Model calibration for ship simulations

Ed van Daalen[1], Joseph Fehribach[2], Tristan van Leeuwen[3], Christian Reinhardt[4], Nick Schenkels[5] and Ray Sheombarsing[4]

[1] MARIN, Wageningen
[2] Worcester Polytechnic Institute, USA
[3] Centrum Wiskunde & Informatica, Amsterdam
[4] Vrije Universiteit, Amsterdam
[5] Universiteit Antwerpen, Belgium

Abstract

Model calibration is an important aspect in ship simulation. Here, ship motion is described by an ODE which includes tuning parameters that capture complex physical processes such as friction of the hull. In order for the simulations to be realistic for a wide range of scenarios these tuning parameters need to be calibrated to scale experiments. In principle, the optimal tuning parameters can be computed for any given scenario, but this would require a corresponding scale experiment to be conducted. The aim is to minimize the number of scenarios that need to be pre-calibrated while still being able to realistically model ship motion for a wide range of scenarios. In this paper we investigate the use of polynomial (sparse grid) interpolation to compute the optimal tuning parameters for *any* scenario from a few pre-calibrated optimal values.

Perturbation analysis of a simple model for roll damping indicates that the optimal tuning parameter may indeed vary strongly with the chosen scenario. Numerical experiments with this model confirm that the optimal tuning parameters vary strongly (but smoothly!) with the scenario and can be well approximated with polynomial interpolants. Further numerical experiments with a more complex modelling code for ship maneuvring are very promising.

KEYWORDS: Model calibration, Parameter estimation, Chebyshev Interpolation, Sparse Grid Interpolation

1 Introduction

Ship simulators, used to train pilots, are based on simplified models for ship motion in order to enable real-time integration of the system. In such simplified models a lot of underlying physics is not explicitly modeled but is parametrized using tuning parameters. In order for the simulator to behave realistically, these models need to be calibrated to real-life (scale) experiments of actual ship motion under a wide variety of scenario's. This calibration process is depicted in figure 1. Here, and throughout the paper, we use the

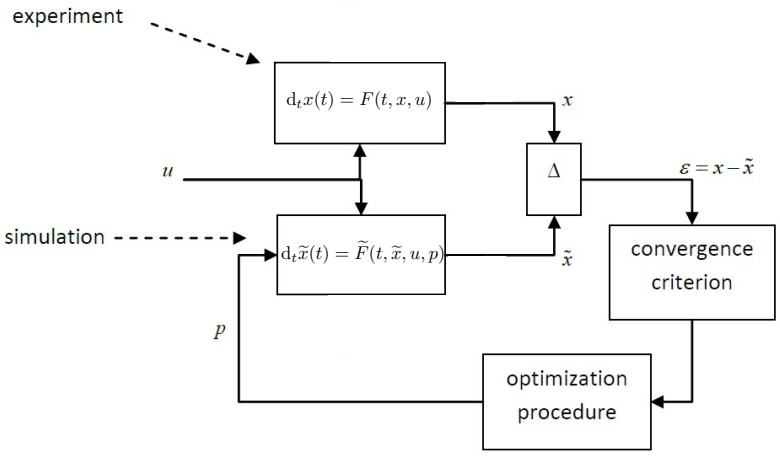


Figure 1: Schematic depiction of the calibration process.

following notation

- $\mathbf{u} = [u_1, u_2, \ldots, u_N]$ - vector with input scenario (rudder angle, propellor rpm, wave-height etc.);
- $\mathbf{p} = [p_1, p_2, \ldots, p_M]$ - vector with tuning parameters;
- $\mathbf{x} = [x_1, x_2, \ldots, x_K]$ - state vector describing *actual* ship movement – in the remainder of the paper we will treat this as the solution of an underlying *complex* model $\mathrm{d}_t \mathbf{x} = F(t, \mathbf{x}, \mathbf{u})$;
- $\widetilde{\mathbf{x}} = [\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_K]$ - state vector describing *modeled* ship movement – this is the solution of the *simplified* model $\mathrm{d}_t \widetilde{\mathbf{x}} = \widetilde{F}(t, \widetilde{\mathbf{x}}, \mathbf{u}, \mathbf{p})$;

Unfortunately, there is no single setting of the tuning parameters for which the simple model will fit the complex model for all possible scenarios. For a given scenario, however, we can find the corresponding optimal calibration parameters as follows. First, we define a cost function $\mathcal{C}(\mathbf{u}, \mathbf{p})$ that measures the mismatch between $\mathbf{x}(t; \mathbf{u})$ and $\widetilde{\mathbf{x}}(t; \mathbf{u}, \mathbf{p})$ for a given $\mathbf{u}$ and $\mathbf{p}$. An example of such a cost function is the least-squares mismatch between the horizontal spatial coordinates (x_1, x_2)

$$\mathcal{C}(\mathbf{u}, \mathbf{p}) = \sum_{i=1}^{2} \int \mathrm{d}t \, \left(x_i(t; \mathbf{u}) - \widetilde{x}_i(t; \mathbf{u}, \mathbf{p}) \right)^2 .$$

Then, the optimal $\mathbf{p}$ for a given scenario $\mathbf{u}$ is given by

$$\mathbf{p}^*(\mathbf{u}) = \underset{\mathbf{p}}{\operatorname{argmin}} \, \mathcal{C}(\mathbf{u}, \mathbf{p}).$$

Given that the parameter space $\mathbf{p}$ is relatively small ($M \approx 20$), we can perform such a single calibration with a simple direct-search method (Kolda et al., 2003). Note, however, that each calibration requires not only multiple evaluations (by time integration) of the simple model, but also *one evaluation of the complex model (i.e. an experiment)*. Therefore, we would like to minimize the number of scenarios for which this calibration is performed. The question is: How can we efficiently calibrate the simple model for a range of scenarios while using only a limited number of evaluations of the complex model (i.e., experiments).

