Iterative Solution in Adaptive Model Predictive Control by Using Fixed-Point Transformation Method

Hamza Khan, Jozsef K. Tar, Imre J. Rudas, and Gyorgy Eigner

***Abstract*—** In this paper the application of a recently introduced simple iterative solution of the Nonlinear Programming approach is investigated for the Adaptive Model Predictive Control of a strongly nonlinear dynamic paradigm, the Duffing oscillator. The main idea is to replace the numerically much more complex Reduced Gradient method in the optimization task under constraints when the cost function has relatively simple structure, but it does not allow the use of the traditional LQR controller. The suggested iterative solution is based on Banach's Fixed Point Theorem that has twofold utilization: in approaching the solution of the optimization task, and in the realization of the adaptive behavior of the controller. In the presented, Julia language-based numerical simulations certain numerical tricks are also applied that were used to stabilize the run of the numerical simulations. The numerical examples illustrate the applicability of the suggested method for a wide class of cost function structures.

***Keywords*—**Nonlinear Programming, Model Predictive Control, Receding Horizon Controller, Adaptive Control, Fixed Point Transformation**.**

I. Introduction

IN the Nonlinear Programming (NP) one of the practical approaches is the Model Predictive Controller (MPC) [1, 2], that traditionally works in the framework of Optimal Control (OC). In general, the goal of an MPC application is approximate tracking of a nominal system trajectory in the possession of the usually approximate model of the controlled system under simultaneous, often contradictory restrictions. In case of an MPC, the actual value of the control signal u is obtained over a discrete time grid by solving an open-loop control problem on the finite horizon at each sampling instant. Such an approach can be applied in control tasks for which only a very approximately known model is available, e.g. in life sciences [3]. In this practical approach, a cost function can be minimized that is a weighted sum of nonnegative contributions expressing penalties from various points of view. The dynamics, i.e. the dynamical equations of motion of the controlled system mathematically are taken into consideration as a set of "constraint equations", and they play a significant role. The cost function normally contains terms that depend on the tracking error, the control signal itself, and optionally, a separate term that gives additional weight to the tracking error at the terminal point of the discrete horizon. It has its wide-spread applications in technical [4] and economic [5] as well as many other fields particularly in the control theory of nonlinear plants in traffic control [6], [7], chemistry [8], [9], life sciences-related problems [10], phenomena associated with web transportation systems [11] etc.

In general, the problem is very similar to finding the local extremum of the action functional in Classical Mechanics. In fact, the use of the so obtained Hamilton-Jacobi-Bellman Equations in control theory as the most general approach is considered too complicated, and the Dynamic Programming (DP) applied for solving them generally needs high computational power [12], [13]. An alternative practical approach is used to tackle the problem by calculating the variables in the discrete points of a finite time-grid which is considered as a "horizon" (Nonlinear Programming (NP)). The traditional Receding Horizon Controller (RHC) is one of the practical methods [14], [15] which works to diminish the effects of modeling imprecisions with finite horizon lengths, in the consecutive horizon-length cycles [16]. The optimization under constraints happens by NP where Lagrange's Reduced Gradient (RG) Method is implemented to proceed the calculation further [17]. In the special case of the Linear Time Invariant (LTI) system models and quadratic cost functions, the problem is significantly reduced: the so obtained Linear Quadratic Regulator (LQR) [18] technically can be realized over a finite horizon by solving the Riccati Differential Equations with a terminal condition for a matrix function that also influences the motion of the system state persisting to an initial condition. In more general cases such a clear separation of the variables cannot be realized, and one must work with Time-dependent Riccati Equations. (In the survey paper [19] several applications were discussed in connection with this problem.). The RG method can be numerically implemented in quite general cases. For not too substantial size of the problem, the *MS EXCEL's* Solver Package (provided by an external firm Frontline Systems, Inc.) yields an exceptional solution in combination with a little programming effort in Visual Basic (VB) in the background. The problem conveniently can be formulated by defining functional

The research was supported by the Doctoral School of Applied Informatics and Applied Mathematics of Obuda University, Budapest Hungary.

Hamza Khan is a PhD student in the Doctoral School of Applied Informatics and Applied Mathematics of Obuda University Budapest, Hungary, (0036204542884; e-mail: hamza.khan@phd.uni-obuda.hu).

Dr. hobil Jozsef K. Tar, is a senior full professor in Obuda University Budapest, Hungary. (e-mail: tar.jozsef@nik.uni-obuda.hu).

Dr. Imre J. Rudas is a Senior professor in the Institute of Applied Mathematics, Óbudai Egyetem. (e-mail: rudas@uni-obuda.hu).

Dr. Gyorgy Eigner is an Assistant Lecturer in Obuda University Budapest Hungary. He was supported by the n'UNKP-17-4/I. New National Excellence Program of the Ministry of Human Capacities.
(e-mail: eigner.gyorgy@nik.uni-obuda.hu).

relationships between the contents of the various cells of the worksheets (user-defined functions can be created in VB). Furthermore, for the Solver, a "model" can be specified by giving the cell that contains the cost to be minimized, the location of the independent variables and the constraints in the worksheets, and the parameter settings of this optimization package. The so defined "model" can be saved somewhere in one of the worksheets. Following that a small program can be written in VB that declares the model parameters as global variables, reads their actual values from the worksheets, loads the "model" for the Solver, and for the horizons under consideration cyclically:

1. writes the relevant data into the appropriate cells including the nominal trajectory to be tracked, the initial values of the variables to be optimized, and that of the control forces.
2. calls the Solver with the options that it must stop optimization if the prescribed limits in the time or step numbers have been achieved, and keep the so obtained results, and
3. writes the optimized results in certain cells of a worksheet.

In the preliminary step, the Solver attempts to find a common point on the constraint surfaces using the Newton-Raphson method [20]. Following that it computes the Reduced Gradient (RG) by calculation the appropriate Lagrange Multipliers, and realizes little steps in the direction of the RG. The algorithm stops when the RG takes zero. At this point, the constraints do not allow more improvement in the cost. The Solver package numerically computes the gradient values, can automatically set the suitable step lengths. The calculation of the Lagrange multipliers in principle needs the calculation of a quadratic matrix that generally may be singular or ill-conditioned, therefore somehow it must tackle these problems, too.

It is a reasonable expectation that this complicated procedure can be eluded in the control of a system class in which a) the cost functions contain separate differentiable contributions for penalizing the tracking error and the too big control effort, and b) the mathematical form of the system's model under control is ab ovo known. In this case, the appropriate gradients can be analytically calculated, and the EXCEL–VB programming background does not promise further suitability, especially if the RG algorithm can be replaced by a simpler one. This program is briefed in the next section.

