ACTA IMEKO
ISSN: 2221-870X
December 2020, Volume 9, Number 4, 30 - 38

An advanced GCode analyser for predicting the build time for additive manufacturing components

Luca Di Angelo[1], Paolo Di Stefano[1], Emanuele Guardiani[1]

[1] *University of L'Aquila, Department of Industrial and Information Engineering and Economics, Monteluco di Roio, 67100 L'Aquila, Italy*

ABSTRACT
Additive manufacturing is a technology for quickly fabricating physical models, functional prototypes, and small batches of parts by stacking two-dimensional layered features directly from computer-aided design data. One of the most important challenges in this sector relates to the capability to predict the build time in advance, since this is crucial to evaluating the production costs. In this paper, an accurate method for obtaining build-time is proposed. This method is based on an advanced GCode analyzer written in Python following an object-oriented paradigm for scalability and maintainability. Various examples are used to demonstrate the reliability of the algorithm, while its potential applications are also illustrated.

Section: RESEARCH PAPER

Keywords: Build time estimation; manufacturing costs; process planning; additive manufacturing

Citation: Luca Di Angelo, Paolo Di Stefano, Emanuele Guardiani, An advanced GCode analyzer to predict build-time in AM components, Acta IMEKO, vol. 9, no. 4, article 5, December 2020, identifier: IMEKO-ACTA-09 (2020)-04-05

Section Editor: Leopoldo Angrisani, University of Naples Federico II, Italy

Received October 15, 2019; **In final form** January 17, 2020; **Published** December 2020

Corresponding author: Emanuele Guardiani, e-mail: emanuele.guardiani@graduate.univaq.it

1. INTRODUCTION

The use of additive manufacturing (AM) technologies is growing day by day due to their various advantages [1]. For one, they allow for creating designed objects using new and innovative shapes, which could not be achieved via traditional manufacturing processes [2]. This is the case with the shape optimisation of structural components in which the unstressed material can be removed to reduce the weight [3], [4]. Moreover, since the AM processes are well integrated with computer-aided design (CAD) instruments, the pre-processing for the physical creation of a given part requires less time compared to classical subtractive technologies. This property presents a key feature for companies that work in a competitive industry in which the reduction of time-to-market determines the competitivity of the company. However, AM technologies are still expensive and the manufacturing process is significantly longer than that of the classical subtractive manufacturing technologies [5]. In a competitive market of AM services, manufacturing costs must be estimated in a reliable way [6]-[8], which is why the accurate estimation of build times is crucial. A reliable quantification of build time also aids ascertainment of the deposition direction, which minimises the manufacturing costs [9]-[23].

In view of this, several studies have been conducted over the few last years that have led to two different strategies for estimating the build time.

The first strategy involves performing a detailed analysis of the manufacturing activity and is regarded as the most reliable approach. Meanwhile, the second strategy involves performing the build time estimation according to appropriately defined parametric functions in which the independent variables are a number of build-time driving factors.

The detailed-analysis-based methods use complete information related to the geometry of the object and to the manufacturing process. In using this method, a highly accurate estimation of the build time could potentially be obtained. However, complete information is required for the evaluation, as is a detailed analysis of the manufacturing activity. For these reasons, the detailed analysis approach is computationally expensive, and more time is required for the build time estimation.

Meanwhile, while the parametric-based methods are less accurate than the first approach, they require fewer data as input and are less expensive in computation terms. The parametric-based methods make use of certain pertinent build time factors, which serve as the independent parameters used in the functions to evaluate the dependent variable, that is, the build time. These parameters can be computed through analysing the geometric

model of the object to be manufactured (volume, bounding box, etc.). However, the most challenging aspect of this type of method lies in identifying these factors. The set of parameters used should take into account all the elements that affect build time, while they must also be mutually independent such that any cross-correlation is avoided. The parametric-based methods would appear to be more promising for many practical applications, such as in the case where an accurate prediction of the build time is required but a limited set of data describing the object is available. An example here is the budgeting process where the customer may not want to provide the full geometrical model of the object to the seller in order to protect their intellectual property but instead will provide a few parameters that affect cost [24]. Moreover, build time estimation is a mandatory step for any optimisation method devoted to ascertaining the best build direction. It is for these reasons that the parametric approaches have been largely proposed as one element of far more complex methods devoted to ascertaining the optimal manufacturing build direction. In [25], a very simplified model is proposed, in which the build time is proportional to the number of layers in the sliced model. Meanwhile, in [21], [23], [24], [26]-[28], more complex formulations of the build time are proposed, with the build time dependent on the volume of the object, the supports, and other geometrical features of the object. While they incorporate many more parameters, the limitation of these methods lies in the function describing the build time, which is linear. The relationship between build time and the attendant driving factors is highly complex and largely unknown. In order to take this issue into account, the use of an adaptive model based on grey modelling has been proposed [29], while an artificial neural network for identifying the relationship between the driving factors and the build time was proposed in [30]. Here, the authors demonstrate that a highly accurate estimation of the build time can be obtained using these methods. However, the drawback of both approaches is that they require a large set of model samples. In order to provide a set of training samples, a number of factors driving build time and the actual build time must be examined. The accuracy of the estimation performed by the neural network, for example, increases with the number of test samples, meaning a large number of actual build time evaluations is required. Clearly, this is neither economically nor temporally feasible.

In order to define a significative set of samples, an accurate detailed-analysis-based method could be used in view of performing a less expansive evaluation of the build time in relation to a real experiment. Such a detailed analysis can be operated by using specialised computer-aided engineering (CAE) programs, such as those supplied with the AM machine. An example here is Simplify 3D, which is a 3D printing slicing software that controls every aspect of the printing process and also performs build time evaluations. Alternatives include Cura by Ultimaker and MatterControl by MatterHackers, both of which are freeware. Nevertheless, the build time estimations provided by this software differ from those performed using a fused deposition modelling (FDM) machine (German RepRap X350). The comparison of the estimated build time with the real situation is reported in Table 1; it involved using the three test cases shown in Figure 1.

Table 1. Comparison between actual build time and estimations provided by professional CAE software for the models reported in Figure 1.

