Int. J. Engng Ed. Vol. 20, No. 4, pp. 594–611, 2004
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Integrated Modular Machine Tool Simulation for Education in Manufacturing Automation*

JINMING LIU and ROBERT G. LANDERS
Department of Mechanical and Aerospace Engineering, University of Missouri–Rolla, Rolla, Missouri 65409-0050, USA. E-mail: landersr@umr.edu

Simulation is increasingly being used as an educational tool in manufacturing classes, especially in the area of automation. However, no simulator currently exists that integrates all aspects of a machining workstation (i.e. physical machine, cutting process, and controller software and hardware) into one simulation in a modular manner. In this paper, the architecture of an integrated, modular machine tool simulator is introduced. This architecture provides the structure to create machine tool simulations with modular components that may be efficiently modified as required. Further, this architecture provides for an integrated machine tool simulation where the interactions between the machine, process, and controller may be efficiently investigated. A detailed two-axis lathe simulator is created from this architecture and examples are provided to present the performance and utility of this simulator as a tool for manufacturing automation education.

INTRODUCTION

SIMULATION is increasingly being used as an educational tool in manufacturing classes, especially in the area of automation. Simulation tools allow students to incorporate complex effects (e.g. nonlinearities such as Coulomb friction) into their analyses; effects that are often not accounted for during the initial design stage. However, most simulations are typically applied to one specific component. For example, the student may use a model of a servomotor to develop and simulate a controller to regulate the servomotor velocity. Currently, no simulator exists that allows students to efficiently integrate all aspects of a machining workstation (i.e. physical machine, cutting process, and controller software and hardware) into a single simulation, thus, providing a tool to analyze the complex interactions that occur between the machine, process, and controller.

There has been an abundance of work in the area of manufacturing simulation [e.g. 1–5] that has largely been focused at the system level (i.e. coordination of multiple machines). Simulation has also been applied to the process planning level. Chui and Wright [6] created Internet-based tools such that a remote user could plan a machining operation, simulate the performance, machine the part, and download measured process data. A web-based tool called OPENFRONT is currently being developed to allow users to efficiently reconfigure setup at the factory level [7]. Web-enabled virtual machining was also conducted in Qiu *et al.* [8] and Ong *et al.* [9]. Min *et al.* [10] augmented virtual factory simulations with the integration of real-time machine tool data. Stori *et al.* [11] presented a methodology for incorporating simulation feedback to fine-tune mechanistic process models to optimize machining process plans. For machine tools, simulation is often used to design individual system components (e.g. spindle, servomechanism control algorithm) or to analyze a specific process phenomenon (e.g. tool wear, chip formation). Very little, if any, simulation is performed of the complete machine tool.

Improvements in computing power and efficiency have led to marked advancements in machine tool simulation in the areas of servomechanisms, tool path verification, and metal cutting processes. Servomechanism simulations have been used in predicting servo drive performance and in designing controllers. The power of these simulations is that nonlinear effects such as stiction and backlash may be included [e.g. 12]. This information is useful in sizing drives for machine tool applications. Servomechanism drive simulation has also been used as a means of optimizing control laws and minimizing motion errors [13]. Machine tool path verification is the process of determining the path of an end effecter from a part program [14]. Through the inclusion of solid modeling techniques (e.g. boundary representation, constructive solid geometry) simulators have begun to add visualization of the actual machining process, illustrating the effect of a particular tool path on a part through the completion of a part program [15]. Huang and Oliver [16] built a simulation that shows tool paths and geometric errors. Simulations using solid modeling techniques

* Accepted 2 February 2004.

have also been extended to include predictions of material removal rates and provide an assessment of force, torque, and dimensional error [16, 17]. Simulation in the machine tool industry is, by and large, verification of part programs [e.g. Applicon Bravo NC, EdgeCAM, MTS CNC Simulator Turning, Sirius NC Verify and Machine Simulator, and I–DEAS SmartCAM]. These simulators do not account for servomechanism and structural dynamics, geometric and thermal errors, and metal cutting phenomena. Johnikin *et al.* [18] developed a comprehensive simulation of machine tool physical components. In this work, the simulation was developed in a modular manner such that it could be easily reconfigured.

Simulation has served as a means of predicting the process phenomena associated with various machining operations such as drilling, milling, and turning. Common process characteristics examined in simulation include cutting force, chatter, etc. Researchers such as Stephenson *et al.* [19] developed a methodology to translate generalized force models into models specific to particular machining processes. Using this methodology, generalized models based on the empirical equations for force or pressure normal and parallel to the rake angle of a tool are first created, and then forces specific to processes such as drilling or milling can be calculated using geometric transformations. In this manner, process models for different machining operations can be developed from a single database of cutting force data and subsequently simulated. The process phenomenon known as chatter has been simulated for many machining operations. Chatter is a regenerative vibration caused by an unstable interaction between the dynamics of the machine structure and the force process [20], and simulations of this phenomenon have been developed for processes such as turning [e.g. 21] and milling [e.g. 22]. Machining simulations have been conducted in an attempt to predict the onset and amplitude of chatter, and to simulate the efficacy of chatter suppression techniques such as spindle speed variation. Characteristics of the surface profile have also been addressed in milling processes [23] and simulations have been conducted to demonstrate improved methods of surface error prediction that take into account conditions where tool deflections significantly affect the chip load. Researchers have also explored the use of process simulation to determine optimal machining parameters such as feedrate and spindle speed for a given set of process criteria. Some simulators that have been created specifically for machining processes include EMSIM (end milling), FMSIM (face milling), DRSIM (drilling), and TAPSIM (tapping) [24] and CutPro (milling and structural testing) [25]. Fang [26] built a computer simulation and animation of a turning process that calculated variables such as cutting power and tool wear and animated the chip formation process.

Simulations that encompass entire machine tools (including control, machine dynamics, and process dynamics) have also been developed. This comprehensive approach was used in developing a simulation for a NC milling station that also included the effects of discretization in sample-data systems [27]. Chen *et al.* [28] created a simulation of a CNC lathe that included servomechanism and process components. But often simulations such as these are limited in that they can only provide information about one particular machine tool configuration. A comprehensive simulation of a turning operation has also been developed, with a limited level of reconfigurability, for a CNC lathe [29]. The simulator developed in this research presented a parameterized machine tool with servo-axis dynamics that could be augmented to include non-linearities such as friction and backlash, and a PID controller that could be extended with process control algorithms. This approach allowed for a variety of different simulation configurations. However, this simulator did not allow the user to easily integrate their own models into the simulator.

In this paper, the architecture of an integrated, modular machine tool simulator is introduced. A detailed two-axis lathe simulator is created from this architecture and examples are provided to present the performance and utility of this simulator as a tool for manufacturing automation education.

MACHINING SIMULATION ARCHITECTURE

This section presents an architecture for integrated modular machine tool simulation. Machining workstations can be divided into three major systems: physical machine, cutting process, and controller. The physical machine consists of components such as the part, cutting tool, structural components (e.g. base, column), spindles, linear axes, and rotary axes. The physical machine determines the relative motion (i.e. position and velocity) between the cutting tool and the part. The cutting process consists of the phenomena that occur due to the contact between the cutting tool and part (e.g. force generation, tool wear). The controller consists of the software algorithms that monitor and regulate the physical machine and cutting process and the hardware that interfaces the sensors and actuators with the computer processor. The general architecture is presented in Fig. 1. Note that some components may be decomposed into finer pieces when the architecture is utilized for a specific machining process. For example, a linear axis may be decomposed into the motor, leadscrew, table, and encoder.

In the simulation, the physical machine components, cutting process phenomena, and controller hardware components are modeled via differential (or difference) equations. In an actual machine tool, the controller is connected to the actuators

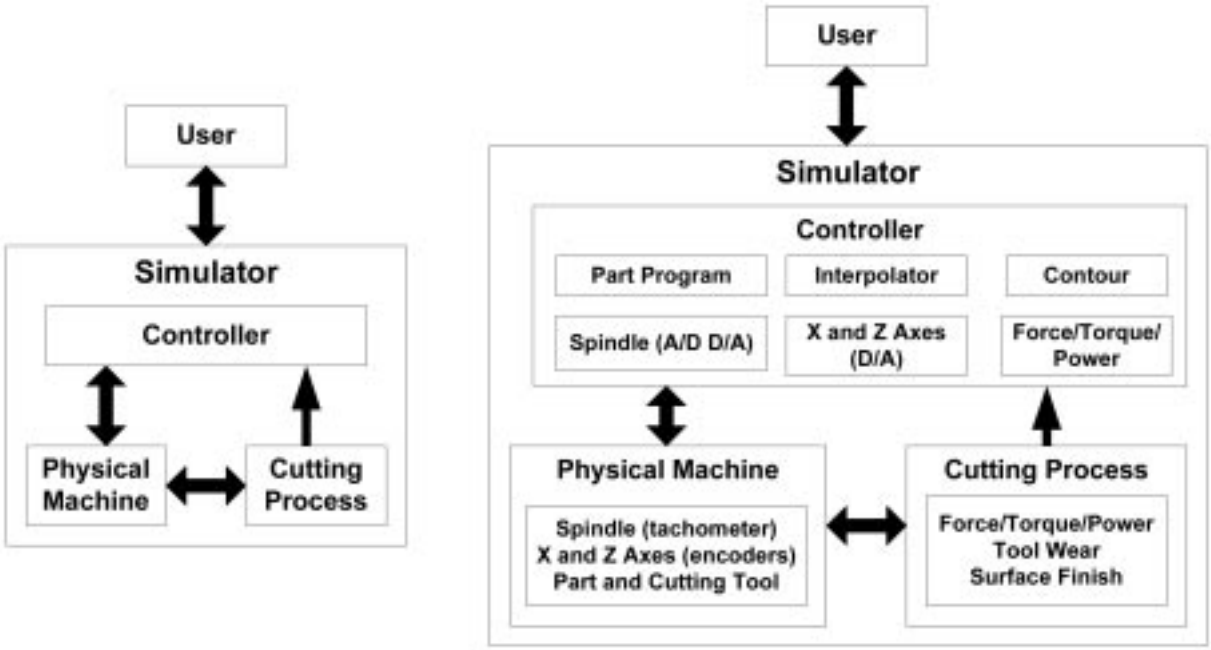

Fig. 1. Architecture of machine tool simulator—general (left) and two-axis lathe (right).

and sensors via wires and the machine-cutting process interactions are due to the physical contact between the cutting tool and the part. In the simulation, all components are connected via input and output data. The data flow is shown in Fig. 2 for a machine tool simulator that includes a machining force process and a force controller. The inputs are the part program and the process references, in this case a reference force. The interpolator determines the reference spindle speed and axis positions. The spindle and axis controllers utilize these references and the *measured* speed and positions, respectively, to calculate motor voltages. The *actual* speeds and positions determine process variables (e.g. feed, cutting speed, depth-of-cut). These variables determine the state of the process phenomena. The *measured* process phenomena are input to the process controllers that subsequently adjust the reference process variables in the interpolator. This cycle is repeated until the tool reference positions coincide with the final position of the last part program block and, thus, the simulation is complete.

