The 4th Joint International Conference on Multibody System Dynamics
May 29 – June 2, 2016, Montréal, Canada

Integration of machining simulation within a multibody framework: application to milling

Hoai Nam Huynh[1], Edouard Rivière-Lorphèvre[2] and Olivier Verlinden[1]

[1] *Theoretical Mechanics, Dynamics and Vibration Unit, Faculty of Engineering of Mons, Belgium, HoaiNam.HUYNH@umons.ac.be, Olivier.VERLINDEN@umons.ac.be*
[2] *Machine Design and Production Engineering Unit, Faculty of Engineering of Mons, Belgium, Edouard.RIVIERE@umons.ac.be*

ABSTRACT — *Machining simulation has become in a few years an essential asset for industries specialized in the manufacturing of high-added value parts. Workpieces are usually milled by machine-tools in order to get an accurate surface. However, machine-tools are very expensive and can operate only over a limited workspace. Therefore, some businesses shift towards robotic machining to produce their moderate-precision parts at a lower cost but all the phenomena involved in this technology aren't well understood. This paper lays the basis of a simulation environment dedicated to robotic machining by combining a milling routine (`DyStaMill`) and a multibody library (`EasyDyn`) which will be used further to simulate the behaviour of the robot. Some standard milling tests have been performed in order to validate the coupling of the milling and the multibody aspects. First, a mill moving at a fixed constant speed and machining a rectangular part is simulated to check the prediction of the cutting forces. Then, some degrees of freedom representing the vibrations of the mill along the x- and y-axes are added to the model through spring and damper systems in order to get closer to real machining conditions. Stability lobes charts are plotted and compared to previous results from the literature. A good agreement is observed among those results, demonstrating the validation of the coupling of the multibody library and the milling routine. Finally, the multibody model is applied to the case of a 3-DOF planar robot submitted to cutting forces.*

1 Introduction

Simulation of the machining process is a common method to predict and improve the interactions between the workpiece and the tool. Optimal cutting parameters can therefore be obtained quickly, which allows a potential cost reduction in the industries. There are at least three scales of modelling in machining for accurate simulations:

- the microscopic scale: the material removing is studied at the atomic level to completely model all physical phenomena such as the appearance of the dislocation during machining;
- the mesoscopic scale: a finite element model is used to describe the chip formation at the cutting edge of the tool. These models require complex constitutive models of material behaviour and are often restricted to the prediction of cutting forces for orthogonal cutting experiments [1];
- the macroscopic scale: whereas the first two scales imply longer duration simulations and complicated models, a mechanistic model allows a simplified modelling of the machining process. This global model considers machining as a force element acting on the system. The convoluted mill geometry is divided into slices along its revolving axis in order to compute the cutting forces with an analytical law [2, 3]. Consequently, it is this kind of model that will be adopted to develop the coupling between the machining and the multibody aspects leading to sufficiently accurate results to represent the real phenomena and reduced time simulations to replicate industrial operations.

The development of machining simulations has considerably evolved over the past decades thanks to experts' contributions in this field of investigation such as Altintas Y.. The latter simulated the machining of complete parts with a virtual environment by taking into account the frequency response function of the tool centre point

along with the mechanics and the dynamics of the milling process to compute the cutting forces [4, 5]. Moreover, Altintas Y. and Budak E. proposed a complete modelling of the milling process by computing analytically the cutting forces and by identifying the workpiece and tool deflections [6, 7, 8]. This analytical model wouldn't have been possible without an appropriate geometric description of the complex shapes of the tool resulting from a collaboration between Altintas Y. and Engin S. [9].

On the other hand, the simulation through the multibody approach is now a well mastered tool whose development was initiated in the 60's. The advantage of a multibody simulation is to retrieve informations about a mechanical system without having to physically build it resulting in a saving of time and money. A key step is the modelling of the mechanical system by expressing the spatial situation of each body with respect to configuration parameters defined according to a certain type of coordinates (commonly cartesian, relative or minimal coordinates). One can mention the famous paper of Hiller M. about minimal coordinates in which he solves the problem of formulating the equations of motion of minimal order for mechanical systems with pure differential equations [10]. Forces contributions are then included into the system before expressing the differential equations of motion by application of one of the theorems of mechanics: Newton-Euler, Lagrange,... These equations are lastly solved by a numerical algorithm: Runge-Kutta, Newmark... [11].

Nowadays, some machining operations as deburring, polishing, surfacing and drilling make a breakthrough in the field of robotics with the emergence of "Robotic machining". This fast-growing technology offers an economic and flexible alternative compared to standard machine tools [12]. Indeed, industrial robots present a great flexibility to access any locations of the workpiece and can deal with large workspaces [13]. However, they suffer from a lack of joints stiffness that influence the precision and the stability of the machining process. Consequently, the resulting vibrations prevent the use of machining robot to the manufacturing of high-precision parts. Since the identification of optimal cutting parameters in robotic machining is still empirical, the use of a multibody framework coupled with a milling routine is particularly welcome to simulate the dynamic behaviour of the machining process and improve the performance.

The present paper focuses on the coupling of a milling routine called `DyStaMill` with a multibody library called `EasyDyn` in order to establish the basis of a simulation environment devoted to robotic machining simulations. Despite the growing demand of machine parts through robotic machining, there is a lack of model combining the milling process with the robot structure to analyse their interactions. Denkena B. and Hollmann F. have nevertheless set up a model combining a rigid multibody dynamic system of a robot extended by joint elasticities and tilting effects with a simulation of material removing to evaluate the path deviation of the tool and the resulting surface [14].
After a brief description of the two simulation tools, a validation of the coupling is initiated first with a model of a mill moving and rotating at a constant speed submitted to cutting forces. Then, the mechanical model is extended to take into account the mill vibrations perpendicular to the revolving axis. The resulting stability lobes are compared to the corresponding literature cases. Ultimately, the validated coupled model is applied to a 3-DOF planar robot submitted to cutting forces.

2 Simulation tools

2.1 The `EasyDyn` library

`EasyDyn` is a `C++` library developed at the Theoretical Mechanics, Dynamics and Vibration Unit of the Faculty of Engineering of Mons (Belgium) by Verlinden O. et al. for teaching purposes [15]. The library is available for free on the internet and allows to easily simulate the motion of a multibody system from the expression of the homogeneous transformation matrices as well as the forces exerted on each body. The kinematics is expressed according to the minimal coordinates approach in terms of the chosen configuration parameters q and their time derivatives $\dot{q}$ and $\ddot{q}$. Consequently, the number of configuration parameters is exactly equivalent to the number of Degrees Of Freedom (DOF). The equations of motion are built in their residual form from the kinematics and the applied forces. The user has also the possibility to extend its system with the equations of controllers and hydraulic or electric actuators. They are then integrated by the application of the implicit Newmark formulas classically used to solve second-order differential equations of the form:

$$f(q,\dot{q},\ddot{q},t) = 0 \tag{1}$$

The EasyDyn package is accompanied by a program called EasyAnim allowing a 3D representation of the mechanical system motion. One can also mention that the EasyDyn library is not restricted to the simulation of simplistic 2D systems thanks to the symbolic tool CAGeM described later.

The EasyDyn library consists of four modules interacting with the main program:

♢ EasyDyn/vec: the vec module refers to the vector calculus by defining entities such as vectors, rotation matrices $\mathbf{R}$, inertia tensors $\mathbf{\Phi}_{G_i}$ and homogeneous transformation matrices $\mathbf{T}$ which are indispensable to compute the kinetics of the multibody system. In that sense, the situation of each body is localized by a frame j whose relative position $\{\underline{\mathbf{r}}_{j/i}\}_i$ and orientation $\mathbf{R}_{i,j}$ are gathered under the form of 4x4 homogeneous transformation matrices $\mathbf{T}_{i,j}$ with respect to a reference frame i (notation $\{\underline{\mathbf{a}}\}_j$ relates to the 3 coordinates of vector $\underline{\mathbf{a}}$ in coordinate system j):

$$\mathbf{T}_{i,j} = \begin{pmatrix} \mathbf{R}_{i,j} & \{\underline{\mathbf{r}}_{j/i}\}_i \\ 0\,0\,0 & 1 \end{pmatrix} \tag{2}$$

Thanks to the property $\mathbf{T}_{i,k} = \mathbf{T}_{i,j} \cdot \mathbf{T}_{j,k}$, that particular form allows the construction of kinematics along a chain by the product of successive matrices corresponding to elementary motions.