Model	Software	Actual	Estimated	*Error*
A	Simplify3D	1532 min	1344 min	*−14 %*
B	Cura	374 min	242 min	*−54 %*
C	MatterControl	326 min	286 min	*−14 %*

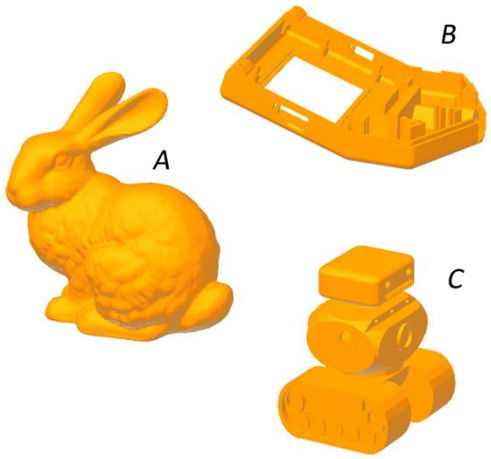

Figure 1. Reference models used for time comparison.

The estimation performed was not accurate enough to be used for obtaining qualified training data for a neural network. The main reason for the evident mismatch in the aforementioned programs is that they do not consider certain process-related parameters of the machine in use, namely, 'acceleration' and 'jerk'.

The control strategy performed by certain machine firmware, such as the open-source RepRap, uses acceleration strategies defined by the two parameters (jerk and acceleration) that widely affect build time and object quality. This is shown in Figure 2, where two cubes of the same dimension were manufactured by a German RepRap X350 using different acceleration parameters. In Figure 2a a module of 300 mm/s2 was selected for the *x-y* axis, while in Figure 2b, twice the value was used.

A comparison between the two objects indicated that the quality of the corners of model (a) was significantly better than that of model (b). This was due to the different inertia forces, which affected the accuracy of the material deposition performances, especially on the corners.

In order to overcome all the previously discussed limitations of the implemented detailed-analysis-based methods, a new method for build time evaluation is developed in this research. The approach followed was a relatively general purpose approach, meaning the method can be used in various AM

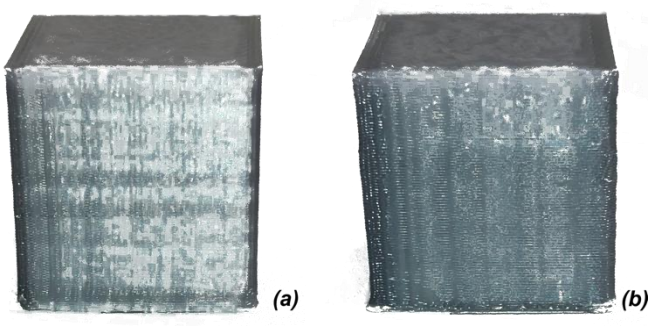

Figure 2. Comparison between two cubes based on the same nominal geometry and manufactured using a different acceleration value.

applications. The method involves performing GCode analysis, taking into account the most important process-related parameters. As such, a highly accurate estimation of the actual build time can be obtained. In the next sections, the method is explained in more detail. A number of test cases are used to test the proposed approach, with the results critically discussed.

GCode

GCode is the most widespread programming language used for giving instructions to a computer numerical control (CNC) machine. It consists of a series of textual instructions (word address) devised in accordance with the ISO 6983 standard, which regulate the behaviour of the machine during the manufacturing process. This includes defining the speed of the axis, regulating the temperature of the extruder, and numerous other aspects. This set of commands is generally called the 'part program'. During the 'first era' of automatic machines, a manual approach was adopted for generating the part program. Here, a highly skilled operator could write the part program alone or, at least, with the aid of visual programming software. Today, the preferred solution consists of an automatic approach based on the integration between the CAD and CAM systems (Figure 3). Following the designing of the object, the geometry is imported into the CAM, where many of the process-related parameters are defined. With reference to AM technologies, these parameters could be represented by the layer thickness, the typology of hatching, the speeds of the motors, and various other aspects. Then, if required by the technology in use, the supports are also generated into the CAM, typically using a graphical procedure. At the end of these steps, the CAM generates the tool path and the post processor transforms all the previous information into a set of instructions, readable by the machine in use, which is the part program, and saved into a textual file that is finally sent to the machine.

Consequently, the part program defines many characteristics of the manufacturing process that will be followed by the machine. However, an analysis limited to the part program is generally not sufficient for establishing the kinematic behaviour of the machine. Many of the attendant decisions are taken in real-time by the controller of the machine. This is the case for acceleration ramps, which are highly dependent on the machine

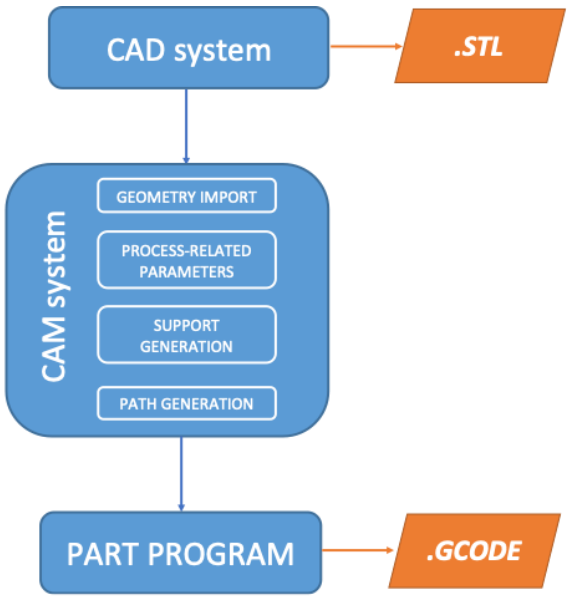

Figure 3. Automatic programming for generating the part program.

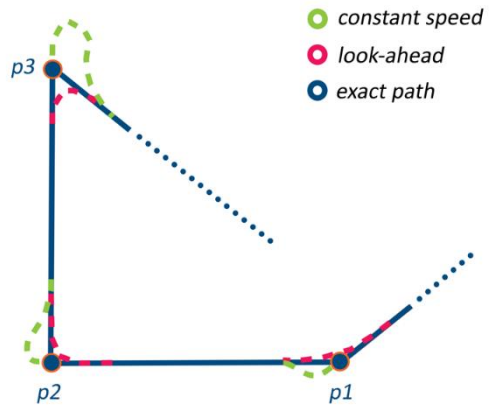

Figure 4. Comparison of tool paths using different acceleration strategies.

in use. Due to the axis inertia and/or torque availability of the motors, any CNC machine is subject to magnitude constraints on the accelerations. Here, three different control strategies can be adopted. Let us suppose that the deposition tool has to follow the path reported in blue in Figure 4, where *p1*, *p2,* and *p3* are three control points interpolated linearly. A first option consists of maintaining a fixed speed along the entire path. This condition represents the optimal solution in terms of time-saving, while a similar behaviour can be realised only theoretically due to the infinite acceleration module required to realise it. In reality, when a fixed speed is adopted (Figure 5), a diverted path (green line of Figure 4) is generated due to the impossibility of realising an infinite module acceleration. In order to avoid a similar behaviour that causes aleatory geometrical errors on the finite piece, another strategy, which is characterised by having a zero speed in correspondence with the control points (blue-dotted line in Figure 5), can be used. In this case, linear acceleration ramps are generally adopted. This allows for obtaining the optimal positioning accuracy, since the theoretic path and the real path are the same.