II. The Basic Idea Of The Nonlinear Programming Approach

Consider the numerical approximation of the problem as follows. A dense enough discrete time-grid as $\{t_0, t_1 = t_0 + \Delta t, ..., t_{n+1} = t_n + \Delta t, ..., t_N\}$ is determined in which $"t_0"$ is considered the initial whereas $"t_N"$ indicates the final time instant of the considered motion. Let the equation of motion of the controlled system be $\dot{x} = f(x,u)$ where $x \in \mathbb{R}^m$ expresses the state variable whereas $u \in \mathbb{R}^k$ represents the control signal, which is also called the input signal, $x_0 \equiv x(t_0)$ corresponds to the initial condition of the motion that is given in advance. The nominal trajectory to be tracked in the given time-grid takes the values $x^N(t_i) \equiv x_i^N$. In the control task, this nominal trajectory cannot be exactly realized because various restrictions can be prescribed using a Cost Function $C(x,u)$ in each point of the grid. The function $C(x,u) \geq 0$ may express various, often contradictory requirements; It can be constructed as the sum of numerous non-negative terms that expediently are differentiable functions of the state variable and the control signal; The use of large control signals can be *prohibited* in the cost function, too, because of the too complicated numerical calculation process. For the last term at $"t_N"$ an extra terminal condition can be prescribed that depends only on $"x_N"$. In the Optimal Control Approach, the sum is minimized as:

$$\sum_{i=0}^{N-1} C(x,u) + F(x_N) \tag{1}$$

in which the last term $F(x_N)$ gives an extra weight to the last point of the trajectory. However, the cost function cannot be arbitrarily minimized; The dynamics of the system expressed by the state propagation equation must be considered as a constraint in the minimization. This constraint can be processed using the Lagrange Multipliers in the following manner. The time-derivative $"\dot{x}"$ has an expression from the state propagation equation, and the numerical estimation as $\frac{x_{i+1} - x_i}{\Delta t} \approx f(x_i, u_i)$. On this basis a supporting function called auxiliary function can be developed in which the Lagrange multipliers in the great majority of applications have clear physical meaning [17]:

$$\Phi = \sum_{i=0}^{N-1} \left[C(x_i, u_i) + \lambda_i^T \left(\frac{x_{i+1} - x_i}{\Delta t} - f(x_i, u_i) \right) \right] + F(x_N) \tag{2}$$

This proposed auxiliary function $"\Phi"$ is the function of a state variable of the system $"x"$, the control signal $"u"$, and the Lagrange Multiplier $"\lambda"$, i.e. $\Phi = \Phi(\{x\}, \{u\}, \{\lambda\})$. The independent variables of the problem are $\{x_1, ..., x_N\}, \{0, ..., u_{N-1}\}$ and $\{0, ..., \lambda_{N-1} \in \mathbb{R}^n\}$ are the Lagrange multipliers. The auxiliary function $"\Phi"$ obviously is unbounded but it has local saddle points when it has zero partial derivatives by its all variables. For $k \in \{1, 2, ..., N-1\}$ we get:

$$\frac{\partial \Phi}{\partial x_k} = \frac{\partial C(x_k, u_k)}{\partial x_k} + \frac{\lambda_{k-1}}{\Delta t} - \frac{\partial \lambda_k}{\Delta t} - \lambda_k^T \frac{\partial f(x_k, u_k)}{\partial x_k} = 0 \tag{3}$$

for k=N:

$$\frac{\partial \Phi}{\partial x_N} = \frac{\lambda_{N-1}}{\Delta t} + \frac{\partial F(x_N)}{\partial x_N} = 0 \tag{4}$$

For $l \in \{0, 1, 2, ..., N-1\}$:

$$\frac{\partial \Phi}{\partial u_l} = \frac{\partial C(x_l, u_l)}{\partial u_l} - \lambda^T{}_l \frac{\partial f(x_l, u_l)}{\partial u_l} = 0 \tag{5}$$

and for $j \in \{0,1,...,N-1\}$:

$$\frac{\partial \Phi}{\partial \lambda_j} = \frac{x_{j+1} - x_j}{\Delta t} - f\left(x_j, u_j\right) = 0 \tag{6}$$

for the given initial value $"x_0"$. Evidently (3) states that the reduced gradient is zero, that is the set of the $\nabla\Phi = 0$ points contains the points where the above-detailed algorithm stops, (4) is associated with the terminal condition, (6) means that the solution must be on the hypersurface determined by the constraints, and (5) expresses the condition for the control forces. It is worth noting that in general if they exist, the local maximums of the cost function also satisfy the $\nabla\Phi = 0$ condition. However, in many practical applications (e.g. in Thermodynamics) it can be known in advance that it corresponds to the local minimum. The traditional approach considers these equations as a starting point for developing the LQR controller for special cost functions and model structures. Instead of tracking the traditional route it is expedient to observe that if in the variable $X \in \mathbb{R}^k$ all the independent variables of $"\Phi"$ are collected, the function $\Psi(X) = \nabla\Phi(X) : \mathbb{R}^k \mapsto \mathbb{R}^k$ is a $k(\in \mathbb{N})$ dimensional vector function, and our goal is to drive the value of this function to zero from an initial point. This task evidently is in severe analogy with the Inverse Kinematic Task of Robots in which the Cartesian Workshop Coordinates $x(q) \in \mathbb{R}^l$ is the function of the Joint Coordinates $q \in \mathbb{R}^s, l, s \in \mathbb{N}$, and for a redundant robot $s > l$ describe the Forward Kinematics of the robot arm.

In the following section, the control for six-point-long grids without any special terminal cost is explained. The appropriate terms depend on the dynamic model of the physical system under control.

In a preliminary paper tackling the same subject area [21], the above analogy was briefly presented by the use of the equations of motion of the van der Pol oscillator [22] for a long grid. Since this system produced very complex equations in the approximation using finite grid points, in the present case a simpler example, the Duffing oscillator [23] is considered that has a non-linear second-order differential equation used to model certain damped and driven oscillators. Its equation of motion is given as

$$\ddot{x} = -\frac{k}{m}x - \frac{l}{m}x^3 - \frac{b}{m}\dot{x} + \frac{1}{m}u \equiv f(x, \dot{x}, u) \tag{7}$$

in which u is the control force, $"x"$ and $"\dot{x}"$ are the state variables, k is considered a kind of spring constant in the case of a common harmonic oscillator, l describes the coefficient of the third order extension and b corresponds to the viscous damping. For a horizon consisting of 6 grid points the first two ones correspond to the "initial state" of the system, i.e. the "initial coordinate" and the "initial velocity", therefore the free variables of the optimum problem are the coordinate values x in grid points 3,4,5, and 6. In similar manner the control force values u in grid points 1,2,3 and 4 are the independent variables of the problem. Since the system response is the second time-derivative of $"\ddot{x}"$ that numerically can be estimated from minimum 3 grid points we have 4 Lagrange multipliers according to (8).

$$\Phi = h_q(x_3^N - x_3^O) + h_q(x_4^N - x_4^O) + h_q(x_5^N - x_5^O) + h_q(x_6^N - x_6^O) + \cdot$$
$$\cdot h_u(u_1) + h_u(u_2) + h_u(u_3) + h_u(u_4) + \lambda_1[-\frac{k\Delta t^2}{m}x_1^O - \frac{l\Delta t^2}{m}x_1^{O^3} - \cdot$$
$$\cdot \frac{b\Delta t}{m}\left(x_2^O - x_1^O\right) + \cdot \frac{\Delta t^2 u_1}{m} - x_3^O + 2x_2^O - x_1^O] + \lambda_2[-\frac{k\Delta t^2}{m}x_2^O - \cdot$$
$$\cdot \frac{l\Delta t^2}{m}x_2^{O^3} - \frac{b\Delta t}{m}(x_3^O - x_2^O) + \frac{\Delta t^2 u_2}{m} - x_4^O + 2x_3^O - x_2^O] + \lambda_3[-\frac{k\Delta t^2}{m} \cdot$$
$$\cdot x_3^O - \frac{l\Delta t^2}{m}x_3^{O^3} - \frac{b\Delta t}{m}(x_4^O - x_3^O) + \frac{\Delta t^2 u_3}{m} - x_5^O + 2x_4^O - x_3^O] + \lambda_4[- \cdot$$
$$\cdot x_4^O - \frac{l\Delta t^2}{m}\frac{k\Delta t^2}{m}x_4^{O^3} - \frac{b\Delta t}{m}(x_5^O - x_4^O) + \frac{\Delta t^2 u_4}{m} - x_6^O + 2x_5^O - x_4^O] \tag{8}$$