In addition to the machine tool systems (i.e. machine, process, and control), an efficient simulation will require graphical user interfaces (GUIs)

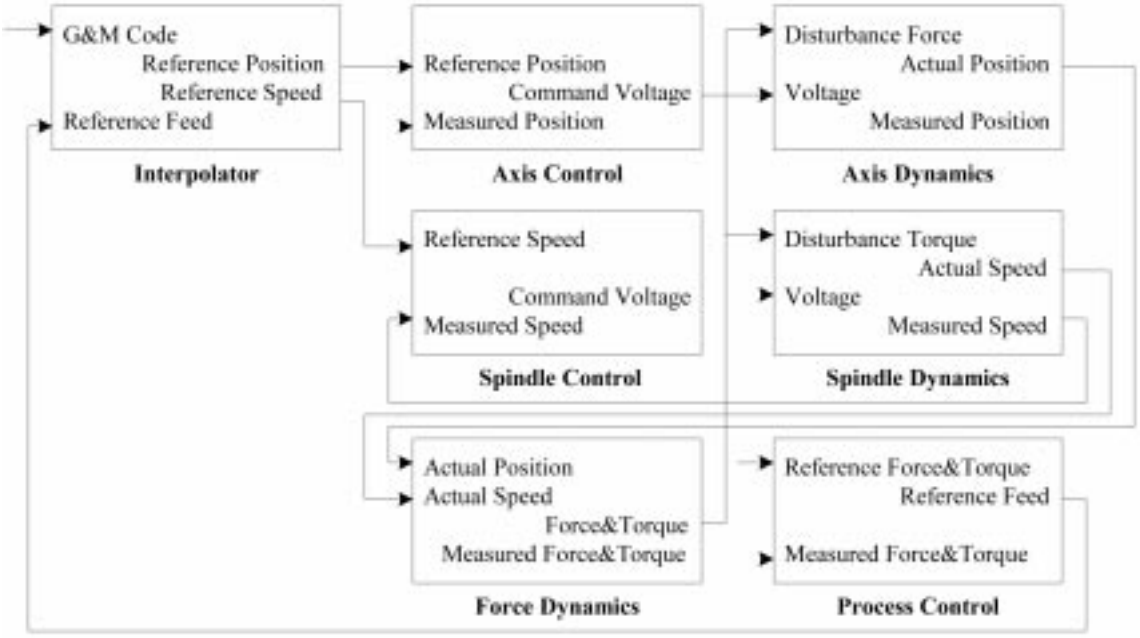

Fig. 2. Simulator program data flow.

that allow the user to easily develop, or reconfigure, the simulation and visualize the results. The simulation development will include creating models of each component, selecting model parameters, loading G&M code, selecting simulation parameters (e.g. sample period), etc. During the simulation, pertinent data are saved at each sample period and the simulation presents data to the user informing him/her of the simulation progress. After the simulation, the stored data may be graphed for analysis.

This simulation architecture provides a systematic means to integrate machine tool systems (i.e. machine, process and control) such that the complex interactions between these systems may be investigated. This architecture also provides a modular simulation structure such that machine tool simulations may be efficiently developed and modified as required.

TWO-AXIS LATHE SIMULATOR

The general machine tool architecture presented above is employed in this section to create a two-axis lathe simulator. The architecture is shown in Fig. 1. The physical machine consists of a spindle and two linear axes: one that moves along the longitudinal direction of the part (z-axis) and one that moves along the radial direction of the part (x-axis). The cutting process includes machining force, tool wear, and surface finish. The controller software algorithms include an interpolator, spindle and axis controllers, force/torque controller, and contour controller. The controller hardware components include digital to analog (D/A) converters that send command voltages to the spindle and axis motors and analog to digital (A/D) converters that receive information from the tachometer and force sensor. The simulation in this paper is developed in Matlab, a powerful and popular platform with excellent computational, programming, and visualization capabilities that is used widely in engineering schools. Each component model is a function and stored in an m-file. In this paper, fourth-order Runge–Kutta numerical integration is employed to simulate all differential equations. For user-defined models (see below), other numerical integration schemes, or the built-in Matlab routines, may be utilized. Several interfaces are built with graphical objects such as buttons, text fields, sliders, and images: objects with which most computer users are already familiar.

The simulator main window is shown in Fig. 3. There are buttons to select the part program, configure the spindle, axes, force process model, process controller, and the part and cutting tool, and to view the simulation results. The RESULTS button is inaccessible until a simulation is complete. Each component is described below. When the user first enters the simulator, default models and parameters are given for each component. As the user builds their own unique

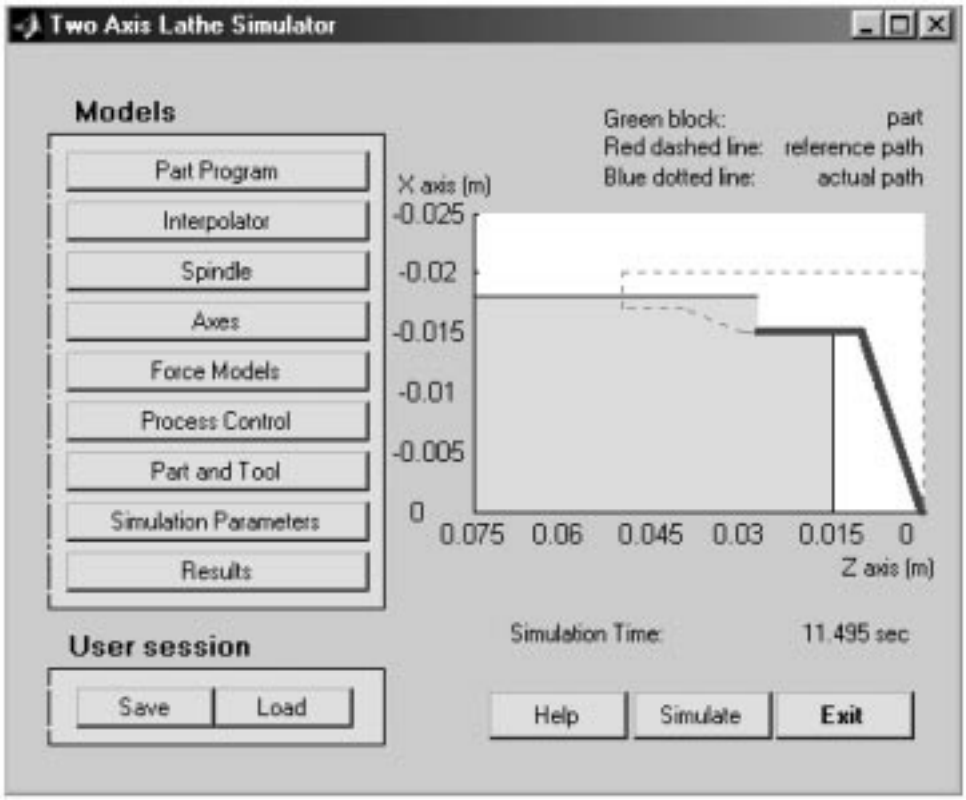

Fig. 3. Two-axis lathe simulator main window.

simulation, they may save it via the SAVE button and then access a saved simulation at any time using the LOAD button. Once the user has selected a part program, the SIMULATION button becomes accessible. During simulation, this button becomes a STOP button that may be used to halt the simulation at any time. Note that even if the simulation is halted prematurely, the data is still stored and is accessible to the user. The machining process is visualized on the main window during simulation. On a graph, there is a block that is half the raw stock, a dashed line that is the reference tool tip position, and a solid line that is the simulated tool tip position. During the simulation, the user may see how well the actual tool path is tracking the reference tool path and the shape of the final part as it is being processed. There is a progress bar that shows the simulation progress and the machining time is also displayed. The HELP button opens an extensive help file and the EXIT button may be selected to exit the simulation program.

Part program

Selecting the PART PROGRAM button creates a GUI (Fig. 4a) with three buttons: LOAD G&M PROGRAM, TAPER CUT, and STRAIGHT CUT. Selecting the LOAD G&M PROGRAM button allows the user to access a variety of text files that contain part program written in G&M code: machine code that specifies the overall motions and sequence of events to perform the machining process. Currently, acceptable codes include: G00, rapid traverse; G01, linear motion; G02 (G03), clockwise (counterclockwise) circular motion (both G02 and G03 may use IJK or R method); G18, circular motion plane in x–z plane; G90 (G91), absolute (incremental); G70 (G71), inches (millimeters); G96 (G97), constant surface (spindle) speed; M03, turn spindle on clockwise; M04, turn spindle on counterclockwise; M05, turn spindle off; X, Z, position of x and z-axes [m], respectively; F, feedrate [mm/min] or feed [mm]; S, spindle speed [rpm] or surface speed [rad/s]. When the user selects G&M code, a translator function is utilized to decompose the G&M code into vectors of information that drive the simulation. Selecting the TAPER CUT or STRAIGHT CUT buttons allows the user to select the geometry and process variables for a simple taper cut or straight cut, respectively. The TAPER CUT GUI is shown in Fig. 4b.

Interpolator

The function of an interpolator is to calculate the reference points at each sample period for the machine tool axes during all movements. The INTERPOLATOR button allows the user to select either a constant velocity or a constant acceleration interpolator. A constant velocity interpolator utilizes a constant reference feederate throughout the movement to calculate the reference positions. A constant acceleration interpolator utilizes a reference feedrate that linearly increases and decreases at the beginning and end, respectively, of the movement, and is constant in the middle, to calculate the reference positions. The acceleration/deceleration value may be set by the user. Note that the reference feederate is proportional to the product of the reference feed and spindle speed. These values may be adjusted by the process controllers described below. Therefore, the interpolators are programmed such that the reference feedrate, given in the part program, is recalculated when the reference feed and spindle speed are adjusted by the process controllers.

Spindle

The SPINDLE GUI (Fig. 5) allows the user to modify the physical spindle components (i.e. motor and spindle), tachometer, spindle controller, and A/D and D/A converters.

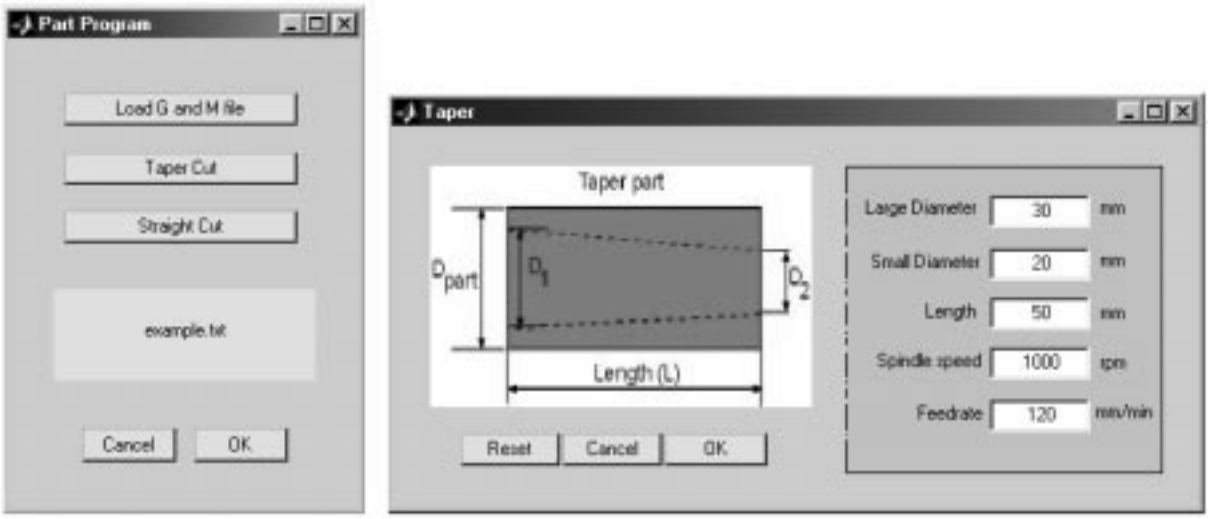

Fig. 4. (a) Part program GUI; (b) Part program taper cut GUI.

