Path Simulator for Machine Tools and Robots

Květoslav Belda
Department of Adaptive Systems
Institute of Information Theory and Automation of the ASCR
Pod Vodárenskou věží 4, 182 08 Prague 8, Czech Republic
E-mail: belda@utia.cas.cz

Pavel Novotný
Department of Physical Electronics
Faculty of Nuclear Science and Physical Engineering, CTU Prague
Trojanova 13, 120 00 Prague 2, Czech Republic
E-mail: p-novotny@atlas.cz

***Abstract*—The paper deals with a set of the mathematical algorithms for a time parameterization of the motion paths of the machine tools. The key descriptors of the paths follow from CAD documentation, technical drawings or from technological demands. The presented algorithms provide a time mapping of individual path coordinates. The algorithms are based on the kinematical relations and on analytical geometry in space. They work with position, velocity, acceleration and jerk quantities. All algorithms are implemented in a path simulator, which was developed as a user friendly utility windows application. The described mathematical principles are demonstrated by several testing motion paths.**

I. Introduction

A conversion of input requirements to a suitable range and format for real control system of machine tool is important procedure at integration or change of product properties into production process. The requirements can be given manually by user or by data from CAD documentation, technical drawings or from technological demands. Relative to control system used in machine tool or robot, the conversion represents a specific task of time parameterization of the motion tool or robot gripper path (Fig. 1) [7]. It is subjected to given machine parameters, surrounding path obstacles and other factors influencing the shape or geometrical profile of the motion paths. The main purpose of the parameterization is to generate the reference inputs i.e. desired, required values with appropriate timing [2]. The timing has to correspond to used control system and to user requirements, demands. In mechatronic field, those demands are usually given by a set of positions with kinematic parameters as desired velocities or constraints specifying a maximum permissible acceleration and jerk. The demands can follow from • *technology of production procedures*: machining velocities, motion orientation; but even from • *system construction*: minimum radiuses, shapes, kinematic and dynamic limits.

From control point of view, the time parameterization itself represents generating a time sequence of the reference values according to some deterministic way defined in advance, where this time reference sequence interpolates the initial parameters from user demands.

This work was partially supported by grant RVO68407700.

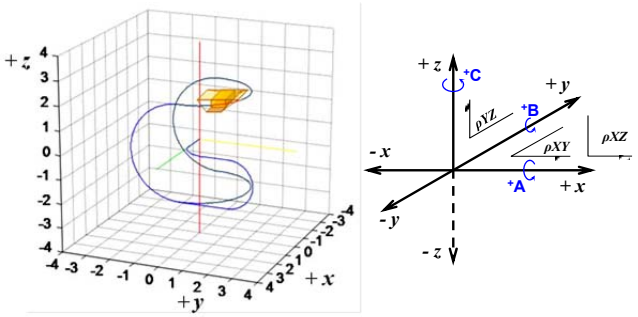

Figure 1. Example of spatial path in 3D Cartesian space

In majority cases, the machine tools and robots are highly dynamic systems [1], [3], therefore the right forming of feasible reference inputs is important. Due to high dynamics, ordered set or pairs of time and reference values (time table) is considered.

The obtained sequence of time reference i.e. ordered pairs of time marks and coordinates as a result of the parameterization are intended for control system either as desired values pre-computed offline or continually computed online similarly by the same algorithms adapted for online processing. The parameterizing algorithms are based on motion kinematics and analytical geometry in space. They work with position, velocity, acceleration and jerk quantities. All presented algorithms are implemented in a path simulator, which is being developed as utility windows application. The described mathematical principles are demonstrated by several examples, which show time dependencies of parameterized tested geometrical paths.

The paper is organized as follows. Section II, the principal section, deals with an explanation of partial mathematical algorithms. The section goes from pre-processing, through core time parameterization up to computation of individual kinematical quantities as individual axis components in 3D Cartesian space. The section III briefly outlines implementation issues of proposed algorithms in the path simulator. The sections IV and V demonstrate the algorithms on the outputs of the path simulator as a set of examples for several selected testing geometrical path segments.