For the notation of the operative indices of the gradient the convention was used as follows:

$$\nabla\Phi_1 = \frac{\partial\Phi}{\partial x_3^O}; \nabla\Phi_2 = \frac{\partial\Phi}{\partial x_4^O}; \nabla\Phi_3 = \frac{\partial\Phi}{\partial x_5^O}$$
$$\nabla\Phi_4 = \frac{\partial\Phi}{\partial x_6^O}; \nabla\Phi_5 = \frac{\partial\Phi}{\partial u_1}; \nabla\Phi_6 = \frac{\partial\Phi}{\partial u_2}$$
$$\nabla\Phi_7 = \frac{\partial\Phi}{\partial u_3}; \nabla\Phi_8 = \frac{\partial\Phi}{\partial u_4}; \nabla\Phi_9 = \frac{\partial\Phi}{\partial \lambda_1}$$
$$\nabla\Phi_{10} = \frac{\partial\Phi}{\partial \lambda_2};\ \nabla\Phi_{11} = \frac{\partial\Phi}{\partial \lambda_3}; \nabla\Phi_{12} = \frac{\partial\Phi}{\partial \lambda_4} \tag{9}$$

$$\nabla\Phi_1 = \frac{\partial\Phi}{\partial x_3^O} = -h_q'(x_3^N - x_3^O) - \lambda_1 - \lambda_2\frac{b\Delta t}{m} + 2\lambda_2 - \frac{k\Delta t^2}{m}\lambda_3 - \cdot$$
$$\cdot \frac{3l\Delta t^2}{m}x_3^{O^2}\lambda_3 + \frac{b\Delta t}{m}\lambda_3 - \lambda_3 \tag{10}$$

$$\nabla\Phi_1 = \frac{\partial\Phi}{\partial x_3^O} = -h_q'(x_3^N - x_3^O) - \lambda_1 + \lambda_2[\frac{b\Delta t}{m} + 2] + \lambda_3[-\frac{k\Delta t^2}{m} - \cdot$$
$$\cdot \frac{3l\Delta t^2}{m}x_3^{O^2} + \frac{b\Delta t}{m} - \lambda_1]$$
(11)

$$\nabla\Phi_2 = \frac{\partial\Phi}{\partial x_4^O} = -h_q'(x_4^N - x_4^O) - \lambda_2 + \lambda_3[-\frac{b\Delta t}{m} + 2] + \lambda_4[-\frac{k\Delta t^2}{m} - \cdot$$
$$\cdot \frac{3l\Delta t^2}{m}x_4^{O^2} + \frac{b\Delta t}{m} - 1] \tag{12}$$

$$\nabla\Phi_3 = \frac{\partial\Phi}{\partial x_5^O} = -h_q'(x_5^N - x_5^O) - \lambda_3 + \lambda_4[-\frac{b\Delta t}{m} + 2] \tag{13}$$

$$\nabla\Phi_4 = \frac{\partial\Phi}{\partial x_6^O} = -h_q'(x_6^N - x_6^O) - \lambda_4 \tag{14}$$

$$\nabla\Phi_5 = \frac{\partial\Phi}{\partial u_1} = h_u'(u_1) + \frac{\lambda_1\Delta t^2}{m} \tag{15}$$

$$\nabla\Phi_6 = \frac{\partial\Phi}{\partial u_2} = h_u'(u_2) + \frac{\lambda_2 \Delta t^2}{m} \quad (16)$$

$$\nabla\Phi_7 = \frac{\partial\Phi}{\partial u_3} = h_u'(u_3) + \frac{\lambda_3 \Delta t^2}{m} \quad (17)$$

$$\nabla\Phi_8 = \frac{\partial\Phi}{\partial u_4} = h_u'(u_4) + \frac{\lambda_4 \Delta t^2}{m} \quad (18)$$

$$\nabla\Phi_9 = \frac{\partial\Phi}{\partial\lambda_1} = -\frac{k\Delta t^2}{m} x_1^O - \frac{l\Delta t^2}{m} x_1^{O^3} - \frac{b\Delta t}{m}(x_2^O - x_1^O) + \frac{\Delta t^2 u_1}{m} - \cdot$$

$$\cdot x_3^O + 2x_2^O - x_1^O \quad (19)$$

$$\nabla\Phi_{10} = \frac{\partial\Phi}{\partial\lambda_2} = -\frac{k\Delta t^2}{m} x_2^O - \frac{l\Delta t^2}{m} x_2^{O^3} - \frac{b\Delta t}{m}(x_3^O - x_2^O) + \cdot$$

$$\cdot \frac{\Delta t^2 u_2}{m} - x_4^O + 2x_3^O - x_2^O \quad (20)$$

$$\nabla\Phi_{11} = \frac{\partial\Phi}{\partial\lambda_3} = -\frac{k\Delta t^2}{m} x_3^O - \frac{l\Delta t^2}{m} x_3^{O^3} - \frac{b\Delta t}{m}(x_4^O - x_3^O) + \cdot$$

$$\cdot \frac{\Delta t^2 u_3}{m} - x_5^O + 2x_4^O - x_3^O \quad (21)$$

$$\nabla\Phi_{12} = \frac{\partial\Phi}{\partial\lambda_4} = -\frac{k\Delta t^2}{m} x_4^O - \frac{l\Delta t^2}{m} x_4^{O^3} - \frac{b\Delta t}{m}(x_5^O - x_4^O) + \cdot$$

$$\cdot \frac{\Delta t^2 u_4}{m} - x_6^O + 2x_5^O - x_4^O \quad (22)$$

The second partial derivatives correspond to the matrix elements of the Jacobian of the problem. In the sequel only, the nonzero terms are detailed as follows (we used the symmetry properties of the second partial derivatives that $J_{ij} = J_{ji}$):

$$J_{1,1} = \frac{\partial\Delta\Phi_1}{\partial x_3^O} = h_q''(x_3^N - x_3^O) - \frac{6\Delta t^2}{m} x_3^O \lambda_3 \quad (23)$$

$$J_{1,9} = \frac{\partial\Delta\Phi_1}{\partial\lambda_1} = J_{9,1} = \frac{\partial\Delta\Phi_9}{\partial x_3^O} = -1 \quad (24)$$

$$J_{1,10} = \frac{\partial\Delta\Phi_1}{\partial\lambda_2} = J_{10,1} = \frac{\partial\Delta\Phi_{10}}{\partial x_3^O} = -\frac{b\Delta t}{m} + 2 \quad (25)$$

$$J_{1,11} = \frac{\partial\Delta\Phi_1}{\partial\lambda_3} = J_{11,1} = \frac{\partial\Delta\Phi_1}{\partial x_3^O} = -\frac{k\Delta t^2}{m} - \frac{3l\Delta t^2}{m} x_3^{O^2} + \cdot$$

$$\cdot \frac{b\Delta t}{m} - 1 \quad (26)$$

$$J_{2,2} = \frac{\partial\Delta\Phi_2}{\partial x_4^O} = h_q''(x_4^N - x_4^O) - \frac{6l\Delta t^2}{m} x_4^O \lambda_4 \quad (27)$$

$$J_{2,10} = \frac{\partial\Delta\Phi_2}{\partial\lambda_2} = J_{10,2} = \frac{\partial\Delta\Phi_{10}}{\partial x_4^O} - 1 \quad (28)$$

