University of Groningen

Numerical methods for studying transition probabilities in stochastic ocean-climate models

Baars, Sven

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2019

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):
Baars, S. (2019). *Numerical methods for studying transition probabilities in stochastic ocean-climate models*. Rijksuniversiteit Groningen.

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license. More information can be found on the University of Groningen website: https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment.

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Download date: 21-06-2022

Numerical methods for studying transition probabilities in stochastic ocean-climate models

Sven Baars

The work in this thesis has been carried out at the Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence of the University of Groningen. It is part of the Mathematics of Planet Earth research program, which is financed by the Netherlands Organization for Scientific Research (NWO).

Copyright © 2019 Sven Baars

Printed by Gildeprint

ISBN 978-94-034-1710-3 (printed version)
ISBN 978-94-034-1709-7 (electronic version)

rijksuniversiteit
groningen

Numerical methods for studying transition probabilities in stochastic ocean-climate models

Proefschrift

ter verkrijging van de graad van doctor aan de
Rijksuniversiteit Groningen
op gezag van de
rector magnificus prof. dr. E. Sterken
en volgens besluit van het College voor Promoties.

De openbare verdediging zal plaatsvinden op

vrijdag 21 juni 2019 om 14:30 uur

door

Sven Baars

geboren op 15 augustus 1990
te Ulrum

Promotor
Prof. dr. ir. R.W.C.P. Verstappen

Copromotor
Dr. ir. F.W. Wubs

Beoordelingscommissie
Prof. dr. R.H. Bisseling
Prof. dr. D.T. Crommelin
Prof. dr. A.J. van der Schaft

CONTENTS

1 Introduction **1**

2 Basic concepts **7**

2.1 Newton's method 7
2.2 Iterative methods 8
2.3 Bifurcation analysis 11
  2.3.1 Pseudo-arclength continuation 12
2.4 Stochastic differential equations 15
  2.4.1 Brownian motion 15
  2.4.2 Stochastic differential equations 16
  2.4.3 The Euler–Maruyama method 17
  2.4.4 The stochastic theta method 17
2.5 Governing equations 18
  2.5.1 The Navier–Stokes equations 18
  2.5.2 The ocean model 18
  2.5.3 Stochastic freshwater forcing 20

3 Linear systems **21**

3.1 The two-level ILU preconditioner 24
  3.1.1 Initialization phase 25
  3.1.2 Factorization phase 27
  3.1.3 Solution phase 28
3.2 The multilevel ILU preconditioner 28
3.3 Skew partitioning in 2D and 3D 30
3.4 Numerical results 33

3.4.1 Weak scalability 35
3.4.2 Strong scalability 39
3.4.3 Lid-driven cavity 41
3.5 Summary and Discussion 44

4 Lyapunov equations 47
4.1 Methods 48
4.1.1 Formulation of the problem 48
4.1.2 A novel iterative generalized Lyapunov solver 50
4.1.3 Convergence analysis 52
4.1.4 Restart strategy 54
4.1.5 Extended generalized Lyapunov equations 55
4.2 Problem setting 57
4.2.1 Bifurcation diagram 57
4.2.2 Stochastic freshwater forcing 58
4.3 Results 58
4.3.1 Comparison with stochastically forced time forward simulation 59
4.3.2 Comparison with other Lyapunov solvers 60
4.3.3 Numerical scalability 66
4.3.4 Towards a 3D model 68
4.3.5 Continuation 70
4.3.6 Extended Lyapunov equations 71
4.4 Summary and Discussion 72

5 Transition probabilities 75
5.1 Definition 75
5.2 The Eyring–Kramers formula 76
5.2.1 Double well potential 77
5.2.2 Computing the transition probability 77
5.3 Covariance ellipsoids 78
5.3.1 Example 79
5.4 Most probable transition trajectories 80
5.5 Computing transition probabilities 81
5.5.1 Direct sampling 81
5.5.2 Direct sampling of the mean first passage time 82
5.5.3 Adaptive multilevel splitting 82
5.5.4 Trajectory-Adaptive Multilevel Sampling 91
5.5.5 Genealogical Particle Analysis 95
5.5.6 Comparison 96
5.6 Summary and Discussion 101

6 Transitions in the Meridional Overturning Circulation 103
6.1 Projected time-stepping in TAMS 104

6.2 Problem setting . . . 105
6.2.1 Bifurcation diagram . . . 105
6.2.2 Stochastic freshwater forcing . . . 106
6.2.3 Reaction coordinate . . . 106
6.3 Results . . . 107
6.4 Summary and Discussion . . . 108

7 Conclusion 111

Publications and preprints 115

Software 117

Bibliography 119

Summary 131

Samenvatting 135

Soamenvatting 139

Acknowledgments 141

INTRODUCTION

You may be reading this thesis on a sunny day in August in your garden surrounded by flowers, or it may be a rainy Sunday afternoon near the fireplace with your feet up, but most likely it's just another day in the office. Fact is that there is a strong variability in the every day weather related to the development of high- and low-pressure fields. These developments do not have anything to do with the external solar forcing, but are due to the internal variability of the atmospheric flow (Dijkstra, 2005). The time scale of this variability of 3-7 days is determined by the nonlinear processes in the atmosphere itself.

Similar processes happen in every part of the climate system on varying time scales. A high-frequency process is the weather as described above; a low-frequency process is the change in land-ice distribution (Mulder et al., 2018). In the ocean, internal variability causes formation of ocean eddies and the meandering of ocean currents such as the Gulf Stream (Schmeits and Dijkstra, 2001). Interaction between all the different processes on different time scales may result in internal variability on other time scales that is not present in decoupled systems. This makes such processes very difficult to study, and difficult to understand. Therefore it is often a good idea to take a step back, and look at a more simplified model to make sure one is at least able to understand the results, and then apply this newly acquired knowledge to the fully coupled system.

In this thesis, we work with such a simplified model of the Meridional Overturning Circulation (MOC). The MOC consists of a global 'conveyor belt' of ocean currents, which are driven by wind stress forces and fluxes of heat and freshwater at the surface. In the Atlantic ocean, it consists of surface currents that transport relatively light water toward high latitudes, deep water currents going in the opposite direction, and sinking and upwelling processes that connect these two. The circulation system contains two overturning cells: one in the north with North Atlantic Deep Water (NADW) and one in the south with Antarctic Bottom Water (AABW). The Atlantic part of the MOC (AMOC) is shown in Figure 1.1.

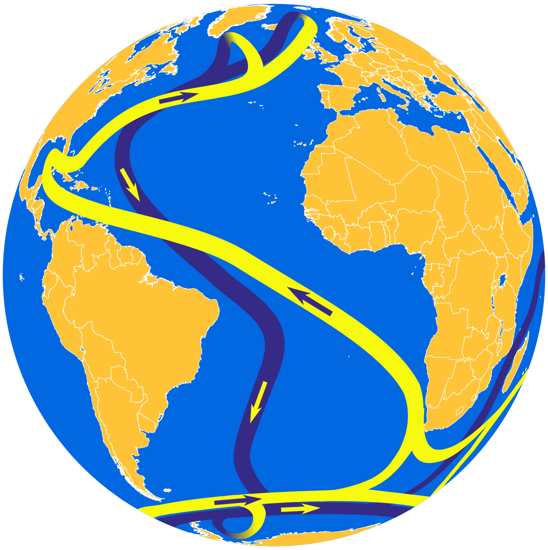


Figure 1.1: Simplified depiction of the Meridional Overturning Circulation in the Atlantic ocean. The yellow paths indicate the circulation at the surface, and the blue paths indicate the deep-water currents. The gradients between the two colors indicate the overturning cells with NADW in the north and AABW cell in the south.

Since the first model proposed by Stommel (1961), many model studies have shown that the MOC may be sensitive to variability in the freshwater forcing. In a global coupled climate model by Vellinga and Wood (2002) the NADW circulation collapses and recovers after 120 years. This is because weakening the MOC by introducing more freshwater in the North Atlantic (melting of the Greenland ice sheet) leads to a reduced northward saltwater transport, which in turn amplifies the freshwater perturbation.

A state of the MOC with a strongly reduced heat transport may have large consequences for the global climate (Rahmstorf, 2000; Lenton et al., 2008). In Vellinga and Wood (2002), a cooling of a few degrees is observed in Europe, which in turn may lead to growing glaciers and then global cooling. Therefore, an estimate of the probability of MOC transitions is crucial for prediction of a collapse of the MOC and with that a rapid climate change.

This collapse may occur due to the existence of a tipping point associated with the salt-advection feedback (Walin, 1985; Lenton, 2011). Mathematically, this tipping point is referred to as a bifurcation point. Tipping points exist due to the presence of multiple stable steady states for the same parameter values. Due to unresolved small-scale variability, however, transitions may even be observed before a tipping point is reached. This unresolved variability is often represented as noise. An example of such a noise induced transition in a simplified 2D model of the MOC is shown in Figure 1.2.

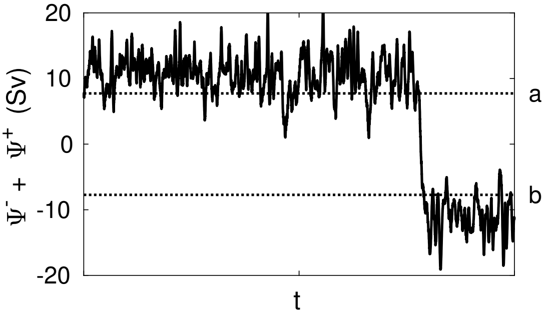


Figure 1.2: Example of a noise induced transition between steady states a and b in a simplified 2D model of the MOC. We are interested in the computation of the probability of such transitions. On the y-axis, the sum of the maximum and minimum of the overturning streamfunction Ψ are shown, meaning that steady state a represents the current state of the MOC as depicted in Figure 1.1, and steady state b rotates in the opposite direction.

The effect of noise on MOC transitions has so far been studied in very idealized models (Cessi, 1994; Monahan et al., 2008; Timmermann and Lohmann, 2000) and more recently in a coupled climate model of intermediate complexity (Sijp et al., 2013). Knowledge of the probability of these transitions is also important to evaluate whether climate variability phenomena in the geological past, such as the Dansgaard–Oeschger events, could be caused by stochastic transitions or by transitions induced by crossing tipping points (Ditlevsen and Johnsen, 2010). The aim of this thesis is to develop methods which can be used for studying transition behavior in the MOC.

To be able to observe these transitions between stable steady states, we first need to be able to compute the steady states themselves. They can be computed by using time integration, which is often expensive, especially if steady states have to be computed for multiple parameter values. Instead one can apply a method called (pseudo-arclength) continuation (Keller, 1977; Crisfield, 1991), which can compute the steady states directly by applying Newton's method. This can speed up the computation of steady states considerably (Dijkstra et al., 2014). Pseudo-arclength continuation is especially useful for computing unstable steady states, i.e. steady states for which a small disturbance causes the state to converge to a different steady state. Time integration methods are usually incapable of computing these unstable steady states.

During the application of Newton's method, many linear systems of equations have to be solved. For the MOC, these are specifically equations that include the Navier–Stokes equations, which are discretized on a staggered grid. In Benzi et al. (2005), the authors gave a survey of methods that are currently used to solve linear systems of this type. Direct (sparse) solvers are not practical since a typical ocean model involves millions of unknowns leading to a huge memory requirement for storing the factorization and an enormous amount of time to compute it. For such problems, iterative methods are preferred, e.g. Krylov subspace methods with suitable preconditioning to ensure robustness, fast convergence and accuracy of the final approximate solution (Benzi et al., 2005; Benzi and Olshanskii, 2006; De Niet et al., 2007; Elman et al., 2002; Kay et al., 2002; Elman et al., 2008). Segal et al. (2010) give an overview of the present state of fast solvers for the incompressible Navier–Stokes equations discretized by the finite element method and linearized by Newton's or Picard's method.

Preconditioners that are often advocated include standard additive Schwarz domain-decomposition, multigrid with aggressive coarsening and strong smoothers (e.g. ILU), and 'block preconditioners' that use an approximate block LU factorization and some approximation of the Schur complement associated with the pressure unknowns, e.g. SIMPLEC, LSC, and PCD (Cyr et al., 2012).

Another class of Schur complement methods is obtained when eliminating the interior of geometric partitions (or subdomains) and constructing a suitable approximation of the reduced system on the separator. In Wubs and Thies (2011); Thies and Wubs (2011) it was shown how to construct a preconditioner for the Navier–Stokes equations, which are discretized on a staggered grid. The resulting operator acts in the divergence-free space, which allows the method to handle the saddle-point structure of the system in a natural way. In Chapter 3 we first present a novel multilevel variant of this method. This method is also described in Baars et al. (2019c), and used in Baars et al. (2019b) for studying bifurcations in a lid-driven cavity.

After computing steady states of the MOC, we are interested in its sensitivity to noise. The sensitivity around a steady state can be determined from the probability density function. It is well known that this probability density function can be computed from the solution of a generalized Lyapunov equation (Gardiner, 1985). Direct methods for solving a generalized Lyapunov equation such as Bartels–Stewart algorithm (Bartels and Stewart, 1972) are based on dense matrix solvers and hence inapplicable for large systems. Other existing methods which use low-rank approximations (Simoncini, 2007; Druskin and Simoncini, 2011; Stykel and Simoncini, 2012; Druskin et al., 2014; Kleinman, 1968; Penzl, 1999) might also become expensive for high-dimensional problems, particularly when trying to use previous initial guesses along a continuation branch. We therefore present a novel method for computing the solution of generalized Lyapunov equations that is particularly well suited for our ocean problem in a continuation context in Chapter 4. Most of this chapter has been published in Baars et al. (2017) and Dijkstra et al. (2016), but some extra work has been done in the context of continuation and extended generalized Lyapunov equations.

A shortcoming to this above approach is that it only describes the sensitivity to noise around a steady state. To study the more global phenomenon of transitions between steady states, stochastic time integration is required. Applying a standard Monte Carlo method, however, is way too expensive, especially for high-dimensional systems and when the probability of a transition is small. Multiple methods exist to work around this problem by applying some form of resampling (Cérou and Guyader, 2007; Rolland and Simonnet, 2015; Lestang et al., 2018; Moral and Garnier, 2005; Moral, 2013; Wouters and Bouchet, 2016). We discuss these methods and show their benefits and shortcomings when applied to transitions between steady states on high-dimensional systems in Chapter 5. In Chapter 6 we propose a method based on the solution of generalized Lyapunov equations that reduces the computational cost and the huge memory requirements that are common for these types of methods. This chapter, and parts of Chapter 5 will appear in Baars et al. (2019a). We also used one of the methods described in Chapter 5 in Mulder et al. (2018) to study transition behavior in marine ice sheet instability problems.

BASIC CONCEPTS

In this chapter we discuss some of the basic concepts that will be used throughout this thesis. We start with Newton's method, which we use in both continuation methods and implicit time stepping methods, and then continue with iterative methods that we use for solving the linear systems that arise in Newton's method. After this, we describe the concept of bifurcation analysis and the pseudo-arclength continuation method which we use to perform this bifurcation analysis. Finally, we describe the equations that are used in our ocean model and discuss stochastic differential equations which we use to represent unresolved small-scale variability in the ocean.

Newton's method 2.1

Newton's method, or the Newton–Raphson method, is an iterative method for computing the roots of a function $F : \mathbb{R}^n \to \mathbb{R}^n$ (Atkinson, 1988). It can be derived directly from the Taylor series

$$F(\boldsymbol{x} + \Delta\boldsymbol{x}) = F(\boldsymbol{x}) + J(\boldsymbol{x})\Delta\boldsymbol{x} + \mathcal{O}(\Delta\boldsymbol{x}^2),$$

where $J(\boldsymbol{x})$ is the Jacobian of F. Assuming that $F(\boldsymbol{x}+\Delta\boldsymbol{x}) = 0$, and neglecting the second order term $\mathcal{O}(\Delta\boldsymbol{x}^2)$, we obtain a linear system

$$J(\boldsymbol{x})\Delta\boldsymbol{x} = -F(\boldsymbol{x}),$$

which can be solved for $\Delta\boldsymbol{x}$. $\boldsymbol{x} + \Delta\boldsymbol{x}$ will now be a better approximation of the actual root. Starting at some initial guess $\boldsymbol{x}_0$, we can apply this procedure iteratively, as is shown in Algorithm 1.

```
input:   $x_0$     Initial guess.
         $F, J$    Function for which we want to find a root
                   and its Jacobian.
output:  $x_k$     Approximate solution.

for $i = 1, 2, \ldots$ until convergence do
    solve $J(x_{i-1})\Delta x_i = -F(x_{i-1})$
    $x_i = x_{i-1} + \Delta x_i$
```

Algorithm 1: Newton's method for computing the roots of $F(\boldsymbol{x})$.

2.2 Iterative methods

Linear systems as the one described in the previous section are often large and sparse. In this case, standard methods such as LU factorization with forward-backward substitution are often not efficient enough, and one instead resides to iterative methods (Van der Vorst, 2003). The general idea behind iterative methods is that we want to solve the system $A\boldsymbol{x} = \boldsymbol{b}$, and at each iteration i, we try to get a better approximate solution $\boldsymbol{x}_i$. We may also write this as $\boldsymbol{x} = \boldsymbol{x}_i + \boldsymbol{\epsilon}_i$, where $\boldsymbol{\epsilon}_i$ is the error at step i. Multiplication by A gives us

$$A\boldsymbol{\epsilon}_i = A(\boldsymbol{x} - \boldsymbol{x}_i) = \boldsymbol{b} - A\boldsymbol{x}_i.$$

Since we do not have the exact solution, we do not know the exact error either. Instead, we can try to solve the system

$$M\boldsymbol{z}_i = \boldsymbol{b} - A\boldsymbol{x}_i$$

for $\boldsymbol{z}_i$, with M an approximation of A that makes this system much easier to solve than the system $A\boldsymbol{x} = \boldsymbol{b}$. Now since M is an approximation of A, $\boldsymbol{z}_i$ is an approximation of the error. Thus solving the easier system leads to a better approximation of the solution: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{z}_i$. The basic iteration introduced here, now leads to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + M^{-1}(\boldsymbol{b} - A\boldsymbol{x}_i),$$

where M is called the *preconditioner*, which is used to improve the spectral properties of the system. Note that M^{-1} is never computed explicitly.

If we now take $M = I$, we obtain the well known Richardson iteration (Van der Vorst, 2003)

$$\boldsymbol{x}_{i+1} = \boldsymbol{b} + (I - A)\boldsymbol{x}_i = \boldsymbol{x}_i + \boldsymbol{r}_i,$$

with $\boldsymbol{r}_i = \boldsymbol{b} - A\boldsymbol{x}_i$ the residual at step i. Multiplying the above relation by $-A$ and adding $\boldsymbol{b}$ gives

$$\boldsymbol{r}_{i+1} = (I - A)\boldsymbol{r}_i = (I - A)^{i+1}\boldsymbol{r}_0.$$

It then follows that the approximate solution $\boldsymbol{x}_{i+1}$ may be written as

$$\boldsymbol{x}_{i+1} = \boldsymbol{r}_0 + \boldsymbol{r}_1 + \ldots + \boldsymbol{r}_i = \sum_{k=0}^{i}(I - A)^k\boldsymbol{r}_0 \tag{2.1}$$

for $\boldsymbol{x}_0 = 0$. We can do this without loss of generality, because in case $\boldsymbol{x}_0$ is nonzero, we could just shift the system by setting $A\boldsymbol{y} = \boldsymbol{b} - A\boldsymbol{x}_0 = \hat{\boldsymbol{b}}$ with $\boldsymbol{y}_0 = 0$. We now observe that

$$\boldsymbol{x}_{i+1} \in \operatorname{Span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^i\boldsymbol{r}_0\} \equiv \mathcal{K}_{i+1}(A; \boldsymbol{r}_0).$$

The space of dimension m, spanned by a given vector $\boldsymbol{v}$, and increasing powers of A applied to $\boldsymbol{v}$ up to the $(m-1)$th power of A is called the m-dimensional *Krylov subspace* generated by A and $\boldsymbol{v}$, and is denoted as $\mathcal{K}_m(A; \boldsymbol{v})$ (Van der Vorst, 2003). In the general case that $M \neq I$, we obtain $\boldsymbol{x}_m \in \mathcal{K}_m(AM^{-1}; \boldsymbol{r}_0)$.

As we know from the power method, repeated multiplication by a matrix A converges to the eigenvector belonging to the largest eigenvalue of A. So (2.1) would converge to the eigenvector belonging to the largest eigenvalue of $I - A$. This means that this iteration without any modification is not a very good way to generate new vectors for the Krylov subspace.

Instead, one can orthogonalize every new vector with respect to all vectors that are already present in the Krylov subspace. This can be done by using for instance modified Gram-Schmidt (Golub and Van Loan, 1996). The power method with inclusion of the modified Gram-Schmidt procedure, which is given in Algorithm 2, is called the Arnoldi method. This method can generate an orthonormal basis of the Krylov subspace, but can also be used for the computation of multiple eigenvectors and eigenvalues of a matrix A.

With the orthonormal basis of the Krylov subspace V_m and the upper Hessenberg matrix $H_{m+1,m}$ that are obtained from the Arnoldi method, we get the relation

$$AV_m = V_{m+1}H_{m+1,m},$$

or alternatively

$$V_m^T AV_m = H_{m,m}.$$

The matrix $H_{m,m}$ is also called the *Galerkin projection* of A onto $\mathcal{K}_m(A; \boldsymbol{v}_1)$. For an iterative method based on this projection, the related *Galerkin condition* is given by

$$V_m^T(\boldsymbol{r}_0 - A\boldsymbol{x}_m) = V_m^T(\boldsymbol{r}_0 - AV_m\boldsymbol{y}_m) = 0$$

```
input:   $A$                                   Matrix.
         $\boldsymbol{v}_0$                    Initial vector.
         $m$                                   Size of the Krylov subspace.
output:  $H_{m+1,m}$                           Upper Hessenberg matrix.
         $V_m = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m]$   Orthonormal basis of the
                                               $m$-dimensional Krylov subspace.

$\boldsymbol{v}_1 = \boldsymbol{v}_0 / \|\boldsymbol{v}_0\|_2$
for $j = 1, \ldots, m-1$ do
    $\boldsymbol{q} = A\boldsymbol{v}_j$
    for $i = 1, \ldots, j$ do
        $H_{i,j} = \boldsymbol{v}_i^T \boldsymbol{q}$
        $\boldsymbol{q} = \boldsymbol{q} - H_{i,j}\boldsymbol{v}_i$
    $H_{j+1,j} = \|q\|_2$
    $\boldsymbol{v}_{j+1} = \boldsymbol{q} / H_{j+1,j}$
```

Algorithm 2: The Arnoldi method with modified Gram-Schmidt. V is an orthonormal basis of the m-dimensional Krylov subspace. The eigenvalues and eigenvectors of A can be obtained from the matrices $H_{m+1,m}$ and $V_m = [v_1, \ldots, v_m]$.

for some $\boldsymbol{x}_m = V_m \boldsymbol{y}_m + \boldsymbol{x}_0$ (Golub and Van Loan, 1996).

Most Krylov subspace methods can be distinguished in three different classes (Van der Vorst, 2003).

1. $\boldsymbol{x}_m$ is taken such that the residual is orthogonal to the Krylov subspace: $\boldsymbol{b} - A\boldsymbol{x}_m \perp \mathcal{K}_m(A; \boldsymbol{r}_0)$. This is called the *Ritz–Galerkin* approach.

2. $\boldsymbol{x}_m$ is taken such that the residual is orthogonal to some other suitable m-dimensional space. This is called the *Petrov–Galerkin* approach.

3. $\boldsymbol{x}_m$ is taken such that $\|\boldsymbol{b} - A\boldsymbol{x}_m\|_2$ is minimal over $\mathcal{K}_m(A; \boldsymbol{r}_0)$. This is called the *minimum residual norm* approach.

The most commonly used iterative methods are the Conjugate Gradient (CG) method for symmetric positive definite matrices, which is based on the Ritz–Galerkin approach, and the MINRES method for symmetric matrices and GMRES method for general matrices, which are based on the minimum residual norm approach.

The convergence behavior of Krylov subspace methods is determined by the spectral properties of A. Fast convergence is obtained when all eigenvalues of the matrix are close to 1, and when the matrix is close to a normal matrix, meaning that $A^T A = AA^T$. As mentioned above, a preconditioner can be used to improve these properties. In Chapter 3 we introduce such a preconditioner for staggered grid discretizations of the Navier–Stokes equations.

Bifurcation analysis 2.3

As mentioned earlier, we are interested in the computation of tipping points in the MOC. In this section we discuss how we compute these tipping or bifurcation points.

Take a dynamical system of the form

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = F(\boldsymbol{x}, \boldsymbol{p}), \tag{2.2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state, $\boldsymbol{p} \in \mathbb{R}^m$ are parameter values and $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. We are interested in sets of solutions which are approached asymptotically when going forward or backward in time. These sets include periodic orbits, and stable and unstable steady states. A bifurcation is a point at which a small change of a so-called bifurcation parameter in $\boldsymbol{p}$ results in a sudden qualitative change of the system, e.g. when a stable steady state turns into an unstable steady state.

We can study bifurcations from a bifurcation diagram, in which a variable of interest $y : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is plotted against the bifurcation parameter. An example of a bifurcation diagram is shown in Figure 2.1, where y is the variable of interest, and μ is the bifurcation parameter. The stable steady states are depicted by a solid line, whereas the unstable steady states are dashed. In the bifurcation diagram, we see two saddle-node bifurcations.

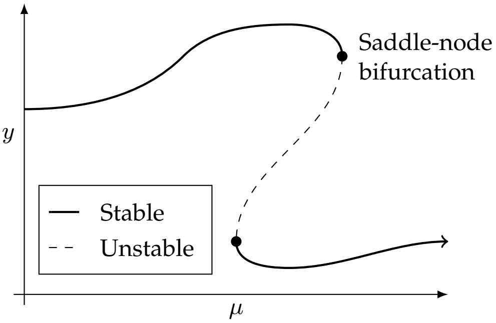


Figure 2.1: Schematic depiction of two saddle-node bifurcations, where at the first bifurcation a branch of stable steady states turns into a branch of unstable steady states, and at the second bifurcation an unstable branch turns into a stable branch. Here y is the variable of interest, and μ is the value of the bifurcation parameter.

The types of bifurcations that we will see in this thesis are saddle-node bifurcations, as shown in Figure 2.1, and pitchfork bifurcations, as shown in Figure 2.2.

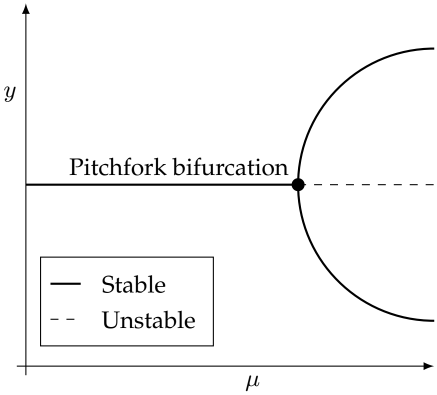


Figure 2.2: Schematic depiction of a supercritical pitchfork bifurcation, where a stable steady state branches into two stable steady states and one unstable steady state. Here y is the variable of interest, and μ is the value of the bifurcation parameter.

2.3.1 Pseudo-arclength continuation

For the study of bifurcation points in a dynamical system of the form (2.2), it is often desirable to compute steady states for a wide variety of parameter values $\boldsymbol{p}$. A common approach is to start from a known steady state at a nearby parameter value, and then perform a time integration to get to the stable steady state at the current parameter value.

A more efficient way of finding the steady state at a certain parameter value when starting from the steady state at a nearby parameter value is by computing the solution of

$$F(\boldsymbol{x}, \boldsymbol{p}) = 0 \tag{2.3}$$

directly by using Newton's method as described in Section 2.1. This is called natural continuation. Neither of these methods, however, is capable of the computation of unstable steady states after a saddle-node bifurcation is reached. The solution will just converge to one of the stable steady states instead.

Instead one can apply a method called pseudo-arclength continuation (Keller, 1977; Crisfield, 1991), which can speed up the computation of steady states considerably Dijkstra et al. (2014). In this method, the branches

$(\boldsymbol{x}(s), \boldsymbol{p}(s))$ in the bifurcation diagram are instead parameterized by an arclength parameter s. An additional normalizing condition

$$N(\boldsymbol{x}, \mu, s) \equiv \zeta \, \|\dot{\boldsymbol{x}(s)}\|_2^2 + (1-\zeta) \, |\dot{\mu}(s)|^2 - 1 = 0$$

is applied, where the dot indicates the derivative with respect to s, $\zeta \in (0, 1)$ is a tunable parameter and μ is the bifurcation parameter in $\boldsymbol{p}$. This condition, however, is not very useful for both implementation or derivation of the method that we want to implement. Instead we reside to the approximation

$$N_2(\boldsymbol{x}, \mu, s) \equiv \zeta \, \|\boldsymbol{x}(s) - \boldsymbol{x}_0\|_2^2 + (1-\zeta) \, (\mu(s) - \mu_0)^2 - (s - s_0)^2 = 0. \tag{2.4}$$

Here $\Delta s = s - s_0$ is the step along the tangent vector at the current position $(\boldsymbol{x}_0, \boldsymbol{p}_0)$. In case $\dot{\boldsymbol{x}}_0$ and $\dot{\mu}_0$ are known, one can instead use the approximation

$$N_3(\boldsymbol{x}, \mu, s) \equiv \zeta \, \dot{\boldsymbol{x}}_0^T (\boldsymbol{x}(s) - \boldsymbol{x}_0) + (1-\zeta) \, \dot{\mu}_0 (\mu(s) - \mu_0) - (s - s_0) = 0.$$

To solve (2.3) together with (2.4), a predictor-corrector scheme is applied. The Euler predictor, which is a guess for the next steady state $\boldsymbol{x}_{i+1}$, is given by

$$\begin{aligned} \boldsymbol{x}_{i+1}^{(0)} &= \boldsymbol{x}_i + \Delta s \, \dot{\boldsymbol{x}}_i, \\ \mu_{i+1}^{(0)} &= \mu_i + \Delta s \, \dot{\mu}_i. \end{aligned}$$

In the Newton corrector, the updates are given by

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \Delta \boldsymbol{x}^{(k+1)}, \\ \mu^{(k+1)} &= \mu^{(k)} + \Delta \mu^{(k+1)}, \end{aligned}$$

where we dropped the $i+1$ subscript for readability. This process is shown in Figure 2.3.

$\Delta \boldsymbol{x}^{(k+1)}$ and $\Delta \mu^{(k+1)}$ can be obtained from the correction equation

$$\begin{pmatrix} J_{\boldsymbol{x}}(\boldsymbol{x}^{(k)}, \mu^{(k)}) & J_\mu(\boldsymbol{x}^{(k)}, \mu^{(k)}) \\ 2\zeta \, (\boldsymbol{x}^{(k)} - \boldsymbol{x}_i)^T & 2(1-\zeta) \, (\mu^{(k)} - \mu_i) \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{x}^{(k+1)} \\ \Delta \mu^{(k+1)} \end{pmatrix} = \begin{pmatrix} -F(\boldsymbol{x}^{(k)}, \mu^{(k)}) \\ -N_2(\boldsymbol{x}^{(k)}, \mu^{(k)}, s) \end{pmatrix},$$

where $J_{\boldsymbol{x}}$ is the Jacobian of F with respect to $\boldsymbol{x}$ and J_μ is the Jacobian of F with respect to μ. This system can be solved as such, which is very robust, but has the downside that a solver for this bordered system has to be implemented.

The solution can also be obtained by first solving

$$\begin{aligned} J_{\boldsymbol{x}}(\boldsymbol{x}^{(k)}, \mu^{(k)}) \boldsymbol{z}_1 &= -F(\boldsymbol{x}^{(k)}, \mu^{(k)}), \\ J_{\boldsymbol{x}}(\boldsymbol{x}^{(k)}, \mu^{(k)}) \boldsymbol{z}_2 &= J_\mu(\boldsymbol{x}^{(k)}, \mu^{(k)}), \end{aligned}$$

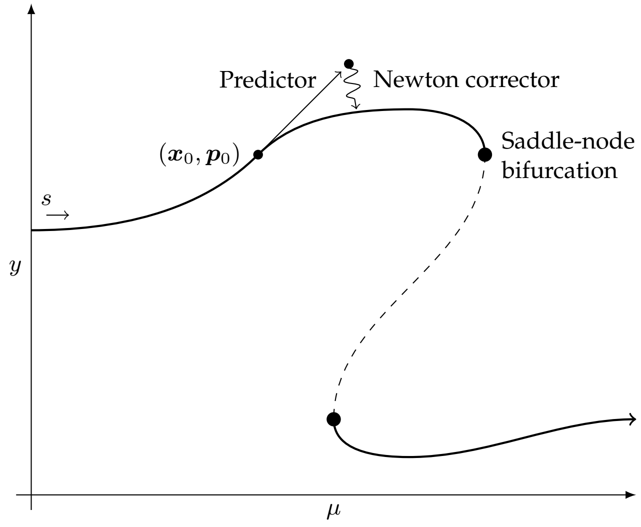


Figure 2.3: Schematic depiction of pseudo-arclength continuation.

after which $\Delta\boldsymbol{x}^{(k+1)}$ and $\Delta\mu^{(k+1)}$ can be found from

$$\begin{aligned}\Delta\mu^{(k+1)} &= \frac{-N_2(\boldsymbol{x}^{(k)}, \mu^{(k)}, s) - 2\zeta\,(\boldsymbol{x}^{(k)} - \boldsymbol{x}_i)^T\boldsymbol{z}_1}{2(1-\zeta)\,(\mu^{(k)} - \mu_i) - 2\zeta\,(\boldsymbol{x}^{(k)} - \boldsymbol{x}_i)^T\boldsymbol{z}_2},\\ \Delta\boldsymbol{x}^{(k+1)} &= \boldsymbol{z}_1 - \Delta\mu^{(k+1)}\boldsymbol{z}_2.\end{aligned}$$

This method is less robust, but has the advantage that only systems with $J_{\boldsymbol{x}}$ have to be solved.

Linear approximations to the derivatives $\dot{\boldsymbol{x}}$ and $\dot{\mu}$, which are required for the Euler predictor, can be obtained from two previous iterations of the method as

$$\begin{aligned}\dot{\boldsymbol{x}}_i &\approx \frac{1}{\Delta s}(\boldsymbol{x}_i - \boldsymbol{x}_{i-1}),\\ \dot{\mu}_i &\approx \frac{1}{\Delta s}(\mu_i - \mu_{i-1}).\end{aligned}$$

This, however, can not be used in the first step. What we do instead is first obtain $\boldsymbol{x}_0$ at parameter μ_0 by using Newton's method, and then compute the tangent by solving

$$J_{\boldsymbol{x}}(\boldsymbol{x}_0, \mu_0)\boldsymbol{z} = J_\mu(\boldsymbol{x}_0, \mu_0)$$

for $\boldsymbol{z}$, after which

$$\dot{\boldsymbol{x}}_0 \approx -\frac{1}{\sqrt{\boldsymbol{z}^T\boldsymbol{z} + \Delta\mu^2}}\boldsymbol{z},$$
$$\dot{\mu}_0 \approx \frac{\Delta\mu}{\sqrt{\boldsymbol{z}^T\boldsymbol{z} + \Delta\mu^2}},$$

where we take $\Delta\mu = 1$.

Stochastic differential equations 2.4

Stochastic differential equations (SDEs) are used in a wide variety of applications areas including biology, chemistry, physics, economics, and of course climate dynamics. In this thesis, we will use them to represent unresolved variability in the freshwater forcing. We only briefly introduce the concepts that are required for understanding the usage of SDEs in this thesis. For a more detailed explanation, we refer to Higham (2001), Gardiner (1985) or Dijkstra (2013).

Brownian motion 2.4.1

A *standard Brownian motion* or *Wiener process* is a random variable W_t on $t \in [0, T]$ which has the following properties:

1. $W_0 = 0$ with probability 1.
2. W has independent increments $W_t - W_s$ for every $t > s \geq 0$.
3. For $0 \leq s < t \leq T$, the increment $W_t - W_s$ is normally distributed with mean zero and variance $t - s$, i.e. $W_t - W_s \sim N(0, t - s)$
4. W_t is continuous in t with probability 1.

In a computational context, we always look at discretized Brownian motion. This means that the value of W_t is only specified at discrete values of t. Say we take a fixed number of time steps N with length $\Delta t = T/N$, then the discretized Brownian motion would be defined as

$$W_i = W_{i-1} + \Delta W_i,$$

where W_i is the value of W_t at time $t_i = i\Delta t$, $i = 0, 1, \ldots, N$ and ΔW_i is the increment at time t_i which is an independent random variable taken from $N(0, \Delta t)$. An example of a discretized Brownian motion can be found in Figure 2.4.

