Non-Markovian Optimal Prediction with Integro-Differential Equations

Benjamin Seibold
Lawrence Berkeley National Laboratory
1 Cyclotron Road
Berkeley, CA 94720

July 16, 2001

Abstract

We apply the non-Markovian Optimal Prediction method to the Hald system of two coupled oscillators to approximate the mean solution by an integro-differential equation. We derive four versions of integro-differential equations, coming from different approximations to the memory kernel, and compare how well they approximate the mean solution. This question is particularly investigated in dependence on the initial values for one oscillator. An error estimate for the different approximations is carried out. We discuss efficient methods for computing the memory kernel, and we generalize classical Runge-Kutta methods to methods for a class of integro-differential equations which includes the equations that arise in this context.

1 Introduction

The method of optimal prediction of Chorin, Kast, Kupferman [7, 6, 5] is an approach to approximating the average solution of a large system of nonlinear equations, whose inital data is only partly known, by a significantly smaller system. The unknown initial data are assumed to be drawn from a probability distribution. Optimal prediction describes a strategy how to find a low dimensional system, whose solution is close to the components of the average solution corresponding to the known initial conditions. Even for Hamiltonian systems this mean solution will decay [4], and this decay is not described by first order optimal prediction [10]. Chorin, Hald and Kupferman observed in [4] that the average solution is described by the Mori-Zwanzig formalism, which allows no obvious way to be solved directly. While first order optimal prediction is shown to be a very crude approximation to the Mori-Zwanzig formula, several better approximations have been proposed, giving the desired decay. The most important are the "t-model" [2] and the approximation by an integro-differential equation. This paper will focus on the latter approach, which has been described in [3].

We consider the "Hald model", a simple model problem, which has been introduced in [1]: A four dimensional Hamiltonian system of two harmonic oscillators with a nonlinear coupling. The initial conditions of the first oscillator are known and the initial conditions of the second oscillator are drawn from the canonical distribution $\rho(x) = Z^{-1}e^{-\frac{H(x)}{T}}$. We seek the mean position and momentum of the first oscillator, which can be obtained by Monte-Carlo sampling. An important weakness of this approach is that everytime the initial conditions for the first oscillator are changed, the whole expensive computation has to be done all over again. Chorin, Hald and Kupferman describe in [3] how to formulate a two dimensional system of integro-differential equations, which approximates the mean solution. It requires a memory kernel $K(t)$, which has to be approximated by Monte-Carlo sampling. This is as expensive as finding the mean solution directly, but the kernel is independent of the particular initial conditions for the first oscillator, which makes this approach valuable for cases of repeated computations with different inital data. Not to forget cases, in which experience from other fields allows obtaining the memory kernel even less expensively. In section 2 the derivation of the optimal prediction equations for the Hald system will be performed. Nevertheless, most of the statements given in section 2 are also valid for general systems.

In applications the smaller system will still be quite large, thus efficient methods for sampling the memory kernel and solving the system of integro-differential equations are required. Section 3 deals with the computation of the memory kernels and an algorithm for fast Monte-Carlo sampling will be stated. In section 4 we generalize Runge-Kutta methods to numerical methods for a class of integro-differential equations which includes the optimal prediction equations. These methods have a reasonable order of accuracy, good stability behaviour and are comparably cheap.

In section 5 the given methods will be applied to the "Hald model" with a special choice of initial data. Section 6 will especially focus on the quality of the different approximations in dependence on the initial values. In section 7 an attempt to estimate the influence of the different approximations on the error is undertaken.

2 Derivation of the Optimal Prediction Equations for the Hald System

The Hald system is a Hamiltonian system which describes two harmonic oscillators with a nonlinear coupling. It was introduced in [3]. The Hald Hamiltonian is

$$H(x_1, x_2, x_3, x_4) = \frac{1}{2}(x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1^2 x_3^2), \tag{2.1}$$

Here x_1, x_2 are position and momentum of the first oscillator and x_3, x_4 are position and momentum of the second oscillator. The resulting equations of motion are:

$$\frac{d}{dt}\varphi = \frac{d}{dt}\begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \end{pmatrix} = \begin{pmatrix} \varphi_2 \\ -\varphi_1 - \varphi_1\varphi_3^2 \\ \varphi_4 \\ -\varphi_3 - \varphi_3\varphi_1^2 \end{pmatrix} = R(\varphi). \tag{2.2}$$

It is assumed that

$$\hat{\varphi}(0) = \begin{pmatrix} \varphi_1(0) \\ \varphi_2(0) \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \hat{x} \tag{2.3}$$

are known initial conditions, while

$$\tilde{\varphi}(0) = \begin{pmatrix} \varphi_3(0) \\ \varphi_4(0) \end{pmatrix} \tag{2.4}$$

are random.

We assume the existence of a canonical measure on $\mathbb{R}^4$

$$\rho(x_1, x_2, x_3, x_4) = Z^{-1} e^{-\frac{H(x_1,x_2,x_3,x_4)}{T}}. \tag{2.5}$$

In the following we set $T = 1$. By fixing $\hat{\varphi}(0) = \hat{x} = (x_1, x_2)$ we obtain a conditioned measure for $\tilde{\varphi}(0) = (\varphi_3(0), \varphi_4(0))$

$$\begin{aligned} \tilde{\rho}(\varphi_3, \varphi_4) &= \tilde{Z}^{-1} e^{-H(x_1,x_2,\varphi_3,\varphi_4)} = \tilde{Z}^{-1} e^{-\frac{1}{2}(x_1^2+x_2^2+(1+x_1^2)\varphi_3^2+\varphi_4^2)} \\ &= \bar{Z}^{-1} e^{-\frac{1}{2}((1+x_1^2)\varphi_3^2+\varphi_4^2)}. \end{aligned} \tag{2.6}$$

2.1 Projections

As in [3] we define the conditional expectation of a function $f(\hat{x}, \tilde{x}) = f(x_1, x_2, x_3, x_4)$ as

$$\mathbb{E}[f|\hat{x}] = \frac{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x_1, x_2, x_3, x_4) e^{-H(x_1,x_2,x_3,x_4)} dx_3 dx_4}{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{-H(x_1,x_2,x_3,x_4)} dx_3 dx_4}. \tag{2.7}$$

It is the orthogonal projection onto the space of functions $v(\hat{x}) = v(x_1, x_2)$ with respect to the inner product

$$(u, v) = \mathbb{E}[uv] = \frac{\int u(x)v(x)e^{-H(x)}dx}{\int e^{-H(x)}dx}, \tag{2.8}$$

so we denote

$$Pf = \mathbb{E}[f|\hat{x}]. \tag{2.9}$$

Another orthogonal projection in the same space is the finite rank projection

$$P'f = \sum_{i=1}^{2} a_i^{-1}(f, x_i)x_i = a_1^{-1}(f, x_1)x_1 + a_2^{-1}(f, x_2)x_2, \tag{2.10}$$

where $a_i = (x_i, x_i)$.

2.2 Mean Solution

Let $\varphi(x,t)$ denote the solution of (2.2) with initial conditions $\varphi(x,0) = x$. Then the mean solution, which we are interested in, is

$$P\varphi(x,t) = \mathbb{E}[\varphi(x,t)|\hat{x}]. \tag{2.11}$$

The conditional expectation P can be approximated by Monte-Carlo sampling. The mean solution can be calculated as follows:

- Fix $\hat{x} = (x_1, x_2)$
- Sample $\tilde{x} = (x_3, x_4)$ N times from the conditioned distribution given by (2.6)
- Solve N times (2.2) with inital values $(\hat{x}, \tilde{x})$
- Average over all solutions

It is obvious that this is extremely expensive.

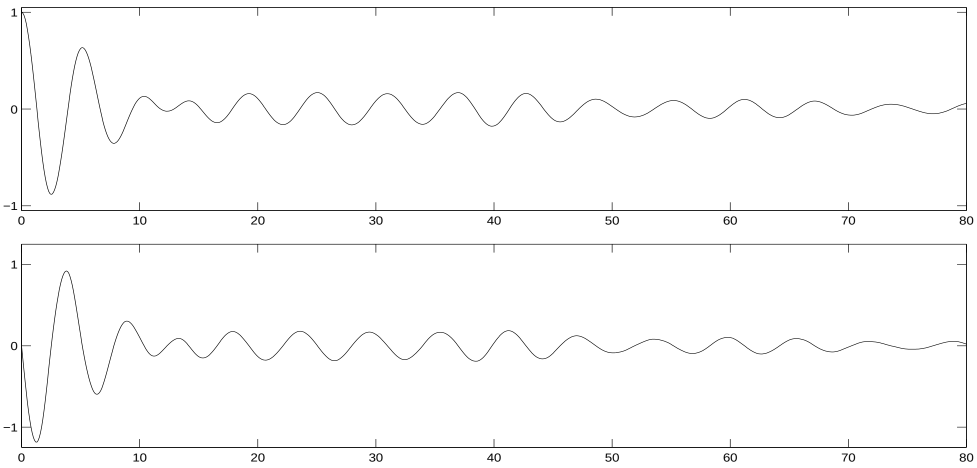


Figure 1: average position and momentum of the first oscillator

We can see in Figure 1 that the mean solution decays, although the Hald system is Hamiltonian. This phenomenon is described in [4]. Note that the interesting behaviour at $t = 12$ is not a numerical artefact but indeed the truth.

2.3 First Order Optimal Prediction

Define $\mathfrak{R} = PR$, then the first order optimal prediction system is given by

$$\frac{d}{dt}\Phi = \frac{d}{dt}\left(\begin{array}{c}\Phi_1\\ \Phi_2\end{array}\right) = \left(\begin{array}{c}\Phi_2\\ -\Phi_1 - \frac{\Phi_1}{1+\Phi_1^2}\end{array}\right) = \left(\begin{array}{c}\mathfrak{R}_1(\Phi_1,\Phi_2)\\ \mathfrak{R}_2(\Phi_1,\Phi_2)\end{array}\right) = \mathfrak{R}(\Phi). \quad (2.12)$$

Hald shows in [8] that the first order optimal prediction system to a Hamiltonian system is again Hamiltonian and therefore a very crude approximation. Figure 2 shows the first component of the decaying mean solution and the first component of the non-decaying solution of the first order optimal prediction equations.

