Univerza *v Ljubljani*
Fakulteta *za strojništvo*

Projektna naloga pri predmetu Napredna računalniška orodja

Izračun porazdelitve temperature - 2D MKR

Luka Uranič (23211047) in Adam Valjavec (23211332)

24. januar 2024

1 Uvod

V tem projektu smo obravnavali časovno neodvisen primer prenosa toplote v 2D prerezu, kjer smo za dane robne pogoje izračunali porazdelitev temperatur. Pri tem smo predpostavili temperaturno neodvisno toplotno prevodnost in robne pogoje. Hkrati smo zanemarili notranjo generacijo toplote.

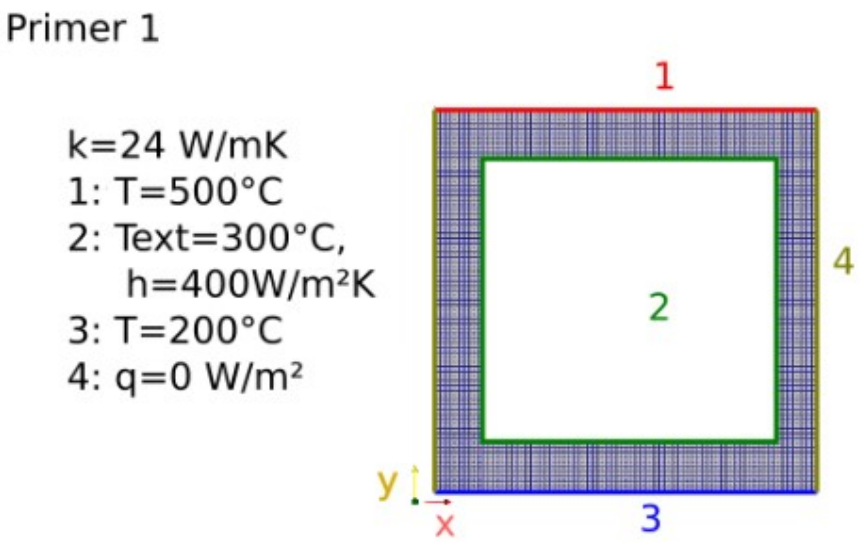
Tu je predstavljena enačba problema:

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

Problem vsebuje tri različne tipe robnih pogojev, ki so:

- Temperatura $T[°C]$
- Toplotni tok $q[\frac{W}{m^2}]$
- Prestop toplote s toplotno prestopnostjo $h[\frac{W}{m^2K}]$.

Problem smo reševali z metodo končnih razlik (MKR). Obravnavani primer pa je bil sledeč:



