Párhuzamosított idődiszkretizáló módszerek

Kupás Vendel Péter

Témavezető: Fekete Imre

2022. december 19.

1. Bevezetés

A projektmunka zárásaként neurális közönséges differenciálegyenletekkel ismerkedtem meg, melyek a reziduális hálók folytonos megfelelőiként is értelmezhetőek. Ebben a folytonos esetben a paraméter optimalizáláshoz a hibavisszaterjesztéses algoritmus diszkrét esethez hasonló módon nem alkalmazható. Helyette az úgynevezett adjuntgált érzékenység (adjoint sensitivity) módszer használható, amely egy időben visszafelé haladó differenciálegyenlet megoldását követeli a tanítás közben.
Ha a veszteségfüggvényben szereplő kezdetiérték-feladat numerikus megolgásához klasszikus egylépéses módszert alkalmazunk, akkor a tanítás továbbra is szekvenciális marad és így sok időt vesz igénybe. Ezen segíthet az időbeli többrácsos redukciós módszer (Multigri Reduction In Time, a továbbiakban MGRIT), amely egy párhuzamos iteratív eljárás differenciálegyenletek megoldásainak közelítésére.

2. Időbeli többrácsos redukciós módszer

Időfüggő differenciálegyenlet numerikus megoldására alkalmazhatunk klasszikus egylépéses módszereket. Ekkor a zárt időintervallimot felosztjuk $t_0, t_1, \ldots, t_N$ pontokra és a t_{i-1} pontbeli közelítésből kiszámoljuk a t_i pontbeli közelítést. Ebben a tipikus esetben sok szekvenciális lépést kell végrehajtanunk. A mai processzorok lassan elérik maximális teljesítőképességük határát, így a többmagos számítógépek megjelenése óta természetes igény alakult ki a párhuzamosítható idődiszkretizációs módszerek felé.

Ezeket a módszereket aszerint csoportosíthatjuk, hogy milyen számításokat végeznek párhuzamosan. Térbeli párhuzamosításról akkor beszélünk, ha a tér irányaiban számolunk egyszerre több értéket. Ilyen jól ismert eljárás többek között a klasszikus többrácsos (multigrid, a továbbiakban MG) módszer.

Az időbeli párhuzamosító módszerek során egy lépésben több időbeli rácspontban végezzük párhuzamosan a számításokat. A valóságban 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.

A MGRIT eljárás egy többrácsos (multigrid, a továbbiakban MG) módszeren alapuló időben párhuzamosított módszer. A módszer alapötlete az, hogy egy egylépéses módszert alakít át időben párhuzamossá. Először röviden írunk a klasszikus MG módszerről, majd lineáris feladatokra definiáljuk az MGRIT algoritmust. Végül a teljes approximációs séma segítségével kiterjesztjük nemlineáris feladatokra is és megnézzük, hogy a Richardson extrapoláció miként gyorsíthatja a módszert.

2.1. Egylépéses módszerek

Tekintsük az

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

kezdetiérték-feladatot és osszuk fel a $[0, T]$ időintervallumot a $t_0, t_1, \cdots, t_N$ pontokkal. Ekkor minden egylépéses módszer $\Phi_i(\cdot)$ függvényekkel és g_i konstansokkal felírható az

$$\begin{aligned} u_i &= \Phi_i(u_{i-1}) + g_i, \quad i = 1, \ldots, N \\ u(0) &= u_0 \end{aligned}$$

alakban, ahol u_i az $u(t_i)$ közelítése. Ekkor a módszer által adott megoldás megegyezik az $Au = g$ egyenletrendszer megoldásával, ahol

$$A = \begin{bmatrix} I & & & \\ -\Phi_1 & I & & \\ & \ddots & \ddots & \\ & & -\Phi_N & I \end{bmatrix}.$$

Tegyük fel, hogy f lineáris és a módszer időfüggetlen, azaz ha $\Phi_i(u_{i-1}) = \Phi u_{u_{i-1}}$ $(i = = 1, \ldots, N)$. Ekkor egy lineáris egyenletrendszert kapunk.

