UNIVERSITATEA "POLITEHNICA" DIN TIMIŞOARA
FACULTATEA DE MECANICĂ
Catedra de Tehnologia Construcţiilor de Maşini

Ing. Dan Micşa

TEZA DE DOCTORAT

Metode de generare, conversie, import, export, analiză şi fabricaţie optimizată a suprafeţelor discrete

Conducător ştiinţific :
Prof. dr. ing. George Drăghici

1998

Mulţumiri

Mulţumesc în primul rând părinţilor că s-au gândit să mă facă şi să mă crească independent şi zvăpăiat, cu înţelegere pentru afecţiunile mele cam "exotice" pentru calculatoare, la începutul anilor '80.

Mulţumesc soţiei pentru liniştea, înţelegerea şi sprijinul oferit în redactarea acestei lucrări. Mulţumesc fiicei mele, care nu m-a lăsat să dorm nopţile şi să lucrez zilele, oferindu-mi timp berechet pentru a gândi.

Autorul se simte dator de a le mulţumi tuturor dascălilor care i-au călăuzit sinuosul drum spre altarul ştiinţei.

Acum, din punct de vedere ştiinţific, primele gânduri de recunoştintă se îndreaptă spre conducătorul tezei, **prof. dr. ing. George Drăghici,** care a îndrumat munca doctorandului cu multă competenţă (via email), punând la dispoziţie atât cunoştinţele sale, cât şi un material bibliografic personal foarte preţios prin conţinut, sferă de cuprindere şi mai ales actualitate.

Gânduri de recunoştinţă se îndreaptă către **prof. dr. ing. Gavril Urdea, prof. dr. ing. Nicolae V. Ivan, prof. dr. ing. George G. Savii,** care m-au onorat, în calitate de preşedinte, respectiv referenţi, în comisia de doctorat.

Alte gânduri se îndreaptă spre **prof. dr. ing. Constantin Stăncescu,** cel care, în decursul anilor, şi-a găsit întotdeauna timp să discute şi să publice în prima revistă de CAD din România, "*Hello CAD fans*", o mulţime de articole despre rezultatele şi produsele autorului, precum şi pentru onoarea de a mă asista în calitate de referent.

Mulţumiri speciale **şef lucr. ing. Mircea Şelariu,** care a fost oricând dispus să stea şi să analizeze cu mine diferite aspecte legate de această lucrare.

De asemenea, gândurile mi se îndreaptă spre prietenul, **ing Dorin Dascălu,** care în ultimii 5 ani a fost primul care a testat şi şi-a expus observaţiile valoroase despre utilitatea şi posibilele căi de îmbunătăţire ale algoritmilor expuşi.

Nu poate fi omis din aceasta sumară înşiruire prietenul şi exvecinul meu, **ing. Mircea Bunea,** care mi-a hrănit spiritul cu o mulţime de cunoştinţe în domeniul DTP (DeskTopPublishing - paginaţie computerizată sau aranjare în pagină) şi m-a ajutat la corectarea acestei lucrări.

Mulţumesc conducerilor intreprinderilor *ROMACOST, ELBA* şi *CAROM* pentru sprijinul material şi ştiinţific acordat în perioada 1991-1993.

Mulţumesc **Domnului ing. David Boucher** - director tehnic la ***Pathtrace Engineering Systems Ltd,*** care mi-a oferit oportunitatea de a-mi continua studiile şi de a găsi algoritmilor şi aparatelor matematice o finalitate comercială.

Dacă ar fi să menţionez un singur lucru care m-a ajutat imens în această muncă de analiză şi sinteză, acesta ar trebui să fie, bineînţeles, ***INTERNETUL,*** cu o mulţime de aparate matematice, descrieri de produse, oameni fără principii, plini numai de cunostinţe şi umor, care de-a lungul vremii mi-au fost de un real ajutor, împingându-mă să continui şi să finalizez această lucrare.

Cuprins

1. Introducere

Prezenta teză de doctorat reprezintă rezultatul unei activităţi de cercetare de peste 7 ani a autorului, în domeniul proiectării şi fabricaţiei asistate, desfăşurată în România, la BillaSoft srl, o mică firmă născută din dorinţa de a face ceva pentru proiectarea şi fabricaţia asistată românească, precum şi în Marea Britanie, la Pathtrace Engineering Systems ltd, unde i s-a oferit oportunitatea de a-şi continua cercetările, în vederea unei finalizări comerciale a rezultatelor. În toată această perioadă s-a încercat să se dezvolte, implementeze şi testeze o nouă modalitate generică de concepţie, introducere, modelare, vizualizare, analizare, fabricare, optimizare, simulare şi verificare a suprafeţelor.

Lucrarea încearcă să îmbine pregătirea şi cunoştinţele din domeniul de specializare al autorului, cel mecanic, cu profundele pasiuni pentru informatică şi matematică. Astfel, subiectul abordat se află în zona de graniţă dintre aceste trei impresionante şi deopotrivă fascinante ştiinţe, îmbinând cunoştinţele mecanice, cu algoritmi, tehnici de programare şi limbajele de programare din ce în ce mai eficiente, într-o evoluţie de o dinamică impresionantă. Teza se doreşte a fi una cu profunde aplicaţii practice, fiind implementată în două produse destinate fabricaţiei asistate (TechnoPack BillaSoft, peste 50 utilizatori şi EdgeCAM Pathtrace ltd, peste 12.000 utilizatori); acesta este şi principalul motiv pentru care s-a optat pentru prezentarea metodelor într-un limbaj de tranziţie între matematică şi limbaje de programare.

Toate capitolele au o structură asemănătoare, începând cu un subcapitol intitulat "**Introducere**", în care se va prezenta problematica propusă spre rezolvare şi un ultim subcapitol "**Concluzii**", în care se vor rezuma cele discutate de-a lungul capitolului respectiv. Aceeaşi structură unificată se doreşte a fi utilizată şi pentru întreaga teză, care începe cu "**Introducere**" şi se sfârşeşte cu "**Concluzii**". Penultimul capitol se va intitula "**Concepte introduse**" şi va rezuma în pseudocod cele discutate pe parcursul tezei (antetele conceptelor introduse).

În capitolul 2, intitulat "**Stadiul actual**", se vor sintetiza câteva aspecte, încercând să se cuprindă evoluţia temporară a unor domenii care au o influenţă importantă asupra problematicii. Se vor discuta, exemplifica şi comenta avantajele şi dezavantajele diferitelor metode şi tehnici de stocare şi analiză. Se va prezenta cronologic evoluţia limbajelor şi a tehnicilor de programare, pentru a justifica crearea şi utilizarea limbajului pseudocod în descrierea unificată a algoritmilor descrişi.

În capitolul 3, intitulat "**Metode de notaţie şi clase utilizate**", se va introduce un limbaj obiectual de tip pseudocod, în care se vor descrie algoritmii (tehnicile, metodele) şi care ajută la o eventuală implementare a conceptelor discutate într-un limbaj de nivel înalt orientat obiect; se vor descrie metodele de stocare a informaţiilor, încercându-se o prezentare cât mai concisă şi consistentă. Se vor pune bazele unei ierarhii de clase, care vor fi folosite în decursul lucrării, acestea fiind: *şirul de caractere (string), fişierul, întregul, booleanul, realul, punctul, curba, curbe superioare, plasa, suprafeţe superioare, masca, suprafaţa digitală, colecţiile (familiile) de obiecte.*

Toate tipurile de dată vor fi prezentate într-un mod organizat, ataşându-li-se atât operaţiile, cât şi metodele şi funcţiile asociate.

Nu se va aborda o prezentare exhaustivă a acestor tipuri de dată, ci doar crearea unui set decent şi relativ bogat de tehnici de operare cu data respectivă, tipurile enumerate fiind larg studiate şi implementate în toate sistemele de proiectare şi fabricaţie.

În capitolul 4, intitulat "**Metode de generare şi modelare**" se vor prezenta pe larg unele metode care stau la baza generării şi modelării suprafeţelor demulabile. De asemenea, se vor prezenta diferite metode de conversie din alte tipuri de dată, specifice importului din alte sisteme de proiectare şi fabricaţie, date importate în format plasă sau listă cu triunghiuri (forma cea mai simplă de export a solidelor).

De asemenea, se vor prezenta şi defini filtrele, se vor prezenta şi exemplifica scopul şi utilitatea lor în reducerea zgomotului introdus în diferiţi paşi de conversie şi analiză, sau datorat diferitelor inflexiuni particulare ale suprafeţei date spre conversie, analiză şi fabricaţie.

Se va descrie un set de algoritmi de modelare: cel al calculului înfăşurătorii şi racordării statice şi dinamice, algoritmi care nu sunt specifici numai modelării, ci şi calculului suprafeţei corecţie de sculă şi a suprafeţei de contact pentru scule generice de orice geometrie. Se vor face particularizările specifice capetelor de freză.

De asemenea, se vor prezenta în premieră trei metodologii noi concepute de către autor:

- ⇨ *algoritmul de import şi conversie a seturilor de puncte şi curbe furnizate fără nici o regulă;*
- ⇨ *algoritmul de calculare a înfăşurătoarei şi racordărilor cu forme de orice geometrie (un caz particular al acestora sunt capetele de sculă suprafeţe de revoluţie utilizate în frezare);*
- ⇨ *reţeaua neuronală pentru antrenarea cu date care nu cad în punctele reţelei, utilizabilă ca o metodă generică de import a tuturor datelor parametrice.*

În capitolul 5, intitulat "**Metode de conversie şi formate de import-export**", se vor prezenta câteva conversii ale suprafeţelor demulabile în reprezentările vectoriale uzuale altor sisteme de proiectare şi fabricaţie, pentru a da nu numai o consistenţă vizuală analizelor şi generărilor, ci şi o finalitate şi utilizabilitate în alte sisteme.

În prima parte a capitolului se vor prezenta metode de culegere a datelor şi de convertire a lor în formate de tip *plasă, solide faţetate,* reprezentări de tip *familie de curbe*, care sunt specifice generării fişierului NC.

Pe parcursul acestui capitol se vor introduce metode de creare a curbelor, proiecţie, offset inteligent şi export în formate simple ASCII, ca: *DXF, STL, CL, NC.* Nu se vor discuta formate evoluate ca IGES, STEP, VDA, pentru a nu îngreuna expunerea.

Acest capitol este de o importanţă notabilă în utilizarea suprafeţelor digitale, în orice sistem de proiectare şi fabricaţie. Se va crea un set nou de obiecte, specifice fiecărui tip de export în parte: *DXFOut, STLOut, NCOut, CLOut.* Aceste noi obiecte (specifice exportului) au fost implementate folosind o metodă unificată de prezentare, încercând să se ascundă detaliile fiecărui format în parte, să se prezinte exemple pentru fiecare format şi listingul asociat.

Se vor expune metode noi destinate conversiei şi exportului, metode care dau utilizabilitate suprafeţelor demulabile, legându-le de alte sisteme de proiectare, ca aparate matematice auxiliare de analiză sau conversie în format NC.

Se va prezenta, în premieră, un algoritm de conversie în curbe de nivel foarte fin (comparativ cu pasul suprafeţei digitale), care permite conversia suprafeţelor demulabile corecţie de sculă în format NC, asigurând erori de ordinul micrometrilor.

Un subcapitol aparte va fi rezervat expunerii problemei creerii unui post procesor generic GNCPP (Generic Numeric Control Post Processor) generator automat de tehnologie, o librărie dinamică foarte complexă, care are scopul de a genera fişier NC specific, virtual, pe orice echipament, optimizat pentru lungime şi timp de rulare.

În capitolul 6, ultimul capitol principal, generic intitulat "**Metode de analiză şi optimizare**", se vor cuprinde câteva dintre cele mai importante aspecte legate de analiza şi generarea optimizată a codului NC pentru fabricarea suprafeţelor demulabile pe maşini-unelte cu comenzi numerice, precum şi câteva tehnici de verificare şi simulare. Cum toate acestea sunt tehnici şi metode de analiză, natural ele îşi vor găsi locul în acest ultim capitol.

Se vor prezenta în premieră câteva contribuţii ale autorului, constând din metode noi de analiză, precum: calculul zonelor plane, calculul zonelor critice la frezarea de secţiuni paralele în planul XY, calculul materialului nefrezabil, calculul curbelor de egală rugozitate, metode pseudoadaptive de variere a avansului şi corecţiei de uzură în timp real, minimizarea mişcărilor în avans rapid, spiralele lui Billator, o metodă nouă de optimizare a traseelor echidistante, prin dublarea sau triplarea locală, o metodă de rezolvare a reţelelor de difracţie.

Pe lângă prezentarea contribuţiilor autorului se vor expune şi câteva metode clasice de generare de cod, considerându-se ca element de noutate metodele de generare a acestora (curbelor echidistante în XY şi Z) utilizând SD în acest domeniu.

Vor fi exemplificate alte concepte, cum ar fi interpolările superioare, eliminarea punctelor de inflexiune, eliminarea punctelor coliniare, doar cu scopul secundar de a da consistenţă şi calitate unei eventuale generări de cod NC.

În penultimul capitol, "**Concepte introduse**", vor fi enumerate toate clasele introduse, cu toţi operatorii şi algoritmii descrişi de-a lungul lucrării.

Ultimul capitol, "**Concluzii**", va sintetiza cele discutate, încercând să scoată în evidenţă contribuţiile autorului şi să puncteze direcţiile rămase deschise cercetării.

În "**Anexe**" se vor prezenta: *definiţiile unor concepte utilizate, abrevieri, legături web utile, specifice domeniului proiectării şi fabricaţiei.*

Multe referinţe făcute pe parcursul lucrării vor fi la pagini de web, documente "vii" care prezintă "în timp real" evoluţia aparatelor matematice discutate, nu documente "moarte" ca şi cele scrise (reviste, cărţi, manuale de prezentare sau utilizare).

Lucrarea se doreşte a fi în primul rând una de sinteză, în care se prezintă o metodă solidă, consistentă şi generică de *stocare, conversie, analiză, generare optimizată, simulare şi verificare* a suprafeţelor discrete demulabile, punându-se accentul pe modul de structurare şi prezentare a conceptelor, pentru a da un caracter cât mai practic expunerii.

2. Stadiul actual

2.1. Introducere

Dată fiind complexitatea şi varietatea tipodimensională, în cursul anilor s-a încercat ca în funcţie de stadiul respectiv de evoluţie al *aparatelor matematice, calculatoarelor, limbajelor de programare, echipamentelor cu comenzi numerice, maşinilor unelte, sculelor aşchietoare,* să se genereze diferite metode de *descriere, stocare, procesare, optimizare, verificare şi prelucrare,* în vederea fabricaţiei diferitelor repere complexe.

Se va încerca doar prezentarea într-o manieră evolutivă a **metodelor de stocare** utilizate în descrierea reperelor sau efectuarea analizelor, precum şi a **limbajelor de programare,** pentru a justifica construcţia şi utilizarea limbajului pseudocod întrebuinţat la expunerea metodelor şi tehnicilor folosite.

Va fi atins doar stadiul actual al evoluţiei proiectării şi fabricării asistate în Romania şi pe plan mondial.

Evoluţia altor domenii, care au o influenţă şi o implicaţie mai puţin importantă, şi care vor fi atinse doar tangenţial (calculatoare, echipamente cu comenzi numerice, maşini unelte, scule aşchietoare, pachete de plăci, teoria sistemelor, inteligenţa artificială, automate celulare, automate neuronale, automate moleculare, automate genetice, teoria dezastrelor, comunicaţii, sisteme de calitate, aparate de măsură şi control, tribologie, mecanisme şi organe de maşini, dispozitive), va rămâne nediscutată.

2.2. Evoluţia metodelor de stocare

Cea mai mare influenţă, pentru domeniului studiat, o au metodele de stocare (baza de date) şi tehnicile sau algoritmii specifici pentru fiecare tip de dată enumerat.

În ultimii 30 de ani au fost create diferite metode de stocare şi prelucrare a datelor în vederea proiectării şi fabricării. Se va încerca o enumerare succintă a acestora, în ordinea complexităţii reprezentării:

2.2.1. Seturi de puncte

❖ ***Definiţie:***

⇨ *Datele sunt stocate ca o colecţie de puncte în spaţiu.*

❖ ***Utilizare:***

⇨ *la discretizări de date provenite de la maşini de scanat, palpat sau cartografiat, la reprezentări moleculare 3D in chimie, in anii '70 – '80.*

❖ ***Avantaje:***

⇨ *sunt foarte simplu de importat şi exportat.*

❖ ***Dezavantaje:***

⇨ *inutilizabile în analiză şi fabricaţie, deoarece informaţii auxiliare, ca aceea de normală, sunt imposibil de obţinut (pentru executarea ofsetului);*

⇨ *sunt foarte dificil de interpretat şi convertit în alte reprezentări.*

❖ ***Exemple:***

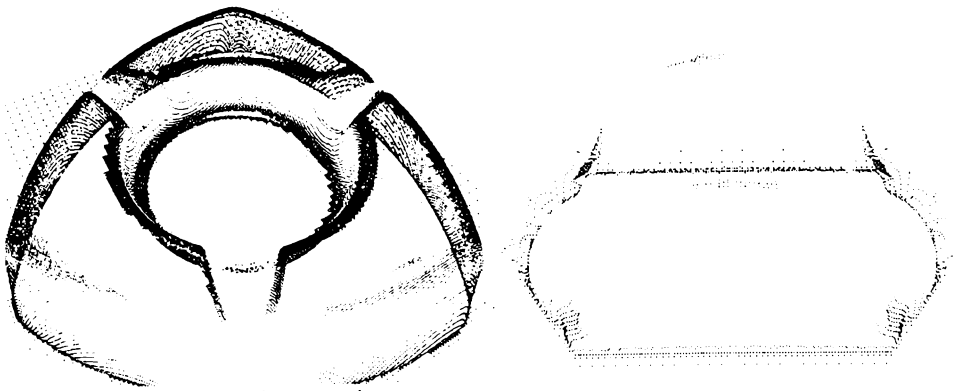


Figura 2.1 *Reprezentare şi stocare prin puncte*

2.2.2. Curbe şi suprafeţe analitice multiparametrice

❖ ***Definiţie:***

⇨ *Suprafeţele şi curbele analitice sunt acelea care pot fi descrise printr-un singur set de ecuaţii matematice (nu conţin seturi de ecuaţii pe porţiuni de curbă sau suprafaţă descrisă).*

❖ ***Utilizare:***

⇨ *la descrierea obiectelor matematice simple (sfere, tori, paralelipipede, conuri).*

❖ ***Avantaje:***

⇨ *sunt foarte precise (comparativ cu metodele de stocare discrete);*

⇨ *asigură precizie mare (teoretic infinită) la calcularea ariei, volumului, perimetrului, existând integrale directe pentru ecuaţiile lor;*

⇨ *sunt compacte, necesitând spaţii de stocare scăzute (de 10-100 ori mai mici decât cele discrete).*

❖ ***Dezavantaje:***

⇨ *tipurile de repere descrise sunt foarte limitate;*

⇨ *importul şi exportul este dificil, necesitând resurse însemnate în scrierea convertoarelor;*

⇨ *suprafeţele descrise sunt nesculpturale.*

❖ ***Exemple:***

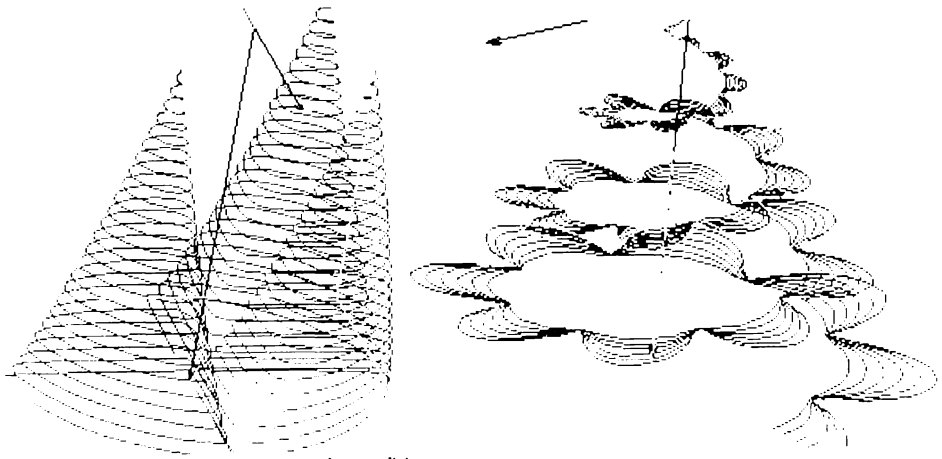


Figura 2.2. *Reprezentare prin curbe analitice*

2.2.3. Curbe la o înălţime dată 2D

❖ ***Definiţie:***

⇨ *Curbele plane sunt stocate ca o serie de* segmente de linie, arce de cerc, sau curbe superioare *în planul XY.*

La început au fost folosite la stocare doar segmente de linie, apoi au fost introduse segmentele de cerc (arce), iar mai nou sunt utilizate şi segmente din curbe de ordin mai mare (cubice).

❖ ***Utilizare:***

⇨ *pe scară largă în prezent, la definirea contururilor pentru strunjiri şi decupări prin electroeroziune, laser, jet de apă, jet de aer etc.Sunt prezente în toate sistemele de proiectare şi sunt suportate, în general, cam de toate formatele de import – export.*

❖ ***Avantaje:***

⇨ *sunt compacte, comparativ cu poliliniile;*

⇨ *sunt independente de toleranţă (stocare analitică).*

❖ ***Dezavantaje:***

⇨ *conversiile (*importul, exportul*) sunt relativ greu de executat;*

⇨ *calculele de* arie, perimetru *trebuie executate pentru fiecare tip de segment în parte.*

2.2.4. Polilinii plane

❖ ***Definiţie:***

⇨ *Poliniile plane sunt stocate ca o serie de segmente de linie în planul XY, fiind deopotrivă un caz particular al curbelor şi al poliliniilor spaţiale.*

❖ ***Utilizare:***

⇨ *pe scară largă în prezent, la definirea contururilor pentru* strunjiri şi decupări prin electroeroziune, laser, jet de apă, jet de aer etc. *Sunt prezente în toate sistemele de proiectare şi sunt suportate, în general, cam de toate formatele de import – export. Toate curbele şi suprafeţele sunt reprezentate vizual utilizând poliliniile.*

❖ ***Avantaje:***

⇨ *export şi import simplu;*

⇨ *procesare simplă a perimetrului, ariei;*

⇨ *sunt generice, descriind orice geometrie sub o toleranţă*

❖ ***Dezavantaje:***

⇨ *necesită spaţii mari de stocare;*

⇨ *sunt dependente de toleranţă;*

2.2.5. Suprafeţe extrudate

❖ ***Definiţie:***

⇨ *Suprafeţele extrudate reprezintă curbe extrudate dea lungul unei direcţii.*

❖ ***Utilizare:***

⇨ *Descrierea primordială a reperelor cu geometrie extrudată simplă 2½ D destinate frezării* (carcase, repere cu găuri, repere turnate/ forjate).

După corpurile simple sunt cele mai simpe metode de stocace a solidelor.

❖ ***Avantaje:***

⇨ *simplu de importat, exportat, procesat şi stocat.*

❖ ***Dezavantaje:***

⇨ *geometriile descrise sunt simple, neoferind posibilitatea descrierii volumelor sculpturale.*

❖ ***Exemple:***

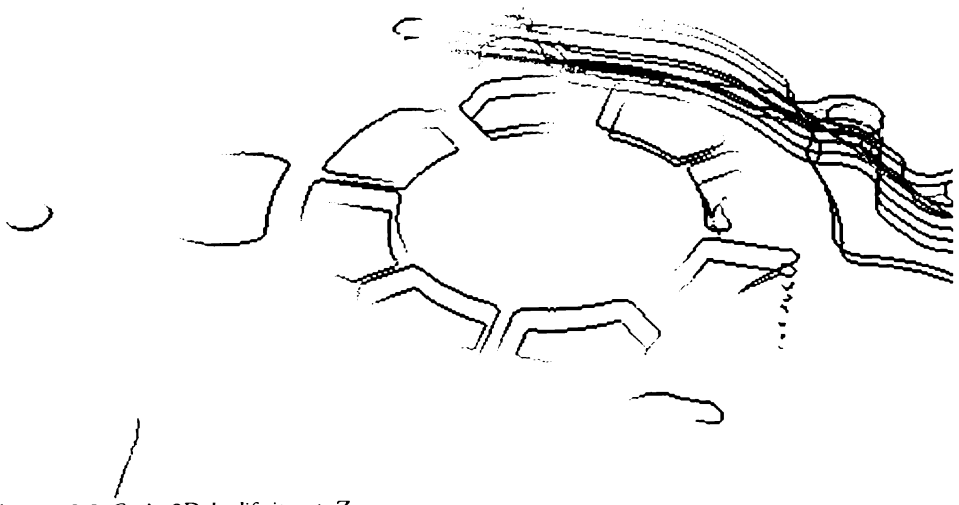


Figura 2.3 *Curbe 2D la diferite cote Z*

2.2.6. Polilinii în spaţiu

❖ ***Definiţie:***

⇨ *poliliniile spaţiale sunt acelea care pot descrie orice curbă în spaţiu, sub o toleranţă. Sunt stocate ca o colecţie de puncte în 3D.*

❖ ***Utilizare:***

⇨ *este tipul principal de dată geometrică existent în fişierele NC destinate frezării în 3 şi mai multe axe. Este utilizat de toate sistemele de fabricaţie care suportă mai mult de 2½ axe. Pe parcursul acestei teze se vor folosi doar curbele de tip* ***polilinie.***

❖ ***Avantaje:***

⇨ *este suportat de toate formatele de import - export;*

⇨ *simplu de procesat şi stocat;*

⇨ *simplu de reprezentat;*

⇨ *orice curbă în spaţiu poate fi convertită într-o polilinie, sub o toleranţă.*

❖ ***Dezavantaje:***

⇨ *sunt dependente de toleranţă;*

⇨ *necesită memorie însemnată pentru stocare.*

❖ ***Exemple:***

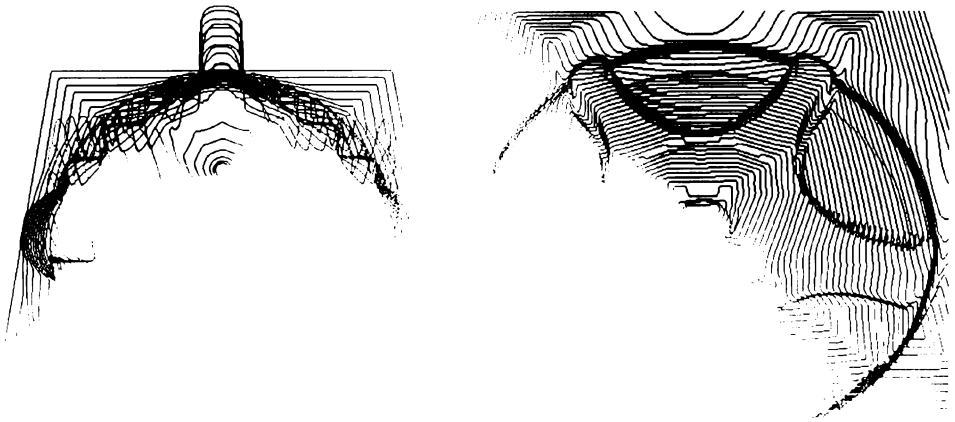


Figura 2.4 *Polilinii în 3D*

2.2.7. Solide simple

❖ ***Definiţie:***

⇨ *solidele simple sunt cele de tip sfere, tori, conuri, piramide, curbe extrudate.*

❖ ***Utilizare:***

⇨ *sunt primele încercări cu adevărat valoroase de a găsi o metodă unificată de a stoca şi analiza solide. Sunt utilizate într-o multitudine de sisteme de proiectare, fiind destul de uşor de utilizat şi implementat.*

❖ ***Avantaje:***

- ⇨ *simplu de editat;*
- ⇨ *interferențe corect detectate;*
- ⇨ *frezări în mai mult de 3 axe;*
- ⇨ *calcul precis al suprafeței și ariei.*

❖ ***Dezavantaje:***

- ⇨ *formate speciale de export și import;*
- ⇨ *pot fi descrie numai geometrii simple.*

❖ ***Exemple:***

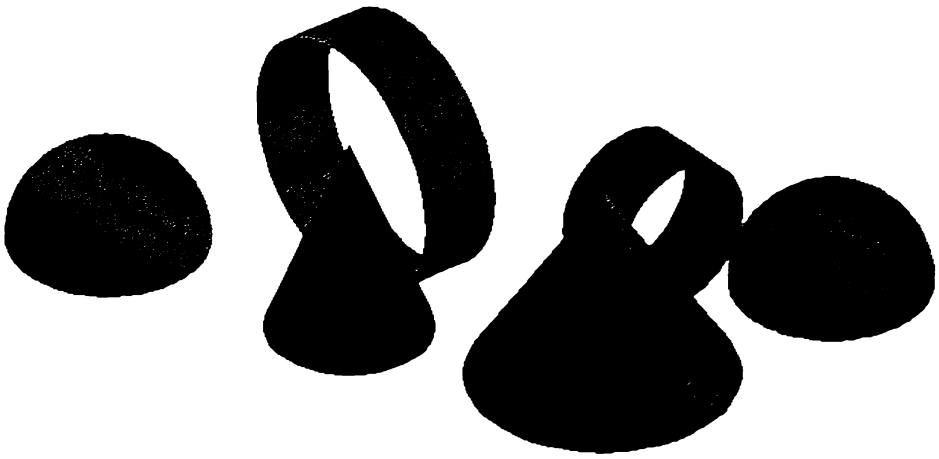


Figura 2.5 *Corpuri simple*

2.2.8. Suprafețe discrete fațetate

❖ ***Definiție:***

- ⇨ *sunt o metodă generică de stocare a unității suprafețe aproximate cu ajutorul unei forme de triunghiuri sau patrulatere. [DM12 - 14 - TPL]*

Se poate demonstra că orice suprafață suficient de netedă poate fi aproximată prin una fațetată finită, sub o toleranță.

❖ ***Utilizare:***

- ⇨ *în formate de export ale unor [illegible] ca DXF și STL;*
- ⇨ *în schimbul de informații cu aplicații de control finit cu elemente finite;*
- ⇨ *în simulatoare de detectare a interferențelor sau detectarea a modificărilor efectuate [illegible];*
- ⇨ *în cartografiere.*

❖ ***Avantaje:***

- ⇨ *foarte generale;*
- ⇨ *capabile să stocheze orice reprezentare;*
- ⇨ *simplu de importat şi exportat;*
- ⇨ *simplu de procesat.*

❖ ***Dezavantaje:***

- ⇨ *foarte dificil de modificat;*
- ⇨ *dependente de toleranţă;*
- ⇨ *mari consumatoare de memorie.*

❖ ***Exemple:***

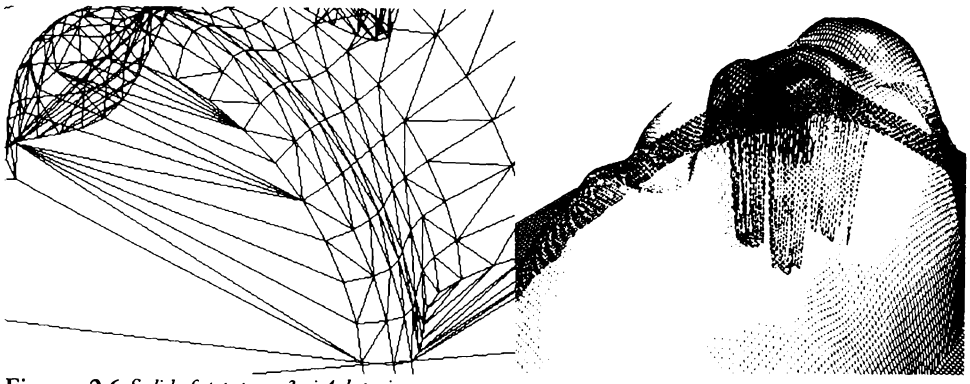


Figura 2.6 *Solide faţetate cu 3 şi 4 laturi*

2.2.9. Suprafeţe discrete demulabile

❖ ***Definiţie:***

- ⇨ *suprafaţa este stocată ca cota Z într-o matrice cu pas variabil sau constant. [DM01 - 11]*

❖ ***Utilizare:***

- ⇨ *este utilizat în produsele din anii '90 care execută diferite analize pe solide, detectează foarte rapid interferenţe, solidifică traiectorii venite de la maşini de palpat sau scanere 3D etc.*

❖ ***Avantaje:***

- *foarte compacte şi uşor de prelucrat;*
- *oferă facilităţi foarte bune de analiză;*
- *uşor de importat şi exportat;*
- *generică (stochează şi analizează orice geometrie cu aceeaşi uşurinţă).*

❖ ***Dezavantaje:***

- *nu conţine informaţii despre muchii şi pereţi verticali.*

❖ ***Exemple:***

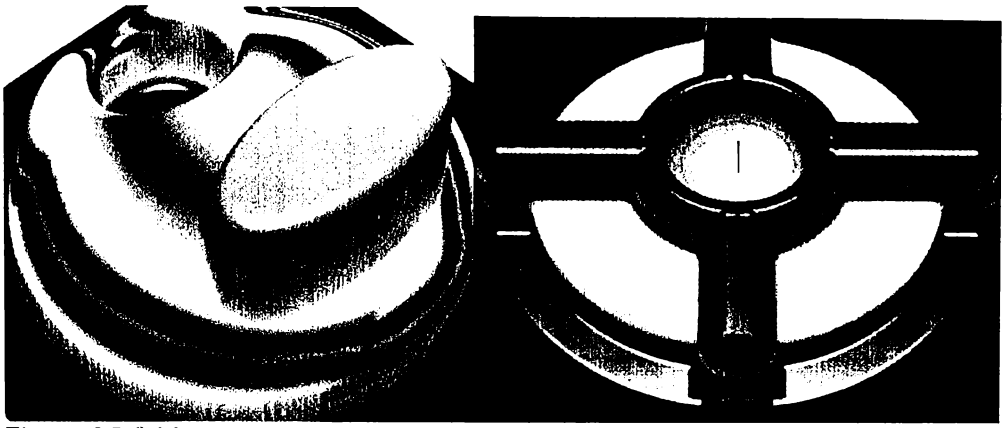


Figura 2.7 *Solide stocate matricial*

Această teză este un caz particular al acestui tip de structură de dată. Metoda va fi limitată doar la repere discrete stocate cu pas constant.

De menționat că pe parcursul lucrării, din rațiunea de a fi succint, acest tip de suprafețe vor fi denumite impropriu "Suprafețe Discrete" (SD), nemaiprecizându-se cuvântul demulabil. "Greșeala" nu este mult prea mare, deoarece multe din tehnicile descrise pot fi, natural, extinse pe alte reprezentări discrete și analitice.

2.2.10. Suprafețe și curbe superioare

❖ ***Definiție:***

⇨ *Sunt acele geometrii care utilizează reprezentări polinomiale pe intervale .*

❖ ***Utilizare:***

⇨ *Sunt folosite crasitotal în toate sistemele de modelare moderne ca și curbe și suprafețe:* spline, B-spline, Bézier, Hermite etc *ECAM, BKOS, DFR, IEMP, LEO, SAU.*

❖ ***Avantaje:***

⇨ *compacte la utilizarea memoriei;*

⇨ *foarte ușor de editat și racordat.*

❖ ***Dezavantaje:***

⇨ *aparatele matemetice de analiză și optimizare sunt foarte rudimentare și greoaie.*

⇨ *dificil de importat și exportat;*

⇨ *dificil de calculat suprafața și volumul.*

❖ ***Exemple:***

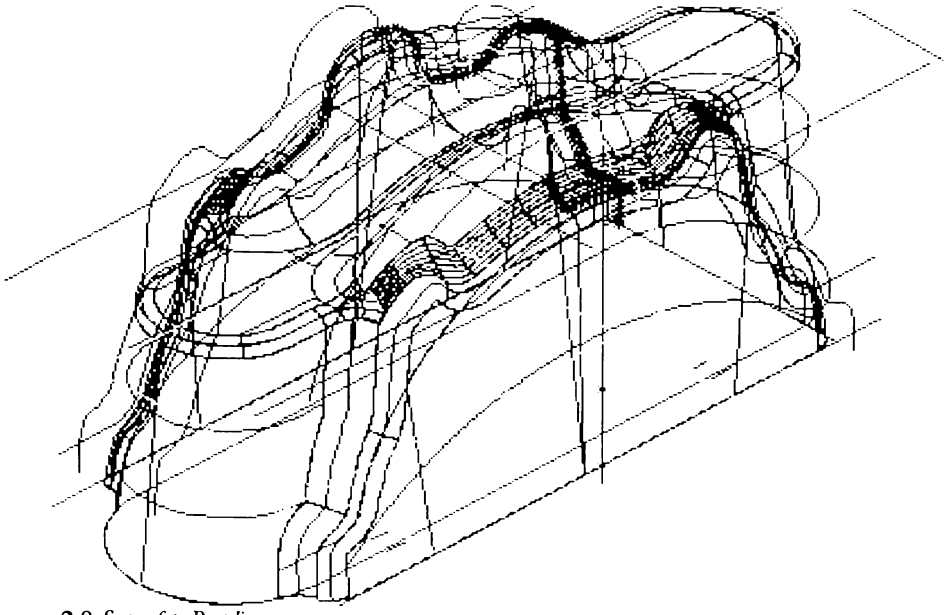


Figura 2.8 *Suprafeţe B-spline*

Constituie începuturile unei adevărate revoluţii în generarea suprafeţelor. Permite ca suprafeţe generate prin diferite metode să fie stocate unitar ca un singur tip de dată, la început ca suprafeţe B-spline după care ca suprafaţă B-spline înconjurată de curbe(trimmed B-spline).

2.2.11. Solide sculpturale

❖ ***Definiţie:***

⇨ *Sunt solide reprezentate analitic pe intervale care trebuiesc să închidă un volum.*

❖ ***Utilizare:***

⇨ *Utilizat în sistemele evoluate de proiectare orientate pe solide* AMD, SolidEdge, Microstation, etc.

❖ ***Avantaje:***

⇨ *facilităţi de modelare extraordinare;*

⇨ *calcul precis al suprafeţei şi volumului;*

⇨ *faciltati bune pentru proiectarea parametrică.*

❖ ***Dezavantaje:***

⇨ *lente pentru analize;*

⇨ *dificil de importat şi exportat;*

⇨ *foarte complexe metode de vehiculare a bazelor de date.*

❖ ***Exemple:***

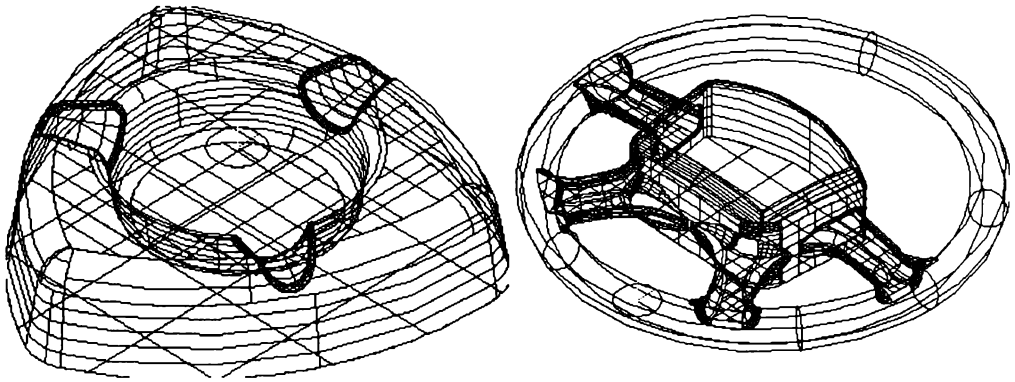


Figura 2.9 *Solide sculpturale*

O metodă cu adevărat revoluţionară de descriere a solidelor de orice fel, cu foarte bune rezultate în importul şi exportul solidelor, modelare, calculul suprafeţelor de offset, reprezentare în mod linii ascunse, faţetat, randat. Este o reprezentare destul de complexă, mare consumatoare de memorie şi foarte lentă, dar analizând evoluţia tehnicii de calcul se pare că se va impune cvasitotal în următorii ani, exceptând probabil unele facilităţi de analiză, care vor lăsa locul aparatelor matematice descrise la punctul precedent [ACIS].

Concluzionând, metodele de stocare a curbelor şi suprafeţelor se împart în două mari categorii: *discrete şi analitice.*

Metodele discrete (*polilinii, suprafeţe discrete faţetate, suprafeţe discrete demulabile, seturi de puncte*)

❖ ***Avantaje:***

- ⇨ *sunt foarte generice, pot stoca orice geometrii cu aceaşi uşurinţă;*
- ⇨ *foarte simplu de analizat;*
- ⇨ *simplu de implementat;*
- ⇨ *simplu de importat şi exportat;*

❖ ***Dezavantaje:***

- ⇨ *sunt dependente de toleranţă;*
- ⇨ *dificil de editat;*
- ⇨ *mari consumatoare de memorie;*
- ⇨ *imprecise în calcule care necisită rezultate foarte precise.*

Metodele analitice (*curbe compuse, suprafeţe analitice, solide sculpturale şi simple*)

❖ ***Avantaje:***

- ⇨ *precise (calcule de volum, perimetru, arie);*
- ⇨ *independente de toleranţă;*
- ⇨ *natural de editat;*
- ⇨ *suport foarte bun pentru proiectare parametrică.*

❖ ***Dezavantaje:***

- ⇨ *dificil de implementat;*
- ⇨ *dificil de importat şi exportat;*
- ⇨ *dificil şi lent de analizat.*

Pe parcursul acestei lucrări vor fi utilizate cvasitotal numai metode de stocare discrete, dorindu-se trasarea de metode generice (care nu sunt particulare unui reper sau familii de repere în parte).

Geometriile analitice vor fi convertite, sub o toleranţă, în geometrii discrete, iar analizele vor fi executate pe reprezentarea discretă, folosind proprietatea acestora de a fi foarte bune reprezentări pentru analize.

2.2. Evoluţia limbajelor de programare

Limbajele de programare au avut o evoluţie foarte spectaculoasă. Scopul primordial a fost acela de a dezvolta cât mai repede aplicaţii fără erori, a organiza şi reutiliza codul, a crea produse program cât mai mentenabile şi portabile.

În ordine cronologică se pot enumera:

- ***cod maşină*** *– programare deosebit de dificilă şi ineficientă;*
- ***asamblare*** *– o evoluţie naturală a codului maşină, prin renumirea într-un format mai accesibil a acestuia. Marele dezavantaj: codul este total neportabil de pe un sistem pe altul.*
- ***C*** *– s-a născut pentru a face mai simplă portarea sistemelor de operare şi a aplicaţiilor de pe un sistem pe altul; limbaj de nivel jos, destinat în general scrierii sistemelor de operare portabile, necesitând crearea pentru fiecare platformă hard a unui compilator simplu de C, după care, pentru implementarea sistemului de operare şi a altor aplicaţii se compilează acesta cu compilatorul respectiv. Născut ca un limbaj primar cu o sintaxă deosebit de simplă, fără verificări de tip, l-a făcut total neinteresant pentru dezvoltatorii de proiectare şi fabricaţie în anii 70;*
- ***FORTRAN (FORmula TRANslation)*** *– primul adevărat limbaj de programare de nivel înalt. Acesta permitea pentru prima dată* ***programarea modulară****, organizarea codului în subrutine şi portarea (interpretarea sau compilarea acestuia) pe diferite sisteme. A fost unul din cele mai folosite limbaje, cu preponderenţă în domeniul ştiinţific. Multe sisteme mari destinate proiectării şi fabricaţiei (ex. Euclid) îşi au originile în anii '60-'70 şi în limbajul FORTRAN;*
- ***BASIC*** *– un limbaj deosebit de simplu, apărut la început fără nici o formă de modularizare a informaţiei, cu excepţia subrutinelor care, însă, nu avea variabile locale. Folosit în general ca limbaj interpretat cablat în ROM, în calculatoarele personale apărute în perioada 1975 – 1985. Acestea erau deosebit de simple (lungime între 1.. 128 k). Nu se poate specifica nici un sistem de proiectare şi fabricaţie comercial care sa fie scris nativ în BASIC. Din cauza "războiului" dintre firmele* ***Microsoft*** *şi* ***Imprise,*** *Basic, prin noua lui încarnare, Visual Basic, a ajuns un limbaj larg întâlnit ca limbaj de comandă în sisteme de proiectare şi fabricaţie, deopotrivă datorită simplităţii, dar şi API-ului "Visual Basic for Applications" furnizat de* ***Microsoft,*** *foarte puternic şi bine documentat;*
- ***Pascal, Modula, Oberon*** *– apărute începând din 1971, ca urmare a lucrărilor profesorului Wirth în domeniul* ***programării structurate****, acestea au fost o adevărată revoluţie în domeniul academic, fiind limbaje de nivel înalt cu verificări de tip, permiţând creerea de librării de algoritmi, refolosirea codului prin creerea modulelor (Modula2 şi Oberon), a listelor de import şi export; au dominat şi încă mai domină mediul universitar, fiind limbajele preferate pentru instruirea studenţilor în algoritmi şi tehnici de programare;*
- ***ADA –*** *unul dintre cele mai complexe şi solide limbaje apărute, conţine verificare puternică de tip metode de detecţie şi tratare de erori, utilizând compilatoare foarte complexe. Este folosit în general în aplicaţiile critice din punct de vedere al erorilor, în domeniul aviatic şi militar, în mainframeuri.*

⇨ ***Objectual Ada, Smaltalk, C++, Eifel, Delphi, Borland Pascal, Object Pascal, Objective C, Java*** – *apariţia programării orientată pe obiecte este una dintre cele mai mari revoluţii în domeniul programării, permiţând, pe lângă avantajele programării modulare şi structurate, noi concepte de modularizare, ca: polimorfism, moştenire simplă şi multiplă, încapsulare, serializare, metode virtuale etc. Se pot realiza librării cu obiecte MFC (Microsoft Foundation Clasa), Java Beans, Turbo Vision. Cu ajutorul acestor librării, aplicaţiile foarte mari pot fi generate foarte uşor cu ajutorul mediilor vizuale şi a "vrăjitoriilor", care fac ca probleme ce altădată necesitau luni de programare să fie rezolvate în timpi de domeniul minutelor, în zona designului interfeţei periferiei şi comunicaţiei. Cea mai importantă evoluţie la ora actuală o are limbajul Java, care este un dialect simplificat de C++ pur obiectual, ce are marele avantaj de a fi portabil pe orice echipament, fără recompilare, de la telefoane şi televizoare inteligente până la cele mai sofisticate calculatoare.*

❖ ***Utilizare:***

La ora actuală **C++** este destinat produselor profesionale de înaltă performanţă uniplatformă; pentru portare pe o altă platformă acestea necesită recompilare.

Java este destinat produselor soft multiplatformă şi este cvasitotal prezent pe internet, dar codul generat este de cca 10 ori mai lent decât cel generat de **C++** sau **Delphi**.

ADA şi **FORTRAN** sunt utilizate în mainframeuri şi aplicaţii din domeniul militar, datorită numărului imens de librării scrise pentru ele, stabilităţii compilatoarelor şi verificarea tipului strict de dată.

Delphi (Object Pascal) este destinat instruirii în tehnici de programare şi tehnici de implementare rapidă a aplicaţiilor (Rapid Application Development RAD).

Basic este un limbaj de batch file sau macro pentru diferite produse, destinat proiectării asistate sau suitelor "office", dar şi pentru dezvoltarea unor produse prin tehnica RAD.

NOTĂ: Nu au fost incluse în această listă o mulţime de alte **limbaje de nivel înalt**: *PL1, Simula, B;* **limbaje de script** sau batch file: *Perl, CGI, diferite alte dialecte de Basic;* **limbaje destinate bazelor de date**: *Cobol, DBase, Paradox, Access, Aproach,* încercând să se cuprindă doar acele limbaje care au avut sau au un real impact în proiectare şi fabricaţie.

Limbajul ales pentru această teză în descrierea unificată a aparatului matematic, algoritmului şi a (eventualei) implementări este bazat pe ***Java*** şi ***C++***, cu traducerea cuvintelor cheie din engleză în română. Se va crea astfel un limbaj orientat obiect simplu, care se doreşte a acoperi cu succes scopul propus.

Dorinţa autorului a fost aceea de a da un caracter practic şi generic expunerii, fără repetarea în trei locuri (*aparat matematic, algoritm în pseudocod şi implementare într-un limbaj*) a aceleaşi metode.

2.3. Evoluţia sistemelor de proiectare şi fabricaţie pe plan mondial

Pe plan mondial, oferta de produse de proiectare şi fabricaţie este într-adevăr impresionantă. Este listată în **Anexa D** numai o enumerare care încearcă să sorteze produsele şi serviciile destinate pieţei de proiectare şi fabricaţie. Vor fi prezentate doar cu numele, deoarece ar fi necinstit să se prezinte doar o parte din produse. În general, fiecare produs sau serviciu este destinat unei anumite pieţe. Cine are prilejul să citească această lucrare în format electronic poate observa că aproape toate titlurile sunt legături la paginile de web ale firmelor producătoare, deci acest subcapitol, ce se referă la stadiul actual, poate fi actualizat permanent, dorindu-se a fi cu adevărat un stadiu actual. Se poate observa că, în general, firmele care dezvoltă programe destinate proiectării şi fabricaţiei mari au apărut în anii '70-'80 şi deţin segmente însemnate din piaţă (peste 95%).

Din punctul de vedere al proiectării, în general, se poate observa că **AutoCAD**-ul devine un standard. Exemplu elocvent al factorul comercial în detrimentul calităţii, o multitudine de alte programe şi-au arătat calităţile remarcabile în domeniul proiectării parametrice (***Ashlar Vellum, TurboCAD, Pro Engineering, Microstation, SolidEdge, SolidWork***) dar, din păcate, în piaţa de consum relativ ridicat al produselor de proiectare se pare că factorul comercial tinde să domine.

În domeniul fabricaţiei, casele mici de software, care ţin relaţia client - producător foarte strânsă, par a domina piaţa. Fiecare firmă producătoare tinde să satisfacă în primul rând doleanţele clienţilor actuali, în baza unor contracte de *colaborare, training şi service*. Odată cu creşterea în facilităţi a acestor produse, care încep să satisfacă din ce în mai mult doleanţe ale consumatorilor, se poate observa tendinţa (naturală) spre comercial şi în piaţa programelor de fabricaţie.

Probabil că din cauza acestei tendinţe (sistemul de proiectare a ajuns un produs de larg consum şi sistemul de fabricaţie este un produs de consum relativ scăzut foarte specializat), la ora actuală companiile care oferă soluţii integrate au un deficit în vânzări, prelevând companiile specializate pe domenii, integrând linii de *proiectare, fabricaţie şi analiză* de la mai mulţi furnizori.

Podusele din domeniul analizei şi optimizării fabricaţiei reperelor complexe sunt foarte puţine şi realizate cu amatorism, din punct de vedere al facilităţilor oferite, dovedind că domeniul este unul foarte nou. *Acesta* este abordat de autor pe parcursul prezentei teze.

Tendinţa este aceea de integrare a proiectării şi fabricaţiei, cu consecinţe favorabile în desfiinţarea graniţei dintre compartimentele de proiectare constructivă (design) şi cel de proiectare tehnologică (fabricaţie); în sensul conceperii unor programe inteligente, care înglobează şi simulează comportamentul uman în situaţii concrete date (sisteme expert); astfel, acestea cuprind decizii în adoptarea unui anumit mod de a realiza un reper, au incluse tehnici de optimizare, şi numai în situaţii extreme necesită intervenţia omului, doar sub forma conversaţională, programele tinzând să propună, prin intermediul "vrăjitorilor", căi de rezolvare, soluţii optimale, în urma analizelor din ce în ce mai sofisticate.

2.4. Evoluţia sistemelor de proiectare şi fabricaţie pe plan naţional

În România, dezorganizarea creată de sistemul comunist (resimţită în toate ţările din estul Europei) a făcut ca domeniul proiectării şi fabricaţiei să fie tratat cu amatorism, nici până la ora actuală nefiind create produse viabile pe piaţa de larg consum. Evoluţia defectuoasă îşi are originile în metodele prin care au fost tratate aceste probleme, în colective reduse, având adesea un caracter academic, şi care erau rareori comunicate publicului larg ori pieţei interesate, având astfel un impact nesemnificativ în industrie. Astfel, în acea perioadă, idei de foarte bună calitate nu au fost cunoscute sau cel puţin comunicate.

Evident, se pot cita câteva materializări ale ingineriei româneşti în acest domeniu (***APT conversaţional, ALMI [ALM], SORI [SOR], MANA [MAN]),*** ca şi o caracteristică comună a acestora fiind definirea separată, negrafică a entităţilor geometrice, şi prezenţa unor ordine de mişcare distincte. Din punct de vedere al performanţelor, aceste programe sunt comparabile între ele, dar cercetarea şi dezvoltarea în continuare a problemei este aproape închisă. Alte produse mai complexe, care posedă o interfaţă grafică, sunt ***BIBEXE [BIB], TechnoPack Lite & Pro. [DM01..11]***

Se pare că singurul produs software care a obţinut recunoaşterea este ***TechnoPack [DM01..11],*** acesta găsindu-şi loc între produsele destinate proiectării şi fabricaţiei pe plan mondial (poate fi găsit la sistemele destinate fabricaţiei). Este produsul care, în decursul vremii, a fost dezvoltat de autor pe baza cercetărilor şi studiilor în domeniul *proiectării, fabricaţiei, dar mai ales a analizei şi optimizării generării codului NC*, în vederea realizării reperelor complexe pe maşini unelte cu comenzi numerice.

Contribuţii valoroase la promovarea proiectării şi fabricaţiei asistate de calculator au fost aduse de către ***prof. dr. ing. Constantin Stăncescu,*** un inimos şi pasionat susţinător al noului în proiectare **[HCF, DMCS07, DMCS08]**. Revista condusă de Domnia sa a prezentat, începând din 1992, noutăţile notabile pe plan mondial şi naţional. Datorită acesteia, a primului club din România, precum şi a învăţământului axat pe proiectarea asistată de calculator promovat la **Facultatea de IMST** din Bucureşti, s-au format o mulţime de departamente de proiectare asistată în intreprinderile româneşti.

O altă apariţie editorială care a dus la promovarea şi dezvoltarea proiectării asistate este **CAD Report,** condusă de un tânăr colectiv din Tg. Mureş **[CREP]**.

⇨

2.5. Concluzii

Având în vedere cele expuse, se poate observa că, în domeniul stocării, comunicării (importului, exportului, DNC), vizualizării şi simulării, lucrurile sunt aproape închise, existând colective şi produse foarte performante. Singurele locuri cu adevărat deschise îmbunătăţirilor sunt:

- ***generarea optimizată a codului NC:***
  - *analizarea şi calculul zonelor neprelucrate din cauza interferenţelor sculă - semifabricat;*
  - *generarea codului, cu condiţia de frezare la rugozitate constantă* - ceea ce duce la reducerea dramatică a timpilor de prelucrare.
  - *detecţia pentru diferite cicluri clasice (echidistante în x, y, z) a zonelor critice, unde nu se poate asigura rugozitatea impusă;*
  - *analiza şi compensarea uzurilor;*
  - *analiza şi compensarea dilatărilor;*
  - *generarea codului NC în curbe cu cât mai puţine inflexiuni - pentru a preveni oprirea şi schimbarea sensului de rotaţie a motoarelor maşinii unelte, ceea ce duce la şocuri, vibraţii şi previne utilizarea eficientă a traseelor de sculă generate la maşini unelte cu avans rapid;*
  - *reducerea mişcărilor rapide - prin optimizarea traiectoriilor în ciclurile de finisare, ceea ce duce la avansuri constante şi timpi mai scăzuţi;*
  - *luarea automată a deciziei privind care tip de frezare este recomandată pentru un anumit tip de reper (în funcţie de gabarit, înclinaţii, inflexiuni, racordări);*
  - *deciderea, în cazul existenţei mai multor scule aşchietoare, a setului optim de scule necesar în vederea frezării reperului dat, la rugozitatea cerută, într-un timp cât mai redus;*
- ***simulări inteligente: (menţin în tot timpul generării fişierului NC starea de frezare a semifabricatului).***
  - *Aceste simulări trebuie să asigure citirea nivelelor de siguranţă pentru mişcări în avans rapid, cât mai reale, deci cât mai joase, optimizând timpul în mişcări rapide;*
  - *Să se poată cunoaşte în fiecare moment cantitatea de material care este prelevată în direcţie radială şi frontală, oferind oportunitatea de a pilota inteligent avansul, încărcând cât mai uniform scula şi maşina unealtă cu solicitări şi scăzând timpii necesari prelucrării unui reper;*
- ***citirea şi convertirea într-un format utilizabil a datelor provenite de la maşini de palpat, scanat şi cartografiat;***
- ***citirea şi convertirea într-un format utilizabil a datelor provenite din alte sisteme destinate proiectării şi fabricaţiei, în format CL sau NC;***

În general, acestea sunt şi domeniile studiate de autor în ultimii ani, domenii în care ar dori să-şi aducă contribuţia.

3. Metode de notaţie şi clase utilizate

3.1. Introducere

Analizând stadiul actual şi observând evoluţia explozivă a tehnicilor de stocare folosite în decursul vremii, nu s-a încercat crearea unei noi metode de stocare, ci folosirea uneia din alte domenii, arătându-se în decursul lucrării imensul potenţial pe care îl posedă în domeniul fabricaţiei.

Metoda de stocare pentru SD utilizată de către autor se încadrează în *suprafeţe discrete demulabile.* Această metodă discretă se regăseşte într-o multitudine de alte domenii, ca metodă de *stocare, vizualizare şi analiză,* utilizată în general pentru stocarea şi procesarea *imaginilor, hărţilor, propagarea dezastrelor (incendii, taifune) în funcţie de geometria terenului, la simularea solidă a frezării în 2, 2½, 3 axe [LWRK], procesare de semnal şi zgomot bidimensional.*

Din cauza problematicii destul de complexe care se doreşte a fi rezolvată, vor fi prezentate şi alte metode de stocare (clase) folosite în decursul tezei; acestea sunt: *şirul de caractere, fişierul, întregul, booleanul, realul, punctul, curba, curbe superioare, plasa, suprafeţe superioare, masca, suprafaţa discretă, colecţiile.*

Toate clasele vor fi prezentate într-un mod organizat, ataşându-li-se atât operaţiile cât şi metodele şi funcţiile asociate, încercându-se în prima parte expunerea unui limbaj pseudocod rudimentar şi simplu de înţeles, orientat pe obiecte.

Nu se doreşte o prezentare exhaustivă a acestor clase, ci doar crearea unui set decent şi relativ bogat de tehnici de operare pe clasa respectivă, tipurile enumerate fiind larg studiate şi implementate în toate sistemele de proiectare şi fabricaţie moderne.

La sfârşitul capitolului, în subcapitolul **Concepte introduse**, se va încerca sintetizarea în pseudocod a celor discutate pe parcursul capitolului. De asemenea, nu se vor expune toate metodele şi funcţiile triviale, deoarece aceasta ar duce la încărarea nejustificată a tezei.

3.2. Limbajul pseudocod

Pe parcursul tezei se va încerca structurarea conceptelor descrise într-un limbaj de tip pseudocod orientat pe obiecte, asemănător cu C++ [CPP], [BST] sau Java [JAV], sintaxa nefiind asemănătoare în totalitate. Scopul principal al acestei structurări este acela de a fi *succintă, flexibilă, sugestivă,* trebuind să posede abilitatea de a descrie *aparatele matematice, structurile de date, designul şi implementarea algoritmului.* Dorinţa autorului este aceea de a prezenta cele discutate într-un singur context.

Facilităţi avansate ca:

- ⇨ *moştenire multiplă sau privată,*
- ⇨ *clase abstracte şi funcţionale,*
- ⇨ *operatori compuşi (+=, -=, *=, etc),*
- ⇨ *constructori şi destructori,*
- ⇨ *interfeţe multiple, protejate sau private,*
- ⇨ *tratarea excepţiilor,*
- ⇨ *spaţiile numelor,*
- ⇨ *pointeri şi referinţe,*
- ⇨ *variabile volatile, constante şi mutabile,*
- ⇨ *mecanisme asincrone de tratare a validităţii datelor,*
- ⇨ *clase şi funcţii template,*
- ⇨ *clase şi funcţii din librăriile standard (cu excepţia celor trigonometrice),*
- ⇨ *metode virtuale,*

nu vor fi folosite, deoarece sunt mai greu de asimilat, în dorinţa de a nu face dificilă înţelegerea algoritmilor.[RBPEL] Structura de clase va fi prezentată doar pentru a forma o ierarhie logică de clase şi a moşteni proprietăţile comune.

În acest limbaj variabila se va numi ***Obiect***. Tipul (sau structura) variabilei, precum şi algoritmii asociaţi, vor fi încapsulaţi într-o ***Clasă***. Această convenţie este necesară pentru a face deosebirea dintre structurile clasice şi *clase* (sau variabilele clasice şi *obiecte*): primele nu pot *conţine metode (algoritmi încapsulaţi, conţinuţi), nu posedă moştenire, polimorfism, abstractizare, metode avansate de creere şi distrugere etc.*

3.2.1. Clasă

Clasele sunt scrise **îngroşat** şi verde închis şi au asociată **obligatoriu** o prescurtare cât mai sugestivă, formată din una sau mai multe caractere scrise cu literă mică. Prefixarea trebuie specificată numai prima oară când clasa este definită.

Clasele încapsulează (conţin) *atribute (set de date), operatori, metode.* Clasele sunt la ora actuală cele mai înalte forme de stocare a informaţiilor.

Ele se pot moşteni, pot asigura mecanisme evoluate de *polimorfism, prototipizare, abstractizare etc.*

❖ ***Exemple:***

Real prefix: **r**; Intreg prefix: **n**; Curbă prefix: **c**; Punct prefix: **p**;

3.2.2. Obiect

Obiectele sunt instanţele (de tipul) unei clase ce sunt declarate în felul următor: TipClasa prefixclasaNumeObiect; numele obiectului este prefixat **obligatoriu** de prescurtarea clasei respective (notaţia ungară). Sunt scrise normal.

❖ ***Exemple:***

Real rToleranţa = 0.0, rRugozitate;

Întreg nIndex = 0, nPuncteCorectate(0);

Punct pTest(0.0, 0.0, 1.0), pMax = {1.0, 1.0, 10.0}, pMin;

Curbă cPătrat;

În cazul mai multor obiecte de un anumit tip se foloseşte un operator "," ca separator. Obiectele pot fi iniţializate în timpul declarării, folosind operatorul de atribuire "=" sau un constructor specific, utilizând operatorul "()", cazul nPuncteCorectate, pTest.

