Journal of Computational Physics 336 (2017) 627–643

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Continuation of probability density functions using a generalized Lyapunov approach

S. Baars [a,*], J.P. Viebahn [b], T.E. Mulder [c], C. Kuehn [d], F.W. Wubs [a], H.A. Dijkstra [c,e]

[a] Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, P.O. Box 407, 9700 AK Groningen, The Netherlands
[b] Centrum Wiskunde & Informatica (CWI), P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands
[c] Institute for Marine and Atmospheric research Utrecht, Department of Physics and Astronomy, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
[d] Technical University of Munich, Faculty of Mathematics, Boltzmannstr. 3, 85748 Garching bei München, Germany
[e] School of Chemical and Biomolecular Engineering, Cornell University, Ithaca, NY, USA

ARTICLE INFO

Article history:
Received 21 March 2016
Received in revised form 6 February 2017
Accepted 7 February 2017
Available online 14 February 2017

Keywords:
Continuation of fixed points
Stochastic dynamical systems
Lyapunov equation
Probability density function

ABSTRACT

Techniques from numerical bifurcation theory are very useful to study transitions between steady fluid flow patterns and the instabilities involved. Here, we provide computational methodology to use parameter continuation in determining probability density functions of systems of stochastic partial differential equations near fixed points, under a small noise approximation. Key innovation is the efficient solution of a generalized Lyapunov equation using an iterative method involving low-rank approximations. We apply and illustrate the capabilities of the method using a problem in physical oceanography, i.e. the occurrence of multiple steady states of the Atlantic Ocean circulation.

1. Introduction

Dynamical systems analysis of fluid flow phenomena has become a mature research direction [1]. The approach is rather complementary to direct numerical simulation of the governing equations in that the focus is on the direct computation of asymptotic sets in phase space (attracting forward or backward in time). The simplest of these sets are stable and unstable fixed points (steady states) and periodic orbits. These sets play a major role in describing transition behavior and associated pattern formation in many fluid flows, such as Rayleigh–Bénard–Marangoni convection and the Taylor–Couette flow [2].

Recently, an overview [3] was given of numerical techniques for computing fixed points and periodic orbits for systems of partial differential equations (PDEs). These PDEs are discretized by any spectral, finite difference or finite element method. Such discretizations result in systems of ordinary differential equations with algebraic constraints (e.g. due to incompressibility) of high dimension, usually of more than 10^6 degrees of freedom. One of the most used methods is the pseudo-arclength continuation method [4], in which branches of steady states are parametrized by an arclength and the augmented problem (an equation representing the arc length normalization is added) is solved by the Newton–Raphson method. The different methods are distinguished by whether the Jacobian matrix is explicitly computed ('matrix-based' methods) or whether matrix vector products of this matrix are used ('matrix-free' methods). The applications shown in [3] range from traditional flows and free-surface flows to magneto-hydrodynamic flows and ocean flows.

* Corresponding author.
E-mail addresses: s.baars@rug.nl (S. Baars), viebahn@cwi.nl (J.P. Viebahn), t.e.mulder@uu.nl (T.E. Mulder), ckuehn@ma.tum.de (C. Kuehn), f.w.wubs@rug.nl (F.W. Wubs), h.a.dijkstra@uu.nl (H.A. Dijkstra).

http://dx.doi.org/10.1016/j.jcp.2017.02.021
0021-9991/

A strong limitation of the numerical bifurcation techniques is that only relatively simple attracting sets can be computed. Although tori can be determined in special cases [5,6], the computation of fractal sets, often associated with chaotic dynamics of fluid systems, is out of the scope of these methods. In this case one has to focus on ergodic properties of these systems, e.g. on the determination of invariant measures, often represented by stationary probability density functions (PDFs). Formally, one can determine these by solving the associated Liouville equation with a number of 'spatial' dimensions equal to the degrees of freedom, but solving these problems is not feasible for high-dimensional systems [7]. Another limitation of numerical bifurcation methods is that when small-scale, non-resolved processes are represented as noise in the equations of the flow model, one needs to solve the Fokker–Planck equation, which has even more computational obstacles than the Liouville equation [8].

In this paper, we focus on stochastic partial differential equations (SPDEs) describing fluid flows for which certain processes have been represented stochastically or for which the forcing of the flow has stochastic properties. The direct way to investigate these flows is to use ensemble simulation techniques for many initial conditions to estimate the PDF for several observables of the flows [9]. Other methods which have been suggested use some form of model order reduction. For example, a stochastic Galerkin technique forms the basis of the Dynamical Orthogonal Field method (DO) [10]. Non-Markovian reduced models can also be obtained by projecting on a basis of eigenvectors of the underlying deterministic system [11].

In a deterministic–stochastic continuation method recently suggested by Kuehn [12], the results from fixed point computation in a deterministic model are used to obtain information on the stationary PDF of the stochastically forced system. A restriction is that linearized dynamics near the fixed point adequately describe the behavior of the stochastic system. In this case, one only needs the leading-order linear approximation and determines the covariance matrix from the solution of a Lyapunov equation. This provides all the information to determine the probability of sample paths. The nice aspect of this method is that one can combine it easily with deterministic pseudo-arclength continuation methods. However, in order to apply the approach to systems of PDEs with algebraic constraints, generalized Lyapunov equations have to be solved.

Direct methods to solve a generalized Lyapunov equation such as Bartels–Stewart algorithm [13] are based on dense matrix solvers and hence inapplicable for large systems. Other existing methods which use low-rank approximations such as Extended and Rational Krylov subspace methods [14–17] and alternating directions implicit (ADI) based iterative methods [18,19] might also become expensive for large-dimensional problems, particularly when trying to use previous initial guesses along a continuation branch.

The aim of this paper is to present new methodology to efficiently trace PDFs of SPDEs with algebraic constraints in parameter space. In Section 2, an extension of the approach suggested in [12] and the novel procedure to efficiently solve a generalized Lyapunov equation numerically are presented. In fact, this solves an open conjecture in [20], which states that it should be possible to reuse previous solutions of a continuation in a specialized Lyapunov solver. We describe our application in Section 3, which is a model of the Atlantic Ocean circulation. The numerical aspects and capabilities of the novel method for this application are shown in Section 4. In Section 5, we provide a summary and discuss the results.

2. Methods

Any ocean-climate model consists of a set of conservation laws (momentum, mass, heat and salt), which are formulated as a set of coupled partial differential equations, that can be written in general form as [21]

$$\mathcal{M}(\mathbf{p})\frac{\partial \mathbf{u}}{\partial t} = \mathcal{L}(\mathbf{p})\mathbf{u} + \mathcal{N}(\mathbf{u},\mathbf{p}) + \mathcal{F}(\mathbf{u},\mathbf{p}), \tag{1}$$

where $\mathcal{L}$, $\mathcal{M}$ are linear operators, $\mathcal{N}$ is a nonlinear operator, $\mathbf{u}$ is the state vector, $\mathcal{F}$ contains the forcing of the system and $\mathbf{p} \in \mathbb{R}^{n_p}$ indicates a vector of parameters. Appropriate boundary and initial conditions have to be added to this set of equations for a well-posed problem.

2.1. Formulation of the problem

When Eq. (1) is discretized, eventually a set of ordinary differential equations with algebraic constraints arises, which can be written as

$$M(\mathbf{p})\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = L(\mathbf{p})\mathbf{x} + N(\mathbf{x},\mathbf{p}) + F(\mathbf{x},\mathbf{p}), \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $M(\mathbf{p}) \in \mathbb{R}^{n\times n}$ is a singular matrix of which every zero row is associated with an algebraic constraint, $L(\mathbf{p}) \in \mathbb{R}^{n\times n}$ is the discretized version of $\mathcal{L}$, and $F : \mathbb{R}^{n_p} \to \mathbb{R}^n$ and $N : \mathbb{R}^n \times \mathbb{R}^{n_p} \to \mathbb{R}^n$ are the finite-dimensional versions of the forcing and the nonlinearity respectively. When noise is added to the forcing, the evolution of the flow can generally be described by a stochastic differential-algebraic equation (SDAE) of the form

$$M(\mathbf{p})\,\mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t;\mathbf{p})\,\mathrm{d}t + \mathbf{g}(\mathbf{x}_t;\mathbf{p})\,\mathrm{d}\mathbf{W}_t, \tag{3}$$

where $\mathbf{f}(\mathbf{x}_t;\mathbf{p}) = L(\mathbf{p})\mathbf{x}_t + N(\mathbf{x}_t,\mathbf{p}) + F(\mathbf{x}_t,\mathbf{p})$ is the right-hand side of Eq. (2), $\mathbf{W}_t \in \mathbb{R}^{n_w}$ is a vector of n_w-independent standard Brownian motions [22], and $\mathbf{g}(\mathbf{x}_t;\mathbf{p}) \in \mathbb{R}^{n_w\times n}$.

Suppose that the deterministic part of Eq. (3) has a stable fixed point $\mathbf{x}^* = \mathbf{x}^*(\mathbf{p})$ for a given range of parameter values. Then linearization around the deterministic steady state yields [12]

$$M(\mathbf{p})\,\mathrm{d}\mathbf{X}_t = A(\mathbf{x}^*;\mathbf{p})\mathbf{X}_t\,\mathrm{d}t + B(\mathbf{x}^*;\mathbf{p})\,\mathrm{d}\mathbf{W}_t, \tag{4}$$

where $A(\mathbf{x};\mathbf{p}) \equiv (D_{\mathbf{x}}\mathbf{f})(\mathbf{x};\mathbf{p})$ is the Jacobian matrix and $B(\mathbf{x};\mathbf{p}) = \mathbf{g}(\mathbf{x}^*;\mathbf{p})$. From now on, we drop the arguments of the matrices A, B and M.

