

Constrained Identification of Virtual Drive Models Using a Genetic Algorithm

W. Wong, K. Erkorkmaz

Precision Controls Laboratory, Department of Mechanical and Mechatronics Engineering

University of Waterloo, Waterloo, Canada

ww3wong@uwaterloo.ca, kaane@uwaterloo.ca

Abstract

This paper presents a technique for extracting virtual models of machine tool drives with minimal intervention to the production machine. The identification is carried out by running a short G-code test and capturing commanded and measured position readings. The methodology is fairly general and can be applied to different types of drives (linear or ball screw) controlled with a large class of servo techniques. Due to the limited excitation which can be delivered by the interpolator, the convergence of parameters to their true values is not guaranteed. However, key dynamic characteristics which represent the tracking and disturbance rejection are captured with sufficient closeness in the frequency range of motion commands. In order to ensure stability of the identified models, bounds are imposed on the frequency and damping ratio of the closed loop poles. The constrained identification problem is solved efficiently using a Genetic Algorithm. It is shown that the identified drive models can be used to predict and optimize the contouring performance of machine tools in a virtual machining environment, as verified by experimental results.

Keywords:

Virtual CNC, Identification, Genetic Algorithms

1 INTRODUCTION

Virtual CNC (VCNC) enables the prediction and optimization of a machine tool's contouring performance at the design, development, and end use stages [1]. To achieve this, the CNC dynamic model needs to be constructed correctly. Typical model building techniques require disconnecting the servo loop or interpolator, and conducting a series of time or frequency domain identification tests. These may not always be practical to apply on production machinery, as they result in significant amount of downtime to the machine and require expertise in controls and identification theory.

To address these issues, a rapid identification strategy was earlier developed by the authors [2]. This technique consists of collecting input and output data from the CNC while executing a short G-code experiment. In order to guarantee stability of the identified model, bounds are imposed on the closed loop pole locations. This results in the identification task to assume the form of a constrained minimization problem, which was solved using Lagrange Multipliers (LM) technique. LM required a nonlinear equation system with 8 unknowns to be solved, considering different constraint activation scenarios (i.e. Kuhn-Tucker switching conditions). Although successful, this was found to be a computationally lengthy approach, requiring the use of a specialized symbolic math solver.

In this paper, a more efficient technique is developed using a Genetic Algorithm (GA). The GA searches for the optimum drive parameters, which best produce the observed response, using trial and error based on the Natural Selection principle. The search is conducted in the domain of stable pole locations, which simplifies the solution significantly.

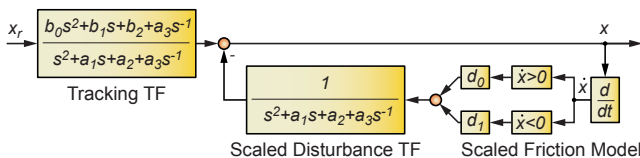


Figure 1: General representation of closed loop dynamics.

2 RAPID IDENTIFICATION PROBLEM

In machine tools in which the dynamics of different axes can be assumed to be decoupled, a single axis can be represented with the 3rd order model as shown in Figure 1. This model assumes that rigid body motion is dominant (i.e. flexible modes are not excited), and that the effect of nonlinearities like torque ripple and backlash are minor. The final position is characterized by equivalent tracking and disturbance transfer functions. When the machine is not cutting, the main disturbance originates from the guideway friction. The axis position can be estimated as:

$$\hat{x}_k = \alpha_i e_{i,k} - \alpha_1 \dot{x}_k - \alpha_2 \ddot{x}_k + \beta_0 x_{r,k} + \beta_1 \dot{x}_{r,k} + \beta_2 \ddot{x}_{r,k} - PV(\dot{x}_k) \cdot \delta^+ - NV(\dot{x}_k) \cdot \delta^- \quad (1)$$

where,

$$\left. \begin{aligned} \alpha_2 &= 1/a_2 & \alpha_1 &= a_1/a_2 & \alpha_i &= a_3/a_2 \\ \beta_2 &= b_0/a_2 & \beta_1 &= b_1/a_2 & \beta_0 &= b_2/a_2 \\ \delta^+ &= d_0/a_2 & \delta^- &= d_1/a_2 \end{aligned} \right\} \quad (2)$$

