CERCETĂRI PRIVIND ECHIPAMENTE SI PROCESE ELECTROTERMICE PERFORMANTE ÎN CÂMP DE MICROUNDE

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea Politehnica Timişoara
în domeniul INGINERIE ELECTRICĂ
de către

Ing. Ion-Cristian Abrudean

Conducător ştiinţific:	prof.univ.dr.ing. Ioan Șora
Referenţi ştiinţifici:	m.c.acad.dr.fiz. Ladislau Vekas
	prof.univ.dr.ing. Mircea Chindriş
	conf.univ.dr.ing. Alexandru Hedeş

Ziua susţinerii tezei: 24.10.2014

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor
12. Ingineria sistemelor
13. Inginerie energetică
14. Calculatoare şi tehnologia informaţiei
15. Ingineria materialelor
16. Inginerie şi Management

Universitatea Politehnica Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii mele în cadrul Departamentului de Inginerie Electrică şi Informatică Industrială din Facultatea de Inginerie Hunedoara a Universităţii Politehnica Timişoara.

Această lucrare constituie un studiu asupra echipamentelor şi proceselor electrotermice în câmp de microunde. Sunt evidențiate atât aspectele teoretice, cât și modelări și simulări ale proceselor electrotermice într-un cuptor multimod. De asemenea, sunt prezentate și interpretate rezultate ale unor măsurători de parametri electrici în timpul funcționării unui cuptor cu microunde.

Timişoara, octombrie 2014 Ion-Cristian Abrudean

Autorul mulțumește conducătorului științific, domnul prof. dr. ing. Ioan Șora, pentru îndrumarea competentă și eficientă de-a lungul perioadei de elaborare a tezei.

Multe mulțumiri domnului conf. dr. ing. Caius Pănoiu pentru ideile, îndrumarea și sprijinul său, esențiale în realizarea acestei lucrări. Autorul mulțumește dnei. asist. dr. ing. Raluca Rob și dlui. conf. dr. ing. Gabriel Nicolae Popa pentru ajutorul acordat în vederea realizării măsurătorilor și redactării tezei.

Autorul le mulțumește colegilor de la Facultatea de Inginerie Hunedoara care într-un fel sau altul au ajutat la punerea în practică a experimentelor din această lucrare.

De asemenea, autorul mulțumește familiei, în mod special fratelui Nicolae Liviu Abrudean, pentru sprijinul acordat.

Abrudean, Ion-Cristian

Cercetări privind echipamente și procese electrotermice performante în câmp de microunde

Teze de doctorat ale UPT, Seria 6, Nr. 38, Editura Politehnica, 2014, 202 pagini, 100 figuri, 4 tabele.

ISSN:1842-7022

ISBN:978-606-554-875-6

Cuvinte cheie: electrotermie, microunde, încălzire, simulare, diferenţe finite, armonici, parametri electrici, C++, Java, LabView

Rezumat,

Teza constituie un studiu asupra echipamentelor şi proceselor electrotermice în câmp de microunde. Sunt evidențiate atât aspectele teoretice, cât și modelări și simulări ale proceselor electrotermice într-un cuptor multimod. De asemenea, sunt prezentate și interpretate rezultate ale unor măsurători de parametri electrici în timpul funcționării unui cuptor cu microunde. În urma studiului realizat asupra cuptoarelor multimod cu cavitate rezonantă au rezultat modele matematice şi programe de simulare validate experimental, care au permis determinarea unor caracteristici extrem de dificil de obţinut altfel. A fost analizată funcţionarea cuptoarelor de serie din punct de vedere electric şi a fost concepută o metodă de reglaj continuu a puterii bazată în cea mai mare parte pe componente ieftine, fabricate în serie mare. Această metodă a fost verificată prin simulări.

CUPRINS

Liste de tabele şi figuri 7
1. Introducere 10
1.1. Încălzirea volumetrică prin mijloace electrice 10
1.2. Evoluţia sistemelor de încălzire cu microunde 13
1.3. Stadiul actual 15
2. Încălzirea în câmp de microunde 17
2.1. Unde electromagnetice 17
2.1.1. Ecuaţiile de bază ale câmpului electromagnetic. Unda plană. 17
2.1.2. Reflexia şi refracţia pe o frontieră plană 20
2.2. Câmpul electric în medii dielectrice 23
2.2.1. Polarizarea dielectricilor în câmpul electric 23
2.2.2. Constanta dielectrică complexă 27
2.2.3. Ecuaţia încălzirii dielectricului 30
2.2.4. Dependenţa de temperatură şi umiditate a proprietăţilor dielectrice . 30
2.2.5. Adâncimea de penetrare 36
2.3. Limitări ale densităţii de putere disipată 37
2.3.1. Limitări termice 38
2.3.2. Străpungerea dielectrică şi producerea de arc electric 38
2.3.3. Ambalarea termică 40
3. Câmpul electromagnetic în structurile de microunde 41
3.1. Propagarea undelor într-un ghid dreptunghiular 42
3.1.1. Lungimea de undă de tăiere 43
3.1.2. Lungimea de undă a ghidului 44
3.1.3. Vitezele de propagare a undelor 45
3.2. Moduri de linie de transmisie 46
3.2.1. Componentele de câmp pentru un mod de linie de transmisie 47
3.2.2. Condiţii de existenţă a modulului linie de transmisie 48
3.3. Moduri de ghid 48
3.4. Ghidul de undă dreptunghiular 50
3.4.1. Soluţiile ecuaţiilor propagării în ghid dreptunghiular 50
3.4.2. Condiţii de tăiere 51
3.4.3. Condiţii de frontieră ale ghidului 52
3.4.4. Componentele de câmp ale undei în ghid. Moduri de ghid 53
3.5. Pierderi în pereţi conductori 59
3.6. Cavităţi rezonante 60
3.6.1. Moduri de oscilaţie într-o cavitate 62
3.6.2. Cavitatea paralelipipedică. Metoda reflexiilor. 63
3.6.3. Parametrii cavităţilor rezonante 66
3.6.4. Perturbaţiile cavităţii rezonante 67
3.7. Aplicatoare electromagnetice 70
3.7.1. Aplicatoare cu undă mobilă 70
3.7.2. Aplicatorul monomod 73
3.7.3. Aplicator cu două cavităţi 74
3.7.4. Aplicatorul multimod 74
3.8. Elemente specifice circuitelor de transmisie a energiei de microunde 76
3.8.1. Ghiduri de undă rectangulare şi curbe 76
3.8.2. Circulatoare 77
4. Modelarea şi simularea proceselor electrotermice într-un cuptor multimod 79
4.1. Generalităţi 79

4.2. Discretizarea spaţială 79
4.3. Diferenţierea numerică 81
4.4. Discretizarea spaţio-temporală a ecuaţiilor lui Maxwell..... 84
4.4.1. Algoritmul lui Yee 86
4.5. Modelarea proceselor termodinamice..... 89
4.5.1. Modelarea prin metoda diferenţelor finite..... 90
4.5.2. Stabilitatea modelului cu diferenţe finite 93
4.6. Câmpul electromagnetic cuplat cu cel termic într-un cuptor multimod 94
4.6.1. Discretizarea spaţială 95
4.6.2. Funcţionarea simulării 96
4.6.3. Rezultate şi interpretare 97
4.7. Validarea experimentală a modelului 105
4.8. Concluzii 109
5. Soluţii constructive pentru partea electrică a unui cuptor multimod 110
5.1. Caracteristici ale regimului periodic nesinusoidal 110
5.1.1. Integrarea numerică 112
5.1.2. Limitări impuse prin reglementări europene 114
5.2. Circuitul electric al cuptoarelor comune.....115
5.2.1. Caracteristici electrice 115
5.2.2. Calculul parametrilor magnetronului.....126
5.3. Simularea funcţionării cuptorului.....127
6. Concluzii şi contribuţii personale 141
Bibliografie.....143
ANEXE.....146

Liste de tabele şi figuri

Pag.

Tabel 1.1 Benzile de frecvenţe ISM 13
Tabel 3.1 Dimensiuni standardizate pentru ghiduri de undă la 2,45GHz 77
Tabel 5.1 Curenţii armonici maximi conform EN 61000–3–2:2006+A1+A2 114
Tabel 5.2 Variaţia parametrilor electrici la modificarea unghiului de comandă 140

Pag.

Fig. 1.1 Spectrul electromagnetic 12
Fig. 2.1 Undă electromagnetică plană în perspectivă 19
Fig. 2.2 Câmpul electric la frontieră 21
Fig. 2.3 Câmpul magnetic la frontieră 21
Fig. 2.4 Undă parţial reflectată 22
Fig. 2.5 Diagrama fazorială a vectorului polarizaţie P şi a intensităţii câmpului electric E 25
Fig. 2.6 Diagrama fazorială a densităţii curentului total indus într-un dielectric 29
Fig. 2.7 Factorul de pierderi efectiv al unui material dielectric neomogen in funcţie de frecvenţă, (b) – relaxarea apei legate, (w) – relaxarea apei libere, (mw) – pierderi Maxwell-Wagner şi (c) – pierderi prin conducţie 31
Fig. 2.8 Factorul efectiv de pierderi în funcţie de umiditate pentru orientări diferite ale câmpului electric 32
Fig. 2.9 Efectele frecvenţei şi temperaturii asupra proprietăţilor dielectrice ale apei pure şi soluţiilor saline 33
Fig. 2.10 Proprietăţile electrice ale hârtiei şi scândurii în funcţie de conţinutul de apă la 2450 MHz 33
Fig. 2.11 Valorile ε'_{eff} şi ε''_{eff} în funcţie de umiditate la lemnul de brad Douglas 34
Fig. 2.12 Valorile ε'_{eff} şi ε''_{eff} în funcţie de umiditate la lemnul de brad Douglas 35
Fig. 2.13 Valorile ε'_{eff} şi ε''_{eff} în funcţie de umiditate la lemnul de brad Douglas (grafice în perspectivă tridimensională) 36
Fig. 2.14 Rigiditatea dielectrică a aerului la presiune mică 39
Fig. 2.15 Rigiditatea dielectrică a vaporilor de apă la presiune mică 39
Fig. 3.1 Ghid de undă cu plăci plane paralele 41
Fig. 3.2 Metoda imaginilor 42
Fig. 3.3 Propagarea undelor în ghid 42
Fig. 3.4 Ghid de undă cu plăci plane paralele 43
Fig. 3.5 Viteza de grup 45
Fig. 3.6 Ghid de undă cu plăci curbate 46
Fig. 3.7 Ghid de undă coaxial 46
Fig. 3.8 Aspectul liniilor de câmp pentru o linie bifilară 47
Fig. 3.9 Ghid de undă dreptunghiular 50
Fig. 3.10 a) Câmpul electric în ghidul de undă pentru o valoare z oarecare. b) Componenta transversală a câmpului magnetic de a lungul axei z. 57
Fig. 3.11 Dependenţa câmpului electric de coordonata z 58
Fig. 3.12 Modurile TE_{11}, TM_{11} şi TM_{21} 58

Fig. 3.13 Componentele câmpului în semispaţiul conductor 59
Fig. 3.14 Cavităţi rezonante care provin dintr-un ghid uniform 60
Fig. 3.15 Condensator plan 60
Fig. 3.16 Funcţii Bessel 61
Fig. 3.17 Distribuţia câmpului electric între armăturile condensatorului 61
Fig. 3.18 Cavitate cilindrică 61
Fig. 3.19 Distribuţia câmpului electromagnetic în cavitate 62
Fig. 3.20 Întreţinerea oscilaţiilor electromagnetice 62
Fig. 3.21 Moduri de oscilaţie în cavitatea cilindrică 63
Fig. 3.22 Cavitatea paralelipipedică 63
Fig. 3.23 Componentele câmpului electromagnetic în cavitatea paralelipipedică 64
Fig. 3.24 Câmpul electromagnetic într-o cavitate paralelipipedică cu mod de oscilaţie TEM_1 (H_{101}) 66
Fig. 3.25 Cavitate rezonantă iniţială şi deformată 68
Fig. 3.26 Cavitate rezonantă iniţială şi perturbată 69
Fig. 3.27 Componentele câmpului electromagnetic şi vectorul Poynting 70
Fig. 3.28 Aplicator axial cu unda mobilă 71
Fig. 3.29 Aplicator în formă de serpentină 72
Fig. 3.30 Variaţia intensităţii câmpului electric într-un aplicator cu opt serpentine 72
Fig. 3.31 Aplicator de tip serpentină cu două generatoare de microunde 73
Fig. 3.32 Variaţia câmpului elctric în cavitatea paralelipipedică 73
Fig. 3.33 Aplicator cu două cavităţi 74
Fig. 3.34 Aplicator multimod rectangular 76
Fig. 3.35 Circulator cu patru intrări 77
Fig. 3.36 Prezentarea schematică a modului de tramsmisie a energiei 78
Fig. 4.1 Dispunerea nodurilor reţelei de discretizare spaţială la intersecţia curbelor de coordonate constante a) sau în poziţii oarecare b) 79
Fig. 4.2 Reţea de elemente dreptunghiulare a), înclinate b), circulare c) şi triunghiulare d) 80
Fig. 4.3 Exemple de discretizare spaţială cu pas variabil 80
Fig. 4.4 Domeniu poligonal specific liniilor cu două conductoare 81
Fig. 4.5 Moduri de aproximare a derivatei unei funcţii f(x) 82
Fig. 4.6 Selecţii de puncte posibile la operatori cu diferenţe finite 83
Fig. 4.7 Celula Yee 86
Fig. 4.8 Succesiunea prelucrării datelor prin MDFDT 88
Fig. 4.9 Dependenţa H(t) pentru apa pură în jurul punctului de topire 89
Fig. 4.10 Geometra cuptorului simulat 95
Fig. 4.11 Regim tranzitoriu, n = 500 97
Fig. 4.12 Regim tranzitoriu, n = 1000 98
Fig. 4.13 Regim staţionar, n = 5000 98
Fig. 4.14 Regim tranzitoriu, n = 500 99
Fig. 4.15 Regim staţionar, n = 5000 99
Fig. 4.16 Regim staţionar în sarcină 100
Fig. 4.17 Regim staţionar cu sarcină mică 100
Fig. 4.18 Temperatura în funcţie de timp şi coordonata z pentru x=y=0 101
Fig. 4.19 Temperatura în funcţie de coordonatele x şi y pentru z=1 la momentul final 101
Fig. 4.20 Temperatura în funcţie de timp şi coordonata z pentru x=y=0,5 102

Fig. 4.21 Temperatura în funcţie de coordonatele x şi y pentru z=0,5 la momentul final 102
Fig. 4.22 Temperatura în funcţie de timp şi coordonata x pentru y=0 şi z=1 103
Fig. 4.23 Temperatura în funcţie de coordonatele z şi y pentru x=0 la momentul final 103
Fig. 4.24 Temperatura în funcţie de timp şi coordonata x pentru z=y=0,5 104
Fig. 4.25 Temperatura în funcţie de coordonatele z şi y pentru x=0,5 la momentul final 104
Fig. 4.26 Imagini în spectru vizibil şi infraroşu suprapus peste vizibil obţinute cu camera Fluke Ti-25 105
Fig. 4.27 Imagine detaliu termic cu selecţie (a), poziţia sarcinii în cuptor (b), distribuţia de temperaturi obţinută experimental la 60 secunde (c) şi prin simulare (d) 106
Fig. 4.28 Imagine detaliu termic cu selecţie (a), poziţia sarcinii în cuptor (b), distribuţia de temperaturi obţinută experimental la 30 secunde (c) şi prin simulare (d) 107
Fig. 4.29 Imagine detaliu termic cu selecţie (a), poziţia sarcinii în cuptor (b), distribuţia de temperaturi obţinută experimental la 60 secunde (c) şi prin simulare (d) 108
Fig. 5.1 Integrarea prin metoda trapezelor 113
Fig. 5.2 Circuitul tipic de alimentare a magnetronului 115
Fig. 5.3 Măsurători pentru cuptorul Vortex MG8021TP–AP 116
Fig. 5.4 Măsurători pentru cuptorul Samsung G2736N 117
Fig. 5.5 Măsurători pentru cuptorul Candy CMG 2071 DS 117
Fig. 5.6 Măsurători pentru cuptorul SMC E70 TF-A 118
Fig. 5.7 Schema cuptorului Vortex MG8021TP–AP 118
Fig. 5.8 Interiorul cuptorului Vortex MG8021TP–AP 119
Fig. 5.9 Schemă de măsură 119
Fig. 5.10 Captură ecran pentru programul de monitorizare continuă 120
Fig. 5.11 Schemă de măsură pentru tensiunea şi curentul în primarul transformatorului 121
Fig. 5.12 Captură ecran pentru programul de monitorizare continuă 121
Fig. 5.13 Programul mwel, date la funcţionare normală (magnetron cald) 123
Fig. 5.14 Programul mwel, date la funcţionare în gol (magnetron nealimentat) 124
Fig. 5.15 Programul mwel, date la funcţionare în regim tranzitoriu (magnetron rece) 125
Fig. 5.16 Programul mwelim şi rezultatele 126
Fig. 5.17 Schemă simulare 127
Fig. 5.18 Model comandă tranzistoare IGBT 127
Fig. 5.19 Model de calcul curenţi armonici 127

1. Introducere

Încălzirea este probabil cel mai utilizat proces tehnologic în industrie. Este deseori folosită în industria alimentară, textilă, chimică şi a materialelor de construcţii pentru a usca, modifica proprietăţile fizice şi chimice ale materialelor, şi în multe alte scopuri. Şi totuşi este una din cele dificil de controlat tehnici, fiind lentă şi imprecisă atunci când se practică în maniera obişnuită: încălzind suprafaţa obiectului prin conducţie, convecţie, radiaţie sau o combinaţie a acestora. Chiar dacă se obţine o încălzire superficială care poate fi controlată, durata procesului este limitată de viteza de transmitere a căldurii în interiorul piesei. Această viteză este complet determinată de trei proprietăţi fizice ale materialului: căldura specifică, densitatea şi conductivitatea termică. Aceste proprietăţi sunt combinate într-un singur parametru, difuzivitatea termică [18], de care depinde în totalitate creşterea temperaturii în interior ca funcţie de timp, distanţa până la suprafaţă şi temperatura suprafeţei. Nu se poate face nimic pentru a accelera procesul de încălzire din momentul în care suprafaţa a atins o temperatură maximă impusă.

Pentru un obiect dat, viteza de încălzire prin metode clasice scade cu volumul obiectului. Procesul nu este doar lent, ci şi neuniform, deoarece suprafeţele şi în special muchiile şi colţurile sunt mult mai calde decât interiorul. În consecinţă, calitatea produsului tratat este variabilă şi deseori inferioară celei dorite.

Încălzirea imperfectă care rezultă din aceste dificultăţi este o cauză uzuală pentru rebuturi şi risipă de energie. Durata lungă a procesului are drept consecinţă rezervarea unei mari arii de lucru cuptoarelor. Cuptoarele mari răspund greu la schimbări de temperatură cerute, se încălzesc încet şi au capacităţi termice mari.

1.1. Încălzirea volumetrică prin mijloace electrice

Încălzirea în tot volumul este posibilă prin transmiterea energiei prin suprafaţa obiectului de încălzit sub formă electromagnetică, şi nu ca flux de căldură ca la încălzirea clasică. Viteza de încălzire în acest caz nu mai este limitată de temperatura suprafeţei şi difuzivitatea termică, iar uniformitatea temperaturii este mult îmbunătăţită. Durata de încălzire se poate reduce uneori de peste 100 de ori faţă de procedeele clasice, cu variaţia de temperatură în interiorul materialului de 10%.

Orice material poate fi încălzit electric în volum cu condiţia să nu fie nici foarte bun conductor nici foarte bun izolator. Nici o tehnică de încălzire electrică în volum nu este eficientă în toate cazurile. Există patru metode folosite în practică, clasificate după rezistivitatea electrică şi proprietăţile fizice ale materialului [2].

Încălzirea prin conducţie şi inducţie

Aceste metode sunt folosite pentru încălzirea metalelor cu rezistivitate scăzută şi implică trecerea unui curent de mare intensitate prin piesa de încălzit. Curentul poate să circule prin conexiuni directe (încălzire prin conducţie) sau piesa de încălzit formează secundarul în scurtcircuit al unui transformator coborâtor de tensiune (inducţie). Frecvenţele folosite sunt sub 60 Hz pentru conducţie şi între 50 Hz şi aproximativ 30 kHz pentru inducţie.

Încălzirea prin conducţie sau inducţie se folosesc în special în industria metalurgică şi construcţii de maşini pentru topire, sudare sau tratamente termice, uneori sub vid pentru a preveni oxidarea.

Încălzirea ohmică

Aceasta este o metodă de încălzire prin conducţie a lichidelor, şi constă în trecerea unui curent alternativ prin lichidul dintre doi electrozi. Soluţiile apoase în special sunt aproape întotdeauna suficient de bune conductoare pentru a permite o densitate de putere mare, deoarece ionii din sărurile dizolvate acţionează ca purtători de sarcină.

Instalaţiile de încălzire ohmică utilizează invariabil surse de frecvenţă industrială (50 sau 60 Hz) şi sunt extrem de eficiente în conversia energiei electrice în căldură, randamentul lor fiind de peste 95%. Este posibilă şi încălzirea lichidelor care conţin solide sau semi-solide, ca în industria alimentară, deoarece particulele solide au de obicei conductivitate electrică apropiată de cea a lichidului.

Încălzirea la frecvenţe radio (capacitivă)

Când materialul de încălzit are rezistivitate mare, tensiunea necesară pentru a transmite suficient curent pentru încălzire prin conducţie este dificil sau imposibil de obţinut. Această problemă se poate rezolva prin creşterea frecvenţei la 1÷100 MHz, cel mai adesea 27,12 MHz, una din frecvenţele alocate pentru acest scop.

Materialele tipice pentru această metodă sunt plastice, lemn, textile, hârtie, alimente şi ceramice. Obiectul de încălzit este plasat între sau trece printre două armături metalice, la care se aplică o tensiune înaltă (de ordinul kilovolţilor). Deoarece materialul de încălzit este un dielectric imperfect, structura se poate echivala cu un condensator ideal înseriat cu un rezistor care reprezintă sursa de căldură din interiorul materialului.

Încălzirea cu microunde

Încă din 1929, P. Debye ([3], [4]) a demonstrat faptul că dezvoltarea căldurii într-un material dielectric este determinată de două fenomene: încălzirea prin efect Joule şi încălzirea asociată fenomenului de histerezis dielectric şi polarizare în câmpuri electrice variabile în timp. Însă deoarece puterea dezvoltată prin histerezis dielectric este semnificativă numai la frecvenţe ridicate, aplicaţiile industriale ale procesului de încălzire a materialelor dielectrice au devenit posibile numai după 1940 când au putut fi realizate surse de microunde cu caracteristicile necesare.

Cercetări din timpul celui de-al 2-lea război mondial privind radarele de înaltă rezoluţie au condus la dezvoltarea tehnicilor de microunde (300 MHz ÷ 300 GHz), şi în special la apariţia magnetronului cu cavităţi (Univ. din Birmingham, Anglia, 1940) – un generator de microunde de mare putere şi randament excepţional.

Cunoaşterea proprietăţilor dielectrice ale materialelor ce urmează a fi procesate cu ajutorul microundelor este esenţială pentru proiectarea şi realizarea propriu-zisă a aplicatoarelor de microunde. Atât constanta dielectrică ε' cât şi factorul de pierderi ε'', şi implicit tangenta unghiului de pierderi tg δ, joacă un rol determinant în încălzirea cu microunde a materialelor dielectrice. Aceste mărimi definesc proprietăţile dielectrice ale unui material şi caracterizează comportamentul acestuia sub influenţa unui câmp de înaltă frecvenţă, fiind dependente de frecvenţă, temperatură şi umiditate.

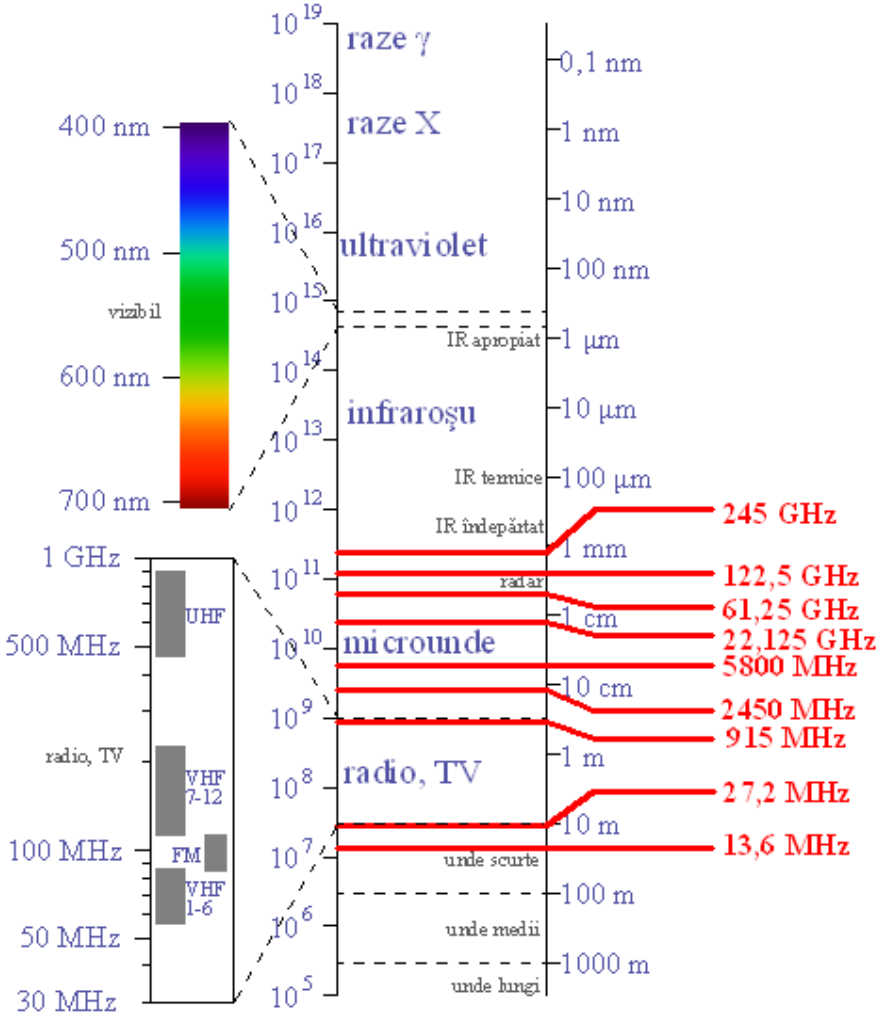


Fig. 1.1. Spectrul electromagnetic

Lungimile de undă ale microundelor folosite la încălzire sunt comparabile şi în general proporţionale cu dimensiunile materialului de procesat. Dimensiunile relative ale lungimii de undă şi ale cavităţii la o anumită frecvenţă sunt astfel alese încât lungimea cavităţii să fie multiplu al lungimii de undă. Adâncimea de pătrundere în material depinde şi ea de frecvenţă, iar utilizarea unei frecvenţe prea mari poate anula cel mai important avantaj al încălzirii cu microunde: faptul că este volumetrică. În fine, pentru a evita interferenţele cu echipamente de comunicaţii, frecvenţa trebuie aleasă dintre cele standard alocate pentru încălzirea cu microunde. Ecranarea trebuie să fie mult mai bună dacă se folosesc alte frecvenţe decât cele autorizate într-o anumită zonă. De exemplu, banda 890–960 MHz este folosită pentru telefonie mobilă (GSM) în Europa şi nu pentru încălzire. În acest caz instalaţiile de încălzire pe 900 MHz trebuie să fie ecranate contra interferenţelor în

timp ce cele care funcţionează la 2450 MHz trebuie ecranate doar pentru a nu avea emisii periculoase din punct de vedere fiziologic, nivelul permis de emisie fiind mult mai mare. Benzile de frecvenţe alocate de Uniunea Internaţională pentru Telecomunicaţii pentru aplicaţii industriale, ştiinţifice şi medicale (uzual abreviat ISM) sunt indicate în tabelul următor [1]. Banda de 915 MHz este folosită numai pe continentul american.

Tab. 1.1. Benzile de frecvenţe ISM

Banda	Frecvenţa centrală
902 ÷ 928 MHz	915 MHz
2400 ÷ 2500 MHz	2450 MHz
5275 ÷ 5875 MHz	5800 MHz
24 ÷ 24,25 GHz	24,125 GHz
61 ÷ 61,5 GHz	61,25 GHz
122 ÷ 123 GHz	122,5 GHz
244 ÷ 246 GHz	245 GHz

În mod paradoxal, deşi banda de 2450 MHz este alocată pentru încălzire peste tot în lume, şi este cea mai folosită în acest scop, multe sisteme electronice de larg consum folosesc exact această frecvenţă pentru comunicaţii. De obicei este vorba de telecomunicaţii pe distanţă scurtă: telefoane cu receptor fără fir, dispozitive BlueTooth, reţele wireless IEEE 802.11b/g etc. Motivaţia alegerii acestei benzi pentru comunicaţii este aceea că se pot emite puteri destul de mari fără a mai cere autorizare, iar emisiile cuptoarelor cu microunde sunt nemodulate şi pot fi filtrate destul de uşor. Chiar şi aşa, un cuptor cu microunde cu scăpări mari poate compromite o reţea wireless.

Chiar dacă dimensiunile materialului nu limitează puterea – şi deci viteza de încălzire, caracteristicile dielectrice ale acestuia pot să introducă limitări. Dacă pierderile cresc cu temperatura, atunci orice neomogenitate de temperatură tinde să se amplifice – fenomen cunoscut ca ambalare termică. În acest caz, puterea trebuie limitată pentru a permite omogenizarea temperaturii prin conducţie termică. Uneori pierderile scad cu temperatura, şi atunci se produce o stabilizare termică ce permite aplicarea unei densităţi de putere mai mare. Un exemplu de ambalare termică este la dezgheţarea alimentelor, deoarece apa în stare lichidă are pierderi mult mai mari decât gheaţa. Stabilizarea termică se obţine de exemplu la uscarea lemnului.

1.2. Evoluţia sistemelor de încălzire cu microunde

Cuptoare destinate acestui scop se produc încă din 1945, concepţia originală a acestui echipament fiind atribuită lui Percy Spencer (*Raytheon*, 1945) [5]. Astfel, el a inaugurat o nouă eră în prepararea alimentelor şi a pus la punct o metodă care a schimbat radical, în toată lumea, concepţia asupra aparaturii de uz casnic. Mai mult sau mai puţin simultan şi alţii au fost interesaţi de aceste aspecte. De exemplu, s-a arătat de către Kinn şi Marcum (1947) că diferite mărimi şi forme de cavităţi folosite la încălzire pot fi obţinute prin modificarea formei terminaţiei ghidului de undă. Efectul încălzirii şi preparării alimentelor cu microunde a fost studiat de Morse şi Rivercomb (1947), [5]. Ei au fost primii care au descris problemele referitoare la ambalarea termică.

Primele eforturi în această direcţie au fost iniţiate de firmele americane Raytheon, General Electric şi Westinghouse. Raytheon a efectuat experimente cu

echipamente la frecvenţa de 2450 MHz, în timp ce General Electric a preferat să lucreze la frecvenţa de 915 MHz.

Dezvoltarea eforturilor făcute de Raytheon şi concentrate asupra echipamentelor de tip "Radarange" rămâne cel mai important program până în prezent, [7]. Evoluţia aplicatoarelor cu microunde a cunoscut importante progrese. Este cunoscut un aplicator cu microunde încă înainte de 1945, utilizat de aviaţia americană, aplicator care era capabil să dezgheţe şi să încălzească 250 de grame de carne de la –12°C la 77°C, într-un minut. Sunt cunoscute mai multe versiuni ale acestor încălzitoare cu microunde.

În 1946 apare un prim model de "încălzitor de sandvişuri", cu cavitate demontabilă lateral. Acest model anticipează cuptoarele fixe, cu durată scurtă a ciclului de funcţionare, care se întâlnesc astăzi în echipamentele de microunde.

În 1947 a apărut modelul 1132, denumit "white range" [5]. Acesta avea o putere de aproximativ 1,6 kW, magnetron răcit cu apă şi magnet permanent, era un echipament greu. Microundele intrau printr-un ghid de undă de tip U într-o fereastră de sticlă Pyrex, aşezată pe capacul superior, chiar deasupra punctului central. Această instalaţie era prevăzută cu un agitator, destinat să omogenizeze distribuţia căldurii. Agitatorul era realizat din două piese de crom având forma a două semisfere scobite, aşezate spate în spate, întreg ansamblul fiind montat în partea superioară a cuptorului şi rotindu-se cu 2-3 rotaţii pe secundă, acţionat de un motor, plasat în compartimentul superior. Magnetronul a fost montat cu antena în poziţie verticală, într-un ghid de undă, amplasat în partea de jos a aplicatorului şi era alimentat în curent continuu. Acest model era un aplicator mic, cu putere de ieşire mare, care dădea un câmp de microunde de mare densitate şi cu o creştere exagerată în distribuţia căldurii.

Următorul aplicator, în ordine cronologică, a fost modelul 1150, care utiliza două magnetroane răcite cu aer, de tip QK-312, având dimensiunile de gabarit mult mai mari: 534x737x660 mm; cele două magnetroane fiind montate direct pe plafonul aplicatorului, [6]. Acest aplicator a apărut înainte de 1951 şi a fost utilizat până la apariţia liniei "United States". Ceea ce îl deosebea de alte aplicatoare era, în primul rând, puterea de ieşire neobişnuită - peste 2 kW teoretic. Puterea utila se situa între 1,1 şi 1,7 kW. Cele două magnetroane erau utilizate sub puterea lor maximă. Operatorul putea să schimbe modul de funcţionare, în funcţie de cerinţele impuse de alimente şi putea sa coboare sub 200 mA sau să urce peste 300 mA pentru fiecare magnetron. Energia furnizată aplicatorului era dependentă de curentul magnetronului care, la rândul lui, era determinat de câmpul magnetic şi de tensiunea aplicată magnetronului. Câmpul magnetic era permanent şi în acest caz curentul era controlat prin modificarea tensiunii aplicate la magnetron. Operatorul putea să lucreze, obţinând timpi de încălzire mai mici şi putând astfel să prepare o cantitate mai mare de alimente. În consecinţă, acest model introducea un echipament cu putere mai mare, volum adecvat şi control flexibil al procesului.

În Europa începând cu anii 1960 au apărut firme având ca domeniu de activitate utilizarea microundelor. Primele firme în acest domeniu au fost Philips şi Atlas Electronik. Aplicaţiile industriale au fost continuate de către firme mai mici, astfel au apărut Industries Micro-ondes Internationales în Franţa, Calores AB în Suedia, etc.

În anul 1966, Litton Industies Atherton Division a utilizat două aplicatoare cu puteri de 50 şi respectiv 80 KW, la 2,45 GHz, utilizate pentru gătirea cărnii de pasăre. Aceste cercetări au fost extinse şi în alte domenii cum ar fi tratamentul spumei de poliuretan, evaporarea alcoolului, uscarea finală a cipsurilor, etc.

În anul 1968, în Japonia au debutat primele aplicaţii industriale care au utilizat microundele. Acestea au fost utilizate pentru distrugerea mucegaiului, uscarea produselor din orez, decongelare.

În anul 1964, Ernie Okress, a organizat în Florida, un simpozion destinat aplicaţiilor cu microunde numit "Microwave Power Application". In cadrul acestui simpozion a fost creat un organism specializat numit "International Microwave Power Institut" (IMPI), care vizează promovarea tuturor aplicaţiilor cu microunde, cu excepţia telecomunicaţiilor şi radarului. Anual sunt organizate şi alte simpozioane unde sunt prezentate cele mai recente cercetări în domeniul microundelor, care ulterior sunt publicate în reviste de specialitate ca de exemplu: Journal of Microwave Power şi Microwave Energy Applications Newsletter.

Totuşi putem spune că piaţa industrială de microunde a cunoscut o dezvoltare destul de lentă, deoarece multe dintre aplicaţii nu au avut un succes imediat, cercetările fiind continuate de către firme mici. Această dezvoltare lentă se datorează în primul rând costului investiţiei, care în cazul aplicaţiilor industriale este de 2500–3000 $/kW putere instalată [7], în timp ce pentru un aplicator casnic costul este de 350 $/kW putere instalată. Cu toate aceste inconveniente microundele sunt preferate în multe din aplicaţiile unde nu pot fi utilizate metodele tradiţionale.

1.3. Stadiul actual

Utilizarea microundelor pentru uscarea materialelor are câteva avantaje destul de importante: sursele de căldură sunt distribuite în întreg volumul materialului; modificarea umidităţii materialului influenţează parametrii dielectrici, astfel încât procesul de uscare capătă "stabilitate", pierderile descresc odată cu umiditatea.

Din păcate, modelarea matematică a procesului de uscare în câmp de microunde este o problemă destul de dificilă. Problemele de câmp electromagnetic, de câmp termic şi de umiditate sunt cuplate. În plus, obţinerea unor rezultate corecte obligă utilizarea unor modele 3D.

Numeroase firme de prestigiu din străinătate (S.U.A. – General Electric, Cober Electronics Inc.; Franţa – CNRS, EDF, SAIREM; Japonia – Toshiba; Olanda – Philips, etc.) fac cercetări cu privire la proprietăţile materialelor dielectrice procesate în câmp de microunde, modelarea proceselor termice, elaborarea unor scheme de automatizare şi control a parametrilor de procesare, concepţia şi execuţia echipamentelor la nivel de laborator [4], [5], [6], [14]. In prezent există preocupări în valorificarea activităţii de cercetare a specialiştilor implicaţi în această activitate [11].

Măsurarea proprietăţilor materialelor dielectrice aflate în câmp de microunde a fost făcută de către Bengtsson şi Risman, 1974 [9]. Studii asupra proprietăţilor dielectrice ale materialelor au fost făcute mai recent şi de către Paltin, 1992.

Turner, 1984 [15], a oferit o oarecare clarificare asupra modurilor de propagare existente în interiorul aplicatoarelor, luând în considerare inexistenţa unor moduri datorită joncţiunii dintre ghid şi cavitate. Totuşi aceste informaţii nu au fost suficiente pentru a afla modul optim de amplasare a unui dielectric în interiorul unui aplicator.

Kashiwa, 1991 [17], a folosit o reţea de cochilii sferice din plastic subţire umplute cu apă pentru a schiţa distribuţia câmpului în interiorul aplicatorului multimod.

S. Lefeuvre, 1993, [16] a efectuat studii asupra proprietăţilor materialelor dielectrice şi modul cum acestea influenţează procesarea în câmp de microunde. A studiat modul de adaptare ghid - cavitate.

O analiză a aplicatoarelor multimod a fost prezentată de către R. J. Meredith în 1994. Acesta arată modul de superpoziţionare a undelor după trei direcţii ortogonale, dezvoltând cu succes o tehnică simplă de măsurare a modurilor într-o cavitate goală. Acest studiu a fost apoi extins de către Chow Ting Chan şi H. C. Reader, 1995 [5], la o cavitate încărcată cu o sarcină, folosind un soft de element finit MSC.

D.C. Dibben, 1996 [13] a studiat modelarea numerică şi experimentală a mai multor tipuri de aplicatoare, informaţii care ulterior au fost folosite la dezvoltarea unor metode complexe de calcul.

H.C. Reader, 1997 [7], a dezvoltat o metodă de detectare a distribuţiei câmpului electric pe suprafaţa interioară a aplicatorului. Această metodă utilizează un analizor de reţea cu două deschideri. O deschidere este folosită pentru excitarea cavităţii iar cealaltă pentru a testa pereţii ortogonali ai cavităţii. Această metodă a fost apoi extinsă şi la studiul aplicatoarelor multimod. Rezultatele obţinute au fost foarte utile, deoarece au oferit un mod rapid şi ieftin de examinare a câmpului electromagnetic în structurile cu microunde.

Metaxas, 2001 [11], a prezentat o metodă laborioasă de analiză a câmpului electromagnetic şi a distribuţiei de putere în mai multe configuraţii de aplicatoare multimod utilizând metoda elementului finit.

2. Încălzirea în câmp de microunde

2.1. Unde electromagnetice

2.1.1. Ecuaţiile de bază ale câmpului electromagnetic. Unda plană.

În ipoteza unui mediu dielectric imobil, omogen, liniar şi izotrop, ecuaţiile lui Maxwell sunt:

$$rot\vec{H} = \frac{\partial \vec{D}}{\partial t} = \varepsilon \frac{\partial \vec{E}}{\partial t} \tag{2.1}$$

$$rot\vec{E} = -\frac{\partial \vec{B}}{\partial t} = -\mu \frac{\partial \vec{H}}{\partial t} \tag{2.2}$$

$$div\vec{B} = 0 \quad \Rightarrow \quad div\vec{H} = 0 \tag{2.3}$$

$$div\vec{D} = 0 \quad \Rightarrow \quad div\vec{E} = 0 \tag{2.4}$$

Dacă expresiei (2.1) i se aplică operatorul rotor se obţine:

$rot\left(rot\vec{H}\right) = rot\left(\varepsilon \frac{\partial \vec{E}}{\partial t}\right)$ sau

$grad\left(div\vec{H}\right) - \Delta\vec{H} = \varepsilon \frac{\partial}{\partial t}\left(rot\vec{E}\right)$ şi ţinând cont de (2.2) rezultă:

$$\Delta\vec{H} - \mu\varepsilon \frac{\partial^2 \vec{H}}{\partial t^2} = 0 \tag{2.5}$$

Analog, aplicând operatorul rotor relaţiei (2.2) şi ţinând cont de (2.1) rezultă:

$$\Delta\vec{E} - \mu\varepsilon \frac{\partial^2 \vec{E}}{\partial t^2} = 0 \tag{2.6}$$

Notând cu $c = \frac{1}{\sqrt{\mu\varepsilon}}$ viteza de propagare a luminii în mediul dielectric considerat, relaţiile (2.5) şi (2.6) devin:

$$\Delta\vec{H} - \frac{1}{c^2} \frac{\partial^2 \vec{H}}{\partial t^2} = 0 \tag{2.7}$$

$$\Delta\vec{E} - \frac{1}{c^2} \frac{\partial^2 \vec{E}}{\partial t^2} = 0 \tag{2.8}$$

relaţii care constituie ecuaţiile undelor electromagnetice în mediul dielectric.

Să presupunem că intensitatea câmpului electric $\vec{E}$ este paralelă cu axa 0x, $\vec{E} = \vec{i}\,E$ şi depinde numai de coordonata z. Deci $\vec{E} = \vec{E}(z,t)$, de unde:

$\Delta\vec{E} = \vec{i}\,\frac{\partial^2 E}{\partial z^2}$, iar ecuaţia (2.8) devine:

$$\frac{\partial^2 E}{\partial z^2} = \frac{1}{c^2}\frac{\partial^2 E}{\partial t^2} \tag{2.9}$$

cu soluţia:

$$E(z,t) = E_d\left(t - \frac{z}{c}\right) + E_i\left(t + \frac{z}{c}\right) \tag{2.10}$$

$E_d(t - z/c)$ este o undă a intensităţii câmpului electric care se propagă în sensul pozitiv al axei 0z cu viteza de propagare c. $E_i(t + z/c)$ este o undă care se propaga în sensul negativ al axei 0z cu aceeaşi viteză c. Formulele analitice pentru E_d şi E_i se determină din condiţiile iniţiale şi condiţiile la limită.

Aceleaşi observaţii sunt valabile şi pentru intensitatea câmpului magnetic.

$$H(z,t) = H_d\left(t - \frac{z}{c}\right) + H_i\left(t + \frac{z}{c}\right) \tag{2.11}$$

Deoarece $\vec{E}$ şi $\vec{H}$ există simultan rezultă că şi undele $E(z,t)$ şi $H(z,t)$ există simultan. Spunem că perechea (E_d, H_d) formează unda electromagnetică directă şi perechea (E_i, H_i) formează unda electromagnetică inversă.

Unda electromagnetică plană

Presupunem că intensitatea câmpului electric şi intensitatea câmpului magnetic depind numai de timp şi coordonata z.

$$\begin{cases} \vec{E} = \vec{E}(z,t) \\ \vec{H} = \vec{H}(z,t) \end{cases} \tag{2.12}$$

Din relaţia (2.1) avem:

$$rot\vec{H} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 0 & 0 & \frac{\partial}{\partial z} \\ H_x & H_y & H_z \end{vmatrix} = \vec{i}\,\varepsilon\frac{\partial E_x}{\partial t} + \vec{j}\,\varepsilon\frac{\partial E_y}{\partial t} + \vec{k}\,\varepsilon\frac{\partial E_z}{\partial t} \text{ sau}$$

$$-\vec{i}\,\frac{\partial H_y}{\partial z} + \vec{j}\,\frac{\partial H_x}{\partial z} = \vec{i}\,\varepsilon\frac{\partial E_x}{\partial t} + \vec{j}\,\varepsilon\frac{\partial E_y}{\partial t} + \vec{k}\,\varepsilon\frac{\partial E_z}{\partial t}$$

şi identificând termenii:

$$-\frac{\partial H_y}{\partial z} = \varepsilon\frac{\partial E_x}{\partial t} \tag{2.13}$$

$$\frac{\partial H_x}{\partial z} = \varepsilon\frac{\partial E_y}{\partial t} \tag{2.14}$$

$$0 = \varepsilon \frac{\partial E_z}{\partial t} \tag{2.15}$$

Analog, din (2.2) se obţine:

$$\frac{\partial E_y}{\partial z} = \mu \frac{\partial H_x}{\partial t} \tag{2.16}$$

$$\frac{\partial E_x}{\partial z} = -\mu \frac{\partial H_y}{\partial t} \tag{2.17}$$

$$0 = -\mu \frac{\partial H_z}{\partial t} \tag{2.18}$$

Din (2.3) şi (2.4) se obţine:

$$\frac{\partial E_z}{\partial z} = 0 \tag{2.19}$$

$$\frac{\partial H_z}{\partial z} = 0 \tag{2.20}$$

Considerând relaţiile (2.15) şi (2.19) rezultă că E_z este constantă în timp şi spaţiu. O componentă care satisface asemenea condiţii nu poate fi o mărime care se propagă, deci o undă electromagnetică, ea ar putea fi doar o componentă a unui câmp electrostatic. Considerăm că înlăturăm toate cauzele de generare a câmpului electrostatic, deci $E_z = 0$. Similar se raţionează şi pentru componenta H_z, pe baza relaţiilor (2.18) şi (2.20), concluzionând că şi $H_z = 0$.

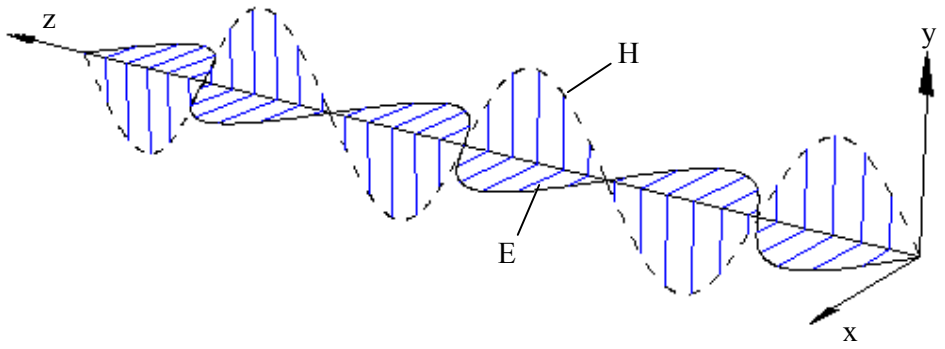


Fig. 2.1. Undă electromagnetică plană în perspectivă

Unda electromagnetică nu are componente după direcţia z, deci este plană, $\vec{E}$ şi $\vec{H}$ fiind conţinuţi în plane perpendiculare pe direcţia de propagare z.

Observând ecuaţiile (2.13), (2.17) şi (2.14), (2.17) se constată că nici una nu leagă pe E_x cu H_x sau pe E_y cu H_y. Componentele interdependente sunt (E_x, H_y) şi (E_y, H_x) dar cele două perechi sunt independente una faţă de cealaltă. Fiecare dintre aceste perechi reprezintă o undă electromagnetică plană polarizată liniar.

Aşadar, o undă electromagnetică plană oarecare poate fi descompusă în două unde liniar polarizate independente una faţă de cealaltă. Pentru simplificarea raţionamentelor vom considera numai perechea (E_x, H_y), cealaltă fiind nulă ($E_y = 0$, $H_x = 0$). Simplificăm notaţiile:

$$\begin{cases} \vec{E} = \vec{i}\,E \\ \vec{H} = \vec{j}H \end{cases} \tag{2.21}$$

Ecuaţiile (2.7) şi (2.8) devin:

$$\frac{\partial^2 H}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 H}{\partial t^2} \tag{2.22}$$

$$\frac{\partial^2 E}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 E}{\partial t^2} \tag{2.23}$$

având soluţii de forma:

$$E(z,t) = E_d\left(t - \frac{z}{c}\right) + E_i\left(t + \frac{z}{c}\right) \tag{2.24}$$

$$H(z,t) = H_d\left(t - \frac{z}{c}\right) + H_i\left(t + \frac{z}{c}\right) \tag{2.25}$$

Introducând soluţiile (2.24) şi (2.25) în ecuaţiile (2.13) şi (2.17) şi identificând între ei termenii funcţiei de (t–z/c) respectiv (t+z/c) se obţin:

$$\frac{E_d}{H_d} = \sqrt{\frac{\mu}{\varepsilon}} \tag{2.26}$$

$$\frac{E_i}{H_i} = -\sqrt{\frac{\mu}{\varepsilon}} \tag{2.27}$$

Se notează cu $\zeta = \sqrt{\frac{\mu}{\varepsilon}}$ impedanţa de undă şi în concluzie:

$$\frac{E_d}{H_d} = \zeta\ ,\ \frac{E_i}{H_i} = -\zeta \tag{2.28}$$

În cazul particular al vidului $\zeta_0 = \sqrt{\frac{\mu_0}{\varepsilon_0}} = 377\Omega$.

2.1.2. Reflexia şi refracţia pe o frontieră plană

Unda electromagnetică descrisă până acum se propagă într-un spaţiu infinit, fără frontiere. Ecuaţiile câmpului electromagnetic au fost rezolvate în absenţa oricărei frontiere. Dacă trebuie să considerăm condiţiile de frontieră, atunci nu poate să existe o undă plană. O undă plană apare practic atunci când sursa se găseşte la o distanţă foarte mare de observator şi se poate neglija curbura frontului de undă. Lumina poate fi considerată în multe cazuri o undă plană, dar la lungimea de undă a

microundelor doar un satelit sau un alt obiect aflat în spaţiul cosmic pot fi considerate surse de unde plane.

Condiţiile de frontieră

La suprafaţa de separaţie dintre două medii (fig. 2.2), pentru câmpul electric condiţiile de trecere sunt:

$$D_{2n} - D_{1n} = \rho_s$$
$$\vec{E}_{1t} = \vec{E}_{2t} \qquad (2.29)$$

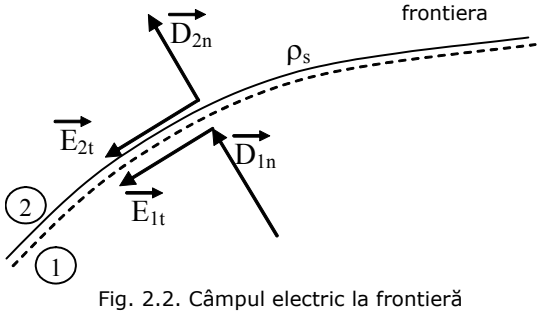


Fig. 2.2. Câmpul electric la frontieră

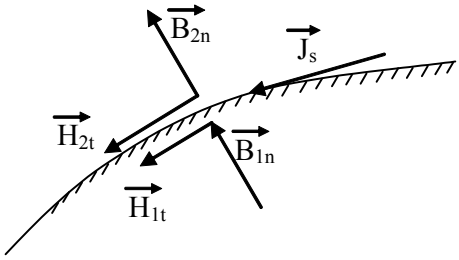


Fig. 2.3. Câmpul magnetic la frontieră

Pentru câmpul magnetic (fig. 2.3), condiţiile de trecere sunt:

$$\vec{H}_{2t} - \vec{H}_{1t} = \vec{J}_s$$
$$B_{1n} = B_{2n} \qquad (2.30)$$

unde $\vec{J}_s$ este pânza superficială de curent la suprafaţa dintre cele două medii.

În cazul particular când materialul 1 este perfect conductor ($\sigma \to \infty$), deoarece $\vec{J}_s = \sigma\vec{E}_t$, rezultă $E_t = 0$, deci intensitatea câmpului electric este perpendiculară pe suprafaţă. Această condiţie este bine satisfăcută de un perete metalic. Se obişnuieşte să se spună că este o condiţie de scurtcircuit.

În cazul unui perete perfect magnetic ($\mu \to \infty$) se obţine că $\vec{H}$ este perpendicular pe suprafaţă. Deşi în probleme de microunde nu se întâlnesc materiale cu permitivitate mare, condiţia este bună pentru a considera simetrii geometrice. Această condiţie mai este numită "conectare în gol".

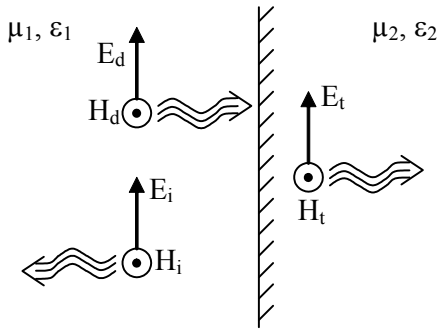


Fig. 2.4. Undă parţial reflectată

În cazul unei unde parţial reflectate la suprafaţa de separare a două materiale (fig. 2.4), am notat:

- E_d, H_d – unda directă
- E_i, H_i – unda inversă (reflectată)
- E_t, H_t – unda transmisă

Impedanţele de undă pentru cele două medii sunt:

$$\zeta_1 = \sqrt{\frac{\mu_1}{\varepsilon_1}} \text{ şi } \zeta_2 = \sqrt{\frac{\mu_2}{\varepsilon_2}}$$

Avem relaţiile:

$$\begin{aligned} E_d &= \zeta_1 H_d \\ E_i &= -\zeta_1 H_i \\ E_t &= \zeta_2 H_t \end{aligned} \tag{2.31}$$

Dacă studiem fenomenul care are loc la suprafaţă, ţinând cont de condiţiile de frontieră (2.29) şi (2.30), rezultă:

$$E_d + E_i = E_t \text{ sau } \zeta_1 H_d - \zeta_1 H_i = \zeta_2 H_t \tag{2.32}$$

$$H_d + H_i = H_t \tag{2.33}$$

Din relaţiile (2.32) şi (2.33) avem:

$$\begin{cases} H_d - H_i = \dfrac{\zeta_2}{\zeta_1} H_t \\ H_d + H_i = H_t \end{cases} \tag{2.34}$$

de unde eliminând pe H_t rezultă:

$$\frac{H_d}{H_i} = \frac{\zeta_1 + \zeta_2}{\zeta_1 - \zeta_2} \tag{2.35}$$

La fel şi pentru $\frac{E_d}{E_i}$. Valorile maxime şi minime ale intensităţii câmpului electric sunt:

$$E_{\max} = E_d + E_i$$
$$E_{\min} = E_d - E_i \qquad (2.36)$$

Aceste valori conduc la un coeficient de undă staţionară:

$$VSWR = \frac{E_{\max}}{E_{\min}} = \frac{E_d + E_i}{E_d - E_i} = \frac{\zeta_2}{\zeta_1} \qquad (2.37)$$

2.2. Câmpul electric în medii dielectrice

Mediile dielectrice se caracterizează prin faptul că nu conţin decât un număr neînsemnat de purtători liberi de sarcini electrice care să se poată deplasa pe distanţe nelimitate sub acţiunea câmpului electric; prin urmare, au conductivitatea electrică egală practic cu zero.

Sub acţiunea câmpului electric exterior, electronii din mediul dielectric, ce gravitează în jurul unor formaţii structurale (atomi, molecule), execută numai o deplasare pe o distanţă limitată, denumită deplasare elastică, fără să se desprindă de formaţiunile cărora le aparţin. Dacă intensitatea câmpului electric din dielectric depăşeşte o anumită limită, electronii sunt "desprinşi" din formaţii, devenind liberi, respectiv mediul îşi pierde calităţile izolatoare, devenind conductor.

Calităţile materialelor izolatoare folosite în industria electrotehnică sunt determinate în primul rând de comportarea lor în câmpul electric.

2.2.1. Polarizarea dielectricilor în câmpul electric

Un atom sau o moleculă dintr-un material izolator sunt neutre din punct de vedere electric în sensul că ele conţin acelaşi număr de sarcini electrice elementare negative şi pozitive. Electronii ce aparţin formaţiunilor structurale ale substanţei izolante execută o mişcare permanentă; sarcinile electronilor în mişcare se manifestă ca şi cum ar fi amplasate într-un centru echivalent de acţiune a sarcinilor negative, care, în lipsa câmpului electric exterior, se suprapune sau nu peste centrul de acţiune a sarcinilor pozitive situate în nucleele atomilor.

O moleculă la care, în lipsa câmpului electric exterior, centrul echivalent de acţiune al sarcinilor electrice negative se suprapune peste centrul de acţiune al sarcinilor electrice pozitive, se numeşte moleculă nepolară, având momentul electric $p = 0$. În caz contrar, când cele două centre nu se suprapun, molecula este polară, având momentul electric $p \neq 0$.

Cel mai simplu atom nepolar este cel al hidrogenului, format din nucleul cu sarcina electrică elementară pozitivă $Q_+=q$ şi un electron cu sarcina electrică elementară negativă $Q_-=-q$. În lipsa câmpului electric exterior, electronul execută aproximativ o mişcare circulară în jurul nucleului, astfel încât centrul echivalent de acţiune al sarcinii sale coincide cu centrul sarcinii pozitive din nucleu.

Dacă atomul de hidrogen se află într-un câmp electric exterior de intensitate E_0, orbita electronului se deformează, devenind eliptică, astfel încât centrul echivalent de acţiune al sarcinii negative nu se mai suprapune peste cel al sarcinii pozitive. Se spune că atomul de hidrogen s-a polarizat electric, comportându-se ca un dipol electric. Fenomenul de constituire şi orientare a dipolilor la un dielectric cu molecule polare, situat într-un câmp electric exterior, se numeşte polarizare electrică de deformare.

La dielectricii cu molecule polare, în prezenţa unui câmp electric exterior, are loc fenomenul de polarizare electrică a unui dielectric adică a dipolilor existenţi. In general, polarizarea electrică a unui dielectric constă atât din polarizarea de deformare, cât şi din polarizarea de orientare.

Polarizarea electrică a unui dielectric poate fi temporară sau permanentă. În primul caz, starea de polarizare se menţine numai în prezenţa unui câmp electric exterior, iar în al doilea caz această stare se menţine independent de existenţa câmpului electric exterior.

Polarizarea temporară poate fi:

- polarizare de deplasare (electronică sau ionică);
- polarizare de orientare dipolară.

Polarizarea de deplasare electronică reprezintă deplasarea limitată şi elastică, sub acţiunea câmpului electric, a învelişurilor electronice ale atomilor dielectricului.

Acest mecanism de polarizare este o proprietate universală a materiei, fiind uneori denumit şi polarizare optică datorită faptului că intervine în propagarea prin dielectric a câmpurilor electromagnetice de frecvenţe foarte ridicate, situate în domeniile infraroşu, vizibil sau ultraviolet.

Polarizarea de deplasare ionică reprezintă deplasarea limitată şi elastică a ionilor dielectricului sub acţiunea câmpului electric. Acest mecanism de polarizare este specific dielectricilor cu legături preponderent ionice.

Polarizarea de orientare dipolară reprezintă orientarea pe direcţia câmpului electric aplicat a momentelor electrice care, în absenţa câmpului, sunt distribuite aleatoriu sub influenţa agitaţiei termice. Această polarizare este tipică materialelor cu molecule polare – care prezintă moment electric propriu.

În cazul materialelor neomogene, mai există un mecanism de polarizare, numit de neomogenitate (Maxwell-Wagner). Acesta este determinat de apariţia unor sarcini pe suprafeţele de separare ale părţilor omogene din dielectricii neomogeni.

În aplicaţiile microundelor efectul polarizărilor de deplasare este neglijabil în raport cu cel al polarizării de orientare şi de neomogenitate. Aceasta din urmă, pe lângă efectul pozitiv (creşterea pierderilor) are şi un posibil efect nedorit, şi anume limitarea densităţii de putere care poate fi aplicată datorită efectului de arc electric.

Polarizarea de orientare (dipolară)

Mecanismul polarizării dipolare sau de orientare se manifestă în gama de frecvenţe a microundelor şi are la bază rotaţia dipolilor permanenţi în câmpul electric aplicat. Aceşti dipoli au tendinţa de orientare - reorientare sub influenţa modificării polarităţii câmpului electric ceea ce conduce la creşterea polarizării. Agitaţia termică determină o mişcare dezordonată a moleculelor, astfel că suma momentelor electrice ale moleculelor este nulă. Dacă se aplică un câmp electric exterior ($E \neq 0$) dipolii tind să se orienteze în direcţia câmpului. Acestei tendinţe de orientare i se opune agitaţia termică. Ca urmare se realizează în medie o rotire (orientare) incompletă a dipolilor în direcţia câmpului, polarizaţia fiind egală cu suma momentelor electrice ale dipolilor conţinuţi în material şi proporţională cu câmpul exterior [3], [4].

La frecvenţe joase, timpul necesar câmpului electric să-şi schimbe direcţia este mai lung decât timpul de răspuns al dipolilor şi polarizaţia dipolară este în fază cu câmpul electric. Câmpul electric furnizează energia necesară orientării moleculelor în câmp. O parte din energie este consumată de mişcarea browniană, de

câte ori un dipol este deplasat din locul său prin ciocnire şi apoi realiniat. Energia transferată este mică, temperatura crescând foarte puţin, [10].

În gama frecvenţelor corespunzătoare microundelor, timpul în care câmpul electric îşi schimbă alternanţa este aproximativ acelaşi cu durata de răspuns a dipolilor. Aceştia se rotesc datorită cuplului exercitat de câmp şi polarizaţia rezultantă rămâne în urmă în raport cu intensitatea câmpului electric fenomen numit relaxare dielectrică sau histerezis dielectric [4], [9].

Considerând un câmp electric uniform de înaltă frecvenţă aplicat unui dielectric:

$$E = E_{\max} \sin \omega t \tag{2.38}$$

Vectorul polarizaţie are următoarea formă:

$$P = P_{\max} \sin(\omega t - \varphi) \tag{2.39}$$

unde: φ - defazajul dintre vectorul polarizaţie P şi vectorul intensităţii câmpului electric E, aşa cum rezultă şi din figura 2.5.

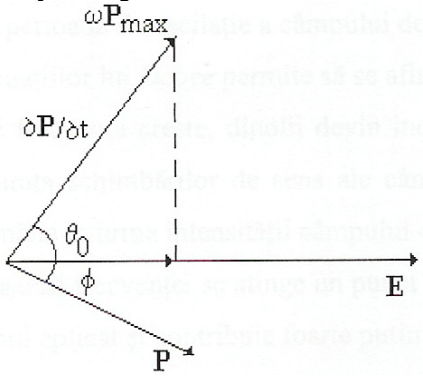


Fig. 2.5. Diagrama fazorială a vectorului polarizaţie P şi a intensităţii câmpului electric E

Curentul rezultant $\partial P/\partial t$ determinat de redistribuţia sarcinilor din material dat de:

$$\frac{\partial P}{\partial t} = \omega P_{\max} \cos(\omega t - \varphi) \tag{2.40}$$

are o componentă în fază cu intensitatea câmpului E, conducând la disiparea puterii în interiorul materialului ceea ce are ca efect încălzirea acestuia.

Puterea medie disipată în unitatea de volum este dată de:

$$P_{med} = \frac{1}{2} E_{\max} P_{\max} \omega \sin(\varphi) = \frac{1}{2} E_{\max} P_{\max} \cos(\theta_0) \tag{2.41}$$

unde cos θ_0 este factorul de putere.

Daca nu ar exista un defazaj între polarizaţie şi câmpul electric nu s-ar mai disipa putere în material şi astfel a fost introdusă noţiunea de *"pierderi"* care se referă la absorbţia în material de energie provenită de la câmpul electromagnetic de înaltă frecvenţă.

O abordare clasică a comportării dipolilor permanenţi dintr-un dielectric căruia i se aplică un câmp alternativ aparţine lui Debye, care a arătat că polarizarea

de orientare este rezultatul rotaţiei dipolilor de formă sferică într-un mediu vâscos în care se manifestă forţe de frecare [3], [4]. Debye a dedus următoarea ecuaţie:

$$\underline{\varepsilon}_d^* = \varepsilon_d' - j\varepsilon_d'' = \varepsilon_\infty + \frac{\varepsilon_S - \varepsilon_\infty}{1 + j\omega\tau} \tag{2.42}$$

unde:

- ε_S şi ε_∞ sunt valori statice ale constantei dielectrice;
- τ este timpul de relaxare dielectrică ce caracterizează rata creşterii şi descreşterii polarizaţiei;
- $\omega = 2\pi f$; f este frecvenţa

Mecanismul de polarizare de orientare este considerat dominant deoarece timpul de relaxare, adică intervalul de timp în care polarizaţia creşte şi apoi scade, este de un ordin de mărime comparabil cu perioada de oscilaţie a câmpului de înaltă frecvenţă.

Interpretarea ecuaţiilor lui Debye permite să se afirme că:

- pe măsură ce frecvenţa creşte, dipolii devin incapabili să-şi regăsească pe deplin poziţiile iniţiale pe durata schimbărilor de sens ale câmpului aplicat şi ca o consecinţă polarizaţia dipolară rămâne în urma intensităţii câmpului electric (relaxarea dielectrică);
- pe măsura creşterii frecvenţei se atinge un punct în care polarizaţia de orientare nu mai poate urmări câmpul aplicat şi contribuie foarte puţin la polarizaţia totală;
- scăderea polarizaţiei efective se manifestă prin scăderea constantei dielectrice şi creşterea factorului de pierderi. Energia este acum preluată de la câmpul electromagnetic şi disipată sub formă de căldură în interiorul materialului.

Utilizând teorema lui Stokes, Debye a dedus expresia pentru timpul de relaxare considerând dipolul o sferă de rază r ce se roteşte într-un mediu de vâscozitate η_v:

$$\tau = \frac{4\pi r^3 \eta_v}{kT} \tag{2.43}$$

în care k este constanta lui Boltzmann, iar T temperatura absolută.

Dacă temperatura creşte şi vâscozitatea mediului este mai redusă, moleculelor de dimensiuni mici le corespund frecvenţe de relaxare ridicate. Timpul necesar dipolilor unui lichid, de exemplu, de a se polariza şi depolariza este determinat de constanta timpului de relaxare τ.

Polarizarea de neomogenitate (Maxwell-Wagner)

Acest tip de polarizare se manifestă mai ales în cazul dielectricilor neomogeni care au în structura lor elemente conductive într-o anumită proporţie dispersate în medii neconductive.

Polarizarea Maxwell-Wagner este determinată de apariţia unor sarcini pe suprafeţele de separare ale părţilor omogene din dielectricii neomogeni [3], [4].

Wagner a arătat că pentru cel mai simplu model care reproduce acest tip de polarizare, modelul constând din două sfere conductoare distribuite într-un mediu neconductor, factorul de pierderi al volumului de material este dat de relaţia:

$$\varepsilon_{mw}'' = \frac{9\nu\,\varepsilon' f_{\max}}{1{,}8 \cdot 10^{10}\sigma} \cdot \frac{\omega\tau}{\left(1 + \omega^2\tau^2\right)} \tag{2.44}$$

în care:

- f_{max} – frecvenţa pierderilor maxime (Hz);
- σ este conductivitatea fazei conductoare (S/m);
- τ este timpul de relaxare (s).

Ca şi o concluzie se poate afirma că procesarea la scară industrială, în banda de frecvenţe 10 MHz ÷ 3 GHz a materialelor dielectrice este practic determinată de cele trei mecanisme: conducţia electrică, polarizarea de orientare şi polarizarea de neomogenitate (Maxwell-Wagner), dacă dielectricul este neomogen [4].

2.2.2. Constanta dielectrică complexă

Constanta dielectrică complexă $\underline{\varepsilon}^*$ este o mărime esenţială care caracterizează comportarea unui dielectric sub influenţa unui câmp de înaltă frecvenţă:

$$\underline{\varepsilon}^* = \varepsilon' - j\varepsilon'' \tag{2.45}$$

unde:

- ε' este partea reală a lui $\underline{\varepsilon}^*$ şi caracterizează dielectricul din punct de vedere al capacităţii sale de a se polariza, înglobând pierderi datorate fenomenului de polarizare dipolară asociat cu frecarea dintre molecule;
- ε'' este partea imaginară a lui $\underline{\varepsilon}^*$ şi reprezintă factorul de pierderi, înglobează toate efectele disipative datorate pierderilor prin efect Joule şi dielectrice.

Caracterul complex al constantei dielectrice se demonstrează pornind de la legea circuitului magnetic:

$$\oint \frac{\vec{B}}{\mu_a} \cdot d\vec{l} = \int_S \vec{J}_c \cdot d\vec{S} + \int_S \frac{\partial}{\partial t}\left(\varepsilon_0 \varepsilon' \vec{E}\right) \cdot d\vec{S} \tag{2.46}$$

unde:

- J_c - este densitatea curentului de conducţie generat de deplasarea sarcinilor libere din dielectric sub acţiunea câmpului electromagnetic [A/m^2];
- B - inducţia câmpului magnetic [T];
- μ_a - permeabilitatea magnetică a mediului [H/m];
- ε_0 este permitivitatea vidului [F/m];
- S este suprafaţa parcursă de fluxul electric total [m^2].

