43. JUPITER konferencija 39. simpozijum

Beograd 2022 NU * ROBOTI * FTS

Erwinski, K., Karasek, G. [1]**, Živanović, S.** [2]**, Dimic, Z.** [3]**, Slavkovic, N.** [2]

REAL-TIME CONTROL OF A KEOPS-DELTA PARALLEL KINEMATICS MACHINE USING LINUXCNC AND ETHERCAT

Abstract

This paper presents a laboratory stand for investigating trajectory optimization algorithms for non-cartesian numerically controlled machines. The stand consists of a Delta machine in KEOPS configuration with linear motors controlled by high performance servo-drives. The machine is controlled by real-time control system with LinuxCNC software. The control is performed via real-time communication bus EtherCAT. The paper also describes the extension of the LinuxCNC control system with NURBS interpolaion and s-curve feedrate profiling. Also research to be performed on the machine is discussed concerning development of trajectory optimization algorithms for parallel kinematics machines.

Key words: *parallel kinematics, real-time Linux, LinuxCNC, Delta, KEOPS*

1. INTRODUCTION

Numerically controlled machines are an important part of modern manufacturing industry. Most of these machines are cartesian machines in which linear axes correspond to axes of the cartesian coordinate system. These axes are often serially linked. Recently multi-axis machines with parallel kinematics configuration are becoming more popular. In parallel kinematics machines the joints are directly coupled to the end effector but usually do not correspond to the cartesian axes. This allows to achieve highly dynamic motion with high speeds and accelerations. This is especially useful in applications which require highly dynamic motion such as laser engraving and pick and place There are numerous types of parallel kinematics machines of which the Delta kinematics is often used in industry.

In Delta kinematics three motors are coupled to the end effector usually via double rods connected by spherical joints [1]. The motors can be rotary or linear. The first type is often used by pick and place robots. The second type is also used in numerical machines such as milling machines or 3d printers due to higher stiffness and better precision. The linear axes can be vertical as in 3d printers, horizontal or angled [2]. Delta kinematics with angled linear axes is known as KEOPS-Delta or Tri-Pyramid kinematics [3]. The linear axes form a tetrahedron and are usually located 120 degrees apart on the base plate. This type provides a good compromise between vertical and horizontal stiffness and significantly better stiffness than a rotary delta machine. The linear axes can be driven by rotary motors with linear motion generated by a toothed belt or ball-screw. This introduces additional inaccuracies, mass and complicates the machine design. An alternative solution is to use linear motors which is presented in this paper.

In this paper a numerical control system is developed for a linear motor driven parallel kinematics machine in KEOPS-Delta configuration. Use of linear motors greatly simplifies the mechanical part of the machine. Because of the end effector rods are driven directly by linear motor forcers the stiffness and accuracy of the machine should also improve. Further accuracy improvement is achieved because each linear motor forcer is directly coupled to a linear encoder.

The paper describes the KEOPS Delta machine including the direct and inverse kinematic transformations. The Linux PC-based control system which is capable of Non-Uniform Rational B-Spline (NURBS) toolpath

[1] dr Krystian Erwinski (erwin@umk.pl), Gabriel Karasek (gabkaras@umk.pl), Institute of Engineering and Technology, Faculty of Physics Astronomy and Informatics, Nicolaus Copernicus University, Torun, Poland

[2] prof. dr Saša Živanović (szivanovic@mas.bg.ac.rs), dr Nikola Slavković (nslavkovic@mas.bg.ac.rs) , University of Belgrade, Faculty of Mechanical Engineering

[3] dr Zoran Dimić, Lola Institute, Belgrade, Serbia, (zoran.dimic@li.rs)

interpolation capability is also described. The last section presents future research planned to be conducted on the research station.

2. DELTA-KEOPS MACHINE AND CONTROL SYSTEM

The parallel kinematics machine in KEOPS-Delta configuration, for which a control system was developed is presented in Figure1. The machine consists of three linear axes driven by Tecnotion UXA3S coreless linear motors with a nominal force of 120N each. Each linear axis is integrated with a linear absolute magnetic encoder (Renishaw LA11) with 1μm resolution and BiSS-C serial interface. The motors are controlled by Beckhoff AX8206 servo drives which communicate over EtherCAT fieldbus [4]. The linear axes are arranged in the form of a tetrahedron with an inclination angle of 45 degrees. Each axis has a movement range of 580mm.