2.2. Többrácsos (MG) módszerek

A klasszikus elliptikus differenciálegyenletek (Laplace- vagy Poisson-feladatok) egyik legjobban bevált megoldási módszere az MG módszer, amely több rácson végzett műveletek segítségevel határoz meg egy közelítő megoldást. Azaz minden $\Omega_l,\ l = 1, \ldots, L$ rácshoz

tartozik egy $A_l u = g_l$ lineáris egyenletrendszer, ahol A_l a differenciáloperátor diszkretizálásából származó véges differencia együtthatómátrixot, míg l az aktuális szintek számát jelöli.

A kétrácsos MG módszernél két rácsunk (szintünk) van, egy finom és egy durva. Természetesen a feladat közelítő megoldását a finom rácson szeretnénk meghatározni. Az optimális műveletszám érdekében az elérhető megoldást elősimítjuk és az ezzel nyert maradékvektort (reziduálist) leszűkítjük (restriktáljuk) a durva rácshálóra. A már szignifikánsabb kisebb méretű durva rácshálón az egyenletrendszert direkt módon megoldjuk, majd utósimítjuk és interpoláljuk (prolongáljuk) a finom rácshálóra. Ezt a lépést (ciklust) többször is megismételjük a kívánt pontosság elérése érdekében. A szűkítés és az interpolálás az azonos dimeziószám elérése miatt fontos, és ezeket az operátorokat a gyakorlatban adott mátrixokkal reprezentáljuk. Az elő- és utósimítás szerepe pedig abban áll, hogy nemtriviális magtér okozta problémát és az abból adódó sajátfüggvény oszcillációját feloldja. Az elő- és utósimitást klasszikus iteratív eljárásokkal (csillapított Jacobi- vagy Gauss–Seidel-iterációkkal) végezzük [3].

A többrácsos eset hasonlóan, rekurzív módon kezelhető. A rekurzív hívások sorrendjének tekintetében többfajta ciklust különböztetünk meg. A gyakorlatban a V-, a W- és az F-ciklust használják. Az MG eljárást nemcsak klasszikus elliptikus, hanem időfüggő parciális differenciálegyenletre is szokták alkalmazni. Ebben az esetben az időtengelyt nem különböztetjük meg a térbeli dimenziótól és ekkor a módszert tér-idő (space-time) MG módszernek hívjuk.

2.3. Parareal algoritmus

A parareal kezdetiérték-feladatra alkalmazható idő-párhuzamosítható eljárást 2001-ben Lions, Maday és Turicini vezetette be [5]. Jelölje a $[0, T]$ időintervallum δt lépésközű felosztását $\{t_i\}_{i=0}^{N}$ és az $m\delta t$ lépésközű felosztását $\{T_j\}_{j=0}^{N/m}$. Előbbi a finom, utóbbi pedig a durva rács analógiája. Alkalmazzuk az F egylépéses módszert a finom, míg a G egylépéses módszert a durva rácson. Ekkor F tekinthető egy a durva rácson értelmezett nagyobb pontosságot adó egylépéses módszernek.

A Parareal algoritmus a durva rács $T_0, T_1, \ldots, T_{N/m}$ időpontokban értelmezett kezdeti $U_j^0,\ j = 0, \ldots, N/m$ közelítéssel indul, melyet tipikusan a G módszer szekvenciálisan alkalmazásával kaphatunk meg. Amennyiben $G(t_2, t_1, u_1)$ jelöli a G egylépéses módszer eredményét a t_2 pontban az $u(t_1) = U_1$ feltétel mellett, akkor a kezdeti közelítést az

$$U_0^0 = u_0,\ U_{n+1}^0 = G(T_{n+1}, T_n, U_n^0)$$

összefüggéssel nyerjük. Ekkor a Parareal algoritmus az

$$U_{n+1}^{k+1} = F(t_{n+1}, t_n, U_n^k) + G(t_{n+1}, t_n, U_n^{k+1}) - G(t_{n+1}, t_n, U_n^k), \quad n = 0, \ldots, N-1,\ k = 0,1,\ldots$$

korrekciós sémát jelenti. Itt az F módszert tudjuk párhuzamosan futtatni a durva rácspontok között, azaz az $F(t_{n+1}, t_n, U_n^k)$ értékeket egyszerre tudjuk kiszámítani.