3.2.3. Operator

Operatorii sunt setul de operaţii posibile definit pe **Clasa** respectivă. Ei sunt utilizaţi în scopul scrierii expresiilor într-un limbaj mai apropiat de cel natural. Câteva exemple de utilizare a operatorilor sunt descrise în continuare:

rA = rB + rC cu Atribuie(rA, Sumă(rB, rC))

rDistXYZ = Radical(rX * rX + rY * rY + rZ * rZ) cu Atribuie(rDistXYZ, Radical(Sumă(Înmulţeşte(rX, rX), Înmulţeşte(rY, rY), Înmulţeşte(rZ, rZ))))

Apariţia operatorilor a fost o evoluţie naturală a limbajelor, făcând scrierea expresiilor mult mai lizibilă şi mai apropiată de cea matematică. Ca exemple de limbaje care nu conţin operatori: *Lisp, Prolog, Forth.*

❖ ***Exemple:***

+, -, *, /, &, |, %, <, >, =, ==, +=, -=, *=, /=, &=, |=, %=, ŞI, SAU, !, (), [];

3.2.4. Lista cu parametri

Lista cu parametri ai unei metode sau funcţii este descrisă utilizând operatorul "()" (paranteza deschisă).

❖ ***Exemple:***

Întreg Max(Întreg n0, n1);

NimicAnalizează(); *lista este vidă, dar marcată, pentru a face diferenţa faţă de un obiect.*

3.2.5. Indexul vectorilor

Iteratorii (indecşii) colecţiilor sunt accesaţi folosind operatorul "[]" (paranteze pătrate).

❖ ***Exemple:***

Nimic fcCurbă[2].Rotunjeşte(Întreg nIndex);

3.2.6. Enumerare

Enumerările sunt utilizate fără operatori speciali, în cazul când este vorba de un element, sau folosind operatorul "{}" (paranteze acolade), dacă sunt mai multe elemente.

❖ ***Exemple:***

```
//enumerare de instrucţiuni
Dacă(condiţie)
{
   rA = rB + rC; //instrucţiunea 1;
   rSumă ++; //echivalent cu rSumă = rSumă + 1; instrucţiunea 2;
} //Dacă
 Altfel
rSumă --; //echivalent cu rSumă = rSumă - 1; instrucţiunea 1;
//enumerare de reali
Punct pMax = {1.0, 12.0, 10.0};
```

3.2.7. Algoritm

Algoritmii (*aparate matematice, proceduri, rutine, funcţii, metode*) se vor regăsi în două locuri, sub formă de **funcţii** (în cazul în care nu sunt conţinuţi într-un corp de clasă) sau **metode** (în cazul în care se găsesc încapsulaţi în interiorul unei clase).

Algoritmii sunt blocuri de cod care pot returna un obiect dintr-o clasă anume, sau un obiect din clasa specială Nimic, dacă nu returnează nimic (numiţi proceduri în unele limbaje).

Algoritmii sunt scrişi subliniat în culoarea roşu închis şi suplimentar trebuiesc însoţiţi de lista de obiecte care trebuiesc transmise, scrisă folosind operatorul "()" (între paranteze rotunde). În cazul când această listă este vidă, trebuie chemată cu paranteză rotundă deschisă, urmată de paranteză închisă, pentru a face distincţia între algoritmi şi obiecte.

Algoritmii pot să aibă unii parametri iniţializaţi în momentul definiţiei, ceea ce înseamnă că următoarele chemări ale aceluiaşi algoritm EsteApropiat() sunt valide, vor fi interpretate şi vor avea ca rezultat, ca în exemplul următor:

```
Boolean EsteApropiat(Real r0 = 0.0, r1= 0.0, rToleranţă = 1e-10)
{
Dacă(r0 – r1 < rToleranţă SAU r1 – r0 < rToleranţă)  echivalent Abs(r1 – r0) < rTol
Întoarce(ADEVĂRAT);
```

```
Altfel
    Întoarce(FALSE);
} EsteApropiat
```

❖ ***Utilizări posibile:***

Chemare	Interpretare	Rezultat
EsteApropiat(10.0, 11.0, 2.0)	EsteApropiat(10.0, 11.0, 2.0)	ADEVĂRAT
EsteApropiat(10.0, 11.0)	EsteApropiat(10.0, 11.0, 1e-10)	FALS
EsteApropiat(3.0)	EsteApropiat(3.0, 0.0, 1e-10)	FALS
EsteApropiat(0.0, _, 0.01)	EsteApropiat(0.0, 0.0, 0.01)	ADEVĂRAT
EsteApropiat(_, 11.0)	EsteApropiat(0.0, 11.0, 1e-10)	FALS
EsteApropiat()	EsteApropiat(0.0, 0.0, 1e-10)	ADEVĂRAT

3.2.7.1. Funcţii

Funcţiile sunt algoritmi liberi, care nu sunt încapsulaţi în interiorul vreunei clase.

❖ ***Exemple:***

```
NimicAnalizează();
Real Max(Real rA, Real rB);
Punct Max(Punct rA, Punct rB);
```

3.2.7.2. Funcţii speciale

a) Dacă şi Dacă-Altfel

Dacă o condiţie este îndeplinită se execută prima enumerare de instrucţiuni, altfel (eventual) a doua enumerare.

Dacă(condiţie){...} [Altfel{...}]

b) PentruFiecare

Pentru fiecare valoare, începând cu condiţia de start, sfârşind cu condiţia de sfârşit satisfăcută şi incrementând cu condiţia de incrementare, execută o enumerare de instrucţiuni.

PentruFiecare(Condiţie de start = 1; condiţia de sfârşit = 1; condiţia de incrementare = 1) {...}.

c) PânăCând

Până când este satisfăcută condiţia execută o enumerare de instrucţiuni.

PânăCând(condiţie){...}

Se poate observa că PânăCând(condiţie) ~ PentruFiecare(_, condiţie, _).

d) Întoarce şi Întoarce(...)

Întoarce opreşte execuţia unui algoritm şi (eventual) întoarce un obiect din clasa returnată de algoritmul respectiv.

Întoarce[(obiect de tipul clasei returnate)];

3.2.7.3. Metode

Metodele sunt algoritmi specifici unui clase şi sintaxa lor este exact ca aceea de la algoritmi, cu excepţia că având acces la datele încapsulate (*private, interioare unei clase*), când o metodă este chemată, aceasta trebuie prefixată de numele obiectului respectiv, urmat de un operator "." (punct).

❖ ***Exemple:***

Nimic dsPiesă.Analizează(); Real cCurbă.PuncLa(Întreg nIndex, Punct pPunct);

3.2.8. Cuvinte cheie

3.2.8.1. Clasa

Clasa (cuvânt cheie) înseamnă începutul definiţiei unei noi clase.

3.2.8.2. DerivatăDin

DerivatăDin (cuvânt cheie) înseamnă moştenirea comportamentului unei clase părinte.

3.2.8.3. Operator

Operator (cuvânt cheie) înseamnă începutul descrierii operatorilor permişi pe clasa respectivă.

3.2.8.4. Enumerare

Enumerare (cuvânt cheie) înseamnă începutul enumerării valorilor posibile pentru un obiect.

3.2.9. Comentariu

Comentariile sunt clarificări ale pseudocodului, sunt scrise *italic* şi sunt prefixate de //.

❖ ***Exemple:***

//acesta este un comentariu

În interiorul comentariilor, în cazul în care se doresc a fi specificate diferite valori pentru un coeficient, acestea vor fi specificate în felul următor:

// 0.001, 0.1 <0.01> valori posibile în intervalul închis 0.001, 0.1 , cu valoarea inițială 0.01

Acestea sunt convenţiile minimale cu care se vor modela unificat diferitele aspecte matematice, de structură, algoritmice sau problemele legate de implementare.

De menţionat că, pe parcursul lucrării, algoritmii au fost expuşi în forma lor primară, fără implementarea unor mecanisme evoluate de *Undo, Redo, contoare de derulare, mecanisme de oprire şi recuperare, tehnici de tratare a erorilor, implementări internaţionale, tehnici de optimizare a algoritmilor, transmiteri prin referinţe sau pointeri, metode avansate de inline sau mecanisme de alocare şi dealocare dinamică de memorie.* Toate acestea numai din considerentul de a face prezentarea cât mai lizibilă, urmărindu-se doar prezentarea derulării pur algoritmice a metodei respective. O implementare cât de cât completă, într-un limbaj de programare, este de circa 3..10 ori mai lungă şi nu face obiectul lucrării.

3.3. Clase utilizate

3.3.1. Clasa Întreg

Întregii reprezintă mulţimea numerelor întregi Z. În interiorul acestei lucrări, întregii sunt utilizaţi pentru indexi, stocarea numerelor de elemente ale diferitelor clase superioare etc.

❖ ***Operatori:***

Op	Expresie	Explicaţii	Op	Expresie	Explicaţii
=	C = A	Egalitatea 5 = 5	==	C == A	(C=A)?1:0;
+	C = A + B	Adunarea 5+3= 8	+=	C += A	C = C + A
-	C = A - B	Scăderea 5-3= 2	-=	C -= A	C = C - A
*	C = A * B	Înmulţirea 5*3=15	*=	C *= A	C = C * A
/	C = A / B	Împărţirea 5/3= 1	/=	C /= A	C = C / A
!	C = !A	NEGAŢIA !5 = 2;!101 =010	!=	C != A	C = !A
&	C = A & B	ŞI logic 5&3= 1; 101&110=001	&=	C &= A	C = C & A
|	C = A | B	SAU logic 5|3= 7; 101&110=111	|=	C |= A	C = C | A
%	C = A % B	SAU exclusiv5%3= 6; 101&110=110	%=	C %= A	C = C % A
ŞI	C = A ŞI B	C = (A&B>0)?1:0	INT	C INT rA	INT 4.5 =4
SAU	C = A SAU B	C = (A|B>0)?1:0	MIN	C MIN A	C = MIN(A,C)
%%	C = A %% B	C = (A%B>0)?1:0	MAX	C MAX A	C = MAX(A,C)
>	C = A > B	C = (A>B)?1:0	>=	C = A>=B	C=(A>=B)?1:0
<	C = A < B	C = (A<B)?1:0	<=	C = A<=B	C=(A<=B)?1:0

❖ ***Metode:***

NimicMin (Întreg nA, nB); *//întoarce minimul dintre valorile nA şi nB*
NimicMax (Întreg nA, nB); *//întoarce maximul dintre valorile nA şi nB*
NimicAbs (Întreg nA); *//întoarce valoarea absolută*

❖ ***Funcţii asociate:***

Întreg Min **(Întreg nA, nB)**; *//întoarce minimul dintre valorile nA şi nB*
Întreg Max **(Întreg nA, nB)**; *//întoarce maximul dintre valorile nA şi nB*
Întreg Abs **(Întreg nA)**; *//întoarce valoarea absolută*

❖ ***Prefixare: n;***

❖ ***Exemple:***

nNumărPuncte – *numărul de puncte;*

3.3.2. Clasa Boolean

Booleenii sunt o clasă particulară derivată din întregi (deci moştenesc toate proprietăţile acestora). Rolul lor este acela de a asigura implementarea logicii matematice.

Singurele valori posibile ale booleenilor sunt ADEVĂRAT (= 1) şi FALS (= 0).

❖ ***Operatori:***

Toţi operatorii posibili pe Clasa Întreg sunt aplicabili specializării (clasei derivate) Boolean; evident, nu toţi îşi au sensul.

❖ ***Metode:***

Toate metodele de la Clasa **Întreg** sunt moştenite.

❖ ***Prefixare: b;***

❖ ***Exemple:***

bStart – *variabila booleană Start;*

3.3.3. Clasa Real

Realii sau numerele reale din **R.** Reprezintă o clasă elementară descrisă pentru a înţelege operaţiile executate cu numere reale. Toate operaţiile, metodele şi funcţiile asociate pot fi extinse foarte uşor la clasele mai evoluate: punct, bază vectorială etc.

❖ ***Operatori:***

Op	Expresie	Explicaţii	Op.	Expresie	Explicaţii
=	C = A	Egalitatea 5.0 = 5.0	==	C == A	(C=A)?1.0:0.0
+	C = A + B	Adunarea 5.0+3.0= 8.0	+=	C += A	C = C + A
-	C = A - B	Scăderea 5.0-3.0= 2.0	-=	C -= A	C = C - A
*	C = A * B	Înmulţirea 5.0*3.0=15.0	*=	C *= A	C = C * A
/	C = A / B	Împărţirea 5.0/3.0= 1.6	/=	C /= A	C = C / A
			Real	C real nA	4.0 = real 4
			min	C min A	C = min(A,C)
			max	C max A	C = max(A,C)
>	C = A > B	C = (A>B)?1.0:0.0	>=	C = A>=B	(A>=B)?1.0:0.0
<	C = A < B	C = (A<B)?1.0:0.0	<=	C = A<=B	(A<=B)?1.0:0.0

❖ ***Metode:***

NimicMin (**Real rA, rB**); *//intoarce minimul dintre valorile rA şi rB*
NimicMax (**Real rA, rB**); *//intoarce maximul dintre valorile rA şi rB*
NimicMorf(**Real rA, rB, rP** = 0.5); *//intoarce rA * rP + rB * (1.0 - rP)*
NimicAbs (**Real rA**); *//intoarce valoarea absolută*

❖ ***Funcţii asociate:***

Real Min (**Real rA, rB**); *intoarce minimul dintre valorile rA şi rB*
Real Max (**Real rA, rB**); *intoarce maximul dintre valorile rA şi rB*
Real Morf(**Real rA, rB, rP** = 0.5); *intoarce rA * rP + rB * (1.0 - rP)*
Real Abs (**Real rA**); *intoarce valoarea absolută*

❖ ***Prefixare: r;***

❖ ***Exemple:***

rMin, rMax – *valorile minim şi maxim;*

3.3.4. Clasa Punct

Punctele sunt entităţi elementare folosite de către toate clasele vectoriale superioare, pentru a stoca valorile diferitelor entităţi care le compun.

De menţionat că, pentru simplicitate, în decursul lucrării se vor folosi doar puncte din $\mathbf{R}^3$. Se poate menţiona aici că, în domeniul ştiinţei sunt folosite puncte în mai multe dimensiuni $\mathbf{R}^n$ sau puncte cu construcţie specială pentru suprafeţe superioare $\mathbf{R}^5$ *u, v, x, y, z,* unde *u, v* sunt coordonatele parametrice ale suprafeţei de pe care derivă.

❖ *Operatori:*

Pe clasa **Punct** se aplică exact aceleaşi tipuri de operaţii ca şi la Clasa **Real,** corespunzător fiecărui membru *rX, rY, rZ.*

Pentru a da consistenţă se implementează operaţiile cu scalari de tip real sau întreg, care pot substitui coordonatele unui punct, spre exemplu:

```
pP0(2.0); //pP0.rX = 2.00; pP0.rY = 2.00; pP0.rZ = 2.00; Iniţializare
pP0 = 0.0; //pP0.rX = 0.00; pP0.rY = 0.00; pP0.rZ = 0.00; Iniţializare
pP0(1.0, 3.9,-1.2); //pP0.rX = 2.00; pP0.rY = 3.90; pP0.rZ = -1.20; Iniţializare
pP0 = {2.0, 1.9,-1.2}; //pP0.rX = 2.00; pP0.rY = 1.90; pP0.rZ = -1.20; Iniţializare
pP1 = 2.0 * pP0; //pP1.rX = 4.00; pP1.rY = 3.80; pP1.rZ = -2.40; Înmulţire scalar
pP1*= pP0; //pP1.rX = 8.00; pP1.rY = 7.22; pP1.rZ = 2.88; Înmulţire point
pP0/= 2.0; //pP0.rX = 1.00; pP0.rY = 0.95; pP0.rZ = -0.60; Împărţire
pP1 = max(pP0, 5.0); //pP1.rX = 8.00; pP1.rY = 7.22; pP1.rZ = 5.00; Similar pP1 MAX 5
pP0 min 0.0; //pP0.rX = 0.00; pP0.rY = 0.00; pP0.rZ = -0.60; Op. MIN cu scalar.
pP0=morph(pP0,pP1,0.5); //pP0.rX = 4.00; pP0.rY = 3.61; pP0.rZ = 2.20; funcţia morph.
```

Se poate observa că setul de operaţii descris este destul de consistent şi permite o operare decentă cu clasele elementare descrise până în prezent: *întreg, boolean, real şi punct.*

❖ *Metode:*

```
NimicMin (Punct pA, pB); //întoarce minimul dintre valorile pA şi pB
NimicMax (Punct pA, pB); //întoarce maximul dintre valorile pA şi pB
NimicMorf(Punct pA, pB, pP = 0.5); //întoarce pA * pP + pB * (1.0 - pP)
NimicAbs (Punct pA); //întoarce valoarea absolută

NimicRotX(Punct pRot, Real rUnghiDeg); // rotaţia în jurul lui X
NimicRotY(Punct pRot, Real rUnghiDeg); // rotaţia în jurul lui Y
NimicRotZ(Punct pRot, Real rUnghiDeg); // rotaţia în jurul lui Z
NimicScalare(Punct pScalare, Real rFactorScalare); // scalarea faţă de un punct
NimicTranslaţie(Punct pTranslaţie); // scalarea faţă de un punct
```

❖ *Funcţii asociate:*

```
Punct Min (Punct pA, pB); // intoarce minimul dintre valorile pA şi pB
Punct Max (Punct pA, pB); // intoarce maximul dintre valorile pA şi pB
Punct Morf(Punct pA, pB, pP = 0.5); // intoarce pA * pP + pB * (1.0 - pP)
Punct Abs (Punct pA); // intoarce valoarea absolută

Punct RotX(Punct p, pRot, Real rUnghiDeg); // rotaţia în jurul lui X
Punct RotY(Punct p, pRot, Real rUnghiDeg); // rotaţia în jurul lui Y
```

Punct RotZ(**Punct p, pRot, Real rUnghiDeg**); *rotaţia in jurul lui Z*
Punct Scalare(**Punct p, pScalare, Real rFactorScalare**); *scalarea faţă de un punct*
Punct Translaţie(**Punct p, pTranslaţie**); *scalarea faţă de un punct*

❖ ***Prefixare: p;***

❖ ***Exemple:***

pMin, pMax – *punctul minim şi maxim;*
p0.rX - *valoarea cotei x din punctul 0;*

3.3.5. Clasa SetDeCaractere

Şirul de caractere este o colecţie de obiecte din clasa **Întreg**. Este utilizat la transmiterea mesajelor. Citirea unui caracter din interiorul unui obiect din clasa SetDeCaractere se face cu operatorul"[]" specific parcurgerii colecţiilor.

❖ ***Prefixare: str;***

❖ ***Exemple:***

strMesaj = "[illegible]";

3.3.6. Clasa BazaVectorială

Ba zaVectorială este o clasă elementară care trasează un comportament comun pentru clasele mai evoluate (*curbe, plase, suprafeţe discrete*).

Această clasă a fost imaginată ca punct de pornire pentru descrierea celorlalte superobiecte bazate pe colecţii de puncte.

Conţine obiectele locale din clasa **Punct (pMin, pMax, pDif),** care reprezintă extensia paralelipipedului spaţial în care poate fi inclusă rescpectiva colecţie de puncte.

❖ ***Prefixare: bv;***

3.3.7. Clasa Curbă

Curbele discrete folosite în acestă lucrare sunt cele mai elementare posibile (polilinii în R^3), având ca reprezentare o colecţie de puncte, cu următoarea convenţie: curba se va genera trasând segmente de dreaptă între fiecare pereche de puncte consecutive.

S-a ales acest mod de reprezentare elementar, deoarece oricare curbă continuă de ordin superior (*analitică, arc de cerc, arc de elipsă etc*), dată într-un spaţiu n dimensional, poate fi convertită într-o astfel de reprezentare, dându-se o toleranţă admisibilă.

Curbele închise se definesc a fi acele curbe a căror prim şi ultim element coincid.

Curbele plane sunt acele curbe care nu conţin informaţii despre cota Z, nu au fost tratate separat, ci sunt utilizate cele 3D, nefolosind cota Z.

Curbele sunt entităţi foarte generale, folosite în:

❖ ***generare*** – *curbe directoare şi generatoare, axe de rotaţie etc;*

❖ ***analiză*** – *marchează şi întorc locuri critice în care anumite condiţii nu se împlinesc;*

❖ ***generare fişier NC*** – *traseul sculei;*

Curbele au un punct de start şi un sens de parcurs. În cazul curbelor închise, sensul de parcurs poate fi trigonometric sau orar.

Apariţia **curbelor** discrete (polilinii) a fost naturală, ele fiind cea mai simplă metodă, utilizată larg în schimbul de informaţii de nivel scăzut despre curbe, între aplicaţii. Sunt suportate de aproape toate formatele (*STEP, IGES, VDA, ACIS, DXF, ACD*) .

Dezavantajul lor major este acela că sunt mari consumatoare de memorie, din cauza stocării discrete a valorilor.

O metodă larg utilizată de scădere a consumului de memorie este aceea de a elimina punctele care sunt coliniare; această metodă reduce consumul de memorie cu 10..95%.

O altă metodă utilizată larg în generarea fişierelor NC, şi care duce implicit la reducerea lungimii acestora, este aceea în care se folosesc curbe cu *X, Y sau Z* constant, convertibile în arce de cerc (aproape toate postprocesoarele suportă acest tip de dată). Scăderea necesarului de memorie este de circa 50%. În ultimii ani au apărut ECN care suportă curbe spline. Prin reconvertirea curbelor elementare în curbe spline se asigură o reducere cu 20..50% a lungimii fişierului generat, precum şi o creştere sensibilă a calităţii suprafeţei generate.

NOTĂ: Toate statisticile au fost executate pe repere de complexitate medie, cu plaje de toleranţă [0.001 – 0.1 mm] şi gabarite specifice domeniului construcţiilor de maşini. Tendinţa generală este aceea că: cu cât un reper este mai simplu (număr scăzut de schimbări de curvatură), cu atât optimizările descrise anterior funcţionează mai bine.

Există metode de a creşte **calitatea** curbei, prin conversia ei într-o curbă **B-Spline** şi eliminarea din nou a punctelor intermediare, cu o toleranţă mai mică.

❖ *Operatori:*

Operatorii specifici clasei **Curbă** sunt cei de la clasa **BazaVectorială;** orice operaţie care conţine ca parametru un scalar sau un punct poate fi definită pe o colecţie de puncte.

Suplimentar se pot imagina operatorii +, +=, care adună la lista de puncte un punct nou sau o listă nouă de puncte. Totuşi, această facilitate nu va fi folosită, datorită necesităţii de a fi consecvent în folosirea operatorilor, lăsându-i pentru transformările datelor în maniera folosită la clasele *real, punct.*

❖ *Metode:*

Metodele descrise sunt cele elementare, de adunare, modificare şi ştergere a punctelor, precum şi cele de schimbare de sens, interogare despre sens, care îşi au rostul doar în cazul curbelor închise.

Punct Operator[] (**Întreg nNr**); *întoarce punctul numărul nNr [1..nMax]*
Punct Operator[] (**Real rT**); *întoarce punctul normând curba rT [0.0..1.0]*

Punct Operator[] (**Întreg nNr**, **Real rT**); *//întoarce punctul normând elementul nNr*
NimicPuncLa(**Întreg nNr**, **Punct p**); *//setează punctul numărul nNr [1..nMax]*
Punct Adună(**Punct p**); *//adună un punct*
Punct Adună(**Întreg nNr**, **Curbă c**); *//adună o curbă la poziţia dată de nNr*
Punct Adună(**Curbă c**); *//adună o curbă la sfârşit*

Punct Del(**Întreg nNr**); *//şterge punctul numărul nNr [1..nMax]*
NimicSchimbăSensul(); *//schimbă sensul de parcurgere al curbei*
NimicFăTrigonometric(); *//face sensul de parcurs al curbei în sens trig.*
NimicFăTrigonometric(); *//face sensul de parcurs al curbei în sens trig.*
Boolean EsteÎnSensTrig(); *//întoarce ADEVĂRAT daca-i în sens trig.*
Boolean EsteÎnSensOrar(); *//întoarce ADEVĂRAT daca-i în sens orar*
Boolean EsteÎnchisă(); *//întoarce ADEVĂRAT dacă curba este închisă*

❖ ***Funcţii asociate:***

Nu există încă.

❖ ***Prefixare:***

c;

❖ ***Exemple:***

cBoundaries.nNr – *numărul de puncte de pe curbe care reprezintă dreptunghiul înconjurător;*
cTemp.rMax.rY – *valoarea Y a punctului maxim de pe curba Temp;*

În general, curbele nu se folosesc singure decât în cazul particular al generării de suprafeţe ca şi curbe directoare sau generatoare. Sunt de preferat colecţiile (familii) de curbe, care sunt mult mai puternice din punct de vedere al reprezentării. Spre exemplu, în cazul în care se doresc locuri critice sau trasee de scule în care trebuie alternate mişcări rapide cu cele în avans de lucru, sau scule de geometrii diferite, curbele singure nu sunt destul de bogate în informaţia conţinută pentru a putea reprezenta pertinent aceste tipuri de date, care conţin discontinuităţi geometrice sau informaţii auxiliare (*tehnologice, geometrie de cap de sculă etc.*).

3.3.8. Clasa CurbăSuperioară

Toate curbele care nu sunt stocate ca o secvenţă de segmente de dreaptă vor fi numite **curbe superioare**. Acestea pot conţine: *arce de cerc, curbe B-Spline, informaţii suplimentare despre sculă, culoare, strat de desenare.* În această lucrare nu vor fi detaliate şi utilizate, pentru a păstra expunerea cât mai simplă. Fiind posibilă derivarea directă din **curbă**, toţi operatorii, metodele şi funcţiile asociate clasei **curbă** se presupun moşteniţi.

3.3.9. Clasa Plasă

Plasele sunt cele mai simple obiecte care pot fi utilizate pentru stocarea suprafeţelor. Sunt extensia naturală înspre stocarea suprafeţelor la curbele discrete (polilinii). În concluzie, pentru a le stoca este nevoie de o matrice bidimensională, în care sunt stocate **punctele** reţelei. Rezultă că forma celulei elementare este patrulateră, dar nu şi plană (conversia într-o reprezentare triunghiulară este trivială, unind două colţuri opuse ale celulei elementare).

In mod similar se poate demonstra că orice suprafaţă continuă de ordin superior poate fi convertită într-o suprafaţă finită de tip plasă, cu o eroare mai mică de **toleranţă** dată.

Apariţia **plaselor** a fost naturală, ele fiind o metodă simplă, utilizată larg în schimbul de informaţii de nivel scăzut despre suprafeţe, între aplicaţii. Este suportată cam de toate formatele care suportă suprafeţe (*STEP, IGES, VDA, ACIS, DXF).*

Dezavantajul lor major este acela că sunt mari consumatoare de memorie, din cauza stocării discrete a valorilor.

Având în vedere acest dezavantaj (al consumului de memorie), de-a lungul vremii sau dezvoltat metode mai evoluate, pentru a reduce consumul de memorie, prin detecţia şi eliminarea porţiunilor plane. O primă observaţie este că această metodă utilizează o structură arborescentă pentru stocare mai eficientă, dar şi mai complicată, din punctul de vedere al parcurgerii şi exemplificării.

Plasele şi plasele optimizate se regăsesc în multe din produsele destinate fabricaţiei actuale, ca un tip de dată ascuns, reprezentând corecţia spaţială de sculă, deoarece, pe acest tip de dată proiecţiile şi secţionăriile se realizează relativ simplu, deci conversia în fişier NC este relativ simplă.

Există metode destul de complicate de a creşte **calitatea** plasei, prin conversia ei într-o plasă **B-Spline** şi stocarea cu un pas mai mare (metoda folosită de autor în produsul **TechnoMesh,** numită **rafinare [DM01..12]**).

❖ *Operatori:*

Operatorii specifici clasei **Plasă** sunt cei de la Clasa **BazaVectorială.** Orice operaţie care conţine ca parametru un scalar sau un punct poate fi definită pe o colecţie de puncte de tip plasă (mesh **[ACAD]**).

❖ *Metode:*

Metodele descrise sunt cele elementare, de adunare, modificare şi ştergere a punctelor, precum şi cele de schimbare de sens interogare despre sens, care îşi au sensul doar în cazul curbelor închise.

Punct Operator[] (**Întreg nU, nV**); *//întoarce pt. nU=[1..nNrU], nV [1..nNrV]*
Punct Operator[] (**Real rU, rV**); *//întoarce nomând plasa rU,rV [0.0..1.0]*
Punct Operator[] (**Întreg nU, nV, Real rU, rV**); *//normează elem. NU, nV*
NimicPuncLa(**Întreg nU, nV, Punct p**); *//setează punctul de la nU, nV*
NimicSchimbăSensul(**Boolean bU, bV**); *//schimbă sensul de parcurgere al plasei*
Curbă laDeLaXY(**Întreg nPlasa, Real rX, rY**); *//proiectează şi returnează o listă cu pt.*

FamiliaDeCurbe laDeLaU(**Real rV**); *//converteşte în f.curbă pt rV*
FamiliaDeCurbe laDeLaV(**Real rU**); *//converteşte în f.curbă pt rU*
FamiliaDeCurbe laDeLaX(**Real rX**); *//converteşte în f.curbă pt rX*
FamiliaDeCurbe laDeLaY(**Real rY**); *//converteşte în f.curbă pt rY*
FamiliaDeCurbe laDeLaZ(**Real rZ**); *//converteşte în f.curbă pt rZ*

Real Aria() *//calculează aria*
Real LungimeaU(**Real rV**); *//lungimea la în dir U la v rV*
Real LungimeaV(**Real rU**); *//lungimea la în dir V la u rU*
Real LungimeaMaxU(); *//lungimea maximă în dir U*
Real LungimeaMaxV(); *//lungimea maximă în dir V*
Curbă CurbăU(**Real rV**); *//converteşte în curbă pt rV*
Curbă CurbăV(**Real rU**); *//converteşte în curbă pt rU*

De menţionat metodele *GetU(), GetV(), GetX(), GetY(), GetZ()*, care interoghează plasa prin proiectare sau tăiere. Acestea sunt folosite în conversia suprafeţelor faţetate corecţie de sculă, în trasee de sculă.

De menţionat că *GetX, GetY, GetZ* sunt metode foarte lente iterative, care solicită intens procesorul. Din cauza acestei probleme, analiza şi prelucrarea lor este foartă lentă.

❖ ***Funcţii asociate:***

Nu există încă.

❖ ***Prefixare:***

m;

❖ ***Exemple:***

mAnalitic.nNrU– *numărul de puncte în direcţia U de pe plasa Analitic;*
mTemp.Put(nU, nV, pPunct)– *pune pe plasa Temp la punctul de coordonate nU, nV valoarea pPunct.*

3.3.10. Clasa SuprafeţeSuperioare

Suprafeţele superiore sunt acele suprafeţe care sunt descrise prin alte metode decât cele de la **plase**. În general, fiecare tip de suprafaţă *(revoluţie, rulată, racordare etc)* este stocată în funcţie de metoda de generare. Această metodă duce la premisele regenerării automate, a proiectării parametrice. Dar, dezavantajul major este acela că fiecare set de metode trebuie implementate separat pe fiecare clasă specifică, lucru care duce la îngreunarea dezvoltării produsului soft.

Cunoscându-se acest dezavantaj al **suprafeţelor superiore,** s-a încercat soluţia de compromis, de a o transforma într-o suprafaţă **B-Spline** şi a implementa setul de metode superioare de *secţionare, proiecţia, offsetul 3D* pe un singur tip de suprafaţă, doar o singură dată pe acest tip de dată. Această metodă este folosită la ora actuală în aproape toate sistemele de proiectare şi fabricaţie. Dezavantajele sunt numai în ceea ce priveşte timpul de procesare, care în cazul utilizării curbelor **B-Spline** de ordine mari creşte exponenţial cu gradul curbei.

Importul şi exportul suprafeţelor **B-Spline** este suportat de formatele de nivel înalt, ca: *STEP, IGES, VDA, ACIS.*

❖ ***Prefixare:***

ss;

❖ ***Exemple:***

ssOffset.nNrU– numărul de puncte în direcţia U de pe plasa Offset;

3.3.11. Clasa CapDeSculă

Această clasă are scopul de a stoca o sculă ISO standard, caracterizată de următorii parametri: diametrul mare al sculei rD, diametrul mic rDb, raza de racordare rR, unghiul la bază (orizontal) rA, *unghiul faţă de verticală rB, lungimea părţii active (auriu) rF, lungimea părţii critice (roşu) rC.*

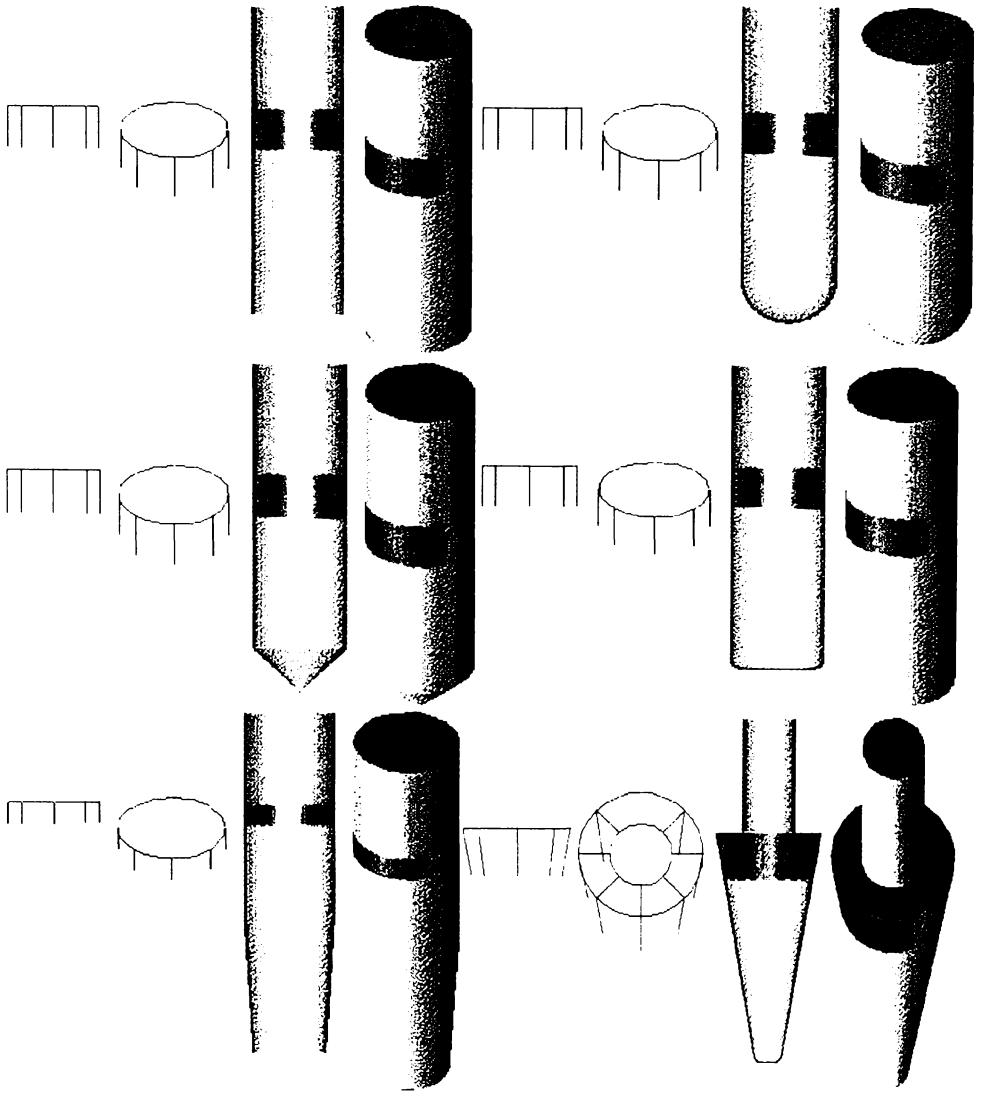


Figura 3.1 *Câteva capete de freză suportate de această lucrare*

Clasa CapDeSculă are rolul de a furniza informaţii despre geometria sculei algoritmilor de calcul a suprafeţei de contact şi a suprafeţei discrete cu materialul nefrezabil, suprafeţe esenţiale în toate procesările de nivel înalt, de optimizare şi procesare a fişierului NC.

Nu posedă *operatori, metode* sau *funcţii asociate.*

Sunt folosite numai cu scopul de a furniza suprafeţei discrete asociate date necesare conversiei, în vederea analizelor de corecţie de sculă.

❖ ***Prefixare:***

sc;

❖ ***Exemple:***

scSferică.rR– *raza capului de sculă Sferic,*

3.3.12. Clasa BazăDiscretă

Clasa **BazăDiscretă** este o clasă de bază derivată din **BazaVectorială**, destinată stocării suprafeţelor mască discretă şi suprafaţă discretă, creând premisele stocării limitelor acestora, precum şi metode de conversie ale originilor, din cele reale ale volumului descris în cele discrete ale matricii de stocare.

❖ ***Variabile:***

Întreg nX; *//numărul de puncte din matrice în direcţia X*
Întreg nY; *//numărul de puncte din matrice în direcţia Y*
Real rStep; *//pasul reprezentării*

❖ ***Metode:***

Real IndexÎnRealPtX(**Întreg nX**); *//conversia din index in real pe X*
Real IndexÎnRealPtY(**Întreg nY**); *//conversia din index in real pe Y*
Întreg RealÎnIndexPtX(**Real nX**); *//conversia din real în index pe X*
Întreg RealÎnIndexPtY(**Real nY**); *//conversia din real în index pe Y*

❖ ***Prefixare:***

Nu există obiecte direct derivate !

3.3.13. Clasa SuprafeţeDiscrete

Suprafeţele discrete sunt metoda de stocare a suprafeţelor discrete.

Forma de stocare este o matrice bidimensională, care stochează o singură cotă z (cea maximă), uniform, într-o reţea parcursă cu **pas** constant, în ambele direcţii, **X** şi **Y,** în plan.

În fapt, metoda se poate asemăna cu realizarea unei fotografii monocrome (neperspectivă) a reperului ce trebuie frezat, în direcţia axei Z a maşinii, stocând în fiecare punct dat, de coordonate x_i, y_i ale fotografiei, cota z maximă vizibilă.

O primă limitare a acestei metode este aceea că nu poate stoca precis suprafeţele neinjective (cele nedemulabile), cele care au puncte în planul XY cu mai mult de o cotă Z sau niciuna. Această problemă a fost rezolvată simulând comportamentul tehnologului, care trebuie să rezolve această problemă dându-i-se spre realizare un astfel de reper.

Astfel:

- ⇨ *în cazul în care nu există valori pentru un punct x, y, se stochează minimul reprezentării sau o altă cotă arbitrară;*
- ⇨ *în cazul în care suprafaţa are o singură cotă Z în spaţiu, aceasta se stochează în matricea cu cotele Z;*
- ⇨ *în cazul în care există mai mult de o cotă pentru aceeaşi valoare x şi y, se stochează valoarea maximă.*

❖ ***Operatori:***

operator +, -, *, /, +=, -=, *=, /=, =, MAX, MIN, ADD, SUB, MULTIPLY, DIVIDE, MORPH, FILSUS, FILJOS, ECDSUS, ECDJOS;

Exemple despre modul în care rezultă suprafaţa, în 12 cazuri de aplicare a operatorilor, în **figura 3.2.**

Mai multe detalii despre aceştia, în capitolul următor.

❖ ***Metode:***

NimicPuneLa(**Întreg nX, Întreg nY, Real rVal, Întreg nMetoda** = PM_NONE, **Real rMorph** = 0.5);
NimicPuneLa(**Real rX, Real rY, Real rVal, Întreg nMetoda** = PM_NONE, **Real rMorph** = 0.5);
Real Operator[] (**Întreg nX, Întreg nY**); *//citeşte un punct*
Real Operator[] (**Real rX, Real rY**); *//citeşte un punct*
NimicPuneLaAll(**Real rValoare**); *//seteaza toată suprafaţa*

Metoda de punere a datelor poate fi: *PM_NONE, PM_MAX, PM_MIN, PM_ADD, PM_SUB, PM_MULTIPLY, PM_DIVIDE, PM_MORPH.*

❖ ***Prefixare:***

sd;

❖ ***Exemple:***

sdReper, sdCorecţie– suprafeţele discrete ale reperului şi ale suprafeţei de corecţie.

3.3.13.1. Operaţii solide

Operaţiile solide între diferitele suprafeţe discrete (combinările) sunt foarte importante pentru diferitele metode de *modelare, import şi analiză.*

Câteva exemple ar putea fi:

- în **modelare,** cu ajutorul operaţiilor *PM_ADD, PM_SUB* se pot aduna sau scădea diferite înscrisuri pe SD, iar cu *PM_FILSUS, PM_FILJOS* se pot executa racordări;
- în **import,** cu operaţiile *PM_MAX, PM_MIN* se asigură continuitatea suprafeţei convertite din diferite petece importate;

- în **analiză,** cu operaţiile *PM_ECDSUS, PM_ECDJOS* se pot calcula suprafeţele traseelor de sculă şi suprafeţele de contact (backoffset), precum utilizând *PM_SUB* se poate calcula SD nefrezabilă în analizele de detecţie a materialului nefrezabil.

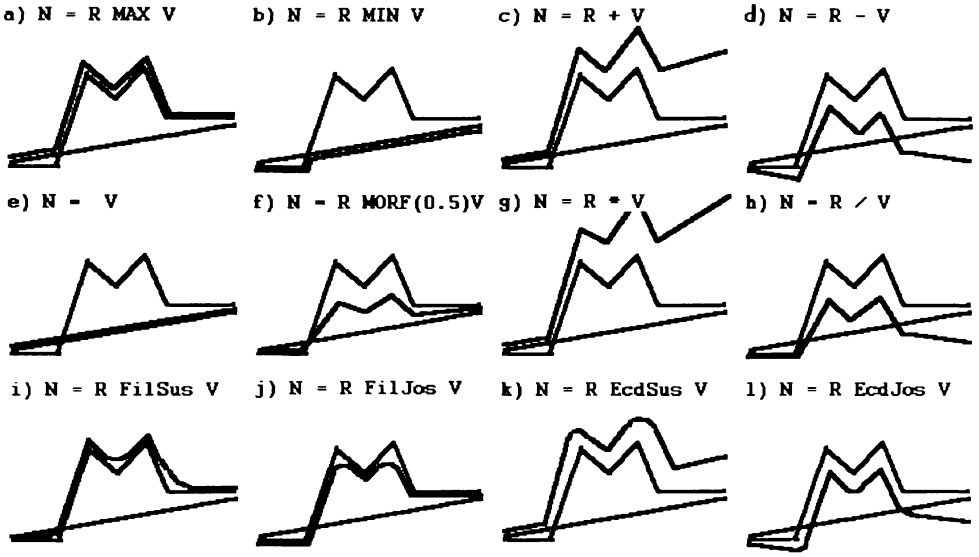


Figura 3.2 *Exemple de operatori solizi posibili între diferite suprafeţe discrete*

Se prezintă în continuare câteva operaţii solide între două SD. Exemple despre cum arată sdRezultante se pot observa în **figura 3.2,** unde s-au reprezentat într-o secţiune cele trei SD: cu negru **sdRezultată**, cu roşu **sdNouă**, cu verde **sdVeche**.

Enumerare ModulDePuncre *//enumerarea modurilor de combinare solidă a SD şi măştilor*
{
PM_NONE, *//sdRezultat sdNouă;*
PM_MAX, *//sdRezultat max(sdVeche, sdNouă);*
PM_MIN, *//sdRezultat min(sdVeche, sdNouă);*
PM_ADD, *//sdRezultat sdVeche + sdNouă;*
PM_SUB, *//sdRezultat sdVeche - sdNouă;*
PM_MULTIPLY, *sdRezultat sdVeche * sdNouă;*
PM_DIVIDE, *//sdRezultat sdVeche sdNouă;*
PM_MORPH, *//sdRezultat morph(sdVeche, sdNouă, rMorfValue);*
PM_FILSUS, *//sdRezultat offset(offset(sdVeche, sdHead, bSus), sdHead, bJos);*
PM_FILJOS, *//sdRezultat offset(offset(sdVeche, sdHead, bJos), sdHead, bSus);*
PM_ECSDUS, *sdRezultat offset(sdVeche, sdNouă);*
PM_ECDJOS, *sdRezultat offset(sdVeche, sdNouă);*
}

3.3.13.2. Necesarul de memorie

În general, încercându-se o analogie cu tehnica stocării imaginilor, SD sunt stocate ca o imagine monocromă, pentru fiecare pixel stocându-se cota Z, iar măştile SDM sunt stocate ca o imagine alb-negru, pentru fiecare pixel fiind stocată o valoare booleană de tip Adevărat – Fals.

Deci, pentru o rezoluţie dată de **m** x **n** avem:

- *Necesarul de memorie = m * n * (4 * 8 + 1) ; //în cazul stocării ca un float*
- *Necesarul de memorie = m * n * (8 * 8 + 1) ; //în cazul stocării ca un double*

Pentru a preîntâmpina consumul neraţional de memorie, datele au fost stocate ca un *float,* adică pe patru octeţi. Această reprezentare asigură peste 6 zecimale exacte pentru valori între [–1000 +1000 mm], adică cele uzuale domeniului de fabricaţie. De menţionat că, atunci când memoria volatilă (RAM) şi spaţiul pe disc nu erau disponibile în cantităţi suficiente, datele erau stocate ca un întreg pe 16 bit, oferind 65536 valori, ceea ce pentru repere cu extensia uzuală pe Z de [–1000...+1000 mm] reprezenta o precizie de 2000/65535 = 0.03 mm, considerându-se suficientă pentru repere gabarite de 2000 mm, şi 0.003 mm pentru cele din plaja de [-100 +100 mm].

Pentru uşurarea calculelor se vor folosi algoritmi bazaţi pe stocare ca *float,* adică un tip de dată real pe 4 octeţi.

3.3.13.3. Mărimea suprafeţelor discrete

În funcţie de rezoluţia de reprezentare şi destinaţia lor, suprafeţele discrete pot fi împărţite în trei mari categorii:

mici, până la 10.000 de puncte (100x100), *rezonabile în cazurile în care se doresc analize şi rapoarte rapide despre diferite moduri de frezare, cazul experţilor auditivi şi ai celor de analiză, care nu execută export grafic, ci doar raportează în mare, spre exemplu, direcţia optimă de frezare, sau dacă este sau nu o frezare echipotenţială recomandată; reprezentare deosebit de rapidă, cu timpi de analiză în domeniul milisecundelor, decizii foarte utile luate în 2..3 secunde. Nu sunt utilizabile în locurile în care se doresc rapoarte grafice;*

timpi medii de răspuns: 0,1-3 [s]

spaţiu necesar pentru stocare 4...40 [ko]

medii, până la 250.000 de puncte (500x500), *utilizate în cazurile în care se doresc rapoarte grafice, ca şi locuri critice din punct de vedere al rugozităţii, pentru diferite cicluri de frezare, detectarea şi exportul zonelor neprelucrate din cauza interferenţelor sculă-semifabricat, simple reprezentări vizuale ale solidului, sau generarea fişierelor de degroşare;*

timpi medii de răspuns: 3-30 [s]

spaţiu necesar pentru stocare 40..1000 [ko]

mari, peste 250.000 (>500x500), *rezonabile numai în cazul generării fişierelor NC de finisare; sunt foarte lente şi mari consumatoare de memorie;*

timpi medii de răspuns: >30 [s]

spaţiu necesar pentru stocare >1000 [ko]

Este foarte important ca, în funcţie de calitatea şi timpul de răspuns dorit, să se aleagă corect mărimea suprafeţei discrete asociate.

De menţionat că, dacă se stochează la o rezoluţie mare, suprafaţa poate fi convertită într-una de rezoluţie mai mică, fără pierdere de calitate, procesul invers nefiind adevărat.

3.3.14. Clasa Mască

Masca reprezintă o matrice bidimensională de aceleaşi dimensiuni cu cele ale SD în care sunt stocate obiecte de tip boolean, şi care este utilizată pentru raportarea diferitelor rezultate ale unor analize.

Suplimentar, în cazul conversiei unei suprafeţe nediscrete (**SN**) într-una discretă (**SD**), s-a creat facilitatea de a stoca extensia în XY a existenţei acesteia, într-o mască numită "masca suprafeţei discrete"(**MSD**).

Măştile sunt concepte mult mai generice, şi sunt folosite pentru definirea diferitelor zone în care se execută operaţiile de combinare a unei suprafeţe cu alta, zone de frezare, sau sunt returnate ca raport al extensiei procesării unei metode de analiză.

❖ *Operatori:*

Cei de la **BazăVectorială,** pentru modificarea *pMin, pMax,* deci scalări şi translaţii.

De asemenea, operatori noi pentru operaţii cu măşti: *AND (Fig.3.2 a), OR (Fig.3.2 b), XOR (Fig.3.2 c), SUB (Fig.3.2 d,e), NOT (Fig.3.2 f);*

❖ *Metode:*

Sunt cele de citire şi scriere.

```
NimicPuncLa(Întreg nX, Întreg nY, Boolean bVal, Întreg nMetoda = PM_NONE);
//pune un bool
Real Operator[] (Întreg nX, Întreg nY); //citeşte un bool
NimicPuncLa(Real rX, Real rY, Boolean bVal, Întreg nMetoda = PM_NONE);  pune un bool
Real Operator[] (Real rX, Real rY); //citeşte un bool
NimicPuncLaAll(Boolean bValoare); //setează toată suprafaţa
NimicRevert(); //schimbă toate valorile
```

Observând metodele de combinare a măştilor s-a creat o metodă inteligentă de a pune datele:

```
NimicPuncLa(Întreg nX, Întreg nY, Boolean bVal, Întreg nMetoda);  pune un bool inteligent
```

Aceasta are ca ultim parametru o valoare *nMetoda,* care poate lua următoarele valori: *PM_NONE, PM_AND, PM_OR, PM_XOR (fig. 3.3).*

❖ ***Prefixare:***

sc;

❖ ***Exemple:***

sdReper.m, sdVârfSculă.m –masca suprafeţei discrete a reperului şi a vârfului sculei;
sdContact.m. Get**(nX, nY),** - masca SD a suprafeţei de contact şi interogarea acesteia despre starea de adevărat sau fals a punctului de coordonate nX, nY;

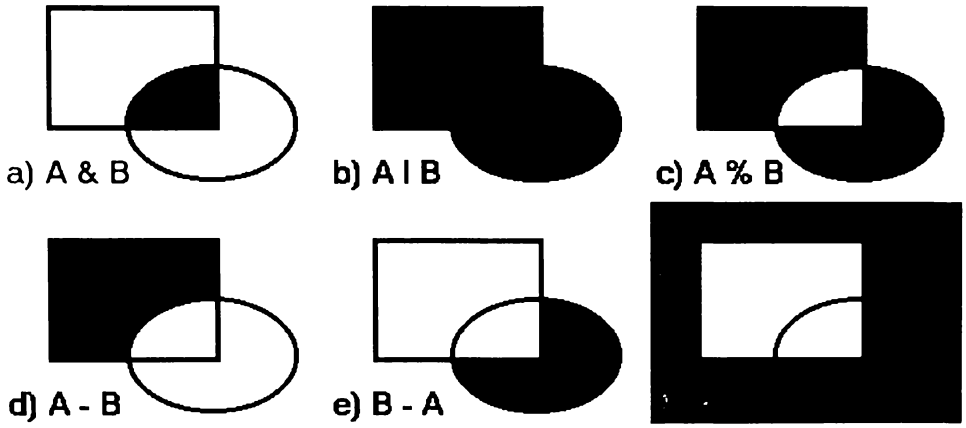


Figura 3.3 *Metode de combinare a măştilor*

3.3.15. Colecţiile

Colecţiile sunt seturi de obiecte derivate din aceeaşi clasă stocate împreună. Accesul la un element dintr-o colecţie se face cu ajutorul operatorul [], care va returna un element din tipul stocat în colecţia respectivă.

Se poate observa imediat că: curba este o colecţie de obiecte de tip **Punct,** suprafaţa demulabilă este o colecţie de **Reali,** şirul de caractere este o colecţie de Întregi etc.

❖ ***Prefixare:***

f + prefixul clasei din colecţie; exceptând cazurile de prefixare discutate

❖ ***Exemple:***

fcAnaliză[3].Desenează() *// desenează curba a treia din setul de curbe fc Analiza.*

3.4. Concepte introduse

```
//Clasa int utilizată pentru operaţiile cu numere intregi
//prefix Variabile: n
Clasa Întreg
{
   operator +, -, *, /, &, |, %, <, >, >=, <=, =, ==, +=, -=, *=, /=, &=, |=, %=, ŞI, SAU, %%, !, min, max; //operatorii şi
   conversiile din tipul real
   Nimic Min (Întreg nA, nB); //întoarce minimul dintre valorile nA şi nB
   Nimic Max (Întreg nA, nB); //întoarce maximul dintre valorile nA şi nB
   Nimic Abs (Întreg nA); //întoarce valoarea absolută
} //EndInt


//funcţii prietene
Întreg Min (Întreg nA, nB); //întoarce minimul dintre valorile nA şi nB
Întreg Max (Întreg nA, nB); //întoarce maximul dintre valorile nA şi nB
Întreg Abs (Întreg nA); //întoarce valoarea absolută


//Clasa bool utilizată pentru operaţiile logice
//prefix Variabile: b
Clasa Boolean DerivatăDin Întreg
{
} //EndBool


//Clasa real utilizată pentru operaţiile cu numere reale
//prefix Variabile: r
Clasa Real
{
//operatorii
   operator +, -, *, /, +=, -=, *=, /=, <, >, <=, >=, =, ==, !=;

//funcţii membru
   Nimic Min (Real rA, rB); //întoarce minimul dintre valorile rA şi rB
   Nimic Max (Real rA, rB); //întoarce maximul dintre valorile rA şi rB
   Nimic Morf(Real rA, rB, rP = 0.5); //întoarce rA * rP + rB * (1.0 - rP)
   Nimic Abs (Real rA); //întoarce valoarea absolută

   Nimic RotX(Punct pRot, Real rUnghiDeg); //rotaţia în jurul lui X
   Nimic RotY(Punct pRot, Real rUnghiDeg); //rotaţia în jurul lui Y
   Nimic RotZ(Punct pRot, Real rUnghiDeg); //rotaţia în jurul lui Z
   Nimic Scalare(Punct pScalare, Real rFactorScalare); //scalarea faţă de un punct
   Nimic Translaţie(Real rTranslaţie); //scalarea faţă de un punct
//suplimentar se pot introduce orice funcţii matematice Ex sin, cos, etc!!
} //real


//funcţii prietene
Real Min (Real rA, rB);   //întoarce minimul dintre valorile rA şi rB
Real Max (Real rA, rB);   //întoarce maximul dintre valorile rA şi rB
Real Morf(Real rA, rB, rP = 0.5);   //întoarce rA * rP + rB * (1.0 - rP)
Real Abs (Real rA);   //întoarce valoarea absolută

Real RotX(Punct pRot, Real rUnghiDeg);   //rotaţia în jurul lui X
Real RotY(Punct pRot, Real rUnghiDeg);   //rotaţia în jurul lui Y
Real RotZ(Punct pRot, Real rUnghiDeg);   //rotaţia în jurul lui Z
Real Scalare(Punct pScalare, Real rFactorScalare);   //scalarea faţă de un punct
```

```
Real Translație(Real rTranslație); //scalarea față de un punct


//Clasa punct utilizată pentru operațiile cu curbe și plase
//prefix Variabile: p
Clasa Punct
{
//variabile locale
   Real rX, rY, rZ; //coordonatele punctului în R^3

//operatorii
   operator +, -, *, /, =, ==, +=, -=, *=, /=, min ,max, morf;

//toate funcțiile următoare suportă orice combinație de puncte și reali !!
   Nimic Min (PunctpA, pB); //întoarce minimul dintre valorile pA și pB
   Nimic Max (PunctpA, pB); //întoarce maximul dintre valorile pA și pB
   Nimic Morf(PunctpA, pB, pP = 0.5); //întoarce pA * pP + pB * (1.0 - pP)
   Nimic Abs (PunctpA); //întoarce valoarea absolută

   Nimic RotX(Punct pRot, Real rUnghiDeg); //rotația în jurul lui X
   Nimic RotY(Punct pRot, Real rUnghiDeg); //rotația în jurul lui Y
   Nimic RotZ(Punct pRot, Real rUnghiDeg); //rotația în jurul lui Z
   Nimic Scalare(Punct pScalare, Real rFactorScalare); //scalarea față de un punct
   Nimic Translație(Punct pTranslație); //scalarea față de un punct
//suplimentar se pot introduce orice funcții matematice Ex sin, cos, etc!!
}//EndPunct

//funcții prietene
Punct Min (Punct pA, pB); //întoarce minimul dintre valorile pA și pB
Punct Max (Punct pA, pB); //întoarce maximul dintre valorile pA și pB
Punct Morf(Punct pA, pB, pP = 0.5); //întoarce pA * pP + pB * (1.0 - pP)
Punct Abs (Punct pA); //întoarce valoarea absolută

Punct RotX(Punct p, pRot, Real rUnghiDeg); //rotația în jurul lui X
Punct RotY(Punct p, pRot, Real rUnghiDeg); //rotația în jurul lui Y
Punct RotZ(Punct p, pRot, Real rUnghiDeg); //rotația în jurul lui Z
Punct Scalare(Punct p, pScalare, Real rFactorScalare); //scalarea față de un punct
Punct Translație(Punct p, pTranslație); //scalarea față de un punct


//Clasa BazăVectorial utilizată la operațiile cu curbe și plase în spațiu
//prefix Variabile: bv
Clasa BazăVectorial
{
   operator +, -, *, /, =, +=, -=, *=, /=, min ,max, morf;

   Punct pMin; //dreptunghiul înconjurător punctul de start
   Punct pMax; //dreptunghiul înconjurător punctul de sfârșit
   Punct pDif; //pMax  pMin
} //EndBazăVectorial


//Clasa curbă utilizată operațiile cu curbe
prefix Variabile: c
Clasa Curbă DerivatăDin BazăVectorial
{
variabile locale
   Întreg nNr; //numărul de puncte (vertex) în R^3

metode
   Punct Operator[] (Întreg nNr);  întoarce punctul numărul nNr  [1..nMax]
   Punct Operator[] (Real rT);  întoarce punctul nomând curba rT  [0.0..1.0]
```

```
    Punct Operator[] (Întreg nNr, Real rT); //întoarce punctul nomând elementul nNr
    Nimic PuneLa(Întreg nNr, Punct p); //setează punctul numărul nNr = [1..nMax]
    Punct Add(Punct p); //adună un punct
    Punct Del(Întreg nNr); //şterge punctul numărul nNr = [1..nMax]
    Punct Add(Întreg nNr, Curbă c); //adună o curbă la poziţia dată de nNr
    Punct Add(Curbă c); //adună o curbă la sfârşit

    Nimic SchimbăSensul(); //schimbă sensul de parcurgere al curbei
    Nimic FăTrigonometric(); //face sensul de parcurs al curbei în sens trig.
    Nimic FăTrigonometric(); //face sensul de parcurs al curbei în sens trig.
    Boolean EsteÎnSensTrig(); //întoarce ADEVĂRAT dacă-i în sens trig.
    Boolean EsteÎnSensOrar(); //întoarce ADEVĂRAT dacă-i în sens orar
    Boolean EsteÎnchisă(); //întoarce ADEVĂRAT dacă curba este închisă
} //Curbă


//Clasa plasă utilizată la operaţiile cu suprafeţe
//prefix Variabile: l
Clasa Plasă DerivatăDin BazăVectorial
    Punct Operator[] (Întreg nU, nV); //întoarce pt. nU=[1..nNrU], nV =[1..nNrV]
    Punct IaDeLaUV(Real rU, rV); //întoarce nomând plasa rU,rV = [0.0..1.0]
    Curbă IaDeLaXY(Real rX, rY); //proiectează şi returnează o listă cu pt.
    Punct Operator[] (Întreg nU, nV, Real rU, rV); //normează elem. NU, nV
    Nimic PuneLa(Întreg nU, nV, Punct p); //setează punctul de la nU, nV
    Nimic SchimbăSensul(Boolean bU, bV); //schimbă sensul de parcurgere al plasei

    FamiliaDeCurbe IaDeLaXY(Întreg nPlasa, Real rX, rY); //proiectează şi returnează o listă
    FamiliaDeCurbe IaDeLaU(Real rV); //converteşte în curbă pt rV
    FamiliaDeCurbe IaDeLaV(Real rU); //converteşte în curbă pt rU
    FamiliaDeCurbe IaDeLaX(Real rX); //converteşte în curbă pt rX
    FamiliaDeCurbe IaDeLaY(Real rY); //converteşte în curbă pt rY
    FamiliaDeCurbe IaDeLaZ(Real rZ); //converteşte în curbă pt rZ

    Real Aria()       //calculează aria
    Real LungimeaU(Real rV); //lungimea la în dir U la v = rV
    Real LungimeaV(Real rU); //lungimea la în dir V la u = rU
    Real LungimeaMaxU(); //lungimea maximă în dir U
    Real LungimeaMaxV(); //lungimea maximă în dir V
} //EndPlasă


//Clasa de bază pentru suprafeţele discrete de tip real şi mască
//prefix Variabile: bd
Clasa BazăDiscret DerivatăDin BazăVectorial
{
    Întreg nX; //numărul de puncte din matrice în direcţia X
    Întreg nY; //numărul de puncte din matrice în direcţia Y
    Real rStep; //pasul reprezentării

    Real IndexÎnRealPtX(Întreg nX); //conversia din index în real pe X
    Real IndexÎnRealPtY(Întreg nY); //conversia din index în real pe Y
    Întreg RealÎnIndexPtX(Real rX); //conversia din real în index pe X
    Întreg RealÎnIndexPtY(Real rY); //conversia din real în index pe Y
} //EndBazăDiscret


  prefix m
Clasa MascăDiscretă DerivatăDin BazăDiscretă
{
    operator &, |, -, &=, |=, -=, ;

  citire scriere date
```

```
    Nimic PuneLa(Întreg nX, Întreg nY, Boolean bVal, Întreg nMetoda = PM_NONE); //pune un bool inteligent
    Boolean Operator[] (Întreg nX, Întreg nY); //citeşte un bool
    Nimic PuneLa(Real rX, Real rY, Boolean bVal, Întreg nMetoda = PM_NONE); //pune un punct
    Boolean Operator[] (Real rX, Real rY); //citeşte un bool
    Nimic PuneLaAll(Boolean bvaloare); //setează toată suprafaţa
    Nimic Revert(); //schimbă toate valorile
} //EndMascăDiscretă


//prefix s
Clasa SuprafaţăDiscretă DerivatăDin BazăDiscretă
{
    MascăDiscretă m; //masca asociată

    operator +, -, *, /, +=, -=, *=, /=, =, MAX, MIN, ADD, SUB, MULTIPLY, DIVIDE, MORPH, FILSUS, FILJOS,
    ECDSUS, ECDJOS;

//citire scriere date
    Nimic PuneLa(Întreg nX, Întreg nY, Real rVal, Întreg nMetoda = PM_NONE, Real rMorph = 0.5);
    Nimic PuneLa(Real rX, Real rY, Real rVal, Întreg nMetoda = PM_NONE, Real rMorph = 0.5);
    Real Operator[] (Întreg nX, Întreg nY); //citeşte un punct
    Real Operator[] (Real rX, Real rY); //citeşte un punct
    Nimic PuneLaAll(Real rValoare); //seteaza toată suprafaţa
}; //EndSuprafaţăDiscretă


Enumerare ModulDePunere //enumerarea modurilor de combinare solidă a SD şi măştilor
{
    PM_NONE, //sdRezultat = sdNouă;
    PM_MAX, //sdRezultat = max(sdVeche, sdNouă);
    PM_MIN, //sdRezultat = min(sdVeche, sdNouă);
    PM_ADD, //sdRezultat = sdVeche + sdNouă;
    PM_SUB, //sdRezultat = sdVeche - sdNouă;
    PM_MULTIPLY, //sdRezultat = sdVeche * sdNouă;
    PM_DIVIDE, //sdRezultat = sdVeche / sdNouă;
    PM_MORPH, //sdRezultat = morph(sdVeche, sdNouă, rMorfValue);
    PM_FILSUS, //sdRezultat = offset(offset(sdVeche, sdHead, bSus), sdHead, bJos);
    PM_FILJOS, //sdRezultat = offset(offset(sdVeche, sdHead, bJos), sdHead, bSus);
    PM_ECSDUS, //sdRezultat = offset(sdVeche, sdNouă);
    PM_ECDJOS, //sdRezultat = offset(sdVeche, sdNouă);
}
```

3.5. Concluzii

În acest capitol s-a dorit prezentarea unitară a modului de organizare şi structurare a datelor, algoritmilor şi aparatelor matematice. Nu s-a folosit nici o metodă clasică descrisă în manualele de prezentare a limbajelor orientate pe obiecte ca Java [**JAV**], Pascal [**PAS**], C++[**CPP**], deoarece folosind o metodă clasică de structurare, prezentarea devine fie prea complexă, fie prea sumară.