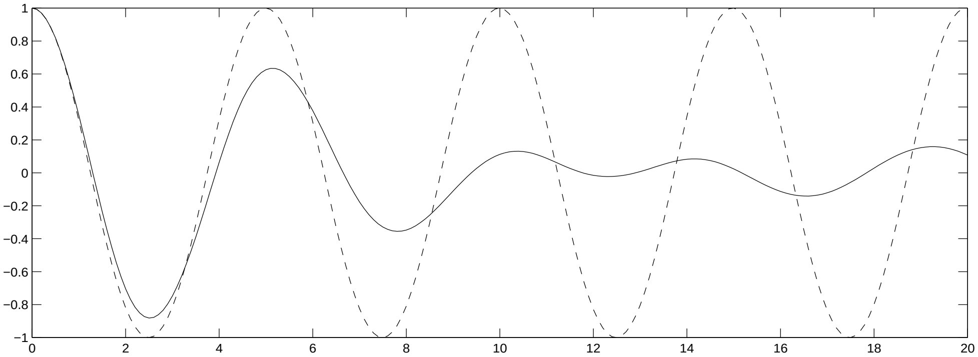


Figure 2: mean solution and first order optimal prediction

2.4 Mori-Zwanzig

Let $L = \sum_{j=1}^{4} R_j(x)\frac{\partial}{\partial x_j}$. Then the Liouville equation

$$u_t(x,t) = Lu(x,t), \quad u(x,0) = x_i \quad (2.13)$$

can be transformed into the Mori-Zwanzig formula [3]

$$\frac{\partial}{\partial t}e^{tL}x_i = e^{tL}\mathfrak{R}_i(x) + \int_0^t e^{(t-s)L}PLe^{sQL}QLx_i ds + e^{tQL}QLx_i, \quad (2.14)$$

which after replacing $\varphi_i(x,t) = e^{tL}x_i$ and applying P yields the identity

$$\frac{\partial}{\partial t}P\varphi_i(x,t) = P\mathfrak{R}_i(\hat{\varphi}(x,t)) + \int_0^t Pe^{(t-s)L}PLe^{sQL}QLx_i ds. \quad (2.15)$$

Now the following approximations are done:

- In the first term interchange P and $\mathfrak{R}_i$
- In the second term replace the second P by P',

which yields

$$\frac{\partial}{\partial t} P\varphi_i(x,t) \approx \mathfrak{R}_i(P\hat{\varphi}(x,t)) + \int_0^t P e^{(t-s)L} P' L e^{sQL} QLx_i ds. \tag{2.16}$$

2.5 The Memory Kernel

The use of P' instead of P allows the following calculation

$$\begin{aligned} P'Le^{sQL}QLx_i &= P'LQe^{sQL}QLx_i = \sum_{j=1}^{2} a_j^{-1}(LQe^{sQL}QLx_i, x_j)x_j \quad (2.17) \\ &= -\sum_{j=1}^{2} a_j^{-1}(e^{sQL}QLx_i, QLx_j)x_j = -\sum_{j=1}^{2} K_{i,j}(s)x_j, \end{aligned}$$

where $K_{i,j}(t) = a_j^{-1}(e^{tQL}QLx_i, QLx_j)$ is the so called "memory kernel". We have used that Q is symmetric and L is skewsymmetric for Hamiltonian systems.

Substituting (2.17) back into (2.16), switching the direction of integration and using $\varphi_j(x,s) = e^{sL}x_j$ yields

$$\frac{\partial}{\partial t} P\varphi_i(x,t) \approx \mathfrak{R}_i(P\hat{\varphi}(x,t)) - \int_0^t \sum_{i=1}^{2} K_{i,j}(t-s) P\varphi_i(x,s) ds. \tag{2.18}$$

Thus we approximate the mean $P\varphi(x,t)$ by the following system of integro-differential equations

$$\dot{y}(t) = \mathfrak{R}(y(t)) - \int_0^t K(t-s) \cdot y(s) ds, \tag{2.19}$$

where

$$y(t) = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \quad \text{and} \quad K(t) = \begin{pmatrix} K_{1,1}(t) & K_{1,2}(t) \\ K_{2,1}(t) & K_{2,2}(t) \end{pmatrix}. \tag{2.20}$$

2.6 Approximating The Memory Kernel

The kernel $K(t)$ is independent of the certain intitial data for $\hat{x}$, so we want to precompute it. Unfortunately there is no obvious way to obtain the orthogonal dynamics semigroup e^{tQL} [12], so one does the third approximation:

- Calculate the kernel $K(t)$ with the original dynamics semigroup e^{tL}

$$K_{i,j}(t) \approx a_j^{-1}(e^{tL}QLx_i, QLx_j) \tag{2.21}$$

A justification for this can be found in [3].

Now the kernel can be calculated using

$$\begin{aligned} e^{tL}QLx_i &= e^{tL}(I-P)Lx_i = e^{tL}Lx_i - e^{tL}PLx_i & (2.22)\\ &= R_i(\varphi(x,t)) - \mathfrak{R}_i(\hat{\varphi}(x,t)) \\ QLx_j &= e^{0L}QLx_j = R_j(\varphi(x,0)) - \mathfrak{R}_j(\hat{\varphi}(x,0)). & (2.23)\end{aligned}$$

For the Hald model one has $R_1(x) = \mathfrak{R}_1(\hat{x})$, so

$$K_{1,1} \equiv K_{1,2} \equiv K_{2,1} \equiv 0. \tag{2.24}$$

The constant $a_2 = (x_2, x_2) = 1$, so

$$K_{2,2}(t) = \mathbb{E}[(R_2(\varphi(x,t)) - \mathfrak{R}_2(\hat{\varphi}(x,t)))\,(R_2(\varphi(x,0)) - \mathfrak{R}_2(\hat{\varphi}(x,0)))] \tag{2.25}$$

Thus we obtain the system of integro-differential equations

$$\begin{aligned} \dot{y}_1(t) &= \mathfrak{R}_1(y_1(t), y_2(t)) & (2.26)\\ \dot{y}_2(t) &= \mathfrak{R}_2(y_1(t), y_2(t)) - \int_0^t K_{2,2}(t-s)y_2(s)ds. \end{aligned}$$

Because of (2.24) we will in the following write $K^{(1)}$ for $K_{2,2}$ given by (2.25). Our system **version 1** is given by

$$\begin{aligned} \dot{y}_1(t) &= y_2(t) & (2.27)\\ \dot{y}_2(t) &= -y_1(t) - \frac{y_1(t)}{1+y_1(t)^2} - \int_0^t K^{(1)}(t-s)y_2(s)ds. \end{aligned}$$

We will introduce some further approximations before describing how to compute the memory kernel.

2.7 Splitting The Kernel

In [3] a further approximation is performed, which can be described as

- "splitting the kernel and freezing one part".

This splitting is motivated by a procedure in physics and although it is a further approximation, it may eventually yield a system which has computational advantages. We start with the kernel given in (2.25) and using the approximations

$$\mathfrak{R}_2(\hat{\varphi}) \approx \mathfrak{R}_2(P\hat{\varphi}) \approx PR_2(\hat{\varphi}) \tag{2.28}$$

we obtain

$$K_{2,2}(t-s) \approx \mathbb{E}[R_2(\varphi(x,t))R_2(\varphi(x,s))] - \mathfrak{R}_2(P\hat{\varphi}(x,t))\mathfrak{R}_2(P\hat{\varphi}(x,s)) \tag{2.29}$$

The first part of (2.29) becomes the new memory kernel

$$K^{(2)}(t) = \mathbb{E}[R_2(\varphi(x,t))R_2(\varphi(x,0))], \tag{2.30}$$

while the second part of (2.29) will be taken into the integro-differential equation using that its solution $y(t)$ should be close to $P\hat{\varphi}(x,t)$. This yields system **version 2**:

$$\begin{aligned} \dot{y}_1(t) &= y_2(t) \\ \dot{y}_2(t) &= -y_1(t) - \frac{y_1(t)}{1+y_1(t)^2} - \int_0^t \left(K^{(2)}(t-s) - f^{(2)}(y_1(t))f^{(2)}(y_1(s))\right) y_2(s)ds, \end{aligned} \tag{2.31}$$

where

$$f^{(2)}(y_1) = f^{(2)}(y_1,y_2) = \mathfrak{R}_2(y_1,y_2) = -y_1 - \frac{y_1}{1+y_1^2}. \tag{2.32}$$

For the Hald system $f^{(2)}$ does not depend on y_2, thus we write $f^{(2)}(y_1)$ instead of $f^{(2)}(y_1,y_2)$. In general $f^{(2)}$ will be a function of $y_1,\ldots,y_m$. A more rigorous motivation for splitting the memory kernel can be found in [16].

2.8 A Different Splitting

The splitting performed in (2.29) is not unique. Instead of splitting

$$R_2(\varphi) - \mathfrak{R}_2(\hat{\varphi}) = -\varphi_1 - \varphi_1\varphi_3^2 + \varphi_1 + \frac{\varphi_1}{1+\varphi_1^2} = -\varphi_1\varphi_3^2 + \frac{\varphi_1}{1+\varphi_1^2} \tag{2.33}$$

into R_2 and $\mathfrak{R}_2$ we now use the same procedure to split into $-\varphi_1\varphi_3^2$ and $-\frac{\varphi_1}{1+\varphi_1^2}$ and obtain system **version 3**:

$$\begin{aligned}
\dot{y}_1(t) &= y_2(t) \qquad (2.34)\\
\dot{y}_2(t) &= -y_1(t) - \frac{y_1(t)}{1+y_1(t)^2} - \int_0^t \left(K^{(3)}(t-s) - f^{(3)}(y_1(t))f^{(3)}(y_1(s))\right) y_2(s)ds,
\end{aligned}$$

where

$$f^{(3)}(y_1) = -\frac{y_1}{1+y_1^2} \tag{2.35}$$

and

$$K^{(3)}(t) = \mathbb{E}[\left(\varphi_1(x,t)\varphi_3(x,t)^2\right)\left(\varphi_1(x,0)\varphi_3(x,0)^2\right)]. \tag{2.36}$$

This yields a function $f^{(3)}(y)$ which is bounded. The importance of this fact will be discussed in section 5.1. Of course there are infinitely many other possible ways of splitting the kernel, and stricly speaking only the choice $f^{(2)}(y_1) = \mathfrak{R}_2(y_1, y_2)$ is in some sense "natural". The splitting perfomed in system version 3 yields an especially easy structure for the Hald system, but for general systems there is no recipe for an "easy" splitting.

