



Budapesti Műszaki és Gazdaságtudományi Egyetem

Gépészmérnöki Kar

Műszaki Mechanikai Tanszék

FORGÁCSOLÓ ERŐ SZIMULÁCIÓJA 3 TENGELYES MEGMUNKÁLÁS ESETÉN DEXEL TESTMODELLEZÉSI ELJÁRÁSSAL

Szerzők: **Sykora Henrik**, Msc. I. évf
Gépészmérnöki szak
Budapesti Műszaki és Gazdaságtudományi Egyetem

Kovács Attila, Msc. I. évf
Gépészeti modellezés szak
Budapesti Műszaki és Gazdaságtudományi Egyetem

Konzulens: **Dr. Bachrathy Dániel**, egyetemi adjunktus
Műszaki Mechanikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Budapest, 2014

Tartalomjegyzék

1. Bevezetés	2
2. Dixel módszer	3
2.1. Dixel elem felépítése	3
2.2. A dixel adatszerkezet megvalósítása	3
2.3. Műveletek	5
2.4. Műveletek végzése	6
3. Testek dexeltérben való modellezése	7
3.1. Munkadarabok létrehozása	7
3.2. Szerszám generálása	7
3.3. Testek generálásáról általánosan	8
3.4. Téglatest	9
3.5. Henger	10
3.6. Ferde henger	10
3.7. Hengerpalást által seprert térfogat	13
4. CWE számítás	15
4.1. CWE előállítás	15
4.2. CWE felbontás megválasztása	17
5. Erőszámítás	19
5.1. Forgácsoló erő számítás alapja	19
5.2. Forgácsoló erő számítás megvalósítása	21
5.3. Előtolási sebesség maximalizálása	21
6. Szerszámpályák	24
7. Szimuláció	25
7.1. Szimuláció pontossága, időigénye	25
7.2. Szimulációra példa	28
8. Összefoglalás	31

1. fejezet

Bevezetés

Egy termék forgácsoló megmunkálásának tervezésénél napjainkra már elengedhetlenné vált a CAM szoftverek (Computer-Aided Manufacturing) alkalmazása. A gyors és hatékony gyártáshoz a megfelelően kialakított szerszámpálya nagyon fontos, amit kézzel csak a legegyszerűbb geometriák esetén lehet elkészíteni, bonyolultabb esetekben (többtengelyes megmunkálás, szoborfelületek) viszont már csak CAM szoftverek segítségével. Továbbá ezek a programok használhatók a szerszámpályák ellenőrzésére (ütközések detektálása, programozási hibák felderítése), míg bonyolultabb algoritmusok segítségével egy adott szerszámpálya optimalizálható erő-, nyomaték és teljesítménymaximumra.

Általában ezek a CAM szoftverek csak a geometriai igényeknek megfelelő szerszámpályát terveznek, de a forgácsolás során fellépő dinamikus hatásokat ezek az algoritmusok nem veszik figyelembe.

Jelenleg kutatások alapján, a CWE ismeretében a felületminőséget és a kialakuló rezgéseket előre tudjuk jelezni, viszont olyan esetekben, amelyekben a szerszámpálya vagy a munkadarab geometriája bonyolult, akkor a CWE nehezen meghatározható. Vannak törekvések arra, hogy a megmunkálás során fellépő dinamikus hatásokat minél pontosabban modellezzék: Tukora [1], Altintas [2]. A kereskedelembe kapható CAM szoftverekből jelenleg nem lekérhetőek ezek az adatok. Ha ismerjük a CWE mellett a szerszámgép dinamikai tulajdonságait és a szerszám pontos paramétereit (élek száma, menetemelkedés, radiális hullámossága, stb.), akkor számolható a forgácsoló erő, becslés adható a stacionárius forgácsolás stabilitására [3], a kialakuló rezgések amplitúdójára ezáltal a felületi minőségre.

A TDK dolgozatnak a célja egy 2D esetre elkészített Matlab program [4] továbbfejlesztése egy csavart élű hengeres szerszámmal való általános 3 tengelyes megmunkálási folyamat szimulációjára, amely alkalmas a megmunkálás során fellépő hatások (erők, nyomatékok) számítására is. A program C++ környezetben készült, amely segítségével bővíthető és gyors szoftvert lehet fejleszteni, amely alkalmas lehet a Budapesti Műszaki és Gazdaságtudományi Egyetem Műszaki Mechanikai tanszéken zajló szerszámgéprezgésekkel kapcsolatos kutatások támogatására.

2. fejezet

Dexel módszer

A dexel módszer (Depth element) egy diszkrétizált test modellezési eljárás, mely során egy alaphalmazból (2D esetben egyenesből, 3D esetben síkból) a testre egyenközűen bocsájtott sugarak és a test határainak metszéspontjai reprezentálják a testet [5]. A 2.1. ábrán egy síkbeli geometria két irányú dexel leírása látható, ha egyszerre mindkettő ismert, akkor a reprezentációt multi-dixelnek nevezik. A 3D leírás a 2.2. ábrán látható, a síkbeli leíráshoz képest csak az alaphalmaz dimenziója különböző, és hogy a teljes tridixel leíráshoz az Z síkra merőleges dexelet is meg kell adni. Ezzel az eljárással modellezett testeken könnyen lehet halmazműveleteket végezni, mivel az egyes dexelsíkok és ezeken belül az egyes dexelet teljesen külön számíthatók, így párhuzamos számításra alkalmas, amely gyorsaság miatt alkalmas CAM számításokra.

2.1. Dexel elem felépítése

A dexelet vektorokkal adhatjuk meg. Minden egyes sugárhoz tartozik egy vektor, amiben a test határaival való metszéspontokat tároljuk. Minden egyes vektor páros számú elemet tartalmaz (be-, és kilépő pontok). Egy dixelhez tartozó vektorra példa (2.1a. ábra alapján):

$$D_2 = (d_{y_2,1} \quad d_{y_2,2}) \quad (2.1.1)$$

A dixelben a metszéspontok mellett eltárolható még a pontban lévő felületi normális, a számított felületminőség, stb.

2.2. A dexel adatszerkezet megvalósítása

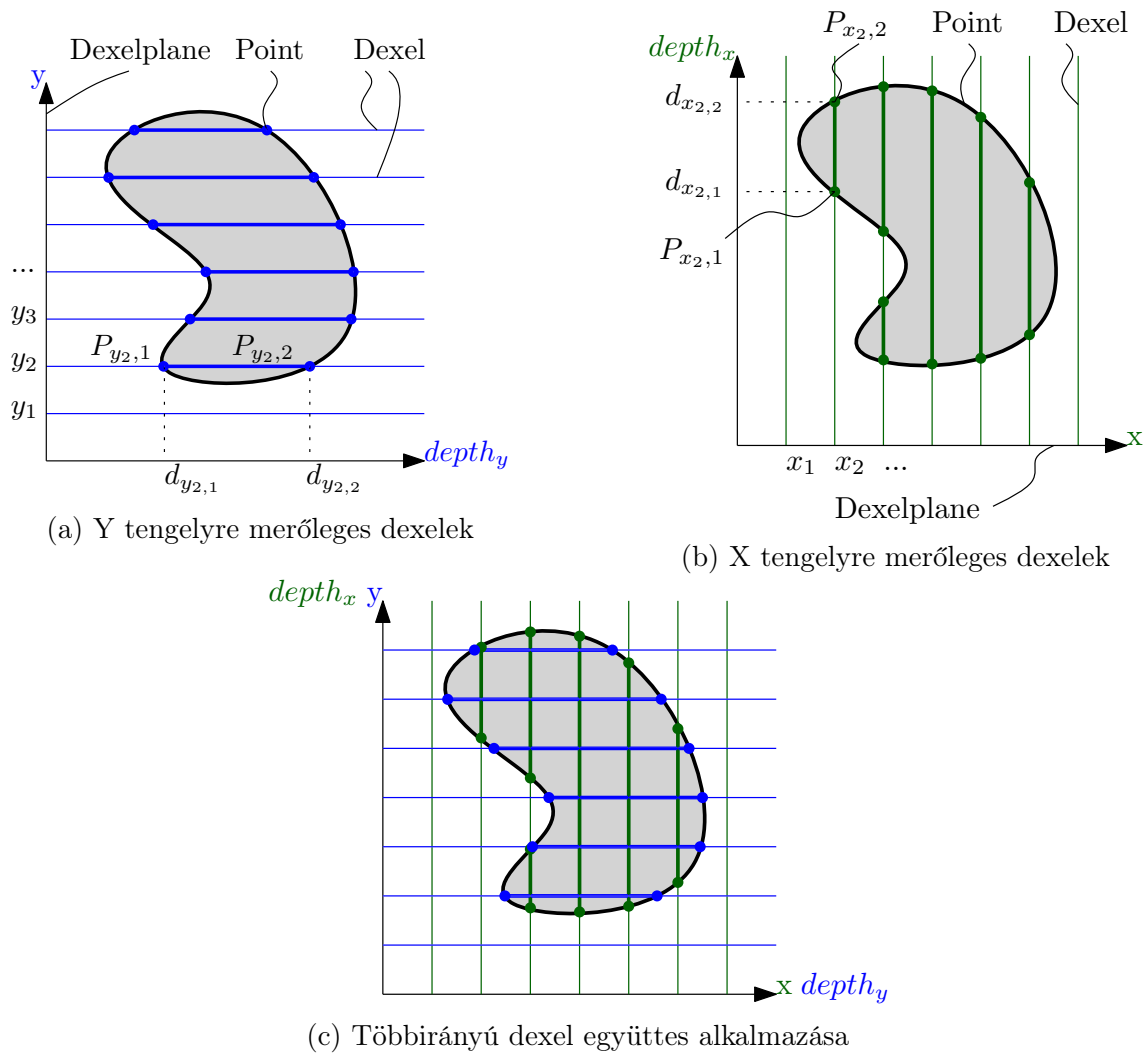
Az eljárás megvalósítása C++ programozási nyelven objektum orientált megközelítéssel készült, szabványos C++ könyvtárak és OpenGL segítségével. A legfontosabb objektumokat a 2.3. grafikon ábrázolja. Főbb osztályok:

Point : Mélységadat, amely a dexel-test metszéspontot adja meg, adott dexelsíktól való merőleges távolságként. Továbbá tartalmaz egy 3D vektort amely az adott pontbeli a felületi normalist adja meg, és plusz adatokat is, amelyek a vizualizációt és a forgácsolási adatok postprocesszálását segítik.

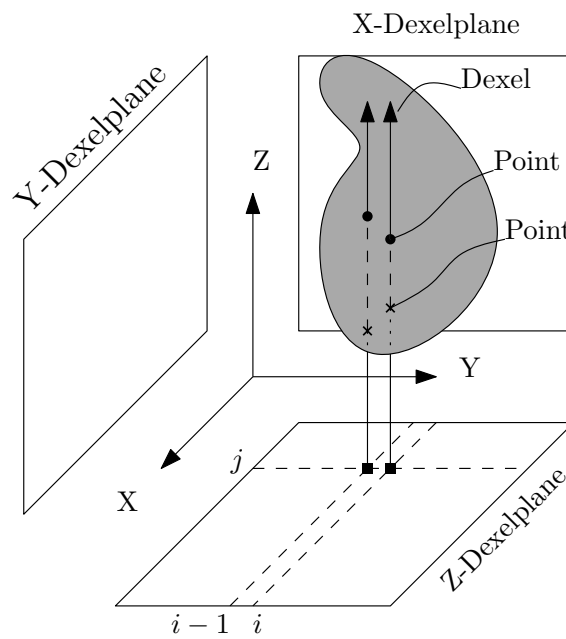
Dexel : Pontok listája dexelműveletekkel ellátva (kivonás, összedás, metszet).

Dexelplane : Dexelet 2D tömbje műveletekkel ellátva.