II. Mathematical Algorithms

The section focuses on mathematical relations of the algorithms for computation of all necessary quantities of time parameterization. The section is divided into three following parts: pre-processing, time parameterization and component computation of the time parameterized kinematical quantities in 3D Cartesian space.

Let us assume a raw set of coordinates of key points, which sufficiently describe the shape of the path as an initial data set for the parameterization. The key points represent specific boundary points, which split the path into specific shorter parts. Let us call these parts as path segments. The segments are considered as the simplest primitive geometrical shapes like abscissa, arc or spline curves. The ordered segment connection forms whole geometrical profile of the parameterized path. This description is sufficient for beginning of the pre-processing stage.

A. Pre-processing

The pre-processing serves for calculation of initial path quantities, which are calculated in boundary points for each segment. The quantities are lengths of segments, transition angle between two neighbouring segments and terminal velocities. The determination of the lengths and transition angles is simply given by Pythagorean theorem and goniometric functions among tangential and normal vectors in individual boundary path points. The calculation of terminal velocities is more complicated. For that reason, it will be explained in detail.

The terminal velocity computation begins by a determination of the segments, which terminate with zero speed. These segments have too sharp transition angles or they are followed by pause/dwell segments. Furthermore, if the segment is the last segment of the path, then it has to terminate with zero speed too [8].

The algorithm can be expressed by the following recurrent equation:

$$v_{f\,max(i)} = v_{f\,max(i+1)} + \sqrt{2\,a_{max}\,l_{(i+1)}}, \quad i = N-1, \cdots, 2, 1 \qquad (1)$$

where i is a number of actual segment, $v_{f\,max(i)}$ is a maximum terminal velocity, $l_{(i)}$ is a segment length and a_{max} is a maximum acceleration – a parameter of the real machine.

B. Time parameterization

The real procedure of time parameterization is based on the computation of time dependent parameter and its appropriate time derivatives for every segment. Altogether, computed values represent the kinematical quantities for tangential direction of parameterized path spread in 1D line (one-dimensional line). The time dependent parameter represents a time dependent distance $s(t)$, and its appropriate derivatives mean velocity $v(t)$, acceleration $a(t)$ and jerk $j(t)$. The geometrical shape of the time parameterized path i.e. generated trajectory is not important in this stage, but the position on the path. The shape is considered in used conditions e.g. in condition of maximum terminal velocity.

The parameterization is based on a definition of the order and the structure of the kinematical model of the acceleration. In this paper, model is considered in the form of the first order polynomial (3). It gives the fastest start-up to the fast manipulation or working velocity. It is possible to choose higher orders e.g. 3rd or 5th [5], [6], which give smoother profiles [12] of all kinematical quantities. They are a little bit slower, but the generated values, due to their smoothness, lead to smaller wear of motors and robot joints, i.e. lead to more flowing motion. The procedure for higher orders is similar.

$$j(t) = k, \quad k \in \{-jerk, 0, +jerk\} \qquad (2)$$

$$a(t) = \int j(t)\,dt = k\,t + c_1 \qquad (3)$$

$$v(t) = \int a(t)\,dt = \frac{1}{2}kt^2 + c_1 t + c_2 \qquad (4)$$

$$s(t) = \int v(t)\,dt = \frac{1}{6}kt^3 + \frac{1}{2}c_1 t^2 + c_2 t + c_3 \qquad (5)$$

where constants c_1, c_2, c_3 are determined according to time position within the topical parameterized segment, see Fig. 2.

For the 1st order kinematical model (3), from acceleration point of view, there are two cases of behaviour of acceleration: triangular and trapezoidal. The both cases are described separately for different way of the time computation [8].

The triangular behaviour appears when the half of maximum velocity is reached earlier than the maximum acceleration. On the contrary, the trapezoidal behaviour is characterised by the reaching the maximum of acceleration earlier than half of the maximum velocity. It implies constant acceleration on its maximum magnitude for certain time.

