Chapter 4
Boosting Ship Simulations at Marin

David Kok[1], Vivi Rottschäfer[2], David van Keulen[3], George A.K. van Voorn[4], Manu Kalia[5], Bas van't Hof[2], L. Gerard van Willigenburg[4]

Abstract:
Two approaches to significantly reduce ship model simulation times at Marin *are presented. Combining them is shown to provide a most interesting and general perspective for improvement. One approach makes use of a highly simplified ship model that can be partly solved analytically. The other applies proper orthogonal decomposition (POD) to reduce the ship model currently used at* Marin*. POD is demonstrated using a model that is more convenient for presentation and implementation than the* Marin *ship model. Arguments are provided why POD is expected to also work for* Marin *ship simulations.*

Keywords: *Singular Value Decomposition, Proper Orthogonal Decomposition, Model Order Reduction, Method of Averaging, Asymptotic approximation*

[1]Mathematical Institute Leiden, The Netherlands
[2]Leiden University, The Netherlands
[3]Fontys Hogeschool, The Netherlands
[4]Wageningen University, The Netherlands
[5]Twente University, The Netherlands

4.1 Introduction

Certain ship simulations performed at MARIN solve a detailed model containing Navier-Stokes type dynamics using computational fluid dynamics (CFD). Currently, these are computationally too expensive. This problem relates to the fact that ship speed is varying very slowly compared to the rotation of the propeller and surrounding water, leading to very different time-scales in the simulation. The challenge is to improve computational efficiency. This will enable valuable ship simulations in regards to improving the design of ships as well as the energy efficiency of steering ships. In studying this challenge we took two approaches that in the end can be combined. For one approach, we developed a highly simplified ship model that reveals the origin of the computational inefficiency. In the other approach, we started from the current computational fluid dynamics computations and searched for ways to improve their efficiency. One method that can be used is proper orthogonal decomposition (POD) [6]. This will provide a way to reduce the model thereby improving computational efficiency. In this paper we first present the highly simplified model revealing the essence of the inefficiency of the ship simulations in Section 4.2. In Section 4.5, POD is presented and applied, not directly to the ship models of MARIN, but to a model that is more suitable for presentation and implementation. Arguments are given why we expect this method to be applicable to the ship models of MARIN as well. In the conclusions we argue that combining the outcomes of both approaches results in recommendations for improving the computational efficiency that are most promising and practical.

4.2 Highly simplified ship model

In this section we present a highly simplified model describing the speed of a ship, driven by a high-frequency propeller in water, near cruising speed. This model provides a computationally very cheap alternative to a full numerical solution of partial differential equations (PDE's) describing water, propeller and ship, while still preserving some of the observed features of the long-term behaviour of the solution.

We model the dynamics of the ship by using Newton's second law, which states that the time derivative of the speed v is given by

$$\frac{dv}{dt} = \frac{1}{m} F_{\text{total}} = \frac{1}{m}(F_{\text{prop}} - F_{\text{water}}) \tag{4.1}$$

with F_{total} the total force acting on the ship. This force consists of two components, a

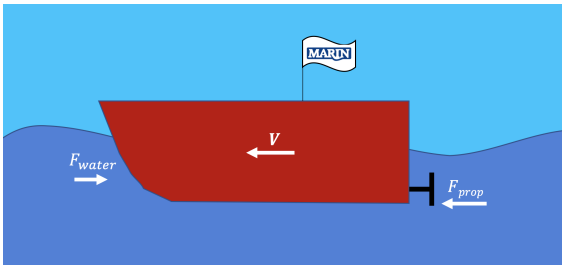

Figure 4.1: Highly simplified model of ship movement.

frictional force F_{water} exerted on the ship by the surrounding water, and the driving force F_{prop} delivered by the propeller. We rescale and thereby set $m = 1$. For the friction force we assume a simple expression consisting of a linear and a quadratic component

$$F_{\text{water}}(v) = p_5 v + p_6 v^2, \tag{4.2}$$

where p_5 and p_6 are positive constants. We note that the propeller force depends on the pressure of the water near the bow of the ship (averaged over the surface of the propeller). When the ship speeds up, this pressure increases, allowing the propeller to deliver more force. In general this means we model the net effect of the water pressure on the propeller as $p = p(v, t)$. Here it should be noted that we need the explicit time dependence of this pressure term. It represents the fact that the ship generates water currents near the bow which the propeller slices through, changing the total pressure across the blades as a function of the angle of the blade.

