Eötvös Loránd Tudományegyetem

Természettudományi Kar

Kupás Vendel Péter

Alkalmazott matematikus MSc

Az időbeli többrácsos redukciós módszer és alkalmazásai

Szakdolgozat

Témavezető:

Dr. Fekete Imre

Alkalmazott Analízis és Számításmatematikai Tanszék

Budapest

2024

Tartalomjegyzék

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Fekete Imrének, aki felkeltette érdeklődésemet a téma iránt és iránymutatásával, meglátásaival segítette a dolgozat létrejöttét.

Hálával tartozom családomnak és barátaimnak a támogatásukért.

Bevezetés

A differenciálegyenletek témaköre régóta kutatott, szerteágazó része a matematikának. A természetben lejátszódó jelenségeket, egyéb társadalmi és pénzügyi folyamatokat gyakran ezek segítségével tudjuk leírni. Sok esetben egy olyan rendszer időbeli viselkedésére vagyunk kíváncsiak, amelynek tudjuk a kezdeti állapotát, és ismerjük a dinamikáját leíró egyenleteit. Sajnos a legtöbb kezdetiérték-problémát nem tudjuk analitikusan megoldani, ezért numerikus módszerekkel kell közelítenünk a megoldást.

Az időfüggő differenciálegyenletek numerikus megoldását sokféle különböző módszerrel végezhetjük, melyek nagy része idődiszkretizáláson alapul. A klasszikus egylépéses módszerek esetében a megoldás közelítését egy pontban az előző pontban felvett értékből számoljuk ki. Ekkor egy szekvenciális algoritmust kapunk, amely futási ideje egyenesen arányos az időbeli rácspontok számával. A mai processzorok elérték a maximális teljesítőképességük határát, így a többmagos számítógépek megjelenése óta természetes igény alakult ki differenciálegyenletek párhuzamos megoldási módszereinek kidolgozására. A szakdolgozatban röviden végigtekintünk a párhuzamosított numerikus módszerek körülbelül 60 éves történelmén, majd részletesen tárgyaljuk az ún. időbeli többrácsos redukciós (Multigrid Reduction In Time, a továbbiakban MGRIT) módszert, melyet 2014-ben vezetett be R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan és J. B. Schroder. [4]

A többrácsos (Multigrid, a továbbiakban MG) módszerek hatékonyan alkalmazhatóak klasszikus térbeli elliptikus egyenletek megoldásainak a közelítésére, viszont nem terjeszthetőek ki magától értetődő módon az idő dimenzióra. Az MGRIT algoritmus tekinthető egy tér-idő MG módszernek, amely az időtengelyt speciálisan kezeli a többi térbeli tengelyhez képest. A módszer bevezetéséhez először megvizsgáljuk a Lions, Maday és Turicini által 2001-ben bevezetett, időpárhuzamosított Parareal algoritmust, amely értelmezhető egy időtengelyre alkalmazott kétszintes többrácsos módszerként is. Az algoritmus jól használható, viszont a két szint használata limitálja a lehetséges párhuzamosítás mértékét. A Parareal algoritmus könnyen kiterjeszthető több rácsra, amely következtében tulajdonképpen az MGRIT módszerhez jutunk. Ennek a megközelítésnek az az előnye, hogy az algoritmus könnyen beépíthető már létező kódokba, mivel csak egy egylépéses módszer szubrutinjára van szükségünk, amivel a következő pontba lépünk

az előző ismeretéből. A [4] cikk szerzői a módszer bevezetése mellett implementálták is az algoritmust, amely elérhető az XBraid publikus GitHub repozitóriumban [22].

Az 1. fejezetben felelevenítjük azokat az alapfogalmakat, amelyekre szükségünk van a dolgozat megértéséhez. A későbbiekben gyakran fogunk visszautalni az itt leírtakra. Bevezetjük a kezdetiérték-probléma fogalmát, röviden megnézzük az egylépéses módszereket, és tömören leírjuk az MG módszerek elméletét.

A 2. fejezetben az időpárhuzamosított módszerek témakörét tárgyaljuk. Négy nagy csoportba soroljuk az algoritmusokat aszerint, hogy milyen műveleteket végeznek párhuzamosan. Megpróbáljuk számba venni a legfontosabb ötleteket, amelyek hozzájárultak a témakör ma is dinamikus fejlődéséhez. A fejezet végén részletesen leírjuk a Parareal algoritmus általános formáját.

A 3. fejezetben kerül bevezetésre az MGRIT módszer. Először megmutatjuk, hogy lineáris feladat esetén a Parareal algoritmus tekinthető egy kétrácsos algoritmusnak az időtengelyen, majd több rácsra történő általánosítással az MGRIT algoritmushoz jutunk. A paraméterek optimalizálása után összehasonlítjuk az MGRIT algoritmus és a klasszikus szekvenciális módszer futási idejét a rendelkezésre álló processzorok számának függvényében. Végül a teljes approximációs séma segítségével kiterjesztjük a módszert nemlineáris feladatokra is.

Az utolsó 4. fejezetben különböző feladatokon végzett mérések eredményeit írjuk le. A fejezet elején olyan problémákkal foglalkozunk, amelyek közönséges differenciálegyenletek segítségével jellemezhetőek. Először egy kétdimenziós lineáris problémán ellenőrizzük az algoritmus helyességét, majd egy nagyobb, ugyancsak lineáris problémán vizsgáljuk a módszer futási idejét. Ezután egy nemlineáris problémát, a SIR-modellt tanulmányozzuk. Részletesen leírjuk, hogy a diszkretizálás és az implementáció során milyen eszközöket használtunk, és azt analizáljuk, hogy mikor konvergál az algoritmus a megoldáshoz. Ezt követően áttérünk a parciális differenciálegyenletek esetére, ahol a legrészletesebb elemzéseket a lineáris és nemlineáris hővezetési egyenleteken végezzük. A módszer paramétereinek a futási időre és a szükséges iterációk számára gyakorolt hatását több szemszögből vizsgáljuk. A tesztekhez az XBraid szoftvert használtuk, és ezeket az ELTE szerverein futtattuk.

1. fejezet

Alapfogalmak, klasszikus numerikus módszerek

Ebben a fejezetben azokat a fogalmakat gyűjtöttük össze, melyek ismerete szükséges a dolgozat későbbi részeinek a megértéséhez. Bár az Olvasóról feltételezzük, hogy tisztában van a differenciálegyenletek alapvető fogalmaival, először leírjuk, hogy a dolgozatban pontosan mit értünk kezdetiérték-probléma alatt. Ezután röviden átvesszük az egylépéses módszereket, mivel az MGRIT algoritmus tulajdonképpen egy egylépéses módszert alakít át időpárhuzamossá. Végül megnézzük a klasszikus, térbeli elliptikus egyenletekre alkalmazható többrácsos módszereket. Az itt szereplő fogalmakra később gyakran hivatkozni fogunk, ezért érdemes gyorsan végigfutni a fejezeten akkor is, ha az Olvasó magabiztos tudással rendelkezik ebben a témában.

1.1. Kezdetiérték-probléma

1.1. Definíció Legyen $d \in \mathbb{N}^+$ szám, $G \subset \mathbb{R} \times \mathbb{R}^d$ egy tartomány (azaz összefüggő nyílt halmaz), $(t_0, u_0) \in G$ egy adott pont és $f : G \to \mathbb{R}^d$ folytonos leképezés. Ekkor az

$$\begin{aligned} u'(t) &= f(t, u(t)) \\ u(t_0) &= u_0 \end{aligned} \tag{1.1}$$

problémát kezdetiérték-feladatnak nevezzük.

Egy kezdetiérték-feladat megoldása azt jelenti, hogy meghatározzuk az összes olyan I nyílt intervallumot és $u : \mathbb{R} \to \mathbb{R}^d$ függvényt, amely az $I \subset \mathbb{R}$ intervallum pontjaiban egyrészt behelyettesíthető a feladatba, másrészt pedig azt ki is elégíti.

1.2. Definíció Az (1.1) kezdetiérték-feladat megoldásai olyan $u : I \to \mathbb{R}^d$ folytonosan differenciálható függvények, melyekre

- $\diamond$ minden $t \in I$ esetén $(t, u(t)) \in G$,
- $\diamond$ $u'(t) = f(t, u(t))$, minden $t \in I$,
- $\diamond$ $t_0 \in I$ és $u(t_0) = u_0$.

Amikor a kezdetiérték-feladattal egy természettudományos problémát, jelenséget modellezünk, akkor elvárjuk, hogy létezzen egyértelmű megoldás. Ennek igazolásához használhatjuk a következő két tételt, melyek elégséges feltételt fogalmaznak meg a megoldás lokális és globális egyértelműségére.

1.3. Tétel (Picard-Lindelöf) Ha az f függvény a $(t_0, u_0) \in G$ pontban a második változójában teljesíti a lokális Lipschitz-feltételt, akkor az (1.1) kezdetiérték-problémának létezik megoldása, és az lokálisan egyértelmű.

1.4. Tétel Ha minden $(t_0, u_0) \in G$ kezdőpont esetén az (1.1) kezdetiérték-probléma lokálisan egyértelműen oldható meg, akkor globálisan is egyértelműen oldható meg.

A továbbiakban feltesszük, hogy az (1.1) kezdetiérték-problémának egyértelmű a megoldása egy adott I intervallumon. Mivel a t változó az időt jelöli, ezért az (1.1) feladat megoldása azt írja le, hogy a rendszer időben hogyan változik. Gyakorlatban ez azt jelenti, hogy ismerjük a rendszer állapotát egy rögzített kezdeti időpontban, és a rendszer későbbi viselkedését szeretnénk meghatározni, azaz az $u(t_0)$ ismeretében az $u(t)$ függvényre vagyunk kíváncsiak, amikor $t > t_0$. Természetesen nem jelent megszorítást, ha a kezdőpontot $t_0 = 0$ értéknek választjuk. Továbbá rögzíthetünk egy végső T időpontot is.
Ekkor a kezdetiérték-feladat a következő alakot ölti:

$$\begin{aligned} u'(t) &= f(t, u(t)), \quad t \in [0, T], \\ u(0) &= u_0 \end{aligned} \tag{1.2}$$

A későbbiekben, ha kezdetiérték-feladatról beszélünk és külön nem emeljük ki, akkor mindig az (1.2) alakú problémára gondolunk.

1.2. Egylépéses módszerek

A megoldás létezését bizonyító tételek, mint például a Picard-Lindelöf tétel, nem konstruktívak, azaz nem adnak eszközt az (1.2) kezdetiérték-probléma megoldásainak előállítására. Általában a megoldások csak speciális jobb oldal esetén adhatóak meg képletek segítségével. Ahelyett, hogy a pontos megoldás kiszámolására koncentrálnánk, a dolgozatban olyan numerikus megoldásra törekszünk, amelyek jól közelítik azt. Az egyik legrégebbi és legelemibb ötlet, hogy használjunk egylépéses numerikus módszereket.

Osszuk fel a $[0,T]$ intervallumot N részre a $\{0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T\}$ rácsháló pontjaival. A megoldás ezen $t_0,\ t_1,\ldots,t_{N-1},\ t_N$ pontokban felvett értékét szeretnénk közelíteni rendre a $u_0,\ u_1,\ldots,u_{N-1},\ u_N$ értékekkel. Legtöbbször ekvidisztáns rácshálót használunk a gyakorlatban, azaz a szomszédos osztópontok között a lépésköz azonosan $h = \frac{T}{N}$. Egylépéses módszerek esetében az u_{i+1} közelítő értéket a megelőző u_i érték segítségével határozzuk meg. Képlettel ez a következőféleképpen fejezhető ki:

$$\begin{aligned} u_0 &= u_0, \\ u_{i+1} &= \Phi_{i+1}(u_i) + g_{i+1}, \quad i = 0,1,\ldots,N-1, \end{aligned} \tag{1.3}$$

ahol a $\Phi_i(\cdot)$ függvények és g_i számok ($i = 1,2,\ldots,N$) karakterizálják a módszert. Explicit módszer esetében könnyen meghatározhatóak a $\Phi_{i+1}(u_i)$ értékek, míg az implicit esetben gyakran egy iteratív módszert kell használnunk. Például a legegyszerűbb explicit módszer, az explicit Euler módszer esetébe $\Phi_{i+1}(u_i) = u_i + hf(t_i,u_i)$ és $g_{i+1} = 0$. A gyakorlatban általánosabb Runge-Kutta módszereket használnak, amelyek már magasabb konzisztencia renddel és nagyobb stabilitási tartománnyal rendelkeznek. Ezen családon belül legnépszerűbb explicit módszer az RK4 módszer, melynek alakja

$$\begin{aligned} k_1 &= f(t_i,u_i), \\ k_2 &= f\left(t_i + 0.5h, u_i + 0.5hk_1\right), \\ k_3 &= f\left(t_i + 0.5h, u_i + 0.5hk_2\right), \\ k_4 &= f\left(t_i + h, u_i + hk_3\right), \\ u_{i+1} &= u_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \end{aligned}$$

amely szintén felírható a fenti (1.3) alakban. Az (1.3) egyenletekből könnyen látszik, hogy az egylépéses módszer által adott megoldás megegyezik a következő $A(u) = g$ egyenletrendszer megoldásával:

$$Au = \begin{bmatrix} I & & & \\ -\Phi_1 & I & & \\ & \ddots & \ddots & \\ & & -\Phi_N & I \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_N \end{bmatrix} = g. \tag{1.4}$$

Érdemes megjegyezni, hogy az A leképezés elemei a Φ_i függvények, amelyek nem feltétlenül lineárisak. Ha ezen függvények mégis lineárisak, akkor A egy mátrix, melynek elemei valós számok. Ekkor az $Au = g$ lineáris egyenletrendszert kell megoldanunk. A triviális ötlet, hogy fentről lefelé sorban haladva számoljuk ki az u vektor elemeit. Ekkor éppen az egylépéses módszert kapjuk vissza. Az időben párhuzamosított numerikus módszerek esetében más módon oldjuk meg az egyenletrendszert.

1.3. Többrácsos (MG) módszerek

A klasszikus elliptikus differenciálegyenletek (Laplace- vagy Poisson-feladatok) egyik legjobban bevált megoldási módszere az MG módszer, amely több rácson végzett műveletek segítségével határoz meg egy közelítő megoldást. Először definiálunk egy finom rácsot a feladaton, amelyen szeretnénk közelíteni a tényleges megoldást. A véges differenciák módszerével, vagy a végeselem módszer használatával a probléma egy $Au = b$ lineáris egyenletrendszerre vezethető vissza. A véges differenciák módszerét általában ekvidisztáns felosztású rácshálón alkalmazzuk, és a módszer azon alapul, hogy az egyenletben szereplő deriváltakat megfelelő különbségi hányadosokkal közelítjük. A kapott A mátrix tipikusan szimmetrikus, sávos szerkezetű és ritka, azaz minden sorában a probléma méretétől függetlenül konstans sok nem 0 elem szerepel. Például egy dimenziós esetben a második derivált közelítésére tipikusan használt másodrendű centrális differenciaséma használatával egy tridiagonális mátrixhoz jutunk. Ha pontos közelítésre van szükségünk, akkor kis lépésközű rácsot használunk, aminek következtében a kapott egyenletrendszer mérete olyan nagy lesz, hogy sok időbe telik meghatározni a megoldást. Ezen segít a többrácsos módszer, amely iteratív módon oldja meg az egyenletrendszert durvább rácsokat használva. Ebben a fejezetben röviden összefoglaljuk az MG módszerek alapvető elméletét a [16] jegyzetre támaszkodva, ahol a témakör részletesebb leírása található példákkal kiegészítve.

Tegyük fel, hogy a véges differencia módszert használva a h lépésközű rácson az

$$A_h u = b_h$$

lineáris egyenletrendszerhez jutunk, és adott egy w közelítése a megoldásnak. Ezt a w vektort szeretnénk javítani, hogy közelebb kerüljünk az igazi megoldáshoz. Először vizsgáljuk azt az esetet, amikor két rácsunk van. A finomabb Ω_h, amihez az $A_h u = b_h$ lineáris egyenletrendszer tartozik és egy durvább H lépésközű Ω_H, amely például minden második pontját tartalmazza a finom rácsnak. Szeretnénk meghatározni egy p javító vektort, hogy az $u = w + p$ érték közelebb legyen az igazi megoláshoz. Legyen

$$r_h = b_h - A_h w$$

a hibavektor a finom rácson (a későbbiekben néha reziduálisként illetve maradékvektorként is hivatkozunk rá). Ekkor ha p-t úgy választjuk, hogy kielégítse az

$$A_h p = r_h$$

egyenletet, akkor nem meglepő módon $u = w + p$ választással épp a pontos megoldást kapjuk

$$A_h u = A_h(w + p) = A_h w + A_h(A_h^{-1}(b_h - A_h w)) = b_h.$$

A kétrácsos módszer esetében a segédfeladatban a p vektor meghatározásánál nem az A_h mátrixot, hanem annak egy A_H közelítését használjuk, melyet a durvább rácson alkalmazott véges differencia módszerből nyerünk. Így továbbra is csak egy közelítését kapjuk az igazi megoldásnak, viszont gyorsabban meg tudjuk határozni. A problémát az jelenti, hogy a dimenziószámok nem egyeznek meg, mert a p és r_h vektorok a finomabb, míg az A_H mátrix a durvább rácson van értelmezve. Ezért definiáljuk az $R : V_h \to V_H$ szűkítő (megszorítás vagy restrikció) és $P : V_H \to V_h$ kiterjesztő (prolongáló) leképezéseket a V_h finom, illetve V_H durva rácshoz tartozó alterek között. Így a következő iterációhoz jutunk:

$$u^{k+1} = u^k - PA_H^{-1}R(A_h u^k - b_h). \tag{1.5}$$

Ha az $r_h \in V_h$ vektornak nincs V_H-beli komponense, akkor $Rr_h = R(A_h u^k - b_h) = 0$, azaz az iterációs lépés nem javít, az iteráció leáll u^k-ban. Ezért az eddig leírtakon túl ún. elősimítást alkalmazunk, hogy a nemtriviális magtér okozta problémát és az abból adódó sajátfüggvény oszcillációját feloldjuk. Az elősimitást általában klasszikus iteratív eljárásokkal (csillapított Jacobi- vagy Gauss–Seidel-iterációkkal) végezzük.

1. Algoritmus Kétrácsos módszer

1: Tegyük fel, hogy adott az u_h^k közelítés.

2: Simítás: A B mátrix alkalmas megválasztásával hajtsunk végre iterációt (Jacobi, Gauss-Seidel):

$$\begin{aligned} v_0 &= u_h^k \\ v_{j+1} &= v_j - B^{-1}(A_h v_j - b_h), \quad j = 0,1,\ldots,l-1 \\ v &= v_l \end{aligned}$$

3: Reziduális kiszámítása és leszűkítése:

$$\begin{aligned} r_h &= A_h v - b_h \\ r_H &= R(r_h) \end{aligned}$$

4: Megoldás a durva rácson:

$$A_H e_H = -r_H$$

5: Prolongálás és korrekció:

$$\begin{aligned} e_h &= P(e_H) \\ u_h^{k+1} &= v + e_h \end{aligned}$$

Az iteratív eljárásoknál tipikusan kétfajta leállási feltételt fogalmazhatunk meg. Az egyik az iterációszámon alapuló, amikor előre meghatározzuk, hogy hány iterációt fogunk végrehajtani. A másik a toleranción alapuló, amikor valamilyen hibamennyiség egy bizonyos érték alá

csökkenése után állunk le. Az MG módszernél a reziduális vektor eukleideszi normája egy jó mérőszám a hibára nézve.

Jelölje u^* a pontos megoldást. Egy kis számolás után belátható, hogy az $e_h^k = u_h^k - u^*$ approximációs hibára fennáll az

$$e_h^{k+1} = \underbrace{(A_h^{-1} - PA_H^{-1}R)A_h(I - B^{-1}A_h)^l}_{K} e_h^k \tag{1.6}$$

egyenlőség. Ha az elősimítás mellett az iteráció végén úgynevezett utósimítást is végrehajtunk a korrekció után, és a megszorító és prolongáló leképezéseket úgy választjuk, hogy $R = P^T$ legyen, akkor az előbb definiált K mátrix a

$$\hat{K} = (I - B^{-1}A_h)^l(A_h^{-1} - PA_H^{-1}P^T)A_h(I - B^{-1}A_h)^l$$

alakot ölti, és szimmetrikus lesz az A_h-skalárszorzatra nézve.