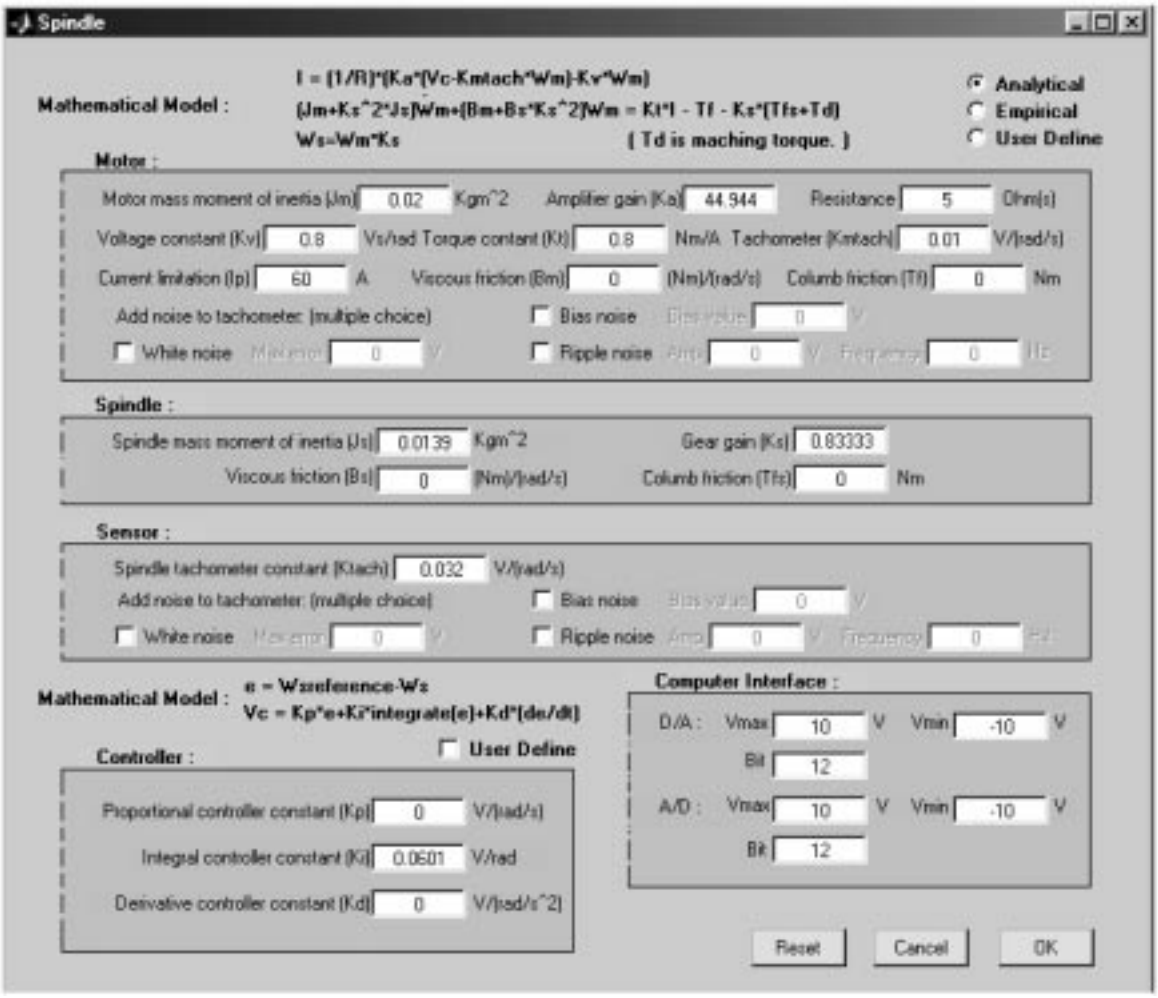

Fig. 5. Spindle GUI.

The user is provided with the following analytical spindle model:

$$I = \frac{1}{R}[K_a(V_c - K_{mtach}\omega_m) - K_v\omega_m] \tag{1}$$

$$(J_m + K_s^2 J_s)\dot{\omega}_m + (B_m + K_s^2 B_s)\omega_m = K_t I - T_f - K_s(T_{fs} + T_d) \tag{2}$$

$$\omega_s = K_s\omega_m \tag{3}$$

where I is the DC motor current [A], R is the DC motor resistance [Ω], K_a is the amplifier gain, V_c is the DC motor input voltage [V], K_{mtach} is the motor tachometer gain [V/(rad/s)], ω_m is the spindle angular velocity [rad/s], K_v is the motor voltage constant [V/(rad/s)], J_m is the DC motor mass moment of inertia [kgm^2], K_s is the spindle gear gain, J_s is the spindle mass moment of inertia [kgm^2], B_m is the motor viscous friction [kgm^2/s], B_s is the spindle viscous friction [kgm^2/s], K_t is the motor torque constant [Nm/A], T_f is the motor Coulomb friction torque [Nm], T_{sf} is the spindle Coulomb friction torque [Nm], T_d is the processing torque acting on the spindle [Nm], and ω_s is the spindle angular velocity [rad/s]. The motor current is saturated; therefore, if $I > I_{\max}$, then $I = I_{\max}$, and if $I < I_{\min}$, then $I = I_{\min}$ where $I_{\max}$ and $I_{\min}$ are the motor maximum and minimum current [A], respectively.

The user may also select an empirical spindle model of the form:

$$\tau\dot{\omega}_s + \omega_s = KV_c - K_d T_d \tag{4}$$

where τ is the time constant [s], K is the voltage gain [(rad/s)/V], and K_d is the disturbance torque gain [(rad/s)/Nm]. Coulomb friction torque is not incorporated into this model. Empirical models are created using experimental data and are required if analytical modeling is not feasible. The user may also create their own spindle model. The process of creating a user-defined model is provided below.

The spindle controller compares the *sensed* and reference spindle velocities, computes a control signal (i.e. voltage), and sends the signal to the spindle motor amplifier. The default controller structure is PID (i.e. proportional, integral and derivative) and is given by the following equation:

$$V_c = K_p e + K_i \int_0^t e(t)\,dt + K_{dc}\dot{e} \tag{5}$$

where V_c is the command voltage [V], $e = \omega_r - \omega_s$ [rad/s], ω_r is the reference spindle speed [rad/s], and K_p, K_i and K_{dc} are the controller's proportional, integral, and derivative gains, respectively. The controller also saturates the command voltage between the minimum and maximum voltages of the D/A converter (discussed below). This saturation provides for integral antiwindup. The user may also create their own spindle controller. A D/A converter transforms the control signal into a voltage to send to the spindle motor. The D/A converter saturates the voltage between $V_{\min}$ and $V_{\max}$ and quantizes the signal with a resolution of $(V_{\max} - V_{\min})/(2^n - 1)$ where n is the number of bits. The parameters $V_{\min}$, $V_{\max}$, and n may be selected by the user.

A tachometer transforms the *actual* spindle angular velocity values into a voltage. This voltage is sent to an A/D converter where it is subsequently transformed into the *measured* spindle angular velocity. Three types of noise can be injected into the tachometer voltage signal: white noise, bias, and ripple. The user may set the characteristics of the noise signal (e.g., amplitude, frequency) and may combine the different types. An A/D converter transforms the voltage signal into a digital signal in the computer processor. The A/D converter saturates the input voltage between $V_{\min}$ and $V_{\max}$ and quantizes the signal with a resolution of $(V_{\max} - V_{\min})/(2^n - 1)$. The signal is subsequently transformed into the units of *rad/s* via the tachometer gain. Again, the parameters $V_{\min}$, $V_{\max}$, and n may be selected by the user.

Axes

The AXES GUI allows the user to modify the physical linear axis components (i.e. motor, leadscrew, and table), encoder, linear axis controller, and D/A converter. The user is provided with the following analytical linear axis model:

$$I = \frac{1}{R}[K_a(V_c - K_{mtach}\omega_m) - K_v\omega_m] \tag{6}$$

$$J\dot{\omega}_m + B\omega_m = K_t I - T_f - K_l T_b - K_l p(T_{fa} + F_d) \tag{7}$$

$$v = \dot{x} = \omega_m K_l p \tag{8}$$

where $J = J_m + K_l^2 J_l + K_l^2 p^2 m$ [kgm^2],

$$B = B_m + K_l^2 B_l + B_l^2 p^2 B_a$$

J_l is the leadscrew mass moment of inertia [kgm^2], K_l is the leadscrew gear gain, [N/(m/s)], p is the leadscrew pitch [m/rad], m is the linear axis mass [kg], B_l is the leadscrew viscous friction [kgm^2/s], B_a is the axis viscous friction [N/(m/s)], T_b is the leadscrew Coulomb friction torque [Nm], T_{fa} is the axis Coulomb friction torque [Nm], F_d is the machining force [N], v is axis velocity [m/s], and x is axis position [m].

The user may also select an empirical linear axis model of the form:

$$\tau\dot{\nu} + \nu = \tau\ddot{x} + \dot{x} = KV_c - K_d F_d \tag{9}$$

where K is the voltage gain [(m/s)/V] and K_d is the disturbance torque gain [(m/s)/N]. Again, Coulomb friction torque is not incorporated into the empirical model. The user may also create their own linear axis model. The process of creating a user-defined model is provided below.

The linear axis controller compares the *sensed* and reference linear axis positions, computes a control signal (i.e. voltage), and sends the signal to the linear axis motor amplifier. The default controller structure is again PID and the controller saturates the command voltage between the minimum and maximum voltages of the D/A converter, providing for integral antiwindup. The user may also select the range and resolution of the D/A converter and create their own linear axis controller. A linear encoder is used to measure the linear axis position and the user may select its resolution. The GUI for the x-axis is shown in Fig. 6.

Contour control

The contour error is defined as the shortest distance between the desired path and the actual tool position. For two-axis linear contours, the contour error is:

$$\varepsilon = e_y \cos\left(\frac{v_x}{v_r}\right) - e_x \sin\left(\frac{v_y}{v_r}\right) \tag{10}$$

where e_x and e_y are the errors [m] of the x and y-axes, respectively, v_x and v_y are the velocities [m/s] of the x and y-axes, respectively, and v_r is desired velocity [m/s] in the direction of motion. The contour error of a circular arc is:

$$\varepsilon = \sqrt{(p_x - X_c)^2 + (p_y - Y_c)^2} - R \tag{11}$$

where p_x and p_y are the actual x and y-axis positions [m], respectively, X_c and Y_c are the x and y coordinates [m], respectively, of the arc center, and R is the arc radius [m].