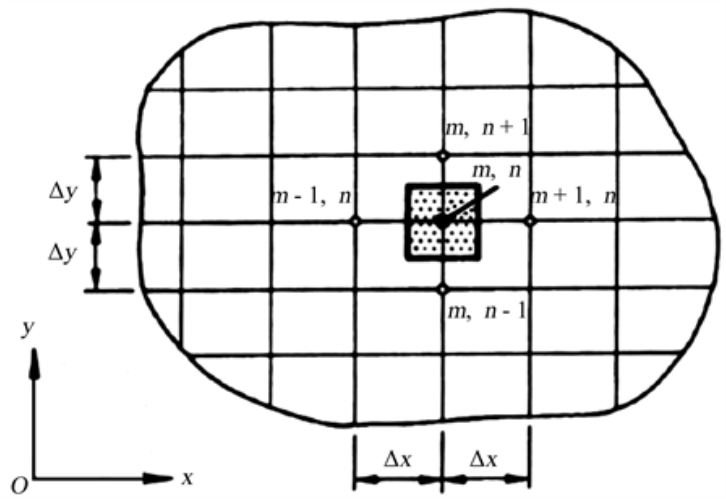
Slika 1: Prikaz obravnavnega problema

2 Fizikalno ozadje problema

V strojništvu, predvsem pa pri numeričnih simulacijah moramo probleme obravnavati celostno. Vprašati se moramo o njihovem fizikalnem ozadju, da na koncu, ko pridemo do rešitve, lahko to rešitev vrednotimo, se vprašamo če je ta rešitev fizikalno sploh smiselna. Tako sva se tudi lotila tega projekta, kot realen strojniški primer iz sveta. Iz podane mreže podatkov, sva ugotovila, da v tem projektu obravnavamo 1m širok kvadratni kanal, po katerem se pretaka tekočina. Na bokih je kanal izoliran. Tako sva že v začetku pričakovala manjši temperaturni gradient v x-smeri in večji temperaturni gradient v y-smeri.

3 Metoda končnih razlik

Dani primer smo reševali z metodo končnih razlik. Ta metoda je interpolacijska aproksimativna metoda, ki temelji na aproksimiranem zapisu odvodov s pomočjo diferenčnih shem. Problem moramo zato običajno diskretizirati, to pomeni ustvariti moramo vozlišča, ki predstavljajo točke, v katerih potem računamo temperaturo.



Slika 2: Prikaz diskretizacije.

Enačba za diferenčno shemo prvega odvoda v x-smeri se glasi takole:

$$\frac{\partial T}{\partial x}\Big|_{m+1/2,n} \approx \frac{T_{m+1,n} - T_{m,n}}{\Delta x}$$

za drugi odvod pa:

$$\frac{\partial^2 T}{\partial x^2}\Big|_{m,n} \approx \frac{T_{m+1,n} - 2T_{m,n} + T_{m-1,n}}{(\Delta x)^2}$$

Diferenčne sheme v y-smeri so precej podobne.

Diferencialno enačbo zapišemo za vsako notranje vozlišče, kjer namesto prvih in drugih parcialnih odvodov uporabimo te diferenčne sheme. Zapišemo tudi robne pogoje in pogoje prehoda, če ti obstajajo, prav tako s pomočjo diferenčnih shem. Na predlogi, smo na našo srečo že imeli zapisane, vse enačbe za različne robne pogoje, pogojev prehoda pa nismo potrebovali, ker je bilo naše telo homogeno, sestavljeno iz enega materiala. Metoda končnih razlik nam omogoča, da sistem enačb zapišemo kot množenje matrike z vektorjem, kar nam omogoča lažji zapis v računalniški spomin. Opazimo, da je matrika večinoma prazna in ima vrednosti le blizu diagonale, zato se je smiselno poslužiti Gauss-Seidlove metode reševanja. Gauss-Seidlova metoda reševanja sistema enačb je zelo primerna ranvo za matrike take narave. Je pa tudi iterativna, tako da potrebujemo tudi začetni približek, za katerega lahko vzamemo enega izmed robnih pogojev.

4 C++ program

V veliko oporo nama je bil demonstracijski program v MatLabu, po katerem sva se zgledovala. Prav tako nam je bilo prihranjeno ustvarjanje lastne diskretizacije, ker smo le to že imeli podano v .txt datoteki. Robni pogoji so tudi že bili zapisani. Najina naloga je bila, da sva napisala program, ki te podatke pravilno prebere, jih razvrsti po vektorjih, jih uporabi za izdelavo matrike