In order to control any numerically controlled machine a centralized drive control system has to be used. Such a system generally consists of a graphical user interface (GUI), program interpreter, trajectory generation block, trajectory interpolation block and drive interface. The program interpreter converts the program to an internal path reference. The trajectory generation algorithm generates a feedrate profile for the desired path. The interpolator block generates consecutive positions on the toolpath which serve as machine position commands. The interpolator outputs are then processed by the inverse kinematics module to generate drive position commands

The Delta KEOPS machine is controlled by a PC with a real-time kernel Linux operating system and LinuxCNC control software. The real-time kernel provides time determinism for the control processes by prioritizing different critical and non-critical processes. The time-critical processes responsible for machine control cannot be preempted by lower priority processes such as handling the user interface which ensures that consecutive position commands will be sent to the drives at equal intervals regardless of system load. In this case the RT-Preempt real-time Linux kernel is used, however the system is capable of using the more mature RTAI kernel [5].

LinuxCNC is an open-source software for controlling Computerized Numerical Control machines. The software integrates a CNC GUI, G-Code interpreter, trajectory generation algorithm and interpolator. Furthermore the software provides an API to interface with the specialized real-time functions of RT-Preempt and RTAI Linux kernels. The API called HAL also facilitates creation of custom user real-time and non-real-time modules. These modules are created in C programming language and can be easily integrated with the rest of the system by using dedicated shared variables. The software also provides several kinematic transformations for 5-axis machines, robots, rotary and linear delta machines. The KEOPS-Delta kinematics transformation is not supported by default and had to be implemented by the authors. The equations describing the KEOPS kinematic transformation are presented in section 3. A block schematic of the control system and the picture of the actual KEOPS-Delta machine is presented on Figure 1.

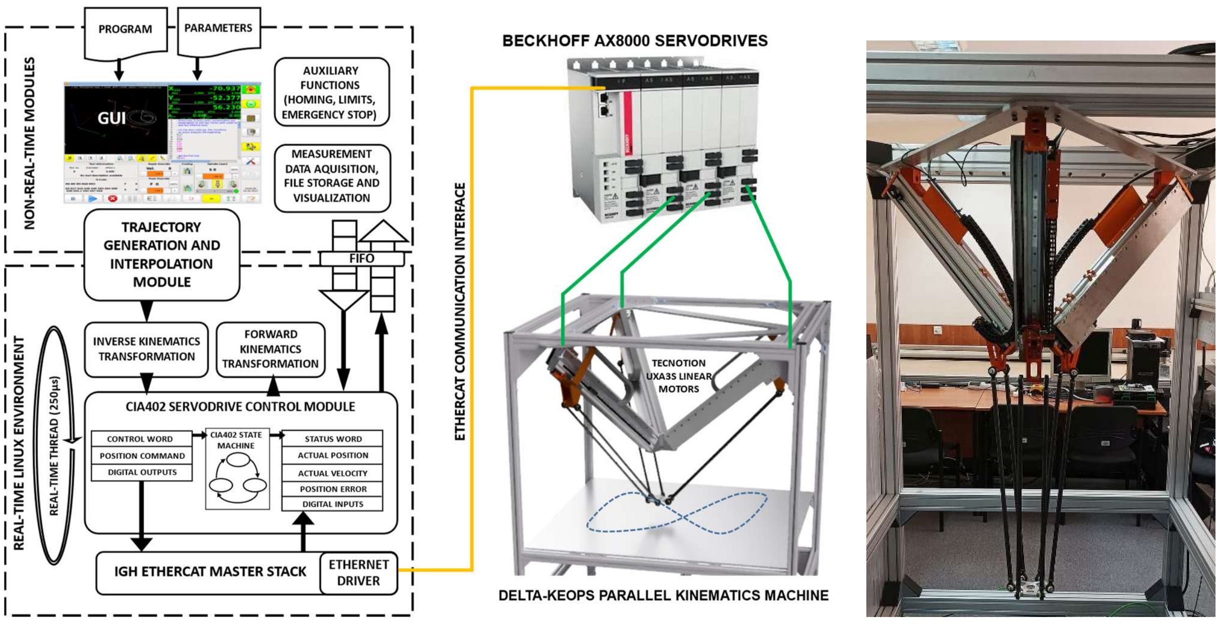
PROGRAM
PARAMETERS
NON-REAL-TIME MODULES
GUI
AUXILIARY FUNCTIONS (HOMING, LIMITS, EMERGENCY STOP)
MEASUREMENT DATA AQUISITION, FILE STORAGE AND VISUALIZATION
TRAJECTORY GENERATION AND INTERPOLATION MODULE
FIFO
REAL-TIME LINUX ENVIRONMENT
REAL-TIME THREAD (250µs)
INVERSE KINEMATICS TRANSFORMATION
FORWARD KINEMATICS TRANSFORMATION
CIA402 SERVODRIVE CONTROL MODULE
CONTROL WORD
POSITION COMMAND
DIGITAL OUTPUTS
CIA402 STATE MACHINE
STATUS WORD
ACTUAL POSITION
ACTUAL VELOCITY
POSITION ERROR
DIGITAL INPUTS
IGH ETHERCAT MASTER STACK
ETHERNET DRIVER
ETHERCAT COMMUNICATION INTERFACE
BECKHOFF AX8000 SERVODRIVES
TECNOTION UXA3S LINEAR MOTORS
DELTA-KEOPS PARALLEL KINEMATICS MACHINE