In the AXES GUI, the user may select the CONTOUR CONTROL button to select a contour controller. In the CONTOUR CONTROL GUI, the user may select a cross-coupling controller (CCC) as described in Koren and Lo [30]. The architecture of the CCC is shown in Fig. 7. The *sensed* axial errors are used to calculate the *sensed* contour error that is input to the CCC, and the *actual* axial errors are used to determine the *actual* contour error that is displayed to the user. The output of the CCC is an offset control signal that is added to the individual axis control signals. A PID structure is used for the CCC and the user may adjust the gains.

Force models

The contact between the cutting tool and part generates significant machining forces. These forces create disturbance forces and torques that act on the linear axes and spindle, respectively. Excessive forces cause cutting tool failure, spindle stall, undesired structural deflections, etc. In the FORCE

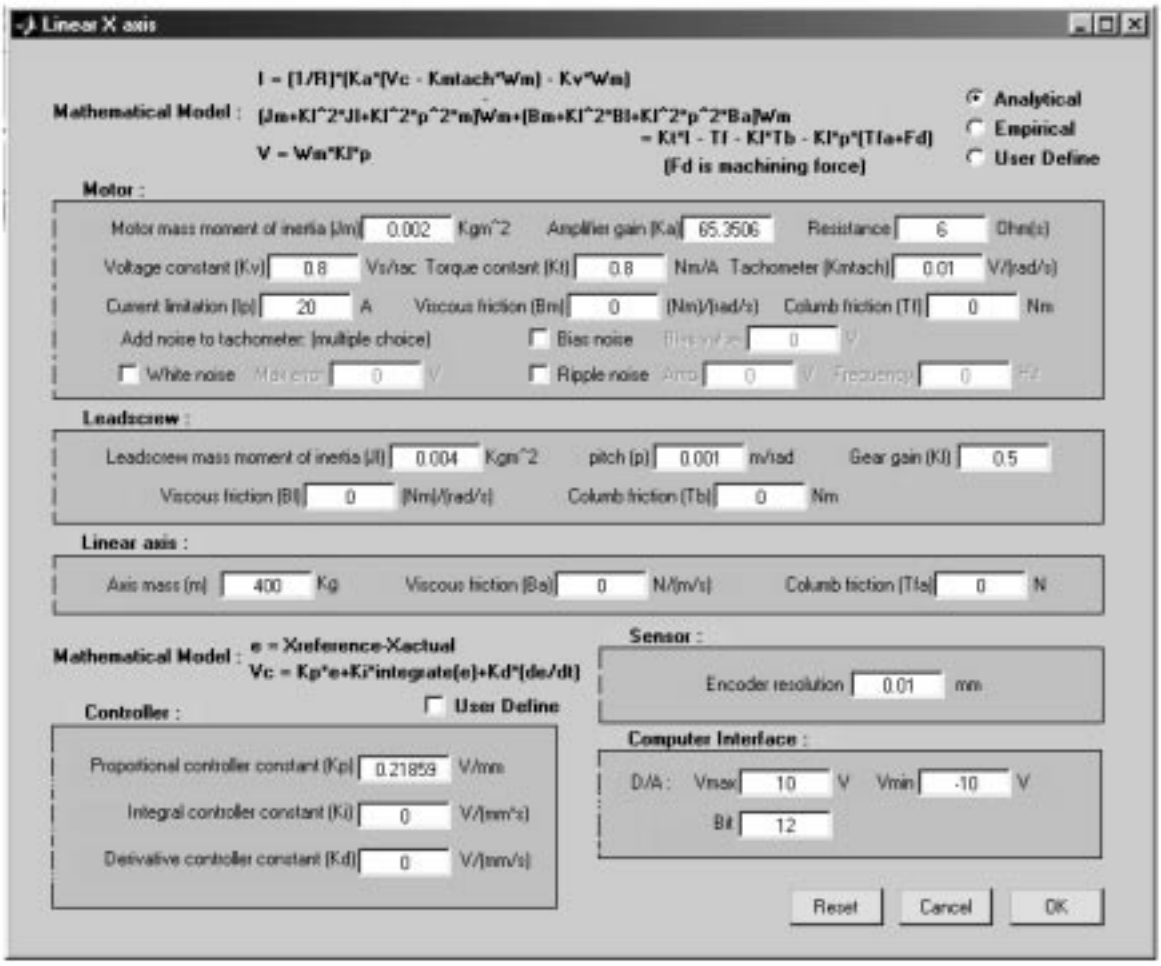

Fig. 6. X-axis GUI.

MODELS GUI, the user may select a linear model, a nonlinear model, or may create their own model. The nonlinear force process GUI is shown in Fig. 8. A similar GUI exists for the linear force process. A diagram of the cutting process is shown in Fig. 9.

For the linear model, the cutting and thrust forces acting on the insert, respectively, are modeled as:

$$F_C = P_C a \tag{12}$$

$$F_T = P_T a \tag{13}$$

where P_C is the cutting pressure [kN/mm^2], P_T is the thrust pressure [kN/mm^2], and a is the undeformed chip cross-sectional area [mm^2] which is modeled as $a = fd$ where f is the instantaneous feed [mm] and d is the depth-of-cut [mm]. The cutting and thrust pressures depend upon such factors as cutting fluid, part material, insert material, process parameters (i.e. depth-of-cut, cutting speed, and feed), etc. and are modeled empirically. In the nonlinear models, however, the cutting and thrust pressures have a nonlinear relationship to

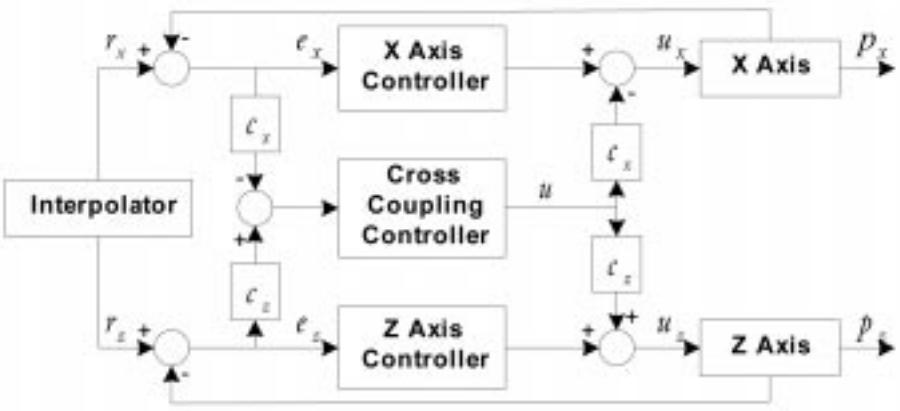

Fig. 7. Cross-coupling contour controller architecture.

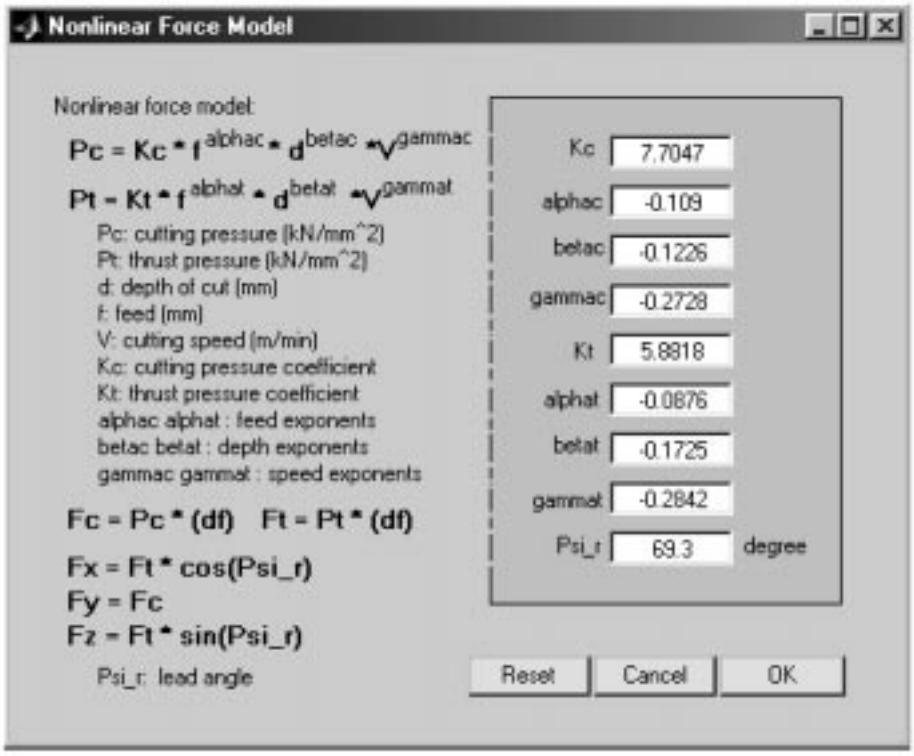

Fig. 8. Nonlinear force process GUI.

the process variables and these models, respectively, are:

$$P_C = K_C f^{\alpha_C} d^{\beta_C} V^{\gamma_C} \tag{14}$$

$$P_T = K_T f^{\alpha_T} d^{\beta_T} V^{\gamma_T} \tag{15}$$

The tangential, longitudinal, and radial forces acting on the insert, respectively, are:

$$F_{\tan} = F_C \tag{16}$$

$$F_{lon} = F_T \sin(\psi_r) \tag{17}$$

$$F_{rad} = F_T \cos(\psi_r) \tag{18}$$

The forces acting on the insert in the Cartesian coordinate system are:

$$F_x = F_{rad} \tag{19}$$

$$F_y = F_C \tag{20}$$

$$F_z = F_{lon} \tag{21}$$

The torque acting on the spindle is given by:

$$T = R_p F_C \tag{22}$$

where R_p is the part radius [m]. These force and torque signals act as disturbances on the spindle and linear axes and are displayed to the user. Also, *sensed* values are fed to the process control routines.

Process control

The PROCESS CONTROL GUI allows the user to implement process controllers to regulate the cutting force, radial force, spindle torque, or spindle power. Four standard process controllers may be implemented (nonlinear, linearization, log transform, and adaptive), or the user may create their own process controller. Figure 10 shows the GUI when the user has selected cutting force control using an adaptive controller. The force is measured via an analog dynamometer and the signal is sent to an A/D converter. The user may

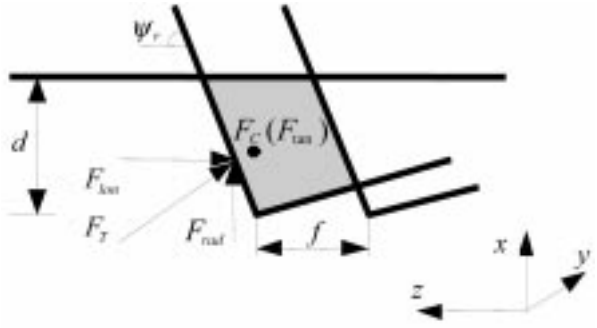

Fig. 9. Detailed schematic of material removal in turning operation (grey area: contact area).