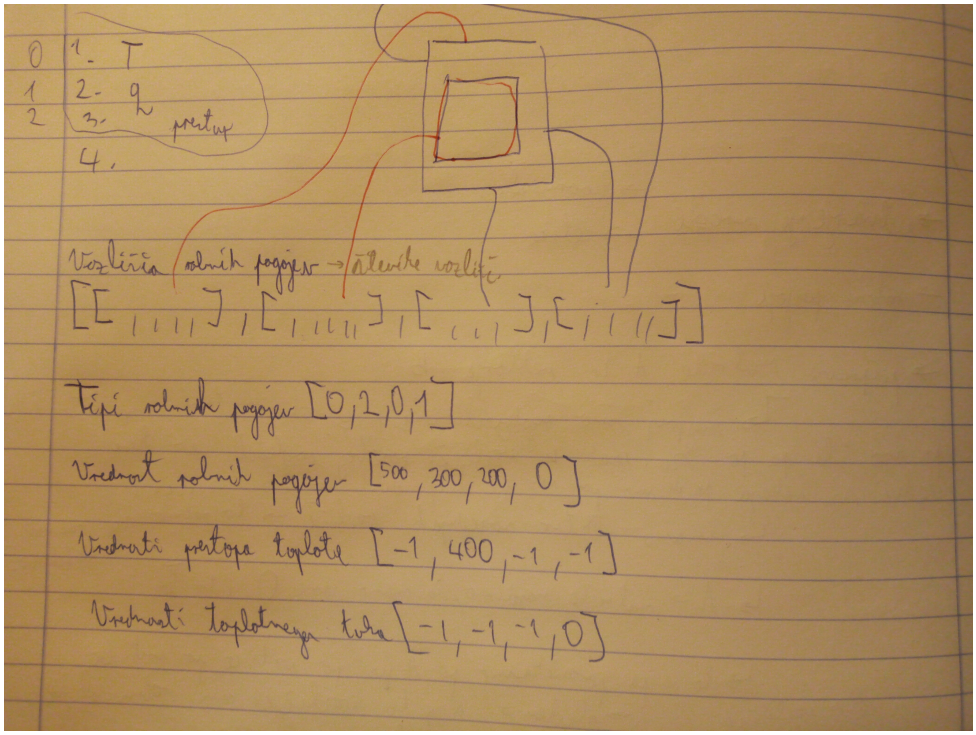
koeficientov in vektorja rešitev, reši sistem enačb in rezultat shrani v obliki, primerni za nadaljnjo grafično obdelavo.

Sprva sva definirala funkcijo, ki je kasneje potrebna v programu. Funkcijo ***preveri*** uporabljamo za določevanje pozicije sosedov. Funkcija uporablja if stavke in najprej preveri, koliko sta točki oddaljeni po eni osi. Če sta točki po eni osi oddaljeni zelo malo, pomeni, da sta po drugi osi soseda. To seveda drži le, če so argumenti te funkcije točke iz iste celice, za kar poskrbimo kasneje v programu. Definirala sva tudi osnovne veličine Δx inΔy, ki sva jih odčitala iz mreže in pa koeficient prevoda toplote.

4.1 Branje datoteke in razvrščanje

Najprej sva definirala vektorje, v katere se shranjujejo prebrani podatki. Program najprej prebere število točk, potem pa z uporabo for zanke zapiše podatke v želeni vektor. Program na enak način deluje tudi naprej pri zapisu celic. Pri robnih pogojih, sva program zapisala za najin primer in ni univerzalen (ne moremo ga uporabiti z drugimi mrežami). Pomikamo se po posameznem robnem pogoju in vrednosti zapisujemo v za to pripravljene vektorje. Najprej izpolnimo vektor tipa robnih pogojev, zatem vektor vrednosti robnih pogojev, potem vektor vrednosti prestopa toplote, nato še vektor vrednosti toplotnega toka in nazadnje zapišemo še vozlišča v zato pripravljen vektor. Tu sva se držala predlogov iz MatLab programa glede katera številka predstavlja posamezeno vrsto robnega pogoja itd.

Sledi še iskanje sosednjih vozlišč, kjer najprej definiramo vektor sosednjih vozlišč. Potem uporabljamo več notranjih for zank in if stavkov ter funkcijo, ki smo jo definirali na začetku, Program deluje tako, da se pomikamo za posamezno vozlišče. Za vsako vozlišče gremo skozi vse celice, kjer iščemo tiste celice, ki vsebujejo to vozlišče. Ko program najde eno izmed teh celic, se z uporabo for zanke, if stavka in prej definirane funkcije ***preveri*** določi položaj drugih vozlišč znotraj te celice na prvotno vozlišče. V kolikor sta vozlišči sosednji, se to zapiše v vektor na pravo mesto.



Slika 3: Robni pogoji na listu (pomoč pred programiranjem).

