VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV STROJÍRENSKÉ TECHNOLOGIE
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MANUFACTURING TECHNOLOGY

VYUŽITÍ PARAMETRICKÉHO PROGRAMOVÁNÍ PRO OBRÁBĚNÍ OBECNÝCH PLOCH

THE USE OF PARAMETRICAL PROGRAMMING FOR COMPLEX PART MACHINING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Jan Skácel

VEDOUCÍ PRÁCE
SUPERVISOR
prof. Ing. Miroslav Píška, CSc.

BRNO 2015

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav strojírenské technologie
Akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Jan Skácel

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Strojírenská technologie a průmyslový management (2303T005)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Využití parametrického programování pro obrábění obecných ploch

v anglickém jazyce:

The use of parametrical programming for complex part machining

Stručná charakteristika problematiky úkolu:

Úvod
1. Teoretický rozbor problému
2. Navržené varianty řešení - matematické, technologické
3. Experimentální zkoušky
4. Diskuze výsledků
5. Závěry

Cíle diplomové práce:

Aplikace parametrického CNC programování na složité tvary výrobků, zejména funkčních tvarů se zvýšenými požadavky kvalitu opracování.

Seznam odborné literatury:

LYNCH, Mike. Parametric programming for computer numerical control machine tools and touch probes: CNC's best-kept secret. Dearborn: Society of Manufacturing Engineers, 1997, 433 s. ISBN 0872634817.

KRUTH, J.P., LAUWERS, B., DOTREMONT, J., DEJONGHE, P. Optimised NC-toolpath generation for 5-axis machining of complex surfaces. Machining impossible shapes. Kluwer Academic Publishers, 1999, pp. 343-350, ISBN 0-412-84680-2.

Merdol, D.S ., Altintas, Y., 2008, Virtual Simulation and Optimization of Milling Operations: Part I – Process Simulation", Trans. ASME, J. Manufac. Sc. and Eng., vol. 130, pp. 051004.

MATTSON, Mike. CNC programming: principles and applications. Albany: Delmar, 2002, 358 s. ISBN 0766818888.

MCMAHON, Chris a Browne JIMMIE. CADCAM principles, practise and manufacturing management. 2nd ed. Harlow: Prentice Hall, 1998, 665 s. ISBN 0201178192.

Vedoucí diplomové práce: prof. Ing. Miroslav Píška, CSc.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2014/2015.

V Brně, dne 21.11.2014

L.S.

prof. Ing. Miroslav Píška, CSc. Ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. Děkan fakulty

ABSTRAKT

V práci je obsažen teoretický úvod do programování v G-kódu, základní matematický aparát a metody programování obecných křivek a ploch. Je provedeno několik praktických ukázek a jsou prezentovány možnosti pro další sofistikovanější programy.

Klíčová slova

parametrické programování, Sinumerik, BSPLINE, NURBS, cykly, pětiosé frézování

ABSTRACT

Thesis consists of theoretical introduction to programming in G-code, underlying mathematical principles and methods how to program general curves and surfaces. There are seven examples and several ways how to make even more sophisticated programs are presented.

Key words

parametric programming, Sinumerik, BSPLINE, NURBS, cycles, five axis milling

BIBLIOGRAFICKÁ CITACE

SKÁCEL, Jan. *Využití parametrického programování pro obrábění obecných ploch.* Brno 2015. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav strojírenské technologie 81 s. 7 příloh. Vedoucí práce prof. Ing. Miroslav Píška, CSc.

PROHLÁŠENÍ

Prohlašuji, že jsem diplomovou práci na téma **Využití parametrického programování pro obrábění obecných ploch** vypracoval samostatně s použitím odborné literatury a pramenů, uvedených na seznamu, který tvoří přílohu této práce.

_____ .....

Datum Bc. Jan Skácel

PODĚKOVÁNÍ

Děkuji tímto prof. Ing. Miroslavu Píškovi, CSc. za cenné připomínky a rady při vypracování diplomové práce a pomoc v profesním růstu. Dále pak děkuji Ing. Aleši Polzerovi, Ph.D. za konzultace k použití systému Sinumerik a panu Jiřímu Čechovi za jeho čas a spolupráci při experimentech.

OBSAH

ÚVOD

Obráběcí stroje (*machine tools*), jak je známe, se objevily už v době průmyslové revoluce. Konvenční stroje jako soustruh (*lathe*), frézka (*milling machine*), nebo vrtačka (*drill press*) jsou ovládány zkušeným obráběčem, který opatrně otáčí ručními kolečky a pákami stroje, aby nastavil obrobek nebo nástroj do požadované polohy. Technologický pokrok přinesl počítačově řízené číslicové stroje (CNC), které zajistili vyšší kvalitu, konzistentnost a zvýšili produktivitu. Stroje stále provádí téměř stejné operace, ale pohyb stroje je řízen elektronicky místo rukou. I přes vyšší náklady jsou obvykle opodstatněné, právě díky jejich produktivitě [1].

V průběhu let se snadnost použití, flexibilita a možnosti CNC obrábění zlepšila obrovským způsobem. První číslicově řízené (NC) stroje byly těžkopádné, slabé a neflexibilní. Typický program musel být napsán ve velmi striktní syntaxi na děrný štítek nebo pásek, jen aby stroj vykonal základní operace. Pokud programátor provedl chybu, celý štítek nebo pásek se musel vyměnit. Mnoho vlastností, které dnes považujeme za samozřejmost, muselo být teprve vyvinuto [1].

Dnes je použití některých nástrojů, které usnadňují programování, vyučováno jako základ a většina programátorů si jimi ulehčuje práci na denní bázi. Patří mezi ně například kompenzace geometrie nástroje, interpolační metody a v neposlední řadě cykly, díky nimž lze s lehkostí vyrobit například složitější závit. Některé novinky byly veřejností přijaty okamžitě, příkladem může být zadávání hodnot v desetinném tvaru kdy místo X24300 stačí zadat pro člověka přirozenější X2.43, pokud mu jde o posuv o/na 2.43mm. Jiné implementace do G-kódu nebyly přijaty s takovým nadšením, nedostalo se jim dostatečného pochopení a využití. Jednou z takových funkcí může být právě parametrické programování, které bylo představeno už před více než 30 lety [1].

Většina kvalitní literatury je psána v anglickém jazyce, nadnárodní korporace angličtinu používají často jako primární nebo sekundární jazyk komunikace a webové stránky, kde programátor bude nejspíše hledat odpověď na svůj problém jako první, také majoritně nalezne v tomto jazyce. Angličtina se stává jakýmsi celosvětovým standardem napomáhajícím globalizaci a rychlosti šíření informací. Z tohoto důvodu je pro odborníky angličtina obvykle nutnou podmínkou a to včetně technických výrazů, které jsou obvykle ustálené a nelze je tedy překládat volně. Z tohoto důvodu jsou v práci také klíčové pojmy uvedeny i v anglickém jazyce a to obvykle v závorce, psané kurzívou. Některé nejsou pro jednoznačnost překládány vůbec, jako např. swivel a frame.

Tato práce si klade za cíl prozkoumat možnosti využití parametrického programování a obrábění obecných ploch. Teoretická část je okleštěna pouze na zajímavosti a pozornost je věnována především postupům výroby součástek a NC programům v závěru práce, se stěžejním programem pro výrobu vaček.

1 TEORETICKÝ ROZBOR PROBLÉMU

Parametrické programování nebylo původně vyvinuto výrobci CNC[1] kontrolerů[2] pro použití CNC uživateli, ale aby umožnili výrobcům nástrojů a doplňků integrovat schopnosti vyšších programovacích jazyků na úroveň G-kódu[3]. To mělo za následek jakousi těžkopádnost při použití při porovnání s jinými vysokoúrovňovými jazyky jako je C, Pascal, nebo Basic, natož na moderní Ruby, nebo Python se svými frameworky a jiné [1].

Na příklad mnoho systémů sond musí být naprogramováno na úrovni G-kódu a zároveň existuje mnoho věcí, které nelze řešit za použití normálních G-kód příkazů. Sonda (*probe*) musí například po kontaktu s obrobkem (*workpiece*) zaznamenat svojí polohu a použít ji v CNC programu. Toho lze dosáhnout pouze za použití parametrického programování, zavoláním systémové proměnné [1] [2].

Na parametrické programování je možné narazit pod několika jmény. Fanuc a každý kontroler, který tvrdí, že je stoprocentně s ním kompatibilní ho označuje jako *custom macro* (vlastní makro); Fadal ho označuje *macro*; Okuma *user task* (uživatelův úkol); Sodik *Q routine* (Q program); Kearney & Trecker *advanced programming language* (pokročilý programovací jazyk); Sharone *arithmetics* (aritmetika); a některé firmy, jako Bridgeport, ho neoznačují nijak a je přirozenou součástí jejich G-funkcí. Jednotliví výrobci mohou mít také několik verzí programovacích funkcí. Například Fanuc a Okuma mají verzi A a B, respektive 1 a 2, kdy druhé, dražší možnosti umožňují více věcí a jsou jednodušší na použití [1].

I přesto, že jednotlivé variace verzí parametrického programování se liší výrobce od výrobce a vedou k jiným specifickým technikám použití, smysl a širší smysl zůstává napříč výrobci velmi podobný. V teoretické části 1.2 jsou kvůli předepsané literatuře prezentovány především funkce systému Fanuc a pro teoretickou část 1.3 a praktickou část byl volen systém Sinumerik, vzhledem k dostupnosti fakultního pětiosého obráběcího centra s tímto systémem [1].

Pro obrábění obecných ploch je zapotřebí poměrně komplexní matematický aparát, který je potřeba pro modelování a reprezentaci geometrií. Ten je přiblížen v kapitolách, které následují za úvodem do programování, tedy v kapitole 1.6.

---

[1] CNC = computer / computerized numerical control; počítačově číslicově řízené stroje

[2] Kontroler, též řídicí systém, nebo řídící jednotka (*MCU, machine control unit, controller*) je jednotka odpovědná za čtení a interpretaci uloženého programu a použití těchto instrukcí pro ovládání stroje skrze aktuátory [7].

[3] G-kód, známý též jako ISO-kód nebo RS-274, je nejpoužívanější programovací jazyk pro ovládání (C)NC strojů. Syntax a význam některých kódů se může lišit dle výrobce kontroleru stroje [28].

1.1 Úvod do obrábění

G-kód je tzv. nízký programovací jazyk, založený na binárních číslech (BCD systém – binary-coded decimal). Protikladem může být APT, tzv. vysoký jazyk, používaný dnes především pro EDM stroje, nebo lasery [3].

Postup tvorby programů pro výrobu složitější součástky může probíhat následovně [3] [4]:

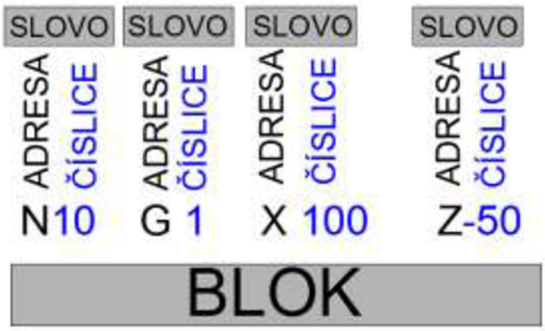
- nastudování výrobního výkresu, rozmyšlení upnutí a použití nástrojů,
- zvolení částí, které budou zadávány pomocí cyklů, maker, parametricky,
- za pomocí flowchartu rozvržení počtu podprogramů a tvorba logické struktury,
- výpočet nebo volba řezných podmínek pro výrobní operace,
- sestavení kódu,
- simulace, ověření a případné iterace.

1.2 Programování v G-Kódu

1.2.1 Syntaxe

NC programy jsou složeny z kódů, které jsou čteny řídící jednotkou, aby provedly rozličné funkce. Existují dva odlišné typy kódů. Ty, které ovládají funkce a nastavení stroje a ty, které se používají na modifikaci a exekuci výrobních funkcí [4].

Každý program je složen z bloků, též řádků, nebo vět, které obsahují jednotlivá slova. Řídící jednotka (*MCU - machine control unit*) čte několik bloků dopředu, ale v jednu chvíli provede vždy jen daný, následující blok. Každý blok je po přečtení strojem převáděn na jednotlivé úkony, operace, jako např. změna polohy nástroje nebo suportu, zastavení vřetene, apod. Jednotlivá slova sestávají z adresy a čísla. Doporučované pořadí adres slov v bloku je N G X Y Z F S T D H M, tedy číslo řádku, G-kódy, všechny ostatní adresy, M-kódy. Bloky však nemusí obsahovat všechny, jednak protože je nepotřebuje programátor deklarovat, nebo je může vynechat, jestliže se dědí z minulých bloků, tedy není potřeba je měnit (více o modalitě v kapitole 1.2.6). Zároveň nesmí být v jenom bloku použity funkce ze stejné skupiny, např. protichůdné G0 a G1, M08 a M09, apod. [4] [5].



Obr. 1.1 Struktura bloků [6].

Slova se dají dělit na rozměrová, která značí fyzikální veličinu, jako je vzdálenost, nebo rychlost, a na slova bezrozměrová, která označují specifickou funkci dané adresou,

tedy číslem. Příkladem je X20 jako posuv o 20, nebo na souřadnici X = 20, dle zvoleného bezrozměrového slova které mu předcházelo, nebo bylo implicitně zvoleno, tedy G90 (absolutně) nebo G91 (inkrementálně). Blok by tedy mohl vypadat následovně:

```
N140 G1 X20
```

a za předpokladu výchozího nastavení, nebo posledního platného deklarování inkrementálního programování a jednotkách v milimetrech by se daný stroj na řádku 140 (obvykle 14. řádku) posunul o 20 mm v ose x [5].

Je zakázáno psát mezery ve slově, tzn. mezi adresu a jeho číslici (X 6.0), nebo v G- a M- kódu (G 01). Podobně chybu způsobí tabulátor. Naopak povoleno je mezery vynechávat plně, což ale není doporučováno pro přehlednost kódu. Typická řídící jednotka povoluje tři místa před a čtyři místa za desetinnou čárkou (př. 123.1234). Nuly, které neovlivňují faktický význam hodnoty, se před a po zápisu mohou a nemusí uvádět a v závislosti na řídící jednotce pokud není desetinná tečka uvedena, může hodnota reprezentovat 1/10 000 milimetru. Pak mohou být ekvivalentní všechny následující zápisy [4]:

```
X.5, X.50, X.500, X.5000, X0.5000, X5000
```

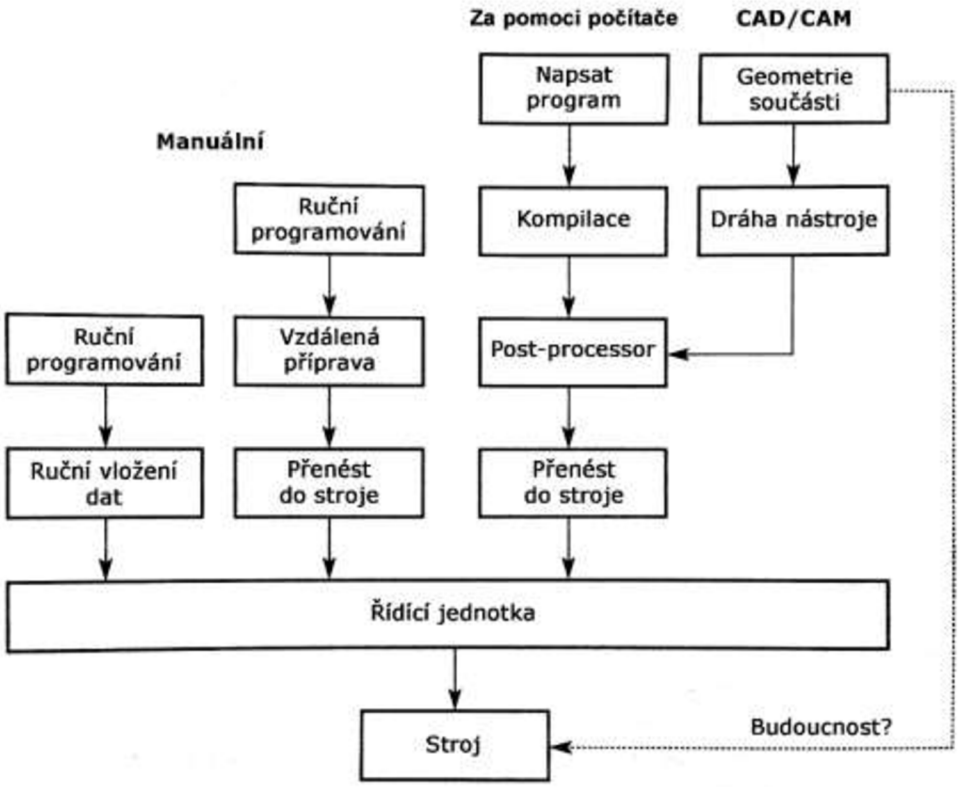
Bloky je také možno komentovat za pomocí kulatých závorek, čehož se dá využít především pro snazší orientaci v kódu, kdy si programátor může poznačit, co daný blok dělá, nebo např. který nástroj volí. Zároveň program obvykle nepíše stejný člověk jako ten, kdo obsluhuje stroj a je tak dobré předat v komentářích důležité informace (které by měly být i součástí dokumentace). Některé z vhodných informací, které by mohl program obsahovat jsou: lokace posunutého počátku; druh, délka a průměr použitých nástrojů; pokyny k upnutí; upozornění na předchozí chyby; pro kterého zákazníka díl je a číslo dílu; jednotlivé technologické procesy v programu; informace o programu jako datum, název souboru, jméno programátora. V praxi se tohoto limitu běžně nedosahuje, ale je třeba mít na paměti, že délka bloku je omezena obvykle na 512 znaků [4] [6].

1.2.2 Zápis a verifikace

Dílenský program lze vytvořit ručním zapisováním G-kódu do ISO programu (*offline programming*), ručním zadáváním do dialogového okna na stroji (*online / shop floor programming*), nebo vygenerováním CAD/CAM softwaru a jim podobných. Ruční zadávání také může využít počítačové podpory např. na dopočet korekcí nástroje nebo matematické výpočty a CAD/CAM software vytváří nejprve na stroji nezávislé reprezentace křivek a povrchů, které následně konvertuje pomocí programu, kterému se říká post-procesor (viz Obr. 1.2). Tato práce se zabývá především prvním způsobem zadávání, které je výhodnější pro rychlejší tvorbu a editaci, možnost verifikace pomocí simulačních programů [2] [7].

Verifikační software může být rozdělen na dvě kategorie. Verifikace cesty nástroje (*toolpath backplotting*) a 3D verifikace (*solids verification*). První software vykreslí cestu středu nástroje pro každý blok kódu. Programátor může zkontrolovat, jestli tato cesta odpovídá jeho záměru a předepsanému výkresu. Pokud by provedl při zadávání chybu například v souřadnici na řádku N100 a předpis měl vypadat takto:

```
N100 G01 X2.5 Y1.5
```



Obr. 1.2 Alternativní možnosti na tvorbu programu [7]

a místo toho vepsal do programu nesprávnou hodnotu:

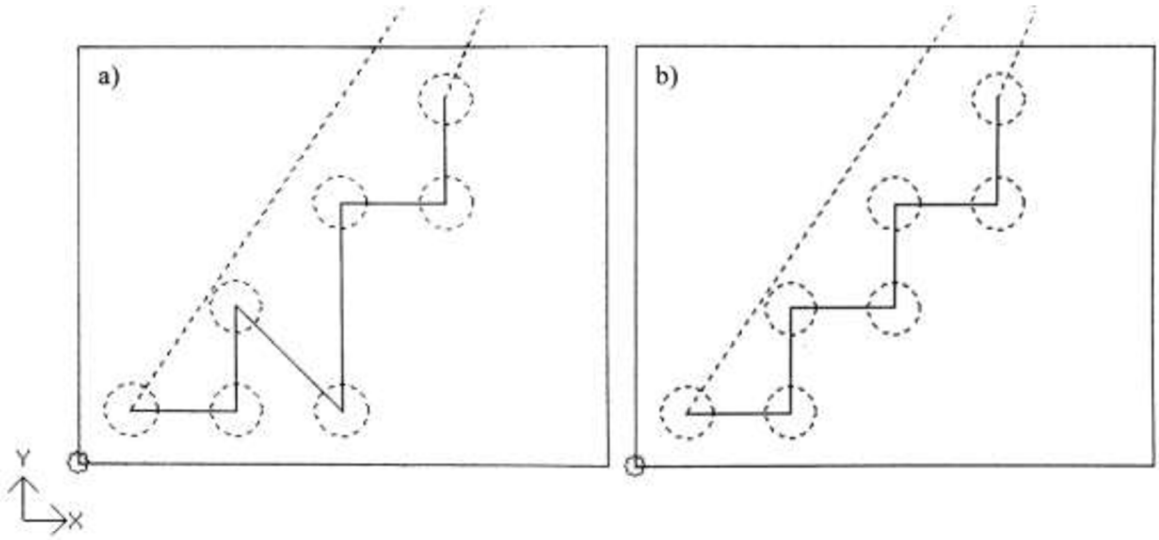
```
N100 G01 X2.5 Y0.5
```

byla by chyba snadno přehlédnutelná v kontrole programu programátorem, jelikož zápis nepůsobí podezřele kvůli validní syntaxi. Chyba by byla pochopitelně zjištěna ihned při prvním pokusu o reálný běh programu a obrobení polotovaru. V tuto chvíli by šlo ale již o nákladnou chybu. Výhoda zachycení chyby již na PC je tedy zřejmá. Verifikace cesty nástroje by tedy v programu vypadala jako např. na Obr. 1.3a) a správná cesta na Obr. 1.3b) [4].

Druhá metoda zobrazí obrobek i nástroj v čase v plně vyrenderovaném 3D modelu a obráběcí proces je tak snazší na představu a tím i ověření [4].

Když je program verifikován v počítači, je potřeba ho ověřit i na stroji. Po přenesení programu do stroje by ani zkušený programátor neměl nikdy spustit proces při plné rychlosti do bloku zamýšleného obráběného materiálu. I přes počítačové ověření se stále mohou vyskytnout různé chyby, které se projeví až na stroji [4].

Příkladem takovýchto chyb, které mohou nastat, jsou uvedeny v následující tabulce (Tab. 1.1). Následek chyby může být zlomený nástroj, zničený obrobek, poškozený stroj, lidské zranění [4].



Obr. 1.3 Vykreslená a) nechtěná cesta nástroje a b) korektní cesta [4].

Tab. 1.1: Příklady chyb a možných následků, nutných verifikovat před samotným procesem odběru materiálu [4].

Chyba	Následek
Vřeteno nebylo zapnuto před obráběním	Zničený nástroj a zničený obrobek
Nástroj se nevrátil do bezpečné vzdálenosti nad dílec před posun nad druhou kapsu	Zmetkový obrobek
Chlazení nebylo zapnuto před obráběním	Zničený nástroj
Svěrák nebo upínka v cestě nástroje	Zlomený nástroj a poškozená upínka
Špatné pořadí operací	Díry vystřeďovány před vrtáním, následkem srážka
Nesprávný nástroj	Díl vyroben se špatnými rozměry
Špatná korekce nástroje	Nástroj příliš vysoko nebo nízko, srážka nebo špatné rozměry
Zadaný nesprávný posuv nebo rychlost	Zničený nástroj, obrobek vytržen z upínek

Správné pořadí ověřovacího procesu je [4]:

1. Verifikace kódu na PC pomocí vizualizace dráhy nástroje.
2. Dokončení kontrolního seznamu ověření kódu.
3. Nahrání programu na stroj a spuštění grafické simulace pokud je dostupná.
4. Zkušební chod:
    a. zapnutí módu čtení po bloku (*single block mode*),
    b. snížení posuvu a rychlosti rychloposuvu,
    c. nastavení kompenzace v ose Z na bezpečnou vzdálenost nad obrobek,
    d. zkušební chod 1,
    e. zkušební chod 2 s vypnutým módem čtení po bloku.
5. Navrácení kompenzace v ose Z do normálu a provedení zkušebního chodu 3 na materiálu, nebo zkušebním, měkčím materiálu (často dřevo, plast, vosk nebo vysoko-hustotní pěna) se sníženými hodnotami rychlostí a posuvu.
6. Kontrola obrobku a případná úprava kompenzací pro přesné rozměry.
7. Běh prvního ‚reálného' výrobku v automatickém módě.

V druhém bodě ověřovacího procesu byl uveden kontrolní seznam (*checklist*) ověření kódu, který je [4]:

- ☐ Všechny rychloposuvy jsou nad vrchní částí obrobku.
- ☐ Čísla nástrojů a jejich korekcí se shodují.
- ☐ Nástroje ve stroji se shodují s těmi vypsanými na technologickém postupu (*setup sheet*).
- ☐ Všechny nástroje byly změřeny (*touched off*) a hodnoty korekcí jsou rozumné.
- ☐ Ponorné frézování (*plunge cuts*) je realizováno frézou se zubem do středu (*center-cutting tool*).
- ☐ Řezné rychlosti a posuv jsou rozumné.
- ☐ Vřeteno (*spindle*) se točí ve správném směru.
- ☐ Chlazení je zapnuto před obráběním.
- ☐ Hrubovací operace jsou před dokončovacími.
- ☐ Nástroje jsou v bezpečné vzdálenosti v ose Z, než se přesouvají na další úkon.
- ☐ Operace na výrobu děr jsou prováděny ve správném technologickém pořadí (př. vrtání, vyvrtávání, vystružování)
- ☐ Upínky jsou utaženy pevně a kolmo.
- ☐ Nástroje jsou upevněny ve svých držácích.

1.2.3 Struktura

Programy mají obvykle podobné elementy, neboli skupiny bloků sloužící dané technologické operaci a tvořící tak strukturu programu. Referenční příklady korespondují s tříosou frézkou s řízením FANUC [3].

Spuštění

Nejprve program deklaruje svůj začátek (%) a svoje jméno (O1000). Tyto dva bloky musí být bez čísla řádku (N**) a nemají žádný efekt na samotné obrábění, ale jsou důležité pro řídící jednotku, aby rozeznala jeden program od druhého [4] [3].

Dále může následovat takzvaný bezpečnostní blok (*safe line*), který nastaví stroj do výchozích hodnot a zamezí se tak možnosti, že v nich stroj není z různých příčin již nastaven. V bloku jsou instrukce, které stroji nastaví, jak si mi v průběhu programu jednotlivé příkazy interpretovat. Řádek zruší veškeré kompenzace nástroje, zvolí milimetry za jednotky a určí počátek souřadného systému, zruší cykly a nastaví u nich návrat do základní roviny a nastaví absolutní programování [4]:

```
N10 G20 G40 G54 G80 G90 G98
```

Poté je vhodné bezpečný návrat do počátku a volba nástroje. Příklad takového spouštěcího bloku (*start-up block*) je [3] [4]:

```
%                          (vyžadováno na začátku a konci programu)
:1234                      (název programu, u soustruhu O a 4 číslice)
N10 G20 G40 G54 G80 G98    (bezpečnostní blok)
N20 G91 G28 Z0             (návrat do počátku v ose Z)
N30 M06 T01                (volba nástroje)
N40 G43 H01                (kompenzace nástroje)
N50 G00 G90 X** Y** S**** M03
                           (rychloposuvem na absolutní souřadnice XYZ
                           roztočit vřeteno ve směru hod. ruč.)
N60 Z5 M08                 (přiblížení k obrobku; zapnutí chlazení)
```

Výměna nástroje

Další typickou skupinou bloků je výměna nástroje. Nejprve je potřeba původní nástroj dostat bezpečně od součásti do počátku, alespoň v ose Z, další blok dle výrobce stroje není potřeba a vypnutí korekce, chlazení a otáček se provede automaticky. Následuje příkaz pro samotnou výměnu nástroje, čtvrtý blok skupiny je zapnutí kompenzací nástroje. Předposlední část je případná volba počátku, rychloposuvem do absolutně zadaných

souřadnic nad obrobek a roztočení vřetene. Poslední blok je zapnutí chlazení a návrat do blízkosti obrobku v ose Z [3] [4]:

```
N** G91 G28 Z0
N** G49 M05 M09
N** M06 T**
N** G43 H**
N** G00 G90 G56 X** Y** S**** M3
N** Z5 M08
```

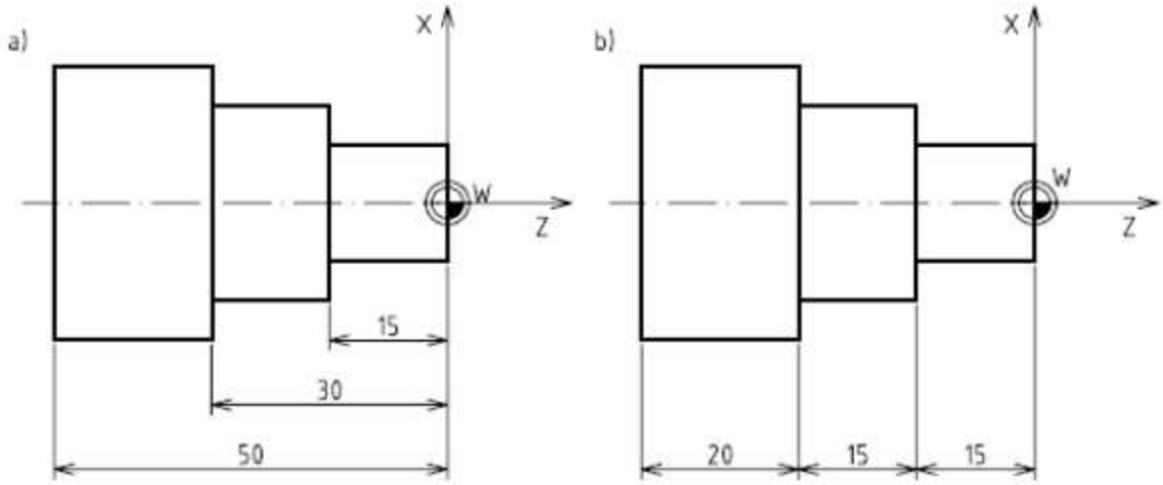
Ukončení

Poslední bloky pro ukončení programů jsou obvyklé funkce a několik administrativních povinností jako zastavení vřetene, chlazení, korekcí, reset programu a deklarace řídící jednotce, kde končí zápis programu. M30 je nejpoužívanějším slovem na signalizaci řídící jednotce, že kód skončil a má dojít k resetu paměti na počátek programu. Všechna modální slova zůstanou v paměti uložena, což lze reflektovat jak ukončením na konci programu, tak přenastavením na jeho začátku. Kód M30 je možné použít kdekoliv v kódu a po něm může následovat např. kód, který není v danou chvíli žádoucí provádět, nebo interní podprogram (*subroutine*). Zápis končí a na posledním řádku tak je symbol procenta (%) [3] [4]:

```
N** G00 Z** M9
N** G91 G28 Z0 M5
N** M30
%
```

1.2.4 Absolutní a inkrementální programování

Základním rozdílem v zadávání souřadnic je programování absolutní a inkrementální. Při absolutním programování, v G-kódu volen pomocí G90, jsou souřadnice zadávány vzhledem k počátku souřadného systému, obvykle voleného jako nulový bod někde na polotovaru. Při inkrementálním programování, zvaném též přírůstkovém jsou souřadnice zadávány vzhledem k poslednímu bodu, zadává se tedy změna souřadnic oproti výchozímu stavu (viz Obr. 1.4). Využití nachází metody jednak v logickém zadávání obrábění, tedy při volbě kótování od základny, nebo při použití řetězových kót, a jednak v rámci technologických postupů, kdy je například u určitých částí parametrického programování vhodnější použití inkrementálního programování (konkrétním případem je vyjetí nástroje nad obrobek po vrtání), nebo naopak vhodnost absolutního, např. při najetí do pozice na výměnu nástroje.