The condition for triangular behaviour can be written as:

$$\frac{a_{max}}{k} \geq \sqrt{\frac{v_{max}}{k}} \qquad (6)$$

The condition follows from a simple integration of the jerk or acceleration polynomial as indicated in (2) - (4). If the condition (6) is not fulfilled, the acceleration is switched to the trapezoidal case. Equation (6) shows that the behaviour type depends only on parameters of used machine. There are some special cases, when the trapezoidal behaviour can change to the triangular behaviour, e.g. for too short segments or in segment with strict maximum value of the velocity.

The graphs of time dependences of jerk, acceleration, velocity and length (graphical illustration of (2) - (5)) are shown in the Fig. 2. In it, there is a simple general case: the machine first accelerates from initial velocity, then it runs steadily for some time and it decelerates finally back to the initial value. The triangular behaviour is on the left side and the trapezoidal behaviour on right side. In the both cases, the maximum velocity is reached. Moreover, in the case of trapezoidal behaviour, the maximum acceleration is reached too.

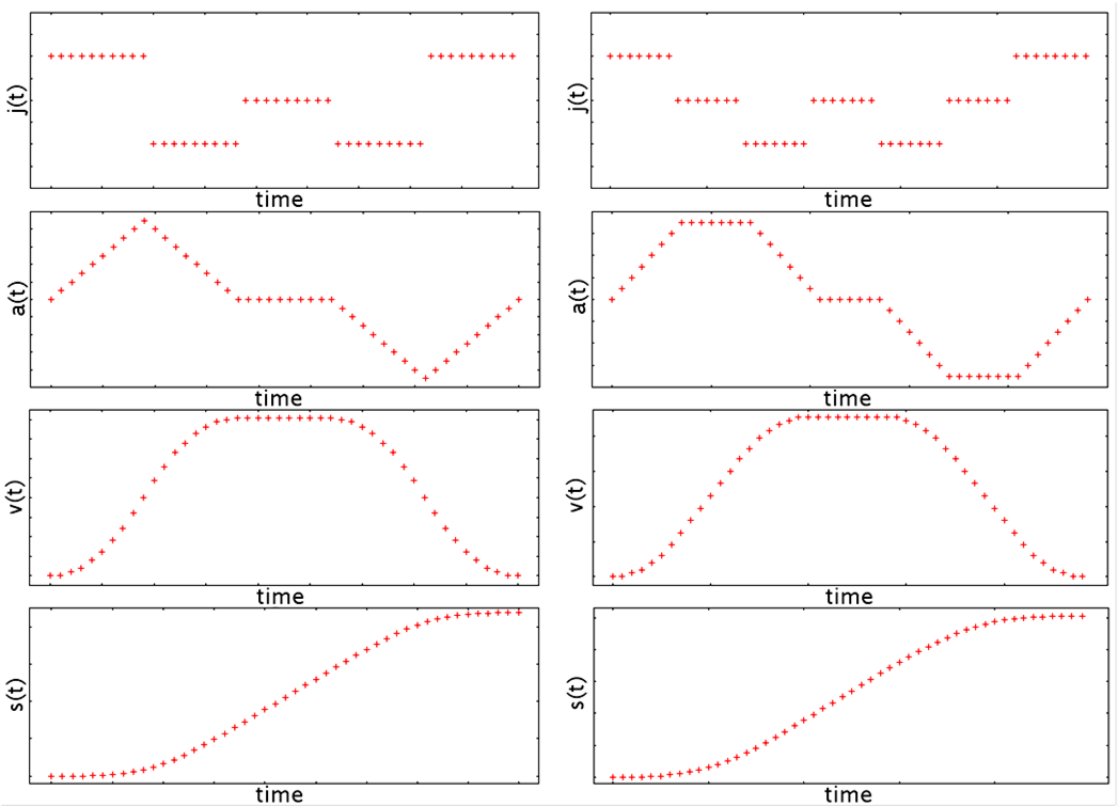

Figure 2. Triangular (left) and trapezoidal (right) acceleration profile $a(t)$ and other kinematical quantities within one parameterized segment

C. *Computation of 3D Cartesian kinematical quantities*

The aim of the computation is to determine the individual components for individual axes in 3D Cartesian space as coordinates and other kinematical quantities. The all quantities are determined for working point in the operational space of the machine tool or robot. The quantities here mean component values of position, velocity, acceleration and jerk related to the time. The computation follows from time dependent parameter and its appropriate derivatives. Thus, the time parameterization by time dependent parameter and computation of appropriate kinematical component quantities are closely related.

