Abstract

The polynomial chaos approach for stochastic simulation is applied to trajectory optimization, by conceptually replacing random variables with free variables. Using the gradient method, we generate with low computational cost an accurate parametrization of optimal trajectories.

*Keywords*: Polynomial chaos, Trajectory optimization.
1. Introduction

Trajectory optimization for dynamic systems has a long and rich history, with many successful applications in diverse areas such as economics, process control, and flight mechanics. Among persistent research topics in the field are optimization for stochastic systems and computational methods (e.g., Carrasco & Banga, 1997; Banga, Alonso & Singh, 1997; Chen, Li & Zhou, 1998; Kohlmann & Zhou, 2000; Chen & Yong, 2001; Sampalli, Papageorgiou & Shah, 1998; Darlington, Pantelides, Rustem & Taniy, 2000). Here, we have extended the classical Maximum Principle to a stochastic domain, although the problem solved is not stochastic optimization in the sense of the references above: We achieve a parametrization of optimal paths with a small number of free variables. These variables may lie in the plant description, in the initial conditions, and in the cost function.

The primary tools to obtain this parametrization are Wiener’s polynomial chaos (Wiener, 1938), and the Cameron-Martin theorem (Cameron & Martin, 1947), which show that a random variable can be accurately approximated with a finite series of orthogonal polynomials in an underlying set of random variables. Ghanem and Spanos were the first to apply these expansions in solving stochastic problems computationally, and in the last decade, the polynomial chaos has been used extensively for structural and fluid mechanics (Ghanem & Spanos, 1991; Xiu, Lucor, Su & Karniadakis, 2002). The use of various distributions is made with the Askey scheme, discussed by Xiu & Karniadakis (2002); we focus on the Legendre chaos here, which correlates with the uniform distribution and is most natural for free parameters taken with equal probability between fixed bounds. Although any suitable distribution and its corresponding orthogonal polynomial family can be used for a given problem, the lowest computational cost is often achieved by choosing the one closest to the actual problem.

We explicitly take advantage of the fact that the polynomial chaos approach provides not only statistics, but in fact a complete map from random or free variables to entire trajectories. The expansions used here are detailed in Hover & Triantafyllou (2005, 2006), as well as in the other literature cited on polynomial chaos.

2. Gradient method with Legendre chaos

We consider the continuous-time, unconstrained, fixed terminal time optimization problem

\[ \dot{x} = f(x, u, t) \]
\[ x(t_0) = x_0 \]
\[ \min_J \frac{\partial}{\partial x} = \Psi(x(t_f)) + \int_{t_0}^{t_f} L(x, u, t) dt \]

where the state vector is \( x \), the control is \( u \), \( t \) is time, and function \( f \) contains the plant dynamics model. The terminal cost is \( \Psi \), and \( L \) is the integral cost kernel; the initial and final times are \( t_0 \) and \( t_f \), respectively. We employ the first-order gradient method based on the Maximum Principle (Bryson & Ho, 1975; Bryson, 1999) due to its ease of use and robustness for general problems. An adaptive iteration gain for speeding up convergence is described in the example below, as is a method for handling strong nonlinearities and open-loop instability. The control update step \( \delta u = -k[L_u^T + \dot{x}_u^T \Lambda] \), where the subscripts denote partial derivatives and \( \Lambda \) the costate vector, is referenced several times later in the paper.

Employing the polynomial chaos, the set \([x, L, u]\) takes on a finite modal expansion, i.e., \( x(t) = \sum_{i=0}^{P} x_i(t) P_i(\xi), \quad L(t) = \sum_{i=0}^{P} \Lambda_i(t) P_i(\xi), \quad u(t) = \sum_{i=0}^{P} u_i(t) P_i(\xi) \). Here \( P \) refers to the size of the expansion; higher \( P \) leads to higher accuracy in stochastic simulations. \( \xi \) is the vector of \( d \) independent free variables, each element of which is assumed to have equal likelihood and lie in the range \([-1,1]\), i.e., it has a uniform distribution. This \( \xi \) vector contains all of the free variables under consideration in the optimization problem, including initial conditions, plant parameters, and parameters of the cost function.

