Optimizing state trajectories using surrogate models with application on a mechatronic example

T.J.P. Lefebvre, F. De Belie, G. Crevecoeur
Department of Electrical Energy, Systems and Automation (EESA)
Ghent University (UGent), Technologiepark 913, B-9052 Zwijnaarde, Belgium
e-mail: Tom.Lefebvre@UGent.be

***Abstract*— The classic design- and simulation methodologies, that are constituting today's engineer main tools, fall behind with industry's ever increasing complexity. The strive for technological advancement heralds new performance requirements and optimality remains no longer a concern limited to regime operation. Since the corresponding dynamic optimization problems incorporate accurate system models, the current techniques are plagued by the high computational weight these multi-disciplinary and highly dimensional system models bear with them. This imbalance advocates for the need to adapt the existing approaches. In this study we propose an algorithmic framework as an extension of the direct transcription method, which has already proven its usefulness concerning this matter. It is suggested to construct a surrogate model of the derivative function that is iteratively refined in a region of interest. Thereafter the method will be illustrated on an academic yet nonlinear example.**

***Index Terms*— Direct transcription, Dynamic optimization, Nonlinear mechatronic systems, Surrogate models**

I. Introduction

Optimality of regime operation no longer suffices and transient phenomena are gaining fast importance in nowadays industry, even more for those applications where regime operation loses all meaning and operation merely entails transient behaviour. Optimality has now direct bearing to the realised state trajectory and therefore they should be addressed within the corresponding optimality formulation. Techniques that cope with such problem formulations require the availability of a dynamic model. Whilst dynamical accuracy benefits from increased model complexity - by including multiple modelling disciplines, and considering high dimensionality and nonlinearity - the solution techniques do not and the designated influence can no longer be evaluated analytically; especially when no analytical expression is available and evaluations correspond with numerical computer experiments [1], [2].

For the purpose of fast and reliable dynamic optimization of yet complex and accurate systems, it is necessary to accommodate this computational burden without affecting optimality. From the successful and repeated application within several engineering disciplines, the computational strength of surrogates already appeared [3]. To that end, we will try to incorporate these surrogate modelling techniques within the framework and methods addressing dynamic optimization.

This article provides a brief introduction to these separate research areas and proposes an algorithm to bridge the gap.

II. Trajectory optimization

Whenever it is one's desire to optimize a system in such manner that the time evolution of states - governed by the system dynamics - is of significance to the interpretation of optimality, it is classified as a dynamic optimization problem [4], [5]. A special class of dynamic optimization problems is referred to as trajectory optimization [6]. Whilst closely related to optimal control problems, trajectory optimization is most often practiced offline and before actual system operation, as part of the design process. The online problem addressing the realisation of the optimized trajectory, as in a tracking problem, is then referred to as the corresponding optimal control problem. So while the input sequence will be considered within trajectory optimization in general, it does not embody its main result.

A. General formulation

A general formulation is presented in (1), where $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and $\boldsymbol{u} \in \mathbb{R}^{n_u}$, represent state- and input variables respectively. Each state- and input combination is related to an instantaneous cost $\mathcal{L}(\cdot)$, the term $E(\cdot)$ expresses a possible additional cost related to the end state of the system. The total cost related to a specific time evolution of $\boldsymbol{u}(t)$ over the time interval $[t_0, t_f]$ is denoted by J. Moreover, the state- and input variables can be subject to constraint functions $\boldsymbol{g}(\cdot)$ and $\boldsymbol{h}(\cdot)$. It is assumed that dynamics are governed by the derivative nonlinear function $\boldsymbol{f}(\cdot)$ and matrix $\boldsymbol{B}(\cdot)$.

$$\min_{\boldsymbol{u}(\cdot)} \; J = \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau))\mathrm{d}\tau + E(\boldsymbol{x}(t_f)) \qquad \text{s.t.} \begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} \\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = 0 \\ \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) < 0 \end{cases} \tag{1}$$

This mathematical formulation can be moulded to fit a vast amount of existing and powerful optimization algorithms; yet, dynamic optimization of physical plant and control system of today's mechatronical applications remains a challenging problem for which the availability of an assessable dynamic system model, and thus function $\boldsymbol{f}(\cdot)$ and $\boldsymbol{B}(\cdot)$, is of vital essence.

