Journal of Scientific Computing, Vol. 24, No. 3, September 2005 (© 2005)
DOI: 10.1007/s10915-004-4794-4

Preconditioning for a Class of Spectral Differentiation Matrices

Weiming Cao,[1] Ronald D. Haynes,[2] and Manfred R. Trummer[3]

Received April 16, 2003; accepted (in revised form) February 19, 2004

We propose an efficient preconditioning technique for the numerical solution of first-order partial differential equations (PDEs). This study has been motivated by the computation of an invariant torus of a system of ordinary differential equations. We find the torus by discretizing a nonlinear first-order PDE with a full two-dimensional Fourier spectral method and by applying Newton's method. This leads to large nonsymmetric linear algebraic systems. The sparsity pattern of these systems makes the use of direct solvers prohibitively expensive. Commonly used iterative methods, e.g., GMRes, BiCGStab and CGNR (Conjugate Gradient applied to the normal equations), are quite slow to converge. Our preconditioner is derived from the solution of a PDE with constant coefficients; it has a fast implementation based on the Fast Fourier Transform (FFT). It effectively increases the clustering of the spectrum, and speeds up convergence significantly. We demonstrate the performance of the preconditioner in a number of linear PDEs and the nonlinear PDE arising from the Van der Pol oscillator.

KEY WORDS: Preconditioners; fast Fourier transform; dynamical systems; invariant torus; spectral collocation; pseudospectral method; iterative methods for nonsymmetric systems

AMS CLASSIFICATIONS: 65P05; 65N30; 65F10; 35K55.

1. INTRODUCTION

The key problem in solving differential equations is often the effective solution of a linear system of equations. Frequently, the matrix of such a

[1] Department of Applied Mathematics, University of Texas at San Antonio, San Antonio, Texas 78249 USA. E-mail: wcao@math.utsa.edu
[2] Department of Mathematics, Simon Fraser University, Burnaby, British Columbia Canada V5A 1S6. E-mail: rhaynes@cs.sfu.ca
[3] Department of Mathematics and Centre for Scientific Computing, Simon Fraser University, Burnaby, British Columbia Canada V5A 1S6. E-mail: trummer@sfu.ca

system is large and sparse, and iterative methods such as conjugate gradient (CG), bi-conjugate gradient (BiCG) and variations, QMR and GMRes have become methods of choice. This is for a variety of reasons, including memory requirements, speed, ability to tackle finer discretizations, and to a good extent, the fact that only an approximate solution is needed – the exact solution of the linear system only provides an approximation to the solution of the original differential equation. Further, in Newton–Krylov approaches to nonlinear problems it is well known that it may not be necessary to compute all Newton steps to high accuracy (see e.g. [23]

In practice, iterative methods need to be preconditioned to make them effective. Instead of solving

$$M\boldsymbol{u}=\boldsymbol{f}$$

one solves

$$P^{-1}M\boldsymbol{u}=P^{-1}\boldsymbol{f},$$

where P is in some sense close to M, and cheaply invertible, i.e., computing $P^{-1}\boldsymbol{z}$, or solving $P\boldsymbol{w}=\boldsymbol{z}$ is cheap. By choosing P close to M, the spectrum of $P^{-1}M$ will have favorable properties for iterative methods (clustered eigenvalues, small spectral radius, etc.).

Taking $P=M$ gives the best spectral properties, leaving us, however, with the original problem when applying the preconditioner. On the other hand, with $P=I$ the preconditioner can be inverted easily, without any improvement to the spectrum. A good preconditioner is obviously somewhere between these two extremes.

In this paper we are interested in finding good preconditioners for solving the periodic first order linear partial differential equation

$$L(u)\equiv a(x,y)\frac{\partial u}{\partial x}+b(x,y)\frac{\partial u}{\partial y}+c(x,y)u=f(x,y). \tag{1}$$

By periodic we mean that the coefficients and the boundary conditions are 2π-periodic in both (x and y) variables. Our preconditioner is related to the discretization of the differential operator in (1) with constant coefficients, i.e.,

$$\bar{L}(u)\equiv\alpha\frac{\partial u}{\partial x}+\beta\frac{\partial u}{\partial y}+\nu u. \tag{2}$$

Applying a full Fourier spectral discretization to (1) leads to a "fairly sparse", highly structured, nonsymmetric linear algebraic system. Its sparsity pattern, however, prohibits a direct solution method. On the other hand, the efficiency of iterative methods relies critically on the distribution of the spectrum and the normality of the eigenvectors of the linear system. Various preconditioning methods have been developed in the past in the context of different applications. For instance, the semi-circulant preconditioner for convection-diffusion equations [15]; the ILU and characteristic Gauss–Seidel preconditioners for the finite volume discretization of hyperbolic problems [18]; the circulant preconditioner for Hermitian systems [3]. These methods are, however, for compact discretization schemes with special matrix structures. They are not applicable in the case of our spectral discretization of (1).

We propose in this paper a preconditioner for the efficient iterative solution of the spectral discretization of (1). The idea is to define the preconditioner as the matrix corresponding to the discretization of (2), i.e., the differential operator with constant coefficients. The preconditioned system can then also be interpreted as an approximation of the linear problem with the operator $\bar{L}^{-1}L$. Hence, the efficiency of the preconditioned iterative methods relies on the spectral properties of $\bar{L}^{-1}L$. When the coefficients in (1) vary only moderately, the spectrum of the preconditioned system will be clustered near the number 1. Roughly speaking, $\bar{L}^{-1}L$ is a differential operator of order 0. Therefore, we may expect more clustering in its spectrum, which is generally good for iterative methods [21]. Another important feature of our preconditioner is that its action can be easily and efficiently accomplished by a fast algorithm using the FFT.

The idea of defining the preconditioner as a discretization of a suitable linear preconditioning operator has been studied previously, and applied widely to linear and nonlinear elliptic problems, see, e.g., [8,9]. Indeed, due to the spectral equivalence of various elliptic operators, the condition numbers of so defined preconditioned systems can be bounded by constants independent of the mesh sizes. For hyperbolic equations, however, there is in general no spectral equivalence relation between the differential operator of the equation and a preconditioning operator with constant coefficients. Therefore, it is in general not possible to prove the mesh size independent boundedness for the spectra of the preconditioned systems. Nevertheless, we find that applying the constant coefficient preconditioner makes the spectra of the preconditioned system more clustered, which is desirable for the iterative methods designed for nonsymmetric systems. Numerical results show that the preconditioner defined by (2) speeds up the convergence of the iterative methods significantly for many of the linear test problems, and, more importantly, for our nonlinear application, namely the Van der Pol oscillator, see Secs. 2.4 and 3.2.

An outline of the paper is as follows. In Sec. 2 we briefly describe the spectral discretization of (1). We introduce the preconditioner and its implementation. We present a number of numerical examples to demonstrate properties of the preconditioned system, and compare the performance of the preconditioner in various cases with different iterative methods, such as BiCGStab(ℓ) [13,20,24], GMRes [19], and CGNR. We find that when the coefficients in (1) vary moderately, preconditioned iterative methods are very fast and reliable. If these coefficients vary strongly, iterative methods in general do not perform very well, but our preconditioner still provides significant gains in speed and accuracy. Difficulties arise when the coefficients a and b in (1) change signs. If only one of them does, our preconditioning continues to be a successful strategy; if, however, both of them change signs, the preconditioner fails. In Sec. 3 we present an application in the context of computing an invariant torus, and introduce the computational methods for the Van der Pol oscillator. This formulation leads to a first order nonlinear hyperbolic differential equation with periodic boundary conditions. We use Newton's method to linearize it, and solve the linearized equation with the methods explored in Sec. 2. The effectiveness of our preconditioner in computing invariant tori for the Van der Pol oscillator is demonstrated for various iterative solvers. Finally, Sec. 4 contains conclusions.

2. DISCRETIZATION AND SOLUTION METHOD

2.1. Fourier Spectral Collocation Method

In this section, we consider the Fourier spectral collocation method for equation (1). We denote by Π_p the set of real-valued trigonometric polynomials of degree $\leqslant p$. Let N be a positive integer, and $x_j = (j/N)2\pi$, $j=0,1,\dots,N-1$, be the set of Fourier collocation points. For any continuously differentiable function $v(x)$, we denote

$$\mathbf{v}=\begin{pmatrix} v(x_0)\\ v(x_1)\\ \vdots\\ v(x_{N-1})\end{pmatrix} \quad\text{and}\quad \frac{d\mathbf{v}}{dx}=\begin{pmatrix} \frac{dv}{dx}(x_0)\\ \frac{dv}{dx}(x_1)\\ \vdots\\ \frac{dv}{dx}(x_{N-1})\end{pmatrix}.$$

The $N\times N$ Fourier spectral differentiation matrix $D=(D_{jk})$ is defined as follows: for even N

$$D_{jk}=\begin{cases} \frac{1}{2}(-1)^{j-k}\cot\frac{(j-k)\pi}{N}, & j\neq k,\\ 0, & j=k\end{cases}$$

and for odd N

$$D_{jk} = \begin{cases} \frac{1}{2}(-1)^{j-k}/\sin\frac{(j-k)\pi}{N}, & j \neq k, \\ 0, & j = k. \end{cases}$$

It is not difficult to verify that for any $v \in \Pi_{[N-1/2]}$, where $[s]$ denotes the integer part of a number s,

$$\frac{d\mathbf{v}}{dx} = D\mathbf{v}.$$

We note that D is a circulant skew-symmetric Toeplitz matrix.