Véges differencia módszer használata esetén az egyik legegyszerűbb prolongáció a lineáris kiterjesztés, amikor a finom rácson értelmezett megoldás értékeit a közeli durva rácspontokban felvett értékek lineáris kombinációjaként számítjuk ki. A legegyszerűbb leszűkítés a megszorítás, amikor a megtartjuk a durva pontokban a finom rácsbeli értékeket. Ezen leképezések mátrixa azonban nem egymás transzponáltja, így a megszorítást úgy szokás felírni, hogy mátrixa a lineáris kiterjesztés mátrixának transzponáltja legyen.

A módszer akkor konvergens, ha a K mátrix spektrálsugara kisebb mint 1, azaz ha $\rho(K) < 1$. Legyen $\|.\|_a$ és $\|.\|_b$ két alkalmasan megválasztott, a finom rácson értelmezett vektornorma, és jelölje

$$\|A\|_{a,b} = \max_{x \neq 0} \frac{\|Ax\|_a}{\|x\|_b}$$

az általuk indukált mátrixnormát. Ekkor a $\rho(K) \leq \|K\|_{a,a} = \|K\|_a$ egyenlőtlenség miatt elég igazolni, hogy $\|K\|_a < 1$.

1.5. Definíció

Egy MG módszerre teljesül az $\alpha-$approximációs tulajdonság, ha létezik olyan $C > 0$ konstans, hogy

$$\|(A_h^{-1} - PA_H^{-1}P^T)\|_{a,b} \leq Ch^\alpha.$$

Egy MG módszerre teljesül az $\alpha-$simító tulajdonság, ha létezik olyan $\varepsilon_l \to 0$ h-tól független sorozat, hogy

$$\|A_h(I - B^{-1}A_h)^l\|_{b,a} \leq \varepsilon_l h^{-\alpha}.$$

1.6. Tétel Ha egy MG módszerre valamely $\alpha > 0$ esetén teljesül az $\alpha-$approximációs és $\alpha-$simító tulajdonság, akkor létezik olyan $l \in \mathbb{N}^+$ szám, hogy az l darab simítást tartalmazó kétrácsos algoritmus konvergens. Azaz van olyan $\sigma < 1$, hogy

$$\|e_h^{k+1}\|_a \leq \sigma \|e_h^k\|_a \quad \forall k \in \mathbb{N}.$$

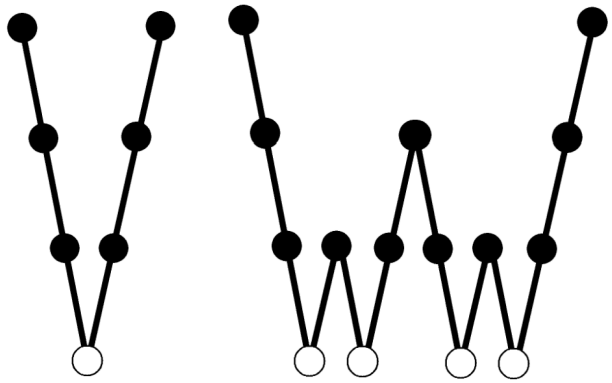
Bizonyítás. A két tulajdonságot kombinálva rögtön kapjuk az eredményt.

$$\|e_h^{k+1}\|_a = \|Ke_h^k\|_a = \|(A_h^{-1} - PA_H^{-1}P^T)A_h(I - B^{-1}A_h)^l e_h^k\|_a \leq$$
$$\leq \|(A_h^{-1} - PA_H^{-1}P^T)\|_{a,b}\|A_h(I - B^{-1}A_h)^l e_h^k\|_b \leq \|(A_h^{-1} - PA_H^{-1}P^T)\|_{a,b}\|A_h(I - B^{-1}A_h)^l\|_{b,a}\|e_h^k\|_a \leq$$
$$\leq Ch^\alpha \varepsilon_l h^{-\alpha}\|e_h^k| = C\varepsilon_l\|e_h^k\|_a$$

Válasszuk l-et akkorára, hogy $\sigma = C\varepsilon_l < 1$ teljesüljön. ■

Természetesen a K mátrix függ a diszkretizálástól, a prolongáló és a leszűkítő operátorok, illetve a simító iteráció választásától. Ezért minden esetben külön meg kell vizsgálnunk, hogy a módszerre teljesülnek-e az előbb leírt tulajdonságok alkalmasan választott normák esetén. Általánosságban az mondható el, hogy elliptikus feladatok esetén mind a véges differenciák módszerét, mind a végeselem módszert használva konvergens algoritmushoz jutunk klasszikus simító iterációk használatával.

A kétrácsos algoritmusnál a futási időben ott nyerünk, hogy a lineáris egyenletrendszert nem a finom, hanem a durva rácson oldjuk meg. De mi van akkor, ha a durva rácshoz tartozó egyenlet is túl nagy ahhoz, hogy direkt oldjuk meg? Egyszerű ötlet, hogy definiálunk egy még durvább rácsot, és a két durvább rácsra is egy kétrácsos módszert alkalmazunk. Így igazából el is jutottunk a háromrácsos módszerhez. Általánosan a többrácsos eset hasonlóan, rekurzív módon kezelhető. A rekurzív hívások sorrendjének tekintetében többfajta ciklust különböztetünk meg. A legegyszerűbb V-ciklusban minden rácson elősimítunk, majd kiszámítjuk és leszűkítjük a reziduálist (lásd 1.1. ábra). A legdurvább rácson megoldjuk a kapott lineáris egyenletrendszert, majd sorban visszaprolongáljuk a kapott megoldást a finomabb rácsokra, és közben utósimítást is végzünk a pontosabb közelítés érdekében. Két rács esetén nincs jelentősége az utósimításnak, mivel a következő iteráció elején elősimítás következik. Viszont ha több rácson dolgozunk, akkor az elősimításhoz hasonlóan egy egyszerű iterációval megvalósítható utósimítást nem mindig elősimítás követ.



1.1. ábra. V- és W-ciklus 4 rács esetén.

Másik változatot kapunk, ha minden szinten kétszer hívjuk meg az algoritmust rekurzívan az eggyel durvább rácson. Ekkor az úgynevezett W-ciklushoz jutunk (lásd 1.1. ábra).

Természetesen nem csak egyszer vagy kétszer lehet meghívni rekurzívan az algoritmust, hanem tetszőleges $\gamma \in \mathbb{N}^+$ számszor. Ha γ értékét nagynak választjuk, akkor egy ciklus (iteráció) alatt többet javítunk a közelítésen, viszont a lépésszám megnő. A többrácsos módszer egyik előnyös tulajdonsága, hogy kis γ értékek esetén tetszőleges szintszám esetén a lépésszám optimális marad, azaz lineáris a feladat (a legfinomabb rács) méretében.

1.7. Tétel Tegyük fel, hogy L szintünk van, és a szinteket a legfinomabbtól a legdurvábbig sorszámozzuk. Jelölje $T(L)$ az algoritmus futási idejét és N_l a feladat méretét, azaz a rácspontok számát az l-edik legfinomabb rácson. Továbbá tegyük fel, hogy a ritkítás mértéke minden szinten ugyanakkora

$$m = \frac{N_l}{N_{l+1}}, \quad l = 1,2\ldots, L-1.$$

Ha γ-ra teljesül, hogy

$$\frac{\gamma}{m} < 1,$$

akkor a többrácsos módszer egy iterációjának a lépésszáma független a szintek számától és lineáris a legfinomabb rács méretében. Azaz létezik $b \in \mathbb{R}$ konstans, hogy

$$T(L) < bN_1.$$

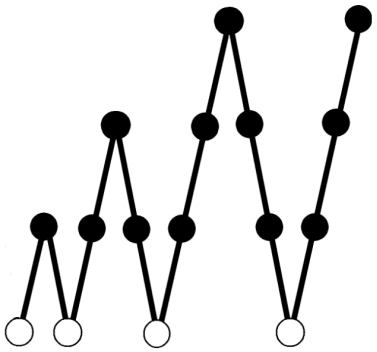
Bizonyítás. Jelölje $T(L)$ az L rácsot használó algoritmus futási idejét. Az l-edik szinten a simítás, a reziduális meghatározása, a restrikció és a prolongáció műveletek elvégezhetőek $O(N_l)$ lépésben, mert az l-edik szinthez tartozó A mátrix ritka és sávos szerkezetű. Így egy rekurzív képlethez jutunk, ahol használva a mértani sor összegképletét a tétel bizonyítását kapjuk.

$$T(L) = bN_1 + \gamma T(L-1) = bN_1 + \gamma(bN_2 + \gamma T(L-3)) =$$

$$= bN_1 + b\frac{\gamma}{m}N_1 + \gamma^2 T(L-3) = \cdots = bN_1 \sum_{l=0}^{L} \left(\frac{\gamma}{m}\right)^l < \frac{b}{1-\frac{\gamma}{m}}N_1$$

■

A gyakorlatban a V- és a W-ciklust szokták használni. A módszer fontos része a jó kezdőpont választás. Ha kezünkben van egy jó közelítés, akkor el tudjuk indítani az algoritmust, viszont gyakran nincs ismeretünk a megoldásról. Ekkor használhatjuk az F-ciklust, az úgynevezett teljes ciklust (gyakran FMG-cikluskénti is szoktak rá hivatkozni). Ekkor a legdurvább rácsról indulunk, ahol megoldjuk a kisebb méretű problémát, majd a megoldást felprolongáljuk az eggyel finomabb rácsra, ahonnan egy V-ciklust indítunk. Ezt rekurzívan addig ismételjük, amíg a legfinomabb rácsra érünk (lásd 1.2. ábra). Érdemes még megjegyezni, hogy az algoritmus lépésszáma lineáris marad a feladat méretében az F-ciklus használatával is.



1.2. ábra. F-ciklus 4 rács esetén.

Több rács esetén a szomszédos rácsok méreteinek a hányadosa nem kell, hogy állandó legyen. A megoldandó feladat karakterisztikájától függően lehetőségünk van az úgynevezett ritkító faktor használatára és értékének változtatására. Sok esetben a megfelelő ritkítási stratégia megválasztásával jelentősen csökkenthető a módszer futási ideje.

Az MG eljárást nemcsak klasszikus elliptikus, hanem időfüggő parciális differenciálegyenletekre is szokták alkalmazni. Ebben az esetben a térbeli irányokhoz hasonlóan az időtengelyen is több felosztást használunk, és a módszert tér-idő (space-time) MG módszernek hívjuk. Mint látni fogjuk, az MGRIT algoritmus is tekinthető tér-idő MG módszernek, viszont speciálisan, különböző módon használja az időtengelyt, mint a térbeli irányokat.

2. fejezet

Időpárhuzamosított módszerek

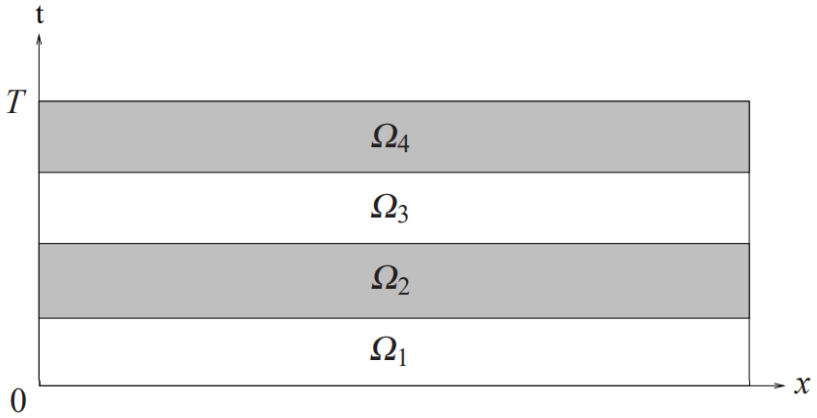
Az időpárhuzamosított módszerek történelme 60 évre tekint vissza, és manapság újra aktívan kutatott területe a matematikának. A klasszikus módszerek az időtengely mentén nem végeznek párhuzamosan számításokat, aminek az az oka, hogy az idő egyirányú. Egy differenciálegyenlet megoldása egy adott időpontban befolyásolja, hogy a későbbiekben miként fog viselkedni, így ebben az esetben nem magától értetődő a párhuzamosítás megvalósítása.
Tegyük fel, hogy egy (1.2) egyenlettel felírt kezdetiérték-problémát szeretnénk numerikusan megoldani egy egylépéses módszerrel. Ha a probléma megköveteli, hogy a megoldásokat nagy pontossággal közelítsük, akkor kis lépésközzel kell dolgoznunk, sok osztópontot kell felvennünk. Ahhoz, hogy egy időpontban meg tudjuk határozni a közelítést, szükségünk van a numerikus megoldás előző időpontban felvett értékére. Így egy szekvenciális algoritmushoz jutunk, amely az osztópontok számával megegyező számú lépést hajt végre egymás után. A szuperszámítógépek elterjedése okán a futási időt tekintve előnyös olyan módszert használnunk, amely különböző időpontokban végez párhuzamosan számításokat.

A fejezet első felében, követve Martin J. Gander összefoglaló munkáját [10], egy részletekbe nem belemenő, átfogó képet adunk az időpárhuzamosított módszerekről, melyek négy nagy csoportba sorolhatóak aszerint, hogy milyen számításokat végeznek párhuzamosan. A fejezet végén külön foglalkozunk a Parareal módszerrel, mert a következő fejezetben az MGRIT algoritmust a Parareal kiterjesztéseként vezetjük majd be.

2.1. Időpárhuzamosított módszerek

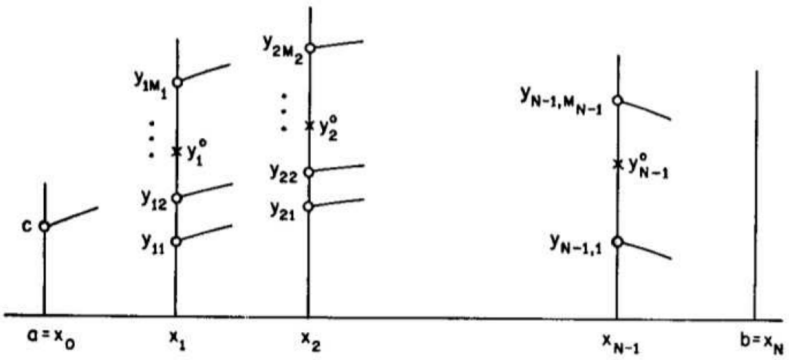
Belövéses módszerek: Ezen módszerek a tér-idő tartományt a 2.1. ábrán látható módon osztják fel, majd a részintervallumok kezdőpontjaiban megpróbálják eltalálni a pontos megoldást. Ezután a részintervallumokon értelmezett megoldásokból egy iteráció segítségével állítják elő

a megoldást a teljes $[0, T]$ intervallumon.



2.1. ábra. A tér-idő tartomány felosztása belövéses módszereknél.

A legelső idesorolható módszer Nievergelt nevéhez fűződik 1964-ből [21]. Az algoritmus alapötlete nagyon egyszerű. A 2.2. ábrán látható módon vegyünk fel véletlenszerűen több kezdőértéket a részintervallumok kezdőpontjaiban, majd párhuzamosan számoljuk ki egy pontos módszerrel a megoldásokat ezeken a részintervallumokon. Végül illesszük össze a megoldáso-



2.2. ábra. Nievergelt módszere [10].

kat szekvenciálisan a következőféleképpen:

- ◇ Tegyük fel, hogy az $[x_0, x_i]$ intervallumon tudjuk a megoldást és szeretnénk kiterjeszteni az $[x_0, x_{i+1}]$ intervallumra.
- ◇ Válasszuk az $[x_i, x_{i+1}]$ intervallumon értelmezett megoldások közül azokat, melyek kezdőpontjai közé esik az eddigi megoldás végpontja majd vegyük ezek lineáris kombinációját úgy, hogy folytonos megoldást kapjunk.

Ekkor az összmunkát tekintve sokkal többet kell dolgoznunk, viszont ez eloszlik sok processzor között, és a szekvenciális összeillesztés gyorsan megtehető. Ez nem egy egzakt algoritmus olyan

értelemben, hogy a részintervallumokon értelmezett megoldások kezdőpontjait az algoritmus elején véletlenül választjuk.

Egy másik ötletet találhatunk Bellen és Zennaro 1989-es cikkében, akik a teljes $[0, T]$ intervallum diszkretizációja után egy iteratív algoritmust vezettek be a megoldás megtalálására. Egy fixpont problémára vezették vissza a feladatot, melyre a Steffensen módszert alkalmazták [1]. Négy évvel később Chartier és Philippe kapcsolta össze a két ötletet [2]. Osszuk fel a $[0, T]$ intervallumot N db részintervallumra a $t_0, t_1, \ldots, t_N$ pontokkal, és jelölje rendre $U_0, U_1, \ldots, U_N$ a megoldás értékét a megfelelő helyen. Az U_{i+1} értéket úgy akarjuk meghatározni, hogy folytonos trajektóriát kapjunk, azaz az U_i kezdőpontú $[t_i, t_{i+1}]$ részintervallumon értelmezett megoldás végpontja legyen. Jelölje $u_i(U_i, t_{i+1})$ a $[t_i, t_{i+1}]$ intervallumon értelmezett pontos megoldás értékét a t_{i+1} helyen az U_i kezdőérték mellett. Ekkor az

$$U_{i+1} - u_i(U_i, t_{i+1}) = 0, \quad i = 0,1,\ldots N-1$$

nemlineáris egyenletrendszert kapjuk az U_i értékekre. Alkalmazzuk a Newton módszert az egyenletrendszerre, és jelöljük U^k-val a k-adik iterációban kapott közelítést. Ekkor a következő iteratív sémát kapjuk:

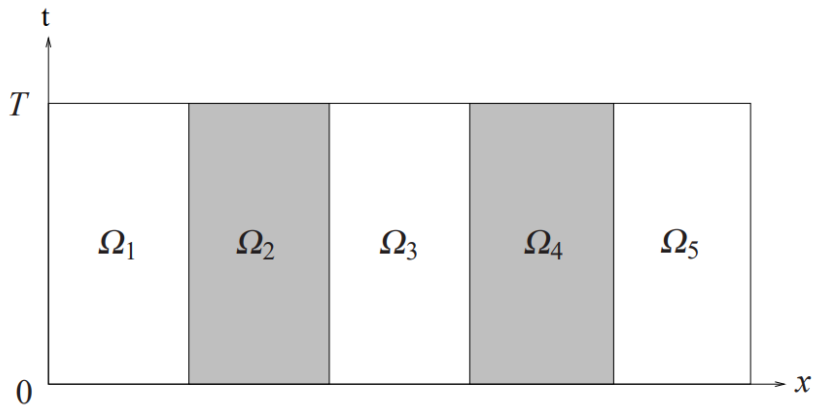
$$\begin{aligned} U_0^{k+1} &= u_0 \\ U_{i+1}^{k+1} &= u_i(U_i^k, t_{i+1}) + \frac{\partial u_i}{\partial U_i}(U_i^k, t_{i+1})(U_i^{k+1} - U_i^k), \quad i = 0,1,\ldots,N-1. \end{aligned} \tag{2.1}$$

A gyakorlatban a $\frac{\partial u_i(U_i^k, t_{i+1})}{\partial U_i}$ derivált értéket különböző differenciahányadosokkal lehet közelíteni. A $k+1$-edik iteráció előtt ismerjük az U^k vektor értékeit, így a részintervallumokon értelmezett megoldásokat párhuzamosan tudjuk kiszámolni. Végül megjegyezzük, hogy a később tárgyalt Parareal algoritmus is tekinthető belövéses módszernek, amely egy nagy lépésközű egylépéses módszer segítségével közelíti a hányadost.

Tartomány felosztás tér-időben: Most fordítva osztjuk fel a tér-idő tartományt (lásd 2.3. ábra.). A megfelelő tér-idő résztartományon kapott megoldások most a teljes $[0, T]$ intervallumon lesznek értelmezve. Ezután itt is egy iterációt hajtunk végre, amelynek következtében egyre jobb és jobb közelítését kapjuk a megoldásnak. Az iteráció során a különböző tér-idő résztartományokon párhuzamosan végzünk számításokat mielőtt a pontos határfeltételeket tudnánk. Ilyen értelemben ezen módszerek is az időpárhuzamosított módszerek közé sorolhatóak.

