Hatékony globális optimalizálási módszerek vizsgálata

Készítették:
Vinkó Tamás és Tóth Boglárka
V. éves programtervező matematikus hallgatók
SZTE Informatikai Tanszékcsoport

Témavezető: Dr. Csendes Tibor
egyetemi docens
SZTE Alkalmazott Informatika Tanszék

2000. április 13.

Köszönettel tartozunk Dr. Csendes Tibornak, a Szegedi Tudományegyetem docensének ötleteiért és a dolgozat megírásához nyújtott segítségéért. Köszönet illeti Baker Kearfott-ot és George Corliss-t, a GlobSol megismeréséhez nyújtott segítségükért, a sok elektronikus levélért; valamint Diamant Tibort és Csúri Miklóst a Szegedi Tudományegyetem Számítóközpontjának munkatársait, akik a program IBM RS/6000-es környezetben való használatát, illetve a GlobSol szerver működtetését lehetővé tették számunkra.

Tartalomjegyzék

1. Bevezetés **3**

2. Intervallumos globális optimalizálás **5**
2.1. Intervallum-aritmetika 5
2.2. Globális optimalizálás 7
2.3. Nemlineáris egyenletrendszerek 9

3. Egy konkrét optimalizáló rendszer **10**
3.1. A GlobSol alapvető felépítése 10
3.1.1. Kódlista 11
3.1.2. Slope mátrixok 12
3.1.3. Korlátozás és szétválasztás 13
3.1.4. Intervallumos lineáris egyenletrendszerek 15
3.1.5. Intervallum felezés 16
3.2. Gyorsító technikák 16
3.2.1. Intervallumos Newton-módszer 16
3.2.2. Korlátozó tesztek 18
3.2.3. Epszilon-kiterjesztés 19
3.2.4. Hámozás (peeling) 19
3.2.5. Feltétel propagáció 20
3.2.6. Prekondícionálók 21

4. Numerikus vizsgálatok **22**
4.1. Minimalizálás 23
4.2. Nemlineáris egyenletrendszerek 32
4.3. Dimenziószám vizsgálata 34
4.4. Összehasonlítás 36

5. Alkalmazások **38**
5.1. Körpakolás 38
5.2. Reaktorhálózat tervezési probléma 39
5.3. EX3 függvény 40
5.4. Nemdifferenciálható problémák 41

5.5. GlobSol szerver 42

6. Összefoglalás **44**

A. Installálás **48**

1. fejezet

Bevezetés

Az optimalizálás mind a matematika, mind az informatika számos területének részét képezi. Ez nagyrészt a sok gyakorlatban, illetve elméletben felmerülő probléma hatása, hiszen az élet legkülönfélébb helyein is optimalizálási problémákba ütközünk. Az optimalizálás ennélfogva rengeteg módszert, és eredményt foglal magában, ide tartoznak az operációkutatás, az analízis és a numerikus matematika bizonyos fejezetei.

Ennek a széles területnek több szempontból való osztályozásait ismerjük, a legfontosabb szűkítés talán az adott feladat folytonosságának feltétele, amivel kizárjuk a kombinatorikus és más diszkrét feladatok halmazát. Az optimalizálási feladatokat két csoportra oszthatjuk aszerint, hogy tartalmaznak-e korlátozó feltételeket, vagy sem. Dolgozatunkban mindkettőre láthatunk majd példát. A korlátozó feltételeket tartalmazó problémák esetén beszélhetünk lineáris, illetve konvex problémákról, ezekben az esetekben speciális algoritmusokat ismerünk az optimum megtalálására. Itt olyan algoritmusokkal dolgoztunk, amelyek nem használják ki a korlátozó feltételek semmilyen speciális tulajdonságát.

Egy következő fontos szempont az optimalizálásban, hogy globális vagy lokális megoldást keresésünk-e az adott problémára. Bizonyos, valós életből vett problémák esetén elegendő egy lokális optimum megtalálása, amelyhez elég jó approximációs algoritmusok léteznek. A másik oldal, a globális megoldások keresése ennél lényegesen átfogóbb feladat, ennek kiszámítása NP-nehéz probléma. Tulajdonképpen átfogalmazhatjuk a fealadatot az összes lokális optimum megkeresésére, ahonnan kiválaszthatjuk a globális optimumot, de ez az átfogalmazás nem javít a problémán. A lokális módszerek jól összehangolt változatai viszont lényegesen jobb algoritmusokat eredményeznek.

A globális optimum megtalálása azonban ilyen módszerek esetén sohasem garantált, hiszen minden esetben az adott lokális eljárás csak lokális (csak helyileg érvényes) információkat képes felhasználni. Ahhoz, hogy megbízható eredményeket szolgáltassunk, olyan módszerekre van szükségünk, amelyek globális (tehát egész tartományra vonatkozik, mint pl. a Lipschitz konstans) információkat számolnak ki és használnak fel a keresés során.

Ilyen algoritmusokhoz vezetett az intervallumanalízis, aminek a segítségével elméletileg bármilyen pontos alsó, illetve felső korlátot tudunk adni az optimumra. Az erre támaszkodó módszerek tárháza folyamatosan bővül, így ezekenek a technikáknak a vizsgálata és javítása elméleti és gyakrolati jelentőséggel bír. Bizonyos módszerek a kapott megoldás helyességét is bizonyítják, így ezek alkalmazása a gyakorlatban is elterjedt.

Dolgozatunk első részében bevezetjük az intervallum-aritmetika alapjait, a globális optimalizálási feladatot és a nemlineáris egyenletrendszer által adott problémát, illetve ezek segítségével megmutatjuk az intervallumos módszerek struktúráját. Az ezt követő fejezetben bemutatunk egy konkrét optimalizáló rendszert, ennek felépítését, speciális tulajdonságait, és az ebben használt gyorsító eljárásokat, módszereket. Az itt bevezetett módszereket megvizsgáljuk az irodalomban jól ismert teszt problémákra, hogy a numerikus eredmények alapján jó elemzést tudjunk adni a módszerekre. Vizsgálataink célja, hogy meghatározzuk, hogy az egyes eljárások milyen probémák esetén segítik a fő algoritmust, a módszerek által megnövekedett számításigény kifizetődő-e a kapott eredmény szempontjából, illetve hogy megállapításokat tegyünk arra, hogy melyek azok a módszerek, amelyek együtt alkalmazandók, és mik azok, amik gyengítik egymás hatását.

Ezeket az eredményeket a későbbiekben szeretnénk felhasználni, hogy a vizsgált eljárásokat jól struktúráltan építsük be más rendszerekbe egy előfeldolgozó segítségével, amely garantálná, hogy azokat az eljárásokat hajtsuk végre, amelyek a legcélravezetőbbek.

2. fejezet

Intervallumos globális optimalizálás

Ebben a fejezetben definiáljuk azokat a fogalmakat, amelyeket dolgozatunkban felhasználunk. Először bevezetjük az intervallum-aritmetika fogalmait, valamint megemlítünk néhány ide vonatkozó alapvető állítást. A következő rész a globális optimalizálás alapfogalmait ismerteti, ahol felvetünk néhány általános problémát, amelyek mind megválaszolhatók az intervallumos világban megvalósított eljárásokkal. A fejezet utolsó részében az intervallumos lineáris egyenletrendszerek definícióit, illetve megoldási módszereit ismertetjük.

2.1. Intervallum-aritmetika

Az $X = [a,b] = \{x \in \mathbb{R}^n : a \leq x \leq b\}$ valós kompakt halmazt (n-dimenziós) intervallumnak hívjuk. Amennyiben a dimenziószám a szövegkörnyezetből egyértelmű, egyszerűen intervallumnak nevezzük. A definícióban a jelöli az intervallum *alsó korlátját*, b pedig a *felső korlátját*. A továbbiakban a valós kompakt intervallumok halmazára az $\mathbb{I}$ jelölést használjuk.

Definiáljuk az intervallumok hosszát, sugarát és középpontját egy adott $X = [a,b]$ intervallumra a következőképpen:

$$\begin{aligned} \mathrm{diam}(X) &= b - a, \\ \mathrm{rad}(X) &= \frac{b-a}{2}, \\ \mathrm{mid}(X) &= \frac{b+a}{2}. \end{aligned}$$

A valós számokon értelmezett négy alapművelet az intervallumokra is kiterjeszthető:

$$X \circ Y = \{x \circ y \mid x \in X, y \in Y\}.$$

Egy intervallumos művelet tehát azon valós számok halmazából áll, amelyek az X és Y által meghatározott összes valós számpáron elvégzett művelet eredményei. Legyenek

$X = [a,b], Y = [c,d]$ intervallumok. Felhasználva az alapműveletek folytonosságát és monotonitását, a következő formulákat kapjuk:

$$\begin{aligned} X+Y &= [a+c, b+d] \\ X-Y &= [a-d, b-c] \\ XY &= [\min\{ac,ad,bc,bd\}, \max\{ac,ad,bc,bd\}] \\ X/Y &= X[1/d, 1/c], \quad \text{ha } 0 \notin Y. \end{aligned}$$

Az így definiált műveletek egyik fontos tulajdonsága az *izotonitás* (azaz befoglaló tulajdonság), ami azt jelenti, hogy ha $X' \subset X$ és $Y' \subset Y$ teljesül, akkor abból $X' \circ Y' \subset X \circ Y$ következik. Láthatjuk, hogy az osztást nem értelmezzük abban az esetben, ha a 0 eleme az Y intervallumnak. Ennek a problémának az elkerülésére alkották meg a *kiterjesztett*, vagy más néven *Kahan-Novoa-Ratz aritmetikát*, ami sok esetben jól használható[1]. Az összeadás, kivonás és szorzás a természetes intervallumkiterjesztésnek megfelelő, az osztást pedig a következőképpen definiálták:

$$\frac{X}{Y} = \begin{cases} X[1/d, 1/c] & \text{ha } 0 \notin Y \\ [-\infty, \infty] & \text{ha } 0 \in X \text{ és } 0 \in Y \\ [b/c, \infty] & \text{ha } b < 0 \text{ és } c < d = 0 \\ [-\infty, b/d] \cup [b/c, \infty] & \text{ha } b < 0 \text{ és } c < 0 < d \\ [-\infty, b/d] & \text{ha } b < 0 \text{ és } 0 = c < d \\ [-\infty, a/c] & \text{ha } 0 < a \text{ és } c < d = 0 \\ [-\infty, a/c] \cup [a/d, \infty] & \text{ha } 0 < a \text{ és } c < 0 < d \\ [a/d, \infty] & \text{ha } 0 < a \text{ és } 0 = c < d \\ \emptyset & \text{ha } 0 \notin X \text{ és } 0 = Y \end{cases}$$

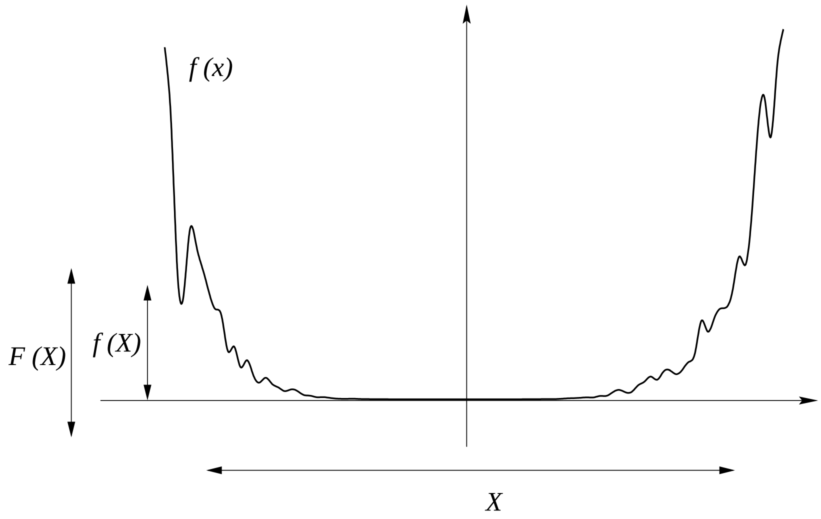
Az $f : S \subset \mathbb{R}^n \to \mathbb{R}$ függvény *befoglalófüggvénye* egy olyan $F : \mathbb{I}^n \to \mathbb{I}$ függvény, amely S minden X részintervallumához hozzárendel egy olyan $\mathbb{I}$-beli intervallumot, amelyre igaz, hogy $f(X) \subseteq F(X)$, ahol $f(X)$ az f függvény X feletti értékkészlete. Ilyen befoglalófüggvény az intervallum-aritmetika segítségével adható és szemléletesen az 2.1. ábrán látható egydimenziós esetben. A kapott befoglalás természetesen lehet tág is, amely finomításának egy lehetséges stratégiája az ún. középponti formulák alkalmazása [11].

Az intervallum-aritmetika egy korai kritikája az volt, hogy a befoglalás lehet túl széles is. Ez könnyen elő is fordulhat az intervallum-aritmetika naív használata esetén. Megvizsgálva és kihasználva azonban tulajdonságait, rendkívül jól használható és effektív algoritmusokat kaphatunk.

Legyen $F(X)$ egy intervallumos kiterjesztése az $f : \mathbb{R}^n \to \mathbb{R}^m$ függvénynek és legyen $F^u(X)$ az F értékkészlete X felett. Ha létezik olyan K konstans, amely független az X-től és amelyre

$$\operatorname{diam}(F(X)) - \operatorname{diam}(F^u(X)) \leq K \operatorname{diam}(X)^\alpha$$

[1]Problémát akkor okozhat, ha a végtelent mint a legnagyobb valós számot ábrázoljuk, de a legtöbb esetben ez elkerülhető.



2.1. ábra.
A befoglaló függvény elvének bemutatása az $f(x) = x^6(\sin\frac{1}{x} + 2)$ függvényen.

minden X-re, amelyre $\mathrm{diam}(X)$ elegendően kicsi és valamely $\alpha > 0$-ra, akkor azt mondjuk, hogy F egy α-rendű befoglalófüggvénye f-nek. Ha $\alpha = 1$ vagy 2, akkor *elsőrendű*-, illetve *másodrendű* befoglalófüggvénynek nevezzük. A definíciót felhasználva állíthatjuk, hogy a természetes intervallum kiterjesztés elsőrendű.

Tegyük fel, hogy $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ folytonosan differenciálható, $X \subset D$ és $x \in X$. Akkor az f függvény x középpontú X feletti *középponti kiterjesztése* az

$$F_2(X, x) = f(x) + \nabla F(X)(X - x)$$

képlettel definiálható, ahol $\nabla F(X)$ komponensenkénti intervallumos befoglalása a ∇f gradiensnek X felett.

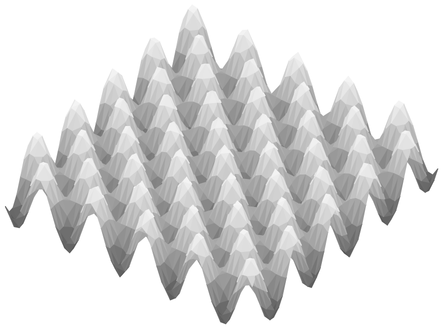
Ezen definíció alapján tegyük fel, hogy $\nabla F(X)$ komponensei a ∇f legalább elsőrendű intervallumos kiterjesztései. Akkor F_2 az f másodrendű kiterjesztése.