Korábban feltettük, hogy lineáris feladatunk és időfüggetlen módszerünk van. Legyen $Au = g$ a finom rácsra alkalmazott F egylépéses módszer egyenletrendszere, $A_\Delta u_\Delta = g_\Delta$ az F egylépéses módszer durva rácshoz tartozó egyenletrendszere, míg $B_\Delta u_\Delta = g'_\Delta$ a durva rácsra alkalmazott G egylépéses módszer egyenletrendszere. Ekkor a fenti korrekciós séma az

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

alakot ölti.

Alkalmazzuk a kezdetiérték-feladatra a következő kétrácsos MG módszert.

1. 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. $u^{k+1} = u^k + Pv$

Az F-relaxáció azt jelöli, hogy minden durva pontok által meghatározott intervallumban szekvenciálisan lépünk az F módszerrel a finom pontokon. Ezt párhuzamosan tudjuk végrehajtani és az elősimítás analógiája. Az R_I leszűkítő és P interpolációs mátrixot alkalmas megválasztásával éppen a parareal algoritmussal megegyező eljárást kapunk [2].

2.4. MGRIT

Az előbb bemutatott parareal algoritmus gyengesége, hogy sokszor a durva rács mérete összemérhető a finom rács méretével, így az 1. algoritmusban a harmadik lépés végrehajtása szintén sok időt vesz igénybe. Ezen segít, ha a MG módszereknél látott módon a harmadik lépésben rekurzívan meghívjuk az algoritmust egy még durvább rácson. Ezt iteráljuk, majd a legdurvább rácson megoldjuk a kapott lineáris egyenletrendszert és végül sorban visszainterpoláljuk a megoldást a legfinomabb rácsra. Így lényegében a MGRIT módszer legegyszerűbb változatához jutunk.

A MGRIT algoritmusnak sok változtatható paramétere van. A [2] cikkben a szerzők ezen paraméterek optimális megválasztását keresték egy parabolikus mintafeladaton, a

homogén Dirichlet peremfeltételű hővezetési egyenleten.

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

A tesztek során a jobboldali és a kezdeti függvények egyszerű szinuszos függvények voltak és a diffúziós együtthatót $\kappa = 1$-nek választották. Térben a klasszikus másodrendű 5-pontos stencilt, míg időben az implicit Euler módszert alkalmazták.

Az első észrevétel az volt, hogy előbb látott természetes kiterjesztés nem vezet hatékony algoritmushoz. Az F-relaxáció helyett ún. FCF-relaxációt érdemes végrehajtani, ahol egy C-relaxációban a durva pontokban felvett értéket frissítjük az F-módszerrel az előző finom pontból.

Ezen felül megválaszhatjuk például a ritkítás mértékét, eldönthetjük, hogy milyen egylépéses módszereket használunk, meghatározhatjuk a rekurzív hívások sorrendjét, vagy alkalmazhatunk konvergenciát gyorsító plusz eljárásokat. A cikk fontos része az implicit egylépéses módszerek esetének vizsgálata. Egy ilyen eljárás használatakor a 2.1. alfejezetben definiált, a módszert leíró $\Phi(\cdot)$ függvény alkalmazása egy térbeli lineáris egyenletrendszer megoldását jelenti. A gyors futási idő érdekében ezeket nem egzaktul, hanem a 2.2. alfejezetben látott térbeli MG módszerrel iteratívan oldjuk meg. Ekkor két kérdés merül fel: hogyan válasszunk jó kezdőértékeket és milyen pontossággal adjuk meg a megoldást.