4.2 Generiranje sistema enačb in reševanje

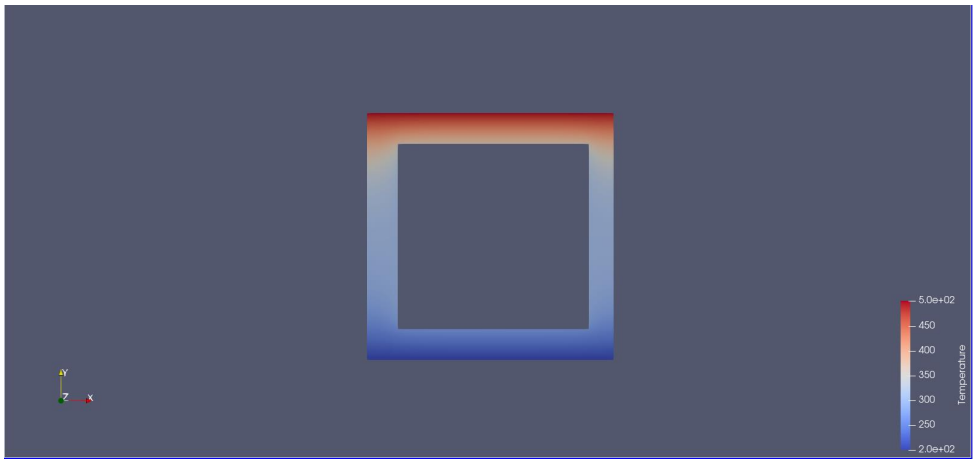
V programu sva najprej ustvarila matriko 0 in vektor 0. Program nato potuje po 1 vozlišče naprej, to tudi pomeni, da se premika po matriki in vektorju za eno vrstico nižje. Z if stavkom se kontrolira, če se vozlišče nahaja znotraj ali na robu. Če se nahaja znotraj, zapišemo primerne koeficiente na primerno mesto. V kolikor se nahaja na robu, s kombinacijo for in if stavkov ugotovimo, kateri robni pogoj je izpolnjen na tem vozlišču. Ko ugotovimo, kateri robni pogoj je izpolnjen si v lokalne vektorje shranimo njegove značilnosti. Potem program glede na tip robnega pogoja, glede na to katero sosednje vozlišče ne obstaja in značilnosti tega robnega pogoja v matriko in vektor vpiše ustrezne koeficiente.

Pri reševanju želimo meriti čas, zato potrebno kodo vključimo tu notri. Ustvarila sva vektor temperatur, ki ima začetne vrednosti, saj sva uporabila iterativno Gauss-Seidel metodo, po zgledu iz predloge.

Mrežo in pa vektor temperatur sva nato ročno zapisala in shranila v datoteko VTK

5 Grafični prikaz rezultatov

Rešitev predstavlja vektor temperatur, ki pa si ga brez vizualizacije težko predstavljamo in težko ugotovimo pravilnost rešitve. Zato sva datoteko VTK naložila v programsko okolje ParaView in si ogledala rešitev. Kot opazimo, so se najine predpostavke o gradientih iz začetka uresničile, zato lahko verjamemo v to rešitev. Gradient praktično nastopa le v smeri y-osi; medtem ko je pri konstantni višini y, v smeri x-osi stanje skoraj izotermno.



Slika 4: Prikaz v grafičnem okolju ParaView.

6 Primerjava: C++ in MatLab

V projektni nalogi nas je zanimalo tudi hitrost reševanja sistema enačb med različnimi programi. Izvedla sva tri poskuse: C++ program (Gauss-Seidel), MatLab-ov program (Gauss-Seidel) in MatLab-ov program (uporaba vgrajene funkcije). Najhitrejši je bil MatLab-ov program z uporabo linsolve. Drugi najhitrejši je bil program C++, najpočasnejši pa demonstracijski program v MatLab-u. Rezultati so naju kanček presenetili, kako hitro je računal MatLab z uporabo vgrajene funkcije. Razlago sva poiskala na spletu in ugotovila, da je MatLab jezik namenjen računanju z matrikami in vektorji in uporablja najbolj dovršene knjižnice, algoritme za delo z njimi, zato je tako hiter pri uporabi lastnih funkcij, ko te prednosti koristimo.