In two-dimensions, let $u(x, y)$ be a trigonometric function of degree $\leqslant [(N-1)/2]$ in both x and y. Let x_j and y_k be the collocation points in the x and y directions, respectively. We arrange the values of the function u at the collocation points in an $N \times N$ array

$$U = \begin{pmatrix} u(x_0, y_0) & u(x_0, y_1) & \cdots & u(x_0, y_{N-1}) \\ u(x_1, y_0) & u(x_1, y_1) & \cdots & u(x_1, y_{N-1}) \\ \vdots & \vdots & \ddots & \vdots \\ u(x_{N-1}, y_0) & \cdots & & u(x_{N-1}, y_{N-1}) \end{pmatrix}. \tag{3}$$

Suppose that the partial derivatives (with respect to x and y) of u at (x_j, y_k) are arranged in the same manner as in (3) in the arrays $\partial U/\partial x$ and $\partial U/\partial y$. Then we have (see [1])

$$\frac{\partial U}{\partial x} = DU \qquad \text{and} \qquad \frac{\partial U}{\partial y} = UD^T = -UD.$$

The Fourier spectral collocation discretization for (1) may now be written as

$$A \cdot (DU) + B \cdot (UD^T) + C \cdot U = F. \tag{4}$$

The matrices A, B, C, and F contain the values of the coefficient functions a, b, c, and f evaluated at (x_j, y_k) and ordered as in (3). The "$\cdot$" represents the entry-wise multiplication or Hadamard product of two matrices.

Equation (4) is a linear equation for the unknown matrix U, which contains the values of the unknown function u at the collocation nodes. We may also write it as a linear system of equations for an unknown vector $\mathbf{u}$. Let $\mathbf{u}$ and $\mathbf{f}$ be the $N^2 \times 1$ vectors constructed from U and F column–wise. Define D_a as the $N^2 \times N^2$ diagonal matrix with the mth diagonal entry equal to A_{jk} for $m = j + (k-1)N$, and define the matrices D_b and D_c accordingly with entries from B and C, respectively. Let

$$M = D_a(D \otimes I_N) + D_b(I_N \otimes D) + D_c I_{N^2},$$

where "$\otimes$" is the tensor product of two matrices, and I_N and I_{N^2} represent the identity matrices of order N and N^2, respectively. Then (4) can be expressed equivalently as the linear system of equations

$$M\mathbf{u} = \mathbf{f}. \tag{5}$$

2.2. Iterative Techniques and Preconditioning

In this section we consider the solution of the linear system (5). The sparsity pattern of the matrix M is illustrated in Fig. 1. It is clear that M is a sparse, structured matrix. The large bandwidth, however, leads to fill–in for standard sparse direct techniques, as shown by the LU factors of M. Typical sparse reordering algorithms have little effect for matrices with this structure [14].

The high cost of direct methods and the ability to provide fast matrix–vector multiplication routines, due to the structure of M, suggest the use of iterative methods. Such an approach often proves necessary for efficient computation with problems, which are memory intensive or have exploitable structure [2,22]. A popular class of iterative methods are Krylov subspace methods. The success of these iterative methods depends on the computation of a good approximation from a suitable Krylov space of moderate dimension. If this is not feasible, we seek to solve a preconditioned system, as mentioned in the introduction, for which iterative methods have improved convergence properties. A good preconditioner should enjoy several properties: memory requirements of the same order as M, cost comparable to the computation of matrix–vector products involving M, and an improved convergence rate which justifies the additional costs.

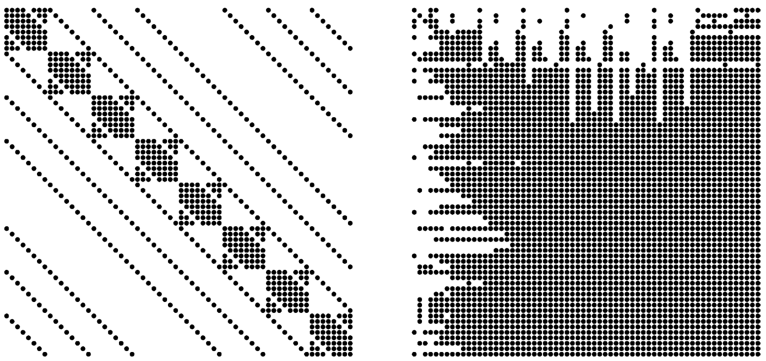


Fig. 1. Structure of the coefficient matrix M (left) and corresponding LU factors (right).

We now propose a preconditioner for the iterative solution of (5). Our basic strategy is to use the constant coefficient problem corresponding to (4) to construct the preconditioner. Let $\bar{a}$ and $\bar{b}$ be the average of the entries of matrices A and B, separately. Let ν be a user-defined parameter. We define a preconditioner, P, for the linear algebraic system (4), by defining its multiplication with any $N \times N$ matrix X as follows

$$PX = \bar{a}DX + \bar{b}XD^T + \nu X. \tag{6}$$

This is indeed the matrix for the spectral collocation discretization of the operator

$$\bar{L}(u) \equiv \bar{a}\frac{\partial u}{\partial x} + \bar{b}\frac{\partial u}{\partial y} + \nu u,$$

i.e., (2) with $\alpha = \bar{a}$, $\beta = \bar{b}$.

The preconditioned matrix $P^{-1}M$ is essentially the discretization of the operator $\bar{L}^{-1}L$, a differential operator of order 0, which in some sense is close to the identity. Therefore, we expect clustering in its spectrum, and, hence also the spectrum of $P^{-1}M$, which is promising for iterative methods. Our preconditioner is intimately connected to the underlying continuous problem, and is not directly derived from the matrix of the system we are trying to solve.

We shall discuss the fast algorithm to implement our preconditioner in Sec. 3. In Sec. 2.4 we demonstrate how an appropriate choice of the constant ν localizes the spectrum of the preconditioned system, which improves the convergence properties of various iterative methods.

2.3. Fast Algorithm to Evaluate the Action of Preconditioner

Preconditioned iterative methods require a routine for evaluating the product of P^{-1} with a (residual) vector. Assuming the residual vector of (5) is expressed in matrix form R, ordered in the same way as U in (3), then calculating $P^{-1}R$ is equivalent to solving the linear equation $PX = R$, or

$$\bar{a}DX + \bar{b}XD^T + \nu X = R \tag{7}$$

for the $N \times N$ matrix X. This is actually a variation of Lyaponov's matrix equation for X [16]. To solve (7), we note that the spectral differentiation matrix D admits the eigen-decomposition

$$D = T\Lambda T^{-1} = T\Lambda T^H,$$

where

$$\Lambda = \begin{cases} \operatorname{diag}(0, i, 2i, \dots, \left(\frac{N}{2}-1\right)i, 0, -\left(\frac{N}{2}-1\right)i, \dots, -i), & \text{for } N \text{ even,} \\ \operatorname{diag}(0, i, 2i, \dots, \left(\frac{N-1}{2}\right)i, -\left(\frac{N-1}{2}\right)i, \dots, -i), & \text{for } N \text{ odd} \end{cases}$$

and

$$T = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & e^{ix_0} & e^{i\cdot 2x_0} & \dots & e^{i\cdot(N-1)x_0} \\ 1 & e^{ix_1} & e^{i\cdot 2x_1} & \dots & e^{i\cdot(N-1)x_1} \\ & \dots & & \dots & \\ 1 & e^{ix_{N-1}} & e^{i\cdot 2x_{N-1}} & \dots & e^{i\cdot(N-1)x_{N-1}} \end{pmatrix}.$$

We observe that the matrix T is unitary, $T^{-1} = T^H$. This suggests the following algorithm to solve Eq. (7):

(i) compute $\hat{R} = T^H R T^H$, and rewrite (7) as

$$\bar{a}\Lambda\hat{X} + \bar{b}\hat{X}\Lambda + \nu\hat{X} = \hat{R},$$

where $\hat{X} = T^H X T^H$;

(ii) solve for $\hat{X} = (\hat{X}_{\ell,m})$ using the formula

$$\hat{X}_{\ell,m} = \frac{\hat{R}_{\ell,m}}{\bar{a}\lambda_\ell + \bar{b}\lambda_m + \nu}, \qquad 1 \leqslant \ell, \quad m \leqslant N.$$

where λ_ℓ, $1 \leqslant \ell \leqslant N$, are the eigenvalues of D;

(iii) calculate $X = T\hat{X}T$.

To analyze the cost of performing the preconditioning, we define W_{Mx} as the number of arithmetic operations required for one coefficient matrix-vector multiplication in the system (5). It mainly involves two products of $N \times N$ matrices, and each product requires $2N^3$ arithmetic operations. Therefore, $W_{Mx} = 4N^3$. Now consider the number of operations needed to carry out the above preconditioning algorithm. Steps (i) and (iii) involve four multiplications of complex $N \times N$ matrices, each requiring $8N^3$ real arithmetic operations. This works out to approximately $32N^3$ arithmetic operations, or about $8W_{Mx}$. The work load for Step (ii) is only $4N^2$ operations, negligible to the work required in steps (i) and (iii). Hence the total work for each each action of the preconditioner is about 8 times the work of a coefficient matrix-vector product. This cost is too high to produce any significant benefits for the preconditioning technique. Indeed, if the preconditioner was implemented as described above, then, for many problems, the preconditioned iterative methods would require no less computational work than those without preconditioning.