A paraméterek optimalizálása utáni mérések azt mutatják, hogy ha sok processzorral rendelkezünk, akkor az MGRIT algoritmus hatékonyan használható a homogén Dirichlet peremfeltételű hővezetési egyenlet megoldásának közelítésére. Ennél a mérésnél a tér-idő tartományt egyenletesen osztottuk fel a processzorok között, azaz mindegyikhez körülbelül ugyanannyi rácspont tartozott a legfinomabb rácshálón. A várakozásoknak megfelelően kevés processzor esetén az egylépéses módszer gyorsabb.

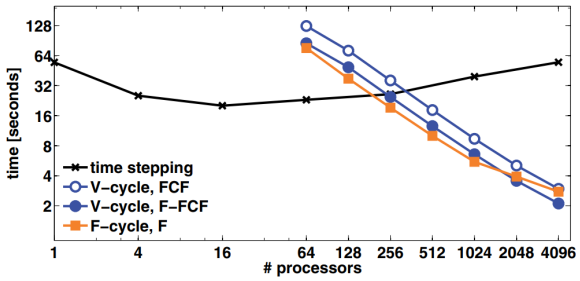

FIG. 9. *Time to solve* Implicit2D($T = \pi^2$) *on a* $129^2 \times 16{,}385$ *space-time grid using sequential time stepping and three MGRIT variants.*

1. ábra. A futási idő a processzorszám tekintetében az egyes eljárásokra nézve [2].

Ahogy a fenti 1. ábrán is látható, ha legalább 256 processzor áll rendelkezésünkre, akkor előnyös az MGRIT módszert használni. Továbbá 4096 processzor esetén már jelentős, 20-szoros gyorsítást lehet elérni.

Ez az algoritmus viszonylag egyszerűen kiterjeszthető nemlineáris feladatokra is a teljes approximációs séma (Full Approximation Scheme, rövidítve FAS) segítségével.

Tegyük fel, hogy egy implicit egylépéses módszert alkalmazunk egy nemlineáris feladatban. Ekkor a Φ_l kiszámolása egy nemlineáris egyenletrendszer megoldásával egyenértékű. A gyakorlatban a Newton-módszert használjuk a megoldás közelítésre. A ritkítás miatt a durva rácsokon nagy az időbeli lépésköz, így az előző időpontban kiszámolt érték messze van az egyenletrendszer megoldásától. Ekkor a gyenge kezdőérték miatt több iterációra van szükség a Newton-módszer során, hogy megfelelő pontosságú megoldást kapjunk. Ennek következtében a futási idő megemelkedik és a módszer használhatatlanná válik. Egy megoldás a problémára a térbeli ritkítás, amelynek köszönhetően a $\Phi(.)$ függvénynek megfelelő lineáris egyenletrendszer mérete csökken. A leszűkítést az R_x restriktáló, míg az interpolációt a P_x prolongáló operátor valósítja meg a térben. A szűkítés miatt a módszer pontatlabbá válik, így több V-ciklusra van szükség a megfelelő pontosság eléréséhez. Tehát a megfelelő ritkítási stratégia megtalálása fontos része az algoritmusnak. A [4] cikk nagyrészt ezzel a problémával foglalkozik.

Vegyük az Ω_l, $l = 0,1,\ldots,L$ felosztásokat a δt, $m\delta t$, stb lépésközzel egy m durvító tényezővel és minden felosztáson használjunk egy egylépéses módszert. Általában minden szinten ugyanazt a módszert használjuk. Jelölje $A_l u^{(l)} = g^{(l)}$ az Ω_l-hez tartozó egyenletrendszert, ahol A_l mátrixot a Φ_l függvény határozza meg. Ezekkel a jelölésekkel a MGRIT módszert a 2 algoritmus írja le.

A lineáris esethez hasonlóan most is sok beállítható paramétere van a módszernek. Ezek optimalizálásához vegyük a Neumann határfeltételű p-dimenziós Laplace parabolikus modell feladatot.

