UNIVERZA V LJUBLJANI

Fakulteta za strojništvo

Izračun porazdelitve temperature v 2D prerezu

Projektna naloga pri predmetu Napredna računalniška orodja

Matija Debeljak, Jure Križman

Ljubljana, januar 2024

Kazalo

1 Uvod

Za projektno nalogo smo obravnavali primer časovno neodvisnega prenosa toplote v 2D prerezu. Podani so nam bili robni pogoji glede na katere smo morali preko programa v jeziku C++ izračunati porazdelitev temperature. Predpostavili smo temperaturno neodvisno prevodnost toplote in robne pogoje. Izhajali smo iz sledeče diferencialne enačbe [1].

$$\frac{\partial}{\partial x}(k\frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(k\frac{\partial T}{\partial y}) + q = 0$$

Za reševanje smo morali izpolniti robne pogoje. Te so vključevali:

- Temperaturo T[°C]
- Toplotni tok [W/m^2]
- Prestop toplote s toplotno prevodnostjo h [W/m^2K] in temperaturo okoliške tekočine T_{ext} [°C]

Ob upoštevanju konstante toplotne prevodnosti in pomanjkanju notranje generacije toplote, se izhodiščna diferencialna enačba poenostavi.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Podano smo imeli .txt datoteko v kateri so bili zapisani podatki o mreži našega nosilca. Napisati smo morali program v jeziku C++, ki podano datoteko prebere in glede na podatke preko metode končnik razlik sestavi sistem enačb za določanje ustaljene temperature po vozliščih nato pa ga reši z metodo Gauss-Seidel. Podatke smo nato vizualizirali s pomočjo programa Paraview. Na koncu pa smo dobljeni sistem enačb rešili s pomočjo programa Wolfram Mathematica in primerjali čas reševanja z našo kodo.

2 Zbiranje podatkov o geometriji in robnih pogojih

2.1 Struktura dane mreže

Podana nam je bila mreža, v obliki .txt datoteke, ki jo moramo prebrati zato, da iz nje dobimo ustrezne podatke. Struktura mreže je podana v navodilih, zato je ne bomo znova prikazali v projektni nalogi.

2.2 Branje vhodne datoteke

2.2.1 Uporaba knjižnice 'fstream'

Za naš primer, smo vzeli knjižnico 'fstream' za branje podatkov v datotekah. V spodnji prikazani vrstici nam knjižnica fstrema ustvari objekt 'indata' tipa 'std::ifstream'. Objekt 'indata' nam odpre datoteko '(filename)'.

```
#include <fstream>

std::ifstream indata;
indata.open(filename);
```

2.2.2 Preberemo datoteko

Sedaj, ko smo odprli datoteko, se lahko lotimo branja datoteke. To naredimo z ukazom 'getline' in shranimo to vrednost v niz 'vrstica'.

```
std::string vrstica;
std::getline(indata, vrstica);
```

2.2.3 Uporaba stringov za branje datoteke

V navodilih je bila dana struktura datoteke, v C++ moramo to strukturo definirati, to naredimo z uporabo ukaza stringstream. Ta ukaz nam prebere vrstico, mi pa mu moramo povedati kakšen tip podatka se nam nahaja v vrstici.

```
int zaporedna;
std::string text;
double a, b, c, d;
char semicolon, colon;

std::stringstream tt(vrstica);
tt >> text >> a;
```

2.2.4 Shranjevanje prebranih podatkov v vektorje

Že na začetku smo definirali vektorje katere uporabljamo v kodi. Ko imamo neke podatke, jih zapišemo v ta vektor z ukazom 'pushback'. V našem primeru jih shranimo v vektor z imenom 'velikost'. V spodnji kodi so prikazani še vsi ostali vektorji , katere uporabimo za shranjevanje podatkov iz vhodne podatkovne mreže.

```
std::vector<std::vector<double>> tocke;
std::vector<std::vector<double>> celice;
std::vector<double> T;
std::vector<std::string> vrsteRP;
std::string vrsta_RP;
std::vector<double> vrednostiRP;
std::vector<std::vector<int>> tockeRP;
std::vector<int> velikosti;
std::vector<int> vrednosti_prestopa;

velikosti.push_back(a);
```

Ta postopek ponovimo za vsako vrstico datoteke, pri čemer preberemo različne podatke glede na potrebe programa (točke, celice, robni pogoji).