B. Direct Transcription

Direct transcription (DT) has proven to be a helpful instrument for solving nonlinear dynamic optimization problems. Its principle is founded on the idea of transcribing the infinite into a finite dimensional problem, resulting into a discretised problem that can be approached as a Nonlinear Program (NLP). The NLP is solved thereafter by practice of an effective nonlinear programming technique.

Transcription is achieved by partitioning and successively parameterizing the continuous problem, leading to the desired discrete equivalent representation. Agreement with dynamics is governed by a set of nonlinear constraints that wards continuity of the time evolution of states.

The technique sets off by partitioning the time interval into n_t time segments, $\mathcal{I}_k$, for which the boundaries are denoted by t_k and t_{k+1}, respectively:

$$\mathcal{I}_k = [t_k, t_{k+1}],\ k \in \chi = \{0,\ \dots\ , n_t - 1\}$$

For each time interval we introduce a parameterized zero-order hold representation of the control variables, $\boldsymbol{u}_k(t), t \in \mathcal{I}_k$, that is uniquely defined by the parameters, $\boldsymbol{q}_k$. Such is an accurate approximation of realistic control scenarios. More elaborate representations exist yet would distract more than they would contribute.

$$\boldsymbol{u}_k(t) = \boldsymbol{q}_k$$

The major trick entails the parameterisation of the state trajectory by introducing $n_t + 1$ state node values, $\boldsymbol{s}_k$, as additional function handles and considering an initial value problem for each time segment. Numerical solution yields the trajectory pieces, $\boldsymbol{x}_k(t|\boldsymbol{s}_k, \boldsymbol{q}_k)$:

$$\begin{cases} \dot{\boldsymbol{x}}_k = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{B}(\boldsymbol{x}_k)\boldsymbol{q}_k,\ t \in \mathcal{I}_k \\ \boldsymbol{x}_k(t_k) = \boldsymbol{s}_k \end{cases}$$

Continuity of the state trajectory is enforced by introducing n_t constraints, $\boldsymbol{\zeta}_k$, demanding that the propagation of the state trajectory starting at time instant t_k is equal to the state value at time instant t_{k+1}:

$$\boldsymbol{\zeta}_k(\boldsymbol{s}_{k+1}, \boldsymbol{s}_k, \boldsymbol{q}_k) \equiv \boldsymbol{s}_{k+1} - \boldsymbol{x}_k(t_{k+1}|\boldsymbol{s}_k, \boldsymbol{q}_k) = 0$$

Arranging these ingredients to our liking allows to cook up a formulation of the problem, that can be approached as a NLP. For a more extensive elaboration of the family of direct dynamic optimization methods, we refer to [7].

$$\begin{aligned} &\min_{\boldsymbol{s}_k, \boldsymbol{q}_k} \sum_{j=1}^{n_t} \int_{t_j}^{t_{j+1}} \mathcal{L}(\boldsymbol{x}_j(\tau), \boldsymbol{u}_j(\tau))\mathrm{d}\tau + E(\boldsymbol{s}_{n_t}) \\ &= \min_{\boldsymbol{s}_k, \boldsymbol{q}_k} \sum_{j=1}^{n_t} \mathcal{L}_j(\boldsymbol{s}_j, \boldsymbol{q}_j) + E(\boldsymbol{s}_{n_t}) \\ &\qquad \text{s.t.} \begin{cases} \boldsymbol{\zeta}_k(\boldsymbol{s}_{k+1}, \boldsymbol{s}_k, \boldsymbol{q}_k) = 0 \\ \boldsymbol{g}(\boldsymbol{s}_k, \boldsymbol{q}_k) = 0 \\ \boldsymbol{h}(\boldsymbol{s}_k, \boldsymbol{q}_k) < 0 \end{cases} \end{aligned} \tag{2}$$

We end this section by noting that the numerical propagation can be as simple as trapezoidal quadrature, simplifying the computation of constraints yet at the cost of an increased discretization error [8].