♢ EasyDyn/sim: the sim module is made up of routines that integrate second-order differential equations written in residual form with the implicit Newmark formulas. The main function of that module is called NewmarkOneStep() which performs an iterative procedure to achieve the convergence of all configuration parameters q and their time-derivatives ($\dot{q}$ and $\ddot{q}$) at each time step. For each iteration n, the procedure consists in a prediction and a correction. The prediction computes $\underline{\mathbf{F}}(\underline{\ddot{\mathbf{q}}}^{t+h})$ which are the equations of motion in terms of the acceleration at time $t+h$ after having replaced the positions q and the velocities $\dot{q}$ by their Newmark's integration formulas. Then, the prediction is directly followed by a correction using the classical Newton-Raphson formulas to compute the accelerations of each configuration parameter at time $t+h$, where the n^{th} estimation is calculated from the preceding one as:

$$\underline{\ddot{\mathbf{q}}}^{t+h,n} = \underline{\ddot{\mathbf{q}}}^{t+h,n-1} - \mathbf{J}^{-1} \cdot \underline{\mathbf{F}}(\underline{\ddot{\mathbf{q}}}^{t+h,n-1}) \tag{3}$$

with

- $\underline{\mathbf{F}}(\underline{\ddot{\mathbf{q}}}^{t+h})$: the equations of motion in terms of the acceleration at time $t+h$ after having replaced the positions q and the velocities $\dot{q}$ by their integration formulas;
- $\mathbf{J}$: the jacobian iteration matrix of the equations $\underline{\mathbf{F}}$ with respect to the unknowns $\ddot{q}^{t+h}$.

The positions q and velocities $\dot{q}$ are accordingly defined thanks to the Newmark integration formulas where β and γ are the Newmark parameters (classically $0.25 \leq \beta \leq 0.5$ and $0.5 \leq \gamma \leq 1$ to assure unconditional stability):

$$\begin{aligned} q^{t+h} &= q^t + h\dot{q}^t + (0.5-\beta)h^2\ddot{q}^t + \beta h^2 \ddot{q}^{t+h} \\ \dot{q}^{t+h} &= \dot{q}^t + (1-\gamma)h\ddot{q}^t + \gamma h \ddot{q}^{t+h} \end{aligned} \tag{4}$$

♢ EasyDyn/mbs: the mbs module is a frontend to the sim module which generates the residual form of the equations of motion from the kinematics and the wrench of the applied forces, both specified for each body. Whereas $\mathbf{J}$ is computed by a numerical derivation, the residuals of the equations of motion $F(\ddot{q})$ are obtained by calling a function called ComputeResidualMBS(). For a multibody system including n_B bodies and n_{cp} configuration parameters, the n_{cp} equations of motion written under their residual form by application of the d'Alembert's principle can be put into the following form:

$$F(\ddot{q}) \equiv \sum_{i=1}^{n_B} [d^{i,j} \cdot (R_i - m_i a_i) + \theta^{i,j} \cdot (M_{G_i} - \mathbf{\Phi}_{G_i}\dot{\omega}_i - \omega_i \times \mathbf{\Phi}_{G_i}\omega_i)] = 0 \quad j = 1, n_{cp} \tag{5}$$

with

- R_i and M_{G_i}: the resultant force an torque at the centre of gravity G_i applied on body i;
- $d^{i,j}$: the partial contributions of $\dot{q}_j$ in the velocity v_i of the centre of gravity of body i: $v_i = \sum_{j=1}^{n_{cp}} d^{i,j} \cdot \dot{q}_j$

  $\theta^{i,j}$: the partial contributions of $\dot{q}_j$ in the rotational velocity ω_i of body i: $\omega_i = \sum_{j=1}^{n_{cp}} \theta^{i,j} \cdot \dot{q}_j$

  The partial contributions can be computed by a numerical derivation from the kinematics.

◇ EasyDyn/visu: the visu module outputs 3D graphical scenes composed of simple shapes like boxes, frustums, triangles representing the multibody system, which can be read by the EasyAnim program.

The following paragraph synthesizes the classical steps of a multibody simulation under the EasyDyn framework. The basic steps are illustrated with the scheme in figure 1:

1. Historically, the user had to express the multibody simulation parameters into a MuPAD script. Data provided by the user concerned the number of rigid bodies n_B, the number of configuration parameters n_{cp}, the inertia data, the homogeneous transformation matrices defining the situation of each body with respect to the chosen configuration parameters and the initial conditions. This script was then used by a MuPAD routine called CAGeM (Computer Aided Generation of Motion) that writes symbolically the whole kinematics derived from the given homogeneous transformation matrices. The output resulted in a prewritten C++ application interacting further with the EasyDyn library: *Applic.cpp*. This process of generating the core of a C++ application was recently shifted towards the promising Python language. Now, the user specifies the multibody simulation parameters in a file called Applic.py and the C++ application *Applic.cpp* is built from the routine CAGeM.py using a Python library called SymPy for symbolic computation of kinematics. The generation of the Python file by the user and the C++ file by CAGeM.py is captured in figure 1 by large arrows on the left side;
2. the created C++ application *Applic.cpp* already gathers some EasyDyn functions prewritten on the basis of the data provided by the user through CAGeM.py: the function SetInertiaData() contains the mass m_i and inertia tensor $\mathbf{\Phi}_{G_i}$ of each body, the function ComputeMotion() contains the kinematics in terms of the configuration parameters, the function AddAppliedEfforts() allows the user to define forces R_i and torques M_{G_i} applied on any body and the function ComputeResidual() gives the possibility to add other second-order differential equations. This C++ application remains entirely editable by the user. These functions are called from the main() statement directly or mostly through the integration routine: NewmarkOneStep() (it is normally called by the NewmarkIntegration() routine which adapts the time-step);
3. the integration routine NewmarkOneStep() exchanges data with the EasyDyn library as illustrated in figure 1 via small arrows. NewmarkOneStep() that achieves the convergence of the configuration parameters needs the residual form of the equations of motion given by ComputeResidualMBS(). Clearly, the residual form of the equations of motion requires to update the kinematics gathered in the function ComputeMotion() with the prediction of the configuration parameters. Then, the applied forces (R_i) and torques (M_{G_i}) are described through AddAppliedEfforts() and introduced into the equations of motion by ComputeResidualMBS(), itself called by ComputeResidual() which may define some extra second-order differential equations.

The vec module interacts permanently with the application and the mbs/sim modules in order to deal with the mathematical entities such as vectors, etc... Interested readers may read contribution [16] for detailed examples and the complete description of the EasyDyn framework.