Pentru câmpuri electrice alternative de forma $E_{max}e^{j\omega t}$, cum sunt cele aplicate în încălzirea dielectrică, ecuaţia (2.46) devine:

$$rot\underline{H} = \underline{J}_c + j\omega\varepsilon_0\varepsilon'\underline{E} \tag{2.47}$$

Luând în considerare numai efectul conducţiei electrice a sarcinilor libere sub acţiunea câmpului electric şi presupunând constanta dielectrică reală, contribuind numai la înmagazinarea de energie în sistem, relaţia (2.47), se poate scrie şi sub forma:

$$\underline{J} = \left(\sigma + j\omega\varepsilon_0\varepsilon'\right)\underline{E} \tag{2.48}$$

Dând factor comun pe $j\omega\varepsilon_0$, obţinem:

$$\underline{J} = j\omega\varepsilon_0\left(\varepsilon' - \frac{j\sigma}{\omega\varepsilon_0}\right)\underline{E} \tag{2.49}$$

Deoarece în spaţiul liber σ = 0, obţinem:

$$\underline{J} = j\omega\varepsilon_0\varepsilon'\underline{E} \tag{2.50}$$

Comparând relaţia (2.49) cu (2.50), rezultă că relaţia (2.49) poate fi scrisă şi astfel:

$$\underline{J} = j\omega\varepsilon_0\underline{\varepsilon}_c^*\underline{E} \tag{2.51}$$

unde:

$$\underline{\varepsilon}_c^* = \varepsilon_c' - \frac{j\sigma}{\omega\varepsilon_0} = \varepsilon_c' - j\varepsilon_c'' \tag{2.52}$$

$\underline{\varepsilon}_c^*$ poate fi considerată ca şi o constantă dielectrică efectivă a materialului atunci când efectul conductiv este dominant. De altfel, efectul disipativ indus de fenomenul de conducţie electrică este reprezentat în relaţia (2.17) de ε_c'', care reprezintă partea imaginară a constantei dielectrice $\underline{\varepsilon}_c^*$.

Contribuţia fiecărui tip de polarizare cum ar fi de exemplu, polarizarea de orientare (dipolară) sau de neomogenitate (Maxwell-Wagner), va fi luata în consideraţie în constanta dielectrică prin termenul imaginar. Astfel, dacă se consideră de exemplu, că polarizarea de orientare este mecanismul dominant, contribuţia sa la încălzirea materialului dielectric poate fi cuantificată printr-o constantă dielectrică complexă de o formă similară celei din relaţia (2.52):

$$\underline{\varepsilon}_d^* = \varepsilon_d' - j\varepsilon_d'' \tag{2.53}$$

în care indicele **d** se referă la polarizarea dipolară.

Factorul de pierderi (termenul imaginar), înglobează efectele disipative ale tuturor mecanismelor ce contribuie la încălzirea dielectrică (ε"). Pentru a satisface legea circuitului magnetic într-un dielectric real, constanta dielectrică trebuie să aibă o formă complexă, ce include toate mecanismele de polarizare, astfel ecuaţia (2.49) devine:

$$\underline{J}_t = \underline{J}_c + \underline{J}_d = j\omega\varepsilon_0\left[\varepsilon' - j\left(\varepsilon'' + \frac{\sigma}{\omega\varepsilon_0}\right)\right]\cdot\underline{E} \tag{2.54}$$

a cărei reprezentare fazorială este prezentată în figura 2.6.

În figură se arată că densitatea curentului total J_t indus într-un dielectric este suma densităţilor curenţilor de conducţie J_c şi de deplasare J_d ca efect combinat al conducţiei electrice şi al polarizaţiei materialului dielectric sub acţiunea câmpului electric de înaltă frecventă.

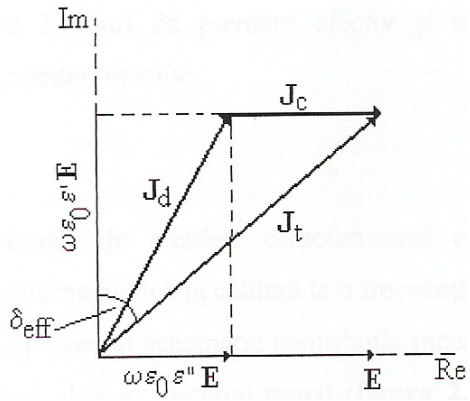


Fig. 2.6. Diagrama fazorială a densităţii curentului total indus într-un dielectric [4]

Deoarece separarea efectelor datorate conducţiei şi a celor datorate polarizaţiei este dificilă, s-a definit un factor de pierderi efectiv ε''_{eff} care însumează efectele conducţiei electrice şi a tuturor mecanismelor de polarizare:

$$\varepsilon''_{eff}(\omega)=\varepsilon''_{mw}(\omega)+\varepsilon''_{d}(\omega)+\varepsilon''_{e}(\omega)+\varepsilon''_{a}(\omega)+\frac{\sigma}{\varepsilon_0\omega} \tag{2.55}$$

unde:

- ε''_{mw} - factorul de pierderi datorate polarizării Maxwell-Wagner;
- ε''_{d} - factorul de pierderi datorate polarizării de orientare (dipolare);
- ε''_{e} - factorul de pierderi datorate polarizării electronice;
- ε''_{a} - factorul de pierderi datorate polarizării atomice;
- $\sigma/\varepsilon_0\omega$ - pierderi datorate conducţiei.

Factorul de pierderi efectiv se poate scrie şi sub forma:

$$\varepsilon''_{eff}(\omega)=\varepsilon''(\omega)+\frac{\sigma}{\varepsilon_0\omega} \tag{2.56}$$

unde: $\varepsilon''(\omega)$ - factorul de pierderi datorate mecanismelor de polarizare.

Constanta dielectrică complexă relativă se poate scrie ţinând cont de relaţia (2.56) astfel:

$$\underline{\varepsilon}^{*}=\varepsilon'_{eff}-j\varepsilon''_{eff} \tag{2.57}$$

Atunci relaţia (2.54) devine:

$$\underline{J}_t=\underline{J}_c+\underline{J}_d=j\omega\varepsilon_0\left(\varepsilon'_{eff}-j\varepsilon''_{eff}\right)\cdot\underline{E}=j\omega\varepsilon_0\underline{\varepsilon}^{*}\cdot\underline{E} \tag{2.58}$$

Raportul dintre factorul de pierdere efectiv şi constanta dielectrică defineşte tangenta unghiului de pierderi efectiv:

$$tg\delta_{eff}=\frac{\varepsilon''_{eff}}{\varepsilon'_{eff}} \tag{2.59}$$

Tangenta unghiului de pierderi caracterizează capacitatea materialului de a transforma energia electromagnetică în căldură la o frecvenţă şi temperatură dată.

2.2.3. Ecuaţia încălzirii dielectricului

Presupunând cazul simplu al unui condensator alimentat la o tensiune alternativă cu pulsaţia ω şi valoarea efectivă U, se poate scrie:

$$\underline{I} = \underline{U} \cdot j\omega C \tag{2.60}$$

În cazul unui condensator plan cu dielectric cu pierderi, avem:

$$\underline{C} = \frac{A\varepsilon_0 \underline{\varepsilon}^*}{d} = \frac{A\varepsilon_0}{d}\left(\varepsilon' - j\varepsilon''\right) \tag{2.61}$$

unde A este aria plăcilor şi d distanţa între ele. Înlocuind în (2.60) rezultă:

$$\underline{I} = \underline{U} \cdot \frac{A\varepsilon_0}{d}\omega\left(j\varepsilon' + \varepsilon''\right) = j\omega\underline{U}\frac{A\varepsilon_0\varepsilon'}{d} + \omega\underline{U}\frac{A\varepsilon_0\varepsilon''}{d} \tag{2.62}$$

Al doilea termen din ecuaţia (2.62) reprezintă disipaţia de căldură, deoarece este în fază cu tensiunea aplicată. Puterea disipată este:

$$P = \mathrm{Re}\left(\underline{U} \cdot \underline{I}\right) = \omega U^2 \frac{A\varepsilon_0\varepsilon''}{d} = 2\pi \cdot Ad \cdot f\left(\frac{U}{d}\right)^2 \varepsilon_0\varepsilon'' \tag{2.63}$$

Densitatea de putere se obţine împărţind puterea din formula (2.63) la volumul considerat ($p = P/Ad$), şi ţinând cont ca $E = U/d$ se obţine ecuaţia încălzirii dielectricului:

$$p = 2\pi f\varepsilon_0\varepsilon''E^2 \tag{2.64}$$

Aşadar densitatea de putere disipată în dielectric este proporţională cu frecvenţa, factorul de pierderi ε" şi pătratul intensităţii câmpului electric. Această formulă este generalizată şi se aplică în orice situaţie, nu numai în cazul banal al condensatorului plan, care a fost folosit doar pentru demonstraţie.

O altă mărime folosită deseori pentru a caracteriza un dielectric cu pierderi este unghiul de pierderi, dat de obicei ca tangentă. Acesta este diferenţa între defazajul ideal de 90° şi defazajul real între curent şi tensiune în cazul unui condensator care foloseşte dielectricul considerat:

$$tg\delta = \frac{\varepsilon''}{\varepsilon'} \tag{2.65}$$

2.2.4. Dependenţa de temperatură şi umiditate a proprietăţilor dielectrice

J. B. Hasted a reprezentat schematic contribuţia mecanismelor luate în consideraţie prin factorul de pierderi al unui material umed (figura 2.7) în care pierderile dipolare depind de starea apei conţinute în material.

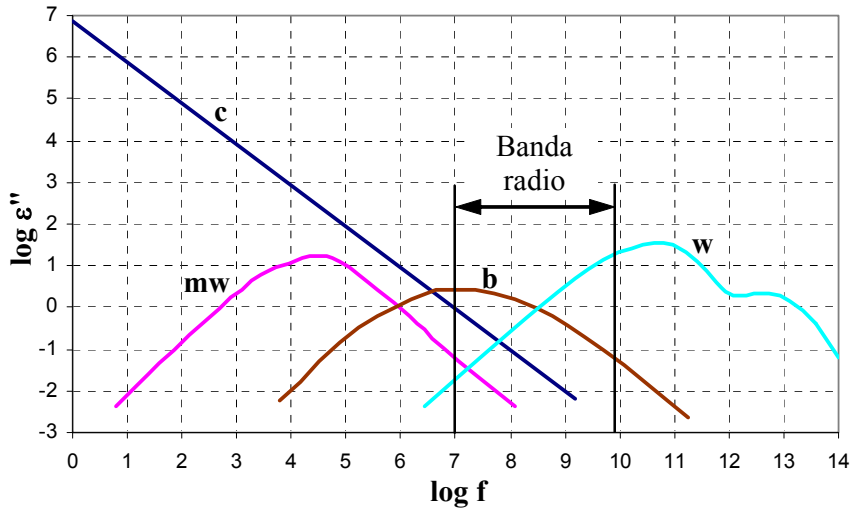


Fig. 2.7. Factorul de pierderi efectiv al unui material dielectric neomogen în funcţie de frecvenţă, (b) - relaxarea apei legate, (w) - relaxarea apei libere, (mw) - pierderi Maxwell-Wagner şi (c) - pierderi prin conducţie

Datorită faptului că frecvenţele industriale de încălzire ale microundelor se situează în banda $10^7 < f < 3*10^9$ Hz, mecanismele luate în consideraţie sunt cele dipolare şi de neomogenitate. Polarizările de tip atomic şi electronic apar la frecvenţele din domeniul infraroşu şi din domeniul vizibil, ca parte al spectrului electromagnetic, acestea nu joacă un rol important în încălzirea cu microunde.

Deoarece multe aplicaţii implică îndepărtarea umezelii din încărcătura de lucru, variaţia lui $\underline{\varepsilon}^*$ şi mai ales ε''_{eff} cu conţinutul de umiditate joacă un rol important în proiectarea instalaţiilor de încălzire microunde.

Pentru aceasta a fost făcut un efort major pentru a stabili variaţia lui ε'_{eff} şi ε''_{eff} cu umiditatea, M, pentru mai multe materiale industriale cum ar fi: hârtie, scândură, alimente, lemne, textile pentru a interpreta rezultatele şi pentru a fi direct aplicabile în proiectarea instalaţiilor cu microunde.

Există două deducţii importante rezultate din factorul de pierderi funcţie de conţinutul de umiditate. În primul rând din caracteristicile (figura 2.8), ε''_{eff} funcţie de umiditatea M, obţinute pentru diferite orientări ale câmpului de frecvenţă înaltă pe direcţia fibră/fir, este arătat cantitativ că pierderile sunt mai mari pentru orientările de câmp paralele cu firul, ceea ce este caracteristic pentru materialele de celuloză, hârtie, scândură, (Driscott, 1976) şi lemn (James şi Hamill, 1965) [11].

În al doilea rând, panta curbei ε''_{eff} funcţie de conţinutul de umezeală, M, este critică pentru aplicaţii industriale unde uniformitatea umezelii unui material în formă de foi este prima cerinţă, efectul fiind arătat calitativ în figura 2.8b. Acest lucru se petrece deoarece uniformitatea umezelii este efectivă pentru conţinuturi de umezeală deasupra valorii critice, deoarece valori mari ale $d\varepsilon''/dM$ asigură faptul

că părţile mai umede ale materialului absorb mai multă putere şi tinde să egalizeze distribuţia de umezeală. În figura 2.8b, efectul de egalizare este mult mai puternic pentru caracteristica III decât pentru caracteristica II, iar un material cu caracteristica I nu manifestă această proprietate. Un material dat poate manifesta proprietăţi asemănătoare cu cele ale curbelor II şi III la frecvenţe diferite *(Metaxas şi Driscoll, 1974)*.

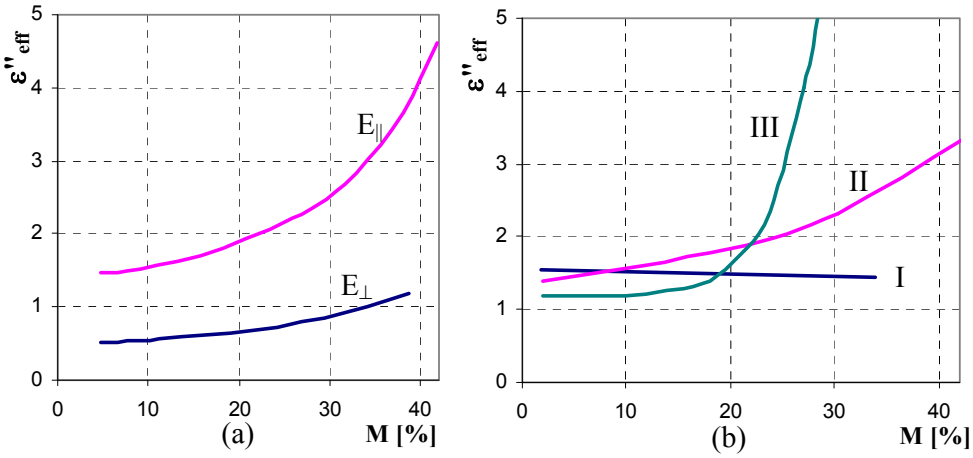


Fig. 2.8. Factorul efectiv de pierderi în funcţie de umiditate pentru orientări diferite ale câmpului electric

O serie de investigaţii au fost făcute pentru a explica dependenţa de temperatură a factorului de pierderi în diverse materiale. Măsurările lui Tinga (1969) [4] pentru bradul Douglas la umezeală scăzută au arătat că factorul de pierderi creşte cu temperatura deoarece legătura fizică se reduce şi dipolurile sunt mai libere pentru reorientare.

Pentru umezeală peste 25%, factorul de pierderi scade cu creşterea temperaturii. Influenţa temperaturii şi frecvenţei asupra constantei dielectrice complexe a multor alimente a fost investigată de către *Bengtsson şi Risman (1971)* la temperaturi între –20°C şi 60°C şi de către *To (1974)* care s-a concentrat pe temperaturi deasupra dezgheţului, până la 65°C [4].

Cercetările lui *Bengtsson şi Risman (1971)* sunt foarte importante pentru că arată clar marile diferenţe ale lui ε''_{eff} şi ε'_{eff} între gheaţă şi apa lichidă. Deoarece majoritatea alimentelor conţin o cantitate apreciabilă de apă, proprietăţile lor dielectrice au o caracteristică asemănătoare cu a apei în funcţie de temperatură. Atât ε''_{eff} cât şi ε'_{eff} manifestă creşteri mari cu temperatura la dezgheţ, după care valorile scad cu creşterea temperaturii exceptând situaţia unei concentraţii mari de sare când creşterea este continuă datorită pierderilor prin conductivitate.

Relaxarea dielectrică în apă lichidă a fost studiată pe larg de *Hasted (1973)*. Proprietăţile dielectrice ale apei în volum nu sunt de mare importanţă în încălzirea cu microunde; pentru majoritatea aplicaţiilor relaxarea apei legate este mult mai importantă. Caracteristicile dielectrice ale apei şi soluţiilor de sare sunt prezentate în figura 2.9a. Vârful lui ε''_{eff} depinde de temperatură şi este situat între aproximativ 9

÷ 30 GHz în cazul temperaturilor între 0 şi 50 °C. Influenţa sării este arătată clar de factorul de pierderi care atinge valoarea de 100 la 9 GHz pentru o soluţie de 0,3 molar. Dependenţa de temperatură a lui ε''_{eff} şi ε'_{eff} în apă pură este prezentată în figura 2.9b la cele două frecvenţe de încălzire cu microunde.

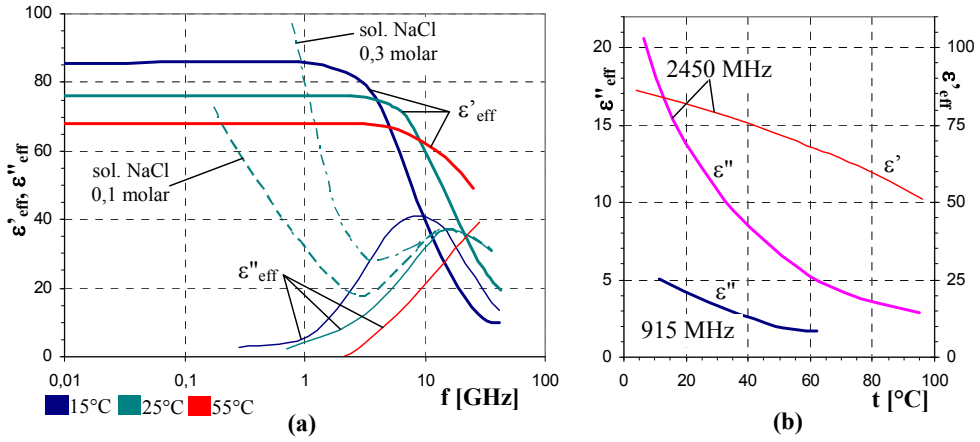


Fig. 2.9. Efectele frecvenţei şi temperaturii asupra proprietăţilor dielectrice ale apei pure şi soluţiilor saline

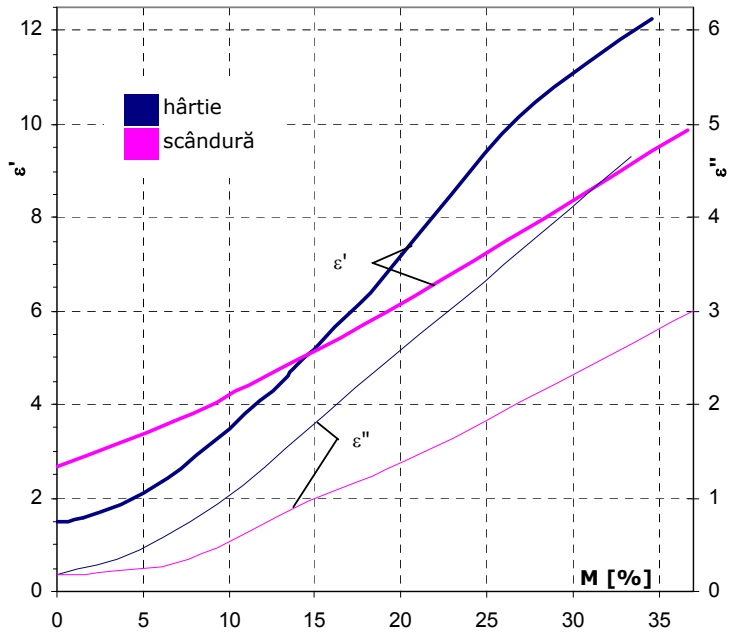


Fig. 2.10. Proprietăţile dielectrice ale hârtiei şi scândurii în funcţie de conţinutul de apă la 2450 MHz

Metaxas şi Driscoll (1974) au efectuat măsurări de proprietăţi dielectrice pentru hârtie şi scândură la 2,45 GHz (figura 2.10), câmpul electric fiind paralel cu fibrele.

Mai multe articole au fost publicate despre proprietăţile dielectrice ale bradului Douglas [4], [6], [9]. În figurile 2.11 ÷ 2.13 sunt prezentate rezultatele lui *Tinga (1969)*, care a studiat variaţia timpului de relaxare cu conţinutul de umezeală şi temperatură, pentru câmpul electric paralel cu fibrele.

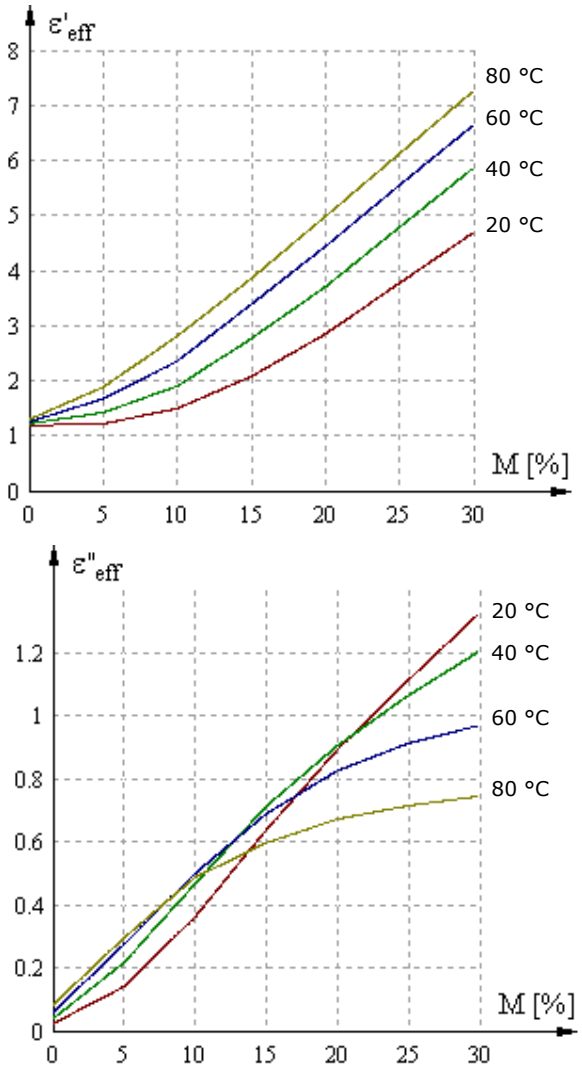


Fig. 2.11. Valorile ε'_{eff} şi ε''_{eff} în funcţie de umiditate la lemnul de brad Douglas

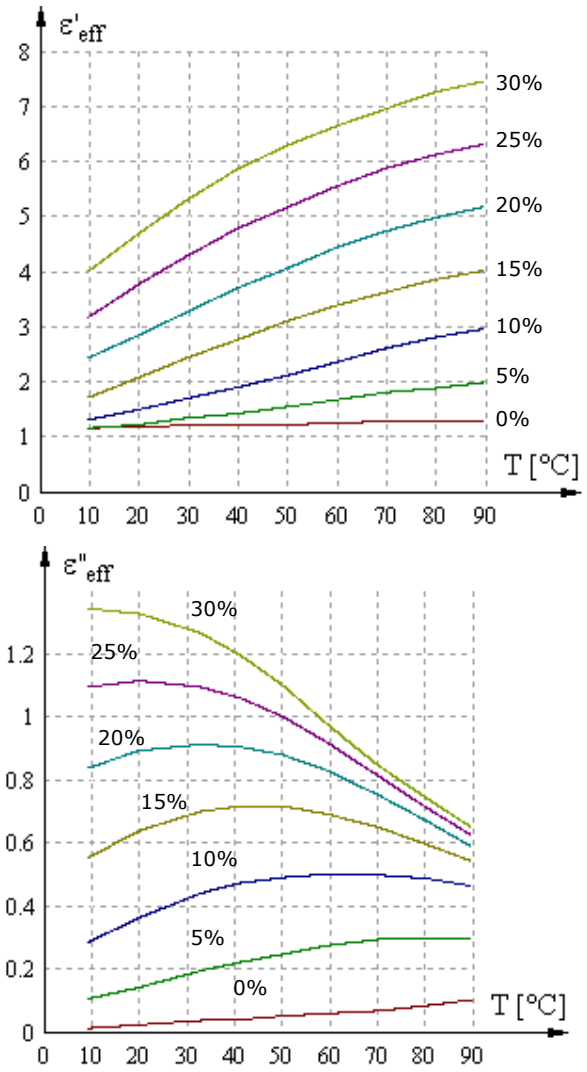


Fig. 2.12. Valorile ε'_{eff} şi ε''_{eff} în funcţie de temperatură la lemnul de brad Douglas

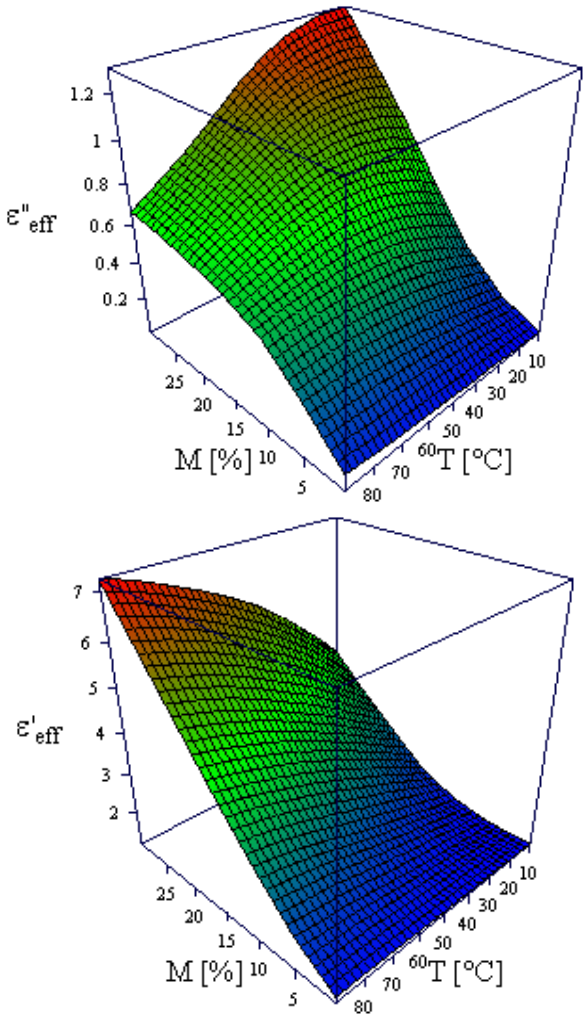


Fig. 2.13. Valorile ε'_{eff} şi ε''_{eff} în funcţie de umiditate şi temperatură la lemnul de brad Douglas (grafice în perspectivă tridimensională)

Pierderile cele mai mari în acest caz sunt la temperatură mică şi umiditate ridicată, ceea ce conduce la stabilitate termică în cazul uscării lemnului.

2.2.5. Adâncimea de penetrare

Pe măsură ce o undă electromagnetică se propagă într-un material dielectric cu pierderi, amplitudinea ei scade din cauza transformării energiei ei în căldură. Dacă facem abstracţie de undele reflectate (semispaţiu infinit), atunci densitatea fluxului de putere scade exponenţial pe măsură ce ne îndepărtăm de suprafaţă. Deoarece densitatea de putere absorbită de material este proporţională cu densitatea fluxului de putere care trece prin el, şi densitatea de putere disipată

scade exponenţial cu distanţa până la suprafaţă. Rata scăderii puterii disipate este funcţie atât de permitivitatea relativă ε' cât şi de factorul de pierderi ε". Adâncimea de penetrare este definită ca adâncimea în material la care fluxul de putere este $e^{-1} \approx 0{,}368$ din valoarea de la suprafaţă. Relaţia de calcul este (*Metaxas şi Meredith, 1983*) [4]:

$$D_p = \frac{\lambda_0}{2\pi\sqrt{2\varepsilon'}} \frac{1}{\sqrt{\sqrt{1+\left(\frac{\varepsilon''}{\varepsilon'}\right)^2} - 1}} \tag{2.66}$$

Atunci când ε" ≤ ε', relaţia (2.66) se simplifică la (2.67) cu o eroare de până la 10%:

$$D_p = \frac{\lambda_0 \sqrt{\varepsilon'}}{2\pi\varepsilon''} \tag{2.67}$$

Adâncimea de penetrare este un parametru de material foarte important deoarece oferă o primă aproximaţie a distribuţiei de putere în volumul materialului. Într-un semispaţiu infinit de material ideal (ε' şi ε" constante) şi cu o undă plană incidentă normal la suprafaţă, densitatea de putere la adâncimea *z* este dată de:

$$p_z = p_0 \exp\left(-\frac{z}{D_p}\right) \tag{2.68}$$

O aproximaţie a distribuţiei de putere într-un material gros încălzit din ambele părţi poate fi făcută prin superpoziţia a două astfel de distribuţii exponenţiale având originile pe cele două feţe.

Integrând relaţia (2.68) pe intervalul $0 \div D_p$ se obţine o altă definiţie a adâncimii de pătrundere: este adâncimea până la care se disipă $(1 - e^{-1}) \approx 63{,}2\%$ din puterea totală, restul fiind disipat la adâncimi mai mari.

Deşi definiţiile sunt asemănătoare, adâncimea de penetrare nu este acelaşi lucru cu adâncimea de pătrundere într-un perete conductor. În cazul peretelui conductor aproape toată energia microundelor este reflectată înapoi, numai o mică parte fiind absorbită. Câmpul magnetic direct al undei determină circulaţia unor curenţi la suprafaţa conductorului, care la rândul lor produc câmpuri magnetice aproximativ egale şi opuse, ceea ce conduce la apariţia unei unde reflectate. Este uzual în calculele instalaţiilor de microunde să se considere pereţii din conductor ideal, deci adâncimea de pătrundere şi implicit pierderile în conductor zero.

Este interesant de observat că dacă factorul de pierderi ε" este constant, adâncimea de penetrare creşte cu permitivitatea relativă ε'. Motivul este acela că impedanţa caracteristică ζ scade (rel. 2.26) şi odată cu ea şi amplitudinea intensităţii câmpului electric, deci şi densitatea de putere disipată.

2.3. Limitări ale densităţii de putere disipată

Pentru a obţine maximum de productivitate şi randament este necesar să se crească la maxim densitatea de putere în scopul reducerii timpului de procesare. Totuşi, ca şi în cazul încălzirii superficiale, puterea aplicată nu poate depăşi o anumită valoare, deşi din motive diferite. Dacă în cazul încălzirii clasice puterea era limitată de volumul materialului şi difuzivitatea termică, în cazul încălzirii

volumetrice cu microunde pot apărea fenomene termice şi electrice nedorite în material dacă puterea este prea mare.

2.3.1. Limitări termice

Limitările termice apar de obicei, dar nu întotdeauna, ca urmare a încălzirii neuniforme a materialelor neomogene. Există mai multe fenomene care pot apărea:

Gradienţii de temperatură pot produce tensiuni mecanice în material şi posibil deteriorări ale acestuia.

Gradienţi de umiditate în cazul uscării, care rezultă de asemenea în tensiuni mecanice şi deteriorări. Aceasta este o problemă importantă în cazul uscării rapide a materialelor ceramice, şi apare chiar dacă uniformitatea încălzirii este perfectă. Căldura din material se propagă nu numai prin convecţie ci şi prin transportul apei fierbinţi care difuzează către suprafaţă. Evaporarea rapidă la suprafaţă poate produce gradienţi mari de temperatură, care pentru a fi controlaţi se practică menţinerea unui mediu controlat cu umiditate mare.

Presiunea generată prin evaporare din cauza fierberii locale în interiorul materialului poate cauza tensiuni mecanice şi rupturi, uneori prin explozie; spargerea ouălor încălzite într-un cuptor de bucătărie este un exemplu cunoscut.

2.3.2. Străpungerea dielectrică şi producerea de arc electric

Aceste fenomene au loc atunci când intensitatea câmpului electric depăşeşte o anumită valoare determinată de proprietăţile materialului, ale mediului şi de existenţa neomogenităţilor conductoare. Deşi în mod normal intensitatea câmpului electric este mult sub valoarea critică, uneori condiţii speciale favorizează străpungerea şi arcul electric. Dacă una sau mai multe din condiţiile următoare se îndeplinesc, atunci străpungerea şi arcul electric trebuie luate în calcul:

- procesul tehnologic cere densităţi mari de putere disipată
- factorul de pierderi în material este mic
- procesarea are loc în vid sau la presiune scăzută
- există incluziuni conductoare în material

Intensitatea câmpului electric necesară menţinerii unui arc electric este mult mai mică decât cea necesară amorsării lui, şi este invariabil necesar să se oprească sursa de microunde pentru a stinge arcul. Este necesară o pauză de câteva secunde pentru a permite gazelor ionizate să se disperseze înainte de a reaplica microundele cu putere mai mică, existând un risc mare ca datorită deteriorării materialului acolo unde a avut loc străpungerea, aceasta să se întâmple din nou în acelaşi loc.

Aerul uscat la presiune atmosferică şi temperatura camerei (18 °C ≈ 288 K) are rigiditatea dielectrică de aproximativ 30 kV/cm, aproape constantă în frecvenţă de la câmp staţionar la microunde, dar într-un câmp uniform. În contextul încălzirii cu microunde, intensitatea câmpului electric care contează este dată de amplitudinea undei şi trebuie să se ţină cont de efectele de concentrare a câmpului şi de orice altceva care ar putea să scadă rigiditatea dielectrică a materialului.

Legea lui Paschen (*Joos, 1958*) [4] afirmă că rigiditatea dielectrică a unui gaz este proporţională cu presiunea absolută sau densitatea, adică cu numărul de molecule de gaz prezent în unitatea de volum. Cum la presiune constantă densitatea scade cu creşterea temperaturii, rezultă că şi rigiditatea dielectrică scade. Astfel, la 200 °C şi presiune atmosferică rigiditatea dielectrică a aerului este de aproximativ

$30 \times 288/473 = 18\ kV/cm$. Este deci important să se ţină cont de temperatura suprafeţei materialului şi a gazului din incinta cuptorului atunci când se estimează limitarea dată de străpungerea dielectrică.

La presiune mică, rigiditatea dielectrică E_b scade liniar după legea lui Paschen până la presiuni de aprox. 15 mbar, când E_b = 450 V/cm. Un minim este atins în jur de 1 mbar după care rigiditatea dielectrică creşte din nou pe măsură ce presiunea scade. Valoarea minimă a rigidităţii dielectrice este dependentă de frecvenţă, şi este de 190 V/cm la 2450 MHz şi de 80 V/cm la 915 MHz. În majoritatea proceselor de uscare în vid presiunea este în jur de 50 ÷ 100 mbar, ceea ce corespunde la o rigiditate dielectrică de 1500 ÷ 3000 V/cm la toate frecvenţele. Figurile 2.14 şi 2.15 arată rigiditatea dielectrică pentru aer şi respectiv vapori de apă.

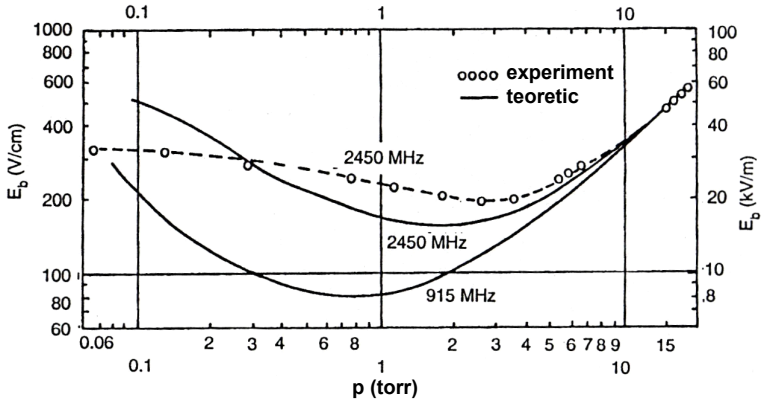


Fig. 2.14. Rigiditatea dielectrică a aerului la presiune mică

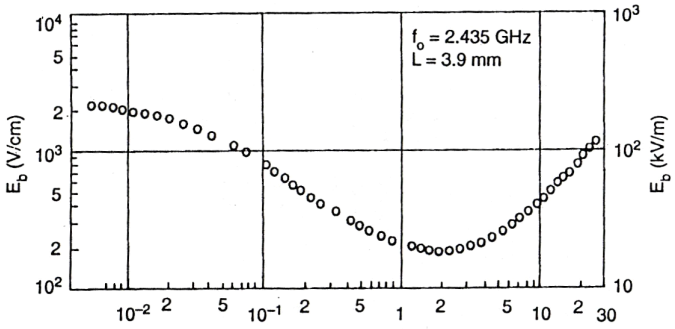


Fig. 2.15. Rigiditatea dielectrică a vaporilor de apă la presiune mică

Din cauza naturii statistice a străpungerii dielectrice şi a faptului că există mulţi factori care pot cauza o ridicare a intensităţii câmpului electric peste rigiditatea dielectrică, în practică se lucrează de obicei cu o marjă de eroare considerabilă.

Arcul electric poate să apară atunci când sarcina este sub formă de granule mari care se ating sau sunt foarte apropiate. Dacă de exemplu granulele sunt de formă sferică iar două dintre ele sunt foarte apropiate şi au centrele pe o axă

paralelă cu intensitatea câmpului electric, curentul de deplasare "prins" de granule va trece de la una la alta prin zona de contact care are o suprafaţă mică. Deci densitatea curentului de deplasare va fi mare, creând o zonă de volum foarte mic în care intensitatea câmpului este mare şi în plus încălzirea este mai puternică. Poate avea loc o străpungere imediată, sau este posibil ca aceasta să aibă loc datorită încălzirii locale. Acest fenomen de concentrare este cu atât mai important cu cât permitivitatea relativă a materialului este mai mare.

2.3.3. Ambalarea termică

Ambalarea termică este o situaţie care apare când distribuţia de putere disipată într-un volum mic din sarcină depăşeşte fluxul termic transmis de acest volum împrejurimilor, astfel încât viteza de creştere a entalpiei este mai mare decât în jur. Temperatura creşte mai repede în acest „punct fierbinte" până are loc descompunerea. Ambalarea termică degenerează invariabil în producerea de arcuri şi e deseori dificil de a realiza care fenomen este cauza şi care efectul.

În practică, mecanismele implicate în ambalarea termică sunt foarte complicate şi aproape imposibil de cuantificat. Este un fenomen statistic; condiţiile de lucru sunt stabilite astfel încât probabilitatea unui incident de ambalare termică să fie în limite acceptabile, care pot fi determinate doar experimental.

Ambalarea termică este determinată de o inegalitate locală în ecuaţia transferului de căldură; mărimile implicate sunt densitatea de putere disipată ca urmare a efectului microundelor şi transferul de căldură de la „punctul fierbinte" zonelor adiacente.

O anomalie locală în distribuţia densităţii de putere poate să apară ca urmare a câmpului electric neuniform sau a unei valori locale mari a factorului de pierderi. Câmpul electric poate fi concentrat în apropierea unei suprafeţe de separare, datorită undelor staţionare sau proximităţii unei alimentări a aplicatorului. Acestea două din urmă se extind însă pe volume mult mai mari decât volumul iniţial a unui punct fierbinte şi de obicei nu sunt implicate. O neomogenitate în permitivitatea relativă ε' poate determina de asemenea concentrarea E. Particulele metalice, în special filiforme, creează concentrări puternice ale câmpului electric.

Factorul de pierderi este deseori o funcţie crescătoare de temperatură. Materialele plastice şi cauciucul prezintă astfel de caracteristici, cu o creştere rapidă atunci când începe descompunerea termică. Produşii de descompunere, îndeosebi carbonul, pot avea o valoare mult mai mare a lui ε''. Dacă tratarea materialului implică transformări de fază (de exemplu dezgheţarea apei) sau reacţii chimice, acestea pot conduce la creşteri bruşte ale ε'' odată cu temperatura. Efectul este că punctul fierbinte se va încălzi încă şi mai repede, de aici şi numele „ambalare termică".

Punctul fierbinte este răcit prin difuzia căldurii materialului înconjurător, determinată complet de gradientul de temperatură şi difuzivitatea termică. Conductivitatea termică (şi implicit difuzivitatea) pot să scadă prin formarea de goluri în apropierea punctului fierbinte. Materialele cu conductivitate termică scăzută sunt mai predispuse la ambalare termică.

3. Câmpul electromagnetic în structurile de microunde

Este cunoscut faptul că o undă electromagnetică plană există numai într-un mediu uniform şi infinit extins. În tehnica microundelor este necesară ghidarea şi controlarea radiaţiei electromagnetice.

Sistemele de medii conductoare sau dielectrice care determină propagarea undelor electromagnetice în lungul unor trasee se numesc ghiduri de undă. Exemple de ghiduri de undă sunt: linia bifilară, linia multifilară, cablul coaxial, etc. Există anumite condiţii în care radiaţia electromagnetică se propagă prin astfel de structuri.

Cel mai simplu ghid de undă este construit din două plăci perfect conductoare în care unda electromagnetică plană se propagă paralel cu pereţii conductori. Sistemul din figura 3.1 realizat din plăci perfect conductoare are extensie finită pe direcţiile **y** şi **x** şi cu extensie mult mai mare în direcţia **z** este un ghid de undă. Câmpul electromagnetic în această structură arată exact ca o undă plană care se propagă paralel cu plăcile conductoare.

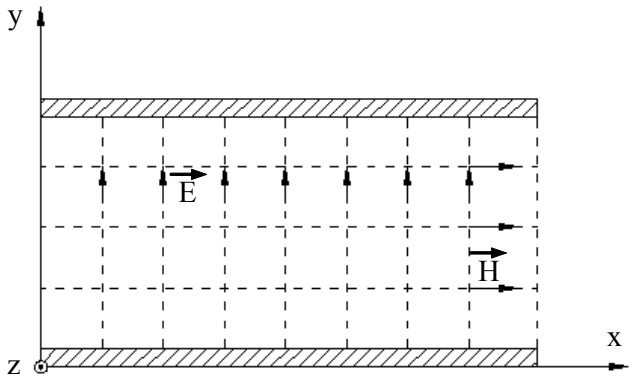


Fig. 3.1. Ghid de undă cu plăci plane paralele

Aspectul câmpului electromagnetic într-un ghid de undă se mai numeşte **mod**. Unda electromagnetică plană este un mod TEM, adică transversal electric şi magnetic, intensităţile câmpului electric şi câmpului magnetic sunt situate în plane normale pe direcţia de propagare. Câmpul electric are componentă numai după *Oy* (E_y) şi induce pe feţele ghidului de undă sarcină electrică repartizată uniform cu densitatea superficială $\rho_s = \varepsilon E_y$, iar câmpul magnetic are componentă numai pe axa *Ox* şi induce pe suprafaţa plăcilor pânze de curent cu densitatea superficială $J_s = H_x$. În acest fel sunt satisfăcute ecuaţiile lui Maxwell şi condiţiile de frontieră.

Modul TEM este realizabil şi în dielectricul dintre doi cilindri coaxiali cu razele r_e şi r_i >> $(r_e - ri)$, dar nu este realizabil în ghidurile de undă tubulare cu contur finit al secţiunii transversale şi nici chiar în ghidul cu plăci paralele, dacă unghiul de incidenţă al undei este diferit de $\pi/2$.

3.1. Propagarea undelor într-un ghid dreptunghiular

Pentru a înţelege mai uşor modul de formare şi propagare a undelor staţionare pe ghid, se va presupune un exemplu simplu. Fie ghidul de undă cu plăcile plane paralele din figura 3.1. Sursa de câmp poate să fie un conductor parcurs de curent paralel cu plăcile conductoare ($+S_0$).

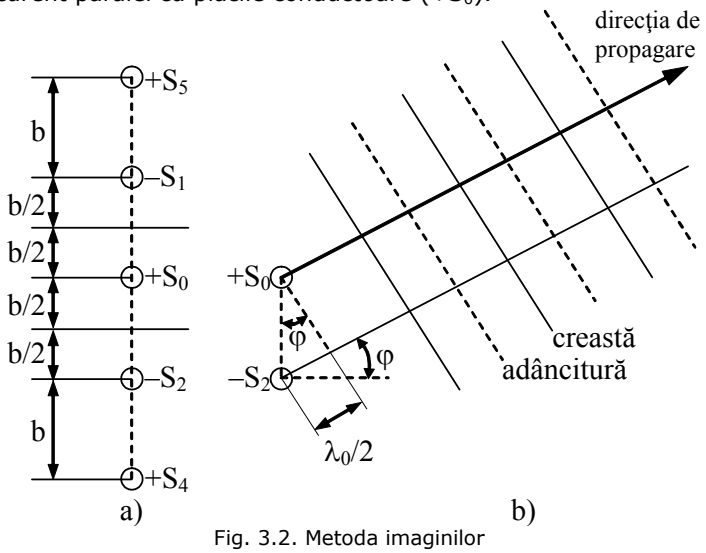


Fig. 3.2. Metoda imaginilor

Pentru soluţionarea problemei de câmp se poate apela la metoda imaginilor. Suprimarea plăcilor conductoare este echivalentă cu introducerea unor surse imagini (figura 3.2a) astfel încât să fie îndeplinite condiţiile de frontieră la suprafaţa plăcilor ($E_t = 0$). Sistemul de surse imagine se extinde la infinit.

Prin compunerea câmpurilor produse de fiecare din aceste surse se obţin undele staţionare, pentru care în figura 3.2b s-au trasat liniile de fază constantă (creasta - maximum amplitudinii, adâncimea - minimum amplitudinii).

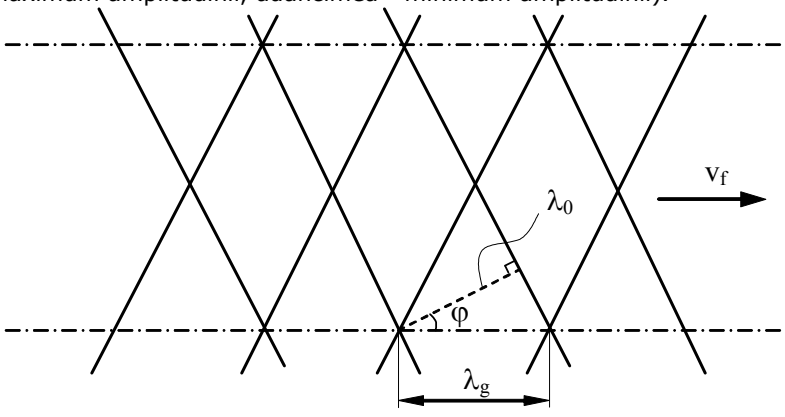


Fig. 3.3. Propagarea undelor în ghid

Direcţia de propagare a undei, va fi aceea pentru care diferenţa de fază în timp, pentru două surse vecine, corespunde la o jumătate de oscilaţie.

Altfel spus $r_2 - r_0 = \frac{\lambda_0}{2}$, iar unghiul φ este dat de relaţia $\sin\varphi = \frac{\lambda_0}{2b}$.

Există evident şi un alt sistem de unde care se propagă în jos (figura 3.3) la unghi simetric faţă de şirul de surse.

Undele obţinute prin suprapunerea unui şir infinit de surse liniare se pot obţine cu două sisteme de unde ce sunt continuu reflectate înainte şi înapoi de două oglinzi perfecte (se ştie că o reflexie reprezintă o inversare de fază). Aceste sisteme de unde reflectate s-ar anihila reciproc până s-ar ajunge la unghiul φ calculat mai sus.

3.1.1. Lungimea de undă de tăiere

Se consideră un ghid de undă cu plăci plane paralele şi o undă plană care se propagă pe o direcţie care face un unghi φ cu planul plăcilor (figura 3.4).

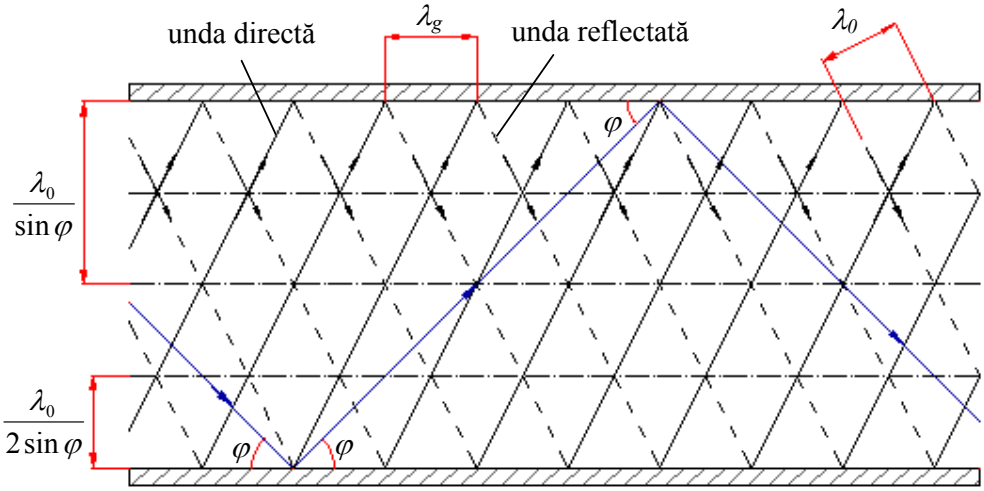


Fig. 3.4. Ghid de undă cu plăci plane paralele

Direcţia de propagare este perpendiculară pe liniile de fază egale, care sunt dispuse la distanţe egale cu lungimea de undă λ_0. Pentru a satisface condiţiile de frontieră pentru câmpul electric (componentele tangenţiale să se anuleze), este necesar ca lăţimea *b* a ghidului de undă să fie un multiplu de jumătăţi de undă:

$$b = \frac{n\lambda_0}{2\sin\varphi} \qquad (3.1)$$

unde *n* este un număr întreg. Altfel spus dacă s-ar dispune o placă conductoare în planul (*P*) nu se schimbă aspectul undei. Relaţia (3.1) reprezintă condiţia de propagare.

Pentru orice *n*, distanţa minimă de realizare a undei este:

$$b = \frac{n\lambda_0}{2} \tag{3.2}$$

care corespunde unui unghi (φ = 90°).

La un *b* dat, cea mai mare lungime de undă posibilă (care „încape" în ghid) este:

$$\lambda_c = \frac{2b}{n} \tag{3.3}$$

numită lungime de undă de tăiere (sau caracteristică) căreia îi corespunde frecvenţa de tăiere:

$$f_c = \frac{c}{\lambda_c} = \frac{nc}{2b} = \frac{n}{2b\sqrt{\mu\varepsilon}} \tag{3.4}$$

3.1.2. Lungimea de undă a ghidului

Lungimea de undă a ghidului (λ_g) este:

$$\lambda_g = \frac{\lambda_0}{\cos\varphi} \tag{3.5}$$

Se observă că, pentru unghiul φ = 90°, $\lambda_g \to \infty$ şi în consecinţă nu mai există propagare. Din acest motiv, relaţia (3.2) se numeşte condiţie de tăiere. Lungimea de undă caracteristică reprezintă în fapt cea mai mare lungime de undă care încape în ghid.

Interesează ce relaţie de legătură există între λ_0, λ_g şi λ_c. Pentru a obţine o astfel de relaţie se ţine cont de relaţiile (3.1) şi (3.5) şi având în vedere că $\sin^2\varphi + \cos^2\varphi = 1$ se obţine:

$$\left(\frac{n\lambda_0}{b}\right)^2 + \left(\frac{\lambda_0}{\lambda_g}\right)^2 = 1 \quad \Rightarrow \quad \lambda_g = \frac{\lambda_0}{\sqrt{1-\left(\frac{n\lambda_0}{b}\right)^2}}$$ şi ţinând cont de relaţia

(3.3) se obţine:

$$\lambda_g = \frac{\lambda_0}{\sqrt{1-\left(\frac{\lambda_0}{\lambda_c}\right)^2}} \tag{3.6}$$

La frecvenţe joase se poate propaga numai modul linie de transmisie (TEM). Pe măsură ce creşte frecvenţa, de exemplu pentru n = 1 (atunci când $\lambda_0 = 2b$) va începe să se propage şi primul mod dintre cele de ghid. La o frecvenţă şi mai mare care corespunde lungimii de unda $\lambda_0 = b$ apare şi al doilea mod de propagare, etc.

3.1.3. Vitezele de propagare a undelor

Viteza de fază

Viteza de fază este definită în studiul undei plane. Deoarece într-un ghid de undă $\lambda_g > \lambda_0$, rezultă că viteza de fază în ghid este mai mare decât viteza de propagare a undei plane.

Viteza de fază a undei în ghid este dată de relaţia:

$$v_f = \frac{\lambda_g}{T} = \frac{\lambda_0}{T\cos\varphi} = \frac{c}{\cos\varphi} \tag{3.7}$$

Viteza de fază în ghid se poate exprima şi în funcţie de ω şi β. Ţinând seama că $\lambda_g = \frac{2\pi}{\beta}$ şi $\lambda_0 = \frac{c}{f}$ rezultă:

$$v_f = \frac{2\pi f}{\beta} = \frac{\omega}{\beta} \tag{3.8}$$

De reţinut că viteza de fază este o viteză relativă, acesta nu este viteza cu care se propagă informaţia ci viteza cu care se deplasează configuraţia de câmp.

Viteza de grup

Pentru determinarea vitezei de grup a undelor în ghid (v_g) se are în vedere că propagarea acesteia se face în zig-zag (figura 3.5). Viteza de grup este viteza cu care energia electromagnetică sau „semnalul" ce transportă informaţia se propagă în lungul ghidului.

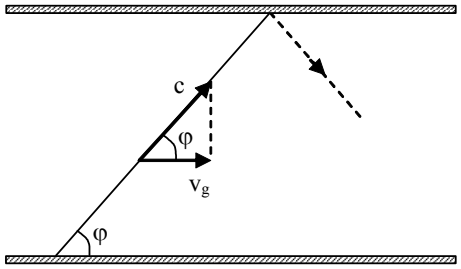


Fig. 3.5. Viteza de grup

Rezultă că:

$$v_g = c\cos\varphi = c\frac{\lambda_0}{\lambda_g} \tag{3.9}$$

Viteza de grup v se poate exprima în funcţie de ω şi β. Având în vedere că $\lambda_g = 2\pi/\beta$ şi $\lambda_0 = 2\pi c/\omega$, din (3.9) rezultă:

$$v_g = c\frac{2\pi c}{\omega}\frac{\beta}{2\pi} = c^2\frac{\beta}{\omega} \tag{3.10}$$

Ţinând cont de expresia vitezei de fază dată de expresia (3.8), se obţine o legătură funcţională între v_g şi v_f:

$$v_g v_f = c^2 \tag{3.11}$$

3.2. Moduri de linie de transmisie

Să presupunem că ghidul de undă realizat din două plăci conductoare paralele (studiat în paragrafele precedente) îl curbăm. Se obţine un ghid ca în figura 3.6, iar dacă îl curbăm complet se obţine un ghid ca în figura 3.7.

Câmpul electromagnetic obţinut în modul reprezentat în figura 3.7 este ca şi cum pe conturul interior potenţialul electric are o valoare V = const, iar pe conturul exterior potenţialul electric este nul. Sau, conturul interior este parcurs de un curent electric de conducţie de „ducere" (perpendicular pe planul secţiunii) iar conturul exterior este parcurs de un curent electric de „întoarcere".

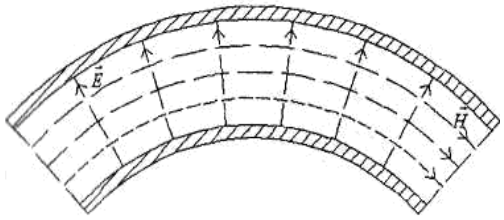


Fig. 3.6. Ghid de undă cu plăci curbate

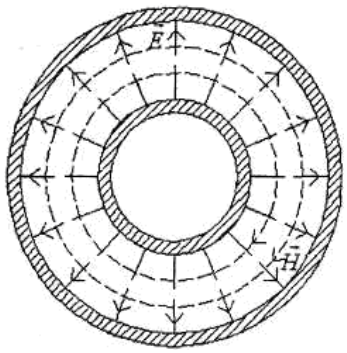


Fig. 3.7. Ghid de undă coaxial

Situaţia câmpului electromagnetic este ca şi în cazul câmpului electromagnetic staţionar (câmp electric staţionar şi câmp magnetic staţionar).

Totuşi exista o deosebire: între intensităţile câmpului electric şi a câmpului magnetic (componente ale undei electromagnetice) este o legătură de forma:

$$\frac{E}{H} = \zeta$$

Pentru unda electromagnetică plană s-a definit o impedanţă intrinsecă. Pentru un ghid de undă se defineşte o impedanţă de undă ca raportul dintre componentele tangenţiale ale câmpului electric şi ale câmpului magnetic:

$$\frac{E_t}{H_t} = \zeta_0 \tag{3.12}$$

O altă structură care permite un mod de transmisie de linie este linia bifilară (figura 3.8).

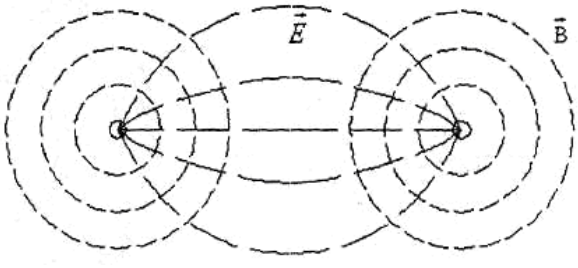


Fig. 3.8. Aspectul liniilor de câmp pentru o linie bifilară

3.2.1. Componentele de câmp pentru un mod de linie de transmisie

Se vor stabili ecuaţiile satisfăcute de componentele câmpului electric şi câmpului magnetic pentru un mod de linie de transmisie. Dacă direcţia Oz este direcţia de propagare a undei, atunci pentru un mod de linie de transmisie componentele E_z şi H_z sunt nule.

Ecuaţiile vectoriale ale câmpului electric şi ale câmpului magnetic sunt:

$$\nabla \times \vec{H} = \varepsilon \frac{\partial \vec{E}}{\partial t}$$
$$\nabla \times \vec{E} = -\mu \frac{\partial \vec{H}}{\partial t} \tag{3.13}$$

Avem o descompunere într-o componentă transversală şi o componentă longitudinală (în lungul direcţiei de propagare). Cum:

$$\nabla = \nabla_t + \vec{k} \frac{\partial}{\partial z} \tag{3.14}$$

Pentru câmpul electric şi magnetic se poate scrie:

$$\vec{H} = \vec{H}_t + \vec{k} H_z$$
$$\vec{E} = \vec{E}_t + \vec{k} E_z \tag{3.15}$$

unde $\vec{H}_t$ şi $\vec{E}_t$ reprezintă componentele în planul perpendicular direcţiei de propagare Oz ($\vec{H}_t = \vec{i} H_x + \vec{j} H_y$; $\vec{E}_t = \vec{i} E_x + \vec{j} E_y$). Componentele E_z şi H_z nu sunt unde, acestea având valori constante în timp. Ecuaţiile (3.13) se mai scriu astfel:

$$\left(\nabla_t + \vec{k} \frac{\partial}{\partial z}\right) \times \vec{H}_t = \varepsilon \frac{\partial \vec{E}_t}{\partial t}$$
$$\left(\nabla_t + \vec{k} \frac{\partial}{\partial z}\right) \times \vec{E}_t = -\mu \frac{\partial \vec{H}_t}{\partial t} \tag{3.16}$$

Identificând componentele se obţine:

$$\nabla_t \times \vec{H}_t = 0 \quad \text{(a)}$$

$$\nabla_t \times \vec{E}_t = 0 \quad \text{(b)}$$

$$\vec{k}\frac{\partial}{\partial z} \times \vec{H}_t = \varepsilon \frac{\partial \vec{E}_t}{\partial t} \quad \text{(c)} \qquad (3.17)$$

$$\vec{k}\frac{\partial}{\partial z} \times \vec{E}_t = -\mu \frac{\partial \vec{H}_t}{\partial t} \quad \text{(d)}$$

Ecuaţiile (3.17a) şi (3.17b) arată că distribuţia câmpului este oarecum ca în regim staţionar (câmpuri irotaţionale) şi în consecinţă pentru calculul capacităţii se procedează ca în electrostatică, iar pentru calculul inductivităţii ca în magnetostatică. Relaţiile (3.17c) şi (3.17d) reprezintă ecuaţiile telegrafiştilor în mărimi de câmp.

3.2.2. Condiţii de existenţă a modulului linie de transmisie

Nu orice structură permite propagarea unui mod de linie de transmisie (TEM). Ţinând seama că pentru un mod de linie câmpul electric este irotaţional (rel. 3.17b), rezultă că este un câmp de gradient ca în electrostatică: $\vec{E} = \nabla V$. Având în vedere că $\nabla \cdot \vec{E} = 0$, rezultă că laplacianul potenţialului electric este de asemenea zero: $\nabla^2 V = 0$.

O structură de tip ghid de undă, formată dintr-un singur conductor, are potenţial constant pe frontieră. Ţinând cont de aceasta, de faptul că există numai componente transversale pentru câmpul electric şi de laplacianul nul dedus anterior, rezultă că potenţialul trebuie să fie constant. Adică *E = 0*.

În consecinţă, pentru o structură cu un singur conductor nu există modul TEM. Pentru un sistem cu două conductoare (gen cablu coaxial sau linie bifilară) nu există nici o restricţie şi modul TEM se propaga la orice frecvenţă.

3.3. Moduri de ghid

Reamintim că orice conductor cilindric având o secţiune oarecare simplu conexă poate fi un ghid de undă. Soluţiile ecuaţiilor lui Maxwell cu condiţii de frontieră impuse de existenţa pereţilor ghidului şi determinate de configuraţia geometrică a acestuia se numesc moduri de propagare de ghid sau pe scurt – moduri de ghid.

Pentru determinarea legăturilor între mărimile câmpului electromagnetic, se va considera un ghid de undă dreptunghiular, frecvent utilizat în tehnica de microunde.

Mărimile de stare locală ale câmpului electromagnetic sunt: $\vec{E}, \vec{D}, \vec{H}, \vec{B},$ deci 12 componente scalare. Se va demonstra că aceste componente scalare se pot exprima funcţie de componentele intensităţilor câmpului electromagnetic pe direcţia de propagare: E_z şi H_z.

În demonstraţie se vor utiliza componentele transversale şi longitudinale ale câmpului electromagnetic şi a operatorilor. Astfel, ecuaţiile lui Maxwell se pot scrie sub forma:

$$\begin{aligned}\nabla\times\vec{E}=-\mu\frac{\partial\vec{H}}{\partial t}\Rightarrow\left(\nabla_t+\vec{k}\frac{\partial}{\partial z}\right)\times\left(\vec{E}_t+\vec{k}E_z\right)=-\mu\frac{\partial}{\partial_t}\left(\vec{H}_t+\vec{k}H_z\right)\\ \nabla\times\vec{H}=\varepsilon\frac{\partial\vec{E}}{\partial t}\Rightarrow\left(\nabla_t+\vec{k}\frac{\partial}{\partial z}\right)\times\left(\vec{H}_t+\vec{k}H_z\right)=\varepsilon\frac{\partial}{\partial t}\left(\vec{E}_t+\vec{k}E_z\right)\end{aligned} \tag{3.18}$$

Dezvoltând relaţiile (3.18) de mai sus, se obţine:

$$\begin{aligned}\nabla_t\times\vec{E}_t+\nabla_t\times\vec{k}E_z+\vec{k}\frac{\partial}{\partial z}\times\vec{E}_t+\vec{k}\frac{\partial}{\partial z}\times\vec{k}E_z=-\mu\frac{\partial}{\partial t}\left(\vec{H}_t+\vec{k}H_z\right)\\ \nabla_t\times\vec{H}_t+\nabla_t\times\vec{k}H_z+\vec{k}\frac{\partial}{\partial z}\times\vec{H}_t+\vec{k}\frac{\partial}{\partial z}\times\vec{k}H_t=\varepsilon\frac{\partial}{\partial t}\left(\vec{E}_t+\vec{k}E_t\right)\end{aligned} \tag{3.19}$$

dar $\vec{k}\frac{\partial}{\partial z}\times\vec{k}E_t=0$ şi $\vec{k}\frac{\partial}{\partial z}\times\vec{k}H_t=0$

Din identificarea componentelor pe direcţia longitudinală şi pe direcţia transversală se obţin ecuaţiile pentru componentele transversale:

$$\begin{aligned}&\nabla_t\times\vec{k}H_z+\frac{\partial}{\partial z}\vec{k}\times\vec{H}_t=\varepsilon\frac{\partial\vec{E}_t}{\partial t} &\text{(a)}\\ &\nabla_t\times\vec{k}E_z+\frac{\partial}{\partial z}\vec{k}\times\vec{E}_t=-\mu\frac{\partial\vec{H}_t}{\partial t} &\text{(b)}\end{aligned} \tag{3.20}$$

În ipoteza unui regim armonic, apelând la reprezentarea în complex, ecuaţiile (3.20) devin:

$$\begin{aligned}&\nabla_t\times\vec{k}\underline{H}_z+\frac{\partial}{\partial z}\vec{k}\times\underline{\vec{H}}_t=j\omega\varepsilon\underline{\vec{E}}_t &\text{(a)}\\ &\nabla_t\times\vec{k}\underline{E}_z+\frac{\partial}{\partial z}\vec{k}\times\underline{\vec{E}}_t=-j\omega\mu\underline{\vec{H}}_t &\text{(b)}\end{aligned} \tag{3.21}$$

Substituind componenta $\underline{\vec{H}}_t$, din relaţia (3.21b) în relaţia (3.21a) se obţine:

$$\nabla_t\times\vec{k}\underline{H}_z+\frac{j}{\mu\omega}\frac{\partial}{\partial z}\vec{k}\times\left(\nabla_t\times\vec{k}\underline{E}_z\right)+\frac{j}{\mu\omega}\frac{\partial^2}{\partial z^2}\vec{k}\times\left(\vec{k}\times\underline{\vec{E}}_t\right)=j\omega\mu\underline{\vec{E}}_t \tag{3.22}$$

Folosind identitatea vectorială $\vec{a}\times\left(\vec{b}\times\vec{c}\right)=\vec{b}\left(\vec{a}\cdot\vec{c}\right)-\vec{c}\left(\vec{a}\cdot\vec{b}\right)$ se obţine:

$$\nabla_t\times\vec{k}\underline{H}_z+\frac{j}{\mu\omega}\frac{\partial}{\partial z}\left(\nabla_t\underline{E}_z\right)-\frac{j}{\mu\omega}\frac{\partial^2\underline{\vec{E}}_t}{\partial z^2}=j\omega\underline{E}_t \tag{3.23}$$

În cazul undelor electromagnetice plane în medii fără pierderi, soluţia este de forma $\underline{E}_t=\underline{E}_{t0}e^{-j\beta_z t}$. În consecinţă $\frac{\partial^2}{\partial z^2}=-\beta_z^2$ şi ecuaţia (3.23) se scrie sub forma:

$$\nabla_t \times \vec{k}\underline{H}_z + \frac{j}{\mu\omega}\frac{\partial}{\partial z}\left(\nabla_t \underline{E}_z\right) + \frac{j\beta^2}{\mu\omega}\underline{\vec{E}_t} = j\omega\underline{E}_t \tag{3.24}$$

de unde:

$$\underline{\vec{E}}_t = \frac{1}{\omega^2\mu\varepsilon - \beta_z^2}\left[\frac{\partial}{\partial z}\left(\nabla_t \underline{\vec{E}}_t\right) - j\omega\mu\nabla_t \times \vec{k}\underline{H}_z\right] \tag{3.25}$$

În mod asemănător se obţine:

$$\underline{\vec{H}}_t = \frac{1}{\omega^2\mu\varepsilon - \beta_z^2}\left[\frac{\partial}{\partial z}\left(\nabla_t \underline{\vec{H}}_z\right) - j\omega\mu\nabla_t \times \vec{k}\underline{E}_z\right] \tag{3.26}$$

Relaţiile (3.25) şi (3.26) arată că se pot calcula numai componentele E_z şi H_z şi apoi în funcţie de acestea se pot calcula şi celelalte componente.

3.4. Ghidul de undă dreptunghiular

În figura 3.9 se prezintă un ghid de undă dreptunghiular, cu direcţia de propagare Oz.