Fortunately, we can take advantage of the connection between the matrix T and the discrete Fourier transform. In fact, T is the matrix associated with the one-dimensional inverse discrete Fourier transform. For any function $v \in \text{Span}\{e^{ikx}, -(N/2) - 1 \leqslant k \leqslant (N/2)\}$, set $\mathbf{v} = (v(x_0), v(x_1), \ldots, v(x_{N-1}))^T$, and denote by $\hat{\mathbf{v}}$ the vector of the Fourier coefficients of v. Then

$$\mathbf{v} = T\hat{\mathbf{v}} \qquad \text{and} \qquad \hat{\mathbf{v}} = T^H \mathbf{v}.$$

Therefore, it is possible to compute the product $T^H R T^H$ in step *(i)* efficiently by the Fast Fourier Transform (FFT). Indeed, we may rewrite the product $T^H R T^H$ as

$$T^H R T^H = (T^H (T^H R)^T)^T$$

Thus, $\hat{R} = T^H R T^H$ is actually the two-dimensional discrete Fourier transform of the data R. Similarly, $X = T\hat{X}T$ is the two-dimensional inverse Fourier transform of the data $\hat{X}$. Consequently, the total work $W_{P^{-1}M}$ per action of the preconditioner is equivalent to that of 2 two-dimensional FFT. If N is an integer power of 2, each two-dimensional FFT requires $O(N^2 \ln N)$ operations. Hence, the action of the preconditioner may be accomplished by using the FFT in $O(N^2 \ln N)$ operations. For large values of N this is far less work than the direct matrix multiplication approach.

Naturally, one may also implement the matrix multiplication Mx by using FFTs. In fact,

$$DX = T\Lambda T^H X = T(\Lambda \hat{X})T \qquad \text{and} \qquad XD^T = XT^H \Lambda T = T(\hat{X}\Lambda)T,$$

where $\hat{X} = T^H X T^H$ is the Fourier transform of X. Therefore, evaluating on the left hand side of (4) requires one two-dimensional FFT to compute the Fourier coefficients of real-valued data U, and two two-dimensional inverse FFTs to compute the function values of U_x and U_y from their Fourier coefficients. Thus, if Mx is calculated in this manner, the total work needed is about three two-dimensional FFTs. Consequently, the work needed for one action of the preconditioner is about 2/3 of the work W_{Mx}.

$$W_{P^{-1}R} \approx \tfrac{2}{3} W_{Mx}. \tag{8}$$

We shall use this work load ratio when comparing the overall cost of iterative methods with and without preconditioners in our numerical examples.

2.4. Numerical Examples for Linear Equations

In this section, we demonstrate the performance of the preconditioner defined in (6) for a number of numerical examples. We shall employ Krylov subspace based iterative methods, in particular BiCGStab(ℓ) [20] (a variant of Gutknecht's BiCGStab2 method [13]), GMRes(k), the restarted GMRes method of Saad [19], and CGNR, the classical conjugate gradient method applied to the normal equations of (5). To implement CGNR we supply the formula for M^T as follows

$$\begin{aligned} M^T &= D_a(D^T \otimes I_N) + D_b(I_N \otimes D^T) + D_c I_{N^2} \\ &= -D_a(D \otimes I_N) - D_b(I_N \otimes D) + D_c I_{N^2}. \end{aligned}$$

Thus, if the unknown $\mathbf{v}$ is arranged in matrix form V, $M^T\mathbf{v}$ is equivalent to

$$-A \cdot (DV) - B \cdot (VD^T) + C \cdot V.$$

In the following examples, we shall use the the four iterative methods BiCGStab(2), BiCGStab(8), GMRes(10) and CGNR to solve the linear system (4) for various cases of the linear PDE (1). We report the number of iterations required as well as the accuracy achieved by the various iteration schemes. We do not discuss how well the solution of the discretized problem (5) approximates the exact solution of the continuous problem (1). Indeed, in all cases, the numerical solutions are found to converge to the exact solution of (1) with the rate of convergence $O(N^{-s})$, where s depends on the smoothness of the exact solution. This is typical for spectral approximation. Break-down or divergence of an iterative method is indicated in the tables by "**".

To compare the overall work used by the iterative methods with and without preconditioning, we note that when preconditioning is used, each coefficient matrix-vector product is accompanied by an action of the preconditioner. Thus each step of the preconditioned iteration requires about 5/3 the work for the unpreconditioned iteration, see (8). The various iterative methods require the following number of matrix-vector products per iteration:

Number of matrix-vector multiplications per step			
BiCGStab(2)	BiCGStab(8)	GMres(10)	CGNR
4	16	11	4

Table I. Maximum Number of Iterations Allowed

N	16	32	64	128	256
BiCGStab(2)	128	256	512	512	768
BiCGStab(8)	32	64	128	128	192
GMres(10)	64	128	256	256	384
CGNR	128	256	512	512	768

To take advantage of the FFT, we choose N as integer powers of 2 for all calculations. We terminate the iteration when the 2-norm of the residual vector is reduced by tol $= N * 10^{-9}$, i.e. when

$$||r_n||_2 < tol||r_0||_2.$$

We scale the tolerance with N, simply because the norm of a vector with all entries equal to 1 is N. The criterion is based on achieving a given reduction of the norm of the residual vectors from its initial value. As a consequence, in certain cases, the preconditioned method may not only be much faster, but may also achieve superior accuracy, as the initial residual for the preconditioned system may be smaller than for the original system by several orders of magnitude. Hence, in our results, we list the number of iterations and the error $||u_n - u_{\text{exact}}||_2$ at the last iteration. We list the error in cases of convergence failure when this error is "reasonably small". We also stop the iterative methods when the number of iterations exceeds an upper bound, (see Table I.) We set these limits so that the total work allowed for the various iterative methods is roughly the same. We allow only modest growth of the upper limit with N, because for good preconditioners, the number of iterations required should be independent of N. We want to emphasize that we are solving systems with matrices of size N^2 by N^2. Hence, in our linear examples, the largest systems we solve have 65536 equations and unknowns.

Example 1. Equation (1) with coefficients

$$a = 1, \qquad b = 100, \qquad c = 1.$$

In this case, the matrix M has eigenvalues

$$\lambda_{\ell m}(M) = c + i(a\ell + bm), \qquad -(N/2 - 1) \leqslant \ell, \quad m \leqslant N/2$$

with the corresponding eigenvectors $U_{\ell m} = (e^{i(\ell x_j + m y_k)})_{0\leqslant j,k\leqslant N-1}$. The preconditioner P has the same eigenvectors, but with corresponding eigenvalues

$$\lambda_{\ell m}(P) = \nu + i(a\ell + bm), \qquad -(N/2-1) \leqslant \ell, \quad m \leqslant N/2.$$

Therefore, the preconditioned coefficient matrix $P^{-1}M$ has the same eigenvectors with corresponding eigenvalues

$$\lambda_{\ell m}(P^{-1}M) = \frac{c + i(a\ell + bm)}{\nu + i(a\ell + bm)}, \qquad -(N/2-1) \leqslant \ell, \quad m \leqslant N/2.$$

It is easy to establish that

$$\begin{aligned}&|\lambda(M)| \subset [c, \sqrt{c^2 + (|a|+|b|)^2 N^2/4}];\\ &\mathrm{Real}(\lambda(M)) = c, \quad \text{and} \quad |\mathrm{Imag}(\lambda(M))| \leqslant (|a|+|b|)\tfrac{N}{2}.\end{aligned}$$

Note that for the function $f(s) = (c+is)/(\nu+is)$ of the real variable s,

$$\begin{aligned}\min(1, |\tfrac{c}{\nu}|) \leqslant{}& \quad |f(s)| &&\leqslant \max(1, |\tfrac{c}{\nu}|).\\ \min(1, |\tfrac{c}{\nu}|) \leqslant{}& |\mathrm{Real}(f(s))| &&\leqslant \max(1, |\tfrac{c}{\nu}|)\\ & |\mathrm{Imag}(f(s))| &&\leqslant \tfrac{1}{2}|1 - \tfrac{c}{\nu}|.\end{aligned}$$

Therefore, we have for $\lambda(P^{-1}M)$ that

$$\begin{aligned}&|\lambda(P^{-1}M)| &&\subset [\min(1, |\tfrac{c}{\nu}|), \max(1, |\tfrac{c}{\nu}|)]\\ &|\mathrm{Real}(\lambda(P^{-1}M))| &&\subset [\min(1, |\tfrac{c}{\nu}|), \max(1, |\tfrac{c}{\nu}|)]\\ &|\mathrm{Imag}(\lambda(P^{-1}M))| &&\subset [0, \tfrac{1}{2}|1 - \tfrac{c}{\nu}|].\end{aligned}$$

Hence, the spectrum of $P^{-1}M$ is contained in a box in the right half-plane, and the size of this box is independent of N. In addition, it is easy to see that when ℓ and m are large, $\lambda_{\ell m}(P^{-1}M)$ is clustered around the number 1 (see Fig. 2) for plots of the eigenvalues of $P^{-1}M$ with $N = 16$.