In the special case that M is a non-singular matrix, the equation (4) can be rewritten as

$$\mathrm{d}\mathbf{X}_t = M^{-1}A\mathbf{X}_t\,\mathrm{d}t + M^{-1}B\,\mathrm{d}\mathbf{W}_t, \tag{5}$$

which represents an n-dimensional Ornstein–Uhlenbeck (OU) process. The corresponding stationary covariance matrix C is determined from the following *Lyapunov equation* [22]

$$M^{-1}AC + CA^\top M^{-\top} + M^{-1}BB^\top M^{-\top} = 0. \tag{6}$$

This equation can be rewritten as a *generalized Lyapunov equation*

$$ACM^\top + MCA^\top + BB^\top = 0. \tag{7}$$

If M is a singular diagonal matrix then Eq. (5) does not apply. However, if the stochastic part is non-zero only on the part where M is non-singular (which occurs often when the noise is in the forcing of the flow), then Eq. (4) can be written as

$$\begin{pmatrix} 0 & 0 \\ 0 & M_{22} \end{pmatrix}\begin{pmatrix} \mathrm{d}\mathbf{X}_{t,1} \\ \mathrm{d}\mathbf{X}_{t,2} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} \mathbf{X}_{t,1} \\ \mathbf{X}_{t,2} \end{pmatrix}\mathrm{d}t + \begin{pmatrix} 0 \\ B_2 \end{pmatrix}\mathrm{d}\mathbf{W}_t, \tag{8}$$

where M_{22} represents the non-singular part of M. Consequently, for non-singular A_{11} we can separate Eq. (8) into an algebraic part and an explicitly time-dependent part,

$$A_{11}\mathbf{X}_{t,1} + A_{12}\mathbf{X}_{t,2} = 0, \tag{9a}$$

$$M_{22}\,\mathrm{d}\mathbf{X}_{t,2} = S\mathbf{X}_{t,2}\,\mathrm{d}t + B_2\,\mathrm{d}\mathbf{W}_t, \tag{9b}$$

where $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is the Schur complement of A, which is well-defined since A_{11} is nonsingular as the Jacobian A has eigenvalues strictly in the left-half complex plane by the assumption on deterministic stability of $\mathbf{x}^*$. Since M_{22} is non-singular by construction we can find the stationary covariance matrix C_{22} by solving the corresponding generalized Lyapunov equation

$$SC_{22}M_{22}^\top + M_{22}C_{22}S^\top + B_2B_2^\top = 0. \tag{10}$$

We remark that Eq. (10) could alternatively be derived using an epsilon-embedding approach for the differential algebraic equations. In order to find the full covariance matrix we use $C_{ij} = \mathbb{E}[\mathbf{X}_{t,i}\mathbf{X}_{t,j}^\top]$ together with Eq. (9a) which gives

$$\begin{aligned} C_{12} &= -A_{11}^{-1}A_{12}\mathbb{E}[\mathbf{X}_{t,2}\mathbf{X}_{t,2}^\top] \\ &= -A_{11}^{-1}A_{12}C_{22}, \\ C_{21} &= -\mathbb{E}[\mathbf{X}_{t,2}\mathbf{X}_{t,2}^\top]A_{12}^\top A_{11}^{-\top} \\ &= -C_{22}A_{12}^\top A_{11}^{-\top} = C_{12}^\top, \\ C_{11} &= A_{11}^{-1}A_{12}\mathbb{E}[\mathbf{X}_{t,2}\mathbf{X}_{t,2}^\top]A_{12}^\top A_{11}^{-\top} \\ &= -A_{11}^{-1}A_{12}C_{21}. \end{aligned}$$

In summary, we can obtain an estimate of the covariance matrix C of the stochastic dynamical system (3) by providing the matrices A, B and M and solving the corresponding generalized Lyapunov equation (7), or (10) in case M is singular. Once the covariance matrix C is computed, the stationary PDF of the approximating OU-process, indicated by $p(\mathbf{x})$ follows as [22,23]

$$p(\mathbf{x};\mathbf{x}^*) = \frac{1}{(2\pi)^{\frac{n}{2}}}\,|\,C\,|^{-1/2}\,\mathrm{e}^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^\top C^{-1}(\mathbf{x}-\mathbf{x}^*)}. \tag{11}$$

The limitations of this approach are, firstly, that only a PDF estimate is provided valid on a subexponential time scale before large deviations occur and, secondly, that only the local behavior near the steady state and Gaussian stochastic behavior of the system are obtained.

2.2. A novel iterative generalized Lyapunov solver

The type of systems of the form (7) that we want to solve are typically sparse and have a dimension $n = \mathcal{O}(10^5)$ or larger. Solving systems of this size results in a C that is generally a dense matrix of the same size. This is computationally very expensive in terms of both time and memory. Consequently, one cannot aim to compute the full C but only a low-rank approximation of the form $C \approx VTV^\top$. In existing iterative solution methods for low-rank approximations [18,19,24,14,16] the matrix V is usually computed using repetitive products with B in every iteration, for instance in such a way that it spans the Krylov spaces $\mathcal{K}_m(A, B)$ or $\mathcal{K}_m(A^{-1}, B)$. In practice, however, the matrix B might have many columns, which means that in every iteration of such a method many matrix–vector products have to be performed or many linear systems have to be solved. These operations take up by far the largest amount of time in every iteration, which is why we would like a method that does not expand the search space with the same amount of vectors as the amount of columns in B.

The solution method we propose is based on a Galerkin projection, and is very similar to the method in [24]. It works by solving projected systems of the form

$$V^\top AVTV^\top M^\top V + V^\top MVTV^\top A^\top V + V^\top BB^\top V = 0,$$

where C is approximated by a low-rank approximation $\tilde{C} = VTV^\top$. Now if we take $\tilde{A} = V^\top AV$, $\tilde{M} = V^\top MV$, $\tilde{B} = V^\top B$, we get the smaller generalized Lyapunov equation

$$\tilde{A}T\tilde{M}^\top + \tilde{M}T\tilde{A}^\top + \tilde{B}\tilde{B}^\top = 0, \tag{12}$$

which can be solved by a dense solution routine [13].

A problem that arises when solving generalized Lyapunov equations in an iterative manner is computing an estimate of the residual

$$R = A\tilde{C}M^\top + M\tilde{C}A^\top + BB^\top, \tag{13}$$

for some approximate solution $\tilde{C}$. The (matrix) norm of this residual, which is generally a dense matrix, can be used in a stopping criterion. The 2-norm of the residual matrix is equal to its spectral radius which is defined by the absolute value of the largest eigenvalue. Since the residual is symmetric, approximations of the largest eigenpairs can be computed using only a few steps of the Lanczos method [25]. Even though we cannot compute R explicitly, it is possible to apply the Lanczos method to determine the eigenpair because only matrix vector products Rx are needed, which are evaluated as

$$Rx = A(V(T(V^\top(M^\top x)))) + M(V(T(V^\top(A^\top x)))) + B(B^\top x).$$

The goal of our method is to compute the matrix V, which we could also view as a search space by considering its columns as basis vectors for a linear subspace of $\mathbb{R}^n$. From now on we assume V to be orthonormalized. We suggest to expand V in every iteration by the eigenvectors associated with the largest eigenvalues of the residual, which we already obtained when computing the norm of the residual. The reasoning behind this will be explained below. Now in every iteration, we solve the projected system (12), but because we expanded our search space (with the largest components of the residual), we hope that the new residual is smaller. The resulting algorithm for solving generalized Lyapunov equations is shown in Algorithm 1. Because of the peculiar choice of vectors to expand our space, we call this method the Residual Approximation-based Iterative Lyapunov Solver (RAILS).

2.3. Convergence analysis

We will now show why we choose the eigenvectors associated with the largest eigenvalues of the residual. Here we use $\text{orth}(B)$ to denote the orthonormalization of B. We first show that in a special case, V_k spans the Krylov subspace

$$\mathcal{K}_k(A, B) = \{B, AB, \ldots, A^{k-1}B\}.$$

Proposition 2.1. *If $M = I$, $B \in \mathbb{R}^{n\times m}$, where m is also the amount of vectors we use to expand the space V_k in every iteration and $V_1 = orth(B)$, then $Range(V_k) \subseteq \mathcal{K}_k(A, B)$.*

Proof. For $k = 1$ this is true by assumption. Now say that in step k, (λ, q) is an eigenpair of the residual R_k, and assume that $\text{Range}(V_k) \subseteq \mathcal{K}_k(A, B)$. Then we can write

$$R_k q = \lambda q = AV_k q_1 + V_k q_2 + Bq_3$$

where $q_1 = T_k V_k^\top q$, $q_2 = T_k V_k^\top A^\top q$, $q_3 = B^\top q$. From this it is easy to see that if we orthonormalize q with respect to V_k, it is only nonzero in the direction of AV_k. Now we take $V_{k+1} = [V_k, Q_k]$, where the columns of Q_k are the eigenvectors associated with the m largest eigenvalues of R_k orthonormalized with respect to V_k. Then

$$\begin{aligned}\text{Range}(V_{k+1}) &\subseteq \text{Range}(V_k) \cup \text{Range}(AV_k)\\ &\subseteq \mathcal{K}_k(A, B) \cup A\mathcal{K}_k(A, B) = \mathcal{K}_{k+1}(A, B). \qquad \square\end{aligned}$$

input: A, B, M The problem where $ACM^\top + MCA^\top + BB^\top = 0$.
V_1 Initial space.
m Dimension increase of the space per iteration.
l Maximum amount of iterations.
ϵ Convergence tolerance.
output: V_k, T_k Approximate solution, where $C \approx V_k T_k V_k^\top$.

Orthonormalize V_1
Compute $\tilde{A}_1 = V_1^\top A V_1$
Compute $\tilde{M}_1 = V_1^\top M V_1$
Compute $\tilde{B}_1 = V_1^\top B$
for $j = 1, \ldots, l$ **do**
  Obtain $\tilde{A}_j = V_j^\top A V_j$ by only computing new parts
  Obtain $\tilde{M}_j = V_j^\top M V_j$ by only computing new parts
  Obtain $\tilde{B}_j = V_j^\top B$ by only computing new parts
  Solve $\tilde{A}_j T_j \tilde{M}_j^\top + \tilde{M}_j T_j \tilde{A}_j^\top + \tilde{B}_j \tilde{B}_j^\top = 0$
  Compute the approximate largest m eigenpairs (λ_p, r_p) of the residual R_j using Lanczos
  Stop if the approximated largest eigenvalue is smaller than ϵ
  $V_{j+1} = [V_j, r_1, \ldots, r_m]$
  Re-orthonormalize V_{j+1}
end for

Algorithm 1. RAILS algorithm for the projection based method for solving generalized Lyapunov equations.

Remark 2.1. In case Q_i has full rank for every $i = 1, \ldots, k$ it is clear that actually the equality $\text{Range}(V_k) = \mathcal{K}_k(A, B)$ holds in Proposition 2.1. This is what we observe in practice.