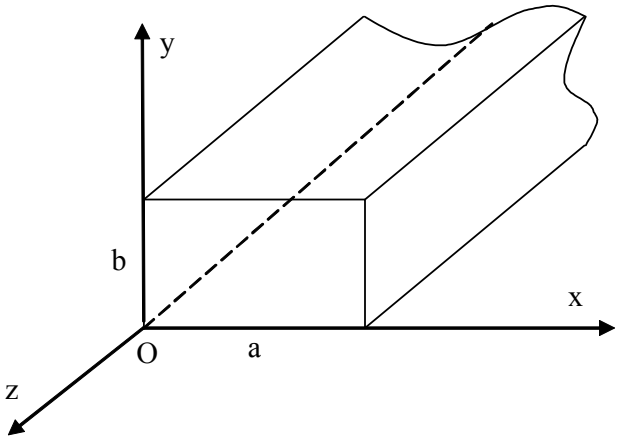


Fig. 3.9. Ghid de undă dreptunghiular

Urmărim să determinăm soluţia de propagare a undelor în ghid.

3.4.1. Soluţiile ecuaţiilor propagării în ghid dreptunghiular

Ecuaţiile satisfăcute de intensitatea câmpului electric pentru regim armonic şi intensitatea câmpului magnetic pentru regim armonic sunt:

$$\begin{aligned}\Delta\underline{\vec{E}} + \omega^2\mu\varepsilon\underline{\vec{E}} &= 0\\ \Delta\underline{\vec{H}} + \omega^2\mu\varepsilon\underline{\vec{H}} &= 0\end{aligned} \tag{3.27}$$

Integrăm mai întâi ecuaţia satisfăcută de componenta $\underline{E}_z$:

$$\frac{\partial^2 \underline{E}_z}{\partial x^2}+\frac{\partial^2 \underline{E}_z}{\partial y^2}+\frac{\partial^2 \underline{E}_z}{\partial z^2}=-\omega^2\mu\varepsilon\underline{E}_z \qquad (3.28)$$

Aplicând metoda separării variabilelor, se caută pentru $\underline{E}_z$ o soluţie de forma:

$$\underline{E}_z(x,y,z)=f_1(x)\cdot f_2(y)\cdot f_3(z)$$

Înlocuind în relaţia (3.28) se obţine:

$$f_1''(x)f_2(y)f_3(z)+f_1(x)f_2''(y)f_3(z)+f_1(x)f_2(y)f_3''(z)=-\omega^2\mu\varepsilon\cdot f_1(x)f_2(y)f_3(z)$$

ceea ce este echivalent cu (s-a pus condiţia $E_z\neq 0$):

$$\frac{f_1''(x)}{f_1(x)}+\frac{f_2''(y)}{f_2(y)}+\frac{f_3''(z)}{f_3(z)}=-\omega^2\mu\varepsilon=-k^2 \qquad (3.29)$$

În relaţia (3.29) avem o sumă de trei funcţii de variabile independente, această sumă având o valoare constantă. Din acest motiv singura posibilitate este ca fiecare raport al sumei să fie o constantă:

$$\frac{f_1''(x)}{f_1(x)}=k_x^2;\ \frac{f_2''(y)}{f_2(y)}=k_y^2;\ \frac{f_3''(z)}{f_3(z)}=k_z^2 \text{ şi } k_x^2+k_y^2+k_z^2=-k^2$$

Notând $-k_t^2=k_x^2+k_y^2$ se obţine $k_z=\pm\sqrt{k_t^2-k^2}$. Ecuaţia satisfăcută de componenta $\underline{E}_z$ este:

$$\frac{\partial^2 \underline{E}_z}{\partial z^2}=k_t^2\underline{E}_z$$

cu soluţia:

$$\underline{E}_z=E_0e^{j(\omega t-\underline{\gamma}z)} \text{ cu } \underline{\gamma}=\alpha+j\beta$$

Mediul în care are loc propagarea este dielectric fără pierderi (nu există atenuare) şi $\alpha=0$.

$$\beta=\frac{k_z}{j}=\pm j\sqrt{k_t^2-k^2}=\pm\sqrt{k^2-k_t^2} \qquad (3.30)$$

$\beta=\sqrt{k^2-k_t^2}$ reprezintă condiţia de dispersie.

Aşadar $f_3(z)=E_0e^{j(\omega t-\beta z)}$ şi atunci componenta $\underline{E}_z$ este dată de:

$$\underline{E}_z=f_1(x)f_2(y)E_0e^{j(\omega t-\beta z)} \qquad (3.31)$$

3.4.2. Condiţii de tăiere

Dacă β este real, atunci $\underline{E}_z$ dat de relaţia (3.31) este o propagare. Dacă β este imaginar, atunci $\underline{E}_z$ se atenuează în direcţia de propagare, într-un mediu presupus fără pierderi şi în consecinţă în acest caz nu există propagare.

Condiţia limită critică este dată de relaţia:

$$\beta^2=0\Rightarrow k^2=k_t^2\Rightarrow k_t=\omega\sqrt{\varepsilon\mu} \qquad (3.32)$$

Corespunzător valorii date de relaţia (3.32) rezultă frecvenţa de tăiere şi lungimea de undă de tăiere:

$$f_t = \frac{k_t}{2\pi\sqrt{\varepsilon\mu}}; \quad \lambda_t = \frac{2\pi}{k_t} \tag{3.33}$$

În paragrafele anterioare s-a stabilit că lungimea de undă în ghid λ_g este definită de lungimea de undă caracteristică undei plane (λ).

Astfel din condiţia $\beta\lambda_g = 2\pi$ se obţine

$$\lambda_g = \frac{2\pi}{\beta} \tag{3.34}$$

iar lungimea de undă pentru unda plană este:

$$\lambda = \frac{2\pi}{k} = \frac{2\pi}{\omega\sqrt{\varepsilon\mu}} \tag{3.35}$$

Dacă k_t, β, k obţinute din relaţiile (3.33), (3.34), (3.35) se introduc în condiţia de dispersie (3.30) se obţine:

$$\frac{1}{\lambda_g^2} + \frac{1}{\lambda_t^2} = \frac{1}{\lambda^2} \tag{3.36}$$

Dacă propagarea are loc în aer, atunci $\varepsilon=\varepsilon_0$, $\mu=\mu_0 \Rightarrow \lambda=\lambda_0$ şi:

$$\lambda_g = \frac{\lambda_0}{\sqrt{1-\left(\frac{\lambda_0}{\lambda_t}\right)^2}} \tag{3.37}$$

3.4.3. Condiţii de frontieră ale ghidului

Pentru ghidul de undă condiţiile de frontieră sunt de tip suprafaţă perfect conductoare. Aşadar la suprafeţele ghidurilor, componentele lui $\underline{E}_z$ se anulează. Deoarece este evident că forma funcţiilor $f_1(x)$ şi $f_2(y)$ sunt asemănătoare cu forma funcţiei $f_3(z)$, acestea pot fi puse sub următoarea formă (ţinându-se seama că atât pe direcţia x cât şi pe direcţia y, dimensiunile fiind finite, există şi unde directe şi unde inverse):

$$f_1(x) = A\sin(jk_x x) + B\cos(jk_x x)$$
$$f_2(y) = C\sin(jk_y y) + D\cos(jk_y y)$$

Aşadar punându-se condiţiile de frontieră pentru suprafaţă perfect conductoare, se obţine:

B = 0

$\sin(jk_x x) = 0$ pentru x = 0 şi x = a $\Rightarrow$

$$k_x = \frac{m\pi}{ja} \tag{3.38}$$

Analog pentru axa Oy vom avea D = 0 şi

$$k_y = \frac{n\pi}{jb} \tag{3.39}$$

Funcţiile $f_1(x)$ şi $f_2(y)$ vor avea expresiile:

$$f_1(x) = A\sin\left(\frac{m\pi}{a}x\right); \quad f_2(y) = C\sin\left(\frac{n\pi}{b}y\right)$$

Având în vedere şi relaţia (3.31), expresia câmpului electric devine

$$\underline{E}_z(x,y,z) = E_0 e^{j(\omega t-\beta z)}\sin\left(\frac{m\pi}{a}x\right)\sin\left(\frac{n\pi}{b}y\right) \tag{3.40}$$

iar k_t are expresia:

$$k_t = \sqrt{\left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2} \tag{3.41}$$

3.4.4. Componentele de câmp ale undei în ghid. Moduri de ghid

Pornim de la ecuaţiile lui Maxwell pentru un dielectric ideal

$$\nabla\times\vec{E} = -\mu\frac{\partial\vec{H}}{\partial t}, \quad \nabla\times\vec{H} = \varepsilon\frac{\partial\vec{E}}{\partial t}$$

şi dezvoltând rotorii se obţine:

$$\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ E_x & E_y & E_z \end{vmatrix} = -\mu\frac{\partial}{\partial t}\left(\vec{i}H_x + \vec{j}H_y + \vec{k}H_z\right)$$

$$\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ H_x & H_y & H_z \end{vmatrix} = \varepsilon\frac{\partial}{\partial t}\left(\vec{i}E_x + \vec{j}E_y + \vec{k}E_z\right)$$

Proiectând ecuaţiile pe cele trei coordonate şi având în vedere că $\frac{\partial}{\partial t} = j\omega$ şi $\frac{\partial}{\partial z} = -j\beta$ se obţine:

$$\frac{\partial E_z}{\partial y} + j\beta E_y = -j\omega\mu H_x \qquad \frac{\partial H_z}{\partial y} + j\beta H_y = j\omega\varepsilon E_x$$
$$\frac{\partial E_z}{\partial x} + j\beta E_x = j\omega\mu H_y \qquad \frac{\partial H_z}{\partial x} + j\beta H_x = j\omega\varepsilon E_y$$
$$\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} = -j\omega\mu H_z \qquad \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} = -j\omega\varepsilon E_z$$

Relaţiile se pot grupa două câte două astfel:

$$\begin{aligned} &\beta E_y + \omega\mu H_x = j\frac{\partial E_z}{\partial y} && \text{(a)} \\ &\omega\varepsilon E_y + \beta H_x = j\frac{\partial H_z}{\partial x} && \text{(b)} \end{aligned} \qquad (3.42)$$

care leagă pe E_y cu H_x şi

$$\begin{aligned} &\beta E_x - \omega\mu H_y = j\frac{\partial E_z}{\partial x} && \text{(a)} \\ &\omega\varepsilon E_x - \beta H_y = -j\frac{\partial H_z}{\partial y} && \text{(b)} \end{aligned} \qquad (3.43)$$

care leagă pe E_x cu H_y.

Se observă că nu există relaţii care să lege componentele E_z, de H_z acestea fiind deci soluţii independente. Soluţiile se împart astfel:

a) Moduri de propagare transversal magnetic (TM): $H_z = 0$, $E_z \neq 0$

b) Moduri de propagare transversal electric (TE): $H_z \neq 0$, $E_z = 0$

Moduri de propagare transversal magnetic

Componentele câmpului electric (E_x, E_y) şi ale câmpului magnetic (H_x, H_y) se obţin din rezolvarea ecuaţiilor (3.42) şi (3.43) pentru $H_z = 0$ şi $E_z \neq 0$.

$$\begin{cases} \beta E_y + \omega\mu H_x = j\dfrac{\partial E_z}{\partial y} \\ \omega\varepsilon E_y + \beta H_x = 0 \end{cases} \Rightarrow \begin{cases} E_y = \dfrac{j\beta}{\beta^2 - \omega^2\varepsilon\mu}\dfrac{\partial E_z}{\partial y} \\ H_x = -\dfrac{j\omega\varepsilon}{\beta^2 - \omega^2\varepsilon\mu}\dfrac{\partial E_z}{\partial y} \end{cases}$$

$$\begin{cases} \beta E_x - \omega\mu H_y = j\dfrac{\partial E_z}{\partial x} \\ \omega\varepsilon E_x - \beta H_y = 0 \end{cases} \Rightarrow \begin{cases} E_x = \dfrac{j\beta}{\beta^2 - \omega^2\varepsilon\mu}\dfrac{\partial E_z}{\partial x} \\ H_y = \dfrac{j\omega\varepsilon}{\beta^2 - \omega^2\varepsilon\mu}\dfrac{\partial E_z}{\partial x} \end{cases}$$

Deoarece $k_t^2 = -\beta^2 + \omega^2\varepsilon\mu$ şi

$$\underline{E}_z(x,y,z) = E_0 e^{j(\omega t-\beta z)} \sin\left(\frac{m\pi}{a}x\right)\sin\left(\frac{n\pi}{b}y\right)$$

componentele câmpului electric şi ale câmpului magnetic pentru modul TM sunt:

$$\begin{aligned}
\underline{E}_x(x,y,z,t) &= -j\frac{\beta}{k_t^2}E_0 e^{j(\omega t-\beta z)}\frac{m\pi}{a}\cos\left(\frac{m\pi}{a}x\right)\sin\left(\frac{n\pi}{b}y\right)\\
\underline{E}_y(x,y,z,t) &= -j\frac{\beta}{k_t^2}E_0 e^{j(\omega t-\beta z)}\frac{n\pi}{b}\sin\left(\frac{m\pi}{a}x\right)\cos\left(\frac{n\pi}{b}y\right)\\
\underline{H}_x(x,y,z,t) &= -j\frac{\omega\varepsilon}{k_t^2}E_0 e^{j(\omega t-\beta z)}\frac{n\pi}{b}\sin\left(\frac{m\pi}{a}x\right)\cos\left(\frac{n\pi}{b}y\right)\\
\underline{H}_y(x,y,z,t) &= -j\frac{\omega\varepsilon}{k_t^2}E_0 e^{j(\omega t-\beta z)}\frac{m\pi}{a}\cos\left(\frac{m\pi}{a}x\right)\sin\left(\frac{n\pi}{b}y\right)
\end{aligned} \qquad (3.44)$$

Observaţie: dacă m = 0 sau n = 0 ar rezulta că $E_z = 0$ şi cum şi $H_z = 0$ am avea mod TEM, imposibil pentru ghidul acesta. Din acest motiv la modurile TM avem m > 0 **şi** n > 0.

Moduri de propagare transversal electric

Aceste moduri de propagare se obţin pentru $E_z = 0$ şi $H_z \neq 0$. Sistemele de ecuaţii (3.42) şi (3.43) se rezolvă astfel:

$$\begin{cases}\beta E_y + \omega\mu H_x = 0\\ \omega\varepsilon E_y + \beta H_x = j\dfrac{\partial H_z}{\partial x}\end{cases}\Rightarrow\begin{cases}E_y = \dfrac{j\omega\mu}{k_t^2}\dfrac{\partial H_z}{\partial x}\\ H_x = -\dfrac{j\beta}{k_t^2}\dfrac{\partial H_z}{\partial x}\end{cases}$$

$$\begin{cases}\beta E_x - \omega\mu H_y = 0\\ \omega\varepsilon E_x - \beta H_y = -j\dfrac{\partial H_z}{\partial y}\end{cases}\Rightarrow\begin{cases}E_x = -\dfrac{j\omega\mu}{k_t^2}\dfrac{\partial H_z}{\partial y}\\ H_y = -\dfrac{j\beta}{k_t^2}\dfrac{\partial H_z}{\partial y}\end{cases}$$

Deoarece nu se poate specifica direct condiţia de frontieră în termeni de H_z, aceasta se va exprima ţinând seama de valorile de câmp electric. Aşadar:

- pentru x=0 şi x=a => E_y=0 şi $\dfrac{\partial H_z}{\partial x} = 0$
- pentru y=0 şi y=b => E_x=0 şi $\dfrac{\partial H_z}{\partial y} = 0$

Ecuaţia satisfăcută de componenta H_z a câmpului magnetic se obţine proiectând pe direcţia Oz ecuaţia (3.27):

$$\frac{\partial^2 \underline{H}_z}{\partial x^2}+\frac{\partial^2 \underline{H}_z}{\partial x^2}+\frac{\partial^2 \underline{H}_z}{\partial x^2}=-\omega^2\varepsilon\mu\underline{H}_z \tag{3.45}$$

Deoarece ecuaţia satisfăcută de componenta H_z este identică cu cea satisfăcută de componenta E_z, soluţia este de aceeaşi formă:

$$\begin{aligned}\underline{H}_z(x,y,z,t)&=\left[A\sin(jk_x x)+B\cos(jk_x x)\right]\\&\cdot\left[C\sin(jk_y y)+D\cos(jk_y y)\right]\cdot e^{j(\omega t-\beta z)}\end{aligned} \tag{3.46}$$

Constantele A, B, C, D sunt determinate din condiţiile de frontieră:

$$\left.\frac{\partial H_z}{\partial x}\right|_{x=0}=0\Rightarrow jk_x A\cos(0)-jk_x B\sin(0)=0\Rightarrow A=0$$

$$\left.\frac{\partial H_z}{\partial x}\right|_{x=a}=0\Rightarrow -jk_x B\sin(jk_x a)=0\Rightarrow k_x=-j\frac{m\pi}{a}$$

$$\left.\frac{\partial H_z}{\partial y}\right|_{y=0}=0\Rightarrow jk_y C\cos(jk_y y)-jk_y D\sin(0)=0\Rightarrow C=0$$

$$\left.\frac{\partial H_z}{\partial y}\right|_{y=b}=0\Rightarrow -jk_y D\sin(jk_y b)=0\Rightarrow k_y=-j\frac{n\pi}{b}$$

Rezultă pentru componenta H_z soluţia:

$$\underline{H}_z(x,y,z,t)=H_0 e^{j(\omega t-\beta z)}\cos\left(\frac{m\pi}{a}x\right)\cos\left(\frac{n\pi}{b}y\right) \tag{3.47}$$

Componentele transversale de câmp pentru modul TE sunt:

$$\begin{aligned}\underline{E}_x&=j\frac{\omega\mu}{k_t^2}\frac{n\pi}{b}H_0 e^{j(\omega t-\beta z)}\cos\left(\frac{m\pi}{a}x\right)\sin\left(\frac{n\pi}{b}y\right)\\ \underline{E}_y&=-j\frac{\omega\mu}{k_t^2}\frac{m\pi}{a}H_0 e^{j(\omega t-\beta z)}\sin\left(\frac{m\pi}{a}x\right)\cos\left(\frac{n\pi}{b}y\right)\\ \underline{H}_x&=-j\frac{\beta}{k_t^2}\frac{m\pi}{a}H_0 e^{j(\omega t-\beta z)}\sin\left(\frac{m\pi}{a}x\right)\cos\left(\frac{n\pi}{b}y\right)\\ \underline{H}_y&=j\frac{\beta}{k_t^2}\frac{n\pi}{b}H_0 e^{j(\omega t-\beta z)}\cos\left(\frac{m\pi}{a}x\right)\sin\left(\frac{n\pi}{b}y\right)\end{aligned} \tag{3.48}$$

Se observă că, spre deosebire de modurile TM, modurile TE există chiar dacă m=0 sau n=0, însă nu şi pentru m=n=0 deoarece în acest caz $H_x=H_y=0$ ceea ce este imposibil, deci modul TE_{00} nu există.

Din condiţia de dispersie (sau din expresia lui k_t) rezultă că pentru fiecare mod (pereche m, n) există o anumită frecvenţă de tăiere.

Dacă ghidul are a > b, atunci cea mai mică frecvenţă de tăiere este cea corespunzătoare modului TE_{10}. Acest mod se numeşte mod dominant.

Modul TE_{10}

Ne propunem să determinăm componentele de câmp pentru modul TE_{10}, care corespunde următoarele valori: m = 1; n = 0; $E_z = 0$; $H_z \neq 0$.

Pentru a obţine componentele de câmp pentru modul TE_{10}, în relaţiile (3.47) şi (3.48) se pun condiţiile: m = 1, n = 0, $E_z = 0$. Se obţin următoarele relaţii:

$$\underline{E}_x = 0, \quad \underline{H}_y = 0$$

$$\underline{E}_y = -j\frac{\omega\mu}{k_t^2}\frac{m\pi}{a}H_0 e^{j(\omega t-\beta z)}\sin\left(\frac{\pi}{a}x\right)$$

$$\underline{H}_x = -j\frac{\beta}{k_t^2}\frac{m\pi}{a}H_0 e^{j(\omega t-\beta z)}\sin\left(\frac{\pi}{a}x\right) \qquad (3.49)$$

$$\underline{H}_z = H_0 e^{j(\omega t-\beta z)}\cos\left(\frac{\pi}{a}x\right)$$

Expresiile analitice ale componentelor arată că liniile de câmp electric sunt paralele cu axa y şi variază sinusoidal pe direcţia x (figura 3.10a). Liniile de câmp magnetic formează contururi închise în planul xOz (figura 3.10b).

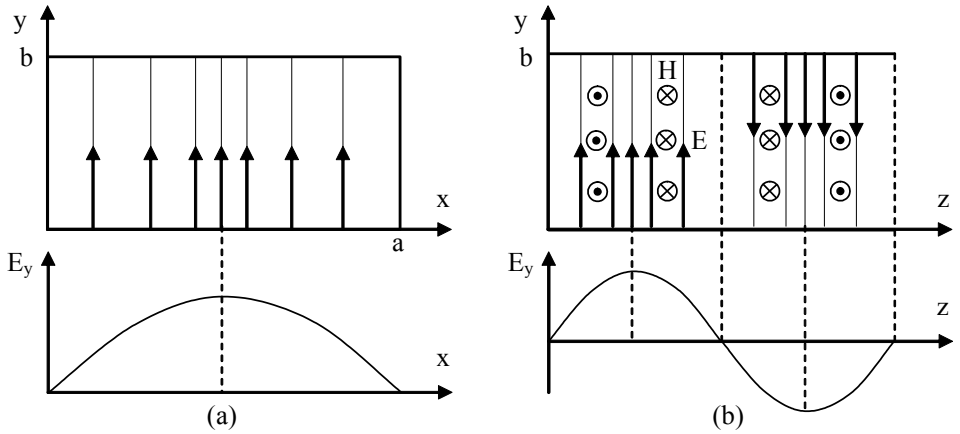


Fig. 3.10. a) Câmpul electric în ghidul de undă pentru o valoare z oarecare.
b) Componenta transversală a câmpului magnetic de a lungul axei z.

În figura 3.11 se prezintă dependenţa câmpului electric de coordonată z.

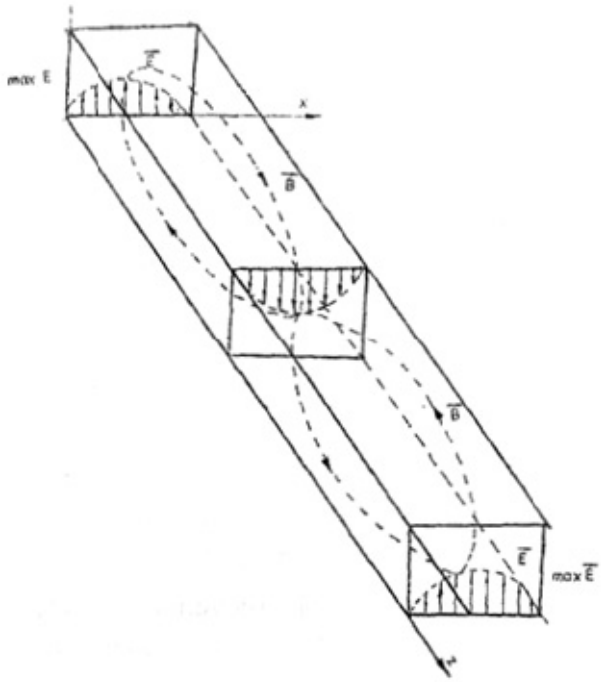


Fig. 3.11. Dependenţa câmpului electric de coordonata z

În mod analog se pot obţine nodurile: TE_{11} (m = 1; n = 1; E_z = 0); TM_{11} (m = 1; n = 1; H_z = 0) şi TM_{21} (m = 2; n = 1; H_z = 0) – reprezentate în figura 3.12.

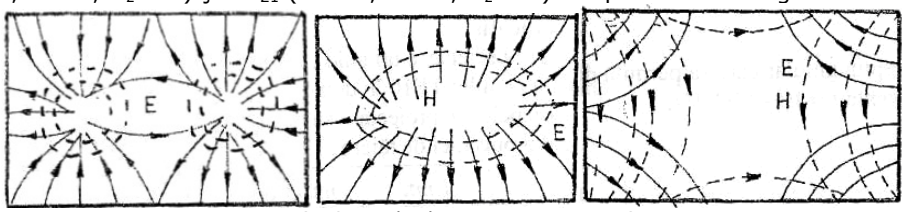


Fig. 3.12. Modurile TE11,TM11 şi TM21

Observaţii:

1. Modul dominant dintr-un ghid dreptunghiular este modul cu cea mai joasă frecvenţă de tăiere (f_0);

2. Fiind dată frecvenţa *f* a undelor ce trebuie transmise printr-un ghid, se obişnuieşte să se aleagă în practică ghidul ale cărui dimensiuni sunt astfel alese încât *f* să fie mai mare decât frecvenţa de tăiere f_0 pentru modul dominant, dar mai mică decât frecvenţa de tăiere pentru celelalte moduri. În aceste condiţii singurul mod ce se poate propaga este modul dominant.

3. Într-un ghid trebuie să se facă distincţie între:

a) Viteza de fază v_f, care este viteza cu care se propagă configuraţia şi

b) Viteza de grup v_g, care este viteza cu care energia electromagnetică sau "semnalul" ce transportă informaţie se propagă în lungul ghidului.

3.5. Pierderi în pereţi conductori

Problema câmpului electromagnetic în medii conductoare este tratată în multe lucrări de specialitate [1], [2], [4], [5], [8], [12], unde s-au determinat şi soluţiile câmpului pentru semispaţiul conductor.

Dacă la suprafaţa semispaţiului (figura 3.13) se precizează valoarea $\vec{\underline{E}}_p = \vec{i}\,\underline{E}_p$ a câmpului electric, atunci

$$\vec{\underline{H}}_p = \vec{k} \times \frac{\vec{\underline{E}}_p}{\underline{\zeta}} \tag{3.50}$$

unde:

$$\underline{\zeta} = (1+j)\sqrt{\frac{\omega\mu}{2\sigma}} \tag{3.51}$$

este impedanţă de undă.

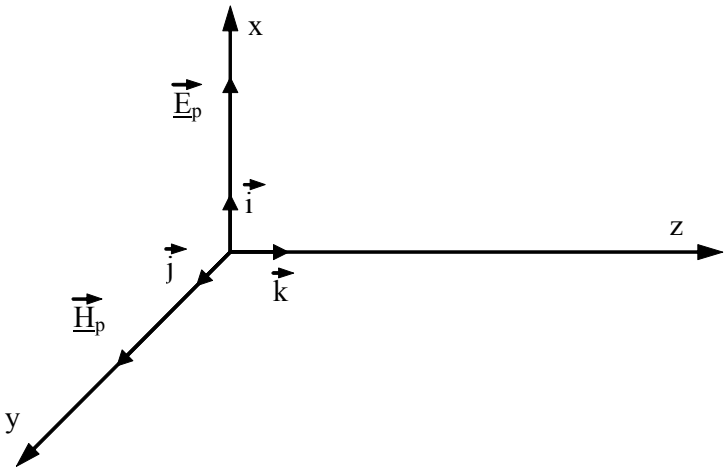


Fig. 3.13. Componentele câmpului în semispaţiul conductor

Pierderile pe unitatea de suprafaţă sunt:

$$p = \mathrm{Re}\left(\frac{E_p^2}{\underline{\zeta}}\right) \tag{3.52}$$

Relaţia (3.52) poate fi considerată condiţie de frontieră pentru dispozitivele cu microunde cu pierderi în pereţi. Aproximarea este cu atât mai bună cu cât adâncimea de pătrundere δ este mai mică în raport cu razele de curbură ale peretelui în punctul considerat. Se admite că grosimea peretelui este mai mare decât adâncimea de pătrundere $\delta = \sqrt{\frac{2}{\sigma\omega\mu}}$.

3.6. Cavităţi rezonante

Cavităţile rezonante sunt rezonatori electromagnetici cu pereţi metalici. Rezonatorii electromagnetici au proprietăţi similare cu circuitele electrice cu parametrii distribuiţi.

Cavităţile rezonante se clasifică în două categorii:

- Cavităţi rezonante care provin dintr-un ghid neuniform;
- Cavităţi rezonante care provin dintr-un ghid uniform (figura 3.14).

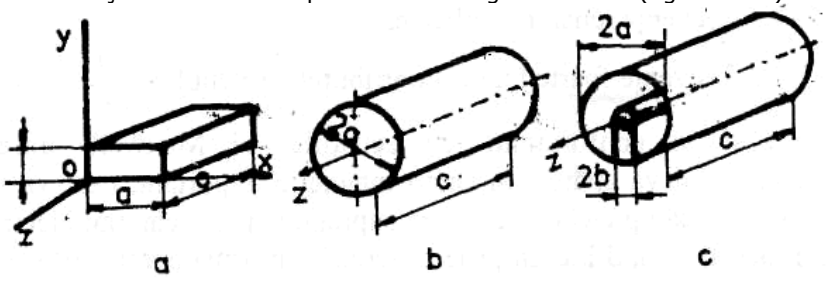


Fig. 3.14. Cavităţi rezonante care provin dintr-un ghid uniform

Pentru că se înţeleg mai uşor fenomenele care determină comportarea cavităţilor metalice ca şi circuite oscilante, precum şi generarea modurilor de oscilaţie, se va studia un exemplu simplu: comportarea unui condensator plan la frecvenţe foarte înalte.

Fie un condensator plan, cu armăturile realizate din discuri de rază R (figura 3.15).

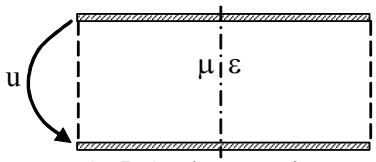


Fig 3.15. Condensator plan

La frecvente joase, câmpul electric din interiorul condensatorului are o distribuţie care poate fi considerată uniformă, cu liniile de câmp perpendiculare pe armături, orientarea acestora alternând în sus şi în jos în funcţie de frecvenţa tensiunii aplicate.

La frecvenţe foarte înalte, contribuţia termenului $\frac{\partial D}{\partial t}$ nu mai poate fi neglijată şi în conformitate cu ecuaţiile lui Maxwell, între plăcile condensatorului va apărea un câmp magnetic important, care va genera la rândul său un câmp electric ce se va suprapune peste câmpul electric iniţial, în felul acesta având loc un proces iterativ. În urma acestui proces iterativ, câmpul electric din interiorul condensatorului este de forma:

$$\underline{E} = E_0 e^{j\omega t} J_0(x)$$

unde J_0 este seria Bessel (figura 3.16) şi

$$c = \frac{1}{\sqrt{\varepsilon\mu}},\ x = \frac{\omega r}{c}$$

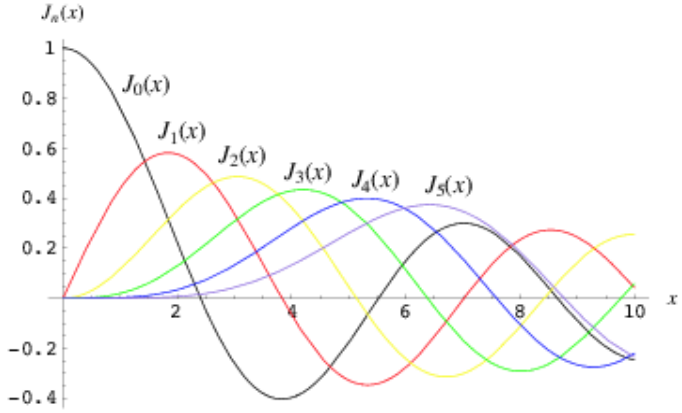


Fig. 3.16. Funcţii Bessel

Dacă frecvenţa este foarte mare, astfel încât la marginea condensatorului (r = R), $x = \frac{\omega r}{c} = 4$ câmpul electric va avea o valoare destul de mare (figura 3.17), dar în sens opus faţă de ceea ce s-ar fi aşteptat, în interiorul condensatorului va exista şi un câmp magnetic. Se poate spune că la frecvenţe foarte înalte, condensatorul se comportă şi ca o inductanţă.

Aşadar se poate considera că s-a obţinut un circuit oscilant. Dacă între plăcile condensatorului se introduce un cilindru metalic de rază a şi înălţime b (figura 3.18), prin acesta nu va exista o scurgere de curent între armături, deoarece E(a)=0.

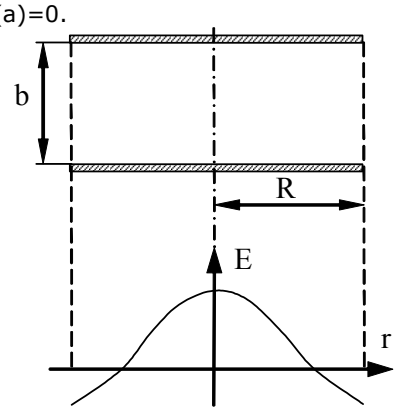


Fig 3.17. Distribuţia câmpului electric între armăturile condensatorului

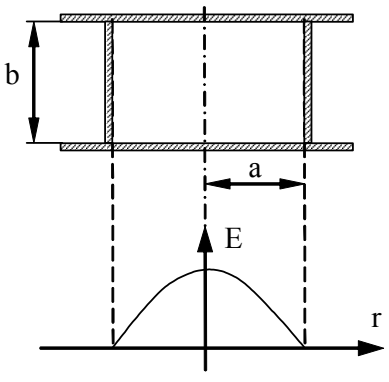


Fig 3.18. Cavitate cilindrică

Înlăturând surplusul de armături ($r > a$), se obţine o cavitate cilindrică fără legătură cu exteriorul, în care câmpul electric şi câmpul magnetic se întreţin reciproc. Amplitudinile câmpurilor şi liniile de câmp variază ca în figura 3.19.

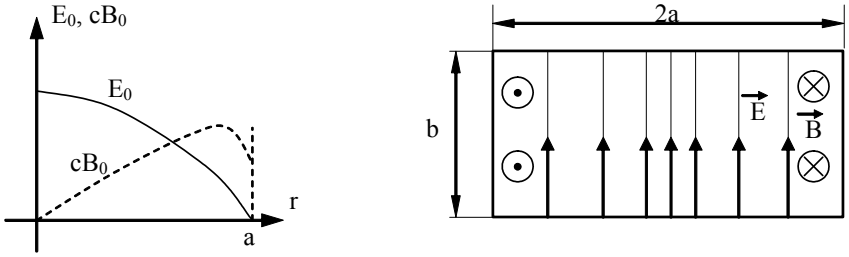


Fig. 3.19. Distribuţia câmpului electromagnetic în cavitate

Observaţii:

– Fenomenele descrise au loc numai pentru frecvenţa $\omega_0 = 2{,}405\frac{c}{a}$

– Pentru o cutie reală, curenţii induşi în pereţii cutiei determină pierderi de energie din cauza rezistenţei electrice a pereţilor şi oscilaţiile vor scădea treptat până la anulare.

– Din graficul variaţiei câmpului magnetic se observă că acesta scade brusc la marginile cutiei, ceea ce înseamnă că în peretele vertical al cutiei apare un curent. Acest curent a apărut după înlăturarea surplusului de armături (r > a), ca urmare a efectelor marginale care apar, efecte ce nu influenţau în situaţia când armăturile aveau raza R. Acest curent care apare în peretele vertical al cutiei este cel care generează sarcini electrice alternative pe armăturile de sus şi de jos.

3.6.1. Moduri de oscilaţie într-o cavitate

Întreţinerea oscilaţiilor electromagnetice în cavitate se poate realiza prin practicarea unui orificiu prin care se introduce o buclă generatoare (figura 3.20), iar pentru a extrage semnalul şi a se putea constata fenomenul de rezonanţă diametral opus se practică un alt orificiu prin care se introduce o buclă similară.

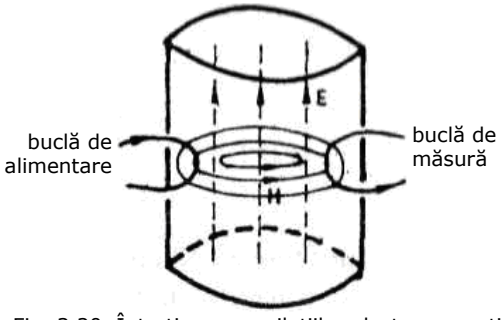


Fig. 3.20. Întreţinerea oscilaţiilor electromagnetice

Dacă frecvenţa generatorului de semnal este egală cu frecvenţa de rezonanţă (f_0) a cavităţii, atunci curentul prin bucla de măsură este maxim.

Dacă frecvenţa generatorului se măreşte peste valoarea f_0, dar nu mai mare decât frecvenţa corespunzătoare celui de al doilea zero al funcţiei Bessel (figura 3.16), se constată că se mai obţin şi alte valori maxime ale curentului prin bucla de măsură. Aceasta înseamnă că mai sunt şi alte frecvenţe de rezonanţă.

Acest lucru se explică astfel: prima frecvenţă de rezonanţă (f_0) corespunde unui aranjament al câmpului electric orientat vertical şi al câmpului magnetic orientat orizontal. Dar mai sunt posibile şi alte configuraţii de câmpuri, care rezultă ca soluţii ale ecuaţiilor lui Maxwell cu condiţia ca liniile câmpului electric să fie perpendiculare pe pereţi. Astfel că în interiorul cavităţii sunt posibile şi alte moduri de oscilaţii (figura 3.21).

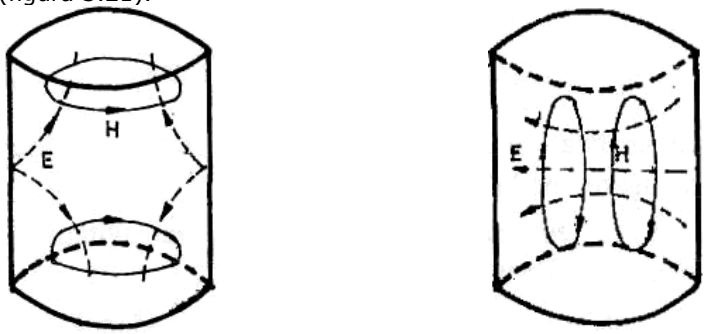


Fig. 3.21. Moduri de oscilaţie în cavitatea cilindrică

3.6.2. Cavitatea paralelipipedică. Metoda reflexiilor.

Pentru studiul cavităţilor provenind din ghid uniform se poate aplica metoda generală pornind de la ecuaţiile lui Maxwell, dar se poate folosi o metodă mai simplă – metoda reflexiilor.

Cavitatea paralelipipedică (figura 3.22) se obţine dintr-un ghid de undă dreptunghiular, la care în planele $z = 0$ şi $z = c$ s-au introdus pereţi plani perfect conductori, perpendiculari pe axa z.

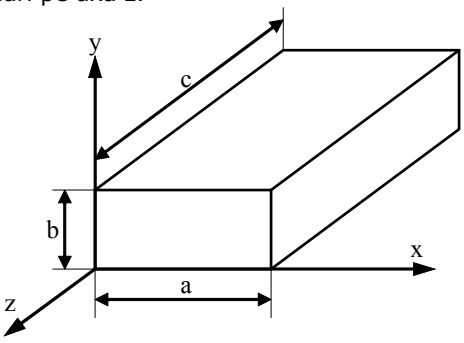


Fig. 3.22. Cavitatea paralelipipedică

Pentru analiza câmpului electromagnetic se va considera un caz simplu de propagare: modul TE_{10} (unda H_{10}). Se pune problema determinării dimensiunii *c* a cavităţii la rezonanţă în funcţie de lungimea de undă în ghid, λ_g.

Componentele câmpului electromagnetic în planul z = c pentru unda incidentă sunt E_{xd}, E_{yd}, E_{zd}, iar pentru unda reflectată E_{xr}, E_{yr}, E_{zr}.

Deoarece peretele z = c este perfect conductor, el reprezintă pentru unda câmpului electric un scurtcircuit şi coeficientul de reflexie –1; componenta tangenţială totală trebuie să fie nulă:

$$\vec{E}_{yd} + \vec{E}_{yr} = 0 \Rightarrow E_{yd} = -E_{yr}$$

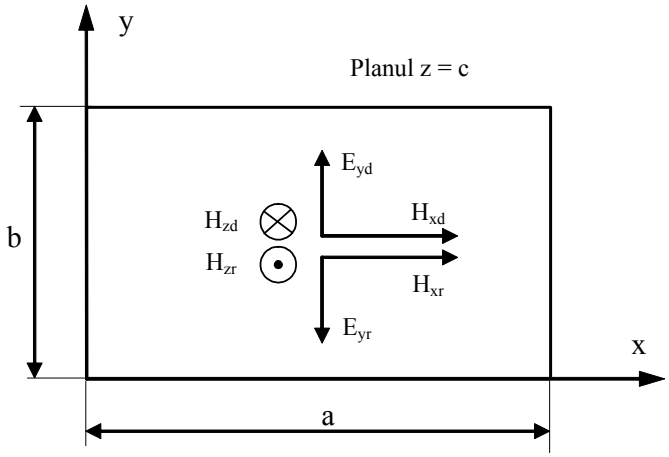


Fig. 3.23. Componentele câmpului electromagnetic în cavitatea paralelipipedică

Deoarece vectorul Poynting al undei directe $\vec{S}_d = \vec{E}_{yd} \times \vec{H}_{xd}$ are sensul axei z pozitiv şi vectorul Poynting al undei reflectate $\vec{S}_r = \vec{E}_{yr} \times \vec{H}_{xr}$ trebuie să aibă sensul axei z negativ, rezultă că H_{xd} şi H_{xr} au acelaşi sens.

Peretele din planul z = c fiind perfect conductor, rezultă că pe direcţia normală componenta rezultantă a intensităţii câmpului magnetic este nulă:

$$\vec{H}_{zd} + \vec{H}_{zr} = 0 \Rightarrow H_{zd} = -H_{zr} \tag{3.53}$$

Ţinând seama de expresiile componentelor câmpului electromagnetic pentru ghidul dreptunghiular, modul TE_{10} (3.49), prin însumarea valorilor acestora în planul z = c se obţine:

$$E_{yt} = E_{yd} - E_{yr} = -jH_0 e^{j\omega t} \frac{\omega\mu}{k_t^2 a}\left(e^{j\beta z} - e^{-j\beta z}\right)\sin\frac{\pi x}{a} \tag{3.54}$$

$$H_{xt} = H_{xd} + H_{xr} = -jH_0 e^{j\omega t} \frac{\beta\pi}{k_t^2 a}\left(e^{j\beta z} + e^{-j\beta z}\right)\sin\frac{\pi x}{a} \tag{3.55}$$

$$H_{zt} = H_{zd} - H_{zr} = -H_0 e^{j\omega t}\left(e^{j\beta z} - e^{-j\beta z}\right)\cos\frac{\pi x}{a} \tag{3.56}$$

Aşadar componentele câmpului electromagnetic în cavitate sunt:

$$E_y = -2H_0 e^{j\omega t}\frac{\omega\mu}{k_t^2 a}\cdot\sin\frac{\pi x}{a}\cdot\sin\beta z$$

$$H_x = -jH_0 e^{j\omega t}\frac{\beta\pi}{k_t^2 a}\cdot\sin\frac{\pi x}{a}\cdot\cos\beta z \qquad (3.57)$$

$$H_z = -2jH_0 e^{j\omega t}\cdot\cos\frac{\pi x}{a}\cdot\sin\beta z$$

Deoarece pe peretele perfect conductor componenta tangenţială a câmpului electric este nulă:

$$E_y\Big|_{z=c} = 0 \Rightarrow \sin\beta c = 0 \Rightarrow \beta c = p\pi \qquad (3.58)$$

şi deoarece $\beta = \frac{2\pi}{\lambda_g}$ rezultă pentru dimensiunea c:

$$c = p\frac{\lambda_g}{2} \qquad (3.59)$$

Aşadar, lungimea unei cavităţi la rezonanţă este un multiplu întreg de jumătăţi de lungimi de undă în ghid. Dacă în relaţia (3.58) se înlocuieşte valoarea constante de defazare în ghid (β) în funcţie de constanta de defazare în aer (β_0) şi numărul de undă k_t se obţine:

$$\omega^2\varepsilon_0\mu_0 - k_t^2 = \left(\frac{p\pi}{c}\right)^2 \qquad (3.60)$$

Deoarece pentru ghidul de undă dreptunghiular:

$$k_t^2 = \left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2$$

rezultă pulsaţia la rezonanţă

$$\omega = \frac{1}{\sqrt{\varepsilon_0\mu_0}}\sqrt{\left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2 + \left(\frac{p\pi}{c}\right)^2} \qquad (3.61)$$

şi lungimea de undă la rezonanţă

$$\lambda_0 = \frac{2}{\sqrt{\left(\frac{m}{a}\right)^2 + \left(\frac{n}{b}\right)^2 + \left(\frac{p}{c}\right)^2}} \qquad (3.62)$$

În cavitatea paralelipipedică, unda directă şi unda reflectată păstrează aceeaşi distribuţie transversală ca şi în ghidul dreptunghiular.

Din analiza fenomenelor din cavitatea rezonantă şi ghidul de undă uniform se deduc următoarele observaţii:

1. În ghidul uniform are loc fenomenul de propagare (se spune că în ghid este o undă progresivă), iar în cavitatea rezonantă unda este staţionară, adică se produce un fenomen de oscilaţie.

2. Ghidul uniform lucrează într-o bandă de frecvenţe [f_c, ∞) pe când spectrul de frecvenţe într-o cavitate este discret, acestea corespunzând unei anumite combinaţii de valori întregi ale numerelor m, n, p.

3. Pe ghidul de undă uniform se transportă energie activă, iar în cavitate se înmagazinează energie reactivă;

4. În ghidul uniform toate componentele transversale ale câmpului electromagnetic sunt în fază şi decalate cu $\pi/2$ în raport cu componenta axială. În consecinţă planele transversale în care valorile componentelor transversale sunt maxime sunt decalate cu $\lambda_g/4$ faţă de planele transversale în care componenta axială este maximă.

În cavitatea rezonantă componentele câmpului electric sunt în fază şi defazate cu $\pi/2$ în raport cu componentele câmpului magnetic care sunt de asemenea în fază. Din acest motiv în cavitate sunt momente în care există numai câmp magnetic şi momente când există numai câmp electric. În figura 3.24 este reprezentat câmpul electromagnetic într-o cavitate paralelipipedică cu mod de oscilaţie TEM_1 (H_{101}).

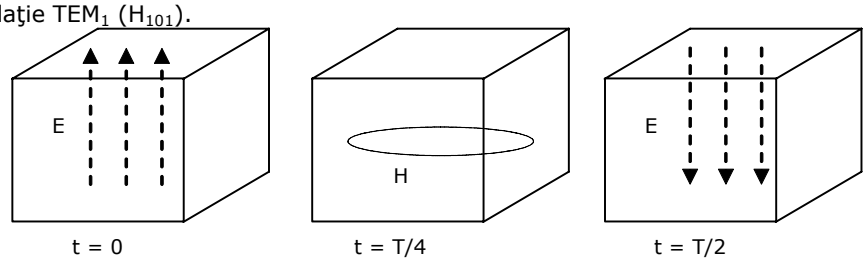


Fig. 3.24. Câmpul electromagnetic într-o cavitate paralelipipedică cu mod de oscilaţie TEM_1 (H_{101})

3.6.3. Parametrii cavităţilor rezonante

Se ştie că un circuit rezonant simplu poate fi caracterizat de parametrii săi R, L, C sau de:

- frecvenţa de rezonanţă f_0
- factorul de calitate Q
- rezistenţa la rezonanţă R_0

Deoarece parametrii R, L, C sunt foarte mici în cazul unei cavităţi rezonante, semnificativi rămân parametrii f_0, Q şi R_0.

Frecvenţa de rezonanţă

Pentru cavitatea paralelipipedică, frecvenţa de rezonanţă este:

$$f_0 = \frac{c_0}{2\pi\lambda_0} \tag{3.63}$$

şi ţinând seama de relaţia (3.62) se obţine:

$$f_0 = \frac{c_0}{2}\sqrt{\left(\frac{m\pi}{a}\right)^2 + \left(\frac{n\pi}{b}\right)^2 + \left(\frac{p\pi}{c}\right)^2} \tag{3.64}$$

Factorul de calitate

Factorul de calitate al unei cavităţi rezonante se defineşte prin relaţia:

$$Q = 2\pi \frac{W_{m0}}{W_p} = 2\pi \frac{W_{e0}}{W_p} = 2\pi \frac{W_0}{W_p} \tag{3.65}$$

unde:

- W_{m0} este energia înmagazinată în câmpul magnetic la rezonanţă;
- W_{e0} este energia înmagazinată în câmpul electric la rezonanţă;
- W_p reprezintă energia activă pierdută în cavitate (de obicei în pereţii conductori).

Energia activă pierdută într-o perioadă este:

$$W_p = P_p T \tag{3.66}$$

Puterea activă pierdută se poate calcula cu relaţia (3.67):

$$P_p = \frac{1}{2} R_p \int_{\Sigma} H_t^2 dA \tag{3.67}$$

iar energia înmagazinată în câmpul magnetic la rezonanţă este:

$$W_0 = \frac{\mu_0}{2} \int_{v_\Sigma} H^2 dv \tag{3.68}$$

Se poate obţine astfel pentru factorul de calitate expresia:

$$Q = \frac{\omega_0 \mu_0}{R_p} \cdot \frac{\int_{v_\Sigma} H^2 dv}{\int_{\Sigma} H_t^2 dA} \tag{3.69}$$

Deoarece $\omega_0 \mu_0 = 2\pi f_0 \mu_0 = 2\pi \frac{c_0}{\lambda_0} \mu_0 = \frac{2\pi}{\lambda_0} \sqrt{\frac{\mu_0}{\varepsilon_0}} = 2\pi \frac{Z_0}{\lambda_0}$

şi $R_p = \frac{1}{\sigma\delta}$ unde $\delta^2 = \frac{2}{\omega_0 \mu_0 \sigma} = \frac{2}{2\pi \frac{Z_0}{\lambda_0} \sigma} = \frac{\lambda_0}{\pi Z_0 \sigma} \Rightarrow \sigma = \frac{\lambda_0}{\pi Z_0}$ rezultă:

$$R_p = \frac{\pi Z_0}{\lambda_0 \delta} \tag{3.70}$$

Introducând relaţia (4.19) în relaţia (4.18) se obţine:

$$Q = \frac{2}{\delta} \cdot \frac{\int_{v_\Sigma} H^2 dv}{\int_{\Sigma} H_t^2 dA} \tag{3.71}$$

3.6.4. Perturbaţiile cavităţii rezonante

Există două moduri de perturbaţie a frecvenţei de rezonanţă a unei cavităţi:

– perturbaţie prin modificarea volumului cavităţii;
– perturbaţie prin modificarea parametrilor electrici şi magnetici ai mediului din interiorul cavităţii.

Perturbaţii prin variaţia volumului

Fie o cavitate rezonantă ideală delimitată de suprafaţa Σ şi cu volumul V (cu pereţii şi dielectricul din interior fără pierderi) pentru care frecvenţa de rezonanţă este f_0, şi cu valorile câmpului electromagnetic $\vec{E}_0$, $\vec{H}_0$ (figura 3.25).

Dacă cavitatea rezonantă suferă o deformaţie astfel încât suprafaţa sa devine Σ', volumul interior V' şi cu valorile câmpului electromagnetic $\vec{E}$, $\vec{H}$, frecvenţa de rezonanţă devine *f* (figura 3.25).

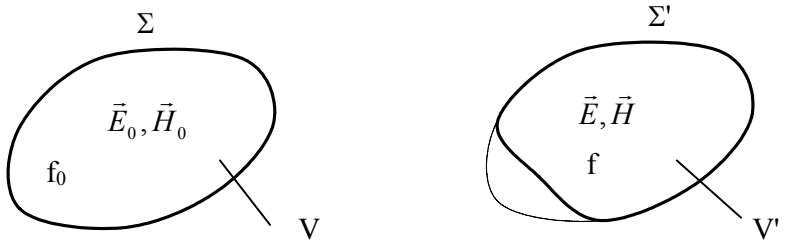


Fig. 3.25. Cavitate rezonantă iniţială şi deformată

Se pune problema determinării variaţiei relative a frecvenţei:

$$\frac{\Delta f}{f_0} = \frac{f - f_0}{f_0}$$

Deoarece frecvenţa de rezonanţă este legată de egalitatea energiilor înmagazinate în câmpul electric şi câmpul magnetic la rezonanţă, pentru calculul variaţiei frecvenţei de rezonanţă se calculează variaţia energiilor înmagazinate în câmp, ca urmare a variaţiei volumului cavităţii cu respectarea condiţiilor de frontieră pentru componentele de câmp.

Soluţionând problema în condiţiile expuse mai sus se obţine pentru variaţia relativă a frecvenţei expresia:

$$\frac{\Delta f}{f_0} = \frac{\Delta W_m - \Delta W_e}{W_0} = \frac{w_m - w_e}{w_0}\frac{\Delta V}{V} \tag{3.72}$$

unde:

$$\Delta W_m = \frac{1}{2}\int_{\Delta V} \mu_0 \underline{\vec{H}}_0^* \underline{\vec{H}} dv = \frac{1}{2}\int_{\Delta V} \mu_0 H^2 dv = w_m \Delta V$$

$$\Delta W_e = \frac{1}{2}\int_{\Delta V} \varepsilon_0 \underline{\vec{E}}^* \underline{\vec{H}} dv = \frac{1}{2}\int_{\Delta V} \varepsilon_0 E^2 dv = w_e \Delta V \tag{3.73}$$

$$W_0 = \frac{1}{2}\int_V \mu_0 \underline{\vec{H}}_0^* \underline{\vec{H}}_0 dv = \frac{1}{2}\int_V \mu_0 H_0^2 dv = w_0 V$$

sau

$$W_0 = \frac{1}{2}\int_V \varepsilon_0 \underline{\vec{E}}_0^* \underline{\vec{E}}_0 dv = \frac{1}{2}\int_V \varepsilon_0 E^2 dv = w_0 V \tag{3.74}$$

Perturbaţii prin variaţia parametrilor mediului din cavitate

Fie o cavitate rezonantă de volum V şi suprafaţă Σ, în interiorul căreia mediul are parametrii ε, μ iar câmpul electromagnetic are valorile $\vec{E}_0$, $\vec{H}_0$ şi frecventa de rezonanţă este f_0. Dacă în interiorul cavităţii care îşi conservă dimensiunile geometrice, parametrii mediului devin $\varepsilon + \Delta\varepsilon$, $\mu + \Delta\mu$, rezultă că valorile câmpului se modifică devenind $\vec{E}$, $\vec{H}$ şi frecventa de rezonanţă *f*, (figura 3.26).

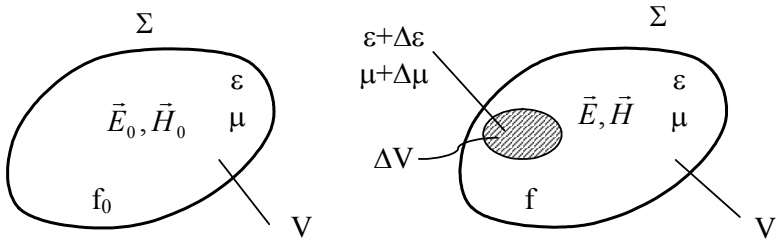


Fig. 3.26. Cavitate rezonantă iniţială şi perturbată

Pentru determinarea variaţiei relative a frecvenţei de rezonanţă procedând în mod analog ca în paragraful precedent şi se obţine:

$$\frac{\Delta f}{f_0} = \frac{w_e \dfrac{d\varepsilon}{\varepsilon} + w_m \dfrac{d\mu}{\mu}}{2w_0} \frac{\Delta V}{V} \tag{3.75}$$

unde w_e, w_m, w_0 au semnificaţia din paragraful precedent şi ΔV este volumul mediului perturbator introdus în cavitate.

$$\begin{aligned} d\varepsilon &= (\varepsilon_r - 1)\varepsilon_0 \\ d\mu &= (\mu_r - 1)\mu_0 \end{aligned} \tag{3.76}$$

3.7. Aplicatoare electromagnetice

Unul din principalele avantaje ale folosirii microundelor constă în pătrunderea instantanee a radiaţiei cu microunde care dezvoltă căldură în toată masa, ceea ce determină o reducere considerabilă a duratei tratamentului termic în raport cu procedeul clasic de transfer de căldură prin conducţie termică.

Adaptată procedeelor industriale, automatizate, încălzirea cu microunde permite o mare flexibilitatea în utilizare, spaţiul redus pe care îl ocupă sistemele de încălzire asigură o asociere mai flexibilă şi cu alte procedee tehnologice.

Sistemele de încălzire cu microunde se pot amplasa la capătul benzilor de fabricaţie sau se pot intercala între alte diferite dispozitive de tratament.

Aparatele cu microunde au în componenţă trei elemente: generatorul de microunde (magnetron sau klystron), dispozitivul de ghidare a energiei de microunde (ghid de unde) şi aplicator (componentele în care se desfăşoară procesul de încălzire propriu zis). Forma aplicatoarelor este diversă în funcţie de aplicaţia dorită.

Pe lângă elementele de bază ale aparatelor cu microunde, acestea mai sunt prevăzute cu:

- capcane în jurul orificiilor în scopul diminuării riscurilor de pierderi de microunde;
- dispozitive de absorbţie a energiei (sarcini de apă) în cazul funcţionării fără sarcină sau în cazul în care produsele care urmează a fi tratate prezintă discontinuităţi;
- detectori de arc şi circulatori (permit trecerea undelor doar într-un singur sens) utilizaţi pentru protecţia generatorului de microunde.

O clasificare a aplicatoarelor poate fi făcută în funcţie de utilizarea în instalaţiile industriale astfel:

- aplicatoare cu undă mobilă;
- aplicatoare monomod;
- aplicatoare multimod;
- aplicatoare cu structuri speciale.

3.7.1. Aplicatoare cu undă mobilă

Aplicatoarele cu undă mobilă permit deplasarea undelor electromagnetice în lungul acestora. Componentele câmpului electromagnetic $\vec{E}$ şi $\vec{H}$ sunt perpendiculare între ele, conţinute în acelaşi plan, variază sinusoidal în timp şi spaţiu şi se deplasează liber în direcţie perpendiculară pe planul determinat de $\vec{E}$ şi $\vec{H}$.

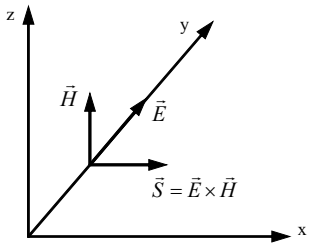


Fig. 3.27. Componentele câmpului electromagnetic şi vectorul Poynting

Într-un aplicator paralelipipedic cu dimensiunile secţiunii transversale a, b se pot propaga o multitudine de moduri de oscilaţie, TE_{mn} sau TH_{mn} unde m şi n indică numărul de semiperioade ale lui E respectiv H în lungul coordonatelor y şi z.

Pentru protejarea sursei de microunde, capătul liber al aplicatorului se conectează pe o sarcină reziduală care poate fi reglată, astfel ca în permanenţă să fie îndeplinite condiţiile de rezonanţă.

Aplicatoarele cu undă mobilă prezintă la intrare un VSWR (*voltage standing wave ratio*) bun şi se utilizează pentru încălzirea materialelor în flux continuu. Nu sunt eficiente pentru încălzirea materialelor cu pierderi mici, deoarece dimensiunile axiale ar fi prea mari.

Din categoria aplicatoarelor cu undă mobilă cele mai utilizate sunt: aplicatorul axial şi aplicatorul serpentină.

Aplicatorul axial

Aplicatorul axial este cel mai simplu aplicator cu undă mobilă. În principiu este un ghid de undă prin care se propagă modul TE_{10} cu componenta câmpului electric E în plan orizontal, în interiorul căruia sarcina se deplasează axial în acelaşi sens sau în sens contrar cu fluxul de microunde.

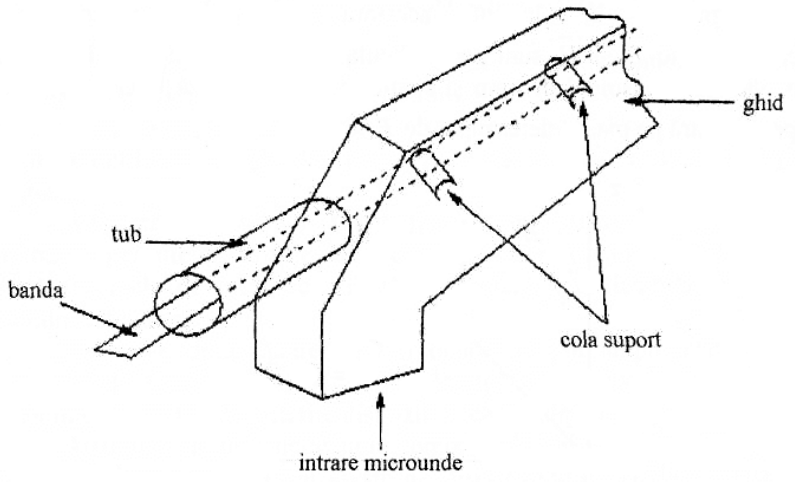


Fig. 3.28. Aplicator axial cu unda mobilă

Aplicatoarele de acest tip sunt utilizate pentru încălzirea materialelor sub formă de benzi, cu lăţimi de aproximativ $\frac{3}{8}\lambda$.

Aplicatorul serpentină

Aplicatorul serpentină (figura 3.29) este realizat din mai multe ghiduri de undă dreptunghiulare înseriate electric şi mecanic. În fiecare ghid se practică câte o fantă dreptunghiulară în direcţie longitudinală, prin care se deplasează materialul de încălzit.

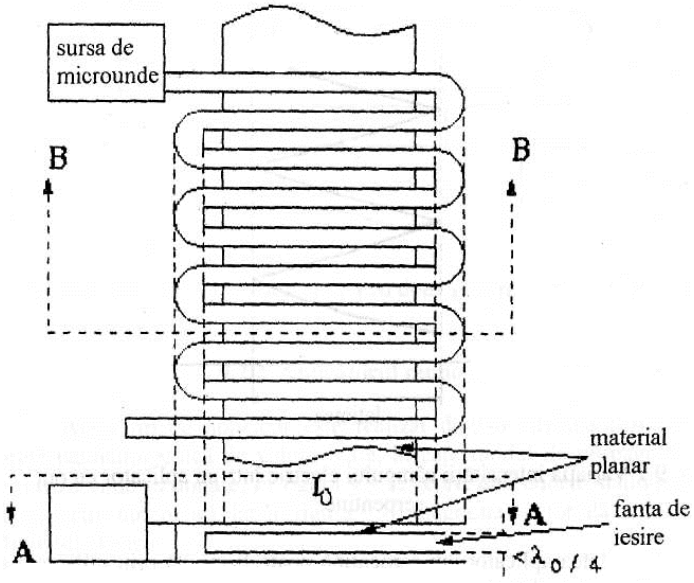


Fig. 3.29. Aplicator în formă de serpentină

În fiecare ghid component al aplicatorului, sarcina absoarbe aproximativ 10 ÷ 40% din energia de microunde, iar atenuarea corespunzătoare este cuprinsă între 0,4 ÷ 1,5 dB.

Intensitatea câmpului electric scade în mod constant de la sursă către capătul aplicatorului, aşa cum reiese din diagrama prezentată în figura 3.30.

Pentru a nu perturba modul de propagare al undelor în aplicator este necesar ca înălţimea fantelor dreptunghiulare să nu depăşească $\lambda/4$.

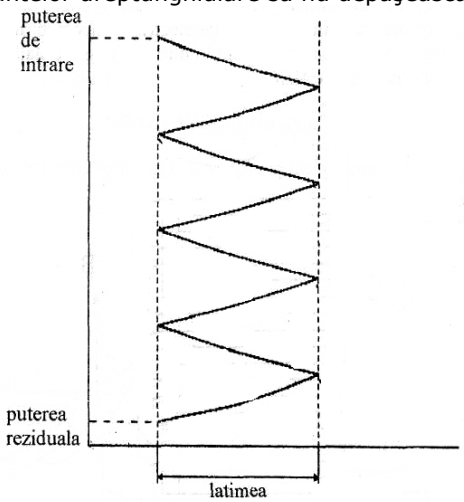


Fig. 3.30. Variaţia intensităţii câmpului electric
într-un aplicator cu opt serpentine

Eficacitatea aplicatorului serpentină poate fi mărită prin utilizarea a două surse identice de microunde care emit energie în ghiduri paralele şi dispuse astfel încât oscilaţiile generate de acestea să fie decalate între ele cu $\lambda/4$ (figura 3.31).

Un astfel de aplicator realizează o distribuţie uniformă a temperaturii pe suprafaţa materialului (sarcină).

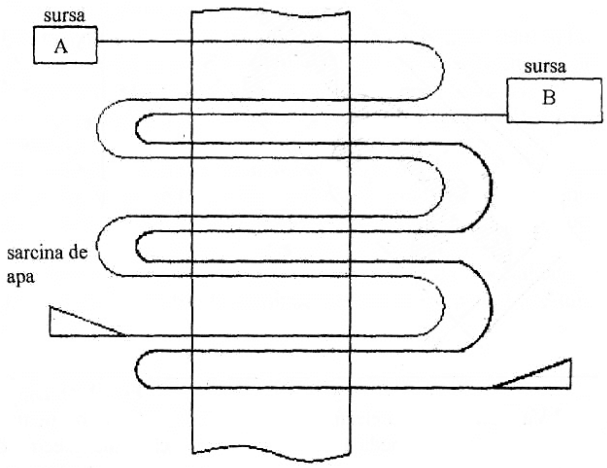


Fig. 3.31. Aplicator de tip serpentină cu două generatoare de microunde

3.7.2. Aplicatorul monomod

Acest tip de aplicator este realizat dintr-o cavitate rezonantă de formă paralelipipedică de volum mare, în care modul de propagare TE_{mnp} formează unde staţionare. În figura 3.32 se reprezintă undele staţionare într-un aplicator monomod.

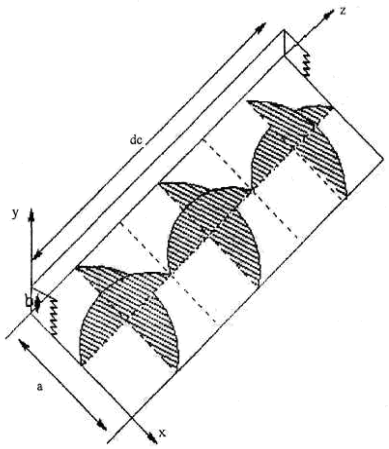


Fig. 3.32. Variaţia câmpului electric în cavitatea paralelipipedică

3.7.3. Aplicator cu două cavităţi

Acest tip de aplicator asigură o distribuţie uniformă a câmpului electric pe înălţimile produselor plane. În figura 3.33 se prezintă un aplicator realizat cu două cavităţi monomod TE_{mn0}.

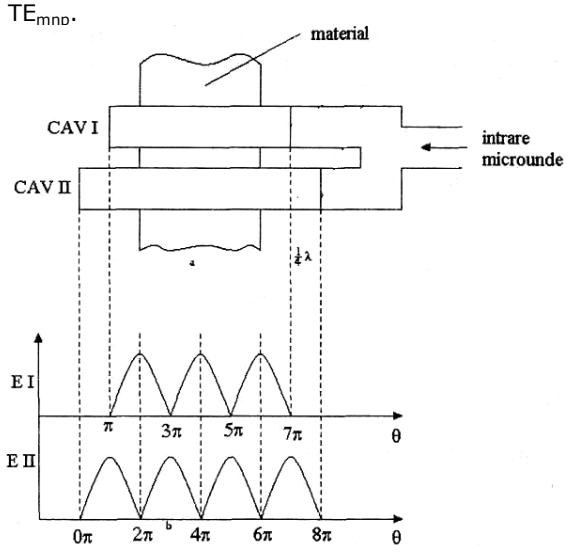


Fig. 3.33. Aplicator cu două cavităţi

Distribuţiile câmpului electric în aplicator sunt:

$$E_1 = \sqrt{2}E_f\sqrt{1-\cos\theta_1}$$

$$E_2 = \sqrt{2}E_f\sqrt{1-\cos\theta_2}$$

Efectul de încălzire este proporţional cu $\left(E_1^2 + E_2^2\right)$. Pentru ca acest efect să fie maxim în practică $\theta_1 = \theta_2 + \varphi$. In acest caz efectul termic obţinut este proporţional cu $4E_f^2$.

Pentru cavităţile cu modul TE_{mnp} de oscilaţie, intensitatea câmpului electric pe înălţimea produsului se poate aprecia cu relaţia:

$$E_y = \frac{8P}{a\lambda \cdot abm}\sqrt{\frac{\mu_0}{\varepsilon_0}}$$

3.7.4. Aplicatorul multimod

Aplicatorul multimod este de departe cel mai des folosit în practică. Din punct de vedere mecanic un astfel de aplicator este foarte simplu, constând dintr-o cutie metalică închisă şi accesorii, deci sunt uşor de fabricat. Popularitatea aplicatoarelor multimod se datorează nu numai simplicităţii constructive, ci şi abilităţii de a procesa o gamă diversă de sarcini, atât ca dimensiuni cât şi ca proprietăţi electrice.

Deşi este constructiv simplu, aplicatorul multimod este dificil de analizat din punct de vedere electromagnetic [2], [13]. În principiu, cuptorul multimod suportă un număr mare de moduri rezonante de ordin mare, care se adună vectorial în spaţiu şi timp. Aplicatorul multimod paralelipipedic poate fi considerat un ghid de undă [4], al cărui secţiune este mare comparativ cu lungimea de undă a excitaţiei şi scurtcircuitat la ambele capete cu pereţi metalici. Ghidul de undă suportă multe moduri TE_{mn} şi TM_{mn}, cu lungimi de undă independente. Fiecare rezonează atunci când lungimea cuptorului este $p\lambda_g/2$ pentru modul dat, unde p este un număr întreg. Numerele m, n, p reprezintă numărul de jumătăţi de lungime de undă ale variaţiei sinusoidale a câmpului electric de-a lungul axelor principale ale cuptorului.