2.9 Normalizing the Memory Kernel

For the plot presented in [3] the memory kernel $K^{(3)}$ was divided by the constant $\mathbb{E}[x_1^2 + x_2^2] \approx 1.715$. In physics the procedure of splitting the kernel is followed by this "normalization".

Thus we define

$$K^{(4)} = \frac{1}{\mathbb{E}[x_1^2 + x_2^2]} \cdot K^{(3)} \tag{2.37}$$

The corresponding system is denoted by system **version 4**:

$$\begin{aligned}
\dot{y}_1(t) &= y_2(t) && (2.38)\\
\dot{y}_2(t) &= -y_1(t) - \frac{y_1(t)}{1+y_1(t)^2} - \int_0^t \left(K^{(4)}(t-s) - f^{(4)}(y_1(t))f^{(4)}(y_1(s))\right) y_2(s)ds,
\end{aligned}$$

where

$$f^{(4)}(y_1) = f^{(3)}(y_1) = -\frac{y_1}{1+y_1^2} \qquad (2.39)$$

It should be pointed out, that the assumptions for applying the physical splitting and dividing procedure are not satisfied here.

3 Computation of the Memory Kernels

The three memory kernels given by (2.25), (2.30) and (2.36) are of the type

$$K^{(i)}(t) = \mathbb{E}[g_i(\varphi(x,t))g_i(\varphi(x,0))], \qquad (3.1)$$

where

$$\begin{aligned}
g_1(\varphi_1, \varphi_2, \varphi_3, \varphi_4) &= \frac{\varphi_1}{1+\varphi_1^2} - \varphi_1\varphi_3^2 && (3.2)\\
g_2(\varphi_1, \varphi_2, \varphi_3, \varphi_4) &= -\varphi_1 - \varphi_1\varphi_3^2 && (3.3)\\
g_3(\varphi_1, \varphi_2, \varphi_3, \varphi_4) &= -\varphi_1\varphi_3^2 && (3.4)
\end{aligned}$$

Remembering that $\varphi(x,t)$ is the solution of (2.2) with inital data x, we can compute $K^{(i)}$ by Monte-Carlo sampling:

- Sample N initial values $(x_1, \ldots, x_4)$ from the canonical distribution given by (2.5)
- Solve N times (2.2) on $[0, T]$ with inital values $(x_1, \ldots, x_4)$
- Average over all values $g_i(\varphi(x,t)) \cdot g_i(x)$

Monte-Carlo sampling with 50000 samples yields the following kernel-functions:

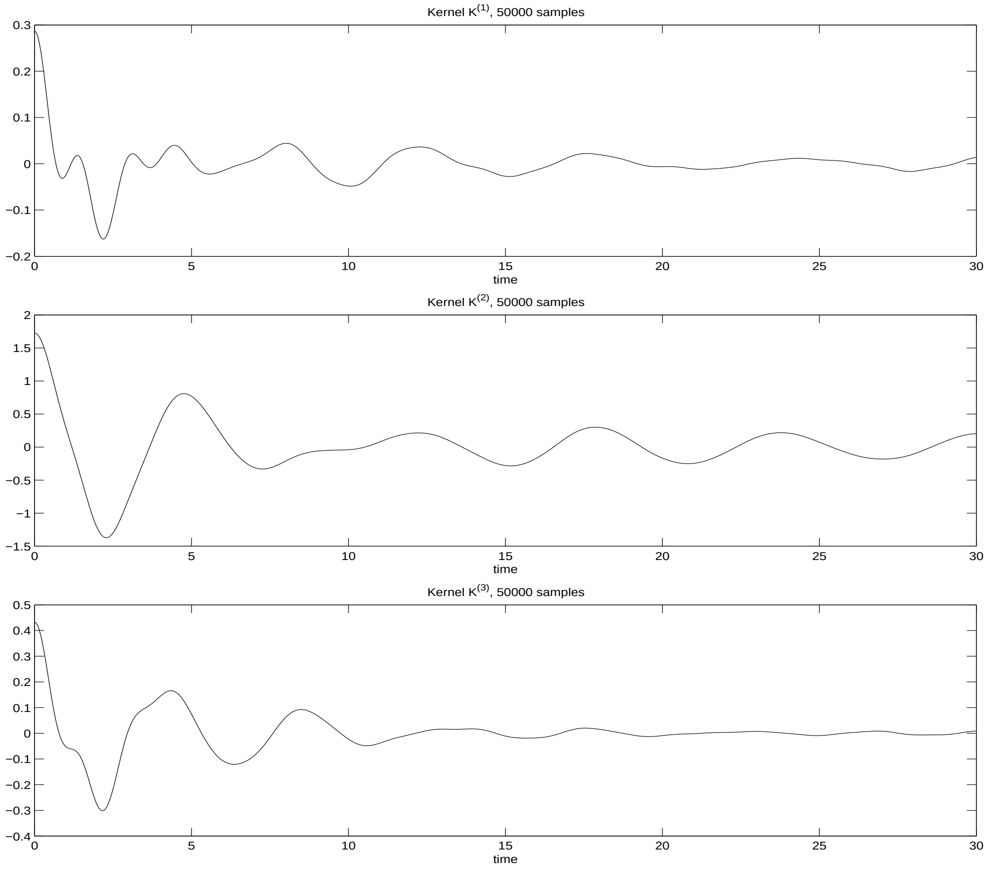


Figure 3: memory kernels $K^{(1)}$,$K^{(2)}$ and $K^{(3)}$

While $K^{(3)}$ is not too far from $K^{(1)}$, $K^{(2)}$ is much larger and is therefore a very bad approximation to $K^{(1)}$. This does not give much hope in the quality of approximation version 2. Note that $K^{(4)}(0) = \frac{1}{1.715}K^{(3)}(0) \approx K^{(1)}(0)$, an interesting effect of the unmotivated normalization. $K^{(3)}$ shows less oscillatory behaviour than $K^{(1)}$, especially for longer times, which may have structural and computational advantages.

3.1 2D-Kernel as Quality Control

The canonical distribution $Z^{-1}e^{-H(x)}$ is invariant under the flow (2.2), therefore

$$\mathbb{E}[g_i(\varphi(x,t))g_i(\varphi(x,s))] = \mathbb{E}[g_i(\varphi(x,t-s))g_i(\varphi(x,0))]. \tag{3.5}$$

This does not mean, that Monte-Carlo sampling with a finite number of samples gives the same values. Computing the kernel as a two dimensional function

$$K^*(t,s) = \mathbb{E}[g_i(\varphi(x,t))g_i(\varphi(x,s))] \tag{3.6}$$

yields a function which is nearly constant along the diagonals $t-s=\sigma$, but not exactly. This observation can be used for two purposes:

- Use the variance of the values along one diagonal as a quality control, respectively as a control for the required number of samples.
- First compute the kernel as a two dimensional function, then set $K(\sigma)$ as the average over all $K^*(t,s)$ with $t-s=\sigma$. This has a point, because solving the system is in general much more expensive than computing the products $g_i(\varphi(x,t))g_i(\varphi(x,s))$.

3.2 Fast Monte-Carlo Sampling

The mean solution and the kernels require the computation of multidimensional integrals of the types

$$\begin{aligned}
\mathbb{E}[\varphi|\hat{x}] &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \varphi(\hat{x},\tilde{x},t)e^{-H(\hat{x},\tilde{x})}d\tilde{x} & (3.7)\\
K^{(i)}(t) &= \mathbb{E}[g_i(\varphi(x,t))g_i(\varphi(x,0))] & (3.8)\\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g_i(\hat{x},\tilde{x})e^{-H(\hat{x},\tilde{x})}d\tilde{x}d\hat{x},
\end{aligned}$$

which can in general only be done approximately, e.g. by Monte-Carlo sampling. The error given by Monte-Carlo sampling decreases like $\frac{1}{\sqrt{N}}$, where N

is the number of samples. Thus already a moderate number of samples yields acceptable results, but its use is very limited, if high accuracy is required.

It has to be pointed out, that in this paper the difference between the mean solution and the various approximations is governed by the several approximations performed in section 2, thus classical Monte-Carlo sampling with about 50000 samples yields a completely sufficient approximation to the mean solution and the kernels.

If the crude approximations in section 2 are improved, an error of order $\frac{1}{\sqrt{N}}$ will not be acceptable anymore. This problem can be overcome by a technique introduced by Chorin in [9], which is based on orthogonal polynomials.

Approximating

$$\int_{-\infty}^{+\infty} g(x)e^{-H(x)}dx \tag{3.9}$$

by Monte-Carlo sampling yields an error of the magnitude of the standard deviation

$$\frac{C(g)}{\sqrt{N}}, \tag{3.10}$$

where

$$C(g) = \sqrt{\frac{1}{\sqrt{\pi}}\int g^2(x)e^{-H(x)}dx - \left(\frac{1}{\sqrt{\pi}}\int g(x)e^{-H(x)}dx\right)^2}. \tag{3.11}$$

Let

$$\{p_0(x), p_1(x), \ldots, p_m(x)\} \tag{3.12}$$

be a set of polynomials of degree $\deg p_m = m$, which are orthonormal with respect to the inner product

$$(u,v) = \int_{-\infty}^{+\infty} u(x)v(x)e^{-H(x)}dx. \tag{3.13}$$

Orthonomality of (3.12) yields

$$\int_{-\infty}^{+\infty} g(x)e^{-H(x)}dx = \int_{-\infty}^{+\infty} \left(g(x) - \sum_{k=1}^{m} b_k p_k(x)\right) e^{-H(x)}dx \tag{3.14}$$

for arbitrary coefficients b_k, because for all $k > 0$

$$\int_{-\infty}^{+\infty} p_k(x) e^{-H(x)} dx = (1, p_k) = 0. \tag{3.15}$$

Choosing the coefficients $b_1, \ldots, b_m$ such that $C\left(g - \sum_{k=1}^m b_k p_k\right)$ is small will dramatically reduce the number of required samples to achieve a given accuracy.

