Stabilization of an active magnetic bearing using a two-step fast nonlinear predictive controller

Stéphane Bonnet, Jérôme De Miras and Borislav Vidolov
HeuDiaSyC laboratory UMR C.N.R.S. 6599
Université de Technologie de Compiègne, BP20529, 60205 Compiègne CEDEX, France
Email: {bonnetst, demiras, bvidolov}@hds.utc.fr

***Abstract*—This paper deals with the stabilization of a magnetic levitating shaft using a simple, fast, nonlinear predictive control approach. The proposed control approach uses an approximate numerical one-step discretization of the nonlinear plant model obtained from offline simulations. Using that discretization, a control minimizing the distance between the plant output and a reference linear system is computed, leading the system to adopt its dynamical behavior. Since the prediction horizon is limited to one time-step, the execution time of the algorithm can be completely bounded. It can thus easily be implemented and used to control fast electromechanical systems. Experimental results obtained from a laboratory device show the performance and robustness of the proposed controller.**

I. INTRODUCTION

Non-contacting magnetic bearings are magnetic levitation devices that are increasingly popular, both in academic and industrial communities. They have many well developed applications, the one receiving the most publicity certainly being MAGLEV trains. However, industrial applications that are less spectacular but more widespread form the bulk of their use. Indeed, electromagnets can be used to levitate rotating shafts in high speed machinery such as turbines, machine tools, vacuum pumps, compressors or flywheel inertial energy storage systems [1] in domains ranging from aerospace to biomedical equipment. These magnetic bearings are demonstrating numerous advantages when compared to traditional, mechanical bearings: as they are contactless devices, friction is low and predictable, they can work at virtually limitless shaft speeds and do not need lubrication, making them suitable for operation in environments that exclude any contamination or are adverse to proper lubrication, such as a complete vacuum.

However, building stable, static magnetically positioned systems using only permanent ferromagnetic magnets, according to EARNSHAW's theorem, is impossible. While passive solutions involving the use of diamagnetic materials, such as supraconductors, are being investigated [2], they are still relatively underdeveloped. Those limitations mean that most bearings are electromagnetic devices requiring active control systems in order to stabilize the shaft [3], known as active magnetic bearings (AMB).

The basic operation of an AMB is straightforward (Fig. 1). Each control axis is fitted with two electromagnets and a gap sensor that measures the rotor displacement. The electromagnets generate a force proportional to the square of the current passing through them and inversely proportional to the square of the rotor-stator air-gap. By controlling those forces, it is possible to steer the position of the shaft along the control axis. It can be stabilized along two degrees of freedom by combining two axes to form a single control plane. However, owing to the nature of the magnetic forces, the model of an axis is highly nonlinear and can lead to many complexities in control synthesis. Furthermore, since AMBs are fast electromechanical systems, considering computational issues is essential when it comes to implementing real-time control strategies [4]. Therefore, extensive studies have been first carried out on linear modeling and control, such as state-space and transfer approaches, $\mathcal{H}_\infty$ control [5], μ-synthesis [6], LQ control [4] or discrete dynamical programming [7] based on linear models. While model predictive control (MPC) is increasingly popular [8], its use on fast electromechanical systems has been held back by its computationally intensive nature: it usually entails solving a complex online dynamical optimization problem over a finite prediction horizon at each control time step [9]. However, the ever increasing computing power available and maturation of linear MPC theory has enabled its use on magnetic suspension devices, still using linear models, as demonstrated in [10]. Yet, in recent years, controllers based on nonlinear models have gained a lot of momentum. Indeed, by using elements of the model neglected by linear controllers, they usually exhibit better precision and efficiency levels. One of the most common approach used is feedback linearization combined with robust control techniques [11], but other nonlinear control techniques such as fuzzy control [12], sliding mode control [13], [14] or input-output linearization [14] have been studied. For the same reasons, nonlinear model predictive control (NMPC) has been recently getting a lot of attention [15]. However, being even more computationally intensive than linear MPC, its use is usually restricted to relatively slow systems, such as industrial

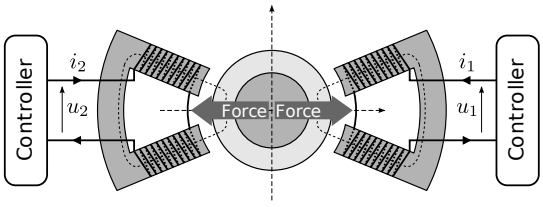

Figure 1. An AMB axis.

Preprint submitted to The European Control Conference 2009.
Received November 2, 2008.

processes [16].