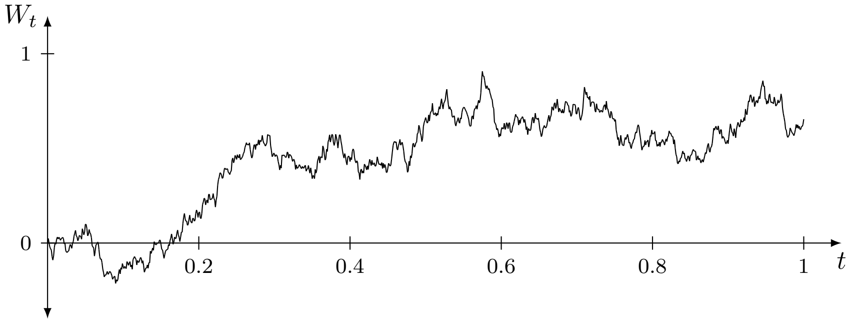


Figure 2.4: Example of a single realization of a one-dimensional discretized Brownian motion.

2.4.2 Stochastic differential equations

Given a real function $f : [0, T] \to \mathbb{R}$, we approximate the integral $\int_0^T f(t)\mathrm{d}t$ by the left Riemann sum

$$\sum_{i=0}^{N-1} f(t_i)(t_{i+1} - t_i), \tag{2.5}$$

where $t_i = i\Delta t$ with $\Delta t = \frac{T}{N}$. For Riemann-integrable functions, (2.5) will converge to the actual value of the integral for $\Delta t \to 0$.

Similarly, we may consider an approximation of the form

$$\sum_{i=0}^{N-1} g(t_i)(W_{i+1} - W_i),$$

for a stochastic integral $\int_0^T g(t)\mathrm{d}W_t$. If we again take $\Delta t \to 0$, this defines the value of the so-called *Itô integral*.

For stochastic integration, one may also use different approximations to obtain a different integral value. After the Itô integral, the most common integral is the Stratonovich integral, which instead of using a left Riemann-type sum uses the midpoint rule. In this thesis, however, we will only use Itô integration and the related Itô calculus.

Using the Itô integral, we may define a stochastic differential equation in integral form

$$X_t = X_0 + \int_0^t f(X_s)\mathrm{d}s + \int_0^t g(X_s)\mathrm{d}W_s, \quad 0 \le t \le T,$$

where the initial condition X_0 is a random variable, and f and g are scalar functions. In differential equation form, this integral is given by

$$\mathrm{d}X_t = f(X_t)\mathrm{d}t + g(X_t)\mathrm{d}W_t, \quad 0 \le t \le T.$$

The ocean can, unfortunately, not be described by a single stochastic differential equation. We will instead look at systems of stochastic differential algebraic equations (SDAEs) of the form

$$M\,\mathrm{d}\boldsymbol{X}_t = F(\boldsymbol{X}_t)\,\mathrm{d}t + g(\boldsymbol{X}_t)\,\mathrm{d}\boldsymbol{W}_t, \tag{2.6}$$

where $M \in \mathbb{R}^{n\times n}$ is usually referred to as the mass matrix, $\boldsymbol{X}_t \in \mathbb{R}^n$, $F : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n_w}$, $\boldsymbol{W}_t \in \mathbb{R}^{n_w}$, and n_w is the number of stochastic variables.

The Euler–Maruyama method 2.4.3

The most basic method for simulating stochastic differential equations in Itô calculus is the Euler–Maruyama method, which is a generalization of the Euler method for ordinary differential equations Higham (2000). Consider the n-dimensional SDE

$$\mathrm{d}\boldsymbol{X}_t = F(\boldsymbol{X}_t)\,\mathrm{d}t + g(\boldsymbol{X}_t)\,\mathrm{d}\boldsymbol{W}_t.$$

The Euler–Maruyama method is then given by

$$\boldsymbol{X}_{i+1} = \boldsymbol{X}_i + \Delta t F(\boldsymbol{X}_i) + g(\boldsymbol{X}_i)\,\Delta\boldsymbol{W}_i. \tag{2.7}$$

where $\Delta\boldsymbol{W}_i$ are independent and identically distributed random variables with mean zero and variance Δt.

The stochastic theta method 2.4.4

In the same sense that the Euler method can be generalized to the theta method for ordinary differential equations, we can also generalize the Euler–Maruyama method (Kloeden and Platen, 1992). The generalization of this method is given by

$$\boldsymbol{X}_{i+1} = \boldsymbol{X}_i + (1-\theta)\Delta t\, F(\boldsymbol{X}_i) + \theta\Delta t\, F(\boldsymbol{X}_{i+1}) + g(\boldsymbol{X}_i)\,\Delta\boldsymbol{W}_i.$$

We will refer to this method as the stochastic theta method, as is also done in Higham (2000). Note that if we take $\theta = 0$, we get the Euler–Maruyama scheme as described above, with $\theta = 1$, we get the stochastic version of the implicit Euler method, and $\theta = \frac{1}{2}$ gives a stochastic trapezoidal rule. In any case, the noise is always added explicitly.

This method can be generalized even further for SDAEs, in which case it is given by

$$M(\boldsymbol{X}_i - \boldsymbol{X}_{i+1}) + (1-\theta)\Delta t\, F(\boldsymbol{X}_i) + \theta\Delta t\, F(\boldsymbol{X}_{i+1}) + g(\boldsymbol{X}_i)\,\Delta\boldsymbol{W}_i = 0. \tag{2.8}$$

Here $\boldsymbol{X}_{i+1}$ can be obtained by applying the Newton method. The Jacobian of (2.8) is given by

$$\theta\Delta t J(\boldsymbol{X}_{i+1}) - M,$$

where J is the Jacobian of F.

2.5 Governing equations

2.5.1 The Navier–Stokes equations

The Navier–Stokes equations, which govern the motion of a fluid such as the ocean, are described by the conservation of mass

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0,$$

and the conservation of momentum

$$\rho \frac{\partial \boldsymbol{u}}{\partial t} + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \nabla \cdot \boldsymbol{\tau},$$

where $\boldsymbol{u}$ is the velocity, p the pressure, ρ the density and $\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\tau}$ is the internal stress tensor (Batchelor, 2000; Landau and Lifshitz, 1959).

In case the fluid is assumed to be incompressible, i.e. the density is constant, the conservation of mass is reduced to

$$\nabla \cdot \boldsymbol{u} = 0.$$

Furthermore, the stress tensor can be simplified and is now given by

$$\boldsymbol{\tau} = \mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T),$$

where μ is the dynamic viscosity. In this case the conservation of momentum can be written as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \boldsymbol{u},$$

where $\nu = \mu/\rho$ is the kinematic viscosity.

2.5.2 The ocean model

Due to the complex nature of the Navier–Stokes equations, many approximations are made to be able to focus on the most important motions that are present in the ocean (Griffies, 2004; Cushman-Roisin and Beckers, 2011). The most important approximations are as follows. First we have the hydrostatic approximation, where the contributions of vertical acceleration and friction to the vertical pressure gradients are neglected. Then we have the shallow ocean approximation, where the depth of the ocean is assumed to be negligible with respect to the radius of the earth. The third approximation is the Boussinesq approximation, which means that any density difference that is not multiplied by the gravitational constant g is ignored. And finally we have the rigid lid approximation which assumes that the sea surface elevation is static.

For our simulations, this results in the spatially quasi two-dimensional primitive equation model of the Atlantic MOC as described in Weijer and Dijkstra (2001); Den Toom et al. (2011). 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

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

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 2.1.

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}	$\mathrm{m^2s^{-1}}$	α_T	$=$	1.0	$\cdot$	10^{-4}	$\mathrm{K^{-1}}$
A_V	$=$	1.0	$\cdot$	10^{-3}	$\mathrm{m^2s^{-1}}$	α_S	$=$	7.6	$\cdot$	10^{-4}	$\mathrm{psu^{-1}}$
K_H	$=$	1.0	$\cdot$	10^3	$\mathrm{m^2s^{-1}}$	ρ_0	$=$	1.0	$\cdot$	10^3	$\mathrm{kg\,m^{-3}}$
K_V	$=$	1.0	$\cdot$	10^{-4}	$\mathrm{m^2s^{-1}}$	τ	$=$	75.0			days

Table 2.1: Fixed model parameters of the two-dimensional ocean model.

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 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

$$\begin{aligned}
-\frac{1}{\rho_0 r_0}\frac{\partial p}{\partial \theta} + A_V \frac{\partial^2 v}{\partial z^2} + \frac{A_H}{r_0^2}\left(\frac{1}{\cos\theta}\frac{\partial}{\partial\theta}\left(\cos\theta\frac{\partial v}{\partial\theta}\right) + \left(1-\tan^2\theta\right)v\right) &= 0,\\
-\frac{1}{\rho_0}\frac{\partial p}{\partial z} + g\left(\alpha_T T - \alpha_S S\right) &= 0,\\
\frac{1}{r_0\cos\theta}\frac{\partial v\cos\theta}{\partial\theta} + \frac{\partial w}{\partial z} &= 0,\\
\frac{\partial T}{\partial t} + \frac{v}{r_0}\frac{\partial T}{\partial\theta} + w\frac{\partial T}{\partial z} = \frac{K_H}{r_0^2\cos\theta}\frac{\partial}{\partial\theta}\left(\cos\theta\frac{\partial T}{\partial\theta}\right) &+ K_V\frac{\partial^2 T}{\partial z^2} + \mathrm{CA}(T),\\
\frac{\partial S}{\partial t} + \frac{v}{r_0}\frac{\partial S}{\partial\theta} + w\frac{\partial S}{\partial z} = \frac{K_H}{r_0^2\cos\theta}\frac{\partial}{\partial\theta}\left(\cos\theta\frac{\partial S}{\partial\theta}\right) &+ K_V\frac{\partial^2 S}{\partial z^2} + \mathrm{CA}(S).
\end{aligned}$$

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 $\theta = \theta_N = 60^\circ$ 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,

$$\begin{aligned} K_V \frac{\partial T}{\partial z} &= \frac{H_m}{\tau}\left(\bar{T}(\theta) - T\right), \\ K_V \frac{\partial S}{\partial z} &= S_0 F_s(\theta). \end{aligned} \tag{2.9}$$

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 Den Toom et al. (2011).

2.5.3 Stochastic freshwater forcing

For the most simple deterministic case, we choose an equatorially symmetric surface forcing as

$$\bar{T}(\theta) = 10.0 \cos\left(\pi\theta/\theta_N\right), \tag{2.10a}$$

$$F_s(\theta) = \bar{F}_s(\theta) = \mu F_0 \frac{\cos(\pi\theta/\theta_N)}{\cos(\theta)}, \tag{2.10b}$$

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

To represent the unresolved variability in the freshwater forcing, we use a stochastic forcing. This forcing is chosen as

$$F_s(\theta, t) = (1 + \sigma\, \zeta(\theta, t))\bar{F}_s(\theta), \tag{2.11}$$

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 $\sigma = 0.1$. The noise is additive and is only active in the freshwater component, only present at the surface, meridionally uncorrelated, and has magnitude σ of 10% of the background freshwater forcing amplitude at each latitude θ. In the context of (2.6), g is given by the discretized version of $\sigma\bar{F}_s(\theta)$.

LINEAR SYSTEMS

During the application of Newton's method in both continuation processes as described in Section 2.3 and implicit time stepping as described in Section 2.4.4, linear systems have to be solved. Ultimately, our goal is to solve linear systems with the Jacobian of the model as described in Section 2.5.2, but for now, we will just focus on solving the incompressible Navier–Stokes without the additional tracers. The most elegant way to solve these linear systems is to replace the complete LU factorization by an accurate incomplete one, which can be used as a preconditioner in an iterative method. Moreover, this preconditioner can be used to find approximate eigenvalues and eigenvectors of the Jacobian matrix during the continuation process. By an appropriate ordering technique and dropping procedure, one can construct an incomplete LU (ILU) factorization that yields grid independent convergence behavior and approaches an exact factorization as the amount of allowed fill is increased.

The incompressible Navier–Stokes equations can be written as

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} &= -\nabla p + \frac{1}{\mathrm{Re}} \Delta \boldsymbol{u}, \\ \nabla \cdot \boldsymbol{u} &= 0, \end{aligned} \tag{3.1}$$

where $\mathrm{Re} = \frac{\rho U}{\mu}$ is the Reynolds number, ρ is the density and μ is the dynamic viscosity. These equations are discretized using a second-order symmetry-preserving finite volume method on an Arakawa C-grid; see Figure 3.1. The discretization leads to a system of ordinary differential equations (ODEs)

$$\begin{aligned} M \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} + N(\boldsymbol{u}, \boldsymbol{u}) &= -G\boldsymbol{p} + \frac{1}{\mathrm{Re}} L\boldsymbol{u} + \boldsymbol{f}_{\boldsymbol{u}}, \\ -G^T \boldsymbol{u} &= \boldsymbol{f}_p, \end{aligned}$$

where $\boldsymbol{u}$ and $\boldsymbol{p}$ now represent the velocity and pressure in each grid point, $N(\cdot,\cdot)$ is the bilinear form arising from the convective terms, L is the discretized Laplace operator, G is the discretized gradient operator, M is the mass matrix, which contains the volumes of the grid cells on its diagonal and f contains the known part of the boundary conditions. Note that minus the transpose of the gradient operator gives us the divergence operator.

If we fix one of the variables in the bilinear form N, it becomes linear, hence we may write

$$N(\boldsymbol{u},\boldsymbol{v}) = N_1(\boldsymbol{u})\boldsymbol{v} = N_2(\boldsymbol{v})\boldsymbol{u},$$

where we assume that $N_1(\boldsymbol{u})$ contains the discretized convection terms, and $N_2(\boldsymbol{u})$ contains the cross terms. Using this notation, the linear system of saddle point type (Benzi et al., 2005) that has to be solved with Newton's method in a continuation is given by

$$\begin{pmatrix} N_1(\boldsymbol{u}) + N_2(\boldsymbol{u}) - \frac{1}{\mathrm{Re}}L & G \\ G^T & 0 \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{u} \\ \Delta\boldsymbol{p} \end{pmatrix} = - \begin{pmatrix} \boldsymbol{f_u} \\ \boldsymbol{f}_p \end{pmatrix}. \tag{3.2}$$

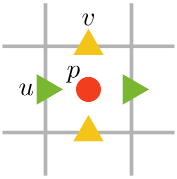


Figure 3.1: Positioning of unknowns in an Arakawa C-grid

It is known (Verstappen and Veldman, 2003), that, on closed domains, dissipation of kinetic energy only occurs by diffusion. Our discretization preserves this property, which has the consequence that for divergence-free $\boldsymbol{u}$, the operator $N_1(\boldsymbol{u})$ is skew-symmetric. This means that if we omit $N_2(\boldsymbol{u})$, the approximate Jacobian will become negative definite on the space of discrete divergence-free velocities. Omitting $N_2(\boldsymbol{u})$ greatly simplifies the solution process since definiteness is a condition under which many standard iterative methods converge. This is a simplification that many authors make, and results in replacing the Newton iteration by a Picard iteration (Elman et al., 2014). This works well for reasonably low Reynolds numbers, and far away from bifurcation points where steady solutions become unstable. The Picard iteration, however, seriously impairs the convergence rate of the nonlinear iteration (Carey and Krishnan, 1987; Baars et al., 2019b).

Similarly some authors use time dependent approaches to study the stability of steady states (Dijkstra et al., 2014). This approach, however, also requires some tricks to obtain the desired information.

Since we want to study precisely the phenomena where the above methods experience difficulty, we would rather use the full Jacobian matrix of the nonlinear equations and apply Newton's method directly.

There are many methods that are based on segregation of physical variables which can solve the linear equations that arise in every Newton iteration. In this approach the system is split into subproblems of an easier form for which standard methods exist. The segregation can already be done at the discretization level, e.g. by doing a time integration and solving a pressure correction equation independently of the momentum equations (Verstappen and Veldman, 2003; Verstappen and Dröge, 2005). Another class of methods performs the segregation during the linear system solve, often in a preconditioning step. Important are physics based preconditioners (De Niet et al., 2007; Klaij and Vuik, 2012; Cyr et al., 2012; He et al., 2018), which try to split the problem in subsystems which capture the bulk of the physics. The subsystems are again solved by iterative procedures, e.g. algebraic multigrid (AMG) for Poisson-like equations. These methods consist of nested loops for: the nonlinear iteration, iterations over the coupled system, and iterations over the subsystems. The stopping criteria of all these different iterations have to be tuned to make the solver efficient. Furthermore, the total number of iterations in the innermost loop is given by the product of the number of iterations performed on all three levels of iteration and thus easily becomes excessive. This is a major problem when trying to achieve extreme parallelism, because each innermost iteration typically requires global communication in inner products. The number of levels of nested iteration may increase even more if additional physical phenomena are added (De Niet et al., 2007; Thies et al., 2009). Our ultimate aim is to get rid of the inner iterations altogether and to solve the nonlinear equations using a subspace accelerated inexact Newton method. In Sleijpen and Wubs (2004) we did this for simple eigenvalue problems using the Jacobi–Davidson method, which is in fact an accelerated Newton method. The method we present here will make this feasible for the 3D Navier–Stokes equations.

For this, fully aggregated approaches are important. In this category, multigrid and multilevel ILU methods for systems of PDEs exist (see Trottenberg et al. (2000); Wathen (2015) and references therein). The former is attractive but for those methods smoothers may fail due to a loss of diagonal dominance for higher Reynolds Numbers, for which a common solution is to use time integration (Feldman and Gelfgat, 2010). The latter comprise AMG algorithms and multilevel methods like MRILU (Botta and Wubs, 1999) and ILUPACK (Bollhöfer and Saad, 2006). ILUPACK has been successful in many fields since it uses a bound to preclude very unstable factorizations. However, this method does not show grid independence for Navier–Stokes problems and is difficult to parallelize (Aliaga et al., 2008). It works well for large problems, but not yet beyond a single shared memory system.

In this chapter we present a novel multilevel preconditioning method which is specialized for the 3D Navier–Stokes equations. In Section 3.1, we first describe the two-level ILU preconditioner as introduced in Wubs and Thies (2011) and Thies and Wubs (2011). After this, we generalize the two-level method to a multilevel method in Section 3.2. To make this method work for the 3D Navier–Stokes equations, we introduce a skew partitioning method in Section 3.3. In Section 3.4 we then discuss the scalability and general performance of the method, and compare it to existing methods, after which we finalize by providing a summary and discussion in Section 3.5.

3.1 The two-level ILU preconditioner

In Wubs and Thies (2011) a robust parallel two-level method was developed for solving

$$A\boldsymbol{x} = \boldsymbol{b},$$

with $A \in \mathbb{R}^{n\times n}$ for a class of matrices known as $\mathcal{F}$-matrices. An $\mathcal{F}$-matrix is a matrix of the form

$$A = \begin{pmatrix} K & B \\ B^T & 0 \end{pmatrix},$$

with K symmetric positive definite and B such that it has at most two entries per row and the entries in each row sum up to 0, as is the case for our gradient matrix G (Tuma, 2002; De Niet and Wubs, 2008). It was shown that the two-level preconditioner leads to grid-independent convergence for the Stokes equations on an Arakawa C-grid, and that the method is robust even for the Navier–Stokes equations, which strictly speaking do not yield $\mathcal{F}$-matrices because K becomes nonsymmetric and possibly indefinite.

Rather than reviewing the method and theory in detail, we will only briefly present it here. For details, see Wubs and Thies (2011) and Thies and Wubs (2011).

To simplify the discussion, we focus on the special case of the 3D incompressible Navier–Stokes equations in a cube-shaped domain, discretized on an Arakawa C-grid (see Figure 3.1). We refer to the velocity variables, which are located on the cell faces as V-nodes, and to the the pressure, which is located in the cell center, as P-node. The variables are numbered cell-by-cell, i.e. the first three variables are the $u/v/w$-velocity at the north/east/top face of the cell in the bottom south west corner of the domain, and variable four is the P-node in its center. Appropriate boundary conditions (e.g. Dirichlet conditions) are easily implemented in this situation. We remark that the algorithm (and software) can handle more general situations like rectangular domains, interior boundary cells, etc., and could be generalized to arbitrary domain shapes and partitionings.

First we describe the initialization phase of the preconditioner, which is the necessary preprocessing that has to be done only once for a series of linear systems with matrices with the same sparsity pattern. Then we describe the factorization phase, which has to be done separately for every matrix. Finally we describe the solution phase, which has to be performed for every linear system.

Initialization phase 3.1.1

First the system is partitioned into non-overlapping subdomains. These subdomains may be distributed over different processes, which allows for parallelization of the computation. The partitioning may be done based on the graph of the matrix, as implemented for instance in METIS (Karypis and Kumar, 1998), or by a geometric approach, e.g. by dividing the domain into rectangular subdomains. An example of a Cartesian partitioning of a square domain is shown in Figure 3.2.

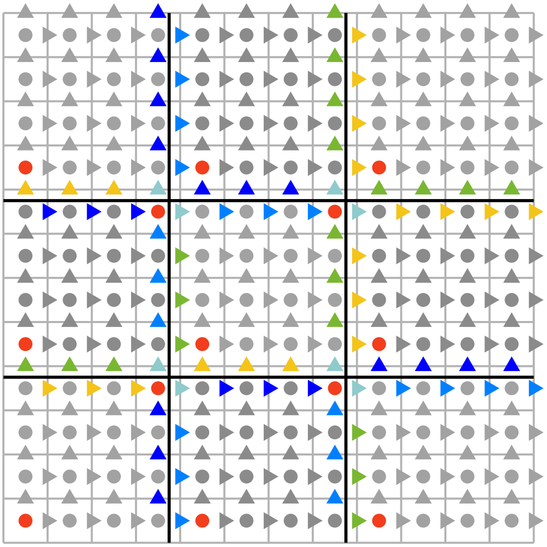


Figure 3.2: Cartesian partitioning of a 12×12 C-grid discretization of the Navier–Stokes equations into 9 subdomains of size $s_x \times s_y$ with $s_x = s_y = 4$. The interiors are shown in gray. Velocities of the same (non-gray) color that are on the same face of neighboring cells form a separator group. The red circles are pressure nodes that are retained.

In the discretization, variables in a subdomain may be coupled to variables in other subdomains. This means that the linear systems associated with the subdomains may not be solved independently. For this reason we define overlapping subdomains, which have *interior nodes* that are in the non-overlapping subdomain and are coupled only to variables in the overlapping subdomain. The interiors of all subdomains may be solved independently. The nodes in the overlapping subdomains that are not coupled only to variables in the same overlapping subdomain constitute the *separator nodes*. One *separator* is defined as a set of separator nodes that are coupled with the same set of subdomains. One *separator group* is defined as the variables on the same separator that have the same variable type. In Figure 3.2 the interior nodes are shown in gray and the different separator groups are denoted by different colors.

We can now reorder the matrix A such that the interiors (I) and separators (S) are grouped. This gives us the system

$$\begin{pmatrix} A_{II} & A_{IS} \\ A_{SI} & A_{SS} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_I \\ \boldsymbol{x}_S \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_I \\ \boldsymbol{b}_S \end{pmatrix},$$

where A_{II} consists of independent diagonal blocks. Since A_{II} consists of independent diagonal blocks, applying A_{II}^{-1} is easy and trivial to parallelize. For this reason, we aim to solve

$$\begin{aligned} S\boldsymbol{x}_S &= \boldsymbol{b}_S - A_{SI}A_{II}^{-1}\boldsymbol{b}_I, \\ \boldsymbol{x}_I &= A_{II}^{-1}\boldsymbol{b}_I - A_{II}^{-1}A_{IS}\boldsymbol{x}_S, \end{aligned}$$

where S is the Schur complement given by $S = A_{SS} - A_{SI}A_{II}^{-1}A_{IS}$.

Whether a variable is coupled to a different subdomain can be determined from the graph of the matrix. It may again also be determined geometrically by additionally defining the overlapping subdomains during the partitioning step, and checking what overlapping subdomains a node of the non-overlapping subdomain belongs to. The separators can be determined by, for every node, making a set of subdomains it is coupled to or overlapping subdomains it belongs to, and then grouping the nodes that have the same set.

There are three types of separators: faces (in 3D), edges and corners. For the Navier–Stokes problem on a C-grid, these separators only consist of V-nodes. The P-nodes are only connected to V-nodes that belong to the same overlapping subdomain, so these should never lie on a separator. We arbitrarily choose one P-node in every interior which we also define to be its own separator group to make sure the Schur complement stays nonsingular.

For every separator we now define a Householder transformation which decouples all but one V-node from the P-nodes (Wubs and Thies, 2011). This transformation is of the form

$$H_j = I - 2\frac{\boldsymbol{v}_j\boldsymbol{v}_j^T}{\boldsymbol{v}_j^T\boldsymbol{v}_j}, \tag{3.3}$$

for some separator group g_j with

$$\boldsymbol{v}_j^{(k)} = \begin{cases} \boldsymbol{e}_j^{(k)} + \|\boldsymbol{e}_j\|_2 & \text{if node } k \text{ is the first node of separator group } g_j \\ \boldsymbol{e}_j^{(k)} & \text{otherwise} \end{cases} \tag{3.4}$$

and

$$\boldsymbol{e}_j^{(k)} = \begin{cases} 1 & \text{if node } k \text{ is in separator group } g_j \\ 0 & \text{if node } k \text{ is not in separator group } g_j \end{cases}$$

for all $k = 1, \dots, n$. We call the one V-node that is still coupled to the P-nodes a V_Σ node.

The physical interpretation of this Householder transformation is that the V_Σ velocities that remain on a separator face provide an approximation of the collective flux through that face. It is therefore important that the variables are correctly scaled before the factorization in a way that they represent fluxes. In the matrix in (3.2) this gives a G-part with entries of constant magnitude, in which case the Householder transformations will exactly decouple the non-V_Σ nodes from the pressures. Since the Householder transformation can be applied independently for every separator, we may collect all these transformations in one single transformation matrix H.

Factorization phase 3.1.2

For every overlapping subdomain Ω_i, $i = 1, \dots, N$, where N is the total number of overlapping subdomains, we can define a local matrix

$$A^{(i)} = \begin{pmatrix} A_{II}^{(i)} & A_{IS}^{(i)} \\ A_{SI}^{(i)} & A_{SS}^{(i)} \end{pmatrix}.$$

After computing the factorization $A_{II}^{(i)} = L_{II}^{(i)} U_{II}^{(i)}$, the local contribution to the Schur complement is given by

$$S_i = A_{SS}^{(i)} - A_{SI}^{(i)} (L_{II}^{(i)} U_{II}^{(i)})^{-1} A_{IS}^{(i)},$$

and the global Schur complement is given by

$$S = \sum_{i=1}^{N} S_i.$$

We now apply the Householder transformation

$$S_T = H(\sum_{i=1}^{N} S_i) H^T = \sum_{i=1}^{N} H_i S_i H_i^T, \tag{3.5}$$

which can be done separately for every separator or subdomain, or on the entire Schur complement. We choose to do this separately for every subdomain, since H is very sparse, and sparse matrix-matrix products are very expensive and memory inefficient.

We now drop all connections between V_Σ and non-V_Σ nodes and between non-V_Σ nodes and non-V_Σ nodes in different separator groups. The dropping that is applied here is what makes our factorization inexact. Not applying the dropping gives us a factorization that can still be partially parallelized, but is also exact.

Our transformed Schur complement is now reduced to a block-diagonal matrix with blocks for the non-V_Σ nodes for every separator and one block for all V_Σ nodes, which we will call $S_{\Sigma\Sigma}$. Separate factorizations can again be made for all these diagonal blocks, which can again be done in parallel. For the non-V_Σ blocks, sequential LAPACK solves can be used, and for $S_{\Sigma\Sigma}$ we can employ a (distributed) sparse direct solver. We denote the factorization that is computed by these solvers by $L_S U_S$.

The full factorization obtained by the method is given by

$$A = \begin{pmatrix} L_{II} & 0 \\ A_{SI} & H^T L_S \end{pmatrix} \begin{pmatrix} U_{II} & (L_{II}U_{II})^{-1} A_{IS} \\ 0 & U_S H \end{pmatrix}.$$

3.1.3 Solution phase

After the preconditioner has been computed, it can be applied to a vector in each step of a preconditioned Krylov subspace method, for which we use the well-known GMRES method (Saad and Schultz, 1986). Communication has to occur in the application of A_{IS} and A_{SI}, and when solving the distributed V_Σ system. The latter was adressed by using a parallel sparse direct solver in Thies and Wubs (2011), but in the next section we propose a different road that does not rely on the availability of such a solver.

3.2 The multilevel ILU preconditioner

The main issue with the above two-level ILU factorization that prevents scaling to very large problem sizes in three space dimensions is that as the problem size increases and the subdomain size remains constant, the size of $S_{\Sigma\Sigma}$ will increase steadily and any (serial or parallel) sparse direct solver will eventually limit the feasible problem sizes. Increasing the subdomain size, on the other hand, leads to more iterations and therefore more synchronization points.

One of the strong points, on the other hand, is the fact that it preserves the structure of the original problem in the sense that, when applied to an $\mathcal{F}$-matrix, it produces a strongly reduced matrix $S_{\Sigma\Sigma}$ which is again an $\mathcal{F}$-matrix. It is therefore intuitive to try applying the scheme recursively to avoid the problem of the growing sparse matrix that has to be factorized. From the structure preserving properties of the algorithm, it is immediately clear that such a recursive scheme will again lead to grid-independent convergence if the number of recursions is kept constant as the grid size is increased.

From now on we will refer to the number of recursions, or the number of times a reduced matrix $S_{\Sigma\Sigma}$ is computed, as the number of levels. Note that this means that the method, which was previously referred to the two-level method is in fact the first level of the multilevel method. Applying a direct solver to S_T from (3.5) would be level zero, since in this case the preconditioner itself is in fact just a direct solver.

In order to apply the method to the reduced matrix $S_{\Sigma\Sigma}$, we need a coarser partitioning for which it is most optimal in terms of communication to make sure subdomains are not split up in the new partitioning. In case we have a regular partitioning like a rectangular partitioning this may be done by multiplying the *separator length* by a certain *coarsening factor*. Having a coarsening factor of 2 would mean that in 3D the separator length is increased by a factor 2, and the number of subdomains is reduced by a factor 8.

As stated in the previous section, we require the velocity variables to be correctly scaled to be able to apply the Householder transformation. However, the V_Σ-variables from the previous level that lie on one separator had a different number of variables in their separator groups. In case of a regular partitioning, an edge separator, for instance, consists of V_Σ-nodes from two edges and one corner from the previous level This leads to a different scaling of the V_Σ-nodes and thus to non-constant entries in the G-part of $S_{\Sigma\Sigma}$. Instead of re-scaling the $S_{\Sigma\Sigma}$ matrix on every level, we instead use a test vector $\boldsymbol{t}$. The test vector is defined initially as a constant vector of ones, and is multiplied by the Householder transformation H at each consecutive level. The Householder transformation is as defined in (3.3) and (3.4), but now with

$$\boldsymbol{e}_j^{(k)} = \begin{cases} \boldsymbol{t}^{(k)} & \text{if node } k \text{ is in separator group } g_j \\ 0 & \text{if node } k \text{ is not in separator group } g_j \end{cases}$$

for all $k = 1, \ldots, n$. After applying the Householder transformation, we can again apply dropping to remove connections between V_Σ and non-V_Σ nodes and between non-V_Σ nodes and non-V_Σ nodes in different separator groups. When the matrix $S_{\Sigma\Sigma}$ is sufficiently small, a direct solver is applied to factorize it.

Instead of just having one separator group per variable per separator, we may also choose to have multiple separator groups, meaning that instead of retaining only one V_Σ node per variable per separator we retain multiple V_Σ nodes. This in turn means that less dropping is applied, and therefore the factorization is more exact. Retaining all nodes in this way, possibly only after reaching a certain level, gives us an exact factorization, which, in terms of iterations for the outer iterative solver, should give the same results as using any other direct solver at that level.

3.3 Skew partitioning in 2D and 3D

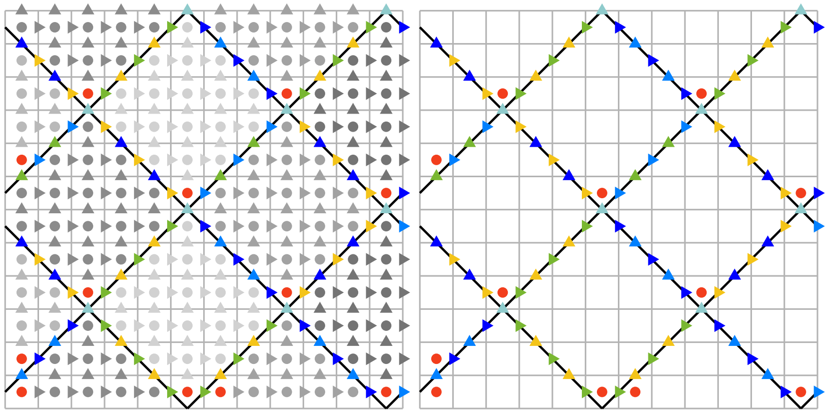
Looking at Figure 3.2, we see that there are pressures that are located at the corners of the subdomains that are surrounded by velocity separators. This means that if we add these pressures to the interior, as was suggested above, we get a singular interior block. We call these pressure nodes that are surrounded by velocity separators *isolated pressure nodes*. For the two-level preconditioner in 2D, it was possible to solve this problem by turning these pressures into their own separator groups. This unfortunately does not work for the multilevel method, since in this case velocity nodes around the isolated pressure nodes get eliminated. It also does not work in 3D because there the isolated pressure nodes also exist inside of the edge separators, where they form tubes of isolated nodes.

A possible way to solve this for the multilevel case and in 3D is to also turn all velocity nodes around the isolated pressure nodes into separate separator groups. This means that they will all be treated as V_Σ nodes and will never be eliminated until they are in the interior of the domain at a later level. This, however, has a downside that a lot more nodes have to be retained at every level, resulting in much larger $S_{\Sigma\Sigma}$ matrices at every level. Another downside is that a lot of bookkeeping is required to properly keep track of all the extra separator groups.

In 2D, we can resolve these problems by rotating the Cartesian partitioning by 45 degrees. The result is shown in Figure 3.3a. It is easy to see that in this case no pressure nodes are surrounded by only velocity separators. We call this partitioning method *skew partitioning*. In Figure 3.3, we also show the workings of the multilevel method, with all the steps being displayed in the different subfigures.

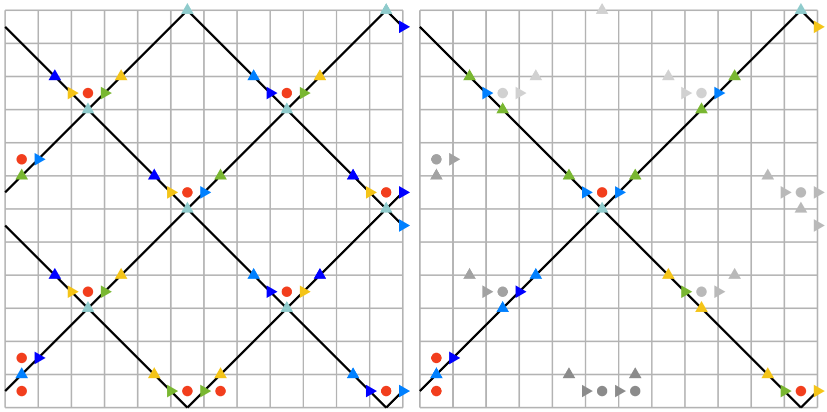
The most basic idea for generalizing the rotated square shape to a 3D setting was to use octahedral subdomains. Partitioning with this design turned out to be unsuccessful, but it is still briefly discussed here since it led to some new insights. Since regular octahedrons (the dual to cubes, having their vertices at the centers of the cube faces) in itself are not space tiling, the octahedrons can be slightly distorted to make them fit within a single cubic repeat unit. The resulting subdomains are space tiling, but only by using three mutually orthogonal subdomain types. The fact that it requires the use of three types of templates increases the programming efforts significantly since it introduces a lot of exceptional cases that should be considered, e.g. for subdomains located at the boundary of the domain.

A problem with the octahedral subdomains is that they are not scalable, meaning that we can not construct a larger octahedral subdomain from mul-



(a) Initial partitioning

(b) After elimination of interiors only the separator groups remain



(c) The Householder transformations leave one V_{Σ}-node per separator group

(d) Partitioning on next level

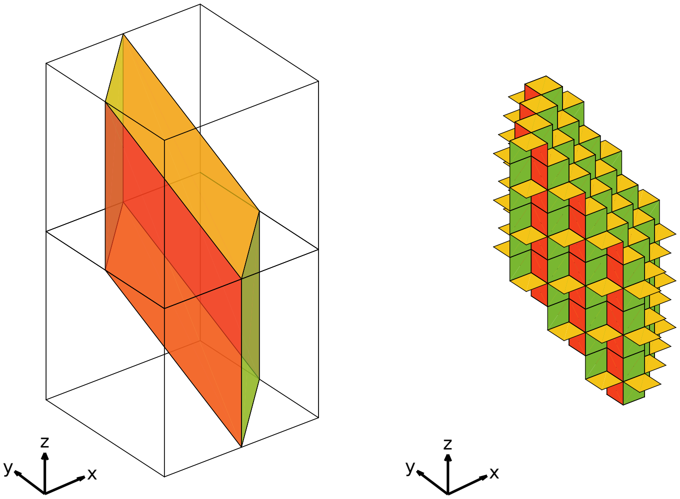
Figure 3.3: Skew partitioning of a 12 × 12 C-grid discretization of the Navier–Stokes equations into 12 subdomains. The interiors are shown in gray. Velocities of the same (non-gray) color that are on the same face of neighboring cells form a separator group. The red circles are pressure nodes that are retained. This example uses a coarsening factor of 2, i.e. the separators on the next level have twice the length of those on the previous level.

tiple smaller octahedral subdomains. This is of course required for the multilevel method. However, scalability can be achieved by splitting the octahedrons into four smaller tetrahedrons, of which six different types are required to fill 3D space. This would introduce additional separator planes that are similar to the 2D skew case and hence it increases the risk of isolating a pressure node when such planes intersect. Especially planar intersections which are parallel to either of the Cartesian axes have a high risk of producing isolating pressure nodes.

