The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-018-2357-3

ORIGINAL ARTICLE

Dystamill: a framework dedicated to the dynamic simulation of milling operations for stability assessment

Hoai Nam Huynh[1] · **Edouard Rivière-Lorphèvre**[2] · **François Ducobu**[2] · **Abdullah Ozcan**[2] · **Olivier Verlinden**[1]

Received: 26 January 2018 / Accepted: 15 June 2018

Abstract

`Dystamill` is an open-source framework, available for free on the Internet, that allows its user to perform milling simulations from a model of the machine tool dynamics, a description of the cutting tool, a representation of the workpiece and a cutting force model. First developed as a program to study the stability of milling operations, the `C++` code was ported into a library so that it can now be coupled with a multi-physics engine or used standalone. Restricted to $2D\frac{1}{2}$ cases, achievable cutting operations include contouring, slotting and face or pocket milling since the tool only moves in a plane perpendicular to its revolution axis. `Dystamill` permits computing the machining forces applied on the tool and generating the geometry of the machined workpiece. Achieving dynamic simulations in a row for various cutting conditions enables determining the chatter stability lobes needed to choose optimal cutting parameters. The `C++` library is composed of 12 modules interacting with each other to model the milling studied case, involving namely a modelling of the tool and the machined surface, a cutting force model and a representation of the system dynamics through a modal model. It is completed with a `GUI` that generates an input file for `Dystamill` while the post process of the results can be carried out by Matlab™. Born from a Ph.D. thesis, `Dystamill` was tested and validated on numerous test cases from the literature and real experiments. This paper is an opportunity to shed light on the functions and algorithms of `Dystamill`. In the hope that it will be useful for users, this document presents the `Dystamill` library, namely its modules are described from a technical point of view. An early example is built up and validated through real measurements. Finally, the capabilities of `Dystamill` are illustrated by comprehensive examples leading to the chatter stability lobes.

Keywords Chatter detection · Dynamic system · Machining simulation · Machine tool · Numerical · Open source · Programming

Nomenclature

a_p, a_e Axial and radial depth of cut

α, β	Cutting edge rake and inclination angles
D, R, R_r	End mill parametric radial dimensions
F_x, F_y, F_z	x, y and z global cutting forces
dF_a, dF_r, dF_t	Axial, radial and tangent force elements
h	Chip thickness
$i(z)$	Helix angle
m, k, c	Modal mass, stiffness and damping
κ	Axial immersion angle
K_{ac}, K_{rc}, K_{tc}	Axial, radial and tangent cutting coefficients
K_{ae}, K_{re}, K_{te}	Axial, radial and tangent edge coefficients
θ	Cutting edge rotation angle
q	Configuration parameter of modal model
R_t, R_q, R_a	Total, quadratic and arithmetic roughness
t	Time
v_f	Feed rate
dz	Tool elementary slice height
Ω	Spindle rotation velocity

✉ Hoai Nam Huynh
hoainam.huynh@umons.ac.be

Edouard Rivière-Lorphèvre
edouard.riviere@umons.ac.be

François Ducobu
francois.ducobu@umons.ac.be

Abdullah Ozcan
abdullah.ozcan@umons.ac.be

Olivier Verlinden
olivier.verlinden@umons.ac.be

[1] Theoretical Mechanics, Dynamics and Vibrations, University of Mons, Place du Parc 20, 7000 Mons, Belgium

[2] Machine Design and Production Engineering, University of Mons, Place du parc 20, 7000 Mons, Belgium

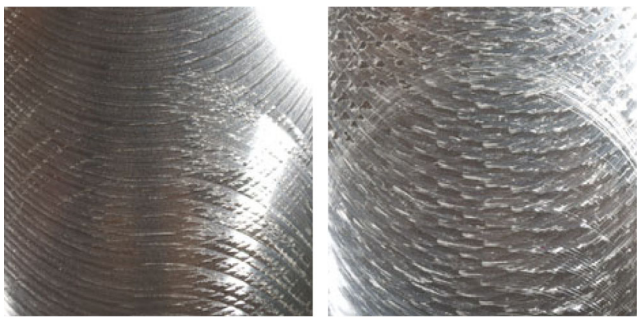
Published online: 30 June 2018

1 Introduction

Milling is still a key technology for the manufacturing of mechanical parts requiring high accuracy. One of the most troublesome and well-known phenomena is called chatter which causes excessive vibrations at tooltip damaging the workpiece (Fig. 1). Two types of chatter were identified by Tobias and Fishwick [1] and Tlusty and Polacek [2], namely the regenerative and the mode coupling chatter. Regenerative chatter occurs when the mill cuts a surface already machined from a previous operation. Mode coupling chatter appears when the machine tool vibrates in the directions of its degrees of freedom. These effects raise the cutting force level by degrading the surface quality. As explained by Altintas [3], the self-excited vibrations from the regenerative chatter happen more often than the mode coupling chatter.

A way to get around this disturbing effect without spoiling milling performances consists in resorting to simulation. The reduction of setup time and the low cost of this approach are crucial to maintain a competitive position among industrial manufacturers. Furthermore, great benefits in terms of part quality are obtainable with the numerical expertise. For instance, as chatter shows up for some combinations of spindle speeds and depths of cut, simulation makes possible the trials of numerous cutting conditions so that optimal cutting parameters can be identified without using a single piece of material.

Functions to detect unstable milling conditions can be classified into four categories ranging from analytical methods, methods studying the properties of delay differential equations, dynamic methods to finite element ones. Analytical methods linearise the machining process in which cutting forces are assumed to be proportional to the chip thickness [4]. Modern techniques of stability detection rely on the study of delay differential equations [5] and prove to be more accurate. Since analytical methods cannot take nonlinearities such as the chip thickness variation into account [6–8], dynamic methods were developed by proceeding to a temporal and a spatial discretisation of the milling process. Dynamic simulations are a fair trade-off between accurate results and simulation time, larger for finite element methods able to account for complex phenomena related to the chip generation mechanisms [9].



Machined surface without chatter

Machined surface with chatter

Fig. 1 Chatter phenomenon in machining

Since stability is a key aspect in machining, `Dystamill`, standing for "Dynamic stability of milling operation", was developed to study the behaviour of milling operations on the basis of dynamic simulations. Born from the Ph.D. thesis of Rivière-Lorphèvre [10], the simulation environment was ported into a `C++` library ready to be used standalone or coupled with a multi-physics engine. It was beforehand tested and validated on numerous test cases from the literature and real experiments. The complete package, freely available from the Internet, comprises the source code, a user guide and examples.

In `Dystamill`, milling simulations rely on the interaction of three elementary components which are the tool, the workpiece and the machine tool to which a cutting force model is paired. The end mill and the workpiece are spatially discretised and the machine tool model is reduced to a modal basis on which cutting forces are applied. The exact kinematic of the cutting edge is computed, combining the feed motion, the rotation of the cutter and the vibration of the TCP. It allows taking into account nonlinearities such as chip thickness variation, time delay and tool periodic entry in the material. Chatter phenomena can therefore be accurately detected and post-chatter motions can be captured if the duration of the simulation is extended.

This paper can be seen as an opportunity to shed light on the functions and algorithms implemented behind the generated results. In the hope that it will be useful for the users, this document is organised as follows: in Section 2, the milling model is presented through its four elementary components which are then linked with the layout of `Dystamill` in Section 3, namely its modules are described. An early example is directly built up and validated through real measurements. Section 4 proposes a complete simulation example leading to the chatter stability lobes while Section 5 deals with one more advanced test case.

2 Milling model

The purpose of this section is to expose the four elementary components implemented in `Dystamill` in order to perform dynamic simulations of milling. Namely, functions to model the tool, the workpiece, the machine tool and the cutting forces are presented. The end of the section explains the simulation process.

2.1 Tool modelling

In Dystamill, all types of solid end mills can be configured according to the requirements. Insert end mills are not considered for the moment. Cylindrical and ball end mills are completely defined by their diameter D while bull nose end mills require one more parameter which is the inner radius of the cutting edge Rc. More generally, the external envelope of any solid end mill can be described by seven parameters as stated in [11] (APT model) (Fig. 2).

Once the exterior shape of the end mill is drawn, it is spatially discretised into superimposed slices of height dz along its revolution axis [12]. Then, from the number of teeth specified by the user, cutting edges of the tooltip slice are positioned. A possible uneven spacing between the teeth and their eccentricity can be taken into account. Lastly, the coordinates of upper slice cutting edges are simply found by wrapping their profiles around the tool envelope knowing the helix angle (Fig. 3). Cutting edge coordinates are consequently obtained for any tool slice. In summary, the tool is represented by a collection of cutting edges in Dystamill. Local cutting forces dF_a, dF_r and dF_t applied on the cutting edges of each slice will be computed by the cutting force model.

2.2 Workpiece modelling

While any workpiece geometry can be modelled as superimposed slices, the default shape is a rectangular blank whose dimensions are pre-defined so that the cutting tool is still in the material at the end of the simulation. The displacement of the end mill, lying in a plane perpendicular to its revolution axis, is assumed to follow a straight line into the material. This approach still allows a milling stability prediction since the in-plane process forces are dominant. Hence, cutting operations are bounded to generate $2\frac{1}{2}D$ shapes by contouring, slotting and face or pocket milling. As depicted in Fig. 4, the default feed direction is settled as the x-direction and a_p and a_e denote the axial and radial depths of cut respectively. In addition, only the roughness of the lateral face can be assessed through this approach.

