

Universitatea Politehnica București
Facultatea de Automatică și Calculatoare

Simularea transferului de căldură folosind ecuații de tip Laplace-Poisson

Autor:

Tuleiu Ana-Maria

Coordonatori științifici:

ing. Mihai Victor Pricop

șl. dr. ing. Emil Slușanschi

București

Iulie 2009

Capitolul 1

Introducere

Această lucrare prezintă o soluție computațională pentru o problemă de simulare ce presupune rezolvarea numerică a ecuațiilor Laplace-Poisson pentru transferul de căldură, problemă des întâlnită în aeronautică în simulări de dinamică a zborului, dar și în alte ramuri ale industriei. Soluția acestei probleme este utilă, spre exemplu, în prezicerea, identificarea și vizualizarea punctelor cu valori termice mari folosind modele computaționale de simulare.

Simularea se face folosind condiții de tip Dirichlet pentru marginile domeniului de calcul. Obținerea soluției în regim staționar implică aproximarea ecuației de transfer prin calcularea diferențelor finite și rezolvarea sistemului de ecuații folosind metode numerice precum Jacobi, Gauss-Seidel și Gauss-Seidel cu suprarelaxare. Obținerea soluției în regim nestaționar implică utilizarea unei scheme explicite de discretizare folosind diferențe finite. Modelul implementat în soluție este folosit pentru simularea la nivel 2D, deși poate fi cu ușurință evoluat la un model 3D.

Rezolvarea problemei Dirichlet pentru ecuația lui Laplace constă în stabilirea unor valori fixe pentru marginile domeniului de calcul, valori obținute folosind o funcție continuă cu derivata de ordinul II egală cu 0. Funcția respectivă va fi folosită și pentru determinarea valorilor din interiorul domeniului. Pentru cazul în care de interes este numai distribuția temperaturii la regim staționar, temperatura

din interiorul domeniului se calculează iterând algoritmul până când valorile se stabilizează și algoritmul atinge condiția de oprire. Luând în considerare și efortul computațional, precum și instabilitatea numerică datorată aproximărilor succesive, condiția de oprire este dată de un număr maxim de iterații în conjuncție cu valoarea minimă a diferențelor între două iterații succesive (permițând algoritmului să se oprească chiar dacă sistemul nu s-a stabilizat complet, dar este totuși în limitele de precizie cerute de utilizator). Pentru cazul în care de interes este evoluția temporală a distribuției de temperatură (regimul nestăționar) temperatura din interiorul domeniului se calculează iterând algoritmul până se atinge condiția de oprire. Aceasta este dată de un număr maxim de iterații calculat în funcție de intervalul de timp fizic care se dorește a fi simulat.

Algoritmii aleși nu sunt neapărat optimi din punct de vedere matematic, însă structura lor permite o paralelizare a calculului avantajoasă (lucrul direct cu matrice, blocuri de tip „for” ușor paralelizabile, număr redus de instrucțiuni de tip „branch”, etc.), ei devenind astfel candidații potriviți pentru implementarea aplicației. De asemenea, implementarea paralelă a aplicației ține cont și de arhitectura sistemului de calcul pe care aceasta rulează. Pentru sisteme de tip mono-procesor cu un singur sau mai multe core-uri, implementarea este dezvoltată folosind API-ul OpenMP (Open Multi-Processing). Pentru sisteme distribuite cu noduri de procesare mono sau multi-procesor implementarea s-a făcut folosind API-ul MPI (Message Passing Interface), iar pentru un câștig suplimentar în viteza de procesare aplicația a fost implementată folosind și cele două API-uri împreună.

Deoarece aplicația a fost implementată pentru a putea fi integrată într-un pachet de aplicații de simulare, componenta sa de calcul trebuie să ofere posibilitatea cuplării cu alte aplicații din cadrul pachetului. Astfel, aplicația oferă utilizatorului posibilitatea modificării tuturor parametrilor de rulare dintr-un fișier de configurare al cărui conținut, spre exemplu, poate fi ușor generat dintr-un alt program. Deci putem integra aplicația și cu alte surse de intrare cum ar fi, de exemplu, partițiile

unui graf obținute cu ParMetis, având astfel posibilitatea să calculăm propagarea printr-un corp de dimensiuni mari, sau forme complexe, rezolvând, pe rând, fiecare din partițiile date la intrare.

Problema pe care aplicația o rezolvă impune existența unei modalități facile de reprezentare a rezultatelor obținute. Din acest motiv, aplicația este capabilă să furnizeze la ieșire datele reprezentate în formatul VTK (Vizualization Tool Kit), care poate fi apoi folosit pentru vizualizare în aplicații precum ParaView.

Capitolul 2

Aspecte teoretice

2.1 Generalități

Termodinamica demonstrează că energia poate fi transferată prin interacții la granița dintre un sistem cu alt sistem sau cu mediul înconjurător. Energiile pot fi de tip lucru mecanic sau căldură. La rândul ei căldura este o formă de transfer de energie microscopică și dezordonată (energie internă), datorată diferențelor de temperatură. Transferul este realizat prin procese spontane de la sistemul cu temperatură mai ridicată la cel cu temperatură joasă, luând în considerare și ireversibilitatea introdusă de principiul II al termodinamicii.

Temperatura se măsoară fie în Kelvini pe scara absolută de temperatură, fie folosind alte entități relative definite pe scări empirice de temperatură, cum ar fi Celsius sau Fahrenheit. Practica obișnuită în literatura de specialitate este aceea de a nota temperatura în valoare absolută cu T , iar pentru valori exprimate pe alte scări, cu t . În lucrarea prezentă, vom exprima valorile variabilelor de temperatură în grade Celsius ($^{\circ}C$), care respectă formula de transformare:

$$T[K] = t[^{\circ}C] + 273,15$$

Pentru un punct oarecare din spațiu $A(x, y, z)$ avem, de exemplu, ecuația pentru un câmp tridimensional tranzitoriu:

$$T_A = T(x, y, z, \tau) \quad (2.1)$$

Câmpul de temperatură reprezintă totalitatea valorilor temperaturii t pentru toate punctele din spațiul considerat la un moment dat τ . În funcție de prezența sau absența lui τ ca variabilă explicită în ecuația (1), avem fie un câmp de temperatură tranzitiv (nestaționar), fie un câmp de temperatură permanent (staționar). Pentru câmpul staționar vom avea, în mod evident, derivata temporală nulă. De asemenea, în funcție de numărul componentelor spațiale de care depinde temperatura, putem avea câmpuri de temperatură unidimensionale, bidimensionale, tridimensionale etc.

Fluxul de căldură, notat cu \dot{Q} , reprezintă energia termică ce se transferă prin suprafața unui sistem către un alt sistem în unitatea de timp, și este descris de ecuația:

$$\dot{Q} = \frac{\partial E}{\partial \tau} \quad [W]$$

2.2 Conducția termică

Conducția termică reprezintă procesul de transfer energetic de la particulele cu energie ridicată, ce aparțin de corpul sau regiunea cu temperatură mai mare, către particulele cu energie scăzută, aparținând corpului sau regiunii de temperatură mai mică.

Densitatea fluxului de căldură reprezintă fluxul de căldură care traversează unitatea de suprafață, și este exprimată prin ecuația:

$$\bar{q}_s = \frac{\dot{Q}}{S} \quad \left[\frac{W}{m^2} \right] \quad (2.2)$$

cu componenta pe axa O_x proporțională cu:

$$\dot{Q}_x [W] \sim S [m^2] \frac{\Delta t}{\Delta x} \quad \frac{[K]}{[m]}$$

adică egală cu:

$$\dot{Q}_x = \lambda S \frac{\Delta t}{\Delta x} \quad [W] \quad (2.3)$$

unde $\lambda \left[\frac{W}{m \cdot K} \right]$ exprimă o proprietate a materialului numită conductivitatea termică. Din ecuațiile (2) și (3), rezultă:

$$q_x = \frac{\dot{Q}_x}{S} = -\lambda \frac{dt}{dx} \quad \left[\frac{W}{m^2} \right] \quad (2.4)$$

unde semnul minus indică propagarea căldurii în sensul descreșterii temperaturii.

Interpretând ecuația (4) determinăm că fluxul termic este o mărime de tip vectorial, paralelă și opusă gradientului de temperatură, și în același timp perpendiculară pe orice suprafață de temperatură constantă. Prin urmare, fluxul termic \dot{Q}_x este un scalar rezultat din produsul scalar a doi vectori, q_x și aria suprafeței perpendiculare pe gradientul de temperatură. Luând aceste observații în considerare, putem formula legea lui Fourier astfel:

$$\vec{q} = -\lambda \vec{\nabla} t = -\lambda \cdot \left(\vec{i} \frac{\partial t}{\partial x} + \vec{j} \frac{\partial t}{\partial y} + \vec{k} \frac{\partial t}{\partial z} \right) \quad \left[\frac{W}{m^2} \right] \quad (2.5)$$

unde ∇ este operatorul diferențial tridimensional, iar $t(x, y, z)$ este câmpul de temperatură.

Ecuația (5) exprimă implicit perpendicularitatea lui \vec{q} pe suprafețele izoterme. De aceea, formularea scalară a legii lui Fourier este:

$$q_n = -\lambda \frac{\partial q}{\partial n} \quad \left[\frac{W}{m^2} \right] \quad (2.6)$$

unde q_n este densitatea de flux termic pe direcția n, direcție perpendiculară pe suprafețele izoterme. Tot din ecuația (5) putem deduce descompunerea fluxului termic pe direcțiile sistemului de coordonate, care în cazul unui sistem tridimensional de coordonate cartezian este de forma:

$$\vec{q} = \vec{i} \cdot q_x + \vec{j} \cdot q_y + \vec{k} \cdot q_z$$

unde:

$$q_x = -\lambda \frac{\partial t}{\partial x}, \quad q_y = -\lambda \frac{\partial t}{\partial y}, \quad q_z = -\lambda \frac{\partial t}{\partial z} \quad (2.7)$$

Proprietatea de izotropie a mediului, conform căreia conductivitatea termică este independentă de direcțiile de propagare a energiei termice, este de asemenea dedusă implicit din ecuația (5).

Conductivitatea termică

Proprietățile termofizice ale materiei sunt deosebit de importante în analiza proceselor de transfer de căldură. Aceste proprietăți se împart în proprietăți de transport (conductivitatea termică λ , vâscozitate cinematică ν , coeficient de difuzie D) și în proprietăți termodinamice (densitate ρ , căldură specifică c). Conductivitatea termică influențează legea lui Fourier deoarece indică efectul transferului de energie prin procesul de conducție. Aceasta proprietate depinde de structura fizică a materiei, care la rândul ei depinde de starea de agregare a materiei.

Ecuația diferențială generală a conducției termice

Analiza transferului de căldură ajută la determinarea câmpului de temperatură sau a distribuției spațiale de temperatură într-un mediu dat, prin impunerea unor condiții la limită. Astfel, cunoscând distribuția, se pot calcula fluxurile de căldură aplicând legea lui Fourier.

Pentru a determina ecuația diferențială a conducției, se aplică legea conservării energiei pentru un element de volum infinitesimal. Soluția ecuației rezultate reprezintă distribuția de temperatura în mediul considerat.

Considerăm un mediu omogen în care există un gradient de temperatură, iar câmpul de temperatură la un moment dat se exprimă în coordonate carteziane sub forma $t(x, y, z)$ [$^{\circ}C$]. Definim elementul de volum infinitesimal $dV = dx \cdot dy \cdot dz$, prin suprafața căruia are loc transfer de energie termică prin fenomenul de conducție. Fluxurile termice normale pe fețele suprafeței de control ce conțin punctul curent (x, y, z) sunt notate $\dot{Q}_x, \dot{Q}_y, \dot{Q}_z$, după cum vectorii care le reprezintă sunt paraleli cu O_x, O_y, O_z . Expresiile lor rezultă din legea lui Fourier:

$$\begin{aligned}
\dot{Q}_x &= -\lambda \cdot dy \cdot dz \cdot \frac{\partial t}{\partial x} \\
\dot{Q}_y &= -\lambda \cdot dx \cdot dz \cdot \frac{\partial t}{\partial y} \\
\dot{Q}_z &= -\lambda \cdot dx \cdot dy \cdot \frac{\partial t}{\partial z}
\end{aligned} \tag{2.8}$$

Considerăm fluxurile termice ca fiind funcții continue de coordonate. Variația lor infinitezimală, pentru fiecare direcție, poate fi aproximată cu o funcție liniară astfel:

$$\begin{aligned}
\dot{Q}_{x+dx} &= \dot{Q}_x + \frac{\partial \dot{Q}_x}{\partial x} dx \\
\dot{Q}_{y+dy} &= \dot{Q}_y + \frac{\partial \dot{Q}_y}{\partial y} dy \\
\dot{Q}_{z+dz} &= \dot{Q}_z + \frac{\partial \dot{Q}_z}{\partial z} dz
\end{aligned} \tag{2.9}$$

Ecuațiile (9) reprezintă fluxurile termice normale pe fețele suprafeței de control care conține punctul de coordonate $(x + dx, y + dy, z + dz)$ conform Figura 1.1:

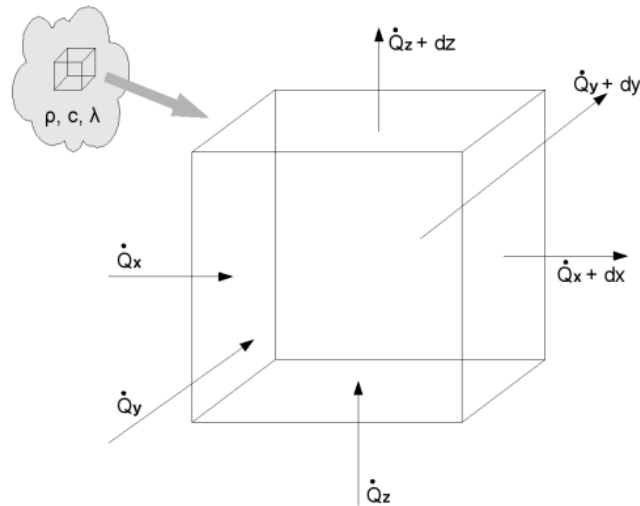


Figura 2.1: Fluxurile termice normale

Putem considera ca în mediu există surse termice de căldură interioare, care pot fi datorate schimbării stării de agregare, unor reacții chimice, etc. Vom nota

$q_v[\frac{W}{m^3}]$ *fluxul termic volumetric*. Energia termică generată de surse în elementul de volum dV va fi reprezentată de:

$$\dot{E}_{gen} = q_v dx \cdot dy \cdot dz \quad [W] \quad (2.10)$$

Variația energiei termice conținute de elementul de volum dV sau energia termică acumulată de acesta într-un interval de timp infinitesimal $\Delta\tau$, poate fi scrisă sub forma:

$$\frac{\partial E}{\partial \tau} = \frac{\partial}{\partial \tau}(\rho c t) dx \cdot dy \cdot dz = \rho c \frac{\partial t}{\partial \tau} dx \cdot dy \cdot dz \quad [W] \quad (2.11)$$

Se poate scrie astfel ecuația de conservare a energiei pentru un element de volum:

$$\frac{\partial E}{\partial \tau} = \dot{Q}_x + \dot{Q}_y + \dot{Q}_z - \dot{Q}_{x+dx} - \dot{Q}_{y+dy} - \dot{Q}_{z+dz} + \dot{E}_{gen} \quad [W] \quad (2.12)$$

Din ecuația (12) se poate deduce:

$$\rho c \frac{\partial t}{\partial \tau} dx \cdot dy \cdot dz = -\frac{\partial \dot{Q}_x}{\partial x} dx - \frac{\partial \dot{Q}_y}{\partial y} dy - \frac{\partial \dot{Q}_z}{\partial z} dz + q_v dx \cdot dy \cdot dz \quad [W] \quad (2.13)$$

Introducând ecuația (8) în ecuația (13) și simplificând relația astfel obținută prin elementul de volum rezultă *ecuația diferențială generală a conducției termice, reprezentată în coordonate carteziane*:

$$\rho c \frac{\partial t}{\partial \tau} = \frac{\partial}{\partial x} \left(\lambda \frac{\partial t}{\partial x} \right) + \frac{\partial}{\partial y} \left(\lambda \frac{\partial t}{\partial y} \right) + \frac{\partial}{\partial z} \left(\lambda \frac{\partial t}{\partial z} \right) + q_v \quad [\frac{W}{m^3}] \quad (2.14)$$

Astfel se demonstrează că variația energiei termice conținute în unitatea de volum trebuie să fie egală cu căldura netă transferată prin conducție acelei unități de volum însumată cu fluxul termic volumetric generat. Rezolvând ecuația (14) se poate obține distribuția instantanee de temperatură $t(x, y, z, \tau)$. În practică se lucrează cu variante simplificate ale acestei ecuații generale. Se poate considera spre exemplu conductivitatea termică λ ca fiind o constantă independentă de coordonate și temperatură, rezultând astfel următoarea formă simplificată a ecuației diferențiale a conducției termice:

$$\frac{1}{a} \frac{\partial t}{\partial \tau} = \frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} + \frac{\partial^2 t}{\partial z^2} + \frac{q_v}{\lambda} = \nabla^2 t + \frac{q_v}{\lambda} \quad [\frac{K}{m^2}] \quad (2.15)$$

Am notat cu $a = \frac{\lambda}{\rho c}$ $\left[\frac{m^2}{s}\right]$ *difuzitatea termică* ce reprezintă o constantă de material. O valoare mare pentru a (λ cu o valoare mare sau o valoare mică pentru ρc) implică o valoare ridicată pentru $\frac{\partial t}{\partial \tau}$, deci poate fi privită ca o corespondență cu un mediu în care temperatura variază rapid în procesul de transfer termic conductiv staționar.