Above, $x_{r,k}$ and x_k are the commanded and actual positions. $e_{i,k}$ is the integrated tracking error. $\dot{x}_{r,k}$ and \dot{x}_k are commanded and actual velocity, $\ddot{x}_{r,k}$ and \ddot{x}_k are commanded and actual acceleration. δ^+ and δ^- are the Coulomb friction parameters in the positive and negative directions of travel. $PV(\cdot)$ and $NV(\cdot)$ are binary functions which assume values of "0" or "1" depending on which direction the axis is moving. When $\dot{x}_k > 0$, $PV = 1$, $NV = 0$. When $\dot{x}_k < 0$, $PV = 0$ and $NV = 1$. The virtual drive parameters are determined by executing a short G-code program, consisting of random movements within the drive's physical limits. The objective is to minimize the difference between estimated and actual position values:

$$\text{Objective : Minimize } f = \frac{1}{2} (\mathbf{Y} - \Phi\theta)^T (\mathbf{Y} - \Phi\theta) \quad (3)$$

Above, $\mathbf{Y} = [x_1 \ x_2 \ \dots \ x_N]^T$ and

$$\Phi = \begin{bmatrix} e_{i,1} & -\dot{x}_1 & -\ddot{x}_1 & x_{r,1} & \dot{x}_{r,1} & \ddot{x}_{r,1} & -PV(\dot{x}_1) & -NV(\dot{x}_1) \\ e_{i,2} & -\dot{x}_2 & -\ddot{x}_2 & x_{r,2} & \dot{x}_{r,2} & \ddot{x}_{r,2} & -PV(\dot{x}_2) & -NV(\dot{x}_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e_{i,N} & -\dot{x}_N & -\ddot{x}_N & x_{r,N} & \dot{x}_{r,N} & \ddot{x}_{r,N} & -PV(\dot{x}_N) & -NV(\dot{x}_N) \end{bmatrix}$$

$$\theta = [\alpha_i \ \alpha_1 \ \alpha_2 \ \beta_0 \ \beta_1 \ \beta_2 \ \delta^+ \ \delta^-]^T$$

Pole locations for the developed model can be factored as: $s^3 + a_1s^2 + a_2s + a_3 = (s + p)(s^2 + 2\zeta\omega_n s + \omega_n^2)$, where

$p_1 = -p$ and $p_{2,3} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$. Given (p, ω_n, ζ) , $(\alpha_i, \alpha_1, \alpha_2)$ can be determined using Equation 2 as,

$$\alpha_i = \frac{a_3}{a_2} = \frac{p\omega_n^2}{\Delta}, \alpha_1 = \frac{a_1}{a_2} = \frac{p + 2\zeta\omega_n}{\Delta}, \alpha_2 = \frac{1}{a_2} = \frac{1}{\Delta} \quad (4)$$

where $\Delta = \omega_n^2 + 2p\zeta\omega_n$. To guarantee stability of the identified model, the pole frequencies (p, ω_n) and damping ratio (ζ) need to be constrained with lower bounds. Also, in order to prevent the 3rd pole (p_3) from reaching zero, an upper bound on ζ is needed, hence:

$$\text{Constraints: } \left. \begin{array}{l} h_1 : p \geq p_{\min} \quad , \quad h_2 : \omega_n \geq \omega_{n,\min} \\ h_3 : \zeta \geq \zeta_{\min} \quad , \quad h_4 : \zeta \leq \zeta_{\max} \end{array} \right\} \quad (5)$$

In the solution developed in [2], the virtual drive parameters are calculated by solving θ from:

$$\mathbf{P}\theta + \begin{bmatrix} \frac{\partial p}{\partial \alpha_i} & \frac{\partial \omega_n}{\partial \alpha_i} & \frac{\partial \zeta}{\partial \alpha_i} \\ \frac{\partial p}{\partial \alpha_1} & \frac{\partial \omega_n}{\partial \alpha_1} & \frac{\partial \zeta}{\partial \alpha_1} \\ \frac{\partial p}{\partial \alpha_2} & \frac{\partial \omega_n}{\partial \alpha_2} & \frac{\partial \zeta}{\partial \alpha_2} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 - \lambda_4 \end{bmatrix} = \mathbf{R} \quad (6)$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are the Lagrange Multipliers and,

$$\mathbf{P} = \begin{bmatrix} p_{11} & \dots & p_{18} \\ \vdots & \ddots & \vdots \\ p_{81} & \dots & p_{88} \end{bmatrix} = \Phi^T \Phi, \quad \mathbf{R} = \begin{bmatrix} r_1 \\ \vdots \\ r_8 \end{bmatrix} = \Phi^T \mathbf{Y} \quad (7)$$

Estimation of θ from Equations 6 and 7 has a lengthy solution, involving the use of a specialized symbolic math toolbox. In this work, a more efficient technique has been developed using a Genetic Algorithm.

