A Metaheuristic Particle Swarm Optimization Approach to Nonlinear Model Predictive Control

Julian Mercieca and Simon G. Fabri
Department of Systems and Control Engineering
University of Malta
Msida MSD 2080, Malta
Email: julianmercieca@gmail.com, simon.fabri@um.edu.mt

***Abstract*—This paper commences with a short review on optimal control for nonlinear systems, emphasizing the Model Predictive approach for this purpose. It then describes the Particle Swarm Optimization algorithm and how it could be applied to nonlinear Model Predictive Control. On the basis of these principles, two novel control approaches are proposed and analysed. One is based on optimization of a numerically linearized perturbation model, whilst the other avoids the linearization step altogether. The controllers are evaluated by simulation of an inverted pendulum on a cart system. The results are compared with a numerical linearization technique exploiting conventional convex optimization methods instead of Particle Swarm Optimization. In both approaches, the proposed Swarm Optimization controllers exhibit superior performance. The methodology is then extended to input constrained nonlinear systems, offering a promising new paradigm for nonlinear optimal control design.**

Keywords-particle swarm optimization; model predictive control; optimal control; nonlinear control; computational intelligence; swarm intelligence; evolutionary intelligence; artificial intelligence; metaheuristic algorithms

I. INTRODUCTION

This paper discusses the use of Particle Swarm Optimization for optimal control of nonlinear systems. It proposes two novel control schemes in this regard and presents a more detailed perspective on earlier work by the same authors [1]. In the case of systems exhibiting linear dynamics, optimal and robust control theory offer well-developed tools to optimize a number of performance indices that embody desirable objectives and ensure performance robustness. These range from classical and fundamental robust control approaches [2] to more advanced, theoretically elegant and computationally tractable solutions [3], [4].

In contrast to linear optimal and robust control, its nonlinear counterpart (namely optimization constrained by a nonlinear dynamical system) is still a developing field. Its roots were laid down in the 1950s with the introduction of the Pontryagin maximum principle (a generalization of the Euler-Lagrange equations derived from the calculus of variations [5]) and dynamic programming, leading to the Hamilton-Jacobi-Bellman partial differential equations [6]. These were more theoretical contributions than practical design techniques. Numerous design methodologies have now been developed for nonlinear optimal control, often following different paths and techniques. The problem is attacked on several different fronts including extensions of linear theory, utilizing generalizations of the Lyapunov methodology, and brute force computation to name a few [7].

The advent of the microprocessor and the subsequent computer revolution opened up an entirely new possibility for optimal control: obtaining solutions directly through numerical computations. While the solution of the Hamilton-Jacobi-Bellman equation remains intractable in all but the simplest of cases, Euler-Lagrange type trajectory optimizations provide an alternative, more computationally feasible approach. Computers are able to provide relatively efficient solutions by solving trajectory optimizations that produce open-loop control trajectories as a function of time (as opposed to a state-feedback law). Feedback can then be incorporated by the repeated on-line solution of these trajectory optimizations, an approach known as receding or moving horizon. A heavy exploitation of the receding horizon methodology spawned the technique of Model Predictive Control [8]. Plants with slow dynamics were among the first candidate applications of this approach because on-line inter-sample computation of a sequence of manipulated variable adjustments in order to optimize the future behaviour of a controlled process using minimal control effort became feasible [9]. Additionally, the receding horizon strategy was a natural approach to constrained systems because constraints could be directly incorporated into the optimizations, enabling plant operators to run the plant near constraint boundaries, which can increase productivity and reduce product quality variation [10], [11]. Furthermore, Model Predictive Control seems extremely powerful for processes with dead-time or if the set-point is programmed. This is evidenced by its successful implementation in industrial process applications [8]–[13].

However, despite being an attractive control scheme for manipulating the behaviour of complex systems [14] and exhibiting excellent dynamic performance in both industrial applications and theoretical studies [15]–[17], the application of Model Predictive Control to nonlinear systems, known as Nonlinear Model Predictive Control (NMPC), is complicated largely due to the optimization method that has come to be used in these controllers. A fundamental difficulty of the NMPC approach is the requirement to solve nonconvex constrained optimization problems. Most existing works are based on nonlinear programming methods [18] that only

yield local optimum values, with the latter depending on the selection of the starting point. For this purpose, Alaniz [19] developed a particular numerical linearization technique to obtain a convex constrained optimization problem, albeit at the cost of performance deterioration. Other attempts to solve the nonconvex optimization problems exploit Genetic Algorithm (GA) optimizers [20]. However these face many challenges, including enormous computational effort due to its natural genetic operations [21], [22]. Although this may be reduced by using a real-value representation in the GA [21], [23], [24], some deficiencies in GA performance have been highlighted in recent research. Applications governed by highly epistatic objective functions [25], [26] reveal shortfalls in performance, which is further worsened by the GA's premature convergence [25].

This paper presents and analyses in depth two novel NMPC controllers based on a powerful optimization paradigm called Particle Swarm Optimization (PSO). PSO was first developed by Kennedy and Eberhart in 1995 [27]. This metaheuristic algorithm has been found to be robust in solving continuous nonlinear optimization problems [23], [26]–[28] and capable of generating high quality solutions with more stable convergence characteristics and shorter calculation times than other stochastic methods [23], [26], [29]. The salient feature of PSO lies in its learning mechanism, distinguishing it from other computational intelligence techniques, such as genetic algorithms, where PSO has been shown to have more attractive properties [30], [31]. PSO is governed by less tunable parameters and is notably easy to program and implement using basic logic and mathematical operations. The swarm intelligence algorithm stands in clear contrast with many optimization techniques for being derivative-free, being less sensitive to the objective function's nature, namely continuity and convexity, and not requiring good initial solutions for the iteration process to start. Furthermore, its flexibility enables its integration with other optimization techniques, forming hybrid tools [32]. Its ability to avoid local minima and to cater for stochastic objective functions, as is the case of representing a random optimization variable, further strengthens PSO's capacity to achieve superior optimization performance [32]. Owing to its simple concept and high efficiency, PSO has become a widely adopted optimization technique and has been successfully applied to many real-world problems [33]–[40]. Moreover, PSO's superiority is confirmed when compared with other optimization algorithms in various application areas [41]–[45]. Also, in the process of validating new global optimization techniques, researchers have proven that PSO performs well in several benchmark optimization problems [46]–[49].

One of the novel controllers presented in this paper is based on a numerical linearization technique first proposed by Alaniz in [19] that is based on conventional convex optimization methods. By contrast, the proposed controller exploits PSO techniques for optimization. The second novel controller proposed in this paper does away with any form of numerical linearization to achieve optimization of the cost function. Both controllers are simulated on an inverted pendulum on a cart problem and compared with the NMPC controller in [19].

The rest of the paper is organized as follows. Section II is an explanation of the implemented PSO algorithm, while Section III outlines the design of the three NMPC controllers evaluated in this paper. Section IV then presents the simulation setup, results and analysis, followed by a brief conclusion in Section V.

II. Particle Swarm Optimization

The particle swarm optimization algorithm is a population-based search algorithm inspired by the social behaviour of birds within a flock [27]. The very simple behaviour followed by individuals in a flock emulates their own successes and the success of neighbouring individuals. The emergent collective behaviour is that of discovering optimal regions of a high dimensional search space.

In a PSO algorithm, each particle representing a potential solution is maintained within a swarm. In simple terms, the particles are therefore "flown" through a multidimensional search space where the position of each particle is adjusted according to the experience of itself and its neighbours. Let $\mathbf{x}_i(t)$ denote the position of particle i in the search space at time step t, which denotes discrete time steps unless otherwise stated. The position of the particle is changed by adding a velocity vector, $\mathbf{v}_i(t)$, to the current position *i.e.,*

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{1}$$

with $\mathbf{x}_i(0) \sim U(\mathbf{x}_{min}, \mathbf{x}_{max})$, where $U(\mathbf{x}_{min}, \mathbf{x}_{max})$ denotes the continuous uniform probability distribution within the real-valued space $(\mathbf{x}_{min}, \mathbf{x}_{max})$. The optimization process is driven by the velocity vector, reflecting both the experiential knowledge of the particle (known as *cognitive component*) and socially exchanged information from the particle's neighbourhood (known as *social component*). In this paper we implement a particular PSO algorithm know as global best PSO [50], which exhibits very fast convergence rates much needed for our predictive control application. For the global best PSO, or *gbest* PSO, the neighbourhood for each particle is the entire swarm, thus employing the social network of the star topology type. In this situation, the social information is the best position found by the swarm, referred to as $\hat{\mathbf{y}}(t)$.