III. Surrogate approximations in Direct Transcription

The direct transcription methodology offers a powerful instrument to handle challenging optimization problems and opens the door to dynamic optimization of systems that exhibit strong nonlinearity. Nonetheless, is does not omit evaluation of the derivative function which is often found to be computationally cumbersome; when for example function evaluations correspond with a finite element analysis. Moreover, algorithms that deal with the resulting NLP, such as Sequential Quadratic Programming (SQP), depend inherently on reliable gradient evaluations of the nonlinear constraints; and thus by action of the chain rule, the gradient of the derivative function.

To accommodate both of these challenges we suggest to replace the functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{B}(\cdot)$ by means of the approximated but computationally beneficial models $\hat{\boldsymbol{f}}(\cdot)$ and $\hat{\boldsymbol{B}}(\cdot)$:

$$\dot{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}\ \rightarrow\ \hat{\dot{\boldsymbol{x}}}(\boldsymbol{x}, \boldsymbol{u}) = \hat{\boldsymbol{f}}(\boldsymbol{x}) + \hat{\boldsymbol{B}}(\boldsymbol{x})\boldsymbol{u} \tag{3}$$

In this way evaluations of the derivative function and its gradient are computationally effortless albeit with a loss of accuracy. Subsequently, the NLP can be solved at lower computational cost, thus resulting in shorter computation times, therewith making the direct transcription methodology applicable for computationally burdened problems.

A. Surrogate modelling

Surrogate modelling techniques address the approximation of computationally cumbersome computer models that effectuate a deterministic input-output relation, $\boldsymbol{\omega} \rightarrow \boldsymbol{\phi}$; based on a limited number of known input-output couples. The techniques aim to construct an elementary model that mimics the generic relation whilst omitting its complex underlying structure, such that the complex model can be evaluated at untried parameters sites with less extensive computational force.

$$\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{\omega})\ \rightarrow\ \hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\phi}}(\boldsymbol{\omega})$$

The overarching principle originates from the notion that the distance between input sites is a measure for the correlation between the resulting output values. This dissertation considers the Kriging predictor and it is assumed that we have a set of m training sites $\boldsymbol{\Omega} = \begin{bmatrix}\boldsymbol{\omega}_1 & \dots & \boldsymbol{\omega}_m\end{bmatrix}'$, $\boldsymbol{\omega}_i \in \mathbb{R}^n$, and responses $\boldsymbol{\Phi} = \begin{bmatrix}\boldsymbol{\phi}_1 & \dots & \boldsymbol{\phi}_m\end{bmatrix}'$, $\boldsymbol{\phi}_i \in \mathbb{R}^q$, to our disposal.

The predictor is premised on a model that expresses the deterministic response, $\boldsymbol{\phi}(\boldsymbol{\omega})$, as the superposition of a regression model, $\boldsymbol{\mathcal{F}}(\boldsymbol{\omega})$, and stochastic process, $\boldsymbol{z}(\boldsymbol{\omega})$. The regression model embodies a linear combination of base functions, $\boldsymbol{\psi}(\boldsymbol{\omega})$, whilst the stochastic process is characterised by a mean value equal to zero and the following covariance expression:

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\omega}) = \boldsymbol{\psi}(\boldsymbol{\omega})'\boldsymbol{\beta} \text{ and } \mathrm{E}(\boldsymbol{z}_i(\boldsymbol{v})\boldsymbol{z}_i(\boldsymbol{w})) = \sigma_i^2 \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{v} - \boldsymbol{w})$$

Where σ_i^2 expresses the process covariance for the i^{th} component of the response and $\mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{v} - \boldsymbol{w})$, defines a parameterized correlation model. As was noted the correlation model depends explicitly on the distance between parameter sites. Further, we

introduce following notations for the design matrix, $\boldsymbol{\Psi}$, the correlation matrix, $\boldsymbol{\Pi}$, and correlation function, $\boldsymbol{\pi}(\boldsymbol{s})$:

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{\omega}_1) & \dots & \boldsymbol{\psi}(\boldsymbol{\omega}_m) \end{bmatrix},$$
$$\boldsymbol{\Pi}_{ij} = \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{\omega}_i - \boldsymbol{\omega}_j) \text{ and}$$
$$\boldsymbol{\pi}(\boldsymbol{s}) = \begin{bmatrix} \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{\omega}_1 - \boldsymbol{\omega}) & \dots & \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{\omega}_m - \boldsymbol{\omega}) \end{bmatrix}$$