Because of the special structure of the eigenvalues and eigenvectors of M, all iterative methods converge quickly in this case (one iteration for BiCGStab(ℓ), one iteration for GMRes(10), and two iterations for CGNR, for all tested values of N). Preconditioning does not change the number of iterations required, but the solutions are often a little more accurate. The results are insensitive to the parameter ν in the preconditioner.

Example 2. Equation (1) with coefficients

$$a = 1, \qquad b = 10 + \exp(2\sin(2x + y)), \qquad c = 1.$$

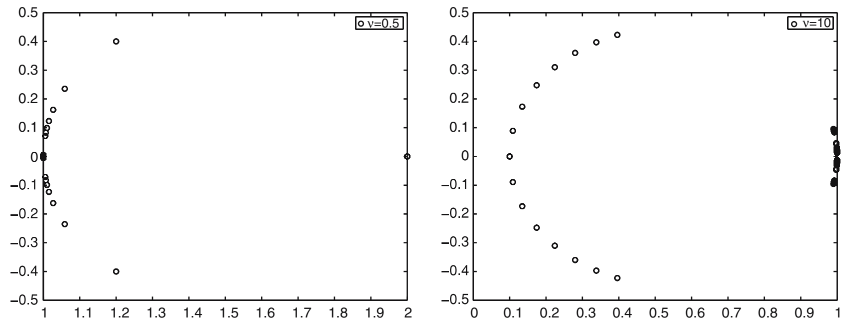


Fig. 2. Example 1. Eigenvalues of $P^{-1}M$ ($N=16$) with $\nu=\frac{1}{2}$ (left) and $\nu=10$ (right).

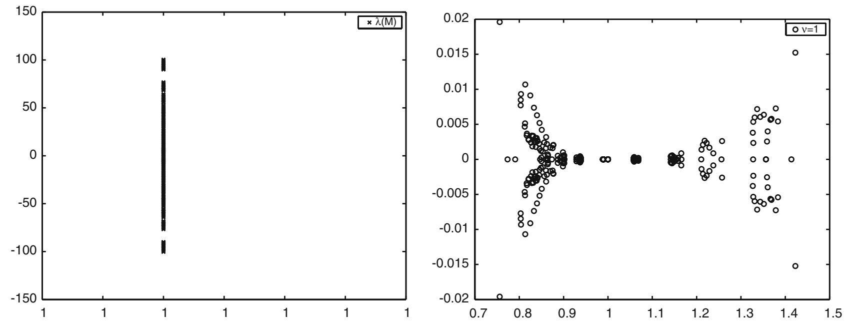


Fig. 3. Example 2. Eigenvalues of M (left) and $P^{-1}M$ with $\nu=1$ (right). $N=16$.

In this example there is no explicit formula for the eigenvalues of M, and its eigenvectors are no longer orthogonal. We plot in Fig. 3 the distribution of the eigenvalues of M (with $N=16$). Note that they are located near the vertical line $Re(\lambda)=1$. BiCGStab(ℓ) converges slowly for both $\ell=2$ and $\ell=8$; GMRes(10) does not converge within the set number of iterations.

On the other hand, with proper choice of ν, the spectrum of the preconditioned system is contained in a small region, clustering around 1 (see Fig. 3). The number of iterations to satisfy the convergence criterion is summarized in Table II for various choices of ν. For the unpreconditioned problem, BiCGStab(ℓ) outperforms CGNR. Preconditioning significantly improves the speed of convergence of GMRes(k) and BiCGStab(ℓ).

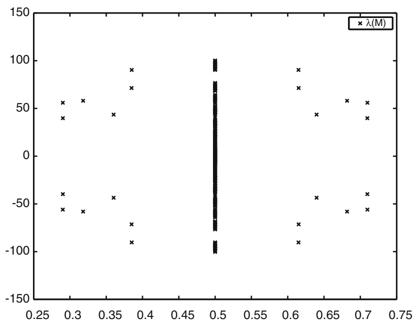
Example 3. Equation (1) with coefficients

$$a=1, \qquad b=10+\exp(2\sin(2x+y)), \qquad c=1-\sin^2 x.$$

Table II. Example 2 Number of Iterations and Errors

N	16	32	64	128	256
BiCGStab(2)					
No P.C.	9	23	57	145	447
	9.6e-007	3.3e-006	7.0e-006	1.5e-005	2.0e-004
$\nu = 0.5$	3	3	3	3	3
	7.3e-009	2.8e-008	6.3e-009	1.4e-006	5.5e-008
$\nu = 1$	3	3	2	2	2
	2.0e-009	2.5e-010	1.1e-006	1.7e-006	2.8e-005
$\nu = 10$	5	5	5	4	5
	4.6e-007	6.4e-007	1.2e-008	4.3e-006	1.6e-006
BiCGStab(8)					
No P.C.	3	6	13	29	73
	5.0e-011	1.6e-006	3.3e-006	1.4e-004	3.4e-004
$\nu = 0.5$	1	1	1	1	1
	2.8e-012	8.1e-012	8.1e-012	1.3e-009	6.0e-009
$\nu = 1$	1	1	1	1	1
	1.4e-013	2.1e-013	3.1e-012	4.0e-010	7.5e-010
$\nu = 10$	2	1	2	1	1
	5.7e-012	4.1e-006	7.8e-011	6.3e-005	2.1e-005
GMRes(10)					
No P.C.	**	**	**	**	**
	2.4e-003	3.4e-002			
$\nu = 0.5$	1	1	1	1	1
	8.1e-009	3.5e-008	1.6e-008	3.0e-008	5.9e-008
$\nu = 1$	1	1	1	1	1
	1.9e-010	2.3e-009	3.2e-009	4.8e-009	9.6e-009
$\nu = 10$	2	2	2	2	2
	1.3e-008	3.4e-008	5.1e-008	7.7e-008	1.5e-007
CGNR					
No P.C.	43	60	115	454	**
	1.4e-008	5.7e-009	2.1e-008	9.2e-008	

Examples 1 and 2, the coefficient c is constant, which results in the eigenvalues of the matrix M being mostly located around a vertical line in the complex plane. In this example, we vary c so that the eigenvalues of M are distributed in a banded region, (see Fig. 4). For the unpreconditioned systems all methods either fail to converge, converge very slowly, or, in the case of CGNR, produce fairly large errors. This is partly due to the initial residual being very large (see Table III).



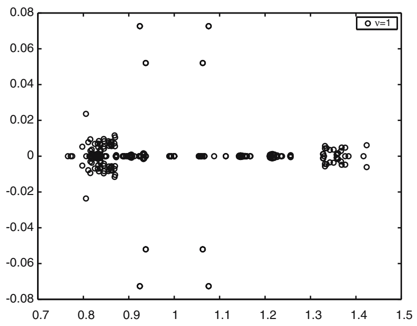


Fig. 4. Example 3. Eigenvalues of M (left) and $P^{-1}M$ with $\nu = 1$ (right). $N = 16$.

Table III. Example 3 Number of Iterations and Errors

N	16	32	64	128	256
BiCGStab(2)					
No P.C.	**	**	**	**	**
$\gamma = 0.5$	3	3	3	3	3
	1.1e-008	1.3e-007	1.5e-007	1.2e-007	2.5e-006
$\gamma = 1$	3	3	3	3	3
	2.1e-008	2.0e-008	2.5e-008	4.8e-008	8.7e-008
$\gamma = 10$	8	8	8	8	11
	4.4e-007	5.9e-006	3.8e-006	1.8e-007	1.2e-004
BiCGStab(8)					
No P.C.	23	**	**	**	**
	9.2e-008				
$\gamma = 0.5$	1	1	1	1	1
	8.6e-012	2.2e-008	4.3e-008	7.1e-008	1.7e-007
$\gamma = 1$	1	1	1	1	1
	3.0e-011	1.2e-008	1.4e-008	2.9e-008	1.5e-007
$\gamma = 10$	2	2	2	2	4
	2.7e-008	1.1e-007	1.0e-006	2.3e-007	4.2e-006
GMRes(10)					
No P.C.	**	**	**	**	**
$\gamma = 0.5$	1	1	1	1	1
	1.0e-007	1.7e-007	1.5e-007	3.0e-007	6.1e-007
$\gamma = 1$	1	1	1	1	1
	2.9e-008	5.5e-008	6.0e-008	1.2e-007	2.3e-007
$\gamma = 10$	5	5	5	6	13
	2.9e-007	6.7e-007	2.0e-007	4.7e-005	3.8e-004
CGNR					
No P.C.	114	163	365	301	**
	1.7e-004	4.1e-004	8.3e-004	4.7e-003	9.5e-003

When c is not constant, we choose the parameter ν in the preconditioner as

$$\nu = \gamma \bar{c}$$

where γ is a parameter, and $\bar{c}$ is the average of the matrix C. This choice of ν works only when $\bar{c} \neq 0$, otherwise the preconditioner P is singular. Clearly, $\gamma = 1$ would appear to be a good choice, so that ν coincides with c when the latter is constant. Our experiments show, however, that other values of γ are often preferable. We test various values of γ in this example. With the preconditioned iterative methods convergence can be achieved easily. By contrast, for the unpreconditioned system all iterative methods except CGNR fail.