A leghíresebb ide tartozó módszer az ún. hullámforma relaxáció, amit Lelarasmee, Ruehli és Sangiovanni-Vincentelli vezetett be 1982-ben [17]. Vegyünk egy (1.2) egyenlettel felírt kezdetiérték-problémát. Jelölje u_i az i-edik komponenst, és közelítsük ezeket a következő iteratív módszerrel: legyen u_i^{k+1} megoldása a

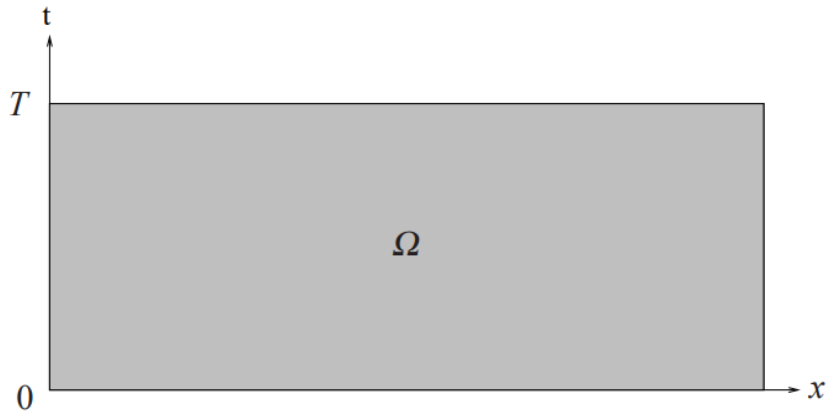
$$\partial_t u_i^{k+1} = f_i(u_1^k \ldots, u_{i-1}^k, u_i^{k+1}, u_{i+1}^k, \ldots, u_N^k), \quad i = 1,2,\ldots,N$$



2.3. ábra. A tér-idő tartomány felosztása a térbeli tengely mentén.

egyenletnek. Ekkor az iteráció minden lépésében N darab egymástól független, egydimenziós kezdetiérték-problémát kapunk, melyek megoldásait meghatározhatjuk párhuzamosan. A szerzők először arra használták a módszert, hogy meghatározzák egy oszcillátor különböző pontjaiban mért feszültségeinek az időbeli változását. A kapott grafikon hullám alakú volt, innen ered a módszer neve. Később parciális differenciálegyenletekre is kiterjesztették ezt a módszert ([6], [7]).

Tér-idő többrácsos módszerek: A belövéses módszereknél több processzort használhatunk az időtengely mentén, míg a tartomány felosztás tér-időben típusú módszerek a tér irányában párhuzamosítanak. Ezzel szemben a tér-idő többrácsos módszerek az egész tér-idő tartományon dolgoznak, de a komponenseik gyakran párhuzamosan számolhatóak (lásd 2.4. ábra).



2.4. ábra. A tér-idő tartomány felosztása többrácsos módszereknél.

Mikor egy időfüggő parciális differenciálegyenletet oldunk meg numerikusan, az időtengely mentén is lehetőség nyílik párhuzamosításra a térbeli irányok mellett. Amint láttuk, az MG módszerek hatékonyan alkalmazhatóak klasszikus térbeli elliptikus problémák esetén, így természetes kérdés, hogy kiterjeszthetőek-e az idő dimenzióra is. Azon többrácsos módszereket,

melyek az idő tengelyt is használják tér-idő MG módszereknek nevezzük.

Vegyünk egy klasszikus parabolikus problémát, például a diffúziós feladatot a szokásos kezdetiérték feltétellel és Dirichlet peremfeltétellel, majd diszkretizáljuk a véges differencia módszerét használva. Ha az egyenlet jobb oldala lineáris, akkor egy lineáris egyenletrendszerhez jutunk. Tegyük fel, hogy az 1.3. fejezetben tárgyalt többrácsos módszerrel akarjuk megoldani az egyenletrendszert a térbeli problémák esetén működő részletek szerint. Ekkor egy nem hatékony algoritmust kapunk, aminek az az oka, hogy a klasszikus iteratív eljárások nem simítják megfelelően az idő irányban a hiba nagyfrekvenciás komponenseit [14]. Az elmúlt 30 évben több cikk jelent meg, melyben speciális simító iterációk használatával, különféle ritkító stratégiák alkalmazásával és megfelelő leszűkítő, illetve prolongáló operátorok választásával jól teljesítő tér-idő módszereket vezettek be ([12], [19], [11]). Az MGRIT algoritmus is ezen módszerek közé sorolható, amellyel majd később részletesen foglalkozunk.

Direkt módszerek: Az előbbi három csoportba sorolható algoritmusok mind iteratívak. Ezzel szemben azon módszerek említhetők meg itt, melyek egyből a végleges megoldást szolgáltatják számunkra. Sokféle módszer tartozik ebbe a csoportba, melyek teljesen különböző ötleteket használnak. Az első párhuzamos direkt módszer Miranker és Liniger nevéhez fűződik, akik az eredetileg szekvenciális predictor-korrektor sémát alakították át úgy, hogy több értéket lehessen egyszerre számolni [20]. Talán érdemes még kiemelni Worley 1991-es cikkét [23], amelyben a masszívan párhuzamosítható cycle reduction módszert vezette be. Ezt az algoritmust ötvözve a többrácsos hullámforma relaxációs módszerrel egy lépésszám tekintetében optimális algoritmust kapunk [15]. Végül megemlítjük Güttel 2012-es ötletét, aki lineáris közönséges differenciálegyenletek megoldására adott egy nagyon hatékony algoritmust kihasználva, hogy gyorsan tudjuk jól közelíteni egy mátrix exponenciális függvényét [9].

2.2. Parareal algoritmus

Az előbb említett sok módszer közül kiemeljük a Parareal algoritmust, amit Lions, Maday és Turicini vezette be 2001-ben [18]. A módszer neve azt jelzi, hogy a párhuzamosított algoritmus jól használható olyan problémák esetén, melyek megoldását nem tudnánk meghatározni valós időben egy processzor használatával (parareal $\sim$ "parallel in real time"). Az algoritmusra, amely több új módszer alapjául szolgál, tekinthetünk egy időtengelyre alkalmazott kétrácsos módszerként is, amely több rácsra történő kiterjesztésével az MGRIT algoritmushoz jutunk.

Legyen F egy tetszőleges numerikus módszer, melyet egy (1.1) alakú kezdetiérték-feladatra alkalmazunk. Ekkor jelölje $F(u_0, t_0, t_1)$ az F módszer által adott megoldás értékét a t_1 helyen, ahol t_0 a kezdőpont és u_0 a kezdetiérték.

Egy (1.2) alakú kezdetiérték-feladat numerikus megoldásához először osszuk fel a $[0, T]$ intervallumot ekvidisztánsan a $T_0, \ldots, T_N$ pontokkal részintervallumokra, melyek hossza $\Delta T = \frac{T}{N}$. A Parareal algoritmus két numerikus módszert használ. A részintervallumokon belül egy pontos F, míg a teljes $[0, T]$ intervallumon egy durva G módszert. Ha a pontos módszert használjuk az egész $[0, T]$ intervallumon, akkor túl sokáig tart a megoldás meghatározása, míg ha a durva módszert alkalmazzuk itt, akkor rossz közelítést kapunk. Az algoritmus alapötlete az, hogy a $[0, T]$ intervallumon értelmezett megoldást iteratív módon közelítjük, miközben a pontos módszert párhuzamosan futtatjuk a $[T_i, T_{i+1}]$ tartományokon ($i = 0, \ldots, N-1$).

A megoldás egy kezdeti U^0 közelítését kaphatjuk a $T_0, \ldots, T_N$ pontokban, ha a durva módszert használjuk a $[0, T]$ intervallumon. Ezután U^k ismeretében az U^{k+1} értékeket az

$$U_{i+1}^{k+1} = F(U_i^k, T_i, T_{i+1}) + G(U_i^{k+1}, T_i, T_{i+1}) - G(U_i^k, T_i, T_{i+1}), \quad i = 0, \ldots N \tag{2.2}$$

képlettel számoljuk ki. A fenti eljárást az motiválja, hogy az algoritmus besorolható a belövéses módszerek csoportjába, ahol a (2.1) egyenletben szereplő pontos megoldást a finom módszer, míg a Jacobi deriváltat tartalmazó második tagot a durva módszer különböző kezdőpntokból indított megoldásainak különbsége adja, mégpedig

$$G(U_i^{k+1}, T_i, T_{i+1}) - G(U_i^k, T_i, T_{i+1}) \approx \frac{\partial u_i}{\partial U_i}(U_i^k, t_{i+1})(U_i^{k+1} - U_i^k),$$

ahol $u_i(U_i^k, t_{i+1})$ a pontos megoldás értéke az i-edik részintervallum végén az U_i^k kezdőérték mellett. Az iteráció végén az U^k értékek egy megoldást adnak a $T_0, \ldots, T_N$ pontokban. Ezekből úgy kapjuk meg a megoldást a teljes $[0, T]$ intervallumon, hogy minden részintervallumban futtatjuk az F finom módszert az U^k vektor által adott kezdőpontból.

A $k+1$-edik iteráció előtt ismerjük az U_i^k értékeket, ezért az $F(U_i^k, T_i, T_{i+1})$ és a $G(U_i^k, T_i, T_{i+1})$ kifejezéseket párhuzamosan tudjuk kiszámolni. A durva módszerrel minden iterációban továbbra is szekvenciálisan kell számolni, de feltettük, hogy ez gyorsan megtehető. Érdemes még megjegyezni, hogy $k \to \infty$ esetén az U^k vektor a finom módszer által adott megoldáshoz tart, azaz $U_i^k \to F(u_0, T_0, T_i)$. Pontosabban a k-adik iteráció után minden $i \leq k$ indexre U_i^k azt az értéket veszi fel, amit a pontos numerikus módszer ad, ha a teljes $[0, T]$ intervallumon használjuk. Az algoritmus tehát N iteráció után mindenképpen véget ér. Gyorsítás akkor várható a $[0, T]$ intervallumon alkalmazott finom módszerhez képest, ha kevés iteráció szükséges a leállási feltétel eléréséhez. Egy szokásos feltétel, hogy akkor áll le az algoritmus, ha elért egy előre rögzített maximális iteráció számot, vagy egy meghatározott értéknél kevesebbet változott az aktuális közelítés, vagy elég közel vagyunk a finom módszer által adott megoldáshoz, vagy.

Végül megemlítünk egy tételt, amely az eljárás használhatóságát biztosítja megfelelő módszerek választása esetén. Tegyük fel, hogy az F finom módszer a pontos megoldást adja, míg a G durva módszer p-ed rendben konzisztens és Lipschitz folytonos az első változójában. Ekkor

a Parareal algoritmus szuperlineárisan konvergál a pontos megoldáshoz a $[0, T]$ intervallumon, azaz

$$\|\bar{u} - U^{k+1}\| \leq C^k \|\bar{u} - U^k\|,$$

ahol $\bar{u}$ jelöli a pontos megoldás megszorítását a $0 = T_0 < T_1 < \ldots < T_N = T$ időpontokban, és $\lim_{k\to\infty} C^k = 0$. A tétel egy részletesebb formája és a bizonyítása megtalálható a [8] cikkben.

3. fejezet

Időbeli többrácsos redukciós módszer

Az MGRIT algoritmus egy időpárhuzamosított módszer, melynek célja, hogy a klasszikus egylépéses módszerekhez képest gyorsabb futási idővel rendelkezzen megfelelő környezet esetén. A módszer tekinthető a Parareal algoritmus kiterjesztésének, így mi is ezen az úton fogjuk bevezetni az MG módszereknél látott eszközök segítségével. A fejezet első felében feltesszük, hogy a feladatunk lineáris, majd később kiterjesztjük nemlineáris feladatokra is a teljes approximációs séma (Full Approximation Scheme, a továbbiakban FAS) segítségével.

A közönséges differenciálegyenletek mellett az MGRIT algoritmus hatékonyan használható időfüggő parciális differenciálegyenletek, mint például parabolikus egyenletek megoldásainak a közelítésére is. Az egyik ötlet alapja az egyenesek módszerének (method of lines, a későbbiekben MOL) alkalmazása, ahol a probléma egy közönséges differenciálegyenlet rendszerre vezethető vissza a térbeli irányok diszkretizálásával. Mint látni fogjuk, a nemlineáris feladatok esetén az algoritmus hatékony működéséhez szükségünk lesz térbeli ritkításra is, így parciális differenciálegyenletek esetén előnyösebb lesz a módszerre ún. tér-idő MG módszerként tekinteni, amely az időtengelyt speciálisan kezeli a térbeli irányokhoz képest.

3.1. A Parareal algoritmus, mint kétrácsos módszer

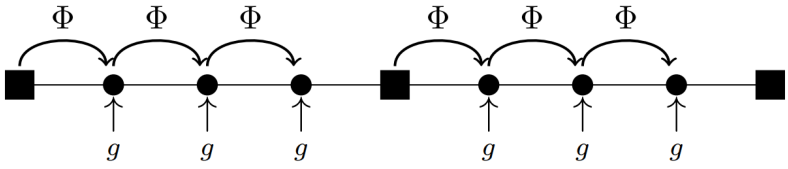
Tegyük fel, hogy a Parareal algoritmusnál definiált finom F és durva G módszer is egy-egy egylépéses módszer, és az (1.2) alakú kezdetiérték-feladat jobb oldala lineáris. A G módszert a Parareal algoritmusnál bevezetett $T_0, \ldots, T_N$ ritka, ΔT lépésközű felosztáson alkalmazzuk, míg az F módszert a $t_0, \ldots, t_{Nm} = t_n$ finom, $\Delta t = \frac{\Delta T}{m}$ lépésközű felosztáson. Vagyis minden részintervallumot további m darab részre osztunk. A ritka felosztáson levő pontokra a későbbiekben gyakran fogunk hivatkozni durva pontokként, míg csak a finom felosztáson található pontokat finom pontoknak nevezzük.

Írjuk fel az egylépéses módszereket az (1.3) egyenletben látható alakban. A G módszer a ritka rácson van értelmezve, míg a finom F módszer értelmezhető a finom és a durva rácson is. A finom rácsra alkalmazott F módszert karakterizálják a $\Phi(F)_j$ mátrixok és a $g(F)_j$ vektorok, a ritka rácson értelmezett F egylépéses módszert a $\Phi_\Delta(F)_i$ mátrixok és a $g_\Delta(F)_i$ vektorok, míg a G módszert a $\Phi_\Delta(G)_i$ mátrixok és a $g_\Delta(G)_i$ vektorok ($i = 1, \ldots, N,\ j = 1, \ldots, n$). Az egyszerűbb követhetőség kedvéért nem tesszük ki a Φ mátrixok után az indexet, mivel például a $\Phi(F)_i$ mátrixot mindig az u_{i-1} vektorral szorozzuk, valamint a legtöbb egylépéses módszer időfüggetlen, azaz mindig ugyanazt a mátrixot használja.

Az előbb bevezetett jelölésekkel megkonstruálhatóak az (1.4) egyenletben szereplő finom és durva módszerhez tartozó mátrixok. Legyen $Au = g$ a finom rácsra alkalmazott F egylépéses módszer egyenletrendszere, $A_\Delta u_\Delta = g_\Delta$ az F egylépéses módszer ritka rácshoz tartozó egyenletrendszere, míg $B_\Delta u_\Delta = g'_\Delta$ a ritka rácsra alkalmazott G egylépéses módszerhez tartozó egyenletrendszer. Ekkor a (2.2) Parareal korrekciós séma az

$$u_\Delta^{k+1} = u_\Delta^k + B_\Delta^{-1}(g_\Delta - A_\Delta u_\Delta^k) \tag{3.1}$$

alakot ölti. Vegyük a 2. algoritmusban leírt kétrácsos MG módszert, és alkalmazzuk az (1.2) kezdetiérték-feladatra. Az elősimítást az ún. F-relaxációval végezzük, ahol minden durva pontok által meghatározott részintervallumban szekvenciálisan lépünk az F módszerrel a finom pontokon (lásd 3.1. ábra).



3.1. ábra. Az F-relaxáció [4].

Képlettel ez a következőt jelenti:

$$u_{im} = u_{im}, \quad \text{ha } i = 0, \ldots, N,$$
$$u_{im+l} = \Phi(F)_{im+l}(u_{im+l-1}) + g(F)_{im+l}, \quad \text{ha } i = 0, \ldots, N-1,\ l = 1, \ldots, m-1$$

Ezt párhuzamosan tudjuk végrehajtani a résztartományokon, így az F-relaxáció $m-1$ lépés alatt elvégezhető elegendő processzor esetén. Ekkor a 2. algoritmus a megfelelő prolongáló és leszűkítő leképezések választásával épp a Parareal algoritmust adja, amit a 3.1. Tételben mondunk ki. A bizonyítás kicsit technikai, sok triviális számolást igényel. Az egyetlen ötlet a (3.7) egyenlet felírásához kell.

2. Algoritmus Parareal kétrácsos MG módszerként

1: Az $Au = g$ rendszer relaxálása F-relaxációval
2: $r_\Delta = R_I(g - Au^k)$
3: Oldjuk meg a $B_\Delta v = r_\Delta$ rendszert
4: $u^{k+1} = u^k + Pv$

3.1. Tétel Futtassuk a Parareal algoritmust az F és G egylépéses módszerekkel, válasszuk a 2. algoritmusban a leszűkítő, illetve prolongáló leképezéseket a természetes injekciónak, valamint annak transzponáltjának.

$$\begin{aligned} (R_I(u))_i &= u_{im}, \quad \text{ha } i = 0,1,\dots,N \\ (P(u_\Delta))_{im+l} &= \begin{cases} u_{\Delta_i}, & \text{ha } l = 0 \\ 0, & \text{ha } l = 1,2,\dots,m-1 \end{cases} \end{aligned} \tag{3.2}$$

Ha megegyező kezdővektorral indítjuk a Parareal módszert és a 2. algoritmusban leírt kétrácsos módszert, akkor azonos megoldást kapunk minden iteráció után. Azaz $U_i^k = u_{im}^k$ minden $i = 0,\dots,N$ esetén, ahol U^k jelöli a Parareal által adott megoldást a ritka felosztáson és u^k a kétrácsos algoritmus által adott megoldás a finom felosztáson.