Figure 1. *Block schematic of the control system* (left) and picture of the KEOPS-Delta machine

The software does not provide any modern industrial communication interface. In order to communicate with the servo drives the EtherCAT communication stack was implemented in software. EtherCAT is an industrial real-time communication bus based on 100Mbps Ethernet physical layer. Due to its compatiblity with standard Ethernet cards the master stack can be implemented on a standard PC. Time deterministic communication is achieved by precise synchronization between slave nodes and the master as well as on-the-fly processing of the protocol frame by each slave. Due to real-time requirements it is critical that the stack is implemented on a platform with a real-time operating system.

At the application layer EtherCAT utilizes the CANOpen standard which is commonly used in industry with CAN BUS. The CANOpen standard defines a set of objects that form the Object Dictionary. Each object has an address (index) and possibly a subaddress (subindex) that identifies each configuration or process variable. These can be mapped to a Process Data Object (PDO) which is a data structure of process variables that are transmitted and recieved between the master and slaves at each communication cycle. Most EtherCAT drives including the AX8000 series require the master to follow the Can in Automation 402 device profile which defines standard Object Dictionary entries common to all drives regardless of the manufacturer. There is also a standard state machine which governs each drive and can be controlled by the master via standardized Control Word and Status Word transmitted cyclically [6].

In order to integrate EtherCAT with LinuxCNC an open-source stack IGH EtherCAT Master [7] was used. The stack uses standard Ethernet cards and system drivers but utilizes the capabilities of the real-time kernel to deterministically communicate with servo-drives. The compiled stack is a kernel module which provides an API to map chosen variables into the Process Data Object and send and recieve it over the network. In order to access this API a dedicated module was developed by the authors in the LinuxCNC HAL that implements variables compatible with the CiA402 standard such as Control Word, Status Word, Position Demand, Actual Position etc. These variables are made available to the LinuxCNC HAL as shared variables (also called pins) which can be connected to any other module such as those responsible for path interpolation or kinematics transformation. A 250μs real-time thread is created to which all functions of modules related to machine control are attached.

3. Delta KEOPS kinematics

Most machine control system perform trajectory planning and interpolation in cartesian space. For any machine with non-cartesian kinematics an inverse kinematics transformation is required to convert position command values in cartesian space to drive commands in axis/joint space. Forward kinematics transformation is used to convert axis positions to cartesian positions which is useful for actual path visualisation and control.

The Delta-KEOPS kinematics is derived by assuming that each linear axis is equally spaced at 120 degrees. The first linear axis q1 corresponds to the x axis of the cartesian coordinate system, which origin is placed at the center of the triangual base. Other important design parameters that influence the computation of forward and inverse kinematics are:

- linear axis inclination angle (α)
- length of rods that connect the axis carriage and the end effector plate (L)
- x-plane projection of the distance between the origin and the axis rod joint (xref)
- y-plane projection of the distance between the origin and the axis rod joint (yref)
- end effector offset (EO) – distance between the end effector plate center and rod joint

It is assumed that the x_{ref} and y_{ref} distances are specified for axis position $q_1=0$ (maximum top position of each linear axis). It is also assumed that the first axis (q_1) is colinear with the x axis of the cartesian system. The design parameters are illustrated on Figure 2.

In order to obtain the forward and inverse kinematics transformations unit vectors a_1, a_2, a_3 have to be computed which point to the vertices of the triangual base plate:

$$a_1 = [-\cos(\alpha); 0; \ -\sin(\alpha)] \tag{1}$$

$$a_2 = [1/2 * \cos(\alpha); \ -1/2 * \sqrt{3} * \cos(\alpha); \ -\sin(\alpha)] \tag{2}$$

$$a_3 = [1/2 * \cos(\alpha); \ 1/2 * \sqrt{3} * \cos(\alpha); \ -\sin(\alpha)] \tag{3}$$

