OpenCN: une commande numérique ouverte pour des applications à haute performance

Raoul HERZOG, Alain SCHORDERET, Daniel ROSSIER

Haute école spécialisée de Suisse occidentale, HEIG-VD, CH-1401 Yverdon-les-Bains

Mail : raoul.herzog@heig-vd.ch

Résumé : Les commandes numériques (CNC) du marché sont des boîtes noires où l'utilisateur n'a quasiment aucun accès aux algorithmes de planification de trajectoires. LinuxCNC est une commande libre et ouverte, mais pas adaptée pour l'usinage à haute performance, car sans contrôle du jerk. OpenCN surmonte ces limitations avec une optimisation de trajectoires en deux étapes :

1. optimisations géométriques comprenant entre autres un lissage optimal des transitions avec une continuité géométrique de type G^2.
2. optimisation temporelle avec contrôle du jerk, basée sur un problème d'optimisation convexe de type LP appliqué sur un horizon reculant.

Les algorithmes sont codés et testés sur Matlab avec une génération automatique du code embarqué C/C++.

Le framework OpenCN est constitué d'un environnement de type Linux asymétrique (AMP), avec un noyau récent, et d'une version revisitée de l'extension temps-réel Xenomai/Cobalt. Les plateformes HW supportées sont x86, Raspberry Pi, Zynq® Ultrascale+, et QEMU/virt64 offrant ainsi la possibilité d'exécuter le framework dans un environnement émulé. Un maître EtherCAT libre est inclus, permettant la transmission fiable de consignes aux drives avec une cadence de 10 kHz grâce à l'horloge distribuée (DC).

Une validation fonctionnelle de OpenCN est présentée sur une mini fraiseuse 3 axes très performante, dont la première fréquence propre est supérieure à 180 Hz. Les tests ont été effectués sur des pièces fraisées en laiton jusqu'à des vitesses d'avance de F9000. Un module cinématique pour des machines 5 axes est en cours de développement.

Mots clés : CNC, Optimisation, Trajectoires, Linux, Temps réel, EtherCAT.

1 Introduction

Les commandes numériques du marché sont des logiciels complexes et fermés, ne permettant pas à l'utilisateur d'intervenir au niveau algorithmique pour la génération de trajectoires. L'expérience a montré qu'il y a des applications, où même des commandes numériques haut de gamme donnent des faibles performances comparées à une génération de trajectoire optimale (Schorderet 2019). Ce constat motivait le développement d'une commande numérique ouverte appelée "OpenCN".

Alors que les logiciels *ouverts* dominent aujourd'hui par exemple le marché des serveurs et le domaine du super computing, les logiciels ouverts restent pour l'instant peu présents dans le marché des machines, à l'exception de LinuxCNC et peut-être de la commande semi-ouverte ctrlX (Bosch Rexroth, 2022). LinuxCNC a ses origines dans un long historique de développements autour du projet EMC dans les années 90 (Staroveški, 2009). Malheureusement LinuxCNC n'est pas à jour concernant son architecture logicielle temps réel, et ne permet pas une génération de trajectoires avec limitation du jerk. La limitation du jerk est un moyen important pour diminuer les vibrations de machine qui dégradent l'état de surface de la pièce usinée.

Cette publication présente la commande numérique ouverte "OpenCN". Basée d'origine sur LinuxCNC, OpenCN incorpore une génération de trajectoire innovante et une transmission de consignes via un bus de terrain temps réel ouvert. Le choix du bus de terrain s'est porté sur EtherCAT, plus répandu que SERCOS III, et offrant une multitude de différents fournisseurs de drives et modules périphériques avec bus EtherCAT. OpenCN implémente un maître EtherCAT avec un débit maximal de 10 kHz sans jigue grâce à l'horloge distribuée. Ceci permet d'utiliser OpenCN pour des applications où une haute dynamique et une précision élevée sont impératives. L'objectif de OpenCN est de fournir un framework ouvert et performant, et ceci dans un premier temps pour des besoins de recherche (différentes topologies cinématiques, nouveaux algorithmes de génération de trajectoire, maîtrise du comportement vibratoire de la machine d'usinage, monitoring temps réel, Industrie 4.0, etc). Ultérieurement, la commande OpenCN pourrait être appliquée à des cas industriels très spécifiques. OpenCN tourne sur les plateformes matérielles x86 et ARM, sans besoin de hardware spécifique.

