

A korrelációs energia számítása nagy rendszerekre lokális közelítésekkel

SZEGEDY LÓRÁNT

Fizika alapszak (BSc)

Budapesti Műszaki és Gazdaságtudományi Egyetem
Természettudományi Kar

Témavezető: Dr. Kállay Mihály

Budapesti Műszaki és Gazdaságtudományi Egyetem
Fizikai Kémia és Anyagtudományi Tanszék

Budapest, 2012

A dolgozat bemutatását a TÁMOP - 4.2.2.B-10/1-2010-0009
"Új tehetséggondozó programok és kutatások a Műegyetem tudományos
műhelyeiben" című program támogatta.

Előszó

Nagy molekulák, szupramolekuláris rendszerek, folyadékok és szilárd testek fizikai és kémiai tulajdonságainak pontos ismerete számos tudományterület számára alapvető fontosságú. Ezen rendszerek tulajdonságainak elméleti meghatározásához a Schrödinger-egyenlet közelítő megoldására van szükségünk. Ehhez manapság a legpontosabb eljárás a coupled-cluster (CC) módszer, amellyel az elektronok korrelációs energiáját tetszőleges pontossággal számíthatjuk. A CC elmélet alapfeltevése, hogy a hullámfüggvény felírható egy exponencializált gerjesztő operátor (klaszteroperátor) segítségével, amely a Fermi-vákuumra hat. A klaszteroperátor olyan gerjesztő operátorok összege, amelyek n elektront gerjesztenek betöltött pályáról virtuálisra ($n=1, 2, \dots$). Ha a klaszteroperátorban csak egyszeres és kétszeres gerjesztéseket veszünk figyelembe, akkor az ún. CCSD (CC singles and doubles) módszert kapjuk. A hullámfüggvény paramétereinek meghatározásához szükséges egyenletek egy nemlineáris egyenletrendszer alkotnak, amelynek a megoldását iteratívan kapjuk. Ennek a módszernek a számításigénye a rendszer méretének az hatodik hatványával nő. Egy ennél pontosabb közelítést kapunk, ha a háromszoros gerjesztéseket perturbatíván figyelembe vesszük. Ezt CCSD(T)-nek (CC singles, doubles and perturbative triples) nevezzük. Ennek a számításigénye a rendszer méretének a hetedik hatványával nő.

Ezeknek a módszereknek a hátránya tehát a számításigény rohamos növekedése a rendszer méretével. Azonban a számításigény nagymértékben csökkenthető, ha kihasználjuk, hogy az elektronkorreláció hatása a távolság hatodik hatványával csökken és ezért a távoli elektronok korrelációját elhanyagolhatjuk. Az ilyen közelítéseket lokális közelítéseknek nevezzük. A vizsgált rendszer egyes részeit különálló rendszereknek (domének) tekintjük. Ezekre a doménekre külön-külön megoldjuk a CCSD, ill. CCSD(T) egyenleteket az adott doménhez hozzárendelt lokalizált molekulapálya bázisban, és a kapott parciális energiákat összeadva megkapjuk a teljes energia közelítő értékét. Ezzel a technikával jóval kisebb lesz a számításigény, mivel a doménekben a lokalizált bázis mérete jóval kisebb, mint a teljes rendszerben. A számításigény lineárisan fog skálázódni a rendszer méretével.

Céлом egy hatékony lokális CCSD(T) program kifejlesztése volt. A manapság használt programok mindegyikét nagy méretű bázisra optimalizálták és általában atompálya bázisban számolnak. Emiatt szükséges volt egy olyan program megírására, amely molekulapálya bázisban számol és teljesítménye kisméretű bázis esetén optimális. Figyelembe véve egy átlagos domén méretét ez azt jelenti, hogy az elektronok száma legfeljebb 100, a bázisfüggvények száma pedig legfeljebb 300. A programot egy legalább 8GB méretű memóriával rendelkező számítógépre terveztem úgy, hogy a memóriát a lehető leghatékonyabban használja fel. Ezt úgy érem el, hogy a program a futás legelején lefoglalja a teljes használni kívánt memóriát és ezt folytonosan tölti meg az aktuálisan használt tömbökkel. Az I/O műveletek számát minimalizáltam, mivel ezek a leglassabbak. A CCSD, ill. CCSD(T) egyenleteket mátrixműveletek formájában programoztam, amiket nagy hatékonyságú ún. BLAS (Basic Linear Algebra Subprograms) rutinokkal számol ki a program. Az egyenleteket úgy írtam át, hogy a műveleteket a lehető leghatékonyabban számolhassam, azaz a legkisebb memória és I/O művelet felhasználásával és a legkevesebb lebegőpontos művelet elvégzésével.

Tesztszámolásokat végeztem szénhidrogénekre különböző bázisokban mind a hagyományos, mind a lokális CCSD(T) módszerekkel. Ezek alapján az általam írt program hagyományos CCSD számításokra több, mint kétszer gyorsabb, mint a jelenlegi leggyorsabb program. Lokális számolásoknál pedig a futási idő drasztikusan csökken.

Tartalomjegyzék

1	A coupled-cluster elmélet	7
1.1.	Bevezetés	7
1.2.	A coupled-cluster egyenletek	9
1.3.	Egy- és kétszeres gerjesztések	11
1.4.	Perturbatív korrekció	12
1.5.	A lokális coupled-cluster elmélet	15
2	A CCSD egyenletek és implementálásuk	19
2.1.	A faktorizált egyenletek	19
2.2.	Az implementáció	22
2.2.1.	Bevezetés	22
2.2.2.	A felhasznált BLAS rutinok	22
2.2.3.	A memória kezelése	22
2.2.4.	Az integrállisták tárolása	24
2.2.5.	A módosított egyenletek	25
2.3.	Az algoritmus	28
2.3.1.	Az iteráció előtti műveletek	28
2.3.2.	Műveletek egy iterációs lépésen belül	29
2.4.	Módosítások a lokális számításhoz	38
3	A perturbatív (T) korrekció	39
3.1.	Az egyenletek	39
3.2.	Az algoritmus	41
3.2.1.	A memória szervezése	41
3.2.2.	A számolás előtti műveletek	41
3.2.3.	A (T) korrekció kiszámítása	42
3.3.	Módosítások a lokális számításhoz	46
3.4.	Algoritmus kevés memória esetén és lokális számoláshoz	48
3.4.1.	A memória szervezése	48
3.4.2.	A számolás előtti lépések	49
3.4.3.	A (T) korrekció kiszámítása	49

4	Tesztszámolások	55
4.1.	Általános megjegyzések a teszt számolásokhoz	55
4.2.	Teszt számolások a CCSD(T) programra	55
4.3.	Teszt számolások a lokális CCSD(T) programra	57
5	Összefoglalás, konklúziók	61
Függelék A A (T) korrekció energiaképletének levezetése		63
Függelék B A lokális (T) korrekció energiaképletéhez szükséges levezetés		71
Függelék C Molekulák geometriái		75
Irodalomjegyzék		81

1. fejezet

A coupled-cluster elmélet

1.1. Bevezetés

A coupled-cluster (CC) módszert Coester és Kümmel dolgozta ki az 1950-es években magfizikai jelenségek elméleti vizsgálatára [1]. Egy évtizeddel később Čížek alkalmazta először a kvantumkémiaiában [2–5]. Publikálta a CC egyenleteket abban az esetben, amikor csak a kétszeres gerjesztéseket vesszük figyelembe (CC doubles, CCD) és ennek a megoldására egy linearizált közelítő módszert (linearized CC, LCC) javasolt. Az összes egyszeres és kétszeres gerjesztést tartalmazó módszert (CC singles and doubles, CCSD) Purvis és Bartlett vezette le és implementálta [6]. Lee és munkatársai dolgozták ki az egy-, két- és háromszoros gerjesztéseket is tartalmazó CC módszert (CC singles, doubles and triples, CCSDT) [7], ennek az implementálásával kapott számításokat Noga és Bartlett publikálta [8]. A négyszeres gerjesztéseket is figyelembe vevő módszert (CC singles, doubles, triples, and quadruples, CCSDTQ) Kucharski és Bartlett [9, 10] valamint Oliphant és Adamowicz [11] vezette le. Az ötszörös gerjesztéseket perturbatíván kezelő módszert is publikáltak [12].

Paldus, Čížek és Shavitt alkalmazott először a magasabb gerjesztések figyelembevételére perturbatív módszereket [13], később Bartlett és munkatársai [7, 14, 15] és még sokan mások is fejlesztettek hasonló közelítéseket. Ezekből a perturbatív közelítő módszerekből léteznek iteratív és nem iteratív módszerek. Az egyik ilyen legegyszerűbb modell az Urban-féle CCSD[T] [más néven CCSD+T(CCSD)] modell, ahol a háromszoros gerjesztések legalacsonyabb rendű perturbatív járulékában szereplő másodrendű klaszteramplitúdókat konvergált CCSD amplitúdókkal számítjuk ki [16]. Ennek a módszernek a továbbfejlesztése az ún. CCSD(T) közelítés, ahol a CCSD[T] energiához hozzáadunk egy ötödrendű tagot, amit szintén a konvergált CCSD amplitúdók segítségével számolunk ki [17, 18]. Ezt először Raghavachari vezette le kanonikus Hartree–

Fock (HF) referencia determinánsra [17], de tetszőleges referenciákra is általánosították [19, 20]. Később több negyed-, ötöd- és magasabb rendű korrekciót is publikáltak [21, 22].