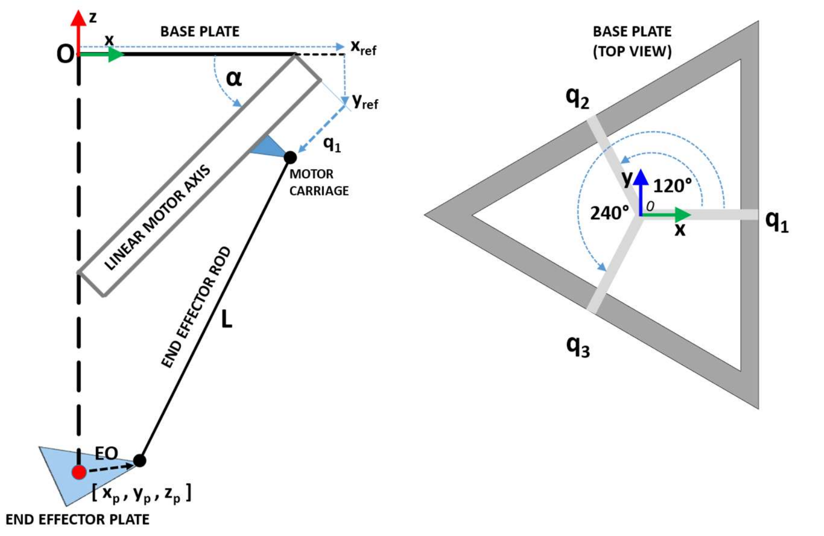

Figure 2. *Ilustration of parameters relevant to forward and inverse kinematic transformation for axis q1*

Then the d_1 vector is computed which is the vector between the q_1 linear axis rod joint and the centerline minus the end effector offset vector (EO). Vectors d_2 and d_3 are created by applying a rotation transformation around the origin to vector d_1. The vectors are rotated by 120 and 240 degrees respectively which corresponds to linear axes q_2 and q_3.

$$d_1 = [-(xref - EO);\ 0\ ;\ yref\ ] \tag{4}$$

$$d_2 = [\cos\left(\frac{2\pi}{3}\right) * d_1(1) - \sin\left(\frac{2\pi}{3}\right) * d_1(2)\ ;\ \cos\left(\frac{2\pi}{3}\right) * d_1(1) + \sin\left(\frac{2\pi}{3}\right) * d_1(2)\ ; yref\ ] \tag{5}$$

$$d_3 = [\cos\left(\frac{4\pi}{3}\right) * d_1(1) - \sin\left(\frac{4\pi}{3}\right) * d_1(2)\ ;\ \cos\left(\frac{4\pi}{3}\right) * d_1(1) + \sin\left(\frac{4\pi}{3}\right) * d_1(2)\ ; yref\ ] \tag{6}$$

Inverse kinematics is obtained by the following formulas:

$$E_i = \mathrm{a_i}(1)\ * \left(\mathrm{x} + \mathrm{d_i}(1)\right) + \mathrm{a_i}(2) * \left(\mathrm{y_p} + \mathrm{d_i}(2)\right) + \mathrm{a_i}(3) * \left(\mathrm{z_p} + \mathrm{d_i}(3)\right) \tag{7}$$

$$F_i = \left(x_p + d_i(1)\right)^2 + \left(y_p + d_i(2)\ \right)^2 + \left(z_p + d_i(3)\ \right)^2 + L^2 \tag{8}$$

$$q_i = E_i - \sqrt{E_i^2 - F_i} \tag{9}$$

where: i – index of each linear axis (1,2,3), q_i – position of the linear axis i, x_p,y_p,z_p – cartesian position of end effector, E_i, F_i – auxiliary variables for each linear axis used to simplify the formulas.
In order to obtain forward kinematics some auxiliary variables are first introduced which use d vectors given by equations 4-6 and positions of the linear axes q_i:

$$A_i = 2 * (\mathrm{d_i}(1) + q_i * \cos(\alpha) * \cos\left((i-1) * \frac{2\pi}{3}\right) \tag{10}$$

$$B_i = 2 * (\mathrm{d_i}(2) + q_i * \cos(\alpha) * \sin\left((i-1) * \frac{2\pi}{3}\right) \tag{11}$$

$$C_i = 2 * (\mathrm{d_i}(3) + q_i * \sin(\alpha) \tag{12}$$

$$\begin{aligned} D_i = \mathrm{d_i}(1)^2 + \mathrm{d_i}(2)^2 + \mathrm{d_i}(3)^2 + q_i^2 - \mathrm{L}^2 + 2 * q_i * (d_i(1) * \cos(\alpha) * \cos\left((i-1) * \frac{2\pi}{3}\right) + \cdots \\ + d_i(2) * \cos(\alpha) * sin\left((1-i) * \frac{2\pi}{3}\right)\ + d_i(3) * \sin(\alpha)) \end{aligned} \tag{13}$$