Example 4. Equation (1) with coefficients

$$a = 1, \qquad b = 1 + \exp(2\sin(2x + y)), \qquad c = 1.$$

This case is identical to Example 2, except that the relative variation of the coefficient b is now much larger. The performance of the iterative methods is, however, quite different for these two examples. Indeed, Example 4 is a much tougher problem, although preconditioning is still very effective. Even when the convergence criterion is not met, the preconditioned methods produce at least acceptable solutions (see Table IV). Example 4 points to a difficulty with prescribed stopping criteria. Sometimes the residual will drop to a certain level, and pretty much stay at this plateau for a long time. Such behavior has been observed by many authors for various problems [10], and attempts at understanding it involve the field of values or the polynomial hull of the matrix (see [11]) (see Fig. 5).

Example 5. Equation (1) with coefficients

$$a = \cos(3x + 4y), \qquad b = 10 + \exp(2\sin(2x + y)), \qquad c = 10(1 + \sin(x + y)).$$

Here we test the case where the coefficients of (1) change signs. It is well known that special care, such as upwinding, must be taken when discretizing this type of problem. It is not appropriate to define the preconditioner by using the average of the coefficients a, b and c of the original differential equation, as this may lead to degenerated $\bar{L}$. Instead, we define $\bar{a}, \bar{b}$ and $\bar{c}$ in (2) as the average of the absolute values of a, b and c, respectively. We report the number of iterations and the solution accuracy in

Table IV. Example 4 Number of Iterations and Errors

N	16	32	64	128	256
BiCGStab(2)					
No P.C.	18	57	118	392	**
	4.1e-007	3.5e-006	3.3e-007	2.1e-005	4.3e+001
$\nu = 0.5$	11	45	16	325	**
	3.1e-007	4.6e-006	1.9e-006	8.0e-006	3.3e-004
$\nu = 1$	7	33	11	56	**
	4.5e-007	4.7e-006	1.9e-006	1.3e-005	8.3e-004
$\nu = 10$	10	16	19	**	**
	3.3e-008	3.5e-006	2.4e-006	3.1e-003	
BiCGStab(8)					
No P.C.	4	13	21	69	185
	1.4e-008	2.2e-006	3.6e-006	3.3e-005	7.7e-005
$\nu = 0.5$	3	14	3	31	**
	1.2e-008	1.7e-006	1.9e-006	1.1e-005	1.1e-003
$\nu = 1$	3	13	4	15	**
	1.3e-009	7.4e-006	6.9e-007	5.6e-006	3.0e-004
$\nu = 10$	3	9	5	**	**
	6.0e-010	4.5e-007	4.1e-006	2.5e-004	5.3e-001
GMRes(10)					
No P.C.	56	**	**	**	**
	8.4e-007	2.9e-004	2.5e-002		
$\nu = 0.5$	5	128	4	**	**
	4.8e-007	8.1e-005	3.2e-006	4.9e-005	1.7e-003
$\nu = 1$	2	128	4	**	**
	7.4e-007	7.8e-005	3.3e-006	2.6e-005	8.6e-004
$\nu = 10$	6	38	**	**	**
	3.4e-007	6.2e-006	4.7e-004	1.1e-001	
CGNR					
No P.C.	63	118	178	370	**
	3.7e-006	3.0e-005	8.4e-005	6.6e-004	7.3e-001

Table V, and the spectrum of A and $P^{-1}A$ in Fig. 6. Our preconditioner improves the convergence of the iterative methods significantly, even though one of the coefficients in (1) changes signs.

Example 6. Equation (1) with coefficients

$$a = \cos(x + y), \qquad b = \sin(x - y), \qquad c = 10.$$

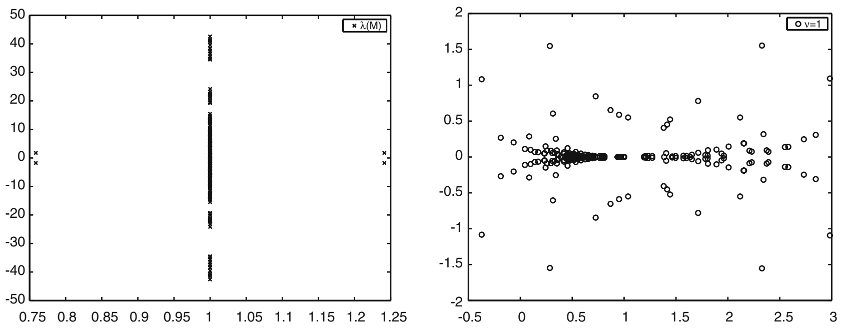


Fig. 5. Example 4. Eigenvalues of M (left) and $P^{-1}M$ with $\nu = 1$ (right). $N = 16$.

In this example both a and b in (1) change signs. By contrast, in Example 5 the coefficient b is bounded below by a positive number. Therefore dividing the equation by b leads to a wave equation with finite wave speed a/b. When both a and b change signs as in the present example, the wave speed would be infinite. In the current example, even though the spectrum of $P^{-1}A$ appears compressed (see Fig. 7), the preconditioning does not improve the convergence of the iterative methods, see Table VI. This example illustrates the limitations of the preconditioner (2).

In summary, the preconditioner defined by discretizing the differential operator with constant coefficients (2) significantly improves the speed of convergence of iterative methods, provided the coefficients vary moderately and are bounded away from zero. When both coefficients a and b change signs, the preconditioner does not appear to be useful.

It is worth mentioning that our results in Tables I–VI appear to indicate the accuracy of the numerical solution depends on the preconditioner. In fact, this is **not** the case. The reason for the varying accuracy is that the residual checked by the iterative methods for convergence is that of the preconditioned system, which depends on the choice of the parameter ν, and that we terminate the iterative methods when the residual reaches the tolerance ($= 10^{-9}N$ in all our tests). We have tested the same iterative methods with or without preconditioning using stricter convergence criteria, and the accuracy of the various approximations is comparable, depending only on the discretization parameter N. The preconditioning by no means compromises the accuracy of the numerical solution when convergence is reached.

Table V. Example 5 Number of Iterations and Errors

N	16	32	64	128	256
BiCGStab(2)					
No P.C.	44	114	251	**	**
	2.0e-007	5.4e-007	3.1e-006	2.1e-004	6.7e-003
$\nu = 0.5$	8	9	8	50	**
	3.1e-008	4.1e-007	2.8e-006	1.7e-005	1.7e-003
$\nu = 1$	6	6	6	6	709
	9.6e-009	8.7e-007	2.5e-007	9.6e-006	3.0e-005
$\nu = 10$	22	25	25	25	71
	1.2e-007	6.5e-007	2.4e-006	1.0e-005	4.0e-005
BiCGStab(8)					
No P.C.	10	26	56	**	**
	1.9e-007	4.7e-007	1.1e-006	1.9e-003	1.6e-002
$\nu = 0.5$	2	2	3	10	**
	2.6e-008	8.2e-007	2.5e-007	1.5e-005	3.1e-004
$\nu = 1$	2	2	2	2	12
	1.7e-011	2.6e-009	8.1e-008	5.0e-006	1.7e-005
$\nu = 10$	6	6	7	16	39
	1.1e-008	1.0e-006	2.1e-007	4.8e-006	2.5e-005
GMRes(10)					
No P.C.	21	48	99	**	**
	1.1e-007	1.2e-006	3.7e-006	3.6e-005	1.5e-003
$\nu = 0.5$	3	4	3	**	**
	4.8e-008	3.6e-008	2.1e-006	4.1e-005	4.2e-003
$\nu = 1$	2	3	2	2	**
	1.8e-007	8.4e-009	9.2e-007	7.8e-006	7.3e-005
$\nu = 10$	10	10	10	10	**
	4.9e-008	1.3e-006	2.1e-006	1.0e-005	1.5e-004
CGNR					
No P.C.	97	196	386	**	**
	1.5e-006	5.7e-006	2.8e-005	8.5e-002	2.6e+00

3. COMPUTATION OF INVARIANT TORI

3.1. Formulation and Linearization

To understand the dynamics of a differential equation, one is often interested in so-called invariant manifolds. A solution trajectory starting on this manifold will always remain on the manifold. Examples of such manifolds are stationary solutions (a fixed point), or invariant circles (a periodic solution). Here we attempt to compute an invariant torus, still one of the simpler types of such manifolds. Indeed, often we have a

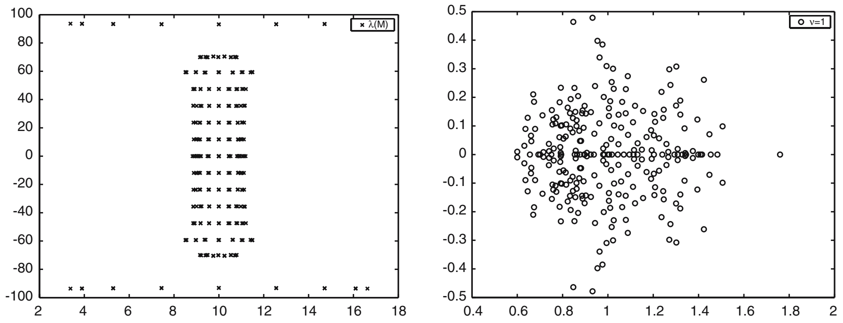


Fig. 6. Example 5. Eigenvalues of M (left) and $P^{-1}M$ with $\nu = 1$ (right). $N = 16$.