The drawback of this strategy lies in the increased build time. This issue has led to the formulation of a third approach, which represents a compromise between geometrical accuracy and build time. In this case, the control unit of the machine in use does not limit itself to executing the instructions provided by the GCode; rather it performs a 'clever' evaluation. In addition to the commands in the execution, various subsequent instructions are read and cached. Then, through a vectorial comparison of the velocities, the acceleration ramps are computed in order to maintain the maximum feedrate possible (red-dotted line in Figure 5). This control strategy is typically known as 'look-ahead'

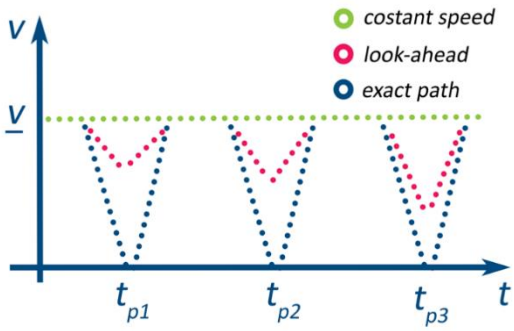

Figure 5. Speed comparison between constant speed, zero-velocity, and look-ahead strategies.

and can be regarded as a multi-objective optimisation problem, where the geometrical accuracy conflicts with the reduction of the build time.

Many researchers have worked on this topic in recent years. Since the related literature is wide and voluminous, the most important and recent methods are outlined here. In [29], it was proposed that each corner is smoothed by replacing a subset of the path that contains it with a conic 'splice' segment, deviating from the exact corner by no more than a prescribed tolerance. In [30], using a fine-interpolation parametric method in which the corners are replaced by arc curves is recommended, while in [31], [32] the B-splines are used to approximate the corners. Meanwhile, in [33], a different kind of approach is used, whereby the control acts on the acceleration ramps instead of the tool paths. The presence of so many different approaches to the 'look-ahead' optimisation can lead to some confusion. This is certainly the case with FDM machines for which, as we verified in the previous section in the case of a German RepRap X350, professional tools are not able to reconstruct the real kinematic behaviour of the machine, leading to a very rough estimation of the build time.

In this paper, a comprehensive investigation of the behaviour of RepRap devices is carried out. The kinematic laws are reconstructed and, lastly, we provide a script for estimating the build time with high accuracy that was written in Python following an object-oriented paradigm for scalability and maintainability.

2. PROPOSED METHOD

The proposed method is based on the analysis of the GCode, the role of which was previously described. Specifically, all the instructions that relate to movement command were considered since these are largely responsible for defining the build time. Other instructions that may contribute to the build time are the temperature commands that, for example, define the temperature of the printing bed. However, for this category of instructions, an exact evaluation of how they contribute to the build time is difficult to attain since they are largely dependent on the environmental conditions. Moreover, their influence is generally negligible with regard to movement commands and they were thus not considered.

The GCode movement commands are described in *Annex E* of ISO 6983, and will thus not be discussed in detail here. As illustrated in Figure 3, as the GCode used for AM applications is generated starting from a triangular mesh (.stl) that is subsequently sliced, most of the analytical information that defines the original geometry is lost during the data exchange. Consequently, the simplest and most efficient way for generating the part program is to define numerous geometrical control points that belong to the model and to interpolate them linearly. As such, most of the movement instructions used for AM applications are linear interpolations, introduced by the code *G01*. The algorithm evaluates each interpolation line using a regular expression matching. Each movement is associated with its relative velocity and length. In Figure 6, the interpolation lines *i* and *i*+1 of a generic GCode file are presented as an example.

```
i)      G1 Xaaa.aaa Ybbb.bbb Ec.cc Fdddd
i+1)    G1 Xeee.eee Yfff.fff Eg.gg
```

Figure 6. Example of two linear interpolation GCode instructions.

The length l_i of path *i-th,* the nominal speed $v1_i$ that should be reached and the length of the extruded filament e_i during the *i-th* step can be formulated as follows:

$$l_i = \sqrt{(eee.eee - aaa.aaa)^2 + (fff.fff - bbb.bbb)^2}$$

$$v1_i = dddd$$

$$e_i = g.gg$$

For each *i-th* step, the build time is provided by the contribution of four terms:

$$t_i = t_{\mathrm{JERK}i} + t_{0i} + t_{1i} + t_{2i} \tag{1}$$

where:

$t_{\mathrm{JERK}i}$ = jerk phase time during the *i-th* step
t_{0i}= acceleration phase time during *i-th* step
t_{1i} = cruise phase time during the *i-th* step
t_{2i}: deceleration phase time during *i-th* step.

To evaluate these terms, the acceleration phase and the jerk phase need to be discussed and illustrated.

The jerk phase is referred to as the 'look-ahead' strategy used by RepRap machines. During this phase, an 'instantaneous' change of speed Δv is applied. This specific behaviour is made possible with the use of stepper motors. Clearly, an 'instantaneous' change of velocity implies high values of acceleration, which means the value of Δv must be limited to avoid both vibration and a loss of steps from the motors. Since the contribution of $t_{\mathrm{JERK}i}$ is generally extremely small compared to the other three terms of formula (1), it is assumed to be negligible.

$$t_{\mathrm{JERK}i} \cong 0 \tag{2}$$

In order to proceed, for each step, a generic *j-th* degree of freedom that satisfies condition (3) is taken as the reference.

$$a_i^j \neq 0 \; j = 1, 2, \ldots, N \tag{3}$$

where a_i^j is the acceleration module of *i-th* step related to the *j-th* degree of freedom, and N is the number of degrees of freedom of the machine. This assumption is justified by the synchronous behaviour that should be exhibited by the axis, that is, the axis should start and stop executing the command of a certain instruction line at the same moment. The following condition can then be evaluated:

$$\left| v2_i^j - v0_{i+1}^j \right| > \Delta v_i^j \tag{4}$$

where $v2_i^j$is the end speed of step *i-th* related to the *j-th* degree of freedom, $v0_{i+1}^j$ is the start speed of step (*i*+1)*-th* related to the *j-th* degree of freedom.