2.3 Machine tool modelling

The dynamic modelling of a machine tool may be difficult if the features of all its components are considered, as is for example a multibody representation in which the tool flexibility, the spindle and the actuator dynamics and the control are possibly taken into account. Besides, experimental models based on modal model of the machine tool dynamics can be exploited when little information is available about its internal structure. The latter approach was implemented in Dystamill by reducing the modal model to the tool dynamics. Consequently, the dynamic model of the machine tool was created as a succession of mass-spring-damper systems distributed along x and y directions and attached, on one end, to a rigid base and on the other end, to the tool. The rigid base moves at a constant velocity v_f which is the feed rate of the milling process. Along each direction, several mass-spring-damper

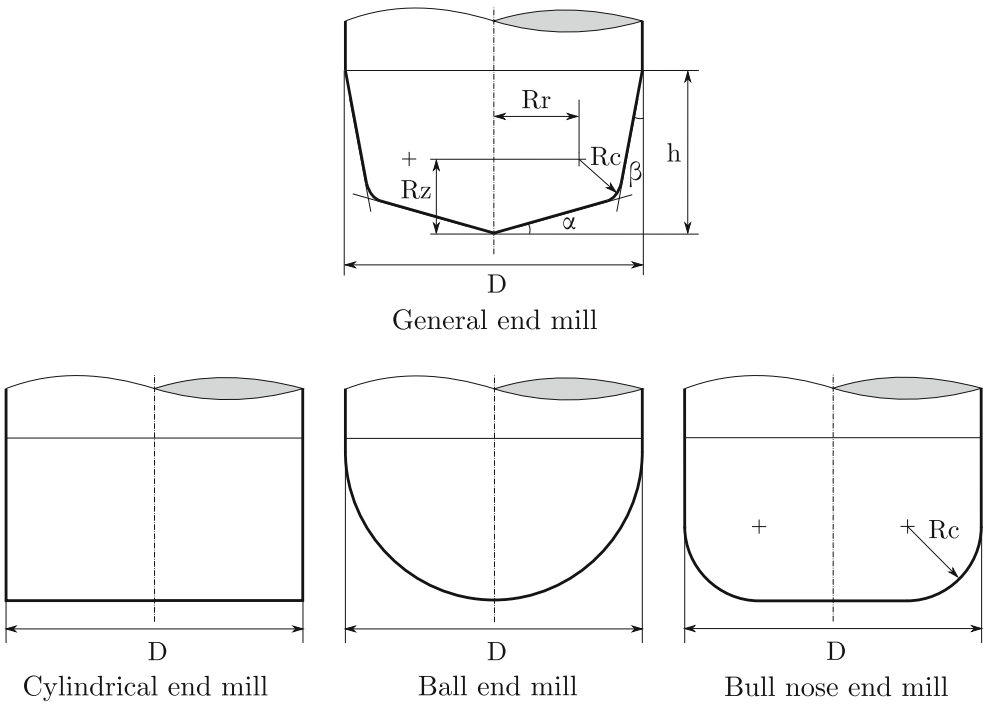


Fig. 2 Tool envelopes available in Dystamill

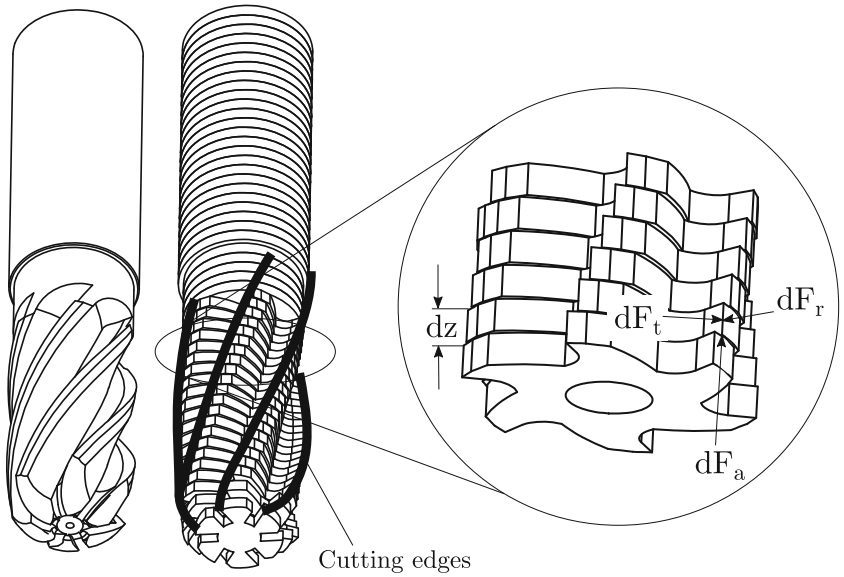


Fig. 3 Cutting edges wrapped around the tool envelope

systems can be set in parallel, each of them representing one vibration mode at tooltip. Figure 5 depicts the dynamic model from the tool point of view reacting to cutting forces F_x and F_y.

The general equations of motion to solve for a dynamic system containing n modes are presented hereafter. Matrices $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{K}$ being the mass, damping and stiffness matrices respectively

$$[\boldsymbol{M}]\left\{\underline{\ddot{\boldsymbol{q}}}\right\}+[\boldsymbol{C}]\left\{\underline{\dot{\boldsymbol{q}}}\right\}+[\boldsymbol{K}]\left\{\underline{\boldsymbol{q}}\right\}=\left\{\underline{\boldsymbol{F}}\right\}. \tag{1}$$

If $\boldsymbol{\Phi}$ is the matrix gathering the eigenvectors, the system can be decomposed into a modal basis by posing $\left\{\underline{\boldsymbol{q}}\right\}=[\boldsymbol{\Phi}]\left\{\underline{\boldsymbol{c}}\right\}$ leading to the following relationship:

$$\begin{aligned}&[\boldsymbol{\Phi}]^T[\boldsymbol{M}][\boldsymbol{\Phi}]\left\{\underline{\ddot{\boldsymbol{c}}}\right\}+[\boldsymbol{\Phi}]^T[\boldsymbol{C}][\boldsymbol{\Phi}]\left\{\underline{\dot{\boldsymbol{c}}}\right\}\\&+[\boldsymbol{\Phi}]^T[\boldsymbol{K}][\boldsymbol{\Phi}]\left\{\underline{\boldsymbol{c}}\right\}\\&=[\boldsymbol{\Phi}]^T\left\{\underline{\boldsymbol{F}}\right\}.\end{aligned} \tag{2}$$

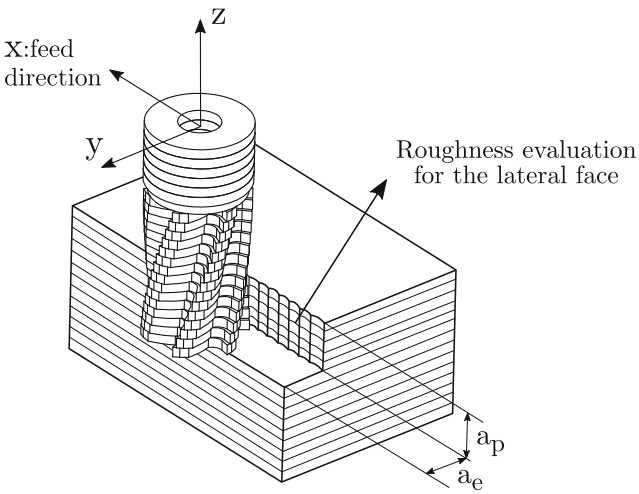


Fig. 4 Typical machining simulation

Assuming a proportional damping and the orthogonality property of eigenvectors, the dynamic system can be classically decoupled as follows:

$$\begin{cases}\ddot{c}_1+2\xi_1\omega_1\dot{c}_1+{\omega_1}^2c_1=\frac{1}{m_1}\left\{\underline{\boldsymbol{\Psi_1}}\right\}^T\left\{\underline{\boldsymbol{f}}\right\}\\ \vdots\\ \ddot{c}_n+2\xi_n\omega_n\dot{c}_n+{\omega_n}^2c_n=\frac{1}{m_n}\left\{\underline{\boldsymbol{\Psi_n}}\right\}^T\left\{\underline{\boldsymbol{f}}\right\}\end{cases} \tag{3}$$

with

- m_i: the modal mass of mode i [kg];
- ω_i: the eigen frequency of mode i [rad/s];
- ξ_i: the damping ratio of mode i [%];
- n: number of the mode.

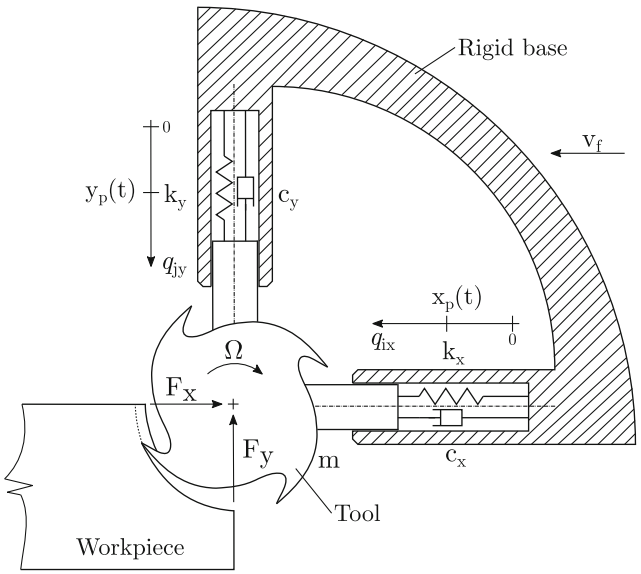


Fig. 5 Modal model of the machine tool

The resulting system is composed of n decoupled equations, meaning that each of them only depends on one single degree of freedom: each vibration mode and its amplitude at tooltip being characterised by a modal mass m, a natural frequency ω and a damping ratio ξ. Classically, experimental modal analysis providing the vibration response at tooltip allows the identification of the above parameters. Modal information is also typically supplied in related papers to characterise the system dynamics.

2.4 Cutting force model

For the modelling of the cutting forces, two varieties of models coexist in the literature which are classified in physical models and mechanistic models [13]. The former predict the evolution of the cutting forces by focussing on physical phenomena. On the other hand, mechanistic models are based on the principle that the exact modelling of physical phenomena can be challenging and hardly exploitable in practice. Therefore, they rely on a simplified modelling whose parameters are identified through experiments. In `Dystamill`, the physical model proposed by Merchant and two mechanistic models, from Kienzle and Altintas respectively, are implemented.

1. The Merchant's model: physical model

   Starting in the 1940s, Merchant proposed a model based on orthogonal cutting in which the tool is moving into the workpiece to remove a layer with a constant thickness [14]. In this model, cutting forces are computed by using the principle of minimum power stating that the shear angle guides the direction of the deformation. Though simplistic, this physical model postulated the fundamental principle claiming that cutting forces were proportional to the chip cross section. Thereafter, as most of the tools introduced helix angle, cutting edges removed material with a certain inclination angle with respect to the feed direction (Fig. 6). For this reason, an orthogonal/oblique transformation was proposed in order to transpose the relationships found in orthogonal cutting to oblique cutting [15]. The orthogonal/oblique model constitutes the most classical representation to compute cutting forces with a physical model.

2. The Kienzle's model: mechanistic model

   Mechanistic models allow the prediction of cutting forces from a given tool/material couple using simple analytical laws. They are also based on the principle that cutting force elements are proportional to the undeformed chip cross section. Models often encountered compute elementary local cutting forces dF on a frame attached to the cutting edges as functions of the undeformed chip thickness h and the height of a tool slice dz.