We expect the behaviour of the ship to be sensitive to the precise interplay between the water currents near the propeller and the movement of the ship itself. To account for this interplay, we model the flow of the water with two components - one rotating along with the propeller (the component generated by the propeller itself) and one stationary (the time-averaged flow generated by the ship moving at cruising speed). In general we expect the water flow profile, and the full long-term solution, to be periodic with a period equal to that of the propeller. Therefore we will use a first-harmonic approximation of the full interplay between water flow and propeller force. This is described by

$$F_{\text{prop}}(v, t) = \left(1 + \frac{1}{2} p_1 \cos\left(\omega t/\epsilon\right)\right)(p_2 + p_3 v). \tag{4.3}$$

Here ϵ is a small parameter introduced to explicitly indicate the occurrence of two timescales. The rotation frequency ω/ϵ is much higher than the typical timescale on which we expect the speed of the ship to change. Furthermore p_2 represents the average water pressure and p_3 the rate of change of water pressure with respect to velocity.

Finally p_1 determines the relative contribution of the two types of water flow near the propeller to the propulsion. All these constants are positive. The full model is then given by

$$\frac{dv}{dt} = \left(1 + \frac{1}{2}p_1 \cos(\omega t/\epsilon)\right)(p_2 + p_3 v) - p_5 v - p_6 v^2. \tag{4.4}$$

4.3 Approximate analytical solutions

4.3.1 Ansatz

Based on numerical simulations of equation (4.4) we expect the solution to consist of a quick oscillation around a slowly varying average, where the amplitude of the oscillation also varies slowly. Furthermore, the amplitude of this oscillation is small.
The standard argument of perturbation theory suggests that, for sufficiently small ϵ, the solution $v(t;\epsilon)$ of equation (4.4) is analytic in ϵ and can therefore be expressed as

$$v(t;\epsilon) = \sum_{n=0}^{\infty} \epsilon^n v_n(t). \tag{4.5}$$

We also expect the solutions of 4.4 to converge to periodic solutions with the same periodicity as the driving term. Restricting to periodic solutions implies that $v(t+T;\epsilon) = v(t;\epsilon)$ for all t, ϵ where $T = \frac{2\pi}{\omega/\epsilon} = \frac{2\pi\epsilon}{\omega}$ is the period of the driving term. However, if this holds for all ϵ, then for fixed t the power series expressions for $v(t;\epsilon)$ and $v(t+T;\epsilon)$ must be identical. Therefore all functions $v_n(t)$ in equation (4.5) above are also periodic with period T. Introduction of a Fourier expansion allows us to write

$$v(t;\epsilon) = \sum_{n=0}^{\infty} \sum_{k=-\infty}^{k=\infty} \epsilon^n v_{n,k} \exp(ki\omega t/\epsilon). \tag{4.6}$$

Numerical simulations show that at order ϵ^0, i.e. leading order, the solution is not oscillating rapidly, so $v_{0,k} = 0$ for $k \neq 0$. This leads to the following Ansatz/approximation

$$v(t) = a(t) + \epsilon b(t)E + \epsilon\overline{b(t)}\bar{E} + O(\epsilon^2, \epsilon E^2) + \text{c.c.} \tag{4.7}$$

where we used the big-O notation and introduced $E = \exp(i\omega t/\epsilon)$ while c.c. denotes complex conjugation. Substituting (4.7) in equation (4.4) gives

$$\begin{aligned}\frac{dv}{dt} &= \frac{da}{dt}(t) + \left(\epsilon\frac{db}{dt}(t)E + i\omega b(t)E\right) + O(\epsilon^2, \epsilon E^2) + c.c. \\ &\quad - p_6(a(t) + O(\epsilon))^2 + c.c. \\ &= \left(p_2 + p_3 a(t) - p_5 a(t) - p_6 a(t)^2\right) + \left(\frac{1}{2}p_1(p_2 + p_3 a(t))\right) E \\ &\quad + O(\epsilon) + c.c.\end{aligned}$$

Collecting terms in orders of ϵ and per frequency we find

$$\mathcal{O}(\epsilon^0 E^0): \; p_2 + (p_3 - p_5)a - p_6 a^2 = \frac{da}{dt} \tag{4.8}$$

$$\mathcal{O}(\epsilon^0 E^1): \; \frac{1}{2}p_1(p_2 + p_3 a) = i\omega b \tag{4.9}$$

where we have suppressed the explicit time dependence from our notation.