Here, $v2_i^j$ and $v0_{i+1}^j$ were initially computed with reference to the feedrate value reported in the GCode instructions and, generally, they are different from each other. This is due to the fact that from step *i-th* to step (*i*+1)*-th,* the vector $\boldsymbol{v1}$ could change in terms of direction and/or magnitude. If condition (4) is true, the end speed $v2_i^j$ or/and the start speed $v0_{i+1}^j$ need to be revaluated. In this case, two possibilities are defined. If

$$v2_i^j \cdot v0_{i+1}^j > 0 \tag{5}$$

then only one of the two velocities requires updating:

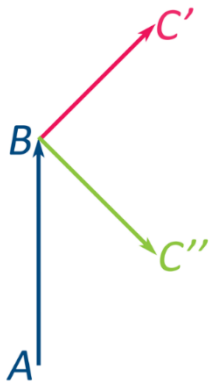

Figure 7. During the jerk phase, the start speed and the end speed of the paths are ingeniously 'overwritten'.

$$\begin{cases} \boldsymbol{v2_i^*} = \dfrac{v0_{i+1}^j + \Delta v_i^j}{v2_i^j}\boldsymbol{v2_i} & |v2_i^j| > |v0_{i+1}^j| \\ \boldsymbol{v0_{i+1}^*} = \dfrac{v2_i^j + \Delta v_i^j}{v0_{i+1}^j}\boldsymbol{v0_{i+1}} & |v2_i^j| < |v0_{i+1}^j| \end{cases} \tag{6}$$

Otherwise, both velocities are recalculated in the following way:

$$\begin{cases} \boldsymbol{v2_i^*} = \dfrac{\Delta v_i^j}{v2_i^j + v0_{i+1}^j}\boldsymbol{v2_i} \\ \boldsymbol{v0_{i+1}^*} = \dfrac{\Delta v_i^j}{v2_i^j + v0_{i+1}^j}\boldsymbol{v0_{i+1}} \end{cases} \tag{7}$$

The physical significance of the jerk phase can be better understood with reference to Figure 7 by assuming that the deposition tool is moving from point *A* to point *B* then onto *C'* or *C"* while maintaining a constant feedrate. In the case of point *C'*, the variation in the direction of the velocity vector is small. Therefore, instead of reducing the speed in point *B* to zero, only a limited deceleration is applied up until a certain speed value, the entity of which depends on the chosen jerk module (Eq. 6). Otherwise, when the deposition tool has to follow the path *ABC"*, the vectorial variation of the velocity is significant. Consequently, both the end speed of path *i-th* $\boldsymbol{v2_i}$ and the start speed of the path (*i*+1)*-th* $\boldsymbol{v0_{i+1}}$ are down-scaled (Eq. 7). Here, it is interesting to note that, in both cases, the speed at point *B* is never equal to zero, which affects the manufacturing process in terms of vibration, quality of deposition, and accuracy, which, by limiting the jerk parameter, can be maintained at acceptable values. Furthermore, this compromise allows for obtaining a drastic reduction of the build time.

Once the jerk phase has defined the start speed $\boldsymbol{v0_i}$ and the end speed $\boldsymbol{v2_i}$ of the *i-th* step, the acceleration ramps can be computed. This type of machine generally makes use of linear acceleration ramps, which are the simplest to handle when a digital electronic device is utilised. Assuming this to be at step *i-th*, four different types of situation may occur due to the GCode's reading. The first, which is shown in Figure 8, represents the condition for which the nominal speed $v1_i^j$ is reached along the *j-th* degree of freedom. The following quantities are thus defined as in equations 8 and 9:

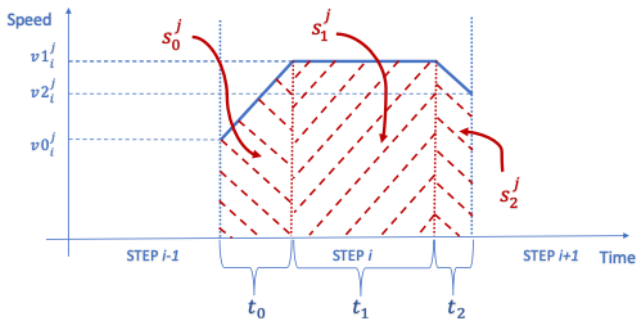

Figure 8. Nominal speed reached during step i-th.

$$\begin{cases} t_{0i} = \dfrac{|v1_i^j| - |v0_i^j|}{a_i^j} \\ s_{0i}^j = |v0_i^j|\, t_{0i} + \dfrac{1}{2}a_i^j\, t_{0i}^2 \end{cases} \tag{8}$$

$$\begin{cases} t_{2i} = -\dfrac{|v2_i^j| - |v0_i^j|}{a_i^j} \\ s_{2i}^j = |v1_i^j|\, t_{2i} - \dfrac{1}{2}a_i^j\, t_{2i}^2 \end{cases} \tag{9}$$

If the following condition (10) becomes true, the behaviour illustrated in Figure 8 is performed during the *i-th* step:

$$s_{0i}^j + s_{2i}^j > s_i^j \tag{10}$$

In this case, t_{1i} and s_1^j can be computed as follows:

$$\begin{cases} t_{1i} = \dfrac{s_1^j}{v1_i^j} \\ s_1^j = s_i^j - s_{0i}^j - s_{2i}^j \end{cases} \tag{11}$$

where s_i^j is total distance during the *i-th* step along the *j-th* degree of freedom, s_{0i}^j is the acceleration distance during the *i-th* step along the *j-th* degree of freedom, s_{1i}^j is cruise distance during the *i-th* step along the *j-th* degree of freedom, and s_{0i}^j is the deceleration distance during the *i-th* step along the *j-th* degree of freedom.