Pe parcursul lucrării se va încerca structurarea conceptelor descrise, într-un limbaj de tip pseudocod orientat pe obiecte, asemănător cu C++ sau Java, sintaxa nefiind asemănătoare în totalitate. Scopul principal al acestei structurări este acela de a fi succintă, flexibilă şi sugestivă.

La sfârsitul fiecărui capitol se vor sintetiza în acest limbaj pur matematic şi algoritmic conceptele studiate în capitolul respectiv. Analizând listingul se poate observa că, deşi succint enumerate, clasele elementare *Intreg, Boolean, Real şi Punct* sunt tratate destul de consistent, permiţându-se o multitudine de **operatori** pe tipul respectiv.

Diferenţele faţă de C++ şi Java sunt:

- ⇨ *contoarele sunt bazate pe numerotaţia naturală începând de la 1, nu de la 0;*
- ⇨ *cuvintele cheie au fost traduse din engleză;*
- ⇨ *metodele şi funcţiile sunt subliniate şi scrise cu roşu închis;*
- ⇨ *toate obiectele trebuiesc prefixate cu prefixul tipului de dată pe care-l reprezintă;*
- ⇨ *funcţiile, metodele şi variabilele triviale s-ar putea să nu fie comentate;*
- ⇨ *pentru simplicitate nu s-a implementat tipul pointer sau adresă;*

În concluzie, s-a încercat crearea unei structuri primare de clase care se moştenesc într-o manieră logică şi acoperă în totalitate ***clasele*** necesare acestei lucrări. Structura aceasta de clase poate fi enumerată astfel:

int	i
bool	b
real	r
punct	p
BazăVectorială	bv
Curbă (superioare)	c, cs
Plasa (superioare)	l, ls
BazăDiscretă	bd
SuprafaţaDiscretă	s
Masca	m
Colecţiile	f
SetDeCaractere	str

S-au pus bazele metodei de stocare a **suprafeţelor discrete** (**SD**).

S-a încercat să se cuprindă sumar doar problematica stocării şi a seturilor de opertori admisibili pe fiecare tip de date în parte, conversia fiecărui tip de dată particular nefăcând parte din tema acestui capitol.

De asemenea, s-a introdus conceptul de **mască**, ca loc în care se raportează diferite aspecte legate de modul de procesare al **SD**.

4. Metode de generare şi modelare

4.1. Introducere

În acest capitol se vor prezenta pe larg câteva metode care stau la baza generării SD (o descriere a tuturor metodelor se poate găsi în [**DM01 – 11, DMW01 - 04**]). De asemenea, se vor prezenta diferite metode de conversie din alte tipuri de dată, specifice *importului* din alte sisteme de proiectare şi fabricaţie, date importate în format plasă (suprafeţe discrete patrulatere) sau listă cu triunghiuri (suprafeţe discrete faţetate) [**DM 12 - 15**], forma cea mai simplă de export a solidelor.

De asemenea, se vor prezenta şi defini filtrele, se vor prezenta şi exemplifica scopul şi utilitatea lor în reducerea zgomotului introdus în diferiţi paşi de conversie şi analiză, sau datorat diferitelor inflexiuni particulare ale suprafeţei date spre conversie şi analiză.

Se va descrie un set de algoritmi de modelare: cel al calculului înfăşurătorii şi racordării statice şi dinamice, care nu sunt specifici numai modelării, ci şi calculului suprafeţei corecţie de sculă şi a suprafeţei de contact, pentru scule generice de orice geometrie; se va face particularizarea specifică capetelor de freză, ca fiind suprafeţe de revoluţie.

De asemenea, se vor prezenta, în premieră, trei algoritmi noi concepuţi de către autor:

- ***algoritmul de import şi conversie a seturilor de puncte şi curbe furnizate fără nici o regulă;***
- ***algoritmul de calculare a înfăşurătoarei şi racordărilor cu forme de orice geometrie (un caz particular al acestora sunt capetele de sculă suprafeţe de revoluţie utilizate în frezare);***
- ***reţeaua neuronală pentru antrenarea cu date care nu cad în punctele reţelei, utilizabilă ca o metodă generică de import a tuturor datelor parametrice.***

4.2. Metode de generare discrete

În această parte a capitolului se va discuta despre metodele de generare şi conversiile discrete. Acestea se pot defini foarte simplu, ca fiind metodele care nu depind de coordonatele reale ale solidului, ci numai de cele întregi. Deci, algoritmii sunt de tipul operaţiilor cu matrici, unii dintre ei apropiindu-se foarte mult de algoritmii specifici prelucrării imaginilor.

Utilizarea acestor tehnici în proiectare şi fabricaţie ar părea bizară, dar pe parcursul capitolului se vor aduce lămuririle necesare privind utilitatea fiecărui algoritm.

4.2.1. Plan orizontal

Umplerea unei suprafeţe cu un plan orizontal este utilă în toate cazurile în care se doreşte umplerea întregii SD (sau a unei porţiuni) cu o cotă constantă.

```
SuprafaţăDiscretă.PuneOrizontal( //pune o valoare orizontală pe o SD
    Real rValoare, //valoarea de umplere
    Întreg nModulDePunere = PM_NONE, //tipul de operaţie de combinare cu SD
    Boolean bTotul = ADEVĂRAT //locul de punere peste tot sau numai în mască
    )
{
    PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
        PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
            Dacă(bTotul SAU m[nI, nJ])
                PuneLa(nI, nJ, rValoare, nModulDePunere);
} //PuneOrizontal
```

4.2.2. Plan înclinat

Rolul acestei metode este de a pune un plan înclinat care trece prin trei puncte.

```
SuprafaţăDiscretă.PunePlan3P( //pune un plan înclinat pe o SD
    Punctp1, p2, p3, //cele 3 puncte prin care trece planul
    Întreg nModulDePunere = PM_NONE, //tipul de operaţie de combinare cu SD
    Boolean bTotul = ADEVĂRAT //locul de punere peste tot sau numai în mască
    )
{
    CPlan3Puncte pl3p; //Creează un obiect de tip plan
    pl3p.Iniţializează(p1, p2, p3); //Iniţializează planul cu cele trei puncte
    PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
        PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
            Dacă(bTotul SAU m[nI, nJ])
        PuneLa(nI, nJ, pl3p.laDeLaZ(IndexÎnRealPtX(nI), IndexÎnRealPtY(nJ)), nModulDePunere);
} PunePlan3P
```

Notă: Clasa CPlan3Puncte nu a fost discutată. S-au folosit două metode ale ei, una (Iniţializează) care o iniţializează cu un set de trei puncte p0, p1, p2 şi alta (laDeLaZ) care returnează valoarea Z pentru un X şi Y. Implementarea acestei clase nu face parte din acestă lucrare.

4.2.3. Funcţie Z(x, y)

Rolul acestei metode este de a pune o funcţie injectivă Z(x, y) pe suprafaţa discretă.

```
SuprafaţăDiscretă.PuneFuncţie( //pune o funcţie Z(x, y) pe o SD
    SetDeCaractere strFuncţia, //funcţia care trebuie pusă
    Întreg nModulDePunere = PM_NONE, //tipul de operaţie de combinare cu SD
    Boolean bTotul = ADEVĂRAT //locul de punere peste tot sau numai în mască
    )
{
    CFuncţie fnFuncţia; //Crează un obiect de tip funcţie
    fnFuncţia.Iniţializează(strFuncţia); //Iniţializează funcţia cu expresia dată
    PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
        PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
            Dacă(bTotul SAU m[nI, nJ])
        PuneLa(nI, nJ, fnFuncţia.laDeLaZ(IndexÎnRealPtX(nI), IndexÎnRealPtY(nJ)), nModulDePunere);
} //PuneFuncţie
```

Notă: Clasa CFuncţie nu a fost discutată. S-au folosit două metode ale ei, una (Iniţializează) care o iniţializează cu un SetDeCaractere de tipul "Z(x, y) = SIN(X/100.0) + COS(Y/200.0)" şi o alta (laDeLaZ) care returnează valoarea ei pentru un X şi Y. Implementarea acestei clase nu face partre din acestă lucrare.

4.2.4. Interpolarea prin secţiuni

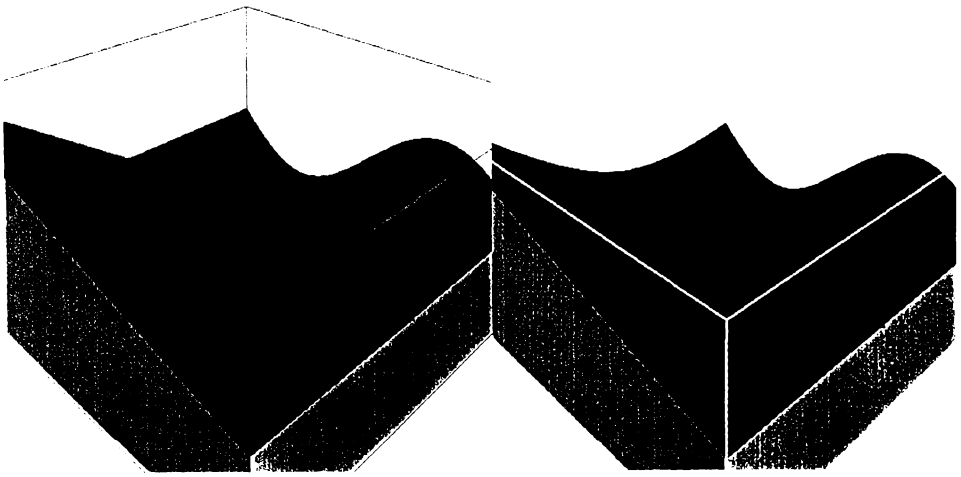
Interpolarea prin secţiuni este o metodă evoluată de generare discretă a SD. Aceasta se defineşte prin trasarea unor secţiuni, prin care se doreşte ca suprafaţa să treacă, marcarea lor în mască (pentru a cunoaşte care zone sunt active) şi interpolarea lor prin linii sau alte curbe mai evoluate.

```
SuprafaţăDiscretă.PuneInterpolareXDir( //pune o altă suprafaţă interpolată prin secţiuni
    SuprafaţăDiscretă sdSecţiuni, //suprafaţa care conţine secţiunile
    Întreg nOrdinCurbei= 1, //Ordinul curbei de interpolare
    Întreg nModulDePunere = PM_NONE, //tipul de operaţie de combinare cu SD
    Boolean bTotul = ADEVĂRAT //locul de punere peste tot sau numai în mască
)
{
    CFuncţie fnFuncţia; //Crează un obiect de tip funcţie
    Punct pCurent;
    Curbă cInterpolator;

    fnFuncţia.Iniţializează(strFuncţia); //Iniţializează funcţia cu expresia dată
    PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
    {
        cInterpolator.Şterge();
        PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
            Dacă(sdSecţiuni.m[nI, nJ])
            {
                pCurent(0, IndexÎnRealPtY(nJ), sdSecţiuni[nI, nJ])
                cInterpolator.Adună(pCurent);
            } Dacă
        cInterpolator.Interpolează(nY, nOrdinCurbei); interpolează

        PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
            Dacă(bTotul SAU m[nI, nJ])
        PuneLa(nI, nJ, cInterpolator[nJ], nModulDePunere);
    } PentruFiecare
```

} *PuneInterpolareYDir*

NOTĂ: Se poate observa că s-au folosit două noi metode ale clasei **Curbă**, una care crează şi interpolează o curbă cunoscând un set de puncte de control şi o alta care şterge toate punctele dintr-o listă cu puncte. De asemennea, funcţia geamănă PuneInterpolareYDir nu a fost prezentată.

Algoritmii din această familie pot fi găsiţi ca produs shareware la adresa ftp.simtel.net\pub\msdos\cad\tmesh0.zip sau ftp.simtel.net\pub\msdos\cad\tpck20lt.zip şi sunt descrişi pe larg în **[DM05..DM11]**.



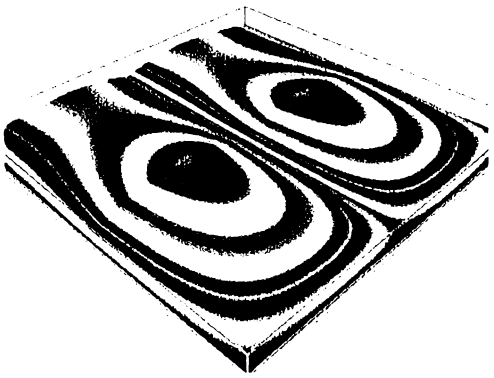


Figura 4.1 *Interpolare prin secţiuni pe SD*

- ⇨ *interpolare liniară*
- ⇨ *interpolare cubică*
- ⇨ *diferenţa dintre cele două*

Exemple procesate utilizând **TechnoCAD** V2.0 si **DSView** V1.0.

În figurile următoare se pot observa diferenţele, pentru acelaşi set de curbe de control (o linie înclinată la –100.0 un **V** la 0.0 şi un sinus la 100.0): două interpolări - una liniară (a), una superioară de ordinul trei (b) şi diferenţa relativă dintre cele două (c) - nuanţele de albastru sunt utilizate pentru a figura valoarea abaterii negative, nuanţele de verde sunt înspre zero roşu înspre plus (palete de analiză standard curcubeu).

4.2.5. Combinarea cu o altă SD

Combinarea cu o altă suprafaţă este o metodă de modelare importantă, fiind utilizată în crearea suprafeţelor superioare complexe care conţin diferite tehnici de modelare şi/sau import.

Având un set bogat de combinare a solidelor (prezentat mai pe larg în figura 3.6), algoritmul devine trivial.

```
SuprafaţăDiscretă.Combină( //pune o funcţie Z(x, y) pe o SD
SuprafaţăDiscretă sdComb, //sd utilizată la combinare
Întreg nModulDePunere = PM_NONE, //tipul de operaţie de combinare cu SD
Boolean bTotul = ADEVĂRAT //locul de punere peste tot sau numai în mască
)
{
   CFuncţie fnFuncţia; //Crează un obiect de tip funcţie
   fnFuncţia.Iniţializează(strFuncţia); //Initializează funcţia cu expresia dată
   PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
      PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
         Dacă(bTotul SAU m[nI, nJ])
            PuneLa(nI, nJ, sdComb[nI, nJ], nModulDePunere);
} //EndCombină
```

Un exemplu de combinare a peste trei sute de plase importate din **EdgeCAM** V3.0 se poate observa în figura 4.2 a, precum şi o analiză a materialului neprelucrat la o frezare cu cap sferic de D = 20.0 mm, în figura 4.2 b.

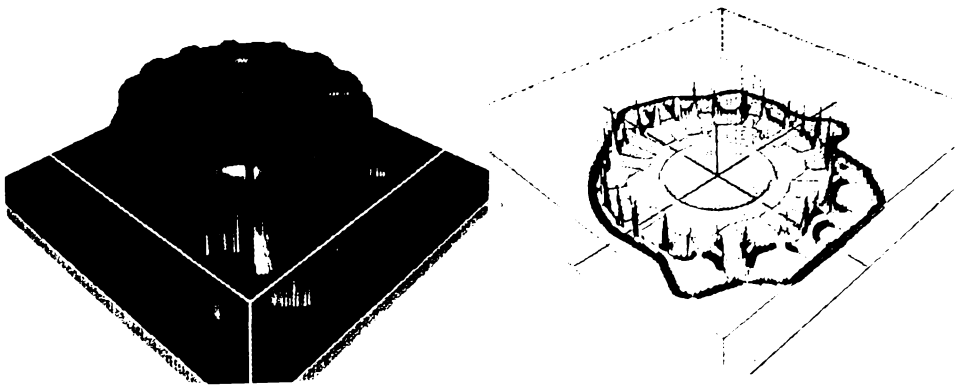
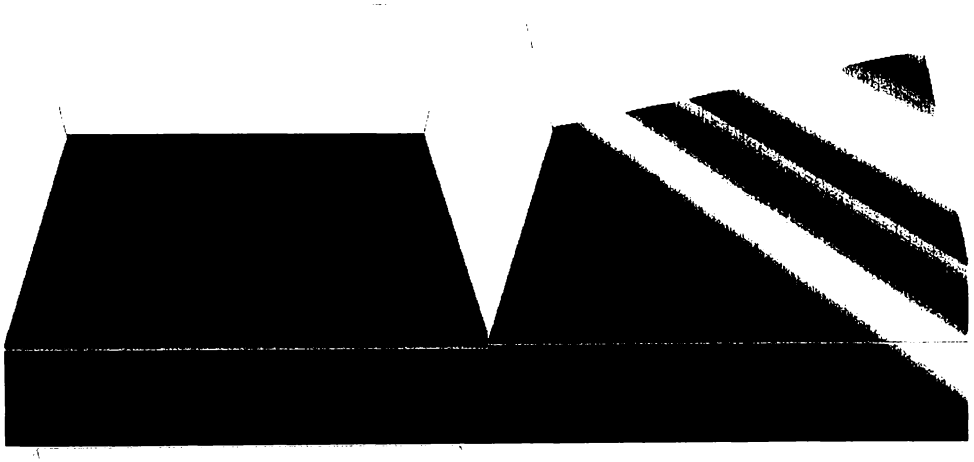


Figura 4.2 *Combinarea solidelor. Exemplu de combinare a 300 SD. Zonele albastre sunt date de zonele neprelucrabile din cauza interferenţei (figura b).*

4.2.6. Exemple despre metodele discutate

Pentru a exemplifica câteva din metodele expuse se va scrie un program minimal şi se vor genera câteva figuri, pentru exemplificarea conceptelor introduse.

```
NimicProgram()
{
   Punct p1(0.0, 0.0, 0.0); //punctul 1;
   Punct p2(1.0, 0.0, 1.0); //punctul 2;
   Punct p3(0.0, 1.0, 1.0); //punctul 3;
   SuprafaţăDiscretă dsTest; //crează o SD
//() iniţializează
   dsTest.Iniţializează(200, 200, (-100.0, -100.0, -100.0), (100.0, 100.0, 100.0), "Test");
   dsTest.PunctLaOrizontal(0.0); //Fig.4.1 s-a utilizat predefinit PM_NONE, ADEVĂRAT
   dsTest.Desenează("Fig. 4.3 a"); //Desenează figura 4.1
   dsTest.PunctLaPlan3p(p0, p1, p2, PM_MAX); //Fig.4.2
   dsTest.Desenează("Fig. 4.3 b"); //Desenează figura 4.2
   dsTest.PunctLaFuncţie("", PM_MAX); //Fig.4.3
   dsTest.Desenează("Fig. 4.3 c"); //Desenează figura 4.3
} //EndProgram
```



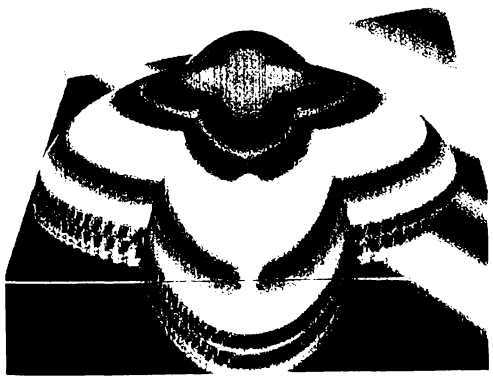


Figura 4.3 *Exemple de modelare elementare*

- ⇨ *umplere cu un plan orizontal (PM_NONE);*
- ⇨ *umplere cu un plan inclinat (PM_MAX);*
- ⇨ *umplere cu două funcţii (PM_MAX);*

4.2.7. Set de puncte

Pentru acest tip de interpolare s-a dezvoltat un algoritm original, care rezolvă interpolarea unui set de date, date în spaţiu după nici o regulă (adică nu sunt de-a lungul nici unei direcţii de generare *X, Y, Z, U, V*) (eng: cloudy set of points). Acest algoritm este prezentat în premieră aici. Utilitatea lui este fără graniţe, fiind deopotrivă utilizabil în proiectarea asistată, la conversia în solizi a datelor provenite de la dispozitivele de palpat sau scanat, precum şi în cartografie, la conversia datelor provenite de la GPS (geographical positioning system) (implementat în **TechnoPoint** V2.0, shareware la adresa ftp.simtel.net\pub\msdos\cad\tpnt0.zip sau ftp.simtel.net\pub\msdos\cad\tpck20lt.zip şi descris pe larg de către autor în **[DM05...DM11]**).

Algoritmul poate fi extins foarte uşor şi la convertirea curbelor, discretizându-le cu un pas constant (folosind metoda Punct **Curbă**.IaDeLaT(Real rT)).

De asemenea, pentru fiecare punct s-a asigurat posibilitatea setării unei ponderi, deci pentru fiecare punct poate fi modificată influenţa acestuia în suprafaţa rezultantă.

De menţionat că algoritmul rezolvă **ORICE** set de puncte şi/sau curbe în spaţiu. În implementarea algoritmului se va presupune că punctele au o valoare a ponderii asociată într-o listă cu reali, care nu a fost discutată în capitolele precedente.

```
NimicAdunăOCurbă( //adună o curbă la lista cu p. pt conv.
    Curbă cListaCuPuncte, //Lista cu puncte pt. conversie
    FamiliaDeReali frListaCuPonderi, //Lista cu ponderi pt. fiecare p.
    Curba CurbaCurentă, //Curba care trebuie convertită
    Real rPonderea, // Ponderea pt. curba curentă
    Real rPas //pasul de parcurgere
    )
{
    Întreg nMax = CurbaCurentă.IaDeLaL() / rPas + 0.5; //nr de puncte
    PentruFiecare(Întreg nI = 0; nI <= nMax, nI++)
    {
        cListaCuPuncte.Adună(CurbaCurentă.GetT(nI/nMax));
        cListaCuPonderi.Adună(rPonderea);
    } //PentruFiecare
} //EndAdunăOCurbă

NimicSuprafaţăDiscretă.InterpolcazăPrinPuncte( interpolează
    Curbă cListaCuPuncte, // Lista cu puncte pt. conversie
    FamiliaDeReali frListaCuPonderi, Lista cu ponderi pt. fiecare p.
    FamiliaDeCurba fcAlteCurbe, Alte Curbe care trebuiesc convertite
    FamiliaDeReali frPonderiPtCurbe, Ponderile pt fiecare curbă
    FamiliaDeReali frPaşiiPtCurbe, Paşii pt fiecare curbă
    Real rRigiditatea Rigiditatea suprafeţei rezultate [0..1]
    )
{
  converteşte rigiditatea într-o formă utilizată de algoritm
    Dacă(rRigiditatea<=0.5)
        rRigiditatea:=2.0 * rRigiditatea;
    Altfel rRigiditatea:=1.0 / (2.0 * (1.0 - rRigiditatea));

  Adună toate curbele potenţiale la lista cu ponderi în funcţie de
  ponderile şi paşii asociaţi lor
    PentruFiecare(Întreg nI = 0; nI <= nMax, nI++)
        AdunăOCurbă(cListaCuPuncte, frListaCuPonderi, fcAlteCurbe[nI],
```

```
        frPonderiPtCurbe[nI], frPaşiiPtCurbe[nI]);

    //interpolează SD
        PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
            PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
            {
                Punct pPunctulCurent(IndexÎnRealPtX(nI), IndexÎnRealPtY(nJ), 0.0);
                Real rDist, rDistSpecial = 0.0, rSumaDistSpecial = 0.0, rZ = 0.0;
                FamiliaDeReali frTemp;
                PentruFiecare(Întreg nPunctul = 1; nPunctul <= cListaCuPuncte.nNr; nPunctul++)
                {
                    rDist = DistXY(pPunctulCurent, cListaCuPuncte[nPunctul]);
                    rDistSpecial:=Pow(1.0 / rDist, rRigiditatea)*
                    frListaCuPonderi[nPunctul];
                    rSumaDistSpecial = rSumaDistSpecial + rDistSpecial;
                    frTemp.Adună(rDistSpecial);
                } //PentruFiecare

            PentruFiecare(Întreg nPunctul = 1; nPunctul <= cListaCuPuncte.nNr; nPunctul++)
            {
                rZ = rZ + cListaCuPuncte[nPunctul) * frTemp[nPunctul);
            } //PentruFiecare

    //Ultima operaţie
            rZ = rZ / rSumaDistSpecial;
        } //PentruFiecare
} //EndInterpoleazăPrinPuncte
```

NOTĂ: Clasa **FamiliaDeReali** nu a fost discutată; utilizarea ei este exact ca si **Curbă,** doar că nu stochează puncte, ci valori reale.

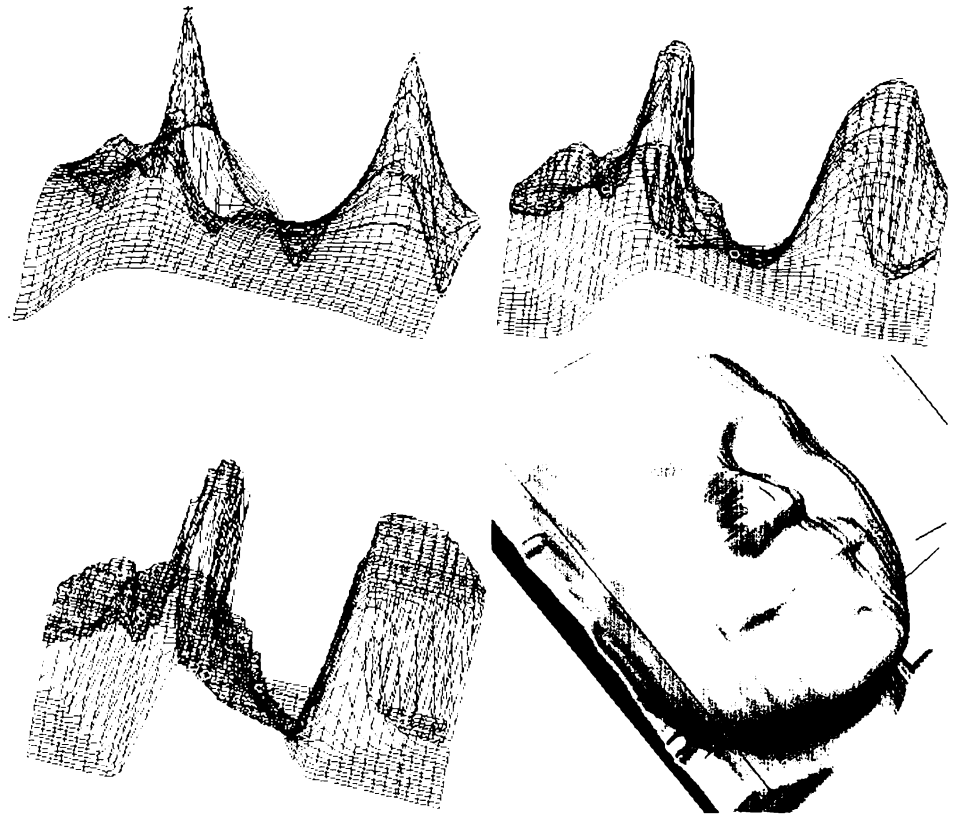


Figura 4.4 *Interpolarea prin puncte:*

- ⇨ *rigiditate 0.0;*
- ⇨ *b) rigiditate 0.5;*
- ⇨ *c) rigiditate 1.0*
- ⇨ *d) set de date palpator, interpolare solidă prin set de puncte* ***TechnoPoint*** *V'2.0*

4.2.8. Interpolarea capetelor de scule

Capetele de sculă au o importanţă deosebită în analizele şi generările de fişiere NC din capitolele ulterioare, precum şi în racordările şi calculul suprafeţei înfăşurătoare necesare realizării suprafeţei poansonului din suprafaţa matriţei sau invers.

Din aceste considerente, s-a încercat să se cuprindă o varietate cât mai mare de capete de sculă, pentru a acoperi toate procesările uzuale.

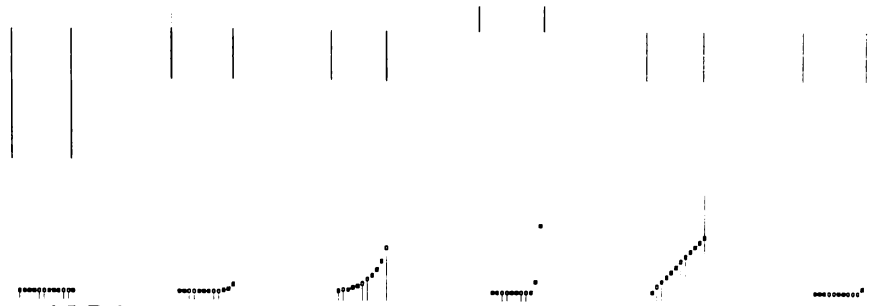


Figura 4.5 *Diferite scule şi valorile lor după conversia în suprafaţă discretă*

În figura 4.5 s-au reprezentat câteva generatoare ale unor scule uzuale (o reprezentare mai completă se poate găsi în figura 3.1).

Prezentarea aceasta ajută la explicarea şi exemplificarea conceptelor necesare unei implementări generice a algoritmului, care permite utilizarea oricărei geometrii injective, care nu trebuie să fie neapărat o geometrie bazată pe o suprafaţă de revoluţie.

Pentru simplificare se v-a utiliza o suprafaţă de revoluţie. Se va presupune că, curba generatoare este o curbă în planul XY, şi se va folosi metoda laDeLaY(Real rX), care întoarce valoarea cea mai mică (spre exemplu), creând seturi de blipsuri îngroşate în figură.

De menţionat că obiectul asociat capului de sculă este o SD, care are acelaşi pas cu SD care trebuie ofsetată sau racordată. Deci, în cazul SD discretizate cu diferiţi paşi, trebuie create diferite discretizări ale capetelor de sculă.

```
NimicSuprafațăDiscretă.InitCapDeSculă( //Crează SD bazându-se pe o geom. de sculă
      Curbă cCapDeSculă, //Curba care descrie geometria capului
      Real rPas  //Pasul de discretizare
      )
{
   Real rRazaCurentă, rRaza = cCapDeSculă.laDeLaX();
   Întreg nNr = rRaza / rPas + 0.5;
   Inițializează(nNr, nNr, (-rRaza, -rRaza, 0.0), (rRaza, rRaza, 0.0), Scul);
   m.PuneLaAll(FALS);  //pune în toată masca fals

   PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
      PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
      {
         rRazaCurentă = Sqrt(IndexÎnRealPtX(nI)* IndexÎnRealPtX(nI) + IndexÎnRealPtY(nJ) *
            IndexÎnRealPtY(nJ));
         Dacă(rRazaCurentă < rRaza)
         {
            PuneLa(nI, nJ, cCapDeSculă.laDeLaY(rRazaCurentă));
            m.PuneLa(ADEVĂRAT);
```

```
        } Dacă
    } PentruFiecare
} EndInitCapDeSculă
```

Această metodă asigură crearea şi generarea sculelor pentru frezare. Similar se pot crea SD care conţin electrozi, la analizarea prelucrării prin electroeroziune.

4.2.9. Calculul SD înfăşurătoare şi SD de racordare statice

Principalii beneficiari ai capetelor de sculă sunt algoritmii de racordare şi de calcul ai înfăşurătoarei.

Dorind a trata generic problema, se poate spune că algoritmul enunţat în continuare calculează SD înfăşurătoare, pentru orice geometrie a SD şi a sculei, deci se poate vorbi de racordări care nu sunt sferice, ci de orice altă formă. Algoritmul are abilitatea de a calcula înfăşurătoarea şi a face calculul total de interferenţă într-un singur pas.

```
NimicSuprafaţăDiscretă.InitÎnfăşurătoareStatic( // înfăşoară în spaţiu o SD sculă
    SuprafaţăDiscretă sdSupafaţa, //suprafaţa care trebuie înfăşurată
    SuprafaţăDiscretă sdSculă, //geometria capului utilizat
    Boolean bDirecţiaSus    = ADEVĂRAT //înfăşurătoarea în sus sau în jos
    Boolean bTotul = ADEVĂRAT //toată suprafaţa sau în mască
    )
{
  Real rtemp, rZ;

  PentruFiecare(Întreg nSDX = 1; nSDX <= nX; nSDX++)
    PentruFiecare(Întreg nSDY = 1; nSDY <= nY; nSDY++)
    {
      Dacă(bDirecţiaSus)
        rTemp = -1e100;
      Altfel
        rTemp = 1e100;
      PentruFiecare(Întreg nSCX = 1; nSCX <= nX; nSCX++)
        PentruFiecare(Întreg nSCY = 1; nSCY <= nY; nSCY++)
          Dacă(bTotul SAU sdSuprafaţa.m[nSCX, nSCY] ŞI
            sdSculă.m[nSCX, nSCY] ŞI nSDX + nSCX < nX ŞI nSDY + nSCY < nY)
            Dacă(bDirecţiaSus)
            {
              rZ=sdSuprafaţă[nSDX+nSCX, nSDY+nSCY]+sdSCulă[nSCX, nSCY];
              Dacă(rZ > rTemp)
                rTemp = rZ;
            } Dacă
            Altfel
            {
              rZ=sdSuprafaţă[nSDX+nSCX, nSDY+nSCY]-sdSCulă[nSCX, nSCY];
              Dacă(rZ < rTemp) rTemp = rZ;
            } Altfel
      PuncLa(nSDX, nSDY, rTemp);
    } PentruFiecare
} EndInitCapDeSculăStatic
```

Se poate observa uşor că algoritmul este de complexitate $O(n^4)$ (patru cicluri Pentru fiecare), deci presupune calcule îndelungate. În prezent, autorul foloseşte o versiune mult îmbunătăţită (de peste 10 ori mai rapidă), dar complexitatea acestuia depăşeşte cadrul lucrării. Timpul necesar algoritmului prezentat este de domeniul zecilor de minute, pentru o reprezentare de 1000 x 1000 mm şi o sculă de 50 x 50 mm, iar cu cel optimizat, de numai 2 minute. Aceste tipuri de optimizări ar conduce la o implementare de peste 1000 de linii de cod, ceea ce ar reduce destul de mult simplitatea algoritmului prezentat.

Calculul suprafeţei de racordare se execută la fel, dar în direcţia opusă, folosind pentru acesta comutatorul *bDirecţiaSus.*

Calculul înfăşurătoarei utilizând o mască se poate face cu acelaşi algoritm, folosind comutatorul *bTotul.*

Calculul suprafeţei de racordare este simplu. De menţionat că suprafaţa de racordare cu o sculă statică (care nu-şi modifică diametrul) este utilizată în fabricaţie la simularea suprafeţei teoretice de contact (backoffset), *care reprezintă suprafaţa teoretică absolută ce poate fi obţinută, folosind o sculă de orice formă şi luând în considerare calculul total de interferenţă.* Făcând o diferenţă discretă între suprafaţa de racordare şi cea normală se va obţine suprafaţa cu materialul nefrezabil.

```
NimicSuprafaţăDiscretă.InitRacordareStatic( // infăşoară în spaţiu o SD sculă
        SuprafaţăDiscretă sdSupafaţa, // suprafaţa care trebuie înfăşurată
        SuprafaţăDiscretă sdScalare, // geometria coeficienţilor de scalare
        SuprafaţăDiscretă sdSculă, // geometria capului utilizat
        Boolean bDirecţiaSus // înfăşurătoarea în sus sau în jos
        Boolean bTotul // toată suprafaţa sau în mască
        )
{
    SuprafaţăDiscretă sdTemp;
    sdTemp.InitÎnfăşurătoareStatic(sdSupafaţa, sdScalare, sdSculă, bDirecţiaSus, bTotul);
    InitÎnfăşurătoareStatic(sdTemp, sdScalare, sdSculă, !bDirecţiaSus, bTotul);
} // EndInitRacordareStatic
```

4.2.10. Calculul SD înfăşurătoare şi SD de racordare dinamice

Algoritmul prezentat în continuare are avantajul de a combina avantajele algoritmului precedent cu facilitatea de a putea modifica raza sculei dinamic, în funcţie de poziţia capului de frezare în spaţiu. Valorile razei sunt citite dinamic de pe o suprafaţă discretă care conţine valoarea razei sau valoarea factorului de scalare care se aplică asupra capului. Acest tip de algoritm este utilizat în cazul în care se doresc grosimi de pereţi variabile sau raze de racordare variabile.

```
NimicSuprafaţăDiscretă.InitÎnfăşurătoareDinamic( // infăşoară în spaţiu o SD sculă
        SuprafaţăDiscretă sdSupafaţa, // suprafaţa care trebuie înfăşurată
        SuprafaţăDiscretă sdScalare, // geometria coeficienţilor de scalare
        SuprafaţăDiscretă sdSculă, // geometria capului utilizat
        Boolean bDirecţiaSus // înfăşurătoarea în sus sau în jos
        Boolean bTotul // toată suprafaţa sau în mască
        )
{
    Real rtemp, rZ;
```

```
PentruFiecare(Întreg nSDX = 1; nSDX <= nX; nSDX++)
    PentruFiecare(Întreg nSDY = 1; nSDY <= nY; nSDY++)
    {
        Dacă(bDirecţiaSus)
            rTemp = -1e100;
        Altfel
            rTemp = 1e100;

        rScala = sdScalare[nSDX, nSDY];
        sdSculă.Scalcază(rScală); // !! Aici scalarea se face cu recrearea SD !!

        PentruFiecare(Întreg nSCX = 1; nSCX <= nX; nSCX++)
            PentruFiecare(Întreg nSCY = 1; nSCY <= nY; nSCY++)
                Dacă(bTotul SAU sdSuprafaţa.m[nSCX, nSCY]) ŞI
                    sdSculă.m[nSCX, nSCY] ŞI nSDX + nSCX < nX ŞI nSDY + nSCY < nY)
                    Dacă(bDirecţiaSus)
                    {
                        rZ=sdSuprafaţă[nSDX + nSCX, nSDY + nSCY] + sdSCulă[nSCX, nSCY];
                        Dacă(rZ > rTemp) rTemp = rZ;
                    } // Dacă
                    Altfel
                    {
                        rZ = sdSuprafaţă[nSDX + nSCX, nSDY + nSCY]-sdSCulă[nSCX, nSCY];
                            Dacă(rZ < rTemp) rTemp = rZ;
                    } // Altfel
        PuneLa(nSDX, nSDY, rTemp);
    } // PentruFiecare
} // EndInitCapDeSculăDinamic
```

Această metodă este foarte simplă şi asigură o varietate de echidistanţări şi racordări, autorul considerând-o unică la ora actuală şi aproape imposibil de realizat folosind alte tehnici de stocare a suprafeţei. În fabricaţie, algoritmii de racordare şi echidistanţare cu calculul total de interferenţă sculă-semifabricat sunt, pe departe, cei mai complecşi. Cea mai importantă utilizare a lor este aceea de a converti SD în trasee de sculă, care se poate observa că devine foarte simplă după aplicarea acestei corecţii de echidistanţare.

Având suprafaţa echidistantă, problema racordărilor globale sau locale, dinamice, cu orice geometrie, devine un algoritm banal. Racodările în sus se execută echidistanţând prima oară în sus, apoi suprafaţa echidistantă se echidistanţează în jos, rezultând SD de pornire, care este racordată cu geometria dorită. Racordările în direcţie contrară se execută pornind echidistanţarea în jos, urmată de cea în sus.

```
NimicSuprafaţăDiscretă.InitRacordareDinamic(  înfăşoară în spatiu o SD sculă
      SuprafaţăDiscretă sdSupafaţa,  suprafata care trebuie înfăşurată
      SuprafaţăDiscretă sdScalare,  geometria coeficienţilor de scalare
      SuprafaţăDiscretă sdSculă,  geometria capului utilizat
      Boolean bDirecţiaSus,  înfăşurătoarea în sus sau în jos
      Boolean bTotul  toată suprafata sau în mască
      )
{
   SuprafaţăDiscretă sdTemp;
   sdTemp.InitÎnfăşurătoareDinamic(sdSupafaţa, sdScalare, sdSculă, bDirecţiaSus, bTotul);
   InitÎnfăşurătoareDinamic(sdTemp, sdScalare, sdSculă, !bDirecţiaSus, bTotul);
} EndInitRacordareDinamic
```

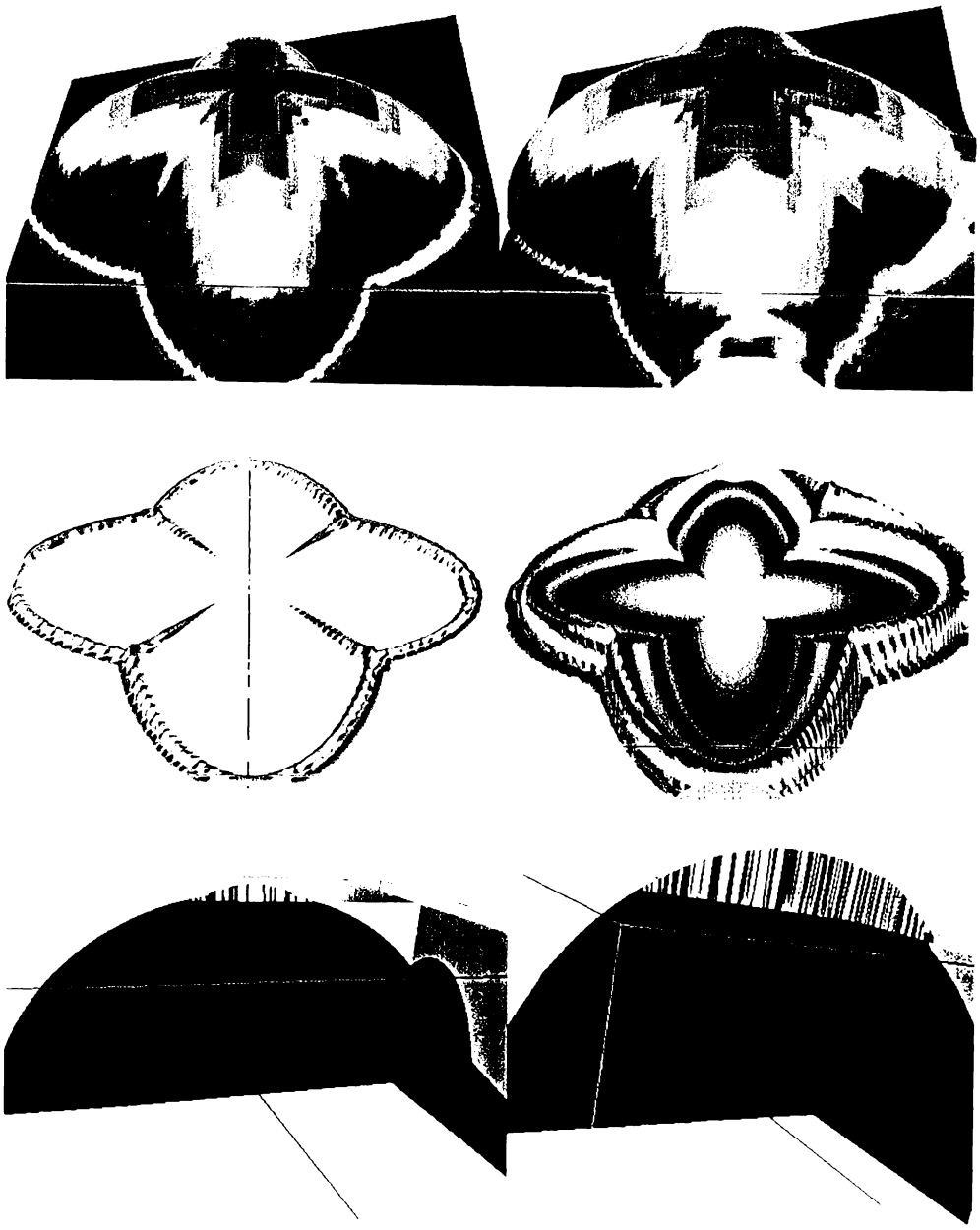


Figura 4.6 *Exemple de înfăşurători şi racordări statice (coloana 1) şi dinamice (coloana 2)*
S-au prezentat suprafeţele după racordare, cu culoarea în funcţie de valoarea normalei, plus materialul adăugat în valoare absolută, şi câteva secţiuni solide.

4.3. Filtre

Filtrele sunt metode auxiliare, care au scopul principal de a curăţa SD de eventualele zgomote create în diferite procese de conversie sau interpolare, sau sunt folosite în algoritmi de export, pentru a asigura exportul unor curbe continue lipsite de zgomot.

Acest tipuri de filtre sunt în general folosite de algoritmii de conversie vectorială, în cazul în care nu sunt descrise pe suprafeţele stocate vectorial funcţii de proiecţie **real** GetZ(**real** rX, **real** rY).

4.3.1. Filtre de creare

Aceste filtre crează un punct ca medie a vecinilor, în cazul în care el este sub o valoare, iar vecinii lui într-o direcţie sunt mai mari decât acea valoare. Se poate observa că toţi algoritmii de filtrare returnează numărul de puncte corectate.

```
Întreg SuprafaţăDiscretă.CreazăMască2()
{
   Întreg nIndex = 0;
   PentruFiecare(Întreg nSDX = 2; nSDX < nX; nSDX++)
      PentruFiecare(Întreg nSDY = 2; nSDY < nY; nSDY++)
      {
         Dacă(m[nSDX – 1, nSDY] ŞI m[nSDX + 1, nSDY])
         {
            PuneLa(nSDX, nSDY, ([nSDX – 1, nSDY] + [nSDX + 1, nSDY])/2);
            m.PuneLa(nSDX, nSDY, ADEVĂRAT);
            nIndex++;
         } /Dacă
         Dacă(m[nSDX, nSDY – 1] ŞI m[nSDX, nSDY + 1])
         {
            PuneLa(nSDX, nSDY, ([nSDX, nSDY – 1] + [nSDX, nSDY + 1])/2);
            m.PuneLa(nSDX, nSDY, ADEVĂRAT);
            nIndex++;
         } /Dacă
      } /PentruFiecare
   Întoarce(nIndex);
} /EndCreazăMască2
```

Un algoritm geamăn este acela care utilizează o analiză suplimentară a unui vecin de pe direcţia perpendiculară, pentru a-l face mai constrictiv, dar şi mai realist.

```
Întreg SuprafaţăDiscretă.CreazăMască3()
{
   Întreg nIndex = 0;
   PentruFiecare(Întreg nSDX = 2; nSDX < nX; nSDX++)
      PentruFiecare(Întreg nSDY = 2; nSDY < nY; nSDY++)
      {
         Dacă(m[nSDX – 1, nSDY] ŞI m[nSDX + 1, nSDY] ŞI m[nSDX, nSDY + 1])
         {
            PuneLa(nSDX, nSDY, ([nSDX–1, nSDY] + [nSDX + 1, nSDY] + [nSDX, nSDY – 1]) / 3.0);
            m.PuneLa(nSDX, nSDY, ADEVĂRAT);
            nIndex++;
         } /Dacă
         Dacă(m[nSDX – 1, nSDY] ŞI m[nSDX + 1, nSDY] ŞI m[nSDX, nSDY – 1])
```

```
        {
            PuneLa(nSDX, nSDY, (|nSDX-1, nSDY| + |nSDX+1, nSDY| + |nSDX, nSDY-1|) / 3.0);
            m.PuneLa(nSDX, nSDY, ADEVĂRAT);
            nIndex++;
        } Dacă

        Dacă(m[nSDX, nSDY - 1] ŞI m[nSDX, nSDY + 1] ŞI m[nSDX + 1, nSDY])
        {
            PuneLa(nSDX, nSDY, (|nSDX, nSDY-1| + |nSDX, nSDY + 1| + |nSDX+1, nSDY|)/3.0);
            m.PuneLa(nSDX, nSDY, ADEVĂRAT);
            nIndex++;
        } Dacă
        Dacă(m[nSDX, nSDY - 1] ŞI m[nSDX, nSDY + 1] ŞI m[nSDX - 1, nSDY])
        {
            PuneLa(nSDX, nSDY, (|nSDX, nSDY-1| + |nSDX, nSDY+1| + |nSDX-1, nSDY|) / 3.0);
            m.PuneLa(nSDX, nSDY, ADEVĂRAT);
            nIndex++;
        } Dacă
    } PentruFiecare
    Întoarce(nIndex);
} EndCreazăMască3
```

Se va opri aici generarea listingului, încercându-se în continuare să se expună doar numele şi destinaţia filtrului respectiv, şi considerându-se implementarea trivială.

Filtrul Întreg CreazăMască4(), folosit la eliminarea punctelor singulare, este cel mai nedistructiv şi foarte utilizat.

Similar cu cele trei filtre expuse aici se definesc trei filtre utilizate pentru asigurarea continuităţii şi crearea curbelor care nu conţin zgomot în algoritmii de export a curbelor echipotenţiale. S-au imaginat trei metode, care au acelaşi nume, dar conţin un parametru de tip real. Se listează în continuare doar Întreg CreazăValoare2(**Real** rZ).

```
Întreg SuprafaţăDiscretă.CreazăValoare2(Real rZ)
{
    Întreg nIndex = 0;
    PentruFiecare(Întreg nSDX = 2; nSDX < nX; nSDX++)
        PentruFiecare(Întreg nSDY = 2; nSDY < nY; nSDY++)
        {
            Dacă(|nSDX - 1, nSDY| > rZ ŞI |nSDX + 1, nSDY| > rZ ŞI |nSDX, nSDY| < rZ)
            {
                PuneLa(nSDX, nSDY, (|nSDX - 1, nSDY| + |nSDX + 1, nSDY|) / 2.0);
                m.PuneLa(nSDX, nSDY, ADEVĂRAT);
                nIndex++;
            } Dacă

            Dacă(|nSDX, nSDY - 1| > rz ŞI |nSDX, nSDY + 1| > rZ ŞI |nSDX, nSDY| < rZ)
            {
                PuneLa(nSDX, nSDY, (|nSDX, nSDY - 1| + |nSDX, nSDY + 1|) / 2.0);
                m.PuneLa(nSDX, nSDY, ADEVĂRAT);
                nIndex++;
            } Dacă
        } PentruFiecare
    Întoarce(nIndex);
} EndCreazăValoare2
```

Un filtru mai evoluat este acela care umple cu o valoare (şi implicit masca), eliminând zgomote de formă mai rebelă, numărând nodurile adiacente mai mici de o valoare dată şi umplând suprafaţa unde acest număr este mai mic decât o valoarea admisibilă. Acest algoritm este utilizat şi în jocul de Go, la numărarea pieselor componente ale unui grup. Este lesne de înţeles că, complexitatea acestui filtru nu este mică, necesitând chemări recursive şi marcări într-o mască auxiliară; din aceste considerente nu va fi exemplificat în cod sursă.

Întreg SuprafaţăDiscretă.CreazăMască(**Întreg nMaxPuncte**);
Întreg SuprafaţăDiscretă.CreazăValoare(**Întreg nMaxPuncte**, **Real rZ**);

4.3.2. Filtre de distrugere

Inversând logica din filtrele de creare se pot genera filtrele de distrugere, obţinându-se astfel familia de filtre.

Întreg SuprafaţăDiscretă.DistrugeMască2();
Întreg SuprafaţăDiscretă.DistrugeMască3();
Întreg SuprafaţăDiscretă.DistrugeMască4();
Întreg SuprafaţăDiscretă.DistrugeMască(**Întreg nMaxPuncte**);

Întreg SuprafaţăDiscretă.DistrugeValoare2(**Real rZ**);
Întreg SuprafaţăDiscretă.DistrugeValoare3(**Real rZ**);
Întreg SuprafaţăDiscretă.DistrugeValoare4(**Real rZ**);
Întreg SuprafaţăDiscretă.DistrugeValoare(**Întreg nMaxPuncte**, **Real rZ**);

Câteva exemple despre **Filtre** se dau în figura următoare, unde s-au prezentat câteva conversii în curbe a unor SD, cu şi fără filtrare.

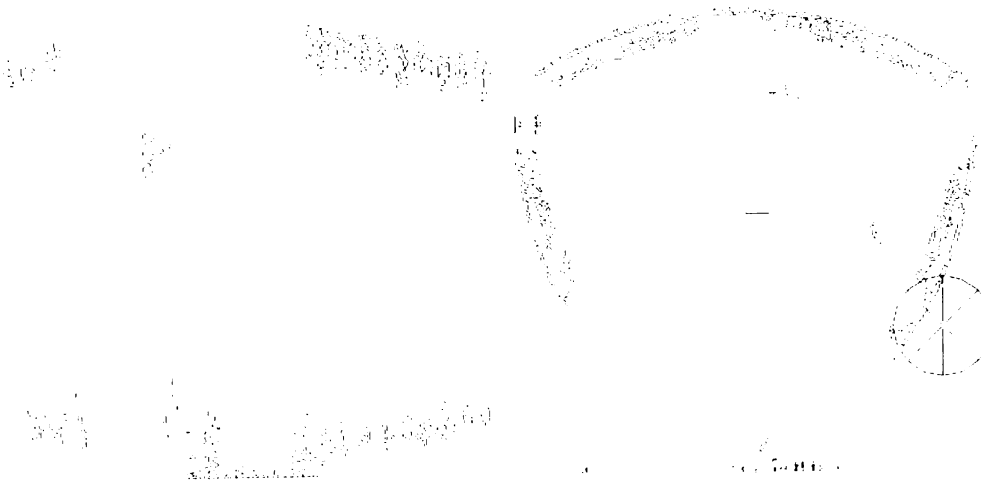


Figura 4.7 *Exemple de export a unor zone critice de detecţie a zonelor plane pe solide, importate din MasterCAM şi MicroStation (aceeaşi analiză, cu şi fără filtrări)*

4.4. Metode de generare vectorială

Metodele de generare vectorială sunt foarte importante în conversia datelor din alte modelatoare, care, în general, se bazează pe această metodă de modelare.

În general, suprafeţele provenite sunt de tip plasă (format **DXF**) sau format listă cu triunghiuri (format **STL**). Acestea sunt cele mai simple formate posibile şi au marele avantaj că sunt foarte uşor de generat metode de *import - export* care să le suporte.

Un alt mare avantaj al reprezentărilor de nivel scăzut este acela că se pot defini un set destul de decent (din punct de vedere al performanţei) de metode de rezolvare neiterative a proiecţiilor şi secţiunilor.

Marele lor dezavantaj este acela că distrug orice informaţie logică despre modul de generare al unui solid, distrug eventualele informaţii parametrice, care în prezent sunt suportate numai de formatele **STEP** şi **SAT (ACIS)**.

În concluzie, pentru acest tip de date de intrare elementare se pot defini, fără nici o problemă, metode de conversie relativ simple şi viabile din punct de vedere al performanţei.

```
NimicSuprafaţăDiscretă.InitPlasă( //converteşte o plasă
      CPlasa IIn //lista cu plasele de intrare
      )
{
   PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
      PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
         if (IIn.ArcOProiecţie(IndexÎnRealPtX(nI), IndexÎnRealPtY(nJ))
            PuncLa(nI, nJ, IIn.IaDeLaZ(IndexÎnRealPtX(nI), IndexÎnRealPtY(nJ)), PM_MAX);
} //InitPlasă

NimicSuprafaţăDiscretă.InitPlase( //converteşte o familie de plase
   FamiliaDePlase flIn //lista cu familia de plasele de intrare
      )
{
   PentruFiecare(Întreg nI = 1; nI <= flIn.nNr; nI++)
      InitPlasă(flIn[nI]);
} //InitPlase
```

Nu s-au discutat listele cu date de tip triunghi, deoarece acestea se pot trata într-un mod similar.

Metodele de conversie discutate asigură un set decent de conversii, dar nu şi eficient, în cazul în care pentru suprafeţele care trebuiesc convertite solidul nu poate sau încă nu sunt generate metode de conversie eficiente în format triunghiular sau patrulater, ci doar foarte lente metode de proiecţie iterative. Aceste condiţii duc la timpi de conversie care sunt de domeniul orelor, în locul celor prezentate, care sunt de domeniul secundelor.

Deci, din punct de vedere al eficenţei se consideră ca fiind inacceptabil. Pentru acest tip de date a fost imaginat un algoritm inspirat din **reţelele neuronale,** care privesc SD ca pe o reţea care se doreşte a fi antrenată cu datele provenite dintr-o reprezentare de curbe B-Spline (spre exemplu) limitată de curbe (trimmed B-Spline). Acest algoritm este larg utilizat de autor în conversia din toate formatele de nivel superior, fiind prezentat în continuare, în premieră.

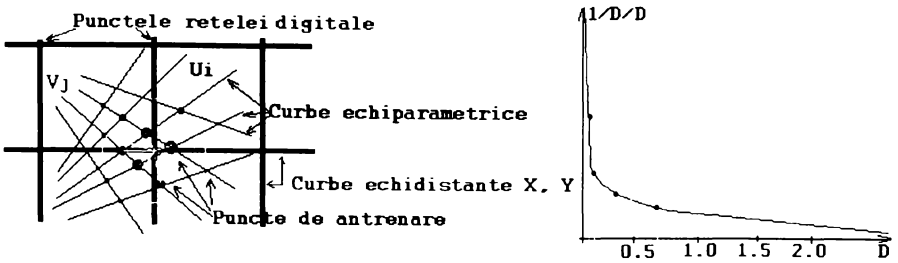


Figura 4.8 *Exemplificarea metodei de antrenare a reţelei neuronale asociată suprafeţei discrete, reprezentarea curbelor echiparametrice U, V, a punctelor de intersecţie*

Algoritmul, utilizează analiza punctelor care sunt în vecinătatea unui punct.

Algoritmul este într-un fel asemănător cu cel al interpolării prin puncte. Diferenţa de bază este că algoritmul de interpolare prin set de puncte operează foarte bine când sunt relativ puţine puncte, timpii de rulare crescând linear cu numărul de puncte. Algoritmul prezentat în prezent este utilizat pentru antrenarea unui punct din 2..10 puncte de intrare.

Acest algoritm este destul de precis, asigurând pentru o medie de trei puncte de antrenare pe punct al SD o precizie medie de 0,001 mm, ceea ce se consideră satisfăcător.

Importanţa acestui algoritm este imensă d.p.d.v. al consumului de timp, deoarece, în realitate, calcularea unui punct echiparametric (U, V) de pe suprafaţă este, în general, de peste 100 de ori mai rapidă decât calcularea iterativă a unui punct la o coordonată (X, Y), deci pentru un set decent de antrenare (de 3 puncte pe punct al SD) se asigură timpi de peste 30 de ori mai mici. În realitate, după ce s-a cronometrat pe un set de 10 repere de complexităţi diferite, se poate spune că s-au obţinut timpi mai mici de 20..100 ori.