2.2. Globális optimalizálás

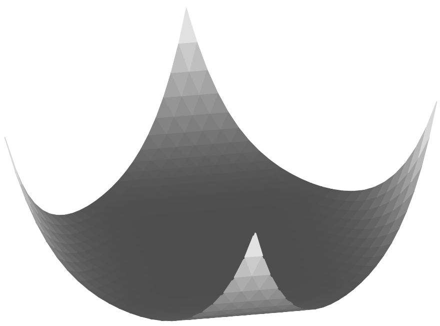
A globális optimalizálás általános feladatát a következőképpen fogalmazhatjuk meg. Legyen $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$ folytonos célfüggvény. A

$$\min_{x \in S} f(x)$$

globális optimalizálási problémát vizsgáljuk, esetleg lineáris vagy nemlineáris korlátozó feltételekkel kiegészítve, amelyek a lehetséges megoldások halmazát szűkítik. Ha nincsenek korlátozó feltételek, akkor természetesen $S = \mathbb{R}^n$, ellenkező esetben a feltételek



(a) Sok helyi minimumponttal rendelkező függvény.



(b) Nagy kiterjedésű minimumhely.

2.2. ábra. Kritikus függvények, amelyek minimalizálása valamely szempontból nehéz.

egy megszokott megadása az $S = \{\forall x \in \mathbb{R}^n \mid g_j(x) \leq 0 \quad j = 1, \ldots, m\}$, ahol m a korlátozó feltételek száma.

Egy megoldást *globálisnak* nevezünk, ha nem adható nála jobb, az esetleges korlátozási feltételeket is kielégítő megoldás a célfüggvény értelmezési tartományán. Ha ez a megoldásnak csak egy szűk környezetére teljesül, akkor a megoldást *lokálisnak* nevezzük.

Dolgozatunkban a feladatok megoldásához az f függvény intervallumos befoglalófüggvényét használjuk, ezért a kapott eredmények a megoldások egy intervallumos befoglalásai lesznek. A felhasznált technikák olyanok, hogy egyben bizonyítást is adnak a létezésre és egyértelműségre.

A következőkben néhány pontban áttekintjük az általános globális optimalizálás nehézségeit. Ezekre a vizsgálódásunk alapját képező program és az általa használt technikák segítségével megoldásokat találhatunk.

Lehetséges megoldás létezése. A korlátozó feltételeket tartalmazó feladatoknál problémát okozhat, hogy van-e lehetséges megoldás. Az intervallumos technikával ki tudjuk zárni azokat a részintervallumokat, amelyek garantáltan nem tartalmaznak megoldást, így szűkítve a keresés alaphalmazát.

Több kritikus lokális pont. Hogyan állapítsuk meg, hogy az a helyi minimum, amelyet megtaláltunk valójában a globális minimum? Egy olyan célfüggvényre, amelynek sok helyi minimumpontja van (lásd például a 2.2(a) ábrát), az intervallum-aritmetika segítségével garantáltan el tudjuk dönteni, hogy egy helyi minimum valójában globális minimum-e.

Szinguláris, vagy közel szinguláris megoldások. Az olyan célfüggvények, amelyek nagyon laposak vagy tartalmaznak ilyen részeket az optimumpont körül, gyakran szinguláris viselkedést mutatnak (lásd a 2.2(b) ábrát). Az intervallumos technika képes arra, hogy a megoldás pontok teljes halmazát befoglalja, ahol azok a globális minimum értékét elérik.

Nem pontos paraméterek. A valós életből vett problémák általában méréseken alapulnak. Ezen paraméter értékek alapján felállított modellek már eleve hibával terheltek, mivel nem tudunk pontosan mérni. Gyakran a pontos érték nem is adható meg. Ilyenkor ha a paramétereket befoglaló intervallumokkal adjuk meg, akkor az intervallumos módszerek ezek figyelembevételével a lehető legszűkebb befoglalását adják a lehetséges eredményeknek.

Magas dimenziószám. Ahhoz, hogy garantáljuk, hogy a globális optimum meghatározható lesz, a korlátozás és szétválasztás módszerét használjuk. A magas dimenziószám azonban nehézségeket okozhat. A legrosszabb esetben az algoritmus $\mathcal{O}(2^n)$ műveletigényű is lehet, ha a probléma n darab független változót tartalmaz. Számos esetben azonban lineáris algebrai eszközök segítségével ez a mérték csökkenthető. A dolgozatunkban tárgyalt GlobSol optimalizálóval eddig megoldott legnagyobb méretű probléma 128 dimenziós volt, itt azonban erősen kihasználták a rendszer ritkaságát, azaz azt a tényt, hogy a feladat nagyon sok paraméterének az értéke nulla volt. További megoldást jelenthet a párhuzamosítás, amelynek egy korai verziójú példáját már el is érhetjük az Interneten keresztül: `http://www.mscs.mu.edu/~globsol/ParaGlobSol.html`.

2.3. Nemlineáris egyenletrendszerek

Dolgozatunkban vizsgáljuk a nemlineáris egyeneletrendszerek gyökeinek meghatározására szolgáló módszereket is. A probléma általánosan a következőképpen fogalmazható meg. Legyen $g : \mathbb{R}^n \to \mathbb{R}^n$ vektorfüggvény. Keressük azon $x^* \in \mathbb{R}^n$ vektorok halmazát, amelyre $g(x^*) = 0$ teljesül. A probléma megoldásához a dolgozatban ismertetésre kerülő program a korlátozás és szétválasztás módszerét használja, kiegészítve számos technikával, amelyeket a következő fejezetben mutatunk be. Ezekkel az eszközökkel lehetőségünk van a gyökök megkeresésére és egyértelműségének bizonyítására.

3. fejezet

Egy konkrét optimalizáló rendszer

Ebben a fejezetben vizsgálataink alapját képező programot mutatjuk be, amely egy forráskód szinten szabadon hozzáférhető intervallumos globális optimalizáló és nemlineáris egyenletrendszert megoldó eszköz. Először az alapjait képező módszereket és technikákat ismertetjük, majd bemutatunk olyan gyorsító technikákat, amelyek empirikus vizsgálatainkhoz lesznek szükségesek.

3.1. A GlobSol alapvető felépítése

A dolgozatban tárgyalt és vizsgált technikákat a GlobSol nevű FORTRAN 90 nyelven írt program segítségével teszteltük, amit Baker Kearfott vezetésével fejlesztettek ki.

Ezzel megoldhatunk globális optimalizálási problémákat, amelyek esetleg egyenlőség vagy egyenlőtlenség formájában adott korlátozó feltételekkel adottak (lásd 2.2. alfejezet), illetve nemlineáris egyenletrendszereket is (lásd 2.3. alfejezet).

A GlobSol-ban a 2.1. fejezetben bevezetett intervallum-aritmetika megvalósítása az `INTERVAL_ARITHMETIC` modulban található. Ez a modul felhasználja az INTLIB csomagot [13], amely FORTRAN-77 nyelven írt standard műveletek és függvények intervallumos megvalósítása. A modulban megtalálható a négy alapművelet, a standard függvények és logikai műveletek megvalósításai, amelyeket önállóan is használhatunk.

A GlobSol két egymástól jól elkülönülő részt tartalmaz, a globális mimimalizálást és a gyökkeresést. A két eljárást külön kell indítanunk, és különböző outputot is kapunk. A minimalizálást a `find_global_min`, míg a gyökkeresést a `find_roots` program végzi. Teljesen nem különül el a kettő egymástól, vannak olyan eljárások, amelyeket mind a kettő használ, de természetesen nem mind ilyen. Az alapalgoritmus mindkét esetben egy korlátozás és szétvásztási algoritmuson alapszik (lásd 3.1.3. alfejezetet, ahol a minimalizálás esetét tárgyaljuk részletesebben). Ebbe épülnek bele az ismertetésre kerülő technikák, eljárások.

A GlobSol-ban a feladatok megoldása a következőképpen történik. Minimalizálás esetében a felhasználónak először Fortran 90 nyelven meg kell adnia a probléma definícióját. Itt külön adattípus létezik a változókra, a minimalizálandó célfüggvényre, valamint az egyenlőség- és egyenlőtlenség feltételekre, hogy azokat a program értelmezhesse. A megfelelően megadott forráskódból a GlobSol egy kódlistát, illetve gradiens kódlistát állít elő (lásd 3.1.1. részben), majd a `find_global_min` azok segítségével végez el műveleteket. Amennyiben a problémát sikeresen megoldja, egy fáljban nézhetjük meg a kapott eredményt.

Gyökkeresés esetén az előzőekhez hasonlóan járhatunk el, de ekkor természetesen a `find_roots` programot kell végrehajtanunk.

A GlobSol-ban lehetőségünk van arra, hogy a megvalósított és jelen dolgozatban tárgyalt gyorsító technikák közül megválasszuk azokat, amelyeket alkalmazni szeretnénk. Ezt egy `GlobSol.CFG` nevű standard konfigurációs állományon keresztül tehetjük meg. Az itt beállított értékeket használja a GlobSol mindkét fő modulja, azonban vannak olyanok melyek csak az egyikükre van hatással.

A továbbiakban részletesen kifejtjük a GlobSol további lényeges tulajdonságait: a kódlista előállítását és alkalmazását, az alkalmazott korlátozás és szétválasztás módszerének főbb lépéseit, valamint az intervallumos lineáris egyenletrendszerekkel és a slope mátrixokkal való kapcsolatát.

3.1.1. Kódlista

A minimalizálandó függvényről a program készít egy úgynevezett kódlistát, amelynek lényege, hogy a megadott függvényt alaputasításokra bontva szimbólikusan tárolja. Ez nagyon fontos, mert így a kódlista segítségével a már tárgyalt intervallum-aritmetika szabályai szerint kiszámíthatjuk a függvény befoglalófüggvényét vagy szimbólikus manipulációval hasznosíthatjuk az ismert képleteket. Másrészt, ez alapján automatikus deriválással készül a derivált kódlista is, amire például az intervallumos Newton módszer támaszkodik.

A kódlistára a 3.1. táblázatban láthatunk egy példát az $f(x, y) = x^2 + y^2$ függvény esetére kiegészítve a $3x + y \leq 6\sqrt{3}$ korlátozó feltétellel. Itt először feltüntettük a függvényből kapott kódlistát valamint a derivált kódlistát, illetve az ebben a példában szereplő összes művelethez tartozó kódszámot. Az (a) és (b) táblázatokban az utolsó két oszlop tartalmazza az egyes konstansok értékeit, amiket a GlobSol intervallumokként kezel. Tehát az itt feltüntetett $-6\sqrt{3}$ értéket nem ebben a formában, hanem a megfelelő lebegőpontos szám egy intervallumos befoglalásával kezeli.

A GlobSol-ban minden eljárás, amelyeket lebegőpontos vagy intervallumos függvények kiszámítására és automatikus deriválására használunk a kódlistával dolgozik. Ezek egy része az általános kódlistát, egy másik része pedig a derivált kódlistát használja. Megemlítjük továbbá, hogy a FORTRAN-90 nyelven megadott feladat kódlistára való fordítása után lehetőségünk van arra, hogy azt optimalizáljuk: az `optimize_codelist` nevű program kiiktatja a redundáns sorokat, amellyel hatékonyság- növekedést érhetünk el.

műv.kód	p	q	r	a	b
5	3	1	0	-	-
5	4	2	0	-	-
20	5	3	4	-	-
22	6	1	0	3	-
20	7	6	0	-	-
23	8	7	0	-	$-6\sqrt{3}$
18	5	0	0	-	-

(a) kódlista

műv.kód	p	q	r	a	b
5	3	1	0	-	-
5	4	2	0	-	-
20	5	3	4	-	-
22	6	1	0	3	-
20	7	6	2	-	-
23	8	7	0	-	$-6\sqrt{3}$
22	9	1	0	2	-
26	10	0	0	3	-
22	11	2	0	2	-
26	12	0	0	1	-
18	5	0	0	-	-
18	9	0	0	-	-
18	11	0	0	-	-

(b) derivált kódlista

műv.kód	művelet
5	$x_p = x_q^2$
18	x_p függő vált.
19	$x_p = x_q^r$
20	$x_p = x_q + x_r$
22	$x_p = ax_q$
23	$x_p = x_q + b$
26	$x_p = a$

(c) műveletek

3.1. táblázat.
Az $f(x,y) = x^2 + y^2$ függvényhez és a $3x + y \leq 6\sqrt{3}$ korlátozó feltételhez tartozó kódlista és derivált kódlista a műveletek kódjával.

3.1.2. Slope mátrixok

Ebben a fejezetben bevezetjük a Lipschitz mátrix és a slope mátrix fogalmát, amely a deriváltfüggvény intervallumos kiterjesztésére jobb korlátot ad.

1. Definíció. *Legyen $g : \mathbb{R}^n \to \mathbb{R}^m$ vektorfüggvény. Azt mondjuk, hogy az $\mathcal{A}$ mátrix a g-hez tartozó X feletti Lipschitz mátrix, ha minden $x \in X$ és $y \in Y$ esetén $g(x) - g(y) = A(x-y)$ teljesül valamely $A \in \mathcal{A}$ mátrixra.*

Valójában, ha $G : X \subset \mathbb{I}^n \to \mathbb{R}^m$, akkor a G Jacobi mátrixának komponensenkénti intervallumos kiterjesztése a G Lipschitz mátrixa lesz X felett.

2. Definíció. *Ha $G : X \subset \mathbb{I}^n \to \mathbb{R}^m$ alakú, akkor $G'(X)$ jelöl minden olyan intervallum mátrixot, melynek komponensei a G-hez tartozó Jacobi mátrix megfelelő komponenseinek természetes intervallumos kiterjesztései X felett.*

Az ilyen tulajdonságú mátrixokat Jacobi mátrixnak nevezzük.

Nézzünk meg egy ezzel felcserélhető definíciót, ami gyengébb tulajdonságra támaszkodik.

3. Definíció. *Legyen $g : \mathbb{R}^n \to \mathbb{R}^m$. Azt mondjuk, hogy az $\mathcal{A}$ márix egy g-hez tartozó X feletti intervallumos slope mátrix az $\check{X}$ középponttal, ha minden $x \in X$ és $\check{x} \in \check{X}$ esetén*

$$g(x) - g(\check{x}) = A(x - \check{x})$$

valamely $A \in \mathcal{A}$ mátrixra.
A legkisebb olyan mátrixot, ami ezt a tulajonságot kielégíti, jelöljük $S^u(g, X, \check{X})$-al. Az összes olyan intervallumos mátrixot ami tartalmazza $S^u(g, X, \check{X})$-ot jelöljük.

Gyakran, bár nem mindig, $\check{X}$ egy pont, vagy egy nagyon kicsi intervallum.

Az automatikus deriválás felhasználásával a slope befoglaló kiszámítása nem nehezebb, mint a derivált befoglalójának kiszámítása és gyakran jobb korlátot is ad. Slope mátrixok kiszámítására több algoritmus is ismert, a GlobSolban az ún. Hansen-féle slope technikával valósították meg. Ennek lényege, hogy szétválasztja a függvény változását a koordináták szerint, a középértéktétel alkalmazásával.