1.1 Approach

The main idea of our approach is to calibrate the simple model for a number of (cleverly chosen) scenarios $\{\mathbf{u}_k\}_{k=1}^{L}$, yielding optimal calibration parameters $\{\mathbf{p}_k^*\}_{k=1}^{L}$. For *any* given scenario $\mathbf{u}$, we then interpolate the optimal $\mathbf{p}^*$ elementwise based on these values

$$p_i^*(\mathbf{u}) = \sum_{k=1}^{L} w_{k,i} \psi_k(\mathbf{u}),$$

where ψ_k are basis functions specific to the type of interpolation used and $w_{k,i}$ are the corresponding weights chosen such that $p_i^*(\mathbf{u}_k) = p_{k,i}^*$. The main assumption here is that $\mathbf{p}^*$ varies smoothly with $\mathbf{u}$.

For a 1D scenario space (i.e., $N = 1$) we use Chebyshev interpolation in order to get high accuracy with only a few samples. In this case, the nodes (Chebyshev points) on $[-1, 1]$ are given by

$$u_k = \cos\left(\frac{2k - 1}{2L} \pi \right).$$

For higher dimensional scenario spaces ($N > 1$), a simple cartesian product approach is not very attractive since the number of samples would grow exponentially with the dimension of the scenario space. In order to avoid this so-called *curse of dimensionality* we will consider *sparse grid* interpolation for $N > 1$ (Barthelmann et al., 2000). In sparse grids, the sampling points are clustered near the boundary of the domain and chosen more sparsely in the interior. Hereby the number of sampling points is considerably reduced when compared to a regular sampling. There are different choices of sparse grids that vary in number of grid points involved. A popular choice for the approximation of smooth functions is the so-called Clenshaw-Curtis grid. An example of such a grid at consecutive stages of refinement is shown in Figure 2. Table 1 shows how the number of sampling points grows with the stage of refinement. Note that L approximately doubles for each stage, whereas we would expect a quadrupling for a regular sampling in 2D. For more details on the accuracy and efficiency of sparse grid interpolation we refer to Barthelmann et al. (2000).

Stage	1	2	3	4	5	6	7
L	5	13	29	65	145	321	705

Table 1: Number of points in the sparse 2D grid in dependence of the stage depth. Note that L approximately doubles for each stage, whereas we would expect a quadrupling for a regular sampling for $N = 2$.

1.2 Outline

The remainder of the paper is organized as follows. First, we consider a model for *roll damping*. In this case, the models predict the roll motion (i.e. oscillations around the longitudinal axis) of the ship for given initial angle and forcing terms (which serve as the scenario parameters). The complex model F contains a non-linear damping term, while the simple model $\widetilde{F}$ contains only an equivalent linear damping term (which serves as the tuning parameter). For this model problem we perform a perturbation analysis and present a closed form solution for the optimal tuning parameter. We also perform a range of numerical experiments with both a 1D (with Chebyshev interpolation) and a 2D (with sparse grid interpolation) scenario space. Next, we present numerical experiments using a 6 degree-of-freedom model for rigid ship motion using the FREDYN modeling code. Here, the complex model is based on a frigate while the simple model is based on a lifeboat. We use a 1D scenario space (rudder angle) and we incorporate 6 tuning parameters governing rotational and drift forces. We compare two different mismatch criteria; based on horizontal spatial coordinates and based on the turning circle radius.

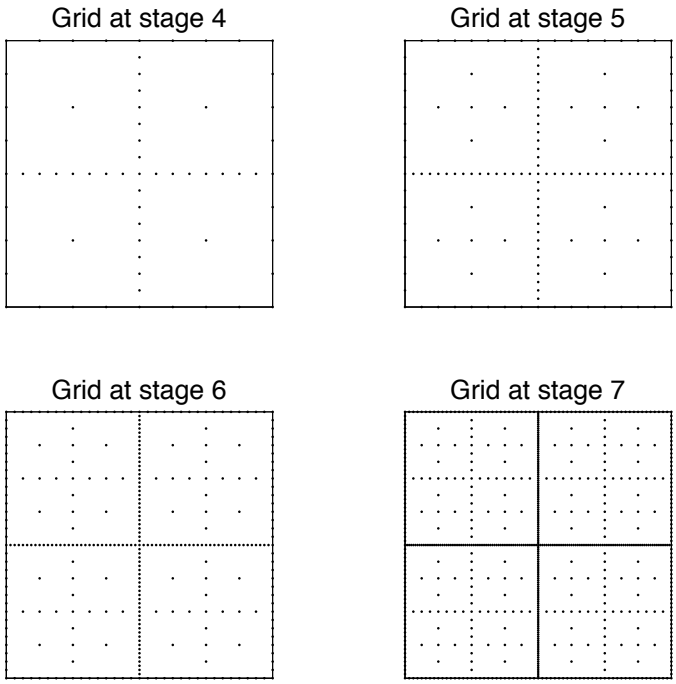


Figure 2: Grids used in various stages of Sparse-Grid interpolation procedure for $N = 2$. Note the clustering of the points at the boundary and sparsity in the interior of the domain.

Finally, we present conclusions and recommendations for further research.

2 A toy calibration problem: equivalent linear damping

The roll motion of a ship is modelled by the following ODE

$$(I + A)\ddot{\phi}(t) + B(\dot{\phi})\dot{\phi}(t) + C\phi(t) = M(t), \tag{1}$$

where I, A and C are constants, $B(\cdot) = b_1 + b_2|\cdot|$ is a damping term and $M(t)$ is a forcing term. In the remainder of the section we will consider this equation with *non-linear damping* (i.e., $b_2 \neq 0$) as the complex model, while the simple model only includes *linear* damping ($b_2 = 0$) and b_1 will serve as the tuning parameter. The initial angle $\dot{\phi}(0)$ and the amplitude and frequency of a periodic damping term $M(t) = a\sin(\omega t)$ will serve as the scenario parameters.