Obr. 1.4 Rozdíl mezi absolutním a inkrementálním programováním [24]

1.2.5 Tabulky kódů

Tab. 1.2 Význam nejpoužívanějších adres, seřazených dle doporučovaného pořadí [4] [3] [8]

Adresa	Popis	Dodatečné informace
N	Číslo řádku	Obvyklé inkrementy 10, 20, 1000 pro snadné vložení bloků,
G	Přípravná funkce	Viz samostatná tabulka 1.3
M	Přídavná pomocná strojní funkce	Viz samostatná tabulka 1.3
X, Y, Z	Základní osy souřadného systému	
A, B, C	Rotace kolem základních os	
I, J, K	Parametry interpolace nebo stoupání závitů ve směru os	Obvykle pro G02 a G03; parametr pro některé cykly
P, Q, R	Pohyb paralelně podél základní os	Nebo proměnné v makru
U, V, W	Druhý pohyb paralelně podél základní os	
P	Volitelný parametr pro více G a M kódů	Např. pro G04, M98, M99
Q	Inkrement v cyklech	Např. G73, G83
R	Poloměr oblouku / volitelný parametr podprogramu	Také návratová výška v cyklech pokud G99
F	Rychlost posuvu	Obvykle [mm/min], nebo mm na otáčku
S	Otáčky vřetene / konstantní řezná rychlost	Obvykle [1/min], [m/min]
T	Výběr nástroje	
D	Paměť korekce nástroje	
L	Čítač smyčky	Nebo specifikace G10
H	Délková korekce nástroje	Nebo typ operace makra
O	Název programu	Např. O2501; samostatný blok
Pozn.:	Psaní N adres není závazné, s výjimkou M99 P adresy a skoků, kdy je psaní nutné. Je také možné psát N slova jen ve zvolených blocích.	

Tabulka 1.2 čítá pouze obvyklé významy adres, v praxi se liší dle výrobce a konkrétního zařízení, zvláště pak G-kódy v druhé tabulce jsou specifické pro FANUC. Je tedy nutné nastudovat manuál ke každému danému přístroji zvlášť.

1.2.6 Modálnost

Některé G-funkce jsou modální, to znamená, že ovlivní daný blok a zbytek programu, dokud nejsou přepsány funkcí ze stejné skupiny, nebo nejsou ukončeny. Je to tedy jejich schopnost zůstat v paměti řídící jednotky. Ostatní, nemodální funkce ovlivní pouze příslušný blok a nejsou dále pamatovány [4] [3].

Kódy jsou seskupeny do několika rozdílných kategorií, ovládající podobné funkce, jako je viděno v tabulce 1.3. Jak již bylo v kapitole o syntaxi naznačeno, v jednom bloku může být volán pouze jeden příkaz z dané skupiny. Tedy nelze kombinovat například lineární interpolaci G1 s rychloposuvem G0, nebo absolutní programování G90 s inkrementálním G91. V jednom bloku však lze však kombinovat více funkcí z různých skupin, například G90 G1, jelikož nejsou příbuzné a tedy netvoří logický konflikt [4].

Výhody modálních kódů jsou zřejmé. Při sérii několika lineárních interpolací G1 tak například není třeba zadávat tento kód v každém bloku a lze jej vynechat, podobně, jako při změně směru není potřeba udávat veškeré dostupné souřadnice (X** Y** Z**), ale pouze ty, které se mění. Z dlouhého zápisu [4]

```
N10 G01 X1.5 Y3.0
N20 G01 X6.0 Y3.0
N30 G01 X6.0 Y6.0
```

tak lze pouhým využitím modálnosti zapsat ekvivalentní

```
N10 G01 X1.5 Y3.0
N20 X6.0
N30 Y6.0
```

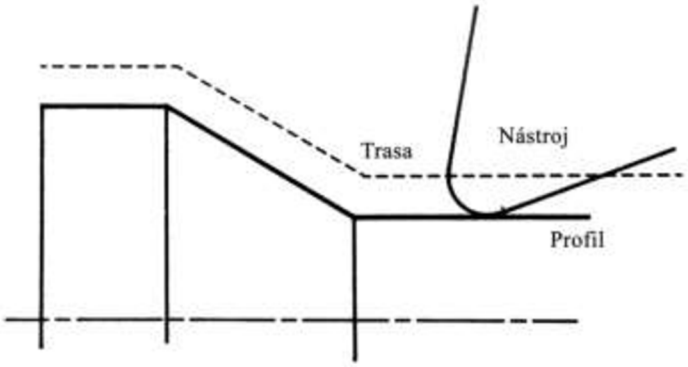
Tab. 1.3 Skupiny kódů pro Fanuc s ohledem na modálnost. Pokud jsou kódy platné i pro Sinumerik, je uvedeno číslo skupiny, kvůli adresaci systémovou proměnnou [4] [9].

Funkční skupina [číslo skupiny]	Kódy a adresy
Modální	
Pohyb os [1]	G00, G01, G02, G03, ASPLINE, POLY,..
Systém polohování [14]	G90, G91
Jednotky [13]	G20, G21 (G70, G71)
Cykly	G80, G81, G82, G83, G84, G85,…

Souřadné systémy [8]	G500, G54, G55, G56,…
Kompenzace rádiusu nástroje [7]	G40, G41, G42
Kompenzace délky nástroje	G43, G44, G49
Návratové roviny pro cykly	G98, G99
Volba pracovní roviny [6]	G17, G18, G19
Pozice os	X, Y, Z, A, B, C, U, V, W
Další adresy	D, F, H, I, J, K, Q, R, S
Chlazení	M07, M08, M09
Kontrola vřetene	M03, M04, M05
Nemodální	
Různé G-kódy	G04, G28, G29, G92
Pomocné adresy	L, P
Různé M-kódy	M00, M01, M02, M06, M30, M97, M98, M99

1.2.7 Korekce nástroje a konstantní řezná rychlost

Korekce nástroje, nebo také kompenzace (*cutter compensation*, nebo *tool nose radius compensation*) je jednak délková a určuje velikost nástroje v ose Z, zapíná se kódem G43 a vypíná pomocí G44 a jednak pro poloměr břitu. Ta je používána, protože nástroje, ani při soustružení, neodebírají materiál v bodě, ale na zakřivené špičce nástroje. Bez korekce by bylo nutné při psaní kódu s touto geometrií počítat a program zadávat pro dráhu nástroje, nikoliv geometrii součásti. Na Obr. 1.5 je vyobrazeno soustružení součástky, kde čárkovaná čára znázorňuje polohu středu nástroje, která je definovaná zvětšením profilu o rádius nástroje. Zároveň tak s použitím korekcí není nutné při změně nástroje měnit celý program [7].



Obr. 1.5 Korekce trasy středu nástroje pomocí ekvidistanty od povrchu geometrie [4].

Obrábění pomocí konstantní řezné rychlosti (*constant surface speed, CSS*) je funkce (G96) především pro soustruhy, která využívá automatického upravování otáček vřetene pro dodržení řezné rychlosti, která by se jinak při stejných otáčkách vřetene měnila v závislosti na poloměru obrobku. Programátor určuje buď řeznou rychlost, nebo otáčky při daném poloměru a řídící jednotka dopočítá hodnoty až do specifikovaného maxima (G50) [7].

V praktické části byla snaha pro obecnou 3D plochu použít automatickou korekci systému Sinumerik, CUT3DF, která buď selhala, nebo byla špatně použita. V kapitole 2 je proto navržena metoda pro výpočet kompenzace.

1.2.8 Kódy pro pohyb os

G00 - Rychloposuv

Rychloposuv, rychlé polohování, nebo také indexace slouží k rychlé změně polohy pro ušetření času. Nikdy není používán pro kontakt s obrobkem, na to je rychlost příliš vysoká a nekonstantní. Jako všechny G-kódy se v zápisu udává koncový bod v alespoň jedné ose (př. N70 X5). Výchozí stroj zná, je identický s aktuální polohou. Při rychloposuvu k obrobku by se měl dodržet určitý odstup, především kvůli tvarovým nepřesnostem součásti a upnutí a následně se volá G-kód pro odběr materiálu. Podobně je dobrá praxe při přibližování k obrobku nejdříve polohovat osy X a Y, před osou Z, př. [4]:

```
N10 G00 X3.0 Y4.0
N20 G00 Z5
N30 G01 Z-.5 F4.0
```

Je důležité vědět, že rychloposuv není vždy přemístění po přímce, naopak servomotory obvykle přemístí každou osu nejrychleji, jak lze, nezávisle na zbylých osách. Tedy na příkaz G91 G00 X2.0 Y4.0 by CNC stroj zahájil pohyb souběžně v obou osách a vykonal by tak v daném souřadném systému pohyb pod úhlem 45° a dokončil zbývající trasu v ose Y [4].

G01 - Lineární interpolace, strojní posuv

Pro kontrolovaný pohyb, takzvaný strojní posuv, nebo lineární interpolace se používá kód G01. Pohybuje nástrojem přes všechny zadané body po přímce specifikovanou rychlostí. Syntaxe je stejná jako u rychloposuvu ale s jedním přidaným slovem, které se skládá z adresy F a čísla udávajícího posuv. Kód se také používá pro zanořování / ponoření (*plunge*) do obrobku, kde je ale často nutné použít frézu se zubem do středu / ponornou frézu (*center-cutting end mill*), nebo snížit posuv na polovinu [4].

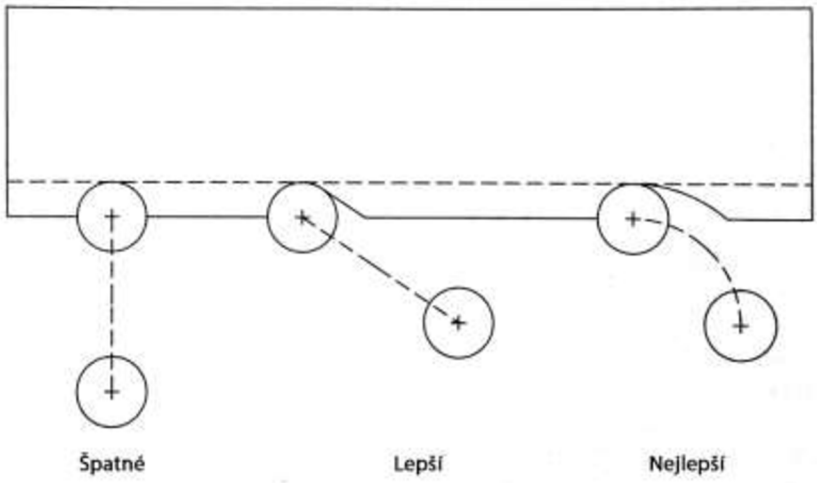
G02 a G03 - Kruhová interpolace

Pro interpolaci kruhů a jejich částí se používají kódy G02 (ve směru hod. ručiček) a G03 (proti směru hod. ručiček). Stejně jako u lineární interpolace se definuje posuv. Jsou dva možné způsoby zápisu,

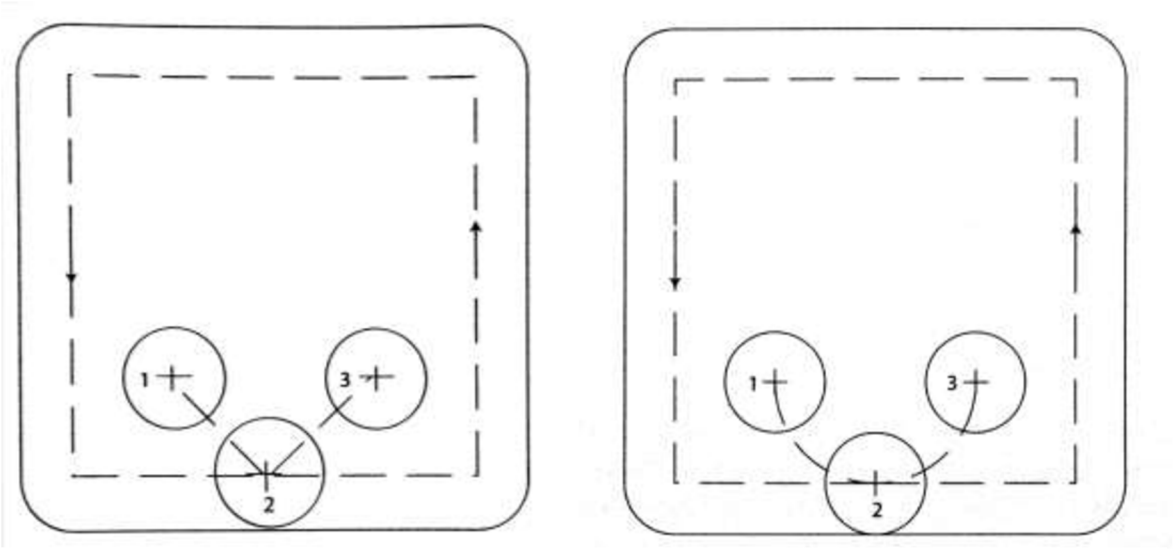
```
G02 Xn.n Yn.n Zn.n In.n Jn.n. Fn.n
G02 Xn.n Yn.n Zn.n Rn.n Fn.n
```

kde souřadnice X a Y určují konečný bod a mohou být zadány absolutně (G90) i inkrementálně (G91); pokud je použito Z, je tvořena spirála, což může být použito pro tvorbu závitů nebo ponoření do kapsy; rádius lze nadefinovat a) přírůstkovou vzdáleností v ose X (I) a Y (J) od počátku oblouku do středu oblouku, kde hodnoty I a J mohou tedy nabývat záporných nebo i nulových hodnot (0°, 90°, 180°, 270°), nebo b) pomocí hodnoty velikosti rádiusu, kde obvykle není možné provést plný 360° kruh a je třeba kód rozdělit do dvou a při obloucích větších jak 180°se používá záporná hodnota.

Při dokončovacích operacích je vhodné vstupovat do materiálu i vystupovat z něj pod malým úhlem, nebo po tangentě (*lead-in, lead-out*), viz Obr. 1.7 a Obr. 1.7. Nerovnoměrné zatížení materiálu a odezva servomotorů mohou zanechat na obrobku geometrickou nepřesnost.



Obr. 1.7 Vhodnost vstupu do materiálu [4]



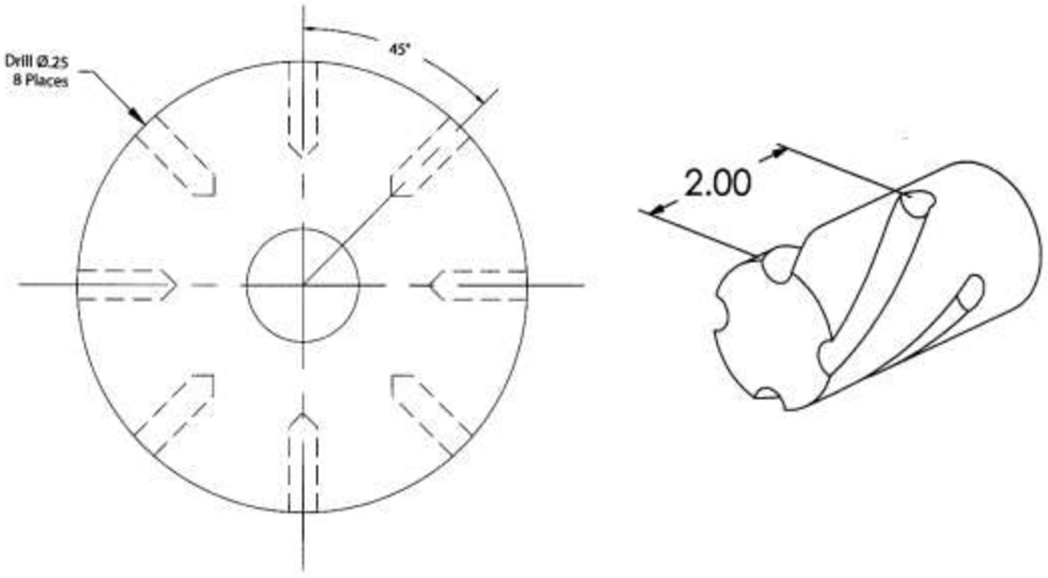
Obr. 1.7 Preferované dráhy nástroje při dokončovací operaci kapsy.
Vlevo vstup do materiálu a výstup z něj pod úhlem, vpravo po kružnici [4].

Programování rotační osy

Rotační osy jsou nejčastěji ve formě indexovací hlavy, nebo otočného stolu. Otočný stůl je zařízení, které lze připevnit k obráběcímu stolu a narotovat do kýžené pozice pomocí G-kódu. Označení rotující osy je obvykle A, sofistikovanější otočné stoly umí rotovat ve dvou osách a značí se A a B nebo C [4].

Rotační osa obvykle může být použita pro rychloposuv G0, strojní posuv G1 a cykly. Pokud je žádané pouze natočení rychloposuvem, jde o indexaci, nebo také poziční obrábění. Posuvu v rotační ose v kombinaci s lineárním posuvem pro obráběcí operace se říká simultánní interpolace ve 4 nebo 5 osách (*simultaneous fourth-axis interpolation*), pohyb os je tedy kontinuální. Toto rozdělení je potřeba kvůli CAD/CAM systémům [4].

Syntaxe je stejná jako v jiných osách, s několika výjimkami. Osa A má obvykle rozlišení 0,001 stupně a čím dále od středu osy je pohybováno, tím více narůstá geometrická lineární chyba. Dá se programovat absolutně i inkrementálně a to do kladných i záporných hodnot. A osa obvykle nemůže být natáčena do jednoho směru bez omezení a je povolen pouze daný počet stupňů. Pokud je tento limit v otáčkách dosažen, je potřeba osu odmotat na začátek. Příklady využití rotační osy jsou díly na [4].



Obr. 1.8 Příklady využití rotační osy pro obrábění radiálních děr a šroubovice [4]

G28 – návrat do strojního počátku

Pokud je potřeba odjet nástrojem od obrobku a to například při výměně nástroje, pro zkontrolování dílu, nebo zapnutí korekcí, lze použít kód G28. Když je slovo zavoláno, nástroj se okamžitě vrací rychloposuvem (jen) ve zvolených souřadnicích. Pokud není žádná osa specifikována, vrací se ve všech osách. Kód je dvou-krokový a z tohoto důvodu je také při čtení ‚blok po bloku' (*single block*) nutné pro provedení stisknout tlačítko dvakrát. Nejprve se nástroj přesune rychloposuvem do přechodného bodu (*intermediate position*), který je v kódu specifikován a následuje přesun do strojního počátku v těchto specifikovaných osách, viz Obr. 1.9 [4] [10].

G91 G28 Z0

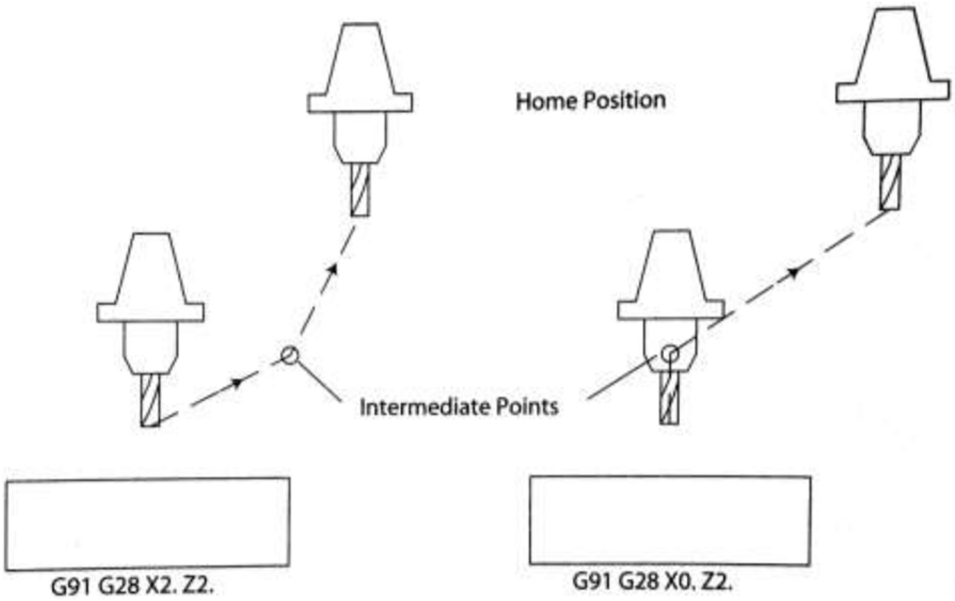
se tak přes nulový přírůstek v ose Z vrátí do strojního počátku v ose Z,

G91 G28 X0 Y0 Z0

se tak (přes nulový přírůstek ve všech osách) vrátí rovnou do strojního počátku,

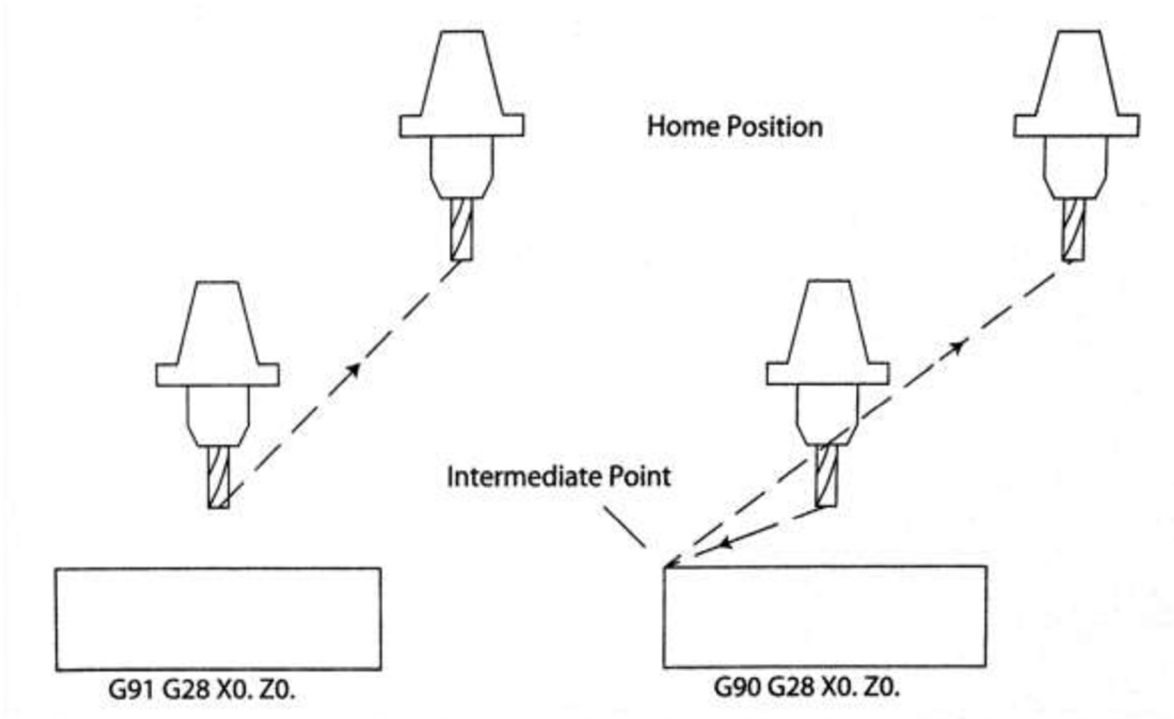
G91 G28 X0 Y0 Z5

se tak nejprve oddálí od obrobku v ose Z o 5 mm a následně se ve všech osách vrátí do strojního počátku [10].



Obr. 1.9 Návrat do strojního počátku přes přechodné body [4]

Kvůli přechodným pozicím se v literatuře silně nedoporučuje použití absolutního programování pro tento blok, které je velmi náchylné k fatální chybě, viz Obr. 1.10 [4] [10].



Obr. 1.10 Možná chyba při použití absolutního programování a G28 [4]

1.2.9 M-kódy

M03, M04 a M05 – zapnutí a vypnutí vřetene

Vřeteno se zapíná kódy M03 pro dopřednou rotací a M04 zpětným chodem. Při čelním pohledu na konec frézy udává M03 rotaci proti směru hodinových ručiček a M04 po jejich směru. Vřetenu je pro zapnutí také potřeba udělit otáčky adresou S [$^{-min}$] (*[RPM]*), kterou není nutno udávat s desetinnou čárkou. Na některých strojích je také potřeba zvolit rozsah převodů (*gear range*), kde M41 udává nižší řadu převodů a M42 vyšší. Pokud je potřeba do nástroj před zařazením do zásobníku nástrojů narotovat do referenční polohy, je možné použít kód M19 pro orientované zastavení vřetene [4].

M07, M08 a M09 – zapnutí a vypnutí chlazení

Chlazení se v obráběcích operacích používá na prodloužení životnosti nástroje, k odplavení třísek a vylepšení obrobeného povrchu. Obecně externí chlazení zapíná kód M08, pokud je CNC stroj vybaven, jde o povodňové chlazení (*flood*) a M07 spouští chlazení mlhou (*mist*). M09 chlazení vypíná a je dobré jej použít i při výměně nástroje, při pauze programu. Při těchto situacích a při ukončení programu kódem M30 vypínají moderní řídící jednotky chlazení obvykle automaticky [4].

M01 a M02 – přerušení programu

Pokud je program potřeba přerušit a to např. kvůli odstranění třísek, nebo pokud program potřebuje změnu upnutí, existují dvě možnosti pro okamžité přerušení. M00 je bezpodmínečné přerušení, které zastaví pohyb všech os do doby, než operátor nestiskne příslušné tlačítko pro start programu. Otáčení vřetene a další funkce nejsou ovlivněny. Příklad pro odstranění třísek [4]:

```
N150 G00 Z30.0 M09        (odjezd od obrobku, vypnutí chlazení)
N160 M05                  (zastavení vřetene)
N170 M00                  (zastavení programu)
N180 M03 S1500            (roztočení vřetene)
N190 G00 Z5 M08           (návrat k obrobku, zapnutí chlazení)
```

Pokud je nutné přerušit program jen za určitých okolností, existuje podmíněné přerušení s kódem M01. Za předpokladu že je na řídícím panelu zapnuté příslušné tlačítko pro podmíněné přerušení, program se na daném bloku přeruší, jinak běh programu pokračuje dále. Funkce může být výhodná pro možné přerušení v očekávané kritické chvíli, nebo na kontrolu nástroje před pokračováním [4].

/ - smazání bloku

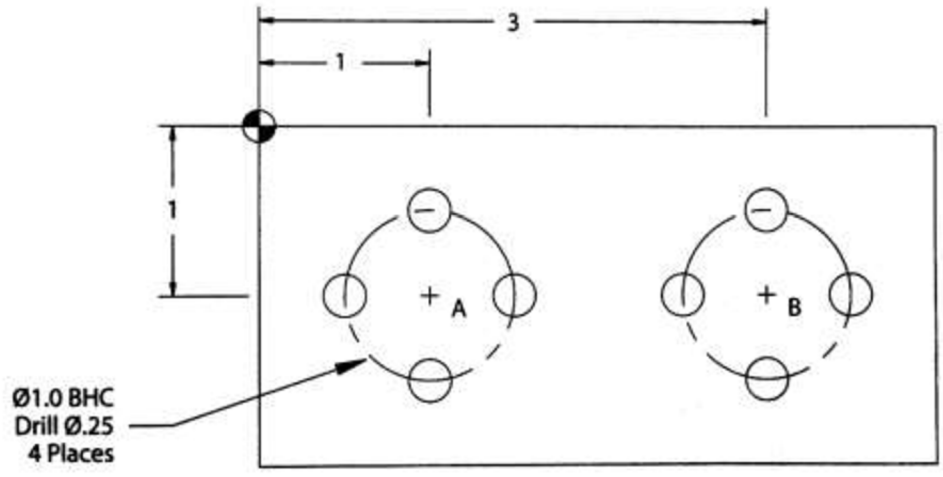
Pokud je potřeba zcela přeskočit několik bloků v programu umístí se před první slovo dopředné lomítko (/). Kód funguje podobně jako podmíněné přerušení programu v tom, že přeskočení bloku je závislé na nastavení konkrétního spínače, zde na spínači ovládání mazání bloků (*block delete switch on control*). Pokud je spínač vypnutý, bloky budou přečteny a zpracovány. Příklad na zpracování dvou podobných dílců jedním programem [4]:

```
N80 G81 X0 Y1. Z-1. R0.2 F5.0 (vrtané díry, první řada)
N90 X0 Y2.
N100 X0 Y3.
/N110 X1. Y1. (druhá řada, pouze pro součástku B)
/N120 X1. Y2.
/N130 X1. Y3.
N140 M05
```

1.2.10 Kódy pro pokročilé metody

G92 – specifikace nového počátku souřadného systému

Často je potřeba přesunout nebo změnit hodnoty kompenzací v osách pro stanovení nového počátku souřadného systému. To může být požadováno z několika důvodů, jako např. pro kompenzaci akumulovaných chyb v systému; není dostupných více souřadných

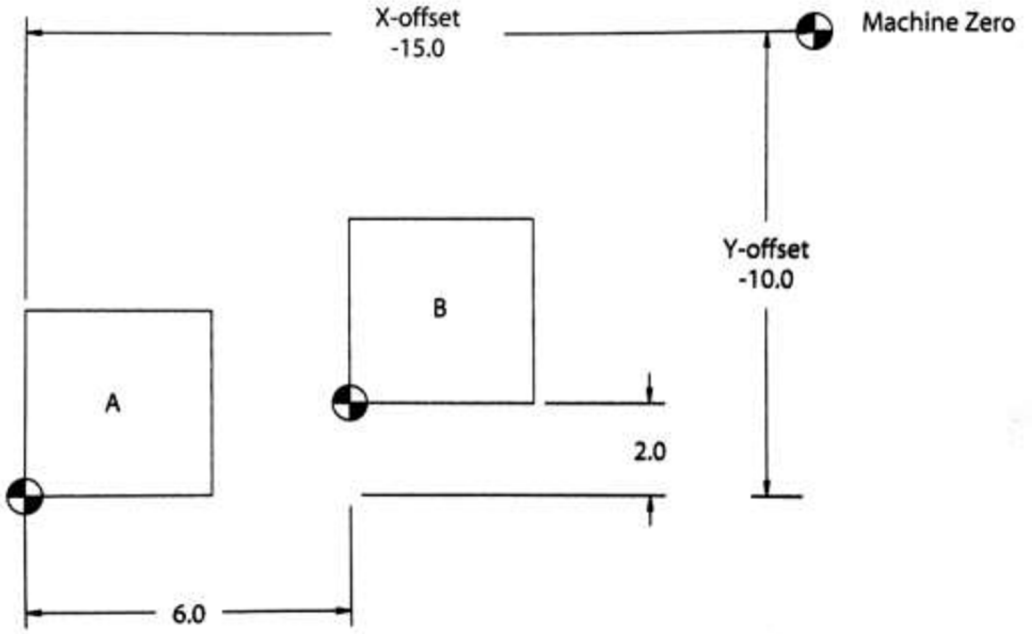


Obr. 1.11 Příklad využití G92 - posun souřadného systému pro vrtání opakujícího se vzoru [4]

systémů (G54, G55,…); pro automatizace programovacích úkonů jako tvorba opakujících se vzorů (*patterns*), jako např. na Obr. 1.11 [4].