4.3.2 Asymptotic behaviour

The first-order ODE (4.8), has an attracting fixed point

$$a_{\text{lim}} = \frac{p_3 - p_5 + \sqrt{(p_3 - p_5)^2 + 4p_2 p_6}}{2p_6}. \tag{4.10}$$

There is also a second fixed point at a negative value of a, which is not only unphysical but also repelling. The solution of (4.8) converges to the positive fixed point. In the $t \to \infty$ limit we find from (4.9) that

$$b_{\text{lim}} = -i\frac{\frac{1}{2}p_1(p_2 + p_3 a_{\text{lim}})}{\omega} \tag{4.11}$$

Some additional remarks are in order:

- The limiting amplitude b_{lim} is purely imaginary, which represents that the movement of the ship is a quarter period out of phase with the driving force. For the actual amplitude of the oscillations around the average we can simply take the modulus. Note that since our Ansatz adds the complex conjugate of the oscillating terms the full solution is still real-valued.

- We have ignored all higher-order terms. So $\overline{v}(t) \equiv a(t) + \epsilon b(t)E + c.c.$ is only an approximation of the real solution. We expect this to be accurate only if ϵ is small, i.e. if $T = \frac{2\pi\epsilon}{\omega}$ is small.

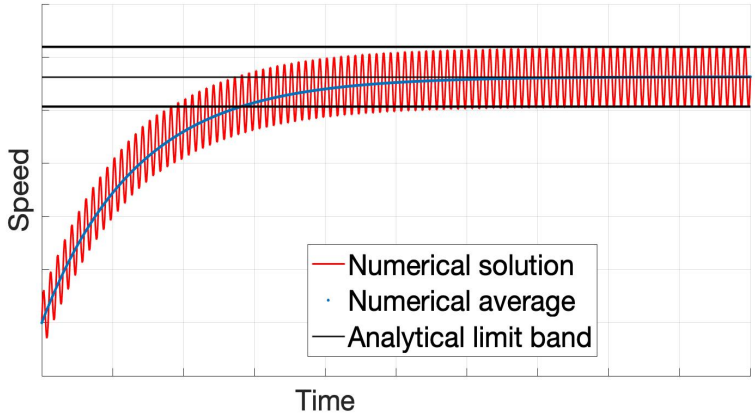

Figure 4.2: A simulation of equation (4.4) *in red with a short-time average in blue. The upper and lower black lines denote* $a_{lim} \pm b_{lim}$ *respectively, with* a_{lim} *plotted in between. The simulation parameters are* $p_1 = 1, p_2 = 1, p_3 = 0.5, T = 0.1, p_5 = 1$ *and* $p_6 = 0.1$.

4.3.3 Comparing numerical results and analytical approximations

In this section we compare the approximations given above with direct numerical integration of equation (4.4). To that end we use the MATLAB ode23s-solver (see the *Appendix* for the code) for a duration of $100T$. In Fig. 4.2 we compare the numerical results with the constants a_{lim} and $a_{\text{lim}} \pm b_{\text{lim}}$ computed with the expressions given above

We also compute the analytical limit parameters and the numerical asymptotic behaviour for a range of values of p_1 fixing the values of $p_2, \ldots, p_6$ and find that the two are in good agreement (see Table 4.1). In Table 4.2 similar results are given but now for various values of p_2.