From Proposition 2.1 and Remark 2.1, we see that when we choose m equal to the amount of columns of B, RAILS is equivalent to the method in [24] as long as Q_k has full rank in every iteration. What is important, is that we want to take m much smaller, in which case we assume that eigenvectors associated with the largest eigenvalues of the residual R_k point into the direction of the most important components of AV_k. A similar result holds when we want to look in the Krylov space $\mathcal{K}_k(A^{-1}, A^{-1}B)$.

Proposition 2.2. *If $M = I$, $B \in \mathbb{R}^{n \times m}$, where m is also the amount of vectors we use to expand the space V_k in every iteration, $V_1 = \text{orth}(A^{-1}B)$ and $V_{k+1} = [V_k, Q_k]$, where Q_k are the m eigenvectors associated with the largest eigenvalues of $A^{-1}R_k$ orthonormalized with respect to V_k, then $\text{Range}(V_k) \subseteq \mathcal{K}_k(A^{-1}, A^{-1}B)$.*

Proof. This can be proved analogously to Proposition 2.1. □

We show this result since most other iterative Lyapunov solvers include an operation with A^{-1}. An example is the Extended Krylov method, which looks in the Krylov space $\mathcal{K}_{2k}(A, A^{-k}B)$. We remark that our method, when we start with $V_1 = \text{orth}([B, A^{-1}B])$ and expand with $[Q_k, A^{-1}Q_k]$ is not equivalent to the Extended Krylov method.

We know that if V_k has n orthogonal columns, it spans the whole space, so the solution is in there. To show that our method has finite termination, we argue that the method has converged when the vectors we generate do not have a component perpendicular to V_k, which means that the size of the search space does not increase anymore.

Proposition 2.3. *Take $M = I$ and $B \in \mathbb{R}^{n \times m}$. After k steps of RAILS, the residual R_k has an eigenpair (λ, q) with $\lambda = \|R_k\|_2$. If $q \in \text{Range}(V_k)$, then $V_k T_k V_k^\top$ is the exact solution.*

Proof. We have

$$R_k q = \lambda q = A V_k T_k V_k^\top q + V_k T_k V_k^\top A^\top q + B B^\top q.$$

Since $q \in \text{Range}(V_k)$ and V_k is orthonormalized, it holds that $q = V_k V_k^\top q$. So then

$$\begin{aligned}
\lambda q = \lambda V_k V_k^\top q &= V_k V_k^\top A V_k T_k V_k^\top q + V_k T_k V_k^\top A^\top V_k V_k^\top q + V_k V_k^\top B B^\top V_k V_k^\top q \\
&= V_k \tilde{A}_k T_k V_k^\top q + V_k T_k \tilde{A}_k^\top V_k^\top q + V_k \tilde{B}_k \tilde{B}_k^\top V_k^\top q \\
&= V_k (\tilde{A}_k T_k + T_k \tilde{A}_k^\top + \tilde{B}_k \tilde{B}_k^\top) V_k^\top q \\
&= 0
\end{aligned}$$

which shows us that the residual is zero, so $V_k T_k V_k^\top$ is the exact solution. □

Corollary 2.1. *From Proposition 2.3 it follows that when $m = 1$, the equality $\text{Range}(V_k) = \mathcal{K}_k(A, B)$ holds in Proposition 2.1.*

input: k Iterations after which to restart.
τ Tolerance for the eigenvalues at a restart.

1: Set converged to true if the approximated largest eigenvalue is smaller than ϵ
2: **if** converged and convergence was already achieved earlier **then**
3: **stop**
4: **else if** converged or $\text{mod}(j,k) = 0$ **then**
5: Compute the eigenpairs (λ_p, q_p) of T_j
6: $U = [\,]$
7: **for all** eigenvalues λ_p larger than τ **do**
8: $U = [U, q_p]$
9: **end for**
10: $V_{j+1} = V_j U$
11: $\tilde{A}_{j+1} = U^\top \tilde{A}_j U$
12: $\tilde{M}_{j+1} = U^\top \tilde{M}_j U$
13: $\tilde{B}_{j+1} = U^\top \tilde{B}_j$
14: **end if**

Algorithm 2. Restart method that replaces Line 11 Algorithm 1.

2.4. Restart strategy

A problem that occurs in the method described above is that the space V might get quite large. This means that it can take up a lot of memory, but also that the reduced system, for which we use a dense solver, can become large and take up most of the computation time. For this reason we implemented a restart strategy, where we reduce the size of V after a certain amount of iterations. Usually, not all directions that are present in V are equally important, so we just want to keep the most important ones. We do this by computing the eigenvectors associated with the largest eigenvalues of $VTV^\top$, which are then used as V in the next iteration of our method. Note that since V is orthonormalized, the nonzero eigenvalues of $VTV^\top$ are the same as the eigenvalues of T. The eigenvectors are given by VU, where U are the eigenvectors of T, which makes it quite easy to obtain them.

Besides limiting the size of the reduced problem we have to solve, another advantage is that we reduce the rank of the approximate solution, since we only keep the most important components. This means that we need less memory to store the solution, but also that when we apply the solution, for instance when computing eigenvalues of the solution, we need fewer operations. To assure that we have a solution that has a minimal size, we also apply a restart when RAILS has converged, after which we let it converge once more. This usually leads to an approximation of lower rank.

A downside of restarting an iterative method is that we lose a lot of convergence properties, like for instance the finite termination property that was shown in Proposition 2.3. Since we keep reducing the size of the search space at the time of a restart, it might happen that stagnation occurs. This can be prevented by (automatically) increasing the tolerance of the vectors that we retain during a restart. If we keep doing this repetitively, eventually, the method should still converge.

To implement this restart method, we replaced Line 11 in Algorithm 1 by Algorithm 2.

3. Application in physical oceanography

Motivated by the possibility of multiple steady states in the climate system, we have chosen a problem from physical oceanography, involving the Atlantic Ocean circulation. Its Meridional Overturning Circulation (MOC) is sensitive to freshwater anomalies [26]. Freshening of the surface waters in the Nordic and Labrador Seas diminishes the production of deepwater that feeds the deep southward branch of the MOC. The weakening of the MOC leads to reduced northward salt transport freshening the northern North Atlantic and amplifying the original freshwater perturbation. A MOC collapse, where a transition to a different steady state (weak MOC) occurs within a few decades, may occur due to the existence of a tipping point (e.g. due to a saddle-node bifurcation) associated with this salt-advection feedback. If the MOC is in a multiple steady state regime it may undergo transitions due to the impact of noise. The stochastic nature of the forcing (wind-stress and/or buoyancy flux) may even lead to transitions *before* the actual saddle-node bifurcation has been reached. Hence, it is of central importance to determine what perturbations can cause a transition to the weak MOC state, in particular, which spatial and temporal correlations in the noise are most effective.

We use the spatially quasi two-dimensional model of the Atlantic MOC as described in [27]. In the model, there are two active tracers: temperature T and salinity S, which are related to the density ρ by a linear equation of state

$$\rho = \rho_0 \left(1 - \alpha_T \left(T - T_0\right) + \alpha_S \left(S - S_0\right)\right), \tag{14}$$

where α_T and α_S are the thermal expansion and haline contraction coefficients, respectively, and ρ_0, T_0, and S_0 are reference quantities. The numerical values of the fixed model parameters are summarized in Table 1.

In order to eliminate longitudinal dependence from the problem, we consider a purely buoyancy-driven flow on a non-rotating Earth. We furthermore assume that inertia can be neglected in the meridional momentum equation. The mixing of momentum and tracers due to eddies is parameterized by simple anisotropic diffusion. In this case, the zonal velocity as

Table 1
Fixed model parameters of the two-dimensional ocean model.

D	=	4.0	$\cdot 10^3$	m	H_m	=	2.5	$\cdot 10^2$	m
r_0	=	6.371	$\cdot 10^6$	m	T_0	=	15.0		°C
g	=	9.8		$\mathrm{m\,s^{-2}}$	S_0	=	35.0		psu
A_H	=	2.2	$\cdot 10^{12}$	$\mathrm{m^2\,s^{-1}}$	α_T	=	1.0	$\cdot 10^{-4}$	$\mathrm{K^{-1}}$
A_V	=	1.0	$\cdot 10^{-3}$	$\mathrm{m^2\,s^{-1}}$	α_S	=	7.6	$\cdot 10^{-4}$	$\mathrm{psu^{-1}}$
K_H	=	1.0	$\cdot 10^3$	$\mathrm{m^2\,s^{-1}}$	ρ_0	=	1.0	$\cdot 10^3$	$\mathrm{kg\,m^{-3}}$
K_V	=	1.0	$\cdot 10^{-4}$	$\mathrm{m^2\,s^{-1}}$	τ	=	75.0		days

well as the longitudinal derivatives are zero and the equations for the meridional velocity v, vertical velocity w, pressure p, and the tracers T and S are given by

$$0 = -\frac{1}{\rho_0 r_0}\frac{\partial p}{\partial \theta} + A_V\frac{\partial^2 v}{\partial z^2} + \frac{A_H}{r_0^2}\left(\frac{1}{\cos\theta}\frac{\partial}{\partial\theta}\left(\cos\theta\frac{\partial v}{\partial\theta}\right) + \left(1-\tan^2\theta\right)v\right), \tag{15a}$$

$$0 = -\frac{1}{\rho_0}\frac{\partial p}{\partial z} + g\left(\alpha_T T - \alpha_S S\right), \tag{15b}$$

$$0 = \frac{1}{r_0\cos\theta}\frac{\partial v\cos\theta}{\partial\theta} + \frac{\partial w}{\partial z}, \tag{15c}$$

$$\frac{\partial T}{\partial t} + \frac{v}{r_0}\frac{\partial T}{\partial\theta} + w\frac{\partial T}{\partial z} = \frac{K_H}{r_0^2\cos\theta}\frac{\partial}{\partial\theta}\left(\cos\theta\frac{\partial T}{\partial\theta}\right) + K_V\frac{\partial^2 T}{\partial z^2} + \mathrm{CA}(T), \tag{15d}$$

$$\frac{\partial S}{\partial t} + \frac{v}{r_0}\frac{\partial S}{\partial\theta} + w\frac{\partial S}{\partial z} = \frac{K_H}{r_0^2\cos\theta}\frac{\partial}{\partial\theta}\left(\cos\theta\frac{\partial S}{\partial\theta}\right) + K_V\frac{\partial^2 S}{\partial z^2} + \mathrm{CA}(S). \tag{15e}$$

Here t is time, θ latitude, z the vertical coordinate, r_0 the radius of Earth, g the acceleration due to gravity, A_H (A_V) the horizontal (vertical) eddy viscosity, and K_H (K_V) the horizontal (vertical) eddy diffusivity. The terms with CA represent convective adjustment contributions.