We did indeed not manage to find any scalable tiling using tetrahedrons that would not give rise to any isolated pressure nodes. Moreover, we would like to have a singular subdomain shape that we can use instead of six, since this would greatly simplify the implementation. A lesson we learned is that isolated pressure nodes always seem to arise when having faces that are aligned with the grid. Therefore, we looked into a rotated parallelepiped shape that does not have any faces that are aligned with the grid (Van der Klok, 2017). This shape is shown in Figure 3.4a, where the cubes represent a set of $s_x \times s_x \times s_x$ grid cells. A welcome property of this domain is that its central cross section resembles the proposed skew 2D partitioning method.

A schematic view of the position of the separator nodes is shown in Figure 3.4b. Here we see that on the side that is facing towards us, only half of the w-nodes are displayed. This is because the other half of the separator nodes is behind the u-nodes and v-nodes that are shown. This alternating property is required to make sure that we do not obtain isolated pressure nodes. If, for instance, all w-separator nodes that stick out in the figure were instead flipped to the inside, we would have an isolated pressure node in the bottom row. A consequence of this alternating property is that the w-planes have to be divided into two separate separator groups; one for each of the two neighboring subdomains in which these nodes are actually located.

Another advantage of using a skew domain partitioning is that the amount of communication that is required is reduced compared to a square partitioning. In Bisseling (2004), it is estimated that for the Laplace problem, the communication is asymptotically reduced by a factor of $\sqrt{2}$ for the 2D diamond shape. If we instead compare the diamond shape to a rectangular domain with the same number of nodes (having the same number of nodes with a square domain is impossible), we find that communication is reduced by a factor of $\frac{3}{2}$. In the same manner, we can compare a 3D domain of size $s_x \times s_x \times s_x/2$ to the rotated parallelepiped, and find a factor of $\frac{4}{3}$. We remark that the truncated octahedron that is used in Bisseling (2004) for the 3D domain and has a factor of 1.68 can not be used for our multilevel method, since truncated octahedra are not scalable.



(a) Shape of the parallelepiped inside of two cubes that exist of $m \times m \times m$ grid cells

(b) Schematic view of the position of the separator nodes, where u-nodes are depicted in red, v-nodes in green and w-nodes in yellow.

Figure 3.4: Parallelepiped shape for partitioning the domain.

Numerical results 3.4

A common benchmark problem in fluid dynamics is the lid-driven cavity problem. We consider an incompressible Newtonian fluid in a square three-dimensional domain of unit length, with a lid at the top which moves at a constant speed U. The equations are given by (3.1). No-slip boundary conditions are applied at the walls, meaning that they are Dirichlet boundary conditions, and the equations are discretized on an Arakawa C-grid as described before. We take $n_x = n_y = n_z$ grid points in every direction.

At first, however, we will only look at the Stokes equations of the form

$$\begin{aligned} \mu \Delta \boldsymbol{u} - \nabla p &= 0, \\ \nabla \cdot \boldsymbol{u} &= 0. \end{aligned}$$

This is because our preconditioner is constructed in such a way that memory usage and time cost for both computation and application of the preconditioner should not be influenced by inclusion of the convective term. After this, we will further investigate the behavior on the lid-driven cavity problem for increasing Reynolds numbers, which constitute harder problems. Therefore we expect an increase in iterations of the iterative solver, but otherwise the same behavior.

For obtaining the exact memory usage, we developed a custom library which overrides all memory allocation routines when linking against it. The library contains a hash table in which the amount of memory that is allocated is stored by its memory address. We keep track of the total amount of memory that is allocated, which is increased on memory allocation, and reduced by the amount that is stored in the hash table when memory is freed. The reason we did this is that existing methods rely on rough estimates of the memory usage of the used data structures, use the data that is available from `/proc/meminfo` which is inaccurate, or actually count memory usage in a similar way as we do (i.e. valgrind), but have a large performance impact.

We perform every experiment twice: once to determine the memory usage, and once to determine the run time, without linking to the memory usage library. This means that when reporting timing results, we are not impacted by the performance impact of tools to determine memory usage. The reason that we are still concerned about their performance impact, even though we developed our own library, is that it adds roughly a constant amount of time per process, which impacts scalability results. The memory usage that we report is the exact difference in memory usage before and after a certain action is performed, e.g. before and after the construction of the preconditioner.

For the timing results, we do not include the time it takes to compute the partitioning. The partitioning is computed by first constructing a template sequentially, which contains all possible nodes of exactly one overlapping subdomain, which may even partially be outside of the domain. This template is then mapped to the correct position for every other overlapping subdomain to determine the interiors and separator groups. However, since the template contains all possible nodes, every time we increase the number of levels by 1, the amount of time to compute this template increases by a factor 8, which is the worst possible scenario. The reason we do not include this in the timing results is that this may be resolved by only computing the partitioning for nodes that are still present in the Schur complement at that level. This would, however, require a full rewrite of the existing partitioning code, while it would not have any impact on the timing results of the rest of the code, since this is completely decoupled. This means that even though the partitioning method does not scale at all, the preconditioning method itself can be studied reliably without including the timing of the partitioning.

For the implementation of the preconditioner and solver, we use libraries from the Trilinos project (Heroux et al., 2005). The libraries we use are Epetra (vector and matrix data structures), IFPACK (direct solver and preconditioning interfaces) (Sala and Heroux, 2005), Amesos (direct solvers) (Sala et al., 2008) and Belos (iterative solvers) (Bavier et al., 2012). As iterative solver we use GMRES(250) (Saad and Schultz, 1986), as parallel sparse direct solver on the coarsest level we use SuperLU_DIST 6.1 (Liu et al., 2018), and as direct solver for the interior blocks we use KLU (Davis and Natarajan, 2010) with the fill-reducing ordering from De Niet and Wubs (2008).

The benchmark is performed on Cartesius, which is the Dutch national supercomputer. We used the Haswell nodes, which consist of 2 × 12-core 2.6 GHz Intel Xeon E5-2690 v3 (Haswell) CPUs per node with 64 GB per node. For all experiments, we disabled multithreading inside the MPI processes, since this is poorly implemented in Epetra, and causes results to become worse instead of better.

Due to issues with the interconnection between nodes, which are most likely caused by hardware issues, we were not able to run our code on more than 2048 cores. Note that this holds for a large number of applications and not just our application, and has been confirmed to not be an issue with our code. For this reason we do not look at the scalability past $8^3 = 512$ cores, even though we also wanted to add experiments for $8^4 = 4096$ cores. The connection issues that we observed are also likely to have impacted performance of the experiments that we report in this section, especially when larger numbers of cores are used.

Weak scalability 3.4.1

First, we look at results obtained when increasing the grid size n_x at the same rate as the number of used cores n_p, i.e. the problem size on each core is kept constant. The exact configurations that we use are 1 core for $n_x = 16$, 1 core for $n_x = 32$, 8 cores for $n_x = 64$, 64 cores for $n_x = 128$ and 512 cores for $n_x = 256$. The size of the subdomains (the size of the cubes in Figure 3.4a) at the first level is $s_x = 8$, while we choose the coarsening factor to be 2, meaning we increase s_x by a factor of 2 at each level. We perform experiments when keeping the number of levels constant at $L = 2$, and when increasing the number of levels by 1 when doubling the grid size. For the latter, we look at three cases where we retain a different number of separator nodes starting at level 2: 1, 4, and all.

For the fixed number of levels, we expect the number of iterations of the iterative solver to converge to a constant number as the domain size increases. For the case where we increase the number of levels as the domain size increases, we expect the number of iterations to only increase mildly, and we expect that retaining more separator nodes starting at level 2 decreases the number of iterations until it again becomes constant as we retain more and more nodes per separator.

The results are shown in Figure 3.5. We see that indeed the number of iterations becomes constant when fixing the number of levels or when retaining all separator nodes. When increasing the number of levels with the grid size, we see that the number of iterations appears to increase linearly. Interesting

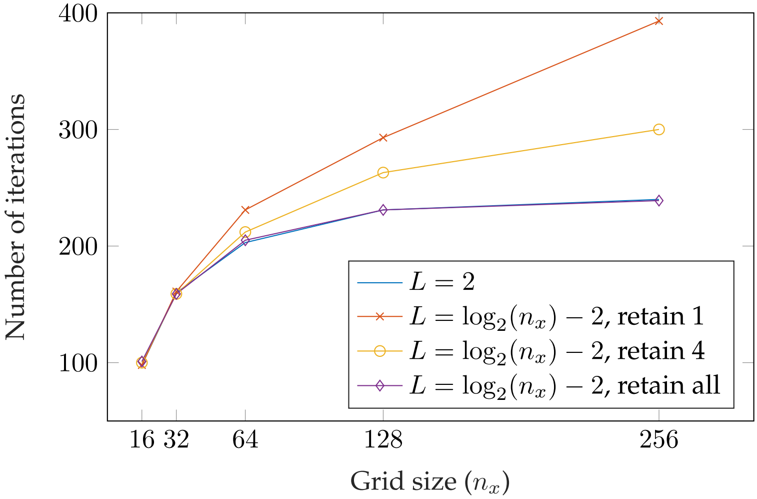


Figure 3.5: Number of iterations of GMRES for the Stokes problem on a grid of size $n_x \times n_x \times n_x$, while keeping the number of levels at $L = 2$, and when increasing the number of levels by 1 when n_x is doubled. A relative residual of 10^{-8} was used as convergence tolerance.

is that when retaining 4 instead of 1 separator node per separator group after level 2, the number of iterations that is required decreases drastically, even though this does not have a significant impact on the memory usage as we will see later.

The computational time of both computing the preconditioner, as well as the solution of the linear system would ideally become constant when keeping the problem size at each core constant while increasing the problem size. However, in practice this is not possible since increasing the number of cores also increases the required amount of communication. The results are shown in Figure 3.6 and Figure 3.7.

When computing the preconditioner, we see that especially at the largest grid sizes, the amount of computational time tends to increase, while for smaller problems it appeared to become constant. This is because for the smaller problems, at most 3 computing nodes were used, meaning very little communication between computing nodes was required. When increasing the number of computing nodes, the amount of required communication increases, which happens mainly at the point where contributions of neighboring subdomains have to be added up in the Schur complement. Since retaining more nodes per level means an increase in the amount of communication, we see that retaining 4 or all nodes takes more time than retaining only 1 node per separator at every level.

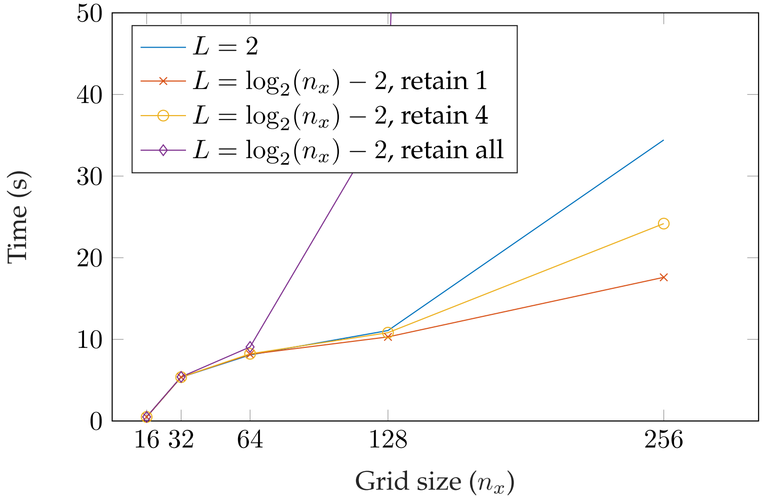


Figure 3.6: Time to compute the preconditioner for the Stokes problem on a grid of size $n_x \times n_x \times n_x$, while keeping the number of levels at $L = 2$, and when increasing the number of levels by 1 when n_x is doubled.

We also notice that retaining all nodes after level 2 is much more inefficient than just using SuperLU_DIST at level 2, meaning that our preconditioner performs very poorly as a direct solver. This is mainly due to the fact that the Schur complement at the last level is quite large and full. The last level Schur complement for a grid of size 256^3 has size 20961×20961 and its factorization is filled with 72% nonzeros. Computing the factorization of this matrix and applying it is therefore really expensive. Using a parallel dense solver instead of SuperLU_DIST should help to make it more efficient. Moreover, since the cost of computing the factorization in a sparse direct solver goes at best with $\mathcal{O}(n^2/n_p) = \mathcal{O}(n_x^3)$ (Liu et al., 2018), we expect that the cost of computing the preconditioner when keeping the number of levels constant, or when retaining all separator nodes, increases with n_x^3.

In Figure 3.7, we show both the time it takes to solve the linear system after computation of the preconditioner, and the time of 1000 applications of the preconditioner, which is not influenced by the number of iterations of the iterative solver and does not include time spent on for instance matrix-vector products and orthogonalization. First of all, we again observe that retaining all separator nodes is a bad idea, since the computation time goes off the chart. For the case where we only use 2 levels, we also see the unwanted behavior of a time that keeps increasing linearly for the total solution time, and superlinearly for the application of the preconditioner, where we would actu-

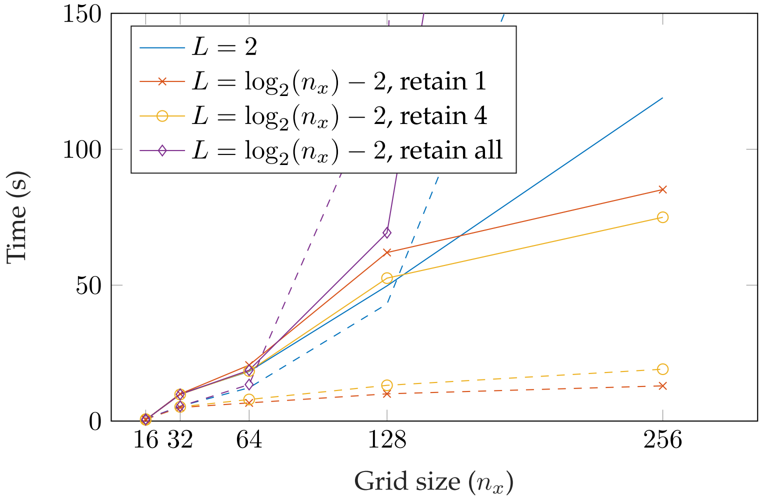


Figure 3.7: Time to solve the linear system with GMRES (lines), and time of 1000 applications of the preconditioner (dashed lines). This is for the Stokes problem on a grid of size $n_x \times n_x \times n_x$, while keeping the number of levels at $L = 2$, and when increasing the number of levels by 1 when n_x is doubled. A relative residual of 10^{-8} was used as convergence tolerance.

ally expect $\mathcal{O}(n^{4/3}/n_p) = \mathcal{O}(n_x)$ behavior from the triangular solve (Liu et al., 2018). This may be caused by disabling multithreading support, which results in a lot of extra communication inside of SuperLU_DIST.

For both cases where we retain 1 and 4 separator nodes after 2 levels, we see that the computational time appears to become constant for larger grid sizes. We also note that even though the case where we retain 4 nodes is slower per application, it is faster in total solution time due to the lower number of iterations that is required, as can also be seen in Figure 3.5, and the fact that applying the preconditioner is so cheap.

In Figure 3.8, we see the average memory usage of the preconditioner per core. Here we again observe that the 2 level case, and the case where we retain all separator nodes after level 3 performs poorly. The case where we retain 1 or 4 separator nodes per separator group after level 2 show similar behavior in terms of memory usage. We see, however, that the memory usage of the latter two cases does not become constant. We will discuss this further in the next section.

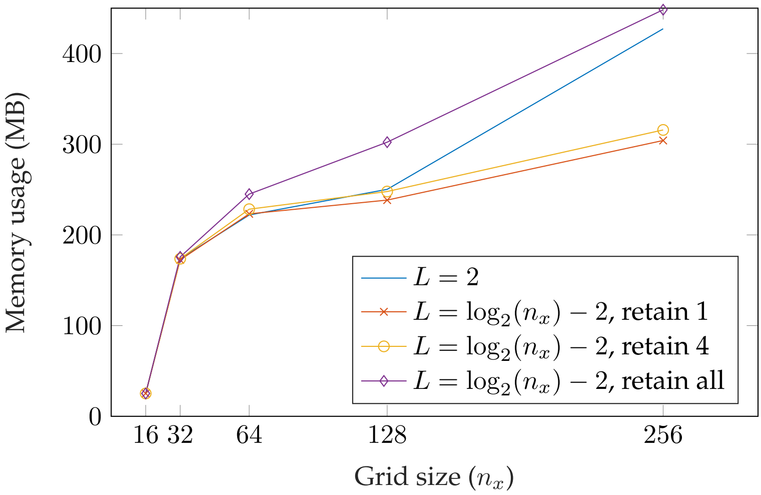


Figure 3.8: Memory usage of the preconditioner per core for the Stokes problem on a grid of size $n_x \times n_x \times n_x$, while keeping the number of levels at $L = 2$, and when increasing the number of levels by 1 when n_x is doubled.

Strong scalability 3.4.2

In this section, we will look at a problem with size $n_x = 128$, with 6 levels and retaining only one node per separator group. We use 1 to 128 cores for the memory usage and 2 to 128 cores for the computational time, with steps of a factor of 2. Reason we do not look at 1 core for the timing is that this configuration caused a memory allocation error in the iterative solver (Belos).

We first look at the total memory usage in Figure 3.9. We observe that there is a large difference between the memory usage on one core, and the memory usage on two cores. This is due to the fact that mainly Epetra uses different implementations of its data structures for serial and parallel computations. We also observe that the total memory usage increases when using more cores. We found that this increase happens when the so called column maps are constructed, which are used in every sparse matrix data structure in Epetra. The matrices in Epetra are distributed by rows, meaning that a row is always stored on a single core. The global indices of local rows of a certain process are stored in the row map. This map is of roughly constant size independent of the number of processes due to its unique nature. The column indices of the nonzeros that are present in one row may, however, not belong to rows that are also stored in the same process. Therefore the corresponding column map, which stores all the column indices, has overlap between different processes, and increases in size when more processes are used. The map is needed in

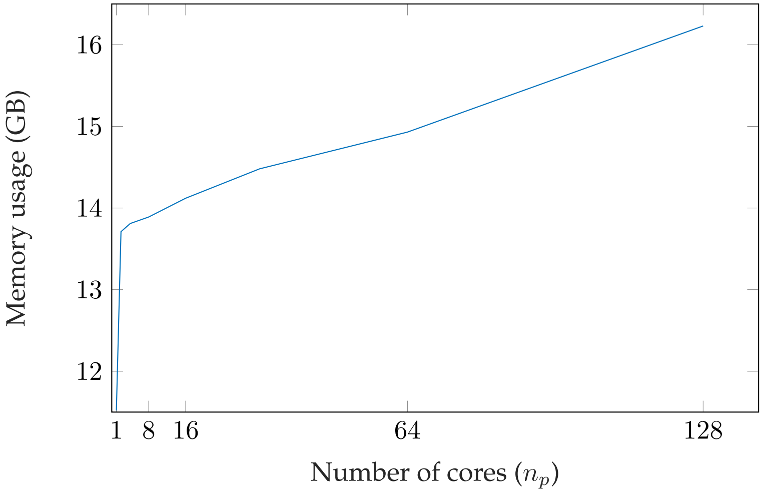


Figure 3.9: Memory usage of the preconditioner for the Stokes problem on a grid of size 128^3, with 1 to 128 cores.

for instance a matrix-vector product. That is, in case of a square matrix, the left- and right-hand side vectors have the same map as the row map of the matrix, so to multiply one row of the matrix with the left hand side vector requires communication with all indices that are stored in the column map. This means that not only the memory usage increases, but also the time cost, since a larger column map means more communication is required. Note that we also see this behavior not only in our preconditioner but also in the storage of the original matrix.

Since the difference in communication between a cubical domain and a parallelepiped shaped domain is a constant factor of $\frac{3}{4}$, we may instead look at a cubical domain to explain this behavior. Say we have a subdomain of size $s_x \times s_y \times s_z$, then communication is required for $2s_xs_y + 2s_xs_z + 2s_ys_z$ grid cells. If we now split this subdomain in the x-direction, then we require communication for $2s_xs_y + 2s_xs_z + 4s_ys_z$ grid cells. If we now assume that $s_x = s_y = s_z$, we see that communication increases with a factor $\frac{4}{3}$, meaning that we expect the size of the column map increases with a factor $\frac{4}{3}$ when doubling the number of cores. This is, however, not what we observe. When increasing the number of cores even further, we even see a factor of 10 instead of $\frac{4}{3}$. This appears to be because of caching that happens inside of both Epetra and MPI itself when the column map is constructed. We checked that the actual column map shows the $\frac{4}{3}$ behavior that we expect.

In Figure 3.10 we look at the speedup when using more cores. Ideally, we would see that using twice the number of cores means half of the computa-

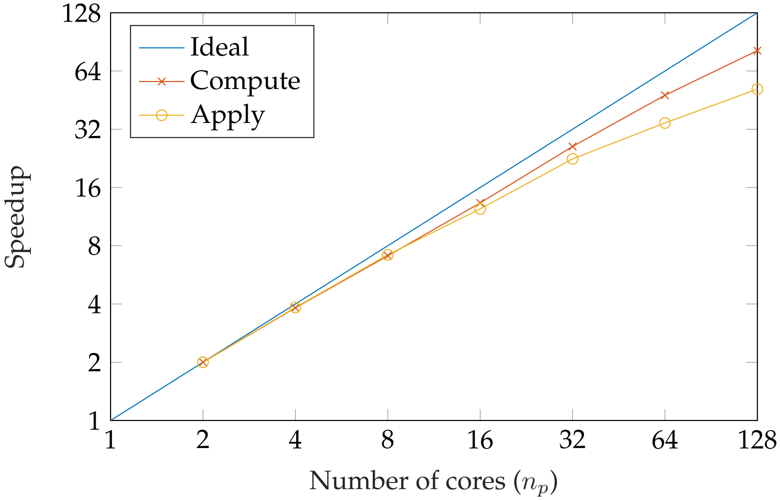


Figure 3.10: Speedup of computation and application of the preconditioner for the Stokes problem on a grid of size 128^3, with 2 to 128 cores.

tional time. We plotted this ideal line for reference. We see that the speedup of the computation of the preconditioner is very close to this ideal line. The application of the preconditioner is a bit further from the ideal line. This is again mostly due to the communication that is required for the non-local column indices, but may also be affected by the issues with the interconnection between nodes that we discussed earlier.

Lid-driven cavity 3.4.3

In the previous sections we determined the strong and weak scalability properties of the preconditioner, which means that we can now continue with the robustness of the solver on the lid-driven cavity problem with increasing Reynolds numbers. We perform a continuation with steps of Re $= 100$ starting from the solution of the Stokes problem. We show the results of the first iteration of Newton at Re $= 500$ and Re $= 2000$. Reason we go up to Re $= 2000$ is that a Hopf bifurcation is located between Re $= 1900$ and Re $= 2000$, which is of interest (Baars et al., 2019b).

In Figure 3.11 and Figure 3.12, we show the results at Reynolds number 500, which show good correspondence with the results on the Stokes problem. Again the number of iterations appears to become constant when the number of levels is kept the same, and retaining more nodes gives better convergence. The main difference is that more iterations are needed, since higher

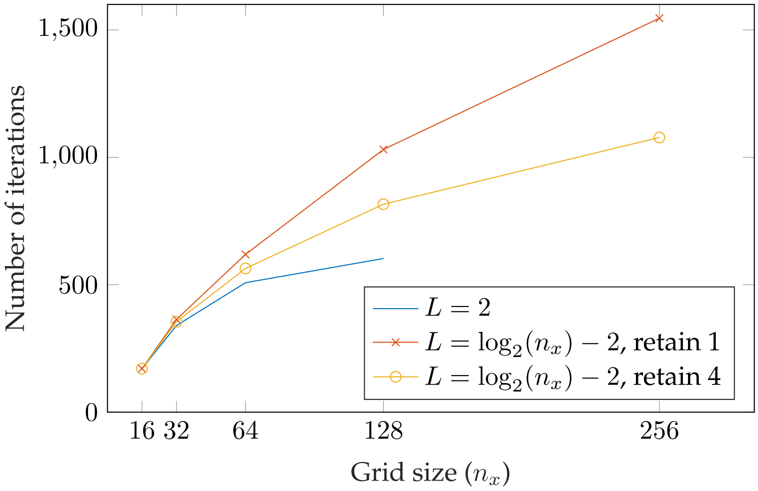


Figure 3.11: Number of iterations of GMRES for the lid-driven cavity problem at Re = 500 on a grid of size $n_x \times n_x \times n_x$, while keeping the number of levels at $L = 2$, and when increasing the number of levels by 1 when n_x is doubled. A relative residual of 10^{-8} was used as convergence tolerance.

Reynolds numbers constitute harder problems. Also interesting is that for the 256^3 grid with 2 levels, SuperLU_DIST aborted without any error message, also when using different amounts of cores. In Baars et al. (2019b), we observed convergence, however, so we suspect this is due to a change that was made in version 6 of SuperLU_DIST.

The results at Reynolds number 2000 are shown in Figure 3.13 and Figure 3.14. We again observe that the number of iterations is much larger. What is odd, however, is that retaining more nodes now actually gives worse convergence. This may be because we use GMRES(250) instead of GMRES due to memory limitations, and therefore do not preserve the convergence properties of GMRES. This effect is more prevalent for this problem because of the large number of iterations that is required. We do observe that for the first 250 iterations, retaining 4 nodes after level 2 gives rise to better convergence, as we expected.

In Table 3.1 we show results for Reynolds number 500 using the block preconditioner LSC (Least-Squares Commutator) implemented in Teko (Cyr et al., 2012). We chose Teko for comparison because it is available in Trilinos and can therefore easily be coupled to our code. We also tried other preconditioners, but those did unfortunately not yield convergence. The stopping criterion of the linear solver is 10^{-8}, as before.

Compared to our method, we see that Teko has much more difficulty with

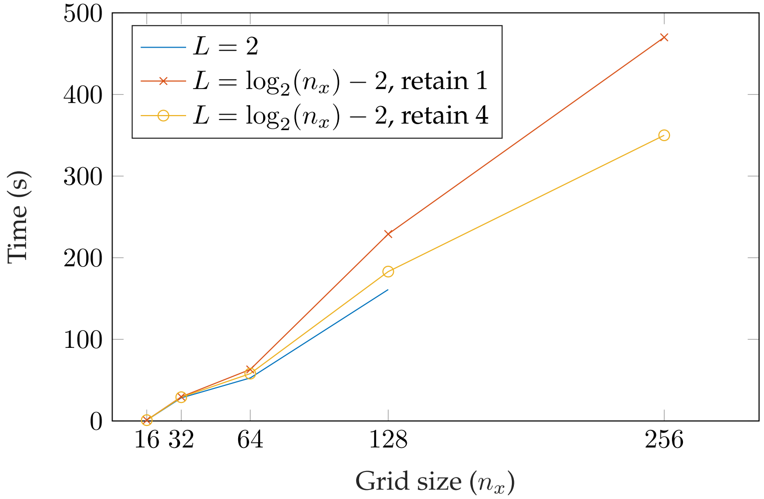


Figure 3.12: Time for GMRES to converge for the lid-driven cavity problem at Re $= 500$ on a grid of size $n_x \times n_x \times n_x$, while keeping the number of levels at $L = 2$, and when increasing the number of levels by 1 when n_x is doubled. A relative residual of 10^{-8} was used as convergence tolerance.

n_p	n_x	L	its	t_c (s)	t_s (s)
1	16	2	142	0.12	0.77
1	32	2	187	9.88	18.80
8	64	3	245	1511.12	313.00

Table 3.1: Performance of Teko with LSC preconditioner for the lid-driven cavity problem at Re $= 500$ on a grid of size $n_x \times n_x \times n_x$. Here n_p is the number of cores, L is the number of levels, its is the number of iterations, t_c is the time to compute the preconditioner and t_s is the time for solving the linear system.

the grid refinement, leading to a huge computational cost already at a grid size of 64^3. A crude computation shows that per grid point the method becomes about 65 times more expensive per iteration. This must be attributed to slow convergence of algebraic multigrid on the subblocks. One of these blocks is the $L + N$ block from (3.2), with N indefinite, and this is something that is difficult for a standard AMG method. We chose the number of levels to be 2 for grid sizes 16^3 and 32^3, and 3 levels for 64^3 since these seemed to give the optimal results. For Reynolds number 2000, we did not observe convergence past the 16^3 grid.

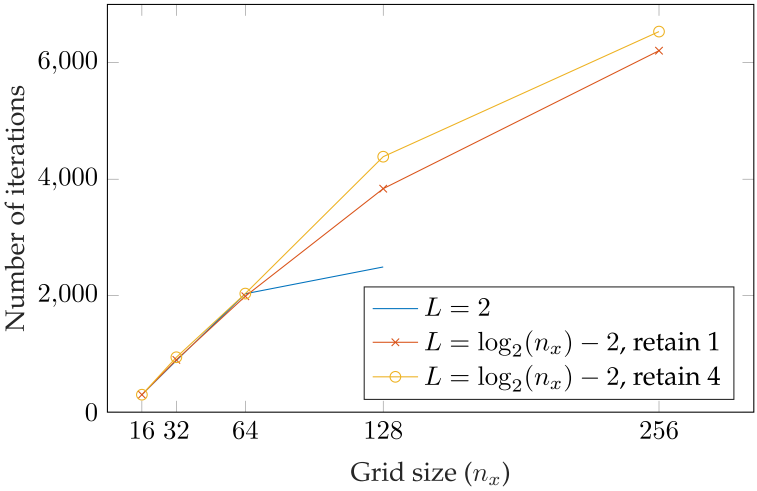


Figure 3.13: Number of iterations of GMRES for the lid-driven cavity problem at Re $=$ 2000 on a grid of size $n_x \times n_x \times n_x$, while keeping the number of levels at $L = 2$, and when increasing the number of levels by 1 when n_x is doubled. A relative residual of 10^{-8} was used as convergence tolerance.

3.5 Summary and Discussion

We presented a robust method for solving the steady, incompressible Navier–Stokes equations, which makes use of parallelepiped shaped overlapping subdomains. The interiors of these overlapping subdomains can be eliminated in parallel. On the interfaces of the subdomains, Householder transformations are applied to decouple all but one velocity node from the pressure nodes, after which all decoupled nodes can also be eliminated in parallel. The key to the multilevel approach is the resulting reduced Schur complement matrix, which has the same structure as the original matrix. Therefore we can recursively apply the preconditioner to this matrix. The shape of the subdomains makes it such that pressure nodes are not isolated in the factorization process, which would result in a singular Schur complement matrix.

Our weak scalability experiments show that if the number of levels of the multilevel approach is kept constant while increasing the problem size, grid independent convergence is obtained. We also show that by increasing the number of levels and processors as the problem size increases, we only require a small amount of additional time and memory for both the factorization and solution process. Moreover, by increasing the number of nodes that is retained per separator after level 2, we can further decrease the time that is required to solve the system.

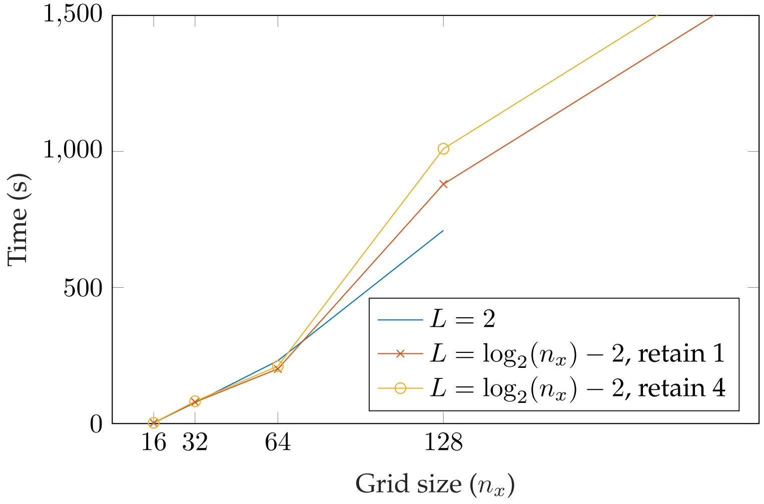


Figure 3.14: Time for GMRES to converge for the lid-driven cavity problem at Re $=$ 2000 on a grid of size $n_x \times n_x \times n_x$, while keeping the number of levels at $L = 2$, and when increasing the number of levels by 1 when n_x is doubled. A relative residual of 10^{-8} was used as convergence tolerance.

Our strong scalability experiments show that the time that is required to compute the preconditioner scales very well. The memory that is used for the preconditioner scales slightly less optimal, but this can be explained by the caching of communication related data structures. The slightly less optimal time it takes to apply the preconditioner can also be explained by means of the increased communication.

We also showed that the preconditioner gives rise to convergence of GMRES for the lid-driven cavity problem at high Reynolds numbers, where other methods, such as the LSC preconditioner that in implemented in Teko, fail to do so.

This leads us to conclude that we implemented a robust solution method for the Navier–Stokes equations on staggered grids which shows good parallel scalability. In the future we want to apply our preconditioner in ocean-climate models. The fact that it has proven to be robust for the lid-driven cavity problem with higher Reynolds numbers leads us to believe that it will also perform well for the ocean model described in Section 2.5.2.

LYAPUNOV EQUATIONS

In this chapter, 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. In our case this is the stochastic representation of the freshwater forcing as described in Section 2.5.2. The direct way to investigate these flows is to use ensemble simulation techniques for many initial conditions to estimate the probability density function (PDF) for several observables of the flows (Slingo and Palmer, 2011). 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) (Sapsis and Lermusiaux, 2009). Non-Markovian reduced models can also be obtained by projecting on a basis of eigenvectors of the underlying deterministic system (Chekroun et al., 2015).

In a deterministic-stochastic continuation method recently suggested by Kuehn (2012), 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 the

Bartels–Stewart algorithm (Bartels and Stewart, 1972) 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 (Simoncini, 2007; Druskin and Simoncini, 2011; Stykel and Simoncini, 2012; Druskin et al., 2014) and alternating directions implicit (ADI) based iterative methods (Kleinman, 1968; Penzl, 1999) might also become expensive for high-dimensional problems, particularly when trying to use previous initial guesses along a continuation branch.

The aim of this chapter is to present new methodology to efficiently trace PDFs of SPDEs with algebraic constraints in parameter space. In Section 4.1, an extension of the approach suggested in Kuehn (2012) and the novel procedure to efficiently solve a generalized Lyapunov equation numerically are presented. In fact, this solves an open conjecture in Kuehn (2015), which states that it should be possible to reuse previous solutions of a continuation in a specialized Lyapunov solver. We describe the results on the model of the Atlantic Ocean circulation in Section 4.2. The numerical aspects and capabilities of the novel method for this application are shown in Section 4.3. In Section 4.4, we provide a summary and discuss the results.

4.1 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 (Griffies, 2004)

$$\mathcal{M}(\boldsymbol{p})\frac{\partial \boldsymbol{u}}{\partial t} = \mathcal{L}(\boldsymbol{p})\boldsymbol{u} + \mathcal{N}(\boldsymbol{u},\boldsymbol{p}) + \mathcal{F}(\boldsymbol{u},\boldsymbol{p}), \tag{4.1}$$

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

4.1.1 Formulation of the problem

When (4.1) is discretized, eventually a set of ordinary differential equations with algebraic constraints arises, which can be written as

$$M(\boldsymbol{p})\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = L(\boldsymbol{p})\boldsymbol{x} + N(\boldsymbol{x},\boldsymbol{p}) + F(\boldsymbol{x},\boldsymbol{p}), \tag{4.2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $M(\boldsymbol{p}) \in \mathbb{R}^{n\times n}$ is a generally singular matrix of which every zero row is associated with an algebraic constraint, $L(\boldsymbol{p}) \in \mathbb{R}^{n\times n}$ is the discretized version of $\mathcal{L}$, and $F : \mathbb{R}^m \to \mathbb{R}^n$ and $N : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ 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

$$M(\boldsymbol{p})\,\mathrm{d}\boldsymbol{X}_t = \boldsymbol{f}(\boldsymbol{X}_t;\boldsymbol{p})\,\mathrm{d}t + \boldsymbol{g}(\boldsymbol{X}_t;\boldsymbol{p})\,\mathrm{d}\boldsymbol{W}_t, \tag{4.3}$$

where $\boldsymbol{f}(\boldsymbol{X}_t;\boldsymbol{p}) = L(\boldsymbol{p})\boldsymbol{X}_t + N(\boldsymbol{X}_t,\boldsymbol{p}) + F(\boldsymbol{X}_t,\boldsymbol{p})$ is the right-hand side of (4.2), $\boldsymbol{W}_t \in \mathbb{R}^{n_w}$ is a vector of n_w-independent standard Brownian motions (Gardiner, 1985), and $\boldsymbol{g}(\boldsymbol{X}_t;\boldsymbol{p}) \in \mathbb{R}^{n_w\times n}$.

