Modelling flow-induced vibrations of gates in hydraulic structures

Erdbrink, C.D.

Publication date
2014

8 Evolutionary computing for system identification[7]

8.1 Introduction

Classes of computational methods for hydraulic structure dynamics can be categorised in a similar way as for general structural analysis, namely by distinguishing direct and indirect methods in the time and frequency domain. The forward approach of Chapter 6 is a direct method in the time domain and has the advantage of capturing potential non-linearities, but the disadvantage that only a single configuration (defined by mass and stiffness of the mass-spring and by the hydraulic boundary conditions) can be considered per simulation. This is highly inconvenient if the added coefficients and response frequency have a mutual dependence – which was rightly assumed not to be the case in Chapter 6 because of a negligible free-surface influence. Frequency domain methods have been mentioned briefly in Sections 2.3 and 7.5.2. The straightforward extension to more degrees of freedom is an advantage of modal analyses. This involves the assumption of equal damping distributions, however, and non-linearities are not apparent from frequency spectra. An approach not yet mentioned is the impulse response method, which derives the response to an arbitrary time-varying force from the response to a unit impulse using Duhamel's convolution integral (e.g. Maymon, 1998). This computation includes the whole frequency range, but the obvious catch is that the unit impulse response has to be known first.

So established computational methods for gate vibrations meet with three obstacles:

- The assumption of low positive damping, this is convenient computationally, but irrealistic in many scenarios.
- The possibility of mutual dependency between added coefficients and response frequency.
- The existence of non-linear behaviour, caused for instance by the stiffness force varying non-linearly with displacement (e.g. when the suspension consists of a hydraulic cylinder) or the combined time-dependent working of negative hydraulic damping and increased structural damping at high amplitudes.

A fourth practical complication is that often only the gate response can be measured, as was the case in the experiment of Chapter 5, and not the flow forces on the gate. This chapter therefore looks into a new way of model building with the gate motion represented as a dynamical system.

System identification studies embarked on the task of inferring ODE models a few decades ago (Åström and Eykhoff, 1971). However, the fixed structure acting as a vehicle for the parameter optimization was usually not an ODE itself, but rather an easy-to-compute basis function like a polynomial. Non-linear system identification techniques for structural

[7] This chapter is partly based on texts and content from the paper "Identifying self-excited vibrations with evolutionary computing", Procedia Computer Science, Vol.29, pp.637-647, 2014. ISSN 1877-0509. This will be presented at the ICCS 2014 conference in June 2014.

dynamics are extensively described in Kerschen et al. (2006). There is no mention of heuristic techniques; the evolutionary approach of this chapter does not appear in this overview. The advent of modern heuristics and the steady increase in computing power has enormously boosted possibilities for regression of all kinds (e.g. by artificial neural networks), but many techniques do not provide clear insights into the working of the system. In applied hydrodynamics, working with pitch-black models is seen as an inconvenient drawback. There is always a desire to build feeling and confidence along with building the model.

The second-order ODE of motion together with initial conditions form an initial value problem (or Cauchy problem). Solving it numerically, for example with Runge-Kutta schemes, is a forward problem: the time series $y(t)$ is unknown. If, on the contrary, $y(t)$ is known, then the inverse task of finding the ODE that produced it is a system identification problem, see Figure 8.1.

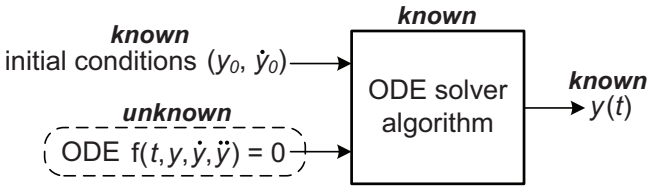
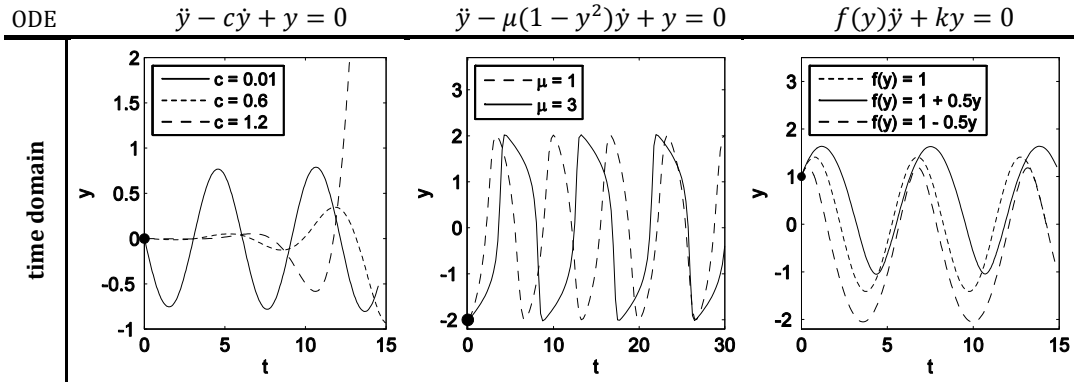


Figure 8.1. The problem of finding the ODE responsible for producing a given one-dimensional time series. The ODE can have an unknown algebraic structure and/or unknown coefficients. This scheme is reminiscent of the optimization scheme in Section 1.2.

If only the numerical coefficients of the ODE are unknown, but the structure of the expression is known, an optimization problem in continuous space (of dimension n equal to the number of coefficients) needs to be solved. Let us look at a few instances of ODEs in Figure 8.2.



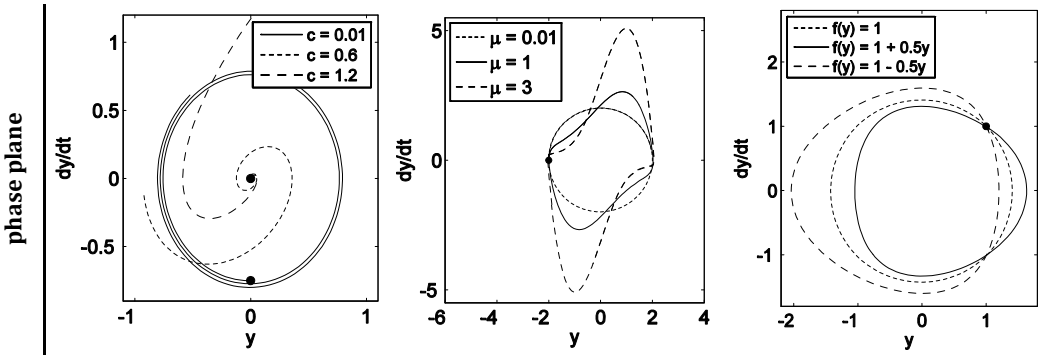


Figure 8.2. Vibrations in time domain (upper row) and phase plane (lower row). Left column: constant negative damping; middle column: non-linear damping by Van der Pol oscillator; right column: non-linear mass term. The corresponding ODE equations are written on top; y is the vertical displacement of the mass. The initial states are indicated by thick dots.

The first two columns in Figure 8.2 show quintessential self-excitation: a negative damping constant and the Van der Pol oscillator. The latter famous example has a non-linear damping term and for high enough values of the parameter μ, so-called 'relaxation vibrations' occur which show sudden transitions with short moments of high velocity at certain parts of the period. The third example in the right column of Figure 8.2 shows an undamped oscillation with a non-linear mass term. A standard way of depicting non-linearities is in the phase-plane; a deformed limit cycle is a good telltale of non-linear behaviour.

The aim in this chapter is to explore evolutionary computing (EC) for uncovering ODEs describing non-linear, in particular self-excited vibration types. This identification overcomes the first and the third obstacle mentioned in this section and certainly meets the criterion of fostering 'Fingerspitzengefühl'. In fact, knowing the exact ODE reveals very useful information for the analyst not found from Fourier analyses: non-linear terms (dependencies on frequency) and added mass terms. The desire is to contribute to the development of a tool for detecting self-excited vibrations before they grow beyond safe limits and cause dangerously high dynamic forces on the structure.