In the paper, we focus on the design of a discrete-time, state-space, nonlinear predictive controller to stabilize each AMB axis of a single plane, using the coil currents as the control inputs. Couplings are dealt with as external unknown perturbations. Since the computational burden gets heavier as the prediction horizon increases, we will use the smallest horizon possible: one time step. By doing so, we foreclose the ability of the controller to handle explicit constraints in favor of getting a fast controller. However, it still has to be robust towards both prediction and modelling errors and external perturbation forces. We are going to first describe the nonlinear model used, then the control algorithm, and finally experimental results obtained on a laboratory AMB, showing the robustness of the proposed approach.

II. AMB MODELLING

As all the axes are similar, since we ignore couplings, only one axis has to be modelled for use by the controller. Hence, we can focus on obtaining a nonlinear state-space model of the y axis for instance, and use it for every axis. Two possible control inputs that can be used for that axis: either the coils currents i_{yp} and i_{ym}, or the coils excitation voltages E_{yp} and E_{ym}. In the latter case, a third order model is needed, as the coil current dynamics have to be taken into account. Since our experimental platform is fitted with high-gain current controllers, we can consider the coil currents as control inputs, making both the model and controller easier to build. However, the proposed controller can also be used with excitation voltages as control inputs, as simulations show in [17].

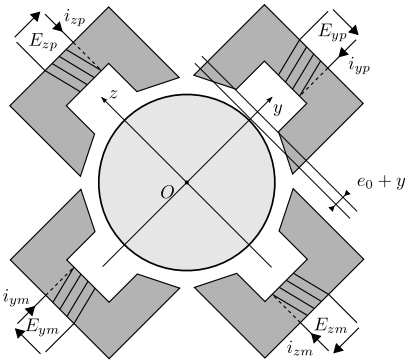

Figure 2. Control plane featuring two perpendicular axes

The model needed for control synthesis does not have to be exact, as modeling errors are taken care of as external perturbation. However, if they are known, incorporating constant perturbations such as gravity into the synthesis model from the start is a good idea. Let $\mathbf{y} = [y, \dot{y}]^{\mathrm{T}}$ be the state vector of the y axis (Fig. 2), with y the shaft position and $\dot{y}$ its speed. Its acceleration can be written as:

$$m\ddot{y} = F_1 - F_2 + F_p, \tag{1}$$

where F_1, F_2 are the electromagnetic forces generated by the coils, and F_p a constant, additive perturbation force such as gravity. Neglecting saturation and hysteresis effects, we can write:

$$F_1 = \frac{\lambda_1 {i_{yp}}^2}{2(e_0 - y)^2} \text{ and } F_2 = \frac{\lambda_2 {i_{ym}}^2}{2(e_0 + y)^2}, \tag{2}$$

where e_0 is the nominal air-gap of the axis and where λ_1 and λ_2 depend on the geometry of the coils and the shaft. As each axis is made of two symmetrical actuators, we can consider them both equal to λ_y. If we combine (1) and (2), we obtain a model that is not linearizable when the shaft is at the origin and the currents equal zero [14]. Traditionally, in order to obtain a linear model suitable to the application of linear control theory, a physical linearization is performed by introducing a bias current I_0 into the coils. That premagnetization current creates a constant magnetic flux in both actuators. The flux build-up time is thus eliminated and a nearly linear response on magnetic flux is obtained by varying the inputs currents around that value as the shaft makes small displacements along the axis.

However, using a bias current is quite energy-consuming since both coils are always active. It is thus more efficient to base the control on the nonlinear model where only one coil is active at any given time. In this mode of operation, currents i_{yp} and i_{ym} are mutually exclusive. They can be replaced by a single control input i_y as follows:

$$i_{ym} = \begin{cases} -i_y & \text{if } i_y < 0 \\ 0 & \text{otherwise} \end{cases}, i_{yp} = \begin{cases} i_y & \text{if } i_y > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Rewriting (2) with that variable change, we obtain:

$$F_1 = -F_2 = \frac{\lambda_y \operatorname{sgn}(i_y) {i_y}^2}{2(e_0 - \operatorname{sgn}(i_y) y)^2}. \tag{4}$$

Since (3) implies that at any given time, either F_1 or F_2 vanishes, the model used to build the controller can be simplified and becomes:

$$\ddot{y} = \frac{\lambda_y \operatorname{sgn}(i_y) {i_y}^2}{2m(e_0 - \operatorname{sgn}(i_y) y)^2} + \frac{1}{m} F_p. \tag{5}$$

III. PROPOSED CONTROL SCHEME

A. Problem formulation

The basic idea behind the proposed discrete-time control strategy is to use the state of a given stable linear system as a reference for the output of the plant. That way, if the plant output trajectory is able at each time step to match the reference system state, it will have the same dynamics and converge to the origin. In the following, vectors and matrices are denoted in bold font.