2.1 Linear damping

For $b_2 = 0$, equation (1) can be written in the form of a standard damped oscillator

$$\ddot{\phi} + 2\zeta\omega_o\dot{\phi} + \omega_o^2\phi = m \tag{2}$$

where

$$\omega_o := \sqrt{\frac{C}{I+A}}, \qquad m := \frac{M}{I+A} \qquad \text{and} \qquad \zeta := \frac{b_1}{2\sqrt{C(I+A)}}.$$

In the above equation, ω_o is the undamped oscillation frequency and ζ is the nondimensional damping coefficient. If $0 < \zeta < 1$, the system is sub-critically damped, and the general solution is

$$\phi(t) = \alpha e^{-\zeta\omega_o t}\cos(\omega_d t - \beta) + \phi_{\mathrm{p}}(t)$$

where α and β are free parameters determined by the initial conditions (α is the amplitude and β is the phase angle). The damped frequency is $\omega_d := \omega_o\sqrt{1-\zeta^2}$, and $\phi_{\mathrm{p}}(t)$ is any particular solution which satisfies the nondimensional equation (2). For example, if $M(t) = M$ is constant, then $\phi_{\mathrm{p}}(t) = m/\omega_o^2 = M/C$ is the simplest particular solution.

2.2 Perturbation analysis

Now consider the homogeneous perturbed nondimensional equation

$$\ddot{\phi} + 2\zeta\omega_o(1 + \epsilon|\dot{\phi}|)\dot{\phi} + \omega_o^2\phi = 0. \tag{3}$$

For convenience, we drop the absolute values and allow the sign of ϵ to change when the sign of $\dot{\phi}$ changes [1]. Assuming that $|\epsilon| << 1$, one can make a regular perturbation expansion using the ansatz $\phi(t) = \phi_0(t) + \epsilon\phi_1(t) + O(\epsilon^2)$. Substituting this ansatz into (3), one finds that as before $\phi_0(t) = \alpha e^{-\zeta\omega_o t}\cos(\omega_d t - \beta_0)$ and that ϕ_1 must satisfy

$$\ddot{\phi_1} + 2\zeta\omega_o\dot{\phi_1} + \omega_o^2\phi_1 = -2\zeta\omega_o(\dot{\phi_0})^2 \tag{4}$$

The form of the homogeneous solution for (4) is the same as before, but unless the initial position or velocity depend on ϵ (which would be unusual), this homogeneous solution is identically zero. Because of the form of the right-hand side of (4), the particular solution must be the linear combination of three terms:

$$\phi_1^{\mathrm{p}}(t) = A^{\mathrm{p}}e^{-2\zeta\omega_o t}\cos^2(\omega_d t - \beta_1) +$$
$$B^{\mathrm{p}}e^{-2\zeta\omega_o t}\cos(\omega_d t - \beta_1)\sin(\omega_d t - \beta_1) + C^{\mathrm{p}}e^{-2\zeta\omega_o t}\sin^2(\omega_d t - \beta_1)$$

where the coefficients A^{p}, B^{p} and C^{p} are determined by substituting this linear combination into (4), and β_1 is a shifted phase angle due to the presence of $\dot{\phi_0}$ in the right-hand side of (4), rather than just having ϕ_0.

2.3 Calibration

Now let us take the perturbed solution (for some ζ and ϵ) as the solution of the *complex system* and let us try to match this to the unperturbed solution using ζ as a tuning parameter. Specifically, let $\phi(t;\epsilon_1,\zeta_1) := \phi_0(t;\zeta_1) + \epsilon_1\phi_1^{\mathrm{p}}(t;\zeta_1)$ be the solution of the complex system and let $\widetilde{\phi}(t;p) = \phi_0(t;p)$ with p being the single tuning parameter. The question then is how should one set p so that the simple solution matches the complex solution?

Consider the absolute difference

$$\begin{aligned} |\phi(t;\epsilon_1,\zeta_1) - \widetilde{\phi}(t;p)| &= |\alpha(e^{-\zeta_1\omega_o t} - e^{-p\omega_o t})\cos(\omega_{\mathrm{d}} t - \beta_0) + \epsilon_1\phi_1^{\mathrm{p}}(t,\zeta_1)| \\ &= \alpha e^{-\zeta_1\omega_o t}|(1 - e^{(\zeta_1 - p)\omega_o t}\cos(\omega_{\mathrm{d}} t - \beta_0) \\ &\quad + \epsilon_1 e^{-\zeta_1\omega_o t}(A^{\mathrm{p}}\cos^2 + B^{\mathrm{p}}\cos\sin + C^{\mathrm{p}}\sin^2)| \end{aligned}$$

Each of the last four trigonometric functions must be evaluated at $(\omega_{\mathrm{d}} t - \beta_1)$. Again the coefficients A^{p}, B^{p} and C^{p} are functions of ζ_1; they are determined by requiring that ϕ_1^{p} is actually a particular solution of (4).

Because of the decaying exponentials, the above absolute difference will decrease in time. But it is also possible to make this difference zero at a

[1] Because the sign of ϵ changes with each half oscillation, one must stop and restart the solutions to follow the motion through multiple oscillations.

specified time, for example, if $t = \beta_0/\omega_d$, then the absolute difference is zero when

$$p = \zeta_1 - \frac{\sqrt{1-\zeta_1^2}}{\beta_0} \ln(1 - \epsilon_1 e^{-\zeta_1\beta_0/\sqrt{1-\zeta_1^2}} D^{\mathrm{p}})$$

where $D^{\mathrm{p}} := A^{\mathrm{p}} \cos^2(\beta_0-\beta_1) + B^{\mathrm{p}} \cos(\beta_0-\beta_1)\sin(\beta_0-\beta_1) + C^{\mathrm{p}} \sin^2(\beta_0-\beta_1))$ So this is a relatively simple formula for determining the tuning parameter p in terms of ζ_1 and ϵ_1, the given parameters in the complex solution. That is, there is an explicit expression for selecting the tuning parameter in terms of the given parameters to minimize the absolute difference at least at one specific time. Of course, this approach only has the two solutions matching at one specified time, and then again for large time.

The tuning parameter of course could be chosen to minimize the absolute difference in other ways, for example, by selecting a different time, or by making a least squares fit across some interval of time. But since the absolute difference already decays in time, one would likely wish to set the difference to zero at some early time. So one could minimize

$$\int_0^T \Big(\phi(t;\epsilon_1,\zeta_1) - \widetilde{\phi}(t;p)\Big)^2 \, dt$$

by finding a stationary point p for which

$$\frac{\mathrm{d}}{\mathrm{d}p} \int_0^T \Big(\phi(t;\epsilon_1,\zeta_1) - \widetilde{\phi}(t;p)\Big)^2 \, dt = 0.$$

We expect that a closed-form expression for the optimal p can be derived in a simular manner as above but this investigation is outside the scope of the current report.