Bizonyítás. A tételt kettős indukcióval bizonyítjuk. A külső indukció az iterációszámra vonatkozik, a belső pedig az indexre. Tudjuk, hogy $k = 0$-ra igaz az állítás, mivel kezdetben ugyanazon megoldásból indulunk ki. Tegyük fel, hogy valamilyen $k \in \mathbb{N}$ számra

$$U_i^k = u_{im}^k, \quad \forall i = 0,1,\dots,N.$$

A célunk annak belátása, hogy $U_i^{k+1} = u_{im}^{k+1}$ minden $i = 0,1,\dots,N$ indexre. Először vegyük észre, hogy a kezdőpontban nem változik a közelítés értéke az algoritmus során, azaz $U_0^{k+1} = u_0^{k+1}$. Legyen $i \in \{0,1,\dots,N-1\}$ olyan, hogy

$$U_i^{k+1} = u_{im}^{k+1}.$$

Ekkor elég bizonyítani, hogy $U_{i+1}^{k+1} = u_{(i+1)m}^{k+1}$, mert az indukció miatt ebből következik a tétel. Először írjuk át az U_{i+1}^{k+1} kifejezést a (2.2) egyenlet szerint, azaz

$$\begin{aligned} U_{i+1}^{k+1} &= F(U_i^k, T_j, T_{i+1}) + G(U_i^{k+1}, T_j, T_{i+1}) - G(U_i^k, T_i, T_{i+1}) = \\ &= \Phi_\Delta(F)U_i^k + g_\Delta(F)_{i+1} + \Phi_\Delta(G)U_i^{k+1} + g_\Delta(G)_{i+1} - \Phi_\Delta(G)U_i^k - g_\Delta(G)_{i+1} = \\ &= \Phi_\Delta(F)U_i^k + g_\Delta(F)_{i+1} + \Phi_\Delta(G)U_i^{k+1} - \Phi_\Delta(G)U_i^k. \end{aligned} \tag{3.3}$$

Ezután vizsgáljuk meg, hogy a 2. algoritmusban hogyan határozzuk meg az $u_{(i+1)m}^{k+1}$ értéket. Az algoritmus utolsó sora szerint

$$\begin{aligned} v_i &= u_{mi}^{k+1} - u_{mi}^k, \\ v_{i+1} &= u_{m(i+1)}^{k+1} - u_{m(i+1)}^k. \end{aligned} \tag{3.4}$$

Írjuk ki az algoritmus harmadik lépésében az $i+1$-edik egyenletet, majd helyettesítsük be az előbb kapott értékeket. Ekkor

$$\begin{aligned} -\Phi_\Delta(G)v_i + v_{i+1} &= r_{\Delta,i+1}, \\ \Phi_\Delta(G)u_{mi}^k - \Phi_\Delta(G)u_{mi}^{k+1} + u_{m(i+1)}^{k+1} - u_{m(i+1)}^k &= r_{\Delta,i+1}. \end{aligned} \tag{3.5}$$

Nézzük az algoritmus második sorában szereplő egyenletrendszer $i+1$-edik egyenletét

$$r_{\Delta,i+1} = g(F)_{m(i+1)} - u_{m(i+1)}^k + \Phi(F)u_{m(i+1)-1}^k = (\Phi(F)u_{m(i+1)-1}^k + g(F)_{m(i+1)}) - u_{m(i+1)}^k. \tag{3.6}$$

Szavakkal leírva ez azt jelenti, hogy a ritka rácson értelmezett reziduális $i+1$-edik komponensét úgy kapjuk meg, hogy a finom módszerrel lépünk a $t_{m(i+1)-1}$ pontból, majd vesszük az így kapott eredmény és az aktuális $u_{m(i+1)}^k$ érték különbségét. Vegyük észre, hogy az algoritmus elején egy $F-$relaxációt hajtottunk végre. Ha a $t_{m(i+1)-1}$ pontból még egyet lépünk az F módszerrel, akkor épp végig érünk a $[T_i, T_{i+1}]$ intervallumon, azaz olyan mintha a ritka rácson léptünk volna egyet az F módszerrel. Így az előző (3.6) egyenlet tovább írható az

$$r_{\Delta,i+1} = \Phi_\Delta(F)u_{mi}^k + g_\Delta(F)_{i+1} - u_{m(i+1)}^k \tag{3.7}$$

alakra. Írjuk be a (3.5) egyenlet jobb oldalába a (3.7) egyenletben kapott kifejezést, majd fejezzük ki az $u_{m(i+1)}^{k+1}$ értéket. Ekkor

$$\begin{aligned} \Phi_\Delta(G)u_{mi}^k - \Phi_\Delta(G)u_{mi}^{k+1} + u_{m(i+1)}^{k+1} - u_{m(i+1)}^k &= \Phi_\Delta(F)u_{mi}^k + g_\Delta(F)_{i+1} - u_{m(i+1)}^k \\ u_{m(i+1)}^{k+1} &= \Phi_\Delta(F)u_{mi}^k + g_\Delta(F)_{i+1} + \Phi_\Delta(G)u_{mi}^{k+1} - \Phi_\Delta(G)u_{mi}^k \end{aligned} \tag{3.8}$$

Végül az indukciós feltevéseket használva a (3.3) és (3.8) egyenletekből a tétel állítását kapjuk:

$$\begin{aligned} u_{m(i+1)}^{k+1} &= \Phi_\Delta(F)u_{mi}^k + g_\Delta(F)_{i+1} + \Phi_\Delta(G)u_{mi}^{k+1} - \Phi_\Delta(G)u_{mi}^k = \\ &= \Phi_\Delta(F)U_i^k + g_\Delta(F)_{i+1} + \Phi_\Delta(G)U_i^{k+1} - \Phi_\Delta(G)U_i^k = U_{i+1}^{k+1}. \end{aligned}$$

■

3.2. MGRIT módszer

Az előbb bemutatott Parareal algoritmus gyengesége, hogy sokszor a ritka felosztás mérete összemérhető a finom felosztás méretével, így a 2. algoritmusban a harmadik lépés végrehajtása szintén sok időt vesz igénybe. Ezen segít, ha az MG módszereknél látott módon a harmadik lépésben rekurzívan meghívjuk az algoritmust egy még durvább felosztáson. Így lényegében el is jutottunk az MGRIT algoritmushoz, amely tekinthető az időtengelyre alkalmazott MG módszernek.

Vegyük az Ω_l ($l = 0,1,\ldots,L$) időbeli felosztásokat a Δt, $m\Delta t$, stb. lépésközzel egy m ritkító tényezővel, és minden felosztáson használjunk egy egylépéses módszert. Általában minden szinten ugyanazt a módszert használjuk. Jelölje $A_l u^{(l)} = g^{(l)}$ az l-edik szinthez tartozó egyenletrendszert, ahol az A_l mátrixot a Φ_i^l függvények határozzák meg az (1.4) egyenlet szerint. Ezekkel a jelölésekkel az MGRIT módszer V-ciklusos variánsát a 3. algoritmus írja le. A leszűkítő és prolongáló leképezéseket a Parareal algoritmusnál látott módon, a 3.1. Tételben leírtak szerint választjuk.

3. Algoritmus MGRIT(l)

1: **if** l a legdurvább szint **then**
2: Oldjuk meg az $A_L u^{(L)} = g^{(L)}$ rendszert
3: **else**
4: Az $A_l u^{(l)} = g^{(l)}$ rendszer relaxálása F- vagy FCF-relaxációval
5: $g^{(l+1)} = R_I(g^{(l)} - A_l u^{(l)})$
6: MGRIT(l+1)
7: $u^{(l)} = u^{(l)} + P u^{(l+1)}$
8: Az $A_l u^{(l)} = g^{(l)}$ rendszer relaxálása F-relaxációval

Ahogy az MG módszereknél említettük, több rács esetén az iteráció végén fontos szerepe van az utósimításnak is, amely most az elősimításhoz hasonlóan egy F-relaxációval valósítható meg. Két rács esetén nincs jelentősége az utósimításnak, mivel a következő iteráció elején elősimítás következik, és két egymás után alkalmazott F-relaxáció esetén a másodiknak nincs hatása. Viszont több rács esetében egy utósimítást nem mindig elősimítás követ.

A módszert a [4] cikkben vezették be, ahol a szerzők az eljárás leírása mellett ki is próbálták az algoritmust egy jól ismert feladaton. A céljuk az algoritmus paramétereinek optimális megválasztása volt, és arra keresték a választ, hogy megfelelő erőforrások birtokában mekkora gyorsítás érhető el a szekvenciális egylépéses módszerhez képest. Az elvégzett mérésekből adódó következtetések többnyire általános érvényűek, így összefoglaljuk a cikkben leírt eredményeket.

Mintafeladatnak a homogén Dirichlet-peremfeltételű hővezetési egyenletet választották:

$$\begin{aligned} \partial_t u(t,x) - \alpha \Delta u(t,x) = b(t,x), \quad & x \in \Omega = [0,\pi]^d, \quad t \in (0,T], \quad \kappa > 0 \\ u(x,0) = u_0(x), \quad & x \in \Omega \\ u(t,x) = 0, \quad & x \in \partial\Omega, \quad t \in [0,T]. \end{aligned} \tag{3.9}$$

A tesztek során a jobb oldali és a kezdeti függvények egyszerű szinuszos függvények voltak. A dimenziószámot $d = 2$-nek és a diffúziós együtthatót $\alpha = 1$-nek választották. A térbeli diszkretizálást a véges differencia módszerrel végezték a klasszikus másodrendű 5-pontos stencilt

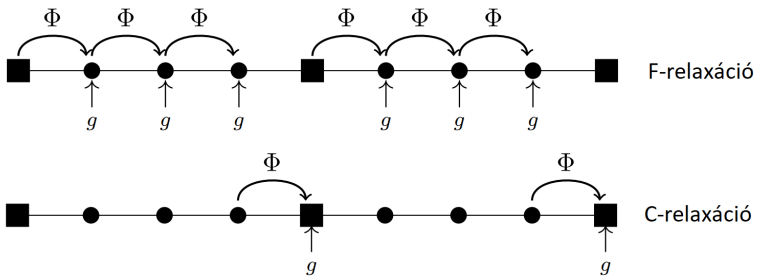
használva, míg időben az implicit Euler módszert alkalmazták. Így egy lineáris diszkretizálását kapták a mintafeladatnak.

Az MGRIT algoritmus implementációja elérhető az XBraid nevű publikus GitHub repozitóriumban [22]. A mintaprobléma megtalálható a példák között, így mi is tudtunk kísérleteket végezni az eredeti kóddal a saját környezetünkben. Ezen felül további feladatokon is kipróbáltuk az algoritmust. A mérések részletei és eredményei a 4. fejezetben találhatóak.

A diszkretizálás után egy háromdimenziós rácsot kapunk két térbeli és egy időtengellyel. A kód egyenletesen osztja fel a rácsot a processzorok között, azaz mindegyik processzor egy ugyanakkora méretű téglatesten dolgozik. Memória megtakarítás céljából minden szinten csak a durva pontokban kiszámolt megoldásokat tároljuk el. Ezekből könnyen, egy F-relaxáció segítségével párhuzamosan számítható ki a finom pontokban a megoldás.

Az elemzés során az első észrevétel az volt, hogy a Parareal algoritmus természetes kiterjesztése nem vezet hatékony algoritmushoz. A V-ciklusos módszer F-relaxáció használatával túl lassú konvergenciát eredményez. Nagyon sok iterációra van szükség a megfelelő pontosság eléréséhez. Ezért az F-relaxáció helyett úgynevezett FCF-relaxációt érdemes használni, ahol először egy F-, majd egy C- és végül ismét egy F-relaxációt hajtunk végre (lásd 3.2. ábra). Egy C-relaxációban a durva pontokban felvett értékeket frissítjük az F egylépéses módszerrel az előző finom pontból a következő képletben leírt módon.

$$u_{im} = \Phi(F)_{im}(u_{im-1}) + g(F)_{im}, \quad \text{ha } i = 1, \ldots, N$$



3.2. ábra. Az F- és C-relaxációk [4].

Ekkor egy V-ciklus végrehajtása több időt igényel a hosszabb relaxációs folyamat miatt, viszont sokkal kevesebb iterációra van szüksége az algoritmusnak ugyanazon pontosság eléréséhez. Azaz egy valamit valamiért (trade-off) helyzet alakul ki. A tesztek azt mutatják, hogy futási idő szempontjából FCF-relaxációt érdemes végrehajtani. Egy köztes megoldás, amely általában a legjobban teljesít, hogy a legfinomabb felosztáson F-, míg a többi szinten FCF-relaxációt használunk.

A módszer fontos része az implicit egylépéses módszerek esetének kezelése. Egy ilyen eljárás használatakor az (1.3) egyenletben definiált, az egylépéses módszert leíró Φ függvény alkalmazása egy térbeli lineáris egyenletrendszer megoldását jelenti. A gyors futási idő érdekében ezeket nem egzaktul, hanem az 1.3. alfejezetben látott térbeli MG módszerrel iteratívan oldjuk meg. Ekkor két kérdés merül fel: hogyan válasszunk jó kezdőértékeket, és milyen pontossággal adjuk meg a megoldást.

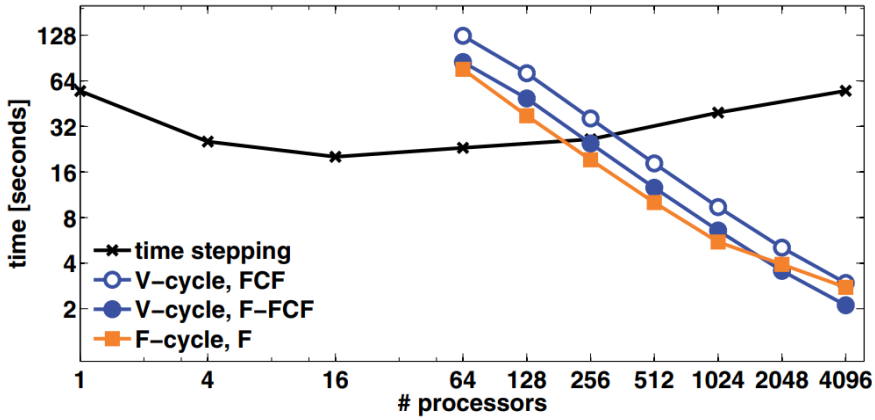
Az MGRIT módszer egy iteratív algoritmus, így egy kézenfekvő megoldás lenne a kezdeti közelítésre az előző iteráció során számolt érték. Viszont mint említettük, memória megtakarítás céljából minden szinten csak a durva rácson szereplő pontokban tároljuk a megoldást. Ezért az egylépéses módszereknél látott módon, az előző lépésben kiszámolt értéket tekintjük kezdeti közelítésnek. A finomabb szinteken, ahol a szomszédos osztópontok közel vannak egymáshoz, ez nem jelent problémát. Ám a durva szinteken a nagy lépésköz miatt a kezdő közelítés pontatlan lesz. Ha a ritkább rácsokon pontosan szeretnénk kiszámolni a megoldást, akkor a rossz kezdő közelítés miatt sok iterációra lenne szüksége a térbeli MG módszernek, amely dominálná az MGRIT futási idejét.

Általánosan kétféle stratégia van a megoldások pontosságának meghatározására. A hibán alapuló, amikor egy adott hiba elérése után áll meg az algoritmus, valamint az iterációszámon alapuló, amikor előre meghatározott számú iterációt hajt végre az algoritmus. A tesztek azt mutatják, hogy hatékony MGRIT algoritmust kapunk, ha ötvözzük a két technikát, a legfinomabb rácson hibán alapuló stratégiát, míg a ritkább rácsokon iterációszámon alapulót használva. Ekkor a durva szinteken pontatlan közelítéseket kapunk. Ez nem probléma, mert a nagy lépésköz miatt az egylépéses módszer már eleve pontatlan.

Az algoritmus futási idejét nagyobb mértékben befolyásolja még a ritkító tényező értéke. A standard időbeli ritkítás esetén minden rácson megfelezzük a pontok számát. Ekkor minden szinten egy fele akkora problémát kapunk, mint a felette lévőn. Ha növeljük a ritkító faktor értékét, akkor a feladat mérete gyorsabban csökken. Így egy iteráció gyorsabban végrehajtható, viszont kevésbé pontos megoldást eredményez, tehát több iterációra van szükség. Általánosságban elmondható, hogy a ritkító tényező növelésével az összfutási idő először csökken a kevesebb processzorok közötti kommunikáció miatt, majd növekszik az FCF-relaxációban szekvenciálisan számolt lépések növekvő száma miatt. A mintafeladat esetén egy jó stratégia, ha a legfinomabb felosztáson felezzük a probléma méretét, majd a többi szinten az m értéket 16-ra állítjuk.

A lineáris esetben hatékony algoritmust kapunk a mintafeladaton térbeli ritkítás nélkül is. A nemlineáris esetben látni fogjuk, hogy az optimális algoritmus használ térbeli ritkítást, mivel a nemlineáris egyenletek megoldására gyakran használt Newton-módszer kritikus pontja a megfelelően pontos kezdőközelítés megválasztása.

A paraméterek optimalizálása utáni mérések azt mutatják, hogy ha sok processzorral rendelkezünk, és figyelembe vesszük a fent leírtakat, akkor az MGRIT algoritmus hatékonyan használható a mintafeladat megoldásának a közelítésére. A 3.3. ábrán látható kísérletnél a tér-idő rács mérete $129^2 \times 16385$ pontból áll. A klasszikus egylépéses módszer esetében csak a térben párhuzamosítottunk, azaz az implicit Euler módszer egy lépésénél kapott lineáris egyenletrendszert egy térbeli MG módszerrel oldottuk meg az összes rendelkezésre álló processzorral. Az MGRIT algoritmus esetében $16 \times P_t$ processzor között osztottuk fel a tér-idő rácsot, ahol P_t jelöli az időbeli párhuzamosításhoz használt processzorok számát. Ekkor minden processzor egy $33^2 \times \left(\frac{16384}{P_t} + 1\right)$ darab ponttal rendelkező téglatesten végzett számításokat. Az ilyen típusú méréseket erősen skálázó (strong scaling) kísérleteknek nevezzük.



3.3. ábra. A futási idő a processzorszám tekintetében az egyes eljárásokra nézve [4].

A klasszikus egylépéses algoritmus futási idejét csak kis mértékben befolyásolja a használt processzorok száma. Az optimális futási idő körülbelül 20 másodperc, amely 16 processzor esetén tapasztalható. Ezzel szemben az MGRIT algoritmus nagyon jó skálázódást mutat. Ha megduplázzuk a processzorok számát, akkor a futási idő körülbelül a felére csökken. A várakozásoknak megfelelően kevés processzor esetén az egylépéses módszer gyorsabb, viszont ha legalább 256 processzor áll rendelkezésünkre, akkor előnyös az MGRIT módszert használni. Továbbá 4096 processzor esetén már jelentős, majdnem 10-szeres gyorsítást lehet elérni a legjobb, 16 processzort használó szekvenciális algoritmushoz képest.

Végül röviden vizsgáljuk meg, hogy a kétszintes módszer milyen gyorsan konvergál a megoldáshoz. A megoldás alatt most a finom rácsra alkalmazott egylépéses módszer által adott értékeket értjük. Ebben a szakaszban a [3] cikkben leírtak egy részét foglaljuk össze. Használjuk továbbra is a 3.1. alfejezet elején bevezetett jelöléseket, és tegyük fel, hogy a finom F és durva G módszer is időfüggetlen. A Parareal algoritmus a kétszintes F-relaxációt használó MGRIT variánsnak felel meg, ezért a (3.1) korrekciós séma segítségével megkaphatjuk, hogy

egy iteráció során hogyan változik az $e_\Delta^k = u - u_\Delta^k$ hiba a ritka hálón. Erre adódik, hogy

$$e_\Delta^{k+1} = (I - B_\Delta^{-1} A_\Delta) e_\Delta^k.$$

Az FCF-relaxáció esetén szintén könnyen felírható a megoldásra vonatkozó korrekciós séma, amelyből meghatározható a hiba vektorra vonatkozó

$$e_\Delta^{k+1} = (I - B_\Delta^{-1} A_\Delta)(I - A_\Delta) e_\Delta^k$$

összefüggés. Ha észrevesszük, hogy a B_Δ mátrix inverze

$$(B_\Delta^{-1})_{ij} = \begin{cases} 0, & \text{ha } i < j, \\ \Phi(G)^{i-j}, & \text{ha } i \geq j, \end{cases}$$

egy jól strukturált alsó háromszög mátrix, akkor a hibavektor komponensei kifejezhetőek a $\Phi(F)$ és $\Phi(G)$ mátrixok segítségével.

F-relaxáció esetén:

$$(e_\Delta^{k+1})_0 = 0$$
$$(e_\Delta^{k+1})_i = \sum_{q=0}^{i-1} \Phi(G)^{k-1-q} (\Phi(F)^m - \Phi(G))(e_\Delta^k)_q, \quad i = 1,2,\ldots,N.$$

FCF-relaxáció esetén:

$$(e_\Delta^{k+1})_0 = 0$$
$$(e_\Delta^{k+1})_1 = 0$$
$$(e_\Delta^{k+1})_i = \sum_{q=0}^{i-1} \Phi(G)^{k-2-q} (\Phi(F)^m - \Phi(G))\Phi(F)^m (e_\Delta^k)_q, \quad i = 2,3,\ldots,N.$$

A továbbiakban tegyük fel, hogy a $\Phi(F)$ és $\Phi(G)$ mátrixok diagonalizálhatóak, méghozzá ugyanazzal az unitér mátrixszal. Ez nem jelent megszorítást, mivel általában ugyanazt a módszert használjuk mind a két rácson. Jelölje λ_ω a $\Phi(F)$ és μ_ω a $\Phi(G)$ mátrix sajátértékeit, ahol $\omega = 1,2,\ldots,d$ és d a probléma térbeli mérete. Ekkor ezen értékek segítségével egy felső becslés adható a konvergencia sebességére.