Suppose that the deterministic part of (4.3) has a stable fixed point $\bar{\boldsymbol{x}} = \bar{\boldsymbol{x}}(\boldsymbol{p})$ for a given range of parameter values. Then linearization around the deterministic steady state yields (Kuehn, 2012)

$$M(\boldsymbol{p})\,\mathrm{d}\boldsymbol{X}_t = A(\bar{\boldsymbol{x}};\boldsymbol{p})\boldsymbol{X}_t\,\mathrm{d}t + B(\bar{\boldsymbol{x}};\boldsymbol{p})\,\mathrm{d}\boldsymbol{W}_t, \tag{4.4}$$

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

In the special case that M is a nonsingular matrix, the equation (4.4) can be rewritten as

$$\mathrm{d}\boldsymbol{X}_t = M^{-1}A\boldsymbol{X}_t\,\mathrm{d}t + M^{-1}B\,\mathrm{d}\boldsymbol{W}_t, \tag{4.5}$$

which represents an n-dimensional Ornstein–Uhlenbeck (OU) process. The corresponding stationary covariance matrix C is determined from the following *Lyapunov equation* (Gardiner, 1985)

$$M^{-1}AC + CA^TM^{-T} + M^{-1}BB^TM^{-T} = 0.$$

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

$$ACM^T + MCA^T + BB^T = 0. \tag{4.6}$$

If M is a singular diagonal matrix then (4.5) does not apply. However, if the stochastic part is non-zero only on the part where M is nonsingular (which occurs often when the noise is in the forcing of the flow), then (4.4) can be written as

$$\begin{pmatrix} 0 & 0 \\ 0 & M_{22} \end{pmatrix}\begin{pmatrix} \mathrm{d}\boldsymbol{X}_{t,1} \\ \mathrm{d}\boldsymbol{X}_{t,2} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} \boldsymbol{X}_{t,1} \\ \boldsymbol{X}_{t,2} \end{pmatrix}\mathrm{d}t + \begin{pmatrix} 0 \\ B_2 \end{pmatrix}\mathrm{d}\boldsymbol{W}_t, \tag{4.7}$$

where M_{22} represents the nonsingular part of M. Consequently, for nonsingular A_{11} we can separate (4.7) into an algebraic part and an explicitly time-dependent part,

$$A_{11}\boldsymbol{X}_{t,1} + A_{12}\boldsymbol{X}_{t,2} = 0, \tag{4.8a}$$

$$M_{22}\,\mathrm{d}\boldsymbol{X}_{t,2} = S\boldsymbol{X}_{t,2}\,\mathrm{d}t + B_2\,\mathrm{d}\boldsymbol{W}_t, \tag{4.8b}$$

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 $\bar{\boldsymbol{x}}$. Since M_{22} is nonsingular by construction we can find the stationary covariance matrix C_{22} by solving the corresponding generalized Lyapunov equation

$$SC_{22}M_{22}^T + M_{22}C_{22}S^T + B_2B_2^T = 0. \tag{4.9}$$

We remark that (4.9) could alternatively be derived using an epsilon-embedding approach for the differential algebraic equations. In order to find the full covariance matrix we use $C_{ij} = \mathbb{E}[\boldsymbol{X}_{t,i}\boldsymbol{X}_{t,j}^T]$ together with (4.8a) which gives

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

In summary, we can obtain an estimate of the covariance matrix C of the stochastic dynamical system (4.3) by providing the matrices A, B and M and solving the corresponding generalized Lyapunov equation (4.6), or (4.9) in case M is singular. Once the covariance matrix C is computed, the stationary PDF of the approximating OU-process, indicated by $p(\boldsymbol{x})$ follows as (Gardiner, 1985; Kuehn, 2011)

$$p(\boldsymbol{x};\bar{\boldsymbol{x}}) = \frac{1}{(2\pi)^{\frac{n}{2}}} \mid C \mid^{-1/2} e^{-\frac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T C^{-1}(\boldsymbol{x}-\bar{\boldsymbol{x}})}. \tag{4.10}$$

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.

4.1.2 A novel iterative generalized Lyapunov solver

The type of systems of the form (4.6) 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 $C \approx VTV^T$. In existing iterative solution methods for low-rank approximations (Kleinman, 1968; Penzl, 1999; Saad, 1990; Simoncini, 2007; Stykel and Simoncini, 2012) 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 number of vectors as the number of columns in B.

The solution method we propose is based on a Galerkin projection, and is very similar to the method in Saad (1990). It works by solving projected systems of the form

$$V^TAVTV^TM^TV + V^TMVTV^TA^TV + V^TBB^TV = 0,$$

where C is approximated by a low-rank approximation $\tilde{C} = VTV^T$, and T is generally a dense symmetric matrix which should not be confused with the superscript T denoting transposition. Now if we take $\tilde{A} = V^TAV$, $\tilde{M} = V^TMV$, $\tilde{B} = V^TB$, we get the smaller projected generalized Lyapunov equation

$$\tilde{A}T\tilde{M}^T + \tilde{M}T\tilde{A}^T + \tilde{B}\tilde{B}^T = 0, \tag{4.11}$$

which can be solved by a dense solution routine (Bartels and Stewart, 1972).

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

$$R = A\tilde{C}M^T + M\tilde{C}A^T + BB^T, \tag{4.12}$$

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 Lanczos (1950). Even though we cannot compute R explicitly, it is possible to apply the Lanczos method to determine the eigenpair because only matrix vector products $R\boldsymbol{x}$ are needed, which are evaluated as

$$R\boldsymbol{x} = A(V(T(V^T(M^T\boldsymbol{x})))) + M(V(T(V^T(A^T\boldsymbol{x})))) + B(B^T\boldsymbol{x}).$$

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 (4.11), 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 3. Because of the peculiar choice of vectors to expand our space, we call this method the Residual Approximation-based Iterative Lyapunov Solver (RAILS).

```
input:   $A, B, M$     Matrices from (4.6).
         $V_1$         Initial space.
         $m$           Dimension increase of the space per iteration.
         $l$           Maximum number of iterations.
         $\epsilon$    Convergence tolerance.
output:  $V_k, T_k$    Approximate solution, where $C \approx V_k T_k V_k^T$.

Orthonormalize $V_1$
Compute $\tilde{A}_1 = V_1^T A V_1$
Compute $\tilde{M}_1 = V_1^T M V_1$
Compute $\tilde{B}_1 = V_1^T B$
for $j = 1, \dots, l$ do
    Obtain $\tilde{A}_j = V_j^T A V_j$ by only computing new parts
    Obtain $\tilde{M}_j = V_j^T M V_j$ by only computing new parts
    Obtain $\tilde{B}_j = V_j^T B$ by only computing new parts
    Solve $\tilde{A}_j T_j \tilde{M}_j^T + \tilde{M}_j T_j \tilde{A}_j^T + \tilde{B}_j \tilde{B}_j^T = 0$
    Compute the approximate largest $m$ eigenpairs ($\lambda_p$, $\boldsymbol{r}_p$) of the residual $R_j$ using Lanczos
    Stop if the approximated largest eigenvalue is smaller than $\epsilon$
    $V_{j+1} = [V_j, \boldsymbol{r}_1, \dots, \boldsymbol{r}_m]$
    Re-orthonormalize $V_{j+1}$
```

Algorithm 3: RAILS algorithm for the projection based method for solving generalized Lyapunov equations.

4.1.3 Convergence analysis

We will now show why we choose the eigenvectors associated with the largest eigenvalues of the residual. Here we use $\operatorname{orth}(B)$ to denote the orthonormalization of B. We first show that in a special case, V_k as defined in Algorithm 3 spans the Krylov subspace

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

Proposition 4.1.1. *If $M = I$, $B \in \mathbb{R}^{n\times m}$, where m is also the number of vectors we use to expand the space V_k in every iteration and $V_1 = \mathrm{orth}(B)$, then $\mathrm{Range}(V_k) \subseteq \mathcal{K}_k(A, B)$.*

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

$$R_k\boldsymbol{q} = \lambda\boldsymbol{q} = AV_k\boldsymbol{q}_1 + V_k\boldsymbol{q}_2 + B\boldsymbol{q}_3$$

where $\boldsymbol{q}_1 = T_kV_k^T\boldsymbol{q}$, $\boldsymbol{q}_2 = T_kV_k^TA^T\boldsymbol{q}$, $\boldsymbol{q}_3 = B^T\boldsymbol{q}$. From this it is easy to see that if we orthonormalize $\boldsymbol{q}$ 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

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

□

Remark 4.1.1. *In case Q_i has full rank for every $i = 1, \dots, k$ it is clear that actually the equality $\mathrm{Range}(V_k) = \mathcal{K}_k(A, B)$ holds in Proposition 4.1.1. This is also the behavior that we observed on a variety of different test problems.*

From Proposition 4.1.1 and Remark 4.1.1, we see that when we choose m equal to the number of columns of B, RAILS is equivalent to the method in Saad (1990) 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 4.1.2. *If $M = I$, $B \in \mathbb{R}^{n\times m}$, where m is also the number of vectors we use to expand the space V_k in every iteration, $V_1 = \mathrm{orth}(A^{-1}B)$ 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 $\mathrm{Range}(V_k) \subseteq \mathcal{K}_k(A^{-1}, A^{-1}B)$.*

Proof. This can be proved analogously to Proposition 4.1.1. □

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 $V_1 = \mathrm{orth}([B, A^{-1}B])$ 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 4.1.3. *Take $M = I$ and $B \in \mathbb{R}^{n\times m}$. After k steps of RAILS, the residual R_k has an eigenpair $(\lambda, \boldsymbol{q})$ with $\lambda = \|R_k\|_2$. If $\boldsymbol{q} \in \operatorname{Range}(V_k)$, then $V_k T_k V_k^T$ is the exact solution.*

Proof. We have

$$R_k\boldsymbol{q} = \lambda\boldsymbol{q} = AV_kT_kV_k^T\boldsymbol{q} + V_kT_kV_k^TA^T\boldsymbol{q} + BB^T\boldsymbol{q}.$$

Since $\boldsymbol{q} \in \operatorname{Range}(V_k)$ and V_k is orthonormalized, it holds that $\boldsymbol{q} = V_kV_k^T\boldsymbol{q}$. So then

$$\begin{aligned}\lambda\boldsymbol{q} = \lambda V_kV_k^T\boldsymbol{q} &= V_kV_k^TAV_kT_kV_k^T\boldsymbol{q} + V_kT_kV_k^TA^TV_kV_k^T\boldsymbol{q} + V_kV_k^TBB^TV_kV_k^T\boldsymbol{q}\\ &= V_k\tilde{A}_kT_kV_k^T\boldsymbol{q} + V_kT_k\tilde{A}_k^TV_k^T\boldsymbol{q} + V_k\tilde{B}_k\tilde{B}_k^TV_k^T\boldsymbol{q}\\ &= V_k(\tilde{A}_kT_k + T_k\tilde{A}_k^T + \tilde{B}_k\tilde{B}_k^T)V_k^T\boldsymbol{q}\\ &= 0\end{aligned}$$

since the part between brackets is the projected Lyapunov equation that we solved for. This shows us that the residual is zero, so $V_kT_kV_k^T$ is the exact solution. □

Corollary 4.1.1. *From Proposition 4.1.3 it follows that when $m = 1$, the equality $\operatorname{Range}(V_k) = \mathcal{K}_k(A, B)$ holds in Proposition 4.1.1.*

4.1.4 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 number 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^T, which are then used as V in the next iteration of our method. Note that since V is orthonormalized, the nonzero eigenvalues of VTV^T 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 4.1.3. 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 3 by Algorithm 4.

```
input:  $k$     Iterations after which to restart.
        $\tau$  Tolerance for the eigenvalues at a restart.

Set converged to true if the approximated largest eigenvalue is smaller than $\epsilon$
if converged and convergence was already achieved earlier then
    stop
else if converged or $j \bmod k = 0$ then
    Compute the eigenpairs ($\lambda_p$, $\boldsymbol{q}_p$) of $T_j$
    $U = [\,]$
    for all eigenvalues $\lambda_p$ larger than $\tau$ do
        $U = [U, \boldsymbol{q}_p]$
    $V_{j+1} = V_j U$
    $\tilde{A}_{j+1} = U^T \tilde{A}_j U$
    $\tilde{M}_{j+1} = U^T \tilde{M}_j U$
    $\tilde{B}_{j+1} = U^T \tilde{B}_j$
```

Algorithm 4: Restart method that replaces Line 11 in Algorithm 3.

Extended generalized Lyapunov equations 4.1.5

To explain the non-Gaussian behavior in sea surface temperature (Sardeshmukh and Sura, 2009) and sea surface height (Sura and Gille, 2010) an additional multiplicative noise term was introduced. The resulting correlated additive and multiplicative (CAM) noise gives rise to linearized stochastic differential-algebraic equations of the form

$$\begin{aligned} M(\boldsymbol{p})\,\mathrm{d}\boldsymbol{X}_t = A(\bar{\boldsymbol{x}};\boldsymbol{p})\boldsymbol{X}_t\,\mathrm{d}t + \\ (B(\bar{\boldsymbol{x}};\boldsymbol{p}) + [N_1(\bar{\boldsymbol{x}};\boldsymbol{p})\boldsymbol{X}_t, \ldots, N_m(\bar{\boldsymbol{x}};\boldsymbol{p})\boldsymbol{X}_t])\,\mathrm{d}\boldsymbol{W}_t, \end{aligned}$$

where m is the dimension of the noise increment $\mathrm{d}\boldsymbol{W}_t$. The corresponding covariance matrix C can be determined from the *extended generalized Lyapunov equation*

$$ACM^T + MCA^T + BB^T + \sum_{j=1}^{m} N_j C N_j^T = 0, \tag{4.13}$$

where the matrices A and N_j are $n \times n$ matrices with A nonsingular, while B is $n \times m$. Note that these equations with $M = I$ are often referred to as generalized Lyapunov equations, but to avoid confusion with generalized Lyapunov equations of the form (4.6), we refer to these equations as extended Lyapunov equations.

Recently a method was proposed in Shank et al. (2015) for solving extended generalized Lyapunov equations. The method consists of applying a stationary iteration during which $N_j C N_j^T$ terms from the previous iteration are added in every iteration. For this method, we assume that the solution C is positive semidefinite, and in case that M is also positive definite, we require that A is negative semidefinite and that $\rho(\mathcal{M}^{-1}\mathcal{N}) < 1$, where $\mathcal{M}(X) = AXM^T + MXA^T$, $\mathcal{N}(X) = \sum_{j=1}^{m} N_j X N_j^T$ and $\rho(\mathcal{L})$ denotes the spectral radius of the operator $\mathcal{L}$ (Shank et al., 2015). The algorithm is shown in Algorithm 5.

```
input:   $A, B, M, N_j$   Matrices from (4.13).
         $l$              Maximum number of iterations.
         $\epsilon$       Convergence tolerance.
output:  $V_k$            Approximate solution, where $C \approx V_k V_k^T$.

solve $ACM^T + MCA^T + BB^T = 0$ for $C_1 = V_1 V_1^T$
for $i = 2, 3, \ldots, l$ do
    $B_i = [N_1 V_{i-1}, \ldots, N_m V_{i-1}, B]$
    solve $ACM^T + MCA^T + B_i B_i^T = 0$ for $C_i = V_i V_i^T$
    if $\|AC_iM^T + MC_iA^T + \sum_{j=1}^{m} N_j C N_j^T\| < \epsilon$ then
        stop
```

Algorithm 5: Stationary iteration for solving extended generalized Lyapunov equations from Shank et al. (2015).

In Shank et al. (2015) the Extended Krylov method is used as method to solve the generalized Lyapunov equations that arise in every iteration. A downside of this algorithm is that B_i can become so large that the extended Krylov subspace that is computed by this method requires a huge amount of memory and that the orthonormalization and the solution of projected generalized Lyapunov equation (4.11) takes up the majority of the time. As a solution in this problem, B_i was split up into multiple single vectors $\boldsymbol{b}_1^{(i)}, \dots, \boldsymbol{b}_{2m}^{(i)}$, after which $2m$ generalized Lyapunov equations had to be solved in every iteration.

The advantage of using RAILS instead of EKSM is twofold. Since we only expand with a few vectors in every iteration, the space does not become as large as when using EKSM, and therefore we do not have to split up the problem. Furthermore, RAILS can be restarted, meaning that we can also use V_{i-1} in the stationary iterations as initial guess for V_i. Moreover, since the matrices A and M are the same in every iteration, AV_{i-1}, $V_{i-1}AV_{i-1}^T$, MV_{i-1} and $V_{i-1}MV_{i-1}^T$, do not have to be recomputed. By doing this, we only require a few iterations of RAILS in every stationary iteration to obtain the approximate solution. The restart method from Algorithm 4 can be called after the first solution of the projected generalized Lyapunov equation (4.11) in every stationary iteration to reduced the size of the initial space.

Problem setting 4.2

In this section we discuss the problem setting of the ocean model as described in Section 2.5.2. In Section 4.2.1, 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 4.3.1), comparison with other methods (Section 4.3.2), numerical aspects (Section 4.3.3 and Section 4.3.4) and application in a continuation (Section 4.3.5). In Section 4.3.6, we discuss the performance on an example of an extended Lyapunov equation.

Bifurcation diagram 4.2.1

For the deterministic case, we fix the equatorially symmetric surface forcing as described in (2.10).

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 $n = 6n_xn_yn_z$, 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 $u = 0$.

The bifurcation diagram of the deterministic model for parameters as in Table 2.1 is shown in Figure 4.1a. 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

$$\frac{\partial \Psi}{\partial z} = v \cos\theta, \; -\frac{1}{r_0 \cos\theta}\frac{\partial \Psi}{\partial \theta} = w. \tag{4.14}$$

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.

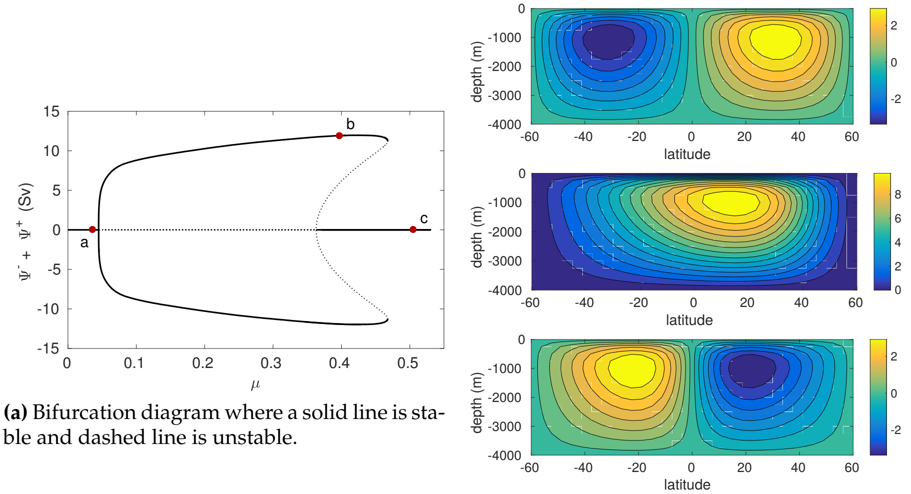
For small values of μ, a unique equatorially anti-symmetric steady MOC exists of which a pattern at location a is shown in Figure 4.1b. This pattern destabilizes at a supercritical pitchfork bifurcation and two asymmetric pole-to-pole solutions appear. An example of the MOC at location b in Figure 4.1b 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 Figure 4.1b) 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.

4.2.2 Stochastic freshwater forcing

The freshwater forcing is chosen as described in Section 2.5.3. In this case, the noise matrix B in (4.4) 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 (2.11)).

4.3 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 (4.9). We use grid sizes of $4 \times n_y \times 16$, where n_y is a varying number of grid points in the meridional direction, 16 is the number of grid points in the vertical direction and 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 (2.11).



(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.

Figure 4.1: (a) Bifurcation diagram of the deterministic equatorially symmetric 2D MOC model, with the forcing as in (2.10). (b) Streamfunction pattern at μ_a, μ_b and μ_c respectively.

For RAILS, we use the algorithm as described in Section 4.1 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 4.3.2), 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 2.5GHz) and each node has 128 GB of memory. Only one core is used in the results below to be able to make fair comparisons.

The results in this section can be reproduced with the C++ and Matlab code at `https://github.com/Sbte/RAILS`.

Comparison with stochastically forced time forward simulation 4.3.1

In climate sciences it is common use empirical orthogonal functions (EOFs) to investigate the variability of a system Dijkstra (2013); Navarra and Simoncini (2010). The EOFs are the eigenvectors of the covariance matrix, and in the context of principal component analysis are also called the principal components.

The EOFs that belong to the largest eigenvalues of the covariance matrix are the ones that can be used to explain a large part of the variance, which is why these are of interest.

A first check of the correctness of the approximate solution of the generalized Lyapunov equations is obtained by comparing the 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 Van der Mheen et al. (2013). The EOFs, which are the eigenvectors of the covariance matrix, are used to get an idea of

This time series (for $n_y = 32$) is plotted in Figure 4.2a 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 Figure 4.2b and Figure 4.2c.

The eigensolutions of the generalized Lyapunov equation, also for $n_y = 32$, are shown in Figure 4.3. Comparing Figure 4.2 with Figure 4.3, 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 4.1. 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 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.

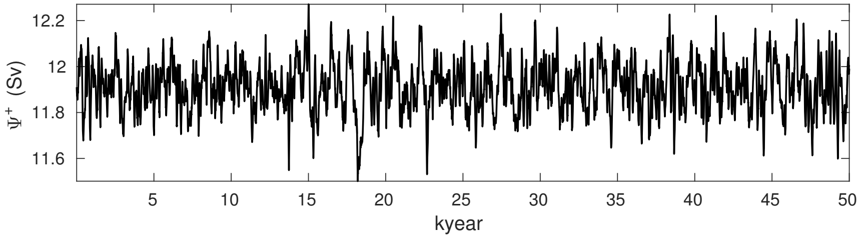
As another check, we use the Bartels–Stewart algorithm (Bartels and Stewart, 1972), which is a dense solver of which the solution time increases with $\mathcal{O}(n^3)$ and the required memory with $\mathcal{O}(n^2)$. The implementation we use is `sb03md` from the SLICOT library (Benner et al., 1999). Results from this method (Table 4.1, also for $n_y = 32$) confirm that our solution method provides correct solutions.

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

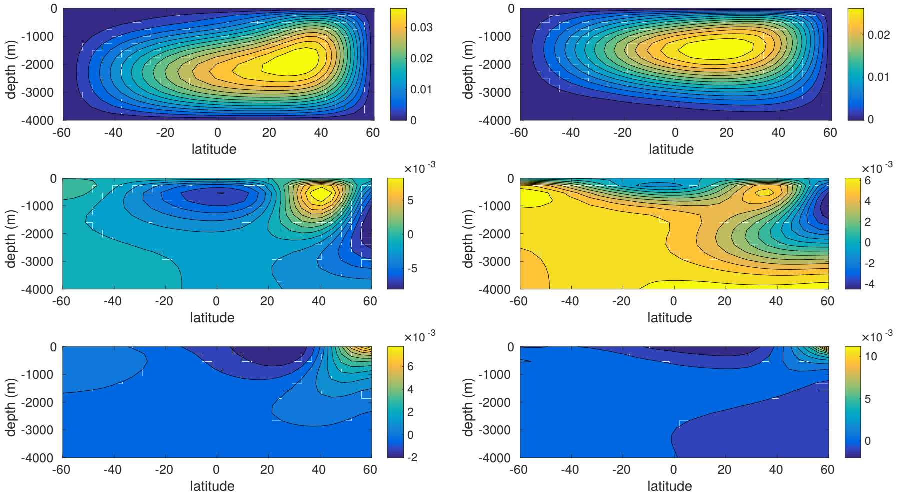
Table 4.1: First four weighted eigenvalues of the covariance matrix. For the RAILS solver $\|R\|_2/\|BB^T\|_2 < 10^{-2}$ was used as stopping criterion.

4.3.2 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) (Simoncini, 2007), Projected Extended Krylov (PEKSM) (Stykel and Simoncini, 2012), Rational Krylov (RKSM) (Druskin and Simoncini, 2011), Tangential Rational Krylov



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



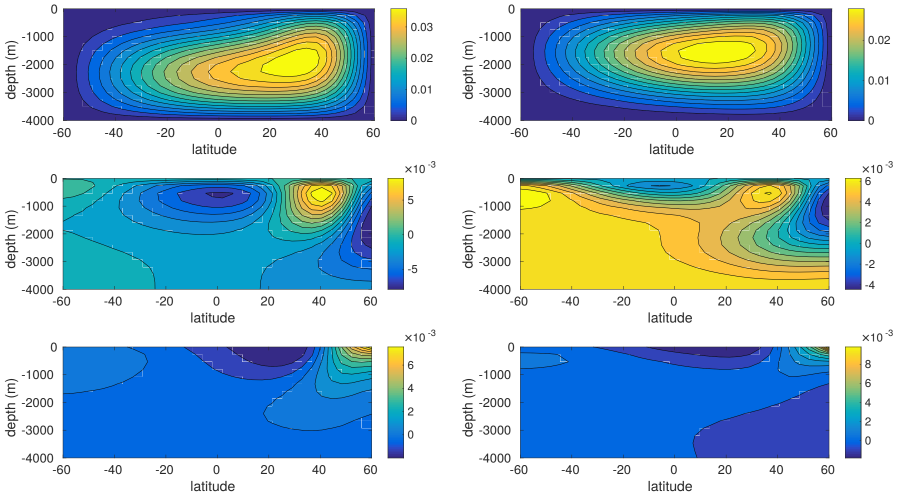
(b) The first EOF.

(c) The second EOF.

Figure 4.2: (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 (2.11). 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.

(TKRSM) (Druskin et al., 2014) and Low-rank ADI (LR-ADI) (Penzl, 1999) methods. For the LR-ADI method, results with the heuristic shifts from Penzl (1999) and self-generating shifts based on a Galerkin projection from Benner et al. (2014) are reported. Implementations of the Extended Krylov and Rational Krylov methods were obtained from the website of Simoncini (Simoncini, 2016), whereas the LR-ADI method from M.E.S.S. was used (Saak et al., 2016).

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



(a) The first EOF.

(b) The second EOF.

Figure 4.3: (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.

experiments with our method, where we expand the search space by $S^{-1}\boldsymbol{r}_i$, where $\boldsymbol{r}_i$ 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 $[\boldsymbol{r}_i, S^{-1}\boldsymbol{r}_i]$, 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 $S\boldsymbol{y} = \boldsymbol{x}$ are implemented by first computing an LU factorization of A using UMFPACK whenever possible (available from `lu` in Matlab), and then solving the system

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

This is similar to what has been used in LR-ADI methods in Freitas et al. (2008). 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 März (1996) for obtaining the spectral projectors that are required and implemented the method according to Stykel and Simoncini (2012). 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^TAV explicitly, and not implicitly as suggested in Stykel and Simoncini (2012) 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 Stykel and Simoncini (2012) also in RAILS, where we chose to expand the basis with $A^{-1}\boldsymbol{r}_i$.

For all methods we use the same stopping criterion, namely that we require the relative residual $\rho = \|R\|_2/\|BB^T\|_2 < \epsilon$, 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 where 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. For Inverse RAILS, we use $S^{-1}B_2$ as initial guess as suggested by Proposition 4.1.2. 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 4.2. Note that B always has n_y columns.

Let us discuss all columns of Table 4.2 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.

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	$6.4 \cdot 10^{-3}$

Table 4.2: 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 number of iterations, MVPs are the number of matrix-vector products, IMVPs are the number 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.

The next column contains the number 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 number 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.

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 4.2, 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 Figure 4.4, 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.

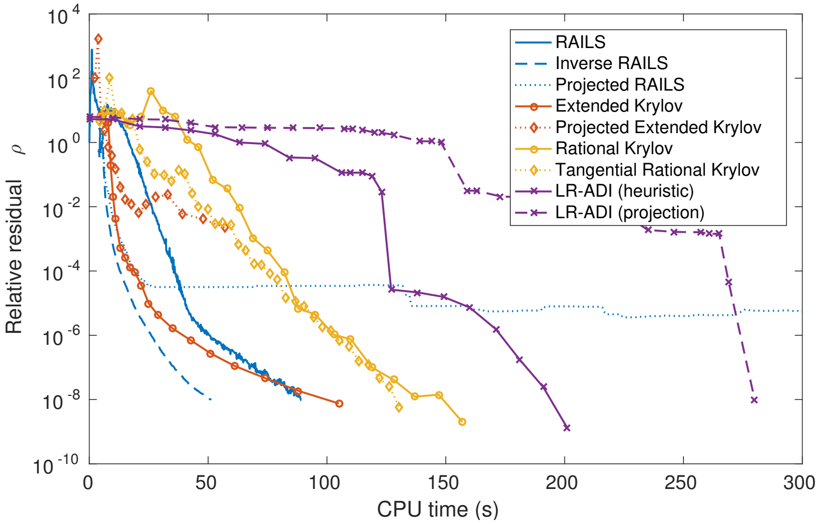


Figure 4.4: Convergence history of the relative residual ρ of all the different methods against CPU time. RAILS was restarted after 15 iterations with a tolerance of 10^{-12} for the eigenpairs that were retained and expanded with 10 vectors per iteration.

What we see in Figure 4.4 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.

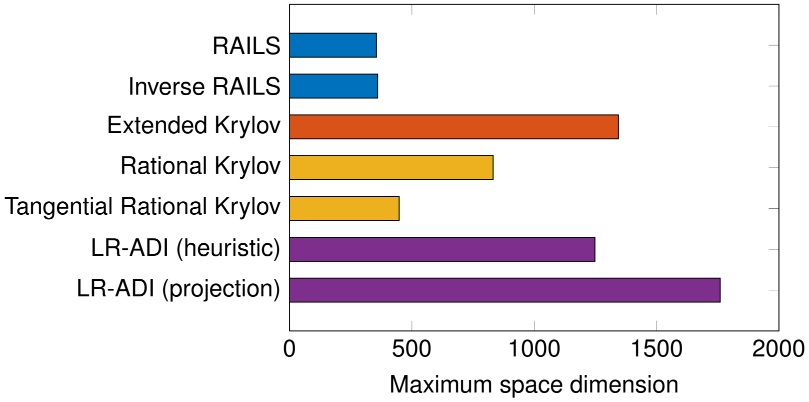


Figure 4.5: Memory usage of the different methods as described in Figure 4.4 in terms of maximum space dimension. Both projected methods were left out of this figure since they did not converge.

In Figure 4.5 we show the memory usage of the methods in Figure 4.4. 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 number of vectors that is used to expand the space and due to the restart strategy.

4.3.3 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 number 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 $\hat{B} = B \cdot \mathbf{1}$, with $\mathbf{1}$ being a vector of ones, as right-hand side, so $\hat{B}$ is a single vector containing 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 4.3 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 4.3.

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

Table 4.3: 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 number of iterations, MVPs are the number 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.

We are of course also interested in the increase of the solution time. There are five factors that influence this: the length of the vectors that span our basis, the number of vectors that span our basis, the number 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 number 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 4.3. 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 number of iterations. In Table 4.3 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 4.4 we see that the number 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 number 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.

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 4.4: Performance of RAILS for different grid sizes with B as the original stochastic forcing. 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 number of iterations, MVPs are the number 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 50 iterations with a tolerance of 10^{-5} for the eigenpairs that were retained.

4.3.4 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 $\hat{B}\hat{B}^T$

can be written as

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

where $\hat{B}_i$ is the ith column of $\hat{B}$. Since the rank of the approximate solution is highly dependent on the number 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 number of parts that resulted in the least amount of solution time. The optimal number 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 number 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 4.5.

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

Table 4.5: Comparison of different Lyapunov solvers for different grid sizes with $\hat{B} = \text{diag}(B \cdot \mathbf{1})$. 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 number of iterations, MVPs are the number of matrix-vector products, IMVPs are the number 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.

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.

4.3.5 Continuation

Now that we have shown the performance of RAILS compared to other methods, we show how it can be useful in the context of continuation. The main idea is that since RAILS can be restarted and since we can choose the number of vectors that we use to expand the space, we can easily start from the low-rank solution of another generalized Lyapunov equation, in our case the solution that was determined in the previous continuation step. We refer to this method as Recycling RAILS. We use the settings from Section 4.3.2, starting at point b from the bifurcation diagram in Figure 4.1, and moving towards the bifurcation with constant step size $\mathrm{d}s = 0.05$. We do this for 20 continuation steps on the problem of size $4 \times 32 \times 16$.

		RAILS		Recycling RAILS	
Step	Par	Its	$t_s(s)$	Its	$t_s(s)$
1	0.403	210	7.3	210	7.0
2	0.406	242	8.4	49	2.0
3	0.410	245	8.6	35	1.3
4	0.413	248	8.8	39	1.5
5	0.415	250	9.1	39	1.5
10	0.428	254	9.1	36	1.3
15	0.439	285	10.1	50	2.2
20	0.448	304	11.1	54	2.3

Table 4.6: Performance of RAILS during 20 steps of the continuation process with fixed step size $\mathrm{d}s = 0.05$. Here Recycling RAILS is RAILS restarted from the solution of the generalized Lyapunov equation at the previous continuation step. Par is the actual parameter value, Its is the number of iterations, 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 results are shown in Table 4.6. It is clear that reusing the solution from the previous continuation step, which we do in Recycling RAILS, is of great benefit. Both the number of iterations and the computational time are reduced by roughly a factor of 6. The local variations in the number of iterations can mostly be explained by the random initial guess that is used for computation of the eigenvectors of the residual. We also see a global pattern of the generalized Lyapunov equations becoming harder to solve when getting closer to the saddle-node bifurcation, which is expected. This does not seem to affect the efficiency of the recycling method.

Extended Lyapunov equations 4.3.6

Since usage of CAM noise for the MOC problem that is discussed in this section has not yet been investigated, we will not look at the performance of RAILS on a MOC related problem. Instead, we will look at Example 2 from Shank et al. (2015), which is described in more detail in Damm (2008). The example consists of a central finite difference discretization of the 2D convection-diffusion operator $L(\boldsymbol{x}) = \Delta\boldsymbol{x} - \boldsymbol{x}_y$ on $\Omega = (0,1)^2$ with Robin boundary conditions $\boldsymbol{n} \cdot \nabla\boldsymbol{x} = \frac{1}{2}u_j(\boldsymbol{x} - 1)$ on an increasing number of sides of the domain, and Dirichlet boundary conditions $\boldsymbol{x} = 0$ on the rest of the boundary. Here the control variable u_j represents the heat transfer coefficient on boundary j. The number of Robin boundary conditions that are imposed determines the number of extra terms of the extended Lyapunov equation. The matrices in (4.13) are given by

$$\begin{aligned} A &= I \otimes D + D \otimes I - I \otimes G + \frac{2}{h}\left(\sum_{j=1}^{m} N_j\right), \\ N_1 &= \frac{1}{2h}(E_1 \otimes I), \qquad N_2 = \frac{1}{2h}(E_n \otimes I), \\ B_1 &= -\frac{1}{2h}(E_1 \otimes \mathbf{1}), \quad B_2 = -\frac{1}{2h}(E_n \otimes \mathbf{1}), \\ B &= [B_1, \ldots, B_m], \qquad M = I, \end{aligned}$$

where $D \in \mathbb{R}^{n\times n}$ is the central finite difference discretization of the 1D Laplace operator, $G \in \mathbb{R}^{n\times n}$ is the central finite difference discretization of the derivative, $E_j = \boldsymbol{e}_j\boldsymbol{e}_j^T$ with canonical unit vector $\boldsymbol{e}_j \in \mathbb{R}^n$, $\mathbf{1} \in \mathbb{R}^n$ is a vector of all ones, $n = 70$, and $h = 1/(n+1)$. The solution C may be interpreted as a controllability Gramian of the system Damm (2008). The implementation of this example which was used in Shank et al. (2015), along with the implementation of the stationary iterations, can be found at Simoncini (2016). We found that the example was not implemented correctly, however, and therefore the results we find here may differ from the results reported in Shank et al. (2015).