Există în prezent trei abordări ale proiectării şi analizei cuptoarelor multimod:

1. Metoda empirică, bazată pe experimente de laborator, înţelegerea principiilor microundelor şi experienţă în domeniu. Predicţiile cantitative obţinute astfel sunt de o acurateţe îndoielnică, iar „mărirea" unui model la scară prezintă dificultăţi.
2. O abordare de tip circuit echivalent electric, prin care structura modurilor este modelată şi performanţa generală este dată de suma modurilor individuale. Această abordare oferă câţiva parametri importanţi, dar nu poate fi aplicată ca procedură de proiectare decât dacă sarcina determină un mod de lucru aperiodic al cuptorului.
3. Metode numerice, bazate de exemplu pe metoda elementului finit în domeniu timp. Aceasta este direcţia principală a cercetărilor contemporane, deşi la momentul actual ele nu sunt suficient de dezvoltate pentru a oferi o procedură de proiectare „prietenoasă" şi de încredere. Există posibilitatea de a oferi – cu limitări în ce priveşte complexitatea configuraţiei – simulări în ce priveşte uniformitatea încălzirii şi eficienţa unei construcţii propuse.

Există două condiţii extreme în care poate funcţiona un cuptor multimod. Prima este atunci când sarcina are dimensiuni comparabile cu adâncimea de penetrare şi reflexie minimă la suprafaţă. O proporţie mare a energiei microundelor direcţionată către sarcină este absorbită. În aceste condiţii rămâne puţină energie stocată, ceea ce înseamnă ca factorii de calitate ai modurilor sunt foarte mici, rezonanţa este practic suprimată şi cuptorul operează aperiodic. A doua apare atunci când sarcina are pierderi scăzute, deci şi adâncimea de penetrare este mare comparativ cu dimensiunile fizice ale sarcinii. Aceasta este o situaţie complicată, când numai o mică parte din energia microundelor direcţionată către sarcină este absorbită la prima trecere, restul fiind reflectat de pereţii cuptorului după care trece a 2-a oară etc. Câmpul de microunde se amplifică, stocând energie în câmpurile electric şi magnetic. În acest caz, cuptorul operează în rezonanţă, cel mai adesea cu câteva moduri rezonante apropiate de frecvenţa generatorului.

În cea mai simplă formă, cuptorul este o cutie metalică rectangulară cu dimensiunile (a, b, c) în care una sau mai multe din aceste dimensiuni este un multiplu întreg al jumătăţii lungimii de undă a excitaţiei $\lambda_0/2$.

O metodă de analiză este de a considera cutia un ghid de undă cu scurtcircuit la capete, însă de fapt sunt trei astfel de ghiduri scurtcircuitate paralele cu axele principale ale incintei. Fiecare din aceste ghiduri pot suporta atât moduri TE cât şi TM.

Cum această construcţie este o cavitate rezonantă paralelipipedică (§3.6.2), modurile de rezonanţă pot fi calculate astfel (din rel. 3.62):

$$\left(\frac{2}{\lambda_0}\right)^2 = \left(\frac{m}{a}\right)^2 + \left(\frac{n}{b}\right)^2 + \left(\frac{p}{c}\right)^2$$

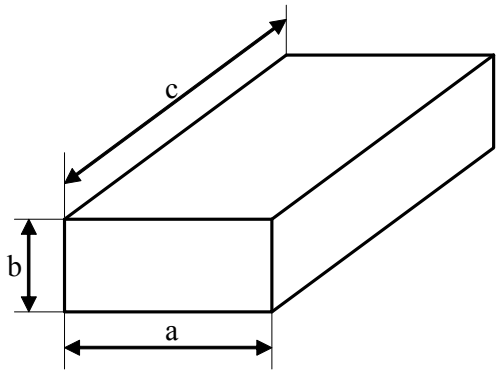


Fig. 3.34. Aplicator multimod rectangular

Este important de observat că m, n, p nu pot fi oricât de mari; ecuaţia poate avea loc numai dacă $m < 2a/\lambda_0$, $n < 2b/\lambda_0$ şi $p < 2c/\lambda_0$.

Valorile m, n, p sunt uneori numite *valori proprii*, şi împreună (m, n, p) *vectori proprii.* Se observă că modurile TE şi TM care corespund aceloraşi vectori proprii au aceeaşi frecvenţă de rezonanţă. În practică, prezenţa sarcinii şi a altor perturbaţii în incintă cauzează modurile TE şi TM corespunzătoare aceloraşi vectori proprii să aibă frecvenţe de rezonanţă diferite.

De remarcat că, în practică, aplicatoarele multimod nu se construiesc sub forma unui paralelipiped exact. Aceasta deoarece – pentru a obţine un câmp cât mai uniform – se doreşte un număr cât mai mare de moduri de rezonanţă.

3.8. Elemente specifice circuitelor de transmisie a energiei de microunde

3.8.1. Ghiduri de undă rectangulare şi curbe

Transmisia energiei de microunde de la generator la aplicator se realizează în cele multe cazuri cu ghiduri de undă cu secţiune dreptunghiulară, având dimensiunile standardizate. Dacă lungimile sunt mari, se folosesc tronsoane de ghiduri dreptunghiulare drepte sau curbate, care sunt cuplate mecanic prin flanşe şi conectate electric.

Gradul de rugozitate al suprafeţelor ghidurilor şi calitatea îmbinărilor influenţează într-o mare măsură randamentul transmisiei de la generator la aplicator. În tabelul de mai jos se prezintă dimensiunile standardizate ale ghidurilor de undă pentru frecvenţa de 2,45GHz (dimensiunile interioare).

Tab. 3.1. Dimensiuni standardizate pentru ghiduri de undă la 2,45GHz

Lăţimea a [mm]	Înălţimea b [mm]	Grosimea peretelui [mm]
109,22	54,61	2,03
110 ±0,3	55 ±0,25	2,5 ±0,25

Pentru schimbarea direcţiei traseului de microunde se recomandă utilizarea coturilor curbe şi coturilor unghiulare cu deschideri mai mari de 90°. Coturile şi ramificaţiile sunt elemente care introduc pierderi suplimentare.

3.8.2. Circulatoare

Circulatorul este un element de circuit de microunde destinul protejării generatorului de unde reflectate de sarcină. În principiu, circulatorul este un ghid de undă cu ferită, cu trei sau mai multe intrări care se intercalează pe traseul dintre generator şi aplicator. În figura 3.35 se prezintă un circulator cu patru intrări.

Funcţionarea circulatorului cu patru intrări este următoarea: semnalul aplicat la intrarea ghidului trece prin ferită şi este defazat cu $\pi/2$, intrând apoi în ghidul 2 va fi rotit cu $\pi/4$ şi în felul acesta propagarea este normală. Semnalul aplicat la ghidul 2 nu poate trece la ghidul 1, deoarece se produce o defazare cu $\pi/2$. orientarea undelor corespunzând trecerii prin ghidul 3. Dacă se aplică semnal în ghidul 3 ieşirea se va face prin ghidul 4, iar dacă se aplică un semnal în ghidul 4 ieşirea se va face în ghidul 1.

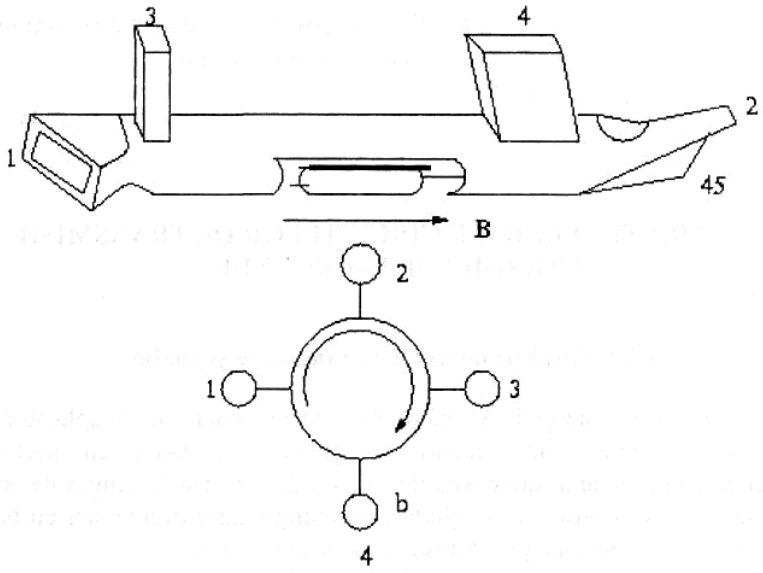


Fig. 3.35 – circulator cu patru intrări

Conexiunea între ghidul de undă şi cavitatea rezonantă se realizează prin practicarea în pereţii cavităţii rezonante a unor fante de cuplare. Fantele de cuplare au rolul de adaptare a impedanţei cavităţii cu sarcina în interior cu cea a ghidului cu care este conectat.

Calculul impedanţei de adaptare este o problemă dificilă şi de regulă aceasta este determinată experimental. În figura 3.36 se prezintă modul de transmisie al energiei de microunde de la generator la aplicator.

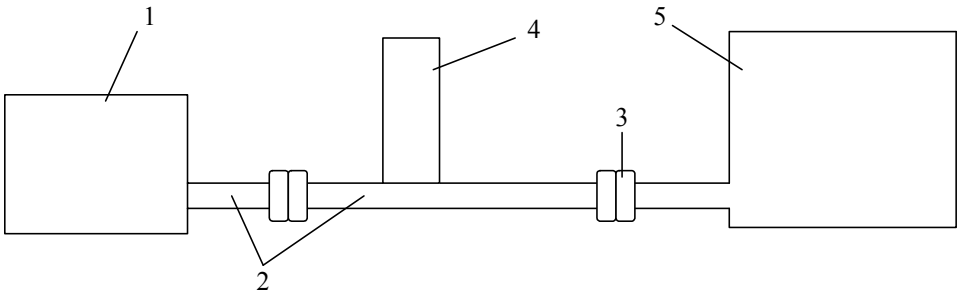


Fig. 3.36 – prezentarea schematică a modului de transmisie a energiei
1 - sursa de putere în microunde; 2 - ghid de undă; 3 - flanşă;
4 - circulator; 5 - cavitatea rezonantă

Fantele practicate în ghidurile de undă au o influenţă deosebită asupra transferului de energie, deoarece fantele devin surse de câmpuri electromagnetice perturbatoare.

4. Modelarea şi simularea proceselor electrotermice într-un cuptor multimod

4.1. Generalităţi

Utilizarea metodelor analitice pentru rezolvarea problemelor de electromagnetism devine aproape imposibilă dacă: ecuaţiile cu derivate parţiale sunt neliniare şi nu se pot liniariza fără a afecta în mod deosebit rezultatul, regiunea analizată este complexă şi condiţiile pe frontieră sunt de tip mixt sau dependente de câmp, în situaţia în care mediul este neomogen sau anizotrop. În aceste situaţii sunt preferate metodele numerice. Dintre aceste metode cea mai frecvent folosită este metoda diferenţelor finite, ca urmare a uşurinţei în înţelegere şi a modului de aplicare [2][11][13].

Din punct de vedere istoric, metoda a fost aplicată pentru prima dată în anul 1920 de A. Thorn pentru rezolvarea unor ecuaţii neliniare din hidrodinamică. În esenţă, metoda constă în înlocuirea ecuaţiilor diferenţiale cu ecuaţii cu diferenţe finite. Aproximările cu diferenţe finite sunt forme algebrice în care valoarea unei variabile dependente de poziţia unui punct în spaţiu se înlocuieşte cu valorile acesteia în punctele învecinate.

Metoda diferenţelor finite presupune parcurgerea următoarelor etape:

- discretizarea regiunii spaţiale într-o reţea de noduri;
- aproximarea ecuaţiilor diferenţiale cu ecuaţii cu diferenţe finite, în care se pune în evidenţă dependenţa valorii unei variabile într-un punct din spaţiu de valorile din zona apropiată;
- rezolvarea ecuaţiilor algebrice în condiţiile impunerii anumitor valori pe frontieră şi/sau iniţiale.

4.2. Discretizarea spaţială

În general, metodele numerice utilizate pentru rezolvarea ecuaţiilor cu derivate parţiale pot fi divizate în două categorii: metode bazate pe aproximarea directă a derivatelor în ecuaţiile cu derivate parţiale şi metode care aproximează soluţia ecuaţiilor diferenţiale printr-o combinaţie de funcţii de testare. Metoda diferenţelor finite (MDF) se încadrează în prima categorie. Cele două categorii de metode diferă în ceea ce priveşte modul de realizare a discretizării spaţiale. La MDF nodurile reţelei se află de-a lungul direcţiilor de coordonate constante, astfel că aproximarea derivatelor se poate realiza uşor (fig. 4.1 a).

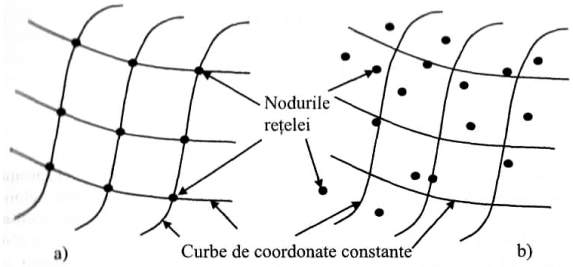


Fig. 4.1. Dispunerea nodurilor reţelei de discretizare spaţială la intersecţia curbelor de coordonate constante a) sau în poziţii oarecare b)

La metodele în care se aproximează soluţia prin funcţii de încercare nu se impun asemenea restricţii (fig. 4.1 b) deoarece derivatele aproximate sunt obţinute după substituţia soluţiei aproximate. Această observaţie conduce la concluzia că MDF nu este suficient de flexibilă pentru a se putea aplica unor domenii cu variaţii puternice ale scărilor caracteristice. Pentru a se remedia deficienţa se pot căuta sisteme de coordonate care să se adapteze cât mai bine frontierelor sau să se utilizeze o tehnică evoluată de reţele adaptive. În cele mai multe aplicaţii regiunea din spaţiu se împarte cu ajutorul unei reţele rectangulare echidistante. În cazul discretizării unor forme particulare de corpuri se poate adopta un alt tip de reţea spaţială. Pentru cazul domeniilor bidimensionale, ilustrat în figura 4.2, sunt prezentate câteva cazuri particulare în care se poate discretiza domeniul analizat.

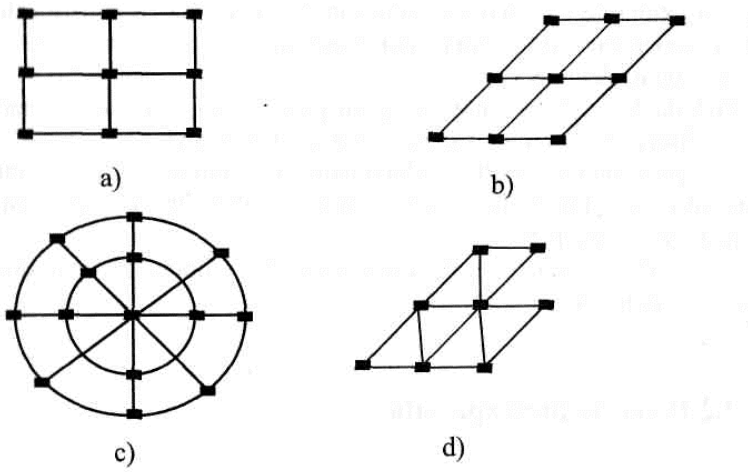


Fig. 4.2. Reţea de elemente dreptunghiulare a), înclinate b), circulare c) şi triunghiulare d)

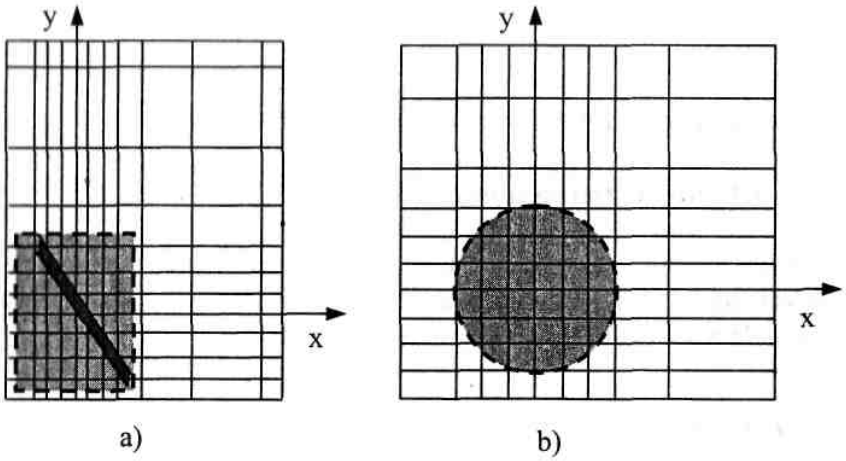


Fig. 4.3. Exemple de discretizare spaţială cu pas variabil

De asemenea, discretizarea se poate face cu pas constant sau pas variabil (vezi fig. 4.3 a sau b). Un exemplu de utilizare a discretizării, în cazul unei linii de transmisiuni realizată din două conductoare de formă poligonală în secţiune transversală, este ilustrat în fig. 4.4. Zona haşurată corespunde regiunii dielectrice în care se vor evalua componentele câmpului electromagnetic.

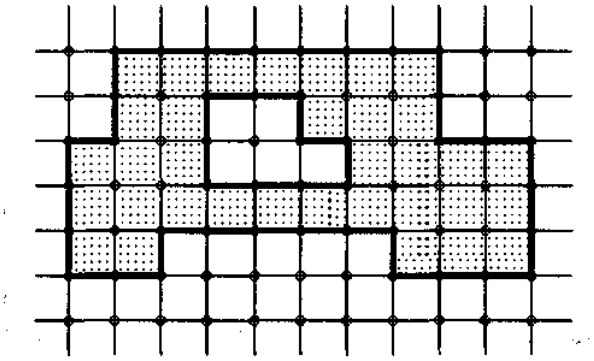


Fig. 4.4. Domeniu poligonal specific liniilor cu două conductoare

4.3. Diferenţierea numerică

Dacă se cunoaşte valoarea unei funcţii în punctul x, atunci se poate calcula noua valoare într-un punct apropiat x + h (h fiind o deplasare foarte mică faţa de x) folosind dezvoltarea în serie Taylor:

$$f(x+h)=f(x)+hf_x'+\frac{1}{2}h^2f_x''+\frac{1}{6}h^3f_x'''+\ldots \tag{4.1}$$

Dacă se scoate f_x' din relaţia anterioară se va obţine:

$$\begin{aligned} f_x' &= \frac{f(x+h)-f(x)}{h}-\frac{1}{2}hf_x''+\frac{1}{6}h^2f_x'''+\ldots= \\ &= \frac{f(x+h)-f(x)}{h}+0(h) \end{aligned} \tag{4.2}$$

Relaţia (4.2) reprezintă o aproximare a derivatei funcţiei *f* printr-o diferenţă înainte şi un termen de eroare de ordinul *h*. În mod similar, pentru punctul x-h se va obţine:

$$f(x-h)=f(x)-hf_x'+\frac{1}{2}h^2f_x''-\frac{1}{6}h^3f_x'''+\ldots \tag{4.3}$$

şi

$$\begin{aligned} f_x' &= \frac{f(x)-f(x-h)}{h}+\frac{1}{2}hf_x''-\frac{1}{6}h^2f_x'''-\ldots= \\ &= \frac{f(x)-f(x-h)}{h}+0(h) \end{aligned} \tag{4.4}$$

Relaţia (4.4) reprezintă o aproximare a derivatei funcţiei *f* printr-o diferenţă înapoi şi un termen de eroare de ordinul h. Scăzând (4.3) din (4.1) va rezulta:

$$f'_x = \frac{f(x+h) - f(x-h)}{2h} + 0(h^2) \tag{4.5}$$

Relaţia (4.5) reprezintă o aproximare a derivatei funcţiei *f* printr-o diferenţă centrată şi un termen de eroare de ordinul h^2.

Ca urmare, pentru evaluarea derivatei de ordinul unu în punctul curent x este necesară specificarea valorilor funcţie în două puncte simetrice, aflate la o distanţă *h* (fig. 4.5). Se observă că scăderea intervalului de discretizare va conduce la îmbunătăţirea acurateţei evaluării derivatei. În plus, scăderea erorii are o variaţie pătratică la diferenţele centrate şi liniară la diferenţele înainte sau înapoi.

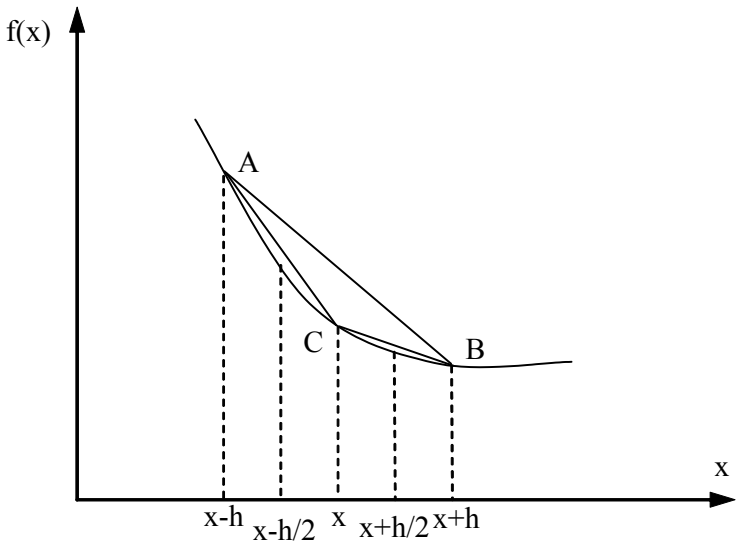


Fig. 4.5. Moduri de aproximare a derivatei unei funcţii f(x)

Dacă se evaluează derivata în punctele x+0,5h şi x-0,5h, folosind formula diferenţelor centrate, se obţine:

$$f'(x+0{,}5h) = \frac{f(x+h) - f(x)}{h},$$
$$f'(x-0{,}5h) = \frac{f(x) - f(x-h)}{h} \tag{4.6}$$

Din relaţiile anterioare se obţine expresia derivatei de ordinul doi:

$$f''(x) = \frac{f'(x+0{,}5h) - f'(x-0{,}5h)}{h} = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} \tag{4.7}$$

Eroarea cu care se face aprecierea diferenţialei de ordinul doi prin operatorul în trei puncte, folosind relaţia anterioară, este de ordinul h^2. Dacă se doreşte o precizie mai bună (de ordinul h^4), atunci se poate utiliza relaţia operatorului diferenţial în cinci puncte:

$$f''(x)=\frac{-f(x-2h)+16f(x-h)-30f(x)-16f(x+h)-f(x+2h)}{12h^2} \qquad (4.8)$$

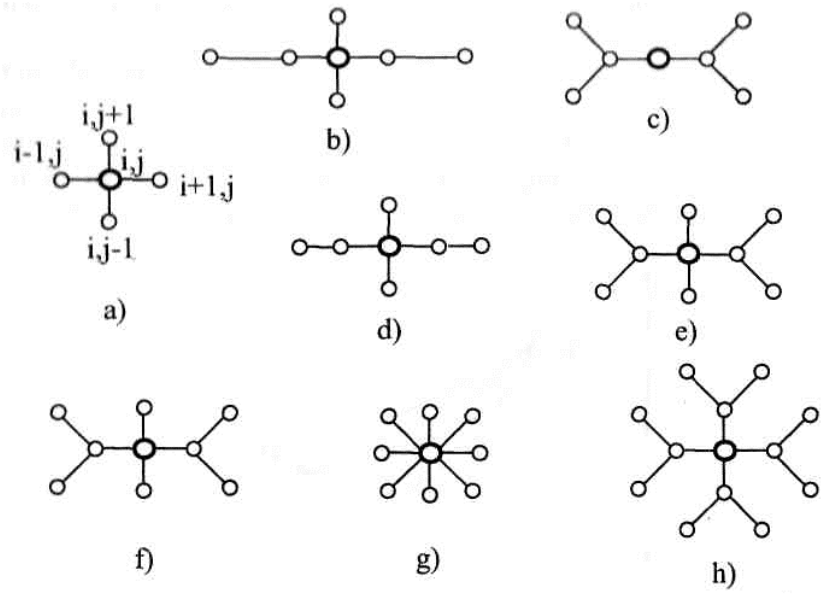


Fig. 4.6. Selecţii de puncte posibile la operatori cu diferenţe finite

În unele aplicaţii se impune un pas variabil. De exemplu, dacă se cunosc valorile în trei puncte x - h, x şi x + ah, atunci derivata de ordinul doi se aproximează cu relaţia:

$$f''(x)=\frac{af(x-h)+f(x+h)-(1+a)f(x)}{a(1+a)h^2} \qquad (4.9)$$

Cel mai des folosit operator în aplicaţiile de microunde este laplacian-ul (∇^2). În plan, laplacian-ul are forma:

$$\nabla^2\Phi=\frac{\partial^2\Phi}{\partial x^2}+\frac{\partial^2\Phi}{\partial y^2}=\frac{\Phi_{i-1,j}+\Phi_{i+1,j}+\Phi_{i,j-1}+\Phi_{i,j+1}-4\Phi_{i,j}}{h^2} \qquad (4.10)$$

Se observă că pentru a se putea evalua operatorul trebuie cunoscute valorile funcţiei în patru puncte aflate la distanţa h faţă de centru (fig. 4.6a). Convergenţa acestui tip de operator este 0(h). Pentru o mai bună acurateţe se vor utiliza mai multe puncte. De exemplu, în fig., 4.6 imaginile c, d, f şi h conduc la o convergenţă pătratică.

4.4. Discretizarea spaţio-temporală a ecuaţiilor lui Maxwell

Termenii din membrul stâng ai ecuaţiilor de evoluţie al câmpului electromagnetic conţin derivatele parţiale de ordinul doi. Abordarea numerică presupune transformarea succesivă a acestora în diferenţe centrate (diferenţe finite). Termenii din membrul doi ai aceloraşi ecuaţii conţin derivatele câmpului în raport cu timpul. În vederea procesării numerice se pot aborda două direcţii: transformarea diferenţialei de ordinul unu în diferenţă centrată (metoda diferenţelor finite în domeniul timp – MDFDT) sau se consideră componentele câmpului electromagnetic scrise în complex, astfel încât derivata se transformă în produsul funcţiei cu termenul jω (metoda diferenţelor finite în domeniul frecvenţă – MDFDF). Fiecare tehnică de modelare are anumite avantaje şi dezavantaje. La unele tipuri de modele tehnica utilizată poate procesa date rapid şi cu acurateţe, iar la alte modele fenomenul ar putea fi mai puţin precis.

MDFDT aparţine clasei generale a metodelor de modelare numerică în domeniul timp a ecuaţiilor diferenţiale. Tehnica de diferenţiere în domeniul timp este cea mai utilizată în domeniul electromagnetismului deoarece: este uşor de înţeles, uşor de implementat în software şi explorează o bandă largă de frecvenţe la o singură rulare a programului. Rezolvarea prin MDFDT presupune abordarea formei diferenţiale a ecuaţiilor lui Maxwell în două etape (de tipul „saltul broaştei”). În prima etapă se determină componenta electrică a câmpului electromagnetic pentru un anumit moment de timp, apoi componenta magnetică ş.a.m.d. Examinând ecuaţiile lui Maxwell se observă că derivata câmpului electric în raport cu timpul este dependentă de rotorul câmpului magnetic. Aceasta conduce la a afirma că variaţia câmpului electric (derivata temporală) depinde de variaţia spaţială a componentei magnetice (rotorul). Dacă se ţine seama de expresia derivatei, atunci noua valoare a câmpului electric este dependentă de vechea valoare a câmpului electric şi de variaţia spaţială a vechii valori a câmpului H în jurul componentei electrice (în descrierea simplificată anterioară s-au omis constantele). Componenta magnetică se va descrie într-o manieră asemănătoare. Noua valoare a câmpului magnetic va depinde de vechea valoare a câmpului magnetic (derivata temporală) şi de variaţia spaţială a componentei electrice în jurul componentei magnetice.

După discretizarea domeniului, trebuie specificat materialul fiecărei celule. Uzual, materialul poate fi spaţiul liber (aer), metal (conductor electric perfect) sau un material oarecare cu parametrii de material specificaţi. Condiţiile de absorbţie simulează efectul spaţiului liber aflat în afara frontierei specificate. Prin această metodă se pot caracteriza bine structurile neomogene unde metodele analitice nu se pot aplica. Calculul este acompaniat de aplicarea unei excitaţii pe un obiect sau un grup de obiecte în spaţiul tridimensional si apoi calcularea evoluţiei în timp a câmpului.

În continuare se stabileşte tipul sursei. Ea poate fi o undă continuă, un curent printr-un fir, un câmp electric între plăci paralele (uzual tensiunea între plăci), depinzând de tipul situaţiei modelate. Deoarece câmpurile E şi H sunt determinate direct, mărimile de ieşire ale procesului simulat sunt câmpurile E şi H într-un punct sau într-o serie de puncte din domeniul procesat.

Analiza se poate face în spaţii mono-, bi- sau tridimensionale. În anul 1966 Kane S. Yee a propus aproximarea în 3D prin diferenţe centrate pentru ecuaţiile diferenţiale ale lui Maxwell, atât în spaţiu cât şi în timp.

MDFDT este o tehnică intuitivă, astfel că utilizatorul poate înţelege uşor cum lucrează şi la ce se poate aştepta de la modelul dat. De asemenea, fiind o tehnică în domeniul timp, când sursa este un impuls în domeniul timp (de exemplu, impulsul gaussian), atunci se determină răspunsul într-o bandă largă de frecvenţe. Situaţia este utilă în aplicaţiile la care nu se cunosc cu exactitate frecvenţele de rezonanţă sau se doreşte analiza răspunsului într-o bandă largă de frecvenţe. Prin această tehnică se determină câmpurile E şi H din interiorul domeniului, ceea ce permite o afişare animată a deplasării câmpului prin model. Acest tip de afişare este util pentru a înţelege exact ce se întâmplă în model şi a se asigura că funcţionează corect.

Domeniul trebuie să fie discretizat, iar grila să fie foarte mică în comparaţie cu lungimea de undă (distanţa mai mică de 0,1λ) şi mai mică decât cele mai mici detalii geometrice ale modelului. Ca urmare, modul de calcul devine greoi şi solicită timp de lucru mare. Modelele de tipul firelor lungi şi subţiri vor fi mai dificil de utilizat. Metoda permite utilizatorului să specifice direct structura materialului (magnetic sau dielectric) în toate punctele, fără a fi nevoie a se recurge la diferite „presupuneri". MDFDT determină direct componentele E şi H, fără a fi nevoie de conversie. Se pot investiga efectul aperturilor, efectul de umbrire etc. Condiţia de absorbţie totală (adaptare) pe frontieră simulează spaţiul liber de după frontieră.

Analizând dezavantajele metodei se constată că deoarece pe întreg domeniul trebuie realizată reţeaua de discretizare, iar pasul trebuie să fie mai mic în comparaţie cu lungimea de undă şi mai mic decât cea mai mică denivelare din model, se obţine un domeniu de calcul foarte mare. În consecinţă, apare un timp de lucru mai mare. Modelele subţiri şi lungi (de tipul firelor) sunt mai dificil de modelat prin MDFDT. În cadrul metodei se determină câmpurile E şi H în tot domeniul. Dacă este necesar a se evalua la o distanţă oarecare (de exemplu, 10 m depărtare), atunci se impune un timp de calcul şi mai mare. Extensia câmpurilor la distanţă este posibilă prin MDFDT, dar este necesară o anumită postprocesare.

Metoda oferă următoarele avantaje:

- pot fi modelate structuri arbitrare din materiale complexe (cu sau fără pierderi, perfect conductoare sau dielectrice, materiale magnetice cu sau fără pierderi materiale anizotrope - de exemplu plasma);
- răspunsul în bandă largă este disponibil după o singură execuţie pe calculator;
- se evaluează direct câmpurile E şi H, iar în aplicaţii nu mai este nevoie de efectuarea diferitelor conversii după rularea programului;
- se pot modela cu destulă acurateţe: linii de transmisiuni de tip microstrip, linii coplanare, linii coaxiale etc.;
- în analiză se poate include interacţiunea cu dispozitive active (diode şi tranzistoare) şi/sau elemente cu constante concentrate;
- se pot calcula parametrii de repartiţie la structuri de antene;
- se poate calcula caracteristica de radiaţie în câmp îndepărtat a antenelor, plecând de la câmpurile din zona apropiată;
- permite determinarea directă a efectelor aperturilor, a efectelor de umbrire. Câmpurile se pot determina direct în interiorul sau în afara structurii.

De îndată ce obiectul a fost definit, din punct de vedere geometric şi electric, simularea poate începe prin introducerea excitaţiei (radiaţie incidenţă, tensiune sau curent). Analiza în timp se face până se ajunge la starea stabilă pentru excitaţia sinusoidală sau converge către zero la o excitaţie în impuls. Datele eşantionate pentru o eventuală postprocesare se referă la tensiuni şi curenţi (pentru parametrii

de repartiţie - S) sau impedanţe şi curenţi de suprafaţă (pentru a determina caracteristica de radiaţie).

4.4.1. Algoritmul lui Yee

În domeniul tridimensional, spaţiul aferent unui anumit obiect se poate descompune într-o sumă de paralelipipede:

$$G = \{(x_i, y_j, z_k), i = 1 \ldots I,\ j = 1 \ldots J,\ k = 1 \ldots K\} \tag{4.11}$$

Numărul total de elemente este N=IxJxK. De-a lungul fiecărei muchii a cubului se presupune că acţionează o anumită diferenţă de potenţial, iar fiecare faţă a acestuia este străbătută de un flux magnetic (fig. 2.18). Fiecare celulă Yee [2][13] va conţine şase componente de câmp E_x, E_y, E_z, H_x, H_y şi H_z.

Rezolvarea ecuaţiei integrale a câmpului electromagnetic ar conduce la scrierea, de exemplu după componenta x, a ecuaţiei:

$$E_z(i, j-1, k) - E_z(i, j+1, k) + E_z(i, j, k+1) - E_z(i, j, k-1) = \\ = \frac{\partial}{\partial t} B_x(i, j, k) \tag{4.12}$$

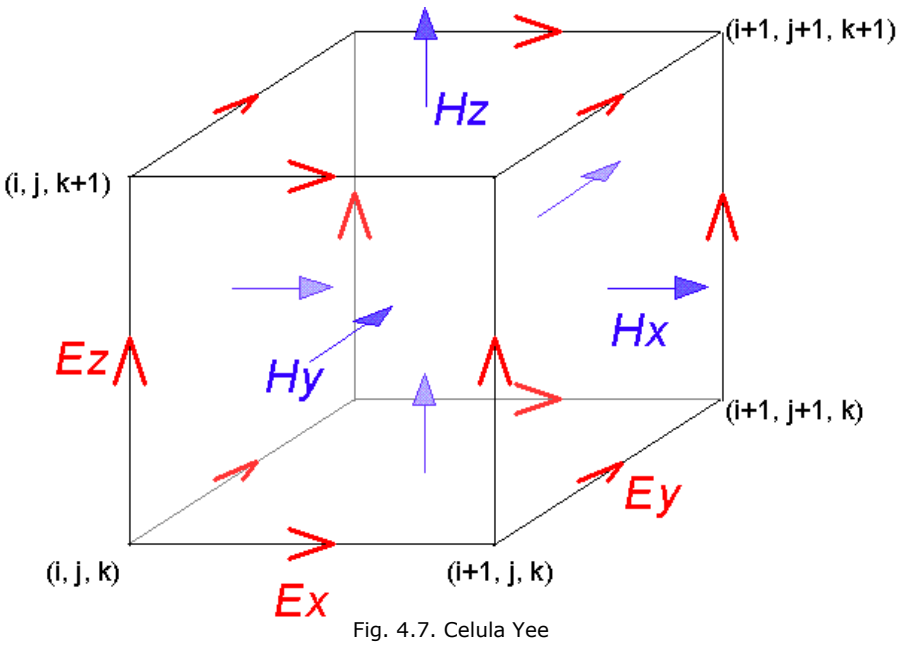


Fig. 4.7. Celula Yee

Celula unitară are dimensiunile Δx, Δy, Δz. Pe această imagine „rotorul" poate fi interpretat relativ uşor: circulaţia câmpului electric determină câmpul magnetic – cele patru componente ale câmpului electric care înconjoară câmpul B_x plasat în punctul M(i, j, k). Derivata parţială în raport cu timpul se poate înlocui prin diferenţe finite. Ca urmare, sistemul format din cele şase ecuaţii care descriu evoluţia câmpului electromagnetic (componentele scalare) va fi:

$$H_{x(i,j,k)}^{n+1/2} = H_{x(i,j,k)}^{n-1/2} + \frac{\Delta t}{\mu \Delta z}\left(E_{y(i,j,k)}^{n} - E_{y(i,j,k-1)}^{n}\right) - \frac{\Delta t}{\mu \Delta y}\left(E_{z(i,j,k)}^{n} - E_{z(i,j-1,k)}^{n}\right) \tag{4.13}$$

$$H_{y(i,j,k)}^{n+1/2} = H_{y(i,j,k)}^{n-1/2} + \frac{\Delta t}{\mu \Delta x}\left(E_{z(i,j,k)}^{n} - E_{z(i-1,j,k)}^{n}\right) - \frac{\Delta t}{\mu \Delta z}\left(E_{x(i,j,k)}^{n} - E_{x(i,j,k-1)}^{n}\right) \tag{4.14}$$

$$H_{z(i,j,k)}^{n+1/2} = H_{z(i,j,k)}^{n-1/2} + \frac{\Delta t}{\mu \Delta y}\left(E_{x(i,j,k)}^{n} - E_{x(i,j-1,k)}^{n}\right) - \frac{\Delta t}{\mu \Delta x}\left(E_{y(i,j,k)}^{n} - E_{y(i-1,j,k)}^{n}\right) \tag{4.15}$$

$$E_{x(i,j,k)}^{n+1} = E_{x(i,j,k)}^{n}\left(1 - \frac{\sigma \Delta t}{\varepsilon}\right) + \frac{\Delta t}{\varepsilon \Delta y}\left(H_{z(i,j+1,k)}^{n+1/2} - H_{z(i,j,k)}^{n+1/2}\right) - \frac{\Delta t}{\varepsilon \Delta z}\left(H_{y(i,j,k+1)}^{n+1/2} - H_{y(i,j,k)}^{n+1/2}\right) \tag{4.16}$$

$$E_{y(i,j,k)}^{n+1} = E_{y(i,j,k)}^{n}\left(1 - \frac{\sigma \Delta t}{\varepsilon}\right) + \frac{\Delta t}{\varepsilon \Delta z}\left(H_{x(i,j,k+1)}^{n+1/2} - H_{x(i,j,k)}^{n+1/2}\right) - \frac{\Delta t}{\varepsilon \Delta x}\left(H_{z(i+1,j,k)}^{n+1/2} - H_{z(i,j,k)}^{n+1/2}\right) \tag{4.17}$$

$$E_{z(i,j,k)}^{n+1} = E_{z(i,j,k)}^{n}\left(1 - \frac{\sigma \Delta t}{\varepsilon}\right) + \frac{\Delta t}{\varepsilon \Delta x}\left(H_{y(i+1,j,k)}^{n+1/2} - H_{y(i,j,k)}^{n+1/2}\right) - \frac{\Delta t}{\varepsilon \Delta y}\left(H_{x(i,j+1,k)}^{n+1/2} - H_{x(i,j,k)}^{n+1/2}\right) \tag{4.18}$$

Notaţia μ din formulele anterioare este permeabilitatea magnetică absolută, $\mu = \mu_0 \mu'$. Cu ε s-a notat permitivitatea electrică absolută reală, $\varepsilon = \varepsilon_0 \varepsilon'$. Conductivitatea σ este specifică metalelor, în cazul dielectricilor cu pierderi se va considera $\sigma = \omega \varepsilon_0 \varepsilon''$.

Exponenţii celor şase componente de câmp se referă la momentele de timp la care se face evaluarea (t=nΔt). Valorile intensităţii câmpului electric sunt considerate la momentele de timp t=nΔt, iar cele ale câmpului magnetic la t = (n+1/2)Δt. Deoarece toate sarcinile se conservă în fiecare moment, există o treaptă de timp limită (criteriul lui Courant) care conferă stabilitate. În sistemul de coordonate cartezian:

$$\Delta t \le \frac{1}{\frac{c}{\sqrt{\varepsilon'}}\sqrt{\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}}} \tag{4.19}$$

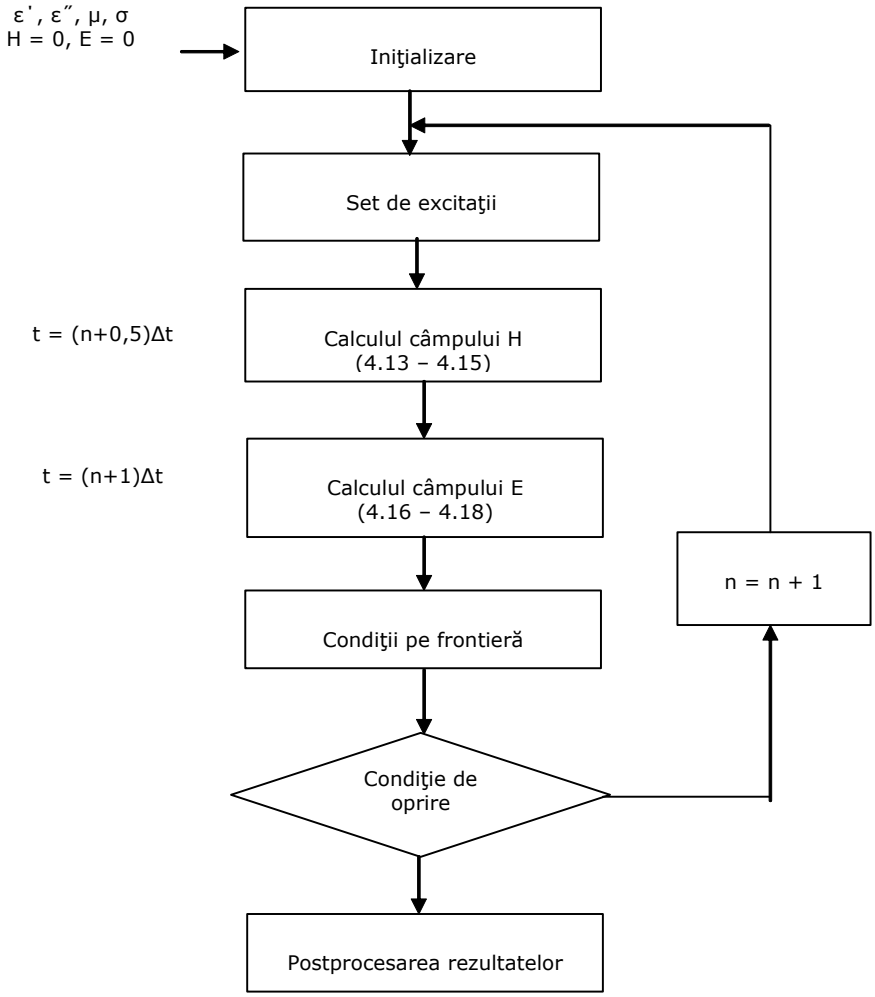


Fig. 4.8. Succesiunea prelucrării datelor prin MDFDT

În cazul în care se cunoaşte cea mai mare frecvenţă, f_{max}, la care răspunde sistemul, atunci se impune şi:

$$\Delta t \le \frac{2}{f_{\max}} \tag{4.20}$$

Se observă că toate operaţiile care apar în setul de ecuaţii sunt uşor de efectuat: adunări, scăderi sau multiplicări. Componentele câmpului din fiecare celulă Yee, caracterizată prin parametrii de material μ şi ε sunt calculate iterativ, cât timp răspunsul prezintă interes. Apoi urmează etapa de postprocesare. Schema logică care permite alcătuirea programului de evaluare a câmpului electromagnetic prin MDFDT este ilustrată în fig. 4.8.

Excitaţia poate fi o undă plană, o „pânză" de curent pe un perete metalic sau un curent printr-un fir – aceasta din urmă fiind situaţia dacă se consideră antena unui magnetron. Condiţiile de frontieră pot fi considerate separat, ca în figură, sau în cadrul calculelor pentru câmpurile E şi H. Condiţia de oprire poate fi o limită de timp (sau număr de iteraţii), sau se poate considera un criteriu de convergenţă, cum ar fi stabilizarea valorii medii a E^2 pe întregul domeniu.

4.5. Modelarea proceselor termodinamice

Fenomenele termodinamice care apar la procesele de încălzire cu microunde constau în transferuri termice în interiorul sarcinii (în principal prin conducţie) şi pierderi de căldură dinspre sarcină spre mediu. În interiorul sarcinii pot să apară conducţia termică şi transferul de entalpie prin difuzia unui fluid, de exemplu difuzia apei înspre suprafaţa mai uscată într-un material poros. Şi desigur, aportul de energie datorat câmpului de microunde.

Transferul termic dinspre sarcină spre mediu poate avea loc prin:

- conducţie termică la contactul cu o suprafaţă de suport
- convecţie
- radiaţie
- transfer de masă

În majoritatea aplicaţiilor se lucrează cu temperaturi relativ scăzute, deci fenomenul de transfer termic prin radiaţie, care depinde de puterea a 4-a a temperaturii absolute poate fi neglijat. Convecţia – liberă sau forţată – este un fenomen complex care depinde de foarte multe mărimi şi este dat în general prin formule empirice. În anumite situaţii – de exemplu la procesele de uscare – se produce un transfer de entalpie spre exterior odată cu o parte din sarcină (apa care se evaporă). Această lucrare nu tratează procese în care are loc transfer de masă.

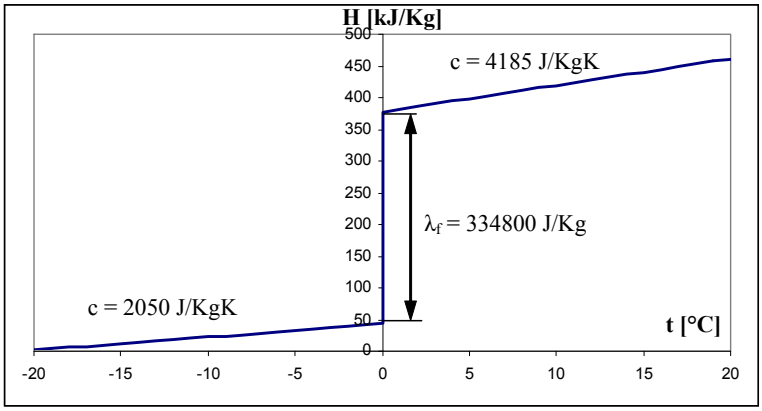


Fig. 4.9. Dependenţa H(t) pentru apa pură în jurul punctului de topire

În mod uzual sarcina constă dintr-un amestec de mai multe substanţe. Unele mărimi, cum ar fi densitatea şi căldura specifică, se pot calcula direct din parametrii componentelor. Altele, inclusiv permeabilitatea electrică şi factorul de pierderi cum s-a arătat în §2.2.5, trebuie măsurate sau preluate din literatură.

Dacă există transformări de fază izoterme sau reacţii chimice atunci este de preferat să se folosească entalpia în locul temperaturii ca mărime de stare. De exemplu, în figura 4.9 este dată entalpia în funcţie de temperatură pentru apa distilată în jurul temperaturii de 0°C. Se observă că la dezgheţ apa absoarbe o energie considerabilă fără să-şi modifice temperatura. Funcţia H(t) nu este continuă, iar inversa ei nu există.

4.5.1. Modelarea prin metoda diferenţelor finite

Această metodă se bazează pe transformarea ecuaţiei diferenţiale a transmiterii căldurii în ecuaţii cu diferenţe finite [36][37][38].

Ecuaţia diferenţială a transmiterii căldurii după cele trei axe are forma:

$$\frac{\partial t}{\partial \tau} = a \cdot \left(\frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} + \frac{\partial^2 t}{\partial z^2} \right) \tag{4.21}$$

unde:

- t este temperatura [°C] (sau [K], nu are importanţă)
- τ – timpul [s]
- a – difuzivitatea termică [m^2/s]
- x, y ,z – coordonate spaţiale

Dacă avem în vedere dependenţa conductibilităţii termice λ de temperatură vom introduce o temperatură redusă care să includă această variaţie:

$$\Phi = \int_{t_0}^{t} \frac{\lambda}{\lambda_0} dt \tag{4.22}$$

unde λ şi λ_0 sunt conductibilităţile termice la temperatura t şi la o temperatură arbitrară t_0. Astfel:

$$\frac{\partial \Phi}{\partial \tau} = a(\Phi) \cdot \left(\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} \right) \tag{4.23}$$

$$\frac{\partial \Phi}{\partial t} \cdot \frac{\partial t}{\partial \tau} \cdot \frac{\rho \cdot c}{\lambda} = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} \tag{4.24}$$

$$\frac{\lambda}{\lambda_0} \cdot \frac{\partial t}{\partial \tau} \cdot \frac{\partial H}{\partial t} \cdot \frac{\rho}{\lambda} = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} \tag{4.25}$$

$$\frac{\partial H}{\partial \tau} = \frac{\lambda_0}{\rho} \cdot \left(\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} \right) \tag{4.26}$$

Ultima relaţie este avantajoasă pentru că face posibil calculul direct al conţinutului de căldură (entalpia H), care include în faza lichidă şi căldura latentă.

S-au folosit relaţiile pentru difuzivitatea termică $a = \frac{\lambda}{\rho \cdot c}$ şi căldura specifică $c = \frac{\partial H}{\partial \tau}$.

Pentru a transforma relaţia (4.26) într-o ecuaţie cu diferenţe finite se exprimă temperatura unui punct (i, j, k) în funcţie de temperatura punctelor vecine. Se consideră pentru început cazul unui punct din interior.

Valorile i–1, i, i+1 se referă la axa X; j–1, j, j+1 la axa Y; k–1, k, k+1 la axa Z, iar n şi n+1 la succesiunea în timp. În toate relaţiile următoare Φ este la momentul de timp "n", chiar dacă acest lucru nu este precizat explicit (pentru a simplifica relaţiile). Dacă se dezvoltă funcţia $\Phi = f(x, y, z)$ în serie Taylor faţă de x şi se neglijează termenii superiori, începând cu ordinul trei, se obţine:

$$\Phi_{i-1,j,k} = \Phi_{i,j,k} - \frac{x_1}{1!} \cdot \frac{\partial \Phi_{i,j,k}}{\partial x} + \frac{x_1^2}{2!} \cdot \frac{\partial^2 \Phi_{i,j,k}}{\partial x^2} \tag{4.27}$$

şi

$$\Phi_{i+1,j,k} = \Phi_{i,j,k} + \frac{x_2}{1!} \cdot \frac{\partial \Phi_{i,j,k}}{\partial x} + \frac{x_2^2}{2!} \cdot \frac{\partial^2 \Phi_{i,j,k}}{\partial x^2} \tag{4.28}$$

de unde rezultă:

$$\frac{\partial^2 \Phi_{i,j,k}}{\partial x^2} = \frac{2}{x_1 x_2 (x_1 + x_2)} \cdot \left[x_2 \Phi_{i-1,j,k} + x_1 \Phi_{i+1,j,k} - (x_1 + x_2) \Phi_{i,j,k} \right] \tag{4.29}$$

Pentru axele y şi z se procedează identic.

Variaţia de entalpie se exprimă sub formă de diferenţă finită:

$$\frac{\partial H}{\partial \tau} \cong \frac{H_{i,j,k,n+1} - H_{i,j,k,n}}{\tau_{n+1} - \tau_n} = \frac{\Delta H_{i,j,k}}{\Delta \tau} \tag{4.30}$$

$$\begin{aligned} \frac{\Delta H_{i,j,k}}{\Delta \tau} = & \frac{2\lambda_0}{\rho \cdot x_1 x_2 (x_1 + x_2)} \cdot \left[x_2 \Phi_{i-1,j,k} + x_1 \Phi_{i+1,j,k} - (x_1 + x_2) \Phi_{i,j,k} \right] + \\ & \frac{2\lambda_0}{\rho \cdot y_1 y_2 (y_1 + y_2)} \cdot \left[y_2 \Phi_{i,j-1,k} + y_1 \Phi_{i,j+1,k} - (y_1 + y_2) \Phi_{i,j,k} \right] + \\ & \frac{2\lambda_0}{\rho \cdot z_1 z_2 (z_1 + z_2)} \cdot \left[z_2 \Phi_{i,j,k-1} + z_1 \Phi_{i,j,k+1} - (z_1 + z_2) \Phi_{i,j,k} \right] \end{aligned} \tag{4.31}$$

În cazul în care distribuţia punctelor de discretizare este omogenă de-a lungul celor două axe (caz frecvent utilizat) şi notând $x_1 = x_2 = x$, $y_1 = y_2 = y$, $z_1 = z_2 = z$ se obţine:

$$\frac{\Delta H_{i,j,k}}{\Delta\tau} = \frac{\lambda_0}{\rho \cdot x^2} \cdot \left[\Phi_{i-1,j,k} + \Phi_{i+1,j,k} - 2\Phi_{i,j,k}\right] + \frac{\lambda_0}{\rho \cdot y^2} \cdot \left[\Phi_{i,j-1,k} + \Phi_{i,j+1,k} - 2\Phi_{i,j,k}\right] + \frac{\lambda_0}{\rho \cdot z^2} \cdot \left[\Phi_{i,j,k-1} + \Phi_{i,j,k+1} - 2\Phi_{i,j,k}\right] \quad (4.32)$$

Ecuaţia cu diferenţe finite face posibilă determinarea variaţiei de entalpie într-un interval de timp $\tau_{n+1}-\tau_n$, în funcţie de temperatura punctelor vecine. Dacă se cunoaşte distribuţia iniţială de temperaturi $\Phi_{i,\,j,\,k,\,0}$ (sau entalpii $H_{i,\,j,\,k,\,0}$) se poate determina pe baza relaţiei (4.31) sau (4.32) distribuţia de temperaturi după un interval dat, rezultând $\Phi_{i,\,j,\,k,\,1}$. Pe baza acestui rezultat se poate determina distribuţia de temperaturi $\Phi_{i,\,j,\,k,\,2}$ etc. Deci printr-o metodă iterativă se poate determina evoluţia distribuţiei de temperaturi.

Ecuaţia (4.31) este valabilă pentru un punct din interior. Pentru un punct situat pe o suprafaţă limită, ecuaţia se modifică. Pentru punctele de pe suprafaţa de separaţie se scrie ecuaţia fluxului de căldură la suprafaţă:

$$W = -\lambda \cdot \frac{\partial \Phi_{i,j,k}}{\partial x} \quad (4.33)$$

unde $\Phi_{i,\,j,\,k}$ este temperatura la suprafaţă (limita considerată paralelă cu axa Y). Expresia $\lambda \cdot \frac{\partial \Phi_{i,j,k}}{\partial x}$ se poate scrie sub formă de diferenţă finită în funcţie de o temperatură $\Phi_{i+1,\,j,\,k}$ a unui punct imaginar situat la distanţa x = x_1 (pentru simplificare) de suprafaţă. Fluxul de căldură prin suprafaţă trebuie să fie:

$$W = \frac{\lambda_0}{2x} \cdot \left(\Phi_{i-1,j,k} - \Phi_{i+1,j,k}\right) \quad (4.34)$$

de unde:

$$\Phi_{i+1,j,k} = \Phi_{i-1,j,k} - \frac{2x}{\lambda_0} \cdot W \quad (4.35)$$

Relaţia (4.29) devine:

$$\frac{\partial^2 \Phi_{i,j,k}}{\partial x^2} = \frac{2}{x^2} \cdot \left[\Phi_{i-1,j,k} - \frac{2x}{\lambda_0} W - \Phi_{i,j,k}\right] \quad (4.36)$$

Pentru celelalte suprafeţe, ca şi pentru punctele din muchii şi colţuri, se procedează similar.

În cazul încălzirii cu microunde avem şi o distribuţie în volum a puterii disipate. Împărţind această valoare la densitatea materialului se obţine puterea specifică pe unitatea de masă, care conduce direct la creşterea entalpiei. Ecuaţia (4.26) devine:

$$\frac{\partial H}{\partial \tau} = \frac{p}{\rho} + \frac{\lambda_0}{\rho} \cdot \left(\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2}\right) \quad (4.37)$$

unde *p* este densitatea de putere obţinută în urma calculului electromagnetic.

4.5.2. Stabilitatea modelului cu diferenţe finite

Una din problemele critice ale aplicării prin iteraţie a ecuaţiei cu diferenţe finite este asigurarea stabilităţii şi acurateţei soluţiei în cursul integrării. Pentru a asigura stabilitatea soluţiei, adică pentru a împiedica oscilaţia soluţiei în cursul integrării, intervalul de timp între iteraţii şi dimensiunile reţelei trebuie alese în conformitate cu criteriile de stabilitate deduse. Acurateţea soluţiei, adică concordanţa cât mai bună cu soluţia analitică este de asemenea de dorit. Aceasta depinde de forma ecuaţiei cu diferenţe finite şi de desimea reţelei.

În concluzie, desimea reţelei se stabileşte pornind de la necesitatea găsirii unei rezolvări a următoarei contradicţii: pe de o parte folosirea unei reţele mai dese măreşte precizia modelului (eroarea introdusă prin ipoteza că suprafaţa adiacentă fiecărui nod are aceeaşi temperatură cu a nodului scade odată cu aria nodului reţelei); pe de altă parte o reţea deasă măreşte durata de efectuare a calculelor atât datorită creşterii numărului de noduri, cât şi reducerii intervalelor de timp între iteraţii dictate de condiţiile de stabilitate a soluţiei.

Din analiza stabilităţii ecuaţiei cu diferenţe finite se deduc următoarele criterii:

în interior

$$\Delta\tau \le \frac{1}{2a\cdot\left[\frac{1}{x_1x_2}+\frac{1}{y_1y_2}+\frac{1}{z_1z_2}\right]} \tag{4.38}$$

pe suprafaţă

$$\Delta\tau \le \frac{1}{2a\cdot\left[\frac{1}{x_1x_2}+\frac{1}{y_1y_2}+\frac{1}{z^2}+\frac{W}{\lambda\cdot z}\right]} \tag{4.39}$$

pe colţ

$$\Delta\tau \le \frac{1}{2a\cdot\left[\frac{1}{x^2}+\frac{1}{y^2}+\frac{1}{z^2}+\frac{W}{\lambda\cdot x}+\frac{W}{\lambda\cdot y}+\frac{W}{\lambda\cdot z}\right]} \tag{4.40}$$

Intervalul de timp ales reprezintă de fapt timpul în care procesul nestaţionar al transmiterii căldurii este aproximat cu un proces staţionar. Din această cauză, cu cât caracteristicile procesului real se abat mai puternic de la cele ale unui proces staţionar, cu atât trebuie să fie mai mică durata iteraţiei.

În relaţiile de calcul efectiv se observă că membrul drept depinde numai de mărimi cunoscute la momentul iteraţiei k. Dacă notăm aceste expresii cu $G_{i,\ j,\ k}$ se obţine:

$$\frac{H_{i,j,k,n+1}-H_{i,j,k,n}}{\tau_{n+1}-\tau_n}=G_{i,j,k,n} \tag{4.41}$$

Mărimea $G_{i,j,k,n}$ reprezintă de fapt viteza de variaţie a entalpiei punctului (i, j, k) în procesul considerat staţionar care începe la momentul τ_n. Cu cât această mărime este mai mare în valoare absolută, cu atât procesul este mai nestaţionar, iar intervalul de timp trebuie ales mai mic.

Soluţia optimă constă în folosirea unui interval de timp variabil şi limitarea variaţiei entalpiei la o valoare fixată ΔH_{max}. Astfel se obţine:

$$\Delta\tau_n = \tau_{n+1} - \tau_n = \frac{\Delta H_{\max}}{\max\limits_{(i,j,k)\in D}\left|G_{i,j,k,n}\right|} \tag{4.42}$$

unde D este domeniul valorilor (i, j, k).

Astfel se obţine un interval de timp mic atunci când variaţia entalpiei este mare şi un interval mare atunci când aproximarea cu un proces staţionar este mai bună.

4.6. Câmpul electromagnetic cuplat cu cel termic într-un cuptor multimod

Pentru a determina distribuţia câmpului electromagnetic într-un cuptor multimod uzual s-a realizat un program de simulare bazat pe metoda diferenţelor finite în domeniu timp [38]. Din punct de vedere informatic, trebuie avut în vedere faptul că discretizarea spaţială trebuie să îndeplinească o limită de fineţe (cea mai mare dimensiune trebuie să fie cel mult 1/10 din lungimea de undă în mediul respectiv), ceea ce limitează inferior numărul de celule din discretizare. Valori de ordinul sutelor de mii pentru numărul de celule sunt uzuale, iar discretizarea temporală are şi ea o limită inferioară de fineţe (rel. 4.19). Numărul de operaţii în virgulă flotantă poate ajunge de ordinul $10^{10} \div 10^{12}$ pentru a ajunge la regimul staţionar, ceea ce înseamnă că timpul de calcul va fi destul de lung.

Într-un sistem de operare modern, interfaţa unui program cu utilizatorul se realizează astfel: o acţiune a utilizatorului (de exemplu apăsarea unui buton) rezultă în apelarea de către sistemul de operare a unei subrutine speciale din cadrul programului cu parametri care să identifice controlul şi natura acţiunii. În timpul procesării acestei subrutine, interfaţa nu mai primeşte nici o interacţiune din partea utilizatorului (e „îngheţată"). Durata procesării trebuie deci să fie scurtă, de ordinul fracţiunilor de secundă. În cazul unui program de simulare prin MDFDT durata calculelor este lungă, deci pornirea lor nu poate fi directă.

Programele care fac calcule intensive [40] sau cele care tratează evenimente nedeterministe (de exemplu transferuri în reţea) trebuie să conţină mai mult de un singur fir de execuţie (*thread*-uri). Thread-ul original cu care este pornit programul se va ocupa de interfaţă, dar trebuie să existe cel puţin un alt thread care sa efectueze calculele. Într-un calculator uniprocesor, sistemul de operare comută în mod transparent între thread-uri suficient de rapid ca ele să pară că se execută simultan, mecanism numit partajare a timpului (*time sharing*). Sistemele multiprocesor execută simultan în mod real mai multe thread-uri, însă deoarece procesoarele sunt în număr fix şi limitat se foloseşte şi partajarea timpului.

În cazul în care se dispune de un sistem multiprocesor, este recomandat să se împartă problema iniţială într-un număr de probleme care pot fi tratate în paralel şi să se folosească mai multe thread-uri de lucru, însă nu mai multe decât numărul de procesoare pentru că în acest caz performanţa va fi afectată negativ. Este posibil să avem şi situaţii în care există mai multe tipuri de calcule intensive, de exemplu mai trebuie să şi generăm imagini care să arate evoluţia simulării, în acest caz se poate rezerva un procesor/thread pentru aceste calcule.

Un exemplu de cum se poate împărţi problema pentru calcul paralel este, dacă celulele Yee sunt numerotate c_0, c_1, ... c_{n-1} şi avem 3 procesoare disponibile atunci primul să efectueze calcule pentru c_0, c_3, c_5..., al doilea pentru c_1, c_4, c_6... iar

al 3-lea pentru c_2, c_5, c_7... Trebuie acordată o atenţie specială sincronizării thread-urilor, de exemplu nu trebuie să se înceapă calculul E la o iteraţie până nu s-a încheiat calculul H (fig. 4.8).

4.6.1. Discretizarea spaţială

Tratarea problemei de simulare începe cu descrierea geometriei sistemului de simulat. În această etapă dimensiunile exacte nu sunt relevante, doar forma şi dimensiunile aproximative.

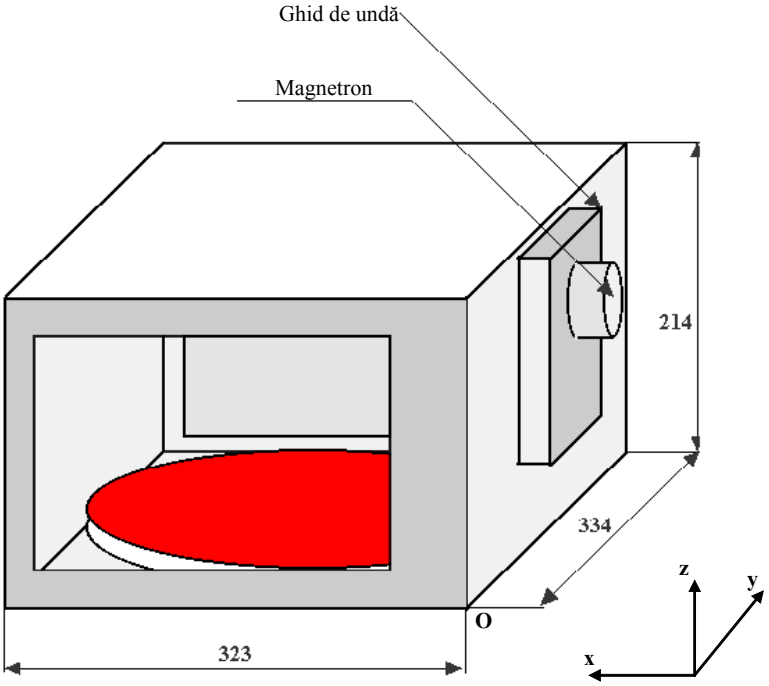


Fig. 4.10. Geometria cuptorului simulat

Geometria cuptorului este prezentată în fig. 4.10. Se observă că, dată fiind originea sistemului de coordonate în punctul O, ghidul de undă este situat la coordonate x negative. Aceasta este o simplă problemă de alegere.

Discretizarea cea mai simplă este omogenă pe axele de coordonate, adică Δx, Δy şi Δz sunt constante pentru toate celulele. Este de dorit să se realizeze discretizarea astfel încât celulele să cuprindă exact diversele elemente ale modelului, ceea ce în cazul discretizării omogene se poate face fie alegând valori foarte mici pentru Δx, Δy şi Δz fie alegând coordonatele elementelor ca multipli de Δx, Δy şi Δz. Cum aceste soluţii sunt ineficiente sau prea aproximative, s-a preferat o discretizare neomogenă, adică Δx, Δy şi Δz nu sunt constante. Aceasta însă prezintă şi un dezavantaj: pentru a determina cărei celule îi aparţine un punct arbitrar (x, y, z) – adică pentru a determina numerele întregi (i, j, k) de exemplu în vederea realizării unei interpolări – sunt necesare trei operaţii de căutare binară.

Deşi căutarea binară este eficientă, nu se compară cu simpla împărţire care ar fi fost necesară în cazul discretizării omogene.

Realizarea discretizării neomogene se face pe baza următorului algoritm pentru fiecare axă: un vector redimensionabil de valori în virgulă flotantă este încărcat iniţial cu valorile corespunzătoare ghidului de undă, antenei magnetronului, sarcinii şi cavităţii de rezonanţă, după care este sortat şi se intercalează valori astfel încât să se obţină o valorile maxime dorite pentru $x_{i+1}-x_i$, $y_{j+1}-y_j$ şi $z_{k+1}-z_k$.

4.6.2. Funcţionarea simulării

Programul a fost realizat cu o interfaţă minimală [38], fără posibilitatea de a modifica parametrii simulării prin interfaţă. Aceşti parametri pot fi modificaţi în sursa programului, care se poate recompila după aceea.

Limbajul de programare ales a fost C++, în mediul Microsoft Visual Studio. Alegerea a ţinut cont în primul rând de cunoştinţele de programare ale autorului, existând posibilitatea de a alege Java în loc de C++. Însă Java este mai puţin eficient în ce priveşte alocarea de tablouri mari de obiecte, consumând mult mai multă memorie. Avantajele Java, cum ar fi portabilitatea, au contat mai puţin în cazul de faţă.

Definiţia unei celule cu toate componentele câmpului electromagnetic şi proprietăţile de material este conţinută în clasa `Cell`. Proprietăţile de material sunt date sub forma unor indecşi întregi, factorul de pierderi, permitivitatea reală şi conductivitatea urmând a fi determinate prin apelul unor funcţii folosind indecşii respectivi. Metoda `val()` a clasei `Cell` are ca parametru un index, care determină natura valorii returnate: 0–2 sunt componentele câmpului electric, 3–5 componentele câmpului magnetic, 6 temperatura (nu este încă implementată), 7 modulul câmpului electric, 8 modulul câmpului magnetic, 9 modulul vectorului Poynting. Această metodă simplifică obţinerea prin interpolări a valorilor într-un punct arbitrar: dacă s-ar fi folosit 10 variabile sau funcţii distincte ar fi fost nevoie de 10 funcţii de interpolare, aşa se poate folosi una singură care pasează valoarea indexului funcţiei `val()`.

Clasa `Plane` este un container pentru toate celulele cu aceeaşi coordonată x. S-a folosit o clasă specială în acest sens deoarece nu toate secţiunile paralele cu yOz au aceleaşi dimensiuni, şi se poate aloca mai puţină memorie pentru valorile negative ale axei x (ghidul de undă), ceea ce duce la economie de memorie.