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

A következő ábrákon látható eredmények a [4] cikkből származnak, ahol a szerzők egylépéses módszerként az implicit Euler-módszert használták míg a térbeli diszkretizációhoz az MFEM software-t vették igénybe. Az összes esetben a tesztek során $p = 4$, $T = 4$, $\Omega = [0,2]^2$, $m = 4$ és a felosztás egyenletes. A $b(x,t)$ függvény úgy lett választva, hogy az egzakt megoldás az alábbi szinuszos függvény legyen:

$$u(x,y) = \sin(\pi x)\sin(\pi y)\sin((13/6)\pi t).$$

2. Algoritmus MGRIT-FAS(l)

if l a legdurvább szint **then**

Oldjuk meg az $A_L(u^{(L)}) = A_L(R_I(u^{(L-1)})) + g^{(L)}$ rendszert

else

Az $A_l(u^{(l)}) = g^{(l)}$ rendszer relaxálása FCF-relaxációval

Reziduális számítása és szűkítése injekcióval: $g^{(l+1)} = R_I(g^{(l)} - A_l(u^{(l)}))$

A megoldás leszűkítése: $u^{(l+1)} = R_I(u^{(l)})$

if Térbeli ritkítás **then**

$u_i^{(l+1)} = R_x(u_i^{(l+1)})$ és $g_i^{(l+1)} = R_x(g^{(l+1)})_i$

end if

Megoldás a következő szinten: $MGRIT(l+1)$.

$e^{(l+1)} = u^{(l+1)} - u^{(l)}$

if Térbeli ritkítás **then**

$e_i^{(l+1)} = P_x(e_i^{(l+1)})$

end if

$u^{(l)} = u^{(l)} + Pe^{(l+1)}$

Az $A_l(u^{(l)}) = g^{(l)}$ rendszer relaxálása F-relaxációval

end if

A célunk a lineáris esetnél látott gyorsítás megközelítése, elérése. A 2. ábrán látható táblázatában a MGRIT módszer 16 variánsa látható.

Az ID-0 indexű a legegyszerűbb, naív megközelítés, az úgynevezett out-of-the-box variáns. Ekkor nem használunk térbeli ritkítást és semmilyen később leírt eszközt sem. A legjobb futási idővel az ID-14 algoritmus rendelkezik, amely kevesebb mint harmadannyi időt vesz igénybe mint a naív módszer.

A két legfontosabb futási időt csökkentő módszer a térbeli ritkítás, illetve a javított kezdőérték használata a Newton-módszernél (a 2. ábrán IIG rövidítéssel szerepel).
A futási időt erősen befolyásolja, hogy mikor ritkítunk. Ha a 2. ábra második oszlopában 1-es szám, akkor nem ritkítottunk. Ha 4*, akkor a 4 legfinomabb rácson alkalmazunk ritkítás. Míg ha 4-es, akkor szintén 4 rácson ritkítunk, de nem a 4 legfinomobban, hanem úgymond késleltetve csak a durvább rácsokon. Érdemes összevetni az ID-1 és ID-2 algoritmusok futási idejét. Jól látszik, hogy ezen modellfeladat esetén a késleltetett ritkítást a legelőnyösebb használni.

Mint már esett szó róla, a futási idő egyik sarkalatos pontja a megfelelő kezdőpont választása a Newton-iterációknál. Ezen segít a következő triviális ötlet: Minden szint minden pontbában mentsük el a $\Phi(.)$ által legutóbb adott eredményt és használjuk ezt kezdőértéknek a következő ciklusban. Az egyedüli hátránya ennek, hogy plusz egy vektort

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

Table 4: Overall runtimes, storage costs, iteration counts and average time per iteration for the various solver options, with a $(64)^2 \times 4096$ space-time grid. A * indicates no delay in spatial coarsening.

2. ábra. A MGRIT variánsai nemlineáris feladatokon [4].

el kell tárolni minden durva rács minden pontjában. A memória használat redukálása érdekében egy köztes megoldás lehet, hogy csak minden szint durva rácsán tesszük meg ezt. Ekkor csak a finom rácson végzett F-relaxáció során a kell a megelőző pontban szereplő értéket használni.