If g allows an expansion in a Hermite series

$$g(x) = \sum_{k=1}^{\infty} a_k p_k(x), \tag{3.16}$$

where

$$a_k = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{+\infty} p_k(x) g(x) e^{-H(x)} dx, \tag{3.17}$$

one can approximate $a_1, \ldots, a_m$ by Monte-Carlo sampling, yielding the values $a_1^*, \ldots, a_m^*$, and set $b_k = a_k^*$.

4 Numerical Methods for the Integro-Differential Equations that Arise in Optimal Prediction

In this section we generalize Runge-Kutta methods to methods for (2.27), (2.31), (2.34) and (2.38) and investigate order of convergence, stability and computational effort. Most ideas can be transferred to other types of methods like e.g. multistep methods, but also for integro-differential equations Runge-Kutta methods bear obvious abvantages as opposed to multistep methods, such as stability and easy starting. Their disadvantage, the larger number of right hand side evaluations, is less important, as we will see. Finally, generalized Runge-Kutta methods qualify for deferred correction methods, if high order accuracy is neccessary.

It should be poined out that great parts of the numerical discussion in this section is valid for much more general problems than the specific problem described in section 2.

The presented integro-differential equations are of the types

$$\dot{y}(t) = \mathfrak{R}(y(t)) + \int_0^t K(t-s)y(s)ds, \quad y(0) = y_0 \tag{4.1}$$

and

$$\dot{y}(t) = \mathfrak{R}(y(t)) + \int_0^t \left(K(t-s) + f(y(t))f(y(s))\right) y(s)ds, \quad y(0) = y_0. \tag{4.2}$$

Both are generalizations of the ordinary differential equation

$$\dot{y}(t) = \mathfrak{R}(y(t)), \quad y(0) = y_0, \tag{4.3}$$

so we want to generalize well-known numerical methods for (4.3) to methods for (4.1) and (4.2), i.e. every method for (4.1) or (4.2) will be a numerical method for (4.3) if $K \equiv f \equiv 0$. The numerical methods will be presented for (4.2), equation (4.1) will then be the special case $f \equiv 0$.

4.1 Constructing Generalized Methods

All methods and their analysis will be presented for the case of equidistant time steps. While the given methods can be generalized to variable stepsizes, their analysis does in general not carry over – a well known issue from ordinary differential equations. We will denote the exact solution of the integro-differential equation by $y(t)$ and the approximate solution on the grid $\Delta = \{t_0, \dots, t_N\} = \{0, h, 2h, \dots, (N-1)h, T\}$ by $u_k = u(hk)$.

The basic idea in generalizing methods is to treat the right hand side of (4.2) in the same way as for ordinary differential equations, with an additional issue that the memory integral has to be approximated numerically.

Let us assume that we have a numerical solution $u_0, \dots, u_n$ at times $t_0, \dots, t_n$ and want to obtain u_{n+1} at time $t_{n+1} = t_n + h$. For the exact solution $y(t)$ the right hand side at t_n is

$$F[t_n, y] = \mathfrak{R}(y(t_n)) + \int_0^{t_n} \left(K(t_n - s) + f(y(t_n))f(y(s))\right) y(s)ds, \tag{4.4}$$

which only makes sense for functions defined on the whole interval $[0, t_n]$. We approximate the integral by an $O(h^q)$-accurate quadrature rule and obtain an approximate right hand side

$$\tilde{F}[t_n, y] = \mathfrak{R}(y(t_n)) + h \sum_{j=0}^{n} w_{n,j} \left(K(t_n - t_j) + f(y(t_n)) f(y(t_j)) \right) y(t_j), \quad (4.5)$$

where $w_{n,j}$ are appropriate weights. This right hand side also makes sense for our numerical solution u, which is only defined at the grid points:

$$\tilde{F}[t_n, u] = \mathfrak{R}(u_n) + h \sum_{j=0}^{n} w_{n,j} \left(K(t_n - t_j) + f(u_n) f(u_j) \right) u_j. \quad (4.6)$$

We are using a q-th order quadrature rule, so smooth functions y satify:

$$\tilde{F}[t_n, y] = F[t_n, y] + O(h^q). \quad (4.7)$$

4.2 Consistency

Suppose now a general one step method for ordinary differential equations

$$u_{n+1} = u_n + h\mathfrak{F}(u_{n+1}, u_n, t_{n+1}, h, \mathfrak{R}), \quad (4.8)$$

which is p-th order accurate for (4.3), i.e.

$$y_{n+1} = y_n + h\mathfrak{F}(y_{n+1}, y_n, t_{n+1}, h, \mathfrak{R}) + O(h^{p+1}), \quad (4.9)$$

is generalized to a method for (4.1) or (4.2)

$$u_{n+1} = u_n + h\mathfrak{F}(u_{n+1}, u_n, t_{n+1}, h, \tilde{F}). \quad (4.10)$$

Substituting the correct solution $y(t)$ of (4.1) respectively (4.2) into (4.10) yields:

$$\begin{aligned} y_{n+1} &= y_n + h\mathfrak{F}(y_{n+1}, y_n, t_{n+1}, h, \tilde{F}) + O(h^{p+1}) \\ &= y_n + h\mathfrak{F}(y_{n+1}, y_n, t_{n+1}, h, F + O(h^q)) + O(h^{p+1}) \\ &= y_n + h\mathfrak{F}(y_{n+1}, y_n, t_{n+1}, h, F) + O(h^{q+1}) + O(h^{p+1}), \end{aligned}$$

which proves the statement:

A p-th order one step method for ordinary differential equations generalized to equations of the type (4.1) respectively (4.2) is of order $\min(p,q)$, *if a q-th order quadrature rule is used and the method is zero-stable.*

It should be pointed out that the order of the quadrature rule is given only by the weights $w_{n,j}$, so a high order quadrature rule has the same cost as a low order rule. Therefore even for a first order method one should not use the rectangle rule $(w_{n,0},\dots,w_{n,n}) = (1,\dots,1,0)$, but the second order trapezoidal rule $(w_{n,0},\dots,w_{n,n}) = (\frac{1}{2},1,\dots,1,\frac{1}{2})$, or the fourth order Simpson's rule $(w_{n,0},\dots,w_{n,n}) = (\frac{1}{3},\frac{4}{3},\frac{2}{3},\frac{4}{3},\dots,\frac{4}{3},\frac{1}{3})$. Although the order of the method cannot be increased, a better quadrature rule may give smaller error constants. Unfortunately, the order of practicable quadrature rules is bounded by the following facts:

- Newton-Cotes rules of higher order than 8 lead to instability caused by negative weights $w_{n,j}$.
- Compound rules of higher order than 3 work only for a certain number of points, but our method requires quadrature rules which work for all possible numbers of points. We will deal with this issue in section 4.6.3.

In the case of variable step sizes the weights will have to be computed online, which is negligible effort for real problems.

4.3 Zero-Stability

A numerical method is called zero-stable, if the numerical solution is Lipschitz with respect to pertubations of the initial values and the right hand side.
It is the condition which guarantees a consistent method to converge.

Although Runge-Kutta methods for (4.3) are always be zero-stable, their generalized version is not automatically zero-stable, because in each step all previous values $u_0,\dots,u_n$ are used. On the other hand it is not a classical multistep method either, because the number of previous values used for each step varies from step to step. Therefore stability analysis of all our methods cannot be just reduced to analyzing roots of polynomials.

Numerical experiments with our generalized methods show zero-stablility for equations (2.27), (2.31) and (2.34). Allow us to state the following assertion, which we will not prove here.

If the kernel K is uniformly bounded on $[0, T]$, i.e. $||K(t, s)|| \leq M$ for all $t, s \leq T$, and f is Lipschitz, then a zero-stable method generalized to (4.1) is zero-stable, if a consistent quadrature rule is used.

4.4 Computational Effort

The memory term in (4.2) requires us to firstly store the whole history of the solution and secondly approximate an integral from 0 to t in every time step. Approximating the memory integral is $O(k)$ work in the k-th step. For N steps this yields a computational effort of $O(N^2)$ for our numerical methods, which can become a problem for long calculations with small time steps. This problem can be remedied, if

$$||K(t, s)|| \to 0 \text{ fairly quickly as } |t - s| \to \infty, \tag{4.11}$$

by setting $K(t, s) = 0$ for $|t - s| > L$ for some reasonable $L > 0$ ("finite memory").

It should be remarked that for our model problem the kernels given in section 3 decay, but keep on oscillating with a relevant amplitude for long times. Numerical experiments show that setting these oscillations to zero can change the solution significantly.

We will measure the cost of our methods in evaluations of $\mathfrak{R}$, K and f. This may seem exagerated for our model problem, especially because K is being precomputed, but in general $K(t_0), \ldots, K(t_n)$ will be large matrices, which may be difficult to compute and store.

4.5 Stiffness

The vague but useful concept of stiffness can be carried over from ordinary differential equations:

A method for (4.2) is called stiff, if explicit methods require unneccessarily small time steps to guarantee stability.

If (4.3) is stiff, in general (4.2) will also be stiff, and even if (4.3) is well behaved, K can make (4.2) stiff. As an example consider (4.1) with $\mathfrak{R} \equiv 0$ and $K = -\alpha^2$. The solution is $y(t) = y_0 \cos(\alpha t)$, so even a constant K can make (4.1) very stiff. Thus implicit methods for (4.2) are important. In the following we will present the trapezoidal rule as an example of an implicit method.

4.6 Generalized Runge-Kutta Methods

We will generalize the explicit and implicit trapezoidal rule as well as the classical Runge-Kutta 4 method to methods for (4.2). All these methods will have the same order of accuracy as for ordinary differential equations, which is not at all trivial for methods using intermediate time steps like Runge-Kutta 4. We assume all our methods to be zero-stable.