\( P \) refers to the \( i \)th scalar Legendre polynomial, and the multi-dimensional case uses the tensor product, that is, the \( P_i \)'s are made up of products of one-dimensional orthogonal polynomials \( P_j \). Hence we use the following form for a two-dimensional chaos that contains polynomials up to second order:

\[ \begin{align*}
\xi(t) &= \sum_{i=0}^{P=5} x_i(t) P_i(\xi) \\
&= x_0(t) p_0(\xi_1) p_0(\xi_2) + x_1(t) p_1(\xi_1) p_0(\xi_2) + x_2(t) p_0(\xi_1) p_1(\xi_2) + x_3(t) p_1(\xi_1) p_1(\xi_2) + x_4(t) p_1(\xi_1) p_2(\xi_2) + x_5(t) p_2(\xi_1) p_2(\xi_2)
\end{align*} \]

We have \( p_0(\xi) = 1 \), \( p_1(\xi) = \xi \), \( p_2(\xi) = (3\xi^2 - 1)/2 \), and so on, so that the zeroth mode of such an expansion is always deterministic, i.e., it does not depend on the free parameters in \( \xi \). Because the \( P_i \) are scalars, the chaos expansion can be carried out cleanly in operations with vectors and matrices.

The specific steps of the gradient procedure are taken in the usual way, but with expanded state, costate, and control vectors, e.g., \( \mathbf{x}^T = [x_0^T x_1^T \cdots x_5^T] \). Further details are given in Hover & Triantafyllou (2005), where variability in the cost kernel \( L \) was considered for the one-dimensional case. We emphasize that \( J \) depends on the free parameters. Once the "chaos optimization" is done, fixing the parameters leads to a realization of \( J \) - and to the entire trajectory.
3. Implementation issues

The gradient method is simple to program and often appropriate for engineering solutions. Practical results can be obtained in reasonable time, and for a wide variety of applications. The gradient method is also a continuous-time approach, and so is amenable to adaptive time step routines that can accelerate the calculation while preserving accuracy. In contrast with the dynamic programming approaches, computational cost of the gradient method grows approximately linearly with the number of states. This makes the method suitable for larger-scale systems, and long-time problems.

Bryson and Ho pointed out two serious deficiencies of the method that have to be addressed in any implementation: poor convergence properties near the optimum point, and the requirement of a stabilizing initial control signal. Convergence properties have to be considered in the context of the problem being solved, however - useful solutions do not necessarily have to be very near the optimum point, and the user thus has available a direct tradeoff between optimality and computational effort. In an effort to streamline this convergence, we employ here an adaptive gain strategy. The idea is as follows (where the lack of time dependence for a vector indicates the entire trajectory): After the modes for the \(j\)th iteration of state \(x'_j(t, u'_j)\), costate \(\lambda'_j(t, x'_j, u'_j)\), and cost \(J_j(x'_j, u'_j)\) are computed for a given control \(u'_j\),

- if \(j < J_j\), set \(J_g = J_j\); increase the iteration gain \(k\), and compute the \(y'_j\) or
- if \(j \geq J_j\), erase the last step by resetting \(x'_j = x'_{j-1}, \lambda'_j = \lambda'_{j-1}\), and \(u'_j = u'_{j-1}\); then decrease the iteration gain, and compute \(y'_{j+1}\).

Here \(J_g\) is the last improving \(J\). Hence non-improving iterations will be erased, and at each iteration, the iteration gain either increases or decreases. We use a gain growth factor of 1.1 and a reduction factor of 0.7 in our implementation.

The requirement for a stabilizing control trajectory on the first iteration raises a difficulty for strongly nonlinear and unstable systems (as in the example below). Our approach is as follows, for the case that all the free variables are in the initial conditions; the plant model has fixed coefficients, as does the cost function. In this instance, a linearization of the plant completely decouples the chaos modes, and they are all controllable from \(u'_j\); each mode of the control vector stabilizes the corresponding mode of the state vector. As a specific case, if \(L\) is quadratic and \(\Psi = 0\), an LQR design will stabilize the system. The initial control trajectories in the gradient iteration can then be generated by applying the feedback law in a time simulation, using the initial state modal amplitudes. For example, if \(x_1(t = 0) = \xi_1\) and \(x_2(t = 0) = \xi_2/2\) in the \(P = 5\), two-dimensional expansion of the previous section, the second and third elements of \(x'(t = 0)\) are one and one-half, respectively.

