An NC Code Based Machining Movement Simulation Method for a Parallel Robotic Machine

Xu Shen[1], Fugui Xie[1,2], Xin-Jun Liu[1,2(✉)], and Rafiq Ahmad[3]

[1] The State Key Laboratory of Tribology and Institute of Manufacturing Engineering, Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China
x-chel4@mails.tsinghua.edu.cn,
{xiefg,xinjunliu}@mail.tsinghua.edu.cn

[2] Beijing Key Lab of Precision/Ultra-Precision Manufacturing Equipments and Control, Tsinghua University, Beijing 100084, China

[3] Laboratory of Intelligent Manufacturing, Design and Automation (LIMDA), Mechanical Engineering Department, University of Alberta, Edmonton, AB T6G 2R3, Canada
rafiq.ahmad@ualberta.ca

Abstract. The virtual machine tool and Computer-Aided Manufacturing (CAM) simulation are widely adopted nowadays to lower the cost and save time. Although parallel robotic machines are becoming popular in industry due to its unique advantages in manufacturing application, few methods are available for its simulation. This paper presents a work achieved by combining conventional CAM analysis tool HSMWorks, Computer-Aided Design (CAD) software SolidWorks, and programming tools such as Python and MATLAB to realize the machining movement simulation of a parallel robotic machine. Firstly, an original NC code interpreter is compiled in Python that interprets G-code generated by HSMWorks. Then, necessary coordinate transformation and kinematic calculation are done by using MATLAB. Finally, driving data are imported into virtual machine tool in SolidWorks, and a complete motion simulation environment is then developed. The proposed method is a general approach, which can be upgraded and modified for the simulation of parallel robotic machines with any structure.

Keywords: Parallel robotic machine · NC code · Machining movement simulation · SolidWorks

1 Introduction

The concept of parallel mechanism machines is initially proposed by Pollard in 1938 [1] and then became an interesting topic both for research [2–9] and industrial applications. Since every chain is connected as a closed loop, errors in every chain are averaged and forces are distributed, which lead to higher accuracy [10], higher load capacity, and higher structure rigidity [11]. Due to its great potential and unique advantages, in cases where the complex faces were hard for the traditional serial machine tools to mill, it is a good choice for parallel robotic machines to finish rapid machining tasks [12–14].

Meanwhile, along with the fast development of the society, manufacturing industry are not very keen to use their resources (time and money) to build physical machine tools for performance test and optimization. The concept of Virtual Manufacturing (VM) is proposed as a system which can operate as a real machine to deal with models in the virtual world [15–19]. With this technology, researchers started to focus on how to better deal with the obstacles [20] and use visible decision-making method to generate safer too-paths [21]. Virtual Manufacturing and computer simulation system greatly avoided the unnecessary space waste in a real machine shop, saved money and work for building prototype and test, and increased the reliability and safety of the final product.

However, since the kinematics of parallel robot is sometimes more complex than traditional serial machine tools, it is troublesome to directly use manufacturing simulation software to simulate the working state of the parallel robotic machine. Therefore, a complete method for carrying out such simulation is in urgent need for manufacturing assessment.

A 5-degree-of-freedom (DOF) spatial parallel kinematic mechanism (PKM) with three limbs was proposed, and its mobility, singularity, and kinematics analysis were analyzed recently [22]. On this basis, the work in this paper is carried out. To achieve the aim of universality, the work here focuses on NC G-code interpreting to provide the robot with detailed machining data. Therefore, as long as there are G-code and its syntax documents, a complete manufacturing plan can be acquired no matter how the G-code is originally generated. After the complete process of "CAM Analysis—Interpreter—Kinematic Operator—Tool and Stock Setting", a machining movement simulation can be produced for further demands such as collision inspection and structure modification.

2 Computer-Aided Manufacturing (CAM) Analysis

The structure of a G-code program for 2D pocket milling generated by CAM software HSMWorks is presented and analyzed as follows. For lack of space, the program posted here contains only example of non-repetitive and motion-related commands.

