POROČILO PROJEKTNE NALOGE PRI PREDMETU NRO

Petja Spačal
januar 2024
Vpisna številka: 23211243
Fakulteta za strojništvo, Ljubljana

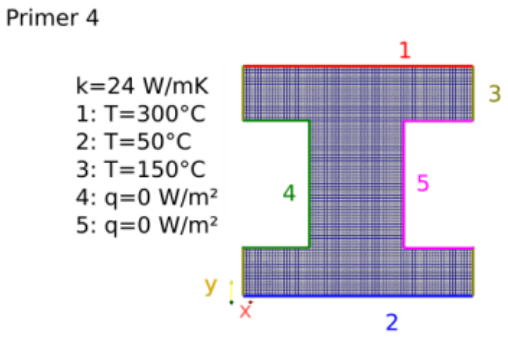
Kazalo vsebine

Kazalo tabel

Kazalo slik

1 Uvod

V projektni nalogi, smo imeli zastavljen primer prenosa toplote. Dodeljen nam je bil lik in njegovi robni pogoji. Naša naloga je bila določitev temperatur po celotnem liku. V našem primeru je lik zgledal tako.



Slika 1: Območje prevoda toplote [1]

Podano smo imeli še .txt datoteko, v kateri so bile zapisane:

- koordinate (x,y) točk, v katerih smo računali temperature,
- vrednost robnih pogojev in točke, kjer veljajo pogoji (temperatura, toplotni tok),
- seznam celic in točk, katere sestavljajo vozlišča celic (vsaka celica ima 4 točke).

Tako smo se lotili problema v programu Visual Studio, kjer smo v jeziku C++ začeli pisati kodo.

2 Pisanje C++ programa

2.1 Branje .txt datoteke

Najprej smo prebrali .txt datoteko, in podatke shranili v int/double spremenljivke, vektorje ali matrike. V funkciji main() smo najprej:

1. Odprli `.txt` datoteko s točkami in robnimi pogoji. Le-to smo shranili v isto mapo kot našo kodo: **std::ifstream indata("preimer4mreza.txt")**.
2. Nato smo brali vrstico po vrstici. Posamezno vrstico, ki je vsebovala številke in besede, smo najprej shranili kot string. Zahtevane podatke smo pretvorili z ukazom: **std::istringstream** v stream, iz katerega smo potem izluščili številko in besedo.
3. Za branje vrstic, v katerih je bilo samo število, smo najprej definirali vektor, npr. `std::vector<double> xkoord`. Nato smo s for zanko prebrali vrstice in jih z ukazom `xkoord.push_back(x)` shranili v vektor. Ker so bile v nekaterih vrsticah tudi vejice, smo uporabili ukaz `indata>>id>>vejica>>x>>vejica>>y`, s katerim smo definirali zaporedje znakov v eni vrstici.

4. Nepomembne podatke smo shranjevali v spremenljivke, katere nismo nikoli uporabili. Za izgradnjo matrike, katera bo vsebovala v vsaki vrstici vektor koordinat vozlišč celice, pa smo najprej definirali matriko `std::vector<std::vector<int>> matrikacell`. Nato smo s for zanko in ukazom `push_back` zgradili matriko.

2.1.1 Branje robnih pogojev

Robne pogoje smo brali na enak način kot prej vozlišča in celice.
Na koncu smo še shranili vrednosti robnih pogojev, prestopa toplote in tip robnega pogoja v vektorsko spremenljivko, da nam bo laže klicanje le-teh pozneje v programu.

2.2 Sestava matrike A in vektorja b

Najprej smo z zunanjo funkcijo preverili katere celice so sosednje, saj bo to pomembno kasneje za pisanje enačb. ID sosednjih celic smo shranili v matriko, katero smo najprej ustvarili tako, da je bila polna števil -1, nato pa se je zapolnila. Vrednosti -1, ki so ostale, so ponazarjale, da celica na tistem mestu nima sosednje celice.

2.2.1 Sestavljanje

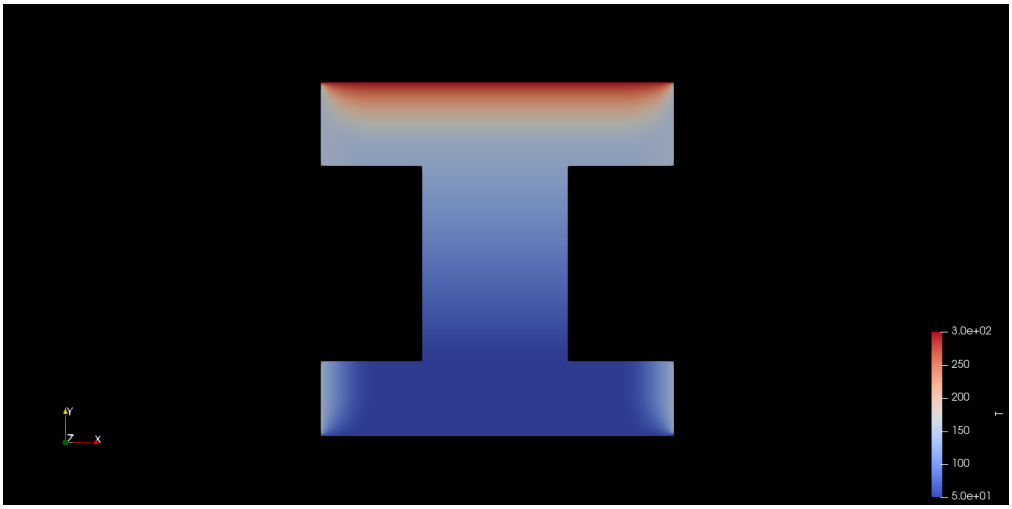
Najprej smo definirali prazno matriko M in vektor b. Nato smo s pomočjo zunanje funkcije sortirali robne pogoje. Sledilo je pisanje veliko for zank, v katerih smo vpisali enačbe, pripadajočemu robnemu pogoju. Na koncu smo dobili matriko M in vektor b. Velikost teh dveh je bila enaka številu vozlišč.