A CC módszerek hátránya, hogy a vizsgált rendszer méretének, a figyelembe vett gerjesztésektől függően, magas hatványával nő a számításigény. A skálázódás csökkenthető perturbatív módszerek használatával, de ezzel nem oldódik meg az a probléma, hogy a teljes rendszer méretének adott hatványával nő a számításhoz szükséges erőforrás. Ennek kiküszöbölésére hatékony módszer lokális közelítések alkalmazása.

Először Pulay Péter kezdett el komolyan foglalkozni lokális közelítésekkel [23]. Lokalizálta a betöltött pályákat [24, 25], virtuális pályáknak pedig projektált atompályákat (projected atomic orbitals, PAO) használt és elhanyagolta az egymástól távol eső pályák kölcsönhatását. Minden betöltött pályapárhoz hozzárendelt korlátozott számú virtuális pályát és elhanyagolta továbbá a betöltött párok és a többi virtuális pálya közti kölcsönhatást. Ilyen módon lokális MBPT (many-body perturbation theory) [26, 27] és CEPA (coupled electron pair) [23] módszereket dolgozott ki. Ezeknek az eredményeknek a hatására sok más kutató kezdett lokális módszerek fejlesztésébe [28–30].

Werner és munkatársai [31, 32], valamint Schütz [33, 34] kezdtek el lokális CC módszerekkel foglalkozni, akik lokális CCSD és CCSD(T) módszereket fejlesztettek ki. A (közel) lineáris skálázódás elérésének érdekében csoportosították a betöltött párokat a távolságuk szerint. A távoliak kölcsönhatását elhanyagolták a nem túl távoliak járulékát pedig perturbatív módszerekkel vették figyelembe. Utánuk sok más lokális CC módszert fejlesztettek ki, ezekben általában felosztják a vizsgált rendszert kisebb részekre és a teljes energiát ezeknek a részeknek az energiájának összegével, ill. a részrendszerek közti kölcsönhatási energiák összegeként számolják [35–37]. Ilyen módszer a Stoll-féle inkrementumok módszerét használó lokális CC közelítés [38], a Kobayashi és Nakai által kidolgozott „oszd meg és uralkodj” (divide-and-conquer) CC módszer [39, 40], a Ziolkowski és munkatársai által javasolt DEC (divide-expand-consolidate) CC modell [41] vagy a Flocke és Bartlett által publikált „természetesen lineárisan skálázódó” (natural linear scaling) CC közelítés [42, 43]. Az általunk használt módszer a Li és munkatársai által kidolgozott „klaszter a molekulában” (cluster-in-molecule, CIM) módszeren alapszik [44, 45], ahol minden lokalizált molekulapályához (LMO) hozzárendelnek egy bonyolult módszerrel bizonyos számú betöltött és virtuális molekulapályát (MO). A mi módszerünk nagyban hasonlít még a Neese és munkatársai által kifejlesztett „pár természetes pálya” (pair natural orbital, PNO) CC modellhez [46, 47], ahol minden LMO párhoz kiszámolják a PNO-kat másodrendű Møller–Plesset (MP2) sűrűségmátrixok vizsgálatával.

1.2. A coupled-cluster egyenletek

A coupled-cluster (CC) elméletben [4, 48, 49] a hullámfüggvényt egy exponenciális gerjesztő operátor segítségével fejezzük ki, ami egy referenciafüggvényre hat:

$$|\Psi\rangle = e^{\hat{T}}|0\rangle. \quad (1.1)$$

Itt $|0\rangle$ a referencia determináns – általában a Hartree–Fock (HF) hullámfüggvény – és $\hat{T}$ gerjesztő operátor (klaszteroperátor). A klaszteroperátort felírhatjuk a

$$\hat{T} = \sum_{k=1}^n \hat{T}_k \quad (1.2)$$

összegként, ahol $\hat{T}_k$ k -szorosán gerjesztő operátorok $t_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k}$ ún. klaszteramplitúdókkal vett lineáris kombinációja:

$$\begin{aligned} \hat{T}_k &= \sum_{\substack{a_1 < a_2 < \dots < a_k \\ i_1 < i_2 < \dots < i_k}} t_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k} a_1^+ i_1^- a_2^+ i_2^- \dots a_k^+ i_k^- \\ &= \frac{1}{(k!)^2} \sum_{\substack{a_1, a_2, \dots, a_k \\ i_1, i_2, \dots, i_k}} t_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k} a_1^+ i_1^- a_2^+ i_2^- \dots a_k^+ i_k^-, \end{aligned} \quad (1.3)$$

$a, b, c, \dots$ virtuális, $i, j, k, \dots$ betöltött pályákat indexelnek a referencia függvényben, $p, q, r, \dots$ általános indexeket jelölnek. Másodkvantált formalizmust használunk: a^+ -szal jelöljük azt a keltő operátort, ami az a . virtuális pályán kelt egy elektront, i^- -szal azt az eltüntető operátort, ami az i . betöltött pályáról vesz le egy elektront. A 50. referencia jelöléseit követve a t_{ab}^{ij} klaszteramplitúdókat

$$t_{ab}^{ij} = T_{ab}^{ij} \quad (1.4)$$

szerint jelöljük a 2. fejezettől kezdve. Ebben a formalizmusban felírhatjuk a Hamilton-operátor másodkvantált alakját:

$$\hat{H} = \sum_{pq} h_{pq} p^+ q^- + \frac{1}{4} \sum_{pqrs} \langle pq || rs \rangle p^+ q^+ s^- r^- \quad (1.5)$$

ahol h_{pq} -vel jelöljük az egyelektron integrálokat:

$$h_{pq} = \int_{-\infty}^{\infty} \varphi_p^*(\mathbf{r}) \left(-\frac{1}{2} \nabla^2 + \sum_A \frac{Z_A}{|\mathbf{r} - \mathbf{r}_A|} \right) \varphi_q(\mathbf{r}) d\mathbf{r}, \quad (1.6)$$

A az atommagokat indexeli, Z_A az A -edik atommag rendszáma, $\mathbf{r}_A$ az A -edik atommag koordinátája és $\varphi_p(\mathbf{r})$ a p -edik elektron spinpályája egyelektron bázisban. $\langle pq||rs \rangle$ -sel az antiszimmetrikus kételektron integrálokat jelöljük:

$$\langle pq||rs \rangle = \langle pq|rs \rangle - \langle pq|sr \rangle, \quad (1.7)$$

ahol $\langle pq|rs \rangle$ a kételektron integrálokat jelöli:

$$\langle pq|rs \rangle = \int_{-\infty}^{\infty} \varphi_p^*(\mathbf{r}_1) \varphi_q^*(\mathbf{r}_2) \frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|} \varphi_r(\mathbf{r}_1) \varphi_s(\mathbf{r}_2) d\mathbf{r}_1 d\mathbf{r}_2, \quad (1.8)$$

ezt a jelölésmódot $\langle 12|12 \rangle$ konvenciónak nevezzük. Használni fogjuk még a $(pq|rs)$ jelölést $[(11|22)$ konvenció]:

$$(pq|rs) = \langle pr|qs \rangle. \quad (1.9)$$

Ezek az integrálok a következő szimmetriákkal rendelkeznek:

$$\begin{aligned} \langle pq|rs \rangle &= \langle rq|ps \rangle = \langle ps|rq \rangle = \langle rs|pq \rangle \\ &= \langle qp|sr \rangle = \langle sp|qr \rangle = \langle qr|sp \rangle = \langle sr|qp \rangle \end{aligned} \quad (1.10)$$

és

$$\begin{aligned} (pq|rs) &= (qp|rs) = (pq|sr) = (qp|sr) \\ &= (rs|pq) = (sr|pq) = (rs|qp) = (sr|qp). \end{aligned} \quad (1.11)$$

Bevezetjük továbbá a Fock-operátor másodkvantált alakját:

$$\hat{F} = \sum_{pq} f_{pq} p^+ q^-, \quad (1.12)$$

ahol f_{pq} -vel jelöltük a Fock-mátrix elemeit. A Fermi vákuumra vett normálrendezett alakját az

$$\hat{F}_N = \sum_{pq} f_{pq} \{p^+ q^-\} \quad (1.13)$$

egyenlettel adjuk meg, ahol a $\{\cdot\}$ jelöli a normálrendezést. Bevezetjük a Hamilton-operátor Fermi vákuumra vett normálrendezett alakját:

$$\hat{H}_N = \hat{F}_N + \frac{1}{4} \sum_{pqrs} \langle pq||rs \rangle \{p^+ q^+ s^- r^-\}. \quad (1.14)$$

Helyettesítsük be az (1.1) hullámfüggvényt a Schrödinger-egyenletbe, szorozzuk meg balról az egyenletet $e^{-\hat{T}}$ -vel! Az így kapott egyenletet a $|0\rangle$ függvénnyel projektálva megkapjuk az energiát:

$$\langle 0 | e^{-\hat{T}} \hat{H} e^{\hat{T}} | 0 \rangle = E. \quad (1.15)$$

Ha a $|0\rangle$ helyett a $|\Psi_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k}\rangle = a_1^+ i_1^- a_2^+ i_2^- \dots a_k^+ i_k^- |0\rangle$ k -szorosán gerjesztett determinánsokkal projektáljuk az egyenletet, megkapjuk az ún. energiafüggetlen CC egyenleteket:

$$\langle \Psi_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k} | e^{-\hat{T}} \hat{H} e^{\hat{T}} | 0 \rangle = 0 \quad (k = 1, \dots, n), \quad (1.16)$$

ahol n a legmagasabb gerjesztés, amit figyelembe veszünk.