The workpiece to be milled is presented in Fig. 1(a). The most important parameter to be set is the work coordinate system (WCS), which is crucial for the robotic machine tool presented in the following simulation. Post this milling process under Siemens SINUMERIK 810D postprocessor configuration, and the corresponding G-code and trajectory simulation can be seen in HSMWorks NC Editor (Fig. 1(b)). In real manufacturing cases, this G-code can be directly imported into the machine and start milling.

G-code:

```
N11 G90 G94
N12 G71
N14 G17
N15 G0 SUPA Z80 D0
N22 X-8.511 Y20.349
N24 Z86
N25 Z76
N26 G1 Z75.499 F500
  N27 G17
  N28 G3 X -8.498 Y20.331
Z75.203 I7.704 J5.559
  ......
  N120 G2 X -78.169 Y23.55
I-151.12 J-64.574
  ......
```

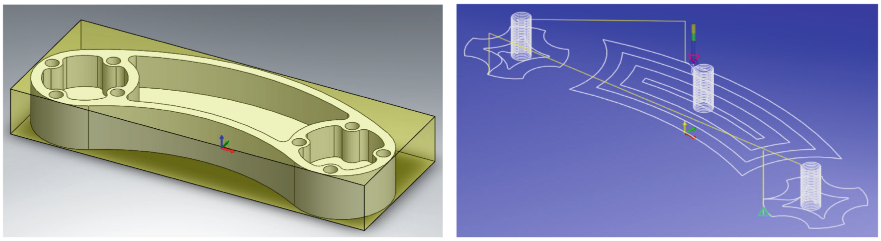
(a) (b)

Fig. 1. Milling work setting: (a) workpiece; (b) CAM plan

According to SIEMENS SINUMERIK 840D sl/840Di sl/840D/840Di/810D Fundamentals [23], key commands in each block are demonstrated in Table 1.

Table 1. Some key commands in G-code program of this task

Commands	Meaning
G90	Absolute coordinate mode
G94	Feedrate per minute
G71	Metric dimensions (length [mm])
G17	XY-plane specification
G0	Rapid move
G1	Linear move
G2/G3	Circular interpolation, clockwise/counterclockwise

3 NC Code Interpreter Design

As a CAD software, SolidWorks is able to receive direct position coordinates rather than G-code commands. It is necessary to design an interpreter to interpret the G-code initially designed for serial machine tools into actual kinematics information.

Recognition method is illustrated in the flowchart as shown in Fig. 2. The interpreter consists of three units: Scanner, State Storage, and Interpolator.

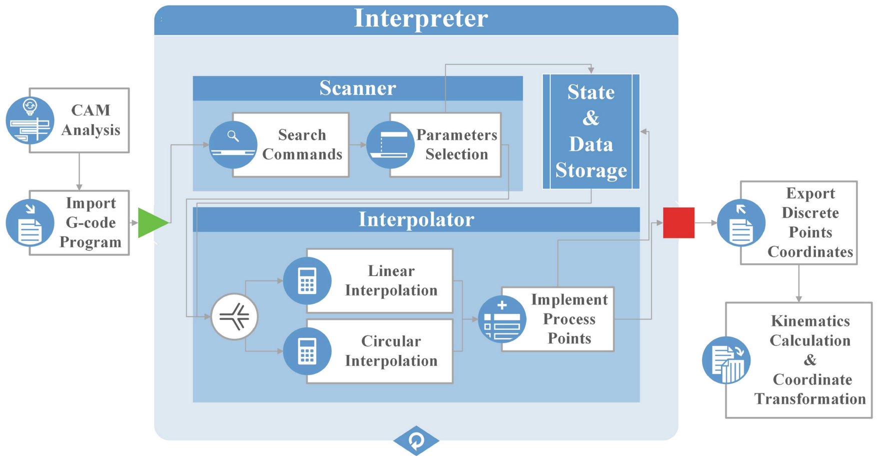

Fig. 2. Working principle of the interpreter

3.1 Scanner

The Scanner is used to read and analyze each G-code block, which contains key commands for tool's motion. By using Python "readlines" function, it is convenient to import each G-code block as a separate line to the interpreter and analyze its components. Commands such as G17/18/19 and G0/1/2/3 and numerical parameters following X/Y/Z/I/J/K are all major searching tasks.