$$J_{2,11} = \frac{\partial\Delta\Phi_2}{\partial\lambda_3} = J_{11,2} = \frac{\partial\Delta\Phi_{11}}{\partial x_4^O} = -\frac{b\Delta t}{m} + 2 \quad (29)$$

$$J_{2,12} = \frac{\partial\Delta\Phi_2}{\partial\lambda_4} = J_{12,2} = \frac{\partial\Delta\Phi_{12}}{\partial x_4^O} = -\frac{k\Delta t^2}{m} - \frac{3l\Delta t^2}{m} x_4^{O^2} + \cdot$$

$$\cdot \frac{b\Delta t}{m} - 1 \quad (30)$$

$$J_{3,3} = \frac{\partial\Delta\Phi_3}{\partial x_5^O} = h_q'(x_5^N - x_5^O) \quad (31)$$

$$J_{3,11} = \frac{\partial\Delta\Phi_3}{\partial\lambda_3} = J_{11,3} = \frac{\partial\Delta\Phi_{11}}{\partial x_4^O} = -1 \quad (32)$$

$$J_{3,12} = \frac{\partial\Delta\Phi_3}{\partial\lambda_4} = J_{12,3} = \frac{\partial\Delta\Phi_{12}}{\partial x_5^O} = -\frac{b\Delta t}{m} + 2 \quad (33)$$

$$J_{4,4} = \frac{\partial\Delta\Phi_4}{\partial x_6^O} = h_q''(x_6^N - x_6^O) \quad (34)$$

$$J_{4,12} = \frac{\partial\Delta\Phi_4}{\partial\lambda_4} = J_{12,4} = \frac{\partial\Delta\Phi_{12}}{\partial x_6^O} = -1 \quad (35)$$

$$J_{5,9} = \frac{\partial\Delta\Phi_5}{\partial\lambda_1} = J_{5,9} = \frac{\partial\Delta\Phi_5}{\partial x_3^O} = \frac{\Delta t^2}{m} \quad (36)$$

$$J_{6,6} = \frac{\partial\Delta\Phi_6}{\partial u_2} = h_u''(u_2) \quad (37)$$

$$J_{6,10} = \frac{\partial\Delta\Phi_6}{\partial\lambda_2} = J_{10,6} = \frac{\partial\Delta\Phi_{10}}{\partial u_1} = \frac{\Delta t^2}{m} \quad (38)$$

$$J_{7,7} = \frac{\partial\Delta\Phi_7}{\partial u_3} = h_u''(u_3) \quad (39)$$

$$J_{7,11} = \frac{\partial\Delta\Phi_7}{\partial\lambda_3} = J_{11,7} = \frac{\partial\Delta\Phi_{11}}{\partial u_3} = \frac{\Delta t^2}{m} \quad (40)$$

$$J_{8,8} = \frac{\partial\Delta\Phi_8}{\partial u_4} = h_u''(u_4) \quad (41)$$

$$J_{8,12} = \frac{\partial\Delta\Phi_8}{\partial\lambda_4} = J_{12,8} = \frac{\partial\Delta\Phi_{12}}{\partial u_4} = \frac{\Delta t^2}{m} \quad (42)$$

III. Clarification of the Analogy with the Solution of the Inverse Kinematic Task of Robots

In open kinematic chain, the inverse kinematic task of robots generally can be explained through differential solutions which are basically determined by a generalized inverse, in other words, called, pseudo-inverse of the Jacobian of the arm [24]. For the joint coordinates of the *n* DoF open kinematic chain $q \in \mathbb{R}^n, n \in \mathbb{N}$ is used. The array made of the Cartesian coordinates is denoted $x \in \mathbb{R}^m, m \in \mathbb{N}$ of the certain points extended with the information on the pose of certain components with respect to the "workshop frame". For the preparation of the motion of the arm that function can be used.

Our task is to find the system states variable "q" for a given desired position of the arm x^{Des} which has closed form solutions only in special cases of the construction, for

example, in the case of a PUMA-type robot [25]. According to the general possibility the differential solution based on the use of the Jacobian $\frac{\partial x}{\partial q}$ in a function of a scalar variable $\xi \in \mathbb{R}$ where $x(\xi) = x(q(\xi))$ is considered in the equation

$$\frac{dx_j}{d\xi} = \sum_i \frac{\partial x_j}{\partial q_i}\frac{dq_i}{d\xi} \equiv \sum_i J_{ji}\frac{dq_i}{d\xi} \tag{43}$$

where the initial conditions as $x(\xi_{ini}) = x_{ini}$ and $q(\xi_{ini})$ are known. In the redundant case, to choose one of the useful solutions, some additional idea is needed. The classical explanations hold some generalized inverse as e.g. the Moore-Penrose Pseudoinverse [26], [27] that is singular in, and ill-conditioned near the kinematic singularities of the robot arm. Other generalized inverses based on the Singular Value Decomposition (SVD) [28] are not related to cost function minimization. Generally, such problems generate a huge complication due to joint coordinate time derivatives, therefore, it is expedient to "tame" the original task to evade the numerical inconveniences e.g. in the method of Damped Least Squares [29]. Instead of using the traditional concept an alternative method in [30] has been proposed where the original task was transformed into a fixed point problem that subsequently was solved by simple iteration. Its special advantage is, without using any complementary trick, its capability to show the stable solution automatically in and near the kinematic singularities. It also selects one of the ambiguous solutions automatically. To drive $\nabla\Phi$ to zero in the novel RHC controller this algorithm is suggested and observed better results. The essence of the method is briefed below.

In the 17th century, the idea of transformation of a given task into fixed point problem had been used. Gradually with the passage of time the usage of this approach got a broader attention and its solution was found through iterations, e.g. Newton-Raphson Algorithm, that has many applications even in our days [31]. In 1922 Stefan Banach extended this way of thinking to quite wide problem classes [32]. According to his theorem, in a linear, complete metric space (i.e. the "Banach Space") the sequence created by the contractive mapping $\psi : \mathbb{R}^m \mapsto \mathbb{R}^m, m \in \mathbb{N}$ as $x_{s+1} = \psi(x_s)$ is a Cauchy Sequence that converges to the fixed point of ψ defined as $\psi(x_*) = x_*$. (A map is contractive if $\exists 0 \le p < 1$ so that $\forall x, y$ elements of the space $\|\psi(x) - \psi(y)\| \le p\|x - y\|$.) In [33] the following transformation was used for this purpose: a real differentiable function $\varphi(\xi) : \mathbb{R} \mapsto \mathbb{R}$ was taken with an attractive fixed point $\varphi(\xi_*) = \xi_*$. It was used for the generation of a sequence of iterative signals as follows. Assume that we have a time grid, the system's variable at time instant *k* is in the close vicinity of the appropriate nominal value x_k, and from this point, we wish to move the system to the next grid point x_{k+1}^{Des} in an iterative manner using the index "*i*" as follows:

$$q_{k,i+1} = \left[\varphi\left(A\left\|x(q_{k,i}) - x_{k+1}^{Des}\right\| + \xi_*\right) - \xi_*\right] \cdot \frac{x(q_{k,i}) - x_{k+1}^{Des}}{\left\|x(q_{k,i}) - x_{k+1}^{Des}\right\|} + q_{k,i}, \tag{44}$$

in which the Frobenius norm was used, and *A* is an adaptive parameter. For $q_{k,i} = q_*$ that provides $x(q_*) = x_{k+1}^{des}$, (44) yields $q_{k,i+1} = q_{k,i}$. It means that if "q_*" is the solution of our task, it also is the fixed point of this function. (Generally, no unique solution is expected.) The convergence of this sequence was investigated in [34] by making the first order Taylor series approximation of $\varphi(\xi)$ in the vicinity of ξ_* and that of $x(q)$ around "q_*". It was found that if the real part of each eigenvalue of the Jacobian $\frac{\partial x}{\partial q}$ is simultaneously positive or negative, an appropriate parameter *A* can be so chosen that it guarantees the convergence.

