Sub-optimal Solution of the Inverse Kinematic Task of Redundant Robots without Using Lagrange Multipliers

Bence Varga
Doctoral School of Applied Informatics and Applied Mathematics
Bánki Donát Faculty of Mechanical and Safety Engineering
Óbuda University
Budapest, Hungary
varga.bence@bgk.uni-obuda.hu

Hazem Issa
Doctoral School of Applied Informatics and Applied Mathematics
Óbuda University
Budapest, Hungary
hazem.issa@uni-obuda.hu

Richárd Horváth
Bánki Donát Faculty of Mechanical and Safety Engineering
Óbuda University
Budapest, Hungary
ORCID: 0000-0002-4924-3244

József K. Tar
Doctoral School of Applied Informatics and Applied Mathematics
John von Neumann Faculty of Informatics
Antal Bejczy Center of Intelligent Robotics
Óbuda University
Budapest, Hungary
ORCID: 0000-0002-5476-401X

***Abstract*—In the paper a novel approach is suggested for solving the inverse kinematic task of redundant open kinematic chains. Traditional approaches as the Moore-Penrose generalized inverse-based solutions minimize the sum of squares of the time-derivative of the joint coordinates under the constraint that contains the task itself. In the vicinity of kinematic singularities where these solutions are possible the hard constraint terms produce high time-derivatives that can be reduced by the use of a deformation proposed by Levenberg and Marquardt. The novel approach uses the basic scheme of the Receding Horizon Controllers in which the Lagrange multipliers are eliminated by direct application of the kinematic model over the horizon in the role of the "control force", and no reduced gradient has to be computed. This fact considerably decreases the complexity of the solution. If the cost function contains penalty for high joint coordinate time-derivatives the kinematic singularities are ab ovo better handled. Simulation examples made for a 7 degree of freedom robot arm demonstrate the operation of the novel approach. The computational need of the method is still considerable but it can be further decreased by the application of complementary tricks.**

***Index Terms*—inverse kinematic task; gradient descent method; reduced gradient algorithm; Moore-Penrose pseudoinverse; receding horizon control.**

I. Introduction

This paper is the extended version of [1], in which a novel approach was outlined for solving the inverse kinematic task of redundant open kinematic chain of general structure. In special cases, e.g., in which three orthogonal rotary axles have to have a common point as in the case of the PUMA type robots and certain other structures (e.g., [2]–[4]) closed form calculations are possible. However, as is well known, generally no closed analytical solutions exist for the inverse kinematic task of general structures. In such cases the differential solutions can be applied in which the Jacobian of the problem has to be computed, and, following that, the ambiguity of the solution has to be tackled. Normally a single solution of the infinite number of the possible ones has to be chosen. The most popular solution is the application of the Moore-Penrose pseudoinverse [5]–[7] that does not exists in the kinematically singular configurations, and in their vicinity it results high joint coordinate time-derivatives. To avoid this situation Levenberg and Marquardt suggested some distortion in the solution of similar tasks [8], [9]. If the parameter of the distortion is well set it causes negligible imprecision in the far from singularity points, and produces acceptable joint coordinate time-derivatives in the near singular configurations. It must be noted that the robot is "clumsy" in the vicinity of the singularities, therefore it cannot be used for realizing precisely prescribed motion. Normally, the so obtained solution can be modified e.g., by adding components from the null space of the Jacobian, or other programming possibilities also exist (e.g., [10]–[12]).

From mathematical point of view the Moore-Penrose Pseudoinverse correspond to the solution of an optimization task under constraints, in a very special case in which this solution

This research was supported by the Doctoral School of Applied Informatics and Applied Mathematics of Óbuda University.

can be obtained without the implementation of some numerical procedure. The sum of the squares of of the joint coordinate derivatives has to be minimized under the constraint that the inverse kinematic task is also solved.

$$\sum_i \dot{q}_i^2 = \min \text{ so that } \dot{x} = J(q)\dot{q} \ , \tag{1}$$

where $x \in \mathbb{R}^m$ contains the components of the Cartesian coordinates of the tool center point of the robot and that describing the pose of the workpiece, $q \in \mathbb{R}^n$, and usually $n, m \in \mathbb{N}$, $n > m$.

II. Receding Horizon Controllers

In traditional control design the basic principle was to maintain some desired properties of the controlled system. The efficiency of the developed control law was highly dependent on the developer him self. To overcome this issue the *Optimal Controller (OC)* framework was developed, which is a cost functional-based solution of the controlled task. In the cost function various, sometime contradictory requirements can be summarized in order to find most appropriate compromise. The cost function usually contains terms depending on the tracking error and also on the control signal (e.g., the high values of the control signal can be penalized). During the solution of the control task the cost function must be optimized under some constraints. For traditional *Optimal Controllers* the cost function optimization task must be solved over an infinite horizon, usually utilizing *Dynamic Programming* approach as a new formalism in the calculus of variations related to the Hamilton-Jacobi-Bellman equations elaborated by Bellman in 1954 [13]. This solution method is computationally highly demanding and also requires very precise model of the controlled system to prevent the accumulation of the errors due to modeling imprecision. For these reasons in industrial (e.g., [14]–[16]) and also in economical (e.g., [17]) applications the *Model Predictive Control (MPC)* used widely where the optimization is calculated over a finite horizon and only the first part is applied for the controlled system. In *Model Predictive Controllers* the dynamic equations of the controlled system are taken account as the constraint of the optimization task. In the late 1970s [18] special heuristic approach for the *MPC* was introduced, called *Receding Horizon Controller (RHC)* which have better properties (e.g., stability, performance) compared to the traditional *Model Predictive Control* [19] in certain applications. In RHC scheme the time variable is approximated over a discrete time grid as $[t_0, t_1 = t_0 + \Delta t, t_2 = t_1 + \Delta t, ..., t_n = t_{n-1} + \Delta t], n \in \mathbb{N}$ which resolution (Δt) must be fine enough to apply Euler integration over the points. The cost function is calculated at each grid point over a finite horizon. As it was mentioned before in the cost function various requirements can be taken into account besides the tracking error. It can be constructed as the sum of multiple non-negative expressions which are differentiable functions of the system's state variables and the control signal. However, it is worth to be mentioned that these contributions might have negative effect on the tracking precision so the weight contribution of each term must be set accordingly. Also the dynamic model of the controlled system must be taken into account as a set of constraints during optimization of the function. However, the optimization task of such a cost function is much simpler (compared to *Dynamic Programming* in case of traditional *OCs*) to solve by utilizing *Non-Linear Programming* methods. For a system with the following *state propagation equation*

$$\dot{x} = f(x, u) \ , \tag{2}$$

in which $x \in \mathbb{R}^N$ denotes the state variable and $u \in \mathbb{R}^M$ stands for the control signal, the optimization task with the appropriate cost function in grid point ℓ, $\Phi\left(x(\ell), u(\ell)\right)$ formulates as

$$\begin{aligned} &\text{minimize } \sum_{\ell=0}^{HL} \Phi\left(x(\ell), u(\ell)\right) \text{ under the constraints} \\ &\forall i \in \{0, \dots, HL-1\}: \\ &\frac{x(i+1) - x(i)}{\Delta t} - f\left(x(i), u(i)\right) = 0 \ . \end{aligned} \tag{3}$$