2.4 Numerical experiments

For the numerical experiments we numerically integrate the roll damping equation (1) with $I = 6.4$, $A = 0$, $C = 1$, $M(t) = a\sin(\omega t)$ and initial condition $\phi(0) = \phi_0$ and $\dot{\phi}(0) = 0$ for $T = 80$ seconds. We will use $\mathbf{u} = [\phi_0, a, \omega]$ as scenario parameters. We denote the solution of the *complex* system (with $b_1 = 0$ and $b_2 = 15$) by $\phi(t;\phi_0,a,\omega)$ while $\widetilde{\phi}(t;\phi_0,a,\omega;p)$ denotes the solution of the simple system (with $b_1 = p$ and $b_2 = 0$).

In these experiments we find the optimal p by minimizing the least-squares cost function

$$\begin{aligned} \mathcal{C}(p,\phi_0,a,\omega) &= \sum_i \Big(\phi(t_i;\phi_0,a,\omega) - \widetilde{\phi}(t_i;\phi_0,a,\omega;p)\Big)^2 \\ &\quad + \Big(\dot{\phi}(t_i;\phi_0,a,\omega) - \dot{\widetilde{\phi}}(t_i;\phi_0,a,\omega;p)\Big)^2 \end{aligned}$$

using Matlabs `fminsearch`. For the 1D interpolation, we use the `chebfun` package (Trefethen, 2013). For 2D interpolation we use the `Sparse Grid Interpolation Toolbox` package (Klimke, 2007).

2.4.1 Case 1: roll decay with varying initial roll angle

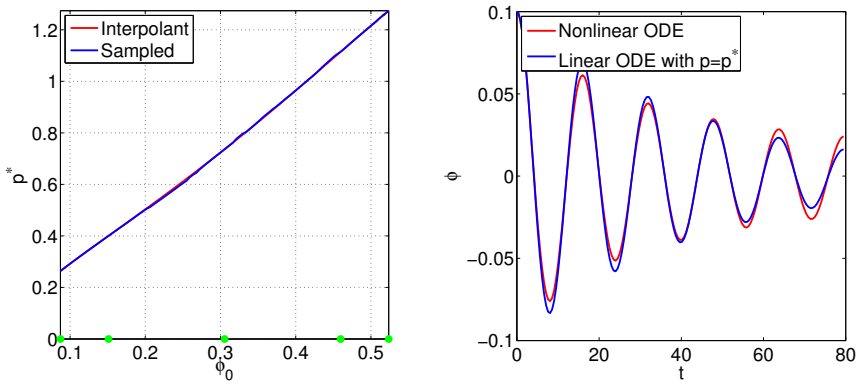
We set $a = 0, \omega = 0$ and vary only the initial condition $\phi_0 \in [\pi/36, \pi/6]$. The optimal values of p as a function of ϕ_0, obtained through Chebyshev interpolation with 5 points and brute-force sampling is shown in Figure 3 (a). The solutions for the complex and simple system (using the optimal p) for $\phi_0 = 0.1$ is shown in Figure 3 (b).

2.4.2 Case 2: regular forcing with varying amplitude

In this experiment, we set $\phi_0 = 0, \omega = 0.395$ and vary only the amplitude $a \in [0, 2]$. The optimal values of p as a function of a, obtained through Chebyshev interpolation with 5 points and brute-force sampling is shown in Figure 4 (a). The solutions for the complex and simple system (using the optimal p) for $a = 1.3$ is shown in Figure 4 (b).

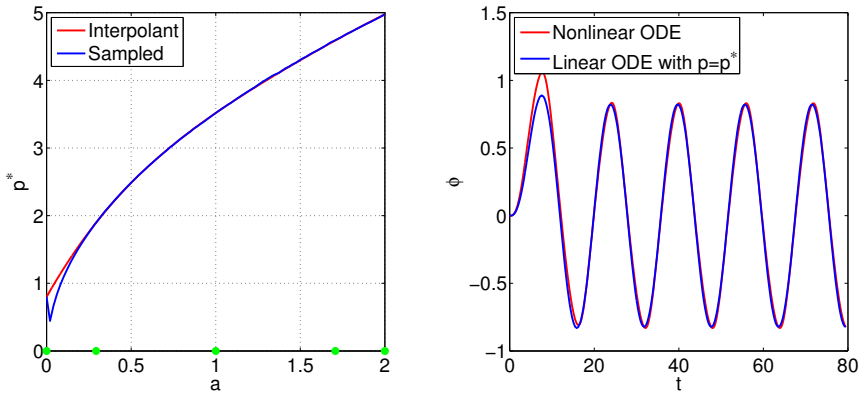
2.4.3 Case 3: regular forcing with varying amplitude and frequency

In this experiment we set $\phi_0 = 0$ and vary both $a \in [0, 2]$ and $\omega \in [0, 2]$. Figure 5 (a) shows a sparse interpolant on the Clenshaw-Curtis grid of stage 5 of the optimal p. The solutions for the complex and simple system (using the optimal p) for a particular choice of (a, ω) is shown in Figure 5 (b).



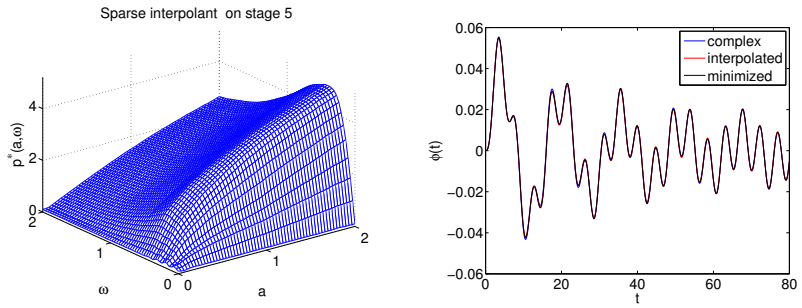
(a) (b)

Figure 3: Case 1: (a) p^* as a function of ϕ_0 obtained through Chebyshev interpolation with 5 points (blue) and brute-force sampling with 100 points (red). (b) Solutions of the complex (red) and simple systems (blue) for the optimal p obtained through interpolation, both for $\phi_0 = 0.1$.