3.2. Tétel Tegyük fel, hogy a két egylépéses módszer stabil, azaz $|\lambda_\omega| < 1$ és $|\mu_\omega| < 1$ minden $\omega = 1,2,\ldots,d$ esetén. Ekkor az alábbi rekurziós összefüggések adhatók a kétszintes módszer hibájára a $k+1$-edik iteráció után:

F-relaxáció esetén:

$$\|e_\Delta^{k+1}\|_2 \leq \max_\omega \left\{ |\lambda_\omega^m - \mu_\omega| \frac{1 - |\mu_\omega|^N}{1 - |\mu_\omega|} \right\} \|e_\Delta^k\|_2.$$

FCF-relaxáció esetén:

$$\|e_\Delta^{k+1}\|_2 \leq \max_\omega \left\{ |\lambda_\omega^m - \mu_\omega| \frac{1 - |\mu_\omega|^{N-1}}{1 - |\mu_\omega|} |\lambda_\omega|^m \right\} \|e_\Delta^k\|_2.$$

A bizonyítás azon alapul, hogy az e_Δ^k és e_Δ^{k+1} vektorokat felírjuk a közös sajátbázisban. A λ_ω és μ_ω sajátértékek függnek a feladattól és a diszkretizáció során használt módszerektől. Vizsgáljuk meg, hogy a mintafeladaton, a lineáris hővezetési egyenleten, miként alakul az értékük. Korábban említettük, hogyha térben a másodrendű centrális differenciasémát használjuk, akkor egy $u'(t) = Au(t)$ lineáris egyenlethez jutunk, ahol az A mátrix szimmetrikus. Ennek következtében az A mátrix diagonalizálható egy X unitér mátrixszal és a sajátértékei valósak, sőt belátható, hogy nempozitívak.

Ha időben mind a két szinten implicit Euler módszert használunk, akkor a $\Phi = (I - \Delta t A)^{-1}$ mátrixot kell vizsgálnunk. Minden $\Delta t > 0$ esetén a Φ mátrix is diagonalizálható az X unitér mátrixszal és a sajátértékei kifejezhetőek az A mátrix sajátértékeivel. Jelölje γ_ω az A mátrix sajátértékeit. Ekkor a Φ mátrix sajátértékei a $(1 - \Delta t\gamma_\omega)^{-1} \in (0,1]$ számok. Tegyük fel, hogy a ritkító faktor $m = 2$. Ekkor a 3.2. Tétel felhasználásával belátható, hogy

F-relaxáció esetén:

$$\|e_\Delta^{k+1}\|_2 \leq \max_\omega \frac{-\Delta t\gamma_\omega}{2(1+\gamma_\omega)^2}\|e_\Delta^k\|_2 \leq \frac{1}{8}\|e_\Delta^k\|_2. \tag{3.10}$$

FCF-relaxáció esetén:

$$\|e_\Delta^{k+1}\|_2 \leq \max_\omega \frac{-\Delta t\gamma_\omega}{2(1+\gamma_\omega)^4}\|e_\Delta^k\|_2 \leq \frac{27}{512}\|e_\Delta^k\|_2 \approx 0.0527\|e_\Delta^k\|_2. \tag{3.11}$$

A konvergencia hányados azt mondja meg, hogy a hiba hányszorosára változik egy iteráció alatt. Ekkor a tétel a kétszintes algoritmus konvergencia hányadosára ad egy felső becslést F- és FCF-relaxáció esetén is. Érdemes összevetni az F- és az FCF-relaxáció esetén kapott két számot. Az FCF-relaxáció használatával az algoritmus több mint kétszer olyan gyorsan konvergál, aminek következtében sokkal kevesebb iterációra van szüksége egy előre adott hiba eléréséhez. A többrácsos esetben nehezebb jó becslést adni a konvergencia faktor értékére [13]. Ekkor még nagyobb a különbség a két relaxáció között, amely következtében az F-relaxációt használó variáns futási ideje nagyon megnő a sok iteráció miatt.

3.3. MGRIT módszer kiterjesztése nemlineáris feladatokra

Az eddigiekben feltettük, hogy az (1.2) alakú kezdetiérték-probléma jobb oldala lineáris. Ekkor az egylépéses módszert leíró Φ függvény lineáris, amely reprezentálható egy mátrixszal. Ennek következtében a diszkretizálás után egy lineáris egyenletrendszerre vezettük vissza a problémát, amit egy, az időtengelyre alkalmazott MG módszerrel oldottunk meg. A Parareal algoritmus hatékonyan alkalmazható nemlineáris feladatok esetén is, így természetesen vetődik fel a kérdés, hogy az MGRIT módszer kiterjeszthető-e ilyen problémákra. A válasz az, hogy igen, a teljes approximációs séma (Full Approximation Scheme, rövidítve FAS) segítségével. A név arra

utal, hogy a 3. algoritmus 6. lépésében a durva rácson nem a hibát számoljuk ki, hanem a teljes approximációt, ahogy az a 4. algoritmus 10. lépésében látható.

4. Algoritmus MGRIT-FAS(l)

1: **if** l a legdurvább szint **then**
2: Oldjuk meg az $A_L(u^{(L)}) = g^{(L)}$ rendszert
3: **else**
4: Az $A_l(u^{(l)}) = g^{(l)}$ rendszer relaxálása FCF-relaxációval
5: Reziduális számítása és szűkítése injekcióval: $r^{(l+1)} = R_I(g^{(l)} - A_l(u^{(l)}))$
6: A megoldás leszűkítése: $u^l_\Delta = R_I(u^{(l)})$
7: **if** Térbeli ritkítás **then**
8: $u^{(l)}_\Delta = R_x(u^{(l)}_\Delta)$ és $r^{(l+1)} = R_x(r^{(l+1)})$
9: $g^{(l+1)} = A_{l+1}(u^{(l)}_\Delta) + r^{(l+1)}$
10: Megoldás a következő szinten: MGRIT-FAS($l + 1$).
11: $e^{(l)} = u^{(l+1)} - u^{(l)}_\Delta$
12: **if** Térbeli ritkítás **then**
13: $e^{(l)} = P_x(e^{(l)})$
14: A hiba prolongálása: $u^{(l)} = u^{(l)} + Pe^{(l)}$
15: Az $A_l(u^{(l)}) = g^{(l)}$ rendszer relaxálása F-relaxációval

A pszeudokódban a lineáris algoritmushoz képest kiemeltük a térbeli ritkítás lehetőségét. Ennek oka az, hogy nemlineáris esetben az algoritmus hatékonyságát illetően fontos szerepe van az időbeli mellett a térbeli ritkításnak is. Az időbeli leszűkítő illetve prolongáló leképezéseket továbbra is a 3.1. Tételben szereplő módon választjuk meg. A térbeli leszűkítést az R_x restriktáló, míg az interpolációt a P_x prolongáló leképezés valósítja meg. A megfelelő leképezések választása még ma is aktív kutatási terület.

Szeretnénk az előző alfejezetben látottakhoz hasonló eredményeket kapni nemlineáris problémák esetén is. Ehhez a módszer finomhangolására van szükség. A lineáris esethez hasonlóan az [5] cikkben, amelyben a nemlineáris algoritmus bevezetésre került, a szerzők egy feladaton keresztül mutatták be, hogy milyen ötletek segítségével tehető hatékonnyá az MGRIT algoritmus.

Mintafeladatnak a $p-$Laplace parabolikus feladatot választották Neumann-peremfeltétellel az alábbi

$$\begin{aligned} \partial_t u(x,t) - \nabla \cdot (|\nabla u(x,t)|^{p-2}\nabla u(x,t)) &= b(x,t), \quad x \in \Omega,\ t \in [0,T] \\ |\nabla u(x,t)|^{p-2}\nabla u(x,t) \cdot n &= g(x,t), \quad x \in \partial\Omega,\ t \in [0,T] \\ u(x,0) &= u_0(x), \quad x \in \Omega \end{aligned} \tag{3.12}$$

alakban, ahol a tesztek során a $p = 4$ esetet vizsgálták. Ekkor az egyenlet a talajerózió modelljét írja le, valamint előfordul a képfeldolgozás és a gépi tanulás témaköreiben is. A kísérleteknél a dimenziót $d = 2$-nek, a térbeli értelmezési tartományt $\Omega = [0,2] \times [0,2]$-nek választották, és $T = 4$ másodpercig vizsgálták a folyamatot. A $b(x,t)$ jobb oldalt és a határfeltételeket úgy állították be, hogy a megoldás az

$$u(x,y,t) = \sin(\kappa x)\sin(\kappa y)\sin(\tau t)$$

alakot öltse, ahol $\kappa = \pi$ és $\tau = \frac{13}{6}\pi$. Érdemes még megjegyezni, hogy $p = 2$ esetben a lineáris diffúziós egyenletet kapjuk vissza. A térbeli diszkretizálást az MFEM végeselem könyvtár használatával valósították meg. Időben most is az implicit Euler módszert alkalmazták.

Tegyük fel, hogy az l-edik szinten vagyunk, és az implicit Euler módszerrel szeretnénk egyet lépni az i-edik pontból. Ekkor az $u_{i+1} = \Phi^l(u_i) + g_{i+1}$ térbeli egyenlet kiszámolása az

$$u_{i+1} = u_i + \Delta t f(t_{i+1}, u_{i+1}) \tag{3.13}$$

nemlineáris egyenlet megoldásával egyenértékű, ahol az f függvény a térbeli ritkítás után kapott (1.2) alakú kezdetiérték-feladat jobb oldala. A gyakorlatban a Newton-módszert használjuk a megoldás közelítésére, amely minden lépésében egy térbeli lineáris egyenletrendszert kell megoldanunk. Ezt megtehetjük pontosan, vagy idő megtakarítás céljából a lineáris esetben látott módon, egy térbeli MG módszer használatával. Ekkor az implicit Euler módszer egy lépésénél most nem csak egy, hanem több térbeli lineáris egyenletrendszert kell megoldanunk nagy pontossággal.

Az időbeli ritkítás miatt a ritka rácsokon nagy a távolság két szomszédos időpont között, így az előző időpontban kiszámolt érték messze van a nemlineáris egyenletrendszer megoldásától. Ekkor a nem annyira pontos kezdő közelítés miatt több iterációra van szükség a Newton-módszer során, hogy megfelelő pontosságú megoldást kapjunk. Ennek következtében a ritka rácsokon a futási idő megemelkedik, és a módszer használhatatlanná válik. A két legfontosabb ötlet a Newton-módszer futási idejének csökkentésére a javított kezdőértékek használata és a térbeli ritkítás.

A Newton-módszer egyik sarkalatos pontja a megfelelő kezdőpont választása. A ritka rácsokon a nagy lépésköz miatt az előző időpontban kiszámolt érték nem elég pontos. Ezen segít a következő triviális ötlet: minden szint minden pontjában tároljuk el a legutóbbi, ott futtatott Newton-módszer által adott eredményt, és használjuk ezt kezdőértéknek a következő ciklusban. Az egyedüli hátránya ennek, hogy plusz egy vektort el kell tárolni minden ritka rács minden pontjában. A memóriahasználat redukálása érdekében egy köztes megoldás lehet, hogy csak minden szint durva rácspontjaiban tesszük meg ezt. Ekkor csak az F-relaxáció során kell a megelőző pontban szereplő értéket használni.

A térbeli ritkításnak köszönhetően a (3.13) nemlineáris egyenletrendszer mérete csökken, így a kisebb méretű problémát gyorsabban tudjuk megoldani. Az ötlet további előnye, hogy az időbeli és térbeli lépésköz négyzetének a hányadosa beállítható. A (3.13) általános egyenlet a mintafeladat esetén az

$$\left(I - \frac{\Delta t}{(\Delta x)^2} G\right)(u_{i+1}) = h(u_i)$$

alakot ölti, ahol G a nemlineáris diffúzió operátora, és Δx a térbeli lépésközt jelöli. A h függvényt a mintafeladat jobb oldala határozza meg. Térbeli ritkítás nélkül a $\frac{\Delta t}{(\Delta x)^2}$ hányados értéke szintről szintre növekszik, így a nemlineáris leképezés egyre távolabb kerül az identikus leképezéstől. Ennek következtében nehezebb megoldani az egyenletrendszert.

A térbeli szűkítés miatt a kapott közelítések pontatlanabbá válnak, így több MGRIT V-ciklusra van szükség a megfelelő pontosság eléréséhez. Ezért a megfelelő ritkítási stratégia megtalálása fontos része az algoritmusnak. A mintafeladat esetén érdemes kipróbálni az ún. késleltetett térbeli ritkítást, amikor a legfinomabb szinteken ugyanazt a térbeli rácsot használjuk, és csak később kezdjük meg a ritkítást.

Ezeken a módszereken túl még beszélhetünk olyan eszközökről, melyek nem olyan nagymértékben befolyásolják a futási időt. Ha kezdetben nincs előzetes ismeretünk a megoldásról, akkor az első iteráció során a rács hierarchiában lefelé menet a relaxáció végrehajtásának nincs előnye. Ezért ezt a lépést érdemes kihagyni. Ilyenkor először a legdurvább rácson oldjuk meg a problémát, majd a kapott megoldást felprolongáljuk sorban a finomabb rácsokra. Ha az MGRIT algoritmus elején jó kezdő közelítésünk van minden időpontban, akkor nem kifizetődő megspórolni a relaxációt.

Az MGRIT algoritmus elején pontatlan értékekkel számolunk, ezért nincs értelme nagy pontossággal végrehajtani az időbeli lépéseket. Az első pár iteráció során a Newton-módszernél előnyös nagyobb toleranciát beállítani a hibára. Az optimális értékeket méréseken keresztül lehet meghatározni.

Megvizsgálhatjuk, hogy miként érdemes megadni a legdurvább rács méretét. Amint már említettük, a ritka rácsokon sokba kerül a Newton-módszer, így sok időt spórolhatunk, ha nem megyünk le kisméretű rácsokra. Minél nagyobb a legdurvább rács, annál több időt vesz igénybe a rajta értelmezett egyenletrendszer megoldása. Ezért létezik egy optimális érték, ahol a futási idő a legkisebb.

Végül röviden összefoglaljuk, hogy a [5] cikkben milyen eredményeket értek el a mintafeladaton az előbb leírt ötletek használatával. A 3.4. ábrán látható táblázat soraiban az MGRIT algoritmus 17 variánsa szerepel. A második oszlop a térbeli ritkítási stratégiának felel meg, ahol az 1-es szám azt jelenti, hogy nem történt térbeli ritkítás, a 4-es azt, hogy késleltetve a negyedik szinten kezdtük el a ritkítást, míg a 4^* azt, hogy késleltetés nélkül ritkítunk. A harma-

dik oszlopban a javított kezdőértékek használatának módja szerepel, amiket igénybe vehetünk bárhol vagy csak a durva pontokban. A következő három oszlop az előbb említett ötleteket reprezentálja. Ha az első iteráció elején kihagyjuk a relaxációt, akkor a negyedik sorban a "Yes" szó áll. A kísérletek során a Newton-módszer hibahatára 10^{-7} volt minden szinten. Ha az ötödik oszlopban jelöltük, akkor az első három iteráció során olcsóbb Newton-módszert végeztünk 10^{-3}-os hibahatárral. A tér-idő tartományon a legfinomabb rácsháló $(64)^2 \times 4096$ pontból áll, melyet 4×128 processzor között osztottunk fel. Az időbeli ritkító faktor 4 volt. A hetedik oszlopban az szerepel, hogy egy processzornak hányszor annyi memóriát kell használnia, mint a szekvenciális algoritmusban $128 \rightarrow 4096$ időbeli processzor esetén.

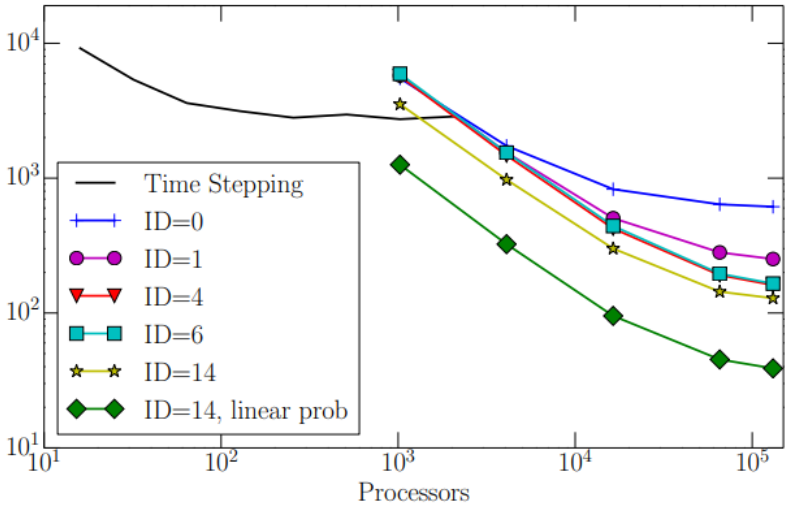
Solver ID	Spatial grids	IIG	Skip	Cheap first 3 iters	Coarsest grid size	Memory mult., $p = 128 \rightarrow 4096$	Runtime per iter	Iterations	Runtime
0	1	Never	No	No	4	$40 \rightarrow 16$	$162s$	7	$1135s$
1	4	Never	No	No	4	$31 \rightarrow 7$	$101s$	7	$712s$
2	4*	Never	No	No	4	$12 \rightarrow 2$	$57s$	30	$1717s$
3	1	C points	No	No	4	$53 \rightarrow 21$	$91s$	7	$638s$
4	4	C points	No	No	4	$42 \rightarrow 10$	$82s$	7	$579s$
5	1	Always	No	No	4	$84 \rightarrow 21$	$92s$	7	$647s$
6	4	Always	No	No	4	$73 \rightarrow 10$	$84s$	7	$591s$
7	1	C points	Yes	No	4	$53 \rightarrow 21$	$64s$	7	$453s$
8	4	C points	Yes	No	4	$42 \rightarrow 10$	$58s$	7	$410s$
9	1	Always	Yes	No	4	$84 \rightarrow 21$	$61s$	7	$429s$
10	4	Always	Yes	No	4	$73 \rightarrow 10$	$55s$	7	$391s$
11	1	Always	Yes	Yes	4	$84 \rightarrow 21$	$56s$	7	$395s$
12	4	Always	Yes	Yes	4	$73 \rightarrow 10$	$51s$	7	$360s$
13	1	Always	Yes	Yes	16	$80 \rightarrow 17$	$58s$	7	$408s$
14	4	Always	Yes	Yes	16	$73 \rightarrow 10$	$50s$	7	$354s$
15	1	Always	Yes	Yes	64	$76 \rightarrow 13$	$63s$	8	$506s$
16	4	Always	Yes	Yes	64	$73 \rightarrow 10$	$47s$	8	$383s$

3.4. ábra. Az MGRIT algoritmus variánsainak a teljesítménye a mintafeladatokon. [5]

Az ID-0 indexű a legegyszerűbb, naiv megközelítés (angolul az ún. out-of-the-box variáns). Ekkor nem használunk egy korábban leírt eszközt sem. Látjuk, hogy a két legfontosabb ötlet a térbeli ritkítás és a javított kezdőértékek használata. Az ID-4 futási ideje körülbelül fele az ID-0 variánsénak. A legjobb futási idővel az ID-14 algoritmus rendelkezik, amely kevesebb, mint harmadannyi időt vesz igénybe, mint a naiv módszer. Ekkor minden előzőekben említett ötletet használtunk, és a legdurvább rács méretét 16-nak választottuk.

Végül a 3.5. ábrán most is mutatunk egy erősen skálázó eredményt, melyen az MGRIT algoritmus különböző variánsainak a futási idejei láthatóak. A legfinomabb rács $(128)^2 \times 16385$ darab pontot tartalmazott, és a lineáris esethez hasonlóan a tér-idő tartományt egyenletesen osz-

tottuk fel a processzorok között, azaz mindegyikhez körülbelül ugyanannyi rácspont tartozott a legfinomabb rácshálón. Az időbeli ritkító tényezőt 4-nek választottuk, és 32 processzorral párhuzamosítottunk a térben.



3.5. ábra. Az MGRIT futási ideje a modell feladaton [5].

