Path Planning Kinematics Simulation of CNC Machine Tools based on Parallel Manipulators

Luc Rolland

Abstract

Since the very successful application of parallel robots in material handling, many projects attempted to implement the Gough platforms as milling machine manipulators with limited success mainly achieving roughing.

The displacement of the milling tool should meet surface finish requirements. Users also wish to increase tool feedrate in order to improve productivity thereby reaching high speed milling levels. Even a constant high speed feedrate brings important challenges since they mean higher actuator accelerations even on straight lines. This work introduces geometric formalization of surface finish which is more realistic then classic error calculations.

This research work proposes an off-line simulation tool analysing the milling task feasibility using a robot constituted by a general hexapod parallel manipulator, namely the Gough Platform, often refered as the Stewart Platform. Moreover, in order to meet the machine-tool standards, the parallel robot will be controlled by a typical CNC controller implementing classic position based algorithms adapted to the parallel robots with any kind of actuator polynomial interpolation. Control sampling rates are studied and their impact evaluated.

High and very high speed milling simulation results show the implementation of linear and third order interpolation between the actuator set-points calculated from the CAD/CAM computed end-effector or tool set-points points. The results show that linear interpolation are not sufficient for high speed milling and then third order interpolation reach the required surface finish at fast and feasible CNC sampling rates.

Luc Rolland
High Performance Robotics Laboratory, Memorial University of Newfoundland, St-John's Campus, St-John's, NL, Canada, e-mail: lrolland@mun.ca

1 Introduction

After the confirmed success of parallel robots as flight simulators followed by their more recent breakthroughs in material handling, they are actually implemented as machine-tools. Several commercialization attempts were made over the years, Fig. 1. With the promise of increased productivity, we aime to achieve the two following goals:

1. To reach higher feedrates while keeping excellent surface finish quality
2. To obtain faster accelerations during path transfers between task trajectories.

The main advantages of these robotic manipulators compared to serial ones are simpler construction, more rigid structures, non-cumulative kinematics chain deflections, greater throughputs from higher accelerations and less energy consumption from smaller actuators. On the other hand, these manipulators feature drawbacks such limited workspace and complex non-linear kinematics.

In material handling applications the ratio between actuator displacement travel and accuracy is around 1000 mm over 1 mm, whereas in milling applications the ratio becomes 1000 mm over 0,001 mmm, meaning it 1000 times larger.

Due to the highly non linear nature of parallel robots, their implementation still pose serious challenges.

Initial path planning investigations for parallel robots were trying to determine if any task would include their entire paths inside the robot workspace, where the notion of trajectory quality has been formulated in terms of distances from actuator limits (Merlet 1993). Kinematics chain collision was added to the analysis, (Chedmail, Hascoet and Guerin 1994). Path planning involved singularity investigation to avoid instantaneous self-motion, (Nenchev and Uchiyama 1996). Singularities were extensively studied (Bhattacharya, Hatwal and Ghosh 1998), (Dasgupta and Mruthyunjaya 1998), (Dash et al. 2003). The problem evolved into multi-objective optimization finding the optimum path according to a certain number of criterias

Fig. 1 Two commercial milling machines : Variax of Giggings and Lewis and CMW-300 of CMW-Marioni

(Carbone et al. 1997), (Merlet 2001). (Chablat and Wenger 1998) introduced collision avoidance to singularity analysis to answer the question of moveability in the presence of obstacles. Planning time-minimal trajectories were introduced by (Abdellatif and Heimann 2005), (Huang T. et al. 2007). In (Khoukhi, Baron and Balazinski 2009), the authors minimize electrical energy, kinetic energy, robot motion time separating two sampling periods, and maximize a measure of manipulability allowing singularity avoidance.

More specifically, implementing the Gough platform as a milling machine, often refered as the Stewart platform, path planning was studied where the contour error was used as a performance criteria to determine the effect of PID controls applied on each actuator (Masory and Xiu 1998). The redundant sixth degree-of-freedom was utilized for optimization according to various criterias (Merlet, Perng and Daney 2000). Path planning schemes also targeted the axial force minimization (Shaw and Chen 2001), where maximum constant cutting force along the contour were maximized (Oen and Wang 2007). Then, added objectives included stiffness maximization (Pugazhenthi, Nagarajan and Singaperumal 2002).

This research work addresses the feasibility of a succesfull machining task in terms of surface finish quality, the manipulator type, the sensor accuracy, the control strategy (position or velocity control), typical feedback servo loops, signal digitization, time digitization, inter-point polynomial interpolation, the related computer numerical control algorithms and even signal synchronization. This general framework allows to study any specific robot controlled by any typical Computer Numerical Controls (CNC). A novel formal approach to evaluate surface finish is proposed including a milling task description. A CNC module simulation block is introduced where the effect of time and signal digitization can be studied allowing to adjust sampling rates. The task is analyzed from a pure kinematics point of view, allowing to determine the best achievable result and eventually increase machining parameters such as feedrates.

In the next section, the high speed milling problem and context are explained. It includes the theortical background on parallel manipulator kinematics and CNC control The third section reviews the machining Process. The fourth section covers the geometric formalization of surface finish. The fifth section presents the path planning simulation results.

2 General Issues with Parallel Kinematic Machines

2.1 Problem Statement

To obtain five axis CNC machining at high speed feedrate levels, the Gough platform or hexapod has to be envisaged with six kinematics chains between the fixed base and the mobile platform where the tool is located, Fig. 2. Then, three possible cases can be derived. The 6UPS/6SPU configuration contains each kinematics chain with

a free prismatic actuator (P) between one Universal joint (U) and one ball joint (S); the 6RUS/6RSU includes kinematics chains constituted by a revolute actuator (R) operating a crank moving a bar including one Universal joint (U) and one ball joint (S); and finally the 6PUS/6PSU replacing the crank by a tracked prismatic actuator (P).

In reality, any robotic system is never constructed identical to the ideally designed one. A significant difference can be often osberved between the theoritical and practical configurations translating into errors on the passive joint positions of the mobile platform and the fixed base. These configuration errors will without doubt have a significant impact on milling precision. These discrepancies will usually grow following various milling operations where unperdictable wear is occuring in the joints. These will also appear following maintenance where the manipulator was reassembled if not followed by an adequate calibration procedure, (Daney 2000).

In the litterature, we can identify several procedures and sofwares analysing the characteristics and performance of robotic manipulators, (Vaishnav and Magrab 1987). These studies seek to evaluate the extremes of a certain number of criterions. More specifically, in parallel robotics, lets highlight some interesting packages proposing some level of verifications:

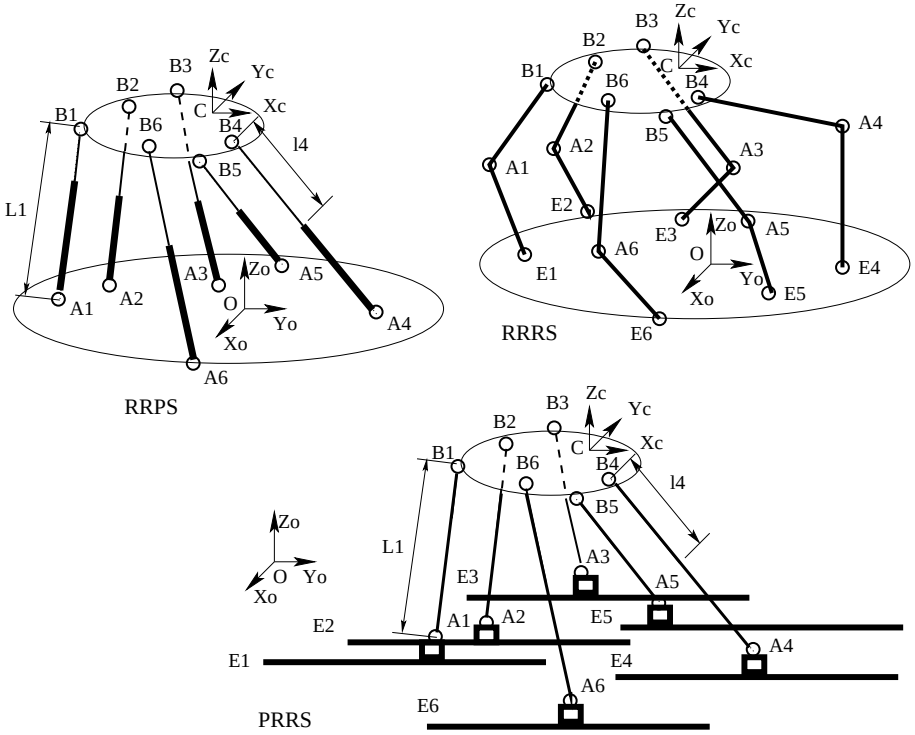

Fig. 2 Typical 6-6 parallel robots: the 6UPS/6SPU, 6RUS/6RSU and 6PUS/6PSU

1. Localisation of robot trajectories inside the workspace, (Merlet 1993) (Merlet and Mouly 1994).
2. Singularities over nominal trajectories inside the workspace, (Merlet 1993) (Nenchev and Uchiyama 1996) (Dasgupta and Mruthyunjaya 1998).
3. Power and torque of motors, (Salerni 1995).
4. Positioning errors, (Patel and Ehmann 1997) (Masory and Xiu 1998).

These analyses concern the entire workspace where performance can be affected by large variations. In many scenarios, it may be possible to achieve the task over a large portion of the workspace and then the task quality may not reach the desirable levels in certains specific areas of the workspace. The performance analysis shifted away from workspace studies towards the task trajectories themselves studying the following factors:

1. the joint travel in terms of the actuators and passive joints, (Merlet 1993)
2. the kinematics chain and platform collisions, (Merlet 1993), (Chedmail 1994), extrapolated from serial robotics work, (Tournassoud 1992)
3. maximum velocity, (Luh and Lin 1981)
4. dynamic rigidity, (Shulz et al 1999)
5. servo modeling, (Masory and Xiu 1998) (Shulz, Gao and Stanik 1999)
6. robot control, (Masory and Xiu 1998)
7. tool deformation in milling tasks, (Depince P., Hascoet and Furet 1997)
8. sixth rotation angle optimization for milling tools, (Merlet, Perng and Daney 2000) (Daney 2000)

These research works do not include all the important criterias. The displacement of the milling tool should meet surface finish requirements and tool feedrate. The second criterion will be increased in order to improve productivity. Even a constant feedrate brings important challenges on trajectories such as arcs since they mean higher accelerations.

The goal of this work is to propose tools analysing the milling task feasibility using a robot constituted by a 6-6 hexapod parallel manipulator, namely the Gough Platform, often refered as the Stewart Platform. Moreover, in order to meet the standards of the machine-tool domain, the parallel robot will be controlled by a typical CNC controller implementing classic algorithms adapted to parallel robots.

The factors influencing robot trajectory following are the sub-space of task execution, tool feedrate, position sensor accuracy and the choice of control algorithms. The milling task is in turn described by several robot trajectories. For high speed milling, surface finish is required to obtain asperities not exceeding 10 to 20 microns over the entire trajectories constituting a milling task. To qualify as high speed milling (HSM), the feedrate should reach 20 m/min and the target is even 60 m/min, classified as ultra high speed milling (UHSM).

The simulation system will require solving the kinematics problems several times. To aleviate many problems related to usual numerical methods, an exact and certified method was derived and will be applied to perform end-effector position and orientation calculations, (Rolland 2005) (Rolland 2008). This method implements ideal based techniques utilizing Groebner bases and rational univariate

representations (RUR) insuring that the produced equivalent system is excactly corresponding to the original system. The RUR system includes one univariate equation from which the real roots are calculated and proven in one-to-one bijective correspondance with the original kinematics problem. Then, proven root isolation techniques will provide for all the exact real roots. The system applies the modular black-box approach where any user can replace the selected kinematics solver by any other, at the condition that it provides for sufficient accuracy to study milling tasks.

In practice, during design, construction, start-up or after robot maintenance, these simulation tools will allow to select the complete control approach including sensors and the the path planning algorithms; The operator will be able to study the control scheme, the path following algorithms, the joint interpolation functions, the axis servo controls, the response-time of the various control levels, the effect of time discretization, the effect of digital conversions and parameter fine-tuning. The proposed tools will allow to determine milling task feasibility.

2.2 Kinematics of the general 6-6 parallel manipulator

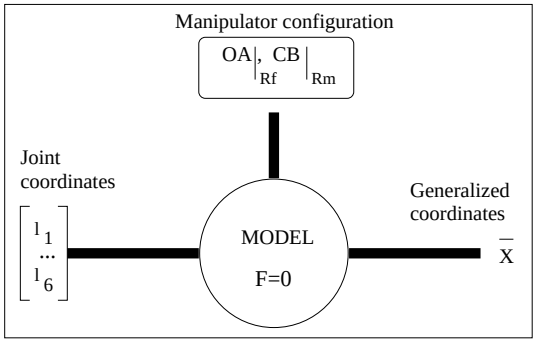

Fig. 3 Kinematics model

Any manipulator is characterized by its mechanical configuration parameters and the posture variables. The configuration parameters are thus $\mathbf{OA}_{|R_f}$, the base attachment point coordinates in R_f (the base reference frame, located at **O**), and $\mathbf{CB}_{|R_m}$, the mobile platform attachment point coordinates in R_m (the mobile platform reference frame, located at **C**). The kinematics model variables are the joint coordinates and end-effector generalized coordinates. The joint variables are described as l_i, the prismatic joint or linear actuator positions. The generalized coordinates are expressed as $\overrightarrow{X}$ comprising the end-effector position and orientation.

The kinematics model is an implicit relation between the configuration parameters and the posture variables, $F(\overrightarrow{X},\overline{L},\mathbf{OA}_{|R_f},\mathbf{CB}_{|R_m})=0$ where $\overline{L}=\{l_1,\ldots,l_6\}$. For the sake of clarity and simplicity, $\mathbf{OA}_{|R_f}$ will be replaced by $\mathbf{OA}_O$ and $\mathbf{CB}_{|R_m}$ by $\mathbf{CB}_O$.

This simulator shall only require successive passages from the joint space to the task space and vice versa, Fig. 3. The Inverse Kinematics Problem (IKP) is defined as:

Definition 1. Given the generalized coordinates of the manipulator end-effector, find the joint positions.

Accordingly, the Forward Kinematics Problem (FKP) is defined as:

Definition 2. Given the joint positions, find the generalized coordinates of the manipulator end-effector.

Usually the IKP is required to model the FKP. To solve the FKP, an exact method based on Groebner bases and rational univariate representations shall be applied, (Rolland 2005) (Rolland 2008).

The forward kinematics problem (**FKP**), Fig. 3, has been identified as a difficult problem (Raghavan and Roth 1995). Usually the *inverse kinematics problem* is required to model the **FKP** and is defined as, (Raghavan 1993): *given the generalized coordinates of the manipulator end-effector, find the joint positions.*

Accordingly, the *forward kinematics problem* is defined as, (Raghavan 1993): *given the joint positions, find the generalized coordinates of the manipulator end-effector.*

The kinematics problem can be described that, contrarily to serial manipulators, the inverse kinamtics problem yields a closed-form explicit solution and the forward kinematics involves the resolution of at least six non-linear equations. These kinematics models play an increasingly important role when robotic manipulator accuracy is decreased to the micron level.

2.3 Vectorial formulation of the implicit kinematics model

Containing as many equations as variables, vectorial formulation constructs an equation system for each kinematics chain (Dieudonne 1972), as a closed vector cycle between the A_i and B_i kinematics chain attachment points, the fixed base reference frame O and the mobile platform reference frame C. For each kinematics chain, an implicit function $\overrightarrow{A_iB_i}=U_1(X)$ can be written between joint positions A_i and B_i. Each vector $\overrightarrow{A_iB_i}$ is expressed knowing the joint coordinates $\overline{L}$ and X giving function $U_2(X,\overline{L})$. The following equality has to be solved: $U_1(X)=U_2(X,\overline{L})$. The distance between A_i and B_i is set to l_i. Thus, the end-effector position X or C can be derived by one platform displacement $\overrightarrow{OC}$ and then one platform general rotation expressed by the rotation matrix $\mathscr{R}$. For each distinct platform point $\overrightarrow{B_{i\,O}}$ with

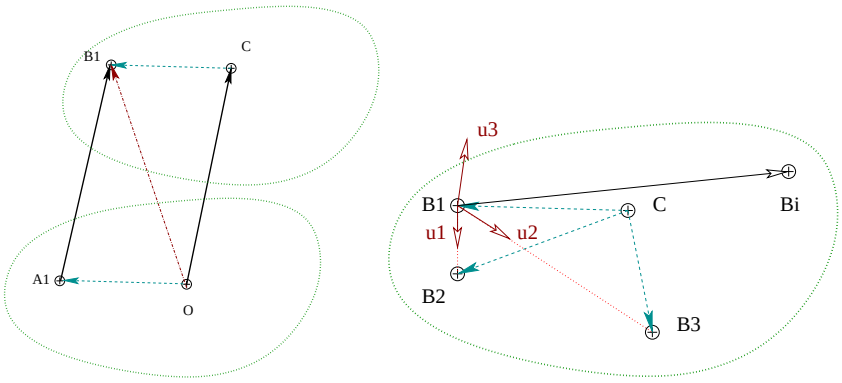

Fig. 4 Kinematics chain and mobile platform vectors

$i = 1,...,6$, see Fig. 4, the position can be calculated in terms of the base reference frame, (Merlet 1997):

$$\overrightarrow{OB_{iO}} = \overrightarrow{OC} + \mathscr{R}\overrightarrow{CB_i} \tag{1}$$

The vectorial formulation evolves as a displacement based equation system using the following relation :

$$\overrightarrow{A_iB_i} = \overrightarrow{OC} + \mathscr{R}\overrightarrow{CB_i} - \overrightarrow{OA_i} \tag{2}$$

These six equations cannot be applied as such. Hence, each kinematics chain can be expressed using the distance norm constraint, (Merlet 1997):

$$l_i^2 = ||A_iB_i||^2 \tag{3}$$

The rotation matrix $\mathscr{R}$ can be written utilizing various orientation models with their specific rotation variable sets such as navigaton angles (yaw, pitch and roll), Euler angles, quaternions or even taking the nine rotation matrix components as variables, (Rolland 2008). Implementing the equation 2 directly, various displacement based equation models can be derived depending on the selected orientation variables, (Rolland 2008).

Another excellent approach is called the position based modeling and consists in considering any rigid object to be positioned into three dimensional space by three distinct points, Fig. 4. Any rigid body three points are actually characterized by three distinct distance constraints and a pointing axis which remain constant. This principle was then applied to the forward kinematics model of parallel manipulators by Lazard (Lazard 1993). It is easy to choose three distinct points which are not co-linear on most mobile platforms. These three points are usually selected to coincide with three joint centers connecting the mobile platform to the kinematics chains allowing to utilize the vectorial model, 4 and to rewrite of $\overrightarrow{A_iB_i}$, 2 as it is explained in details in (Rolland 2008).

Two reasons justify the choice of the position based model. Every variable yield the same units and their ranges are equivalent leading to the same weight in the equation system. The rotation impact is included into the point parameters and made equivalent to the translation impact.

The coordinates of the three distinct joint center points become the nine variables from which constraints equation can be written. The three platform distinct points are usually selected as the three first joint centers, namely B_1, B_2 anbd B_3. Each coordinate of the selected joint centers becomes a variable. The nine end-effector variables are set to : $\overrightarrow{OB}_{i|O} = [x_i, y_i, z_i]$ for $i = 1 \dots 3$. To simplify computations, we choose one non-Cartesian reference frame R_{b_1} to be located at B_1 joint center. Then, we define u_1, u_2 and u_3 as R_{b_1} reference frame axes which are calculated by:

$$u_1 = \frac{\overrightarrow{B_1B_2}}{||\overrightarrow{B_1B_2}||},\ u_2 = \frac{\overrightarrow{B_1B_3}}{||\overrightarrow{B_1B_3}||},\ u_3 = u_1 \wedge u_2 \tag{4}$$

This new reference frame R_{b_1} is applied instead of R_m as the mobile platform Cartesian reference frame and has its origin located at B_1 and the reference frame axes u_1 and u_2 point towards B_2 and B_3 respectively. The third reference frame u_3 points perpendicular to the plane determined by B_1, B_2 and B_3. It becomes the mobile platform pointing axis. This transformation is achieved to produce a simpler equation system.

Knowing that the mobile platform is supposed infinitely rigid, any platform point M can be expressed in the reference frame R_{b_1} by calculating the following linear composition:

$$\overrightarrow{B_1M} = a_M u_1 + b_M u_2 + c_M u_3 \tag{5}$$

where a_M, b_M, c_M are constants in terms of these three points. Hence, in the case of the **IKP** , the constants are noted $a_{B_i}, b_{B_i}, c_{B_i}$, $i = i \dots 6$ and can explicitly be deduced from the mobile platform fixed distances $\mathbf{CB}_{|C}$ by solving the following linear system of equations :

$$\overrightarrow{B_1B}_{i|R_{b_1}} = a_{B_i} u_1 + b_{B_i} u_2 + c_{B_i} u_3\ ,\ i = 1 \dots 6. \tag{6}$$

where $\overrightarrow{B_1B}_{i|R_{b_1}} = \overrightarrow{B_1B}_{i|C}$.