In Table 4.7 and Table 4.8 we show the performance of RAILS when expanded with $\boldsymbol{r}_i$ (RAILS), $A^{-1}\boldsymbol{r}_i$ (Inverse RAILS) and $[\boldsymbol{r}_i, A^{-1}\boldsymbol{r}_i]$ (Extended RAILS), EKSM, and EKSM where B_i is split into single vectors. The first table contains results from the case with one Robin boundary condition, the second table contains results with two Robin boundary conditions. In Table 4.7, we see that Extended RAILS and Inverse RAILS perform very well in terms of matrix-vector products, the number of linear system solves and the required CPU time. The rank of the final solution of all methods is comparable. The reason why the number of matrix-vector products and the number of linear system solves is so much smaller than those of EKSM is that we restart from the solution of the previous stationary iteration.

Method	Rank	MVPs	IMVPs	t_s (s)	ρ
RAILS	66	1084	0	58.0	$8.1 \cdot 10^{-9}$
Inverse RAILS	68	264	264	5.3	$6.4 \cdot 10^{-9}$
Extended RAILS	68	274	137	3.8	$6.4 \cdot 10^{-9}$
EKSM	69	2461	2656	67.6	$6.1 \cdot 10^{-9}$
EKSM + splitting	69	1992	2186	8.7	$7.7 \cdot 10^{-9}$

Table 4.7: Comparison of RAILS and EKSM with and without splitting of B_i on the example described in this section with one Robin boundary condition. Rank is the rank of the final approximate solution, MVPs are the number of matrix-vector products, IMVPs are the number of inverse matrix-vector products and t_s is the time required for solution of the extended Lyapunov equation. For all methods the stopping criterion is a relative residual of 10^{-8}. The number of eigenvectors of the residual in RAILS was set to be the same as the number of columns of the matrix B_i. RAILS was restarted after 10 iterations.

In Table 4.8, we see again that Extended RAILS performs well in all categories. Because we use two Robin boundary conditions in this case, an extra extended term is added, and therefore also the number of columns of B_i of the stationary iteration is larger. As we also noticed for the MOC problem, this is something for which RAILS performs increasingly well compared to other methods. We notice that even though the number of linear system solves that is required for Extended RAILS is more than a factor 20 lower than that of EKSM with splitting, it is only a factor 3 faster. This is because the problem that is solved here is relatively easy, meaning that solving a linear system is relatively cheap. Therefore all instances of RAILS are actually dominated by the cost of computing the eigenvectors. We expect that RAILS performs even better for more complex systems like an ocean-climate model with CAM noise.

4.4 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 Kuehn (2011, 2015) to larger systems of SPDEs with algebraic constraints and exploits the structure of the generalized Lyapunov equation in the computa-

Method	Rank	MVPs	IMVPs	t_s (s)	ρ
RAILS	123	1793	0	246.3	$9.6 \cdot 10^{-9}$
Inverse RAILS	123	416	416	9.2	$9.7 \cdot 10^{-9}$
Extended RAILS	123	464	232	8.0	$9.6 \cdot 10^{-9}$
EKSM	123	4429	4865	197.6	$9.6 \cdot 10^{-9}$
EKSM + splitting	134	4652	5090	23.5	$8.7 \cdot 10^{-9}$

Table 4.8: Comparison of RAILS and EKSM with and without splitting of B_i on the example described in this section with two Robin boundary conditions. Rank is the rank of the final approximate solution, MVPs are the number of matrix-vector products, IMVPs are the number of inverse matrix-vector products and t_s is the time required for solution of the extended Lyapunov equation. For all methods the stopping criterion is a relative residual of 10^{-8}. The number of eigenvectors of the residual in RAILS was set to be the same as the number of columns of the matrix B_i. RAILS was restarted after 10 iterations.

tions. This approach fits nicely within purely deterministic numerical bifurcation computations (Keller, 1977) and methods to tackle the SPDEs directly (Sapsis and Lermusiaux, 2009). 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 (Simoncini, 2007; Druskin and Simoncini, 2011; Stykel and Simoncini, 2012; Druskin et al., 2014; Kleinman, 1968; Penzl, 1999) 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 (Dijkstra et al., 2014). 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 (Kuehn, 2011). 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 (Van der Mheen et al., 2013).

We have shown that for these types of problems, RAILS performs especially well since it can be restarted using the approximate solution of the previous continuation step, which results in rapid convergence. We also applied RAILS in a stationary iteration to solve extended generalized Lyapunov equations. This proved to be very efficient and may be used when CAM noise is applied instead of the additive noise that we used for the MOC.

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 Dijkstra et al. (2008). Note that for studying the effect of wind-stress noise, a full three-dimensional model, as developed in De Niet et al. (2007) 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.

TRANSITION PROBABILITIES

In this chapter we give an overview of the methods that are available for studying transition probabilities. Many different kinds of methods exist, but they all serve a different purpose. Therefore, we also give a quantitative comparison of methods that may seem suited for computing actual transition probabilities. To our knowledge, such an overview does not yet exist in the literature.

In Section 5.1, we first discuss our definition of a transition probability. In Section 5.2, we then discuss the Eyring–Kramers formula, which can be used to analytically study transitions in gradient systems. We then make a detour to covariance ellipsoids in Section 5.3, which may be used to study local variability around a steady state, and to most probable transition paths in Section 5.4. After this we discuss how to actually compute transition probabilities in Section 5.5. In this section we also give a comparison between all the different methods. We then discuss what method we will actually use for computations on the MOC based on these results in Section 5.6.

Definition 5.1

Take a stochastic differential-algebraic equation (SDAE) of the form

$$M(\boldsymbol{p})\,\mathrm{d}\boldsymbol{X}_t = F(\boldsymbol{X}_t;\boldsymbol{p})\,\mathrm{d}t + g(\boldsymbol{X}_t;\boldsymbol{p})\,\mathrm{d}\boldsymbol{W}_t, \tag{5.1}$$

as defined in Section 2.4.2.

We define the *transition probability* as the probability that we go from a neighborhood A near a deterministic steady state $\bar{\boldsymbol{x}}_A$ to a neighborhood B near a deterministic steady state $\bar{\boldsymbol{x}}_B$ within time T. It is important to know that there are many similar quantities that are related, but may not lead to the computation of transition probabilities as we defined them here. In Rolland and Simonnet (2015), the crossing probability is defined as the probability that a trajectory starting from some initial condition $\boldsymbol{X}_0$ reaches a set B before some set A with $A \cap B = \emptyset$. This is dependent on $\boldsymbol{X}_0$ instead of some maximum time T, and in our case, where we would choose $\boldsymbol{X}_0 \in A$, the crossing probability would be 0. Another quantity that is mentioned often (Rolland et al., 2015) is the transition rate, which is the probability that in a unit time, a transition occurs.

Something that makes computing transition probabilities for ocean-climate models more difficult, is that we generally do not have a gradient system, which, in a stochastic sense, is an overdamped Langevin equation of the form

$$M(\boldsymbol{p})\,\mathrm{d}\boldsymbol{X}_t = -\nabla V(\boldsymbol{X}_t;\boldsymbol{p})\,\mathrm{d}t + g(\boldsymbol{X}_t;\boldsymbol{p})\,\mathrm{d}\boldsymbol{W}_t,$$

with a potential $V : \mathbb{R}^n \to \mathbb{R}$. Note that a system $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$ with $F : \mathbb{R}^3 \to \mathbb{R}^3$ is a gradient system if $\nabla \times F = 0$. For a gradient system, and under the assumption of small noise, the Eyring–Kramers formula can be applied (Hänggi et al., 1990), which is something that is often exploited, but not something we can use due to the nature of our problem. For non-gradient systems, the Freidlin–Wentzell theorem in large deviations theory (Freidlin and Wentzell, 1998) and a related generalization of the Eyring–Kramers formula exist (Bouchet and Reygner, 2016), but this is much harder, if not impossible, to compute.

5.2 The Eyring–Kramers formula

Even though we can not apply the Eyring–Kramers formula directly, there is much that we can learn from looking at it. Take an SDE of the form

$$\mathrm{d}\boldsymbol{X}_t = -\nabla V(\boldsymbol{X}_t)\,\mathrm{d}t + \sqrt{2\epsilon}\,\mathrm{d}\boldsymbol{W}_t, \tag{5.2}$$

with $\epsilon > 0$, which originally was a temperature parameter. Say that in between the two steady states $\bar{\boldsymbol{x}}_A$ and $\bar{\boldsymbol{x}}_B$ that we are interested in, we also have an unstable steady state $\bar{\boldsymbol{x}}_C$. The Eyring–Kramers formula (Eyring, 1935; Kramers, 1940) is then given by

$$\mathbb{E}[\tau^{\epsilon}_{\bar{\boldsymbol{x}}_A \to \bar{\boldsymbol{x}}_B}] \underset{\epsilon \downarrow 0}{\simeq} \frac{2\pi}{-\lambda_C} \sqrt{\frac{|\det(\nabla^2 V(\bar{\boldsymbol{x}}_C))|}{\det(\nabla^2 V(\bar{\boldsymbol{x}}_A))}}\, e^{\frac{V(\bar{\boldsymbol{x}}_C) - V(\bar{\boldsymbol{x}}_A)}{\epsilon}}, \tag{5.3}$$

where λ_C is the single negative eigenvalue of the Hessian matrix $\nabla^2 V(\bar{\boldsymbol{x}}_C)$ and $\tau^{\epsilon}_{\bar{\boldsymbol{x}}_A \to \bar{\boldsymbol{x}}_B}$ is the first passage time. The first passage time is the time it takes to reach $\bar{\boldsymbol{x}}_B$ from $\bar{\boldsymbol{x}}_A$ for the first time. The quantity $\tau_{\text{MFPT}} = \mathbb{E}[\tau^{\epsilon}_{\bar{\boldsymbol{x}}_A \to \bar{\boldsymbol{x}}_B}]$ is the mean first passage time.

Double well potential 5.2.1

An example of (5.2) which is often used is the double well potential, which has two local minima with a saddle point in between. The one dimensional symmetric double well potential is given by

$$V(x) = \frac{1}{4}x^4 - \frac{1}{2}x^2,$$

which is displayed in Figure 5.1. We can substitute this in (5.2), in which case the mean first passage time for small noise can be obtained from (5.3). It tells us that

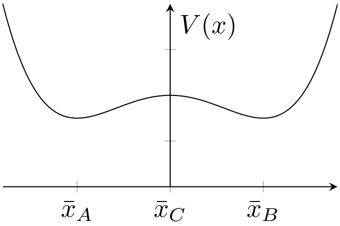


Figure 5.1: Double well potential

$$\tau_{\text{MFPT}} \simeq \frac{2\pi}{-V''(0)} e^{\frac{V(0)-V(-1)}{\epsilon}},$$

since $\bar{x}_A = -1$ and $\bar{x}_C = 0$.

Computing the transition probability 5.2.2

The transition rate η of (5.1) is the probability that a transition occurs in a unit time. This means that in an infinitesimal interval of time $\mathrm{d}t$, the probability of having a transition is $\eta \, \mathrm{d}t$, which in turn means that the probability of not having a transition is $1 - \eta \, \mathrm{d}t$. If, for some time T, we take $\mathrm{d}t = \lim_{N\to\infty} \frac{T}{N}$ we have that the probability that in a time T no transitions occur is given by

$$\mathbb{P}(\boldsymbol{X}_t \notin B \mid 0 \leq t \leq T) = \lim_{N\to\infty} \left(1 - \eta \frac{T}{N}\right)^N = e^{-\eta T},$$

assuming that a transition itself happens within time $\mathrm{d}t$. This means that the probability that transitions do occur in a time T is given by

$$\mathbb{P}(\boldsymbol{X}_t \in B \mid 0 \leq t \leq T) = 1 - e^{-\eta T}.$$

Note that this is the cumulative distribution function of the exponential distribution. Since the occurence of transitions in a time T is equivalent to saying that the first transition has occurred before time T, it is easy to see that the relation between the transition rate and the mean first passage time is given by

$$\tau_{\mathrm{MFPT}} = \frac{1}{\eta},$$

which means that we could also write

$$\mathbb{P}(\boldsymbol{X}_t \in B \mid 0 \leq t \leq T) = 1 - e^{-\frac{T}{\tau_{\mathrm{MFPT}}}}. \tag{5.4}$$

5.3 Covariance ellipsoids

In Kuehn (2012) it was suggested that one might get an insight into transition probabilities by looking at the spatial distance between high (n)-dimensional ellipsoidal confidence regions. These confidence ellipsoids are determined by the probability density functions (PDFs) and indicate, with a certain confidence, where a state may be during a transient computation. The stationary PDF of the approximating n-dimensional Ornstein–Uhlenbeck process around a steady state $\bar{\boldsymbol{x}}$ is described as as (Gardiner, 1985; Cowan, 1998)

$$p(\boldsymbol{x}; \bar{\boldsymbol{x}}) = \frac{1}{(2\pi)^{\frac{n}{2}} \,|\, C \,|^{\frac{1}{2}}} e^{-\frac{1}{2} Q(\boldsymbol{x}; \bar{\boldsymbol{x}})},$$

where

$$Q(\boldsymbol{x}; \bar{\boldsymbol{x}}) = (\boldsymbol{x} - \bar{\boldsymbol{x}})^T C^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}}),$$

and C is the covariance matrix at $\bar{\boldsymbol{x}}$. The associated ellipsoid can be described as

$$E = \{\boldsymbol{x} \in \mathbb{R}^n : Q(\boldsymbol{x}; \bar{\boldsymbol{x}}) \leq Q_\alpha\}, \tag{5.5}$$

where Q_α is the quantile of confidence level $1 - \alpha$ of the χ^2-distribution (Cowan, 1998). Note that (5.5) is properly defined in case C is invertible. However, since we compute low-rank approximations of C as discussed in the previous chapter, this is never the case. Instead we may use an alternative formulation of the ellipsoid

$$E = \left\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{v}^T \boldsymbol{x} \leq \boldsymbol{v}^T \bar{\boldsymbol{x}} + \sqrt{\boldsymbol{v}^T \tilde{C} \boldsymbol{v}} \quad \forall\, \boldsymbol{v} \in \mathbb{R}^n\right\}, \tag{5.6}$$

where $\tilde{C} = Q_\alpha C$ is the positive semi-definite shape matrix associated with the confidence ellipsoid. Note that such confidence regions are limited to representing the PDF under the assumption of linearized dynamics according to

$$M(\boldsymbol{p}) \,\mathrm{d}\boldsymbol{X}_t = A(\bar{\boldsymbol{x}}; \boldsymbol{p}) \boldsymbol{X}_t \,\mathrm{d}t + B(\bar{\boldsymbol{x}}; \boldsymbol{p}) \,\mathrm{d}\boldsymbol{W}_t.$$

The distance between ellipsoids associated with equilibria at different branches, but with the same parameter value, provides information about the relative frequency of noise induced transitions (Kuehn, 2012). Hence, calculating the distance between ellipsoids can be worthwhile when transient computations are expensive. To determine the distance d between ellipsoids, a convex minimization problem of the form

$$-d = \min_{\|\boldsymbol{v}\|_2=1} \left(-\boldsymbol{v}^T \bar{\boldsymbol{x}}_A + \sqrt{\boldsymbol{v}^T \tilde{C}_1 \boldsymbol{v}} + \boldsymbol{v}^T \bar{\boldsymbol{x}}_B + \sqrt{\boldsymbol{v}^T \tilde{C}_2 \boldsymbol{v}} \right)$$

is solved, where $\bar{\boldsymbol{x}}_A$ is an equilibrium at one branch and $\bar{\boldsymbol{x}}_B$ the equilibrium at the other branch for the same parameter value μ. $\tilde{C}_1 = Q_\alpha C_1$ and $\tilde{C}_2 = Q_\alpha C_2$ are the respective shape matrices.

In Mulder et al. (2018) we investigated patterns of transition behavior in marine ice sheet instability problems. Here we actually saw a good correspondence with results that were obtained with transient computations.

Example 5.3.1

To get a feeling for these covariance ellipsoids, we produce results with the two-dimensional double well potential defined by

$$V(x, y; \mu) = \frac{1}{4}x^4 - \frac{1-\mu}{2}x^2 + y^2,$$

where we added a parameter μ in which we will perform a continuation. The corresponding SDE that we solve is given by

$$\mathrm{d}\boldsymbol{X}_t = -\nabla V(\boldsymbol{X}_t; \mu)\,\mathrm{d}t + g(\boldsymbol{X}_t; \mu)\,\mathrm{d}\boldsymbol{W}_t, \tag{5.7}$$

where we take $g(\boldsymbol{X}_t; \mu) = 0.4$. The bifurcation diagram of the deterministic part of the SDE, together with the ellipsoids in parameter steps of 0.2 are shown in Figure 5.2. For the ellipsoids, we take Q_α corresponding to the confidence level $1 - \alpha = 0.95$.

In Figure 5.3 we show ten transient simulations up to $T = 10$ for $\mu = 2$, which show good correspondence with the ellipsoids. From the linearized dynamics around the steady states, which the ellipsoids describe, we expect 95% of a transient to be within the ellipsoids. Indeed we see in the figure that most trajectories spend the majority of the time inside the ellipsoids. However, the trajectory that actually transitions to the other steady state spends a lot of time outside either of the ellipsoids. In this figure, the actual amount of time spent inside the ellipsoids is 92% of the total time. If we take 1000 samples instead of just 10, we see that this percentage converges to 90%. This makes sense since the covariance matrix only describes the local behavior around the steady state, while the transitions happen on a global scale.

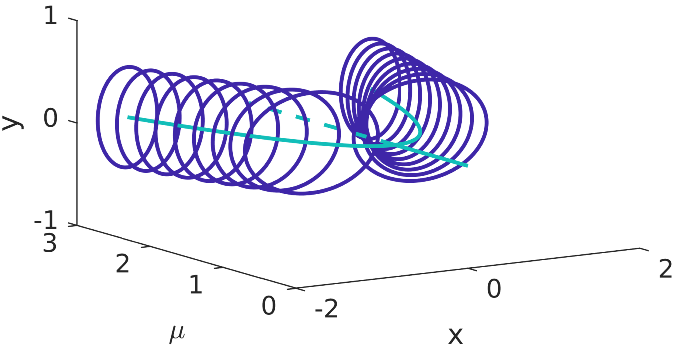


Figure 5.2: Bifurcation diagram of (5.7) with ellipsoids in parameter steps of 0.2.

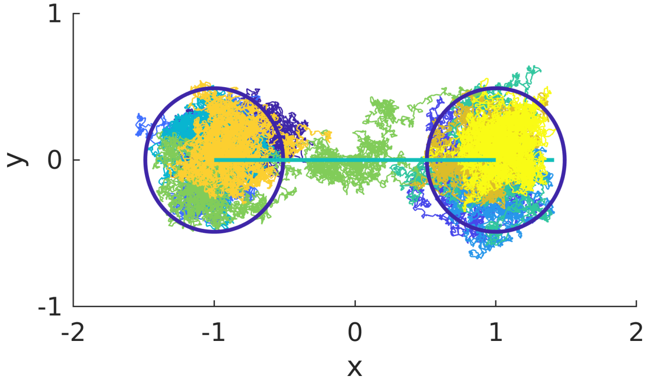


Figure 5.3: Ten transients of (5.7) up to $T = 10$ at $\mu = 2$ with corresponding ellipsoids. Each color represents a different transient.

5.4 Most probable transition trajectories

Another quantity of interest when investigating transition probabilities is the most probable transition trajectory. Other terms that are used are path of maximum likelihood, minimum action path, minimum energy path or instanton, depending on the application and the method that is used to compute them. Where Eyring–Kramers tells us something about the transition rate, Freidlin–Wentzell (Freidlin and Wentzell, 1998) large deviations theory gives us information about the most probable transition trajectories. For gradient systems, methods for computing these trajectories include the nudged elastic band method (Henkelman et al., 2000; Henkelman and Jónsson, 2000) or the string method (E et al., 2002, 2007). Methods which also work for more general systems are the minimum action method and its geometric and adaptive variants (E et al., 2004; Vanden-Eijnden and Heymann, 2008; Zhou et al., 2008; Grafke et al., 2017).

In Bouchet et al. (2014); Laurie and Bouchet (2015) the two-dimensional barotropic quasi-geostrophic (QG) equations, which are approximations of the shallow water equations for small Rossby numbers, were studied in the context of rare transitions. The minimum action method was applied to obtain the most probable transition trajectories. This gives us an indication that this may also work for our 2D MOC problem, since from our experience, investigating transitions in the QG model is harder than investigating transitions in the 2D MOC.

It may be possible to simplify computations of most probable transition trajectories by minimizing only in the space spanned by the low-rank solutions of the Lyapunov equations as described in the previous chapter. Ideally, one could also exploit the knowledge about these paths to compute actual transition probabilities, for instance by using the location of the path to only sample near that path. One would have to investigate how this can be done without losing information about the probabilities. We will, however, not go further into this direction in this thesis.

Computing transition probabilities 5.5

In this section we investigate methods that actually compute transition probabilities. We start with the most naive methods, then discuss more advanced methods, and finally give a quantitative comparison between all of these methods.

Direct sampling 5.5.1

The most naive method of computing a transition probability as defined above is by computing samples using the Euler–Maruyama method (2.7) or some other stochastic time stepping method like the stochastic theta method (2.8) until the end time T, and then counting the number of times a transition occurred. This is just a standard Monte Carlo method, which converges with $\mathcal{O}((\alpha N)^{-1/2})$, where α is the transition probability and N is the number of samples (Rubino and Tuffin, 2009). This means that especially for small probabilities, this method is very expensive, and for high-dimensional problems unfeasible. The method is described in Algorithm 6.

```
input:   $F(\boldsymbol{x}), g(\boldsymbol{x})$   Functions as described in (5.1) with $M = I$.
         $\Delta t$                          Time step.
         $\boldsymbol{x}_0$                  Starting point.
         $T$                                 End time.
         $N$                                 Number of samples.
output:  $\alpha$                            Transition probability.

$n_t = 0$
for $i = 1, \dots, N$ do
    $\boldsymbol{x} = \boldsymbol{x}_0$
    for $t = \Delta t, 2\Delta t, \dots, T$ do
        $\boldsymbol{x} = \boldsymbol{x} + F(\boldsymbol{x})\Delta t + \sqrt{\Delta t} g(\boldsymbol{x})\Delta \boldsymbol{W}$
        if $\boldsymbol{x} \in B$ then
            $n_t = n_t + 1$
$\hat{\alpha}_N = n_t / N$
```

Algorithm 6: Direct sampling method for computing transition probabilities.

5.5.2 Direct sampling of the mean first passage time

Instead of computing the transition probability directly, one might also first compute the mean first passage time in a naive way, and then compute the transition probability using (5.4). This method is described in Algorithm 7. Again, the Euler–Maruyama scheme can be replaced by another stochastic time stepper like the stochastic theta method.

5.5.3 Adaptive multilevel splitting

There exist more efficient ways of computing the mean first passage time that are based on the idea that there are more trajectories that leave A that come back to A than there are trajectories that reach B. A method that makes use of this concept is called the Adaptive Multilevel Splitting method (AMS) (Cérou and Guyader, 2007; Rolland and Simonnet, 2015). It was inspired by multilevel splitting methods which date back to Kahn and Harris (1951) and Rosenbluth and Rosenbluth (1955). The multilevel splitting methods are all based on the same idea of discarding trajectories that are unlikely to reach B and splitting (or branching) from trajectories that are more likely to reach B. An additional advantage of these methods is that no assumptions have to be made on the amplitude and color of the noise, unlike for theoretical values obtained through Eyring–Kramers or Freidlin–Wentzell (Rolland and Simonnet, 2015). We will now explain in more detail how it works.

```
input:   $F(\boldsymbol{x}), g(\boldsymbol{x})$   Functions as described in (5.1) with $M = I$.
         $\Delta t$                               Time step.
         $\boldsymbol{x}_0$                       Starting point.
         $N$                                      Number of samples.
output:  $\tau$                                   Mean first passage time.

$\hat{\tau} = 0$
for $i = 1, \dots, N$ do
    $\boldsymbol{x} = \boldsymbol{x}_0$
    for $t = \Delta t, 2\Delta t, \dots$ do
        $\boldsymbol{x} = \boldsymbol{x} + F(\boldsymbol{x})\Delta t + \sqrt{\Delta t} g(\boldsymbol{x})\Delta \boldsymbol{W}$
        if $\boldsymbol{x} \in B$ then
            $\hat{\tau} = \hat{\tau} + t/N$
            break
```

Algorithm 7: Direct sampling method for computing the mean first passage time.

We start with the neighborhood A near a deterministic steady state $\bar{\boldsymbol{x}}_A$ and the neighborhood B near a deterministic steady state $\bar{\boldsymbol{x}}_B$ as defined in Section 5.1. Now we define a surface C that encloses A. This surface C can be chosen arbitrarily, but the speed of the algorithm greatly depends on the choice of C (Rolland and Simonnet, 2015). We also define a so-called *reaction coordinate*, which is a smooth one-dimensional function

$$\phi \,:\, \mathbb{R}^n \to \mathbb{R}$$

that defines how close $\boldsymbol{x}$ is to B. In this thesis, we assume that the reaction coordinate should satisfy

$$|\nabla\phi(\boldsymbol{x})| \neq 0, \ \forall x \in \mathbb{R}^n \backslash (A \cup B), \tag{5.8a}$$

$$A \subset \{\boldsymbol{x} \in \mathbb{R}^n \,:\, \phi(\boldsymbol{x}) < z_{\min}\}, \tag{5.8b}$$

$$B \subset \{\boldsymbol{x} \in \mathbb{R}^n \,:\, \phi(\boldsymbol{x}) > z_{\max}\}, \tag{5.8c}$$

$$C = \{\boldsymbol{x} \in \mathbb{R}^n \,:\, \phi(\boldsymbol{x}) = z_{\min}\}, \tag{5.8d}$$

where $z_{\min} < z_{\max}$ are two given real numbers. These properties were slightly adapted from Cérou et al. (2011). For multi-dimensional problems, we propose some additional properties to make sure the method actually converges towards B, and not somewhere completely different which has the same value of ϕ. These properties are

$$\{\boldsymbol{x} \in \mathbb{R}^n \,:\, \phi(\boldsymbol{x}) = \inf\{\phi(\boldsymbol{y}) \,:\, \boldsymbol{y} \in \mathbb{R}^n\}\} \subset A,$$
$$\{\boldsymbol{x} \in \mathbb{R}^n \,:\, \phi(\boldsymbol{x}) = \sup\{\phi(\boldsymbol{y}) \,:\, \boldsymbol{y} \in \mathbb{R}^n\}\} \subset B.$$

Since the gradient is not allowed to be zero outside of A and B, this means that the reaction coordinate is always increasing towards B and decreasing towards A.

We now explain step by step how the Adaptive Multilevel Splitting method works.

1. First generate N independent trajectories $(\boldsymbol{x})^{(1)}, \ldots, (\boldsymbol{x})^{(N)}$, that start in A, pass through C, and then end up in either A or B. Here $(\boldsymbol{x})^{(i)} = (\boldsymbol{x}_0^{(i)}, \boldsymbol{x}_1^{(i)}, \ldots, \boldsymbol{x}_{M_i-1}^{(i)})$ is a trajectory of length M_i. From this we can see how important the choice of C is. The further away from A, the longer trajectories take to reach C, so the longer this step takes.

2. Each trajectory $(\boldsymbol{x})^{(i)}$, $i = 1, \ldots, N$, has a certain maximum value of the reaction coordinate (or maximum distance from A), which we define to be Q_i. Set $k = 1$, $w_0 = 1$.

3. Take $L = \{j \ : \ Q_j = \inf\{Q_i \ : \ i \in \{1, \ldots, N\}\}\}$, which are the indices for which the maximum value of the reaction coordinate is minimal, and take $\ell_k = \mathrm{card}(L)$ to be the number of elements in L. Since the trajectories $\{(\boldsymbol{x})^{(j)} \ : \ j \in L\}$ have the smallest value of Q_i, all other trajectories have some j for which $\phi(\boldsymbol{x}_j) > Q_l$ for all $l \in L$.

4. Set $w_k = \left(1 - \frac{\ell_k}{N}\right) w_{k-1}$. For all $l \in L$, repeat steps 5-7.

    5. Select a random trajectory $(\boldsymbol{x})^{(r)}$ with r from $\{1, \ldots, N\} \backslash L$, and set $(\tilde{\boldsymbol{x}})^{(l)} = (\boldsymbol{x}_0^{(r)}, \ldots, \boldsymbol{x}_{j_{\min}}^{(r)})$, where $j_{\min}$ is the smallest value for which $\phi(\boldsymbol{x}_{j_{\min}}^{(r)}) \geq Q_l$.

    6. Generate the rest of the trajectory starting from $\boldsymbol{x}_{j_{\min}}^{(r)}$ until again you reach either A or B. This trajectory has a new maximum value of the reaction coordinate $\tilde{Q}_l$, which is always greater than or equal to Q_l. Note that this is the branching or splitting which gave the method its name.

    7. Set $(\boldsymbol{x})^{(l)} = (\tilde{\boldsymbol{x}})^{(l)}$ and $Q_l = \tilde{Q}_l$.

8. Repeat steps 3-7 with $k = k + 1$ until $Q_i \geq z_{\max}$, $\forall\, i = 1, \ldots, N$.

The weights w_i that are computed in every step represent the probability of a trajectory reaching iteration $i + 1$. So say we start with 100 trajectories, and we have 2 trajectories for which the maximum value of the reaction coordinate is minimal. Since these two trajectories are eliminated, the probability of a trajectory reaching iteration 2 is $1 - 2/100 = 98/100$. We repeat this process, multiplying the probabilities that we find in every step. This gives us an unbiased estimator of the probability of observing a reactive trajectory

$$\hat{\alpha}_N = \frac{N_B w_k}{N} = \frac{N_B}{N} \prod_{i=0}^{k} \left(1 - \frac{\ell_i}{N}\right), \tag{5.9}$$

where k is the number of iterations it took for all trajectories to converge, and N_B is the number of trajectories that reached B (Bréhier et al., 2016). More precisely, it is the probability that a trajectory starting from C reaches B before A.

Note that the formula we give here is different from the one given in Cérou and Guyader (2007) or Rolland and Simonnet (2015), which is

$$\hat{\alpha}_N = \frac{N_B}{N} \left(1 - \frac{1}{N}\right)^k. \tag{5.10}$$

The reason for this is that the formula from Bréhier et al. (2016) accounts for the case when two trajectories have the same maximum value of the reaction coordinate. It may seem that this has probability zero, but it may happen that it branches from the maximum value of the reaction coordinate of another trajectory. If the reaction coordinate only decreases from there, this means that now both trajectories have the same maximum value of the reaction coordinate. In case ℓ_i is equal to 1 for all i, (5.9) and (5.10) are the same.

Of course we are not exactly interested in this quantity, but instead we are interested in the transition probability, which we can compute from the mean first passage time. To compute this time we split up reactive trajectories into three parts. The first part is given by the initial step in which trajectories are generated that reach C when starting in A within time t_1. The second part is given by the trajectories that reach A before they reach B when starting in C within time t_2. These are computed during the initial step of AMS as described above, and are observed with probability $1 - \alpha$. The third part is the trajectories that reach B before A when starting in C within time t_3. These are the trajectories that we obtain from the AMS method, and are observed with probability α.

The mean first passage time is then given by

$$\tau_{\text{MFPT}} = \left(\frac{1-\alpha}{\alpha}\right) \mathbb{E}(t_1 + t_2) + \mathbb{E}(t_1 + t_3).$$

All of the quantities required for computing the mean first passage time are obtained during the AMS method, but more samples can be generated for computing $\mathbb{E}(t_1 + t_2)$ to make the result more accurate. Depending on how close C is to A, this should take a relatively small amount of time.

Unlike $\hat{\alpha}_N$, the estimator of the actual transition probability that can be obtained by using (5.4) is biased (Lestang et al., 2018).

Example

As an example we will look at the two-dimensional double well potential defined by

$$V(x, y) = \frac{1}{4}x^4 - \frac{1}{2}x^2 + y^2,$$

which we show in Figure 5.4.

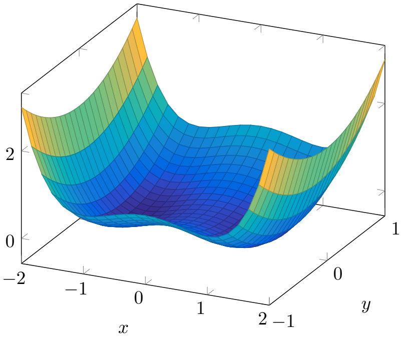


Figure 5.4: The two-dimensional double well potential.

The corresponding SDE that we solve is given by

$$\mathrm{d}\boldsymbol{X}_t = -\nabla V(\boldsymbol{X}_t)\,\mathrm{d}t + g(\boldsymbol{X}_t)\,\mathrm{d}\boldsymbol{W}_t,$$

where we take $g(\boldsymbol{X}_t) = 0.4$. It is easy to see that there are stable steady states at $(-1, 0)$ and $(1, 0)$ and an unstable steady state at $(0, 0)$. We take $\bar{\boldsymbol{x}}_A = (-1, 0)$ and $\bar{\boldsymbol{x}}_B = (1, 0)$, and use

$$\phi(x, y) = \frac{1}{2} - \frac{1}{2}e^{-8((x+1)^2+y^2)} + \frac{1}{2}e^{-8((x-1)^2+y^2)}$$

as the reaction coordinate. This function satisfies all the conditions that we defined earlier.

In this example, we will perform AMS with 3 trajectories. We define A and B to be

$$A = \{\boldsymbol{x} \in \mathbb{R}^2 \ : \ \phi(\boldsymbol{x}) < 0.1\}, \quad B = \{\boldsymbol{x} \in \mathbb{R}^2 \ : \ \phi(\boldsymbol{x}) > 0.9\},$$

$z_{\text{min}} = 0.2$ and $z_{\text{max}} = 0.9$. The initial step, where we compute trajectories that start in A and go through C to either A or B is shown in Figure 5.6.

Next we compute the maximum values of the reaction coordinate for each trajectory, Q_1, Q_2 and Q_3 as shown in Figure 5.7. We then pick a random trajectory, in this case $(\boldsymbol{x})^{(1)}$, and generate a new trajectory which is branched

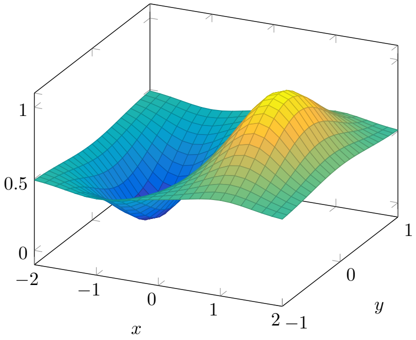


Figure 5.5: Reaction coordinate $\phi(x, y)$.

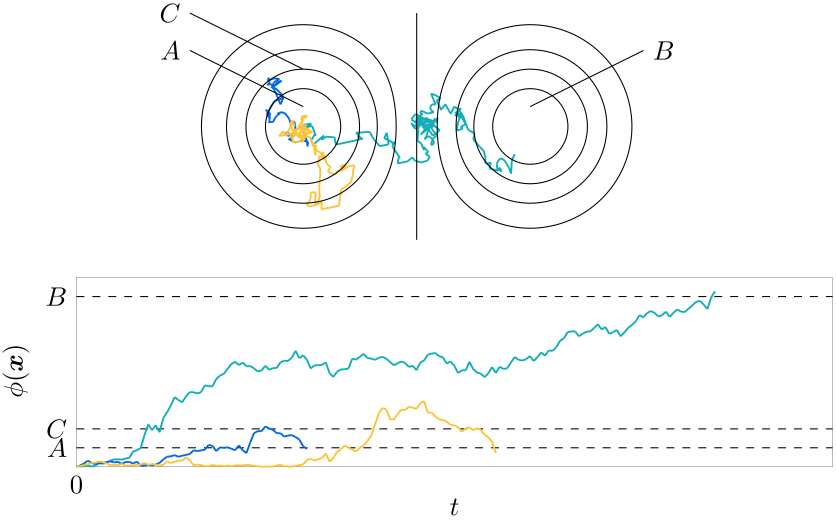


Figure 5.6: First step of AMS where trajectories are computed that start in A and go through C to either A or B. On the top, the trajectories are shown in the xy-plane with contours for $\phi(\boldsymbol{x}) = 0.1, 0.2, \ldots, 0.9$. On the bottom, the reaction coordinate values of the same trajectories are plotted against time. Each trajectory has its own color.

from the position where $(\boldsymbol{x})^{(1)}$ reaches Q_2. The newly generated trajectory $(\tilde{\boldsymbol{x}})^{(2)}$ has a new maximum value of the reaction coordinate $\tilde{Q}_2$, which is also shown in Figure 5.7.

It is clear that $\tilde{Q}_2 > Q_2$, meaning that the trajectory got closer to B. This process is repeated until all trajectories reach B. Of course a sample size of 3 is much too small to obtain an accurate transition rate, so we will not compute this here.