For *gbest* PSO, the velocity of particle i is calculated as

$$\begin{aligned} v_{ij}(t+1) = \quad & v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + \\ & c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \end{aligned} \tag{2}$$

where $x_{ij}(t)$, $y_{ij}(t)$ and $v_{ij}(t)$ are the position, personal best position and velocity of particle i in dimension $j = 1, \ldots, n_x$ at time step t respectively, $\hat{y}_j(t)$ is the global best position in dimension j, c_1 and c_2 are positive acceleration constants used to scale the contribution of the cognitive and social components respectively, and $r_{1j}(t), r_{2j}(t) \sim U(0,1)$ are random values in the range $[0,1]$, sampled from a continuous uniform distribution. These random values introduce a random element to the algorithm.

The personal best position, $\mathbf{y}_i$, associated with particle i is the best position the particle has visited since the first time

step. Considering a minimization problem, the personal best position at the next time step, $t+1$, is calculated as

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases} \quad (3)$$

where $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ is the fitness function, which is a measure of how close the corresponding solution is to the optimum, quantifying the performance, or quality, of a particle (or solution).

The global best position, $\hat{\mathbf{y}}(t)$, at time step t, is defined as

$$\hat{\mathbf{y}}(t) \in \{\mathbf{y}_0(t) \dots \mathbf{y}_{n_s}(t)\} | f(\hat{\mathbf{y}}(t)) = \min\{f(\mathbf{y}_0(t)) \dots f(\mathbf{y}_{n_s}(t))\} \quad (4)$$

where n_s is the total number of particles in the swarm. Equation 4 therefore states that $\hat{\mathbf{y}}(t)$ is the best position discovered by any of the particles so far. In this paper, it is calculated as the best personal-best position. Algorithm 1 summarizes the *gbest* PSO algorithm.

Further to the basic PSO algorithm just described, the speed of convergence and quality of solutions are improved using velocity clamping and inertia weight [51]. The efficiency and accuracy of our optimization algorithm is governed by the exploration-exploitation trade-off [52]. *Exploitation* is the ability of a search algorithm to concentrate the search around a promising area in order to refine a candidate solution. *Exploration*, on the other hand, is the ability to locate a global optimum by exploring different regions of the search space. A good optimization algorithm balances these contradictory objectives in an optimal manner. For the PSO algorithm, these objectives are reached using the velocity update equations.

Algorithm 1 *gbest* PSO [50]

Create and initialize an n_x-dimensional swarm
repeat
  for *each particle* $i = 1, \dots, n_s$ **do**
    //set the personal best position
    if $f(\mathbf{x}_i) < f(\mathbf{y}_i)$ **then**
      $\mathbf{y}_i = \mathbf{x}_i$;
    end
    //set the global best position
    if $f(\mathbf{y}_i) < f(\hat{\mathbf{y}})$ **then**
      $\hat{\mathbf{y}} = \mathbf{y}_i$;
    end
  end
  for *each particle* $i = 1, \dots, n_s$ **do**
    update the velocity using Equation (2);
    update the position using Equation (1);
  end
until *stopping condition is true*;

The velocity update presented in Equation (2) comprises three terms that contribute to the step size of particles. The early applications of basic PSO revealed that the velocity quickly explodes to large values, especially for particles located far from the neighbourhood best and personal best positions. As a consequence, particles have large position updates resulting in them leaving the boundaries of the search space and diverging. The global exploration of particles may be controlled by clamping velocities to stay within boundary constraints [53]. If a specified maximum velocity is exceeded, the particle's velocity is set to the maximum velocity. Let $V_{max,j}$ denote the maximum allowed velocity in dimension j. Particle velocity is then adjusted prior to the position update using,

$$v_{ij}(t+1) = \begin{cases} v_{ij}'(t+1) & \text{if } v_{ij}'(t+1) < V_{max,j} \\ V_{max,j} & \text{if } v_{ij}'(t+1) \geq V_{max,j} \end{cases} \quad (5)$$

where v_{ij}' is calculated using Equation (2).

The value of $V_{max,j}$ is essential to control the granularity of the search by clamping escalating velocities. Large values of $V_{max,j}$ facilitate global exploration, while smaller values encourage local exploitation. Too small values of $V_{max,j}$ leads to insufficient exploration beyond locally good regions, increasing the number of time steps to reach an optimum, with the risk of the swarm becoming trapped in a local optimum, with no means of escape. On the other hand, too large values of $V_{max,j}$ risk the possibility of missing a good region, having the particles possibly jumping over good solutions and continuing to search in fruitless regions of the search space. Despite this disadvantage of particles possibly jumping over optima, particles move faster.

The problem of finding a good value for each $V_{max,j}$ still stands. We require the balance between (a) moving too fast or too slow, and (b) exploration and exploitation. Here, we select $V_{max,j}$ values to be a fraction of the domain of each dimension of the search space. That is,

$$V_{max,j} = \delta(x_{max,j} - x_{min,j}) \quad (6)$$

where $x_{max,j}$ and $x_{min,j}$ are the maximum and minimum values of the domain of $\mathbf{x}$ in dimension j respectively, and $\delta \in (0,1]$. In a number of empirical studies it was found that the value of δ is problem-dependent [54], [55], and the best value for our situation was therefore obtained empirically.

While velocity clamping exhibits the advantage of a controlled explosion of velocity, it also presents a difficulty when all velocities are equal to the maximum velocity. If no precautionary measures are implemented, particles remain searching on the boundaries of a hypercube defined by $[\mathbf{x}_i(t) - \mathbf{V}_{max}, \mathbf{x}_i(t) + \mathbf{V}_{max}]$. A particle may stumble upon the optimum, but in general exploiting this local area is difficult. This problem is solved in our algorithm by introducing an inertia weight, a concept introduced by Eberhart and Shi [28] as a mechanism of controlling the exploitation and exploration abilities of the swarm, with the original intention of eliminating the need for velocity clamping [33]. The inertia weight was successful in addressing the first objective, but failed to completely eliminate the need for velocity clamping. The inertia weight, w, controls the particle's momentum by weighting the contribution of the previous velocity - in other words, controlling how much memory of the previous flight

direction will influence the new velocity. For the *gbest* PSO, the velocity equation changes from Equation (2) to

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \quad (7)$$

The value of w is essential to ensure convergent behaviour while optimally trading off exploration and exploitation. For $w \geq 1$, velocities increase over time, accelerating towards the maximum velocity (assuming a velocity clamping strategy), and the swarm diverges. Particles fail to change direction to return back towards promising areas. For $w < 1$, particles undergo a deceleration until their velocities reach zero (depending on the values of the acceleration coefficients). Large values of w facilitate exploration with increased diversity, while small w promotes local exploitation. However, very small values eliminate the exploration ability of the swarm since little momentum is then preserved from the previous time step, enabling quick changes in direction. The smaller w, the more do the social and cognitive components control position updates.

As with the maximum velocity, the optimal value for the inertia weight is problem-dependent [55]. Here, we make use of dynamically changing inertia values, starting with large inertia values that decrease over time to smaller values. This way, particles are allowed to explore in the initial search steps, while favouring exploitation as time increases.

The inertia weight is dyamically varied using the linear decreasing method, where an initially large inertia weight (usually 0.9) is linearly decreased to a small value (usually 0.4). Following Yoshida *et al.* [56], Suganthan [57], Ratnaweera *et al.* [58], Naka *et al.* [59], we set

$$w(t) = (w(0) - w(n_t))\frac{(n_t - t)}{n_t} + w(n_t) \quad (8)$$

where n_t is the maximum number of time steps that the algorithm is executed, $w(0)$ is the initial inertia weight, $w(n_t)$ is the final inertia weight, and $w(t)$ is the inertia at time step t. Note that $w(0) > w(n_t)$.

III. Nonlinear Model Predictive Control

A nonlinear dynamic system may be represented by a set of nonlinear differential equations [60] that may be discretized for computational purposes using Euler's method, where T_s is the sampling period and k is the sample index in discrete-time, as follows:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + T_s f(\mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k), k) \quad (9)$$

$$\mathbf{y}(k) = g(\mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k), k) \quad (10)$$

Arguments of the nonlinear function f include a state vector $\mathbf{x}(k)$, a control input $\mathbf{u}(k)$, and a disturbance input $\mathbf{v}(k)$. The set of physical quantities that can be measured from the system constitute the output, $\mathbf{y}(k)$, which is also a nonlinear function g of the same arguments. More accurate discretization approximations, such as the Runge-Kutta methods, can be used if the system dynamics are highly nonlinear or the sampling period is large.