Further auxiliary variables are introduced to simplify the formulas:

$$B_4 = -\frac{B_1 - B_2}{A_1 - A_2}\ ;\ C_4 = -\frac{C_1 - C_2}{A_1 - A_2}\ ;\ D_4 = -\frac{D_1 - D_2}{A_1 - A_2} \tag{14}$$

$$C_5 = -\frac{C_2 - C_3 + (A_2 - A_3) * C_4}{B_2 - B_3 + (A_2 - A_3) * B_4} \tag{15}$$

$$D_5 = -\frac{D_2 - D_3 + (A_2 - A_3) * D_4}{B_2 - B_3 + (A_2 - A_3) * B_4} \tag{16}$$

$$C_6 = C_4 + B_4 * C_5 \tag{17}$$

$$D_6 = D_6 + B_6 * D_5 \tag{18}$$

$$K_1 = C_6^2 + C_5^2 + 1 \tag{19}$$

$$K_2 = 2 * C_6 * D_6 + 2 * C_5 * D_5 + A_3 * C_6 + B_3 * C_5 + C_3 \tag{20}$$

$$K_3 = D_6^2 + D_5^2 + A_3 * D_6 + B_3 * D_5 + D_3 \tag{21}$$

$$K_1 z_p^2 + \ K_2 z_p + K_3 = 0 \tag{22}$$

The solutions to the forward kinematics problem are given by a solution to quadratic equation (22). One of two solutions is chosen which corresponds to the case where the end effector lies below the linear axes:

$$z_p = \frac{-K_2 - \sqrt{K_2^2 - 4 * K_1 * K_3}}{2 * K_1} \tag{23}$$

$$y_p = C_5 * z_p + D_5 \tag{24}$$

$$x_p = C_6 * z_p + D_6 \tag{25}$$

4. EXTENSION OF THE CONTROL SOFTWARE AND PLANNED RESEARCH

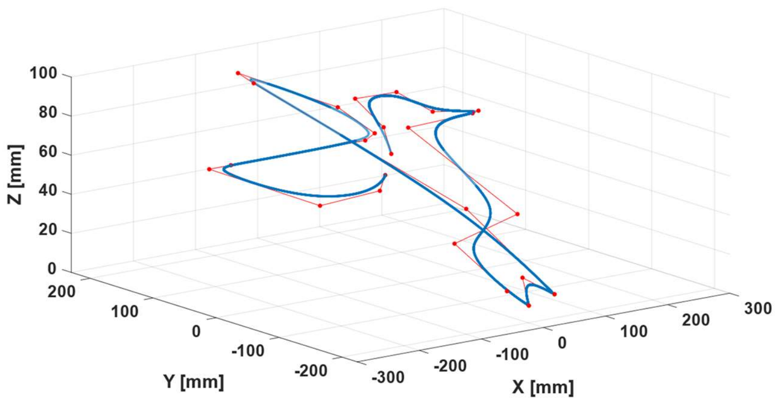

Figure 3. *Example of a 3d NURBS Curve (blue) with control points (red)*

The KEOPS-Delta parallel kinematics machine motion can be programmed using standard G-Code by utilizing the standard LinuxCNC block AXIS. This block performs linear and circular interpolation with look-ahead function and trapezoidal feedrate profiling. Due to high dynamics of a linear motor based parallel kinematics machine more advanced profiling and interpolation methods are required. For several years Non-Uniform Rational B-Spline (NURBS) polynomial curves are used in research for defining toolpaths [8, 9]. NURBS can define complex shapes with much less data points than g-code. The toolpath defined by NURBS is also smooth which is important when travelling along the path with high speeds. The curve shape is defined by control points which influence the curve locally. Any point on the curve is defined by a unique value of a unitless curve parameter (u). Position on the curve as well as curve derivatives for a given value of u are computed using de Boor's algorithm [10]. An example NURBS toolpath is shown in Figure 3.

The NURBS curve toolpath interpolation is more complex than linear and circular interpolation due to non-linear and unique relationship between the curve's arc length and parameter value. At each interpolation step a curve parameter increment is computed which corresponds to the desired arc-lenght increment which in turn depends on the feedrate value. The feedrate value (velocity tangent to the toolpath) is defined by a feedrate profile which is a function of the curve parameter or time. This non-linear interpolation problem can be solved using Taylor series or Predictor-Corrector methods [11, 12].