In equation (3) $HL \in \mathbb{N}$ denotes the *horizon length* and the $\frac{x(i+1)-x(i)}{\Delta t} - f\left(x(i), u(i)\right) = 0$ term is the constraint which takes into consideration the dynamic equations of the controlled system with some numerical approximation.

The optimization problem can be solved by utilizing *Lagrangian multipliers*. The independent variables of the above described problem are $[x(1), x(2), ..., x(HL)], [u(0), u(1), u(2), ..., u(HL-1)]$ and $[\lambda(0), \lambda(1), \lambda(2), ..., \lambda(HL-1)], \lambda(i) \in \mathbb{R}^N$, since x_0 refers to the initial state of the system which can be measured or observed somehow. Also, due to the law of causality that is formulated in the form of the forward differences, the given state $x(i)$ and force $u(i)$ in the grid point i determines the state variable $x(i+1)$ at the next grid point. By introducing a $\lambda(i)$ *Lagrangian multiplier* for each constraint equation in the i^{th} grid point the reduced gradients formulates as

$$\begin{aligned} &\nabla\left(\sum_{\ell=0}^{HL} \Phi\left(x(\ell), u(\ell)\right)\right) + \\ &\sum_{k=1}^{N} \lambda_k(i) \nabla\left(\frac{x_k(i+1) - x_k(i)}{\Delta t} - f^{(k)}\left(x(i), u(i)\right)\right) \ . \end{aligned} \tag{4}$$

The *Lagrangian multipliers* must be so chosen that the constraint function must stagnate for a small step in the direction of the reduced gradient. This way optimization problem can be solved by applying *Lagrange's Reduced Gradient* algorithm, as move with small steps along the Reduced Gradient until it becomes 0. Then a local solution of the constrained optimization problem has reached. Assuming that the constraint equations have solution, the initial point for the optimization can be found by utilizing *Newton-Raphson* algorithm [20]. In the practice, for evading the accumulation of the effects of external disturbances and modeling imprecision, only the control signal $(u(i))$ to be exerted in the first point of the horizon is taken into account, and the system's state actually achieved by this force $(x(i+1))$ has to be directly measured or somehow observed in the starting point of the next horizon.

In the present paper it will be shown how the above described formalism can be used for the solution of the inverse kinematic task for a redundant manipulator. Also, since we do not wish to utilize any physical analogy for the *Lagrange Multipliers* (in some cases the Lagrenge multipliers might have important physical meaning e.g., [21]) instead of the *Reduced Gradient Algorithm* a simpler *Gradient Descent Algorithm* is used that way getting rid of gradient reduction.

III. Receding Horizon Scheme-based Approach for Solving the Inverse Kinematic Task

As it was mentioned before when formulating cost functional for our task, huge control forces can be penalized in *Receding Horizon Control* algorithm. This property can be very useful during the solution of the inverse kinematic task since near to singular positions, as $\dot{x}$ approaches the null space of J^T, $J^T\dot{x} \to 0$, therefore $\dot{q} \to \infty$ must compensate this effect, resulting in very high joint velocities.

The above described formalism can be applied for the solution of the inverse kinematic task in the following manner, in place of the state propagation equation (2), the non-differential mapping of the forward kinematic task $F(q) : \mathbb{R}^n \mapsto \mathbb{R}^6$ can be placed and the u control signal is replaced the possible values of joint velocities $\dot{q} \in \mathbb{R}^n$, which are the independent variables of the optimization task. That way a simple Euler integration can be applied over the grid points as

$$q(i+1) \approx q(i) + \Delta t \dot{q}(i) \ , \qquad (5)$$

from the starting point $q(0)$ that is given in advance (in practice it can be measured by the encoders in each joint of the manipulator). Next the cost function can be constructed which has two terms, one penalizes the trajectory tracking error, which can be calculated based on the $X^N(i) - F(q(i))$ error terms, and the other one penalizes the high joint velocity values $(\dot{q}(i))$. The independent variables of the so constructed cost function are the $[\dot{q}(0), \dot{q}(1), ..., \dot{q}(HL-1)]$ values where HL denotes the prediction horizon length. Also, since no constraint applied for the cost function the gradient reduction and the application of the *Lagrange Multipliers* can be evaded and a simpler *Gradient Descent Algorithm* can be used to minimize the cost function. For realizing this program an appropriate representation of the "pose" coordinates of the tool must be elaborated. It can be done as follows.

Traditionally the pose of the tool is represented by a *Homogeneous Transformation Matrix*, the left upper 3×3 part of which describes the orientation of the tool and the right upper 3×1 part represents position of the *Tool Center Point (TCP)* and in the 4^{th} row [0, 0, 0, 1] components are placed as

$$H(\xi, e, r) \equiv \left[\begin{array}{c|c} O(\xi, e) & r \\ 0^T & 1 \end{array}\right] \ . \qquad (6)$$

Such a representation can be very useful during the solution of the different kinematic tasks since with *Homogeneous Transformation Matrices* the rotations and shifts can be described simultaneously, however, it is a highly redundant representation of the pose of the tool. The 3×3 rotation matrix contains only 3 independent elements which can be easily extracted by considering the following equations. As it is well known every quadratic matrix can be reconstructed from a symmetric and an skew-symmetric part as

$$O = \frac{1}{2}\left(O + O^T\right) + \frac{1}{2}\left(O - O^T\right) \ . \qquad (7)$$