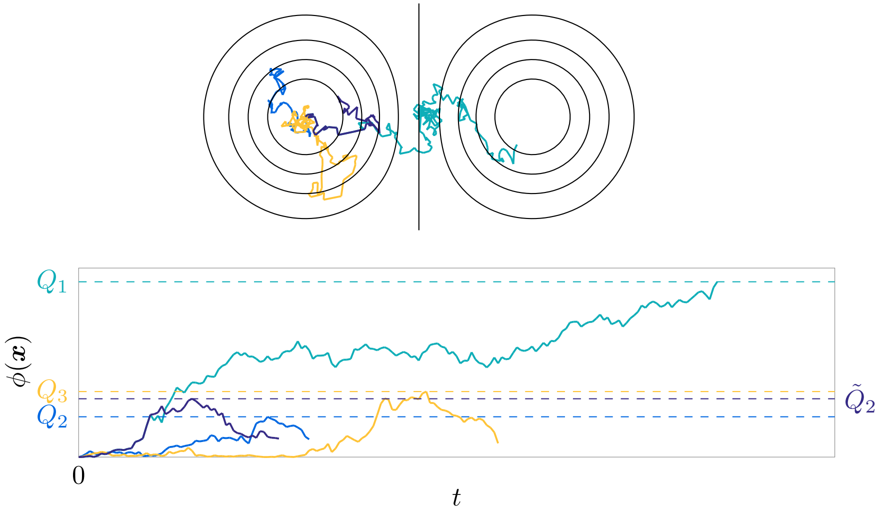


Figure 5.7: The second step of AMS where the second trajectory, which had the lowest value of the reaction coordinate, is replaced with a trajectory that branches from the first trajectory.

Optimizations

Say we have a problem of dimension 10000 with a time step of 0.01, a mean first passage time of 1000, and 1000 samples, which is not at all unreasonable for the problems that we want to solve. In this case we would need at least 7.28 TB of memory to store all of the states that we compute. This is a very large amount of memory, but fortunately, some simple optimizations exist to deal with this.

First, it is important to observe that the only place where we actually use a state is when we branch a trajectory. Other than in this place, we only use the times to compute the quantities that we need. This means that we can discard any state $\boldsymbol{x}$ for which $\phi(\boldsymbol{x}) < Q_l$, since these will never be used for branching. Discarding these states can be done for instance every time a trajectory is branched, every so many iterations, or even in every AMS iteration.

Secondly, since we only use the first $\boldsymbol{x}$ for which $\phi(\boldsymbol{x}) \geq Q_l$, we only have to store $\{\boldsymbol{x}_i \in (\boldsymbol{x}) \;:\; \phi(\boldsymbol{x}_i) > \sup\{\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_{i-1})\}\}$. This means that if we iteratively determine the value of Q, we only store a state $\boldsymbol{x}$ if $\phi(\boldsymbol{x}) > Q$. In addition, one could also limit the number of states that we store by only storing one state per interval of ϕ. So for instance if $z_{\text{min}} = 0.1$, $z_{\text{max}} = 0.9$ and we only store a state with intervals of 0.005, at most 160 states would be stored.

Optimizations can also be done in terms of parallelization. Since all trajectories are computed independently in the first step, parallelization of the first step is trivial. In the later steps, the same holds, but the branching makes it a bit more difficult. What one could do is making a pool of unused trajectories (trajectories that are not being regenerated), and select both trajectories that are being regenerated and random trajectories from which to branch from this pool. In a shared memory parallelization model, this is easy to implement. One just lets every thread draw trajectories from this pool until all trajectories in the pool have reached B.

There are, however, many small details that might cause big problems. For instance, it might happen that one copies the initial part of a branched trajectory at the same time at which this part is being overwritten. Also it might happen that the pool from which we can select trajectories is not updated in all the different threads. The easiest solution is to put everything other than the transient part, where the new part of the trajectory is generated, in a critical section and to make sure to synchronize every time you enter a critical section.

An implementation in pseudocode that utilizes some of these optimizations is shown in Algorithm 8. Changes that can be made to this pseudocode that were discussed before are replacing line 7 and line 33 by $\phi(\boldsymbol{x}_j^{(l)}) > Q_l + 0.005$ or replacing $\boldsymbol{x}_j^{(l)} \in B$ in line 12 and line 35 by $\phi(\boldsymbol{x}_j^{(l)}) > z_{\max}$.

Accuracy of the method

Methods that fall into the Generalized Adaptive Multilevel Splitting framework produce an unbiased estimator $\hat{\alpha}_N$ of the probability α (Bréhier et al., 2016). Generally, however, one realization of the algorithm is not enough to get an accurate answer. Also, from just one realization, we can not tell how accurate the answer actually is. Therefore, one computes K realizations of the algorithm, and uses those to compute both a more accurate answer, and an estimate of the error. From this we get K probability estimates $\hat{\alpha}_N$, which we can use to compute the mean

$$\mu_\alpha = \frac{1}{K}\sum_{i=1}^{K} \hat{\alpha}_N^{(i)}$$

and the variance

$$\sigma_\alpha^2 = \frac{1}{K-1}\sum_{i=1}^{K} \left(\hat{\alpha}_N^{(i)} - \mu_\alpha\right)^2.$$

An estimate of the relative error is given by

$$\epsilon_\alpha = \frac{\sigma_\alpha}{\mu_\alpha},$$

```
input:   $F(\boldsymbol{x}), g(\boldsymbol{x})$   Functions as described in (5.1) with $M = I$.
         $\Delta t$                               Time step.
         $\boldsymbol{x}_0$                       Starting point.
         $N$                                      Number of samples.
output:  $\tau$                                   Mean first passage time.

for $i = 1, 2, \ldots, N$ do
    $t = 0,\ t_1^{(i)} = 0,\ t_2^{(i)} = 0,\ t_3^{(i)} = 0$
    $Q_i = 0$
    for $j = 1, 2, \ldots$ do
        $t = t + \Delta t$
        $\boldsymbol{x}_j^{(i)} = \boldsymbol{x}_{j-1}^{(i)} + F(\boldsymbol{x}_{j-1}^{(i)})\Delta t + \sqrt{\Delta t} g(\boldsymbol{x}_{j-1}^{(i)})\Delta \boldsymbol{W}$
        if $\phi(\boldsymbol{x}_j^{(i)}) > Q_i$ then
            $Q_i = \phi(\boldsymbol{x}_j^{(i)})$
        if $t_1^{(i)} = 0$ and $\phi(\boldsymbol{x}_j^{(i)}) > z_{\min}$ then
            $t_1^{(i)} = t$
            $t = 0$
        else if $t_1^{(i)} > 0$ and $\boldsymbol{x}_j^{(i)} \in B$ then
            $t_3^{(i)} = t$
            break
        else if $t_1^{(i)} > 0$ and $\boldsymbol{x}_j^{(i)} \in A$ then
            $t_2^{(i)} = t$
            break
```

Algorithm 8: AMS step 1: generation of the initial trajectories that start in A, pass through C, and end in A or B.

which in turn can be used for computing error estimates for the mean first passage time and the transition probability (Lestang et al., 2018).

However, since we are interested in probabilities close to 0, a standard confidence interval obtained from the standard deviation σ is not a good representation of the actual error (DasGupta et al., 2001). This comes from the fact that the distribution of samples is usually assumed to be normal, and especially for small sample sizes near 0, this is clearly not the case. For smaller sample sizes the variance is larger, meaning the tail of the normal distribution that is below 0 is also larger. However, it is of course not possible to have a probability smaller than 0, and hence the confidence interval is incorrect.

Instead, we choose to use the interquartile range (IQR), which is the interval between the 25th percentile and 75th percentile, to give an indication of

```
$w_0 = 1$
for $k = 1, 2, \dots$ do
    $L = \{j \,:\, Q_j = \inf\{Q_i \,:\, i \in \{1, \dots, N\}\}\}$
    $\ell_k = \mathrm{card}(L)$
    if $\ell_k = N$ then
        break
    $w_k = \left(1 - \frac{\ell_k}{N}\right) w_{k-1}$
    for all $l \in L$ do
        $r = \mathrm{rand}(\{1, \dots, N\} \backslash L)$
        $j_{\min} = \arg\min_j (\phi(\boldsymbol{x}_j^{(r)}) \,:\, \phi(\boldsymbol{x}_j^{(r)}) \geq Q_l)$
        $(\boldsymbol{x})^{(l)} = (\boldsymbol{x}_0^{(r)}, \dots, \boldsymbol{x}_{j_{\min}}^{(r)})$
        $t = \Delta t \cdot j_{\min}$
        for $j = j_{\min} + 1, j_{\min} + 2, \dots$ do
            $t = t + \Delta t$
            $\boldsymbol{x}_j^{(l)} = \boldsymbol{x}_{j-1}^{(l)} + F(\boldsymbol{x}_{j-1}^{(l)})\Delta t + \sqrt{\Delta t} g(\boldsymbol{x}_{j-1}^{(l)})\Delta \boldsymbol{W}$
            if $\phi(\boldsymbol{x}_j^{(l)}) > Q_l$ then
                $Q_l = \phi(\boldsymbol{x}_j^{(l)})$
            if $\boldsymbol{x}_j^{(l)} \in B$ then
                $t_3^{(l)} = t$
                break
            else if $\boldsymbol{x}_j^{(l)} \in A$ then
                break
```

Algorithm 8: AMS step 2: branching of the trajectories until all of them end up in B.

the error. This can be obtained by sorting the data, and then taking the value at 25% and the value at 75%. Since the IQR is determined directly from the data, we do not obtain any probabilities smaller than 0. Note that the $1.5 \cdot \mathrm{IQR}$ and $3 \cdot \mathrm{IQR}$ values, which are often used in box plots, may still include values smaller than 0, which is why we do not use those.

5.5.4 Trajectory-Adaptive Multilevel Sampling

So far we have been busy computing the mean first passage time, after which we can use (5.4) to compute the actual transition probability under the assumption that this probability follows the exponential distribution. However, it would be much nicer to compute the transition probability directly. Fortunately, a variant of AMS called Trajectory-Adaptive Multilevel Sampling

```
40: $t_1 = \frac{1}{N}\sum_{i=1}^{N} t_1^{(i)},\ t_2 = \frac{1}{N_I}\sum_{i=1}^{N} t_2^{(i)},\ t_3 = \frac{1}{N_B}\sum_{i=1}^{N} t_3^{(i)}$
41: $\hat{\alpha}_N = \frac{N_B w_k}{N}$
42: $\hat{\tau}_{\text{MFPT}} = \left(\frac{1-\hat{\alpha}_N}{\hat{\alpha}_N}\right)(t_1+t_2)+(t_1+t_3)$
```

Algorithm 8: AMS step 3: computation of the mean first passage time. Here N_B is the number of trajectories that reached B, and N_I is the number of initial trajectories that ever reached A before B.

(TAMS) (Lestang et al., 2018) exists, which allows for the direct computation of the transition probability when given a maximum time T.

In TAMS, there is an optional maximum number of iterations $k_{\max}$ that can be set, which allows us to stop even before all trajectories have reached B.

The difference between the two algorithms is the absence of C, not stopping when reaching A, and the usage of a maximum time T. These differences only show in steps 1 and 6, but simplifies the implementation considerably. Since a correct implementation is crucial, we again list all steps that are needed to compute the transition probability like we did for AMS.

1. First generate N independent trajectories $(\boldsymbol{x})^{(1)}, \dots, (\boldsymbol{x})^{(N)}$ that start in A until $t = T$ is reached, or the trajectory ends up in B. Here $(\boldsymbol{x})^{(i)} = (\boldsymbol{x}_0^{(i)}, \boldsymbol{x}_1^{(i)}, \dots, \boldsymbol{x}_M^{(i)})$ is a trajectory of length $M = T/\Delta t$.

2. Each trajectory $(\boldsymbol{x})^{(i)}$, $i = 1, \dots, N$, has a certain maximum value of the reaction coordinate (or maximum distance from A), which we define to be Q_i. Set $k = 1$, $w_0 = 1$.

3. Take $L = \{j \ : \ Q_j = \inf\{Q_i \ : \ i \in \{1, \dots, N\}\}\}$, which are the indices for which the maximum value of the reaction coordinate is minimal, and take $\ell_k = \text{card}(L)$ to be the number of elements in L. Since the trajectories $\{(\boldsymbol{x})^{(j)} \ : \ j \in L\}$ have the smallest value of Q_i, all other trajectories have some j for which $\phi(\boldsymbol{x}_j) > Q_l$ for all $l \in L$.

4. Set $w_k = \left(1 - \frac{\ell_k}{N}\right) w_{k-1}$. For all $l \in L$, repeat steps 5-7.

   5. Select a random trajectory $(\boldsymbol{x})^{(r)}$ with r from $\{1, \dots, N\} \backslash L$, and set $(\tilde{\boldsymbol{x}})^{(l)} = (\boldsymbol{x}_0^{(r)}, \dots, \boldsymbol{x}_{j_{\min}}^{(r)})$, where $j_{\min}$ is the smallest value for which $\phi(\boldsymbol{x}_{j_{\min}}^{(r)}) \geq Q_l$.

6. Generate the rest of the trajectory starting from $\boldsymbol{x}^{(r)}_{j_{\min}}$ until again you reach either $t = T$ or B. This trajectory has a new maximum value of the reaction coordinate $\tilde{Q}_l$, which is always greater than or equal to Q_l.

7. Set $(\boldsymbol{x})^{(l)} = (\tilde{\boldsymbol{x}})^{(l)}$ and $Q_l = \tilde{Q}_l$.

8. Repeat steps 3-7 with $k = k + 1$ until $Q_i \geq z_{\max}$, $\forall\ i = 1, \ldots, N$ or $k = k_{\max}$.

Now the quantity that we can compute is again the estimate of the probability α of observing a reactive trajectory, but since we now introduced a maximum time, this is now actually the transition probability (Lestang et al., 2018). So an unbiased estimator of the transition probability is again given by

$$\hat{\alpha}_N = \frac{N_B w_k}{N} = \frac{N_B}{N} \prod_{i=0}^{k} \left(1 - \frac{\ell_i}{N}\right),$$

where k is the number of iterations it took for all trajectories to converge, and N_B is the number of trajectories that reached B.

Example

We can again use the example from Section 5.5.3 to show how TAMS works exactly. The initial step, where we compute trajectories that start in A and either go to B or end after time T is shown in Figure 5.8.

This figure shows basically the same behavior as we saw in Figure 5.6, except that the trajectories keep going for longer. This is not very representative for an actual application with small noise, where trajectories generated by AMS are often much longer, especially when they reach B.

Next we again compute the maximum values of the reaction coordinate, and branch from the trajectory with the lowest value. The result is shown in Figure 5.9. We now actually see different behavior compared to Figure 5.7, since even though the new trajectory came back to A, we did not stop there, and ended up with a larger value of the reaction coordinate afterwards. In this case $\tilde{Q}_2$ is larger than Q_3, so the third trajectory is now the next one that will be killed and branched, where this was the second trajectory in Figure 5.7.

The main big advantage of this method is not really displayed here, because of the nature of the example. The example was chosen in such a way that trajectories actually manage to reach B, which means that the mean first passage time is quite short. This also means that all the trajectories that we show actually reached A or B before time T. Usually this is not the case, and generally, trajectories take much longer than time T to actually reach either A or B, so with TAMS, it is much faster to generate new trajectories.

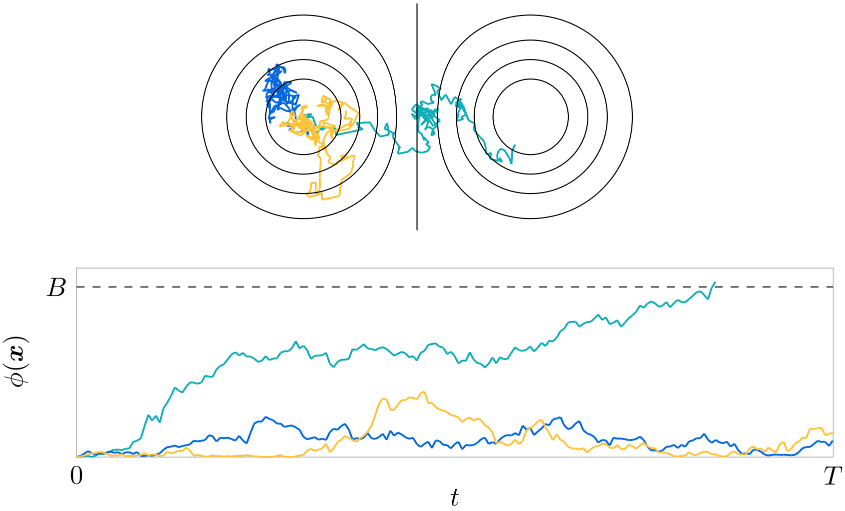


Figure 5.8: First step of TAMS where we compute trajectories that start in A and either go to B or end after time T. On the top, the trajectories are shown in the xy-plane with contours for $\phi(\boldsymbol{x}) = 0.1, 0.2, \ldots, 0.9$. On the bottom, the reaction coordinate values of the same trajectories are plotted against time. Each trajectory has its own color.

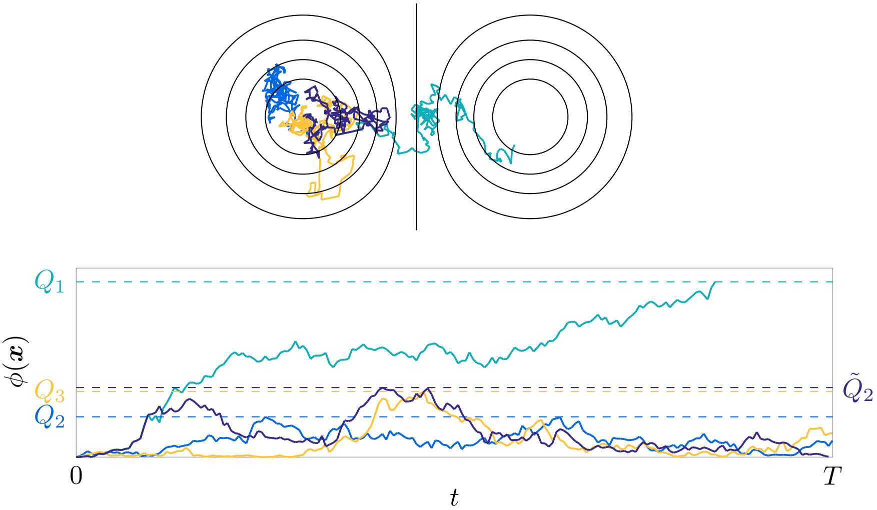


Figure 5.9: The second step of TAMS where the second trajectory, which had the lowest value of the reaction coordinate, is replaced with a trajectory that branches from the first trajectory.

Genealogical Particle Analysis 5.5.5

Yet another method for computing probabilities of rare events is the Genealogical Particle Analysis (GPA) method (Moral and Garnier, 2005; Moral, 2013; Wouters and Bouchet, 2016). Similarly to TAMS, GPA also works by generating N independent trajectories starting in A and ending at a certain time T. The difference is that instead of killing and branching trajectories, trajectories are resampled at certain time intervals based on weights that are assigned to each trajectory. Additionally, one keeps track of the probability of observing a certain trajectory, which can ultimately be used to compute the transition probability. Again, there are many variations of this algorithm, but here we will describe only the first variant that is discussed in Moral and Garnier (2005).

We start with defining the function $W : \mathbb{R}^n \to \mathbb{R}$ that is used to compute the weights of the trajectories

$$W(\boldsymbol{x}) = e^{\beta\phi(\boldsymbol{x})},$$

where we use the same function $\phi : \mathbb{R}^n \to \mathbb{R}$ that we used in AMS, since this gives a good quantification of when we are close to B.

The following steps define a general GPA algorithm:

1. First take N initial conditions $\boldsymbol{x}_0^{(1)}, \dots, \boldsymbol{x}_0^{(N)}$ in A and assign weights $w_0^{(i)} = 1$ and observation probabilities $p_0^{(i)} = 1$ with $i = 1, \dots, N$. We call the tuple $(\boldsymbol{x}, w, p)$ a particle. Take the interval size τ, and set $k = 0$.

2. Compute the normalizing factor
$$\eta = \frac{1}{N}\sum_{i=1}^{N} w_k^{(i)}.$$

3. Choose N independent particles with probability proportional to their weight from
$$(\boldsymbol{x}_k^{(1)}, w_k^{(1)}, p_k^{(1)}), \dots, (\boldsymbol{x}_k^{(N)}, w_k^{(N)}, p_k^{(N)}).$$
The new particles are denoted by
$$(\tilde{\boldsymbol{x}}_k^{(1)}, \tilde{w}_k^{(1)}, \tilde{p}_k^{(1)}), \dots, (\tilde{\boldsymbol{x}}_k^{(N)}, \tilde{w}_k^{(N)}, \tilde{p}_k^{(N)}).$$

4. Generate new independent trajectories starting in $\tilde{\boldsymbol{x}}_k^{(1)}, \dots, \tilde{\boldsymbol{x}}_k^{(N)}$ until they reach τ. $\boldsymbol{x}_{k+1}^{(1)}, \dots, \boldsymbol{x}_{k+1}^{(N)}$ are defined by the end points of the generated trajectories.

5. Compute the associated weights and observation probabilities

$$p_{k+1}^{(i)} = \frac{\eta\, \tilde{p}_k^{(i)}}{\tilde{w}_k^{(i)}}, \tag{5.11}$$
$$w_{k+1}^{(i)} = W(\boldsymbol{x}_{k+1}^{(i)}),\ i = 1, \ldots, N.$$

6. Repeat steps 2-5 with $k = k + 1$ until $k\tau = T$.

To compute the transition probability, we additionally have to store for every particle, if the associated trajectories ever reached B. We can do this by instead defining a particle to be a tuple $(\boldsymbol{x}, \gamma, w, p)$, where $\gamma = 0$ if the associated trajectories never reached B, and $\gamma = 1$ if they did. The approximation to transition probability is then given by

$$\hat{\alpha}_N = \frac{1}{N} \sum_{i=1}^{N} \gamma_k^{(i)}\, p_k^{(i)}.$$

An implementation that computes the transition probability in this way is shown in Algorithm 9.

Example

We used the same example we used for AMS in Section 5.5.3 to show the behavior of GPA. The time is split into four intervals, and we use four trajectories. At the end of each interval, the trajectories are resampled. The result is shown in Figure 5.10.

Note that this method differs a lot from the AMS methods, and is also much easier to understand from one picture. In this case, we see that the top trajectories usually have more trajectories that branch from them. In the end, one of the trajectories reaches B, but unlike in AMS, we still have to keep iterating with this trajectory, since it might have to be used for resampling later.

5.5.6 Comparison

We compare the behavior of the different methods that are described in the previous sections, again, using the example from Section 5.5.3. For the noise coefficient we take $g(\boldsymbol{X}_t) = \sqrt{0.1}$. Note that in Section 5.5.3 we took a value of 0.4, which was only for the sake of being able to generate more informative figures. We perform a range of experiments in Matlab 2018a for different values of the maximum time T, using $N = 10000$ samples and repeating the experiment 1000 times.

```
input:  $F(\boldsymbol{x}), g(\boldsymbol{x})$   Functions as described in (5.1) with $M = I$.
        $\Delta t$                                 Time step.
        $\tau$                                     Time interval size. Multiple of $\Delta t$.
        $T$                                        End time. Multiple of $\tau$.
        $\boldsymbol{x}_0$                         Starting point.
        $N$                                        Number of samples.
output: $\alpha$                                   Transition probability.

for $i = 1, 2, \ldots, N$ do
    $(\boldsymbol{x}_0^{(i)}, \gamma_0^{(i)}, w_0^{(i)}, p_0^{(i)}) = (\boldsymbol{x}_0, 0, 1, 1)$
for $k = 0, 1, \ldots, T/\tau - 1$ do
    $\eta = \frac{1}{N} \sum_{i=1}^{N} w_k^{(i)}$.
    for $i = 1, 2, \ldots, N$ do
        $r = \mathrm{rand}([0, \eta])$
        for $j = 1, 2, \ldots, N$ do
            if $\sum_{l=1}^{j} w_k^{(l)} \geq r$ then
                $(\tilde{\boldsymbol{x}}_k^{(i)}, \tilde{\gamma}_k^{(i)}, \tilde{w}_k^{(i)}, \tilde{p}_k^{(i)}) = (\boldsymbol{x}_k^{(j)}, \gamma_k^{(j)}, w_k^{(j)}, p_k^{(j)})$
    for $i = 1, 2, \ldots, N$ do
        $\boldsymbol{y}_0 = \tilde{\boldsymbol{x}}_k^{(i)}$
        for $j = 1, 2, \ldots, \tau/\Delta t$ do
            $\boldsymbol{y}_j = \boldsymbol{y}_{j-1} + F(\boldsymbol{y}_{j-1})\Delta t + \sqrt{\Delta t} g(\boldsymbol{y}_{j-1}) \Delta \boldsymbol{W}$
            if $\boldsymbol{y}_j \in B$ then
                $\gamma_{k+1}^{(i)} = 1$
        $p_{k+1}^{(i)} = \eta \, \tilde{p}_k^{(i)} / \tilde{w}_k^{(i)}$
        $\boldsymbol{x}_{k+1}^{(i)} = \boldsymbol{y}_{\tau/\Delta t}$
        $w_{k+1}^{(i)} = W(\boldsymbol{x}_{k+1}^{(i)})$
$k = T/\tau$
$\hat{\alpha}_N = \frac{1}{N} \sum_{i=1}^{N} \gamma_k^{(i)} p_k^{(i)}$
```

Algorithm 9: Genealogical Particle Analysis implementation for determining the transition probability.

The methods can be divided into two categories, being methods which compute the transition probability from the mean first passage time and methods that compute the transition probability directly. Since this system is a gradient system, we can use the Eyring–Kramers formula to compute the mean first passage time under the assumption of the noise being sufficiently small.

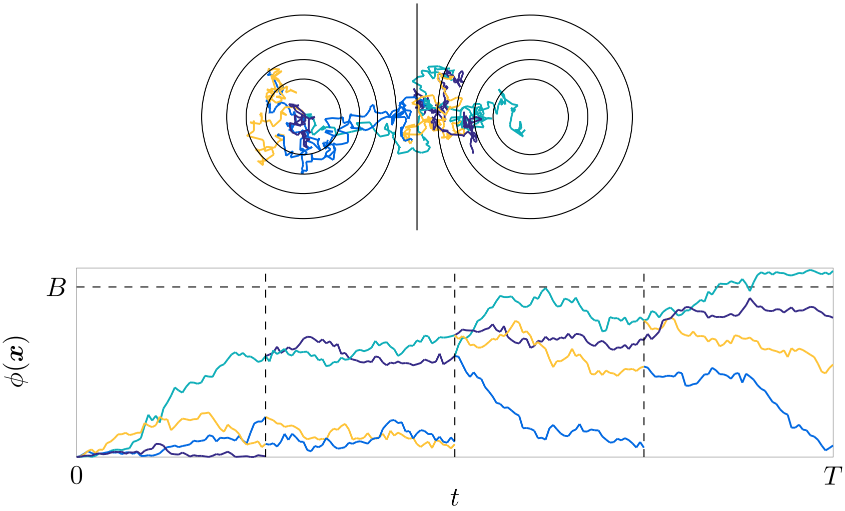


Figure 5.10: GPA for computing trajectories that start in A and either go to B or end after time T. On the top, the trajectories are shown in the xy-plane with contours for $\phi(\boldsymbol{x}) = 0.1, 0.2, \ldots, 0.9$. On the bottom, the reaction coordinate values of the same trajectories are plotted against time. Each trajectory in each interval has its own color.

We will also compare the first class of methods to this theoretical value.

The results with T in the range [1,50] can be found in Figure 5.11. A more detailed view of the range [1,10] can be found in Figure 5.12. Since we are interested in rare transition events, this range is of more interest to us.

The first thing we notice in Figure 5.11 is that GPA gives really bad approximations for larger values of T. This is partially explained by the fact that (5.11) is not limited from above by 1. It actually happens that there are values of $\hat{\alpha}_N$ that are larger than 1, which also explains the larger peaks, and the fact that both 25th and 75th percentile are below the mean, which gives an IQR that is below the actual line.

Other than that we see that the direct sampling method and TAMS look very similar, and that near 1, the theory, the direct sampling of the mean first passage time and AMS are very close to each other. However, both groups of methods do not seem to agree, especially for smaller values of T.

When we zoom in on the interval [1,10], which can be found in Figure 5.12, we see that here GPA agrees more with TAMS and the direct sampling method as expected, since this method also computes the transition probability directly. Between these three methods, TAMS has the smallest error bars. We also observe that when using estimates of the mean first passage time, we obtain a much worse estimate of the transition probability. This even holds for the value that was computed directly from the Eyring–Kramers formula. This

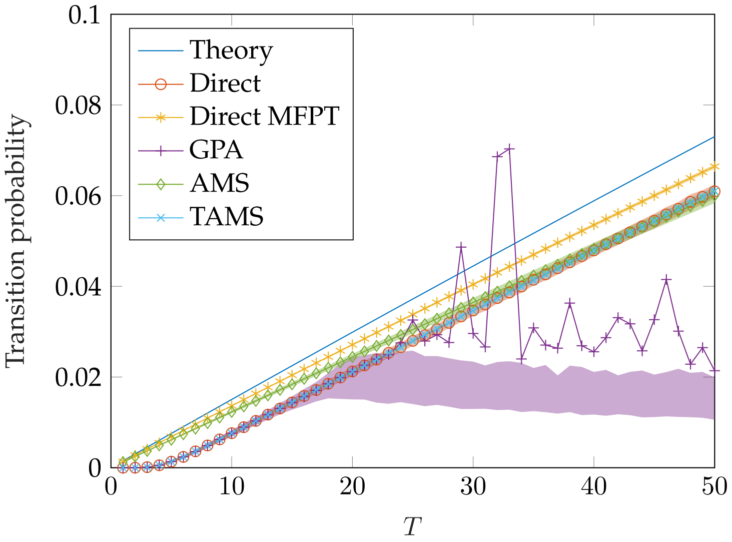


Figure 5.11: Comparison of 1000 experiments with all described methods with maximum times in the range [1,50] and with $N = 10000$. The shaded areas show the IQR.

can be explained from the fact that the transitions do not happen instantly as was assumed in Section 5.2.2.

The fact that TAMS has smaller error bars does not necessarily mean that is it also the most efficient method, which is why we also plot the work-normalized relative error (Glynn and Whitt, 1992), which is given by

$$\epsilon_\omega = \frac{\sqrt{\omega}\sigma_\alpha}{\mu_\alpha}$$

for some amount of work ω. Since Matlab does not allow for a fair comparison of the computational time, we instead use the simulated time to describe the work. This makes sense, since we use a constant time step, and for high-dimensional models, time stepping is by far the most expensive part of the computation. The results can be found in Figure 5.13.

Here we see that TAMS is slightly more efficient than GPA and direct sampling of the transition probability. We also observe that the work-normalized relative error increases with a factor $\alpha^{-1/2}$ for smaller maximum times for methods that compute the transition probability directly. This is as expected for direct sampling, which converges with $\mathcal{O}((\alpha N)^{-1/2})$ (Rubino and Tuffin, 2009), and the worse case scenario for TAMS, which can show convergence behavior between $\mathcal{O}(\sqrt{-\log \alpha} N^{-1/2})$ (optimal) and $\mathcal{O}((\alpha N)^{-1/2})$ (worst case) (Simonnet, 2014; Lestang et al., 2018). The choice of the reaction coordinate has a large impact on the efficiency of (T)AMS (Rolland and Simonnet, 2015;

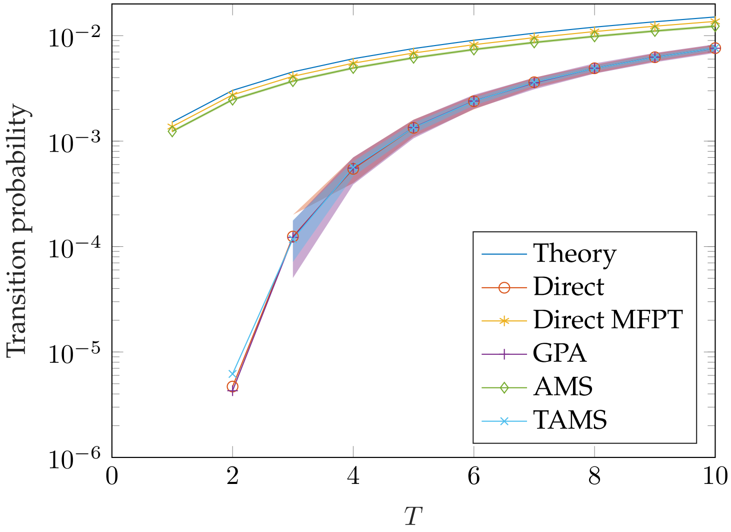


Figure 5.12: Comparison of 1000 experiments with all described methods with maximum times in the range [1,10] and with $N = 10000$. The shaded areas show the IQR. The results of the Direct, GPA and TAMS methods where no shaded area is present have a 25th percentile of 0.

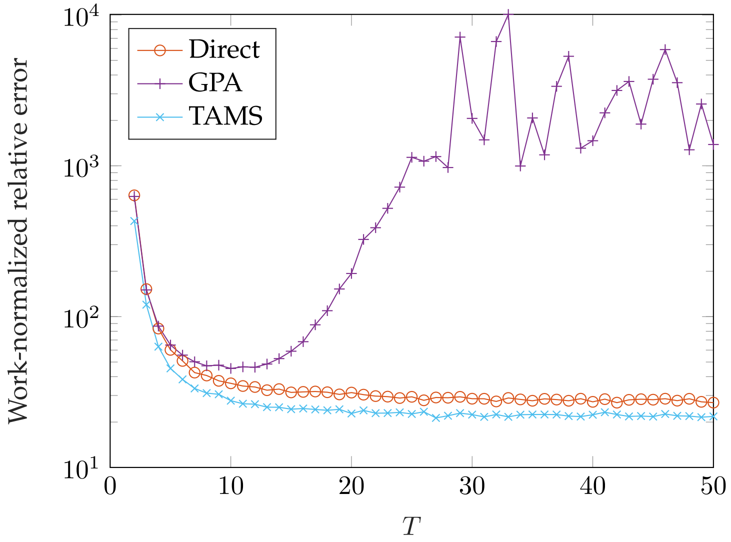


Figure 5.13: The work-normalized relative error for 1000 experiments with the Direct, GPA and TAMS methods with maximum times in the range [1,50] and with $N = 10000$.

Rolland et al., 2015; Lestang et al., 2018), which means that a better reaction coordinate can be used to further reduce the work-normalized relative error and improve the convergence behavior. The optimal convergence behavior, however, is only attained when the optimal reaction coordinate, which is also referred to as the committor, is used, but in practice this is never known.

The images in this section can be reproduced with the Matlab code at `https://github.com/Sbte/transitions`.

Summary and Discussion 5.6

In this chapter we discussed many different methods for studying transition probabilities, and analyzed their behavior on the two-dimensional double well potential. We started with describing what a transition probability actually is, which is the probability of going from a neighborhood A near a deterministic steady state $\bar{\boldsymbol{x}}_A$ to a neighborhood B near a deterministic steady state $\bar{\boldsymbol{x}}_B$ within time T.

We then described the Eyring–Kramers formula for computing the mean first passage time for gradient systems. The mean first passage time can in turn be used for the computation of the transition probability by using the cumulative distribution function of the exponential distribution under the assumption that transitions are instantaneous.

Covariance ellipsoids can be used to study the behavior around a steady state, and can be used to compare the sensitivity to noise at different parameter values, but we found that it is insufficient to compute actual transition probabilities. We used these covariance ellipsoids in Mulder et al. (2018) to investigate patterns of transition behavior in marine ice sheet instability problems. There a good correspondence with results that were obtained with transient computations was observed.

It is also possible to compute most probable transition trajectories, as was done in Laurie and Bouchet (2015) for the barotropic quasi-geostrophic equations. This may also be done for the MOC problem in the future, to obtain more knowledge about the transition behavior, since from our experience, investigating transitions in the QG model is harder than investigating transitions in the 2D MOC.

After this we described five different numerical methods for computing transition probabilities: direct sampling, direct sampling of the mean first passage time, AMS, TAMS and GPA. We saw that TAMS gives the best results for all ranges of transition probabilities. We will therefore use TAMS in the next chapter to compute transition probabilities for the MOC. To further improve the results of TAMS, it would be interesting to investigate the effect of different reaction coordinates in the future.

TRANSITIONS IN THE MERIDIONAL OVERTURNING CIRCULATION

For high-dimensional problems, the computation of transition probabilities of rare events is not feasible with standard Monte Carlo methods as mentioned in the previous chapter. Therefore specialized methods such as Genealogical Particle Analysis (GPA) (Moral and Garnier, 2005; Moral, 2013; Wouters and Bouchet, 2016), Adaptive Multilevel Splitting (AMS) (Cérou and Guyader, 2007; Rolland and Simonnet, 2015; Rolland et al., 2015), and based on that Trajectory-Adaptive Multilevel Sampling (TAMS) (Lestang et al., 2018) exist. The idea behind these methods is similar: they try to push samples in the direction of the transition to make sure transitions actually occur, also for smaller sample sizes, while still being able to compute the transition probability for the original problem.