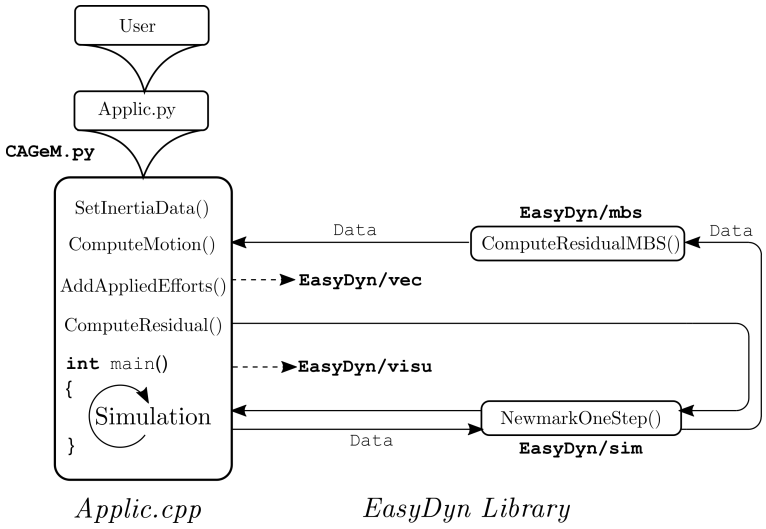


Fig. 1: Data flow of the EasyDyn library

2.2 The `DySTaMill` library

Until now, `DySTaMill` standing for "DYnamic STAbility of MILLing operations" was used as an autonomous homemade software for stability milling investigations. It was also developed at the Faculty of Engineering of Mons by Rivière E. during his PhD thesis about the stability of high-speed milling implementing a macroscopic model. The software was built to simulate operations generating $2\frac{1}{2}$D shapes such as slotting, contouring or pocket milling. The displacement of the tool lies in a plane perpendicular to its axis. This hypothesis is not restrictive for stability prediction due to the fact that the stiffness is significantly higher in that direction as compared to the perpendicular ones. The software is able to predict the cutting forces acting on the tool and the geometry of the machined surface. However, since only specific functions of the software are involved to perform the coupling with the multibody library, these functions were recently merged into a `C++` library for an upcoming public release.

Overall, the `DySTaMill` library is based on an analytical model computing the cutting forces thanks to the position of the tool in order to predict the stability lobes of the milling process. More precisely, the library comprises a cutting forces generator based on the tool position and an "eraser of matter" to update the geometry of the virtual workpiece. The widespread technique of tool discretization into elementary discs along the revolving axis is implemented to end up with a $2\frac{1}{2}$D model.

The `DySTaMill` library consists of nine modules interacting with the main program which simply calls the gathered functions. The following paragraph gives an overview of the principal functions of each module by emphasizing the underlying physics phenomena. The linear data flow of the library is illustrated in the next section introducing the coupling with the multibody library.

- ♢ `Dystamill/processing`: the `DySTaMill` library works somewhat in a similar way to the multibody library. The user provides first a textfile gathering all the informations about the milling simulation parameters: the geometric description of the tool (cylindrical, ball or bull nose end mill, number of teeth, helix angle,...), technological parameters (up- or down-milling, Axial Depth Of Cut (ADOC), the feed, the spindle speed,...), the material properties according to the chosen cutting force model (cutting force coefficients $\underline{\mathbf{K}}_c$, $\underline{\mathbf{K}}_e$) and the simulation parameters (number of revolutions of the spindle, number of steps per revolution). So far, the two most popular cutting force models are implemented to retrieve the orientation of all milling forces:

  1. the *Kienzle* model is particularly useful when the variation of the cutting parameters is large. The cutting forces are proportional to the removed chip section ($h \cdot da$) in which the chip thickness h is adorned with an exponent x_F:

  $$\underline{\mathbf{dF}} = \underline{\mathbf{K}} \cdot h^{x_F} \cdot da \tag{6}$$

  with
  - $\underline{\mathbf{dF}}$: infinitesimal cutting forces;
  - $\underline{\mathbf{K}}$: cutting force coefficients or 'specific pressure';
  - h: undeformed chip thickness;
  - da: infinitesimal axial length of the cutting edge.

  2. the *Altintas* model is frequently used when the cutting parameters are rather constant. The model comprises two contributions: the one concerning the removed material section and a contribution involving a friction component along the cutting edge:

  $$\underline{\mathbf{dF}} = \underline{\mathbf{K}}_e \cdot dS + \underline{\mathbf{K}}_c \cdot h \cdot db \tag{7}$$

  with
  - $\underline{\mathbf{K}}_c$: cutting force coefficients;
  - $\underline{\mathbf{K}}_e$: edge force cutting coefficients;
  - dS: local cutting edge length;
  - db: projected length of an infinitesimal cutting flute in the direction along the cutting velocity.

  The cutting force coefficients $\underline{\mathbf{K}}_c$ and $\underline{\mathbf{K}}_e$ need to be identified in advance since no valid correlation has been found yet with material properties such as the Young's modulus or the yield strength.
  The textfile is read by the `processing` module to feed the database of the library.

♢ `Dystamill/tools`: the `tool` module contains simple functions to open, read the textfile, initialize mathematical entities (vectors, matrices) and evaluate mathematical expressions.

♢ `Dystamill/common`: the `common` module includes common operations for cutting forces computation such as the orthogonal/oblique transformation to calculate them properly with respect to the tilted cutting edges of the tool. The procedure determines the adequate angles in order to transpose the local orthogonal cutting coefficients to 3D cutting coefficients.

♢ `Dystamill/varpitch`: the `varptich` module computes the geometrical initial values for a mill on the basis of a generalized mathematical model proposed by Engin S. and Altintas Y. in [9] in which they use helical flutes wrapped around a parametric envelope to model the cutting edges.

♢ `Dystamill/geometric`: the `geometric` module is employed in the determination of the chip thickness which is essential to compute the corresponding proportional cutting forces. Its main components help to find the intersection between the workpiece and the tool cutting edges.

♢ `Dystamill/efforts`: the `efforts` module aims to compute the global cutting forces $[F_x, F_y, F_z]$ projected in a reference frame. As previously announced, the cutting tool is discretized into superimposed disks of a thickness d_z along its revolving axis (figure 2).

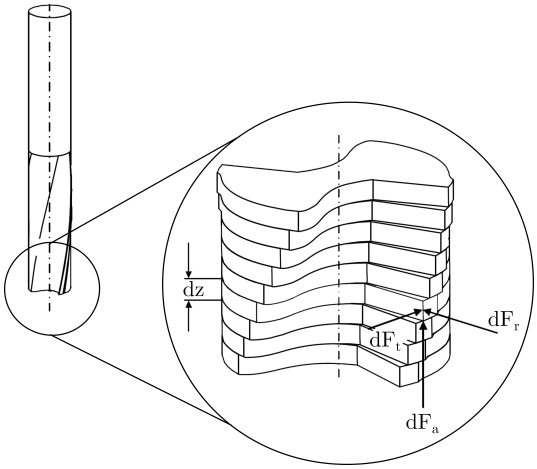


Fig. 2: Axial tool discretization

Once the geometry of the mill has been generated, one can compute the local cutting forces exerted on each disk representing the tool. Then, an integration is performed over the number of teeth n_z and over the number of discs n_e in order to sum the local cutting contributions and consequently obtain the global cutting forces projected into a global reference frame:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \sum_{e=1}^{n_e} \sum_{j=1}^{n_z} [\mathbf{B}] \begin{bmatrix} dF_t \\ dF_r \\ dF_a \end{bmatrix} \tag{8}$$

with

- $[\mathbf{B}]$: transformation matrix to project the local forces into the reference frame;
- dF_t, dF_r, dF_a: infinitesimal tangential, radial and axial forces applied on each tooth.

♢ `Dystamill/surface`: the `surface` module aims to update the material removal at the workpiece level. The workpiece surface is made of linked segments implemented through a double chain list. The update of the surface is the part of the simulation which is the most time-consuming because at each time step, the module has to determine if the tool edges intersect the segments constituting the surface with a binary search algorithm. Improvements were introduced as much as possible to accelerate the procedure, for example by detecting whether the tool was completely outside the workpiece.

♢ `Dystamill/linked`: the `linked` module gathers the whole double chain list constituting the workpiece surface. The advantage of using a double chain list is that each point of the list knows his predecessor and successor which improves the efficiency of the binary intersection search algorithm.

♢ `Dystamill/bezier`: the `bezier` module helps to determine the intersection with the surface considering a spline interpolated through the two previous positions of the mill cutting edges. More informations may be found in [17] about the model to generate the milled surface.

Interested readers may read contribution [18] for theoretical phenomena completion about milling simulation using `DySta Mill` as well as their variables allocation.