Note that the mobile platform fixed distances $\mathbf{CB}_{|C}$ are given by the configuration which is obtained from the design values or deduced from a calibration procedure after the Gough platform manipulator construction. The configuration file is providing the position of all six joints of the mobile platform relative to the mobile platform reference frame and this ensure that the points belong to the same rigid body which is the mobile platform.

Equation 7 requires that we calculate the configuration distances with:

$$\overrightarrow{B_1B}_{i|C} = \overrightarrow{CB_i} - \overrightarrow{CB_1}\ ,\ i = 1 \dots 6. \tag{7}$$

Hence, the remaining three mobile platform joint centers B_4, B_5 anbd B_6 are expressed in terms of the nine end-effector variables.

Using the relations Eq. (6), the distance constraint equations $l_i^2 = ||\overrightarrow{A_i B_{i|O}}||^2$, $i = 1 \ldots 6$ can be expressed Thus, for $i = 1 \ldots 6$, the **IKP** is obtained by isolating the l_i actuator variables in the six following equations:

$$l_i^2 = (x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 + (z_i - OA_{iz})^2 \ , \ i = 1 \ldots 3 \tag{8}$$

$$l_i^2 = ||\overrightarrow{B_1 B_{i|_{R_{b_1}}}} - \overrightarrow{OA_{iO}}||^2 \ , \ i = 4 \ldots 6 \tag{9}$$

2.4 The Inverse Kinematics Problem

The 3 or 9 are actually the two general forms of the explicit IKP.

2.5 The Forward Kinematics Problem

For the general Gough platform parallel manipulator, it is actually not possible to express the FKP directly or explicitely, [**?**]. We have to revert to the **IKP** expression which gives an algebraic system comprising six equations in terms of three point variables : $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$, Eq. (9). This system contains algebraic (polynomial) functions which can be handled by the numerical solvers implemented in all genetic algorithms.

The usual method advocated for writing the FKP equation system starts by rewriting the IKP as functions. This produces an algebraic system of three leg equations and three functions in terms of the nine variables: x_i, y_i, z_i , $i = 1, 2, 3$.

$$F_i = (x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 + (z_i - OA_{iz})^2 - l_i^2 \ , \ i = 1 \ldots 3 \tag{10}$$

$$F_i = ||\overrightarrow{B_i}_{|_{R_{b_1}}} - \overrightarrow{OA_{iO}}||^2 - l_i^2 \ , \ i = 4 \ldots 6 \tag{11}$$

When solving the FKP with numeric or algebraic methods, it is necessary to provide a zero-dimensionnal system, meaning an equation system which contains as many equations as their are variables, [**?**] and [**?**]. In this case, this means that to the six equations provided by the IKP, three more shall be selected to close the system.

Moreover, the actual FKP is derived directly from the IKP model, 11, and it does not provide for any information to constrain the position of the mobile platform joint positions which are necessary to describe the FKP.

Hence, to complete the algebraic system and to constrain the mobile platform joint positions, three constraints are derived from the following three functions. Two functions can be written using two characteristic platform distances, expressed as

norms between the B_1, B_2 distinct points and the B_1, B_3 ones. The computations will select the variables which are only at the right distance from the B_1 reference joint point. These constraint equations require one last equation. The points are known relative to each other in terms of distance but the mobile platform alignement is left undetrmined. To alleviate this problem, the third constraint equation will determine where the mobile platform is pointing. The pointing vector is selected as the one perpendicular to the three points B_i , $i = 1,2,3$ by calculating the vectorial multiplication of the two vectors separating B_2 and B_3 from B_1:

$$F_7 = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 - ||\overrightarrow{B_2B_1}_{|_{R_{b_1}}}||^2 \tag{12}$$

$$F_8 = (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2 - ||\overrightarrow{B_3B_1}_{|_{R_{b_1}}}||^2 \tag{13}$$

$$F_9 = (x_3 - x_1)(x_2 - x_1) + (y_3 - y_1)(y_2 - y_1) + (z_3 - z_1)(z_2 - z_1) - ||\overrightarrow{B_3B_1}_{|_{R_{b_1}}}|| \wedge ||\overrightarrow{B_2B_1}_{|_{R_{b_1}}}|| \tag{14}$$

The choice of F_9, the last function, provided an important mobile platform constraint related to the pointing axis. For F_9, it would be possible to write a function related to the distance between B_2 and B_3 but our experience shows us that it does lead to better results then the platform pointing function.

The result constitutes then an algebraic system with nine equations in the former nine unknowns.

2.6 Machine Tool Control

In a high speed milling machine, a typical Gough platform being a general **6-6** or hexapod robot is constituted by several parts driven by a controller connected to a remotely located CAD-CAM computer, (Mery 1997), and, for each kinematics chain, it can be described by the following components:

- A manipulator end-effector where its position and orientation are indirectly controlled, since an external sensor system cannot be implemented in milling operations.
- One prismatic axis per kinematics chain identified as an active joint.
- One DC electrical motor on each prismatic axis as one actuator.
- One position sensor on each prismatic axis measuring its length.
- One pulse-width-modulation (PWM) amplifier for each DC electrical motor.

As it is explained in (Mery 1997), one CNC machine-tool is essentially considered identical to a robot achieving arbitrary and predetermined continuous path following.

Definition 3. NCMT - A numerically controlled machine-tool is defined as high precision machine-tool associated to a control unit of quality, (Marty, Cassagnes and Martin 1993).

A milling task is achieved by an NCMT divided into five main elements, (Mery 1997):

- The tool: the device which performs the process of material removal.
- The end-effector: the unit which holds and activates the tool.
- The tool carrier machine: the specific robotic manipulator including actuators and instrumentation.
- The CNC: the computer numerical controller.

Mery defines a computer numerical controller (CNC) in the following terms, (Mery 1997):

The CNC is defined as the control system capable to manage the machine-tool and its control in order to follow a program achieving a milling task.

Practically, the CNC handles a written program in standard format constituded by G codes from the ISO standard, (Magnin and Urso 1991) (Marty, Cassagnes and Martin 1993). Note that this machine-tool industry considers this format mandatory for machine-tool controls. Any simulation package shall consider that CNC systems handle these codes and simulate their operations. In Fig. 7, the basic elements of any CNC are presented. The goal of such control system is to ensure that any machining task is carried automatically. This particularily includes trajectory pursuit of the robot and the operation of the tool.

In typical CNC, the control unit is further divided into three control stages or levels: the off-line CAD-CAM providing the task set-points describing the nominal paths, the on-line nominal path following as the upper controller level and the motor servoing as the lower controller level, usually driving directly the actuators by implementing one **PID** feedback loop for each axis. The connection between each

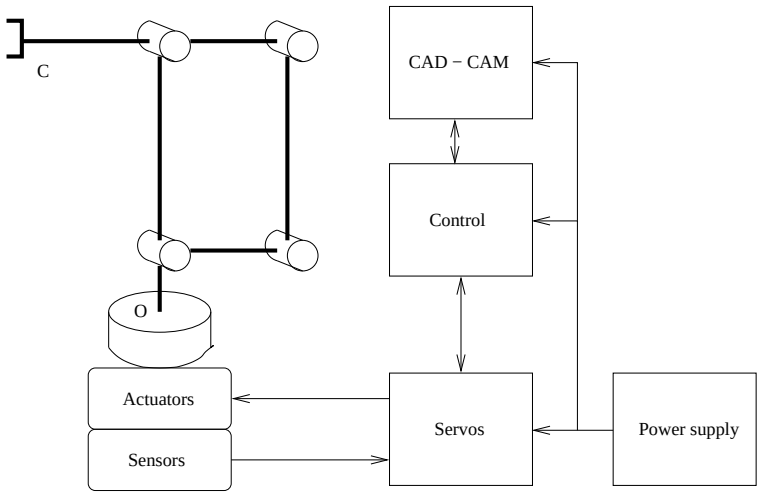

Fig. 5 Typical robot schematic

Fig. 6 Example of a CNC machine-tool

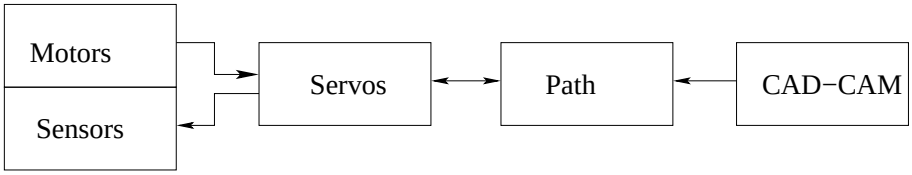

Fig. 7 Block diagram of machine-tool numerical control

stage is through data tables. Each stage operates in discrete time according its own cycle time or sampling rates. Lets define the following cycle times or sampling rates:

- T_c: the task trajectory set-point file sampling rate produced by the CAM program.
- T_p: the path following cycle time corresponding to the time required to calculate the joint servo trajectory set-points.
- T_s: the motor servo cycle time corresponding to the time dedicated to PID loop computation.
- T_a: the motor amplifier sampling rate which gives the time at which their output is being refreshed.

The simulation module will allow to test and verify the three first cycle times. The amplifier sampling rates will not be included in the simulation work. The task follows one or several nominal functions from which discretization produces the task path file containing a large number of points being dependant on the sampling

rates. The number of points will have an impact on surface finish and impact CNC's ability to follow the nominal path.

The machine-tool operates in a spatial continuous domain which is completely described by 6 dimensions (3 translations and 3 rotations), $\lambda = 6$, with parameters $\in \Re$. To execute a milling task, the path following algorithm may require from three to five axes control. The sixth axis corresponds to the tool spindle rotation axis and therefore does not participate to the trajectory pursuit. The CNC should then receive five analog inputs or encoder inputs for actuator axis positions and drive five analog or direct pulse-width-modulation outputs for actuator positioning. The simulation will not include the tool spindle axis angular speed control.

The CNC can either implement one of the two control types: position and speed control, (Coiffet 1986):

- Position control is preferred when you can calculate the IKP. Joint position control follows the trajectory profile at the axis level from interpolated point to interpolated point and does not control the velocities between these points leading to a discrepency between the exact nominal trajectory and the achieved trajectory at the tool level. If the range of motion is important then the robot reaches its destination with larger inaccuracies. The traditional solution is to slowdown robots.
- Speed control is based on small displacements and implements the computation of the inverse Jacobian matrix. You will need to calculate the FKP.

2.7 Task space conversion to joint space

In principle, implemented in the off-line CAM, the trajectory planning algorithm calculates one inverse kinematics problem from the Cartesian-space set-point trajectory functions to determine the six actuator-space fonctions which are then called the joint set-point trajectories. The real continuous signals are computed from these fonctions. Then, the continuous signals are sampled according to the first level cycle time T_p corresponding to the time required to calculate these points and the signal magnitude discretized into a certain number of bits.

When planning and following any task path, the upper level controller calculates, in advance and in real time at each $t = kT_p$ for $k = 1 \ldots n_p$ where n_p is the number of points provided by CAD/CAM, all interpolated points between joint set-points that will then serve as set-points to the six lower level servo controllers driving the actuators. It is interpolating these reference values using a polynomial interpolation function or blended polynomial function sets. Since the majority of control algorithms calculate the instructions in the joint space and there are no sensors for performing a return position on the end-effector where the milling tool is located in task space, then the controller must perform the forward kinematics problems (FKP) calculations to return the tool Cartesian position and orientation.

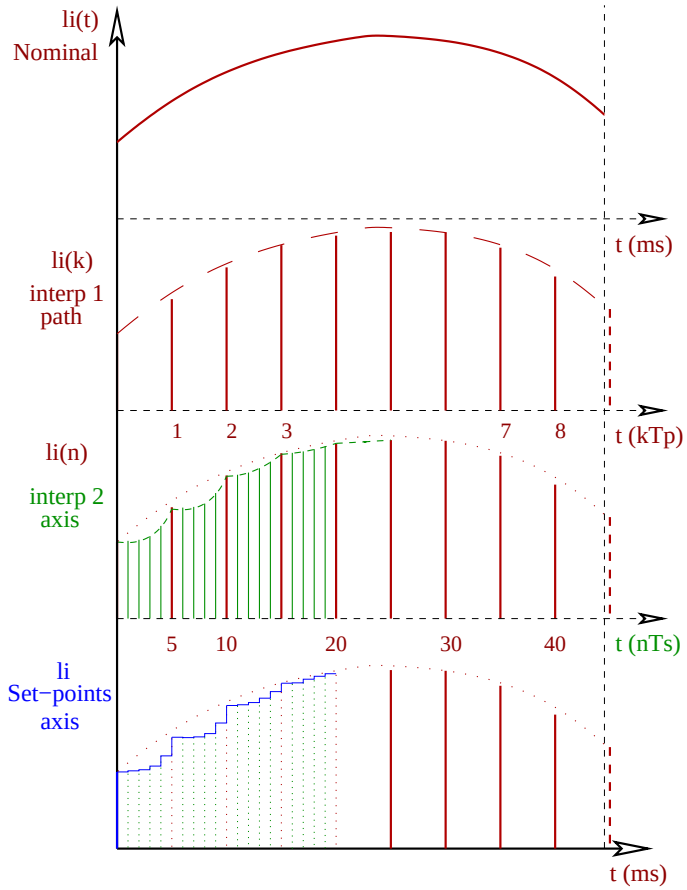

Fig. 8 Example of signal and time digitization of nominal actuator function

3 CNC Handling of the Machining Process

3.1 Introduction on milling

Tournassoud emphasizes that the robotic task is defined in terms of constraint verification for a set of measurements applied on the system, (Tournassoud 1992). All the performance of a robotic task is then reduced to trajectory tracking and is expressed as follows:

Let q_0 be an initial configuration and q_f a final configuration, both achievable, that is to say, within the robot workspace and non-singular, then one trajectory $H(\lambda)$ with $\lambda \in [0,1]$ is calculated in the free space, such that $H(0) = q_0$ and $H(1) = q_f$.

Nilsson and Udupa proposed initial work on robotic tasks for specific robots (Nilsson 1969) (Udupa 1977). In (Lozano-Perez and Wesley 1979), a first general approach included the first trajectory planning algorithm. In (Brady et al. 1982) (Latombe 1991), numerous work summary indicates mostly obstacles avoidance. Coiffet extends the application of constraints to the end-effector member maintained in a constant orientation, singularity avoidance and sampling rates, (Coiffet 1986). Specifically, the milling goal is to produce a workpiece by material removal (Mery 1997). The end result is an object whose surfaces are characterized by a certain quality of surface finish. This quality is normally defined by a permissible error denoted by a tolerance in terms of the part's drawing and an index describing the surface quality. The part is thus represented like a geometric object drawn using one typical CAD software. The CAM functionality translates the virtual object shape into a certain number of paths spanning and scanning the part. These task paths are the CNC set-points in one machining file.

The machining path is defined as the functional path that determines the contact position between the tool tip and the workpiece, (Chedmail 1994).

3.2 Description

Several parameters are required to proceed with tool operation description: tool tip position, tool tip orientation, tool feedrate, nominal trajectory to follow during machining and tool rotational speed. These parameters, except the last one, have been integrated into the simulation tool since they are all specifically related to the robot operation. Machining consists of a set of task trajectories, (10).

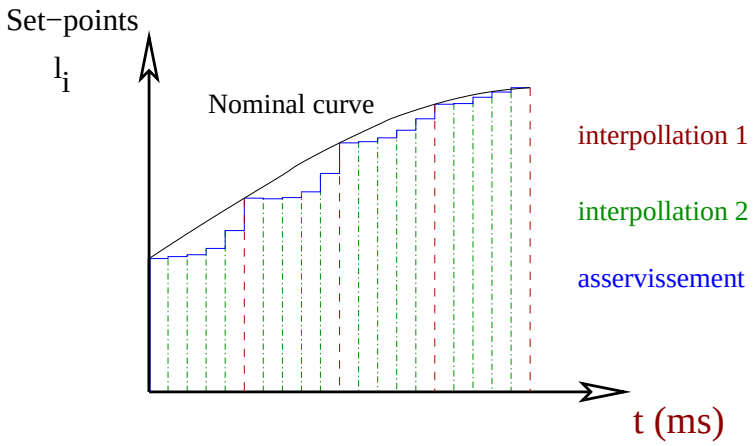

Fig. 9 Details of actuator signal digitization

Simulation proceeds with surfacing tasks which are easy to visualize and simple to represent. However, from the point-of-view of the robot control, they are not necessarily easier with a parallel manipulator featuring non-linear kinematics.

Definition 4. Let H be a machining task, cut into a set of m paths, $H = h_1, h_2, \ldots, h_m$. Let τ_{tot} be called the total time to perform all machining and let τ_i be path i duration.

The trajectory P_d departure point and the P_f arrival point or final point are respectively correspoding to time $t = 0$ and $t = \tau_{tot}$ (Taylor 1979) (Luh and Lin 1981). We know that the end-effector is at rest at the point of departure and arrival, where the velocity and acceleration are then set to zero at these points. For each task path h_i, the start point and the end point are made to respectively correspond to times $t_i = \sum_{k=1}^{i} \tau_{k-1}$ and $t = \sum_{k=1}^{i} \tau_k$.

The realization of the task is essentially reduced to the location of the tool tip in task space. According to Chedmail and Mery (Chedmail 1994) (Mery 1997), the majority of machining tasks consists of two types of paths, Fig. 10: Continuous machining path and transition paths between them when there is no contact between the tool and the part. The transitions can be described as robotics classical point-to-point motion which should last a minimum amount of time, (Mery 1997), justifying the parallel robot choice.

Definition 5. The functional paths are defined as continuous paths corresponding to the machining process of the workpiece, (Chedmail 1994).

These paths are usually made at a constant feedrate to ensure the quality of the finished surface. Each functional path is defined by two nominal functions: one function describing the tool Cartesian position, a second function describing orientations. For example, in Figure (6.6), we observe that the straight line segments are the machining paths. A task is defined by a succession of displacements when the tool is actually in operation, (Mery 1997):

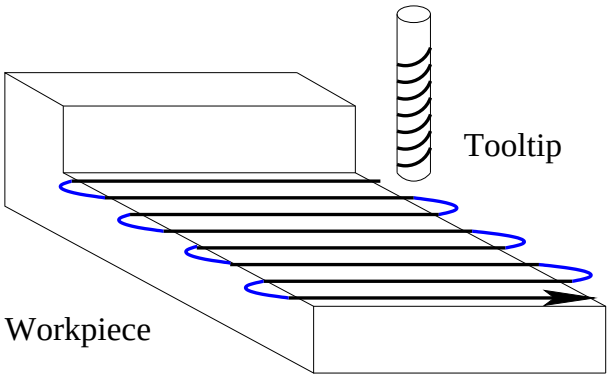

Fig. 10 Example of a typical milling task

$$X_i^{nom}|_{R_f} = (x_i(t), y_i(t), z_i(t), \theta_{1_i}(t), \theta_{2_i}(t), \theta_{3_i}(t))^t \tag{15}$$

Typically, milling tasks generally consist of sets of arcs, straight lines, spirals and eventually splines. The robot moves the end-effector at constant feedrate. This translates by the following Cartesian constraint: $||\overrightarrow{V_c(t)}|| = F_r$ where F_r is a constant. Thus, the speed being the velocity magnitude is always constant. These tasks are usually defined on planes parallel to the XY plane of R_f, the robot reference frame, meaning that we must ensure that: $P_d[z] = P_f[z] = P_i[z]$.

3.3 Trajectory Position Nominal Function

A task is defined by a parametered nominal function set where each function is defined as $\overrightarrow{P}^{nom}(\lambda)$ with $\lambda \in [0,1]$ to exactly describe the task trajectory to follow. It covers the vast majority of machining work in the industry, (Mery 1997). For each segment, we assign $\lambda = 0$ to the start point P_d to and end point P_f with $\lambda = 1$. The task will seek to move the robot tool along a function whose general implicit form is defined as follows:

$$\overrightarrow{P}^{nom}(\lambda) = f(P_d, P_f, \lambda) \tag{16}$$

In the case of a constant feedrate, τ represents the time to complete a path, one can express the parameter λ versus time t according to the following relationship: $\lambda = \frac{t}{\tau}$. The implicit function becomes:

$$\overrightarrow{P}^{nom}(t) = f(P_d, P_f, t, \tau). \tag{17}$$

Knowing that the travelled distance δS is the actual distance along the path between P_d the start point and P_f the final point and is calculated by $\delta S = F_r\ \tau$ where F_r is the constant tool feedrate. Then, the implicit function is expressed by:

$$\overrightarrow{P}^{nom}(t) = f(P_d, P_f, t, F_r). \tag{18}$$

This form will be retained for the simulation since, in the machine-tool domain, it is customary to specify the machining tasks in terms of initial points, endpoints, path type and feedrate, (Mery 1997).

3.3.1 Trajectory in a general plane

For reasons of simplicity, the machining majority is arranged on planes parallel to the XY plane.

3.3.2 Straight line segment formulation

The straight line segment starts by calculating the trajectory time :

$$\tau = \frac{||P_d - P_f||}{F_r} \tag{19}$$

Then, the segment equation is determine :

$$\overrightarrow{P}^{nom} = P_d + \frac{(P_f - P_d)}{\tau}\, t \tag{20}$$

3.3.3 Arc formulation

It is therefore proposed several methods to evaluate an arc depending on the data entered :

- First case - start point: P_d, end point: P_f, feedrate: F_r, centre of rotation: CC and radius: r;
- Second case - start point: P_d or end point: P_f, displacement angle: $\delta\phi$, feedrate: F_r, centre of rotation : CC and radius: r;
- Third case: start angle: ϕ, end angle: Φ, feedrate:: F_r, centre of rotation : CC and radius: r.