Our goal is to compute transition probabilities in the Meridional Overturning Circulation (MOC). Rare transitions in the two-dimensional barotropic quasi-geostrophic equations have been studied in Bouchet et al. (2014); Laurie and Bouchet (2015), but there only the most likely transition path was determined by means of the minimum action method (E et al., 2004; Vanden-Eijnden and Heymann, 2008; Zhou et al., 2008; Grafke et al., 2017). To our knowledge, actual transition probabilities have never been computed for high-dimensional problems on a scale that we present here.

Methods for finding transition probabilities have two main culprits. The first is that they always require some form of time integration, which is very expensive for high-dimensional systems. This is especially the case when an implicit time-stepping method is used, since then in every time step a linear system has to be solved. The other is that for methods based on AMS, a large number of states at different time steps have to be stored in memory.

We therefore propose a projected time-stepping method in Section 6.1, for which in every time step, only a small projected linear system has to be solved, and for which only the projected states have to be stored, which are much smaller than the original states. Based on the results that were obtained in the previous chapter, we decided to apply this method to TAMS, but it may be applied to any method for computing transition probabilities. In Section 6.2 we then discuss the problem setting and in Section 6.3 we discuss the results. The results and the method are summarized and discussed in Section 6.4.

6.1 Projected time-stepping in TAMS

In AMS and TAMS, but also in GPA, one is free to choose any stochastic time stepping method. In case the stochastic theta method (2.8) is used with $\theta \neq 0$, multiple linear systems have to be solved in every time step. In that case, solving these linear systems is usually the computationally most expensive part of TAMS, even if the linear system is sparse, and an iterative solver with a good preconditioner is used.

At a time step i of the stochastic theta method, take

$$\hat{F}(\boldsymbol{x}) = M\boldsymbol{x}_i - M\boldsymbol{x} + (1-\theta)\Delta t F(\boldsymbol{x}_i) + \theta \Delta t F(\boldsymbol{x}) + g(\boldsymbol{x}_i)\,\Delta \boldsymbol{W}_i$$

with Jacobian

$$\hat{J}(\boldsymbol{x}) = \theta \Delta t J(\boldsymbol{x}) - M.$$

A Newton solve for one implicit time step would normally look like this

```
$\boldsymbol{y}^{(0)} = \boldsymbol{x}_i$
for $j = 1, 2, \dots$ until convergence do
    solve $\hat{J}(\boldsymbol{y}^{(j-1)})\Delta\boldsymbol{y}^{(j)} = -\hat{F}(\boldsymbol{y}^{(j-1)})$
    $\boldsymbol{y}^{(j)} = \boldsymbol{y}^{(j-1)} + \Delta\boldsymbol{y}^{(j)}$
$\boldsymbol{x}_{i+1} = \boldsymbol{y}^{(j)}$
```

A small potential optimization that can be made is computing $\hat{J}(\boldsymbol{x}_i)$ beforehand, and using this instead of $\hat{J}(\boldsymbol{y}^{(j-1)})$. This will make it such that Newton does not converge quadratically, but saves the time of computing the Jacobian matrix multiple times.

Instead of solving with this large sparse matrix $\hat{J}(\boldsymbol{y}^{(j-1)})$ in every Newton iteration, we instead propose to solve with a smaller matrix obtained from a Galerkin type projection $V^T\hat{J}(\boldsymbol{x}_i)V$, where V consists of an orthogonal basis of the most important directions that are used by the stochastic perturbation.

These vectors, at least around a steady state, can be obtained from the eigenvectors belonging to the largest eigenvalues of the covariance matrix, which can in turn be obtained from the basis vectors of the low-rank solution of a generalized Lyapunov equation as described in Chapter 4.

Say we have an orthonormal basis V spanned by $\bar{\boldsymbol{x}}_A$, the basis of the low-rank Lyapunov solution V_A at $\bar{\boldsymbol{x}}_A$, $\bar{\boldsymbol{x}}_B$, the basis of the low-rank Lyapunov solution V_B at $\bar{\boldsymbol{x}}_B$, and the noise matrices B_A and B_B at $\bar{\boldsymbol{x}}_A$ and $\bar{\boldsymbol{x}}_B$. We can then replace the Newton iteration with

```
$\boldsymbol{y}^{(0)} = V^T \boldsymbol{x}_i$
$A = V^T \hat{J}(\boldsymbol{x}_i) V$
for $j = 1, 2, \dots$ until convergence do
    solve $A \Delta \boldsymbol{y}^{(j)} = -V^T \hat{F}(V \boldsymbol{y}^{(j-1)})$
    $\boldsymbol{y}^{(j)} = \boldsymbol{y}^{(j-1)} + \Delta \boldsymbol{y}^{(j)}$
$\boldsymbol{x}_{i+1} = V \boldsymbol{y}^{(j)}$
```

If we apply this in TAMS, we see that between two consecutive time steps, we always apply $V^T V \boldsymbol{y}^{(j)}$. Now since V is orthonormal, this means that we might as well apply TAMS itself to $\boldsymbol{y}^{(j)}$ directly. Doing this actually solves the largest problem we observe when applying TAMS to high-dimensional systems, which is the storage of the trajectories, since we now only have to store the trajectories restricted to the space V. Say we have a system of size 10000 and a space V consisting of 100 vectors, which is a realistic scenario, then this reduces the required storage by a factor 100.

Problem setting 6.2

In this section we discuss the problem setting of the ocean model as described in Section 2.5.2. The results with this model are shown in Section 6.3.

Bifurcation diagram 6.2.1

For the deterministic model, we take the surface forcing slightly different from the forcing that was described in Section 2.5.3 and more similar to what was used in Van der Mheen et al. (2013) as

$$\bar{T}(\theta) = 10.0 \cos\left(\pi\theta/\theta_N\right), \tag{6.1a}$$

$$F_s(\theta) = \bar{F}_s(\theta) = \mu F_0 \frac{\cos(\pi\theta/\theta_n)}{\cos(\theta)} + \beta F_0 F_p(\theta), \tag{6.1b}$$

where $\mu = 0.3$ is the strength of the mean freshwater forcing, $F_0 = 0.1$ m yr^{-1} is a reference freshwater flux, β is the strength of an anomalous freshwater flux F_p, which is 1 in the area $[45^\circ N, 60^\circ N]$, and 0 elsewhere.

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 $n = 6n_xn_yn_z$, 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 $u = 0$.

The bifurcation diagram of the deterministic model for parameters as in Table 2.1 is shown in Figure 6.1a. 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

$$\frac{\partial \Psi}{\partial z} = v \cos\theta, \quad -\frac{1}{r_0 \cos\theta}\frac{\partial \Psi}{\partial \theta} = w. \tag{6.2}$$

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.

We are interested in transitions from the top branch to the bottom branch of the bifurcation diagram. We will investigate transitions at two parameter values $\beta_a = -0.2162$ and $\beta_b = -0.1982$. The transitions will be from point a to a' and from point b to b'. The pattern of the asymmetric overturning streamfunction at location b with sinking in the northern part of the basin is shown in Figure 6.1b, as well as the overturning streamfunction at location b' with sinking in the southern part of the basin. The saddle-node bifurcations are located at $\beta = -0.2320$ and $\beta = 0.3371$.

6.2.2 Stochastic freshwater forcing

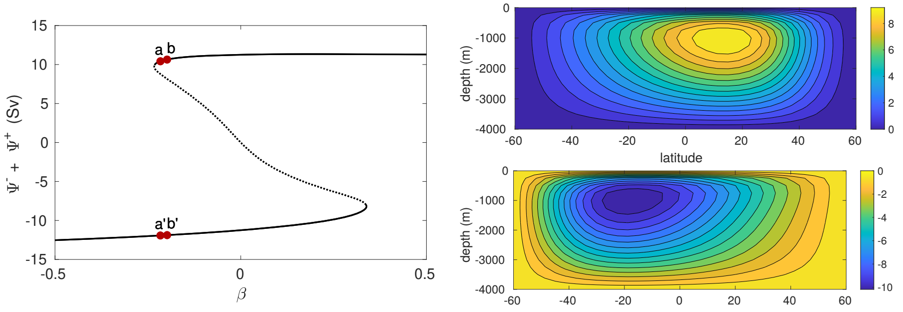
For the stochastic model, we use the same freshwater forcing as given in Section 2.5.3. However, because we use a different deterministic forcing, we can not simply describe F_s in terms of $\bar{F}_s$ as given in Section 6.2.1. The more precise description is given by

$$F_s(\theta, t) = (1 + \sigma\zeta(\theta, t))\,\mu F_0 \frac{\cos(\pi\theta/\theta_n)}{\cos(\theta)} + \beta F_p(\theta), \tag{6.3}$$

where $\zeta(\theta, t)$ is as described in Section 2.5.3.

6.2.3 Reaction coordinate

Since p and w are determined by the algebraic constraints up to a constant, and therefore do not have a unique solution, computing the reaction coordinate



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

(b) The streamfunction at b and b' respectively.

Figure 6.1: (a) Bifurcation diagram of the deterministic 2D MOC model, with the forcing as in (6.1). (b) Streamfunction pattern at b and b' respectively.

by using the norm of the state is not possible. We instead want to define the reaction coordinate in such a way that the distance between the meridional streamfunctions goes to zero, and therefore we define a norm on the v-part of the state. Combining this with (5.8), this gives us a reaction coordinate of the form

$$\phi(\boldsymbol{x}) = \eta - \eta e^{-8\|\boldsymbol{x}-\bar{\boldsymbol{x}}_A\|_v^2} + (1-\eta)e^{-8\|\boldsymbol{x}-\bar{\boldsymbol{x}}_B\|_v^2},$$

where $\|\cdot\|_v$ denotes the 2-norm on only the v-velocities, and $\eta = \|\boldsymbol{x}_C - \bar{\boldsymbol{x}}_A\|_v / \|\boldsymbol{x}_B - \bar{\boldsymbol{x}}_A\|_v$ is the normalized difference between the unstable steady state $\boldsymbol{x}_C$ and the stable steady state $\boldsymbol{x}_A$. We assume that a trajectory has reached B if $\phi(\boldsymbol{x}) > z_{\max} = 0.75$, since $\boldsymbol{x}_C$ is really close to $\boldsymbol{x}_A$.

6.3 Results

In this section we report on the transition probabilities that we find and compare the results from TAMS with the results from its projected variant.

Computations were performed on Peregrine, the HPC cluster of the University of Groningen. Peregrine has nodes with 2 Intel Xeon E5 2680v3 CPUs (24 cores at 2.5GHz) and each node has 128 GB of memory. For every run of TAMS, four cores were used to allow for efficient parallelization. Due to the problem size and the way the vertical mixing is implemented, it did not make sense to use more than four cores.

For our experiments, we investigated the probability of a transition within 2000 years. We ran TAMS 100 times with 300 samples. The stochastic theta method was used with $\theta = 0.5$ and a time step of 2 years. For projected TAMS we used the approximate solution of the generalized Lyapunov equation with a tolerance of the relative residual of 10^{-8}. The results are shown in Table 6.1.

Method	β	$\beta + 0.2320$	Mean	Variance
TAMS	-0.2162	0.0158	0.0495	$1.48 \cdot 10^{-4}$
Projected TAMS	-0.2162	0.0158	0.0496	$1.47 \cdot 10^{-4}$
TAMS	-0.1982	0.0338	$2.52 \cdot 10^{-7}$	$2.16 \cdot 10^{-13}$
Projected TAMS	-0.1982	0.0338	$3.36 \cdot 10^{-7}$	$5.35 \cdot 10^{-13}$

Table 6.1: Mean and variance of the probability of a transition within $T = 2000$ yr with 100 runs of TAMS and the projected variant with 300 samples. $\beta + 0.2320$ is the distance to the bifurcation point, which is located at $\beta = -0.2320$.

Here we see that a small step away from the bifurcation means a drastic decrease of the transition probability. Due to the convergence behavior of the relative error, the variance is relatively large for the rightmost point. We are only interested in the order of magnitude of the probabilities, however, and in the comparison between the two methods. To further decrease the variance in the future, we have to find a better reaction coordinate.

We notice a slight difference in the probability and an increased variance for projected TAMS at point b, which is the furthest away from the bifurcation. This can be explained by the fact that the probability is close to the tolerance of the Lyapunov solver. This may be solved by using a smaller tolerance for the Lyapunov solver, at the cost of more computational time and more memory usage. For the point that is closest to the bifurcation, the probabilities and variances are nearly the same.

In Table 6.2 we show the average computational time and memory usage of each time step in TAMS. We see that the projected method is approximately 30% faster, but more interestingly, uses only about 4% of the memory.

	$t(s)$	Mem (kB)
Normal time step	0.452	96
Projected time step	0.311	4

Table 6.2: Time and memory usage per time step in TAMS at point b.

To investigate if the time step of 2 years was small enough, we also ran the experiment at a time step of 1 year, of which the results are shown in Table 6.3. We see that the results are very similar, which confirms that a time step of 2 years is sufficiently small.

6.4 Summary and Discussion

Based on the results that were obtained in the previous chapter, we decided to use TAMS to compute transition probabilities in the idealized 2D MOC model. We showed that this is indeed possible, and observed that the transi-

Time step	β	Mean	Variance
2 yr	-0.1982	0.0495	$1.48 \cdot 10^{-4}$
1 yr	-0.1982	0.0493	$1.45 \cdot 10^{-4}$

Table 6.3: Different time steps for $T = 2000$ yr with 100 runs of TAMS with 300 samples for time steps of 1 and 2 years.

tion probability decreased rapidly when moving away from the saddle-node bifurcation.

The computation of these transition probabilities is very expensive in terms of both time and memory usage. To be able to also compute transition probabilities on more realistic models, or in case the transition probability is small, the computational cost has to be reduced. The proposed projected time stepping method was shown to be 30% faster, and uses 96% less memory. We applied the projected time stepping method in TAMS, but the method may also be used in other methods for investigating rare events such as AMS and GPA.

In the future, it would be interesting to investigate the possibility of using a combination of most probable transition paths as computed in Laurie and Bouchet (2015) and a resampling method as used in AMS or GPA to obtain the transition probability in an even more efficient way. Such a method is very likely to still include stochastic time stepping, which means it can still be combined with the projected time stepping method described in this chapter.

Something that also needs further investigation is the reaction coordinate. In this thesis, we just chose a reaction coordinate that works, but it may be possible to find a reaction coordinate that works better. It would especially be nice if we can find a reaction coordinate that works well for a wide variety of high-dimensional systems.

CHAPTER 7

CONCLUSION

The goal of this thesis was to develop methods which can be used for studying transition behavior in the Meridional Overturning Circulation (MOC). The reason why we want to study such behavior is that melting of the polar ice sheets, and thereby increasing the freshwater in the North Atlantic, may cause a weakening of the MOC, which in turn may cause a reduced average temperature in especially Europe. These transitions may happen due to tipping points, or bifurcation points, that are associated with the salt-advection feedback.

State of the art methods are not efficient enough for computing probabilities of transitions in a realistic MOC model, and therefore techniques for computing other indicators of these transitions are being developed. In this thesis, we introduced novel preconditioning and Lyapunov solution methods, and improved the efficiency of methods which can be used for computing actual transition probabilities.

Linear systems

In Chapter 3, a novel multilevel preconditioner was introduced, which is specialized for the Navier–Stokes equations which are discretized on a staggered grid. The preconditioner works by partitioning the domain into parallelepiped shaped subdomains, of which the interior can be eliminated in parallel. On the interfaces, Householder transformations are applied to decouple all but one velocity node from the pressure nodes, after which all decoupled nodes can also be eliminated in parallel. The preconditioner can then be applied recursively to the leftover coupled velocity nodes until a direct solver is used at the last level.

We showed that the preconditioner has grid-independent convergence in case the number of levels is kept constant. Increasing the number of levels with the grid size was actually more efficient in terms of time, but did not show grid-independent convergence. We can again come close to grid-independent convergence by retaining a few more velocity nodes per level.

We also found the preconditioner to be robust for a 3D lid-driven cavity problem with increasing Reynolds numbers, which leads us to believe that it will also perform well when applied in our ocean solver.

Lyapunov equations

The sensitivity of the MOC to noise around a steady state can be expressed in a probability density function. To compute this probability density function, we have to compute the covariance matrix at the steady state. This can be done by solving a generalized Lyapunov equation.

We developed a novel method for computing low-rank solutions of generalized Lyapunov equations in Chapter 4. The method works by applying a Galerkin type projection with the space built from the eigenvectors belonging to the largest eigenvalues of the residual at every iteration of the method. Most important is that the method can be restarted, which allows for less memory usage, faster iterations, and recycling of previous solutions. We showed that for an idealized 2D MOC model, our method is the most efficient method in terms of both memory usage and time.

Due to the recycling properties, the method is also very efficient for solving extended generalized Lyapunov equations, and for solving generalized Lyapunov equations during a continuation process.

Transition probabilities

In Chapter 5 we discussed what a transition probability is, and how to compute them. We defined the transition probability to be the probability of going from a neighborhood A near a deterministic steady state $\bar{\boldsymbol{x}}_A$ to a neighborhood B near a deterministic steady state $\bar{\boldsymbol{x}}_B$ within time T.

Before actually computing transition probabilities, we discussed the possibility to compute covariance ellipsoids, which can give us some idea of the variability around a steady state. We then discussed most probable transition paths, which can give us some idea of the path along which a transition may happen. Both of these ellipsoids and paths, however, do not allow us to compute transition probabilities.

To compute actual transition probabilities, we can use methods like Adaptive Multilevel Splitting (AMS), Trajectory-Adaptive Multilevel Sampling (TAMS) and Genealogical Particle Analysis (GPA). All of these methods work by resampling of the trajectories based on how close they are to the other steady state. We gave a quantitative comparison of these methods, and found that TAMS gave results close to the results obtained with direct sampling, but with a smaller variance.

Transitions in the Meridional Overturning Circulation

To compute transitions in the MOC, we used TAMS in Chapter 6, as this gave us the best results in the previous chapter. To improve on this method, we came up with a projected time stepping method, which reduces the memory usage for our idealized 2D MOC model with 96%, and the time consumption by 30%. We showed that the probability of a transition increases drastically when getting closer to a bifurcation point, and that the projected method is able to obtain the same results as standard TAMS.

Since we only looked at an idealized 2D MOC model, we can not really say anything about the transition probabilities of the MOC in the actual Atlantic ocean, but we did provide methods with which this becomes feasible.

Prospects

In the various summaries that are present in this thesis, we already mentioned several prospects that are applicable to the methods that are presented in the corresponding chapters. In this section we aim to give some prospects for the project as a whole.

Something that we briefly mentioned in Section 5.4 are most probable transition trajectories. These have already been computed for another model that is used in oceanography, which is the two-dimensional barotropic quasi-geostrophic model. Even though we did not go into this direction in this thesis, it would be interesting to also compute these trajectories for the MOC. This would give us insight in the states that exist between the present day MOC and a collapsed MOC.

To compute transition probabilities for the MOC that are more meaningful than the ones we computed with our idealized 2D model, we require a more realistic 3D ocean-climate model. Fortunately, the implementation that we used for the idealized 2D model, the I-EMIC, already allows for this (Mulder, 2019). It would be very interesting to see if we can compute transitions with this 3D model, and if we can, what the probability of such a transition is.

It would also be interesting to see how our preconditioner performs in combination with the I-EMIC. At present, the I-EMIC uses the tailored preconditioner from De Niet et al. (2007); Thies et al. (2009). We expect to achieve more stability and much better scalability with the preconditioner presented in this thesis, which would allow for simulations with a higher resolution.

PUBLICATIONS AND PREPRINTS

S. Baars, J. Viebahn, T. Mulder, C. Kuehn, F. Wubs, and H. Dijkstra. Continuation of Probability Density Functions Using a Generalized Lyapunov Approach. *Journal of Computational Physics*, 336:627–643, May 2017. doi: `10.1016/j.jcp.2017.02.021`.

S. Baars, D. Castellana, F. Wubs, and H. A. Dijkstra. Application of Adaptive Multilevel Splitting to High-Dimensional Dynamical Systems. *preprint*, 2019a.

S. Baars, M. van der Klok, W. Song, J. Thies, A. Veldman, and F. Wubs. Performance of the HYMLS Multilevel ILU Preconditioner on 3D Flow Equations. *submitted to Computers & Mathematics with Applications*, 2019b.

S. Baars, M. van der Klok, J. Thies, and F. Wubs. SMILU: a Staggered-Grid Multi-Level ILU for Steady Coupled Fluid-Transport Equations. *preprint*, 2019c.

B. Carpentieri, J. Liao, M. Sosonkina, A. Bonfiglioli, and S. Baars. Using the VBARMS Method in Parallel Computing. *Parallel Computing*, 57:197–211, Sep 2016. doi: `10.1016/j.parco.2016.01.005`.

H. A. Dijkstra, A. Tantet, J. Viebahn, E. Mulder, M. Hebbink, D. Castellana, H. van den Pol, J. Frank, S. Baars, F. Wubs, M. Chekroun, and C. Kuehn. A Numerical Framework To Understand Transitions in High-Dimensional Stochastic Dynamical Systems. *Dynamics and Statistics of the Climate System*, 1(1), 2016. doi: `10.1093/climsys/dzw003`.

T. E. Mulder, S. Baars, F. W. Wubs, and H. A. Dijkstra. Stochastic Marine Ice Sheet Variability. *Journal of Fluid Mechanics*, 843:748–777, Mar 2018. doi: `10.1017/jfm.2018.148`.

W. Song, F. Wubs, J. Thies, and S. Baars. Numerical Bifurcation Analysis of a 3D Turing-Type Reaction–Diffusion Model. *Communications in Nonlinear Science and Numerical Simulation*, 60:145–164, Jul 2018. doi: `10.1016/j.cnsns.2018.01.003`.

SOFTWARE

This chapter contains a list of software that has been developed, or has been contributed to, as part of this work.

FVM A finite volume package which contains implementations of a lid-driven cavity, a differentially heated cavity, Rayleigh-Bénard convection, and convection in a differentially heated rotating cavity. `https://bitbucket.org/hymls/hymls/src/master/fvm`

HYMLS A Hybrid direct/iterative solver for the Jacobian of the incompressible Navier-Stokes equations on structured grids. The implementation of the preconditioner and the domain decomposition are described in this thesis. `https://github.com/nlesc-smcm/hymls`

I-EMIC An Implicit Earth System Model of Intermediate Complexity. The I-EMIC contains a number of implicit submodels coupled through a modular framework. At the center of the coupled model is the implicit primitive equation ocean model THCM that has been used in the past to study bifurcations in the thermohaline circulation. This package interfaces to parallel C++ versions of RAILS and the methods for computing transition probabilities that were discussed in this thesis. `https://github.com/nlesc-smcm/i-emic`

JDQZ++ A templated C++ implementation of the JDQZ generalized eigenvalue problem solver. `https://github.com/erik808/jdqzpp`

MOxUnit A lightweight unit test framework for Matlab and GNU Octave. `https://github.com/MOxUnit/MOxUnit`

OpenBLAS An optimized BLAS library based on GotoBLAS2 1.13, BSD version. `http://www.openblas.net`

PHIST A Pipelined Hybrid Parallel Iterative Solver Toolkit. PHIST provides implementations of and interfaces to block iterative solvers for sparse linear and eigenvalue problems. `https://bitbucket.org/essex/phist`

RAILS An implementation of the Residual Approximation-based Iterative Lyapunov Solver as described in this thesis in Matlab and a parallel implementation in C++. `https://github.com/Sbte/RAILS`

Transitions A Matlab implementation of the methods for computing transition probabilities as described in this thesis. `https://github.com/Sbte/transitions`

Trilinos The Trilinos Project is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. A unique design feature of Trilinos is its focus on packages. `https://trilinos.org`

BIBLIOGRAPHY

J. I. Aliaga, M. Bollhöfer, A. F. Martín, and E. S. Quintana-Ortí. Design, Tuning and Evaluation of Parallel Multilevel ILU Preconditioners. In *High Performance Computing for Computational Science VECPAR 2008*, pages 314–327. Springer Berlin Heidelberg, 2008. doi: `10.1007/978-3-540-92859-1_28`.

K. Atkinson. *An Introduction To Numerical Analysis*. Wiley, New York, 1988. ISBN 9780471624899

S. Baars, J. Viebahn, T. Mulder, C. Kuehn, F. Wubs, and H. Dijkstra. Continuation of Probability Density Functions Using a Generalized Lyapunov Approach. *Journal of Computational Physics*, 336:627–643, May 2017. doi: `10.1016/j.jcp.2017.02.021`.

S. Baars, D. Castellana, F. Wubs, and H. A. Dijkstra. Application of Adaptive Multilevel Splitting to High-Dimensional Dynamical Systems. *preprint*, 2019a.

S. Baars, M. van der Klok, W. Song, J. Thies, A. Veldman, and F. Wubs. Performance of the HYMLS Multilevel ILU Preconditioner on 3D Flow Equations. *submitted to Computers & Mathematics with Applications*, 2019b.

S. Baars, M. van der Klok, J. Thies, and F. Wubs. SMILU: a Staggered-Grid Multi-Level ILU for Steady Coupled Fluid-Transport Equations. *preprint*, 2019c.

R. H. Bartels and G. W. Stewart. Solution of the Matrix Equation AX + XB = C [f4]. *Communications of the ACM*, 15(9):820–826, Sep 1972. doi: `10.1145/361573.361582`.

G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 2000. ISBN 9780511800955. doi: `10.1017/cbo9780511800955`.

E. Bavier, M. Hoemmen, S. Rajamanickam, and H. Thornquist. Amesos2 and Belos: Direct and Iterative Solvers for Large Sparse Linear Systems. *Scientific Programming*, 20(3):241–255, 2012. doi: `10.1155/2012/243875`.

P. Benner, V. Mehrmann, V. Sima, S. V. Huffel, and A. Varga. SLICOT-A Subroutine Library in Systems and Control Theory. *Applied and Computational Control, Signals, and Circuits*, pages 499–539, 1999. doi: `10.1007/978-1-4612-0571-5_10`.

P. Benner, P. Kürschner, and J. Saak. Self-Generating and Efficient Shift Parameters in ADI Methods for Large Lyapunov and Sylvester Equations. *Electron. Trans. Numer. Anal.*, 43:142–162, 2014. doi: `10.17617/2.2071065`.

M. Benzi and M. A. Olshanskii. An Augmented Lagrangian-Based Approach To the Oseen Problem. *SIAM Journal on Scientific Computing*, 28(6):2095–2113, Jan 2006. doi: `10.1137/050646421`.

M. Benzi, G. H. Golub, and J. Liesen. Numerical Solution of Saddle Point Problems. *Acta Numerica*, 14:1–137, May 2005. doi: `10.1017/s0962492904000212`.

R. H. Bisseling. *Parallel Scientific Computation*. Oxford University Press, Mar 2004. ISBN 9780198529392. doi: `10.1093/acprof:oso/9780198529392.001.0001`.

M. Bollhöfer and Y. Saad. Multilevel Preconditioners Constructed From Inverse-Based ILUs. *SIAM Journal on Scientific Computing*, 27(5):1627–1650, Jan 2006. doi: `10.1137/040608374`.

E. F. F. Botta and F. W. Wubs. Matrix Renumbering ILU: an Effective Algebraic Multilevel ILU Preconditioner for Sparse Matrices. *SIAM Journal on Matrix Analysis and Applications*, 20(4):1007–1026, Jan 1999. doi: `10.1137/s0895479897319301`.

F. Bouchet and J. Reygner. Generalisation of the Eyring–Kramers Transition Rate Formula To Irreversible Diffusion Processes. *Annales Henri Poincaré*, 17 (12):3499–3532, Jun 2016. doi: `10.1007/s00023-016-0507-4`.

F. Bouchet, J. Laurie, and O. Zaboronski. Langevin Dynamics, Large Deviations and Instantons for the Quasi-Geostrophic Model and Two-Dimensional Euler Equations. *Journal of Statistical Physics*, 156(6):1066–1092, Jul 2014. doi: `10.1007/s10955-014-1052-5`.

C.-E. Bréhier, M. Gazeau, L. Goudenège, T. Lelièvre, and M. Rousset. Unbiasedness of Some Generalized Adaptive Multilevel Splitting Algorithms. *The Annals of Applied Probability*, 26(6):3559–3601, Dec 2016. doi: `10.1214/16-aap1185`.

G. Carey and R. Krishnan. Convergence of Iterative Methods in Penalty Finite Element Approximation of the Navier-Stokes Equations. *Computer Methods in Applied Mechanics and Engineering*, 60(1):1–29, Jan 1987. doi: `10.1016/0045-7825(87)90127-7`.

F. Cérou and A. Guyader. Adaptive Multilevel Splitting for Rare Event Analysis. *Stochastic Analysis and Applications*, 25(2):417–443, Feb 2007. doi: `10.1080/07362990601139628`.

F. Cérou, A. Guyader, T. Lelièvre, and D. Pommier. A Multiple Replica Approach To Simulate Reactive Trajectories. *The Journal of Chemical Physics*, 134 (5):054108, Feb 2011. doi: `10.1063/1.3518708`.

P. Cessi. A Simple Box Model of Stochastically Forced Thermohaline Flow. *Journal of Physical Oceanography*, 24(9):1911–1920, Sep 1994. doi: `10.1175/1520-0485(1994)024⟨1911:asbmos⟩2.0.co;2`.

M. D. Chekroun, H. Liu, and S. Wang. *Stochastic Parameterizing Manifolds and Non-Markovian Reduced Equations*. Springer International Publishing, 2015. ISBN 9783319125206. doi: `10.1007/978-3-319-12520-6`.

G. Cowan. *Statistical Data Analysis*. Clarendon Press, 1998. ISBN 0198501552.

M. A. Crisfield. *Nonlinear Finite Element Analysis of Solids and Structures, Vol 1: Basic Concepts*, chapter 1. Wiley, 1991.

B. Cushman-Roisin and J.-M. Beckers. *Introduction To Geophysical Fluid Dynamics - Physical and Numerical Aspects*. Elsevier, 2011. ISBN 9780120887590

E. C. Cyr, J. N. Shadid, and R. S. Tuminaro. Stabilization and Scalable Block Preconditioning for the Navier–Stokes Equations. *Journal of Computational Physics*, 231(2):345–363, Jan 2012. doi: `10.1016/j.jcp.2011.09.001`.

T. Damm. Direct Methods and ADI-Preconditioned Krylov Subspace Methods for Generalized Lyapunov Equations. *Numerical Linear Algebra with Applications*, 15(9):853–871, Nov 2008. doi: `10.1002/nla.603`.

A. DasGupta, T. T. Cai, and L. D. Brown. Interval Estimation for a Binomial Proportion. *Statistical Science*, 16(2):101–133, May 2001. doi: `10.1214/ss/1009213286`.

T. A. Davis and E. P. Natarajan. Algorithm 907. *ACM Transactions on Mathematical Software*, 37(3):1–17, Sep 2010. doi: `10.1145/1824801.1824814`.

H. A. Dijkstra. *Nonlinear Physical Oceanography*. Springer Netherlands, 2005. ISBN 9781402022623. doi: `10.1007/1-4020-2263-8`.

H. A. Dijkstra. *Nonlinear Climate Dynamics*. Cambridge University Press, 2013. ISBN 9781139034135. doi: `10.1017/cbo9781139034135`.

H. A. Dijkstra, L. M. Frankcombe, and A. S. von der Heydt. A Stochastic Dynamical Systems View of the Atlantic Multidecadal Oscillation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1875):2543–2558, Jul 2008. doi: `10.1098/rsta.2008.0031`.

H. A. Dijkstra, F. W. Wubs, A. K. Cliffe, E. Doedel, I. F. Dragomirescu, B. Eckhardt, A. Y. Gelfgat, A. L. Hazel, V. Lucarini, A. G. Salinger, E. T. Phipps, J. Sanchez-Umbria, H. Schuttelaars, L. S. Tuckerman, and U. Thiele. Numerical Bifurcation Methods and Their Application To Fluid Dynamics: Analysis Beyond Simulation. *Communications in Computational Physics*, 15(01): 1–45, Jan 2014. doi: `10.4208/cicp.240912.180613a`.

H. A. Dijkstra, A. Tantet, J. Viebahn, E. Mulder, M. Hebbink, D. Castellana, H. van den Pol, J. Frank, S. Baars, F. Wubs, M. Chekroun, and C. Kuehn. A Numerical Framework To Understand Transitions in High-Dimensional Stochastic Dynamical Systems. *Dynamics and Statistics of the Climate System*, 1(1), 2016. doi: `10.1093/climsys/dzw003`.

P. D. Ditlevsen and S. J. Johnsen. Tipping Points: Early Warning and Wishful Thinking. *Geophysical Research Letters*, 37(19):n/a–n/a, Oct 2010. doi: `10.1029/2010gl044486`.

V. Druskin and V. Simoncini. Adaptive Rational Krylov Subspaces for Large-Scale Dynamical Systems. *Systems & Control Letters*, 60(8):546–560, Aug 2011. doi: `10.1016/j.sysconle.2011.04.013`.

V. Druskin, V. Simoncini, and M. Zaslavsky. Adaptive Tangential Interpolation in Rational Krylov Subspaces for MIMO Dynamical Systems. *SIAM Journal on Matrix Analysis and Applications*, 35(2):476–498, Jan 2014. doi: `10.1137/120898784`.

W. E, W. Ren, and E. Vanden-Eijnden. String Method for the Study of Rare Events. *Physical Review B*, 66(5), Aug 2002. doi: `10.1103/physrevb.66.052301`.

W. E, W. Ren, and E. Vanden-Eijnden. Minimum Action Method for the Study of Rare Events. *Communications on Pure and Applied Mathematics*, 57(5):637–656, 2004. doi: `10.1002/cpa.20005`.

W. E, W. Ren, and E. Vanden-Eijnden. Simplified and Improved String Method for Computing the Minimum Energy Paths in Barrier-Crossing Events. *The Journal of Chemical Physics*, 126(16):164103, Apr 2007. doi: `10.1063/1.2720838`.

H. Elman, V. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. A Taxonomy and Comparison of Parallel Block Multi-Level Preconditioners for the Incompressible Navier–Stokes Equations. *Journal of Computational Physics*, 227(3):1790–1808, Jan 2008. doi: `10.1016/j.jcp.2007.09.026`.

H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers*. Oxford University Press, Jun 2014. ISBN 9780199678792. doi: `10.1093/acprof:oso/9780199678792.001.0001`.

H. C. Elman, D. J. Silvester, and A. J. Wathen. Performance and Analysis of Saddle Point Preconditioners for the Discrete Steady-State Navier–Stokes Equations. *Numerische Mathematik*, 90(4):665–688, Feb 2002. doi: `10.1007/s002110100300`.

H. Eyring. The Activated Complex in Chemical Reactions. *The Journal of Chemical Physics*, 3(2):107–115, 1935. doi: `10.1063/1.1749604`.

Y. Feldman and A. Y. Gelfgat. Oscillatory Instability of a Three-Dimensional Lid-Driven Flow in a Cube. *Physics of Fluids*, 22(9):093602, Sep 2010. doi: `10.1063/1.3487476`.

M. I. Freidlin and A. D. Wentzell. *Random Perturbations of Dynamical Systems*. Springer-Verlag, New York,, 1998. doi: `10.1007/978-1-4612-0611-8`.

F. Freitas, J. Rommes, and N. Martins. Gramian-Based Reduction Method Applied To Large Sparse Power System Descriptor Models. *IEEE Transactions on Power Systems*, 23(3):1258–1270, Aug 2008. doi: `10.1109/tpwrs.2008.926693`.

C. W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*. Springer Berlin Heidelberg, 1985. ISBN 9783662024522. doi: `10.1007/978-3-662-02452-2`.

P. W. Glynn and W. Whitt. The Asymptotic Efficiency of Simulation Estimators. *Operations Research*, 40(3):505–520, Jun 1992. doi: `10.1287/opre.40.3.505`.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996. ISBN 9781421407944

T. Grafke, T. Schäfer, and E. Vanden-Eijnden. *Long Term Effects of Small Random Perturbations on Dynamical Systems: Theoretical and Computational Tools*, pages 17–55. Springer New York, 2017. ISBN 9781493969692. doi: `10.1007/978-1-4939-6969-2_2`.

S. M. Griffies. *Fundamentals of Ocean Climate Models*. Princeton University Press, 2004. ISBN 0691187126. doi: `10.2307/j.ctv301gzg`.

P. Hänggi, P. Talkner, and M. Borkovec. Reaction-Rate Theory: Fifty Years After Kramers. *Reviews of Modern Physics*, 62(2):251–341, Apr 1990. doi: `10.1103/revmodphys.62.251`.

X. He, C. Vuik, and C. M. Klaij. Combining the Augmented Lagrangian Preconditioner With the Simple Schur Complement Approximation. *SIAM Journal on Scientific Computing*, 40(3):A1362–A1385, Jan 2018. doi: `10.1137/17m1144775`.