The component computation differs for different shapes of path segments: for abscissa segments, for arc segments or further e.g. for spline segments. In this computation, the shape of the segment is important on the contrary to factual time parameterization in previous subsection.

The abscissa segment (Fig. 3) can be described as follows:

$$x(t) = x(0) + p(t)\frac{u_x}{\|\mathbf{u}\|} \tag{7}$$

$$y(t) = y(0) + p(t)\frac{u_y}{\|\mathbf{u}\|} \tag{8}$$

$$z(t) = z(0) + p(t)\frac{u_z}{\|\mathbf{u}\|} \tag{9}$$

where $p(t) \in \langle 0, \ell \rangle$ is the time dependent parameter as a function of time t, $\mathbf{u} = [u_x, u_y, u_z]$ is a directional vector of the abscissa, $x(0)$, $y(0)$, $z(0)$ are individual coordinates of the initial point of the abscissa.

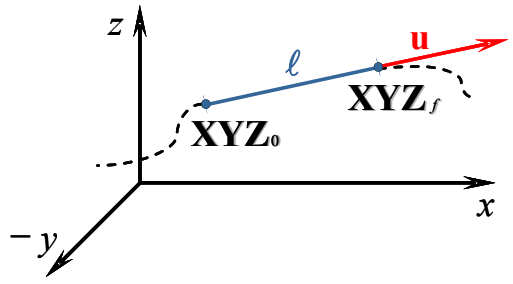

Figure 3. Abscissa segment

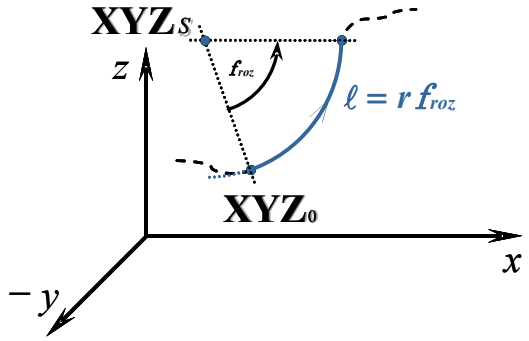

Figure 4. Arc segment

The arc segment (Fig. 4) can be considered either in some of the three main planes or in parallel planes of 3D Cartesian system only or in some general position. The former assumption allows us to use simpler description of the arc in the space, e.g. for the arc in parallel plane to horizontal ρ*XY* plane:

$$x(t) = x_C + r\cos(\varphi(t) + \delta) \tag{10}$$

$$y(t) = y_C + r\sin(\varphi(t) + \delta) \tag{11}$$

$$z(t) = z(0) \tag{12}$$

where r is arc radius, δ is angle between initial radius vector and normal vector of ρ*XY* plane [1; 0; 0] and $\varphi(t)$ is time-dependent parameter defined as follows:

$$\varphi(t) = \frac{p(t)}{r} \tag{13}$$

where $p(t) \in \langle 0, \ell \rangle$ represents lengths of arc elements and ℓ is length of whole arc. As it is obvious, the z-coordinate is constant. If the arc segment is located in other plates ρ*XZ* or ρ*YZ*, the constant coordinate will be y or x respectively.

For completeness, i.e. for some general position of arc segment in 3D space, the individual coordinates of arc segment are defined as follows:

$$\mathbf{XYZ}(t) = \mathbf{XYZ}_C + \mathbf{h}_1\cos(\varphi(t)) + \mathbf{h}_2\sin(\varphi(t)) \tag{14}$$

where $\mathbf{h}_1$ and $\mathbf{h}_2$ are appropriate vectors of the primary and secondary arc radiuses and **XYZ** is a vector [$x(t)$, $y(t)$, z(t)].