Kriging considers the linear predictor:

$$\hat{\boldsymbol{\phi}}(\boldsymbol{\omega}) = \boldsymbol{\alpha}(\boldsymbol{\omega})'\boldsymbol{\Phi}$$

A definition for the function, $\boldsymbol{\alpha}(\boldsymbol{\omega})$, can be obtained by expressing the error $\boldsymbol{\epsilon}(\boldsymbol{\omega}) = \hat{\boldsymbol{\phi}}(\boldsymbol{\omega}) - \boldsymbol{\phi}(\boldsymbol{\omega})$. Given the stochastic interpretation of $\boldsymbol{\phi}(\boldsymbol{\omega})$, we can define the mean squared error (MSE): $\mathrm{E}\left(\boldsymbol{\epsilon}(\boldsymbol{\omega})^2\right)$. Minimization of this expression to the designated model parameters, yields the desired expression for $\boldsymbol{\alpha}(\boldsymbol{s})$, and thus $\hat{\boldsymbol{\phi}}(\boldsymbol{s})$:

$$\hat{\boldsymbol{\phi}}(\boldsymbol{\omega}) = \boldsymbol{\psi}(\boldsymbol{\omega})'\tilde{\boldsymbol{\beta}} + \boldsymbol{\pi}(\boldsymbol{s})'\boldsymbol{\Pi}^{-1}(\boldsymbol{\Phi} - \boldsymbol{\Psi}\tilde{\boldsymbol{\beta}})$$

Where $\tilde{\boldsymbol{\beta}}$, $\tilde{\sigma}^2$ and $\boldsymbol{\theta}$ are determined by means of a maximum likelihood estimation. It is easily seen that the predictor entails the superposition of the regression model, and a distance based weighing of the observed deviation between output values and the regression model at training sites. For a more extensive elaboration of the methodology and an exposition of typical regression and correlation models, we refer to [9].

B. Sketch of an algorithmic architecture

Since the surrogate model has to be initiated with no prior knowledge available about the solution nor the generic dynamical behaviour, a space filling approach ought to be preferred. Amongst these space filling methods Latin Hypercube Sampling (LHS) is most common practice [10].

The resulting *surrogate* solution, $\{\hat{\boldsymbol{x}}^*\}$, will in general, however optimal under the surrogate dynamics, not be in agreement with the simulated trajectory, $\boldsymbol{x}^{sim}_{\hat{\boldsymbol{u}}^*}$, - that is the trajectory resulting from simulation of the optimized control sequence with real dynamics - since the surrogate dynamic model is only an approximation, and as such only reliable close to sampled sites. This clearly indicates that the sampling strategy is of crucial importance when surrogate models are included within the trajectory optimization framework.

An iterative solution approach (Fig. 1) is proposed that assesses the surrogate performance along the momentary trajectory. Whenever it is found necessary, the surrogate model shall be refined in the neighbourhood of the optimal trajectory resulting from the optimization with the current surrogate model. A similar approach has been suggested by Allison and Deshmukh [11] but has not yet been elaborated.

IV. The Adaptive Surrogate Refinement Algorithm

The proposed algorithm starts by initialising n_s training sites - by application of the LHS strategy - in the training set, D_{ini}, where we evaluate the functions $\{\boldsymbol{f}, \boldsymbol{B}\}$. These evaluations are used to construct initial surrogate models $\{\hat{\boldsymbol{f}}_0, \hat{\boldsymbol{B}}_0\}$ by use of Kriging. Each step is illustrated unambiguously hereafter:

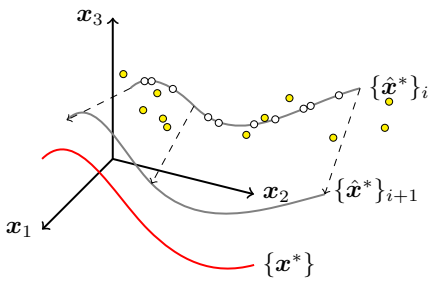

Fig. 1: Illustration of the iterative solution approach through the feasible solution space. The white dots denote the evaluation sites whilst the yellow present a possible refinement sampling plan.