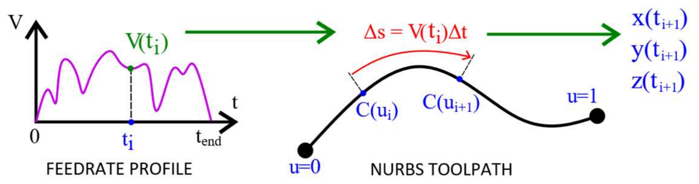

Figure 4. *Illustration of NURBS interpolation.*
$C(u_i)$, $C(u_{i+1})$ – curve positions at current and next interpolation points, $V(t_i)$ – current feedrate value, $x,y,z(t_{i+1})$ – next cartesian axis position values, t_i – current time step, u_i, u_{i+1} – current and next curve parameter values, Δs – desired arc-length increment, Δt – interpolation time step

Trapezoidal profiling implemented in LinuxCNC is not optimal for highly dynamic motion as the lack of jerk limitation can cause unwanted vibrations in the relatively light effector of the KEOPS-Delta machine. The authors developed a custom NURBS toolpath feedrate planning and interpolation module. A planar or spatial curve is interpolated based on a jerk-limited (s-curve) feedrate profile which is modified to take into account the curvature of the curve. The feedrate profile is generated by scanning the NURBS toolpath and computing a feedrate limiting function (FLF). This function returns maximum allowable feedrate which is the minimum value of maximum feedrate, feedrate limit based on a linearization of cartesian axis acceleration and jerk constraints and feedrate limit due to limitation of chord error. The chord error is the error caused by interpolation of a curved toopath by discrete points. The FLF is computed with the following equations [13]:

$$F_{clim} = \frac{2}{T}\sqrt{\rho^2 - (\rho - \varepsilon)^2}\ ;\ \ F_{alim} = \sqrt{\frac{A_{max}}{\kappa}}\ ;\ \ F_{jlim} = \sqrt[3]{\frac{J_{max}}{\kappa^2}} \tag{26}$$

$$\kappa = \frac{1}{\rho} = \frac{\|C' \times C''\|}{\|C'\|^3} \tag{27}$$

$$FLF = \min\left(F_{max}, F_{clim}, F_{alim}, F_{jlim}\right) \tag{28}$$

where: ρ – curvature radius, κ – curvature, ε – maximum chord error, A_{max}, J_{max} – maximum acceleration and jerk, T – interpolation time, C', C'' – first and second derivative of the NURBS toolpath

This function is just an approximation of actual axis limits but allows to indicate critical points for which the feedrate achieves local minima. The S-Curve feedrate profile is then planned between these minima where the start and end feedrates are obtained from the FLF. Based on initial, maximum and end feedrates, maximum acceleration and jerk the distance covered by the profile is checked against the arc-length between feedrate critical points. If the distance is too large the maximum feedrate value is decreased by way of one-dimensional bisection search until the profile fits the distance. In order to handle actual axial limits the The s-curve generation algorithm adds another step that checks if the axial limits are violated. If so the profile acceleration or jerk are iteratively decreased until there is no axial violation. This method is described in detail in the authors' paper [14]. This method works well for cartesian machines however for machines with non-linear kinematics such as KEOPS-Delta the axis limitations cannot be guaranteed.

The main aim of the research, which will be performed on the presented station, is the development of computationally efficient feedrate optimization algorithms. These will minimize trajectory execution time taking account machine limits. These limits should include limitations of the machine axes such as maximum velocity, maximum drive current. Such a task is non-trivial and has been investigated for cartesian machines. It is even more complex for parallel kinematics machines due to inlcusion of non-linear forward and inverse kinematics transformation.

The s-curve feedrate planning method developed by the authors for limiting axial acceleration and jerk for planar high speed machines will be used as an initial guess. The generated s-curve feedrate profile will be converted to a B-Spline polynomial curve profile. The Particle Swarm Optimization algorithm will be used to adjust the initial feedrate profile. The authors developed a similar method for planar cartesian machines [15] which will be extended to handle non-linear kinematics of the Delta-KEOPS. The algorithms will be implemented in the machine as a user-space program, which will generate optimized velocity profile for the NURBS interpolator. The aim is to maximize the dynamics of the Delta-KEOPS machine while maintaining the linear axes within their limits of acceleration, jerk and error.