The tables confirm that Ansatz is quite good. This suggests that the method of averaging might be an effective approach to simplify the full PDE model. This suggestion is explored in the next section.

p_1	a_{lim}	numerical	b_{lim}	numerical
0.1	1.53	1.54	0.0028	0.0075
0.2	1.53	1.55	0.0056	0.0069
0.3	1.53	1.55	0.0084	0.0095
0.4	1.53	1.54	0.011	0.012
0.5	1.53	1.55	0.014	0.004
0.6	1.53	1.54	0.017	0.018
0.7	1.53	1.55	0.020	0.020
0.8	1.53	1.53	0.022	0.024
0.9	1.53	1.54	0.025	0.026
1.0	1.53	1.54	0.028	0.028
1.1	1.53	1.54	0.031	0.031
1.2	1.53	1.56	0.034	0.034
1.3	1.53	1.54	0.037	0.036
1.4	1.53	1.56	0.039	0.040
1.5	1.53	1.55	0.042	0.042

Table 4.1: Comparison of simulated and analytical limit (after $100T$) behaviour of equation (4.4)*. The simulation parameters are $p_2 = 1, p_3 = 0.5, T = 0.1, p_5 = 1$ and $p_6 = 0.1$ where p_1 varies.*

p_2	a_{lim}	numerical	b_{lim}	numerical
0.1	0.19	0.20	0.003	0.003
0.2	0.37	0.38	0.006	0.006
0.3	0.54	0.54	0.009	0.009
0.4	0.70	0.70	0.012	0.012
0.5	0.85	0.86	0.015	0.015
0.6	1.00	1.01	0.018	0.018
0.7	1.14	1.15	0.020	0.020
0.8	1.27	1.29	0.023	0.023
0.9	1.41	1.42	0.026	0.026
1.0	1.53	1.54	0.028	0.028

Table 4.2: Comparison of simulated and analytical limit (after $100T$) behaviour of equation (4.4)*. The simulation parameters are $p_1 = 1, p_3 = 0.5, T = 0.1, p_5 = 1$ and $p_6 = 0.1$ where p_2 varies.*

4.4 The 'Method of averaging'

Here we show that the results obtained above are identical to those that would be obtained using the method of averaging [4]. This method can be applied to systems of the form

$$\frac{dx}{dt} = \epsilon f(x,t), \tag{4.12}$$

where $x \in D \subset \mathbb{R}$ for some domain D and f and $\frac{\partial f}{\partial x}$ are continuous and bounded in $D \times [0,\infty)$. The function f must be T-periodic in t and T independent of ϵ. Then, upon introducing the averaged system

$$\frac{dy}{dt} = \epsilon \frac{1}{T} \int_0^T f(y,s)ds, \tag{4.13}$$

the Averaging Theorem states that

$$x(t) - y(t) = \mathcal{O}(\epsilon), \tag{4.14}$$

on the time-scale $\frac{1}{\epsilon}$ This averaging method can be applied to the toy model of Section 4.2 by introducing the timescale $\tau \equiv \frac{t}{\epsilon}$. Then, equation (4.4) transforms to

$$\frac{dv}{d\tau} = \epsilon \left(\left(1 + \frac{1}{2} p_1 \cos(\omega\tau) \right) (p_2 + p_3 v) - p_5 v - p_6 v^2 \right) \tag{4.15}$$

Now taking the average over one period of the right-hand-side with v fixed, we remove the oscillating exponentials and this reduces the equation to

$$\frac{d\tilde{v}}{d\tau} = \epsilon \left(p_2 + (p_3 - p_5)\tilde{v} - p_6 \tilde{v}^2 \right). \tag{4.16}$$

Note that, to leading order, this is equal to (4.8). Since the method of averaging doesn't heavily rely on specifics of the dynamical system, and our analysis of the previous and upcoming sections suggest that the results obtained from this method are accurate, it is likely that we can apply this strategy to the full PDE model to obtain a simplified time-averaged model.

4.4.1 Method of averaging for Marin's ships

This section provides a rough sketch of how the method of averaging could be applied to the Marin models.

The Marin model of ship and water consists of a model of the flow around the propeller, coupled to a model of the flow around the ship, as shown in Figure 4.3. The propeller-model is a cylindrical, rotating domain, and the ship-flow model moves with the ship. Apart from their moving domains both models have constant geometry.