3.2 State Storage

The State Storage contains various vital state markers which will define the working mode and all further operations. It receives information from Scanner and stores those states in some variables. As long as there is no other mode change appearing, these variables stored in the State Storage unit will keep the same, thus offering guidance for interpolation. Some of the relevant variables are Working Plane, Cutting Mode (rapid/linear/circular), Rotating Direction of the circular interpolation, Feedrate (machining speed), Step Size of the interpolation and so forth.

3.3 Interpolator

The Interpolator is the core unit of the interpreter. It will be utilized to interpolate processing points between the starting point and end point following a certain kind of motion. During the milling process, there will be four different motion modes—Rapid traverse motion (G0), Linear interpolation (G1), Circular interpolation clockwise (G2), and Circular interpolation counter-clockwise (G3).

Rapid Traverse Motion (G0): G-code program does not contain any speed definition for the rapid traverse. Because G0 only makes it faster than G1 to move tool to the working region, it is acceptable to set a certain larger G0 speed according to the feed rate for G1.

Linear Interpolation (G1): The program will supplement processing points linearly between the starting point and end point using particular step size, while defining the exact time at every point. As a result, the interpreter will export an extensive data matrix made up of four columns, where every row is a 4-dimensional array—(Time, X coordinate, Y coordinate, Z coordinate).

Circular Interpolation (G2/G3): According to mathematical theory, in 3D space, a parametric equation of a circle with radius r is given by

$$\begin{cases} x(\psi) = p_x + r\cos(\psi)\cdot a_x + r\sin(\psi)\cdot b_x \\ y(\psi) = p_y + r\cos(\psi)\cdot a_y + r\sin(\psi)\cdot b_y \\ z(\psi) = p_z + r\cos(\psi)\cdot a_z + r\sin(\psi)\cdot b_z \end{cases} \tag{1}$$

Where the center of the circle is $\mathbf{p_c} = (p_x, p_y, p_z)$, and two orthonormal vectors in the plane containing the circle are $\overrightarrow{\mathbf{a}} = (a_x, a_y, a_z)$ and $\overrightarrow{\mathbf{b}} = (b_x, b_y, b_z)$.

Table 2. Key data and parameters picked by the Scanner

Name & Type	Source command	Extraction result
Starting point coordinate	`G0/1/2/3 X... Y... Z...`	$\mathbf{p_s} = (x_s, y_s, z_s)$
Endpoint coordinate	`G2/3 X... Y... Z...`	$\mathbf{p_e} = (x_e, y_e, z_e)$
Center point offset	`G2/3 I... J... K...`	$\mathbf{c_o} = (i, j, k)$

As is demonstrated previously, key data can be acquired by the Scanner (Table 2): These parameters are further processed to fit in the mathematical theory in (1):

- Center of the circle: $\mathbf{p_c} = (p_x, p_y, p_z) = (x_s + i, y_s + j, z_s + k)$
- Two vectors in the plane containing the circle are $\overrightarrow{\mathbf{a}}_\mathbf{0} = (x_s - p_x, y_s - p_y, z_s - p_z)$, $\overrightarrow{\mathbf{c}}_\mathbf{0} = (x_e - p_x, y_e - p_y, z_e - p_z)$, then normalized to $\overrightarrow{\mathbf{a}} = \overrightarrow{\mathbf{a_0}}/\left|\overrightarrow{\mathbf{a_0}}\right|$, $\overrightarrow{\mathbf{c}} = \overrightarrow{\mathbf{c_0}}/\left|\overrightarrow{\mathbf{c_0}}\right|$
- Radius of circle is $r = \sqrt{i^2 + j^2 + k^2}$.

To finish the parametric Eq. (1), the vector $\vec{\mathbf{b}}$, which is orthonormal with $\vec{\mathbf{a}}$, is still in need. The normal vector of the circle plane is given by $\overrightarrow{\mathbf{n}} = \overrightarrow{\mathbf{a}} \times \overrightarrow{\mathbf{c}}$. So the vector $\vec{\mathbf{b}}$ can be calculated by $\overrightarrow{\mathbf{b_0}} = \overrightarrow{\mathbf{n}} \times \overrightarrow{\mathbf{a}}$, which is then normalized to $\overrightarrow{\mathbf{b}} = \overrightarrow{\mathbf{b_0}}/\left|\overrightarrow{\mathbf{b_0}}\right|$.