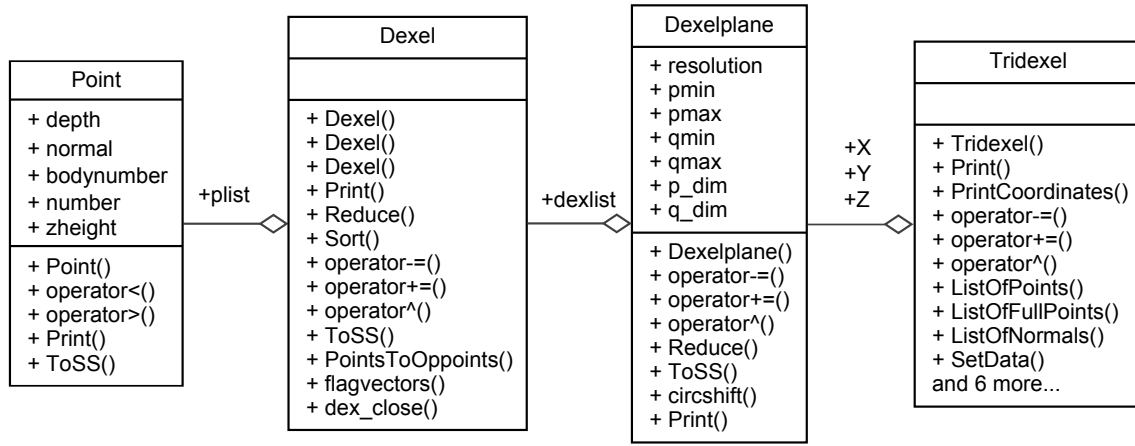
Tridixel : 3 egymásra merőleges dexelsík, műveletekkel ellátva.



2.1. ábra. Síkbeli geometria leírása



2.2. ábra. Tridexel módszer 3D esetre



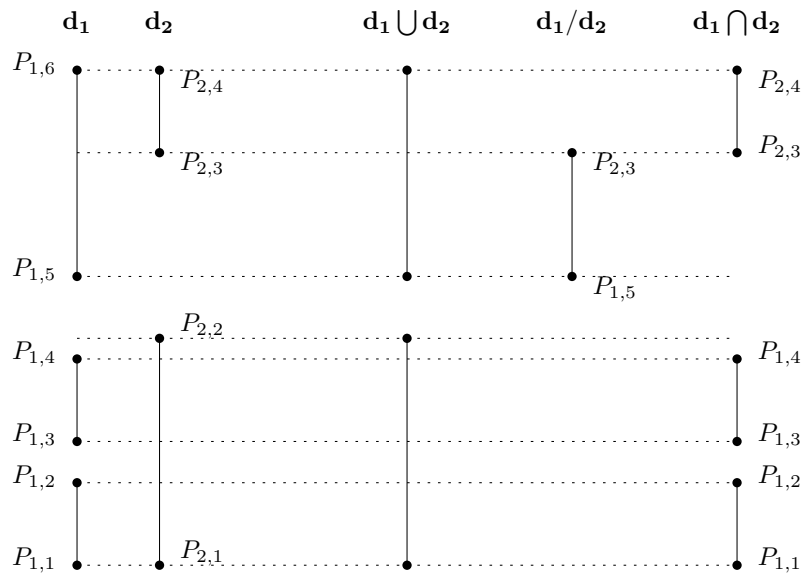
2.3. ábra. Tridexel osztály diagram

2.3. Műveletek

Szükséges az adatszerkezet mellett műveleteket is definiálni, melyeket a megfelelő helyen (azonos a dexelsík, azonos a globális koordináta) elhelyezkedő dexelek között végzünk el. Három lényeges műveletet különböztethetünk meg (2.4. ábra):

- Összeadás ($\cup$)
- Kivonás ($/$)
- Metszet ($\cap$)

Mivel ezek a műveletek csak két dexel között történnek a számítások jól párhuzamosíthatóak, ezért alkalmas lehet nagy párhuzamos számításkapacitású GPGPU-n (videokártyán) történő megvalósításra is. [1]



2.4. ábra. Két dexel közötti boolean műveletek

2.4. Műveletek végzése

A dexelekkel való műveletek végzése 3 fő lépésből áll:

- ún. logikai flagvektorok előállítása amely megmutatja, hogy hol van anyag (2.4. ábra - ahol van anyag: igaz (true $\equiv$ 1), ahol nincs: hamis (false $\equiv$ 0))
- műveletek elvégzése
- flagvektorok alapján eredménydixel előállítása

A flagvektorokat úgy hozzuk létre, hogy a két dixelből (amikkel a műveleteket végezzük) vesszük az összes pontot, és egy közös vektorba rendezzük. Mindkét dixel pontjaihoz hozzárendelünk egy-egy számot, amely mutatja, hogy a dixel melyik testhez tartozott, ezzel reprezentálva a **Point** osztály többi változóját. Ezek után a mélységadatok alapján a pontokat növekvő sorrendbe rendezzük. A pontok által tárolt számok segítségével elkészíthetők most már a dexelekhez tartozó flagvektorok. Ezután a flagvektoron elvégezzük a műveleteket, és visszaválogatjuk az eredményként kapott flagvektor segítségével az eredménydexelt.

Egy példa a folyamatra:

- adott két dixel (a pontok koordinátáihoz hozzárendelve az őket tartalmazó dixel sorszáma):

$$D_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1.1 & 2.3 & 6.0 & 8.9 \end{pmatrix} \quad (2.4.1)$$

$$D_1 = \begin{pmatrix} 1 & 1 \\ 3.2 & 7.5 \end{pmatrix} \quad (2.4.2)$$

- ezeket a pontokat egy vektorba helyezzük, majd növekvő sorrendbe rendezzük, és elkészítjük a flagvektorokat:

$$D = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 1.1 & 2.3 & 6.0 & 8.9 & 3.2 & 7.5 \end{pmatrix} \quad (2.4.3)$$

$$D_{növ} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1.1 & 2.3 & 3.2 & 6.0 & 7.5 & 8.9 \end{pmatrix} \quad (2.4.4)$$

$$flag_0 = (1 \ 0 \ 0 \ 1 \ 1) \quad (2.4.5)$$

$$flag_1 = (0 \ 0 \ 1 \ 1 \ 0) \quad (2.4.6)$$

- A flagvektorokon végrehajtjuk a műveletet (most kivonás):

$$flag_{result} = flag_1 - flag_0 = (0 \ 0 \ 1 \ 0 \ 0) \quad (2.4.7)$$

- Most az eredményként kapott flagvektor segítségével visszaválogatjuk az eredménydexelt. Ezt úgy tehetjük meg a legegyszerűbben, hogy azokat az elemeket tesszük be az eredménydixelbe, amelyeknél a flagvektor vált (0-ról 1-re, vagy fordítva - xor bitművelet). Ez a jelen esetben a következő (az eredeti dexelek sorszámozását most már elhagyhatjuk):

$$D_{result} = (3.2 \ 6.0). \quad (2.4.8)$$

Továbbá a pontokhoz tartozó normálvektorok változnak a kivonás során: a kivágott rész normálvektorának ellentettje másolódik ki a kivonás utána dixel megfelelő pontjához (a példában az eredeti, kivont dixel 6.0 értékű pontjához tartozó normálvektor ellentettje lesz az eredmény 6.0 értékű pontjában a felületi normális).

3. fejezet

Testek dexeltérben való modellezése

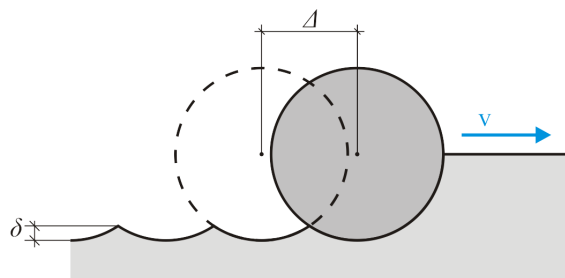
3.1. Munkadarabok létrehozása

A munkadarabokat legegyszerűbben egy téglatest előgyártmányból állíthatjuk elő, továbbá a programban lehetőséget van arra, hogy az előző fejezetekben ismertetett primitívekkel végzett műveletek segítségével bonyolultabb kezdeti munkadarabokat vagy előgyártmányokat állítunk elő.

3.2. Szerszám generálása

Egy megmunkálás szimulációjakor szükséges, hogy előállítsunk egy szerszámgeometriát, amit kivonunk a megmunkált darabból. Marási folyamatoknál jól közelíthető a szerszám egy egyszerűbb geometriával, ha élünk azzal a feltételezéssel, hogy az élek sebességének szerszám forgásából származó sebességkomponense jóval nagyobb, mint az előtolásból származó sebességkomponense. Ilyen egyszerűbb geometria egy kör (síkban), egy henger (térben), vagy egy henger és egy félgömb uniója (térben - gömbvégű szerszám esetén).

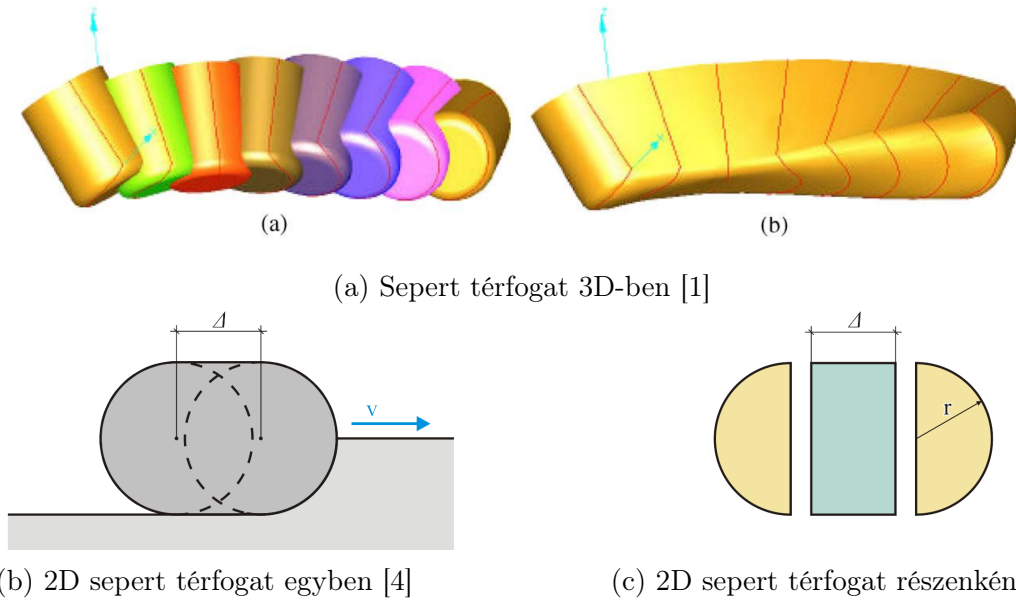
Ezeket a szerszámgeometriákat a szerszám pálya minden egyes pontján le kell generálni. Ha a szerszám pályát adott Δ felbontással felosztjuk, diszkrétizáljuk, akkor a szimulált forgácsolt felületen visszamarad a diszkrétizálásból származó, Δ -tól függő δ felületi hiba (3.1. ábra). Ez a δ felületi hiba megszüntethető, hogy ha a szerszám



3.1. ábra. Szerszám pálya felbontásából adódó hiba [4]

által két szimulációs lépés között seprt teljes térfogatot legeneráljuk [1]. Ekkor egy 3 tengelyes megmunkálás során a szerszám pálya diszkrétizálásából származó hiba, csak a pontok között egyenes pályán való haladásból származó „töredezettség” lesz, de ez elfogadható a dixel módszer diszkrétizációjából származó hiba mellett.

A separt térfogatra létezik néhány közelítő eljárás, de zárt alakú analitikus megoldás még nem áll rendelkezésre [1] generálásukhoz. Viszont 2,5-, 3 tengelyes megmunkálásnál, ahol a szerszám egyenesek mentén mozog és a szerszám tengelye mindig párhuzamos a globális z tengellyel önmagával párhuzamos helyzetben, ez a separt térfogat felbontható egyszerűbb testprimitívekre (hengerekre, hasábkra, gömbökre), amelyek uniójából előállítható a separt térfogat. A program 3 tengelyes meg-



3.2. ábra. Separt térfogat

munkálás szimulációjára alkalmas, így szerszámot egy általános mozgás során három fő primitívre bonthatjuk:

- a mozgás eleji és végi helyzetben lévő hengerekre
- a mozgás során henger palástja által separt ferde hasábra
- és a szerszám alja és teteje (körlap) által separt ferde hengerekre.