3 Coupling of EasyDyn and DyStaMill libraries

The coupling of both multibody and milling libraries is discussed by describing the data flow from the user point of view with a schematic block diagram depicted in figure 3. The user first describes the mechanical system giving the situation of the mill with respect to the chosen configuration parameters (since the studied systems are rheonomic by imposing the mill feed, the configuration parameters representing the longitudinal tool vibrations will be classically defined with respect to that constant velocity). The Python script CAGeM generates the C++ application which will interact with the EasyDyn library. Concurrently, the milling simulation parameters are set in the DyStaMill textfile. The DyStaMill functions are then incorporated into the C++ script generated by CAGeM.

The interaction of the two libraries happens after the prediction of the configuration parameters at time $t+h$. The prediction of the mill position is used to determine what amount of material will be removed at time $t+h$ and therefore compute the chip thickness with the so-called function compute_chip_thickness(). Consecutively, DyStaMill deduces the global cutting forces at time $t+h$ in the reference frame. These cutting forces are reinjected into the EasyDyn function AddAppliedEfforts() to be treated as applied forces exerted on the multibody system. One may really see the DyStaMill contribution as a force element for the mechanical system. Once the configuration parameters prediction and the cutting forces determined, the correction part of NewmarkOneStep() takes place as usual to refine the predictions of q, $\dot{q}$ and $\ddot{q}$. As soon as the iterative process has finished, the final position of the mill at time $t+h$ is exploited to update the surface of the workpiece through the double chain list of DyStaMill. It is necessary to refresh the workpiece geometry to properly compute the next cutting edges intersections. The interactions between the two libraries are represented with dashed lines in figure 3.

At the end of the milling simulation, an animation file can be read through the EasyAnim program showing a moving and revolving mill (for the moment, the animation does not allow an update of the workpiece geometry). Figure 3 below displays as well an example of the resulting tool vibrations during a milling instability.

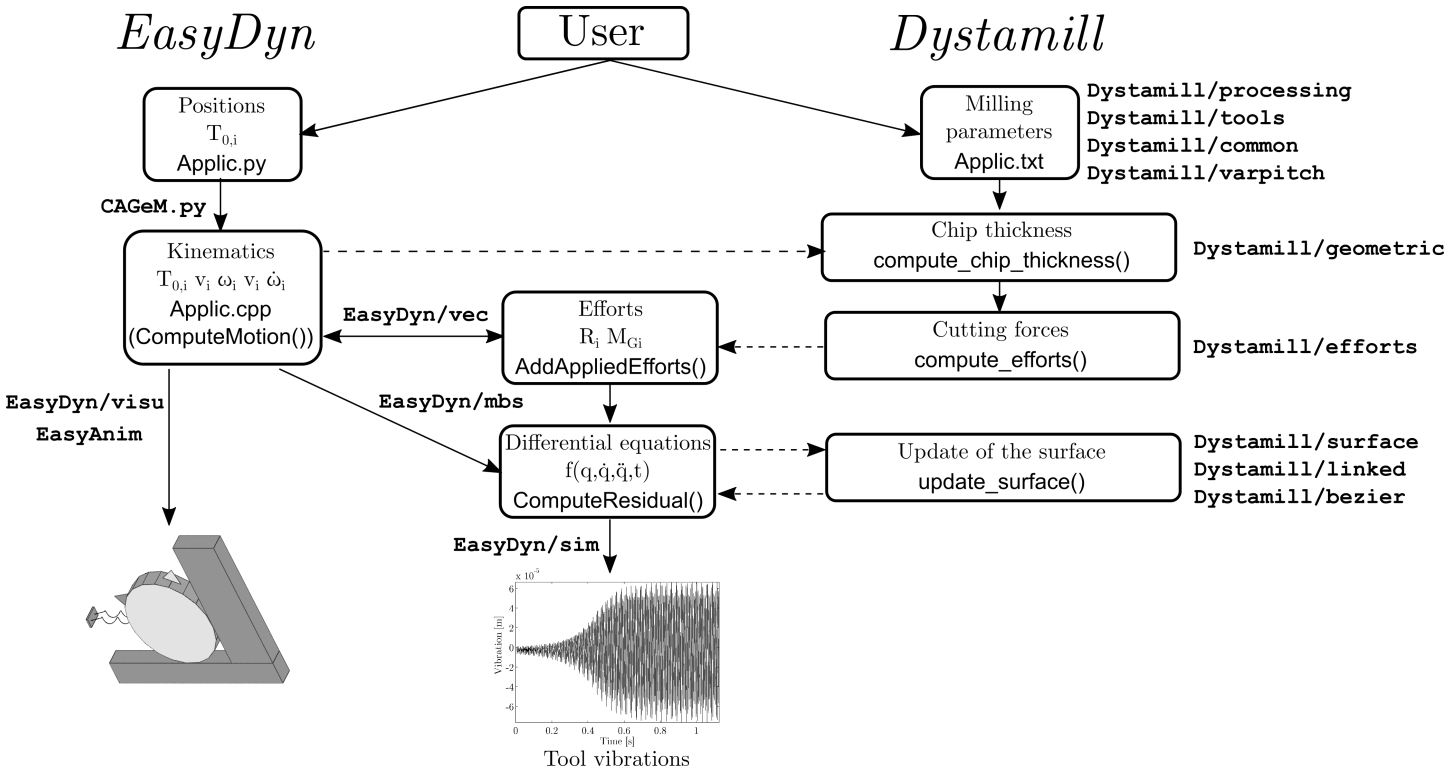


Fig. 3: Data flow when combining the C++ EasyDyn and DyStaMill libraries

4 Validation of the multibody-milling coupling

4.1 Cutting forces validation

After coupling the two libraries, the first validation step referred to the retrieved cutting forces compared to a milling test from Altintas Y. and Lee P. [2]. The considered multibody system is composed of a cylindrical end mill modelled by a frustum and four triangles under `EasyAnim` moving at a constant speed v along the x-axis (figure 4). The mill rotates at a constant speed Ω around the z-axis counter clockwise for a half immersion up-milling. The multibody model has accordingly 2 imposed motions. Consequently, the mill advances and rotates against the Ti6Al4V alloy thus generating cutting forces. The simulation parameters are presented in table 1.

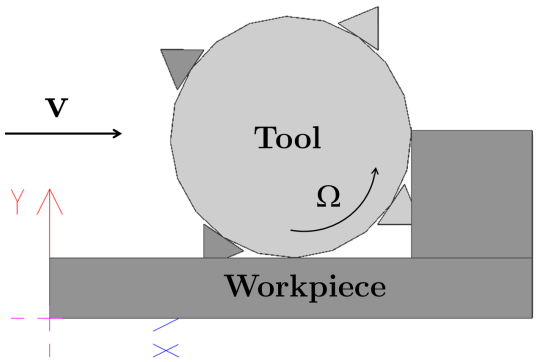
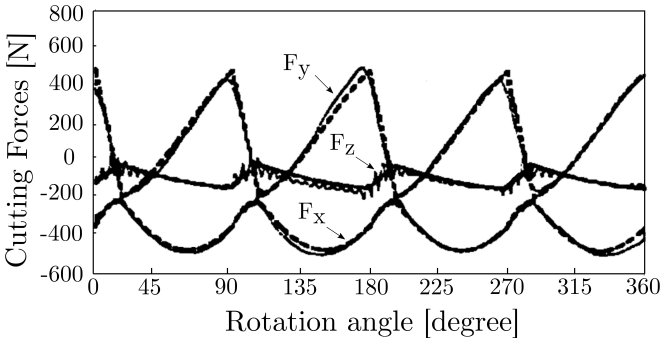