   Kienzle proposed a cutting law in which the chip thickness h is adorned with an exponent n [16]:

$$dF_i = K_i \cdot h^n \cdot dz \tag{4}$$

   with

   - $i = \{t, r, a\}$: t along the cutting speed, r along the local normal of the tool envelope and a along the third direction generating an orthogonal frame (Fig. 7)
   - K_i: cutting force coefficients [N/m^2]
   - n: exponent which commonly varies between 0.6 and 0.8 (Kienzle's model is also named the exponential model)

   Exponent n is an empirical parameter to adjust the simulated cutting force level to the experiments. Cutting force coefficients K_i, also known as specific pressures, are assumed to be constant for a given tool/material couple. Those have to be identified from inverse

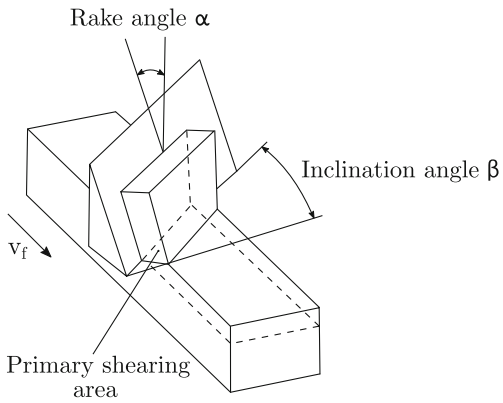


Fig. 6 Oblique cutting

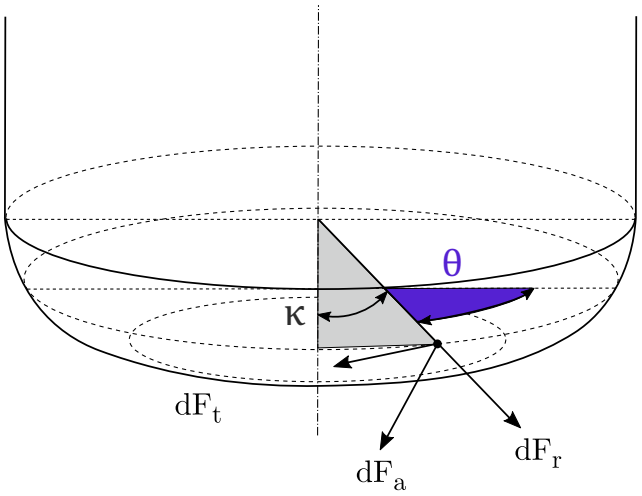


Fig. 7 Local frame of the cutting forces

methods since no relationship exists between them and material properties.

3. The Altintas' model: mechanistic model

   Altintas et al. [17] proposed another popular model taking into account the effect of friction along the local segment dS of the cutting edge. As a result, cutting forces comprise not only the forces generated in the primary shearing area but also forces emerging from the friction between the chip and the cutting edges.

$$dF_i = K_{i,c} \cdot h \cdot dz + K_{i,e} \cdot dS \tag{5}$$

with

- $K_{i,c}$: cutting force coefficients [N/m^2]
- $K_{i,e}$: edge force coefficients to consider the frictional effect [N/m]

Cutting tests are required to experimentally identify parameters $K_{i,e}$ by inverse techniques [3]. It is a commonly accepted model to assess stability issues.

Building on the determined local cutting forces dF_i from the different modellings, the elementary contributions of all cutting edges are projected onto a global reference frame linked to the ground using two tool angles. As depicted in Fig. 7, the considered cutting edge is located around the revolution axis by angle θ and vertically by angle κ also called the axial immersion angle.

Obtained angles allow computing the global cutting forces of each cutting edge.

$$\begin{Bmatrix} dF_x \\ dF_y \\ dF_z \end{Bmatrix} = \begin{bmatrix} -\cos\theta & -\sin\theta\cdot\sin\kappa & -\sin\theta\cdot\cos\kappa \\ \sin\theta & -\cos\theta\cdot\sin\kappa & -\cos\theta\cdot\cos\kappa \\ 0 & -\cos\kappa & -\sin\kappa \end{bmatrix} \cdot \begin{Bmatrix} dF_t \\ dF_r \\ dF_a \end{Bmatrix} \tag{6}$$

with

- $dF_{x,y,z}$: infinitesimal cutting forces in the global reference frame [N]

Resultant cutting forces F_x, F_y and F_z are obtained from the spatial integration of all their elementary contributions over each tooth and each tool slice.

2.5 Simulation process

Elementary components necessary for milling simulations being defined, they all interact as part of the simulation process. The `Dystamill` literal algorithm is presented in Fig. 8.

`Dystamill` loads the data once user has completed an input file gathering all the needed information to define the

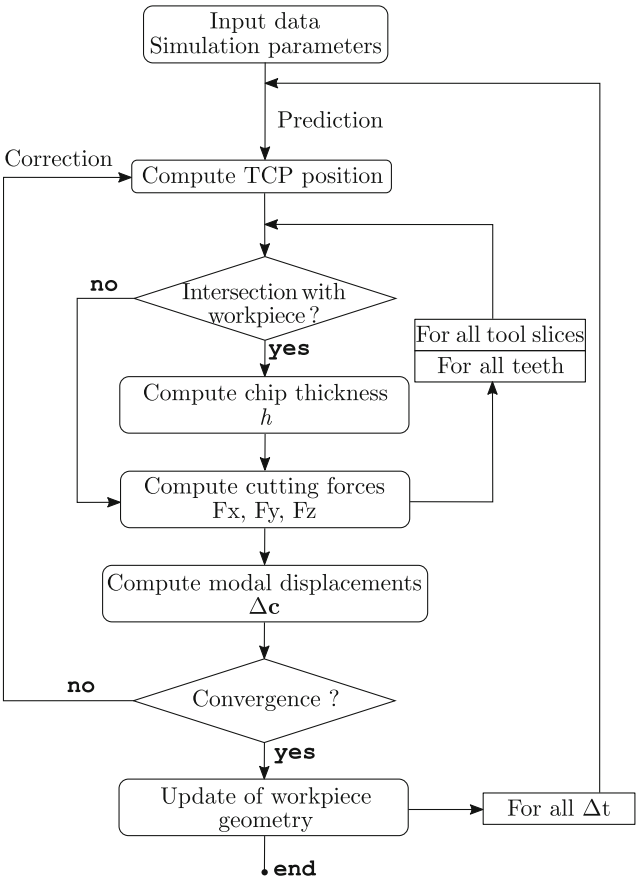


Fig. 8 Algorithm to perform milling simulations

tool, the workpiece, the modal properties of the machine tool, the cutting force model and lastly, the simulation parameters such as the duration of the simulation or the spatial and time refinements. The simulation starts while the tool is outside the workpiece. Driven by a specified feed rate v_f, the tool moves towards the workpiece by following a straight line along the x-axis. At each time step, the algorithm is therefore able to predict the position of the tool centre point *TCP* on the basis of the kinematics. Moreover, the location of its cutting edges, rotated by angle $d\theta = \Omega \cdot \Delta t$ through the spindle rotation velocity Ω, can be inferred at any time from the monitoring of the *TCP*. As soon as one of its cutting edges hits the workpiece in the predicted position, the portion of material to remove is computed in order to deduce the chip thickness. From the chip dimensions, elementary cutting forces dF_x, dF_y and dF_z associated with the considered cutting edge can be determined on the basis of the assumption that they are proportional to the quantity of material cut out. If no intersection occurs, the computation of the chip thickness is skipped as it is time consuming and no cutting force is computed. The intersection procedure must be completed

for all cutting edges and all teeth along the entire length of the tool represented by superimposed slices. Once global cutting forces F_x, F_y and F_z are calculated, the equations of motion governing the modal representation of the machine tool are built and solved (1). The integration of the cutting forces allows correcting the prediction of the *TCP* position at the end of time step, i.e. in $t + \Delta t$. In `Dystamill`, the n decoupled equations of motion are integrated separately using the Newmark's scheme where ζ and γ are the parameters of the numerical integration scheme, corresponding to the standard Newmark 1/4

$$c^{t+\Delta t} = c^t + \Delta t \dot{c}^t + (0,5 - \zeta)\,\Delta t^2 \ddot{c}^t + \zeta\,\Delta t^2 \ddot{c}^{t+\Delta t} \tag{7}$$

$$\dot{c}^{t+\Delta t} = \dot{c}^t + (1 - \gamma)\,\Delta t \ddot{c}^t + \gamma\,\Delta t \ddot{c}^{t+\Delta t}. \tag{8}$$

The transformation from the modal basis to the physical quantities is carried out using $\{\underline{q}\} = [\Phi]\{\underline{c}\}$. The corrected *TCP* position is then compared to the initial prediction throughout a convergence test. If the deviation exceeds a certain tolerance, the new *TCP* position becomes the *TCP* prediction for the next iteration within the same time step. The convergence loop was added in order to enable the selection of greater time step Δt without adversely affecting the computation of the cutting forces. These implicit iterations allow a reduction of `CPU` time by a factor up to ten [18]. Finally, once the magnitudes of the cutting forces have stabilised over one time step Δt, the workpiece geometry can be updated by using an "eraser of matter" technique. The process is naturally repeated over the entire duration of the simulation.

3 Dystamill library

The C++ library of `Dystamill` is organised in 12 modules interacting with each other. `Dystamill` gathers three types of module: data, conversion or routine. The "data" type concerns modules including the definition of global variables while the "conversion" type uses the values of the global variables to set operational variables (actually used for computation, e.g. entry and exit angles in the material). The "routine" type applies to modules clustering functions. In this section, an overview of the library will be first portrayed before presenting the main features of each module individually. A first illustrative example is directly proposed at the end of the section.

3.1 `Dystamill` layout

The description of the `Dystamill` structure will rely on the data flow depicted in Fig. 9. Its central portion displays the four steps to get through in order to output the desired results. Each step is then associated with its main substeps on the sides of the data flow, the name of the modules being indicated near the arrows linking the step to its substep(s).

The `Dystamill` data flow follows the same logic as the simulation algorithm. As a first step, user completes an input file under the form of a text file, whose extension is ".dat", including all the necessary information to execute the simulation. The input file can be easily generated thanks to a graphical user interface (`GUI`).