3 Ustvarjanje matrike A in vektorja b

3.1 Določanje sosedov vozlišč

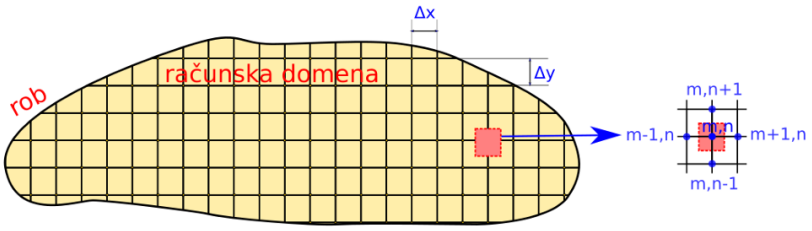
Sedaj, ko smo ustvarili sezname robnih pogojev, celic in koordinat vozlišč, lahko pričnemo z ustvarjanjem matrike A in vektorja b, s katerima bomo izračunali vrednosti temperature po vozliščih.

Napisali smo void funkcijo ŠestaviMatrikeš katero oblikujemo matriko A in vektor b. Koda najprej inicializira 2 dimenzionalen vektor A in vektor b ter ju napolni z vrednostmi 0. Njuna velikost je pogojena s številom prostostnih stopenj, ki so shranjene v vektorju "velikosti"na mestu [0]. Nato iteriramo po vseh naših vozliščih in preverimo kateri so njihovi sosedi, te se lahko nahajajo levo, desno, zgoraj ali spodaj glede na naše vozlišče. Kje se nahajajo določimo z odštevanjem x in y koordinat našega vozlišča in sosednjega. Za vsako sosednje vozlišče shranimo, kje se nahaja glede na naše vozlišče, nato pa ustvarimo seznam sosednjih vozlišč za vsako vozlišče "`sosednja_vozlisca`". Če ima vozlišče manj kot 4 sosede, je na mestu manjkajočih sosedov njihova številka označena z -1.

3.2 Določanje vrednosti A in b

3.2.1 Vozlišče v notranjosti

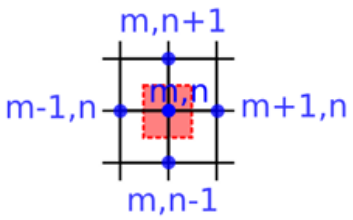
Sedaj ko imamo seznam sosedov za vsako vozlišče lahko glede na sosede določimo, če se vozlišče nahaja na robu. Če nobeden izmed sosedov ni označen s številko -1 vemo, da vozlišče ni na robu, torej ima vse 4 sosede. Zanj lahko uporabimo enačbo (12) iz predloge. Enačbe, iz katerih bomo določili vrednosti



Slika 1: Slika mreže vozlišč in vozlišča s svojimi sosedi, [1]

matrike A in vektorja b, izhajajo iz metode končnih razlik, ki temelji na transformiranju diferencialnih operatorjev v diferenčne, to pa stori z diferenčnimi shemami.

$$T_{m-1,n} + T_{m+1,n} + T_{m,n-1} + T_{m,n+1} - 4T_{m,n} = 0$$



Slika 2: Vozlišče v notranjosti, [1]

Našemu vozlišču v matriki A prištejemo vrednost -4, vsem ostalim sosedom pa vrednost 1. Vrednost vektorja b za vozlišče [m,n] pa je 0.