For studying the convergence of this sequence in [35] it was assumed that $A\left\|x(q_{k,i}) - x_{k+1}^{Des}\right\|$ was small in comparison with ξ_* therefore, as an approximation, by the first order Taylor series expansion of $x(q)$ around "q_*", and $\varphi(\xi)$ in the vicinity of ξ_*. It can be shown that the convergence will be determined by the positive semidefinite matrix. The results for the redundant robot arms of non-quadratic Jacobians in [30] has been explained where instead of an original problem $x^{Des} = x(q)$ the modified $J^T(q)x^{Des} = J^T(q)x(q)$ was solved. Also, the convergence will be determined by the positive semidefinite matrix $J^T(q)J(q)$ that has non-negative eigenvalues. For adaptively tracking the "optimized trajectory" a similar transformation into a fixed point problem was applied as it is briefed in the sequel.

IV. Fixed Point Transformation-based Adaptive Control

Investigations on the application of a "Fixed Point Transformation based adaptive control" have obtained some attention in the recent years. This is an alternative approach of the Lyapunov function-based adaptivity that seems to be usual in the design of controllers for nonlinear systems as e.g. robots from the nineties of the past century [36]- [38].

The main idea of transforming an adaptive control task into a fixed point problem has been considered in various tasks in the beginning of this decade [39]. According to Fig. 1 it can be presently illustrated for the digital control of a second order system as follows: by applying a proper tracking error feedback in the "Kinematic Block" to calculate the "Desired Tracking Error Damping" in case of a PD-type controller it is

$$\ddot{q}^{Des}(t) = \ddot{q}^{N}(t) + 2\Lambda\left(\dot{q}^{N}(t) - \dot{q}(t)\right) + \Lambda^2\left(q^{N}(t) - q(t)\right) \tag{45}$$

for a constant $\Lambda > 0$ time-exponent by the use of this signal the elements of the sequence of the Deformed Control Signals $\ddot{q}^{Def}(t)$ are created by the function in (45); this deformed signal is used as the input of the available "Approximate

Model" of the controlled system for the calculation of the control force $Q(t)$ that is exerted on the actually controlled system that generates the realized response $\ddot{q}(t)$. (The symbolic integrations at the bottom of the figure are done by the dynamics of the controlled system in a real control situation, or, in the case of a simulation study, they must be implemented numerically.) After converging to the fixed point the kinematically prescribed trajectory tracking error damping will be precisely realized. In [16] the same control was implemented in an EXCEL-Solver-Visual Basic environment. In the present research, the same structure is used in the proposed adaptive RHC controller.

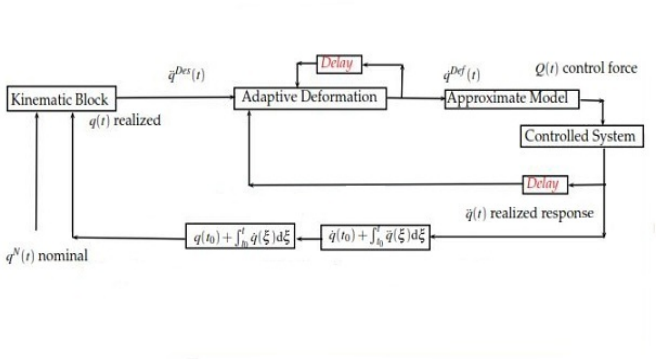

Fig.1: Schematic structure of the "Fixed Point Transformation-based Adaptive Controller" taken from [40]

IV. SIMULATION INVESTIGATION RESULTS

The appropriate model parameters of the Duffing oscillator considered are given in Table 1.

Parameter	Exact	Approximate	Traj. Generators
$m\ kg$	1.0	1.2	0.9
$k\ Nm^{-1}$	100.0	120.0	90.0
$l\ Nm^{-3}$	0.5	0.6	0.4
$b\ Nsm^{-1}$	1.2	1.5	0.7

In the cost function the "original form" of the tracking error contribution was $h_q\left(x^N - x\right) = \left(\frac{\left|x^N - x\right|}{A_x}\right)^{\alpha_x}$, and for the prohibition of the too big control effort $h_u(u) = \left(\frac{|u|}{A_u}\right)^{\alpha_u}$ were used. For very big α_x the penalty has very fast increase in the region $\left|x^N - x\right| > A_x$ and it is very small for $\left|x^N - x\right| < A_x$. Since for great α_x this may result in numerical difficulties, these power functions were "tamed" in the following manner: at first the cost functions were modified as $h = ln\left(1 + M\left[\frac{sgn(x)\,x}{A}\right]^{\alpha}\right)$, and in its 1st and 2nd time-derivatives the function *sgn*(*x*) was considered as a classical function that cannot be differentiated in a single point $x = 0$. In the next steps, in the derivatives, this function was "softened" as $sgn(x) \approx tanh\left(\frac{x}{w}\right)$. The 1st and 2nd order derivatives of the cost function had the structure:

$$x' = \frac{\alpha}{A_x} M \frac{1}{1 + \left(\frac{tanh\left(\frac{x}{w}\right)x}{A_x}\right)^{\alpha} M} \left(\frac{tanh\left(\frac{x}{w}\right)x}{A_x}\right) \bullet$$

$$\bullet tanh\left(\frac{x}{w}\right)^{\alpha-1} \tag{46a}$$

$$x'' = \frac{\alpha}{A_x^2} M \frac{1}{1 + \left(\frac{tanh\left(\frac{x}{w}\right)x}{A_x}\right)^{\alpha} M} \left[\frac{-M\alpha\left(\frac{tanh\left(\frac{x}{w}\right)x}{A_x}\right)^{2(\alpha-1)}}{1 + M\left(\frac{tanh\left(\frac{x}{w}\right)x}{A_x}\right)^{\alpha}}\right] + \bullet$$

$$\bullet(\alpha - 1)\left(\frac{tanh\left(\frac{x}{w}\right)x}{A_x}\right)^{\alpha-2} \tag{46b}$$

The other control parameters we used are: $A_u = 1.0$, $\alpha_u = 3.0$, $B_u = 1\times10^0$, where A_u used for basis, and α_u is an exponent for punishing the control force, B_u is the weighting coefficient for the control force restriction in the cost function, and for the starting value for tuning u variable was used is $u_{ini} = 10.0$ was taken.

The control input used for dynamic tracking is $\Lambda = 1.0 s^{-1}$, $w = 1.0\times10^{-6}$ was used for softening of the cost function near zero, and $M = 0.01$ was used for softening the cost function for large costs.

In the sequel the following essential parameters were varied: $\alpha_x = 2$ corresponds to "soft tracking", $\alpha_x = 6$ corresponds to "sharp tracking". Parameter $A_x = 5\times10^{-2}$ means "strict tracking", while $A_x = 1\times10^{-1}$ was used for "loose tracking". The parameter in the scheme in Fig. 1 $A = -3\times10^{-1}$ corresponds to "slow dynamic tracking", and $A = -3\times10^{0}$ means "fast dynamic tracking".