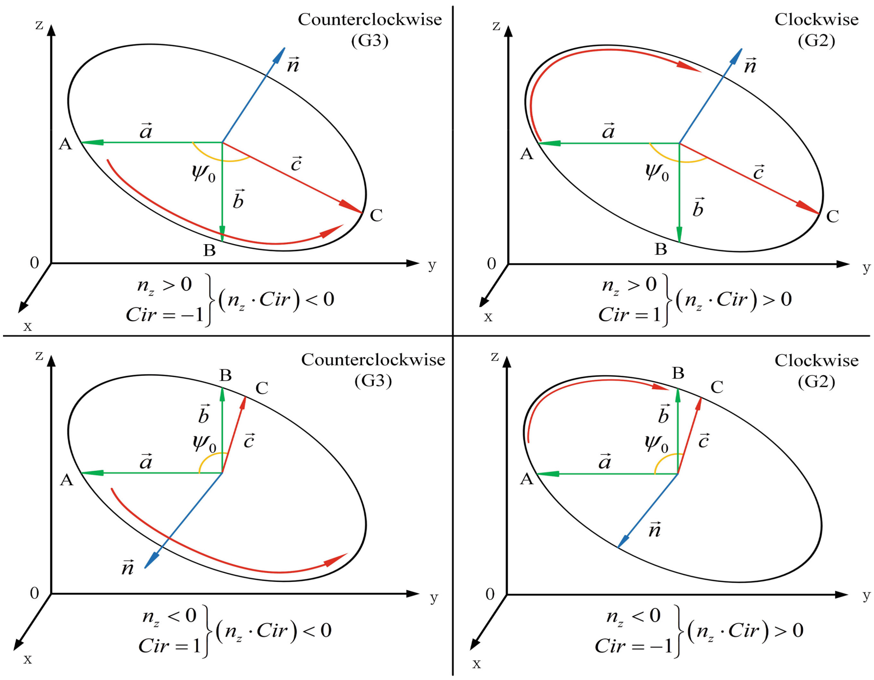
Counterclockwise (G3)
Clockwise (G2)
Counterclockwise (G3)
Clockwise (G2)
$\vec{a}$
$\vec{b}$
$\vec{c}$
$\vec{n}$
ψ_0
A
B
C
z
y
x
0
$n_z > 0$, $Cir = -1$ } $(n_z \cdot Cir) < 0$
$n_z > 0$, $Cir = 1$ } $(n_z \cdot Cir) > 0$
$n_z < 0$, $Cir = 1$ } $(n_z \cdot Cir) < 0$
$n_z < 0$, $Cir = -1$ } $(n_z \cdot Cir) > 0$

Fig. 3. All possible cases considering starting and end point positions and rotating directions

Considering the interpolating process, when $\psi = 0$, it is easy to find the relationship $(x(0), y(0), z(0)) = \overrightarrow{\mathbf{a_0}}$, which means the start of the circular motion.

Define the angle between $\overrightarrow{\mathbf{a_0}}$ and $\overrightarrow{\mathbf{c_0}}$ to be ψ_0, then the angle the cutting tool needs to cover is ψ_0 or $2\pi - \psi_0$. The interpolator can start to create processing points from the starting point to the endpoint by changing ψ step by step under the restriction $|\psi| < \psi_0$ or $|\psi| < 2\pi - \psi_0$. The problem arises here: how to distinguish these two cases?

In the parametric Eq. (1), if ψ increases from 0 to ψ_0, the result $(x(\psi), y(\psi), z(\psi))$ will always move in the direction that passing a minor arc from point A to point B along the circle. In contrast, the effect of decreasing ψ from 0 to $-(2\pi - \psi_0)$ will step on the other way. So the location of point B will determine the effect of changing ψ. Since the vector $\overrightarrow{\mathbf{b}}$ is derived from $\overrightarrow{\mathbf{n}} \times \overrightarrow{\mathbf{a}}$, the orientation of $\overrightarrow{\mathbf{n}}$ (more specifically, the sign of n_z) will be a major focus in the end.