(a) (b)

Figure 4: Case 2: (a) p^* as a function of a obtained through Chebyshev interpolation with 5 points (blue) and brute-force sampling with 100 points (red). (b) Solutions of the complex (red) and simple systems (blue) for the optimal p obtained through interpolation, both for $a = 1.3$.



(a) (b)

Figure 5: Case 3: (a) p^* as a function of (a, ω) obtained through sparse grid interpolation with 145 samples. (b) Solutions of the complex (blue) and simple systems for the optimal p obtained through interpolation (red) and by direct optimization (black), all for a randomly chosen (a, ω).

3 FREDYN code

In this section we present experiments with the `FREDYN` program, (Ypma, 2014). The complex model is based on a frigate and replaces the real life experiment. The simple model is based on a lifeboat. In these experiments $\mathbf{u}$ represents the rudder angle (i.e. $\mathbf{u}$ is 1-dimensional) which lies within the range $[5, 30]$ deg. As tuning parameters we use the following parameters that govern the drift and rotational forces:

$$\mathbf{p} = [X_{vv}, X_{rr}, X_{vr}, Y_{uv}, Y_{vv}, Y_{ur}].$$

The default values of these parameters are given by

$$\mathbf{p} = [18508, 0, 4117877, -82292, -201134, 3075682]. \tag{5}$$

To compare the simulations, we plot both the horizontal coordinates $(x_1(t), x_2(t))$ and the turning radius which is defined as

$$R(t) = \frac{\sqrt{v_1(t)^2 + v_2(t)^2}}{v_r(t)},$$

where v_1, v_2 are the horizontal velocities (surge, sway) in m/s and v_r is the angular velocity (yaw) in rad/s. We stop the simulations after the ship has completed a full turn. This means that simulations with a smaller rudder angle will run longer.

Figure 6 shows the behaviour of the complex and simple model for a rudder angle of 5 deg using the default $\mathbf{p}$. This clearly illustrates the need to fit the tuning parameters.

3.1 Finding an optimal p

We consider two different cost functions. The first cost function measures the misfit between the horizontal coordinates (x_1, x_2):

$$\mathcal{C}_1(\mathbf{u}, \mathbf{p}) = \int \mathrm{d}t \ (x_1(t) - \widetilde{x}_1(t))^2 + (x_2(t) - \widetilde{x}_2(t))^2 \,. \qquad \text{(Method 1)}$$

The second cost function measures the misfit between the turning radii (cf. equation (3)),

$$\mathcal{C}_2(\mathbf{u}, \mathbf{p}) = \int \mathrm{d}t \ \left(R(t) - \widetilde{R}(t)\right)^2 . \qquad \text{(Method 2)}$$

We use a direct-search method (Matlab's `fminsearch`) to find the optimal $\mathbf{p}$.

Figure 7 shows the simulations for the optimal $\mathbf{p}$ as obtained via Method 1 and Figure 8 shows the simulations for the optimal $\mathbf{p}$ as obtained via Method

2, both for a rudder angle of 5 deg. Comparing these to Figure 6 we see a dramatic improvement in the fit.

The 6^{th} component of the optimal $\mathbf{p}$ (Y_{ur}) as a function of the rudder angle using 5 and 10 Chebyshev points is shown in Figure 9. We observe that the optimal $\mathbf{p}$ does not vary as smoothly with $\mathbf{u}$ as in the case of roll damping. In particular, we see a *staircase* effect that we do not fully understand. Still, the Chebyshev interpolation is able to capture the general trend. Figures 10 and 11 show how the interpolated optimal $\mathbf{p}$ for a rudder angle of 15 deg is able to produce a very good match between the simple and complex models.

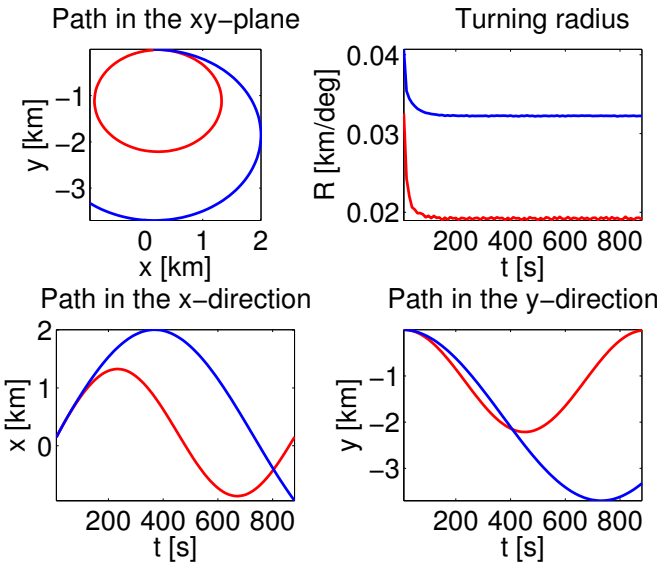


Figure 6: Simulation for the simple and complex model with a rudder angle of 5 deg using the default values for $\mathbf{p}$. The red and blue lines represents the complex and simple model respectively.

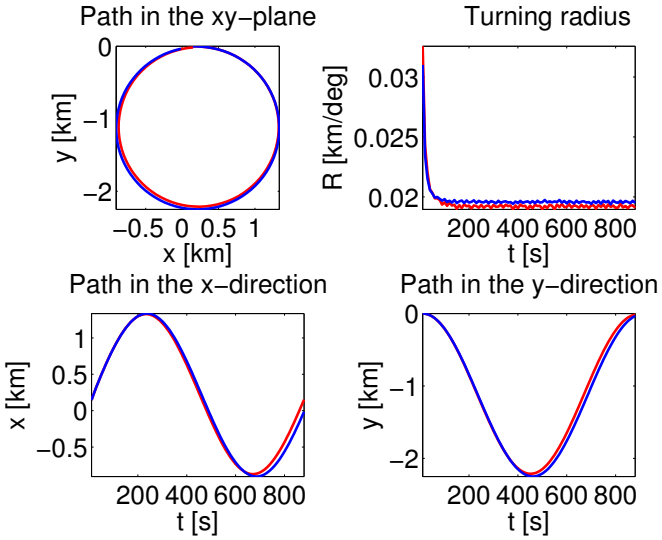


Figure 7: Simulation for the simple (blue) and complex (red) model with a rudder angle of 5 deg and the optimal $\mathbf{p}$ found by Method 1.