Jsou rozšířené tři G-kódy pro reset souřadného systému: G92, G50 na mnoha soustruzích a G10 na některých novějších obráběcích centrech a všechny fungují podobně. G92 je však pro svůj vznik před kódy G54, G55, atd. podporován nejvíce řídícími jednotkami. Protože se korekce v osách mění permanentně, je potřeba myslet na jejich reset v závěru nebo úvodu programu, jak bylo nastíněno v kapitole o struktuře programu [4].

Kód G92 se používá nejčastěji pro změnu v osách X a Y a při jeho vyvolání změní řídící jednotka souřadný systém s ohledem na současnou polohu tak, že specifikované hodnoty X a Y budou udávat polohu nástroje v novém souřadném systému. To také znamená, že nástroj před voláním kódu musí být pro programátora ve známé poloze [4].



Obr. 1.12 Změna počátku souřadného systému [4]

Konfigurace na Obr. 1.12 je taková, že obrobek A vychází z počátku původního souřadného systému, který byl nastaven operátorem stroje a obrobek B reprezentuje místo, kde je požadován nový počátek. Pro tuto změnu lze postupovat několika ekvivalentními cestami. Jedna je posun nástroje do známé polohy, do počátku a poté se vyvolá kód G92. Zápis by mohl vypadat následovně:

```
N320 G00 X0.0 Y0.0
N330 G92 X-6.0 Y-2.0
```

tak by řídící jednotka přidala -6.0 do korekce v ose X a -2.0 do osy Y a posunula tak počátek souřadného systému do požadovaného bodu. V registru kompenzací by se tak změnila kompenzace v ose X z -15 na -9 a v ose Y z -10 na -8. A právě proto, že se pro kód G92 udává nová poloha nástroje v novém souřadném systému, zadává se hodnota X-6.0 a Y-2.0, místo X6.0 a Y2.0. Nástroj se tedy druhým blokem zápisu nepohne, ale řídící jednotka ho bude reprezentovat na poloze [-6;-2]. Ekvivalentním zápisem je například posun nástroje do polohy obrobku B a poté volání G92 [4]:

```
N320 G00 X6.0 Y2.0
N330 G92 X0.0 Y0.0
```

Podprogramy

Využití jednoho nebo více podprogramů je metoda, jak snadno omezit opakování v kódu. Jeho použití se dá snadno ilustrovat na příkladu, kdy je potřeba v obrobku vyvrtat několik identických děr v průběhu stejného technologického cyklu. Bylo by nevýhodné psát příslušný počet bloků kódu n-krát, bloků, které by byly až na souřadnice identické. Aby bylo zabráněno možným chybám plynoucí z opakování, zpřehlednil a zkrátil se kód, dá se použít podprogramu [1] [4].

Využití podprogramů je v některých situacích evidentně žádoucí. Nicméně pokud by příklad s vrtáním děr vyžadoval jakoukoliv alternaci mezi jednotlivými dírami, jsou podprogramy nepoužitelné a je třeba využít např. parametrického programování [1].

V G-kódu existují dvě velmi podobné metody pro podprogramy a to interní podprogram (*subroutine*), volaný příkazem M97 a externí podprogram (*subprogram*), volaný příkazem M98. Interní podprogram je vepsán v hlavním programu a je jeho součástí. Externí je uložen v jiném souboru než hlavní program [4].

Základ syntaxe pro kód M97 je následující:

```
N130 M97 P400                (zavolá podprogram na řádku 400)
...
...
N400                         (začátek podprogramu)
...
...
N450 M99                     (konec podprogramu)
```

Podprogram je zavolán na řádku s kódem M97, řídící jednotka pak skočí na řádek specifikovaný adresou P a pokračuje ve čtení od tohoto řádku do doby než podprogram ukončen. Když řídící jednotka narazí na slovo M99, podprogram je ukončen a chod programu pokračuje na řádku pod slovem M97, které daný podprogram volalo.

V předchozím příkladu by tedy pořadí řádků bylo N10 až N130, N400 až N450, N140, a dále [4].

Externí podprogram funguje podobně. Slovo M98 podprogram volá, adresa P specifikuje číslo podprogramu a adresa L specifikuje počet opakování podprogramu, pokud stačí jedno provedení, je možné adresu L zcela vynechat. Tedy blok:

```
N050 M98 P1000 L5
```

řekne řídící jednotce, aby provedl/spustil podprogram O1000 pětkrát. Po provedení podprogramu se řídící jednotka opět vrací do nadřazeného programu a pokračuje ve čtení pod řádkem s kódem M98, který podprogram volal [1] [4].

Adresa L se dá využít např. pro opakující se vzory (*patterns*), nebo řady (*arrays*). Aby bylo možné takovéto operace provést, je nutné použít některou z následujících metod:

- psát podprogram inkrementálně v G91 a polohovat v hlavním programu absolutně pomocí G90, nebo
- psát podprogram v absolutních souřadnicích G90 a v hlavním programu měnit souřadný systém (G92 nebo G54, G55,…) [4].

Program může obsahovat a volat několik interních i externích podprogramů [4].

1.3 Parametrické programování v Sinumeriku

Skupiny výrobků

Mnoho CNC uživatelů obrábí skupiny velmi podobných obrobků. Těmto skupiny se v angličtině používá výraz *part families*. V ideálním případě se mění pouze rozměr, příkladem jsou šrouby, vruty, matice, podložky, čepy, které se vyrábějí v několika velikostech. Výrobní parametry jsou nejlépe také podobné, v ideálním případě jsou výrobní parametry a použité nástroje v rodině výrobků shodné [1].

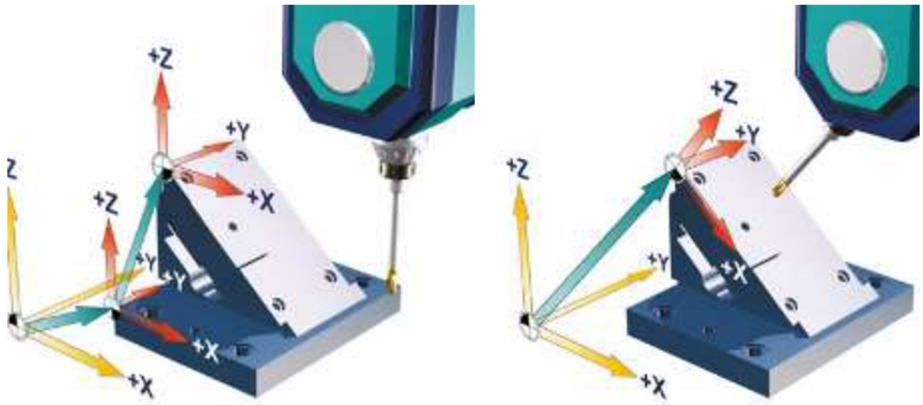
Pokud je v rodině 40 výrobků potřeba změnit jeden parametr, bylo by při 40 samostatných programech potřeba nejen provést tento počet změn, ale také 40 ověření správnosti při prvním chodu upravených programů. Může se jednat o změny nejen geometrií, ale také snah prodloužit životnost nástroje, zkrátit výrobní čas, nebo dosáhnutí jakéhokoliv vylepšení obráběcího procesu. Úprava 40 samostatných programů by značně snížila flexibilitu, dobu uvedení na trh, možnost zkoušení nových postupů, apod. [1].

Geometrické prvky

G-kód nabízí základní aritmetické funkce a tvary, které se dají popsat matematicky tak lze pomocí G-kódu vyrobit. Tomuto se věnuje praktická část práce [1].

1.3.1 Transformace

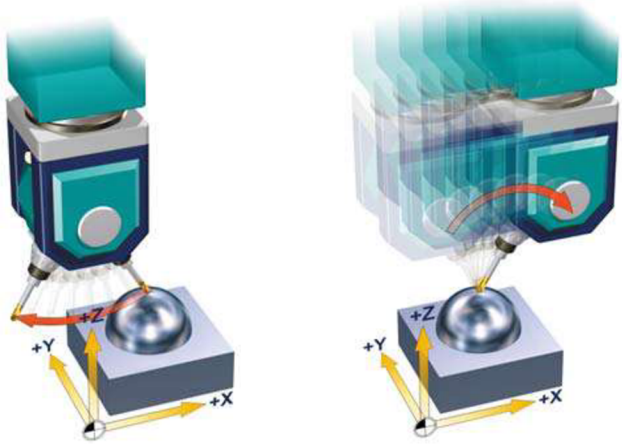
Pro ovládání os a natočení nástroje vůči obrobku existují tři základní skupiny prvků. Swivel (Cycle800), Frames a TRAORI [9] [2].



Obr. 1.13 Znázornění frames vlevo a cyklu 800 vpravo [11].

Cyklus 800 natočí rotační osy jednou a v rámci obrábění se pohybují lineární osy XYZ, jde tedy o statickou transformaci (viz Obr. 1.13) [11].

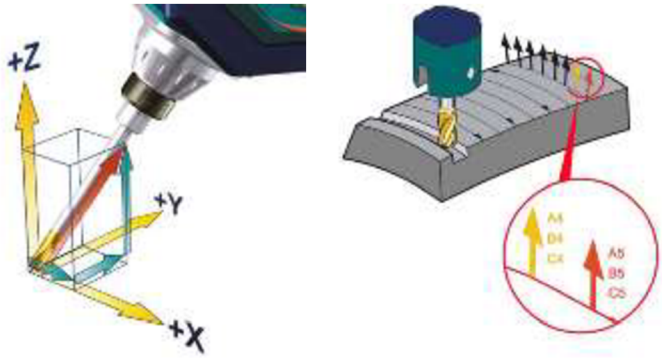
TRAORI je proces dynamický. Rotační i lineární osy mohou být používány simultánně. Nástroj tak může být neustále orientován při obrábění k obrobku. Délka nástroje a kompenzace jsou při pohybu rotačních os brány v potaz. Orientace probíhá ke špičce nástroje, která zůstává při orientaci stacionární [11].



Obr. 1.14 Rozdíl ve změně natočení nástroje.
Bez TRAORI vlevo a s TRAORI vpravo [11].

Je možné, že funkce vynuluje pracovní ofset (př. G54) a kompenzaci nástroje (D1), proto je doporučováno tyto příkazy po TRAORI znovu deklarovat. Samotná funkce se vypíná příkazem TRAFOOF [11].

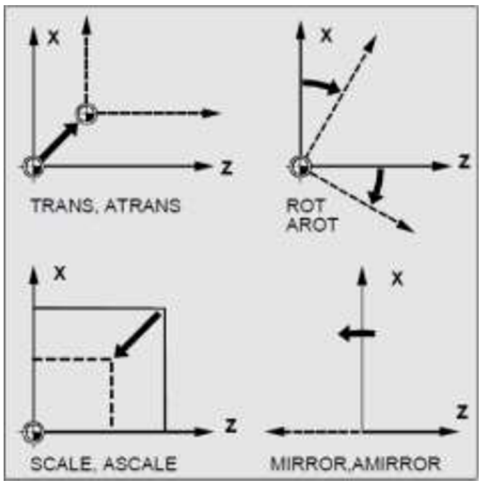
Samotná orientace nástroje může být zadána několika způsoby. Jednak přímým zadáním natočení os A, B nebo C. Doporučené je ale programování směrovým vektorem A3, B3, C3, směřujícím od špičky nástroje k upínači. Další možností je např. určením normály k povrchu na začátku bloku A4, B4, C4, nebo na konci bloku A5, B5, C5 a deklarací natočení nástroje vůči normále pomocí LEAD a TILT [11].



Obr. 1.15 Orientace nástroje. Vlevo pomocí směrového vektoru, vpravo pomocí normál k povrchu [11].

Frames ovlivňují pouze souřadný systém a pouze ten je modifikován příkazy jako TRANS, nebo ASCALE. Cyklus 800 bere v úvahu kinematiku stroje, v rámci frames na ně musí myslet programátor (viz Obr. 1.13). Možné frames jsou: Základní frame (základní ofset, G500), nastavitelný frame (G54, G55,…) a programovatelné frames (TRANS, ROT, …) [11].

V programu je možné transformovat souřadný systém programovatelnými frames posunutím, rotací, změnou velikosti, nebo zrcadlením. Pro posunutí je adresa TRANS a ATRANS, pro rotaci ROT a AROT, změnu velikosti SCALE a ASCALE, zrcadlení MIRROR a AMIRROR. Adresy začínající písmenem A jsou aditivní transformace a adresy bez nich absolutní transformace (viz Obr. 1.16) [9].



Obr. 1.16 Transformace souřadného systému [9].

1.3.2 Proměnné

Parametrické programování dostalo svůj název právě pro použití parametrů, tedy jistých proměnných a právě ty umožňují tvořit flexibilní programy, společně s matematickými funkcemi a řídícími strukturami. V Sinumeriku je možné použít několik typů proměnných. Dělí se na [2]:

- Systémové proměnné,
  - Proměnné předběžného zpracování,
  - Proměnné hlavního zpracování (synchronní akce),
- Uživatelské proměnné,
  - Předdefinované uživatelské proměnné,
    - Početní parametr R,
    - Linkové proměnné (v ethernetu),
  - Uživatelem definované proměnné,
    - LUD - lokální uživatelské proměnné,
    - PUD - programově globální uživatelské proměnné,
    - GUD - globální uživatelské proměnné.

Systémové proměnné jsou v systému Sinumerik definovány a je jim přiřazen význam, jako např. $P_F, určující naposledy zadaný feed, nebo $P_GG[n] nabývající hodnoty konkrétního posledně zadaného G-kódu ze skupiny n [2].

Předdefinované uživatelské proměnné jsou zastoupeny především parametry R s datovým typem REAL, které nemají přiřazený význam a uživatel je může používat bez deklarace. Dají se používat buď zápisem např. R37, nebo také R[37]. Tedy přiřazení hodnoty 12 uživatelské proměnné 37 vypadá takto: R37=12, nebo R[37]=12. Parametrů je <0;$MN_MM_NUM_R_PARAM > [2].

Uživatelem definované proměnné mohou mít uživatelem definovaný název i datový typ a podle definice je možné určovat jejich dostupnost. Je nutno zadat na začátku kódu (DEF), před ostatními věcmi, jako je např. geometrie polotovaru. Výjimku tvoří PROC (viz dále), který patří vždy na první řádek. GUD se definují v datovém modulu a jejich hodnota je dostupná všem programům a to i po restartu stroje [2].

1.3.3 Matematické funkce a řídící struktury

V programu je možné zadávat základní matematické operace a porovnávat hodnoty, především proměnných. Slouží k tomu běžné znaky na klávesnici (+-*/<>=). V programech je možné měnit sled čtení bloků pomocí nepodmíněných funkcí GOTO, nebo nechat vyhodnocovat podmínky pomocí funkcí typických ze všech oblastí programování, IF, WHILE, FOR, CASE [2].

1.3.4 Podprogramy

Úvod do podprogramů byl uveden v kapitole 1.2.10 pro Fanuc. Následuje upřesnění pro Sinumerik.

Podprogramy poskytují zvýšení čitelnosti a přehlednosti programu. Pravidla pro názvy podprogramů v Sinumeriku jsou: první dva znaky musí být písmena; další znaky mohou být [a-z], [0-9] a podtržítko "_", až do celkové délky 31 znaků; majuskule a minuskule systém nerozlišuje [2].

Hlavní programy mají koncovku .MPF a vedlejší .SPF, uvnitř řídicího systému jim je přidělena předpona _N_ a přípona _MPF resp. _SPF. Hlavní a vedlejší program mohou mít stejný název, ale je nutné je v tomto případě volat i se zmíněnou příponou (př. _N_PRAVE_KRIDLO_SPF), protože hlavní program jde volat jako vedlejší a naopak. Systém první hledá, zda existuje soubor s tímto názvem s koncovkou _MPF, poté _SPF. Kvůli možné lidské chybě tak je doporučována jednoznačnost názvů [2].

Stejně jako u řídící struktury je možné 16 programových úrovní (viz Obr. XX). Což při odečtení hlavního programu dává možnost volat cykly Sinumeriku, které zabírají 3 úrovně, maximálně z 11 vnořeného podprogramu [2].

Pokud se podprogram nachází v jiném adresáři, je vhodné tuto skutečnost zadat a to sice tak, že na rozdíl od adres v systému Windows se pro složky používá předpona _N_ a přípona _DIR, tedy např.: /_N_PODPROGRAMY_DIR/_N_PRAVE_KRIDLO_SPF [2].

Pokud je podprogramu potřeba předat určité parametry, vepíše se na první řádek podprogramu slovo PROC, název programu a do závorky všechny parametry, které podprogram očekává, že mu budou předány. Pokud by mu nějaký parametr předán nebyl, jeho hodnota standardně nabyde nulu. Výjimku tvoří parametry typu Call-by-reference (viz dále) a datový typ AXIS, které nelze vynechat [2].

Parametrům, které jsou definovány ve volajícím programu, se říká skutečné a parametrům, kterým je jejich hodnota ve volaném podprogramu přidělena se říká formální. Příkladem je následující hlavní program [2]:

```
N60 DEF INT DELKA; Definice proměnné

N65 DELKA=5.0 ; Přiřazení hodnoty proměnné

N70 PROFIL (,TRUE,2.14,DELKA) ; Volání podprogramu se
skutečnými parametry: 2.14 a DELKA

...

N100 M30
```

A vedlejší program:

```
PROC PROFIL (REAL C, BOOL H, REAL X, INT Y, REAL Z) ;
```

Formální parametry by nabyly hodnotu: C=0, protože nebyl předán, H=TRUE, X=2.14, Y=DELKA=5.0 , Z=0, protože také nebyl zadán

```
;

N30 X1=X Y1=Y ; Najetí osou X1 na pozici 2.14 a osou Y1 na 5.0

...
```

```
N100 RET ;návrat z podprogramu do hlavního programu
```

Zda byl parametr předán, nebo vypuštěn lze zjistit proměnnou $P_SUBPAR [n], kde n označuje pořadí zkoumaného parametru a nabývá BOOL hodnot. Tedy pro minulý příklad programu by výsledkem logické operace $P_SUBPAR [4]==FALSE bylo FALSE, jelikož 4. hodnota byla zadána, takže je porovnáváno TRUE==FALSE. Testování lze použít například do řídících struktur (IF) [2].

Co se týče proměnných LUD, existují dva typy předávání parametrů, Call-by-Value a Call-by-Reference. Rozdíl je v ovlivnění volaného parametru pomocí proměnné podprogramem (ke globálním proměnným má podprogram přístup). Při použití Call-by-Value, které bylo použito v příkladu výše, se předávají podprogramu pouze hodnoty. Při použití Call-by-Reference se odkazuje na skutečný parametr, podprogram má tak oprávnění jej měnit i pro hlavní program. Rozdíl v syntaxi je pouze v použití, nebo vynechání slova VAR před typem předávané proměnné v prvním řádku podprogramu. Lze se také odvolávat na pole [2].

```
PROC <název programu> (VAR <typ parametru> <název parametru>, ...)
```

```
PROC <název programu> (VAR <typ pole> <název pole> [<m>,<n>,<o>],
...)
```

Pokud je potřeba uložit modální G-funkce ve volajícím hlavním programu, napíše se ve volaném podprogramu na konec prvního řádku slovo SAVE. Po návratu se tak hlavní program bude chovat nezávisle na změnách v podprogramu (např. při přepnutí G90/91) [2].

Rozdíl v ukončení pomocí M17/30 a RET spočívá v ukončení resp. zachování řízení režimu po dráze (G641 až G645). RET tak pochopitelně nelze kombinovat se SAVE [2].

Další výhodu má RET v možnostech parametrů, kdy se lze vracet na jiné místo, a to dokonce přes více úrovní [2].

Pokud je potřeba podprogram vícekrát opakovat, použije se při volání adresa P a číslo udávající počet opakování [2].

Pokud je program potřeba opakovat na jiných místech, modální volání spustí a vykoná podprogram na každém následujícím bloku s dráhovým pohybem. V činnost se uvádí předpisem MCALL a názvem podprogramu a jeho funkce se ukončí stejným slovem, ale bez atributu [2].

Pokud je potřeba vykonat jen část podprogramu, vyvolá se příkazem CALL, název programu, odkaz na návěstí počátku zpracování, TO, odkaz na návěstí konce zpracování [2].

1.3.5 Hladké translační pohyby

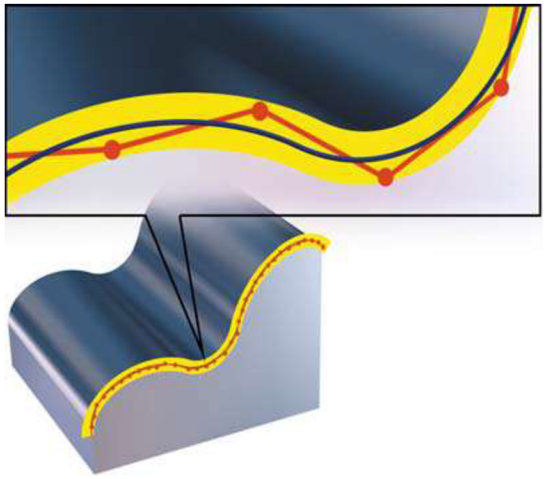
CAM systémy obvykle umí exportovat vytvořenou geometrii pouze jako lineární aproximace ve stanovených tolerancích. Tyto krátké lineární entity, G1 X Y Z, však vedou ke ztrátě přesnosti, která může být dokonce viditelná. Mezi přechody lineární interpolace dochází ke skokové změně rychlostí pohybu os, což může mít za následek rezonanci mezi obráběnými elementy a tak zkosený (*bevelled*) povrch, nebo povrch ovlivněný vibracemi. Pro přesné, nebo vysokorychlostní obrábění tak vzniká požadavek na hladké pohyby [11].

Pro hladké translační pohyby má Sinumerik dvě možnosti, první je využití interpolačních spline křivek ASPLINE, BSPLINE, CSPLINE, nebo polynomiální funkce POLY; nebo vyhladí bloky zadané lineárním pohybem pomocí kompresoru bloků [2] [11].

Kompresor

Sinumerik má funkci kompresoru (COMPCAD), která v rámci zadané tolerance umí proložit sekvencí G1 příkazů spline křivku, která je následně přímo zpracována řídící jednotkou (viz Obr. 1.17). Kompresor vytváří hladké dráhy s konstantním zakřivením. To má za následek plynulé charakteristiky rychlosti a zrychlení. Díky tomu může stroj operovat ve vyšších rychlostech a zvýšit tak svojí produktivitu [11].

Vyhlazení nesouvislých bloků může být také vyhlazeno funkcí G645, která vloží mezi bloky hladký geometrický element. Jejich tolerance je nastavitelná [11].

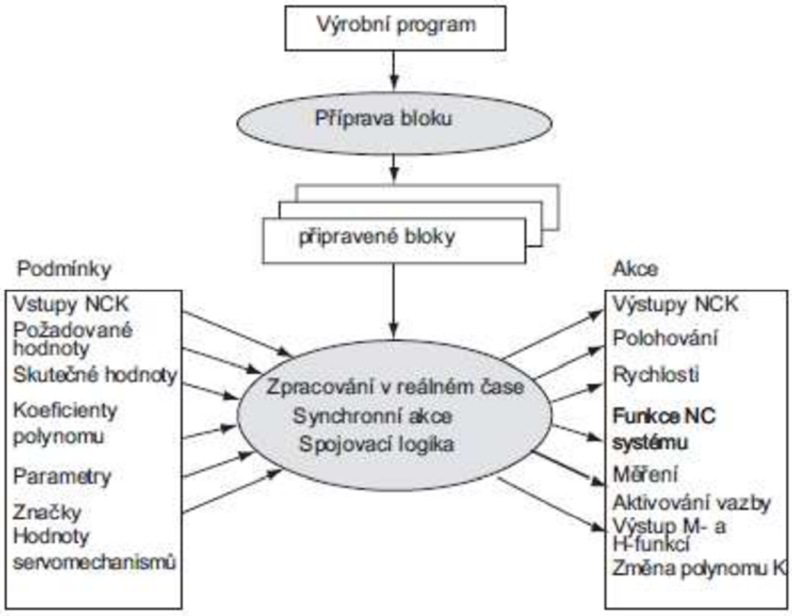


Obr. 1.17 Proložení spline křivky lineárními úseky kompresorem [11].

Pro snížení ryvu a vyhlazení pohybu složí i jiné funkce, jako např. celá série G64n, SOFT, DYNFINISH, před-načítání do bufferu FIFOCTRL, kontrolu feedu FFWON, které jsou zároveň obsaženy v cyklu 832 při volbě dokončovací operace. [2].

1.2.1 Synchronní akce ID … DO ..

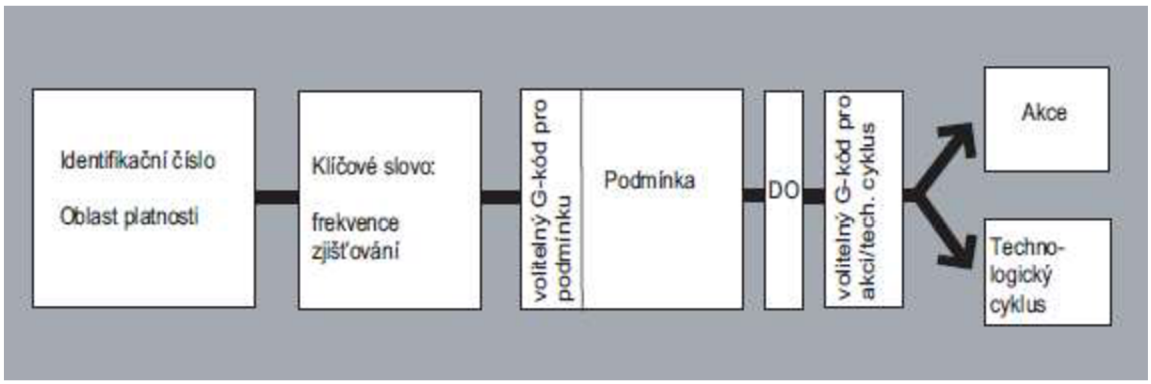
Synchronní akce jsou funkce systému, která umožňuje provádět určité akce zároveň se zpracováním bloků programu. Akce jsou odezvou na událost v reálném čase (Obr. 1.18). Kdy se příslušná operace provede, je řízeno podmínkami, je také předepsáno jak často se má podmínka testovat [2].



Obr. 1.18 Znázornění fungování synchronních akcí [2].

Synchronní akce neprávem nepatří mezi často používané funkce systému. Mají mnoho využití, jako pro příkazy citlivé na dobu zpracování (výměna nástroje), pokud je vyžadována okamžitá reakce na událost (kritické stavy systému, kontrola kolizí) apod. [2].

Zapisuje se na vlastní blok a je vyhodnocována od dalšího bloku s funkcemi stroje. Syntaxe má až pět prvků viz Obr. 1.12 [2].



Obr. 1.19 Syntaxe synchronních funkcí [2].

1.2.1 Vlastní program pro oddělování třísky

Zajímavou funkcí systému Sinumerik je kromě již hotových obráběcích cyklů možnost sestavit si uživatelský obráběcí cyklus. Příkazem CONTPRON se uloží do tabulky souřadnice konturových prvků, druh a směr opracování a systém si sám vyhodnotí, zda nedojde někde k podříznutí, které případně vypíše jako chybová hlášení. Bohužel tato funkce umožňuje použití pouze G-skupiny 1, tedy lineárních a kruhových interpolací, spline ani polynomy nelze body proložit a je tak pro tuto práci použitelná jen z hlediska využití s kompresí bloků [2].

1.3.6 Operace se STRING

S jednotlivými datovými typy (REAL, INT, BOOL, STRING,..) lze provádět mnoho operací a obvykle je lze mezi sebou za různých podmínek převádět [2].

Jedním z nejčastěji využívaných převodů jsou operace s řetězci (STRING). Lze do nich převádět jiné typy (AXSTRING), je převádět do jiných typů (NUMBER, ISNUMBER, AXNAME), spojit více řetězců dohromady, tzv. zřetězení (<<), převést string na majuskule a minuskule (TOLOWER, TOUPPER), zjistit, jak je STRING dlouhý (STERLEN) a několik dalších operací. Nula v řetězci značí jeho konec, tedy dosazením nuly doprostřed řetězce maže jeho konec [2].

Operace zřetězení (<<) nejen připojí jeden řetězec k druhému, ale automaticky umí převést a připojit jako string i typy BOOL CHAR, INT a REAL. Následující kód tak vypíše pozici osy X [2]:

```
MSG("Position:"<<$AA_IM[X])
```

1.4 Regulární výrazy

Regulární výrazy (regular expression), také regex, umožňují snadno reprezentovat hledaný výraz v řetězci. Pokud má exportovaná geometrie několik tisíc řádků, nelze kód pochopitelně procházet řádek po řádku a ručně provádět editaci. Regex, funkční napříč programy a programovacími jazyky je jediným řešením, pokud není k dispozici dedikovaný software [12].

Zápis je snadný a výrazů není mnoho, jejich kombinací však vznikají poměrně komplexní vyhledávací funkce. Tečka zde například reprezentuje jakýkoliv znak, "[0-9]" jakékoliv číslo, "\n" zalomení řádku. Existují zde např. i zástupné symboly pro počet opakování [12].