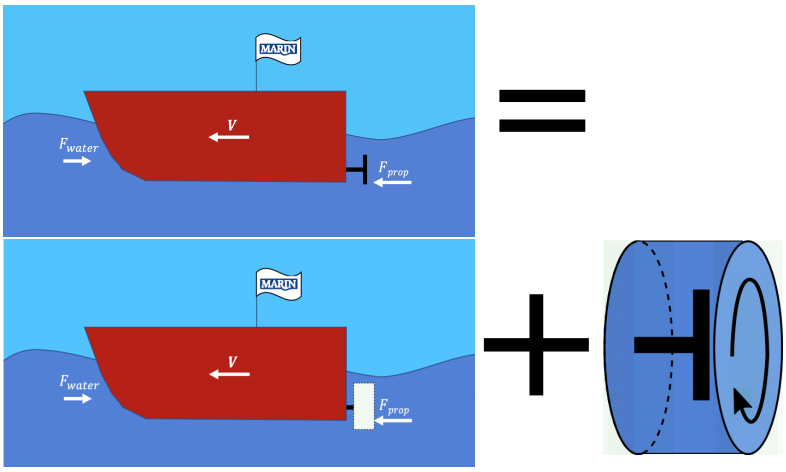

Figure 4.3: Decomposition of the Marin ship model into two coupled models having very different time-scales: one of the ship and water and one of the propeller and its surrounding water.

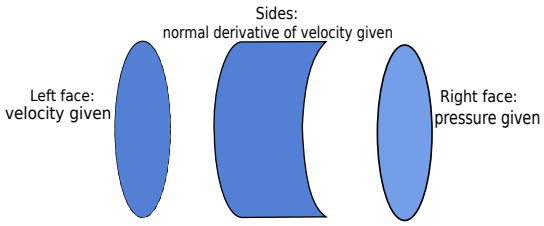

Figure 4.4: The propeller model can be solved for given boundary conditions along its cylindrical domain.

Before applying the averaging-method, we replace the propeller-model by a forcing term. To do this, a set of boundary conditions is chosen, as shown in Figure 4.4. Each of the chosen boundary conditions is constant in time in the ship-flow model, and is therefore periodic in the (rotating) propeller model. Next, the flow equations are solved in the propeller model until the solution is (almost) periodic. When the solution is (almost) periodic, the solution is recorded at the boundary and time-averaged over a period. After the chosen boundary conditions have been processed this way, and the results stored in a table, interpolation is used to approximate the solution on the boundary of the propeller-model for any given boundary condition. Now, the propeller model can be replaced by the interpolation tables thus obtained, which is a time-averaged forcing without the small time scale inherent to the full propeller model.

The interpolation tables obtained this way are very similar to the *actuator disc* used

at Marin to obtain initial estimates of the water flow around a ship. Like the interpolation tables, the actuator disc is a forcing term without a short time scale, that provides the thrust necessary to propel the ship. However, the approach roughly sketched in this section produces a forcing term directly based on the model of the propeller used in reality (including all the details of cavitions etcetera), and is expected to be much more accurate than the actuator disc.

4.4.2 Introducing multiple time-steps in ship simulation

During full-scale simulations, MARIN noticed that the solution changes rapidly in only approximately 30% of all the grid-points.

This observation suggests to not update all model states at the same rate. Instead, the computational domain should be split in a 'fast changing' part and a 'slowly varying' part, so that a larger time step could be applied in the 'slowly varying' part of the domain.

4.5 Model order reduction using POD

Since computational fluid dynamics (CFD), used by MARIN for ship simulation, is too computationally expensive, in the last decade or so there is an increased interest in producing *Reduced Order Models* (ROMs). These reduced models are intended to keep the dominant flow dynamics while reducing the size and hence the computational costs of fluid dynamics models, including RANS models and large Eddy simulations.