In the figures depicted below, results for the "*nominal*", "*optimized*", and "*realized*" trajectories have been clarified where the correlations between them can be observed. For the varied parameters, similarly, the difference between the nominal and optimized trajectories, and the real part of the eigenvalues also show dissimilar situations.

In Fig. 2 the adjustment of "*fast, loose, and sharp*" parameters, in Fig. 8 "*fast, strict, and sharp*" adjusted parameters, in Fig. 14, "*slow, loose, and sharp*", whereas in Fig. 20 "*slow, strict, and sharp*" parameters adjusted for tracking were chosen. The conditions of trajectories in Figs. 5, 11, 17, and 23 illustrate an assorted scenario where the

"*optimized*" and "*realized*" trajectories gradually track and meet the "*nominal*" one after exhibiting an initial jump, and the difference between "*optimized*" and "*nominal*" state variables is declining with an irregular shape. Similarly, identical shapes with minor error occurred while subtracting "*optimized*" state from "*nominal*" state.

In Figs. 3, 9, 15, and 21 the gradient trajectories are decreasing with a consistent form, whereas in Figs. 6, 12, 18, and 24 such gradients decrease with an irregular and not consistent form. The figures are depicted below:

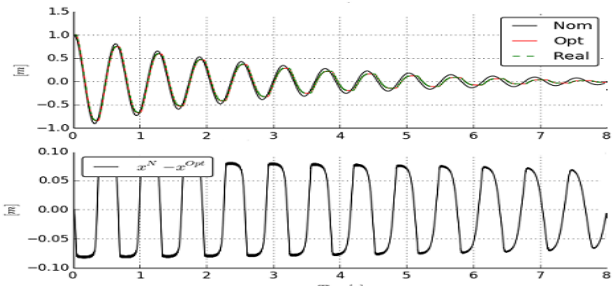

Fig.2 The Nominal, Optimized and Realized Trajectory and Difference for fast, loose, and sharp tracking.

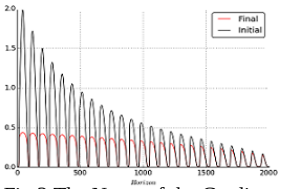

Fig.3 The Norm of the Gradient: Iterations=100 for fast, loose, and sharp tracking.

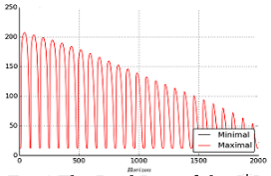

Fig.4 The Real Part of the $J^{1}J$ Eigenvalues for fast, loose, and sharp tracking.

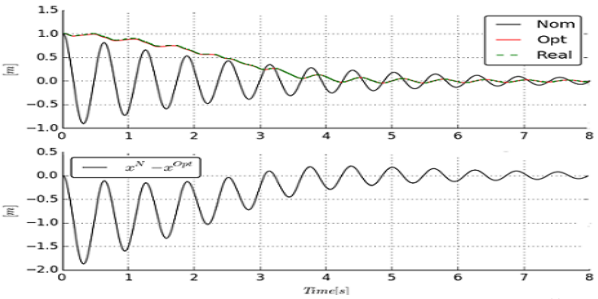

Fig.5 The Nominal, Optimized, and Realized Trajectory and Difference for fast, loose, and soft tracking.

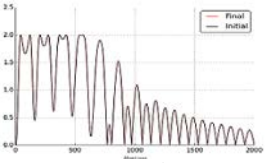

Fig.6 The Norm of the Gradient: Iterations=100 for fast, loose, and soft tracking.

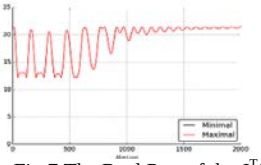

Fig.7 The Real Part of the $J^{T}J$ Eigenvalues for fast, loose, and soft tracking.

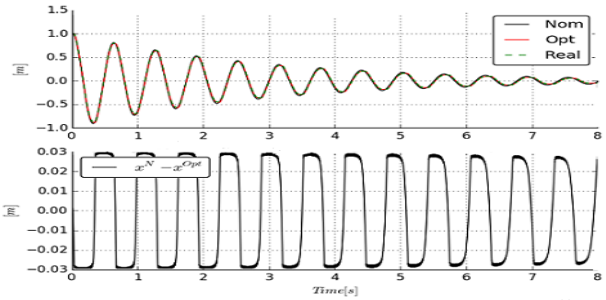

Fig.8 The Nominal, Optimized, and Realized Trajectory and Difference for fast, strict, and sharp tracking

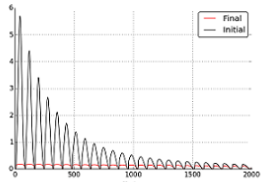

Fig.9 The Norm of the Gradient: $J^{T}J$Iterations=100 for fast, strict, and andsharp tracking.

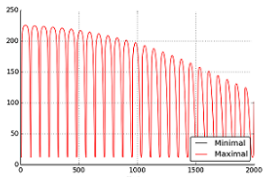

Fig.10 The Real Part of the Eigenvalues for fast, strict, sharp tracking.

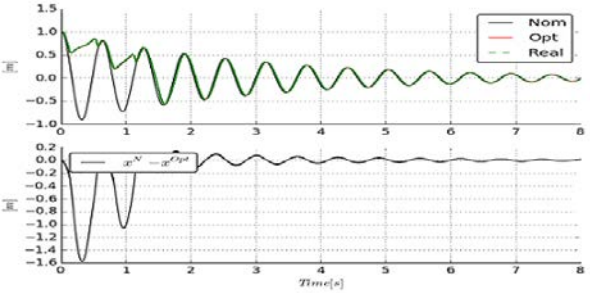

Fig.11 The Nominal, Optimized, and Realized Trajectory and Difference for fast, strict, and soft tracking.

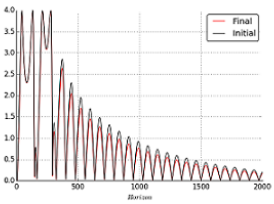

Fig.12 The Norm of the Gradient: Iterations=100 for fast, strict, and soft tracking.

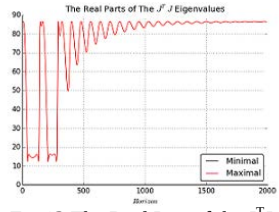

Fig.13 The Real Part of the $J^{T}J$ Eigenvalues for fast, strict, and soft tracking.

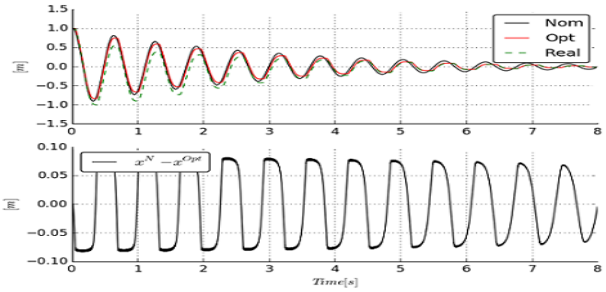

Fig.14 The Nominal, Optimized, and Realized Trajectory and Difference for slow, loose, and sharp tracking.

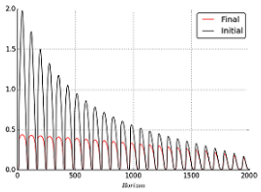

Fig.15 The Norm of the Gradient: Iterations=100 for slow, loose, and sharp tracking.

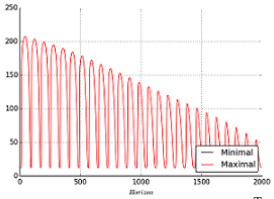

Fig.16 The Real Part of the J^TJ Eigenvalues for slow, loose, and sharp tracking.