The key idea then is to gradually build the nonlinear terms from zero to their full values during a suitable number of gradient iterations, say \(N\). It is the iteration itself that allows this entirely pragmatic procedure to be attempted; convergence to a static solution is effectively replaced by tracking a changing optimization problem. Once the nonlinearity has “matured,” convergence in the usual sense continues. A notable sensitivity to the iteration gain exists for this procedure, however; if the gain is too small, then the differential control \(\delta u\) applied at each iteration may not keep pace with the growing effect of the nonlinearity. On the other hand, if the gain is too large then of course divergence can result. We found that the iteration gain adaptation mentioned above should be postponed until \(j > N\) for best results.

Variations on this procedure, of gradually bringing into force the destabilizing components of the plant model, may be useful in a wider class of problems.

4. Example

We consider a multi-dimensional (\(d = 2\)) optimization problem:

\[
\begin{align*}
\dot{x}_1 &= -\frac{1}{10} x_1 - x_2 - \frac{3}{4} x_1^2 - \frac{3}{4} x_1 x_2 - \frac{3}{2} x_2^2 \\
\dot{x}_2 &= x_1 \\
x_1(t=0) &= U(0,1/3) = \xi_1 \\
x_2(t=0) &= U(0,1/3) = \xi_2 \\
\min_u J &= \frac{1}{2} x_2^2(t=10) + \frac{1}{2} \int_0^{10} (x_1^2 + x_2^2 + u^2) \, dt
\end{align*}
\]

Each of the state initial conditions is parameterized with a single uniformly-distributed variable having zero
mean and variance $1/3$, i.e., the support of $\xi$ is $[-1, 1]$. Considering the open-loop behavior of this system, Figure 1 confirms that the nonlinearity is quite pronounced and destabilizing over more than ninety percent of the initial condition space; only in a small neighborhood of the origin is the plant open-loop stable. An unstable fixed point exists at $(x_1 = 0, x_2 = -2/3)$. The plant as written is not difficult to stabilize with feedback, however, if the control also explicitly cancels out the modelled nonlinearities.

We applied the Legendre chaos in two dimensions to the problem, so as to generate a parametrization of optimizing trajectories for all $(x_1(t = 0), x_2(t = 0))$ in the support. Figure 2 shows the realizations of optimum trajectories from some of the initial points. Overall, the trajectories are as expected, with roughly exponential approach to the origin. The curves are negative-asymmetric; that is to say, paths beginning at $x$ and $-x$ are not simply rotated versions of each other. Hence, the trajectories are unlike those that would be achieved with pure cancellation of the nonlinear effects, such as in sliding mode control. A number of the trajectories go outside the support of the initial conditions, but this is allowed because the support of the evolved distributions can be larger. We note also that the initial condition $(0, 0)$ leads to a non-degenerate state trajectory, even though it is obviously a stable fixed point for the open-loop plant. This effect arises because what is computed is a distribution of control and state trajectories, and this particular realization is evidently sub-optimal for the benefit of the entire set. The lower graph of Figure 2 shows that this is not a phenomenon that can be eliminated with higher $P$. A second illustration of the effects of optimizing the distribution, as opposed to any single realization, is shown in Figure 3, wherein dependence of the control on the initial condition is shown at various times. At the origin the control is clearly nonzero except at large times - hence, trajectories that start near the origin are driven away. As time increases, the contour lines evolve and eventually flatten out to near zero levels. An interesting consequence of this fact is that even though the control distribution is parameterized with initial conditions, $u'$ taken with $t = 0$ and $\xi = \xi(t)$ will fail as a feedback law.