Pokud je tak v možnosti 'najít a nahradit' vepsán do políčka 'najít' výraz "A" a do pole 'nahradit ' výraz "A3=", přepíše se každé 'A' na 'A3='. Pokud by byl hledaný výraz "\n" a nahrazující " ;\nX=", bude tak na každý konec řádku připsána mezera a středník, na začátek každého dalšího řádku 'X=' [12].

Kolikrát se může předchozí znak opakovat, určuje kvantifikátor. Otazník definuje, že se daný výraz musí opakovat maximálně jednou, hvězdička, že se daný výraz může, nebo nemusí vyskytovat, znaménko plus vyhledá výrazy opakující se alespoň jedenkrát. Pokud je potřeba předpis přesného počtu opakování, zadává se do složených závorek jako {min,max},

pokud není uvedeno n, hledá předpis minimální počet opakování, pokud je uvedeno pouze n, hledaný výraz musí být obsažen právě n-krát [12].

Pokud je potřeba hledat jeden z deklarovaných znaků, uzavře se do hranatých závorek a hodnoty lze zadávat jako posloupnost, např. "[3-8]", "[x-z]". Pro nejpoužívanější kombinace slouží výrazy: "\d", pro všechna čísla, "\w" pro všechny znaky slova (tj. 0-9, a-z, A-Z, podtržítko), "\s" pro mezeru, tabulátor a zalomení. Pro výběr všeho, kromě "\d", "\w" a "\s" se použije "\D", "\W" a "\S". Pro vyhledávání sekvence znaků se používají kulaté závorky [12].

Speciální znaky, využívané pro regulární výrazy, jako je +, *, ?, ., (, lze vyhledávat s předponou lomítka. Tedy např. \+, \^, \\ [12].

Důležitou funkcí pro chod (nejen) makra v Notepadu++ v praktické části je funkce, která umožňuje v části 'nahradit' použít výraz $n, který je nahrazen n-tou závorkou v hledaném výrazu. Potom při hledání "(-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+)\n", nahrazovaný regulárním výrazem "X$1 Y$2 Z$3\n" přidá před souřadnice adresy X, Y a Z.

1.5 Systémy modelování v 3D

Konvenční reprezentace pomocí výkresové dokumentace, případně pomocí diagramů má řadu silných stránek a slouží inženýrům dekády. Mají ale také své limity. První je nutnost znalosti čtení a tvorby výkresové dokumentace, druhou představuje možnost konfliktních, nebo chybných modelů. Třetím limitem, důležitým pro tuto práci, je neschopnost vypovídajícího záznamu velmi komplexních geometrií, jako jsou např. zakřivené povrchy na automobilech, nebo letadlech [7].

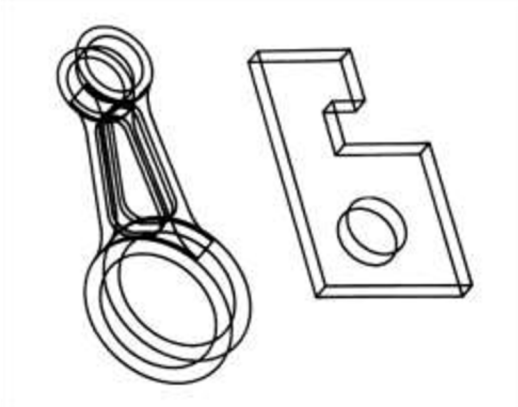
Jako důsledek těchto limitů byly vyvinuty různé metody reprezentací geometrií, které využívají systémy, které nespoléhají na projekci do rovinného prostoru, ale na jediné reprezentaci v prostoru třídimenzionálním (3D). Nejen že se tak snižuje riziko spojené s více pohledy, ale především lze takto reprezentovanou geometrii podrobit dalším analýzám [7].

Systémy pro reprezentaci geometrie ve 3D využívají skupiny přímek a jiných křivek, nebo ploch, nebo plných tělech v prostoru [7].

1.5.1 Wire-frame geometrie

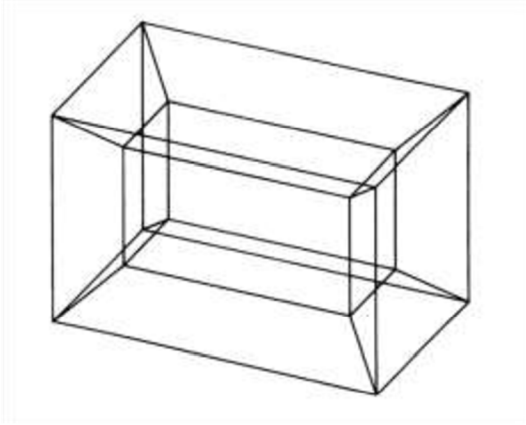
Wire-frame, nebo také drátový, či drátěný model, je výpočetně nejpřímočařejším systémem. Jak již název vypovídá, geometrie je definována sérií přímek a křivek, které reprezentují hrany objektu. Jde o rozšíření technik pro navrhování ve 2D o jednu dimenzi [7].

Systém je vhodný především pro předběžný nákres; pro řešení některých geometrických záležitostí; nebo pokud je vyžadována dynamická manipulace s modelem (jako animace pohybu mechanizmu) a to především pro jednoduchost zadání i výpočetní nenáročnosti. Stejně tak se systém využívá pro objekty definované projekcí profilu po normále, jako jsou výstřižky z plechu, nebo definované rotací kolem osy, které nevyžadují sofistikovanější systémy. Takovýmto objektům se říká dvou-a-půl-dimenzionální (2,5D), příklad na Obr. 1.20 [7].



Obr. 1.20 Drátový model 2,5D součástí [7]

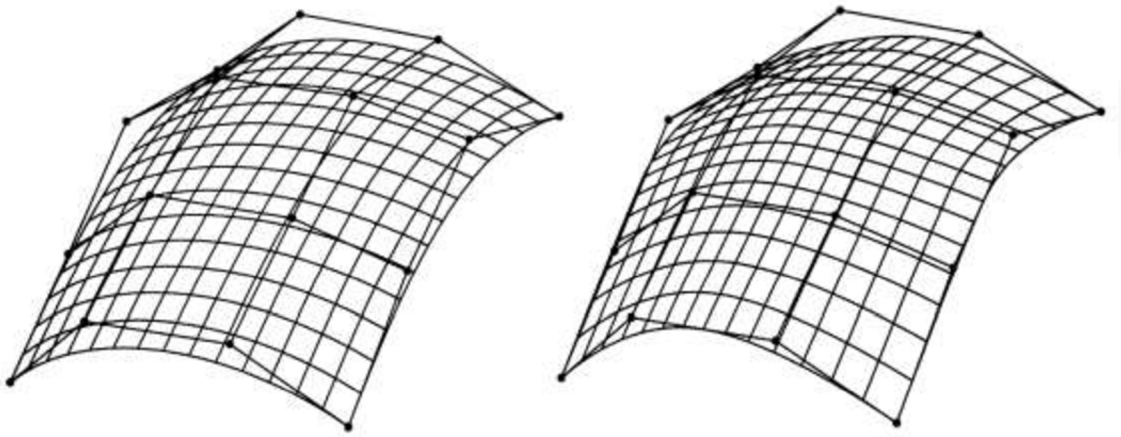
Nedostatkem je dvojznačnost v reprezentaci, jako je vidět na Obr. 1.21, ze kterého není patrné, kudy prochází díra. Stejně tak není z obrázku patrné, který roh je nejblíže divákovi (chyba, která se v zobrazovacích programech často řeší různou tloušťkou čar dle hloubky). Stejně tak má reprezentace omezené možnosti výpočtu mechanických vlastností a geometrických průsečíků [7].



Obr. 1.21 Nejednoznačnost drátových modelů [7]

1.5.2 Systém reprezentace pomocí ploch

Mnohé z dvojznačností drátové reprezentace jsou překonávány použitím druhým ze tří systémů, modelováním pomocí ploch. Model je reprezentován některými, nebo všemi plochami na součásti [7].



Obr. 1.22 Příklad ploch definovaných body. Reprezentace sítí, ve skutečnosti kontinuální. Vlevo Bézierova plocha, vpravo kvadratická B-spline plocha [7].

Nejjednodušším typem plochy je rovina, která může být definována paralelními přímkami, přímkou a bodem, nebo třemi body. Definice dalších typů ploch obvykle spadají do jedné z následujících tří kategorií [7]:

1. Plocha je připevněna k datovým bodům, které se nazývají ‚kontrolní body' a plocha je generována tak, aby procházela přes ně, nebo je interpolovala. Příklad na Obr. 1.22.
2. Plocha je založena na křivkách. Pro snazší představu, plochy se tvoří jako „kůže na kostře z drátů". Příkladem je válcová a přímková plocha; plocha vzniklá rotací křivky kolem osy; nebo spline křivky; nebo tvarované povrchy (*sculptured surfaces*) definované mřížkou generujících křivek, jako jsou Coonsovy plochy.
3. Plocha se tvoří jako interpolace mezi existujícími plochami, např. přechody. Tato kategorie může obsahovat např. plochy úkosů (*chamfer*) a vnitřní zaoblení (*fillet*).

Výše uvedené kategorie popisují, jak lze plochu definovat. Matematický základ pro reprezentaci plochy a pro metodu uložení systémem však může být stejný i pro plochy definované jinak. Dokonce je jednotná reprezentace trend v komerčních modelářích. Ty také často povolují tzv. kompozitní plochy, sestávající z několika elementárních ploch, spojenými dohromady tak, že pro uživatele působí jako jedna plocha [7].

1.5.3 Modelování plných těles

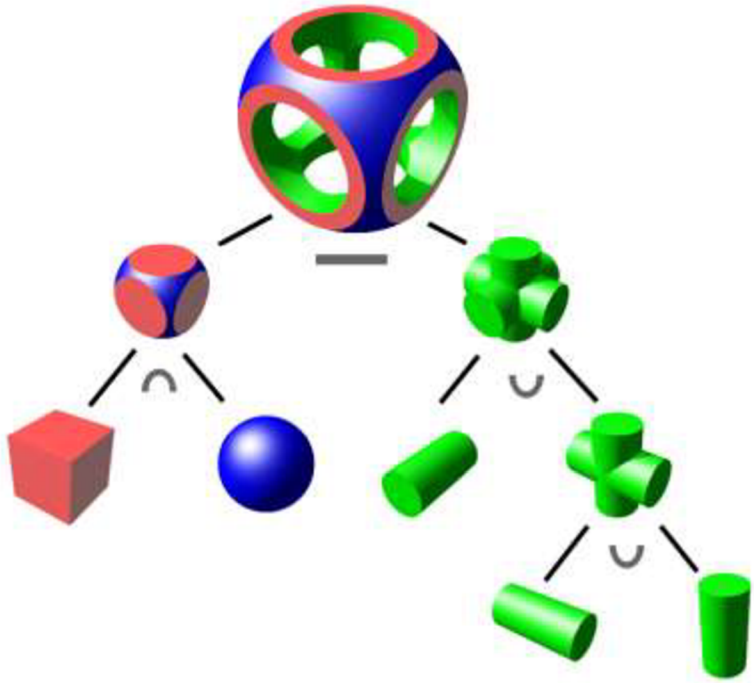
Předchozí dva systémy modelují objekt pouze částečně a plné těleso (*solid*) z nich musí být odvozeno. Pro některé aplikace je to dostačující, ale zvyšující se využití počítačů na analýzu nebo generaci výrobních informací znamená požadavek na co nejkomplexnější reprezentaci [7].

Vzniklo několik systémů na modelování plných těles, z nichž dva jsou částečně úspěšné. Prvním je konstruktivní přístup s nejpoužívanější variantou Constructive solid geometry (také zkráceně CSG nebo C-rep), která doznala největšího úspěchu v začátcích CAD programů a druhým je Boundary representation (zkráceně B-rep), která dominuje v současných aplikacích [7].

Constructive solid geometry

Tato metoda využívá vektorového modelování geometrických těles. Tyto tělesa se konstruují kombinacemi z primitivních geometrických těles (*simple solid primitives*) (koule, kvádr, válec, kužel, toroid, klín) operacemi sjednocení, průniku a rozdílu (*union, intersection, difference*). CSG objekty tak nejsou unikátní reprezentací tělesa. Grafické znázornění na Obr. 1.23 [7].

CSG modely mají výhodu ve své kompaktnosti a záruce, že jednoznačně modelují validní plná tělesa. Modely jsou ale ukládány v nevyhodnocené podobě, kdy hrany a plochy, které mají vzniknout kombinací primitiv, se dopočítávají kdykoliv je potřeba, např. při generaci zobrazení modelu. Zásadní nevýhodou tak je nutnost velkých výpočtů při práci s komplexnějšími tělesy a náročné modelaci. Příkladem ilustrující nevýhody s narůstající komplexností je kvádr, od kterého je odečten menší kvádr tak, že výsledný objekt tvoří jakousi jednoduchou hranatou misku, nebo vanu. Objekt by vyžadoval 2 primitiva a 1 Booleovu operaci. Pokud by se však měl objekt vyrábět odléváním, potřeboval by jen kvůli technologickým úkosům 22 primitiv a 21 Booleových operací [7].



Obr. 1.23 Příklad tvorby objektů pomocí constructive solid geometry [26].

Boundary representation

Systémy reprezentace pomocí ploch (kapitola 1.5.2) neobsahují žádnou informaci o spojení mezi plochami, ani o tom, která část tělesa je plné těleso. Pokud je informace o napojení ploch, zde stěn, dodána, stejně jako která strana stěny je součást tělesa, vznikne druhá z hlavních modelovacích metod plných těles, tzv. boundary representation (reprezentace hranicemi). Reálné systémy jdou dále a zahrnují metody na kontrolu topologické a geometrické konzistentnosti těles, tedy zda něco nechybí, nebo nepřebývá a tvořené těleso je „rozumné“. Moderní modelářské programy dovolují i použití NURBS křivek a ploch [7].

Na rozdíl od CSG B-rep skladuje informace o stěnách a hranách objektu ve vyhodnocené formě a tak lze některé hodnoty extrahovat přímo z datové struktury. Nevýhodou jsou poměrně velké datové soubory [7].

CGS modely jsou robustnější a poskytují rychlejší vyhodnocení testu příslušnosti. B-rep nabízí rychlejší generaci zobrazení a větší flexibilitu v modelování. Pro nejednoznačnost superiority kombinovaly starší programy oba přístupy, ale v moderních převážil B-rep. Jeden z důvodů je snadná možnost převodu z CSG na B-rep, naopak to však neplatí, dalším může být narůstající kombinace modelování plných těles a ploch, které použití B-rep usnadňuje [7].

1.6 Metody pro modelování geometrií

Počítačová reprezentace geometrie byla dosud hodnocena pouze kvalitativně. V této podkapitole bude popsán matematický aparát pro křivky a plochy. Plná tělesa jsou podstatná pro obrábění pouze do bodu exportu dat z CAD/CAM systému a budou zde tedy vynechána. Matematicky nejpřímočařejší entity jsou křivky, zároveň vyjadřují dráhu nástroje a budou tak rozebrány nejpodrobněji. Geometrie ploch je obvykle pouze rozšíření konceptů křivek o dimenzi. Vzhledem k jisté návaznosti složitějších a používanějších konceptů na jednodušší (Hermite -> Béziere -> B-spline -> NURBS), budou popsány i méně používané reprezentace [7].

1.6.1 Reprezentace křivek

Obecné vztahy pro explicitní a implicitní zadání přímky:

$$y = kx + q \tag{1.1}$$

$$ax + by + c = 0 \tag{1.2}$$

Obvyklý způsob zadávání křivek je pomocí explicitních nebo implicitních rovnic. Příkladem jsou rovnice pro přímku (1.1) a (1.2), kde první je zadaná explicitně a druhá implicitně. Tento typ zadávání však není vhodný pro počítačové použití, tedy ani pro CAD, nebo CNC řídící jednotky. V případě s přímkou nastává problém mj. k numerickým chybám při směrnici blížící se k nekonečnu, ale obecně jde o nevhodný zápis, protože [7]:

- reprezentují neomezenou geometrii, např. přímku, místo úsečky,
- rovnice se liší v závislosti na souřadném systému,
- křivky mohou nabývat více hodnot pro jednu proměnnou, např. kuželosečky,
- v CAD systémech je obvykle nutné vyhodnotit uspořádanou sekvenci bodů na křivce a např. u implicitního zadání paraboly postupné dosazování bodů X přinese velmi nerovnoměrné rozložení bodů na křivce a některé hodnoty nebudou mít ani řešení,
- rovnice vzniklé i jednoduchým průnikem primitivních těles jsou neúnosně komplexní a je jednoduší je interpolovat sérií bodů.

Naopak vlastnosti, které bývají požadovány jsou [13]:

- stálost ke změnám afinními transformacemi a k projekcím,
- křivka leží v konvexní obálce svých řídících bodů,
- změna řídícího bodu se projeví pouze na malé části křivky,
- křivka může vycházet a končit v prvním, resp. posledním řídícím bodě.

1.6.2 Parametrická reprezentace geometrie

Řešením problémů analytických vyjádření je např. parametrické vyjádření. To vyjadřuje vztahy mezi *X, Y* a případně *Z* souřadnicemi bodů na křivce, ploše nebo plném tělese a to ne mezi sebou navzájem, ale v závislosti na jedné, nebo více nezávislých proměnných zvaných parametry. Pro křivky se obvykle používá jeden parametr, pro plochy dva a pro tělesa tři [7].

Parametrické vyjádření křivky

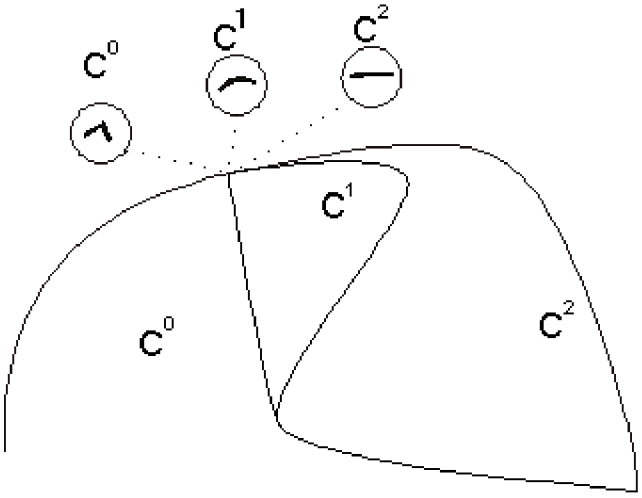
Rovinná křivka daná parametricky je množina bodů v rovině, které jsou dány rovnicemi

$$x = \varphi(t),$$
$$y = \omega(t), \qquad t \in I,$$

kde t je parametr. Předpokladem je spojitost funkcí φ(t) a ω(t) a jejich alespoň po částech spojité první derivace dle parametru t na daném intervalu. Interval <α;β> určuje počáteční bod [φ(α);ω(α)] a koncový bod [φ(β);ω(β)] [14].

Jestliže je počáteční a koncový bod identický, je křivka uzavřená, jsou-li různé, je otevřená. Pokud jsou první derivace funkcí dle parametru spojité na intervalu, pak je křivka hladká, tedy lze sestrojit (s možnou výjimkou počátečního a koncového bodu) tečnu v každém jejím bodě. Pokud je spojitá i v derivaci stupně n, je spojitost křivky (C^n). Rozlišuje se geometrická spojitost (G^n) a analytická spojitost (C^n). Křivka může být složená z hladkých částí navazujících na sebe a je tak po částech hladká. Křivka neprotínající se na intervalu mimo počáteční a koncový bod je jednoduchá [14].

Jedna křivka může mít také vlivem validních matematických úprav nejednu možnost zápisu. Situace je analogický se soustavou dvou rovnic, kdy je možné bez změny významu obě rovnice např. vynásobit stejným číslem [14].



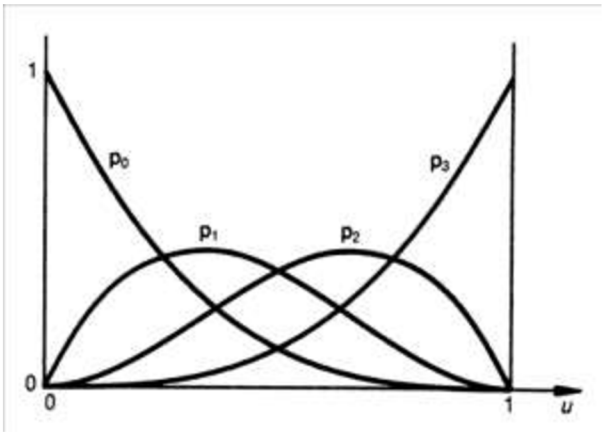
Obr. 1.24 Rozdíl mezi analytickou spojitostí C^0, C^1, C^2 [15].

1.6.3 Reprezentace geometrie interpolačními a aproximačními křivkami

Druhým řešením analytických vyjádření jsou interpolační a aproximační metody. Geometrie je zadána množinou bodů, kterými se křivka prokládá. Interpolační křivky prochází všemi řídícími body, aproximační jsou jimi pouze řízeny a procházet jimi nemusí [7] [15].

Existuje několik interpolačních a několik aproximačních metod. Mezi nejznámější interpolační patří: Lagrangeova interpolace, Hermitova. Mezi nejznámější aproximační patří: Bézierovy, Coonsovy, B-spline, NURBS, T-spline [7] [15].

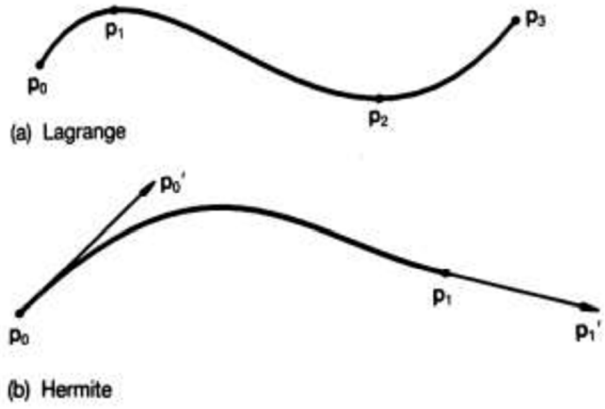
Většina z nich potřebuje pro výpočet bázové rovnice, které určují, jak moc ovlivní každý z řídících bodů právě počítaný bod na křivce. Bernsteinovy polynomy stupně n=3 tak např.



Obr. 1.25 Bázový prostor určený Bernsteinovými polynomy třetího stupně [7].

pro Bézierovu kubiku tvoří bázový prostor jako na Obr. 1.25. Pro každý bod na intervalu <0;1> na ose *u*, tedy i pro každý bod na křivce, tak lze graficky vidět, jak velkou mírou je jeho poloha určena. Krajní body křivky jsou určeny pouze krajními řídícími body a tak v bázovém prostoru dosahují Bernsteinovy polynomy p_0 a p_{max} svého maxima, tj. jedné, tedy stoprocentního vlivu na bod. Bázové křivky p_n dosahují svého maxima v místě kde jsou nadefinovány jejich příslušné řídící body [7].

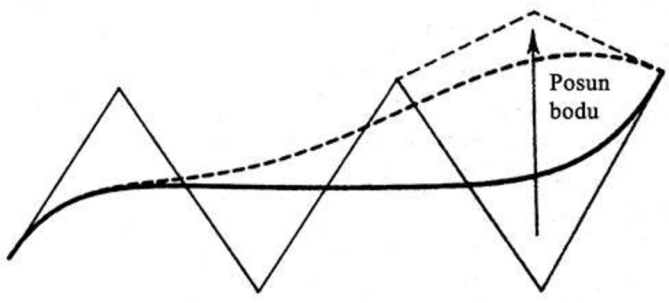
Hermitovské kubiky jsou dány dvěma řídícími body a dvěma vektory, **Lagrangeova kubika** je dána 4 body (viz Obr. 1.26) [7] [15].



Obr. 1.26 Lagrangeova kubika a Hermitova [7].

Bézierovy křivky prochází koncovými řídícími body a jsou náchylné na změnu polohy jakéhokoliv řídícího bodu Obr. 1.27. To znamená, že při editaci jednoho bodu se změní tvar celé křivky. Křivky jsou dány vztahem 1.3, kde $B_{i,n}$ je i-tý Bernsteinův polynom n-tého stupně, což jsou rekurentní vztahy určující bázi vektorového prostoru. Jejich náchylnost na změnu lze pozorovat už v bázovém prostoru, kdy jsou s výjimkou krajních bodů všechny bázové křivky nenulové [7] [15].

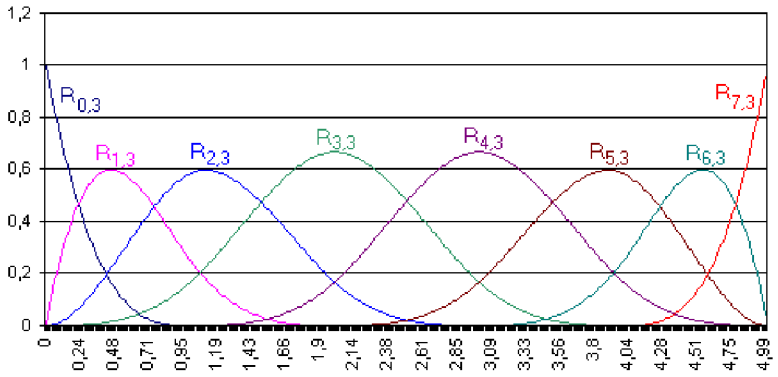
$$C(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i, \tag{1.3}$$



Obr. 1.27 Globální projev posunu bodu [4].

NURBS křivky jsou dnes nejpoužívanější reprezentací geometrie, jelikož splňují většinu požadavků, které programátoři, či grafici nárokují. NURBS je akronym pro NeUniformní Racionální B-spline. Jsou zobecněním Bézierových a B-spline křivek, které jsou jejich speciálním případem. Neuniformnost značí, že odpadl nárok na konstantní vzdálenost uzlů a racionalita dovoluje přiřazovat bodům váhu [7] [2] [15].

Z bázového prostoru na Obr. 1.28 je vidět, že jednotlivé body mají pouze lokální vliv. Např. vliv prvního bodu $R_{0,3}$ končí kolem vzdálenosti 1,0.



Obr. 1.28 Racionalita B-spline báze [15].

Metody výpočtu NURBS křivek lze stejně, jako předchozí metody, rozšířit o dimenzi a získat tak popis ploch. Jejich výpočet je již natolik komplikovaný, že akronym mnozí matematikové (př. Watt 2014, ÚM FSI VUT 2012, Iglesias 2011, Piegl&Tiller 1996) překládají úsměvně jako Nobody Understands Rational B-Splines. NURBS plochy jsou definovány rovnicí 1.4, do kterých je potřeba dopočítat pro každý bod vztah 1.5, do kterých je potřeba dopočítat vztah 1.6 [7] [2] [15].

$$Q(r,s) = \sum_{i=0}^{a}\sum_{j=0}^{b} P_{ij} R_i^j(r,s), \tag{1.4}$$

$$\text{kde } R_i^j(r,s) = \frac{w_{ij} N_i^m(r) N_j^n(s)}{\sum_{k=0}^{a}\sum_{l=0}^{b} w_{kl} N_k^m(r) N_l^n(s)}. \tag{1.5}$$

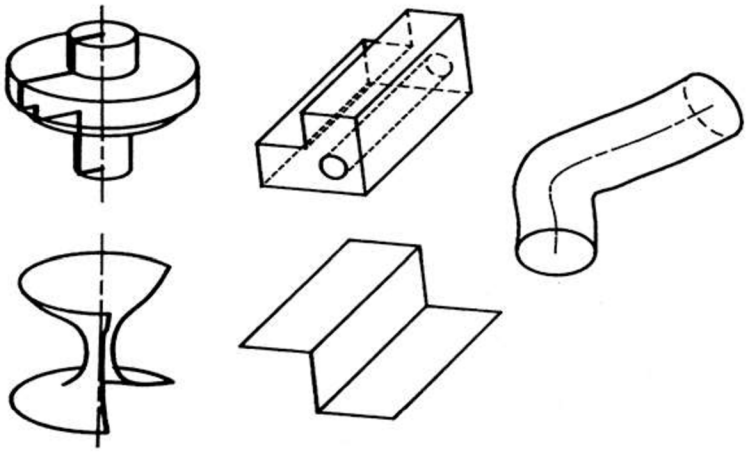
$$N_i^0(t) = \begin{cases} 1 & \text{pro} \quad t \in \langle t_i, t_{i+1}) \\ 0 & \text{jinde} \end{cases}$$

$$N_i^k(t) = \frac{t - t_i}{t_{i+k} - t_i} N_i^{k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1}^{k-1}(t), \tag{1.6}$$
$$\text{pro } 1 \le k \le n-1,\ 0 \le i \le n-k-1$$

V praxi však obvykle stačí neuniformnost a racionality se nevyužívá. Ještě o řád obecnějším popisem ploch jsou T-splines, které umožňují porušit obdélníkovou topologii řídících bodů a v kritických místech tak přidat řídící body, naopak v nekomplexních je ubrat [7] [2] [15].

1.6.4 Kinematický popis ploch

Poslední možností řešící neduhy analytického vyjádření je kinematický popis ploch. Plochy lze totiž také zadefinovat translací, rotací (*revolve*), nebo kombinací – šroubovým vytažením křivky v prostoru (*screw*). Dále existují *skinned*, nebo *lofted* plochy, tedy plochy vzniklé potažením křivek, nebo plochy vzniklé šablonováním křivky po trase (*sweep*). Příklady na Obr. 1.29 [7].



Obr. 1.29 Příklady povrchů vzniklých kinematickým popisem [7].

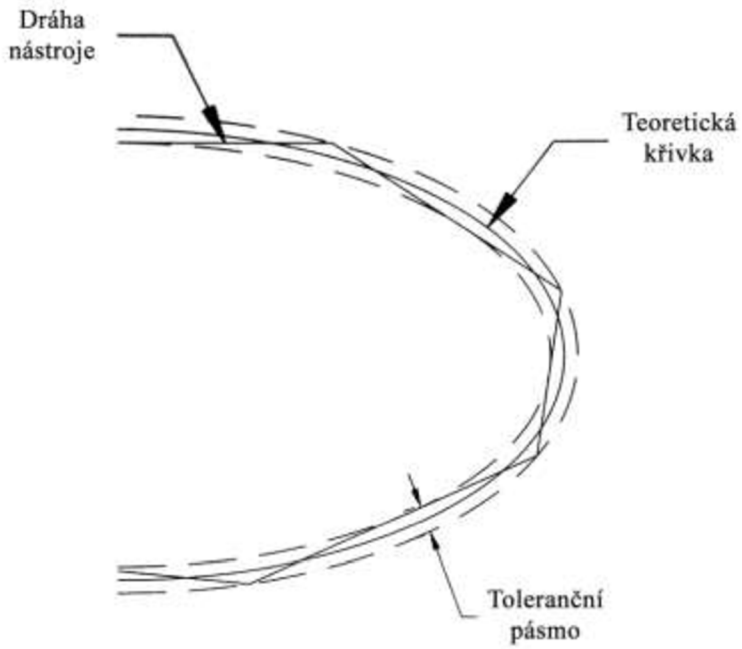
2 NAVRŽENÉ VARIANTY ŘEŠENÍ – MATEMATICKÉ, TECHNOLOGICKÉ

V kapitole je rozebráno, jak postupovat při tvorbě geometricky složité součástky, jsou uvedeny důvody pro parametrizaci a je vysvětleno řešení problému složitých výpočtů versus času dostupného na výpočet pro hladký chod stroje.