The Model Predictive Control (MPC) design methodology is characterized by three main features: an explicit model of the plant, computation of control signals by optimizing predicted plant behaviour, and a receding horizon [10]. MPC's receding horizon strategy can be explained using Figure 1. An internal model predicts how the plant will react, starting at the current time k, over a discretized prediction interval. The letter l denotes the number of discrete steps in this interval. Each discrete step spans a time of T_s seconds, therefore the prediction interval lasts lT_s seconds. The predicted behaviour is governed by the present state $\mathbf{x}(k)$, an estimated disturbance history $\mathbf{v}$, and a control history $\mathbf{u}$ that is to be applied. The objective is to select the control history that yields the best predicted behaviour with respect to a reference trajectory and optimization parameters.

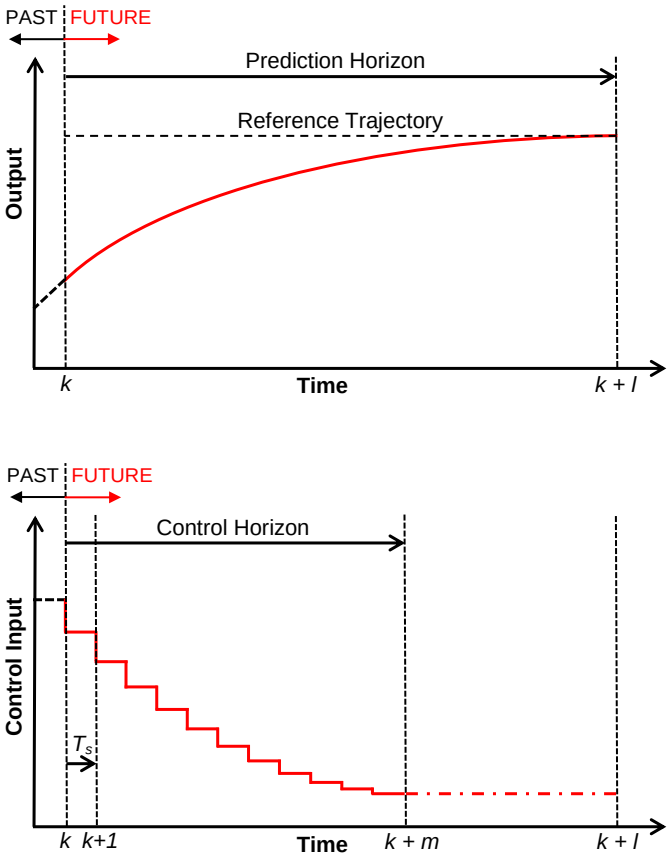

Fig. 1. Nonlinear Model Predictive Control: A receding horizon strategy

MPC solves for the control history, which is a sequence of m vector values. Two adjacent control values are separated by a time step of T_s, therefore having a control history spanning mT_S seconds. During each time step, the control values are held constant and the values are assumed to change instantaneously as soon as a new time step is started. After the control history has ended, the control signal is held constant until the prediction interval is over.

Once the optimal control history has been computed, the first N time steps of the solution are applied to the plant and

the rest are discarded. After these N time steps have passed, the cycle of forming predicted behaviours and computing the control history is repeated. In this paper, we choose N to be unity, however this number can be increased to reduce the rate of production of solutions.

The cost function used in our performance evaluation is given by Equation (11) having a quadratic structure comprising two terms. The first term, weighted by a symmetric weighting matrix $\mathbf{Q}(k)$, penalizes the deviations from a reference trajectory that occur throughout the prediction interval. A specific value of the reference is denoted by $\tilde{\mathbf{y}}(k)$. The second term, weighted by a symmetric weighting matrix $\mathbf{R}(k)$, penalizes the magnitude of each control value in the control history.

$$J = \sum_{i=0}^{l-1} ||(\mathbf{y}(k+i) - \tilde{\mathbf{y}}(k+i))||^2_{\mathbf{Q}(k+i)} + \sum_{i=0}^{m-1} ||\mathbf{u}(k+i)||^2_{\mathbf{R}(k+i)} \tag{11}$$

As previously described, if m is in the range $1 \leq m \leq l$, then the last value in the control history is held constant for the final $(l-m)$ time steps. The value of $\mathbf{R}(k+m-1)$ should therefore have a different magnitude to compensate for the added duration of $\mathbf{u}(k+m-1)$.

Control and output constraints are considered in their inequality form, with the constraints being enforced at each discretized point in the control history and output trajectory, as shown in equations (12) and (13).

$$\mathbf{u}(k)_{min} \leq \mathbf{u}(k) \leq \mathbf{u}(k)_{max} \tag{12}$$

$$\mathbf{y}(k)_{min} \leq \mathbf{y}(k) \leq \mathbf{y}(k)_{max} \tag{13}$$

The MPC problem in this setting is to minimize J by choosing $\mathbf{u}$, subject to the constraints in equations (12) and (13) and the dynamics of equations (9) and (10). We will now describe the three nonlinear model predictive controllers considered in this paper, two of them representing the novel contributions of this work.

A. A numerical linearization method

This method, proposed by Alaniz in [19], centres around a particular numerical linearization technique for generating the predicted output trajectory $\mathbf{y}$. A nominal control history $\bar{\mathbf{u}}$ is first chosen, then the corresponding nominal output trajectory $\bar{\mathbf{y}}$ is computed through numerical integration. Typically $\bar{\mathbf{u}}$ is the previous optimal solution, but it can be set equal to zero if none exist. The predicted output is then based on linearizing the control perturbation $\Delta\mathbf{u}$ about the nominal trajectory as follows:

$$\begin{aligned}
\mathbf{y}(k) &= \bar{\mathbf{y}}(k) + \boldsymbol{\alpha}_0 \Delta\mathbf{u}(k) \\
\mathbf{y}(k+1) &= \bar{\mathbf{y}}(k+1) + \boldsymbol{\alpha}_1 \Delta\mathbf{u}(k) + \boldsymbol{\beta}_0 \Delta\mathbf{u}(k+1) \\
\mathbf{y}(k+2) &= \bar{\mathbf{y}}(k+2) + \boldsymbol{\alpha}_2 \Delta\mathbf{u}(k) + \boldsymbol{\beta}_1 \Delta\mathbf{u}(k+1) + \\
&\quad \boldsymbol{\gamma}_0 \Delta\mathbf{u}(k+2) \\
&\vdots
\end{aligned} \tag{14}$$

The coefficients $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, $\boldsymbol{\gamma}_i$, ... are produced by computing a perturbed trajectory for each $\Delta\mathbf{u}(k+i)$ and finding the subsequent deviation from the nominal trajectory. Perturbed trajectories are the result of adding a pulse of magnitude one to the nominal control history at time $(k+i)$. Each trajectory is formed by propagating the present state $\mathbf{x}(k)$ over a fixed interval of time while applying an associated control history. The prediction interval and control history are divided into l and m discrete steps, respectively, of length T_s where $m \leq l$. After the control history has ended, it is held constant for the final $(l-m)$ time steps.

The MPC problem is to solve for the optimal control perturbation $\Delta\mathbf{u}^*$ by minimizing a cost function with respect to a reference trajectory and optimization parameters. The optimal control history is then the sum of the nominal control history and the optimal control perturbation [19]. By rearranging and simplifying the form of Equation (11), a set of matrices is obtained that leads to the unconstrained and constrained optimization problems. For the unconstrained case, Alaniz [19] presents a solution by using an equivalent least squares technique, while for the constrained case, the problem is reinterpreted so as to obtain the standard form handled by quadratic programming solvers.

Once the optimal control history is chosen, the first N time steps of the solution are applied to the plant. The cycle of forming predicted behaviours and solving for the optimal control perturbations is then repeated using the most recent feedback from the plant. The interested reader is referred to [19] for further detail about this technique.