Ezen módszereken túl még beszélhetünk olyan eszközökről, melyek nem olyan nagymértékben befolyásolják a futási időt. Ha nincs ismeretünk a megoldásról, akkor az első iteráció során a rács-hierarchiában lefelemenet kihagyhatjuk a relaxációt. Ennek következtében az első pár iteráció során pár Newton-eljárás drágább lehet, de összfutási időt tekintve a 2. ábra táblázata alapján előnyös ezt csinálni. Az első pár iteráció során pontatlan kezdőértékei vannak a Newton iterációnak, ezért érdemes ekkor nagyobb toleranciát beállítani a hibára. Végül megvizsgálhatjuk, hogy miként érdemes beállítani a legdurvább

rács méretét. Mint már említettük, a durva rácsokon sokba kerül a Newton-módszer, így sok időt spórolhatunk. Viszont az algoritmus szekvenciális része ekkor nő, mert a legdurvább rács mérete is növekedik.

Összefoglalva a 2. ábra táblázata azt mutatja, hogy az ID-14 módszer a leghatékonyabb. Tehát érdemes késleltetett térbeli ritkítást használni, javított kezdőértékekkel dolgozni, az első iterációban elhagyni a relaxálást, nagyobb tűréshatárt beállítani az első 3 iterációban szereplő Newton-módszerre és érdemes 16-nak választani a legdurvább rács méretét.

Végül a nemlineáris esetben is mutatunk egy eredményt a fent leírt modellproblémára (3. ábra). A lineáris esethez hasonlóan a tér-idő tartományt egyenletesen osztottuk fel a processzorok között, azaz mindegyikhez körülbelül ugyanannyi rácspont tartozott a legfinomabb rácshálón.

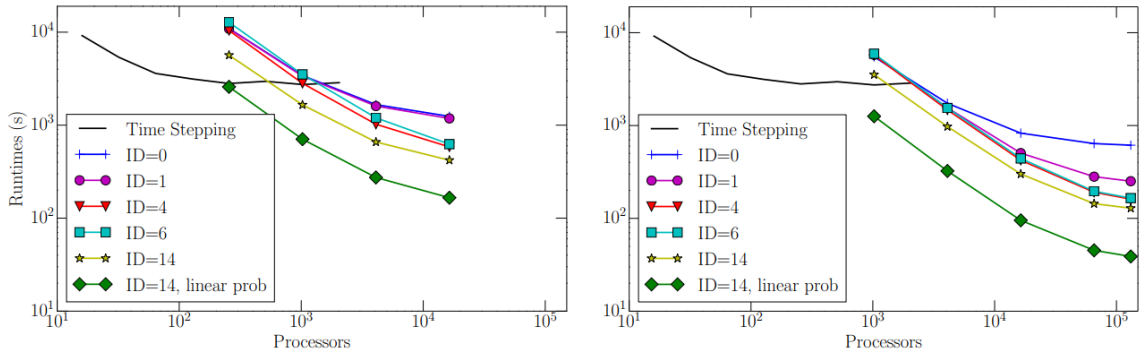

Fig. 5: Strong scaling study for a $(128)^2 \times 16385$ space-time grid, Left: 16 processors in space, $m = 16$, Right: 32 processors in space, $m = 4$.

3. ábra. Az MGRIT futási ideje a modellfeladaton [4].

Az (a) esetben a váltópont, ahonnan az MGRIT algoritmust érdemes használni körülbelül 1024 processzor, ami $\frac{1024}{16} = 64$ idő-tengely beli processzort jelent. Továbbá 16384 processzor esetén 6,4-szeres gyorsítást lehet elérni a szekvenciális algoritmushoz képest.
A (b) esetben a váltópont kb 2000 processzor (50 processzor az idő-tengelyen) és 130 ezer processzor esetén 21-szeres gyorsítás érhető el.
Érdemes összevetni az ID-14-hez tartozó görbét az ID-14, linear görbével, amelyet egy hasonló, de lineáris feladatra, a hővezetési egyenletre alkalmazva kaptunk ($p = 2$ eset). A zöld görbe a másik 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, körübelül 3.