Two additional inputs are necessary. To calculate the path as such, P_f is not directly used and it will only used calculate the total time τ.

Firstly, the angular velocity is calculated and then, the circular fonction is instantiated. Particular attention must be brought to the ϕ angle calculation which corresponds to either the starting point or end point :

- to match the start time which is not always zero,
- to proceed with quadrant verification related to trigonometric function inversion.

The following algorithm is implemented :
This formulation has two disadvantages :

- The path calculation cannot include tasks where the arc is greater than a circle.
- We must ensure that the end point is on the arc.

The implemented solution is rather technical since it is cutting an arc being longer then a circle into two arcs. If the arc rotates several turns, this technique is applied whenever a complete rotation of 2π is encountered.

An alternative is to replace the endpoint P_f by a final angle Φ. It can handle circles and arcs with a rotation angle larger than 2π. The algorithm becomes:

Another alternative is to replace the starting point P_d by a start angle ϕ.

These different Arc algorithms would adapt perfercly to object oriented programmning.

Arc(Input) $\omega = \frac{F_r}{r}$
if $P_d[2] - CC[2] \geq 0$ then
$\phi = arccos(\frac{P_d[1]-CC[1]}{||P_d-CC||})$
else
$\phi = \pi - arccos(\frac{P_d[1]-CC[1]}{||P_d-CC||})$
if $P_f[2] - CC[2] \geq 0$ then
$\Phi = arccos(\frac{P_f[1]-CC[1]}{||P_f-CC||})$
else
$\Phi = \pi - arccos(\frac{P_f[1]-CC[1]}{||P_f-CC||})$
$\overrightarrow{P}^{nom} = [CC[1] + rcos(\omega t + \phi), CC[2] + rsin(\omega t + \phi), P_d[3]]$
$tau = \frac{\Phi - \phi}{\omega}$
return($\overrightarrow{P}^{nom}, \tau$)

Arc(Input) $\omega = \frac{F_r}{r}$
if $P_d[2] - CC[2] \geq 0$ then
$\phi = arccos(\frac{P_d[1]-CC[1]}{||P_d-CC||})$
else
$\phi = \pi - arccos(\frac{P_d[1]-CC[1]}{||P_d-CC||})$
$\overrightarrow{P}^{nom} = [CC[1] + rcos(\omega t + \phi), CC[2] + rsin(\omega t + \phi), P_d[3]]$
$tau = \frac{\Phi - \phi}{\omega}$
return($\overrightarrow{P}^{nom}, \tau$)

Arc(Input) $\omega = \frac{F_r}{r}$
$\overrightarrow{P}^{nom} = [CC[1] + rcos(\omega t + \phi), CC[2] + rsin(\omega t + \phi), P_d[3]]$
$\tau = \frac{\Phi - \phi}{\omega}$
return($\overrightarrow{P}^{nom}, \tau$)

3.3.4 Constant velocity circle formulation

Using arc functions, any circle can be parametered, taking another path time calculation which takes into account that the path is returning to starting point :

$$\tau = \frac{2\pi}{\omega} \tag{21}$$

And if the task involves machining a plurality of turns, this value is multiplied by the number of turns.

3.3.5 Variable radius spiral

Using arc functions, spirals can be parametered. A spiral is in most cases located in a plane parallel to the XY plane.

Hence, the radius becomes a parametric function versus time. The evaluation algorithm of the spiral function is the same as for the arc.

The first idea that comes to mind is to follow the path at constant feedrate, meaning, it results in a spiral at constant tangential velocity.

Suppose that the path is traversed in the concentric direction, when you approach the center, the angular velocity tends to infinity. So, there is a minimum radius below which the angular velocity exceeds the physical capabilities of the robot speed and acceleration. This also means it will be impossible to machine a very small disk.

The solution is not mathematical but technical: at the time of machining preparation, the operator will mill the spiral until minimum radius is reached and then finish milling the small residual disk by successive straight line passes.

A minimum radius calculation is provided in terms of the maximum feedrate. One maximum acceleration calculation is added.

One alternative consists in traversing the spiral constant angular velocity, for this, the function of tangential velocity corresponding to the feedforward will vary accordingly: $V_c(t) = r(t)\ \omega$. When you approach the center, then radius thends towards zero, the tangential velocity tends towards zero and then the machining time tends towards infinity. Again, this is another case which is not physically realistic. The machine-tool cannot cut a small disk of radius using spiral paths below a certain radius or the operator will need to select a smaller machine-tool.

The calculation of minimum radius will in terms of minimal speed which is the tool tip velocity vector magnitude.

3.3.6 Helix formulation

The helix or screw or spin ressembles the spiral, except that the radius does not vary against the helix axis or rotation. To simplify calculations, the tool axis will be positionned parallel to the helix axis, meaning perpendicular to the XY plane, then the tooltip position will vary only in terms of its vertical position but its position projection on the XY plane will coincide with a constant radius circle.

The arc equation can be implemented where the vertical component which is calculated by one parametric function which is usually set to be linear :

$$P_z^{nom} = \nu\ \omega t \tag{22}$$

The vertical helix is determined by the screw thread ν.

If we implement a general pointing axis $\overrightarrow{u}$, then the formulation becomes:

$$\overrightarrow{P}^{nom} = \{calR\,\overrightarrow{P}_z^{nom} \tag{23}$$

where $\overrightarrow{P}_z^{nom}$ describes the vertical pointing helix and $\{calR$ is the rotation matrix related to the displacement of the helix axis from the vertical position to its actual position. The rotation matrix handles orientation changes which can be expressed by equation 29 explained in the following orientation section.

3.3.7 Constant speed spline formulation

A spline is a set of $f_i(\lambda)$ with $i = 1,\ldots,n$ parameterized polynomial functions. Let a nominal milling path be defined by a set of these functions. We will focus on one of them where the required trajectory continuity will be to ensure that the end of a $f_i(\lambda)$ spline trajectory is aligned and continuous with the beginning of next $f_{i+1}(\lambda)$ function, matching velocities, accelerations and jerks at transitions.

Let the vector parameters $A_0,\ldots,A_n$ be constitued by constant real values, we evaluate a path such that :

$$\overrightarrow{P}^{nom} = A_0 + tA_1 \ldots A_n + t^n \tag{24}$$

We seek to establish these equations in order to have a constant feed rate is $F_r = ||\frac{d}{dt}(P^{nom}(t))||$.

3.4 Trajectory Orientation Nominal Function

The end-effector motion can be modeled to obtain decoupled translation and rotation displacements (Coiffet 1996) (Dombre and Khalil 1999). Many methods exist for modeling orientations and their displacements: navigation angles (roll, pitch, yaw), two types of Euler angles, quaternions, Rordrigues parameters, the normal vector to the mobile platform, the pointing vector of the tool axis, etc.

3.4.1 Constant orientation modelling

The first set of encountered trajectories are the so-called 3 DOFs milling tasks or surfacing tasks. These are performed at constant orientation where the tool axis is kept perpendicular to the workpiece. To simplify calculations, the parallel robot is positionned to keep the tool axis parallel to the base reference frame z axis. Then, the rotation matrix is equal to the identity matrix. This means that the end-effector axis is set to $N_c = [0,0,1]$ which is selected for orientation formulation, since many rotation formulations lead to singularities when $R = I$ (Euler angles , Bryant angles, etc.) as shown in (Coiffet 1996) (Dombre E. and Khalil 1999). Path planning can be simplified with the calculations avoiding rotation matrix transformations. This axis can be called pointing axis or normal axis since it is usually selected the mobile platform normal axis coinciding with the tool axis.

It is possible to apply the same formulation for any other constant pointing axis displacement. The normal vector becomes $N_c = [n_x, n_y, n_z]$. However, in this case, the normal rotation parameters are the converted into a rotation matrix. This is used to calculate the IKP in trajectory analysis.

3.4.2 Variable orientation around one standard axis

The former tasks can be extrapolated into 4 DOFs milling tasks where the displacement includes one rotation about a single axis staying constant over the entire trajectory.

For reasons of simplicity, we chose to perform this rotation around an axis parallel to the base reference frame X or Y axis. The rotation formulation will be expressed with the intuitive angles of aviation (yaw, roll, pitch).

Konwing that the CAM file data provides orientation in terms of tool axis orientation. Thus, the normal axis will be converted in a the rotation matrix format exploitable by kinematics calculations. The conversion application will either produce θ_x or θ_y. The third angle θ_z will not impact orientation sinced it corresponds to the tool spindle angle. It must be either determined or optimized using a performance criterion, (Daney 2002).

The rotation around an arbitrary axis is representated by a parameterized function that can be instantiated with initial and final rotation θ_d or θ_f. Then, we obtain a function θ_{nom}:

$$\theta_{nom} = F_{ori}(\theta, \theta_d, \theta_f, t, \tau) \tag{25}$$

3.4.3 General variable orientation

The former tasks can be generalized into 5 DOFs milling tasks where the displacement includes one rotation about a single axis being arbitrary and changing over the entire trajectory.

We could use the previous method to express changes of uniform rotations using standard formulations (aviation angles , Euler angles , quaternion , etc. .) but these formulation are not intuitive and some yield mathematical singularities at specific configurations. Moreover, navigation angles do not correspond to the θ_x, θ_y and θ_z since these last angles appear one after the other in the kinematics calculations based on homogeneous transforms.

Euler's theorem states that a finite displacement from a rotational movement about a fixed point is equivalent to a rotation of a certain angle about an axis passing through that fixed point, (Coiffet 1996). Generally, for any robot, any orientation can be calculated by the following rotation matrix, (Luh and Lin 1981) (Coiffet 1996) (Dombre and Khalil 1999) :

$$\mathscr{R} = [\overline{s}, \overline{n}, \overline{a}] \tag{26}$$

where $\overline{n},\overline{n},\overline{a}$ are the unitary normal vector being colinear to the tool axis, the sliding unitary vector and the approach vector being colinear with the tooltip feedrate velocity vector respectively, Fig. 11. The sliding unitary vector is also refered to the transverse vector being perpendicular to the two others.

The rotation matrix can be generally expressed as:

$$\mathscr{R} = \mathbf{I}\, cos(\theta) + \overline{q}\,\overline{q}^{t}\ (1 - cos(\theta)) + [q^{x}]\, sin(\theta) \tag{27}$$

Where q^{x} is the following matrix :

$$\mathscr{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{28}$$

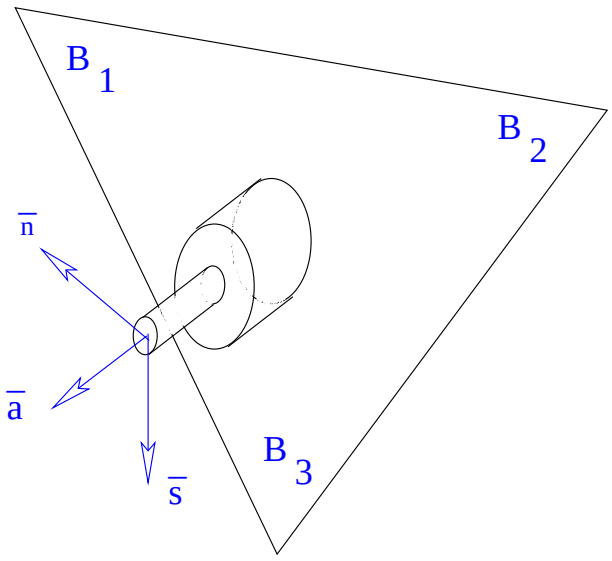

Fig. 11 The axis $\overline{n},\overline{s},\overline{a}$ of the robot end-effector

Since the CAM program formalizes tool orientation using N_c the normal unitary axis of the end-effector, the Rodrigues formula represents a displacement of a rotation angle around the resulting general motion rotation axis, (Coiffet 1996). This formulation is related to quaternion formulation, (Coiffet 1996) (Dombre E. and Khalil 1999). Let the vector $vectP$ be subjected to a finite rotational displacement of an angle θ about an axis $\overline{q} = [q_x, q_y, q_z]$ then the resulting rotation matrix is :

$$\mathscr{R} = \mathbf{I}\, cos(\theta) + \overline{q}\,\overline{q}^{t}\ (1 - cos(\theta)) + [q^{x}]\, sin(\theta) \tag{29}$$

Where q^{x} is the following matrix :

$$[q^x] = \begin{pmatrix} q_x^2\, a & q_x q_y a - q_z sin\theta & q_x q_z a + q_y sin\theta \\ q_x q_y a + q_y sin\theta & q_y^2 a + cos\theta & q_y q_z a - q_x sin\theta \\ q_x q_z a - q_y sin\theta & q_y q_z a + q_x sin\theta & q_z^2 a + cos\theta \end{pmatrix} \tag{30}$$

The inverse problem is then solved, (Coiffet 1996). Given the rotation matrix computed from the normal vector $\overrightarrow{N_c}$, $\overline{q}$ and θ are then calculated. Note that the passage from the mobile platform vector to the rotation matrix can be done with most formulations.

The trigonometric functions are derived for positive rotations constraining the angle to: $0 < \theta < \pi$:

$$c\theta = (r_{11} + r_{22} + r_{33} - 1)/2 \tag{31}$$

$$s\theta = \pm[(r_{32} - r_{23})^2 + (r_{13} - r_{31})^2 + (r_{21} - r_{12})^2]/2 \tag{32}$$

The angle value is found by calculating : $\theta = arctan(c\theta\ /s\theta)$. For improved accuracy, the following equations will be implemented :

$$\overline{q} = \begin{pmatrix} sign(r_{32} - r_{23})[\frac{r_{11} - c\theta}{1 - c\theta}]^{1/2} \\ sign(r_{13} - r_{31})[\frac{r_{22} - c\theta}{1 - c\theta}]^{1/2} \\ sign(r_{21} - r_{12})[\frac{r_{33} - c\theta}{1 - c\theta}]^{1/2} \end{pmatrix} \tag{33}$$

An orientation change is described passing from matrix $\mathscr{R}_{i-1}$ to matrix $\mathscr{R}_{\rangle}$ by rotation variable θ_i around a predefined arbitrary vector $\overline{q}$, Fig. 12. In the general case, note that the $\overline{q}$ axis does not necessarily correspond to the $\overrightarrow{N_c}$ mobile platform normal vector.

The rotation axis can be determined by calculating :

$$\overline{q} = \frac{1}{2sin(\theta_i)} \begin{bmatrix} (\overline{a}^i)^t\ \overline{s}^{i-1} - (\overline{s}^i)^t\ \overline{a}^{i-1} \\ (\overline{n}^i)^t\ \overline{a}^{i-1} - (\overline{a}^i)^t\ \overline{n}^{i-1} \\ (\overline{s}^i)^t\ \overline{n}^{i-1} - (\overline{n}^i)^t\ \overline{s}^{i-1} \end{bmatrix} \tag{34}$$

The rotation angle is obtiained henceforth :

$$\theta_i = arccos(\frac{(\overline{n}^i)^t\ \overline{n}^{i-1} + (\overline{s}^i)^t\ \overline{s}^{i-1} + (\overline{a}^i)^t\ \overline{a}^{i-1}}{2}) \tag{35}$$

This formulation allows to represent the orientation evolution with two fonctions :

- l'angle de rotation $\theta^{nom}(t)$,
- l'axe de rotation exprim par le vecteur $\overline{q}^{nom}(t) = [q_x^{nom}(t), q_y^{nom}(t), q_z^{nom}(t)]$.

Note that the quaternion formulation has the advantage of avoiding the calculation of the transition to the Rodrigues formula.

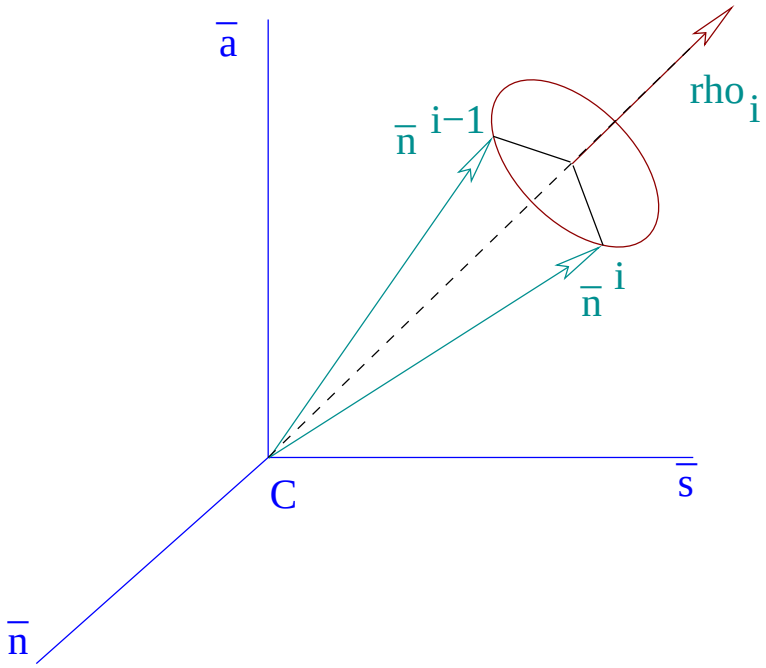

Fig. 12 Orientation change with axes $\overline{n}, \overline{s}, \overline{a}$

3.5 Task transition trajectories

The task is defined by a set of m functions $H = h_1, h_2, \ldots, h_m$. In the implementation of the trajectory simulator, at each extremity of the successive paths, positions and velocities are made to correspond with each other. The end point of path $i-1$ must be equal to the starting point of the path i. The path transitions also require speed equivalence. According to the strategy, we can also match the transition accelerations and jerks.

The transition trajectories are usually following some arcs as shown on Fig. 10.

3.6 Milling task preparation

The machining engineer draws a mechanical workpiece on his prefered CAD program and the result is a virtual solid. The CAM machining module is used to define cutting planes on the workpiece. The program proceeds by intersecting the cutting planes with the virtual solid to determine several parallel surfaces. The CAM program then fills the surface with cutting paths This results into a set of nominal

Cartesian trajectory functions that are saved in a nominal Cartesian trajectory file. certain CAM programs allow the user to access this file.

The CAM program further transforms the nominal Cartesian trajectory functions of the milling task into sets of points that are saved in a theoretical Cartesian trajectory file. This file can also be retrieved is refered to the Cartesian set-point file.

3.7 Initial digitization of milling trajectories

As input, a task definition file comprises a series of nominal functions; each function is of the form $\overrightarrow{X}^{nom}(t)_{|_{r_F}}$. The points of departure and arrival P_D and P_f are known for each function. Theoretical positions are thus calculated from these nominal symbolic functions: $\overrightarrow{X}^{th}(c)_{|_{r_F}} = \overrightarrow{X}^{nom}_{|_{r_F}}(t)$ at each T_c sampling cycle. Time T_c is assumed to remain constant throughout the process. The total time is therefore set to $t = c\ T_c$ for $c = 1,\ldots,s$. Firstly, a first time digitization occurs at the sampling rate, Fig. 13, which has the effect of transforming the paths in point series.

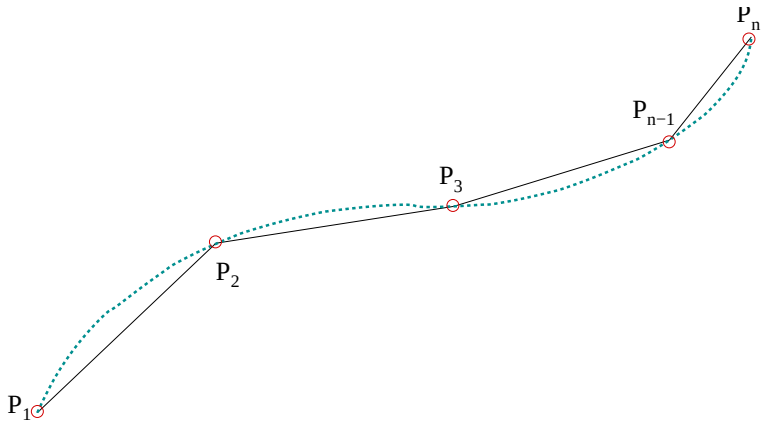

Fig. 13 Digitization of task Cartesian theoritical path

Finally, the CAM program considers that all theoritical points are connected by line segments in some kind of linear approximation, Fig. 13. Further point sampling is then performed by separating the points selected by a calculated distance in accordance with a chord error E_c, Fig. 14. Thus, as an arc is bent by a straight line rope, each pair of set points sees a line segment connecting them. This cord is at a maximum distance of E_c from the nominal trajectory. Let the arc be of radius R and lenght L, then $E_c = R - Rcos(\frac{L}{2R})$. In order to obtain a predermined E_c cord error, the arc point distance is calculated by: $L = arccos(1 - \frac{E_c}{R})$. Then, the cord distance is calculated by: $D = 2\sqrt{E_c^2 - 2R\,E_c}$. Knowing the constant feedrate and the cord

distance, the sampling time T_p is then calculated. Each new point will then add to the original theoritical path file. The resulting file is called the complete theoritical Cartesian path. The *CAM* program linearization is typically already introducing an error, so that the accuracy of the robot can never be better than this $E_c$cord error value.

Then, the IKP is calculated on each theoritical Cartesian path. For each pose point comprising the position and orientation, the actuator positions are calculated. The result will be written in an actuator theoritical set-point file which is then uploaded to the CNC controller.