Cette publication est organisée de la manière suivante: le chapitre 2 explique la planification de trajectoires; le chapitre 3 montre le framework logiciel temps réel Linux/Xenomai AMP; le chapitre 4 explique le hardware de la mini machine de fraisage 3 axes, et le chapitre 5 donne une validation expérimentale. Les conclusions sont présentées dans le chapitre 6.

2 Planification optimale de trajectoires

L'objectif de la planification optimale de trajectoires consiste à générer les consignes temporelles optimale pour les axes de la machine sous contrainte d'avance, accélération et jerk maximaux, et sous contrainte de tolérances géométriques par rapport à la trajectoire programmée par le code G (ISO). La littérature sur la planification de

trajectoire est abondante, et elle peut grossièrement être divisée en deux groupes : optimisation *simultanée* de la géométrie et de l'avance, et optimisation *séquentielle* de la géométrie, suivie par une optimisation temporelle de la vitesse d'avance (feedrate). Une optimisation simultanée a été proposée par (Mercy, 2019) et (Haas, 2019). Le problème d'optimisation résultant est non-convexe, ce qui peut potentiellement aboutir à un manque de fiabilité numérique.

Pour cette raison, OpenCN a donné la préférence à une optimisation géométrique et temporelle *séquentielle*, en exploitant au maximum des solutions *analytiques*, des problèmes d'optimisation *convexes* et des algorithmes numériques éprouvés.

2.1 Lissage géométrique à continuité G^2, compressage et partitionnement

L'interprétation du code G (RS274) produit une liste de morceaux de courbes paramétriques décrite par $\boldsymbol{r}_k(u_k),\ k = 1, \dots, N$, où chaque abscisse paramétrique peut être normalisée $0 \leq u_k \leq 1$. Pour éviter les sauts d'accélération, les transitions entre les différents morceaux de courbes doivent être lissées afin d'atteindre une continuité géométrique G^2. Une interpolation G^2 optimale de type Hermite a été proposée par (Herzog, 2019), basée sur la minimisation de la fonction coût $\int_0^1 \|\boldsymbol{r}'''(u)\|^2 du$ sous contrainte de continuité G^2 aux points de décrochage. Ce critère d'optimalité permet d'atteindre une courbe de transition naturelle. La courbe de transition résultante est donnée par un polynôme de degré 5. La détermination de ses coefficients remonte à trouver les racines positives d'un polynôme de degré ≤ 9. Cette approche de OpenCN peut être calculée de manière très rapide et fiable (12 µs par transition sur un core Intel i7) avec une analyse des tolérances géométriques. L'approche (Herzog, 2019) pour le calcul des transitions est généralisable pour des morceaux de courbes en R^n, permettant de gérer également des machines d'usinage 5 axes.

Les opérations géométriques de OpenCN incluent également la détection de points de rebroussement de la courbe programmée. Un point de rebroussement engendre forcément un arrêt de machine. Avant de poursuivre avec l'optimisation temporelle, une homogénéisation des longueurs des différents morceaux de courbe est nécessaire. Les courbes trop longues sont partitionnées en une série de courbes plus petites. Une séquence de petits morceaux G01 est comprimée dans une seule longue B-spline à l'aide de l'algorithme de Lee (Lee, 1989).