Let a nonlinear, time-invariant model of the plant of the form:

$$\begin{cases} \dot{\mathbf{s}}(t) &= f(\mathbf{s}(t), u(t)) \\ v(t) &= h(\mathbf{s}(t)) \end{cases} \tag{6}$$

where $\mathbf{s} \in \mathcal{S} \subset \mathbb{R}^n$ is the plant state vector, $u \in \mathcal{U} \subset \mathbb{R}$ its scalar input control, and $v \in \mathbb{R}$ its output. Suppose now the following discrete-time prediction map obtained from that model exists:

$$\mathbf{s}_{k+1} = p(\mathbf{s}_k, u_k) \tag{7}$$

Preprint submitted to The European Control Conference 2009.
Received November 2, 2008.

where the function g is simply the sample-and-hold discretization of the state-function f in (6) using a time-step of duration Δt such as $t_k = k \cdot \Delta t$. In the general case, obtaining an exact analytic expression for g is difficult. However, its approximate values for a given state and input vector can usually be computed through numerical integration of f. Suppose now the following stable, homogen linear system:

$$\dot{\mathbf{w}}(t) = \mathbf{A} \cdot \mathbf{w}(t), \mathbf{w} \in \mathbb{R}^m, \mathbf{A} \in \mathcal{M}^{m \times m}. \quad (8)$$

We want the plant output to converge to a given set point $\bar{v}$. If we write

$$\mathbf{w} = \left[w(t) - \bar{v}, dw(t)/dt, \cdots, d^{m-1}w(t)/dt^{m-1}\right]^{\mathrm{T}},$$

from (8) the trajectory of $w(t)$ will also converge towards that point with a dynamic of order m. Let

$$\mathbf{v} = \left[v(t), dv(t)/dt, \cdots, d^{m-1}v(t)/dt^{m-1}\right]$$

be the vector of the plant output and its $m-1$ successive derivatives. The objective of the controller is to find at each time-step k a constant control u_k that will make the next-step plant output $\mathbf{v}_{k+1}$ be as close of $\mathbf{w}_{k+1}$ as possible, with $\mathbf{w}_{k+1}$ obtained from the discretization of (8) as follows:

$$\mathbf{w}_{k+1} = e^{-\mathbf{A}\cdot\Delta t}\left(\mathbf{v}_k - [\bar{v}, 0, \cdots]^{\mathrm{T}}\right) + [\bar{v}, 0, \cdots]^{\mathrm{T}}. \quad (9)$$

The number $m-1$ of output derivatives to take depends on the order of the model (6), but it is essential that the reference state trajectories and the output trajectory share dynamics of the same order. At each time step, the plant output vector $\mathbf{v}_{k+1}$ has to be computed. That means that in addition to the state predictor, we need an output prediction map q such as:

$$\mathbf{v}_{k+1} = q(\mathbf{s}_k, u_k). \quad (10)$$

Finally, computing the control input at each time step can be formally written as solving the following Euclidean norm minimization problem:

$$u_k = \arg\min_{u \in \mathcal{U}} \|\mathbf{w}_{k+1} - q(\mathbf{s}_k, u)\|_2. \quad (11)$$

B. Online control algorithm

We suppose that at step k, an estimation $\hat{\mathbf{s}}_k$ of the plant state is available, and that its output v_k is measured, allowing to compute the vector $\mathbf{v}_k$. We also make the hypotheses that the control input computation is instantaneous, and that the state estimation does not introduce any delays: from the measurements at step k, the control input u_k can be readily computed and immediately applied to the plant. Since theses hypotheses generally do not hold, section III-D below will show how to handle those issues.

The aim of the control algorithm is to compute an approximate solution to (11), as analytical forms for p and q are not known in general. We still suppose that specific approximate values of theses functions can be computed at any point within the continuous set $\mathcal{S} \times \mathcal{U}$. The algorithm can be split into four main parts:

1) State estimation and output vector computation, supposed available,
2) Prediction error estimation,
3) Reference point calculation, as shown above in (9)
4) Control input approximation.

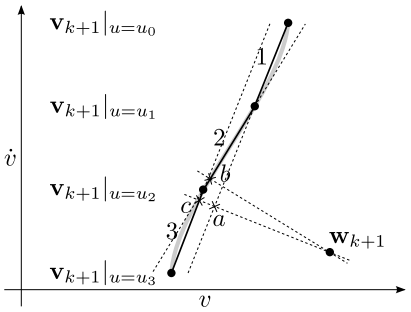

Figure 3. Approximate distance minimisation algorithm

The prediction maps are based on a model of the system. As precise as it it may be, it necessarily contains approximations that make the prediction maps inherently unreliable. In addition to modelling approximations, external perturbations that are not modelled at all also lead to incorrect predictions: they have to be corrected. Fundamentally, the prediction error is simply the difference between the predicted plant state and outputs for step k obtained from data at step $k-1$ and their estimations obtained from measurements at step k. We can define the error vectors $\varepsilon_{\mathbf{s}}$and $\varepsilon_{\mathbf{v}}$ for the state and output predictions, respectively, such as:

$$\varepsilon_{\mathbf{s}} = p(\mathbf{s}_{k-1}, u_{k-1}) - \mathbf{s}_k, \varepsilon_{\mathbf{v}} = q(\mathbf{v}_{k-1}, u_{k-1}) - \mathbf{v}_k.$$