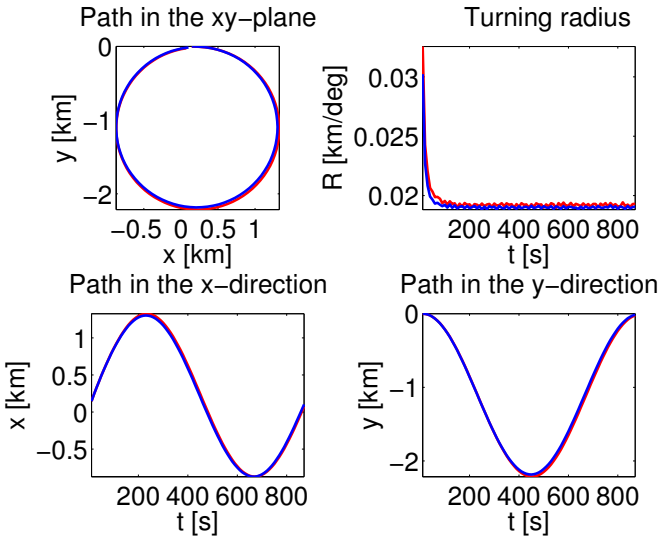


Figure 8: Simulation for the simple (blue) and complex (red) model with a rudder angle of 5 degrees and the optimal $\mathbf{p}$ found by Method 2.

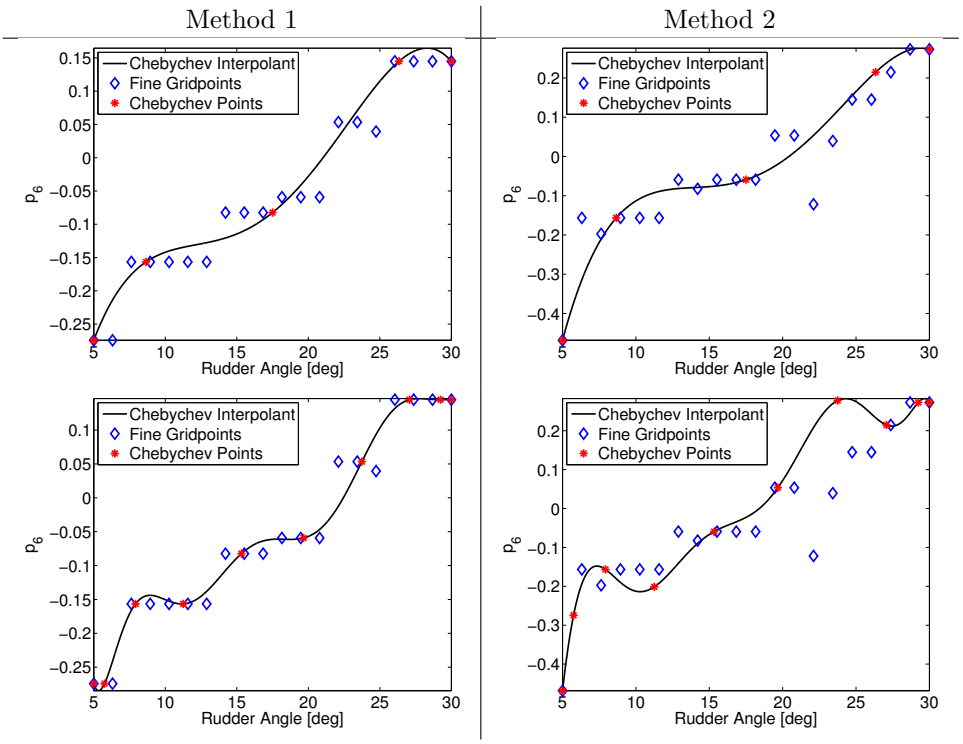


Figure 9: Chebyshev interpolation of the 6th component of $\mathbf{p}^*$ for the two methods. The first and second row use 5th and 10th order Chebyshev interpolation respectively. The values of p_6^* are scaled with the default value p_6 (see (5)).

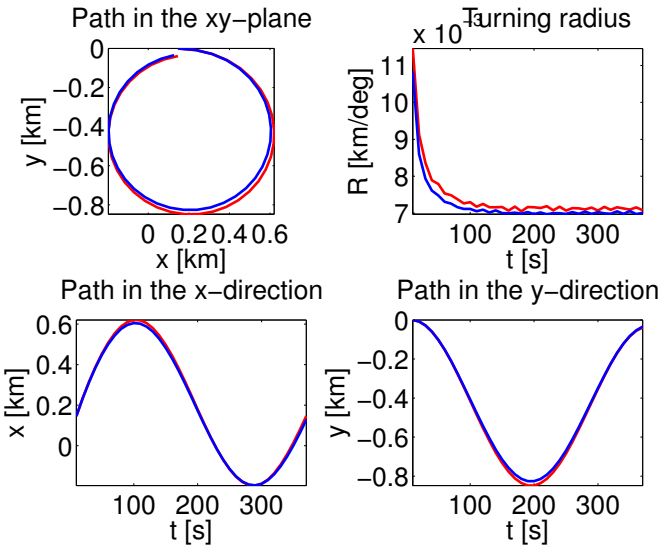


Figure 10: Simulation for the simple and complex model with a rudder angle of 15 degrees. The red and blue lines represents the complex and simple model respectively. Here the first method was used to determine the optimal value for p with Chebyshev interpolation of order 5.