2.2 Planification de la vitesse d'avance avec horizon reculant

Les dérivées temporelles $\frac{d^n}{dt^n}\,\boldsymbol{r}(u(t))$ peuvent être calculées en utilisant la règle de la chaîne et de la multiplication

$$\dot{\boldsymbol{r}} = \boldsymbol{r}'\,\dot{u} \tag{1}$$

$$\ddot{\boldsymbol{r}} = \boldsymbol{r}''\,\dot{u}^2 + \boldsymbol{r}'\,\ddot{u} \tag{2}$$

$$\dddot{\boldsymbol{r}} = \boldsymbol{r}'''\,\dot{u}^3 + 3\,\boldsymbol{r}''\,\dot{u}\,\ddot{u} + \boldsymbol{r}'\,\dddot{u} \tag{3}$$

où $\dot{} = \frac{d}{dt}$ et $' = \frac{\partial}{\partial u}$. Il fut observé dans (Verscheure, 2009) qu'avec un changement nonlinéaire de variables $q(u) = \dot{u}^2$, l'accélération devient linéaire dans la nouvelle variable $q(u)$, et le problème d'optimisation temporelle devient *convexe* en laissant de côté les contraintes de jerk. Il s'exprime alors sous la forme suivante :

$$\text{minimiser} \int_0^1 \frac{1}{\sqrt{q(u)}}\,\text{du} \tag{4}$$

sous contraintes :

$$q(u) \leq v_{max}^2 / \|\boldsymbol{r}'(\text{u})\|^2 \tag{5}$$

$$-\boldsymbol{a}_{max} \leq \mathbf{r}''(\text{u})\,\text{q(u)} + \frac{1}{2}\boldsymbol{r}'(u)\,q'(u) \leq \boldsymbol{a}_{max} \tag{6}$$

où l'inégalité (Eq. 6) est exprimée composante par composante.

La contrainte de jerk est par contre intrinsèquement non-convexe, et plusieurs approches ont été publiées pour s'affranchir de ce problème. (Consolini, 2021) a proposé un algorithme d'optimisation basé sur un "line search" séquentiel. (Erkorkmaz, 2017) a montré que le problème d'optimisation temporelle sans les contraintes de jerk peut être reformulé par un problème de programmation linéaire (LP), en observant que minimiser le temps de parcours est équivalent à maximiser $\int_0^1 q(u)\,du$. A cet effet, $q(u)$ est discrétisé par une B-Spline avec un degré et une répartition de nœuds donnés. Les variables de décision correspondent aux coefficients x de la B-Spline. La contrainte de jerk prend ainsi la forme suivante

$$\left|\boldsymbol{r}'''(u)\,q(u) + \frac{3}{2}\,\boldsymbol{r}''(u)\,q'(u) + \frac{1}{2}\boldsymbol{r}'(u)\,q''(u)\right| \sqrt{q(u)} \leq \boldsymbol{j}_{max} \tag{7}$$

(Erkorkmaz, 2017) a proposé de remplacer le terme non-convexe $\sqrt{q(u)}$ dans (Eq. 7) par une borne supérieure précalculée $\sqrt{q^*(u)}$, où $q^*(u)$ est la solution de l'optimisation temporelle sans contraintes de jerk. Cette approche introduit un conservatisme qui se manifeste surtout par des démarrages et des arrêts beaucoup trop lents. Pour remédier à ce problème, les phases de démarrage et d'arrêt sont gérées séparément, voir chapitre 2.3. L'approximation de (Erkorkmaz, 2017) permet de garder un problème d'optimisation convexe de type LP standard

$$maximiser\ \boldsymbol{c}^T\mathbf{x} \quad \text{sous contraintes} \begin{cases} \boldsymbol{A}\,\boldsymbol{x} \leq \boldsymbol{b} \\ \boldsymbol{A_{eq}}\,\boldsymbol{x} = \boldsymbol{b_{eq}} \end{cases}. \tag{8}$$

Le contenu des matrices dans (Eq. 8) est facile à déterminer en évaluant les fonctions de base de la paramétrisation B-Spline de q(u) sur la grille de discrétisation de u. A noter que la matrice $\boldsymbol{A}$ a une structure creuse. OpenCN utilise le solveur Simplex CLP (COIN-OR) pour résoudre les problèmes LP. Une approche à horizon déroulant est utilisée permettant de trouver l'avance optimale au fur et à mesure. Il s'est avéré qu'un horizon déroulant de 5 morceaux de courbe est suffisant sans perte significative en optimalité, sous condition que la longueur d'arc des morceaux dans l'horizon ne soit pas trop faible.