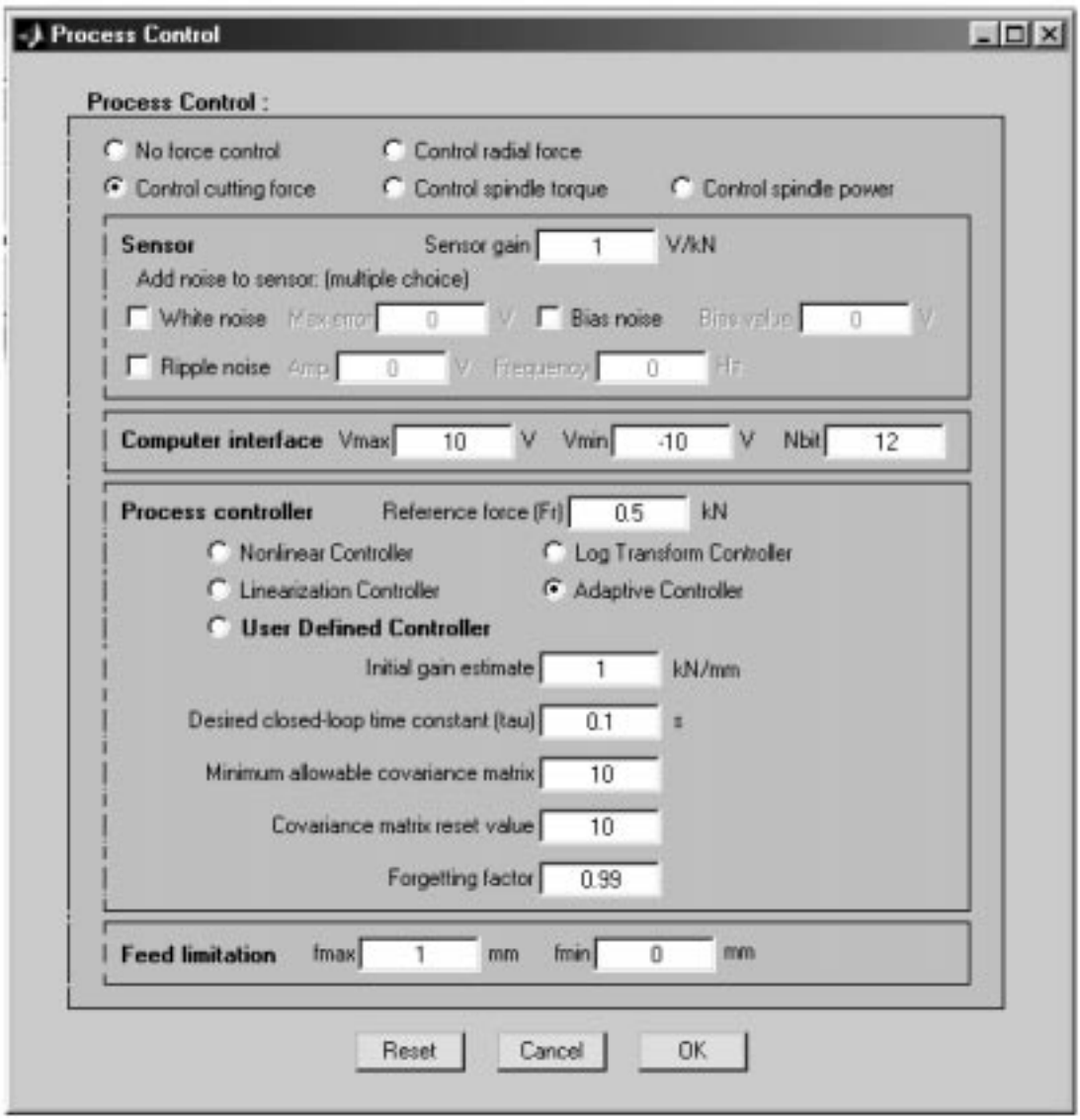

Fig. 10. Process control GUI with adaptive cutting force control selected.

select the sensor gain and the resolution, minimum voltage, and maximum voltage of the A/D converter. All process controllers adjust the feed to track the reference force/torque/power. The user selects this reference value and the maximum and minimum commanded feeds.

The nonlinear force controller [31] utilizes the following process model:

$$F = Kf^{\alpha}d^{\beta}V^{\gamma} \tag{25}$$

The following integral control law is used:

$$\dot{u}_C = K_i(F_r - F) \tag{26}$$

where $u_C = f^{\alpha}$ is the control variable, K_i is the controller gain, F_r is the reference force [kN] and F is the *sensed* force [kN]. The inverse of equation (25) is taken to determine the commanded feed. The user selects the estimated values of K, α, β, and γ [see equation (14)] and the controller gain is calculated given the desired closed-loop time constant, as specified by the user.

For the linearization force controller, standard linearization techniques are utilized to convert the nonlinear force process in equation (25) into an approximate linear system. The control law is:

$$\Delta\dot{f} = K_i[(F_r - F_0) - (F - F_0)] \tag{27}$$

$$f = \overline{f}_0 + \Delta f \tag{28}$$

where $\overline{f}_0$ and F_0 are nominal feed [mm] and force [kN], respectively. The user selects the estimated values of K, α, β, and γ and the controller gain is calculated given the desired closed-loop time constant, as specified by the user. The nominal feed is automatically calculated and the nominal force is taken as the reference force.

The log transform force control approach [32]

transforms a nonlinear force processes given by $F = \theta f^{\alpha}$ into:

$$\ln(F) = \ln(\theta) + \alpha \ln(f) \tag{29}$$

where the term $\ln(\theta)$ is viewed as an additive disturbance. Letting $\tilde{f} = \ln(f)$, $\tilde{F} = \ln(F)$, and $\tilde{F}_r = \ln(F_r)$, the control law is:

$$\dot{\tilde{f}} = K_i(\tilde{F}_r - \tilde{F}) \tag{30}$$

The user selects the estimated value of α and the controller gain is calculated given the desired closed-loop time constant, as specified by the user.

The most common methodology for machining process control is adaptive [33]. The model is:

$$F = \hat{\theta} f \tag{31}$$

The parameter $\hat{\theta}$ is an on-line estimate of the force process gain using the least squares method. The control law is:

$$\dot{f} = K_i(F_r - F) \tag{32}$$

The user selects the desired closed-loop pole location, initial parameter estimate, forgetting factor, covariance reset value, and minimum allowable covariance value.

Part and cutting tool

The PART AND TOOL GUI (Fig. 11) allows the user to adjust the dimensions of the raw stock and cutting tool, and adjust the tool wear model parameters. The default raw stock is a cylindrical part and the user may select the radius and length. The user may also specify the cutting tool corner radius r_{ε}. This parameter is used to calculate the average surface roughness using the following model [34]:

$$R_a = \frac{f^2}{32 r_{\varepsilon}} \tag{33}$$

The tool wear model is the classical Taylor tool wear model [35, 36]:

$$Vt^n = C \tag{34}$$

where V is the cutting velocity [m/min] and the parameters C and n are empirical constants. The user may select the part and cutting tool materials from a partial database to determine the parameters C and n. Alternatively, the user may define the parameters C and n or use their own tool wear model.

Simulation parameters

The SIMULATION PARAMETERS GUI allows the user to select the simulation parameters. The simulation can be set to a predetermined amount of time or it can be set to the completion of the part program. During the initial analysis stage, the user will typically run the simulation for only a short period of time to debug their models. During the final simulations, the user will run the simulation until the program is complete. The simulation sample period is selected by the user. The cutting animation refresh rate may be adjusted by the user,

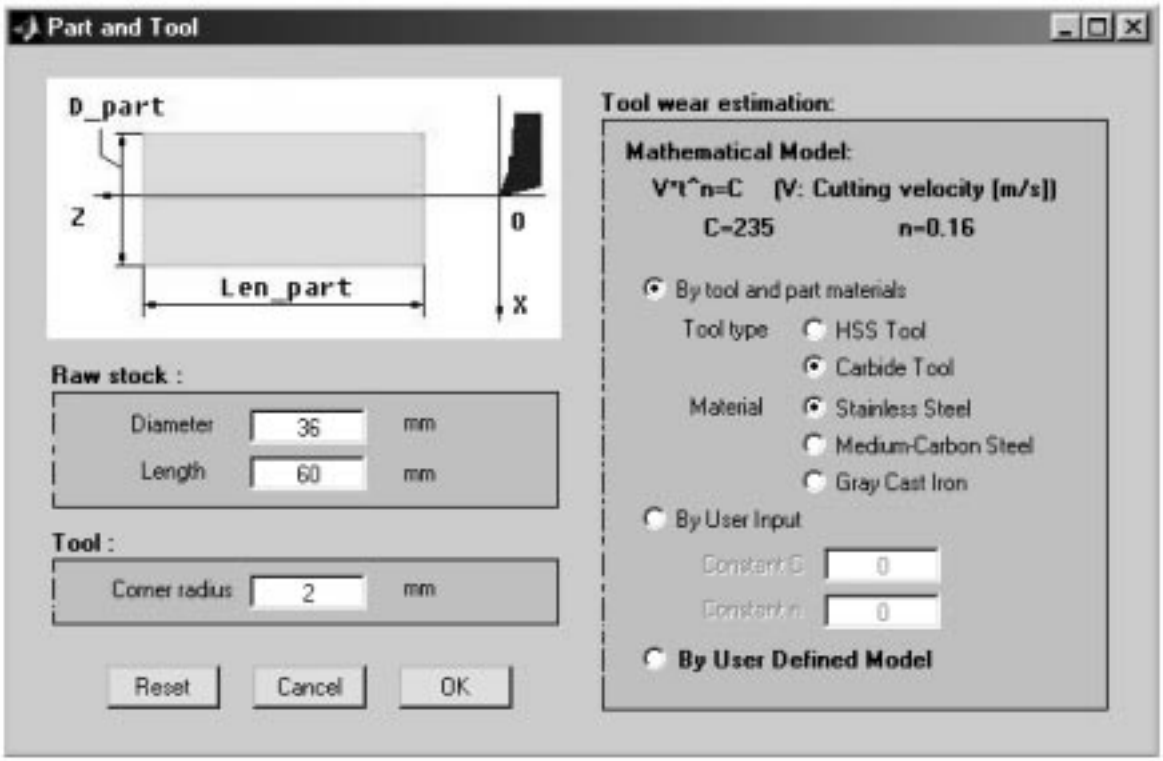

Fig. 11. Part and tool GUI.

or the animation can be switched off to reduce the computational cost.

Results

The RESULTS GUI allows the user to plot the following data: radial and longitudinal thrust forces, cutting force, spindle torque and power, feed, tool wear, surface finish, spindle velocity, velocity error, voltage and current, x and z-axis positions, velocities, position errors, voltages, and currents, contour error and CCC command voltage. The results may be accessed when the simulation is complete or even if the simulation is halted prematurely. All results are plotted versus time. Figure commands in Matlab (e.g., zoom, rotate) can be applied.

User-defined

Some of the simulator components may be set to user-defined, allowing the user to implement their own models. These components are: the spindle physical model, spindle controller, x and z-axis physical models, x and z-axis controllers, force process model, process controller and tool wear model.