5. PRELIMINARY EXPERIMENTAL RESULTS

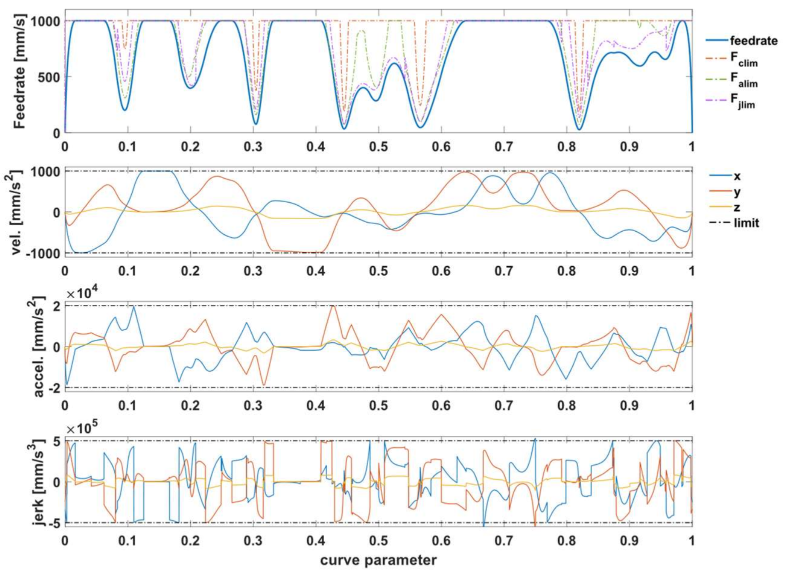

Figure 5. *Feedrate profile and cartesian axis velocity, acceleration and jerk generated for »bird« NURBS curve. Note that the minima of the feedrate profile correspond to the sharp corners on the curve.*

As described in the previous section, an s-curve feedrate profile was generated for the »bird« NURBS toolpath presented on Figure 3. The profile was generated with the velocity, acceleration and jerk limits set to 1000 mm/s, 10000 mm/s2 and $5*10^5$ mm/s^3 respectively. The chord error limit was set to 0.001 mm. The generated feedrate profile as well as axis velocity, acceleration and jerk profiles are presented on Figure 5. In can be seen that the feedrate generation algorithm generates the s-curve profiles so that it lies beneath the FLF. The algorithm limits the cartesian axis velocity, acceleration and jerk to the prescribed limits.

The feedrate profile shown above was used to interpolate the »bird« toolpath and generate position commands for the KEOPS-Delta machine. The interpolated positions were converted to linear axes position commands and run on the machine. Values of position, velocity, acceleration and jerk for each linear axis are shown in Figure 6. It can be seen that the individual values significantly violate the axial limits which are imposed on the cartesian axes. However the achieved values are still within the capabilities of the linear motors. This shows that the non-linear kinematics significantly influences the generated profiles. Utilization of optimization algorithms to generate a feedrate profile that does not violate the linear axis limits is necessary which will be the goal of further research.

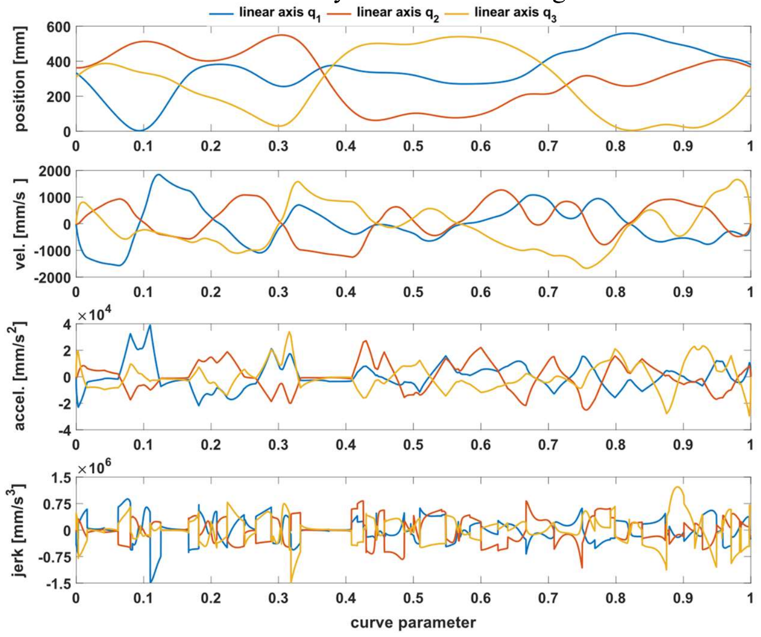

Figure 6. *Position, velocity, acceleration and jerk profiles for linear axes (after inverse kinematics*

6. CONCLUSION

The paper presents a control system for a parallel kinematics machine in the KEOPS-Delta configuration. Instead of commonly used linear belt drives or ballscrews the machine is driven by linear motors. The machine's PC-based control system utilizing the LinuxCNC control software, was described. The forward and inverse kinematic transformations, necessary for control of non-cartesian machines, was also presented. Further research goals which include development of constrained feedrate optimization algorithms are presented along with preliminary results. The presented research station with KEOPS-Delta parallel kinematics machine can achieve highly dynamic motion useful for high speed manufacturing.