Further predictions used in the algorithm to compute the control input are corrected used those two vectors, under the hypothesis that the error made at step k is close enough to that made at step $k-1$. In other words, the error dynamics is supposed slow relative to the controller. Since it may not be the case in noisy conditions for instance, where the measurements can vary a lot between two steps, the error signal is not used as is, but filtered to remove high frequency content, the main objective being to compensate for static errors or slowly varying perturbations. Trying to compensate for high frequency errors can indeed lead to oscillations and controller instability. The error signal for the state prediction used at step k can thus be written as:

$$\varepsilon_k = (1-\alpha)\,\varepsilon_{k-1} + \alpha\,(p(\mathbf{s}_{k-1}, u_{k-1}) - \mathbf{s}_k)$$

with α a damping coefficient in $]0, 1]$, forming a numerical low-pass filter. The error signal for the output predictions is handled by the same method.

Since the expression for p and q are supposed unkown but we are able to perform numerical evaluations for each one, we will use an iterative algorithm to approximate the solution of 11. This algorithms starts by uniformly discretizing the input interval $\mathcal{U} = [\underline{u}, \overline{u}]$ to yield the input set

$$\mathcal{U}_d = \left\{u_i | u_i = \underline{u} + i\frac{\overline{u} - \underline{u}}{N}, i \in 0, \ldots, N\right\}$$

Preprint submitted to The European Control Conference 2009.
Received November 2, 2008.

and proceeds by building a piecewise linear approximation of the function $q(\mathbf{v}, u)_{\mathbf{v}=\mathbf{v}_k} = q_{\mathbf{v}_k}(u)$ and using a parametric orthogonal projection on each segment. The algorithm goes as follows:

1: $i \leftarrow 0$ ▷ Iteration counter
2: $P_i \leftarrow q_{\mathbf{v}_k}(u_i) - \varepsilon_k$ ▷ First point
3: $a_0 \leftarrow 1$ ▷ Coordinate of the projection on the segment
4: **repeat**
5: $i \leftarrow i + 1$
6: $P_i \leftarrow q_{\mathbf{v}_k}(u_i) - \varepsilon_k$ ▷ Second point
7: $a_i \leftarrow$PROJECTONSEGMENT$(P_{i-1}, P_i, \mathbf{v}_{k+1})$
8: **if** $a_i < 0$ **then**
9: **if** $a_{i-1} < 1$ **then**
10: $u \leftarrow (1 - a_{i-1})u_{i-1} + a_{i-1}u_i$
11: **else**
12: $u \leftarrow u_{i-1}$
13: **end if**
14: Terminate.
15: **else if** $a_i < 1$ **then**
16: **if** $a_{i-1} < 1$ **then**
17: $u \leftarrow (1 - a_{i-1})u_{i-2}a_{i-1}u_{i-1}$
18: $u \leftarrow (u + (1 - a_i)u_{i-1} + a_i u_i)/2$
19: Terminate.
20: **end if**
21: **end if**
22: **until** $i > N$
23: $u \leftarrow u_N$

The input control is computed according to the coordinate of the parameter along the segment. There are mutliple cases to handle. The first case, line 10 happens when the projection is not on the current segment, but may be on the previous segment. If it actually is, then the value obtained from that previous segment is returned. If it is not, the minimum value for the current segment is returned. The second case, lines 17 and 18, is more complex, and is illustrated figure 3. In this figure, the real function q can be seen in light gray, its piecewise linear approximation being the black thin line segments. In this example, projection of the reference point on the linear approximation yields valid solutions that are within the segments for segments 2 and 3, at the points b and c, respectively. It means that when a solution is found on a segment, the next segment has to also be examined, and if it contains a solution too, an average is returned. Finally, the third case, line 23 happens when there is no solution after every segment has been checked: the maximum input control value is returned.

Once the distance minimizing control input has been computed, it can be applied to the plant, ending the current step. While it only finds an approximate value, the exectution time of this algorithm can be bounded, and depends only on the number $N + 1$ of input discretization points used. However, the error due to the linear approximations is proportional to the square of the discretization grid size: a trade-off has to be found between the minimization error and execution time. Another important factor is the time taken by the numerous evaluations of the prediction maps p and q.