2.3 Échantillonnage temporel de la trajectoire et gestion de vitesse nulle

La période d'échantillonnage est définie par Δt. Les valeurs échantillonnées de u sont définies par $u_k = u(t_k)$ et $u_{k+1} = u(t_k + \Delta t)$.

La série de Taylor de $u(t)$ coupée après le terme de degré 2 donne l'approximation suivante

$$u_{k+1} = u_k + \sqrt{q(u_k)}\,\Delta t + \frac{1}{4} q'(u_k)\,(\Delta t)^2. \quad (9)$$

Cette manière de faire permet de générer les consignes de position $\boldsymbol{r}(u_k)$ pour les drives des différents axes de la machine. Si $u_{k+1} > 1$, une transition vers le prochain morceau de courbe doit être effectuée. A cet effet, on calcule d'abord le temps écoulé T_r depuis le point précédent u_k jusqu'à $u = 1$. En approximant $q(u)$ par une fonction linéaire par morceau, l'intégration analytique de $T_r = \int_{u_k}^{1} \frac{1}{\sqrt{q(u)}} \mathrm{d}u$ est donnée par

$$T_r = \frac{2\,(1 - u_k)}{\sqrt{q(1)} + \sqrt{q(u_k)}} \quad (10)$$

Le prochain échantillon temporel du prochain morceau de courbe doit commencer à un instant raccourci $\Delta t_0 = \Delta t - T_r$ en utilisant (Eq. 9).
La condition zéro vitesse au démarrage et à l'arrêt correspondent à une singularité et est traitée différemment. Pour cela, un pseudo jerk constant j_{ps} est appliqué pendant une courte durée au démarrage et à l'arrêt

$$u_k = \frac{1}{6} j_{ps}\,(\mathrm{k}\,\Delta t)^3. \quad (11)$$

Finalement, la continuité de la vitesse d'avance et de l'accélération tangentielle aux bords de morceaux adjacents de trajectoire peut être formulée par des contraintes d'égalité linéaires à incorporer dans le problème d'optimisation LP.

3 Framework logiciel temps réel OpenCN

Dans le logiciel libre dédié aux commandes numériques, LinuxCNC apparaît comme une solution logicielle bien connue qui peut s'adapter à une large gamme d'outils et de bus de terrain, avec diverses options de fabrication.
LinuxCNC résulte d'une longue histoire de divers développements autour du projet EMC au début des années 90, destiné à être une implémentation de référence neutre (sans dépendance du fournisseur) du langage standard de l'industrie RS-274D (code G) (T. Staroveški, 2009) pour le contrôle numérique des opérations d'usinage.
Le projet EMC2 a été définitivement renommé LinuxCNC en 2013 et peut fonctionner sur diverses plates-formes x86 et ARM. Il prend en charge plusieurs extensions temps réel dans Linux telles que RTAI, Xenomai et RT-Preempt qui peuvent garantir un haut degré de déterminisme requis par le bus de terrain EtherCAT par exemple.
Au fil des années, l'architecture logicielle de LinuxCNC est devenue très complexe ; la mise à niveau vers un noyau récent, le rajout de nouveaux algorithmes d'optimisation, la customisation d'une distribution afin de la maintenir aussi légère et personnalisable que possible, et l'objectif d'avoir un démarrage de l'environnement complet en moins de 30 secondes était difficile à réaliser. Pour ces raisons, le développement du framework OpenCN a conduit à utiliser un noyau Linux récent (5.10) disposant d'un support à long terme (LTS) dans lequel l'approche par composant de LinuxCNC et sa gestion par *pin* pour le pilotage des périphériques ont été revues et intégrées dans le framework.