First a ridiculously short introduction to heuristics and evolutionary computing is given in Section 8.2. Then Section 8.3 zooms in on differential evolution, which is applied to the problem of system identification (Section 8.4). Section 8.5 shares thoughts on and gives a proof of concept of the application of the presented method to the experimental data of Chapter 5. Section 8.6 then gives remarks on genetic programming and symbolic regression and Section 8.7 draws conclusions and gives an outlook on future work.

8.2 Meta-heuristics and evolutionary computing

In optimization there are two kinds of methods: exact methods that guarantee to find an optimal solution, and heuristic methods that do not give this guarantee. The issue with exact methods is that they become practically unusable for larger and harder optimization problems. Heuristic optimization methods were developed as faster alternatives, but without the assurance of returning the optimal solution. Rothlauf (2011) remarks that standard

heuristics "are problem-specific and exploit known rules of thumb, tricks or simplifications to obtain a high-quality solution". The same author distinguishes three classes of heuristic optimization methods:

- heuristics (divided in construction heuristics and improvement heuristics)
- approximation algorithms (which provide a quality bound for the solution)
- modern heuristics, also called meta-heuristics.

A typical feature of meta-heuristics is the iterative improvement of solutions, while at the same time solutions of lower quality are allowed to be part of the search. This has to do with exploitation and exploration phases in the search. Meta-heuristics are seen as general-purpose techniques that can be applied to many different problems.

A popular meta-heuristic, evolutionary computing (EC) is a form of computational intelligence inspired by nature. Evolutionary algorithms (EAs) are stochastic population-based search algorithms with a common basic loop: the reproduction cycle. A comprehensive introduction to EC is given by Eiben and Smith (2007). In short, a population of candidate solutions (individuals) gradually evolves under the influence of one or more objectives dictated by a fitness function. In the reproduction cycle, successively parents are selected from the population to engage in variation operations (recombination and mutation), giving offspring, from which eventually individuals are selected to move to the next generation (survivor selection). Figure 8.3 depicts a (debatable) family tree of evolutionary algorithms.

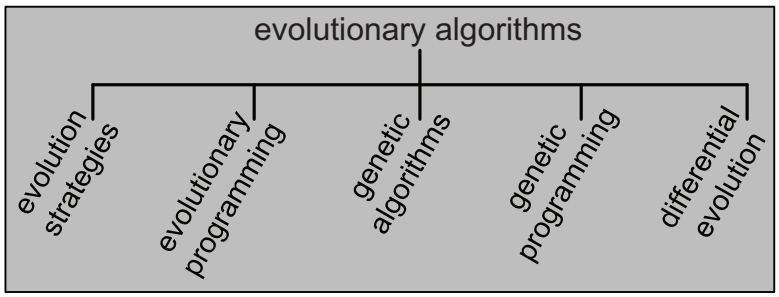


Figure 8.3. Family tree of evolutionary algorithms, listed chronologically from left to right.

8.3 Differential evolution

Not generally considered as one of the classic EAs, differential evolution (DE) is a competitive derivative-free global optimization method introduced by Storn and Price (1997). Its distinguishing feature is the use of difference vectors in the mutation operation. The global scheme of DE is shown in Figure 8.4, adopting the terminology by Das and Sugantham (2011). This is the most basic version with a mutation based on three vectors and strategy parameters *F* and *CR*.

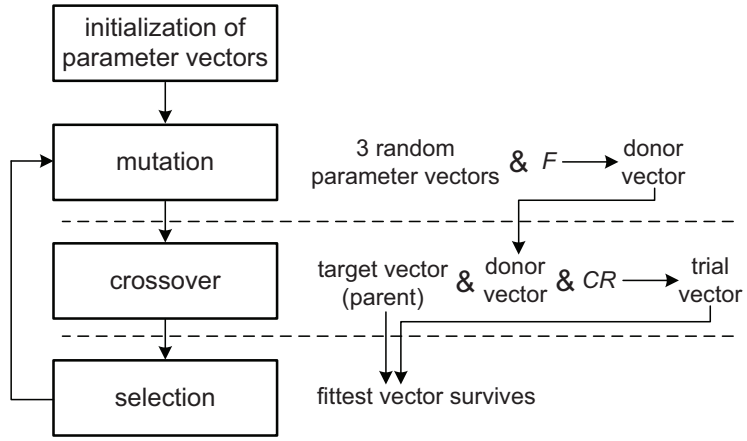


Figure 8.4. Global scheme of basic differential evolution: definition of vectors.

It has a natural robustness that makes it stand out from some earlier EAs. Its performance has grown by several improvements (Das and Suganthan, 2011), most notably by the use of parameter control. In this technique the strategy parameters change during the run, see Eiben et al. (1999). We apply a recent version of DE by Choi et al. (2013) that has self-adaptive parameter control. They call it 'Cauchy DE' because the strategy parameters are varied by drawing from Cauchy distributions. The pseudo-code of the algorithm is given below.

Initialization
-Initialize population of NP vector individuals $X_{1,G}, \ldots, X_{NP,G}$ where $X_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{C,i,G}]$ and where C is the number of coefficients that are being evolved, G the generation ($G = 0, \ldots, G_{max}$). Entries $x_{j,i,0}$ are uniformly random from [-1,1] for $i = 1, \ldots, NP$ and $j = 1, \ldots, C$.
-Initialize control parameters $CR_{i,0} = 0.25$, $F_{i,0} = 0.6$ (acc. to Choi et al. 2013) and adaptation parameters $CR_{avg,0} = CR_{i,0}$ and $F_{avg,0} = F_{i,0}$
-Generate target data $y(t)$, $y'(t)$ and divide into training and test sets.

```
FOR R = 1 to R_max DO              % run loop
  FOR G = 1 to G_max DO            % generation loop
    FOR i = 1 to NP DO             % individuals loop
      Main loop: Differential Evolution
      -Determine fitness f(X_i,G) of individuals (see routine)
      -Mutation: generate a mutant vector V_i,G = X_r1 + F_i,G*(X_r2 – X_r3) from three donor vectors X_r1, X_r2,
       X_r3 randomly selected from the individuals of generation G-1.
      -Crossover: generate a trial vector U_i,G composed of u_j,i,G (j = 1, … , C) by applying the rule IF
       rand[0,1] ≤ CR_i,G-1 OR j = j_rand THEN u_j,i,G = v_j,i,G , ELSE u_j,i,G = x_j,i,G-1 , where j_rand is a random
       integer 1 ≤ j_rand ≤ C and v_j,i,G is an entry of V_i,G.
      -Selection: determine fitness f(U_i,G) of trial vectors. IF f(U_i,G) ≥ f(X_i,G-1) THEN X_i,G = U_i,G and
       inherit associated fitness and control parameters, ELSE X_i,G = X_i,G-1 and leave fitness and
       control parameters unchanged.
    END FOR
    -Update control parameters F_i,G and CR_i,G by adding a randomly drawn number from the Cauchy
     distribution Ca(0,0.1) to the mean value of the control parameters of all successfully evolved
     vectors. Truncate if necessary.
    -Replace a non-fittest individual by a newly generated individual.
  END FOR
```

Post analysis
Compute test error of run R by solving the ODE with the winning set of coefficients and determining the mean absolute error of all predicted values compared to the test values.
END FOR
Compute mean duration and mean and min of test errors of all runs $R…R_{max}$.

Fitness computation
-Insert the coefficients of each candidate vector in the fixed, assumed ODE equation structure.
-Apply Runge-Kutta, with adaptive step-size and predetermined relative error tolerance for numerical integration.
-Fitness := -1*MAE*penalty, where MAE is the mean absolute error of the training data compared to the result from solving the ODE with candidate coefficients. The penalty punishes candidate models for which the integration failed to determine values at all training times, penalty := ((size of training set – size of candidate set) / size of candidate set)*100 and penalty = 1 if the integration was completely successful.

8.4 Identifying self-excited vibrations using differential evolution

8.4.1 Approach

The aim of this section is to identify vibrations from only a displacement (output) signal $y(t)$. That is, without using signals of pressures on the gates (input). This is preferably done in a way that permits speed-up to practical time frames for early-warning systems. It will be assumed that the main part of the structure is already known: the second order ODE for all vibrations without external forcing, with optional unknown non-linear terms. Differential evolution is used to optimise a set of coefficients of these ODEs.