3 EVOLUTIONARY COMPUTATION APPROACH

The Genetic Algorithm (GA) identification technique is shown in Figure 2. It is similar to the structure in [3]. First part of the cycle is to start with a Parent Population of solution candidates. This population is used to produce a new generation in the Crossover phase, which inherits characteristics from parent pairs. Randomness is introduced by perturbing the new solutions in the Mutation phase. After ensuring compliance with the stability constraints [4], all solution candidates are evaluated for how well they minimize the objective. The best solutions are carried forward to spawn the next generation using a Selection process. This cycle is repeated until the solution pool converges to an optimal set of parameters.

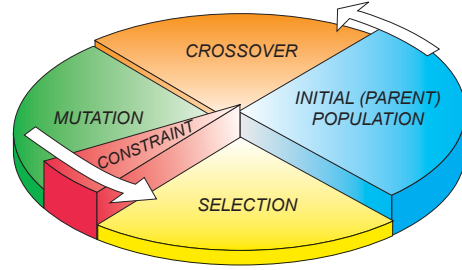


Figure 2: Iterative cycles of genetic algorithm technique.

In setting up the GA identification, the solution search is conducted directly in terms of the closed loop pole frequency (p, ω_n) and damping (ζ) values, as shown in Figure 3. Doing so simplifies incorporation of the stability constraints in Equation 5. The upper bounds for the pole frequencies $(p_{\max}, \omega_{n,\max})$ are set as half of the sampling frequency (i.e. Nyquist frequency) used in the data collection. A solution candidate is expressed as:

$$\mathbf{u}_i^t = \mathbf{u}_i^t(p_i^t, \omega_{n,i}^t, \zeta_i^t) \quad \text{where } \begin{cases} t : \text{Generation Number} \\ i : \text{Candidate Number} \end{cases} \quad (8)$$

Given the pole frequency and damping in $\mathbf{u}_i^t(p_i^t, \omega_{n,i}^t, \zeta_i^t)$, normalized denominator parameters $\theta_1 = [\alpha_i \ \alpha_1 \ \alpha_2]^T$ are determined using Equation 4. The remaining parameters $\theta_2 = [\beta_0 \ \beta_1 \ \beta_2 \ \delta^+ \ \delta^-]^T$ are computed by constructing a Least Squares sub-problem (i.e. $\Phi_2\theta_2 = \mathbf{Y} - \Phi_1\theta_1$). Defining $\mu_i^t(\mathbf{u}_i^t) = [1 \ \alpha_i \ \alpha_1 \ \alpha_2]^T$, the normalized drive parameters are solved as:

$$\theta_i^t = \begin{bmatrix} 0 & I_3 \\ \mathbf{R}^* & -\mathbf{P}^* \end{bmatrix} \mu_i^t, \quad \text{where } \mathbf{R}^* = \begin{bmatrix} p_{44} & \dots & p_{48} \\ \vdots & \ddots & \vdots \\ p_{84} & \dots & p_{88} \end{bmatrix}^{-1} \begin{bmatrix} r_4 \\ \vdots \\ r_8 \end{bmatrix} \quad (9)$$

$$\text{and } \mathbf{P}^* = \begin{bmatrix} p_{44} & \dots & p_{48} \\ \vdots & \ddots & \vdots \\ p_{84} & \dots & p_{88} \end{bmatrix}^{-1} \begin{bmatrix} p_{41} & p_{42} & p_{43} \\ \vdots & \vdots & \vdots \\ p_{81} & p_{82} & p_{83} \end{bmatrix}$$