The provided data are then loaded into the `Dystamill` variables as soon as the simulation is launched. The-

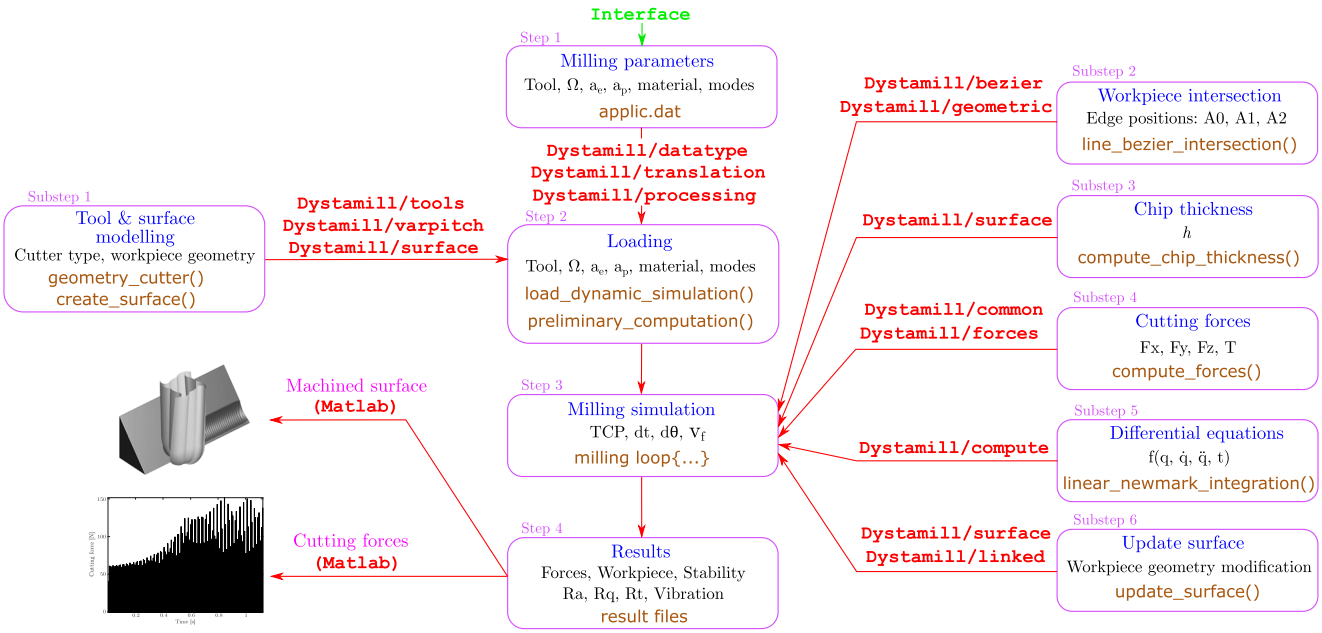


Fig. 9 Dystamill data flow

datatype module gathers the variables and structures needed to achieve the milling simulation. A bit aside the algorithm, the translation module defines the variables containing the error messages to be printed either in French or in English. Once data are loaded into their variables through the processing module, function

preliminary_computation() converts them into operational variables, for instance to determine the thickness of a tool slice. Dystamill then proceeds to the modelling of the tool and the workpiece through functions geometry_cutter() and create_surface(). Module varpitch achieves the modelling while module surface initialises variables to represent the workpiece geometry. The two modules operate with the help of the tools module offering, despite its misleading name, mathematical utilities. It relies on the GSL (GNU Scientific Library [19]) to perform complex computations requiring matrix operations, optimisation algorithms or algebra calculus.

The tool, the workpiece and the machine tool being defined by their modal model, the simulation starts upon entering into the milling_loop{...} which accomplishes the aforementioned substeps of the algorithm. The detection of the intersection between a cutting edge and the workpiece is carried out with the geometric module. The curved shape of the chip is obtained thanks to the bezier module which interpolates a spline between the two last positions of the considered cutting edge (points A1 and A2 in Fig. 13) using function line_bezier_intersection(); A0 being the current cutting edge position. The chip thickness in its undeformed configuration is then computed in the surface module. It is directly used to infer the cutting forces either through the common module, implementing the oblique cutting precepts, or either through the forces module, implementing the mechanistic models. In addition to the global cutting forces, the spindle torque T is also deduced. Cutting forces are then integrated alongside the dynamics of the machine tool through function linear_newmark_integration() inside the compute module. As soon as convergence is reached within the simulation process, the corrected TCP position enables the access to the last substep which is the update of the workpiece geometry. The portion of material swept by the cutting edges is removed from the blank. Function update_surface() of the surface module engages the workpiece geometry modifications. Namely, it calls functions inside the linked module to revise the workpiece model represented as a doubly linked list of coordinate points.

When the simulation is completed, Dystamill outputs result files containing the time evolution of cutting forces and tool vibrations, the machined workpiece geometry and the surface finish characteristics. The stability monitoring is achieved online but the outcome is provided at the end of the simulation.

3.2 Input file

The input file is simply a text file whose extension is ".dat" used to perform dynamic simulations of milling. It can be completed with the help of a graphical user interface simply called Interface programmed with Qt [20]. The user window consists of five tabs explicitly named after the simulation parameters to be entered.

1) Figure 10 introduces the first tab named Tool. User selects the type of tool envelope from a drop menu. Cylindrical, ball, bull nose and general end mills are selectable and configurable by specifying the values of the envelope parameters as depicted in Fig. 2. In addition, user must provide the number of teeth and the helix angle of the tool. Optional variables allow assigning a runout for each tooth and a variable pitch between them.
2) Tab Technological parameters defines the cutting conditions, namely the cutting direction (either down- or up-milling), the spindle rotation velocity Ω, the feed speed, the axial depth of cut a_p and the lateral positioning of the workpiece from a drop menu. Within the menu, user decides which milling operation to execute bearing in mind that the default motion of the end mill is a straight line into the material. Available operations are slot milling, half immersion milling and shoulder milling by specifying the radial depth of cut a_e.
3) Tab Material allows choosing the cutting force model. If the physical model is chosen, user must

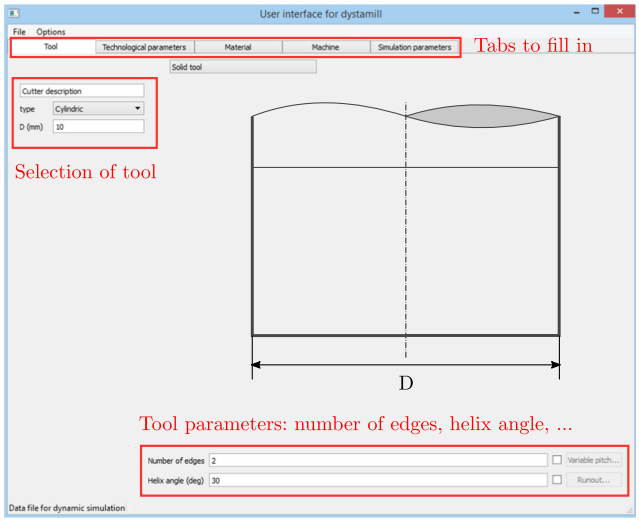


Fig. 10 *G*raphical *U*ser *I*nterface

provide the rake angle α and the inclination angle β (Fig. 6) as well as one cutting coefficient K, experimentally identified through orthogonal tests and reflecting the tool/material couple. On the other hand, if a mechanistic model is selected, user must supply the cutting force coefficients $K_{i,c}$, and possibly the edge force coefficients $K_{i,e}$, for the linear model of Altintas. The value of the exponent adorned to the chip thickness is required to apply Kienzle's model.

4) Tab `Machine` is concerned with the dynamic characteristics of the machine tool represented through a modal model from the point of view of the tool. For each component mode identified at tooltip along the x and y directions, user must provide its natural frequency, its damping ratio and its modal mass.
5) Tab `simulation parameters` permits to adjust the duration of the simulation in terms of number of tool turns and step per revolution $d\theta$ (a minimum of 30 steps per turn is highly recommended to predict stability). The axial refinement of the surface linked to tool spatial discretisation can also be modified within the tab. Default values are buried in `Dystamill` if no information is given for the discretisation. A drop menu offers user a selection of three types of workpiece: the default rectangular block, the same block with a pre machined corner or the possibility to load its own blank. In the latter case, user must provide a text file whose extension is "name[i].sf1" containing a list of (x, y) coordinates delimiting a closed curve, [i] being the number of the considered slice. Lastly, checkboxes display options to save or not the machined surface, the cutting forces and a log file.

3.3 `Dystamill` modules

1. `Datatype`

As its name indicates, the `Datatype` module is a "data" type module. More specifically, the variables are defined from eight global structures. The assignation of their value is carried out by functions found inside the `Processing` module. Hereafter are the descriptions of the height structures.

1.1 Structure `t_mill` gathers general information that can be associated with an end mill of any type, namely the number of teeth, their runout and their pitch. It is a frontend to the next structure completing the definition of the tool around the tool envelope.
1.2 Structure `t_solid_mill` contains variables to clarify the tool geometry as defined in [11]. The contained information permits to completely define the cutting edge profiles.
1.3 Structure `t_RDOC` provides parameters for the definition of the radial depth of cut a_e such as the entry φ_{in} and exit φ_{out} angles of the mill into the workpiece. For instance in Fig. 11, the entry angle is 90° and the exit angle is 30°.
1.4 Structure `t_material` contains variables to specify the chosen cutting force model and its parameters such as the cutting force coefficients $K_{i,c}$ and the edge force coefficients $K_{i,e}$.
1.5 Structure `t_structure` refers to variables related to the representation of the machine tool through its modal model.
1.6 Structure `t_points` gathers variables to represent the coordinates of one point of the doubly linked list used to model the workpiece geometry.
1.7 Structure `t_parameters` holds variables related to the milling operation (feed rate, depths of cut, spindle speed, direction of cut,...) and to the simulation (number of tool revolutions, tool angular step, height of tool slice,...).
1.8 Structure `t_results` contains all simulation results, namely a list of variables representing the cutting forces, the vibrations at tooltip, the chip thickness and the roughness. A boolean variable simply tells user whether the simulation was stable or not.

2. `Translation`

The second and last "data" type module is called the `translation` module. It defines variables containing all messages to be printed on screen during the simulation or if an error occurs. An option is available to change the language of the displayed messages from English to French by setting the flag `LANGUAGE`. By default, all messages are set to be printed in English (`LANGUAGE=1`).

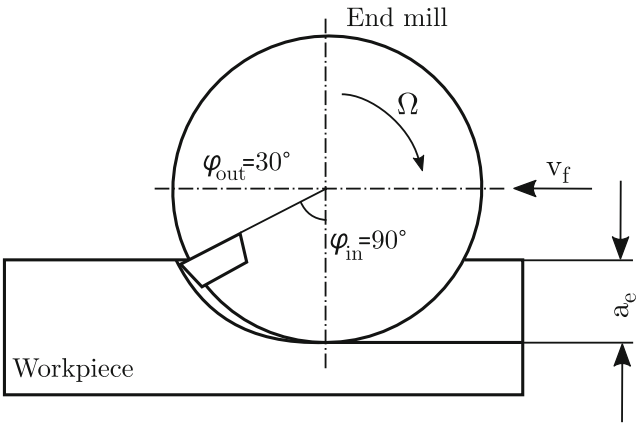


Fig. 11 Entry and exit angles of the end mill

3. `Processing`

   The `processing` module is the only "conversion" type module. Indeed, its main functions `compute_default_parameters()` and `preliminary_computation()` aim to compute the values of operational variables from default or input data. Nevertheless, the module is also versatile and contains other functions in charge to load, save or display the variable values coming from the `Datatype` structures. The load of the parameters is carried out by function `load_dynamic_simulation()`. During data loading, all unit conversions are undertaken towards the International System of Units since numerical data are entered into practical units. For instance, cutting force coefficients are entered in [MPa] and directly converted in [N/m^2].