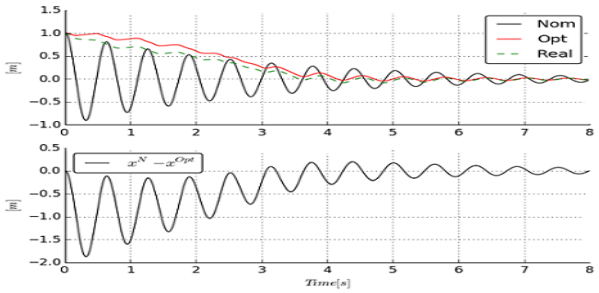

Fig.17 The Nominal, Optimized, and Realized Trajectory and Difference for slow, loose, and soft tracking.

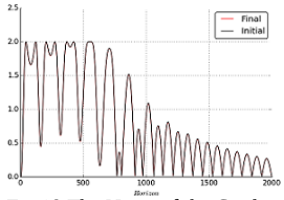

Fig.18 The Norm of the Gradient: Iterations=100 for slow, loose, and soft tracking.

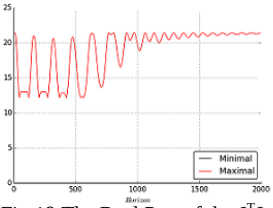

Fig.19 The Real Part of the J^TJ Eigenvalues for slow, loose, and soft tracking.

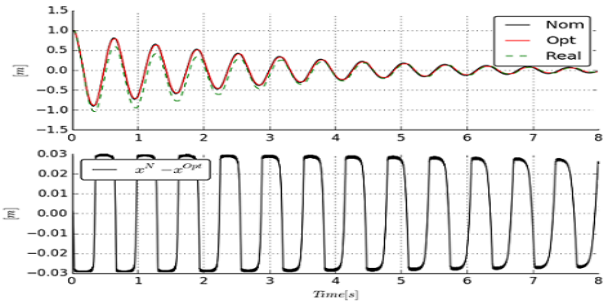

Fig.20 The Nominal, Optimized, and Realized Trajectory and Difference for slow, strict, and sharp tracking.

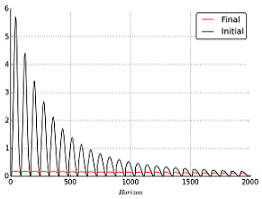

Fig.21 The Norm of the Gradient: Iterations=100 for slow, strict, and sharp tracking.

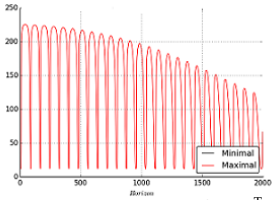

Fig.22 The Real Part of the J^TJ Eigenvalues for slow, strict, and sharp tracking.

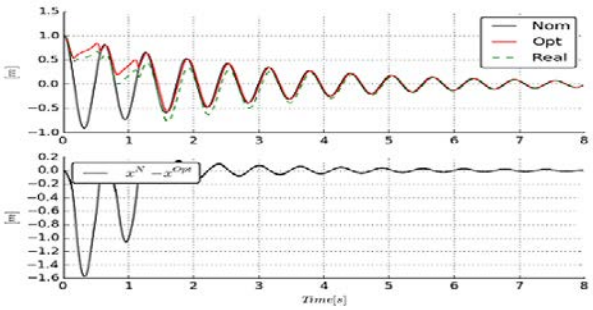

Fig.23 The Nominal, Optimized and Realized Trajectory and Difference for slow, strict, and soft tracking.

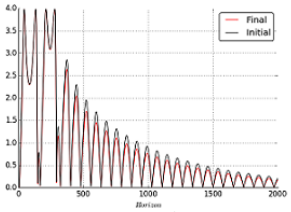

Fig.24 The Norm of the Gradient: Iterations=100 for slow, strict, and sharp tracking.

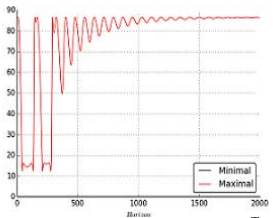

Fig.25 The Real Part of the J^TJ Eigenvalues for slow, strict, and sharp tracking.

CONCLUSION

As it can be seen from the simulations, the suggested numerical approach yielded well interpretable results. The observable effects of the typical parameter values qualitatively can be well identified and correspond to the expectations. It has been demonstrated, too, that the suggested approach can reach some suboptimal solution because the gradient of the auxiliary function was not driven to zero exactly. However, this situation can occur in the case of using the sophisticated Solver package when the algorithm is stopped before reaching the local optimum. The difficulties of the here suggested method reveal themselves in the fact that we have “analytically’ construct the elements of the gradient of the auxiliary function and that of the Jacobian. In the future work the introduction of some numerical approximation of these matrix elements can be considered.

ACKNOWLEDGMENT

Authors are very thankful to the Doctoral School of Applied Informatics and Applied Mathematics Obuda University Budapest Hungary to fully support in this research.

REFERENCES

[1] L. Grune and J. Pannek, “Nonlinear Model Predictive Control,” *Springer* 2011.

[2] Grancharova and T.A. Johansen, “Explicit Nonlinear Model Predictive Control,” Springer 2012.

[3] Hamza Khan, Jozsef K. Tar, Imre Rudas, Levente Kovacs, and Gyorgy Eigner, “Receding Horizon Control of Type 1 Diabetes Mellitus by Using Nonlinear Programming,”, *Journal of Complexity*, submitted for publication.

[4] N. Moldovanyi, “Model Predictive Control of Crystallisers” Ph.D. Thesis, Department of Process Engineering, University of Pannonia, Veszprem, Hungary, 2012.

[5] N. Muthukumar, S. Srinivasan, K. Ramkumar, K. Kannan, and V. Balas, “Adaptive model predictive controller for web transport systems,” *Acta Polytechnica Hungarica*, vol. 13, no. 3, pp. 181–194, 2016.