A generated synthetic data set is randomly divided into a training set and a test set, based on a chosen percentage of data to be used for training. After the evolution has ended, the unseen target points are used to quantify the predictive power of the candidate model by computing a test error. In order to compare the test errors of different target data sets, we normalize the mean absolute test error (MAE) as follows:

$$NMAE(\hat{\boldsymbol{y}}, \boldsymbol{y}) := \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - \bar{y}}{\sigma_y} - \frac{y_i - \bar{y}}{\sigma_y}\right| = \frac{1}{n}\frac{1}{\sigma_y}\sum_{i=1}^{n}|\hat{y}_i - y_i| = \frac{MAE(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\sigma_y}, \qquad (8.1)$$

where $\hat{\boldsymbol{y}}$ contains the predictions and $\boldsymbol{y}$ the target values for testing, $\bar{y}$ is the mean of $\boldsymbol{y}$ and σ_y is the standard deviation of $\boldsymbol{y}$. All results in the figures and in the table of this paper are given as normalized mean absolute test errors (NMAE). The computations were done unparallellized on a single Intel i7 processor, 2.93 GHz, 8 Gb RAM.

The numerical experiments consist of three parts: a validation case, the self-excited cases and a sensitivity analysis. The results are reported in Sections 8.4.2 and 8.4.3.
The case of forced vibrations for a linear system with constant coefficients is used for validation:

$$m\ddot{y} + c\dot{y} + ky = F_0 \sin(\omega t), \qquad (8.2)$$

where y is the displacement, t is time (the independent variable) and all other symbols are physical constants. Newtonian notation is used for time derivatives. Together with the real-valued initial conditions y_0 and $\dot{y}_0$, equation 8.2 constitutes an initial value problem that will be solved in two ways: (i) non-linear regression on the analytical solution and (ii) regression on a fixed ODE structure:
The first approach uses the sum of the general and particular solution of forced vibration with viscous damping as an assumed equation structure:

$$y(t) = C_1 e^{C_2 t} \sin(C_3 t + C_4) + C_5 \sin(C_6 t + C_7), \qquad C_i \in \mathbb{R} \tag{8.3}$$

The second approach stays at the level of ODE:

$$XC_1'\ddot{y} + C_2'\dot{y} + C_3'y = C_4'\sin(C_5' t), \quad y_0 = C_6', \quad \dot{y}_0 = C_7', \qquad C_i' \in \mathbb{R} \tag{8.4}$$

For both approaches, the coefficients are initialized randomly between -1 and 1. They are stored in a seven-dimensional vector and optimized via DE, as described in the pseudo-code in Section 8.3. A variation of the second approach where only five coefficients are evolved is also considered, where the initial conditions (IC) are assumed known.

Practically all non-linear vibration problems defy full analytical treatment, so there is no closed-form equation available for $y(t)$. For these problems we work with the ODE structure $m\ddot{y} + c\dot{y} + ky = 0$, where mass m or damping c are replaced by a first or second order polynomial term in y to account for the non-linearity. The results of this are summarised in Section 8.4.2. Also, a non-linear mass system and a system with time-varying stiffness (Mathieu equation) are examined. These are all unforced oscillators where chaos does not play a role.

8.4.2 Results

Validation: Forced vibrations with constant coefficients
The following target function is defined for the validation runs:

$$2.5\ddot{y} + 0.35\dot{y} + 0.3y = 3.0\sin(2.2t), \text{with initial conditions } y_0 = 0, \dot{y}_0 = 0.25. \tag{8.5}$$

Figure 8.5 (left) shows the sampled target data-set and an example of a candidate solution. The total data set consists of 465 points, meets the Nyquist criterion, and is split in training and test data in different ratios (in Figure 8.5 half of the data are training points, and half are test points). Population size was set at 80 and 250 generations were computed.

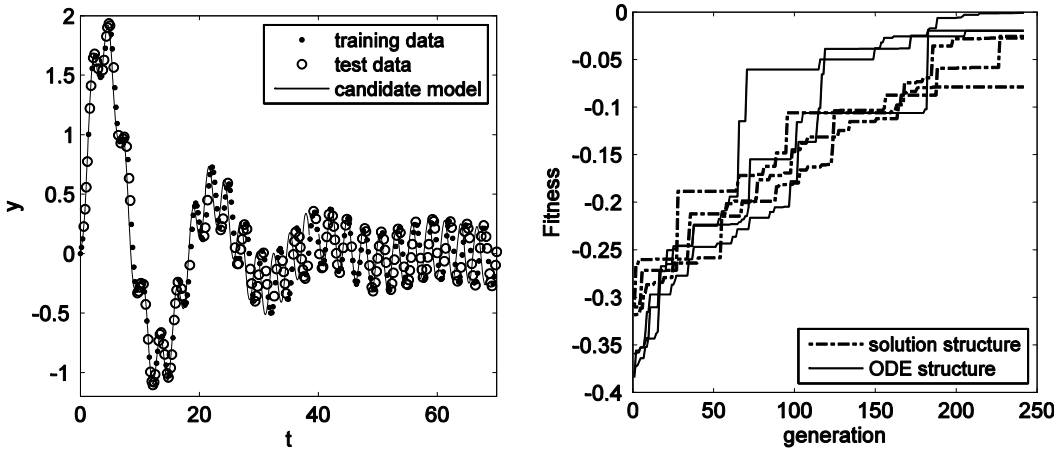


Figure 8.5. Differential evolution applied to regression on the signal of a forced vibration. Left: the target data. In this example 50% of the data is used for training. Right: The best-so-far fitness of three runs, for an assumed structure of the analytical solution and for the ODE structure.

The right plot in Figure 8.5 shows three examples of how the solutions improved with generations. The two applied expression structures were laid out in the previous section, the only necessary addition is that the fitness evaluation of the analytical solution structure runs differs from the pseudo-code in Section 8.3 because there is no need to solve an ODE; the candidate values follow right away after substitution in the assumed $y(t)$ expression. The resulting coefficients reflect the multimodality, since for example $\sin(t) = \sin(t+2\pi)$. It was generally found that the less successful computed functions capture the low-frequency damped free vibration quite well, but give a rather poor estimate of the forced vibration.

Figure 8.6 below gives an overview of the results based on 10 runs per plotted point. The plot on the left gives test errors expressed as NMAE, according to equation 8.1. The plot on the right shows the average runtime in seconds.

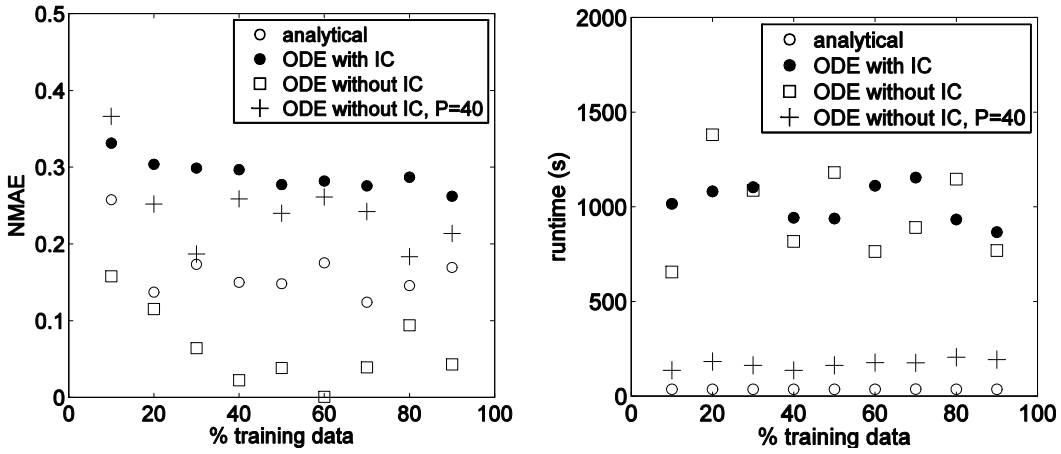


Figure 8.6. Results of evolving the forced vibration based on analytical solution structure and ODE structure as a function of the percentage of data used for training. Left: test errors (NMAE); right: computing time. IC stands for initial conditions, P stands for population size. Every point represents an average of 10 runs, for an evolution of 250 generations with a population of 80; except for the ODE with initial conditions, for which only five lengthy runs were made for each training data set.