2.1 Postup tvorby součástky

Pokud by byla známa rovnice křivky nebo plochy, stačilo by pomocí ní nakreslit geometrii v CAD programu a přímo zapsat do NC programu. Tento přístup by přinesl sice výborné výsledky, nicméně v praxi to není často reálné. Hlavním důvodem neproveditelnosti může být například fakt, že lidská složka si snadno představí a vymyslí tvar světlometu, formy na formu na bábovku apod., ale u takto komplikovaných tvarů nevymyslí rovnici. Proto používaným druhým způsobem je prvně modelace v CAD systému a následná snaha o získání dat, které geometrii popisují.

Data z CAD systému lze získat trojím způsobem. Jednak lze součást okótovat a případně vytvořit výkres, což však pro obecné plochy obvykle ani reprezentativní, natož vhodné. Další variantou je extrakce geometrických dat ze souboru (např. *.igs) a třetí možností je export do CAM systému. CAM systém pak díky postprocessoru umí vytvořit přímo NC kód.



Obr. 2.1 Aproximace elipsy tangenciálními úsečkami, které jsou v daném tolerančním pásmu [4].

2.1.1 Výběr způsobu pohybu os

Některé CAM systémy a některé CNC stroje rozumí z kódů pro pohyb os pouze lineární a kruhové interpolaci. Proto je možnost složitější úseky aproximovat krátkými lineárními a kruhovými elementy (viz Obr. 2.1). Navíc je tato metoda poměrně přímočará a jednoduchá a tak se etablovala jako téměř konvenční metoda řešení složitějších úseků [4].

Tento přístup má však několik nevýhod. První komplikací je exponenciálně narůstající objem dat se zužujícím se tolerančním pásmem. Druhým je ryv (*jerk*), tedy změna zrychlení, vznikající neustálým rozjížděním a zpomalováním na začátku a konci každého dalšího aproximačního úseku.

Vylepšení této rozšířené metody přinesly moderní řídící jednotky, které obsahují tzv. kompresory. Ty prokládají zadanými body křivky v daných tolerancích, které jednak aproximují lépe původní tvar než lineární interpolace a jednak omezují ryv.

Dalším přístupem, který umožňují modernější CAM systémy, je export spline interpolací. CAM tak exportuje program s příslušnou adresu skupiny příkazů pro pohyb, např. BSPLINE a doplní sérii řídících bodů, kterými řídící jednotka spline proloží.

Výše uvedené jsou však pouze interpolační nebo aproximační metody. To znamená, že i přesto, že jejich toleranční pole může být velice úzké a reálně chybu způsobují jiné faktory, jako přesnost servomotorů, je stroji de facto předem zadáno, aby se pohyboval po trajektorii různé od vymodelované/kýžené.

Možností, jak deklarovat pohyb přesně je použití, v úvodu kapitoly uvedených, polynomiálních, parametrických křivek. U nich je však obtíž s jejich nalezením a následně i interpretací, viz [16].

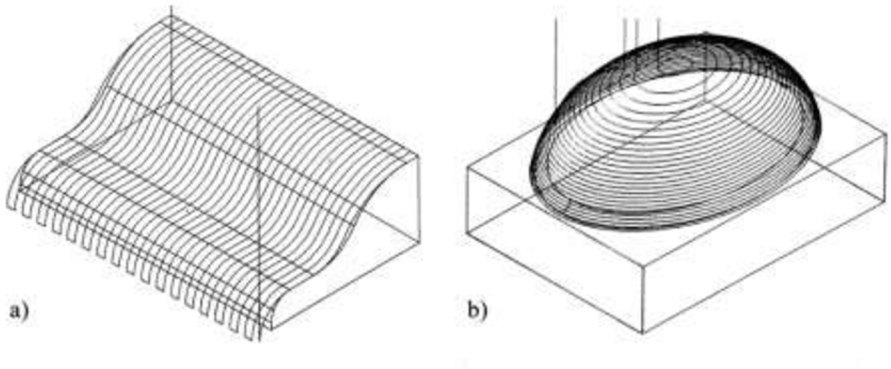
Vzhledem k tomu, že se NURBS křivky používají i jako reprezentace geometrie v CAD/CAM systémech, jsou relativně snadné na tvorbu, jeví se pro většinu případů, i vzhledem k výše uvedenému, jako nejvýhodnější řešení. Je však u nich nutné dát si pozor na interpretaci jednotlivými systémy, kdy každý může řídícími body prokládat křivku jiným způsobem.

2.1.2 Dráhy nástroje

Možností jak k 3D plochám přistoupit existuje mnoho. Z hlediska dráhy programu existují pro dokončovací operace např. dráhy po paralelu (*iso-parametric tool paths*), dráhy s konstantní hladinou Z (*waterline*), jejich kombinace, dokončení tužkou (*pencil finish*), nebo komplexní metody konstantního vrubu (*constant scallop*) [17].

Pro praktickou část byla zvolena metoda paralelních průjezdů, která není pro zadaný tvar samostatně příliš vhodná, ale je programátorsky nejjednodušší. Stačí u ní vyfiltrovat všechny hodnoty řídících bodů pro jednu konstantní osu a křivku generovat po souřadnicích ve zbývajících osách.

Vhodná by pro podobný povrch byla pochopitelně tvorba kódu exportem z CAM systému a přeložení pozic nástroje (CLDATA) pomocí postprocessoru, který ale nebyl k dispozici. Výhodou by u sofistikovanějších CAM programů byla možnost zjistit přesnou polohu normály a řídit tak lépe natočení nástroje, nebo detekce kolizí.



Obr. 2.2 Dráhy nástroje. a) izo-parametrické, tedy po paralelu
a b) waterline, tedy s konstantní výškou v ose Z [4].

2.2 Parametrizace programů

Parametrem se označuje pomocná proměnná, veličina nebo soubor veličin, které charakterizují daný stav, jev nebo proces. V informačních technologiích slouží k ukládání vstupních hodnot funkce, metody, podprogramu, apod. Důvodem pro zavedení parametrů je snaha zobecnit kód, program, jelikož v praxi nemusí být vstupní data přesně známa, nebo jsou očekávány podobné operace, ale s jinými hodnotami [18].

Parametrické programování pro CNC lze snadno využít pro technologické a logické operace v programu, jisté 2D křivky (např. program Vačka) a jednoduché 3D plochy (jako kvadriky), avšak pro skutečně obecné plochy je parametrizace velmi náročná a snad i lehce diskutabilní.

Pro parametrizaci obecné NURBS plochy dané řídícími body si autor neumí představit program/cykly, samotnou parametrizaci plochy a ani využití celé snahy. Na opačné straně stojí samotné parametrické vyjádření ploch, které se importuje pomocí funkce POLY, ale, jak již bylo naznačeno, v praxi jsou jeho rovnice obtížně zjistitelné.

2.2.1 Návrh výpočtu parametrizace obecných ploch zadaných řezy

Nejblíže kam se dá v obecnosti zajít, je pravděpodobně zadání pomocí křivek v řezu. Parametrizaci takovýchto ploch si lze představit v CAM systému, podobným způsobem jako v G-kódu. Pokud by byla zadána např. plocha lopatky turbíny za pomocí křivek v řezech, byla by úprava v CAM systému poměrně jednoduchá a vyžadovala by především vhodně parametrizované náčrtky, přes které je pak plocha tažena.

V G-kódu by bylo zapotřebí nahrazovat funkci celého CAM systému např. tímto způsobem: Nástroj by se pohyboval po trajektorii paralelní s danými křivkami po spline křivkách, přes řídící body generované jako poloha bodu na křivce prokládající body křivek v řezu v požadovaném místě. Jinak řečeno, algoritmus by tak nejprve napříč křivkami v řezu proložil první křivku a uložil do proměnné typu pole n souřadnic X, Y, Z, ležících na křivce, kde n je požadovaný počet průjezdů po povrchu. Algoritmus by pokračoval prokládáním sousedních křivek a ukládáním jejich hodnot do dalších proměnných typu pole. Po dokončení by byla vytvořena interpolační křivka přes všechny i-té body křivek vytvářených

v prvním kroku. Jedná se tedy o jistý druh *sweepu*. Navržená metoda je ale nejspíše náchylná na odlehlé hodnoty, počet zadaných křivek v řezu a jemnosti dělení/inkrementů.

2.2.2 Některé postupy s ohledem na použití parametrického programování

Pro obrobení obecných ploch je z hlediska parametrického programování tedy možné použít tyto postupy:

A. Použít parametrické programování pouze na pomocné výpočty a geometrii exportovat z CAM systému jako

A.1. lineární a kruhové interpolace,

A.1.1. bez možnosti komprese řídící jednotkou CNC stroje; nevýhodou nutnost CAM systému, ryv/jerking stroje, nejvyšší míra nepřesnosti;

A.1.2. s kompresí řídící jednotkou CNC stroje a tak vyhlazení dráhy; nutnost moderní řídící jednotky, výhodou je, že odpadá do značné míry jerking a nekvalitní povrch;

A.2. spline křivky, případně včetně specializovaných funkcí daného řídicího systému; hranice možností těchto automatizovaných metod bez zásahu do kódu, vysoká kvalita povrchu nevýhodou složitá editace, málo SW řešení, které je drahé, horší kontrola přesnosti.

B. Použít parametrické programování pouze na pomocné výpočty a geometrii exportovat z CAD systému

B.1. jako řídící body a proložit jimi interpolační křivku. Podobně, jako je v programech 2D křivky a 3D plochy v kapitolách 3.2 a 3.3, přílohy 2-6; výhodou možná absence CAM systému, snadná editace, nevýhodou problém s přesností, někdy s náročností výpočtu;

B.2. a najít polynomy popisující křivku, a její koeficienty dosadit např. v Sinumeriku do funkce POLY; nevýhodou je extrémní složitost a dokonce problém s interpretací (viz Ohnišťová, 2014 [16]).

C. Použít parametrické programování na výpočet dráhy nástroje za pomocí cyklů,

C.1. jednodušších tvarů se známým matematickým předpisem, jako je program Vačka (kapitola 3.4, příloha 7); výhodou jednoduchá parametrizace, možnost part families; nevýhodou omezenost na ne zcela obecné tvary jako jsou 2D křivky, kvadriky, plochy vzniklé rotací, apod.;

C.2. složitých tvarů popsaných křivkami v řezech, jako jsou lopatky. Návrh této metody je uveden v předchozí kapitole (2.2.1). V této metodě jsou však očekávány velké komplikace;

C.3. složitých obecných křivek a ploch zadaných jako řídící body, pomocí vzorců výpočtů interpolačních, nebo aproximačních křivek, jako je NURBS. Jak bude upřesněno v kapitole 2.3.2, postup nedává smysl.

2.3 Řešení pro složité výpočty v G-kódu

Doktor Polzer navrhuje pro složitější výpočty využití jedné z funkcí systému Sinumerik, konkrétně čtení a především zápisu z a do externích souborů [19]. Tímto způsobem lze vyřešit nejtíživější problém komplexních programů, kterým je výpočetní čas. Zvláště u vysokorychlostního obrábění je požadavek na zpracování dalšího bloku se souřadnicemi v jednotkách milisekund, čehož nelze dosáhnout při dnešní technice téměř jinak, než po sobě jdoucích kódů s přímo určeným pohybem, které výpočetně nezatěžují systém. Prodlevy, které jsou na offline simulaci nezaznamenatelné, mohou být při náročném obrábění kritické.

Bylo tak navrhnuto, aby byl logicky oddělen program pro výpočet a program pro online obrábění. První, výpočetní program (případně série programů, podprogramů a cyklů) by tak obsahoval veškeré výpočty, které by byly evaluovány v rámci offline simulace a jejichž výsledky by byly zapisovány do nového souboru, programu pro samotný běh obrábění.

2.3.1 Možné využití a přínosy postupu

Tato metodika byla úspěšně otestována v poslední skupině programů teoretické části na výrobu vaček, jejichž přesnější popis je v kapitole 3.4. Zároveň tento způsob ale vrací do úvah o reálné využití i o stupeň komplexnější programování obecných ploch, kde je zapotřebí ne nutně logicky, ale výpočetně náročné operace. Je tu tak možnost naprogramovat cykly, které by při zadání řídících bodů, ať už do jednotlivých řádků, nebo do pole, jako globální proměnné volily trasu pro spline interpolaci, o něco složitěji i normálu k povrchu pro řízení polohy nástroje a jeho kompenzaci a spíše teoreticky i kontrolu kolizí. Případ volby trasy přes jednotlivé řídící body je v posledním programu druhé skupiny programů teoretické části, jako Příloha 6.

Díky této metodice lze také začít uvažovat o složitějších matematických operacích, jakým je diferenciální počet, řady, limity apod., a to díky kombinaci s numerickými metodami.

2.3.2 Výpočet NURBS G-kódem

Byl by také možný přímý výpočet polohy nástroje přes matematické rovnice interpolačních křivek. Tedy po zadání řídících bodů (bez jakékoliv parametrizace) by cyklus propočítal následující polohu nástroje. Tato metoda se však v průběhu práce ukázala jako slepá ulička. Zbytečná a pro NURBS plochy extrémně náročná. Např. pro zmíněné NURBS plochy by bylo nutné pro každý následující bod vypočítat Q(r,s), pro které je nutné vypočítat Ri,j(r,s), které obsahuje dvě sumy z rekurzivních bázových rovnic, které mají platnost jen pro určitý interval, který by bylo nutné otestovat. A výsledek by nepřinesl nic lepšího, než funkci BSPLINE v Sinumeriku, nebo obdobné funkce v jiných systémech.

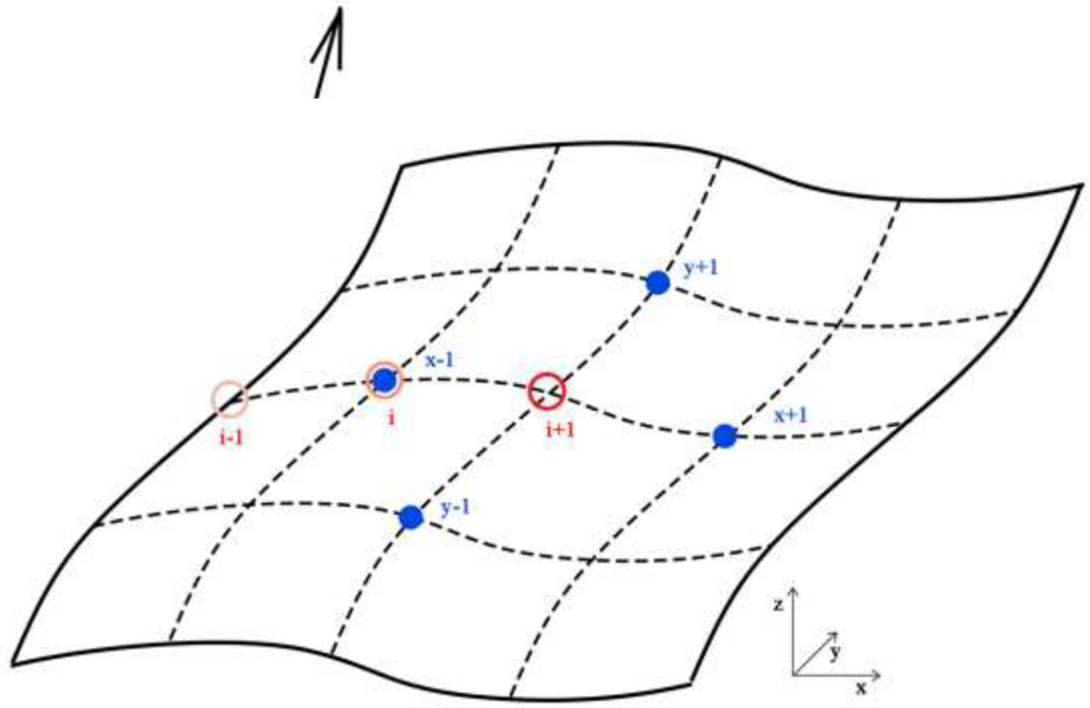
2.3.3 Návrh výpočtu dráhy nástroje

Protože izo-parametrické dráhy nejsou např. pro tvar 3D plochy z kapitoly 3.3 nejvhodnější a waterline dráhy by byly z hlediska parametrického programování pro G-kód obtížně řešitelné pro obecné případy, byl navržen odlišný postup.

Algoritmus, při kterém by byly tvořeny spirály směrem dovnitř. Systém by si ze zadaných souřadnic řídících bodů vybral všechny dosud nepoužité body v daném směru, proložil jimi funkci SPLINEPATH, otočil kolem osy Z o 90° a opakoval algoritmus.

2.3.4 Návrh výpočtu normály k povrchu

Pro výpočet normály k povrchu navrhuje autor použít následující aproximaci. Je dáno pole [m,n], pro které platí, že m je dle metodiky větší nebo rovno počtu řídících bodů, n=4 pro souřadnice *X, Y, Z* a váhu bodu *W*. Pro každý následující bod jsou vybrány 2 nejbližší okolní řídící body v ose *X* a 2 v ose Y (viz Obr. 2.3). Dvěma vybranými body v dané ose je proložena přímka, na kterou je dopočítána kolmice (viz Obr. 2.4). Vektor této kolmice je pro



přepony, pro konvexní křivky a pod pro konkávní křivky.

Obr. 2.3 Pro výpočet normály k povrchu pro následující bod i+1 je navrženo vybrat 2 okolní body pro dvě osy.

jednu osu úhel lambda a pro druhou omega. Úhel je možné připsat po propočtu za vybraný bod, tedy souřadnice *X, Y, Z*, do parametrů A[p], B[p], C[p], kde p náleží {1, 2, 3, 4}, dle zvolené metody orientace nástroje. Aktuální bod pro výpočet normálového vektoru není použit.

2.3.5 Návrh výpočtu kompenzace nástroje

Na návrhu výpočtu normály k povrchu je založen i návrh na výpočet kompenzace nástroje. Pro kompenzaci nástroje je nutné spočítat ekvidistantní plochu. To by mohlo být aproximováno takto:

Z každého řídícího bodu plochy je zkonstruována úsečka o délce kompenzace nástroje, obvykle poloměru *r*, která má stejnou směrnici, jako normála k ploše a je orientována v kladném směru osy *Z*.

S výslednou sítí bodů je zacházeno jako s geometrií obrobku, a je pro ní volena dráha nástroje, tedy např. proložení izo-parametrickými BSPLINE křivkami. BSPLINE parametry *X*, *Y*, *Z* by tak odpovídaly bodům na ekvidistantní síti bodů a parametry A, B, C vektoru normály.

3 EXPERIMENTÁLNÍ ZKOUŠKY

Pro praktickou část bylo zvoleno několik různých tvarů pro výrobu a k nim několik možných přístupů. U všech programů však bylo cílem využít splinové interpolace prokládané řídícími body, jelikož pouze v tomto přístupu vidí autor cestu pro tvorbu skutečně obecných ploch.

3.1 Makro pro převod z IGES do NC a dávková editace NURBS souřadnic

Následující programy, pro obecnou 2D křivku a 3D plochu, nazvané pracovně Podpis a Blob, byly vytvořeny za pomocí CAD/CAM systému CATIA. V systémové části pro CAM byla vytvořena simulace obráběcího procesu, ale popis není dále uveden, jelikož zajímavým výsledkem by pro tuto práci byl především export NC kódu s XH, YH, ZH souřadnicemi, což pro absenci postprocessoru nebylo možné.

Byly tak exportovány pouze křivky, respektive plochy, reprezentované řídícími body a to do univerzálního formátu IGES (koncovka *.iges nebo *.igs), se kterým se dále pracovalo. Formát IGES není závislý na výrobci a je podporován napříč platformami, práce s ním je tedy shodná nehledě na CAD systém, ve kterém byl vytvořen. Obsahuje pouze geometrická data, má předepsanou strukturu a lze tak automatizovat extrakci potřebných dat z něj. Předchozí popis platí také pro novější formát STEP (koncovka *.step nebo *.stp). V IGES souboru jsou NURBS a B-spline plochy definovány číslem 128 resp. 126, za kterými následuje série čísel upřesňující křivku a poté souřadnice řídících bodů [20] [21].

Pro extrakci geometrie lze sestrojit plně automatizovaný program, jako to udělali zajisté mnozí dříve, např. Liang a Li (2009), ty ale buďto nejsou veřejně dostupné, placené, nebo obtížně dohledatelné. Jelikož nejde o velký problém, bylo rozhodnuto sestrojit semi-automatický postup na převod z IGES do NC kódu na bázi jednoduchého makra [22].

Pro otevření a editaci obou IGES souborů, pro obě vytvořené geometrie, byl použit program Notepad++. Ten, podobně jako mnohé jiné, umožňuje funkci 'najít a nahradit' (find and replace) s podporou regulárních výrazů, tvorbu maker a z komfortních důvodů i další prvky usnadňující programátorům práci, nehledě na programovací jazyk.

Úvod do regex, který byl použit pro automatizaci konverze z IGES do NC kódu, byl uveden v kapitole 1.4. V konkrétním případě požadavku na extrakci NURBS a B-spline entit byly použity např. následující výrazy:

Hledáno: "{5,}\w{2,4} {2,7}\d{1,5}",

nahrazeno: "", předpis odmaže nepotřebné konce řádků.

Hledáno: ";128,..,..,.,(.)",

nahrazeno: "\nBSPLINE SD=$1 F=#9 ;nurbs surf\n",

předpis odsadí každou NURBS plochu na zvláštní řádek a přiřadí jí G-kód pro pohyb po BSPLINE.

Hledáno: "(-*[0-9]+\.[0-9]+),(-*[0-9]+\.[0-9]+),(-*[0-9]+\.[0-9]+),",

nahrazeno : "X$1 Y$2 Z$3\n",

převede sérii souřadnic zapsaných jako desetinná čísla s možným znaménkem mínus, oddělených od sebe pouze čárkou do jednotlivých řádků s přiřazením G-kódů pro pohyb v jednotlivých osách.

Pro průjezdy s konstantní hodnotou v ose Y bylo hledáno

"X50.0 Y(-*[0-9]+\.[0-9]+) Z(-*[0-9]+\.[0-9]+)\s{0,2}X(-*[0-9]+\.[0-9]+) Y(-*[0-9]+\.[0-9]+) Z(-*[0-9]+\.[0-9]+)",

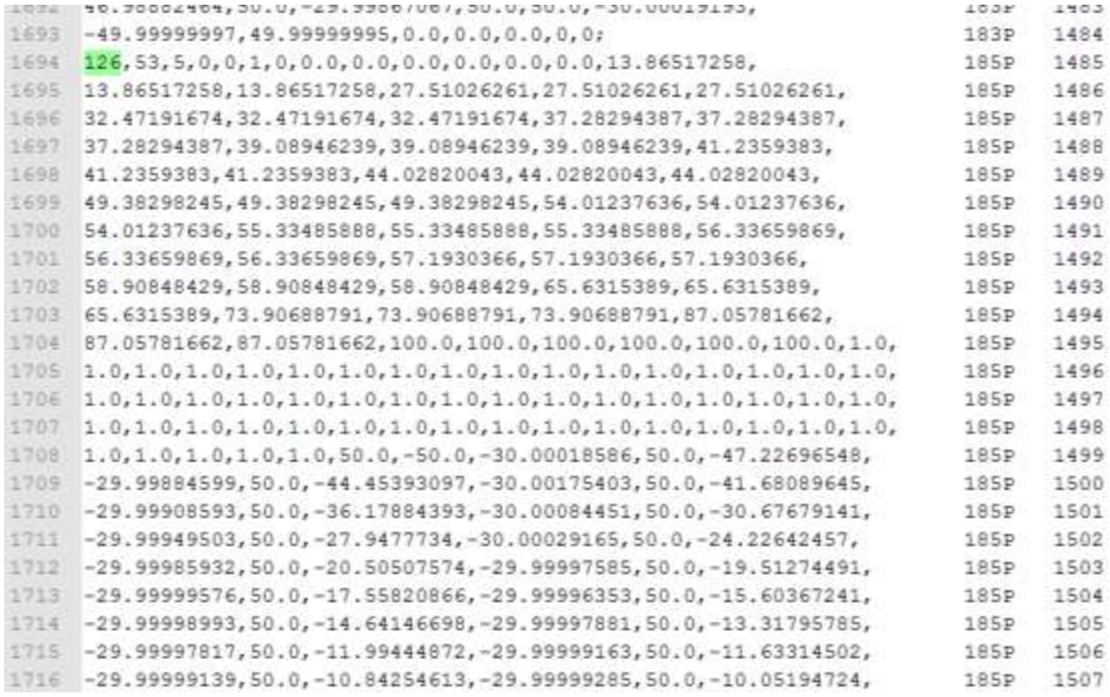
nahrazeno

"X50.0 Y$1 Z$2\nG0 Z5\nX$3 Y$4\nG1 Z$5\n\nBSPLINE\nX$3 Y$4 Z$5"

a obdobně pro průjezdy s konstantní hodnotou v ose X bylo hledáno

"X(-*[0-9]+\.[0-9]+) Y-50.0 Z(-*[0-9]+\.[0-9]+)\s{0,2}X(-*[0-9]+\.[0-9]+) Y(-*[0-9]+\.[0-9]+) Z(-*[0-9]+\.[0-9]+)",

nahrazeno



Obr. 3.1Výchozí podoba IGES souboru se zaměřením na začátek zápisu NURBS entity.

"X$1 Y-50.0 Z$2\nG0 Z5\nX$3 Y$4\nG1 Z$5\n\nBSPLINE\nX$3 Y$4 Z$5".

Předpisy tak naleznou poslední souřadnici jednoho průjezdu, první souřadnici nového a vloží mezi ně G-kódy pro přejezd nástroje. Výchozí podoba IGES souboru je na Obr. 3.1 a po použití napsaného makra, uvedeného v příloze 1 je na Obr. 3.2.

Při simulaci s velmi malým záběrem se ukázalo, že rotace stolu se současným vjížděním frézy s kulovým čelem působí problém a bylo potřeba na místě hbitě doplnit kód o nájezdy a přejezdy. Toho šlo snadno docílit i pro velké množství řádků pomocí regex výrazu

```
1781  X43.97764927 Y-50.0 Z-30.0020299
1782  X46.98882464 Y-50.0 Z-29.99872786
1783  X50.0 Y-50.0 Z-30.00018586
1784  G0
1785  M17
1786
1787
1788  BSPLINE SD=5 F=#9 ;bspline curve
1789  X-50.0 Y50.0 Z-30.00012772
1790  X-50.0 Y47.41156332 Z-30.00006662
1791  X-50.0 Y44.82312665 Z-29.99966836
1792  X-50.0 Y42.23468997 Z-30.00021838
1793  X-50.0 Y37.01606755 Z-29.99983002
1794  X-50.0 Y31.79744514 Z-30.00022515
1795  X-50.0 Y29.1672594 Z-29.99988877
1796  X-50.0 Y24.88200385 Z-30.00033753
1797  X-50.0 Y20.59674831 Z-29.99972967
```

Obr. 3.2 Podoba souboru IGES po spuštění vytvořeného makra.

"([A-Z]*-*[0-9]+\.[0-9]+) ([A-Z]*-*[0-9]+\.[0-9]+) ([A-Z]*-*[0-9]+\.[0-9]+)\s{1,}.{1,}\s{1,}.{1,}\s{1,}.{1,}\s{1,}.{1,}\s{1,}BSPLINE\s{1,}([A-Z]*-*[0-9]+\.[0-9]+) ([A-Z]*-*[0-9]+\.[0-9]+) ([A-Z]*-*[0-9]+\.[0-9]+) ([A-Z]*-*[0-9]+\.[0-9]+)"

hledajícího opět poslední a první řádek NURBS křivek, včetně technologicky chybného kódu mezi nimi. Nahrazeno bylo pomocí výrazu

"$1 $2 $3\nG1 X70\nG0 Z10\nX-70 $5\nG1 $6\nG1 X-50\n\nBSPLINE\n$4 $5 $6",

který doplní první a poslední řádek tak, jak byly a mezi ně vepíše přejezd o 20, rychloposuvem do Z nad obrobek, nad nájezd, pracovním posuvem do Z další křivky, přiblížení k začátku křivky a spuštění BSPLINE funkce.