In similar way, it is possible to use time dependent parameter for other parametric curves (elliptic, parabolic or hyperbolic arcs or spline curves). The formulas of parametric curves represent similarly expressions for computation for coordinates. From practical use point of view, they can be simply replaced (approximated) by composition via sets of very small elemental abscissa segments, if appropriate curve format is not directly supported by a control system or it is not required by user.

The relations for velocities, accelerations and jerks are calculated by derivatives the equations (7) - (9) and (10) - (13) or (14) for appropriate segments respectively.

III. Implementation of Algorithms in the Path Simulator

The described mathematical relations of algorithms in previous section are implemented in user friendly path simulator application. The application was programed as platform independent application in C++ using standard templates [9]. To create the graphical user interface, the Qt libraries [10] were used together with OpenGL libraries [11]. They provide functionalities for 3D space trajectory visualisation.

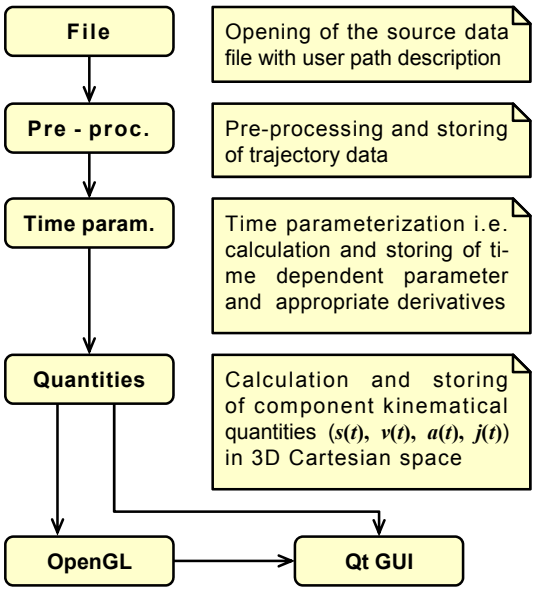

Figure 5. Diagram of logical blocks within the path simulator

The path simulator consists of many classes as collaborating logical blocks. Their diagram is shown in Fig. 5. The classes for storing the results are not shown. The class File reads the trajectory description and its result is continuously processed by class Gcode (Pre-processing). When all path segments are processed, the time parameterization is performed. Then, the individual kinematical quantities are calculated. The front-end interface manages interactions with user, application logic and displays of numerical results.

A. Expected user input parameters

The path simulator processes and analyses the trajectory obtained from user defined path. As the initial description of the trajectory, the G code is used, but it can be considered another unified format containing key descriptors of the path considered for time parameterizing in the simulator. The path processing algorithms use both path geometrical descriptors and machine parameters. The machine parameters are generally independent of the path geometry. Therefore, they are set up individually in simulator via appropriate application menu.

The generated trajectory consists of abscissa and arc segments generally. The abscissa segments are described by final coordinates and maximum velocity. The initial coordinates can be derived from terminal coordinates of previous segment. The arc segments are described by final coordinates, maximum velocity and by coordinates of the arc center. These parameters were used due to compatibility with industrial control systems.

The parameters without direct connection with the input path are the following: initial coordinates of the machine tool, maximum jerk, maximum acceleration, maximum manipulation velocity and maximum working velocity. These parameters can be changed independently by the graphical user interface within the simulator.