Fig. 4: Milling of the Ti6Al4V alloy with a cylindrical end mill

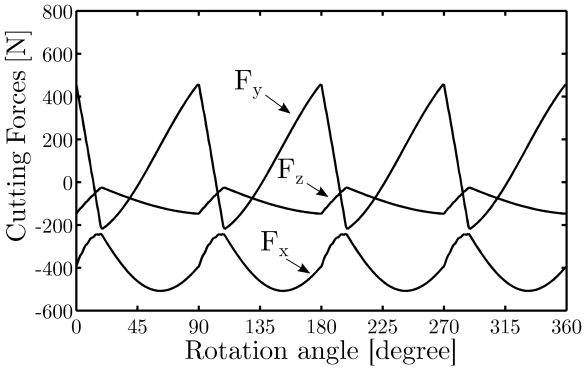
Simulation parameters		
Material	Ti6Al4V	[alloy]
$\underline{\mathbf{K}}_e$=[K_{te},K_{re},K_{ae}]	[24, 43, 0]	[N/mm]
Shear stress τ	613	[MPa]
Rake angle α	12	[degree]
Friction angle β	22.58	[degree]
Mill diameter	18.1	[mm]
Number of edges	4	[teeth]
Helix angle	30	[degree]
Feed	0.05	[mm/tooth]
Spindle speed	527	[RPM]
Axial Depth Of Cut (ADOC)	5.08	[mm]
Up-milling		

Tab. 1: Cutting conditions

The resulting cutting forces along the x-, y- and z-axes are plotted in figure 5b and compared to the corresponding test case of Altintas Y. and Lee P. (figure 5a) (authors give an extra comparison with experimental data). Cutting forces are shown over one tool rotation [0°to 360°] and turn out to be exactly the same as the reference. The cutting forces periodicity is due to the periodic entry of the mill into the workpiece. This first validation allows to make sure that the computed cutting forces are valid compared to a real test case. The mechanical model can thus be extended to incorporate the mill flexibility.



(a) Reference from Altintas Y. and Lee P. [2]



(b) Coupling `EasyDyn-DyStaMill`

Fig. 5: Cutting forces validation

4.2 Validation on a single-axis vibrating mill model

The second studied system comprises one additional degree of freedom which is the vibration of the mill along the x-axis with respect to its constant speed position. The whole mechanical system (figure 6) thus gathers 2 imposed motions which are the constant longitudinal speed v and the constant angular velocity Ω and 1 configuration parameter which is the vibration of the mill x_p materialized by a spring and damper system whose modal properties are depicted in table 2. The cutting forces are only analysed for the direction along the x-axis. As for the previous case, the mill moves forward into the material (aluminium 7075-T6) under half immersion up-milling conditions. The simulation parameters are presented in table 3.

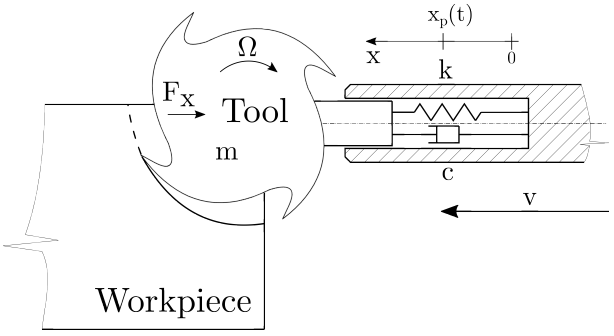


Fig. 6: Single-axis vibrating mill model

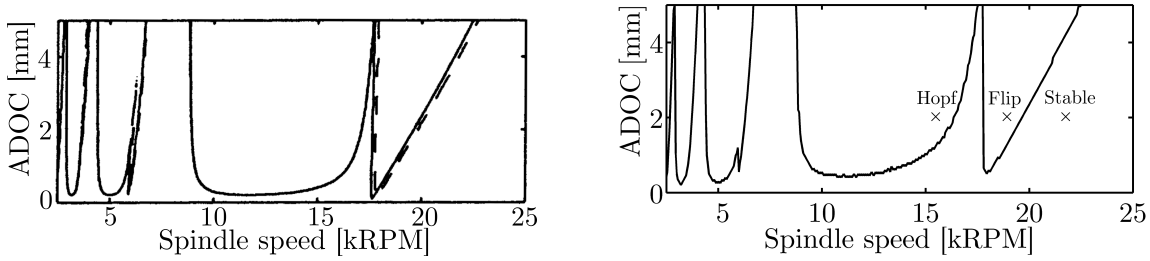
Modal properties		
Modal mass m	2.573	[kg]
Damping ratio ξ	0.32	[%]
Frequency f	146.5	[Hz]

Tab. 2: Mechanical parameters for the single-axis vibrating mill

Simulation parameters		
Material	7075-T6	[aluminium]
$\underline{\mathbf{K}}_c$=[K_{tc},K_{rc},K_{ac}]	[550, 200, 0]	[MPa]
Mill diameter	10	[mm]
Number of edge	1	[tooth]
Helix angle	0	[degree]
Feed	0.05	[mm/tooth]
Spindle speed	2500→25000	[RPM]
Axial Depth Of Cut (ADOC)	0.1→5	[mm]
Up-milling		

Tab. 3: Cutting conditions for the single-axis vibrating mill

The results of the libraries coupling are illustrated in terms of stability lobes. Stability lobes are a common way in milling to identify the milling stable operating conditions. The corresponding graph displays the axial depth of cut (ADOC) as a function of the spindle speed so that the stable operating zones are under the limiting resulting curve. The stability lobes are obtained through an iterative process consisting in increasing the axial depth of cut for a fixed spindle speed. For each axial depth of cut, a complete milling simulation is performed and the cutting forces are analysed. The distinction between stable and unstable simulation is classically set by a threshold between instantaneous undeformed chip thickness and the theoretical value obtained for a rigid test case. In this situation, the reference case is taken from an article of T. Insperger et al. [19] who retrieve the stability lobes for the studied system of which spindle speeds range from 2500 to 25000 [RPM]; they limit the axial depth of cut until 5 [mm] (figure 7a). The stability lobes arising from the libraries coupling are plotted in figure 7b and are consistent with the expected outcome.



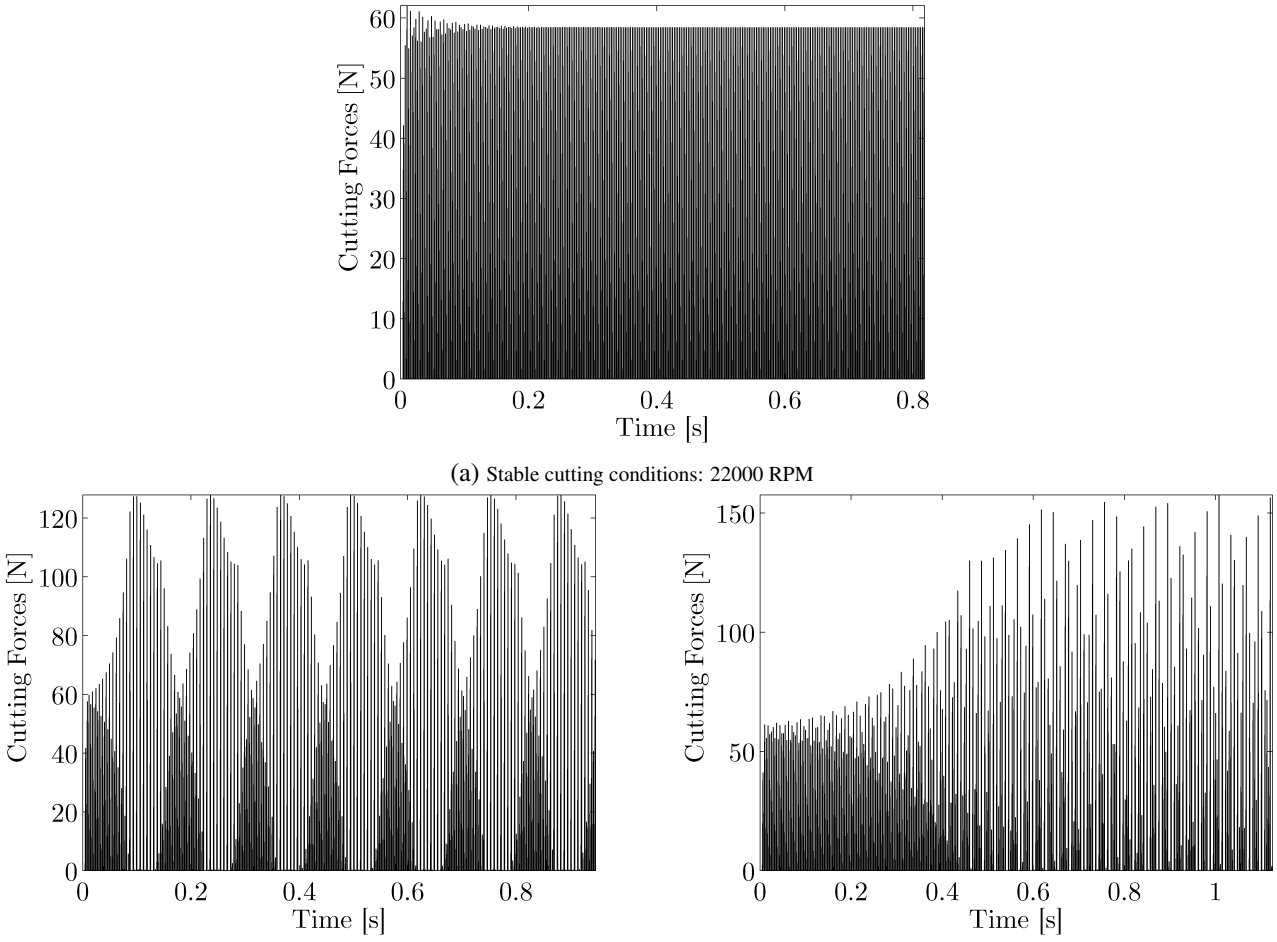
(a) Reference from T. Insperger [19]