3. Neurális közönséges differenciálegyenletek

A neurális közönséges differenciálegyenletek kaphatóak a reziduális hálókból folytonos határátmenetet képezve [1]. Ezek olyan differenciálegyenletek, ahol a jobboldal parametrizálva van a tanulási paraméterekkel.

$$u(0) = u_0, \quad u'(t) = f_\theta(t, u(t))$$

Kiindulva az $u(0)$ bemeneti rétegből definiálhatjuk az $u(T)$ outputot, mint a kezdetiérték probléma megoldását a T pontban. A legtöbb kezdetiérték probléma nem oldható meg egzaktul, így egy megfelelő numerikus módszert (ODESolve) használunk erre a célra.

A legnagyobb technikai kihívás a folytonos mélységű hálók tanítása, mert ez nem valósítható meg a diszkrét esetnél bevált hibavisszaterjesztéses (error backpropagation) eljárással. Egy természetes módszer, hogy a használt numerikus módszer minden lépésében kiszámítjuk a megfelelő deriváltérékeket az előző lépés segítségével. Ez viszont a diszkrét esethez hasonlóan túl sok memóriafelhasználást igényel és további numerikus hibához vezet.

Erre nyújt egy megoldást az adjungált érzékenység módszer. Tekintsük a következő skalárértékű veszteségfüggvényt:

$$L(u(T)) = L(u(0) + \int_0^T f_\theta(t, u(t))dt) = L(ODESolve(u_0, f, 0, T, \theta)),$$

ahol a függvény bemenetét a numerikus eljrárás szolgáltatja. Ekkor az L veszteségfüggvény paraméterekre vonatkozó, azaz θ szerinti gradiensét kell kiszámolnunk az L optimalizálásához.

Először meghatározzuk, hogy a veszteségfüggvény gradiense hogyan függ az $u(t)$ rejtett rétegektől, azaz legyen $a(t) = \frac{\partial L}{\partial u(t)}$ az úgynevezett adjungált függvény. Ekkor $a(T) = \frac{\partial L}{\partial u(T)}$ ismert és belátható, hogy a

$$\frac{da(t)}{dt} = -a(t)^T \frac{\partial f_\theta(t, u(t))}{\partial z}$$

közenséges differenciál megoldása szolgáltatja az $a(t)$ adjungáltat. Így $a(t)$-re egy olyan differenciálegyenletet kaptuk, amelynek a végén tudjuk az értékét. Tehát egy másik numerikus módszerrel visszafelé haladva megkapjuk az $a(t)$ függvényt. Az egyetlen nehézség, hogy tudnunk kell az $u(t)$ értékeket az egész $[0, T]$ intervallumon. Viszont ezen értékek könnyen megkaphatók visszafelé számolva az előző differenciálegyenlet megoldása közben az $u(T)$ értékből.

Végül a veszteségfüggvény θ szerinti gradiense kiszámolható a

$$\frac{dL}{d\theta} = -\int_0^T a(t)^T \frac{\partial f_\theta(t, u(t))}{\partial \theta} dt.$$

integrállal. Az

$$a(t)^T \frac{\partial f_\theta(t, u(t))}{\partial z} \text{ és a } a(t)^T \frac{\partial f_\theta(t, u(t))}{\partial \theta}$$

vektor-Jacobi mátrix típusú szorzások az úgynevezett automatikus differenciálással kiértékelhetőek f kiértékelésével összemérhető költségen.

A három függvény egy vektorba fűzhető és egy numerikus módszerrel kiszámolható a következő algoritmussal.