Also, by utilizing that the odd powers of the generator (G) of the rotation are $G^{2n+1} = -G$, $n \in \mathbb{N}$, the odd and even powers of G can be separated in the power series of $\exp\left(q_i G^{(i)}\right)$ so the Rodrigues formula [22] is obtained in (8) as

$$O = \exp(qG) = I + \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \sin q + \\ + \begin{bmatrix} e_1^2 - 1 & e_1 e_2 & e_1 e_3 \\ e_2 e_1 & e_2^2 - 1 & e_2 e_3 \\ e_3 e_1 & e_3 e_2 & e_3^2 - 1 \end{bmatrix} (1 - \cos q) \ , \qquad (8)$$

$$G = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \ , \ \sum_{i=1}^{3} e_i^2 = 1 \ .$$

From the above equations it can be clearly seen that the 3 independent element of the rotation matrix can be expressed from the skew-symmetric part, $O^A = \frac{1}{2}\left(O - O^T\right)$. On this basis a *"model function"* $F : \mathbb{R}^n \mapsto \mathbb{R}^6$ is used for describing the forward kinematic task as $F_1(q_1, \ldots, q_n) = O^A_{12}$, $F_2(q_1, \ldots, q_n) = O^A_{13}$, $F_3(q_1, \ldots, q_n) = O^A_{23}$, $F_4(q_1, \ldots, q_n) = r_1$, $F_5(q_1, \ldots, q_n) = r_2$, and $F_6(q_1, \ldots, q_n) = r_3$. In the horizon the following elements are filled in:

- The "Nominal Cartesian Trajectories" $X^N(1), \ldots, X^N(HL)$ that are known in advance since definite ideas are available for the future of the nominal motion;
- The also known initial joint coordinates in the first grid point $q(1)$ are taken from the last "actual" value (in the control applications it is measured or somehow "observed");
- By using the $\dot{q}$ values in the grid points $\{\dot{q}(1), \ldots, \dot{q}(NL-1)\}$, with Euler integration, fill in the horizon points as $q(i+1) = q(i) + \Delta t \dot{q}(i)$ for $i = 1, \ldots, HL-1$;
- By the use of the *model function* compute the *assumed future Cartesian values* $X^O(i) = F(q(i))$ for $i \in \{2, \ldots, HL\}$;
- Compute the cost functions over the grid points $i \in \{2, \ldots, HL\}$ and the penalty values for the $\dot{q}$ components

over the grid points $i \in \{1, \dots, HL-1\}$ as

$$\Phi = \sum_{j=1,i=2}^{6,HL} \Phi_j^X(i) \; + \sum_{j=1,i=1}^{6,HL-1} \Phi_j^U(i)$$

$$\Phi_j^X(i) = \begin{cases} \left|\frac{X_j^N(i)-X_j^O(i)}{\Delta x}\right|^{P_X} & \text{if } \left|\frac{X_j^N(i)-X_j^O(i)}{\Delta x}\right| > 1 \\ \left|\frac{X_j^N(i)-Xj^O(i)}{\Delta x}\right|^2 & \text{if } \left|\frac{X_j^N(i)-X_j^O(i)}{\Delta x}\right| \le 1 \end{cases}$$

$$\Phi^U(i) = \begin{cases} \left|\frac{\dot{q}(i)}{\Delta \dot{q}}\right|^{P_{\dot{q}_{out}}} & \text{if } \left|\frac{\dot{q}(i)}{\Delta \dot{q}}\right| > 1 \\ \left|\frac{\dot{q}(i)}{\Delta \dot{q}}\right|^{P_{\dot{q}_{in}}} & \text{if } \left|\frac{\dot{q}(i)}{\Delta \dot{q}}\right| \le 1 \end{cases}$$

- The so computed cost function can be minimized by using the simple *Gradient Descent Algorithm* so that the components of $\nabla\Phi$ can be arranged in a matrix of size $\mathbb{R}^{6\times(HL-1)}$;
- The starting point of the next horizon will be the element $q(2)$ of the so optimized horizon, and the joint coordinates' time-derivatives can be estimated by the value that is found in $\dot{q}(1)$ of the present horizon after the optimization.

In the cost contribution $\Phi_j^X(i)$ the component tracking errors that in absolute value are greater than Δx are very strongly penalized if $P_X > 1$, and for smaller error components the usual quadratic tracking rule is prescribed. In similar manner, in the term $\Phi_j^U(i)$, the joint coordinate time-derivatives are very seriously penalized if their absolute value is bigger than $\Delta\dot{q}$ and $P_{\dot{q}_{out}} > 1$, but the small values' penalty contribution is very small if $P_{\dot{q}_{in}} > 1$. Therefore it is expected that by the use of these "shape parameters" the trajectory tracking precision can be "relaxed" if the high $\dot{q}$ values are seriously penalized. In the sequel simulation results will be shown and the operation of the above idea will be discussed on the basis of the computations.

IV. The Simulation Results

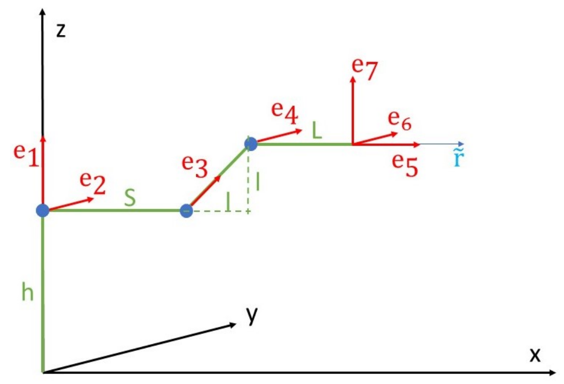


Fig. 1. The kinematic structure and parameters of a 7Dof robot arm in *"Home Position"*

In this paper a 7DoF, redundant robot arm with six rotational (r) and one prismatic (s) joint (arranged as [r, r, s, r, r, r, r]) is analyzed via Simulation programs made in Julia programming language [23]. The basic kinematic structure of the robot arm is shown in Fig. 1. In *"Home Position"* where all the joint coordinates are 0 the unit vectors of the joint axels (e_i) can be described by matrix E in which each column represents a unit vector, furthermore, the length parameters of each segment are also arranged into a matrix R,

$$E = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} , \tag{9}$$

$$R = \begin{pmatrix} 0 & 0 & S & l & L & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h & 0 & 0 & l & 0 & 0 & 0 \end{pmatrix} , \tag{10}$$
$$S = 2m, h = l = 1m, L = 1.5m$$