[6] T. Tettamanti and I. Varga, Distributed Traffic Control System based on Model Predictive Control, *Periodica Polytechnica ser*. Civil Eng. 54(1), 2010, pp. 3–9.
[7] S. Lin, B. De Schutter, Y. Xi and J. Hellendoorn, Fast model predictive control for urban road networks via MILP, *IEEE Transactions on Intelligent Transportation Systems* 12, 2011, pp. 846–856.
[8] J.W. Eaton and J.B. Rawlings, Feedback control of chemical processes using on-line optimization techniques, *Computers & Chem. Eng.* 14, 1990, pp. 469–479.
[9] N. Moldovanyi, Model Predictive Control of Crystallisers (Ph.D. Thesis), *Department of Process Engineering, University of Pannonia, Veszprem, Hungary* 2012.
[10] I. Nascu, R. Oberdieck and E.N. Pistikopoulos, Offset-free explicit hybrid model predictive control of intravenous anesthesia, *In: Proc. of the 2015 IEEE International Conference on Systems, Man, and Cybernetics, October 9-13, 2015, Hong Kong 2015*, pp. 2475–2480.
[11] N. Muthukumar, Seshadhri Srinivasan, K. Ramkumar, K. Kannan and V.E. Balas, Adaptive Model Predictive Controller for Web Transport Systems, *Acta Polytechnica Hungarica* 13(3), 2016, pp. 181–194.
[12] R.E. Bellman, Dynamic Programming and a new formalism in the calculus of variations, *Proc. Natl. Acad. Sci.* 40(4), 1954, pp. 231–235.
[13] R.E. Bellman, Dynamic Programming, *Princeton Univ. Press, Princeton, N. J.* 1957.
[14] J. Richalet, A. Rault, J.L. Testud and J. Papon, Model predictive heuristic control: Applications to industrial processes, *Automatica* 14(5), 1978, pp. 413–428.
[15] A. Jadbabaie, Receding Horizon Control of Nonlinear Systems: A Control Lyapunov Function Approach (Ph.D. Thesis), *California Institute of Technology, Pasadena, California*, USA 2000.
[16] H. Khan, A. Szeghegyi and J.K. Tar, Fixed Point Transformation-based Adaptive Optimal Control Using NLP, *In the Proc. of the 2017 IEEE 30th Jubilee Neumann Colloquium, November 24-25, 2017, Budapest, Hungary* 2017, pp. 35–40.
[17] J.L. Lagrange, J.P.M. Binet and J.G. Garnier, Mecanique analytique (Eds. J.P.M. Binet and J.G. Garnier), *Ve Courcier, Paris* 1811.
[18] R.E. Kalman, Contribution to the Theory of Optimal Control, *Boletin Sociedad Matematica Mexicana* 5(1), 1960, pp. 102–119.
[19] Tayfun C¸ imen, State-Dependent Riccati Equation in Nonlinear Optimal Control Synthesis, *In the Proc. of the Special International Conference on Complex Systems: Synergy of Control, Communications, and Computing- COSY 2011, Hotel Metropol Resort, Ohrid, Republic of Macedonia, September 16 –20, 2011 2011*, pp. 321–332.
[20] Tjalling J. Ypma, Historical development of the Newton-Raphson method, *SIAM Review* 37(4), 1995, pp. 531–551.
[21] Hamza Khan, Jozsef K. Tar, Imre J. Rudas, and Gyorgy Eigner, "Adaptive Model Predictive Control Based on Fixed Point Iteration," *WSEAS Transactions on Systems and Control*, 12(37), 2017, pp. 347-354.
[22] Hamza Khan, A. Galantai and J.K. Tar, Adaptive solution of the inverse kinematic task by fixed point transformation, *In the Proc. of the 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI), January 26-28, 2017, Herl'any, Slovakia 2017*, pp. 1-6.
[23] R. E. Mickens, "Mathematical and numerical study of the Duffing-harmonic oscillator," *Journal of Sound and Vibration*, 244 (3), 2001, pp. 563-567.
[24] Ivana Kovacic Ronald E. Mickens, " A generalized van der pol type Oscillator: Investigation of its Limit Cycle," *Mathematical and Computer Modelling*, 55(3–4), 2012, pp. 645-653
[25] C.S.G. Lee and M. Ziegler, RSD-TR-1-83 A geometric approach is solving the inverse kinematics of PUMA robots, *The University of Michigan, Ann Arbor, Michigan 48109-1109* 1983.
[26] E.H. Moore, On the reciprocal of the general algebra Dineva Ph.D.:2016ic matrix, *Bulletin of the American Mathematical Society* 26(9), 1920, pp. 394–395.
[27] R. Penrose, A generalized inverse for matrices, *Proceedings of the Cambridge Philosophical Society* 51, 1955, pp. 406–413.
[28] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *SIAM Journal on Numerical Analysis*, vol. 2, pp. 205–224, 1965.
[29] S. Chiaverini, O. Egeland, and R.K. Kanestrom, Achieving User-Defined Accuracy with Damped Least Squares Inverse Kinematics, *In the Proc. of the 1991 IEEE International Conference on Robotics and Automation, June 19-22, 1991, Pisa, Italy* 1991.
[30] I. Csanadi, J.K. Tar, and J.F. Bito, Matrix Inversion-free Quasi Differential Approach in Solving the Inverse Kinematic Task, In Proc. of the 17 IEEE International Symposium on Computational Intelligence and Informatics (CINTI 2016), 17-19 November 2016, Budapest, Hungary 2016, pp. 61–66.
[31] C.T. Kelley, Solving Nonlinear Equations with Newton's Method, no 1 in Fundamentals of Algorithms, *SIAM* 2003.
[32] S. Banach, Sur les operations dans les ensembles abstraits et leur application aux equations integrals, *Fund. Math*. 3, 1922, pp. 133–181.
[33] A. Dineva, J.K. Tar, A. Varkonyi-Koczy and V. Piuri, Adaptive Control of Underactuated Mechanical Systems Using Improved "Sigmoid Generated Fixed-Point Transformation" and Scheduling Strategy, *In Proc. of the 14 IEEE International Symposium on Applied Machine Intelligence and Informatics, January 21-23, 2016, Herl'any, Slovakia* 2016, pp. 193–197.
[34] Dineva, Non-conventional Data Representation and Control (Ph.D. Thesis), *Obuda University, Budapest, Hungary* 2016.
[35] Csanadi, J. Tar, and J. Bito, "Matrix inversion-free quasi-differential approach in solving the inverse kinematic task," *In Proc. of 17th IEEE International Symposium on Computational Intelligence and Informatics, 17-19 November 2016, Budapest, Hungary*, pp. 1–6, 2016.
[36] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall International, Inc., Englewood Cliffs, New Jersey, 1991.
[37] R. Isermann, K. Lachmann, and D. Matko, *Adaptive Control Systems*. Prentice-Hall, New York DC, USA, 1992.
[38] J. Somlo, B. Lantos, and P. Cat, *Advanced Robot Control*. Akademiai Kiado, Budapest, 2002.
[39] J.K. Tar, J.F. Bito, L. Nadai and J.A. Tenreiro Machado, Robust Fixed-Point Transformations in Adaptive Control Using Local Basin of Attraction, *Acta Polytechnica Hungarica* 6(1), 2009, pp. 21–37.
[40] H. Redjimi and J.K. Tar, On the Effects of Time Delay on Precision Degradation in Fixed Point Transformation-based Adaptive Control, *In the Proc. of the 2017 IEEE 30th Jubilee Neumann Colloquium, November 24-25, 2017, Budapest, Hungary* 2017, pp. 125–130.

Hamza Khan

M.Phil. in Mathematics from Pakistan. He is a Ph.D Student in Doctoral School of Applied Informatics and Applied Mathematics, Obuda University Budapest, Hungary. Becsi ut 96/B, H-1034, Budapest, Hungary.
Email: hamza.khan@phd.uni-obuda.hu

Jozsef K. Tar

Dr. habil, PhD, DSc. He is a certified Physicist. Full professor in the Institute of Applied Mathematics, Óbudai Egyetem. He does research in Adult Education, Higher Education and Science Education. Currently working at Obuda University Antal Bejczy Center for Intelligent Robotics (ABC iRob), Becsi ut 96/B, H-1034 Budapest HUNGARY
Email: tar.jozsef@nik.uni-obuda.hu

Imre J. Rudas

Dr. PhD, DSc. Full professor at University of Obuda Janos Neumann Faculty of Informatics Faculty of Intelligent Engineering Systems, 1034 Budapest, Becsi ut 96/B. Currently working at Obuda University Antal Bejczy Center for Intelligent Robotics (ABC iRob), Becsi ut 96/B, H-1034 Budapest HUNGARY,
Email: rudas@uni-obuda.hu

Gyorgy Eigner

PhD in Applied Informatics. György Eigner currently works as an Assistant Lecturer in John von Neumann Faculty of Informatics, Physiological Controls Research Center, Óbudai Egyetem.
Email: eigner.gyorgy@nik.uni-obuda.hu