3.1.3. Korlátozás és szétválasztás

A GlobSol alapszerkezete a korlátozás és szétválasztás elvén alapszik, ennek különböző megvalósításait is megtalálhatjuk benne. A globális minimalizálásnál használt algoritmus általános szerkezetét tárgyaljuk a továbbiakban. A legfőbb cél, hogy pontos és szűk korlátot adjunk az optimumra, és kizárjuk az optimumot nem tartalmazó részeket. Ehhez három intervallumlistát használunk:

- $\mathcal{L}$ a még nem vizsgált intervallumok listája;
- $\mathcal{R}$ olyan részintervallumok, amelyek bizonyítottan tartalmaznak lokális minimumot;
- $\mathcal{U}$ olyan részintervallumok, amelyekről nem tudjuk biztosan, hogy tartalmaznak-e lokális minimumot, vagy sem;

Hogy találjunk egy nagyon szűk intervallumot az x^* minimumpontra és az f^* minimumértékre, amelyekre $f^* = f(x^*) \le f(x)$ minden $x \in X_0 \subset \mathbb{R}^n$ esetén, hajtsuk végre a következő algoritmust:

Inicializáljuk az $\mathcal{L} = \{X_0\}$ intervallumlistát
Legyen $\bar{f} = +\infty$ // ebben tároljuk az aktuális felső korlátot; $f^* \le \bar{f}$
while $\mathcal{L}$ nem üres **do**

$X^{(akt)} = \{$a következő intervallum az $\mathcal{L}$ listában$\}$
töröljük $X^{(akt)}$-ot az $\mathcal{L}$ listából
Tesztek segítségével próbáljuk bizonyítani, hogy nem tartalmaz globális minimumot:
Középponti teszt:
if $\bar{f} < \underline{F}(X^{(akt)})$ **then**
  dobjuk el az $X^{(akt)}$ intervallumot
  continue while
else
  próbálunk javítani az $\bar{f}$ korláton:
  $\overline{F} = \min\{\bar{f}, \overline{F}(X)\}$
end if
Monotonitási teszt:
if $0 \notin \nabla F(X)$ **then**
  dobjuk el az $X^{(akt)}$ intervallumot
  continue while
end if
Konkavitási teszt:
if $\nabla^2 F$ mátrix X-re nem pozitív definit **then**
  dobjuk el az $X^{(akt)}$ intervallumot
  continue while
end if
if az intervallumos Newton módszerrel bizonyítani tudjuk egyetlen helyi minimum létezését az $X^{(akt)}$ intervallumban **then**
  összehúzzuk az $X^{(akt)}$ intervallumot a helyi minimum körül a Newton módszer segítségével, és hozzáadjuk az $\mathcal{R}$ listához
  continue while
end if
if az intervallumos Newton módszer bizonyítja, hogy nem tartalmaz globális minimumot **then**
  dobjuk el az $X^{(akt)}$ intervallumot
  continue while
end if
Nem tudtunk semmit megállapítani:
if az $X^{(akt)}$ intervallum elég kicsi **then**
  hozzáadjuk az $X^{(akt)}$ intervallumot az $\mathcal{U}$ listához
else
  felosztjuk az $X^{(akt)}$ intervallumot részintervallumokra és az $\mathcal{L}$ listához fűzzük azokat.
end if
end while

A tényleges kód ennél több részletben bonyolultabb, de a fő elv ezzel megegyező.

Amennyiben korlátozott a feladat, az algoritmus elején tesztelnünk kell, hogy az intervallumunknak van-e olyan része, amely kielégíti a feltételeket. Ebben az esetben az $\bar{f}$ javításánál, illetve a Newton-módszernél is vigyáznunk kell, hogy a korlátozó feltételeknek megfelelően járjunk el. Az itt leírt tesztek csak a legalapvetőbbek, ezekről, illetve más nem mindenütt alkalmazható tesztekről további részleteket a 3.2.2. részben, illetve a [17] könyv 145–149. oldalain találhatunk.

3.1.4. Intervallumos lineáris egyenletrendszerek

Az intervallumos lineáris egyenletrendszerek nagyon fontosak az intervallumos Newton-módszerben, mert ez az elsődleges eszköz a gyökök és kritikus pontok létezésének és egyértelműségének igazolására.

Tegyük fel, hogy a $g : \mathbb{R}^n \to \mathbb{R}^n$ függvény $x^* \in X \subset \mathbb{R}^n$ zérushelyét keressük. Legyen $x \in X$. A középérték tételből következik, hogy

$$0 = g(x^*) = g(x) + A(x^* - x)$$

teljesül valamely $A \in S(f, X, x)$-re. Ezért $x^* - x$ az

$$A(X - x) = -g(x) \tag{3.1}$$

rendszer megoldáshalmazában van. Ezért g bármely $x^* \in X$ gyökét tartalmazza $x+\tilde{X}$ is, ahol $\tilde{X}$ egy korlátozó intervalluma a 3.1 rendszer megoldáshalmazának. Ezen megoldáshalmaznak a korlátját az intervallumos Gauss-Seidel módszerrel vagy az intervallumos Gauss eliminációval határozhatjuk meg.

Az intervallumos lineáris egyenletrendszerek általános formája a következő:

$$\mathcal{A}X = \mathcal{B}, \tag{3.2}$$

ahol $\mathcal{A}$ és $\mathcal{B}$ intervallumos mátrixok. Az egyenletrendszer megoldására használt intervallumos Newton-módszernél (lásd később) általában az

$$\mathcal{A}(X - x) = \mathcal{B}_0$$

formát használjuk, ahol $x \in \mathbb{R}^n$ valós vektor és $\|w(\mathcal{B}_0)\|$ rendszerint kicsi.

Azt mondjuk, hogy az $\mathcal{A}$ intervallumos mátrix *reguláris*, ha minden $A \in \mathcal{A}$ mátrix nemszinguláris. A regularitás fontos szerepet játszik a megoldások egyértelműségének bizonyításában. Egy intervallumos lineáris egyenletrendszer megoldásainak halmaza akkor és csak akkor korlátos, ha $\mathcal{A}$ reguláris.

A $\Sigma(\mathcal{A}, \mathcal{B}) \in \mathbb{R}^n$ halmazt a 3.2 egyenletrendszer megoldásának nevezzük, ha teljesül, hogy ha $X \in \Sigma(\mathcal{A}, \mathcal{B})$, akkor léteznek olyan $A \in \mathcal{A}$ és $B \in \mathcal{B}$ mátrixok, hogy $AX = B$. Általában $\Sigma(\mathcal{A}, \mathcal{B})$ egy konkáv poligon.

A $\Sigma(\mathcal{A}, \mathcal{B})$ halmaz általános értelemben vett meghatározása NP-teljes probléma [26], azonban lehet speciális szerkezetű is, amikor viszonylag hatékony megoldást is adhatunk.

Dolgozatunkban ismertetésre kerülő programban az intervallumos lineáris egyenletrendszerek megoldására két módszert találhatunk: az intervallumos Gauss eliminációt és az intervallumos Gauss-Seidel módszert. Általában szükséges, hogy a 3.2 rendszert prekondícionáljuk egy $Y \in \mathbb{R}^{n\times n}$ valós mátrixszal ahhoz, hogy a módszerek hatékonyak legyenek. Így az eljárások tulajdonképpen az $Y\mathcal{A}X = Y\mathcal{B}$, azaz

$$\mathcal{M}X = \mathcal{C}$$

rendszerrel dolgoznak, ahol $\mathcal{M} = YA$ és $\mathcal{C} = Y\mathcal{B}$. Itt az Y-t úgy választjuk, hogy az $\mathcal{M}X = \mathcal{C}$ megoldáshalmazának korlátozása könnyebb legyen.

A prekondícionálás általában megnöveli az aktuális megoldáshalmaz méretét:

$$\Sigma(\mathcal{A}, \mathcal{B}) \subseteq \Sigma(\mathcal{M}, \mathcal{C}) = \Sigma(Y\mathcal{A}, Y\mathcal{B}).$$

Az intervallumos Gauss elimináció teljesen hasonlóan működik, mint a jól ismert valós megfelelője. Az intervallumos Gauss-Seidel módszer elindításához kell egy kiindulási intervallum, de ezzel gyakran a megoldáshalmaz élesebb befoglalását tudjuk megadni. Az utóbbi módszer koordinátáról koordinátára dolgozik, ezért a megfelelő koordinátát tekinteve is élesebb befoglalást tud adni, amikor lehetséges, hogy az intervallumos Gauss elimináció elakad, például ha az $\mathcal{A}$ mátrix tartalmaz szinguláris mátrixot.

3.1.5. Intervallum felezés

Az intervallum felezés az $\mathcal{L}$ listán dolgozik és egy intervallumot két kisebb intervallumra osztja fel. Ezzel az intervallumok kiterjesztésekor keletkező túlbecsléseket csökkenthetjük vagy lehetővé tesszük olyan kisebb intervallumok vizsgálatát, melyekben az intervallumos Newton-módszer már konvergálhat. Az általános algoritmus mindig annál a dimenziónál felezi az intervallumot, amelyik a legszélesebb. Könnyen látható, hogy erre az eljárásra a legkülönfélébb módszerek dolgozhatóak ki, például Ratz állítása szerint sokkal jövedelmezőbb, ha a

$$|\nabla f(X)| \mathrm{diam}(X - x)$$

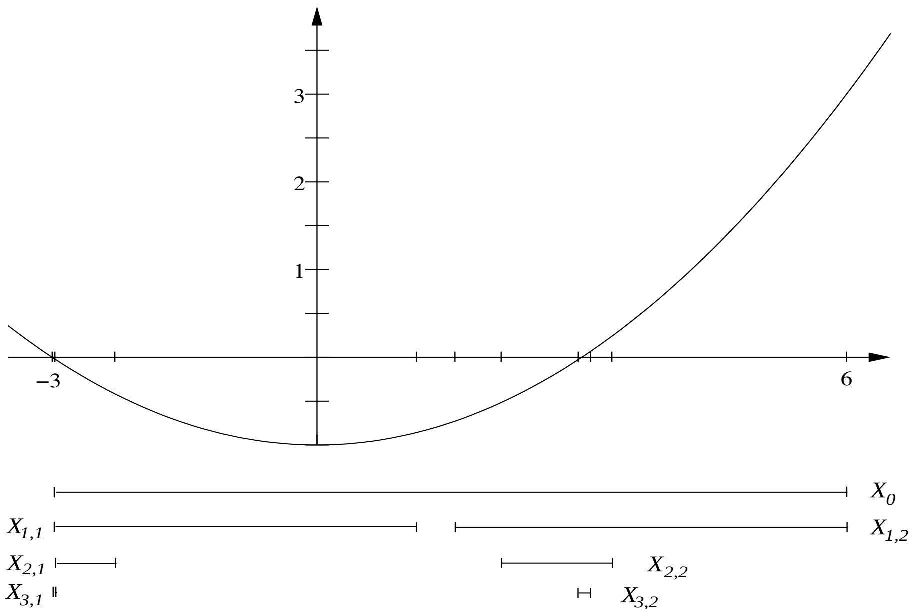
méretének megfelelően választjuk a dimenziót. Sokan vizsgálták a több részre osztás (multisection) esetét is, erről olvashatunk [4]-ben bővebben.

A GlobSol által használt algoritmus mégis az egyszerű intervallumfelezési eljárást használja. Az intervallum felezés nemlineáris egyenletrendszerekre való alkalmazását a [20] könyv 78–81. oldalán találhatjuk, az optimalizálás esetét pedig a [25] cikkben.

3.2. Gyorsító technikák

3.2.1. Intervallumos Newton-módszer

A numerikus matematikából jól ismert Newton-módszer – amelyet nemlineáris egyenletek, illetve egyenletrendszerek megoldására használhatunk – az intervallumos világba is



3.1. ábra.
Az $f(x) = \frac{1}{9}x^2 - 1$ függvényre az intervallumos Newton-módszer eredménye.

átvihető. A hagyományos algoritmusra jellemző erősség, a négyzetes helyi konvergencia, itt is érvényben marad. Egyváltozós esetben az átírás meglehetősen egyszerű, itt is a középérték tételből indulunk ki, azaz ha f kétszer folytonosan deriválható egy X intervallumon, és létezik olyan $x^* \in X$, hogy $f(x^*) = 0$, akkor bármely $\check{x} \in X$-re

$$0 = f(x^*) = f(\check{x}) + f'(\xi)\,(x^* - \check{x})$$

valamely $\xi \in X$-re. Vagyis ha $f'(\xi)$ helyett a derivált befoglaló függvényével számolunk az X intervallumon, akkor

$$x^* \in \check{x} - f(\check{x})/F'(X) \quad \text{minden} \quad \check{x} \in X$$

Hasonlóan, ha $\mathbf{S}(f, X, \check{x})$ f-nek egy slope mátrixa $\check{x}$ középponttal (ahogy azt a 3. definícióban megadtuk), akkor

$$x^* \in \check{x} - f(\check{x})/\mathbf{S}(f, X, \check{x}) \quad \text{minden} \quad \check{x} \in X$$

Ezek segítségével jelölje általánosan a kapott intervallumot

$$N(f; X_k, \check{x}_k) = \check{x}_k - \frac{f(\check{x}_k)}{F'(X_k)}, \qquad \text{vagy} \qquad N(f; X_k, \check{x}_k) = \check{x}_k - \frac{f(\check{x}_k)}{\mathbf{S}(f, X_k, \check{x})}$$

Ez az intervallum nem feltétlenül része a kiinduló intervallumnak, ezért úgy kapjuk meg az iteráció következő elemét, ha vesszük ezek metszetét.

$$X_{k+1} = N(f; X_k, \check{x}_k) \cap X_k,$$

ahol $\breve{x}_k$ jelölje az X_k intervallum középpontját. Említésre méltó az a tétel, miszerint ha $N(f; X, \breve{x}) \subseteq X$, akkor az X intervallumban pontosan egy zérushely van.

Nézzünk erre egy egyszerű példát. Legyen $f(x) = \frac{1}{9}x^2 - 1$, és keressük a gyökeit az $X_0 = [-3, 6]$ intervallumon. Ekkor $\breve{x}_0 = 1.5$, valamint a derivált legszűkebb befoglalófüggvényével, az $F'(X) = \frac{2}{9}X$-el számolva $F'(X_0) = [-4, 8]$. Könnyen átgondolhatók a következő intervallumos számítások, melyek részletes definícióit megtalálhatjuk az [2] könyvben:

$$\begin{aligned} N(f; X_0, \breve{x}_0) &= 1.5 - \frac{-0.75}{[-0.667, 1.333]} = 1.5 - ([-\infty, -0.0625] \cup [0.375, \infty]) \\ &= ([-\infty, 1.125] \cup [1.5625, \infty]) \end{aligned}$$

$$N(f; X_0, \breve{x}_0) \cap X_0 = [-3, 1.125] \cup [1.5625, 6] = X_{1,1} \cup X_{1,2}$$

$$\begin{aligned} N(f; X_{1,2}, \breve{x}_{1,2}) &= 3.781 - \frac{0.5887}{[0.347, 1.333]} = 3.781 - [0.442, 1.696] \\ &= [1.516, 2.285] \end{aligned}$$

$$N(f; X_{1,2}, \breve{x}_{1,2}) \cap X_{1,2} = [2.086, 3.340] = X_{2,2}$$

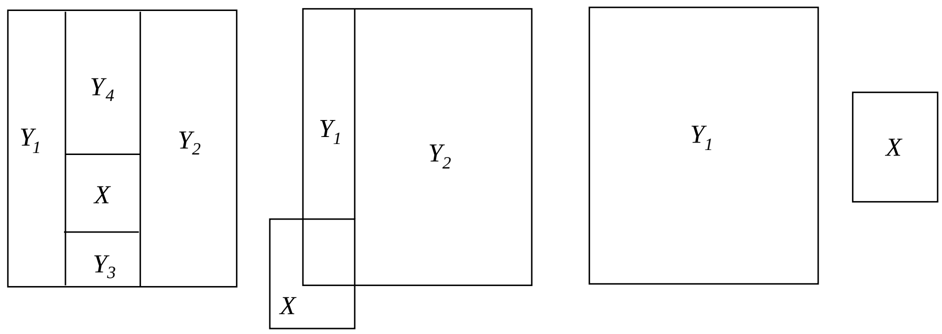
A számítások további eredményeit lásd a 3.1. ábrán. Többváltozós esetben számos változat elképzelhető arra vontkozóan, hogy a kapott lineáris rendszer megoldáshalmazát hogyan korlátozzuk. Az algoritmust itt felhasználhatjuk a megoldás létezésének és *egyértelműségének* bizonyítására. A módszerről bővebben az [12, 23] könyvekben olvashatunk.