Ideális esetben, ahogyan a lineáris feladatnál már tárgyaltuk, a grafikonon egy egyenest látunk. Most azonban, ha megduplázzuk a processzorok számát, akkor egyre kisebb és kisebb gyorsítást érünk el. Ennek ellenére az eredmények nem rosszak. Ha legalább 1500 processzor áll a rendelkezésünkre, akkor az MGRIT algoritmust érdemes használni a klasszikus szekvenciális helyett. Továbbá 130 ezer processzor esetén nagymértékű, 21-szeres gyorsítás érhető el.

Érdemes összevetni az ID-14-hez tartozó sárga görbét az ID-14, linear prob nevű zöld görbével, amelyet egy hasonló, de lineáris feladatra, a hővezetési egyenletre alkalmazva kaptunk. A kódban annyi változtatás történt, hogy a p értékét 4-ről 2-re változtattuk. A zöld görbe a sárga alatt megy és nagyjából párhuzamosak. Ez azt jelenti, hogy nem meglepő módon a lineáris feladatra kapott algoritmus gyorsabb, viszont a futási idők hányadosa tetszőleges processzorszám esetén konstans, kb. három.

Hasonlítsuk össze a 3.5. ábra $p = 2$ esethez tartozó zöld görbéjét a 3.3. ábrán látható eredményekkel. Ez azért érdekes, mert mindkét mérést ugyanazon a feladaton végeztük, ennek ellenére most gyengébb eredményeket kaptunk. Ennek az a magyarázata, hogy a két kísérlet között több különbség is van. Az első, hogy most a végeselem módszert használtuk a térbeli diszkretizálás során a véges differenciák módszere helyett. A második, hogy másfajta MG módszert használtunk az implicit Euler módszer egy lépésénél a térbeli lineáris egyenletrendszer megoldásánál. A harmadik, hogy a lineáris esetben az implicit Euler módszer egy lépésénél a térbeli lineáris probléma együtthatói konstansok, nem függnek az aktuális közelítéstől. Ekkor az algoritmus

elején elég egyszer felépíteni az együttható mátrixot. A nemlineáris esetben viszont a Newton-módszer minden lépésében változik az együttható mátrix, ezért mindig újra inicializálni kell a mátrixot, amely összességében sok időbe telik. A skálázódásbeli különbséget a harmadik észrevétel jelenti, melynek megoldása még nem megoldott implementációs probléma.

Összefoglalva elmondhatjuk, hogy az MGRIT algoritmus hatékonyan alkalmazható lineáris és nemlineáris kezdetiérték-feladatok megoldására is sok processzor használatával. Először a 3.1. alfejezetben megmutattuk, hogy a Parareal algoritmus felfogható az időtengelyre alkalmazott kétrácsos módszerként. Ezután a 3.2. alfejezetben az MG módszereknél látott eszközök segítségével általánosítottuk az algoritmust több rácsra, melynek következtében az MGRIT módszerhez jutottunk. A természetes kiterjesztés nem vezet jól teljesítő algoritmushoz, ezért különböző megfontolásokat téve a paraméterek optimalizálására volt szükség. A [4] cikkben kipróbálták az algoritmust a (3.9) alakú lineáris hővezetési egyenleten, ahol jó eredményeket kaptak (lásd 3.3. ábra). Láthattuk, hogy 4096 processzor használatával majdnem 10-szeres gyorsítást tudtak elérni a szekvenciális algoritmushoz képest. Végül a 3.3. alfejezeteben kiterjesztettük a módszert nemlineáris feladatokra is a teljes approximációs séma segítségével. A célunk a lineáris esetben látottakhoz hasonló eredmények elérése volt. A nemlinearitás miatt az időbeli lépések megvalósítása komplikáltabb, ezért több ötletre is szükségünk volt, hogy a módszert használhatóvá tegyük. Az [5] cikk szerzői a (3.12) alakú, kétdimenziós, nemlineáris p-Laplace feladaton tesztelték az MGRIT algoritmust. Az eredmények, melyek hasonlóak a lineáris esetben tapasztaltakhoz, a 3.5. ábrán láthatóak.

A két mintafeladaton mi is tudtunk kísérletezni az eredeti implementációval a saját környezetünkben. Sajnos esetünkben nem állt rendelkezésünkre nagyon sok processzor, ezért a futási időre vonatkozó skálázó kísérleteket nem tudtuk reprodukálni. A részletekről és a további feladatokon elvégzett tesztekről a következő fejezetben írunk bővebben.

4. fejezet

Alkalmazások

Ebben a fejezetben különböző feladatokon végzett mérések eredményeit írjuk le. Először olyan kisebb méretű feladatokkal foglalkozunk, melyek közönséges differenciálegyenletek segítségével jellemezhetőek. Ezután áttérünk a lineáris és nemlineáris hővezetési egyenlet vizsgálatára, ahol a módszer paramétereinek a futási időre és a szükséges iterációk számára gyakorolt hatását elemezzük. Amint már a bevezetőben is említettük, az algoritmus implementációja elérhető a nyilvános XBraid GitHub repozitóriumban [22], amely C nyelven íródott és MPI-t (Message Passing Interface) használ a párhuzamosítás megvalósítására. A név elején lévő X a görög idő szó első betűje, míg a Braid szó arra utal, hogy az algoritmus különböző felosztású rácsok együttesét használja. A program könnyen kezelhető interfésszel rendelkezik, a felhasználónak csak az egylépéses módszert leíró függvényeket kell megadnia, és élvezheti az XBraid által nyújtott párhuzamosítást.

A kísérleteket az ELTE szerverein futtattuk, ahol 24 darab 2.20 GHz intel Xeon Silver 4214 típusú processzor állt a rendelkezésünkre, és az Open MPI 4.1.0. könyvtárat használtuk az MPI implementációjaként. Amint korábban említettük, az algoritmus optimális futási idejű $O(N)$, viszont a konstans nagyobb, mint a klasszikus szekvenciális algoritmusnál, így a gyorsítást a párhuzamosítás adja. Ahhoz, hogy gyorsabbak legyünk a szekvenciális algoritmusnál, tipikusan sok, több száz processzorral kell rendelkeznünk. Ennek hiányában a célunk az előző fejezetben leírt ötletek kipróbálása, valamint a skálázódás demonstrálása különböző feladatokon.

Az összes kísérlet esetében az (1.2) alakú kezdetiérték-probléma időbeli diszkretizálása során mindig állandó lépésközű felosztást használtunk. Az előző fejezetekkel összhangban jelölje $N+1$ az osztópontok számát, $u_0, u_1, \ldots, u_N$ a keresett közelítést rendre a $0 = t_0 < t_1 < \ldots < t_N = T$ pontokban, Δt a lépésközt és u_0 az ismert kezdetiértéket. A magasabb szinteken a ritkítás miatt megnő a lépésköz, így stabilitási szempontokat is figyelembe véve az

$$u_{i+1} = u_i + \Delta t f(t_{i+1}, u_{i+1}), \quad i = 0,1,\ldots,N-1 \tag{4.1}$$

implicit Euler módszert használtuk az időbeli diszkretizálás során. Ha külön nem említjük, akkor a tolerancia értékét 10^{-9}-nek állítottuk be, azaz annyi iterációt hajtottunk végre, hogy a reziduális értéke a legfinomabb rácson 10^{-9} alá csökkenjen. A ritkító tényezőt 2-nek választottuk, azaz feleztük a pontok számát.

4.1. Közönséges differenciálegyenletek

4.1.1. Kétdimenziós lineáris feladat

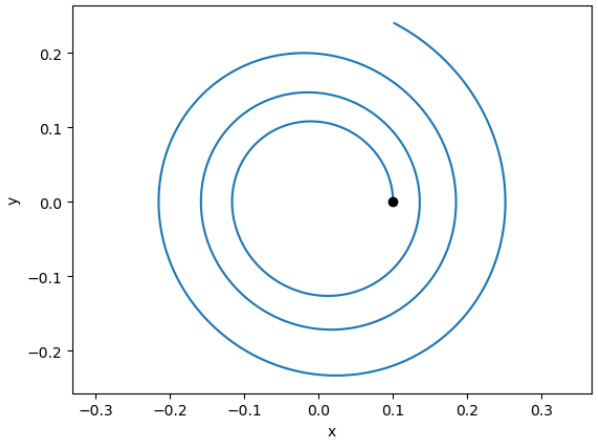
Először egy kétdimenziós, állandó együtthatós, lineáris jobb oldalú kezdetiérték-problémán próbáltuk ki az algoritmust, ahol az együttható mátrixot és a kezdeti értéket az alábbi módon választottuk:

$$A = \begin{pmatrix} 0 & -1 \\ 1 & 0.1 \end{pmatrix} \quad \text{és} \quad u_0 = \begin{pmatrix} 0.1 \\ 0 \end{pmatrix}.$$

Ekkor az implicit Euler módszer (4.1) egyenlettel felírt lépése az

$$(I - \Delta t A)u_{i+1} = u_i, \quad i = 0,1,\ldots,N-1$$

lineáris egyenletrendszer megoldásához vezet. Ebben a kétdimenziós esetben az $(I - \Delta t A)$ mátrix inverze könnyen, kézzel kiszámítható, így u_i ismeretében u_{i+1} értéket pontosan meg tudjuk határozni. Az MGRIT algoritmus egy iteratív módszer, így szükségünk van minden pontban egy kezdeti közelítésre, melyet most mindenhol u_0-nak választottuk. Tulajdonképpen ezzel azt tettük fel, hogy nincs előzetes tudásunk a megoldás menetéről. A mátrix sajátértékei pozitív valós részű komplex számok, így a megoldás trajektóriája egy kifelé haladó spirált ad, melyet a futtatás után a 4.1. ábrán is láthatunk.



4.1. ábra. A kétdimenziós feladat numerikus megoldása a $[0, 20]$ intervallumon.

Itt a legegyszerűbb variánst használtuk, azaz FCF-relaxációt hajtottunk végre minden szinten, a legdurvább szinten 4 részre osztottuk a [0,20] intervallumot, és a durvító tényező értékét 2-nek választottuk. Nem végeztünk méréseket a futási idővel kapcsolatban. A kis dimenziószám miatt az algoritmus nagyon gyorsan lefutott sok osztópont esetén is, ezért az egyéb műveletek (például inicializálás, vagy eredmények kiíratása) dominálták a futási időt. Viszont érdemes megjegyezni, hogy sokkal kevesebb iterációra van szükség, ha az F-ciklust használjuk a V-ciklus helyett. Ha az osztópontok száma 8193 volt, akkor a V-ciklusos variánsnak 16 iterációra volt szüksége a 10^{-9}-es hiba eléréséhez, míg az F-ciklus esetében ez a szám mindössze 6 volt. Ennek oka, hogy az u_0 vektor nem pontos közelítése a megoldásnak a későbbi időpontokban, így az F-ciklus jelentősen javít a kezdő közelítésen.

4.1.2. Többdimenziós lineáris feladat

Ezután végeztünk egy hasonló, de már nagyobb rendszerre vonatkozó kísérletet is, melynek dimenziószáma $d = 20$ volt. Az A ritka mátrixot úgy határoztuk meg, hogy véletlenül, egyenletes valószínűséggel kiválasztottuk az elemeinek az $1/5$ részét, ahova egy véletlen számot írtunk a $[-0.5,0.5]$ intervallumból. Ekkor minden sorban átlagosan 4 darab nem 0 elem van.

A kétdimenziós esettel ellentétben most nem tudjuk kézzel kiszámolni az $(I - \Delta tA)$ mátrix inverzét, ezért valamilyen iterációval kell közelítenünk a következő pontbeli megoldást. A tesztek során a csillapított Jacobi iterációt használtuk, amely koordinátás alakja valamely $\omega \in (0,1]$ esetén az $(I - \Delta tA)u_{i+1} = u_i$ lineáris egyenletrendszerre alkalmazva:

$$u_{i+1,j}^{k+1} = (1-\omega)u_{i+1,j}^{k} + \frac{\omega}{1-\Delta t a_{jj}}\left(u_{i,j} + \Delta t \sum_{l \neq j} a_{jl} u_{i+1,l}^{k}\right), \quad j = 1,2,\ldots,d,$$

ahol $u_{i+1,j}^{k}$ jelöli az u_{i+1} vektor k-adik közelítésének a j-edik komponensét. A Jacobi iteráció konvergenciájához elégséges feltétel, ha a mátrix diagonálisan domináns, amely elérhető finom lépésköz választásával, mivel $\Delta t \to 0$ esetén az $(I - \Delta tA)$ mátrix főátlóbeli elemei 1-hez tartanak, míg a többi a 0-hoz. Igaz, hogy a durvább szinteken a lépésköz növekszik, de a mátrixban kis abszolútértékű számok vannak, így továbbra is diagonálisan domináns marad az $(I - \Delta tA)$ mátrix. Az MGRIT algoritmushoz hasonlóan a Jacobi iterációval akkor állunk le, ha az $(I - \Delta tA)u_{i+1}^{k} - u_i^{k}$ lokális reziduális vektor eukleideszi értéke kisebb, mint 10^{-9}.

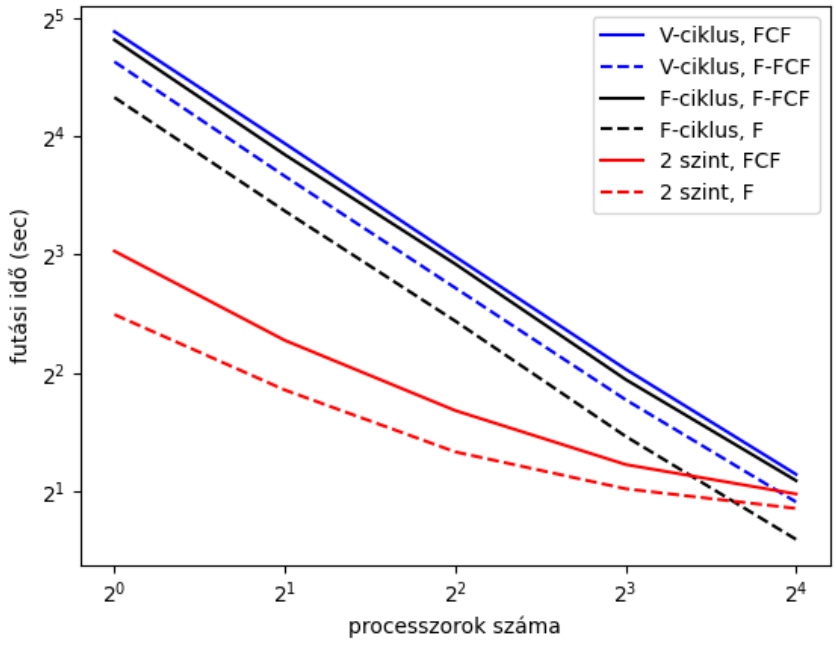
Először azt néztük meg, hogy az $\omega \in (0,1]$ választásától hogyan függ az algoritmus futási ideje V-ciklust és FCF- relaxációt használva a [0,5] időintervallumon. Az időintervallumot 8192 részre osztottuk fel, az MGRIT algoritmus hibahatárát 10^{-7}-nek választottuk, az u_0 kezdőérték minden koordinátáját véletlenszerűen választottuk a $[-1,1]$ intervallumból, és a kezdő közelítés a kétdimenziós esethez hasonlóan minden időpontban u_0 volt.

A lenti 4.1. táblázatból világosan látszik, hogy minél nagyobb ω értéke, annál gyorsabb az algoritmus. Ez nem meglepő, hiszen ekkor a Jacobi iterációnak kevesebb lépésre van szüksége a tolerancia eléréséhez. Viszont néha érdemes relaxálni, mert ekkor az iteráció stabilabb. Így előfordulhat az az eset, hogy a durvább rácsokon a csillapított Jacobi iteráció ugyan lassabb, de konvergens, míg $\omega = 1$ esetben nem az.

ω	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Idő (sec)	14.06	9.56	7.2	5.5	4.5	3.34	2.43	1.11

4.1. táblázat. A futási idő ω függvényében.

Ezután a futási időt vizsgáltuk a processzorszám függvényében különböző variánsok esetében (lásd 4.2. ábra). A kísérlet paraméterei megegyeznek az előző mérés paramétereivel. Itt ω értékét 0.5-nek választottuk, és a V-, illetve az F-ciklus esetében 10 szintet használtunk, így a legdurvább rácson 17 osztópont szerepelt. A feladat mérete állandó, így a processzorok számát növelve, az egy processzorra jutó osztópontok száma csökken, ezért azt várjuk, hogy a futási idő is csökken. Az ilyen típusú méréseket erősen skálázó (strong scaling) kísérleteknek nevezzük. Ideális esetben a processzorok száma fordítottan arányos a futási idővel, azaz, ha kétszer annyi processzort használunk, akkor a futási idő felére csökken.



4.2. ábra. A lineáris feladat megoldása V-ciklussal, F-ciklussal és 2 szintes algoritmussal.

Az eredményeket a fenti, log-log alapú 4.2. ábrán láthatjuk, ahol mindkét tengelyt kettes alapú logaritmus szerint skáláztuk. A V- és F-ciklust használó variáns esetében olyan egyenesek adódnak, amely meredeksége csak kis mértékben tér el az optimális értéktől. Nevezetesen ha

duplázzuk a processzorok számát, akkor a futási idő átlagosan 0.53-szorosára csökken. Ezzel szemben a két szintes algoritmus kisebb processzorszám esetén gyorsabb, viszont nem mutatja ugyanazt a jó skálázódást. Ennek oka, hogy a durvább szinten még mindig nagy a lineáris egyenletrendszer mérete, melynek megoldása még mindig sok szekvenciális lépést igényel. Ezen példán 1 processzor használatával az F-ciklust és F-relaxációt használó variáns körülbelül 4-szer annyi ideig fut, mint a két szintes algoritmus, viszont 16 processzor esetén már kevesebb futási időre van szüksége.

4.1.3. SIR-modell

Az utóbbi években sajnos fontossá vált a járványterjedés dinamikájának tanulmányozása, így az MGRIT módszert alkalmaztuk az egyik legegyszerűbb nemlineáris epidemiológiai modellre, az ún. SIR modellre is. Ez a modell feltételezi, hogy a populáció mérete állandó a vizsgált időtartam alatt, és az egyének az alábbi három csoportba oszthatóak:

- Fogékonyak ($S \sim$ susceptible): Megbetegedésre fogékony egyedek.
- Fertőzöttek ($I \sim$ infectious): Fertőzésre képes, jelenleg is fertőzött egyedek.
- Gyógyultak ($R \sim$ recovered): Fertőzésen átesett már immunis egyedek.

A rendszer dinamikáját az adja, hogy a fogékony egyedek megfertőződhetnek, és a fertőzöttek meggyógyulhatnak. Feltesszük, hogy minden fertőzött γ rátával gyógyul, és a fertőzés egyenesen arányos a fogékonyak és a fertőzöttek számának szorzatával. Ez azt jelenti, hogy ha egy fogékony találkozik egy fertőzöttel, akkor valamilyen valószínűséggel ő is megbetegszik. Ezen elveket felírva az alábbi

$$\begin{aligned} S'(t) &= -\frac{\beta I(t) S(t)}{N}, \quad S(0) = S_0, \\ I'(t) &= \frac{\beta I(t) S(t)}{N} - \gamma I(t), \quad I(0) = I_0, \\ R'(t) &= \gamma I(t), \quad R(0) = R_0, \end{aligned}$$

nemlineáris kezdetiéték-problémához jutunk, ahol

- N a populáció állandó mérete,
- β azt jelöli, hogy egy fertőzött egy időegység alatt mennyi fogékony egyedet betegít meg,
- γ pedig azt, hogy egy időegység alatt a fertőzöttek mekkora hányada gyógyul meg.