4. `Tools`

   The `tools` module is a "routine" type module. It mainly provides functions which can be regarded as mathematical tools, namely the computation of normalised values, the filling of vectors with random numbers and the resolution of quadratic equations. The GSL is used to build the functions. Lastly, functions to handle files and initialise tables are also offered.

5. `Varpitch`

   From the user data concerning the tool, the `varpitch` module provides functions to generate the cutting edges wrapped around the end mill envelope. Function `geometry_cutter()` is called to process the tool modelling. Cutting edge coordinates are consequently obtained at any point of the tool. The number of tool slices can be either set by the user through a maximum slice height (spatial discretisation) or defined by default values, i.e one single slice for a cylindrical end mill without helix angle, computed with Eq. 9 for a cylindrical end mill with helix angle $i(z)$ or ten slices for the other types of end mills.

$$N_{slices} = \frac{a_p}{\frac{2\pi\frac{D}{2}}{tan(i(z))\cdot 360}} \tag{9}$$

6. `Surface`

   The `surface` module implements a generation surface algorithm drawn from Peigné [21]. This "eraser of matter" states that the material area swept by the cutting edge(s) at each time step is simply removed from the part. The algorithm allows computing the chip thickness and the profile of the machined surface without approximations and for each slice of material. First of all, the workpiece surface is created by calling function `create_surface()`. If user has selected in the GUI to start milling a block with a pre machined corner, the shape of the surface to machine is such that it outlines the curvature of the end mill (Fig. 12). Such a profile allows starting the milling simulation in steady state. Otherwise, the end mill engages the corner of the block as it is the case in reality. When starting any milling simulation, the end mill, represented by its centre and its edges, is offset by one feed per tooth and one mill radius from the workpiece.

When the removal of material has been initiated, three cases can occur during the update of the workpiece geometry. By designating O, the position of the centre of the mill, A_0, A_1 and A_2; the successive positions of one of the cutting edges respectively in t, t_{-1} and t_{-2}; and the different scenarios, depicted in Fig. 13, may arise. Only positions A_0 and A_1 are necessary to detect in which scenario the considered cutting edge is. Position A_2 will be later used in the material removal phase (`bezier` module).

1) *One of the cutting edges enters into the material.*

   By definition, the entrance of one of the cutting edge into the material is characterised by the detection of an intersection with the surface. Point A_0 is inside the material whereas point A_1 is outside. It is then possible to define two other points in order to remove the corresponding chip: point B which represents the intersection between segment $|OA_0|$ and the surface and point C which marks the intersection between the interpolation of the tooth positions and the surface. The chip bounded by points A_0, B and C is eventually removed from the workpiece geometry.

2) *One of the cutting edges is located inside the material.*

   It is the most common scenario encountered. During one time step, the considered cutting edge remains into the material. In this case, both points A_0 and A_1 are inside the material and point B still represents the intersection between segment $|OA_0|$ and the surface. Point A_1 coincides with

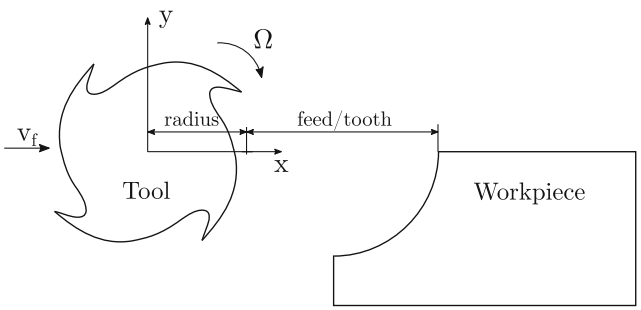


Fig. 12 Workpiece with a pre machined corner

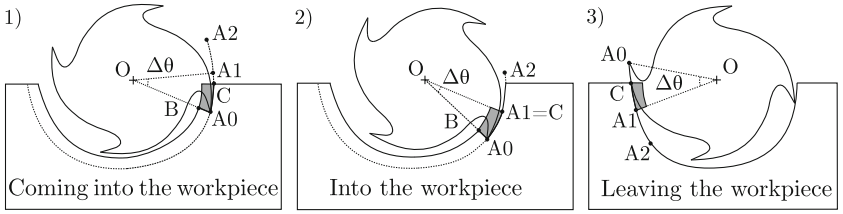


Fig. 13 Update of the workpiece geometry

point C, thus reducing the computational load since the intersection search is time consuming. At the end of the operation, the chip is removed from the workpiece geometry as before.

3) *One of the cutting edges leaves the material.*

The exit of the considered cutting edge from the workpiece is detected when point A_0 is outside the material while point A_1 is still inside. Point B is consequently undefined since there is no intersection between segment $|OA_0|$ and the surface. The update of the workpiece geometry consists in finding point C in order to remove the chip delimited by points A_1 and C.

In `Dystamill`, the function updating the workpiece geometry is called at the end of each time step and is named `update_surface()`.

7. `Bezier`

The interpolation between points A_0, A_1 and A_2 (Fig. 13) is carried out by a Bézier spline using the `bezier` module in order to preserve the curved shape of the chip. As the surface profile is modelled by linked points, it is necessary to interpolate the successive positions of the edges for the purpose to rearrange the coordinates of the point list. For instance, in the first case exhibiting the entering of the tool into the workpiece, points representing the surface between points A_1 and B are deleted and replaced by new ones equally spaced along the spline.

8. `Linked`

As suggested before, the workpiece profile is mapped with coordinate points linked together by the `linked` module. A doubly linked list stores all the points. This structure of data was selected due to the frequent modifications being made to the linked list. The inclusion or the suppression of elements is therefore facilitated and faster. The second reason is related to the different search of intersection with the workpiece when the profile is updated. With a list of points linked in both directions, the intersection search can be made in both ways, thus also reducing the operation time (Fig. 14).

9. `Geometric`

The `geometric` module implements functions to determine the locations of points B and C during the computation of the chip thickness. As just mentioned, the intersection search with the workpiece surface is the operation demanding the largest computation time as it is sometimes necessary to scan the entire profile. To speed up the intersection procedure, it is possible to exclude from the intersection search some *a priori* cases which could not lead to an update of the geometry. For that purpose, the extreme positions around the workpiece are saved at the start of the simulation (Fig. 15) and the intersection will be activated when the cutting edge coordinates will be within the limits. Function `compute_chip_thickness()` pertaining to the `surface` module calls intersection function `line_bezier_intersection()` of the `geometric` module.

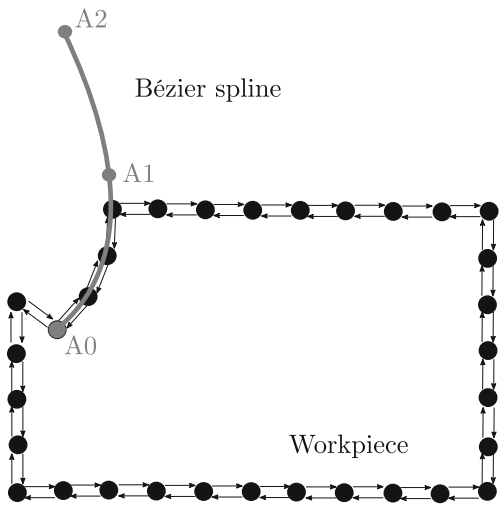


Fig. 14 Doubly linked list of coordinate points

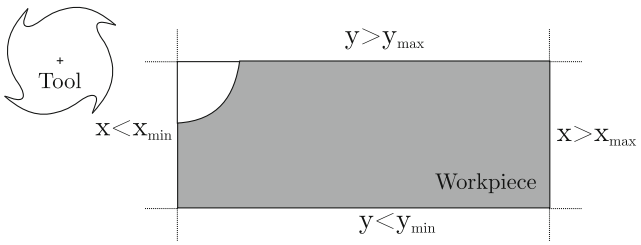


Fig. 15 Intersection detection with the workpiece

10. `Common`

    The `common` module is one of the two modules that implement functions needed to compute cutting forces. Cutting coefficients K_i, derived from the orthogonal/oblique transformation or directly entered by user for the Altintas' and the Kienzle's models, are used to compute the local cutting forces dF_i accounting for the computed chip thickness. The `common` module implements the classical orthogonal/oblique transformation. More specifically, the two angles describing the oblique cutting, namely the rake angle α and inclination angle β, are used to derive local, radial and axial cutting coefficients [22]. Then, using the assumption of physical models, cutting forces can be deduced from the quantity of material removed (`forces` module).

11. `Forces`

    The `forces` module is the other module that implements functions to compute cutting forces. Resultant cutting forces (F_x, F_y and F_z) are obtained from successive calls of function `compute_forces()` within the loop that achieves the spatial integration over the number of teeth and tool slices. Module `forces` also provides a function to compute the maximum cutting force F which is simply expressed as follows:

$$F = \sqrt{F_x^2 + F_y^2 + F_z^2}. \tag{10}$$

12. `Compute`

    The `compute` module is devoted to the time integration of the modal model accounting for the cutting forces. Each component mode is integrated on the basis of the Newmark's scheme presented through (7) and (8). Function `linear_newmark_integration()` gathering the Newmark's formula is called from function `displacement()` which handles the n decoupled equations of motion. The TCP position is eventually corrected by the end of each time step in order to remove the appropriate quantity of material from the workpiece by calling function `update_surface()`.

3.4 Post processing of the results

The post processing can be managed by the graphics functions of Matlab™ or by any other graphing utility since results are organised in text file with different extensions depending on the variable to plot. Once simulation is completed, `Dystamill` generates the result files according to the flags ticked in the `GUI`. Generally, common quantities such as the time [s], the rotation angle θ [rad], the spindle speed [RPM], the spindle torque [Nm], the spindle power [W] and the tooltip vibrations, i.e. position [m], velocity [m/s] and acceleration [m/s^2]. Other results related to the tool geometry and the machined surface can be obtained by ticking the flag "*save surface*" in the `GUI`.

Along with the provided Matlab™ routines, it is possible to visualise the surface finish for roughness assessment (Fig. 16) (.sf2 file(s) for the surface shape), the cutter geometry (Fig. 17) (.geo file for the cutter geometry and .hel file for the edge profile) and the machined workpiece (Fig. 9).

4 Experimental validation

Although general milling applications involve the machine tool dynamics, this first illustration is focussed on a static case. The system is assumed infinitely stiff in order to draw the attention on the time evolution of the cutting forces without any machine disturbance. It was first simulated by Budak [22] and validated experimentally using a cylindrical end mill in half immersion conditions. Table 1 summarises the cutting parameters.