Nadalje sva se spraševala, zakaj pa je bil potem na 2. mestu C++ in ne MatLab-ov program z uporabo Gauss-Seidla, saj sva ravno tako delala z matrikami in vektorji. Izkaže se, da je tu

odločilna razlika o vrsti programskega jezika. V programu C++ sva uporabila paralerizacijo, ki omogoča hitrejše reševanje, saj izvaja več procesov hkrati. Prav tako se C++ programski jezik prevede v strojni jezik, ki ga procesorji zelo hitro lahko izvajajo. MatLab-ov jezik pa je interpretiran, kar pomeni, da ko poženemo kodo, mora vmesnik vsakič prebrati vrstico po vrstico in jo interpretirati, šele potem pride ukaz do procesorja, kar vzame nekaj časa, zato so v splošnem pri večjih količinah kode interpretirani programi počasnejši (MatLab-ov program ni uporabil vgrajene funkcije s katero bi dostopal do hitrih aloritmov, ampak je za računanje uporabljal naše algoritme in jih pošiljal skozi interpretator, zato je trajal čas reševanja toliko dlje) .

7 Možnosti za nadaljnje delo

V kodo bi lahko vstavili del, ki preverja, da nobena temperature ni višja od maksimalne temperature najvišjega robnega pogoja in nižja od minimalne temperature najnižjega robnega pogoja. Takšno je realno stanje v naravi. S tem bi se izognili nekaterim nepravilnim rešitvam. Seveda pa tak del kode podaljšuje delovanje programa, zato je to smiselno le v fazi testiranja programa.

Za nadaljnje delo, bi kodo lahko izboljšali tudi tako, da bi število iteracij pri reševanju sistema enačb zamenjali s spremembo povprečne temperature glede na prejšnjo iteracijo z uporabo while. S tem bi dosegli poljubno natančnost. Če ne potrebujemo visoke natančnosti bi s tem še bolj zmanjšali čas (dosežemo najmanjše število iteracij).

Še bolj bi čas reševanja sistema zmanjšali z uporabo knjižnic namenjenih za reševanje sistemov enačb z redkimi matrikami. S tem bi v programskem jeziku C++ prišli do bolj dovršenih algoritmov za reševanje dotičnega primera.

8 Zaključek

Prvo prebiranje navodil je v nama vzbudilo ne sigurnost in mnoga vprašanja, kako se bova naloge sploh lotila, kako bova vedela, je izračun pravilen... Naloge sva se lotila čim bolj celostno in kot realen primer, tako da sva najprej malo osvežila znanje prenosa toplote in pa ozadje metode končnih razlik. Okrepljena s teoretičnim predznanjem sva se podala v programiranje, kjer sva program sestavljala po delih in ga sproti testirala. Pomagala sva si z demonstracijskim primerom programa iz MatLab-a, iz katerega sva jemala koncepte za reševanje različnih ovir. Po urah in urah programiranja, testiranja kode, odpravljanja neukrotljivih "errorov", sva le prispela do končnega rezultata. Ta rezultat pa ni le vektor temperatur in slikica temperaturne porazdelitve v ParaView-u. Je znanje, ki smo ga pridobili tekom projekta. Povezali smo snov različnih prejšnih predmetov in se naučili, kako rešiti tako zahteven problem s pomočjo računalnika. Hkrati smo spoznali značilnosti različnih programskih jezikov. Zapolnila si bova, da je za dolge simulacije, z mnogo točkam smiselno programe prevesti s »compilerji« , saj jih tako procesorji hitreje rešujejo, še bolje pa je za take primere uporabljati programska okolja namenjena reševanju takih primerov, saj vsebujejo že zato dobro pripravljene, hitre algoritme in je računanje sila učinkovito.