Clasa `Discret` conţine toate coordonatele discretizării şi un tablou unidimensional de clase `Plane`, deci toate valorile câmpului în domeniul studiat la un moment dat. Metoda `init()` realizează şi iniţializează discretizarea aşa cum s-a descris în paragraful 4.1. Metoda `defaults()` iniţializează geometria sistemului. Membrul static `__workerThreadLauncher()` şi metoda privată `workerThread()` pornesc un fir de execuţie de calcul. Numărul acestora în cazul sistemelor multiprocesor va fi numărul procesoarelor minus unu, un procesor fiind rezervat pentru trasarea şi actualizarea graficelor. Fiecare thread de calcul va procesa o parte din celule, dacă este cazul oprindu-se înainte de calculul E pentru a le aştepta pe celelalte, aceasta se realizează prin metoda privată `waitForMidThreads()` şi tabloul de evenimente de sincronizare `calcMidEvents`. În cazul în care un thread extern trebuie să aştepte pentru terminarea tuturor calculelor, de exemplu pentru a porni un grafic, aceasta se face prin metoda publică `waitForThreads()` şi tabloul de evenimente de sincronizare `calcStopEvents`.

Comparând timpul de execuţie pe acelaşi sistem quad-core (4 procesoare), faţă de varianta uniprocesor (făcând programul să considere că are un singur procesor) calculul multiprocesor a durat cu 60% mai puţin. De precizat că un calcul paralel implicând n procesoare nu este de exact n ori mai rapid, ceva mai puţin.

4.6.3. Rezultate şi interpretare

Toate simulările prezentate au fost făcute pentru frecvenţa de 2,45 GHz, la care corespunde o perioadă de aproximativ T = 0,41 ns şi o lungime de undă în vid de 122,4 mm. Pasul de timp a fost ales uniform de T/100. S-a constatat că după aproximativ 5000 de iteraţii (50 de perioade) câmpul nu se mai modifică semnificativ între perioade, deci se poate considera că s-a atins regimul staţionar. Sarcina folosită are în toate cazurile $\varepsilon' = 4$ şi $\varepsilon'' = 1$.

Imaginile reprezintă intensitatea câmpului electric în modul folosind un gradient de culoare prezentat în partea dreaptă. Culoarea pentru zero este albastru, care evoluează în verde şi roşu pentru maxim. Pentru a evidenţia detaliile, valoarea corespunzătoare verdelui nu este 0,5 din maxim, ci mai puţin. De fapt programul scalează intervalul $0 \div E_{max}$ în $0 \div 1$, ridică această valoare la puterea 0,25 după care calculează culoarea. Efectul este de concentrare a culorilor distincte spre valori mici, asemănător cu o scară logaritmică.

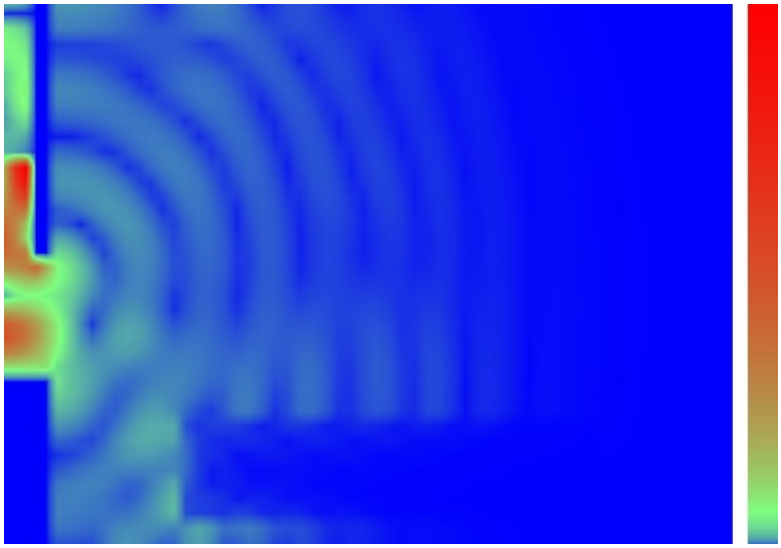


Fig. 4.11. Regim tranzitoriu, n = 500

În figura 4.11 se arată un regim tranzitoriu. De remarcat faptul că intensitatea relativ mare a câmpului electric din ghidul de undă produce chiar şi în acest caz un efect de "orbire", de aceea vom omite ghidul de undă în cele ce urmează. Sarcina este aparentă, caracterizată prin valori foarte mici ale câmpului în acest caz. Secţiunea prezentată este la mijlocul valorilor y, paralelă cu planul xOz. Dimensiunile sarcinii, aceleaşi şi în continuare, sunt 200×100×40 mm.

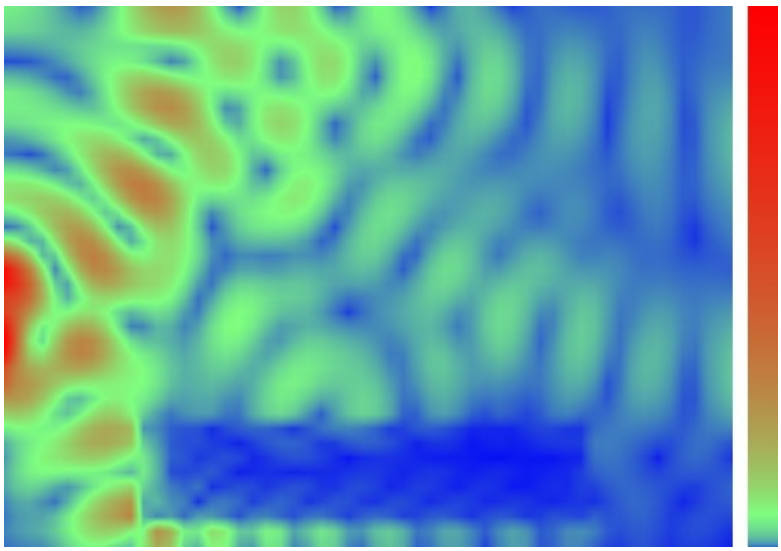


Fig. 4.12. Regim tranzitoriu, n = 1000

În fig. 4.12. este tot un regim tranzitoriu dar mai avansat şi fără ghidul de undă. Se remarcă formarea undelor staţionare şi refracţia în sarcină.

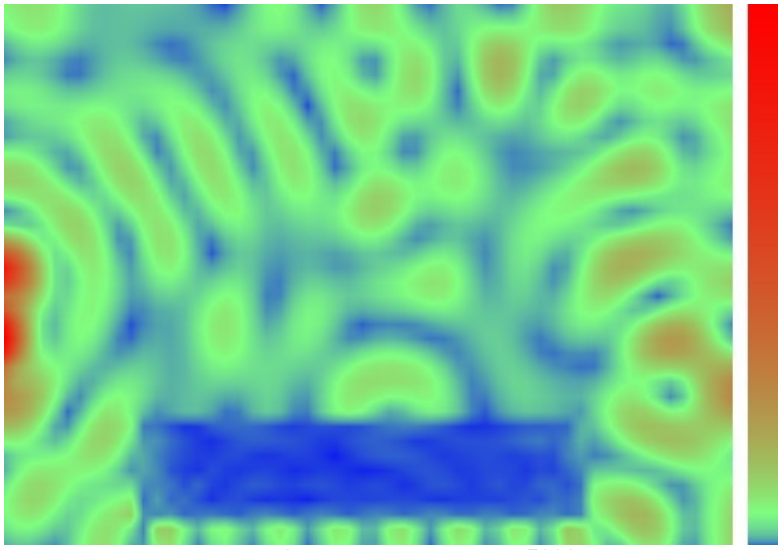


Fig. 4.13. Regim staţionar, n = 5000

Figurile 4.14 şi 4.15 prezintă un regim tranzitoriu şi regimul staţionar într-o secţiune paralelă cu planul xOy care taie sarcina la mijloc.

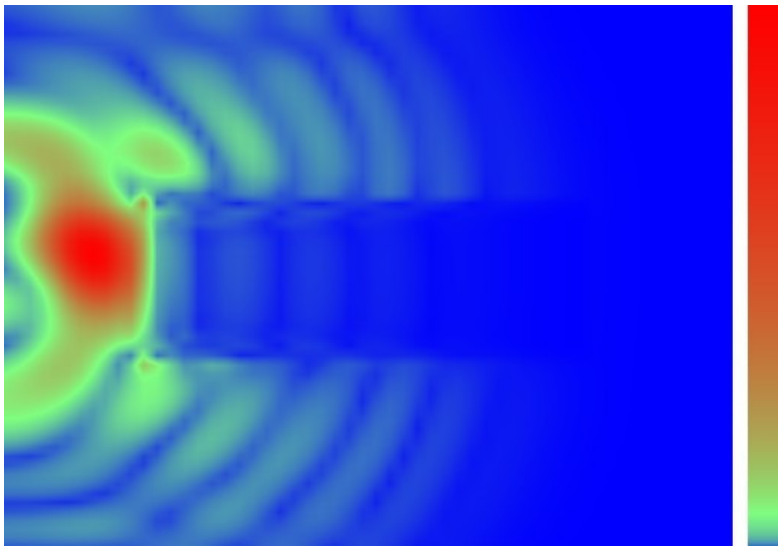


Fig. 4.14. Regim tranzitoriu, n = 500

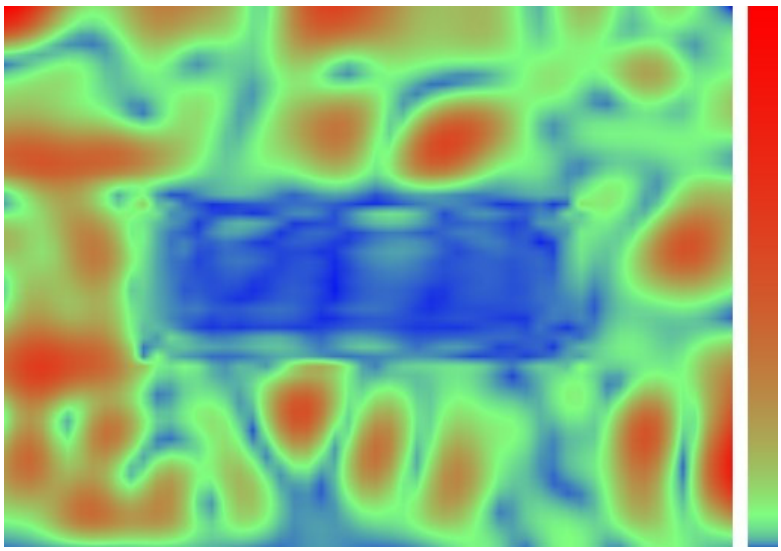


Fig. 4.15. Regim staţionar, n = 5000

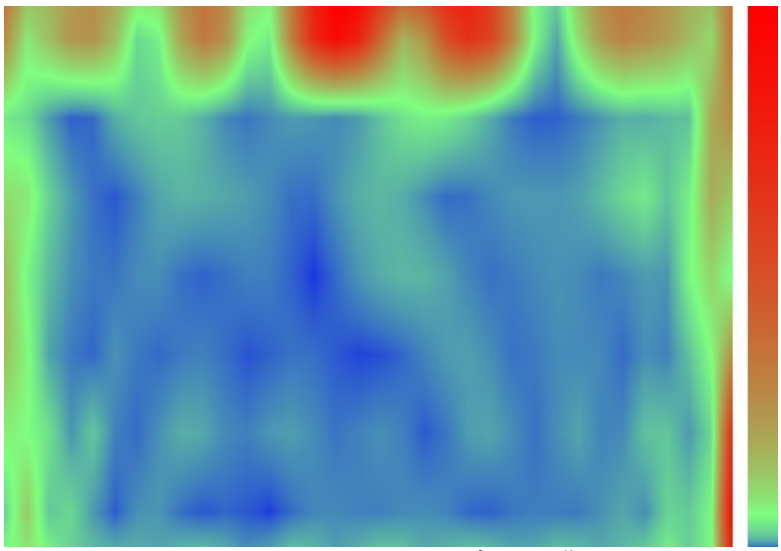


Fig. 4.16. Regim staţionar în sarcină

În fig. 4.16 se prezintă regimul staţionar obţinut în sarcină, planul xOz cu y la mijlocul sarcinii.

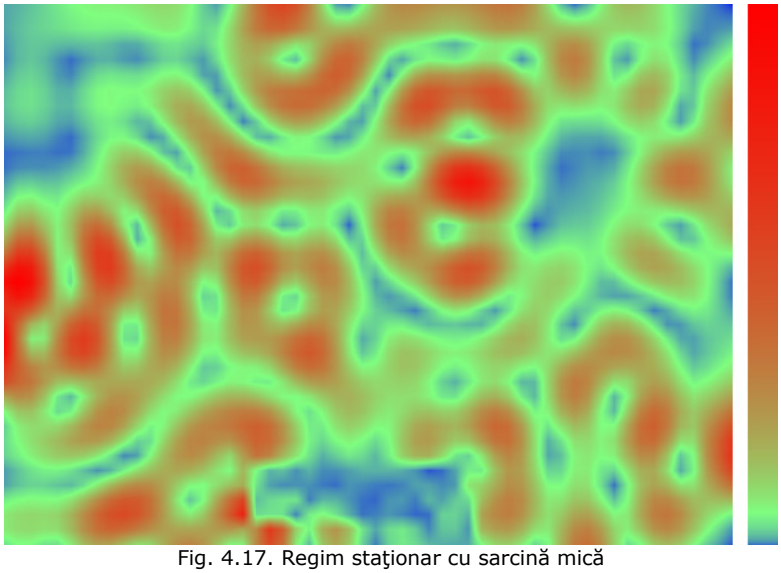


Fig. 4.17. Regim staţionar cu sarcină mică

Figura 4.17 arată un regim staţionar cu sarcină de dimensiuni înjumătăţite (volum 1/8). În comparaţie cu figura 4.13 – identică dar cu sarcina mare – se remarcă faptul că valorile mari ale câmpului sunt acum distribuite în cavitate. Aceasta deoarece sarcina mai mică duce la un factor de calitate mai mare al sistemului cavitate-sarcină iar undele se reflectă mai mult în pereţi.

Graficele următoare sunt realizate pentru o sarcină de 200×100×40 mm, cu ε'=4 şi ε"=1 la 0°C, densitate ρ = 830 Kg/m^3, căldură specifică 1670 J/KgK,

conductivitate termică 1,7 W/mK constantă. Dimensiunile geometrice sunt scalate în intervalul [0, 1], punctul (0, 0, 1) fiind cel mai apropiat de alimentarea cavităţii. Temperatura maximă în sarcina imobilă rezultată în urma simulării a fost de 265°C după 120 secunde la puterea de 900W.

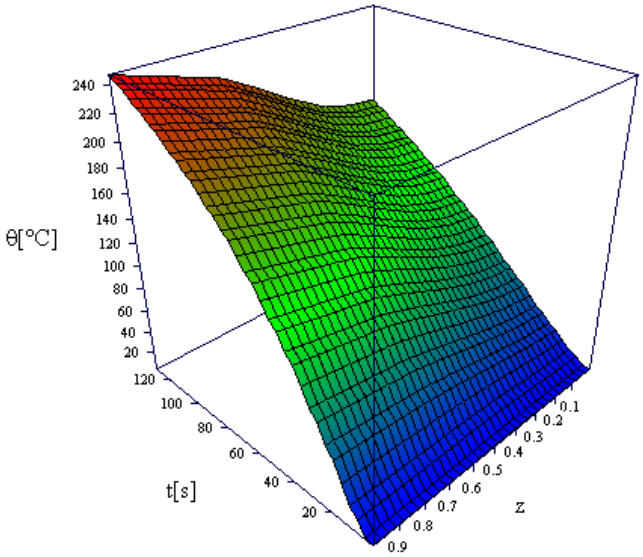


Fig. 4.18. Temperatura în funcţie de timp şi coordonata z pentru x=y=0

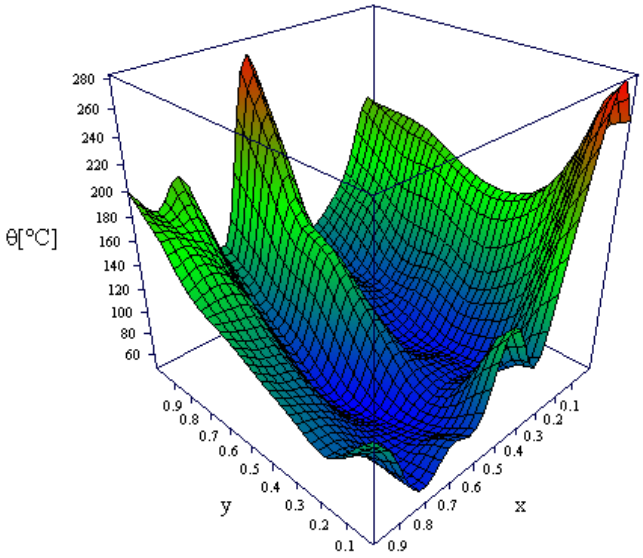


Fig. 4.19. Temperatura în funcţie de coordonatele x şi y pentru z=1 la momentul final

Temperatura mare obţinută se datorează materialului considerat, ce are pierderi neobişnuit de mari.

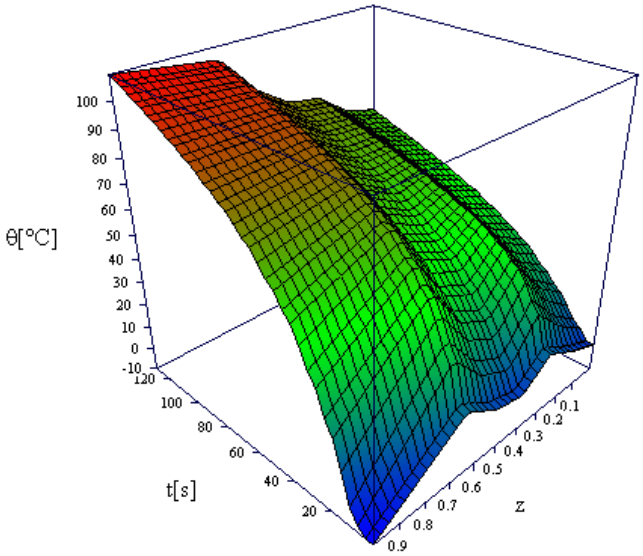


Fig. 4.20. Temperatura în funcţie de timp şi coordonata z pentru x=y=0,5

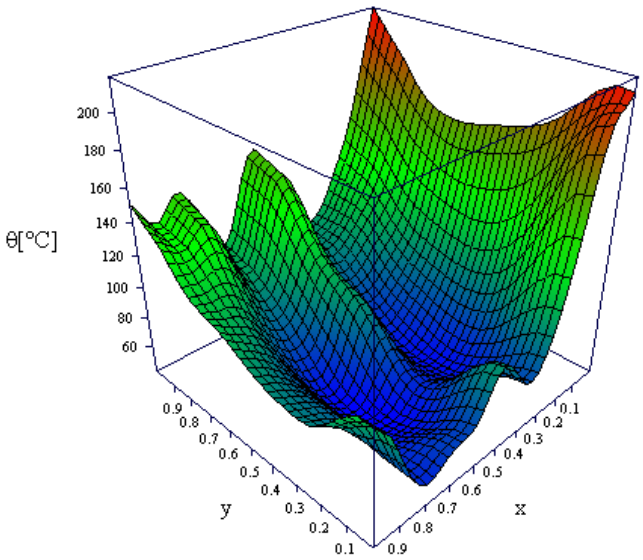


Fig. 4.21. Temperatura în funcţie de coordonatele x şi y
pentru z=0,5 la momentul final

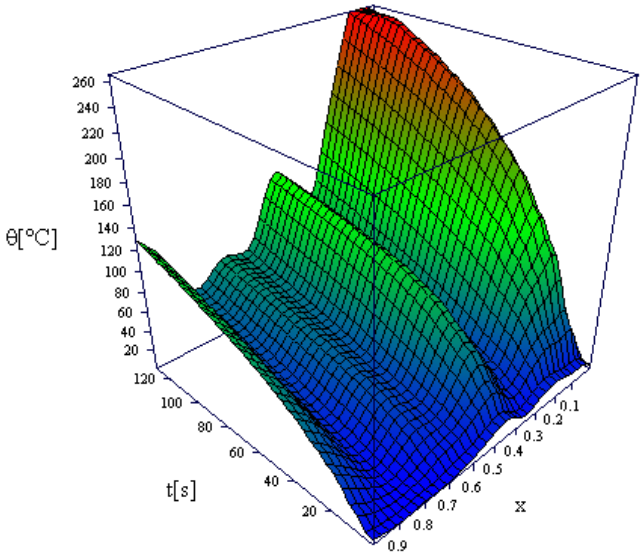


Fig. 4.22. Temperatura în funcţie de timp şi coordonata x pentru y=0 şi z=1

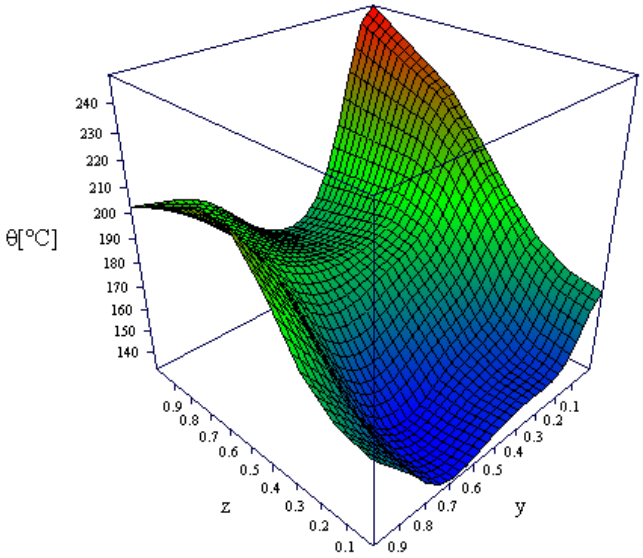


Fig. 4.23. Temperatura în funcţie de coordonatele z şi y
pentru x=0 la momentul final

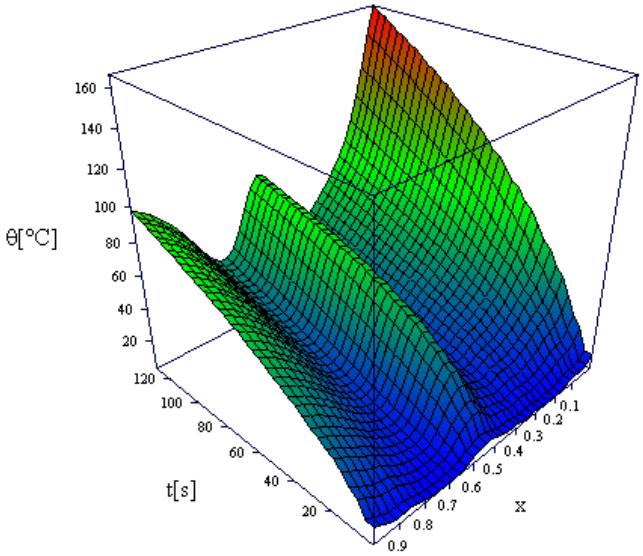


Fig. 4.24. Temperatura în funcţie de timp şi coordonata x pentru z=y=0,5

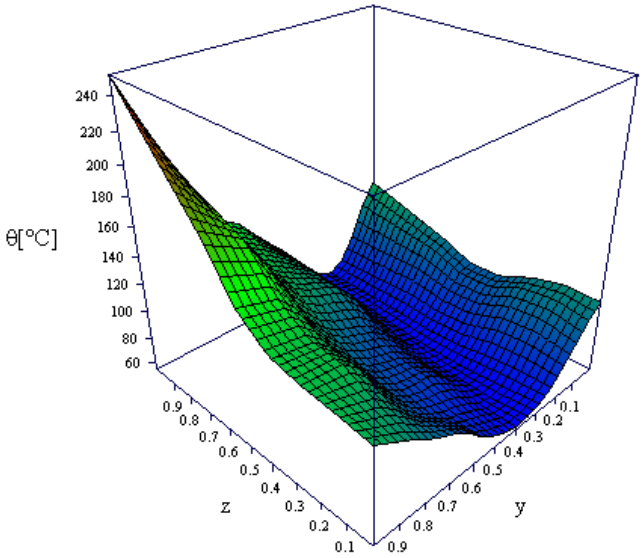


Fig. 4.25. Temperatura în funcţie de coordonatele z şi y
pentru x=0,5 la momentul final

4.7. Validarea experimentală a modelului

Măsurarea câmpului electromagnetic într-o cavitate rezonantă se face de obicei indirect, prin măsurarea încălzirii unui material cu caracteristici cunoscute. Pentru a valida modelul descris în acest capitol este nevoie de o sarcină cu caracteristici electrice şi termice cunoscute, expusă la microunde pe un timp determinat şi de măsurarea temperaturii cel puţin pe o suprafaţă. Distribuţia de temperaturi obţinută va fi comparată cu rezultatele unor simulări care respectă aceleaşi condiţii.

Cuptorul folosit în experimente este de tip Vortex MG8021TP-AP, un cuptor electrocasnic cu o putere de ieşire de 800W. Ca sarcini s-au folosit plăci de scândură de lemn de brad uscat cu grosimea de 20 mm şi următoarele caracteristici [7]:

- permitivitatea dielectrică $\varepsilon' = 2$
- tangenta unghiului de pierderi $\operatorname{tg} \delta = 0{,}1$
- densitatea $\rho = 380\ kg/m^3$
- căldura specifică $c = 2200\ J/kgK$
- conductivitatea termică $\lambda = 0{,}16\ W/mK$

În cazul lemnului uscat aceste caracteristici variază suficient de puţin în funcţie de temperatură pentru ca aceste variaţii să fie ignorate în simulare.

Pentru a măsura temperaturile s-a folosit o cameră de termoviziune (FLIR) de tip Fluke Ti-25 şi pachetul software asociat acesteia, SmartView 3.1. Măsurarea s-a făcut prin instantanee direct în cuptor cu uşa deschisă când a fost posibil sau extrăgând sarcina din cuptor şi măsurând imediat temperaturile.

Camera Fluke Ti-25 generează imagini mixte, cu componentă în infraroşu suprapusă peste o imagine în spectru vizibil pentru a facilita identificarea elementelor calde. Imaginile pot fi transferate pe un calculator sub forma unor fişiere binare cu extensia is2 care conţin toate informaţiile. Pachetul SmartView permite extragerea şi exportarea imaginilor în spectru vizibil, infraroşu sau suprapuse precum şi exportarea temperaturilor din imaginile în infraroşu sub forma unei matrice de 160x120 puncte sau mai puţine dacă s-a selectat o porţiune din imagine.

Fig. 4.26. Imagini în spectru vizibil şi infraroşu suprapus peste vizibil obţinute cu camera Fluke Ti-25

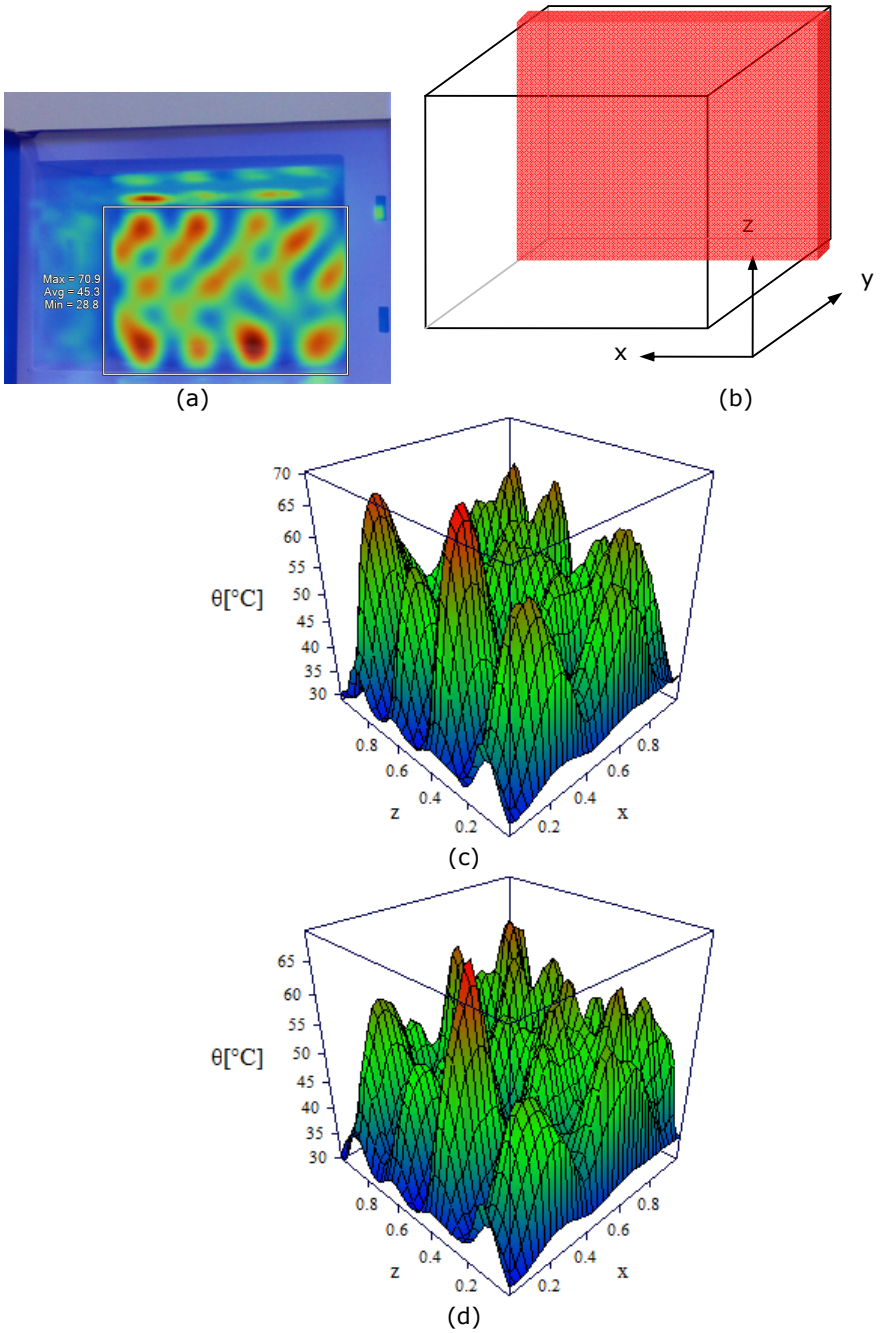
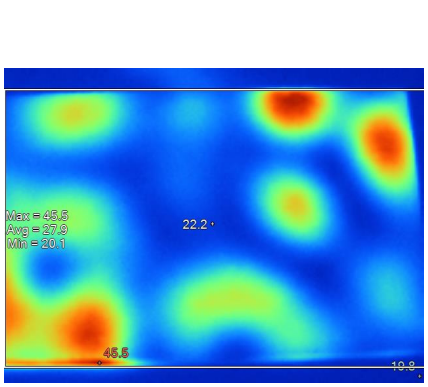
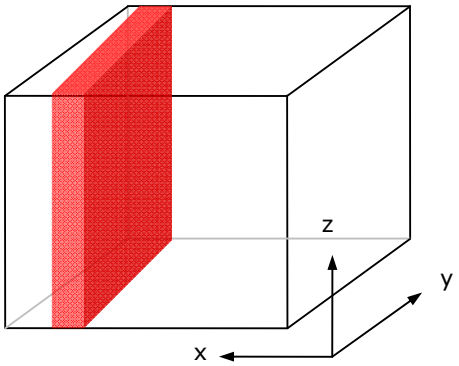


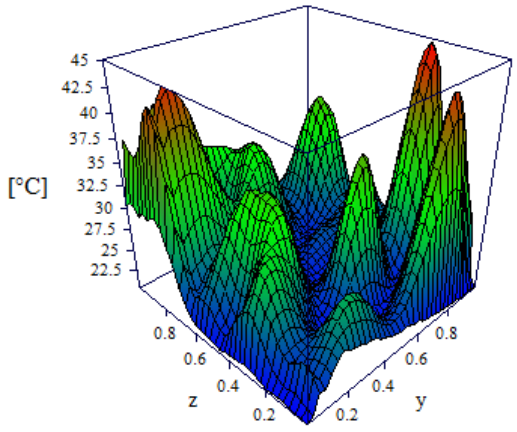
Fig. 4.27. Imagine detaliu termic cu selecţie (a), poziţia sarcinii în cuptor (b), distribuţia de temperaturi obţinută experimental la 60 secunde (c) şi prin simulare (d)



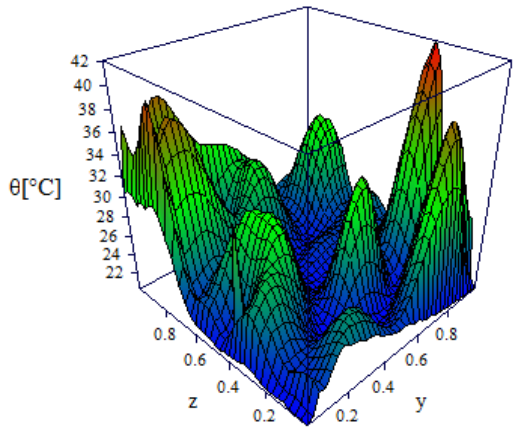
(a)



(b)

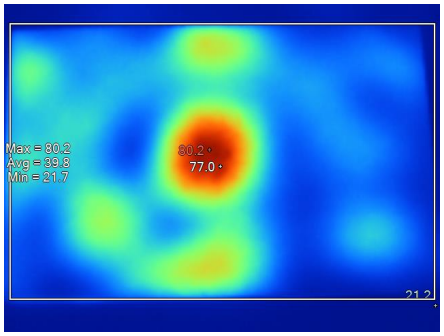


(c)

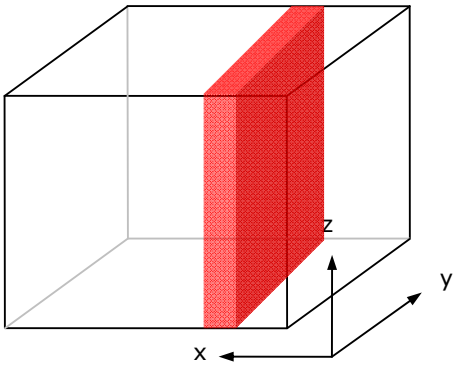


(d)

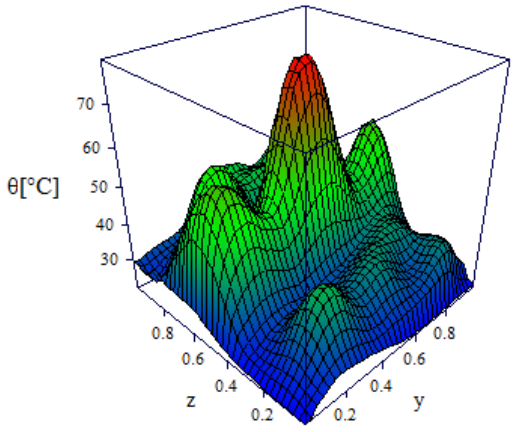
Fig. 4.28. Imagine detaliu termic cu selecţie (a), poziţia sarcinii în cuptor (b), distribuţia de temperaturi obţinută experimental la 30 secunde (c) şi prin simulare (d)



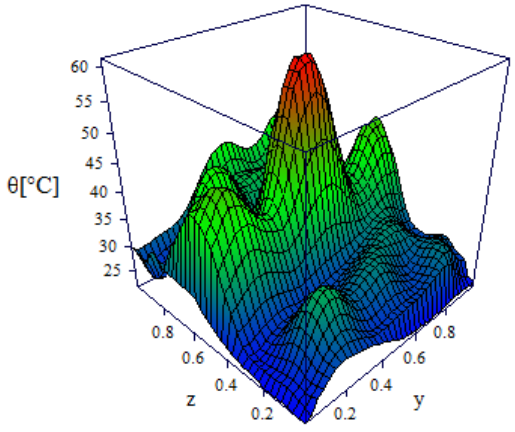
(a)



(b)



(c)



(d)

Fig. 4.29. Imagine detaliu termic cu selecţie (a), poziţia sarcinii în cuptor (b), distribuţia de temperaturi obţinută experimental la 60 secunde (c) şi prin simulare (d)

4.8. Concluzii

Metodele de simulare prezentate în acest capitol sunt relativ simple din punct de vedere matematic, mai puţin sofisticate decât cele în domeniu frecvenţă şi/sau pe bază de element finit însă mai uşor de implementat. Tehnici de programare avansate au asigurat timpi de execuţie rezonabili pentru simulări, de ordinul minutelor. Simularea câmpului electromagnetic în domeniu timp permite în plus determinarea regimurilor tranzitorii.

După cum s-a văzut în capitolul 2.3 există trei condiţii care impun limitarea puterii aplicate sarcinii: limitarea gradienţilor de temperatură, străpungerea dielectrică şi ambalarea termică. Un aparat de încălzire cu microunde trebuie deci să fie capabil să-şi ajusteze puterea. Pentru a realiza acest lucru se folosesc patru metode [41]:

a) alimentarea în impulsuri modulate al magnetronului
b) modificarea curentului anodic al magnetronului
c) modificarea câmpului magnetic intern al magnetronului
d) redirecţionarea unei părţi din energia microundelor în altă parte

Cea mai folosită metodă este de departe prima. Celelalte au un impact negativ asupra randamentului instalaţiei, iar (b) şi (c) impun folosirea de magnetroane de construcţie specială.

Cu ocazia simulărilor s-a constatat că sistemul cavitate rezonantă – sarcină ajunge la regim staţionar după câteva mii de alternanţe ale câmpului de excitaţie. În cazul sarcinilor normale acest lucru se întâmplă după cel mult 5000 alternanţe, ceea ce înseamnă aproximativ 2 μs. Pentru sarcini foarte mici timpul de stabilire va fi mai lung, dar rămâne în domeniul microsecundelor. Cum timpul de răspuns al magnetronului este şi el foarte scurt, se poate considera că aplicarea tensiunii anodice pe magnetronul cu filament cald duce practic instantaneu la regim staţionar.

Metoda (a) de ajustare a puterii nu poate fi folosită pentru a reduce intensitatea câmpului electric. Pentru aceasta ar fi nevoie de o frecvenţă a impulsurilor în domeniul megaherţilor, ceea ce este dificil din punct de vedere tehnic şi ar pune probleme serioase de compatibilitate electromagnetică.

5. Soluţii constructive pentru partea electrică a unui cuptor multimod

5.1. Caracteristici ale regimului periodic nesinusoidal

Tensiunea şi curentul care apar în regim periodic nesinusoidal pot fi descompuse în serii Fourier de forma:

$$u(t) = U_0 + \sqrt{2}\sum_{k\neq 0}^{\infty} U_k \sin\left(k\omega t + \alpha_k\right) \tag{5.1}$$

$$i(t) = I_0 + \sqrt{2}\sum_{k\neq 0}^{\infty} I_k \sin\left(k\omega t + \beta_k\right) \tag{5.2}$$

unde U_0, I_0 reprezintă componentele continue ale tensiunii, respectiv curentului, U_k, I_k reprezintă valorile efective ale armonicii de ordin k din tensiune şi curent, α_k este faza iniţială a armonicii k din tensiune, iar β_k este faza iniţială a armonicii k din curent.

Valorile efective ale tensiunii şi curentului sunt date de relaţiile [42]:

$$U = \sqrt{U_0^2 + \sum_{k=1}^{\infty} U_k^2} \tag{5.3}$$

$$I = \sqrt{I_0^2 + \sum_{k=1}^{\infty} I_k^2} \tag{5.4}$$

Separând componenta fundamentală de celelalte componente armonice, se obţine:

$$U^2 = U_1^2 + U_d^2 \tag{5.5}$$

$$I^2 = I_1^2 + I_d^2 \tag{5.6}$$

unde U_d este reziduul deformant al tensiunii, iar I_d este reziduul deformant al curentului:

$$U_d^2 = \sum_{k\neq 1} U_k^2 \tag{5.7}$$

$$I_d^2 = \sum_{k\neq 1} I_k^2 \tag{5.8}$$

Pentru a evalua deformarea unui semnal sunt utilizate următoarele mărimi:

• conţinutul de armonică de rang k:

$$s_k = \frac{C_k}{C_1}\cdot 100 \ [\%] \tag{5.9}$$

exprimat în raport cu componenta fundamentală C_1 (valoare efectivă), C_k fiind componenta de rang k (valoare efectivă).

• distorsiunea armonică totală:

$$THD = \frac{\sqrt{\sum_{k=2}^{40} C_k^2}}{C_1} \tag{5.10}$$

un coeficient global, definit pentru primele 40 de armonici.

• distorsiunea armonică parţial ponderată:

$$THD_p = \frac{\sqrt{\sum_{k=2}^{40} k \cdot C_k^2}}{C_1} \tag{5.11}$$

introdus pentru a se asigura că odată cu creşterea rangului k, armonicile descresc.

În regim periodic nesinusoidal se pot defini următoarele puteri corespunzătoare unui receptor monofazat [42]:

• puterea activă P :

$$P = U_0 \cdot I_0 + \sum_{k=1}^{\infty} U_k \cdot I_k \cdot \cos\varphi_k \tag{5.12}$$

unde φ_k reprezintă defazajul dintre armonica k a tensiunii şi armonica k a curentului:

$$\varphi_k = \alpha_k - \beta_k \tag{5.13}$$

• puterea reactivă Q :

$$Q = \sum_{k=1}^{\infty} U_k \cdot I_k \cdot \sin\varphi_k \tag{5.14}$$

• puterea aparentă S :

$$S = U \cdot I \tag{5.15}$$

U şi I fiind valorile efective ale tensiunii şi curentului, date de relaţiile (5.3) şi (5.4).

Ţinând cont de relaţiile (5.5) şi (5.6), puterea aparentă se poate exprima astfel:

$$S^2 = S_1^2 + D^2 \tag{5.16}$$

unde $S_1^2 = P_1^2 + Q_1^2$ este puterea aparentă fundamentală, cu componentele sale:

$$P_1 = U_1 \cdot I_1 \cdot \cos\varphi_1 \tag{5.17}$$

$$Q_1 = U_1 \cdot I_1 \cdot \sin\varphi_1 \tag{5.18}$$

U_1, I_1 fiind valorile efective ale fundamentalei tensiunii, respectiv curentului electric, iar φ_1, defazajul dintre aceste mărimi (exprimate în funcţie de timp).

D reprezintă puterea deformantă care se poate exprima cu ajutorul a trei componente:

$$D^2 = S^2 - S_1^2 = \left(U_1 \cdot I_d\right)^2 + \left(U_d \cdot I_1\right)^2 + \left(U_d \cdot I_d\right)^2 = D_I^2 + D_U^2 + D_{UI}^2 \tag{5.19}$$

U_d, I_d fiind date de relaţiile (5.7) şi (5.8).

D_I este puterea deformantă datorată curentului (de obicei reprezintă termenul dominant), D_U este puterea deformantă datorată tensiunii, iar D_{UI} este puterea aparentă armonică.

Se poate scrie:

$$D_{UI}^2 = \left(U_d \cdot I_d\right)^2 = P_H^2 + Q_H^2 \tag{5.20}$$

unde P_H reprezintă puterea activă armonică totală, iar Q_H reprezintă puterea reactivă armonică totală.

$$P_H = \sum_{k=2}^{\infty} U_k \cdot I_k \cdot \cos\varphi_k \tag{5.21}$$

$$Q_H = \sum_{k=2}^{\infty} U_k \cdot I_k \cdot \sin\varphi_k \tag{5.22}$$

U_k şi I_k fiind valorile efective ale armonicilor de ordin k din tensiune şi curent, iar φ_k, defazajul dintre aceste armonici.

În cazul unui consumator monofazat neliniar, factorul de putere în regim deformant se defineşte astfel:

$$k_p = \frac{P}{S_1} = \frac{P_1 + \sum_{k\neq 1} U_k \cdot I_k \cdot \cos\varphi_k}{U_1 \cdot I_1} = \cos\varphi_1 + \sum_{k\neq 1} s_{kU} \cdot s_{kI} \cdot \cos\varphi_k \tag{5.23}$$

unde $s_{kU} = \frac{U_k}{U_1}$ şi $s_{kI} = \frac{I_k}{I_1}$ reprezintă ponderile armonicilor nefundamentale de tensiune şi curent faţă de armonicile lor fundamentale.

În aceste condiţii, se poate scrie:

$$k_p = k_p^1 + k_p^d \tag{5.24}$$

k_p^1 fiind factorul de putere fundamental, iar k_p^d factorul de putere deformant.

Puterea aparentă armonică normalizată se defineşte ca:

$$k_{pdh} = \frac{D_{UI}}{S_1} = \frac{U_d \cdot I_d}{U_1 \cdot I_1} = UTHD \cdot ITHD \tag{5.25}$$

unde UTHD şi ITHD reprezintă distorsiunea armonică totală a tensiunii şi curentului.

Se poate defini şi puterea deformantă normalizată:

$$\begin{aligned} k_{pd}^2 &= \left(\frac{D}{S_1}\right)^2 = \left(\frac{I_d}{I_1}\right)^2 + \left(\frac{U_d}{U_1}\right)^2 + \left(\frac{U_d \cdot I_d}{U_1 \cdot I_1}\right)^2 \\ &= ITHD^2 + UTHD^2 + \left(ITHD \cdot UTHD\right)^2 \end{aligned} \tag{5.26}$$

5.1.1. Integrarea numerică

Pentru a obţine seriile Fourier (5.1) şi (5.2) în cazul unor semnale obţinute ca serii de eşantioane este nevoie de integrare numerică. Coeficienţii Fourier pentru o funcţie y(t) pot fi calculaţi astfel:

$$A_0 = \frac{1}{T}\int_0^T y(t)dt \tag{5.27}$$

$$A_k = \frac{2}{T}\int_0^T y(t)\cos(k\omega t)dt, \quad k > 0 \tag{5.28}$$

$$B_k = \frac{2}{T}\int_0^T y(t)\sin(k\omega t)dt, \quad k > 0 \tag{5.29}$$

unde T este durata fundamentalei sau un multiplu al acesteia, iar ω este pulsaţia fundamentalei.

Funcţia originală poate fi recompusă astfel:

$$y(t) = A_0 + \sum_{k\geq 1}\left(A_k \cos(k\omega t) + B_k \sin(k\omega t)\right) \tag{5.30}$$

În cazul nostru este preferabil să exprimăm componentele armonice ca amplitudine şi fază:

$$A_0 = C_0, \quad A_k = C_k \sin\varphi_k, \quad B_k = C_k \cos\varphi_k \tag{5.31}$$

deci:

$$y(t) = C_0 + \sum_{k\geq 1} C_k\left(\sin\varphi_k \cos(k\omega t) + \cos\varphi_k \sin(k\omega t)\right) \tag{5.32}$$

$$y(t) = C_0 + \sum_{k\geq 1} C_k \sin(k\omega t + \varphi_k) \tag{5.33}$$

unde:

$$C_k = \sqrt{A_k^2 + B_k^2}, \quad \varphi_k = arctg\frac{A_k}{B_k} \tag{5.34}$$

Este de remarcat cazul particular $B_k=0$ în calculul φ_k. Multe limbaje de programare pun la dispoziţie o funcţie specială, numită de obicei "atan2", pentru a calcula arctangenta dintr-un raport inclusiv în cazul în care numitorul este zero pentru a evita erorile. Coeficienţii C_k din relaţiile de mai sus reprezintă amplitudini, nu valori efective.

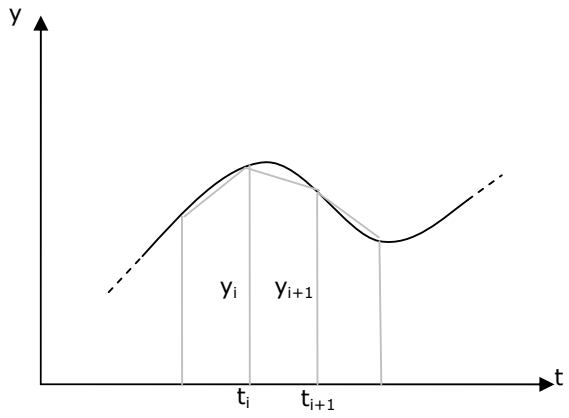


Fig. 5.1. Integrarea prin metoda trapezelor

În cazul în care funcţia y(t) este dată printr-o serie de valori distincte uniform distribuite în timp, ceea ce este cazul pentru semnale eşantionate, se poate folosi metoda trapezelor pentru integrare numerică cu condiţia – esenţială oricum

pentru semnale eşantionate – ca frecvenţa de eşantionare să fie mult mai mare decât frecvenţa fundamentalei. Presupunem un număr n de eşantioane, notate y_0 ... y_{n-1}, frecvenţa de eşantionare fiind f_s.

Aria trapezului din figura 5.1 este:

$$s_i = \frac{y_i + y_{i+1}}{2}\left(t_{i+1} - t_i\right) = \frac{y_i + y_{i+1}}{2 \cdot f_s} \tag{5.35}$$

Suma ariilor trapezelor, deci aria subgraficului adică valoarea integralei este:

$$S = \sum_{i=0}^{n-2} \frac{y_i + y_{i+1}}{2 \cdot f_s} = \frac{1}{f_s}\left(\frac{y_0 + y_{n-1}}{2} + \sum_{i=1}^{n-2} y_i\right) \tag{5.36}$$

5.1.2. Limitări impuse prin reglementări europene

Standardul european care reglementează valorile maxime ale armonicilor de curent, aplicabil în cazul receptoarelor la tensiunea reţelei (220V) şi curent până la 16A, atât monofazate cât şi trifazate, este **EN 61000–3–2:2006+A1+A2** [43]. Acest standard defineşte patru clase de consumatori, codificate de la A la D.

Clasa A include consumatorii trifazaţi precum şi orice alt consumator care nu intră în altă clasă. Clasa B include uneltele electrice portabile şi echipamentele de sudură cu arc electric neprofesionale. Clasa C se referă la instalaţii de iluminat, iar clasa D la echipamente electronice de cel mult 600W cum ar fi computere sau televizoare.

Pentru clasa A, unde intră cuptoarele cu microunde, limitele armonicilor de curent sunt date ca valori absolute maxime ale curenţilor efectivi, date în amperi până la armonica 40 inclusiv:

Tab. 5.1. Curenţii armonici maximi conform EN 61000–3–2:2006+A1+A2

Arm.	I [A]	Arm.	I [A]	Arm.	I [A]	Arm.	I [A]
		11	0,33	21	0,10	31	0,07
2	1,08	12	0,15	22	0,08	32	0,05
3	2,30	13	0,21	23	0,09	33	0,06
4	0,43	14	0,13	24	0,07	34	0,05
5	1,14	15	0,15	25	0,09	35	0,06
6	0,30	16	0,11	26	0,07	36	0,05
7	0,77	17	0,13	27	0,08	37	0,06
8	0,23	18	0,10	28	0,06	38	0,04
9	0,40	19	0,11	29	0,07	39	0,05
10	0,18	20	0,09	30	0,06	40	0,04

5.2. Circuitul electric al cuptoarelor comune

Marea majoritate a cuptoarelor cu microunde, electrocasnice sau de laborator, utilizează magnetroane cu undă continuă fabricate în serii mari pe linii de producţie complet automatizate [41]. Acestea au timp de viaţă destul de scurt (sute de ore) dar sunt ieftine şi au puteri de până la 1kW.

Nu doar magnetroanele sunt standard şi produse în serii mari, ci şi componentele circuitelor de alimentare ale acestora. Cuptorul tipic conţine un transformator alimentat la tensiunea reţelei în primar, cu două înfăşurări în secundar: una de 3,3V la 10A pentru filamentul catodului magnetronului, cealaltă de înaltă tensiune (2200 V) pentru tensiunea catodică. Un condensator şi o diodă de înaltă tensiune, împreună cu caracteristica de diodă a magnetronului, formează un redresor cu dublare de tensiune (fig. 5.2).

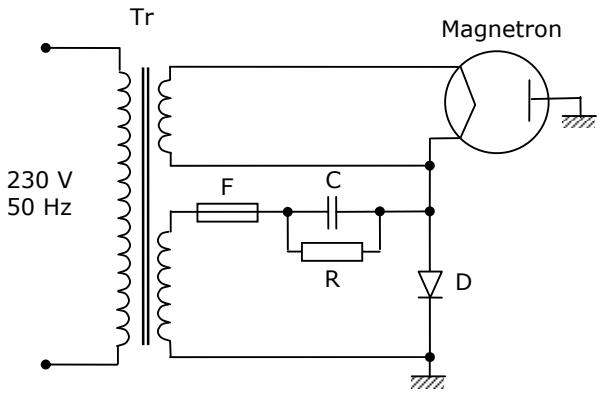


Fig. 5.2. Circuitul tipic de alimentare a magnetronului

Anodul magnetronului este cuplat la masă prin carcasa acestuia. Componenta F este o siguranţă fuzibilă de înaltă tensiune. R şi C nu sunt discrete; condensatorul este construit deliberat cu pierderi pentru a se descărca atunci când circuitul nu este alimentat. Valorile uzuale pentru C şi R sunt 1 μF şi 10 MΩ, de unde rezultă o constantă de timp de 0,1 secunde. Dioda D este de înaltă tensiune, cu tensiune inversă maximă de 12 kV şi tensiune directă în conducţie în jur de 10V.

Puterea este ajustată prin alimentarea intermitentă a întregului circuit de către sistemul de comandă şi control al cuptorului. Deoarece acelaşi circuit alimentează şi filamentul magnetronului ciclul de funcţionare trebuie să fie lung, de ordinul zecilor de secunde.

5.2.1. Caracteristici electrice

Circuitul de înaltă tensiune al magnetronului este de departe cel mai mare consumator de energie dintr-un cuptor. Filamentul, motoarele de antrenare pentru ventilatorul de răcire şi (dacă e cazul) agitator de câmp, precum şi circuitele de control au consumuri relativ neglijabile. Dacă alimentarea magnetronului este realizată similar la multe cuptoare, atunci caracteristicile curentului absorbit de acestea ar trebui să fie asemănătoare.

Pentru a verifica acest lucru s-au făcut o serie de măsurători folosind un cleşte ampermetric multifuncţional de tip **Wavetek Meterman AC68 AC/DC TRUERMS** pe mai multe cuptoare. Acest instrument este capabil să exporte caracteristicile măsurate pe un calculator sub formă de imagini. În următoarele imagini sunt: formele de undă, spectrul tensiunilor, puterile şi spectrul curenţilor pentru patru cuptoare comerciale. Se observă caracteristicile asemănătoare, ceea ce susţine ipoteza unor structuri similare.

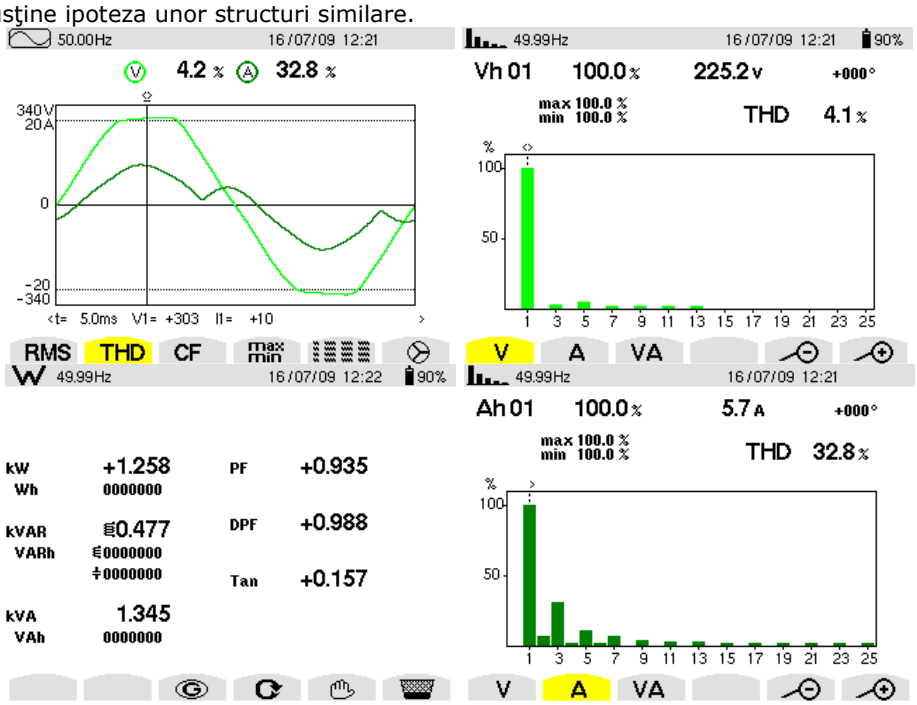
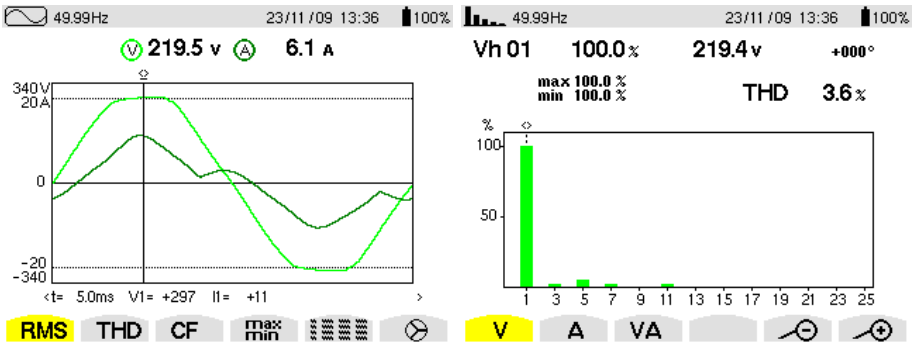


Fig. 5.3. Măsurători pentru cuptorul Vortex MG8021TP–AP



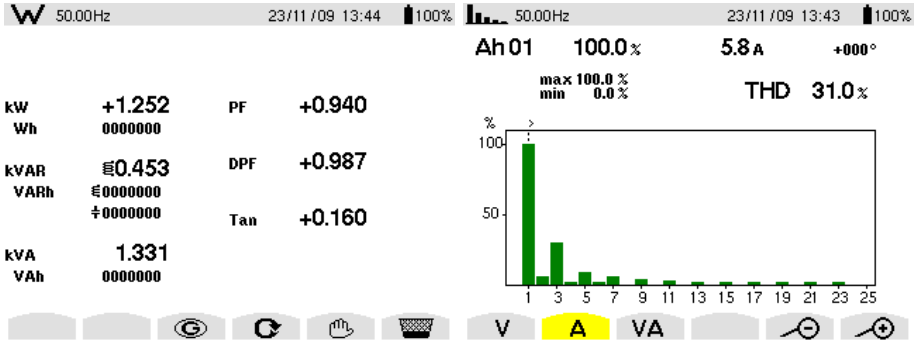


Fig. 5.4. Măsurători pentru cuptorul Samsung G2736N

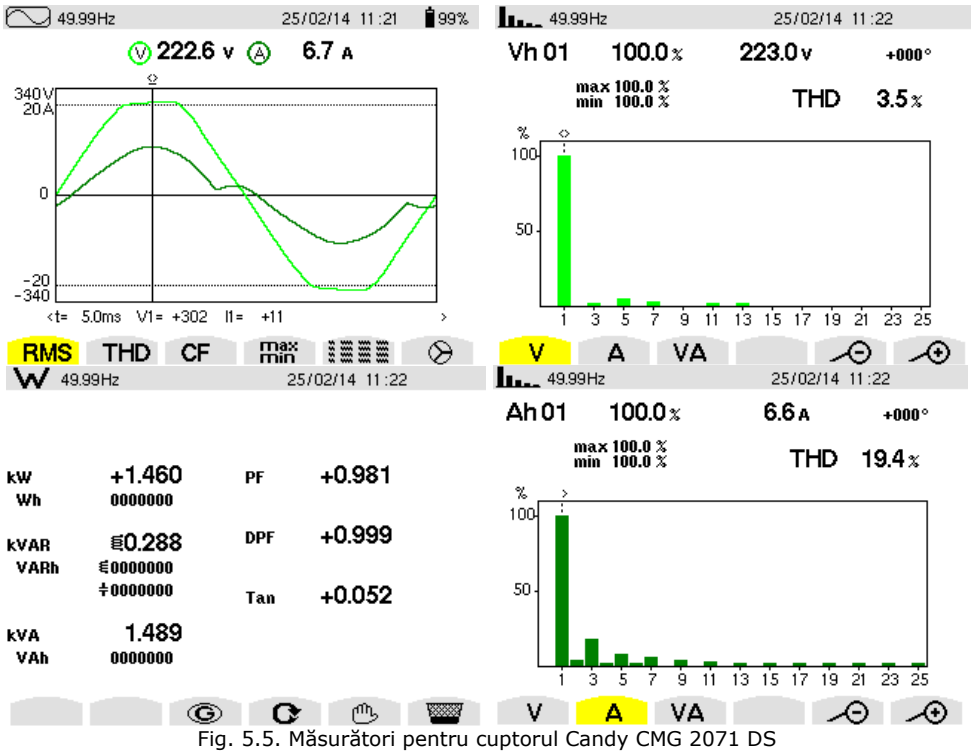


Fig. 5.5. Măsurători pentru cuptorul Candy CMG 2071 DS

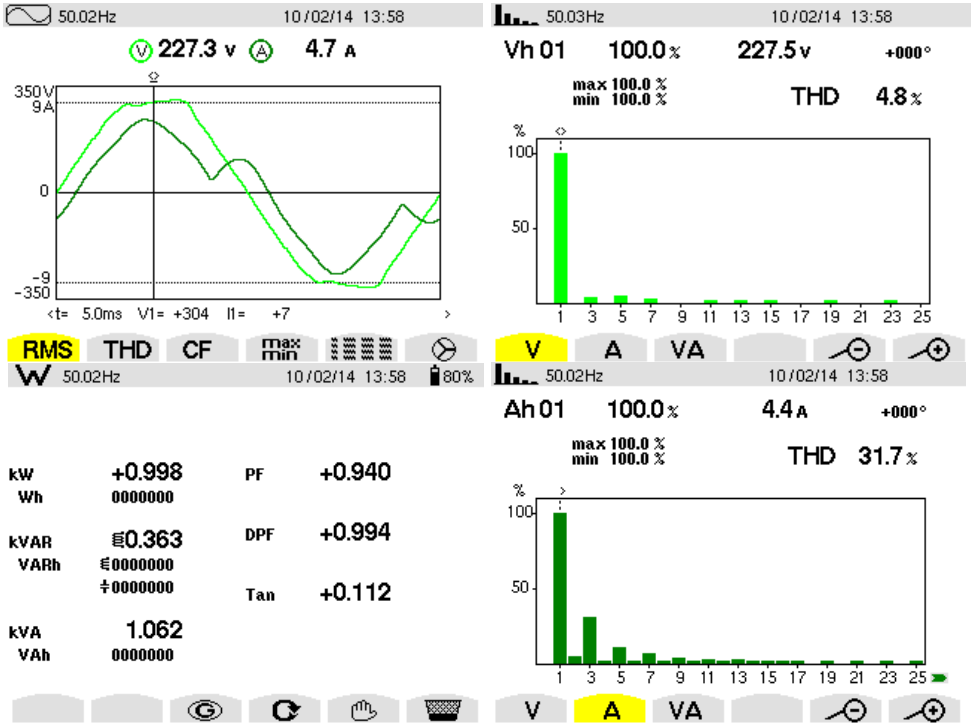
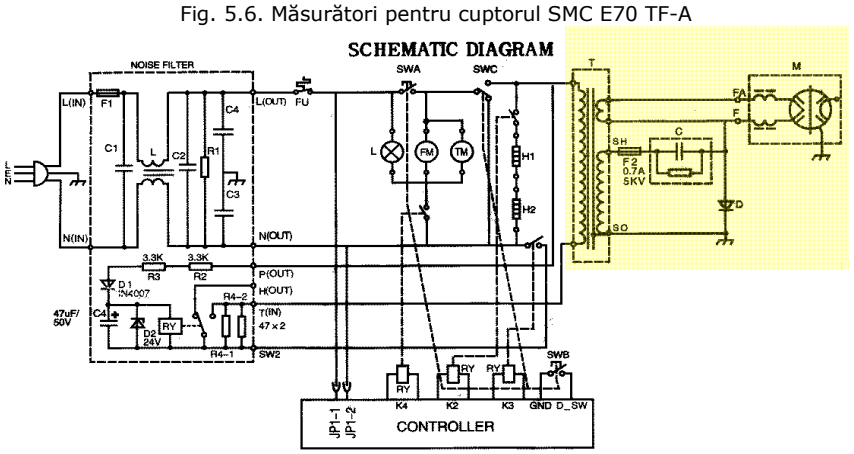


Fig. 5.6. Măsurători pentru cuptorul SMC E70 TF-A



SWA: PRIMARY INTERLOCK SWITCH
SWB: SECONDARY INTERLOCK SWITCH
SWC: THE MONITOR INTERLOCK SWITCH

L: LAMP
T: TIMER MOTOR
FM: FAN MOTOR
TM: TURNTABLE MOTOR
H1,H2:HEATER YUBE

NOTE:OVEN DOOR IN OPEN STATE
*CIRCUIT SUBJECT TO CHANGE WITHOUT NOTICE

Fig. 5.7. Schema cuptorului Vortex MG8021TP–AP

Unul dintre aceste cuptoare, modelul Vortex MG8021TP–AP, a fost analizat mai în amănunt. Schema originală a acestuia, utilizată pentru reparaţii, este prezentată în fig. 5.7, cu partea de alimentare a magnetronului evidenţiată.

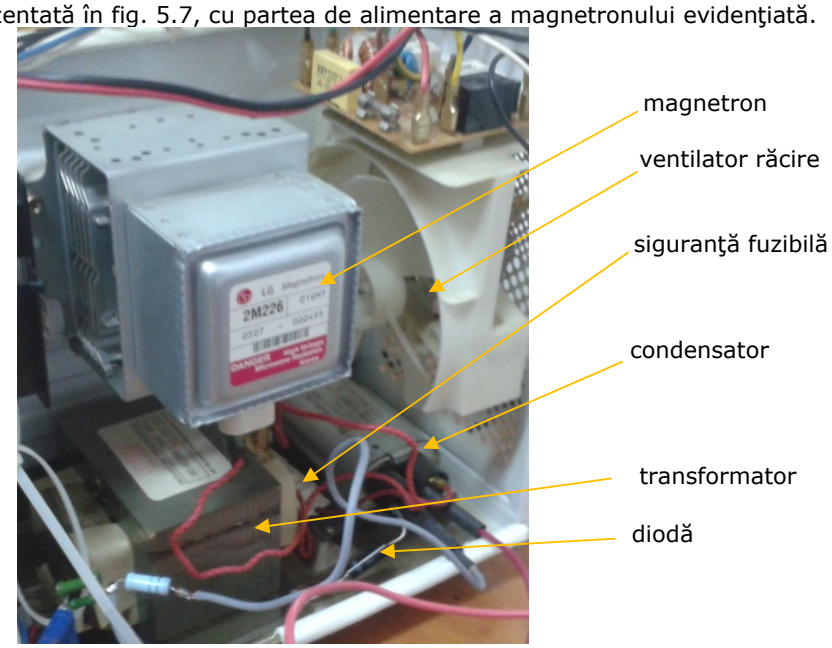


Fig. 5.8. Interiorul cuptorului Vortex MG8021TP–AP

Pentru a colecta cât mai multe informaţii despre funcţionarea cuptorului s-a folosit o placă de achiziţie National Instruments NI PCI-6221, programul LabView, un bloc de adaptare conţinând cuploare optice şi transformatoare de curent (câte trei) şi două traductoare Hall (fig. 5.9).

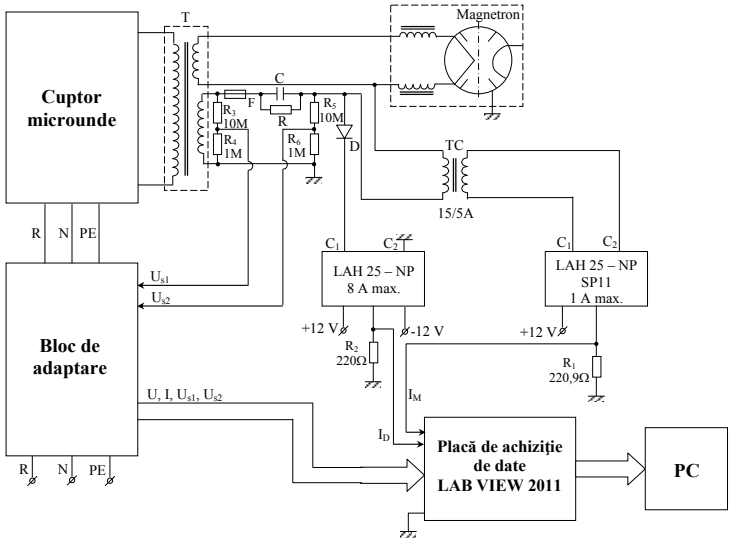


Fig. 5.9. Schemă de măsură

Mărimile măsurate sunt: tensiunea şi curentul în reţea, tensiunea în secundarul transformatorului, curentul prin diodă, tensiunea şi curentul prin magnetron. Rolul principal al blocului de adaptare este de a proteja placa de achiziţie prin izolare galvanică. Placa de achiziţie poate eşantiona simultan opt canale diferenţiale cu tensiuni maxime de ±10V la 25 kHz. Frecvenţa maximă de achiziţie este 200 kHz, dar pe un singur canal.

S-au conceput şi folosit două programe LabView: unul pentru monitorizare şi afişare continuă şi celălalt pentru achiziţie timp de 20 secunde cu datele colectate în fişiere text pentru prelucrare ulterioară.