The plots show that the analytical structure requires the least computation time, but it is significantly less accurate than the ODE structure with five evolved coefficients where the initial conditions are known ("ODE without IC"). The analytical structure is more accurate than the ODE case that also evolves the two initial conditions ("ODE with IC"). An attempt to reduce the computation time for the ODE structure by using a smaller population of 40 ("ODE without IC, P=40") resulted in higher test errors and computation times comparable to the analytical runs. The results show that including or excluding the two initial conditions makes no difference for computation time. Additionally, the validation proves that there is little overall dependence on the percentage of data used for training. The test errors are only slightly worse when less than 40% of the data is used for training. Computational factors related to the convergence of the DE algorithm and the ODE solution process are apparently dominant. In particular, it was found that the settings of relative tolerance that determine the number of iterations of the ODE solver during the error computation have a profound influence on the standard deviation of the achieved total runtimes.

Main runs

Table 8.1 shows the results of computing coefficients for various self-excited vibrations. All computations had a population of 50 individuals with 120 generations computed, and used 50% of 500 data points for training, and the remainder for testing. For each case the mean and standard deviation of the NMAE test error is computed over 25 evolutionary runs.

Table 8.1. Computation results based on normalized mean absolute error (NMAE). #C denotes the number of evolved coefficients. For each case, 25 runs of 120 generations were done with a population size of 50, using 250 training points and 250 test points.

vibration	target ODE	#C	average of NMAE	standard deviation of NMAE
linear, constant coeff.*	$2.5\ddot{y} + 0.35\dot{y} + 0.3y = 3.0\sin(2.2t)$	5	$39.2{\cdot}10^{-3}$	$69.5{\cdot}10^{-3}$
negative damping	$\ddot{y} - 0.1434\dot{y} + y = 0$	1	$3.44{\cdot}10^{-3}$	$0.89{\cdot}10^{-18}$
		2	$3.07{\cdot}10^{-3}$	$1.77{\cdot}10^{-18}$
		3**	$3.23{\cdot}10^{-3}$	$1.88{\cdot}10^{-3}$
non-linear damping	$\ddot{y} + (-0.450 + 2.728y + 1.903y^2)\dot{y} + y = 0$	3	$287{\cdot}10^{-3}$	$373{\cdot}10^{-3}$
Van der Pol oscillator	$\ddot{y} - 1.2218(1 - y^2)\dot{y} + y = 0$	2	$3.13{\cdot}10^{-3}$	$0.42{\cdot}10^{-3}$
		3	$3.26{\cdot}10^{-3}$	$0.82{\cdot}10^{-3}$
non-linear mass	$(1.7120 - 1.4815y + 0.4903y^2)\ddot{y} + y = 0$	3	$3.28{\cdot}10^{-3}$	$0.0776{\cdot}10^{-3}$
Mathieu equation***	$\ddot{y} + (0.25 + 0.34\sin(2.18t))y = 0$	3	$70.8{\cdot}10^{-3}$	$25.9{\cdot}10^{-3}$

* Based on the ODE-runs with 50% training data and without evolving initial conditions.
** Only three runs were made due to poor convergence of ODE solver.
*** The Mathieu equation describes not self-excited but parametrically excited vibrations.

The results in Table 8.1 show that the constant negative damping and non-linear mass cases have low test errors compared to the validation case of the forced linear vibration with constant coefficients. Moreover, their test errors show very little variation. For the negative damping case, it makes no difference whether the constant coefficient is found using a single coefficient, C_1, or a linear term with two coefficients, C_1+C_2y, or a second-order polynomial term with three coefficients, C_1+C_2y+C_3y^2. Similarly, the Van der Pol oscillator shows a small, insignificant deterioration when a linear term is added to the C_1+C_2y^2 term that is strictly required. The poor result for the non-linear damping case is due to suboptimal convergence of eight runs out of 25. Extending the runs to more generations will most likely improve the mean NMAE. The same can be said of the Mathieu equation, which belongs to the class of parametrically excited vibrations.

The test errors of the best runs are plotted in Figure 8.7 as function of their computation times. The errors are the minima of the NMAE values of the 25 runs for the vibrations mentioned in Table 8.1. There are two outliers: the negative damping evolved with a polynomial term took much longer to compute and the best run for the Mathieu equation is significantly less accurate. It is remarkable that the best non-linear damping run is slightly better than the other non-linear cases.

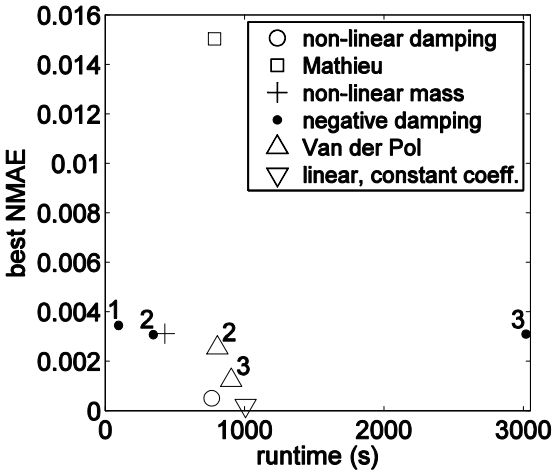


Figure 8.7. The best achieved test errors (NMAE) out of 25 runs as function of the computing time of the best runs for the cases listed in Table 8.1. The numbers inside the figure denote the number of evolved coefficients.

8.4.3 Sensitivity analysis

A sensitivity analysis was done to study the effect of different population sizes, number of generations and tolerance settings of the ODE solver. The results are summarized in Figure 8.8. The test errors are NMAE values over 25 runs, as defined in equation 8.1. The sensitivity analysis is based on the three-dimensional optimization problem of finding the coefficients of an unforced vibration with non-linear damping term $-0.4501+1.0283y+1.903y^2$.

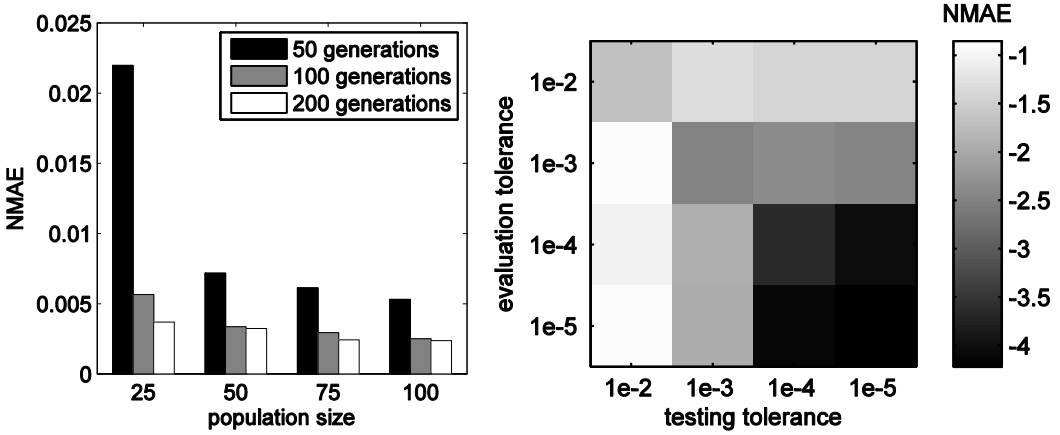


Figure 8.8. Sensitivity analysis results. Left: sensitivity on population size and number of generations showing normalized mean absolute errors (NMAE) of 25 runs. Right: sensitivity of test error (NMAE) on tolerances of ODE solvers for computing fitness values ("evaluation tolerance") and test errors ("testing tolerance"). On the axes, "1e-2" means 10^{-2}, etc. and the grey scale refers to base-10 logarithms of NMAE values.

Population size and number of generations
The results show that a population size of 50 yields far better results than a population size of 25 (Figure 8.8, left), and 100 generations score far better than 50 generations. Further increases in population size and generations give considerably smaller improvements.