(b) Coupling `EasyDyn-DyStaMill`

Fig. 7: Stability lobes validation

Additional results concern the evolution of the cutting forces for three specific operating conditions pointed out in figure 7b:

- figure 8a exposes the evolution of the cutting forces under stable operating conditions: 22000 [RPM] for an axial depth of cut of 2 [mm]. As announced, cutting forces are rather constant.
- figure 8b presents the evolution of the cutting forces under unstable operating conditions: 19000 [RPM] for an axial depth of cut of 2 [mm]. These unstable conditions are known as "flip bifurcation" or "period doubling bifurcation" and are related to the impact effects of entering and leaving the workpiece material [19]. The cutting forces are no longer constant and their amplitude has increased drastically.
- figure 8c shows the evolution of the cutting forces under unstable operating conditions: 16000 [RPM] for an axial depth of cut of 2 [mm]. These unstable conditions are known as "Hopf bifurcation" characterized by a change of the system stability and a periodic solution.



(a) Stable cutting conditions: 22000 RPM

(b) Flip milling instability: 19000 RPM

(c) Hopf milling instability: 16000 RPM

Fig. 8: Different milling conditions for an Axial Depth Of Cut (ADOC) of 2 [mm]

4.3 Validation on a bi-axis vibrating mill model

The mechanical system is subsequently extended with a supplementary vibrating axis along the y-axis. The whole system is therefore moving with possible vibrations along the x- and y-axes. This system is taken from an article of Smith S. and Tlusty J. [20] and the stability lobes are provided by Altintas Y. and Budak E. [6] (figure 11a). It is said that the milling operation consists in a half immersion up-milling of an aluminium workpiece milled by a cylindrical mill of 100 mm diameter with 8 teeth. The peculiarity of the mechanical system lies in the fact that each vibrating axis has 2 modes presented in table 5. The system is illustrated in figure 9. The milling simulation parameters are gathered in table 4.

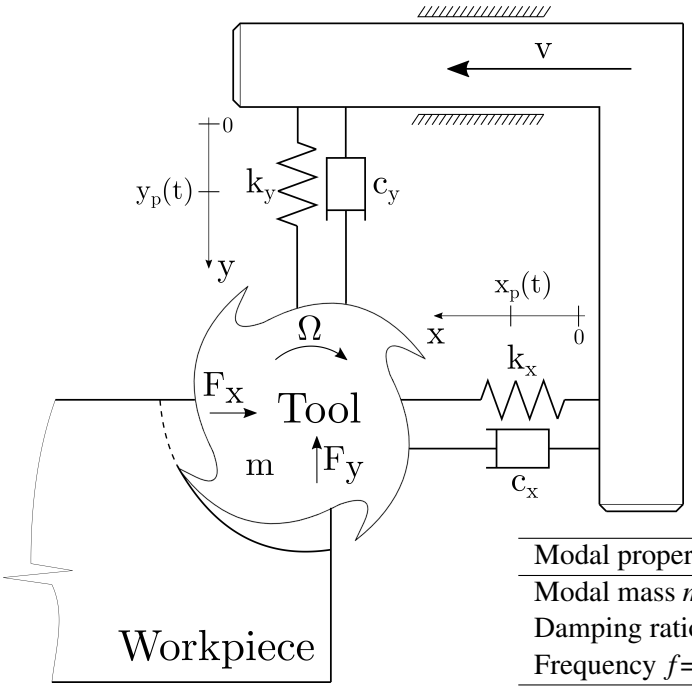


Fig. 9: Bi-axis vibrating mill model

Simulation parameters		
Material	aluminium	
$\underline{\mathbf{K}}_c$=[K_{tc},K_{rc},K_{ac}]	[1500, 450, 0]	[MPa]
Mill diameter	100	[mm]
Number of edge	8	[tooth]
Helix angle	0	[degree]
Feed	0.1	[mm/tooth]
Spindle speed	1000→8000	[RPM]
Axial Depth Of Cut (ADOC)	2→12	[mm]
Up-milling		

Tab. 4: Cutting conditions for the bi-axis vibrating mill

Modal properties		
Modal mass m=[m_{x_1},m_{x_2},m_{y_1},m_{y_2}]	[84.684, 9.273, 239.793, 4.47]	[kg]
Damping ratio ξ=[ξ_{x_1},ξ_{x_2},ξ_{y_1},ξ_{y_2}]	[12, 4, 10, 10]	[%]
Frequency f=[f_{x_1},f_{x_2},f_{y_1},f_{y_2}]	[260, 389, 150, 348]	[Hz]

Tab. 5: Mechanical parameters for the bi-axis vibrating mill

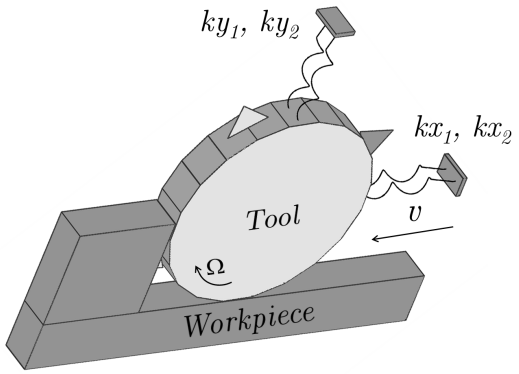
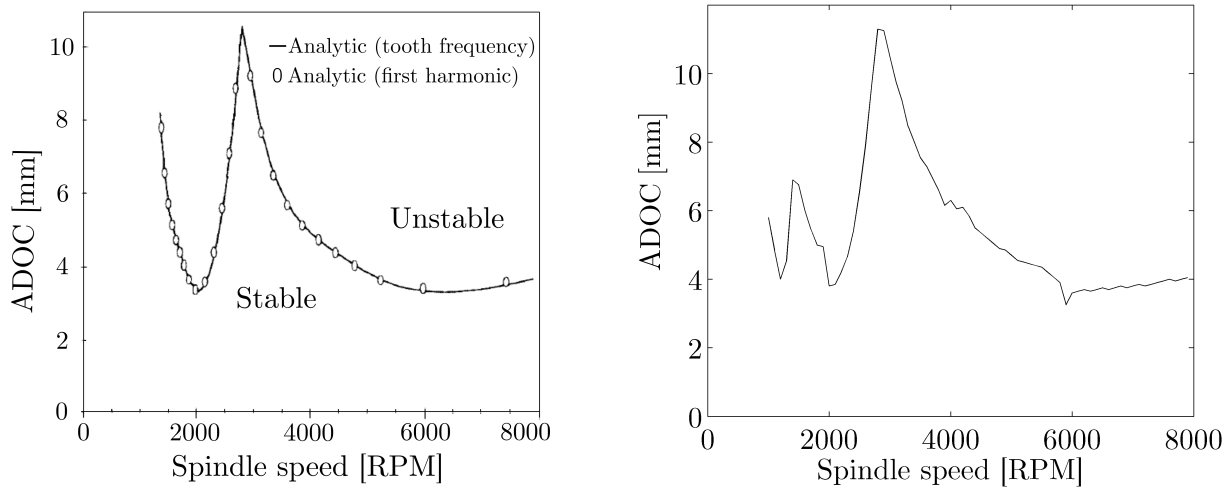