Other cases can occur when the nominal speed $\boldsymbol{v1_i^j}$ along the *j-th* degree of freedom cannot be reached due to an insufficient path length $\boldsymbol{s_i^j}$. In such a case, we can distinguish three different

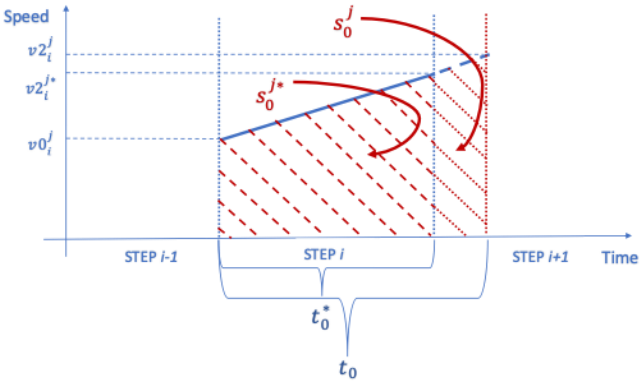

Figure 9. Insufficient distance for acceleration; new end speed.

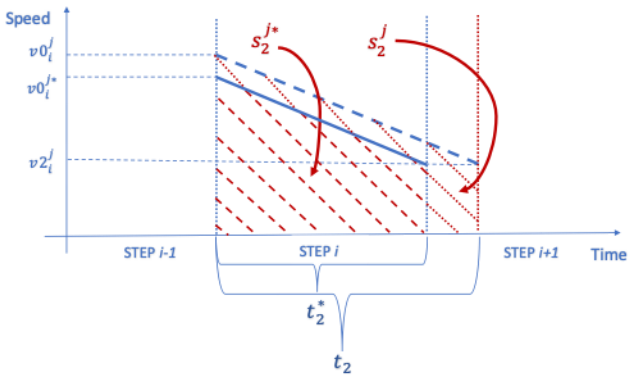

Figure 10. Insufficient space for deceleration; new start speed.

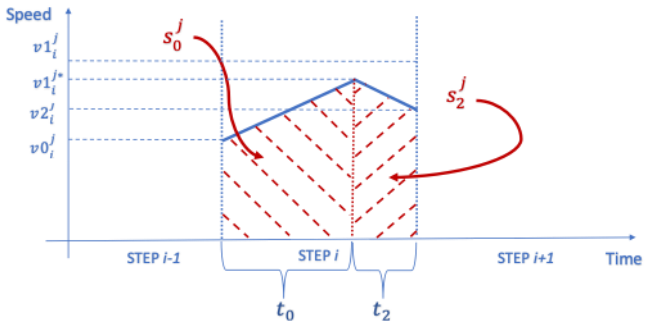

Figure 11. Nominal speed not reached. Sufficient space for accelerating and decelerating.

possibilities, the first of which is illustrated in Figure 9 and occurs when condition (12) becomes true.

$$\begin{cases} s_{0i}^j > s_i^j \\ v0_i^j < v2_i^j \end{cases} \tag{12}$$

$$\begin{cases} t_{0i} = \dfrac{|v2_i^j| - |v0_i^j|}{a_i^j} \\ s_{0i}^j = |v0_{j_i}| t_{0i} + \dfrac{1}{2} a_i^j t_{0i}^2 \end{cases} \tag{13}$$

In this case, a new end speed $v2_i^j$ is necessary and, consequently, condition (4) must be re-evaluated between step *i-th* and step *(i+1)-th*. This specific behaviour requires the use of a 'closed-loop' control. For each *i-th* step, updated evaluations can also be performed for the *(i-1)-th* and *(i+1)-th* steps, if required. For this reason, the step elaborated by the 'look-ahead' algorithm is a little further forward in relation to the step that the machine is realising. The new end speed $v2_i^{j*}$ and acceleration time t_{0i}^* are evaluated in the following way:

$$\begin{cases} v2_i^{j*} = \sqrt{2\, a_i^j s_i^j + |v0_i^j|} \\ \boldsymbol{v2_i^*} = \dfrac{v2_i^{j*}}{v1_i^j} \boldsymbol{v2_i} \\ t_{0i}^* = \dfrac{|v2_i^{j*}| - |v0_i^j|}{a_i^j} \end{cases} \tag{14}$$

The opposite situation is illustrated in Figure 10. In this case, there is insufficient space for decelerating and, consequently, the start speed $v0_i^j$ needs to be updated. This occurs when condition (15) is verified:

$$\begin{cases} s_{2i}^j > s_i^j \\ v0_i^j > v2_i^j \end{cases} \tag{15}$$

$$\begin{cases} t_{2i} = \dfrac{|v2_i^j| - |v0_i^j|}{-a_i^j} \\ s_2^j = |v0_{j_i}| t_{2i} - \dfrac{1}{2} a_i^j t_{2i}^2 \end{cases} \tag{16}$$

As such, much like in the previous case, the new start speed $v0_i^{j*}$ and deceleration time t_{2i}^* can be formulated as follows:

$$\begin{cases} v0_i^{j*} = \sqrt{2\, a_i^j s_i^j + |v2_i^j|} \\ \boldsymbol{v0_i^*} = \dfrac{v0_i^{j*}}{v1_i^j} \boldsymbol{v0_i} \\ t_{2i}^* = \dfrac{|v2_i^j| - |v0_i^{j*}|}{a_i^j} \end{cases} \tag{17}$$

A last possible situation is presented in Figure 11. In this case, the nominal speed $v1_i^j$ is not reached but there is sufficient distance for both accelerating and decelerating. This behaviour is realised when condition (18) is verified:

$$s_{0i}^j + s_{2i}^j > s_i^j \tag{18}$$

$$\begin{cases} t_{0i} = \dfrac{|v1_i^j| - |v0_i^j|}{a_i^j} \\ s_{0i}^j = |v0_i^j|\, t_{0i} + \dfrac{1}{2} a_i^j\, t_{0i}^2 \end{cases} \tag{19}$$

$$\begin{cases} t_{2i} = -\dfrac{|v2_i^j| - |v0_i^j|}{a_i^j} \\ s_{2i}^j = |v1_i^j|\, t_{2i} - \dfrac{1}{2} a_i^j\, t_{2i}^2 \end{cases} \tag{20}$$

In this case, a new cruise speed $v1_i^{j*}$ is defined:

$$v1_i^{j*} = \sqrt{\frac{{v0_i^j}^2 + {v2_i^j}^2}{2\, a_i^j} + a_i^j s_i^j} \tag{21}$$

After calculating $v1_i^{j*}$ as described in equation (21), the terms t_{0i} and t_{2i} can finally be evaluated.