For each solution candidate, the value of the objective function (i.e. fitness) is evaluated as:

$$f(\mathbf{u}_i^t) = \frac{1}{2}([\mu_i^t]^T \mathbf{A}\mu_i^t + \mathbf{B}\mu_i^t + \Gamma)$$

$$\text{where } \mathbf{A}_{4 \times 4} = \begin{bmatrix} 0 & \mathbf{R}^{*T} \\ I_3 & -\mathbf{P}^{*T} \end{bmatrix} \mathbf{P} \begin{bmatrix} 0 & I_3 \\ \mathbf{R}^* & -\mathbf{P}^* \end{bmatrix} \quad (10)$$

$$\mathbf{B}_{4 \times 1} = -2\mathbf{R}^T \begin{bmatrix} 0 & I_3 \\ \mathbf{R}^* & -\mathbf{P}^* \end{bmatrix}, \quad \Gamma_{1 \times 1} = \mathbf{Y}^T \mathbf{Y}$$

Above, matrices \mathbf{A} , \mathbf{B} , and Γ depend only on the experimental data and are computed prior to running the

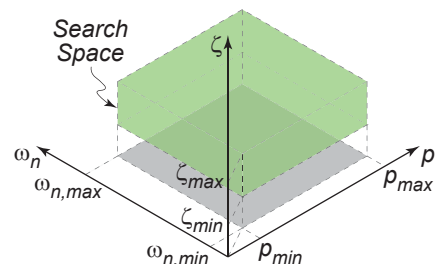


Figure 3: Solution search space for GA identification.

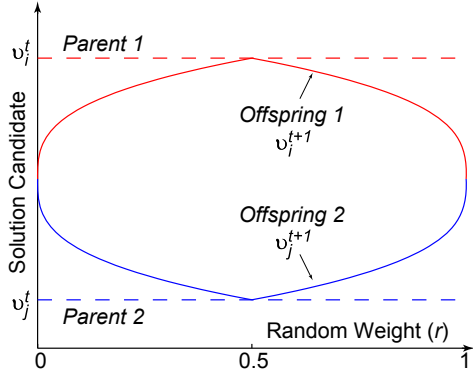


Figure 4: Simulated binary crossover (SBX) operation.

Genetic Algorithm. Given $\mathbf{u}_i^t(p_i^t, \omega_{n,i}^t, \zeta_i^t)$, the evaluation of the objective function in Equation 10 requires 32 multiplications, 22 additions and 1 division. If Equation 3 were used for the same purpose, the computational load would be $9N + 23$ multiplications, $9N + 16$ additions, and 1 division, where N is the number of identification samples (typically 1000). Hence, reorganizing the terms in the objective function has resulted in 2-3 orders of magnitude reduction in the computational load for evaluating the fitness of each candidate. This step was crucial in realizing a numerically efficient GA solution. In the following, the phases of GA identification are detailed.

3.1 Initial (Parent) Population

The initial population is generated with a uniform distribution within the highlighted search space in Figure 3. Following the first cycle, the new parent generation is determined through a selection process applied on mutated candidates of the earlier generation.

3.2 Crossover Operation

The crossover operation spawns the next generation of candidates by combining pairs from the current generation. It has a smoothening effect on the solution pool, which facilitates convergence. Every solution candidate (\mathbf{u}_i^t) is randomly matched with another one (\mathbf{u}_j^t). These are then used to produce two new candidates in the next generation: \mathbf{u}_i^{t+1} and \mathbf{u}_j^{t+1} . Since the solution candidates are made up of real numbers, the crossover needs to be conducted in the real domain instead of binary. For this purpose the Simulated Binary Crossover (SBX [5]) function was used, which has the expression,

$$\mathbf{u}_i^{t+1} = \frac{1+\alpha}{2}\mathbf{u}_i^t + \frac{1-\alpha}{2}\mathbf{u}_j^t, \mathbf{u}_j^{t+1} = \frac{1-\alpha}{2}\mathbf{u}_i^t + \frac{1+\alpha}{2}\mathbf{u}_j^t \quad (11)$$

where: $\alpha = \begin{cases} (2r)^{1/3} & , r < 0.5 \\ [2(1-r)]^{1/3} & , r \geq 0.5 \end{cases}$

Above, r is a random weighting factor which determines the closeness of the offspring to one of the parents. Possible outcomes of SBX are illustrated in Figure 4.