and the homogeneous coordinates of the vector pointing into the *Tool Center Point* are $\tilde{r} = [1, 0, 0, 1]$. The time-resolution of the grid was $\Delta t = 10^{-3}\,s$, the horizon length was $HL = 3$, the gradient descent algorithm was stopped when the $\|\nabla\Phi\| \le \mu = 20$ was achieved, the components of the gradient were estimated with a finite step length $\delta\dot{q}_j(i) = 10^{-3}$ for each j, i index pairs. The maximum allowed number of steps in the optimization was 450. The Gradient Descent algorithm was realized as $\dot{q}(next) = \dot{q}(now) - \alpha_2\nabla\Phi(now)$ with $\alpha_2 \in [10^{-8}, 5\cdot10^{-4}]$. To achieve fast convergence, the actual value of α_2 was increased by the factor 1.2 for the next cycle if $\|\nabla\Phi\|$ was reduced in the previous step, but it was decreased by dividing it with 1.2 if this value have increased. However, its value was kept between the above given range. In the first series a relatively fast trajectory was tracked by using the above parameters. In a basic settings $\Delta x = 10^{-3}\,rad$ or m, $P_X = 3$, $P_{\dot{q}_{out}} = P_{\dot{q}_{in}} = 4$, $\Delta\dot{q} = 0.8\,rad\cdot s^{-1}s$ or $m\cdot s^{-1}$ were chosen. For the simulation the trajectory to be tracked was generated in the joint space using the equation below

$$q_i^{Nom} = A_i \sin(\omega t) \ , \tag{11}$$
$$A = [0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0], \omega = 3 \ .$$

Then by utilizing the forward kinematic equations the nominal trajectory was mapped to the work space, $X^{Nom} = F(q^{Nom})$. Figures 2 and 3 reveal that in this manner it was possible to track a relatively fast motion with acceptable precision. Figure 4 shows that the nominal and the realized trajectories are different so the solution is ambiguous and an optimal one was chosen based on the requirements formulated in the cost function. However, Fig. 5 shows considerable fluctuation in the joint velocities which might cause problem in real applications.

The Julia programming language also makes it possible the measure time need of our calculations. The Fig. 6 shows nicely that each step over the horizon takes around $1s$ and it is proportional to the amount of step done during the gradient descent algorithm. In most of the time maximal number of allowed steps was reached. It can be concluded that this solution method is very time consuming, but it might not be a problem during offline applications. The simulations were running from a Lenovo X250 Laptop, under Windows10 operating system.

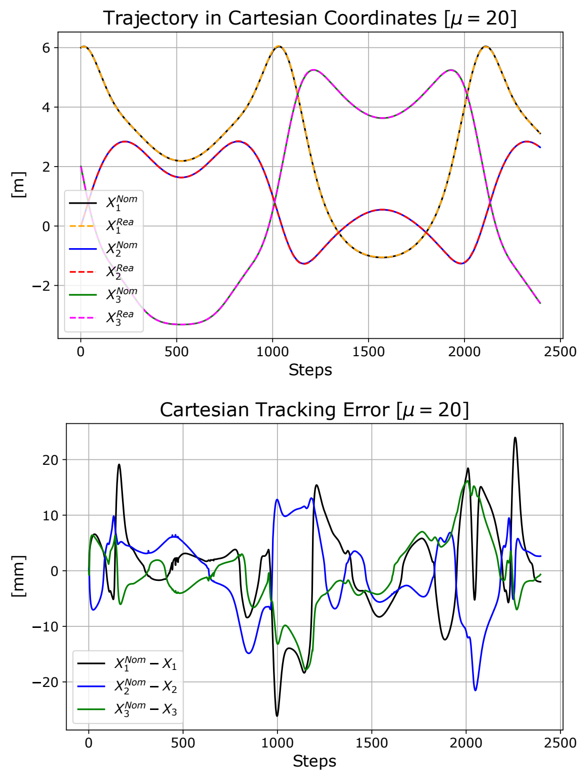


Fig. 2. Cartesian coordinates (up) and Cartesian tracking error (down) for fast motion

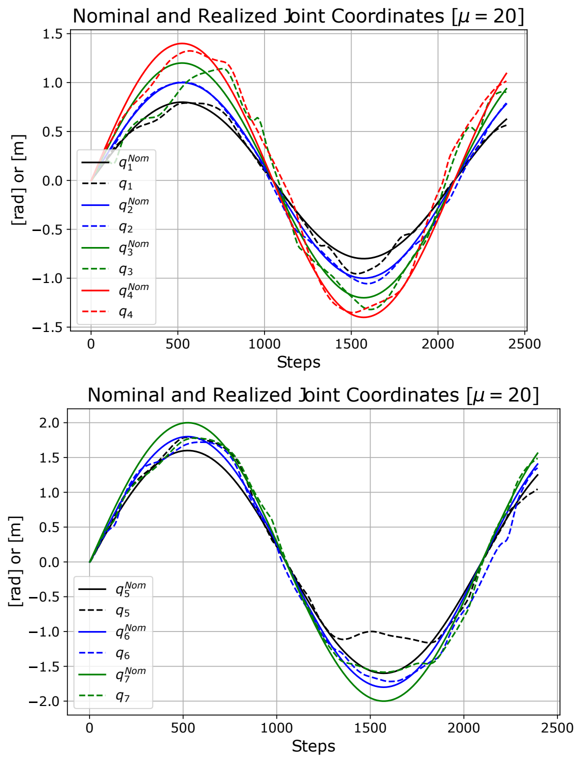
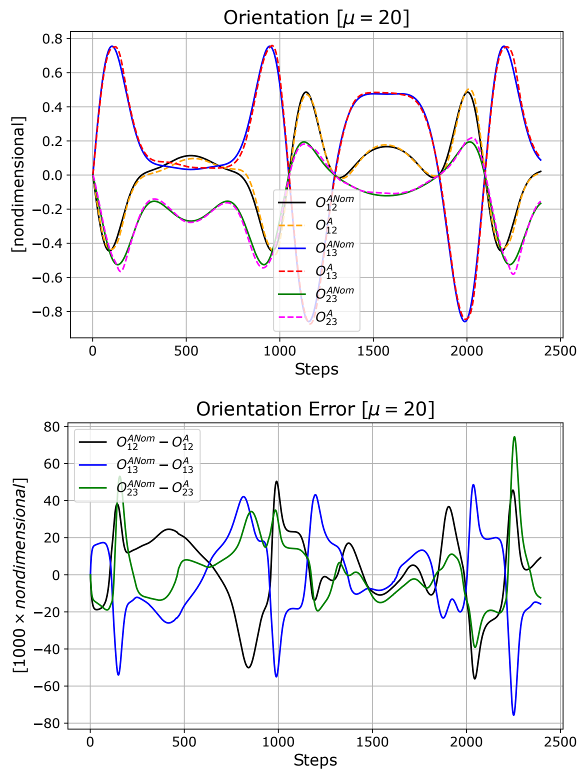