De plus, les composants de base comme *streamer*, *sampler*, *lcec*, etc. ont été également adaptés.
L'approche AMP (*Asymmetric Multi-Processing*) de OpenCN signifie une répartition sélective et contrôlée des activités sur les différents cœurs CPU disponibles dans le processeur/microcontrôleur. A cet effet, OpenCN ne peut tourner que sur une architecture disposant de quatre cœurs au minimum (CPU quadcore).
Aujourd'hui, OpenCN peut tourner sur les plates-formes suivantes : PC/x86[1], Raspberry Pi 4 (ARM 64-bits), Zynq/Ultra-scale+ (ZCU106) et QEMU/virt64 permettant d'exécuter le framework dans un environnement émulé et permettant ainsi de facilité la mise au point et la validation de certaines fonctionnalités.

La figure 1 ci-dessous montre l'architecture système du framework OpenCN utilisant l'approche AMP.

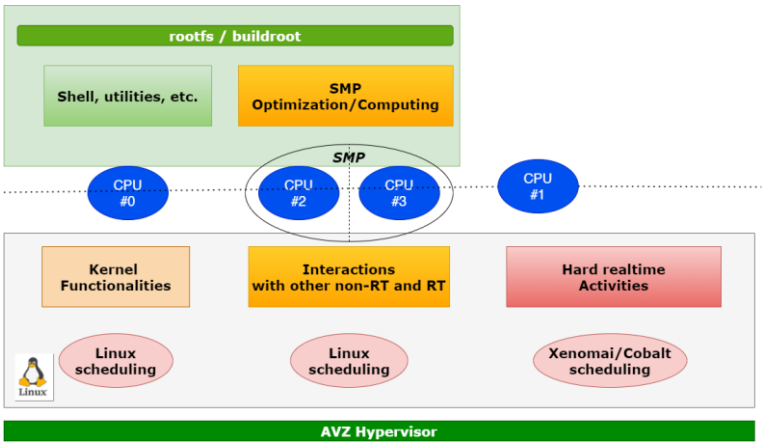

Figure 1: Architecture générale du framework OpenCN/AMP

Le premier cœur (CPU #0) est normalement utilisé pour le démarrage du noyau et des tâches initiales. Il est réservé pour l'exécution des applications standards telles qu'une application graphique (GUI). Ce *domaine* d'exécution ne nécessite pas du temps réel dur. Les cœurs CPU #2 et CPU #3 sont réservés à l'exécution des algorithmes d'optimisation *multithreadés* ; dans ce contexte, les deux cœurs sont symétriques (SMP) et peuvent exécuter indifféremment les différents *threads*. Là aussi, ce domaine composé de ces deux CPUs ne s'exécute pas avec du temps réel dur : la planification de la vitesse d'avance (*feedrate*), par exemple, n'est pas soumise à des contraintes strictes en temps réel.

L'extension temps réel dur (RT) présente dans le noyau Linux du framework OpenCN est basée sur le conoyau Cobalt issue de Xenomai (Gerum, 2016). Celui-ci fournit des services RT au niveau le plus bas (noyau). Par choix architectural, OpenCN exécute toutes les tâches nécessitant du temps réel dur dans l'espace noyau sur CPU #1, le cœur CPU dédié à ce type d'activités (domaine RT). Afin d'isoler les activités du domaine RT des

[1] Sur x86, la fonction *Hyperthreading* doit être désactivée afin de simplifier les interactions au niveau CPU et de limiter le risque de latences au niveau des caches.

autres domaines non temps réel, le noyau Linux a été *patché* afin d'assurer l'isolement des activités, en attachant toutes les activités RT au CPU #1 et en empêchant l'ordonnanceur de Linux d'intervenir lors de l'exécution des threads RT. C'est pourquoi, l'environnement du conoyau Cobalt a subi également des retouches afin de supprimer l'abstraction I-pipe (Adeos) gérant le traitement des IRQs entre Cobalt et Linux, devenu inutile avec l'approche AMP. La suppression de la couche I-Pipe réduit également la surcharge de traitement lors du traitement des IRQs. Les interactions entre le domaine RT et les domaines non RT (CPU #0, CPU #2 et #3) sont prises en charge par un hyperviseur très léger (appelé AVZ) permettant l'envoi et la réception d'événements entre les processeurs (*Inter-Processor Interrupt*) ainsi qu'une gestion de buffers performante basée sur des anneaux partagés (Warfield, 2005).