4.6.1 Explicit Trapezoidal Rule

We approximate the integral in

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} \left(\mathfrak{R}(y(\sigma)) + \int_0^{\sigma} \left(K(\sigma - s) + f(y(\sigma)) f(y(s)) \right) y(s) ds \right) d\sigma \tag{4.12}$$

by the trapezoidal rule at t_n and t_{n+1} and discretize the memory integrals by an at least second order quadrature rule

$$\begin{aligned} u_{n+1} = u_n + \frac{h}{2} \Bigg(& \mathfrak{R}(u_n) + h \sum_{j=0}^{n} w_{n,j} \left(K(t_n - t_j) + f(u_n) f(u_j) \right) u_j \\ & + \mathfrak{R}(u_{n+1}) + h \sum_{j=0}^{n+1} w_{n+1,j} \left(K(t_{n+1} - t_j) + f(u_{n+1}) f(u_j) \right) u_j \Bigg) . \end{aligned} \tag{4.13}$$

To make sense of u_{n+1} in the right hand side we now perform one explicit Euler predictor step to approximate u_{n+1}. This corresponds to the explicit method with Butcher array $\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$.

An interesting feature is that in each step only one memory sum has to be computed, although two sums appear in (4.13). The reason is that the second sum in the n-th step is basically the first sum in the $(n+1)$-st step, only the last summand can vary. The following algorithm gives an efficient implementation of the explicit trapezoidal rule:

```
$k_{saved} = \mathfrak{R}(u_0)$
$I_2 = \frac{h}{2} f(u_0) u_0$
do
        $I_1 = \frac{h}{2}(K_{n+1}u_0 + 2K_n u_1 + ... + 2K_1 u_n)$
        $k_1 = k_{saved}$
        $\tilde{u} = u_n + h k_1$
        $\tilde{f} = f(\tilde{u})$
        $k_2 = \mathfrak{R}(\tilde{u}) + \left(I_1 + \frac{h}{2} K_0 \tilde{u}\right) + \tilde{f} \cdot \left(I_2 + \frac{h}{2} \tilde{f} \cdot \tilde{u}\right)$
        $u_{n+1} = u_n + h \frac{k_1 + k_2}{2}$
        $f_{n+1} = f(u_{n+1})$
        $k_{saved} = \mathfrak{R}(u_{n+1}) + \left(I_1 + \frac{h}{2} K_0 u_{n+1}\right) + f_{n+1} \cdot \left(I_2 + \frac{h}{2} f_{n+1} u_{n+1}\right)$
        $I_2 = I_2 + h f_{n+1} u_{n+1}$
loop
```

evaluations of	$\mathfrak{R}$	f	K
k-th step	2	2	$k+1$
N steps	$2N$	2N	$\frac{1}{2}(N^2+3N+2)$

As stated before we have $O(N)$ evaluations of $\mathfrak{R}$ and f and $O(N^2)$ evaluations of K.

4.6.2 Implicit Trapezoidal Rule

We start again with (4.13), but now we solve for u_{n+1}, which corresponds to the implicit trapezoidal rule with Butcher array $\begin{array}{c|cc} 0 & & \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$.

Putting everything depending on u_{n+1} to the left hand side yields the following generally nonlinear implicit equation for u_{n+1}

$$Au_{n+1} - \frac{h}{2}\mathfrak{R}(u_{n+1}) - Bf(u_{n+1}) - \frac{h^2}{2}w_{n+1,n+1}f^2(u_{n+1})u_{n+1} = c, \tag{4.14}$$

where

$$A = I - \frac{h^2}{2}w_{n+1,n+1}K(0), \tag{4.15}$$

$$B = \frac{h^2}{2}\sum_{j=0}^{n} w_{n+1,j}f(u_j)u_j, \tag{4.16}$$

$$\begin{aligned} c = u_n + \frac{h}{2}\Bigg(\mathfrak{R}(u_n) + h\sum_{j=0}^{n} w_{n,j}\left(K(t_n - t_j) + f(u_n)f(u_j)\right)u_j \\ + h\sum_{j=0}^{n} w_{n+1,j}K(t_{n+1} - t_j)u_j\Bigg). \end{aligned} \tag{4.17}$$

This equation can be solved by Newton iteration. In the case $f \equiv 0$, i.e. for equation (4.1) it takes an especially simple structure, which differs from ordinary differential equations only by having A instead of I in front of u_{n+1}.

Once the constants have been computed, the iteration itself does not require any evaluations of K anymore. As we have seen before the cost of long time calculations will be governed by evaluations of K, which makes the normally very expensive iteration steps of implicit methods less painful in relation to ordinary differential equations.

4.6.3 Runge-Kutta 4

The classical Runge-Kutta 4 method with Butcher array

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6}
\end{array}$$

can be generalized to a fourth order method for (4.2), but the implementation effort is tremendous compared to the version for ordinary differential equations. The three major problems are as follows:

- To obtain fourth order we have to use at least Simpson's rule for the memory term. The compound Simpson's rule is only defined for an odd number of points, but the number of points in the memory sum takes every value from 1 to N. Using a four-point Newton-Cotes rule for the last points, if the number of points is even, will preserve fourth order. While this fix works for Simpson's rule, numerical experiments show that it fails for higher order integration rules like compound Boole's rule, which leaves Simpson's rule as the highest order quadrature rule that works for our numerical methods.

- In the 2^{nd} and 3^{rd} stage steps with length $\frac{h}{2}$ are performed. This yields values at $t_0, t_1, \ldots, t_n, t_n + \frac{h}{2}$, which is nonequidistant. If our grid is equidistant, we can use the nonequidistant three-point rule $(\frac{3}{8}, \frac{9}{8}, 0)$, respectively four-point rule $\frac{5}{192}(13, 50, 25, 8)$ for the last points. In the case of nonequidistant time steps we will have to calculate weights anyway, which makes this a minor issue. It should be pointed out that using halfsteps for the approximation of the memory integral will destroy the fourth order accuracy, because halfsteps are not fourth order accurate.

- In the first two steps there may be just two values available for the memory integral, which therefore can only be approximated by the trapezoidal rule. Still our method will be fourth order, because approximating an interval of length h, respectively $\frac{h}{2}$ by the trapezoidal rule yields an error of $O(h^3)$ in the right hand side. Therefore the local error in this one step is $O(h^4)$ instead of $O(h^5)$, which maintains the fourth order, because it is just one step.

As with the trapezoidal rule one can save half of the evaluations of K, because the two halfsteps use the same history and the last stage corresponds to the first stage in the next step.

4.7 Conclusions

The presented generalized Runge-Kutta 4 method works very well for the optimal prediction equations in our model problem. For more complex applications the equations that arise are expected to be stiff, which makes the implicit trapezoidal rule a good method. All methods become easier if $f \equiv 0$. While explicit methods for (4.2) are just more complicated to implement, for implicit methods an $f \not\equiv 0$ may yield an additional nonlinear term. Thus just from a computational point of view splitting the kernel should be avoided.

The presented way of generalizing numerical methods could not be successfully applied to achieve methods of order higher than four, because of the previously descibed problems with high order quadrature rules. For a higher order approach deferred correction methods will be preferable, for which the presented methods qualify as the required low order methods.

5 Numerical Results for Fixed Initial Data

In this section we solve the four integro-differential equations (2.27), (2.31), (2.34) and (2.38) for inital data $\hat{x} = (x_1, x_2) = (1, 0)$ and compare the results with the mean solution, which is being computed by Monte-Carlo sampling as described in section 2.2. The memory kernels $K^{(1)}, \ldots, K^{(4)}$ are precomputed by Monte-Carlo sampling as described in section 3. They will be used for all following computations with different initial data. For all Monte-Carlo computations 50000 samples were used. Checks with different numbers of samples show that the error produced by Monte-Carlo sampling is smaller than the resolution of the plots.

All integro-differential equations were solved numerically with the generalized Runge-Kutta 4 method presented in section 4.6.3 with constant step size $h = 0.1$. To provide the neccessary values of $K^{(i)}$ at the halfsteps the kernels were computed by the classical Runge-Kutta 4 method with step size $h = 0.05$. The mean solution was integrated by the classical Runge-Kutta 4 method with step size $h = 0.1$. Checks with smaller step sizes show that the error produced by numerically integrating the equations is smaller than the resolution of the plots.

The computations yield the following results:

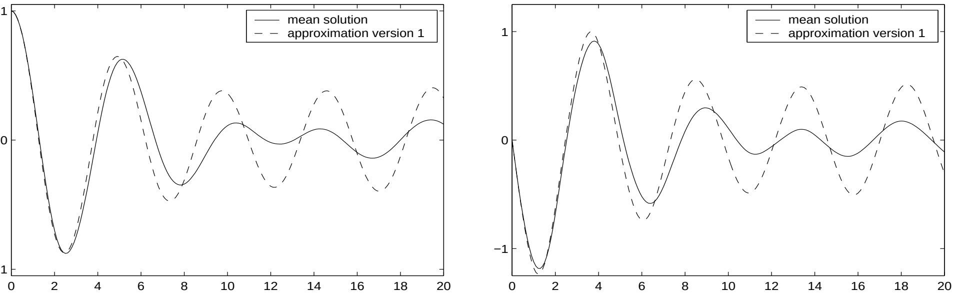


Figure 4: mean solution and approximation version 1

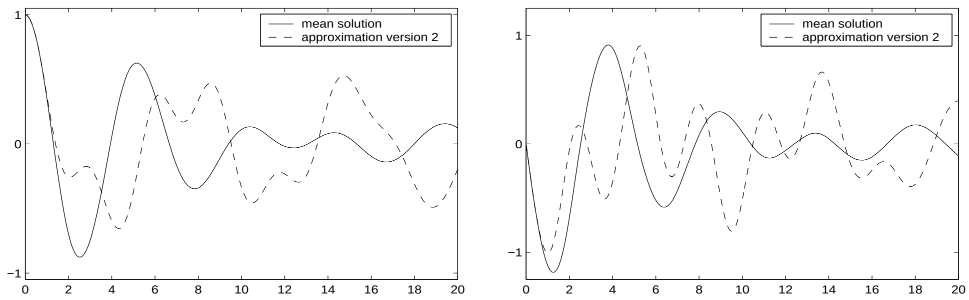


Figure 5: mean solution and approximation version 2

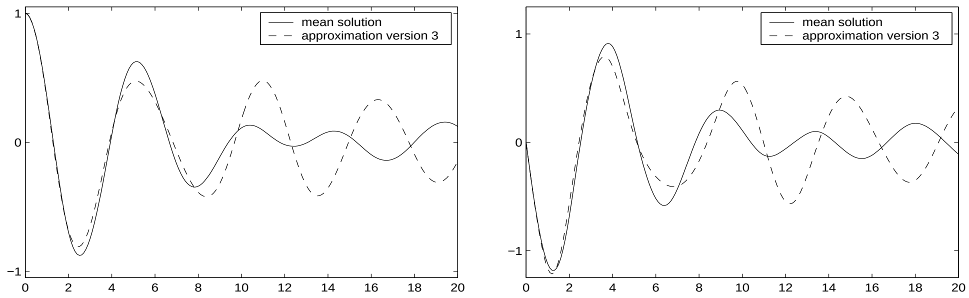


Figure 6: mean solution and approximation version 3

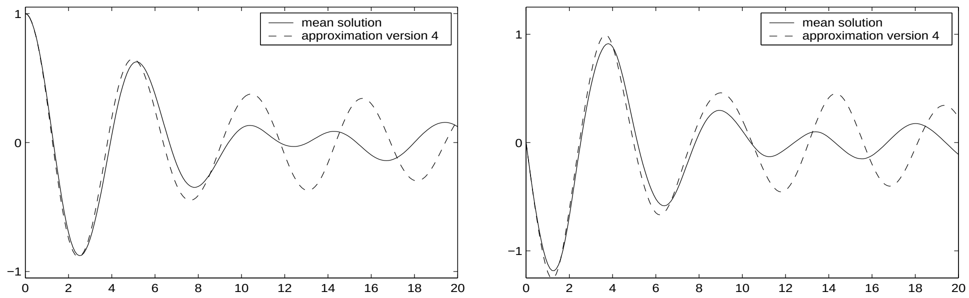


Figure 7: mean solution and approximation version 4

5.1 Observations and Interpretation

None of the approximations gives overwhelming results, but obviously approximation 1,3 and 4 look similar to the mean solution, while approximation 2 shows a totally different behaviour. Systems 2,3 and 4 differ from system 1 in that way, that one further approximation in the derivation of the integro-differential equations is performed. Thus the apparent discrepancy in the behaviour must result from this fact. An explanation why system 2 gives such different results than systems 3 and 4 is given by the following facts:

- $K^{(3)}$ and $K^{(4)}$ are about as large as $K^{(1)}$, while $K^{(2)}$ has much larger values (see section 3).
- $f^{(3)} = f^{(4)} = -\frac{y_1}{1+y_1^2}$ is bounded by $\frac{1}{2}$, thus the products $f(y_1(t))f(y_1(s))$ in (2.34) and (2.38) are always bounded by $\frac{1}{4}$ and become negligibly small for values $y_1 \ll 1$ or $y_1 \gg 1$. On the other hand $f^{(2)} = -y_1 - \frac{y_1}{1+y_1^2}$ is unbounded and therefore the right hand side of (2.31) is not even Lipschitz. Indeed, Okunev shows in [16] that system version 2 can produce a blow up for large initial data.

This leads us to the conclusion that the procedure of splitting the memory kernel is only reasonable, if the new kernel K is not too far away from the old kernel and the function f is bounded and not too large. For this reason we will from now on not focus on system version 2 anymore.

Approximations version 1,3 and 4 follow the mean solution quite well up to $t = 10$, then decay too slowly or fall out of phase. One reason for this fact is that the mean solution shows very irregular behaviour at $t = 12$, which an integro-differential equation of type (2.27) or (2.34) is unlikely to reproduce. Up to $t = 6$ systems 1 and 4 produce pretty good approximations to the mean solution, while system 3 shows too strong decay. It seems as if the "normalization" of the kernel gives exactly the correct rate of decay for short times.

The question, if the procedure of splitting the kernel has advantages, cannot be answered yet. On the one hand system 4 yields a slightly better approximation than system 1. On the other hand system 2 gives horrible results. In section 6 we will investigate if these observations are valid for general initial data.

6 Scaling of the Mean Solution and Approximations

In this section we will focus on qualitative behaviour of the mean solution and the approximations, if the initial values for the first oscillator x_1, x_2 are being scaled.

We will consider

- the mean solution $\mathbb{E}[\varphi(x,t)|\hat{x}]$

and the approximations

- first order optimal prediction $\dot{\Phi} = \mathfrak{R}(\Phi)$
- the t-model
$$\dot{\Phi}(t) = \mathfrak{R}(\Phi(t)) + \mathfrak{S}(t, \Phi(t)), \tag{6.1}$$
where
$$\begin{pmatrix} \mathfrak{S}_1(t, \Phi_1, \Phi_2) \\ \mathfrak{S}_2(t, \Phi_1, \Phi_2) \end{pmatrix} = \begin{pmatrix} 0 \\ -t\frac{\Phi_1^2\Phi_2}{(1+\Phi_1^2)^2} \end{pmatrix} \tag{6.2}$$
The derivation of (6.1) is given in [2].
- the integro-differential equation version 1
$$\dot{y}(t) = \mathfrak{R}(y(t)) - \int_0^t K^{(1)}(t-s) \cdot y(s)ds \tag{6.3}$$
- the integro-differential equation version 3
$$\dot{y}(t) = \mathfrak{R}(y(t)) - \int_0^t \big(K^{(3)}(t-s) - f(y(t))f(y(s))\big) \cdot y(s)ds \tag{6.4}$$
- the integro-differential equation version 4, i.e. with the renormalized kernel.

As seen before, the mean solution is a decaying oscillation, first order optimal prediction oscillates without decay, and both the t-model and the integro-differential equations yield decaying oscillations.

We now consider initial values

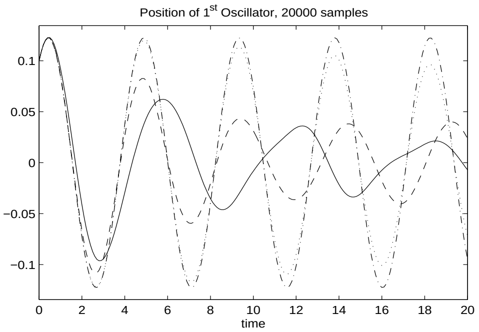
$$\begin{pmatrix} \varphi_1(0) \\ \varphi_2(0) \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \alpha \in (0, \infty) \tag{6.5}$$

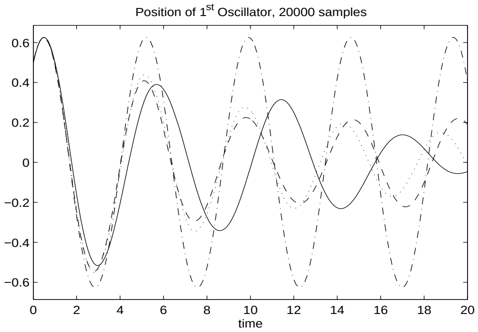
and ask, how period and decay of the mean solution change under scaling the initial conditions and how well the different approximations reflect this behaviour. Such rather qualitative analysis may say more about the quality of the approximations than comparing the results for a certain choice of initial values.

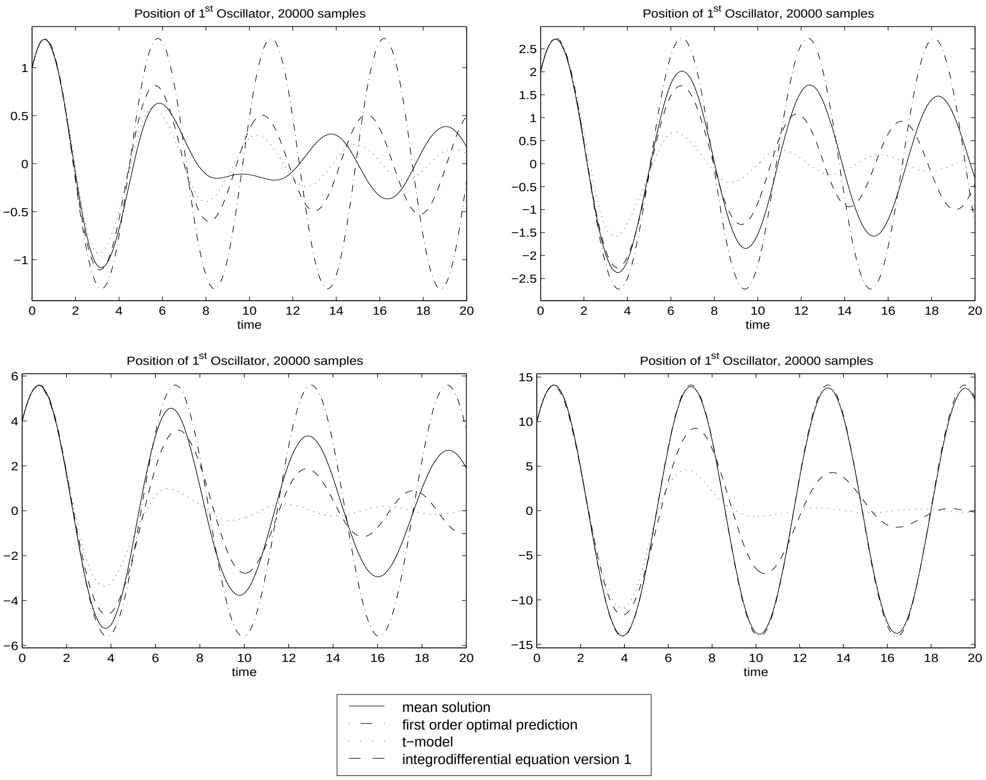
6.1 Numerical Results

The following graphs show the first component (position) of the mean solution and its approximations for initial values $\alpha \in \{0.1, 0.5, 1, 2, 4, 10\}$. The second components (momentum) show the same qualitative behaviour.

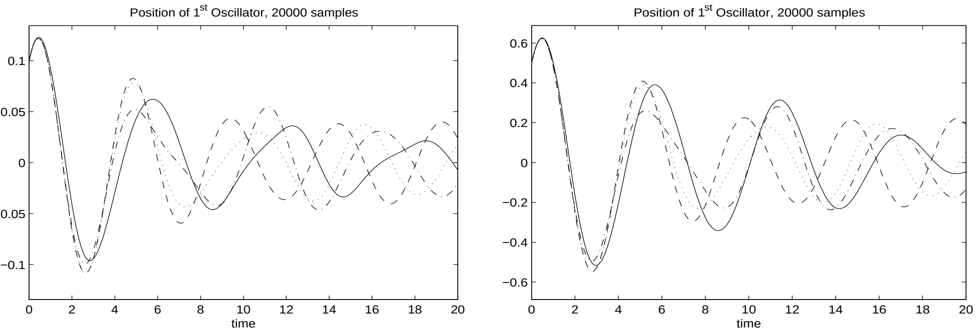
The first set of graphs compares first order optimal prediction, the t-model and the integro-differential equation version 1 with the mean solution.