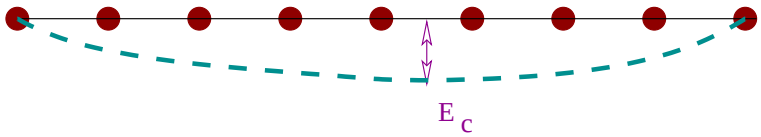

Fig. 14 Digitization of task theoritical section

3.8 Second digitization of milling trajectories

Running at a smaller cycle time, the six servo feedback loops traditionnaly implement a PID feedback loop on each linear axis position. During each T_p cycle time, the path following level interpollates a certain number points inside the interval determined by each point pair in the actuator theoritical set-point file. The number of points is determined by: $N = floor(\frac{T_p}{T_s})$ where T_s is the servo feedback loop cycle time determined by the time to calculate the PID algorithm. Actuator point sampling is then performed by utilizing a polynomial interpollation function.

Typically, in many CNC controllers, it is observed that the servo sampling rate (second level) can be ten times the cycle time of the first level.

4 Verification criteria for machining

4.1 Machining accuracy

The most important performance criterion is the machining surface finish. Since machining requires a trajectory following with high precision, we must ensure that the path is simulated within a given precision, (Mery 1997).

In classic robotics, the majority of path planning applications are classified as point-to-point and a marginal number are concerned by continuous paths such as

in machining. However, even when implementing continuous, the robot control algorithms handles points. The main difference with point-to-point control is that the task is defined by several hundreds of points instead of a few points. Liege and Coiffet define four types of precision : static accuracy, dynamic accuracy, repeatability and resolution, (Liegeois 1984) (Coiffet 1986). Repeatability stands for the reproduction accuracy of the same movement and does not really apply for continuous trajectory tasks. The resolution is the smallest amount of change in the positions and orientations. It is determined by robot component choices.

Definition 6. Static accuracy is defined as the ability of the robot to position and orient the end-mechanism in accordance with the programmed instructions.

This notion is applicable to a specific point and then cannot extrapolated to one entire continuous trajectory.

Definition 7. Dynamic accuracy is the ability of the robot to follow a path by the end-effector mechanism in accordance with the programmed path.

In principle, the error is calculated at all points along its theoritical path $X(kT)^{th}$ where $k = 1,\ldots,k_{max}$ where is the k_{max} number of discretized points. The error vector between the nominal path and the simulated path is then:

$$\overrightarrow{\varepsilon}(kT) = \overrightarrow{X(kT)}^{th} - \overrightarrow{X(kT)}^{nom} \tag{36}$$

The distance or error vector magnitude is also calculated:

$$\varepsilon(kT) = ||\overrightarrow{X(kT)}^{th} - \overrightarrow{X(kT)}^{nom}|| \tag{37}$$

After calculating the error vector or value of distance for a path, we determine the overall path accuracy for each error vector component and the error vector distance by choosing the largest value.

4.2 Error over the Cartesian position

4.2.1 Calculation of the absolute error and the error vector between the points

In practice, the end-effector precision calculation is divided into two task space parts : Cartesian position and Cartesian orientation. For the error in the Cartesian position, we obtain the equation is calculated for each theoretical Fig. (14) :

$$\varepsilon(kT) = ||\overrightarrow{X(kT)}^{sim} - \overrightarrow{X(kT)}^{th}|| \tag{38}$$

We also study the nature of the error vector.

$$\overrightarrow{\varepsilon(kT)} = \overrightarrow{X(kT)}^{sim} - \overrightarrow{X(kT)}^{th} \tag{39}$$

This calculation is also applicable on theoretical points of the *CAD* produced files.

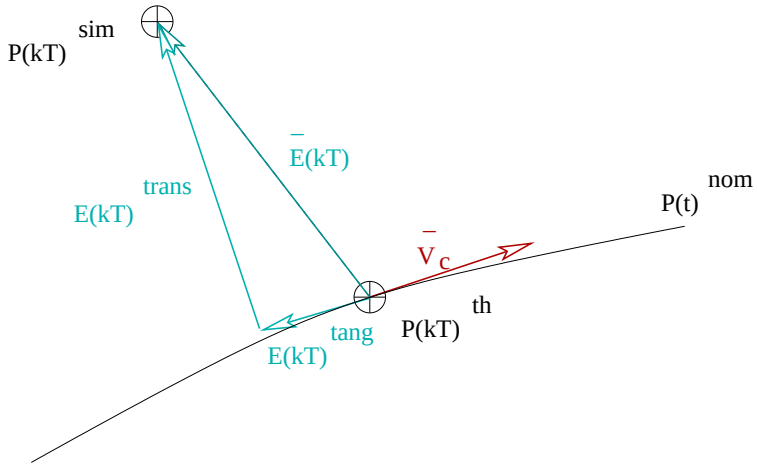

Fig. 15 Error vector, tangentielle error and transverse error

Since the error along the trajectory is not as significant as the transversal path error, we calculate the tangential error and transversal error, Fig. 15. The transverse error can also be called cross-sectional, normal or perpendicular error.

The tangential error allows us to evaluate if the simulated path is ahead or behind the nominal planned route. A tangential error indicating that the real path is followed ahead of time is of course advantageous because it means that the trajectory can be continued in a shorter time than expected. In fact, Liegeois states that a robot can be late in the path set without the finished surface being affected, (Liegeois 1984). A tangential error indicating that the real path is plagued by a slowdown may not necessarily affect the surface finish as such and therefore is not so considered important.

On the other hand, the transversal error will directly affect the surface finish. It corresponds to the difference between the simulated path and the nominal path at time $t = kT$ where $k = 1, /ldots, m$ with m the number of points. Then, we try to determine if the simulated path is located within a given path tube with a predefined radius. The tube radius is determined by machining tolerances.

To calculate the vector tangential error, we must determine the unit vector tangential to the nominal curve through the velocity vector :

$$\overrightarrow{u(t)} = \frac{\overrightarrow{V_c(t)}}{||\overrightarrow{V_c(t)}||} \tag{40}$$

The value of the tangential error is obtained by:

$$\varepsilon(kT)^{tang} = \overrightarrow{u(t)} \cdot \delta\overrightarrow{P(kT)} \tag{41}$$

Applying the Pythagorean theorem, we finally find the value of the transversal error :

$$\varepsilon(kT)^{trans} = [(\varepsilon(kT))^2 - (\varepsilon(kT)^{tang})^2]^{1/2} \tag{42}$$

The calculation of transversal error with respect to the nominal trajectory is not exact but an approximate value of the deviation sought because it is obtained from the digitized values and is not necessarily the perpendicular error defined as the minimal distance between the nominal and theoritical trajectories. It is necessary to nuance this comment. The perpendicular error may not be a direct measurement of surface finish. For example, during 3D milling, the robot is positioned so as to obtain the Z-axis of the terminal member perpendicular to the surface to be machined. Then, we seek to mill a planar surface that is positioned parallel to the XY plane and the finished surface will be evaluated by calculating $\varepsilon(kT)_z$. Upon reaching the portion of the part where a wall is reached, the wall perpendicular error will be determined.

4.3 Calculate the actual deviation from a nominal curve

To be meaningful, dynamic precision must be defined relative to the nominal path, (Liegeois 1984). On the Fig. 15, we note that $\overrightarrow{\varepsilon(kT)}^{trans}$ is not the actual deviation from the nominal curve. To achieve this, we must calculate the point $\tilde{P}$ being the closest to $\overrightarrow{P(kT)}^{sim}$ on the nominal curve. To do this , we determine the time t^{devi} which corresponds to the point $\tilde{P}$ on the nominal curve, Fig. 15, and two methods can be derived.

The first method consists in determining the normal to the nominal curve which is performed by solving the following system :

$$(\overrightarrow{P(kT)}^{sim} - \overrightarrow{P(t)}^{nom}) \cdot \overrightarrow{V_c(t)} = 0 \tag{43}$$

The second method consists in searching the minimum distance between $\overrightarrow{P(kT)}^{sim}$ and $\overrightarrow{P(t)}^{nom}$ by calculating the minimum of the function :

$$G(t) = ||\overrightarrow{P(kT)}^{sim} - \overrightarrow{P(t)}^{nom}|| \tag{44}$$

which corresponds to determining the time at which the derivative of the function is zero, that is to say when $G^{'}(t) = 0$.

Introducing t^{devi} time in the function, we obtain $\tilde{P}$ and then the deviation is calculated :

$$\varepsilon(kT) = ||\overrightarrow{P(kT)}^{sim} - \tilde{P}|| \tag{45}$$

Deviation value is determined by calculating the maximum deviation of an entire trajectory. The second approach for calculating the deviation t^{devi} has the advantage of being less complex in terms of calculations and therefore will be preferred.

4.4 Calculation of deflection from a straight line segment

When the nominal paths are straight lines, it is not necessary to perform the calculation of the deviation to approach presented in the previous section. Determining the deviation $\overrightarrow{P(kT)}$ directly by calculating the distance between the simulated $\overrightarrow{P(kT)}$ and the line defined by the starting point $\overrightarrow{P_1}$ and the arrival point $\overrightarrow{P_2}$ of the section :

$$\varepsilon(kT) = (||P_1\overrightarrow{P(kT)}||^2 - ||P_1\overrightarrow{P(kT)} * \frac{\overrightarrow{P_1P_2}}{||\overrightarrow{P_1P_2}||}||^2)^{1/2} \tag{46}$$

4.5 Calculation of the deviation from a theoretical curve

There are many cases where the nominal functions are not available and the curves are not necessarily straight lines. For example, as we have already explained, many *CAD* program produce files with an E_c cord error between selected points. Not knowing the curve profile between these points, the *CAM* module interpolates using a linear function, that is to say, we assume that the points are connected by line segments, being different from the exact shape having then an unknown curvature. The curvature was lost in the digitization process. The deviation calculation takes then equation (46). The question to be carefully addressed is the choice of the points P_1 and P_2. We wish to determine the theoretical interval being closer to $\overrightarrow{P_k^{sim}}$, the point simulated, Fig. 16. The comparison is limited to adjacent intervals : the $i-1$ segment before and the segment i after the point $\overrightarrow{P_k^{th}}$.

There are two possible methods for interval selection. The first method is selecting the interval by the scalar products respectively for the interval $i-1$ and i :

$$v_{k-1} = (\overrightarrow{P_{k-1}}^{th} - \overrightarrow{P_k}^{th}) \cdot \varepsilon\overrightarrow{P(kT)} \tag{47}$$

$$v_k = (\overrightarrow{P_{k+1}}^{th} - \overrightarrow{P_k}^{th}) \cdot \varepsilon\overrightarrow{P(kT)} \tag{48}$$

The closest interval will be identified by selecting the positive result between v_{i-1} and v_i.

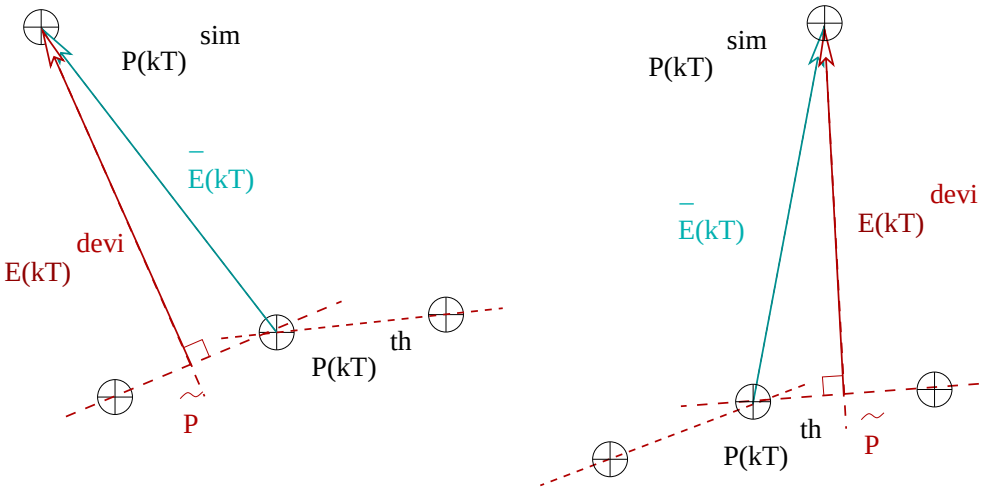

Fig. 16 Deviation vector from the theoretical points

The second method ivolves the calculation of the time corresponding to the point on each straight line segment :

$$t_{k-1} = T_p \frac{\overline{P_x}^{sim} + \overline{P_y}^{sim} + \overline{P_z}^{sim}}{||\overrightarrow{P_{k+1}}^{th} - \overrightarrow{P_k}^{th}||^2} \text{ where } \overline{P}^{sim} = \overrightarrow{P_k}^{sim} - P_{k-1}^{th} \tag{49}$$

$$t_k = T_p \frac{\overline{P_x}^{sim} + \overline{P_y}^{sim} + \overline{P_z}^{sim}}{||\overrightarrow{P_{k+1}}^{th} - \overrightarrow{P_k}^{th}||^2} \text{ where } \overline{P}^{sim} = \overrightarrow{P_k}^{sim} - P_k^{th} \tag{50}$$

The two times are then compared with the cycle time T_p and the closest interval from the point is the one confirming $0 \leq t \leq T_p$.

The second approach is less complex to implement and has been chosen.

The distance is determined by replacing P_1 and P_2 by the extrema of the chosen interval in equation (46). This distance is not equal to the actual deviation since each interpolation corresonds to the straight line segment between two points. It is necessary to take the deviation vector and add the vector related to the E_c error being perpendicular to the straight line segment and included in the plane defined by the velocity vector at point i and the vector aligned with straight line segment.

Note that if the theoretical path is a straight line, then we can calculate the deviation directly with equation (46).

4.6 Calculate the actual deviation from a theoretical curve with a small radius of curvature

In the case where the radius of curvature is high, this method is not guaranteed to calculate the minimum distance, since the theoretical $\overrightarrow{P_k}^{th}$ is not necessarily the closest to the simulated $\overrightarrow{P_k}^{sim}$ point. For example, such a situation is encountered when machining rectangles with corners with radii of curvature tending towards 0.

To remedy this problem, an added algorithm determines $\overrightarrow{P_n}^{th}$, the closest theoretical simulated point, by seeking the value of n such that $(||\overrightarrow{P_n}^{th} - \overrightarrow{P_k}^{sim}||)$ is minimized by varying n from $n-20$ to $n+20$. Indeed, it is not necessary to test all trajectory points. Then, the deviation is calculated with the aforementionned method.

4.7 Orientation errors

There would as many methods to calculate errors over the orientations as there exists representation models. We chose to determine the orientation error by calculating the variations on the normal vector because it is more ergonomic to visualize the movement of a vector that characterizes the parallel robot mobile platform.

$$\delta\overrightarrow{N_c(kT)} = \overrightarrow{N_c(kT)} - \overrightarrow{N(kT)}_c^{th} \tag{51}$$

In addition, *CAD* programs represent orientations by expressing the pointing vector colinear with the tool axis which, in the case of parallel robots, is commonly corresponding to the mobile platform normal vector.

4.8 Actuator joint errors

The simulator also compares the theoretical and simulated actuator joint trajectories, thereby obtaining the actuator error for the six actuators. For $i = 1,\ldots,6$, we calculate $\zeta_i^{sim} = L_i^{sim} - L_i^{th}$.

4.9 Error Models

In order to simulate a realistic trajectory pursuit, error models are introduced at different levels of the simulator. The majority of errors are introduced by adding

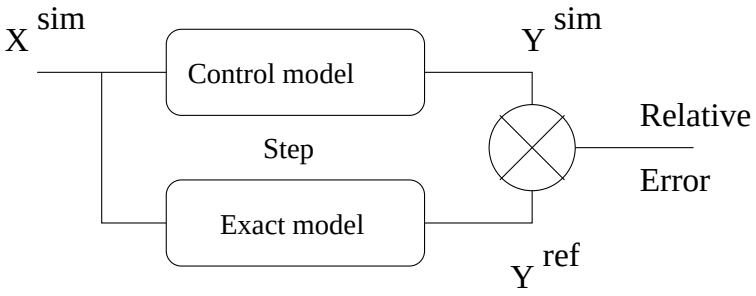

Fig. 17 Calculation block diagram of the end-effector relative error for a step

a parameter to a function determined by randomly selecting a value in a specific interval $[-max, +max]$.

We have chosen to the modeling of all the following errors:

- *CAD* file precision,
- sensor accuracy, $\delta\ l_i$,
- configuration precision, δOA_i and δCB_i,
- precision on the calculation of the **FKP**,

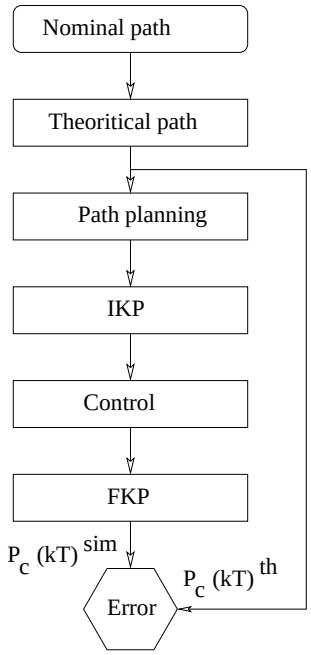

Fig. 18 Calculation flow chart of the end-effector absolute error

- resolution of time measurement Δt and temporal digitization,
- the resolution of signal digitization,
- the asynchronous nature of joint signal updates.

We can thus simulate a trajectory introducing all errors, any combination of these or even only one. The simulation can be tailored to the actual study and it is possible to isolate errors and investigate their impact on surface finish. We propose two alternative calculation errors :

- the relative error between two steps, Fig. 17,
- the absolute error giving the end-effector accuracy Fig. 18.

5 Results of path simulation

5.1 Operative part of the trajectory simulator

A simulator is proposed where a typical machining task is performed by a general **6-6** hexapod manipulator. Knowing that path planning and following is essentialy a kinematics problem, the principle of the kinematics simulation is based on the alternate use of **IKP** and **FKP**.

The robot operative part consists of the following : the mechanism described by the actual geometry (passive joints), sensors (instrumented joints), motors (actuated joints) and amplifiers, (Mery 1997). The following models were incorporated : the calibrated values of the configuration and the position sensors. Both are characterized by a certain measurement accuracy. At the simulation beginning, we determine the actual real configuration values by adding a random contribution to the ideal configuration values $\mathbf{OA}_{|r_F}$ and $\mathbf{CB}_{|R_m}$. Then, at each point of the theoretical Cartesian path, we calculate the theoretical joint values using the **IKP** using the actual real values of the configuration. For each calculated $\mathbb{L}_{i_k}$ (for $i = 1, \ldots, K$) where K is the number of actuator set-points. Realistic joint values are calculated by adding random contributions δL_{i_k} on the six actuators (for $i = 1, \ldots, 6$). We will continue with the **FKP** calculation for each trajectory point. The simulation ends with the surface finish evaluations.

The simulation does not address the impact of amplifiers and motors. It implements a nominal trajectory (continuous) or theoretical (discrete) as the set-points and the real simulated trajectory (computed on points).

Consider that the random error on the measurements of the passive joint positions are determined at the beginning of the machining process and remains constant throughout the simulation. This assumption means that we neglect the configuration change values caused by mechanical wear. The position sensor random error is recalculated each time the measure is used by the control unit, meaning at each T_s sampling period.

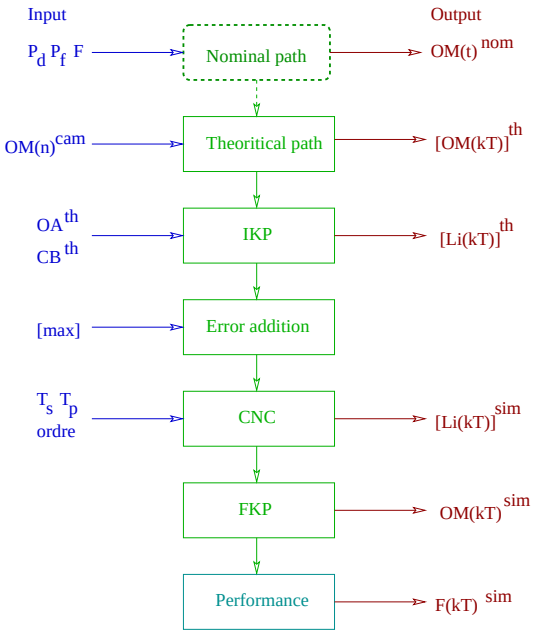

Fig. 19 Simulator Diagram of the general hexapod robot with control

5.2 Trajectory simulator including the controller

The operative part could be studied by itself in order to identify the configuration error impacts. After studying the robot operative part during a machining task, a complete analysis of the parallel robot path planning is proposed incorporating a model of the **CNC**. The controller performance is then investigated in terms of surface finish, (Marty, Cassagnes and Martin 1993). The simulator then implements a conventional joint control strategy as usually found in the majority of **CNC**. Each servo can only ensure the setting of one single actuated joint variable. In the majority of current controllers, this variable is either the position or the speed. CNC Speed control analysis will be left for another article.

5.2.1 Trajectory simulation for position control

The position control simulator implements a joint strategy of trajectory planning and following for indirectly controlling the end-effector position through the calculation of the **FKP**. The principle of the simulation remains basically the same kinematics one as before, meaning that it is based on the alternate use of **IKP** and **FKP**, Fig. 19. Between the two calculations of the geometric patterns of the simulated and theoritical trajectory, in addition to adding the various error contributions, a module

that models the command is integrated and it calculates the servo set-points using various polynomial interpolation functions. The surface finish evaluation proceeds applying the same calculations as before.