3.2.2. Korlátozó tesztek

A korlátozás és szétválasztás algoritmusának végrehajtása során két alapvető tesztet láthattunk, a monotonitási tesztet és a konkativitási tesztet. Ezek nagyon kézenfekvő gondolatokon alapulnak. A monotonitási teszt lényege, hogy ha az adott intervallum gradiensvektora nem tartalmazza a 0-t, akkor biztosan tudjuk, hogy az adott intervallumban a függvény minden irányban szigorúan monoton nő, vagy csökken. Ilyen esetben az intervallumban illetve annak egy kis környezetében nem lehet szélsőérték, így ezt az intervallumot eldobhatjuk.

A konkativitási teszt kizárja a minimum létezését, ha a Hesse-mátrix seholsem pozitív definit. Ez egydimenziós esetben azzal egyenértékű, hogy ha a második derivált az intervallumban kisebb mint 0, akkor a függvényünk konkáv, tehát csak maximumot vehet fel az adott intervallumban, minimumot soha.

Mindkét tesztnél, ha korlátozó feltételek adottak, figyelnünk kell, nehogy az intervallum határát adja a korlátozó feltétel, mert ilyen esetben az intervallum határán lehet a megoldás. Ilyenkor az intervallum eldobása helyett, azt összehúzzuk a minimális határra.



3.2. ábra. A komplementerképzés három esete.

3.2.3. Epszilon-kiterjesztés

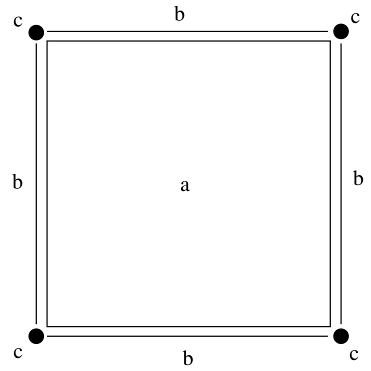
Ezt a módszert gyökök keresésekor érdemes használni. Ha van egy olyan intervallumunk, ami az eddigi tesztek alapján tartalmaz gyököt (esetleg gyököket), akkor először egy helyi keresővel próbálunk keresni egyet. Amennyiben egy jó közelítéshez jutottunk, akkor meghatározzuk azt a legkisebb intervallumot, ahol a létezése már igazolható. Ha tudjuk igazolni az egyetlenségét is az adott részintervallumban, akkor meghatározunk egy olyan, lehetőleg nagy intervallumot a gyök körül, hogy a gyök egyértelműségét még meg tudjuk mutatni az intervallumon belül [18].

Ez tulajdonképpen egy iteratív eljárás, amely a közelítő gyök egy kis környezetével kezdve, lépésenként ε-nal növelve az intervallum szélességét, igazolja a megkövetelt létezést és egyértelműséget addig, ameddig ez lehetséges. Természetesen, ha a létezés egyszer már igazolt, akkor az újabb, megnövelt intervallumra elegendő megmutatni, hogy másik gyök nem létezhet ezen belül. Az iterációt addig folytatjuk, amíg az egyértelműség megmutatható. Ha már nem tudjuk tovább növelni az intervallumot, akkor vegyük ki ezt a keresési térből, megkönnyítve a megmaradó intervallumok eliminálását.

Az adott intervallumot úgy vetjük el, hogy képezzük a komplementerét a keresési térre nézve. Ez egy intervallum listát hoz létre, aminek az uniója adja ki a megmaradt keresési tartományt. A használt $\mathcal{O}(n)$ műveletigényű algoritmus $2n$ darab új intervallumot hozhat be legfeljebb. Amennyiben a keresési tér már egy intervallum listával volt megadva, ez az intervallumszám-növekedés nem jelentős.

3.2.4. Hámozás (peeling)

Ha a problémánk olyan, hogy alsó- felső-korlátok is adottak, akkor a program egy ún. hámozási technikát (peeling) [15] alkalmaz arra, hogy a feltételek által meghatározott tartományból kiválogassa az összes lehetséges alacsonyabb dimenziós részt. Az így kapott intervallumok azután ugyanúgy feldolgozásra kerülnek, azonban az alacsonyabb dimenziószám miatt várhatóan kevesebb műveletet igényelnek. Ezen kívül ha a globális optimumra van egy jó felső korlátunk, akkor ezzel az eljárással a hámozás során keletkező intervallumok nagy része eldobható.



3.3. ábra. Egy kétdimenziós intervallum hámozása.

A kétdimenziós esetre a 3.3. ábrán láthatunk példát. Itt összesen 9 új intervallumot kapunk: egyet, amely $\mathbb{R}^2$-beli, négyet, amelyek $\mathbb{R}$-beliek, és négyet, amelyek diszkrét pontok. Ezeket az ábrán rendre az a, b és c betűk jelölik.

3.2.5. Feltétel propagáció

A kódlista sok információt tartalmaz egy adott intervallum-kiterjesztés kiértékelése alatt keletkező közbülső mennyiségek közötti kapcsolatról. Ez a kapcsolat néha használható arra, hogy csökkentsük a túlbecslések mértékét az ilyen intervallumos kiterjesztésekben, vagy hogy segítsen a nemlineáris egyenletrendszerek megoldásában.

Tekintsük a következő példát [14] [31]. Legyen $f : \mathbb{R}^2 \to \mathbb{R}^2$ a következőképpen definiálva:

$$\begin{aligned} f_1(x_1, x_2) &= x_1^3 + x_1^2 x_2 + x_2^2 + 1 \\ f_2(x_1, x_2) &= x_1^3 - 3x_1^2 x_2 + x_2^2 + 1, \end{aligned}$$

az $X = ([-2, 0], [-1, 1])^T$ kiindulási intervallummal és $x = (-1, 0)^T$ kiindulási hellyel. Az egyenletrendszernek pontosan egy megoldása van X-ben és ez a $(-1, 0)^T$ kiindulási pont. A rendszer azonban olyan, hogy a kifejezések között nagyszámú függőség van, ami a függvény és deriváltjának intervallumos kiterjesztésekor jelentős túlbecslést okoz. Például a Hansen-féle slope mátrix

$$S(f, X, x) = \begin{pmatrix} [0, 7] & [-1, 5] \\ [0, 7] & [-13, 1] \end{pmatrix}$$

lesz, így az tartalmazza a zéró mátrixot. Sőt, az $YS(f, X, x)$ is tartalmazza a zéró mátrixot bárhogyan is válsztjuk az Y prekondícionáló mátrixot. Mivel $\|f(x)\|$ kicsi, ezért az intervallumos Gauss-Seidel módszer nem lesz hatékony.

A kiterjesztett rendszerhez azonban egy speciális kódlista összetétel tartozik:

	v_1	$=$	x_1		v_4	$=$	x_2
(i)	v_3	$=$	v_1^2	(ii)	v_4	$=$	v_2^2
(iii)	v_5	$=$	$v_3 v_2$	(iv)	v_6	$=$	v_1^3
(v)	v_7	$=$	$v_6 + v_4 + 1$				
(vi)	$v_7 + v_5$	$=$	0				
(vii)	$v_7 - 3v_5$	$=$	0				

A kódlistában lévő információ az eredeti rendszerről szimbólikusan manipulálható, amivel a túlbecslések száma csökkenthető. Ezen kívül numerikus módszereket is alkalmazhatunk, amelyek közül kettőt tárgyalunk.

Figyeljük meg, hogy a fenti táblázatban minden egyenlet csupán egyetlen invertálható műveletet tartalamaz, ezért minden változó kifejezhető a többi közbülső változóval. Ezt az eljárást, amelyet *substitution/iteration* technikának nevezünk, addig folytathatjuk, amíg a közbülső változók minden értéke stacionáriussá nem válik.

A másik módszer egy intervallumos Newton módszer, amelyet azonban nem az eredeti egyenletrendszerre alkalmazunk, hanem a kiterjesztett renszerre (tehát a fenti táblázatra). Optimális prekondícionálóval kombinálva az intervallumos Newton módszer valószínűleg élesebb korlátot ad a kiterjesztett rendszerre.

3.2.6. Prekondícionálók

Mint azt már a 3.1.4. részben említettük, a lineáris egyenletrendszerek prekondícionálása bár a megoldáshalmaz növekedéséhez vezet, annak elvégzése szükséges és hatékony. Ebben a részben először az inverz középpontos prekondícionálót tárgyaljuk, amely általában használatos módszer.

Legyen $\mathcal{A}X = \mathcal{B}$ n-változós lineáris egyenletrendszer. Az inverz középpontos prekondícionálót az

$$Y^{mid} = \mathrm{mid}(\mathcal{A})^{-1}$$

összefüggéssel definiáljuk. Számos esetben amikor az Y^{mid} felhasználásával prekondícionált rendszer megoldáshalmazát kell meghatároznunk a számítások egyszerűsíthetőek [27].

A [10] cikkben, illetve a [12] könyvében Hansen kifejti, hogy az inverz középpontos prekondícionáló az intervallumos Gauss-eliminációhoz általában a legjobb választás. A gyakorlatban pedig az epszilon kiterjesztéssel kapcsolatos eljárásokban is nagy hasznát vehetjük.

Azonban például amikor a lineáris egyenletrendszer megoldása az intervallumos Newton-módszer egy lépése, más prekondícionálók használata jobb lehet. Erre egyik klasszikus példa a Brown-féle majdnem lineáris függvény, amelyet a 4. fejezetben vizsgálunk majd. A GlobSol-ban magvalósított C-LP prekondícionáló számos esetben jobb lehet, ennek részletezésére azonban itt nem térünk ki. További lehetőségekről, az ide vonatkozó elméleti munkákról a [17] könyv 3.2. fejezetében olvashatunk.

4. fejezet

Numerikus vizsgálatok

Ebben a fejezetben az elvégzett numerikus vizsgálatokat tárgyaljuk. Vizsgálatainkat a szakirodalomból vett összesen 30 darab tesztfüggvényen végeztük, melyek a következők:

Korlátozás nélküli feladatok. Shekel 5, 7, 10 (S5, S7, S10); Hartman 3, 6 (H3, H6); Goldstein-Price (GP); Six-Hump-Camel-Back (SHCB); Branin (BR); Rosenbrock (RB); Levy 3, 5, 12 (L3, L5, L12); Schwefel 2.1, 3.1, 2.5, 2.18, 3.2, 3.7 (Sch21, Sch31, Sch25, Sch218, Sch32, Sch37); Griewank 5, 10 (G5, G10); Ratz 4 (R4); EX2.

Korlátozó feltétellel megadott feladatok. UPP jelöli a [9] könyv 4.10. alfejezetében definiált korlátozó feltételekkel megadott egyváltozós polinomfüggvényt; qcp2 és qcp3 jelöli a [9] könyv 3.2. és 3.3. alfejezeteiben definiált kvadratikus függvényekkel korlátozott problémákat; MS pedig a [22]-ben ismertetett egyenlőség-feltételekkel korlátozott kétszer folytonosan deriválható célfüggvénnyel megadott feladatot.

Nemlineáris egyenletrendszerek. Itt inteq jelöli a [21] cikkben ismertetett egyenletrendszert, amely egy nemlineáris integrálegyenlet diszkretizálásából adódik; chemeq a [9] könyv 14.1.2. alfejezetében megtalálható kémiai egyensúly problémából adódó feladat (lásd még [19]); I1 jelöli a [32]-ben megadott egyenletrendszert; FT pedig a [9] könyv 14.1.4. alfejezetében leírt problémát jelöli (lásd még [8]) található.

A GlobSol egy konfigurációs fájlon keresztül számos korábban ismertetett gyorsítótechnika paraméter-beállításait, illetve ki/be kapcsolását teszi lehetővé. Ezáltal lehetőségünk van arra, hogy empirikus úton elemezzük azoknak a hatékonyságra gyakorolt hatását.

A fejezet felépítése a következő. Először minimalizálási feladatokat vizsgálunk a `find_global_min` felhasználásával, majd a nemlineáris egyenletrendszerek zérushelyeinek megadására szolgáló `find_roots` modult vizsgáljuk. Ezután a dimenziószám-növekedésre való érzékenységet elemezzük két skálázható feladaton keresztül, végül adunk egy összevetést egy másik intervallumos globális optimalizálóval.

4.1. Minimalizálás

Először a minimalizálási feladatokat tárgyaljuk, tehát a kapott eredményeket a `find_global_min` futtatásával kaptuk. Az általunk vizsgált beállítások eredményeit a 4.1 – 4.7. táblázatokban találhatjuk, ahol az egyes oszlopok jelentése a következő:

- probléma neve és dimenziószám;
- a bisec. oszlop az adott feladat megoldásakor alkalmazott intervallum felezések száma;
- nfe jelöli a függvénykiértékelések számát. Itt az outputban szereplő `No. dense interval residual evaluations - gradient code list` és a `Total number second-order interval evaluations of the original function` sorokhoz tartozó számértékek összegét tüntettük fel;
- az nge oszlopa a gradiens kiértékelések számát mutatja. Itt az outputból a `Number of gradient evaluations from a gradient code list` és a `Total number dense interval reduced gradient evaluations` sorokhoz tartozó számok összegét vettük figyelembe;
- a bp oszlopa a feldolgozott intervallumok számát adja meg;
- local mutatja azt, hogy a GlobSol a minimalizálás során hányszor alkalmazta a helyi keresőt;
- az stu oszlopában pedig az adott feladat megoldásához szükséges ún. standard időegységek számát tüntettük fel. Az stu kiszámítása a következő módon történt [7]. Mérjük le, hogy a Shekel-5 függvény $(4,4,4,4)^T$ pontban való 1000-szeres kiértékeléséhez hány másodperc kell. Ez az általunk használt Pentium-133 gépen Linux operációs rendszer alatt működő Fujitsu FORTRAN-90 fordítóval 0.00696 másodperc volt. A GlobSol outputjában megtaláljuk a feladat megoldásához felhasznált CPU időt, amelyet le kell osztani ezzel a számmal.

Először az alapbeállítással kapott eredményeket láthatjuk (4.1. táblázat), amelyet a GlobSol installálásakor rendelkezésre bocsátott konfigurációs fájl beállításait használva kaptunk. Ebben az esetben minden lehetséges és ígéretesnek mutatkozó gyorsító technika be van kapcsolva, valamint számos (a dolgozatban nem részletezett) paraméterérték egy általában hatékonynak feltételezett értékre van beállítva. Megjegyezzük, hogy az összes elvégzett beállításnál (kivéve egyetelen esetet) az $\varepsilon = 10^{-9}$ helyre vonatkozó toleranciaértékkel dolgoztunk. Minden további beállítás eredményességének vizsgálatához az itt kapott értékeket használjuk fel.

probléma	dim.	bisec.	nfe	nge	box_proc	local	stu
S5	4	9	273	89	17	7	515.80
S7	4	16	345	118	20	9	547.41
S10	4	17	359	123	21	10	693.97
H3	3	21	470	188	44	8	1 066.09
H6	6	42	1 744	490	90	88	10 436.78
GP	2	2 828	38 119	22 449	4 205	8	10 045.98
SHCB	2	44	720	366	72	6	283.05
BR	2	6	180	71	11	6	66.09
RB	2	13	234	108	16	4	25.86
L3	2	267	4 433	1 560	356	17	10 577.59
L5	2	62	861	366	100	3	1 951.15
L12	10	45	1 618	749	46	19	19 287.36
Sch21	2	10	215	81	28	6	97.70
Sch31	3	4	217	74	8	6	66.09
Sch25	2	0	127	25	3	9	12.93
Sch218	2	0	95	21	8	6	12.93
Sch32	3	1	139	47	5	5	30.17
Sch37	10	0	116	12	0	2	449.71
G5	5	0	67	8	0	2	199.71
G10	10	0	117	13	0	2	1 405.17
R4	5	35	679	331	63	8	541.67
EX2	5	132	3 028	1 799	216	33	21 876.44
UPP9	2	149	1 975	1 976	280	5	955.46
qcp2	5	0	537	232	46	45	517.24
qcp3	6	3 504	35 236	23 908	3 661	251	21 254.31
MS	5	244	3 478	4 129	331	18	4 718.39
inteq	15	0	326	15	0	2	146.55
chemeq	5	6 237	122 688	20 413	7 921	11	48 400.86
I1	10	641	12 242	1 997	694	9	16 392.24
FT	2	72	1 772	281	133	5	579.02

4.1. táblázat. Az alapbeállításokkal megadott eredmények.