Algoritmul asigură o rezolvare destul de simplă şi elegantă a problemei. Metoda constă din:

- ⇨ *pentru fiecare punct P(U,V) se calculează cei patru vecini discreţi;*
- ⇨ *se calculează distanţa pătratică, cu relaţia* $D^2 = DX^2 + DY^2$;
- ⇨ *aceasta este utilizată pentru a antrena cei patru vecini ai unui punct P(u, v) după legea* $1/D^2$ *(variaţia acestei funcţii se găseşte în figura 4.8 b);*
- ⇨ *la sfârşitul algoritmului, pentru a întoarce valoarea reală, se divide cu suma ponderilor.*

Matematic:

$$\Sigma(P(U, V) / D^2) / \Sigma D^2$$

SuprafaţăDiscretă sdSumaPD, sdSumaD;

```
NimicSuprafațăDiscretă.InitPuncLaR()
{
    sdSumaPD = sdAceasta; // inițializează SumaPD
    sdSumaPD.PuncLaAll(0.0); // o inițializează pe 0.0
    sdSumaD = sdAceasta; // inițializează SumaD
    sdSumaD.PuncLaAll(0.0); // o inițializează pe 0.0
} //EndInitPutR

SuprafațăDiscretă.DonePuneLaR()
{
    sdSumaPD /= sdSumaD; //împarte sd SumaPD/SumaD
    PânăCând(sdSumaPD.Crează2()); //filtrează sd pentru eventualitatea pt. puncte necreate
    PânăCând(sdSumaPD.Distruge2()); //filtrează sd pentru eventualitatea pt. puncte necreate
    Combină(sdSumaPD, PM_MAX, FALS); //combină sd actual cu sd, SumaPD luând numai masca
} //EndDonePutR

NimicSuprafațăDiscretă.PuncLaR(PunctpP, Real rTaieSus = 2.0) //antrenează vecinii
{
    Real rX = RealÎnIndexPtXR(pP.iX); // convertește valoarea reală dar din spațiul intreg
    Real rY = RealÎnIndexPtYR(pP.iY); // convertește valoarea reală dar din spațiul intreg
    Real rD2;
 //P00
    rD2 = (rX – int(rX))*(rX – int(rX)) + (rY – int(rY))*(rY – int(rY));
    Dacă (rD2 < rTaieSus)
    {
        sdSumaPD.PuncLa(int(rX) + 0, int(rY) + 0, pP.Z/rD2); //adună pe PD
        sdSumaD .PuncLa(int(rX) + 0, int(rY) + 0, 1/rD2); //adună pe PD
    } //Dacă

 //P01
    rD2 = (rX – int(rX))*(rX – int(rX)) + (rY + 1 – int(rY))*(rY + 1 – int(rY));
    Dacă (rD2 < rTaieSus)
    {
        sdSumaPD.PuncLa(int(rX) + 0, int(rY) + 1, pP.Z/rD2); //adună pe PD
        sdSumaD .PuncLa(int(rX) + 0, int(rY) + 1, 1/rD2); //adună pe PD
    } //Dacă

 //P10
    rD2 = (rX + 1 – int(rX))*(rX + 1 – int(rX)) + (rY – int(rY))*(rY – int(rY));
    Dacă (rD2 < rTaieSus)
    {
        sdSumaPD.PuncLa(int(rX) + 1, int(rY) + 0, pP.Z/rD2); //adună pe PD
        sdSumaD .PuncLa(int(rX) + 1, int(rY) + 0, 1/rD2); //adună pe PD
    } //Dacă

  P11
    rD2 = (rX + 1 – int(rX))*(rX + 1 – int(rX)) + (rY + 1 – int(rY))*(rY + 1 – int(rY));
    Dacă (rD2 < rTaieSus)
    {
        sdSumaPD.PuncLa(int(rX) + 1, int(rY) + 1, pP.Z/rD2); //adună pe PD
        sdSumaD .PuncLa(int(rX) + 1, int(rY) + 1, 1/rD2); //adună pe PD
    } //Dacă
} //EndPutR

NimicSuprafațăDiscretă.PuncLaSuprafațaSuperioară(SuprafațăSuperioară sS; Real rUStep, rVStep)
{
    InitPuncLaR();
    PentruFiecare(Real rU = 0.0; rU<= 1.0 rU += rUStep)
        PentruFiecare(Real rV = 0.0; rV<= 1.0 rV += rVStep)
            PuncLaR(sS[rU, rV]);
    EndPuncLaR();
```

} *EndPutSuprafataSuperioară*

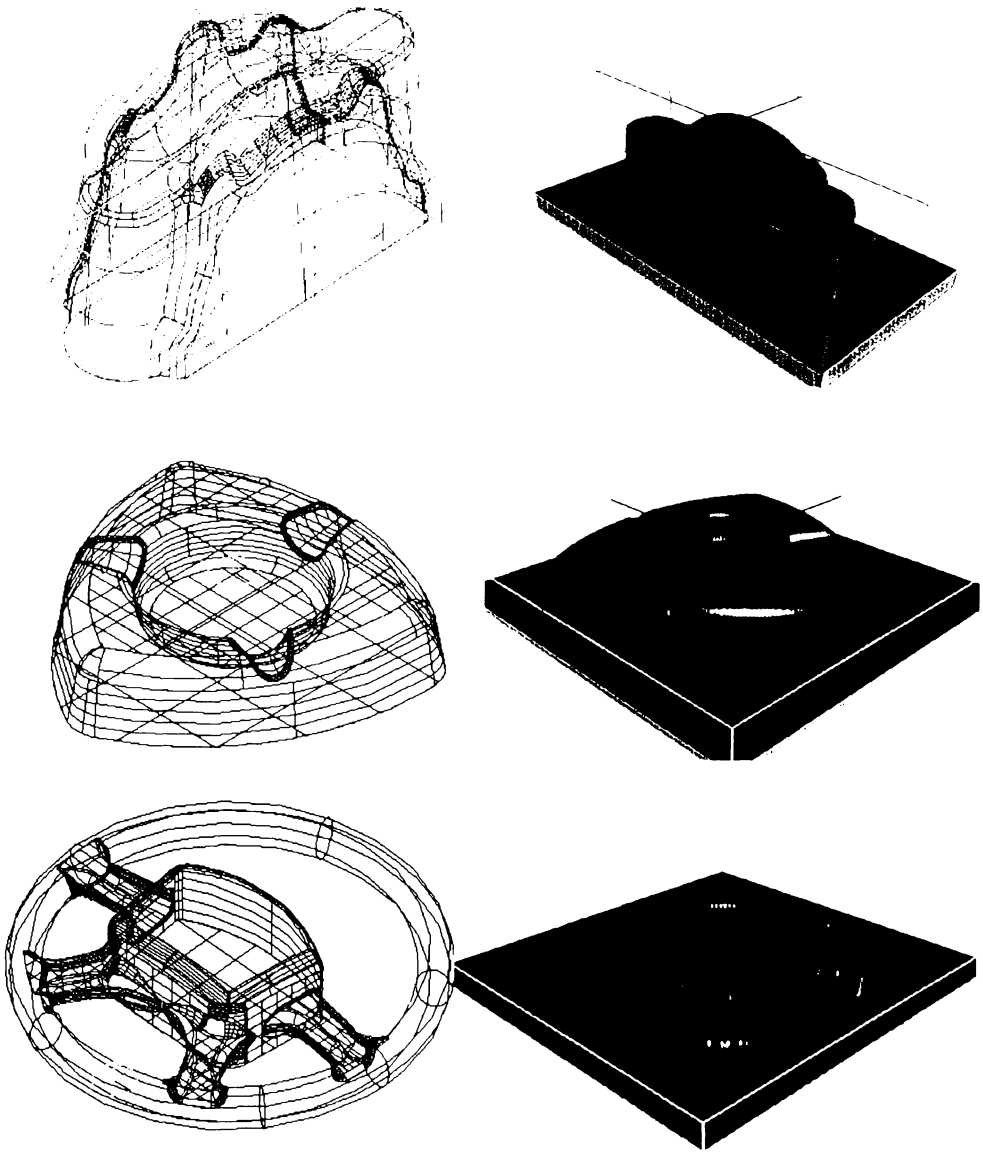


Figura 4.9 *Importul a trei repere exportate ca suprafaţe BSpline sau solide BRep din EdgeCAM, AutoCAD şi Microstation (Solid Edge).*
Exemple în care s-a folosit reţeaua neuronală pentru antrenarea SD. Timpi de import 1..3 minute. Reprezentarea solidă utilizează nuanţele de albastru pentru marcarea zonelor nefrezabile cu o sculă dată de r=20 mm.

4.5. Concepte introduse

```
Clasa SuprafaţăDiscretă DerivatăDin BazăDiscretă
{
//.. metodele din capitolele precedente !
   Nimic Combină(SuprafaţăDiscretă sdComb, Întreg nModulDePunere = PM_NONE, Boolean bTotul =
   ADEVĂRAT);
   Nimic PuneLaOrizontal(Real rValoare, Întreg nModulDePunere = PM_NONE, Boolean bTotul = ADEVĂRAT);
   Nimic PuneLaPlan3P(Punctp1, p2, p3, Întreg nModulDePunere = PM_NONE, Boolean bTotul = ADEVĂRAT);
   Nimic PuneLaFuncţie(SetDeCaractere strFuncţia, Întreg nModulDePunere = PM_NONE, Boolean bTotul =
   ADEVĂRAT);

   Nimic PuneLaInterpolareXDir(SuprafaţăDiscretă sdSecţiuni, Întreg nOrdinCurbei= 1, Întreg nModulDePunere =
   PM_NONE, Boolean bTotul = ADEVĂRAT);
   Nimic PuneLaInterpolareYDir(SuprafaţăDiscretă sdSecţiuni, Întreg nOrdinCurbei= 1, Întreg nModulDePunere =
   PM_NONE, Boolean bTotul = ADEVĂRAT);

   Nimic InterpoleazăPrinPuncte(Curbă cListaCuPuncte, FamiliaDeReali lrListaCuPonderi, FamiliaDeCurbe
   fcAlteCurbe, FamiliaDeReali lrPonderiPtCurbe, FamiliaDeReali lrPaşiiPtCurbe, Real rRigiditatea);

   Nimic InitCapDeSculă(Curbă cCapDeSculă, Real rPas);
   Nimic InitÎnfăşurătoareStatic(SuprafaţăDiscretă sdSupafaţa, sdSculă, Boolean bDirecţiaSus, Boolean bTotul);
   Nimic InitRacordareStatic(SuprafaţăDiscretă sdSupafaţa, sdSculă, Boolean bDirecţiaSus, Boolean bTotul);
   Nimic InitÎnfăşurătoareDinamic(SuprafaţăDiscretă sdSupafaţa, sdScalare, sdSculă, Boolean bDirecţiaSus,
   Boolean bTotul);
   Nimic InitRacordareDinamic(SuprafaţăDiscretă sdSupafaţa, sdScalare, sdSculă, Boolean bDirecţiaSus, Boolean
   bTotul);

   Nimic InitPlasă(CPLasa lIn);
   Nimic InitPlase(FamiliaDePlase flIn);
   Nimic PuneLaSuprafaţaSuperioară(SuprafaţăSuperioară sS; Real rUStep, rVStep);
   Nimic PuneLaSuprafeţeSuperioare(FamiliaDeSuprafaţeSuperioare fsFS);

//câteva filtre
   Întreg CreazăMască2();
   Întreg CreazăMască3();
   Întreg CreazăMască4();
   Întreg CreazăMască(Întreg nMaxPuncte);
   Întreg CreazăValoare2(Real rZ);
   Întreg CreazăValoare3(Real rZ);
   Întreg CreazăValoare4(Real rZ);
   Întreg CreazăValoare(Întreg nMaxPuncte, Real rZ);
   Întreg DistrugeMască2();
   Întreg DistrugeMască3();
   Întreg DistrugeMască4();
   Întreg DistrugeMască(Întreg nMaxPuncte);
   Întreg DistrugeValoare2(Real rZ);
   Întreg DistrugeValoare3(Real rZ);
   Întreg DistrugeValoare4(Real rZ);
   Întreg DistrugeValoare(Întreg nMaxPuncte, Real rZ);

  Metode utilizate de automatul neuronal la importul fanilulor de Supr. sup
   Nimic InitPuncLaR();
   Nimic DonePuncLaR();
   Nimic PuncLaR(PunctpP, Real rTaieSus = 2.0); antrenează cei patru vecini
} EndSuprafaţăDiscretă
```

4.6. Concluzii

În acest capitol au fost descrise tehnici şi metode de *creare, modelare, conversie, import şi filtrare* a SD.

Scopul acestuia este ca, utilizând metodele descrise, un număr cât mai mare de repere provenite din diferite sisteme de proiectare, fabricaţie sau modelate direct, să poată beneficia de soluţiile sofisticate de analiză şi generare a codului NC care vor fi discutate în viitor.

De asemenea, s-au prezentat în premieră trei algoritmi noi, concepuţi de către autor:

- ⇨ *algoritmul de import şi conversie a seturilor de puncte şi curbe furnizate fără nici o regulă;*
- ⇨ *algoritmul de calculare a înfăşurătoarei şi racordărilor cu forme de orice geometrie (un caz particular al acestora sunt capetele de sculă suprafeţe de revoluţie, utilizate în frezare);*
- ⇨ *reţeaua neuronală pentru antrenarea cu date care nu cad în punctele reţelei utilizabile, ca o metodă generică de import a tuturor datelor parametrice.*

A fost prezentată o familie de metode auxiliare (filtrele) utilizate pentru reducerea zgomotului introdus de diferite metode de conversie şi analiză. S-a exemplificat acest concept.

5. Metode de conversie şi formate de import – export

5.1. Introducere

În acest capitol se vor sintetiza şi prezenta câteva conversii ale SD în reprezentările vectoriale uzuale altor sisteme de proiectare şi/sau fabricaţie, pentru a da nu numai o consistenţă vizuală analizelor şi generărilor, ci şi o finalitate şi utilizabilitate în alte sisteme de proiectare - fabricare.

În prima parte a capitolului se vor prezenta metode de culegere a datelor şi de convertire a lor în formate de tip **plasă**, solide discrete faţetate reprezentate ca **triunghiuri**, reprezentări de tip **familie de curbe**, specifice generării fişierului NC via CL.

Se vor introduce metode de creare de *curbe, proiecţie, offset inteligent şi export* în formatele simple ASCII, ca: ***DXF, STL, CL, NC.*** Nu se vor discuta, pentru a nu încărca expunerea, formatele evoluate, ca ***IGES, STEP, VDA,*** având în vedere şi utilizarea lor destul de restrânsă, deocamdată. Expunerea fiecărui format superior ar necesita un minim de 20 pagini, ceea ce nu asigură cadrul lucrării de faţă.

Metodele de conversie şi export sunt de o mare importanţă în utilizarea suprafeţelor discrete demulabile, virtual, în orice sistem de proiectare şi fabricaţie. Ideea prezentării fiecărui format constă în a îmbina prezentarea acestuia, însoţită de expunerea algoritmilor în pseudocod, cu exemplificarea fiecărui concept introdus, cu ajutorul exemplelor şi a listingului potenţial, realizat de către clasa expusă.

Se va crea un set nou de clase, specifice exportului fiecărui tip în parte: ***Fişier, CDXFOut, CSTLOut, CLOut, CNCOut.*** Aceste clase vor fi implementate folosind o metodă unificată de prezentare, încercând să se ascundă detaliile fiecărui format în parte.

Se vor expune metode noi, destinate conversiei şi exportului, metode care dau o utilizabilitate SD, legându-le de alte sisteme de proiectare, ca aparate matematice auxiliare de analiză sau conversie în format NC.

Se va prezenta, în premieră, un algoritm de conversie în curbe de nivel foarte fin (comparativ cu pasul suprafeţei discrete), care permite conversia SD corecţie de sculă în format NC, asigurând erori de ordinul micrometrilor.

Un subcapitol aparte va fi rezervat expunerii problemei unui postprocesor generic de control numeric, **GNCPP (generic numeric control post processor)**, o librărie dinamică complexă, care are scopul de a genera fişier NC specific, virtual, pe orice echipament, optimizat pentru lungime şi timp de rulare.

Se vor trasa câteva statistici despre frecvenţa utilizării diferitelor formate în Marea Britanie şi SUA, în funcţie de numărul de maşini unelte şi numărul de angajaţi.

În finalul capitolului se vor rezuma, în pseudocod, toate metodele şi obiectele noi, şi se va concluziona asupra problemelor expuse pe parcursul capitolului.

5.2. Metode de conversie

Datorită modului oarecum neuzual de stocare a informaţiilor în suprafeţele discrete, pentru ca acestea să fie utilizabile în cât mai multe sisteme de proiectare şi fabricaţie, s-au creat tehnici şi metode de conversie şi export a SD în diferite reprezentări geometrice. Acestea sunt: *curbele, plasele patrulatere şi cele triunghiulare.*

5.2.1. Conversia în curbe

Conversiile SD şi proiecţiile unor curbe aplicate pe SD au un rol important în generarea fişierelor NC, sau în crearea unor extensii a unor zone critice.

Unicul caz de conversie a SD în curbe discutat în această lucrare va fi acela de conversie a unei SD în curbe de nivel.

Acest algoritm are utilitate nu numai în generarea fişierelor NC cu Z =ct, cum s-ar putea crede la o analiză superficială, ci aplicaţiile lui se pot extinde în orice generare de curbă de pe SD ! În capitolul următor, destinat analizei şi optimizării, se vor prezenta o multitudine de transformări particulare (manipulatori) specifice **analizelor** SD, în vederea utilizării acestui algoritm foarte puternic şi GENERIC, de export.

Problematica acestui algoritm este aceea că trebuie creată o familie de curbe care să despartă SD în două regiuni, una cu valori mai mari decât cea analizată, iar alta cu valori mai mici decât aceasta. În general, această problemă nu este nouă, şi se regăseşte într-o mulţime de programe **[LWRK], [CTR]**. Soluţiile propuse până acum sunt, în general, precare din punct de vedere al **calităţii** curbelor exportate, total improprii folosirii direct ca şi trasee de sculă, fără o rafinare uzual manuală, care încetineşte procesul şi duce la distrugerea generării şi regenerării automate.

În general, acest algoritm îşi găseşte utilitatea în toate locurile unde trebuie recunoscută o formă, făcând analogia cotei Z cu o culoare sau intensitate luminoasă, în cazul utilizării imaginilor monocrome, sau a trei componente (*roşu, verde, albastru*), în cazul analizării imaginilor color. Alte posibile utilizări ale algoritmului ar putea fi: *creşterea rezoluţiilor analogice ale fotografiilor, recunoaşterea formelor, convertirea imaginilor în format vectorial etc.*

Se consideră că algoritmul descris în continuare este unicul algoritm, cunoscut de autor, care utilizează proprietatea remarcabilă a unei secţiuni (sau proiecţii) *de a fi o curbă discretă continuă.* Deci, folosind teorema lui Cauchy între 2 puncte, unul mai mic şi unul mai mare decât o valoare dată, se găseşte cel puţin un punct care intersectează nivelul de analiză.

Imaginea de mai jos îşi propune prezentarea problematicii acestui algoritm.

Algoritmul se bazează pe proprietatea **remarcabilă** a fiecărui pătrat elementar (format din patru segmente de dreaptă, şi care formează întotdeauna un patrulater închis) de a avea un număr par de intersecţii cu un plan orizontal [0, 2, 4]. Deci, orice curbă care intră **trebuie** să şi iasă din pătrat, curbele neputând fi deschise (sau închise) decât în cazul în care încep şi se termină pe marginea SD.

Precizia curbelor create de acest algoritm (negru + roşu, punctele de control) este de ordinul 1/1000 * **pasul**, iar a celui bazat pe culegerea pasului, de ordinul **pasului** (verde + albastru, punctele de control).

În urma analizelor făcute, acest algoritm, combinat cu cel de detecţie a muchiilor vii, a produs, în toate cazurile analizate, rezultate similare: până la trei zecimale exacte, comparat cu algoritmi care folosesc tehnici pur vectoriale.

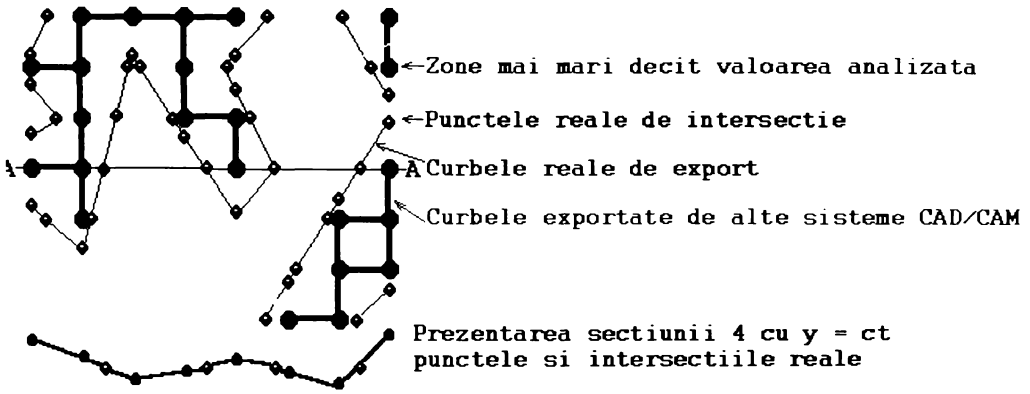


Figura 5.1 *Prezentarea algoritmului de conversie a SD în curbe echipotenţiale, folosind metoda clasică a punctelor de control şi cea nouă, care foloseşte continuitatea secţiunilor.*

Din păcate, erorile nu sunt liniare, ci sunt o funcţie de înclinare a suprafeţei. Astfel, la 90° , eroarea de detecţie a unui zid poate deveni comparabilă cu **pasul**, ceea ce duce la erori de domeniul 0.1..1.0 * pasul, de multe ori **netolerabile** ! Din această cauză, în sistemele evoluate destinate fabricaţiei, *TechoPack şi EdgeCAM*, la ora actuală se foloseşte o tehnologie mixtă de generare a fişierului NC: în proporţie de 90% din suprafaţă de pe SD, iar în zonele care au înclinaţii mari sunt folosite suprafeţele vectoriale, care sunt insensibile la schimbarea înclinaţiei, dar sunt de peste 500 de ori mai lente !

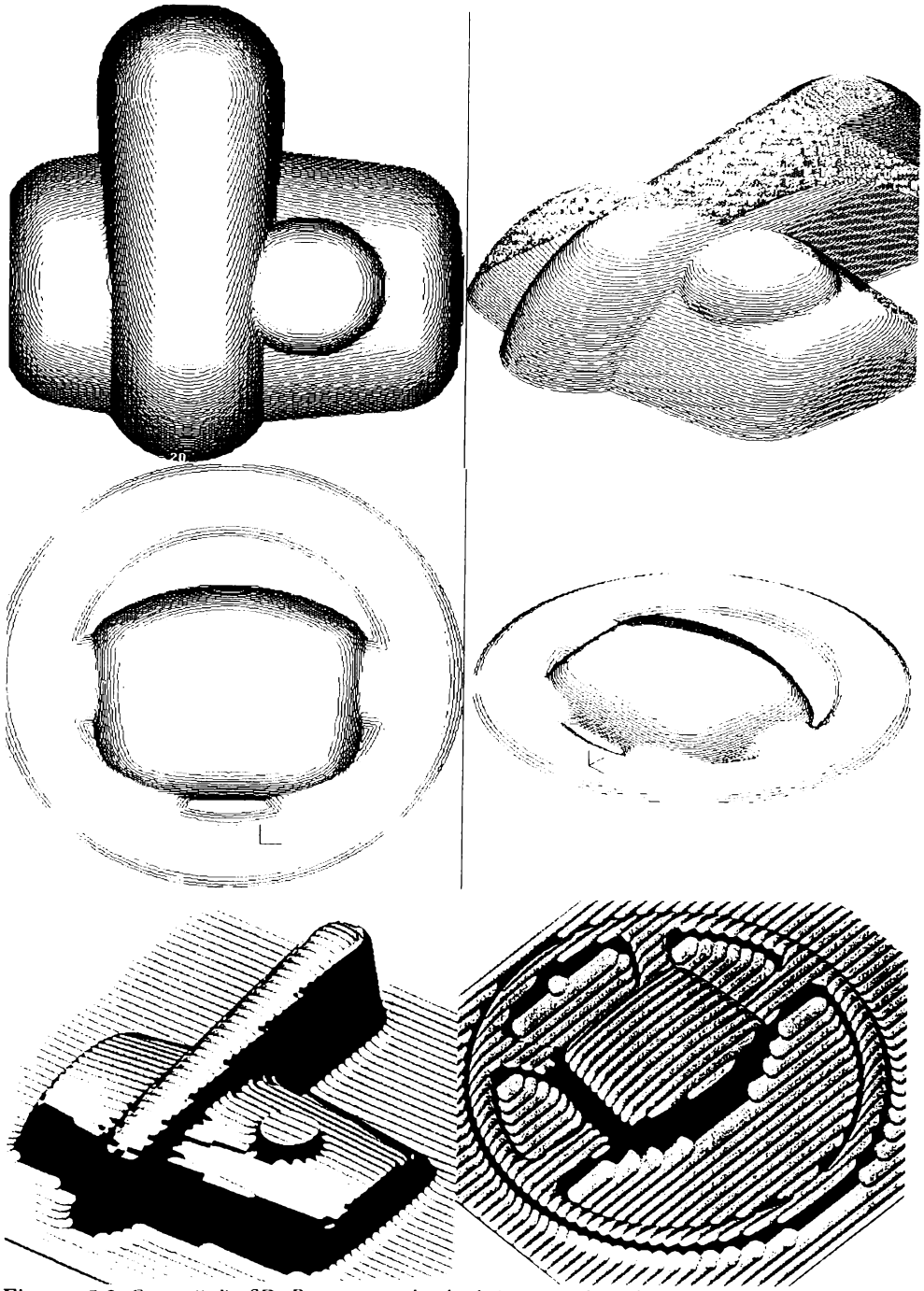


Figura 5.2 *Conversii din SD. Reprezentare de tehnologie mixtă, de combinare a ciclurilor de frezare curbe nivel şi secţiuni echidistante.*

Toate analizele şi fişierele NC au fost generate folosind reprezentarea discretă, cu algoritmul prezentat. Simularea şi verificarea este făcută cu DSView.

Algoritmul poate fi descris în pseudocod astfel:

```
NimicCDiscretSurface.AdunăCurbă(FamiliaDeCurbe fcOut, int nX, nY)
{
  prea laborioasă pentru a fi descrisă.
} //EndAdunăCurbă

NimicCDiscretSurface.CreazăCurbe(FamiliaDeCurbe fcOut, Real rPentruZ)
{
//y0
   PentruFiecare(Întreg nI = 1; nI < nX; nI++)
      Dacă ([nI, 0] > rPentruZ ŞI [nI + 1, 0] < rPentruZ SAU
            [nI, 0] < rPentruZ ŞI [nI + 1, 0] > rPentruZ)
         AdunăCurbă(fcOut, nI, 0);

//y max
   PentruFiecare(Întreg nI = 1; nI < nX; nI++)
      Dacă ([nI, nNrY] > rPentruZ ŞI [nI + 1, nNrY] < rPentruZ SAU
            [nI, nNrY] < rPentruZ ŞI [nI + 1, nNrY] > rPentruZ)
         AdunăCurbă(fcOut, nI, nNrY);

//x 0
   PentruFiecare(Întreg nJ = 1; nJ < nY; nJ++)
      Dacă ([0, nJ] > rPentruZ ŞI [0, nJ] < rPentruZ SAU
            [0, nJ] < rPentruZ ŞI [0, nJ] > rPentruZ)
         AdunăCurbă(fcOut, 0, nJ);

//x max
   PentruFiecare(Întreg nJ = 1; nJ < nY; nJ++)
      Dacă ([nNrX, nJ] > rPentruZ ŞI [nNrX, nJ] < rPentruZ SAU
            [nNrX, nJ] < rPentruZ ŞI [nNrX, nJ] > rPentruZ)
         AdunăCurbă(fcOut, nNrX, nJ);

//curbele închise interioare
   PentruFiecare(Întreg nI = 2; nI < nX-1; nI++)
      PentruFiecare(Întreg nJ = 2; nJ < nY-1; nJ++)
         AdunăCurbă(fcOut, nI, nJ);
} //EndCreazăCurbe

NimicCDiscretSurface.CreazăToateCurbele( //crează toate curbele
         FamiliaDeCurbe fcOut, //locul unde sunt stocate curbele
         Real rZStep = 1.0, //pasul
         Boolean  bAuto = ADEVĂRAT, //dacă calculează automat cotele de start şi sfârşit
         Real rZStart = 0.0, //cota de pornire
         Real rZEnd = 10.0) //cota de oprire

{
   Dacă(bAuto)
   {
      rZStart = pMax.rZ; //atribuie automat Z maxim
      rZEnd = pMin.rZ; //atribuie automat Z minim
   }
   PentruFiecare(Real rZ = rZStart; rZ < rZEnd; rZ += rZStep)
         CreazăCurbe(fcOut, rZ);
} //EndCreazăToateCurbele
```

NOTĂ: Detaliile de implementare a metodei AdunăCurbă() nu au fost expuse, deoarece metoda este destul de simplă, dar lungă; ea se bazează pe conceptele expuse la începutul capitolului.

5.2.2. Proiectarea unei familii de curbe pe suprafaţa discretă

Această metodă dă consistenţă şi generalitate algoritmului de conversie în curbe echipotenţiale şi deschide posibilităţi nelimitate pentru toţi algoritmii de generare a fişierului NC prin tehnici proiective (*spirale, radiale, offset, morf* etc).

Tehnica este destul de simplă şi este expusă în continuare:

```
NimicCDiscretSurface.Proiectează Curbă(Curbă cProiectează)
{
   PentruFiecare(Întreg nI = 1; nI < c.nNr; nI++)
   {
      Punctp = cProiectează[nI];
      p.rZ = laDeLaR(p.rX, p.rY);
      cProiectează.PuncLa(nI, p);
   } //PentruFiecare
} //EndProiectează Curbă

NimicCDiscretSurface.Proiectează Curbe(FamiliaDeCurbe fcProiectează)
{
   PentruFiecare(Întreg nI = 1; nI < c.nNr; nI++)
      Proiectează Curbă(fcProiectează[nI])
} //EndProiectează Curbe
```

Notă: algoritmul expus proiectează doar noduri; în mod similar se poate descrie un algoritm care proiectează toate liniile, cu un anumit pas dat; un exemplu despre cele două familii de algoritmi, cel de conversie în curbe echipotenţiale a unei SD, după care proiectarea pe o altă SD, se prezintă în figura 5.3.

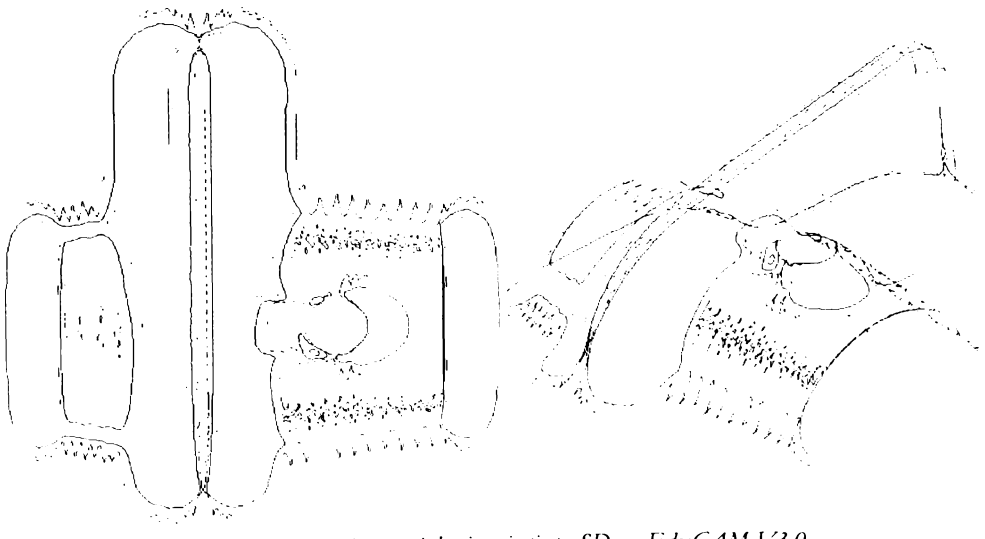


Figura 5.3 *Exemplu de conversie echipotenţială şi proiecţie pe SD, cu EdgeCAM V3.0*

5.2.3. Offset pe SD

Offsetul sau echidistanţarea unei curbe pe o SD este deosebit de utilă pentru generarea fişierelor NC de frezare cu rugozitate constantă (cea mai eficientă metodă de frezare). Algoritmul va fi prezentat în detaliu în subcapitolul "**Calculul curbelor echirugozitate**". Câteva exemple ale conceptului se pot observa în următoarea figură, unde se prezintă, în nuanţe diferite, offsetarea pe SD a unui dreptunghi sau a două drepte paralele.

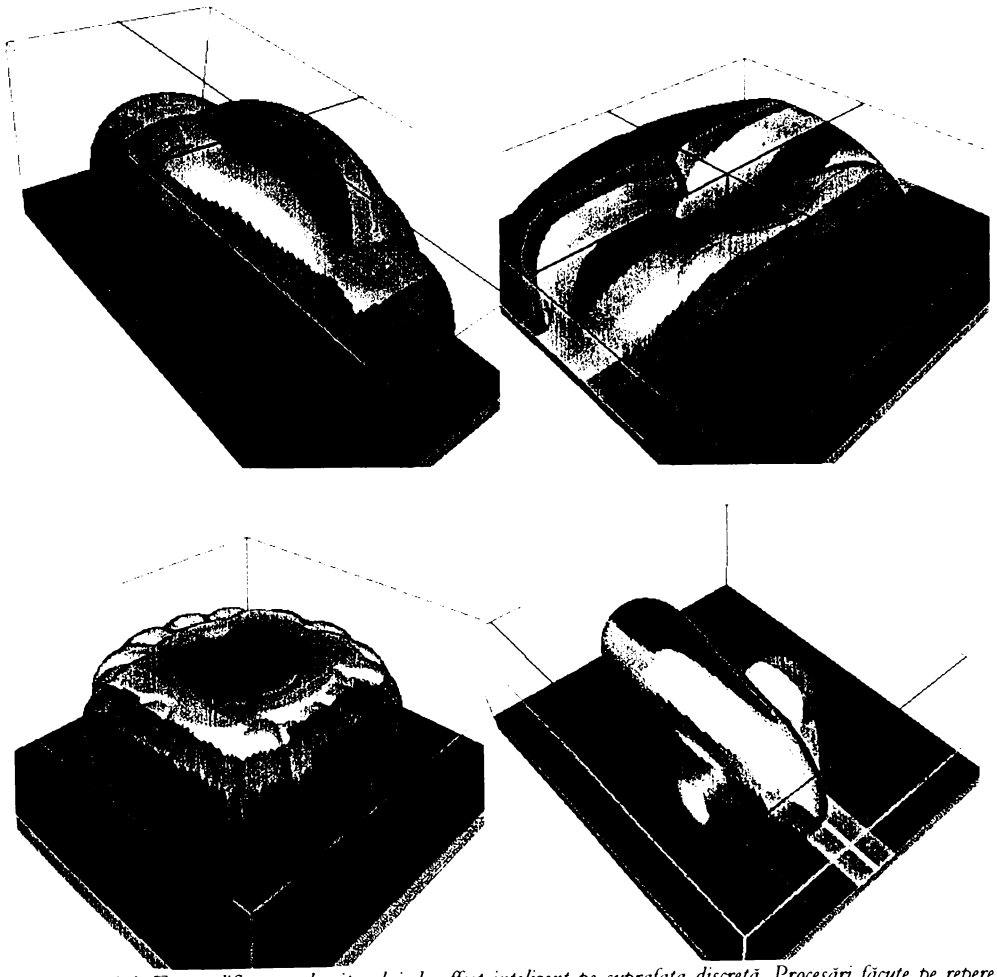


Figura 5.4 *Exemplificarea algoritmului de offset inteligent pe suprafaţa discretă. Procesări făcute pe repere importate din diferite sisteme CAD/CAM, pornind de la dreptunghi sau de la drepte paralele*

5.2.4. Convertirea în plase patrulatere

Această conversie este utilizată în cazul în care se doreşte exportul întregii pânze, din suprafaţa discretă într-un alt produs, via un format extern de export.

Cel mai simplu format care suportă plasele, ca entitate, este formatul **DXF**.

Se va exemplifica, în continuare, în pseudocod, algoritmul necesar creerii unei plase dintr-o SD:

```
NimicCDiscretSurface.ExportPlasă(SetDeCaractere strNumeDXF)
{
   Întreg nCuloare = 15
   SetDeCaractere strLayer = "Plasa";
   CDXFExport dxfout;

   dxfout.Iniţializează(strNumeDXF);
      dxfout.InitPlasă(nX, nY, nCuloare, strLayer);
         PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
            PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
            {
               Punctp(IndexÎnRealPtX(nI), IndexÎnRealPtY(nJ), |nI, nJ|);
               dxfout.PuncVertex(p);
            } //PentruFiecare
      dxfout.DonePlasă();
   dxfout.Sfârşeşte();
} //EndExportPlasă
```

Se poate observa că, conversia şi exportul într-o plasă uniform riglată este relativ simplă, atâta vreme cât este definită o clasă **CDXFOut**, care va fi prezentată pe parcursul capitolului.

Se prezintă, în continuare, două plase exportate din **TechnoCAD V2.0 [DM04..06]** şi vizualizate cu **DSView** .

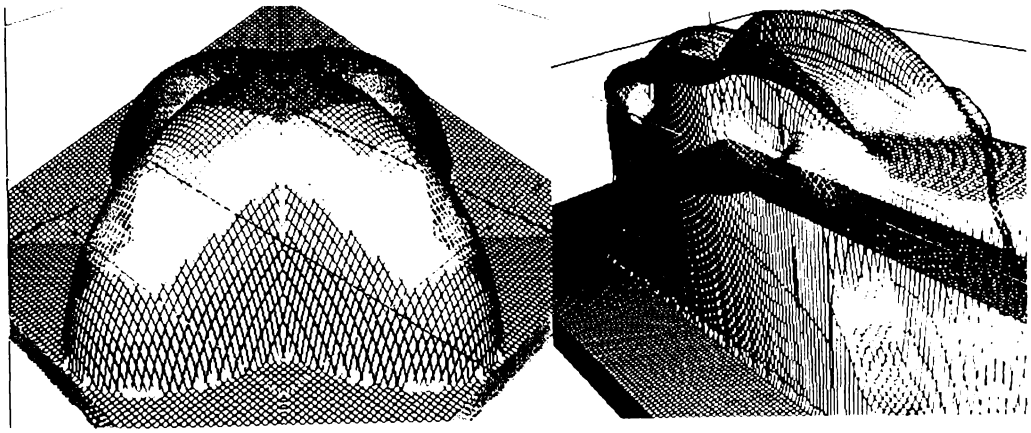


Figura 5.5 *Plase exportate şi convertite din suprafeţe discrete*

5.2.5. Convertirea în reprezentare triunghiulară

Reprezentarea triunghiulară, stocată în formatul **STL** (stereolitography format), este cea mai simplă reprezentare posibilă pentru exportul şi importul solidelor. În prezent, STL este exportat de către toate sistemele de proiectare orientate pe solide (*Solid Edge, Solid Works, ProEngineering, Autodesk Mechanical Desktop*), ca export generic în sistemele de analiză cu elemente finite (FEA), în cele de fabricaţie (CAM) sau de verificare.

Teoria triangularizării, folosită pentru convertirea în acest format, este foarte vastă având aplicaţii într-o multitudine de domenii, ca: *analiză cu elemente finite, trasări de hărţi, stocări optimizate a zonelor plane.*

Primele încercări, cu adevărat remarcabile, de rezovare a triangularizării suprafeţelor şi tetraedrizării volumelor au fost făcute de **Delaunay**, bazându-se pe polinoamele convexe **Voronoi** [TRI].

Se va prezenta, în continuare, un algoritm minimal de conversie a SD în reprezentare solidă triunghiulară neoptimizată.

```
NimicCDiscretSurface.ExportPlasă(SetDeCaractere strNumeDXF)
{
   CSTLExport stlout;
   Punctp1, p2, p3, p4;

   stlout.Iniţializează(strNumeDXF);
      PentruFiecare(Întreg nI = 1; nI < nX; nI++)
         PentruFiecare(Întreg nJ = 1; nJ < nY; nJ++)
         {
            p1(IndexÎnRealPtX(nI + 0), IndexÎnRealPtY(nJ + 0), [nI + 0, nJ + 0]);
            p2(IndexÎnRealPtX(nI + 0), IndexÎnRealPtY(nJ + 1), [nI + 0, nJ + 1]);
            p3(IndexÎnRealPtX(nI + 1), IndexÎnRealPtY(nJ + 0), [nI + 1, nJ + 0]);
            p4(IndexÎnRealPtX(nI + 1), IndexÎnRealPtY(nJ + 1), [nI + 1, nJ + 1]);
            stlout.PuncLa(p1, p2, p3);
            stlout.PuncLa(p2, p3, p4);
         } PentruFiecare
   stlout.Sfârşeşte();
} EndExportPlasă
```

Notă: algoritmul prezentat este cel mai simplu posibil; în realitate, autorul utilizează unul mult mai sofisticat, care reduce mărimea fişierelor STL de peste 4 ori; acest algoritm încearcă, prin diferite metode, să găsească şi să elimine punctele coplanare (fig. 2.1), după care trece la optimizarea setului de puncte. Lungimea acestui algoritm, în limbaj C++, este de peste 8000 linii cod, în comparaţie cu cel listat, de sub 20 !

Exemple de solizi, în formatul solid triunghiular **STL**, se prezintă în figurile 2.5 şi 2.6.

5.3. Formate de import - export

Formatele de import - export au o mare importanţă în automatizarea circulaţiei informaţiei într-o intreprindere. Din păcate, la ora actuală, se pare că sunt încă preferate formatele standard de nivel jos, ca *STL, CL, NC, PLT,* pentru singurul motiv că **sunt uşor de implementat**, ceea ce duce la transferul aşteptat, fără prea multe dificultăţi.

Formatele de nivel mai ridicat des utilizate sunt cele proprietare sistemelor de proiectare (CAD), ca *DXF (AutoCAD), DWG (AutoCAD), SAT (ACIS), VDA (Microstation),* deoarece rutinele de import - export sunt furnizate de către o singură companie (cea proprietară), care se ocupă de dezvoltarea şi implementarea importului şi exportului în formatul respectiv.

Formatele standard de nivel înalt, ca *IGES, STEP,* sunt mult mai bogate în forma de reprezentare, dar şi mult mai complicate, existând dificultăţi de a scrie rutine de import - export şi probabil, din această cauză, sunt şi mai puţin utilizate. Se pare, însă, că următorii ani vor aparţine acestor formate, impuse de ingineria concurentă.

În tabelele următoare se vor prezenta formatele cele mai utilizate în anul 1997, în două ţări industrializate - Marea Britanie şi SUA, pentru comunicarea între sistemele CAD şi CAM. Datele au fost procesate şi puse la dispoziţia autorului de către domnul Geofrey Taylor, director de marketing la ***Pathtrace ltd.*** Eşantionul analizat a fost de peste 100 de intreprinderi, în fiecare ţară. Datele obţinute sunt folosite pentru definirea ordinii de dezvoltare a convertoarelor, încercând să se tragă concluzii practice pertinente despre adevăratul stadiu actual al comunicaţiei între sistemele de proiectare şi fabricaţie. În special în întreprinderile mici şi mijlocii se pare că cele mai folosite metode de comunicaţie sunt încă desenele imprimate şi formatele proprietare Autodesk (DXF şi DWG). Formatul standard IGES are o pondere scăzută (sub 10 %), iar formatul STEP aproape că nu există. Într-o evoluţie ascendentă spectaculoasă este formatul SAT (format proprietar de descriere a solidelor ASCII), care are o creştere continuă de utilizare, de peste 4 % pe an.

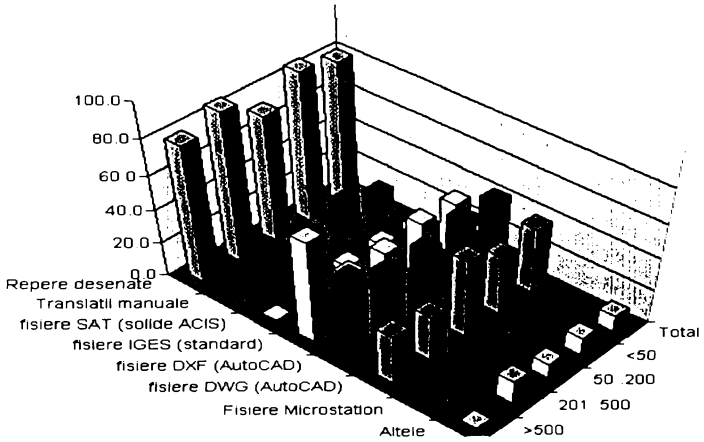
Sintetizând tendinţele de utilizare pe 1997, acestea sunt:

Metoda de transfer	Proporţia de utilizare, în [%]	Tendinţa de creştere în 1997, [%/ an]
Repere desenate	75	- 5
Translaţii manuale	13	0
Fişiere SAT (solide ACIS)	14	+4
Fişiere IGES (standard)	25	+1
Fişiere STEP (standard)	5	+1
Fişiere DXF (AutoCAD)	35	+1
Fişiere DWG (AutoCAD)	40	+2
Fişiere VDA (Microstation)	5	+1
Fişiere STL (solide faţetate)	2	+3

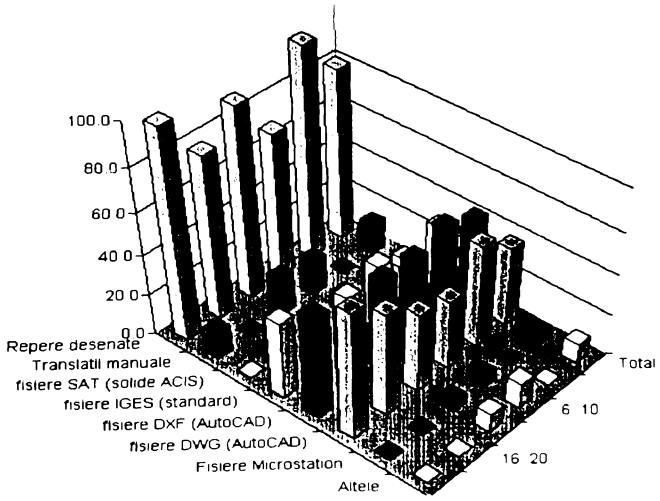
Se poate observa că, însumând coloana a doua, rezultă o valoare mai mare de 100 %, din cauză că majoritatea intreprinderilor utilizează mai mult de un format de transfer. Cu cât o intreprindere este mai mare, cu atât va avea tendinţa de a folosi schimbul electronic de date, în timp ce intreprinderile mici, care sunt reprezentate de subcontractori, schimbă încă datele în format desenat.

Marea Britanie		Număr angajaţi				Număr maşini unelete				
Despre intreprindere	**Total**	**<50**	**50.. 200**	**201.. 500**	**>500**	**1..5**	**6..10**	**10..15**	**16.. 20**	**>20**
Repere desenate	81.5	86.7	73.3	86.7	80.0	100.0	69.6	90.9	78.6	100.0
Translaţii manuale	14.6	21.7	13.3	0.0	0.0	0.0	16.1	18.2	0.0	12.5
Fişiere SAT (solide ACIS)	5.4	6.7	6.7	0.0	0.0	0.0	7.1	6.8	0.0	0.0
Fişiere IGES (standard)	30.8	31.7	26.7	26.7	60.0	0.0	32.1	27.3	28.6	37.5
Fişiere DXF (AutoCAD)	43.1	45.0	40.0	40.0	50.0	50.0	44.6	45.5	35.7	50.0
Fişiere DWG (AutoCAD)	38.5	35.0	46.7	26.7	30.0	50.0	33.9	38.6	50.0	62.5
Fişiere VDA Microstation	3.8	5.0	4.4	0.0	0.0	0.0	5.4	4.5	0.0	0.0
Altele	9.2	10.0	8.9	13.3	0.0	0.0	12.5	9.1	0.0	0.0

Formate utilizate în GB funcţie de numărul de angajaţi

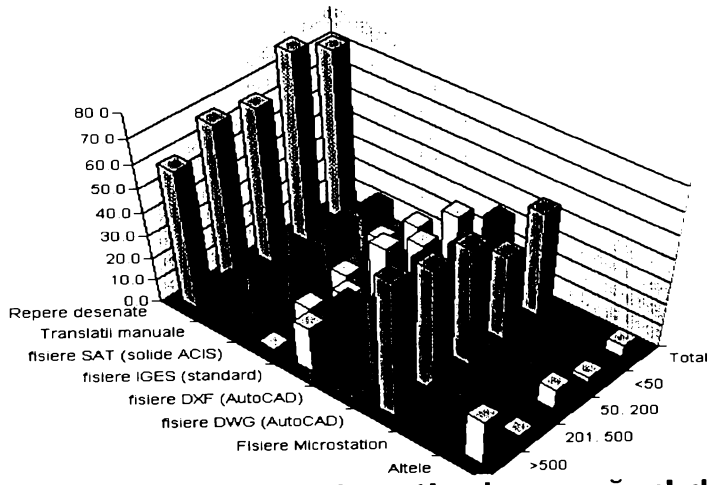


Formate utilizate în GB funcţie de numărul de MU

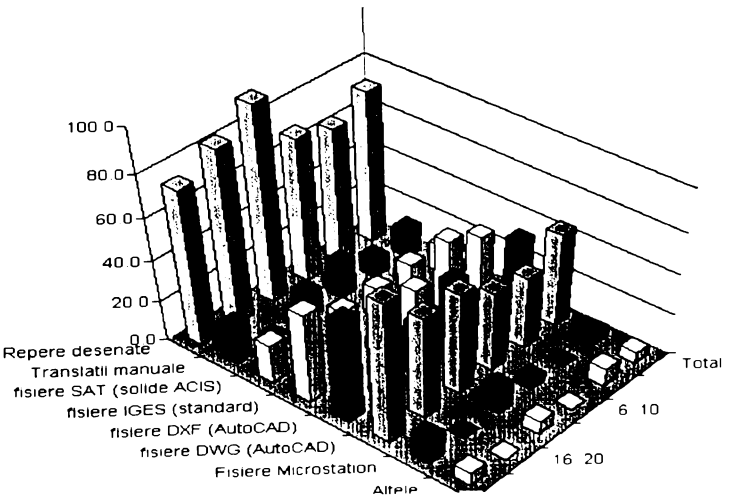


SUA		Număr angajaţi				Număr maşini unelete				
Despre intreprindere	**Total**	**<50**	**50.. 200**	**201.. 500**	**>500**	**1..5**	**6..10**	**10..15**	**16.. 20**	**>20**
Repere desenate	72.9	80.0	66.7	69.2	60.0	64.7	69.6	93.8	83.3	75.0
Translaţii manuale	13.5	15.6	12.1	7.7	20.0	8.8	17.4	12.5	0.0	18.8
Fişiere SAT (solide ACIS)	13.5	17.8	12.1	7.7	0.0	14.7	13.0	12.5	0.0	18.8
Fişiere IGES (standard)	27.1	24.4	33.3	23.1	20.0	35.3	21.7	6.3	0.0	43.8
Fişiere DXF (AutoCAD)	33.3	31.1	36.4	30.8	40.0	2.9	34.8	25.0	16.7	50.0
Fişiere DWG (AutoCAD)	45.8	37.8	51.5	53.8	60.0	35.3	39.1	50.0	50.0	68.8
Fişiere VDA Microstation	5.2	2.2	9.1	0.0	20.0	0.0	4.3	12.5	0.0	12.5
Altele	5.2	2.2	9.1	0.0	20.0	8.8	0.0	6.3	0.0	6.3

Formate utilizate în SUA funcţie de numărul de angajaţi



Formate utilizate în SUA funcţie de numărul de MU



5.3.1. Formatul DXF

Este unul dintre cele mai utilizate formate de nivel jos. A fost creat de firma **Autodesk** în anii '80 şi este larg utilizat pentru schimbarea datelor de tip curbe sau plase, cu celulă elementară patrulateră, având în 1997 o pondere de peste 20% în schimbul de informaţii dintre sistemele de proiectare şi fabricaţie. Este continuu dezvoltat de firma Autodesk şi, împreună cu formatul binar DWG, tot al firmei Autodesk, deţine jumătate din transferurile dintre sistemele CAD şi CAM.

În această lucrare se va prezenta doar partea de export a formatului, deoarece în lucrare este folosit ca export a curbelor de analiză sau a suprafeţelor discrete, în vederea vizualizărilor.

Ca o curiozitate, aproape toate reprezentările în care se utilizează librăria profesională de randare în timp real **OpenGL (c)SGI** au fost transferate utilizând formatul **DXF**. Produsul care a fost proiectat şi realizat special pentru această lucrare, în vederea vizualizării de calitate a SD, se numeşte **DSView**.

De-a lungul vremii, autorul a încercat să sintetizeze o formă cât mai simplă de export în acest format, prezentat mai pe larg în [HCF] şi [DXF].

```
NimicCDXFExport.Iniţializează(SetDeCaractere strNume)  iniţializarea obiectului
{
   fişier.Crează(strNume);
   fişier.Scrie(                                          ).  scrie antetul DXF
} EndInit

NimicCDXFExport.Sfârşeşte()  terminarea sesiunii de export
{
   fişier.Scrie(                                          ).  scrie sfârşitul DXF
   fişier.Închide();
} Sfârşeşte

NimicCDXFExport.InitCurbă(Întreg nCuloare = 15. SetDeCaractere strLayer =          )
{
  iniţializează exportul unei curbe
   fişier.Scrie(                                  . strLayer. nCuloare).
   fişier.Scrie(                                                       ).
} EndInitCurbă

NimicCDXFExport.DoneCurbă()  terminarea sesiunii de export curbă
{
   fişier.Scrie(                                  ).  scrie sfârşitul curbă
} EndDoneCurbă

NimicCDXFExport.InitPlasă(Întreg nU. nV. nCuloare = 15. SetDeCaractere strLayer =          )
{
  iniţializează exportul unei plase
   fişier.Scrie(                                  . strLayer. nCuloare).
   fişier.Scrie(                                                       ).
   fişier.Scrie(                                  . nU. nV).
} EndInitPlasă

NimicCDXFExport.DonePlasă()  terminarea sesiunii de export plasă
{
   fişier.Scrie(                                  ).  scrie sfârşitul plasă
} EndDonePlasă
```

```
NimicCDXFExport.Exportă(Punct pIn)  exportă un vertex
{
   fişier.Scrie("0\nVERTEX\n8\n0\n10\n"+pIn.rX+"\n20\n"+pIn.rY+"\n30\n"+pIn.rZ+"\n",
                pIn.rX, pIn.rY, pIn.rZ);
}  EndExportă
```

Acesta este un obiect minimal, care asigură în totalitate exportul plaselor şi curbelor 3D, via formatul **DXF**.

Într-o prezentare simplificată, filozofia formatului **DXF** este următoarea:

```
dxfin.Iniţializează("Test DXF");  iniţializarea obiectului
   dxfin.InitCurbă();  pot fi specificate eventual culoarea sau layerul
      PentruFiecare(Întreg nI = 1; nI < 10, nI++)
         dxfin.Exportă(pPunct);
   dxfin.DoneCurbă();  termină o curbă

   dxfin.InitPlasă(10, 20);  pot fi specificate eventual culoarea sau layerul
      PentruFiecare(Întreg nI = 1; nI < 10, nI++)
         PentruFiecare(Întreg nJ = 1; nJ < 20, nJ++)
         dxfin.Exportă(pPunct);
   dxfin.DonePlasă();  termină o curbă

dxfin.Sfârşeşte();  închide fişierul;
```

Deci, fişierul, ca şi orice altă entitate care se exportă, trebuie iniţializat şi închis prin secvenţa Init'() ... Done'() .

De menţionat că formatul **DXF** suportă şi o versiune binară, cu extensia "**.DXB**", care nu face însă obiectul acestei prezentări, fiind prea criptică.

În continuare se pot observa două exemple de export **DXF** care folosesc acest obiect.

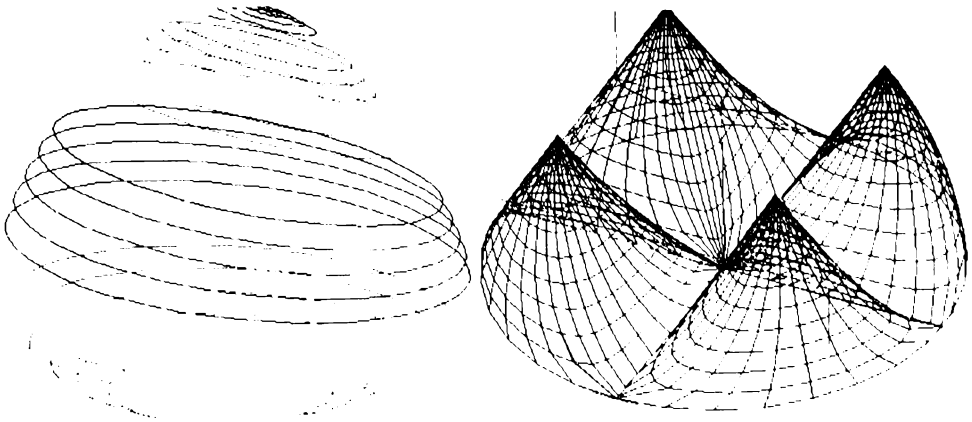


Figura 5.6 *Exemple de export şi listing în format DXF, create cu TechnoFunction V1.0 (două funcţii excentrice Şelariu)*

```
0                0                62               20               70
SECTION          POLYLINE         13               10.000000        16
2                8                10               30               66
ENTITIES         0                270.000000       450.000000       1
```

71	VERTEX	20	SEQEND	0
19	8	0.000000	8	EOF
72	0	30	0	
91	10	0.000000	0	
0	2.000000	0	ENDSEC	

5.3.2. Formatul STL

Formatul **STL** s-a născut cu mai mult de 10 ani în urmă, ca un format de import, fiind folosit pentru maşinile speciale de prototipizare rapidă, prin polimerizare cu ajutorul laserului.

De-a lungul vremii, acesta a fost utilizat destul de rar, deoarece este total impropriu şi distructiv reprezentărilor nesolide **B-Spline**, condiţia de bază a descrierii unui solid fiind aceea că o latură a unui triunghi trebuie să corespundă la numai două triunghiuri, condiţie destul de restrictivă în conversia altor tipuri de dată.

Odată cu impunerea modelatoarelor solide, formatul **STL** a renăscut, fiind unicul format de nivel jos comun tuturor sistemelor de proiectare, pentru transferul **solidelor**. Astfel, el este folosit la ora actuală de următoarele tipuri de aplicaţii:

- **sisteme de proiectare** (în general *import - export*);
- **sisteme de fabricaţie** (*import - export* şi procesarea suprafeţelor corecţie de sculă);
- **sisteme de analiză cu elemente finite** (*import* şi utilizare directă a reprezentării);
- **vizualizoare realistice şi cadru de sârmă** (*import*);
- **sisteme de verificare a corectitudinii fişierului NC** (*import*);

Concluzia este că a început să reprezinte un limbaj comun şi simplu de comunicare.

Se va prezenta, în continuare, forma de fişier ASCII încapsulată într-o clasă numită **CSTLOut.**

```
NimicCSTLExport.Iniţializează(SetDeCaractere strNume) // iniţializarea obiectului
{
   fişier.Crează(strNume);
   fişier.Scrie("solid SID Export");
} Iniţializează

NimicCSTLExport.Sfârşeşte()  terminarea sesiunii de export
{
   fişier.Scrie( "endsolid SID" );
   fişier.Închide();
} Sfârşeşte

NimicCSTLExport.Exportă(Punct p1, p2, p3)
{
fişier.Scrie(                                    );
fişier.Scrie(                  );
fişier.Scrie(                    , p1.rX, p1.rY, p1.rZ);
fişier.Scrie(                    , p2.rX, p2.rY, p2.rZ);
fişier.Scrie(                    , p3.rX, p3.rY, p3.rZ);
fişier.Scrie(                );
fişier.Scrie(              );
} Exportă
```

Metoda de utilizare a formatului **STL** este următoarea:

```
CSTLExport stlout;
stlout.Iniţializează("Test STL");
   for (Întreg nI = 1; nI<= nNrTriunghiuri; nI++)
      stlout.Export(p1, p2, p3);
stlout.Sfârşeşte();
```

Se poate observa că exportul este ceva mai simplu, comparabil cu cel de la formatul **DXF**, deoarece **STL** nu suportă decât o singură entitate, cea de faţetă trunghiulară.
De menţionat că formatul **STL** suportă şi o versiune binară, cu aceeaşi extensie, care nu face însă obiectul acestei prezentări.
În continuare se arată două imagini în format **STL**, una în reprezentare cadru de sârmă, alta render plan, şi un listing de fişier **STL**.

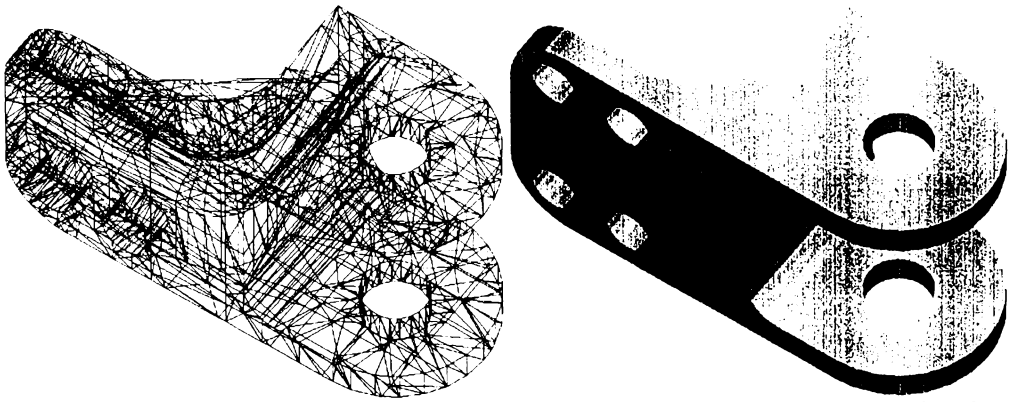


Figura 5.7 *Reprezentare cadru de sârmă şi solidă, importată ca fişier STL, pentru verificare solidă a frezării capturate din NCVerify.*

```
solid SD Export
   facet normal  0.0 0.0 0.0
      outer loop
         vertex 0.038 0.038 0.045
         vertex -0.007 0.0384 0.045
         vertex -0.007 -0.03 0.045000
      endloop
   endfacet
   facet normal  0.0 0.0 0.0
      outer loop
         vertex -0.00 -0.036 0.040
         vertex -0.007 -0.03 0.045
         vertex -0.007 0.038 0.045
      endloop
   endfacet
endsolid
```

5.3.3. Formatul CL

Formatul **CL** este un format vectorial, deosebit de important în transferurile de trasee de sculă, deoarece în el pot fi înmagazinate date despre *sculă, compensaţii, informaţii geometrice de mişcare şi informaţii tehnologice* despre avansuri şi turaţie.

Marele avantaj al acestui format este acela că este **generic**, înmagazinând toate informaţiile necesare generării fişierului **NC**, pentru un anumit reper, independent de echipament. Deci, **CL** este ieşirea comună a tuturor sistemelor destinate fabricaţiei, fiind un dialect al limbajului **APT**.

Fiind un format relativ complex, în această lucrare nu va fi folosit. Vor fi generate direct fişiere **NC**, care sunt destul de asemănătoare ca sintaxă.

Pentru exemplificare se vor prezenta câteva figuri şi o porţiune de listing în format **CL**.