2.3 Transmiterea căldurii prin conducție în regim staționar

2.3.1 Abordare analitică

În această secțiune se tratează situațiile în care transferul termic conductiv se desfășoară invariant în timp. Pentru soluționarea transferului unidimensional este necesară o singură coordonată pentru descrierea distribuției spațiale a temperaturii. Ecuația diferențială a conducției termice devine:

$$\vec{\nabla} \left(\lambda \vec{\nabla} t \right) + q_v = 0 \quad (2.16)$$

Dacă se consideră geometrii plane, câmpul unidimensional de temperatură se exprimă uzual sub forma $t = t(x)$. Dacă se consideră geometrii cilindrice sau sferice, noțiunea de unidimensional se referă la direcția radială, rezultând astfel o exprimare pentru câmpul de temperatură de forma $t = t(r)$. Dacă se consideră geometrii mai complexe, diferite de cele elementare prezentate mai sus, atunci acestea sunt approximate prin geometrii mai simple, calculul transferului conductiv unidimensional conducând la rezultate în limita unor erori admisibile.

Există totuși situații în care abordarea unidimensională a conducției termice conduce la aproximații nesatisfăcătoare din punctul de vedere al erorilor. Se va presupune astfel un sistem bidimensional în care se realizează un transfer termic conductiv de la o suprafață termică cu temperatura t_1 la o suprafață cu temperatura t_2 . Dacă se consideră tot un transfer în regim staționar, și absența unor surse interioare de

caldură, atunci distribuția spațială a temperaturii este dată de ecuația:

$$\frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} = 0 \quad (2.17)$$

Soluția acestei ecuații exprimă distribuția temperaturii în funcție de cele două coordonate x și y . Conform legii lui Fourier, fluxul termic într-un punct $N(x,y)$, este un vector perpendicular pe izoterma care trece prin acel punct. Dacă se cunoaște distribuția de temperatură atunci și componentele vectorului flux pe cele două axe de coordonate vor fi cunoscute, putând fi obținute din ecuația (7). Singura problemă în analiza conducției bidimensionale în această situație este determinarea câmpului de temperaturi. Acest lucru se poate face prin metode analitice, grafice sau numerice.

Abordarea analitică a problemelor de conducție bidimensională se poate face prin soluționare matematică exactă a ecuației (17). Principalul avantaj pe care metoda analitică îl oferă este determinarea temperaturilor în orice punct al sistemului considerat. Prin celelalte metode se determină temperaturile unui număr finit de puncte discrete ale sistemului. Dezavantajul acestei metode este însă acela că ea poate fi folosită numai pentru geometrii simple și în anumite condiții la limită, devenind astfel aproape de neutilizat în practică.

2.3.2 Metode numerice de rezolvare a conducției termice staționare

Datorită geometriilor complexe sau a condițiilor la limită multe probleme de conducție termică bidimensională nu pot fi rezolvate analitic. Pentru astfel de situații se utilizează metode numerice de rezolvare: metoda diferențelor finite (MDF), metoda elementelor finite (MEF), etc. În continuare în cadrul acestei lucrări va fi prezentată metoda diferențelor finite (MDF).

Metodele numerice pot determina temperaturile doar într-un număr discret de puncte. Primul pas în metoda diferențelor finite constă în selectarea mulțimii punctelor pentru care se vor calcula temperaturile. Acest lucru se va realiza prin împărțirea

mediului considerat într-un număr finit de regiuni cu dimensiuni mici. Punctul central al fiecărei astfel de regiuni este denumit punct nodal (nod central). Ansamblul acestor puncte formează rețeaua de noduri. Poziția unui punct nodal în raport cu axele de coordonate carteziene O_x și O_y este stabilită printr-o pereche (x,y) . De asemenea se introduce convenția conform căreia temperatura medie a regiunii este egală cu temperatura punctului nodal asociat acesteia.

Pasul doi, ulterior stabilirii rețelei, constă în scrierea ecuațiilor de conservare a energiei pentru fiecare regiune a sistemului considerat. Necunoscutele sistemului de ecuații astfel obținut reprezintă temperaturile în punctele nodale. Aceste ecuații se numesc "ecuații nodale cu diferențe finite". Se consideră nodul $N(m,n)$, situat în interiorul regiunii. În cazul regimului staționar, și în absența surselor interioare de căldură, schimbul de căldură este determinat exclusiv de conducția între nodul $N(m,n)$ și cele patru noduri vecine.

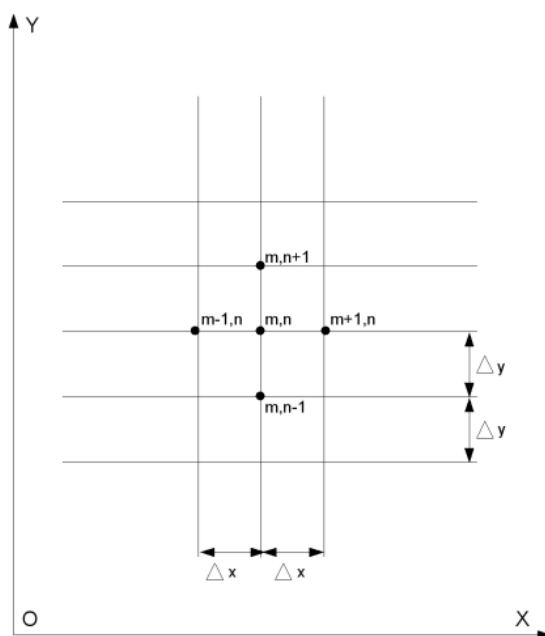


Figura 2.2: Discretizarea domeniului

Pornind de la forma diferențială:

$$\frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} = 0 \quad (2.18)$$

Se consideră:

$$\frac{\partial^2 t}{\partial x^2} \big|_{m,n} = \frac{\frac{\partial t}{\partial x} \big|_{m+\frac{1}{2},n} - \frac{\partial t}{\partial x} \big|_{m-\frac{1}{2},n}}{\Delta x} \quad (2.19)$$

$$\frac{\partial t}{\partial x} \big|_{m+\frac{1}{2},n} = \frac{t_{m+1,n} - t_{m,n}}{\Delta x} \quad (2.20)$$

$$\frac{\partial t}{\partial x} \big|_{m-\frac{1}{2},n} = \frac{t_{m,n} - t_{m-1,n}}{\Delta x} \quad (2.21)$$

$$\frac{\partial^2 t}{\partial x^2} \big|_{m,n} = \frac{t_{m+1,n} + t_{m-1,n} - 2t_{m,n}}{(\Delta x)^2} \quad (2.22)$$

$$\frac{\partial^2 t}{\partial y^2} \big|_{m,n} = \frac{t_{m,n-1} + t_{m,n+1} - 2t_{m,n}}{(\Delta y)^2} \quad (2.23)$$

$$\Rightarrow t_{m,n} = \frac{1}{4}(t_{m,n+1} + t_{m,n-1} + t_{m+1,n} + t_{m-1,n}) \quad (2.24)$$

Dacă se aplică raționamentul anterior se poate obține și aproximarea prin diferențe finite a ecuației diferențiale a conducției termice pentru sisteme bidimensionale în regim constant cu surse interioare de căldură

$$\frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} + \frac{q_v}{\lambda} = 0 \quad (2.25)$$

Se aproximează:

$$\frac{t_{m+1,n} + t_{m-1,n} - 2t_{m,n}}{(\Delta x)^2} + \frac{t_{m,n-1} + t_{m,n+1} - 2t_{m,n}}{(\Delta y)^2} + \frac{q_v}{\lambda} = 0 \quad (2.26)$$

Dacă $\Delta x = \Delta y$ (grilă pătratică)

$$t_{m,n} = \frac{1}{4}(t_{m,n+1} + t_{m,n-1} + t_{m+1,n} + t_{m-1,n} + \frac{q_v(\Delta x)^2}{\lambda}) \quad (2.27)$$

Gauss-Seidel După ce au fost stabilite rețeaua nodală și ecuațiile caracteristice fiecărui nod, problema determinării distribuției de temperatură în sistemul considerat

se reduce la rezolvarea sistemului de ecuații algebrice liniare. Când acest sistem are un număr mare de ecuații cel mai eficient mod de rezolvare este acela care utilizează o metodă numerică iterativă.

Vom nota temperaturile nodurilor cu un singur indice pentru a ușura lizibilitatea. Unei rețele cu N noduri îi vor corespunde N ecuații nodale cu diferențe finite, în care temperaturile t_i , $i = 1, \bar{N}$ constituie necunoscutele. Acest sistem poate fi scris sub forma:

$$\begin{aligned} &a_{11}t_1 + a_{12}t_2 + \cdots + a_{1N}t_N = c_1 \\ &a_{21}t_1 + a_{22}t_2 + \cdots + a_{2N}t_N = c_2 \\ &\vdots \qquad \qquad \qquad \vdots \qquad \qquad \qquad \vdots \qquad \qquad \qquad \vdots \\ &a_{N1}t_1 + a_{N2}t_2 + \cdots + a_{NN}t_N = c_N \end{aligned} \tag{2.28}$$

Coeficienții $a_{11}, a_{12}, \dots, a_{NN}$ și termenii liberi c_1, \dots, c_N se determină din valorile mărimilor $\Delta x, \lambda, \alpha, t_2$

Metoda Gauss-Seidel folosește aproximații succesive pentru a rezolva un sistem diagonal de ecuații liniare. Numim diagonal, un sistem de ecuații dacă pentru fiecare ecuație i ($1 \leq i \leq N$) valoarea absolută a coeficientului a_{ii} este mai mare decât valoarea absolută a oricărui alt coeficient a_{ij} ($1 \leq j \leq N, i \neq j$). Se presupune că sistemul (28) este diagonal și se explicitează din fiecare ecuație a sistemului variabila cu cel mai mare coeficient.

[illegible]

Etapele următoare ale calculului sunt:

1. alegerea unui set de valori pentru temperaturile necunoscute $t_i, i = 1, \bar{N}$.
Alegerea acestor variabile poate fi făcută pe baza unor aproximări sau intuitiv.
2. Procesul iterativ ce constă din repetarea calculului temperaturii nodale pe baza ecuațiilor din sistemul (28). În membrii secunzi ai acestor ecuații se introduc ultimele valori aproximative determinate pentru temperaturi. La pasul K vom avea:

$$t_i^{(k)} = \frac{c_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} t_j^{(k-1)} - \sum_{j=i+1}^N \frac{a_{ij}}{a_{ii}} t_j^{(k-1)}$$

În cazul primei iterații ($k=1$) $t_j^{k-1} = t_j^0$ reprezintă temperaturile alese inițial.

3. Se repetă calculul până ce se obține o diferență între valorile a două aproximări succesive, mai mică sau egală cu o valoare ε impusă.

$$| t_i^{(k)} - t_i^{(k-1)} | \leq \varepsilon$$

Metoda Gauss-Seidel este convergentă și în situația în care sistemul considerat nu este diagonal, dar rapiditatea convergenței este mult diminuată.

2.4 Transmiterea căldurii prin conducție în regim nestaționar multidimensional

2.4.1 Abordare analitică

Ecuția generală a conductivității termice este de forma:

$$\frac{\partial t}{\partial \tau} = a \nabla^2 t = \frac{q - v}{\rho c} \quad \left(a = \frac{\lambda}{\rho c}\right) \quad (2.30)$$

Integrarea acestei ecuații este posibilă numai dacă se cunosc condițiile de unicitate care determină problema în timp și spațiu:

1. condiția de spațiu precizează distribuția temperaturii în corp la timpul inițial τ_0 (de obicei $\tau_0 = 0$) : $t_0 = f(x, y, z)$
2. condiția de timp prezintă modul în care evoluează în timp schimbul de căldură cu mediul ambient
3. condiția de contur definește legătura corpului cu mediul ambient precizând forma exterioară a corpului și dimensiunile acestuia

Vom considera că temperatura inițială este distribuită uniform în toată masa corpului și vom demonstra că se poate determina temperatura în orice punct situat pe axele sau în centrul acestor corpuri. Dacă se consideră corpuri având formă de prisma tridimensională (cu axele O_x, O_y, O_z) cu dimensiunile $2\delta_x, 2\delta_y, 2\delta_z$, cu temperatura inițială t_c diferită de temperatura mediului ambiant, notată t_a . Rezultă astfel o diferență de temperatură $\Theta_c = t_c - t_a$

Temperatura într-un punct oarecare se poate determina pornind de la relația pentru placa plană :

$$\frac{\Theta}{\Theta_c} = \left(\frac{\Theta}{\Theta_c}\right)_x \cdot \left(\frac{\Theta}{\Theta_c}\right)_y \cdot \left(\frac{\Theta}{\Theta_c}\right)_z = \frac{t_p(x, y, z) - t_a}{t_c - t_a} \quad (2.31)$$

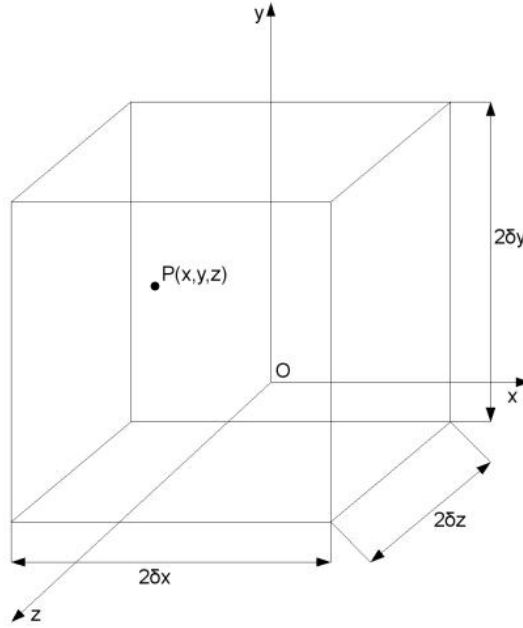


Figura 2.3: Exemplu corp

$$\begin{aligned}\left(\frac{\Theta}{\Theta_c}\right)_x &= \frac{t_x - t_a}{t_c - t_a} \\ \left(\frac{\Theta}{\Theta_c}\right)_y &= \frac{t_y - t_a}{t_c - t_a} \\ \left(\frac{\Theta}{\Theta_c}\right)_z &= \frac{t_z - t_a}{t_c - t_a}\end{aligned}$$

Pornind de la ecuația generală a conducției termice și considerând că există și surse interne de căldură putem scrie:

$$\frac{\partial t}{\partial \tau} = a \cdot \left(\frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} + \frac{\partial^2 t}{\partial z^2} \right) + \frac{q_v}{\rho c} \quad (2.32)$$

Am notat cu $q_v [\frac{W}{m^3}]$ densitatea volumică de flux a sursei interne de căldură. Aceste ecuații nu pot fi integrate printr-o metodă analitică nici măcar pentru cazul unidimensional. Mai departe vor fi prezentate cazuri particulare de corpuri geometrice.