Legyen most $|0\rangle$ a Fermi-vákuum és írjuk át az (1.15)-(1.16) egyenleteket a Hamilton-operátor Fermi vákuumra nézve normálrendezett alakjával ($\hat{H}_N$). Jelölje E_0 a Hamilton-operátor várható értékét a $|0\rangle$ függvénnyel, ekkor $\hat{H} = \hat{H}_N + E_0$. Ezt behelyettesítve átírhatjuk a CC egyenleteket a

$$\langle 0 | e^{-\hat{T}} \hat{H}_N e^{\hat{T}} | 0 \rangle = E_{CC}, \quad (1.17)$$

$$\langle \Psi_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k} | e^{-\hat{T}} \hat{H}_N e^{\hat{T}} | 0 \rangle = 0 \quad (k = 1, \dots, n) \quad (1.18)$$

formába, ahol E_{CC} a korrelációs energia és $E = E_0 + E_{CC}$. Felhasználhatjuk továbbá, hogy az $e^{-\hat{T}} \hat{H}_N e^{\hat{T}}$ operátor – amit a továbbiakban $\hat{\mathcal{H}}$ -pal jelölünk – csak kapcsolt (connected) tagokból áll, azaz

$$\begin{aligned} \hat{\mathcal{H}} &= e^{-\hat{T}} \hat{H}_N e^{\hat{T}} = [\hat{H}_N e^{\hat{T}}]_C \\ &= \left[\hat{H}_N \left(1 + \hat{T}_1 + \hat{T}_2 + \hat{T}_3 + \dots \right. \right. \\ &\quad \left. \left. + \frac{1}{2!} \hat{T}_1^2 + \hat{T}_1 \hat{T}_2 + \frac{1}{3!} \hat{T}_1^3 + \hat{T}_1 \hat{T}_3 + \dots \right) \right]_C. \end{aligned} \quad (1.19)$$

Mivel a Hamilton operátor csak egy- és kételektron operátorokat tartalmaz, belátható, hogy az exponenciális operátor sorfejtésében csak a legfeljebb negyedik hatványú tagok adhatnak járulékot az egyenletekhez [4, 48, 49]. A klaszteramplitúdókra ilyen módon megkapott egyenletek nem lineárisak, ezért iterációval oldjuk meg őket.

1.3. Egy- és kétszeres gerjesztések

Az általunk használt módszer esetén $n = 2$ a legmagasabb gerjesztés, ez a CCSD (CC singles and doubles) módszer. Ekkor

$$\langle 0 | \left[\hat{H}_N (1 + \hat{T}_1 + \hat{T}_2 + \frac{1}{2} \hat{T}_1^2) \right]_C | 0 \rangle = E_{CC}, \quad (1.20)$$

$$\langle \Psi_i^a | \left[\hat{H}_N \left(1 + \hat{T}_1 + \hat{T}_2 + \frac{1}{2} \hat{T}_1^2 + \hat{T}_1 \hat{T}_2 + \frac{1}{3} \hat{T}_1^3 \right) \right]_C | 0 \rangle = 0, \quad (1.21)$$

$$\langle \Psi_{ij}^{ab} | \left[\hat{H}_N \left(1 + \hat{T}_1 + \hat{T}_2 + \frac{1}{2} \hat{T}_1^2 + \hat{T}_1 \hat{T}_2 + \frac{1}{3!} \hat{T}_1^3 \frac{1}{2} \hat{T}_2^2 + \frac{1}{4!} \hat{T}_1^4 \right) \right]_C | 0 \rangle = 0 \quad (1.22)$$

formában írhatók fel az egyenletek [50]. A CC módszer pontosabban adja meg a korrelációs energiát, mint a többi korrelációs módszer [perturbációs számítás, sűrűségfukcionál elmélet, CI (configuration interaction) módszer], viszont a számításigénye a rendszer méretével rohamosan nő. Azért előnyös csak egy- és kétszeres gerjesztéseket figyelembe venni, mert ekkor a számításigény még nem túl nagy (a számításigény kb. $n_o^2 n_v^4$, ahol n_o a betöltött pályák száma, n_v pedig a virtuálisaké), viszont a korrelációs energiát elég pontosan megkapjuk. Ha magasabb gerjesztéseket is figyelembe vennénk a számításigény még nagyobb lenne. A háromszoros gerjesztéseket figyelembe vehetjük egy perturbatív eljárással, aminek a számításigénye lényegesen kisebb, mintha a klaszteroperátorban a háromszorosan gerjesztő operátor is szerepelne.

1.4. Perturbatív korrekció

A perturbációs számítás során a Hamilton-operátort két részre osztjuk [49]:

$$\hat{H} = \hat{H}_0 + \hat{V}, \quad (1.23)$$

ahol $\hat{H}_0$ -al a nulladrendű Hamilton-operátort és $\hat{V}$ -vel a perturbációs operátort jelöljük. Az energiát valamint a $\hat{T}$ operátort felbontjuk a perturbáció rendje szerint, amit szögletes zárójelben adunk meg:

$$E = E^{[0]} + E^{[1]} + E^{[2]} + \dots \quad (1.24)$$

$$\hat{T} = \hat{T}^{[0]} + \hat{T}^{[1]} + \hat{T}^{[2]} + \dots \quad (1.25)$$

Behelyettesítjük az energia és a klaszteroperátor fenti kifejtését az effektív Hamilton-operátor ($\hat{\mathcal{H}}$) kifejezésébe. Ekkor ez az operátor szintén felírható egy perturbációs sor formájában:

$$\hat{\mathcal{H}} = e^{-\hat{T}} \hat{H} e^{\hat{T}} = \hat{\mathcal{H}}^{[0]} + \hat{\mathcal{H}}^{[1]} + \hat{\mathcal{H}}^{[2]} + \dots \quad (1.26)$$

Az azonos rendű tagokat összegyűjtjük és elvégezzük a levezetést. A nulladrendű effektív Hamilton-operátorra a $\hat{H}_0$ -at kapjuk, az elsőrendű tagot a

$$\hat{\mathcal{H}}^{[1]} = \left[\hat{H}_0, \hat{T}^{[1]} \right] + \hat{V} \quad (1.27)$$

egyenlet adja meg, míg a másodrendű tagot a

$$\hat{\mathcal{H}}^{[2]} = [\hat{H}_0, \hat{T}^{[2]}] + [\hat{V}, \hat{T}^{[1]}] \quad (1.28)$$

összefüggés definiálja. Általános alakban pedig az n -edrendű Hamilton-operátor:

$$\hat{\mathcal{H}}^{[n]} = [\hat{H}_0, \hat{T}^{[n]}] + [\hat{V}, \hat{T}^{[n-1]}] + [[\hat{V}, \hat{T}^{[1]}], \hat{T}^{[n-2]}] + \dots \quad (1.29)$$

Az n -ed rendű klaszteramplitúdókra úgy kaphatunk egyenleteket, ha a CC egyenletekbe a teljes effektív Hamilton-operátor helyett $\hat{\mathcal{H}}^{[n]}$ -t helyettesítjük:

$$\langle \Psi_{a_1 a_2 \dots}^{i_1 i_2 \dots} | \hat{\mathcal{H}}^{[n]} | 0 \rangle = 0. \quad (1.30)$$

A CC energiához a következőképpen vezetünk le perturbációs korrekciókat. A közelítés az n -ed rendű energián alapul, amit megkaphatunk, ha a CC energiakifejezésben az effektív Hamilton-operátort $\hat{\mathcal{H}}^{[n]}$ -re cseréljük:

$$E^{[n]} = \langle 0 | \hat{\mathcal{H}}^{[n]} | 0 \rangle = \langle 0 | [\hat{V}, \hat{T}_2^{[n-1]}] + [[\hat{V}, \hat{T}_1^{[2]}], \hat{T}_1^{[n-3]}] + \dots | 0 \rangle. \quad (1.31)$$

Vegyük észre, hogy az energia csak az egyszeres és kétszeres gerjesztések (különböző perturbációs rendű) amplitúdóitól függ.