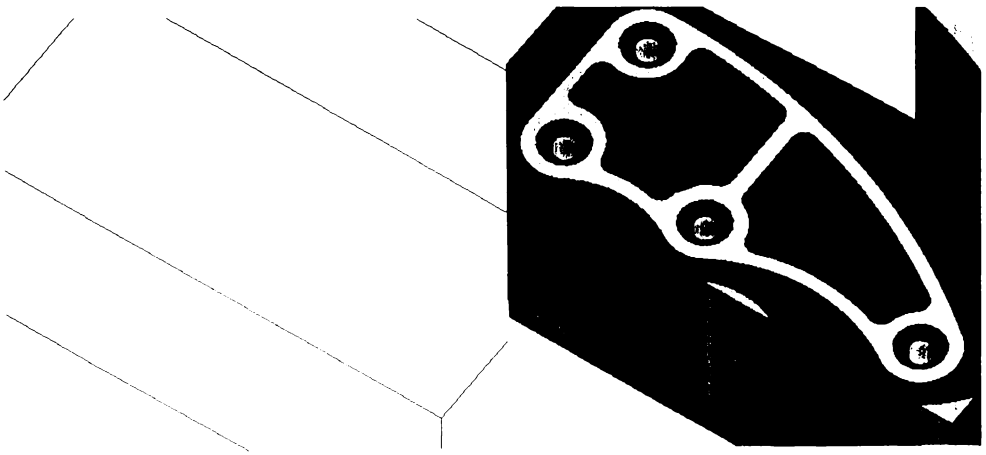


Figura 5.8 *Exemple de fişier CL importat şi simulat cu NCVerify.*

```
1 CBOX -1.8744 -3.3565 -0.1330 9.4887 4.1458 1.0000
2 NSIDES 16
2 UNITS/ INCHES
3 PARTNO shador
4 PPRINT PROGRAM FOR PART NAME shado3Wr
5 PPRINT PROGRAM FOR MFG PLAN NAME shado3Wr
6 PPRINT PROGRAM TIME OF CREATION Apr 23 1994 15:05:40
7 MULTAX/ON
9 PPRINT SETUP: Setup10
11 PPRINT Drill.CBHoleGroup.1 TOOLPATH
12 CUTTER/ 0.500000, 0.000000, 0.250000, 0.000000, 31.000000, 0.000000, 2.0000
13 LOADTL/ 1
14 SPINDL/ 474, RPM, CLW
15 RAPID
.....
862 GOTO/ 0.0, 2.5, 1.1, 0.0, 0.0, 1.0
863 COOLNT/OFF
864 FINI
```

5.3.4. Formatul NC

Prin convertirea formatul generic **CL**, pentru un echipament specific, acesta îşi pierde generalitatea şi devine un fişier **NC** specific pentru un echipament **NC** dat. Teoretic, limbajul **NC** este un dialect simplificat de **BASIC**, standardizat. Din păcate, fiecare echipament respectă, mai mult sau mai puţin, acest standard.

În general, mişcările *G0, G1, G2, G3* şi regiştrii *X, Y, Z, I, J, K, M, F, S* se respectă de toate echipamentele.

În această lucrare vor fi generate fişiere **NC** direct din familii de curbe. Pentru acest deziderat a fost creată o clasă minimală care exportă fişier standard **NC** într-un fişier, utilizând doar *N, G0, G1, X, Y, Z* pentru a asigura o implemetare simplă şi o portabilitate cvasitotală.

```
NimicCNCExport.Iniţializează(SetDeCaractere strNume) //crează un fisier NC
{
   nLinie = 1;
   fişier.Deschide(strNume);
   fişier.Scrie("SD generare cod NC");
   fişier.Scrie("%1");
} //Init

NimicCNCExport.Sfârşeşte() //închide fişierul NC
{
   pLast.Z += 10.0;
   Exportă(pLastZ);
   fişier.Scrie("%1");
   fişier.Închide();
} //Sfârşeşte

NimicCNCExport.Exportă(Punct p, Boolean bRapid = FALS)
{
   pLast = p;
   Dacă (bRapid)
      fişier.Scrie("N%dG0X%dY%dZ%d", nLinie++, p.rX, p.rY, p.rZ);
   Altfel fişier.Scrie("N%dG1X%dY%dZ%d", nLinie++, p.rX, p.rY, p.rZ);
} //Exportă
```

Această clasă este utilizată astfel:

```
CNCExport ncout;

ncout.Iniţializează("test.NC");
   PentruFiecare(nI = 1; nI <= cExportă.nNr; nI++)
   ncout.Exportă(cExportă[nI])
ncout.Sfârşeşte();
```

Notă: În realitate, autorul utilizează la generarea fişierului **NC** o librărie creată special pentru toate produsele din seria **Techno***; aceasta a fost numită **Generic Numeric Control Post Procesor V2.0 (GNCPP)** şi a fost dezvoltată pe parcurs, pentru a rezolva problema generării.

Librăria **GNCPP** este un produs *freeware*, care poate fi găsită în multe website-uri cu produse *shareware* şi *freeware*, având până în prezent peste o sută de utilizatori cunoscuţi.

Se vor prezenta câteva tehnologii care sunt utilizate în **GNCPP V2.0** (prezentarea în pseudocod depăşeşte spaţiul lucrării, lungimea lui fiind de peste 20.000 linii cod):

Postprocesorul este un modul unitar, şi se ocupă de un singur lucru: *modul în care se generează fişierul* ***NC***. Este realizat pentru a genera fişiere standard **ANSI-DIN** sau are posibilitatea programării lui pentru diferite dialecte nestandardizate.

Are peste 200 de variabile de configurare, care sunt foarte bine documentate şi uşor de configurat, creându-se astfel familii de postprocesoare nestandardizate, reale sau virtuale (pentru diferite analize, sau vizualizări).

Este folosit în mod unitar de toate modulele care generează fişiere **NC** (**TehnoCAM, TehnoBulge, Tehno2D, TehnoMesh**). În România, fişierele generate de el rulează pe ECN foarte variate, ca: **CNC600, Fanuc, Sinumeric, Heidenheim, NUM, NUMEROM, Elerofil, DEM** etc.

Poate genera (la cerere) fişiere **DXF**, în paralel cu cele **CNC**, pentru verificări pe alte sisteme de proiectare. Calculează automat lungimea pe care o execută capul în timpul frezării, în mişcare de lucru şi în mişcare rapidă, şi timpul necesar frezării unei suprafeţe, creând automat şi un fişier cu datele tehnologice.

Programele **NC** pot fi fragmentate automat, în funcţie de memoria **MUCN** sau numărul de linii suportate de aceasta.

Converteşte datele geometrice, din 3 în 2,5 axe, punând astfel în valoare parcul de maşini unelte din ţară şi scurtând cu **30 %** fişierele **NC**.

Posedă un mod **euristic** de rezolvare a *problemei comis voiajorului* (TSP), foarte rapid şi performant, eliminând peste **90 %** din mişcările de avans rapid, scurtând semnificativ unele tipuri de fişiere **NC**. Acest algoritm este prezent doar în seria **TechnoPack**, fiind creat în premieră la ***BillaSoft*** **[DM01..DM11]**.

Conţine modul **Adaptiv**, care analizează frezarea şi (în urma analizei) schimbă valoarea avansului automat în timpul rulării fişierului **NC**, încărcând uniform cu solicitări freza şi scula; în funcţie de geometria acesteia şi modul de frezare (longitudinal, radial, mixt) încetineşte mişcarea de avans şi corectează spaţial mişcarea capului, în raport cu uzura sculei (optimizare pt. timp).

Modul **Stealth** (ascuns) stă tot timpul în acţiune (cât postprocesorul este în funcţiune) şi analizează scrierea fişierelor **NC**, hotărând dacă informaţiile care se doresc a fi scrise în fişierul **NC** sunt importante sau dacă se pot deduce din alte linii de informaţii, creând astfel fişiere **NC** cât mai scurte. Fişierele generate cu **Stealth** activ sunt cu **60...90 %** mai scurte decât cele normale, nepierzându-se informaţii geometrice utile (optimizare pentru lungime).

Se prezintă un listing parţial al variabilelor care stau la baza configurării postprocesorului:

```
[GenericNCPostProcessor V2.0 ACTIVE variables ]
[WARNING: IF YOU PERFORM MODIFICATIONS DO NOT KILL INDENTATION ! ]
[Variable = Current:][Possible ]<Def.> Comment ]
[=============================================================================]
```

[CNC.Process]
CNCExport = **Y** ;C[**Y_**]< **Y**> *Process CNCFile else not*
DXFExport = N ;C[**Y_**]< **N**> *Process DXFFile else not*
RepExport = Y ;C[**Y_**]< **Y**> *Process ReportFile else not*
NameDigitNr = 3 ;I[0..5]< 3> *Nr. of digits in CNC Name*
LongName = Y ;C[**Y_**]< **Y**> *If N cut in name CNC type inf.*
Extension =CNC ;S[**Y_**]< **CNC**> *Name of extension of CNCFile*

[CNC.ConstraÎntregs]
OptFilter = 0.200;R[0.01..0.5]< 0.2> *Value of optimizing filter*
BreakKilo = Y ;C[**Y_**]< **Y**> *Break CNCFile after MaxKilo or MaxLine*
MaxKilo = 320.000;R[1..1000]< 32> *Number of Kilos in each CNC file*
MaxLine = 990 ;I[10..10000]< 990> *Number of Lines in each CNC file*
ExportValue = 2.000;R[0..1000]< 2> *ExportValue of z in case of break CNCFile*
AutoBreak = 10.000;R[0..10]< 10> *Break value in CNC file < ExportValue*
CondensedExport= Y ;C[**Y_**]< **Y**> *Y purge trivial data in CNC File*
IndentedExport = N ;C[**Y_**]< **N**> *Y indent for good visibility*
Enable2Half = Y ;C[**Y_**]< **Y**> *Enable 2.5 axis generation*
Frontal = Y ;C[**Y_**]< **Y**> *Enable frontal milling*
Relative = N ;C[**Y_**]< **N**> *Enable relative coordinate*
RelativeG2G3= N ;C[**Y_**]< **N**> *Enable relative coordinate for I, J, K*
DecimalNr = 3 ;I[0..4]< 3> *Nr. of decimals in CNC File*
ForceDecimal= N ;C[**Y_**]< **N**> *Force decimal in all real variables*
G2G3 = Y ;C[**Y_**]< **Y**> *Enable arc interpolation*
G2G3Radius = 2.000;R[-1e3..1e3]< 2> *If G2G3 N then value to discreet arc*

Un exemplu de fişier **NC** generat prin **GNCPP** de către un client (**TechnoCAM V2.0**):

-Generic NC V2.00 (c)1990-96 Dan MICSA---
==========

- **Tool** : CF10-10;
- **Name** : QQ [1];
- **ProcessBy** : Select.RECTANGLE.**CNCExport**;
- **RegUser** : Dan MICSA;
- **RegCompany** : ?=>! BillaSoft;
- **SerialNr** : ROTM0001;
- **ProcessDate**: 13/01/97;
- **ProcessHour**: 10:28:46;
- **Processor** : TechnoCAD V2.0;
- **Cfg.Name20N**: X:\W\EXE\CNC.20N;

- **Bounds.XMin**: 5.0;*XMax*: 21.0 [*mm*];
- **Bounds.YMin**: -7.3;*YMax*: 7.5 [*mm*];
- **Bounds.ZMin**: -5.0;*ZMax*: 45.0 [*mm*];
- **SpeedLength**: 10.000[*mm*]; 5.934[*%*];
- **Work Length**: 158.523[*mm*]; 94.066[*%*];
- **TotalLength**: 168.523[*mm*];
- **SpeedTime** : 0[*h*] 0.010[*mm*]; 0.627[*%*];
- **Work Time** : 0[*h*] 1.585[*mm*]; 99.373[*%*];
- **TotalTime** : 0[*h*] 1.595[*mm*];
- **SpeedCost** : 0.001[*S*]; 0.314[*%*];
- **Work Cost** : 0.264[*S*]; 99.686[*%*];
- **TotalCost** : 0.265[*S*];
- **Nr.of Lines**: 66
- **Nr.of Bytes**: 1342
- **Optimizing** : -2[*%*];

%1G71N5P0.100.000
N6S100F=P0
N7G0X21Y7.5Z45

```
N8G1Z-5
N9X20
N10X19Z-2.2
N11X18Z2.6
N12X17Z4.8
N13X16Z5.9
N14X15
. . .
N70X20.5Y7.5
N71Z5
N72G0Z15
N73G0Z17
N9999%1G71
```

Câteva figuri în care s-a folosit **GNCPP** ca generator şi analizor de tehnologie.

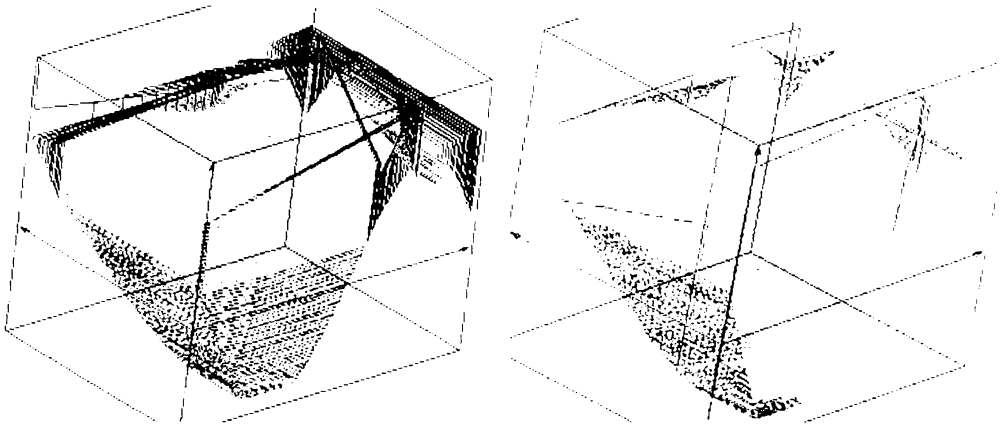


Figura 5.10 *Export fişier NC cu modul FHTSP activat, exemplificarea reducerii mişcărilor în avans rapid cu peste 95%.*

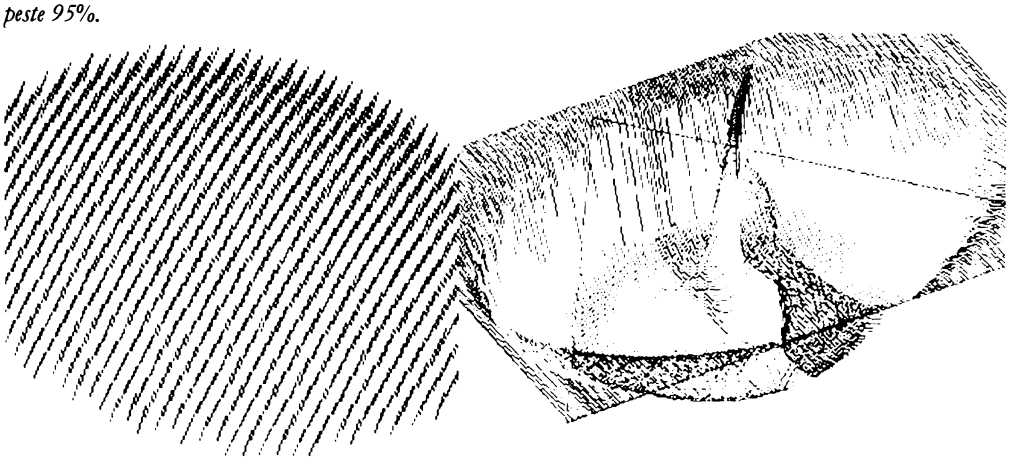


Figura 5.10 *Alte exemplificări ale utilizării TechnoCAM şi GNCPP*
a) reţea de difracţie, b) reper complex cu pas variabil, pentru frezare la rugozitate constantă

5.4. Concepte introduse

```
Clasa SuprafaţăDiscretă DerivatăDin BazăDiscretă
{
 //Concepte introduse în celelalte capitole
    Nimic CreazăCurbe(FamiliaDeCurbe fcOut, Real rPentruZ);
    Nimic CreazăToateCurbele(FamiliaDeCurbe fcOut, Real rZStep = 1.0, Boolean bAuto = ADEVĂRAT, Real
    rZStart, rZEnd);
    Nimic Proiectează Curbă(Curbă cProiectează);
    Nimic Proiectează Curbe(FamiliaDeCurbe fcProiectează);
    Nimic Proiectează Curbă(Curbă cProiectează, Real rPas = 0.1); //cu pas constant
    Nimic Proiectează Curbe(FamiliaDeCurbe fcProiectează, Real rPas = 0.1); //pas ct.
    Nimic ExportDXF(Întreg nCuloare = 15, SetDeCaractere strLayer = "SD ieşire")
} //SuprafaţăDiscretă

Clasa Fişier //obiect necesar fisierelor de export - import
{
    Nimic Deschide(SetDeCaractere strNume); //deschide sau creează un fişier
    Nimic Închide(); //închide un fişier
    Nimic Scrie(SetDeCaractere strOut); //scrie un SetDeCaractere în fisier
    Nimic Citeşte(SetDeCaractere strIn); //citeşte un SetDeCaractere din fisier
} //Fişier

Clasa DXFExport //o rudimentare a formatului de export DXF care suportă curbe şi plase
{
    Fişier fişier; //fişierul de export

    Nimic Iniţializează(SetDeCaractere strNume);
    Nimic Sfârşeşte();

    Nimic InitCurbă(Întreg nCuloare = 15, SetDeCaractere strLayer = "SD ieşire");
    Nimic DoneCurbă();

    Nimic InitPlasă(Întreg nU, nV, nCuloare = 15, SetDeCaractere strLayer = "SD ieşire");
    Nimic DonePlasă();

    Nimic Exportă(Punct pIn);
} //DXFExport

Clasa STLExport //o rudimentare a formatului de export STL care suportă doar ASCII Export
{
    Fişier fişier; //fişierul de export

    Nimic Iniţializează(SetDeCaractere strNume);
    Nimic Sfârşeşte();

    Nimic Exportă(Punct p1, p2, p3);
} STLExport

Clasa NCExport //o rudimentare a formatului de export NC ce suportă numai informaţii geometrice de tip segment
{
    Fişier fişier; //fişierul de export

    Nimic Iniţializează(SetDeCaractere strNume);
    Nimic Sfârşeşte();

    Nimic Exportă(Punct p, Boolean bRapid);
} NCExport
```

5.5. Concluzii

În acest capitol au fost introduse metode de creare de curbe, proiecţie, offset inteligent şi export în formate simple ASCII, ca: ***DXF, STL, NC.***

Acest capitol, al **metodelor de conversie şi export**, este de mare importanţă în utilizarea suprafeţelor discrete, virtual, în orice sistem de proiectare şi fabricaţie. Ideea prezentării lui a fost aceea de a îmbina prezentarea riguroasă, însoţită de expunerea algoritmilor în pseudocod, cu exemplificarea fiecărui concept introdus, cu ajutorul exemplelor şi a listingului potenţial, realizat de către clasele expuse.

A fost creat un set nou de clase specifice fiecărui tip de export în parte: ***Fişier, CDXFOut, CSTLOut, CNCOut***. Aceste clase au fost implementate folosind o metodă unificată de prezentare, încercând să se ascundă detaliile fiecărui format în parte.

Au fost create **metode** noi, destinate conversiei şi exportului, metode care dau o utilizabilitate SD, legându-le de alte sisteme de proiectare, ca aparate matematice auxiliare de analiză sau conversie în format NC.

S-a prezentat, în premieră, un algoritm de conversie în curbe de nivel foarte fin, care permite conversia SD corecţie de sculă în format NC, şi s-au prezentat aplicaţiile lui, potenţial nelimitate, în detectarea formelor de analiză şi vectorizarea fotografiilor.

S-a expus şi exemplificat **GNCPP**, un postprocesor generic de format NC, probabil unul dintre cele mai elaborate postprocesoare la ora actuală, realizat ca o librărie dinamică, foarte complexă, care are scopul de a genera fişier **NC** specific, şi virtual, pentru orice echipament, optimizat pentru lungime şi timp de rulare.

6. Metode de analiză şi optimizare

6.1. Introducere

În acest capitol, generic intitulat "**Metode de analiză şi optimizare**", se vor cuprinde câteva dintre cele mai importante aspecte legate de analiza SD şi generarea optimizată a codului NC pentru fabricarea suprafeţelor discrete pe MUCN, precum şi câteva tehnici de *verificare, simulare şi vizualizare.* Cum toate acestea sunt tehnici şi metode de ***analiză***, ele îşi vor găsi locul în acest capitol.

Principalul scop al capitolelor prezentate până în prezent a fost acela de a proiecta şi expune un set consistent de obiecte, (relativ) bogat în metode de *modelare, import, export, conversii şi proiecţii de curbe*, în vederea analizării şi a generării optimizate de cod NC.

Metodele expuse vor fi grupate în funcţie de similitudinea algoritmilor în:

- ⇨ *metode de analiză a SD;*
- ⇨ *metode de analiză a curbelor;*
- ⇨ *metode de analiză mixte;*
- ⇨ *optimizări posibile ale traseelor de scule;*
- ⇨ *calculul reţelei de difracţie pentru elementele optice;*
- ⇨ *metode de vizualizare a curbelor;*

Se vor prezenta, în premieră, metode noi de analiză: *calculul zonelor plane, calculul zonelor critice la frezarea de secţiuni paralele în planul XY, calculul materialului nefrezabil, calculul curbelor de egală rugozitate, metode pseudoadaptive de variere a avansului şi corecţiei de uzură în timp real, minimizarea mişcărilor în avans rapid, spiralele lui Billator, optimizarea traseelor echidistante prin dublarea sau triplarea locală, rezolvarea reţelelor de difracţie.*

Pe lângă prezentarea contribuţiilor autorul, se vor expune şi câteva metode clasice de generare de cod, considerându-se ca elemente de noutate metodele de generare ale acestuia (*curbe echidistante în XY şi Z*), utilizând SD în acest domeniu.

Vor fi exemplificate şi alte concepte, cum ar fi *interpolările superioare, eliminarea punctelor de inflexiune, eliminarea punctelor coliniare*, doar cu scopul secundar de a da consistenţă şi calitate unei eventuale generări de cod NC.

6.2. Concepte introductive

Metodele de **analiză** şi **optimizare** expuse în acest capitol se împart în trei mari categorii, care au, la rândul lor, alte subsecţiuni. Acestea sunt:

- ***metode care operează pe suprafaţa discretă. Utilizează o SD intermediară numită SDCST (suprafaţă discretă corecţie de sculă transformată), creată, în general, din SDCS (suprafaţă discretă corecţie de sculă), care este secţionată, convertită în curbe echipotenţiale, care sunt proiectate înapoi pe SDCS. Aceste transformări intermediare sunt cele necesare pentru:***
-
  - *calculul secţiunilor paralele cu Z constant;*
  - *calculul secţiunilor paralele în planul XY;*
  - *calculul de detecţie a zonelor plane (flat land detection);*
  - *calculul zonelor critice la frezarea planului XY (steep walls detection);*
  - *calculul curbelor de egală rugozitate;*
  - *calculul materialului nefrezabil;*
- *metode care sunt efectuate după crearea, conversia şi proiectarea curbelor pe SDCS, analizând şi optimizând setul de curbe creat prin acest proces, în vederea convertirii lor în fişier NC optimizat (acestea, în general, au fost realizate într-un produs separat, GNCPP, care nu are nimic comun cu SD). Acestea sunt:*
  - *rejecţia punctelor coliniare;*
  - *minimizarea mişcărilor în avans rapid;*
  - *modul pseodoadaptiv fără DS;*
  - *eliminarea punctelor de inflexiune;*
- *metode mixte, care implică modificarea curbelor în acord cu SD, care stochează stadiul actual de frezare al semifabricatului:*
  - *spiralele lui Billator;*
  - *cota Z de securitate minimă;*
  - *modul pseudoadaptiv cu DS;*
  - *detecţia interferenţelor.*

6.3. Metode de analiză a SD

Metodele de transformare a suprafeţelor discrete sunt de o mare importanţă în metodele de analiză, optimizare, conversie în curbe şi apoi în generarea fişierului NC.

Se menţionează, încă o dată, că ***toate*** metodele care **optimizează** generarea codului NC, bazându-se pe **transformarea** suprafeţelor discrete, analizează **SDCS** şi nu suprafaţa reală (**SDR**). Toate metodele expuse au, în general, aceeaşi derulare, care poate fi sintetizată astfel:

- ⇨ *se crează **SD** (import, modelare, combinări etc);*
- ⇨ *se alege o sculă ISO (librărie electronică, magazia intreprinderii);*
- ⇨ *se calculează **SD** înfăşurătoarea statică **SDCS** (pag. 65);*
- ⇨ *se calculează şi se crează transformarea acesteia în **SDCST,***
- ⇨ *se secţionează cu algoritmul de conversie în curbe de nivel **SDCST** (pag. 80);*
- ⇨ *Apar două căi posibile de utilizare a curbelor:*
  - ⇨ *Utilizarea ca şi curbe care restricţionează generarea fişierului (fig. 5.2)*
  - ⇨ *Utilizarea ca şi fişier NC (fig. 5.8, 5.9)*
    - ⇨ *se proiectează înapoi pe SDCS (pag. 84);*
    - ⇨ *se optimizează curbele generate;*
    - ⇨ *se exportă curbele proiectate (pag. 96);*

În continuare sunt prezentate câteva transformări intermediare ale SD. Ele reprezintă stadiul actual de studiu şi analiză în domeniul optimizării fabricaţiei suprafeţelor discrete, şi sunt doar un modest început, căruia autorul i-a dedicat ultimii şapte ani. Se consideră că potenţialul acestei metodologii de generare a fişierelor NC, sau a altor analize, utilizând transformările intermediare, este nelimitat şi se redau, în cele ce urmează, doar câteva, utilizate şi testate de autor, lăsând spaţiu de cercetare în acest domeniu.

Prezentarea se va face într-o ordine graduală a complexităţii algoritmului de transformare.

6.3.1. Calculul secţiunilor paralele cu Z constant

Calculul secţiunilor paralele cu Z constant reprezintă cea mai simplă transformare a SDCS în vederea convertirii ei în fişier NC. Aceasta este transformarea vidă, deoarece algoritmul de conversie asigură convertirea în curbe de nivel direct, deci SDCST = SDCS !

Rezumând în pseudocod, se poate scrie:

```
NimicSuprafaţăDiscretă.TransfDetectPtZ(SuprafaţăDiscretă sdAnaliză)
{
   sdAceasta = sdAnaliză;
} TransfDetectPtZ

NimicSuprafaţăDiscretă.DetectăCurbeleZ(
        FamiliaDeCurbe fcOut, Real rStep = 1.0)
{
```

```
    SuprafaţăDiscretă sdTransformare;  pregăteşte suprafata Discretă a transformăru
    sdTransformare = sdAceasta;  o initializează
    sdTransformare.TransfDetectPtZ(sdAceasta);  o calculează
    CreazăToateCurbele(fcOut);  exportă curbele
} DetectăCurbeleZ
```

Exemplificarea conceptelor introduse, în figura următoare (cu roşu sunt reprezentate secţiunile echipotenţiale).

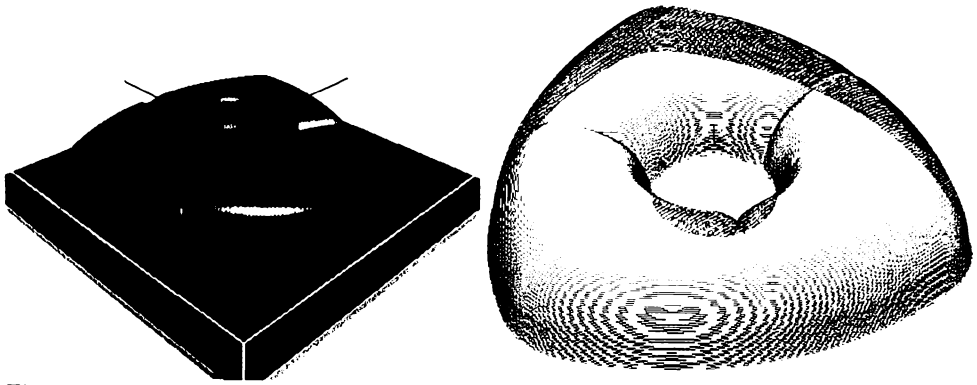


Figura 6.1 *Convertirea în curbe de nivel. Curbele echipotenţiale (în b) sunt mai bombate, din cauza aplicării algoritmilor pe SDCS. Curbele de nivel sunt cele cu culoare închisă (roşu)*

6.3.2. Calculul secţiunilor paralele în planul XY

O altă transformare simplă este cea a calculului secţiunilor paralele în planul XY. Aceasta este probabil cea mai neuzuală transformare posibilă. Ea este prezentată doar pentru a arăta gradul de generalitate al algoritmului de conversie în curbe echipotenţiale (**ACCE**).

Scopul este de a demonstra că pentru orice problemă în care trebuie calculată o familie de curbe care sunt caracterizate printr-un parametru constant (în acest caz o secţionare cu curbe paralele în planul XY), se poate găsi o tehnică de transformare care reduce problema la conversie, utilizând ACCE.

Se va prezenta în continuare o metodă care încapsulează această transformare:

```
NimicSuprafaţăDiscretă.TransfDetecteazăPentruPlanulXY(
            SuprafaţăDiscretă sdAnaliză, Real rUnghi = 45.0)
{
   PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
      PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
         PuneLa(nI, nJ, IndexÎnRealPtX(nI)*cos(rUnghi) + IndexÎnRealPtY(nJ)*sin(rUnghi));  initializare cu dist
} TransfDetecteazăPentruPlanulXY

NimicSuprafaţăDiscretă.DetecteazăCurbeleÎnXY(
```

```
        FamiliaDeCurbe fcOut, Real rStep = 1.0, rUnghi = 45.0)
{
   SuprafaţăDiscretă sdTransformare;  pregăteşte suprafaţa Discretă a transformării
   sdTransformare = sdAceasta;  o iniţializează
   sdTransformare.TransfDetectcazăPentruPlanulXY(sdAceasta, rUnghi);  o calculează
   CreazăToateCurbele(fcOut, rZStep);
} Detectează Curbele În XY
```

Cele expuse pot fi cuprinse în figurile următoare.

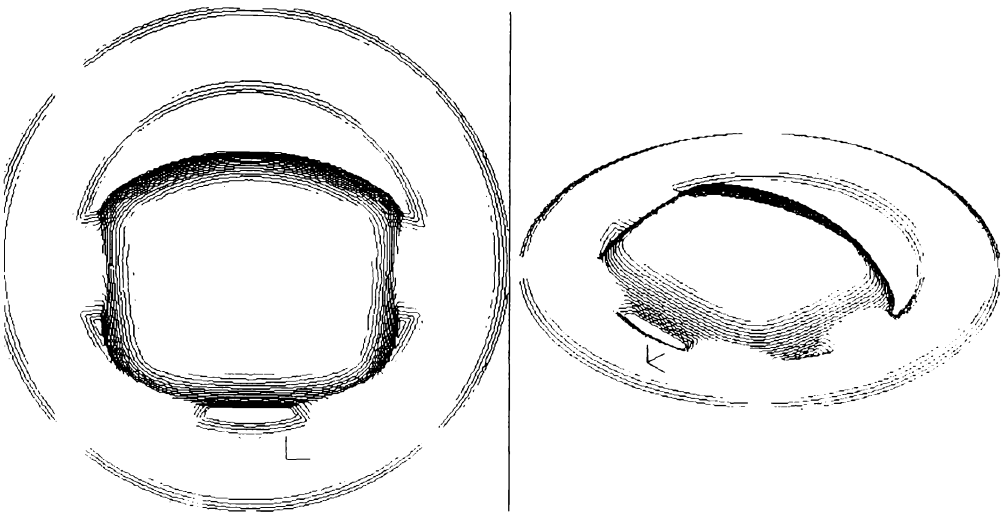


Figura 6.2 *Convertirea în curbe paralele în planul XY (zonele deschise la culoare - crem)*

6.3.3. Calculul de detecţie a zonelor plane

Detecţia zonelor plane este de o deosebită importanţă în împărţirea SD în două porţiuni, una specifică frezării cu Z constant şi alta (cea a zonelor plane) cu cicluri specifice frezării cu drepte paralele în planul XY.

Cele două tipuri de frezare sunt, la ora actuală, cele mai utilizate metode de frezare. Avantajele constau în faptul că:

- ⇨ ***traseele sculei pot fi găsite relativ uşor,*** *condiţiile de intersectare a SDCS cu plane paralele sunt mai uşoare, comparativ cu alte metode (**Notă:** cea mai simplă şi naturală metodă este cea echiparametrică, în lungul parametrului U sau V al suprafeţei);*
- ⇨ ***fişierele NC sunt mai scurte cu 30%,*** *deoarece nu conţin decât mişcări în două axe, în fiecare moment;*
- ⇨ ***se pot folosi interpolările circulare ale ECN,*** *rezultă calitate foarte bună a suprafeţei şi fişier NC mai scurt.*

⇨ ***pot fi folosite de toate echipamentele care sunt limitate de 2.5 axe; la ora actuală, peste 90% din echipamentele din ţară nu suportă simultan mai mult de 2 axe*** *(considerente strategice de realizare de suprafeţe complexe, din perioada războiului rece).*

Transformarea aplicată SDCS este aceea de a o converti în valoarea absolută a unghiului de înclinare a normalei faţă de planul XY.

Rezumarea în pseudocod este :

```
NimicSuprafaţăDiscretă.TransfDetecteazăSuprafaţaPlană(SuprafaţăDiscretă sdAnaliză)
{
   PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
      PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
         PuneLa(nI, nJ, sdAnaliză.laDeLaNormală(nI, nJ));
} //TransfDetecteazăSuprafaţaPlană


NimicSuprafaţăDiscretă.DetecteazăSuprafaţaPlană(
            FamiliaDeCurbe fcOut, Real rPentruZ = 45.0)
{
   SuprafaţăDiscretă sdTransformare; //pregăteşte suprafaţa discretă a transformării
   sdTransformare = sdAceasta; //o iniţializează
   sdTransformare.TransfDetecteazăSuprafaţaPlană(sdAceasta); //o calculează
   CreazăCurbe(fcOut, rPentruZ);
} //DetecteazăSuprafaţaPlană
```

Notă: metoda laDeLaNormală() nu a fost discutată. Ea returnează înclinaţia normalei faţă de orizontală, în punctul respectiv.

Figurile 6.1 şi 6.2 reprezintă două exemple de cicluri combinate echipotenţiale echidistante în planul XY.

În figurile următoare sunt prezentate, pentru două repere, cum arată SDCST care stochează variaţia normalei şi evoluţia în spaţiu a acestor curbe echinormale, după ce au fost proiectate pe SDCS.

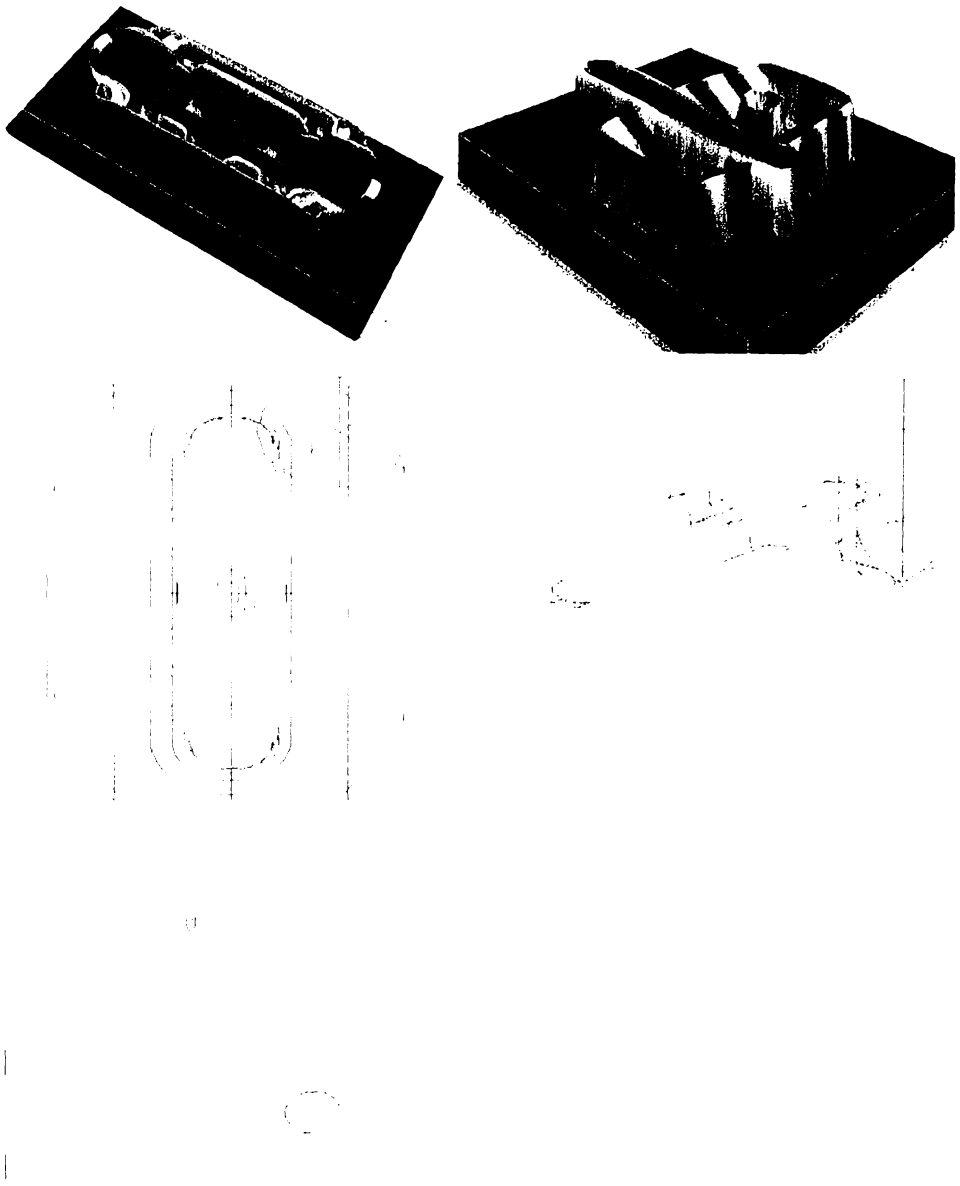


Figura 6.3 *SD care stochează evoluţia (în mărime absolută) a unghiului normalei cu planul orizontal, pentru două SD, şi familii de curbe echinormale proiectate pe SDCS, procesate din 10 în 10 grade.*

Deoarece acest gen de transformare reprezintă una dintre cele mai importante contribuţii în domeniul îmbunătăţirii ciclurilor clasice de frezare a suprafeţelor, se va prezenta, pe un reper dat, evoluţia rugozităţii în funcţie de unghiul de detecţie a zonelor plane.

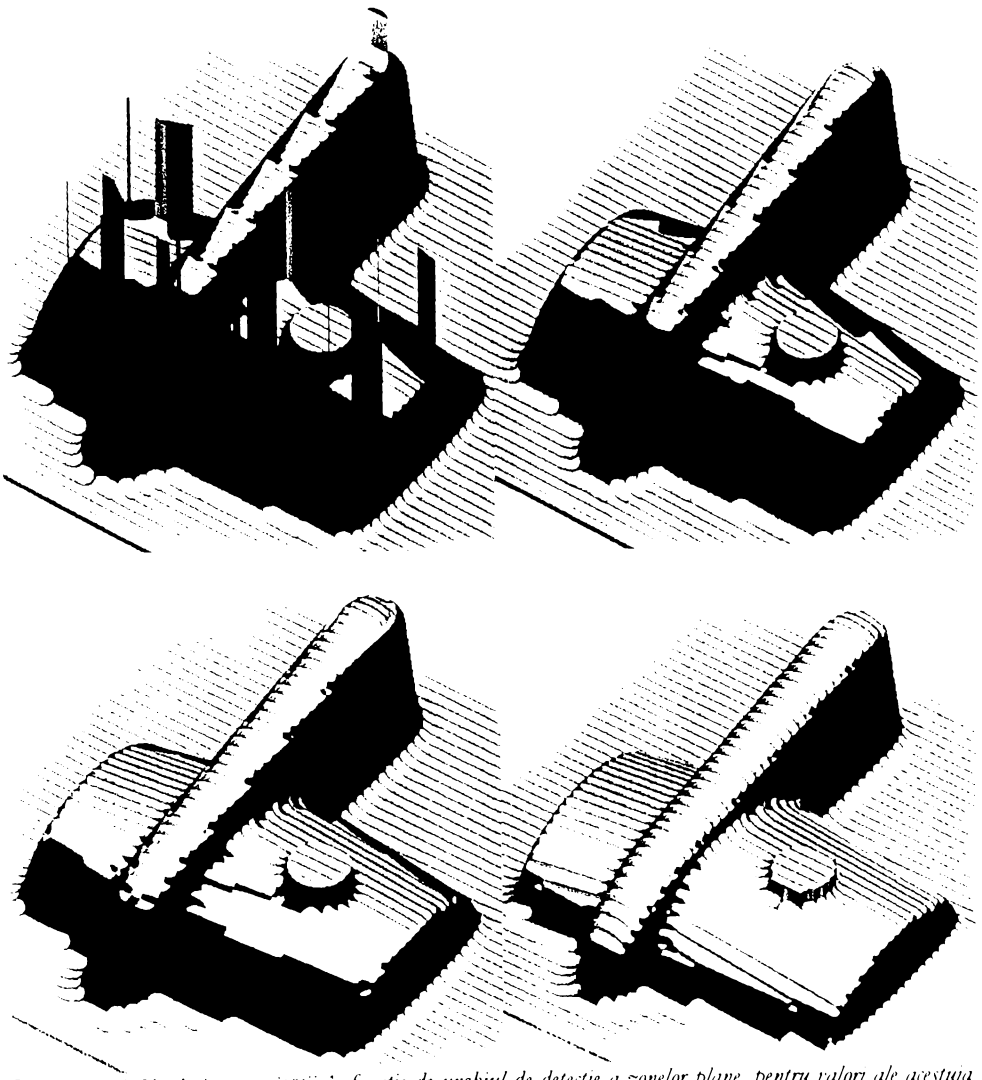


Figura 6.4 *Evoluţia rugozităţii în funcţie de unghiul de detecţie a zonelor plane, pentru valori ale acestuia de 30°, 45°, 60°, 75°. Pasul pe Z şi în planul XY este constant. (EdgeCAM V3.0)*

Analizând aceste patru repere se poate observa că, în cazul frezelor sferice, unghiul optim de detectare a zonelor plane se situează la 45°.

6.3.4. Calculul zonelor critice la frezarea secţiunilor paralele în planul XY

O problemă similară cu cea precedentă este aceea de a descoperi zonele critice, când se frezează într-o anumită direcţie în planul XY, adică locurile unde înclinaţia suprafeţei în direcţie perpendiculară este mare, şi de a compensa acest neajuns cu frezări în direcţii perpendiculare.

Problema fiind geamănă cu cea precedentă, se va încerca prezentarea în mod analog.

Se menţionează că, pentru stadiul actual al evoluţiei maşinilor unelte cu comenzi numerice (MUCN) şi a echipamentelor (ECN) din ţara noastră, precum şi pentru a obţine lungimea fişierului generat cât mai scurtă, cazul particular al frezării cu unghiuri de înclinaţie de 0º sau 90º (în planul XY) joacă un rol foarte important.

În pseudocod, algoritmul are următoarea înfăţişare:

```
NimicSuprafaţăDiscretă.TransfDetecteazăPerpendicular(SuprafaţăDiscretă sdAnaliză, Real rPentruUnghiu = 45.0)
{
   PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
      PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
         PuneLa(nI, nJ, sdAnaliză.laDeLaPerpendicular(nI, nJ, rPentruUnghiu));
} //TransfDetecteazăPerpendicular


NimicSuprafaţăDiscretă.DetecteazăPerpendicular(
            FamiliaDeCurbe fcOut, Real rPentruUnghiu = 45.0)
{
   SuprafaţăDiscretă sdTransformare; //pregăteşte suprafaţa Discretă a transformării
   sdTransformare = sdAceasta; //o iniţializează
   sdTransformare.TransfDetecteazăPerpendicular(sdAceasta); //o calculează
   CreazăCurbe(fcOut, rPentruUnghiu);
} //DetecteazăPerpendicular
```

Notă: metoda laDeLaPerpendicular() nu a fost discutată. Ea nu returnează altceva decât valoarea înclinaţiei curbei în direcţie perpendiculară, în punctul respectiv, faţă de orizontală.

Se prezintă, în continuare, câteva figuri în care se pot observa SD transformate, pentru a analiza locurile critice în direcţia de frezare.

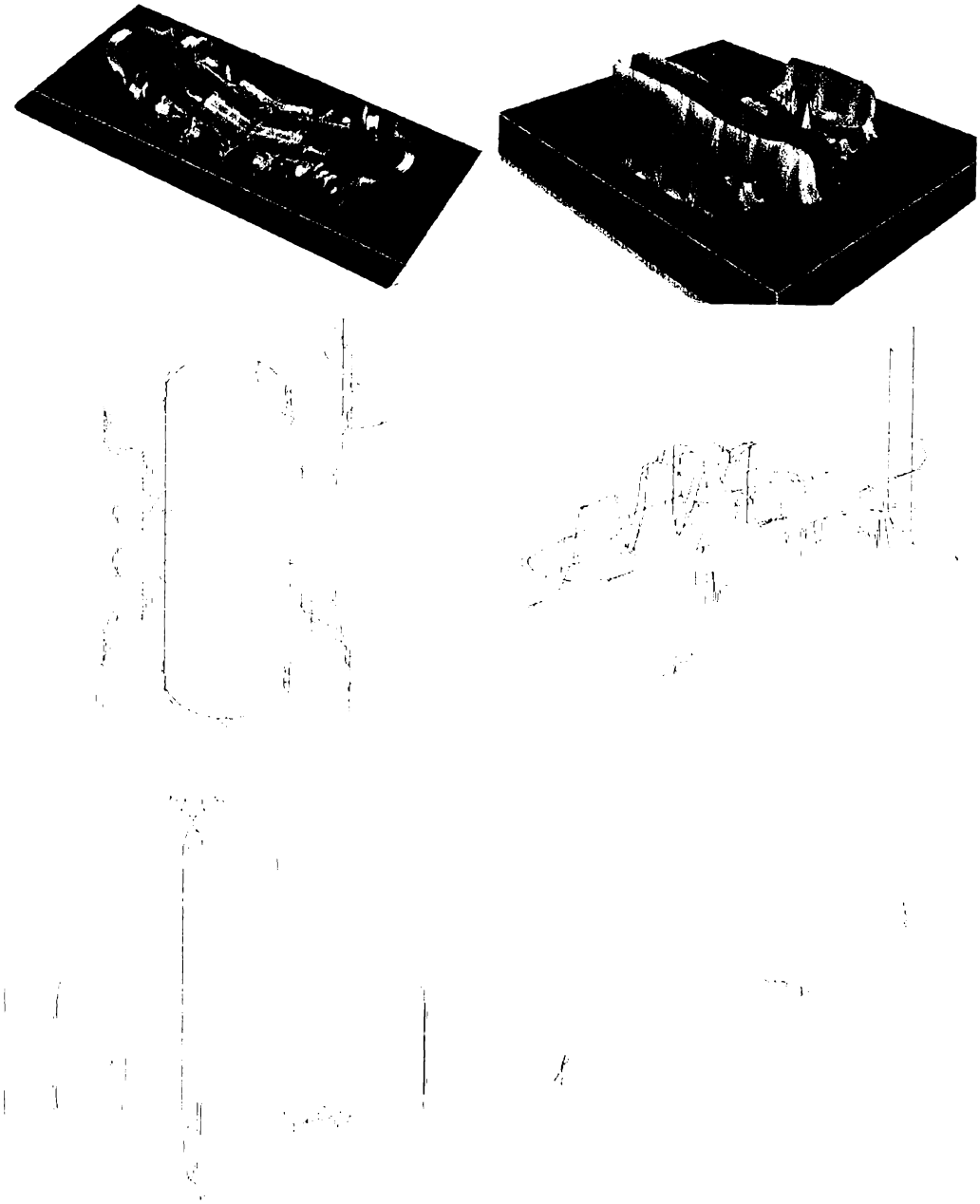


Figura 6.5 Evoluția SD care stochează variația (în mărime absolută) unghiului în direcție perpendiculară, pentru două SD, și familii de curbe echinormale proiectate pe SDCS, procesate din 10 în 10 grade.

Familia de curbe procesate cu aceste tipuri de detecţie a zonelor critice (care reprezintă ropunerea de a freza SD numai cu cicluri paralele în planul XY) arată ca în figura următoare.

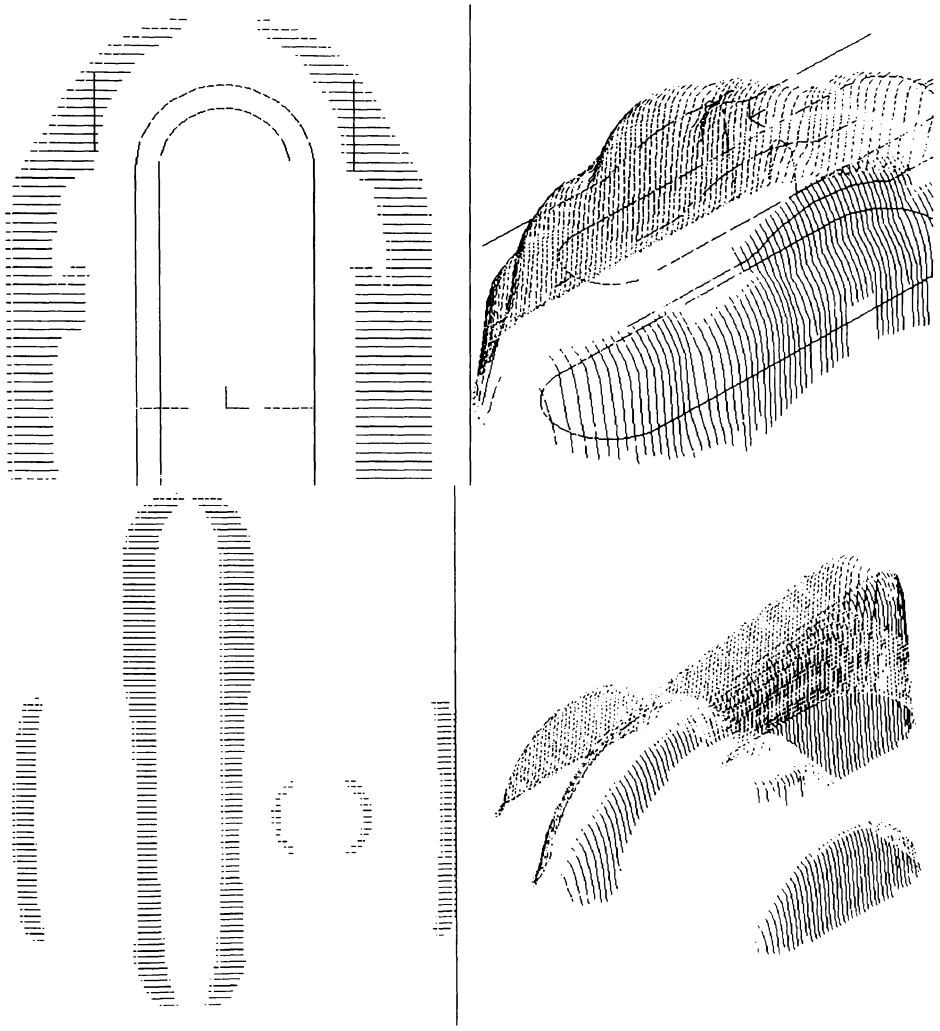


Figura 6.6 Zonele critice şi umplerea lor cu curbe paralele echidistante, care vor fi convertite în fişier NC.

Deoarece acest gen de transformare reprezintă una dintre cele mai importante contribuții în domeniul îmbunătățirii ciclurilor clasice de frezare a suprafețelor, se va prezenta, pe un reper dat, evoluția rugozității în funcție de unghiul de detecție a zonelor plane.

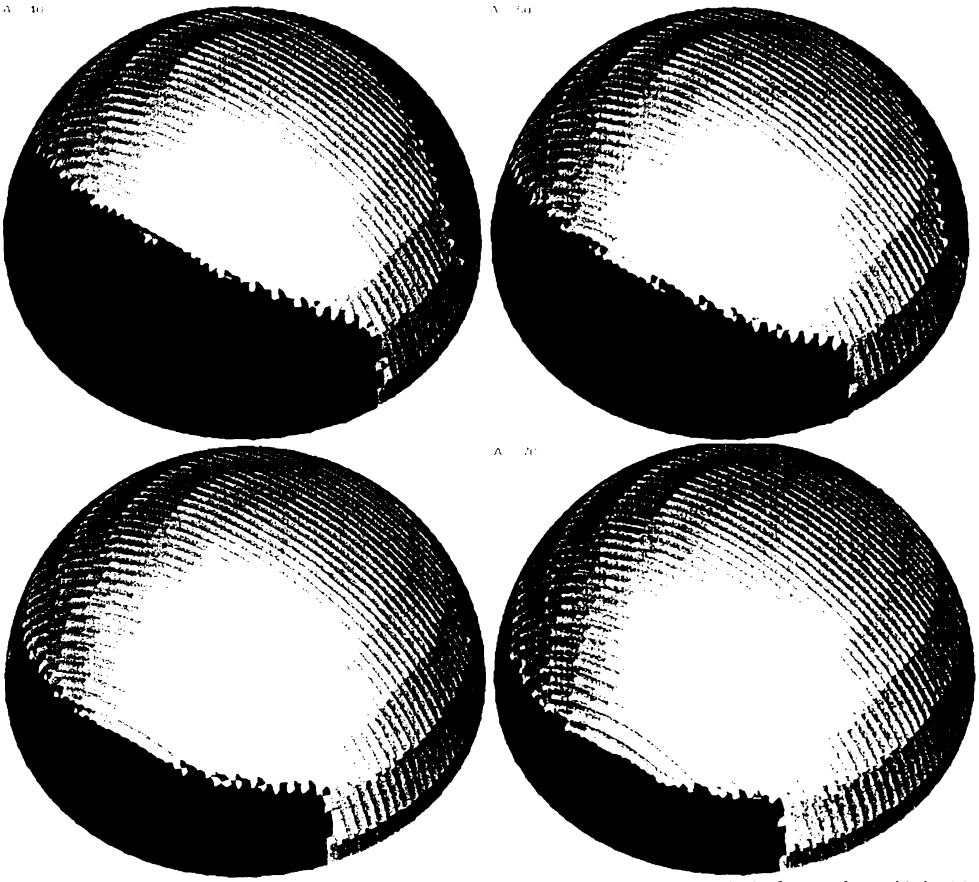


Figura 6.7 Evoluția rugozității și a petei critice (neatingerea rugozității impuse), în funcție de unghiul critic. În aceste zone, pentru atingerea rugozității impuse, este utilizat un ciclu echidistant în direcție perpendiculară. Unghiurile analizate sunt 40°, 50°, 60°, 70°. Se poate observa că cele mai bune rezultate se obțin la 45°.

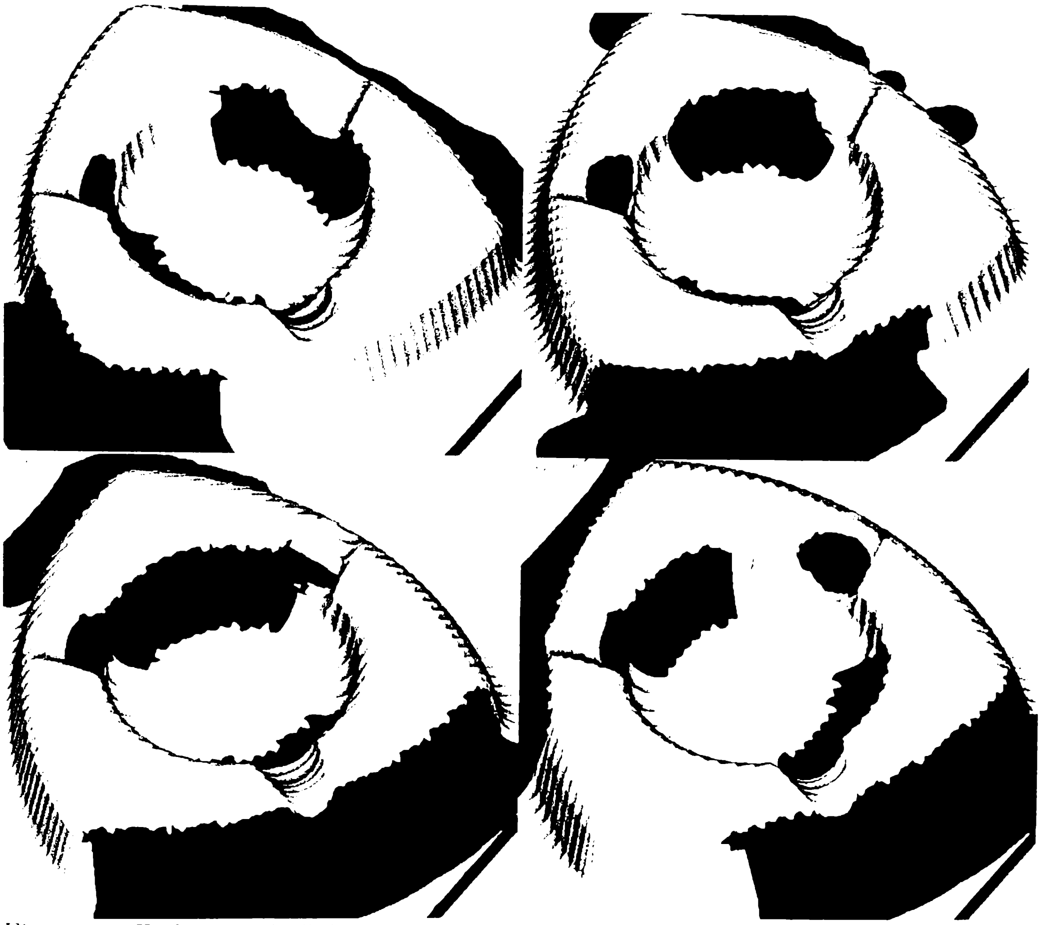


Figura 6.8 Evoluția rugozității și a petei critice (neatingerea rugozității impuse) în funcție de unghiul principal de frezare în planul XY. În aceste zone, pentru atingerea rugozității impuse este utilizat un ciclu echidistant în direcție perpendiculară.
Unghiurile analizate în planul XY sunt 0°, 30°, 60°, 90°. Toate comutările dintre cicluri au fost făcute când normala a avut un unghi mai mare de 45°.

6.3.5. Suprapunerea familiilor de curbe

Pentru a reduce zonele "martor", acelea care rămân la marginea dintre două tipuri de frezări (diferite din cauza detecției zonelor critice), se aplică o metodă de offsetare (echidistantare) a curbelor alese, pentru conversia în fișier NC.

Acest tip de echidistantare poate fi executat prin trei metode:

a) ***în 2D*** *– nerecomandabil, deoarece acest tip de offset nu analizează înclinația suprafeței față de planul orizontal, ar putea, în unele cazuri, să propună spre frezare întregi pereți, aproape verticali (figura următoare);*

b) ***offset inteligent*** *- este exact ca şi cel în 2D, numai că după ce s-a terminat de offsetat, curbele acestea sunt culcate (mulate) pe suprafeţe şi rezultă un nou set de curbe, care sunt în concordanţă cu curvatura suprafeţei de dedesubt (figura următoare);*

c) ***offset în planul unghiular de tăiere*** *– acest tip de offset este mult mai simplu de calculat decât cel precedent, deoarece el creează două tipuri de curbe echiunghiulare de tăiere, "offsetate" cu 1...2 grade. Curbe echiunghiulare de la 10^0 la 80^0 se află în figurile 6.3 c, d şi 6.5 c, d.*

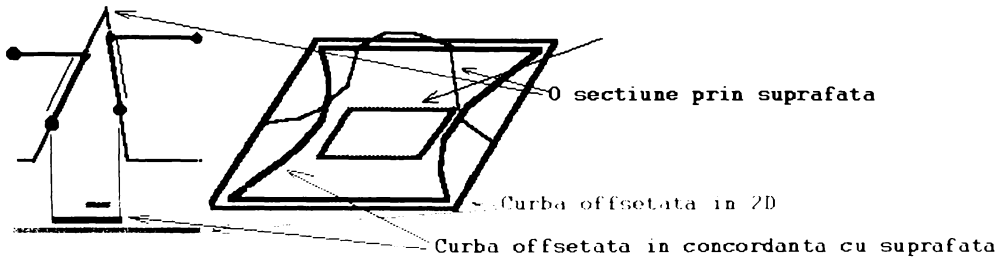


Figura 6.9 Comparaţie între două metode de offset: cea în 2D (portocaliu) şi cea culcată pe suprafaţă.

Autorul foloseşte în prezent numai offsetări de la punctele b şi c, pentru a preîntâmpina frezarea de două ori în vecinătatea pereţilor verticali.

6.3.6. Calculul curbelor echirugozitate

Curbele echirugozitate *sunt cele mai interesante curbe d.p.d.v. al frezării optimizate. Ele reprezintă locul geometric al curbelor care au proprietatea că: dacă se frezează cu o anumită sculă se va ajunge la o rugozitate constantă pe toată suprafaţa, atunci înseamnă că nu se vor suprafreza anumite zone (vor fi lustruite) pentru a se asigura rugozitatea stabilită în alte zone (fig. 6.10). Deci, în concluzie, timpul de frezare este minim (deoarece asigură rugozitatea maximă admisă, constant, pe toată suprafaţa).*

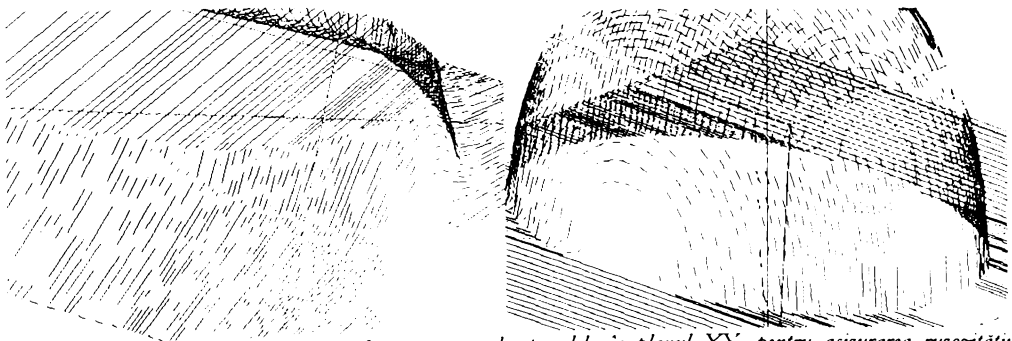


Figura 6.10 Varierea pasului la frezarea cu curbe paralele în planul XY, pentru asigurarea rugozităţii constante. Se poate observa suprafrezarea în zonele orizontale (TechnoCAM V2.0)

În urma analizelor făcute de autor pe 10 repere diferite, pentru aceeaşi rugozitate sau obţinut fişiere NC mai scurte, în general cu 20..50%.