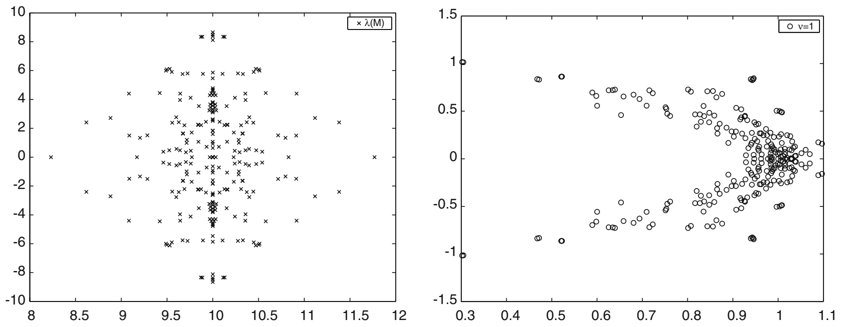


Fig. 7. Example 6. Eigenvalues of M (left) and $P^{-1}M$ with $\nu = 1$ (right). $N = 16$.

parameter in our differential equation, and as this parameter increases, one may observe a bifurcation sequence from stationary solution to periodic solution to invariant torus.

Our methods are based on solving an associated partial differential equation (PDE), an approach first suggested for computation by Dieci *et al.* [7]. We consider the autonomous first-order system of ODEs

$$\frac{dw}{dt} = F(w), \qquad w \in W \subset \mathbb{R}^n, \tag{9}$$

where F is a smooth mapping from W to $\mathbb{R}^n$. We assume that a smooth invariant manifold $\Omega \in W$ exists. As in [7] we assume (9) is of the form

$$\theta_t = f(\theta, r), \qquad r_t = g(\theta, r) \tag{10}$$

with $\theta \in \mathcal{U} \subset \mathbb{R}^p$, $r \in \mathcal{V} \subset \mathbb{R}^q$, and that the manifold Ω can be written as $\{(\theta, \Lambda(\theta)) : \theta \in \mathcal{U}\}$, where $\Lambda : \mathcal{U} \to \mathcal{V}$, i.e., Ω can be parameterized over

Table VI. Example 6 Number of Iterations and Errors

N	16	32	64	128	256
BiCGStab(2)					
No P.C.	5	8	12	21	46
	5.0e-008	2.5e-007	3.7e-007	3.7e-007	1.1e-005
$\nu=0.5$	14	155	**	**	**
	1.0e-007	1.8e-006	1.0e-002	1.3e-001	1.1e+000
$\nu=1$	10	38	**	**	**
	2.4e-009	3.7e-007	1.0e-002	1.2e-002	9.5e-002
$\nu=10$	5	11	17	62	**
	7.0e-008	6.9e-008	1.6e-006	7.7e-006	1.1e-003
BiCGStab(8)					
No P.C.	2	2	3	5	8
	1.5e-013	3.1e-007	9.4e-007	3.4e-006	3.4e-005
$\nu=0.5$	4	18	**	**	**
	8.2e-010	1.5e-006	3.3e-002	2.1e-002	3.2e-001
$\nu=1$	2	8	**	**	**
	1.1e-007	7.8e-007	6.6e-006	2.3e-002	6.6e-002
$\nu=10$	2	3	4	13	125
	1.8e-012	3.0e-007	8.2e-007	5.6e-006	2.5e-005
GMRes(10)					
No P.C.	2	3	4	7	13
	1.2e-008	2.3e-007	7.5e-007	4.3e-006	2.4e-005
$\nu=0.5$	6	59	**	**	**
	8.8e-008	1.4e-006	1.2e-002	4.7e-002	3.2e-001
$\nu=1$	4	17	**	**	**
	7.1e-009	6.4e-007	8.5e-004	4.7e-003	7.1e-002
$\nu=10$	2	4	6	20	**
	3.7e-008	8.7e-008	1.2e-006	6.9e-006	3.7e-005
CGNR					
No P.C.	11	13	16	35	58
	2.8e-007	8.0e-007	2.0e-006	2.6e-006	1.9e-005

$\mathcal{U}$. This is a severe restriction, even though the implicit function theorem states that such a splitting is always possible **locally**.

To compute Λ, i.e. Ω, one has to solve the system of nonlinear PDEs

$$\mathbf{J}_\Lambda(\theta) f(\theta, \Lambda(\theta)) = g(\theta, \Lambda(\theta)), \qquad \theta \in \mathcal{U}, \tag{11}$$

where $\mathbf{J}_\Lambda$ denotes the Jacobian matrix of Λ, with appropriate boundary conditions (see [7]). This procedure can be seen as a natural extension of phase space analysis for systems of ODEs in $\mathbb{R}^2$.

Since we assume that $\mathcal{U}$ is a torus, a manifold without boundary, the boundary conditions for (11) are periodic in each component of the unknown function r.

We denote by

$$T^p := \left\{ \theta = (\theta_1, ..., \theta_p) : \theta_j \in \mathbb{R} \mod 2\pi \right\} \tag{12}$$

the p-dimensional torus. We rewrite the PDE (11) for the case, where $\mathcal{U}$ can be identified with T^2 and where $\mathcal{V} = \mathbb{R}$, i.e., the function r is a scalar. Equation (11) simply becomes

$$f_1(\theta_1, \theta_2, r)\frac{\partial r}{\partial \theta_1} + f_2(\theta_1, \theta_2, r)\frac{\partial r}{\partial \theta_2} = g(\theta_1, \theta_2, r), \text{ on } [0, 2\pi]^2, \tag{13}$$

$$r(\theta_1, 0) = r(\theta_1, 2\pi), \qquad r(0, \theta_2) = r(2\pi, \theta_2). \tag{14}$$

We adopt the standard Newton iteration to solve this nonlinear first order PDE. Let $r^{(0)}$ be an initial guess of the unknown function r. Then the Newton update $r^{(n)} = r^{(n-1)} + u^{(n)}$ is determined by a PDE of the form (1) as follows

$$a^{(n)}(\theta_1, \theta_2)\frac{\partial u^{(n)}}{\partial \theta_1} + b^{(n)}(\theta_1, \theta_2)\frac{\partial u^{(n)}}{\partial \theta_2} + c^{(n)}(\theta_1, \theta_2)u^{(n)} = g^{(n)}(\theta_1, \theta_2), \tag{15}$$

where

$$\begin{aligned}
&a^{(n)} = f_1(r^{(n-1)}), \quad b^{(n)} = f_2(r^{(n-1)}),\\
&c^{(n)} = \frac{\partial f_1}{\partial r}(r^{(n-1)})\frac{\partial r^{(n-1)}}{\partial \theta_1} + \frac{\partial f_2}{\partial r}(r^{(n-1)})\frac{\partial r^{(n-1)}}{\partial \theta_2} - \frac{\partial g}{\partial r}(r^{(n-1)}),\\
&g^{(n)} = g(r^{(n-1)}) - f_1(r^{(n-1)})\frac{\partial r^{(n-1)}}{\partial \theta_1} - f_2(r^{(n-1)})\frac{\partial r^{(n-1)}}{\partial \theta_2}.
\end{aligned}$$

In the remainder of this paper we will restrict ourselves to the computation of the invariant 2–torus for the forced Van der Pol oscillator [4–6, 17]

$$\ddot{x} - \lambda(1 - x^2)\dot{x} + x = \beta \cos(\omega t). \tag{16}$$

Changing to polar co-ordinates, Eq. (16) may be rewritten as an autonomous first–order system of ordinary differential equations

$$\begin{aligned}
\dot{\theta}_1 &= \omega && =: f_1(\theta_1, \theta_2, r),\\
\dot{\theta}_2 &= -1 + \frac{1}{r}\left(\lambda p(r\cos\theta_2)\sin\theta_2 + \beta\cos\theta_2\cos\theta_1\right) && =: f_2(\theta_1, \theta_2, r),\\
\dot{r} &= -\lambda p(r\cos\theta_2)\cos\theta_2 + \beta\sin\theta_2\cos\theta_1, && =: g(\theta_1, \theta_2, r),
\end{aligned} \tag{17}$$

where

$$p(x) = \frac{x^3}{3} - x.$$

For appropriate constants λ, β, and ω, the existence of an invariant 2–torus for (17) can be shown [12]. In our computations, in each Newton step we effectively have to solve (15) for the specific example (17).