5.2.2 Interpolations of actuator set-points

The **CNC** is equipped with polynomial interpolators which calculates several intermediate points between the theoretical set-points. These intermediate point calculation actually determines the higher level control cycle time, T_p. New interpolated points are calculated at all times of T_s . It then follows that the polynomial joint movements give a Cartesian movement which polynomial is assumed to approach the scheduled task for the nominal machining movement . Note that the movement will never be obtained equivalent to nominal displacement .

Interpolation modules are used on a segment defined by two points of joint instructions $\overrightarrow{l_i}^{th}(k\ T_p)$ and $\overrightarrow{l_i}^{th}((k+1)\ T_p)$ upon which typical polynomial curve fitting is performed. The time interval is the time for calculating the interpolation order, or T_p. They use polynomial functions whose parameters are measured by matching certain amounts (positions, velocities, accelerations, jerks, etc.) depending on the polynomial order. The following interpolations can be simulated :

- *first order* : passing a straight line between each point , then the linear function $f(l_i) = c_1\ l_i + c_0$, we calculate coeffients c_0 and c_1 with the positions at the beginning and end of the interval $l_i(\overrightarrow{k\ T_p})^{th}$ and $l_i(\overrightarrow{[k+1]T_p})^{th}$, this technique is widely used in usual **CNC** machine-tools;
- *second order* : it calculates the parameters of the quadratic function $f(l_i) = c_2\ l_i^2 + c_1\ l_i + c_0$ from two positions and speed;
- *second and first order* : it divides the section into two parts : an acceleration phase and a phase constant speed can choose two methods with calculated acceleration or acceleration set (often the maximum acceleration of the axis);
- *third order* : it softens the movement using a cubic function $f(l_i) = c_3\ l_i^3 + c_2\ l_i^2 + c_1\ l_i + c_0$ which coefficients are calculated from the known positions and velocities at the beginning and end of the interval values;
- *fifth order*: one takes the two positions, two gears and two known start and end accelerations interval to calculate six parameters of a fifth degree polynomial function.

5.2.3 Algorithm to simulate the control position

The former algorithm now includes one typical **CNC** containing the control levels with each its cycle time :

- The first level for path planning and following,
- The second level comes from the six motor servo controllers.

The simulator requires the same inputs and produces the same outputs has shown above.

SimulatorCNC Configuration of robot : $\mathbf{OA}_{|_{R_f}}, \mathbf{CB}_{|_{R_m}}$

Configuration of commande : T_p, T_s

\# of control intervals : $n_p = round(\frac{\delta t}{T_p})$

\# of servo intervals : $n_s = round(\frac{T_p}{T_s})$

Interpolation order : $Or = 5$

Configuration of the nominal path : P_d, P_f and F_r

Nominal path : $X(t) = \overrightarrow{X(t)}^{nom}$

Path duration : $\delta t = \frac{\delta s}{F_r}$

Cartesian nominal derivatives $V_c(t)^{nom} = \frac{d}{dt} X(t), A_c(t)^{nom} = \dot{V_c(t)}^{nom}$

Actuator nominal functions $L(t)^{nom} = MGI(OM(t)^{nom})$

Actuator nominal derivatives $vL_i(t)^{nom} = \frac{d}{dt} L(t)^{nom}$ and $aL_i(t)^{nom} = \dot{vL_i(t)}^{nom}$

For $k = 0 \rightarrow n_p$ do

  For $u = 0 \rightarrow n_s$ do

    $t_w = k\, T_p + u\, T_s$

    $OM_w^{th} = OM^{nom}(t_w)$

    $Li_w^{th} = interpolle(Or, L_i^{nom}(t_w), T_p, T_s, t_w)$

  $\mathbf{OA}_{|_{R_f}} = \mathbf{OA}_{|_{R_f}} + Alea(OA_{max})$

  $\mathbf{CB}_{|_{R_m}} = \mathbf{CB}_{|_{R_m}} + Alea(CB_{max})$

For $k = 0 \rightarrow n_p$ do

  For $u = 0 \rightarrow n_s$ do

    $t_w = k\, T_p + u\, T_s$

    If $k = 0$ and $u = 1$ do

      For $i = 1 \rightarrow 6$ do

        $Li_k^{sim} = Li_k^{th} + Alea(Li_{max})$

      $sys = \mathbf{modeleFA7ent}(\mathbf{OA}_{|_{R_f}}, \mathbf{CB}_{|_{R_m}}, Li_k^{sim})$

      $res = \mathbf{MGDexact}(sys)$

      For $j = 1 \rightarrow \#(res)$

      $solc_j = CalculCentre(res)$

      $OM_k^{sim} = Closer(solc, P_d)$

    else

      For $i = 1 \rightarrow 6$ do

        $Li_k^{sim} = Li_k^{th} + Alea(Li_{max})$

      $sys = \mathbf{modeleFA7rat}(\mathbf{OA}_{|_{R_f}}, \mathbf{CB}_{|_{R_m}}, Li_k^{sim})$

      $res = \mathbf{MGDhybride}(sys, OM_{k-1}^{sim})$

      $OM_k^{sim} = CalculCentre(res)$

    Calculation of precision : $\overrightarrow{\varepsilon P_k}^{sim}, \varepsilon_k^{sim}$

    Calculation of deviation : ε_k^{sim}

    Calculation of actuator precision : ζ_k^{sim}

return $OM^{sim}, Li^{sim}, \varepsilon^{sim}, varepsilon^{sim}, \zeta^{sim}$

The simulator uses the following functions :

- **modeleFA7ent** $(\mathbf{OA}_{|_{r_F}}, \mathbf{CB}_{|_{R_m}}, Li_k^{sim})$: computing system of **FKP** as the mobile platform three points with integer coefficients,

- **modeleFA7rat** $(\mathbf{OA}_{|r_F}, \mathbf{CB}_{|R_m}, Li_k^{sim})$: computing system of **FKP** according to the mobile platform three points with rational coefficients,
- $Alea(val_{max})$ a function that computes a random value taken between 0 and val_{max},
- **PlusPres** $(solc, P_D)$ function that computes the closest parameter in the list $solc$ to point P_D
- **CalcCentre** $(\{\overrightarrow{X}\})$ function that computes the position of the center $OA_{|r_F}$ from each parameter in the list $(\overrightarrow{X})$
- **Interpolle** function that calculates the interpolated set-points. This function receives the order of the polynomial function selected by the user.

6 Results of path simulation

In this section, as part of the path planning related to milling and by extension to all high accuracy applications, kinematics simulation results calculates end-effector surface finish impact integrating configuration inaccuracies and postion based CNC control strategies. The results are compiled, presented, analyzed and compared.

Firstly, the error impact of the joint positions is studied. The actuator lenghts are subject to an error and their introduction will be studied. Then, the passive joints correspond to the kinematics chain attachment points A_i and B_i for $i = 1, \ldots, 6$. The section continues providing an analysis of trajectory simulation with classic CNC position control which will be performed on **6-6** parallel robots for the first time. Various parameters will be varied and the results will allow to determine how to improve finish surface.

6.1 Parallel robot configuration

Let us take a typical **6-6** configuration example written in a configuration text file which includes the manipulator essential parameters: the coordinates of the joint center positions $OA_{|R_f}$ in the fixed base reference frame R_f and the coordinates of the joint center positions $CB_{|R_m}$ in the mobile platform reference frame R_m. In the computations, we use their simplified format, respectively identified as A and B. Here is a typical configuration example with a real manipulator configuration. The unit is the millimeter. The values were the ones calculated by a calibration procedure:

We try one difficult FKP example on a typical 6-6 hexapod with 40 complex solutions out of which 16 real solutions can be extracted. Its configuration is given on Table 6.1 where the fixed base and mobile platform joint coordinates, OA_O, CB_C are given with units being the millimeter.

Table 1 Parallel manipulator configuration table

Joint Coordinates	Respective Values
$OA_1(x)$ $OA_1(y)$ $OA_1(z)$	464.141 389.512 -178.804
$OA_2(x)$ $OA_2(y)$ $OA_2(z)$	569.471 207.131 -178.791
$OA_3(x)$ $OA_3(y)$ $OA_3(z)$	529.050 -597.151 -178.741
$CB_1(x)$ $CB_1(y)$ $CB_1(z)$	68.410 393.588 236.459
$CB_2(x)$ $CB_2(y)$ $CB_2(z)$	375.094 -137.623 236.456
$CB_3(x)$ $CB_3(y)$ $CB_3(z)$	306.664 -256.012 236.461

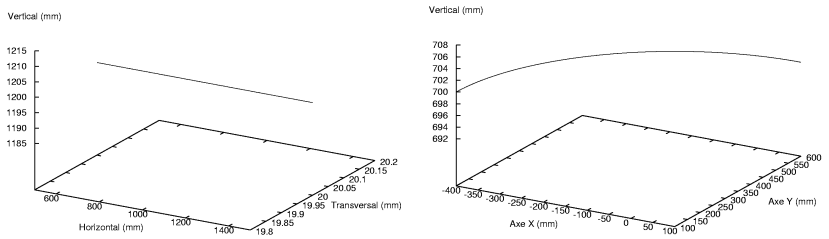

Fig. 20 Selected nominal paths

6.2 Typical trajectory and realistic milling configuration

We have implemented various control strategies in position by interpolating points by polynomial functions of the first degree, third degree and fifth degree. We also tested a preceded by an acceleration phase linear interpolation. The latter has been interpolated with a quadratic function for setting acceleration. This type of interpolation requires smoothing functions of first and second order. We must determine four boundary conditions of the interval and two alternatives were studied : the two positions and joint velocities at the ends and three end conditions by setting the acceleration phase of the second order. In the first case, one has to calculate the acceleration as a function of the end conditions.

We chose two nominal paths located on planes parallel to the XY plane. These path nominal functions are respectively determined by the following configurations, Fig. 20 :

- a line segment starting at point $[500,20,1200]$ and ending at point $[1500,20,1200]$ traveled at three constant feed forward speeds : 30, 45 and 60 m/min.
- an arc of radius 500 mm from the point $[100,600,700]$ to reach point $[-400,100,700]$ using the same three feed rates. The center of the arc is point $[100,100,700]$.

Note that the selected tasks are simulated tyring to reproduce realistic milling conditions. We will study the trajectories at different feed rates which are set to 30, 45 and 60 m/min. The feedrates of 30 and 60 m/min speeds correspond to the

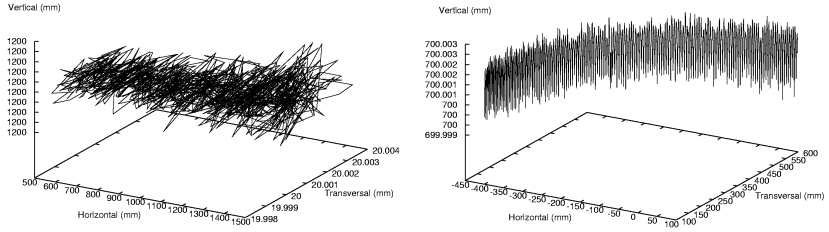

Fig. 21 Simulated path pursuits: straight line segment and arc

speeds of high speed milling *HSM* and ultra high speed milling *UHSM* respectively. A study is also conducted on the impact of path following cycle times which will be set at 5, 10 and 20 ms.

The simulator computes and sketches the two resulting paths utilizing a controller with a cycle time of $10ms$ and a feed rate of $30m/min$, Fig. 21 where the one dimension is exagerated to visualize the path errors. There is a complex high-frequency noise on every simulated patterns which highlight sudden and unpredictable changes in the continued trajectory.

Firstly, we compute the error vector of the Cartesian position. The error vector norm determines the accuracy of the robot at each control point and gives an idea of the behavior of the robot itself. Since we are in the specific case of continuous trajectory, the deviation of the path relative to the nominal path is calculated. In these two cases of calculation of performance , we retain outliers . Finally evaluate the surface finish , we study the vertical component of the error vector . In order to simplify our study , there is provided the machining surfaces of which are parallel to the XY plane . It also retains the maximum and minimum values and the difference therebetween reflect surface finish .

The simulations are performed with actuator set-point interpolations of the first (linear), third and fifth order.

6.3 Control with linear interpolation

6.3.1 Straight line segment with linear interpolation

The first tests with the simulator implements the first level control proceeding with actuator joint set-point interpollation utilizing a linear interpollation. In the first analysis, we calculate the deviation of a typical path segment simulated over a nominal path. We therefore study the straight line segment path by first varying the cycle time of the order and the results are shown in Fig. 26.

At 5 ms, the deviation is a high frequency signal oscillating around a straight line function $f(t) = t/4+1$ in microns. The amplitude increases and has peaks reaching

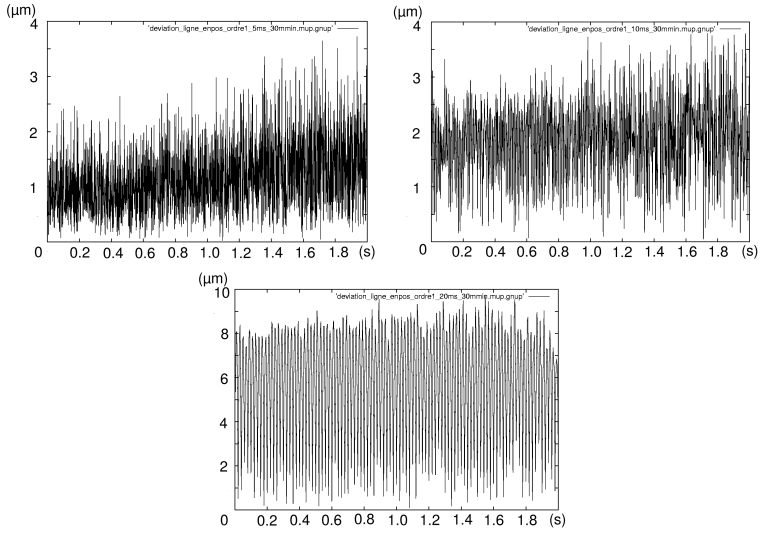

Fig. 22 Simulated path deviation for a straight line segment : cycle times of 5, 10 et 20 ms, linear joint interpolation

3.6 microns. At 10 ms, the signal oscillates around a constant straight line at 1.8 micron and oscillations thend to increase very slowly. At 20 ms, the average rose to 4.75 micron and the extrema of the oscillations are 1 and 8.5 microns with peaks at 9.5 microns.

In the second analysis, we study the same trajectory by now varying the feedrates and the results are shown in Fig. 23.

At feedrates of 30 m/min, the average is near 2 microns and the high frequency oscillations feature peaks from 0 to 3.5 microns. At 45 m/min, a similar signal is obtained where the average rises to 3.5 microns and peaks reach 6 microns. A 60 m/min, the oscillation average reaches 5 microns with 10 microns peaks.

We continue the analysis by showing graphs of vertical errors that are perpendicular to the machined surface errors since they provide with an excellent account of surface finish. The first graph shows the results at the selected feedrates, Fig. 24.

At feedrates of 30 m/min, the signal shows a high frequency oscillation with an average of approximately 1,25 microns with peaks as low as -1 microns and as high as 3 microns. At 45 m/min, there is an oscillation between -1 and 5 microns with an average of just over 2,5 microns. A 60 m/min, it is observed that the oscillation evolves mainly around 4.7 micron between -1 and 8.5 microns with some peaks at 10 and -1.5 microns.

We close this simulation cycle with vertical errors at the selected cycle times, Fig. 29.

At 5 ms, the signal is oscillating at a high frequency of around 0.3 micron. The amplitude of oscillation increases significantly. Peaks reached 2.4 and -1.6 micron

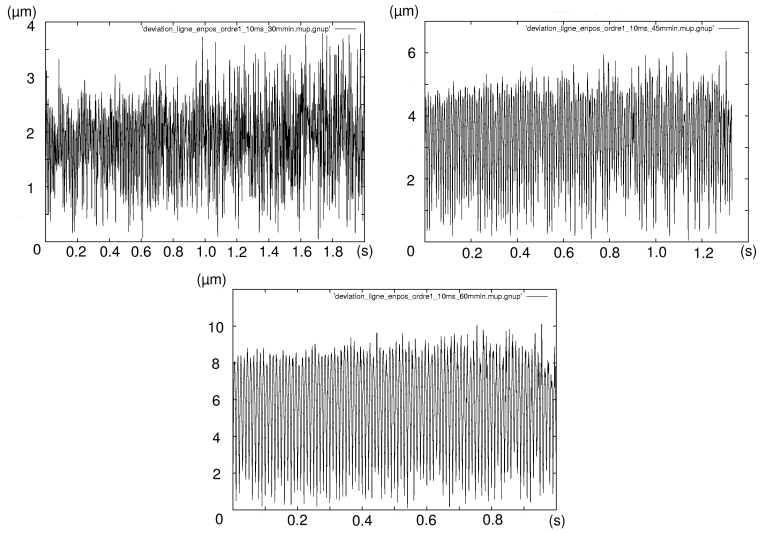

Fig. 23 Simulated path deviation for a straight line segment : feedrates of 30, 45 et 60 m/min, linear joint interpolation

causing surface finish error to become 4 microns. At 10 ms, the signal oscillates around 1.6 micron with an amplitude increasing less rapidly where extremas of 3.75

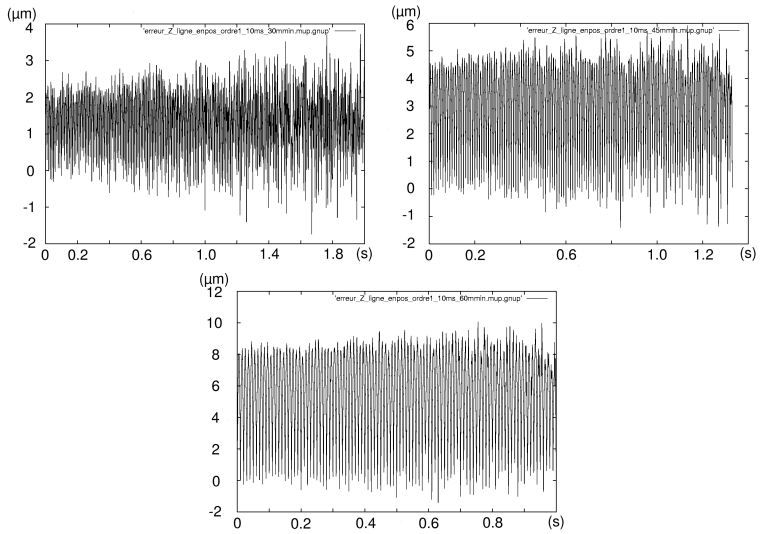

Fig. 24 Simulated vertical error for a straight line segment : feedrates of 30, 45 et 60 m/min, linear joint interpolation

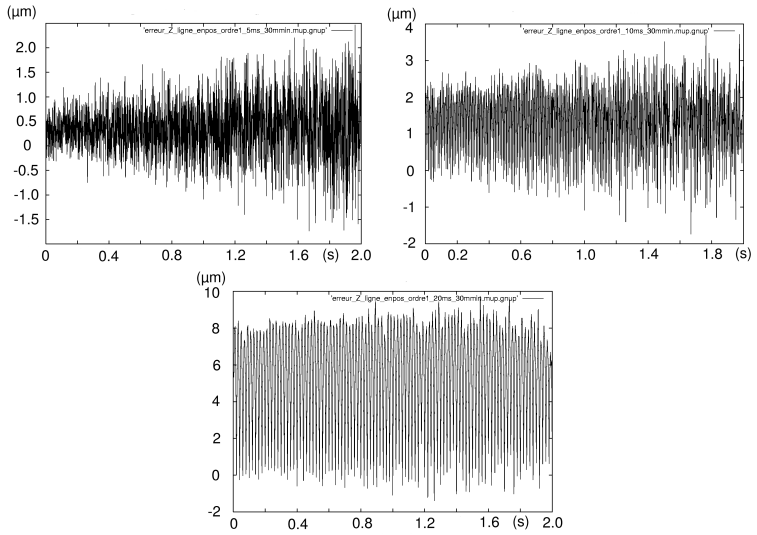

Fig. 25 Simulated vertical error for a straight line segment : cycle times of 5, 10 et 20 ms, linear joint interpolation

and -1.6 micron are extracted giving surface finish variations of 5.35 micron. At 20 ms, the oscillation is constant between 8.5 and -0.5 with an average of 4.2 microns. Peaks reach -1.4 and 9.5 microns leading to vertical variations of almost 11 microns.

Simulation results are collected in Table 2. On the table, the order of interpolation functions, the T_p cycle time in ms, the F_r feed rate in m/min, then the minimum and maximum extremas for the ε vector error magnitude in microns, the ε_Z vertical error in microns and $||\delta||$ deviation in microns.