Při tvorbě programu k 3D ploše, konkrétně pro podprogram obsahující natočení nástroje v ose průjezdu byl zapotřebí použití programu MS Excel, právě na výpočet normály. Zpátky do editoru byly zkopírovány pouze číselné hodnoty a po nahrazení desetinné čárky za tečku a mezer za tabulátory bylo potřeba znovu oddělit křivky průjezdů od sebe a tentokrát do kódu mezi nimi vepsat i natočení v ose A pro první bod následující BSPLINE křivky. Hledaným výrazem bylo

"(-*[0-9]+\.[0-9]+) 31.450+ (-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+)\s{1,}(-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+)",

nahrazeným pak

"$1 31.45 $2\nG1 Y65\nG0 Z5\nY-65\nG1 A=$7 C=12.5\nX$4 Z$6\nY$5\n\n BSPLINE\n$4 $5 $6 $7",

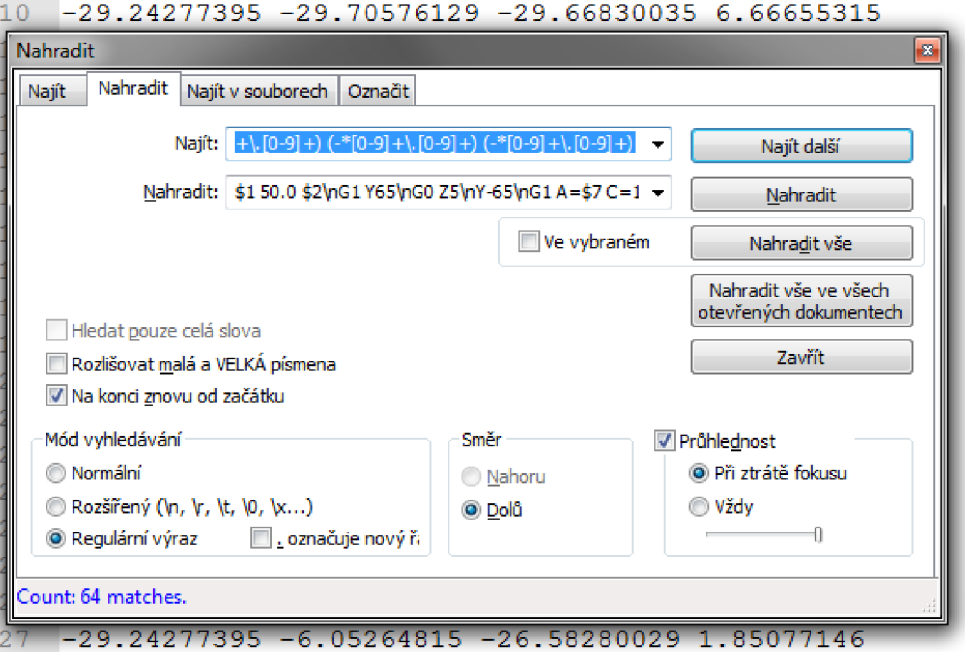
to ilustruje Obr. 3.4 a Obr. 3.3. Poté stačilo doplnit adresy X, Y, Z, A a konstantní úhel C pomocí nahrazovaného

"(-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+) (-*[0-9]+\.[0-9]+)\s{1,}",

nahrazeném

"X$1 Y$2 Z$3 A=$4 C=12.5\n".

```
104  -30.02138697 23.28310649 -29.42793453 -2.70600841
105  -30.02138697 26.56561999 -29.67458699 -2.44541368
106  -30.02138697 28.19374666 -29.78512995 -2.41596916
107  -30.02138697 29.82187333 -29.89434030 -2.32699394
108  -30.02138697 31.45000000 -29.99952414 -0.00040417
109  -29.24277395 -31.45000000 -29.99881873 6.79702633
110  -29.24277395 -29.70576129 -29.66830035 6.66655315
```




```
127  -29.24277395 -6.05264815 -26.58280029 1.85077146
```

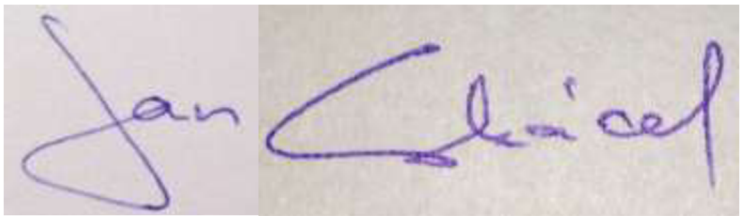
Obr. 3.4 Nalezení hranice dvou průjezdových křivek regulárními výrazy mezi souřadnicemi exportovanými z MS Excel.

```
51  -30.80000000 26.56561999 -30.00021838 0.01217485
52  -30.80000000 28.19374666 -29.99966836 -0.00881560
53  -30.80000000 29.82187333 -30.00006662 -0.00135247
54  -30.80000000 31.45 -30.00012772
55  G1 Y65
56  G0 Z5
57  Y-65
58  G1 A=3.42003983 C=12.5
59  X-30.02138697 Z-29.99949295
60  Y-31.45000000
61
62  BSPLINE
63  -30.02138697 -31.45000000 -29.99949295 3.42003983
64  -30.02138697 -29.70576129 -29.83377100 3.32639921
65  -30.02138697 -27.96152258 -29.67259687 3.29552322
66  -30.02138697 -26.21728387 -29.51292210 3.17396367
```

Obr. 3.3 Regulárními výrazy vložený kód do oblasti definované na Obr. 3.4.

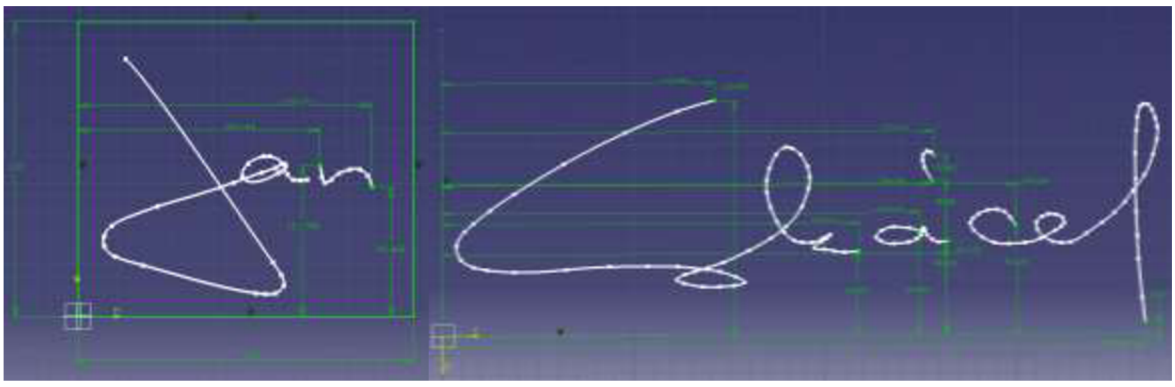
3.2 Program obecné křivky - Podpis

Cílem prvního programu bylo vyrobit obecnou křivku v ploše. Požadavkem byla tedy nespojitost, inflexní body, smyčky a prakticky nepopsatelnost polynomy. Tato kritéria splňují i podpisy, proto byl jeden vyfocen (Obr. 3.5), v CATII definován řídícími body a proložen NURBS křivkou (Obr. 3.6). Geometrie byla exportována do IGES souboru a přeložena do G-kódu dle postupu uvedeného v předchozí kapitole. Následně byly doplněny příslušné technologické bloky, nástroj byl napolohován dle vektoru, jednak pro simulaci držení pera pod úhlem, ale opět především ze cvičných důvodů. NURBS křivky byly původně volány podprogramy kvůli přehlednosti kódu, to však bylo pro online chod změněno a tento celistvý program je uveden kompletní v elektronické příloze a kvůli velkému počtu souřadnic je ve fyzické podobě zkrácen, stejně jako další programy.

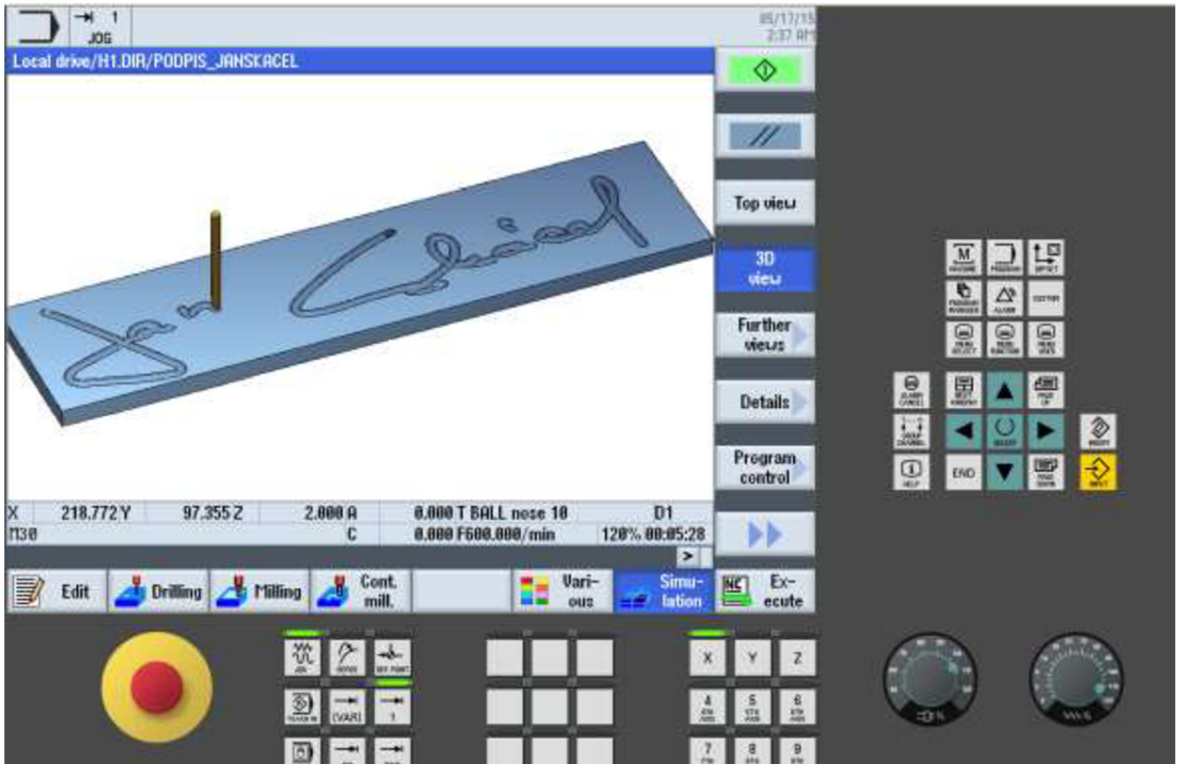


Obr. 3.5 Použité výchozí části podpisu.

Výroba proběhla na pětiosé frézce s frézou s kulovým čelem o průměru 4, S8000, F400. Program musel být alterován škálováním a transformací pro přesné umístění na obrobek a byla zjištěna chyba v generaci kódu, viz níže. I přes rotaci stolu v obou osách a plynulou změnu v ose Z se jedná o jednoduché použití interpolačních křivek, podobné spíše 2,5D obrábění. Výsledek tak dopadl dle očekávání ze simulací dobře, viz Obr. 3.7.



Obr. 3.6 Křivky s řídícími body v systému CATIA.

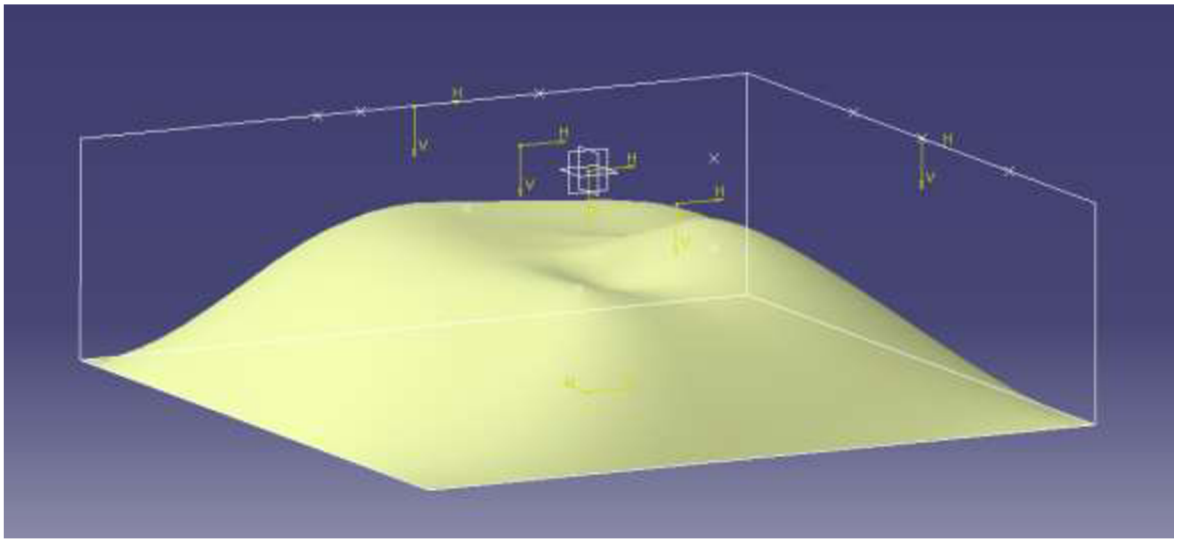


Obr. 3.8 Proběhlá simulace programu podpisu, bez polohování v A a C, včetně prostředí SinuTrain pro Sinumerik OPERATE V4.5 Ed.2.

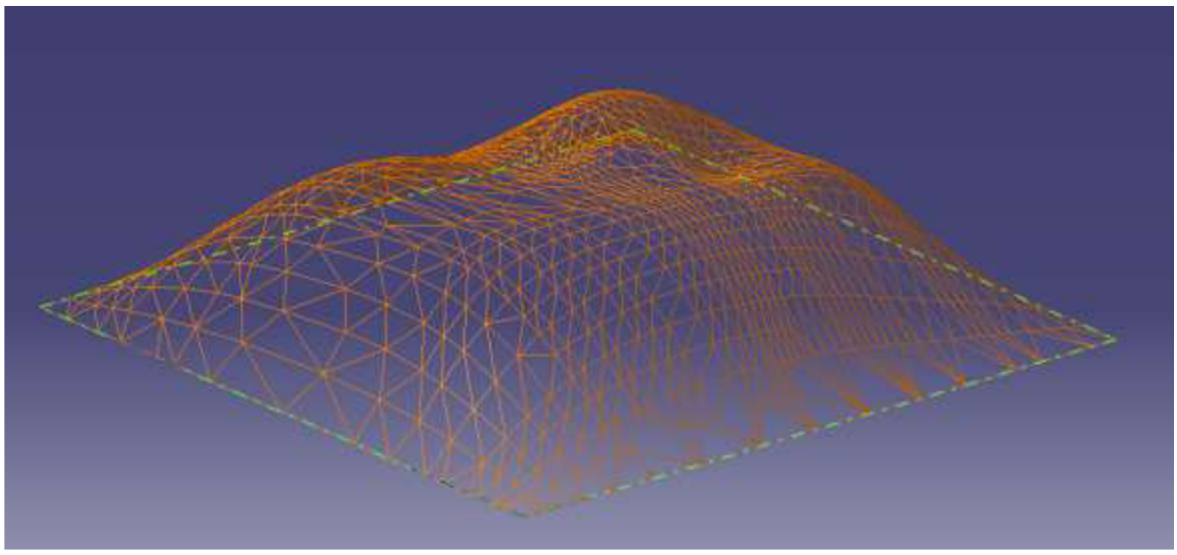
Obr. 3.7 Obrobek s vyfrézovanou konturou podpisu.

3.3 Programy obecné 3D plochy

Pro obecnou 3D plochu bylo v CATII vymodelováno těleso s pracovním názvem Blob (Obr. 3.10 a Obr. 3.9). Jedná se o obecnou plochu, komplikovanou výrazným stoupáním a inflexními body, měnící tak několikrát konvexnost a konkávnost povrchu. Na obrobku se tak dobře projeví jakákoliv chyba v postupu.



Obr. 3.10 Reprezentace plochy v systému CATIA.

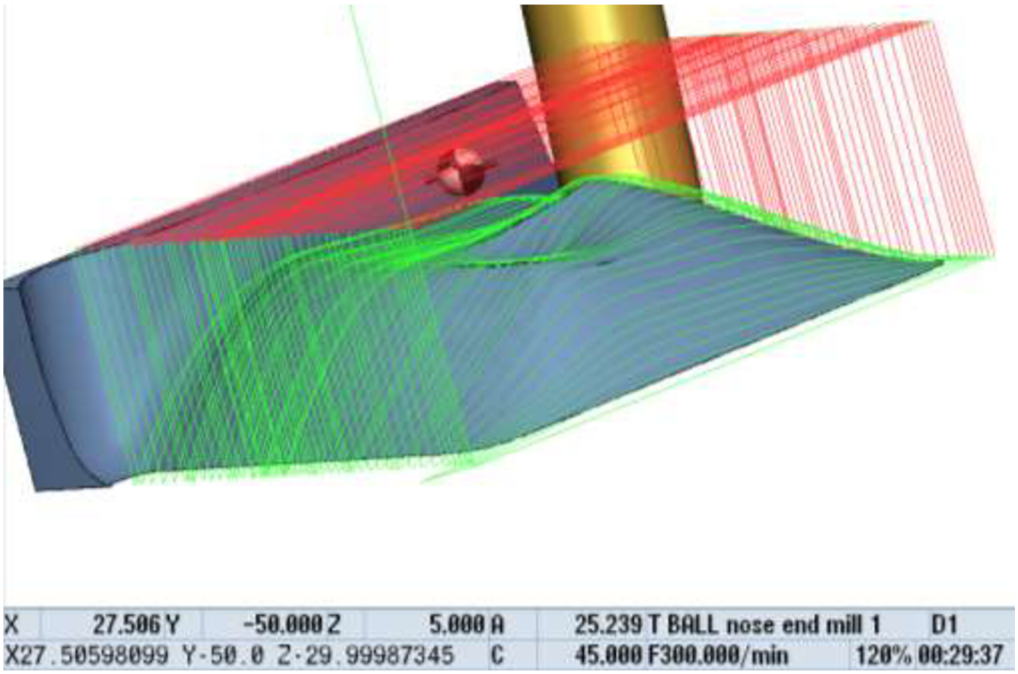


Obr. 3.9 Reprezentace plochy pomocí sítě z trojúhelníků.

Bylo vytvořeno několik programů, lišících se ve funkčnosti, nebo formě zápisu. Základní varianta umí natáčet nástroj dle zvoleného vektoru, druhá natáčí plynule nástroj ve směru posuvu, třetí umožňuje hrubování a čtvrtá využívá volání souřadnic zapsaných pod hlavní částí programu. První tři jsou volány jako podprogram, čtvrtá je samostatný .mpf.

3.3.1 První tři varianty

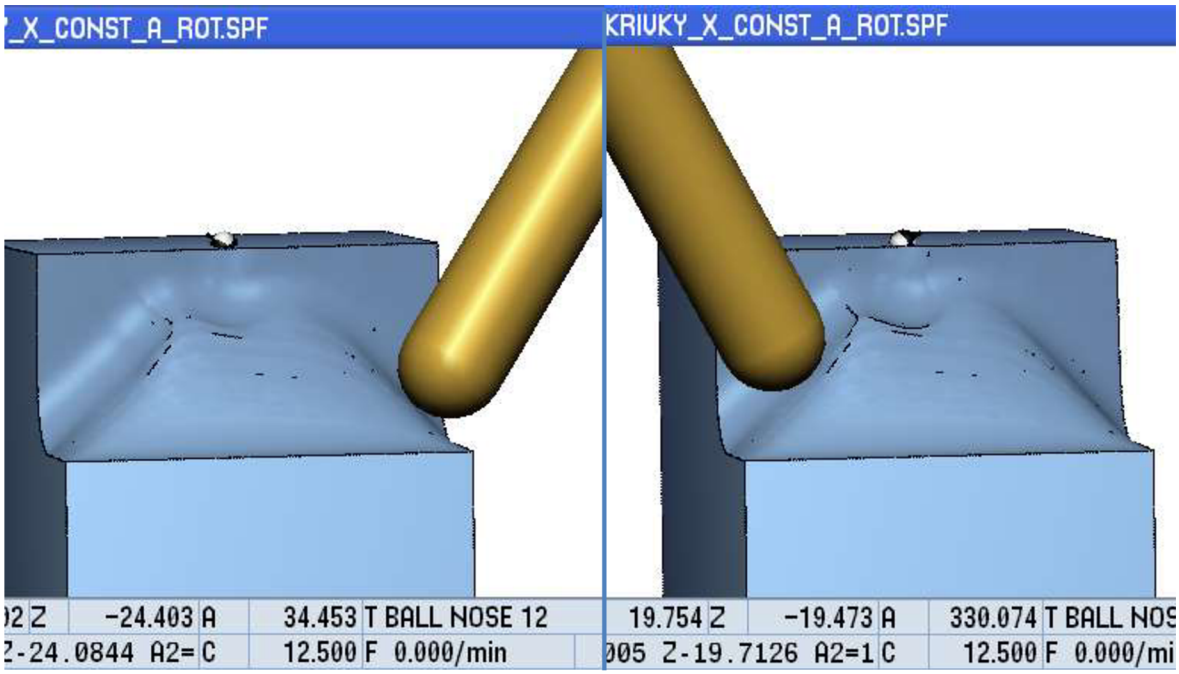
První varianta podprogramu, pojmenovaná KRIVKY_Y_CONST_SPF, byla použita po alteraci i pro výrobu. Změny kvůli výrobě byly jednak přepsání hlavičky a ukončení, aby bylo možné spustit kód jako samostatný program a jednak po simulaci byly připsány přeběhy a náběhy na spline křivky. Podprogram lze snadno vytvořit makrem uvedeným v kapitole 3.1 a příloze 1 s připsáním úvodních a ukončujících bloků. Také byl narotován stůl, aby na střed nástroje s nulovou řeznou rychlostí nebyl vyvíjen takový nápor, což je dáno vektorem A3=1 B3=2 C3=6. V příloze je uveden kód použitý pro výrobu.



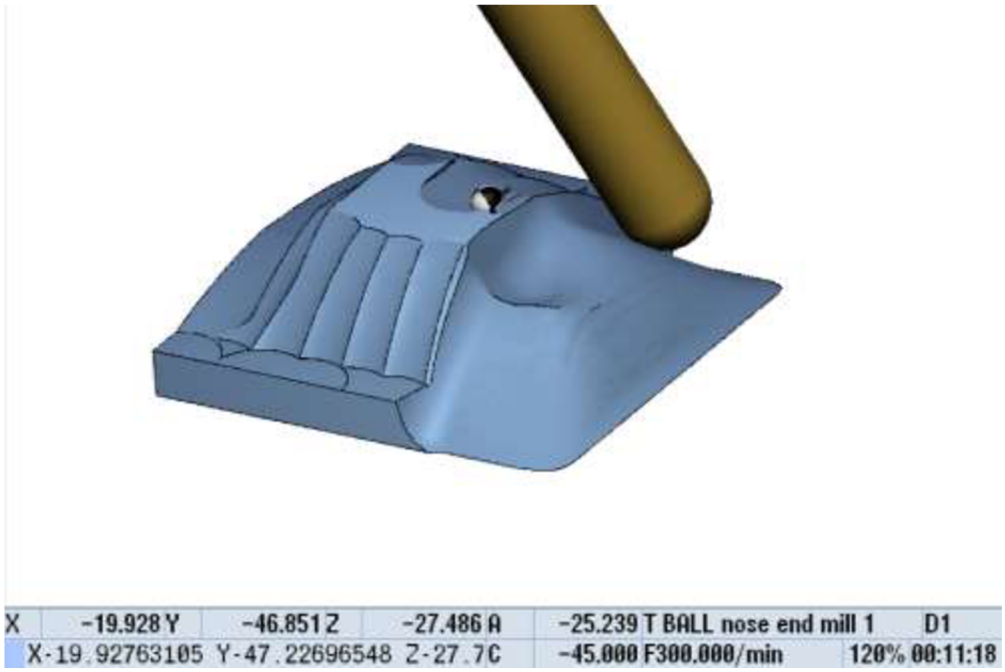
Obr. 3.11 Simulace varianty programu číslo jedna, s pohybem v jednom směru a rychloposuvem (červené linky) nad obrobek a počátek další spline křivky. Konstantní natočení nástroje.

Druhá varianta se od první liší dopočtem normál v ose průjezdu v MS Excel. Metoda je podobná navrhované v kapitole 2.3.4, ale je počítána pouze pro jednu osu, druhá byla volena konstantní. Zadaná plocha je však ve svém konkávním středu natolik prohnutá, že v simulaci docházelo ke kolizi, když se nástroj natáčel do tak silného úhlu, že odebíral materiál na hotovém povrchu. Možným řešením by bylo pro konkávní plochy vydělit úhel jistou konstantou, což by mohlo zhoršit o něco životnost, ale stále by šlo o vhodnější postup, než nenatáčet nástroj vůči obrobku vůbec. V příloze je pojmenována jako KRIVKY_X_CONST _A_ROT_SPF.

Třetí varianta byla naprogramována tak, aby obsahovala i hrubovací cyklus, ale nebyly by pro něj duplicitně zadávány stejné souřadnice průjezdu. Program se tak na základě parametru rozhodne, zda jde o hrubování a pokud ano, bude projíždět pouze každou n-tou spline křivku s odstupem v ose *Z*. V příloze má jméno KRIVKY_Y_CONST_HRUB_SPF.



Obr. 3.12 Program pro výrobu obecné plochy s rotací nástroje v rovině pohybu.



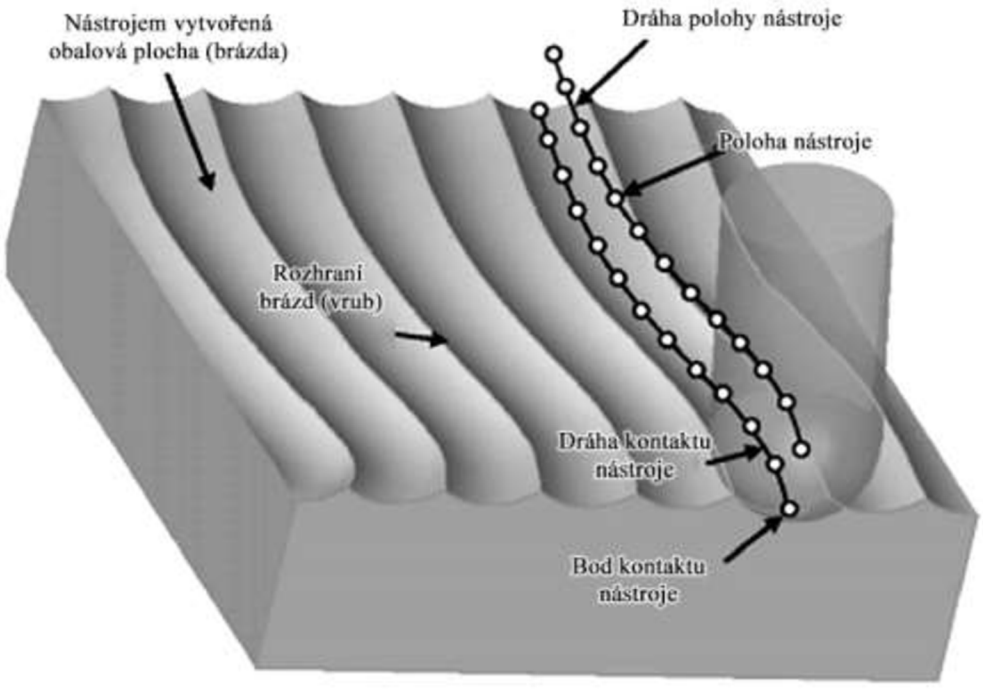
Obr. 3.13 Simulace programu s funkcí hrubování.

3.3.2 Čtvrtá varianta

Zcela odlišnou metodou byla naprogramovaná čtvrtá varianta, kdy stěžejní myšlenkou je deklarace všech souřadnic pod vlastní tělo programu, a souřadnice jsou volány, když jsou potřeba. Řídících bodů je pro danou geometrii 65*54 a každý bod má tak svojí polohu *i* a *j* v rovině *X-Y* a poloha bodu tak lze pro rovinu *X-Y* zapsat jako [i,j]. Toho bylo využito a každému bodu tak bylo přiřazeno unikátní číslo bloku N, např. N1632, odpovídající poloze [16,32]. Díky tomuto kroku lze v těle programu značně parametrizovat technologické operace, jako např. změnu směru průjezdů; hrubovací cyklus, který si bere pouze každou *n*-tou souřadnici v ose *X* a *m*-tou v ose *Y*; počátek a konec křivek; pochopitelně změna G[1] skupiny; apod.

V hlavní části programu je především řídící struktura vypočítávající další bod, který je potřeba. Program není plně odlazen a není tak zcela správně funkční. Pro odladění by bylo nejprve vhodné R parametry zaměnit za LUD, připsat generaci debugovacího logu a podobně jako u programu Vačka vypsat vypočítané hodnoty do externího souboru, aby online běh nebyl zdržován výpočty.

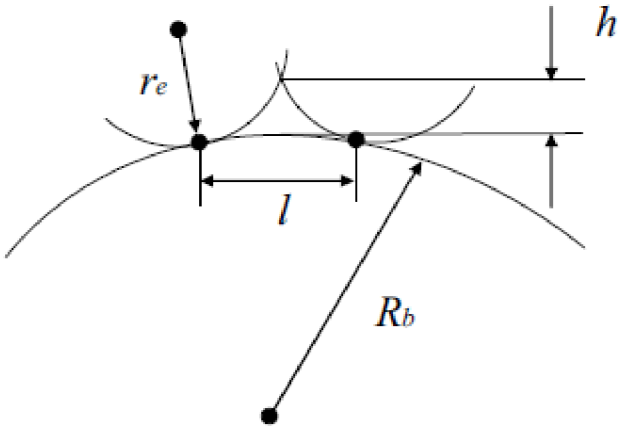
Program by šlo dále alternovat zápisem souřadnic do globální proměnné typu pole o velikosti [m,4,2], kde m by byl počet prvků, o sloupcích *X,Y,Z,W* a případně další dimenze pro *XH, YH, ZH*. Řídící body by bylo také možné číst z externího souboru pomocí funkce READ. A na tomto základu by bylo také možné naprogramovat zmíněnou trasu nástroje přes konstantní Z, zmíněným algoritmem.



Obr. 3.14 Ilustrace brázd po nástroji a vrubů mezi nimi [27].

Na otestování naprogramovaného kódu bylo opět použito fakultní pětiosé frézovací centrum, fréza s kulovým čelem o průměru 16,

Výsledný povrch je dle očekávání špatný v rovině kolmé na průjezdy. Dochází zde ke vrubům, které jsou funkcí poloměru povrchu, nástroje a vzdálenosti dalšího průjezdu (viz Obr. 3.15). Tato chyba je tedy odstranitelná změnou geometrie (obvykle nemožné), nástrojem s větším rádiusem (problematické pro konkávní plochy), nebo zvýšením počtu průjezdů (obvykle nejschůdnější). Ve vyrobené ploše nebyl počet průjezdů zvýšen jednak s ohledem na výrobní čas a jednak kvůli nutnosti jiného pre-processingu. V programu totiž byly použity všechny body vygenerované exportem geometrie z programu CATIA do formátu IGES, kterých bylo v ose Y pouze 52.



Obr. 3.15 Výška vrubu *h* (*scallop* nebo *cusp*) je závislá na poloměru nástroje r_e, poloměru povrchu R_b a vzdálenosti dálšího průjezdu *l* [22].