C. Prediction maps

The prediction maps evaluations could be done each time they are needed through numerical integration of (6). However, this would be computationnally expensive, and it is difficult to bound the execution time of the usual integration algorithms, which is problematic in a real-time implementation. A better approach is to build a piecewise approximation of those maps, and use it each time an evaluation has to be done.

In order to build that piecewise approximation, a regular rectangular grid $\mathcal{G}$ is constructed on $\mathcal{S}\times\mathcal{U}$. That grid is a spatial discretization of the joint state and input spaces. Each point of $\mathcal{G}$ represent a possible initial state and a constant control input that can be applied to the model. For each point of the grid, the model equation is solved on one time-step with the given initial state and using the associated control input, using simulation tools such as Simulink. The solution vectors $\mathbf{s}$ and $\mathbf{v}$ are then stored in a table. The values of p and q at unknown points can be interpolated from that table, using barycentric linear interpolation as described in [18] for instance. While linear interpolation leads to errors proportional to the square of the grid size, its main advantage is the low computing effort needed to get interpolated values, hence the choice of that approach instead of more precise but less efficient methods such as spline interpolation.

D. Real-time implementation of the algorithm

In order to build a sound implementation of the previous algorithm, two issues are to be cared for:

- Only position sensors are available
- The control input computed at step k is only applied at step $k + 1$

The first step of the algorithm is to build an estimation of the state and compute the output derivatives at step k. In the case of an AMB axis, they are the same quantities, and for the y axis, we can write $\mathbf{s}_k = \mathbf{v}_k = [y_k, \dot{y}_k]^{\mathrm{T}}$ where y_k is the measured shaft position at step k. We can compute the shaft speed $\dot{y}$ from its position by using a centered difference of the following form:

$$\dot{y}_{k-1} \backsimeq \frac{y_k - y_{k-2}}{2\Delta t}. \tag{12}$$

The purpose of using a two-step estimator is to obtain a full, single Δt period delay on the speed estimation. If a single step was used, only a half-time-step delay would occur, which would be problematic for the predictor to handle. However, in that case (12) needs the position at step $k - 2$ to produce a speed estimation at the step $k - 1$.

Suppose the algorithm is being run at step k. The computations go as follows:

- compute the state measurement $\mathbf{v}_{k-1}$ from (12) and y_{k-1}:

$$\mathbf{v}_{k-1} = \left[y_{k-1}, \frac{y_k - y_{k-2}}{2\Delta t} \right]^{\mathrm{T}},$$

Preprint submitted to The European Control Conference 2009.
Received November 2, 2008.

- compute the error improvement using the prediction obtained for step $k-1$ knowing $k-2$:

$$\varepsilon_{k-1} = (1-\alpha)\,\varepsilon_{k-2} + \alpha\,(q\,(\mathbf{v}_{k-2}, i_{k-2}) - \mathbf{v}_{k-1})$$

- compute an estimation of the current state, accounting for prediction errors:

$$\hat{\mathbf{v}}_k = q(\mathbf{v}_{k-1}, i_{k-1}) - \varepsilon_{k-1}.$$

The problem now is to compute the control input to be applied to the plant at the start of the next control step, that is i_{k+1}. We suppose that the current control input being used is i_k. The state at the end of the current step k can thus be estimated as follows:

$$\hat{\mathbf{v}}_{k+1} = q(\hat{\mathbf{v}}_k, i_k) + \varepsilon_{k-1}.$$

Since the only error estimation available at the time is the error from the previous step, we still use the same value, under the hypothesis that the variation between ε_{k-1} and ε_k will be small. Using $\mathbf{v}_{k+1}$, the computation of the control is then carried out as shown at the end of section III-B. Note that every added prediction step increases the error-induced noise, and even though it can be damped by adjusting the α parameter, that fact can lead to a degradation of the controller performance.

IV. EXPERIMENTAL SETUP

The device used for experiments is a laboratory AMB test bench (Fig. 4) from MECOS-TRAXLER AG, model miniVS. It is is built around a magnetic suspension unit made of a rotor and a stator, one industrial PC featuring a Pentium IV processor running The MathWorks xPC Target real-time executive linked to a desktop computer running MATLAB and Simulink through an Ethernet network, a power electronics control interface, a signal conditioning interface, and a power supply. The magnetic suspension device is made of a rotor shaft driven by an electric motor along the x axis, and five magnetic bearings controlling the shaft position along the x, $y_{1,2}$ and $z_{1,2}$ and z. Those magnetic bearings are current-controlled through a digital-to-analog converter interface card from National Instruments that in turn drives the power electronics. They are fitted with magnetic transducers in charge of measuring the shaft position, and current-sensing devices. The sensor outputs are acquired by the industrial PC through an analog-to-digital converter card also made by National Instruments. The magnetic transducers precision is limited to $1\,\mu\mathrm{m}$ by the 12-bit ADC digital quantification.