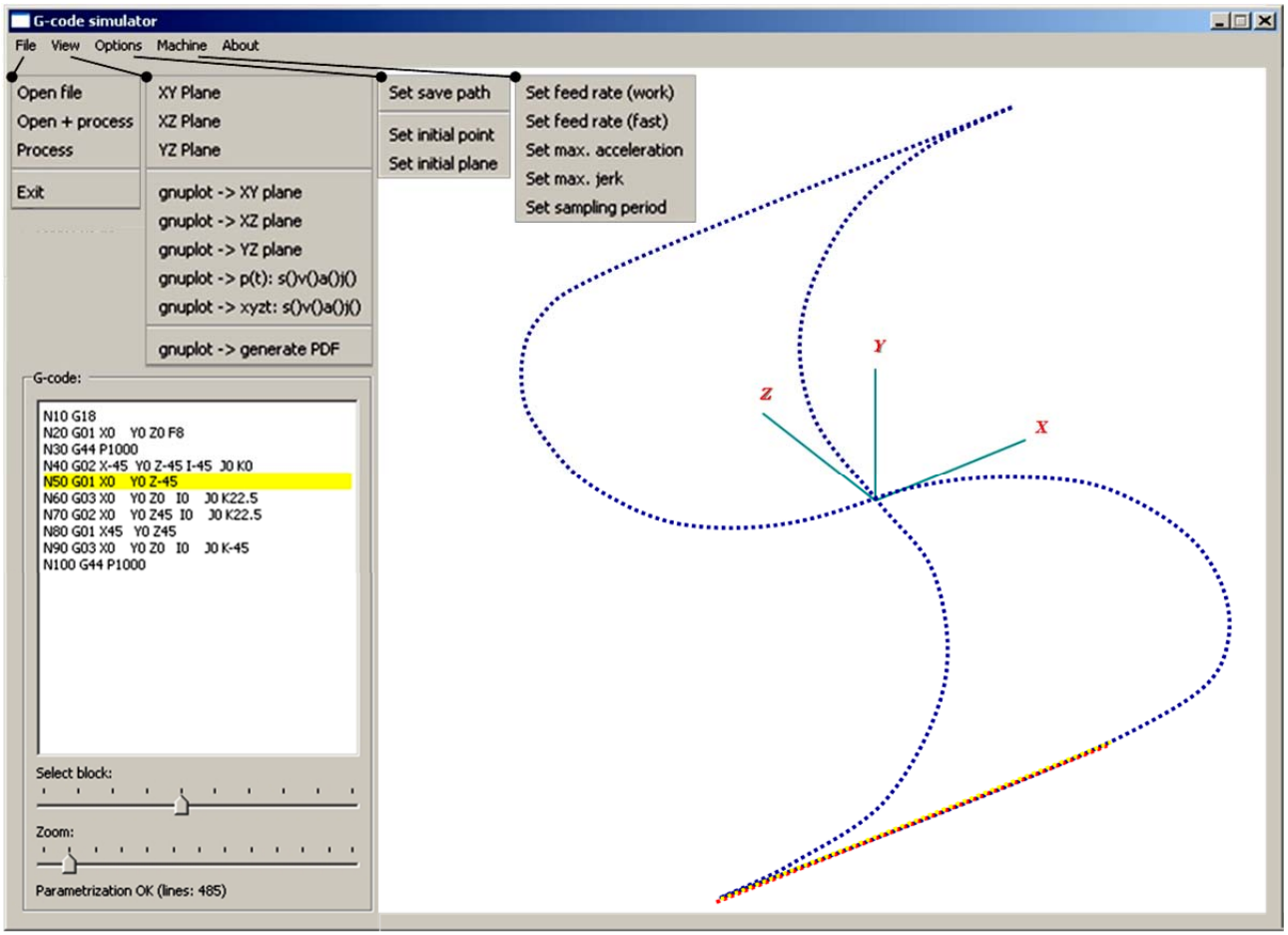

Figure 6. Screenshot of the path simulation application

IV. Graphical User Interface

The program is equipped by a standard window graphical user interface to set up the trajectory and the machine tool parameters and show and save the results. To develop GUI, standard Qt [10] and OpenGL [11] libraries were used.

The main window of the application is shown in the Fig. 6. Shape of the trajectory is drawn in OpenGL area on the right. The *Zoom* slider allows user to zoom in or out the OpenGL area. Rotation of the view can be realized by a dragging of the mouse cursor. The path description is written on the left in text-box. The *Select block* slider on the left down allows user to highlight any instruction line of the path description simultaneously with appropriate shape segment.

All other functions are accessible from the menu bar. For better illustration, all sub-menus are expanded in the figure. The *File sub-menu* controls the opening of files containing path description, the parameterization process execution and proper exit of the application. The *View sub-menu* contains options for visualization of the results. The *sub-menu Options* allows user to set up the application variables like file path for storing the results and initial parameters. The machine parameters can be set up by options in the *Machine sub-menu*. The last option *About* displays the window with the information about author and application version.