Budak et al. performed several orthogonal cutting and milling tests on titanium alloy ($Ti6Al4V$) in order to identify the cutting force $K_{i,c}$ and edge force coefficients $K_{i,e}$. The simulation of this example using `Dystamill` relies on the orthogonal/oblique transformation to compute the cutting forces. Hence, the former coefficients can be estimated from both the rake α and the friction β angles and from the shear stress τ (Table 2). All the parameters are eventually entered into the `GUI` to generate the *.dat* file.

From a programming point of view, `Dystamill` functions are simply called from a `main()` script listed below. After having submitted the input file *.dat*, all the parameters are loaded through function `load_dynamic_simulation()`. User has next the choice of the criterion on which the stability will be assessed by passing through

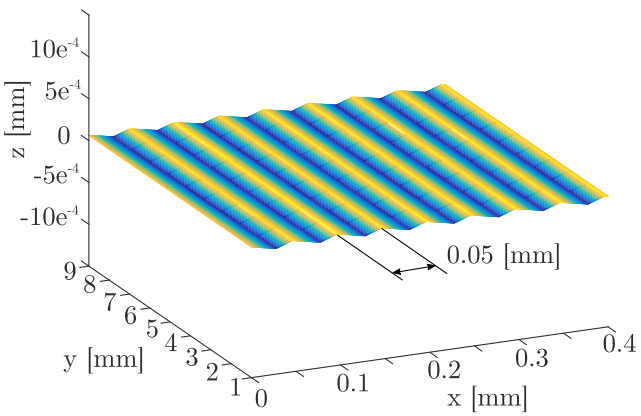


Fig. 16 Roughness evaluation of the lateral face

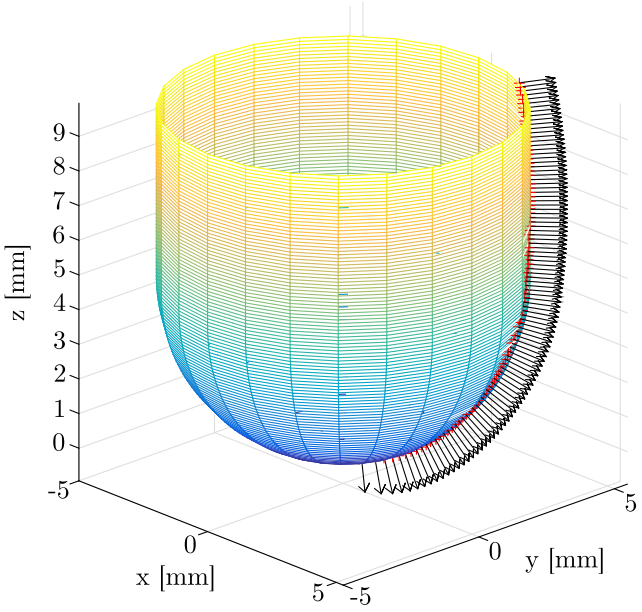


Fig. 17 End mill 3D visualisation

`Dystamill_Preprocess.cpp`. Available stability criteria are either a maximum chip thickness, a maximum force, a maximum vibration or a maximum lateral roughness (R_a, R_q or R_t), the threshold being defined at the beginning of the simulation by user.

The functions achieving the data loading step and the preliminary computations of the operational variables are gathered in `Dystamill_Init.cpp`. The simulation loop calling functions to move the end mill into the material, to compute the chip thickness and the cutting forces, to solve the equations of motion and lastly to update of the workpiece geometry is located in `Dystamill_Milling.cpp`. Once the simulation is over, the roughness of the lateral face can be computed based on the machined workpiece profile for each slice `Dystamill_PostProcess.cpp`. Finally, the stability of the simulation can be appraised by calling function `is_stable()` (`forces` module) comparing the generated results to the stability threshold(s). In the present example, as the system is static, no instability can occur;

Table 1 Cutting parameters in static case Budak 1996 [22]

Simulation parameters	
Cutter diameter	19.05 mm
Number of edge(s)	4
Helix angle	30°
Feed	0.0127 mm/tooth
Spindle speed	527 RPM
Axial depth of cut	5.08 mm
Number of slices	51
Down-milling	Half immersion

Table 2 Cutting coefficients for static case E.Budak 1996

Cutting coefficients	
α	0°
β	19.1°
τ	613 MPa
$K_{t/r/a,e}$	24/ 43/ 0 MPa

consequently, its usage will be more appropriate for the dynamic simulation example presented in Section 4.

Figure 18 depicts the time history of cutting forces F_x, F_y and F_z obtained from the simulation over one tool rotation from 0° to 360°. Collected values are the same as those found in the reference paper of Budak et al. (Fig. 19). As expected, cutting force periodicity is due to the recurrent cutting edge entries into the workpiece. Lastly, on the perspective of simulation time, it takes less than 1 s to simulate three tool revolutions with a refinement of 360 steps per revolution.

The validity of the implemented models was attested by confronting simulation and experiment. The test consisted in a shoulder milling operation with a radial depth of cut of 1 mm into a steel block (St 52-3).

In this case, the cutting tool featured a runout of 2.2 μm which was accounted thanks to the `Dystamill` library. Measurements of cutting forces F_x, F_y and F_z were conducted with a fixed dynamometer. Cutting force coefficients were then identified using an inverse model [23]. For information purposes, Table 3 displays the cutting parameters used to fit a linear cutting force model (5 without edge force coefficients) to the measurements.

As displayed in Fig. 20, cutting forces are well predicted in steady state and the effect of the runout is finely captured. The fitting along z-axis is less accurate since the level

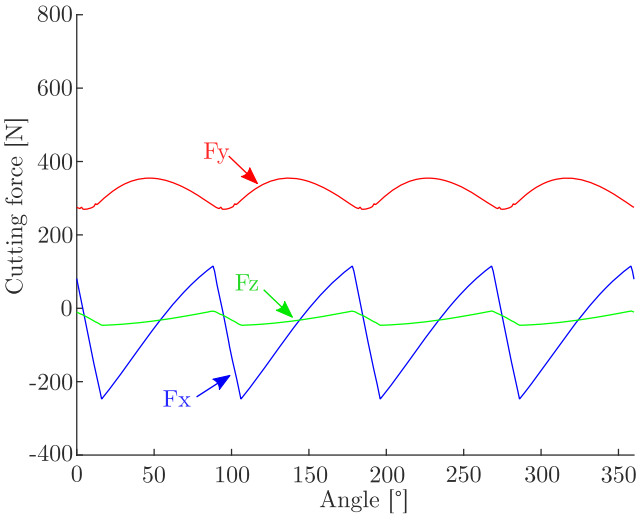


Fig. 18 Simulated cutting forces: static case

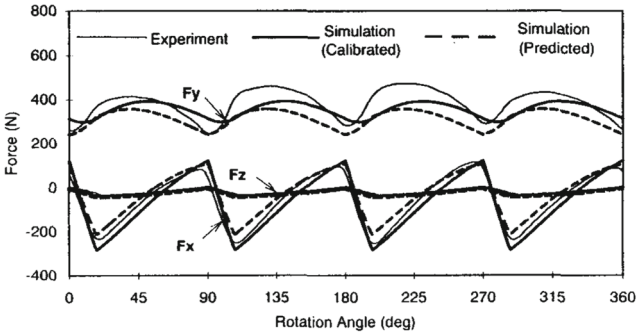


Fig. 19 Original cutting force signals from Budak [22]

of the measured force is usually low, resulting in a poor identification of cutting coefficient $K_{a,c}$.

5 Dynamic milling example

The `Dystamill` framework was set up to predict potential milling instabilities that may occur since a machine tool is a dynamic system. It may not look that significant; nevertheless, the accurate simulation of unstable machining operations is an important field of study in order to develop precise detection of specific vibration and functions to recover stability [24]. Classically, milling simulations are said unstable if the chip thickness exceeds by 25 % its size in static case. This criterion is applied to the illustrative example and the section thus presents a kind of benchmark to verify whether the machine tool dynamics is sufficiently detailed with a modal decomposition.

5.1 Example of unstable simulations

The studied system comprises one single degree of freedom which is the vibration of the end mill along the feed direction. As mentioned, the machine tool is modelled as a single mass-spring-damper system along the x-direction

Table 3 Identified parameters to fit to measurements

Simulation parameters	
Cutter diameter	2 mm
Number of edge(s)	2
Helix angle	20°
Feed	0.01 mm/tooth
Spindle speed	11940.1 RPM
Axial depth of cut	2 mm
Radial depth of cut	1 mm
$K_{t/r/a,c}$	2511.6/1921.9/106.5 MPa
Number of slices	201
Down-milling	

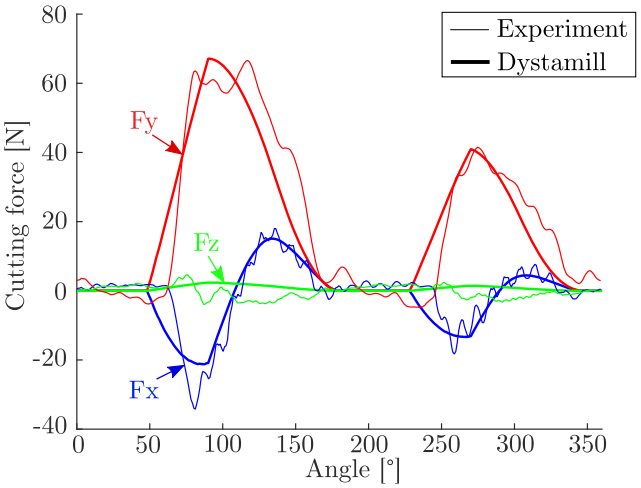


Fig. 20 Experimental validation

(Fig. 5). The whole moves at a constant feed rate v_f and the end mill rotates at a constant angular velocity Ω into the workpiece made in 7075-T6 aluminium alloy. Thus, only cutting force F_x reacts towards the compliant system. The benchmark is inspired from an example of T. Insperger [5] whose simulation parameters and modal properties are brought together in Table 4.

Three specific milling conditions are simulated in order to reveal three different configurations for the cutting force time evolution. To be more precise, only the spindle speed will vary and successively become 16000, 19000 and 22000 RPM. Figure 21 plots the three resulting behaviours for the total cutting force (Euclidean norm).