Parmi les activités de type RT, on y trouve le composant *lcec* implémentant un maître EtherCAT et le *driver* réseau permettant l'accès à l'interface Gigabit Ethernet.

La figure montre la répartition des activités du framework OpenCN sur les différents cœurs.

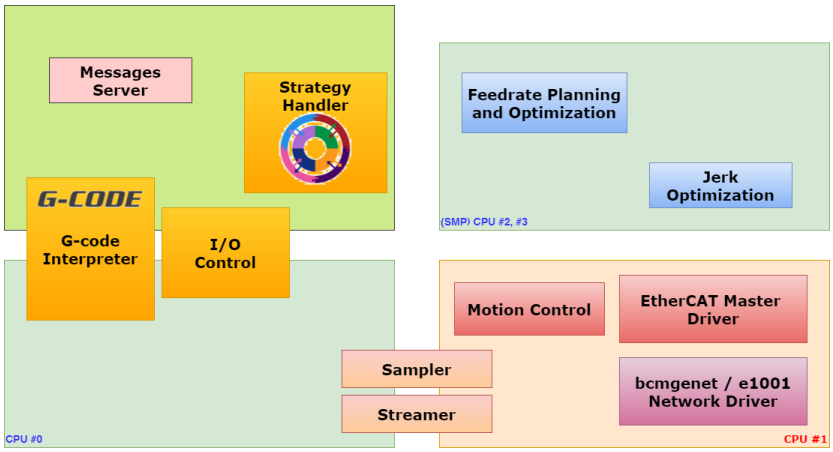

Figure 2: Répartition des activités RT et non-RT entre les cœurs CPU

Le framework OpenCN implémente le maître EtherCAT IgH largement répandu et disponible en logiciel libre. Ce composant a été adapté dans OpenCN pour le rendre exécutable dans le domaine RT (modèle RTDM de Cobalt).

D'autres adaptations ont été nécessaires afin de disposer d'une horloge distribuée conforme au mode B entre esclaves EtherCAT et d'assurer une transmission temps-réel fiable à 10 kHz.

Différents tests ont été effectués sur plusieurs heures avec OpenCN et ont montré la stabilité et la robustesse du framework ; les latences et jigues sont faibles et bornées de l'ordre de 100 ns sur x86 et 6 us sur Raspberry Pi 4.

Une application graphique permettant la configuration et le monitoring de la pièce usinée est également disponible.

Sur les différentes plates-formes supportées par OpenCN, l'architecture globale conduit à un comportement très stable et robuste sur plusieurs heures. En revanche, dans le cas PC/x86, l'utilisation de l'application graphique introduit de la jigue sur l'interface réseau utilisée par EtherCAT dans le domaine RT. Cette latence est due principalement à certaines requêtes DMA propres à la GPU. Celle-ci se trouvant sur la carte-mère, elle nécessite une priorité élevée sur le bus de données lors d'opérations en lien avec l'interface graphique bloquant le CPU #1 lors de ses propres requêtes sur le bus,

provoquant ainsi la perte occasionnelle de paquets EtherCAT. Malheureusement, il a été impossible de patcher les pilotes GPU ou DMA en absence d'informations techniques détaillées. Ce problème n'est pas apparu sur les systèmes à base de processeur ARM.
Afin de résoudre le problème, OpenCN permet de faire tourner l'application graphique sur un PC déporté relié à la plate-forme tournant OpenCN via TCP/IP.