Fig. 4. Realized and Nominal joint coordinates



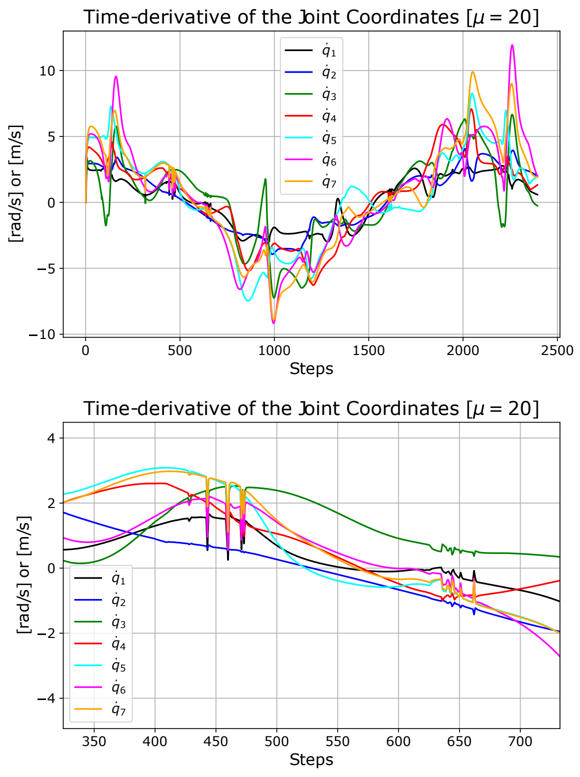


Fig. 5. Realized joint velocities (up) and zoomed in to a domain where considerable fluctuations occur (down)

Fig. 3. Orientation tracking (up) and orientation tracking error (down) for fast motion

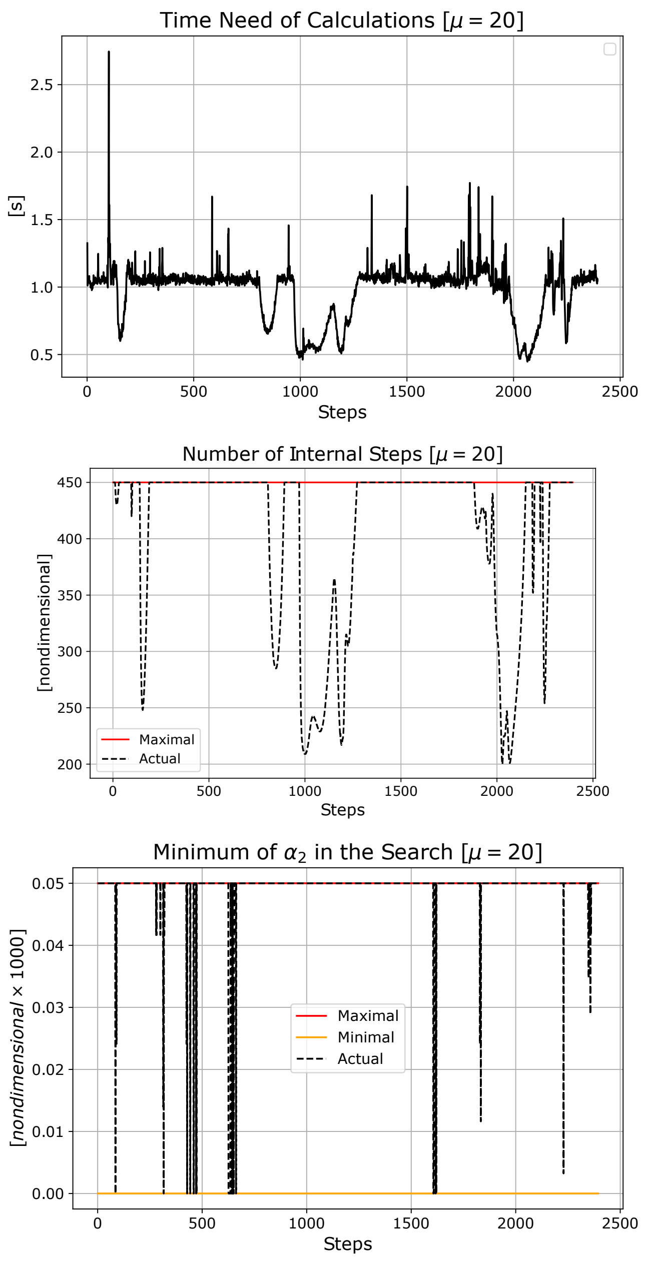


Fig. 6. Time need of the calculations (up) amount of steps made during the gradient descent algorithm (middle) and values at each step (down)

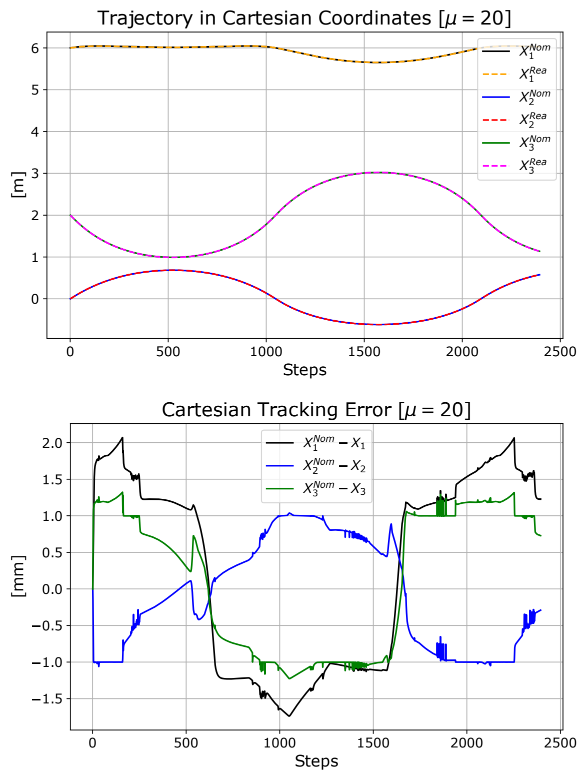


Fig. 7. Cartesian coordinates (up) and Cartesian tracking error (down) for slow motion