3.2.2 Vozlišče na robu

Če pa je kateri izmed sosedov vozlišča označen z -1 vemo, da se nahajamo na robu, določiti moramo še kateri robni pogoji veljajo na tem robu. Iteriramo po seznamu "tockeRP", ki vsebuje številke vozlišč za vsakega izmed robnih pogojev. Tako določimo kateri robni pogoj velja za naše vozlišče.

- Temperatura T[°C]

  Če je robni pogoj temperatura shranimo v diagonalo matrike A na mestu našega vozlišča vrednost 1, v vektor b pa vrednost temperature robnega pogoja.

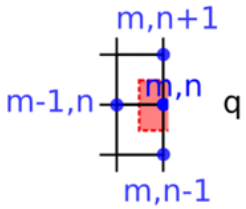
- Toplotni tok [W/m^2]

  Če je robni pogoj toplotni tok imamo več opcij:

  - Vrednost levega soseda je -1, naše vozlišče je na levem robu Diagonali matrike A prištejemo vrednost -4, desnemu sosedu vozlišča prištejemo vrednost 2, spodnjemu in zgornjemu sosedu pa 1. Vektor b dobi vrednost $-2 \cdot \frac{q \cdot \Delta x}{k}$, kjer je q vrednost toplotnega toka za naš robni pogoj, Δx je razdalja med vozlišči po osi x, k pa je toplotni prevodnostni koeficient našega materiala.
  - Vrednost desnega soseda je -1, naše vozlišče je na desnem robu Diagonali matrike A prištejemo vrednost -4, levemu sosedu vozlišča prištejemo vrednost 2, spodnjemu in zgornjemu sosedu pa 1. Vektor b dobi vrednost $-2 \cdot \frac{q \cdot \Delta x}{k}$.

– Vrednost spodnjega soseda je -1, naše vozlišče je na spodnjem robu Diagonali matrike A prištejemo vrednost -4, zgornjemu sosedu vozlišča prištejemo vrednost 2, levemu in desnemu sosedu pa 1. Vektor b dobi vrednost $-2 \cdot \frac{q \cdot \Delta x}{k}$.

– Vrednost zgornjega soseda je -1, naše vozlišče je na zgornjem robu Diagonali matrike A prištejemo vrednost -4, spodnjemu sosedu vozlišča prištejemo vrednost 2, levemu in desnemu sosedu pa 1. Vektor b dobi vrednost $-2 \cdot \frac{q \cdot \Delta x}{k}$.



Slika 3: Primer robnega pogoja toplotnega toka na desnem robu, [1]

- Prestop toplote s toplotno prevodnostjo h [W/m^2K] in temperaturo okoliške tekočine T_{ext} [°C] Spet moramo ločiti enačbe, glede na položaj našega vozlišča.

  – Vrednost levega soseda je -1, vozlišče se nahaja na levem robu. Diagonala matrike A dobi vrednost $-2 \cdot \frac{h \cdot \Delta x}{k+2}$. Desnemu sosedu vozlišča prištejemo vrednost 2, spodnjemu in zgornjemu pa vrednost 1. Vektor B dobi vrednost $\frac{-2h \cdot \Delta x \cdot T_{ext}}{k}$.

  – Vrednost desnega soseda je -1, vozlišče se nahaja na desnem robu. Diagonala matrike A dobi vrednost $-2 \cdot \frac{h \cdot \Delta x}{k+2}$. Levemu sosedu vozlišča prištejemo vrednost 2, spodnjemu in zgornjemu pa vrednost 1. Vektor B dobi vrednost $\frac{-2h \cdot \Delta x \cdot T_{ext}}{k}$.

  – Vrednost spodnjega soseda je -1, vozlišče se nahaja na spodnjem robu. Diagonala matrike A dobi vrednost $-2 \cdot \frac{h \cdot \Delta x}{k+2}$. Zgornjemu sosedu vozlišča prištejemo vrednost 2, levemu in desnemu pa vrednost 1. Vektor B dobi vrednost $\frac{-2h \cdot \Delta x \cdot T_{ext}}{k}$.