The USER-DEFINED GUI describes the steps required to create and implement a user-defined component. First, the user must create a function in an m-file according to the directions in the README file. This file provides the inputs and outputs for each component, including units for each variable, as well as a list of the global variables. Next, the user must put this file in the appropriate folder, which is specified in the README file. In the specific USER-DEFINED GUI, the list of user-defined programs is displayed. To select a specific program, the user highlights the program and clicks the OK button. The simulator is automatically reconfigured such that the selected program will be implemented at run time. The USER-DEFINED GUI for the force process model is shown in Fig. 12. An example of a user-defined function is shown in Example 2 below.

MANUFACTURING AUTOMATION COURSE DESCRIPTION

The two-axis lathe simulator was utilized in the following course at the University of Missouri at Rolla: ME 355—Automation in Manufacturing. This is an introductory graduate course in the Mechanical Engineering department that is also offered as an elective to undergraduate students with senior standing. This course instructs students in the fundamentals of manufacturing automation at the equipment level: control system hardware components (DC motors, tachometers), servo-mechanism (spindles, linear axes) modeling and

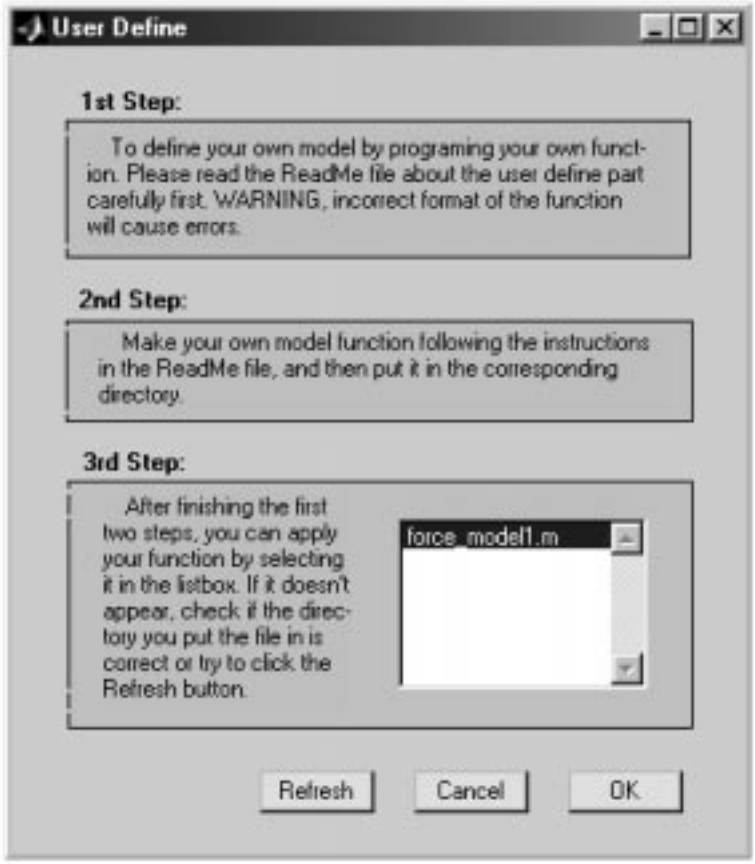

Fig. 12. User-defined GUI—force process model.

control, logic modeling, kinematic and geometric error modeling, and industrial robotics. During the course, the students complete several assignments where they model and control individual components without considering the complex interactions between the equipment components (dynamics and controllers) with the manufacturing process. It would be infeasible for each student, or even groups of students, to build a complex simulation in the course of one semester. Thus, the two-axis lathe simulator provides a means for the students to analyze the complex interactions between the individual equipment components, their controllers, the process plan given in the G&M code, the manufacturing process, and the process controllers. The two-axis lathe simulator was utilized in ME 355 as a course project, which is described in Example 3.

EXAMPLES AND DISCUSSION

Examples are provided to illustrate the utility of the two-axis lathe simulator for education in manufacturing automation.

Example 1: Servomechanism control

In the first example, the student will explore the differences, in terms of contour error, when using different types, and combinations, of servomechanism controllers and interpolators. Figure 13 shows the three-dimensional model and the dimensional drawing of the sample part. The diameter and length of the raw stock are 36 mm and 60 mm, respectively. The tool initial position is 15 mm away from the raw stock centerline along the z-axis. The CNC program is: N10 G90 G94 G71; N20 G00 X15.0 Z10.0; N30 M03 G01 Z30.0 F120 S1000; N40 X17.0 Z40.0 F180; N50 Z50.0; N60 X20.0; N70 M00 G00 Z0.0; N80 X0.0. Line N10 specifies the feederate commands, motions as incremental, and metric dimensional units. Line N20 rapidly moves the cutting tool 5 mm away from the part. Machining begins at line N30 where the spindle speed and feedrate are set to 1000 rpm and 120 mm/min, respectively. The feedrate is changed in line N40 to 180 mm/min. Machining ends at line N60. Lines N70 and N80 move the cutting tool back to the home position. No process control is utilized and the model and controller parameters for the spindle, axes, tool, force, and part are the default values on the GUIs. These model parameters for the spindle and axes are taken from Sandoval *et al.* [29]. The force model parameters were determined by fitting the force data for a steel part and a coated carbide tool to the linear and nonlinear force structures via least squares. The sample period is 5 ms and the simulation is run to completion.

Eight cases, shown in Table 1, are explored. The P axis controllers utilize the nominal controller gains provided in the simulator except as noted in Cases II and VI. For the cases that utilize a PI axis controller, the integral gain is 0.4 V/(mm sec). For the cases that utilize a P CCC, the proportional gain is 0.25 V/mm. For the cases that utilize constant acceleration interpolators, the acceleration value is provided in Table 1.

The results are presented in Table 1 and the contour errors are plotted in Fig. 14 during the time when the cutting tool and part are in contact, as denoted by the stars. The maximum of the absolute value of the contour error and the integral of the contour error squared are calculated while the cutting tool and part are in contact (i.e. from t_1 to t_2). Comparing Cases I and II, it is seen that increasing the servomechanism controller proportional gains decreased the contour error since the individual axial errors were reduced. Comparing Cases I and III, it is seen that the utilization of integral control in the servomechanism controllers decreased the contour error since the individual axial errors were reduced. Comparing Cases I and IV, it is seen that a constant acceleration interpolator will decrease the contour error since the transitions between line segments is smoother. However, the simulator shows that cycle time will significantly increase. Comparing Cases I and V, it is seen that a CCC will decrease the contour error. In Case VI, the servomechanism controllers have an integral term and their proportional gains are doubled, and a P CCC is utilized. It is seen that the maximum contour error is significantly decreased. This simulation also allows the student to see that the maximum contour error always occurs when the cutting tool makes a 90° motion as it begins the motion to leave the part.

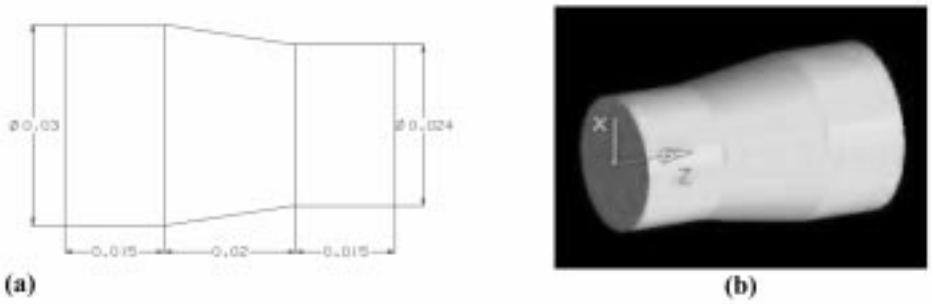

Fig. 13. Example part: (a) dimensions [m] and (b) three-dimensional model.

Table 1. Results for Example 1

Case	Axes controllers	Interpolator	Contour controller	max $\|\varepsilon\|$ (mm)	$\int_{t_1}^{t_2} \varepsilon^2 dt$ (mm^2s)	Cycle Time (s)
I	P	constant velocity	none	0.68494	0.10769	18.48
II	P*	constant velocity	none	0.33000	0.02189	18.48
III	PI	constant velocity	none	0.50210	0.08971	18.48
IV	P	constant acceleration ($0.01\ m/s^2$)	none	0.43540	0.07783	27.8
V	P	constant velocity	P	0.68508	0.04003	18.48
VI	P*I	constant velocity	P	0.21220	0.05213	18.48

The servomechanism proportional gains for cases II and VI are twice their nominal values.

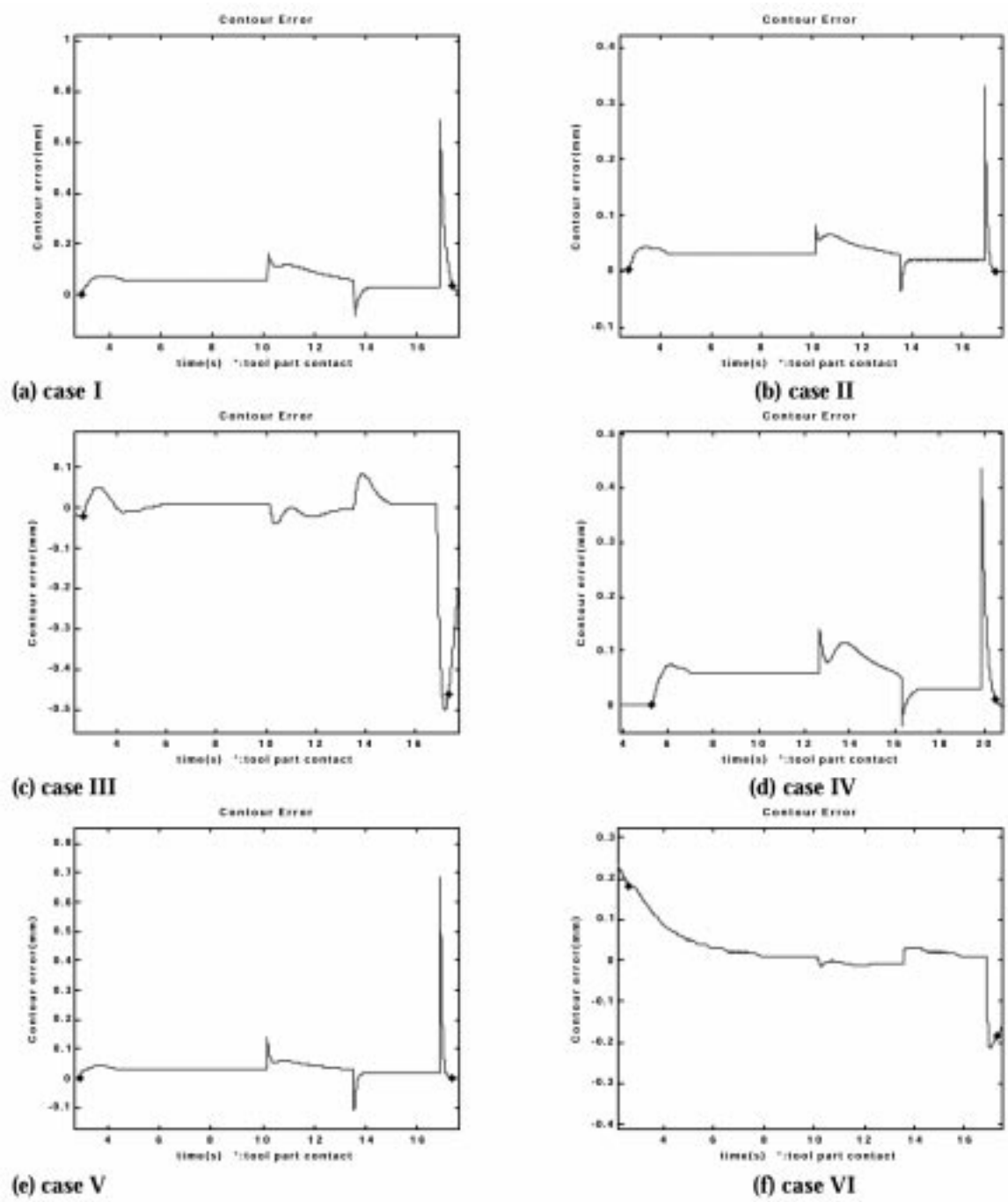

Fig. 14. Contour error results for different cases in Example 1.