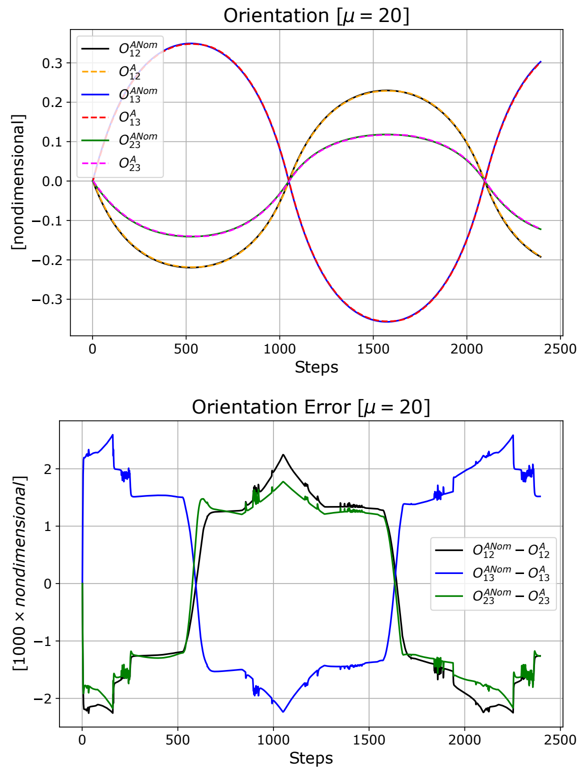


Fig. 8. Orientation tracking (up) and orientation tracking error (down) for slow motion

In the next step a much slower motion was analyzed. For generating the trajectory of the slower motion a sigmoid function was intduced, $f(x) = \frac{x}{1+|x|}$ and the trajectory in the joint space was generated as follows

$$q_i^{Nom} = Sf(A_i \sin(\omega t)\ ), \qquad (12)$$

$$A = 0.5 \cdot [0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0], \omega = 3, S = 0.3\ .$$

The results given in Figs. 7, 8 9 and 10 reveal that for much slower motion more precise trajectory tracking can be achieved. Since the joint velocity limitations has less affect on the tracking error, on the other hand more significant fluctuations occurred in the joint velocities.

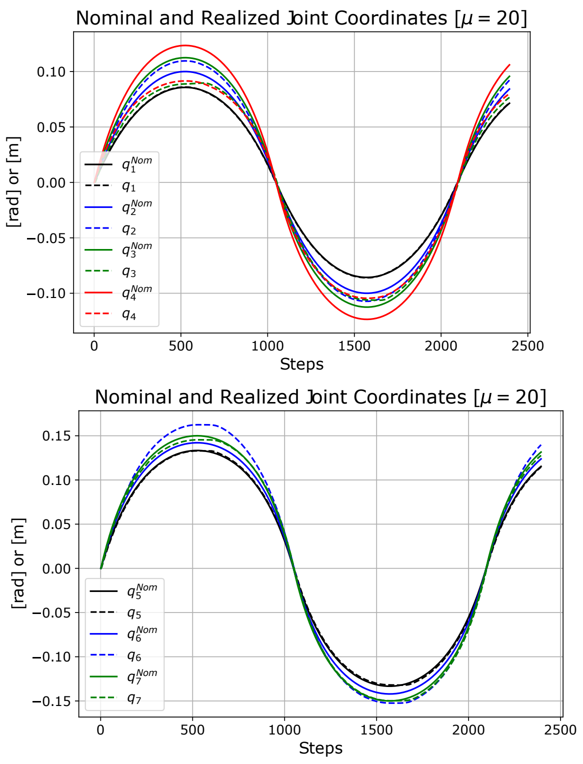


Fig. 9. Realized and Nominal joint coordinates

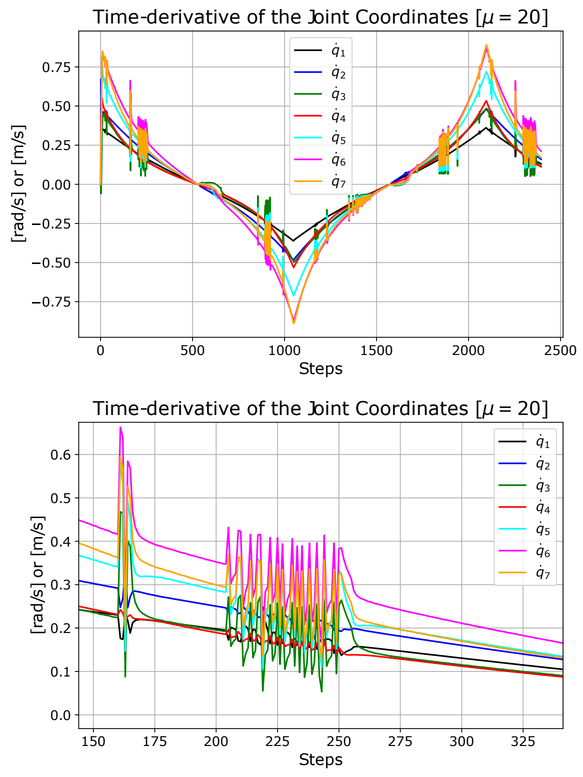


Fig. 10. Realized joint velocities for the slow motion (up) and zoomed in to a domain where considerable fluctuations occur (down)

V. Second Order Trajectory Smoothing

As it was shown in the previous section the control feedback caused high fluctuation of the derivatives of the joint coordinates (5), which is obviously getting worse when the second order derivative ($\ddot{q}$) is considered. It may result in unwanted vibrations in the mechanical system also may cause serious problem during the control design. These sudden changes in the derivatives might make the trajectory tracking for the control algorithm impossible and also stability issues may occur. To resolve these issues a second order smoothing function suggested in this section. Lets introduce $\mathbb{R} \ni \Lambda > 0$ constant *"error relaxation time constant"* and a $g(t) \in \mathbb{R}^n$ noisy function which must be tracked by function $f(t) \in \mathbb{R}^n$. By utilizing the the differential operator $\left(\Lambda + \frac{\mathrm{d}}{\mathrm{d}t}\right)^n, n \in \mathbb{N}$ the following set of differential equations can be written

$$\left(\Lambda + \frac{\mathrm{d}}{\mathrm{d}t}\right)^n f(t) = \Lambda^n g(t) \ . \tag{13}$$