Acest gen de curbe pot fi create printr-o metodă asemănătoare, ca şi celelalte curbe de parametru constant, deci vor fi numite curbe echirugozitate (Eng. Equicusp).

În pseudocod, aceste transformări se pot rezuma astfel:

```
NimicSuprafaţăDiscretă.TransfEchirugozitate(SuprafaţăDiscretă sdAnaliză, Întreg nMetoda)
{
   PutAll(0.0); //setează toată pe 0.0
   SuprafaţăDiscretă sdTempX, sdTempY, sdTemp;

   sdTempX = sdAceasta; //crează ca aceasta
   sdTempY = sdAceasta; //crează ca aceasta

   Dacă(nMetoda = ct_nEchirugozitateX SAU nMetoda = ct_nEchirugozitateXY)
   {
      sdTemp = sdAceasta; //crează ca aceasta
      PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
         PentruFiecare(Întreg nJ = 2; nJ < nY; nJ++)
         {
            PuneLa(nI, nJ, DistXZY(IndexÎnRealPtX(nI-1), IndexÎnRealPtY(nJ), [nI-1, nJ],IndexÎnRealPtX(nI),
               IndexÎnRealPtY(nJ), [nI, nJ]));
            sdtemp.PuneLa(nX - nI, nY - nJ, DistXZY(IndexÎnRealPtX(nX-nI+1),
               IndexÎnRealPtY(nJ), [nX-nI+1, nJ],
               IndexÎnRealPtX(nX - nI), IndexÎnRealPtY(nJ), [nX - nI, nJ]));
         } //PentruFiecare
         Combina(sdTemp, PM_MIN); //combină şi lasă minimul
         sdTempX = sdAceasta; //crează ca aceasta
   } //Dacă

   Dacă(nMetoda = ct_nEchirugozitateY SAU nMetoda = ct_nEchirugozitateXY)
   {
      sdTemp = sdAceasta; //crează ca aceasta
      PentruFiecare(Întreg nI = 2; nI < nX; nI++)
         PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
         {
            PuneLa(nI, nJ, DistXZY(
               IndexÎnRealPtX(nI), IndexÎnRealPtY(nJ-1), [nI, nJ-1], IndexÎnRealPtX(nI),
               IndexÎnRealPtY(nJ), [nI, nJ]));
            sdtemp.PuneLa(nX - nI, nY - nJ, DistXZY(IndexÎnRealPtX(nI),
               IndexÎnRealPtY(nJ), [nX-nI+1,
               nJ], IndexÎnRealPtX(nX - nI), IndexÎnRealPtY(nJ), [nX - nI, nJ]));
         } //PentruFiecare
      Combina(sdTemp, PM_MIN); combină şi lasă minimul
      sdTempY = sdAceasta; crează ca aceasta
   } Dacă

   Dacă(nMetoda = ct_nEchirugozitateXY)
         Combina(sdTempX, PM_MIN); combină şi lasă minimul

} TransfEchirugozitate

NimicSuprafaţăDiscretă.DetectăEchirugozitate( FamiliaDeCurbe fcOut, Întreg nMetoda =ct_nEchirugozitateXY,
   Real rPasul = 1.0)
{
   SuprafaţăDiscretă sdTransformare; pregăteşte suprafata discretă a transformării
   sdTransformare = sdAceasta; o initializează
   sdTransformare.TransfEchirugozitate(sdAceasta, nMetoda); o calculează
```

```
    CreazăCurbe(fcOut, rPentruZ);
}   Detectăîchirugozitate
```

Algoritmul nu face altceva decât să adune distanţa dintre două puncte, suprafeţele rezultante arătând ca în figura 6.11.

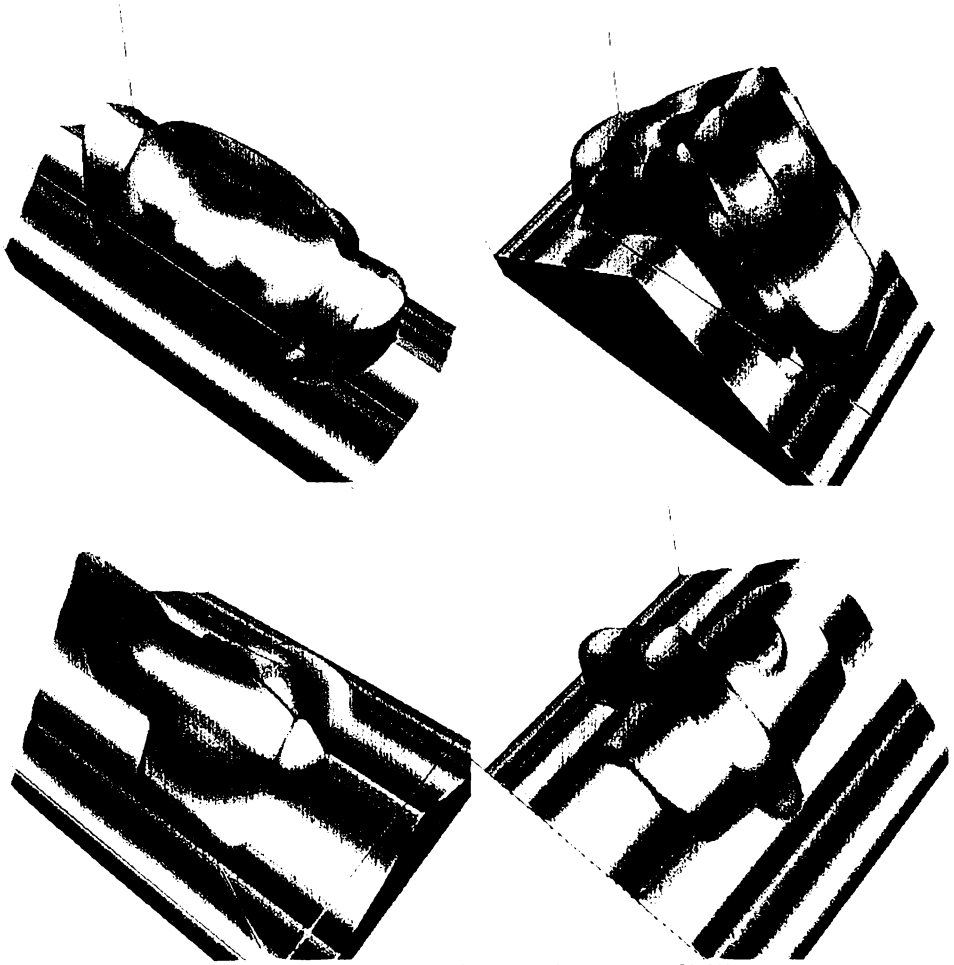


Figura 6.11 Suprafeţe discrete care stochează transformarea echirugozitate. (SDER)

Algoritmul poate avea mai multe materializări, în funcţie de câte începuturi are de pe marginile dreptunghiului care conţine piesa.

Aceaste transformări au fost făcute cu condiţia de a se putea asigura o transformare intermediară, deci ele nu conţin alte informaţii auxiliare despre curbe care taie curbele, destinate fişierului NC. Suplimentar, se poate vorbi despre aceste tipuri de fişiere, care conţin curbe înconjurătoare, şi se pot construi SDER care conţin distanţele proiectate pe SDCS ale curbelor care se intenţionează să fie folosite ca şi curbe de tăiere a fişierului NC.

Folosind variaţia culorii şi desenând SDCS în aceste culori, se obţin reprezentările din figura 5.3, algoritmul nefiind altceva decât cazul particular al offsetului pe suprafaţă a unui dreptunghi sau a unor drepte paralele.

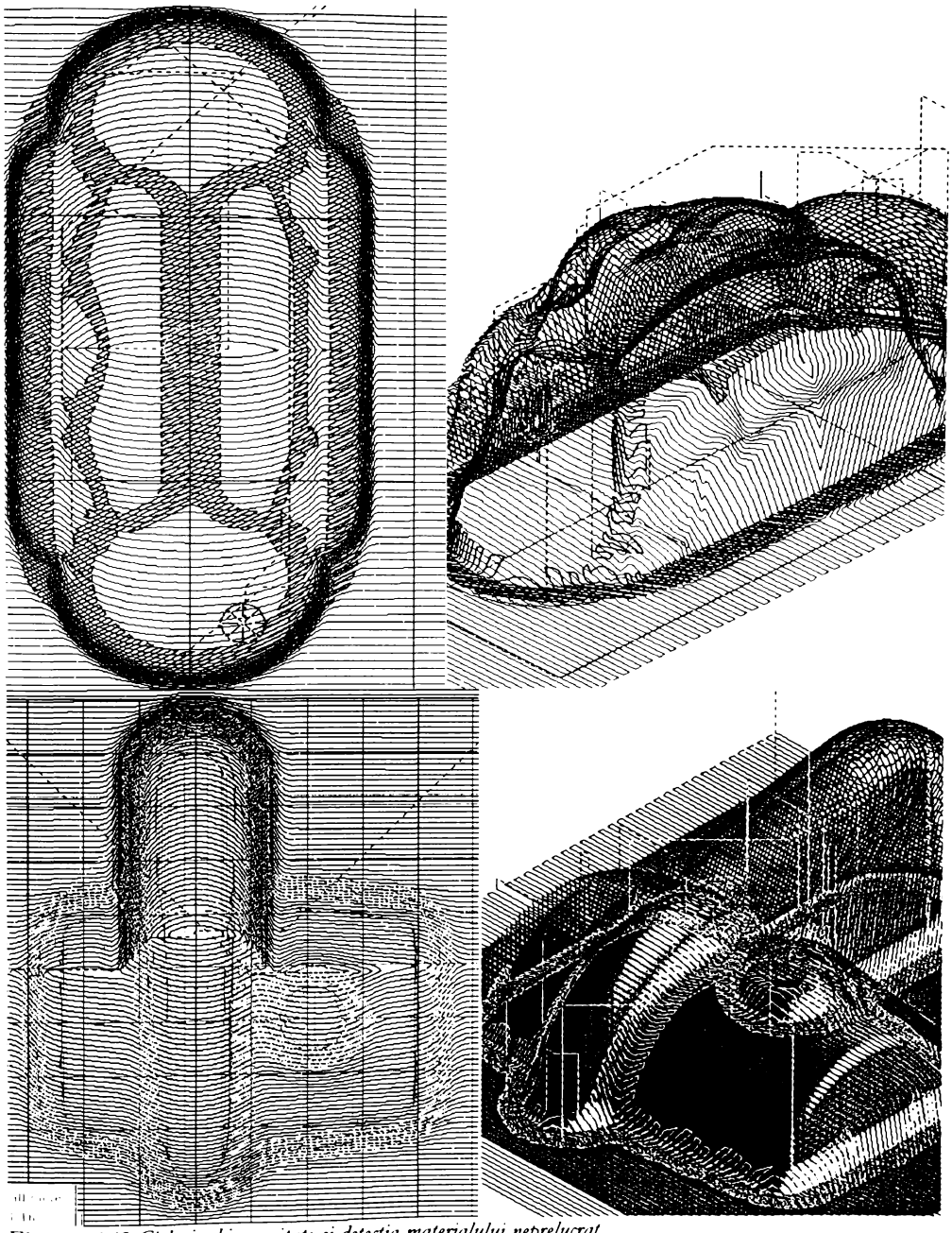


Figura 6.12 Cicluri echirugozitate şi detecţia materialului neprelucrat

6.3.7. Calculul materialului nefrezabil

Materialul nefrezabil *este totalitatea materialului care rămâne neprelucrat, din cauza apariţiei interferenţei sculă - semifabricat. Deci, el nu este o variabilă de pasul sau tipul fişierului NC (a cărui analiză generează calculul* ***materialul nefrezat****), fiind doar o variabilă de geometria semifabricatului şi a sculei, şi se calculează foarte simplu, făcând diferenţa dintre SD de contact (SDC) (Eng.: backoffset) şi SDR.*

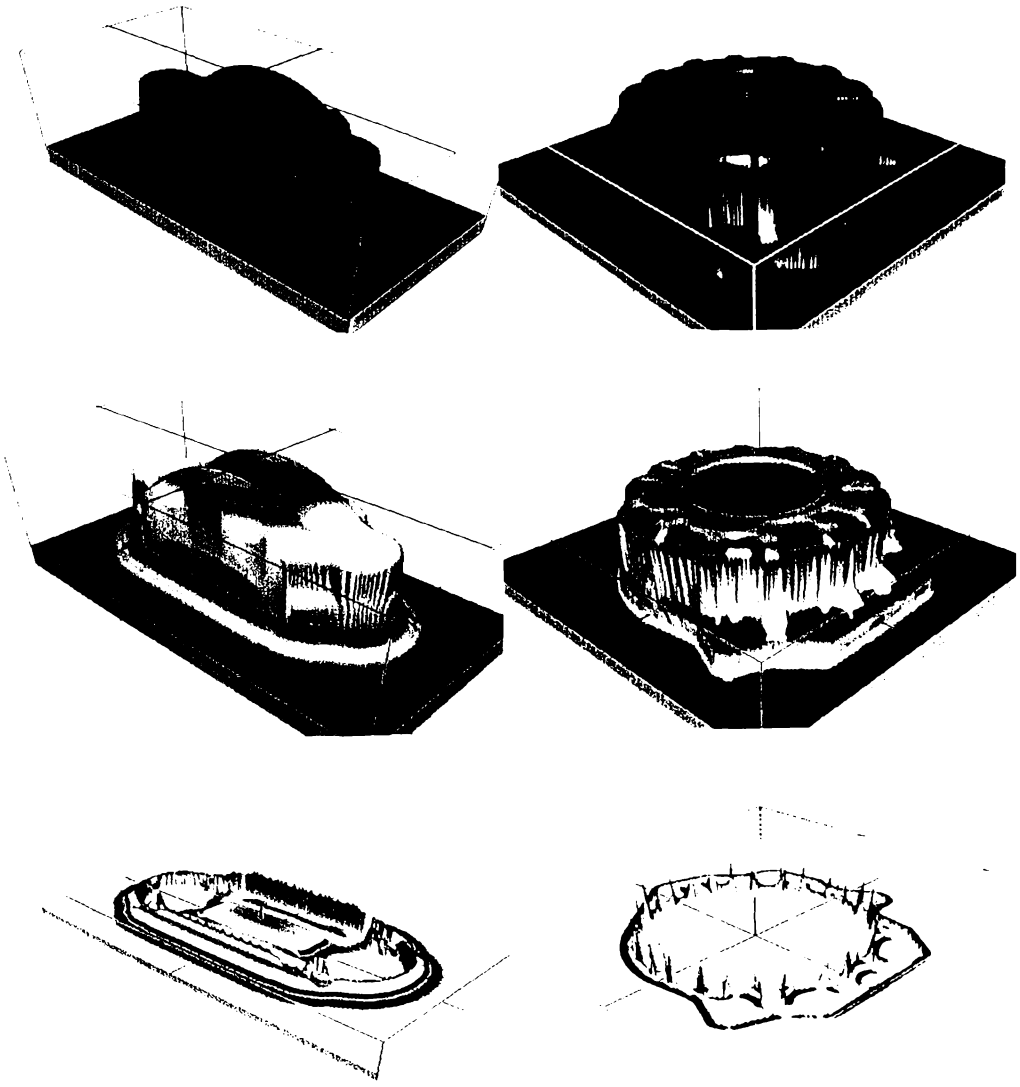


Figura 6.13 Suprafeţele discrete ale reperului (SDR) a, b, ale suprafeţei de contact (SDC) c, d, şi ale materialului nefrezabil (SDN) e, f, pentru două repere complexe importate.

În figura precedentă s-au prezentat pentru două repere, SD reper, SD contact şi SD nefrezabil, pentru clarificarea conceptelor.

Deci, aplicând pe SDN o conversie echipotenţială în curbe de nivel cu o valoare Z puţin mai mare ca 0 (actual autorul foloseşte valori între 0.01 şi 0.2), se obţin curbele căutate ale detecţiei materialului neprelucrat din cauza interferenţei sculă - semifabricat.

De menţionat că aceste curbe mai trebuiesc offsetate pe direcţia normalei la SDR, cu o distanţă egală cu raza sculei următoare, pentru a preîntâmpina urcarea pe pereţi (*antiskating*), propusă pentru eliminarea SDN. În următoarea figură se exemplifică cele expuse în acest subcapitol, într-o secţiune 2D.

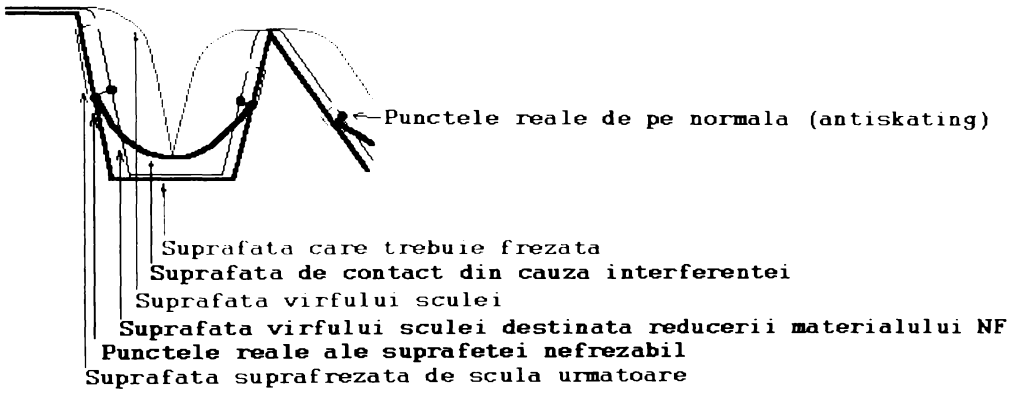


Figura 6.14 *Exemplificarea algoritmului de detectare a materialului nefrezabil şi a necesităţii calculului unui offset pe normală, pentru prevenirea fenomenului de antiskating*

În pseudocod, algoritmul are următoarea înfăţişare:

```
NimicSuprafaţăDiscretă.DetectăNefrezabil(FamiliaDeCurbe fcNefrezabil,
      SuprafaţăDiscretă sdReper, sdSculaAnal, sdSculaMică, Boolean bDetectPlan = FALS)
{
   SuprafaţăDiscretă sdVârfSAculă, sdContact, sdNefrezabil;
   sdVârfSAculă.InitÎnfăşurătoareStatic(sdReper, sdSculaAnal);
   sdContact.InitÎnfăşurătoareStatic(sdVârfSAculă, sdSculaAnal, FALS);
   sdNefrezabil = sdContact – sdReper;
   Dacă(bDetectPlan)  în cazul în care se doreşte frezarea cu o freză cilindrică
   PentruFiecare(Întreg nI = 1; nI <= nX; nI++)
      PentruFiecare(Întreg nJ = 1; nJ <= nY; nJ++)
         Dacă(sdReper[nI, nJ] = sdReper[0, 0]) sdNefrezabil.PuncLa(nI, nJ, 0.0);
   sdNefrezabil.CreazăCurbe(fcNefrezabil, 0.05);  crează curbele de pe sdNefrezabil
   sdReper.Offsetează Curbe();  offsetează cu raza următoarei scule
} TransfÎchirugozitate
```

Câteva repere analizate cu acest algoritm se pot observa în figurile ce urmează. Suplimentar s-au trasat curbele cu cele două scule: cele roşii corespund sculei mari, iar cele negre pentru scula mică, în cazul în care *bDetectPlan* = FALS.

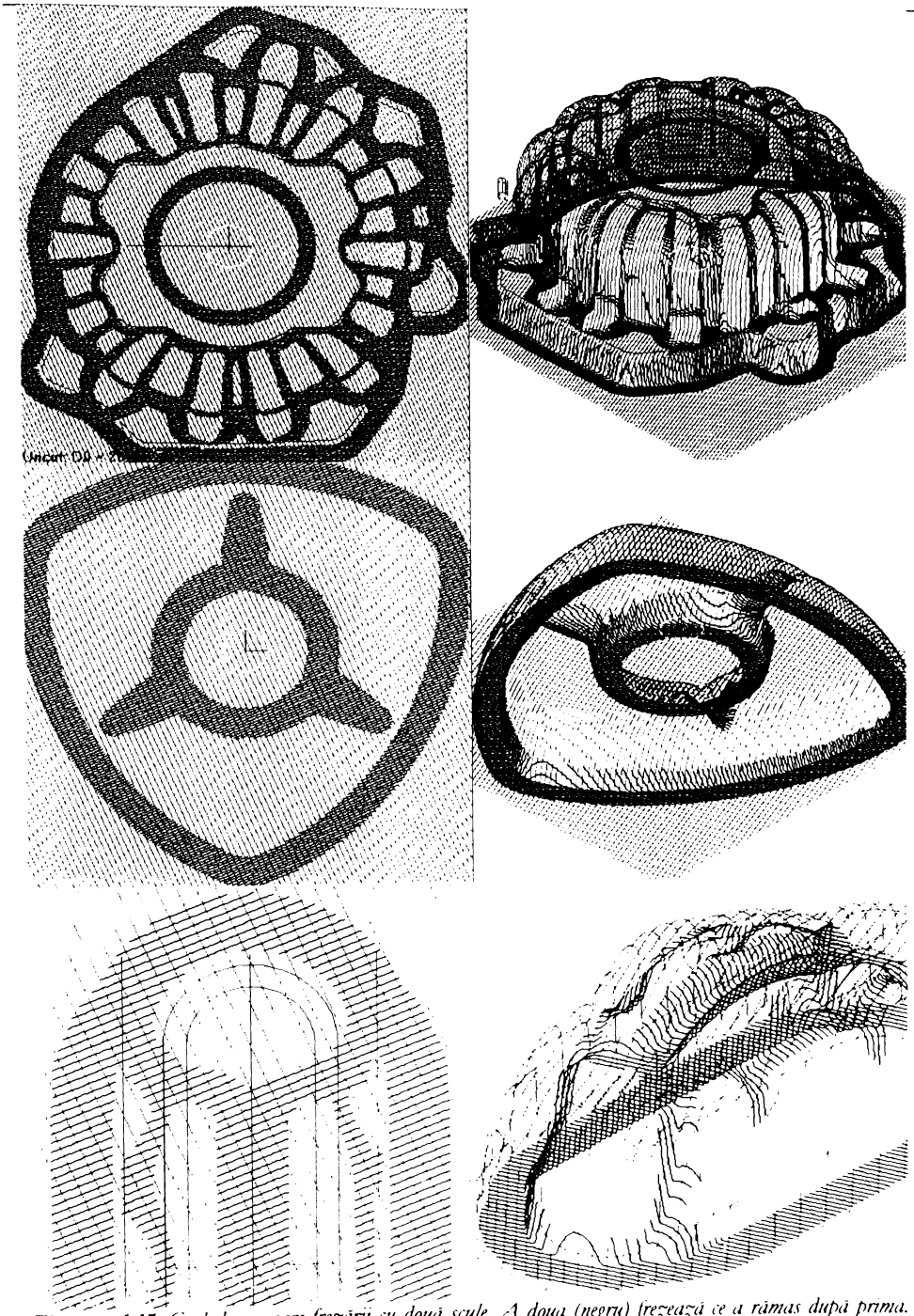


Figura 6.15 *Curbele necesare frezării cu două scule. A doua (negru) frezează ce a rămas după prima. Detectarea materialului nefrezat, fără detectarea zonelor plane.*

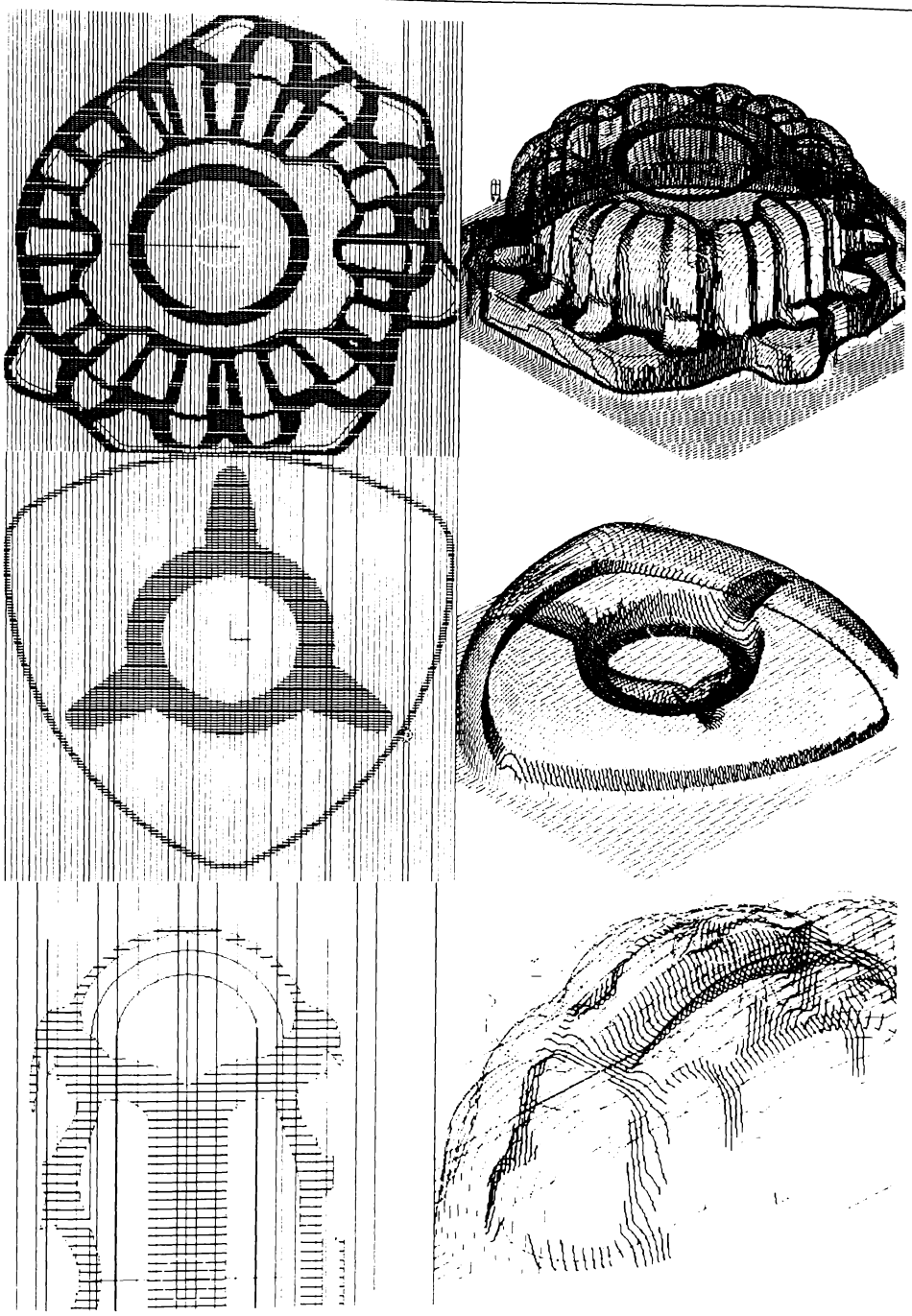


Figura 6.16 *Curbele necesare frezării cu două scule. A doua (negru) frezează ce a rămas după prima. Detectarea materialului nefrezat, cu detectarea zonelor plane.*

În figurile **6.15** şi **6.16** au fost prezentate, pentru trei repere, detectarea materialului neprelucrat, *cu* şi *fără* detectarea zonelor plane.

Se poate observa că în cazul în care se foloseşte o strategie mixtă pentru eliminarea materialului neprelucrat (frezare 3D, sculă sferică, zone curbe + frezare 2D, sculă plană, zone plane) scurtarea timpului de frezare cu a doua sculă este în intervalul procentual de 10% - 50%. Se consideră ca fiind o optimizare semnificativă, deoarece pentru repere de complexitate mare, timpul necesar eliminării materialului nefrezabil este comparabil cu cel al ciclului de finisare.

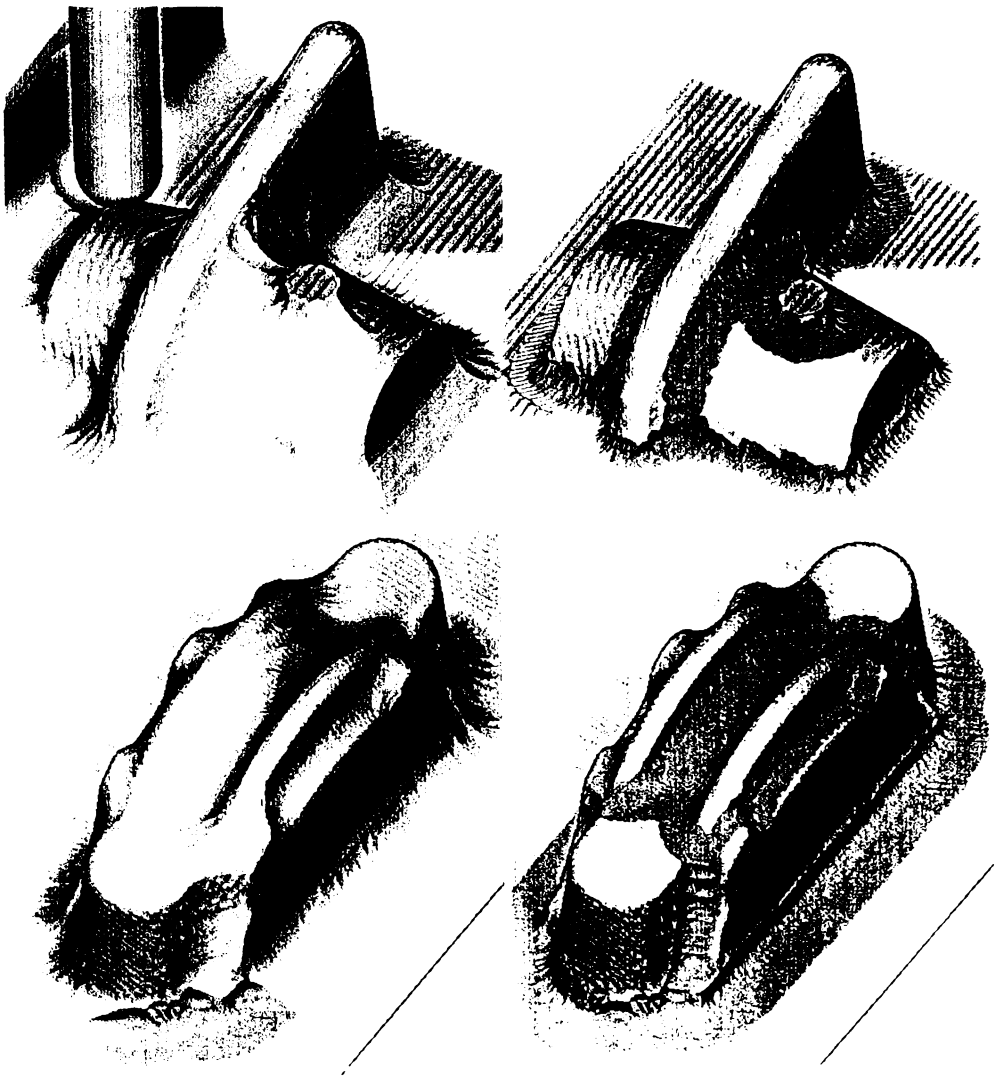


Figura 6.17 *Simularea frezării pe două repere. Ciclul de finisare, cu cap sferic* R=10.0 *mm, frezare echirugozitate, urmat de eliminarea zonelor de interferență, cu cap sferic* R = 2.5 *mm*

De menţionat că cele mai bune rezultate (puţine inflexiuni şi mişcări în avans rapid) au fost obţinute folosind colapsarea echirugozitate a curbelor care definesc materialul nefrezat. Câteva exemple sunt date în figurile următoare.

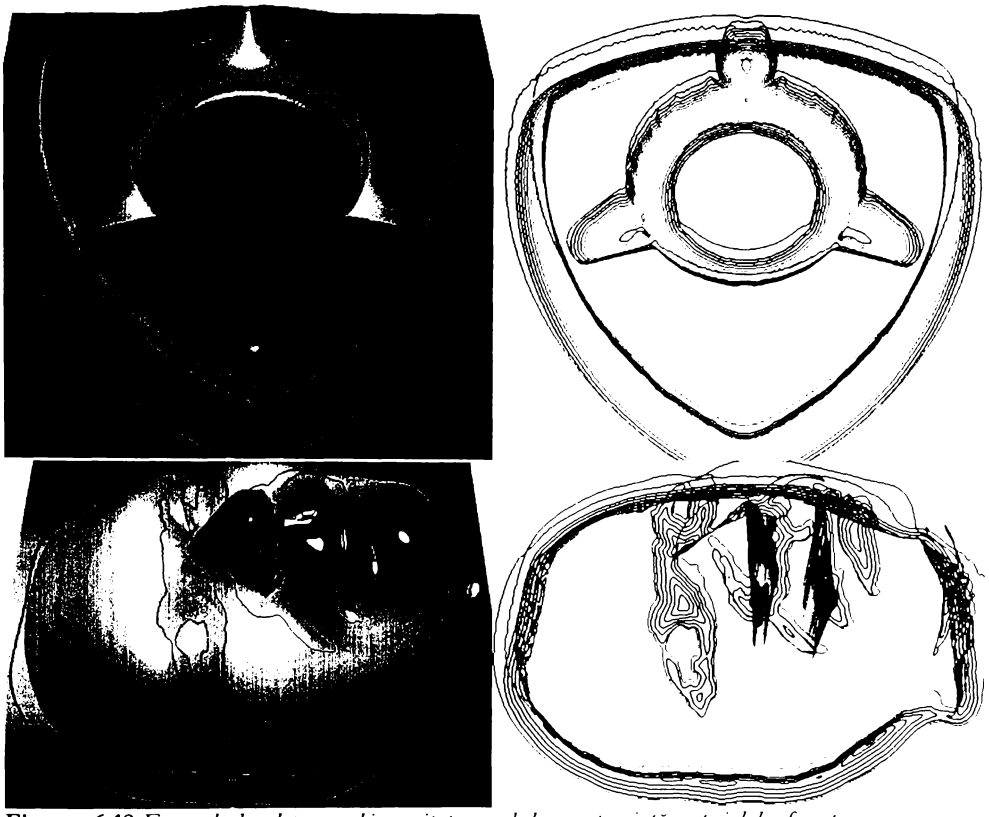


Figura 6.18 *Exemple de colapsare echirugozitate a curbelor ce reprezintă materialul nefrezat.*
În coloana întâi sunt reprezentate suprafeţele şi curbele care delimitează materialul nefrezabil. În coloana a doua, curbele sunt colapsate în 3D, pentru a genera un ciclu de frezare cu condiţia de rugozitate constantă.

Acestea au fost contribuţiile autorului în domeniul frezării optimizate a suprafeţelor discrete, care utilizează algoritmii bazaţi pe transformările discrete ale acestora.

În continuare se va trece la tehnici de optimizare care se aplică pe familia de curbe, sau folosesc tehnici mixte de analiză şi optimizare.

6.4. Metode de analiză a curbelor

O parte din acest gen de optimizări sunt prezente, într-o proporţie mai mică sau mare, în sistemele de fabricaţie. În general, aceste optimizări nu au nimic comun cu geometria suprafeţei, ele legându-se numai de forma şi poziţia curbelor care alcătuiesc traseul vârfului sculei.

6.4.1. Rejecţia punctelor coliniare

Una din cele mai des utilizate metode de optimizare pentru lungimea fişierului NC este aceea de a rejecta punctele intermediare care sunt la o distanţă mai mică decât toleranţa impusă.

Nu se va exemplifica în pseudocod implementarea, fiind lungă şi simplă. Oricum, în Clasa **Curbă** a fost prevăzută o astfel de metodă.

În general, acest tip de rejecţie aduce reduceri ale fişierului NC de 30..90%.

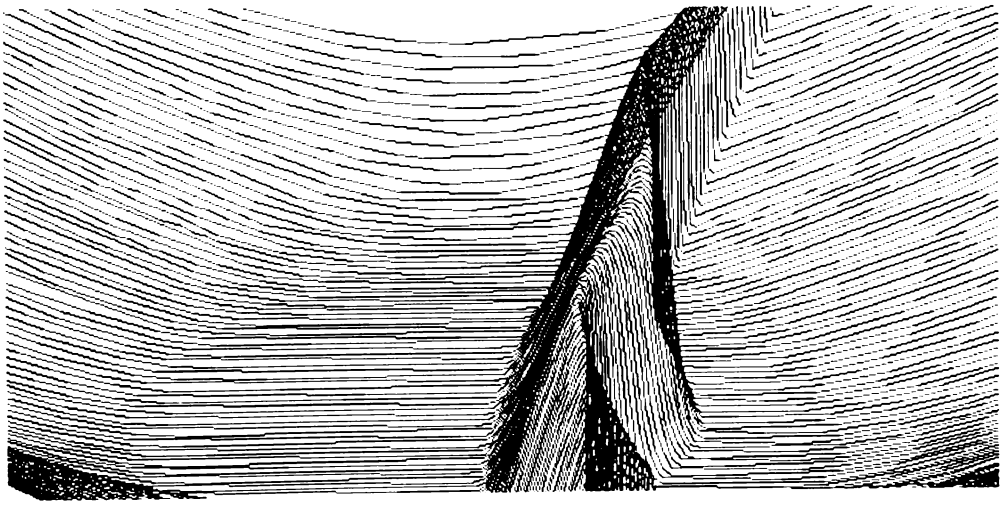


Figura 6.19 *Exemplificarea rejecţiei punctelor intermediare pe un fişier NC exportat cu pas constant de pe o suprafaţă discretă (SD)*

6.4.2. Interpolări superioare

La ora actuală, interpolarea circulară pentru mişcări simple în planele XY, XZ, YZ este prezentă în toate ECN, fiind standardizată de foarte multă vreme. Unele dintre ECN au început să suporte arce de cerc în spaţiu şi curbe spline (*Fanuc, Heidenheim*). Din aceste motive, în sistemele de fabricaţie actuale şi-au făcut apariţia aceste tipuri de interpolări speciale, care asigură o calitate îmbunănătăţită a suprafeţei şi, în general, o lungime mai scăzută a fişierului NC. După unele studii efectuate de autor, lungimea fişierului NC cu acest gen de interpolări superioare activat este de 50%...125%, **deci nu în toate cazurile se asigură o scurtare a fişierului NC, dar se asigură o calitate mai bună a suprafeţei frezate**. O altă raţiune de a implementa interpolările superioare este **moda:** în general utilizatori care au MUCN care suportă aceste interpolări doresc să-şi folosească cât mai multe din facilităţile acesteia.

6.4.3. Minimizarea mişcărilor în avans rapid

Acest tip de optimizare este unul dintre cele mai importante, reducând lungimea mişcărilor în avans rapid cu peste 95% pentru unele frezări particulare şi deschizând premisele unor generări foarte eficiente de fişiere NC pentru MUCN de viteză mare.

Această problemă îşi găseşte o paralelă în optimizarea ***drumului dintre n localităţi*** (problema comis voiajorului), materializat aici prin trasee de sculă.

Rezolvarea completă a problemei comportă o dificultate polinomială de **O(n!)**, cu *n* trasee. Dacă se ia în considerare şi posibilitatea că fiecare traseu poate fi reversibil, complexitatea problemei creşte la **$O(n!*2^n)$**, ceea ce, din punct de vedere al timpilor de rezolvare, duce la un dezastru, în cazul în care numărul de trasee depăşeşte 50. Spre exemplu, pentru rezovarea unei probleme de 100 trasee, un calculator cu procesor Pentium Pro 200Mz a lucrat mai mult de o zi.

În realitate, fişierele NC conţin uzual 200...5000 de curbe individuale, ceea ce constituie o complexitate imposibil de rezolvat la ora actuală. Din această cauză, soluţionarea problemei comis voiajorului **TSP** (***Traveling Salesman Problem***) prin alte metode este una dintre cele mai importante probleme de optimizare, pentru care, în literatura de specialitate încearcă timid să-şi facă apariţia algoritmi din familia celor bazaţi pe automate neuronale sau celulare, sau triangularizările Delaunay.

În cazul frezării, sistemele de fabricaţie folosesc metoda de frezare în zig-zig, optimizarea oprindu-se atunci când se depăşesc cazurile particulare cunoscute.

Având în vedere importanţa pe care o are pentru generarea optimizată a fişierului NC, s-a încercat să se creeze un algoritm care să rezolve această problemă cât mai rapid şi cât mai aproape de situaţia teoretică. Acest algoritm poartă numele "Metodă euristică rapidă de rezolvare a problemei comis voiajorului" - **FHTSP** (***Fast Heuristically Traveling Salesman Problem***).

Complexitatea a scăzut de la **Π n** la **Σ n**, ceea ce face ca timpii de rezolvare a unor probleme de 500 trasee să fie de numai 0.5 secunde, iar pentru 2000 trasee, de 10 secunde.

Soluţia găsită este ideală în proporţie de peste 90%, iar timpii de calcul nu există practic. Acest algoritm **generic** este implementat de către autor în **GNCPP şi EdgeCAM**, iar comportarea lui în practică este ireproşabilă.

Algoritmul se poate rezuma în pseudocod astfel (o versiune unidirecţională):

```
FamiliaDeCurbe.OptimizeazăFHTSP(Întreg nRezolvăComplet = 40)
{
   FamiliaDeCurbe fcTemp;
   Curbă cStart, cCurent;
   Punct pCurent, pStart;

 dacă sunt puţine curbe rezolvă complet problema
   Dacă(nNr <= nRezolvăComplet)
      OptimizeazăTSP();
   Altfel
   {
      cStart = [1];
      fcTemp.PuneLa(cCurent); //adună curba 1
      pStart = cStart[cStart.nNr]; // capătul de sfârşit

      Şterge(1); //şi o şterge
      PentruFiecare(Întreg nI = 2; nI <= nNr; nI++)
      {
         Real rDist = 1e100;
         Întreg nDist = 0;
         PentruFiecare(Întreg nJ = 1; nJ <= nNr; nJ++)
         {
            cCurent = [nI];
            pCurent = cCurent[1]; // capătul de start
            Dacă(DistXYZ(pCurent, pStart) < rDist)
            {
               rDist = DistXYZ(pCurent, pStart);
               nDist = nI;
            } //Dacă
         } //PentruFiecare
         cStart = [nDist];
         pStart = cStart[cStart.nNr]; //capătul de sfârşit
         Şterge(nDist);
      } //PentruFiecare
   } //Altfel
   sdAceasta = fcTemp;   // actualizează familia de curbe
} //OptimizeazăFHTSP
```

Algoritmul nu face altceva decât să sorteze curbele, în funcţie de distanţa minimă dintre capetele de sfârşit şi start, pornind cu prima curbă. Algoritmul care analizează distanţa dintre ambele capete, şi în cazul în care este parcursă în ordine inversă o inversează, este prea complicat pentru a fi expus.

Acest algoritm este însoţit de altul, care leagă capetele mai multor curbe, mai mici decât o distanţă (uzual 3..6 * pasul de procesare), şi astfel, necesarul de ieşiri şi intrări în material se reduce drastic (în exemplul ilustrat, de la circa 500 de intrări şi ieşiri au mai rămas 10, deci o optimizare de circa 98 %, iar timpul necesar acestei optimizări a fost de 2 secunde !).

Se prezintă, spre exemplificare, un caz de fişier NC echipotenţial, cu şi fără acest tip de optimizare, procesat cu aceleaşi condiţii iniţiale.

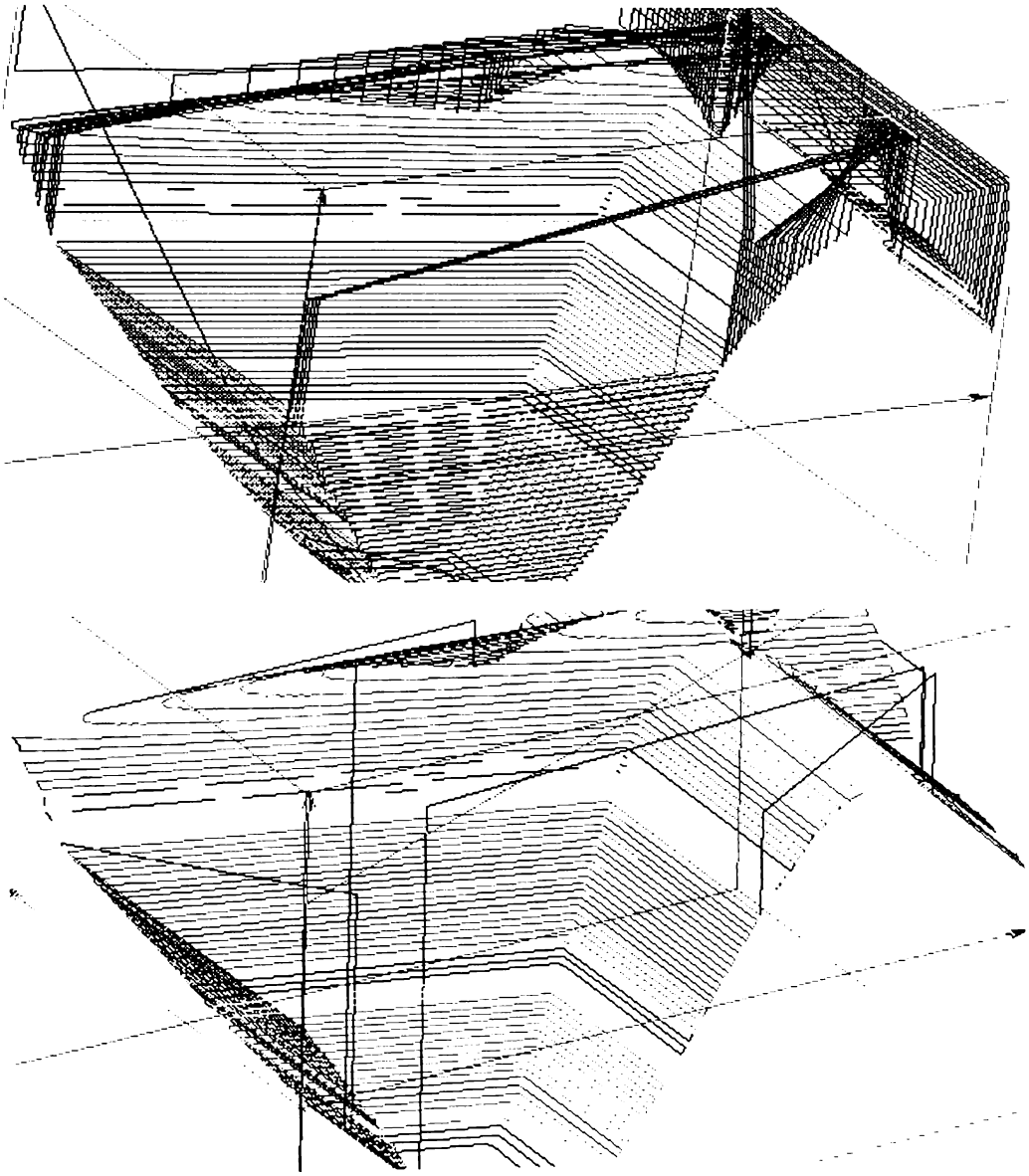


Figura 6.20 *Optimizarea FHTSP pentru reducerea mişcărilor în avans rapid*

- ⇨ *fără FHTSP*
- ⇨ *cu FHTSP*

6.4.4. Modul pseodoadaptiv fără DS

La ora actuală, un sistem adaptiv de variere a avansului în funcţie de cantitatea de material prelevată este foarte costisitor.

Se va prezenta, în două subcapitole separate, două metode (una care utilizează SD şi alta care nu utilizează SD) de a analiza fişierul NC şi de a varia în timp diferiţi parametri, ca avansul sau offsetul, pentru a lua în considerare geometria sculei, a compensa uzurile şi variaţiile termice.

Cea mai simplă metodă de analiză este cea prezentată în continuare, care nu ia în considerare SD, ci numai curbele care alcătuiesc fişierul NC, încercând ca, în urma analizării geometriei acestora, să varieze avansul şi cota pe Z.

În funcţie de rigiditatea sa, dictată de numărul de tăişuri, lungime şi diametru, scula are tendinţa de a freza mai bine radial sau axial. Astfel, un burghiu, care are numai 2 tăişuri, şi deci este puţin rigid, poate fi folosit pentru a freza axial, dar este aproape imposibil de a freza radial cu el, pe când cu o freză care are mai multe tăişuri, este invers.

O primă optimizare este aceea de a varia avansul în funcţie de unghiul de înclinare faţă de planul orizontal şi de numărul de dinţi pe care îi are scula.

De menţionat că acest tip de optimizare îşi găseşte raţiunea doar în ciclurile care sunt destinate degroşării, când maşina unealtă şi scula sunt supuse unor regimuri care depind de rigiditatea lor. Acest tip de optimizare duce la reduceri semnificative de timp, de 50..500%, deoarece în ciclurile de degroşare se folosesc scule foarte rigide, care favorizează frezările radiale, în detrimentul celor axiale, cu un factor de 3:1 ... 5:1. Deci, folosind un avans diferit (mai mare) la frezarea radială faţă de cea axială, scula şi MU pot fi încărcate eficient. Cum la reperele uzuale frezările axiale ocupă un timp de cca 20%, rezultă că, folosindu-se acelaşi avans, impus de frezarea axială, MU lucrează 80% din timp cu un avans de 3..5 ori mai mic, fiind deci posibilă o optimizare de 240% ... 400% !

Compensarea uzurii se poate face analizând câtă cantitate de fişier NC a fost frezată în fiecare moment, definind un coeficient procentual care reduce avansul după fiecare (spre exemplu) metru de material prelucrat.

Autorul utilizează valori de 0.98...0.999 ale coeficientului de micşorare care se aplică multiplicativ valorii avansului pe metru de material frezat.

La compensarea uzurii se poate vorbi şi de o compensare posibilă în direcţia Z, de circa 0.001...0.01 mm/m. Este posibilă doar pe direcţia Z, deoarece în momentul generării fişierului NC, compensări spaţiale de uzură nu se mai pot face, din cauza complexităţii calculelor implicate.

Se consideră discuţia despre compensarea dilataţiilor termice doar una pur academică, ea nefiind practic operabilă, din cauza mult prea multor factori de care depinde. Aceşti parametri, ca *starea termică a maşinii, sculei, lichidului, mediului, variaţiile în timp ale acestor parametri,* duc la necunoscute şi complexităţi mult prea mari pentru a fi analizate şi compensate cu precizie.

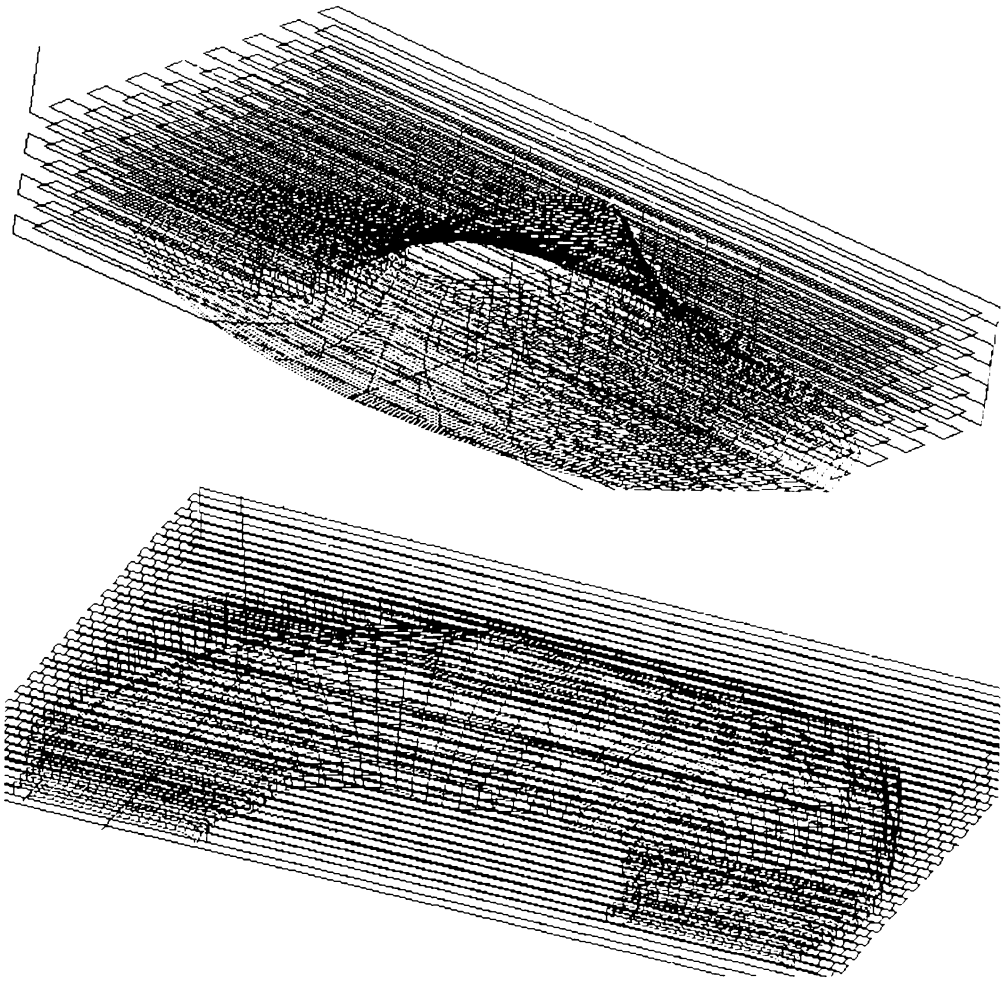


Figura 6.21 *Fişiere de degroşare cu pilotarea inteligentă a avansului, în funcţie de de tipul mişcării axială sau radială (roşu mişcării axiale) (TechnoCAM V2.0)*

În figura **6.21** se pot observa câteva aspecte discutate în acest subcapitol şi cantitatea relativ scăzută a mişcărilor axiale în comparaţie cu cele radiale.

6.4.5. Eliminarea punctelor de inflexiune

Eliminarea punctelor de inflexiune în fişierele NC duce la:

- ⇨ *lungirea vieţii MU, prin nesupunerea lanţurilor cinematice la şocuri de oprire şi pornire;*
- ⇨ *scăderea semnificativă a timpului de frezare la ciclurile de mare viteză;*
- ⇨ *calitate îmbunătăţită a suprafeţei (vibraţiile fiind scăzute).*

De-a lungul vremii s-a încercat transformarea ciclurilor clasice echipotenţiale şi echidistante în XY, în vederea satisfacerii acestui deziderat. Metoda cea mai simplă de transformare este aceea a adunării unui arc de cerc, pentru a asigura continuitatea derivatei întâi pe două curbe echidistante. Arcul se trasează tangent la trei curbe, două fiind traseele, iar a treia fiind dreapta care uneşte cele două capete (fig. 6.22 a, b). Această transformare înmoaie dicontinuitatea din punctele de inflexiune.

O altă metodă este de a adăuga o cubică, care este constrânsă de capetele curbei şi direcţiile acestora (fig. 6.22 c).

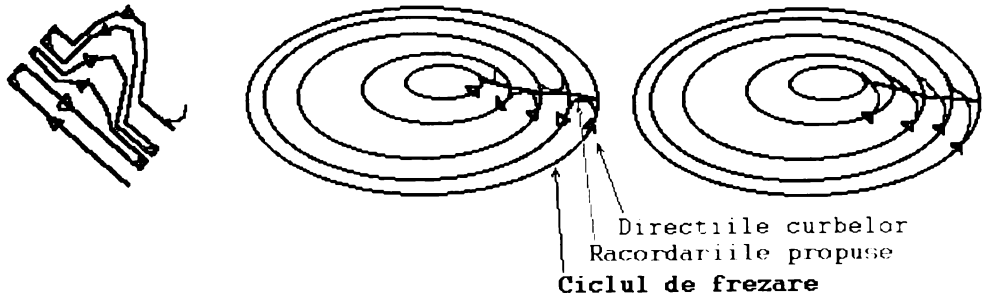


Figura 6.22 *Diferite metode de transformare a ciclurilor clasice în vederea reutilizări lor la frezare de viteză mare.*

De menţionat că transformările propuse sunt în general destul de bune şi satisfac, în general, dorinţele utilizatorilor. Dezavantajul lor este acela că sunt destul de greu de implementat.

6.5. Metode de analiză mixtă

Metodele de analiză şi optimizare mixtă sunt foarte utile şi eficace, ele permiţând analize si generări de fişiere NC sofisticate, foarte rapide, ca: analiza curvaturii, analiza zonelor de securitate minime pentru mişcări rapide, detecţii de interferenţe cu luarea în considerare a elementelor de fixare sculă şi semifabricat, cota minimă Z de securitate la mişcări în avans rapid etc.

Ele nu sunt aşa de generale ca şi cele care nu utilizează SD, din cauză că trebuiesc create SD, dar odată create SD asociate, se pot folosi metodele acestora, care permit analizele enumerate.

6.5.1. Spiralele lui Billator

Contribuţiile autorului în domeniul fabricaţiei cu viteză mare sunt transformările curbelor **închise echirugozitate** în spirale. Acestea au fost denumite ***"Spiralele lui Billator"***, după cyberporecla autorului, şi sunt o familie specială de spirale, care se calculează printr-un morf continuu pe familia de curbe care se conţin una pe alta. În cazul în care există ramificaţii în arborele de curbe, se procedează recursiv la calcularea spiralelor lui Billator, pentru fiecare ramificaţie în parte. Acest tip special de frezare va fi conţinut în exclusivitate în produsul **EdgeCAM V4.0** şi sunt în întregime generate de autor.

Ordinea operaţiilor necesare pentru a transforma **orice suprafaţă demulabilă** în spirale continue, fără puncte de inflexiune, care sunt generate pentru a lăsa o rugozitate constantă (deci timpul de fabricaţie este minim, nu există suprafrezare, ca în cazul altor tipuri de frezări) este următoarea:

- ⇨ ***se modelează reperul SDR;*** *(*import, modelare etc*);*
- ⇨ ***se calculează corecţia de sculă SDC;***
- ⇨ ***se transformă SDC în SDER*** *(*echirugozitate*);*
- ⇨ ***se filtrează,*** pentru a înmuia colţurile şi a rotunji curba de 3..10 ori;
- ⇨ ***se secţionează echipotenţial*** *(*se obţine un set de curbe închise, în 90% din cazuri cu un singur maxim local - punct de colapsare*);*
- ⇨ ***se ordonează toate curbele în acelaşi sens;***
- ⇨ ***se rotesc curbele până în punctul de start cel mai apropiat;***
- ⇨ ***se aplică un morf continuu pe curbe***
- ⇨ ***se proiectează curba rezultată pe SDC;***
- ⇨ ***se exportă ca fişier NC curbele rezultante;***

De menţionat că, pe parcursul lucrării, au fost atinse toate aceste metode, cu excepţia celor de ordonare şi rotire a curbelor pentru a avea punctul de start cât mai apropiat.

Se vor prezenta pentru prima oară, în pseudocod, aceste două metode:

```
NimicCurbă.Roteşte(Întreg nDeCâteOri)  roteşte o curbă cu nDeCâteOri pasi
{
   PentruFiecare(Întreg nI = 1; nI < nNr; nI++)
```

```
    {
        PuncLa(nNr, [1]);
        Sterge(1);
    } //PentruFiecare
} //Roteşte


Real Curbă.AriaCuSemn()  //returnează aria cu semn pentru a detecta direcţia
{
    Real rAria = 0.0;
    Punct pStart = [1], pEnd;
    PentruFiecare(Întreg nI = 2; nI < nNr; nI++)
    {
        pEnd = [nI];
        rAria = rAria + (pEnd.rX – pStart.rX) * (pEnd.rY + pStart.rY);
    } //PentruFiecare
    return(rAria / 2.0);
} //AriaCuSemn


NimicCurbă.FăTrigonometric() //transformă o curbă în sens trigonometric
{
    Dacă(AriaCuSemn > 0.0)
        SchimbăSensul();
} //FăTrigonometric


NimicCurbă.FăOrar() //transformă o curbă în sens orar
{
    Dacă(AriaCuSemn < 0.0)
        SchimbăSensul();
} //FăOrar


NimicFamiliaDeCurbe.FăTrigonometric() //transformă curbele în sens trigonometric
{
    Curbă c;
    PentruFiecare(Întreg nI = 1; nI < nNr; nI++)
    {
        c = [nI];
        c.FăTrigonometric();
    } //PentruFiecare
} //FăTrigonometric


NimicFamiliaDeCurbe.FăOrar()  //transformă curbele în sens orar
{
    Curbă c;
    PentruFiecare(Întreg nI = 1; nI < nNr; nI++)
    {
        c = [nI];
        c.FăOrar();
    } //PentruFiecare
} //FăOrar


NimicFamiliaDeCurbe.Roteşte()  //roteşte curbele până au punctul de start minim
{
    Curbă cStart, cEnd;
    cStart = [1];

    PentruFiecare(Întreg nI = 2; nI < nNr; nI++)
```

```
    {
        cEnd = [nI];
        pStart = cStart[1];
        Întreg nRoteşte = 0;
        Real rDist = 1e100;
        PentruFiecare(Întreg nCurbă = 1; nCurbă < cEnd.nNr; nCurbă++)
        {
            pEnd = cEnd[nCurbă];
            if (DistXYZ(pStart, pEnd) < rDist)
            {
                rDist = DistXYZ(pStart, pEnd);
                nRoteşte = nCurbă;
            } //Dacă
        } //PentruFiecare
        Roteşte(nRoteşte);
        cStart = cEnd;
    } //PentruFiecare
} //Roteşte


NimicFamiliaDeCurbe.Spirală(Întreg nPuncte = 100) //transformă curbele în spirale
{
    Curbă cStart, cEnd;
    cStart = [1];

    PentruFiecare(Întreg nI = 2; nI < nNr; nI++)
    {
        cEnd = [nI];
        PentruFiecare(Întreg nT = 2; nT < nPuncte; nT++)
        {
            pStart = cStart.laDeLaT(nT/nPuncte);
            pEnd = cEnd.laDeLaT(nT/nPuncte);
            Add(pStart + (pEnd - pStart) * nT / nPuncte);
        } //PentruFiecare
        cStart = cEnd;
    } //PentruFiecare
} //Spirală


//transformă SDC în spirale
NimicSuprafaţăDiscretă.SpiraleleLuiBillator(FamiliaDeCurbe fcOut, Real rStep = 1.0)
{
    SuprafaţăDiscretă sdTemp;
    sdTemp.TransfEchirugozitate(sdAceasta, 255);
    sdTemp.Înmoaie(5);  face media celor patru vecini de 5 ori creşte continuitatea '
    sdtemp.CreazăToateCurbele(fcOut, rStep);
    fcOut.FăOrar();
    fcOut.Roteşte();
    fcOut.Spirală();
    ProiecteazăToateCurbele(fcOut);  cteşte cota Z '
}  SpiraleleLuiBillator
```

Aşa arată, în pseudocod, o versiune simpificată a calculului "**Spiralelor lui Billator**" (SLB), care suportă doar un singur maxim local pe SDER. De menţionat că, rezolvarea cazurilor cu multiple maxime locale este o problemă destul de complicată, care necesită liste de incluziune locală şi tehnici de "scanline". Implemetarea algoritmului care rezolvă toate cazurile necesită mai mult de 40 pagini de pseudocod.