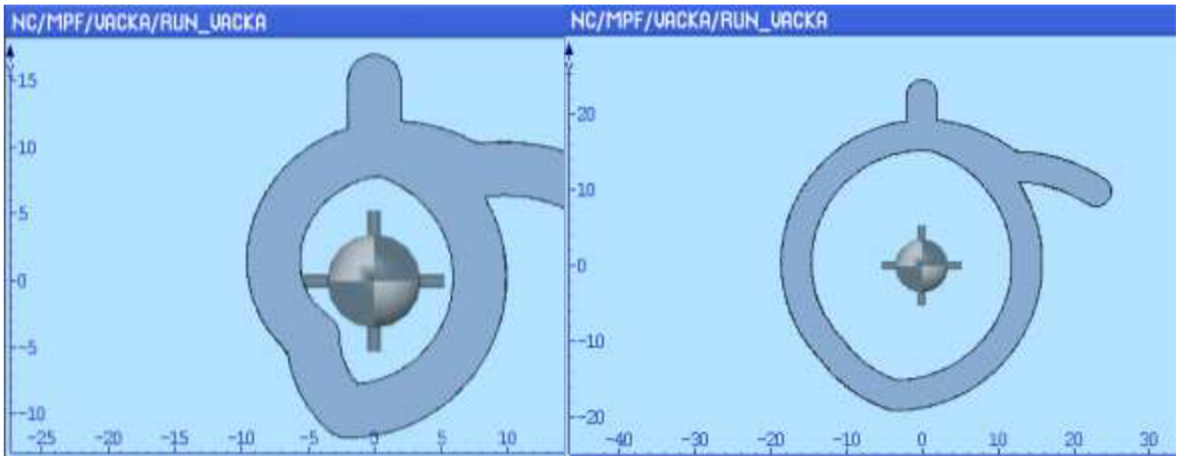
Větším problémem jsou na první pohled méně viditelné chyby v rovině průjezdů. Na většině míst je povrch hladký a spojitý, ale v nejkomplexnějších místech došlo k nesrovnalostem oproti modelu. Vina je jednak na straně korekcí geometrie nástroje, které byly špatně řešeny, a jednak na špatně zvoleném výrobním programu z výše uvedených, kdy byl pro výrobu zvolen kód s dopočítanými body v ose X. Tyto body v ose X nejsou potřeba, jelikož je v tomto směru geometrie s lepším výsledkem interpolována. Naopak měla být strategie zdvojnásobení bodů použita pro zvýšení počtu průjezdů. To se ovšem jedná o chybu lidské složky, pro další výrobek lehce odstranitelnou.

Obr. 3.16 Kompletní obrobek se zaměřením na 3D plochu.
Na fotografii jsou dobře patrné vruby mezi hrubými průjezdy.

3.4 Parametrické obrábění obecných křivek - program Vačka

Vačky slouží k transformaci rotačního pohybu na translační. Jsou definovány obecnou křivkou, kterou lze parametrizovat. Řídící body křivky, kterými lze proložit NURBS křivku, jsou vypočítávány na základě zadané geometrie. Lze tak hovořit o parametrickém obrábění obecné křivky. Protože by ale výpočet mohl trvat déle, a protože lze parametrizací vytvořit cyklus pro výrobu většiny tvarů vaček, byl použit nápad pana Polzera, Ph.D., popsaný výše. Byl tak vytvořen hlavní program, se zadáním geometrie, podprogram/cyklus s výpočty, tvořící program se souřadnicemi pro běh online a log pro debugging. Příklady různých geometrií vypočítaných dle jiných zadaných parametrů geometrie jsou na Obr. 3.17.

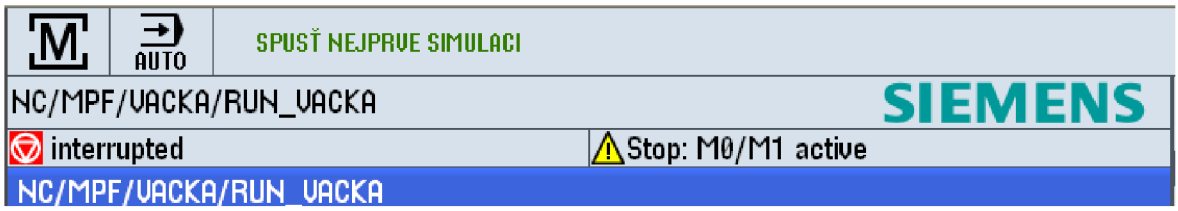
Použité techniky programování: řídící struktury IF, WHILE, FOR, CASE; globální proměnná GUD definovaná jako pole; definice LUD a jejich předávání; systémové proměnné; zápis do souboru a mazání.



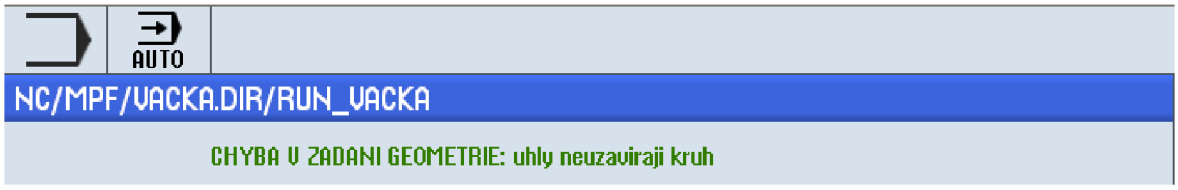
Obr. 3.17 Příklady různých geometrií vytvořených stejným cyklem.

3.4.1 Hlavní program

Hlavní program je schopen se rozhodnout, zda je soubor s geometrií vytvořen (Obr. 3.19) a použít ho, nebo ho nejprve vytvořit; otestovat, zda se úhly v geometrii rovnají 360 (Obr. 3.18); a vypsat na obrazovku podstatné milníky.



Obr. 3.19 Přerušení, pokud operátor zadal příkaz EXECUTE pro odeslání programu do zpracování a nebyla simulací vytvořena geometrie.



Obr. 3.18 Test na úplnost zadání geometrie s ohledem na celkový součet úhlů.

Program lze snadno alterovat, změnou proměnných, aby se změnila geometrie, přesnost, nebo technologické parametry. Změnou kódu, aby např. hlavní program vždy vygeneroval nový spouštěcí program s geometrií.

Původně bylo pro rozhodnutí, zda je geometrie vytvořena, použita systémová proměnná $P_ISTEST. Od toho bylo upuštěno, protože v simulaci spouštěnou z programovacího okna Sinumerik nezapíše do souboru vypočítané hodnoty. Místo toho je použita funkce ISFILE, která testuje, zda je program v parametru vytvořen, či nikoliv.

I přesto, že jsou některé ukončovací příkazy uvedeny v generovaném souboru geometrie, jsou ponechány záměrně i v řídícím programu. Je to z toho důvodu, že na místě nezpůsobují problém, ale pokud by byla například jistá geometrie přidána do hlavního programu, je stroj správně nastaven. Naopak není potřeba přidávat modální funkce platné před voláním podprogramu, toho je dosaženo pomocí slova SAVE v prvním řádku (PROC) podprogramu.

3.4.2 Podprogram / cyklus

Podprogram/cyklus s výpočty umí převzít proměnné a definuje vlastní; pokud dojde k chybě v zápisu do generovaného programu vypsat srozumitelně důvod neúspěchu (Obr. 3.20); převzít adresy g-skupin, feedu a speedu z hlavního programu, generuje lineární interpolaci po krátkých úsecích aproximujíc tak požadovanou křivku a pokud není změna v rádiusu, použije program interpolaci kruhovou.

Program lze snadno alterovat změnou kódu, aby: aby se do logu vypisovalo jen to, co programátor vyžaduje, odkomentováním nebyl program závislý na hlavním programu, generované soubory se ukládaly jinam a s jiným obsahem.

Obr. 3.20 Výpis chyby zápisu i s vysvětlením pro případ ERROR=1.

Vytvořený program geometrie lze snadno upravit tak, aby nebylo k jeho spuštění potřeba hlavního programu, na což jsou odkomentovány jednak pozice pro výběr nástroje a jednak pro cykly, např. 800 a 832, které je nutné tvořit s ohledem na řídící jednotku stroje.

Proměnným byly ponechány dlouhé názvy, jelikož tak lze lehce zjistit, co daný blok počítá a jednak se kód nepoužívá pro samotné obrábění.

Jako počátek funkce FOR byla zvolena nula, protože je pro naší kulturu přirozenější počítání od jedné, a jednak bylo potřeba zohlednit počáteční úhel. Tedy, na prvním řádku pole v proměnné VAC_POLE je pouze výchozí poloměr a 1. prvek geometrie začíná na řádku 1.

Programy jsou uvedeny v příloze 7.

Výroba

Samotná výroba proběhla hladce, z hlediska výroby není kontura vačky při opatrných podmínkách výroby jednoho kusu náročnou operací. Kontura byla obrobena do stěny kvádru do hloubky 4 mm, na obrobku jsou tak vidět dráhy ke kontuře a od ní.

Stejně tak je patrná v dolní části změna kruhové interpolace na spirálu, což by pro použití nejspíše nebylo chtěné a bylo by potřeba zadat lépe vypočítané geometrické podmínky.

Geometrie vačky byla zadána evidentně tak, že smysl otáčení by byl určen zdvihátkem, které by na skokové změně padalo dolů. I přesto by bylo možná vhodné upravit zadanou geometrii tak, aby ani na tomto cíleném skoku nebyla natolik ostrá hrana.

Je potřeba podotknout, že na výsledném tvaru je požadovaným tvarem pouze vnitřní geometrie tvořící vnější konturu vačky. Výroba do stěny plného bloku má smysl pouze pro znázornění cesty nástroje.

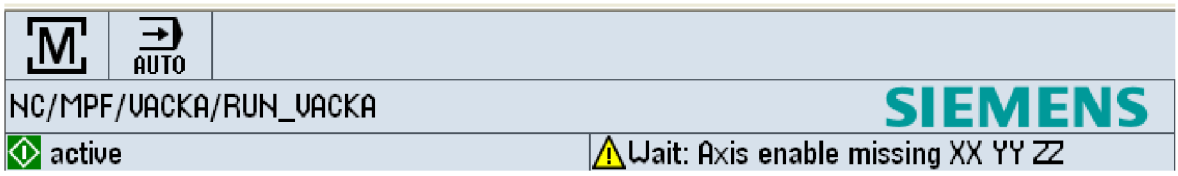
Obr. 3.21 Kontura vačky, v levé části náběh a výběh z kontury, v pravé horní části viditelný přechod z kruhové interpolace nahoře a spirály vpravo a dole.

3.5 Některé problémy a jejich řešení

Práce s většinou programovacích kódů je spojena vysokou dávkou snahy eliminovat chyby, odlaďování a vylepšování funkčnosti. Kódování pro Sinumerik dělají těžší zvláště tyto dva faktory: manuály jsou pro méně obvyklé funkce a především systémové proměnné a chyby nedostatečně popsány; a zadruhé je silně závislý na dalších navázaných systémech, především na konkrétním stroji. Následuje popis vybraných komplikací, které nastaly a stojí z jistých důvodů za zmínku.

Axis enable missing XX YY ZZ

Virtualizovaný stroj se systémem Sinumerik hlásí při spuštění programu, nebo simulace z hlavního okna HMI „Machine“ chybu „Wait: Axis enable missing XX YY ZZ“. Řešení v manuálu chyb výraz neobsahuje, a na webových stránkách jsou návrhy v podobě kontroly kanálů, nastavení stejných parametrů 36970 až 36978, kontrola SBH, zapojení kabelů, oprava modulu Simodrive a další kontroly nastavení.



Obr. 3.22 Chybové hlášení "Axis enable missing XX YY ZZ".

Řešení problému se ale ukázalo jako banální, i přes to, že je stroj pouze virtualizovaný a je spouštěna simulace, je potřeba na řídícím panelu spustit otáčky a posuv.

Chyba nepředání nástroje

Pro předání nástroje lze u strojů bez tool managementu volat systémovou proměnnou $P_TOOLNO. Pro stroje s tool managementem nikoliv, je dle manuálu nutné použít předdefinovanou proceduru GETEXET a místo $P_TOOLP GETSELT. Ty jsou ale popsány vágně bez příkladů použití syntaxe a procedury se tak nepodařilo uvést v chod a použít.

Definice EXTERN a DEF

Pro EXTERN i DEF platí, že je nutné zadat je na začátku programu a nelze je přerušit ani vložením hodnoty do proměnné, pokud nejde o definici proměnné i s přiřazením hodnoty.

Cykly

S cykly nastaly 4 komplikace. Jednak není možné je přímo zadat do příkazu WRITE, pokud obsahují uvozovky. Jednak jsou cykly obecně rozdílné pro různé verze systémů, je tedy při psaní programu nutné počítat s konkrétním strojem, na kterém bude kód aplikován. Třetím problémem byla nepovolená opce na zvolené frézce pro cyklus 832, což šlo obejít ručním vepsáním slov, které by cyklus vyvolal (SOFT, DYNFINISH, COMPCAD, G645, FIFOCTRL, FFWON, případně TRAORI, UPATH, ORIAXES, ORIWKS, ORISON). A poslední hlavní komplikací, na kterou nebylo nalezeno řešení, je tvorba cyklů v SinuTrain pro Sinumerik OPERATE V4.5 Ed.2, kde na rozdíl od SinuTrain 7.5 Ed.2 nejsou soubory cov.com a uc.com a napsaný cyklus vačky tak byl spouštěn pouze jako podprogram.

Téměř vše v řídící logice výpočtu vačky

I přes krátký kód nebylo jednoduché přesně nastavit řídící logiku cyklu výpočtu vačky. Vhodně zvolené proměnné, inkrementy a řídící funkce. Hlavním pomocníkem při řešení byl výpis nejen vypočítaných, ale i počítaných hodnot do logu.

Adresace skupiny kódů

Až při první simulaci programu na frézce se projevilo selhání lidské složky, kdy byla původně v cyklu generujícím geometrii špatně adresována skupina kódů 1. Kvůli absenci předpisu kódu z této skupiny byla pro generovaný program volána hodnota G[1]=1. Číslování začíná bohužel od jedné a tento předpis tedy znamená G0, G1 je druhá ve skupině a má tedy předpis G[1]=2. Rychlou opravou bylo předepsání funkce BSPLINE nebo G1 před generovaný kód, bez pohybu os. Problém lze ale také snadno vyřešit předáním hodnoty z hlavního programu a dovolit tak operátorovi volbu typu pohybu, kde by bylo zároveň vhodné omezit výběr testem na G1 a spline interpolace.

Přeběhy na 3D ploše – viz kapitola 3.1

4 DISKUZE VÝSLEDKŮ

4.1 Diskuze k praktické části

Při samotné výrobě na fyzickém stroji nedošlo k vážným, nebo neočekávaným chybám, jelikož SinuTrain umožňuje simulaci běhu programu a kritické chyby, které nejsou při prvotním napsání kódu neobvyklé, lze podchytit.

U programu obecné 3D plochy by bylo vhodné porovnat, jaký by nastal rozdíl mezi plochou vytvořenou pomocí spline křivek a pomocí komprese lineárních elementů.

4.1.1 Převod z CAD systému do NC kódu

Pro praktickou část bylo zvoleno simulovat si absenci CAM systému a exportovat geometrii do IGES souboru. Regulární výrazy se prokázaly jako velmi vhodné řešení semi-automatického převodu, kdy odstranily dlouhou rutinní práci.

Na druhou stranu makro z nich vytvořené není ideální řešení, protože s každou změnou požadavků je nutné odladit funkčnost regulárních výrazů a editovat specifický soubor s makry. A při stále se měnících požadavcích se ukázalo rychlejší mít základní regulární výrazy makra poznačené bokem a po kýžené alteraci je do ‚hledat a nahradit' dosazovat.

Pro univerzálnější metodu by bylo potřeba napsat samostatný program.

4.1.2 Programy obecné 3D plochy

Výroba obecné 3D plochy proběhla na úroveň semi-roughingu. Pro dokončující operaci by bylo potřeba zvolit $f_z \approx a_e$, aby se dosáhla hladká a symetrická struktura ve všech směrech, kterou tak lze snadno vyleštit nehledě na zvolenou leštící metodu, viz Obr. 4.1. Pro výrobu s trasou nástroje po izo-parametrických křivkách by tedy bylo potřeba zvýšit počet řídících bodů v příslušné ose [23].

Obr. 4.1 Pro semi-roughing (vlevo) se používá $f_z << a_e$.
Pro dokončovací operaci (finishing) $f_z \approx a_e$ [23].

Primárním problémem u obecných ploch a tedy i v tomto programu je natočení nástroje a jeho geometrická korekce. Natočení bylo v jednotlivých programech realizováno různě, v jednom dokonce proměnlivě vzhledem ke směru průjezdu za pomocí výpočtů v MS Excel, a tilt byl volen konstantní. Nicméně korekce rádiusu za pomocí funkce CUT3DF nefungovala, nebo byla špatně aplikována. Po volené geometrii tak jel střed nástroje místo bodu dotyku. Proto jsou např. na vrcholu obrobku viditelné ostré hrany.

Na natočení nástroje i kompenzaci rádiusu byly navrženy metody, které jsou aplikovatelné pomocí metody zápisu výsledků do externího programu.

Stejně tak byla v posledním programu obecné 3D plochy rozpracována metoda automatizace výroby ploch zadaných řídícími body. Po importu řídících bodů do globální proměnné se umí program rozhodnout, kterými body proloží další průjezd nástroje.

4.1.3 Program s parametrizovanou geometrií křivky

Ve třetí části bylo hojně využito možností parametrického programování v Sinumeriku, kdy po zadání geometrie podprogram/cyklus spočítá geometrii a exportuje výsledek do zvláštního souboru, který je teprve použit na samotné obrábění. Bylo tak dosáhnuto eliminace výpočetního času při běhu obrábění i přes komplexní rozhodovací strukturu.

4.2 Diskuse k použitým metodám

Parametrické programování je silnou součástí programování v G-kódu. Jeho využití je však komplikované a je potřeba velkého vhledu do problematiky na jeho použití pro obrobení skutečně obecných ploch. Jeho použití má především smysl pro technologické operace, nebo geometrie se známým matematickým předpisem, což obecné plochy téměř nikdy nebývají.

Mnozí zahraniční autoři se zcela vyhýbají řešení skrze programování v G-kódu a v odborné komunitě převládá snaha především zdokonalit CAM systémy a řídící jednotky. Kromě mnoha odborných publikací na toto téma už jsou delší dobu viditelné také výsledky, jako v práci několikrát zmíněné možnosti exportu spline křivek přímo z CAM systému, nebo proložení spline křivek lineárními elementy kompresoru řídící jednotky.

Nehledě na výše uvedené bylo předvedeno, že parametrickým programováním lze dosáhnout obstojné komplexnosti a v případě očekávané geometrie CAM systém zcela obejít.

Na druhou stranu, původně zamýšlená varianta naprogramování výpočtu NURBS plochy se ukázala jako slepá ulička a bylo od ní opuštěno. Proto také chybí detailnější popis matematického aparátu a je pouze využívána funkce BSPLINE pro interpolaci Sinumerikem.

Funkce Sinumeriku ASPLINE, CSPLINE a POLY nebyly využívány, jelikož první dvě jsou metody interpolační, nikoliv aproximační a pro daný případ byly nevhodné a řízení pomocí polynomů, kterému se věnuje např. Ohnišťová ve své bakalářské práci [16] bylo zavrhnuto z jednak z důvodů, ke kterým sama dospěla:

„Sestavení matematické definice je však časově velmi náročné…“ [16]

tedy pro časovou náročnost vyřazující použití metody v praxi pro opravdu komplexní plochy, a jednak pro nepřesvědčivé výsledky aplikace.

Jak bylo ukázáno v kapitole 2.2 o parametrizaci programů, parametrizovat skutečně obecné plochy je jen extrémně obtížně řešitelné a často postrádá smysl. Jakmile je však znám např. matematický předpis křivek (nábojnice, vačka, atd.) a mění se pouze dopředu očekávané rozměry, je parametrizace snadná.

ZÁVĚRY

V práci je především rozbor postupů tvorby součástky, jednotlivé přístupy jsou analyzovány. Bylo nastíněno, že pro CNC je nejvýhodnější realizovat hladké pohyby os buďto kompresí lineárních elementů, nebo rovnou zadáním hladkých křivek a k druhé variantě jsou rozpracovány některé metody.

Praktické příklady se do značné míry přiblížili stanovenému cíli, tedy:

„Aplikace parametrického CNC programování na složité tvary výrobku, zejména funkčních tvaru se zvýšenými požadavky kvalitu opracování.“,

ale jsou stále plny možností na vylepšení. Byly však popsány a použity některé funkce pro Sinumerik, patřící do manuálu pro pokročilé a byly tak předvedeny možnosti jejich aplikace.

Byla vyřčen postulát, že generace matematického předpisu pro funkci POLY je natolik náročná, že je pro praxi nepoužitelná a nebyl tak na tuto metodu dále brán zřetel.

Podobně byla nalezena slepá ulička v pokusu napsat cyklus počítající některou z interpolačních, nebo aproximačních metod s tím, že by nebylo dosaženo přínosu oproti implementovaným funkcím řídící jednotky jako je BSPLINE.

Velkým přínosem je myšlenka doktora Polzera řešit složité výpočty cyklem a vypočítanou geometrii zapisovat do souboru na běh obrábění.

Byly navrhnuty metody:

- výpočtu parametrizace obecných ploch zadaných řezy,
- výpočtu dráhy nástroje,
- návrh výpočtu normály k povrchu,
- návrh výpočtu kompenzace rádiusu nástroje.

SEZNAM POUŽITÝCH ZDROJŮ

1. **LYNCH, Mike.** *Parametric Programming for Computer Numerical Control Machine Tools and Touch Probes: CNC's Best-kept Secret.* Dearborn : Society of Manufacturing Engineers, 1997. p. 433. 0-87263-481-7.
2. **Siemens AG.** *SINUMERIK 840D sl / 828D, Pro pokročilé, Programovací příručka.* NÜRNBERG : Siemens AG, 2010. str. 822. Order no.: 6FC5398-2BP20-1UA0.
3. **ASWORTH, David:** Podklady pro výuku předmětu ADMAN. Lyngby: DTU, 2014
4. **MATTSON, Mike.** *CNC programming: principles and applications.* Albany : Delmar, 2002. str. 358. 0-7668-1888-8.
5. **ŠTULPA, Miloslav.** *CNC obráběcí stroje a jejich programování.* 1. vyd. Praha : BEN - technická literatura, 2006. str. 126. 80-7300-207-8.
6. **POLZER, Aleš.** Akademie CNC obrábění (4). *Technický týdeník.* [Online] Business Media CZ, 14. 02 2009. [Citace: 26. 04 2015.] http://www.technickytydenik.cz/rubriky/serialy/akademie-cnc-obrabeni/akademie-cnc-obrabeni-4_8539.html.
7. **MCMAHON Chris, BROWNE Jimmie.** *CADCAM: principles, practise and manufacturing management.* 2nd. Harlow : Prentice Hall, 1998. str. 665. 0-201-17819-2.
8. **GROOVER, Mikell P.** *Automation, Production Systems, and Computer-Integrated Manufacturing.* 3rd ed. Upper Saddle River : Prentice Hall, 2007. str. 815. 978-0132393218.
9. **Siemens AG.** *SINUMERIK 840D/810D/FM-NC Short Guide Programming (PGK).* 10.2000. 2000. Order No. 6FC5298-6CA00-0BG0.
10. **LYNCH, Mike.** How the heck does G28 work? *CNC Concepts, Inc.* [Online] CNC Concepts, Inc., 2007. [Citace: 30. 04 2015.] https://www.cncci.com/resources/tips/how%20g28%20works.htm.
11. **Siemens AG.** *SINUMERIK, Manual, Mold Making, 3 to 5-Axis Simultaneous Milling.* 09/2011. 2011. Order No. 6FC5095-0AB10-0BP2.
12. **PECKA, Miroslav.** Základy regulárních výrazů. *Regulární Výrazy.* [Online] [Citace: 21. 05 2015.] http://www.regularnivyrazy.info/regularni-vyrazy-zaklady.html.
13. **Bémová, Kristýna.** Křivky v počítačové grafice . *MFF UK.* [Online] 13. 12 2007. [Citace: 05. 03 2015.] http://cgg.mff.cuni.cz/~pepca/ref/krivkyBemova.pdf.
14. **ÚM FSI VUT.** KŘIVKY DANÉ PARAMETRICKY A POLÁRNĚ. *MATEMATIKA ONLINE.* [Online] 2005. [Citace: 23. 03 2015.] Podklady k předmětu 1M - Matematika I. http://mathonline.fme.vutbr.cz/download.aspx?id_file=919.
15. **Alexandr, Lubomír.** *Diplomová práce, Výuka počítačové grafiky cestou WWW.* 1999.
16. **OHNIŠŤOVÁ, Petra.** *Pětiosé frézování s polynomiální trajektorií: Bakalářská práce.* Brno : Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. str. 50. bez příloh., vedoucí práce prof. Ing. Miroslav Píška, CSc.
17. CNCCookbook. *3D CAM Toolpath Fundamentals.* [Online] CNCCookbook, Inc. [Citace: 29. 04 2015.] http://www.cnccookbook.com/CC3DCAMToolpaths.html.
18. Parametr. *Wikipedie.* [Online] 19. 08 2014. [Citace: 13. 05 2015.] https://cs.wikipedia.org/wiki/Parametr.
19. **POLZER, ALEŠ**. Výpočet drah v simulaci [konzultace]. 22.5.2015.
20. **IGLESIAS, Andrés.** Visualización e Interacción Gráfica (2009): Material de clase 2. *University of Cantabria.* [Online] 22. 08 2011. [Citace: 2015. 4 17.] Materiály k výuce,

licence Creative Commons 4.0. http://ocw.unican.es/ensenanzas-tecnicas/visualizacion-e-interaccion-grafica/material-de-clase-2. ISBN: 978-84-693-4865-9.
21. **Technická univerzita v Liberci.** Formát IGES. *Univerzitní e-learningový systém.* [Online] [Citace: 12. 05 2015.] http://blade1.ft.tul.cz/elearning/Xslt/publ/36/36_418.pdf.
22. *A 5-axis Milling System Based on a New G code for NURBS Surface.* **LIANG, Hongbin a LI, Xia.** místo neznámé : IEEE, 20-22. Nov. 2009, Proceedings - 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009, stránky 600-606. ISBN: 978-1-4244-4738-1.
23. **SANDVIK Coromat.** Generation of sculptured surfaces. *SANDVIK Coromat.* [Online] 01 2013. [Citace: 18. 04 2015.] http://www.sandvik.coromant.com/en-gb/knowledge/milling/application_overview/profile_milling/generation_of_sculptured_surfaces/pages/generation-of-sculptured-surfaces.aspx.
24. **HRABAL, J.** Možnosti systému Sinumerik 840D při programování obráběcích strojů. Brno : Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2011. str. 65. Vedoucí diplomové práce Ing. Aleš Polzer, Ph.D..
25. **BĚLÍK, Milan, Ph.D., Ing.** KEE/POE - Počítače v energetice. *Fakulta elektrotechnická ZČU v Plzni.* [Online] 14. 10 2010. [Citace: 01. 05 2015.] Podklady pro výuku předmětu KEE/POE. https://home.pilsfree.net/fantom/FEL/POE/Poe06.doc.
26. **ZOTTIE, Wikipedia user.** Demo of constructive solid geometry tree. Created in POV-Ray. 10. August 2005. licence Creative Commons Attribution-Share Alike 3.0 Unported.
27. **CHEN, Zezhong C. a Wei CAI.** An Efficient, Accurate Approach to Representing Cutter-Swept Envelopes and Its Applications to Three-Axis Virtual Milling of Sculptured Surfaces. *Journal of Manufacturing Science and Engineering.* vol. 130, 2008, issue 3.
28. **CNCCOOKBOOK, INC.** CNC Machine Overview and Computer Numerical Control History. *CNCCookbook: Making CNC Faster and Easier.* [Online] CNCCookbook, Inc. [Citace: 06. 05 2015.] http://www.cnccookbook.com/CCCNCMachine.htm.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Zkratka	Jednotka	Popis
APT	[-]	Automatically Programmed Tool, vysoký programovací jazyk pro instrukce NC strojů
CAD	[-]	Computer Aided Design, tj.modelování geometrie součásti s pomocí PC
CAM	[-]	Computer Aided Manufacturing, tj.modelování obráběcího procesu s pomocí PC
CNC	[-]	Computer Numerical Control, tj.číslicové řízení s pomocí PC
CLDATA	[-]	Cutter Location data, tj.údaje o poloze nástroje
HW	[-]	Hardware, fyzické zařízení
IGES	[-]	Univerzální formát souboru pro přenos geometrie
NC	[-]	Numerical Control, tj. číslicové řízení
NURBS	[-]	Non-uniform rational B-Spline křivky a plochy
SW	[-]	Software, instrukce ovládající HW

Symbol	Jednotka	Popis
A, B, C	[-]	rotační osy obráběcího stroje
F	[mm/min]	posuvová rychlost
S	[min^{-1}]	otáčky vřetene
W	[-]	váha bodu NURBS křivek a ploch
X, Y, Z	[-]	translační osy obráběcího stroje

SEZNAM PŘÍLOH

PŘÍLOHA 1

Makro nutné připsat do souboru shortcuts, který je defaultně v: C:\Users\[user]\AppData\Roaming\Notepad++\shortcuts.xls

Význam popsán na http://docs.notepad-plus-plus.org/index.php/Editing_Configuration_Files

<Macro name="IGS Splines 2 NC" Ctrl="yes" Alt="yes" Shift="yes" Key="35">

<Action type="3" message="1700" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1601" wParam="0" lParam="0" sParam=" {5,}\w{2,4} {2,7}\d{1,5}" />

<Action type="3" message="1625" wParam="0" lParam="2" sParam="" />

<Action type="3" message="1602" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1653" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1652" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1702" wParam="0" lParam="768" sParam="" />

<Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />

<Action type="3" message="1700" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1601" wParam="0" lParam="0" sParam="\s" />

<Action type="3" message="1625" wParam="0" lParam="2" sParam="" />

<Action type="3" message="1602" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1653" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1652" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1702" wParam="0" lParam="768" sParam="" />

<Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />

<Action type="3" message="1700" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1601" wParam="0" lParam="0" sParam=";128,..,.,.,(.)" />

<Action type="3" message="1625" wParam="0" lParam="2" sParam="" />

<Action type="3" message="1602" wParam="0" lParam="0" sParam="\nBSPLINE SD=$1 F=#9 ;nurbs surf\n" />

<Action type="3" message="1653" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1652" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1702" wParam="0" lParam="768" sParam="" />

<Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />

<Action type="3" message="1700" wParam="0" lParam="0" sParam="" />

<Action type="3" message="1601" wParam="0" lParam="0" sParam=";126,..,(.)" />

<Action type="3" message="1625" wParam="0" lParam="2" sParam="" />

<Action type="3" message="1602" wParam="0" lParam="0" sParam="\nBSPLINE SD=$1 F=#9 ;bspline curve\n" />