1) *Acquire momentary solution:* Solve the NLP of (2) with the current surrogate dynamics using a standard NLP solver, resulting in a momentary optimum $\{\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{u}}^*\}_i$.
2) *Generation of validation sites:* We generate n_v time instants $t_j,\ j \in \vartheta_i = \{1,\ \dots\ , n_v\}$ by sampling from a piece wise linear distribution $p_{\hat{\boldsymbol{x}}^*}(t_k)$ that relates to the surrogate's accuracy along the current solution, and such that samples are more likely to be drawn at sites where accuracy is lowest. The authors propose:

$$\{t_j\}_{j\in\vartheta_i} \sim p_{\hat{\boldsymbol{x}}^*}(t_k) = \max(\mathrm{MSE}_f(\hat{\boldsymbol{x}}^*(t_k)), \mathrm{MSE}_B(\hat{\boldsymbol{x}}^*(t_k)))$$

   The accuracy of the matrix $\boldsymbol{B}$ is assessed by considering each column individually.
3) *Error estimation:* Evaluate the surrogate dynamics along the solution by defining an error estimate that compares real derivative function evaluations with surrogate evaluations at the time instants t_j. The authors propose:

$$\max\left(\frac{t_f - t_0}{n_v}\sum_{j=1}^{n_v}\left|\dot{\boldsymbol{x}}\left(\hat{\boldsymbol{x}}^*(t_j), \hat{\boldsymbol{u}}^*(t_j)\right) - \hat{\dot{\boldsymbol{x}}}(\hat{\boldsymbol{x}}^*(t_j), \hat{\boldsymbol{u}}^*(t_j))\right|\right)$$

   With $\dot{\boldsymbol{x}}(\cdot)$ and $\hat{\dot{\boldsymbol{x}}}(\cdot)$ defined as in in (3), $\hat{\boldsymbol{x}}^*(t_j)$ a spline interpolation of the discrete vector function $\hat{\boldsymbol{x}}^*(t_k)$, since generally the sampled time instants t_j will not coincide with the time nodes t_k, $\exists j \in \vartheta_i : t_j \notin \{t_k\}_{k\in\chi}$; and $\hat{\boldsymbol{u}}^*(t_j)$ an evaluation of the parameterized input.

   The corresponding value can be interpreted as the maximum integrated mean derivative error over the considered time interval, which functions as an estimate for the expected absolute surrogate induced error between the optimal end state, $\hat{\boldsymbol{x}}^*(t_{n_t})$, and $\boldsymbol{x}^{sim}_{\hat{\boldsymbol{u}}^*}(t_f)$. As such we will be able to define an intuitive and generic value for the convergence threshold.
4) *Surrogate refinement:* As long as the convergence condition has not been met, construct $\{\hat{\boldsymbol{f}}_{i+1}, \hat{\boldsymbol{B}}_{i+1}\}$ by adding the function evaluations from the previous step, to the training set $D_{i+1} = D_i \cup \{\hat{\boldsymbol{x}}^*(t_j)\}_{j\in\vartheta_i}$ and repeat steps 1:4. As such dynamics will become more accurately from iteration to iteration, in that region seems to contain the true optimum. Crucial is to initialize the next iteration with the solution of the prior.

The algorithm will generate a solution whose proposed trajectory will be in agreement with simulation of the suggested control sequence. That is, the trajectory resulting from the optimization will be in a vicinity of the trajectory resulting from simulation of the optimized control sequence and the real dynamics, assuring practical feasibility of the optimum.

However, the solution will be optimal under the final surrogate dynamics; which only approximate the real dynamics accurately in the vicinity of the surrogate optimum. This self-referencing property will result in an optimum that may deviate from the real optimum. For the time being no measures are taken to drive the surrogate optimum to the real optimum and it is assumed that the initial sampling delivers a sufficient approximation of the global dynamics to assure correspondence with the real optimum.

V. Illustrative Examples

In this section we will address a mechatronic problem to illustrate the applicability and performance of the proposed algorithm. For this problem an analytical expressions is available and therefore no real urge is present to apply the algorithm. Yet the example has the specific property that it exhibits heavily nonlinear dynamics, and is as such an ideal candidate to demonstrate the potential benefits of the algorithm. It should be noted that for this problem no reduction of computation time shall be observed and performance will be quantified by examining the total number of derivative function evaluations.