Pentru a studia transferul de căldură prin conducție prin

Placa plană subțire infinită cu surse interne de căldură se presupune că la momentul $\tau = 0$ acestea din urmă devin active. Vom considera sursele de căldură uniform

distribuite în volumul plăcii. Se notează cu $Q[W]$ fluxul de căldură degajată de sursa internă de căldură și cu $V[m^3]$ volumul plăcii plane.

$$q_v = \frac{Q}{V} \quad \left[\frac{W}{m^3} \right]$$

Se notează cu $\delta[m]$ grosimea plăcii plane, subțire, având la momentul $\tau = 0$ o temperatură oarecare $t = t_c$ distribuită uniform. Temperatura mediului ambiant t_a este constantă pe ambele fețe ale plăcii. Schimbul de căldură între placă și mediul ambiant are loc în intervalul $\Delta\tau$. Notăm cu $\alpha[\frac{W}{m^2K}]$ coeficientul mediu de transfer termic. Astfel se pot scrie relațiile pentru un interval de timp foarte mic $\Delta\tau$:

1. $dq_1 = \dot{q}_v \cdot (\delta \cdot 1)d\tau$: căldura degajată de sursele interne de căldură pentru un volum de placă având aria suprafeței $1m^2$
2. $dq_2 = mcdt = \delta\rho d\tau$: fracțiune din dq_2 acumulată în placă ce crește astfel temperatura plăcii. Am notat cu ρ masa specifică și cu c capacitatea termică specifică. Se consideră că acestea sunt valori medii pe intervalul $\Delta\tau$ deci sunt constante.
3. $dq_3 = 2 \cdot \alpha(t - t_a)d\tau$: căldura cedată prin convecție sau radiație mediului ambiant. Deoarece placa are lungime infinită și grosime foarte mică se consideră, pentru simplitate, că transferul de energie termică se realizează numai prin cele două fețe laterale ale plăcii.

Temperatura t este temperatura suprafeței plăcii, aproximativ egală cu temperatura în mijlocul plăcii. Pentru un interval de timp $\Delta\tau$ se poate scrie bilanțul termic:

$$dq_1 = dq_2 + dq_3$$

$$d\dot{q}_v d\tau = \delta\rho c dt + 2 \cdot \alpha(t - t_a)d\tau$$

Notăm $t - t_a = \theta \Rightarrow dt = d(t - t_a) = d\theta \Rightarrow (d\dot{q}_v - 2\alpha\theta)d\tau = \delta\rho c d\theta$. Prin separarea variabilelor τ și θ se obține:

$$d\tau = \frac{d\theta}{\frac{\dot{q}_v}{\rho c} - \frac{2\alpha}{\rho c\delta}\theta}$$

Integrând și notând numitorul relației cu u se obține:

$$u = \frac{\dot{q}_v}{\rho c} - \frac{2\alpha}{\rho c \delta} \theta$$

Prin diferențiere se obține:

$$du = -\frac{2\alpha}{\rho c \delta} d\theta = -md\theta \quad (m = \frac{2\alpha}{\rho c \delta} = ct)$$

$$d\tau = \frac{d\theta}{U} = -\frac{du}{mu} \quad \text{sau} \quad -\frac{du}{u} = md\tau$$

Integrând relația se obține:

$$\ln u = -m\tau - C_1$$

Notăm cu $C_1 \ln u_0$

$$\ln u = \ln u_0 = -m\tau \quad \text{sau} \quad u = u_0 e^{m\tau}$$

pentru $\tau = 0 \Rightarrow \theta = \theta_c = t_c - t_a \Rightarrow$

$$u_0 = \frac{\dot{q}_v}{\rho c} - \frac{2\alpha}{\rho c \delta} \theta_c \Rightarrow$$

$$\theta = \theta_c e^{-m\tau} + \frac{\dot{q}_v}{2\alpha} (1 - e^{-m\tau})$$

Dacă $\tau \rightarrow \infty$ se obține condiția de regim staționar $\theta = \frac{\dot{q}_v \delta}{2\alpha}$

2.4.2 Metode numerice de rezolvare a conducției termice tranziatorii

Datorită faptului că în practică formele geometrice ale corpurilor pentru care calculăm conducția în regim nestaționar sunt de cele mai multe ori complexe, iar în multe situații chiar și condițiile la limită variază în timp, probleme de acest tip sunt rezolvate folosind metode numerice de calcul, implementate prin algoritmi computaționali specializați. Această metodă, folosită în speță pentru domenii și sisteme bidimensionale sau tridimensionale, prezintă foarte multe avantaje în contrast cu alte metode folosite pentru calculul conducției termice, cum ar fi, de exemplu, metoda lui Dussanberre sau metoda grafică a lui Schmidt.

Metode pentru domenii bidimensionale

Se presupune o secțiune plană a corpului prin care se face transferul de căldură de tip nestaționar, pe direcțiile O_x și O_y . Discretizarea corpului se face împărțind planul într-o rețea de noduri, pentru care poziția pe axa O_x este dată de indicele m , iar poziția pe axa O_y este dată de indicele n . Astfel, ecuația diferențială a conductivității termice în regim nestaționar devine:

$$\frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} = \frac{1}{a} \cdot \frac{\partial t}{\partial \tau} \quad (2.33)$$

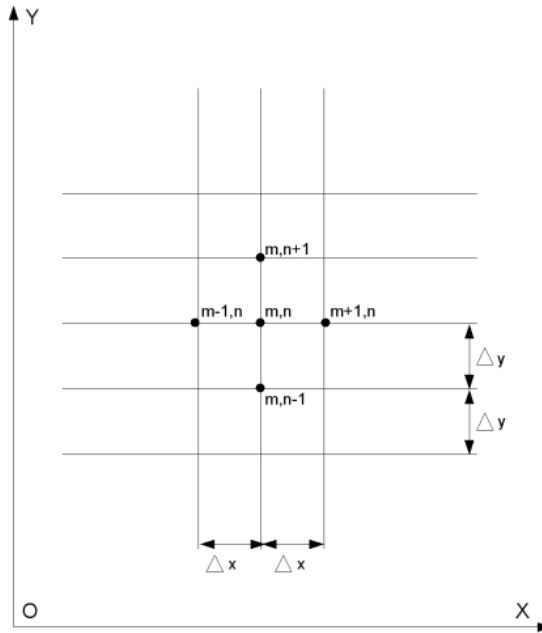


Figura 2.4: Discretizarea domeniului

Utilizând ecuația (33) vom determina temperaturile în toate centrele volumelor finite. Pentru aproximarea pașilor de divizare a coordonatelor de temperatură și spațiu se folosesc diferite metode: metoda diferențelor finite (MDF), metoda elementelor finite (MEF) sau metoda volumelor finite (MVF).

Metoda Diferențelor Finite

Granularitatea unei discretizări, dependentă de valoarea pasului între centrele elementelor de volum (sau lungimea barelor fictive), influențează direct exactitatea soluției obținute. Cu cât avem pași mai mulți și mai mici, cu atât crește acuratețea valorilor obținute prin calcul în raport cu valorile reale.

Punctele nodale, adică centrele elementelor volumetrice, vor avea temperaturile $t_{0,0}$, $t_{0,1}$, ... , $t_{1,0}$, $t_{1,1}$, ... , $t_{m,n}$, $t_{m,n+1}$, ..., $t_{m+1,n}$, ... , etc. Folosind aceste notații putem scrie gradientii de temperatură astfel:

$$\begin{aligned}\frac{\partial t}{\partial x}|_{m+\frac{1}{2},n} &\cong \frac{t_{m+1,n} - t_{m,n}}{\Delta x} \\ \frac{\partial t}{\partial y}|_{m,n+\frac{1}{2}} &\cong \frac{t_{m,n+1} - t_{m,n}}{\Delta y} \\ \frac{\partial t}{\partial x}|_{m-\frac{1}{2},n} &\cong \frac{t_{m,n} - t_{m-1,n}}{\Delta x} \\ \frac{\partial t}{\partial y}|_{m,n-\frac{1}{2}} &\cong \frac{t_{m,n} - t_{m,n-1}}{\Delta y}\end{aligned}$$

Corpul considerat este discretizat într-un număr finit de elemente volumetrice mici, pentru fiecare element în parte considerându-se că spațiul ocupat de acesta are aceeași temperatură ca și centrul elementului. Putem considera astfel, la nivel abstract, că sistemul fizic real devine o rețea de bare fictive conductoare termice care unesc centrele elementelor volumetrice.

Pentru M puncte alese se obțin prin acest procedeu M ecuații rezolvabile prin metode numerice.

Derivatele parțiale din ecuația (33) se pot aproxima cu valorile:

$$\begin{aligned}\frac{\partial^2 t}{\partial x^2}|_{m,n} &\cong \frac{\frac{\partial t}{\partial x}|_{m+\frac{1}{2},n} - \frac{\partial t}{\partial x}|_{m-\frac{1}{2},n}}{\Delta x} = \frac{t_{m+1,n} + t_{m-1,n} - 2t_{m,n}}{(\Delta x)^2} \\ \frac{\partial^2 t}{\partial y^2}|_{m,n} &\cong \frac{\frac{\partial t}{\partial y}|_{m,n+\frac{1}{2}} - \frac{\partial t}{\partial y}|_{m,n-\frac{1}{2}}}{\Delta y} = \frac{t_{m,n+1} + t_{m,n-1} - 2t_{m,n}}{(\Delta y)^2}\end{aligned}$$

iar derivata în raport cu timpul din ecuația (33) se poate aproxima:

$$\frac{\partial t}{\partial \tau} = \frac{t_{m,n}^{p+1} - t_{m,n}^p}{\Delta \tau}$$

Luând în considerare ecuațiile de mai sus, putem aproxima ecuația (33) cu:

$$\frac{t_{m+1,n}^p + t_{m-1,n}^p - 2t_{m,n}^p}{(\Delta x)^2} + \frac{t_{m,n+1}^p + t_{m,n-1}^p - 2t_{m,n}^p}{(\Delta y)^2} = \frac{t_{m,n}^{p+1} - t_{m,n}^p}{a\Delta\tau}$$

unde $t_{m,n}^p$ este temperatura nodului (m, n) la momentul p , iar $t_{m,n}^{p+1}$ este temperatura nodului (m, n) la momentul $p + \Delta\tau$, adică la momentul $p + 1$.

Putem deci determina temperaturile fiecărui nod după scurgerea fiecărui $\Delta\tau$, cu condiția ca la timpul τ oarecare să fie cunoscute valorile temperaturilor pentru nodurile: $t_{m,n}^p$, $t_{m+1,n}^p$, $t_{m-1,n}^p$, $t_{m,n+1}^p$, $t_{m,n-1}^p$.

Iterând ecuațiile de N ori, obținem distribuția temperaturii după N momente de timp, adică distribuția la $N\Delta\tau$. Alegând nodurile ca fiind echidistante, adică să satisfacă $\Delta x = \Delta y$, vom obține de fapt:

$$t_{m,n}^{p+1} = \frac{a\Delta\tau}{(\Delta x)^2} \cdot (t_{m+1,n}^p + t_{m-1,n}^p + t_{m,n+1}^p + t_{m,n-1}^p) + \left(1 - \frac{4a\Delta\tau}{(\Delta x)^2}\right) \cdot t_{m,n}^p \quad (2.34)$$

Diviziunile de timp și spațiu sunt astfel alese astfel încât:

$$\frac{a\Delta\tau}{(\Delta x)^2} = \frac{1}{M}$$

unde M poate lua valorile 2 sau 4.

Pentru $M = 4$, ecuația (34) devine:

$$t_{m,n}^{p+1} = \frac{t_{m+1,n}^p + t_{m-1,n}^p + t_{m,n+1}^p + t_{m,n-1}^p}{4} \quad (2.35)$$

determinându-se astfel valoarea temperaturii din nodul $N(m, n)$ prin efectuarea mediei aritmetice a temperaturilor celor 4 noduri vecine de pe cele 2 direcții. Aceeași valoare a lui M , determină pasul de timp:

$$\Delta\tau = \frac{(\Delta x)^2}{a \cdot M}$$

Pentru un sistem unidirecțional, ecuația (34) devine:

$$t_m^{p+1} = \frac{a\Delta\tau}{(\Delta x)^2} \cdot (t_{m+1}^p + t_{m-1}^p) + \left(1 - \frac{2a\Delta\tau}{(\Delta x)^2}\right) \cdot t_m^p$$

Pentru un astfel de sistem facem alegerea $M = 2$ și $\Delta\tau = \frac{(\Delta x)^2}{2a}$ vom avea:

$$t_m^{p+1} = \frac{t_{m+1}^p + t_{m-1}^p}{2} \quad (2.36)$$

deci valoarea nodului se obține efectuând media aritmetică a temperaturilor celor 2 nodurilor vecine de pe direcția dată.

Alegerea valorii pentru M devine foarte importantă sub aspectul stabilității calculelor și influențează direct complexitatea calculului. De exemplu, dacă alegem $M < 2$ pentru un sistem unidimensional sau $M < 4$ pentru un sistem bidimensional, coeficientul pentru $t_{m,n}^p$ sau t_m^p , din ecuația (35), respectiv ecuația (36), devine negativ. Acest lucru înseamnă că la iterarea calculului, dacă vecinii nodului m au valori mai mici decât $t_{m,n}^p$, respectiv t_m^p , după scurgerea lui $\Delta\tau$, temperatura $t_{m,n}^{p+1}$, respectiv t_m^{p+1} , vor scădea, fapt care este imposibil. Prin urmare, M devine *parametru de stabilitate*, pentru care introducem următoarele restricții:

$$M \geq 2, \text{ pentru sisteme unidimensionale}$$

$$M \geq 4, \text{ pentru sisteme bidimensionale}$$

Capitolul 3

Detalii de implementare

Deoarece soluția de simulare a transferului de căldura prezentată în această lucrare se dorește a fi parte integrantă dintr-un pachet software de simulare extins, a fost necesară păstrarea unei relative uniformități în modul de funcționare a acestora. Datorită diferențelor majore din punct de vedere fizic și implicit și numerice între cazul de simulare în regim staționar și simularea în regim nestaționar acestea au fost implementate separat. Pentru a veni în întâmpinarea unor categorii cât mai variate de utilizatori și pentru a face posibilă utilizarea simulatoarelor atât pe o arhitectură mono-procesor, multi-procesor (multi-core) sau pe o arhitectura distribuită, pentru fiecare regim de funcționare (staționar sau nestaționar) a fost implementată o soluție de simulare folosind API-uri dedicate. Pentru sisteme de tip mono-procesor cu un singur sau mai multe core-uri, implementarea este dezvoltată folosind API-ul OpenMP (Open Multi-Processing). Pentru sisteme distribuite cu noduri de procesare mono sau multi-procesor implementarea s-a făcut folosind API-ul MPI (Message Passing Interface), iar pentru un câștig suplimentar în viteză de procesare aplicația a fost implementată și folosind cele două API-uri împreună.