3.2. Numerical Results for the Nonlinear Problem

In this section, we present numerical results from the computation of the invariant torus for the Van der Pol oscillator. The bifurcation parameters are set as $\omega = \sqrt{0.84}$, $\beta = 0.32$, and $\lambda = 0.4$, which yields a smooth invariant manifold. For the Newton iteration, we choose $r^{(0)} = 2$ as the initial guess, and we stop the iteration, when the 2-norm of the difference between two consecutive Newton iterates falls below $\mathrm{tol}_{nt} = N * 10^{-8}$. To solve the linear system in each Newton step, we use our iterative methods with and without preconditioner. Figure 8 shows the results from a typical Newton step (namely step 4), indicating the relative performance of the iterative methods we have tested, and giving an idea of the effect of the preconditioner. GMRes(k) gives a smoother convergence, but the BiCGStab(ℓ) methods converge faster. The convergence criterion for the iterative methods is for the 2-norm of the residual to drop by a factor of $N * 10^{-8}$ from the initial residual, or to drop below an absolute tolerance of $N * 10^{-13}$. We would like to point out, however, that this criterion is set for the purpose of studying the performance of the iterative solvers. Actually, one may set a weaker criterion, taking advantage of the fact that in the intermediate Newton steps the linear systems need not be solved to high accuracy, and hence save computational work (see e.g. [23]).

As demonstrated in Sec.2.4, the performance of iterative methods depends on the variation of the coefficients in (1). While in (15) $a^{(n)} = \omega$ for all n, we plot in Fig. 9 the graph of the coefficients $b^{(n)}$ and $c^{(n)}$ for $n = 7$, when the Newton iteration reaches convergence. This gives a rough idea what these coefficients look like. We also display in Fig. 10 the distribution of the eigenvalues of M corresponding to the last step of the Newton iteration. It is easy to see that the eigenvalues of M are evenly distributed in a narrow, long band parallel to the imaginary axis. Therefore, it is not surprising that even for small values of N, the unpreconditioned iterative methods converge very slowly.

Next we examine the effectiveness of the preconditioner defined in (6), whose performance depends on the choice of the parameter ν. We again

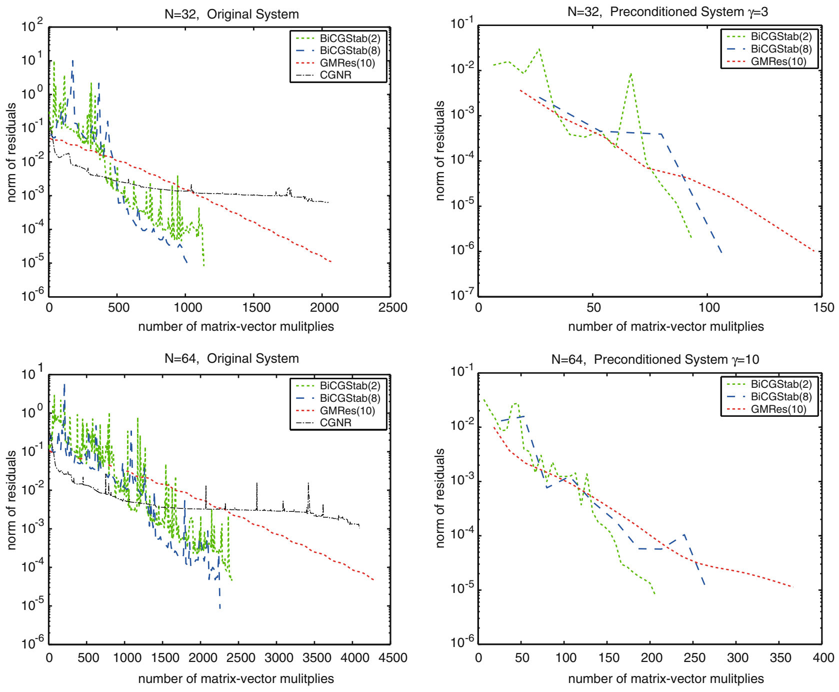


Fig. 8. Convergence history of the iterative methods BiCGStab(2), BiCGStab(8), GMRes(10), and CGNR for $N = 32$ and $N = 64$. Original system (left), preconditioned (right). The residuals are plotted against the workload, i.e., matrix-vector products.

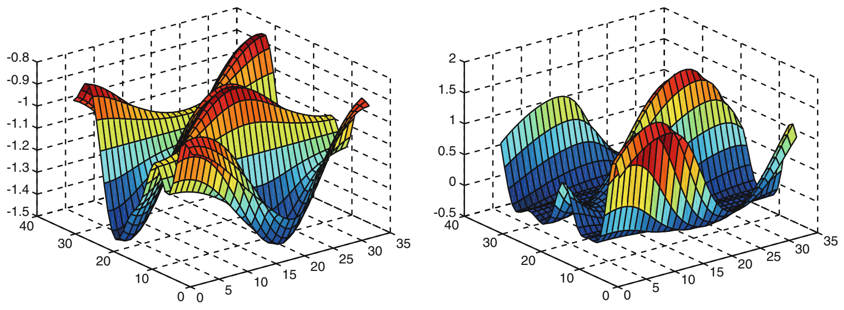


Fig. 9. Coefficients $b^{(n)}$ and $c^{(n)}$ in Eq.(15), $n = 7$ at which the Newton iteration reaches convergence.

set the parameter $\nu = \gamma\bar{c}$, where $\bar{c}$ is the average of the entries of matrix C in (4). In Fig. 11 we plot the eigenvalues of the coefficient matrix $P^{-1}M$ for the preconditioned system with $N = 32$.

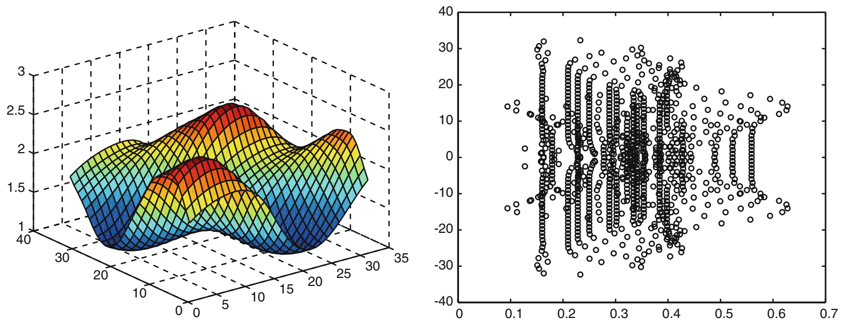


Fig. 10. Solution r of Eq.(11) (left), and the eigenvalues of the linearized equation (15) for $N = 32$ (right).

Comparing the plots in Figs. 10 and 11, it is clear that the spectrum of $P^{-1}M$ is compressed in the imaginary direction. As γ is increased from 0.1 to 10, the spectrum changes from being distributed across the imaginary axis to being clustered away from the imaginary axis, a desirable configuration for fast convergence of iterative methods. These patterns suggest an efficient preconditioner with a parameter $\gamma \in (1, 5)$. To verify this, we plot the convergence history of the preconditioned BiCGStab(2) and GMRes(10) with the above parameters in Fig. 12. The results confirm that the iterative methods converge faster for preconditioned systems whose coefficient matrix has more compact eigenvalue distributions. Furthermore, the convergence rate can be altered by tuning the parameter γ. It appears that $\gamma = 3$ is a good choice for the parameter in this case. The optimal value of γ, however, depends on the size N of system (4) and, to a lesser extent, on the particular Newton step from which it is derived.

In general, choosing larger values for γ pulls the eigenvalues of $P^{-1}M$ further from the origin, while for larger N, the eigenvalues of M get closer to the imaginary axis. To avoid the situation that $P^{-1}M$ has eigenvalues across the imaginary axis, it is judicious to choose a larger γ when N increases. For example, in our computation with $N = 64$, the parameter $\gamma = 10$ gives faster convergence than $\gamma = 1$ does. Figure 8 illustrates the convergence histories for the preconditioned methods.

Finally, we list in Table VII the number of iteration used by various iterative methods for each step of the Newton iteration, for $N = 32$ and 64. When moving to $N = 128$, BiCGStab(8) becomes vastly superior to the other methods (see also [23]). Preconditioning provides substantial speed-up for BiCGStab(8), more pronounced in the earlier Newton iterations than in the later. The overall speed-up is about a factor of 4.

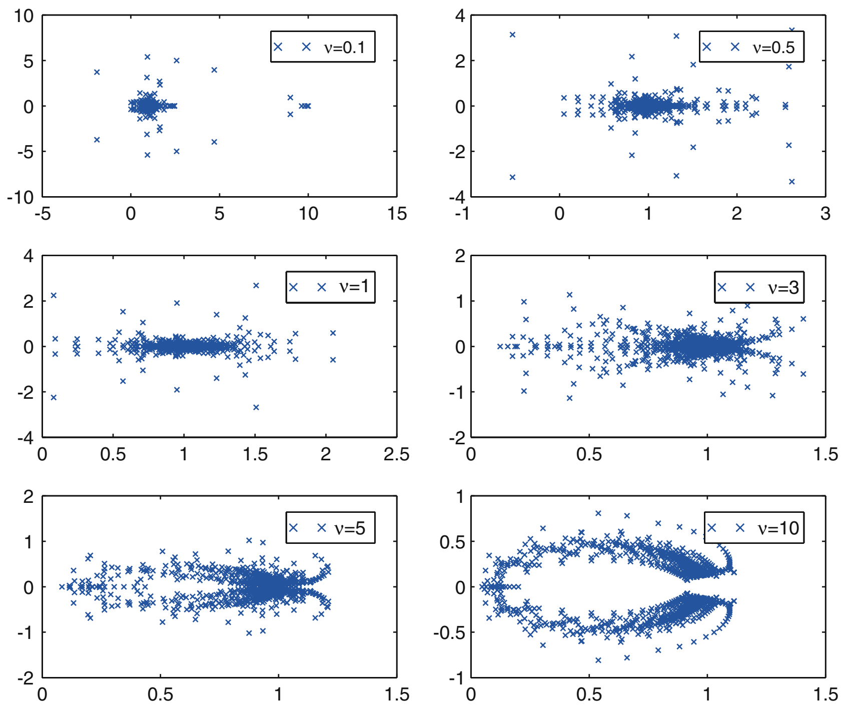


Fig. 11. Eigenvalues of the coefficient matrices $P^{-1}M$ of the preconditioned system with various parameters $\nu = \gamma\bar{c}$ $(N = 32)$.