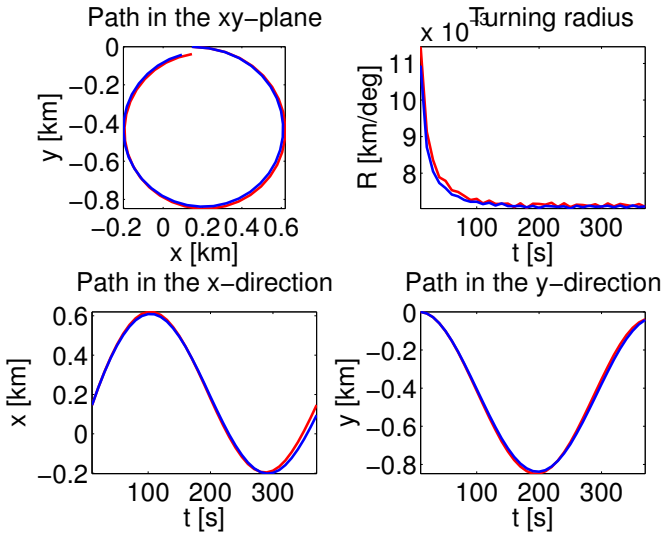


Figure 11: Simulation for the simple and complex model with a rudder angle of 15 degrees. The red and blue lines represents the complex and simple model respectively. Here the second method was used to determine the optimal value for p with Chebyshev interpolation of order 5.

4 Conclusions

We have presented a method for model calibration with application to ship simulations. The goal is to find the optimal tuning parameters $\mathbf{p}$ such that the solution of a *simple* system, $\widetilde{\mathbf{x}}(t; \mathbf{u}, \mathbf{p})$, matches the solution of a more complex system, $\mathbf{x}(t; \mathbf{u})$, for a range of scenarios $\mathbf{u}$. For a single scenario, this calibration can by done by minimizing a cost function that measures the difference between $\widetilde{\mathbf{x}}$ and $\mathbf{x}$. There are only a few (≈ 20) tuning parameters so that this minimization can be done with so-called direct-search methods. Such methods are very suitable for black-box optimization problems since they do not require gradient calculations of the cost function w.r.t. the tuning parameters.

However, each calibration requires an evaluation of the complex system (i.e., a scale experiment). In order to minimize the number of scale experiments that need to be done, we calibrate the simple model only for a small number of well-chosen scenarios $\{\mathbf{u}_k\}$, giving us the corresponding tuning parameters $\{\mathbf{p}_k\}$. We assume that the optimal tuning parameters vary smoothly with $\mathbf{u}$ and use polynomial interpolation to compute the optimal $\mathbf{p}$ for any given $\mathbf{u}$ from these points. When there is only one scenario parameter we use Chebyshev interpolation. This approach does not generalize well to higher dimensional scenario space as the required samples would grow exponentially with the dimension. To avoid this *curse of dimensionality* we resort to sparse-grid interpolation techniques.

Perturbation analysis of a model-problem (roll-damping) indicates that it is possible to obtain closed-form solutions for the optimal tuning parameter (the equivalent linear damping) in some specific cases. Numerical experiments indicate that the optimal $\mathbf{p}$ varies smoothly with $\mathbf{u}$. Both Chebyshev and sparse-grid interpolation perform well in this setting, as is confirmed by numerical experiments. Numerical experiments with a more complex system of ODEs that models full ship motion (using the FREDYN code) show promising results.

5 Recommendations

- The perturbation analysis and numerical experiments on the roll damping equation give some insight in how to choose the optimal equivalent linear damping term. It would be very insightful to verify the findings from the perturbation analysis numerically. The analysis might also be extended by considering other mismatch criteria and include a driving term. Such analysis should be able to predict the observed smooth dependency of the optimal $\mathbf{p}$ w.r.t. $\mathbf{u}$ and may even tell us how smooth the function is, allowing us to compute a-priori error estimates for the

interpolation.

- Further numerical testing with the FREDYN code with time-varying rudder setting, using the optimal tuning parameters appropriate for each rudder setting. This requires the ability to vary the tuning parameters with time.

- An alternative avenue – not tested in this report – is to try to find a $\mathbf{p}$ that is optimal over a range of scenarios $\mathbf{u}$

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \int \mathrm{d}\mathbf{u}\, \mathcal{C}(\mathbf{u}, \mathbf{p}) \pi(\mathbf{u}),$$

  where $\pi(\mathbf{u})$ is a weighting function used to emphasize scenarios that are deemed more important [2]. A first step would be to use a brute force approach to compute the integral (i.e., by dense sampling of the scenario space). If the results are satisfactory, a generalized Polynomial Chaos expansion (gPC) can be employed to efficiently estimate the expectation (Xiu, 2010). The basic ideas are very similar to the ones discussed above; we need to sample a number of scenarios according to a quadrature rule (which will depend on π) and sparse grid techniques can be used to generalize to higher dimensions. Note that we cannot employ Monte-Carlo sampling methods to estimate the integral, as this would require evaluation of the misfit for many scenarios, which in turn would mean doing many experiments to evaluate the complex model for those scenarios.

- The optimization problem for a single scenario is very likely to have multiple minima (local *and* global) and suffer from ill-conditioning. This means that the variability of the optimal $\mathbf{p}$ as a function of $\mathbf{u}$ we observed may to some extent be artificial, in particular for the FREDYN examples. A sensitivity analysis of the problem (for example through the Jacobian of $\mathbf{p}^*$) may give some insight.

[2] Alternatively, we can interpret $\pi(\mathbf{u})$ as a probability, in which case we are aiming to find a $\mathbf{p}^*$ that minimizes the *expected* misfit over all possible scenarios.

A Matlab code

We give a short overview of the scripts and functions related to the generation of the above described results. We restrict to describe the dependencies and short explanations of the functionalities. Additional comments can be found in the files themselves.

A.1 1D Interpolation for damped oscillator

- `compareP`: computes two numerical approximations of $\theta_0 \mapsto p^*(\theta_0)$, where θ_0 is the parameter to be varied in an interval specified by the user. One approximation is determined by brute-force sampling and the other by Chebyshev interpolation. Moreover, $\theta_0 = \phi_0$ in the absence of an external force, and $\theta_0 = a$ in the presence of an external force. This subroutine calls `interpolateP` and `optimizeP`.