We can expect errors in the calculation to occur from several sources. These include the inner products used in computing the evolution of the distributions (see Ghanem & Spanos, 1991), the time integrations, a finite number of iterations in the gradient method, and the finite truncation of the Legendre chaos. The inner products are constructed using 16th-order Gauss-Legendre quadrature, and this provides about eight significant digits. After the integrals are computed, we
are suitable, and not a cause for systematic errors. Ter-
ple are confirmed to be larger than 10
the remaining terms for the conditions of this exam-
(absolute error 1

ing the truncation of series, clearly there is a strong
P = 15 is worse than
P = 10. Further, the table does not adequately support any assertion that the consis-
tance level of the deterministic case can be achieved at
all, even with very high P. This is a topic that has to be consi-
dered in future work. Despite these concerns, the procedure does successfully create a stabilizing set of feedforward control trajectories, that are certainly close to optimal.

The cost of computing this parametrization of trajectories is also given in Table 1. We see that the number of iterations is effectively constant for all P, or at worst slowly growing. The relative cost of the entire optimization is given with respect to the deterministic

![Figure 4: Left: State trajectories for deterministic and chaos optimizations with the initial condition \( x = (1, 1) \), \( P = 36 \). Right: Error between the chaos state solution \( x^* \) and a deterministic state solution \( x_d \); only the first plant state is shown. The error between \( x^* \) and the states as driven by the chaos control solution, i.e., \( x(u^*) \), indicates consistency of the optimization; this comparison is also shown for the deterministic case, denoted \( x_d \) and \( x(u_d) \).](image1)

![Figure 5: The consistency check of Figure 4, for various chaos orders P, first plant state.](image2)

perform a threshold operation wherein elements with absolute value less than 10\(^{-4}\) are set explicitly to zero; the remaining terms for the conditions of this example are confirmed to be larger than 10\(^{-2}\). For the time integration, we employ fourth-order Runge-Kutta with adaptive stepsize within fixed, uniformly-spaced frames having \( \Delta t = 0.01 \). Linear interpolation of the control is made when propagating the state forward, and we similarly interpolate the state when propagating the costate backward. Inspection of the control and state trajectories confirms that the integration adaptation setting (absolute error 1 \( \times 10^{-6} \)) and the linear interpolation are suitable, and not a cause for systematic errors. Ter-
mination of the iteration loop is of course a user choice. We compared results for \( ||\delta u||_2/||u||_2 < 1 \times 10^{-4} \) and \( 1 \times 10^{-8} \), and found no appreciable difference. Regarding the truncation of series, clearly there is a strong tradeoff between computational cost and accuracy, especially in this case of strong nonlinearities and fairly long simulation time; the distributions can become quite complex, e.g., see Hover & Triantafyllou (2006).

These truncation errors can be assessed in at least two ways, which are illustrated for the case \( x_1(t = 0) = 1 \) and \( x_2(t = 0) = 1 \) in Figure 4 and Figure 5. First, a given realization of the parameterized solution can be compared with a single deterministic optimization, using the same gradient technique with the same algo-
algorithmic parameters. This is given in Figure 4; the error
between these two signals for the first channel is about 0.05 during the transient, and then less than 0.01 as the origin is approached. The signals are not in close agreement because, as noted above, what is optimized is a family of trajectories; any specific realization does not represent an optimal path by itself. Shown in the same figure, a state trajectory is created by driving a deterministic time simulation with a particular realiza-
tion \( u^* \) from the optimization, that is \( x(u^*) \); it can be compared with a realization \( x^* \) taken directly from the optimization. This comparison indicates whether the actual system dynamic response is consistent with the chaos optimal solution. We see in the figure that the chaos state trajectory (c subscript) is not consistent with the chaos control, whereas this relationship is very good for the deterministic optimization (d subscript). That this discrepancy is directly related to truncation in the chaos series is illustrated in Figure 5. Here we plot the errors for the first state channel, with different \( P \) values; the error rapidly diminishes as \( P \) increases. The value \( P = 36 \) is sufficient to reach a level of 0.01, but better could possibly be achieved with higher \( P \). The peak absolute value errors are tabulated in Ta-
ble 1 for all \( P \) considered, and these show that the consistency is not always monotonic in \( P \); the case of \( P = 15 \) is worse than \( P = 10 \). Further, the table does not adequately support any assertion that the consist-
ency level of the deterministic case can be achieved at all, even with very high \( P \). This is a topic that has to be considered in future work. Despite these concerns, the procedure does successfully create a stabilizing set of feedforward control trajectories, that are certainly close to optimal.