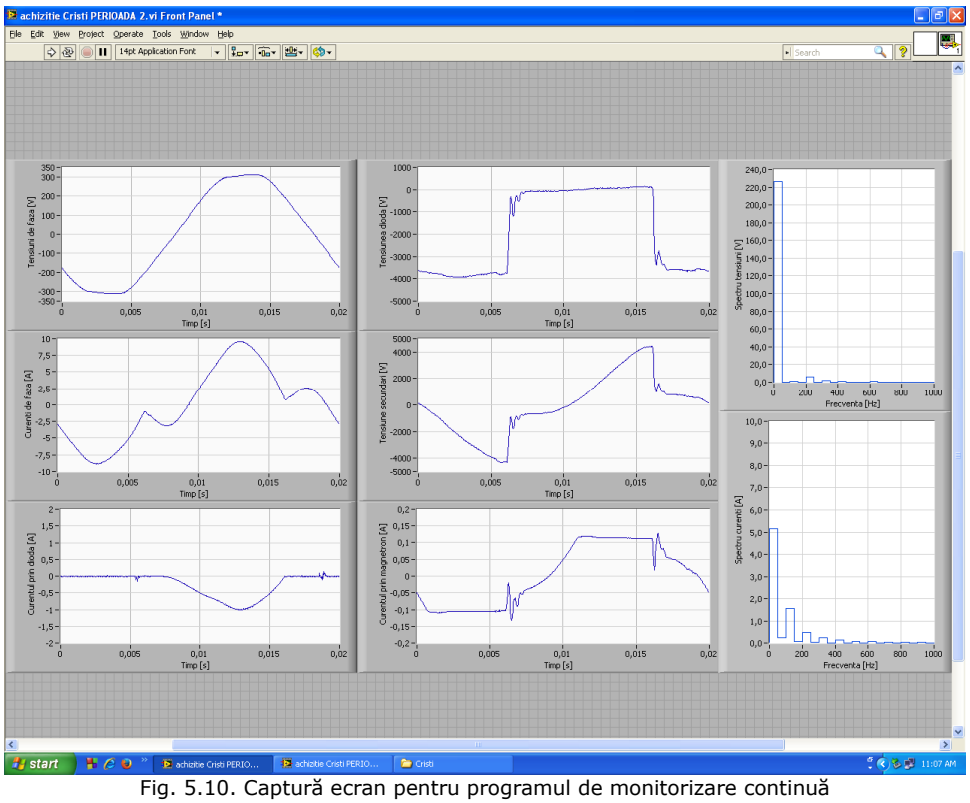


Fig. 5.10. Captură ecran pentru programul de monitorizare continuă

Pentru a determina rolul jucat de circuitul notat "*noise filter*" în figura 5.7 s-a conceput o altă schemă de măsură şi program aferent:

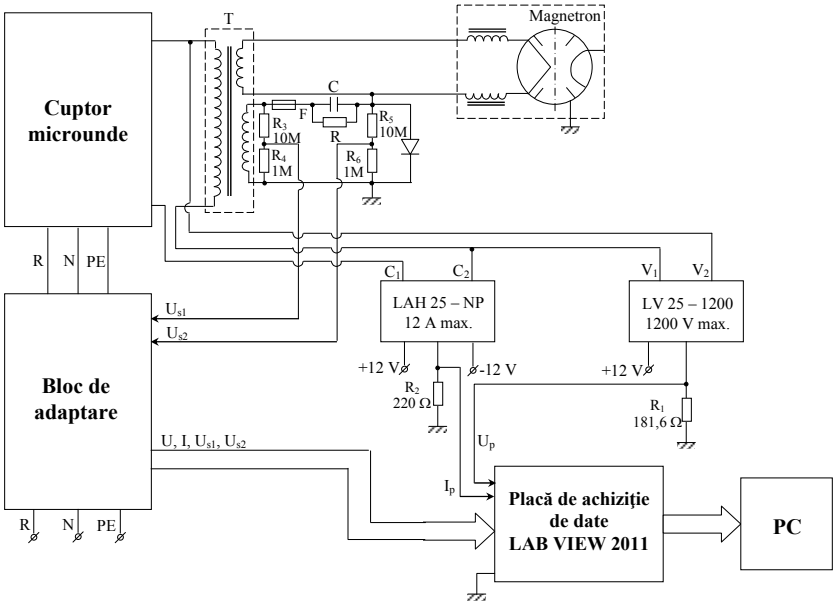


Fig. 5.11. Schemă de măsură pentru tensiunea şi curentul în primarul transformatorului

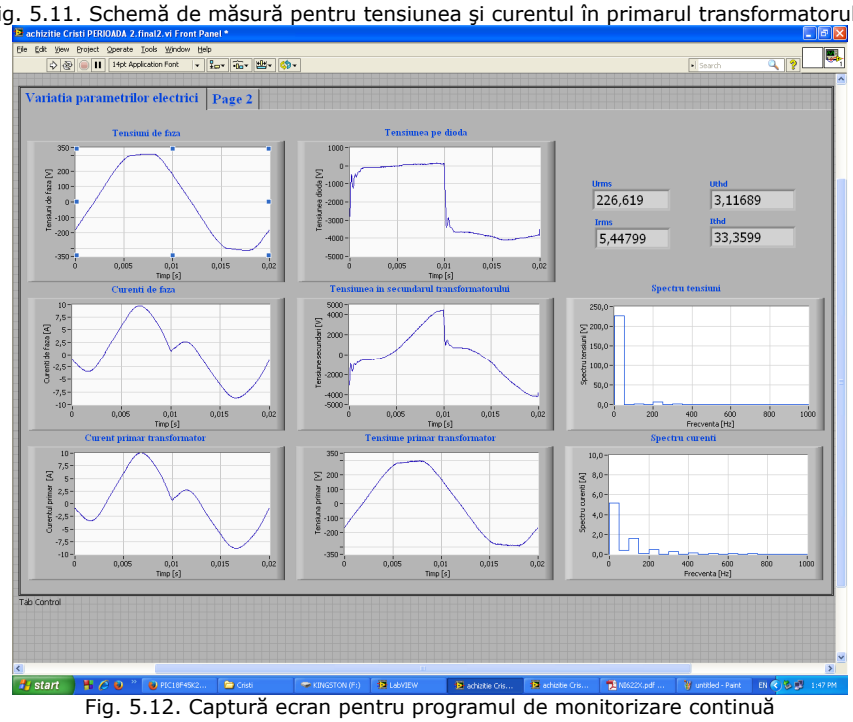


Fig. 5.12. Captură ecran pentru programul de monitorizare continuă

Se observă că nu există diferenţe semnificative între curentul din reţea şi cel din primarul transformatorului. Filtrul din schema 5.7 acţionează cel mai probabil pe frecvenţe înalte.

Pentru a prelucra datele salvate în fişiere text, care conţin valori reduse la ±10V achiziţionate la 25 kHz timp de 20 secunde, a fost scris un program în limbajul Java numit **mwel** (numele vine de la *microwave* şi *electrical*). Limbajul Java a fost preferat faţă de C++ în acest caz, fiindcă spre deosebire de programele din capitolul 4 aici nu avem de a face cu masive de memorie sub forma unor matrice tridimensionale imense care ar pune probleme maşinii virtuale Java. În schimb Java oferă biblioteci de funcţii şi facilităţi superioare.

Programul mwel are o interfaţă grafică care permite selectarea fişierului de intrare, afişează date şi grafice. La prima deschidere a unui fişier text de intrare cu extensia `txt`, acesta este interpretat şi salvat în format binar cu extensia `float`. De asemenea, se calculează frecvenţa exactă a fundamentalei şi se descompun în serii Fourier pe domenii scurte (100 ms) toate cele 6 semnale, aceste date fiind salvate în format binar în fişiere cu extensia `spec` (de la spectru). Aceste fişiere auxiliare vor fi folosite în cazul unor încărcări ulterioare, deoarece calculele făcute pentru generarea lor sunt intensive şi astfel se evită repetarea lor.

Frecvenţa exactă a fundamentalei este necesar să fie calculată pentru precizie, deoarece frecvenţa reţelei nu este exact 50 Hz. Acest lucru se face printr-un algoritm de tip *goal seeking* care caută să maximizeze coeficientul Fourier C_1 (relaţiile 5.28, 5.29, 5.34) modificând pulsaţia fundamentalei ω. Practic se pleacă de la o frecvenţă a fundamentalei de f = 50 Hz şi un increment Δf = 1 Hz, care se aplică peste f până se ajunge la un C_1 mai nefavorabil (mai mic) decât precedentul. Atunci Δf este înmulţit cu –0,5 şi se repetă procedeul până Δf scade sub 10^{-6} în valoare absolută.

Programul generează trei grafice. Primul arată puterile aparentă, activă, reactivă şi deformantă pe acelaşi grafic în culorile negru, albastru, verde şi roşu respectiv. Acest grafic este interactiv: un clic cu mouse-ul pe el va determina calcularea unui moment de timp şi redesenarea celorlalte grafice în funcţie de acesta. Pe abscisă este timpul în secunde, pe ordonată puterea în VA respectiv W, VAR, VAD (dimensional sunt aceleaşi).

Al doilea grafic arată tensiunea şi curentul în reţea pe acelaşi grafic cu două ordonate. Pe abscisă este timpul în milisecunde (40 ms, două alternanţe), pe ordonata stângă tensiunea în volţi şi pe ordonata dreaptă curentul în amperi.

Al treilea grafic arată spectrul curentului cu bare albastre suprapuse peste limitele din standardul EN 61000–3–2:2006+A1+A2 (tabelul 5.1) colorate în verde, pentru armonicile între 2 şi 40. Figura 5.13 arată rezultatele în regim de funcţionare normal, cu magnetronul alimentat şi filamentul catodului acestuia cald. Figura 5.14 reprezintă situaţia cu magnetronul nealimentat (în pauză), consumul fiind dat de motorul ventilatorului de răcire şi cel al platanului. Figura 5.15 arată ce se întâmpla în cazul în care magnetronul este alimentat, dar filamentul catodului e încă rece.

Durata regimului tranzitoriu este de 3,2 secunde. Aceasta înseamnă că un ciclu pornit/oprit nu poate fi mai scurt, de fapt durata acestor cicluri este de ordinul zecilor de secunde.

S-au făcut mai multe măsurători, inclusiv la diverse sarcini (corpuri de încălzit). Aşa cum era de aşteptat, am constatat că nu există nici o influenţă a sarcinii asupra parametrilor electrici. Magnetronul şi circuitul aferent nu sunt influenţate deloc de ce se află în cavitatea rezonantă.

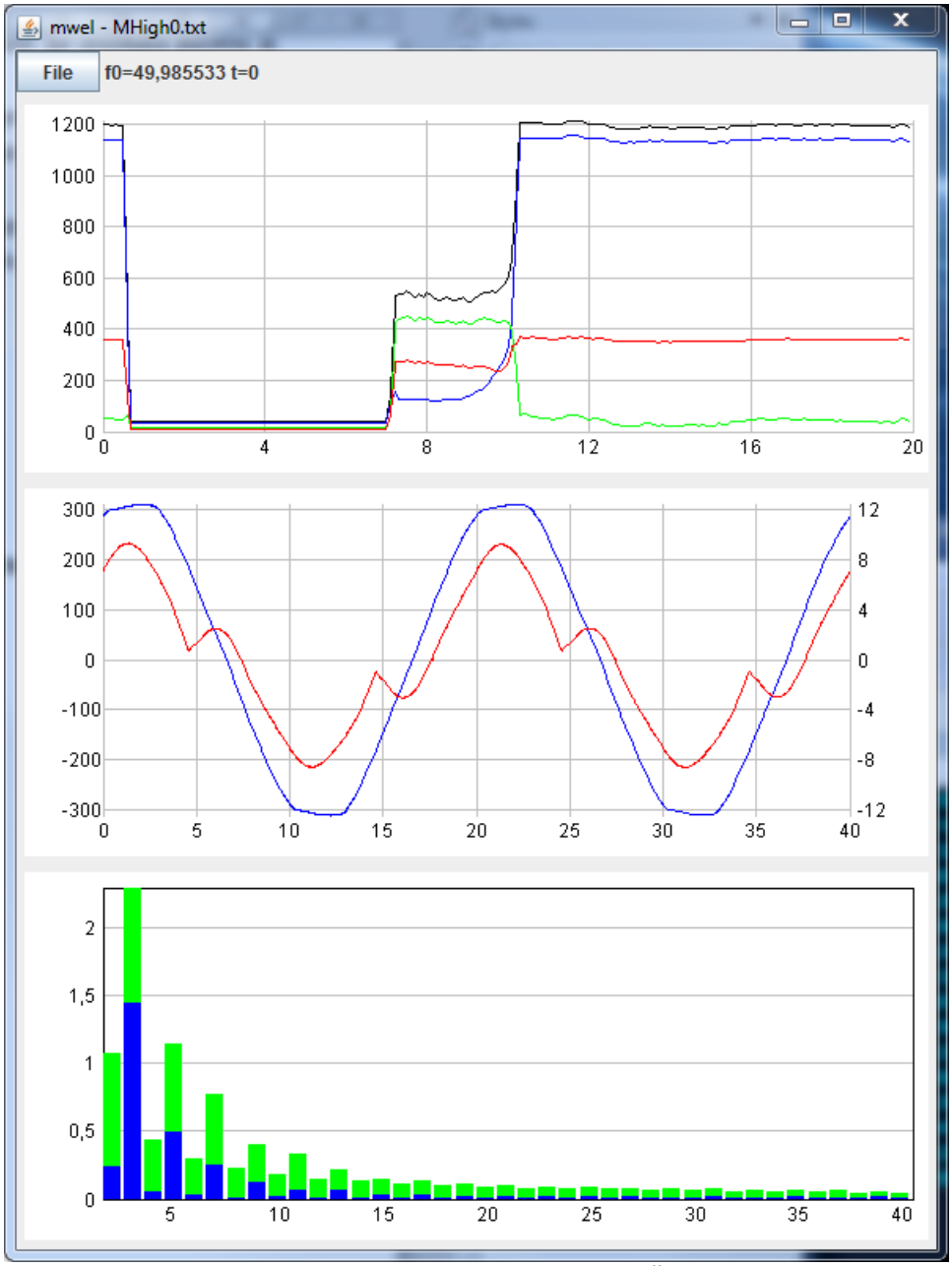


Fig. 5.13. Programul mwel, date la funcţionare normală (magnetron cald)

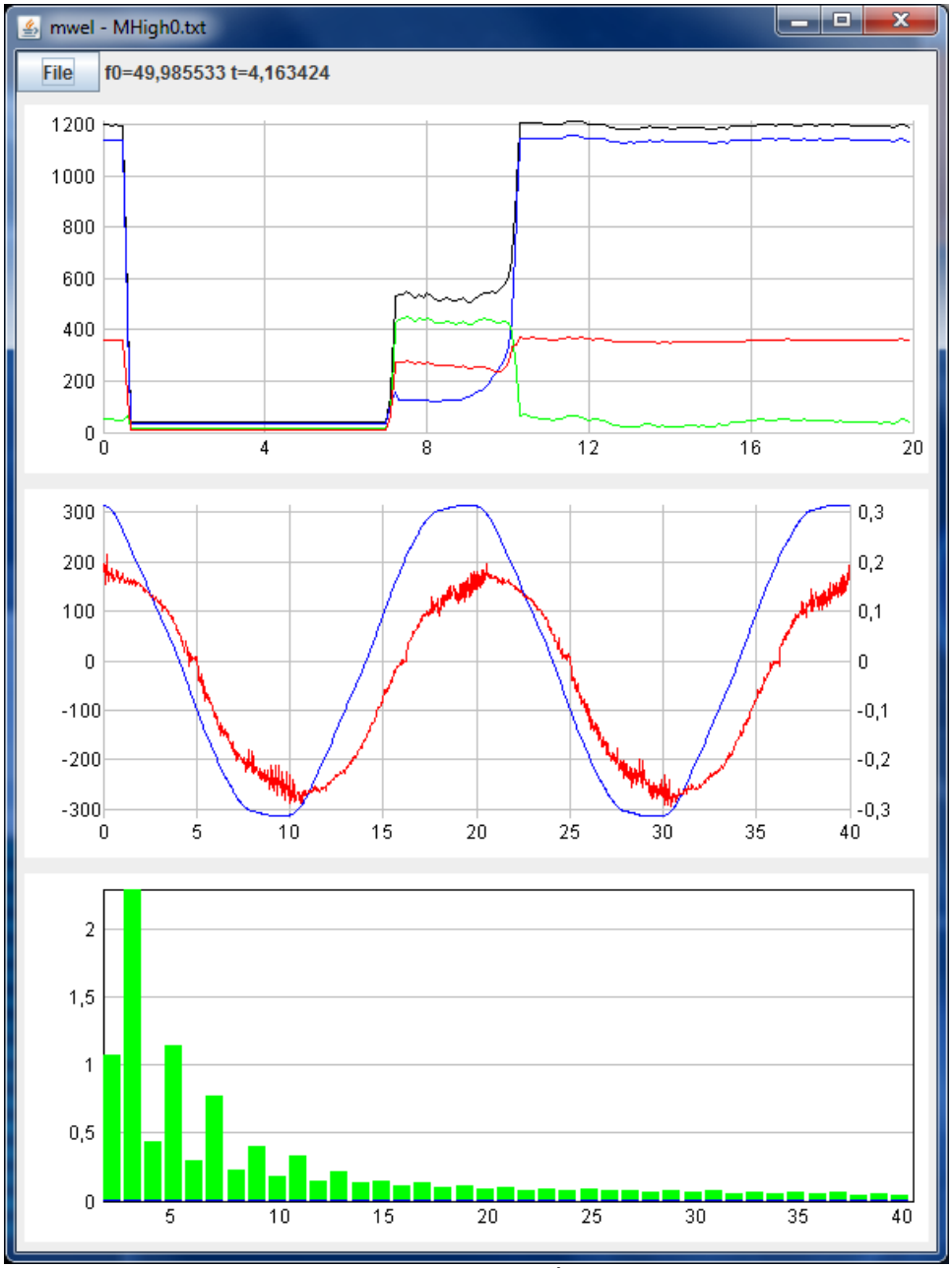


Fig. 5.14. Programul mwel, date la funcţionare în gol (magnetron nealimentat)

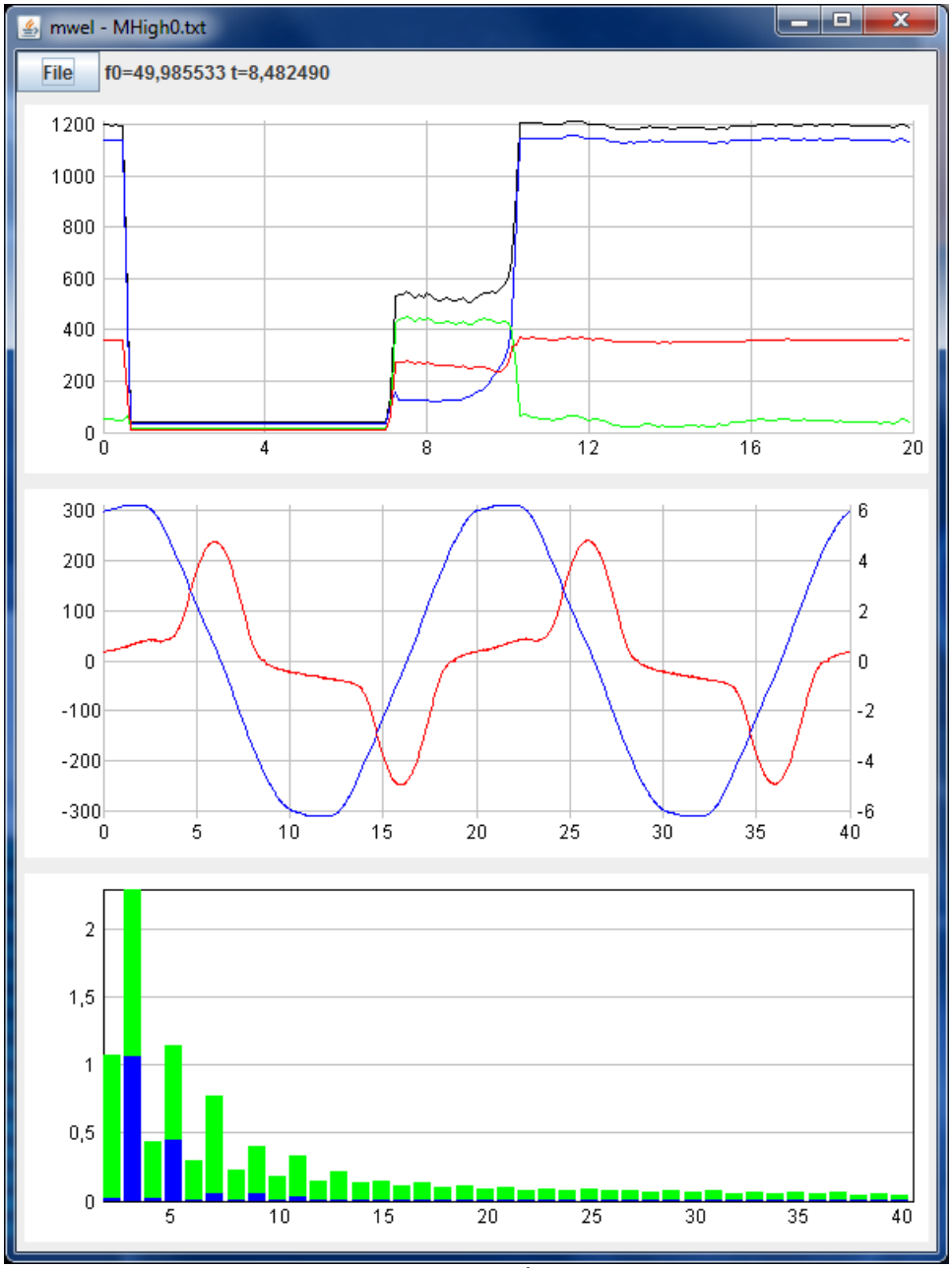


Fig. 5.15. Programul mwel, date la funcţionare în regim tranzitoriu (magnetron rece)

5.2.2. Calculul parametrilor magnetronului

Magnetronul poate fi modelat [44] printr-o diodă perfectă înseriată cu o sursă de tensiune (tensiunea de deschidere) şi un rezistor (rezistenţa dinamică). Aceşti parametri sunt dificil de măsurat direct, însă pot fi deduşi din forme de undă achiziţionate la funcţionare normală. Pentru aceasta s-au făcut măsurători conectând direct un traductor Hall bine izolat în circuitul catodului magnetronului, tensiunea fiind măsurată în continuare conform schemei 5.9.

Formele de undă au fost introduse într-un alt program scris în Java, numit **mwelim**. Acesta caută cei doi parametri printr-un algoritm asemănător celui folosit la calculul frecvenţei exacte a fundamentalei. Scopul urmărit de algoritm este să obţină o apropiere maximă între curentul măsurat (al 2-lea grafic din fig. 5.16) şi cel dedus din model şi tensiunea pe magnetron (primul grafic). Forma de undă a curentului rezultat din model este în al treilea grafic.

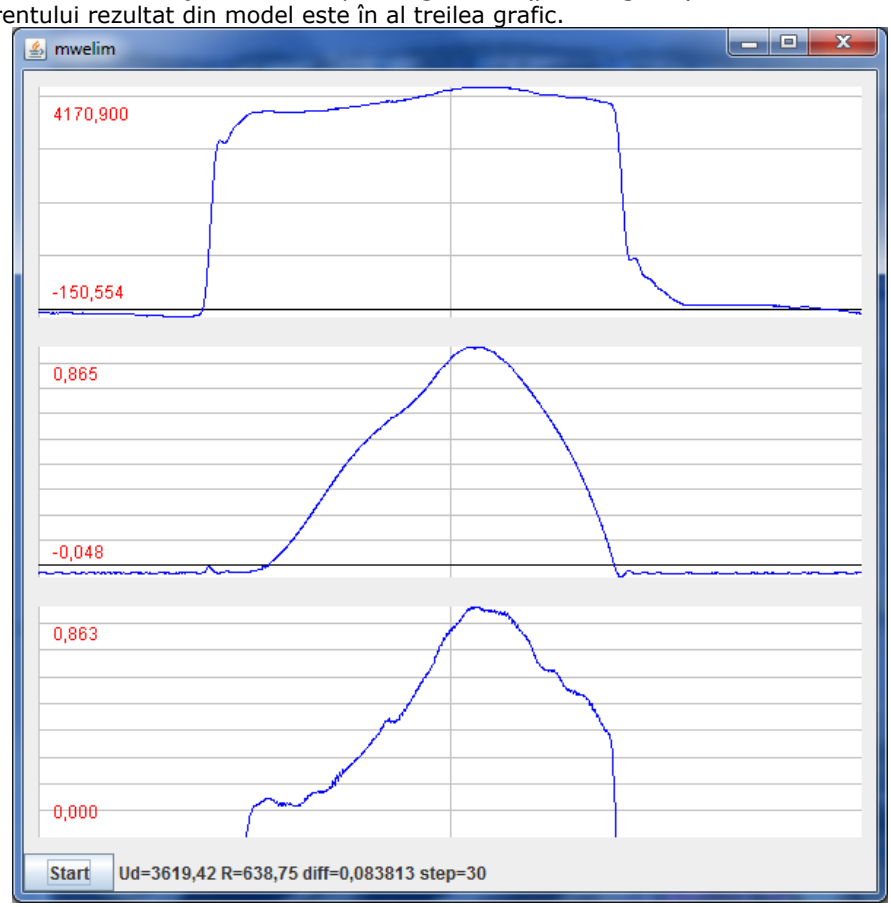


Fig. 5.16. Programul mwelim şi rezultatele

5.3. Simularea funcţionării cuptorului

Simularea funcţionării cuptorului cu microunde s-a realizat utilizând schema din figura 5.17. Această schemă cuprinde un transformator cu caracteristici identice cu ale transformatorului cuptorului cu microunde la care s-au măsurat parametrii şi un condensator de 1μF care împreună cu dioda de înaltă tensiune realizează un dublor de tensiune. Simularea funcționării diodei de HV s-a făcut utilizând o diodă ideală și o sursă de tensiune contraelectromotoare de 9V. Magnetronul a fost simulat utilizând o diodă ideală, o sursă de tensiune contraelectromotoare de 3,62kV și o rezistență serie de 639 ohmi – parametri rezultaţi din cap. 5.2.2. Deoarece în practică reglarea puterii cuptorului cu microunde se face prin modificarea intervalului de timp în care este alimentat magnetronul, s-a pus problema dacă se poate face un reglaj utilizând o punte cu două tranzistoare IGBT la care se modifică unghiul de comandă. Simulările s-au făcut la diverse unghiuri de comandă: 0°, 30°, 45°, 60°, 90° si 135°.

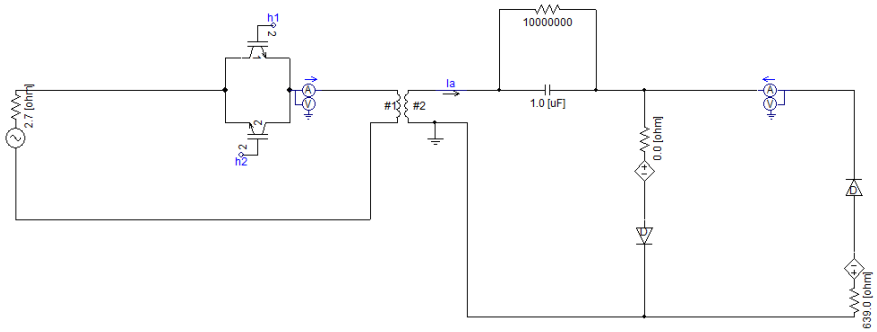


Fig. 5.17. Schemă simulare

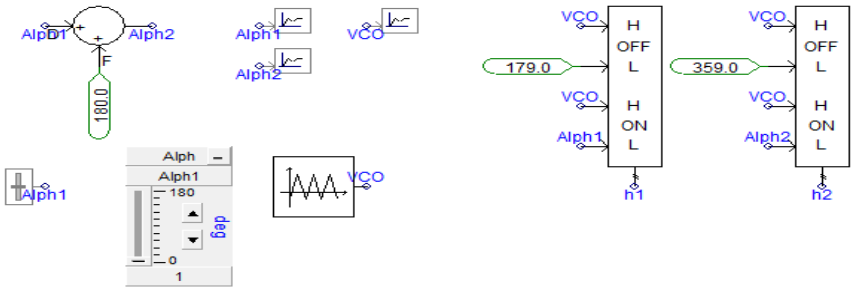


Fig. 5.18. Model comandă tranzistoare IGBT

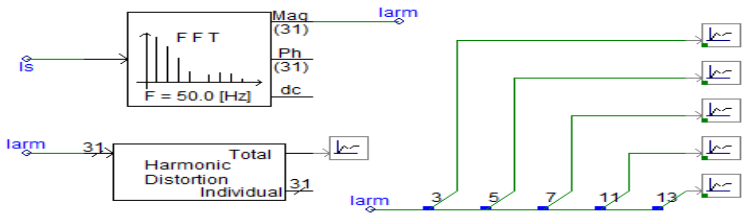
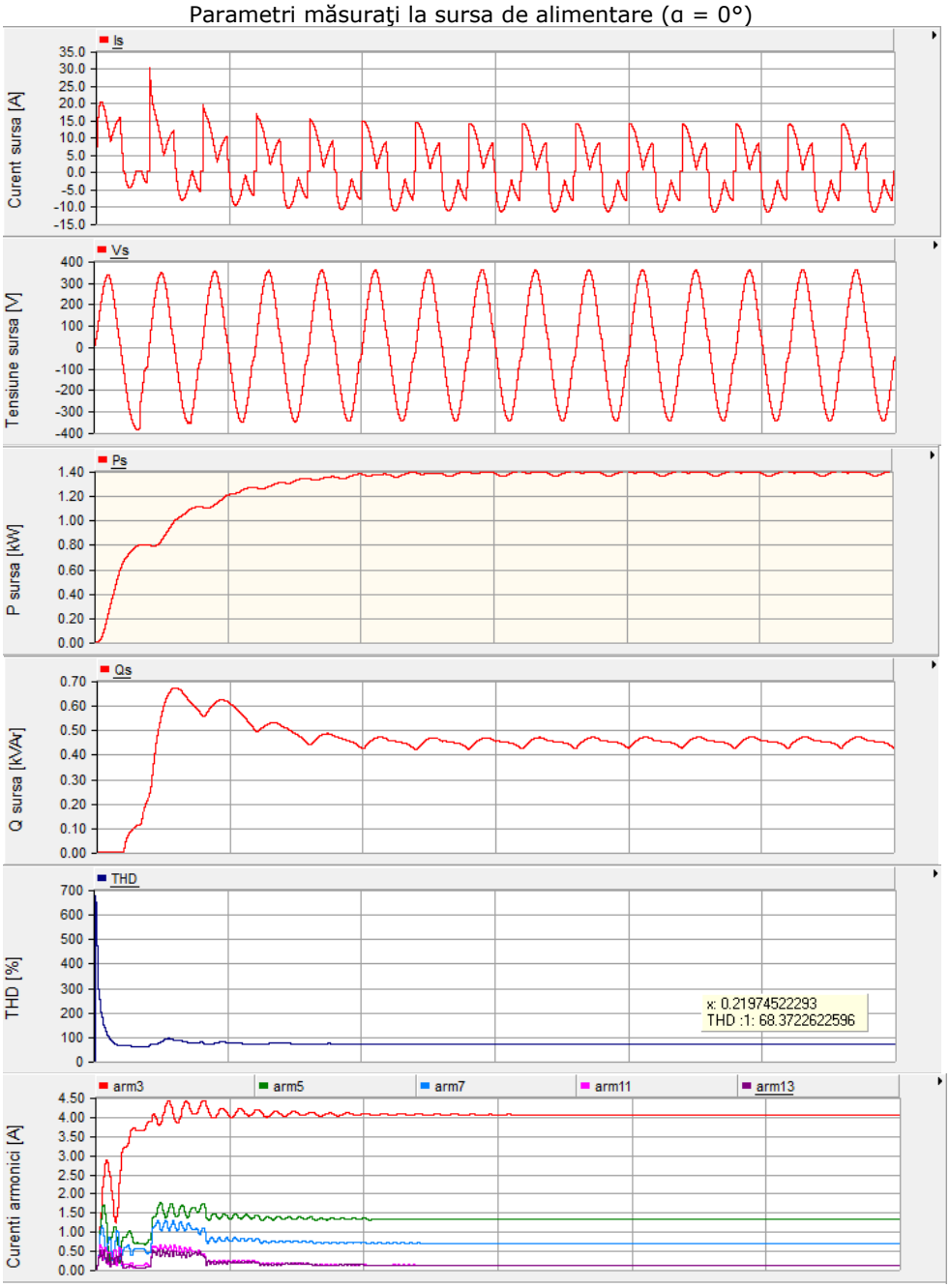
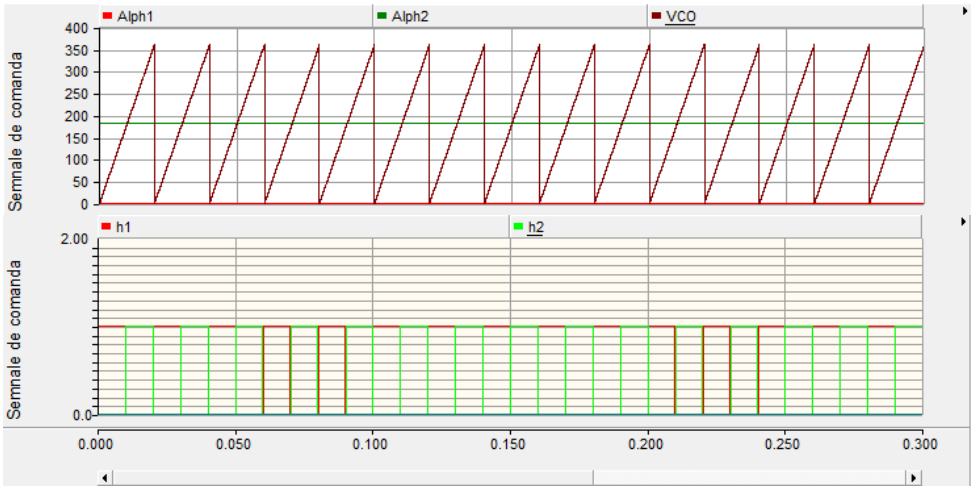


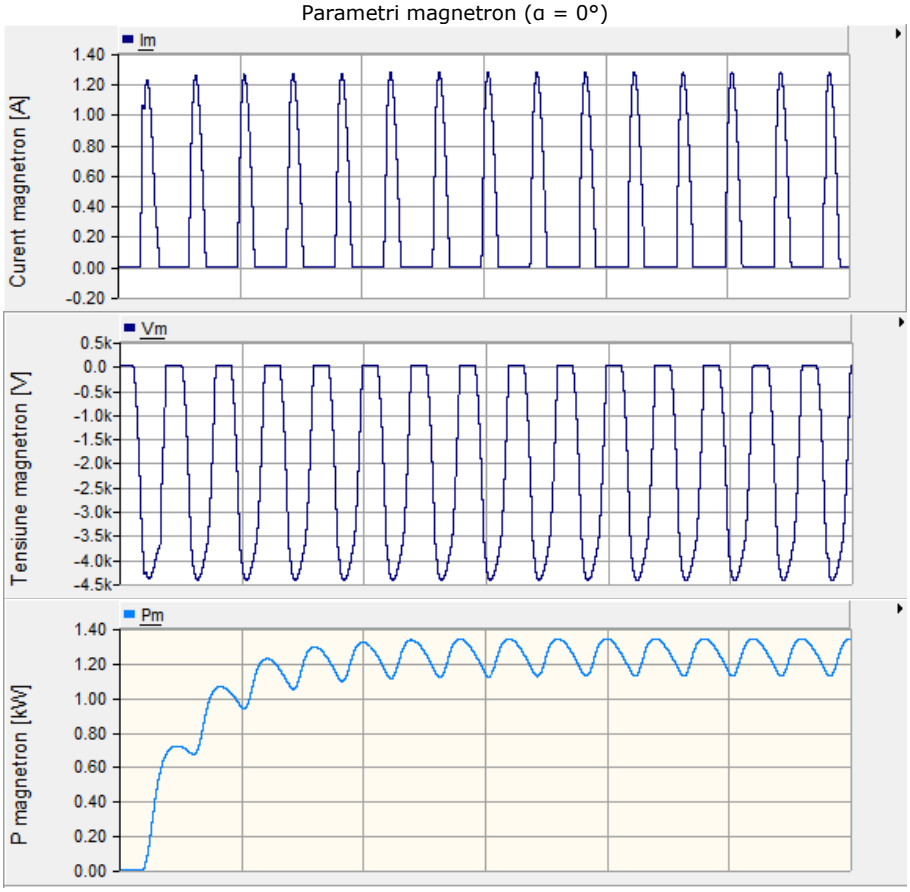
Fig. 5.19. Model de calcul curenţi armonici

Parametri măsuraţi la sursa de alimentare (α = 0°)

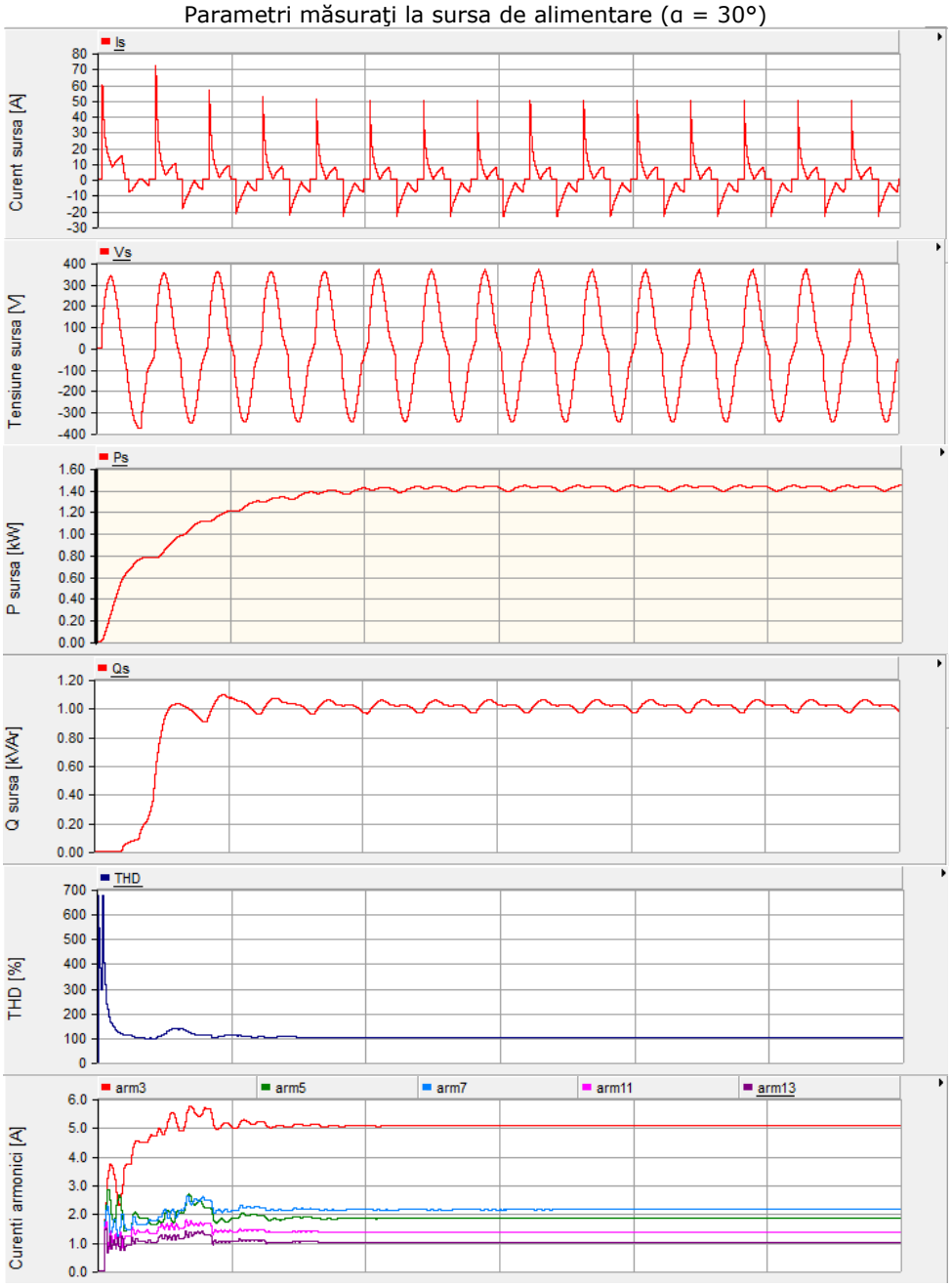


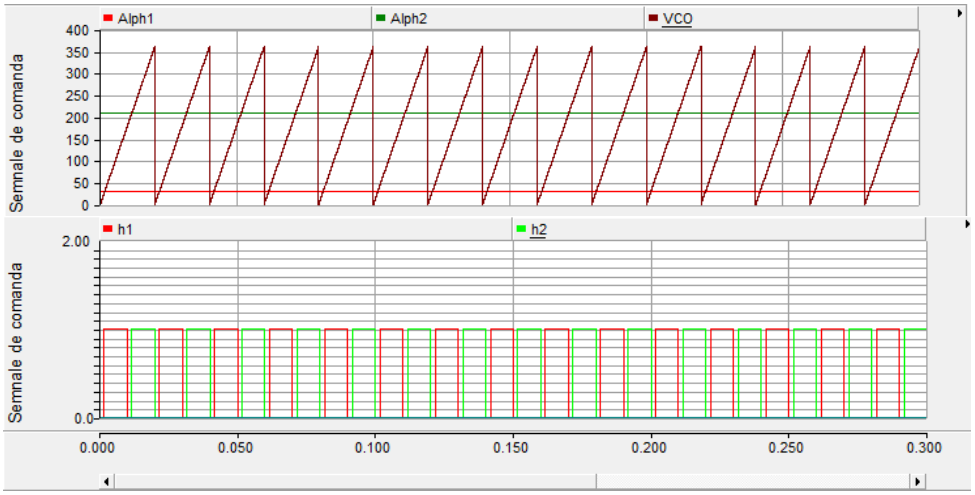


Parametri magnetron (α = 0°)

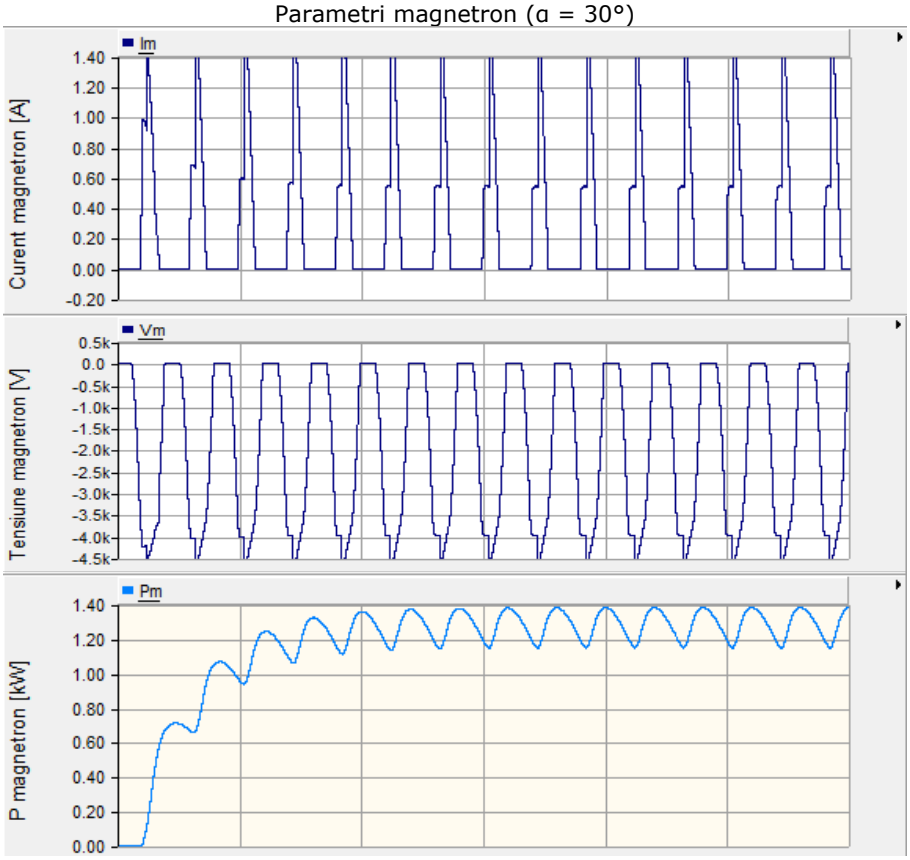


Parametri măsuraţi la sursa de alimentare (α = 30°)

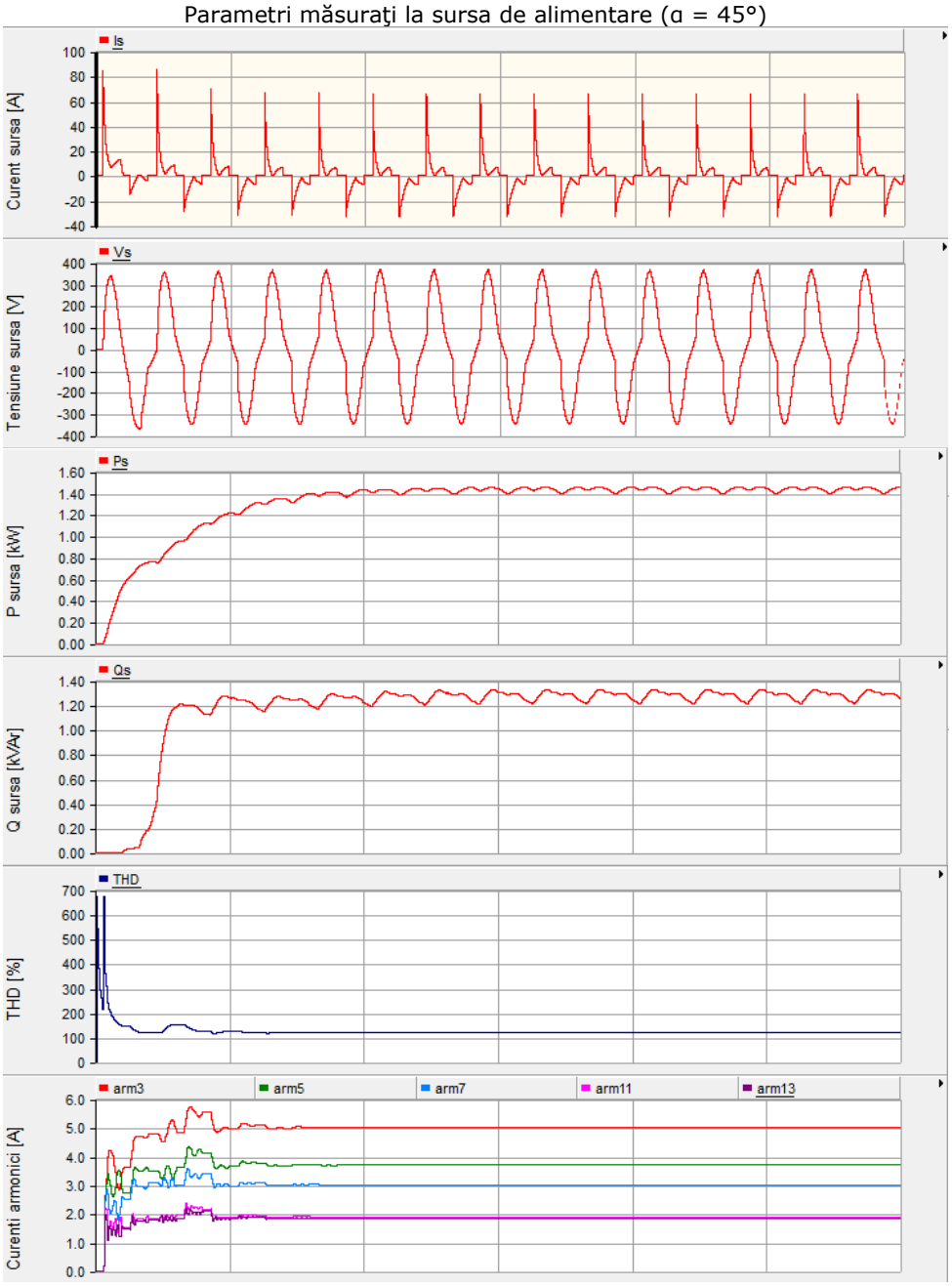


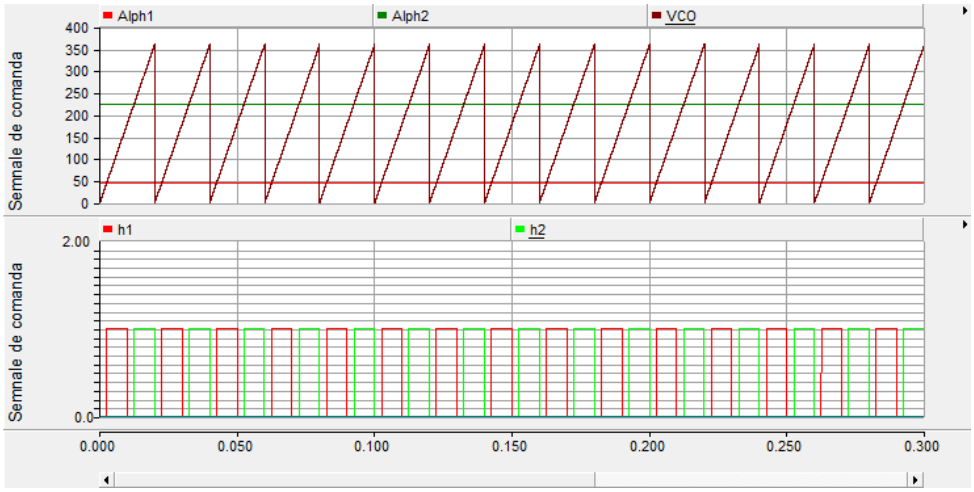


Parametri magnetron (α = 30°)

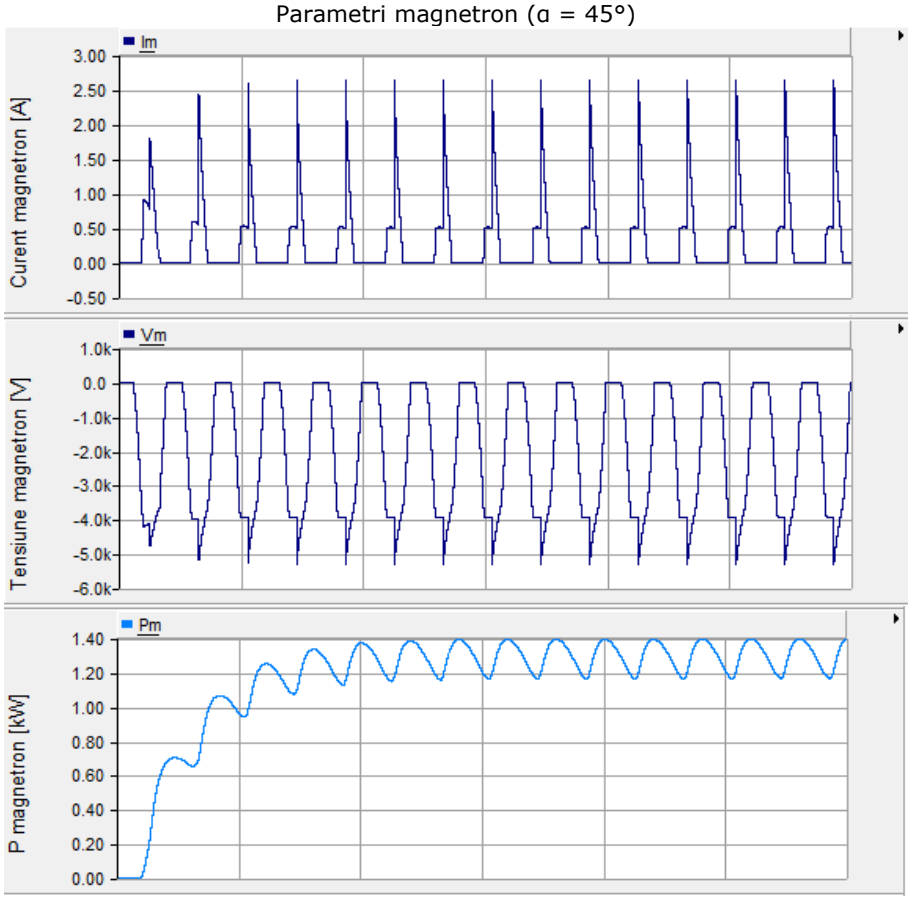


Parametri măsuraţi la sursa de alimentare (α = 45°)

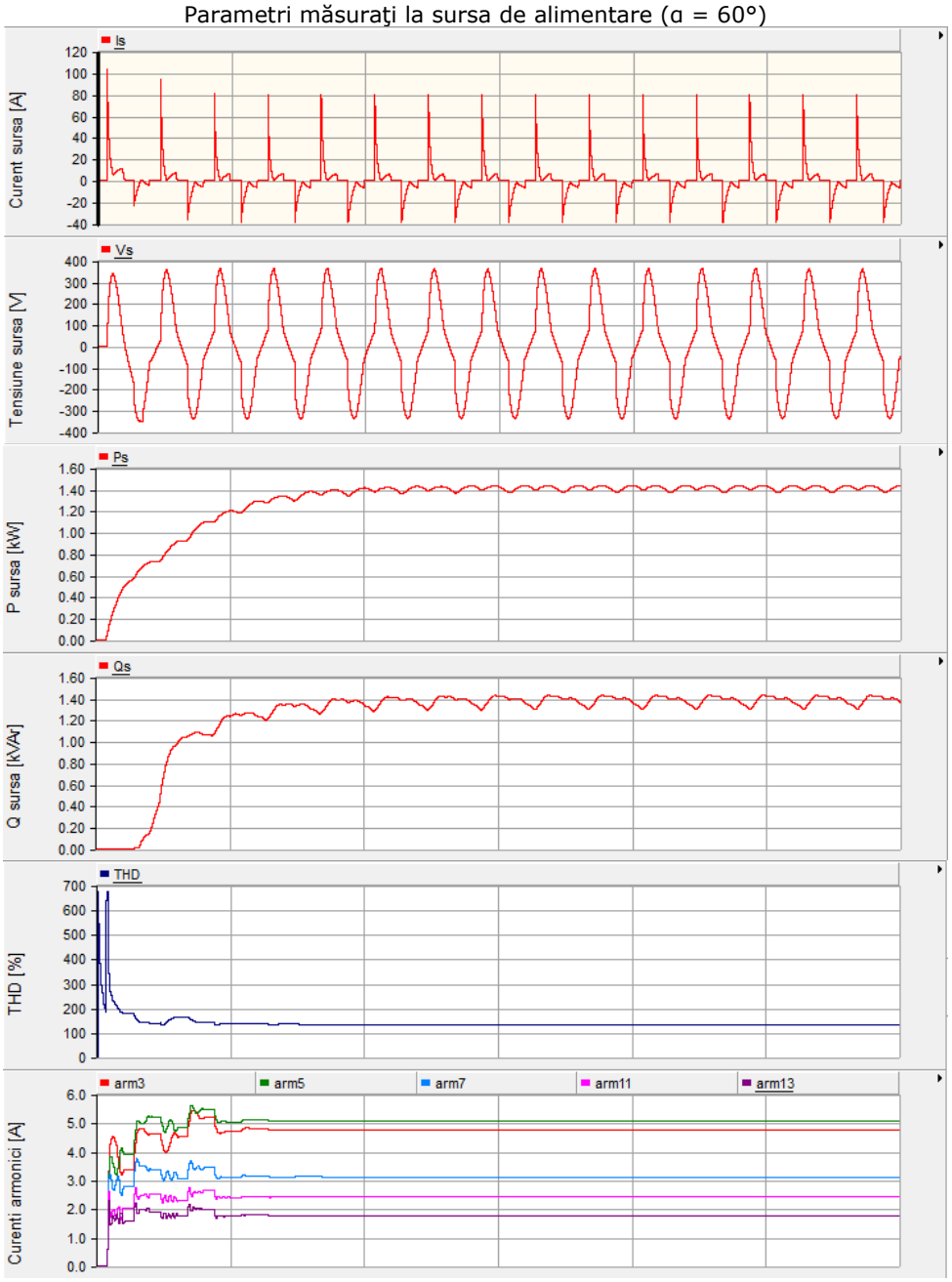


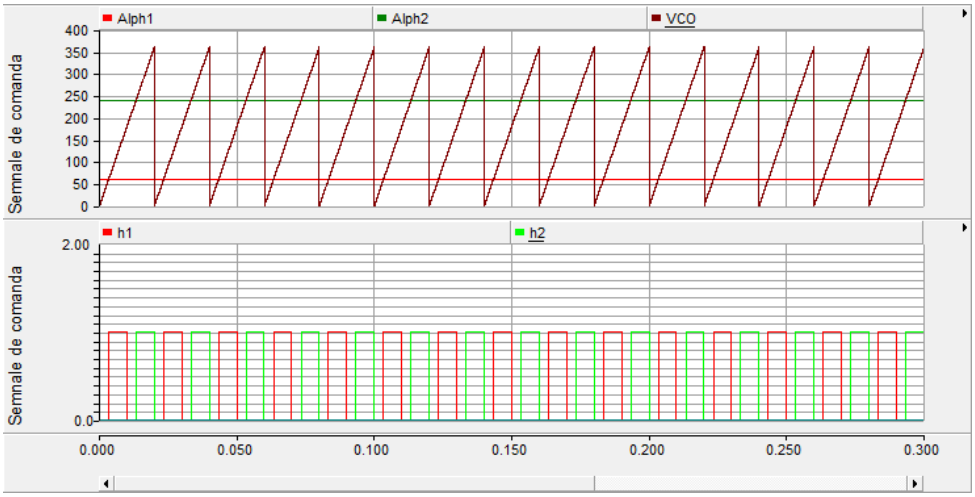


Parametri magnetron (α = 45°)

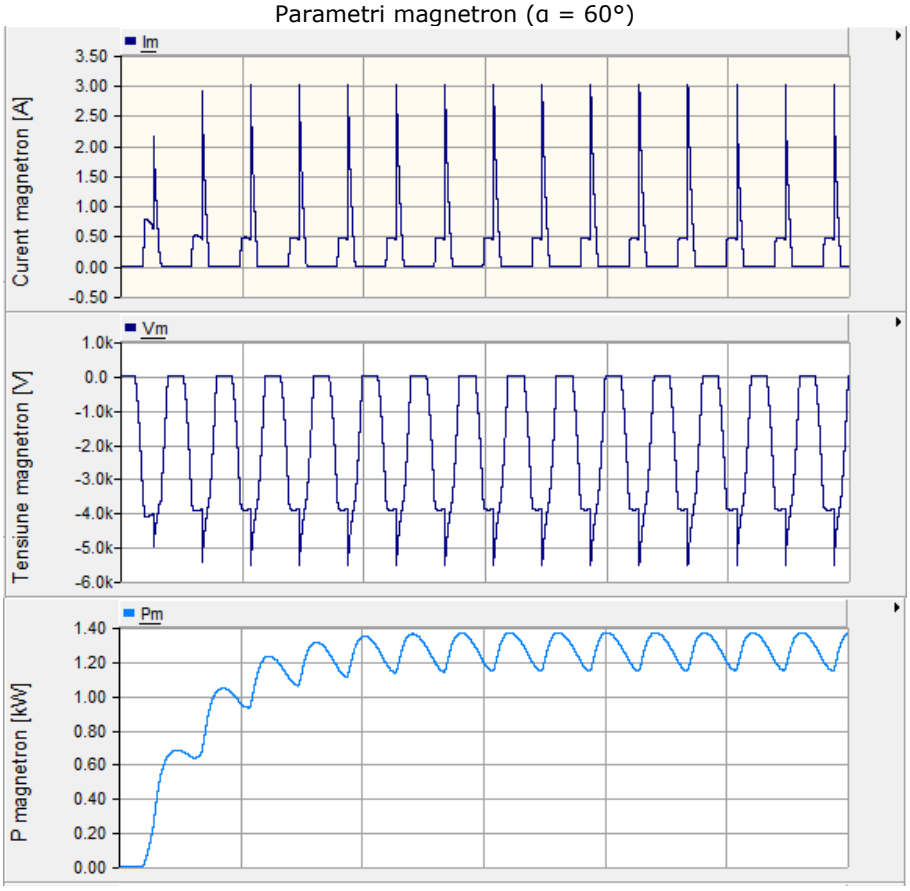


Parametri măsuraţi la sursa de alimentare (α = 60°)

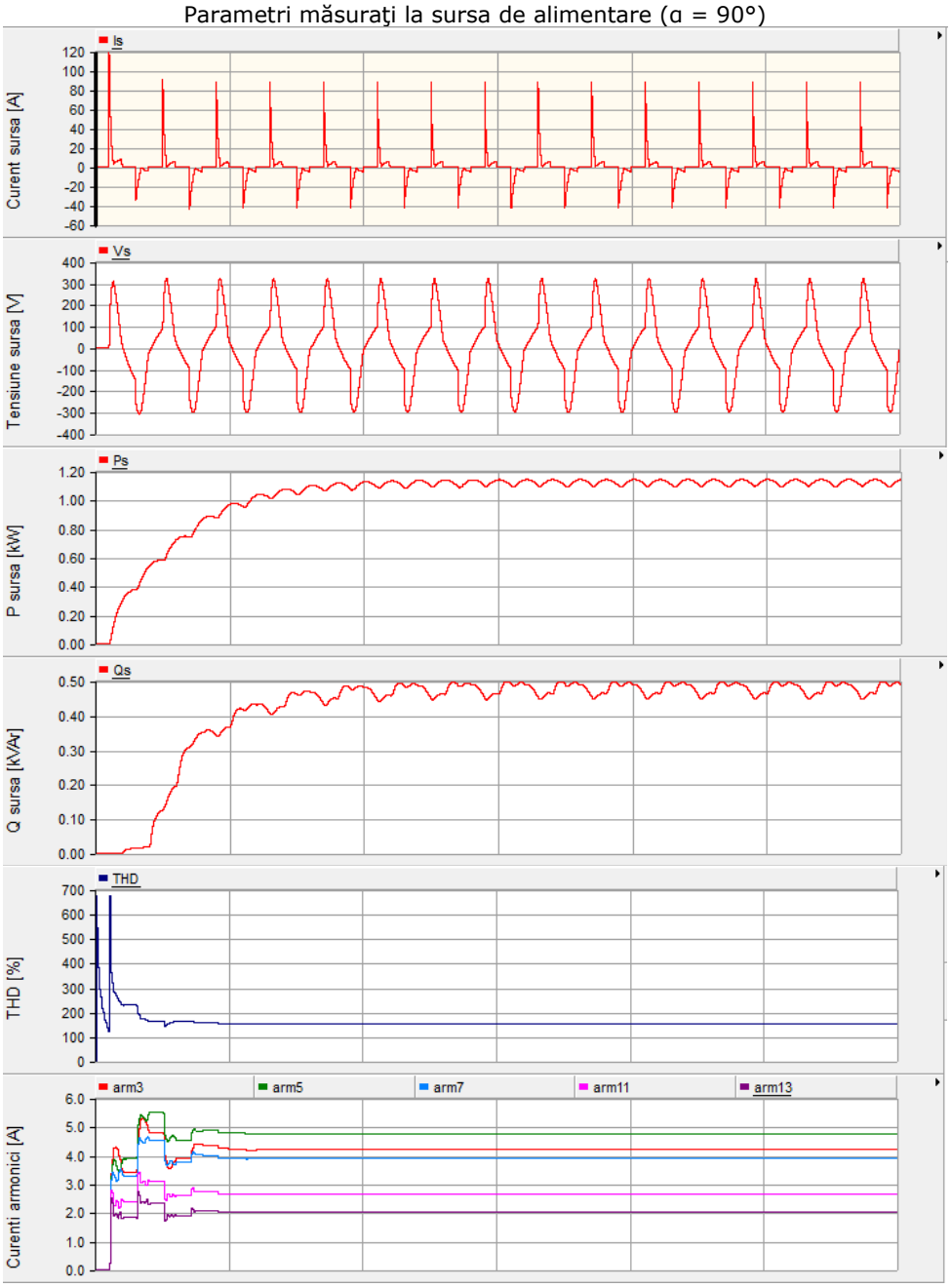


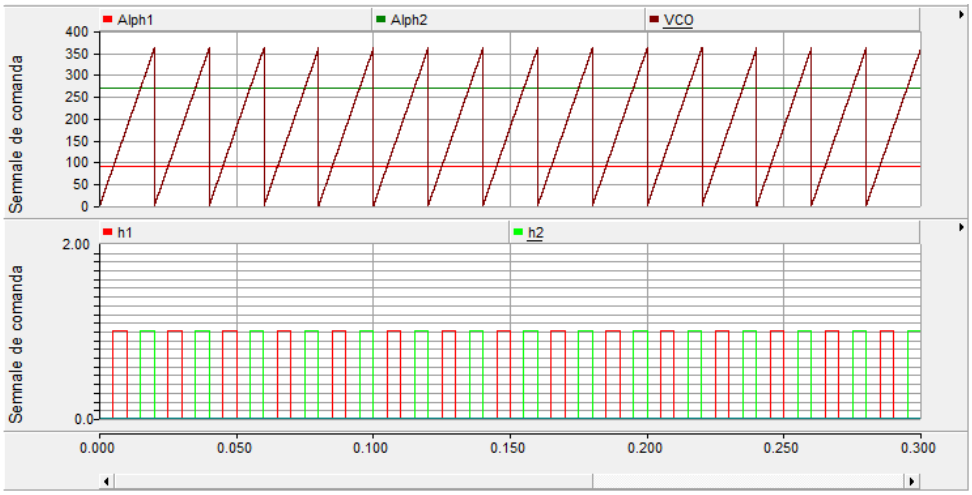


Parametri magnetron (α = 60°)

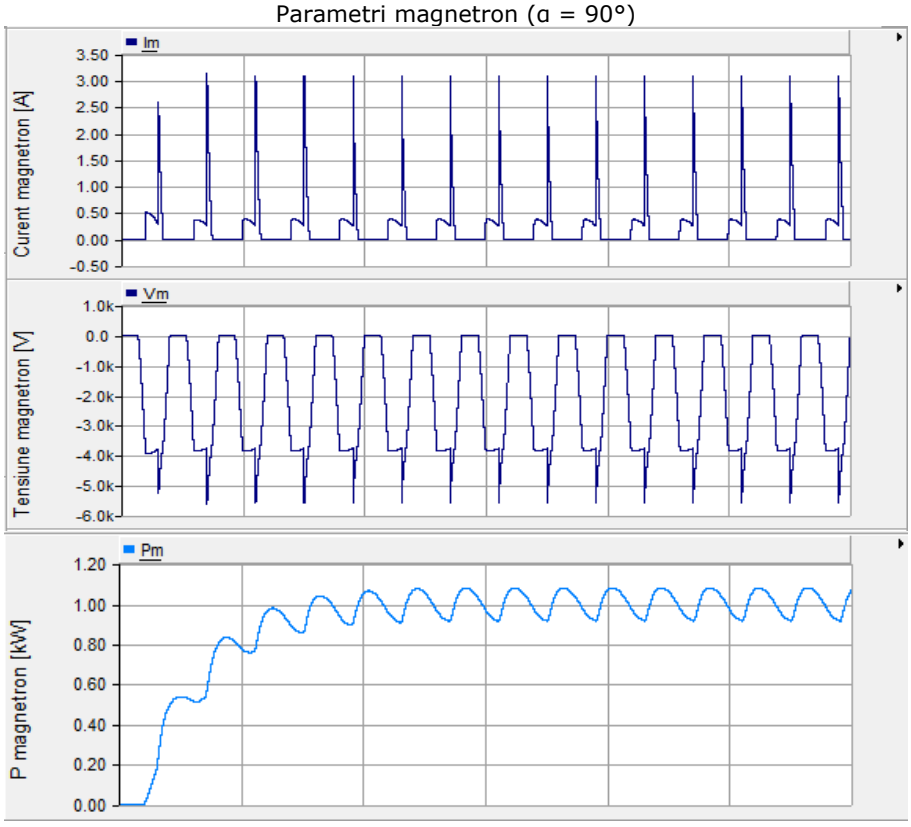


Parametri măsuraţi la sursa de alimentare (α = 90°)

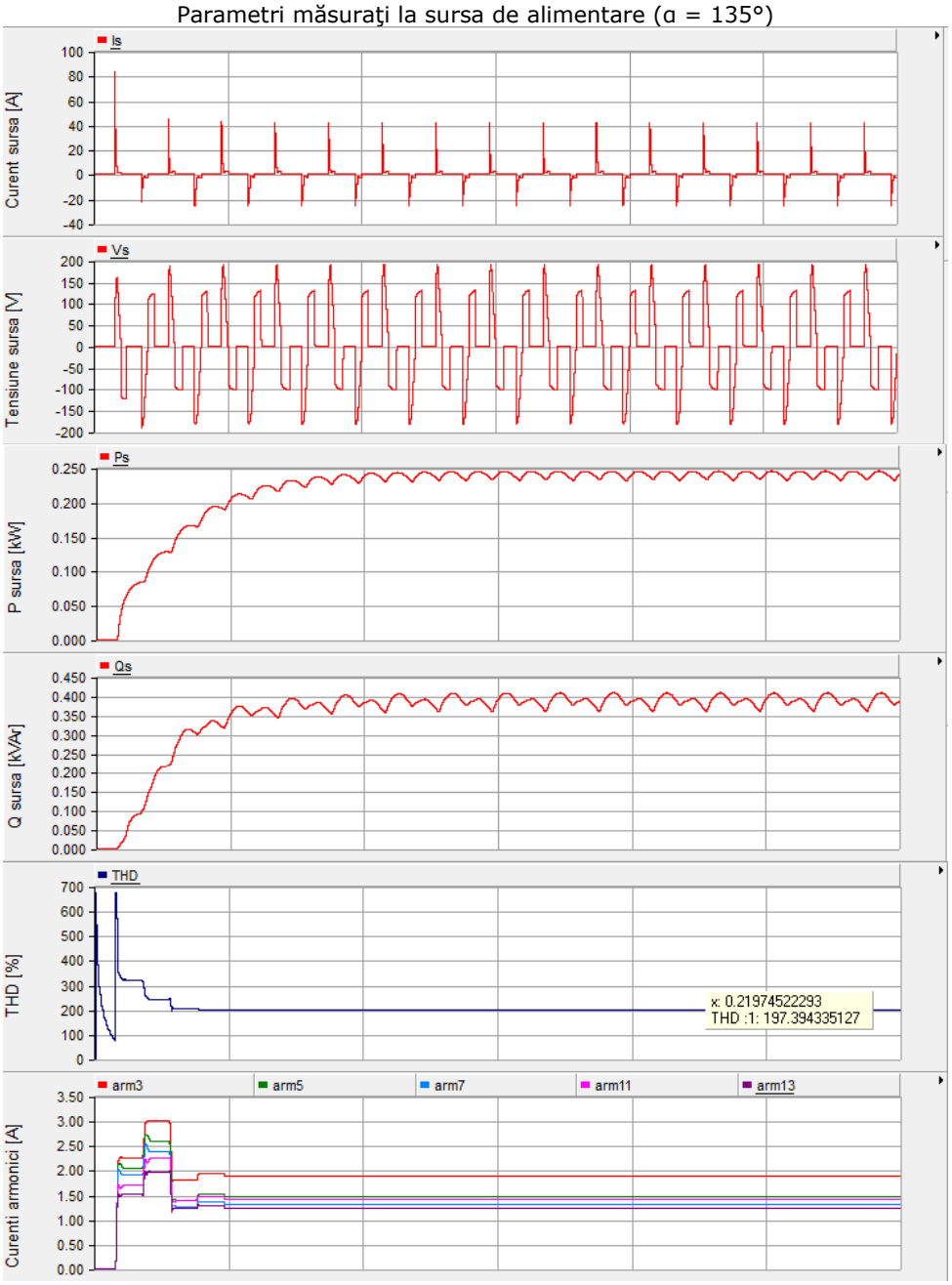


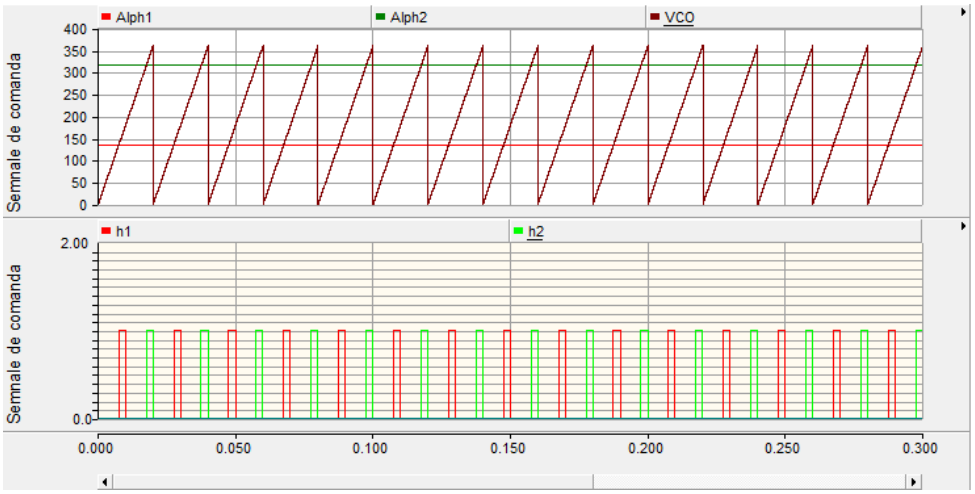


Parametri magnetron (α = 90°)

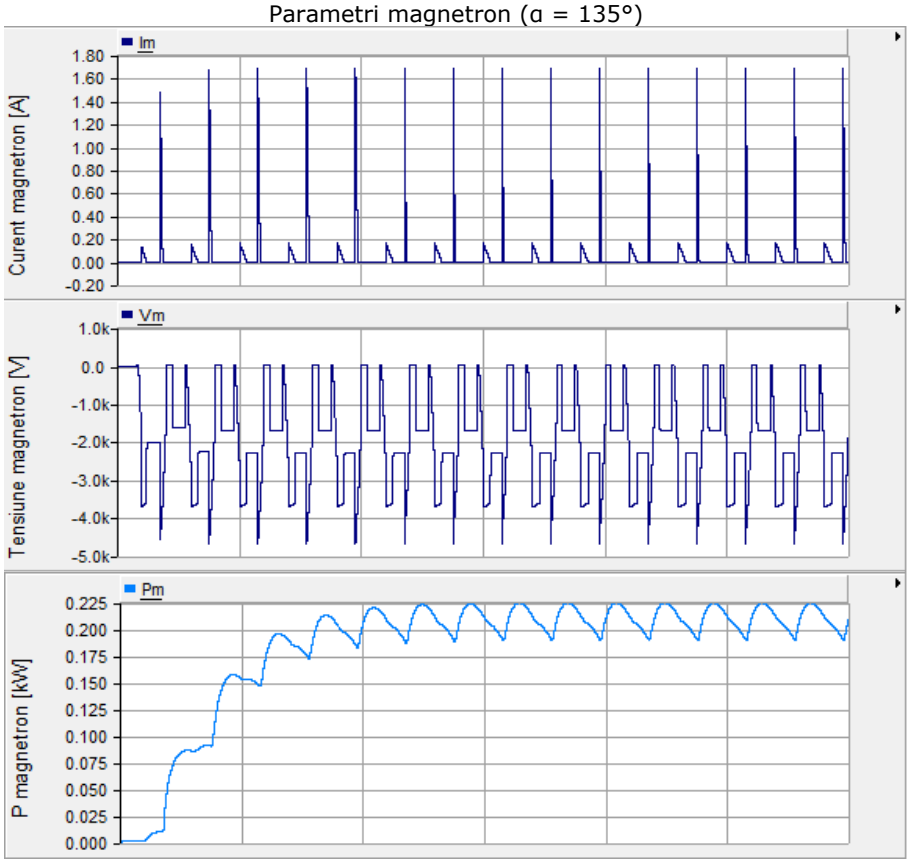


Parametri măsuraţi la sursa de alimentare (α = 135°)





Parametri magnetron (α = 135°)



În urma simulărilor s-a constatat că odată cu creșterea unghiului de comandă scade puterea activă absorbită de la sursă, dar crește distorsiunea armonică. Din această cauză la reglarea prin modificarea unghiului de comandă apar două efecte:

- crește complexitatea schemei de comandă, deci și prețul cuptorului cu microunde
- crește gradul de poluare armonică a celorlalți consumatori conectați la aceeași rețea de distribuție

Tab. 5.2. Variația parametrilor electrici la modificarea unghiului de comandă

Unghi comandă	THD [%]	Putere magnetron [kW]
0°	68	1,38
30°	99	1,31
45°	117	1,28
60°	133	1,26
90°	152	0,92
135°	197	0,23

6. Concluzii şi contribuţii personale

Această lucrare constituie un studiu privind echipamente şi procese electrotermice în câmp de microunde. În urma studiului realizat asupra cuptoarelor multimod cu cavitate rezonantă au rezultat modele matematice şi programe de simulare validate experimental, care au permis determinarea unor caracteristici extrem de dificil de obţinut altfel. A fost analizată funcţionarea cuptoarelor de serie din punct de vedere electric şi a fost concepută o metodă de reglaj continuu a puterii bazată în cea mai mare parte pe componente ieftine, fabricate în serie mare. Această metodă a fost verificată prin simulări.