3.3 Mutation

Mutation is used to randomly perturb candidates in the new generation. This helps prevent the GA from converging to local optima. Each candidate is perturbed in the form:

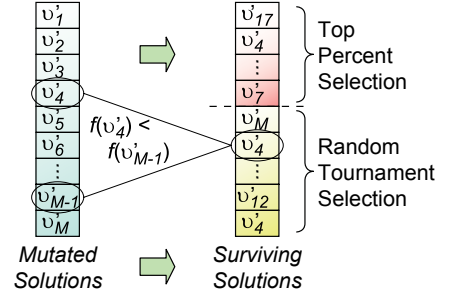


Figure 5: Selection of the best solution candidates.

$$[\mathbf{u}_i^{t+1}]' = \mathbf{u}_i^{t+1} + R_i \eta_i N_i(0,1) \quad (12)$$

Above, $N_i(0,1)$ is a random number generated from a normal distribution with zero mean and unit variance. This number is determined separately for each candidate. R_i is a scaling factor defined as,

$$R_i = \begin{cases} (\mathbf{u}_i^{t+1} - \mathbf{u}_{\min})/3 & , N_i(0,1) \leq 0 \\ (\mathbf{u}_{\max} - \mathbf{u}_i^{t+1})/3 & , N_i(0,1) > 0 \end{cases} \quad (13)$$

Considering that 99.73% of the values assumed by $N_i(0,1)$ will be in the range of -3 to +3, the scaling factor R_i maps these outcomes such that the perturbed solutions span the search space in Figure 3, bounded by points $\mathbf{u}_{\min}(p_{\min}, \omega_{\min}, \zeta_{\min})$ and $\mathbf{u}_{\max}(p_{\max}, \omega_{\max}, \zeta_{\max})$.

The η_i term represents a momentum step size, which is employed in evolutionary programming to facilitate larger perturbation in initial cycles. As iterations continue, η_i converges to a stationary random sequence around "1". Its mathematical expression is given as [6],

$$\eta_i = \exp(\tau' N_t(0,1) + \tau N_i(0,1)) \quad (14)$$

where: $\tau = 1/\sqrt{2\sqrt{t}}$, $\tau' = 1/\sqrt{2t}$

Above, $N_t(0,1)$ is generated from a normally distributed random sequence, independent of $N_i(0,1)$. $N_t(0,1)$ is updated only once for each generation. Essentially, $N_i(0,1)$ represents individual mutations which can affect each candidate separately while $N_t(0,1)$ represents an overall mutation affecting the whole generation. τ and τ' ensure that the step size is different in each iteration.

3.4 Constraint Checking

Each candidate is checked for compliance with the constraints. If any value for p , ω_n , or ζ is outside the valid search range, it is replaced by the closest bound.

3.5 Selection

Selection of successful candidates is carried out using hybrid approach which combines Top Percent and Random Tournament techniques, as shown in Figure 5. Top Percent ensures that solutions which yield the lowest value for the objective are included in the crossover process. Random Tournament arbitrarily matches pairs out of the mutated pool, choosing the superior candidate in each case. This approach may lead to the same candidate being chosen more than once, as seen for \mathbf{u}_4 .

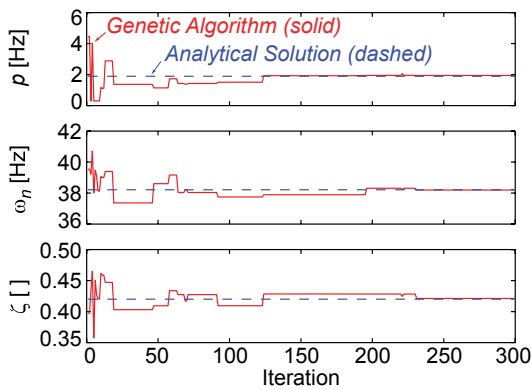


Figure 6: Convergence of GA estimated parameters.

4 SIMULATION AND EXPERIMENTAL RESULTS

The GA identification technique has been validated in simulations, by benchmarking its convergence with the analytical solution in [2]. The rapid identification technique was applied on a virtual drive model, based on the x axis dynamics of a Fadal VMC 2216. The position loop, in the virtual model, was closed using a PID controller.