Ordre	T_p ms	F_r m/min	ε^{max} micron	ε^{min} micron	ε_Z^{max} micron	ε_Z^{min} micron	$\|\|\delta\|\|^{max}$ micron	$\|\|\delta\|\|^{min}$ micron
1	10	30	4.900	0.142	3.716	-1.738	3.796	0.047
1	10	45	7.198	0.219	5.918	-1.403	6.055	0.047
1	10	60	11.872	0.026	10.067	-1.403	10.106	0.096
1	5	30	3.783	0.142	2.470	-1.738	3.726	0.021
1	10	30	4.900	0.142	3.716	-1.738	3.796	0.047
1	20	30	11.487	0.258	9.734	-1.403	9.747	0.096

Table 2 Simulated errors and deviations for a straight line segment : position control with linear joint interpolation

As might be suspected by intuition, we get better results by reducing the path controller (first level) cycle time and also the feedforward velocity. At very high speeds or with long cycle times, we met and exceeded the threshold of 10 microns.

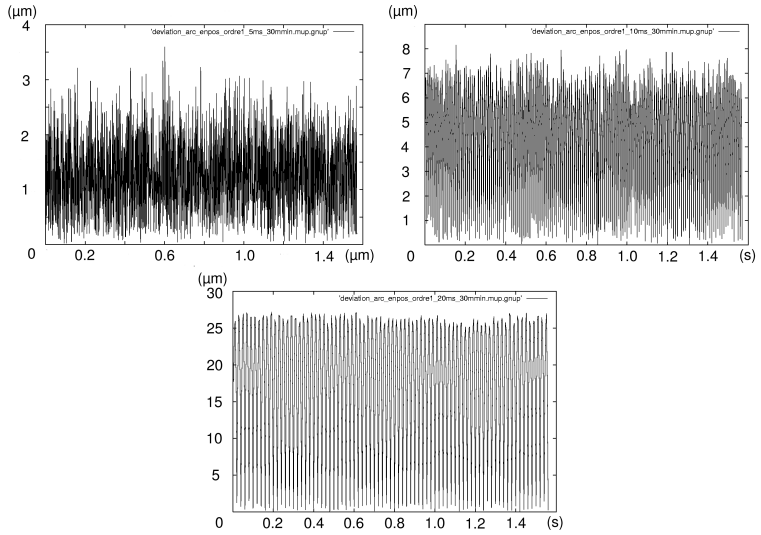

Fig. 26 Simulated path deviation for an arc : cycle times of 5, 10 et 20 ms, linear joint interpolation

At feedrates below or equal to 30 m/min and cycle times equal or less than 10 ms, the kinematics surface finish or the best feasible surface finish would reach 5 microns.

6.3.2 Arc with linear joint interpolation

In the second analysis, the same simulation process is repeated for a typical arc path y first varying the cycle time of the order and the results are shown in Fig. 26.

On Fig. 26, the signals are high frequency oscillations around a constant value. At 5 ms, the signal oscillates around an average of 1.5 micron with peaks evolving from 0 to 3 microns. At 10 ms, the signal oscillates around the value of 4 microns between extremas of 0.5 and 7.5 microns with peaks near 0 and 8 microns. Increasing to 20 ms, the average increases to 14 microns. The signal resembles a very regular high frequency sinusoidal curve ranging from near 0 to 27 micron.

We continue the study using the same arc path and varying the feedrate, Fig. (27).

The feedrate change from 30 m/min speed to 45 m/min doubles the signal average and its oscillation amplitude (from [0, 8] to [0, 16]). Similarily, The feedrate change from 30 m/min speed to 60 m/min triples the signal average and its oscillation amplitude (from [0, 8] to [0, 27]). In the later, the signal average is 15 microns.

In the second analysis, we then continue the arc path analysis by plotting vertical errors at the usual different feed-rates and the results are shown in Fig. 24.

The feedrate change from 30 m/min speed to 45 m/min doubles the signal average and its oscillation amplitude (from [-0.5, 3] to [-0.5, 6]). Similarily, The feedrate change from 30 m/min speed to 60 m/min triples the signal average and its oscilla-

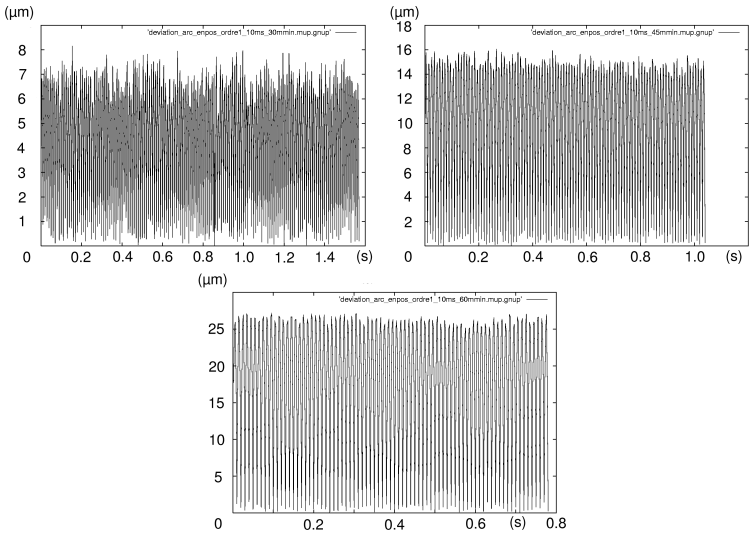

Fig. 27 Simulated path deviation for an arc : feed-rates of 30, 45 et 60 m/min, linear joint interpolation

tion amplitude (from [-0.5, 3] to [-0.5, 10.5]). In the later, the signal average nears 5.5 microns.

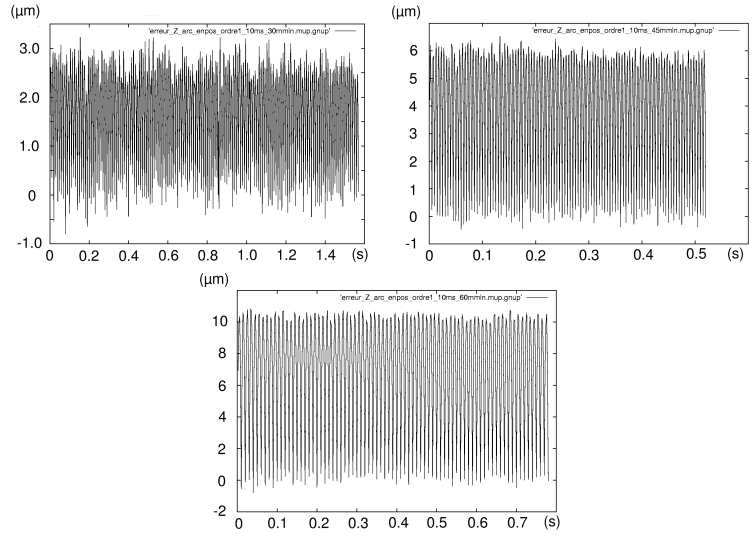

Fig. 28 Simulated vertical error for an arc : feed-rates of 30, 45 et 60 m/min, linear joint interpolation

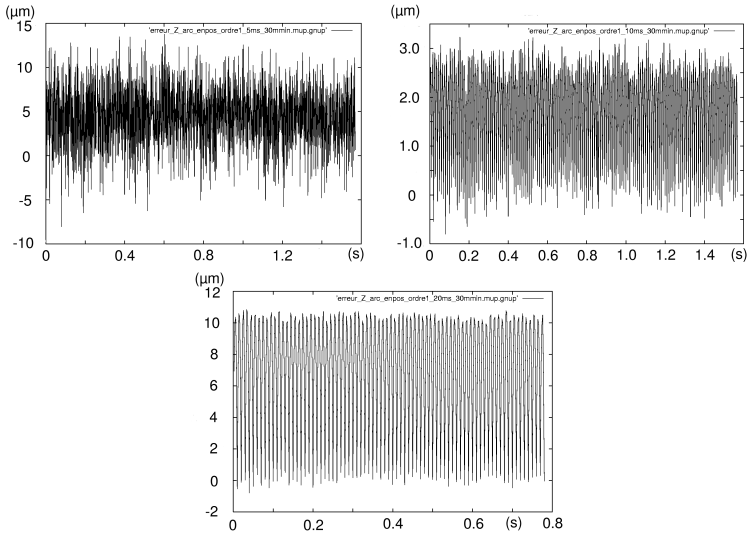

Fig. 29 Simulated vertical error for an arc : cycle times of 5, 10 et 20 ms, linear joint interpolation

Ordre	T_p ms	F_r m/min	ε^{max} micron	ε^{min} micron	ε_Z^{max} micron	ε_Z^{min} micron	$\|\|\delta\|\|^{max}$ micron	$\|\|\delta\|\|^{min}$ micron
1	10	30	8.164	$3e^{-5}$	3.2325	-0.806	8.157	0.016
1	10	45	16.128	0.023	6.533	-0.481	6.533	-0.481
1	10	60	27.178	0.129	10.803	-0.806	27.093	0.043
1	5	30	3.602	0.023	1.378	-0.806	3.598	0.023
1	10	30	8.164	$3e^{-5}$	3.2325	-0.806	8.157	0.016
1	20	30	27.178	0.129	10.803	-0.806	27.093	0.037

Table 3 Simulated errors and deviations for an arc : position control with linear joint interpolation

To end this simulation cycle, vertical errors are computed at the selected cycle times, Fig. 29.

The oscillating signals are similar to the high frequency previous ones. At 5 ms, the oscillation ranges from -0.25 and 1.25 microns with an average at around 0.5 micron. At 10 ms, the oscillation extremes reach -0.2 and 2.8 microns with an average at 1,5 micron. At 20 ms, the signal is a high frequency composite oscillation with extremas at 0 and 10.5 microns and peaks at -1 and 11 microns.

Table 3 compiles the results of kinematics simulations for the arc path tests.

The results confirm the former results obtained with straight line segments. The simulator can provide surface finish of 10 microns, only in the case of high speed milling ($< 30m/min$).

6.3.3 Discussion on the linear joint interpolation

From the kinematics analysis simulation, providing the lowest performance bounds, the CNC robot controller with linearily approximated trajectories can reach the surface finish if the feed-rate and path following cycle time are set properly. The controller can provide surface finish of 10 microns, only in cases up to high speed milling ($< 30m/min$). The surface finish is not met at faster feedrates. Linear interpollators should keep control cycle times relatively short ($< 10ms$) in order to reach the required surface finish. The surface finish is not met at longer cycle times.

To achieve an accuracy of less than 10 microns, one should set the response time at 10 msec or less and maintain the feed-rates below 45 m/min. This also means that **UHSM** is not feasible.

Those linear displacements are performed by a robotic system which is not linear. The interpolators try to transform curved path segments into linear path segments leading to interrupted segments. The linear interpollation is only matching the positions at the ends of the intervals, Fig. 13. As an advantage, the linear interpollation algorithm implementation is easy and does not require difficult computations leading to smaller cycle times. As disadvantage, with the application of parallel robots, the control system will not be able to reach 10 micron surface finish without very fast controllers featuring small cycle times. Moreover, rapid feedrates are not practical.

Let us add that the Cartesian velocity vector undergoes abrupt changes when passing from one linear segment to another which will result in dynamics overshoot. In fact, the continuation of this type of movement by an effective robot is impossible without stopping at each interval change which would mean slowing down the milling process.

This type of interpolation is only recommended for roughing milling.

6.4 Control with third order interpolation

6.4.1 Straight line segment with third order interpolation

The second simulation test series implement the first level control proceeding with actuator joint set-point interpollation utilizing a third order polynomial interpollation. In order to ensure the continuity of movement, it is then found to match the positions and joint velocities at the ends of intervals. As it was done for linear interpollation, tests begin with an analysis of deviation with the different selected feed-rates, Fig. 30.

The three signals are featuring growing high frequency oscillations until the trajectory ends. At 30 m/min, the signal oscillates around a straight line described by equation $f(t) = 0.375t + 0.75$ and it peaks at 3.75 microns. At 45 m/min, the curve deviation appears more advantageous since it oscillates about the same straight line axis as with 30 m/min and the signal peaks do reach just under 3 microns. At

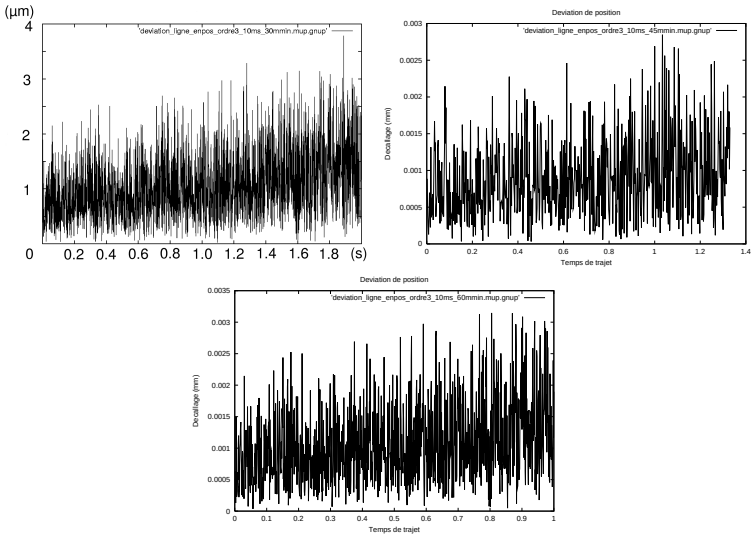

Fig. 30 Simulated path deviation for a straight line segment : feedrates of 30, 45 et 60 m/min, cubic joint interpolation

60 m/min, the same conclusions can be deduced and the peaks exceed 3 microns slightly.

The test are repeated by varying the cycle time of the CNC controller. The results are shown in Fig. (31).

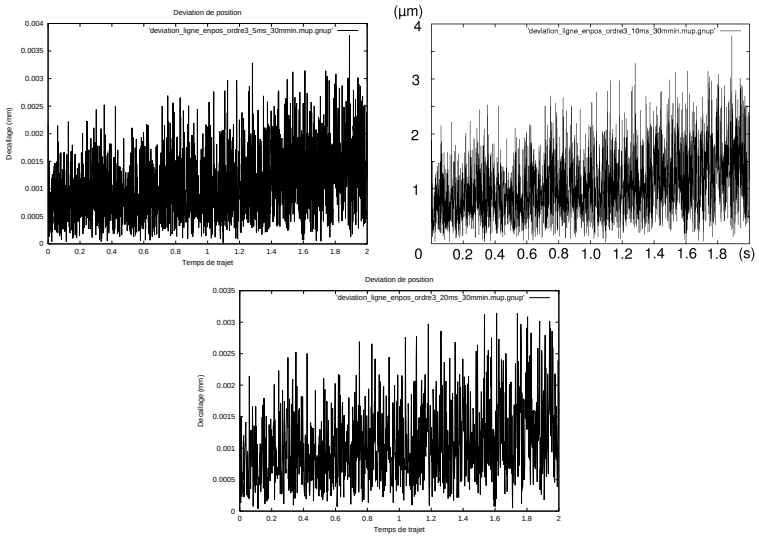

Fig. 31 Simulated path deviation for a straight line segment : cycle times of 5, 10 et 20 ms, cubic joint interpollation

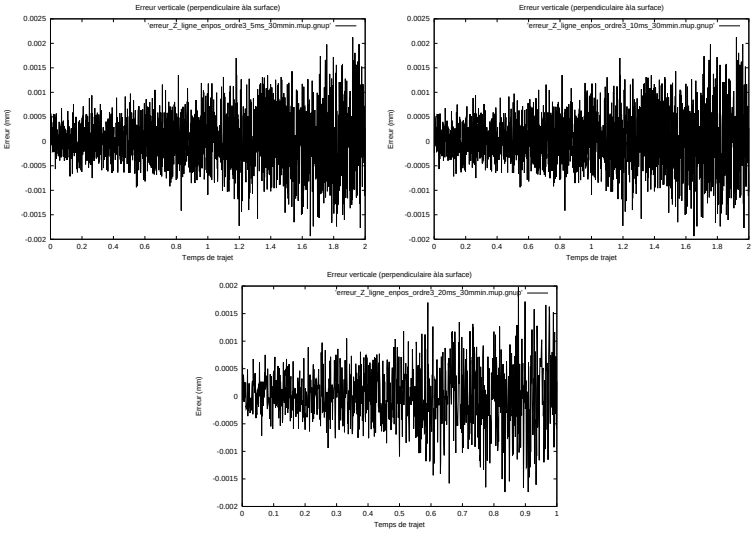

Fig. 32 Simulated vertical error for a straight line segment : cycle times of 5, 10 et 20 ms, cubic joint interpolation

The three signals are actually very similar and are featuring growing high frequency oscillations until the trajectory ends. The signals oscillate around a line described by equation $f(t) = 0.375 + 0.75t$. The peaks reach 3.75 microns. Note that the feed-rate does not seem to impact deviation significantly. The difference between the error vector $||\varepsilon||$ and deviation $||\delta||$ is less than one micron. This result means that the third order interpolation allows accurate theoretical trajectory following. This also means that the path following will take place without undue delay or advance.

We continue the analysis by showing graphs of vertical errors where the cycle times are varied, Fig. 37.

The three signals are actually very similar and are featuring growing high frequency oscillations until the trajectory ends. The graphs show signals which are centered on 0.2 micron with increasing oscillation with peaks getting close to 2 and -2 microns.

The vertical error is simulated at various feed-rates, Fig. 33.

The three signals are actually very similar and are featuring growing high frequency oscillations until the trajectory ends. The signals are around the constant value 0.1 micron with peaks from -1.8 to 2 micronz. At 45 m/min, the peaks are ± 1.4 micron.

Table 4 shows a compilation of results.

All error and deviation values remain below or equal to 4 microns.

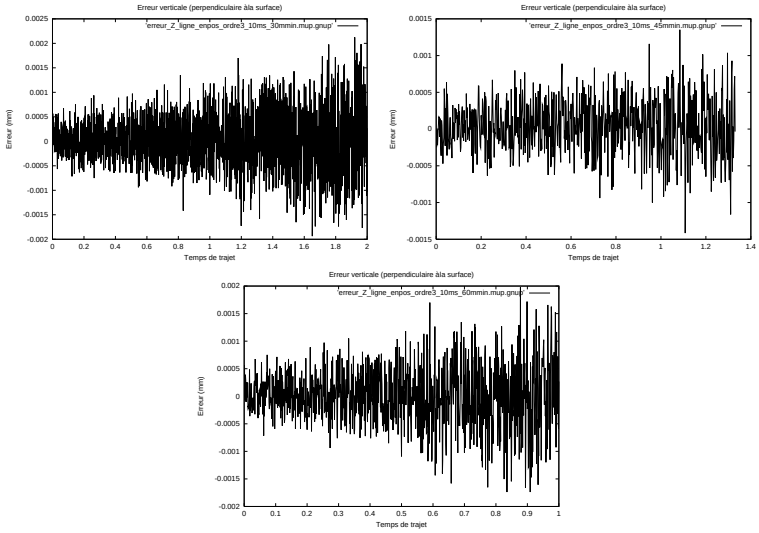

Fig. 33 Simulated vertical error for a straight line segment : feedrates of 30, 45 et 60 m/min, cubic joint interpolation

Ordre	Trajet ms	F_r m/min	ε^{max} micron	ε^{min} micron	ε_Z^{max} micron	ε_Z^{min} micron	$\|\|\delta\|\|^{max}$ micron	$\|\|\delta\|\|^{min}$ micron
3	10	30	3.787	0.069	2.124	-1.936	3.786	0.021
3	10	45	4.053	0.069	2.351	-2.059	3.285	0.032
3	10	60	3.692	0.069	1.978	-1.738	3.146	0.035
3	5	30	3.787	0.069	2.124	-1.936	3.786	0.021
3	10	30	3.787	0.069	2.124	-1.936	3.786	0.021
3	20	30	3.692	0.069	1.978	-1.738	3.146	0.035

Table 4 Simulated errors and deviations for a straight line segment : position control with cubic joint interpolation

6.4.2 Arc with third order interpolation

The same simulation process is repeated for a typical arc path by first varying the feed-rate and the results are shown on Fig. 34.

The three signals are actually very similar and are featuring irregular oscillation signals with averages at approximately 0.5 micron with peaks at 0,02 and 1.85 micron. As the feedrate increases, the deviation signal becomes less dense indicating a reduction of oscillation frequencies.

The simulation is repeated by varying the cycle time of the CNC controller and the results are shown in Fig. (35).