B. A novel numerical linearization technique using PSO

A novel application of PSO proposed here exploits the aforementioned numerical linearization technique used in conjunction with the PSO algorithm, where the convex least squares or quadratic programming optimization methods are now replaced by the global best PSO algorithm. The evaluation function is the cost given by Equation (11), so that PSO searches for the optimal perturbed control history of Equation (14), denoted by $\Delta\mathbf{u}(k)^*$, in order to obtain the optimal control history $\mathbf{u}(k)^*$ that minimizes J. For this purpose we require an m-dimensional PSO, with each particle's position defined by $\mathbf{K}$, an m-dimension column vector equal to $\Delta\mathbf{U}(k)^*$, which is a column vector having $\Delta\mathbf{u}(k+i)^*$ as its elements.

C. A novel PSO-based nonlinear MPC strategy

The second novel controller makes use of the PSO search algorithm for obtaining the optimal control history that minimizes directly the cost function J given by Equation (11) without resorting to numerical linearization as represented by Equation (14). In this manner we simply use Equation (11) as the evaluation function to be minimized using global best PSO, thereby avoiding any linearization technique or mathematical result for minimization, albeit at an increased computational complexity. Each particle's position in the swarm represents the m-dimension column vector defining the optimal control history, $\mathbf{U}(k)^*$.

As we shall see, this remarkably straightforward approach produces the best results for the controllers studied in this

paper in terms of the performance index obtained. The block diagram in Figure 2 illustrates the structure of the proposed predictive control loop. A particle swarm optimizer uses the reference input and predicted output trajectories to minimize the quadratic cost function given by Equation (11) and compute the optimal control history that is then applied to the plant. The proposed controller is further enhanced by actively correcting the weighting matrix **R** in an adaptive manner, so that the chattering effect of the control input observed about the equilibrium point is reduced.

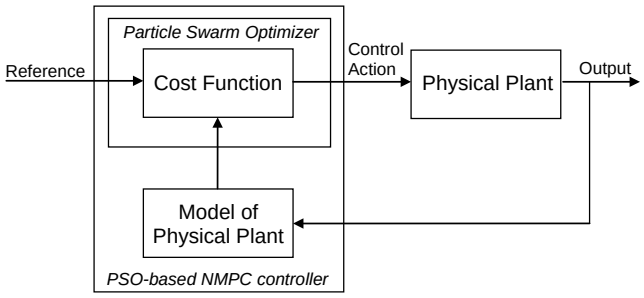

Fig. 2. PSO-based nonlinear MPC loop

IV. Performance Evaluation: Inverted Pendulum on Cart

The performance of the proposed controllers is evaluated by analyzing the results from simulation experiments. The plant chosen for simulation is an inverted pendulum on a cart and two types of controllers are generated for the three methods presented in the previous section; an unconstrained and constrained version. The latter problem shall only consider a single constraint that restricts the input as per the inequality given by Equation (12). Hence, no penalty functions are required. The pendulum is initially at the stable equilibrium point and the purpose of each controller is to invert the pendulum. Since the dynamics at the stable and unstable equilibrium points are very different, this is a good problem to demonstrate the effectiveness of our nonlinear MPC controllers.

A. Plant Model

The nonlinear model of the plant is derived by applying Newton's Laws of Motion to the free body diagrams in Figure 3. The resulting equations of motion are given by equations (15) and (16). A complete derivation is given in [19].

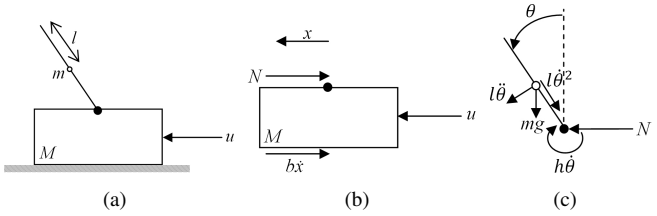

Fig. 3. (a) Inverted Pendulum on a Cart; (b) Free body diagram 1 (cart); (c) Free body diagram 2 (pendulum).

$$\ddot{x} = \frac{1}{M+m}[u - b\dot{x} - ml\ddot{\theta}\cos(\theta) + ml\dot{\theta}^2\sin(\theta)] \quad (15)$$

$$\ddot{\theta} = \frac{3}{4ml^2}[mgl\sin(\theta) - ml\ddot{x}\cos(\theta) - h\dot{\theta}] \quad (16)$$

M represents the mass of the cart that slides along a surface, m is the uniformly distributed mass of an ideal pendulum, $2l$ is the length of the ideal pendulum, b is the surface friction damping constant, h is the rotational friction damping coefficient, u is the force applied to the block, θ is the clockwise angle between the normal and the pendulum (as shown in Figure 3(c)), and x is the cart's horizontal displacement from its equilibrium position. To allow the model to be numerically integrated, equations (15) and (16) are expressed in terms of the state variables x, $\dot{x}$, θ, and $\dot{\theta}$. The second-order differential equations have the form given by Equation (17), where $\boldsymbol{\chi}$ is a vector variable. A vector field g is also created to combine the states into one state vector. The second-order differential equations are then discretized using the fourth-order Runge-Kutta method [19].

$$\ddot{\boldsymbol{\chi}} = f(\dot{\boldsymbol{\chi}}, \boldsymbol{\chi}, u), \quad \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = g(\dot{\boldsymbol{\chi}}, \boldsymbol{\chi}), \; \boldsymbol{\chi} = \begin{bmatrix} x \\ \theta \end{bmatrix} \quad (17)$$

B. Controller Layout

The simulation experiments were run on the Simulink software package [61]. The layout shown in Figure 4 is the simulated realization of the control loop given in Figure 2. It makes use of Matlab S-Functions that implement constrained or unconstrained versions of the PSO-based NMPC controller.

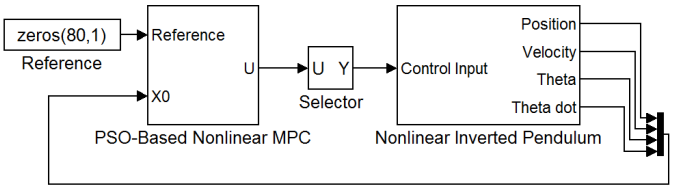

Fig. 4. Nonlinear MPC Simulink layout

C. Controller Parameters

The MPC controller rate is $\frac{1}{NT_s}$, where N is the number of controls in the control history that are applied to the plant. $N = 1$ is used in the controller since this is the typical value selected in MPC [10]. The computational load of MPC can be reduced if N is increased, but a disadvantage to having $N > 1$ is that some of the controls applied to the plant are based on old feedback. The fourth-order Runge-Kutta method is tested using different values for T_s, and it is established that the response with $T_s = 0.1s$ is almost indistinguishable from the actual response, thus using this value for the controller.

Since this controller is very computationally intensive, it is not feasible to have a long prediction length or control history. A value of $l = m = 20$ is chosen as a balance between

performance and computation time. This results in a controller capable of predicting for 2 seconds.

The two novel PSO-based controllers use the following PSO parameters, which were derived empirically through successive simulations:

- Each particle consists of 20 members, corresponding to the 20 elements that make up the optimal control perturbation history column vector, $\Delta\mathbf{U}(k)^*$, for the PSO-based numerical linearization method, or the optimal control history column vector, $\mathbf{U}(k)^*$, for the PSO-based NMPC controller.
- Swarm size, n_s = 30.
- Inertia weight w is set by Equation (8), where $w(0) = 0.9$ and $w(n_t) = 0.4$.
- Velocity clamping is governed by equations (5) and (6), with $\delta = 0.5$.
- Search space is limited to real-valued variables between $-300N$ ($x_{min,j}$) and $300N$ ($x_{max,j}$) for the unconstrained case.
- Acceleration coefficients $c_1 = 2$ and $c_2 = 2$.
- Number of iterations = 30.