The GA was configured to a population size of 4000 solution candidates in each generation. The search range was set to $\rho_{\min} = \omega_{n,\min} = 0.1$ Hz, $\rho_{\max} = \omega_{n,\max} = 800$ Hz,

$\zeta_{\min} = 0.2$, and $\zeta_{\max} = 2.0$. The selection operation was carried out by choosing the 5 best solutions (i.e. 0.125%) with the Top Percent approach, and the remaining with the Random Tournament approach. Convergence of the GA estimated parameters are shown in Figure 6. After the 300 iterations, the drive parameters are identified with 2-3% closeness to the analytical solution. This takes about 2.5 minutes on a Pentium IV PC running Matlab. The analytical solution, when all constraint activation cases need to be checked, takes about 15-20 on the same platform. Since the GA does not require a symbolic solver, it can be easily implemented in different programming languages such as C++. The GA identified model is fairly successful in predicting the tracking performance of the real drive, as seen in Figure 7.

The GA identification strategy was tested on a Deckel Maho 80P hi-dyn machining center (Figure 8). The motion data was captured using the oscilloscope function in the Heidenhain 430M controller. After identifying the x and y axes, the virtual drive model was validated by contouring circular and diamond toolpaths at 200 mm/sec. A sample result is shown in Figure 9, where the machine's contouring performance is closely predicted with the constructed virtual model, validating the practicality and effectiveness of the GA identification technique.

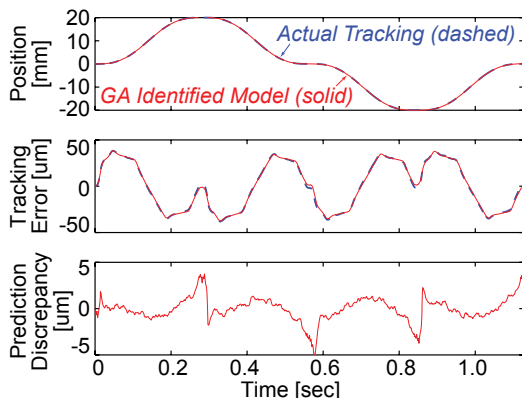


Figure 7: Predicted and actual tracking performance of virtual drive (simulation).



Figure 8: Machining center used in experiments.

5 CONCLUSIONS

In this paper, an efficient numerical solution has been presented for building virtual drive models of existing CNC machine tools. After running a short G-code test and collecting axis movement data, the closed loop drive model is obtained by running a Genetic Algorithm (GA). The GA converges quickly, while also guaranteeing stability of the identified drive model. The GA identification technique has been verified on experiments conducted on virtual and real machine tool drives.

6 ACKNOWLEDGMENT

This research is sponsored by NSERC and OCE granting agencies in Canada. Industrial support has been provided by Manufacturing Automation Laboratories Inc. and Mechworks Systems Inc.

7 REFERENCES

- [1] Erkorkmaz, K., Altintas, Y., Yeung, C.-H., 2006, Virtual Computer Numerical Control System, Annals of CIRP 55/1: 399-402.
- [2] Erkorkmaz, K., Wong, W., 2006, Rapid Identification Technique for VCNC Drives, International Journal of Machine Tools and Manufacture: Special Edition on High Performance Cutting (*in press*).
- [3] Fogel, D.B., Fogel, L.J., Atmar, J.W., 1991, Meta-Evolutionary Programming, Conference Record - Asilomar Conference on Circuits, Systems & Computers, 1: 540-545.
- [4] Fogel, D.B., 1991, System Identification through Simulated Evolution: A Machine Learning Approach to Modeling, Needham, Ginn Press.
- [5] Dek, K., Beyer H.-G., 2001, Self-Adaptive Genetic Algorithms with Simulated Binary Crossover, Evolutionary Computation, 9/2: 197-221.
- [6] Fogel, D.B., 2006, Evolutionary Computation: Toward A New Philosophy of Machine Intelligence 3rd Ed. IEEE Press.

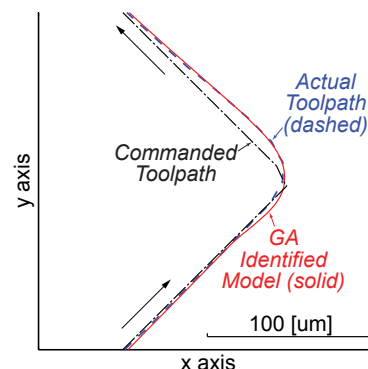


Figure 9: Experimental and predicted contouring.