Solver tolerance
The right plot of Figure 8.8 shows the effect of different combinations of termination settings for the numerical integration algorithm used for computing errors of the candidate ODE models ("evaluation tolerance") and of the winning model ("testing tolerance"). Unsurprisingly, stricter (lower) tolerances lead to more accurate results ("-4" in the colourbar refers to a NMAE value of 10^{-4}, etc.). The worst results occur for a strict evaluation tolerance in combination with a coarse testing tolerance. Furthermore, it is seen that relatively low test errors are found if the relative tolerance of the ODE solver for evaluation and testing are the same. Of course we need to realise that the lower the evaluation tolerance, the longer on average the runs are likely to be and that the testing tolerance should always be relatively strict in order to make a fair judgment. Based on these observations, a relative evaluation tolerance of 10^{-3} and a testing tolerance of 10^{-5} were chosen for the simulations in the previous sections.

Solver algorithm
Apart from the choice of residual error as a model parameter, a suitable choice of integration algorithm is also paramount accuracy and runtime. MATLAB suggests the use of specific build-in ODE solvers for stiff problems (MathWorks, 2010). Quateroni et al. (2010) discuss the application of different ODE solvers in MATLAB. From this, stiff solver 'ode15s' was selected to compete with 'ode45', the standard solver that was used in all work in this chapter and the next. The Van der Pol oscillator was used as test problem: increasing μ values make the problem stiffer. The results of 25 runs for each solver are plotted in Figure 8.9.

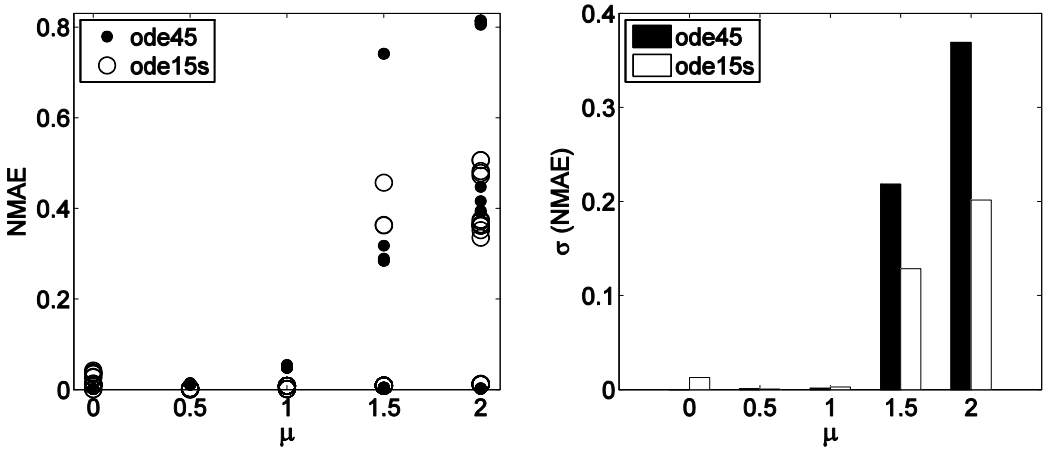


Figure 8.9. Comparing ODE solvers of MATLAB. Left: NMAE test errors over 25 runs plotted as a function of the μ parameter of a Van der Pol oscillator test problem with two constants in the non-linear damping term. All other settings were identical to the main runs in Section 8.4.2. Right: the standard deviation of these errors.

The results indicate that for $\mu \leq 1.0$, both solvers have much lower errors and smaller spread than for $\mu > 1.0$. Moreover, the stiff solver 'ode15s' has lower maximum test errors and a smaller standard deviation of errors for $\mu > 1.0$. It was indeed expected that the stiff solver performs relatively well for increasing μ. This probably becomes even more apparent for stiffer test problems. Next to testing the ODE solvers, this was of course at the same time a test for the DE algorithm to solve increasingly hard optimisation problems.

8.4.4 Improving the search by including results from FFT

An idea put forward by Howard and Oakley (1994) is to use information from the frequency domain of the target data in the evolutionary search. This section gives a brief exploration of this idea within the context of the preceding numerical experiments. The spectral information can for instance be applied to introduce a bias at the initialisation or during the search. Let us have a look how the period and stability of the total system relate to the coefficients.

The following non-linear ODE with a polynomial damping factor is considered:

$$\ddot{y} - \mu(1 - \nu y - \rho y^2)\dot{y} + y = 0, \tag{8.6}$$

which includes the Van der Pol oscillator as a special case. A few definitions:

$\omega_0 = \sqrt{k/m}$	undamped natural radial frequency, as before
ω_1	natural radial frequency of the system as a whole
$\Omega_1 = \omega_1/\omega_0$	standardized natural radial frequency of the system as a whole

In equation 8.2, $k = m = 1$, such that for this system $\omega_0 = 1$ and $\Omega_1 = \omega_1$. If a one-to-one relation between Ω_1 and the coefficients μ, ν and ρ would be found, then measuring Ω_1 would immediately give good estimates of their values. However, the effect of non-linear damping on frequency of free vibrations is much less obvious than for non-linearity in the springs. Two illustrative cases are treated.

Case 1: $\nu = 0$, $\rho = 1$

Schmild and Guicking (1980) give a good approximation of Ω_1 for the standard unforced Van der Pol system. Figure 8.10 shows three analytically derived relations mentioned in Schmild and Guicking (1980) as lines. Numerical experiments were done by simply solving the Van der Pol equation numerically and performing a FFT of the result, to find Ω_1 (plotted as +'s in Figure 8.10).

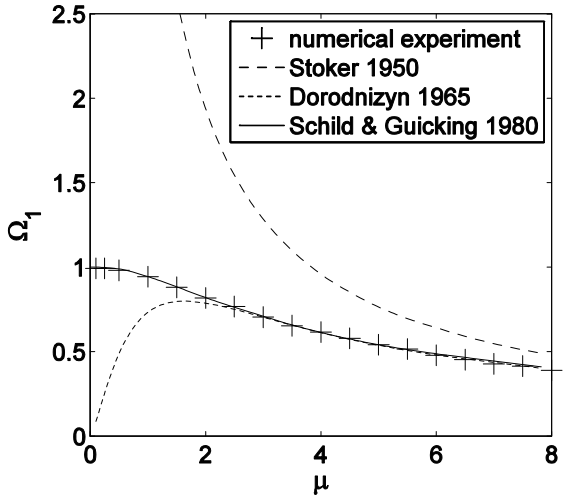


Figure 8.10. Analytical estimates of the natural frequency of the Van der Pol system. The values from the numerical experiment were achieved by solving the ODE over a reasonably long time interval and subsequently performing a fast Fourier transform with triangular windowed smoothing to find the dominant frequency.

Indeed, the analytical model and the Schild and Guicking model coincide. This implies that for the standard Van der Pol ODE, performing a FFT gives μ directly and further computations are not necessary.

Case 2: $\rho = 1$
Now, a more difficult case: $\ddot{y} - \mu(1 - \nu y - y^2)\dot{y} + y = 0$. A series of experiments was done for various μ and ν. Figure 8.11 (left) shows the phase plane for three different ν for constant μ. Figure 8.11 (right) gives the equivalent of Figure 8.10.

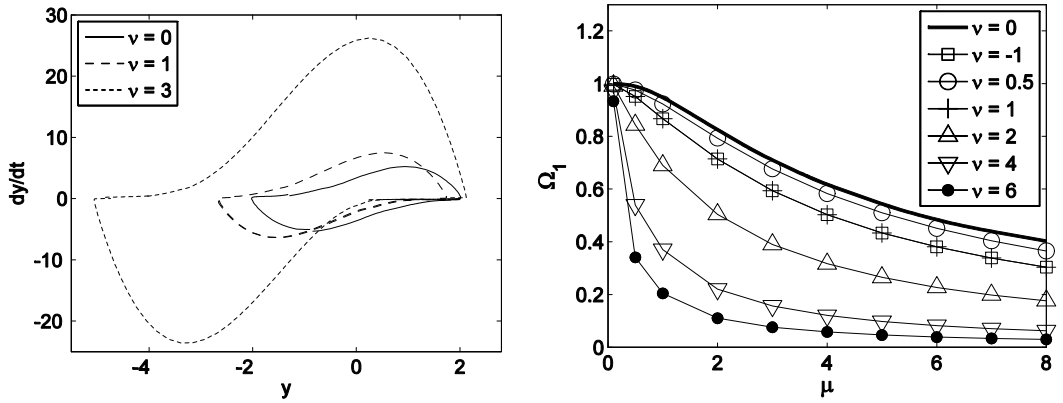


Figure 8.11. Non-linear damping simulations. Left: μ = 3, varying linear term coefficient ν. Right: normalized natural frequency of the oscillator as function of μ and ν.