$$\begin{cases} t_{0i} = \dfrac{|v1_i^{j*}| - |v0_i^j|}{a_i^j} \\ s_{0i}^j = |v0_i^j|\, t_{0i} + \dfrac{1}{2} a_i^j\, t_{0i}^2 \end{cases} \tag{22}$$

$$\begin{cases} t_{2i} = -\dfrac{|v2_i^j| - |v1_i^{j*}|}{a_i^j} \\ s_{2i}^j = |v2_i^j|\, t_{2i} + \dfrac{1}{2} a_i^j\, t_{2i}^2 \end{cases} \tag{23}$$

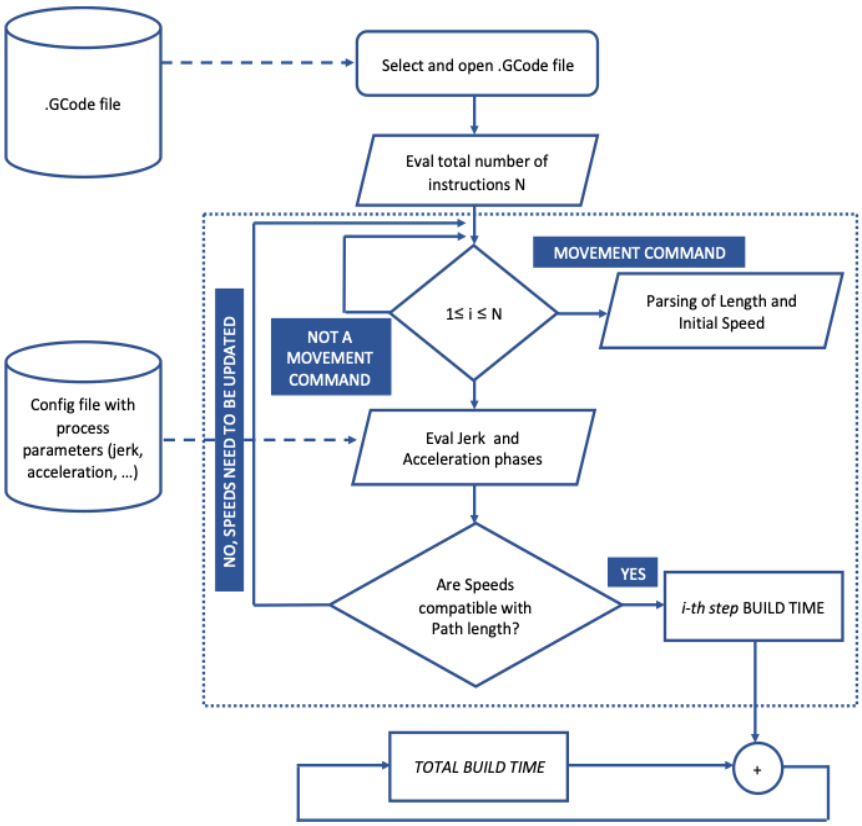

Figure 12. Algorithm flow-chart.

The previous relations and considerations were implemented into a Python script, as illustrated in the flowchart presented in Figure 12. At the initial stage, the software evaluates the total number of instructions contained in the part program. Then, using a regular expression matching, the movement commands are identified and, for each, a corresponding speed and length is extracted and memorised. The next phases consist of applying the jerk and acceleration phases for each movement command in accordance with the relationships previously illustrated. At the end of the cycle, the total build time estimated by the algorithm is provided.

3. RESULTS

In order to validate and refine the model, various comparisons were made between the theoretical build time and the actual time required by an FDM machine (German RepRap X350). Nine objects characterised by different topological and geometrical features were chosen for the experiment, as shown in Figure 13. Every part was physically realised to obtain the real value of the build time. Different CAE software (Cura, MatterControl and Simplify3D) were used for generating the part program, with the estimations compared both with the actual build time and the estimation calculated by the proposed algorithm. Since each CAE software program could implement a different slicing strategy, the part program of the same part obtained using different automatic procedures cannot be compared, which is also the case if the same process parameters are used. Therefore, for each test case, only one of the three CAE software programs was used. Table 2 lists the parameters used for the experiment.

Table 2. Process-related parameters used for the experiment.

Setting	Value
Acceleration	300 mm/min² (slow) 1000 mm/min² (rapid)
Jerk (Δv)	18 mm/min
Layer thickness	0,1 mm
Maximum speed	100 mm/s (X-Y) 10 mm/s (Z) 100 mm/s (E)
Retract length	2 mm

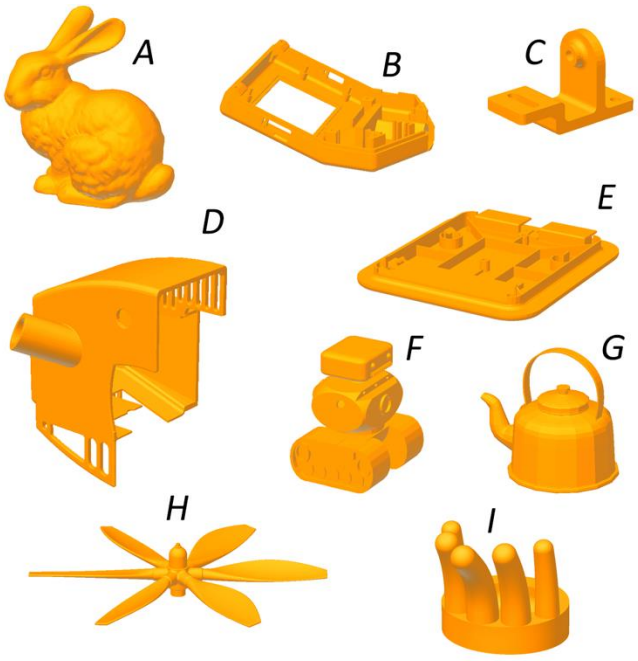

Figure 13. Models used for analysing the accuracy of the proposed algorithm.

Table 3. Experimental results.