V. Application Outputs and Examples

The path simulator offers several outputs to user. The first and most obvious result is visualization output by the graphical window of the application. It was designed to provide scalable and rotatable view on shape the trajectory with ability of highlighting each segment. The axes controls for the manipulating with the model are shown in Fig. 6. There are three pre-set basic views: *XY Plane*, *XZ Plane* and *YZ Plane* here. The further output consists in graphs (Fig. 7). The graphs serve for detail analysis of the time dependent parameter and the quantities in each main plane of 3D coordinate space (three main views). These graphs are drawn in separate window out of the main application graphical window.

Moreover, in the application, there is a possibility to export the output data into text files. These files are in wide compatible format, which can be imported into any standard application as MS Excel or MATLAB, or in general, directly into real control system of the machine tool or robot.

The following illustrative examples show time behaviors of the time dependent parameter and appropriate kinematical quantities (Fig. 7) for the parameterized path visualized in the OpenGL area (Fig. 6). This graphical output represents the main and the most widely used user output of the application.

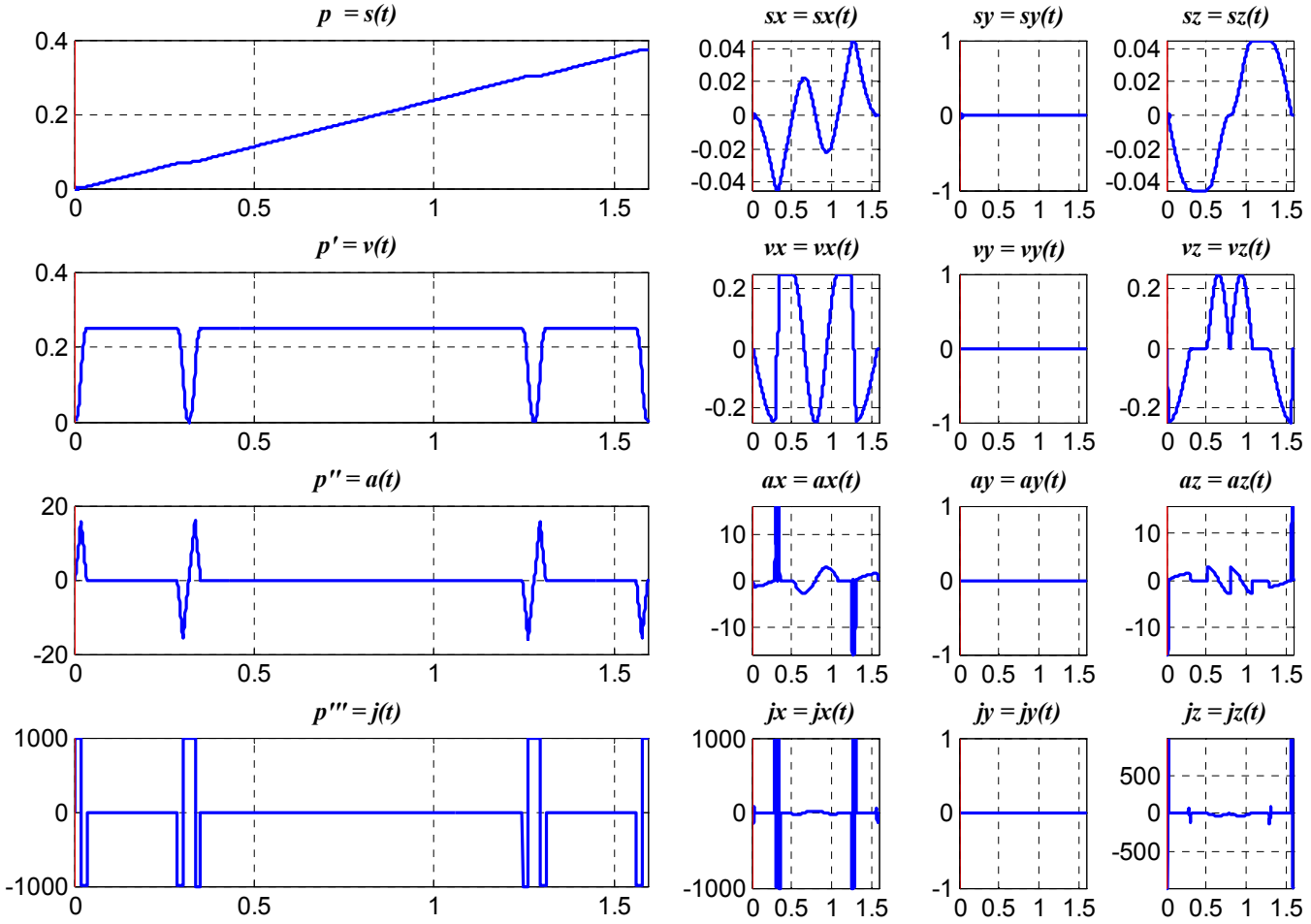

Figure 7. Time behaviours of time dependent parameter (4 subfigures on the left) and appropriate kinematical quantities (12 subfigures on the right) for the parameterized path shown in the OpenGL area in Fig. 6 (test machine parameters: $v_{max} = v_{work} = 0.25\text{ms}^{-1}$; $a_{max} = 20\text{ms}^{-2}$; jmax = 1000 ms^{-3})

The graphs in the Fig. 7 are closely related to the trajectory or parameterized path shape. It is visible, that representative path from Fig. 6 lies in the vertical *XZ plane* (all *y-components* are constantly zeros). In general, the user can analyse behaviour for individual axes in relation to the permitted or recommended technological conditions. In the inadmissible cases, the user can change the initial properties of the parameterization and optimize the desired behaviour of the machine tool or robot motion.