2.3 Reševanje sistema enačb

Za reševanje sistema enačb smo uporabili Gauss-Seidelovo metodo, katera je primerna za reševanje redkih matrik. Za to reševanje smo najprej definirali vektor T, kateremu smo dali začetno ugibano temperaturo. Izvedli smo 1000 iteracij računanja in s tem dobili vektor rešitev temperatur v celicah.

3 Vizualizacija

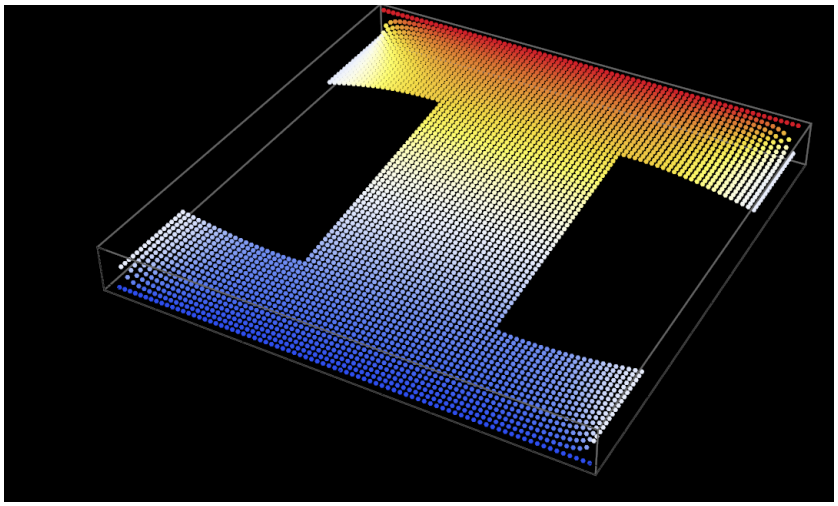
Za vizualizacijo smo shranili matriko temperatur in koordinat vozlišč v vtk format. Tako shranjene datoteke smo vizualizirali v okolju ParaView, ki avtomatsko prebere vtk datoteke. Dobili smo sledeče rezultate:



Slika 2: Potek temperature po liku

4 Program v Wolphram Matematici

Matriko M in vektor b smo v C++ programu skranili v .txt datoteko in poklicali v programu Wolphram Matematica, kjer smo merili hitrost reševanja sistema enačb. V Matematici je reševanje vzelo 63,01 s.



Slika 3: Izris v programu Matematica

5 Primerjava hitrosti

Nato smo merili čas še v C++ programu in dobili čas reševanja 239.62 s. Program smo poskušali izboljšati s paralelnim programiranjem. Uporabili smo knjižnico `#include <omp.h>`. S paralelnim programiranjem smo dobili na mojem računalniku, ki ima 8 jederni AMD Ryzen 7 4800H procesor, čas 55,09 sekund. S paralelnim programiranjem smo pridobili na kar 4,8

kratni pohitritvi. Nato smo uporabili še način branja podatkov s pointerji in s tem še dodatno pohitrili program na čas reševanja 10 sekund. Pointer je definiran npr. tako `double* A_row = A[jj].data()` in omogoča hitrejše branje podatkov, saj samo kaže na pot, kjer so le-ti shranjeni. Zanimivo je tudi to, da izračuni niso bili vedno enako hitri, ampak so varirali za nekaj sekund. To pa je najbrž zaradi zasedenosti računalnika.

	čas računanja osnovnega programa [s]	paraleriziran [s]	paraleriziran+pointerji [s]
Matematica	63	/	/
C++	239.62	55,09	10

Tabela 1: Primerjava hitrosti

6 Zaključek

Po napisanem programu v C++, smo dobili orodje za reševanje 2D problemov pri prevodu toplote. V projektni nalogi smo podrobneje spoznali programiranje v jeziku C++, hkrati pa smo utrdili znanje prenosa toplote in numeričnega reševanja po metodi končnih razlik. Ko smo napisali naš program, je trajanje računanja potekalo kar zajeten čas. Ko pa smo program parareliziral, smo ga pohitrili za skoraj petkrat. Z uporabo pointerjev pa smo izboljšali program še za petkrat.
V našem C++ programu, smo tudi oblikovali matriko A in vektor b, katere smo izvozili v .txt datoteko. To datoteko smo odprli v programu Matematica, kjer smo prav tako rešili sistem enačb. Čas reševanja je bil v tem primeru dober, vendar ne tako dober kot dobro optimiziran C++ program. Spoznali smo, da lahko vsak program izračuna prave vrednosti. Da pa bodo računi hitri, je potrebno uporabiti še razna orodja za paralerizacijo in pohitreno branje podatkov iz spomina.

Reference

[1] Matic Brank. Nro. vaje, 2023.