Table 2. Results for Example 2

Case	Force controller	Max surface Finish (μm)	Cycle Time (s)
I	none	0.541	13.17
II	nonlinear	3.0	8.695
III	linearization	2.8	9.055
IV	log transform	unstable	unstable
V	adaptive	0.837	11.37

Example 2: Process control

In this example, the student will investigate different process control algorithms; namely, the nonlinear, linearization, log transform, and adaptive. The force models presented in [29] for a steel part and coated carbide tool are utilized. The cutting, longitudinal, and radial models, respectively, are:

$$F_C = 2864.63 d^{0.69} f^{0.75} - 4.28 V \quad (35)$$

$$F_{lon} = 1907.00 d^{0.74} f^{0.79} - 3.08 V \quad (36)$$

$$F_{rad} = 1464.28 d^{0.08} f^{1.09} + 0.02 V \quad (37)$$

Since these models do not fit the available nonlinear force model, the models in Equations (35)–(37) must be implemented via a user-defined module. The Matlab function is shown below.

```
function [fFdx,fFdz,fFdc]=
  force_model1(actf,fDpc,Vy)
% actf: actual feed [m]
% fDpc: actual depth-of-cut [m]
% Vy: actual cutting speed [m/min]
fFdc=2864.63*(fDpc*1000)^0.69*
  (actf*1000)^0.75-4.28*Vy;
  % cutting force [N]
fFdx=1464.28*(fDpc*1000)^0.08*
  (actf*1000)^1.09+0.02*Vy;
  % radial force [N]
fFdz=1907.00*(fDpc*1000)^0.74*
  (actf*1000)^0.79-3.08*Vy;
  % longitudinal force [N]
```

To implement the nonlinear, linearization, and log transform controllers, the cutting force model must be of the form given in Equation (25). Using the data provided in [29], the cutting force model utilized by the nonlinear, linearization, and log transform controllers is:

$$F_C = 7705 f^{0.891} d^{0.877} V^{-0.273} \quad (38)$$

where f and d are in units of mm and V is in units of km/min. Each controller has a closed-loop time constant of 0.4 s, a minimum feed of 0.0001 mm, and a maximum feed of 0.5 mm. The reference cutting force is 0.5 kN and a taper part is selected with a large diameter of 34 mm, a small diameter of 30 mm, and a length of 20 mm. In addition, the

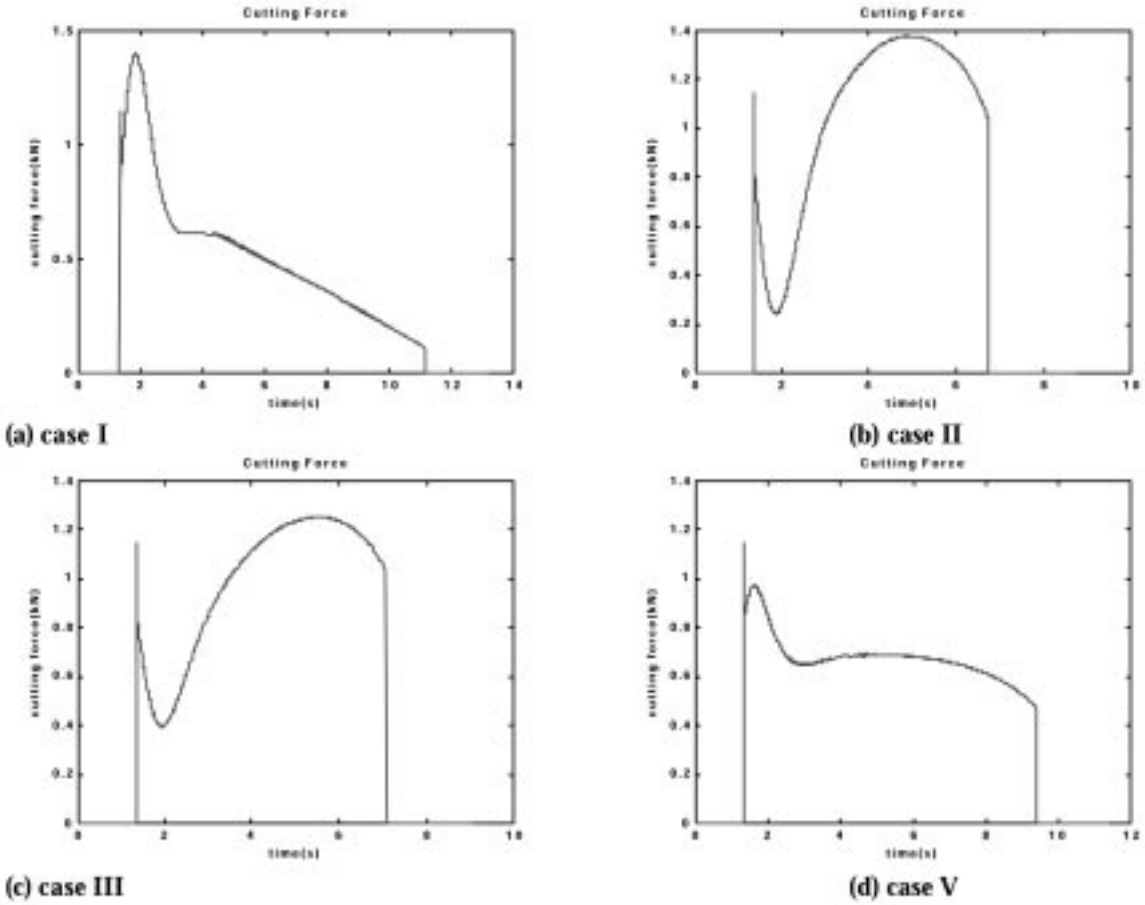

Fig. 15. Cutting force results for different cases in Example 2—Case IV was unstable.

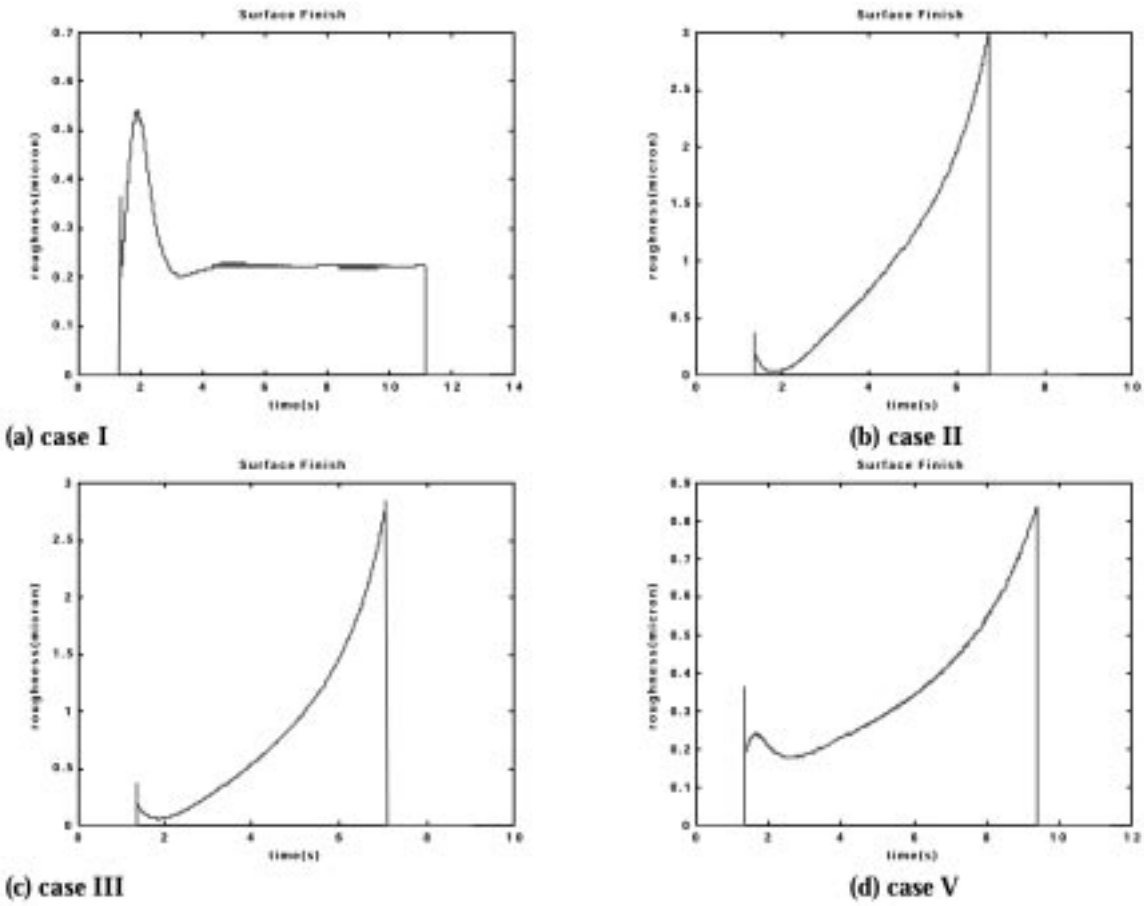

Fig. 16. Surface finish results for different cases in Example 2—Case IV was unstable.

adaptive controller has an initial gain estimate of 1, a minimum and reset covariance of 10, and a forgetting factor of 1.

As a baseline, the part is machined without force control. The results are summarized in Table 2 (cycle time and maximum surface roughness), Fig. 15 (cutting force), and Fig. 16 (surface roughness). By applying force control, the cycle time is reduced because the force controllers increase the feed to track the reference force. However, since the depth-of-cut is constantly decreasing when machining a taper part, the reference force is never tracked. Note that the log transform controller was unstable as it was not robust to the variation in the model structure. Since the surface roughness is proportional to the square of the feed, the surface roughness was greater for the cases where force control was implemented. This example allows the student to explore the tradeoffs between part quality and operation productivity.