4 Mini machine de fraisage

OpenCN a été validé expérimentalement principalement sur une mini-machine de fraisage expérimentale (Figure 2a) conçue initialement pour valider le comportement dynamique d'une mini-machine 5 axes commercialisée depuis (Chiron 2020), (Precitrame 2021). Le rapport machine-pièce est petit (5:1), la consommation totale est inférieure à 780 W. L'architecture cartésienne comprend 2 axes superposés (X et Y). Les axes Y et Z sont posés sur le bâti. Les trois axes sont actionnés par des vis à billes et fournissent un volume de travail de 50 x 50 x 30 mm typique pour les applications de haute précision horlogère et nombre d'applications médicales et dentaires. La masse des composants en mouvement est inférieure à 10 kg. La rigidité statique au porte-outil est de 5.0 $x10^6$ N/m, la fréquence du premier mode vibratoire est supérieure à 180 Hz. Les vitesses et accélérations maximales sont de 30 m/s et 20 m/s^2 respectivement. La machine est équipée d'une broche d'usinage Meyrat de 240 W permettant d'usiner jusqu'à 80'000 tpm. Les axes et la broche sont pilotés au moyen de drives EtherCAT haut de gamme Triamec TSD80E, exploitant un contrôle de type dual-loop basé sur les signaux des encodeurs linéaires des axes et de rotation des moteurs. La boucle de régulation interne est de 100 kHz pour toutes les boucles de contrôle et de commande anticipée. Les échantillons de consigne de position sont fournis avec un débit EtherCAT de 10 kHz, puis interpolés finement et sur-échantillonnés à 100 kHz.

5 Validation expérimentale

La première étape de validation consiste à vérifier que, pour des séries points de passage calculés par OpenCN, les valeurs de vitesse, accélération et jerk respectent les contraintes imposées à chaque pas de temps. Le principe du maximum de Pontryagin pour l'optimalité du temps impose que la solution doit avoir à chaque pas de temps au moins une contrainte active, c.-à-d. une contrainte saturée (comportement bang-bang). La trajectoire commence par un départ de l'arrêt avec un Jerk maximum, jusqu'à atteindre l'accélération maximale sur un axe, et finalement la vitesse d'avance maximale admissible. Cette validation expérimentale a été réalisée sur la mini-machine en utilisant les paramètres de la Table 1.

Tableau 1: Paramètres utilisés lors de la validation expérimentale.

Paramètre	**Symbole**	**Valeur**	**Unité**
Cutoff length for smoothing	L_{cut}	0.1	mm
Threshold for splitting	L_{split}	2.0	mm
Max speed	v_{max}	fig. 3b	mm/min
Max acceleration per axis	a_{max}	$20 \cdot 10^3$	mm/s^2
Max jerk per axis	j_{max}	$1.5 \cdot 10^6$	mm/s^3
EtherCAT period	Δt	0.1	ms
Receding horizon depth	N_h	3	-
B-spline degree for q(u)	d	3	-
# of knots for B-spline q(u)	N_k	10	-
# of grid points for inequalities	N_g	20	-

La seconde étape réalisée à ce jour consiste en une validation fonctionnelle de l'usinage. Un code-G généré par Autodesk HSMWorks a été exploité par OpenCN pour réaliser la pièce en laiton de la figure 3b. Un outil de diamètre 1 mm a été utilisé avec une vitesse de rotation de broche de 45'000 tpm.

Figure 3. A gauche : Mini-machine de fraisage
A droite : pièce de validation - contournage (1) : ébauche F5000, finition F1000; poches (2): F272; surfaçage (3): F2000

La dernière étape de validation concerne les performances de OpenCN en relation avec la qualité de l'usinage : précisions dimensionnelles, géométriques et état de surface de la pièce usinée. Cette étape avec des résultats quantitatifs et comparatifs est actuellement en cours, aussi les résultats ne sont pas publiés à ce stade. Le principe de cette validation expérimentale repose sur la comparaison des résultats d'usinage obtenus sur une mini-machine industrielle 5 axes de type micro5, équipée d'une commande numérique standard Beckhoff (configuration de référence) et de la commande OpenCN, opérée sur un PC x86 ou sur une Raspberry Pi4 (configurations évaluées).