The problem considers a trajectory optimization of a whirling pendulum, to demonstrate the working principle and iterative character of the algorithm and its handling of nonlinear dynamics. We assumed a piecewise constant control scenario. The resulting NLP was solved by use of *fmincon*, a standard function provided by the software package MATLAB (R2013a; Mathworks, Natick, MA, USA).

A. Dynamical model

The whirling pendulum can be represented by a nonlinear 4-dimensional state space model. Rotation about the z-axis of the observers frame is denoted by ϕ. A coordinate frame was fixed to the pendulum's point of suspension and rotation about the pendulum's x-axis is denoted by θ, consequently the pendulum's state can be represented by $\boldsymbol{x} = [\,\theta\ \dot{\theta}\ \phi\ \dot{\phi}\,]'$. The state space model has been derived through application of the Euler-Lagrange principle.

$$\mathrm{L}(q,\dot{q}) = \mathrm{T} + \mathrm{D} - \mathrm{V} \tag{4a}$$

$$\frac{\partial \mathrm{L}(q,\dot{q})}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathrm{L}(q,\dot{q})}{\partial \dot{q}} = \Upsilon = \begin{bmatrix} u_\theta \\ u_\phi \end{bmatrix} \tag{4b}$$

Expressions for kinetic, potential and dissipative energy, are given respectively by T, V and D:

$$\mathrm{T} = \frac{1}{2}[\,\dot{\theta}\ \dot{\phi}\,]\mathrm{diag}(I_x, I_y \sin^2\theta + I_z\cos^2\theta)[\,\dot{\theta}\ \dot{\phi}\,]',$$

$$\mathrm{V} = -\frac{mg}{2} r\cos\theta \text{ and } \mathrm{D} = \frac{1}{2}[\,\dot{\theta}\ \dot{\phi}\,]\mathrm{diag}(b_\theta, b_\phi)[\,\dot{\theta}\ \dot{\phi}\,]'$$

The moments of inertia are defined in the coordinate system fixed to the pendulum and located at the suspension point with the z-axis according to the longitudinal dimension of the cylindrical pendulum. The length of the cylindrical pendulum is denoted by r, the mass of the system by m and the gravitational constant by g. Values are noted in table I.

parameter	m	r	$I_x = I_y = 2I_z$	$b_\theta = b_\phi$
value	10 [kg]	1 [m]	4.44 $[\mathrm{kgm}^2]$	0.5 $\left[\frac{\mathrm{kg\cdot m}}{\mathrm{s}}\right]$

TABLE I: Parameter values of the whirling pendulum model.

B. Optimization problem

We defined an illustrative minimal input cost problem for which the optimal control sequence had to be determined over a predefined time span of 1.5 seconds. Optimality had to be achieved for the quadratic cost, $\mathcal{L}_j(\boldsymbol{q}_j)$, and was subject to the following boundary states:

$$\mathcal{L}_j(\boldsymbol{q}_j) = \boldsymbol{q}_j'\boldsymbol{q}_j,$$
$$\boldsymbol{x}_0 = \left[\tfrac{\pi}{2}\quad 0\quad 0\quad 0\right]' \text{ and } \boldsymbol{x}_{n_t} = \left[\tfrac{\pi}{4}\quad 0\quad \tfrac{\pi}{2}\quad 0\right]'$$

Dynamics were discretised using trapezoidal quadrature with $n_t = 80$ steps, resulting in a step size of about 19 ms.

C. Results

The iterative character of the algorithm is illustrated in Fig. 2. As anticipated, the algorithm gradually forces the optimal trajectory towards the true optimum. Despite the considerable increase of necessary iterations before a feasible solution was attained, the overall number of derivative function evaluations is drastically diminished. A reduction of 97.57% was achieved, having evaluated the derivative function 2880 times to obtain the real optimum compared to 70 evaluations using ASR. Included are 30 evaluations to initialize the surrogate models and 5×8 additional refinement evaluations.

The final control sequence has been simulated with the real dynamics as a visual validation of the measure that was taken to guarantee practical feasibility.