The cost of computing this parametrization of trajectories is also given in Table 1. We see that the number of iterations is effectively constant for all \( P \), or at worst slowly growing. The relative cost of the entire optimization is given with respect to the deterministic
case, and grows almost linearly with $P$; that is, the integration cost scales linearly with the number of expanded states.

5. Summary

Trajectory optimization for a continuous-time dynamic system can be carried out such that a family of trajectories is directly parameterized with a few free variables. These free parameters can be designed by the user to lie in the model description, in the cost function, and in the initial conditions. In the last case, studied here, we find that the trajectories are close approximations to those created from deterministic optimization, except near the origin. We covered a two-dimensional space of initial conditions with about one order of magnitude more effort than a single deterministic optimization.

Acknowledgements

Work was supported by the Office of Naval Research’s National Naval Responsibility (NNR) program, grant number N00014-02-1-0623, monitored by Dr. Sharon Curtin-Beerman, and ONR grant N00014-02-1-0623. We also acknowledge ONR’s Autonomous Operations Future Naval Capability (AOFNC) program, grant N00014-02-C-0202, monitored by J. Valentine.

References


Franz Hover received the M.S. and Sc.D. degrees from the Woods Hole Oceanographic Institution/Massachusetts Institute of Technology Joint Program in Applied Ocean Physics and Engineering. He is currently a Principal Research Engineer with the Department of Mechanical Engineering at the Massachusetts Institute of Technology, with research interests in the area of applied control and robotics, fluid mechanics, and offshore structures. Dr. Hover is a member of the IEEE and the ASME.

Michael Triantafyllou is Professor of Mechanical and Ocean Engineering at the Massachusetts Institute of Technology, Director of the Center for Ocean Engineering, and Director of the Testing Tank and Marine Hydrodynamics Laboratories. His research interests are in the dynamics and control of marine vehicles, and biomimetics. He is a member of the American Physical Society, the Society of Naval Architects and Marine Engineers, and the International Society of Offshore and Polar Engineers.
<table>
<thead>
<tr>
<th>polynomial order</th>
<th>P (2D)</th>
<th>iterations</th>
<th>relative cost</th>
<th>cost/P 1.00</th>
<th>1.00</th>
<th>cost/P 1.00</th>
<th>consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>244</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>$1.0 \times 10^{-4}$</td>
<td>unstable</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>203</td>
<td>1.55</td>
<td>0.26</td>
<td>2.7</td>
<td>$10^{-1}$</td>
<td>unstable</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>223</td>
<td>2.33</td>
<td>0.23</td>
<td>6.2</td>
<td>$10^{-2}$</td>
<td>unstable</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>233</td>
<td>3.35</td>
<td>0.22</td>
<td>8.6</td>
<td>$10^{-2}$</td>
<td>unstable</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>235</td>
<td>4.55</td>
<td>0.22</td>
<td>3.5</td>
<td>$10^{-2}$</td>
<td>unstable</td>
</tr>
<tr>
<td>6</td>
<td>28</td>
<td>247</td>
<td>7.19</td>
<td>0.26</td>
<td>8.5</td>
<td>$10^{-3}$</td>
<td>unstable</td>
</tr>
<tr>
<td>7</td>
<td>36</td>
<td>236</td>
<td>8.93</td>
<td>0.25</td>
<td>7.7</td>
<td>$10^{-3}$</td>
<td>unstable</td>
</tr>
</tbody>
</table>

Table 1: Modes required for various polynomial orders in the two-dimensional case; the zero’th order case corresponds to a deterministic optimization. The iterations and costs listed achieve an $L_2$ relative tolerance on the control update of $10^{-4}$. Consistency is the maximum absolute value of $x^*_c - x(u^*_c)$, as illustrated in Figure 5.