Ha kanonikus HF [RHF (restricted Hartree–Fock) vagy UHF (unrestricted Hartree–Fock)] pályákat használunk a referenciaállapothoz, akkor a nulladrendű Hamilton-operátor a Fock-operátor diagonális része:

$$\hat{H}_0 = \hat{F}^{aa} + \hat{F}^{ii} = \sum_a f_{aa} a^+ a^- + \sum_i f_{ii} i^+ i^-, \quad (1.32)$$

ahol f_{aa} és f_{ii} a Fock-mátrix diagonális elemei, a perturbációs operátor pedig a normálsorrendű Hamilton-operátor kételektronos része:

$$\hat{V} = \hat{H}_N - \hat{H}_0. \quad (1.33)$$

Mivel $\hat{H}_0$ diagonális az n -ed rendű klaszteramplitúdókat egyszerűen a

$$\hat{T}_k^{[n]} = \hat{R}_k \left(\hat{\mathcal{H}}^{[n]} - [\hat{H}_0, \hat{T}^{[n]}] \right) \quad (1.34)$$

képlet adja meg, ahol

$$\hat{R}_k(\hat{X}) = \sum_{\substack{a_1 < a_2 \dots < a_k \\ i_1 < i_2 \dots < i_k}} \frac{\langle \Psi_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k} | \hat{X} | 0 \rangle}{D_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k}} a_1^+ i_1^- a_2^+ i_2^- \dots a_k^+ i_k^- \quad (1.35)$$

a rezolvens operátor, $D_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k}$ pedig a

$$D_{a_1 a_2 \dots a_k}^{i_1 i_2 \dots i_k} = \sum_{j=1}^k f_{i_j i_j} - \sum_{j=1}^k f_{a_j a_j} \quad (1.36)$$

képlettel definiált MBPT (many-body perturbation theory) energianevező. A fenti és az (1.27) és (1.28) egyenletekből a következő egyszerű képleteket kapjuk az első- és másodrendű klaszteramplitúdókra:

$$\hat{T}_2^{[1]} = \hat{R}_2(\hat{V}) \quad (1.37)$$

$$\hat{T}_1^{[2]} = \hat{R}_1(\hat{V}\hat{T}_2^{[1]}) \quad (1.38)$$

$$\hat{T}_3^{[2]} = \hat{R}_3(\hat{V}\hat{T}_2^{[1]}). \quad (1.39)$$

Most vezessük le a CCSD módszerhez egy korrekciót a háromszoros gerjesztések figyelembevételével, az (1.31) egyenlet alapján! Ekkor a legalacsonyabb, azaz másodrendű $\hat{T}_3$ klaszter ($\hat{T}_3^{[2]}$) járulékát kell kiszámítanunk a kétszeres gerjesztések amplitúdóihoz és közvetve az energiához. (1.34) és (1.29) alapján $\hat{T}_3^{[2]}$ a harmadrendű $\hat{T}_2$ -höz ad járulékot, amelyre a

$$\hat{T}_2^{[3]} = \hat{R}_2(\hat{V}\hat{T}_3^{[2]}) \quad (1.40)$$

egyenletet kapjuk. (1.31) alapján $\hat{T}_2^{[3]}$ a negyedrendű energiához járul hozzá:

$$E^{[4]} = \langle 0 | \hat{V} \hat{T}_2^{[3]} | 0 \rangle = \langle 0 | \hat{V} \hat{R}_2(\hat{V}\hat{T}_3^{[2]}) | 0 \rangle = \langle 0 | (T_2^{[1]})^\dagger \hat{V} \hat{T}_3^{[2]} | 0 \rangle \quad (1.41)$$

Ha a negyedrendű energia kifejezésében az elsőrendű $T_2^{[1]}$ -t a konvergált CCSD $\hat{T}_2$ -vel helyettesítjük, a CCSD[T] módszer

$$\Delta E_{\text{CCSD[T]}} = \langle 0 | T_2^\dagger \hat{V} \hat{T}_3^{\{2\}} | 0 \rangle \quad (1.42)$$

korrekciós tagját kapjuk [16], ahol a kapcsos zárójel arra utal, hogy az alacsonyabb rendű perturbatív klaszteramplitúdókat a konvergáltakra cseréltük.

Numerikus eredmények bizonyítják, itt nem részletezett okokból, hogy egy jobb korrekciót kapunk, ha a fenti korrekciót a

$$\Delta E_{\text{CCSD(T)}} = \Delta E_{\text{CCSD[T]}} + \langle 0 | T_1^\dagger \hat{V} \hat{T}_3^{\{2\}} | 0 \rangle \quad (1.43)$$

egyenlet szerint módosítjuk [17, 51]. Ezt CCSD(T) módszernek hívjuk, ezzel már 1kcal/mol-os pontosságot érhetünk el.

Ennek a korrekciónak a számításgénye kb. $n_o^3 n_v^4$, azaz a rendszer méretének hetedik hatványával skálázódik. Mivel a CCSD és CCSD(T) módszerek műveletigénye a rendszer méretével rohamosan nő, nagy rendszerekre a számítások nagy erőforrást és sok időt igényelnek. Ennek elkerülése érdekében lokális közelítéseket használunk, amely segítségével lineáris skálázódást értünk el.

1.5. A lokális coupled-cluster elmélet

Minden CC módszer esetében – így a CCSD módszernél is – a CC korrelációs energiát az

$$E_{\text{CC}} = \sum_{ia} f_{ai} t_a^i + \frac{1}{4} \sum_{abij} \langle ij || ab \rangle \tau_{ab}^{ij} \quad (1.44)$$

képlettel számíthatjuk, ahol $\tau_{ab}^{ij} = t_{ab}^{ij} + t_a^i t_b^j - t_a^j t_b^i$. A korrelációs energia nem változik, ha a betöltött, illetve a virtuális pályákat unitér módon transzformáljuk, ezért feltehető, hogy a betöltött pályák lokalizáltak. A fenti kifejezést átírjuk az egyes lokalizált pályákhoz tartozó járulékok összegeként:

$$E_{\text{CC}} = \sum_i \delta E_i, \quad (1.45)$$

ahol

$$\delta E_i = \sum_a f_{ai} t_a^i + \frac{1}{4} \sum_{abj} \langle ij || ab \rangle \tau_{ab}^{ij}. \quad (1.46)$$

A δE_i parciális energia is invariáns a virtuális-virtuális valamint az i . pályát nem érintő betöltött-betöltött forgatásokra. Emiatt minden egyes δE_i járulékot egy külön bázisban számíthatunk. A bázist úgy célszerű megválasztani, hogy könnyen meghatározhatók legyenek azok a pályák, amelyek valamilyen szempontból erősen kölcsönhatnak az i . pályával. Jelöljük az ilyen pályák halmazát P_i -vel. Ekkor a

$$\delta E_i = \sum_{a \in P_i} f_{ai} t_a^i + \frac{1}{4} \sum_{abj \in P_i} \langle ij || ab \rangle \tau_{ab}^{ij} \quad (1.47)$$

képlettel számítva jó közelítéssel kapjuk meg δE_i -t. P_i -t az i . pálya lokális kölcsönható alterének nevezzük. A fenti egyenletben a klaszteramplitúdók csak a P_i tér pályáinak indexeitől függenek, elég megoldanunk a P_i téren belül a CC egyenleteket, külön az összes i -re. Mivel a CC módszerek skálázódása gyorsan nő a pályák számával, és a P_i alterekben feltehetően jóval kevesebb pálya van, mint az eredeti teljes térben, a CC egyenletek megoldása sokkal gyorsabb lesz.

Tehát meg kell keresni a lehető legkisebb lokális kölcsönható alteret minden lokalizált pályához úgy, hogy ne okozzon számottevő hibát a bázis méretének csökkentése. Ehhez itt nem részletezett módon [52] MP2 (másodrendű Møller–Plesset módszer) természetes pályákat állítunk elő. A sűrűségmátrixok sajátértékeit (a természetes pályák betöltési számai) vizsgáljuk. Ha az adott pályához tartozó sajátérték nagyobb, mint egy paraméterként megadott érték

ε , akkor azt a pályát bevesszük P_i altérbe, ha nem akkor befagyasztjuk. Egy lokális kölcsönható altér felépítése után az altér pályáit a lokális kanonikus bázisba transzformáljuk [45], amelyben az adott altéren számított Fock-mátrix diagonális, és a δE_i parciális energia számítását ezen a bázison végezzük. Lokális kanonikus bázist egyrészt azért előnyös használnunk, mert a perturbatív CC módszerek [pl. CCSD(T)] kanonikus bázisra levezetett kifejezéseit itt is használhatjuk. Másrészt a kanonikus bázisban gyorsabb az iteratív CC számítások konvergenciája. Mivel a lokális kölcsönható alterek mérete várhatóan kicsi, az altéren belüli integráltranszformáció számításigénye sem lesz magas.