If $g(t) \equiv const.$ (or it is changing very slowly) then the derivatives of $f(t)(\dot{f}(t), \ddot{f}(t) etc.)$ must be zero, that way it can be obtained that $\Lambda^n f(t) = \Lambda^n g(t)$. For higher frequency signals Laplace transform of (13) can be investigated,

$$(\Lambda + s)^n F(s) = \Lambda^n G(s) \ . \tag{14}$$

By utilizing the Binomial Theorem (14) can be rewritten as,

$$F(s) = \frac{1}{\sum_{k=0}^{n} \frac{n!}{k!(n-k)!} \Lambda^{n-k} s^k} \Lambda^n G(s) \ . \tag{15}$$

From (15) can be seen that for low frequencies when $s \approx 0$ the tracking signal is nicely following the *"noisy"* signal, $g(t) = f(t)$. However, for higher frequencies the noisy signal is approximately damped by factor $\frac{1}{s^n}$. Basically it means that the higher frequency values of signal $g(t)$ are cut off so the smoothed trajectory can be utilized for control purposes. Instead of $g(t)$ the calculated $q(t)$ values are written, which have high fluctuation so they are "noisy". The $f(t)$ function is replaced by the smoothed trajectory (q^{Smo}) and second order differential equation is written based on (13) as $\left(\Lambda + \frac{\mathrm{d}}{\mathrm{d}t}\right)^2 q^{Smo}(t) = \Lambda^2 q(t)$ from which the $\ddot{q}^{Smo}$ can be expressed:

$$\ddot{q}^{Smo}(t) = \Lambda^2 \left(q(t) - q^{Smo}(t)\right) - 2\Lambda q^{Smo} \ . \tag{16}$$

Than the above described differential equation can be easily solved by applying Euler integration

$$\begin{aligned} \dot{q}^{Smo}(t + \Delta t) &= \dot{q}^{Smo}(t) + \Delta t \ddot{q}^{Smo}(t) \ , \\ q^{Smo}(t + \Delta t) &= q^{Smo}(t) + \Delta t \dot{q}^{Smo}(t) \ . \end{aligned} \tag{17}$$

For simulation results presented down below the smoothing parameter $\Lambda = 300\frac{1}{s}$. From Fig. 11 can be seen that the simulations result meet our expectations and the fluctuation of the joint velocities is significantly reduced. However, Figs. 13 and 14 show that for the smoothed trajectories the tracking error is significantly higher, although it is worth to mention that for fast motions this precision can acceptable.

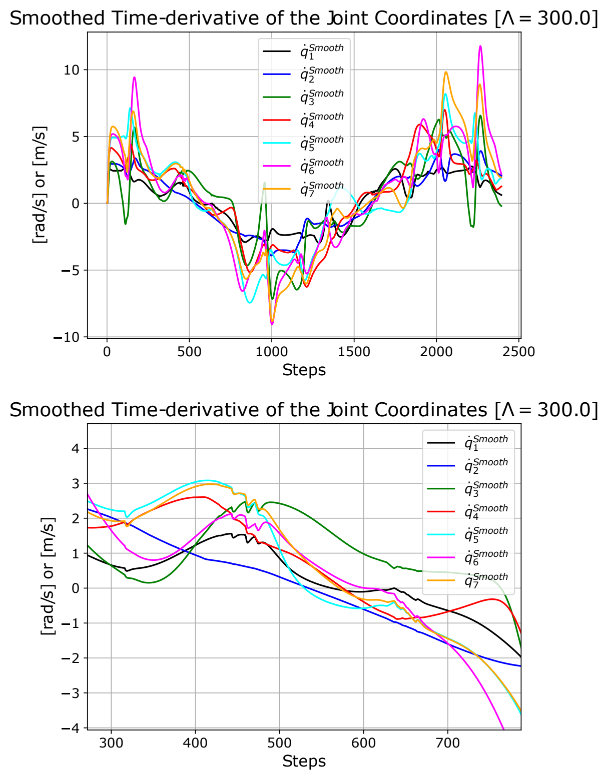


Fig. 11. Smoothed Joint velocities (up) and zoomed in to the domain as in Fig. 5 (down)

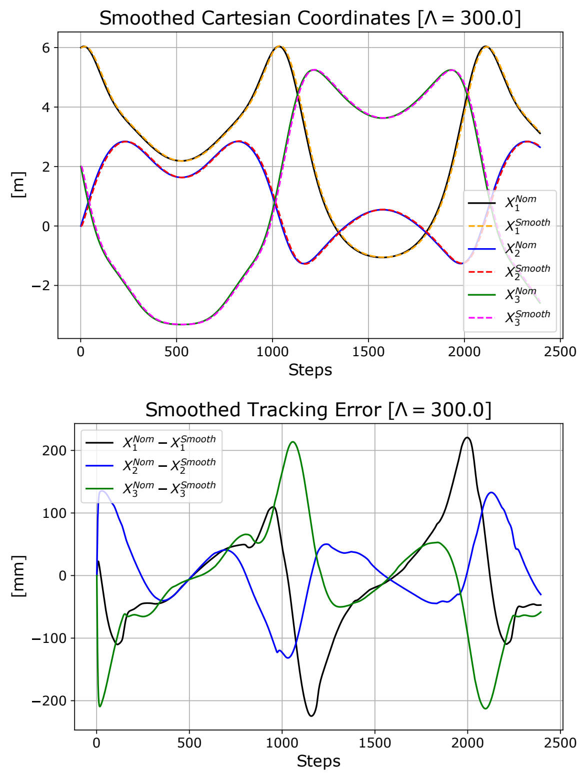


Fig. 13. Cartesian coordinates (up) and Cartesian tracking error (down) for fast motion and smoothed joint velocities

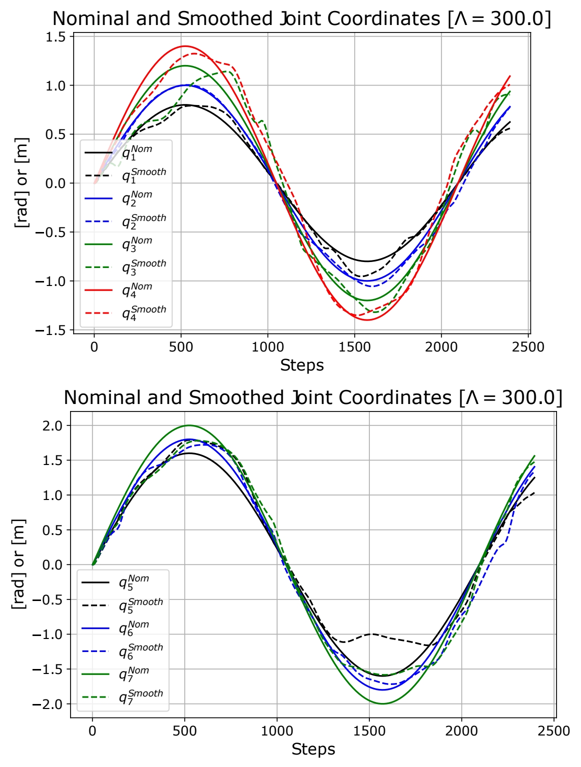


Fig. 12. Realized and Nominal joint coordinates for fast motion and smoothed joint velocities