It is seen from Figure 8.11 (right) that different (μ, ν)-pairs give identical system frequency. The plot shows that the sign of ν is irrelevant for Ω_1 and hence cannot be found from Fourier analysis, which is also true for the damping coefficient c in the standard linear ODE. These findings suggest the following procedure for applying spectral information into system identification computations for this ODE:

- FFT on the (mildly filtered) time signal gives a number of spectral peaks. Take the highest peak (at nonzero frequency) as the dominant system frequency and plot this as a horizontal line in the μ versus Ω_1 − diagram (dashed line in Figure 8.12, left).
- It is then deduced from this diagram that the observed frequency could have been generated by a combination of (μ, ν)-values according to all intersections with the line of constant Ω_1, this line is shown in Figure 8.12 (right).
- The DE search algorithm should then be adjusted so it biases values in the neighbourhood of this line.

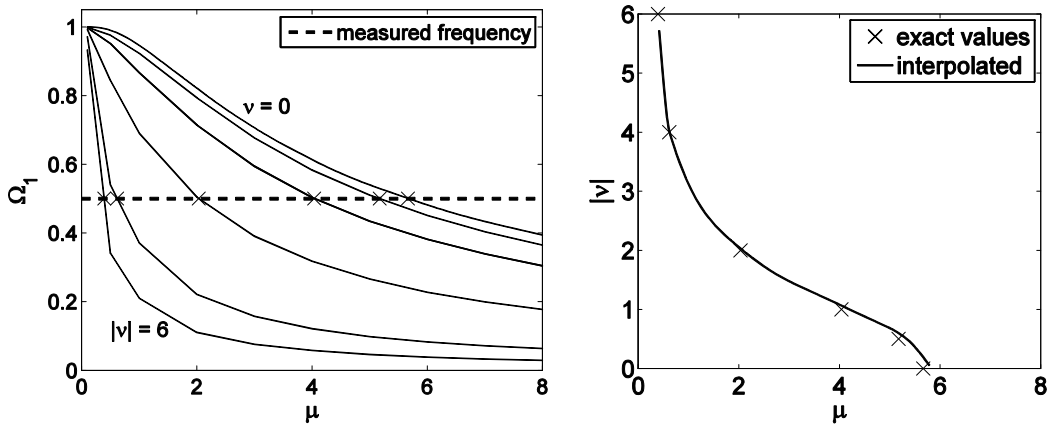


Figure 8.12. Estimating coefficients from spectral information. Left: plotting measured frequency. Right: interpolated line of possible (μ, ν)-values.

Additionally, It is useful to have information about the stability of the system as a function of the three parameters, so that for instance candidate models in unstable regions can be skipped. Figure 8.13 shows two Monte Carlo simulations of solution stabilities for a range of μ and ν values, keeping ρ = 1 (left) and ρ = 0 (right). Having this information before starting could improve the evolutionary search.

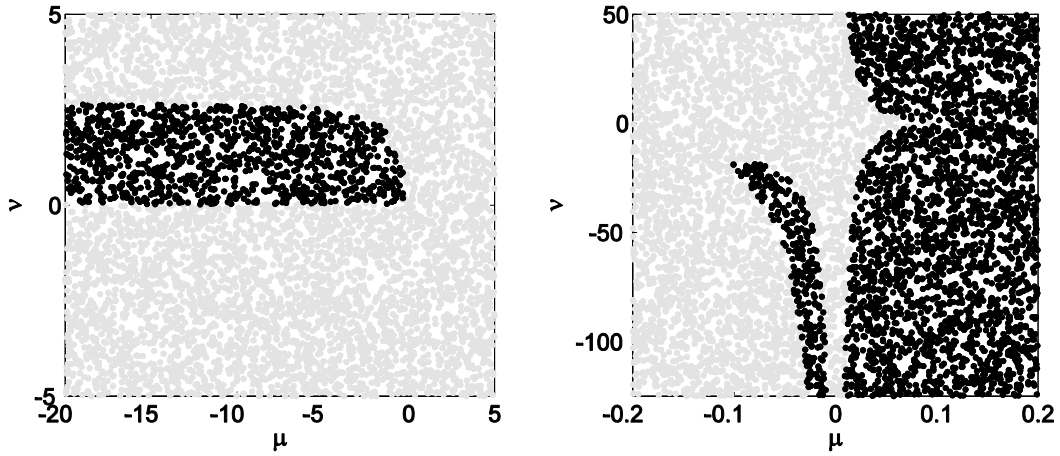


Figure 8.13. Monte Carlo simulations of stability in (μ, ν)-plane for ρ = 1 (left) and ρ = 0 (right). The ODE was integrated over ten periods with a relative solver accuracy of 1e-3 ('ode45' solver). Light grey dots: full time computed (stable), black dots: solver failed or solution was out of set bounds before end time was reached (unstable). Varying initial conditions did not influence the region boundaries. 6000 trials were made for both plots.

Numerical observations of stability such as these are certainly not impossible to explain, but generally the analytical analyses become inconvenient. Surely, naive intuition still has some value in relatively easy cases: for ρ = 0 (Figure 8.13, right), the damping term reads $(-\mu + \nu\mu y)\dot{y}$ and one could have guessed that, roughly, $\mu > 0$ gives unstable and $\mu < 0$ gives stable solutions. If the damping term would have had the form $F(|y|, |\dot{y}|)\dot{y}$, with F a polynomial, then a positive constant term of this polynomial would always imply a stable, and a negative constant an unstable equilibrium position. Schmidt and Tondl (1986) name this as a simple example of a self-excited system with a singular point in the phase-plane, which represents a single equilibrium position. Alternatively, the stability analysis of self-excited oscillations focuses around (stable or unstable) limit cycles (e.g. Figure 8.11, left). However, most non-linear cases require a large toolbox of analytical methods for revealing stability properties, especially when more than one d.o.f. exists or when external or parametric excitation is added. The textbooks by Schmidt and Tondl (1986) and Verhulst (1996) provide bridges to advanced literature on non-linear vibration analysis.

8.4.5 Discussion

It has been shown that coefficients sets of ODEs can be found from time series such that the errors between target and model are small. But to determine actual values of independent coefficients, knowledge of the physical domain is required. This can be done in various ways, e.g. solving $m\ddot{y} + c\dot{y} + ky = 0$ as a constrained optimization problem by prescribing upper and lower bounds for m and k; or by solving $\ddot{y} + A\dot{y} + By = 0$ with $A = 2\zeta\omega_0$ and $B = \omega_0^2$ and applying available knowledge on the mass afterwards. All knowledge and estimation techniques from Section 2.4 are welcomed for formulating constraints. The goal is to let the system identification focus on finding the toughest physical quantities. Structural mass and stiffness are obviously more easily estimated from mechanical pre-studies than hydraulic damping is from a hydrodynamic pre-study. See also Section 8.6.

The ODE solver tests helped to reveal the dilemma of using different integration schemes for specific problems, versus maintaining robustness – at the cost of longer computation times or loss of accuracy. Too coarse solvers have the effect that promising candidates do not come out on top. The next chapter attempts to make a first step at automated customisation of solver algorithms.

8.5 Application of evolutionary system identification to experimental data

The experimental data set analysed in Chapter 5 consists of response forces for quasi-stationary hydrodynamic situations of constant discharge, water levels and mean gate opening, and where a certain dynamic equilibrium between hydrodynamics and structural response has been achieved. The absence of excitation signals of pressures acting on the gate makes it hard or impossible for many techniques to identify the system. Ideally, we would like to find the full motion equation, the displacement ODE, from the measured force signal. Conforth and Lipson (2013) discuss possibilities and examples of inferring ODEs for systems with hidden variables, but do not give helpful tips for the present case.