Fig. 10: Four modes vibrating mill under `EasyAnim`

The original mechanical system was adapted for the sake of simplicity in `EasyDyn`. For each vibrating axis, the 2 modes were split into 2 spring and damper systems (figure 10) and their vibrations were combined at each time step. The model was then used to get the stability lobes for a spindle speed ranging from 1000 [RPM] to 8000 [RPM] and an ADOC from 2 [mm] to 12 [mm]. The resultant stability lobes are depicted in figure 11b and are compared to the reference from Altintas Y. and Budak E. [6] in figure 11a. A good agreement is observed.



(a) Reference from Altintas Y. and Budak E. [6]

(b) Coupling `EasyDyn-DyStaMill`

Fig. 11: Stability lobes for a 4 vibrating modes mill

5 Application to a 3-DOF planar robot

The last application of the coupled system concerns a 3-DOF rigid planar robot made of three rigid links and three revolute joints. The three degrees of freedom of the planar robot are named q_1, q_2, q_3 and the length of each link is 500 [mm] as seen in figure 12. The mass of each link is set to 1 [kg] (I_{zz}=1 $[kg \cdot m^2]$). No more extra degree of freedom is allowed for the mill vibrations in this example. The whole structure is driven so that the robot end-effector follows the x-axis at a constant speed v reacting to milling forces on the same axis F_x. Lateral milling forces aren't taken into account. The milling parameters are taken from the single-axis vibrating mill model for the machining of an aluminium workpiece under half immersion up-milling conditions (table 6).

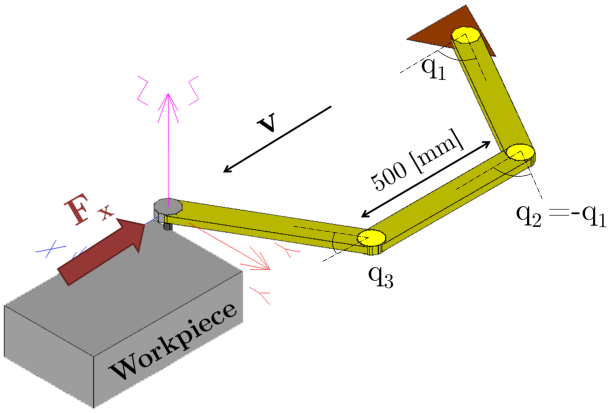


Fig. 12: 3-DOF rigid planar robot during a milling operation

Simulation parameters		
Material	7075-T6	[aluminium]
$\underline{\mathbf{K}}_c$=[K_{tc},K_{rc},K_{ac}]	[550, 200, 0]	[MPa]
Mill diameter	10	[mm]
Number of edge	1	[tooth]
Helix angle	0	[degree]
Feed	0.05	[mm/tooth]
Spindle speed	16000	[RPM]
Axial Depth Of Cut (ADOC)	2	[mm]
Up-milling		

Tab. 6: Cutting conditions for the 3-DOF robot

The adopted control scheme is based on the "Resolved Motion Rate Control" (RMRC) controller proposed by Daniel E. Whitney in 1969 for manipulator arms [21] (figure 13). The controller is rather simplistic and combines the forward and the inverse kinematics. By examining the scheme from left to right, the desired end effector position $\underline{\mathbf{x}}_d$ is specified in the operational space (or cartesian space) and is compared to the computed end effector position from the forward kinematics $\underline{\mathbf{x}}$. As a result, the end effector position error $\delta\underline{\mathbf{x}}$ is deduced. In order to find the corresponding position of the joints in the so-called joint space, the end effector position error $\delta\underline{\mathbf{x}}$ is multiplied by the inverse of the jacobian matrix $\mathbf{J}^{-1}$ composed of the configuration parameters partial velocities (cf. recall with equation 5) such as:

$$\delta\underline{\mathbf{q}} = \mathbf{J}^{-1}(q)\delta\underline{\mathbf{x}} \tag{9}$$

The jacobian matrix $\mathbf{J}$ was reduced to a [2x3] matrix since it is a planar model with 3 configuration parameters. Indeed, according to figure 13, $\delta\underline{\mathbf{x}}$ contains the x- and y-errors of the end-effector position that propagate for the 3 joints. In order to make the reduced jacobian matrix invertible, a condition was imposed so that the 2^{nd} link remains horizontal: $q_2 = -q_1$. A [2x2] invertible reduced jacobian $\mathbf{J}_{reduced}$ was therefore computed by the following expression:

$$\delta\underline{\mathbf{q}} = \underbrace{\left[\underbrace{\mathbf{J}(q)}_{2\text{x}3} \cdot \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix}\right]^{-1}}_{\mathbf{J}_{reduced}} \delta\underline{\mathbf{x}} \tag{10}$$

The position errors $\delta\underline{\mathbf{q}}$ in the joint space are then managed by a classical controller to minimize those errors. In that example, an arbitrary proportional gain was set to K = 100 to fastly reduce the errors of the configuration parameters. Using the robot dynamics, the controlled configuration parameters $\underline{\mathbf{q}}$ are finally used to recover the end effector position in the operational space $\underline{\mathbf{x}}$.

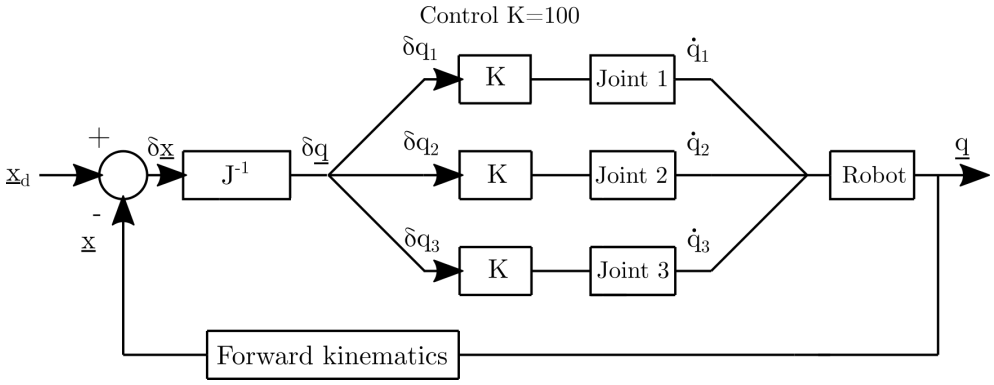


Fig. 13: Resolved Motion Rate Control (RMRC) controller

Figure 14 shows the evolution of the cutting forces for a milling simulation using the described planar robot. Milling conditions are clearly stable since the cutting forces converge towards a constant value. Cutting forces increase at the beginning of the simulation because the mill enters progressively in the workpiece. In this particular case, no instabilities can occur as the mill is not allowed to vibrate and the robot structure can be considered as rigid even if the proportional gains introduce a slight flexibility . The controller tracking error between the set point and the real position of the robot end-effector along the x-axis is plotted in figure 15 in which its steady state error reaches 0.134 [mm].