Figure 3 illustrates four possible situations during circular motion. For the convenience of programming and analysis, a marker "*Cir*" is used to describe expecting rotating direction, that is, G2 (clockwise) will lead to "$Cir = -1$" and G3 (counterclockwise) will result in "$Cir = 1$".

By investigating all four cases, it can be summarized that:

(I) $(n_z \times Cir) > 0$: ψ should increase from 0 to ψ_0.
(II) $(n_z \times Cir) < 0$: ψ should decrease from 0 to $-(2\pi - \psi_0)$.

Until now it is finally clear that how the Interpolator should control variable ψ to finish circular interpolation. By following the changing direction and the extremum restriction, it only needs to change ψ by a certain step size to supplement processing points between the starting point and end point.

It is worthwhile to make a further explanation that if only two parameters show in (I, J, K) but three in (X, Y, Z), it means that there will be circular motion on the plane indicated by the existing two coordinates in (I, J, K) while a linear move occurs perpendicular to it, thus resulting in a helix interpolation as a whole.

To conclude this part, in a complete G-code program, there will be multiple G0/G1/G2/G3 commands, so the interpolation methods described above will be reused for a significant number of times. Via the interpreter, the G-code program will finally be translated into a large data matrix ${}^{s}\mathbf{I}=\begin{bmatrix}\vec{\mathbf{t}} & {}^{s}\mathbf{X} & {}^{s}\mathbf{Y} & {}^{s}\mathbf{Z}\end{bmatrix}$ in stock frame describing discrete points along its path (each line of $\begin{bmatrix}{}^{s}\mathbf{X} & {}^{s}\mathbf{Y} & {}^{s}\mathbf{Z}\end{bmatrix}$, Fig. 4) and their corresponding arriving time (column vector $\vec{\mathbf{t}}$). It will be then processed as the input for parallel robotic machine inverse kinematic calculation.

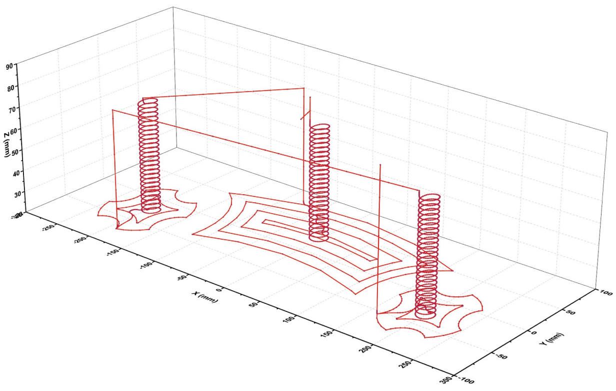

Fig. 4. Trajectory redrawn after interpretation

4 Machining Motion Simulation

The SPR-2(2UP-R-S) parallel robotic machine has been proposed in Ref. [22], the CAD model and kinematic scheme are shown in Fig. 5. From the inverse kinematics described in Ref. [22], it is clear that as long as there is a tool position coordinate, the length of five limbs will be available. However, the coordinate frames of the Interpreter and the Inverse Kinematic Operator (IKO) are different. The interpreter uses coordinate frame attached to stock while IKO uses frame attached to the machine base. So the combining of these two parts requires the transformation of the stock frame $\Re_{s}$ to the machine frame $\Re_{m}$.