Câteva repere transformate în SLB şi frezate se pot observa în figurile următoare.

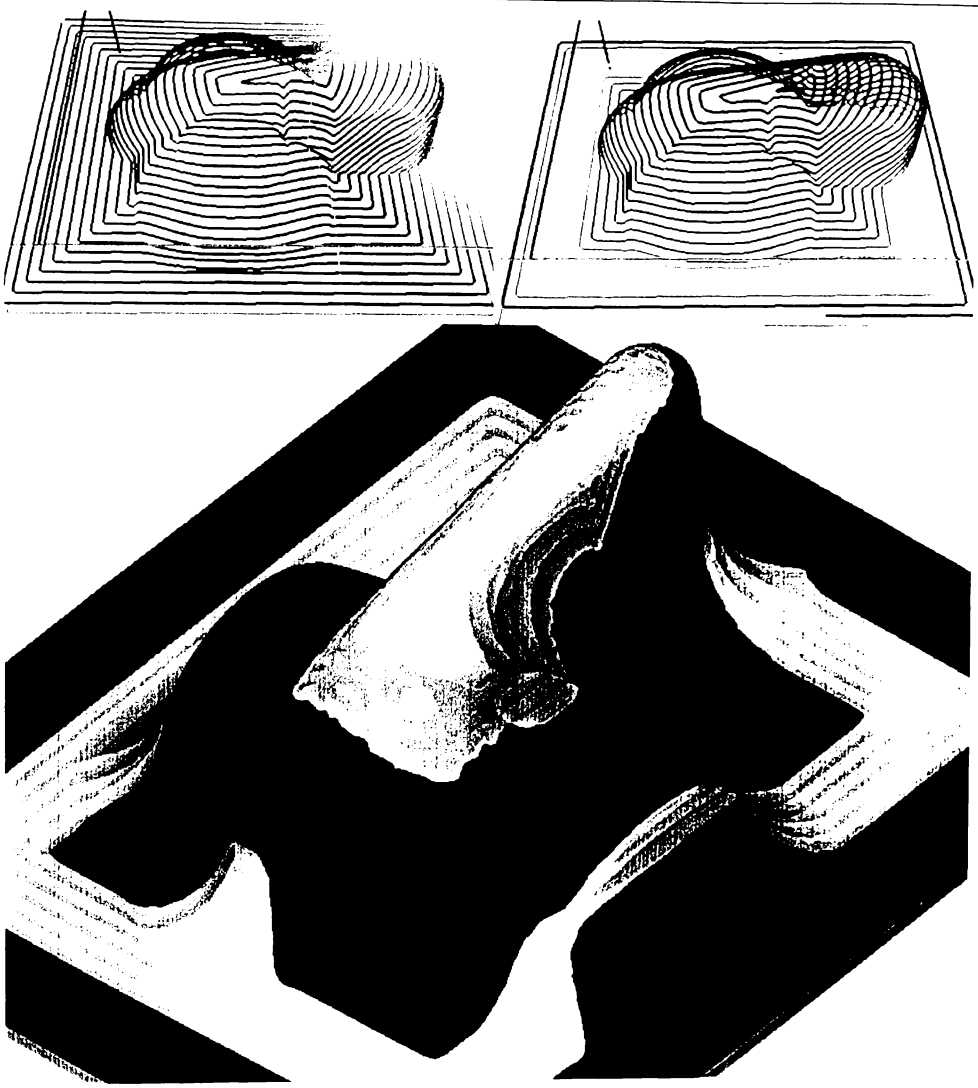


Figura 6.23 *Un reper analizat în vederea frezării cu curbe echirugozitate.*

- ⇨ *curbele echirugozitate proiectate pe suprafaţa discretă corectte de sculă (SDCS);*
- ⇨ *curbele transformate într-o spirală continuă şi proiectate pe SDCS (**Spirala lui Billator**);*
- ⇨ *simularea fişierului NC rezultant, cu variereа culorii din 5 în 5 spire.*

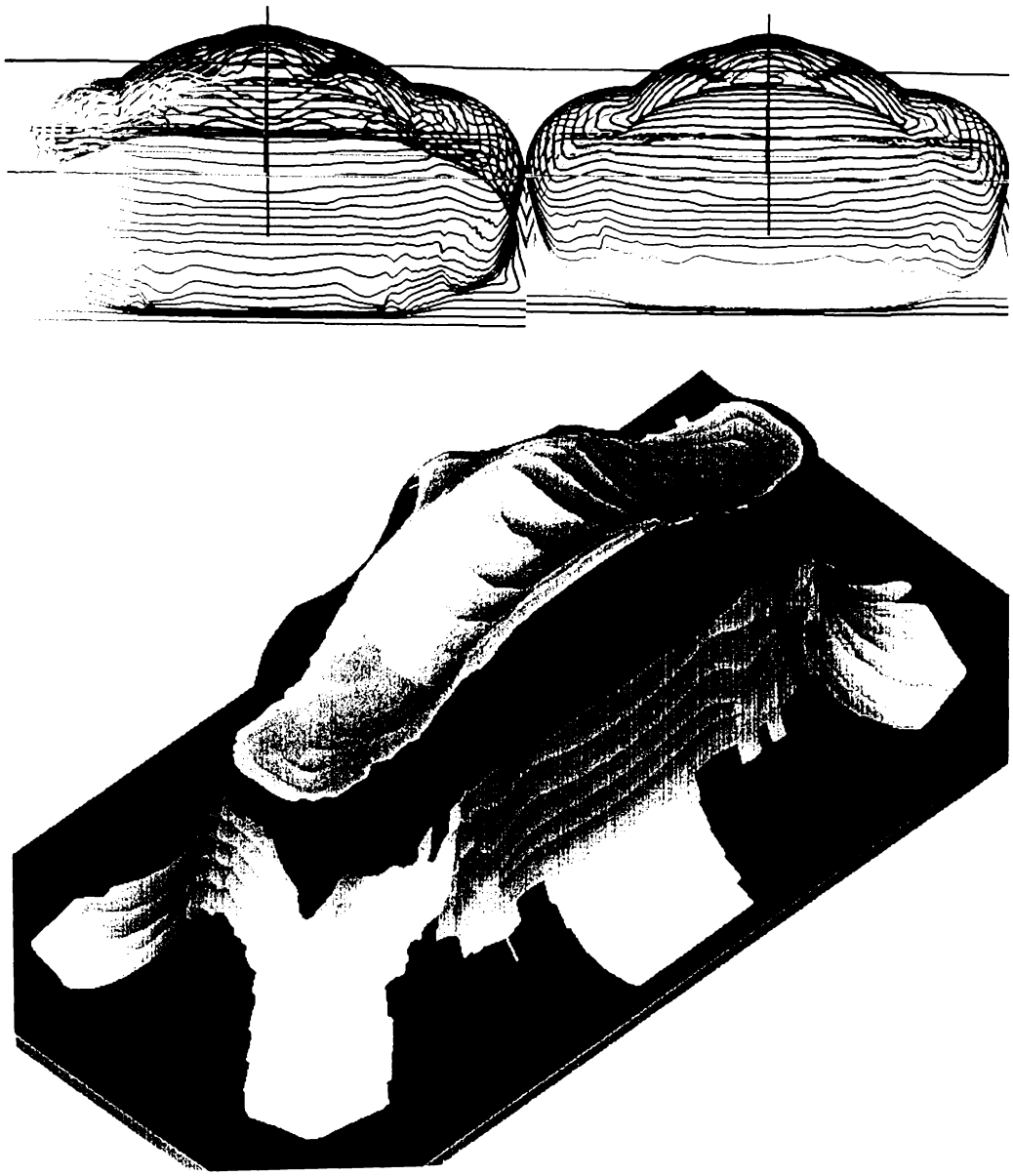


Figura 6.24 *Un reper analizat în vederea frezării cu curbe echirugozitate.*

- ⇨ *curbele echirugozitate proiectate pe suprafaţa discretă corecţie de sculă (SDCS);*
- ⇨ *curbele transformate într-o spirală continuă şi proiectate pe SDCS (**Spirala lui Billator**);*
- ⇨ *simularea fişierului NC rezultant, cu variaţia culorii din 5 în 5 spire.*

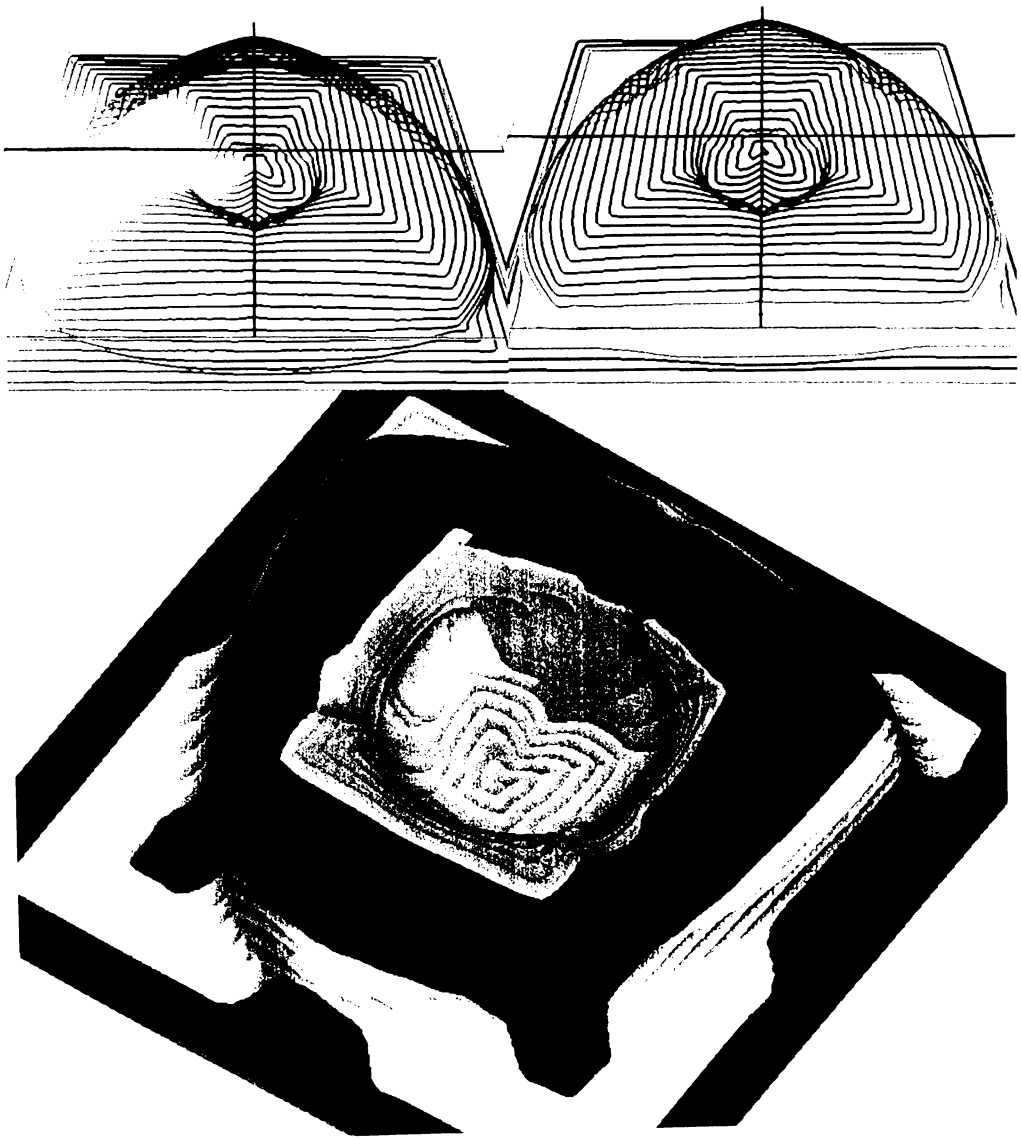


Figura 6.25 *Un reper analizat în vederea frezării cu curbe echirugozitate.*

- ⇨ *curbele echirugozitate proiectate pe suprafaţa discretă corecţie de sculă (SDCS);*
- ⇨ *curbele transformate într-o spirală continuă şi proiectate pe SDCS (**Spirala lui Billator**);*
- ⇨ *simularea fişierului NC rezultant, cu varierea culorii din 5 în 5 spire.*

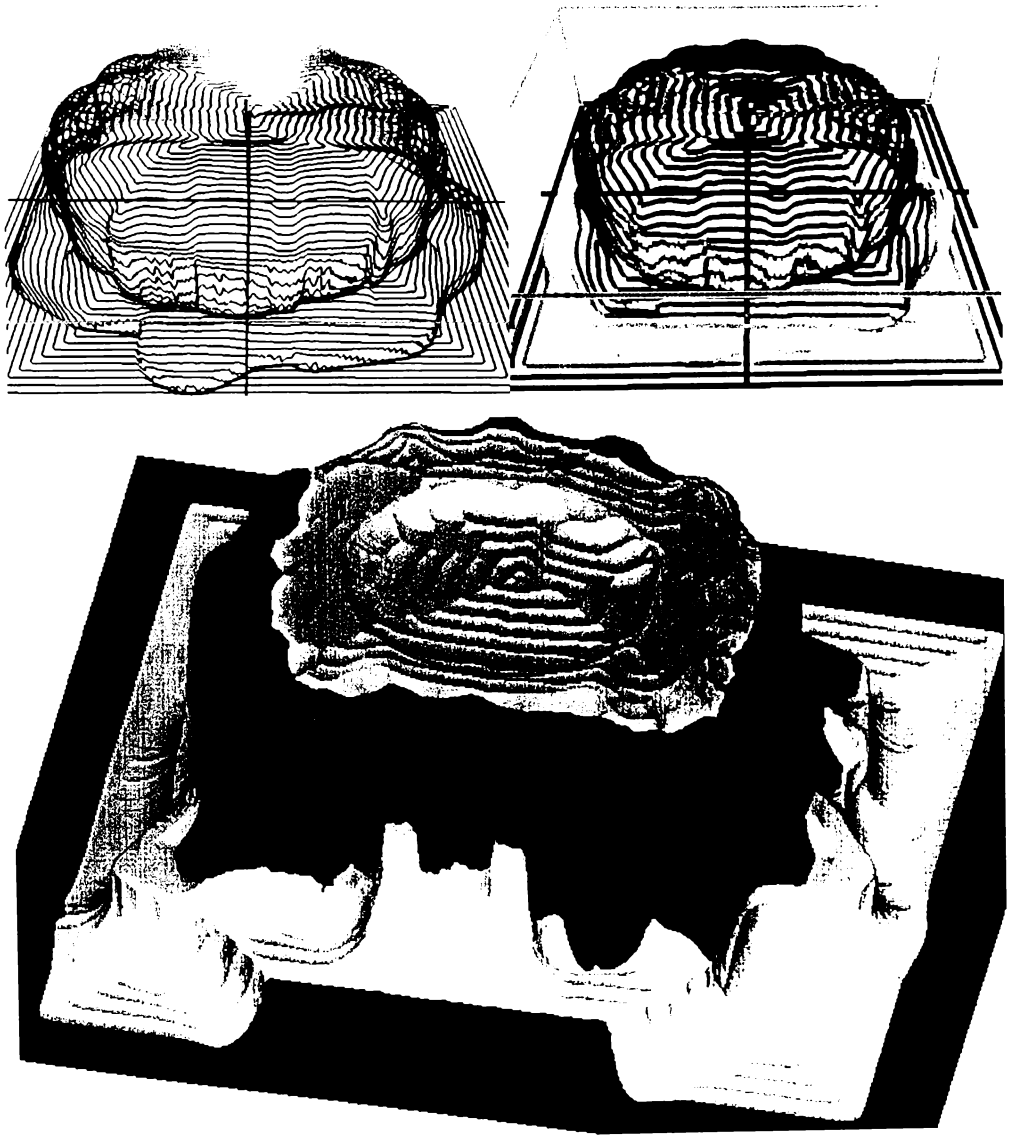


Figura 6.26 *Un reper analizat în vederea frezării cu curbe echirugozitate.*

- ⇨ *curbele echirugozitate proiectate pe suprafaţa discretă corecţie de sculă (SDCS);*
- ⇨ *curbele transformate într-o spirală continuă şi proiectate pe SDCS (**Spirala lui Billator**);*
- ⇨ *simularea fişierului NC rezultant, cu variarea culorii din 5 în 5 spire.*

Fiecare reper a fost prezentat în trei ipostaze: momentul în care a fost convertit în curbe echirugozitate ordonate şi cu acelaşi sens (a), momentul în care a fost convertit în SLB (b), şi

calitatea şi distribuţia rugozităţii, după ce a fost simulată o frezare cu fişierul NC generat din SLB (c).

6.5.2. Modul pseudoadaptiv cu SD

Modul pseudoadaptiv, în cazul în care se cunoaşte SD, este mult mai realist, putându-se şti în fiecare moment cantitatea de material prelevată în direcţie radială şi transversală. Se pot rupe, spre exemplu, linii lungi în bucăţi mici, în funcţie de cantitatea de material prelevată, şi se poate încetini ciclul, în oricare moment, pentru a preveni ruperea sculei etc.

Calculele de uzură a sculei se pot face foarte precis, cunoscându-se în fiecare moment cât material a fost frezat cu fiecare porţiune de pe generatoarea sculei.

Acest mod de a pilota scula, variind avansurile în timp real, duce la încărcări optime ale sculei şi maşinii unelte, scăzând timpii de frezare şi crescând calitatea suprafeţei şi viaţa MU, datorită reducerii şocurilor şi vibraţiilor.

6.5.3. Detecţia interferenţelor

În cazul analizării frezării, în timp real, se poate cunoaşte starea semifabricatului, putându-se detecta interferenţe de tipul:

- ⇨ *portsculă - semifabricat;*
- ⇨ *elemente de fixare - sculă;*
- ⇨ *portsculă - elemente de fixare;*

În cazul în care se cunoaşte şi SDR se pot analiza, suplimentar, zonele în care suprafaţa a fost **distrusă** (fişier NC prost generat) sau zonele cu material nefrezat, analizând semnul SD, care reprezintă diferenţa dintre SD simulare şi SDR.

Analiza frezării, în timp real, se poate face relativ simplu, cunoscând geometria fişierului NC şi a sculei, folosind modul de punere PM_MIN a SD sculă (**SDSC**) în fiecare poziţie, folosind un algoritm simplu de trasare a liniei într-un raster (Bresenham).

O figură capturată dintr-un sistem care simulează frezarea prin această tehnologie se poate observa în figura următoare (6.27).

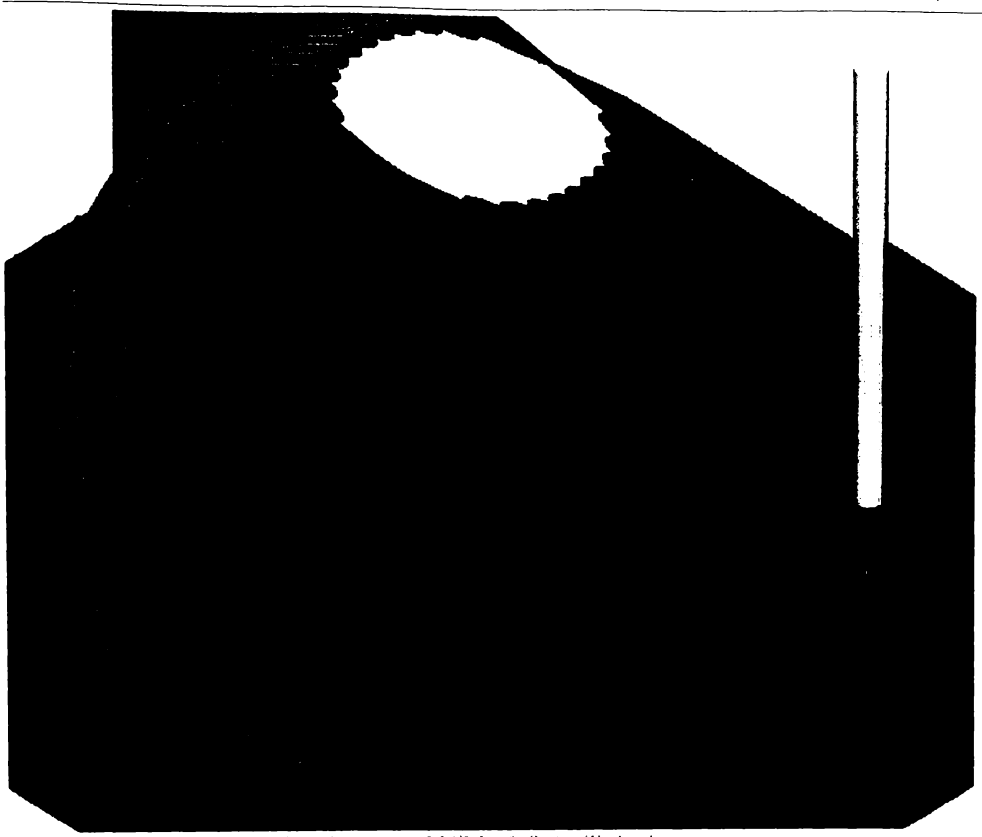


Figura 6.27 *Simularea solidă a frezării, cu NCVerify Sirius Technology*

Alte beneficii imediate ale simulării în timp real a frezării sunt cele ale găsirii cu acurateţe a cotei Z minime, pentru mişcările cu avans rapid.

6.6. Optimizarea traseelor de sculă

În acest subcapitol va fi expusă ca metodă de optimizare generarea fişierului NC într-un singur ciclu, în cazul în care nu se doresc cicluri combinate pentru eliminarea zonelor critice.

De-a lungul vremii s-au încercat diferite metode de a scădea rugozitatea suprafeţelor prelucrate. Astfel, în sistemele de fabricaţie actuale există o metodă de frezare cu pas variabil, în ciclurile clasice de frezare cu Z constant sau după o direcţie în planul XY. Prin această metodă (fig. 6.29 a, b) se asigură o rugozitate constantă, într-un timp rezonabil, în general mai mic cu 10..90% din timpul necesar frezării suprafeţei cu pasul cel mai mic, constant. Această metodă are însă un dezavantaj, acela de a suprafreza întreaga porţiune care nu este înclinată, lungind timpii teoretici cu 30..90%.

Autorul a încercat să elimine acest dezavantaj şi a dezvoltat o metodă care rezolvă problema, crescând doar local numărul de treceri. Această metodă este relativ simplă şi duce la scăderi ale timpului de frezare, cu 20...60% în comparaţie cu metoda clasică, cea cu pas dinamic.

Metoda descrisă a fost numită "cu pas inteligent" (smart step) şi, în general, ea este doar cu 10..15% mai lungă decât cea teoretică, de frezare la rugozitate constantă !

Deci, sintetizând pentru o suprafaţă oarecare şi o rugozitate dată, se poate afirma că timpii necesari frezării pentru diferite cicluri sunt proporţionali cu următoarele valori:

Echirugozitate:	**1.0**
Pas inteligent:	**1.10 .. 1.15**
Pas dinamic:	**1.6 .. 3.0**
Fără variere de pas:	**5.0 .. 20.0**

Tab 6.1

Date sunt trasate după analiza unui set de peste 50 de repere discretizate.

La ora actuală, ciclurile de frezare "cu pas inteligent" sunt implementate doar în sistemele de fabricaţie **EdgeCAM V4.x, TechnoPack V2.x** (implementate de autor) şi **HyperMill**, în ciclurile de colapsare a curbelor cu mulare pe suprafaţă.

Frezarea echidistantă cu pas inteligent (fig. 6.29 c, d) există doar în **TechnoPack V2.x** şi se preconizează a fi introdusă şi în **EdgeCAM V5.0**, fiind descoperită şi implementată în exclusivitate de autor.

Dezvoltarea în pseudocod este destul de sofisticată şi lungă, din acest considerent se va încerca doar o explicare sumară a metodei (care principial se apropie destul de mult de **calculul zonelor critice la frezarea secţiunilor paralele în planul XY**).

Se consideră, pentru orice mişcare elementară din traseul sculei, înclinaţia acesteia în direcţie perpendiculară. Dacă aceasta este în intervalul 0...30^0 nu se execută nimic, dacă este în intervalul 30^0 ... 60^0 se activează un comutator **b2x** şi se marchează punctul de start, iar dacă este mai mare de 60^0 se activează un alt comutator **b3x** şi se marchează punctul de start.

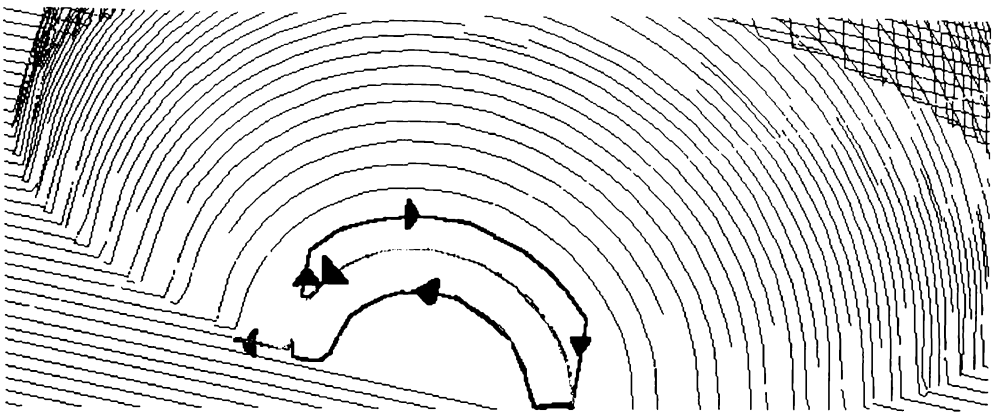


Figura 6.28 *Dublarea / triplarea pasului, când înclinaţia devine mare.*

În cazul în care se schimbă un interval 0^0 - 30^0 - 60^0 - 90^0, şi unul din comutatoarele b2x sau b3x este activat, se execută o înjumătăţire (sau o treime) a pasului şi se face o buclă, ca în figura următoare.

NOTĂ:Mişcare elementară înseamnă că fiecare segment de dreaptă este mai mic, spre exemplu de 1 mm.Unghiurile de 30^0, 60^0 sunt orientative.

O simulare solidă este prezentată în figura **6.28**, în care se poate observa (relativ) rugozitatea constantă.

Avantajul frezării cu dublarea sau triplarea pasului este acela că se poate implementa relativ uşor, nefiind necesar un modul de suprafeţe discrete, utilizându-se codul existent pentru ciclurile echidistante; urmărind **tabelul 6.1** se poate observa că timpii, în general, sunt foarte apropiaţi de frezarea teoretică la rugozitate constantă (fiind, în urma unui studiu efectuat pe peste 20 de repere doar cu 10..15 % mai mari !), care necesită însă calcule relativ complexe.

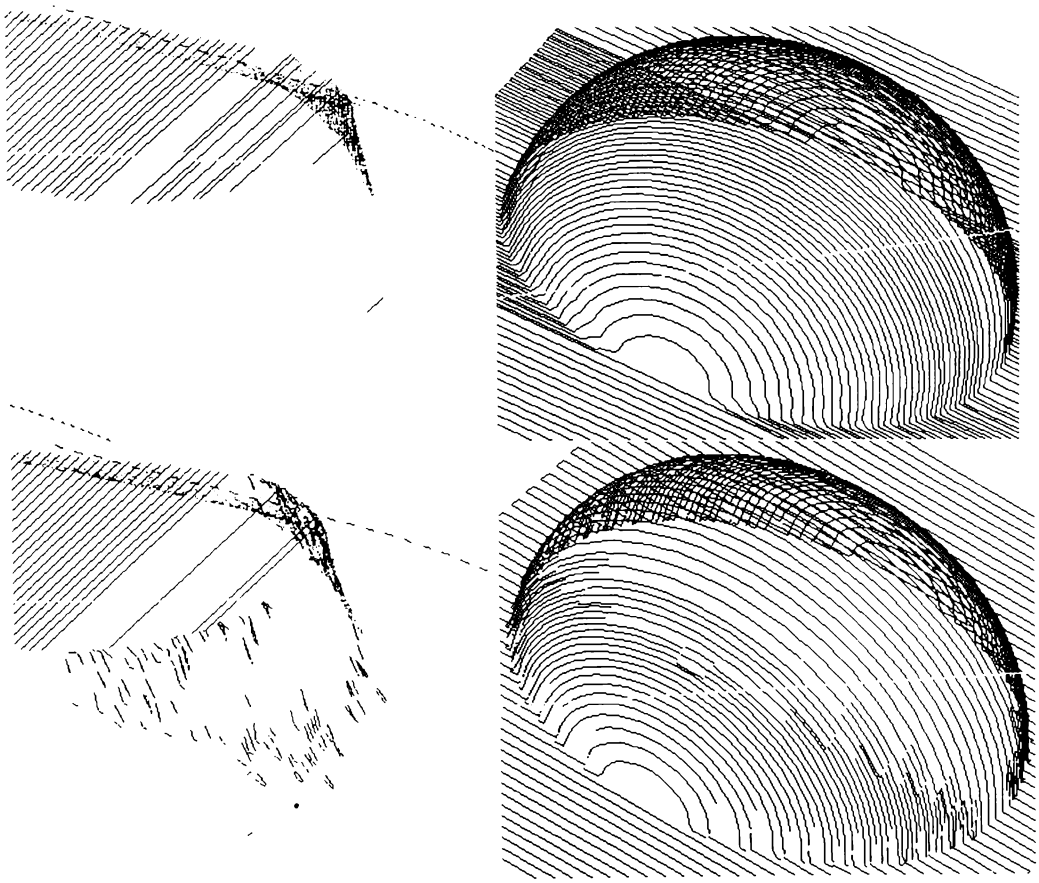


Figura 6.29 *Trasee de sculă cu varierea pasului*
a, b) dinamic, pe toată lungimea;
c, d) inteligent, doar în locurile unde este necesar.

6.7. Calculul reţelelor de difracţie

Reţelele de difracţie îşi găsesc utilitatea, în particular, pentru realizarea elementelor optice şi, în general, în orice altă metodă de a grava. Reţele de difracţie au rolul de a crea lumină difuză din lumină directă. Această metodă are aplicaţie la realizarea farurilor din spate de la autovehicole, unde nu este permisă transmiterea luminii directe.

Problema se pune în felul următor: se cere a genera o reţea patrulateră care să acopere o suprafaţă complexă, şi care să aibă celulele relativ egale. Este clar că nu se poate genera o reţea patrulateră perfect echidistanţată care să acopere orice suprafaţă şi să aibă celule egale, scopul este doar estetic ca celulele să fie relativ armonios poziţionate, considerentele funcţionale fiind realizate deja de prezenţa "bulinelor" din nodurile reţelei.

În acest domeniu, autorul a încercat să proiecteze o metodă generală de "bulinare", cu care, dându-se o suprafaţă complexă şi două curbe de ghidare, să se umple suprafaţa cu o reţea cât mai uniformă.

Algoritmul poate fi sintetizat în felul următor:

- ⇨ *se dau două curbe, o SD şi o rază a bulinei, doi paşi de parcurgere pentru cele două curbe;*
- ⇨ *se calculează SDCS cu o sculă sferică cu raza egală cu raza bulinei;*
- ⇨ *se mulează o curbă pe SDCR (curba directoare);*
- ⇨ *se multiplică cealaltă curbă şi se aplică pe curba directoare (curbele generatoare);*
- ⇨ *se mulează pe SDCR curbele generatoare;*
- ⇨ *se parcurg generatoarele, începând de la intersecţia lor cu curba directoare, cu pasul curbei generatoare constant, spre stânga şi spre dreapta;*
- ⇨ *se parcurge în zig-zag reţeaua de puncte rezultantă şi se crează fişierul NC.*

Cele enumerate pot fi observate în figura următoare.

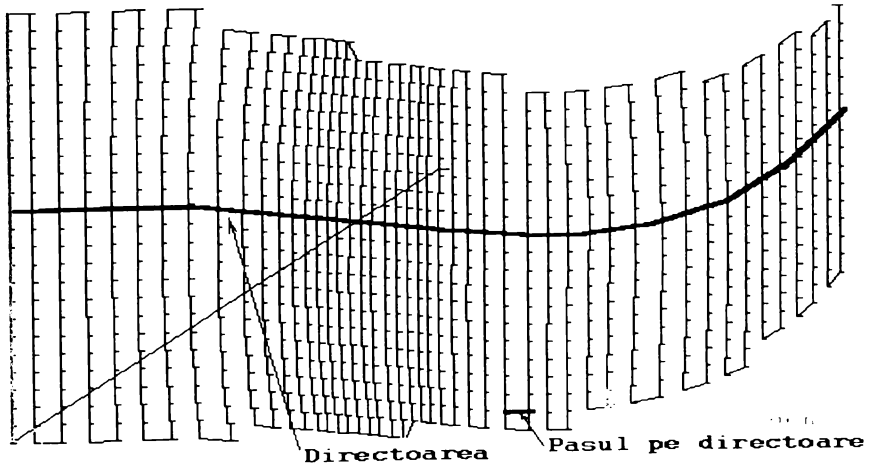


Figura 6.30 *Concepte introductive pentru realizarea reţelei de difracţie.*

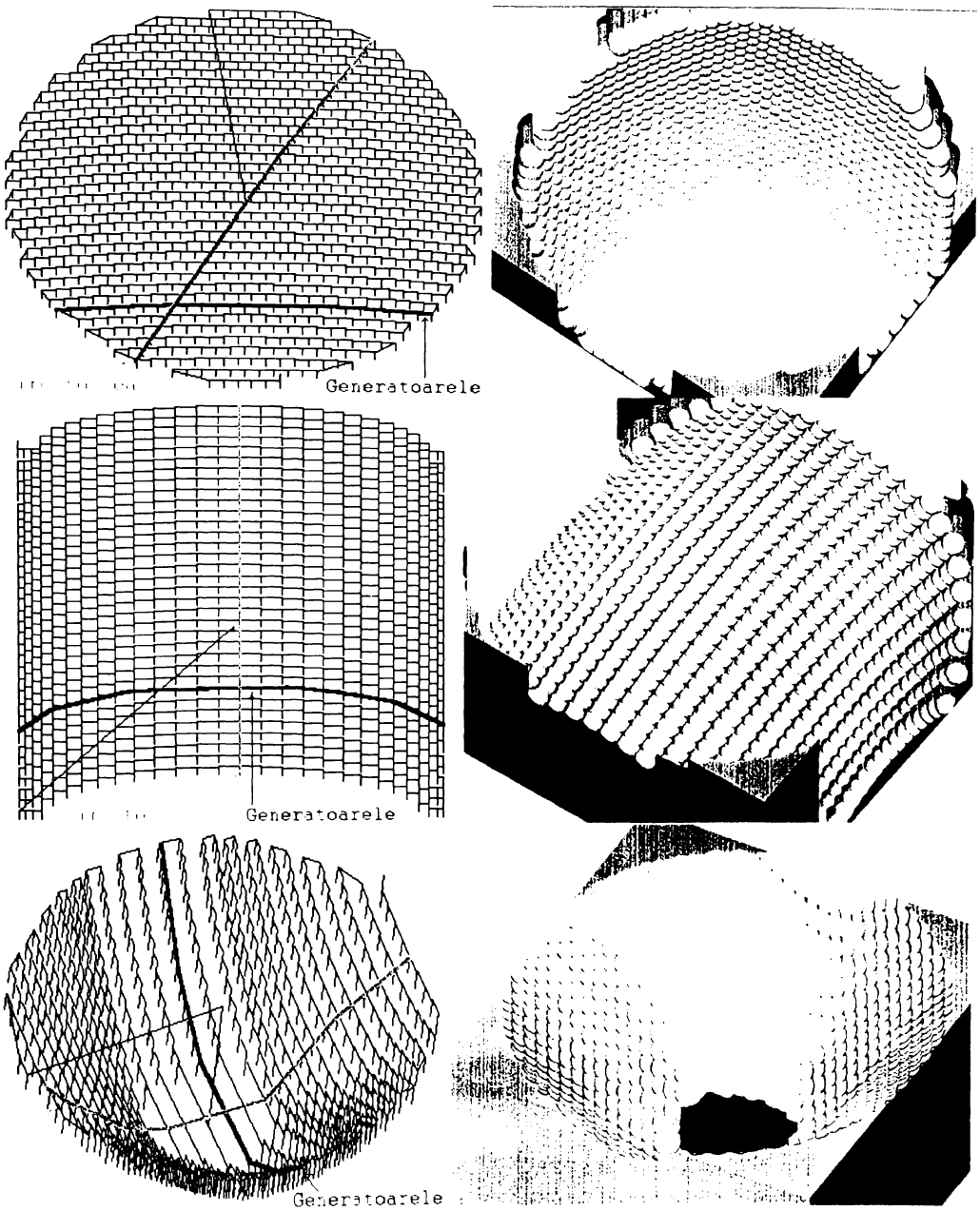


Figura 6.31 *Trei seturi de curbe necesare realizării reţelei de difracţie, pentru trei repere practice. Coloana a II-a reprezintă calitatea şi amplasamentul fiecărei buline.*

Algoritmul în pseudocod este destul de lung şi nu va fi expus. Din exemplificare se poate observa că toţi algoritmii implicaţi au fost expuşi. Se vor prezenta, în continuare, trei exemple reale executate de autor la **SC ELBA SA**, care reprezintă:

a şi b - un far de bicicletă, în care cele două curbe sunt drepte, unghiul dintre ele este de 60^0, iar pasul este egal pe cele două direcţii, curbele fiind tăiate după un cerc, în XY;

c şi d - o suprafaţă mai complexă, în care se propune o reţea rectangulară cu laturi inegale, pe o suprafaţă cu curvaturile variabile în cele două direcţii de frezare;

e şi f – un reper teoretic, complex, cu schimbări rapide de curvatură (discontinuităţi ale derivatei întâi), reţea rectangulară în mască; curbele sunt tăiate după două cercuri concentrice.

Timpii de frezare au fost sub o oră, pentru fiecare reper, o reducere semnificativă, luând în considerare că, practic la o singură suprafaţă, prin metodele normale, se lucra **manual** (neexistând un aparat matematic) mai mult de o lună (într-un schimb), iar calitatea reţelei a fost mult superioară în algoritmul prezentat. Ultima suprafaţă este aproape de realizat manual.

Pentru a putea analiza calitatea reţelei obţinute au fost simulate cele două fişiere şi au fost prezentate în figura 6.31, c şi d.

6.8. Metode de vizualizare a curbelor

Acest subcapitol se va ocupa, în excusivitate, cu diferite metode de a vizualiza, în general, orice curbă sau familie de curbe, în particular, trasee de sculă. Necesitatea vizualizării curbelor (traseelor de sculă) în diferite culori a fost multă vreme una din dorinţele nesatisfăcute ale utilizatorilor de sisteme de fabricaţie, care şi-au dorit un mod de reprezentare a acestora într-o manieră cât mai uşor şi rapid de interpretat . Această necesitate a fost observată de către autor la începutul anilor '90, şi rafinată pe parcurs. Multe din aceste metode folosesc paleta standard de analiză spectrală (curcubeul), începutul fiind reprezentat de albastru, mijlocul de verde şi sfârşitul de roşu. De-a lungul lucrării au fost folosite, fără explicaţii, diferite metode de a reprezenta curbele. Acestea ar putea fi sortate în această ordine:

- ⇨ *toate traseele specifice unei scule sunt desenate într-o culoare (tehnică uzuală sistemelor de fabricaţie actuale);*
- ⇨ *fiecare traseu de sculă este desenat în paleta spectrală - pentru a ne forma o idee despre* începutul, ordinea şi sfârşitul *unui ciclu;*
- ⇨ *fiecare traseu de sculă care începe cu o mişcare de lucru este desenat în paleta spectrală - pentru a ne forma o idee despre* începutul, ordinea şi sfârşitul *fiecărui traseu elementar dintr-un ciclu (util pentru spiralizări, analize de rugozitate);*
- ⇨ *curbele sunt desenate în degradé în paleta spectrală, folosind ca parametru axa X, Z, Y sau variaţia unui alt parametru:* unghiul de variaţie al normalei, eforturile radiale, longitudinale, mixte, materialul nefrezat, rugozitatea *etc., normate în [0...1];*

Suplimentar se poate schimba culoarea (sau numai intensitatea nuanţei respective), obţinându-se informaţii despre cantitatea de vectori din curba respectivă (*toleranţă, ştergerea punctelor coliniare, curvatură etc.*).

Un exemplu complex, în care s-au utilizat câteva din tehnicile expuse de reprezentare a variaţiei rugozităţii (albastru - bună, verde, roşu - critică), compensarea acesteia printr-un ciclu inteligent de frezare şi cantitatea de vectori (varierea intensităţii în noduri) poate fi observată pe următorul reper (un om de ciocolată pentru Crăciun), scanat cu un scaner laser în Germania, care a fost procesat numai folosind algoritmi prezentaţi de-a lungul tezei.

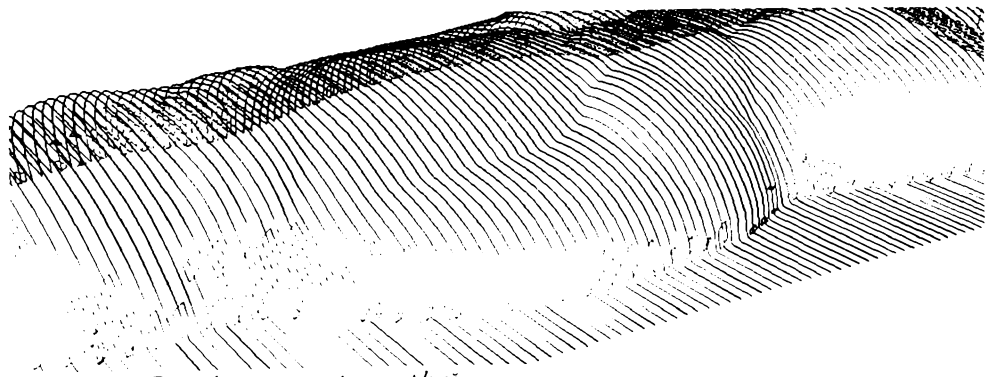


Figura 6.32 *Reper într-o reprezentare complexă.*

Alte exemple de reprezentare, mai simple, se pot observa în figurile următoare.

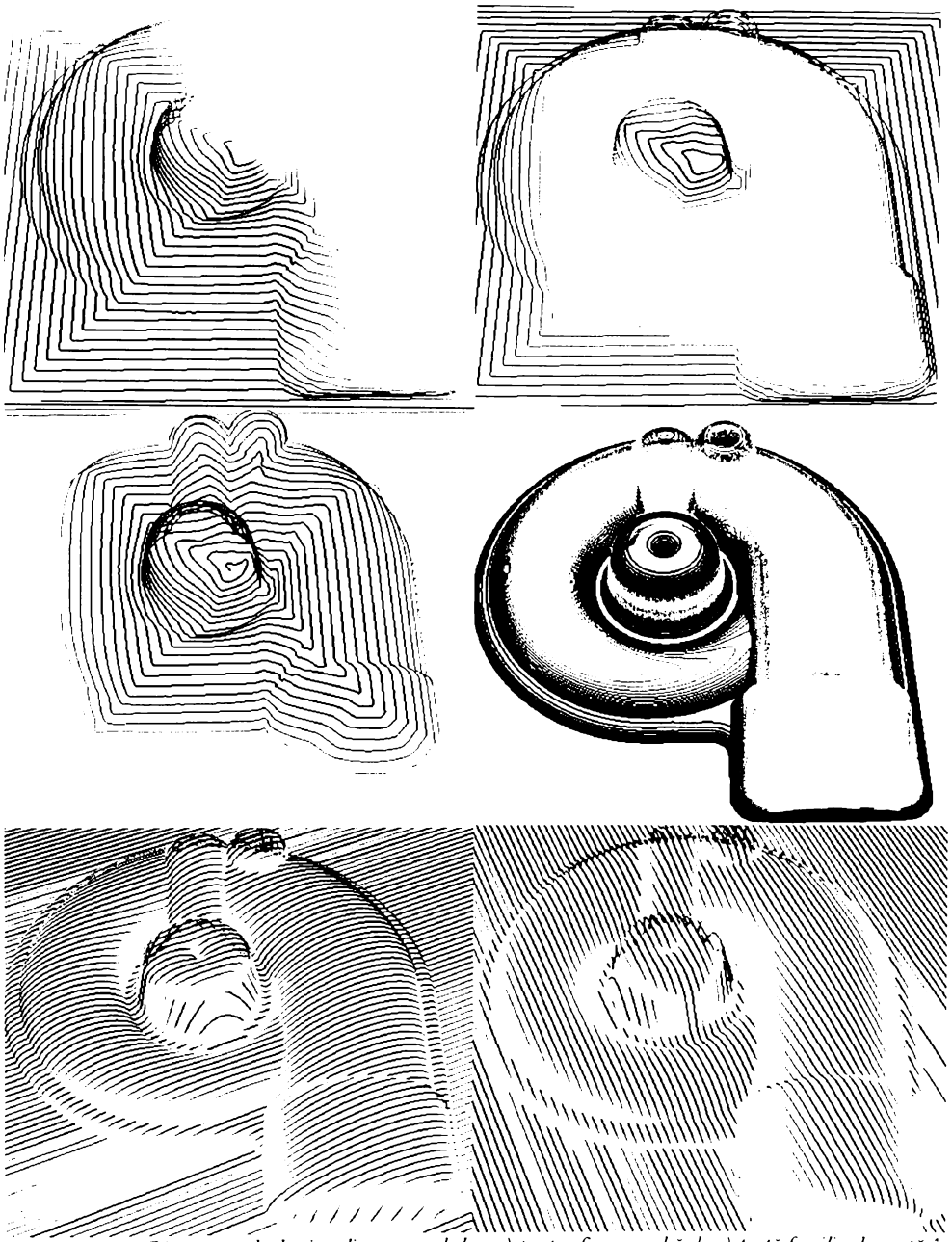


Figura 6.33 *Câteva metode de vizualizare a curbelor. a) pentru fiecare curbă; b, c) toată familia desenată în degradé; d) degradé variind axa Z; e, f) degradé după rugozitatea obținută*

6.9. Concepte introduse

```
Clasa SuprafaţăDiscretă DerivatăDin BazăDiscretă
{

//Concepte introduse în celelalte capitole

    Nimic TransfDetectPtZ(SuprafaţăDiscretă sdAnaliză);
    Nimic DetectăCurbeleZ(FamiliaDeCurbe fcOut, Real rStep = 1.0);

    Nimic TransfDetectează PentruPlanulXY(SuprafaţăDiscretă sdAnaliză, Real rUnghi = 45.0);
    Nimic Detectează CurbeleÎnXY(FamiliaDeCurbe fcOut, Real rStep = 1.0, rUnghi = 45.0);

    Nimic TransfDetectează SuprafaţaPlană(SuprafaţăDiscretă sdAnaliză);
    Nimic Detectează SuprafaţaPlană(FamiliaDeCurbe fcOut, Real rPentruZ = 45.0);

    Nimic TransfEchirugozitate(SuprafaţăDiscretă sdAnaliză, Întreg nMetoda);
    Nimic DetectăEchirugozitate(FamiliaDeCurbe fcOut, Întreg nMetoda = ct_nEchirugozitateXY, Real rPasul = 1.0);
    Nimic TransfNefrezabil(SuprafaţăDiscretă sdNefrezabil, Întreg nMetoda);
    Nimic DetectăNefrezabil(FamiliaDeCurbe fcNefrezabil, SuprafaţăDiscretă sdReper, sdSculaAnal, sdSculaMică,
    Boolean bDetectPlan = FALS);

    Nimic SpiraleleLuiBillator(FamiliaDeCurbe fcOut, Real rStep = 1.0); //transformă SDC în spirale
} //SuprafaţăDiscretă


Clasa Curbă DerivatăDin BazăVectorială
{

//Concepte introduse în celelalte capitole

    Nimic Roteşte(Întreg nDeCâteOri); //roteşte o curbă cu nDeCâteOri
    Real AriaCuSemn(); //returnează aria cu semn pentru a detecta direcţia
    Nimic FăTrigonometric(); //transformă o curbă în sens trigonometric
    Nimic FăOrar(); //transformă o curbă în sens orar
} //Curbă


Clasa FamiliaDeCurbe DerivatăDin BazăVectorială
{

//Concepte introduse în celelalte capitole

    Nimic OptimizazăFHTSP(Întreg nRezolvăComplet = 40);
    Nimic FăTrigonometric(); transformă curbele în sens
    Nimic FăOrar(); transformă curbele în sens orar
    Nimic Roteşte(); roteşte curbele până au punctul de start minim
    Nimic Spirală(Întreg nPuncte = 100); transformă curbele în spirale

} FamiliaDeCurbe
```

6.10. Concluzii

În acest capitol, intitulat "**Metode de analiză şi optimizare**", s-a încercat să se cuprindă câteva dintre cele mai importante aspecte legate de analiza şi generarea optimizată a codului NC pentru fabricarea suprafeţelor discrete pe MUCN, precum şi câteva tehnici de verificare şi simulare.

Cum toate ***optimizările, generările de cod NC, verificările şi simulările*** sunt tehnici şi metode de ***analiză***, în mod natural ele şi-au găsit locul în acest capitol. S-a preferat gruparea tuturora la un loc, deoarece între multe dintre acestea există o interpendenţă ascunsă (d.p.d.v. matematic) şi se consideră că după expunerea capitolului, sortată din punct de vedere logic, al complexităţii şi asemănării aparatelor matematice, se poate observa asemănarea unor tehnici de ***analiză*** a suprafeţei (*zone plane, material nefrezat etc.*) cu tehnicile de ***generare a codului NC*** (*echidistante, echirugozitate etc.*). Tehnicile de ***verificare*** sunt aceleaşi, într-o foarte mare proporţie, cu cele de ***simulare***.

Cum lucrarea are un caracter primordial de sinteză, s-a încercat gruparea logică, matematică, în detrimentul celei funcţionale, nedorindu-se pierderea relaţiei dintre tehnicile de analiză.

În acest capitol s-au introdus şi expus trei mari aspecte ale analizei: ***metode de analiză a SD şi generarea optimizată de cod NC***, câteva ***optimizări posibile ale codului NC în cazul traseelor echidistante*** şi ***calculul reţelei de difracţie pentru elementele optice***.

S-au prezentat, în premieră, câteva metode noi, precum:

- ⇨ *calculul zonelor plane;*
- ⇨ *calculul zonelor critice la frezarea secţiunilor paralele în planul XY;*
- ⇨ *calculul materialului nefrezabil;*
- ⇨ *calculul curbelor de egală rugozitate;*
- ⇨ *metode pseudoadaptive de variere a avansului şi corecţiei de uzură în timp real;*
- ⇨ *minimizarea mişcărilor în avans rapid;*
- ⇨ *spiralele lui Billator;*
- ⇨ *o metodă nouă de optimizare a traseelor echidistante, prin dublarea sau triplarea locală;*
- ⇨ *o metodă de rezolvare a reţelelor de difracţie;*

Pe lângă prezentarea contribuţiilor autorul au fost expuse şi câteva metode clasice de generare de cod, considerându-se ca elemente de noutate metodele de generare ale lor (***curbelor echidistante în XY şi Z***) utilizând SD în acest domeniu.

Alte concepte, care sunt enumerate şi exemplificate pe parcursul capitolului, cum ar fi: ***interpolările superioare, eliminarea punctelor de inflexiune, eliminarea punctelor coliniare*** au fost expuse doar cu scopul secundar de a da consistenţă şi calitate unei eventuale generări de cod NC.

7. Concepte introduse

```
//----- Clasele utilizate pe parcursul lucrării ----------------------------------

//Clasa int utilizată pentru operațiile cu numere întregi
//prefix Variabile: n
Clasa Întreg
{
    operator +, -, *, /, &, |, %, <, >, >=, <=, =, ==, +=, -=, *=, /=, &=, |=, %=, ŞI, SAU, %%, !, min, max; //operatorii şi
    conversiile din tipul real
    Nimic Min (Întreg nA, nB); //întoarce minimul dintre valorile nA şi nB
    Nimic Max (Întreg nA, nB); //întoarce maximul dintre valorile nA şi nB
    Nimic Abs (Întreg nA); //întoarce valoarea absolută
} //EndInt

//funcții prietene
Întreg Min (Întreg nA, nB); //întoarce minimul dintre valorile nA şi nB
Întreg Max (Întreg nA, nB); //întoarce maximul dintre valorile nA şi nB
Întreg Abs (Întreg nA); //întoarce valoarea absolută


//Clasa bool utilizată pentru operațiile logice
//prefix Variabile: b
Clasa Boolean DerivatăDin Întreg
{} //Boolean


//Clasa real utilizată pentru operațiile cu numere reale
//prefix Variabile: r
Clasa Real
{
//operatorii
    operator +, -, *, /, +=, -=, *=, /=, <, >, <=, >=, =, ==, !=;

//funcții membru
    Nimic Min (Real rA, rB); //întoarce minimul dintre valorile rA şi rB
    Nimic Max (Real rA, rB); //întoarce maximul dintre valorile rA şi rB
    Nimic Morf(Real rA, rB, rP = 0.5); //întoarce rA * rP + rB * (1.0 - rP)
    Nimic Abs (Real rA); //întoarce valoarea absolută

    Nimic RotX(Punct pRot, Real rUnghiDeg); //rotația în jurul lui X
    Nimic RotY(Punct pRot, Real rUnghiDeg); //rotația în jurul lui Y
    Nimic RotZ(Punct pRot, Real rUnghiDeg); //rotația în jurul lui Z
    Nimic Scalare(Punct pScalare, Real rFactorScalare); //scalarea față de un punct
    Nimic Translație(Real rTranslație); //scalarea față de un punct
//suplimentar se pot introduce orice funcții matematice Ex sin, cos, etc!!
} //Real

//funcții prietene
Real Min (Real rA, rB); //întoarce minimul dintre valorile rA şi rB
Real Max (Real rA, rB); //întoarce maximul dintre valorile rA şi rB
Real Morf(Real rA, rB, rP = 0.5); //întoarce rA * rP + rB * (1.0 - rP)
Real Abs (Real rA); //întoarce valoarea absolută
Real RotX(Punct pRot, Real rUnghiDeg); //rotația în jurul lui X
Real RotY(Punct pRot, Real rUnghiDeg); //rotația în jurul lui Y
Real RotZ(Punct pRot, Real rUnghiDeg); //rotația în jurul lui Z
Real Scalare(Punct pScalare, Real rFactorScalare); //scalarea față de un punct
Real Translație(Real rTranslație); //scalarea față de un punct


//Clasa punct utilizată pentru operațiile cu curbe şi plase
//prefix Variabile: p
```

```
Clasa Punct
{
 //variabile locale
    Real rX, rY, rZ; //coordonatele punctului în R3

 //operatorii
    operator +, -, *, /, =, ==, +=, -=, *=, /=, min ,max, morf;

//toate funcţiile următoare suportă orice combinaţie de puncte şi reali !!
    Nimic Min (Punct pA, pB); //întoarce minimul dintre valorile pA şi pB
    Nimic Max (Punct pA, pB); //întoarce maximul dintre valorile pA şi pB
    Nimic Morf(Punct pA, pB, pP = 0.5); //întoarce pA * pP + pB * (1.0 - pP)
    Nimic Abs (Punct pA); //întoarce valoarea absolută

    Nimic RotX(Punct pRot, Real rUnghiDeg); //rotaţia în jurul lui X
    Nimic RotY(Punct pRot, Real rUnghiDeg); //rotaţia în jurul lui Y
    Nimic RotZ(Punct pRot, Real rUnghiDeg); //rotaţia în jurul lui Z
    Nimic Scalare(Punct pScalare, Real rFactorScalare); //scalarea faţă de un punct
    Nimic Translaţie(Punct pTranslaţie); //scalarea faţă de un punct
//suplimentar se pot introduce orice funcţii matematice Ex sin, cos, etc!!
} //Punct

 //funcţii prietene
Punct Min (Punct pA, pB); //întoarce minimul dintre valorile pA şi pB
Punct Max (Punct pA, pB); //întoarce maximul dintre valorile pA şi pB
Punct Morf(Punct pA, pB, pP = 0.5); //întoarce pA * pP + pB * (1.0 - pP)
Punct Abs (Punct pA); //întoarce valoarea absolută

Punct RotX(Punct p, pRot, Real rUnghiDeg); //rotaţia în jurul lui X
Punct RotY(Punct p, pRot, Real rUnghiDeg); //rotaţia în jurul lui Y
Punct RotZ(Punct p, pRot, Real rUnghiDeg); //rotaţia în jurul lui Z
Punct Scalare(Punct p, pScalare, Real rFactorScalare); //scalarea faţă de un punct
Punct Translaţie(Punct p, pTranslaţie); //scalarea faţă de un punct


 //Clasa BazăVectorial utilizată pentru operaţiile cu curbe şi plase în spaţiu
 //prefix Variabile: bv
Clasa BazăVectorial
{
    Punct pMin; //dreptunghiul înconjurător punctul de start
    Punct pMax; //dreptunghiul înconjurător punctul de sfârşit
    Punct pDif; //pMax - pMin
} //BazăVectorial


 //Clasa curbă utilizată pentru operaţiile cu curbe
 //prefix Variabile: c
Clasa Curbă DerivatăDin BazăVectorial
{
 //variabile locale
    Întreg nNr; //numărul de puncte (vertex) în R3

  metode
    Punct Operator[](Întreg nNr);   întoarce punctul numărul nNr   [1..nMax]
    Punct Operator[](Real rT);   întoarce punctul normând curba rT   [0.0..1.0]
    Punct Operator[](Întreg nNr, Real rT);   întoarce punctul normând elementul nNr
    Nimic PuncLa(Întreg nNr, Punct p);   setează punctul numărul nNr   [1..nMax]
    Punct Add(Punct p);   adună un punct
    Punct Del(Întreg nNr);   şterge punctul numărul nNr   [1..nMax]
    Punct Add(Întreg nNr, Curbă c);   adună o curbă la poziţia dată de nNr
    Punct Add(Curbă c);   adună o curbă la sfârşit
```

```
    Nimic SchimbăSensul(); //schimbă sensul de parcurgere al curbei
    Nimic FăTrigonometric(); //face sensul de parcurs al curbei în sens trig.
    Nimic FăOrar(); //face sensul de parcurs al curbei în sens orar.
    Boolean EsteÎnSensTrig(); //întoarce ADEVĂRAT dacă toate sunt în sens trig.
    Boolean EsteÎnSensOrar(); //întoarce ADEVĂRAT dacă toate sunt în sens orar
    Boolean EsteÎnchisă(); //întoarce ADEVĂRAT dacă toate sunt închise

    Nimic Roteşte(Întreg nDeCâteOri) //roteşte o curbă cu nDeCâteOri
    Real AriaCuSemn() //returnează aria cu semn pentru a detecta direcţia
} //Curbă


//Clasa plasă utilizată pentru operaţiile cu suprafeţe
//prefix Variabile: l
Clasa Plasă DerivatăDin BazăVectorial
    Punct Operator[] (Întreg nU, nV); //întoarce pt. nU=[1..nNrU], nV =[1..nNrV]
    Punct IaDeLaUV(Real rU, rV); //întoarce normând plasa rU,rV = [0.0..1.0]
    Curbă IaDeLaXY(Real rX, rY); //proiectează şi returnează o listă cu pt.
    Punct Operator[] (Întreg nU, nV, Real rU, rV); //normează elem. NU, nV
    Nimic PuneLa(Întreg nU, nV, Punct p); //setează punctul de la nU, nV
    Nimic SchimbăSensul(Boolean bU, bV); //schimbă sensul de parcurgere al plasei

    FamiliaDeCurbe IaDeLaXY(Întreg nPlasa, Real rX, rY); //proiectează şi returnează o listă
    FamiliaDeCurbe IaDeLaU(Real rV); //converteşte în curbă pt rV
    FamiliaDeCurbe IaDeLaV(Real rU); //converteşte în curbă pt rU
    FamiliaDeCurbe IaDeLaX(Real rX); //converteşte în curbă pt rX
    FamiliaDeCurbe IaDeLaY(Real rY); //converteşte în curbă pt rY
    FamiliaDeCurbe IaDeLaZ(Real rZ); //converteşte în curbă pt rZ

    Real Aria(); //calculează aria
    Real LungimeaU(Real rV); //lungimea la în dir U la v = rV
    Real LungimeaV(Real rU); //lungimea la în dir V la u = rU
    Real LungimeaMaxU(); //lungimea maximă în dir U
    Real LungimeaMaxV(); //lungimea maximă în dir V
} //Plasă


//Clasa de bază pentru suprafeţele discrete de tip real şi mască
//prefix Variabile: bd
Clasa BazăDiscret DerivatăDin BazăVectorial
{
    Întreg nX; //numărul de puncte din matrice în direcţia X
    Întreg nY; //numărul de puncte din matrice în direcţia Y
    Real rStep; //pasul reprezentării

    Real IndexÎnRealPtX(Întreg nX); //conversia din index in real pe X
    Real IndexÎnRealPtY(Întreg nY); //conversia din index in real pe Y
    Întreg RealÎnIndexPtX(Real rX); //conversia din real in index pe X
    Întreg RealÎnIndexPtY(Real rY); //conversia din real in index pe Y
} //BazăDiscret


//Clasa Mască
//prefix m
Clasa MascăDiscretă DerivatăDin BazăDiscret
{
    operator &, |, -, &=, |=, -=, ;

//citire scriere date
    Nimic PuncLa(Întreg nX, Întreg nY, Boolean bVal, Întreg nMetoda = PM_NONE); //pune un bool inteligent
    Boolean Operator[](Întreg nX, Întreg nY); //citeşte un bool
    Nimic PuncLa(Real rX, Real rY, Boolean bVal, Întreg nMetoda = PM_NONE); //pune un punct
```

```
    Boolean Operator[](Real rX, Real rY); //citeşte un bool
    Nimic PuneLaAll(Boolean); //setează toată suprafaţa
    Nimic Revert(); //schimbă toate valorile
} //MascăDiscretă


//Clasa suprafaţa discretă
//prefix s
Clasa SuprafaţăDiscretă DerivatăDin BazăDiscret
{
    MascăDiscretă m; //masca asociată

    operator +, -, *, /, +=, -=, *=, /=, =, MAX, MIN, ADD, SUB, MULTIPLY, DIVIDE, MORPH, FILSUS, FILJOS,
    ECDSUS, ECDJOS;

//citire scriere date
    Nimic PuneLa(Întreg nX, Întreg nY, Real rVal, Întreg nMetoda = PM_NONE, Real rMorph = 0.5);
    Nimic PuneLa(Real rX, Real rY, Real rVal, Întreg nMetoda = PM_NONE, Real rMorph = 0.5);
    Real Operator[] (Întreg nX, Întreg nY); //citeşte un punct
    Real Operator[] (Real rX, Real rY); //citeşte un punct
    Nimic PuneLaAll(Real rValoare); //seteaza toată suprafaţa

    Nimic Combină(SuprafaţăDiscretă sdComb, Întreg nModulDePunere = PM_NONE, Boolean bTotul =
    ADEVĂRAT);
    Nimic PuneLaOrizontal(Real rValoare, Întreg nModulDePunere = PM_NONE, Boolean bTotul = ADEVĂRAT);
    Nimic PuneLaPlan3P(Punctp1, p2, p3, Întreg nModulDePunere = PM_NONE, Boolean bTotul = ADEVĂRAT);
    Nimic PuneLaFuncţie(SetDeCaractere strFuncţia, Întreg nModulDePunere = PM_NONE, Boolean bTotul =
    ADEVĂRAT);

    Nimic PuneLaInterpolareXDir(SuprafaţăDiscretă sdSecţiuni, Întreg nOrdinCurbei= 1, Întreg nModulDePunere =
    PM_NONE, Boolean bTotul = ADEVĂRAT);
    Nimic PuneLaInterpolareYDir(SuprafaţăDiscretă sdSecţiuni, Întreg nOrdinCurbei= 1, Întreg nModulDePunere =
    PM_NONE, Boolean bTotul = ADEVĂRAT);

    Nimic Interpolează PrinPuncte(Curbă cListaCuPuncte, FamiliaDeReali lrListaCuPonderi, FamiliaDeCurbe
    fcAlteCurbe, FamiliaDeReali frPonderiPtCurbe, FamiliaDeReali frPaşiiPtCurbe, Real rRigiditatea);

    Nimic InitCapDeSculă(Curbă cCapDeSculă, Real rPas);
    Nimic InitÎnfăşurătoareStatic(SuprafaţăDiscretă sdSupafaţa, sdSculă, Boolean bDirecţiaSus, bTotul);
    Nimic InitRacordareStatic(SuprafaţăDiscretă sdSupafaţa, sdSculă, Boolean bDirecţiaSus, bTotul);
    Nimic InitÎnfăşurătoareDinamic(SuprafaţăDiscretă sdSupafaţa, sdScalare, sdSculă, Boolean bDirecţiaSus,
    Boolean bTotul);
    Nimic InitRacordareDinamic(SuprafaţăDiscretă sdSupafaţa, sdScalare, sdSculă, Boolean bDirecţiaSus, Boolean
    bTotul);
    Nimic InitPlasă(CPLasa llIn); //importul unei plase
    Nimic InitPlase(FamiliaDePlase flIn); //importul unei liste cu plase

    Nimic PuneLaSuprafaţaSuperioară(SuprafaţăSuperioară sS, Real rUStep, rVStep);
    Nimic PuneLaSuprafeţeSuperioare(FamiliaDeSuprafeţeSuperioare fsFS);

//conversii şi proiecţii
    Nimic CreazăCurbe(FamiliaDeCurbe fcOut, Real rPentruZ);
    Nimic CreazăToateCurbele(FamiliaDeCurbe fcOut, Real rZStep = 1.0, Boolean bAuto = ADEVĂRAT, Real
    rZStart, rZEnd);

    Nimic ProiecteazăCurbă(Curbă cProiectează);
    Nimic ProiecteazăCurbe(FamiliaDeCurbe fcProiectează);

    Nimic ProiecteazăCurbă(Curbă cProiectează, Real rPas = 0.1); //cu pas constant
    Nimic ProiecteazăCurbe(FamiliaDeCurbe fcProiectează, Real rPas = 0.1); //pas ct

    Nimic ExportDXF(Întreg nCuloare = 15, SetDeCaractere strLayer = "SD ieşire")
```

```
//câteva filtre
    Întreg CreazăMască2();
    Întreg CreazăMască3();
    Întreg CreazăMască4();
    Întreg CreazăMască(Întreg nMaxPuncte);

    Întreg CreazăValoare2(Real rZ);
    Întreg CreazăValoare3(Real rZ);
    Întreg CreazăValoare4(Real rZ);
    Întreg CreazăValoare(Întreg nMaxPuncte, Real rZ);

    Întreg DistrugeMască2();
    Întreg DistrugeMască3();
    Întreg DistrugeMască4();
    Întreg DistrugeMască(Întreg nMaxPuncte);

    Întreg DistrugeValoare2(Real rZ);
    Întreg DistrugeValoare3(Real rZ);
    Întreg DistrugeValoare4(Real rZ);
    Întreg DistrugeValoare(Întreg nMaxPuncte, Real rZ);

//Metode utilizate de automatul neuronal la importul familiilor de Supr. sup.
    Nimic InitPuneLaR();
    Nimic DonePuneLaR();
    Nimic PuneLaR(PunctpP, Real rTaieSus = 2.0); //antrenează cei patru vecini

//trasnformări, analize şi optimizări ale generării de fişier NC
    Nimic TransfDetectPtZ(SuprafaţăDiscretă sdAnaliză);
    Nimic DetectăCurbeleZ(FamiliaDeCurbe fcOut, Real rStep = 1.0);

    Nimic TransfDetectează PentruPlanulXY(SuprafaţăDiscretă sdAnaliză, Real rUnghi);
    Nimic DetectăzăCurbeleÎnXY(FamiliaDeCurbe fcOut, Real rStep = 1.0, rUnghi = 45.0);

    Nimic TransfDetectează SuprafaţaPlană(SuprafaţăDiscretă sdAnaliză);
    Nimic Detectează SuprafaţaPlană(FamiliaDeCurbe fcOut, Real rPentruZ = 45.0);

    Nimic TransfEchirugozitate(SuprafaţăDiscretă sdAnaliză, Întreg nMetoda);
    Nimic DetectăEchirugozitate(FamiliaDeCurbe fcOut, Întreg nMetoda = ct_nEchirugozitateXY, Real rPasul = 1.0)

    Nimic TransfNefrezabil(SuprafaţăDiscretă sdNefrezabil, Întreg nMetoda)
    Nimic DetectăNefrezabil(FamiliaDeCurbe fcNefrezabil,
    SuprafaţăDiscretă sdReper, sdSculaAnal, sdSculaMică, Boolean bDetectPlan = FALS);

    Nimic SpiralaLuiBillator(FamiliaDeCurbe fcOut, Real rStep = 1.0);
} //SuprafaţăDiscretă


Clasa Fişier //obiect necesar fişierelor de export  import
{
    Nimic Deschide(SetDeCaractere strNume);  deschide sau crează un fişier
    Nimic Închide(); //închide un fişier
    Nimic Scrie(SetDeCaractere strOut);  scrie un SetDeCaractere în fişier
    Nimic Citeşte(SetDeCaractere strIn);  citeşte un SetDeCaractere din fişier
} //SuprafaţăDiscretă


Clasa DXFExport  o rudimentare a formatului de export DXF care suportă curbe şi plase
{
    Fişier fişier;  fişierul de export

    Nimic Iniţializează(SetDeCaractere strNume);
```

```
    Nimic Sfârşeşte();

    Nimic InitCurbă(Întreg nCuloare = 15, SetDeCaractere strLayer = "SD ieşire");
    Nimic DoneCurbă();

    Nimic InitPlasă(Întreg nU, nV, nCuloare = 15, SetDeCaractere strLayer = "SD ieşire");
    Nimic DonePlasă();

    Nimic Exportă(Punct pIn);
} //DXFExport


Clasa STLExport //o rudimentare a formatului de export STL care suportă doar ASCII Export
{
    Fişier fişier; //fişierul de export

    Nimic Iniţializează(SetDeCaractere strNume);
    Nimic Sfârşeşte();

    Nimic Exportă(Punct p1, p2, p3);
} //CSTLExport


Clasa NCExport //o rudimentare a formatului de export NC
{
    Fişier fişier; //fişierul de export

    Nimic Iniţializează(SetDeCaractere strNume);
    Nimic Sfârşeşte();

    Nimic Exportă(Punct p, Boolean bRapid);
} //NCExport


Enumerare ModulDePunere //enumerarea modurilor de combinare solidă a SD şi măştilor
{ PM_NONE, PM_MAX, PM_MIN, PM_ADD, PM_SUB, PM_MULTIPLY, PM_DIVIDE, PM_MORPH,
  PM_FILSUS, PM_FILJOS, PM_ECDLSUS, PM_ECDJOS, PM_AND, PM_OR, PM_XOR }
```