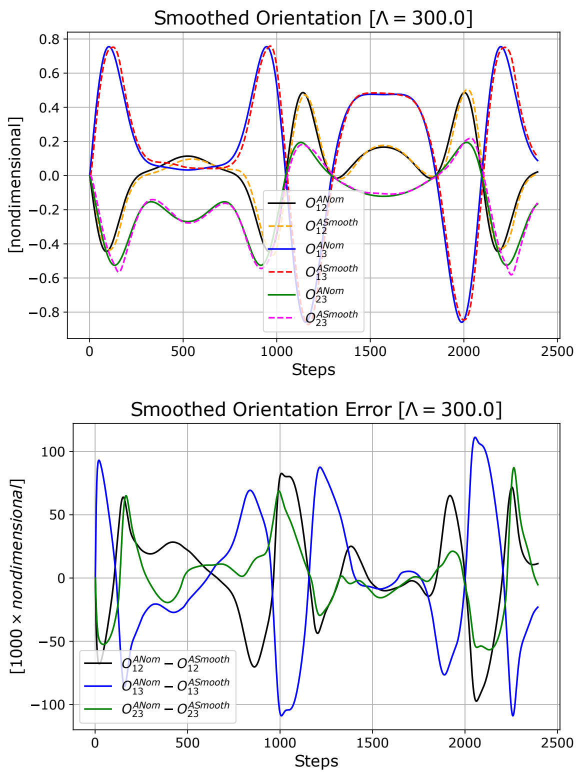


Fig. 14. Orientation tracking (up) and orientation tracking error (down) for fast motion and smoothed joint velocities

VI. Conclusion

In this paper a flexible approach was considered for the solution of the inverse kinematic task of redundant manipulators with open kinematic chain. Instead of the traditional generalized inverse and quadratic cost function optimization-based solution a Receding Horizon Scheme-based approach was characterized. The inverse kinematic task was formulated without using constraint functions and associated Lagrange Multipliers so the gradient reduction has been evaded. Such a solution can be very useful as in the cost function multiple, sometimes contradictory requirements can be taken into account with different weight contribution which makes this solution very flexible. Through some simulation results it was shown that such solution can provide acceptable tracking errors even for higher velocities. However, during the simulations two issues occurred. One of them is that in some steps high fluctuation of joint velocities could be observed which can result in unwanted vibrations in the mechanical system and also can cause problems during control design. To resolve this problem a smoothing function was introduced in this paper. The other one is that this solutions is computationally very demanding so it is possible that such a solution mainly can be used for offline applications.

Acknowledgment

This research was supported by the Doctoral School of Applied Informatics and Applied Mathematics of Óbuda University.

References

[1] H. Issa, B. Varga, and J.K. Tar. A receding horizon-type solution of the inverse kinematic task of redundant robots. *In proc. of the IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI 2021), May 19-21, 2021 in Timisoara, Romania.*, pages 231–236, 2021.

[2] 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.

[3] A. Benitez, I. Huitzil, A. Casiano, J. De La Calleja, and M.A. Medina. Puma 560: Robot prototype with graphic simulation environment. *Advances in Mechanical Engineering*, 2(1):15–22, 2012.

[4] P. Chang. A closed-form solution for the control of manipulators with kinematic redundancy. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 9–14. IEEE, 1986.

[5] E.H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26(9):394–395, 1920.

[6] R. Penrose. A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society*, 51:406–413, 1955.

[7] A. Ben-Israel and T.N.E. Greville. *Generalized Inverses*. Springer-Verlag, 2003.

[8] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[9] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11(2):431–441, 1963.

[10] J. Baillieul. Kinematic programming alternatives for redundant manipulators. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 722–728. IEEE, 1985.

[11] D.N. Nenchev. Restricted jacobian matrices of redundant manipulators in constrained motion tasks. *The International journal of robotics research*, 11(6):584–597, 1992.

[12] J. Baillieul. Avoiding obstacles and resolving kinematic redundancy. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1698–1704. IEEE, 1986.

[13] R.E. Bellman. Dynamic programming and a new formalism in the calculus of variations. *Proc. Natl. Acad. Sci.*, 40(4):231–235, 1954.

[14] A. Reda, A. Bouzid, and J. Vásárhelyi. Model predictive control for automated vehicle steering. *Acta Polytechnica Hungarica*, 17(7):163–182, 2020.

[15] N. Moldoványi. *Model Predictive Control of Crystallisers*. PhD Thesis, Department of Process Engineering, University of Pannonia, Veszprém, Hungary, 2012.

[16] S. Rohani, M. Haeri, and H.C. Wood. Modeling and control of a continuous crystallization process Part 2. Model predictive control. *Computers & Chem. Eng.*, 23:279, 1999.

[17] N. Muthukumar, Seshadhri Srinivasan, K. Ramkumar, K. Kannan, and Valentina Emilia Balas. Adaptive model predictive controller for web transport systems. *Acta Polytechnica Hungarica*, 13(3):181–194, 2016.

[18] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, 1978.

[19] W.H. Kwon and S. Han. *Receding Horizon Control*. 01 2005.

[20] T.J. Ypma. Historical development of the Newton-Raphson method. *SIAM Review*, 37(4):531–551, 1995.

[21] H. Karabulut. Physical meaning of Lagrange multipliers. *European Journal of Physics (physics.ed-ph); General Physics (physics.gen-ph)*, 27:709–718, 2007.

[22] O. Rodrigues. Des lois géometriques qui regissent les déplacements d' un systéme solide dans l' espace, et de la variation des coordonnées provenant de ces déplacement considérées indépendent des causes qui peuvent les produire (Geometric laws which govern the displacements of a solid system in space: and the variation of the coordinates coming from these displacements considered independently of the causes which can produce them). *J. Math. Pures Appl.*, 5:380–440, 1840.

[23] J. Bezanson, A. Edelman, S. Karpinski, and V.B. Shah. Julia. *https://julialang.org*, Last time visited: 5 May 2019.