Capitolul 1.

În primul capitol este făcută o introducere asupra problematicii încălzirii volumetrice prin mijloace electrice, este prezentată evoluţia şi stadiul actual al sistemelor de încălzire cu microunde.

Capitolul 2.

Al doilea capitol conţine consideraţii teoretice asupra încălzirii în câmp de microunde şi a condiţiilor şi fenomenelor care limitează densitatea de putere disipată, şi în consecinţă productivitatea şi randamentul sistemelor de încălzire cu microunde.

Capitolul 3.

Al treilea capitol conţine consideraţii asupra câmpului electromagnetic în structurile de microunde: ghiduri de undă, cavităţi rezonante, aplicatoare.

Capitolul 4.

Acest capitol tratează modelarea şi simularea funcţionării unui cuptor multimod în condiţiile unei excitaţii constante cu frecvenţă fixă. Câmpul electromagnetic din aplicator şi câmpul termic din sarcină sunt cuplate, simularea ţinând cont de dependenţa caracteristicilor electrice care determină disipaţia termică de temperatură. Modelul electromagnetic este în domeniu timp, ceea ce îl face intensiv din punct de vedere al calculelor dar permite analizarea regimurilor tranzitorii. Modelul termodinamic al sarcinii permite simularea transformărilor de fază, chiar şi a celor izoterme. Utilizând tehnici de programare avansate, autorul a reuşit să obţină timpi de execuţie rezonabili pentru simulări. Modelele sunt validate folosind măsurători realizate cu o cameră FLIR şi sarcini speciale cu caracteristici cunoscute.

Contribuţiile personale ale autorului sunt programe de simulare scrise în C++, măsurători experimentale, compararea rezultatelor şi validarea modelelor. S--au dezvoltat două programe: primul face simulările propriu-zise şi salvează rezultatele pe disc, al doilea este folosit la vizualizarea acestora şi compararea cu ce s-a obţinut din măsurători.

Capitolul 5.

În acest capitol sunt descrise şi analizate soluţiile constructive pentru cuptoarele de mică putere pe partea electrică. Au fost concepute şi realizate scheme de măsură şi achiziţie pentru curenţi şi tensiuni atât la reţea cât şi pentru magnetron şi celelalte componente. Au fost dezvoltate aplicaţii în mediul de

programare LabView care permit atât vizualizarea în timp real a formelor de undă achiziţionate şi analiza armonică a acestora, cât şi salvarea pe disc a eşantioanelor pe durată de 20 secunde. A fost conceput un program scris în Java pentru analiza detaliată a semnalelor şi un altul pentru a calcula parametri electrici ai magnetronului folosind formele de undă obţinute. Utilizând aceşti parametri s-a realizat o simulare în mediul PSCAD-EMTDC pentru a obţine un reglaj continuu al puterii.

Contribuţiile personale ale autorului sunt: schemele de măsură, aplicaţiile LabView, măsurătorile propriu-zise, programele Java şi simularea în PSCAD.

Bibliografie

1. N. Golovanov, I. Şora ş.a., *Electrotermie si electrotehnologii*, Ed. Tehnică, Bucureşti, 1997
2. A. C. Metaxas, *Foundations of Electroheat*, Wiley, 1996
3. L. Assinder, *Microwave for food processing – industrial application of microwave energy*, Edited by R. Smith, IMPI Transactions 2, 92, 1974
4. A. C. Metaxas, R. J. Meredith, *Industrial Microwave Heating*, Peter Peregrinus Ltd., London, 1983
5. D. A. Copson, *Microwave heating*, A VI Publishing Co. Inc. New York 1975
6. N. Meisel, *Microwave energy application*, Newsletter XII, no. 6, pp.6-9, 1979
7. L. Bandici, *Modelarea numerică a câmpurilor electromagnetic şi termic cuplate din instalaţiile cu microunde*, teză de doctorat, Univ. Oradea, 2003
8. N. Voicu, *Sisteme cu microunde – elemente de teorie, construcţie*, Matrix Rom Bucureşti, 2004
9. E. N. Bengtsson, T. Ohlsson, *Microwave heating in the food industry*, IEEE Proceedings 62, No. 1, pp. 44–45, 1974
10. R. E. Nistor, R. Chirilă, *Unde electromagnetice în medii anizotrope*, Editura Tehnică, Bucureşti, 1999
11. A. C. Metaxas, *The use of modeling in RF and microwave heating*, HIS–01, Heating by Internai Sources - Padua, September 12-14, 2001, pp. 319-328
12. R. J. Meredith, *Engineers' Handbook of Industrial Microwave Heating*, IEE Power Series 25, 1998
13. D. C. Dibben, A. C. Metaxas, *Time domain finite element analysis of multimode microwave applicators*, IEEE Transactions of Magnetics, 32, Number 3, pp.942-945, 1996
14. R. J. Cook, *Microwave cavity methods in high frequency dielectric measurements*, IPC Science And Technology Press Ltd., New York, 1973
15. P. W. McMillan, R. M. Clements, *A simple technique for measuring high microwave electric field strengths*, Microwave Power 15(1), 1980
16. S. Lefeuvre, M. Majdapadino, *Methode mixte pour calcul des champ dans un four charge*, Journees Europeenes sur des methodes numeriques en Electromagnetisme, Toulouse, 1993
17. T. Leuca, C. Molnar, L. Bandici, *The Electromagnetic Field Distribution Within Microwave Equipment.* ATEE - 27 November 2002, Bucureşti, România
18. H. S. Carslaw, J. C. Jaeger, *Conduction of heat in solids*, Oxford University Press, 1959
19. G. Rulea, *Tehnica Microundelor*, Editura Didactică şi Pedagogică, Bucureşti, 1981.
20. G. Rulea, *Bazele teoretice şi experimentale ale tehnici microundelor*, Editura Ştiinţifică şi Enciclopedică, Bucureşti, 1989.
21. S.T. Saad, *Microwave engineers' handbook*, vol I, Publishers Artech house, Dedham, Massachusetts, 1989.
22. D.D. Sandu, *Dispozitive electronice pentru microunde*. Editura Ştiinţifică şi Enciclopedică, Bucureşti, 1982.
23. M.A. Silaghi, *Analiza câmpului electromagnetic în instalaţiile de încălzire prin microunde*, teză de doctorat, Universitatea din Oradea, 1999.
24. M.A. Silaghi, H. Silaghi, *Tehnologii cu microunde. Tehnici informatice*. Editura Treira, Oradea, 2001.

25. C. Şora, *Bazele electrotehnicii*, Editura Didactică şi Pedagogică, Bucureşti, 1982.
26. A. Tomescu, F.M.G. Tomescu, *Sisteme cu microunde*. Editura Matrix Rom, Bucureşti, 2001.
27. E. Nicolau, *Radiaţia şi propagarea undelor electromagnetice*. Editura Academiei, 1989
28. C. Popescu, *Materiale electrotehnice. Proprietăţi şi utilizări.* Editura tehnică, Bucureşti, 1976.
29. T.G. Mihran, *Microwave oven mode tuning by slab dielectric loads.* IEEE Transactions of Microwave Theory and Techniques MIT–26, no.6, pp. 380, 1978.
30. G. E. Lojevschi, *Linii de transmisie pentru frecvenţe înalte.* Editura Tehnică, Bucureşti, 1996.
31. T. A. Johnk, *Engineering Electromagnetic field and waves*, Wiley, New York, 1975.
32. P. F. Collins, *Field Theory of Guided Waves.* 2nd ed. IEEE Press, New York, 1991.
33. Şt. Cantaragiu, *Circuite de microunde. Metode de calcul.* Editura All Educaţional, Bucureşti, 2000.
34. I. Botez-Casian, *Teoria şi proiectarea circuitelor de microunde.* Editura Matrix Rom, Bucureşti, 1998.
35. A. Bossavit, *Solving Maxwell's Equations in a Closed Cavity, and the Question of Spurious Modes*. IEEE Trans. MAG-26, p.702-705, 1990.
36. Gh. Gavriloaia, *Analiza numerică a câmpului de microunde*, Teora, Bucureşti, 2001
37. I. Tripşa, M. Sălcudeanu, M. Costescu, *Optimizarea proceselor de turnare şi solidificare a oţelului*, Editura Tehnică, Bucureşti 1975
38. C. Abrudean, M. Pănoiu, C. Pănoiu, I. Şora, *Using Finite Difference Time Domain Method to Simulate the Electromagnetic Field in a Multimode Microwave Oven*, Proceedings of the 13th WSEAS International Conference on Computers, July 23 25, 2009, Rhodes, Greece, ISBN 978-960-474-099-4, ISSN 1790-5109
39. C. Abrudean, *Numerical Simulation of Phase Transitions in Heating or Refrigeration Processes*, 11th International Conference of Numerical Analysis and Applied Mathematics (ICNAAM), Greece, 2013, AIP Conference Proceedings vol. 1558 pages: 1329-1332, ISBN 978-0-7354-1185-2, ISSN 0094-243X
40. C. Abrudean, *Optimization of Iterative Calculus over Large Arrays in Multiprocessor Systems*, 11th International Conference of Numerical Analysis and Applied Mathematics (ICNAAM), Greece, 2013, AIP Conference Proceedings vol. 1558 pages: 1341-1344, ISBN 978-0-7354-1185-2, ISSN 0094-243X
41. T. Koryu Ishii (editor), *Handbook of Microwave Technology*, vol. 2, Academic Press Inc., ISBN 0-12-374697-3, San Diego, 1995
42. Rob Raluca, *Reducerea poluării electromagnetice la echipamente electrotermice de înaltă frecvență*, Ed. Politehnica, 2013, ISBN 978-606-554-705-6, ISSN 1842-7022
43. http://www.rfemcdevelopment.eu/index.php/en/emc-emi-standards/en-61000-3-2-2006-a1-a2 - descărcat 09.06.2014

44. B. Bahani et. al, *Modeling of a New High Voltage Power Supply for Microwave Generators with Three Magnetrons*, International Journal of Electrical and Computer Engineering (IJECE), Vol. 3, No. 2, April 2013, ISSN 2088-8708
45. A. Iagăr, C. Abrudean, *Studiul câmpurilor electromagnetic şi termic în cazul cuptoarelor de inducţie cu creuzet pentru topirea oţelului feromagnetic*, Analele Facultăţii de Inginerie Hunedoara, tom I, Fasc.3, ISSN 1454 – 6531, 1999
46. A. Iagăr, C. Abrudean, *Optimizarea procesului de topire prin inducţie a aluminiului*, Analele Facultăţii de Inginerie Hunedoara, tom II, Fasc.2, ISSN 1454 – 6531, 2000
47. C. Pănoiu, A. Iagăr, C. Abrudean, *The Functioning Simulation of an Electrical Installation Afferent to a Crucible Induction Furnace, By Using the PSCAD EMTDC Program*, VIIth International Symposium Interdisciplinary Regional Research – ISIRR 2003, pag. 462-467, 2003
48. A. Iagăr, C. Abrudean, *Numerical computation of the resistance furnaces masonry*, SIELMEN 2007, 6th International Conference on Electromechanical and Power Systems, Chişinău, Rep. Moldova, ISSN 1842-4805, pp. 408-411, 2007
49. A. Iagăr, C. Abrudean, C. Diniş, C. Pănoiu, *Research Upon Optimization of the Volume Induction Heating Process*, Metalurgia International, vol. XIII (2008), no. 9, pag. 27, 2008
50. A. Iagăr, I. Şora, D. Radu, C. Pănoiu, C. Abrudean, *Technological Practicability of the Numerical Modeling of Induction Heating Process in Steel Pieces*, Revista de metalurgia, ISSN 0034-8570, Vol. 45, No 1, 2009, pag. 20-31, 2009
51. C. Abrudean, I. Şora, M. Panoiu, *Numerical Simulation of Electrothermal Processes With Phase Transitions in Materials With Nonlinear Thermal Characteristics*, 6th International Conference: Days of the Academy of Technical Science from Romania, Timişoara, 22-23 septembrie 2011, ISBN: 2066-6586, pag. 26-32, 2011

ANEXE

A1. Diagrame LabView

Diagrama LabView pentru aplicaţia de achiziţie 20 secunde cu mărimi pe magnetron.

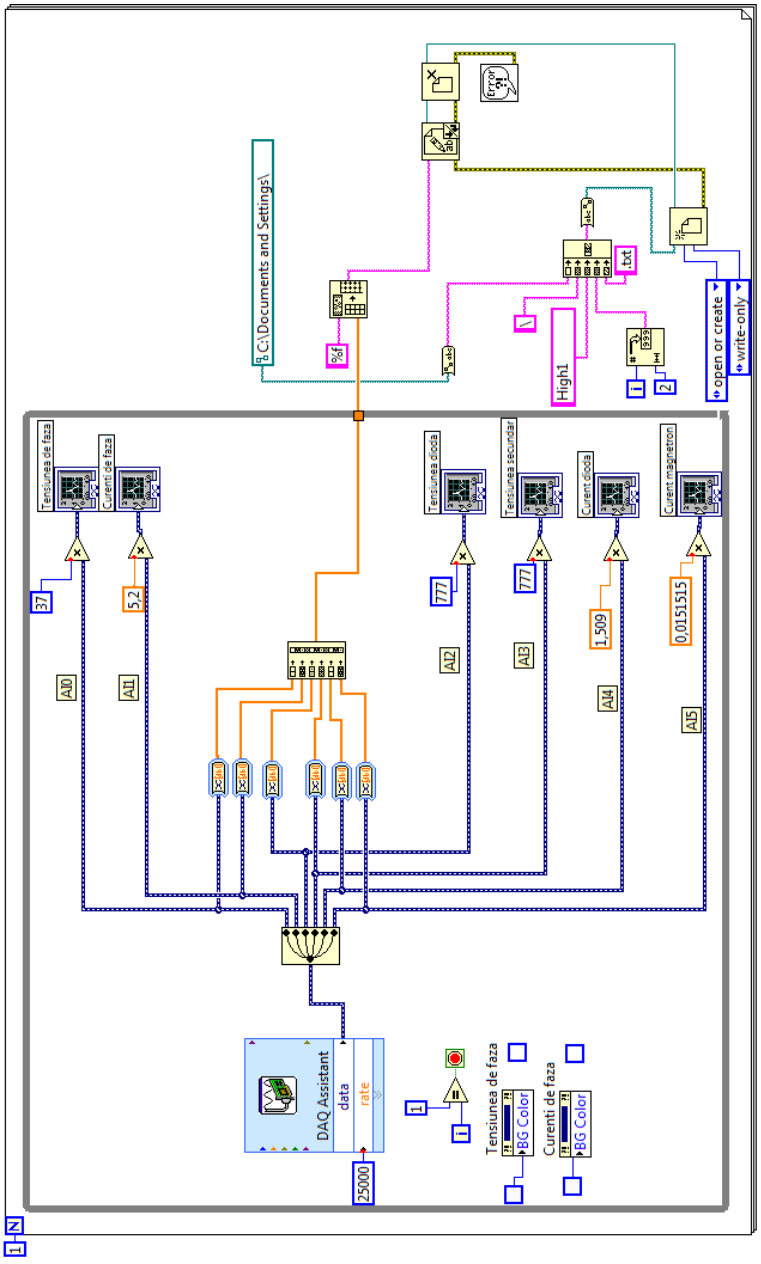


Diagrama LabView pentru aplicaţia de achiziţie 20 secunde cu mărimi pe primarul transformatorului.

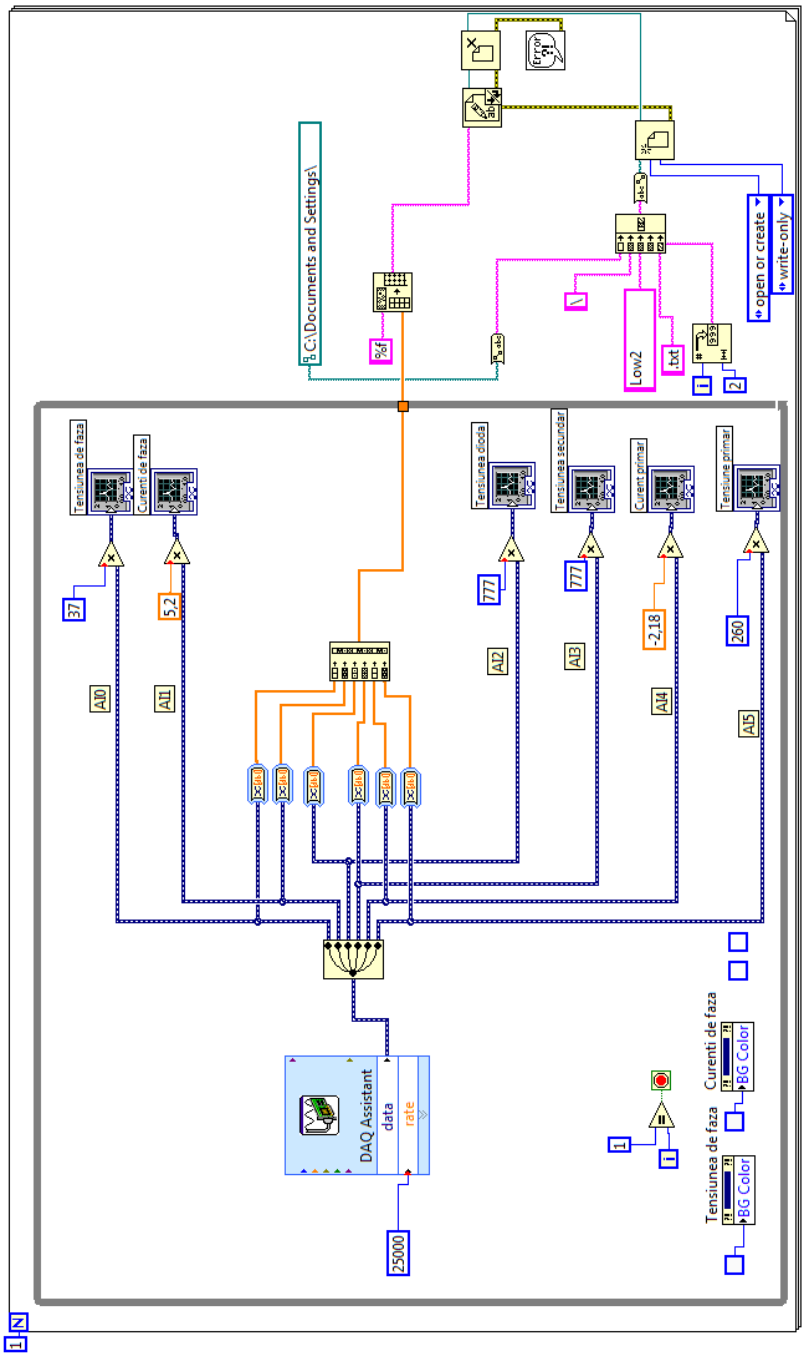
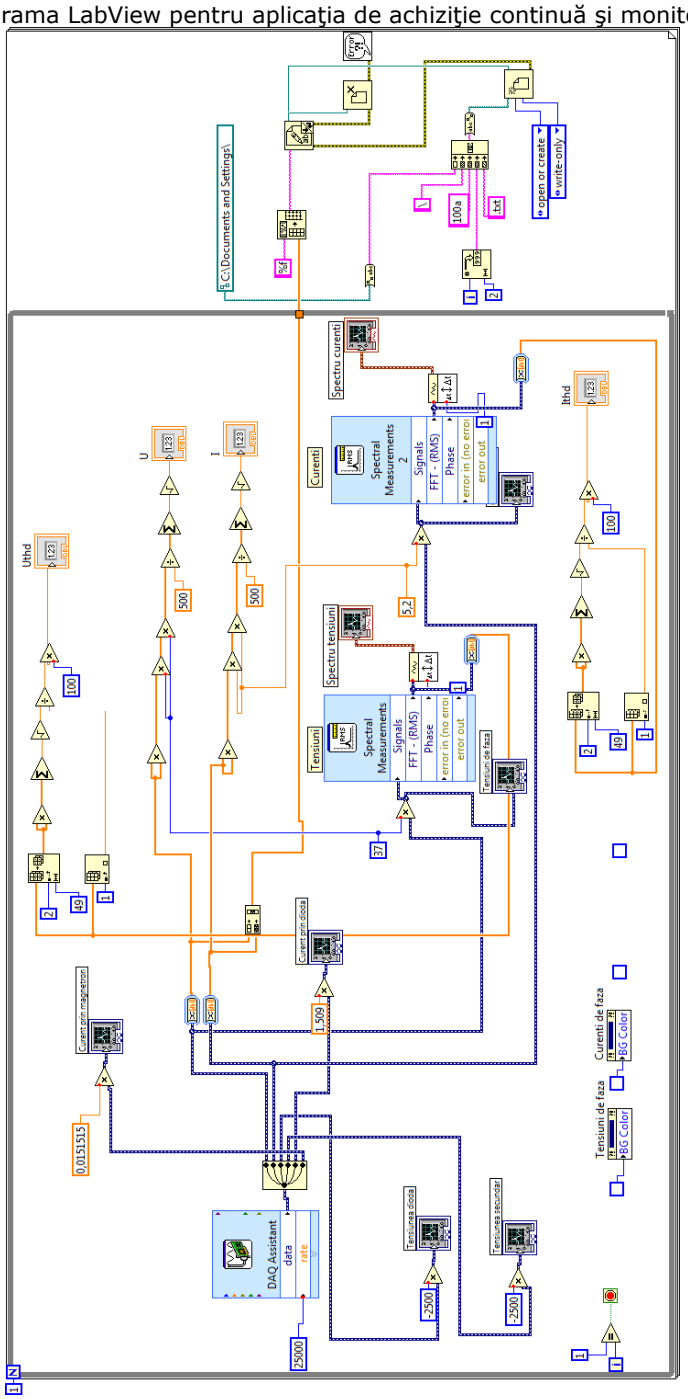


Diagrama LabView pentru aplicaţia de achiziţie continuă şi monitorizare.



A2. Programul mwel (Java)

Fişierul Main.java

```
package mwel;
import java.io.*;

public class Main {
    public static File INDIR
        = new File("d:\\Masuratori Cristi 10.02\\Alte 20 secunde");
    public static float[] FACTS = new float[] {
        37, 5.2f, 777, 777, 1.509f, 0.0151515f
    };
    public static void main(String[] args) throws Exception {
        //prepareFiles();
        new MainFrame();
    }
    private static void prepareFiles() throws Exception {
        if (!INDIR.isDirectory()) return;
        for (File f : INDIR.listFiles()) {
            if (!f.getName().toLowerCase().endsWith(".txt")) continue;
            if (f.getName().toLowerCase().startsWith("test")) continue;
            System.out.println("File: " + f);
            new DataFile(f, FACTS);
        }
    }
}
```

Fişierul DataFile.java

```
package mwel;
import java.io.*;
import java.util.*;
import java.util.zip.*;

public class DataFile {
    public final static int VERSION = 3;
    public final static double SFREQ = 25000;
    public final static int NALTSPEC = 5;

    private File infile, floatfile = null, specfile = null;
    private int samples, channels, specs;
    private float[][] data;
    private float[] facts;
    private double f0;
    private Spectrum[][] spec;

    public int countChannels() {return channels;}
    public int countSamples() {return samples;}
    public int countSpectrums() {return specs;}
    public float getSample(int channel, int sample) {
        return data[channel][sample];
    }
    public double getF0() {return f0;}
    public Spectrum getSpectrum(int channel, int index) {
        return spec[channel][index];
```

```
    }

    public DataFile(File file, float[] facts) throws IOException {
        this.infile = file;
        this.facts = facts;
        channels = facts.length;
        boolean flag = parseFloatFile();
        if (!flag) {
            System.out.println("parseFloatFile failed");
            parseInputFile();
            writeFloatFile();
        }
        if (flag) {
            flag = parseSpecFile();
            if (!flag) System.out.println("parseSpecFile failed");
        }
        if (!flag) {
            calculateSpectrums();
            writeSpecFile();
        }
    }
    private void writeSpecFile() throws IOException {
        DataOutputStream os = null;
        try {
            os = new DataOutputStream(new BufferedOutputStream(
                new FileOutputStream(getSpecFile())));
            os.writeInt(VERSION);
            os.writeDouble(SFREQ);
            os.writeLong(infile.lastModified());
            os.writeLong(infile.length());
            os.writeInt(channels);
            os.writeInt(specs);
            os.writeDouble(f0);
            for (float x : facts) os.writeFloat(x);
            for (Spectrum[] sa : spec) {
                for (Spectrum s : sa) s.toFile(os);
            }
        } finally {
            try {os.close();} catch (Exception e) {}
        }
    }
    private boolean parseSpecFile() {
        if (!getSpecFile().isFile()) return false;
        DataInputStream is = null;
        try {
            is = new DataInputStream(new BufferedInputStream(
                new FileInputStream(getSpecFile())));
            if (is.readInt() != VERSION) return false;
            if (is.readDouble() != SFREQ) return false;
            if (is.readLong() != infile.lastModified()) return false;
            if (is.readLong() != infile.length()) return false;
            if (is.readInt() != channels) return false;
            specs = is.readInt();
            f0 = is.readDouble();
            int i, j;
            for (i = 0; i < channels; i++) {
                if (facts[i] != is.readFloat()) return false;
            }
            spec = new Spectrum[channels][specs];
            for (i = 0; i < channels; i++) {
                for (j = 0; j < specs; j++) {
```

```
                    spec[i][j] = Spectrum.fromFile(is);
                }
            }
            return true;
        } catch (Exception e) {
            return false;
        } finally {
            try {is.close();} catch (Exception e) {}
        }
    }
    private void calculateSpectrums() {
        f0 = calculateRealFrequency(0, 50);
        int nsp = (int)(NALTSPEC * SFREQ / 50); // esantioane pe spectru
        specs = samples / nsp;
        System.out.printf("samples=%d, specs=%d, nsp=%d\n", samples, specs,
nsp);
        spec = new Spectrum[channels][specs];
        int i, j;
        for (i = 0; i < channels; i++) {
            for (j = 0; j < specs; j++) {
                spec[i][j] = Spectrum.fromData((j * nsp) / SFREQ, data[i],
                    j * nsp, nsp, SFREQ, f0);
            }
        }
    }
    private File getSpecFile() {
        if (specfile != null) return specfile;
        String fn = infile.getName();
        int k = fn.indexOf('.');
        if (k != -1) fn = fn.substring(0, k);
        fn += ".spec";
        File file = infile.getParentFile();
        if (file == null) specfile = new File(fn);
        else specfile = new File(file, fn);
        return specfile;
    }
    private void parseInputFile() throws IOException {
        InputStream is = null;
        try {
            is = new FileInputStream(infile);
            if (infile.getName().toLowerCase().endsWith(".txt.gz")) {
                is = new GZIPInputStream(is);
            }
            is = new BufferedInputStream(is);
            ArrayList<Float> list = new ArrayList<>();
            ByteArrayOutputStream buf = new ByteArrayOutputStream();
            int k; float x;
            while ((k = is.read()) != -1) {
                if (Character.isWhitespace((char)k)) {
                    if (buf.size() == 0) continue;
                    list.add(Float.parseFloat(buf.toString().replace(',',
'.')));
                    buf.reset();
                } else buf.write(k);
            }
            if (buf.size() != 0) {
                list.add(Float.parseFloat(buf.toString().replace(',', '.')));
            }
            samples = list.size() / channels;
            int kmax = channels * samples;
            data = new float[channels][samples];
```

```
            Iterator<Float> it = list.iterator();
            for (k = 0; k < kmax; k++) {
                int chn = k % channels;
                x = facts[chn] * it.next();
                data[chn][k / channels] = x;
            }
        } finally {
            try {is.close();} catch (Exception e) {}
        }
    }
    private File getFloatFile() {
        if (floatfile != null) return floatfile;
        String fn = infile.getName();
        int k = fn.indexOf('.');
        if (k != -1) fn = fn.substring(0, k);
        fn += ".float";
        File file = infile.getParentFile();
        if (file == null) floatfile = new File(fn);
        else floatfile = new File(file, fn);
        return floatfile;
    }
    private void writeFloatFile() throws IOException {
        DataOutputStream os = null;
        try {
            os = new DataOutputStream(new BufferedOutputStream(
                new FileOutputStream(getFloatFile())));
            os.writeInt(VERSION);
            os.writeDouble(SFREQ);
            os.writeLong(infile.lastModified());
            os.writeLong(infile.length());
            os.writeInt(channels);
            os.writeInt(samples);
            for (float x : facts) os.writeFloat(x);
            for (float[] xa : data) {
                for (float x : xa) os.writeFloat(x);
            }
        } finally {
            try {os.close();} catch (Exception e) {}
        }
    }
    private boolean parseFloatFile() {
        if (!getFloatFile().isFile()) return false;
        DataInputStream is = null;
        try {
            is = new DataInputStream(new BufferedInputStream(
                new FileInputStream(getFloatFile())));
            if (is.readInt() != VERSION) return false;
            if (is.readDouble() != SFREQ) return false;
            if (is.readLong() != infile.lastModified()) return false;
            if (is.readLong() != infile.length()) return false;
            if (is.readInt() != channels) return false;
            samples = is.readInt();
            int i, j;
            for (i = 0; i < channels; i++) {
                if (facts[i] != is.readFloat()) return false;
            }
            data = new float[channels][samples];
            for (i = 0; i < channels; i++) {
                for (j = 0; j < samples; j++) {
                    data[i][j] = is.readFloat();
                }
```

```
            }
            return true;
        } catch (Exception e) {
            return false;
        } finally {
            try {is.close();} catch (Exception e) {}
        }
    }
    private double calculateRealFrequency(int ch, double startf) {
        double df = 0.1;
        double f = startf, c, oldc;
        oldc = c = calculateHarmAmpl(ch, f);
        while (Math.abs(df) > 1.0e-6) {
            do {
                oldc = c; f += df;
                c = calculateHarmAmpl(ch, f);
            } while (c >= oldc);
            df = - df / 2;
        }
        return f;
    }
    private double calculateHarmAmpl(int ch, double f) {
        double omi = 2 * Math.PI * f / SFREQ;
        double a, b, s, a1, b1;
        int i, nmax = data[ch].length, pnmax = nmax - 1;
        double T = nmax / SFREQ;
        a1 = data[ch][0] / 2; b1 = 0;
        for (i = 1; i < pnmax; i++) {
            s = data[ch][i];
            a = s * Math.cos(omi * i);
            b = s * Math.sin(omi * i);
            a1 += a; b1 += b;
        }
        s = data[ch][i];
        a = s * Math.cos(omi * i);
        b = s * Math.sin(omi * i);
        a1 += a / 2; b1 += b / 2;
        a1 /= SFREQ; b1 /= SFREQ;
        return (2 / T) * Math.sqrt(a1 * a1 + b1 * b1);
    }
}
```

Fişierul Spectrum.java

```
package mwel;
import java.io.*;

public class Spectrum {
    public final static int NH = 41;
    public final static double SQRT2 = Math.sqrt(2);
    private double time;
    private double[] c, phi;
    private double THD, THDp;

    protected Spectrum(double time) {
        this.time = time;
        c = new double[NH];
        phi = new double[NH];
    }
    public static Spectrum fromData(double time, float[] data,
```

```
int start, int length, double sfreq, double f0) {
    Spectrum s = new Spectrum(time);
    s.calculateHarmonic0(data, start, length, sfreq);
    int h;
    for (h = 1; h < NH; h++) {
        s.calculateHarmonic(h, data, start, length, sfreq, f0);
    }
    double thds = 0, thdps = 0;
    for (h = 2; h < NH; h++) {
        thds += s.c[h] * s.c[h];
        thdps += s.c[h] * s.c[h] * h;
    }
    s.THD = Math.sqrt(thds) / s.c[1];
    s.THDp = Math.sqrt(thdps) / s.c[1];
    return s;
}
public static Spectrum fromFile(DataInputStream is) throws IOException {
    Spectrum s = new Spectrum(is.readDouble());
    int k; s.phi[0] = 0;
    for (k = 0; k < NH; k++) s.c[k] = is.readDouble();
    for (k = 1; k < NH; k++) s.phi[k] = is.readDouble();
    s.THD = is.readDouble(); s.THDp = is.readDouble();
    return s;
}
public void toFile(DataOutputStream os) throws IOException {
    int k;
    os.writeDouble(time);
    for (k = 0; k < NH; k++) os.writeDouble(c[k]);
    for (k = 1; k < NH; k++) os.writeDouble(phi[k]);
    os.writeDouble(THD); os.writeDouble(THDp);
}
public double getTime() {return time;}
public double getC(int h) {return c[h];}
public double getCeff(int h) {return c[h] / SQRT2;}
public double getPhi(int h) {return phi[h];}
public double getTHD() {return THD;}
public double getTHDp() {return THDp;}
public int getNH() {return NH;}
private void calculateHarmonic(int h, float[] data, int start, int length,
double sfreq, double f0) {
    if (h <= 0 || h >= NH) throw new AssertionError();
    double omi = 2 * Math.PI * f0 * h / sfreq;
    double a, b, s, ai, bi;
    int i, nmax = start + length, pnmax = nmax - 1;
    double T = length / sfreq;
    ai = data[start] / 2; bi = 0;
    for (i = start + 1; i < pnmax; i++) {
        s = data[i];
        a = s * Math.cos(omi * i);
        b = s * Math.sin(omi * i);
        ai += a; bi += b;
    }
    s = data[i];
    a = s * Math.cos(omi * i);
    b = s * Math.sin(omi * i);
    ai += a / 2; bi += b / 2;
    ai *= (2 / T) / sfreq; bi *= (2 / T) / sfreq;
    c[h] = Math.sqrt(ai * ai + bi * bi);
    phi[h] = Math.atan2(ai, bi);
}
private void calculateHarmonic0(float[] data, int start, int length,
```

```
    double sfreq) {
        int i, nmax = start + length, pnmax = nmax - 1;
        double T = length / sfreq;
        double a = data[start] / 2;
        for (i = start + 1; i < pnmax; i++) a += data[i];
        a += data[i] / 2;
        a *= (1 / T) / sfreq;
        c[0] = a; phi[0] = 0;
    }
}
```

Fişierul MainFrame.java

```
package mwel;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.filechooser.*;
import mwel.graph.*;

public class MainFrame extends JFrame {
    JFileChooser fileChooser;
    File file  = null;
    PowerGraph powerGraph;
    HarmonicGraph harmonicGraph;
    UIGraph uiGraph;
    JLabel statusLabel;
    DataFile dataFile = null;

    public MainFrame() {
        setTitle("mwel");
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        if (Main.INDIR.isDirectory())
            fileChooser = new JFileChooser(Main.INDIR);
        else fileChooser = new JFileChooser();
        fileChooser.setFileFilter(
            new FileNameExtensionFilter("Text files", "txt"));
        setLayout(new BorderLayout(0, 3));
        JPanel upPanel = new JPanel(new BorderLayout(3, 3));
        JButton fileButton = new JButton("File");
        fileButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                if (fileChooser.showOpenDialog(MainFrame.this)
                    != JFileChooser.APPROVE_OPTION) return;
                file = fileChooser.getSelectedFile();
                try {
                    statusLabel.setText("wait ...");
                    dataFile = new DataFile(file, Main.FACTS);
                    powerGraph.setDataFile(dataFile);
                    harmonicGraph.setDataFile(dataFile, 1, 0, true);
                    uiGraph.setDataFile(dataFile, 0, 1, 0);
                    MainFrame.this.setTitle("mwel - " + file.getName());
                    if (harmonicGraph.isOverEN())
                        statusLabel.setText("Over EN!");
                    else statusLabel.setText(String.format(
                        "f0=%f t=0", dataFile.getF0()));
```

```
                } catch (Exception e) {
                    e.printStackTrace();
                    statusLabel.setText("Error!");
                }
            }
        });
        statusLabel = new JLabel("open a file ...");
        upPanel.add(fileButton, BorderLayout.WEST);
        upPanel.add(statusLabel, BorderLayout.CENTER);
        add(upPanel, BorderLayout.NORTH);
        JPanel centerPanel = new JPanel(new GridLayout(3, 1));

        powerGraph = new PowerGraph();
        uiGraph = new UIGraph();
        harmonicGraph = new HarmonicGraph();
        centerPanel.add(powerGraph);
        centerPanel.add(uiGraph);
        centerPanel.add(harmonicGraph);

        add(centerPanel, BorderLayout.CENTER);
        powerGraph.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent evt) {
                Point p = evt.getPoint();
                harmonicGraph.setDataFile(dataFile, 1, powerGraph.getX(p.x),
true);
                uiGraph.setDataFile(dataFile, 0, 1, powerGraph.getX(p.x));
                if (harmonicGraph.isOverEN()) statusLabel.setText("Over EN!");
                else statusLabel.setText(String.format("f0=%f t=%f",
                    dataFile.getF0(), powerGraph.getX(p.x)));
            }
        });

        Dimension scs = Toolkit.getDefaultToolkit().getScreenSize();
        int winw = Math.min(600, scs.width);
        int winh = Math.min(800, scs.height - 60);
        setSize(winw, winh);
        setLocation((scs.width - winw) / 2, (scs.height - winh - 30) / 2);
        setVisible(true);
    }
}
```

Fişierul graph/AbstractGraph.java

```
package mwel.graph;
import java.awt.*;
import javax.swing.*;

public abstract class AbstractGraph extends JPanel {
    public final static int BORDER = 5;
    protected final static int HIN = 50;

    protected Rectangle rtot, rtin;
    protected double xmin, xmax, ymin, ymax;
    protected double px, py;
    protected int hin, vin;

    protected AbstractGraph() {
        super();
        setDoubleBuffered(true);
        rtot = rtin = null;
```

```
        xmin = xmax = ymin = ymax = 0;
        px = py = 0; hin = vin = 0;
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.white);
        rtot = new Rectangle(BORDER, BORDER,
            getWidth() - 2 * BORDER, getHeight() - 2 * BORDER);
        g.fillRect(rtot.x, rtot.y, rtot.width, rtot.height);
        g.setClip(rtot.x, rtot.y, rtot.width, rtot.height);
        g.setColor(Color.lightGray);
    }
    protected abstract void calculateLimits();
    protected int transx(double x) {
        return (int)(rtin.x + rtin.width * x / (xmax - xmin));
    }
    protected int transy(double y) {
        return (int)((rtin.y + rtin.height) - rtin.height *
                (y - ymin) / (ymax - ymin));
    }
    protected static String d2s(double x) {
        if (x == 0) return "0";
        if (Math.abs(x - Math.round(x)) < 1e-3) x = Math.round(x);
        if (x - (int)x == 0) return String.valueOf((int)x);
        String s = String.format("%s", x).replace('.', ',');
        return s.replaceFirst("0{4,}\\d$", "");
    }
    protected double autoScaleY(double ymin, double ymax) {
        double ym = ymax - ymin;
        if (ym == 0) return 0;
        double yl = Math.pow(10, Math.floor(Math.log10(ym)));
        double ylm = ym / yl;
        if (ylm < 2) return yl / 5;
        else if (ylm < 4) return yl / 2;
        else return yl;
    }
}
```

Fişierul graph/PowerGraph.java

```
package mwel.graph;

import java.awt.*;
import mwel.DataFile;

public class PowerGraph extends AbstractGraph {
    private double[] S, P, Q, D;
    private double dt;

    public PowerGraph() {
        super();
        S = P = Q = D = null;
    }
    public void setDataFile(DataFile df) {
        dt = DataFile.NALTSPEC * 0.02;
        int specs = df.countSpectrums();
        S = new double[specs]; P = new double[specs];
        Q = new double[specs]; D = new double[specs];
        int s, h;
        for (s = 0; s < specs; s++) {
```

```
            mwel.Spectrum us = df.getSpectrum(0, s);
            mwel.Spectrum is = df.getSpectrum(1, s);
            double S1 = Math.abs(us.getC(1) * is.getC(1)) / 2;
            P[s] = us.getC(0) * is.getC(0); Q[s] = 0;
            double U = us.getC(0) * us.getC(0), I = is.getC(0) * is.getC(0);
            for (h = 1; h < us.getNH(); h++) {
                U += us.getC(h) * us.getC(h);
                I += is.getC(h) * is.getC(h);
                P[s] += 0.5 * us.getC(h) * is.getC(h)
                    * Math.cos(us.getPhi(h) - is.getPhi(h));
                Q[s] += 0.5 * us.getC(h) * is.getC(h)
                    * Math.sin(us.getPhi(h) - is.getPhi(h));
            }
            U = Math.sqrt(U / 2); I = Math.sqrt(I / 2);
            S[s] = U * I;
            D[s] = Math.sqrt(S[s] * S[s] - S1 * S1);
        }
        repaint();
    }
    protected void calculateLimits() {
        if (S == null) return;
        xmin = 0; xmax = dt * S.length;
        int i;
        ymin = 0; ymax = S[0];
        for (i = 0; i < S.length; i++) {
            if (ymin > S[i]) ymin = S[i];
            if (ymin > P[i]) ymin = P[i];
            if (ymin > Q[i]) ymin = Q[i];
            if (ymin > D[i]) ymin = D[i];
            if (ymax < S[i]) ymax = S[i];
            if (ymax < P[i]) ymax = P[i];
            if (ymax < Q[i]) ymax = Q[i];
            if (ymax < D[i]) ymax = D[i];
        }
    }
    public double getX(int h) {
        if (rtin == null) return 0;
        double x = (h - hin) * (xmax - xmin) / rtin.width;
        if (x < xmin) return xmin;
        if (x > xmax) return xmax;
        return x;
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (S == null) return;
        calculateLimits();
        double[][] ya = new double[][] {S, P, Q, D};
        Color[] ca = new Color[] {Color.black, Color.blue, Color.green,
Color.red};
        int k, i; double x1, x2, x, y;
        rtin = rtot;
        hin = HIN; vin = 15;
        FontMetrics fm = g.getFontMetrics();
        px = 4;
        rtin.height -= fm.getHeight() + 5 + vin;
        rtin.y = vin;
        py = autoScaleY(ymin, ymax);
        rtin.x = rtot.x + hin;
        rtin.width -= hin + 10;
        for (x = 0; x <= xmax; x += px) {
            g.setColor(Color.lightGray);
```

```
            g.drawLine(transx(x), transy(ymin), transx(x), transy(ymax));
            g.setColor(Color.black);
            String s = d2s(x);
            g.drawString(s, transx(x)
                - ((int)fm.getStringBounds(s, g).getWidth()) / 2,
                transy(ymin) + 14);
        }
        for (y = 0; y <= ymax; y += py) {
            g.setColor(Color.lightGray);
            g.drawLine(transx(xmin), transy(y), transx(xmax), transy(y));
            g.setColor(Color.black);
            String s = d2s(y);
            g.drawString(s, rtin.x
                - ((int)fm.getStringBounds(s, g).getWidth()) - 5,
                transy(y) + 5);
        }

        for (k = 0; k < 4; k++) {
            g.setColor(ca[k]);
            for (i = 1; i < S.length; i++) {
                x1 = (i - 1) * dt; x2 = i * dt;
                g.drawLine(transx(x1), transy(ya[k][i-1]),
                    transx(x2), transy(ya[k][i]));
            }
        }
    }
}
```

Fişierul graph/UIGraph.java

```
package mwel.graph;

import java.awt.*;
import mwel.DataFile;

public class UIGraph extends AbstractGraph {
    private final static int NS = 1000;
    private double[] ua, ia;
    private double y2max;
    public UIGraph() {
        super();
        ua = ia = null; y2max = 0;
    }
    public void setDataFile(DataFile df, int chu, int chi, double time) {
        int i, n = (int)(time * DataFile.SFREQ);
        if (n < 0) n = 0;
        if (n > df.countSamples() - NS) n = df.countSamples() - NS;
        ua = new double[NS]; ia = new double[NS];
        for (i = 0; i < NS; i++) {
            ua[i] = df.getSample(chu, i + n);
            ia[i] = df.getSample(chi, i + n);
        }
        repaint();
    }
    protected void calculateLimits() {
        if (ua == null) return;
        xmin = 0; xmax = NS / DataFile.SFREQ;
        int i; double v;
        ymax = Math.abs(ua[0]);
        y2max = Math.abs(ia[0]);
```

```
        for (i = 1; i < NS; i++) {
            v = Math.abs(ua[i]);
            if (ymax < v) ymax = v;
            v = Math.abs(ia[i]);
            if (y2max < v) y2max = v;
        }
        ymin = -ymax;
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (ua == null) return;
        calculateLimits();
        rtin = rtot;
        hin = HIN; vin = 15;
        FontMetrics fm = g.getFontMetrics();
        px = xmax / 8;
        rtin.height -= fm.getHeight() + 5 + vin;
        rtin.y = vin;
        py = autoScaleY(ymin, ymax);
        rtin.x = rtot.x + hin;
        rtin.width -= 2 * hin;
        double py2 = autoScaleY(-y2max, y2max);
        if (Math.floor(ymax / py) * py2 < y2max) py2 *= 2;
        y2max = py2 * ymax / py;
        double x, y;
        for (x = 0; x <= xmax; x += px) {
            g.setColor(Color.lightGray);
            g.drawLine(transx(x), rtin.y, transx(x), rtin.y + rtin.height);
            g.setColor(Color.black);
            String s = d2s(x * 1000);
            g.drawString(s, transx(x)
                - ((int)fm.getStringBounds(s, g).getWidth()) / 2,
                rtin.y + rtin.height + 14);
        }
        for (y = 0; y <= ymax; y += py) {
            g.setColor(Color.lightGray);
            g.drawLine(rtin.x, transy(y), rtin.x + rtin.width, transy(y));
            g.drawLine(rtin.x, transy(-y), rtin.x + rtin.width, transy(-y));
            g.setColor(Color.black);
            String s = d2s(y);
            g.drawString(s, rtin.x
                - ((int)fm.getStringBounds(s, g).getWidth()) - 5,
                transy(y) + 5);
            if (y != 0) g.drawString("-" + s, rtin.x
                - ((int)fm.getStringBounds(s, g).getWidth()) - 5,
                transy(-y) + 5);
            s = d2s(y2max * y / ymax);
            g.drawString(s, rtin.x + rtin.width + 5, transy(y) + 5);
            if (y != 0) g.drawString("-" + s,
                rtin.x + rtin.width + 5, transy(-y) + 5);
        }
        for (int i = 0; i < NS - 1; i++) {
            x = xmax * i / NS;
            g.setColor(Color.blue);
            g.drawLine(transx(x), transy(ua[i]),
                transx(xmax * (i + 1) / NS), transy(ua[i + 1]));
            g.setColor(Color.red);
            g.drawLine(transx(x), transy2(ia[i]),
                transx(xmax * (i + 1) / NS), transy2(ia[i + 1]));
        }
    }
```

```
    protected int transy2(double y) {
        return (int)((rtin.y + rtin.height) - rtin.height *
                (y + y2max) / (2 * y2max));
    }
}
```

Fişierul graph/HarmonicGraph.java

```
package mwel.graph;

import java.awt.*;
import mwel.DataFile;
import mwel.Spectrum;

public class HarmonicGraph extends AbstractGraph {
    private final double[] EN = new double[] {
        0, 0, 1.08, 2.3, 0.43, 1.14, 0.3, 0.77, 0.23, 0.4, 0.18, 0.33, 0.15,
        0.21, 0.13, 0.15, 0.11, 0.13, 0.1, 0.11, 0.09, 0.1, 0.08, 0.09, 0.07,
        0.09, 0.07, 0.08, 0.06, 0.07, 0.06, 0.07, 0.05, 0.06, 0.05, 0.06, 0.05,
        0.06, 0.04, 0.05, 0.04
    };
    Spectrum sp;
    boolean useEN;

    public HarmonicGraph() {
        super();
        sp = null; useEN = false;
    }
    public void setDataFile(DataFile df, int channel, double time, boolean
useEN) {
        this.useEN = useEN;
        int i, specs = df.countSpectrums();
        for (i = 0; i < specs; i++) {
            sp = df.getSpectrum(channel, i);
            if (sp.getTime() >= time) break;
        }
        repaint();
    }
    protected void calculateLimits() {
        if (sp == null) return;
        xmin = 0; xmax = sp.getNH();
        int i;
        ymin = 0; ymax = sp.getCeff(2);
        if (useEN) ymax = Math.max(ymax, 2.3);
        for (i = 2; i < sp.getNH(); i++) {
            if (ymax < sp.getCeff(i)) ymax = sp.getCeff(i);
        }
    }
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (sp == null) return;
        calculateLimits();
        rtin = rtot;
        hin = HIN; vin = 15;
        FontMetrics fm = g.getFontMetrics();
        px = 5;
        rtin.height -= fm.getHeight() + 5 + vin;
        rtin.y = vin;
        py = autoScaleY(ymin, ymax);
        rtin.x = rtot.x + hin;
```

```
        rtin.width -= hin + 10;

        double y;
        for (y = 0; y <= ymax; y += py) {
            g.setColor(Color.lightGray);
            g.drawLine(rtin.x, transy(y), rtin.x + rtin.width, transy(y));
            g.setColor(Color.black);
            String s = d2s(y);
            g.drawString(s, rtin.x
                - ((int)fm.getStringBounds(s, g).getWidth()) - 5,
                transy(y) + 5);
        }
        g.setColor(Color.black);
        g.drawRect(rtin.x, rtin.y, rtin.width, rtin.height);
        int v, i, bwidth = rtin.width / (sp.getNH() - 3) - 2;
        for (i = 2; i < sp.getNH(); i++) {
            int h1 = rtin.x + (i - 2) * rtin.width / (sp.getNH() - 2);
            if (i % 5 == 0) {
                String s = String.valueOf(i);
                g.setColor(Color.black);
                g.drawString(s, h1, rtin.y + rtin.height + 14);
            }
            if (useEN) {
                g.setColor(Color.green);
                v = transy(EN[i]);
                g.fillRect(h1, v, bwidth, vin + rtin.height - v);
            }
            g.setColor(Color.blue);
            v = transy(sp.getCeff(i));
            g.fillRect(h1, v, bwidth, vin + rtin.height - v);
        }
    }
    public boolean isOverEN() {
        for (int i = 2; i < sp.getNH(); i++) {
            if (sp.getCeff(i) > EN[i]) return true;
        }
        return false;
    }
}
```

A3. Programul mwelim (Java)

Fişierul Main.java

```
package mwelim;
import java.io.*;

public class Main {
    private final static int SKIP_SAMPLES = 125000 + 0;

    private static double readDouble(InputStream is) throws IOException {
        ByteArrayOutputStream buf = new ByteArrayOutputStream(16);
        int ch;
        while ((ch = is.read()) != 9) {
            if (ch == -1) throw new IllegalStateException();
            buf.write(ch);
```

```
        }
        if (buf.size() < 3) throw new IllegalStateException();
        return Double.parseDouble(buf.toString().replace(',', '.'));
    }
    public static void main(String[] args) throws Exception {
        ByteArrayOutputStream buf = new ByteArrayOutputStream();
        InputStream is = new BufferedInputStream(new FileInputStream(
            "d:\\Masuratori Cristi 10.02\\imag-bun.txt"));
        int i, ch;
        for (i = 6 * SKIP_SAMPLES; i > 0; ) {
            ch = is.read();
            if (ch == -1) throw new IllegalStateException();
            if (ch == 9) i--;
        }
        double pm = 0, pa = 0, pax = 0;
        Alt au = new Alt(500), ai = new Alt(500);
        for (i = 0; i < 6 * 500; i++) {
            double x = readDouble(is);
            switch (i % 6) {
                default: break;
                case 0: pax = x * 37; break;
                case 1: pax *= x * 5.2; pa += pax; break;
                case 2:
                    x *= 2500;
                    au.set(i / 6, x);
                    break;
                case 4:
                    x *= -0.181;
                    ai.set(i / 6, x);
                    break;
            }
        }
        is.close();

        for (i = 0; i < 500; i++) {
            pm += au.get(i) * ai.get(i);
        }
        pa /= 500; pm /= 500;
        System.out.println("Pa = " + pa + " W");
        System.out.println("Pm = " + pm + " W");

        new MainFrame(au, ai);
    }
}
```

Fişierul Alt.java

```
package mwelim;
import java.util.*;

public class Alt {
    private double[] v;
    private double[] limits = null;
    public Alt(int size) {
        v = new double[size];
    }
    public int getCount() {return v.length;}
    public double get(int index) {return v[index];}
    public void set(int index, double x) {v[index] = x;}
    public double interpolate(double t) {
```

```
        t -= 0.02 * Math.floor(t / 0.02);
        int vl = v.length;
        int i = (int)(vl * t / 0.02);
        double t1 = (0.02 * i) / vl;
        double t2 = (0.02 * ((i + 1) % vl)) / vl;
        double v1 = v[i], v2 = v[(i + 1) % vl];
        return ((t2 - t) * v1 + (t - t1) * v2) / (t2 - t1);
    }
    public double[] getLimits() {
        if (limits != null) return limits;
        limits = new double[2];
        limits[0] = limits[1] = v[0];
        for (int i = 1; i < v.length; i++) {
            if (limits[0] > v[i]) limits[0] = v[i];
            if (limits[1] < v[i]) limits[1] = v[i];
        }
        return limits;
    }
    public void dump(java.io.PrintStream out) {
        for (int i = 0; i < v.length; i++) {
            out.printf("%f\t%f\n", (0.02 * i) / v.length, v[i]);
        }
    }
    private void level1() {
        double[] nv = new double[v.length];
        for (int k = 0; k < v.length; k++) {
            int km = (k == 0) ? (v.length - 1) : (k - 1);
            int kp = (k + 1) % v.length;
            nv[k] = (v[km] + v[kp]) / 2;
        }
        v = nv;
    }
    public void level(int steps) {
        for (int k = 0; k < steps; k++) level1();
    }
    public void clearNoise(int maxharm) {
        double[] A = new double[1 + maxharm];
        double[] B = new double[1 + maxharm];
        Arrays.fill(A, 0); Arrays.fill(B, 0);
        int k, h; double omegat;
        for (k = 0; k < v.length; k++) A[0] += v[k];
        A[0] /= v.length;
        for (h = 1; h <= maxharm; h++) {
            for (k = 0; k < v.length; k++) {
                omegat = k * Math.PI / 250.0;
                A[h] += v[k] * Math.cos(h * omegat);
                B[h] += v[k] * Math.sin(h * omegat);
            }
            A[h] *= 2.0 / v.length;
            B[h] *= 2.0 / v.length;
        }
        double[] nv = new double[v.length];
        for (k = 0; k < v.length; k++) {
            omegat = k * Math.PI / 250.0;
            nv[k] = A[0];
            for (h = 1; h <= maxharm; h++) {
                nv[k] += A[h] * Math.cos(h * omegat);
                nv[k] += B[h] * Math.sin(h * omegat);
            }
        }
        v = nv;
```

```
    }
}
```

Fişierul Magnetron.java

```
package mwelim;

public class Magnetron {
    private final Alt au, ai;
    private Alt ax;
    private final double R, Ud;
    private boolean calculated;
    private final int count;

    public Magnetron(Alt au, Alt ai, double R, double Ud) {
        if (au.getCount() != ai.getCount()) throw new AssertionError();
        this.au = au; this.ai = ai;
        count = ai.getCount();
        this.R = R; this.Ud = Ud;
        ax = new Alt(au.getCount());
        calculated = false;
    }
    private void calculate() {
        if (calculated) return;
        for (int k = 0; k < count; k++) {
            ax.set(k, au.get(k) > Ud ? (au.get(k) - Ud) / R : 0);
        }
        calculated = true;
    }
    public Alt getAlt() {
        calculate();
        return ax;
    }
    public double getDiff() {
        calculate();
        double s = 0, x;
        for (int k = 0; k < count; k++) {
            x = ai.get(k) - ax.get(k);
            s += x * x;
        }
        return Math.sqrt(s / count);
    }
    public double getR() {return R;}
    public double getUd() {return Ud;}
}
```

Fişierul MainFrame.java

```
package mwelim;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MainFrame extends JFrame {
    private final static int BORDER = 10;
    private final static double TMAX = 0.02;
    private class GraphPanel extends JPanel {
        private Rectangle rt = null;
```

```
        private Alt alt = null;
        private GraphPanel(Alt alt) {
            super();
            this.alt = alt;
            setDoubleBuffered(true);
        }
        private int transx(double t) {
            return (int)(rt.x + rt.width * t / TMAX);
        }
        private int transy(double v) {
            return (int)((rt.y + rt.height) - rt.height *
                (v - alt.getLimits()[0]) / (alt.getLimits()[1] -
alt.getLimits()[0]));
        }
        private void setAlt(Alt alt) {
            this.alt = alt;
            repaint();
        }
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            double t, oldt, v, oldv;
            int x, y; String str;
            g.setColor(Color.white);
            rt = new Rectangle(BORDER, BORDER,
                getWidth() - 2 * BORDER, getHeight() - 2 * BORDER);
            g.fillRect(rt.x, rt.y, rt.width, rt.height);
            g.setClip(rt.x, rt.y, rt.width, rt.height);
            g.setColor(Color.lightGray);
            for (t = 0; t < TMAX; t += 0.01) {
                x = transx(t);
                g.drawLine(x, rt.y, x, rt.y + rt.height);
            }
            v = Math.max(Math.abs(alt.getLimits()[0]),
Math.abs(alt.getLimits()[1]));
            if (v > 0) {
                oldv = Math.pow(10, Math.floor(Math.log10(v)));
                for (v = oldv; ; v += oldv) {
                    boolean flag = false;
                    if (v < alt.getLimits()[1]) {
                        flag = true; y = transy(v);
                        g.drawLine(rt.x, y, rt.x + rt.width, y);
                    }
                    if (-v > alt.getLimits()[0]) {
                        flag = true; y = transy(-v);
                        g.drawLine(rt.x, y, rt.x + rt.width, y);
                    }
                    if (!flag) break;
                }
            }
            g.setColor(Color.black);
            y = transy(0);
            if (y >= rt.y && y <= rt.y + rt.height) {
                g.drawLine(rt.x, y, rt.x + rt.width, y);
            }
            g.setColor(Color.red);
            FontMetrics fm = g.getFontMetrics();
            str = String.format("%.3f", alt.getLimits()[1]);
            g.drawString(str, rt.x + 10, rt.y + fm.getAscent() + 10);
            str = String.format("%.3f", alt.getLimits()[0]);
            g.drawString(str, rt.x + 10, rt.y + rt.height - fm.getDescent() -
10);
```

```
            g.setColor(Color.blue);
            oldt = 0; oldv = alt.interpolate(0);
            for (int i = 1; i <= 2000; i++) {
                t = i * TMAX / 2000;
                v = alt.interpolate(t);
                g.drawLine(transx(oldt), transy(oldv), transx(t), transy(v));
                oldt = t; oldv = v;
            }
        }
    }
    private class Worker extends Thread {
        private Worker() {
            setDaemon(true);
        }
        public void run() {
            double oldd, dR, dUd;
            int step = 0;
            while (worker != null) {
                Magnetron oldm = mag;
                dR = mag.getR() / 100;
                while (Math.abs(dR) > 1e-6) {
                    do {
                        oldd = mag.getDiff();
                        mag = new Magnetron(au, ai, mag.getR() + dR,
mag.getUd());
                    } while (mag.getDiff() < oldd);
                    dR = -dR / 2;
                }
                dUd = mag.getUd() / 100;
                while (Math.abs(dUd) > 1e-6) {
                    do {
                        oldd = mag.getDiff();
                        mag = new Magnetron(au, ai, mag.getR(), mag.getUd() +
dUd);
                    } while (mag.getDiff() < oldd);
                    dUd = -dUd / 2;
                }
                String text = String.format("Ud=%.2f R=%.2f diff=%f step=%d",
                    mag.getUd(), mag.getR(), mag.getDiff(), step++);
                statusLabel.setText(text);
                magPanel.setAlt(mag.getAlt());
                Thread.yield();
                if (Math.abs(oldm.getDiff() - mag.getDiff()) < 1e-9) worker =
null;
            }
            startButton.setText("Start");
        }
    }

    private Alt au = null, ai = null;
    private GraphPanel magPanel = null;
    private Magnetron mag;
    private Worker worker;
    private JLabel statusLabel;
    private JButton startButton;

    public MainFrame(Alt au, Alt ai) {
        worker = null;
        this.au = au; this.ai = ai;
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
```

```
                System.exit(0);
            }
        });
        setTitle("mwelim");
        setLayout(new BorderLayout());
        GraphPanel uPanel = new GraphPanel(au);
        GraphPanel iPanel = new GraphPanel(ai);
        JPanel centerPanel = new JPanel(new GridLayout(3, 1));
        centerPanel.add(uPanel);
        centerPanel.add(iPanel);
        mag = new Magnetron(au, ai, 560, 3700);
        magPanel = new GraphPanel(mag.getAlt());
        centerPanel.add(magPanel);
        add(centerPanel, BorderLayout.CENTER);

        JPanel lowPanel = new JPanel(new BorderLayout(3, 1));
        statusLabel = new JLabel("ready");
        lowPanel.add(statusLabel, BorderLayout.CENTER);
        startButton = new JButton("Start");
        lowPanel.add(startButton, BorderLayout.WEST);
        add(lowPanel, BorderLayout.SOUTH);

        startButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                if (worker == null) {
                    startButton.setText("Stop");
                    worker = new Worker(); worker.start();
                } else {
                    worker = null;
                }
            }
        });
        Dimension scs = Toolkit.getDefaultToolkit().getScreenSize();
        int winw = Math.min(600, scs.width);
        int winh = Math.min(600, scs.height - 60);
        setSize(winw, winh);
        setLocation((scs.width - winw) / 2, (scs.height - winh - 30) / 2);
        setVisible(true);
    }
}
```

A4. Programul de simulare a încălzirii în cuptor (C++)

Fişierul StdAfx.h

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if !defined(AFX_STDAFX_H__D7E99CBD_0C33_425C_B11C_63D82DD9704B__INCLUDED_)
#define AFX_STDAFX_H__D7E99CBD_0C33_425C_B11C_63D82DD9704B__INCLUDED_
#define WINVER 0x0501

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN    // Exclude rarely-used stuff from Windows headers
```

```
#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdtctl.h>      // MFC support for Internet Explorer 4 Common
Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>        // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

#pragma warning (disable: 4244 4305)
#include <stdio.h>
#include <io.h>
#include <math.h>
#include <float.h>
#include <direct.h>
#include <afxmt.h>

#define PAYLOAD 3

#if (PAYLOAD == 1)
   #define MAXTIME 60
#elif (PAYLOAD == 2)
   #define MAXTIME 30
#elif (PAYLOAD == 3)
   #define MAXTIME 60
#endif

#define EXCIT 800
#define DIVFACT 20
#define TDIV 100
#define INIT_STEPS 5000
#define STEPS 200
#define MAXTEMP 5000
#define MAXSTEPS 2000000

//Constante matematice
#define NAN (sqrt(-1.0)) //Not a Number
#define M_PI      3.141592653589793238
#define M_PI_2    1.570796326794896619
#define M_2_PI    6.283185307179586477
#define M_PI_180  0.017453292519943296
#define M_180_PI  57.29577951308232088
#define M_E       2.718281828459045235
#define M_1_E     0.367879441171442322

//Constante fizice
#define MIU0 1.2566370614359173e-6  //permeabilitatea magnetica a vidului
#define EPS0 8.8541878176e-12    //permitivitatea dielectrica a vidului
#define LIGHT 2.99792458e8       //viteza luminii in vid
#define J_KCAL 4185.5               //transformarea kcal -> J
#define STEFAN_BOLTZMANN 5.66961E-8 //constanta Stefan-Boltzmann
#define KELVIN 273.16

#define DEL(x) if (x) {delete x; x = NULL;}
#define MIN(x, y) ((x < y) ? (x) : (y))
#define MAX(x, y) ((x > y) ? (x) : (y))
#define ABS(x) ((x < 0) ? (-(x)) : (x))

#define THREAD_PRIORITY THREAD_PRIORITY_BELOW_NORMAL
CString getWorkDir();
```

```
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before
the previous line.