Both weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ given in Equation (11) are set equal to the values shown in Equation (18), which includes a vector showing the order of the outputs. The objective of this controller is to invert the pendulum. Hence the penalty on θ is increased relative to the other states so that the controller focuses more on decreasing angular deviations than other state deviations. Initially, the cart must move back and forth until the pendulum gains enough momentum to swing up. Increasing the θ penalty would reduce the time required for the pendulum to swing up. Through successive simulation trials, a value of 100 was finally chosen as the penalty on the θ state by comparing the time required to achieve pendulum inversion.

$$\mathbf{y} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}, \ \mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ R = 1 \qquad (18)$$

Deviations are measured from the reference trajectory that is set equal to a zero column vector, as given by Equation (19). In this example, we set each reference state variable to zero at each time step of the prediction interval. This reference remains constant for the duration of the simulation.

$$\tilde{\mathbf{Y}} = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}^T \qquad (19)$$

D. Simulation Results

The responses of the unconstrained controller, using the three control schemes described in sections IIIA, IIIB, and IIIC, are shown in Figure 5 as blue, green and red curves respectively. The pendulum is initially set at the stable equilibrium point (180 degrees), hanging straight down. The performance index graph plots the value of the cost function J for each time step as calculated by the control law using Equation (11). The cart's position moves back and forth so that the pendulum gains momentum. This continues until there is enough momentum to swing up and invert the pendulum in the 0 degree position. Table I shows typical performance results obtained in this unconstrained control case. The performance index values quoted in the table are obtained from Equation (11) but this time using the actual output trajectory, instead of the predicted one, and the actual control inputs applied, instead of the computed control history. The summation is calculated for a sufficiently large amount of time ensuring that the system has settled into steady state. Table I therefore reveals the *true* performance index for the whole control action in the unconstrained case, demonstrating the typically superior performance of the proposed PSO-based NMPC controller. The results show that when PSO is used in conjunction with the numerical linearization technique, only a minimal advantage is obtained over the least squares method (an improvement in J of only 1.46%), as expected for the convex optimization problem being solved. On the other hand, the second proposed nonlinear PSO controller gives a significant improvement in J of 8.04% over the numerical linearization (least squares) counterpart.

TABLE I
UNCONSTRAINED NMPC: TRUE PERFORMANCE INDEX VALUES

Method	Numerical Linearization (Least Squares) [19]	Numerical Linearization (PSO)	PSO
True Performance Index J ($\times 10^6$)	1.4538	1.4326	**1.3369**

Figure 6 shows the response plots of the constrained controller when a single constraint, restricting the control input of the cart to be within $-45N$ and $45N$, is made active. In Figure 6, the final angular deflection is either 0° or 360°. Note that both these angles correspond to the same inverted position of the pendulum. For the novel PSO-based NMPC controller, the cart is noted to move a much smaller distance to achieve swing-up. In real-world terms, this translates to a more efficient process, with less work being done by the cart to achieve swing-up and equilibrium. This is further evidenced by Table II, which indicates that the novel PSO-based nonlinear MPC controller has the edge over the numerical linearization technique that uses quadratic programming, a method known to have problems in getting stuck at local minima [62]. We record a 14.78% improvement in J, accompanied by a very low standard deviation when the experiment is repeated over

TABLE II
CONSTRAINED NONLINEAR MPC: TRUE PERFORMANCE INDEX VALUES

Method	Numerical Linearization (Quadratic Programming) [19]	PSO (mean J)	PSO (standard deviation) ($\times 10^6$)
True Performance Index J ($\times 10^6$)	2.8534	**2.4316**	**0.02396**

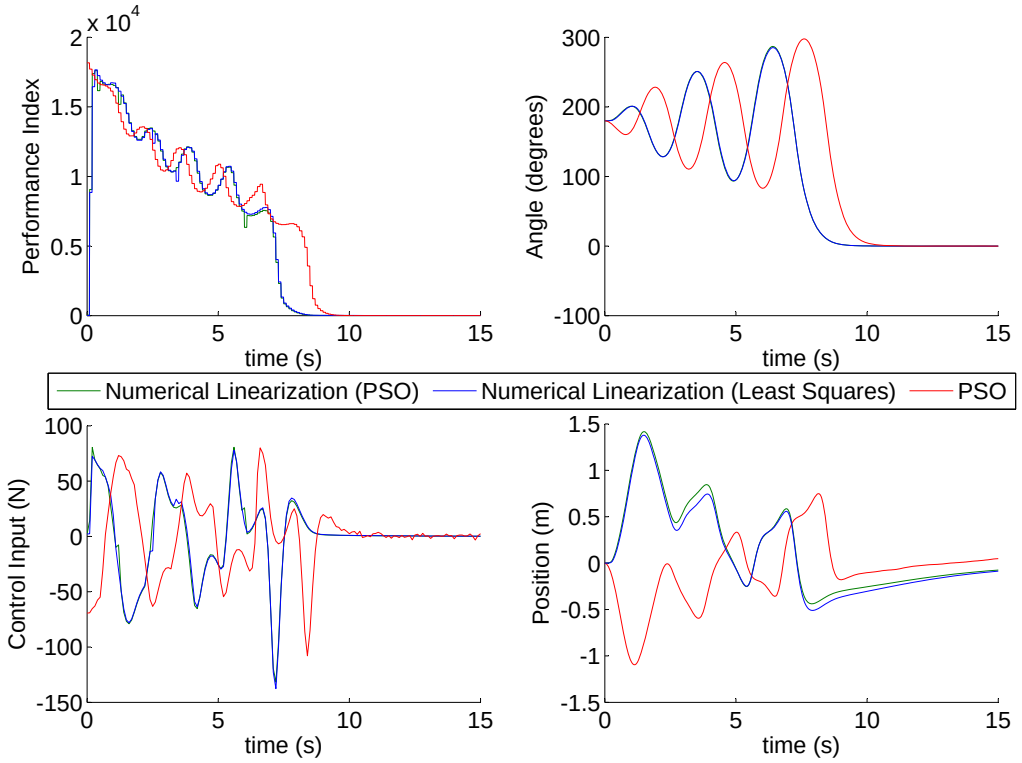

Fig. 5. Unconstrained nonlinear MPC: A comparison

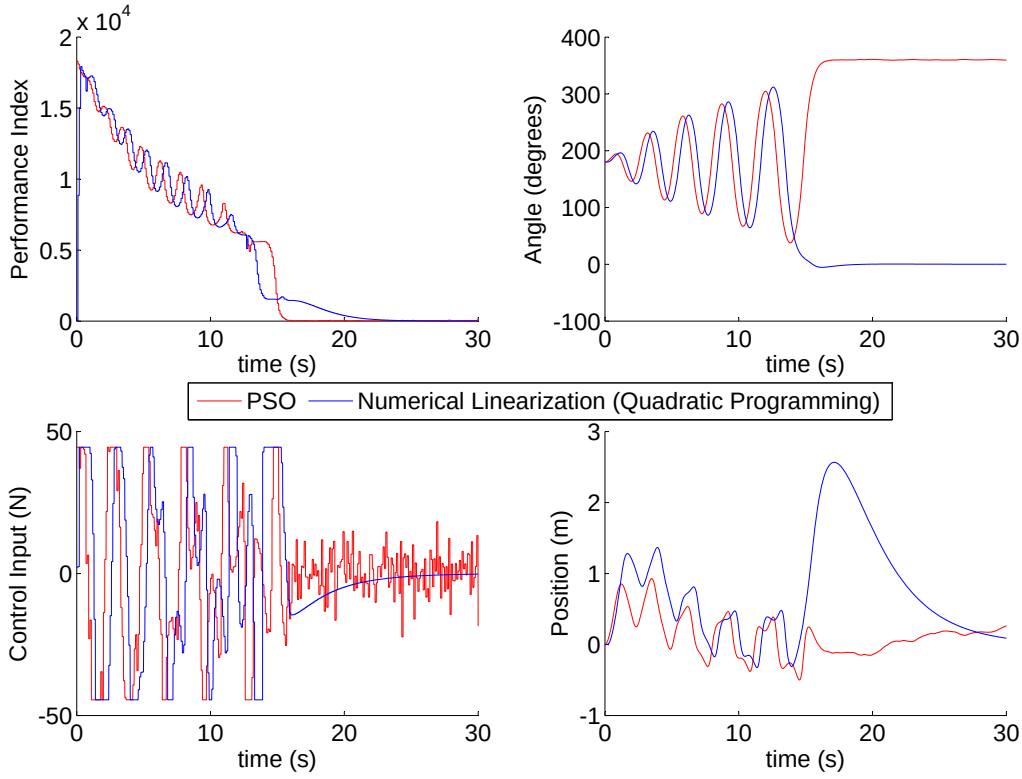

Fig. 6. Constrained nonlinear MPC: Restricting control input to within $-45N$ and $45N$ (10 independent trials)

10 trials indicating PSO's repeatable nature despite being a metaheuristic optimization method. Note that for the constrained case, using the numerical linearization technique in conjunction with PSO is computationally inefficient since every particle must be checked for its corresponding optimal control history, doubling the workload of its unconstrained counterpart, rendering it practically useless to investigate for this purpose.