A 3.3. alfejezetben láttuk, hogy nemlineáris problémák esetén minden időbeli lépésben egy nemlineáris egyenletrendszert kell megoldanunk. Először részletesen leírjuk, hogy hogyan valósítottuk meg az implicit Euler módszer egy lépését a Newton módszer használatával. A fenti SIR modellt kompakt alakban az $u'(t) = f(u(t))$ alakban írhatjuk, ahol

$$u'(t) = \begin{pmatrix} S'(t) \\ I'(t) \\ R'(t) \end{pmatrix} \quad \text{és} \quad f(u(t)) = \begin{pmatrix} -\frac{\beta I(t) S(t)}{N} \\ \frac{\beta I(t) S(t)}{N} - \gamma I(t) \\ \gamma I(t) \end{pmatrix}.$$

Az implicit Euler módszer egy lépése során a (4.1) nemlineáris egyenletrendszert kell megoldanunk. Egy oldalra rendezve a tagokat ez átfogalmazható az

$$F(u_{i+1}) = u_i - u_{i+1} + \Delta t f(u_{i+1})$$

függvény gyökének a megkeresésére. Ezen feladat megoldására egy bevett módszer a Newton-iteráció

$$u_{i+1}^{k+1} = u_{i+1}^{k} - (J_F(u_{i+1}^{k}))^{-1} F(u_{i+1}^{k}),$$

ahol a gyakorlatban a J_F Jacobi mátrix invertálása helyett a

$$J_F(u_{i+1}^{k})(u_{i+1}^{k+1} - u_{i+1}^{k}) = -F(u_{i+1}^{k}) \tag{4.2}$$

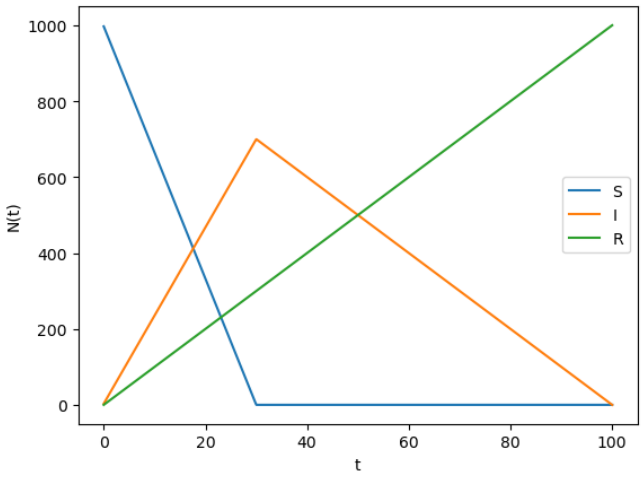
lineáris egyenletrendszert oldjuk meg. A SIR-modell esetén a Jacobi mátrix a következő alakot ölti:

$$J_F(S, I, R) = \begin{pmatrix} -1 - \Delta t \frac{\beta I}{N} & -\Delta t \frac{\beta S}{N} & 0 \\ \Delta t \frac{\beta I}{N} & -1 + \Delta t (\frac{\beta S}{N} - \gamma) & 0 \\ 0 & \Delta t \gamma & -1 \end{pmatrix}.$$

A kis dimenziószám miatt a Newton algoritmus minden lépésében a (4.2) egyenletet pontosan meg tudtuk oldani, nem volt szükségünk közelítő algoritmusok használatára. A Newton-módszer leállási feltétele hasonlóan az előző példákhoz az volt, hogy a lokális reziduális $F(u)$ függvény eukleideszi normája egy adott toleranciaérték, 10^{-9} alá essen.

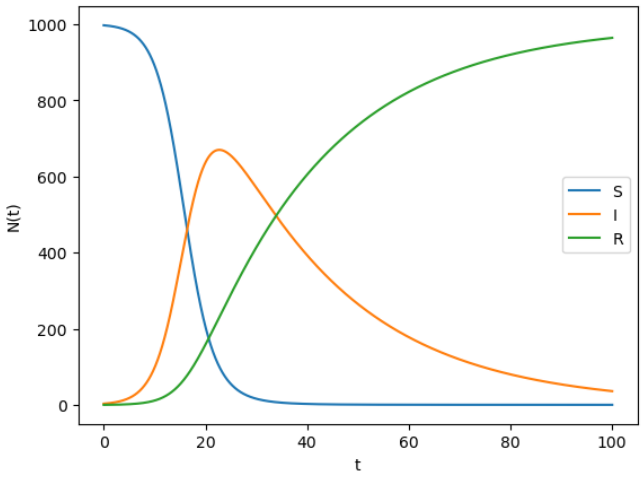
A probléma kis mérete miatt itt sem végeztünk futási idővel kapcsolatos méréseket. Helyette azt teszteltük, hogy milyen esetekben konvergens az algoritmus, és helyes eredményt kapunk-e. A rendszert 100 időegységig vizsgáltuk, míg a teljes populáció méretét $N = 1000$-nek választottuk. Feltettük, hogy kezdetben $I_0 = 3$ fertőzött egyedünk van, és mindenki fogékony a járványra, azaz $S_0 = 997$ és $R_0 = 0$. A két paramétert $\beta = 0.4$ és $\gamma = 0.04$ értékeknek választottuk. A toleranciát 10^{-6}-nak választottuk, és az időtengelyt 2048 részre osztottuk. Az MGRIT egy iteratív algoritmus, ezért minden $t_i > 0$ időpontban meg kell adnunk egy kezdeti közelítést. A várakozások szerint az idő előrehaladtával a fogékony egyedek száma csökken, a gyógyultak száma nő, míg a fertőzöttek száma először nő, majd elér egy maximum értéket és csökken.

Tegyük fel, hogy birtokában vagyunk egy korábban lejátszódott, hasonló paraméterekkel rendelkező folyamat adatainak, így van egy előzetes képünk a megoldás menetéről. Feltételezzük, hogy a három csoport elemszáma lineárisan változott a 4.3. ábra szerint. Természetesen nem ez a valós helyzet, de első közelítésként megfelel.



4.3. ábra. Az MGRIT kezdeti közelítése a SIR-modell esetén.

Először a kétszintes algoritmust próbáltuk ki. Az eredmény a lenti 4.4. ábrán látható. A szekvenciális algoritmussal összevetve helyes megoldást kaptunk. Ezután kipróbáltuk az algoritmust több szinttel is. Ha négy vagy több szintet használtunk, akkor az első pár MGRIT iterációban pár Newton-módszerrel végrehajtott lépés nem konvergens. Ez nem jelent gondod egészen $L = = 6$ szintig.



4.4. ábra. Az SIR-modell egyes osztálybeli egyedszámainak időbeli alakulása.

Alapértelmezettként az előző pontban kiszámolt értékből indítjuk a Newton-iterációt. Ezek folyamatosan javulnak, így pár MGRIT iteráció után minden időpontban megfelelő a kezdőérték és konvergens a lépés. Ugyanakkor, ha legalább $L = 7$ szintes a módszer, akkor a legdur-

vább rácson túl messze vannak egymástól a szomszédos pontok, és így az algoritmus divergál. Ez nem meglepő, mivel $L = 7$ szint esetén 33 osztópont van a legritkább rácson. Ekkor két szomszédos pont között több mint 3 időegység telik el. Végül megjegyezzük, hogy a javított kezdőértékek ötletének használata sem tudott segíteni a konvergencián. A kezdő közelítés nem elég pontos, így az első MGRIT iteráció során pár pontban ezek használatával sem konvergál a Newton-módszer.

4.2. Egydimenziós hővezetési egyenlet

4.2.1. Lineáris hővezetési egyenlet

Ha egy numerikus módszer hatékonyságát parabolikus parciális differenciálegyenletek esetén szeretnénk vizsgálni, akkor először a hővezetési egyenleten szokás azt tesztelni. Az MGRIT algoritmust a [4] cikk szerzői már kipróbálták a kétdimenziós egyenleten, de természetesen adódott az az ötlet, hogy vizsgáljuk meg a módszert egydimenziós esetben Dirichlet-peremfeltétel mellett.

$$\begin{aligned} \partial_t u(t,x) &= \alpha \partial_{xx} u(t,x), \quad x \in [0,1],\ t \in [0,T] \\ u(0,x) &= g(x), \quad x \in [0,1] \\ u(t,0) &= u_0(t), \quad t \in [0,T] \\ u(t,1) &= u_1(t), \quad t \in [0,T] \end{aligned}$$

A kísérletek során az α diffúziós együtthatót egynek választottuk, a két végponton a hőmérsékletet $u_0(t) = 1$ és $u_1(t) = 0$ fix értékeken tartottuk, míg a rendszert a $T = 5$ időpillanatig vizsgáltuk. A kezdetiérték függvényt, azaz a $g(x)$ függvényt a (0,1] intervallumon konstans 0-nak, míg a baloldali határpontban egynek választottuk. Ez fizikailag azt jelenti, hogy kezdetben mindenhol 0°C fok a hőmérséklet, majd a baloldali határponton keresztül hirtelen elkezdünk hőt közölni a rendszerrel, amely felmelegszik.

Először a térben diszkretizáltuk a feladatot, ahol a [0,1] térintervallumot $N_x + 1$ osztóponttal osztottuk fel ekvidisztánsan. Jelölje Δx térbeli lépésközt és $u_j(t),\ t \in [0,T]$ a rendszer időbeli viselkedését az x_j helyen. A szokásos másodrendű centrális differenciasémát használva egy

$$\begin{aligned} u_0'(t) &= 0, \\ u_j'(t) &= \alpha \frac{u_{j+1}(t) - 2u_j(t) + u_{j-1}(t)}{(\Delta x)^2}, \quad j = 1,2,\ldots,N_x - 1, \\ u_{N_x}'(t) &= 0 \end{aligned}$$

szemidiszkrét lineáris rendszer adódott, melynek tömör mátrix-vektor felírása

$$u'(t) = \frac{\alpha}{(\Delta x)^2} \begin{pmatrix} 0 & 0 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ & & & & & 0 & 0 \end{pmatrix} u(t).$$

Az idődiszkretizálás során most is az implicit Euler módszert használtuk. Ez egy sokdimenziós lineáris feladat, de most nincs szükségünk iterációs eljárást használni a módszer egy lépésénél kapott lineáris egyenletrendszer megoldásához. Az $(I - \Delta t A)$ mátrix tridiagonális, ezért optimálisan, $O(N_x)$ lépésben meg tudjuk határozni minden lépésben az $(I - \Delta t A)u_{i+1} = u_i$ rendszer pontos megoldását.

Mivel az MGRIT egy iteratív algoritmus, ezért minden $t_i > 0$ időpontban meg kell adnunk egy kezdeti közelítést. Ezeket most úgy határoztuk meg, hogy miden tér-idő rácspontban véletlen számokat vettünk a [0,0.1] intervallumból. Ez azt jelenti, hogy nem rendelkezünk előzetes tudással a megoldás időbeli változásáról. Természetesen fizikai megfontolások révén szofisztikáltabban is megadhattuk volna a kezdeti közelítéseket, amely egy gyorsabb algoritmust eredményezett volna.

Először a módszer konvergencia sebességét teszteltük. Azt vizsgáltuk, hogy egy iteráció alatt hányszorosára változik az MGRIT által adott megoldás eltérése a pontos megoldástól. Ezt nevezzük konvergencia faktornak. Továbbá azt is szeretnénk tudni, hogy a (3.10) és (3.11) egyenletekben adott becslések mennyire élesek. Az eredmények a 4.2. táblázatban találhatóak. Az első három oszlopban azt tüntettük fel, hogy a módszer melyik variánsát használtuk. A következő négy oszlopban találhatóak a mérési eredmények a probléma mérete szerint rendezve. Minden oszlopban az első koordináta az időtengelyen felvett pontok számát, míg a második a térbeli tengelyen felvett pontok számát jelöli. Az utolsó oszlopban a 3.2. Tételből adódó elméleti határ látható a kétszintes algoritmus esetén.

Összességében elmondható, hogy a konvergencia sebessége nem függ lényegesen az osztópontok számától. Ha a kétszintes algoritmust használjuk, és az m ritkító faktort kicsinek választjuk, akkor a konvergencia faktor a probléma méretének növelése közben állandó marad, míg a többi esetben is csak kis mértékben nő az értéke.

Érdemes összehasonlítani az F- és FCF-relaxációt használó variánsokat. Minden esetben az FCF-relaxációt használó módszer egy iteráció alatt kevesebb mint fele akkorára csökkentette a hibát, mint az F-relaxációt használó. A V-ciklust és F-relaxációt használó természetes kiter-

Relaxáció	Ciklus	m	512×64	1024×128	2048×256	4096×512	Határ
F-relax	2-szint	2	0.12	0.12	0.12	0.12	0.125
F-relax	2-szint	4	0.199	0.198	0.198	0.199	0.204
F-relax	2-szint	32	0.256	0.257	0.279	0.28	0.284
F-relax	V-ciklus	2	0.449	0.481	0.512	0.532	-
F-relax	V-ciklus	4	0.323	0.344	0.365	0.381	-
F-relax	V-ciklus	32	0.256	0.257	0.279	0.281	-
FCF-relax	2-szint	2	0.0494	0.0498	0.0499	0.0492	0.0527
FCF-relax	2-szint	4	0.0759	0.0764	0.0775	0.0777	0.0812
FCF-relax	2-szint	32	0.0515	0.0886	0.103	0.101	0.108
FCF-relax	V-ciklus	2	0.098	0.109	0.116	0.121	-
FCF-relax	V-ciklus	4	0.0922	0.0954	0.0981	0.101	-
FCF-relax	V-ciklus	32	0.0515	0.0886	0.103	0.101	-

4.2. táblázat. A konvergencia sebessége különböző variánsok esetében.

jesztés esetén a konvergencia faktor magas, kb. 0.5. Ennek következtében az algoritmusnak sok iterációra van szüksége egy előre maghatározott hiba eléréséhez.

Végül hasonlítsuk össze a kétszintes algoritmusnál a mért eredményeket az utolsó oszlopban található elméleti értékekkel. A vártnak megfelelően a tesztek során kapott eredmények kisebbek, mint a megfelelő elméleti határ, viszont elég közel vannak hozzá. Ez azt mutatja, hogy a [3] cikkben leírt becslések élesek ezen a feladaton.

A második kísérletben azt vizsgáltuk, hogy különböző méretű felosztások esetén hány iterációra van szüksége az algoritmusnak különböző variánsok esetén. Az eredmények a 4.3. táblázatban láthatóak, ahonnan leolvasható, hogy a feladat méretének növekedésével az iterációszámok korlátosak, kivéve a természetes kiterjesztés esetén (V-ciklus, F-relaxáció), amelynek használata nem vezet hatékony algoritmushoz. Az F-ciklus használata miatt egy iteráció gyorsabban lefut, viszont ez nem tudja kompenzálni az iterációszám drasztikus növekedését, így futási időt tekintve összességében egy lassabb algoritmust kapunk. Érdemes továbbá megjegyezni, hogy a várakozásoknak megfelelően mindegyik ciklus esetén F-relaxáció használatával több iterációra van szükség, mint FCF-relaxáció esetében.

Ezután megnéztük, hogy a ritkító faktor milyen hatással van a futási időre és az iterációszámra (lásd 4.4. táblázatban, ahol m a ritkító faktori jelöli). A kísérletet 16 processzorral végeztük, az időtengelyt 2048 részre osztottuk, míg a térben 8193 osztópontot vettünk fel. Minél nagyobb a ritkító faktor értéke, annál több lépést kell végeznünk párhuzamosan, viszont pontosabb közelítést kapunk egy iteráció során. Így a ritkító faktor növelésével csökken az iterációk száma,

	$N_t \times N_x =$	64×64	128×128	256×256	512×512	1024×1024
V-ciklus	F-relax	16	19	24	30	35
V-ciklus	FCF-relax	9	10	11	12	13
V-ciklus	F-FCF-relax	13	13	14	14	14
2-szint	F-relax	12	13	13	13	13
2-szint	FCF-relax	8	9	9	10	10
F-ciklus	F-relax	12	12	12	13	13
F-ciklus	FCF-relax	8	9	9	9	10
F-ciklus	F-FCF-relax	12	13	13	13	13

4.3. táblázat. A szükséges iterációk száma különböző variánsok esetében.

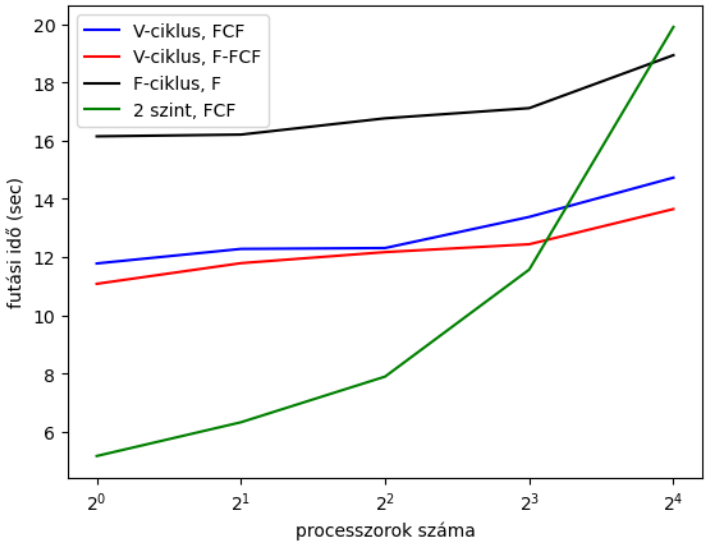
viszont a futási idő tekintetében egyfajta valamit valamiért (trade-off) helyzet alakul ki. Ugyan kevesebb iterációra van szükség, de egy iteráció tovább tart. A táblázatban azt látjuk, hogy az elején csökken a futási idő, viszont a végén a sok szekvenciális lépés miatt újra növekszik. Az optimum értéket $m = 128$ esetén érjük el, amikor kevesebb, mint negyed annyi időre van szükségünk, mint ha minden szinten feleznénk a problémát. Emeljük még ki az utolsó oszlopot, ahol mindösszesen egy iterációra volt szükség, mivel az egész $[0, T]$ intervallumon szekvenciálisan lépkedtünk végig.

m	2	4	8	16	32	64	128	256	512	1024	2048
Idő (sec)	8.58	4.97	3.54	3.24	3.67	2.91	2.02	2.40	3.82	6.42	8.32
Iteráció	13	13	12	12	12	11	7	4	3	2	1

4.4. táblázat. A futási idő, illetve a szükséges iterációszám különböző ritkító faktorok esetén.

Végül most is végeztünk egy mérést a futási időre vonatkozólag. Jelölje p a processzorok számát. Minden futásnál az időtengelyt $1024 \times p$ részre osztottuk fel, és a térben 2049 rácspontot vettünk fel. Ekkor az egy processzorra eső pontok száma megegyezik az összes mérésnél, így ideális esetben a futási idő változatlan marad. Természetesen egy kismértékű növekedést várunk a futási időben, mivel a processzoroknak kommunikálniuk kell egymással. Az ilyen típusú méréseket gyengén skálázó méréseknek nevezzük. Az eredmények a 4.5. ábrán láthatóak, ahol a vízszintes tengely kettes alapú logaritmus szerint van skálázva, míg a függőleges tengely lineáris.

A V- és az F-ciklus esetén nem emelkedett a futási idő drasztikusan. Például a V-ciklusos F-FCF relaxációt használó variáns esetében 1 processzor használatával 11.08 másodperc volt a futási idő, míg 16 processzor esetén 13.65 másodperc. Ez azt mutatja, hogy az algoritmus jól skálázódik. Ha kétszeresére növeljük a feladat méretét, és kétszer annyi processzort használunk,



4.5. ábra. A lineáris hővezetési egyenlet megoldásához szükséges futási idő.

akkor a futási idő nem változik lényegesen. A kétszintes algoritmus viszont nem rendelkezik ezzel a tulajdonsággal, mivel a durvább szint mérete még mindig nagy. Ha kevés processzor áll a rendelkezésünkre, akkor érdemes a kétszintes algoritmust használni, viszont már 16 processzor esetén jobban teljesítenek a többszintes variánsok.