VI. Conclusion

The paper deals with a time parameterization of the motion paths for machine tools and robots. The result of time parameterization of the path is a time parameterized trajectory. The information involved with the trajectories is ordered pair of the time t and kinematical quantities ($s(t)$, $v(t)$, $a(t)$ and $j(t)$). The presented theoretical expressions are demonstrated by the graphical and numerical outputs of path simulator application.

References

[1] L.W. Tsai, Robot Analysis: The Mechanics of Serial and Parallel manipulators. John Willey Inc., New York, 1999.

[2] L. Sciavicco, and B. Siciliano, Modeling and Control of Robot Manipulators, The Mc-Graw Hill Companies, Inc., New York, 1996.

[3] M. Valášek, "Dynamic Time Parameterization of Manipulator Trajectories", Kybernetika, vol. 23, UTIA CSAS 1987, pp. 154-174.

[4] V. Stejskal and M. Valášek, Kinematics and Dynamics of Machinery, Marcel Dekker, Inc., New York, 1996.

[5] K. Belda, J. Böhm, and P. Píša, "Concepts of Model-Based Control and Trajectory Planning for Parallel Robots". Proc. of 13th IASTED Int. Conference on Robotics and Applications. (CD ROM). RA 2007, Würzburg, Germany. Acta Press, Zurich, 2007, pp. 15-20.

[6] K. Belda, "On Time Parameterizations of User Demands in Mechatronics". Proceedings of the 10th International PhD Workshop on Systems and Control. (CD ROM). UTIA ASCR, Prague, 2009, pp. 1-6.

[7] P. Keller, Programming & control of CNC machnes (in Czech). Liberec, TUL, 2010, <http://www.kvs.tul.cz/download/cnc_cadcam/pnc_2.pdf>.

[8] P. Novotný, Graphical G-code Simulator for Machine Robot Tools. Master Thesis, CTU in Prague, 2011.

[9] C++ Programming language documentation, [cited 8.3.2012], Available on: < http://www.cplusplus.com/doc/ >

[10] Qt Developer Network documentation, [cited 9.3.2012] Available on: < http://qt-project.org/doc/qt-4.8/modules.html >

[11] Technical documentation of OpenGL Graphic System, [cit. 12.3.2012]: < http://www.opengl.org/registry/doc/glspec40.core.20100311.pdf >

[12] A. Gasparetto, V. Zanotto, "Optimal trajectory planning for industrial robots", Advances in Engineering Software 41 (2010), pp. 548-556.