Az általános mozgás mellett a szerszám generálásakor megkülönböztetünk gyakran előforduló speciális mozgási lehetőségeket (a számítás gyorsításához):

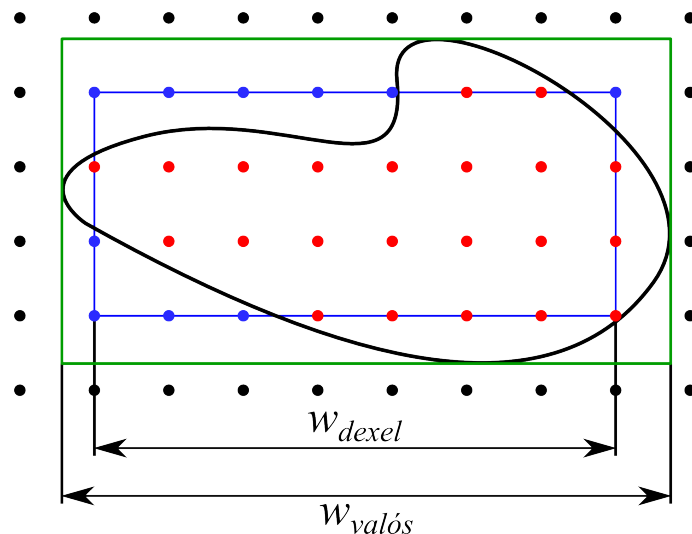
- z tengely mentén mozgás: egy, z irányban megnyújtott hengerrel modellezhető a szerszám
- x vagy y tengely mentén: két henger és koordinátasíkokkal párhuzamos lapokkal rendelkező téglatest
- általános xy , xz , vagy yz síkban való mozgás: két henger és egy téglatest, melynek két lapja párhuzamos az xy síkkal, a többi lap általános helyzetű

A szerszám minden esetben létrehozható a 3.4-3.7 fejezetekben ismertetett testprimitívek felhasználásával.

3.3. Testek generálásáról általánosan

Első lépésben a generált test valós befoglaló téglatestjét kell meghatározni (3.3. ábra - $w_{valós}$). Ez a test kijelöli a dexelsíkokon, hogy mely dexelek „döfik” át a modellezett testet. Mivel a dexelek a síkok diszkrét pontjai fölött vannak, ezért ez a lépés a síkkal

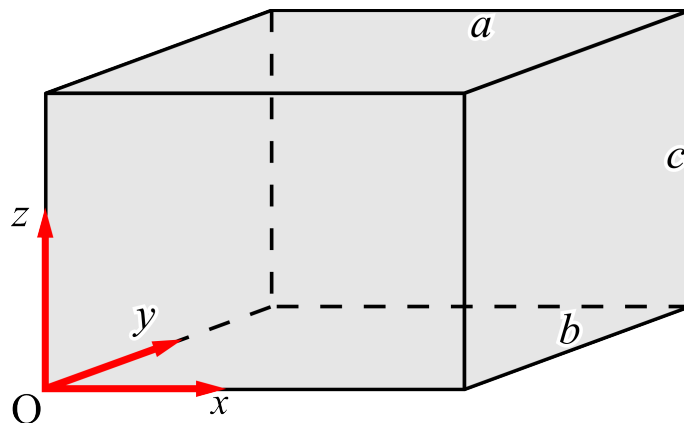
párhuzamosan egy dexelfelbontással arányos pontatlanságot okoz (3.3. ábra - w_{dixel} és $w_{valós}$ különbsége), de ezt a merőleges síkokról bocsájtott dexelek kompenzálják ezt a hibát, ahogy a 2.1c. ábrán szerepel.



3.3. ábra. Testek befoglaló méreteinek alakulása

3.4. Téglatest

A téglatest, amelynek a lapjai párhuzamosak az xy , xz és yz síkokkal, a legegyszerűbb test primitív amit létre lehet hozni dixel módszerrel. A dixel irányától függően a dixelbe az 3.4. ábrán látható O pont megfelelő koordinátája kerül (x irányú dixel esetén: O_x , y esetén: O_y , z esetén pedig: O_z), majd az adott irányú dimenzióval növelt értéke (x : $O_x + a$, y : $O_y + b$, z : $O_z + c$). Abban az esetben, ha a , b , vagy c értéke negatív, akkor a megfelelő irányú dixel pontjai fordított sorrendben kerülnek a dixelbe (a dixel növekvő sorrendben tartalmazza a mélységadatokat). A pontokhoz tartozó normálvektorok pedig a dixel irányának megfelelő koordinátaírányok egységvektorai, illetve annak ellentettjei.



3.4. ábra. Téglatest

3.5. Henger

A következő egyszerűen létrehozható testprimitív a z tengellyel párhuzamos egyenes henger. A henger definiálásához szükséges alsó alapkörének a középpontjainak koordinátái, sugara és a henger magassága. A z irányú dexelekhez csupán a henger alsó alapjának középpontjának P_z koordinátája, illetve a henger h magassága szükséges (3.5. ábra), az x és y irányú dexelek feltöltésekor pedig meg kell nézni, hogy az adott dixel által kijelölt egyenes milyen messze vannak a kör középpontjától. Ez a távolság y irányú dexelek esetén:

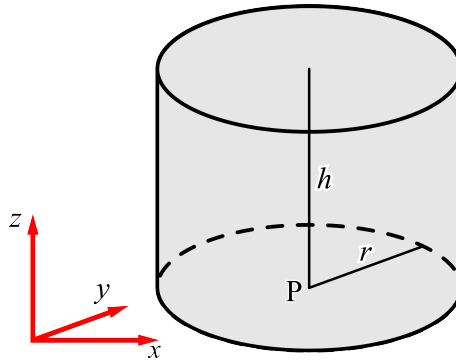
$$\Delta x_{i,j} = d_{i,j,x}^{(y)} - P_x, \quad (3.5.1)$$

ahol P_x a henger alsó alapkörének középpontjának x koordinátája, a $d_{i,j,x}$ pedig az xz sík i -edik sorban és j -edik oszlopában lévő y irányú dixel. Ebből a dixelhez tartozó vektor elemei számítható az alábbi összefüggés szerint:

$$p_{i,j:1,2}^{(y)} = P_y \mp \sqrt{r^2 - \Delta x_{i,j}^2}, \quad (3.5.2)$$

Ugyanezekhez a pontokhoz tartozó normálvektorok koordinátái a következők:

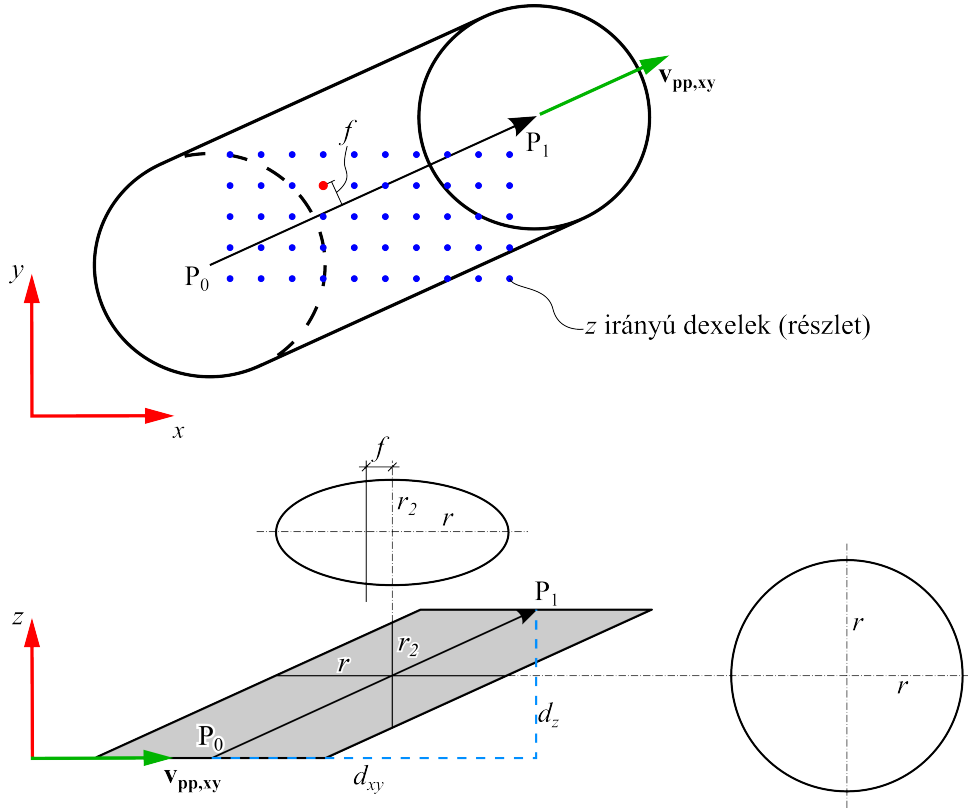
$$\begin{aligned} n_{x:1,2} &= \frac{\Delta x_{i,j}}{r} \\ n_{y:1,2} &= \mp \frac{\sqrt{r^2 - \Delta x_{i,j}^2}}{r} \\ n_{z:1,2} &= 0 \end{aligned} \quad (3.5.3)$$



3.5. ábra. Henger

3.6. Ferde henger

A programban lehetőség van ferde henger létrehozására is. A henger alapjai az xy síkkal párhuzamosak, a henger irányát a $\overrightarrow{P_1P_2}$ vektor jelöli ki (3.6. ábra). Ha a $\overrightarrow{P_1P_2}$ vektor z irányú, akkor a ferde henger egy egyenes hengerre vezethető vissza, vagy ha az xy síkban fekszik, akkor pedig nem értelmezett. Az alábbiakban ismertetett generálási mód általános helyzetű $\overrightarrow{P_1P_2}$ vektor esetére érvényes. Az xy síkbeli dexelekhez tartozó vektorok előállítása hasonló a hengeréhez, mivel a ferde



3.6. ábra. Ferdehenger vázlata

henger xy síkkal párhuzamos metszetei körök (3.6. ábra). A körök középpontjának x és y koordinátáját interpolálhatjuk P_1 és P_2 pontok közé a x és y irányú dexelek z koordinátája alapján (mivel a $\overline{P_1P_2}$ szakaszon helyezkednek el. Azok a dexelek üresen maradnak, amelyek vízszintes síkban messzebb vannak a kör középpontjától (3.6. ábra), mint a kör sugara.

A normálvektorok iránya viszont változik. Egy olyan vektorra van szükség, ami a körre is, és a ferde henger alkotójára is merőleges. Az alábbiakban egy gyors, de csak közelítő megoldást ismertetünk a normálvektorok megadásához. A körhöz kiszámoljuk a normálvektorokat, és alkalmazunk rá egy forgatási transzformációt. A z irányhoz tartozó normálvektorok (3.5.3) alapján számolható. A normálvektorokra alkalmazandó forgatási transzformáció megegyezik azzal a forgatási transzformációval, amit a z tengelyre kéne alkalmaznunk, ha párhuzamossá szeretnénk tenni a $\overline{P_1P_2}$ vektorral. Az ehhez tartozó forgatási transzformációt Rodriguez formulával számoljuk, amihez szükség van a forgás tengelyére és a forgás szögére. A forgatás tengelye:

$$\mathbf{v} = \frac{\mathbf{k} \times \overrightarrow{P_1P_2}}{|\mathbf{k} \times \overrightarrow{P_1P_2}|}, \quad (3.6.1)$$

ahol $\mathbf{k}$ a z irányhoz tartozó egységvektor. A forgatás szöge:

$$\vartheta = \arccos \frac{\mathbf{k} \cdot \overrightarrow{P_1P_2}}{|\overrightarrow{P_1P_2}|}. \quad (3.6.2)$$

A transzformációs mátrix:

$$\mathbf{R}(\vartheta, \mathbf{v}) = \mathbf{I} + \sin(\vartheta) \cdot \mathbf{V}(\mathbf{v}) + (1 - \cos(\vartheta)) \cdot \mathbf{V}(\mathbf{v}) \cdot \mathbf{V}(\mathbf{v}), \quad (3.6.3)$$

ahol $\mathbf{I}$ 3×3 -as egységmátrix, $\mathbf{V}(\mathbf{v})$ pedig a következőképpen áll elő:

$$\mathbf{V}(\mathbf{v}) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \quad (3.6.4)$$