Looking at the acquired quasi-stationary signals, these can roughly be divided into three groups: low-level noise, regular high-level amplitude oscillations and transitions between these two. The irregular noisy signals sometimes display vibrations at the natural frequency, but these typically die out again after a few seconds. This means that the present (external) excitation is damped. In the interesting transitional zone, occasional higher amplitudes feed energy to the oscillation. Here it seems that the damping consists of both positive and negative contributions, which may vary with time and/or with displacement (amplitude). Then, at large amplitudes, the sines have become very regular; precisely sufficient structural damping has been mobilised to balance the self-excitation forces so that the amplitudes are as good as stable.

For system identification only the transitional signals are interesting. The challenge lies in the fact that the damping force is always small compared to the other forces; and thus potential damping non-linearities appear only as very slight deviations from the standard sine. Moreover, non-linearity in the inertia term due to added mass effects (which are usually small) can cause a distortion. A number of preliminary identification tests on raw transitional signals with DE based on plausible equation structures (as investigated in Section 8.4.2) proved demanding. The issue with studying the irregularities is that a lot of analysis (many signals, many parts of one signal) is needed before meaningful hypotheses come about.

Instead, an unsteady signal of the same experiment series is used to explain a possible application procedure. This concerns merely a proof of concept. Without giving unnecessary details for this illustrative example, at a constant installed stiffness the water levels were gradually (but unevenly) lowered giving a changing head difference and response frequency. As V_r also slowly changed, a vibration region was encountered and vibrations started to appear. The analysed part of the signal $F(t)$, measured at 200 Hz, captures 10 seconds of a growth phase of regular vibrations. The experimental data is depicted as dots in Figure 8.14.

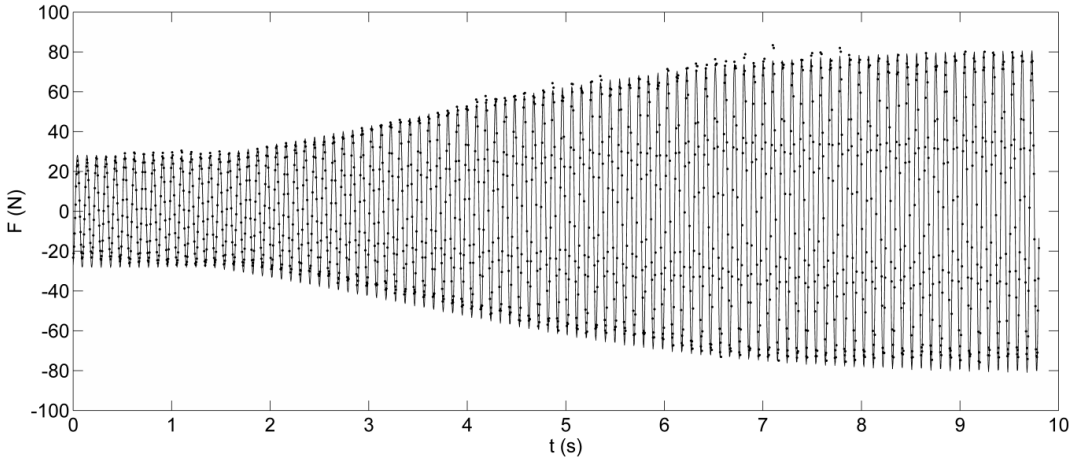


Figure 8.14. Part of a support force signal (same set-up as in Chapter 5) under unsteady hydrodynamic conditions. The points are measurement data, the line is the identified model. The support force is computed from the equation of motion, the coefficients of which were evolved with differential evolution.

The main issue is dealing with the fact that the support force $F(t)$ was measured while we are interested in the motion equation $\ddot{y} = f(y, \dot{y}, t)$. Some trial and error resulted in the following approach:
- To find the initial conditions, an undamped force signal is derived with an amplitude that is the average amplitude of the first few periods of the target signal. Assuming $c = 0$ and linearity, the support force is $F = kA\sin\omega t$ for an assumed motion $y = A\sin\omega t$. This gives right away $y_0 = F_0/k$, with the subindex 0 indicating values at $t = 0$, because the stiffness k was measured. The initial velocity $\dot{y}_0$ and angle are evolved in a DE pre-run with structure $F(t) = C_1\sin(C_2t + C_3)$. Then C_3 is the required initial angle and $\dot{y}_0 = \omega C_1/k$.
- As a first check, the system of equations consisting of $\ddot{y} + \omega^2 y = 0$, $F = ky$, y_0, $\dot{y}_0$ can be solved to retrieve an undamped force signal with equivalent amplitude, that should match the first part of the signal where damping has little influence.
- Then, in the main run the evolved initial conditions (y_0, $\dot{y}_0$) are used in the evolution of coefficients of ODEs $\ddot{y} = f(y, \dot{y}, t)$, with structures hypothesized from theory. An error-based fitness function is used based on the mean absolute error:

$$\text{fitness} = -\frac{1}{n}\left|f_1(y,t)\dot{y}(t) + f_2(y,t)y(t) - F_{\text{exp}}(t)\right|, \tag{8.7}$$

where the f_1-term is the evolved damping force and f_2 is the evolved stiffness force, F_{exp} is the measured support force and n the number of data points.
- For the pre-run and the main run, all measured points were used in training and hence no test errors are computed. A comparison of errors (after normalisation) between pre-run and main run indicates of the conjectured model structure is an improvement with respect to a zero-damping model. This helps to confirm or refute analytical theories.

Figure 8.14 shows the best fitting model (shown as a line) with a non-linear damping term. The plotted line is the sum of the cimputed damping and stiffness terms. Two notable,

manual additions were necessary to adjust the model structure: the moment that the damping first becomes negative could not be found, this was therefore added as a step-function $H(t)$, and a time-dependent term was added to the stiffness to account for a slowly dropping frequency due to the falling water levels (not visible with the naked eye from the plot). The evolved ODE is thus:

$$19.14\ddot{y} + H(t-1.4)(-12.18 + 4.473 \cdot 10^7 y^2)\dot{y} + 81360(1 - 2.986 \cdot 10^{-3} H(t-1.4)t)y = 0,$$
$$\text{with } y_0 = -0.3368 \text{ mm and } \dot{y}_0 = 5.8 \text{ mm/s}. \qquad (8.8)$$

In this expression, only the installed stiffness (81360 N/m) has been used as domain knowledge. The only applied pre-processing was subtraction of the moving average. The coefficients are physical; 19.14 is the total vibrating mass in kg, the negative hydrodynamic damping term is -12.18 Ns/m and the extra structural damping entering at high amplitude is $4.473 \cdot 10^7$ Ns/m^3. How sensible these values are is impossible to say without further analysis or comparison with other data, because all knowledge of the experimental conditions is already used. The time-varying stiffness is perhaps unlikely physically; the frequency probably changes via the inertial term. These two terms are notoriously hard to distinguish since they are in phase. It was chosen to use the stiffness here, because it relates directly to the measured force signal (see remark below).
A nice consequence of the method is that a hysteresis plot can be made, see Figure 8.15. The dots are modelled data points of the oscillating force on the mass. Normally the elliptic area gives the hysteresis loss (work done by damper), but here it represents energy fed into the system.

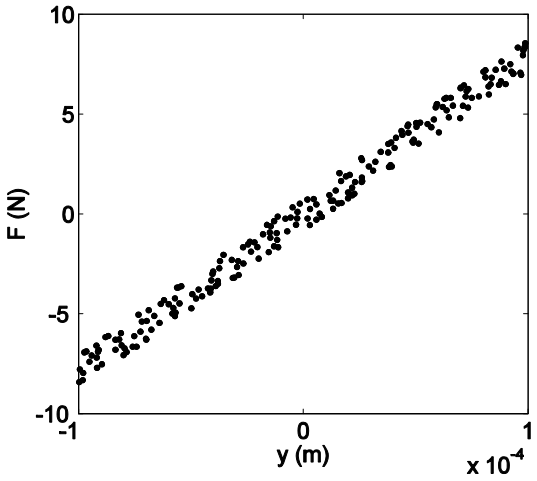


Figure 8.15. A 'deflection versus force' plot (zoomed in) showing modelled data points of the vibration in the previous figure. The area enclosed by the ellipse (no lines were plotted between dots for clarity) equals the energy fed into the system.

