, *An adaptive multilevel approach to parabolic equations I. General theory and 1d implementation*, Impact of Computing in Science and Engineering, 2, 279-317, 1990.

[3] D. Cai, L. Tao, M. Shelley and D.W. McLaughlin, *An effective kinetic representation of fluctuation-driven neuronal networks with application to simple and complex cells in visual cortex*, Proc. Nat'l Acad. Sci. (USA), 101, 7757-7762, 2004.

[4] J.W. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997.

[5] L.C. Evans, *Partial Differential Equations*, AMS, 1998.

[6] Jie-Cai Han and Bao-Lin Wang, *Electromechanical model of periodic cracks in piezoelectric materials*, Mechanics of Materials, 37, 1180-1197, 2005.

[7] H.B. Keller, *Numerical Methods for Two-Point Boundary Value Problems*, Dover, 1992.

[8] C.G. Lai, A. Callerio, E. Faccioli, V. Morelli and P. Romani, *Prediction of railway-induced ground vibrations in tunnels*, Journal of vibration and acoustics-transactions of the ASME, 127, 503-514, 2005.

[9] J.Y. Lee and L. Greengard, *A fast adaptive numerical method for stiff two-point boundary value problems*, SIAM, J. Sci. Comput., 18, 403-428, 1997.

[10] V.S. Nikishin, *The steady motion of a circular crack along the interface of a layer and a half-space*, J. Appl. Math. Mech., 69, 447-472, 2005.

[11] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover, 1998.

[12] A.V. Rangan and D. Cai, *Maximum-entropy closures for kinetic theories of neuronal network dynamics*, Phys. Rev. Lett., 96, 1-4, 2006.

[13] A.V. Rangan, D. Cai and L. Tao, *Numerical methods for solving moment equations in kinetic theory of neuronal network dynamics*, J. Comput. Phys., 221, 81-798, 2007.

[14] P. Starr and V. Rokhlin, *On the numerical solution of two-point boundary value problems II*, Appl. Math., 79, 271-289, 1988.

[15] L.N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, 1997.