A pontokhoz tartozó normálvektorokra történő becslés pedig a következőképpen alakul:

$$\hat{\mathbf{n}}_{1,2} = \mathbf{R} \cdot \mathbf{n}_{1,2}. \quad (3.6.5)$$

A z irányú dexelek feltöltése során a ferde henger függőleges metszetét kell venni, amely egy ellipszis. Az ellipszis z irányú r_2 tengelye az alábbi összefüggéssel számítható (lásd.: 3.6. ábra):

$$r_2 = r \cdot \frac{d_z}{d_{xy}}. \quad (3.6.6)$$

Szükség van a z irányú dexelek $\overline{P_1 P_2}$ szakasztól mért f távolságára (minden $d^{(z)}_{i,j}$ -re). Ha a $\overline{P_1 P_2}$ szakaszt és a z irányú dexelt is egyaránt levetítjük az xy síkra, akkor a probléma visszavezethető egy pont és egy egyenes távolságára [6]:

$$f_{i,j}^2 = \frac{(P_{2,x} - P_{1,x})(P_{1,y} - d^{(z)}_{i,j,y}) - (P_{1,x} - d^{(z)}_{i,j,x})(P_{2,y} - P_{1,y})}{(P_{2,x} - P_{1,x})^2 + (P_{2,y} - P_{1,y})^2}. \quad (3.6.7)$$

A távolságot négyzetesen hagyhatjuk, mivel a pontok számításában négyzetes formában lesz rá szükség. Ekkor az ellipszis pontjainak z koordinátája:

$$p^{(z)}_{i,j:1,2} = h_{k,i,j} \mp \frac{r_2}{r} \cdot \sqrt{r^2 - f_{i,j}^2}, \quad (3.6.8)$$

ahol a $h_{k,i,j}$ az ellipszis középpontjának z koordinátája, amely az alábbi egyenlet alapján számolható:

$$h_{k,i,j} = P_{1,z} + \frac{P_{2,z} - P_{1,z}}{P_{2,x} - P_{1,x}} \cdot (d^{(z)}_{i,j,x} - P_{1,x}), \quad (3.6.9)$$

vagy, ha a $\overline{P_1 P_2}$ vektor párhuzamos az x tengellyel:

$$h_{k,i,j} = P_{1,z} + \frac{P_{2,z} - P_{1,z}}{P_{2,y} - P_{1,y}} \cdot (d^{(z)}_{i,j,y} - P_{1,y}). \quad (3.6.10)$$

A pontokhoz tartozó normálvektorokra való becslése jelen esetben úgy állt elő, hogy egy x tengellyel párhuzamos, ellipszis alapú henger normálvektorait számoltuk, majd először ezt forgatási transzformációval elforgattuk a z tengely körül úgy, hogy a henger vezéregyenese párhuzamos legyen a $\overrightarrow{P_1 P_2}$ vektor xy síkra vett vetületével, majd innen továbbforgattuk úgy hogy a vezéregyenese párhuzamos legyen a $\overrightarrow{P_1 P_2}$ vektorral. Az x tengely irányú ellipszis alapú henger normálvektorai:

$$\begin{aligned} n_{x:1,2} &= 0, \\ n_{y:1,2} &= \frac{f_{i,j}}{\sqrt{\frac{r^2 - f_{i,j}^2}{h^2} + f_{i,j}^2}}, \\ n_{z:1,2} &= \mp \frac{\sqrt{r^2 - f_{i,j}^2}}{h \sqrt{\frac{r^2 - f_{i,j}^2}{h^2} + f_{i,j}^2}}. \end{aligned} \quad (3.6.11)$$

Az forgatás mátrixa pedig a következő:

$$\mathbf{R}_z = \mathbf{R}(\vartheta_v, \mathbf{v}) \cdot \mathbf{R}(\vartheta_x, \mathbf{k}), \quad (3.6.12)$$

ahol $\mathbf{k}$ a z tengely egységvektora, és a többi változó a következőképpen áll elő:

$$\vartheta_x = \arccos \frac{\mathbf{i} \cdot \overrightarrow{P_1 P_{2xy}}}{|\overrightarrow{P_1 P_{2xy}}|}, \quad (3.6.13)$$

$$\vartheta_v = \frac{(\mathbf{R}(\vartheta_x, \mathbf{k}) \cdot \mathbf{i}) \cdot \overrightarrow{P_1 P_2}}{|\overrightarrow{P_1 P_2}|}, \quad (3.6.14)$$

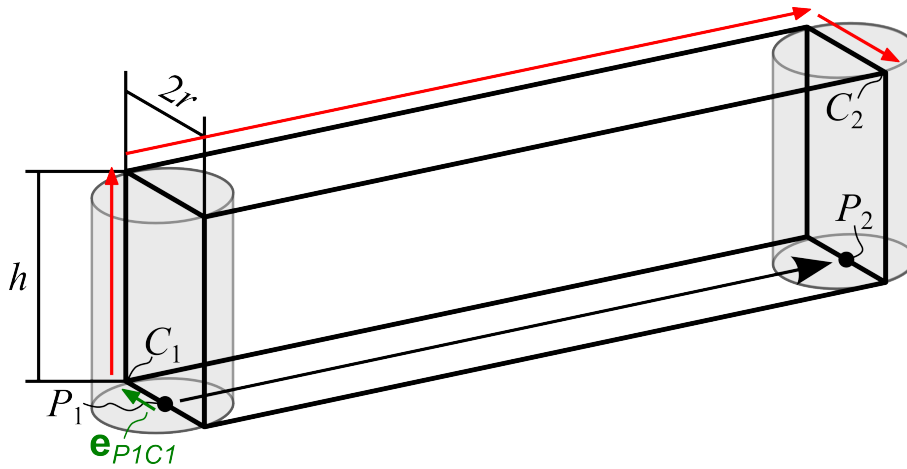
$$\mathbf{v} = \frac{(\mathbf{R}(\vartheta_x, \mathbf{k}) \cdot \mathbf{i}) \times \overrightarrow{P_1 P_2}}{|(\mathbf{R}(\vartheta_x, \mathbf{k}) \cdot \mathbf{i}) \times \overrightarrow{P_1 P_2}|}, \quad (3.6.15)$$

ahol $\overrightarrow{P_1 P_{2xy}}$ a $\overrightarrow{P_1 P_2}$ xy síkba vett vetülete, és az $\mathbf{i}$ pedig az x irányba mutató egységvektor.

3.7. Hengerpalást által seprt térfogat

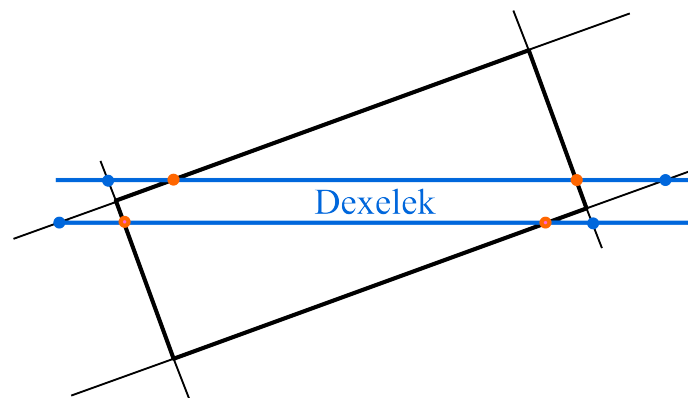
A hengerpalást által seprt térfogat előállításához szükséges egy olyan ferde hasáb, amely a 3.2c. ábrán lévő zöld téglalaphoz hasonlóan áll elő. A haladás kezdeti és végponti helyzetben lévő hengereket nem tartalmazza, csak a hengerpalást mozgás irányú érintői által határolt területet.

A ferdehasáb két szerszámhelyzettel, a szerszám sugarával és a szerszám magasságával definiált. Ezeket a szerszámhelyzeteket jelölje mozgásirány szerint növekvő számozással P_1 és P_2 pont (3.7. ábra). A hasáb két szemközti csúcsát jelölje pl. C_1 és C_2 . Első lépésként kiszámoljuk ezeket a pontokat. A 3.7. ábrán C_1 úgy áll elő, hogy a $\overrightarrow{P_1 P_2}$ xy síkra vett vetületével merőleges $\mathbf{e}_{P_1, C_1}$ egységvektor irányába r távolságnyt haladunk a P_1 pontból. Majd a piros nyíl mentén haladva eljuthatunk a C_2 pontba: z irányba h távolságnyt, majd innen $\overrightarrow{P_1 P_2}$ vektorral tovább, és $-2 \cdot r$ távolságot $\mathbf{e}_{P_1, C_1}$ irányába.



3.7. ábra. Hengerpalást által seprt térfogat

Ezekből a pontokon átmenő síkokon vannak a hasáb lapjai. Ha mindegyik síkkal vesszük a dexelek metszéspontját, akkor általános esetben hat darab pontot kapunk.



3.8. ábra. Dixel-hasáb metszéspontok szemléltetése 2D-ben

Ezeket növekvő sorrendbe rendezve a középső kettő adja meg a hasábok lapjain elhelyezkedő pontokat (3.8. ábra).

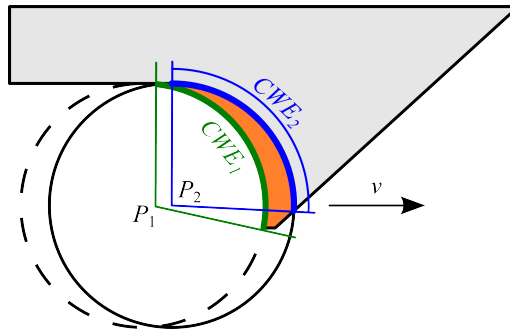
A fent ismertetett módszer a hasáb elágazásmentes (if nélküli) generálását teszi lehetővé, amely majd a jövőben a program videokártyás implementációját teszi lehetővé. (Az előző egyszerű primitívek esetén is elágazásmentesen volt lehetőség a generálásra.)

4. fejezet

CWE számítás

A CWE a Cutter-Workpiece Engagement angol szó rövidítése, és a megmunkálás során létrejövő szerszám-munkadarab kapcsolatba lépési viszonyát, az átfogási szög alakulását jellemzi (4.1. ábra). TDK munkánk, így a program elsődleges célja ennek meghatározása. Mivel a dixel módszer a testek egy diszkretizált reprezentációja, ezért a CWE számítása is diszkretizált formában történik. A CWE-t fel lehet használni különböző szerszámgeometriák esetén erőszámításra, és ezen keresztül lehet forgácsolóerőre optimalizálni a megmunkálást, illetve lehetőség van a felületminőség, és a kialakuló szerszámgéprezgések becslésére is.

Mivel a szerszám pályája felbontása is diszkretizált, ezért pontosabb CWE becsléshez átlagolni érdemes. Ezt a legegyszerűbben úgy tehetjük meg, ha két szerszám-pályapont közötti CWE-t súlyozzuk a szerint, hogy csak a lépés eleji állapothoz (P_1 : CWE_1), csak a lépés végi állapothoz (P_2 : CWE_2), vagy mindkét állapothoz tartozik a CWE adott részlete (4.2. ábra).



4.1. ábra. CWE egy szimulációs lépésben

4.1. CWE előállítás

Vegyünk egy $[0, 2\pi]$ szögtartomány $\Delta\varphi$ -vel való felosztását. Minden így kapott tartományt sorszámozhatunk: 0. szakasz: $0-\Delta\varphi$; 1. szakasz: $\Delta\varphi-2 \cdot \Delta\varphi$, stb. Ehhez a felosztáshoz előállítható egy egész számokat tartalmazó vektor (ezek lesznek az oszlopok). A szerszám magasságát pedig szintén egy Δh osztással felbonthatjuk a szögfelosztáshoz hasonlóan (a programban memóriatakarékossági okok miatt a forgács dixelreprezentációjának legmagasabb pontját vesszük a szerszám legmagasabb pontjának). Így a magasság mentén létrehozható egy vektor (ezek lesznek a sorok),

ami a szögfelbontás alapján létrehozott egész számokat tartalmazó vektorokat tartalmazza (és így összességében egy mátrixot kapunk). Az így előállt mátrix neve legyen CWE-mátrix (4.3b).