Part	Volume, in cm³	Real build time, in min	Software	CAE build time, in min	Error	Proposed method, in min	Error
A	210	1532	Simplify 3D	1344	−14 %	1535	0.2 %
B	23	374	Cura	242	−54 %	373	−0.1 %
C	14	156	Simplify 3D	146	−7 %	156	0 %
D	127	1206	Simplify 3D	976	−24 %	1205	−0.01 %
E	27	402	MatterControl	388	−4 %	402	0 %
F	21	326	MatterControl	286	−14 %	326	0 %
G	245	2177	MatterControl	2040	−7 %	2173	−0.2 %
H	7	101	Cura	64	−58 %	101	0 %
I	25	257	Cura	170	−51 %	257	0 %

The results of the tests are presented in Table 3. Here, it is clear that for each of the CAE software programs, the error values cannot be ignored. Specifically, for the Cura platform, the average error for the three analysed test cases amounted to 54 % of the actual build time. A better situation was attained using Simplify 3D, which returned an average error of 15 % in relation to the actual build time. However, such as error cannot be accepted in most of the applications. Finally, it was confirmed that the optimal behaviour was obtained using MatterControl. The average error for this software was 8 % of the actual build time. A similar value could be accepted in certain contexts, including the budgeting process. This notwithstanding, two aspects must be noted. First, the error for the test case *F* was equal to 14 % of the actual build time, which indicates that the error was not characterised by a well-known trend, meaning the level of confidence of the estimations is low. Moreover, a parametric method such as that adopted in [34] can provide a better estimation of the actual build time while requiring significantly less time for the set-up.

Meanwhile, the valuation obtained using the proposed algorithm was highly satisfactory, with the error negligible (≤1 %) in all cases. In fact, for parts *B, E* and *F*, the estimation was equal to the actual build time. These results confirm that the proposed method can accurately evaluate the real behaviour of RepRap machines. Moreover, all the factors that systematically affect the build time were identified and taken into account.

4. CONCLUSIONS

In this paper, a new method for performing accurate estimations of build times was proposed. The resulting accuracy was made possible by an advanced analysis of the part program of the object to be manufactured, which was integrated with the information related to the control strategies.

The kinematic behaviour of a CNC machine is partially defined by the part program since the tool movements are also determined by the control strategies implemented in the machine. This is the case with the so-called 'look-ahead' algorithm, which is implemented to control the machine movements with the aim of reducing the build time while maintaining the geometrical and dimensional quality of the final object.

It was ascertained that the professional CAE tools developed for AM applications are not able to accurately evaluate the build time required for fabricating objects realised using additive technologies. Here, the results appear largely negative since they were obtained from the software that performs the manufacturing process from the given geometric data (.stl file).

Therefore, with the aim of testing the procedure for determining the build time, the case study of RepRap was taken as reference, since it presents a highly diffused controller for AM machines. Through detailed analysis of the RepRap machines, the control strategies were identified and reproduced inside the proposed method. Specifically, the method provides a mathematical formulation of the 'look-ahead' control strategy ('jerk phase').

In order to compute the build time, a custom Python application was developed. As outlined in the results section, the method provided a highly accurate estimation of the build time. For each of the nine test cases analysed, the error was less than 0.2 % of the actual build time, while in some cases, the estimated time was equal to the actual build time.

One limitation of this method is that it requires the part program, which is only obtained following a more-or-less significant number of steps. The approach is thus fairly time-consuming and would not be the optimal solution when a rapid evaluation is required or when some information is missing, such as specific parameters of the machinery in use (e.g. acceleration and jerk). In these cases, the optimal solution remains the parametric methods, for which the proposed method can present a powerful instrument for the set-up.

Further work is required to test the proposed methodology for other typologies of AM machines that do not implement the RepRap controller and which use other control strategies.

REFERENCES

[1] K. V. Wong, A. Hernandez, A review of additive manufacturing, ISRN Mech. (2012) pp. 21-31. DOI: https://doi.org/10.5402/2012/208760

[2] S. S. Babu, R. Goodridge, Additive manufacturing, Materials Science and Technology 31 (2015) pp. 881-883. DOI: https://doi.org/10.1179/0267083615Z.000000000929

[3] D. Brackett, I. Ashcroft, R. Hague, Topology optimization for additive manufacturing, 22nd Annual International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, SFF 2011, Austin, Texas, August 2011, pp. 679-692.

[4] F. Cucinotta, M. Raffaele, F. Salmeri, A stress-based topology optimization method by a Voronoi tessellation Additive Manufacturing oriented, Int. J. Adv. Manuf. Technol. 103 (2019) pp. 1965-1975.

[5] D. S. Thomas, S. W. Gilbert, Costs and cost effectiveness of additive manufacturing: A literature review and discussion, in: Additive Manufacturing: Costs, Cost Effectiveness and Industry Economics. F. Brewer (editor). Nova Sciences, New York, 2015. ISBN: 978-1-63483-364-6, 77 p. DOI: http://dx.doi.org/10.6028/NIST.SP.1176

[6] P. Alexander, S. Allen, D. Dutta, Part orientation and build cost determination in layered manufacturing, CAD Comput. Aided

Design 30, 5, April 1998, pp. 343-356.
DOI: https://doi.org/10.1016/S0010-4485(97)00083-3

[7] G. Costabile, M. Fera, F. Fruggiero, A. Lambiase, D. Pham, Cost models of additive manufacturing: A literature review, Int. J. Ind. Eng. Comput. 8 (2017), pp. 263–282
DOI: https://doi.org/10.5267/j.ijiec.2016.9.001

[8] S. L. Chan, Y. Lu, Y. Wang, Data-driven cost estimation for additive manufacturing in cybermanufacturing, J. Manuf. Syst. 46 (2018) 115-126
DOI: https://doi.org/10.1016/j.jmsy.2017.12.001

[9] E. Asadollahiyazdi, J. Gardan, P. Lafon, Multi-objective optimization of additive manufacturing process, IFAC-PapersOnLine 51 (2018) pp. 152-157.
DOI: https://doi.org/10.1016/j.ifacol.2018.08.250

[10] K. Thrimurthulu, P. M. Pandey, N. V. Reddy, Optimum part deposition orientation in fused deposition modeling, Int. J. Mach. Tools Manuf., 44, 6, May 2004, pp. 585-594.
DOI: https://doi.org/10.1016/j.ijmachtools.2003.12.004

[11] Y. Zhang, A. Bernard, R. K. Gupta, R. Harik, Feature based building orientation optimization for additive manufacturing, Rapid Prototyp. Journal, Vol. 22 No. 2, 2016, pp. 358-376.
DOI: https://doi.org/10.1108/RPJ-03-2014-0037

[12] Y. Zhang, A. Bernard, R. Harik, K. P. Karunakaran, Build orientation optimization for multi-part production in additive manufacturing, J. Intell. Manuf. 6/2017, pp. 1393–1407.
DOI: https://doi.org/10.1007/s10845-015-1057-1

[13] P. Das, R. Chandran, R. Samant, S. Anand, Optimum part build orientation in additive manufacturing for minimizing part errors and support structures, Procedia Manufacturing, vol. 1, 2015, pp. 343-354.
DOI: https://doi.org/10.1016/j.promfg.2015.09.041