7. LITERATURE

[1] Pierrot, F., Reynaud C., Fournier, A.: *DELTA: a simple and efficient parallel robot.* Robotica 8.2, pp. 105-109, 1990

[2] Bouri, M., Clavel, R.: *The linear delta: Developments and applications,* ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics),pp. 1–8, Munich, Germany, 7-9.06.2010

[3] Zeng, Q., Ehmann, K. F., Cao, J.: *Tri-pyramid Robot: Design and kinematic analysis of a 3-DOF translational parallel manipulator*. Robotics and Computer-Integrated Manufacturing, Vol. 30, No. 6, pp. 648-657, 2014.

[4] Jansen, D., Buttner, H.,: *Real-time Ethernet: the EtherCAT solution,* Computing and Control Engineering, Vol. 15, No. 1, pp. 16-21, 2004.

[5] Jo, Y. H., Choi, B. W.: *Performance Evaluation of Real-time Linux for an Industrial Real-time Platform,* International journal of advanced smart convergence, Vol. 11, No. 1, pp. 28-35, 2022.

[6] Paprocki, M., Erwinski, K.: *Synchronization of Electrical Drives via EtherCAT Fieldbus Communication Modules*, Energies, Vol. 15 No. 2, 604, 2022.

[7] IGH EtherCAT master for Linux, https://etherlab.org/en/ethercat/index.php, 09.2022

[8] Zhao H., Zhu L., Ding H.: *A real-time look-ahead interpolation methodology with curvature-continuous B-spline transition scheme for cnc machining of short line segments*, International Journal of Machine Tools and Manufacturing, Vol. 65, pp. 88–98, 2013

[9] Mercy T., Jacquod N., Herzog R., Pipeleers G., *Spline-based trajectory generation for cnc machines*, IEEE Transactions on Industrial Electronics, Vol. 66, No. 8, pp. 6098–6107, 2018

[10] Piegl L., Tiller W.: *The NURBS book,* Springer Science & Business Media, Berlin, Heidelberg, 1997

[11] Erwinski, K., Paprocki, M., Karasek, G.: *Comparison of NURBS trajectory interpolation algorithms for high-speed motion control systems,* IEEE 19th International Power Electronics and Motion Control Conference (PEMC), pp. 527-533. Gliwice, Poland, IEEE, 25-29.04.2021

[12] Jia Z.-Y., Song D.-N., Ma J.-W., Hu G.-Q., Su W.-W., *A nurbs interpolator with constant speed at feedrate-sensitive regions under drive and contour-error constraints*, International Journal of Machine Tools and Manufacturing, Vol. 116, pp. 1–17, 2017

[13] Sun Y., Chen M., Jia J., Lee Y.-S., Guo D., *Jerk-limited feedrate scheduling and optimization for five-axis machining using new piecewise linear programming approach*, Science China Technological Sciences, Vol. 62, No. 7, pp. 1067–1081, 2019

[14] Erwinski, K., Wawrzak, A., Paprocki, M.: *Real-time jerk limited feedrate profiling and interpolation for linear motor multi-axis machines using NURBS toolpaths*, IEEE Transactions on Industrial Informatics (Early Access, 1.02.2022), doi: 10.1109/TII.2022.3147806.

[15] Erwinski, K., Szczepanski, R., Tarczewski, T.: *Nature inspired optimization of jerk limited feedrate profile for NURBS toolpaths in CNC machines*, IOP Conference Series: Materials Science and Engineering, Vol. 1140, Modern Materials and Manufacturing (MMM 2021), 27-29.04.2021, Tallinn, Estonia.

Erwinski, K., Karasek, G., Živanović, S., Dimic, Z., Slavkovic, N.

UPRAVLJANJE U REALNOM VREMENU KEOPS-DELTA MAŠINOM SA PARALELNOM KINEMATIKOM BAZIRANO NA LINUXCNC I ETHERCAT-u

U ovom radu je predstavljena laboratorijska postavka za razvoj algoritama optimizacije trajektorije numerički upravljanih mašina alatki sa spegnutim osama. Laboratorijsku postavku čine DELTA mehanizam u KEOPS konfiguraciji sa linearnim osnaženim osama pogonjenih servo pogonima visokih performansi. Upravljanje mašinom je bazirano na LinuxCNC softverskom sistemu. Komunikacija pri upravljanju se vrši u realnom vremenu preko EtherCAT-a. U radu je takođe opisano proširenje LinuxCNC upravljačkog sistema sa NURBS interpolacijom i profilisanjem brzine pomoćnog kretanja pomoću s-krive. Dalja istraživanja će se odnositi na razvoj algoritama optimizacije trajektorije za mašine alatke sa paralelnom kinematikom.

Ključne reči: *paralelna kinematika, Linux, LinuxCNC, Delta, KEOPS*