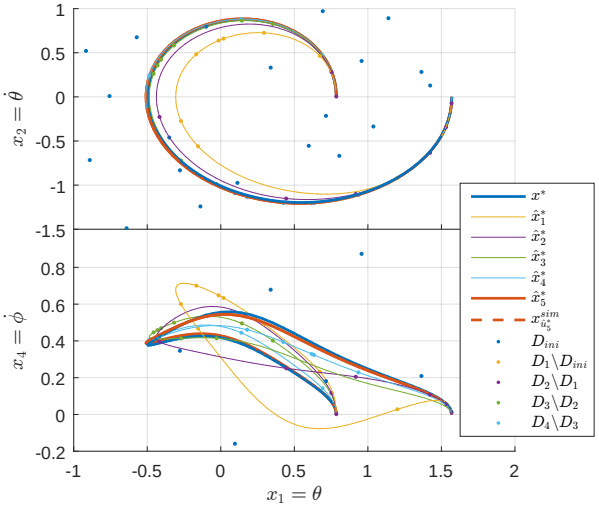

Fig. 2: The true optimal trajectory, $\boldsymbol{x}^*$, the iterative trajectories, $\hat{\boldsymbol{x}}_i^*$, and the simulated trajectory for the final control sequence, $\boldsymbol{x}_{\hat{\boldsymbol{u}}^*}^{sim}$, which coincides with $\hat{\boldsymbol{x}}_5^*$. ($n_t = 80, n_s = 90, n_v = 8$).

VI. Conclusions and future development

In this treatise we proposed to tackle computational unwieldiness, introduced by derivative function evaluations in a dynamic optimization context, by exploitation of the computational grace of surrogate models. The proposed algorithmic framework managed to enforce reductions in evaluations up to 97.57%, when applied on a highly nonlinear mechatronic problem, and thus exhibits large potential for future applications where derivative function evaluations correspond with extensive computation time and as such offer a lever to extend the applicability of trajectory optimization to highly accurate models. For the time being a self-referencing mechanism is still inherent to the algorithm and partially subverts the global optimization, moreover due to its stochastic nature the solution may differ for each run. Therefore further sculpting of the algorithm shall be required before a robust and wide applicability will be possible. Yet the authors are confident that within a reasonable timespan such improvements can be obtained.

Acknowledgements

The authors acknowledge the support of the Strategic basic research project EMODO of the Flanders Make Strategic Research Centre for the Manufacturing Industry, and the FWO Research project G.0D93.16N.

References

[1] P. J. Moriarty and A. C. Hansen. *AeroDyn Theory Manual*. National Renewable Energy Laborator, 1617 Cole Blvd., Golden, CO 80401-339, August 2005.

[2] Jason P Halloran, Ahmet Erdemir, and Antonie J van den Bogert. Adaptive surrogate modeling for efficient coupling of musculoskeletal control and tissue deformation models. *Journal of biomechanical engineering*, 131(1):011014, 2009.

[3] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[4] Rush D Robinett III, David G Wilson, G Richard Eisler, and John E Hurtado. *Applied dynamic programming for optimization of dynamical systems*, volume 9. SIAM, 2005.

[5] Lorenz T. Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043 – 1053, 2007. Special Issue on Process Optimization and Control in Chemical Engineering and Processing.

[6] P. Fiorini and Z. Shiller. Time optimal trajectory planning in dynamic environments. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1553–1558 vol.2, Apr 1996.

[7] Moritz Diehl, Hans Georg Bock, Holger Diedam, and Pierre-Brice Wieber. Fast direct multiple shooting algorithms for optimal robot control. *Fast Motions in Biomechanincs and Robotics*, 2005.

[8] Oskar Stryk. *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*, chapter Numerical Solution of Optimal Control Problems by Direct Collocation, pages 129–143. Birkhäuser Basel, Basel, 1993.

[9] H. B. Nielsen, S. N. Lophaven, and J. Søndergaard. *DACE - A Matlab Kriging Toolbox*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2002.

[10] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[11] Anand P Deshmukh and James T Allison. Design of nonlinear dynamic systems using surrogate models of derivative functions. pages V03BT03A011–V03BT03A011, 2013.