4.2.2. Nemlineáris hővezetési egyenlet

A lineáris hővezetési egyenlet után kipróbáltuk az MGRIT algoritmust a nemlineáris hővezetési egyenleten is, melyre az irodalomban gyakran porózus közeg egyenletként hivatkoznak, melynek alakja

$$\begin{aligned} \partial_t u(t,x) &= \alpha \partial_{xx}(u^p(t,x)), \quad x \in [0,1],\ t \in [0,T] \\ u(0,x) &= g(x), \quad x \in [0,1] \\ u(t,0) &= u_0, \quad t \in [0,T] \\ u(t,1) &= u_1, \quad t \in [0,T] \end{aligned}$$

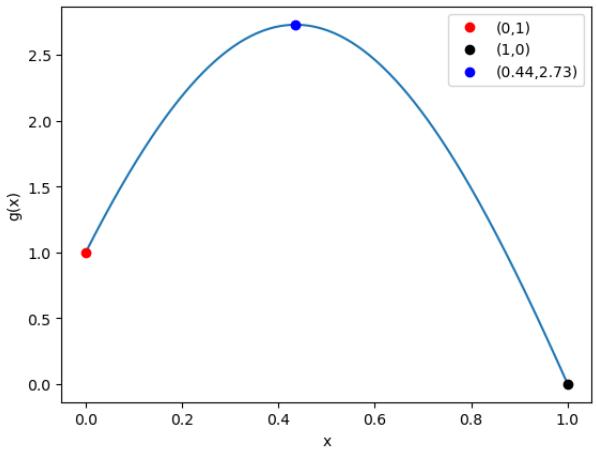
ahol mi a $p = 2$ esettel foglalkoztunk. Megjegyezzük, hogy a $p = 1$ esetben a lineáris hővezetési egyenletet kapjuk vissza. Ennek következtében az itt kapott eredmények összevethetőek az előző alfejezetbeli problémára adott algoritmus eredményeivel. A nagyobb nemlineáris problémákon több, a 3.3. alfejezben említett eszközt kell használnunk, hogy a lineáris esetben látott hatékonyságot érjünk el. Az itt bemutatott mérésekkel ezen ötletek hatását szeretnénk vizsgálni.

A kísérletek során a diffúziós konstanst $\alpha = \frac{1}{10}$-nek választottuk, hogy lassítsunk a folyamaton. A két végponton a hőmérsékletet fixen tartottuk, mégpedig $u_0 = 1$ és $u_1 = 0$ értékeken, míg a rendszert a $T = 5$ időpillanatig vizsgáltuk. A $g(x)$ kezdetiérték függvényt úgy határoztuk meg,

hogy teljesítse a peremfeltételeket és a [0,1] térintervallum belsejében legyen egy maximuma, ahonnan a peremek felé lecsökken. A választásunk a

$$g(x) = 10e^{-(x-0.435576)^2} - 7.27185$$

függvényre esett, melynek grafikonja a 4.6. ábrán látható.



4.6. ábra. Kezdetiérték függvény a nemlineáris hővezetési egyenletnél.

A lineáris esethez hasonlóan először térben, majd időben diszkretizáltuk a problémát. Írjuk át a jobb oldalt az összetett függvény deriválási szabályát alkalmazva

$$\alpha\partial_{xx}(u^2(t,x)) = 2\alpha\partial_x(u(t,x)\partial_x u(t,x)) = 2\alpha((\partial_x u(t,x))^2 + u(t,x)\partial_x u(t,x)),$$

és osszuk fel [0,1] térintervallumot a $0 = x_0 < x_1 < \ldots < x_{N_x} = 1$ pontokkal ekvidisztánsan, Δx lépésközzel. Jelölje most is $u_j(t)$ a rendszer időbeli viselkedését az x_j helyen. Az első és másodrendű centrális differenciaséma térbeli alkalmazásával egy nemlineáris közönséges differenciálegyenlethez jutunk, amelynek alakja

$$\begin{aligned}
u_0'(t) &= 0, \\
u_j'(t) &= \frac{2\alpha}{(\Delta x)^2}\left(\frac{(u_{j+1}(t) - u_{j-1}(t))^2}{4} + u_j(t)(u_{j+1}(t) - 2u_j(t) + u_{j-1}(t))\right), \quad j = 1,2,\ldots,N_x - 1, \\
u_{N_x}'(t) &= 0.
\end{aligned}$$

Az implicit Euler módszer egy lépésének megvalósításához az SIR modellnél látottakat követtük. Vezessük be az

$$F(u_{i+1}) = u_i - u_{i+1} + \Delta t f(u_{i+1})$$

nemlineáris függvényt, melynek a gyökét keressük minden időbeli lépésnél. A könnyebb kö-

vethetőség kedvéért írjuk ki az $F(u_{i+1}) = 0$ egyenletrendszert koordinátánként.

$$u_{i,0} = u_{i+1,0},$$
$$u_{i,j} = u_{i+1,j} - \frac{\alpha\Delta t\left((u_{i+1,j+1} - u_{i+1,j-1})^2 + 4u_{i+1,j}(u_{i+1,j+1} - 2u_{i+1,j} + u_{i+1,j-1}\right)}{2(\Delta x)^2}, \quad j = 1, \ldots, N_x - 1,$$
$$u_{i,N_x} = u_{i+1,N_x},$$

ahol $u_{i,j}$ jelöli a megoldás t_i időpontbeli közelítésének a j-edik komponensét. A megoldások közelítésére most is az

$$J_F(u_{i+1}^k)(u_{i+1}^{k+1} - u_{i+1}^k) = -F(u_{i+1}^k) \tag{4.3}$$

Newton-módszert használtuk. A $J_F(u_{i+1}^k)$ Jacobi-mátrix minden esetben tridiagonális, ugyanis $u_{i+1,j}^k$ csak az $u_{i+1,j+1}^k, u_{i+1,j}^k$ és $u_{i+1,j-1}^k$ értékektől függ. Ezért a (4.3) egyenletrendszert a térbeli felosztás N_x méretére vonatkozóan lineáris időben meg tudjuk oldani pontosan. Azaz a Newton lépéseket most is pontosan végre tudtuk hajtani optimális időben. A tesztek során a legfinomabb szinten annyi Newton-iterációt hajtottunk végre, hogy az $F(u_{i+1})$ reziduális függvény euklideszi normája egy bizonyos érték alá csökkenjen, míg a durva szinteken előre meghatározott számú iterációt csináltunk.

A kísérletek során, eltérve az eddigiektől, most 10^{-6} hibahatárral dolgoztunk, és a ritkító faktort $m = 4$-nek választottuk. Meg kell még adnunk a kezdeti közelítést minden $0 \leq t_i \leq T$ időpontban. Először a kezdeti közelítéseket $g(x)$-nek választottuk minden időpontban nem tudva, hogy a megoldás hogyan változik az időben. Ekkor azt tapasztaltuk, hogy kevés szint használatával konvergens volt az algoritmus, viszont, ha a legdurvább rácson az időbeli felosztás mérete kisebb volt, mint 512, akkor divergált a módszer a paraméterek megválasztásától függetlenül. Az implicit Euler módszert megvalósító Newton-módszer kezdőpontja túl messze volt a megoldástól. Ezért pontosabb kezdő közelítésre volt szükségünk. Ha a rendszer egyensúlyban van, azaz nem változik az időben, akkor $\partial_t u(t, x) = 0$. Így egyensúly esetén az

$$\begin{aligned}(u^2(x))'' &= 0, \quad x \in [0,1] \\ u(0) &= 1, \\ u(1) &= 0\end{aligned}$$

közönséges differenciálegyenlet írja le a rendszert. Ennek a megoldása $u(x) = \sqrt{1-x}$. Egy jobb kezdő közelítést kaphatunk, ha feltesszük, hogy a rendszer az egyensúlyi helyzet felé mozdul el az idő előrehaladtával, azaz

$$u^0(t, x) = \left(1 - \frac{t}{T}\right) g(x) + \frac{t}{T}\sqrt{1-x}, \quad x \in [0,1],\ t \in [0, T].$$

Ezen értékek használatával már mindig konvergens az algoritmus.

A 3.3. alfejezetben már említettük, hogy a nagy lépésköz miatt a durva rácsokon költséges a Newton-módszert végrehajtani. A legfinomabb rácson toleranción alapuló leállási feltételt használtunk. Annyi Newton-iterációt hajtottunk végre, hogy az $F(u_{i+1})$ reziduális függvény eukleideszi normája 10^{-9} alá csökkenjen. A durvább rácsokon, ahol nincs szükségünk olyan pontoson megoldásra, legfeljebb 10 iterációt hajtottunk végre. A másik gyorsító ötlet a térbeli ritkítás használata. Ha egy teszt során az idő mellett térben is ritkítottunk, akkor minden szinten felezték a térbeli osztópontok számát. Ennek következtében a $\frac{\Delta t}{(\Delta x)^2}$ hányados értéke állandó maradt minden rácson.

Először azt vizsgáltuk meg, hogy milyen méretű probléma esetén érdemes a térben is ritkítani. A $[0, T]$ időintervallumot 4096 részre osztottuk, és a térbeli lépésköz függvényében vizsgáltuk a futási időt. A mérések során 8 processzort használtunk. A legdurvább időbeli felosztás méretét 4-nek állítottuk be, így az algoritmus $L = 6$ szintes volt. Az eredményeket a 4.5. táblázat tartalmazza. A második oszlopban szereplő számok azt jelölik, hogy hanyadik szinten kezdjük a ritkítást, így 0 esetén rögtön, míg 1 esetén késleltetve, azaz csak a következő szinten.

N_x	Térbeli ritkítás	Iterációk száma	Idő (sec)
64	Nem	10	0.26
64	0	17	0.30
64	1	11	0.24
1024	Nem	10	2.77
1024	0	10	2.26
1024	1	10	2.61
4096	Nem	11	13.49
4096	0	11	9.54
4096	1	11	11.31

4.5. táblázat. A futási idő különböző méretű feladatokon térbeli ritkítással.

Ha kicsi a térbeli probléma mérete (a 4.5. táblázatban az $N_x = 64$ esete), akkor nem érdemes a térben ritkítani. A térbeli ritkítás miatt a módszer pontatlanabbá válik és annyival több iterációra van szüksége, amit nem tud kompenzálni a gyorsabb időbeli lépésekkel. Egy köztes megoldás a késleltetett ritkítás, ami most a leggyorsabb futást adja.
A táblázat második két részében azt látjuk, hogy ugyanannyi iterációra van szükségünk, ha ritkítunk, ha nem. Azaz a térbeli ritkítás nem ront a módszer konvergenciájának a sebességén. Ennek következtében a futási idő akkor lesz optimális, ha rögtön az első szinttől kezdve ritkítunk. Például az $N_x = 4096$ esetben a futási idő kb. 70%-ára csökkent.

Ezután azt néztük meg, hogy a 3.3 alfejezetben leírt további eszközök hogyan javítanak a futási időn. A [5] cikkben leírt, a 3.4. ábrán látható táblázatban összefoglalt kísérlethez hason-

ló mérést végeztünk. Az MGRIT algoritmus különböző variánsainak a futási idejét vizsgáltuk. Egy nagyobb problémát választottunk, így időben és térben is $N_t + 1 = N_x + 1 = 4097$ darab osztópontot vettünk fel. Az eredmények a 4.6. táblázatban láthatóak. Az első oszlopban (index) a variánsok indexei láthatóak. A második oszlopban azt jelöltük, hogy használtunk-e térbeli ritkítást (TR). Ha igen, akkor az előző kísérlet eredményeit figyelembe véve rögtön az első szinttől kezdve alkalmaztuk. A harmadik oszlop a javított kezdőértékeknek (JKÉ) felel meg. Ezeket használhatjuk mindig, vagy memória megtakarítás céljából minden szinten csak a durva pontoknál. Ha a negyedik, Skip címkéjű oszlopban igen szerepel, akkor az első iteráció első felében kihagyjuk a relaxációt. Ezeken kívül előnyös lehet az első pár iterációban megengedőbb Newton-toleranciát használni. Itt az első 4 iterációban 10^{-4}-es toleranciát használtunk. Ezt az ötödik oszlopban láthatjuk, ahol az OEPI mozaikszó az olcsó első pár iterációt rövidíti. Végül a legritkább rács méretét a hatodik, LRM oszlopban jelölt értékre állítottuk, ahol az LRM rövidítés a legritkább rács méretére utal.

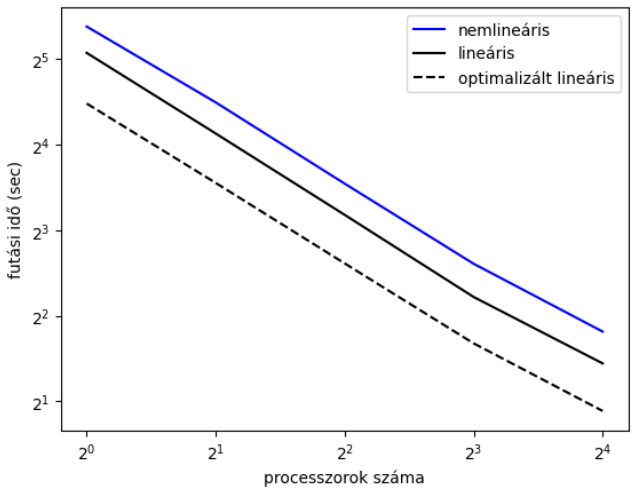
Index	TR	JKÉ	Skip	OEPI	LRM	idő (sec)
0	Nem	Nem	Nem	Nem	4	13.49
1	Igen	Nem	Nem	Nem	4	9.54
2	Igen	C-pontok	Nem	Nem	4	9.16
3	Igen	Mindig	Nem	Nem	4	8.13
4	Igen	Mindig	Igen	Nem	4	7.22
5	Igen	Mindig	Igen	Igen	4	6.12
6	Igen	Mindig	Igen	Igen	16	6.08

4.6. táblázat. A futási idő a különböző eszközök használatával.

A 4.6. táblázatban láthatjuk, hogy mindegyik ötlet gyorsabb futási időt eredményezett ennél a feladatnál. A legnagyobb javítást a térbeli ritkítás használata jelentette, és összességében negyedére csökkent a szükséges idő. Érdemes megjegyezni, hogy az első iterációban a kezdeti közelítés lehet, hogy messze van a megoldástól, így a javított kezdőértékek használata nem célszerű ekkor. Ha az első iterációban skipelünk, azaz kihagyjuk lefele menet a relaxációt, és a javított kezdőértékeket használjuk, akkor az algoritmus divergál. Ezért csak a második iterációtól kezdve használjuk a javított kezdőértékeket. Az iterációk száma minden esetben 11 volt, tehát a térbeli ritkítás nem rontott a konvergencia sebességén.

Végül összehasonlítottuk a lineáris és nemlineáris hővezetési egyenlet megoldásához szükséges futási időt. A probléma paramétereit az ebben az alfejezetben leírtak szerint választottuk, és a nemlineáris esetben az összes előbb említett gyorsító eszközt használtuk. Az algoritmus két variánsát próbáltuk ki a lineáris feladaton. A folytonos vonallal jelölt esetben az előző fejezetben bemutatott algoritmust futtattuk, míg a szaggatott vonallal jelölt esetben térbeli ritkítást és

a skip ötletet is használtuk. A probléma méretét most is $N_t = N_x = 4096$-nak választottuk. Az eredmények a 4.7. ábrán láthatóak.



4.7. ábra. A lineáris és nemlineáris hővezetési egyenletek megoldásához szükséges futási idők.

A grafikon három egyenes, amely azt jelenti, hogy a lineárishoz hasonlóan a nemlineáris feladaton is tapasztalható az algoritmus jó skálázódása. A futási idők sincsenek messze egymástól. A lineáris feladaton az optimalizált algoritmus körülbelül fele annyi idő alatt futott le, mint a nemlineáris esetben. Érdemes még megjegyezni, hogy a nemlineáris esetben 11, míg a lineáris esetben csak 8 iterációra volt szükség a leállási feltétel eléréséig. Ennek az lehet az oka, hogy a nemlineáris esetben a Newton-módszerrel csak közelíteni tudtuk u_i ismeretében az u_{i+1} vektort, míg a lineáris esetben pontosan ki tudtuk számolni.

Összességében elmondható, hogy sikerült igazolni az [5] cikkben leírtakat. Az ott megemlített eszközök segítségével jelentősen csökkent az algoritmus futási ideje, és nem sokkal lassabb algoritmust kaptunk, mint a lineáris esetben. Sőt, a 4.7. ábra jobb skálázódást mutat, mint az [5] cikkből származó 3.5. ábra. Ennek az oka az lehet, hogy az egydimenziós esetben a Newton lépéseket pontosan végre tudtuk hajtani optimális idő alatt, nem volt szükségünk térbeli MG módszereket használni ellentétben a cikkbeli kétdimenziós esettel.

Irodalomjegyzék

[1] A. Bellen and M. Zennaro: *Parallel algorithms for initial value problems for difference and differential equations*, Journal of Computational and Applied Mathematics, 25 (1989), pp. 341–350.

[2] P. Chartier and B. Philippe: *A parallel shooting technique for solving dissipative ODEs*, Computing, 51 (1993), pp. 209–236

[3] V. Dobrev, Tz. Kolev, N. A. Petersson, and J. B. Schroder: *Two-level convergence theory for Multigrid Reduction in Time (MGRIT)*, SIAM Journal on Scientific Computing, 39 (2017), pp. 501-527.

[4] R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, and J. B. Schroder: *Parallel Time Integration with Multigrid*, SIAM Journal on Scientific Computing, 36 (2014), pp.C635-C661.

[5] R. D. Falgout, T. A. Manteuffel, B. O'Neill, and J. B. Schroder: *Multigrid Reduction in Time for Nonlinear Parabolic Problems: A Case Study*, SIAM Journal on Scientific Computing, 39 (2017), pp 298-322.

[6] M. J. Gander: *Overlapping Schwarz for linear and nonlinear parabolic problems*, in Proceedings of the 9th International Conference on Domain Decomposition, (1996), pp. 97–104.

[7] M. J. Gander, L. Halpern, and F. Nataf: *Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation*, in Eleventh international Conference of Domain Decomposition Methods, (1999).

[8] M. J. Gander and E. Hairer: *Nonlinear convergence analysis for the parareal algorithm*, in Domain Decomposition Methods in Science and Engineering XVII, Springer, Berlin, Berlin, Heidelberg, (2008), pp. 45–56.

[9] M. J. Gander and S. Güttel: *ParaExp, A parallel integrator for linear initial-value problems*, SIAM Journal on Scientific Computing, 35 (2013), pp. C123–C142.

[10] M. J. Gander: *50 years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition Methods, Springer Verlag, (2015), pp. 69–114

[11] M. J. Gander and M. Neumüller: *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM Journal on Scientific Computing 38 (2016), pp. A2173-A2208.

[12] W. Hackbusch: *Parabolic multigrid methods*, Computer Methods in Applied Mechanics and Engineering, 7 (1984), pp. 189–197.

[13] A. Hessenthaler, B. S. Southworth, D. Nordsletten, O. Rohrle, R. D. Falgout, and J. B. Schroder, *Multilevel convergence analysis of multigrid-reduction-in-time*, SIAM Journal on Scientific Computing, 42 (2020), pp. A771-A796.

[14] G. Horton and S. Vandewalle: *A space-time multigrid method for parabolic partial differential equations*, SIAM Journal on Scientific Computing, 16 (1995), pp. 848–864.

[15] G. Horton, S. Vandewalle and P. Worley: *An algorithm with polylog parallel complexity for solving parabolic partial differential equations*, SIAM Journal on Scientific Computing, 16 (1995), pp. 531–541.

[16] Horváth Róbert, Izsák Ferenc és Karátson Kános: *Parciális differenciálegyenletek numerikus módszerei számítógépes alkalmazásokkal*, Elektronikus jegyzet, (2013)

[17] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli: *The waveform relaxation method for time-domain analysis of large scale integrated circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1 (1982), pp. 131–145.

[18] J. L. Lions, Y. Maday and G. Turnici: *A 'parareal' in time discretization of PDEs*, Comptes Rendus de l'Académie des Sciences, Série I, 332 (2001), pp. 661-668

[19] M. L. Minion: *A hybrid parareal spectral deferred corrections method*, Communications in Applied Mathematics and Computational Science, 5 (2010), pp. 265–301.

[20] W. L. Miranker and W. Liniger: *Parallel methods for the numerical integration of ordinary differential equations*, Mathematics of Computation , 91 (1967), pp. 303–320.

[21] J. Nievergelt: *Parallel methods for integrating ordinary differential equations*, Communications of the ACM, 7 (1964), pp. 731–733.

[22] XBraid: Parallel Time Integration with Multigrid, http://llnl.gov/casc/xbraid

[23] P. Worley: *Parallelizing across time when solving time-dependent partial differential equations*, in Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, (1992), pp. 246-252.