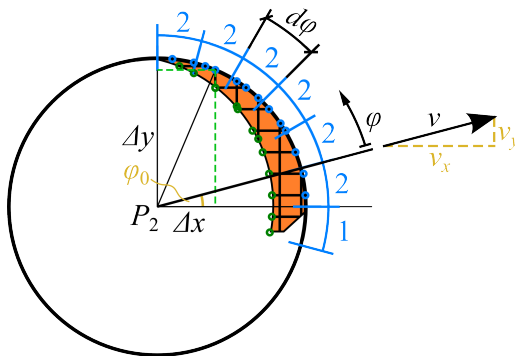
Ezután a mátrix elemeinek feltöltése következik. Ehhez szükséges egy olyan dixel test, ami a munkadarab és az adott lépéshez tartozó seprő térfogat metszete (4.3a. ábra). Ennek minden egyes dixellel meghatározott pontjáról eldöntjük, hogy melyik szerszámhelyzethez tartozik. Ha a P_2 -vel jelölt helyzethez tartozik, akkor a pont xy síkba vett vetülete P_2 -től r távolságra van (r a szerszám sugara). Ha a pont az P_2 szerszámhelyzethez tartozik, akkor haladási irányhoz képesti szöget kiszámítjuk (4.1.1) - (4.1.3) összefüggések felhasználásával (4.2. ábra).

$$\varphi_0 = \arctan \left(\frac{v_y}{v_x} \right) \quad (4.1.1)$$

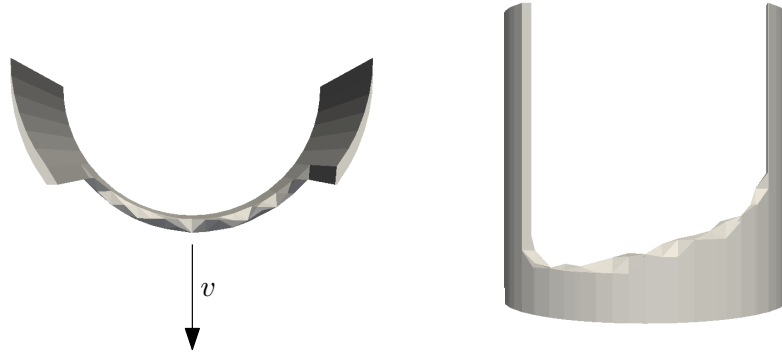
$$\varphi = \arctan\left(\frac{d_y}{d_x}\right) - \varphi_0 \quad (4.1.2)$$

$$\varphi_{CWE} = \varphi + \pi \quad (4.1.3)$$

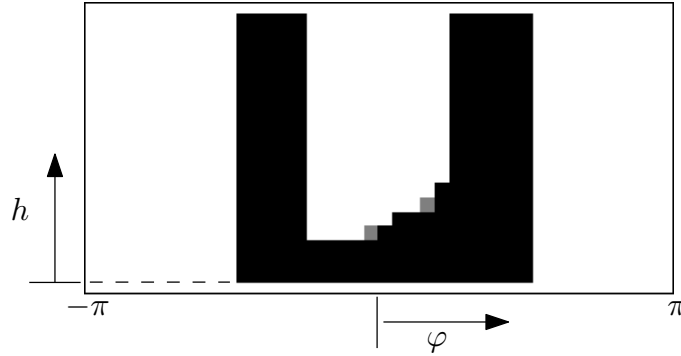
Az így kapott szöveget φ_{CWE} elosztjuk a $\Delta\varphi$ -vel, vesszük az eredmény alsó egészrészét, és így kapunk egy egész számot, amit jelöljünk i_φ -vel. A (4.1.3)-ban φ -ban hozzáadott π -re azért van szükség, mert így a CWE mátrix első oszlopa a $-\pi$ szöghöz tartozik (a mátrix középső oszlopában lévő elemei fognak a haladási irányhoz képest kis szöget bezárni). A pont z koordinátájából kivonjuk a P_2 szerszámhelyzet z koordinátáját, és ezt elosztjuk Δh -val, és ennek is vesszük az alsó egészrészét, amelyet most i_z -val jelöljünk. Az i_φ általában a CWE mátrix i_z -adik sorában kijelöl egy elemet. Viszont az i_φ lehet negatív, illetve nagyobb mint a CWE mátrix sorainak hossza az (4.1.2)-ban kivont φ_0 miatt. Ahhoz hogy az i_φ jó elemet jelöljön ki, annak a sorhosszokkal vett modulóját kell venni (osztás utáni maradékát). A (4.1.1) és (4.1.2) numerikus megvalósításánál az $\text{atan2}(x, y)$ függvényt használtuk fel, amely $[-\pi, \pi]$. közötti szöget határoz meg.



4.2. ábra. CWE számítás megvalósításának vázlata



(a) Forgácsdixel 3D reprezentációja



(b) Forgácsdixelből készült CWE mátrix

4.3. ábra. CWE 3D-ben

4.2. CWE felbontás megválasztása

Ahhoz, hogy minél pontosabban meg tudjuk határozni a CWE-t, ahhoz a lehető legnagyobb felbontással kell a $[-\pi, \pi]$ tartományt felbontani. Viszont ha túl nagy felbontást választunk, akkor lehetsége az, hogy pont két dixel által meghatározott pont közé esik a szakasz, és úgy tűnik, mintha azon a rövid szögtartományon a szerszám nem dolgozna, annak ellenére, hogy dolgozik (4.4a. ábra). Ezért van egy alsó határa a CWE $\Delta\varphi$ szögfelbontásának (4.4b. ábra):

$$r \cdot \Delta\varphi_{min} > \sqrt{2} \cdot res_{dex} \quad (4.2.1)$$

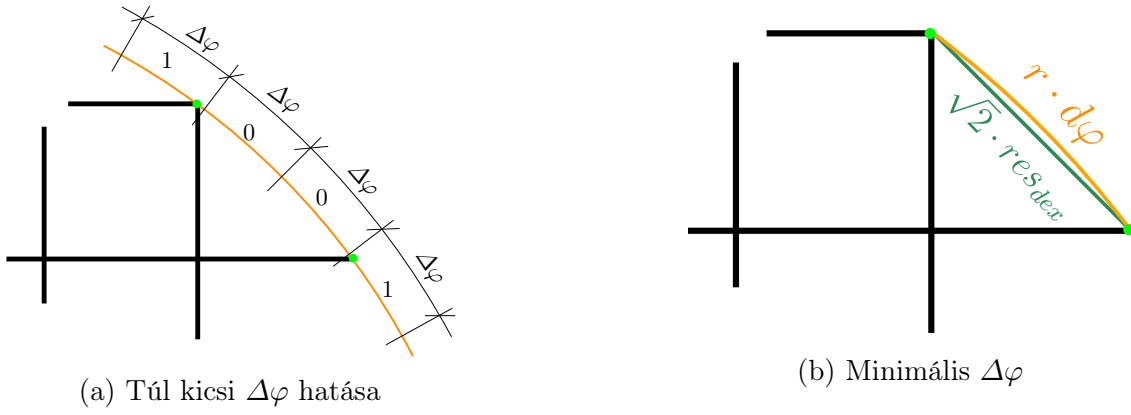
$$\Delta\varphi_{min} > \frac{\sqrt{2} \cdot res_{dex}}{r} \quad (4.2.2)$$

Gyakorlati szempontokból praktikusabb azt megállapítani, hogy a CWE mátrix sorai milyen hosszúak legyenek (az elemek számát jelölje a φ_{finess}):

$$\varphi_{finess} < \frac{2 \cdot \pi}{\Delta\varphi_{min}} = \frac{2 \cdot \pi \cdot r}{\sqrt{2} \cdot res_{dex}} = \frac{\sqrt{2} \cdot \pi \cdot r}{res_{dex}} \approx \frac{4.44 \cdot r}{res_{dex}}. \quad (4.2.3)$$

Ezek alapján egy célszerű választás a

$$\varphi_{finess} := \frac{4 \cdot r}{res_{dex}}. \quad (4.2.4)$$

4.4. ábra. $\Delta\varphi$ méretének megválasztása

A program ez alapján számol egy $\varphi_{fitness}$ -t, de megadható saját kezűleg egy $\varphi_{maxfitness}$ érték, amelyek közül a program a kisebbet választja, ha nincs szükségünk ilyen mértékű felbontásra.

A magasság alapján is érdemes olyan felosztást választani, hogy az a dexelben lévő pontok közötti távolságnál nagyobb legyen. A pontok között maximum res_{dex} felbontás van. A program megnézi, hogy a forgácsdexel legalsó, és legfelső pontja között mekkora a távolság ($h_{forgács}$), és ennek alapján számol egy Δz felbontást ((4.2.5) - (4.2.8)).

$$h_{forgács} = z_{min} - z_{max} \quad (4.2.5)$$

$$z_{fitness,0} = \left\lfloor \frac{h_{forgács}}{res_{dex}} \right\rfloor \quad (4.2.6)$$

$$z_{fitness} = z_{fitness,0} + 1 \quad (4.2.7)$$

$$\Delta z = \frac{h_{forgács}}{z_{fitness}} (> res_{dex}) \quad (4.2.8)$$

Az így meghatározott CWE mátrix alapján már számíthatóak az x , y , z irányú forgácsoló erők (lásd.: következő fejezet).

Ha a szerszámot jellemző dinamikai tulajdonságok is ismertek (pl tool-tip FRF), akkor az x , y irányú erők alapján felületi hibák és stabilitási tulajdonságok is számíthatóak. A dolgozat során csak az erőszámítással foglalkozunk, de további tervek között szerepel a stabilitás és a felületi tulajdonságok számítása is.

5. fejezet

Erőszámítás

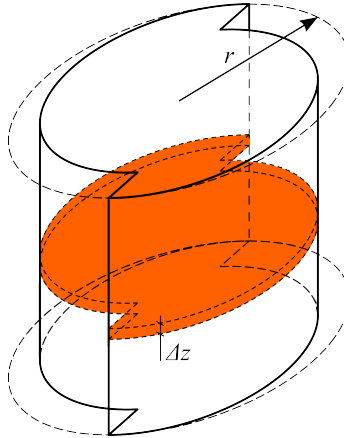
5.1. Forgácsoló erő számítás alapja

A forgácsolóerőszámítás során egy egyszerű, egyenes élű szerszámot feltételezünk (5.1. ábra). Ezt a szerszámot a felbonthatjuk Δz magasságú szeletekre (érdeemes a CWE mátrix sorai alapján), ezekre kiszámolható, hogy mekkora erő hat, és ezek összegzéséből kaphatjuk meg a teljes szerszámmra ható erőt. Az egy szeletre ható erő számítására a háromnegyedes szabályt alkalmaztuk:

$$\Delta F_r = k_r \cdot h^{\frac{3}{4}} \cdot \Delta z, \quad (5.1.1)$$

$$\Delta F_t = k_t \cdot h^{\frac{3}{4}} \cdot \Delta z, \quad (5.1.2)$$

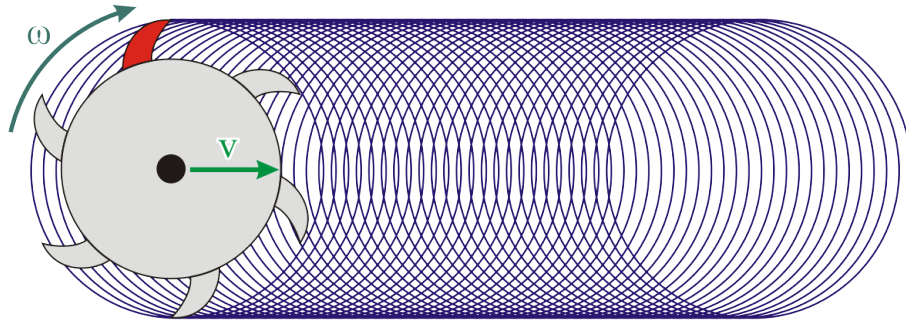
ahol a ΔF_r és a ΔF_t egy élszegmensre ható radiális- és tangenciális erőkomponensek, k_r és k_t a radiális- és tangenciális forgácsolási együtthatók (vagy fajlagos forgácsoló erő), h a forgácsvastagság, Δz a fogásmélység (szelet magassága).