The equations are solved on an equatorially symmetric, flat-bottomed domain. The basin is bounded by latitudes $\theta = -\theta_N$ and $\theta = \theta_N = 60°$ and has depth D. Free-slip conditions apply at the lateral walls and at the bottom. Rigid lid conditions are assumed at the surface and atmospheric pressure is neglected. The wind stress is zero everywhere, and "mixed" boundary conditions apply for temperature and salinity,

$$K_V\frac{\partial T}{\partial z} = \frac{H_m}{\tau}\left(\bar{T}(\theta) - T\right), \tag{16a}$$

$$K_V\frac{\partial S}{\partial z} = S_0 F_s(\theta). \tag{16b}$$

This formulation implies that temperatures in the upper model layer (of depth H_m) are relaxed to a prescribed temperature profile $\bar{T}$ at a rate τ^{-1}, while salinity is forced by a net freshwater flux F_s, which is converted to an equivalent virtual salinity flux by multiplication with S_0. The specification of the CA terms is given in [27].

In Section 3.1 below, we first determine the bifurcation diagram of the deterministic model using pseudo-arclength continuation methods. In the next sections, the case with stochastic freshwater forcing is considered, focusing on validation of the new methods (Section 4.1), comparison with other methods (Section 4.2), numerical aspects (Section 4.3) and potential extensions (Section 4.4).

3.1. Bifurcation diagram

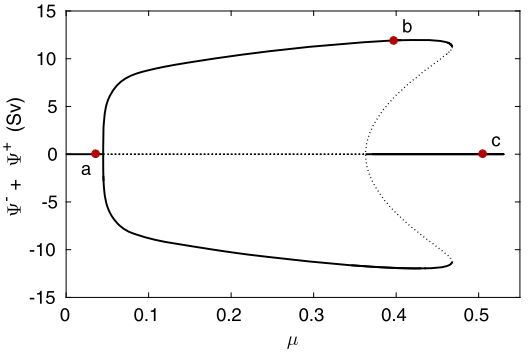
For the deterministic case, we fix the equatorially symmetric surface forcing as

$$\bar{T}(\theta) = 10.0\cos\left(\pi\theta/\theta_N\right), \tag{17a}$$

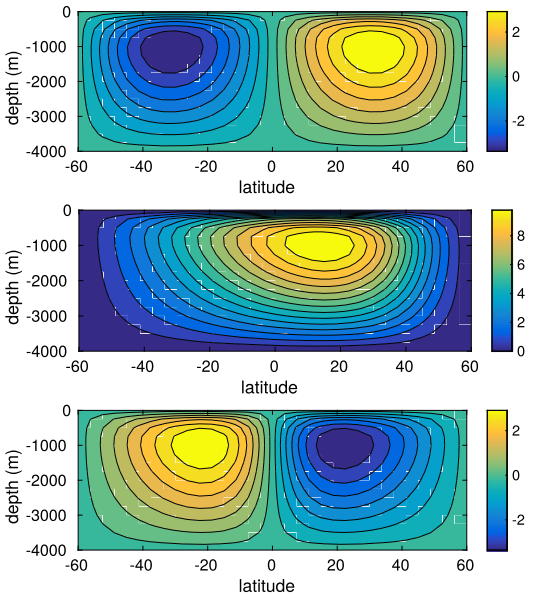
$$F_s(\theta) = \bar{F}_s(\theta) = \mu F_0\frac{\cos(\pi\theta/\theta_n)}{\cos(\theta)}, \tag{17b}$$

where μ is the strength of the mean freshwater forcing (which we take as bifurcation parameter) and $F_0 = 0.1$ m $\mathrm{yr^{-1}}$ is a reference freshwater flux.

The equations are discretized on a latitude-depth equidistant $n_x \times n_y \times n_z$ grid using a second-order conservative central difference scheme. An integral condition expressing the overall conservation of salt is also imposed, as the salinity equation is only determined up to an additive constant. The total number of degrees of freedom is $n = 6n_x n_y n_z$, as there are six unknowns per point. The standard spatial resolution used is $n_x = 4, n_y = 32, n_z = 16$ and the solutions are uniform in the zonal direction, with the zonal velocity $u = 0$.



(a) Bifurcation diagram where a solid line is stable and dashed line is unstable.



(b) The streamfunction at μ_a, μ_b and μ_c respectively.

Fig. 1. (a) Bifurcation diagram of the deterministic equatorially symmetric 2D MOC model, with the forcing as in Eq. (17). (b) Streamfunction pattern at μ_a, μ_b and μ_c respectively.

The bifurcation diagram of the deterministic model for parameters as in Table 1 is shown in Fig. 1a. On the y-axis, the sum of the maximum (Ψ^+) and minimum (Ψ^-) values of the meridional streamfunction Ψ is plotted, where Ψ is defined through

$$\frac{\partial \Psi}{\partial z} = v \cos\theta, \quad -\frac{1}{r_0 \cos\theta}\frac{\partial \Psi}{\partial \theta} = w. \tag{18}$$

For the calculation of the transports, the basin is assumed to have a zonal width of 64°. The value of $\Psi^+ + \Psi^-$ is zero when the MOC is symmetric with respect to the equator.

For small values of μ, a unique equatorially anti-symmetric steady MOC exists of which a pattern at location a is shown in Fig. 1b. This pattern destabilizes at a supercritical pitchfork bifurcation and two asymmetric pole-to-pole solutions appear. An example of the MOC at location b in Fig. 1b shows a stronger asymmetric overturning with sinking in the northern part of the basin. The pole-to-pole solutions cease to exist beyond a saddle-node bifurcation near $\mu = 0.47$ and both branches connect again with the anti-symmetric solution at a second supercritical pitchfork bifurcation. At this bifurcation, the anti-symmetric solution with equatorial sinking (see MOC at location c in Fig. 1b) appears which is stable for larger values of μ. The value of μ at the point b, $\mu_b = 0.40$, will be our reference freshwater forcing.

3.2. Stochastic freshwater forcing

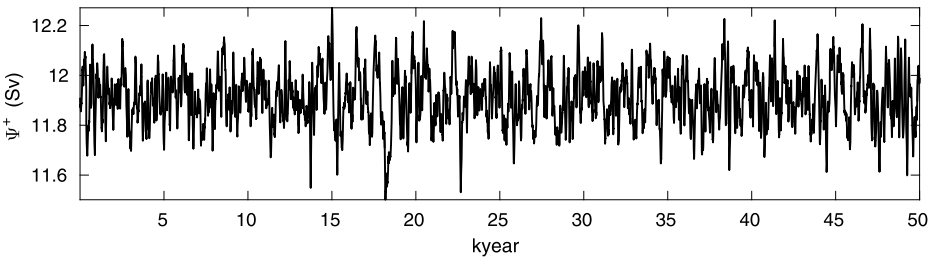
The freshwater forcing is chosen as

$$F_s(\theta, t) = (1 + \sigma\ \zeta(\theta, t))\bar{F}_s(\theta) \tag{19}$$

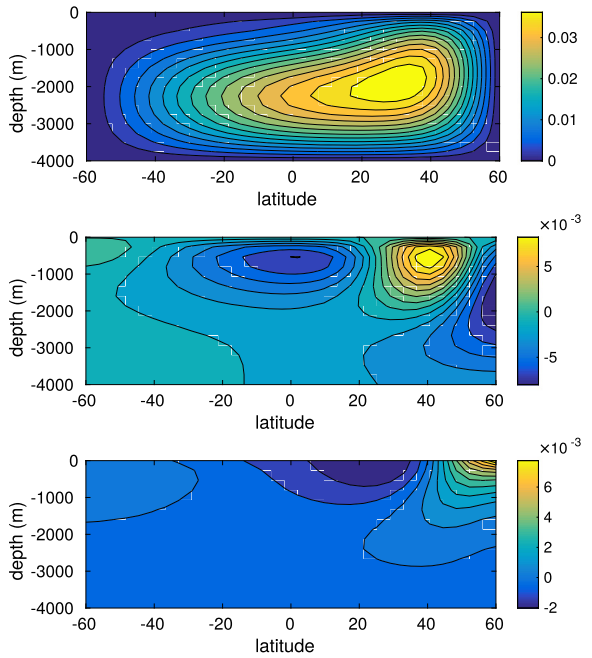
where $\zeta(\theta, t)$ represents zero-mean white noise with a unit standard deviation, i.e., with $\mathbb{E}[\zeta(\theta, t)] = 0$, $\mathbb{E}[\zeta(\theta, t)\zeta(\theta, s)] = \delta(\theta, t - s)$ and $\mathbb{E}[\zeta(\theta, t)\zeta(\eta, t)] = \delta(\theta - \eta, t)$. The constant σ is the standard deviation of the noise which we set to $\sigma = 0.1$. In this case, the noise matrix B in Eq. (4) simply represents additive noise which is (i) only active in the freshwater component, (ii) only present at the surface, (iii) meridionally uncorrelated (unless stated otherwise), and (iv) has magnitude σ of 10% of the deterministic freshwater forcing amplitude at each latitude θ (see Eq. (19)).