A 4.2 táblázatban a intervallumos Newton-módszer kikapcsolásával kapott eredményeket találjuk. Itt megállapíthatjuk, hogy minden esetben rosszabb eredményt kaptunk. Egyetlen kivétel a Schwefel 2.1 függvény, ahol minimális teljesítmény javulás tapasztalható. Ebből arra a következtetésre juthatunk, hogy az intervallumos Newton-módszer használata feltétlen szükséges a hatékonyság érdekében.

A 4.3. táblázat tartalmazza a substitution/iteration technika kiiktatásával kapott eredményeket. Itt már minden oszlop mellett feltüntettük az alapbeállításhoz viszonyított százalékos arányokat. A kapott értékek alapján több problémára itt is jelentős teljesítmény csökkenést tapasztalhatunk. Ilyen eredményt az összes korlátozó feltételt

probléma	dim.	bisec.	nfe	nge	bp	local	stu
S5	4	139	1 688	776	242	7	1 097.70
S7	4	141	1 754	802	252	9	1 547.41
S10	4	144	1 799	822	259	10	2 050.29
H3	3	315	3 760	1 805	580	8	4 902.30
H6	6	906	11 306	5 112	1 599	87	34 989.94
GP	2	7 812	71 554	31 950	8275	8	17 586.21
SHCB	2	386	4 409	2 111	667	6	758.62
BR	2	239	2 241	981	249	5	520.11
RB	2	90	874	373	92	4	63.22
L3	2	1 048	11 543	5 313	1 591	17	19 422.41
L5	2	221	2 222	1 013	282	3	3 833.33
L12	10	268	3 071	1 159	269	19	24 114.94
Sch21	2	10	201	67	28	6	93.39
Sch31	3	307	2 920	1 260	311	6	550.29
Sch25	2	96	941	392	97	7	57.47
Sch218	2	288	3 099	1 441	440	6	239.94
Sch32	3	189	1 782	768	190	4	247.13
Sch37	10	0	116	12	0	2	449.71
G5	5	0	67	8	0	2	199.71
G10	10	0	117	13	0	2	1 400.86
R4	5	690	7 584	3 591	1 098	8	3 060.34
EX2	5	11 716	124 086	59 032	17 785	28	520 830.47
UPP9	2	16 447	155 164	69 344	20 000	4	32 882.18
qcp2	5	19 905	179 713	79 757	20 000	43	49 107.76
qcp3	6	19 892	180 633	79 799	20 000	244	68 317.53
MS	5	19 958	179 879	79 898	20 000	17	72 696.84
inteq	15	0	326	15	0	2	146.55
chemeq	5	19 457	353 802	78 389	20 000	11	26 057.47
I1	10	1 616	29 534	6 495	1 626	9	2 778.74
FT	2	544	10 116	2 219	582	5	1 297.41

4.2. táblázat.
Az intervallumos Newton-módszer használata nélkül kapott eredmények.

is tartalmazó függvény esetében kaptunk. Ezek alapján egyértelmű, hogy az ilyen típusú problémákra jelentős hatással van a substitution/iteration technika alkalmazása. További példa a Hartman-6, a Schwefel-2.1 és a Ratz-4 függvény. Kivételt képeznek a Schwefel 2.5, Schwefel 3.7, Griewank 5, Griewank 10, inteq és a chemeq feladatok, ahol ugyanazokat az eredményeket kaptuk, mint az alapbeállításnál. Minimális javulás figyelhető meg a Rosenbrock és az L12 függvények esetében. Észrevehetjük továbbá, hogy bizonyos feladatok esetén a műveletszám növekedése mellett a standard CPU idő mennyisége kevesebb. Itt arra következtethetünk, hogy bár a substitution/iteration

p	bisec.	%	nfe	%	nge	%	bp	%	stu	%
S5	23	255	388	142	141	158	24	141	382.18	74
S7	25	156	434	125	153	129	26	130	577.59	105
S10	25	147	440	122	153	124	26	123	767.24	110
H3	56	266	847	180	381	202	75	170	1 606.32	150
H6	352	838	5 191	297	2 178	444	443	492	24 077.59	230
GP	3 292	116	43 528	114	25 530	113	4 551	108	8 571.84	85
SHCB	50	113	816	113	431	117	72	100	162.36	57
BR	9	150	241	133	117	164	14	127	66.09	100
RB	12	92	230	98	112	103	14	87	21.55	84
L3	490	183	7 651	172	3 406	218	514	144	17 126.44	162
L5	157	253	1 774	206	854	233	160	160	3 705.46	189
L12	42	93	1 530	94	724	96	43	93	18 100.57	93
Sch21	93	930	1 533	713	921	1 137	138	492	231.32	238
Sch31	27	675	541	249	261	352	31	387	143.68	216
Sch25	0	100	127	100	25	100	3	100	10.06	83
Sch218	0	100	107	112	29	138	8	100	8.62	67
Sch32	7	700	206	148	93	198	10	200	40.23	133
Sch37	0	100	116	100	12	100	0	100	448.28	99
G5	0	100	67	100	8	100	0	100	199.71	100
G10	0	100	117	100	13	100	0	100	1 400.86	99
R4	114	325	1 672	246	783	236	156	247	877.87	162
EX2	128	96	3 260	107	1 977	109	230	106	22 290.23	101
UPP9	877	588	9 579	485	8 801	445	1 111	396	1 722.70	180
qcp2	32	-	1 150	214	1 352	582	80	174	1 022.99	197
qcp3	19 859	566	191 109	542	151 303	633	20 000	546	109 770.12	516
MS	910	372	11 971	344	15 002	363	1 089	329	12 135.06	257
inteq	0	100	326	100	15	100	0	100	145.11	99
chemeq	6 237	100	122 688	100	20 413	100	7 921	100	48 469.83	100
I1	914	142	17 210	140	2 825	141	976	141	20 811.78	127
FT	120	167	2 756	155	445	158	202	151	649.43	112
Σ	33 851	264	427 605	195	238 458	240	450	191	295 543.11	144

4.3. táblázat. A substitution/iteration kikapcsolásával kapott eredmények.

technika könnyíti a feladat megoldását, azonban gyakran ehhez több időre van szüksége. Érdekes lehet annak a kérdésnek a megválaszolása, hogy mely esetben érdemes ezt a technikát alkalmazni. Láttuk, hogy a korlátozott feladatokra ennek kikapcsolása nagyon rossz hatással volt, de vajon minden esetben igaz-e ez? A GlobSol alapjait lefektető könyvben erre a kérdésre nem kapunk teljes választ, ezért további kutatási céljaink egyikét ez képezi.

p	bisec.	%	nfe	%	nge	%	bp	%	stu	%
S5	9	100	255	93	80	89	10	58	280.17	54
S7	17	106	375	108	117	99	19	95	558.91	102
S10	18	105	374	104	122	99	21	100	721.26	104
H3	21	100	470	100	188	100	44	100	1 086.21	101
H6	56	133	1 948	112	581	118	118	131	11 377.87	109
GP	2 821	99	38 148	100	22 498	100	4 196	99	9 998.56	99
SHCB	44	100	712	99	362	99	72	100	238.51	100
BR	6	100	172	96	69	97	11	100	60.34	91
RB	13	100	219	93	101	94	17	106	24.43	94
L3	267	100	4 440	100	1 555	99	360	101	10 668.10	100
L5	62	100	861	100	366	100	100	100	1 951.15	100
L12	45	100	1 570	97	701	93	46	100	18 329.02	95
Sch21	7	70	176	82	60	74	22	78	54.60	56
Sch31	4	100	215	99	72	97	8	100	64.66	98
Sch25	0	100	127	100	25	100	3	100	11.49	89
Sch218	0	100	95	100	21	100	8	100	12.93	100
Sch32	1	100	139	100	47	100	5	100	30.17	100
Sch37	0	100	116	100	12	100	0	100	449.71	100
G5	0	100	69	103	10	125	0	100	204.02	102
G10	0	100	117	100	13	100	0	100	1 386.49	98
R4	36	102	701	103	336	102	64	101	533.05	98
EX2	132	100	3 028	100	1 799	100	216	100	21 765.80	99
UPP9	149	100	1 975	100	1 976	100	280	100	961.21	100
qcp2	0	100	537	100	232	100	46	100	517.24	100
qcp3	3 504	100	35 236	100	23 908	100	3 661	100	21 130.75	100
MS	244	100	3 478	100	4 129	100	331	100	4 688.22	99
inteq	0	100	326	100	15	100	0	100	147.99	100
chemeq	4 448	71	88 080	72	14 645	71	5 731	72	35 896.55	74
I1	618	96	11 780	96	1 920	96	663	95	15 188.22	92
FT	72	100	1 772	100	281	100	133	100	574.71	99
Σ	12 594	99	197 511	98	76 241	98	450	98	158 912.34	95

4.4. táblázat. Másodrendű kiterjesztés használata nélkül kapott eredmények.

A másodrendű kiterjesztés kiiktatásával kapott eredményeket a 4.4. táblázat mutatja. Itt megfigyelhetjük, hogy majdnem minden esetben jobb eredményeket kaptunk, illetve egyes feladatokra nincs hatással ezen művelet kiiktatása. Egyetlen kivétel a H6 függvény, ahol jelentős teljesítmény csökkenést állapíthatunk meg. Észrevehetjük továbbá, hogy a legkisebb négyzetes problémák közül a chemeq feladatnál sokkal jobb teljesítmény-mutatók adódnak. Ebből arra következtethetünk, hogy egy feladat megoldásának hatékonyságát általában nem befolyásolja ezen művelet megvalósítása.

p	bisec.	%	nfe	%	nge	%	bp	%	stu	%
S5	9	100	255	93	80	89	10	58	278.74	53
S7	17	106	375	108	117	99	19	95	553.16	101
S10	18	106	374	104	122	99	21	100	732.76	105
H3	21	100	470	100	188	100	44	100	1 083.33	101
H6	56	133	1 948	116	581	118	118	131	11 341.95	108
GP	2 821	99	38 148	100	22 498	100	4 196	99	9 988.51	99
SHCB	44	100	712	98	362	99	72	100	238.51	84
BR	6	100	172	95	69	97	11	100	67.53	102
RB	13	100	219	90	101	93	17	106	24.43	94
L3	267	100	4 440	100	1 555	99	360	101	10 632.18	100
L5	62	100	861	100	366	100	100	100	1 952.59	100
L12	45	100	1 570	97	701	93	46	100	18 212.64	94
Sch21	7	142	176	82	60	74	22	78	56.03	57
Sch31	4	100	215	99	72	97	8	100	64.66	96
Sch25	0	100	127	100	25	100	3	100	11.49	88
Sch218	0	100	95	100	21	100	8	100	12.93	100
Sch32	1	100	141	101	49	104	5	100	30.17	100
Sch37	0	100	116	100	12	100	0	100	448.28	99
G5	0	100	69	103	10	125	0	100	201.15	101
G10	0	100	119	102	15	86	0	100	1 405.17	100
R4	36	103	701	103	336	101	64	101	528.74	97
EX2	116	88	2 716	89	1 545	85	196	90	19 363.51	88
UPP9	154	103	2 044	96	2 035	103	291	104	948.28	99
qcp2	0	100	545	101	228	98	46	213	518.68	100
qcp3	3 341	95	32 981	93	22 052	92	3 498	95	19 396.55	91
MS	243	99	3 518	101	4 146	100	339	102	4 731.32	100
inteq	0	100	326	100	15	100	0	100	145.11	100
chemeq	5 760	92	113 490	92	18 876	92	7 338	92	44 123.56	91
I1	618	96	11 780	96	1 920	96	663	95	15 133.62	92
FT	72	100	1 734	98	274	97	127	95	491.38	84
Σ	7 971	96	106 852	92	59 534	91	450	95	118 580.47	88

4.5. táblázat. Az $\varepsilon = 10^{-6}$ beállítással kapott eredmények.

A 4.5. táblázatban felsorolt eredményeket az $\varepsilon = 10^{-6}$ helyre vonatkozó tolerancia érték beállításával kaptuk. Jelentős teljesítménycsökkenés csak a Hartman-6 függvény esetén jelentkezik, viszont az EX2 függvényre jobb eredményeket kaptunk. Itt azonban figyelembe kell venni azt is, hogy milyen pontosságot tudunk így elérni. Ennek vizsgálatához a 4.1. táblázatban feltüntettük az egyes feladatokra az eredeti beállítással és a $\varepsilon = 10^{-6}$ beállítással kapott helyes jegyek számát. Ennek kiszámítása a következő. Az $\tilde{x}$ közelítő szám értékes jegyeinek a száma $-\log_{10}|\frac{x^*-\tilde{x}}{x^*}|$, ha tudjuk az x^* pontos

probléma	dim.	$\varepsilon = 10^{-9}$	$\varepsilon = 10^{-6}$
S5	4	14.3	10.7
S7	4	14.2	10.3
S10	4	14.2	10.3
H3	3	14.1	14.1
H6	6	14.0	14.0
GP	2	12.3	0.8
SHCB	2	14.2	14.2
BR	2	13.3	12.5
RB	2	13.6	0.8
L3	2	13.3	13.3
L5	2	13.4	13.4
L12	10	22.7	25.5
Sch21	2	19.2	?
Sch31	3	14.2	14.3
Sch25	2	27.1	27.1
Sch218	2	35.4	31.3
Sch32	3	25.7	19.6
Sch37	10	39.0	19.0
G5	5	14.0	14.1
G10	10	13.8	13.8
R4	5	13.3	13.3
EX2	5	12	0.4
UPP9	2	12.5	1.7
qcp2	5	9.0	10.0
qcp3	6	12.9	10.0
MS	5	9.6	4.8
inteq	15	37.0	33.0
chemeq	5	?	?
I1	10	30.6	17.8
FT	2	26.7	18.7

4.6. táblázat. A helyes jegyek száma.

megoldást, és nem 0. Ha 0, akkor nem kell leosztani az x^*-gal. Intervallumos esetben a kapott intervallum szélessége kerül a számlálóba és a középpontja a nevezőbe. A táblázatból kiderül, hogy általában rosszabb pontosságot kaptunk, különösen igaz ez két nehéz problémára: a Goldstein-Price és az EX2 függvényekre, ezen kívül a Rosenbrock, az MS és az UPP9 feladatokra.