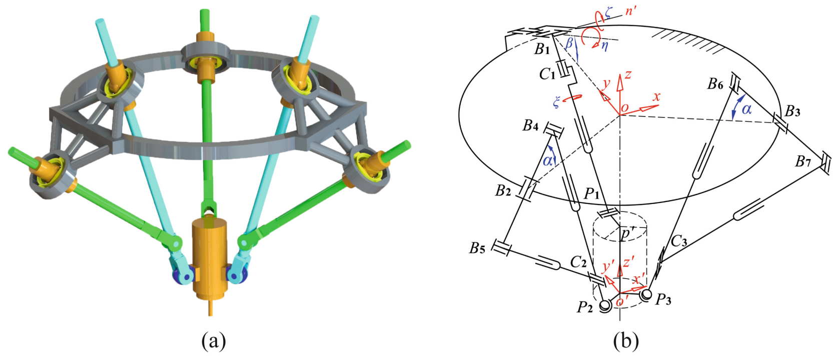

Fig. 5. The parallel robotic machine: (a) CAD model; (b) kinematic scheme

The force and position control of such a pocket milling job is vital for the final product's quality, where the chip removal will be a matter of concern with high priority. Therefore, the horizontal configuration is adopted for workpiece installation and machining (Fig. 6). In this case, chips will either automatically fall or be easily swept, avoiding the accumulation of the chips in vertical case and the corresponding problems. Moreover, the coolant can easily flow out in this configuration without accumulating and flooding in the pocket.

According to the definition of the two frames, the stock frame can be arrived by firstly a $\phi = 90^\circ$ rotation along the X axis of the machine frame and then a translation $[\mathrm{s_x} \quad \mathrm{s_y} \quad \mathrm{s_z}]^T$ to the stock position ($\mathrm{s_x}$, $\mathrm{s_y}$, $\mathrm{s_z}$ are coordinates of the stock frame origin in machine frame). The transformation can be achieved by applying homogeneous coordinate transformation matrix

$$ {}^{\mathbf{m}}_{\mathbf{s}}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & \mathrm{s_x} \\ 0 & \cos\phi & -\sin\phi & \mathrm{s_y} \\ 0 & \sin\phi & \cos\phi & \mathrm{s_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \mathrm{s_x} \\ 0 & 0 & -1 & \mathrm{s_y} \\ 0 & 1 & 0 & \mathrm{s_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

The coordinates in stock frame collected by the Interpreter will then be transformed to the machine frame by multiplying ${}^{\mathbf{m}}_{\mathbf{s}}\mathbf{T}$. This process is necessary before transporting data into the IKO.

$$ [{}^{\mathbf{m}}\mathbf{X} \quad {}^{\mathbf{m}}\mathbf{Y} \quad {}^{\mathbf{m}}\mathbf{Z} \quad \mathbf{1}]^T = {}^{\mathbf{m}}_{\mathbf{s}}\mathbf{T} \cdot [{}^{\mathbf{s}}\mathbf{X} \quad {}^{\mathbf{s}}\mathbf{Y} \quad {}^{\mathbf{s}}\mathbf{Z} \quad \mathbf{1}]^T \tag{3} $$

Moreover, there are still two parameters required in IKO not defined— the azimuth angle and the tilt angle (φ, θ). Since this task is a 2D milling job, (φ, θ) are constants that won't change through the whole process. Input initial values $(\varphi_0, \ \theta_0)$ to generate two column vectors $\vec{\varphi} = [\varphi_0 \quad \cdots \quad \varphi_0]^T$ and $\vec{\boldsymbol{\theta}} = [\theta_0 \quad \cdots \quad \theta_0]^T$, both of which have the same number of rows as $\vec{\mathbf{t}}$. Here the machine tool is set to work under horizontal configuration, so $\varphi_0 = 90^o$ and $\theta_0 = -90^o$.

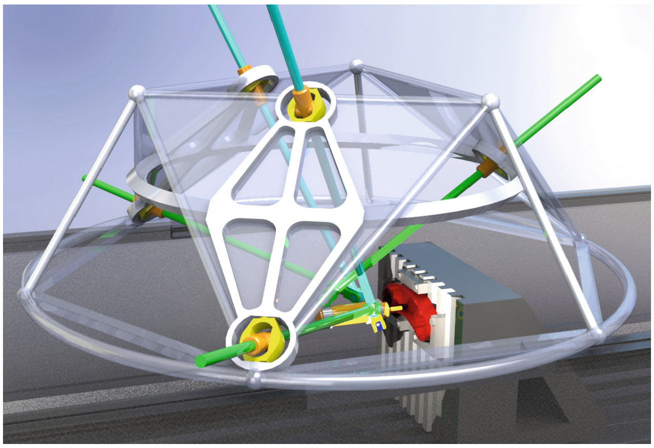

Fig. 6. Workpiece installation and machine tool application scene

Now all necessary inputs for IKO are ready as

$$^{\mathbf{m}}\mathbf{I} = \left[\, \vec{\mathbf{t}} \quad {}^{\mathbf{m}}\mathbf{X} \quad {}^{\mathbf{m}}\mathbf{Y} \quad {}^{\mathbf{m}}\mathbf{Z} \quad \overrightarrow{\varphi} \quad \overrightarrow{\theta} \,\right] \tag{4}$$