Example 3: Lathe design

The lathe simulator was utilized in a course project to design the physical and control components of a lathe. The part described in Example 1 and the nonlinear force process described in Example 2 are used. The task for each group was to design a lathe while trying to minimize the following performance index:

$$PI = \frac{C}{50{,}000} + \frac{T}{5} \tag{39}$$

The parameter T is the total operation time in seconds and is found by running the simulation. The parameter C is the total cost in dollars and is given by:

$$C = 25{,}000 + C_s + C_x + C_z \tag{40}$$

The parameter C_s is the spindle cost: \$1000 for every 1000 rpm of maximum spindle velocity and \$1000 for every 10 rev/s^2 of maximum spindle acceleration. The minimum and maximum voltages of the A/D and D/A converters are -10 V and 10 V, respectively. Each bit in the A/D and D/A converters costs \$50. The tachometer has white noise. If the maximum amount is 1 V, there is no cost. If the maximum amount is 0 V, the cost is \$1000. The cost is inversely proportional if the maximum amount is between 0 V and 1 V. The parameter C_x is the x-axis cost: \$5000 for every 1 m/s of maximum velocity and \$5000 for every 1 m/s^2 of maximum spindle acceleration. The minimum and maximum voltages of the D/A converter are -10 V and 10 V, respectively. Each bit in the

Table 3. Results for Example 3

Group	Total Cost ($)	Cycle Time (s)	PI
I	109,350	24	6.987
II	105,437	6.85	3.479
III	46,191	10.79	3.082
IV	72,747	7.64	2.983

D/A converter costs $50. The encoder costs $500 if the resolution is 10 μm or greater, $1000 if the resolution is 1 μm, and is inversely proportional between these two resolutions. Below 1 μm, the cost increases at a rate of $100/0.1 μm with a minimum resolution of 0.1 μm. The parameter C_z is the cost of the z-axis and is calculated in the same manner as the cost of the x-axis. The controller gains could be adjusted at no cost. The thrust force in the x direction is limited to 600 N continuous and 1200 N peak. The thrust force in the z direction is limited to 1200 N continuous and 2400 N peak. The spindle torque is limited to 50 Nm continuous and 100 Nm peak. The spindle power is limited to 10 kW continuous and 20 kW peak. The feed is limited to 0.5 mm. The surface finish is limited to 1 μm. The contour error is limited to 0.5 mm.

Four groups participated in the course project. Their results are shown in Table 3. Interestingly, the designs of the second, third, and fourth groups had very similar PI values. A definite tradeoff between cost and cycle time can be seen in these three solutions: to decrease the cycle time, a faster and, hence, more expensive, machine is required. It is also interesting to note that the group that recorded the lowest PI had the middle cost and middle cycle time of these three groups. This project allowed the students to integrate much of the course material and investigate the tradeoffs between cost and performance. The students also learned how the various components worked within a complex system, how these components interacted with the machining process, and the effect of the part program. Further, the students learned that a real design requires many iterations and the utilization of good engineering judgment.

SUMMARY AND CONCLUSIONS

An architecture for creating integrated modular machine tool simulators for manufacturing automation education was presented in this paper. The architecture was applied to create a two-axis lathe simulator. The components of the simulator were described and examples were provided to illustrate the simulator's utility. The examples demonstrated how the simulator's integrated and modular characteristics allowed for the efficient development of machine tool simulations where the complex interactions of the physical machine, cutting process, and controller could be investigated. This simulator, and others built using the architecture in this paper, will be an invaluable tool for manufacturing automation education.

REFERENCES

1. G. Bengü, Computer-aided education and manufacturing systems with simulation and animation tools, *Int. J. Eng. Educ.*, **9**(6) 1993, pp. 484–494.
2. B. U. Nwoke and D. R. Nelson, An overview of computer simulation in manufacturing, *Ind. Engineering*, **25**, 1993, pp. 43–45.
3. E. E. Velazco, Simulation of manufacturing systems, *Int. J. Continuing Eng. Educ.*, **4**, 1994, pp. 80–92.
4. S. Narayanan, D. A. Bodner, U. Sreekanth, T. Govindaraj, L. F. McGinnis and C. M. Mitchell, Research in object-oriented manufacturing simulations: an assessment of the state of the art, *IIE Trans.*, **30**(9) 1998, pp. 795–810.
5. P. Klingstam and P. Gullander, Overview of simulation tools for computer-aided production engineering, *Computers in Industry*, **38**(2) 1999, pp. 173–186.
6. W. H. Chui and P. K. Wright, A WWW computer integrated manufacturing environment for rapid prototyping and education, *Int. J. Integrated Manufacturing*, **12**(1) 1999, pp. 54–60.
7. Z. J. Pasek, B.-K. Min, S. Marker and F. Husted, Web-enabled monitoring and control of manufacturing systems, *Int. Conf. Monitoring and Supervision in Manufacturing*, Warsaw, Poland, August 2001.
8. Z. M. Qiu, Y. P. Chen, Z. D. Zhou, S. K. Ong and A. Y. C. Nee, Multi-user NC machining simulation over the WWW, *Int. J. Advanced Manufacturing Technology*, **18**, 2001, pp. 1–6.
9. S. K. Ong, L. Jiang and A. Y. C. Nee, An Internet-based virtual CNC milling system, *Int. J. Advanced Manufacturing Technology*, **20**(1), 2002, pp. 20–30.
10. B.-K. Min, Z. Huang, Z. J. Pasek, D. Yip-Hoi, F. Husted and S. Marker, Real-time simulation for virtual manufacturing, *Int. J. Advanced Manufacturing Systems*, **1**(1) 2002, pp. 67–87.
11. J. A. Stori, P. K. Wright and C. King, Integration of process simulation in machining parameter optimization, *ASME J. Manufacturing Science and Engineering*, **121**(1) 1999, pp. 134–143.
12. G. W. Youkin, Modeling machine tool feed servo drives using simulation techniques to predict performance, *IEEE Trans. Industry Applications*, **27**(2) 1992, pp. 386–374.
13. Y. Koren, *Computer Control of Manufacturing Systems*, McGraw-Hill (1983).
14. N. L. Laurence, H. B. Voelcker, and A. A. G. Requicha, CNC machining: simulation, verification, programming, planning, communication and control, in *Manufacturing Processes, Machines and Systems*, 1st edn, 1986, pp. 243–251.

15. R. Uptal, 3-D object decomposition with extended Octree model and its application in geometric simulation of NC machining, *Robotics and Computer Integrated Manufacturing*, **14**(4) 1998, pp. 317–327.
16. Y. Huang and J. H. Oliver, Integrated simulation, error assessment, and tool path correction for five-axis NC milling, *J. Manufacturing Systems*, **14**(5) 1995, pp. 331–344.
17. A. D. Spence and Y. Altintas, A solid modeler based milling process simulation and planning system, *J. Engineering for Industry*, **116**, 1994, pp. 61–69.
18. V. P. Johnikin, B.-K. Min, R. G. Landers and A. G. Ulsoy, *Comprehensive, Modular Machine Tool Servomechanism Simulation*, ERC/RMS Technical Report, University of Michigan, Ann Arbor, Michigan (2001).
19. D. A. Stephenson and P. Bandyopadhyay, Process-independent force characterization for metal cutting simulation, *J. Engineering Materials and Technology*, **119**, 1997, pp. 86–94.
20. E. Usui, Progress of predictive theories in metal cutting, *JSME Int. J.*, **31**(2), 1998, pp. 363–369.
21. I. E. Minis, E. B. Marrab and I. O. Pandelids, Improved methods for the prediction of chatter in turning, *ASME J. Engineering for Industry*, **122**, 1990, pp. 12–35.
22. R. Radulescu, S. G. Kapoor and R. E. DeVor, An investigation of variable spindle speed face milling for tool-work structures with dynamics, Part 1: Simulation results, *ASME J. Manufacturing Science and Engineering*, **119**, 1997, pp. 266–272.
23. J. W. Sutherland and R. E. DeVor, An improved method for cutting force and surface error prediction in flexible end milling systems, *ASME J. Engineering for Industry*, **108**, 1986, pp. 269–279.
24. Machine Tool Agile Manufacturing Research Institute (MTAMRI), 2002. http://mtamri.me.uiuc.edu/
25. Manufacturing Automated Laboratories (MAL), 2001. http://www.malinc.com/
26. X. D. Fang, Application of computer aided animation of machining operations in support of a manufacturing course, *Int. J. Eng. Educ.*, **11**(6) 1995, pp. 435–440.
27. K. Martin and M. Ebrahimi, Modeling and simulation of the milling action, *Proc. Institute of Mechanical Engineers Part B, J. Engineering Manufacture*, **213**(B6) 1999, pp. 539–554.
28. S.-G. Chen, A. G. Ulsoy and Y. Koren, Error source diagnostics using a turning process simulator, *ASME J. Manufacturing Science and Engineering*, **120**(2), 1998, pp. 409–416.
29. J. E. Sandoval, R. G. Landers and A. G. Ulsoy, *Reconfigurable CNC Lathe Simulation System*, ERC/RMS Technical Report, University of Michigan, Ann Arbor, Michigan (2001).
30. Y. Koren and C. C. Lo, Advanced controllers for feed drives, *Annals of the CIRP*, **41**(2) 1992, pp. 689–698.
31. R. G. Landers and A. G. Ulsoy, Model-based machining force control, *ASME J. Dynamics Systems, Measurement, and Control*, **122**(3), 2001, pp. 521–527.
32. L. Harder, 1995, Cutting Force Control in Turning—Solutions and Possibilities, Ph.D. dissertation, Department of Materials Processing, Royal Institute of Technology, Stockholm.
33. A. G. Ulsoy and Y. Koren, 1989, Applications of adaptive control to machine tool process control, *IEEE Control Systems Magazine*, **9**(4), pp. 33–37.
34. G. Boothroyd and W. A. Knight, *Fundamentals of Machining and Machine Tools*, Marcel Dekker, New York (1989).
35. P. Wilkins, S. McLachlan, P. Shelton and M. Walker, Tool wear monitoring using the performance index method, in *32nd ISATA–Proc.*, Vienna Austria, June 1999, pp. 403–410.
36. L. A. Kendall, Tool wear and tool life, in *ASM International Metals Handbook: Machining*, Vol. 16, pp. 37–48 (1999).

Jinming Liu received his BS degree from Tsinghua University in 2001 in mechanical engineering. Mr Liu is currently a Masters student at the University of Missouri at Rolla in the Manufacturing Engineering department and serves as a research assistant under Dr. Landers. His research interests are in the areas of machining system modeling, control and simulation, control system design and applications, design and analysis of manufacturing systems.

Robert G. Landers received the BS degree (summa cum laude) from the University of Oklahoma in 1990, the ME degree from Carnegie Mellon University in 1992, and the Ph.D. degree from the University of Michigan at Ann Arbor in 1997, all in mechanical engineering. From 1997 to 2000 he was an assistant research scientist at the University of Michigan at Ann Arbor in the Mechanical Engineering department. Currently he is an assistant professor of mechanical engineering in the department of Mechanical and Aerospace Engineering at the University of Missouri at Rolla. Dr Landers is also a faculty member of the Manufacturing Engineering Education Program and a research investigator in the Intelligent Systems Center, both at the University of Missouri at Rolla. His research interests include the modeling, analysis, monitoring, and control of manufacturing processes, metal cutting processes, integrated design and control, supervisory control, and digital control applications. Dr. Landers has authored over fifty technical publications, including three book chapters. He is a member of the ASEE, ASME, IEEE, SME, and Pi Tau Sigma and chaired the ASME Dynamic Systems and Control Division's Manufacturing Systems Technical Panel from 1997 to 2000.