A következőkben azokat az eredményeket vizsgáljuk, amelyeket az eredeti forráskódban történt változtatásokkal kaptunk. A helyi kereső alkalmazásának kiiktatásával kapott eredményeket a 4.7. táblázatban láthatjuk. Megállapíthatjuk, hogy a helyi kereső csak a korlátozási feltételekkel megadott feladatok esetében volt hatással. Ezekre

p	bisec.	%	nfe	%	nge	%	bp	%	stu	%
S5	9	100	273	100	89	100	17	100	533.05	103
S7	16	100	345	100	118	100	20	100	566.09	103
S10	17	100	359	100	123	100	21	100	705.46	101
H3	21	100	470	100	188	100	44	100	1 099.14	103
H6	42	100	1 744	100	490	100	90	100	10 689.66	102
GP	2 828	100	38 119	100	22 449	100	4 205	100	10 346.26	103
SHCB	44	100	720	100	366	100	72	100	280.17	99
BR	6	100	180	100	71	100	11	100	63.22	95
RB	13	100	234	100	108	100	16	100	25.86	100
L3	267	100	4 433	100	1 560	100	356	100	10 668.10	100
L5	62	100	861	100	366	100	100	100	1 991.38	102
L12	45	100	1 618	100	749	100	46	100	19 336.21	100
Sch21	10	100	215	100	81	100	28	100	99.14	102
Sch31	4	100	217	100	74	100	8	100	66.09	100
Sch25	0	100	127	100	25	100	3	100	11.49	88
Sch218	0	100	95	100	21	100	8	100	12.93	100
Sch32	1	100	139	100	47	100	5	100	30.17	100
Sch37	0	100	116	100	12	100	0	100	445.40	99
G5	0	100	67	100	8	100	0	100	199.71	100
G10	0	100	117	100	13	100	0	100	1 426.72	101
R4	35	100	679	100	331	100	63	100	550.29	101
EX2	132	100	3 028	100	1 799	100	216	100	22 168.10	101
UPP9	149	100	1 974	99	1 976	100	280	100	974.14	102
qcp2	0	100	361	63	230	99	42	91	320.40	56
qcp3	3 535	101	34 535	98	23 018	96	3 670	100	17 343.39	81
MS	287	117	3 851	110	4 530	109	362	109	4 709.77	99
Σ	7 523	84	94 782	82	58 821	83	405	83	104 649.41	81

4.7. táblázat. Lokális kereső kikapcsolásával kapott eredmények.

a feladatokra lényegesen kevesebb műveletszámot tapasztalhatunk a a qcp2 és qcp3 problémáknál. A qcp2 esetében a függvénykiértékelések számának jelentős csökkenésével a felhasznált CPU idő közel felére csökkent. A qcp3 problémánál pedig, ahol az alap táblázat szerint a helyi kereső összesen 251 esetben került alkalmazásra szintén a felhasznált CPU idő csökkenése a jelentős változás. A kapott eredmények alapján kijelenthetjük, hogy a lokális kereső egyszerű kiiktatása jelentős teljesítménycsökkenést nem eredményez, egyes esetekben pedig kifejezetten rossz irányba befolyásolja a keresés hatékonyságát. Mint azt Baker Kearfott is javasolta könyvében, egy jobb teljesítményű helyi kereső használatával elképzelhető, hogy jobb eredményeket kaphatunk. Ennek megvalósítását és vizsgálatát egy további kutatási célként jelöljük meg.

prob.	bisec.	%	nfe	%	nge	%	bp	%	stu	%
S5	9	100	183	67	79	88	17	100	406.61	78
S7	16	100	231	67	106	89	20	100	356.32	65
S10	17	100	239	66	111	90	21	100	442.53	63
H3	11	52	218	46	95	50	26	59	461.21	43
H6	68	162	1 087	62	588	120	144	160	5 139.37	49
GP	2 817	99	37 907	99	22 363	99	4 179	99	9 619.25	95
SHCB	40	90	626	86	336	91	62	86	252.87	89
BR	4	66	112	62	57	80	6	54	40.23	60
RB	13	100	208	89	105	97	15	93	24.43	96
L3	267	100	4 369	98	1 556	99	356	100	10 321.84	97
L5	62	100	843	97	364	99	100	100	1 892.24	97
L12	45	100	1 024	63	667	89	46	100	11 364.94	59
Sch21	10	100	153	71	73	90	24	86	84.77	86
Sch31	4	100	193	89	71	96	8	100	57.47	86
Sch25	0	100	44	34	8	32	1	33	4.31	33
Sch218	0	100	33	34	13	62	4	50	7.18	58
Sch32	3	300	99	71	44	93	6	120	22.99	76
Sch37	0	100	50	43	2	16	0	100	186.78	41
G5	0	100	31	46	3	37	0	100	84.77	42
G10	0	100	51	43	3	23	0	100	571.84	40
R4	35	100	625	92	327	98	63	100	497.13	91
EX2	104	79	2 074	68	1 442	80	158	73	14 806.03	67
UPP9	108	72	1 482	75	1 628	82	197	70	581.90	60
qcp2	1	0	132	24	236	101	12	26	234.20	45
qcp3	-	-	-	-	-	-	-	-	-	-
MS	223	91	3 055	87	3 828	92	292	88	3 991.38	84
inteq	0	100	136	41	0	0	0	100	83.33	56
chemeq	5 688	91	112 430	91	18 729	92	7 353	93	44 602.01	92
I1	330	194	6 342	51	1 041	52	381	55	11 199.71	68
FT	70	97	1 676	94	271	96	130	97	554.60	95
Σ	9 945	99	175 653	65	54 146	74	435	85	117 892.24	67

4.8. táblázat. A hámozás kiiktatásával kapott eredmények.

Végül a 4.8. táblázat a hámozás kikapcsolásával kapott eredményeket mutatja. Itt teljesen egyértelmű, hogy ezzel sokkal jobb teljesítmény mutatókat kaptunk: a függvény- és gradienskiértékelések száma lényegesen kevesebb (65%, illetve 74%) Feltehetjük tehát a kérdést, hogy érdemes-e használni ezt az eljárást, hiszen anélkül összességében sokkal kevesebb (egyes esetekben közel fele annyi) CPU időre van szükség a problémák megoldásához. Megjegyezzük azonban, hogy ebben az esetben a GlobSol a qcp3 problémára elfogadható időn belül nem tudott megoldást adni.

probléma	dim.	bisec	gs	secord	epsinf	local	stu
BrownAL	5	0	1	1	0	0	1.43
FT	2	15	138	24	2	10	109.20
I1	10	100	1 128	65	1	1	474.14
KN	8	0	1	1	0	0	2.87
chemeq	5	1 109	4 658	734	0	97	4 794.54
inteq	15	0	1	1	0	0	323.27

4.9. táblázat. A Jacobi mátrix használatával kapott eredmények.

probléma	dim.	bisec	gs	secord	epsinf	local	stu
BrownAL	5	0	1	1	0	0	1.43
FT	2	15	124	22	2	10	104.89
I1	10	100	1 100	67	1	1	474.14
KN	8	0	1	1	0	0	2.87
chemeq	5	1 109	4 658	0	0	97	4 538.79
inteq	15	0	1	0	0	1	244.25

4.10. táblázat.
A Jacobi mátrix használatával a finomítási eljárás kikapcsolásával kapott eredmények.

probléma	dim.	bisec	gs	secord	epsinf	local	stu
BrownAL	5	0	1	1	0	0	1.43
FT	2	24	156	2	2	10	112.07
I1	10	100	1 128	1	1	1	405.17
KN	8	0	1	1	0	0	2.87
chemeq	5	1 109	4 658	0	0	97	3 869.25
inteq	15	0	1	0	0	1	244.25

4.11. táblázat.
A Jacobi mátrix használatával a másodrendű kiterjesztés nélkül kapott eredmények.

4.2. Nemlineáris egyenletrendszerek

Ebben a részben a nemlineáris egyenletrendszerek megoldására alkalmazható `find_roots` eljárást vizsgáljuk. A GlobSol konfigurációs állománya lehetővé teszi a 3.2. fejezetben tárgyalt prekondícionálók (ezek az inverz középpontos és C-LP prekondícionálók), a slope mátrix és a Jacobi mátrix használata közötti választás, valamint az intervallumos Newton-módszer megoldásakor alkalmazott Gauss elimináció vagy a Gauss-Seidel eljárás választásának lehetőségét.

Az egyes táblázatokban feltüntetett oszlopok jelentése a következő:

- a bisec. oszlop az adott feladat megoldásakor alkalmazott intervallum felezések száma;

probléma	dim.	bisec	gs	secord	epsinf	local	stu
BrownAL	5	143	889	101	2	41	479.89
FT	2	14	131	24	2	9	113.51
I1	10	107	1 095	53	1	1	560.34
KN	8	1 229	10 397	671	16	9	5 725.57
chemeq	5	657	2 714	463	1	91	4 127.87
inteq	15	26	1 637	17	1	2	9 770.11

4.12. táblázat. A slope mátrix használatával kapott eredmények.

- gs jelzi az elvégzett Gauss-Seidel lépések számát;
- a secord oszlopban a feladat megoldása során elvégzett másodrendű kiterjesztések számát tüntettük fel;
- az epsinf jelzi az epszilon kiterjesztések alkalmazásainak számát;
- az stu oszlopban pedig a standard CPU időegységek számát találhatjuk. Ennek kiszámítását az előző részben már megadtuk.

Vizsgáljuk először a Jacobi mátrix felhasználásával kapott eredményeket. Itt megjegyezzük, hogy az 5 tesztfeladatból csak kettő esetén kaptunk ténylegesen megadott megoldást FT és az I1 problémák esetében. Ez azt jelenti, hogy a GlobSol a helyi kereső felhasználásával közelít egy olyan intervallumban, amelyről már bebizonyosodott, hogy egyértelműen tartalmaz zérushelyet. A többi probléma esetében nem tudott ilyen közelítő értéket adni, ilyenkor megoldásként (lehetőleg kis méretű) intervallumokat ad.

A `REFINEMENT_IN_VERIFICATION` kapcsoló hamis értékre állításával a műveletszám minimális csökkenését tapasztalhatjuk. A kapcsoló hatása a következő: amennyiben egy intervallumot már leellenőriztünk, hogy tartalmaz legalább egy gyököt, akkor az intervallumos Newton-módszer felhasználásával ezt az intervallumot kisebbé tesszük. Kikapcsolásával ez a művelet elmarad. Lényeges változást a másodrendű kiterjesztés kikapcsolásával sem kaptunk.

Érdekesebb eredményeket a slope mátrix használatával kaphatunk. Láthatjuk, hogy ekkor a GlobSol mind az 5 problémát sikeresen megoldotta.
A `REFINEMENT_IN_VERIFICATION` hamis értékre állításával az eredmények nem változtak, egyedül a BrownAL problémánál eredményezett kevesebb műveletszámot. A másodrendű kiterjesztés nélkül a műveletszám nem csökkent, de a felhasznált CPU idő viszont igen.

Amennyiben a `find_global_min` eljárást alkalmazzuk a nemlineáris egyenletrendszerekre, az intervallumos Newton-módszer alkalmazásánál választhatunk a Gauss-Seidel módszer és a Gauss-elimináció között. Az így kapott eredményeket tartalmazza a 4.15. és 4.16. táblázat. Itt észrevehetjük, hogy az FT probléma kivételével a Gauss-Seidel módszer jobb teljesítményt mutat. Észrevehetjük azonban, hogy a Gauss-elimináció használatával a felhasznált CPU idő lényegesen kevesebb.

probléma	dim.	bisec	gs	secord	epsinf	local	stu
BrownAL	5	133	836	94	2	40	459.77
FT	2	14	131	24	2	9	113.51
I1	10	107	1 095	53	1	1	564.66
KN	8	1 229	10 397	671	16	9	5 788.79
chemeq	5	657	2 714	463	1	91	4 142.24
inteq	15	26	1 637	17	1	2	9 824.71

4.13. táblázat.
A slope mátrix használatával a finomítási eljárás kikapcsolásával kapott eredmények.

probléma	dim.	bisec	gs	secord	epsinf	local	stu
BrownAL	5	143	889	2	2	41	458.33
FT	2	17	155	2	2	10	123.56
I1	10	107	1 095	1	1	1	501.44
KN	8	1 231	10 401	24	16	9	4 849.14
chemeq	5	657	2 714	2	1	91	3 706.90
inteq	15	26	1 637	2	1	2	8 770.11

4.14. táblázat.
A slope mátrix használatával a másodrendű kiterjesztés nélkül kapott eredmények.

probléma	dim.	bisec.	nfe	nge	bp	local	stu
inteq	15	0	326	15	0	2	145.11
chemeq	5	4 448	88 080	14 645	5 731	11	36 225.57
KN	?	10362	199 086	32 902	12 178	17	148 660.92
I1	10	641	12 242	1997	694	9	16 369.25
FT	2	72	1 772	281	133	5	577.59

4.15. táblázat. A Gauss-Seidel módszer használatával kapott eredmények.

probléma	dim.	bisec.	nfe	nge	bp	local	stu
inteq	15	0	326	15	0	2	153.74
chemeq	5	5 692	106 698	17 748	6 346	11	24 015.81
KN	?	11 126	206 878	33 734	11 478	17	107 735.63
I1	10	967	17 852	2 932	977	9	14 147.99
FT	2	68	1 486	231	92	5	308.91

4.16. táblázat. A Gauss-elimináció használatával kapott eredmények.

4.3. Dimenziószám vizsgálata

Ebben a részben a Schwefel-1.2 és Schwefel-3.7 függvények segítségével megvizsgáljuk a GlobSol dimenziószám növekedésre való érzékenységét. A 4.17. táblázatban a

dimenzió	bisec.	nfe	nge	bp	local	stu
4	0	321	47	30	26	117.82
5	0	847	120	78	72	446.84
6	0	2 385	331	218	210	1 778.74
7	0	6 959	956	634	624	6 908.05
8	0	20 481	2 803	1 858	1 846	27 056.03
9	0	60 087	8 212	5 438	5 424	104 247.13

4.17. táblázat. A Schwefel-1.2 függvény dimenziószám-növelésével kapott eredmények.

dimenzió	bisec.	nfe	nge	bp	local	stu
4	0	399	107	30	26	89.08
5	0	1 051	276	78	72	352.01
6	0	2 943	767	218	210	1 451.15
7	0	8 531	2 224	634	624	5 814.66
8	0	24 937	6 519	1 858	1 846	23 247.13
9	0	72 739	19 088	5 438	5 424	91 685.34

4.18. táblázat.
A Schwefel-1.2 függvény dimenziószám-növelésével és a substitution/iteration technika használata nélkül kapott eredmények.

Schwefel-1.2 függvényre az alapbeállításra kapott eredményeket láthatjuk, míg a 4.18. táblázatban ugyanezen függvényre a substitution/iteration kikapcsolásával kapott eredményeket. Észrevehetjük, hogy az utóbbi esetben a függvény- és gradiens kiértékelések száma lényegesen több volt, a felhasznált CPU idő viszont kevesebb. Mindkét esetben az tapasztalhatjuk, hogy az elvégzett műveletek száma kb. duplázódik a dimenziószám növekedésével.

A 4.19. táblázat tartalmazza a Schwefel-3.7 függvény vizsgálatakor kapott eredményeket. Itt csak az alapbeállítással végeztünk el futtatásokat, mivel korábbi tapasztalataink azt mutatják, hogy az egyes technikák ki/be kapcsolása erre a függvényre nincs hatással. A táblázatból kiderül, hogy a dimenziószám növekedésével az egyes műveletek

dimenzió	bisec.	nfe	nge	bp	local	stu
10	0	116	12	0	2	449.71
15	0	166	17	0	2	1 443.97
20	0	216	22	0	2	3 308.91
25	0	266	27	0	2	6 347.70
30	0	316	32	0	2	10 847.70
35	0	366	37	0	2	17 066.09
40	0	416	42	0	2	25 323.28

4.19. táblázat. A Schwefel-3.7 függvény dimenziószám növelésével kapott eredmények.

probléma	nfe	nfeG	nge	ngeG	stu	stuG
S5	144	193	88	45	129.19	515.80
S7	149	246	86	64	186.60	547.41
S10	158	257	90	67	277.51	693.97
H3	303	356	193	96	162.68	1 066.09
H6	1 681	1 203	1 137	274	178 468.90	10 436.78
GP	15 694	32 428	9 195	9 865	271 291.87	10 045.98
SHCB	734	586	435	164	95.69	283.05
BR	202	138	123	25	28.71	66.09
RB	316	184	180	45	28.71	25.86
L3	2 225	3 213	1 326	894	71 770.34	10 577.59
Sch21	918	167	582	52	110.05	97.70
Sch31	12	160	66	19	23.92	66.09
G5	5 869	38	3 648	5	189 952.15	199.71
R4	1 053	548	650	137	110.05	541.67
EX2	274 229	2 542	171 409	281	20 928 230.00	21 876.44
Σ	303 687	42 259	189 208	12 033	21 640 866.37	57 040.23
$\Sigma-$EX2	29 458	39 717	17 799	11 752	712 636.37	35 163.79

4.20. táblázat.
Egy másik intervallumos globális optimalizáló rendszerrel történt összehasonlítás eredményei.

száma nem mutat gyors növekedést. Érdekes megfigyelni, hogy a függvénykiértékelések száma mindig 50-esével, a gradiens kiértékelések száma pedig 5-ösével növekszik.