API-ul OpenMP suportă programare paralelă multi-platform shared-memory în C/C++ având la dispoziție orice arhitectură ce rulează Unix sau Windows. Rezultat al colaborării celor mai mari dezvoltatori de hardware și software, OpenMp este portabil și oferă programatorilor o interfață flexibilă dar în același timp intuitivă

pentru a-și dezvolta aplicații pentru platforme ce pot varia de la simple desktop-uri la super-calculatoare.

MPI este o interfață de programare a aplicațiilor folosind comunicația prin mesaje, și reprezintă, de fapt, un protocol de comunicație independent de limbaj, folosit în programarea distribuită. Acesta suportă atât comunicația punct-la-punct, cât și comunicare în cadrul unui grup. Sunt specificate deasemenea reguli semantice și comportamentale. MPI este la momentul actual modelul dominant în high-performance computing datorită performanțelor, scalabilității și portabilității crescute pe care le oferă. Modelele de programare bazată pe thread-uri și memorie partajată, precum OpenMP și modelele de programare folosind comunicația prin mesaje, precum MPI, pot fi considerate complementare și vor fi utilizate împreună în cadrul acestei aplicații deoarece se pliază pe o arhitectură cu multiple noduri de tip shared-memory (cluster).

3.1 Regim staționar

Soluția implementată își propune simularea transferului de căldură pe un domeniu 2D de formă pătratică. Aceste ipoteze simplificatoare sunt justificate de existența unor aplicații software capabile să discretizeze domenii de orice formă sau dimensiune în subdomenii de formă pătratică. Folosind schema de discretizare cu diferențe finite centrate se obține o grilă pătratică de puncte pentru care se calculează distribuția de temperatură. Acest lucru poate fi simulat în mod natural cu o matrice pătratică ale cărei valori $uc[i][j]$ corespund temperaturii în punctul de coordonate (i,j) .

Datorită necesității de integrare cu alte aplicații software și pentru a putea acoperi prin simulare toate cazurile ce pot fi întâlnite în procesul fizic toți parametrii necesari rulării simulării sunt furnizați aplicației printr-un fișier de intrare configurabil de către utilizator. S-a constituit astfel ideea de scenariu de test care să cuprindă o specificare completă a parametrilor de intrare și a algoritmului de soluționare numerică dorit pentru rezolvare. Pentru ușurință în testare și pentru alcătuirea de

statistici care de cele mai multe ori sunt deosebit de necesare, un fișier de intrare poate include mai multe scenarii de test. Un exemplu de fișier de intrare:

```
[NUM_SCENARIO] = 1

[SCENARIO] = 0
[OMP_THREADS] = 2
[SIZEX] = 100
[SIZEY] = 100
[BETA] = 1.65
[H] = 1
[EPS] = 0.0000001
[MAXITER] = 1000000
[ALG] = P_GAUSS_SEIDEL_RELAX
[SOURCEX] = 0.2
[SOURCEY] = 0.2
[SOURCERADIUS] = 4
[SOURCETEMP] = 1.2
[FUNC] = 3 * x ^ 2 - 3 * y ^ 2 + 10 * x
```

Parametrii unui scenariu de test sunt:

[SCENARIO]: indică numărul scenariului curent

[OMP_THREADS]: indică numărul de threaduri create de OpenMp pentru paralelizarea regiunilor critice

[SIZEX],[SIZEY]: indică dimensiunile domeniului.

[BETA]: reprezintă coeficientul de relaxare necesar algoritmului Gauss-Seidel cu supra-relaxare.

[H]: reprezintă valoarea raportului $\frac{\lambda}{\rho c}$ pentru domeniul considerat.

[EPS]: este criteriu de convergență pentru regimul staționar reprezentând eroarea maximă admisibilă.

[MAXITER]: reprezintă numărul maxim de iterații admisibil pentru atingerea condiției de convergență.

[ALG]: reprezintă metoda numerică folosită pentru rezolvarea sistemului de ecuații cu diferențe finite.

[SOURCEX],[SOURCEY]: reprezintă coordonatele centrului sursei de caldură, relativi la colțul din stânga sus al domeniului și la dimensiunile maxime ale acestuia.

[SOURCETEMP]: reprezintă densitatea de flux termic a sursei.

[FUNC]: reprezintă funcția cu care se inițializează frontiera domeniului.

În cazul regimului termic invariant în timp, pentru calculul distribuției de temperatură se aproximează ecuația $\vec{\nabla} \left(\lambda \vec{\nabla} t \right) + q_v = 0$. Aproximarea prin diferențe finite a ecuației diferențiale a conducției termice pentru sisteme bidimensionale în regim constant cu surse interioare de căldură în cazul unei grile pătratice este: $t_{m,n} = \frac{1}{4}(t_{m,n+1} + t_{m,n-1} + t_{m+1,n} + t_{m-1,n} + \frac{q_v(\Delta x)^2}{\lambda})$. Rezultă astfel un sistem liniar de ecuații ce poate fi rezolvat numeric prin metode iterative Jacobi, Gauss-Seidel, Gauss-Seidel cu suprarelaxare. În funcție de API-ul folosit, se poate alege metoda numerică dorită prin specificarea ei în campul [ALG] din scenariu. În cadrul aceluiași scenariu se pot folosi una, două sau toate trei metodele numerice prezentate mai sus.

Pentru varianta serială [ALG] poate lua valorile:

S_JACOBI (rezolvat prin funcția):

```
double s_jacobi(double **uc, double **ua, double **q, double h, int nx, int ny)
{
1   ....
2   while ((maxrez >= EPS) && (step <= MAX_ITER))
3   {
4   ....
5   for(i=1;i<ny-1;i++)
6   for(j=1;j<nx-1;j++)
7   {
8   uc[i][j] = 0.25 * (q[i][j]*h*h + ua[i-1][j] + ua[i][j-1] +
9   ua[i+1][j] + ua[i][j+1]);
10  rez = fabs(ua[i][j]-uc[i][j]);
11  if(maxrez < rez) maxrez = rez;
12  }
13  xchg = ua;
14  ua = uc;
15  uc = xchg;
16  step++;
17 }
18 ....
19 }
```

S_GAUSS_SEIDEL (rezolvat prin funcția):

```
double s_gauss_seidel(double **uc, double **q, double h, int nx, int ny)
{
1   ....
2   while ((maxrez >= EPS) && (step <= MAX_ITER))
3   {
4   ....
```

```

5     for(i=1;i<ny-1;i++)
6         for(j=1;j<nx-1;j++)
7         {
8             ug = 0.25 * (q[i][j]*h*h + uc[i-1][j] + uc[i][j-1] +
9                         uc[i+1][j] + uc[i][j+1]);
10            rez = fabs(uc[i][j]-ug);
11            if(maxrez < rez) maxrez = rez;
12            uc[i][j] = ug;
13        }
14    step++;
15 }
16 ....
}

```

S_GAUSS_SEIDEL_RELAX (rezolvat prin funcția):

```

double s_gauss_seidel_relax(double **uc,double beta,double **q,double h,int nx,int ny)
{
1     ....
2     while ((maxrez >= EPS) && (step <= MAX_ITER))
3     {
4         ....
5         for(i=1;i<ny-1;i++)
6             for(j=1;j<nx-1;j++)
7             {
8                 ug = 0.25 * (q[i][j]*h*h + uc[i-1][j] + uc[i][j-1] +
9                             uc[i+1][j] + uc[i][j+1]);
10                rez = fabs(uc[i][j]-ug);
11                if(maxrez < rez) maxrez = rez;
12                uc[i][j] = (1-beta)*uc[i][j] + beta*ug;
13            }
14        step++;
15    }
16    ....
}

```

Pentru varianta utilizând API-ul OpenMp, [ALG] poate lua valorile:

P_JACOBI, *P_GAUSS_SEIDEL*, *P_GAUSS_SEIDEL_RELAX*. Implementarea numerică este similară cu cea serială, adăugându-se pentru execuția paralelă pragmele specifice OpenMP, înaintea instrucțiunilor de tip "for":

```
#pragma omp parallel for private(i,j)
```

Pentru varianta utilizând API-ul MPI [ALG], poate lua valorile:

M_JACOBI, *M_GAUSS_SEIDEL*, *M_GAUSS_SEIDEL_RELAX*. Implementarea numerică a algoritmului propriu zis este identică cu varianta utilizând OpenMP atunci când vine vorba de calculul efectiv. Rank-ul 0 va face o împărțire a domeniului inițial pe care algoritmul se execută, trimițând celorlalte rankuri subdomenii (grupuri de linii) pe care să le prelucreze.

Rezolvarea problemei Dirichlet pentru ecuația lui Laplace constă în stabilirea unor valori fixe pentru marginile domeniului de calcul, valori obținute folosind o funcție continuă cu derivata de ordinul II egală cu 0. Funcția respectivă va fi folosită și pentru determinarea valorilor din interiorul domeniului. Posibilitatea introducerii (schimbării) expresiei acesteia de către utilizator este esențială pentru caracterul aplicației. Utilizatorul poate introduce expresia acesteia în câmpul [FUNC] din cadrul scenariului. Forma de introducere este cea infixată, cu mențiunea că atât operanzii cât și operatorii funcției trebuie introduși cu spațiu între ei din rațiuni ce țin de parsarea fișierului de intrare. Forma infixată este transformată în formă poloneză inversată prin funcția *reverse_polish_notation*, pentru a putea fi evaluată numeric, cu ajutorul funcțiilor *eval* și *calculate*. Aceasta din urmă efectuează calculul operațiilor simple, (de exemplu: adunare, scădere, înmulțire, împărțire, ridicare la putere) cu doi operanzi, rezultate din parcurgerea formei postfixate.

```

1  for(i=0;i<ny;i+=1)
2  {
3    uc[i][0] = ua[i][0] = func(i,0,func_polish);
4    uc[i][nx-1] = ua[i][nx-1] = func(i,nx-1,func_polish);
5  }
6  for(i=0;i<nx;i+=1)
7  {
8    uc[0][i] = ua[0][i] = func(0,i,func_polish);
9    uc[ny-1][i] = ua[ny-1][i] = func(ny-1,i,func_polish);
10 }
```

După calcularea valorilor pentru frontiera domeniului și inițializarea matricei corespunzătoare acestuia, se inițializează și se completează matricea corespunzătoare valorilor fluxului sursei de căldură. Acest lucru se calculează verificând dacă punctul de coordonate (i,j) se află la o distanță mai mică decât raza specificată față de centrul sursei.

Algoritmii de rezolvare a sistemului de ecuații liniare sunt aleși în funcție de valorile din vectorul `scenario[scn_index].alg[i]`. Parcurgând elementele acestui vector se verifică care sunt nenule și se apelează funcția corespunzătoare algoritmului ales. Dacă mai există mai mult de o valoare nenulă în acest vector, se reapelează funcția de inițializare a matricilor corespunzătoare domeniului și valorilor fluxului sursei de

căldură pentru a asigura aplicarea corectă a noului algoritm numeric pe domeniul inițial.

Procesul de calcul, indiferent de algoritmul numeric utilizat, se încheie în momentul atingerii condiției de oprire. Condiția de oprire reprezintă conjuncția dintre condiția de stabilitate și atingerea numărului maxim de iterații admis. Condiția de stabilitate constă în obținerea unei diferențe între valorile de temperatură corespunzătoare unui punct, la două iterații succesive, mai mică decât valoarea parametrului [EPS] primit în cadrul scenariului. Numărul maxim de iterații admise este transmis prin scenariu în câmpul [MAXITER].

Având în vedere caracterul lucrării, simularea transferului de căldură, o modalitate facilă de reprezentare a rezultatelor obținute este deosebit de importantă. Din acest motiv, aplicația este capabilă să furnizeze la ieșire datele reprezentate în formatul VTK (Vizualization Toolkit), care poate fi apoi folosit pentru vizualizare în aplicații precum Paraview.

Am folosit formatul particular al Vizualization Toolkit (VTK) deoarece acesta este un pachet software open-sorce pentru procesare și vizualizare de grafică 2D și 3D, folosit pe scară largă în cercetare și industrie. VTK suportă o varietate largă de algoritmi de vizualizare, incluzând metode scalare, vectoriale, volumetrice, suportă procesare paralelă și este independent de platformă putând rula pe sisteme UNIX, Windows, MacOS.

Formatul VTK este următorul:

# vtk DataFile Version 2.0	(1)
ASCII string describing the data goes here.	(2)
ASCII BINARY	(3)
DATASET type	(4)
....	
POINT_DATA n	(5)
....	
CELL_DATA n	
....	

- (1): Conține versiunea fisierului și identificatorul

- (2): Conține o descriere a fișierului printr-un string de maxim 256 de caractere.
- (3): Conține formatul fișierului (poate fi: ASCII sau BINARY)
- (4): Conține tipul structurii de date. VTK suportă cinci formate diferite:

STRUCTURED_POINTS, STRUCTURED_GRID, RECTILINEAR_GRID, POLYDATA, UNSTRUCTURED_GRID.

- (5): Conține atributele structurii

În cadrul aplicației exportul datelor către fișierul .VTK se realizează prin funcția *export_to_vtk* apelată în momentul în care s-a atins condiția de stabilitate.

```
{
1  long int i,j;
2  fprintf(f,"# vtk DataFile Version 2.0\n");
3  fprintf(f,"This is a test file for the vtk format file export\n");
4  fprintf(f,"ASCII\n");
5  fprintf(f,"DATASET UNSTRUCTURED_GRID\n\n");
6  fprintf(f,"POINTS %d double\n",(nx+1)*(ny+1));
7  for(i=0;i<ny+1;i++)
8    for(j=0;j<nx+1;j++)
9      fprintf(f,"%f %f %f\n",(double)i/(double)ny,(double)j/(double)nx,0.0);
10 fprintf(f,"nCELLS %d %d\n",nx*ny,5*(nx*ny));
11 for(i=0;i<ny;i++)
12   for(j=0;j<nx;j++)
13     fprintf(f,"4 %ld %ld %ld %ld\n",j+i*nx,i,j+(i+1)*nx,i+2,j+(i+1)*nx,i+1);
14 fprintf(f,"nCELL_TYPES %d\n",nx*ny);
15 for(i=0;i<ny*nx;i++)
16   fprintf(f,"9 ");
17 fprintf(f,"nCELL_DATA %d\n",nx*(ny));
18 fprintf(f,"SCALARS temp FLOAT\n");
19 fprintf(f,"LOOKUP_TABLE values_table\n");
20 for(i=0;i<ny;i++)
21   for(j=0;j<nx;j++)
22     fprintf(f,"%f\n",mat[i][j]);
23 fclose(f);
}
```

Având în vedere că aplicația permite introducerea mai multor scenarii simultan cu dimensiuni ale domeniului analizat diferite, cu diferite valori ale fluxului sursei sau ale valorilor de la marginile domeniului, o modalitate de alcătuire a unor grafice sau statistici ale rezultatelor obținute, este foarte importantă. Folosind fișierul de ieșire corespunzător în Gnuplot se poate vizualiza o comparație între timpii de rulare pentru diferite mărimi ale domeniului considerat și se poate determina speedup-ul obținut prin diversele modalități de paralelizare. A fost ales acest format și

program pentru portabilitate și usurința în folosire. Gnuplot este un utilitar în linie de comandă pentru generarea de grafice, independent de platformă (UNIX, Windows, MacOS)

Exportul pentru gnuplot se realizează prin funcția *export_to_gnuplot*. Aceasta a fost apelată pentru a putea crea o statistică cu privire la timpii de rulare pentru fiecare algoritm numeric utilizat. Scop realizării acestei statistici pur didactic pentru alcătuirea unei statistici cu privire la speedup-ul obținut prin diversele metode de paralelizare. Mentru a asigura o manipulare mai ușoară a datelor și pentru a putea crea o statistică ușor de urmarit și interpretat înainte de exportul pentru gnuplot, vectorul de scenarii este sortat după dimeniunea domeniul de simulare.

```
....
1 switch(alg_type)
2 {
3     case M_JACOBI:
4         sprintf(path,"%s/export/m_jacobi.dat",buf);
5         break;
6     case M_GAUSS_SEIDEL:
7         sprintf(path,"%s/export/m_gauss_seidel.dat",buf);
8         break;
9     case M_GAUSS_SEIDEL_RELAX:
10        sprintf(path,"%s/export/m_gauss_seidel_relax.dat",buf);
11        break;
12    }
13    ....
14    fprintf(f,"%f    %f\n",cells,time);
```

Pentru optimizarea folosind calcul paralel am folosit OpenMP. Structura de simulare folosind scenarii a fost păstrată, la fel și logica de funcționare a programului. Pentru a putea crește eficiența prin paralelizare a fost analizat codul serial, stabilindu-se buclele care necesitau cel mai mult efort computațional.