In order to simplify the development of the real-time controller implementation, we are using The MathWorks xPC Target rapid prototyping system. The controller is described using Simulink on the desktop PC through a model mixing built-in blocks when possible and custom C language function blocks for the prediction and control input computation parts of the controller. Indeed, optimizing these parts make sense as they form the bulk of the control algorithm. After code generation and compilation stages, the control application is loaded on the industrial PC and run, using a $700\,\mu\mathrm{s}$ cycle time. Table I summarizes the experimental test bench parameters.

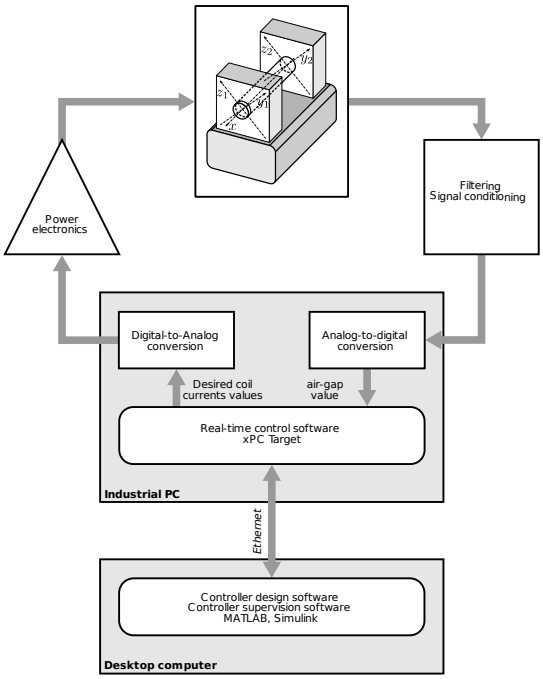

Figure 4. Experimental setup

A few implementation issues have to be taken into account to get a working controller. The main issue is the lack of shaft translational-speed sensing, which can be approximated by a centered difference as exposed above. However, the addition of several successive prediction steps and the speed approximation errors tend to introduce some noise into the error correction signal that can lead to oscillations and instability. That noise can be attenuated by choosing a small enough noise correction damping coefficient. Here we arbitrarily set $\alpha = .25$.

Table I
AMB PARAMETERS

Parameter	Value	Description
m	$3.097\,\mathrm{kg}$	Shaft mass
e_0	$0.4 \cdot 10^{-3}\,\mathrm{m}$	Nominal air-gap
$\lambda_{\{y,z\}}$	$1.2 \cdot 10^{-6}\,\mathrm{mH \cdot m}$	λ for axes y and z
$\dot{\phi}_{max}$	$30\,000\,\mathrm{rpm}$	Max. shaft angular speed
I_{max}	$6\,\mathrm{A}$	Max. coil current
V_{max}	$50\,\mathrm{V}$	Max. coil voltage
F_p	$m \times \sqrt{2}/2 \times 9.81\,\mathrm{N}$	Gravity for axes y and z

V. RESULTS

All the experiments conducted are based on tracking a reference trajectory. Each axis is stabilized using the a single implementation of the control strategy presented in section III. In order to make the implementation independent of the axis it is used on, the perturbation force due to gravity has been considered non existent. As it is obviously not the case on the experimental test bench for the axes y and z, that means

Preprint submitted to The European Control Conference 2009.
Received November 2, 2008.

that they are both subject to an unmodelled perturbation. The internal reference linear state-space system has an equivalent canonical second-order transfer function with a natural frequency of $w_0 = 200\,\mathrm{Hz}$ and a damping ratio $\zeta = 1.1$.

In the first experiment, the z-axis position is made to follow a $5\,\mathrm{Hz}$ square wave signal with an amplitude of $e_0/4$ around the origin. Fig. 5 shows the step response of the z axis (solid line) and the reference input (dashed line). It can be seen that even if control synthesis neglected gravity, the reference is followed and reached without any static error, showing the robustness of the controller to some external perturbations. Moreover, it can be noted that there is no overshoot. It is to be expected, as the internal reference system is damped and the shaft is made to effectively exhibit the same dynamics.