G. Henkelman and H. Jónsson. Improved Tangent Estimate in the Nudged Elastic Band Method for Finding Minimum Energy Paths and Saddle Points. *The Journal of Chemical Physics*, 113(22):9978–9985, Dec 2000. doi: `10.1063/1.1323224`.

G. Henkelman, B. P. Uberuaga, and H. Jónsson. A Climbing Image Nudged Elastic Band Method for Finding Saddle Points and Minimum Energy Paths. *The Journal of Chemical Physics*, 113(22):9901–9904, Dec 2000. doi: `10.1063/1.1329672`.

M. A. Heroux, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, K. S. Stanley, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, and R. P. Pawlowski. An Overview of the Trilinos Project. *ACM Transactions on Mathematical Software*, 31(3):397–423, Sep 2005. doi: `10.1145/1089014.1089021`.

D. J. Higham. Mean-Square and Asymptotic Stability of the Stochastic Theta Method. *SIAM Journal on Numerical Analysis*, 38(3):753–769, Jan 2000. doi: `10.1137/s003614299834736x`.

D. J. Higham. An Algorithmic Introduction To Numerical Simulation of Stochastic Differential Equations. *SIAM Review*, 43(3):525–546, Jan 2001. doi: `10.1137/s0036144500378302`.

H. Kahn and T. E. Harris. Estimation of Particle Transmission By Random Sampling. *National Bureau of Standards applied mathematics series*, 12:27–30, 1951.

G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1): 359–392, Jan 1998. doi: `10.1137/s1064827595287997`.

D. Kay, D. Loghin, and A. Wathen. A Preconditioner for the Steady-State Navier–Stokes Equations. *SIAM Journal on Scientific Computing*, 24(1):237–256, Jan 2002. doi: `10.1137/s106482759935808x`.

H. B. Keller. Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems. In P. H. Rabinowitz, editor, *Applications of Bifurcation Theory*, pages 359–384. Academic Press, New York, U.S.A., 1977.

C. M. Klaij and C. Vuik. SIMPLE-Type Preconditioners for Cell-Centered, Colocated Finite Volume Discretization of Incompressible Reynolds-Averaged Navier–Stokes Equations. *International Journal for Numerical Methods in Fluids*, 71(7):830–849, May 2012. doi: `10.1002/fld.3686`.

D. Kleinman. On an Iterative Technique for Riccati Equation Computations. *IEEE Transactions on Automatic Control*, 13(1):114–115, Feb 1968. doi: `10.1109/tac.1968.1098829`.

P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer Berlin Heidelberg, 1992. ISBN 9783662126165. doi: `10.1007/978-3-662-12616-5`.

M. van der Klok. *Skew Partitioning for the Hybrid Multilevel Solver*. Bachelor's thesis, University of Groningen, Jul 2017.

H. Kramers. Brownian Motion in a Field of Force and the Diffusion Model of Chemical Reactions. *Physica*, 7(4):284–304, Apr 1940. doi: `10.1016/s0031-8914(40)90098-2`.

C. Kuehn. A Mathematical Framework for Critical Transitions: Bifurcations, Fast-Slow Systems and Stochastic Dynamics. *Physica D: Nonlinear Phenomena*, 240(12):1020–1035, Jun 2011. doi: `10.1016/j.physd.2011.02.012`.

C. Kuehn. Deterministic Continuation of Stochastic Metastable Equilibria Via Lyapunov Equations and Ellipsoids. *SIAM Journal on Scientific Computing*, 34(3):A1635–A1658, Jan 2012. doi: `10.1137/110839874`.

C. Kuehn. Numerical Continuation and SPDE Stability for the 2D Cubic-Quintic Allen–Cahn Equation. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):762–789, Jan 2015. doi: `10.1137/140993685`.

C. Lanczos. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *Journal of Research of the National Bureau of Standards*, 45(4):255, Oct 1950. doi: `10.6028/jres.045.026`.

L. Landau and E. Lifshitz. *Fluid Mechanics*. Elsevier, 1959. ISBN 9780080291420

J. Laurie and F. Bouchet. Computation of Rare Transitions in the Barotropic Quasi-Geostrophic Equations. *New Journal of Physics*, 17(1):015009, Jan 2015. doi: `10.1088/1367-2630/17/1/015009`.

T. M. Lenton. Early Warning of Climate Tipping Points. *Nature Climate Change*, 1(4):201–209, Jun 2011. doi: `10.1038/nclimate1143`.

T. M. Lenton, H. Held, E. Kriegler, J. W. Hall, W. Lucht, S. Rahmstorf, and H. J. Schellnhuber. Tipping Elements in the Earth's Climate System. *Proceedings of the National Academy of Sciences*, 105(6):1786–1793, Feb 2008. doi: `10.1073/pnas.0705414105`.

T. Lestang, F. Ragone, C.-E. Bréhier, C. Herbert, and F. Bouchet. Computing Return Times Or Return Periods With Rare Event Algorithms. *Journal of Statistical Mechanics: Theory and Experiment*, 2018(4):043213, Apr 2018. doi: `10.1088/1742-5468/aab856`.

Y. Liu, M. Jacquelin, P. Ghysels, and X. S. Li. Highly Scalable Distributed-Memory Sparse Triangular Solution Algorithms. *2018 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing*, pages 87–96, Jan 2018. doi: `10.1137/1.9781611975215.9`.

R. März. Canonical Projectors for Linear Differential Algebraic Equations. *Computers & Mathematics with Applications*, 31(4-5):121–135, Feb 1996. doi: `10.1016/0898-1221(95)00224-3`.

M. van der Mheen, H. A. Dijkstra, A. Gozolchiani, M. den Toom, Q. Feng, J. Kurths, and E. Hernandez-Garcia. Interaction Network Based Early Warning Indicators for the Atlantic MOC Collapse. *Geophysical Research Letters*, 40(11):2714–2719, Jun 2013. doi: `10.1002/grl.50515`.

A. H. Monahan, J. Alexander, and A. J. Weaver. Stochastic Models of the Meridional Overturning Circulation: Time Scales and Patterns of Variability. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1875):2525–2542, Jul 2008. doi: `10.1098/rsta.2008.0045`.

P. D. Moral. *Mean Field Simulation for Monte Carlo Integration*. Chapman and Hall/CRC, May 2013. ISBN 9781466504172. doi: `10.1201/b14924`.

P. D. Moral and J. Garnier. Genealogical Particle Analysis of Rare Events. *The Annals of Applied Probability*, 15(4):2496–2534, Nov 2005. doi: `10.1214/105051605000000566`.

T. E. Mulder. *Design and Bifurcation Analysis of Implicit Earth System Models*. PhD thesis, Utrecht University, Jun 2019.

T. E. Mulder, S. Baars, F. W. Wubs, and H. A. Dijkstra. Stochastic Marine Ice Sheet Variability. *Journal of Fluid Mechanics*, 843:748–777, Mar 2018. doi: `10.1017/jfm.2018.148`.

A. Navarra and V. Simoncini. *A Guide to Empirical Orthogonal Functions for Climate Data Analysis*. Springer Netherlands, 2010. ISBN 9789048137022. doi: `10.1007/978-90-481-3702-2`.

A. de Niet, F. Wubs, A. T. van Scheltinga, and H. A. Dijkstra. A Tailored Solver for Bifurcation Analysis of Ocean-Climate Models. *Journal of Computational Physics*, 227(1):654–679, Nov 2007. doi: `10.1016/j.jcp.2007.08.006`.

A. C. de Niet and F. W. Wubs. Numerically Stable LDLT-Factorization of F-Type Saddle Point Matrices. *IMA Journal of Numerical Analysis*, 29(1):208–234, Feb 2008. doi: `10.1093/imanum/drn005`.

T. Penzl. A Cyclic Low-Rank Smith Method for Large Sparse Lyapunov Equations. *SIAM Journal on Scientific Computing*, 21(4):1401–1418, Jan 1999. doi: `10.1137/s1064827598347666`.

S. Rahmstorf. The Thermohaline Ocean Circulation: a System With Dangerous Thresholds? *Climatic Change*, 46(3):247–256, Aug 2000. doi: `10.1023/a:1005648404783`.

J. Rolland and E. Simonnet. Statistical Behaviour of Adaptive Multilevel Splitting Algorithms in Simple Models. *Journal of Computational Physics*, 283: 541–558, Feb 2015. doi: `10.1016/j.jcp.2014.12.009`.

J. Rolland, F. Bouchet, and E. Simonnet. Computing Transition Rates for the 1-D Stochastic Ginzburg–Landau–Allen–Cahn Equation for Finite-Amplitude Noise With a Rare Event Algorithm. *Journal of Statistical Physics*, 162(2):277–311, Nov 2015. doi: `10.1007/s10955-015-1417-4`.

M. N. Rosenbluth and A. W. Rosenbluth. Monte Carlo Calculation of the Average Extension of Molecular Chains. *The Journal of Chemical Physics*, 23(2): 356–359, Feb 1955. doi: `10.1063/1.1741967`.

G. Rubino and B. Tuffin. *Rare Event Simulation using Monte Carlo Methods*. John Wiley & Sons, Ltd, Mar 2009. ISBN 9780470772690. doi: `10.1002/9780470745403`.

Y. Saad. Numerical Solution of Large Lyapunov Equations. In *Signal Processing, Scattering and Operator Theory, and Numerical Methods, Proc. MTNS-89*, pages 503–511. Birkhauser, 1990.

Y. Saad and M. H. Schultz. GMRES: a Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, Jul 1986. doi: `10.1137/0907058`.

J. Saak, M. Köhler, and P. Benner. M-M.E.S.S.-1.0.1 - the Matrix Equation Sparse Solver Library, Apr. 2016. doi: `10.5281/zenodo.50575`.

M. Sala and M. A. Heroux. Robust algebraic preconditioners using IFPACK 3.0. Technical report, Sandia National Laboratories, Jan 2005.

M. Sala, K. S. Stanley, and M. A. Heroux. On the Design of Interfaces To Sparse Direct Solvers. *ACM Transactions on Mathematical Software*, 34(2):1–22, Mar 2008. doi: `10.1145/1326548.1326551`.

T. P. Sapsis and P. F. Lermusiaux. Dynamically Orthogonal Field Equations for Continuous Stochastic Dynamical Systems. *Physica D: Nonlinear Phenomena*, 238(23-24):2347–2360, Dec 2009. doi: `10.1016/j.physd.2009.09.017`.

P. D. Sardeshmukh and P. Sura. Reconciling Non-Gaussian Climate Statistics With Linear Dynamics. *Journal of Climate*, 22(5):1193–1207, Mar 2009. doi: `10.1175/2008jcli2358.1`.

M. J. Schmeits and H. A. Dijkstra. Bimodal Behavior of the Kuroshio and the Gulf Stream. *Journal of Physical Oceanography*, 31(12):3435–3456, Dec 2001. doi: `10.1175/1520-0485(2001)031⟨3435:bbotka⟩2.0.co;2`.

A. Segal, M. ur Rehman, and C. Vuik. Preconditioners for Incompressible Navier-Stokes Solvers. *Numerical Mathematics: Theory, Methods and Applications*, 2010. doi: `10.4208/nmtma.2010.33.1`.

S. D. Shank, V. Simoncini, and D. B. Szyld. Efficient Low-Rank Solution of Generalized Lyapunov Equations. *Numerische Mathematik*, 134(2):327–342, Nov 2015. doi: `10.1007/s00211-015-0777-7`.

W. P. Sijp, J. D. Zika, M. d'Orgeville, and M. H. England. Revisiting Meridional Overturing Bistability Using a Minimal Set of State Variables: Stochastic Theory. *Climate Dynamics*, 43(5-6):1661–1676, Nov 2013. doi: `10.1007/s00382-013-1992-5`.

V. Simoncini. A New Iterative Method for Solving Large-Scale Lyapunov Matrix Equations. *SIAM Journal on Scientific Computing*, 29(3):1268–1288, Jan 2007. doi: `10.1137/06066120x`.

V. Simoncini. Available Software, 2016. URL `http://www.dm.unibo.it/~simoncin/software.html`.

E. Simonnet. Combinatorial Analysis of the Adaptive Last Particle Method. *Statistics and Computing*, 26(1-2):211–230, Jul 2014. doi: `10.1007/s11222-014-9489-6`.

G. L. G. Sleijpen and F. W. Wubs. Exploiting Multilevel Preconditioning Techniques in Eigenvalue Computations. *SIAM Journal on Scientific Computing*, 25(4):1249–1272, Jan 2004. doi: `10.1137/s1064827599361059`.

J. Slingo and T. Palmer. Uncertainty in Weather and Climate Prediction. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1956):4751–4767, Oct 2011. doi: `10.1098/rsta.2011.0161`.

H. Stommel. Thermohaline Convection With Two Stable Regimes of Flow. *Tellus B*, 13(2), May 1961. doi: `10.3402/tellusb.v13i2.12985`.

T. Stykel and V. Simoncini. Krylov Subspace Methods for Projected Lyapunov Equations. *Applied Numerical Mathematics*, 62(1):35–50, Jan 2012. doi: `10.1016/j.apnum.2011.09.007`.

P. Sura and S. T. Gille. Stochastic Dynamics of Sea Surface Height Variability. *Journal of Physical Oceanography*, 40(7):1582–1596, Jul 2010. doi: `10.1175/2010jpo4331.1`.

J. Thies and F. Wubs. Design of a Parallel Hybrid Direct/Iterative Solver for CFD Problems. *2011 IEEE Seventh International Conference on eScience*, Dec 2011. doi: `10.1109/escience.2011.60`.

J. Thies, F. Wubs, and H. A. Dijkstra. Bifurcation Analysis of 3D Ocean Flows Using a Parallel Fully-Implicit Ocean Model. *Ocean Modelling*, 30(4):287–297, Jan 2009. doi: `10.1016/j.ocemod.2009.07.005`.

A. Timmermann and G. Lohmann. Noise-Induced Transitions in a Simplified Model of the Thermohaline Circulation. *Journal of Physical Oceanography*, 30 (8):1891–1900, Aug 2000. doi: `10.1175/1520-0485(2000)030⟨1891:nitias⟩2.0.co;2`.

M. den Toom, H. A. Dijkstra, and F. W. Wubs. Spurious Multiple Equilibria Introduced By Convective Adjustment. *Ocean Modelling*, 38(1-2):126–137, Jan 2011. doi: `10.1016/j.ocemod.2011.02.009`.

U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, Nov 2000. ISBN 9780127010700

M. Tuma. A Note on the LDLT Decomposition of Matrices From Saddle-Point Problems. *SIAM Journal on Matrix Analysis and Applications*, 23(4):903–915, Jan 2002. doi: `10.1137/s0895479897321088`.

E. Vanden-Eijnden and M. Heymann. The Geometric Minimum Action Method for Computing Minimum Energy Paths. *The Journal of Chemical Physics*, 128(6):061103, Feb 2008. doi: `10.1063/1.2833040`.

M. Vellinga and R. A. Wood. Global Climatic Impacts of a Collapse of the Atlantic Thermohaline Circulation. *Climatic change*, 54(3):251–267, 2002. doi: `10.1023/a:1016168827653`.

R. Verstappen and M. Dröge. A Symmetry-Preserving Cartesian Grid Method for Computing a Viscous Flow Past a Circular Cylinder. *Comptes Rendus Mécanique*, 333(1):51–57, Jan 2005. doi: `10.1016/j.crme.2004.09.021`.

R. Verstappen and A. Veldman. Symmetry-Preserving Discretization of Turbulent Flow. *Journal of Computational Physics*, 187(1):343–368, May 2003. doi: `10.1016/s0021-9991(03)00126-8`.

H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, 2003. ISBN 9780511615115. doi: `10.1017/cbo9780511615115`.

G. Walin. The Thermohaline Circulation and the Control of Ice Ages. *Palaeogeography, Palaeoclimatology, Palaeoecology*, 50(1):323–332, Jan 1985. doi: `10.1016/s0031-0182(85)80020-1`.

A. J. Wathen. Preconditioning. *Acta Numerica*, 24:329–376, Apr 2015. doi: `10.1017/s0962492915000021`.

W. Weijer and H. A. Dijkstra. A Bifurcation Study of the Three-Dimensional Thermohaline Ocean Circulation: the Double Hemispheric Case. *Journal of Marine Research*, 59(4):599–631, Jul 2001. doi: `10.1357/002224001762842208`.

J. Wouters and F. Bouchet. Rare Event Computation in Deterministic Chaotic Systems Using Genealogical Particle Analysis. *Journal of Physics A: Mathematical and Theoretical*, 49(37):374002, Aug 2016. doi: `10.1088/1751-8113/49/37/374002`.

F. W. Wubs and J. Thies. A Robust Two-Level Incomplete Factorization for (Navier–)Stokes Saddle Point Matrices. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1475–1499, Oct 2011. doi: `10.1137/100789439`.

X. Zhou, W. Ren, and W. E. Adaptive Minimum Action Method for the Study of Rare Events. *The Journal of Chemical Physics*, 128(10):104111, Mar 2008. doi: `10.1063/1.2830717`.

SUMMARY

There is a strong variability in the everyday weather related to the development of high- and low-pressure fields. These developments do not have anything to do with the external solar forcing, but are due to the internal variability of the atmospheric flow. The time scale of this variability of 3-7 days is determined by the nonlinear processes in the atmosphere itself. Similar processes happen in every part of the climate system on varying time scales. A high-frequency process is the weather as described above; a low-frequency process is the change in land-ice distribution. In the ocean, internal variability causes formation of ocean eddies and the meandering of ocean currents such as the Gulf Stream. Interaction between all the different processes on different time scales may result in internal variability on other time scales that is not present in decoupled systems. This makes such processes very challenging to study, and difficult to understand. Therefore it is often a good idea to take a step back, and look at a more simplified model to make sure one is at least able to understand the results, and then apply this newly acquired knowledge to the fully coupled system.

We work with such a simplified model of the Meridional Overturning Circulation (MOC). The MOC consists of a global ‘conveyor belt’ of ocean currents, which are driven by wind stress forces and fluxes of heat and freshwater at the surface. In the Atlantic ocean, it consists of surface currents that transport relatively light water toward high latitudes, deep water currents going in the opposite direction, and sinking and upwelling processes that connect these two. The circulation system contains two overturning cells: one in the north with North Atlantic Deep Water (NADW) and one in the south with Antarctic Bottom Water (AABW).

Since the first model proposed by Stommel, many model studies have shown that the MOC may be sensitive to variability in the freshwater forcing. In a global coupled climate model by Vellinga and Wood the NADW circulation collapses and recovers after 120 years. This is because weakening the MOC by introducing more freshwater in the North Atlantic (melting of the Greenland ice sheet) leads to a reduced northward saltwater transport, which in turn amplifies the freshwater perturbation.

A state of the MOC with a strongly reduced heat transport may have large consequences for the global climate. Cooling of a few degrees may be observed in Europe, which in turn may lead to growing glaciers and then global cooling. Therefore, an estimate of the probability of MOC transitions is crucial for prediction of a collapse of the MOC and with that a rapid climate change.

This collapse may occur due to the existence of a tipping point associated with the salt-advection feedback. Tipping points exist due to the presence of multiple stable steady states for the same parameter values. Due to unresolved small-scale variability, however, transitions may even be observed before a tipping point is reached. This unresolved variability is often represented as noise. We aimed to develop methods which can be used for studying transition behavior in the MOC.

To be able to observe these transitions between stable steady states, we first need to be able to compute the steady states themselves. They can be computed by using time integration, which is often expensive, especially if steady states have to be computed for multiple parameter values. Instead one can apply a method called (pseudo-arclength) continuation, which can compute the steady states directly by applying Newton's method. This can speed up the computation of steady states considerably. Pseudo-arclength continuation is especially useful for computing unstable steady states, i.e. steady states for which a small disturbance causes the state to converge to a different steady state. Time integration methods are usually incapable of computing these unstable steady states.

During the application of Newton's method, many linear systems of equations have to be solved. For the MOC, these are specifically equations that include the Navier–Stokes equations discretized on a staggered grid. Direct (sparse) solvers are not practical since a typical ocean model involves millions of unknowns leading to a huge memory requirement for storing the factorization and an enormous amount of time to compute it. For such problems, iterative methods are preferred, e.g. Krylov subspace methods with suitable preconditioning to ensure robustness, fast convergence and accuracy of the final approximate solution.

Preconditioners that are often advocated include standard additive Schwarz domain-decomposition, multigrid with aggressive coarsening and strong smoothers (e.g. ILU), and 'block preconditioners' that use an approximate block LU factorization and some approximation of the Schur complement associated with the pressure unknowns, e.g. SIMPLEC, LSC, and PCD.

Another class of Schur complement methods is obtained when eliminating the interior of geometric partitions (or subdomains) and constructing a suitable approximation of the reduced system on the separator. In this class of methods, we presented a multilevel preconditioner for the Navier–Stokes equations discretized on a staggered grid. The resulting operator acts in the divergence-free space, which allows the method to handle the saddle-point structure of the system in a natural way.

After computing steady states of the MOC, we are interested in its sensitivity to noise. The sensitivity around a steady state can be determined from the probability density function. It is well known that this probability density function can be computed from the solution of a generalized Lyapunov equation. Direct methods for solving a generalized Lyapunov equation such as Bartels–Stewart algorithm are based on dense matrix solvers and hence inapplicable for large systems. Other existing methods which use low-rank approximations might also become expensive for high-dimensional problems, particularly when trying to use previous initial guesses along a continuation branch. We, therefore, presented a novel method for computing the solution of generalized Lyapunov equations that is particularly well suited for our ocean problem in a continuation context. The method works by applying a Galerkin type projection with the space built from the eigenvectors belonging to the largest eigenvalues of the residual at every iteration of the method. Most important is that the method can be restarted, which allows for less memory usage, faster iterations, and recycling of previous solutions. We showed that for an idealized 2D MOC model, our method is the most efficient method in terms of both memory usage and time.

A shortcoming to this above approach is that it only describes the sensitivity to noise around a steady state. To study the more global phenomenon of transitions between steady states, stochastic time integration is required. Applying a standard Monte Carlo method, however, is way too expensive, especially for high-dimensional systems and when the probability of a transition is small. Multiple methods exist to work around this problem by applying some form of resampling. To improve on these methods, we came up with a projected time stepping method, which reduces the memory usage for our idealized 2D MOC model with 96%, and the time consumption by 30%. We showed that the probability of a transition increases drastically when getting closer to a bifurcation point, and that the projected method is able to obtain the same results as standard methods.

Since we only looked at an idealized 2D MOC model, we can not really say anything about the transition probabilities of the MOC in the actual Atlantic ocean, but we did provide methods with which this becomes feasible.

SAMENVATTING

Ontwikkelingen van hoge- en lagedrukgebieden zorgen voor een sterke variabiliteit van het weer. Deze ontwikkelingen worden niet veroorzaakt door stralingsenergie afkomstig van de zon, maar zijn te wijten aan de interne variabiliteit van de atmosferische stroming. De tijdschaal van 3 tot 7 dagen van deze variabiliteit wordt bepaald door niet-lineaire processen in de atmosfeer zelf. Soortgelijke processen vinden op verschillende tijdschalen plaats in elk deel van het klimaatsysteem. Een hoogfrequent proces is het weer zoals hierboven beschreven; een laagfrequent proces is de verandering in de dekkingsgraad van ijs op land. In de oceaan veroorzaakt interne variabiliteit vorming van wervels en het meanderen van oceaanstromingen zoals de Golfstroom. Interactie tussen alle verschillende processen op verschillende tijdschalen kan leiden tot interne variabiliteit op andere tijdschalen die niet aanwezig is in de afzonderlijke systemen. Dit maakt dergelijke processen zeer lastig om te bestuderen en moeilijk om te begrijpen. Daarom is het vaak een goed idee om een stap terug te doen en te kijken naar een eenvoudiger model om er zeker van te zijn dat men in ieder geval in staat is om de resultaten te begrijpen, en om deze nieuw verworven kennis vervolgens toe te kunnen passen op het volledig gekoppelde systeem.

We werken met een dergelijk vereenvoudigd model van de meridionale omwentelingscirculatie (MOC). De MOC bestaat uit een mondiale 'transportband' van zeestromingen, die wordt aangedreven door windspanningskrachten en fluxen van warmte en zoet water aan het oppervlak. In de Atlantische oceaan bestaat het uit oppervlaktestromen die relatief licht water naar hoge breedtegraden transporteren, diepe waterstromen die in de tegenovergestelde richting gaan en zink- en opwellingprocessen die deze twee verbinden. Het circulatiesysteem bevat twee omwentelingscellen: één in het noorden met

Noord-Atlantisch diep water (NADW) en één in het zuiden met Antarctisch bodemwater (AABW).

Sinds het eerste door Stommel voorgestelde model hebben veel modelstudies aangetoond dat de MOC gevoelig kan zijn voor variabiliteit in de zoetwaterforcering. In een gekoppeld klimaatmodel van Vellinga en Wood stort de NADW-circulatie in en herstelt zich na 120 jaar. De verzwakking van de MOC door de introductie van meer zoet water in de Noord-Atlantische oceaan (het smelten van de Groenlandse ijskap) leidt namelijk tot een verminderd zoutwatertransport naar het noorden, wat op zijn beurt de zoetwaterverstoring versterkt.

Een toestand van de MOC met een sterk gereduceerd warmtetransport kan grote gevolgen hebben voor het mondiale klimaat. In Europa kan een afkoeling van enkele graden worden waargenomen, die op haar beurt kan leiden tot groeiende gletsjers en vervolgens tot wereldwijde afkoeling. Daarom is een schatting van de kans op MOC-transities cruciaal voor het voorspellen van een ineenstorting van de MOC en daarmee een snelle klimaatverandering.

Deze ineenstorting kan optreden vanwege het bestaan van een kantelpunt die gerelateerd is aan de zout-advectieterugkoppeling. Kantelpunten bestaan vanwege de aanwezigheid van meerdere stabiele evenwichtstoestanden voor dezelfde parameterwaarden. Vanwege kleinschalige variabiliteit die niet in het model wordt gevangen kunnen echter transities worden waargenomen zelfs voordat een kantelpunt wordt bereikt. Deze kleinschalige variabiliteit wordt vaak beschreven door middel van ruis. Ons doel was om methoden te ontwikkelen die kunnen worden gebruikt voor het bestuderen van transitiegedrag in de MOC.

Om deze overgangen tussen stabiele evenwichtstoestanden te kunnen waarnemen, moeten we eerst de evenwichtstoestanden zelf kunnen berekenen. Ze kunnen worden berekend met behulp van tijdsintegratie, wat vaak duur is, vooral als evenwichtstoestanden berekend moeten worden voor meerdere parameterwaarden. In plaats daarvan kan men een methode toepassen genaamd (pseudo-arclength) continuatie, die de evenwichtstoestanden rechtstreeks kan berekenen door de methode van Newton toe te passen. Dit kan de berekening van evenwichtstoestanden aanzienlijk versnellen. Pseudo-arclength continuatie is vooral nuttig voor het berekenen van onstabiele evenwichtstoestanden, d.w.z. evenwichtstoestanden waarbij een kleine verstoring ervoor zorgt dat de toestand convergeert naar een andere evenwichtstoestand. Tijdsintegratiemethoden zijn meestal niet in staat om deze onstabiele evenwichtstoestanden te berekenen.

Bij de toepassing van de methode van Newton moeten veel lineaire stelsels van vergelijkingen worden opgelost. Voor de MOC zijn dit specifiek vergelijkingen die de Navier-Stokes-vergelijkingen omvatten, gediscretiseerd op een versprongen rooster. Directe (ijle) oplossingsmethoden zijn niet praktisch, omdat een typisch oceaanmodel miljoenen onbekenden omvat, wat leidt tot een grote geheugenvereiste voor het opslaan van de factorisatie en een enorme hoeveelheid tijd die nodig is om deze te berekenen. Voor dergelijke problemen wordt de voorkeur gegeven aan iteratieve methoden, bijvoorbeeld de Krylov-deelruimtemethoden met geschikte preconditionering om de robuustheid, snelle convergentie en nauwkeurigheid van de uiteindelijke benaderende oplossing te garanderen.

Preconditioneerders die vaak worden bepleit zijn standaard additieve Schwarz domein-decompositie, multigrid met agressieve verruwing en sterke gladstrijkers (bijv. ILU), en 'blok-preconditioneerders' die een benaderende blok-LU-factorisatie en een benadering van het Schur-complement geassocieerd met de druk-onbekenden gebruiken, bijvoorbeeld SIMPLEC, LSC en PCD.

Een andere klasse van Schur-complementmethoden wordt verkregen wanneer het interieur van geometrische partities (of subdomeinen) wordt geëlimineerd en een geschikte benadering van het gereduceerde systeem op de separator wordt geconstrueerd. In deze klasse van methoden stelden we een meerlaagse preconditioneerder voor de Navier–Stokes-vergelijkingen die gediscretiseerd zijn op een versprongen rooster. De resulterende operator werkt in de divergentievrije ruimte, waardoor de methode de zadelpuntstructuur van het systeem op een natuurlijke manier kan behandelen.

Na het berekenen van evenwichtstoestanden van de MOC zijn we geïnteresseerd in de gevoeligheid voor ruis. De gevoeligheid rond een evenwichtstoestand kan worden bepaald aan de hand van de kansdichtheid. Het is algemeen bekend dat deze kansdichtheid kan worden berekend uit de oplossing van een gegeneraliseerde Lyapunov-vergelijking. Directe methoden voor het oplossen van een gegeneraliseerde Lyapunov-vergelijking zoals het Bartels–Stewart-algoritme zijn gebaseerd op solvers voor volle matrices en zijn daarom niet toepasbaar voor grote systemen. Bestaande methoden die gebruik maken van benaderingen met een lage rang kunnen ook duur worden voor hoogdimensionale problemen, met name bij het gebruik van eerdere initiële schattingen langs een continuatietak. We ontwikkelden daarom een nieuwe methode voor het berekenen van de oplossing van gegeneraliseerde Lyapunov-vergelijkingen die bijzonder goed geschikt is voor ons oceaanprobleem in een continuatiecontext. De methode werkt door een Galerkin-type projectie toe te passen met de ruimte die is opgebouwd uit de eigenvectoren die behoren tot de grootste eigenwaarden van het residu in elke iteratie van de methode. Het belangrijkste is dat de methode kan worden herstart, wat zorgt voor minder geheugengebruik, snellere iteraties, en wat hergebruik van eerdere oplossingen toestaat. We hebben laten zien dat voor een geïdealiseerd 2D MOC-model onze methode de meest efficiënte methode is in termen van zowel geheugengebruik als tijd.

Een tekortkoming van deze aanpak is dat deze alleen de gevoeligheid voor ruis rond een evenwichtstoestand beschrijft. Om het meer globale fenomeen van transities tussen evenwichtstoestanden te bestuderen, is een stochastische tijdsintegratie nodig. Het toepassen van een standaard Monte Carlo methode is echter veel te duur, vooral voor hoogdimensionale systemen en wanneer de kans op een transitie klein is. Er bestaan meerdere methoden om dit probleem te omzeilen door een of andere vorm van resampling toe te passen. Om deze methoden te verbeteren, hebben we een geprojecteerde tijdstapmethode bedacht, die het geheugengebruik voor ons geïdealiseerde 2D MOC-model met 96% en het tijdsverbruik met 30% vermindert. We toonden aan dat de kans op een transitie drastisch toeneemt wanneer we dichter bij een bifurcatiepunt komen, en dat de geprojecteerde methode in staat is dezelfde resultaten te verkrijgen als standaardmethoden.

Omdat we alleen naar een geïdealiseerd 2D MOC-model hebben gekeken, kunnen we niet echt iets zeggen over de transitieskansen van de MOC in de werkelijke Atlantische oceaan, maar we hebben wel methoden ontwikkeld waarmee dit haalbaar wordt.

SOAMENVATTING

Klaainschoalege veranderlekhaid ien t klimoat kin grode effecten hebben op mondioale oceoancirculoatsie. Veurbeelden van zukke klaainschoalege veraanderns binnen schommelingen ien houveulhaid zuit wotter deur t smelten van t ies op Gruinlaand. Dizze schommelingen kinnen aanlaaiden wezen tot n òfnoame van haile globoale oceoancirculoatsie, wat op zien beurt òfkouln van n poar groad kin geven ien Uropoa. Om der achter te kommen houveul kaans wie moaken op zo'n klimoatomslag, mouten der berekens doan wòrren mit grootschoalege oceoanmodèllen. Jammer genog binnen bestoande rekentechnieken nait vlug genog en doarom mouten der nije technieken bedòcht wòrren. Ien dit proufschrift stellen wie drij veur. Eerste is n methode om grode systemen van vergeliekens op te lössen, twijde n methode om gevuileghaid veur schommelingen wied genog vot van t omslaggebied te bepoalen, en leste methode berekent waarkelke kaans op n klimoatomslag ien grootschoalege modèllen. Wie loaten zain hou dizze methoden waarken op n idealiseerd twijdimensionoal oceoancirculoatsiemodèl. Ien toukomst kinnen dizze methoden bruukt wòrren om echte omslagkaansen ien realistische drijdimensionoale klimoatmodèllen mit n hoge rezeluutsie te bereken. Veur n oetgebraaidere soamenvatting verwiezen wie joe geern deur noar Nederlandse soamenvatting.

ACKNOWLEDGMENTS

Welcome to the acknowledgments, which is very likely to be one of the very few parts of my thesis that you will ever read, maybe together with the summary. I do not mind this at all, since I am also guilty of this habit, especially when receiving a thesis that is unrelated to my field of research. For this reason, I also provide the summary not only in Dutch, as seems to be common, but also in English and Gronings.

First and foremost, I would like to thank my supervisor Fred Wubs. I do not think that any PhD student could ever wish for a better supervisor. You gave me full freedom to pursue my own interests, even if they did not fully overlap with your initial plans for this project, and were always interested in the progress that I made. And whenever I had any questions, you were there in your office, ready to answer them. I also enjoyed the travels we made together to conferences and to Jonas in Cologne, especially the cycling trip we made to Bonn and back.

I would also like to thank Henk Dijkstra, who acted as my second supervisor, for always being enthusiastic and for providing us with a vision, which ultimately led me to the topics that were discussed in this thesis. I look forward to the continuation of working together with both of you, Fred and Henk, in the coming years.

Furthermore, my thanks goes to my promotor, Roel Verstappen, and the assessment committee, consisting of Rob Bisseling, Daan Crommelin and Arjan van der Schaft, for their thorough reading of my thesis and their helpful comments.

Now I get to the two people I spent most time with during the past few years, even before we started with our PhDs, which are Ronald and Erik. We have been sharing an office on and off for the past six years or so, and it has been a pleasure. Thanks especially for not being too annoyed when I distract you for no particular reason other than to talk to you.

Thanks to Erik for providing me with the finest details on the subjects of mechanical keyboards(!), modular synthesizers, coffee grinding, growing edible mushrooms and baking bread. Thanks to Ronald for your love for seals, powder, Beijum and bananas. Thanks to Daniele, Christian and Jonas for our pleasant long distance collaborationships during the past few years. Thanks to Mark for being such an excellent student and for coming up with the rotated parallelepiped, which solved all of our problems. Thanks to David for the lengthy discussions over a cup of coffee (thanks also for the coffee). Ahmad, Cristóbal, Donglin, Henk, Hugo, Jeremías, Jigar, Larissa, Luca, Maurits, Nico, Paolo, Peter, Reza, Sourabh, Weiyan, thanks for the nice conversations we had often during, but not limited to, the lunch breaks.

Thanks to Paulus, Sjieuwe, Peter, Lanting and Jan, for being my friends since secondary school, but also while studying together with me at this university with the exception of Jan, who went to Delft. Thanks to the *mooie gekken* at T.F.V. 'Professor Francken' for taking klaverjassen to a next level and for the countless hours of fun we had together. Thanks to the Marks of the s[ck]rip(t|t?c)ie for the random acts of programming under the supervision of Kathinka. Thanks to Roel for teaching me how to perform a scout rush.

Finally, thanks to my parents, for still keeping up with me after so long. Thanks for all of the Christmas dinners. Thanks for never having forced me to do anything, and for always being supportive. Thanks for always being interested in what I do, even if you do not fully understand it. The same holds for my sister, who in turn I often do not fully understand either, but who is also very dear to me. Also thanks to Bono and Waldo for respectively warming and eating my feet. I am happy to have such a wonderful family.

CURRICULUM VITAE

Sven Baars was born on August 15, 1990 in Eenrum, The Netherlands. After completing his primary education in Eenrum, he attended secondary education at the Praedinius Gymnasium in Groningen. From 2008 to 2014 he was a student at the University of Groningen, where he obtained his bachelor's and master's degree (cum laude) in the field of numerical mathematics. During this time, he also did an internship at the German Aerospace Center in Cologne.

From 2015 to 2019, Sven was a PhD student at the university of Groningen in the group of prof. dr. ir. R.W.C.P. Verstappen under the supervision of dr. ir. F.W. Wubs. The topic of his research was numerical methods for studying transition probabilities in stochastic ocean-climate models.