Azonban a P_i altéren belüli kanonikus transzformáció összekeveri az i . pályát a többivel, ezért az (1.47) energiakifejezés számítása magyarázatra szorul. Kihasználva az egyenlet unitér invarianciáját átírhatjuk azt a

$$\delta E_i = \sum_{a'i'j' \in P_i} U_{ii'} U_{ij'} f_{a'i'} t_{a'}^{j'} + \frac{1}{4} \sum_{a'b'i'j'k' \in P_i} U_{ii'} U_{ik'} \langle i'j' || a'b' \rangle \tau_{a'b'}^{k'j'} \quad (1.48)$$

alakba, ahol az $\mathbf{U}$ mátrix forgatja az i' kanonikus pályákat az i . eredeti pályába:

$$|i\rangle = \sum_{i'} U_{ii'} |i'\rangle. \quad (1.49)$$

A fenti kifejezés alapján látszik, hogy legegyszerűbb, ha a CC egyenletek megoldása után a $t_{a'}^{i'}$ és $t_{a'b'}^{i'j'}$ klaszteramplitúdók, Fock-mátrix elemek és az $\langle i'j' || a'b' \rangle$ integrállista egyik betöltött indexét visszaforgatjuk az i . pályába, és a δE_i járulékot a

$$\delta E_i = \sum_{a' \in P_i} f_{a'i} t_i^{a'} + \frac{1}{4} \sum_{a'b'j' \in P_i} \langle ij' || a'b' \rangle \tau_{ij'}^{a'b'} \quad (1.50)$$

kifejezéssel számítjuk. Fontos megjegyezni, hogy az $\varepsilon = 0$ esetben a lokális kölcsönható alterek a teljes egyelektron bázis tartalmazzák, és így a δE_i járulékok összegzésével visszkapjuk a teljes CC korrelációs energiát.

A perturbatív CC közelítések energiaképlete különbözik az (1.44) egyenlettől, de a fenti stratégia bármilyen energiakifejezésre adaptálható, amely a betöltött pályákra való összegzést tartalmaz. Így a háromszoros gerjesztéseket perturbatíván kezelő módszerek korrekciós tagja általánosan az

$$E^T = \sum_{abcijk} L_{abc}^{ijk} T_{abc}^{ijk}, \quad (1.51)$$

alakba írható. Például a CCSD[T] módszer esetében [vö. (1.42) egyenlet] T_{abc}^{ijk} a $\hat{T}_3^{\{2\}}$ operátor megfelelő amplitúdója, míg $L_{abc}^{ijk} = \langle 0 | T_2^\dagger \hat{V} | \Psi_{ijk}^{abc} \rangle$. Az (1.51)

kifejezést könnyen felbonthatjuk a betöltött pályák

$$\delta E_i^T = \sum_{abcjk} L_{abc}^{ijk} T_{abc}^{ijk} \quad (1.52)$$

járulékainak összegére, ahol az L_{abc}^{ijk} és T_{abc}^{ijk} mennyiségeket ismét a megfelelő lokális kölcsönható alterekben számítjuk. Az altereken belül itt is célszerű áttérni kanonikus bázisra, ekkor ugyanis a T_{abc}^{ijk} amplitúdók kiszámításához nincs szükség iteratív eljárásra. Ezért a δE_i^T járulékokat gyakorlatban a

$$\delta E_i^T = \sum_{a'b'c'i'l'j'k'l'} U_{ii'} U_{il'} L_{a'b'c'}^{i'j'k'} T_{a'b'c'}^{l'j'k'} = \sum_{a'b'c'j'k'} L_{a'b'c'}^{ij'k'} T_{a'b'c'}^{ij'k'} \quad (1.53)$$

kifejezés segítségével számítjuk.

2. fejezet

A CCSD egyenletek és implementálásuk

2.1. A faktorizált egyenletek

A Werner és munkatársai közleményében található egyenleteket használjuk [50]. Ez azért előnyös számunkra, mert itt már mátrix- és vektorszorzások formájában adták meg az egyenleteket, amiket a BLAS rutinokkal tudunk számolni lásd: (2.2.2). Az alábbiakban megadjuk a közleményben található egyenleteket. A közlemény jelöléseit használjuk a kódban is, az eltéréseket az implementáció résznél részletezzük. A félkövér betűk mátrixokat, ill. vektorokat jelölnek, a dőlt betűk skalárokat, ill. mátrix- és vektorelemeket jelölnek, az írógép betűtípussal szedett betűk a kódban szereplő jelöléseket jelentik.

A $\mathbf{J}^{kl}$ ún. belső Coulomb és $\mathbf{K}^{kl}$ kicserélődési mátrixokat a

$$J_{ab}^{kl} = (kl|ab), \quad (2.1)$$

$$K_{ab}^{kl} = (ak|lb) \quad (2.2)$$

módon definiálunk. Bevezetjük továbbá az $\mathbf{L}^{kl}$ mátrixot, amit az

$$L_{ab}^{kl} = 2(ak|lb) - (al|kb) \quad (2.3)$$

képlettel definiálunk. Az RHF Fock-mátrixot $\mathbf{F}$ -fel jelöljük és a

$$F_{pq} = h_{pq} + \sum_i [2(pq|ii) - (pi|i q)] \quad (2.4)$$

összefüggéssel definiáljuk, ahol a $\mathbf{h}$ az egyelektron integrál mátrix. Az egy virtuális indexű oszlopvektorait $\mathbf{f}^k$ -val jelöljük és az

$$f_a^k = F_{ak} \quad (2.5)$$

módon definiáljuk. Ehhez hasonlóan a $\mathbf{k}^{kli}$ és $\mathbf{l}^{kli}$ vektorokat

$$k_a^{kli} = (ak|li), \quad (2.6)$$

$$l_a^{kli} = 2(ak|li) - (al|ki) \quad (2.7)$$

képletekkel definiáljuk. Az általánosított $\mathbf{J}\mathbf{E}^{ij}$ Coulomb és $\mathbf{K}\mathbf{E}^{ij}$ kicserélődési mátrixokat a

$$JE_{ab}^{ij} = \sum_c (ab|ic)t_c^j, \quad (2.8)$$

$$KE_{ab}^{ij} = \sum_c (ai|bc)t_c^j \quad (2.9)$$

módon adjuk meg. Az effektív T_2 amplitúdókat a

$$\mathbf{C}^{ij} = \mathbf{T}^{ij} + \mathbf{t}^i \mathbf{t}^{j\dagger} \quad (2.10)$$

$$\bar{\mathbf{C}}^{ij} = \frac{1}{2} \mathbf{T}^{ij} + \mathbf{t}^i \mathbf{t}^{j\dagger}, \quad (2.11)$$

$$\mathbf{D}^{ij} = \mathbf{C}^{ij} + \mathbf{E}^{ij} + \mathbf{E}^{ji\dagger} \quad (2.12)$$

egyenletekkel vezetjük be, ahol az $\mathbf{E}^{ij}$ mátrixokat az

$$E_{pa}^{ij} = \delta_{pi} t_a^j \quad (2.13)$$

összefüggéssel definiáljuk. Az ún. külső kicserélődési mátrixot a

$$KD_{pq}^{ij} = \sum_{rs} D_{rs}^{ij} (pr|sq) \quad (2.14)$$

egyenlettel definiáljuk. Ennek a három betöltött indexű részét $\mathbf{k}\mathbf{d}^{ijk}$ vektorral jelöljük és a

$$kd_a^{ijk} = KD_{ai}^{jk} \quad (2.15)$$

képlettel definiáljuk, a két virtuális indexű részét $\mathbf{K}\mathbf{D}\mathbf{V}^{ij}$ -vel jelöljük és a

$$KDV_{ab}^{ij} = KD_{ab}^{ij} \quad (2.16)$$

képlettel definiáljuk.

Bevezetjük az

$$\alpha_{ij,kl} = (ij|kl) + \text{tr}(\mathbf{C}^{ij}\mathbf{K}^{lk}) + \mathbf{t}^{i\dagger}\mathbf{k}^{klj} + \mathbf{t}^{i\dagger}\mathbf{k}^{klj}, \quad (2.17)$$

$$\beta_{ki} = F_{ki} + \sum_l [\text{tr}(\mathbf{L}^{kl}\mathbf{C}^{li}) + \mathbf{t}^{l\dagger}\mathbf{k}^{lki}] + \mathbf{f}^{k\dagger}\mathbf{t}^i \quad (2.18)$$

skalár, a

$$\mathbf{r}^k = \mathbf{f}^k + \sum_l \mathbf{L}^{kl} \mathbf{t}^l, \quad (2.19)$$

$$\begin{aligned} \mathbf{s}^i &= \mathbf{f}^i + (\mathbf{h} - \mathbf{A}^\dagger) \mathbf{t}^i - \sum_{kl} \mathbf{T}^{lk} \mathbf{l}^{kli} \\ &+ \sum_k (2\mathbf{k} \mathbf{d}^{kik} - \mathbf{k} \mathbf{d}^{kki}) \end{aligned} \quad (2.20)$$

vektor és az

$$\mathbf{A} = \sum_{kl} \mathbf{L}^{kl} \mathbf{T}^{lk}, \quad (2.21)$$

$$\mathbf{X} = \mathbf{F} - \mathbf{A} + \sum_k (2\mathbf{J} \mathbf{E}^{kk} - \mathbf{K} \mathbf{E}^{kk} - \mathbf{r}^k \mathbf{t}^{k\dagger}), \quad (2.22)$$

$$\begin{aligned} \mathbf{Y}^{kj} &= \mathbf{K}^{kj} + \mathbf{K} \mathbf{E}^{kj} - \frac{1}{2} (\mathbf{J}^{kj} + \mathbf{J} \mathbf{E}^{kj}) \\ &+ \frac{1}{2} \sum_l \mathbf{L}^{kl} (2\bar{\mathbf{C}}^{lj} - \bar{\mathbf{C}}^{jl}) - \frac{1}{2} \sum_l \mathbf{l}^{lkj} \mathbf{t}^{l\dagger} + \mathbf{f}^k \mathbf{t}^{j\dagger}, \end{aligned} \quad (2.23)$$