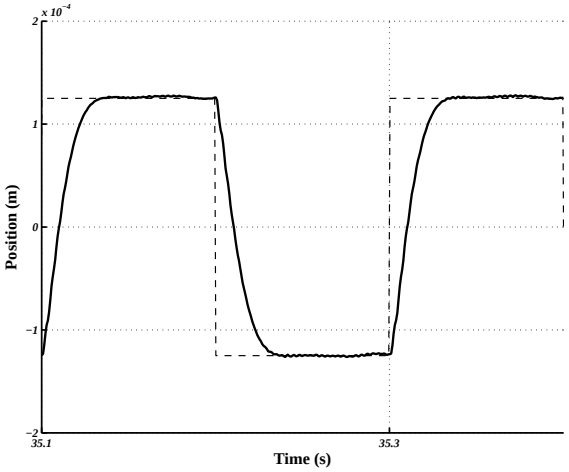

Figure 5. $5\,\mathrm{Hz}$ square wave tracking – z axis position

The second experiment makes the shaft go in circles by having two $2\,\mathrm{Hz}$ quadrature sinusoidal references on the y and z axes. Fig. 6 shows the temporal response of axis z (solid line) and its reference (dashed line), and Fig. 7 the circular shaft trajectory in the y-z plane. As the z position exhibits a light delay in following its reference, another experiment has been conducted with a $10\,\mathrm{Hz}$ sine wave to underline the phenomenon. There are three different curves on Fig. 8. The first one (solid line) is the shaft position. The second one (dashed) is the external tracking reference the position is supposed to follow. Finally, the third one, (dash-dotted) is the internal linear second-order reference system trajectory. It is interesting to note the shaft position actually follows quite well its internal reference, while there is a noticeable amplitude and phase error with the external reference trajectory. To explain that phenomenon, one has to remember the external trajectory is in fact used internally as an input to the internal reference system which then acts as a low-pass filter. One solution is simply to increase the internal reference system cutting frequency. However, it has to stay compatible with the dynamic capabilities of the physical device which can become unstable if it is unable to match the reference system dynamics. It can be also noted that there is a small delay between the internal reference and the shaft position at any given step. This is explained by the fact that the internal reference computed at step k is where the system should be at step $k + 2$, owing to the three successive predictions that have to be made to compute the control input. While is is not clear on the figure due to size constraints, that delay is exactly $1.4\,\mathrm{ms}$.

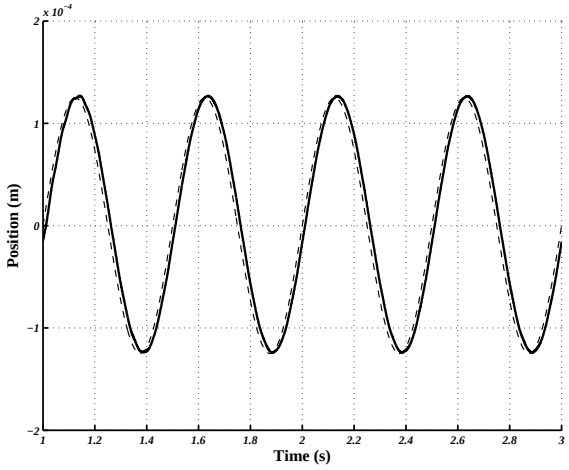

Figure 6. $2\,\mathrm{Hz}$ sinusoidal reference tracking – z axis

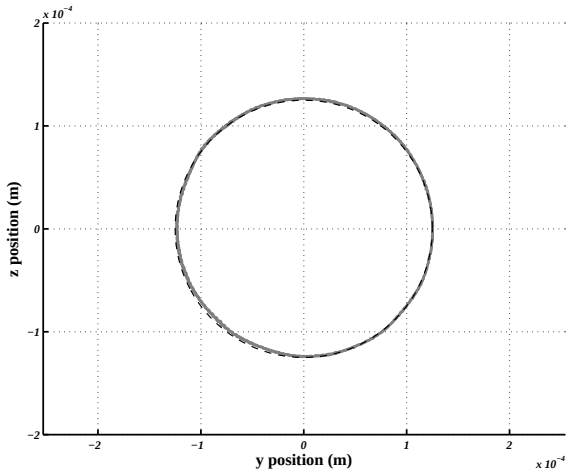

Figure 7. Quadrature sinusoidal reference tracking – y-z plot

VI. CONCLUSION

Through real experiments on a laboratory magnetic levitation shaft, a novel predictive nonlinear controller has been presented. Since the zero-bias behavior of such a device is highly nonlinear and its efficient control known to be a challenging task, it is a a good illustration of the proposed controller properties. That controller seems generic enough to be applied to other systems, while still maintaining those desirable properties. However, being based on heuristics, its convergence properties have not been studied. In particular, the error correction step is essential to the algorithm and needs further refinement. Whether a well-chosen reference system

Preprint submitted to The European Control Conference 2009.
Received November 2, 2008.