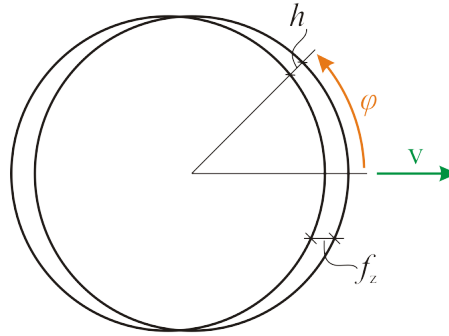
5.1. ábra. Egyenes élű szerszám vázlata

Feltételezhetjük, hogy a szerszám élei jó közelítéssel körpályán haladnak, mivel a szerszám éleinek előtolásból származó sebessége jóval kisebb, mint a szerszám forgásából származó sebessége. Az 5.2. ábrán szemléltettük, hogy a szerszámél valószínűleg pályája nagyjából egymás után következő körökkel is jól leírható lenne. Így a közelített forgácsvastagság:

$$h = f_z \cos \varphi, \quad (5.1.3)$$



(a) Szerszám pirossal jelölt élének pályája [4]



(b) Közelítő forgácsvastagság [4]

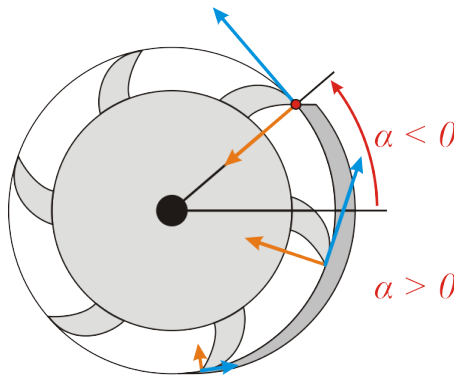
5.2. ábra. Forgácsvastagság alakulása

ahol f_z a fogankénti előtolás $[\frac{\text{mm}}{\text{fog}}]$ -ban:

$$f_z = \frac{v_f}{N} \cdot \frac{1}{2 \cdot \pi \cdot n_{orsó}}, \quad (5.1.4)$$

ahol v_f az előtolási sebesség $[\frac{\text{mm}}{\text{perc}}]$ -ben, N a szerszámélek száma, $n_{orsó}$ a szerszámorsó fordulatszáma $[\frac{\text{perc}}{\text{ford}}]$ -ben.

A „szerszámszeletre” ható eredő erőt az összes fogra ható erő vektoriális összege adja. Ahhoz, hogy az egy fordulat alatt bekövetkező teljes erőlefutásról információt kapjunk, a 5.3. ábrán, piros ponttal jelölt élhez tartozó α szöggel végig kell sűrolni a teljes $[-\pi, \pi]$ szögtartományt egy választott $d\varphi$ lépésközzel, az összes élre ki kell számolni a forgácsolóerőt, majd vektoriálisan összegezni.



5.3. ábra. Fogakra ható erők [4]

5.2. Forgácsoló erő számítás megvalósítása

Adott egy N élű szerszám. A forgácsoló erőt kiszámítjuk minden egyes élre és vektorhelyesen összegezzük (i index), majd mindezt megtesszük minden egyes szerszámszeletre, majd ezt is összegezzük (j index):

$$\mathbf{F}(\alpha) = \sum_{j=1}^{n_{szelet}} \sum_{i=1}^N \mathbf{F}_{i,j}(\varphi_i(\alpha)), \quad (5.2.1)$$

ahol n_{szelet} a szeletek száma, amely megegyezik a CWE mátrix sorainak számával.

Az élekre ható erőkomponensek (globális koordináarendszerben):

$$\varphi_i = \alpha + i \cdot \frac{2 \cdot \pi}{N} - \pi, \quad (5.2.2)$$

$$F_{i,j,x}(\varphi_i) = g_{i,j} \cdot h(\varphi_i)^{\frac{3}{4}} \cdot \Delta z \cdot (k_r \cdot \cos(\varphi_i) - k_t \cdot \sin(\varphi_i)), \quad (5.2.3)$$

$$F_{i,j,y}(\varphi_i) = g_{i,j} \cdot h(\varphi_i)^{\frac{3}{4}} \cdot \Delta z \cdot (k_r \cdot \sin(\varphi_i) + k_t \cdot \cos(\varphi_i)), \quad (5.2.4)$$

és $\alpha \in [-pi, -pi + \frac{2\pi}{N}]$ intervallumban változik általunk választott $\Delta\varphi$ lépésekkel. Láthatjuk, hogy α -val elegendő a teljes szögtartomány N -ed részét lefedni, mert utána az erőlefutás ismétlődik (egyenletes élelosztást feltételezünk). A $g_{i,j}$ CWE mátrix (i, j)-edik eleméből származtatható, a következőképpen:

$$g_{i,j} = \begin{cases} 1 & | \quad CWE_{i,j} \neq 0 \\ 0 & | \quad CWE_{i,j} = 0 \end{cases}$$

ahol $CWE_{i,j}$ a CWE mátrix i -edik oszlopában (szöghelyzet) és j -edik sorában (magasság) lévő elemére utal.

A főorsó által leadandó nyomatékot, pedig a következőképpen számolhatjuk:

$$M_{orsó}(\alpha) = r \cdot \sum_{j=1}^{n_{szelet}} \sum_{i=1}^N k_t \cdot h(\varphi_i)^{\frac{3}{4}} \cdot \Delta z \quad (5.2.5)$$

5.3. Előtolási sebesség maximalizálása

A termelékenység növelése érdekében érdemes a szerszámgépeknél a lehető legnagyobb előtolási sebességet alkalmazni az alkatrészek megmunkálása során. Viszont az alkalmazható pályasebességet lekorlátozza a forgácsolás során fellépő erők, nyomatékok nagysága, és a szerszámgép teherbírása, valamint a szerszám típusára megengedett maximális fogankénti előtolás. A program képes a terheléseket számolni, így tudja az előtolási sebességet maximalizálni a szerszámgép terhelhetőségének ismeretében.

Az előtolási sebességek maximalizálásánál figyelembe kell venni az orsó maximális teljesítményét (a főorsó által leadott nyomatékon keresztül) ($P_{határ}$), a szerszám terhelhetőségét ($F_{határ}$), a szerszámgép főorsója által leadható legnagyobb nyomatékot ($M_{határ}$), és a szerszám által biztosítható legnagyobb előtolási sebességet (gyorsmeneti sebesség - $v_{f,gyorsmenet}$).

A program első futtatása után vesszük minden pályaponthoz tartozó számított erő abszolút értékének, és a nyomatéknak a maximumát (F_{max} , M_{max}). Mivel az erő jó közelítéssel (5.1.2) alapján változik, ezért a maximális előtolást a következőképpen számolhatjuk:

- Erőmaximumhoz:

$$v_{f,új} = v_{f,régi} \cdot \left(\frac{F_{határ}}{F_{max}} \right)^{\frac{4}{3}} \quad (5.3.1)$$

- Nyomatékmaximumhoz:

$$v_{f,új} = v_{f,régi} \cdot \left(\frac{M_{határ}}{M_{max}} \right)^{\frac{4}{3}} \quad (5.3.2)$$

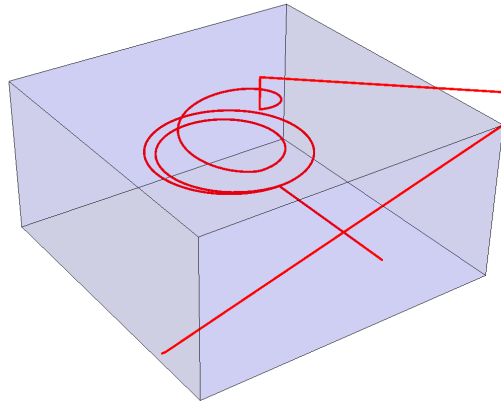
- Teljesítménymaximumhoz:

$$v_{f,új} = v_{f,régi} \cdot \left(\frac{P_{határ}}{M_{max} \cdot 2 \cdot \pi \cdot n_{orsó}} \right)^{\frac{4}{3}} \quad (5.3.3)$$

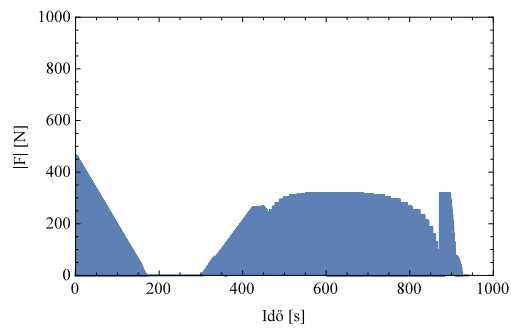
- Gyorsmeneti sebességhez:

$$v_{f,új} = v_{f,gyorsmenet} \quad (5.3.4)$$

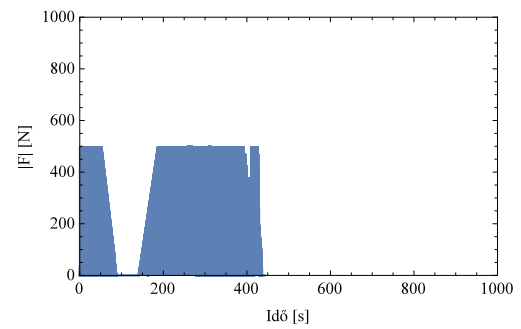
A számolt értékek minimuma lesz a végleges, optimalizált előtolási sebesség. Egy ilyen optimalizálásra láthatunk példát a 5.5. ábrán.



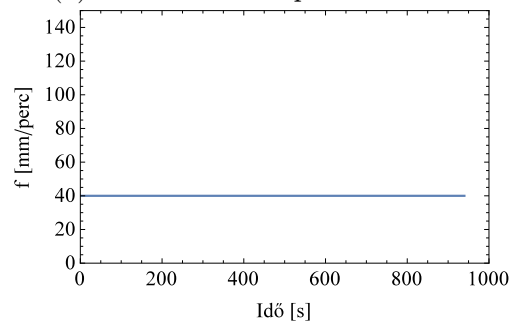
5.4. ábra. Példa szerszámpálya



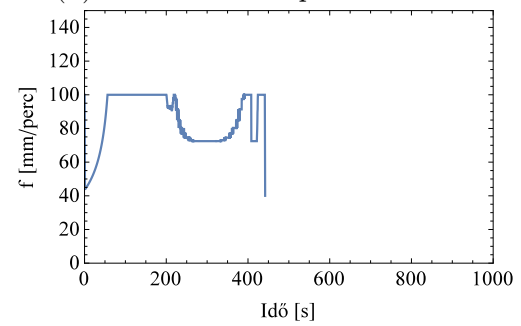
(a) Erő alakulása optimalás előtt



(b) Erő alakulása optimalás után



(c) Előtolási sebesség alakulása optimalás előtt



(d) Előtolási sebesség alakulása optimalás után

5.5. ábra. 5.4. szerszám-pálya jele

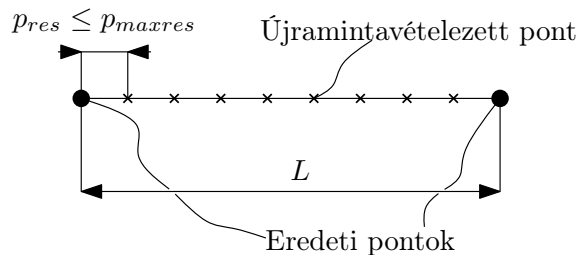
6. fejezet

Szerszámpályák

Ahhoz, hogy egy általános 3 tengelyes megmunkálást szimulálni tudjunk, szükség van egy szerszámpályára, amin végigvezethetjük a szerszámot. Ezt a pályát célszerű egy szabványos G-kódból beolvasni, amit akár egy valós szerszámgépen is lehet futtatni. Az egyszerűség érdekében csak a lineáris interpoláció parancsát használjuk fel, hiszen elég kis lépésekkel jól közelíthető a görbült pálya.