According to the algorithm in Ref. [22], $\mathbf{T} = [x, y, z]^T$ repeatedly picks three coordinates from every row in $[\,{}^{\mathbf{m}}\mathbf{X} \quad {}^{\mathbf{m}}\mathbf{Y} \quad {}^{\mathbf{m}}\mathbf{Z}\,]$ and $\mathbf{R}(\varphi, \theta, \sigma)$ picks (φ, θ) from every row in $\overrightarrow{\varphi}$ and $\overrightarrow{\boldsymbol{\theta}}$.

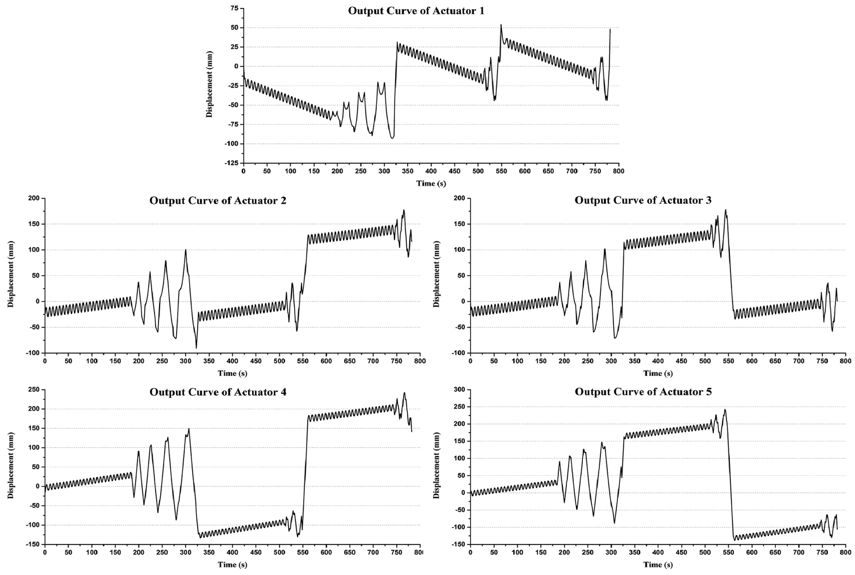

Fig. 7. Continuous output curve of five actuators L_1, L_2, L_3, L_4, and L_5

By defining the initial state of the machine tool and calculating all corresponding limb lengths as a datum, the displacement change that five actuators should generate will be determined. However, now they are only timeline-based discrete data showing the displacement change.

In SolidWorks, the feature "Motion Study" can define linear motors on actuators and import displacement data from files to offer instruction for motors' movement. It can also help users to plot discrete data points thus transforms discrete information into a continuous working curve (Fig. 7).

After finishing all setups and data import for actuators, click "Play" or export motion as a video file, the simulation is finally finished. Different camera views can be added to observe every desired detail of the machine tool motion. Moreover, various other analysis functions are provided by the software to provide the performance information of the robot.

5 Conclusion

In this paper, a machining motion simulation method is proposed for parallel robotic machines by utilizing G-code generated by CAM analysis and motion function in SolidWorks. There is no strict demand for CAM software or specific interface as long as manufacturing G-code for some machine tools and its supporting document are available. This feature significantly broadens the area where this method can be applied. In the part of the Interpreter design, Python programming language is employed because of its brevity in reading and processing information in each line of G-code text. Necessary interpolation in milling process is demonstrated with emphasis since G2 and G3 commands are hardest to be translated into motion data available for parallel kinematics. As for the IKO, the core solving principle is based on the result of previous work done in the lab. The last step requires some precise adjustment such as locating the stock position and choosing a proper machine tool configuration for manufacturing. The proposed simulation method can achieve the goal of closed-loop analysis to better detect the defects in the machine tool or manufacturing plan with high efficiency, which is crucial for fast design and prototyping.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under Grants 51425501 and 51675290.