4.4. Összehasonlítás

Ebben a részben egy összehasonlítást végzünk egy másik intervallumos technikákat felhasználó programmal [5]. A kapott eredményeket a 4.4. táblázatban foglaltuk össze.

Az egyik legszembetűnőbb eredményt az EX2 függvény esetében kapjuk: a GlobSol nagyságrendekkel gyorsabban oldja meg ezt a feladatot. Ezen táblázat utolsó két sorában az összegzésnél két értéket is feltüntettünk, az első tartalmazza az EX2-re kapott eredményeket, a másik pedig nem. Ez utóbbit megvizsgálva láthatjuk, hogy több függvénykiértékelésre volt szükség a GlobSol esetében (134%), kevesebb gradiens kiértékelésre (66%) és lényegesen kevesebb CPU időre (4%). Erre az értékre legnagyobb hatással a Hartman-6, a Goldstein-Price és a Griewank-5 függvények voltak hatással, ezekre a függvényekre a GlobSol sokkal gyorsabban szolgáltatott megoldást.

Mindebből láthatjuk, hogy a GlobSol hatékonyságát tekintve igéretesnek mutatkozik az intervallumos globális optimalizálás világában. Természetesen azzal, hogy ezen néhány feladaton keresztül *egyetlen* más rendszerrel vetjük össze eredményeinket, még nem tudjuk teljes általánosságban alátámasztani a GlobSol hatékonyságát. Tudomásunk szerint a rendszer számos eljárása forráskód szinten további optimalizálást igényel. Ennek elvégzését a program kiismerésével egy további feladatként jelöljük ki.

5. fejezet

Alkalmazások

Ebben a fejezetben bemutatunk néhány olyan érdekes problémát, amelyek megoldására a GlobSol-t használtuk fel. Leírjuk tapasztalatainkat, illetve megemlítünk néhány olyan problémát, amely rámutat arra, hogy a program rendelkezik bizonyos hiányosságokkal. Ezek megoldása képzi további munkáink egy részét. Ismertetésre kerül egy globális optimalizálási szerver is, amely a GlobSol-t felhasználva elektronikus levelezés formájában teszi lehetővé a feladatok megoldását.

Az érdeklődő olvasó figyelmébe ajánljuk a GlobSol honlapját:
`http://interval.louisiana.edu/GlobSol/`
ahol érdekes gyakorlati alkalmazásokról található leírásokat és esettanulmányokat, amelyek címszavakban a következők: polinomok nempontos legnagyobb közös osztójának megkeresése, rakétamotor fúvókájának tervezése, portfólió kezelés, valutakereskedelem, génkutatás, orvosi képfeldolgozás, kémiai számítások és jelfeldolgozás. Ezen alkalmazások rövid ismertetését a [3] cikkben is megtalálhatjuk.

5.1. Körpakolás

A feladat leírása a következő: adott egy egységnégyzet, helyezzünk el ebben n darab azonos átmérőjű kört úgy, hogy ezen körök sugara a lehető legnagyobb legyen. Optimalizálási feladatként: célfüggvényünk a

$$\min -m,$$

a korlátozó feltétlek pedig

$$\begin{aligned} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} &\geq m \quad (1 \leq i < j \leq n) \\ x_i, y_i &\in [0,1] \times [0,1]. \end{aligned}$$

Bővebb leírást, a probléma különböző ekvivalens megfogalmazásait a [29] cikkben találhatjuk.

A probléma megoldásának van egy olyan lépése, amely a kapott megoldások helyességének ellenőrzésére szolgál. Amikor ugyanis adott egy pakolás, a körök egymáshoz viszonyított elhelyezkedéséből egy nemlineáris egyenletrendszert tudunk felírni. Itt azt kell figyelembe venni, hogy két kör érintkezik-e. A kapott nemlineáris egyenletrendszert megoldva adódik, hogy a módszerünk által szolgáltatott koordináta-értékek helyesek-e, illetve mennyire pontosak.

Példaként említhetjük, hogy 10 kör esetében egyszerűsítéseket alkalmazva egy 5 változós nemlineáris egyenletrendszert tudunk felírni [24]:

$$
\begin{aligned}
0 &= 1 - 2x_1 + x_1^2 + x_4^2 - x_5^2 \\
0 &= x_1^2 - 2x_1x_5 + x_3^2 \\
0 &= 2 - 4x_2 + 4x_2^2 + x_4^2 + 2x_4x_5 - 2x_4 - 2x_5 \\
0 &= 5 - 6x_5 - 8x_2 + x_5^2 + 8x_5x_2 + 4x_2^2 + 4x_4^2 - \\
&\quad -4x_4 - 4x_4x_3 - 4x_4x_5 + 2x_3 + x_3^2 + 2x_3x_5 \\
0 &= 4x_2^2 + 1 - 2x_3 - 2x_5 + x_3^2 + 2x_3x_5 - 3x_5^2.
\end{aligned}
$$

Az egyenletrendszer megoldásából az x_5 értéke az, amely számunkra fontos, ez adja meg ugyanis a középpontok közti minimális távolság maximumát. A GlobSol az így kapott problémát viszonylag rövid idő alatt megoldja és az x_5 értékére eredményül a

$$[0.421279543983902000, 0.421279543983904800]$$

intervallumot adja.

A probléma átírása azonban nehezen automatizálható. Ha semmilyen – kézi vagy gépi – egyszerűsítési módszert sem vetünk be, akkor a nemlineáris egyenletrendszerünk dimenziószáma meglehetősen nagy lesz, amelynek megoldása kihívást jelent a GlobSol számára.

Megjegyezzük továbbá, hogy az eredeti, teljes körpakolási feladat különös módon már kis körszám esetén is nehézségeket okoz, a dolgozat írásakor elérhető GlobSol verzió már 5 kör esetére sem ad kielégítő eredményt. A probléma onnan eredhet, hogy az eredeti megfoglamazásban a körök számának növekedésével egyre több korlátozó feltétel adódik, míg a célfüggvény egyetlen változó minimalizálása. Ez a változó szerepel az összes egyenlőtlenség feltételben is, ami problémát jelenthet. További vizsgálataink egyik részét ez képezi: kideríteni azt, hogy a probléma ebben a formában a GlobSolban miért nem oldható meg.

5.2. Reaktorhálózat tervezési probléma

Feladatunk egy kémiai reaktorhálózat tervezésekor adódó korlátozó feltételekkel kiegészített minimumfeladat megoldása [28], ami a következőképpen írható fel. Keressük a

$$\min -x_4$$

értékét az alábbi korlátozó feltételek mellett:

$$\begin{aligned} x_1 + 0.09755988 x_1 x_5 &= 1 \\ x_2 - x_1 + 0.09658428 x_2 x_6 &= 0 \\ x_3 + x_1 + 0.0391908 x_3 x_5 &= 1 \\ x_4 - x_3 + x_2 - x_1 + 0.9 x_4 x_6 &= 0 \\ x_5^{0.5} + x_6^{0.5} &\leq 4 \\ (0,0,0,0,10^{-5},10^{-5}) \leq (x_1,x_2,x_3,x_4,x_5,x_6) &\leq (1,1,1,1,16,16). \end{aligned}$$

A megadott optimalizálási probléma egyik jellemzője, hogy egyetlen globális minimuma van, azonban van még két lokális minimuma is, amelyek értékben nagyon közel, a megoldáshelyeik szerint viszont nagyon távol vannak a globális optimumtól. Így ezek a lokális megoldások két teljesen különböző konfigurációt írnak le. A megoldások a következők [1]:

	obj.	x_1	x_2	x_3	x_4	x_5	x_6
globális	-0.38881	0.772	0.517	0.204	0.388	3.036	5.097
lokális1	-0.38808	1.000	0.393	0.000	0.388	10^{-5}	15.975
lokális2	-0.37461	0.391	0.391	0.375	0.375	15.975	10^{-5}

5.1. táblázat. A reaktorhálózat tervezési probléma globális és lokális optimumai.

Megadva a GlobSol-nak a fenti problémát meglepő módon a második lokális minimumot adja meg, ráközelítve a másik két megoldásra, azt a választ kapjuk, hogy nincs olyan intervallum, amely igazolt lehetséges megoldást tartalmazna.

Észrevehetjük, hogy a probléma felírása sokban hasonlít a körpakolásnál leírt feladatra: egyetlen változó minimalizálása több olyan korlátozó feltétel mellett, amelyekben szerepel ez a változó. Mint azt fentebb is említettük, további vizsgálataink részét képezi a helytelen működés kiderítése.

5.3. EX3 függvény

Tekintsük az

$$f(x) = \begin{cases} \sum_{i=1}^{4} x_i^6 (\sin(\frac{1}{x_i}) + 2) & \text{ha } \prod_{i=1}^{4} \neq 0, \text{ és} \\ 0 & \text{különben.} \end{cases}$$

függvényt. Ezen függvény globális minimumának megkeresése hagyományos eljárásokkal igazoltan reménytelen [6]. Ha a GlobSol segítségével próbáljuk minimalizálni ezt a függvényt a $[-1,1]$ intervallumon, akkor azt a hibaüzenetet kapjuk, hogy a 0 szerepel a nevezőben. Ezt a hibaüzenetet az intervallum aritmetika szabályai miatt kapjuk (lásd 2.1. fejezet), ugyanis nem megengedett olyan intervallummal osztani, amely tartalmazza a 0-t.

Az EX3 függvény minimumhelyének megkeresésére a következő ötlettel próbálkozhatunk. Vegyük külön-külön a $[-1, -\varepsilon]$ és a $[\varepsilon, 1]$ intervallumot, mint a keresés intervallumait, ahol $\varepsilon > 0$ egy rögzített kicsi lebegőpontos szám. Ezekre az esetekre a GlobSol már be tudja határolni a minimumot, ellenőrzött megoldáshelyként olyan kis intervallumokat ad, amelynek bal végpontja az ε szám. A kapott eredmény a következő (itt $\varepsilon = 10^{-36}$):

$$\begin{aligned}
x_1 \in &\quad [0.999999999999999900 \times 10^{-36},\ 0.261136016744936800 \times 10^{-02}] \\
x_2 \in &\quad [0.999999999999999900 \times 10^{-36},\ 0.344052782716820600 \times 10^{-02}] \\
x_3 \in &\quad [0.999999999999999900 \times 10^{-36},\ 0.344052782716820600 \times 10^{-02}] \\
x_4 \in &\quad [0.999999999999999900 \times 10^{-36},\ 0.344052782716820600 \times 10^{-02}] \\
f(x) \in &\quad [0.399999999999998900 \times 10^{-215},\ 0.158790117101244000 \times 10^{-13}]
\end{aligned}$$

A $[-1, -\varepsilon]$ intervallumra hasonló eredményeket kapunk (ezen intervallumok -1-szeresét). Ez alapján azt mondhatjuk, hogy a megoldásunk közelít az egyetlen minimumhoz, ami a $(0, 0, 0, 0)^T$ pontban van.

5.4. Nemdifferenciálható problémák

A valós életből vett problémák megoldása során számos esetben találkozhatunk olyan célfüggvénnyel, amelyben abszolútérték vagy két kifejezés maximuma szerepel. Ilyen esetekben gondot okozhat az, hogy a függvényünk nem differenciálható. Így az alkalmazott automatikus deriválás miatt nem használhatjuk a Fortran 90 nyelvben rendelkezésre álló utasításokat, amelyek az esetszétválasztást kezelni tudják. Hogy ilyen függvényeket is tudjunk optimalizálni, a GlobSol-ban rendelkezésre áll a [16] cikkben bevezetett $x_p = \chi(x_s, x_q, x_r)$ függvény, melynek használatával a GlobSol kódlistát generál a megfelelő IF-THEN-ELSE típusú elágazásokhoz. A valós értékű χ függvény definíciója a következő:

$$\chi(x_s, x_q, x_r) = \begin{cases} x_q & \text{ha} \quad x_s < 0, \text{ és} \\ x_r & \text{különben.} \end{cases}$$

A χ függvény használatával megoldható az $x_p = |x_q|$, valamint az $x_p = \max\{x_q, x_r\}$ alakú függvények definiálása és használata.

Tekintsük például a következő feladatot. Adott a síkon egy egységnyi szakasz, az egyszerűség kedvéért legyen ez az x tengelyen. Az optimalizálandó függvényünk értéke ettől a szakasztól vett távolság lesz, azaz vagy a közelebbi végponttól vett távolság, vagy pedig a megfelelő y koordintátaérték abszolútértéke. Feladatunk az így megadott függvény minimalizálása.

A probléma leírásából rögtön látszik, hogy a minimum a megadott egységnyi intervallumon vétetik fel, tehát válaszként a teljes intervallumot kell megadni. A GlobSol-ban ezt a problémát a χ függvény segítségével oldhatjuk meg, hiszen ezzel tudjuk kezelni az esetszétválasztást.

Az eredeti probléma célfüggvénye azonban differenciálható volt, így jogos annak a kérdésnek a felvetése, hogy miért kell akkor a χ függvényt használni. Erre azt a választ adhatjuk, hogy ennek segítségével gyorsan és áttekinthetően tudjuk kezelni a problémát, valamint a GlobSol is a helyes választ adja, ha így adjuk meg a fenti függvényt (visszaadja a szakaszt több rövidebb részre felbontva):

$$\begin{aligned}
x_1 &\in [0.99729504188789030,\ 1.00731810763550700] \\
x_2 &\in [1.00731810763550700,\ 1.01374222539583000] \\
x_3 &\in [1.01374222539583000,\ 1.02805078246765900] \\
x_4 &\in [1.02805078246765900,\ 1.05901406363648100] \\
x_5 &\in [1.05901406363648100,\ 1.12094062597412700] \\
x_6 &\in [1.12094062597412700,\ 1.24479375064941800] \\
x_7 &\in [1.24479375064941800,\ 1.49250000000000000] \\
x_8 &\in [1.49250000000000000,\ 1.50750000000000000] \\
x_9 &\in [1.50750000000000000,\ 1.75562500000000000] \\
x_{10} &\in [1.75562500000000000,\ 1.87968750000000000] \\
x_{11} &\in [1.87968750000000000,\ 1.94171875000000000] \\
x_{12} &\in [1.94171875000000000,\ 1.97058583984375000] \\
x_{13} &\in [1.97058583984375000,\ 1.98376188928222600] \\
x_{14} &\in [1.98376188928222600,\ 2.00369919470214800] \\
y_i &\in [-0.520417042793042300 \times 10^{-17}, 0.477048955893622000 \times 10^{-17}] \\
f(x_i, y_i) &\in [0.000000000000000000,\ 0.270833898429455500 \times 10^{-34}]
\end{aligned}$$

5.5. GlobSol szerver

Ezeket a problémákat, vagy más feladatokat elvégezhetünk a az egyetem Számítóközpontjában lévő IBM RS/6000 gépen. Itt a rendszergazdák jóvoltából egy GlobSol szervert üzemeltetünk, amellyel lehetőségünk van kipróbálni az ötleteinket. A biztonságosság érdekében egyenlőre csak a Szegedi Egyetem gépeiről küldött leveleket fogadja. Ha igénybe akarjuk venni ezt a szolgáltatást, nincs más dolgunk, csak küldjük el e-mailben az optimalizálandó problémánkat a következő címre:

`h531733@mirror.cc.u-szeged.hu`

Itt egy program értelmezi a levelet, tehát ennek megadott módon tartalmaznia kell a fájlokat. Ha a levelünk `Subject:` mezője a `globsol_info_file` bejegyzést tartalmazza, akkor a szerver visszaküld egy levelet, amely tartalmaz egy kidolgozott példát, amit itt rövidítve láthatunk.

```
To: h531733@mirror.cc.u-szeged.hu
Cc:
Attachmnt:
Subject: globsol
---Message text---

>> problem <<
         HERE COMES THE NAME OF YOUR PROBLEM!
Be careful! In the problem.f90 the OUTPUT_FILE_NAME must be problem.CDL
>> problem.f90 <<
         HERE COMES THE problem.f90 FILE!
>> problem.DT1 <<
         HERE COMES THE problem.DT1 FILE!
>> makefile <<
         HERE COMES THE makefile, OR IF THERE ISN'T,
         WE USE THE GENERIC makefile!
>> GlobSol.CFG <<
         HERE COMES THE GlobSol.CFG, OR IF THERE ISN'T,
         WE USE THE GENERIC GlobSol.CFG!
>> other.file1 <<
         IF YOU HAVE OTHER FILES, WHAT WE HAVE TO USE
         FOR YOUR PROBLEM, PUT HERE!
         Of course, change the "other.file1" to the
         name of your file...
>> other.file2 <<
 ...
>> end <<
```

Ez alapján a szerver elindítja az optimalizálást, és a kapott eredményt visszaküldi a feladónak. Ha valamilyen hibával leáll a fordító, akkor a hibaüzenetet küldi vissza. Ha a program 2 napon belül nem fut le, akkor a szerver leállítja az optimalizálást, és 'it hasn't stopped' hibaüzenetet küld.