- `interpolateP`: computes the Chebyshev interpolant of the function $\theta_0 \mapsto p^*(\theta_0)$ by using the Chebfun-package developed in [reference to homepage of Chebfun]. This subroutine calls `optimizeP`.

- `optimizeP`: computes an optimal value for the damping coefficient p, such that the error between the solutions of the nonlinear and linear model is minimized, by applying the Matlab-subroutine `fminsearch` to the function-handle `difference`.

- `difference`: computes the error in the Euclidian norm between the time series of the nonlinear and linear model for a given value of p. This subroutine calls `integrateF`.

- `integrateF`: numerically integrates the ODE

$$(I + A)\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \left(b_1 + b_2\left|\frac{\mathrm{d}\theta}{\mathrm{d}t}\right|\right)\frac{\mathrm{d}\theta}{\mathrm{d}t} + \theta = M(t) \tag{6}$$

  by using the Matlab-subroutine `ode45`. This subroutine calls `F`.

- `F`: implements the first-order vector field associated to (6).

A.2 2D Interpolation for damped oscillator

- `interpolant_script`: main script implementing above described comparison. Used to produce Figure 5 (Note the randomized input a and ω hence output will not be the same). Calls `optimal_p` in the construction of the sparse interpolant

- `optimal_p`: implements optimization procedure, calls `difference` in the optimization using `fminsearch`
- `difference`: implements the least-squares misfit, calls `compute_timeseries`.
- `compute_timeseries` performs the simulation of the complex and simple system by writing the as first order systems, calls `nonlinear_osc_rhs` within `ode45`
- `nonlinear_osc_rhs`: implements the right hand side of the first order systems. Notation used: $I\ddot{\theta} + \left(b_1 + b_2|\dot{\theta}|\right)\dot{\theta} + \theta = M(t)$

A.3 1D Interpolation with `FREDYN` code

The m-files should be places in a folder which must contain two copies of the `./examples/manoeuvring/leander` folder, one called *leander_simpel* and the other `leander_complex`[3]. This is used to run both the complex fregat model and the simpel lifeboat model at the same time while keeping the results seperated.

In both folders the `leander_ship.xmf` file should be changed. The `RandomRudder` script must be removed or commented out and be replaced by the line

```
scripting::Scripting "ConstantRudder" {};
```

Some remarks:

- In all the m-files `dir` is short for directory and normaly this will be the string `leander_simpel` or `leander_complex`.
- The m-file `ReadData` loads the content of the `leander.dat` file into Matlab. This is usually refered to as `OutputC` for the complex model and *OutputS* for the simpel model one.
- Some of the m-files use the argument `Mode`. This is either 1 or 2 and refers to the way the variable p is optimized. For more details we refer to section 3.

[3] e.g. we placed them in `./examples/manoeuvring/`

M-Files

These first few m-files are functions that are used to change and run the simulation of the complex and simple model in various ways, as well as to determine the optimal value for the parameter p for a certain rudderangle.

- `WriteTweaking:` Used to change the value of the variable p in the `mo_leander_hull.xmf` file in the directory denoted by `dir`, which is either `leander_simpel` or `leander_complex`. All 26 components can be changed, but in the other m-files only the first six in the `leander_simpel` folder have been modified.

- `WriteParameters:` In order to easily change the rudderangle from within Matlab this m-file creates a python script `ConstantRudder.py` in the directory pointed to by dir (again, either `leander_simpel` or `leander_complex`). This is done by adding a line to the `ConstantRudderTemplate.py` defining the rudderangle and copying this to the chosen directory.

- `RunSimulation:` Runs the simpulation of the simple (`dir` = `leander_simpel`) or complex (`dir` = `leander_complex`) model and deletes the created `*_leander.out` files.

- `ReadData:` Reads the `leander.dat` file in the directory denoted by `dir` and loads it into Matlab.

- `PlotTraces:` Makes a plot of the data in `OutputC` and `OutputS`. Top left: the path in the xy-plane of both models. Top right: the radial distance R of both models. Botom left: timetrace of x of both models. Botom right: timetrace of y for both models.

- `OptimalP:` Given a `Mode` and a rudderangle (argument u) this runs the complex model and then searches for an optimal value for p (only the first six components are changed) by using the Matlab function `fminsearch` (only 20 iterations are used). Using this optimal p the simple model is computed. The function returns the output of both simulations (`OutputC` and `OutputS`) and the optimal value for p `OptP` (corresponding to this rudderangle u!).

- `NormDiff:` This function is used to determine how closely the simple model matches the complex one, e.g. it is used by `OptimalP`

The following m-files where used to create a.o. the results shown in the presentation. The can be seen as an example how the previous m-files can be used.

- `Example0:` Runs the complex and the simple model with the default value of p for three different rudderangles ($5, 15$ and 30) to show that this is not always a good choice of parameters. The results are saved.
- `Example1:` Determines the optimal value of p for an oversampling of the range $[5, 30]$ which is used as a reference value for the Chebychev interpolation. The results are saved.
- `Example2_5:` Determines the 5 Chebychev points in $[5, 30]$ and calculates the optimal value of p for them. The results are saved.
- `Example2_10:` Idem for 10 Chebychev points.
- `MakePlots:` Using the data calculated by the `Example` m-files, this m-files makes plots of the results and calculate the Chebychev interpolation. The plots are all saved in a subfolder `./Plots`

References

Volker Barthelmann, Erich Novak, and Klaus Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12:273–288, 2000. URL `http://link.springer.com/article/10.1023/A:1018977404843`.

Andreas Klimke. Sparse Grid Interpolation Toolbox – user's guide. Technical Report IANS report 2007/017, University of Stuttgart, 2007.

Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Review*, 45(3):385–482, January 2003. ISSN 0036-1445. doi: 10.1137/S003614450242889. URL `http://epubs.siam.org/doi/abs/10.1137/S003614450242889`.

L. N. Trefethen. Chebfun version 4.3, 2013. `http://www.chebfun.org`.

Dongbin Xiu. *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010. ISBN 9781400835348.

E. Ypma. Generic forward-aft manoeuvring model. *MARIN Technical Report*, 2014.

, , , , , , ,