The advantage of the novel PSO-based NMPC controller is even more evident in Figure 7, where the proposed active correction for the chattering effect of the control input is implemented for the same constrained NMPC problem by dynamically changing **R** appearing in Equation (11). The control input is being more heavily penalized when the angle approaches the equilibrium point by increasing R from 1 to 30. In other words, we are telling the system that in the close neighbourhood of the equilibrium point, minimal control effort is required, mitigating the effect of metaheuristic stochasticity. This reduces the true performance index even further, giving an improvement in J of 16.65% (see Table III), making the process even more efficient.

TABLE III
TRUE PERFORMANCE INDEX VALUE COMPARISON FOR ACTIVE R CORRECTION

Method	Numerical Linearization (Quadratic Programming) [19]	PSO
True Performance Index J ($\times 10^6$)	2.8534	**2.3810**

The system's robustness to model uncertainty is best illustrated by the simulation results of Figure 8. This is tested by randomly increasing or decreasing each of the plant model's parameters by 5% (all parameters are changed for every trial). Thus, the constrained NMPC controller is using a severely inaccurate model for its predictions and we are not actively controlling the control input weight R (to consider the worst case). Despite these adverse conditions, the results of Figure 8 show excellent performance and the pendulum swings up normally except for a larger distance now required. Table IV shows the corresponding changes implemented in the model parameters of one particular trial picked up at random in the simulation results of Figure 8. The corresponding performance index values are given in Table V, where although both controllers manage swing-up and equilibrium similarly as for the results shown in Figure 6, the novel PSO-based NMPC controller exhibits an improvement in J of 12.07%.

Repeatability is tested by performing several trials with different constraints, as shown in Table VI. The novel PSO nonlinear controller shows consistently better performance, with a mean improvement in J of 8.73%.

E. Scalability and Constraints

The inverted pendulum considered in these simulation experiments is a non-trivial example of a control engineering problem. It is characterized by nonlinear dynamics of an

TABLE IV
ACTUAL PLANT AND MODEL PARAMETERS (FOR A PARTICULAR TRIAL)

Parameter	Units	Actual Plant	Model
M	Kg	14.6	15.33
m	Kg	7.3	6.935
$2l$	m	2.4	2.52
b	Kg/s	14.6	15.33
h	Kgm^2/s	0.0136	0.0129

TABLE V
TRUE PERFORMANCE INDEX VALUES OBTAINED FOR THE ROBUSTNESS TEST

Method	Numerical Linearization (Quadratic Programming) [19]	PSO
True Performance Index J ($\times 10^6$)	2.7369	**2.4065**

TABLE VI
SIMULATION RESULTS FOR DIFFERENT CONSTRAINTS (10 INDEPENDENT TRIALS WITH CONSTANT R)

Constraint	Numerical Linearization (Quadratic Programming) [19]	PSO
$-30N \leq U \leq 30N$	2.7182	**2.4026**
$-35N \leq U \leq 35N$	2.5374	**2.3947**
$-40N \leq U \leq 40N$	2.4590	**2.3880**
$-45N \leq U \leq 45N$	2.8534	**2.4316**

overall order of four, dynamic interaction between the state variables, and under-actuation (one control input and two controlled outputs). Nevertheless, there do exist more complex plants having higher order and/or more inputs and outputs which would require a scale up of the proposed PSO-based controllers to higher dimensions. Current literature indicates that the PSO's performance remains robust even when applied to relatively high dimensions as typically found in control applications (in the order of tens). Indicative of this is Engelbrecht's work [63], which reveals the *gbest* PSO algorithm's superior performance with respect to all other homogeneous PSO algorithms considered therein. This is investigated for several benchmark unimodal and multimodal functions, and the issue of scalability for the *gbest* PSO algorithm starts becoming increasingly pronounced only for dimensions of order hundred or above. Following this empirical analysis, the heterogeneous PSO algorithm proposed by Engelbrecht in [64] is shown to be significantly more scalable to higher dimensions when compared with other algorithms [63], [65]. The foregoing analysis is further confirmed by Piccand *et al.* [66], who show how the *gbest* PSO algorithm employed in this paper exhibits a unity success rate for all the benchmark unimodal and multimodal functions considered therein, up to several tens of dimensions. Increasing the swarm size may sometimes be necessary to handle higher dimensions, however such tuning is also problem dependent [66]. In view of this published evidence, we envisage that for systems governed by increased state variables and/or inputs and outputs, the proposed PSO controllers are not expected to perform less

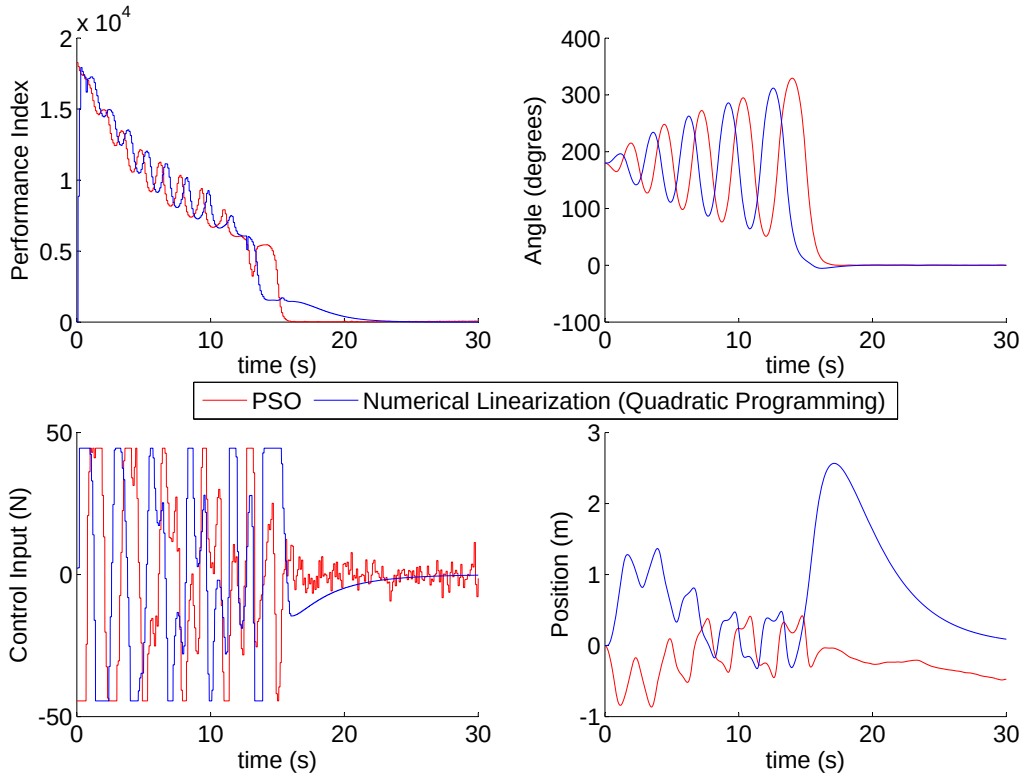

Fig. 7. Actively controlling control input weight R for reduced chattering.

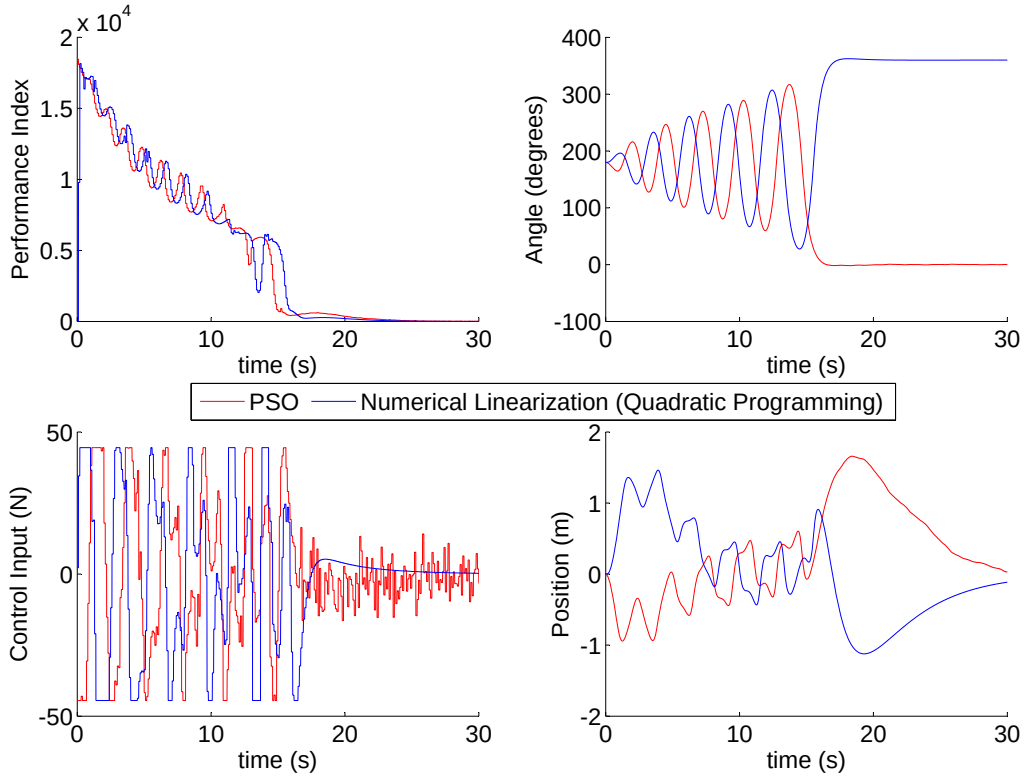

Fig. 8. Robustness Test: Model's parameters are significantly different from the actual plant parameters (constrained NMPC problem results shown).

reliably than the presented example. The process will necessarily be more computationally expensive, however this is an implementation issue which goes beyond the current scope of this work.