8. Concluzii

Programarea asistată de calculator a MUCN a cunoscut o dinamică de dezvoltare extraordinară în ultimii ani, favorizată de dezvoltarea explozivă a componentelor hard, apariţia unor limbaje tot mai performante (*C++, Java, Delfi*), a unor sisteme de operare foarte performante (*Linux, BeOS, Windows 95 şi NT, Rhapsody*), a unor sisteme de proiectare asistată orientate pe solide (*SolidWorks, Autodesk Mechanical Desktop, SolidEdge, ProEngineering*) şi a unor sisteme de fabricaţie asistată (*MasterCAM, EdgeCAM, DelCAM, HyperMill, SurfCAM*).

Analizând critic situaţia existentă la momentul actual a rezultat că există un complex de *modele matematice, pachete soft şi resurse tehnologice* care acoperă, mai mult sau mai putin complet, latura fabricaţiei.

Având în vedere cele expuse în capitolul privitor la "**Stadiul actual**", se poate observa că în domeniul *proiectării, stocării, comunicării (importului, exportului, DNC), vizualizării şi simulării,* lucrurile sunt aproape închise, existând colective şi produse foarte performante. Singurele locuri deschise îmbunătăţirilor sunt:

- ***Generarea optimizată a codului NC:***
  - ⇨ *analizarea şi calculul zonelor neprelucrate din cauza interferenţelor sculă - semifabricat;*
  - ⇨ *generarea codului, cu condiţia de frezare la rugozitate constantă,* ceea ce duce la reducerea dramatică a timpilor de prelucrare.
  - ⇨ *detecţia zonelor critice, unde nu se poate asigura rugozitatea impusă, pentru diferite cicluri clasice (echidistante în x, y, z);*
  - ⇨ *analiza şi compensarea uzurilor;*
  - ⇨ *analiza şi compensarea dilatărilor;*
  - ⇨ *generarea codului NC în curbe, cu cât mai puţine inflexiuni,* pentru a preveni oprirea şi schimbarea prea deasă a sensului de rotaţie a motoarelor maşinii unelte, ceea ce duce la şocuri, vibraţii şi previne utilizarea eficientă a codului generat la maşini unelte de turaţii mari, cu avans rapid;
  - ⇨ *reducerea mişcărilor rapide,* prin optimizarea traiectoriilor în ciclurile de finisare, ceea ce duce la avansuri constante şi timpi mai scăzuţi;
  - ⇨ *luarea automată a deciziei* privind care tip de frezare este recomandată pentru un anumit tip de reper, în funcţie de gabarit, înclinaţii, inflexiuni, racordări;
  - ⇨ *deciderea, în cazul existenţei mai multe scule aşchietoare, a setului optim necesar în vederea frezării reperului dat, la rugozitatea cerută, într-un timp cât mai redus;*
- **Simulări inteligente**, *care să menţină în tot timpul generării fişierului NC starea de frezare a semifabricatului. Aceste simulări trebuie să asigure: citirea nivelelor de siguranţă pentru mişcări în avans rapid, cât mai reale, deci cât mai joase, optimizând timpul în mişcări rapide; cunoaşterea în fiecare moment a cantităţii de material care este prelevată în direcţie radială şi frontală, oferind oportunitatea de a pilota inteligent avansul, încărcând cât mai uniform scula şi maşina unealtă şi scăzând timpul necesar prelucrării unui reper.*
- ***Citirea şi convertirea într-un format utilizabil a datelor provenite de la maşini de palpat, scanat şi cartografiat.***
- ***Citirea şi convertirea într-un format utilizabil a datelor provenite din alte sisteme de proiectare şi fabricaţie în format CL sau NC.***
- ***Tehnici de simulare rapidă a fişierelor NC.***
- **Tehnici de verificare,** *cu raportarea zonelor nefrezate şi (eventual) a locurilor unde s-au produs interferenţe;*
- **Experţi** *care să ajute utilizatorul să ia o decizie despre modul optim de frezare a unei suprafeţe.*

În general, acestea sunt şi domeniile studiate de autor în ultimii ani, domenii în care şi-a adus contribuţia în prezenta teză de doctorat.

Teza s-a dorit a fi în primul rând una de sinteză, prezintându-se o metodă ***solidă, consistentă şi generică*** *de stocare, modelare, conversie, import-export, analiză, generare optimizată, simulare şi verificare,* bazată în totalitate pe suprafeţelor discrete uniform riglate (demulabile), punându-se un mare accent pe modul de structurare şi prezentare.

Lucrarea a fost împărţită în patru capitole principale:

- ***"Metode de notaţie şi clase utilizate"***
- ***"Metode de generare şi modelare"***
- ***"Metode de conversie şi formate de import – export"***
- ***"Metode de analiză şi optimizare"***

Această organizare prezintă graduat problemele care apar în ***analiza, designul şi implementarea*** unei noi metodologii de stocare, care se doreşte a fi una viabilă din punctul de vedere al satisfacerii unui număr cât mai mare de necesităţi.

În capitolul intitulat "**Metode de notaţie şi clase utilizate**" s-a prezentat modul de organizare şi structurare a datelor. Nu s-a folosit nici o metodă clasică descrisă în manualele de prezentare a limbajelor orientate pe obiect, ci:

- *S-a descris un limbaj pseudocod, orientat pe obiecte, utilizat în descrierea unificată a modelelor matematice, algoritmilor şi a codului sursă;*
- *S-a încercat crearea unei structuri primare de obiecte care se moştenesc într-o manieră logică şi acoperă într-o mare măsură tipul de structuri de dată necesare acestei lucrări;*
- *S-au pus bazele metodei de stocare a suprafeţelor discrete (SD), precum şi câteva metode de rezolvare a cazurilor în care suprafaţele care se doresc a fi convertite nu sunt discrete. S-a încercat să se cuprindă sumar doar problematica stocării, conversia fiecărui tip particular de dată nefăcând parte din tema acestui capitol.*
- *S-a introdus conceptul de mască, ca loc în care se raportează diferite aspecte legate de modul de procesare al SD.*

În capitolul intitulat "***Metode de generare şi modelare***" au fost descrise tehnici si metode de *creare, modelare, conversie, import şi filtrare* a SD.

Scopul acestui capitol a fost acela ca un număr cât mai mare de repere provenite din diferite sisteme de proiectare sau fabricaţie, reprezentate prin diferite metode sau modelate direct, utilizând metodele descrise, să poată beneficia de tehnicile sofisticate de analiză şi generare a codului NC discutate pe parcursul capitolelor următoare.

De asemenea, au fost prezentate, în premieră, trei metode noi, concepute de către autor:

- *Algoritmul de import şi conversie a seturilor de puncte şi curbe furnizate fără nici o regulă;*

- ⇨ *Algoritmul de calculare a înfăşurătoarei şi racordărilor cu forme de orice geometrie (un caz particular al acestora sunt capetele de sculă suprafeţe de revoluţie utilizate în frezare);*
- ⇨ *Reţeaua neuronală pentru antrenarea cu date care nu cad în punctele reţelei, utilizabilă ca o metodă generică de import a tuturor datelor parametrice.*

A fost prezentată o familie de metode auxiliare, **filtrele**, utilizate pentru reducerea zgomotului introdus de diferite metode de conversie şi analiză, exemplificându-se acest concept.

În capitolul intitulat "**Metode de conversie şi formate de import - export**" au fost introduse metode de: *creare de curbe, proiecţie, offset inteligent şi export în formate simple* ASCII, ca: *DXF, STL, NC, CL.*

Acest capitol, al metodelor de conversie şi export, este de o importanţă capitală în utilizarea suprafeţelor discrete, virtual, în orice sistem de proiectare şi fabricaţie. Ideea prezentării lui a fost aceea de a îmbina prezentarea riguroasă, însoţită de expunerea algoritmilor în pseudocod, cu exemplificarea fiecărui concept introdus, cu ajutorul exemplelor şi a listingului potenţial realizat de către obiectele expuse.

Contribuţiile esenţiale aduse de acest capitol sunt:

- ⇨ *A fost creat un set nou de obiecte specifice fiecărui tip de export în parte:* Fişier, DXFOut, STLOut, NCOut, CLOut. *Aceste clase au fost implementate folosind o metodă unificată de prezentare, încercând să se ascundă detaliile fiecărui format în parte.*
- ⇨ *Au fost create metode noi destinate conversiei şi exportului, metode care dau utilizabilitate SD, legându-le de alte sisteme de proiectare, ca aparate matematice auxiliare de analiză sau conversie în format NC.*
- ⇨ *S-a prezentat, în premieră, un algoritm de conversie în curbe de nivel foarte fin, care permite SD corecţie de sculă să fie convertită în format NC, şi s-au prezentat aplicaţiile lui, potenţial nelimitate, în detectarea formelor, analizarea şi vectorizarea fotografiilor.*
- ⇨ *S-a expus şi exemplificat GNCPP, generator automat de tehnologie, o librărie dinamică complexă, care are scopul de a genera fişier NC specific, virtual, pe orice echipament, optimizat pentru lungime şi timp de rulare.*

În capitolul intitulat generic "**Metode de analiză şi optimizare**" au fost cuprinse câteva dintre cele mai importante aspecte legate de *analiza şi generarea optimizată a codului* ***NC***, pentru fabricarea suprafeţelor discrete pe maşini unelte cu comenzi numerice, precum şi câteva tehnici de verificare şi simulare.

Cum toate *optimizările, generările de cod NC, verificările şi simulările* sunt tehnici şi metode de ***analiză***, natural că ele şi-au găsit locul în acest ultim capitol.

S-a preferat gruparea la un loc, deoarece, între multe dintre acestea există o interpendenţă ascunsă, iar autorul consideră că, după expunerea capitolului, sortată din punct de vedere logic, al complexităţii şi asemănării aparatelor matematice, se poate observa asemănarea unor tehnici de analiză a suprafeţei (*zone plane, material nefrezat etc.*) cu unele tehnici de generare a codului NC (*echidistante, echirugozitate, etc.*). Tehnicile de **verificare** sunt într-o mare proporţie identice cu cele de **simulare**.

Cum lucrarea are un caracter primordial de sinteză, s-a încercat gruparea logică, matematică, în detrimentul celei funcţionale, nedorindu-se pierderea relaţiei dintre tehnicile de analiză.

S-au introdus şi prezentat trei aspecte ale analizei:

- ⇨ *Metode de analiză a SD şi generarea optimizată de cod NC,*
- ⇨ *Câteva optimizări posibile ale codului NC în cazul traseelor echidistante,*
- ⇨ *Calculul reţelei de difracţie pentru elementele optice.*

S-au prezentat, în premieră, câteva metode noi:

- ⇨ *Calculul zonelor plane;*
- ⇨ *Calculul zonelor critice la frezarea secţiunilor paralele în planul XY;*
- ⇨ *Calculul materialului nefrezabil;*
- ⇨ *Calculul curbelor de egală rugozitate;*
- ⇨ *Metode pseudoadaptive de variere a avansului şi corecţiei de uzură în timp real;*
- ⇨ *Minimizarea mişcărilor în avans rapid;*
- ⇨ *Spiralele lui Billator;*
- ⇨ *Metoda de optimizare a traseelor echidistante prin dublarea sau triplarea locală;*
- ⇨ *Metoda de rezolvare a reţelelor de difracţie.*

Pe lângă prezentarea contribuţiilor autorul au fost expuse şi câteva metode clasice de generare de cod, considerându-se ca elemente de noutate metodele de generare a acestora (curbelor echidistante în XY şi Z) utilizând SD în acest domeniu.

Alte concepte enumerate şi exemplificate pe parcursul capitolului, cum ar fi *interpolările superioare, eliminarea punctelor de inflexiune, eliminarea punctelor coliniare* au fost expuse doar cu scopul secundar de a da *consistenţă, generalitate şi calitate* unei eventuale generări de cod NC.

Direcţiile de cercetare abordate în continuare de către autor ţin de integrarea domeniului inteligenţei artificiale în fabricaţie, referindu-se la următoarele aspecte:

- ⇨ *detecţia şi prelucrarea automată a găurilor şi holurilor (feature recognition),*
- ⇨ *poziţia optimă a unui solid în sculă (în vederea unei extracţii simple),*
- ⇨ *optimizarea setului de scule pentru frezarea în timp minim a oricărei geometrii.*
- ⇨ *cicluri complexe automate: degroşare + semifinisare + finisare + "frezare creion" (pencil milling) + cicluri de eliminarea materialului nefrezabil;*

Aceste analize sunt executate, în general, pe solide discrete (faţetate şi demulabile). Câteva rezultate de ultima oră pot fi găsite în prezentările şi analizele de pe CD-ul însoţitor.

Anexe

Anexa A: Clase utilizate

Tip de dată	Prefix	Operatori
Nimic	v	
SetDeCaractere	str	+, -
Întreg	n	+, -, *, /, &, |, %, <, >, =, ==, +=, -=, *=, /=, &=, |=, %=, ŞI, SAU, !, **real;**
Boolean	b	+, -, *, /, &, |, %, <, >, =, ==, +=, -=, *=, /=, &=, |=, %=, ŞI, SAU, !, **real;**
Real	r	+, -, *, /, <, >, =, ==, +=, -=, *=, /=, **int;**
Punct	p	+, -, *, /, =, ==, +=, -=, *=, /=, translaţie, rotaţie, scalare, min ,max;
Curbă (superioare)	c, cs	+, -, *, /, =, ==, +=, -=, *=, /=, translaţie, rotaţie, scalare, min, max;
Plasa (superioare)	l, ls	+, -, *, /, =, ==, +=, -=, *=, /=, translaţie, rotaţie, scalare, min, max;
Masca	m	+, -, *, /, +=, -=, *=, /=, =, ŞI, SAU, XSAU, NOT
Supr. Demulabilă	s	+, -, *, /, +=, -=, *=, /=, =, MAX, MIN, ADD, SUB, MULTIPLY, DIVIDE, MORPH, FILSUS, FILJOS, ECDSUS, ECDJOS
STL Export	stlout	asigură exportul în format STL;
DXF Export	dxfout	asigură exportul în format DXF;
NC Export	ncout	asigură exportul în format NC;
CL Export	clout	asigură exportul în format CL;
Fişier	fişier	un fişier generic fără nici o formatare specială;
FamilieDe*	f*	Reali (fr), Întregi (fn), Booleeni (fb), Puncte Curbe (fc), Plase (fm), etc.

Anexa B: Abrevieri

SD	*Suprafaţă Discretă (demulabilă)*
SDR	*Suprafaţă Discretă (demulabilă) Reper*
SDCS	*Suprafaţă Discretă (demulabilă) Corecţie Sculă*
SDCST	*Suprafaţă Discretă (demulabilă) Corecţie Sculă Teoretică*
SDN	*Suprafaţă Discretă (demulabilă) Nefrezabilă*
SDNT	*Suprafaţă Discretă (demulabilă) Nefrezabilă Teoretică*
SDER	*Suprafaţă Discretă (demulabilă) EchiRugozitate*
SDSC	*Suprafaţă Discretă (demulabilă) sculă*
SDT	*Suprafaţă Discretă (demulabilă) Transformată*
SLB	*Spiralele lui Billator*
ACCE	*Algoritmul de Conversie în Curbe Echipotenţiale*
ASCII	*Format citibil cu un editor de text*
DXF	*Format de import export ASCII pentru repere*
STL	*Format de import export ASCII pentru repere solide reţea triunghiulară*
APT	*Limbaj de programare pentru proiectarea tehnologiei*
CL	*Format standard de export a informaţiilor necesare fabricării unui reper*
NC	*Numerical Control (control numeric)*
ECN	*Echipament cu Comandă Numerică*
CNC	*Control numeric cu calculator intern*
DNC	*Direct Numerical Control (control numeric direct)*
GNC	*Generic Numerical Control (control numeric generic)*
GNCPP	*Generic Numeric Control PostProcessor*
MSD	*Masca Suprafeţei Digitale*
MU	*Maşină-Unealtă*
MUCN	*Maşini-Unelte cu Comenzi Numerice*
TSP	*Traveling Salesman Problem (problema comis voiajorului)*
FHTSP	*Fast Heuristically Traveling Salesman Problem (rezolvarea rapidă a problemei comis voiajorului);*
CAD	*Computer Aided Design (proiectare asistată de calculator)*
CAM	*Computer Aided Manufacturing (fabricaţie asistată de calculator)*
CAE	*Computer Aided Engineering (inginerie asistată de calculator)*
TechnoPack	*Pachet de programe realizat de autor pentru această lucrare*
TechnoCAD	*Sistemul de proiectare şi analiză a SD*
TechnoCAM	*Sistemul de generare cod NC pe baza SD*
TechnoFunction	*Convertor din reprezentarea analitică a funcţiilor în DXF*
TechnoMesh	*Convertor 2 ½ şi 3 axe din DXF în NC via GNCPP pentru plase*
TechnoCurves	*Convertor 2 şi 2 ½ axe din DXF în NC via GNCPP şi modelator*
Techno2D	*Convertor 2 şi 2 ½ axe din DXF în NC via GNCPP pentru curbe*
TechnoBulge	*Modul în TechnoCAD care rezolvă reţele de difracţie*

Anexa C: Definiţii

ADA (limbaje): unul dintre cele mai complexe şi solide limbaje apărute; conţine verificare puternică de tip metode de detecţie şi tratare de erori; utilizează compilatoare foarte complexe.

Algoritmi (pseudocod): aparate matematice, proceduri, rutine, funcţii, metode; se vor regăsi în două locuri, sub formă de funcţii (în cazul în care nu sunt conţinuţi într-un corp de clasă) sau metode (în cazul în care se găsesc încapsulaţi în interiorul unei clase).

Aparate matematice (pseudocod): vezi algoritmi.

Asamblare (limbaje): o evoluţie naturală a codului maşină, prin renumirea într-un format mai accesibil a acestuia.

BASIC (limbaje): limbaj deosebit de simplu, apărut la început fără nici o formă de modularizare a informaţiei, cu excepţia subrutinelor, care, însă, nu puteau avea variabile locale.

BazăDiscretă (clasă): clasă de bază derivată din baza vectorială, destinată stocării suprafeţelor mască şi suprafaţă discretă; conţine informaţii despre pasul şi numărul de puncte discrete în cele două direcţii X şi Y.

BazaVectorială (clasă): clasă elementară care trasează un comportament comun pentru clasele mai evoluate (curbe, plase, suprafeţe discrete).

Booleeni (clasă): clasă particulară derivată din întregi.

Borland Pascal (limbaje): limbaj evoluat orientat obiect.

C (limbaje): limbaj de nivel jos, destinat în general scrierii sistemelor de operare portabile.

C++ (limbaje): limbaj evoluat orientat obiect.

CAD Report (reviste): revistă despre proiectarea asistată, editată la Tg. Mureş.

Clasa (cuvânt cheie, pseudocod): începutul definiţiei unei noi clase.

Clasele (pseudocod) sunt scrise îngroşat şi verde închis şi au asociată obligatoriu o prescurtare cât mai sugestivă, formată din una sau mai multe caractere scrise cu literă mică; prefixarea trebuie specificată obligatoriu numai prima oară când clasa este definită.

Cod maşină (limbaje): limbaj de programare rudimentar.

Colecţiile (clasă): seturi de obiecte de acelaşi tip stocate împreună; accesul la un element dintr-o colecţie se face cu ajutorul operatorul [] care va returna un element din tipul stocat în colecţia respectivă.

Comentariile (pseudocod): clarificări ale pseudocodului; sunt scrise în italic şi sunt prefixate de //.

Curbe superioare (metode de stocare): curbe care utilizează reprezentări polinoamiale pe intervale (mai ales cubice); sunt folosite cvasitotal, în toate sistemele de modelare moderne, ca şi curbe şi suprafeţe spline, B-Spline, Bézier, Hermite etc.

Curbele (clasă) folosite în acestă lucrare sunt cele mai elementare posibile (polilinii în R^3), folosind ca reprezentare o colecţie de puncte, cu singura convenţie că, curba se va genera trasând segmente de dreaptă între fiecare pereche de puncte consecutive.

Curbele analitice (metode de stocare): curbe care pot fi descrise printr-un singur set de ecuaţii matematice (nu conţin seturi de ecuaţii pe porţiuni de curbă sau suprafaţă descrisă).

Curbele închise (clasă): curbe a căror prim şi ultim element coincide.

Curbele plane (clasă): curbe care nu conţin informaţii despre cota Z; nu au fost tratate separat, ci sunt utilizate cele 3D, nefolosind cota Z.

Curbele plane (metode de stocare): curbe care sunt stocate ca o serie de segmente de linie, arce de cerc, sau curbe superioare în planul XY.

Curbele superioare (clasă): curbe care nu sunt stocate ca o secvenţă de segmente de dreaptă, deci pot conţine segmente de cerc sau de alte curbe analitice.

Cuvinte cheie (pseudocod): cuvintele rezervate în limbajul pseudocod (Clasa, DerivatăDin, Operator, Enumerare).

Dacă (funcţie specială, pseudocod) o condiţie este îndeplinită, atunci se execută prima enumerare de instrucţiuni, altfel (eventual) se execută a doua enumerare.

Delphi (limbaje): limbaj evoluat orientat obiect.

DerivatăDin (cuvânt cheie, pseudocod): moştenirea comportamentului unei clase părinte.

Eifel (limbaje): limbaj evoluat orientat obiect.

Enumerare (cuvânt cheie, pseudocod): începutul enumerării valorilor posibile pentru un obiect.

Enumerările (pseudocod): sunt fără paranteze acolade în cazul când este vorba de un element, sau între paranteze acolade dacă sunt mai multe.

FORTRAN - FORmula TRANslation (limbaje): primul adevărat limbaj de programare de nivel înalt; permitea pentru prima dată programarea modulară, organizarea codului în subrutine şi portarea (interpretarea sau compilarea acestuia) pe diferite sisteme.

Funcţii speciale (pseudocod): funcţii implementate pentru asigurarea logicii algoritmilor (Dacă, Altfel, Pentru, PânăCând, Întoarce).

Funcţiile (pseudocod): algoritmi liberi, care nu sunt încapsulaţi în interiorul vreunei clase (vezi algoritmii).

Hello CAD Fans (revistă): revistă despre proiectarea asistată, editată în Bucureşti.

Indecşii vectorilor (pseudocod): sunt între paranteze pătrate.

Întoarce (funcţie specială, pseudocod): opreşte execuţia unui algoritm şi (eventual) întoarce un obiect din clasa returnată de algoritmul respectiv.

Întregii (clasă): mulţimea numerelor întregi Z.

Java (limbaje): limbaj evoluat orientat obiect.

Limbajul pseudocod (pseudocod): limbaj de tip pseudocod orientat pe obiecte, asemănător cu C++[CPP] sau Java[JAV], sintaxa nefiind asemănătoare în totalitate, folosit pentru structurarea conceptelor descrise pe parcursul lucrării; scopul principal al acestei structurări este acela de a fi succintă, flexibilă, sugestivă, trebuind să posede abilitatea de a descrie aparatele matematice, structurile de date, designul şi implementarea algoritmului într-un singur loc.

Lista cu parametri (pseudocod) ai unei metode sau funcţii; este întotreauna între paranteze rotunde.

Masca (clasă): matrice bidimensională de aceleaşi dimensiuni cu cea a SD, în care sunt stocate obiecte de tip Boolean, şi care este utilizată pentru raportarea diferitelor rezultate ale unor analize.

Matrici cu pas uniform sau neuniform (metode de stocare): matrice în care suprafaţa este stocată ca şi cotă Z.

Metodele (pseudocod): algoritmi specifici unui clase; sintaxa lor este exact ca aceea de la algoritmi, cu excepţia că, având acces la datele private interioare unei clase, când se cheamă o metodă, aceasta trebuie prefixată de numele obiectului respectiv, urmată de un punct (vezi algoritmii).

Metodele analitice (metode de stocare): curbe compuse, suprafeţe analitice, solide sculpturale şi simple.

Metodele discrete (metode de stocare): polilinii, suprafeţe faţetate, stocări matriciale, seturi de puncte.

Modula (limbaje): limbaje apărute începând din 1971, în urma lucrărilor profesorului Wirth în domeniul programării structurate; au fost o adevărată revoluţie în mediul academic.

Multiliniile plane (metode de stocare): caz particular al curbelor şi al poliliniilor spaţiale, deopotrivă; sunt stocate ca o serie de segmente de linie în planul XY,.

Multiliniile spaţiale (metode de stocare): pot descrie orice curbă în spaţiu, sub o toleranţă; sunt stocate ca o colecţie de puncte în 3D.

Oberon (limbaje): limbaje apărute, începând din 1971, în urma lucrărilor profesorului Wirth în domeniul programării structurate; au fost o adevărată revoluţie în mediul academic.

Obiectele (pseudocod): instanţele (de tipul) unei clase, fiind declarate în felul următor: Clasa tvNume, numele obiectului prefixat obligatoriu de prescurtarea clasei respective (notaţia ungară); ele sunt scrise normal.

Object Pascal (limbaje): limbaj evoluat orientat obiect.

Objective C (limbaje): limbaj evoluat orientat obiect.

Objectual Ada (limbaje): limbaj evoluat orientat obiect.

Operator (cuvânt cheie, pseudocod): începutul descrierii operatorilor permişi pe clasa respectivă.

Operatorii (pseudocod): setul de operaţii posibile cu Clasa respectivă; sunt utilizaţi în scopul de a scrie expresiile într-un limbaj mai apropiat de cel natural.

PânăCând (funcţie specială, pseudocod) este satisfăcută condiţia, execută o enumerare de instrucţiuni.

Pascal (limbaje): limbaje apărute începând din 1971, în urma lucrărilor profesorului Wirth în domeniul programării structurate; au fost o adevărată revoluţie în mediul academic.

Pentru (funcţie specială, pseudocod) fiecare valoare, începând cu condiţia de start şi sfârşind cu condiţia de sfârşit satisfăcută, incrementând cu condiţia de incrementare, execută o enumerare de instrucţiuni.

Plasele (metode de stocare): cele mai simple obiecte care pot fi utilizate pentru stocarea suprafeţelor.

Poliliniile spaţiale (metode de stocare): pot descrie orice curbă în spaţiu, sub o toleranţă; sunt stocate ca o colecţie de puncte în 3D.

Poliniile plane (metode de stocare): caz particular al curbelor şi al poliliniilor spaţiale, deopotrivă; sunt stocate ca o serie de segmente de linie în planul XY.

Proceduri (pseudocod): vezi algoritmii.

Punctele (clasă): entităţi elementare, folosite de către toate clasele vectoriale superioare, pentru a stoca valorile diferitelor entităţi care le compun.

Reali (clasă) sau numerele reale R: clasă elementară, descrisă pentru a înţelege operaţiile executate cu numere reale.

Rutine (pseudocod): vezi algoritmii.

Seturi de puncte (metode de stocare): datele sunt stocate ca o colecţie de puncte în spaţiu.

ŞirulDeCaractere (clase): colecţie de obiecte din clasa Întreg; este utilizat la transmiterea mesajelor.

Smaltalk (limbaje): limbaj evoluat orientat obiect.

Solide sculpturale (metode de stocare): solide reprezentate analitic pe intervale care trebuie să închidă un volum.

Solidele simple (metode de stocare): solide de tipul sfere, tori, conuri, piramide, curbe extrudate.

SuprafaţaDiscretă (clasă): metoda de stocare a suprafeţelor discrete; sunt matrici bidimensionale, în care sunt stocate obiecte de tip Real.

Suprafeţe extrudate (metode de stocare): curbe (polinii, multilinii) în 2D, care au un început şi un sfârşit pe axa Z.

Suprafeţe faţetate (metode de stocare): metodă generică de stocare a oricărei suprafeţe superioare, sub formă de triunghiuri sau patrulatere.

Suprafeţe superioare (metode de stocare): utilizează reprezentări polinoamiale pe intervale (mai ales cubice); sunt folosite cvasitotal în toate sistemele de modelare moderne, ca şi curbe şi suprafeţe spline, B-Spline, Bezier, Hermite etc.

Suprafeţele analitice (metode de stocare): pot fi descrise printr-un singur set de ecuaţii matematice (nu conţin seturi de ecuaţii pe porţiuni de curbă sau suprafaţă descrisă).

Suprafeţele superiore (clasă): suprafeţe care sunt descrise prin alte metode decât cele de la plase.

Anexa D: Legături web

D1. Sisteme mari de proiectare şi fabricaţie

Produs, legătură web	Compania
CADDS-5, http: //www.cv.com/	Computervision USA
CATIA, http: //www.catia.ibm.com/	Dassault France
EMS, http: //www.intergraph.com/mech/ems/ems.htm	Intergraph USA
EUCLID, http: //www.matra-datavision.com/	Matra Datavision France
MASTER SERIES, http: //www.sdrc.com/	SDRC USA
Pro/ENGINEER, http: //www.ptc.com/	PTC USA
UNIGRAPHICS, http: //www.ug.eds.com/	EDS USA

D2. Alte sisteme de proiectare şi fabricaţie

Produs, legătură web	Compania
http: //www.anvil5k.com	MCS USA
CADAM, http: //www.clearlake.ibm.com/MFG/engineering/consider.html	Dassault France
Cadkey, http: //www.cadkey.com/	Baystate Technologies USA
CADRA, http: //www.adra.com/	ADRA USA
CAELUM, http: //caelum.co.jp/	CAELUM Japan
Cimatron, http: //http:/www.cimatron.com/	Cimatron Israel
Cimalog, http: //www.swissprecision.com/SPEE3.html	Swiss Precision/Engineer
GMS, http: //www.graftek.com/	Graftek
ICEM, http: //www.cdc.com/icem.html	ICEM USA
I-DEAS, http: //www.sdrc.com/	SDRC USA
MASTERCAM, http: //www.mastercam.com/	CNC Software USA
MicroCADAM, http: //www.microcadam.com/	MICROCADAM USA
METALMAN, http: //www.unm.edu/~baltz/	Metalman USA
MOZAIC, http: //www.auto-trol.com	Auto-Trol
Mr Machinist , http: //www.pennet.net/commercial/f1/	F1 Computing USA
Para-Max, http: //www.innotts.co.uk/~mecs/	M.E.C.S. UK
Prelude, http: //www.matra-datavision.com/	Matra Datavision France
SURFCAM, http: //www.surfware.com/	Surfware USA
VisiCAD/VisiCAM, http: //www.vero.it/	Vero Italy
Varimetrix, http: //www.vx.com/	
Virtual Gibbs, http: //www.bureaug.co.uk/gibbs/index.html	Gibbs USA

D3. Sisteme de proiectare

Produs, legătură web	Compania
Ashar Vellum, http: //www.ashlar.com/	Ashlar USA
AutoCAD, http: //www.autodesk.com/	Autodesk USA
Accugraph, http: //www.accugraph.com/	Accugraph USA
ArchiTECH.PC, http: //www.softcad.com/	SoftCAD Belgium
CADVANCE, http: //www.cadvance.com/	Furukawa Info Technology USA
CAD/DRAW, http: //www.tommysoftware.com/etsw001.htm	Thomas Maier Germany
Drafix, http: //www.softdesk.com/	Softdesk USA
DesignCAD, http: //www.viagrafix.com/	ViaGraphix USA
DynaCADD, http: //www.ditek.com/	Ditek Canada
Douglas CAD/CAM, http: //www.douglas.com/	Douglas Electronics USA
FastCAD, http: //com.primenet.com/evcomp/	Evolution computing USA
Form.Z, http: //www.formz.com/ads.html	auto"des"sys USA
FreeHand, http: //www-1.macromedia.com/Tools/Studios/FGS/index.html	Macromedia USA
GMPCAD, http: //www.ens-cachan.fr/~gmpcad/	IUT Cachan France
HVACp, http: //www.thecube.com/	MC Squared USA
Mechanical Desktop, http: //www.autodesk.com/	Autodesk USA

Microstation, http: //www.bentley.com/	Bentley USA
MiniCAD, http: //www.graphsoft.com/	Graphsoft Inc USA
Powerlib, http: //www.cadsoftware.co.uk/plib.htm	CAD Software UK
QuickDraw, http: //www.wombat.com.au/wombat/showcase/sa/qikdraw/index.html	QuickDraw Australia
SolidBuilder, http: //www.solidbuilder.com/	Builders for Builders USA
RAY DREAM, http: //www2.us.com/raydream/	Fractal USA
SolidEdge, http: //www.intergraph.com/solidedge	Intergraph USA
SolidWorks, http: //www.solidworks.com/	SolidWorks USA
SolidDesigner, http: //www.hp.com/	Hewlett Packard USA
TopCAD, http: //http:/www.merituk.co.uk/topcad/	TopCAD France
TrueCAD , http: //www.choicecomp.com/	
Trispectives, http: //www.eye.com/	3D Eyes USA
CADMAX TrueSurf Master, http: //www.cadmax.com/	CADMAX USA
TurboCAD, http: //www.imsisoft.com/catprod.html	IMSI USA
Visio, http: //205.185.183.34/html/a.html	Visio USA
Visual CADD, http: //www.numera.com/	Numera USA

D4. Procesoare APT

Produs, legătură web	**Compania**
APT/AC, http: //www.clearlake.ibm.com/MFG/engineering/consider.html	IBM USA
HMS APT, http: //www-europe.sgi.com/Products/appsdirectory.dir/Applications/	Houtzeel USA
PC APT, http: //www.ncsoft.com/index.html	N/C Software Inc. USA

D5. Sisteme de fabricație

Produs, legătură web	**Compania**
ACU-CARV, http: //www.gtii.com/olmsted/oec-home.htm	Olmsted Engineering USA
AlphaCAM, http: //www.licom.com/	LicomUK
ArtCAM, http: //www.spline.nl/	Delft Spline SystemThe Netherlands
AutoCODE, http: //www.autocode.com/	Kramer ConsultingUSA
AutoPRO,	Intercim USA
BioSculptor, http: //www.oandp.com/commerci/biosculptor/index2.htm	O&P Discret Technologies USA
CAMAND, http: //www.camax.com/	CamaxUSA
CNC Code Creator, http: //www.solas-data.ie/creator.htm	SolasData Ireland
CamModul, http: //www.microtech.com/	MicroTech Sweden
CAMSoft, http: //pages.prodigy.com/CAMSOFT/	CAMSoft USA
CAMWrite, http: //ourworld.compuserve.com/homepages/tech_group/home.htm	TheTechGroup UK
CAPSMill and Turn, http: //www.cadem.com/	Cadem India
CIMPLEX, http: //www.csn.net/metal/home/cimplex/	Cimplex USA
DUCT, http: //www.delcam.com/	Delcam UK
EZFeatureMill, http: //www.enggeo.com/	E.G.S.USA
EdgeCAM, http: //www.pathtrace.com/	Pathrace UK
ESPRIT, http: //www.dptechnology.com/	DPTechnology USA
FlexCAM, http: //www.aii.com/	Software Magic USA
GeoPath, http: //www.solution-ware.com/	SolutionWare USA
Goelan, http: //www.cni.fr/	CN Industries France
HyperMILL-HyperCAM, http: //www.openmind.de/	OPENMIND Germany
HI-MILL, http: //www.fidia.it/	FIDIA Italy
NC Works, http: //www.industry.net/c-a/showfile/00cm7/sponsors/psc/psc02	Progressive Software Corporation
PEPSCUT, http: //www.peps.com/	Camtek UK
ShopCAM	ShopSystem USA
SmartCAM, http: //www.camax.com/	Camax USA
Synergy, http: //www.weber.com/	Weber SystemUSA
TechnoCAM, http: //cccsat.sorostm.ro/billasoft/	Billasoft Romania
TekSoft, http: //www.teksoft.com/	TekSoft USA
WorkNC	Sescoi France
XCAM, http: //www.industry.net/c-a/showfile/00cm7/sponsors/psc/psc03/DoAuth	Progressive Software Corporation

D6. Informaţii despre proiectare şi fabricaţie

Produs, legătură web	Compania
3Dview, http: //www.millennion.com/	Millennion USA
ACIS, Strata, http: //www.csn.net/spatial/	Spatial Technology USA
Algor, http: //www.algor.com/	Algor USA
ANSYS, http: //www.ansys.com/	ANSYS USA
AnthroCAM, http: //www.greatlakesmet.com/faro2.htm	Faro Technologies USA
BOX-NET DNC, http: //www.manusoft.fi/boxnet.html	Manusoft Finland
CAM-POST, http: //www.icam.com/	ICAM Canada
Code-EX , http: //www.iasweb.com/index.htm	Industrial Automation Solutions USA
DADS, http: //www.cadsi.com/	CADSi USA
http: //www.camsoftware.com/DClasa, http: //www.camsoftware.com/	CAM Software USA
DesignBase, http: //www.ricoh.co.jp/swd/designbase/index_e.html	Ricoh Japan
Director 4000, http: //www.peps.com/	Camtek UK
DloG, http: //mfginfo.com/cadcam/dlog/dnc.htm	DloG Germany
CamLink, http: //www.proaxis.com/~griffobros/	Griffo Brothers USA
EDS-DNC, http: //www.eds.com/edsdnc	EDS USA
FlexIt, http: //ourworld.compuserve.com/homepages/pilontech/	Pilon Technology USA
IGES PRO, http: //www.compunix-usa.com/	Compunix USA
LightWorks NC/MachineWorks, http: //www.lightwork.com/nc_overview.html	LightWorks UK
MetCAPP, http: //www.iams.org/cim/metcapp/desc.htm	I.A.M.S. USA
MSC/NASTRAN, MSC/PATRAN, http: //www.macsch.com/index.html	MacNeal Schwendler USA
Mechanica, http: //www.rasna.com/	Rasna USA
Metaphase, http: //www.sdrc.com/metaphase/intro/	SDRC USA
Modflow, http: //www.worldserver.pipex.com/moldflow/	Modflow Autralia
NCTool, http: //www.ventura.co.il/c/cadtech/index.htm	CadTech Israel
NCV, http: //www.csn.net/metal/home/cimplex/	Cimplex USA
NCSimul, http: //www.spring.fr/	Spring France
Predator, http: //www.teleport.com/~predator/	Predator Software USA
NC Verify, http: //www.siriussys.com/	Sirius USA
PDE Wizard, http: //www.pdew.com/	P.D.E Wizard USA
Pro NC Edit, http: //www.cncci.com/soft.htm	CNC Concepts USA
RevPost, http: //www.4dcadcam.com/	4 D Engineering UK
RobCAD, http: //www.tecnomatix.com/	Tecnomatix Israel
Shape, http: //www.xox.com/	XOX USA
Sherpa, http: //www.sherpa.com/	Sherpa USA
SoftMachine, http: //www.silma.com/	Silma USA
Spaceball, http: //www.spacetec.com/Hard Disk/WEB_SITE/PRODUCTS/2003/2003.html	Spacetec Inc. USA
Surfacer, http: //www.iware.com	Imageware USA
Valisys, http: //www.tecnomatix.com/	Tecnomatix Israel
Tailor Made Software, http: //www.serv.net/tms/	Tailor Made Software USA
Vericut, http: //www.vericut.com/	CGTech USA
Virtual NC, http: //www.deneb.com/	Deneb USA
HP WorkManager, http: //www.hp.com/	Hewlett Packard USA
XYZPro EDIT, http: //www.ascendtec.com/	Ascendant USA

D7. Informaţii generale despre proiectare şi fabricaţie

Nume, legătură web	Comentarii
Society of CAD/CAM Engineer, http: //chopin.kist.re.kr/cadcam/cadcam.htm	
CAD/CAM Headlines, http: //www.newspage.com/NEWSPAGE/cgi-bin/walk.cgi/	
CAD/CAM Recruiters, http: //web.infoave.net/cadcam/welcome.html	Căutaţi o slujbă ?
CAD Society, http: //www.wbh.com/cadsociety.html	
CAD Rating Guide, http: //www.wbh.com/crg4description.html	
CIMdata, http: //www.cimdata.com/	
CIMWorld, http: //www.cimworld.com/	Stiri pentru utilizatorii profesionişti
Daratech, http: //www.daratech.com/	

DH Brown,http: //www.dhbrown.com/	
Specifying CAE/CAD/CAM system, http: //www.os.kcp.com/home/catalog/speevacae.html	Teste si expertize
Portland Microstation Community, http: //www.teleport.com/~timothyd/tmc/tmc.htm	microstation din Portland
SunPTC, http: //www.sunptc.com/	Pagina PTC prezentata de SUN
Society of CAD Engineer, http: //www.cadsociety.org/	
SGI Mechanical CAD/CAM, http: //www.sgi.com/Products/appsdirectory.dir/	
Wahoo's PC CAD/CAM Page, http: //www.neca.com/~wahoo/camindx.html	
METAL Machining and Fabrication Internet Directory, http: //www.mmf.com/metal/	
Manufacturers Information Net Home Page, http: //mfginfo.com/home.htm	
Solid Design Automation Newsletter, http: //www.ppgsoft.com/sol.html	
TechniCOM Newsletter, http: //www.technicom.com/	Noutati interesante CAD/CAM
Pro/E Job Network, http: //www.pejn.com/	Lista de lucuri de munca in Pro/ENG
I-DEAS Job Network, http: //www.ideasjn.com/	Lista de lucuri de munca in I-DEAS
UNIGRAPHICS Job Network, http: //www.ugjn.com/	Lista de lucuri de munca in UNIGR
CATIA Job Network, http: //www.catjn.com/	Lista de lucuri de munca in CATIA

D8. Grupuri de discuţii despre proiectare şi fabricaţie

Grupul de noutăţii	Comentarii
alt.cad, news:alt.cad	
alt.cad.autocad, news:alt.cad.autocad	
alt.cad.cadkey, news:alt.cad.cadkey	
alt.comp.acad-freedom.news, news:alt.comp.acad-freedom.news	
alt.comp.acad-freedom.talk, news:alt.comp.acad-freedom.talk	
comp.cad.microstation, news:comp.cad.microstation	
comp.cad.i-deas, news:comp.cad.i-deas	
comp.cad.autocad, news:comp.cad.autocad	
comp.cad.pro-engineer, news:comp.cad.pro-engineer	
comp.sys.intergraph, news:comp.sys.intergraph	

D9. Alte liste de pagini web destinate proiectării şi fabricaţiei WEB List

Lista, legătură web	Comentarii
http: //pcfolini.eng.unipr.it/sites.html	Lista lui Franco Folini
http: //www.free.cts.com/crash/p/patcad/	Noutatile lu Pat
http: //www-cad.fnal.gov/otherlinks.html	Lista de Fermi Lab
http: //wwwcadcam.de	O lista Germana despre CAD/CAM
http: //www.cadshack.com/support.htm	O baza de date cu cunostiinţe despre CAD
http: //tribeca.ios.com/~compvent/cadstuff.html	Accesori CAD
http: //promo.net/pdm/bm_cad_i.htm	Lista lui Pietro Di Micelli
http: //www.cadonline.com/woc.htm	The world of CAD online by CADALYST
http: //www.renature.com/instarc/cad_net.html	Resource CAD in web
http: //haven.uniserve.com/~ralphg/cadsite1.htm	Pagini CAD in web
http: //www.mcp.com/newriders/cad/sites.html	Listă CAD
http: //www.arch.buffalo.edu/pairc/cad.html	Listă CAD
http: //xenoy.mae.cornell.edu/cad-info.html	Resurse CAD/CAM
http: //www.headquarters.com/CGI-WIN/homepage.exe?USE/Search_cad1	Lista cu aplicatii in mecanicaCAD/CAM
http: //www.webcom.com/%7Eimt/other.html	Lista de CAD de la I.M.T.
http: //www.nerdworld.com/nw672.html	Lista de CAD de la Nerd World CAD
Cadsyst CAD Links, http: //www.buildingweb.com/cadsyst/cadlink.html	Lista de CAD despre AutoCAD

BIBLIOGRAFIE

ACAD **** Autodesk - AutoCAD, manual de utilizare.*

ALM **** ALMI, manuale de utilizare.*

BEN *Benchimol G., Lévine P., Pomerol J.-C. – Systèmes experts dans l'entreprise, Hermès, Paris, 1990.*

BIB **** BIBEXE, manual de utilizare, 1994.*

BJS *Stroustroup B. – C++ Programing Language Third Edition, Addison Wesley, 1997.*

BKOS *de Berg M., van Kreveld M., Overmars M., Schwartzkopf O. – Computational Geometry – algorithms and applications, Springer – Verlag, Berlin Heildelberg, 1997.*

BOURDI *Bourdichon P. – L'ingénierie simultanée et la gestion d'information, Hermès, Paris, 1994.*

BOW *Bowyer A., Woodwark J. – A programmer's geometry Butterworths, University of Bath, 1983.*

CAT *Catrina O., Cojocaru I. – Turbo C++, Editura Teora, Bucuresti, 1993.*

CNC **** CNC 600, CNC 6001, CNC 646 Manuale de utilizare. 1990.*

COH *Cohen P.H., Joshi S.B. – Advances in integrated product design and manufacturing, PED–Vol. 47, ASME, New York, 1990.*

COR *Corney J. – 3D Modeling with the ACIS Kernel, Wiley, Ontario, 1997.*

CRI *Cristea V. – Tehnici de programare, Editura Teora, Bucuresti, 1992.*

CTR **** CorelTrace, manual de utilizare, fişier help, www.corel.com, 1998.*

DFR *Rogers D. F., Adams J. A. – Mathematical Elements for Computer Graphics, McGraw Hill International, Singapore, 1989.*

DM01 *Micşa D. – TechnoPack V1.0 Manual de utilizare, BillaSoft, Timişoara, 1992.*

DM02 *Micşa D. – TechnoPack V1.1 Manual de utilizare, BillaSoft, Timişoara, 1993.*

DM03 *Micşa D. – TechnoPack V1.2 Manual de utilizare, BillaSoft, Timişoara, 1994.*

DM04 *Micşa D. – TechnoPack V2.0 Lite Manual de utilizare, BillaSoft, Timişoara, 1996.*

DM05 *Micşa D. – TechnoPack V2.0 Pro Manual de utilizare, BillaSoft, Timişoara, 1997.*

DM06 *Micşa D. – TechnoPack V2.0 Pro, prezentare produs, Chip nr. 14. Braşov, 1996.*

DM09 *Micşa D. – Analytical Model – User requirements, Pathtrace Engineering, Reading, 1997.*

DM10 *Micşa D. – Analytical Model – Functional specification, Pathtrace Engineering, Reading, 1997.*

DM11 *Micşa D. – Analytical Model – Design specification, Pathtrace Engineering, Reading 1997*

DM12 *Micşa D. – Faceted Model – User requirements, Pathtrace Engineering, Reading, 1998.*

DM13 *Micşa D. – Faceted Model – Functional specification, Pathtrace Engineering, Reading,1998.*

DM14 *Micşa D. – Faceted Model – Design specification, Pathtrace Engineering, Reading,1998.*

DM15 *Micşa D. – Faceted Model – Functional test, Pathtrace Engineering, Reading,1998.*

DMCS07 *Stăncescu C., Micşa D. – TechnoPack V1.2 prezentare produs Hello CAD Fans nr. 6, Bucureşti, 1993.*

DMCS08 *Stăncescu C., Micşa D. – TechnoPack V2.0 Pro prezentare produs Hello CAD Fans nr. 23, Bucureşti, 1996.*

DMW0 *Micşa D. – TechnoPack Prezentare produs, document www, site www.dnttm.com, Timişoara, 1996.*

DMW1 *Micşa D. – EdgeAnalyser – Prezentare produs, document www, site www.pathtrace.com Pathtrace, Reading, 1998.*

DMW2 *Micşa D. – Prezentare TechnoPack Pagina web: www.sorostm.ro\billasoft, BillaSoft, Timişoara, 1997.*

DMW3 *Micşa D. – Analytical Model – Functional TestingPagina web www.pathtrace.com Pathtrace, Reading, 1997.*

DMW4 *Micşa D. – Faceted Model – Functional Testing Pagina web www.pathtrace.com Pathtrace, Reading, 1998.*

DRA92a *Drăghici G. – MasterCAM Tournage, IUT Béthune, 1992.*

DRA92b *Drăghici G. – MasterCAM Fraisage, IUT Béthune, 1992.*

DRA95a *Drăghici G., Slavici T. – Contribuţii la determinarea şi prelucrarea entităţilor geometrice complexe, Lucrările celei de–a VII–a Conferinţe Internaţionale de Inginerie Managerială şi Tehnologică, Timişoara, iunie 1995.*

DRA95b *Drăghici G., Slavici T., Dumitru E. – Some Contributions for Defining and Determination of the Geometrical Offset Entities, Buletinul Stiinţific al Universităţii Tehnice din Timişoara, Tom 40 (54), Mecanică, 1995.*

DUT *Dutta D., Woo A.C., Chandrashekhar S., Bailey S., Allen M. – Concurrent engineering, PED–Vol. 59, ASME, New York, 1992.*

DWG **** Autodesk – Formatul DWG, www.autodesk.com. 1985 – 1998.*

DXF **** Autodesk – Formatul DXF, www.autodesk.com. 1985 – 1998.*

ECAM **** Pathtrace – EdgeCAM, manual de utilizare; www.pathtrace.com. 1985 – 1998.*

ELLI *Ellis M., Stoustrup B. – The Annotated C++ Reference Manual, Addison–Wesley, 1990.*

GAR90a *Gardan Y. – Etudes en CFAO. Outils et applications de l'intelligence artificielle en CFAO, Hermes, Paris, 1990.*

GAR90b *Gardan Y., Minich C. – La modélisation géométrique et l'extraction de caractéristiques de forme, La gamme automatique en usinage, Hermès, Paris, 1990.*

GAR92 *Gardan Y. – La CFAO, Hermès Paris, 1992.*

GF *Farin, G. – Curves and surfaces for Computer Aided Geometric Design Academic Press Inc. Harcout Brace Jovanovich, Publishers, San Diego, CA92101, 1990.*

HCF *Hello CAD Fans, Colecţia revistei 1992 – 1998.*

IFMP *I.D.Faux, M.J.Prat – Computational Geometry for Design and Manufacturing, Ellis Horwood Publishers, Halsted Press, 1987.*

IGES **** ANSI – Formatul IGES document www www.tailormade.com/iges1.htm; 1996.*

JAG *Jagou P. – Concurrent Engineering, Hermès, Paris, 1993.*

LEF *Lefur E., Mathieu L. – Méthodes d'optimisation sous contraintes appliquées à la determination des conditions de coupe, La gamme automatique en usinage, Hermès, Paris, 1990.*

LEO91 *Leon J.–C. – Modélisation et construction de surfaces pour la CFAO, Hermès, Paris, 1991.*

LWRK **** RapidWorks, manual de utilizare; www.ligthwork.com, 1998.*

MANU **** The Manufacturer, colecţia revistei, The Manufacturing Knowledge Group, 1996 – 1998.*

MAZ **** MAZATROL, manual de utilizare.*

MOL *Moldoveanu F., s.a. – Grafică pe calculator, Editura Teora, Bucureşti, 1996.*

MOR *Moreau R. – L'approche objets, Masson, Paris, 1994.*

MSDN **** Microsoft Developer Network, colecţia revistei, Microsoft Press, 1996 – 1998.*

MUS *Muslea I. – C++ Programarea orientată pe obiecte, Microinformatica, Cluj, 1992.*

NCV **** NC Verify, Sirirus technology, fisier help, V2.x – V5.0, 1995 – 1998.*

NUM **** NUMAFORM, manual de utilizare.*

RAN *Randoing J.–M. – Les SGDT, Hermès, Paris, 1995.*

RBPEL *Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W. – Object Oriented Modeling and Design Prentice Hall Intl., Englewood Cliffs, New Jersey, 1991.*

SAT **** Spatial technology – Formatul SAT http://www.acis.com/, 1995 – 1998.*

SAV *Savii G. G. – Bazele proiectării asistate de calculator, Editura Mirton, Timişoara, 1997.*

SAV96 *Savii G. – Medii de dezvoltare pentru aplicaţii de inteligenţă artificială, Universitatea "Politehnica" din Timişoara, 1996.*

SHI *Shin Y.C., Abodelmonem A.H., Kumara S. – Neural Networks in Manufacturing and Robotics, PED–Vol. 57, ASME, New York, 1992.*

SLA94a *Slavici T. – Contribuţii la programarea asistată de calculator a maşinilor unelte cu comandă numerică în vederea prelucrării entităţilor geometrice complexe, Teza de doctorat, Universitatea Tehnică Timişoara, 1994.*

SLA94b *Slavici T., Marinceu D. – AutoCAD şi alte tehnici CAD/CAM, Editura Mirton, Timişoara, 1994.*

SLA95a *Slavici T., Drăghici G. – Some Contributions to Determination and Manufacturing of Complex Geometrical Entities, Buletinul Stiinţific al Universităţii Tehnice Timişoara, Tom 40 (54), Mecanică, 1995.*

SLA95b *Slavici T., Drăghici G. – Tehnici de baleiere a suprafeţelor complexe în spaţiu, Lucrările celei de–a VII–a Conferinţe Internaţionale de Inginerie Managerială şi Tehnologică, Timişoara, iunie 1995.*

SLA96 *Slavici T. – Conducerea cu calculatorul a sistemelor tehnologice, Universitatea "Politehnica" din Timişoara, 1996.*

SOM *Somnea D., s.a. – Programarea in Assambler, Editura Tehnica, Bucuresti, 1992.*

SOR **** SORI, manuale de utilizare.*

STL **** Formatul STL, document www.*

TOM *Toma L. – Sisteme de conversie, achiziţie şi prelucrare a datelor, Universitatea tehnică, Timişoara, 1993.*

TRI *Midtbo T. – Spatial Modeling by Delaunay Networks of of Two and Three Dimensions, University of Trondheim, Teza de doctorat, document www, 1996.*

VCPP *Williams M. – Bazele Visual C++ 4, Editura Teora, Bucureşti, 1996.*

ZEI *Zeid I. – CAD/CAM Theory and Practice, McGraw–Hill, 1991.*