A fájlok leírásáról egy példát találhatunk a [30] cikkben. A `Subject:` mezőbe `info`-t írva bármely kérésre, kérdésre szívesen válaszolunk.

6. fejezet

Összefoglalás

Dolgozatunkban az intervallumos globális optimalizálás és nemlineáris egyenletrendszerek megoldásához felhasználható kifinomult technikákat vizsgáltuk. A számos eljárás megvalósítását tartalmazó, kutatási célokra kifejlesztett GlobSol nevű FORTRAN 90 nyelven írt program lehetővé tette számunkra, hogy empirikus vizsgálatokat végezve tapasztalatokat szerezzünk .

Főbb eredményeink közül első, hogy az intervallumos globális optimalizás átfogó tárgyalását adjuk, amely a technikák tulajdonságainak vizsgálatát elméleti és gyakorlati szempontból is lehetővé teszi. Dolgozatunkkal először adtunk olyan részletes teszteredményeket, amelyek a helyes használathoz adnak útmutatást. Elvégeztünk egy összehasonlító tesztet is egy másik intervallumos optimalizáló rendszerrel, amely alapján a GlobSol hatékonysága kimutatható. A numerikus vizsgálatoknál számos továbbfejlesztési és továbblépési pontot adtunk. Tapasztalataink azt mutatják, hogy például a substitution/iteration technika jelentős teljesítmény növekedést eredményez a korlátozó feltételeket tartalmazó feladatok esetében; a másodrendű kiterjesztés használata talán nem annyira fontos; a hámozási technika gyakran jelentősen ront a teljesítményen és hogy a helyre vonatkozó tolerancia megfelelő megválasztása döntő fontosságú lehet a megoldások pontos megadásához. Megadtunk néhány olyan feladatot is, amelyek megoldása elvileg lehetséges a GlobSol segítségével, azonban ezeknél olyan jelenségeket is tapasztaltunk, amelyek hiányosságok meglétére hívják fel a figyelmet. Ezekkel a program fő szerzőjének érdeklődését is felkeltettük.

Mivel a GlobSol hozzávetőlegesen egy évtized munkája, így annak teljes kiismerése, átlátása nem könnyű feladat. Jelen munkánkkal ezen a téren tudunk előrelépni, amely kiindulási pontját képezi a tervezett doktori programunk kutatási területének. A program megírásával sokan foglalkoztak, illetve foglalkoznak, ezért annak mély ismerete lehetővé teszi számunkra egy nemzetközi szintű tudományos együttműködés létrejöttét.

Irodalomjegyzék

[1] C.S. Adjiman, I. P. Androulakis, and C. A. Floudas. A global optimization method, αBB, for general twice-differentiable NLPs – II. implementation and computational results. *Comput. Chem. Eng.*, 22:1159–1179, 1998.

[2] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, New York, 1983.

[3] G.F. Corliss and R.B. Kearfott. Rigorous global search: Industrial applications. In T. Csendes, editor, *Developments in Reliable Computing.* Kluwer, Dordrecht, 1999.

[4] Csallner András. *Intervallumfelosztási eljárások a globális optimalizálásban.* PhD thesis, József Attila Tudományegyetem, Szeged, 1999.

[5] T. Csendes. New subinterval selection criteria for interval global optimization. *Journal of Global Optimization, közlésre benyújtva.*
Elektronikus változat: `http://www.inf.u-szeged.hu/~csendes/pfconv.ps.gz`

[6] T. Csendes. Nonlinear parameter estimation by global optimization – efficiency and reliability. *Acta Cybernetica*, 8:361–370, 1989.

[7] L. C. W. Dixon and G. P. Szegö. The global optimization problem: An introduction. In L. C. W. Dixon and G. P. Szegö, editors, *Towards Global Optimization 2*, pages 1–15, Amsterdam, Netherlands, 1978. North-Holland.

[8] G. B. Ferraris and E. Tronconi. BUNLSI - a FORTRAN program for solution of systems of nonlinear algebraic equations. *Comput. Chem. Eng.*, 10:129–141, 1986.

[9] C. A. Floudas and P. M. Pardalos *et al.* *Handbook of Test Problems in Local and Global Optimization.* Nonconvex Optimization And Its Application. Kluwer, Netherland, 1999.

[10] E. R. Hansen. Interval arithmetic in matrix computations, part 1. *SIAM J. Numer. Anal.*, 2:308–320, 1965.

[11] E. R. Hansen. Global optimization using interval analysis: The one-dimensional case. *J. Optim. Theory Appl.*, 29(3):331–44, 1979.

[12] E. R. Hansen. *Global Optimization Using Interval Analysis.* Marcel Dekker, Inc., New York, 1992.

[13] R. B. Kearfott, M. Dawande, K.-S. Du, and C.-Y. Hu. Algorithm 737: INTLIB, a portable FORTRAN 77 interval standard function library. *ACM Trans. Math. Software*, 20(4):447–459, December 1994.

[14] R.B. Kearfott. Decomposition of arithmetic expression to improve the behavior of interval iteration for nonlinear systems. *Computing*, 47(2):169–191, 1991.

[15] R.B. Kearfott. An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization*, 2:259–280, 1992.

[16] R.B. Kearfott. A Fortran 90 environment for research and prototyping of enclosure algorithms for nonlinear equations and global optimization. *ACM Trans. Math. Software*, 21(1):63–78, 1995.

[17] R.B. Kearfott. *Rigorous Global Search: Continuous Problems.* Kluwer, Dordrecht, 1996.

[18] G. Mayer. Epsilon–inflation in verification algorithms. *J. Comput. Appl. Math.*, 60:147–169, 1994.

[19] K. Meintjes and A. P. Morgan. Chemical-equilibrium systems as numerical test problems. *ACM Trans. Math. Software*, 16:143–151, 1990.

[20] R. E. Moore. *Methods and Applications of Interval Analysis.* SIAM, Philadelphia, 1979.

[21] J. J. Moré and M. Y. Cosnard. Numerical solution of nonlinear equations. *ACM Trans. Math. Software*, 5:64–85, 1979.

[22] B. A. Murtagh and M. A. Saunders. Minos 5.4 user's guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Dept. Operations Res., Stanford University, 1993.

[23] A. Neumaier. *Interval Methods for System of Equations.* Cambridge Univerity Press, Cambridge, England, 1990.

[24] R. Peikert, D. Würtz, M. Monagan, and C. de Groot. Packing cirlces in a square: A review and new results. In P. Kall, editor, *System Modelling and Optimization, volume 180 of Lecture Notes in Control and Information Sciences*, pages 45–54. Springer-Verlag, Berlin, 1992.

[25] D. Ratz and T. Csendes. On the selection of subdivision directions in interval branch-and-bound methods for global optimization. *Journal of Global Optimization*, 7:183–207, 1995.

[26] J. Rohn. NP-hardness results for linear algebraic problems with interval data, 1994. Preprint, Faculty of Math. and Physics, Charles University, Malostranske nam. 25, 11800 Praha 1, Czech Republic.

[27] Siegfried M. Rump. Verification methods for dense and sparse systems of equations. In Jürgen Herzberger, editor, *Topics in Validated Computations: Proceedings of the IMACS-GAMM International Workshop on Validated Computations, University of Oldenburg*, Elsevier Studies in Computational Mathematics. Elsevier, to appear.

[28] H. S. Ryoo and N. V. Sahinidis. Global optimization if nonconvex NLPs and MINLPs with applications in process design. *Comput. Chem. Eng.*, 19(5):551–566, 1995.

[29] Szabó Péter Gábor. Optimális körelhelyezések a négyzetben. *Polygon, közlésre elfogadva.*
Elektronikus változat: `http://www.inf.u-szeged.hu/~pszabo/polygon.ps.gz`

[30] Tóth Boglárka és Vinkó Tamás. Egy hatékony számí tógépes eszköz matematikai problémák megoldására. *Polygon, közlésre benyújtva.*
Elektronikus változat: `http://www.cab.u-szeged.hu/~h531733/gspoly.ps.gz`

[31] P. Van Hentenryck, D. McAllester, and D. Kapur. Solving polynomial systems using a branch and prune approach. Technical Report CS-95-01, Dept. of Comp. Sci., Brown University, 1995.

[32] P. Van Hentenryck, D. McAllester, and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM J. Numer. Anal.*, 34(2):797–827, 1997.

A. Függelék

Installálás

Az ebben a dolgozatban ismertetett globális optimalizáló program FORTRAN 90 forráskódja szabadon hozzáférhető a

`http://interval.louisiana.edu/GlobSol/download_GlobSol.html`

címen. A programot eredetileg UNIX munkaállomásra tervezték, de használható személyi számítógépeken Linux, illetve MS-DOS alatt is. Az alapvető standard beállítások elsősorban a SunOS-t támogatják, de apró módosításokkal más UNIX rendszeren is működik. A fenti címről letölthető tömörített állomány mérete kb. 1,1 Mbyte, a teljes forráskód pedig 5,5 Mbyte. A fordításhoz szükséges kb. 20 Mbyte szabad hely és tapasztalat szerint működik 16 Mbyte memóriával is. Használatához szükségünk van egy Fortran 90 fordítóprogramra. Tudomásunk szerint semelyik operációs rendszerhez sincs olyan ingyenes Fortran 90 fordító, amellyel a GlobSol sikeresen lefordítható és használható. Másrészt megemlítjük, hogy különböző operációs rendszerekhez ingyenes próbaverziókat találhatunk a Fujitsu Ltd. honlapján: `http://www.fujitsu.co.uk/devtools.htm`, valamint a Numerical Algorithms Group honlapján:
`http://www.nag.co.uk/nagware/NP/NP_trial.html`. Ezekkel a fordítókkal sikeresen le tudtuk fordítani a GlobSol-t, illetve tapasztalatokat szerezhettünk még egy AIX 4.3.2 operációs rendszeren működő (2.2(261) verziójú) NAGWare f90 fordítóval kapcsolatban is. (Ezen a rendszeren működik az ismertetett GlobSol szerver is.)

A következő 4 fordítóhoz külön installáló fájlokat találunk, de további fordítók is használhatók, amennyiben megváltoztatjuk a `makefile` megfelelő sorait. Az installáló (kicsomagoló) fájlokat szintén a GlobSol Download oldalán érhetjük el.

- `unpack_sun_sun` — SunOS Sun Fortran fordítóhoz
- `unpack_sun_nag` — SunOS NAG Fortran fordítóhoz
- `unpack_dos_dec.bat` — MS-DOS DEC Fortran fordítóhoz
- `unpack_dos_nag.bat` — MS-DOS NAG/Salford Fortran fordítóhoz

Amennyiben Linux operációs rendszerre telepítjük, az `unpack_sun_nag` fájlt használhatjuk.

Az `unpack_dos_dec.bat` és az `unpack_dos_nag.bat` batch fájlok használják a WinZip parancssori változatát, a `wzunzip` parancsot, de ezt kicserélhetjük `unzip` vagy `pkunzip` parancsra, amennyiben ez szükséges. Ha egyik sem áll rendelkezésre, a WinZip parancssori segédeszközei elérhetőek a WinZip honlapján a `http://www.winzip.com/` címen. Az MS-DOS DEC Fortran fordítóhoz szükség van GNU make-re, amit letölthetünk az `ftp://ftp.gnu.org/pub/gnu/make/` címről, illetve a GlobSol Download oldaláról is. Az `unpack_sun_sun` és az `unpack_sun_nag` shell-script fájlok az `unzip` parancsot használják, de ez általában minden UNIX rendszerű gépen elérhető.

Miután lefuttattuk a megfelelő `unpack` fájlt, meg kell változtatnunk a `makefile`-ban a 116. sort, vagyis írjuk be a GlobSol könyvtárunk elérési útját, illetve ugyanezt írjuk be az `fmake` fájlba is. Ha nem a megadott installáló fájlhoz tartozó fordítót használjuk, akkor a `makefile` 150-200. sorában az `F90`, `F77`, illetve az `LNK` deklarációkat kell még megváltoztatnunk. A Fujitsu fordító esetén ez a következőképpen néz ki:

```
F90 = frt -O0 -X9 -Am -M$(MODULEDIR) -I$(MODULEDIR) -c
F90 = frt -O0 -X9 -Am -M$(MODULEDIR) -I$(MODULEDIR)
F90 = frt -O0 -X9 -c
```

a NAGWare Fortran 90 fordítóhoz pedig a következő bejegyzések szükségesek:

```
F90 = f90 -O -I$(MODULEDIR) -c
LNK = f90 -O -I$(MODULEDIR)
F77 = f90 -O -c
```

Ezek után adjuk ki a `make` parancsot. A fordítás hosszú időt vehet igénybe. Amennyiben problémák merülnek fel a fordítás során, probálkozzunk a következőkkel:

- Ha a `dsecnd.f90` fájl fordításánál akadunk el, akkor az azt jelenti, hogy a mi fordítónk más időfüggvényeket használ. Ha ismerjük a megfelelő függvények neveit, cseréljük ki a fájlban feltételezett függvényeket. Ha nem tudjuk, vagy nem sikerül úgy se lefordítani, akkor egyszerűen a `GlobSol.zip` fájlunkból tömörítsük ki az általános `dsecnd.generic.f90` fájlt, majd nevezzük át `dsecnd.f90`-re. Ez egy "nem csinál semmit" fájl, tehát nem tudjuk majd, hogy esetenként mennyi CPU időt használtunk, de ettől a programunk még tökéletesen fog működni.
- Amennyiben a `sun_stubs.f90` lefordítása jelent gondot, itt is először próbálkozhatunk a fordítónk IEEE lebegőpontos számábrázolást beállító függvényeinek beírásával. Azok ismerete nélkül a szintén az eredeti `GlobSol.zip` fájlba tömörített `sun_stubs.generic.f90` fájlt nevezzük át a GlobSol könyvtárunkban `sun_stubs.f90`-re. Így a fordítással nem lehet probléma, hiszen ez is egy "nem csinál semmit" fájl.
- Megjegyezzük, hogy ha a `unix_make_library` (vagy `make_library`) futása során áll le a fordítónk hibaüzenettel, akkor az általában az előző két fájl hibájából adódik.