Lastly, two remarks about the application of DE. First, some assumed model structures are much harder to compute, for example when a sign($\dot{y}$)-term is included for Coulomb damping.

Typically, a standard run of 80 generations with a population of 35 took roughly 20 minutes. Secondly, evolving coefficients that are indirectly involved in the computation of the fitness was found very tricky. For instance, the initial conditions, or a time-dependent external force term in the motion equation are practically impossible to evolve when the fitness computation uses damping and stiffness forces. Such terms have to be found in a different way – here the solution was to assume unforced self-excitation and find the initial conditions in a pre-run.

8.6 Genetic programming and symbolic regression

In genetic programming (GP), introduced by Koza (1992), the individuals are executable programs represented by tree structures (parse trees) and dedicated to solving complex problems. It can be applied very effectively to symbolic regression (SR), a form of system identification aimed at finding analytical expressions, for example $0.25x^2-\sin(3.6x)$, to describe trends in a numeric dataset. What makes SR so appealing is that it simultaneously strives to find coefficients and model structure with a minimum of presupposed domain knowledge. Symbolic regression has grown from a benchmark for testing new GP techniques to a competitive tool in scientific computation. The general caution in regression analysis and system identification that a choice for a particular non-linear model structure is decisive for performance does not hold for GP: the freedom of the tree representation enables extreme flexibility and moreover contains linear model structures as special cases (e.g. for polynomials, a_1x+a_0 is a special case of $\sum_{i=0}^{n} a_i x^i$). Innumerous authors have contributed to the development of GP in the 1990s and 2000s; Poli et al. (2008) serves as an introduction to literature.
A recurring point of attention in GP is 'bloat', defined as offspring growing in model size without clear benefit in terms of fitness (Poli et al., 2008), can be controlled by somehow including the length of candidate models in the evolution. The application of GP to SR brings additional challenges. To name two major subjects, hitherto without universally applied solutions: determining numerical constants (i.e. coefficients) and dealing with noisy data. It was originally proposed to let numerical constants evolve in the same way as the independent variables. This proved to be an inadequate approach in practice. Among the numerous improvements is the use of differential evolution to find the constants (Cerny et al., 2008). Noise is an indigenous topic in machine learning; it is well known that determining numerical constants from noisier training data increases the ambiguity of model finding since there will be more combinations of structure and constants possible with comparable accuracies (Rogers and Girolami, 2012). Generalization properties of the final model are a key issue, this is known as the bias/variance tradeoff (Bishop, 2006). In GP, the presence of noise tends to promote complexity of the evolved programs and bloat (Zhang and Mühlenbein, 1995; Keijzer and Babovic, 2000). The problem of balancing the search for a fit model with an acceptable complexity (model length, number of terms) is referred to in GP as "accuracy versus parsimony".

The fact that GP-based SR is more than a data regularization technique and exceeds the possibilities of preceding system identification approaches (Conforth and Lipson, 2013) is reflected in the term 'knowledge discovery'. The training data is not just used to infer a numerical model, but to derive usable domain principles in concise mathematical form; all imaginable closed-form equations and ordinary and partial differential equations. This puts

the dichotomy between white-box and black-box modelling on its head. A number of PhD studies have worked on SR by GP, from a computer science perspective (Keijzer 2001 and Kromberger 2011) and from a more applied perspective (Vladislavleva 2008). An elegant demonstration of the power of GP for identifying non-linear dynamical systems automatically and directly from experimental data was given by Schmidt and Lipson (2009) in a seminal double pendulum experiment. A consequence of this work and the related commercial software package Eureqa (Eureqa, 2011) has been an explosion of applications.

The system identification efforts in this chapter were limited to searching for coefficients after fixing the model structure, a so-called parametric approach to regression, according to Vladislavleva (2008). The goal of extending the evolutionary algorithm with GP –not for finding the ODE structure, but for determining the non-linear terms– would naturally be to discover new model structures intangible to analysis from principles (because they are far-fetched, or the involved analysis would be too time-consuming). As part of this study, a GP code was written using gene expression programming (GEP), a GP method introduced by Ferreira (2001), and tested on a number of benchmark regression problems. It was not applied to the experimental data, because this would enable an analysis of a maximum of only two seconds with long computation times. Also, some trials were done with Eureqa (Eureqa, 2011). There was no option here to infer the motion equation from the force signal. So this resulted in rather hard-to-interpret expressions (depending on selected building blocks) of $F(t)$ and $\ddot{F}(t) = f\left(\dot{F}, F, t\right)$, requiring about half an hour to one hour for three seconds of data.

8.7 Conclusions and outlook

In this chapter it was examined how the differential evolution algorithm can be applied to identify several vibration types by performing regression on the coefficients of the motion ODE. Irrespective of the percentage of data used in training, an ODE structure produced more accurate results than an analytical solution structure of a forced vibration, but required more computational time. A number of synthetic self-excited oscillations was identified with reasonable accuracy. The presence of superfluous non-linear terms proved to have an influence on the achieved computation times, but not directly on test errors. Sensitivity analyses exposed the impacts of tolerance settings of the ODE solver and the choice of solver algorithm itself. The search could benefit from spectral information of the output signal to reduce the search space, but (the success of) this process depends on the investigated equation structure of the non-linear terms. The tests showed running times that are at present too high for quick assessment applications in early warning systems. Speed-up can be achieved by faster coding and parallelization.

Two ideas for future possibilities are given. On the one hand GP-based SR is a revolution in how meaningful models are derived from data, on the other hand this automated discovery in no way exempts scientists from making thorough interpretations of the results, as already noted by Schmidt and Lipson (2009).

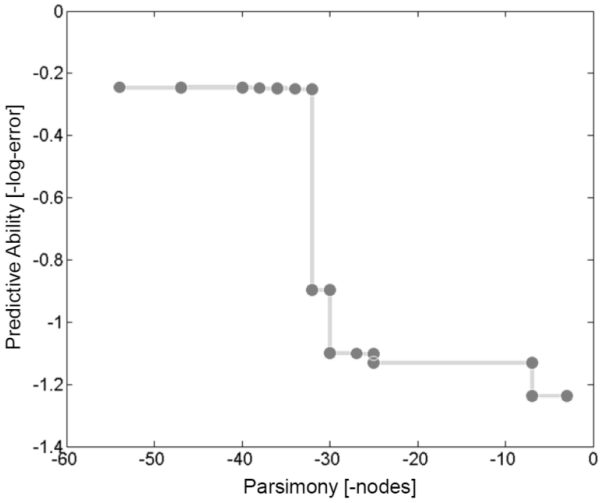


Figure 8.16. The accuracy/complexity front, a plot from Schmidt and Lipson (2009).

The result of an evolutionary search for analytical models is an accuracy/complexity Pareto front. See Figure 8.16; Schmidt and Lipson (2009) refer to this as "predictive ability versus parsimony". They settle this tradeoff by suggesting the model at the inflection point, i.e. right after the accuracy has made a jump and before the front's tail has a steep increase in complexity achieves little extra accuracy. It would be interesting to see if this choice is in fact the best, for a variety of applications, in terms of generalisation and insight. What computational tools can assist in making this choice and does the evolution benefit from this?

A second future question: what is the minimum amount of data (from physical or numerical experiments) necessary for automated derivation of the full system of equations of the fluid-solid interaction, i.e. the Navier-Stokes and structural equations? A first answer would be to feed the system with time series of flow velocity and pressure, at the interface and other strategic locations, that represent diverse hydrodynamic conditions. In particular, the data should sufficiently cover different ranges (flow regimes) of the Reynolds number. Training the model to recognise and describe different Reynolds number flows will be very challenging and rewarding at the same time. Apart from the theoretical impact, this will find applications in turbulence modelling in CFD and e.g. in models for anomaly detection in early warning systems, not unlike the control system of Chapter 7.