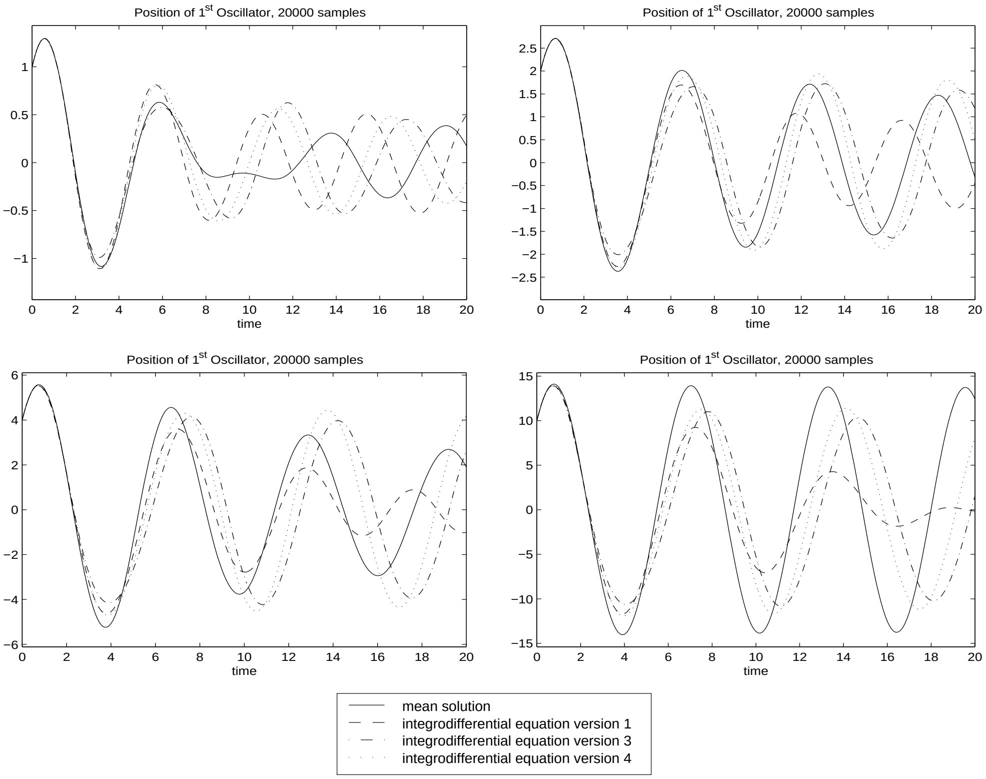






The second set of graphs compares the integro-differential equations version 1, 3 and 4 with the mean solution.





Allow us to first list the results in the following table:

	period		decay		
	α small	α large	α small	α medium	α large
mean solution	less than 2π	2π	fast	medium	slow
1st order OP	$\sqrt{2}\pi$	2π		none	
t-model	$\approx$ like 1st order OP		slow	fast	slow
IDE 1	$\approx$ like 1st order OP		fast	fast	fast
IDE 3	longer than 1st order OP		fast	medium	slow
IDE 4	between IDE 1 and IDE 3		fast	medium	slow

6.2 Arguments for Some Observations

For most of the observations listed above heuristic explanations can be given. Some are rigorous arguments, others are based on experience and may be consindered more “handwaving”. Still they give some insight into the behaviour of the different approximations, thus we do not omit them.

- *The period of the mean solution is about 2π for large α and smaller for small α:*

  The mean solution is not periodic, so we define by “period” heuristically the distance between two consecutive maxima with a value which is not too small. The first and the second maximum of the first components e.g. have the distance 2π for large α and about 5.5 for small α. From (2.2) it follows that

  $$\ddot{\varphi}_1 = -(1 + \varphi_3(t)^2)\varphi_1 \tag{6.6}$$

  The comparison theorem [17] yields, that the period of $\varphi_1(t)$ is always less than or equal 2π, and it will be the smaller the larger $\varphi_3(t)$ is. For short times the value of $\varphi_3(t)$ will be governed by the value of the initial values x_3, x_4, which are sampled from the distribution

  $$\rho(x_3, x_4) = Z^{-1} \exp\left(-\frac{1}{2}\left((1+\alpha^2)x_3^2 + x_4^2\right)\right). \tag{6.7}$$

  For large α the variance of f is much smaller than for small α making shorter periods less important. Thus for large α the mean solution’s period is about 2π, while our experiments show, that for small α it is shorter, but still longer than the approximations’ period.

- *The mean solution decays the slower the larger α is:*

  The rate of decay is given by the fact how quickly all possible solutions with the same x_1, x_2 depart from each other, which is like the period governed by the variance of the initial values x_3, x_4. Thus large α yield a much smaller rate of decay.

- *First order optimal prediction oscillates faster for small α:*

  The equations of motion

  $$\begin{pmatrix} \dot{\Phi}_1 \\ \dot{\Phi}_2 \end{pmatrix} = \begin{pmatrix} \Phi_2 \\ -\Phi_1 - \frac{\Phi_1}{1+\Phi_1^2} \end{pmatrix} \tag{6.8}$$

become asymptotically equal to $\begin{pmatrix} \dot{\Phi}_1 \\ \dot{\Phi}_2 \end{pmatrix} = \begin{pmatrix} \Phi_2 \\ -2\Phi_1 \end{pmatrix}$ for α (and therefore Φ_1) small and $\begin{pmatrix} \dot{\Phi}_1 \\ \dot{\Phi}_2 \end{pmatrix} = \begin{pmatrix} \Phi_2 \\ -\Phi_1 \end{pmatrix}$ for α large. Thus first order optimal prediction has a period of about $\sqrt{2}\pi$ for small α and about 2π for large α.

- *The t-model has about the same period as first order optimal prediction:* Equation (6.1) is basically first order optimal prediction with an additional term in the right hand side, which produces decay. Unless the decay is too fast (as e.g. for $\alpha \approx 2$), this "damping" term is small and hence does not influence the period much.

- *The t-model decays fast for moderate values of α, otherwise slowly:* Figure 10 shows the solution of the t-model for the cases $\alpha \in \{0.1, 2, 1000\}$. The t-model produces a slow and "smooth" decay for small α, a very fast decay for moderate α, and for large α the solution does at first decay slowly and then quickly decays to 0.

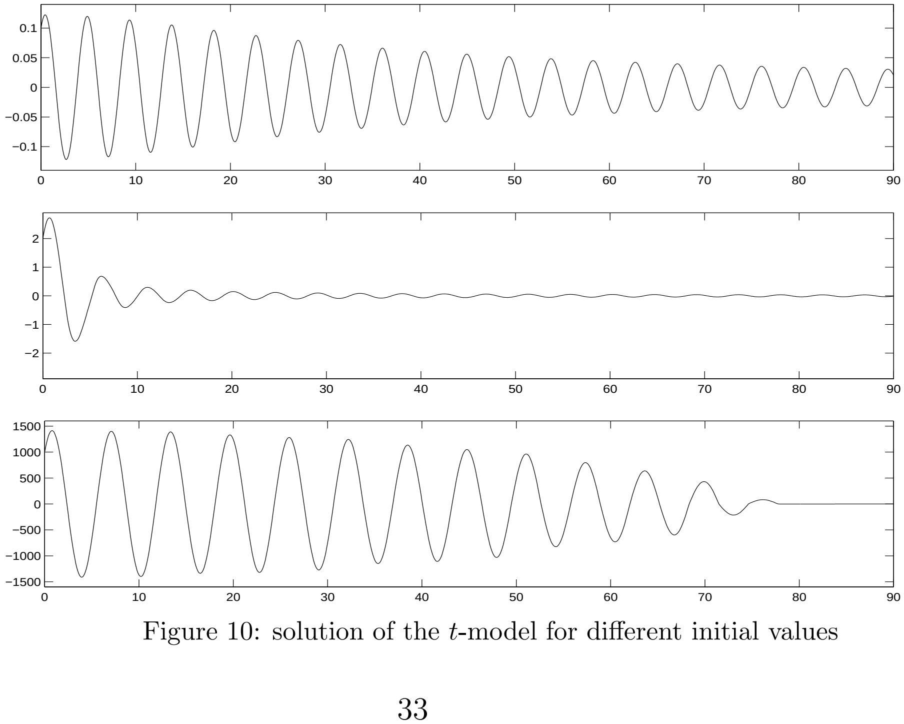


Figure 10: solution of the t-model for different initial values

This behaviour can be explained by the function $\mathfrak{S}(t,\Phi)$, which for every fixed t vanishes for $\Phi \to 0$ as well as $\Phi \to \infty$, but has a value for moderate values of Φ.

6.3 Scaling of the Integro-Differential Equations

The effect of the convolution term in the right hand side of (2.27) and (2.34) is much more complicated than the influence of $\mathfrak{S}$ in the t-model.

To be able to compare the equations let us state that just setting $f \equiv 0$ in (2.34) produces a visible difference in the results only for $\alpha \approx 1$. Thus approximations version 3 and 4 can basically be obtained by solving (2.27) with the kernels $K^{(3)}$ respectively $K^{(4)}$. This observation is based on numerical experiments, supported by the fact that $f(y)$ vanishes for $y \to 0$ and $y \to \infty$.

Although $K^{(3)}$ and $K^{(4)}$ look very similar to $K^{(1)}$, surpisingly the corresponding solutions show a very different behaviour:

- While approximation 1 has nearly the same period as first order optimal prediction, approximation 3 oscillates slower, approximating the mean solution much better for small α. The period of approximation 4 is slightly shorter than the period of approximation 3, which was to expect, because the kernel is smaller, i.e. closer to first order optimal prediction. For large α on the other hand approximation 1 has the same period as the mean solution, while approximations 3 and 4 oscillate too slowly.