  – Vrednost zgornjega soseda je -1, vozlišče se nahaja na zgornjem robu. Diagonala matrike A dobi vrednost $-2 \cdot \frac{h \cdot \Delta x}{k+2}$. Spodnjemu sosedu vozlišča prištejemo vrednost 2, levemu in desnemu pa vrednost 1. Vektor B dobi vrednost $\frac{-2h \cdot \Delta x \cdot T_{ext}}{k}$.

4 Računanje vrednosti T vozlišč

Vrednosti vozlišč smo izračunali iz matrike A in vektorja b. Izhajali smo iz sledeče enačbe.

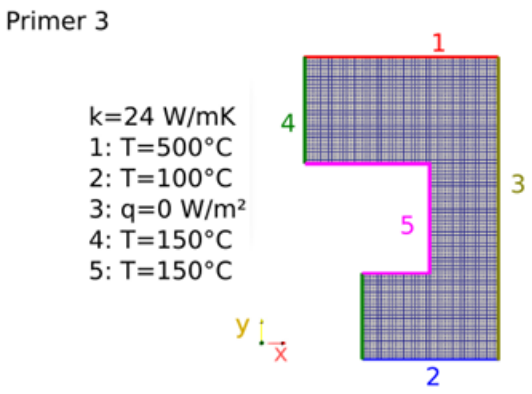
$$[A][T] = [b]$$

Ta matrična enačba predstavlja sistem enačb s toliko enačbami, kot je vozlišč v našem nosilcu. Ker je matrika A "redka", to pomeni, da je večina njenih elementov enakih 0, neničelni pa se nahajajo na in okoli diagonale, lahko uporabimo metodo Gauss-Seidel za reševanje sistema enačb. Gauss-Seidel metoda spada pod iterativne metode reševanja sistema enačb. Najprej določimo približek vektorja T, v našem primeru smo vsem vozliščem določili temperaturo 100°C.

Metoda iterira po vrsticah, temperaturo v n-tem vozlišču izračuna tako, da vzame n-to vrednost vektorja b[n] in od nje odšteje $A[n,i] \cdot T[i]$ za vse i tiste vrstice, razen i = n. Nato dobljeno vrednost T[n] deli z vrednostjo A[n]. Tako dobimo novo vrednost T[n], ki se nato upošteva pri računanju vrednosti T[n+1] in tako dalje. Ko izračunamo vrednosti vseh temperatur, postopek večkrat ponovimo in vsakič upoštevamo prejšnje izračunane vrednosti T, več kot naredimo iteracij, bližje smo eksaktni rešitvi.

Za reševanje sistema enačb smo v svoji kodi napisali void funkcijo "ResevanjeMatrik", ki za input vzame matriko A in vektor b, nato pa preko metode Gauss-Seidel s 1000 iteracijami izračuna vrednosti T po vozliščih in jih shrani v vektorju T.

5 Naš primer



Slika 4: Slika prereza našega nosilca s podatki o prevodnosti in robnih pogojih, [1]

6 Rezultati in vizualizacija

6.1 Pridobivanje rezultatov

Ko zaženemo kodo se najprej izvede funkcija "PreberiTocke", podati ji moramo lokacijo kjer imamo shranjeno datoteko s podatki za naš primer. Ta funkcija v različne vektorje in matrike shrani podatke o koordinatah vozlišč, celicah in njihovih vozliščih, ter robnih pogojih.

Po tem se izvede funkcija "SestaviMatrike", ki ustvari matriko A in vektor b.

Na koncu pa se izvede funkcija "ResevanjeMatrik", ki preko metode Gauss-Seidel reši sistem enačb podan v matriki A in vektorju b, ter nam vrne vektor temperatur po vozliščih T.