– At 16,000 RPM, the dynamic system is driven into an instability known as the *Hopf* instability. The dynamics of the system, and more particularly the tooltip vibrations, become chaotic. The total cutting force level increases sharply and the frequency content of such a

Table 4 Parameters for benchmark Insperger 2003 [5]

Simulation parameters	
Cutter diameter	10 mm
Number of edge(s)	1
Helix angle	0°
Feed	0.05 mm/tooth
Spindle speed	16,000|19, 000|22,000 RPM
Axial depth of cut	2 mm
Number of slices	1
Up-milling	Half immersion
$K_{t/r/a,c}$	550/200/0 MPa
Modal mass m	2.573 kg
Damping ratio ξ	0.32 %
Frequency f	146.4 Hz

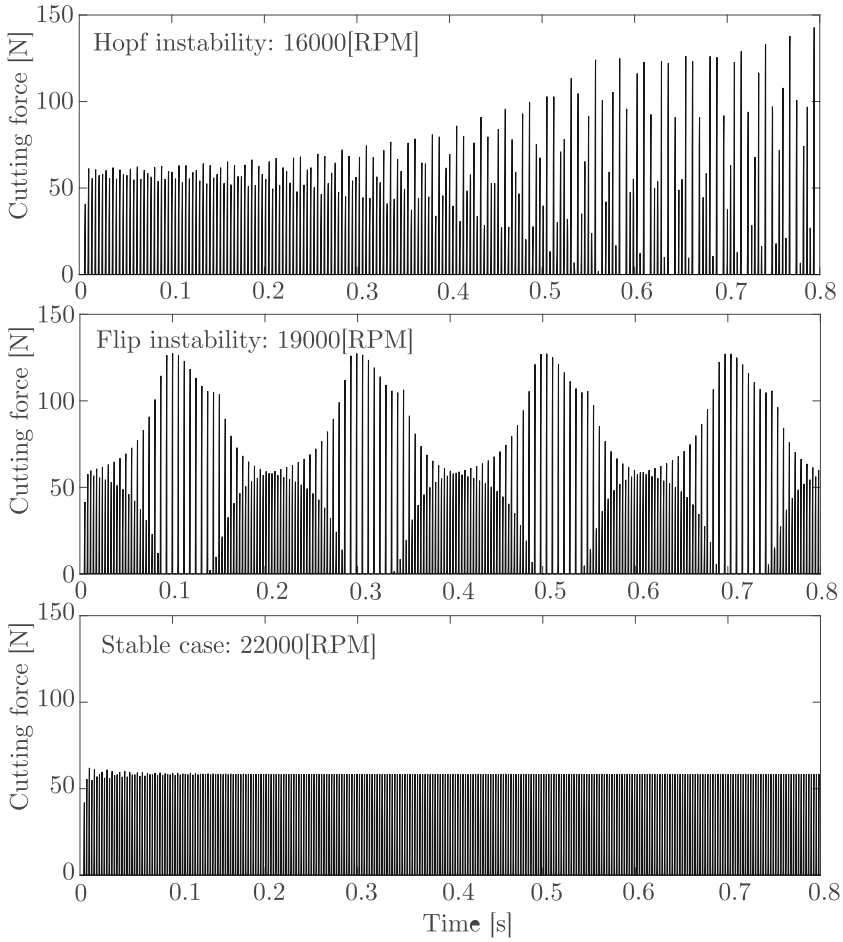


Fig. 21 Simulated cutting forces and instability detection

signal is dominated by the so-called *chatter* frequency close to the machine tool natural frequency.

– At 19,000 RPM, the dynamic system is also exposed to an instability known as the *flip* instability or "period doubling bifurcation". An analysis in the frequency domain shows resonance peaks at dual frequencies. The total cutting force level stands high.
– At 22,000 RPM, the dynamic system reaches stable conditions. The total cutting force level remains constant. As expected for such conditions, the roughness of the lateral face presented in Fig. 16 is smooth and in the range of a few hundredth of a micrometer. The marks left by the end mill on the surface are regularly spaced by 0.05 mm corresponding to the feed/tooth.

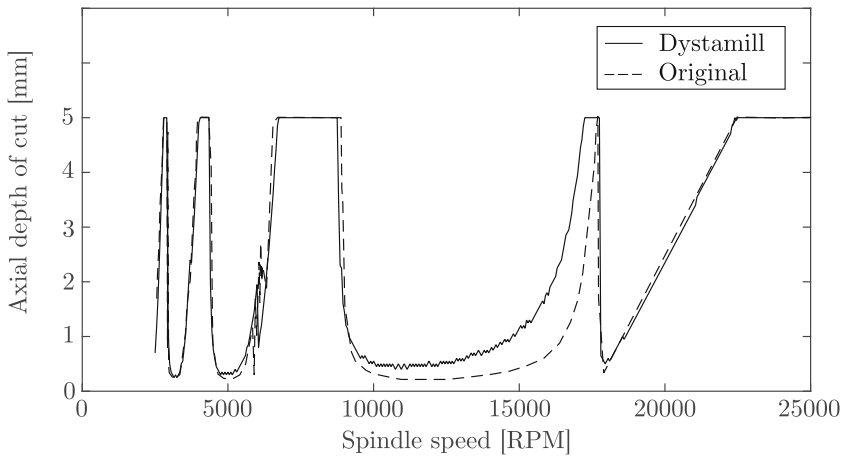


Fig. 22 2D stability lobes using the chip thickness criterion (original from [5])

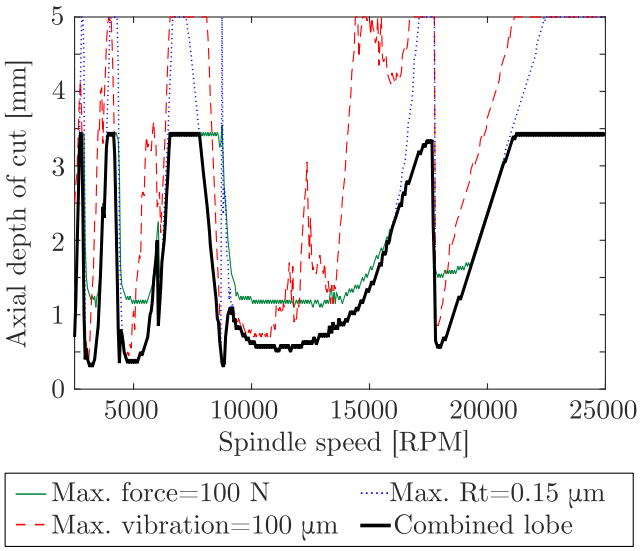


Fig. 23 Combined stability lobe diagram

From those simulations, it can be inferred that the `Dystamill` library is able to predict milling instabilities. As a complementary note, each simulation took approximately 1 s with a 16-Gb RAM laptop fitted with an i7-4700HQ processor.

5.2 Simulation of stability lobes

The validation of this benchmark, on the basis of the results obtained by T. Insperger, was performed by plotting the stability lobe diagram. It is an elegant approach to figure out the behaviour of a milling operation. The diagram, displaying the axial depth of cut with respect to the spindle speed, highlights a pattern delimiting the stable zone from which said unstable. Unsurprisingly, the stable zone is the area for which the combination of parameters a_p and Ω will result in a rather constant cutting force level, that is the area below the curve. As usual, the detection of instability can be completed using the aforementioned criteria by providing a certain tolerance on the quantity monitored. Reference results were yielded using the chip thickness criterion and proved to be similar to those obtained by T. Insperger (Fig. 22); the original curve was digitalised in order to overlap the simulated results. The benchmark clearly indicates that coherent results can be delivered using the `Dystamill` library.

Although results were validated using the criterion based on an excess of 25 % in chip size, it is also interesting to combine the different criteria available in `Dystamill` to determine stable conditions respecting all requirements, each criterion leading to its own stability lobe diagram. The superimposition of different stability lobe diagrams leads to the so-called *technological lobes*. It is of interest to note that high-performance cutting conditions can be deduced from that kind of diagrams as stable settings can be found without impacting the workpiece quality.

Returning to the benchmark, three types of stability lobe were superimposed in Fig. 23 using three different criteria:

- The total cutting force being limited to 100 N
- The vibration at tooltip being limited to 100 μm
- The total roughness being limited to 0.15 μm

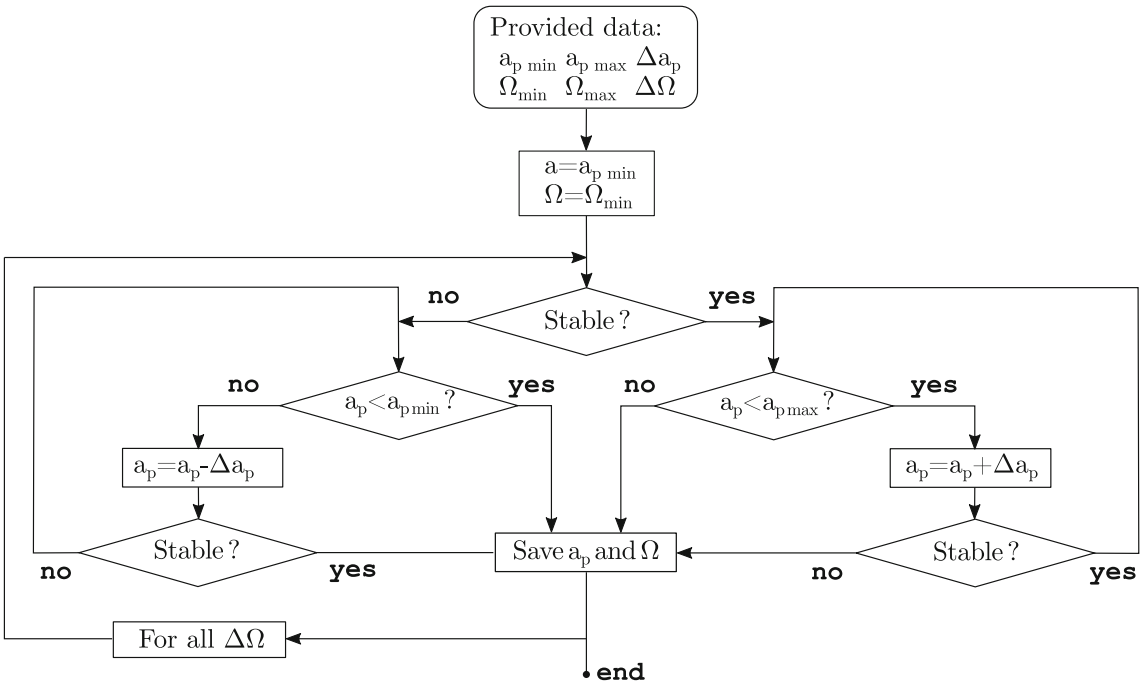


Fig. 24 Stability lobe search algorithm

The resulting curve meeting all requirements was drawn in bold, defined as the minimum of each individual lobe.