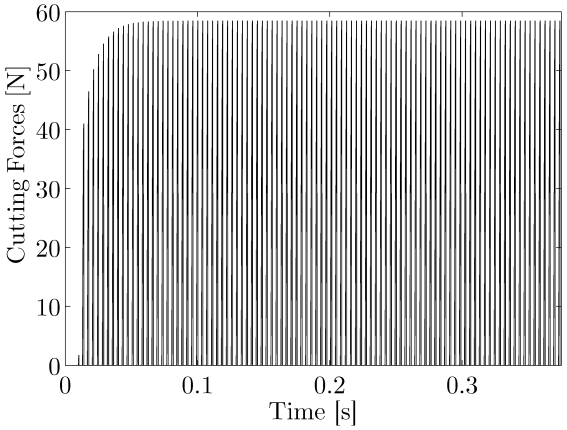


Fig. 14: Retrieved cutting forces with the 3-DOF robot

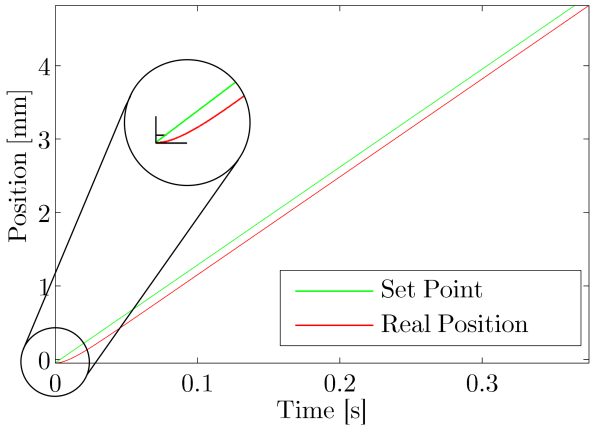


Fig. 15: Tracking error of the robot end-effector position along the x-axis

6 Conclusions

Robotic machining still has a bright future in mechanical manufacturing for further developments since it constitutes a flexible alternative for less accurate machining operations at a reduced cost. The low stiffness of industrial robots restricts their application to small process force load.

This paper has presented a coupling between two in-house libraries:

1. the `EasyDyn` library is a multibody library that allows to simulate the motion of a multibody system from a description of the motion in terms of the chosen configuration parameters and the expression of the applied forces. In this case, `EasyDyn` is used to simulate the mill dynamics as well as the machining robot.
2. the `DyStaMill` (DYnamic STAbility of MILLing operations) library is based on a milling analytical model to predict the cutting forces thanks to the position of the mill and the geometry update.

The coupling takes place by inputing the mill position computed from `EasyDyn` to `DyStaMill` which deduces the cutting forces and sends them back to the multibody library to perform the numerical integration. The workpiece surface is then updated according to the converged mill position.

Some milling simulations have proven a consistent coupling between the two libraries compared to literature milling tests. Ultimately, the coupling was implemented on a 3-DOF controlled planar robot model by analysing

the resultant cutting forces and the tracking errors. Note as well that the libraries coupling gives exactly the same results as the standalone version of `DyStaMill`.

Further developments will improve the 3-DOF planar with flexible links by assessing several control configurations.

Acknowledgements

The authors would like to thank the Belgian Fund for Scientific Research (FRIA-FNRS) for the financial support.

References

[1] F. Ducobu, E. Rivière, E. Filippi, "Dynamic simulation of the micro-milling process including minimum chip thickness and size effect," *Key Engineering Material*, vol. 504-506, pp. 1269–1274, doi: 10.4028/www.scientific.net/KEM.504–506.1269, 2012.

[2] Y. Altintas, P. Lee, "A general mechanics and dynamics model for helical end mills," *CIRP Annals - Manufacturing Technology*, vol. 45, pp. 59–64, doi: 10.1016/S0007–8506(07)63017–0, 1996.

[3] E. Rivère-Lorphèvre, E. Filippi, "Mechanistic cutting force model parameters evaluation in milling taking cutter radial runout into account," *Int J Adv Manuf Technol, Springer*, vol. 45, pp. 8–15, doi: 10.1007/s00170–009–1943–9, 2009.

[4] C. Brecher, M. Esser, S. Witt, "Interaction of manufacturing process and machine tool," *CIRP Annals - Manufacturing Technology*, vol. 58, pp. 588–607, 2009.

[5] Y. Altintas, M. Weck, "Chatter stability in metal cutting and grinding," *Annals of CIRP*, vol. 53, pp. 619–642, 2004.

[6] Y. Altintas, E. Budak, "Analytical prediction of stability lobes in milling," *Annals of CIRP*, vol. 44, pp. 357–362, 1995.

[7] E. Budak, "Analytical Model for High Performance Milling. Part I. Cutting Forces, Structural Deformations and Tolerance Integrity," *International Journal of Machine Tools and Manufacture*, vol. 46, pp. 1478–1488, 2006.

[8] E. Budak, "Analytical Model for High Performance Milling. Part II. Process Dynamics and Stability.," *International Journal of Machine Tools and Manufacture*, vol. 46, pp. 1489–1499, 2006.

[9] S. Engin, Y. Altintas, "Mechanics and dynamics of general milling cutters. Part I: helical end mills," *International Journal of Machine Tools and Manufacture*, vol. 41, pp. 2195–2212, 2001.

[10] M. Hiller, A. Kecskeméthy, "Dynamics of multibody systems with minimal coordinates," *Computer-Aided Analysis for Rigid and Flexible Mechanical Systems*, vol. 268, NATO ASI Series, pp. 61–100, 1994.

[11] *Computer-Aided Analysis of Mechanical Systems Prof. Olivier VERLINDEN Faculté Polytechnique de Mons Service de Mécanique Rationnelle 31 Bd Dolez B-7000 Mons (Belgium) Olivier.Verlinden@umons.ac.be*.

[12] J. Bauer, M. Friedmann, T. Hemker, *Process Machine Interactions*, vol. 1. pp. 245-263: Springer, 2013.

[13] U. Schneider, M. Ansaloni, M. Drusti, "Experimental investigation of sources of error in robot machining," *CCIS*, vol. 371, no. 1, pp. 14–26, 2013.

[14] B. Denkena, F. Hollmann, *Process Machine Interactions: Prediction and Manipulation of Interactions between Manufacturing Processes and Machine Tool Structures*. Chap: 11: Analysis of Industrial Robot Structure and Milling Process Interaction for Path Manipulation, pp. 250-263, doi: 10.1007/978-3-642-32448-2, Springer, 2013.

[15] O. Verlinden, L. Ben Fékih and G. Kouroussis, “Symbolic generation of the kinematics of multibody systems in EasyDyn: From MuPAD to Xcas/Giac,” *Theoretical and Applied Mechanics Letters*, vol. 3, no. 1, pp. 013012, doi: 10.1063/2.13013012, 2013.

[16] O. Verlinden, G. Kouroussis, C. Conti, “EasyDyn: a framework based on free symbolic and numerical tool for teaching multibody systems,” in *Multibody Dynamics 2005, ECCOMAS Thematic Conference*, Madrid, Spain, 21-24 June 2005.

[17] G. Peigné, H. Paris, D. Brissaud, “A model of milled surface generation for time domain simulation of high-speed cutting,” *Proceedings of the Institution of Mechanical Engineers*, vol. 217, pp. 919–930, 2003.

[18] E. Rivière, E. Filippi, P. Dehombreux, “Forces, vibrations and roughness prediction in milling using dynamic simulation,” *Proceedings*, (Fifth International Conference on High Speed Machining (HSM 2006)), Mars, Metz, France 2006.

[19] T. Insperger, B.P. Mann, G. Stépán, P.V. Bayly, “Stability of up-milling and down-milling, part I: alternative analytical methods,” *International Journal of Machine Tools and Manufacture*, vol. 43, pp. 25–34, 2003.

[20] S. Smith, J. Tlusty, “Update on High Speed Milling Dynamics,” *Annals of CIRP*, vol. 39, pp. 517–521, 1990.

[21] Daniel E. Whitney, “Resolved Motion Rate Control of Manipulators and Human Prostheses,” *IEEE Transactions on man-machine systems*, vol. MMS-10, no 2, pp. 47–53, June, 1969.