4. Results

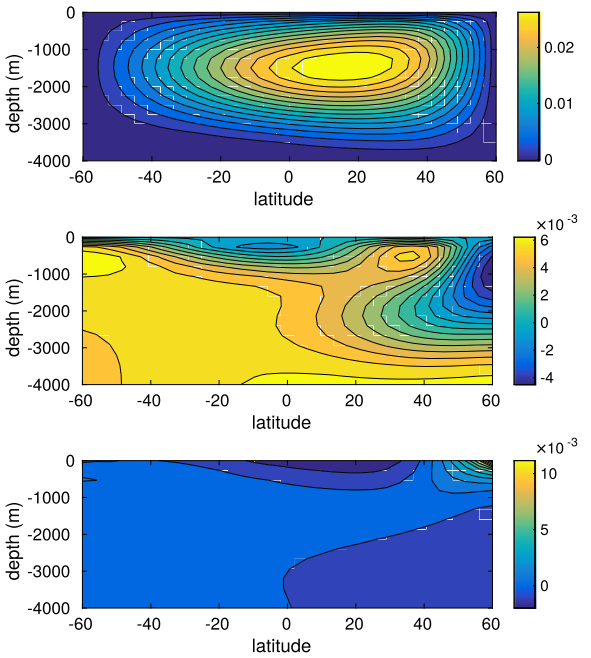
Using the available Jacobian A of the deterministic continuation, the mass matrix M, which is a diagonal matrix with non-zero elements in the T and S rows, and the forcing B as described above, we can determine the local probability distribution of a steady state using the generalized Lyapunov equation (10). We use grid sizes of $4 \times n_y \times 16$, where n_y is a varying amount of grid points in the meridional direction, 16 is the amount of grid points in the vertical direction and



(a) Time series of the MOC maximum Ψ^+.



(b) The first EOF.



(c) The second EOF.

Fig. 2. (a) Time series of the maximum MOC strength Ψ^+ for a 50,000 years simulation of the model for $\mu = \mu_b$ under the freshwater forcing as in Eq. (19). The variability has a slight negative skewness of -0.05 and a kurtosis of 3.04. (b) Patterns of Ψ (top), isothermals (middle) and isohalines (bottom) for the first EOF of this simulation. (c) Similar to (b) but for the second EOF.

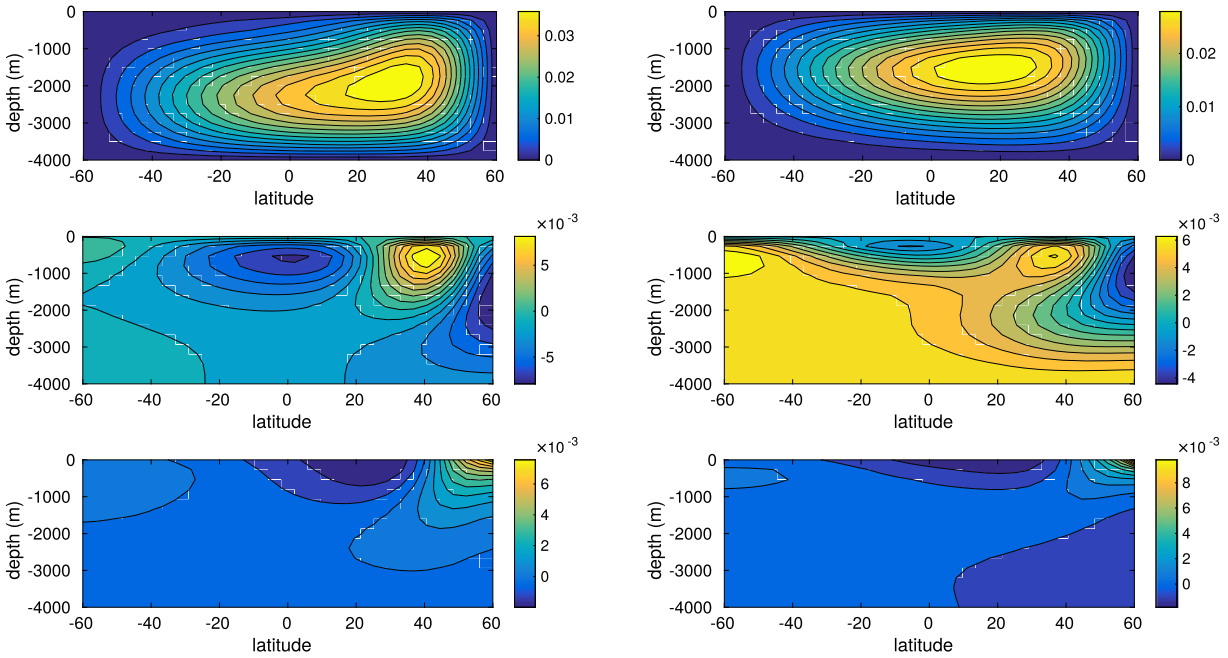
there are 4 grid points in the zonal direction. Since our model is two-dimensional, both the forcing and the solution will be constant in this direction. For the forcing, this means that B contains n_y vectors with the forcing as described in Eq. (19).

For RAILS, we use the algorithm as described in Section 2 and always expand with $m = 3$ vectors per iteration unless stated otherwise. When comparing to other Lyapunov solvers, which were mostly written in Matlab, we use a Matlab implementation of RAILS (Section 4.2), but when we solve larger systems, we prefer to use a C++ implementation (all other sections). Computations are performed on one node of Peregrine, the HPC cluster of the University of Groningen. We use this machine to be able to make fair comparisons with other methods, which use large amounts of memory. Peregrine has nodes with 2 Intel Xeon E5 2680v3 CPUs (24 cores at 2.5 GHz) and each node has 128 GB of memory. Only one core is used in the results below to be able to make fair comparisons.

4.1. Comparison with stochastically forced time forward simulation

A first check of the correctness of the approximate solution of the generalized Lyapunov equations is obtained by comparing the empirical orthogonal functions (EOFs) and weighted eigenvalues of the covariance matrix that we get from both the Lyapunov solver and a stochastically forced time forward simulation at $\mu = \mu_b$, similar to those performed in [28]. This time series (for $n_y = 32$) is plotted in Fig. 2a and shows that Ψ^+ fluctuates around the mean MOC value at μ_b. The patterns of the MOC, the temperature field and the salinity field of both EOF1 and EOF2 are shown in Fig. 2b and c.

The eigensolutions of the generalized Lyapunov equation, also for $n_y = 32$, are shown in Fig. 3. Comparing Fig. 2 with Fig. 3, we see that the results from the transient flow computation and the approximate solution of the generalized Lyapunov equation look very similar. Also, the eigenvalues we find with both methods are very similar, as can be seen in Table 2. The fact that they are not exactly the same is most likely due the fact that the time series takes a long time to converge to a



(a) The first EOF.

(b) The second EOF.

Fig. 3. (a) Patterns of Ψ (top), isothermals (middle) and isohalines (bottom) respectively for the first EOF obtained by solving the generalized Lyapunov equation. (b) Similar to (a) but for the second EOF.

Table 2
First four weighted eigenvalues of the covariance matrix. For the RAILS solver $\|R\|_2/\|BB^T\|_2 < 10^{-2}$ was used as stopping criterion.

Method	λ_1	λ_2	λ_3	λ_4
RAILS	0.677	0.176	0.078	0.033
Dense Lyapunov	0.677	0.176	0.079	0.033
Time series	0.679	0.170	0.082	0.033

statistical steady state. Since the eigenvalues are really close nevertheless, we can indeed use RAILS to compute an estimate of the local probability distribution of steady states of the MOC.

As another check, we use the Bartels–Stewart algorithm [13], which is a dense solver of which the solution time increases with $\mathcal{O}(n^3)$ and the required memory with $\mathcal{O}(n^2)$. The implementation we use is sb03md from the SLICOT library [29]. Results from this method (Table 2, also for $n_y = 32$) confirm that our solution method provides correct solutions.

4.2. Comparison with other Lyapunov solvers

In this section, we compare the results of RAILS to those obtained with standard implementations of the Extended Krylov (EKSM) [14], Projected Extended Krylov (PEKSM) [16], Rational Krylov (RKSM) [15], Tangential Rational Krylov (TKRSM) [17] and Low-rank ADI (LR-ADI) [19] methods. For the LR-ADI method, results with the heuristic shifts from [19] and self-generating shifts based on a Galerkin projection from [30] are reported. Implementations of the Extended Krylov and Rational Krylov methods were obtained from the website of Simoncini [31], whereas the LR-ADI method from M.E.S.S. was used [32].

What we want to show is that RAILS can be competitive without requiring linear system solves in every iteration, which all the methods we compare to require. However, convergence properties of the other methods might be better since they use these linear system solves. For this reason, we also add experiments with our method, where we expand the search space by $S^{-1}r_i$, where r_i are the eigenvectors of the residual associated with the largest eigenvalues. From now on we will refer to this method as Inverse RAILS. To be able to better compare to Extended Krylov, we could also expand our search space by $[r_i, S^{-1}r_i]$, but some preliminary experiments showed that Inverse RAILS performs slightly better.

For Inverse RAILS, Extended/Rational Krylov and ADI based methods, the linear system solves of the form $Sy = x$ are implemented by first computing an LU-factorization of A using UMFPACK whenever possible (available from `lu` in Matlab), and then solving the system

Table 3
Comparison of different Lyapunov solvers. Rank is the rank of the final approximate solution, Dim is the maximum dimension of the approximation space during the iteration, Its is the amount of iterations, MVPs are the amount of matrix–vector products, IMVPs are the amount of inverse matrix–vector products and t_s is the time required for solution of the Lyapunov equation, which includes the computation of the LU-factorization when necessary. For all methods the stopping criterion is a relative residual of 10^{-2}. RAILS was restarted after 50 iterations with a tolerance of $4 \cdot 10^{-6}$ for the eigenpairs that were retained. This is the same tolerance that was used for the other methods to minimize the rank of the approximate solution.

Method	Rank	Dim	Its	MVPs	IMVPs	t_s (s)	ρ
RAILS	59	204	217	634	0	8	$9.2 \cdot 10^{-3}$
Inverse RAILS	60	127	34	128	128	7	$9.5 \cdot 10^{-3}$
Projected RAILS	80	133	36	134	134	9	$9.9 \cdot 10^{-3}$
EKSM	60	448	7	224	256	11	$6.2 \cdot 10^{-3}$
PEKSM	81	704	11	704	384	24	$6.3 \cdot 10^{-3}$
RKSM	60	448	13	448	416	64	$9.2 \cdot 10^{-3}$
TRKSM	60	187	16	219	155	46	$9.7 \cdot 10^{-3}$
LR-ADI (heuristic)	60	800	25	0	480	129	$6.1 \cdot 10^{-3}$
LR-ADI (projection)	60	1312	41	0	800	212	$6.4 \cdot 10^{-3}$

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \tilde{y} \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ x \end{pmatrix}.$$

This is similar to what has been used in LR-ADI methods in [33]. Alternatives would be first computing S and then making a factorization of S, which is not feasible since S tends to be quite dense, or using an iterative method where we need repeated applications of S, which includes solving a system with A_{11}. Both alternatives tend to be slower than the method described above if we add up factorization and solution times.