From a practical point of view, in `Dystamill`, stability lobe diagrams are generated according to the algorithm presented in Fig. 24. Starting from lower and upper bounds for the axial depth of cut a_p and for the spindle revolution velocity Ω, all their possible combinations are typically tested with respect to a certain step Δa_p and $\Delta\Omega$. To speed up the conventional approach, the proposed algorithm begins with the minimal values for a_p and Ω and performs a complete milling simulation to appraise the stability. Depending on the outcome, the spindle speed Ω remains constant and the axial depth of cut is increased, in case of stable conditions, or decreased, in case of unstable conditions. The process is then repeated until reaching a stability limit or one of the axial depth of cut boundaries, i.e. $a_{p\,max}$ or $a_{p\,min}$. At this point, current axial depth of cut and spindle speed are saved and the next spindle speed is evaluated but starting from the just determined a_p. Proceeding this way allows saving a substantial amount of simulation time. For the record, it took about 180 s to generate all three stability lobes depicted in Fig. 23.

6 More advanced example

The last section is dedicated to a machine tool whose dynamics was modelled with two modes along the x-direction and two other modes along the y-direction. It is inspired from Smith and Tlusty [25]. The operation consists in a half immersion test into aluminium using a cylindrical shell mill. The corresponding system was illustrated in Fig. 5 with springs and dampers in both directions. The simulation parameters are gathered in Table 5.

The 2D stability lobe diagram was plotted as a first step and overlaid with the digitalised original results simulated by Altintas in [26] (Fig. 25). A total of 222 s were needed to complete all the calculations using the algorithm presented in Fig. 24. Although minor discrepancies are noticeable, the overall trend is again well captured by the `Dystamill` library. The rising edge representing the Hopf bifurcation between 2000 and 3500 RPM nearly coincides with the original curve.

Table 5 Milling parameters found in Smith 1990 [25]

Simulation parameters	
Cutter diameter	100 mm
Number of edge(s)	8
Helix angle	0°
Feed	0.1 mm/tooth
Spindle speed	1000→8000 RPM
Axial depth of cut	2→12 mm
Number of slices	1
Up-milling	Half immersion
$K_{t/r/a,c}$	1500/ 450/ 0 MPa
Modal mass $m_{x_1}\|m_{x_2}\|m_{y_1}\|m_{y_2}$	84.7|9.3|239.8|4.5 kg
Damping ratio $\xi_{x_1}\|\xi_{x_2}\|\xi_{y_1}\|\xi_{y_2}$	12|4|10|10 %
Frequency $f_{x_1}\|f_{x_2}\|f_{y_1}\|f_{y_2}$	260|389|150|348 Hz

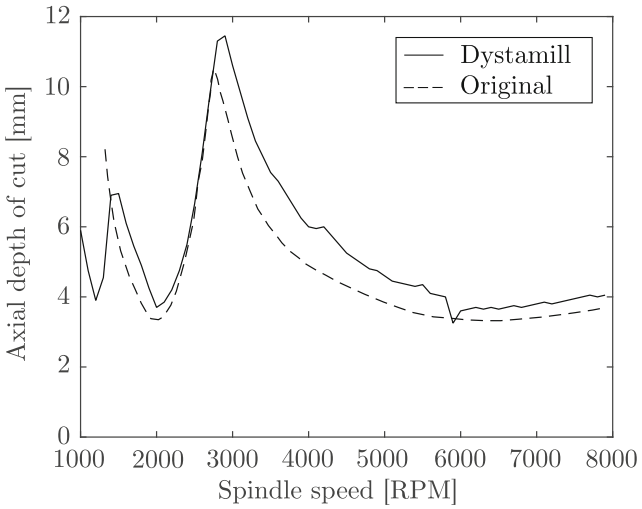


Fig. 25 Comparison with the original 2D stability lobes (original from [26])

A complete stability lobe diagram was eventually simulated using the `Dystamill` library, i.e. the stability of all combinations of a_p and Ω was assessed. Furthermore, for each simulation, the maximum cutting force was backed up leading to a 3D stability lobe diagram (Fig. 26).

It is worth noting that each a_p and Ω combination was simulated using 40 tool revolutions and 1000 steps per turn. Consequently, it took about 7 h to complete all the 7000 individual simulations [18].

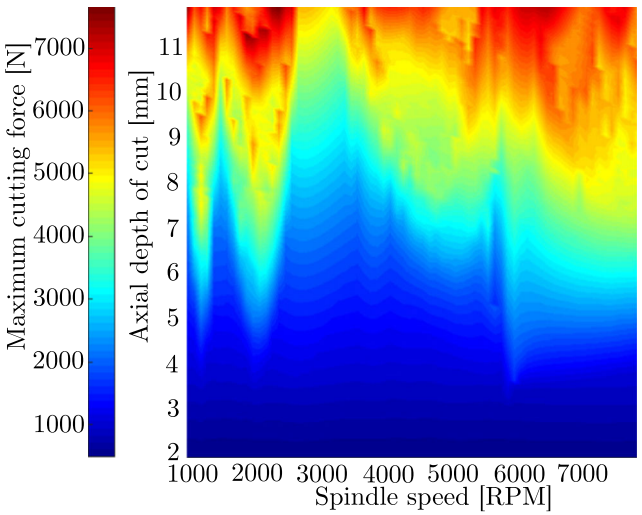


Fig. 26 3D stability lobes

7 Summary

The presented C++ library, called Dystamill, allows to easily perform milling simulations in $2D\frac{1}{2}$. The library is based on one physical model and two mechanistic models to predict the cutting forces thanks to the consideration of the system dynamics and the update of the workpiece geometry. The machine tool dynamics is reduced to the structural modes at tooltip measurable from experiments whereas the workpiece surface is modelled as a list of coordinate points. As the end mill progresses into the workpiece, its geometry is updated and the removed quantity of material provides enough information to predict the cutting forces.

The primary function of Dystamill is the stability assessment of milling operations from cutting parameters. Several criteria of stability detection are implemented and running simulations in loop results in determining the optimal operating conditions ensuring high-added value parts. Hence, the benefit to accurately predict the time evolution of cutting forces and the stability lobe diagrams.

From a practical point of view, the library was organised in 12 modules, each with a specific role. For instance, one module deals with the update of the workpiece geometry while another one is focussed on determining the intersection point between the cutting edges and the part. The function gathered into the modules are then called from a C++ main script. The Dystamill library is now used for at least four years in Master theses and PhD theses at the University of Mons (Belgium) and this paper was the opportunity to deploy a public release of the library. Clearly, the proposed framework is not a replacement of available commercial software but offers useful tools to simulate milling operations. Authors are also convinced that Dystamill constitutes a good foundation for the development of "proof of concept" applications or an extension module to be coupled with a multi-physics environment.

The library can be downloaded from the following website: http://hosting.umons.ac.be/html/dystamill.

Acknowledgements The authors would like to acknowledge the Belgian National Fund for Scientific research (FNRS-FRS) for the FRIA grant allotted to H.N. Huynh.

References

1. Tobias SA, Fishwick W (1958) The chatter of lath tools under orthogonal cutting conditions. Trans ASME 80:1079–1088
2. Tlusty J, Polacek M (1963) The stability of the machine tool against self-excited vibration in machining. ASME Int Res Prod Eng, 465–474
3. Altintas Y (2000) Manufacturing automation: metal cutting mechanics, machine tool vibrations and CNC design, 1st edn. Cambridge University Press, USA
4. Altintas Y, Budak E (1995) Analytical prediction of stability lobes in milling. Annals of CIRP 44:357–362
5. Insperger T, Mann BP, Stépán G, Bayly PV (2003) Stability of up-milling and down-milling, part I: alternative analytical methods. Int J Mach Tools Manuf 43:25–34
6. Long X-H, Balachandran B (2007) Stability analysis for milling process. Nonlin Dyn 49(3):349–359
7. Long X-H, Balachandran B (2010) Stability of up-milling and down-milling operations with variable spindle speed. J Vib Control 16(7-8):1151–1168
8. Balachandran B (2001) Nonlinear dynamics of milling processes. Philos Trans R Soc London Series Math Phys Eng Sci 359:793–819
9. Ducobu F, Rivière-Lorphèvre E, Filippi E (2016) Application of the coupled Eulerian-Lagrangian (CEL) method to the modeling of orthogonal cutting. European J Mech A/Solids 59:58–66
10. Rivière-Lorphèvre E (2007) Study and simulation of high speed machining processes: cutting forces, stability, surface finish; Etude et simulation de procédés de fraisage grande vitesse : efforts de coupe, stabilité, états de surface. PhD Thesis, Faculty of Engineering UMONS (Belgium)
11. Engin S, Altintas Y (2001) Mechanics and dynamics of general milling cutters Part I: helical end mills. Int J Mach Tools Manuf 41:2195–2212
12. Compomanes ML, Altintas Y (2003) An improved time domain simulation for dynamic milling at small radial immersion. Trans ASME 125:416–422
13. Childs THC, Maekawa K, Obikawa T, Yamane Y (2000) Metal cutting: theory and practice. Arnold Publishers
14. Merchant E (1945) Mechanics of the metal cutting process. I. Orthogonal cutting. J Appl Phys, 16. https://doi.org/10.1063/1.1707586
15. Shamoto E, Altintas Y (1999) Prediction of shear angle in oblique cutting with maximum stress and minimum energy principle. J Manuf Sci Eng 121:399–407
16. Kienzle O, Victor H (1957) Spezifische Schnittkrfte bei der Metallbearbeitung. Werkstattstechnik und Maschinenbau 47:22–25
17. Altintas Y, Lee P (1996) A general mechanics and dynamics model for helical end mills. CIRP Ann Manuf Technol 45:59–64. https://doi.org/10.1016/S0007–8506(07)63017–0
18. Rivière-Lorphèvre E, Huynh HN, Verlinden O (2018) Optimal time step selection on dynamic simulation of milling operation. International Journal of Advanced Manufacturing Technology, https://doi.org/10.1007/s00170-017-1570-9
19. Galassi M et al GNU Scientific library manual, 3rd edn. ISBN 0954612078
20. Heikkinen Jani Qt 5.9.1 Released. In: Qt Blog. The Qt Copany.
21. Peigné G, Paris H, Brissaud D (2003) A model of milled surface generation for time domain simulation of high-speed cutting. Proc Institut Mech Eng 217:919–930
22. Budak E, Altintas Y, Armarego EJA (1996) Prediction of milling force coefficients from orthogonal cutting data. J Manuf Sci Eng 118(2):216–224
23. Rivière-Lorphèvre E, Letot C, Ducobu F, Dehombreux P, Filippi E (2017) Dynamic simulation of milling operations with small diameter milling cutters: effect of material heterogeneity on the cutting force model. Meccanica 52:35–44
24. Kuljanic E, Sortino M, Totis G (2012) Multisensor approaches for chatter detection in milling. J Sound Vib 56:10–16
25. Smith S, Tlusty J (1990) Update on high speed milling dynamics. Ann CIRP 39:517–521
26. Altintas Y, Budak E (1995) Analytical prediction of stability lobes in milling. Ann CIRP 44:357–362