In this Section we discuss model order reduction based on the method of POD (Proper Orthogonal Decomposition; [6]). Let the dynamics of the system be described by

$$\dot{x} = f(x) \ , \ x \in \mathbb{R}^N \ , \tag{4.17}$$

where x is the vector with continuous state variables, $f(\cdot)$ are smooth functions, and $N \approx 10^8$ in the case of the MARIN model. Numerical simulation of the full model is costly, and therefore the simulations done so far only cover the small time period of transient behaviour from initial conditions. From the transient behaviour stage of the simulation snapshot data can be taken, put into the matrix

$$W = (w_1 \cdots w_M) \ , \tag{4.18}$$

of dimensions $N \times M$, where w_i are the vectors containing the snapshot data, and M is the number of snapshots. Keep in mind that each snapshot vector has the same length as x (namely N) and hence represents several GB of data, while $M \lll N$.

The matrix W will now be used for singular value decomposition (SVD) as a basis for linearisation. The most accurate approximation would be to use the SVD

$$W = U\Sigma V^* \,, \tag{4.19}$$

where U is a unitary $N \times N$ matrix, V^* is the conjugate transpose of a unitary $M \times M$ matrix, and Σ is a diagonal $N \times M$ matrix containing non-negative real numbers, the so-called singular values. In practice there will be, say, 600 snapshots, conveniently chosen at equidistant time intervals to cover a time span with information-rich model dynamics. This leads to a matrix of about 4000 GB. As a result, the matrices U and Σ would be large also. The singular values can also be obtained as the eigenvalues of the matrix

$$R = W^T W \,, \tag{4.20}$$

which has dimensions $M \times M$, and hence is much less costly to handle, at the price of losing some accuracy. There are standard packages available for extracting the eigenvalues λ_i of matrix R, where $i = 0, 1, \ldots, M$. In SVD the singular values are ordered from large to small. For POD we select only the top-ranking singular values, indicated by index j, where $j = 0, 1, \ldots, P$, with $P \ll M$. The cut-off will be determined by the orders of magnitude the eigenvalues differ. Below we will demonstrate for a test case that the cut-off can potentially be very low, resulting in a reduction where $P \lll M \lll N$. For instance, [5] reduced a Navier-Stokes model for driven cavity flows with high Reynolds number using POD, and found reasonable approximations using the first 20 singular values. For the MARIN model, it remains to be seen what an acceptable P could be.

The Galerkin projection of the full model does not yet include the nonlinear terms, and the approximation of the nonlinear terms still depends on N though. To complete the model order reduction, the nonlinear terms need to be approximated as well. Several techniques are discussed in the literature to estimate the nonlinear terms. Among them, precomputing techniques and empirical interpolation methods. See for instance [2] for a modern treatment of POD.

In general, the scheme for model order reduction is as follows:

- Convert the non-linear PDE to a full, discretized system based on ODEs with dimension N, and solve this system to produce snapshots (this step has already been performed by MARIN);
- Take snapshots from the simulation data;
- Determine the POD basis using SVD;

- Use Galerkin projection to produce the reduced discretized system consisting of ODEs, with linear terms of dimension $M \ll N$ but non-linear terms still of dimension N. This saves memory and enhances the accuracy;
- Use a method like pre-computing to approximate the non-linearities and produce a reduced discretized system of ODEs with linear terms of dimension $M \ll N$ and non-linear terms of dimension $Q \ll N$. This improves efficiency, i.e., it saves time.

4.6 Numerical experiment using the Lorenz 96 model

We use the Lorenz 96[6] model (see [3]) as a test model to evaluate the plausibility and feasibility of the approach. The model is given as

$$\dot{X}_i = -X_{i-2}X_{i-1} + X_{i-1}X_{i+1} - X_i + F \,, \tag{4.21}$$

where $i = \{1, 2, 3 \dots N\}$ for arbitrary N and F is some forcing. For the testing, we assume $N = 999$. We know that for $F = 1.2$, the system exhibits a stable periodic orbit, which we plot in Figure 4.5 **A**. Using POD, we construct a reduced order model (ROM) of dimension $k = 10 \ll N$ to reconstruct this periodic solution. In Figure 4.5 **B**, we show the periodic solution exhibited by the ROM, which is qualitatively very similar to that of the original system. In Figure 4.5 **C**, we plot the 2-norm of the state variables for every time step. We see that the reconstructed solution stabilizes quickly to a periodic regime with constant 2-norm, as exhibited by the original Lorenz-96 system.