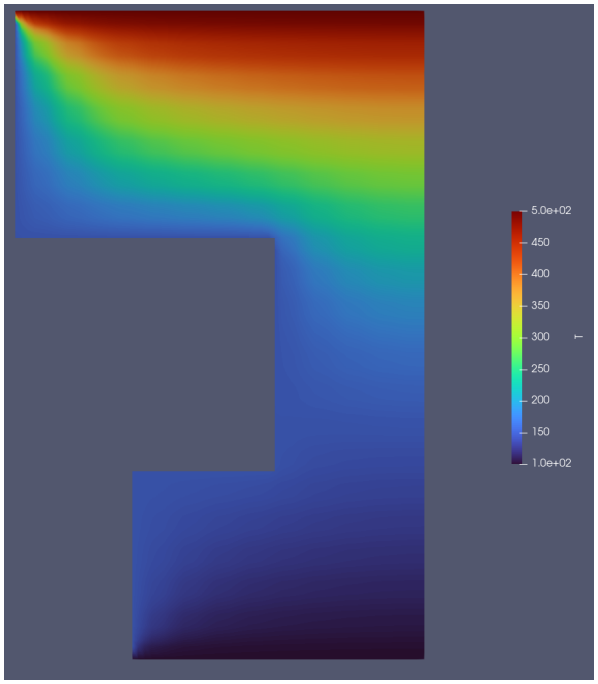
6.2 Prikaz rezultatov

6.2.1 Tekstovna datoteka

Za prikaz rezultatov program najprej izpiše vektor T v tekstovno datoteko "numbers.txt". Temperatura vsakega vozlišča je tu zapisana v svoji vrstici.

6.2.2 Vizualizacija

Za še boljši prikaz rezultatov smo te shranili v datoteki formata .vtk, ki jo lahko prikažemo s programsko opremo Paraview. V .vtk datoteki so zapisani podatki o temperaturah vozlišč in njihovih položajih za pravokotne celice. Datoteko, ki smo jo generirali z našo C++ kodo smo nato uvozili v Paraview in dobili sledeče rezultate.



Slika 5: Slika prereza našega nosilca z barvnim gradientom temperature

Temperatura na vrhu nosilca je najvišja - 500°C, na dnu pa je najnižja temperatura - 100°C. To se ujema z našimi robnimi pogoji. Na nosilcu vidimo lep toplotni gradient, kjer temperatura pada od vrha proti dnu nosilca.

6.2.3 Časi reševanja

Da bi dobili občutek, kako zahtevno je reševanje po metodi Gauss-Seidel, smo uporabili knjižnico chrono, da smo izmerili čas delovanja našega programa. Poleg tega smo uporabili še knjižnico OpenMP za paralelizacijo reševanja for zank. Pri računanju po metodi Gauss-Seidel smo uporabili parallel for.

```
#pragma omp parallel shared(A,B,T,n) private(d)
    {
        for (int iitt = 0; iitt < 1000; iitt++)
        {
#pragma omp for
            for (int jj = 0; jj < n; jj++) {
                d = B[jj];


                for (int ii = 0; ii < n; ii++) {
                    if (jj != ii) {
                        d = d - A[jj][ii] * T[ii];

                    }
                }
                T[jj] = d / A[jj][jj];
                if (A[jj][jj] == 0) {

                }
            }
        }
    }
```

Program smo zagnali na HPC-ju, da smo lahko dosegli večje število jeder.

Število niti	Čas reševanja [s]
1	55,8881
20	3,47518
50	2,29765
100	3,16766

Tabela 1: Tabela časov reševanja sistema enačb glede na število niti

Vidimo da, kot po pričakovanjih, povečanje števila niti pohitri čas reševanja. Čas reševanja je bil pri 20 nitih dosti hitrejši kot pri eni sami, pri 50 nitih pa se čas v primerjavi z 20 nitmi ni dosti izboljšal. S 100 nitmi se je čas celo podaljšal na 3,16766 sekund. Vidimo, da obstaja neka meja, do katere se še splača dodati več niti, nadaljnje izboljšave pa bodo minimalne. Do tega pojava verjetno pride, ker se delitev dela pri veliko jedrih zakomplicira in računalnik porabi veliko časa, da delo razdeli med jedri, nato pa sestavi posamezne rešitve v skupno