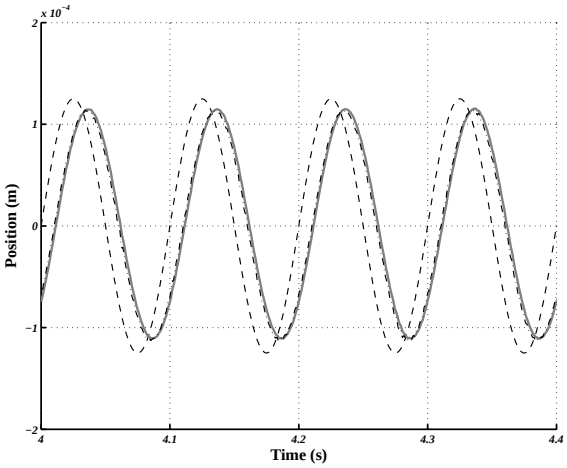

Figure 8. 10 Hz sinusoidal reference tracking – z axis

could be used to constrain the state and/or outputs within specified bounds remains to be investigated.

REFERENCES

[1] A. Chiba, D. G. Dorrell, T. Fukao, O. Ichikawa, M. Oshima, and M. Takemoto, *Magnetic Bearings and Bearingless Drives*. New York, NY, USA: Elsevier Science Publishers, March 2005.

[2] R. Moser, J. Sandtner, and H. Bleuler, "Diamagnetic suspension system for small rotors," *Journal of Micromechatronics*, vol. 1, no. 2, pp. 131–137, January 2001.

[3] G. Schweitzer, H. Bleuler, and A. Traxler, *Active Magnetic Bearings*. Zürich, Switzerland: V.d.f. edition, 1994.

[4] W. Grega and A. Pilat, "Comparison of linear control methods for an AMB system," *International Journal of Applied Mathematics and Computer Science*, vol. 15, no. 2, pp. 245–255, 2005.

[5] M. Fujita, F. Matsumara, and M. Shimizu, "$\mathcal{H}_\infty$ robust control design for a magnetic suspension system," in 2^{nd} *Int. Symp. on Magnetic Bearings*, Tokyo, Japan, 1990, pp. 349–356.

[6] K. Nonami and H. Yamaguchi, "μ-synthesis of a flexible rotor magnetic bearing system," in 4^{th} *International Symposium on Magnetic Bearings*, Zürich, Switzerland, 1994, pp. 73–78.

[7] H. F. Steffani, W. Hofmann, and B. Cebulski, "A controller for a magnetic bearing using the dynamic programming of Bellman," in 6^{th} *International Symposium on Magnetic Bearings*, Massachusets Institute of Technology, Cambridge, MA, USA, 1998.

[8] J. S. Qin and T. A. Badgewell, "An overview of industrial model predictive control technology," *Chemical Process Control*, vol. 93, no. 316, pp. 232–256, 1997.

[9] G. C. Goodwin, M. M. Seron, and J. A. De Doná, *Constrained Control and Estimation – An Optimisation Approach*, ser. Computer and Control Engineering. London: Springer–Verlag, 2005.

[10] J. Huang, L. Wang, and Y. Huang, "Continuous time model predictive control for a magnetic bearing system," *Progress In Electromagnetics Research Symposium Online*, vol. 3, no. 2, pp. 202–208, 2007.

[11] M. Chen and C. R. Knospe, "Feedback linearization of active magnetic bearings: current-mode implementation," *IEEE/ASME Trans. Mechatron.*, vol. 10, no. 6, pp. 632–639, December 2005.

[12] B. Vidolov, C. Melin, J. De Miras, and A. Charara, "Two-rules-based fuzzy logic control and sliding mode control of an active magnetic bearing," in *FUZZ-IEEE'96*, vol. 2, New Orleans, LA, USA, 1996, pp. 1205–1209.

[13] D. Cho, Y. Kato, and D. Spilman, "Sliding mode and classical controllers in magnetic levitation systems," *IEEE Control Syst. Mag.*, vol. 13, no. 1, pp. 42–48, 1993.

[14] A. Charara, J. De Miras, and B. Caron, "Nonlinear control of a magnetic levitation system without premagnetization," *IEEE Trans. Contr. Syst. Technol.*, vol. 4, no. 5, pp. 513–523, sep 1996.

[15] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," in 21^{st} *Benelux Meeting on Systems and Control*, Veldhoven, Holland, 2002.

[16] J. S. Qin and T. A. Badgewell, *An Overview of Nonlinear Model Predictive Control Applications*, ser. Progress in Systems and Control Theory. Basel, Switzerland: Birkhäuser-Verlag, 2000, vol. 26, pp. 369–391.

[17] S. Bonnet, J. De Miras, and B. Vidolov, "Nonlinear implicit control of a magnetic bearing without premagnetization," in *International Conference on Computational Intelligence, Control and Automation*, vol. 2. Vienna, Austria: IEEE Computer Society, 2005, pp. 432–437.

[18] R. Munos and A. Moore, "Barycentric interpolators for continuous space & time reinforcement learning," in *Proceedings of the 1998 conference on Advances in neural information processing systems II*. Cambridge, MA, USA: MIT Press, 1999, pp. 1024–1030.