- More surpising is the rate of decay. While the rate of decay of approximation 1 is almost independent of α and therefore much too strong for large α, approximations 3 and 4 decay slower for large α, i.e. they reflect the behaviour of the mean solution — a very interesting effect of the splitting.

6.4 Conclusions

The mean solution oscillates and decays the slower the larger the initial values are. Qualitatively all approximations reflect this behaviour for the period. The approximations that arise from splitting the kernel also show the correct behaviour for the rate of decay, while the other approximations fail to do so.

For small initial values the integro-differential equations are the closest approximations to the mean solution. Among them approximation 3 captures the period the best.

An almost perfect approximation for large initial values is first order optimal prediction, all other approximations decay too fast. Among the integro-differential equations version 1 has the correct period, but decays much too fast, while approximations 3 and 4 decay much slower, but their period is too large.

7 An Estimate of the Errors Produced by the Different Approximations

As derived in section 2.4 the following identity is valid

$$\frac{\partial}{\partial t} P\varphi_i(x,t) = P\Re_i(\hat{\varphi}(x,t)) + \int_0^t Pe^{(t-s)L}PLe^{sQL}QLx_i ds, \tag{7.1}$$

where $P\varphi_i(x,t)$ is the i-th component of the mean solution. The first order optimal prediction equations can be obtained by performing the following approximations:

- Drop the memory term $\int_0^t Pe^{(t-s)L}PLe^{sQL}QLx_i ds$
- Interchange P and $\Re_i$

Let $\Phi_i(t)$ denote the i-th component of the solution of the first order optimal prediction equations

$$\dot{\Phi}(t) = \Re(\Phi(t)), \quad \Phi(0) = \hat{x} \tag{7.2}$$

To estimate the error made by first order optimal prediction

$$E_i(t) = P\varphi_i(x,t) - \Phi_i(t) \tag{7.3}$$

we introduce the following functions:

$$\psi_i(t) \;=\; \hat{x} + \int_0^t P\mathfrak{R}_i\left(\hat{\varphi}(x,s)\right) ds \tag{7.4}$$

$$\zeta_i(t) \;=\; \hat{x} + \int_0^t \mathfrak{R}_i\left(P\hat{\varphi}(x,s)\right) ds \tag{7.5}$$

Both ψ_i and ζ_i can be approximated by Monte-Carlo sampling. ζ_i is obtained by applying $\mathfrak{R}_i$ to the mean solution and approximating the integral by the trapezidal rule. ψ_i is approximated as follows:

- Fix $\hat{x} = (x_1, x_2)$ and sample $\tilde{x} = (x_3, x_4)$ N times from the conditioned distribution given by (2.6)
- Solve N times (2.2) with inital values $(\hat{x}, \tilde{x})$
- Apply $\mathfrak{R}_i$ to each of the N solutions, then average
- Approximate the integral by the trapezoidal rule

The error E_i can then be represented as

$$\begin{aligned} E_i(t) &= P\varphi_i(x,t) - \Phi_i(t) \qquad (7.6)\\ &= (P\varphi_i(x,t) - \psi_i(t)) + (\psi_i(t) - \zeta_i(t)) + (\zeta_i(t) - \Phi_i(t)) \end{aligned}$$

The three terms have the following interpretations:

- $P\varphi_i(x,t) - \psi_i(t)$ Dropping the memory term
- $\psi_i(t) - \zeta_i(t)$ Interchanging P and $\mathfrak{R}_i$
- $\zeta_i(t) - \Phi_i(t)$ Going over to an ordinary differential equation

For the Hald oscillator the second components of the functions above are of interest, because $\mathfrak{R}_2$ is nonlinear. Figure 11 shows numerical approximations to the functions $P\varphi_2(x,t)$,$\psi_2(t)$,$\zeta_2(t)$ and $\Phi_2(t)$.

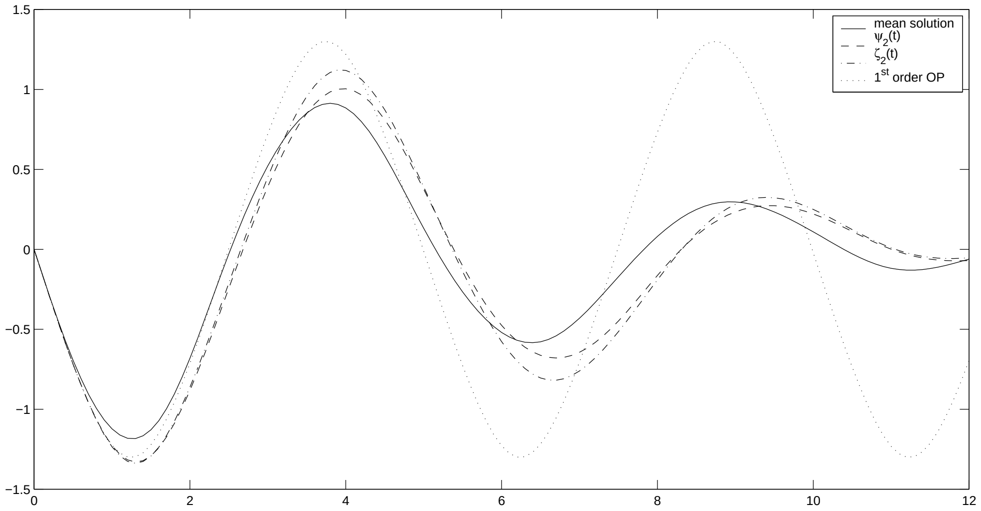


Figure 11: mean solution, first order optimal prediction and functions “in between”

One can observe that neither dropping the memory term nor interchanging P and $\mathfrak{R}_i$ produce large errors, while going over to an ordinary differential equation changes the function significantly. In other words, none of the approximations yields a large error in one step, but putting a slightly incorrect value back into the equations right hand side lets errors build up.

In comparison of the two approximations interchanging P and $\mathfrak{R}_i$ yields a smaller error than dropping the memory term. On the other hand the latter approximation has been significantly improved by non-Markovian optimal prediction by integro-differential equations or the t-model, giving rise to the assertion that exacly the interchange of P and $\mathfrak{R}_i$ is the step which prevents better results. A similar investigation for non-Markovian optimal prediction could clarify this issue.

8 Conclusions

We have described how to apply the non-Markovian optimal prediction theory efficiently to Hamiltonian systems, including improvements in the computation of the memory kernels and generalizing Runge-Kutta methods to numerical methods for the integro-differential equations that arise. We applied the stated methods to the Hald oscillator as a simple model problem and performed a comparison of the different versions of non-Markovian optimal prediction.

None of the described approximations gave overwhelming results, although some approximations gave comparably good resuls for special choices of initial values. Surpizingly an approximation based on splitting and normalizing the kernel often gave better results than the version which calculates the kernel directly. This was to observe especially for large initial values. On the other hand an other version of splitting gave horrible results and could even produce a blow up. Because there is no reason to prefer a priori one splitting to an other, we draw the conclusion that splitting the kernel is a procedure which should be avoided.

Our attempt to get an impression about the importance of the several approximations has circumstantiated the assertion that a way has to be found to avoid interchanging the right hand side of the first order optimal prediction equations with the conditional expectation projection.

9 Acknowledgements

We would like to thank Prof. A.J. Chorin and Prof. O.H. Hald for useful advice and encouragement and Ms. T. Chow, Mr. E. Ingerman, Mr. K. Lin, Mr. P. Okunev and Ms. H. Shvets for helpful discussions and comments.

References

[1] J. Bell, A.J. Chorin and W. Crutchfield, Stochastic optimal prediction with application to averaged Euler equations, Proc. 7th Nat. Conf. Comput. Fluid Mech., C.A. Lin (ed), Pingtung, Taiwan, (2000), pp. 1-13.

[2] A.J. Chorin, O. Hald and R. Kupferman, Optimal prediction with memory, submitted for publication, 2001.

[3] A.J. Chorin, O. Hald and R. Kupferman, Non-markovian optimal prediction, Monte Carlo Methods and Applications, Vol. 7, No. 1-2, pp. 99-109 (2001).

[4] A.J. Chorin, O. Hald and R. Kupferman, Optimal prediction and the Mori-Zwanzig representation of irreversible processes, Proc. Nat. Acad. Sc. USA, 97, (2000), pp. 2968-2973.

[5] A.J. Chorin, A. Kast and R. Kupferman, On the prediction of large-scale dynamics using unresolved computations, Contemp. Math., 238, (1999), pp. 53-75.

[6] A.J. Chorin, A. Kast and R. Kupferman, Unresolved computation and optimal prediction, Comm. Pure Appl. Math., 52, (1999), pp. 1231-1254.

[7] A.J. Chorin, A. Kast and R. Kupferman, Optimal prediction of under-resolved dynamics, Proc. Nat. Acad. Sc. USA, 95 (1998), pp. 4094-4098.

[8] A.J. Chorin, Probability, Mechanics, and Irreversibility, Lecture notes, UC Berkeley Math. Dept., 2000.

[9] A.J. Chorin, Hermite expansions in Monte-Carlo computation, J. Comp. Phys. 8, (1971), pp. 171-186.

[10] A.J. Chorin, R. Kupferman and D. Levy, Optimal prediction for Hamiltonian partial differential equations, J. Comput. Phys., 162, (2000), pp. 267-297.

[11] O. Hald, Optimal prediction and the Klein-Gordon equation, Proc. Nat. Acad. Sc. USA, 96, (1999), pp. 4774-4779.

[12] O. Hald and R. Kupferman, Existence of orthogonal dynamics, preprint, 2001.

[13] O. Hald and R. Kupferman, Convergence of optimal prediction for nonlinear Hamiltonian systems, submitted for publication, 2000.

[14] A. Kast, Optimal prediction of stiff oscillatory mechanics, Proc. Nat. Acad. Sci. USA, 97, (2000), in press.

[15] P. Linz, Analytical and numerical methods for Volterra equations, Philadelphia, SIAM (1985).

[16] P. Okunev, On comparative performance of three different algorithms for the non-markovian optimal prediction applied to the Hald system, LBNL-Report, 2001.

[17] V.V. Stepanov, A course of differential equations, Leningrad, State puplishing house for physical and mathematical literature (1959).