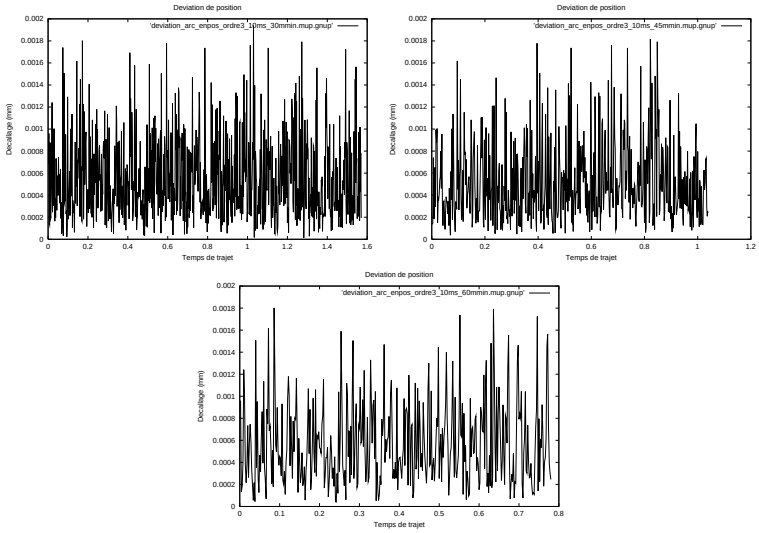

Fig. 34 Simulated path deviation for an arc : feedrates of 30, 45 et 60 m/min, cubic joint interpolation

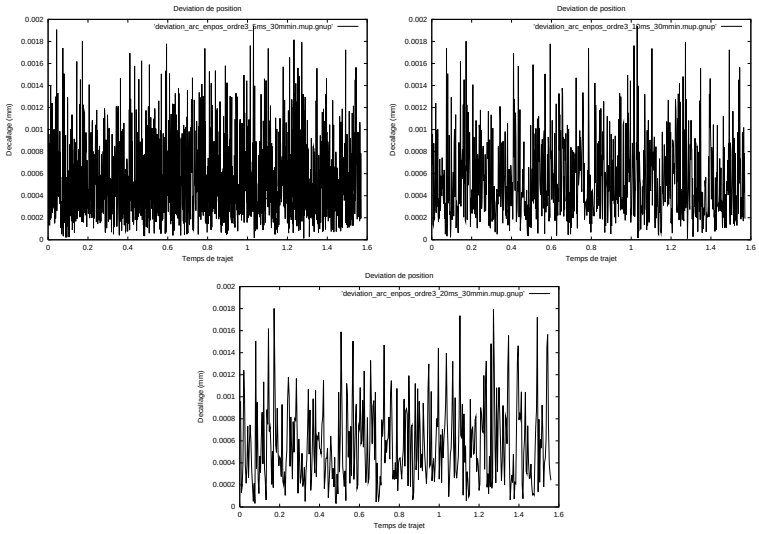

Fig. 35 Simulated path deviation for an arc : cycle times of 5, 10 et 20 ms, cubic joint interpolation

The three signals are actually very similar and are featuring irregular oscillation signals with averages at approximately 0,5 microns. Moreover, the deviation remains below 2 microns regardless of the case.

In Fig. 33, the simulation results are shown for the selected feedrates.

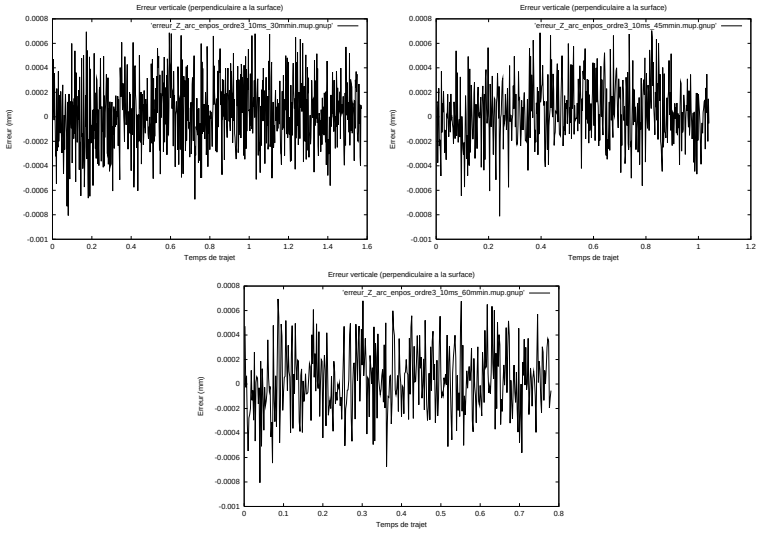

Fig. 36 Simulated vertical error for an arc : feedrates of 30, 45 et 60 m/min, cubic joint interpolation

The three signals are actually very similar and are featuring irregular oscillation signals with averages at approximately 0,1 micron with extremas at -0.4 and 0.6 micron and peaks at ± 0.8 micron.

We then study the vertical error where the controller cycle times are varied, Fig. 37.

As it was observed for the former tests, the vertical error signals are very similar and their density is inverserly proportional to the controller cycle time.

Tests ends by collecting the results onto the following table 5.

Ordre	T_p ms	F_r m/min	ε^{max} micron	ε^{min} micron	ε_Z^{max} micron	ε_Z^{min} micron	$\|\|\delta\|\|^{max}$ micron	$\|\|\delta\|\|^{min}$ micron
3	10	30	2.034	0.042	0.695	-0.806	1.939	0.012
3	10	45	1.991	0.023	0.704	-0.812	1.816	0.037
3	10	60	1.823	0.058	0.695	-0.806	1.802	0.040
3	5	30	2.196	0.023	0.704	-0.812	1.939	0.012
3	10	30	2.034	0.042	0.695	-0.806	1.939	0.012
3	20	30	1.823	0.058	0.695	-0.806	1.802	0.033

Table 5 Simulated errors and deviations for an arc : position control with cubic joint interpolation

The results barely exceed the value of 2 microns whatever the speed and response time.

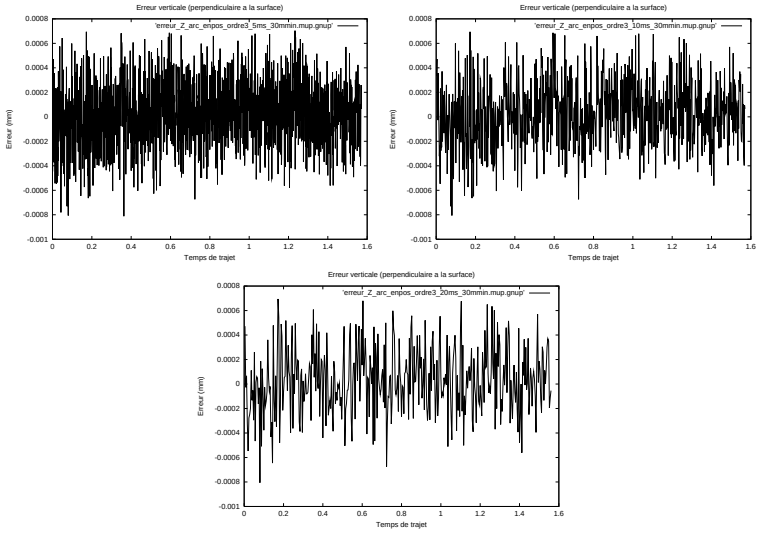

Fig. 37 Simulated vertical error for an arc : cycle times of 5, 10 et 20 ms, cubic joint interpolation

6.4.3 Discussion of the third order joint interpolation

A trajectory tracking using third order interpolators gives very satisfactory results. In all instances, deviation of less than 2 microns are obtained.

It is notable that the arc path results are better than for straight line segment. The difference between the error vector and deviation is at most 0.2 micron. As a consequence, the simulated path is not significantly delayed or ahead of the nominal path. It is observed that the curve is simulated even closer to the theoretical curve for the case of the line segment.

The results of surface finish indicate milling quality within 5 and 2 microns respectively for the line segments and circular arcs. Indeed, the results of the third order interpollation show that hexapod performance should be sufficient for UHSM (feedrate of 60 m/min or higher). Position control with cubic interpolators are highly recommended.

Furthermore, algorithms can be implemented in a conventional **CNC** adjusted with relatively slow response time.

Among other advantages, the following can be observed :

- The relative ease for calculating joint speeds at the beginning and the end of a trajectory interval.
- The continuity of movement is ensured.

The only drawback is that the acceleration continuity will not be ensured. Indeed, nothing prevents large acceleration variations to be applied on the motors.

6.5 Control with fifth order interpolation

6.5.1 Straight line segment with fifth order interpolation

Implementing interpolators with fifth order polynomial functions follows the goal to ensure acceleration continuity of interval transitions. Fig. 38 shows the results for the same straight line segment according to different selected feedrates.

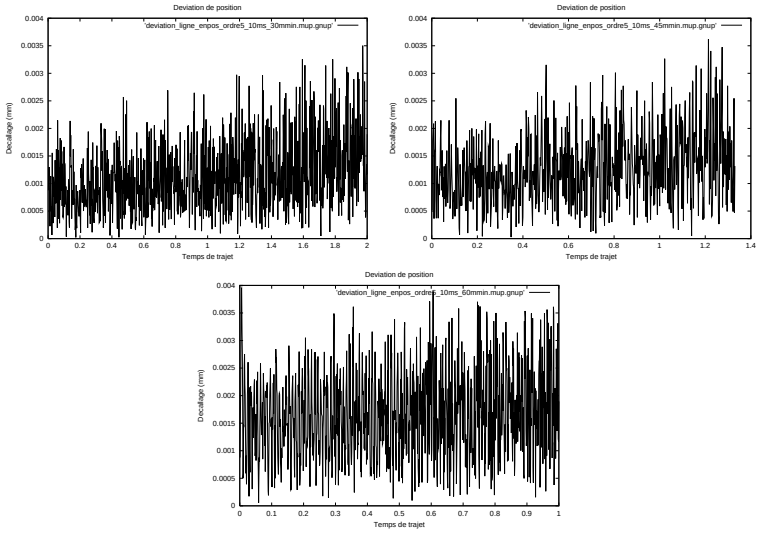

Fig. 38 Simulated path deviation for a straight line segment : feedrates of 30, 45 et 60 m/min, quintic joint interpolation

The three signals are featuring growing high frequency oscillations until the trajectory ends. These deviation signals seem similar to the third order results. The deviation remains below 4 microns. At the higher feedrate, the signal amplitude is sligthly larger at the beginning.

The simulation is repeated by varying the cycle time of the CNC controller and the results are shown in Fig. (39).

The three signals are featuring growing high frequency oscillations until the trajectory ends. The deviation signals share some commonality and contain oscillations remaining below 4 microns. Moreover, as the cycle time gets shorter, the signal becomes denser.

We continue the analysis by showing graphs of vertical errors. The following graph shows the results at different cycle times, 40.

The three signals are actually relatively similar and are featuring growing high frequency oscillations until the trajectory ends. At feedrates of 30 and 45 m/min, the graphics show signals which are centered on -0.2 micron with increasing oscillation

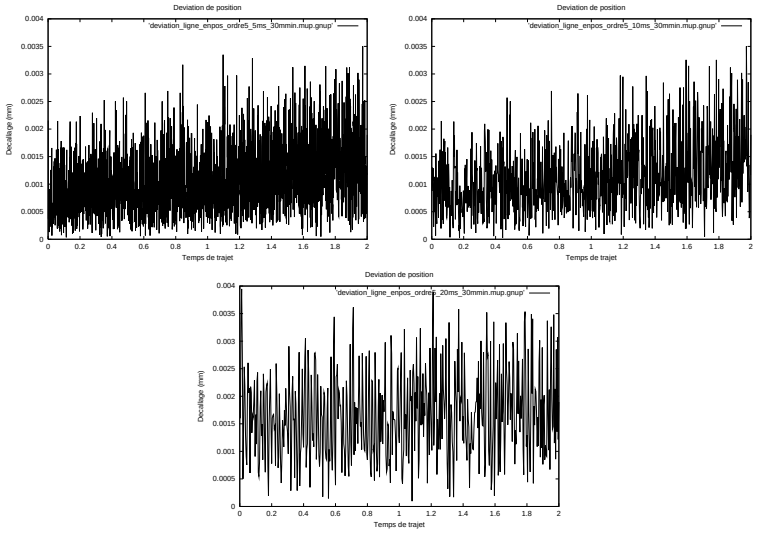

Fig. 39 Simulated path deviation for a straight line segment : cycle times of 5, 10 et 20 ms, quintic joint interpolation

with peaks getting close to -2 and 2 microns. At 60m/min, the oscillations start with larger amplitudes and grow less rapidly to reach -3,5 and 2 micron peaks.

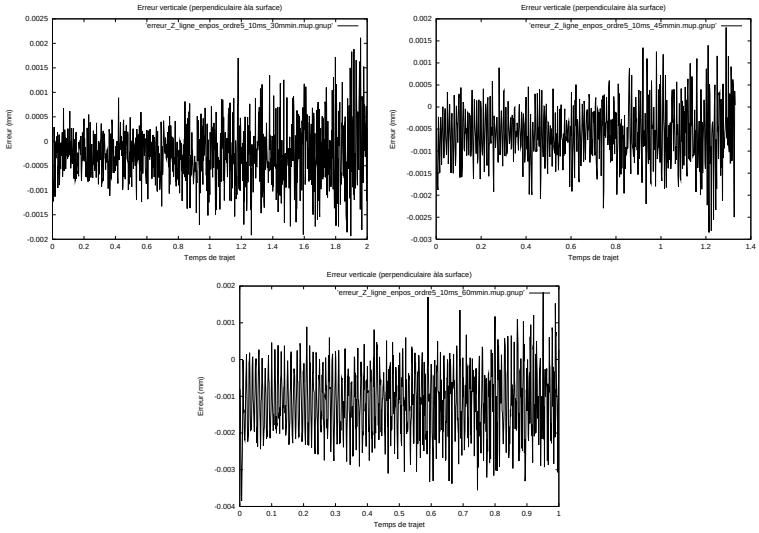

Fig. 40 Simulated vertical error for a straight line segment : feedrates of 30, 45 et 60 m/min, quintic joint interpolation

The next error results are calculated at cycle times, Fig. 40.

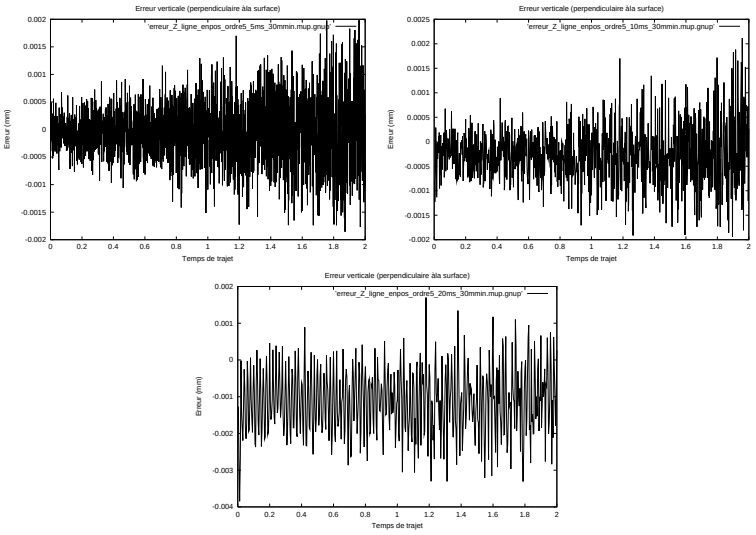

Fig. 41 Simulated vertical error for a straight line segment : cycle times of 5, 10 et 20 ms, quintic joint interpolation

These results show a similar trend then for changing the feedrates. At feedrates of 30 and 45 m/min, the graphics show signals which are centered on -0.2 micron with increasing oscillation with peaks getting close to -2 and 2 microns. At 60m/min, the oscillations start with larger amplitudes and grow less rapidly to reach -3 and 2 micron peaks.

Several tests were performed and the results are summarized on the following table 6.

Ordre	T_p ms	F_r m/min	ε^{max} micron	ε^{min} micron	ε_Z^{max} micron	ε_Z^{min} micron	$\|\|\delta\|\|^{max}$ micron	$\|\|\delta\|\|^{min}$ micron
5	10	30	3.886	0.087	2.113	-1.935	3.502	0.021
5	10	45	4.578	0.119	1.806	-2.847	3.621	0.028
5	10	60	4.992	0.136	1.830	-3.847	3.948	0.56
5	5	30	3.730	0.082	1.985	-1.857	3.502	0.035
5	10	30	3.886	0.087	2.113	-1.935	3.502	0.021
5	20	30	4.992	0.258	1.701	-3.847	3.948	0.096

Table 6 Simulated errors and deviations for a straight line segment : position control with quintic joint interpolation

Table 6 shows that the values are between 3.5 and 5.5 microns. Note that these values are proportional to the feed-rate or cycle time.

6.5.2 Arc with fifth order interpolation

The simulation calculations are repeated with the straight line nominal trajectory being replaced by the arc. Fig. 42 shows the results for different feedrates.

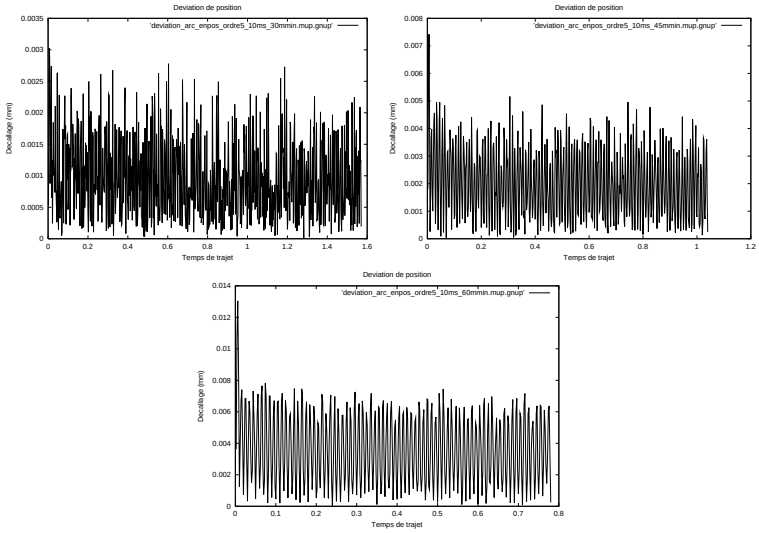

Fig. 42 Simulated path deviation for an arc : feedrates of 30, 45 et 60 m/min, quintic joint interpolation

The results show oscillations which tend to be slightly decreasing with a significant peak at the beginning. Signals are centered on a decreasing affine curve. At 30 m/min, the largest oscillation deviation is 3 microns. Nothwithstanding the starting peak, at 45 m/min, the deviation remains below the value of 5 microns. At 60/min, the oscillation deviation starts at 8 microns.

After varying the feed rate, the simulation is repeated varying the cycle time of the CNC controller and the results are shown in Fig. 43.

These deviation signals seem similar to the results of the former tests where feedrates are modified. At a cycle time of 5 ms, the high frequency oscillating signal is irregular and slowly reducing. Some peaks are observed at 2 microns. At 10 ms, similar observations can be made but the peaks reach 2,7 microns. At 20 ms, not considering the starting peak, the oscillatory signal is slightly decreasing and the peaks start at 8 microns.

In Fig. 40, the vertical error is calculated on the selected feed-rates.

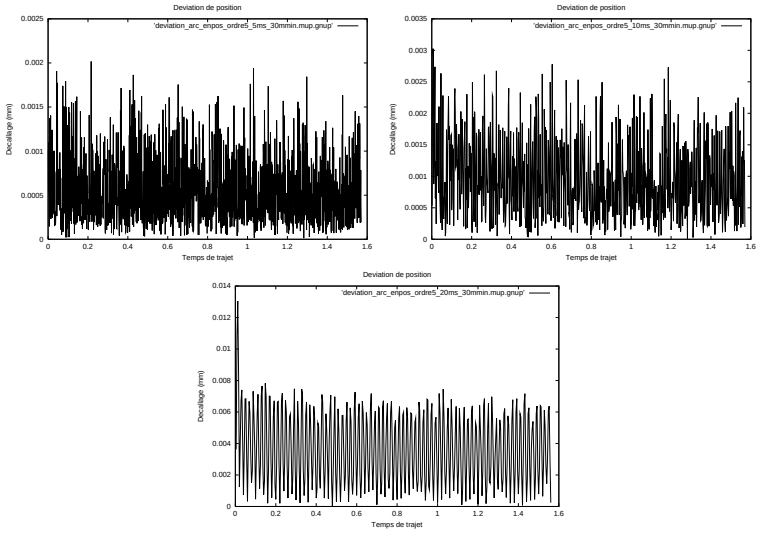

Fig. 43 Simulated path deviation for an arc : cycle times of 5, 10 et 20 ms, quintic joint interpolation

The curves are oscillations of relatively constant amplitude around a slightly increasing curve. As the feed-rate increases, the oscillation center is shifted down and

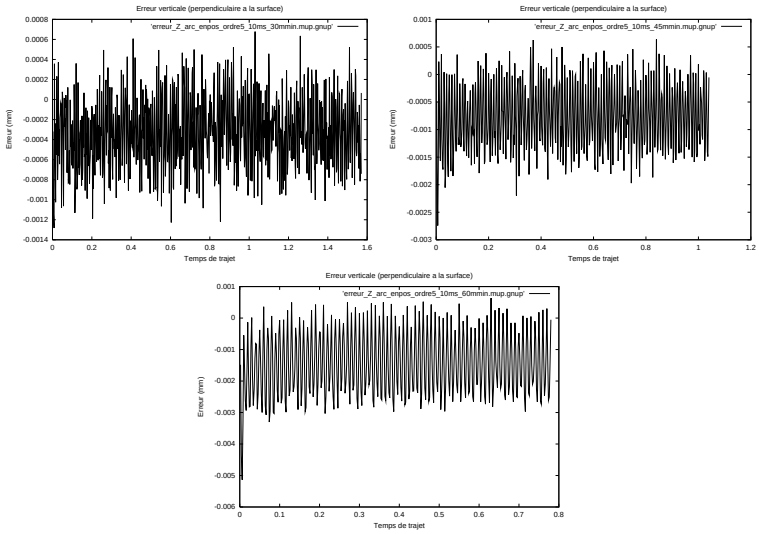

Fig. 44 Simulated vertical error for an arc : feedrates of 30, 45 et 60 m/min, quintic joint interpolation

the amplitude becomes larger. For the cycle time of 30 m/min, 45 m / min and 60 m / min, we observe linearized averages of respectively 0.3, 0.9, 1.2 micron and amplitudes of [-1, 0.6], [-1.8, 0.4], [-3, 0.5] microns respectively.