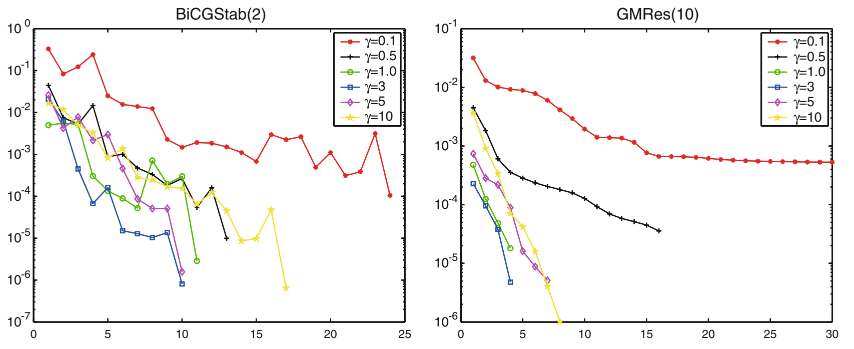


Fig. 12. Convergence history of preconditioned BiCGStab(2) (left) and GMRes(10) (right) methods using various parameters $\nu = \gamma\bar{c}$ (N=32).

Table VII. Number of Iterations in Solving (15) with $\beta = 0.32$ without and with Preconditioning ($\nu = 3\bar{c}$ for $N = 32$ and $\nu = 10\bar{c}$ for $N = 64$) in each Newton Iteration. Required Matrix-Vector Products (MVs) are also Listed

	BiCGStab(2)			BiCGStab(8)			GMRes(10)			
	No P.C.	P.C.	Speed-up	No P.C.	P.C.	Speed-up	No P.C.	P.C.	Speed-up	CG-N
$N = 32$										
Step 1	99	4	14.9	22	3	4.4	47	1	28.2	110
Step 2	124	5	14.9	32	3	6.4	95	2	28.5	241
Step 3	220	5	26.4	49	3	9.8	167	2	50.1	512**
Step 4	293	8	22.0	56	3	11.2	188	4	28.2	512**
Step 5	333	11	18.2	64	3	12.8	183	5	22.0	512**
Step 6	503	16	18.9	98	4	14.7	202	10	12.1	512**
Step 7	36	3	7.2	11	3	2.2	19	1	11.4	512**
Total	1608	52	18.6	332	22	9.1	901	25	21.6	2911
MVs	6432	347	18.6	5312	587	9.1	9911	458	21.6	11644
$N = 64$										
Step 1	231	7	19.8	44	3	8.8	88	3	17.6	169
Step 2	260	12	13.0	71	3	14.2	197	4	29.6	438
Step 3	524	18	17.5	114	8	8.6	356	11	19.4	1024**
Step 4	601	30	12.0	127	9	8.5	392	20	11.8	1024**
Step 5	548	37	8.9	151	12	7.6	380	29	7.9	1024**
Step 6	816	78	6.3	186	17	6.6	440	62	4.3	1024**
Step 7	119	13	5.5	10	5	1.2	51	10	3.1	1024**
Total	3099	195	9.5	703	57	7.4	1904	139	8.2	5727
MVs	12396	1300	9.5	11248	1520	7.4	20944	2548	8.2	22908

4. CONCLUSIONS

We present an effective preconditioner for the iterative solution of the linear algebraic system arising from Fourier spectral discretization of a class of first order PDEs. We test the preconditioner with the iterative solvers BiCGStab(ℓ) and GMRes(k). In the case of constant coefficient c in (1), the unpreconditioned methods and CGNR perform well; CGNR converges linearly with respect to N. When the coefficient c varies, however, the preconditioned BiCGStab(ℓ) and GMRes(k) methods are significantly faster than their unpreconditioned counterparts, and also much faster than the unpreconditioned CGNR method.

We apply the proposed preconditioner to the computation of an invariant torus. The dynamical system is reduced to a nonlinear first-order PDE after parametrization. This PDE with periodic boundary conditions is solved by a Newton iteration; in each Newton step a linear PDE of the form (1) is discretized with a Fourier spectral collocation method. In this calculation a large structured system of linear algebraic equations must be solved. Our proposed preconditioner significantly improves the efficiency of iterative methods for this linear system of equations. This allows a more detailed study of the evolution of the torus as parameters in the dynamical system change and approach critical values where the torus breaks down.

ACKNOWLEDGEMENTS

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant OGP0036901, an NSERC Postgraduate Fellowship, and by National Science Foundation USA (NSF) grant DMS-0209313.

REFERENCES

1. Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. (1988). *Spectral Methods in Fluid Dynamics*, Series of Computational Physics, Springer–Verlag, Heidelberg, Berlin, New York.
2. Cao, W., Haynes, R. D., and Trummer, M. R. (2000). *Preconditioning spectral methods for first-order equations*, Copper Mountain Conference on Iterative Methods.
3. Chan, R. H., Yip, A. M., and Ng, M. K. (2000). The best circulant preconditioners for Hermitian Toeplitz systems. *SIAM J. Numer. Anal.*, **38**, 876–896.
4. Dieci, L. and Bader, G. (1994). Solution of the systems associated to invariant tori approximation. II: Multigrid methods. *SIAM J. Sci. Comput.* **15**, 1375–1400.
5. Dieci, L., and Bader, G. (1995). Block iterations and compactification for periodic block dominant systems associated to invariant tori approximation. *Appl. Numer. Math.* **17**, 251–274.

6. Dieci, L., and Lorenz, J. (1992). Block M-Matrices and computation of invariant tori, *SIAM J. Sci. Stat. Comput*. **13**, 885–903.
7. Dieci, L., Lorenz, J., and Russell, R. D. (1991). Numerical calculation of invariant tori. *SIAM J. Sci. Stat. Comput*. **12**, 607–647.
8. Faber, V. Manteuffel, T. and Parter, S. V. (1990). On the theory of equivalent operators and application to the numerical solution of uniformly elliptic partial differential equations, *Adv. in Appl. Math*. **11**, 109–163.
9. Faragó, I., and Karátson , J. (2002). *Numerical Solution of Nonlinear Elliptic Problems via Preconditioning Operators: Theory and Applications*, Nova Science Publishers, Inc., New York.
10. Greenbaum, A., Ptak, V., and Strakos, Z. (1996). Any nonincreasing convergence curve is possible for GMRES. *SIAM J. Matrix Anal. Appl*. **17**, 465–469.
11. Greenbaum, A. *Personal Communication*.
12. Guckenheimer, J., and Holmes, P. (1983). Nonlinear oscillations, dynamical systems, and bifurcations in vector fields, *Applied Mathematics Science* 42, Springer-Verlag, New York.
13. Gutknecht, M. H. (1993). Variants of BiCGStab for matrices with complex spectrum, *SIAM J. Sci. Stat. Comput*. **14**, 1020–1033.
14. Haynes, R. D. (1998). *Invariant Manifolds of Dynamical Systems: Theory and Computation*, M.Sc. thesis, Simon Fraser University, Canada.
15. Hemmingsson, L. (1998). A semi-circulant preconditioner for the convection-diffusion equation. *Numer. Math*. **81**, 211–248.
16. Horn, R. A., and Johnson, C. R. (1988). *Topics in Matrix Analysis*, Cambridge University Press, Cambridge.
17. Huang, M., Küpper, T., and Masbaum, N. (1997). *Computation of invariant tori by the Fourier method, SIAM J. Sci. Comput*. **18**, 918–942.
18. Meister, A., and Vömel, C. (2001). Efficient preconditioning of linear systems arising from the discretization of hyperbolic conservation laws. *Advances in Comput. Math*. **14**, 49–73.
19. Saad, Y. *Iterative Methods for Sparse Linear Systems*, PWS, 1996 (out of print); http://www-users.cs.umn.edu/ saad/books.html.
20. Sleijpen, G. L. G., and Fokkema, D. R. (1993). BiCGStab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum, Electronic Trans. on Num. Anal. **1**, 11–32.
21. Trefethen, L. N., *Spectra and Pseudospectra: The Behavior of Non-normal Matrices and Operators*. Book in preparation.
22. Trummer, M. R. (1994). *Non-symmetric Systems Arising in the Computation of Invariant Tori*, Copper Mountain Conference on Iterative Methods.
23. Trummer, M. R. (2000). Spectral methods in computing invariant tori. *Appl. Numer. Math*. **34**, 275–292.
24. van der Vorst, H. A. (1992). Bi-CGSTAB: A fast and smoothly convergent variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput*. **13**, 631–644.