References

1. Pollard, V., Willard, L.: Position-controlling apparatus. US, US2286571 (1942)
2. Fichter, E.F.: A Stewart platform-based manipulator: general theory and practical construction. Int. J. Robot. Res. **5**(2), 157–182 (1986)
3. Waldron, K.J., Hunt, K.H.: Series-parallel dualities in actively coordinated mechanisms. Int. J. Robot. Res. **10**(5), 473–480 (1991)
4. Hunt, K.H.: Structural kinematics of in-parallel-actuated robot-arms. ASME J. Mech. Trans. Autom. Des. **105**(4), 705–712 (1983)

5. Nanua, P., Waldron, K.J., Murthy, V.: Direct kinematic solution of a Stewart platform. IEEE Trans. Robot. Autom. **6**(4), 438–444 (1990)
6. Sugimoto, K.: Kinematic and dynamic analysis of parallel manipulators by means of motor algebra. ASME J. Mech. Trans. Autom. Des. **109**(1), 3–7 (1987)
7. Nguyen, C.C., Pooran, F.J.: Kinematic analysis and workspace determination of a 6 DOF CKCM robot end-effector. J. Mech. Work. Technol. **20**, 283–294 (1989)
8. Gosselin, C.: Determination of the workspace of 6-DOF parallel manipulators. ASME J. Mech. Des. **112**(3), 331–336 (1990)
9. Gosselin, C., Angeles, J.: Singularity analysis of closed-loop kinematic chains. IEEE Trans. Robot. Autom. **6**(3), 281–290 (1990)
10. Huang, Z., Kong, L.F., Fang, Y.F.: Theory and Control Mechanism for Parallel Robotics (1997)
11. Wang, Z., Wang, Z., Liu, W., Lei, Y.: A study on workspace, boundary workspace analysis and workpiece positioning for parallel machine tools. Mech. Mach. Theory **36**(5), 605–622 (2001)
12. Stewart, D.: A platform with six degrees of freedom. ARCHIVE Proc. Inst. Mech. Eng. 1847–1982 (vols. 1–196) **180**(1965), 371–386 (2013)
13. Kim, J., et al.: Performance analysis of parallel manipulator architectures for CNC machining applications. In: Proceedings of the IMECE Symposium on Machine Tools, Dallas (1997)
14. Ryu, S.-J., et al.: Eclipse: an overactuated parallel mechanism for rapid machining. In: Boër, C.R., Molinari-Tosatti, L., Smith, K.S. (eds.) Parallel Kinematic Machines. Springer, London (1999). doi:10.1007/978-1-4471-0885-6_32
15. Qingke, Y., Rujia, Z.: Virtual manufacturing system. China Mech. Eng. **4**, 10–12 (1995)
16. Iwata, K., et al.: Virtual manufacturing systems as advanced information infrastructure for integrating manufacturing resources and activities. CIRP Ann.-Manuf. Technol. **46**(1), 335–338 (1997)
17. Altintas, Y., et al.: Virtual machine tool. CIRP Ann. Manuf. Technol. **54**(2), 115–138 (2005)
18. Lin, W., Fu, J.: Modeling and application of virtual machine tool. In: International Conference on Artificial Reality and Telexistence–Workshops IEEE, pp. 16–19 (2006)
19. Carpenter, I.D., et al.: Virtual manufacturing. Manuf. Eng. **76**(S1), 113–116 (1997)
20. Ahmad, R., Plapper, P.: Safe and automated assembly process using vision assisted robot manipulator. Procedia CIRP **41**, 771–776 (2016)
21. Ahmad, R., Tichadou, S., Hascoet, J.-Y.: Generation of safe and intelligent tool-paths for multi-axis machine-tools in a dynamic 2D virtual environment. Int. J. Comput. Integr. Manuf. **29**(9), 982–995 (2016)
22. Xie, F., et al.: Mobility, singularity, and kinematics analyses of a novel spatial parallel mechanism. J. Mech. Robot. **8**(6), 061022 (2016)
23. Siemens Ltd.: Programming Manual of SINUMERIK 840D sl/840Di sl/840D/840Di/810D Fundamentals, 6FC5398-1BP10-2BA0, November 2006