[14] S. Chowdhury, K. Mhapsekar, S. Anand, Part build orientation optimization and neural network-based geometry compensation for additive manufacturing process, J. Manuf. Sci. Eng. Trans. ASME, March 2018, 140(3), 031009, 15 pages.
DOI: https://doi.org/10.1115/1.4038293

[15] G. Strano, L. Hao, R. M. Everson, K. E. Evans, Multi-objective optimization of selective laser sintering processes for surface quality and energy saving, in Proc. of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, vol. 22, issue 9 (2011), pp. 1673-1682.
DOI: https://doi.org/10.1177/0954405411402925

[16] V. Canellidis, J. Giannatsis, V. Dedoussis, Genetic-algorithm-based multi-objective optimization of the build orientation in stereolithography, Int. J. Adv. Manuf. Technol., 45, 2009, pp. 714-730.
DOI: https://doi.org/10.1007/s00170-009-2006-y

[17] S. K. Singhal, P. K. Jain, P. M. Pandey, A. K. Nagpal, Optimum part deposition orientation for multiple objectives in SL and SLS prototyping, Int. J. Prod. Res, Volume 47, 2009, Issue 22, pp. 6375-6396.
DOI: https://doi.org/10.1080/00207540802183661

[18] A. Li, Z. Zhang, D. Wang, J. Yang, Optimization method to fabrication orientation of parts in fused deposition modeling rapid prototyping, in 2010 International Conference on Mechanic Automation and Control Engineering, MACE2010, Wuhan, China, 26-28 June 2010, pp. 36-42.

[19] P. Jaiswal, J. Patel, R. Rai, Build orientation optimization for additive manufacturing of functionally graded material objects, Int. J. Adv. Manuf. Technol. 96, 2018, pp. 223-235.
DOI: https://doi.org/10.1007/s00170-018-1586-9

[20] N. Padhye, K. Deb, Multi-objective optimisation and multi-criteria decision making in SLS using evolutionary approaches, Rapid Prototyp. J. 17 (6) 2011, pp. 458-478.
DOI: https://doi.org/10.1108/13552541111184198

[21] S. Khodaygan, A. H. Golmohammadi, Multi-criteria optimization of the part build orientation (PBO) through a combined meta-modeling/NSGAII/TOPSIS method for additive manufacturing processes, Int. J. Interact. Des. Manuf. 12 (2) 2018, 15 pages.
DOI: https://doi.org/10.1007/s12008-017-0443-7

[22] A. M. Phatak, S. S. Pande, Optimum part orientation in rapid prototyping using genetic algorithm, Transactions of the North American Manufacturing Research Institution of SME 31 (2012), pp. 395-402.

[23] S. E. Brika, Y. F. Zhao, M. Brochu, J. Mezzetta, Multi-objective build orientation optimization for powder bed fusion by laser, J. Manuf. Sci. Eng. Trans. ASME, Volume 6, Issue 4, 2016, 1000236, 9 pages.
DOI: https://doi.org/10.4172/2169-0316.1000236

[24] L. Di Angelo, P. Di Stefano, Parametric cost analysis for web-based e-commerce of layer manufactured objects, Int. J. Prod. Res., Volume 48, Issue 7, 2010, pp. 2127-2140.
DOI: https://doi.org/10.1080/00207540802183653

[25] P. T. Lan, S. Y. Chou, L. L. Chen, D. Gemmill, Determining fabrication orientations for rapid prototyping with stereolithography apparatus, CAD Computer Aided Design, Volume 29, Issue 1, 1997, pp. 53-62.
DOI: https://doi.org/10.1016/S0010-4485(96)00049-8

[26] H. S. Byun, K. H. Lee, Determination of the optimal build direction for different rapid prototyping processes using multi-criterion decision making, Robotics and Computer-Integrated Manuf., Volume 22, Issue 1, February 2006, pp. 69-80.
DOI: https://doi.org/10.1016/j.rcim.2005.03.001

[27] D. T. Pham, S. S. Dimov, R. S. Gault, Part orientation in stereo-lithography, Int J Adv Manuf Technol (1999) 15, pp. 674–682.
DOI: https://doi.org/10.1007/s001700050118

[28] I. Campbell, J. Combrinck, D. De Beer, L. Barnard, Stereo-lithography build time estimation based on volumetric calculations, Rapid Prototyp. J. 14(5), September 2008, pp. 271-279.
DOI: https://doi.org/10.1108/13552540810907938

[29] C. A. Ernesto, R. T. Farouki, High-speed cornering by CNC machines under prescribed bounds on axis accelerations and toolpath contour error, Int. J. Adv. Manuf. Technol., 58 (2012), pp. 327–338.
DOI: https://doi.org/10.1007/S00170-011-3394-3

[30] Y. A. Jin, Y. He, J. Z. Fu, Z. W. Lin, W. F. Gan, A fine-interpolation-based parametric interpolation method with a novel real-time look-ahead algorithm, CAD Computer Aided Design, Volume 55, October 2014, pp. 37-48.
DOI: https://doi.org/10.1016/j.cad.2014.05.002

[31] S. Sun, H. Lin, L. Zheng, J. Yu, Y. Hu, A real-time and look-ahead interpolation methodology with dynamic B-spline transition scheme for CNC machining of short line segments, Int. J. Adv. Manuf. Technol. 84 (2016), pp. 1359–1370.
DOI: https://doi.org/10.1007/s00170-015-7776-9

[32] Y. Zhang, M. Zhao, P. Ye, J. Jiang, H. Zhang, Optimal curvature-smooth transition and efficient feedrate optimization method with axis kinematic limitations for linear toolpath, Int. J. Adv. Manuf. Technol. 99 (2018), pp. 169-179.
DOI: https://doi.org/10.1007/s00170-018-2496-6

[33] L. Wang, J. Cao, A look-ahead and adaptive speed control algorithm for high-speed CNC equipment, Int. J. Adv. Manuf. Technol. 63 (2012), pp. 705-712.
DOI: https://doi.org/10.1007/s00170-012-3924-7

[34] L. Di Angelo, P. Di Stefano, E. Guardiani, A build-time estimator for additive manufactured objects, in: Design Tools and Methods in Industrial Engineering. C. Rizzi, F. Leali (editors). Springer International Publishing, Berlin, 2020, ISBN: 978-3-030-31153-7, pp. 925-935.