$$\mathbf{Z}^{kj} = \mathbf{J}^{kj} + \mathbf{J} \mathbf{E}^{kj} - \sum_l [\mathbf{K}^{lk} \bar{\mathbf{C}}^{jl} + \mathbf{k}^{lkj} \mathbf{t}^{l\dagger}], \quad (2.24)$$

$$\begin{aligned} \mathbf{G}^{ij} &= \mathbf{s}^i \mathbf{t}^{j\dagger} - \sum_k (\mathbf{k} \mathbf{d}^{kij} + \mathbf{k}^{ijk}) \mathbf{t}^{k\dagger} - \sum_k \beta_{ki} \mathbf{C}^{kj} \\ &+ \mathbf{T}^{ij} \mathbf{X} + \sum_k [(2\mathbf{T}^{ik} - \mathbf{T}^{ki}) \mathbf{Y}^{kj} - \frac{1}{2} \mathbf{T}^{ki} \mathbf{Z}^{kj} - (\mathbf{T}^{ki} \mathbf{Z}^{kj})^\dagger] \end{aligned} \quad (2.25)$$

mátrix intermediereket. A $\mathbf{v}^i$ és $\mathbf{V}^{ij}$ reziduális vektort és mátrixot a

$$\mathbf{v}^i = \mathbf{s}^i + \sum_k [(2\mathbf{T}^{ik} - \mathbf{T}^{ki}) \mathbf{r}^k - \beta_{ki} \mathbf{t}^k], \quad (2.26)$$

$$\mathbf{V}^{ij} = \mathbf{K}^{ij} + \mathbf{K} \mathbf{D} \mathbf{V}^{ij} + \sum_{kl} \alpha_{ij,kl} \mathbf{C}^{kl} + \mathbf{G}^{ij} + \mathbf{G}^{ji\dagger} \quad (2.27)$$

egyenletekkel adjuk meg.

A korrelációs energiát az

$$E_{\text{CCSD}} = 2 \sum_{ai} f_a^i t_a^i + \sum_{abij} L_{ab}^{ij} C_{ab}^{ij} \quad (2.28)$$

egyenlet adja meg.

2.2. Az implementáció

2.2.1. Bevezetés

A közlemény egyenleteit lokális CC számolásokra implementáltuk. A közleményben atompálya integrálokra épülő egyenletek vannak megadva, de nekünk molekulapálya integrálokra épülő kódra van szükségünk, ezért az egyenleteket át kell írunk ennek megfelelően. A gyakorlatban a lokális CC számolások során egy doménen belül használt betöltött pályák száma nem haladja meg az ötvenet, a bázisfüggvények száma pedig legfeljebb háromszáz. Egy a napjainkban használatos számítógép legalább 8 GB memóriával rendelkezik, ezért a programkód írásánál ezt feltételeztük. Ahhoz, hogy egy ilyen számítógépen effektíven számoljunk, minimalizálni kell az I/O műveletek számát, viszont a rendelkezésre álló memóriát ésszerűen be kell osztanunk.

2.2.2. A felhasznált BLAS rutinok

A közleményben megadott egyenletek vektorizáltak, azaz mátrix-mátrix, mátrix-vektor szorzások, ill. vektorok diadikus szorzataként adták meg őket. Ezeket a műveleteket előre megírt és a számítógépek erőforrásainak maximális kihasználására optimalizált BLAS rutinokkal végezzük el. A BLAS a „Basic Linear Algebra Subprograms” rövidítése, azaz „egyszerű lineáris algebrai alprogramok”.

A (2.1) táblázat tartalmazza a programban használt rutinok nevét és rövid ismertetését. A rutinok kezdőbetűje a duplapontosságú számok használatára utal.

2.2.3. A memória kezelése

A (2.2) táblázatban összefoglaljuk a fellépő tömbök tárolási helyét és a kódban használt elnevezését. A táblázatban a „mem” azt jelöli, hogy a memóriában tároljuk az adott tömböt, a „diszk” pedig, hogy a merevlemezen. Ha egy adott tömb mellett egy másik tömb nevét találjuk, akkor azt annak a helyén tároljuk, ill. közvetlenül oda számoljuk ki. A „mem*” azt jelöli, hogy csak ideiglenesen tároljuk az adott tömböt a memóriában.

Az összes memóriában levő tömbnél a virtuális index megelőzi a betöltöttet, mivel a FORTRAN konvenciója szerint oszlop folytonosan tárolódnak a tömbök és a mátrixszorzásoknál az első index (sorindex) a gyorsabb. Azaz például az $\mathbf{f}^i$ vektor a . elemére az $\mathbf{fai}(a, i)$ módon hivatkozhatunk.

A $\mathbf{C}^{ij}$ -t és a $\bar{\mathbf{C}}^{ij}$ -t minden alkalommal felépítjük az amplitúdókból az adott ij indexekre. Erre a célra létrehozunk egy $n_v \times n_v$ méretű tömböt, amit `scrs-`

2.1. táblázat. BLAS rutinok

rutin neve	leírás
dcopy	egy vektort másol egy másik vektorba
dscal	egy vektor elemeit megszorozza egy számmal
daxpy	egy vektor számszorosát adja hozzá egy másik vektorhoz
ddot	kiszámolja két vektor skaláris szorzatát, az eredmény egy szám
dnrm2	egy vektor euklideszi normáját számolja ki, az eredmény egy szám
dger	kiszámolja két vektor diadikus szorzatát, az eredmény egy mátrix
dgemv	kiszámolja egy mátrix és egy vektor szorzatának számszorosát, és hozzáadja egy másik vektor számszorosához
dgemm	kiszámolja két mátrix szorzatának számszorosát, és hozzáadja egy másik mátrix számszorosához
dsymm	kiszámolja egy szimmetrikus mátrix és egy másik mátrix szorzatának számszorosát úgy, hogy az első mátrixnak csak az alsó vagy felső háromszögét használja, és hozzáadja egy harmadik mátrix számszorosához

nek nevezzük. A

$$T_{ab}^{ij} = T_{ba}^{ji}, \quad (2.29)$$

$$V_{ab}^{ij} = V_{ba}^{ji} \quad (2.30)$$

szimmetria tulajdonságok miatt a $\mathbf{T}^{ij}$ és $\mathbf{V}^{ij}$ mátrixok közül elég csak azokat tárolni, ahol az $i \leq j$ megszorítás teljesül. Az indexekre való megkötést kihasználva a négy indexes tömbök helyett három indexes tömböket vezetünk be a

$$T_{ab}^{ij} \rightarrow \text{tt}(\mathbf{a}, \mathbf{b}, j * (j - 1)/2 + i) \quad (2.31)$$

módon, ahol a harmadik indexet a fenti képlettel definiáljuk. $\mathbf{V}$ -nél hasonlóan járunk el. Ha a $\mathbf{T}^{ij}$ mátrixra van szükségünk abban az esetben, amikor $i > j$, akkor a $\mathbf{T}^{ji}$ mátrix transzponáltját használjuk. A fenti megfontolásokat követve csak kétindexes mátrixokat tárolunk a memóriában, ill. a $\mathbf{T}$ -t és a $\mathbf{V}$ -t, amik teljes mérete kb. $n_o^2 n_v^2$.

A program indulásakor lefoglaljuk a felhasználó által megadott méretű memóriát. Ezt a területet úgy tekintjük, mint egy nagy egyindexes tömb és ezt osztjuk fel úgy, hogy az elejétől kezdve az egyes tömböknek megfelelő méretű részt rendelünk hozzá a tömbökhöz. A tömbök sorrendje a (2.2) táblázatban található meg.

2.2. táblázat. Tömbök elnevezése és tárolása		
név az egyenletekben	név a kódban	tárolás
-	scrs	mem
-	bmat	mem
-	bvec	mem
-	invbmat	mem
-	cvec	mem
h	h	mem
F_{ij}	fij	mem
F_{ab}	fab	mem
f_a^i	fai	mem
β	beta	mem
X	x	mem
r	rr	mem
-	w	mem
v	v	mem
V	vv	mem
t	t	mem
T	tt	mem
s	ss	v
KDV	kdv	vv
kd	kdo	vv
α	alpha	mem*
G	g	vv
JE	je	diszk
KE	ke	diszk
Y	y	diszk
Z	z	diszk
A	aa	x
L	ll	diszk
l	ls	diszk
K	kk	diszk
k	ks	diszk
J	jj	diszk
C	cc	-
$\bar{C}$	co	-

2.2.4. Az integrállisták tárolása

A kételektron integrállistákat minden esetben az $\langle 12|12 \rangle$ konvenciót használva tároljuk. A program a merevlemezre írva kapja meg az $\langle ij|kl \rangle$ ($i \geq k, j \geq$

$l, ij \geq kl$, $\langle ai|jk \rangle$ ($i \geq k$), $\langle ab|ij \rangle, \langle ai|bj \rangle$ ($a \geq b, i \geq j$), $\langle ab|ci \rangle$ ($a \geq c$) integrállistákat a zárójelben megadott megkötésekkel úgy, hogy az első három index együtt van minden blokkban, amit az utolsó index jelöl. Az egyes olvasásoknál így egy három indexes blokkot lehet olvasni a memóriába és azt ott kicsomagolni, azaz feloldani az indexekre a megkötéseket. A blokkok olvasására fenntartunk egy w tömböt a memóriában. Az $\langle ab|cd \rangle$ integrálokat két külön listában tároljuk, ennek az okára később térünk ki.