This experiment demonstrates that the POD approach significantly reduces model order when we reconstruct periodic solutions. As general polynomial systems can be reduced to quadratic systems using lifting transformations (see [1]), we expect similar behaviour in other smooth non-linear systems as well.

[6]Note that the model was never published by Lorenz in 1996, but was finally presented at the ECMWF 2006 meeting on predictability.

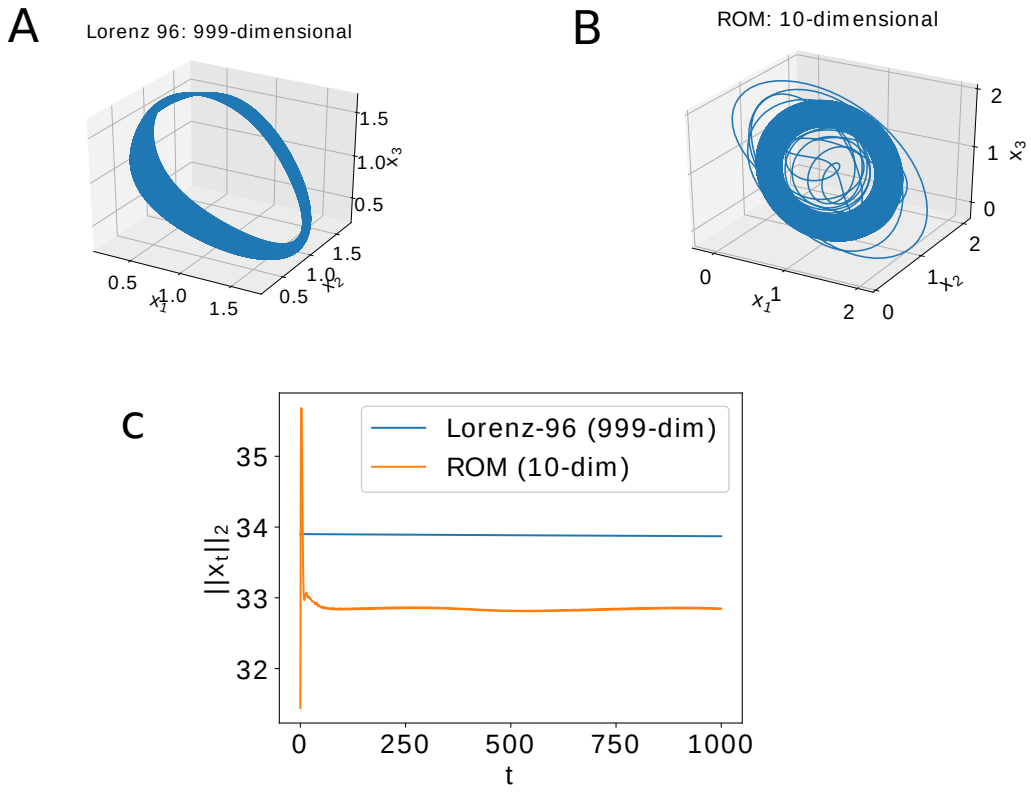

Figure 4.5: Demonstration of POD-based model order reduction applied to the Lorenz-96 model. The parameter $F = 1.2$ for these simulations. In ***A****, we see the stable periodic solution corresponding to the Lorenz-96 system with $N = 999$. In* ***B****, we plot the projection of the 10-dimensional ROM on the original coordinates. Thus we see a reconstruction of the periodic solution via a reduced order model. In* ***C****, we plot the 2-norm of the state variables at each time step.*

4.7 Conclusions and recommendations

A highly simplified one-state model describing ship velocity has been developed. This model still captures the major features, being fast rotation of the ship propeller and surrounding water on the one hand, versus slow changes of ship speed on the other. Approximate analytic solutions of this model were verified numerically and show that 'the method of averaging' may be applied. When applied to computational fluid dynamics this method recommends to not update all model states at the same rate. Instead, fast states associated with the propeller and surrounding water should be updated at a high rate, whereas updating the other states should be performed at a much slower rate. In this manner computational effort is seriously reduced since the fast states make up approximately 30% of all the states according to MARIN. Alternatively a 'time av-

eraged propeller model' might be developed, that is updated at the same slow rate as the remaining part of the model. Model reduction of the full-state model by means of proper orthogonal decomposition (POD) was also investigated and applied to the Lorenz 96 model that is quite suitable for presentation and implementation. POD is expected to work well since it essentially reduces the model by also averaging fast states appearing in the model. Successful application of this method does require an empirical interpolation method to approximate non-linear terms in the model. As to POD we therefore recommend to further investigate this promising approach.