Pentru optimizarea folosind calcul distribuit am folosit MPI. Pentru a putea păstra unitară modalitatea de simulare folosind scenarii, și în același timp, pentru a putea transmite tuturor nodurilor de calcul parametrii simulării, am creat un tip de date nou *MPI_SCENARIO* corespunzător structurii anterioare de tip *scenario_t* care poate fi trimis prin MPI.

```
1  offsets[0] = 0;
2  oldtypes[0] = MPI_INT;
3  blockcounts[0] = 4;
4  MPI_Type_extent(MPI_INT,&extent);
5  offsets[1] = 4*extent;
6  oldtypes[1] = MPI_DOUBLE;
7  blockcounts[1] = 8;
```

```

10 MPI_Type_extent(MPI_DOUBLE,&extent);
11 offsets[2] = offsets[1] + 8*extent;
12 oldtypes[2] = MPI_CHAR;
13 blockcounts[2] = 100;
14 MPI_Type_struct(3,blockcounts,offsets,oldtypes,&MPI_SCENARIO);
15 MPI_Type_commit(&MPI_SCENARIO);

```

Rankul 0 citește datele de intrare din fișier, inițializează matricea coresunzătoare domeniului și cea corespunzătoare valorilor fluxului sursei și trimite fragmentele corespunzătoare din acestea, împreună cu structura scenariu celorlalte rankuri. Deoarece schema de discretizare presupune ca valoarea temperaturii într-un punct din domeniu să fie calculată ca medie a valorilor temperaturii vecinilor de la coordonatele $(i+1,j)$ $(i-1,j)$ $(i,j-1)$ $(i,j+1)$ se pune problema consistenței datelor la marginile subdomeniilor atribuite fiecărui rank. Pentru a soluționa această problemă fiecare nod computațional va avea alocat un domeniu cu două linii mai mare decât cel pe care trebuie să îl proceseze. Pe prima linie (0) va primi de la nodul care procesează subdomeniul adiacent din partea superioară, ultima linie din domeniul acestuia, iar pe ultima linie (n_y+1) va primi de la nodul care procesează subdomeniul adiacent din partea inferioară prima linie din domeniul acestuia. La rândul său fiecare nod de procesare va trimite nodurilor ce procesează subdomeniile adiacente prima (1) respectiv ultima linie (n_y) din domeniul sau, în afara primului și ultimului nod.

Pentru a asigura consistența datelor primite și pentru eficiență sporită, comunicația între noduri se realizează pe baza parității rankului. Astfel un nod de procesare cu rank impar va trimite întâi "în sus" prima linie din domeniul său după care va aștepta să primească "de sus" noile valori pentru frontieră; va trimite apoi "în jos" ultima linie din domeniul său și va aștepta să primească "de jos" ultima linie pentru frontieră. Trebuie precizat că sincronizarea comunicației este implicită, cu ajutorul funcțiilor blocante.

```

1  if(rank%2 == 1 && rank != numtask-1)
2  {
3    MPI_Send(ua[1],nx,MPI_DOUBLE,rank-1,1,MPI_COMM_WORLD);
4    MPI_Recv(ua[0],nx,MPI_DOUBLE,rank-1,1,MPI_COMM_WORLD,&status);
5    MPI_Send(ua[local_ny],nx,MPI_DOUBLE,rank+1,1,MPI_COMM_WORLD);
6    MPI_Recv(ua[local_ny+1],nx,MPI_DOUBLE,rank+1,1,MPI_COMM_WORLD,&status);

```

```

7  }
8  else if(rank%2 == 0 && rank != numtask-1)
9  {
10 MPI_Recv(ua[local_ny+1],nx,MPI_DOUBLE,rank+1,1,MPI_COMM_WORLD,&status);
11 MPI_Send(ua[local_ny],nx,MPI_DOUBLE,rank+1,1,MPI_COMM_WORLD);
12 MPI_Recv(ua[0],nx,MPI_DOUBLE,rank-1,1,MPI_COMM_WORLD,&status);
13 MPI_Send(ua[1],nx,MPI_DOUBLE,rank-1,1,MPI_COMM_WORLD);
14 }

```

Deoarece condiția de stabilitate presupune ca diferența dintre valorile temperaturilor oicărui nod din domeniu, între doua iterații succesive, să fie mai mică decât [EPS], a fost necesară calcularea reziduurilor maxime locale, corespunzătoare fiecărui subdomeniu, care la fiecare iterație sunt trimise rankului 0. Pe baza acestora, rankul 0 determină reziduul maxim global pe care îl compară cu [EPS]. În momentul în care reziduul maxim global devine mai mic sau egal decât [EPS] înseamnă că s-a atins condiția de stabilitate și că nu mai sunt necesare iterații suplimentare. Rankul 0 comandă încetarea execuției algoritmului pentru celelalte rankuri și așteaptă de la acestea valorile actualizate pentru subdomeniile prelucrate de acestea.

Dacă nu se atinge condiția de stabilitate prin executarea a [MAXITER] iterații, rankul 0 comandă încetarea execuției algoritmului pentru celelalte rankuri și își actualizează valorile distribuției de temperatură. La sfârșitul execuției rankul 0 realizează exportul către VTK și Gnuplot.

3.2 Regim nestaționar

La fel ca în cazul regimului staționar soluția implementată își propune simularea transferului de căldură pe un domeniu 2D de formă pătratică. În acest caz însă de interes nu mai este distribuția temperaturii în mometul stabilizării, ci evoluția acestei distribuții în timp. Pentru o simulare a unui timp fizic suficient de mare, distribuția de temperatură evoluează spre configurația de regim staționar.

În cazul simulării pentru regim variant în timp, pentru calculul distribuției de temperatură se utilizează formula $\frac{\partial t}{\partial \tau} = a \nabla^2 t$. Aceasta este discretizată prin metoda diferențelor finite, schema explicită rezultând forma $t_{m,n}^{p+1} = \frac{a\Delta\tau}{(\Delta x)^2} \cdot (t_{m+1,n}^p + t_{m-1,n}^p + t_{m,n+1}^p + t_{m,n-1}^p) + (1 - \frac{4a\Delta\tau}{(\Delta x)^2}) \cdot t_{m,n}^p$. Această relație este folosită pentru calculul direct al temperaturii curente în punctul de coordonate(i,j) în funcție de media temperaturii vecinilor direcți și de valoarea anterioară (în urma cu $\Delta\tau$) a temperaturii în punctul curent.

Pentru a menține consistența în utilizarea simulatorului s-a păstrat ideea de scenariu de test care să cuprindă specificarea completă a parmetrilor de intrare. Pentru ușurință în testare și pentru alcătuirea de statistici care de cele mai multe ori sunt deosebit de necesare un fișier de intrare poate include mai multe scenarii de test, corespunzătoare unor dimensiuni diferite ale domeniului de simulare, ale funcției de pe frontiera domeniului, poziționării sursei, etc. Un exemplu de fișier de intrare pentru cazul nestaționar:

```
1  [NUM_SCENARIO] = 1
2  [OMP_THREADS] = 2
3  [SCENARIO] = 0
4  [SIZEX] = 100
5  [SIZEY] = 100
6  [H] = 1
7  [W] = 250
8  [MAX_TIME] = 3
9  [TIME_SLICE] = 0.001
10 [STEP] = 70
11 [SOURCEX] = 0.2
12 [SOURCEY] = 0.2
13 [SOURCERADIUS] = 4
14 [SOURCETEMP] = 1.2
15 [FUNC] = 3 * x ^ 2 - 3 * y ^ 2 + 10 * x
```

Similar cu parametrii scenariului pentru regim staționar, parametrii scenariului pentru regim nestaționar sunt:

[SCENARIO]:indică numărul scenariului curent

[OMP_THREADS]:indică numărul de threaduri create de OpenMp pentru paralelizarea regiunilor critice

[SIZEX],[SIZEY]:indică dimensiunile domeniului.

[H]:reprezintă valoarea raportului $\frac{\lambda}{(\Delta x)^2}$ pentru domeniul considerat.

[W]:reprezintă valoarea raportului $\frac{1}{\rho c}$

[MAX_TIME]:reprezintă timpul fizic simulat exprimat în secunde.

[TIME_SLICE]:reprezintă valoarea intervalului de timp între două iterații consecutive ($\Delta\tau$)

[STEP]:reprezintă numărul de iterații la care se face eșantionarea pentru vizualizare.

[SOURCEX],[SOURCEY]:reprezintă coordonatele centrului sursei de caldură, relativ la colțul din stânga sus al domeniului și la dimensiunile maxime ale acestuia.

[SOURCETEMP]:reprezintă densitatea de flux termic a sursei.

[FUNC]:reprezintă funcția cu care se inițializează frontiera domeniului.

Deoarece soluția implementată folosește schema explicită de discretizare cu diferențe finite centrate, pentru a asigura stabilitatea numerică a algoritmului, datele introduse trebuie să respecte următoarea restricție: $[H] \cdot [W] \cdot [TIME_SLICE] \leq \frac{1}{4}$. Ținând cont că λ , ρ , c sunt constante, pentru a asigura stabilitatea numerică trebuie acordată atenție deosebită alegerii pasului de discretizare a domeniului fizic și a timpului.

Modalitățile de preluare a datelor din fișierul de intrare, de inițializare a frontierelor domeniului și a matricei cu valoarea fluxului corespunzător sursei sunt identice cu cele prezentate pentru cazul staționar. Diferența față de regimul staționar constă în necesitatea păstrării valorilor distribuției temperaturii la intervalul de timp $\tau - \Delta\tau$ pentru a putea calcula distribuția la momentul curent de timp. Acest lu-

cru se realizează prin utilizarea unei a doua matrice, inițializată cu valorile de pe frontiera domeniului, care va pastra valorile de la iterația anterioară. Distribuția temperaturii se calculează direct repetând algoritmul până se atinge numărul de iterații corespunzător timpului fizic care se dorește a fi simulat. Numărul de iterații se calculează după formula $MAX_ITER = MAX_TIME/TIME_SLICE$. Există deasemenea trei implementări ale algoritmului: implementarea serială pentru sisteme de tip mono-procesor; implementarea folosind API-ul OpenMP pentru sisteme mono-procesor cu mai multe core-uri; implementarea folosind API-ul MPI pentru sisteme distribuite cu noduri de procesare mono sau multi-procesor (pentru un câștig suplimentar în viteza de procesare aplicația a fost implementată folosind cele două API-uri împreună).

```
double s_mdf(double **uc, double **ua, double **q, double h, double w, int nx, int ny)
{
1   ....
2   time_t start = time(NULL);
3   while (count <= MAX_ITER)
4   {
5       for(i=1;i<ny-1;i++)
6           for(j=1;j<nx-1;j++)
7               uc[i][j] = (1 - 4*h*w*t)*ua[i][j] + h*w*t*(ua[i-1][j] +
8                   + ua[i][j-1] + ua[i+1][j] + ua[i][j+1]) + w*t*q[i][j];
9       xchg = ua;
10      ua = uc;
11      uc = xchg;
12      if(count%(scenario[scn_index].step) == 0)
13          export_to_vtk(uc,nx,ny,scn_index,count);
14      count++;
15  }
16  ....
17  return difftime(stop,start);
}
```

Pentru implementarea paralela folosind OpenMPI soluția numerică (denumirea funcției va deveni *p_mdf*) este similară cu cea serială, adăugându-se pentru execuția paralelă înaintea instrucțiunilor de tip "for" instrucțiuni de forma:

```
#pragma omp parallel for private(i,j)
```

Pentru implementarea distribuită soluția numerică a algoritmului (*m_mdf*) este identică cu varianta utilizând OpenMP diferența fiind dimensiunea matricei pentru care acesta se execută. Rank-ul 0 va face o împărțire a domeniului inițial pe care

algoritmul se execută, trimițând celorlalte rankuri subdomenii (grupuri de linii) pe care să le prelucreze. Similar cu soluția de simulare în regim staționar, pentru calculul distribuit s-a păstrat modalitatea de simulare folosind scenariii ce permite transmiterea parametrilor simulării către toate nodurile de calcul. A fost creat tip de date nou *MPI_SCENARIO* corespunzător structurii anterioare de tip *scenario_t* care poate fi trimis prin MPI.

Rank-ul 0 citește datele de intrare din fișier, inițializează cele două matrice corespunzătoare domeniului și cea corespunzătoare valorilor fluxului sursei și trimite fragmentele corespunzătoare din acestea, împreună cu structura scenariu celorlalte rankuri. Deoarece schema de discretizare presupune ca valoarea temperaturii într-un punct din domeniu să fie calculată ca medie a valorilor anterioare ale temperaturilor vecinilor de la coordonatele $(i+1,j)$ $(i-1,j)$ $(i,j-1)$ $(i,j+1)$, cât și valorii anterioare a temperaturii nodului curent (i,j) se pune problema consistenței datelor la marginile subdomeniilor atribuite fiecărui rank. Pentru a soluționa această problemă fiecare nod computațional va avea alocat un domeniu cu două linii mai mare decât cel pe care trebuie să îl proceseze. Pe prima linie (0) va primi de la nodul care procesează subdomeniul adiacent din partea superioară, ultima linie din domeniul acestuia, iar pe ultima linie (n_y+1) va primi de la nodul care procesează subdomeniul adiacent din partea inferioară prima linie din domeniul acestuia. La rândul său fiecare nod de procesare va trimite nodurilor ce procesează subdomeniile adiacente prima (1) respectiv ultima linie (n_y) din domeniul său.