Then, we study the vertical error for different cycle times, Fig. 45.

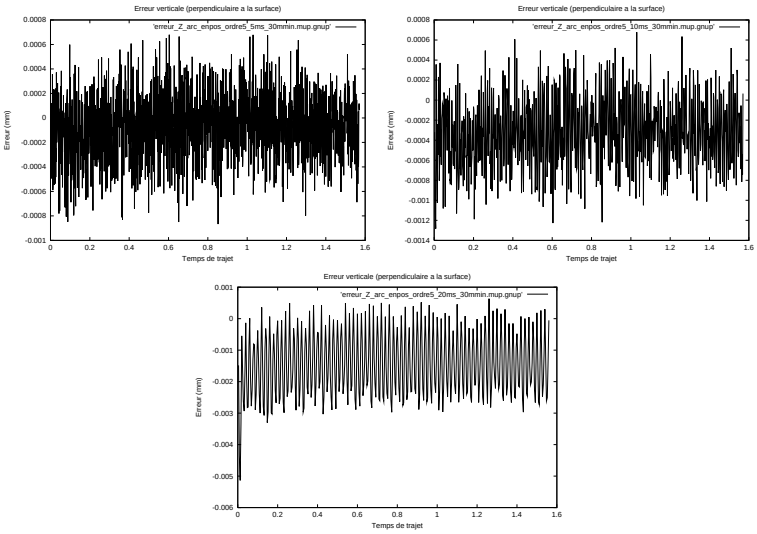

Fig. 45 Simulated vertical error for an arc : cycle times of 5, 10 et 20 ms, quintic joint interpolation

As expected, these vertical error signals seem similar to the results of the former tests where feedrates are modified. The curves are determined by a center following a slightly non-linear curve. At the cycle time of 5 ms, the average is approximated to -0.1 micron, with sligthly decreasing amplitudes, where the extremas are -1 and 0.5 micron and peak reaches -1.2 and 0.6 micron. Time of 20 ms, the average signal is -1 micron with peaks amount to 0.6 micron and down almost to 3 microns.

To terminate this fifth order interpollation simulation cycle, the results are collected and presented in table 7.

Ordre	T_p ms	F_r m/min	ε^{max} micron	ε^{min} micron	ε_Z^{max} micron	ε_Z^{min} micron	$\|\|\delta\|\|^{max}$ micron	$\|\|\delta\|\|^{min}$ micron
5	10	30	3.037	0.042	0.679	-1.283	3.028	0.012
5	10	45	7.425	0.023	0.635	-2.742	7.423	0.037
5	10	60	13.052	0.129	0.635	-5.136	13.043	0.043
5	5	30	2.156	0.023	0.680	-0.867	2.017	0.012
5	10	30	3.037	0.042	0.679	-1.283	3.028	0.012
5	20	30	13.052	0.129	0.635	-5.136	13.0	0.037

Table 7 Simulated errors and deviations for an arc : position control with quintic joint interpolation

We observe that the performance starts to decrease as the feedrate exceeds 45 m/min or the response time exceeds 10 ms. For UHSM, the calculated surface finish is as high as 13 microns. Similar surface finish are reached if cycle times are as high as 20 ms. These poor results come from those first excessive unique peaks appearing at the beginning of the path error signals. If we remove this first dedrimental peak, then the table becomes the following 8. After excessive peak removal, at high feedrates and slower cycle times, deviations reach 8 microns and vertical errors vary between peaks of -3 and 0.7 microns.

Ordre	T_p ms	F_r m/min	ε^{max} micron	ε^{min} micron	ε_Z^{max} micron	ε_Z^{min} micron	$\|\|\delta\|\|^{max}$ micron	$\|\|\delta\|\|^{min}$ micron
5	10	30	3.037	0.042	0.679	-1.283	3.028	0.012
5	10	45	7.425	0.023	0.635	-2.00	5.10	0.037
5	10	60	13.052	0.129	0.635	-3.00	8.00	0.043
5	5	30	2.156	0.023	0.680	-0.867	2.017	0.012
5	10	30	3.037	0.042	0.679	-1.283	3.028	0.012
5	20	30	NA	0.129	0.635	-3.00	8.00	0.037

Table 8 Second table or errors and deviations for an arc : position control with quintic joint interpolation

6.5.3 Discussion on fifth-order interpolation

Simulations of the fifth order give satisfactory results for high speed milling providing the initial excessive peak is not taken into account. However, performance is not as good as for third order interpollations for very high speeds.

This interpollation stratedy is useful if it is necessary to ensure jerk continuity at path interval transitions. However, this approach requires the calculation of transitional jerks which may increase implementation complexity with time consumming computations where Jacobians and their derivatives are involved. These computations should be not be performed on-line by the robot controller but handled off-line by the CAM program on the remote computer. The CAM program may not be capable to calculate actuated joint jerks.

This strategy is recommended as a second choice but it would be advisable to apply a third order interpollation costing less computation time.

6.6 Discussion on the results of position control

Linear orders are not recommended despite their simplicity because you can not perform high-speed machining. The third order gives the best results because the accuracy is always ensured to remain under 4 and 2 microns respectively for straight

line segments and arcs. Order 5 provides slightly less favorable results and it is more complex to implement.

The implementation of high order interpolators becomes difficult because you have to compute interval transition conditions that are not easy to calculate.

All interpolators allow to follow trajectories with feed-rates up to 30 m/min corresponding to **HSM** at rapid cycle times of 5 ms or less. Note that trying to verify **UHSM** with feedrates up to 60 m/min, the results indicate the application of order three or five. Any case is feasible with third order inrerpollations.

7 Conclusion

The existence of an exact method for solving the **FKP** of the general **6-6** hexapod allows the design of a complete kinematics simulator to study milling processes. A certified calculation method of the robot end-effector position has been implemented in the analysis of milling task. It consists of a trajectory following algorithm required for task planning applications, simulation and control. Several modeling modules can simulate various essential elements: parallel manipulator configuration, kinematics modeler and solver, CNC control algorithms, set-point interpollators and performance calculations. For performance evaluation, new metrics were proposed to evaluate surface finish more accurately.

This simulation package provides a kinematics result in the form of the trajectory deviation and vertical error as a lower bound on the estimation of the surface finish of any milling task.

We studied the performance of the classic CNC position control scheme applied to the general **6-6** parallel robots and compared it with an existing hexapod. Modeling of various interpollation strategies at various feedrates and cycle times allowed us to determine that milling quality surface finish can be obtained for **HSM** if third-order interpolations are implemented. We can also implement functions interpolations of the fifth order, but we must implement control cycle time less than or equal to 10 ms but they remain more mathematically involved to prepare. Linear interpollations will not allow for **HSM** and will only be limited to roughing at feedrates slower then 20 m/min.

With position control, **UHSM** becomes only feasible if third order interpolations are established. Results are slightly better for arcs then for straight line segments.

This work has allowed the design and programming of a complete robotic simulation package served as the backbone for the complete high speed milling simulation program prepared as a collaboration of the INRIA in Nancy and Paris VI Universityto fine-tune general Gough platforms and their position-based CNCs.

Acknowledgments

This research work was produced by the author during his PhD and with special funding from the Lorraine Region, the INRIA and CMW-Marioni. It has helped French hexapod manufacturers to fine-tune their milling machines.

References

1. Abdellatif H. and Heimann B. Adapted time-optimal trajectory planning for parallel manipulators with full dynamic modelling. In: IEEE Int. Conf. on Robotics and Automation, pages 413-418, Barcelona, 19-22 Avril 2005
2. Bayaziz O.B., Xie D., and Anamato N.M. Iterative relaxation of constraints: a framework for improving automated motion planning. In: IEEE Int. Conf. on Robotics and Automation, Barcelona, 19-22 Avril 2005
3. Bhattacharya S., Hatwal H., and Ghosh A. Comparison of an exact and an approximate method of singularity avoidance in platform type parallel manipulators. Mechanism and Machine Theory, 33(7):965-974, Octobre 1998
4. Bohigas O. et al. A singularity-free path planner for closed-chain manipulators. In IEEE Int. Conf. on Robotics and Automation, pages 2128-2134, Saint Paul, 14-18 May 2012
5. Bohigas O., Manubens M., and Ros L. Planning singularity-free force-feasible paths on the Stewart platform. In ARK, pages 245-253, Innsbruck, 25-28 June 2012
6. Brady M. et al. Robot motion : planning and control. MIT Press, 1982.
7. Briot S. and Arakelian V. Optimal force generation in parallel manipulators for passing through the singular positions. Int. J. of Robotics Research, 27(2):967-983, August 2008
8. Carbone G, Gmez-Bravo F, Selvi O. An Experimental Validation of Collision-Free Trajectories for Parallel Manipulators. Mechanics Based Design of Structures and Machines, 10/2012, Volume 40, Issue 4, pp. 414 - 433
9. Carbone G et al. An optimum path planning for Cassino Parallel Manipulator by using inverse dynamics. Robotica, 1997, Volume 26 / Issue 02 / March 2008, pp 229 239.
10. Chablat D. and Wenger P. Moveability and collision analysis for fully-parallel manipulators. In 12th RoManSy, pages 61-68, Paris, 6-9 July 1998.
11. Chedmail P., Hascoet J.Y. and Guerin F. Collision detection analysis for milling. Advances in Manufacturing Systems, 1 :pages 247-252, 1994.
12. Chen C-T. and Chi H-W. Singularity-free trajectory planning of platform-type parallel manipulators for minimum actuating efforts and reactions. Robotica, 26(3):371-384, May 2008
13. Chen C-T. and Liao T.T. Optimal path programming of the Stewart platform manipulator using the Boltzmann-Hamel-d'Alembert dynamics formulation model. Advanced Robotics, 22(6-7):705-730, 2008.
14. Chen Y., McInroy J.E., and Yi Y. Optimal, fault-tolerant mappings to achieve secondary goals without compromising primary performance. IEEE Trans. on Robotics and Automation, 19(4):681-691, August 2003
15. Coiffet P. Les robots, tome 1, modelisation et commande. Hermes, Paris, 1986.
16. Corts J. Motion planning algorithms for general closed-chain mechanisms. Ph.D. Thesis, Institut National Polytechnique de Toulouse, Toulouse, 16 Decembre 2003
17. Corts J., Simon T., and Laumond J-P. A random loop generator for planning the motions of closed kinematic chains using PRM methods. In IEEE Int. Conf. on Robotics and Automation, pages 2141-2146, Washington, 11-15 May 2002
18. Corts J. and Simon T. Probabilistic motion planning for parallel mechanisms. In IEEE Int. Conf. on Robotics and Automation, pages 4354-4359, Taipei, 14-19 Septembre 2003.

19. Dallefrate D. and others . A feed rate optimization technique for high-speed CNC machining with parallel manipulators. In 3rd Chemnitzer Parallelkinematik Seminar, pages 371-388, Chemnitz, 23-25 April 2002
20. Daney D. Etalonnage geometric des robots paralleles. PhD thesis, Universite de Nice-Sophia Antipolis, 2000.
21. Dasgupta B. and Mruthyunjaya T.S. Singularity-free path planning for the Stewart platform manipulator. Mechanism and Machine Theory, 33(6):711-725, August 1998
22. Dash A.K. et al. Workspace analysis and singularity-free path planning of parallel manipulators. In Int. Conf. on Mechatronics Technology (ICMT), pages 457-462, Fukuoka, 29 Septembre-3 Octobre, 2002.
23. Dash A.K. et al. Singularity-free path planning of parallel manipulators using clustering algorithm and line geometrie. In IEEE Int. Conf. on Robotics and Automation, pages 761-766, Taipei, 14-19 Septembre 2003
24. Dash A.K. et al. Workspace generation and planning singularity-free path for parallel manipulators. Mechanism and Machine Theory, 40(7):778-805, Juillet 2005.
25. Depince P., Hascoet J.Y. and Furet B. Compensation de trajectoire d'usinage : simulation et experimentation. In Proceedings of 13e Congrs franais de mcanique, volume 3, pages 293-296, Septembre 1997.
26. Dombre E. and Khalil W. Modelisation, identification et commande des robots, seconde dition. Robotique. Hermes, traite des nouvelles technologies edition, 1999.
27. Huang T. and others . Time minimum trajectory planning of a 2-dof translation parallel robot for pick-and-place operations. Annals of the CIRP, 56/1/2007:365-368, 2007.
28. Jui C.K.K. and Sun Q. Path trackability and verification for parallel manipulators. In IEEE Int. Conf. on Robotics and Automation, pages 4336-4341, Taipei, 14-19 Septembre 2003.
29. Khoukhi A., Baron L., and Balazinski M. Constrained multi-objective trajectory planning of parallel kinematic machines. Robotics and Computer-Integrated Manufacturing, 25(4-5):756-769, Aot 2009
30. Samir Lahouar, Sad Zeghloul, Lotfl Romdhane. Singularity Free Path Planning for Parallel Robots. Advances in Robot Kinematics: Analysis and Design 2008, pp 235-242
31. Latombe J.C. Robot motion planning. Kluwer Academic Publisher, 1991.
32. Liegeois A. Les robots, tome 7, analyse des performances et CAO. Hermes, Paris, 1984.
33. Liu G., Trinkle J.C., and Shvalb N. Motion planning for a class of planar closed-chain manipulators. In IEEE Int. Conf. on Robotics and Automation, pages 133-138, Orlando, 16-18 Mai 2006
34. Lozano-Prez T. et Wesley M. An algorithm for planning collision-free paths among polyhedral obstacles. In Communitations of ACM, volume 22, pages 560-570, 1979.
35. Luh J.Y.S. and Lin C.S. Optimum path planning for mechanical manipulators. Transations of the ASME, pages 142-151, June 1981.
36. R. Magnin et J.-P. Urso. Commande numerique, programmation. Memotech. 1991.
37. Marty C., Cassagnes C. and Martin P. La pratique de la commande numerique des machines-outils. Technique et documentation. Lavoisier, Paris, 1993.
38. Masory O. and Xiu D. Contour errors in a new class of CNC machine tools. In Proceedings of WAC98, volume 1, pages 791-798, 1998.
39. Merlet J.P. Manipulateurs paralleles, septieme partie : Verification et planification de trajectoire dans l'espace de travail. Technical Report 1940, INRIA, Sophia-Antipolis, June 1993.
40. Merlet J-P. An efficient trajectory verifier for motion planning of parallel machine. In Parallel Kinematic Machines Int. Conf., Ann Arbor, 14-15 Septembre 2000
41. Merlet J-P. A Generic Trajectory Verifier for the Motion Planning of Parallel Robots. Journal of mechanical design (2001). Volume: 123, Issue: 4, Page: 510-515
42. Merlet J-P. A local motion planner for closed-loop robots. In IEEE Int. Conf. on Intelligent Robots and Systems (IROS), pages 3088-3093, San Diego, 22-26 Septembre 2007
43. Merlet J-P. and Mouly N. Espace de travail et planification de trajectoire des robots parallles plans. Technical Report 2291, INRIA, Sophia-Antipolis, February 1994.

44. Merlet J.P., Perng M.W. and Daney D. Optimal trajectory planning of 5-axis machine-tool based on a 6-axis parallel manipulator. Advances in Robot Kinematics, 1(1) :pages 315-322, 2000.
45. Mery B. Machines a commande numerique. Hermes, Paris, 1997.
46. Nenchev D.N. and Uchiyama M. Singularity-consistent path planning and control of parallel robot motion through instantaneous-self-motion type. In IEEE Int. Conf. on Robotics and Automation, pages 1864-1870, Minneapolis, 24-26 Avril 1996
47. Nguyen C.C. and others . Trajectory planning and control of a Stewart platform-based end-effector with passive compliance for part assembly. J. of Intelligent and Robotic Systems, 6(2-3):263-281, Dcembre 1992
48. N. Nilsson. A mobile automaton : an application of artificial intelligence. In Proceedings of the International Joint Conference on Artificial Intelligence, pages 509-520, 1969.
49. Oen K-T. and Wang L-C T. Optimal dynamic trajectory planning for linearly actuated platform type parallel manipulators having task space redundant degree of freedom. Mechanism and Machine Theory, 42(7):727-750, Juin 2007.
50. Patel A. and K. Ehmann K. Volumetric error analysis of a stewart platform based machine tool. In Annals of the CIRP, volume 46, pages 287-290, 1997.
51. Pouyan A. et al. Eliminating redundancy and singularity in robot path planning based on masking. Expert Systems with Applications. Volume 37, Issue 9, September 2010, Pages 6213-6217
52. Pugazhenthi S., Nagarajan T. and Singaperumal M. Optimal trajectory planning for a hexapod machine tool during contour machining. Proceedings of the Institution of Mechanical Engineers. Part C, Journal of mechanical engineering science A. 2002, vol. 216, n 12, pp. 1247-1257.
53. Merlet J. P. Les Robots Parallles, 2nd edition. Herms, Paris (1997).
54. Dieudonne J. E., Parrish R. V. and Bardusch R. E. An actuator extension transformation for a motion simulator and an inverse transformation applying Newton?Raphson?s method, Technical Report D-7067, NASA, Washington, (1972).
55. Lazard D. On the representation of rigid-body motions and its application to generalized platform manipulators, Journal pf Computational Kinematics, Volume 1, Pages 175?182 (1993).
56. Raghavan M. The Stewart platform of general geometry has 40 configurations, ASME Journal of Mechanical Design, Volume 115, Pages 277?282 (1993).
57. Raghavan M. and Roth B. Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators, Transactions of ASME, Volume 117, Pages 71?79 (1995).
58. Rolland L. Introduction to algebraic methods for solving the forward kinematics problem of parallel robots applied to high throughput and high accuracy. In 3rd European-Asian Congress on Mechatronics, Besancon, 9-11 Octobre 2001.
59. Rolland L. Certified solving of the forward kinematics problem with an exact algebraic method for the general parallel manipulator Advanced Robotics, Vol. 19, No. 9, pages 995?1025 (2005)
60. Rolland L. Synthesis on modeling and certified solving of the kinematics problems of Gough-Type parallel Manipulator with an exact algebraic method, book chapter in Parallel Manipulators, Towards New Applications, Huapeng Wu editor, I-Tech Education and Publishing, Vienna, 2008, pp. 175-206.
61. Salerni G. The linear delta. Technical report, University of Pisa, 1995.
62. Shulz et al. Dynamic stiffness and contouring accuracy of a hsc linear motor machine. In Proceedings of the 2nd International Conference on High Speed Machining, volume 1, pages 75-83, 1999. Darmstadt
63. Shulz H., Gao H. and Stanik B. Analysis and optimization of the dynamic contouring accuracy using the example of a linear motor machine tool. In Proceedings of the 2nd International Conference on High Speed Machining, volume 1, pages 107-115, 1999. Darmstadt.
64. Sen S., Dasgupta B., and Mallik A.K. Variational approach for singularity-path planning of parallel manipulators. Mechanism and Machine Theory, 38(11):1165-1183, Novembre 2003

65. Shaw D. and Chen Y-S. Cutting path generation of the Stewart platform-based milling machine using an end-mill. Int. J. Prod. Res., 39(7):1367-1383, 2001.
66. Soni A.H., Tanasi G.C., and Varanasi S. Closed-loop multi-degree freedom mechanisms for surface generation and patching in machining 3d surfaces. In 9th IFToMM World Congress on the Theory of Machines and Mechanisms, pages 2668-2674, Milan, 30 August-2 Septembre, 1995
67. Su H-J., Dietmaier P., and J.M. McCarthy. Trajectory planning for constrained parallel manipulators. ASME J. of Mechanical Design, 125(4):709-716, Decembre 2003
68. Takeda Y. Kinematic analysis of parallel mechanisms at singular points at which a connecting chain has local mobility. In Computational Kinematics, Cassino, 4-6 May 2005
69. R. Taylor. Planning and execution of straight line manipulator. IBM Journal of Research Developpment, 23(4) : pages 424-436, 1979.
70. Tchon K. et al. Motion planning for parallel robots with non-holonomic joints. In ARK, pages 115-122, Innsbruck, 25-28 Juin 2012
71. P. Tournassoud. Planification et controle en robotique, application aux robots mobiles et manipulateurs. Robotique. Hermes, Paris, Traité des nouvelles technologies edition, 1992.
72. Trinkle J.C. and R.J. Milgram. Complete path planning for closed kinematic chains with spherical joints. Int. J. of Robotics Research, 21(9):773-789, Septembre 2002
73. S.M. Udupa. Collision detection and avoidance in computer controlled manipulators. In Proceedings of the International Joint Conference on Artificial Intelligence, pages 737-748, 1977.
74. Ur-Rehman R., Caro S., Chablat D., and Wenger P. Multi-objective path placement of parallel kinematics machines based on energy consumption, shaking forces and maximum actuator torques: application to the Orthoglide. Mechanism and Machine Theory, 45(8):1125-1141, August 2010
75. Vaca R., Aranda J., and Thomas F. Simpified Voronoi diagrams for motion planning of quadratically-solvable Gough-Stewart platforms. In ARK, pages 157-164, Innsbruck, 25-28 June 2012
76. Vaishnav R.N. and Magrab E.B. A general procedure to evaluate robot positioning error. The International Journal of Robotics Research, 6(1) : pages59-74, 1987.
77. Yakey J.H. et al. Randomized path planning for linkages with closed kinematic chains. IEEE Trans. on Robotics and Automation, 17(6):951-958, Decembre 2001