This paper has only considered the control input constrained problem for the NMPC case. Implementing PSO-based controllers for both control and output constraints, as given by equations (12) and (13), would require an elegant constraint handling approach. As with most applications of PSO, this could include the use of penalty methods to penalize those particles that violate constraints [67]–[69]. Other solutions, as reported by Shi and Krohling [70] and Laskari *et al.* [71], convert the constrained problem to an unconstrained Lagrangian. Also, repair methods, which allow particles to roam into feasible space, may be implemented by applying repairing operators to change infeasible particles to represent feasible solutions. A few examples include the work of Hu and Eberhart [72] that developed an approach where particles are not allowed to be attracted by infeasible particles, and that of El-Gallad *et al.* [73], which replaced infeasible particles with their feasible personal best positions. Other works by Venter and Sobieszczanski-Sobieski [74], [75] proposed a way of repairing infeasible solutions. Such methods may be investigated to determine the most efficient way of dealing with both input and output constraints.

V. Conclusion and Future Work

This paper has addressed the use of PSO for the design of model predictive control as applied to nonlinear systems. Following a detailed description of PSO and MPC theory, two novel controllers were proposed for the receding horizon model predictive strategy when applied to nonlinear dynamic systems. Both controllers exploit the well-known desirable properties of PSO. One makes use of a numerical linearization technique where, instead of convex optimization methods, we employed a PSO strategy. As expected, when simulated on an inverted pendulum on cart problem, this technique only yielded a minor improvement in performance over its convex optimization counterpart. By contrast, the second proposed controller proved superior to both, approaching up to 16% less performance cost at best. In addition, we proposed a further enhancement for this novel scheme by actively controlling the control input weight $\mathbf{R}$ to reduce the chattering effect of the control signal that is often observed in nonlinear model predictive control. This framework was also shown to be extensible to input constrained systems, thereby providing a foundation to include other advances in control theory as they become available.

This work may be further extended by an investigation on the use of PSO to obtain the much needed connection between the selection of $\mathbf{Q}$ and $\mathbf{R}$ (the two weighting matrices) and the performance specifications; possibly through some time-domain performance criterion. A similar investigation may be carried out for other control schemes, including linear quadratic optimal control strategies. Another interesting idea for future work is to run two optimizers in parallel. One strategy could employ a random search algorithm [76]–[78] running in parallel with PSO. Their possible collaboration could yield the right answer faster, whilst also being more robust to pathological cases. Furthermore, different variations of the PSO algorithm could be implemented, comparing their performance in the process, and also addressing scalability issues.

Having successfully implemented PSO for an NMPC problem, one should note that in certain circumstances when the problem is very well known, gradient-based methods may solve the NMPC problem more efficiently than PSO, albeit not as accurately. This calls for a detailed comparison between a PSO-based implementation and, for instance, one using a state-of-the-art NMPC solver such as the SQP-based method in ACADO [79], [80], or ICLOCS in combination with the interior point solver IPOPT [81]–[83].

All these issues may be investigated in future work for the purpose of establishing more effective ways of optimizing the NMPC problem.

References

[1] J. Mercieca and S. G. Fabri, "Particle swarm optimization for nonlinear model predictive control," in *Proceedings of the Fifth International Conference on Advanced Engineering Computing and Applications in Science - ADVCOMP 2011*, Lisbon, Portugal, November 2011, pp. 88–93.

[2] B. Anderson and J. Moore, *Optimal control: linear quadratic methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990.

[3] M. Dahleh and J. Pearson, J., "I-optimal feedback controllers for mimo discrete-time systems," *IEEE Transactions on Automatic Control*, vol. 32, no. 4, pp. 314–322, Apr 1987.

[4] J. Doyle, K. Glover, P. Khargonekar, and B. Francis, "State-space solutions to standard H_2 and H_∞ control problems," *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 831–847, Aug 1989.

[5] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. Mishchenko, *The mathematical theory of optimal processes (International series of monographs in pure and applied mathematics)*. Interscience Publishers, 1962.

[6] R. Bellman, "On the Theory of Dynamic Programming," in *Proceedings of the National Academy of Sciences*, vol. 38, 1952, pp. 716–719.

[7] L. D. Berkovitz and N. G. Medhin, *Nonlinear Optimal Control Theory*, ser. Applied Mathematics and Nonlinear Science Series. Chapman & Hall/CRC, 2012.

[8] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practice - a survey," *Automatica*, vol. 25, pp. 335–348, May 1989.

[9] J. Richalet, "Industrial applications of model based predictive control," *Automatica*, vol. 29, no. 5, pp. 1251–1274, 1993.

[10] J. Maciejowski, *Predictive control: with constraints*. Prentice-Hall, Harlow, UK, 2002.

[11] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[12] C. Cutler and B. Ramaker, "Dynamic matrix control-a computer control algorithm," in *Proceedings of the Joint Automatic Control Conference*, 1980.

[13] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[14] D. Mayne, J. B. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000.

[15] E. F. Camacho and C. A. Bordons, *Model Predictive Control in the Process Industry*. Secaucus, NJ, USA: Springer-Verlag, New York, 1997.

[16] D. W. Clarke, *Advances in Model-Based Predictive Control*. Oxford University Press, 1994.

[17] K. R. Muske and J. B. Rawlings, "Model predictive control with linear models," *AIChE Journal*, vol. 39, no. 2, pp. 262–287, 1993.

[18] S. Shin and S. Park, "GA-based predictive control for nonlinear processes," *Electronics Letters*, vol. 34, no. 20, pp. 1980–1981, Oct 1998.

[19] A. Alaniz, "Model predictive control with application to real-time hardware and a guided parafoil," Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA, 2004.

[20] X. Blasco, M. Martinez, J. Senent, and J. Sanchis, "Generalized predictive control using genetic algorithms (GAGPC): an application to control of a non-linear process with model uncertainty," in *Methodology and Tools in Knowledge-Based Systems*, ser. Lecture Notes in Computer Science. Springer, 1998, pp. 428–437.

[21] T. Kawabe and T. Tagami, "A real coded genetic algorithm for matrix inequality design approach of robust PID controller with two degrees of freedom," in *Proceedings of the 1997 IEEE International Symposium on Intelligent Control*, Jul 1997, pp. 119–124.

[22] R. Krohling, H. Jaschek, and J. Rey, "Designing PI/PID controllers for a motion control system based on genetic algorithms," in *Proceedings of the 1997 IEEE International Symposium on Intelligent Control*, Jul 1997, pp. 125–130.

[23] P. Angeline, "Using selection to improve particle swarm optimization," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence.*, May 1998, pp. 84 –89.

[24] R. Krohling and J. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 78–82, Feb 2001.

[25] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence*. Piscataway, NJ, USA: IEEE Press, 1995.

[26] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming VII*, ser. EP '98. London, UK: Springer-Verlag, 1998, pp. 611–616.

[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, nov/dec 1995, pp. 1942–1948.

[28] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings on Evolutionary Computation, IEEE World Congress on Computational Intelligence.*, May 1998, pp. 69–73.

[29] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1232–1239, Nov 2000.

[30] D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 3, pp. 771–779, March 2004.

[31] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *46th AIAA, ASME, ASCE, AHS, ASC Structures, Structural Dynamics and Materials Conference*, Austin, USA, April 18-21 2005.