For the (Tangential) Rational Krylov methods we precompute the initial values $s_0^{(1)}$ and $s_0^{(2)}$ as the eigenvalues of S with the smallest and largest real part. The time it required to compute the eigenvalues is not included in the results.

For the Projected Extended Krylov method, we followed [34] for obtaining the spectral projectors that are required and implemented the method according to [16]. The advantage of this method is that it does not require the Schur complement as we described above. The disadvantage is that instead spectral projectors are required. For this method, we have to compute the projected matrix $V^\top AV$ explicitly, and not implicitly as suggested in [16] to obtain sufficient accuracy. Without this, the Lyapunov solver that was used for the small projected Lyapunov equation would fail to find a solution. For comparison, we implemented the same projection method that was used in [16] also in RAILS, where we chose to expand the basis with $A^{-1}r_i$.

For all methods we use the same stopping criterion, namely that we require the relative residual $\rho = \|R\|_2/\|BB^\top\|_2 < \epsilon$, where $\epsilon = 10^{-2}$, which is sufficient for our application. The resulting absolute residual norm will be around 10^{-5} for the MOC problem with $n_y = 32, 64, 128$. We also show the final relative residual in our results. For RAILS we use a random V_1 as initial guess, since this seemed to give slightly better results than taking B as initial guess. Using a random initial guess also shows that even if storing B in a dense way requires too much memory, we are still able to start our method even if the other methods cannot. During a continuation run, we can of course do much better than a random initial guess by using the approximate solution space from the previous continuation step, but we will not look into that here. For Inverse RAILS, we use $S^{-1}B_2$ as initial guess as suggested by Proposition 2.2. For the same reason, we use $A^{-1}B$ as initial guess for Projected RAILS. Results for the MOC problem with $n_y = 32$ are shown in Table 3. Note that B always has n_y columns.

Let us discuss all columns of Table 3 separately. The first column contains the rank of the approximate solution, which is more-or-less the same for every method, because we did some post-processing on the non-RAILS methods to only keep eigenvectors belonging to eigenvalues that are larger than a certain tolerance. In this case we took $4 \cdot 10^{-6}$. For RAILS, we do not have to do this since the restart method already takes care of this. The projected variants have a larger rank, because they iterate on the full space instead of only the space belonging to the Schur complement.

Dim shows the maximum dimension of the search space during the iteration. Note that this is much smaller for all variants of RAILS than for almost all of the other methods. For the standard RAILS method this is due to the restart that we use, but we restart after 50 iterations, so both other variants of RAILS do not restart, except in the last step when the method already converged and the rank is being minimized. The only other method that has a small maximum space dimension is the Tangential Rational Krylov method, which also expands the space by only a few vectors in each iteration. This is also the reason why we compare to this method.

The next column contains the amount of iterations. We show this just for reference purposes. Every method has a different notion of what an iteration is, so we can not really compare these values. For instance in this case RAILS expands by 3 vectors per iteration, Extended Krylov by 64, and LR-ADI by 32.

Then we show the MVPs, which are the amount of matrix–vector products. For RAILS this is quite high compared to the other methods, but then the advantage is that no linear solves are required, which can be seen if we look at the IMVPs. If we include these in RAILS, which we did in the Inverse RAILS method, we see that it needs far fewer linear solves than the other methods.

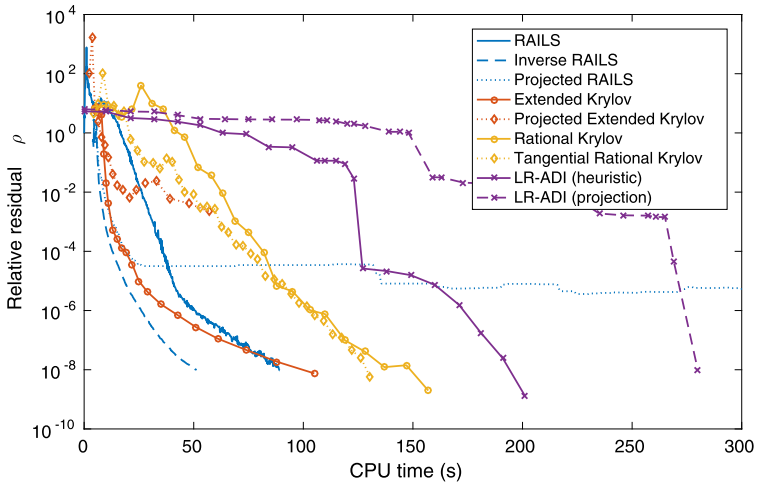


Fig. 4. Convergence history of the relative residual ρ of all the different methods against CPU time. RAILS was restarted after 15 iterations with a tolerance of 10^{-12} for the eigenpairs that were retained and expanded with 10 vectors per iteration.

The most important part of this table is the solution time t_s, from which we can see that all variants of RAILS are faster than all other methods. The fastest method is Inverse RAILS. However, it is only a bit faster than standard RAILS, at the cost of having to solve linear systems in each iteration. For the solution of the linear systems, an LU-factorization was computed beforehand that was utilized by the two variants of RAILS that require a solve, and by the (Projected) Extended Krylov method. The time this takes is included in the solution time. For the LR-ADI and Rational Krylov methods precomputing the LU-factorization is not possible, because in each iteration a different shifted linear system has to be solved. Mainly for this reason RAILS is much faster than Tangential Rational Krylov, even though it uses a larger space.

Lastly, we added a column with the explicitly computed final residual norms, which are all between $6 \cdot 10^{-3}$ and 10^{-2}.

We use quite a loose tolerance for the results in Table 3, which is sufficient for our purposes. We will continue with this loose tolerance in later experiments. However, with this tolerance, it is quite hard to see the convergence properties of the different methods. In Fig. 4, we show a convergence plot with a much smaller tolerance: a relative residual of 10^{-8}. We choose to put CPU time on the x-axis, since there is not really any other quantity that represents a similar amount of work for each method.

What we see in Fig. 4 is that two methods perform really badly: the Projected Extended Krylov method and Projected RAILS. Projected Extended Krylov actually breaks down due to the small projected Lyapunov equation having eigenvalues with opposite signs and Projected RAILS stagnates. This might be due to round-off errors during the projection, so it seems that here the projected variants are not very well suited for solving our problem. The other methods all converge, but for the Krylov type methods, we clearly see that the cost per iteration increases the further they converge. This is because it becomes increasingly expensive to solve the small projected Lyapunov equation. On the other hand, the LR-ADI methods performed increasingly well for smaller tolerances. They are still really expensive however, due to the shifted solves in every iteration. The fastest method, again, is the Inverse RAILS. It also shows almost monotone convergence, except in the first few steps, and converges very rapidly. Standard RAILS performs very well, but convergence is more erratic. It is, however, promising that a method that only uses matrix–vector products can converge faster than the other existing methods that we tried.

In Fig. 5 we show the memory usage of the methods in Fig. 4. We left out the methods that did not converge. Here it is clear that RAILS uses the least amount of memory due to the small amount of vectors that is used to expand the space and due to the restart strategy.

4.3. Numerical scalability

Now we want to show how RAILS behaves when we solve larger systems. We do this by solving the same problem with different grid sizes, namely $4 \times 32 \times 16$, $4 \times 64 \times 16$ and $4 \times 128 \times 16$. For the solution of the generalized Lyapunov equation, when increasing the dimension of our model by a factor 2, the size of the solution increases by a factor 4, since the solution is a square matrix. However, we try to compute a low-rank approximation of the solution, so if the rank of the approximate solution stays the same when we increase the dimension of our model by a factor 2, the size required to store our approximate solution is also increased by a factor 2.

For the MOC problem, if we increase n_y by a factor 2, we know that the amount of columns in B also increases by a factor 2, since those two are equal. From this, we would also expect the rank of the approximate solution to increase. Therefore, we first look at a simplified problem where we use $\hat{B} = B \cdot \mathbf{1}$ as right-hand side, so $\hat{B}$ is a single vector containing

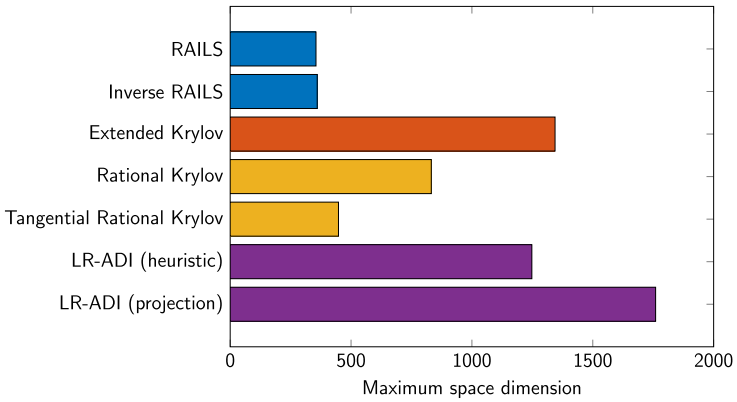


Fig. 5. Memory usage of the different methods as described in Fig. 4 in terms of maximum space dimension. Both projected methods were left out of this figure since they did not converge.

Table 4
Performance of RAILS for different grid sizes with $\hat{B} = B \cdot \mathbf{1}$. The grids are of size $4 \times n_y \times 16$ where n_y is the size in the first column. Rank is the rank of the final approximate solution, Dim is the maximum dimension of the approximation space during the iteration, Its is the amount of iterations, MVPs are the amount of matrix–vector products, t_s is the time required for solution of the Lyapunov equation and t_m is the cost of the matrix–vector product. The stopping criterion is a relative residual of 10^{-2}. RAILS was restarted after 30 iterations with a tolerance of 10^{-5} for the eigenpairs that were retained. We expanded the space with 1 vector in each iteration.

Size	Rank	Dim	Its	MVPs	t_s (s)	t_m (S)
32	12	41	231	223	3	1.5
64	13	42	350	338	13	10
128	13	42	536	518	58	52

the row sums of the original B. For our test problem, this can be seen as a fully space correlated stochastic forcing. We expect that the rank of the approximate solution stays the same.