Programunkban a beolvasás szövegfájlból történik. A program soronként végighalad a fájlban majd reguláris kifejezések segítségével értelmezi az adott sort: megkeresi a pályapontok koordinátáit és az előtolási sebességet. Ezeket az adatokat egy listában tárolja. A fájl beolvasása után a pályát újramintavételezzük megadott méretű közel egyenlő szakaszokra, mivel így kaphatunk megfelelő pontosságú erőszámítást.

Az újramintavételezés során a szerszámpályát adott maximális p_{maxres} hosszúságú szakaszokra bontjuk fel. Ha két szerszámhelyzet távolsága nem egész számszorosa p_{maxres} -nek, akkor olyan minimális számú egyenközű egységekre bontjuk, amellyel p_{maxres} -nél kisebb osztást kapunk. (6.1. ábra).



6.1. ábra. Szerszámpálya újramintavételezése

Mivel a program tud előtolási sebességet is optimalizálni (lásd 5.3. fejezet), ezért van függvény a pályapontok G-kód formájában való fájlba írására is.

7. fejezet

Szimuláció

A szimuláció során az előzőleg ismertetett számításokat és eljárásokat végezzük el, melynek lépései sorrendben:

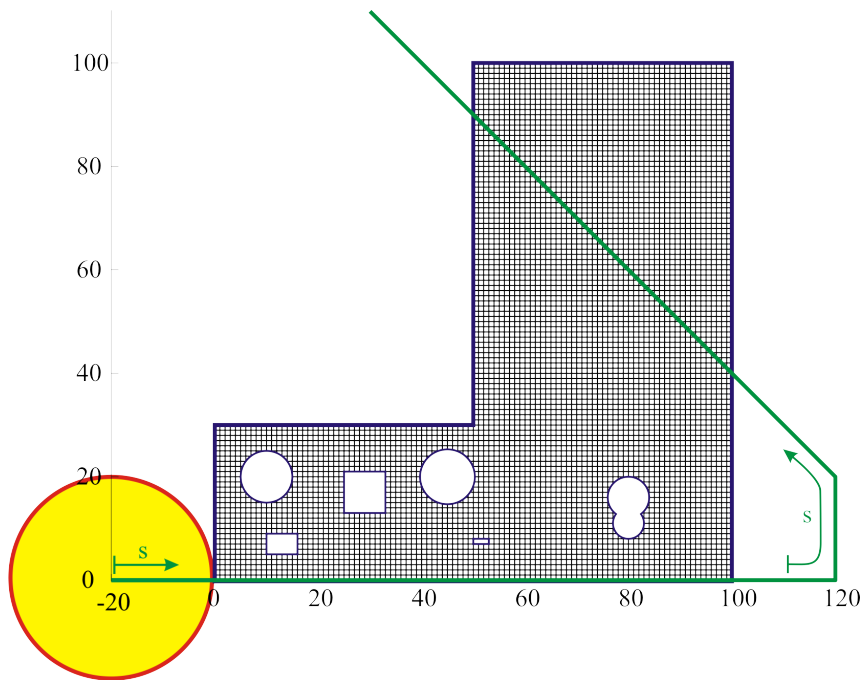
- Szerszám, előgyártmány és szerszámpálya beolvasása.
- Előgyártmány generálása és belőle tridexel generálása.
- Szerszám és seprert térfogatok generálása minden szimulációs lépéshez és tridexel generálás több szálon. (3.2. és 3.7. fejezetben részletesen leírva)
- Metszetek generálása és kivonások elvégzése (2. fejezetben)
- Minden metszetre CWE számítása (4. fejezetben)
- Minden CWE-re dinamika számítása (5. fejezetben)
- Pálya optimalizálása megadott feltételek alapján. (5. fejezetben)
- Adatok fájlba írása (módosított/eredeti G-kód, erőlefutás előtte/utána, eredmény munkadarab kiírása)
- 3D vizualizáció

7.1. Szimuláció pontossága, időigénye

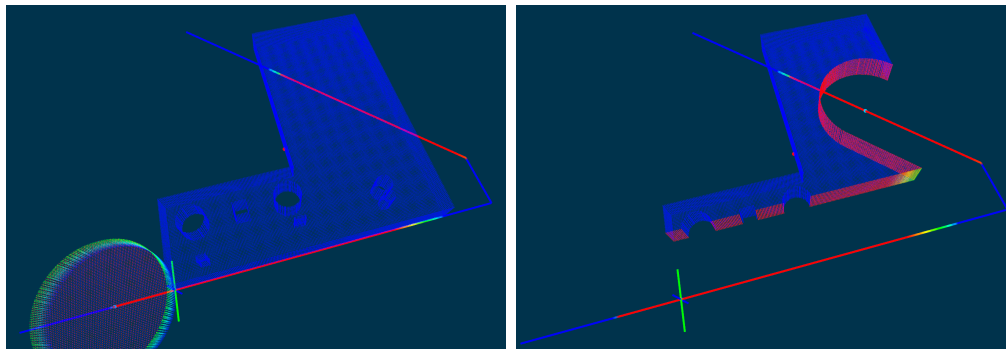
A szimuláció idejének, és pontosságának alakulását a felbontás függvényében a 7.1. brán látható megmunkálási példán keresztül került vizsgálatra. A vizsgálat eredmények a 7.2. ábrán látható táblázatban szerepelnek. Három-három különböző felbontással vizsgáltuk a P_{res} pályafelbontás és a D_{res} dexelháló felbontás hatását az szerszámra ható átlagos erő abszolút értékének átlagát.

Látható, hogy a forgácsolóerő számítási pontossága a pályafelbontástól nagyobb mértékben függ (7.2. ábra sorai: balról jobbra romlik a pályafelbontás). Ez azért van, mert a túl nagy lépésközű felbontás esetén a kisebb geometriai változásokat a szimuláció „átugorja”, így azok hatása az erőlefutás alakulásában sem jelenik meg. A dexelháló felbontásának hatása a keresztirányú dexelek miatt kevésbé jelentős.

A szimulációk idejének alakulása a 7.3. ábrán szerepel. Megfigyelhető, hogy a pályafelbontástól a szimulációs idő lineárisan függ, a dexelfelbontástól pedig jó közelítéssel négyzetesen. Ez abból adódik, hogy a pályafelbontás csökkentése esetén a kiszámolásra kerülő lépések száma növekszik (egyenes arányban a pályafelbontással). A dexelfelbontás csökkentése esetén pedig a dexelek száma négyzetesen növekszik, mert a dexelek a dexelsíkokon két irányban helyezkednek el.



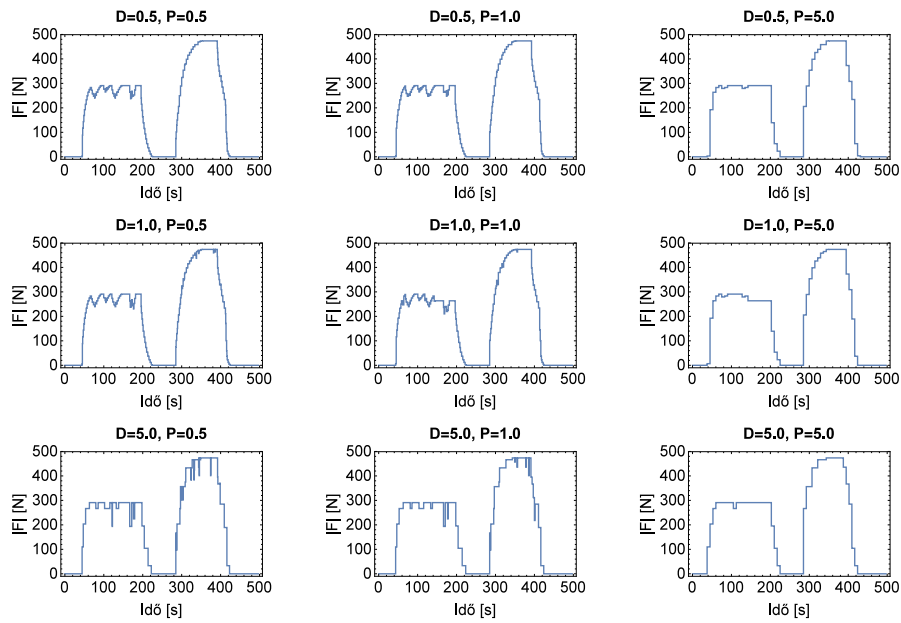
(a) Pontosságvizsgálat geometriájának vázlata [4]



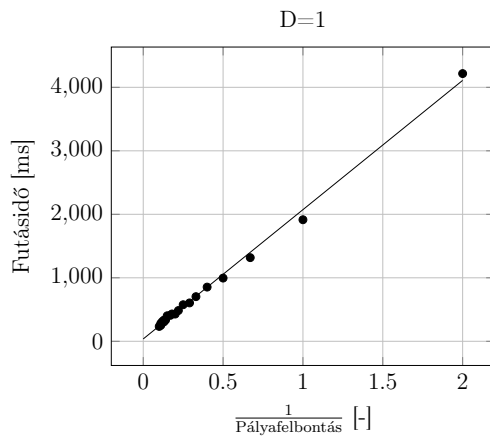
(b) Megmunkálás előtt

(c) Megmunkálás közben

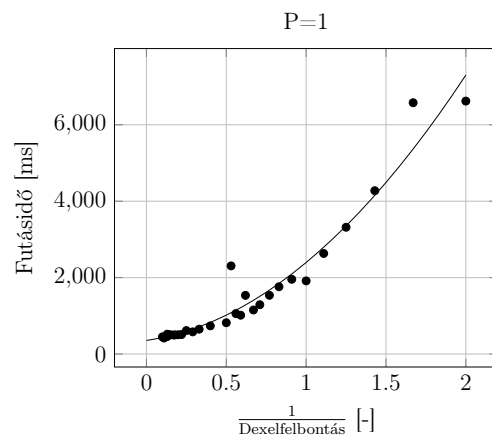
7.1. ábra. Pontosság- és időigényvizsgálathoz tartozó geometria



7.2. ábra. Átlagos erő alakulása dixel-, illetve pályafelbontás függvényében



(a) Idő alakulása pályafelbontás függvényében



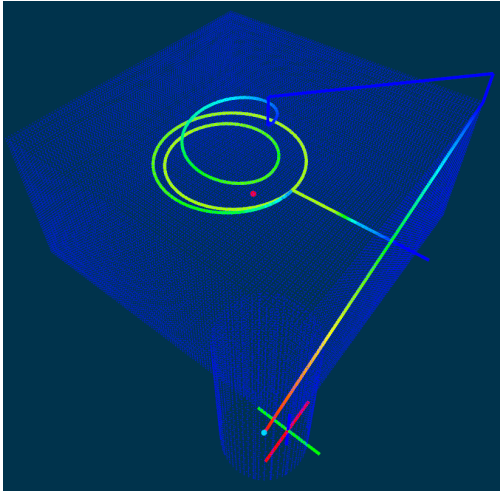
(b) Idő alakulása dexelfelbontás függvényében

7.3. ábra. Szimuláció idejének alakulása rögzített dixel- (D), illetve pályafelbontás függvényében (P)

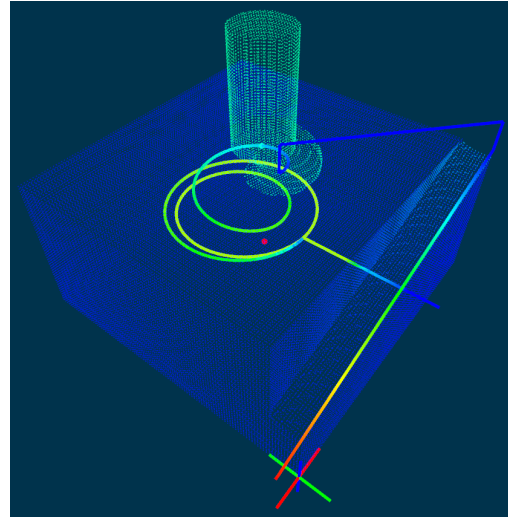
7.2. Szimulációra példa

A program működését az előző fejezetben ismertetett próbatesten kívül egy bonyolultabb szerszámpályájú eseten is személtetjük. Az 7.4a ábrán szerepel a kiindulási munkadarab geometria és a szerszámpálya. A szerszámpálya a pálya mentén a szerszámra ható maximális erő alapján van színezve. A 7.4b - 7.4d ábrákon látható, hogy hogyan alakul a munkadarab geometriája megmunkálás során. A pontok a szerint vannak színezve, hogy hányadik szimulációs lépés során jött létre az a felület, amelyhez tartoznak. Az 7.4e és 7.4f ábrán a megmunkálás utáni geometria látható: a 7.4e képen a szerszámra (és a felületre) ható maximális erő szerint, az 7.4f képen pedig a dolgozó szerszámmagasság (a szerszám mely magasságú szakasza dolgozott a felületen) szerint színezve.