<Action type="3" message="1653" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1652" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1702" wParam="0" lParam="768" sParam="" />
<Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />
<Action type="3" message="1700" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1601" wParam="0" lParam="0" sParam="STARTRECORDGOHERE.{1,}\n" />
<Action type="3" message="1625" wParam="0" lParam="2" sParam="" />
<Action type="3" message="1602" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1653" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1652" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1702" wParam="0" lParam="768" sParam="" />
<Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />
<Action type="3" message="1700" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1601" wParam="0" lParam="0" sParam=".{5,}(1.0,){5,}" />
<Action type="3" message="1625" wParam="0" lParam="2" sParam="" />
<Action type="3" message="1602" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1653" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1652" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1702" wParam="0" lParam="768" sParam="" />
<Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />
<Action type="3" message="1700" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1601" wParam="0" lParam="0" sParam="([-]*[0-9]+\.[0-9]{1},){4,5}0,0.*" />
<Action type="3" message="1625" wParam="0" lParam="2" sParam="" />
<Action type="3" message="1602" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1653" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1652" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1702" wParam="0" lParam="768" sParam="" />
<Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />
<Action type="3" message="1700" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1601" wParam="0" lParam="0" sParam="([+-]*[0-9]+.[0-9]+),([+-]*[0-9]+.[0-9]+),([+-]*[0-9]+.[0-9]+)," />
<Action type="3" message="1625" wParam="0" lParam="2" sParam="" />
<Action type="3" message="1602" wParam="0" lParam="0" sParam="X$1 Y$2 Z$3\n" />
<Action type="3" message="1653" wParam="0" lParam="0" sParam="" />
<Action type="3" message="1652" wParam="0" lParam="0" sParam="" />

```
        <Action type="3" message="1702" wParam="0" lParam="768" sParam="" />
        <Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />
        <Action type="3" message="1700" wParam="0" lParam="0" sParam="" />
        <Action type="3" message="1601" wParam="0" lParam="0" sParam="((.{1,}\n{1,}){1,})" />
        <Action type="3" message="1625" wParam="0" lParam="2" sParam="" />
        <Action type="3" message="1602" wParam="0" lParam="0" sParam="$1\n" />
        <Action type="3" message="1653" wParam="0" lParam="0" sParam="" />
        <Action type="3" message="1652" wParam="0" lParam="0" sParam="" />
        <Action type="3" message="1702" wParam="0" lParam="768" sParam="" />
        <Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />
        <Action type="3" message="1700" wParam="0" lParam="0" sParam="" />
        <Action type="3" message="1601" wParam="0" lParam="0" sParam="\n[^BX]" />
        <Action type="3" message="1625" wParam="0" lParam="2" sParam="" />
        <Action type="3" message="1602" wParam="0" lParam="0" sParam="\nG0\nM17\n\n\n" />
        <Action type="3" message="1653" wParam="0" lParam="0" sParam="" />
        <Action type="3" message="1652" wParam="0" lParam="0" sParam="" />
        <Action type="3" message="1702" wParam="0" lParam="768" sParam="" />
        <Action type="3" message="1701" wParam="0" lParam="1609" sParam="" />
</Macro>
```

PŘÍLOHA 2

```
WORKPIECE(,"",,"BOX",0,0,-60,0,0,0,140,30)
CYCLE800(1,"0",100000,57,0,0,0,0,0,0,0,0,0,-1,100,1)
G54 G710 G90
R1=0.18
ATRANS X=55 Y=4
ASCALE X=R1 Y=R1 Z=R1
T="BALL nose end mill 4"
M06 S2000 F600 M03 M08
TRAORI
TOROT
SOFT
DYNFINISH
FIFOCTRL
CUT3DC
G54
G0 X142.4457397 Y123.0827484 Z5 A3=1 B3=-1 C3=3
G1 Z-4
BSPLINE ;Sk
X142.4457397 Y123.0827484 Z-4
X138.5227268 Y122.1403168 Z-4
X134.6282361 Y121.1036846 Z-4
…
X216.4805343 Y43.79412594 Z-4
X217.7135925 Y43.87189484 Z-4
G0 Z1
X249.1038649 Y54.55160017
BSPLINE ;a
X249.1038649 Y54.55160017 Z-4
X248.2016551 Y54.96107949 Z-4
…
X264.1554532 Y49.12888972 Z1
G0 Z5
X257.0213318 Y93.88116455
BSPLINE ;aa
X257.0213318 Y93.88116455 Z0
```

```
…
X256.0624084 Y80.95957947 Z1
G0 Z5
X299.5344153 Y57.43506619
BSPLINE ;cel
X299.5344153 Y57.43506619 Z-4
…
X367.2463294 Y5.84736249 Z0
G0 Z10
ASCALE X0.75 Y0.75
ATRANS X-300
X17.53468103 Y176.0408353
BSPLINE ;Ja
X17.53468103 Y176.0408353 Z-4
…
X152.8999459 Y86.33217894 Z-4
G0 Z5
X180.2632751 Y112.7455444
BSPLINE ;n
X180.2632751 Y112.7455444 Z-4
X218.7722778 Y97.35520172 Z-4
G0 Z10
X367.2463294 Y10
TCARR=0
PAROTOF
TOROTOF
M09 M05
M30
```

PŘÍLOHA 3

```
WORKPIECE(,"",,"RECTANGLE",0,0,-60,0,62.9,61.6)
G54 G71 G90
G0 X200 Y200 Z200
CYCLE800(1,"0",100000,57,0,0,0,0,0,0,0,0,-1,,1)
T="BALL NOSE 15"
M6 F1000 S2000 M8
SCALE X=0.616 Y=0.629 Z=0.7
CYCLE832(0.05,_FINISH,1)
TRAORI TOROT FFWON
FIFOCTRL CUT3DF ORIWKS
OTOL=0.5
G54

G0 X-70 Y-50 A0 C0 M3
G1 Z-30
X-50

BSPLINE
X-50.0 Y-50.0 Z-30.00016717 A3=1 B3=2 C3=6
X-48.73601782 Y-50.0 Z-29.99949295
X-47.47203563 Y-50.0 Z-29.99881873
X-46.20805344 Y-50.0 Z-30.00032511
X-44.94407125 Y-50.0 Z-30.00183149
…
X48.49441232 Y-50.0 Z-29.99945686
X50.0 Y-50.0 Z-30.00018586
G1 X70
G0 Z10
X-70 Y-47.22696548
G1 Z-29.99924165
G1 X-50

BSPLINE
X-50.0 Y-47.22696548 Z-29.99924165
X-48.73601782 Y-47.22696548 Z-29.83377100
```

```
X-47.47203563 Y-47.22696548 Z-29.66830035
…
X48.49441232 Y-47.22696548 Z-29.74944554
X50.0 Y-47.22696548 Z-29.99884599
G1 X70
G0 Z10
X-70 Y-44.45393097
G1 Z-30.00100907
G1 X-50
BSPLINE
X-50.0 Y-44.45393097 Z-30.00100907
X-48.73601782 Y-44.45393097 Z-29.67259687
…
…
X48.49441232 Y50.0 Z-29.99943130
X50.0 Y50.0 Z-30.00019193
G1 X70
SUPA
G0 Z200
FFWOF
TOROTOF
TRAFOOF
M9 M5
M17
```

PŘÍLOHA 4

```
WORKPIECE(,"",,"RECTANGLE",0,0,-60,0,62.9,61.6)
G54 G71 G90
G0 X200 Y200 Z200
CYCLE800(1,"0",100000,57,0,0,0,0,0,0,0,0,-1,,1)
T="CUTTER 4"
M6 F1000 S2000 M8
CYCLE832(0.05,_FINISH,1)
TRAORI
TOROT
FFWON
FIFOCTRL
CUT3DF
ORIWKS
OTOL=0.5
G54
G0 X-50 Y-31.45 A0 C0 M3
G1 Z-30
X-30
BSPLINE
X-30.80 Y-31.450 Z-30.0001 A2=12.5 B2=0.0191
X-30.80 Y-29.7057 Z-29.9992 A2=12.5 B2=-0.0365
X-30.80 Y-27.9615 Z-30.0010 A2=12.5 B2=0.0304
X-30.80 Y-26.2172 Z-29.9995 A2=12.5 B2=-0.0090
…
…
X-4.1021 Y28.1937 Z-24.1621 A2=12.5 B2=-47.5049
X-4.1021 Y29.8218 Z-26.9874 A2=12.5 B2=-49.3302
X-4.1021 Y31.45 Z-29.9999

G1 Y45
G0 Z5
Y-45
G1 A2=12.5 B2=40.4887
X-3.8527 Z-30.0000
Y-31.450
```

BSPLINE

X-3.8527 Y-31.450 Z-30.0000 A2=12.5 B2=40.4887

X-3.8527 Y-29.7057 Z-27.6326 A2=12.5 B2=38.7731

X-3.8527 Y-27.9615 Z-25.4051 A2=12.5 B2=36.9901

X-3.8527 Y-26.2172 Z-23.3162 A2=12.5 B2=34.8570

X-3.8527 Y-22.7564 Z-19.4841 A2=12.5 B2=30.5501

X-3.8527 Y-19.2957 Z-16.2366 A2=12.5 B2=28.0274

X-3.8527 Y-17.5791 Z-14.7839 A2=12.5 B2=26.8642

X-3.8527 Y-15.2384 Z-12.8989 A2=12.5 B2=26.3499

X-3.8527 Y-12.8976 Z-11.0556 A2=12.5 B2=26.0192

X-3.8527 Y-12.2735 Z-10.5712 A2=12.5 B2=25.5876

X-3.8527 Y-11.0441 Z-9.6353 A2=12.5 B2=22.6186

X-3.8527 Y-9.8147 Z-8.8210 A2=12.5 B2=19.9369

X-3.8527 Y-9.2094 Z-8.4719 A2=12.5 B2=27.6829

X-3.8527 Y-8.3769 Z-7.7776 A2=12.5 B2=18.9003

X-3.8527 Y-7.5445 Z-7.3244 A2=12.5 B2=14.3385

X-3.8527 Y-7.3172 Z-7.2321 A2=12.5 B2=6.3609

X-3.8527 Y-6.8199 Z-7.1440 A2=12.5 B2=-11.0356

X-3.8527 Y-6.3226 Z-7.2981 A2=12.5 B2=-19.8365

X-3.8527 Y-6.0526 Z-7.4530 A2=12.5 B2=-22.7919

X-3.8527 Y-5.4313 Z-7.8681 A2=12.5 B2=-19.9788

X-3.8527 Y-4.8100 Z-8.2271 A2=12.5 B2=-10.4382

X-3.8527 Y-4.4587 Z-8.3300 A2=12.5 B2=-0.1605

X-3.8527 Y-3.4338 Z-8.3346 A2=12.5 B2=16.5966

X-3.8527 Y-2.4089 Z-7.8490 A2=12.5 B2=6.1831

X-3.8527 Y-1.7353 Z-7.7329 A2=12.5 B2=-3.9676

X-3.8527 Y-0.4793 Z-7.8714 A2=12.5 B2=-9.4753

X-3.8527 Y0.7766 Z-8.2047 A2=12.5 B2=-9.1723

X-3.8527 Y1.3590 Z-8.3542 A2=12.5 B2=-6.3894

X-3.8527 Y2.1077 Z-8.4875 A2=12.5 B2=-3.8417

X-3.8527 Y2.8565 Z-8.5674 A2=12.5 B2=-3.2172

X-3.8527 Y3.0228 Z-8.5823 A2=12.5 B2=-2.5695

X-3.8527 Y3.3152 Z-8.6032 A2=12.5 B2=-1.6798

X-3.8527 Y3.6076 Z-8.6168 A2=12.5 B2=-1.3871

X-3.8527 Y3.7336 Z-8.6217 A2=12.5 B2=-1.2134

X-3.8527 Y3.9674 Z-8.6295 A2=12.5 B2=-0.8882

```
X-3.8527 Y4.2012 Z-8.6353 A2=12.5 B2=-0.7374
X-3.8527 Y4.3089 Z-8.6375 A2=12.5 B2=-0.5349
X-3.8527 Y4.6324 Z-8.6423 A2=12.5 B2=0.0613
X-3.8527 Y4.9560 Z-8.6417 A2=12.5 B2=0.4637
X-3.8527 Y5.1718 Z-8.6390 A2=12.5 B2=0.7499
X-3.8527 Y6.2333 Z-8.6169 A2=12.5 B2=2.2402
X-3.8527 Y7.2949 Z-8.5509 A2=12.5 B2=3.3257
X-3.8527 Y8.1407 Z-8.4727 A2=12.5 B2=3.7412
X-3.8527 Y10.0275 Z-8.2766 A2=12.5 B2=0.4083
X-3.8527 Y11.9143 Z-8.2552 A2=12.5 B2=-3.9259
X-3.8527 Y12.9553 Z-8.3688 A2=12.5 B2=-9.9802
X-3.8527 Y15.6507 Z-9.1228 A2=12.5 B2=-24.2925
X-3.8527 Y18.3462 Z-11.0570 A2=12.5 B2=-31.7915
X-3.8527 Y20.0005 Z-12.6873 A2=12.5 B2=-36.4693
X-3.8527 Y23.2831 Z-16.5446 A2=12.5 B2=-43.3365
X-3.8527 Y26.5656 Z-21.4686 A2=12.5 B2=-45.7064
X-3.8527 Y28.1937 Z-24.1217 A2=12.5 B2=-47.6918
X-3.8527 Y29.8218 Z-26.9655 A2=12.5 B2=-49.5346
X-3.8527 Y31.45 Z-29.9999
…
…
X30.80 Y31.450 Z-30.0001 A2=12.5 B2=-30.9639
G1 Y50
SUPA
N120 G0 Z200
TOROTOF TRAFOOF
FFWOF
M9 M5
M17
```

PŘÍLOHA 5

```
CYCLE800(1,"0",100000,57,0,0,0,0,0,0,0,0,0,-1,100,1)
X-60 Y60
FFWON
GROUP_BEGIN(0,"IF HRUBOVANI",0,0)
   IF R1==1
   T="CUTTER 20"
   M6
   Z40 C0 F1000 S2000 M3 M8
   TRANS Z10
   G1 X-50 Y50 Z25
   GOTO N10
   ENDIF
GROUP_END(0,0)
GROUP_BEGIN(0,"IF DOKONCOVANI",0,0)
   TRANS Z0
   T="BALL nose end mill 1"
   M6
   A0 C0 M3 F400 S2000 M8
   CYCLE832(0.5,_FINISH,1)
   TRAORI TOROT FIFOCTRL
   CUT3DC ORIWKS OTOL=0.5
   G54
   G1 X-65 Y50
   Z-30
GROUP_END(0,0)
BSPLINE
X-50.00000000 Y50.00000000 Z-30.00012772 A3=-1 B3=1 C3=4
X-50.00000000 Y47.41156332 Z-30.00006662
…
X-50.00000000 Y-50.0 Z-30.00016717
G0 Z5
N10 X-48.73601782 Y50.00000000
G1 Z-29.99952414
BSPLINE
```

```
X-48.73601782 Y50.00000000 Z-29.99952414
…
X-48.73601782 Y-50.0 Z-29.99949295
G0 Z5
IF R1==1
GOTO N20
ENDIF
X-47.47203563 Y50.00000000
G1 Z-29.99892056
BSPLINE
X-47.47203563 Y50.00000000 Z-29.99892056
…
X-43.68008907 Y-50.0 Z-30.00043948
G0 Z5
N20 X-42.41610688 Y50.00000000
G1 Z-29.99915497
BSPLINE
X-42.41610688 Y50.00000000 Z-29.99915497
…
X-42.41610688 Y-50.0 Z-29.99904746
G0 Z5
IF R1==1
GOTO N30
ENDIF
X-39.86775566 Y50.00000000
G1 Z-29.99995203
BSPLINE
X-39.86775566 Y50.00000000 Z-29.99995203
…
X50.00000000 Y-50.00000000 Z-30.00018586
N120 G0 Z200
TOROTOF TRAFOOF
M17
```

PŘÍLOHA 6

```
;_____________POCATECNI HODNOTY_____________________
R1=1   ;počáteční bod na ose x
R2=1   ;počáteční bod na ose y
;R3     rezervováno pro výpočty(výpočet posunu mezi iso-křivkami)
R4=65  ;i bodu na ose x
R5=54  ;j bodu na ose y
R6=1   ;1=draha s i=const, 2=draha s j=const
R7=1   ;1=G01, 2=BSPLINE
R8=40  ;návratová rovina (osa Z)
;R10-13 rezervováno pro výpočty (ekvivalent R1,2,4,5)
R14=1  ;bodový inkrement ve směru j
R15=1  ;bodový inkrement ve směru i
;R16   rezervováno pro výpočty(výpočet posunu mezi iso-křivkami)
R17=0  ;1=s hrubovanim, else-pouze finis
R18=10  ;násobek R14 pro hrubovani
R19=5  ;násobek R15 pro hrubovani
;_____________________NASTAVENI_____________________________
WORKPIECE(,"",,"RECTANGLE",0,0,-60,0,100,100)
CYCLE800(1,"TC1",0,57,0,0,0,0,0,0,0,0,-1,)
;_____________PREDBEZNE VYPOCTY_____________________
PREDBEZNE:
IF R6==2      ;R10 konstantní osa, R11 proměnná
  R10=R2 R11=R1 R12=R5 R13=R4 ;R6=1
ELSE
  R10=R1 R11=R2 R12=R4 R13=R5
ENDIF
TRAFOOF
G0 G90 G54 G71 A0 C0
X200 Y200 Z200

HRUBOVANI:
IF R17==1
  R14=R14*R18
  R15=R15*R19
  T="CUTTER 32"
```

```
  M6
  TRANS Z15
  Z=R8 S5000 M3 M8
  G1 Z0 F5000
GOTOF N1
ENDIF

FINIS:
TRANS Z0
T="BALL nose end mill 1"
M6
;FFWON SOFT G642 DYNFINISH;odkomentovane prikazy nahrazeny cyklem
CYCLE832(0.5,_FINISH,1)
TRAORI
TOROT
FIFOCTRL
CUT3DCC ;nebo CUT3DC
ORIWKS
G54
Z=R8 S2000 M3 M8
G1 Z0 F2000
;_________________HLAVNI VYPOCET_____________________
N1                              ;TAM-------------------
IF R17==1   ;pokud hrubovani
  G1
  GOTO N2
ELSE        ;pokud finis
  A3=1 B3=-1 C3=4
  BSPLINE
ENDIF
N2                  ;feed řídících bodů tam
IF R11<=R13
  R3=100*R10+R11 R9=2  ;vypocet dalsiho ridiciho bodu
  R11=R11+R14
  GOTOF "N"<<R3
ENDIF
```

```
GOTOF N5
N3                          ;ZPĚT-------------------
IF R17==1    ;pokud hrubovani
  G1
  GOTO N4
ELSE         ;pokud finis
  A3=1 B3=1 C3=3
  BSPLINE
ENDIF
N4                  ;feed řídících bodů zpět
IF R11>=1
  R3=100*R10+R11 R9=4  ;vypocet dalsiho ridiciho bodu
  R11=R11+R14
  GOTOF "N"<<R3
ENDIF
N5                  ;test na konec-------------------
IF R10>=R12 GOTOF END
R10=R10+R15         ;+1 const ose
IF R10>=R12
  R10=R12           ;if posun na vic nez 65 then 65
ENDIF
;R11=R11-R14   ;návrat hodnoty R11 na 1 nebo R13
R14=-R14      ;otočení směru
IF R14>0
  R9=1 R11=R2
ELSE
  R9=3 R11=R13
ENDIF
G1
R16=100*R10+R11
GOTOF "N"<<R16
;____________________UKONCENI_____________________________
END:
G0 Z200
X200 Y200 A0 C0
IF R17==1
```

```
  R17=0
  R14=ABS(R14/R18)
  R15=ABS(R15/R19)
  ;SUPA
  GOTOB PREDBEZNE
ENDIF
FFWOF
M5 M9
M30
;____________________SOURADNICE_____________________________
N101 X-50.0 Y50.0 Z-30.00012772
                                    GOTOB "N"<<R9
N102 X-50.0 Y47.41156332 Z-30.00006662
                                    GOTOB "N"<<R9
N103 X-50.0 Y44.82312665 Z-29.99966836
                                    GOTOB "N"<<R9
…
…
N1652 X-23.50642844 Y-44.45393097 Z-25.74642220
                                    GOTOB "N"<<R9
N1653 X-23.50642844 Y-47.22696548 Z-27.84627539
                                    GOTOB "N"<<R9
N1654 X-23.50642844 Y-50.0 Z-29.99990710
                                    GOTOB "N"<<R9
N1701 X-21.45724995 Y50.0 Z-30.00002474
                                    GOTOB "N"<<R9
N1702 X-21.45724995 Y47.41156332 Z-28.04538303
                                    GOTOB "N"<<R9
…
…
N6552 X50.0 Y-44.45393097 Z-30.00175403
                                    GOTOB "N"<<R9
N6553 X50.0 Y-47.22696548 Z-29.99884599
                                    GOTOB "N"<<R9
N6554 X50.0 Y-50.0 Z-30.00018586
                                    GOTOB "N"<<R9
```

PŘÍLOHA 7

```
   RUN_VACKA_MPF


   PROC RUN_VACKA
   EXTERN VACKA_CYKLUS(REAL,REAL,REAL)
   DEF REAL POCATECNI_R, TLOUSTKA, INKREMENT_UHLU
   DEF REAL UHEL_1, RADIUS_1, UHEL_2, RADIUS_2, UHEL_3, RADIUS_3,
UHEL_4, RADIUS_4, UHEL_5, RADIUS_5
   ;--------ZADÁNÍ GEOMETRIE--------
   POCATECNI_R=10
   TLOUSTKA=-3
   INKREMENT_UHLU=1
   UHEL_1=12 RADIUS_1=10
   UHEL_2=35 RADIUS_2=6.25
   UHEL_3=175 RADIUS_3=11
   UHEL_4=25 RADIUS_4=17.5
   UHEL_5=113 RADIUS_5=17.5
   ;-----------NASTAVENÍ-----------
   WORKPIECE(,"",,"RECTANGLE",0,0,-30,0,100,100)
   CYCLE800(1,"0",100000,57,0,0,0,0,0,0,0,0,0,-1,,0)
   VAC_POLE[0,0]=SET(0,POCATECNI_R,UHEL_1, RADIUS_1, UHEL_2,
RADIUS_2,UHEL_3, RADIUS_3,UHEL_4, RADIUS_4,UHEL_5, RADIUS_5)
   ;TRAFOOF
   G17 G54 G71 G90
   G0 X200 Y200 Z200 A=0 C=0
   ;---PROSTOR NA PŘÍPRAVU OBROBKU--
   ;---PŘÍPRAVA NA OBROBENÍ VAČKY---
   Z200
   T="CUTTER 4"
   M6
   S2000 F2000
   IF (UHEL_1+UHEL_2+UHEL_3+UHEL_4+UHEL_5)<>360
       MSG("CHYBA V ZADANI GEOMETRIE: uhly neuzaviraji kruh")
       M0
   ENDIF
   ;-----------GEOMETRIE-----------
   IF NOT ISFILE("_N_GEOMETRIE_VACKA_SPF")
```

```
      MSG("spoustim tvorbu programu")
      VACKA_CYKLUS(POCATECNI_R, TLOUSTKA, INKREMENT_UHLU)
      MSG("hotovo, program GEOMETRIE vytvoren")
   ENDIF
   _N_GEOMETRIE_VACKA_SPF
   ;-----------UKONČENÍ-------------
   X200 Y200 A0 C0
   SUPA
   M5 M9
   M30



   VACKA_CYKLUS_SPF


   PROC VACKA_CYKLUS (REAL POCATECNI_R, REAL TLOUSTKA, REAL
INKREMENT_UHLU) SAVE
   DEF REAL AKT_R, AKT_UHEL, AKT_X, AKT_Y, KONECNY_UHEL, DELTA_R,
VAR_i
   DEF STRING[50]
DEBURGER="/_N_MPF_DIR/_N_VACKA_DIR/_N_DEBURGER_SPF"
   DEF STRING[50]
PROGRAM="/_N_MPF_DIR/_N_VACKA_DIR/_N_GEOMETRIE_VACKA_SPF"
   DEF INT ERROR
   ;pohyb po:
   BSPLINE


   AKT_R=POCATECNI_R
   VAC_POLE[0,1]=SET(AKT_R)


   GROUP_BEGIN(0,"DEBURGER INIT",0,0)
   IF(ISFILE(PROGRAM))
      DELETE(ERROR,PROGRAM)
   ENDIF
   WRITE(ERROR,PROGRAM,";Date:
"<<$A_DAY<<"."<<$A_MONTH<<"."<<($A_YEAR+2000)<<"  Time:
"<<$A_HOUR<<":"<<$A_MINUTE<<":"<<$A_SECOND)
   GROUP_END(0,0)
   GROUP_BEGIN(0,"IF ERROR",0,0)
```

```
   IF ERROR
       CASE ERROR OF 1 GOTOF N21 2 GOTOF N22 3 GOTOF N23 4 GOTOF N24
10 GOTOF N25 11 GOTOF N26 12 GOTOF N27 13 GOTOF N28 20 GOTOF N29
       N21 MSG ("CHYBA ZÁPISU: " << ERROR << " Cesta není povolena")
       M0
       N22 MSG ("CHYBA ZÁPISU: " << ERROR << " Cesta není nalezena")
       M0
       N23 MSG ("CHYBA ZÁPISU: " << ERROR << " Soubor není nalezen")
       M0
       N24 MSG ("CHYBA ZÁPISU: " << ERROR << " nesprávný datový
typ")
       M0
       N25 MSG ("CHYBA ZÁPISU: " << ERROR << " Soubor je plný")
       M0
       N26 MSG ("CHYBA ZÁPISU: " << ERROR << " Soubor je využíván")
       M0
       N27 MSG ("CHYBA ZÁPISU: " << ERROR << " žádné volné
kapacity")
       M0
       N28 MSG ("CHYBA ZÁPISU: " << ERROR << " žádná přístupová
oprávnění")
       M0
       N29 MSG ("CHYBA ZÁPISU: " << ERROR << " jiná chyba")
       M0
   ENDIF
   GROUP_END(0,0)
   GROUP_BEGIN(0,"HLAVICKA",0,0)
   WRITE(ERROR,PROGRAM,"G[06]="<<$P_GG[6]<<"    ; G17/G18/G19")
   WRITE(ERROR,PROGRAM,"G90 G94")
   WRITE(ERROR,PROGRAM,";NASTROJ a M6")
   WRITE(ERROR,PROGRAM,"F"<<$P_F) ;feed posledniho nastaveni
   WRITE(ERROR,PROGRAM,"S"<<$P_S[$AC_MSNUM]<<"
M"<<$P_SDIR[$AC_MSNUM]) ;speed a smer otaceni posledniho nastaveni
   WRITE(ERROR,PROGRAM,";CYCLE800(1,0,0,57,0,0,0,0,0,0,0,0,0,-1)")
   WRITE(ERROR,PROGRAM,";CYCLE832(0.01,_FINISH,1)")
   WRITE(ERROR,PROGRAM,"FIFOCTRL CUT2DF KONT")
   WRITE(ERROR,PROGRAM,"G[08]="<<$P_GG[8]<<"    ;G500,G54,G55,...")
   WRITE(ERROR,PROGRAM,"G[13]="<<$P_GG[13]<<"    ;G70,G71")
```

```
   GROUP_END(0,0)
   GROUP_BEGIN(0,"EXPORT UVOD",0,0)
   WRITE(ERROR,PROGRAM,"G0 X0 Y"<<POCATECNI_R+5<<"")
   WRITE(ERROR,PROGRAM,"G1 Z"<<TLOUSTKA<<"")
   WRITE(ERROR,PROGRAM,"Y"<<POCATECNI_R<<"")
   GROUP_END(0,0)
   WRITE(ERROR,DEBURGER,"AKT_R= "<<AKT_R<<"")

   ;-----------GEOMETRIE-----------

   FOR var_i = 0 TO 20

     WRITE(ERROR,DEBURGER,"")
     WRITE(ERROR,PROGRAM,"")
     WRITE(ERROR,DEBURGER,"VAR_I= "<<VAR_I<<"")

     TEST_POKRACOVANI:
     IF VAC_POLE[var_i+2,0]==0 GOTOF END

     KONECNY_UHEL=AKT_UHEL+VAC_POLE[var_i+1,0]
     DELTA_R=(VAC_POLE[var_i,1]-
VAC_POLE[var_i+1,1])/VAC_POLE[var_i+1,0]*INKREMENT_UHLU
     WRITE(ERROR,DEBURGER,"DELTA_R= "<<DELTA_R<<"")

     IF VAC_POLE[var_i,1]==VAC_POLE[var_i+1,1] GOTOF KRUH

     WRITE(ERROR,DEBURGER,"AKT_UHEL= "<<AKT_UHEL<<"")
     WRITE(ERROR,DEBURGER,"VAC_POLE[var_i+1,0]=
"<<VAC_POLE[var_i+1,0]<<"")
     WRITE(ERROR,DEBURGER,"KONECNY_UHEL= "<<KONECNY_UHEL<<"")
     WRITE(ERROR,PROGRAM,"G[01]="<<$P_GG[1]<<"")

     WHILE AKT_UHEL < KONECNY_UHEL
       AKT_UHEL=AKT_UHEL+INKREMENT_UHLU
       AKT_R=AKT_R-DELTA_R
       AKT_X=-SIN(AKT_UHEL)*AKT_R
       AKT_Y=-COS(AKT_UHEL)*AKT_R
```

```
      WRITE(ERROR,DEBURGER,"AKT_UHEL= "<<AKT_UHEL<<"")
      WRITE(ERROR,DEBURGER,"AKT_R= "<<AKT_R<<"")
      WRITE(ERROR,PROGRAM,"X="<<AKT_X<<" Y="<<AKT_Y<<"")
    ENDWHILE

    COUNTER:
    AKT_R=VAC_POLE[var_i+1,1]
    AKT_UHEL=KONECNY_UHEL

  ENDFOR

  KRUH:
    AKT_UHEL=KONECNY_UHEL
    AKT_X=-SIN(AKT_UHEL)*AKT_R
    AKT_Y=-COS(AKT_UHEL)*AKT_R
    WRITE(ERROR,DEBURGER,"AKT_R= "<<AKT_R<<"")
    WRITE(ERROR,DEBURGER,"AKT_UHEL= "<<AKT_UHEL<<"")
    WRITE(ERROR,DEBURGER,"AKT_X= "<<AKT_X<<"")
    WRITE(ERROR,DEBURGER,"AKT_Y= "<<AKT_Y<<"")
    WRITE(ERROR,PROGRAM,"G2 X="<<AKT_X<<" Y="<<AKT_Y<<"
CR="<<AKT_R<<"")
  GOTOB COUNTER

  ;-----------UKONČENÍ-------------
  END:
    WRITE(ERROR,PROGRAM,"G0 Z200")
    WRITE(ERROR,PROGRAM,"X200 Y200 A0 C0")
    WRITE(ERROR,PROGRAM,"SUPA FFWOF")
    WRITE(ERROR,PROGRAM,"M5 M9")
    WRITE(ERROR,PROGRAM,"M17")
  ;VAC_POLE[0,0]=REP(0) ;anulace tabulky geometrie
  M17
```