From Table 4 we see that indeed the rank of the approximate solution does not change when we increase the dimension of the model. However, the number of iterations to find the approximate solution is dependent on the spectral properties of A. And since we are refining, new high frequency parts in the approximate solution will be amplified strongly in the residual evaluation and appear also in the search space. This will slow down the convergence process as can also be seen in Table 4.

We are of course also interested in the increase of the solution time. There are five things that influence this: the length of the vectors that span our basis, the amount of vectors that span our basis, the amount of iterations, the cost of the matrix–vector product, and the cost of solving the projected Lyapunov equation. First of all, the length of the vectors that span our basis increases by a factor 2 when the dimension of the problem is increased by a factor 2, so we can expect a factor 2 in time increase from this. Secondly, the rank of the approximate solution and the dimension of the search space does not increase. This means that also the cost of solving the projected Lyapunov equation does not increase, so we do not expect a solution time increase from this. Then we have the amount of iterations, which does seem to increase by a factor 1.5. And finally we have the cost of the matrix–vector product, which we listed in an extra column of Table 4. Note that this is actually a matrix–vector product with the Schur complement S which requires a linear solve with the matrix block A_{11}, which is relatively expensive. If we subtract the cost of the matrix–vector product, we would expect a factor 3 in time cost increase from the increased vector length and the increased amount of iterations. In Table 4 we observe roughly a factor 2. This being less than 3 might be due to higher efficiency of vector operations for larger vectors.

Going back to the original problem, where we take B as the original stochastic forcing, we expect the solution time to increase by a larger factor, since the projected Lyapunov equation solve and the vector operations become more costly, because the maximum search space dimension and the rank of the approximate solution now do increase. In Table 5 we see that the amount of iterations from $n_y = 32$ to $n_y = 64$ increases by a factor 1.5 again, so we would expect the solution time without matrix–vector products to increase by a factor 3, which is true. The amount of iterations from $n_y = 64$ to $n_y = 128$ increases by a factor 2, so we would expect the time cost to increase by a factor 4, and this is also true. This is because the solution of the projected Lyapunov equation is still relatively cheap for the problems we have here, since those projected equations have at most size 247×247. For problems with a larger maximum search space dimension, the cost of solving this projected Lyapunov equation would become dominant.

Table 5
Performance of RAILS for different grid sizes. The grids are of size $4 \times n_y \times 16$ where n_y is the size in the first column. Rank is the rank of the final approximate solution, Dim is the maximum dimension of the approximation space during the iteration, Its is the amount of iterations, MVPs are the amount of matrix–vector products and t_s is the time required for solution of the Lyapunov equation and t_m is the cost of the matrix–vector product. The stopping criterion is a relative residual of 10^{-2}. RAILS was restarted after 50 iterations with a tolerance of 10^{-5} for the eigenpairs that were retained.

Size	Rank	Dim	Its	MVPs	t_s (s)	t_m (s)
32	59	198	233	682	7	2
64	86	223	340	997	31	17
128	147	247	652	1915	178	115

Table 6
Comparison of different Lyapunov solvers for different grid sizes with $\hat{B} = \mathrm{diag}(B \cdot \mathbf{1})$. The grids are of size $4 \times n_y \times 16$ where n_y is the size in the second column. Rank is the rank of the final approximate solution, Dim is the maximum dimension of the approximation space during the iteration, Its is the amount of iterations, MVPs are the amount of matrix–vector products, IMVPs are the amount of inverse matrix–vector products t_f is the time required for computing the LU-factorization and t_s is the time required for solution of the Lyapunov equation. For all methods the stopping criterion is a relative residual of 10^{-2}. RAILS was restarted after 50 iterations with a tolerance of 10^{-6} for the eigenpairs that were retained. The tolerance that was used in Extended Krylov and Tangential Rational Krylov to minimize the rank of the approximate solution in such a way that the residual was still below the tolerance was set to 10^{-6} and $4 \cdot 10^{-7}$ for the 32 and 64 problems respectively.

Method	Size	Rank	Dim	Its	MVPs	IMVPs	t_f (s)	t_s (s)
EKSM	32	172	763	6	1114	1241	3	43
	64	309	1353	5	1979	2234	15	278
TRKSM	32	177	681	16	808	554	0	109
	64	314	1232	17	1487	977	0	645
RAILS	32	166	312	252	739	0	0	16
	64	276	419	406	1189	0	0	82
	128	563	699	651	1912	0	0	584

4.4. Towards a 3D model

Ultimately, we want to do computations on a full 3D model. To illustrate what will happen for a full 3D model, we include some results where we take the forcing in such a way that it is not zonally averaged, but it is taken such that there is no correlation in the zonal direction. The new forcing can be seen as a diagonal matrix, where all salinity nodes at the surface have a nonzero on the diagonal, and the rest of the matrix is zero. We can write our new forcing, using Matlab notation, as $\hat{B} = \mathrm{diag}(B \cdot \mathbf{1})$ where $\hat{B}$ is the new forcing and B is the original forcing. This means that there are $4 \cdot n_y - 1$ nonzero columns in $\hat{B}$.

We choose to compare RAILS to Extended Krylov, which was the only method that came close to RAILS in the earlier experiments in terms of time, and to the Tangential Rational Krylov method, since this was the only method that came close in terms of maximum space dimension. Since Extended Krylov does not perform well anymore when the projected system becomes really large, which would be the case for this problem, we split $\hat{B}$ into multiple parts. We can do this because for our problem, the right-hand side $\hat{B}\hat{B}^\top$ can be written as

$$\hat{B}\hat{B}^\top = \sum_{i=1}^{4 \cdot n_y - 1} \hat{B}_i \hat{B}_i^\top$$

where $\hat{B}_i$ is the ith column of $\hat{B}$. Since the rank of the approximate solution is highly dependent on the amount of vectors in $\hat{B}$, we expect the maximum space dimension that is needed, and therefore also the size of the projected system, to decrease. After splitting $\hat{B}$, we separately solve the Lyapunov equations with these new right-hand sides, of which we reduce the rank of the approximate solution separately to save memory. Afterwards, we merge the low-rank solutions back into one low-rank solution, which is the approximate solution for the system with $\hat{B}$. We report results with the amount of parts that resulted in the least amount of solution time. The optimal amount of parts that we found was 4, so that is three of size n_y and one of size $n_y - 1$. The maximum space dimension that we report is the maximum space size of solving one part plus the amount of vectors that are required to store the reduced approximate solution of the previous solves. The results with the 3D forcing are shown in Table 6.

We see that in this case RAILS can solve systems much faster and with much less memory than the Extended Krylov and Tangential Rational Krylov methods, even when applying the additional trick that we described above to reduce the

memory usage of the Extended Krylov method. It is also clear that both computing and applying the inverse becomes a real problem for these kinds of systems. RAILS, however, does not need an inverse, and employs a restart strategy to reduce the space size, which is why it performs so much better. The reason why we do not show any results with the Extended Krylov and Tangential Rational Krylov methods for the $4 \times 128 \times 16$ problem is that we did not have enough memory to do these computations, since we chose to not employ an iterative solver.

5. Summary and discussion

We have presented a new method for numerically determining covariance matrices, and hence we can compute probability density functions (PDFs) near steady states of deterministic PDE systems which are perturbed by noise. This method enables the application of the approach suggested by [23,20] to larger systems of SPDEs with algebraic constraints and exploits the structure of the generalized Lyapunov equation in the computations. This approach fits nicely within purely deterministic numerical bifurcation computations [4] and methods to tackle the SPDEs directly [10]. It provides a Gaussian PDF which is valid under linearized dynamics near the deterministic steady state.

Our new algorithm to solve generalized Lyapunov equations is based on a projection method, where solutions are found iteratively by using a set of subspaces V_k. The key new aspects are (i) that at each iteration k, the subspace V_k is expanded with the eigenvectors corresponding to the largest eigenvalues of the residual matrix R, orthogonalized with respect to V_k and itself, and (ii) the restart strategy that is used to keep the space dimension low. We applied this method to a test problem consisting of a quasi two-dimensional model of the Atlantic Ocean circulation. Our method, RAILS, outperforms both Krylov and ADI based methods [14–19] for this test problem.

In practice, one has to provide the Jacobian matrix A of a deterministic fixed point and the matrix B representing the stochastic forcing in order to solve for the stationary covariance matrix C. In a matrix-based pseudo-arclength continuation method the Jacobian is available since a Newton–Raphson method is used [3]. The mass matrix M is readily available from the model equations. The method hence enables us to determine the continuation of local PDFs in parameter space. In particular, the projection approach provides subspaces, which may be re-used directly or by computing predictors along continuation branches.

The availability of the new method opens several new directions of analysis. In one set of applications, A is varied whereas the structure of B is fixed. This typically corresponds to varying a bifurcation parameter and analyzing the corresponding changes in PDF (i.e. covariance matrix C) and transition probabilities (i.e. overlap of PDFs of two steady states) as the steady states (i.e. A) change and a bifurcation point is approached [23]. For example, this could be used in the test problem considered in this study in order to investigate the scaling law in PDF near the saddle-node bifurcation on the branch of pole-to-pole solutions. In this way, the critical slowdown near a tipping point can be studied [28].

For these types of problems, we expect RAILS to perform especially well since it can be restarted using the approximate solution of the previous continuation step, which should result in rapid convergence.

Another interesting application is to fix A (or a set of As that exists for fixed parameter values) and vary B. In this case one steady state is fixed (or two in a regime of two steady states) and the impact of different B ("noise products") on the PDF/covariance matrix C (and possibly transition probabilities) is investigated. Different B can be constructed by changing (a) the dynamical component which is stochastically active (freshwater flux, wind, heat flux), (b) the magnitude, and (c) the pattern (i.e. the cross-correlation structure and the spatial weighting of the auto-covariances). Results from these computations will show the effect of the representation of small scale processes (the 'noise') on the probability density function (under the restriction of linear dynamics).

For the ocean circulation problem it would be interesting to compare the different impacts of freshwater flux versus wind stress noise e.g. on the PDF of the MOC or heat transports. Moreover, regarding the wind stress and (c) one could construct B based on EOFs of the atmospheric variability, as for example considered in [35]. Note that for studying the effect of wind-stress noise, a full three-dimensional model, as developed in [36] is required. In order to construct B from more than one EOF one has to generalize the stochastic forcing to a sum of OU processes. In that case we can use the linearity of the generalized Lyapunov equation and solve for each term separately and combine later on.