Bibliography

[1] Kramer and Willcox, Nonlinear Model Order Reduction via Lifting Transformation and Proper Orthogonal Decomposition, arXiv:1808.0208v2[cs.NA], 2019

[2] Brenner, P., Cohen, A., Ohlberger, M., and Willcox, K., Model reduction and approximation – Theory and algorithms, SIAM Computational Science and Engineering, 2017

[3] Lorenz, E.N., Predictability – A problem partly solved, Editors: Palmer, T., Hagedorn, R., Cambridge University Press, 2006

[4] Verhulst, F., Methods and applications of singular perturbations: Boundary layers and multiple timescale dynamics, Springer Science & Business Media, 2005

[5] Cazemier, W., Verstappen, R.W.C.P., Veldman, A., Proper orthogonal decomposition and low-dimensional models for driven cavity flows, Physics of Fluids, 1998

[6] Lumley, J.L., The structure of inhomogeneous turbulence. Atmospheric turbulence and wave propagation. Editors: Yaglom, A.M., Tatarski, V.I., 1967

Appendix

4.A Matlab code to simulate highly simplified Marin ship model

```
% simship: simulations highly simplified MARIN ship model
%
% Programmer: GvW @ SWI2019-MARIN
clear; close all; clc; tb=[];
% Generate table from ship simulations with different p2 values
for p2=0.1:0.1:1
  T=0.1; p=zeros(6,1); p=[1;p2;0.5;1/T;1;0.1];
  if isinf(T); T=0.1; end
  ta=0:0.1*T:100*T; % Time axis
  x0=[0.15*p(1)/p(3);0.15*p(1)/p(3)]; % Initial state
  % Numerical integration
  [t,x]=ode23s(@(t,x)dnship(t,x,p),ta,x0);
  lx=size(x,1); lxm=round(lx/10); % 10% of response
  % Terminal & limiting values
  mblim=0.5*(max(x(end-lxm:end,1))-min(x(end-lxm:end,1)));
  malim=sum(x(end-lxm:end,1))/lxm;
  alim=(p(3)-p(5)+sqrt((p(3)-p(5))^2+4*p(2)*p(6)))/(2*p(6));
  blim=p(1)*(p(2)+p(3)*alim)*T/(2*pi);
  dmblim=mblim-blim;
  % Plot two velocity responses
  figure(1); plot(t,x(:,1),'-',t,x(:,2),'.');
  hold on;
  % Plot model terminal value a(tf)
  line([ta(1),ta(end)],[alim,alim]);
   %Plot model max(v)
  line([ta(1),ta(end)],[x(end,1)+blim,x(end,1)+blim]);
   %Plot model min(v)
  line([ta(1),ta(end)],[x(end,1)-blim,x(end,1)-blim]);
  hold off;
  % Display alim,model alim,blim,model blim
  disp('     alim        model alim      blim        model mblim')
  disp([alim,malim,blim,mblim])
  % Extend table
  tb=[tb; p2,alim,malim,100*(malim-alim)/alim,blim,mblim,...
  100*(mblim-blim)/blim];
  pause
end
tb % Show table

```

```
function [f]=dnship(t,x,p)
% State-space representation highly simplified MARIN ship model
% [f]=dnship(t,x,p)
%
% t,x,p : time, state & parameter vector
% f     : state derivatives
%
% Programmer: GvW @ SWI2019-MARIN
Fp1=(1+p(1)*cos(2*pi*p(4)*t))*(p(2)+p(3)*x(2)); % Propeller force
Fp2=(p(2)+p(3)*x(2)); % Average propeller force
Fw=p(5)*x(2)+p(6)*x(2)*x(2); % Water force
f=[Fp1-Fw; Fp2-Fw]; % State & averaged state derivative
```