2.2.5. A módosított egyenletek

A fenti egyenleteket a következőképpen módosítottuk. A (2.12) és (2.13) egyenletet beírjuk a (2.14) egyenletbe, elvégezzük az összegzést a Kronecker deltát tartalmazó tagra. Ekkor a

$$KDV_{ab}^{ij} = \sum_c (\langle ab|ic \rangle t_c^j + \langle ab|cj \rangle t_c^i) + \sum_{cd} \langle ab|cd \rangle C_{cd}^{ij}, \quad (2.32)$$

$$kd_a^{kij} = \sum_b (\langle ak|ib \rangle t_b^j + \langle ak|bj \rangle t_b^i) + \sum_{bc} \langle ak|bc \rangle C_{bc}^{ij} \quad (2.33)$$

mátrixokat kapjuk, így $\mathbf{KD}^{ij}$ helyett a $\mathbf{KDV}^{ij}$ -t és a $\mathbf{kd}^{kij}$ -t használjuk.

A (2.32) egyenlet második tagjának kiszámítása igényli a legnagyobb erőforrást, nagyságrendileg $2n_v^4 n_o^2$ darab lebegőpontos műveletet igényel a teljes tömb felépítése. Nevezük el ezt a

$$Q_{ab}^{ij} = \sum_{cd} \langle ab|cd \rangle C_{cd}^{ij} \quad (2.34)$$

tömbnek. Ahhoz, hogy ezt kiszámítsuk, a memóriában kell lennie az $\langle ab|cd \rangle$ integrállistának, ami a tervezett méretű rendszer esetén biztosan nem fér bele. Az I/O műveletek számát minimalizálni szeretnénk, ezért ezt az integrállistát az $\langle ij|kl \rangle$ integrállistához hasonlóan kell tárolnunk a merevlemezen az indexek megszorításával. Ezzel a teljes méret kb. 1/8-át kell tárolni és így az I/O műveletek (merevlemezzől olvasások) száma is az 1/8-odára csökken.

Mivel nem fér be a memóriába a teljes integrállista, ezért darabokban tudjuk csak beolvasni, viszont ekkor újra és újra végig kell olvasni az egész integrállistát, ahhoz, hogy az adott darabban feloldhassuk az indexek megkötéseit. Ez jelentősen lelassítaná a program futását, ezért ezt a megoldást elvetettük.

A fenti probléma elkerülésére a következő módszert alkalmaztuk. Vezessük be az

$$\langle ab||cd \rangle = \langle ab|cd \rangle - \langle ab|dc \rangle \quad (2.35)$$

$$\langle ab|||cd \rangle = \langle ab|cd \rangle + \langle ab|dc \rangle \quad (2.36)$$

integrállistákat. Az elsőt nevezzük el antiszimmetrikus listának, a másodikat szimmetrikus listának. Behelyettesítéssel egyszerűen belátható, hogy az

$$\begin{aligned} \langle ab||cd \rangle &= -\langle ab||dc \rangle = -\langle ba||cd \rangle = \langle ba||dc \rangle \\ &= \langle cd||ab \rangle = -\langle dc||ab \rangle = -\langle cd||ba \rangle = \langle dc||ba \rangle \end{aligned} \quad (2.37)$$

és az

$$\begin{aligned} \langle ab|||cd \rangle &= \langle ab|||dc \rangle = \langle ba|||cd \rangle = \langle ba|||dc \rangle \\ &= \langle cd|||ab \rangle = \langle dc|||ab \rangle = \langle cd|||ba \rangle = \langle dc|||ba \rangle \end{aligned} \quad (2.38)$$

indexfelcserélési szimmetriával rendelkeznek az újonnan bevezetett listák. Például

$$\langle cd||ba \rangle = \langle cd|ba \rangle - \langle cd|ab \rangle = \langle ba|cd \rangle - \langle ab|cd \rangle \quad (2.39)$$

$$= \langle ab|dc \rangle - \langle ab|cd \rangle = -\langle ab||cd \rangle \quad (2.40)$$

Ezen szimmetriák miatt ugyanolyan megkötéseket alkalmazhatunk az indexekre, mint az $\langle ab|cd \rangle$ integrállistánál.

Bevezetjük továbbá a

$$C_{cd}^{ij}(-) = \frac{1}{2}(C_{cd}^{ij} - C_{cd}^{ji}) \quad (2.41)$$

$$C_{cd}^{ij}(+) = \frac{1}{2(1 + \delta_{cd})}(C_{cd}^{ij} + C_{cd}^{ji}) \quad (2.42)$$

tömböket. Ha felhasználjuk, hogy (2.29) és (2.10) alapján

$$C_{cd}^{ij} = T_{cd}^{ij} + t_c^i t_d^j = T_{dc}^{ji} + t_d^j t_c^i = C_{dc}^{ji}, \quad (2.43)$$

a $C_{cd}^{ij}(-)$ és $C_{cd}^{ij}(+)$ mennyiségekről megmutatható, hogy a

$$C_{cd}^{ij}(-) = -C_{dc}^{ij}(-) = -C_{cd}^{ji}(-) = C_{dc}^{ji}(-) \quad (2.44)$$

és, hogy

$$C_{cd}^{ij}(+) = C_{dc}^{ij}(+) = C_{cd}^{ji}(+) = C_{dc}^{ji}(+) \quad (2.45)$$

indexfelcserélési szimmetriával rendelkeznek. Például

$$C_{dc}^{ij}(-) = \frac{1}{2}(C_{dc}^{ij} - C_{dc}^{ji}) = \frac{1}{2}(C_{cd}^{ji} - C_{cd}^{ij}) = -C_{cd}^{ij}(-). \quad (2.46)$$

Ezek miatt itt is bevezethetünk megkötéseket az indexekre. Legyen $i \leq j$ és $c \geq d$ mindkét tömbnél.

Vezessük be végül az

$$N_{ab}^{ij} = \sum_{c \geq d} \langle ab || cd \rangle C_{cd}^{ij}(-) \quad (2.47)$$

és az

$$M_{ab}^{ij} = \sum_{c \geq d} \langle ab || | cd \rangle C_{cd}^{ij}(+) \quad (2.48)$$

tömböket. Ezek a tömbök öröklík a szimmetriákat és a megkötéseket az indexeikre $C(+)$ -tól és $C(-)$ -tól, azaz $i \leq j$, ill. az $\langle ab || | cd \rangle$ és az $\langle ab || cd \rangle$ integrállistáktól, azaz $a \geq b$ mindkét tömbnél.

Megmutatjuk, hogy

$$Q_{ab}^{ij} = N_{ab}^{ij} + M_{ab}^{ij}. \quad (2.49)$$

Fejtsük ki az egyenlet jobb oldalán szereplő tagokat! Mivel az $\langle ab || cd \rangle$ integrállista az utolsó két indexében antiszimmetrikus, az N_{ab}^{ij} kifejtésében elég a szummában a $c > d$ eseteket nézni ($c = d$ esetén a megfelelő tagok eltűnnek).

$$\begin{aligned} N_{ab}^{ij} &= \frac{1}{2} \sum_{c \geq d} (\langle ab || cd \rangle - \langle ab || dc \rangle) (C_{cd}^{ij} - C_{cd}^{ji}) \\ &= \frac{1}{2} \sum_{c > d} \langle ab || cd \rangle (C_{cd}^{ij} - C_{cd}^{ji}) - \frac{1}{2} \sum_{c > d} \langle ab || dc \rangle (C_{cd}^{ij} - C_{cd}^{ji}) \\ &= \frac{1}{2} \sum_{c > d} \langle ab || cd \rangle (C_{cd}^{ij} - C_{cd}^{ji}) - \frac{1}{2} \sum_{d > c} \langle ab || cd \rangle (C_{dc}^{ij} - C_{dc}^{ji}) \\ &= \frac{1}{2} \sum_{c > d} \langle ab || cd \rangle (C_{cd}^{ij} - C_{cd}^{ji}) - \frac{1}{2} \sum_{d > c} \langle ab || cd \rangle (C_{cd}^{ji} - C_{cd}^{ij}) \\ &= \frac{1}{2} \sum_{c \neq d} \langle ab || cd \rangle (C_{cd}^{ij} - C_{cd}^{ji}) \end{aligned} \quad (2.50)$$

$$\begin{aligned}
M_{ab}^{ij} &= \frac{1}{2(1 + \delta_{cd})} \sum_{c \geq d} (\langle ab|cd \rangle + \langle ab|dc \rangle) (C_{cd}^{ij} + C_{cd}^{ji}) \\
&= \frac{1}{2} \sum_{c > d} (\langle ab|cd \rangle + \langle ab|dc \rangle) (C_{cd}^{ij} + C_{cd}^{ji}) + \frac{1}{4} \sum_c 2 \langle ab|cc \rangle 2 C_{cc}^{ij} \\
&= \frac{1}{2} \sum_{c > d} \langle ab|cd \rangle (C_{cd}^{ij} + C_{cd}^{ji}) + \frac{1}{2} \sum_{c > d} \langle ab|dc \rangle (C_{cd}^{ij} + C_{cd}^{ji}) + \sum_c \langle ab|cc \rangle C_{cc}^{ij} \\
&= \frac{1}{2} \sum_{c > d} \langle ab|cd \rangle (C_{cd}^{ij} + C_{cd}^{ji}) + \frac{1}{2} \sum_{d > c} \langle ab|cd \rangle (C_{dc}^{ij} + C_{dc}^{ji}) + \sum_c \langle ab|cc \rangle C_{cc}^{ij} \\
&= \frac{1}{2} \sum_{c \neq d} \langle ab|cd \rangle (C_{cd}^{ij} + C_{cd}^{ji}) + \sum_c \langle ab|cc \rangle C_{cc}^{ij} \tag{2.51}
\end{aligned}$$