#endif //
!defined(AFX_STDAFX_H__D7E99CBD_0C33_425C_B11C_63D82DD9704B__INCLUDED_)
```

Fişierul Bitmap.h

```
#ifndef __BITMAP_H
#define __BITMAP_H

BOOL Save256ColorBitmap(HDC hdc,RECT& r,LPCTSTR filename);
BOOL SaveMonochromeBitmap(HDC hdc, RECT& r,
   LPCTSTR filename, COLORREF bkcolor);
BOOL SaveTrueColorBitmap(HDC hdc, RECT& r, LPCTSTR filename);
BOOL SaveTrueColorBitmap(HDC hdc, RECT& r, HBITMAP bmp, LPCTSTR filename);

#endif//__BITMAP_H
```

Fişierul Bitmap.cpp

```
#include "stdafx.h"
#include "buffers.h"

#include "utils.h"

BOOL BuildHistogram(HDC hdc, RECT& rect,
   COLORREF *buffer, int *ncolor) {
   int j,k,x,y,ncol=0;
   COLORREF c;
   BOOL changed;
   int *nhits=new int[256];
   if (!nhits) return FALSE;
   for (k = 0;k < 256;k++) {
      buffer[k] = CLR_INVALID; nhits[k] = 0;
   }
   for (x = rect.left; x < rect.right; x++) {
      for (y = rect.top; y < rect.bottom; y++) {
         c = GetPixel(hdc, x, y);
         for (k = 0;(k < ncol) && (buffer[k] != CLR_INVALID); k++)
            if (c == buffer[k]) {nhits[k]++; break;}
         if (c != buffer[k]) {ncol++; buffer[k] = c;}
      }
   }
   do { //sort...
      changed=FALSE;
      for (k=0;k<ncol-1;k++) {
         if (nhits[k]<nhits[k+1]) {
            changed=TRUE;
            j=nhits[k];nhits[k]=nhits[k+1];nhits[k+1]=j;
            c=buffer[k];buffer[k]=buffer[k+1];buffer[k+1]=c;
         }
      }
      for (k=ncol-1;k>0;k--) {
         if (nhits[k-1]<nhits[k]) {
            changed=TRUE;
            j=nhits[k];nhits[k]=nhits[k-1];nhits[k-1]=j;
            c=buffer[k];buffer[k]=buffer[k-1];buffer[k-1]=c;
```

```
         }
      }
   } while (changed);
   *ncolor=ncol; delete nhits;
   return TRUE;
}
BOOL Save256ColorBitmap(HDC hdc, RECT& r, LPCTSTR filename) {
   int x, y, dmx, k;
   int dimx = r.right - r.left, dimy = r.bottom - r.top;
   UCHAR val;
   COLORREF cp;
   int ncol;
   COLORREF *histo = new COLORREF[256];
   if (!histo) return FALSE;
   BuildHistogram(hdc, r, histo, &ncol);
   FILE * file = fopen(filename,"wb");
   BITMAPFILEHEADER bf;
   BITMAPINFOHEADER bi;
   RGBQUAD rgbq;
   memset(&bf, 0, sizeof(BITMAPFILEHEADER));
   memset(&bi, 0, sizeof(BITMAPINFOHEADER));
   memset(&rgbq, 0, sizeof(RGBQUAD));
   dmx = ((dimx & 0x03) ? (1 + (dimx >> 2)) : (dimx >> 2)) << 2;
   DWORD datasize = dmx * dimy;
   DWORD rgbqsize = ncol*sizeof(RGBQUAD);
   bf.bfType = *(WORD *)("BM");
   bf.bfSize = sizeof(BITMAPFILEHEADER)+
      sizeof(BITMAPINFOHEADER)+rgbqsize+datasize;
   bf.bfOffBits = sizeof(BITMAPFILEHEADER) +
      sizeof(BITMAPINFOHEADER) + rgbqsize;
   bi.biSize = sizeof(BITMAPINFOHEADER);
   bi.biWidth = dimx;
   bi.biHeight = dimy;
   bi.biPlanes = 1;
   bi.biBitCount = 8;
   bi.biCompression = BI_RGB;
   bi.biClrUsed=ncol;
   fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, file);
   fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, file);
   for (k = 0; k < ncol; k++) {
      cp = histo[k];
      rgbq.rgbRed = GetRValue(cp);
      rgbq.rgbGreen = GetGValue(cp);
      rgbq.rgbBlue = GetBValue(cp);
      fwrite(&rgbq, sizeof(RGBQUAD), 1, file);
   }
   for (y = dimy - 1; y >= 0; y--) {
      for (x = 0;x < dmx; x++) {
         cp = GetPixel(hdc, r.left + x, r.top + y);
         val = 0; k = 0;
         for (k = 0; k < ncol; k++) if (cp == histo[k]) {
            val = (UCHAR)k; break;
         }
         if ((x > dimx) || (cp == CLR_INVALID)) val=0;
         fputc(val, file);
      }
   }
   fclose(file); delete histo;
   return TRUE;
}
BOOL SaveMonochromeBitmap(HDC hdc, RECT& r,
```

```
   LPCTSTR filename, COLORREF bkcolor) {
   int x, y, dmx;
   int dimx = r.right - r.left, dimy = r.bottom - r.top;
   int val;
   COLORREF cp;
   FILE * file=fopen(filename,"wb");
   BITMAPFILEHEADER bf;
   BITMAPINFOHEADER bi;
   RGBQUAD rgbBlack={0,0,0,0}, rgbWhite={255, 255, 255, 0};
   memset(&bf, 0, sizeof(BITMAPFILEHEADER));
   memset(&bi, 0, sizeof(BITMAPINFOHEADER));
   dmx = ((dimx & 0x03) ? (1 + (dimx >> 2)) : (dimx >> 2)) << 2;
   DWORD datasize=dmx * dimy / 8;
   DWORD rgbqsize = 2*sizeof(RGBQUAD);
   bf.bfType = *(WORD *)("BM");
   bf.bfSize = sizeof(BITMAPFILEHEADER) +
      sizeof(BITMAPINFOHEADER) + rgbqsize + datasize;
   bf.bfOffBits = sizeof(BITMAPFILEHEADER)+
      sizeof(BITMAPINFOHEADER)+rgbqsize;
   bi.biSize = sizeof(BITMAPINFOHEADER);
   bi.biWidth = dimx;
   bi.biHeight = dimy;
   bi.biPlanes = 1;
   bi.biBitCount = 1;
   bi.biCompression = BI_RGB;
   bi.biClrUsed = 2;
   fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, file);
   fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, file);
   fwrite(&rgbWhite, sizeof(RGBQUAD), 1, file);
   fwrite(&rgbBlack, sizeof(RGBQUAD), 1, file);
   int count, bytes;
   for (y = dimy-1;y >= 0; y--) {
      val = count = bytes = 0;
      for (x = 0;x < dimx; x++) {
         cp = GetPixel(hdc, r.left + x, r.top + y);
         if ((cp != bkcolor) && (x < dimx))
            val |= (0x80 >> count);
         if (count == 7) {
            fputc(val, file); bytes++;
            count = 0; val = 0;
         } else count++;
      }
      if (count) {fputc(val, file); bytes++;}
      while (bytes % 4) {fputc(0, file); bytes++;}
   }
   fclose(file); return TRUE;
}
BOOL SaveTrueColorBitmap(HDC hdc, RECT& r, LPCTSTR filename) {
   FILE * file=fopen(filename,"wb");
   if (!file) return FALSE;
   int x, y, dmx;
   int dimx = r.right - r.left, dimy = r.bottom - r.top;
   COLORREF cp;
   BITMAPFILEHEADER bf;
   BITMAPINFOHEADER bi;
   RGBQUAD rgbBlack={0,0,0,0}, rgbWhite={255, 255, 255, 0};
   memset(&bf, 0, sizeof(BITMAPFILEHEADER));
   memset(&bi, 0, sizeof(BITMAPINFOHEADER));

   dmx = dimx*3;
   dmx = ((dmx & 0x03) ? (1 + (dmx >> 2)) : (dmx >> 2)) << 2;
```

```
   DWORD datasize = dmx * dimy * 3;
   bf.bfType = *(WORD *)("BM");
   bf.bfSize = sizeof(BITMAPFILEHEADER) +
      sizeof(BITMAPINFOHEADER) + datasize;
   bf.bfOffBits = sizeof(BITMAPFILEHEADER)+
      sizeof(BITMAPINFOHEADER);
   bi.biSize = sizeof(BITMAPINFOHEADER);
   bi.biWidth = dimx;
   bi.biHeight = dimy;
   bi.biPlanes = 1;
   bi.biBitCount = 24;
   bi.biCompression = BI_RGB;
   bi.biClrUsed = 0;
   fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, file);
   fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, file);
   int bytes;
   for (y = dimy-1; y >= 0; y--) {
      for (x = 0; x < dimx; x++) {
         cp = GetPixel(hdc, r.left + x, r.top + y);
         fputc(GetBValue(cp), file);
         fputc(GetGValue(cp), file);
         fputc(GetRValue(cp), file);
      }
      for (bytes = 3 * dimx; bytes % 4; bytes++) fputc(0, file);
   }
   fclose(file); return TRUE;
}
BOOL SaveTrueColorBitmap(HDC hdc, RECT& r, HBITMAP bmp, LPCTSTR filename) {
   FILE * file=fopen(filename,"wb");
   if (!file) return FALSE;
   int dmx, dimx = r.right - r.left, dimy = r.bottom - r.top;
   BITMAPFILEHEADER bf;
   BITMAPINFOHEADER bi;
   memset(&bf, 0, sizeof(BITMAPFILEHEADER));
   memset(&bi, 0, sizeof(BITMAPINFOHEADER));

   dmx = dimx*3;
   dmx = ((dmx & 0x03) ? (1 + (dmx >> 2)) : (dmx >> 2)) << 2;
   DWORD datasize = dmx * dimy;
   bf.bfType = *(WORD *)("BM");
   bf.bfSize = sizeof(BITMAPFILEHEADER) +
      sizeof(BITMAPINFOHEADER) + datasize;
   bf.bfOffBits = sizeof(BITMAPFILEHEADER)+
      sizeof(BITMAPINFOHEADER);
   bi.biSize = sizeof(BITMAPINFOHEADER);
   bi.biWidth = dimx;
   bi.biHeight = dimy;
   bi.biPlanes = 1;
   bi.biBitCount = 24;
   bi.biCompression = BI_RGB;
   bi.biClrUsed = 0;
   fwrite(&bf, sizeof(BITMAPFILEHEADER), 1, file);
   fwrite(&bi, sizeof(BITMAPINFOHEADER), 1, file);

   char *buffer = new char[datasize];
   int res = GetDIBits(hdc, bmp, 0, dimy, buffer, (BITMAPINFO*)&bi,
DIB_RGB_COLORS);
   fwrite(buffer, 1, datasize, file);

   delete buffer;
   fclose(file); return TRUE;
```

```
}
```

Fişierul buffers.h

```
////////////////////////////////////////////////////////////////////////////
// buffers.h - header file
////////////////////////////////////////////////////////////////////////////

#include "DataFile.h"

#ifndef __BUFFERS_H
#define __BUFFERS_H

//////////////////////////////////////////////////////////////////////////
//class DoubleBuffer
class DoubleBuffer {
   double *buf;
   int n;
public:
   DoubleBuffer();
   DoubleBuffer(DoubleBuffer& fb);
   DoubleBuffer(double *fb, int count);
   ~DoubleBuffer();
   void empty() {setSize(0);}
   bool isEmpty() {return (n == 0);}
   void setSize(int size);
   int  size() {return n;}
   void add(double x);
   void insert(double x, int pos);
   void remove(int pos);
   void sort();
   void multiply(double x);
   int  binarySearch(double x);
   int  binarySearch2(double x);
   CString toString();
   void fromString(LPCTSTR s);
   void fromRealBuffer(DoubleBuffer& fb);
   void fromArray(double *fb, int count);
   void writeToFile(DataFile *df);
   void readFromFile(DataFile *df);
   double& operator[] (UINT index) {return buf[index];}
   double interpolate(double x);
};

#endif//__BUFFERS_H
```

Fişierul buffers.cpp

```
#include "stdafx.h"
#include "buffers.h"

bool isRealPart(char ch) {
   return isdigit(ch) || (ch == '.') || (tolower(ch) == 'e');
}

//////////////////////////////////////////////////////////////////////////
//class DoubleBuffer
DoubleBuffer::DoubleBuffer() {
```

```
   n = 0; buf = NULL;
}
DoubleBuffer::DoubleBuffer(DoubleBuffer& rb) {
   n = 0; buf = NULL;
   fromRealBuffer(rb);
}
DoubleBuffer::DoubleBuffer(double *rb, int count) {
   n = 0; buf = NULL;
   fromArray(rb, count);
}
DoubleBuffer::~DoubleBuffer() {
   if (buf) delete buf;
}
void DoubleBuffer::setSize(int size) {
   if (size <= 0) {
      if (buf) delete buf; n = 0; buf = NULL;
      return;
   }
   if (buf == NULL) {
      buf = new double[size]; n = size;
      memset(buf, 0, sizeof(double)*size);
      return;
   }
   int nx = (size < n) ? size : n;
   double *nwbuf = new double[size];
   memset(nwbuf, 0, sizeof(double)*size);
   memcpy(nwbuf, buf, sizeof(double)*nx);
   delete buf; buf = nwbuf;
   n = size; return;
}
void DoubleBuffer::add(double x) {
   setSize(n+1);
   buf[n-1] = x;
}
void DoubleBuffer::insert(double x, int pos) {
   setSize(n+1);
   memmove(buf+pos+1, buf+pos, sizeof(double)*(n-pos-1));
   buf[pos] = x;
}
void DoubleBuffer::remove(int pos) {
   if (n <= 1) {setSize(0); return;}
   memmove(buf+pos, buf+pos+1, sizeof(double)*(n-pos-1));
   setSize(n-1);
}
void DoubleBuffer::sort() {
   if (n <= 1) return;
   bool done;
   int k; double aux;
   do {
      done = true;
      for (k = 0; k < n-1; k++) {
         if (buf[k] > buf[k+1]) {
            done = false;
            aux = buf[k];
            buf[k] = buf[k+1];
            buf[k+1] = aux;
         }
      }
   } while(!done);
}
void DoubleBuffer::multiply(double x) {
```

```
   if (!buf) return;
   for (int k = 0; k < n; k++) buf[k] *= x;
}
int DoubleBuffer::binarySearch(double x) {
   if (!buf) return 0;
   int a = 0, b = n - 1, c = (a + b) >> 1;
   if (x >= buf[b]) return b - 1;
   while (c != a) {
      if (x >= buf[c]) a = c;
      else b = c;
      c = (a + b) >> 1;
   }
   return c;
}
int DoubleBuffer::binarySearch2(double x) {
   if (!buf) return -1;
   if (x < buf[0] || x > buf[n - 1]) return -1;
   int a = 0, b = n - 2, c = (a + b) >> 1;
   if (x >= buf[b]) return b - 1;
   while (c != a) {
      if (x >= buf[c]) a = c;
      else b = c;
      c = (a + b) >> 1;
   }
   return c;
}
CString DoubleBuffer::toString() {
   int k; CString res, cs;
   for (k = 0; k < n; k++) {
      cs.Format("%g", buf[k]);
      res += cs;
      if (k < n-1) res+=' ';
   }
   return res;
}
void DoubleBuffer::fromString(LPCTSTR s) {
   double x;
   DEL(buf); n = 0;
   if (s == NULL) return;
   int j = 0, len = (int)strlen(s);
   do {
      for (; j < len; j++) if (isRealPart(s[j])) break;
      if (j < len) if (sscanf(s+j, "%lg", &x)) add((double)x);
      for (; j < len; j++) if (!isRealPart(s[j])) break;
   } while(j < len);
}
void DoubleBuffer::fromRealBuffer(DoubleBuffer& rb) {
   empty();
   if (rb.buf == NULL) return;
   buf = new double[rb.n]; n = rb.n;
   memcpy(buf, rb.buf, sizeof(double)*n);
}
void DoubleBuffer::fromArray(double *rb, int count) {
   empty();
   if (count <= 0) return;
   buf = new double[count]; n = count;
   memcpy(buf, rb, sizeof(double)*count);
}
void DoubleBuffer::writeToFile(DataFile *df) {
   df->writeInt(n);
   df->write(buf, n*sizeof(double));
```

```
}
void DoubleBuffer::readFromFile(DataFile *df) {
   empty();
   n = df->readInt();
   setSize(n);
   df->read(buf, n*sizeof(double));
}
double DoubleBuffer::interpolate(double x) {
   int k = (int)(x * n);
   if (k < 0) k = 0;
   if (k > n - 2) k = n - 2;
   double kx = x * n;
   return buf[k + 1] * (kx - k) + buf[k] * (k + 1 - kx);
}
```

Fişierul DataFile.h

```
#ifndef __DATA_FILE_H
#define __DATA_FILE_H

class DataFile {
   FILE *file;
   CString filename;
   BOOL bWrite, bRead;
   void tempName();
public:
   DataFile();
   ~DataFile();
   void assign(LPCTSTR filename);
   CString getName() {return filename;}
   void openRead();
   void openWrite();
   void openExisting();
   void openCreate();
   void close();
   void remove();
   BOOL rename(LPCTSTR name);
   BOOL copy(LPCTSTR name);
   BOOL isOpen() {return (file != NULL);}
   FILE *getFile() {return file;}
   void flush();
   void seek(long pos);
   long tell();
   long size();
   BOOL canRead() {return bRead;}
   BOOL canWrite() {return bWrite;}

   void writeChar(char x);
   void writeByte(BYTE x);
   void writeShort(short x);
   void writeWord(WORD x);
   void writeInt(int x);
   void writeDword(DWORD x);
   void writeFloat(float x);
   void writeDouble(double x);
   void writeString(LPCTSTR x);
   void write(LPVOID x, size_t length);

   char readChar();
   BYTE readByte();
```

```
   short readShort();
   WORD readWord();
   int readInt();
   DWORD readDword();
   float readFloat();
   double readDouble();
   CString readString();
   void read(LPVOID buf, size_t length);
};
#endif//__DATA_FILE_H
```

Fişierul DataFile.cpp

```
#include "stdafx.h"
#include "DataFile.h"

//////////////////////////////////////////////////////////////////////
//class DataFile
DataFile::DataFile() {
   file = NULL; bRead = bWrite = FALSE;
}
DataFile::~DataFile() {
   close();
}
void DataFile::close() {
   if (file) {fclose(file); file = NULL;}
   bRead = bWrite = FALSE;
}
void DataFile::remove() {
   close();
   DeleteFile(filename);
   filename.Empty();
}
BOOL DataFile::rename(LPCTSTR name) {
   close();
   BOOL res = MoveFileEx(filename, name,
      MOVEFILE_COPY_ALLOWED | MOVEFILE_REPLACE_EXISTING);
   if (res) filename = name;
   return res;
}
BOOL DataFile::copy(LPCTSTR name) {
   close();
   return CopyFile(filename, name, FALSE);
}
void DataFile::assign(LPCTSTR filename) {
   close();
   DataFile::filename = filename;
}
void DataFile::openRead() {
   close(); if (filename.IsEmpty()) tempName();
   file = fopen(filename, "rb");
   if (file) bRead = TRUE;
}
void DataFile::openWrite() {
   close(); if (filename.IsEmpty()) tempName();
   file = fopen(filename, "wb");
   if (file) bWrite = TRUE;
}
void DataFile::openExisting() {
   close(); if (filename.IsEmpty()) tempName();
   file = fopen(filename, "r+b");
```

```
   if (file) bRead = bWrite = TRUE;
}
void DataFile::openCreate() {
   close(); if (filename.IsEmpty()) tempName();
   file = fopen(filename, "w+b");
   if (file) bRead = bWrite = TRUE;
}
void DataFile::flush() {
   ASSERT(file); fflush(file);
}
void DataFile::seek(long pos) {
   ASSERT(file); fseek(file, pos, SEEK_SET);
}
long DataFile::tell() {
   ASSERT(file); return ftell(file);
}
long DataFile::size() {
   ASSERT(file);
   return _filelength(_fileno(file));
}
void DataFile::tempName() {
   char temppath[MAX_PATH], fn[MAX_PATH];
   GetTempPath(MAX_PATH, temppath);
   GetTempFileName(temppath, "ing", 0, fn);
   filename = fn;
}

//writing
void DataFile::writeChar(char x) {
   ASSERT(file); fwrite(&x, sizeof(char), 1, file);
}
void DataFile::writeByte(BYTE x) {
   ASSERT(file); fwrite(&x, sizeof(BYTE), 1, file);
}
void DataFile::writeShort(short x) {
   ASSERT(file); fwrite(&x, sizeof(short), 1, file);
}
void DataFile::writeWord(WORD x) {
   ASSERT(file); fwrite(&x, sizeof(WORD), 1, file);
}
void DataFile::writeInt(int x) {
   ASSERT(file); fwrite(&x, sizeof(int), 1, file);
}
void DataFile::writeDword(DWORD x) {
   ASSERT(file); fwrite(&x, sizeof(DWORD), 1, file);
}
void DataFile::writeFloat(float x) {
   ASSERT(file); fwrite(&x, sizeof(float), 1, file);
}
void DataFile::writeDouble(double x) {
   ASSERT(file); fwrite(&x, sizeof(double), 1, file);
}
void DataFile::writeString(LPCTSTR x) {
   ASSERT(file);
   size_t len = strlen(x);
   fwrite(x, 1+len, 1, file);
}
void DataFile::write(LPVOID x, size_t length) {
   ASSERT(file); fwrite(x, length, 1, file);
}
```

```
//reading
char DataFile::readChar() {
   ASSERT(file);
   char x; fread(&x, sizeof(char), 1, file); return x;
}
BYTE DataFile::readByte() {
   ASSERT(file);
   BYTE x; fread(&x, sizeof(BYTE), 1, file); return x;
}
short DataFile::readShort() {
   ASSERT(file);
   short x; fread(&x, sizeof(short), 1, file); return x;
}
WORD DataFile::readWord() {
   ASSERT(file);
   WORD x; fread(&x, sizeof(WORD), 1, file); return x;
}
int DataFile::readInt() {
   ASSERT(file);
   int x; fread(&x, sizeof(int), 1, file); return x;
}
DWORD DataFile::readDword() {
   ASSERT(file);
   DWORD x; fread(&x, sizeof(DWORD), 1, file); return x;
}
float DataFile::readFloat() {
   ASSERT(file);
   float x; fread(&x, sizeof(float), 1, file); return x;
}
double DataFile::readDouble() {
   ASSERT(file);
   double x; fread(&x, sizeof(double), 1, file); return x;
}
CString DataFile::readString() {
   ASSERT(file);
   CString cs; char ch;
   for (ch = fgetc(file); ch && (!feof(file)); ch = fgetc(file))
      cs += ch;
   return cs;
}
void DataFile::read(LPVOID buf, size_t length) {
   ASSERT(file); fread(buf, length, 1, file);
}
```

Fişierul discret.h

```
////////////////////////////////////////////////////////////////////////////
// discret.h - header file
////////////////////////////////////////////////////////////////////////////

#include "buffers.h"
#include "Material.h"

#ifndef __DISCRET_H
#define __DISCRET_H

#pragma pack (push)
#pragma pack (1)
class Cell {
private:
```

```
   // Ex. Ey, Ez, Hx, Hy, Hz, p, th
   double fv[8];
   int esize, eidx;
   double *ebuf;
public:
   int matidx;
public:
   Cell();
   ~Cell();
   double& Ex() {return fv[0];}
   double& Ey() {return fv[1];}
   double& Ez() {return fv[2];}
   double& Hx() {return fv[3];}
   double& Hy() {return fv[4];}
   double& Hz() {return fv[5];}
   double& p() {return fv[6];}
   double& th() {return fv[7];}
   double val(int idx);
   double E() {return val(17);}
   double H() {return val(18);}
   double S() {return val(19);}
   double sigma() {return mat_sigma(matidx, th());}
   double eps() {return mat_eps(matidx, th());}
   double miu() {return mat_miu(matidx, th());}
   void initEBuf(int size);
   BOOL hasEBuf() {return ebuf != NULL;}
   void pushEBuf(double x);
   double avgEBuf();
   void calcAvgEBuf();
   void copyEBuf(Cell *c);
   void logEBuf();
};
#pragma pack (pop)

class Plane {
   Cell *cells;
   int jmax, kmax;
public:
   Plane();
   ~Plane();
   void init(int jmax, int kmax);
   Cell& operator ()(int j, int k);
   BOOL isValid() {return cells != NULL;}
   int size() {return jmax * kmax;}
   int getJMax() {return jmax;}
   int getKMax() {return kmax;}
};

class Discret {
private:
   int if0, if1, jf1, jf2, kf1, kf2, kf3, kfex;
   int is1, is2, js1, js2, ks1, ks2;
   Plane *planes;
   struct WORKER_THREAD_INFO {
      Discret *_this, *prev;
      int iinit, istep;
      double thtime;
   };
   int running_threads, thread_count;
   CEvent **calcStopEvents, **calcMidEvents;
   struct WORKER_THREAD_INFO *wtis;
```

```
   int sequence, references;
public:
   double ttime;
   double lambda0, freq, dt;
   DoubleBuffer xcd, ycd, zcd;
   double xoven, xf1, xf2, xs1, xs2;
   double yoven, yf1, yf2, ys1, ys2;
   double zoven, zs1, zs2, zf1, zf2, zf3, zfex;
private:
   void sortAndDivideScalar(DoubleBuffer& rb, double dmin);
   Cell *getCell(int i, int j, int k);
   Cell *getCellX(int i, int j, int k);
   void calch(Discret *pd, int i, int j, int k);
   void calce(Discret *pd, int i, int j, int k, double thtime);
   void calcth(Discret *pd, int i, int j, int k, double thtime);
   static UINT __workerThreadLauncher(LPVOID param);
   UINT workerThread(struct WORKER_THREAD_INFO *wti);
   void waitForMidThreads(int iinit);
   void clearEvents();
public:
   Discret();
   ~Discret();
   void defaults();
   void init(int divfact, int tdiv, BOOL uselog = FALSE);
   void init(Discret *pd);
   BOOL isRunning() {return running_threads > 0;}
   void startCalculate(Discret *pd, double thtime = 0);
   void waitForThreads();
   double value(int idx, double x, double y, double z);
   void excit(double ex);
   int getSequence() {return sequence;}
   void refer() {references++;}
   void discard() {if (--references <= 0) delete this;}
   double avgPwr();
   double maxth();
   void initThFile();
   void updateThFile();
};

#endif//__DISCRET_H
```

Fişierul discret.cpp

```
#include "stdafx.h"
#include "discret.h"
#include "Utils.h"

DataFile thfile;

////////////////////////////////////////////////////////////////////////
//class Cell
Cell::Cell() {
   memset(this, 0, sizeof(Cell));
}
Cell::~Cell() {
   if (ebuf) delete ebuf;
}
double Cell::val(int idx) {
   switch(idx) {
      default: return fv[idx];
```

```
      case 17: return sqrt(sq(fv[0]) + sq(fv[1]) + sq(fv[2]));
      case 18: return sqrt(sq(fv[3]) + sq(fv[4]) + sq(fv[5]));
      case 19: return sqrt(sq(fv[1]*fv[5] - fv[2]*fv[4]) +
         sq(fv[2]*fv[3] - fv[0]*fv[5]) +
         sq(fv[0]*fv[4] - fv[1]*fv[3]));
      case 20: return sq(fv[0]) + sq(fv[1]) + sq(fv[2]);
      case 21: return avgEBuf();
      case 22: return sigma() * avgEBuf();
   }
}
void Cell::initEBuf(int size) {
   if (ebuf) delete ebuf;
   ebuf = new double[size];
   memset(ebuf, 0, size * sizeof(double));
   esize = size; eidx = 0;
}
void Cell::copyEBuf(Cell *c) {
   if (ebuf) {delete ebuf; ebuf = NULL;}
   esize = c->esize;
   if (!esize) return;
   eidx = c->eidx; ebuf = new double[esize];
   memcpy(ebuf, c->ebuf, esize * sizeof(double));
}
void Cell::pushEBuf(double x) {
   ebuf[eidx++] = x;
   if (eidx >= esize) eidx = 0;
}
double Cell::avgEBuf() {
   return fv[6];
}
void Cell::calcAvgEBuf() {
   if (!ebuf) return;
   int k; double s = 0;
   for (k = 0; k < esize; k++) s += ebuf[k];
   fv[6] = s / esize;
}
void Cell::logEBuf() {
   CString cs;
   cs.Format("ebuf: %d/%d avg=%lg", eidx, esize, avgEBuf());
   for (int k = 0; k < esize; k++) cs += Format(" %lg", ebuf[k]);
   log(cs);
}
Cell NullCell;

////////////////////////////////////////////////////////////////////////
//class Plane
Plane::Plane() {
   jmax = kmax = 0;
   cells = NULL;
}
Plane::~Plane() {
   if (cells) delete [] cells;
}
void Plane::init(int jmax, int kmax) {
   ASSERT(jmax > 0 && kmax > 0);
   this->jmax = jmax; this->kmax = kmax;
   cells = NULL; cells = new Cell[jmax * kmax];
}
Cell& Plane::operator () (int j, int k) {
   ASSERT((j >= 0) && (j < jmax));
   ASSERT((k >= 0) && (k < kmax));
```

```
   return cells[j * kmax + k];
}


////////////////////////////////////////////////////////////////////////
//class Discret
Discret::Discret() {
   planes = NULL; dt = 0;
   running_threads = thread_count = 0;
   calcStopEvents = calcMidEvents = NULL;
   SYSTEM_INFO si;
   GetSystemInfo(&si);
   thread_count = si.dwNumberOfProcessors;
   if (thread_count < 1) thread_count = 1;
   sequence = 0; references = 1; ttime = 0;
   defaults();
}
Discret::~Discret() {
   if (planes) delete [] planes;
   clearEvents();
}
void Discret::defaults() {
   freq = 2.45e9; lambda0 = LIGHT / freq;
   xoven = 306; xf1 = -10; xf2 = -25;
   yoven = 306; yf1 = 105; yf2 = 230;
   zoven = 184; zf1 = 70; zf2 = 115; zf3 = 210;

#if (PAYLOAD == 1)
   xs1 = 3; xs2 = xoven - 3;
   ys1 = 287 - 3; ys2 = ys1 + 20;
   zs1 = 3; zs2 = zoven - 3;
#elif (PAYLOAD == 2)
   xs1 = xoven - 23; xs2 = xs1 + 20;
   ys1 = 3; ys2 = yoven - 3;
   zs1 = 3; zs2 = zoven - 3;
#elif (PAYLOAD == 3)
   xs1 = xoven - 207; xs2 = xs1 + 20;
   ys1 = 3; ys2 = yoven - 3;
   zs1 = 3; zs2 = zoven - 3;
#endif

   //xs1 = 61.5; xs2 = 261.5;
   //ys1 = 117; ys2 = 217;
   //zs1 = 15; zs2 = 55;
   //xs1 = 111.5; xs2 = 211.5;
   //ys1 = 142; ys2 = 192;
   //zs1 = 15; zs2 = 35;

   xoven /= 1000; xf1 /= 1000; xf2 /= 1000;
   yoven /= 1000; yf1 /= 1000; yf2 /= 1000;
   zoven /= 1000; zf1 /= 1000; zf2 /= 1000; zf3 /= 1000;
   xs1 /= 1000; ys1 /= 1000; zs1 /= 1000;
   xs2 /= 1000; ys2 /= 1000; zs2 /= 1000;
}
void Discret::init(Discret *pd) {
   defaults();
   zfex = pd->zfex;
   if0 = pd->if0; if1 = pd->if1; jf1 = pd->jf1; jf2 = pd->jf2;
   kf1 = pd->kf1; kf2 = pd->kf2; kf3 = pd->kf3; kfex = pd->kfex;
   is1 = pd->is1; is2 = pd->is2;
   js1 = pd->js1; js2 = pd->js2;
```

```
   ks1 = pd->ks1; ks2 = pd->ks2;
   xcd.fromRealBuffer(pd->xcd);
   ycd.fromRealBuffer(pd->ycd);
   zcd.fromRealBuffer(pd->zcd);
   planes = new Plane[xcd.size() - 1];
   int i, j, k;
   for (i = 0; i < xcd.size() - 1; i++) {
      planes[i].init(pd->planes[i].getJMax(), pd->planes[i].getKMax());
   }
   for (i = is1; i <= is2; i++) {
      for (j = js1; j <= js2; j++) {
         for (k = ks1; k <= ks2; k++) {
            getCell(i, j, k)->matidx = pd->getCell(i, j, k)->matidx;
            getCell(i, j, k)->th() = pd->getCell(i, j, k)->th();
         }
      }
   }
   ttime = pd->ttime;
}
void Discret::init(int divfact, int tdiv, BOOL uselog) { // divfact >= 10
   zfex = zf3 - lambda0 / 2;
   xcd.empty(); ycd.empty(); zcd.empty();
   xcd.add(0); xcd.add(xoven); xcd.add(xf1); xcd.add(xf2);
   ycd.add(0); ycd.add(yoven); ycd.add(yf1); ycd.add(yf2);
   zcd.add(0); zcd.add(zoven);
   zcd.add(zf1); zcd.add(zf2); zcd.add(zf3);
   zcd.add(zfex);
   xcd.add(xs1); xcd.add(xs2);
   ycd.add(ys1); ycd.add(ys2);
   ycd.add(zs1); ycd.add(zs2);
   double dmax = lambda0 / divfact;
   sortAndDivideScalar(xcd, dmax);
   sortAndDivideScalar(ycd, dmax);
   sortAndDivideScalar(zcd, dmax);
   if0 = xcd.binarySearch(0);
   if1 = xcd.binarySearch(xf1);
   jf1 = ycd.binarySearch(yf1);
   jf2 = ycd.binarySearch(yf2);
   kf1 = zcd.binarySearch(zf1);
   kf2 = zcd.binarySearch(zf2);
   kf3 = zcd.binarySearch(zf3);
   kfex = zcd.binarySearch(zfex);
   if (uselog) {
      logf("imax = %d, jmax = %d, kmax = %d",
         xcd.size() - 1, ycd.size() - 1, zcd.size() - 1);
      logf("if0 = %d, if1 = %d, jf1 = %d, jf2 = %d, kf1 = %d, kf2 = %d, kf3 =
%d",
         if0, if1, jf1, jf2, kf1, kf2, kf3);
   }
   int i, j, k, sz = 0;
   planes = new Plane[xcd.size() - 1];
   for (i = 0; i < if1; i++) {
      planes[i].init(jf2 - jf1, kf3 - kf1);
      sz += planes[i].size();
   }
   for (; i < if0; i++) {
      planes[i].init(jf2 - jf1, kf2 - kf1);
      sz += planes[i].size();
   }
   for (; i < xcd.size() - 1; i++) {
      planes[i].init(ycd.size() - 1, zcd.size() - 1);
```

```
      sz += planes[i].size();
   }
   if (uselog) {
      logf("number of cells: %d", sz);
      logf("total memory per discretization: %d KB", sz * sizeof(Cell) / 1024);
   }

   is1 = xcd.binarySearch(xs1); is2 = xcd.binarySearch(xs2);
   js1 = ycd.binarySearch(ys1); js2 = ycd.binarySearch(ys2);
   ks1 = zcd.binarySearch(zs1); ks2 = zcd.binarySearch(zs2);
   for (i = is1; i <= is2; i++) {
      for (j = js1; j <= js2; j++) {
         for (k = ks1; k <= ks2; k++) {
            getCell(i, j, k)->matidx = 1;
            getCell(i, j, k)->initEBuf(tdiv);
         }
      }
   }
}
void Discret::sortAndDivideScalar(DoubleBuffer& rb, double dmax) {
   int i, j, k; rb.sort();
   double r1, r2;
   for (k = rb.size() - 2; k >= 0; k--) {
      r1 = rb[k]; r2 = rb[k + 1];
      i = (int)ceil((r2 - r1) / dmax);
      if (i <= 1) continue;
      for (j = i - 1; j > 0; j--) {
         rb.insert(r1 + (r2 - r1) * j / i, k + 1);
      }
   }
   for (k = rb.size() - 2; k >= 0; k--) {
      if (rb[k] == rb[k + 1]) rb.remove(k + 1);
   }
   ASSERT(rb.size() > 2);
}
Cell *Discret::getCell(int i, int j, int k) {
   if (i < 0) {
      return NULL;
   } else if (i < if1) {
      if (j < jf1 || k < kf1 || j >= jf2 || k >= kf3) return NULL;
      return &planes[i](j - jf1, k - kf1);
   } else if (i < if0) {
      if (j < jf1 || k < kf1 || j >= jf2 || k >= kf2) return NULL;
      return &planes[i](j - jf1, k - kf1);
   } else if (i < xcd.size() - 1) {
      if (j < 0 || k < 0 ||
         j >= ycd.size() - 1 || k >= zcd.size() - 1) return NULL;
      return &planes[i](j, k);
   } else return NULL;
}
Cell *Discret::getCellX(int i, int j, int k) {
   Cell *c = getCell(i, j, k);
   return c ? c : &NullCell;
}
void Discret::calch(Discret *pd, int i, int j, int k) {
   Cell *c = getCell(i, j, k); if (!c) return;
   Cell *pc = pd->getCell(i, j, k); ASSERT(pc);
   double dx = xcd[i+1] - xcd[i];
   double dy = ycd[j+1] - ycd[j];
   double dz = zcd[k+1] - zcd[k];
   double u, v, dt = this->dt / 2;
```

```
   Cell *pci, *pcj, *pck;
   pci = pd->getCellX(i-1, j, k);
   pcj = pd->getCellX(i, j-1, k);
   pck = pd->getCellX(i, j, k-1);

   u = (pc->Ey() - pck->Ey()) / dz;
   v = (pc->Ez() - pcj->Ez()) / dy;
   c->Hx() = pc->Hx() + (u - v) * dt / c->miu();
   ASSERT(!_isnan(c->Hx()));
   u = (pc->Ez() - pci->Ez()) / dx;
   v = (pc->Ex() - pck->Ex()) / dz;
   c->Hy() = pc->Hy() + (u - v) * dt / c->miu();
   ASSERT(!_isnan(c->Hy()));
   u = (pc->Ex() - pcj->Ex()) / dy;
   v = (pc->Ey() - pci->Ey()) / dx;
   c->Hz() = pc->Hz() + (u - v) * dt / c->miu();
   ASSERT(!_isnan(c->Hz()));
}
void Discret::calce(Discret *pd, int i, int j, int k, double thtime) {
   Cell *c = getCell(i, j, k); if (!c) return;
   Cell *pc = pd->getCell(i, j, k); ASSERT(pc);
   double dx = xcd[i+1] - xcd[i];
   double dy = ycd[j+1] - ycd[j];
   double dz = zcd[k+1] - zcd[k];
   double u, v, dt = this->dt / 2;
   double w = dt / c->eps(), ww = 1 - w * c->sigma();
   Cell *ci, *cj, *ck;
   ci = getCellX(i+1, j, k);
   cj = getCellX(i, j+1, k);
   ck = getCellX(i, j, k+1);

   u = (cj->Hz() - c->Hz()) / dy;
   v = (ck->Hy() - c->Hy()) / dz;
   c->Ex() = pc->Ex() * ww + (u - v) * w;
   ASSERT(!_isnan(c->Ex()));
   u = (ck->Hx() - c->Hx()) / dz;
   v = (ci->Hz() - c->Hz()) / dx;
   c->Ey() = pc->Ey() * ww + (u - v) * w;
   ASSERT(!_isnan(c->Ey()));
   u = (ci->Hy() - c->Hy()) / dx;
   v = (cj->Hx() - c->Hx()) / dy;
   c->Ez() = pc->Ez() * ww + (u - v) * w;
   ASSERT(!_isnan(c->Ez()));

   if (pc->hasEBuf()) {
      c->copyEBuf(pc);
      c->pushEBuf(c->val(20));
      c->calcAvgEBuf();
      calcth(pd, i, j, k, thtime);
   }
}
void Discret::calcth(Discret *pd, int i, int j, int k, double thtime) {
   if (thtime == 0) return;
   Cell *c = getCell(i, j, k); if (!c) return;
   Cell *pc = pd->getCell(i, j, k); ASSERT(pc);
   double dx = xcd[i+1] - xcd[i];
   double dy = ycd[j+1] - ycd[j];
   double dz = zcd[k+1] - zcd[k];
   double a = mat_a(c->matidx, c->th());
   double lambda = mat_lambda(c->matidx, c->th());
```

```
   Cell *cip, *cjp, *ckp, *cim, *cjm, *ckm;
   cip = getCellX(i+1, j, k); cim = getCellX(i-1, j, k);
   cjp = getCellX(i, j+1, k); cjm = getCellX(i, j-1, k);
   ckp = getCellX(i, j, k+1); ckm = getCellX(i, j, k-1);
   double tx, ty, tz;
   if (i == is1) tx = 2 * (cip->th() - c->th());
   else if (i == is2 - 1) tx = 2 * (cim->th() - c->th());
   else tx = cip->th() + cim->th() - 2 * c->th();
   if (j == js1) ty = 2 * (cjp->th() - c->th());
   else if (j == js2 - 1) ty = 2 * (cjm->th() - c->th());
   else ty = cjp->th() + cjm->th() - 2 * c->th();
   if (k == ks1) tz = 2 * (ckp->th() - c->th());
   else if (k == ks2 - 1) tz = 2 * (ckm->th() - c->th());
   else tz = ckp->th() + ckm->th() - 2 * c->th();

   c->th() += a * (c->p() / lambda + tx / sq(dx) + ty / sq(dy) + tz / sq(dz));
}
UINT Discret::__workerThreadLauncher(LPVOID param) {
   struct WORKER_THREAD_INFO *wti = (struct WORKER_THREAD_INFO *)param;
   UINT res = wti->_this->workerThread(wti);
   return res;
}
UINT Discret::workerThread(struct WORKER_THREAD_INFO *wti) {
   running_threads++;
   int i, j, k; BOOL first;
   int jmin, jmax, kmin, kmax, imax = xcd.size() - 1;
   for (first = TRUE, i = wti->iinit; i < imax; i += wti->istep) {
      if (first) {
         if (i < if0) {
            jmin = jf1; jmax = jf2;
            kmin = kf1; kmax = (i < if1) ? kf3 : kf2;
         } else {
            jmin = 0; jmax = ycd.size() - 1;
            kmin = 0; kmax = zcd.size() - 1;
            first = FALSE;
         }
      }
      for (j = jmin; j < jmax; j++) for (k = kmin; k < kmax; k++) {
         calch(wti->prev, i, j, k);
      }
   }
   waitForMidThreads(wti->iinit);
   for (first = TRUE, i = wti->iinit; i < imax; i += wti->istep) {
      if (first) {
         if (i < if0) {
            jmin = jf1; jmax = jf2;
            kmin = kf1; kmax = (i < if1) ? kf3 : kf2;
         } else {
            jmin = 0; jmax = ycd.size() - 1;
            kmin = 0; kmax = zcd.size() - 1;
            first = FALSE;
         }
      }
      for (j = jmin; j < jmax; j++) for (k = kmin; k < kmax; k++) {
         calce(wti->prev, i, j, k, wti->thtime);
      }
   }
   running_threads--;
   calcStopEvents[wti->iinit]->SetEvent();
   return 0;
}
```

```
void Discret::startCalculate(Discret *pd, double thtime) {
   clearEvents();
   sequence = 1 + pd->sequence;
   wtis = new struct WORKER_THREAD_INFO[thread_count];
   calcStopEvents = new CEvent *[thread_count];
   calcMidEvents = new CEvent *[thread_count];
   int k;
   for (k = 0; k < thread_count; k++) {
      wtis[k]._this = this;
      wtis[k].istep = thread_count;
      wtis[k].prev = pd;
      wtis[k].iinit = k;
      wtis[k].thtime = thtime;
      calcStopEvents[k] = new CEvent(FALSE, TRUE);
      calcMidEvents[k] = new CEvent(FALSE, TRUE);
   }
   for (k = 0; k < thread_count; k++) {
      CWinThread *t = AfxBeginThread(__workerThreadLauncher,
         &wtis[k], THREAD_PRIORITY, 0, CREATE_SUSPENDED);
      SetThreadAffinityMask(t->m_hThread, 1 << k);
      t->ResumeThread();
   }
}
void Discret::waitForThreads() {
   CMultiLock lock((CSyncObject **)calcStopEvents, thread_count);
   lock.Lock(INFINITE, TRUE);
}
void Discret::waitForMidThreads(int i) {
   if (thread_count <= 1) return;
   calcMidEvents[i]->SetEvent();
   CMultiLock lock((CSyncObject **)calcMidEvents, thread_count);
   lock.Lock(INFINITE, TRUE);
}
void Discret::clearEvents() {
   int k;
   for (k = 0; k < thread_count; k++) {
      if (calcStopEvents) delete calcStopEvents[k];
      if (calcMidEvents) delete calcMidEvents[k];
   }
   if (calcStopEvents) delete calcStopEvents;
   if (calcMidEvents) delete calcMidEvents;
   calcStopEvents = calcMidEvents = NULL;
}
double Discret::value(int idx, double x, double y, double z) {
   int i, j, k;
   i = xcd.binarySearch2(x);
   j = ycd.binarySearch2(y);
   k = zcd.binarySearch2(z);
   if (i < 0 || j < 0 || k < 0) return NAN;
   double qz = (z - zcd[k]) / (zcd[k+1] - zcd[k]);
   double vz00 = qz * getCellX(i, j, k+1)->val(idx) +
      (1 - qz) * getCellX(i, j, k)->val(idx);
   double vz01 = qz * getCellX(i, j+1, k+1)->val(idx) +
      (1 - qz) * getCellX(i, j+1, k)->val(idx);
   double vz10 = qz * getCellX(i+1, j, k+1)->val(idx) +
      (1 - qz) * getCellX(i+1, j, k)->val(idx);
   double vz11 = qz * getCellX(i+1, j+1, k+1)->val(idx) +
      (1 - qz) * getCellX(i+1, j+1, k)->val(idx);
   double qy = (y - ycd[j]) / (ycd[j+1] - ycd[j]);
   double vy0 = qy * vz01 + (1 - qy) * vz00;
   double vy1 = qy * vz11 + (1 - qy) * vz10;
```

```
   double qx = (x - xcd[i]) / (xcd[i+1] - xcd[i]);
   double res = qx * vy1 + (1 - qx) * vy0;
   return res;
}
void Discret::excit(double ex) {
   int i, j, k; Cell *c;
   k = kfex;
   for (i = 0; i < if1; i++) {
      for (j = jf1; j < jf2; j++) {
         c = getCell(i, j, k);
         c->Ex() = ex;
      }
   }
}
double Discret::avgPwr() {
   double res = 0;
   Cell *c; int i, j, k;
   for (i = is1; i <= is2; i++) {
      for (j = js1; j <= js2; j++) {
         for (k = ks1; k <= ks2; k++) {
            c = getCell(i, j, k);
            if (!c->hasEBuf()) continue;
            res += c->avgEBuf() * c->sigma() *
               (xcd[i+1] - xcd[i]) *
               (ycd[j+1] - ycd[j]) *
               (zcd[k+1] - zcd[k]);
         }
      }
   }
   return res;
}
double Discret::maxth() {
   double res = -300;
   Cell *c; int i, j, k;
   for (i = is1; i <= is2; i++) {
      for (j = js1; j <= js2; j++) {
         for (k = ks1; k <= ks2; k++) {
            c = getCell(i, j, k);
            if (!c->hasEBuf()) continue;
            if (res < c->th()) res = c->th();
         }
      }
   }
   return res;
}
void Discret::initThFile() {
   thfile.assign(getWorkDir() + "\\thf.fdtd");
   thfile.openWrite();
   thfile.writeString("FDTD2");
   thfile.writeFloat(xs2 - xs1);
   thfile.writeFloat(ys2 - ys1);
   thfile.writeFloat(zs2 - zs1);
   thfile.writeInt(is2 - is1 + 1);
   thfile.writeInt(js2 - js1 + 1);
   thfile.writeInt(ks2 - ks1 + 1);
}
void Discret::updateThFile() {
   if (!thfile.isOpen()) return;
   thfile.writeString("seq");
   thfile.writeInt(sequence);
   thfile.writeFloat(ttime);
```

```
   Cell *c; int i, j, k;
   for (i = is1; i <= is2; i++) {
      for (j = js1; j <= js2; j++) {
         for (k = ks1; k <= ks2; k++) {
            c = getCell(i, j, k);
            thfile.writeFloat(c->E());
            thfile.writeFloat(c->p());
            thfile.writeFloat(c->th());
         }
      }
   }
}
```

Fişierul fdtd1Dlg.h

```
// fdtd1Dlg.h : header file
//
#include "discret.h"

#if !defined(AFX_FDTD1DLG_H__2ADE0C74_AB3C_4D0F_AE66_0B86324B2473__INCLUDED_)
#define AFX_FDTD1DLG_H__2ADE0C74_AB3C_4D0F_AE66_0B86324B2473__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define STATE_INIT 0
#define STATE_RUNNING 1
#define STATE_PAUSED 2

/////////////////////////////////////////////////////////////////////////////
// CFdtd1Dlg dialog

class CFdtd1Dlg : public CDialog {
private:
   CDC *mdc;
   CBitmap *mBitmap, *oldBitmap;
   Discret *dis, *prevdis, *paintdis;
   volatile int state;
private:
   static UINT __threadLauncher(LPVOID _this);
   void thread();
   BOOL paintGraph(Discret *d, LPCTSTR imgFileName);
   static UINT __paintThreadLauncher(LPVOID _dis);
   void paintThread();
   void startPaintThread();

// Construction
public:
   CFdtd1Dlg(CWnd* pParent = NULL); // standard constructor

// Dialog Data
   //{{AFX_DATA(CFdtd1Dlg)
   enum { IDD = IDD_FDTD1_DIALOG };
      // NOTE: the ClassWizard will add data members here
   //}}AFX_DATA

   // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CFdtd1Dlg)
   protected:
```

```
   //}}AFX_VIRTUAL

// Implementation
protected:
   HICON m_hIcon;

   // Generated message map functions
   //{{AFX_MSG(CFdtd1Dlg)
   virtual BOOL OnInitDialog();
   afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
   afx_msg void OnPaint();
   afx_msg HCURSOR OnQueryDragIcon();
   virtual void OnOK();
   afx_msg void OnDrawItem(int nIDCtl, LPDRAWITEMSTRUCT lpDrawItemStruct);
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before
the previous line.

#endif //
!defined(AFX_FDTD1DLG_H__2ADE0C74_AB3C_4D0F_AE66_0B86324B2473__INCLUDED_)
```

Fişierul fdtd1Dlg.cpp

```
#include "stdafx.h"
#include "fdtd1.h"
#include "fdtd1Dlg.h"
#include "Utils.h"
#include "Bitmap.h"
#include "discret.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

COLORREF colorv(double x) {
   if (_isnan(x)) return (COLORREF)0xFFFFFF;
   if (x < 0) return 0;
   x = pow(x, 0.25);
   int r, g, b;
   r = 255 * x; b = 255 * (1 - x);
   g = (x > 0.5) ? (510 * (1 - x)) : (510 * x);
   return RGB(r, g, b);
}

CString getWorkDir() {
   return CString(getenv("temp")) + "\\_fdtd2_workdir";
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
   CAboutDlg();
```

```
// Dialog Data
   //{{AFX_DATA(CAboutDlg)
   enum { IDD = IDD_ABOUTBOX };
   //}}AFX_DATA

   // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CAboutDlg)
   protected:
   //}}AFX_VIRTUAL

// Implementation
protected:
   //{{AFX_MSG(CAboutDlg)
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
   //{{AFX_DATA_INIT(CAboutDlg)
   //}}AFX_DATA_INIT
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
   //{{AFX_MSG_MAP(CAboutDlg)
      // No message handlers
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFdtd1Dlg dialog

CFdtd1Dlg::CFdtd1Dlg(CWnd* pParent /*=NULL*/)
   : CDialog(CFdtd1Dlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(CFdtd1Dlg)
      // NOTE: the ClassWizard will add member initialization here
   //}}AFX_DATA_INIT
   // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
   m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

BEGIN_MESSAGE_MAP(CFdtd1Dlg, CDialog)
   //{{AFX_MSG_MAP(CFdtd1Dlg)
   ON_WM_SYSCOMMAND()
   ON_WM_PAINT()
   ON_WM_QUERYDRAGICON()
   ON_WM_DRAWITEM()
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFdtd1Dlg message handlers
BOOL CFdtd1Dlg::OnInitDialog() {
   CDialog::OnInitDialog();
   ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
   ASSERT(IDM_ABOUTBOX < 0xF000);
   CMenu* pSysMenu = GetSystemMenu(FALSE);
   if (pSysMenu != NULL) {
      CString strAboutMenu;
```

```
      strAboutMenu.LoadString(IDS_ABOUTBOX);
      if (!strAboutMenu.IsEmpty()) {
         pSysMenu->AppendMenu(MF_SEPARATOR);
         pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
      }
   }
   SetIcon(m_hIcon, TRUE); SetIcon(m_hIcon, FALSE);
   // TODO: Add extra initialization here
   CreateDirectory(getWorkDir(), NULL);
   CWnd *wnd; CRect rect; CDC *dc;

   wnd = GetDlgItem(IDC_GRAPH);
   SetOwnerDraw(wnd); wnd->GetClientRect(rect);
   dc = wnd->GetDC();
   mdc = new CDC; mBitmap = new CBitmap;
   mdc->CreateCompatibleDC(dc);
   mBitmap->CreateCompatibleBitmap(dc, rect.Width(), rect.Height());
   oldBitmap = mdc->SelectObject(mBitmap);
   dis = prevdis = paintdis = NULL;
   state = STATE_INIT;
   return TRUE;  // return TRUE  unless you set the focus to a control
}
void CFdtd1Dlg::OnSysCommand(UINT nID, LPARAM lParam) {
   if ((nID & 0xFFF0) == IDM_ABOUTBOX) {
      CAboutDlg dlgAbout;
      dlgAbout.DoModal();
   } else CDialog::OnSysCommand(nID, lParam);
}
void CFdtd1Dlg::OnPaint() {
   if (IsIconic()) {
      CPaintDC dc(this);
      SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
      int cxIcon = GetSystemMetrics(SM_CXICON);
      int cyIcon = GetSystemMetrics(SM_CYICON);
      CRect rect; GetClientRect(&rect);
      int x = (rect.Width() - cxIcon + 1) / 2;
      int y = (rect.Height() - cyIcon + 1) / 2;
      dc.DrawIcon(x, y, m_hIcon);
   } else CDialog::OnPaint();
}
HCURSOR CFdtd1Dlg::OnQueryDragIcon() {return (HCURSOR) m_hIcon;}

BOOL CFdtd1Dlg::paintGraph(Discret *d, LPCTSTR imgFileName) {
   CWnd *wnd; CRect rt, irt;
   int i, j, imax, jmax;
   double x, y, z, u, umin, umax;
   double *fp;

   wnd = GetDlgItem(IDC_GRAPH); wnd->GetClientRect(rt);
   mdc->FillRect(rt, &CBrush(RGB(255, 255, 255)));
   irt = rt; irt.DeflateRect(0, 0, 30, 0);
   imax = irt.Width(); jmax = irt.Height();
   umin = umax = 0;
   fp = new double[imax * jmax];
   for (i = 0; i < imax; i++) {
      for (j = 0; j < jmax; j++) {
         //x = d->xf2 + (d->xoven - d->xf2) * i / imax;
         //x = d->xoven * i / imax;
         //y = d->yoven * 0.5;
         //z = (d->zs1 + d->zs2) * 0.5;
         //z = d->zoven * j / jmax;
```

```
#if (PAYLOAD == 1)
         x = d->xs1 + (d->xs2 - d->xs1) * i / imax;
         y = d->ys1;
         z = d->zs1 + (d->zs2 - d->zs1) * j / jmax;
#else
         x = d->xs2;
         y = d->ys1 + (d->ys2 - d->ys1) * i / imax;
         z = d->zs1 + (d->zs2 - d->zs1) * j / jmax;
#endif
         fp[i + j*imax] = u = sq(d->value(22, x, y, z));
         if (_isnan(u)) continue;
         if (umin > u) umin = u;
         if (umax < u) umax = u;
      }
   }
   SetDlgItemText(IDC_TEXT, Format("P: %lg, u: %lg - %lg, th: %lg, time: %lg",
      d->avgPwr(), umin, umax, d->maxth(), d->ttime));
   if (umax - umin == 0) {
      umax = 1 + umin; delete fp; return FALSE;
   }
   for (i = 0; i < imax; i++) {
      for (j = 0; j < jmax; j++) {
         mdc->SetPixelV(i, jmax - 1 - j,
            colorv((fp[i + j*imax] - umin) / (umax - umin)));
      }
   }
   delete fp;
   CPen *pen, *oldpen;
   for (j = 0; j < jmax; j++) {
      pen = new CPen(PS_SOLID, 0, colorv(1 - (double)j/jmax));
      oldpen = mdc->SelectObject(pen);
      mdc->MoveTo(irt.right + 10, j);
      mdc->LineTo(irt.right + 30, j);
      mdc->SelectObject(oldpen);
      delete pen;
   }
   if (imgFileName) {
      SaveTrueColorBitmap(mdc->m_hDC, rt, *mBitmap, imgFileName);
   }
   wnd->Invalidate();
   return TRUE;
}
void CFdtd1Dlg::OnDrawItem(int nIDCtl, LPDRAWITEMSTRUCT p) {
   CDialog::OnDrawItem(nIDCtl, p);
   CWnd *wnd; CRect rt;
   switch (nIDCtl) {
   case IDC_GRAPH:
      wnd = GetDlgItem(nIDCtl); wnd->GetClientRect(rt);
      wnd->GetDC()->BitBlt(0, 0, rt.Width(), rt.Height(),
         mdc, 0, 0, SRCCOPY);
      break;
   }
}
void CFdtd1Dlg::OnOK() {
   GetDlgItem(IDOK)->EnableWindow(FALSE);
   if (state == STATE_INIT) {
      SetDlgItemText(IDOK, "Pause");
      state = STATE_RUNNING;
      SYSTEM_INFO si;
      GetSystemInfo(&si);
      CWinThread *t = AfxBeginThread(__threadLauncher,
```

```
            this, THREAD_PRIORITY, 0, CREATE_SUSPENDED);
         SetThreadAffinityMask(t->m_hThread,
            1 << (si.dwNumberOfProcessors - 1));
         t->ResumeThread();
      } else if (state == STATE_RUNNING) {
         state = STATE_PAUSED;
         SetDlgItemText(IDOK, "Resume");
      } else {
         state = STATE_RUNNING;
         SetDlgItemText(IDOK, "Pause");
      }
   }
   UINT CFdtd1Dlg::__threadLauncher(LPVOID _this) {
      ((CFdtd1Dlg*)_this)->thread(); return 0;
   }
   void CFdtd1Dlg::thread() {
      SetDlgItemText(IDC_STATUS, "Started");
      BOOL dopics = IsDlgButtonChecked(IDC_SAVE_PICS);
      double time = 0, T, dt, tht;
      int step;
      prevdis = new Discret();
      prevdis->init(DIVFACT, TDIV, TRUE);
      T = 1 / prevdis->freq;
      dt = T / (2 * TDIV);
      prevdis->initThFile();
      while (true) {
         GetDlgItem(IDOK)->EnableWindow(TRUE);
         if (state == STATE_PAUSED) {
            Sleep(100); continue;
         }
         step = 1 + prevdis->getSequence() - INIT_STEPS;
         if (step >= 0 && step % STEPS == 0) tht = 1;
         else tht = 0;
         dis = new Discret();
         dis->init(prevdis);
         dis->dt = dt;
         prevdis->excit(EXCIT * sin(M_2_PI * dis->freq * time));
         dis->startCalculate(prevdis, tht);
         dis->waitForThreads();
         time += dt;

         if (dopics) {
            if (dis->getSequence() % TDIV == 0) {
               paintGraph(dis, getWorkDir() +
                  Format("\\%08d.bmp", dis->getSequence()));
               SetDlgItemText(IDC_STATUS,
                  Format("Step %d", dis->getSequence()));
            }
         } else {
            if (!paintdis) {
               if (dis->getSequence() % TDIV == 0) {
                  paintdis = dis;
                  startPaintThread();
               }
            }
         }
         prevdis->discard(); prevdis = dis;

         if (tht != 0) {
            dis->ttime += tht;
            dis->updateThFile();
```

```
      }

      if (dis->ttime >= MAXTIME
         || dis->getSequence() >= MAXSTEPS
         || dis->maxth() > MAXTEMP) {
         state = STATE_PAUSED;
         SetDlgItemText(IDOK, "Resume");
      }
   }
}
UINT CFdtd1Dlg::__paintThreadLauncher(LPVOID _this) {
   ((CFdtd1Dlg*)_this)->paintThread(); return 0;
}
void CFdtd1Dlg::paintThread() {
   ASSERT(paintdis);
   paintGraph(paintdis, NULL);
   SetDlgItemText(IDC_STATUS,
      Format("Step %d", paintdis->getSequence()));
   paintdis->discard(); paintdis = NULL;
}
void CFdtd1Dlg::startPaintThread() {
   paintdis->refer();
   SYSTEM_INFO si;
   GetSystemInfo(&si);
   CWinThread *t = AfxBeginThread(__paintThreadLauncher,
      this, THREAD_PRIORITY, 0, CREATE_SUSPENDED);
   SetThreadAffinityMask(t->m_hThread,
      1 << (si.dwNumberOfProcessors - 1));
   t->ResumeThread();
}
```

Fişierul Material.h

```
////////////////////////////////////////////////////////////////////////////
// Material.h - header file
////////////////////////////////////////////////////////////////////////////

#ifndef __MATERIAL_H
#define __MATERIAL_H

double mat_sigma(int idx, double th);
double mat_eps(int idx, double th);
double mat_miu(int idx, double th);
double mat_a(int idx, double th);
double mat_c(int idx, double th);
double mat_ro(int idx, double th);
double mat_lambda(int idx, double th);

#endif//__MATERIAL_H
```

Fişierul Material.cpp

```
#include "stdafx.h"
#include "Material.h"

#define EPS2 (EPS0 * M_2_PI * 2.45E9)

double mat_miu(int idx, double th) {
```

```
   return MIU0;
}
double mat_sigma(int idx, double th) {
   double x;
   switch (idx) {
      default: return 0;
      case 1: x = EPS2 * 0.2;
   }
   return EPS2 * (x > 0.01 ? x : 0.01);
}
double mat_eps(int idx, double th) {
   switch (idx) {
      default: return EPS0;
      case 1: return 2 * EPS0;
   }
}
double mat_a(int idx, double th) {
   switch (idx) {
      case 1: return 1.2e-6;
   }
   return mat_lambda(idx, th) / (mat_c(idx, th) * mat_ro(idx, th));
}
double mat_c(int idx, double th) {
   switch (idx) {
      default: return 0;
      case 1: return 2200;
   }
}
double mat_ro(int idx, double th) {
   switch (idx) {
      default: return 0;
      case 1: return 380;
   }
}
double mat_lambda(int idx, double th) {
   switch (idx) {
      default: return 0;
      case 1: return 0.16;
   }
}
```

Fişierul Utils.h

```
#ifndef __UTILS_H
#define __UTILS_H

void SetOwnerDraw(HWND hwnd);
void SetOwnerDraw(CWnd *wnd);
CString Format(LPCTSTR lpszFormat, ...);
void MsgBox(LPCTSTR lpszFormat, ...);
CString hms(int nsec);
CString mseconds(double x);

double __forceinline sq(double x) {return x*x;}
double ipwr(double x, int k);
int round(double x);

void sizeRectToInclude(CRect& rt, POINT& p);
void setRectMaximalSize(CRect& rtmax, CRect& rt);
void drawRectangle(CDC *dc, CRect& rt);
```

```
POINT pointOnLine(POINT& p, POINT& q, float t);
CString getExeDir();
CString getDefaultsFile();

void log(LPCTSTR string);
void logf(LPCTSTR lpszFormat, ...);

#endif//__UTILS_H
```

Fişierul Utils.cpp

```
#include "stdafx.h"
#include "Utils.h"

void SetOwnerDraw(HWND hwnd) {
    SetWindowLong(hwnd, GWL_STYLE,
        SS_OWNERDRAW|GetWindowLong(hwnd, GWL_STYLE));
}
void SetOwnerDraw(CWnd *wnd) {
    SetOwnerDraw(wnd->m_hWnd);
}

CString Format(LPCTSTR lpszFormat, ...) {
   CString cs;
   va_list argList;
   va_start(argList, lpszFormat);
   cs.FormatV(lpszFormat, argList);
   va_end(argList);
   return cs;
}
void MsgBox(LPCTSTR lpszFormat, ...) {
   CString cs;
   va_list argList;
   va_start(argList, lpszFormat);
   cs.FormatV(lpszFormat, argList);
   va_end(argList);
   AfxMessageBox(cs);
}
CString hms(int nsec) {
   if (nsec <= 0) return "00";
   CString cs;
   cs.Format("%02d", nsec % 60); nsec /= 60;
   if (nsec) {
      cs.Insert(0, Format("%02d:", nsec % 60)); nsec /= 60;
      if (nsec) cs.Insert(0, Format("%02d:", nsec));
   }
   return cs;
}
CString mseconds(double x) {
   ASSERT(x >= 0);
   if (x < 1e-3) return Format("%.1lf us", x*1e6);
   if (x < 1) return Format("%.1lf ms", x*1e3);
   return Format("%.1lf s", x);
}
double ipwr(double x, int k) {
   if (k > 0) {
      double xx = x, res = 1;
      for( ; k; k >>= 1, xx *= xx) {
         if (k & 1) res *= xx;
      }
      return res;
```

```
   } else if (k < 0) {
      return 1/ipwr(x, -k);
   } else return 1;
}
int round(double x) {
   int r = (int)x;
   if (x - r <= 0.5) return r;
   return r + 1;
}

void sizeRectToInclude(CRect& rt, POINT& p) {
   if (rt.left > p.x) rt.left = p.x;
   if (rt.right < p.x) rt.right = p.x;
   if (rt.top > p.y) rt.top = p.y;
   if (rt.bottom < p.y) rt.bottom = p.y;
}
void setRectMaximalSize(CRect& rtmax, CRect& rt) {
   if (rt.left   < rtmax.left)   rt.left   = rtmax.left;
   if (rt.top    < rtmax.top)    rt.top    = rtmax.top;
   if (rt.right  > rtmax.right)  rt.right  = rtmax.right;
   if (rt.bottom > rtmax.bottom) rt.bottom = rtmax.bottom;
}
void drawRectangle(CDC *dc, CRect& rt) {
   dc->MoveTo(rt.left, rt.top);
   dc->LineTo(rt.left, rt.bottom);
   dc->LineTo(rt.right, rt.bottom);
   dc->LineTo(rt.right, rt.top);
   dc->LineTo(rt.left, rt.top);
}
POINT pointOnLine(POINT& p, POINT& q, float t) {
   POINT r;
   r.x = p.x + (q.x - p.x)*t;
   r.y = p.y + (q.y - p.y)*t;
   return r;
}
CString getExeDir() {
   CString cs(GetCommandLine());
   cs = cs.Mid(1, cs.Find('\"', 1) - 1);
   return cs.Left(1 + cs.ReverseFind('\\'));
}
CString getDefaultsFile() {
   return getExeDir() + "ingot-defaults.mdf";
}
void log(LPCTSTR string) {
   FILE *f = fopen(getWorkDir() + "\\fdtd1.log", "at+");
   fputs(string, f); fputc('\n', f); fclose(f);
}
void logf(LPCTSTR lpszFormat, ...) {
   CString cs;
   va_list argList;
   va_start(argList, lpszFormat);
   cs.FormatV(lpszFormat, argList);
   va_end(argList);
   log(cs);
}
```