Asigurarea consistenței datelor este asigurată de funcțiile blocante și de modalitatea de intercalare a trimiterii și recepționării valorilor la marginile subdomeniului similară celei prezentate pentru cazul staționar. Deoarece de interes în cazul nestaționar este evoluția distribuției temperaturii, este necesar ca la iterațiile cu număr multiplu de [STEP] toate rankurile să trimită rankului 0 configurațiile subdomeniului lor, pentru ca acesta să realizeze exporturi intermediare către VTK, pentru a putea obține o simulare continuă a evoluției

Capitolul 4

Rulare și testare

4.1 Arhitectura sistemului de testare

Pentru testarea aplicației clusterul folosit avea următoarea infrastructură: 51 noduri Pentium 4 HyperThreading, 32 noduri Dual Xeon, 32 noduri Dual Quad-Core Xeon, 4 noduri Dual PowerXCell 8i, rezultând astfel 443 core-uri fizice de procesare. Capacitatea totală de stocare existentă este de 36TB

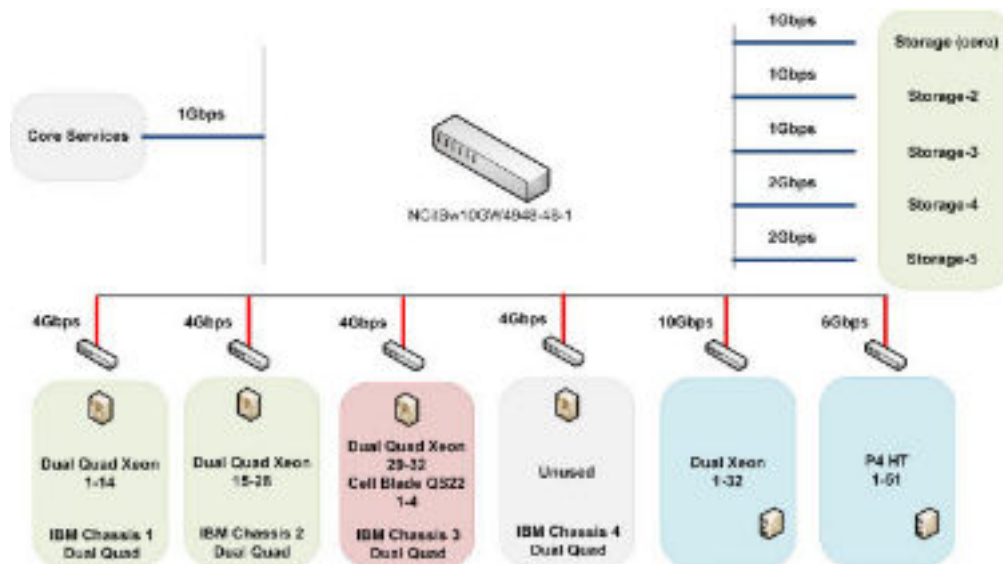


Figura 4.1: Infrastructura

Nodurile de procesare pe care a rulat efectiv aplicația curentă au fost 4 Quad Core E5420 – Penryn, Putere: 80W, Frecvență: 2.0GHz, FSB: 1333MHz, L2 Shared

Cache: 12MB, Main Memory: 16GB RAM / nod

4.2 Rezultate obținute

4.2.1 Implementare staționară

Pentru o mai buna observare a modului în care funcția ce determină condițiile la marginile domeniului și dimensiunea și fluxul sursei de caldură influențează distribuția de temperatură în regim staționar, mai jos sunt prezentate două scenarii de rulare și rezultatele vizuale corespunzătoare obținute în urma simulării.

```
[NUM_SCENARIO] = 2
```

```
[SCENARIO] = 0  
[OMP_THREADS] = 2  
[SIZE_X] = 200  
[SIZE_Y] = 200  
[BETA] = 1.65  
[H] = 1  
[EPS] = 0.0000001  
[MAX_ITER] = 1000000  
[ALG] = M_GAUSS_SEIDEL_RELAX  
[SOURCE_X] = 0.2  
[SOURCE_Y] = 0.2  
[SOURCE_RADIUS] = 4  
[SOURCE_TEMP] = 1.2  
[FUNC] = 5 * x ^ 2 - 5 * y ^ 2 + 10 * y
```

```
[SCENARIO] = 1  
[OMP_THREADS]  
[SIZE_X] = 200  
[SIZE_Y] = 200  
[BETA] = 1.65  
[H] = 1  
[EPS] = 0.0000001  
[MAX_ITER] = 1000000  
[ALG] = M_GAUSS_SEIDEL_RELAX  
[SOURCE_X] = 0.3  
[SOURCE_Y] = 0.4  
[SOURCE_RADIUS] = 4  
[SOURCE_TEMP] = 0  
[FUNC] = 5 * y ^ 2 - 5 * x ^ 2 + 10 * x
```

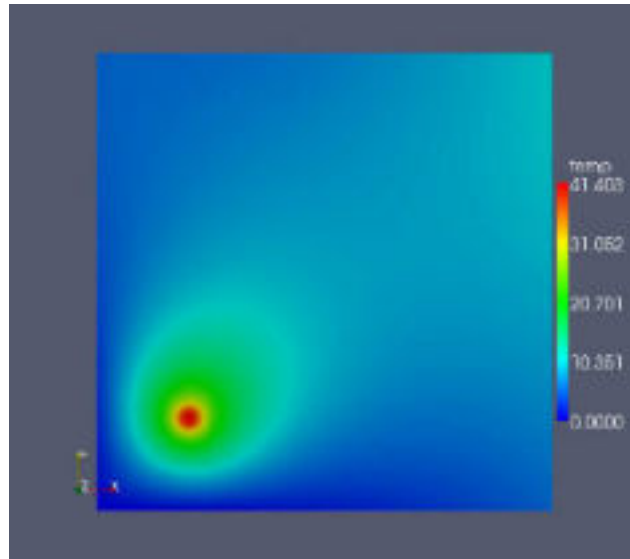


Figura 4.2: Simulare regim staționar cu sursă de caldură

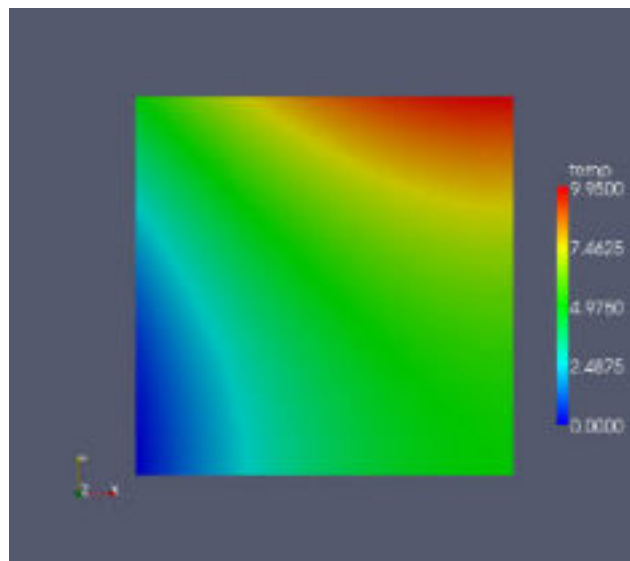


Figura 4.3: Simulare regim staționar fără sursă de caldură

4.2.2 Implementarea nestaționară

În cazul simulării regimului nestaționar de interes este studiul evoluției distribuției de temperatură în cadrul domeniului analizat. Pentru același scenariu prezentat mai sus, se obține următoarea variație în timp:

```
[SCENARIO] = 0  
[OMP_THREADS] = 2  
[SIZEX] = 200  
[SIZEY] = 200  
[H] = 1  
[W] = 250  
[MAX_TIME] = 3  
[TIME_SLICE] = 0.001  
[STEP] = 70  
[SOURCEX] = 0.2  
[SOURCEY] = 0.2  
[SOURCERADIUS] = 4  
[SOURCETEMP] = 1.2  
[FUNC] = 5 * x ^ 2 - 5 * y ^ 2 + 10 * x
```

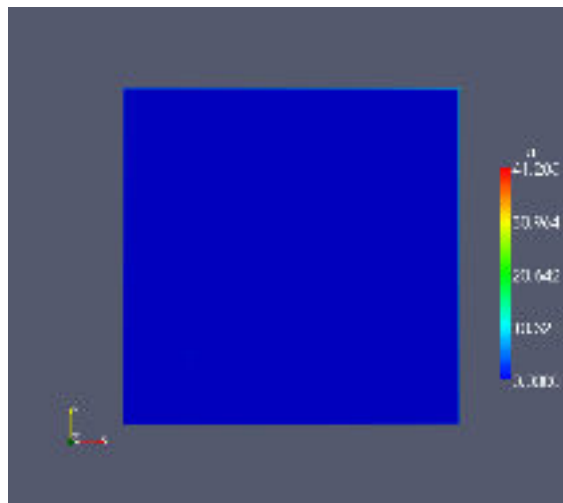


Figura 4.4: Simulare regim nestaționar cu sursă de căldură

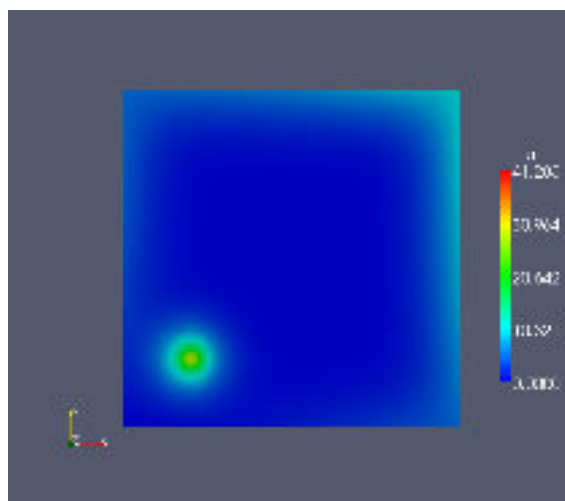


Figura 4.5: Simulare regim nestaționar cu sursă de căldură

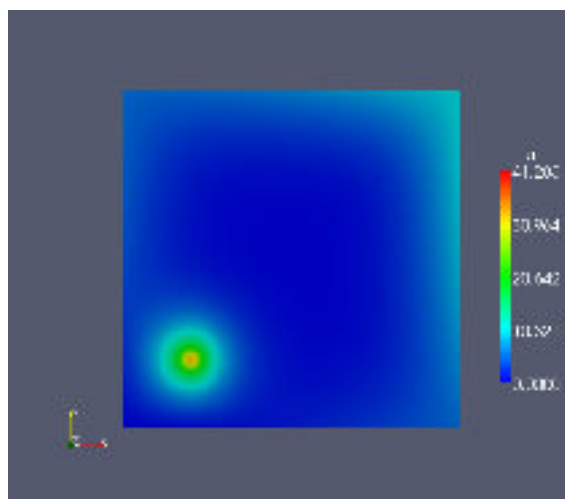


Figura 4.6: Simulare regim nestaționar cu sursă de căldură

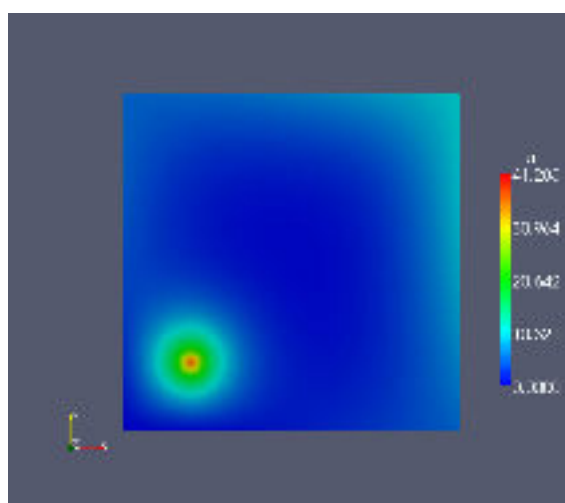


Figura 4.7: Simulare regim nestaționar cu sursă de căldură

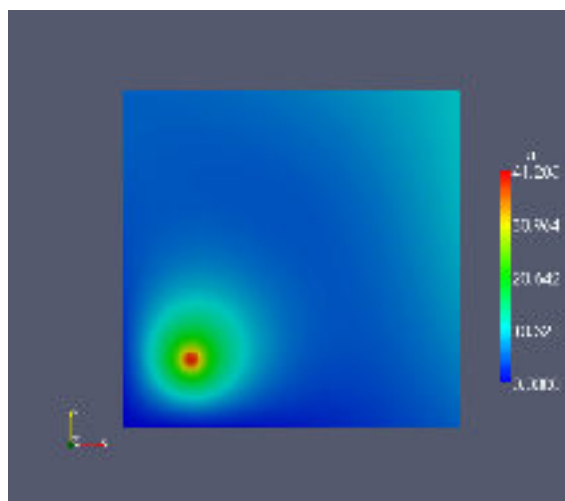


Figura 4.8: Simulare regim nestaționar cu sursă de căldură

Capitolul 5

Performanțe

5.1 Regim staționar

Algoritmii utilizați în cadrul acestei aplicații sunt compute-intensive, aproximativ 95% din timpul de rulare este dedicat calculului efectiv, așa cum se poate observa din analiza făcută cu Sun Studio Analyzer.





Functions	Callers-Callees	Source	Disassembly	Timeline	Experiments
 User CPU (sec.)	 User CPU (sec.)	 User CPU (sec.)	Name		
21.534	1.188	21.534	<libc-2.5.so>		
 0.	0.	21.534	main		
21.149	19.917	21.149	s_gauss_seidel_relax		
0.363	0.	0.363	export_to_vtk		
0.022	0.011	0.022	init		

Figura 5.1: Regim staționar.Sun Studio Analyzer: Funcții apelate de de main

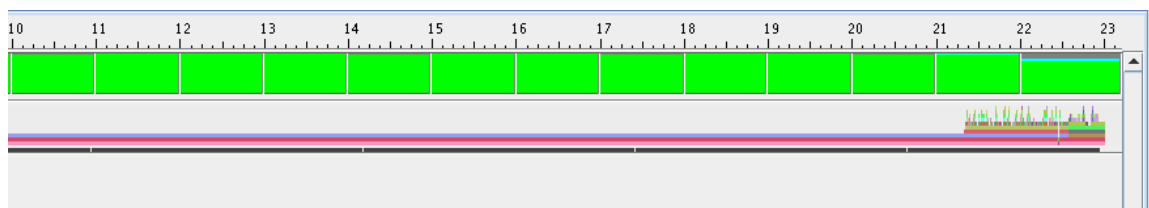


Figura 5.2: Regim staționar. Sun Studio Analyzer: Timeline

Acest capitol își propune să analizeze timpii de rulare necesari diferiților algoritmi numerici utilizați în cadrul aplicației, pentru diferite valori ale domeniului fizic considerat. Statisticile sunt create cu ajutorul gnuplot pentru fiecare algoritm numeric separat, cât și pentru același algoritm utilizând diferite metode de optimizare (paralelizare, calcul distribuit). Vom analiza prima dată timpii de rulare ai fiecărei implementări pornind de la versiunea serială, până la cea paralelizată și distribuită. Apoi se va prezenta o comparație a timpilor de rulare necesari aceluiași tip de algoritm, pe același domeniu simulat, dar implementat folosind cele două API-uri OpenMp și MPI.

Se observă astfel că, în cazul implementării seriale, se obține un speed-up mediu de 2.2x folosind metoda Gauss-Seidel cu suprarelaxare față de utilizarea metodei Jacobi, sau Gauss-Seidel, care pentru scenariile rulate, la dimensiuni mari ale domeniului, au obținut performanțe comparabile.