3. Algoritmus Reverse-mode derivative of an ODE initial value problem

Input: $\theta, 0, T, u_0, \frac{dL}{du(T)}$

$\frac{dL}{dT} = \frac{dL}{du(T)} f_\theta(T, u(T))$ ▷ Gradiens T szerint

$s_0 = [u(T), \frac{dL}{du(T)}, 0_{|\theta|}, -\frac{dL}{dT}]$ ▷ Kezdeti értékek definiálása

def $aug\_dynamics([u(t), a(t), \cdot], t, \theta)$: ▷ Dinamika definiálása

  return $[f_\theta(u(t)), t), -a(t)^T \frac{\partial f}{\partial u}, -a(t)^T \frac{\partial f}{\partial \theta}]$ ▷ Vektor-Jacobi számolás

$[u(0), \frac{\partial L}{\partial u(0)}, \frac{\partial L}{\partial \theta}] = ODESolve(s_0, aug_dynamics, T, 0, \theta)$ ▷ Vissza idejű megoldás

return $\frac{\partial L}{\partial u(0)}, \frac{\partial L}{\partial \theta}$ ▷ Gradiensek

A gépi tanulási módszerek egyik legjobban teljesítő eszközei a neurális hálók. Nagy pontosság érhető el velük a legtöbb területen. Alkalmazzák őket például klasszifikációs feladatokban, képfeldolgozásban és természetes nyelvfeldolgozásban. Egyik hátránya a hosszú tanulítási idő. A legelterjedtebb tanítási módszer a gradiens módszer, amely a hibavisszaterjesztéses algoritmust használja a veszteségfüggvény gradiensének kiszámításához. Ez a módszer a rétegeken visszafele haladva szekvenciálisan határozza meg a megfelelő parciális deriváltakat.

A fejezetben megismerkedtünk a neurális differenciálegyenletek fogalmával, melyek tekinthetők egy folytonos mélységű neurális hálónak. A veszteségfüggvény optimalizálásban az adjungált érzékenység módszerén túl az MGRIT algoritmus választása ígéretes lehet, mert a segítségével a megfelelő kezdetiérték feldatok megoldása párhuzamosítható, így megfelelő környezet esetén a tanítás folyamata gyorsabbá tehető.

4. Kísérletek és további tervek

A szakdolgozatomban az itt megkezdett témát fejezem be, ezért szeretnék pár mondatot írni a további munkáról. Az XBraid szoftver a MGRIT módszert implementálja, melyet

az amerikai Lawrence Livermore Nemzeti Laboratórium fejleszt [6]. A csomag egy publikus GitHub repozitóriumban elérhető. A beszámoló írásának a pillanatáig az ebben a mappában található példafeladatokat sikerült lefuttatni az ELTE GPU szerverén. Következő lépésként szeretnénk kipróbálni az algoritmust a 2.4. alfejezetben látott homogén Dirichlet peremfeltételű hővezetési egyenleten. A célunk egy 1 ábrához hasonló grafikon elkészítése. Ezután az [1] cikkben látott eredményeket szeretnénk reprodukálni a MGRIT algoritmust használva mint differenciálegyenlet megoldó módszer (ODESolve).

A beszámolóm zárásaként szeretnék köszönetet mondani és hálámat kifejezni témavezetőm kollaborátorának, Sidafa Condénak, az amerikai Sandia Nemzeti Laboratórium munkatársának, aki az XBraid szoftver használatában nyújt technikai segítséget.

Hivatkozások

[1] R. T. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud: *Neural ordinary differential equations*, Advances in Neural Information Processing Systems 31 (NeurIPS 2018), pp. 6571-6583, 2018

[2] R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, and J. B. Schroder: *Parallel Time Integration with Multigrid*, SIAM J. Sci. Comput., 36 (2014), pp.C635-C661. LLNL-JRNL-645325.

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

[4] 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 J. Sci. Comput., 39 (2017), pp 298-322. LLNL-JRNL-692258

[5] J.-L. Lions, Y. Maday, and G. Turnici: *A parareal in time discretization of PDEs*, C.R. Acad. Sci. Paris, Serie I, 332 (2001), pp. 661-668.

[6] XBraid: Parallel multigrid in time. http://llnl.gov/casc/xbraid