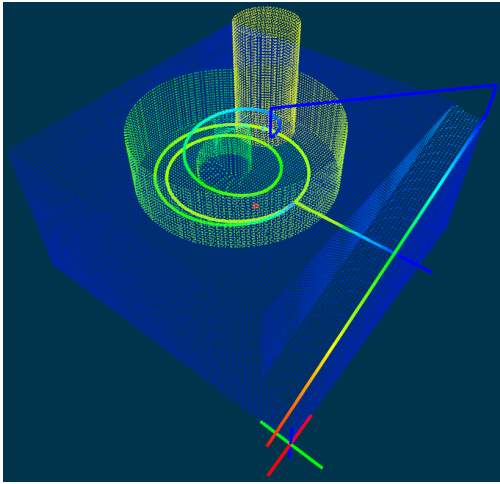
A 7.5. ábrán látható a futtatáshoz tartozó forgácsoló erő alakulása. Több dixel és pályafelbontáson is elvégeztük a futtatást. Jelen esetben a pályafelbontás hatása kevésbé érzékelhető (az oszlopok különböző pályafelbontáshoz tartoznak), mert a köríves szakaszok felbontása szinte mindig kisebb, mint a pálya újramintavételezéses felbontásához tartozó szakaszok hosszaként megadott maximális lépéshossz. A dexelfelbontás (a sorok a különböző dexelfelbontásokhoz tartoznak) viszont jelentős mértékben befolyásolta a számított erőlefutást: nagyjából 800s körül az erőlefutás „lépcsőzetessége” a dexelfelbontásból adódó diszkretizálási hiba hatása. Egyrészt a dexelek messzebb vannak egymástól kisebb felbontás esetén, másrészt a CWE számítás során is a dexelfelbontástól függ a szögfelbontás, és ez körpályán való marás esetén a 7.5. ábrán látható „lépcsős” erőlefutást ad. Ez a lépcsőzetesség csökkenthető a dexelfelbontás növelésével.



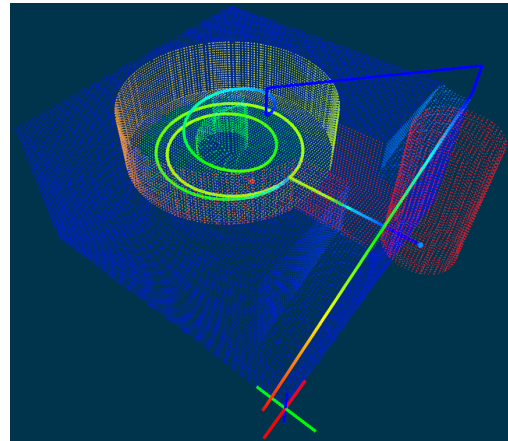
(a) Geometria vázlata



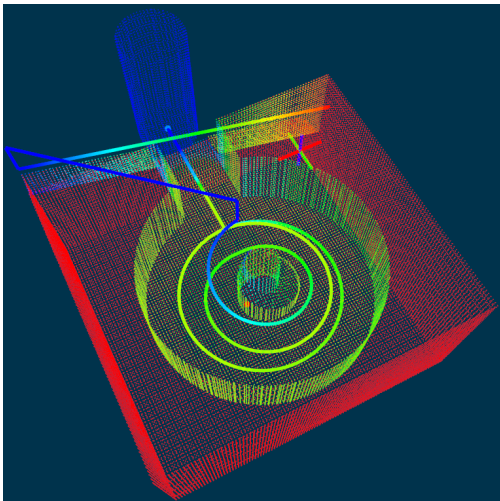
(b) Megmunkálás közben, megmunkálási lépés sorszáma szerint színezve



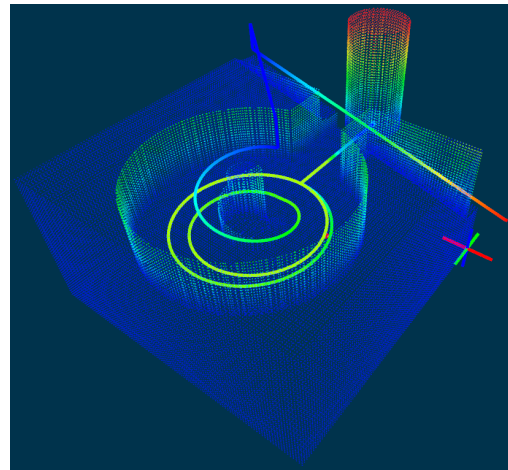
(c) Megmunkálás közben, megmunkálási lépés sorszáma szerint színezve



(d) Megmunkálás után, megmunkálási lépés sorszáma szerint színezve

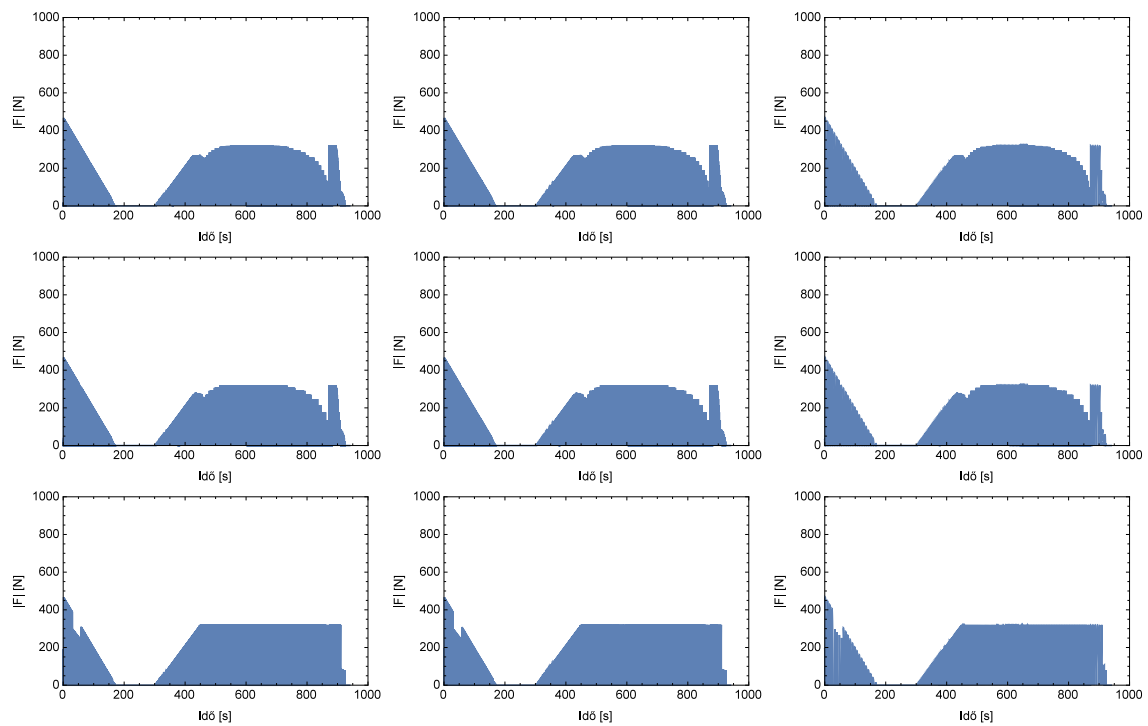


(e) Megmunkálás után, a szerszámra ható maximális erő szerint színezve



(f) Megmunkálás után, szerszámmagasság szerint színezve

7.4. ábra. Megmunkálási példa futás utáni vizualizációja



7.5. ábra. A szerszámmra ható erő abszolút értékének időjele különböző dixel- (sorok: $D = 0.5, 1, 5$) és pályafelbontások (oszlopok: $P = 0.5, 1, 5$) esetén 7.4. ábrán szereplő példafuttatáshoz

8. fejezet

Összefoglalás

A dolgozatban átvettük a dixel testmodellezési eljárás alapjait 2D és 3D esetekre. Egy már meglévő 2D-s megmunkálás szimulációjára alkalmas MATLAB kód [4] alapján újragondoltuk a dixel módszer megvalósítását, majd korszerű, objektumorientált megközelítéssel készítettünk egy 3 tengelyes megmunkálás szimulációjára alkalmas programot C++ környezetben. Nagy hangsúlyt fektettünk a dixel műveleteinek gyorsaságára, mivel már 3 dimenzióban modelleztük a testeket, így a számítási igény is számottevően megnövekedett.

A mozgó megmunkálós szerszám modellezésének problémakörével is foglalkoztunk. Elkészítettünk függvényeket, amely nem csupán a szerszám különböző pozíciókban való generálására képes, hanem két szerszámhelyzet közötti haladás során eltávolított anyagmennyiséget is figyelembe vettük. Ehhez egyszerű testprimitívek uniójával előállított úgynevezett seprert térfogatot hoztunk létre.

A szerszám mozgásának szimulációjához szükséges szerszámpályát egy szabványos G kódból olvassuk be. Ezt a szerszámpályát a szimuláció pontosságának növelése céljából újramintavételezzük. A programban optimalizált szerszámpályából G kód is generálható, amely közvetlenül használható egy szerszámgépen, és aminek segítségével a későbbiekben a megfelelő erőmérésekkel a szimulációk validálhatóak.

A forgácsoló erő számításához meghatároztuk a pályamenti CWE-t, azaz, hogy a megmunkálás során hogy alakul az átfogási szög a szerszám hossztengelye mentén. Ez alapján számoltuk a szerszámra ható erőket szerszámhelyzet függvényében, és amely segítségével akár szerszámra ható erők időbeli alakulására is becslés adható. Az így számolt forgácsoló erőt felhasználtuk a szerszámpályán való előtolási sebesség optimalizálására.

Vizsgáltuk a dolgozat során elkészített programmal való számítás időigényét a dixel- és a szerszámpálya felbontástól függően. A különböző felbontások során vizsgáltuk az szimulált erőlefutás alakulását.

A program nagy számban tartalmaz jól párhuzamosítható (elsősorban dixel) műveleteket, ezért ezeket tervezzük az nVidia cég általános célú grafikus processzorai (GPGPU - videokártya) által támogatott CUDA programozási környezetre átírni. Továbbá a jövőbeli célok közé között szerepel minél több funkció beépítése, mint a felületi hiba számítása és a stabilitási jellemzők meghatározása. Munkánk motivációja volt, hogy a program jól alkalmazható legyen a Budapesti Műszaki és Gazdaságtudományi Egyetem Gépészmérnöki Kar Műszaki Mechanikai tanszékén zajló, az Európai Kutatás Tanács (ERC) által támogatott szerszámgéprezgésekkel kapcsolatos kutatások („SIREN”) során is.

Irodalomjegyzék

- [1] Tukora, B., 2012, *Material removal simulation and cutting force prediction of multi-axis machining processes on general-purpose graphics processing units*, Phd. Thesis, Budapest University of Technology and Economics.
- [2] Altintas, Y., Kersting, P., Biermann, D., Budak, E., Denkena, B., and Lazoglu, I., 2014, „Virtual process systems for part machining operations,” *CIRP Annals - Manufacturing Technology*.
- [3] Bachrathy, D., 2006, *A study of the milled surface in the dynamics of milling*, MSc. Thesis, Budapest University of Technology and Economics.
- [4] Sykora, H., 2013, *Forgácsoló erő optimalizálása dixel elemekkel*, BSc. Thesis, Budapest University of Technology and Economics.
- [5] Ren, Y., Zhu, W., and Lee, Y.-S., 2008, „Feature conservation and conversion of tri-dixel volumetric models to polyhedral surface models for product prototyping,” *Computer-Aided Design and Applications*.
- [6] Mathworld, 2014, „Point-line 2d distance,” URL <http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>.