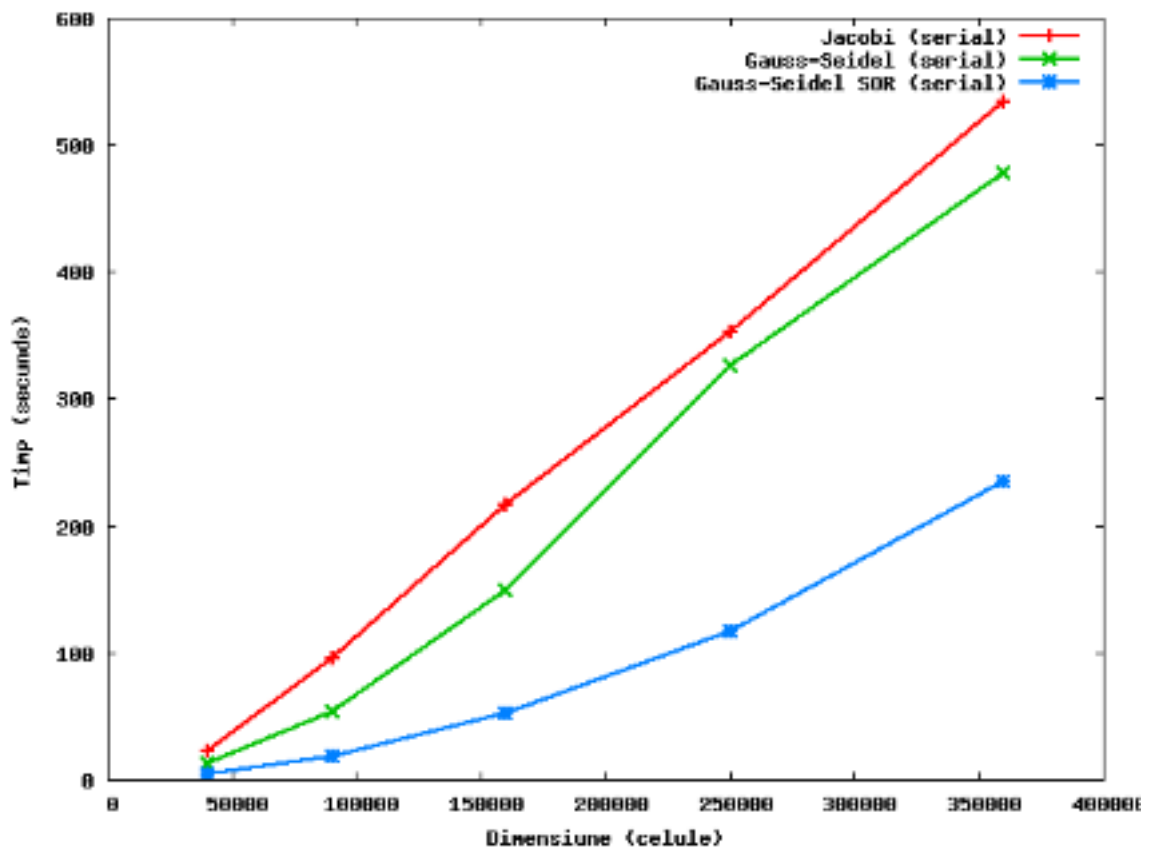


Figura 5.3: Regim staționar. Comparație între timpii de rulare în varianta serială

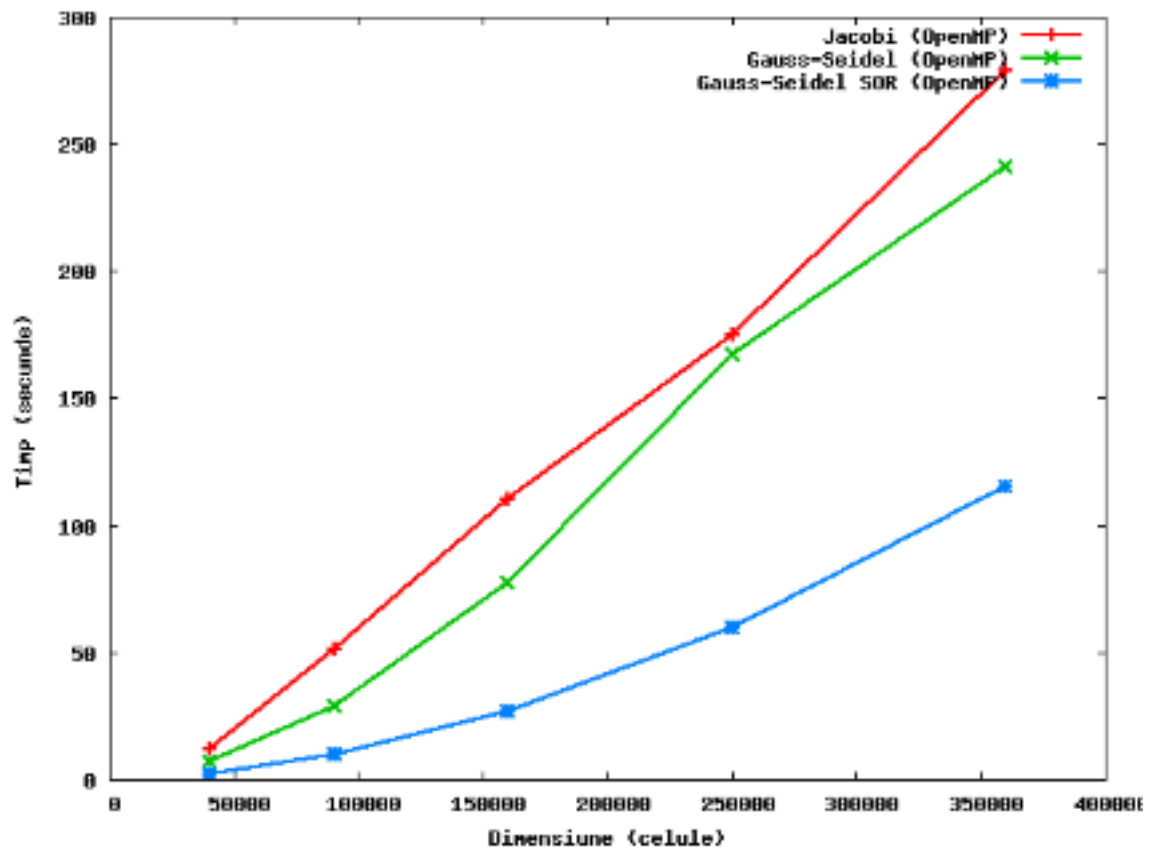


Figura 5.4: Regim staționar.Comparație între timpii de rulare în varianta paralelizată

Se obține, în cazul implementării folosind API-ul OpenMP (2 threaduri), un speed-up față de cazul staționar de 1.9x. De asemenea între implementarea folosind Jacobi și cea folosind Gauss-Seidel cu suprarelaxare, pentru implementarea paralelă se obține un speed-up de 2,42x.

În cazul implementării hibride (8 core-uri : 4 noduri MPI * 2 threaduri fiecare), față de cea serială, se obține un speed-up de 4.2x în cazul algoritmului Jacobi, de 4,5 în cazul algoritmului gauss-Seidel și un speed-up de 4.5x în cazul algoritmului Gauss-Seidel cu suprarelaxare. Folosind atât API-ul OpenMP cât și API-ul MPI se observă o scădere a timpului de aproximativ 12 ori în cazul utilizării algoritmului Gauss-Seidel SOR față de utilizarea algoritmului Jacobi. La fel ca și în cazul implementării seriale performanțele algoritmilor Jacobi și Gauss-Seidel sunt asemănătoare.

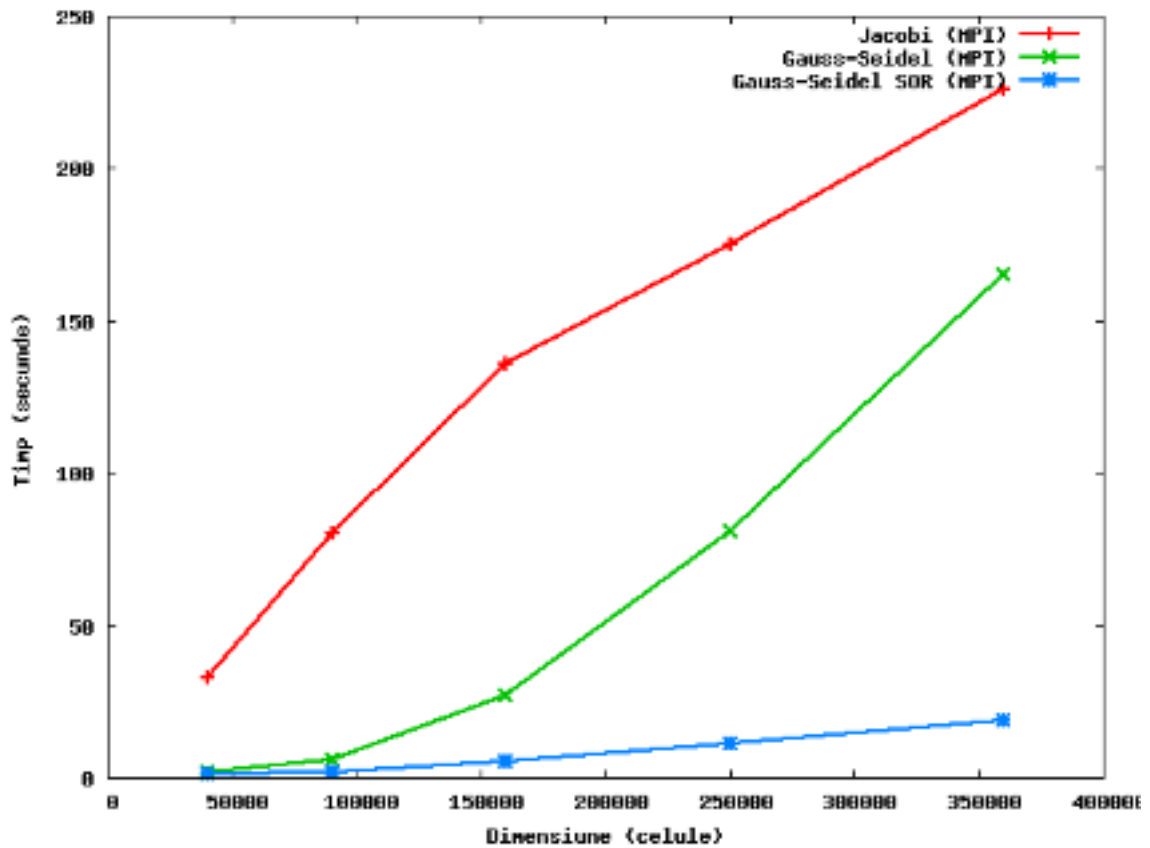


Figura 5.5: Regim staționar. Comparație între timpii de rulare în varianta distribuită

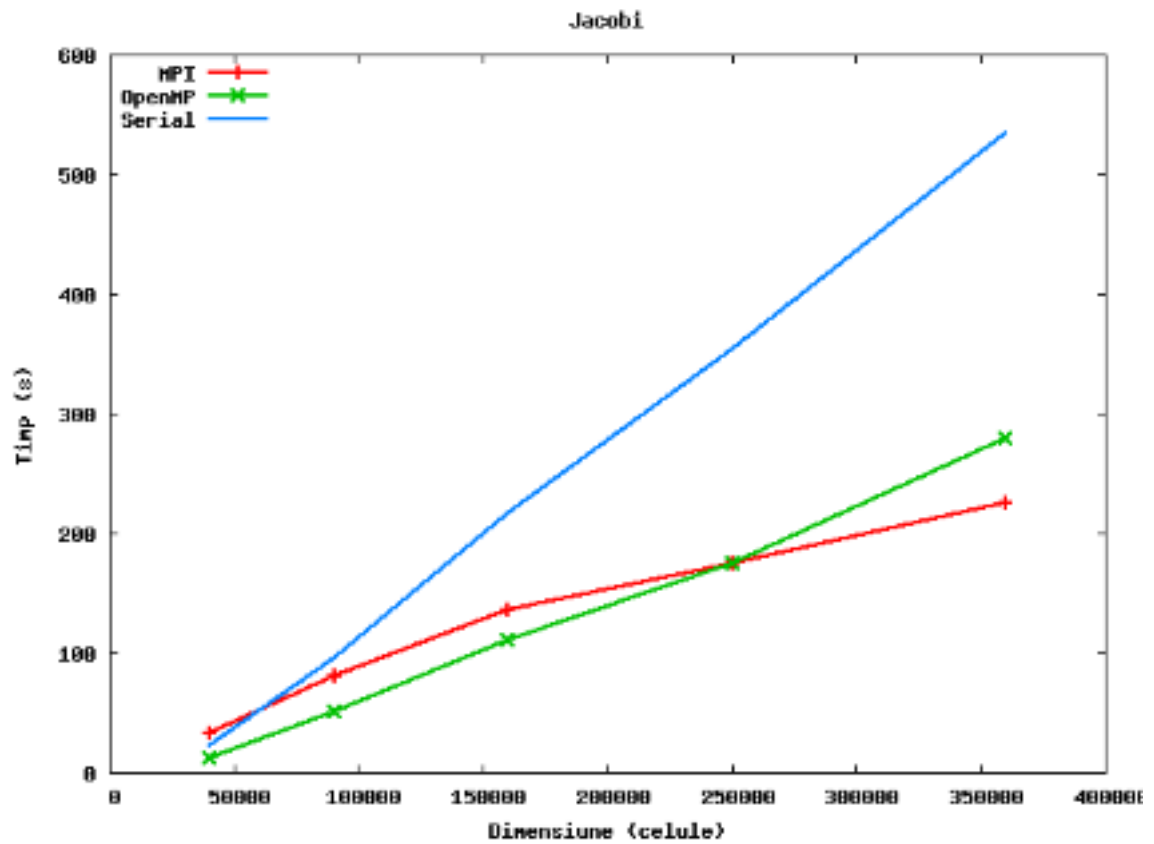


Figura 5.6: Regim staționar. Comparație între timpii de rulare ai algoritmului Jacobi

Se observă, în cazul algoritmului Jacobi, un speed-up de 1.9x în cazul implementării folosind OpenMp față de cea serială, și un speed-up suplimentar prin implementarea cu MPI de 1.2x. Rezultă astfel un speed-up total în cazul implementării cu OpenMp și MPI față de aplicația serială de 2.36x

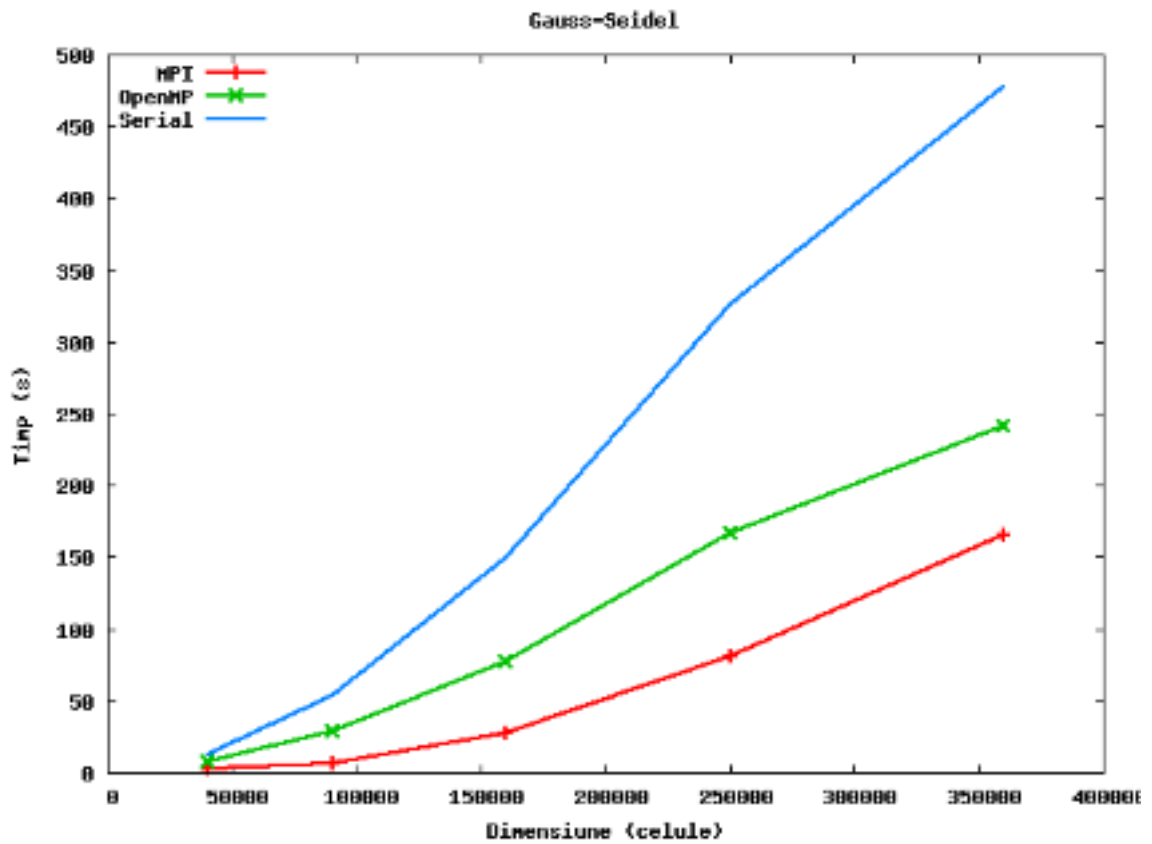


Figura 5.7: Regim staționar. Comparație între timpii de rulare ai algoritmului Gauss-Seidel