6.2.4 Reševanje z Wolfram Mathematico

Sistem enačb smo rešili še z Wolfram Mathematico, da bi imeli primerjavo časov reševanja in rezultatov med našo programsko opremo in komercialno opremo. Matriko A in vektor b, ki smo ju ustvarili z našo C++ kodo smo shranili v datoteko formata .json. To smo naredili s pomočjo knjižnice nlohmann/json (https://github.com/nlohmann/json). Koda te knjižnice je dokaj preprosta.

```
json jsonData;
jsonData["Array"] = mojA;
jsonData["Vector"] = mojB;

std::ofstream outputFile("output1.json");
outputFile << std::setw(4) << jsonData << std::endl;
outputFile.close();
std::cout << "JSON data written to output.json" << std::endl;
```

Koda nam v datoteki "output1.json" shrani podatke o matriki A pod imenom "Array", podatke o vektorju b pa pod imenom "Vector".

Datoteko "output1.json"nato uvozimo v Wolfram Mathematico. Podatke iz datoteke .json shranimo v matriko A in vektor b, nato pa rešimo sistem enačb s funkcijo Linearsolve[A,b]. Dobili smo, da je čas reševanja našega sistema enačb znašal 0.1075 sekund, kar je precej hitreje kot z našim C++ programom. Ko smo primerjali naše C++ rezultate s temi Wolfram Mathematice, smo videli, da so praktično identični. Wolfram Mathematica bi lahko bila hitrejša iz več razlogov; lahko da bolje izkoristi multithreading, ima bolj dodelane metode reševanja sistema enačb ali pa izvede manj iteracij.

7 Povzetek

Za projektno nalogo smo imeli podano tekstovno datoteko, ki je vsebovala podatke o mreži nosilca. Podane smo imeli x in y koordinate vozlišč in katere celice jih vsebujejo, poleg tega pa smo imeli podane še robne pogoje, ki so lahko predstavljali temperaturo roba, toplotni tok ali pa konvekcijo.

S pomočjo knjižnice fstream smo prebrali podatke in jih shranili v sezname. Te so vsebovali vrednosti števila prostostnih stopenj (torej števila vozlišč), koordinate vozlišč, celice in njihova vozlišča, vrednosti robnih pogojev in njihova vozlišča.

Nato smo z uporabo metode končnih razlik, ki diferencialne operatorje preko diferenčnih shem pretvori v diferenčne operatorje, ustvarili sistem enačb in ga shranili v obliki matrike A in vektorja b. Sistem enačb smo rešili z uporabo iterativne metode Gauss-Seidel, izvedli smo 1000 iteracij. Reševanje sistema enačb smo izvedli tudi s pomočjo paralelizacije in primerjali koliko se časi izboljšajo. Čas reševanja s samo eno nitjo je bil 55,8881 sekund, z 20 nitmi je bil 3,47518 sekund, s 50 nitmi pa 2,29765 sekund. S 100 nitmi je čas reševanja narasel na 3,16766 sekund. Videli smo, da se čas ne zmanjšuje proporcionalno s številom niti, če je niti preveč celo naraste, saj računalnik zapravi veliko časa z organiziranjem dela med nitmi.

Dobljene rezultate smo shranili v datoteki za vizualizacijo .vtk in jo uvozili v programsko okolje Paraview, na ta način smo vizualizirali podatke, da so bili bolj pregledni. Rezultati so se smiselno ujemali z našimi robnimi pogoji.

Sistem enačb smo nato še shranili v obliki .json datoteke in ga rešili z uporabo Wolfram Mathematice. Wolfram Mathematica je sistem enačb rešila dosti hitreje, v le 0,1075 sekunde, rezultati pa so bili enaki tem, ki smo jih dobili z našo C++ kodo.

8 Viri

Literatura

[1] Matic Brank, Jernej Kovačič, Janez Povh, and Leon Kos. Izračun porazdelitve temperature v 2d prerezih. 2023.