6 Conclusions

OpenCN (OpenCN, 2019) est un logiciel ouvert pour la commande numérique, avec génération de trajectoires à jerk limité et une communication par bus EtherCAT. La génération des consignes pour les drives est calculée en temps réel basé sur des algorithmes numériques éprouvés comme p.ex. le calcul de racines de polynômes et l'appel de solveurs pour des problèmes d'optimisation convexe. Les algorithmes de

génération de trajectoires sont codés en Matlab puis traduits automatiquement en C++ par la boîte à outil "Matlab Coder". Ceci permet une meilleure maintenabilité et une efficacité de codage élevée. A ce stade, OpenCN n'a pas encore toutes les fonctionnalités d'une commande numérique du marché, mais c'est une solution complètement ouverte, basée sur un framework logiciel innovant. OpenCN a été validé sur une mini machine de fraisage 3 axes. Le support pour cinématiques 5 axes ainsi que des résultats comparatifs avec des CN du marché est en cours. L'utilisation et le support de OpenCN par d'autres groupes de recherche serait fortement apprécié.

Remerciements

Les auteurs remercient particulièrement la Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (HEIG-VD) pour son financement du projet OpenCN.
Ils remercient également la Haute Ecole Spécialisée de Suisse Occidentale (HES-SO) pour son financement du projet de recherche E2C permettant le développement du 5 axes.

Références

X. Beudaert, , P.-Y. Pechard, Ch. Tournier (2011) 5-Axis Tool Path Smoothing based on Drive Constraints, International Journal of Machine Tools & Manufacture 51, pp. 958-965

Bosch Rexroth (2022) https://apps.boschrexroth.com/microsites/ctrlx-automation/en/

L. Consolini, M. Locatelli, A. Minari (2021) A sequential Approach for Speed Planning under Jerk Constraints, Optimization and Control (math.OC), doi 10.48550/ARXIV.2105.15095

K. Erkorkmaz, Q.-G. Chen, M.-Y. Zhao, X. Beudaert (2017) Linear Programming and Windowing based Feedrate Optimization for Spline Toolpaths, CIRP Annals - Manufacturing Technology 66, Elsevier, pp. 393-396

T. Haas, S. Weikert, K. Wegener, (2019) MPCC-Based Set Point Optimisation for Machine Tools. International Journal of Automation Technology. 13. 407-418. 10.20965/ijat.2019.p0407

R. Herzog, Ph. Blanc (2019) Optimal G^2 Hermite Interpolation for 3D Curves, Computer-Aided Design, Elsevier, Vol. 11

T. Mercy, N. Jacquod, R. Herzog, G. Pipeleers (2019) Spline-Based Trajectory Generation for CNC Machines, IEEE Transactions on Industrial Electronics, Vol. 66, Issue 8

OpenCN (2019) https://opencn.heig-vd.ch/

A. Schorderet, R. Herzog, N. Jacquod, Y. Marchand, Ch. Prongue (2019) Productivity Increase of High Precision Micro-Milling by Trajectory Optimization, Modern Machinery (MM) Science Journal, pp. 3179-3186

T. Staroveški, D. Brezak,T. Udiljak, D. Majetić (2009) Implementation of a Linux-based CNC open control system, 12th Int. scientific conference on production engineering

D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, M. Diehl (2009) Time-Optimal Path Tracking for Robots: A Convex Optimization Approach, IEEE Trans. on Automatic Control, Vol. 54, No. 10, pp. 2318-2327

E.T.Y. Lee (1989). Choosing nodes in parametric curve interpolation. Computer-Aided Design, 21:363–370, 1989

A. Warfield, K. Fraser, S. Hand, T. Deegan (2005) Facilitating the development of soft devices, USENIX annual technical conference

P. Gerum (2016), Xenomai 3 : hybride et caméléon, Editions Diamond

Chiron (2020) https://www.factory5.tech/micro-5

Precitrame (2021) https://precitrame.com/en/product/k5/