Se observă, în cazul algoritmului Gauss-Seidel, un speed-up de 1.9x în cazul implementării folosind OpenMp față de implementarea serială, și un speed-up suplimentar prin implementarea cu MPI de 1.4x. Rezultă astfel un speed-up total în cazul implementării cu OpenMp și MPI față de aplicația serială de 2.8x

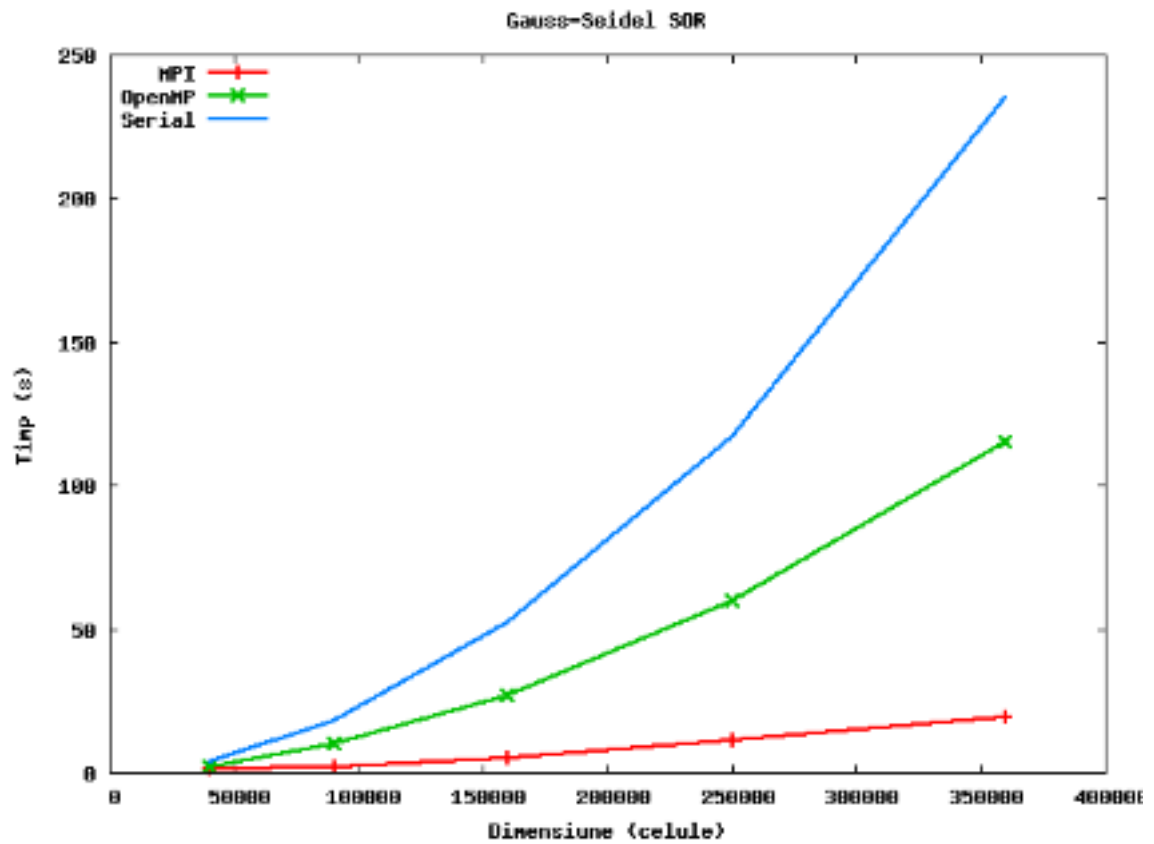
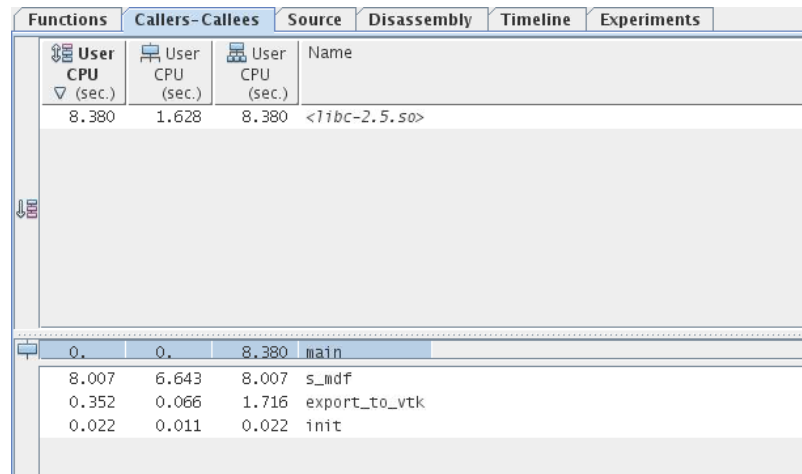


Figura 5.8: Regim staționar. Comparație între timpii de rulare ai algoritmului Gauss-Seidel SOR

Se observă, în cazul algoritmului Gauss-Seidel cu suprarelaxare, un speed-up de 1.9x în cazul implementării folosind OpenMp față de cea serială, și un speed-up suplimentar prin implementarea cu MPI de 4x. Rezultă astfel un speed-up total în cazul aplicației cu OpenMp și MPI față de implementarea serială de 7.6x

5.2 Regim nestaționar

Algoritmii utilizați în cadrul acestei aplicații sunt compute-intensive, aproximativ 75% din timpul de rulare este dedicat calculelor efective, așa cum se poate observa din analiza făcută cu Sun Studio Analyzer.



Functions	Callers-Callees	Source	Disassembly	Timeline	Experiments
User CPU (sec.)	User CPU (sec.)	User CPU (sec.)	Name		
8.380	1.628	8.380	<libc-2.5.so>		
0.	0.	8.380	main		
8.007	6.643	8.007	s_mdff		
0.352	0.066	1.716	export_to_vtk		
0.022	0.011	0.022	init		

Figura 5.9: Regim nestaționar.Sun Studio Analyzer: Funcții apelate de main

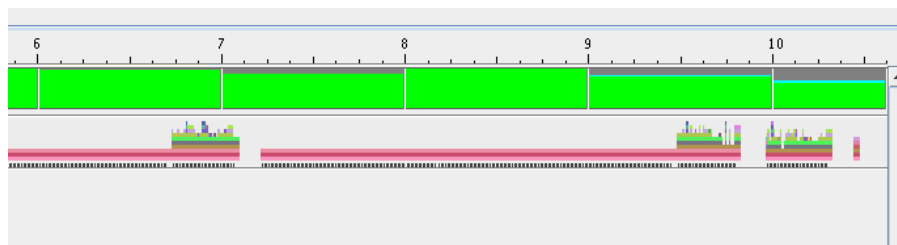


Figura 5.10: Regim nestaționar. Sun Studio Analyzer: Timeline

Acest capitol își propune să analizeze timpii de rulare necesari algoritmului numeric utilizat în cadrul aplicației, pornind de la versiunea serială și comparând-o cu cea paralelizată și cea distribuită.

În cazul regimului nestaționar se observă un speed-up de 3,2x față de timpul obținut în cazul implementării seriale în cazul paralelizării folosind API-ul OpenMP (2 threaduri) și un speed-up de 3,7x în cazul utilizării implementării hibride utilizând atât API-ul OpenMp cat si API-ul MPI (8 core-uri: 4 noduri MPI * 2 threaduri OpenMP).

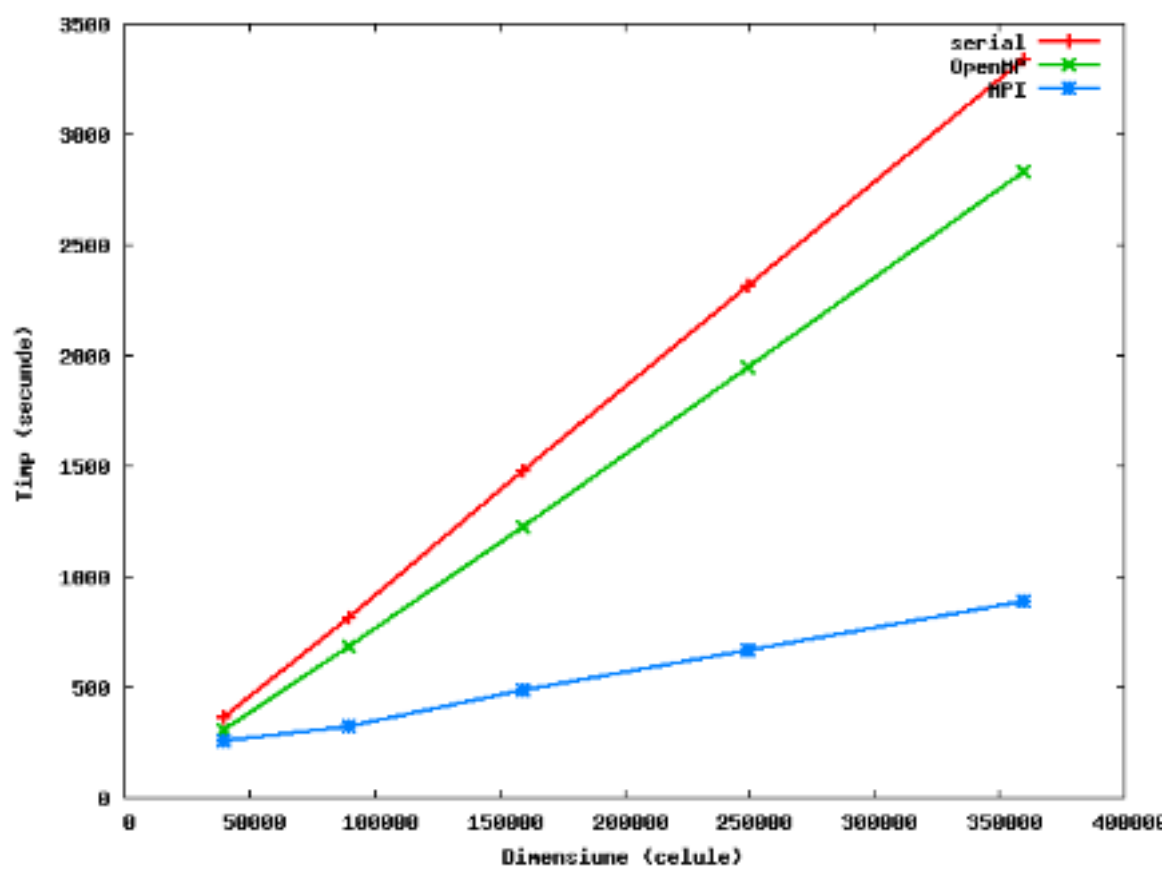


Figura 5.11: Regim nestationar.Comparație între timpii de rulare

Capitolul 6

Concluzii

Deși schemele de discretizare și metodele numerice utilizate în cadrul acestei aplicații, pentru a simula numeric transferul de căldură nu sunt cele mai performante, simplitatea și potențialul mare de paralelizare pe care îl au, au condus la obținerea unor rezultate satisfăcătoare din punct de vedere al timpului necesar simulării și al efortului computațional implicat. În plus, diferite tehnici de optimizare a codului (scoaterea cât mai multor calcule în afara buclelor, "loop-merging") combinate cu optimizarea la compilare -O3/-xO3 îmbunătățesc considerabil performanțele aplicației. Așa cum se poate observa în profiling soluția implementată este una compute-intensive,

Având în vedere faptul că scopul final al lucrării a fost dezvoltarea unei aplicații care să poată rula peste o rețea de noduri de procesare interconectate (cluster), alegerea unei implementări hibride folosind atât API-ul OpenMP cât și API-ul MPI s-a dovedit a fi deosebit de inspirată, producând o creștere semnificativă a performanțelor aplicației, reducând timpul necesar procesului computațional, și în același timp oferind posibilitatea realizării de simulări pentru domenii mari (aprox. 1.000.000 puncte). Pentru cel mai eficient algoritm numeric utilizat pentru simularea în regim staționar, în speță Gauss-Seidel cu suprarelaxare, s-a obținut un speed-up de 7.6x la o rulare pe 8 core-uri (4 noduri MPI * 2 threaduri) față de implementarea serială. În cazul nestaționar s-a obținut un speed-up de 3.7x la o rulare folosind OpenMP

și MPI pe 8 core-uri (4 noduri MPI * 2 threaduri) față de rularea serială.

Ca implementări viitoare se poate lua în considerare o schemă de discretizare folosind elemente finite sau volume finite, acestea din urmă fiind tendința actuală în domeniu. Se poate deasemena considera un domeniu 3D pentru calculul distribuției temperaturii. În scopul realizării extinderii la 3D se poate deasemenea integra aplicația cu un pachet software de partiționare a domeniului, cum este ParMetis. Datele furnizate de acesta din urmă pot fi folosite ca fișiere de intrare pentru simulator.

Bibliografie

- [1] Ioan Dumitrescu – *Simularea Cîmpurilor potențiale*, 1983, Editura Academiei RS Romania București
- [2] Marin Bică, Mădălina Călbureanu, Corina Cernăianu, Gabriela Demian, *Transfer de căldură*, 2003, Editura ICMET Craiova
- [3] Aureliu Leca, Emilia-Cerna Mladin, Mihaela Stan, *Transfer de căldură și masă - o abordare inginerescă*, 1998, Editura Tehnică București
- [4] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, 1998, Springer-Verlag
- [5] Quinn, M. J. , *Parallel Programming in C with MPI and OpenMP*, 2003 , McGraw Hill Higher Education
- [6] Gropp, W., Lusk, E. Skjellum, A., *Using MPI, Portable Parallel Programming with the Message-Passing Interface*, 1998, Cambridge, MA MIT PRESS, 2nd edition
- [7] Smith, L. and Bull, M., *Development of mixed mode MPI / OpenMP applications*, 2001, Sci. Program. 9, 2,3 (Aug. 2001), 83-98
- [8] Smith, L. and Bull, M. , *Development of mixed mode MPI / OpenMP applications*, 2001, Sci. Program. 9, 2,3 (Aug. 2001), 83-98
- [9] <http://www.open-mpi.org>
- [10] <http://www.paraview.org>

- [11] http://developers.sun.com/sunstudio/overview/topics/analyzer_index.html
- [12] <http://cluster.grid.pub.ro/index.php/ncit-cluster/38-profiling/81-profiling-with-sun-studio-analyzer>

Cuprins

1	Introducere	2
2	Aspecte teoretice	5
2.1	Generalități	5
2.2	Conducția termică	6
2.3	Transmiterea căldurii prin conducție în regim staționar	11
2.3.1	Abordare analitică	11
2.3.2	Metode numerice de rezolvare a conducției termice staționare	12
2.4	Transmiterea căldurii prin conducție în regim nestaționar multidimensional	17
2.4.1	Abordare analitică	17
2.4.2	Metode numerice de rezolvare a conducției termice tranzitorii	20
3	Detalii de implementare	25
3.1	Regim staționar	26
3.2	Regim nestaționar	36
4	Rulare și testare	40
4.1	Arhitectura sistemului de testare	40
4.2	Rezultate obținute	41
4.2.1	Implementare staționară	41
4.2.2	Implementarea nestaționară	43

5	Performanțe	46
5.1	Regim staționar	46
5.2	Regim nestaționar	53
6	Concluzii	55