In future studies we aim to apply continuation methods, via Kramers'-type formulas adjoined to the continuation problem [23] as well as via transition path numerical methods [37], to perform a full study of transition probabilities of transitions between MOC states in a three-dimensional global ocean model [38]. Knowledge of these transition probabilities is important to evaluate whether climate variability phenomena in the geological past (such as the Dansgaard–Oeschger events) could be caused by multiple steady states of the MOC and whether the transitions are stochastic or induced by crossing bifurcation points [39].

Acknowledgements

We would like to thank the creators of the M.E.S.S. package to make their code available. We also thank Valeria Simoncini for making her code available and for giving suggestions on how to improve our results. And finally we would like to thank the reviewers for their constructive comments. This work is part of the Mathematics of Planet Earth research program with project number 657.000.007, which is financed by the Netherlands Organisation for Scientific Research (NWO). TEM and HAD acknowledge support by the Netherlands Earth System Science Centre (NESSC), financially supported by the Ministry

of Education, Culture and Science (OCW), Grant no. 024.002.001. CK has been supported by an APART fellowship of the Austrian Academy of Sciences and by a Lichtenberg Professorship of the VolkswagenStiftung.

References

[1] R. Teman, Infinite-Dimensional Dynamical Systems in Mechanics and Physics, Springer Series in Applied Mathematics, Springer-Verlag, Berlin, 1988.
[2] E.L. Koschmieder, Bénard Cells and Taylor Vortices, Cambridge University Press, Cambridge, UK, 1993.
[3] H.A. Dijkstra, F.W. Wubs, A.K. Cliffe, E. Doedel, I.F. Dragomirescu, B. Eckhardt, A.Yu. Gelfgat, A.L. Hazel, V. Lucarini, A.G. Salinger, E.T. Phipps, J. Sanchez-Umbria, H. Schuttelaars, L.S. Tuckerman, U. Thiele, Numerical bifurcation methods and their application to fluid dynamics: analysis beyond simulation, Commun. Comput. Phys 15 (1) (2014) 1–45, http://dx.doi.org/10.4208/cicp.240912.180613a.
[4] H.B. Keller, Numerical solution of bifurcation and nonlinear eigenvalue problems, in: P.H. Rabinowitz (Ed.), Applications of Bifurcation Theory, Academic Press, New York, U.S.A., 1977, pp. 359–384.
[5] J. Sánchez, M. Net, A parallel algorithm for the computation of invariant tori in large-scale dissipative systems, Physica D 252 (2013) 22–33, http://dx.doi.org/10.1016/j.physd.2013.02.008.
[6] D. Puigjaner, J. Herrero, C. Simó, F. Giralt, From steady solutions to chaotic flows in a Rayleigh–Bénard problem at moderate Rayleigh numbers, Physica D 240 (11) (2011) 920–934, http://dx.doi.org/10.1016/j.physd.2011.01.007.
[7] A. Lasota, M.C. Mackey, Chaos, Fractals and Noise, 2nd edition, Springer-Verlag, Berlin/Heidelberg, 1994.
[8] J.S. Chang, G. Cooper, A practical difference scheme for Fokker–Planck equations, J. Comput. Phys. 6 (1) (1970) 1–16, http://dx.doi.org/10.1016/0021-9991(70)90001-X.
[9] J. Slingo, T. Palmer, Uncertainty in weather and climate prediction, Philos. Trans. R. Soc., Math. Phys. Eng. Sci. 369 (1956) (2011) 4751–4767, http://dx.doi.org/10.1098/rsta.2011.0161.
[10] T.P. Sapsis, P.F.J. Lermusiaux, Dynamically orthogonal field equations for continuous stochastic dynamical systems, Physica D 238 (23–24) (2009) 2347–2360, http://dx.doi.org/10.1016/j.physd.2009.09.017.
[11] M.D. Chekroun, H. Liu, S. Wang, Stochastic Parameterizing Manifolds and Non-Markovian Reduced Equations, Springer International Publishing, 2015.
[12] C. Kuehn, Deterministic continuation of stochastic metastable equilibria via Lyapunov equations and ellipsoids, SIAM J. Sci. Comput. 34 (3) (2012) A1635–A1658, http://dx.doi.org/10.1137/110839874.
[13] R.H. Bartels, G. Stewart, Solution of the matrix equation AX + XB = C [F4], Commun. ACM 15 (9) (1972) 820–826, http://dx.doi.org/10.1145/361573.361582.
[14] V. Simoncini, A new iterative method for solving large-scale Lyapunov matrix equations, SIAM J. Sci. Comput. 29 (3) (2007) 1268–1288, http://dx.doi.org/10.1137/06066120X.
[15] V. Druskin, V. Simoncini, Adaptive rational Krylov subspaces for large-scale dynamical systems, Syst. Control Lett. 60 (8) (2011) 546–560, http://dx.doi.org/10.1016/j.sysconle.2011.04.013.
[16] T. Stykel, V. Simoncini, Krylov subspace methods for projected Lyapunov equations, Appl. Numer. Math. 62 (1) (2012) 35–50, http://dx.doi.org/10.1016/j.apnum.2011.09.007.
[17] V. Druskin, V. Simoncini, M. Zaslavsky, Adaptive tangential interpolation in rational Krylov subspaces for MIMO dynamical systems, SIAM J. Matrix Anal. Appl. 35 (2) (2014) 476–498, http://dx.doi.org/10.1137/120898784.
[18] D. Kleinman, On an iterative technique for Riccati equation computations, IEEE Trans. Autom. Control 13 (1) (1968) 114–115, http://dx.doi.org/10.1109/TAC.1968.1098829.
[19] T. Penzl, A cyclic low-rank Smith method for large sparse Lyapunov equations, SIAM J. Sci. Comput. 21 (4) (1999) 1401–1418, http://dx.doi.org/10.1137/S1064827598347666.
[20] C. Kuehn, Numerical continuation and SPDE stability for the 2D cubic–quintic Allen–Cahn equation, SIAM/ASA J. Uncertain. Quantificat. 3 (1) (2015) 762–789, http://dx.doi.org/10.1137/140993685.
[21] S.M. Griffies, Fundamentals of Ocean-Climate Models, Princeton University Press, Princeton, USA, 2004.
[22] C. Gardiner, Stochastic Methods: A Handbook for the Natural and Social Sciences, Springer, Berlin, 2009.
[23] C. Kuehn, A mathematical framework for critical transitions: bifurcations, fast–slow systems and stochastic dynamics, Physica D 240 (12) (2011) 1020–1035, http://dx.doi.org/10.1016/j.physd.2011.02.012.
[24] Y. Saad, Numerical solution of large Lyapunov equations, in: Signal Processing, Scattering and Operator Theory, and Numerical Methods, Proc. MTNS-89, Birkhäuser, 1990, pp. 503–511.
[25] C. Lanczos, An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators, United States Governm. Press Office, 1950.
[26] H. Stommel, Thermohaline convection with two stable regimes of flow, Tellus 2 (1961) 224–230, http://dx.doi.org/10.1111/j.2153-3490.1961.tb00079.x.
[27] M. den Toom, H.A. Dijkstra, F.W. Wubs, Spurious multiple equilibria introduced by convective adjustment, Ocean Model. 38 (1–2) (2011) 126–137, http://dx.doi.org/10.1016/j.ocemod.2011.02.009.
[28] M. van der Mheen, H.A. Dijkstra, A. Gozolchiani, M. den Toom, Q. Feng, J. Kurths, E. Hernandez-Garcia, Interaction network based early warning indicators for the Atlantic MOC collapse, Geophys. Res. Lett. 40 (11) (2013) 2714–2719, http://dx.doi.org/10.1002/grl.50515.
[29] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, A. Varga, SLICOT – a subroutine library in systems and control theory, in: Applied and Computational Control, Signals, and Circuits, Springer, 1999, pp. 499–539.
[30] P. Benner, P. Kürschner, J. Saak, Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations, Electron. Trans. Numer. Anal. 43 (2014) 142–162, http://dx.doi.org/10.17617/2.2071065.
[31] V. Simoncini, Available online software [cited 26-09-2016].
[32] J. Saak, M. Köhler, P. Benner, M-M.E.S.S.-1.0.1 – The Matrix Equations Sparse Solvers library, Apr. 2016, http://dx.doi.org/10.5281/zenodo.50575.
[33] F.D. Freitas, J. Rommes, N. Martins, Gramian-based reduction method applied to large sparse power system descriptor models, IEEE Trans. Power Syst. 23 (3) (2008) 1258–1270, http://dx.doi.org/10.1109/TPWRS.2008.926693.
[34] R. März, Selected topics in numerical methods canonical projectors for linear differential algebraic equations, Comput. Math. Appl. 31 (4) (1996) 121–135, http://dx.doi.org/10.1016/0898-1221(95)00224-3.
[35] H.A. Dijkstra, L.M. Frankcombe, A.S. von der Heydt, A stochastic dynamical systems view of the Atlantic Multidecadal Oscillation, Philos. Trans. R. Soc., Math. Phys. Eng. Sci. 366 (1875) (2008) 2543–2558, http://dx.doi.org/10.1098/rsta.2008.0031.
[36] A. De Niet, F. Wubs, A.T. van Scheltinga, H.A. Dijkstra, A tailored solver for bifurcation analysis of ocean-climate models, J. Comput. Phys. 227 (1) (2007) 654–679, http://dx.doi.org/10.1016/j.jcp.2007.08.006.

[37] G. Henkelman, B. Uberuaga, H. Jónsson, A climbing image nudged elastic band method for finding saddle points and minimum energy paths, J. Chem. Phys. 113 (2000) 9901–9904, http://dx.doi.org/10.1063/1.1329672.
[38] J. Thies, F. Wubs, H.A. Dijkstra, Bifurcation analysis of 3D ocean flows using a parallel fully-implicit ocean model, Ocean Model. 30 (4) (2009) 287–297, http://dx.doi.org/10.1016/j.ocemod.2009.07.005.
[39] P.D. Ditlevsen, S.J. Johnsen, Tipping points: early warning and wishful thinking, Geophys. Res. Lett. 37 (19) (2010) L19703, http://dx.doi.org/10.1029/2010GL044486.