[32] M. R. AlRashidi and M. E. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 4, pp. 913 –918, aug. 2009.

[33] R. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, 2001, pp. 81–86.

[34] X. Hu, Y. Shi, and R. Eberhart, "Recent advances in particle swarm," in *Congress on Evolutionary Computation, CEC2004*, vol. 1, June 2004, pp. 90–97.

[35] Y. del Valle, G. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, April 2008.

[36] M. Wachowiak, R. Smolikova, Y. Zheng, J. Zurada, and A. Elmaghraby, "An approach to multimodal biomedical image registration utilizing particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 289–301, June 2004.

[37] L. Messerschmidt and A. Engelbrecht, "Learning to play games using a PSO-based competitive learning approach," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 280–288, June 2004.

[38] N. Franken and A. Engelbrecht, "Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 562–579, Dec. 2005.

[39] X. Li and A. P. Engelbrecht, "Particle swarm optimization: an introduction and its recent developments," in *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, ser. GECCO '07. New York, USA: ACM, 2007, pp. 3391–3414.

[40] C. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, June 2004.

[41] Z.-L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *Power Systems, IEEE Transactions on*, vol. 18, no. 3, pp. 1187 – 1195, aug. 2003.

[42] B. Zhao, C. X. Guo, and Y. J. Cao, "Improved particle swam optimization algorithm for OPF problems," in *Power Systems Conference and Exposition, 2004. IEEE PES*, oct. 2004, pp. 233 – 238 vol.1.

[43] C.-M. Huang, C.-J. Huang, and M.-L. Wang, "A particle swarm optimization to identifying the armax model for short-term load forecasting," *Power Systems, IEEE Transactions on*, vol. 20, no. 2, pp. 1126 – 1133, may 2005.

[44] J.-B. Park, K.-S. Lee, J.-R. Shin, and K. Y. Lee, "A particle swarm optimization for economic dispatch with nonsmooth cost functions," *Power Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 34 – 42, feb. 2005.

[45] W. Zhang and Y. Liu, "Reactive power optimization based on PSO in a practical power system," in *Power Engineering Society General Meeting, 2004. IEEE*, june 2004, pp. 239 – 243 Vol.1.

[46] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 1, pp. 58 –73, feb 2002.

[47] G. Coath and S. Halgamuge, "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 4, dec. 2003, pp. 2419 – 2425.

[48] A. I. El-Gallad, M. E. El-Hawary, and A. A. Sallam, "Swarming of intelligent particles for solving the nonlinear constrained optimization problem," *International Journal of Engineering Intelligent Systems*, vol. 9, no. 3, pp. 155–163, 2001.

[49] K. Yasuda, A. Ide, and N. Iwasaki, "Stability analysis of particle swarm optimization," in *Proceedings of The Fifth Metaheuristics International Conference*, 2003, pp. 341–346.

[50] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th IEEE International Symposium on Micro Machine and Human Science (MHS '95)*, Nagoya, Japan, October 1995, pp. 39–43.

[51] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*, 1st ed., ser. The Morgan Kaufmann Series in Evolutionary Computation. San Francisco, USA: Morgan Kaufmann, 2001.

[52] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. Wiley Publishing, 2007.

[53] R. Eberhart, P. Simpson, and R. Dobbins, *Computational intelligence PC tools*. San Diego, CA, USA: Academic Press Professional, Inc., 1996.

[54] M. Omran, A. Salman, and A. Engelbrecht, "Image classification using particle swarm optimization," in *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning*, 2002, pp. 370–374.

[55] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming VII*, ser. EP '98. London, UK: Springer-Verlag, 1998, pp. 591–600.

[56] H. Yoshida, Y. Fukuyama, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control in electric power systems considering voltage security assessment," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 6, 1999, pp. 497–502.

[57] P. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of the Congress on Evolutionary Computation CEC 99.*, vol. 3, 1999, pp. 3 vol. (xxxvii+2348).

[58] A. Ratnaweera, S. Halgamuge, and H. Watson, "Particle swarm optimization with self-adaptive acceleration coefficients," in *Proceedings of the First International Conference on Fuzzy Systems and Knowledge Discovery*, 2003, pp. 264–268.

[59] S. Naka, T. Genji, T. Yura, and Y. Fukuyama, "Practical distribution state

estimation using hybrid particle swarm optimization," in *IEEE Power Engineering Society Winter Meeting*, vol. 2, 2001, pp. 815–820.

[60] J. J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1991.

[61] The MathWorks, Inc. (2012, Dec) Simulink - simulation and model-based design. [Online]. Available: http://www.mathworks.com/products/simulink/

[62] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, USA: Cambridge University Press, 2004.

[63] A. P. Engelbrecht, "Scalability of a heterogeneous particle swarm optimizer," in *Swarm Intelligence (SIS), 2011 IEEE Symposium on*, april 2011, pp. 1 –8.

[64] A. Engelbrecht, "Heterogeneous particle swarm optimization," in *Swarm Intelligence*, ser. Lecture Notes in Computer Science, vol. 6234. Springer Berlin Heidelberg, 2010, pp. 191–202.

[65] B. J. Leonard and A. P. Engelbrecht, "Scalability study of particle swarm optimizers in dynamic environments," in *Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, C. Blum, A. L. Christensen, A. P. Engelbrecht, R. Gross, and T. Stützle, Eds., vol. 7461. Springer Berlin Heidelberg, 2012, pp. 121–132.

[66] S. Piccand, M. O'Neill, and J. Walker, "On the scalability of particle swarm optimisation," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, june 2008, pp. 2505 –2512.

[67] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," in *Intelligent Technologies – Theory and Applications: New Trends in Intelligent Technologies*. IOS Press, 2002, pp. 214–220.

[68] V. Tandon, H. El-Mounayri, and H. Kishawy, "NC end milling optimization using evolutionary computation," *International Journal of Machine Tools and Manufacture*, vol. 42, no. 5, pp. 595–605, 2002.

[69] F. Zhang and D. Xue, "An optimal concurrent design model using distributed product development life-cycle databases," in *Sixth International Conference on Computer Supported Cooperative Work in Design*, 2001, pp. 273–278.

[70] Y. Shi and R. Krohling, "Co-evolutionary particle swarm optimization to solve min-max problems," in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC '02.*, vol. 2, 2002, pp. 1682–1687.

[71] E. Laskari, K. Parsopoulos, and M. Vrahatis, "Particle swarm optimization for integer programming," in *Proceedings of the Congress on Evolutionary Computation, CEC '02.*, vol. 2, 2002, pp. 1582–1587.

[72] X. Hu and R. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

[73] A. El-Gallad, M. El-Hawary, A. Sallam, and A. Kalas, "Enhancing the particle swarm optimizer via proper parameters selection," in *Canadian Conference on Electrical and Computer Engineering, IEEE CCECE*, vol. 2, 2002, pp. 792–797.

[74] G. Venter and J. Sobieszczanski-Sobieski, "Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization," *Structural and Multidisciplinary Optimization*, vol. 26, pp. 121–131, 2004.

[75] G. Venter and J. Sobieszczanski-Sobieski, "Particle swarm optimization," *Journal for the American Institute of Aeronautics and Astronautics*, vol. 41, no. 8, pp. 1583–1589, 2003.

[76] J. C. Spall, "Stochastic optimization," in *Handbook of Computational Statistics*, J. Gentle, W. Härdle, and Y. Mori, Eds. Springer-Verlag, New York, 2004, ch. II.6, pp. 169–197.

[77] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Review*, vol. 45, pp. 385–482, 2003.

[78] D. C. Karnopp, "Random search techniques for optimization problems," *Automatica*, vol. 1, pp. 111–121, 1963.

[79] B. Houska and H. J. Ferreau. (2012, Dec) ACADO toolkit: Automatic control and dynamic optimization. [Online]. Available: http://www.mathworks.com/products/simulink/

[80] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit - an open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[81] Paola Falugi, Eric Kerrigan and Eugene van Wyk. (2012, Dec) Imperial College London, optimal control software (ICLOCS). [Online]. Available: http://www.ee.ic.ac.uk/ICLOCS/

[82] A. Wächter, "An interior point algorithm for large-scale nonlinear optimization with applications in process engineering," Ph.D. dissertation, Carnegie Mellon University, 2002.

[83] A. Wächter and L. Biegler. (2012, Dec) IPOPT - an interior point optimizer. [Online]. Available: https://projects.coin-or.org/Ipopt