Ezt a két tagot összeadva

$$\begin{aligned}
N_{ab}^{ij} + M_{ab}^{ij} &= \frac{1}{2} \sum_{c \neq d} [\langle ab|cd \rangle (C_{cd}^{ij} - C_{cd}^{ji}) + \langle ab|cd \rangle (C_{cd}^{ij} + C_{cd}^{ji})] + \sum_c \langle ab|cc \rangle C_{cc}^{ij} \\
&= \sum_{cd} \langle ab|cd \rangle C_{cd}^{ij} = Q_{ab}^{ij} \tag{2.52}
\end{aligned}$$

visszakapjuk a számunkra szükséges Q_{ab}^{ij} tömböt.

Ezzel a módszerrel két kb. $n_v^4/8$ méretű tömböt kell a merevlemezen tárolni és onnan olvasni, viszont csak kétszer kb. $2n_v^4 n_o^2/4$ darab lebegőpontos műveletet kell elvégezni a $2n_v^4 n_o^2$ helyett és csak egyszer kell végigolvasni mindkét új integrállistát, ami jóval előnyösebb.

2.3. Az algoritmus

2.3.1. Az iteráció előtti műveletek

Az iteráció előtt felépítjük az (2.3) $\mathbf{L}^{ij}$ mátrixot és a (2.7) $\mathbf{I}^{ijk}$ vektort, majd ezeket kiírjuk a merevlemezre úgy, hogy ezeket is az integrállistákhoz hasonlóan lehessen olvasni. Ez után felépítjük a Fock-mátrixot a memóriába (2.4) alapján. A csak virtuális indexeket tartalmazó részét az **f**ab tömbbe, az egy virtuális és egy betöltött indexű részét az **f**ai tömbbe, a csak betöltött indexű részt pedig az **f**ij tömbbe helyezzük. A magtasztási energiához hozzáadva az egyelektron integrál mátrix és a Fock-mátrix betöltött indexű részének nyomát megkapjuk a Hartree–Fock energiát.

A klaszteramplitúdók inicializálásához a **V** tömbbe beolvassuk az $\langle ab|ij \rangle$

integrállistát. Az amplitúdók a

$$t_a^i = \frac{f_a^i}{f_{ii} - f_{aa}}, \quad (2.53)$$

$$T_{ab}^{ij} = \frac{V_{ab}^{ij}}{f_{ii} + f_{jj} - f_{aa} - f_{bb}} \quad (2.54)$$

értékeket kapják. Ez megegyezik azzal, mintha nullákkal töltenénk fel az amplitúdókat és egy iterációs lépést elvégeznénk. Az így kapott amplitúdókat kiírjuk a merevlemezre, mert később még szükségünk lesz rá. Ezekkel az amplitúdókkal kiszámoljuk az energiát, ami a másodrendű Møller-Plesset-féle perturbációs energiával egyezik meg.

2.3.2. Műveletek egy iterációs lépésen belül

Egy iterációs lépésen belül ki kell számolnunk a reziduális vektort és mátrixot, majd a (2.28) képlettel kiszámoljuk az új amplitúdókat és ezekkel a korrelációs energiát. Ahhoz, hogy megkapjuk a reziduális vektort és mátrixot ki kell számolnunk a fent megadott intermediereket. Ehhez mátrix-mátrix szorzásokat, mátrix-vektor szorzásokat és vektorok diadikus szorzatát kell kiszámolni, amiket az előre megírt BLAS rutinok használatával végezhetünk hatékonyan. Az alábbiakban részletezve felsoroljuk az egyes műveleteket. Az egyenletekben szereplő tagok járulékát a következőkben $\rightarrow$ jelöli. Lásd például: (2.55).

1. (a) Bemásoljuk az **fai** tömböt az **rr** tömbbe a (2.19) alapján és a **v** tömbbe a (2.20) alapján. Így elkerüljük azt, hogy külön tároljuk az **ss** tömböt.
- (b) Bemásoljuk az **fii** tömböt a **beta** tömbbe a (2.18) alapján.
- (c) Kiszámoljuk a (2.18) egyenlet negyedik tagját. Ez adott k -ra és i -re az $\mathbf{f}^k$ és $\mathbf{t}^i$ vektorok skaláris szorzata, amit a **ddot** BLAS-rutinnal számolhatunk ki. Észrevehetjük azonban, hogy ha felírjuk ezt a szorzást a

$$\sum_a \mathbf{fai}(\mathbf{a}, \mathbf{k}) \mathbf{t}(\mathbf{a}, \mathbf{i}) \rightarrow \mathbf{beta}(\mathbf{k}, \mathbf{i}) \quad (2.55)$$

alakban, akkor ez egy mátrixszorzás

$$\tilde{\mathbf{F}}^\dagger \tilde{\mathbf{t}} \rightarrow \tilde{\beta} \quad (2.56)$$

szerint, ahol most $\tilde{\mathbf{F}}$ jelöli az $\mathbf{fai}(\mathbf{a}, \mathbf{k})$ $n_v \times n_o$ -s mátrixot, a $\tilde{\mathbf{t}}$ a $\mathbf{t}(\mathbf{a}, \mathbf{k})$ $n_v \times n_o$ -s mátrixot és $\tilde{\beta}$ a $n_o \times n_o$ -s mátrixot. Ezt a műveletet a **dgemm** BLAS-rutinnal végezhetjük el a leghatékonyabban.

2. (a) A **kk** tömböt beolvassuk **vv**-be a (2.27) egyenlet alapján. A merevlemezen a **kk** tömböt, azaz a $\langle ai|bj \rangle$ integrállistát az $(i \geq j)$ és az $(a \geq b)$ megkötésekkel tároljuk. A beolvasáskor a memóriában feloldjuk az utóbbi megkötést.
- (b) A (2.17) egyenlet alapján beolvassuk az $\langle ij|kl \rangle$ integrállistát a memóriába (a tömböknek nem lefoglalt részébe) úgy, hogy az indexekre vonatkozó összes megkötést feloldjuk. Ezt a memóriaterületet innen-től kezdve **alpha**-nak hívjuk. Később, amikor már nem lesz rá szükségünk, ezt a memóriaterületet felülírjuk.
- (c) Kiszámoljuk az **alpha** tömb második tagját ugyan ezen egyenlet alapján. A $\mathbf{C}^{ij}$ mátrixot felépítjük a (2.10) egyenlet alapján az **scrs** tömbbe. A $\mathbf{K}^{lk}$ mátrix már a **vv** tömbben van a memóriában, minden lk indexre megkapható ($l > k$ esetén transzponálással). Ezt a két mátrixot kell összeszorozni és az eredmény nyomát venni, hogy megkapjuk az **alpha** tömb ij, kl elemét. Ezt írjuk fel a

$$\begin{aligned} \text{tr}(\mathbf{C}^{ij} \mathbf{K}^{lk}) &= \sum_{ab} C_{ab}^{ij} K_{ba}^{lk} = \sum_{ab} C_{ab}^{ij} (K^{lk})_{ab}^{\dagger} \\ &= \sum_{ab} C_{ab}^{ij} K_{ab}^{kl} \rightarrow \text{alpha}(i, j, k, l) \end{aligned} \quad (2.57)$$

alakban, azaz a $\mathbf{C}^{ij}$ és a $\mathbf{K}^{kl}$ mátrix oszlopait egymás után írva egy-egy vektorba egy skalárszorozást kapunk. Ezt a **ddot** BLAS-rutinnal számolhatjuk ki hatékonyan. Mivel a $\mathbf{K}^{lk}$ mátrixok közül az $l \leq k$ mátrixok érhetőek el közvetlenül, a $\mathbf{C}^{ij}$ mátrixokat pedig $i \leq j$ megkötéssel tudjuk kiszámolni közvetlenül, az i, j, k, l -re menő ciklusokat fel kell bontanunk

- $i > j$ és $l > k$
- $i \leq j$ és $l > k$
- $i > j$ és $l \leq k$
- $i \leq j$ és $l \leq k$

esetekre és az ezeknek megfelelő mátrixokat kell használnunk.

- (d) Beolvassuk a **ks** tömböt a fennmaradó memóriába.
- (e) Kiszámoljuk a (2.17) egyenlet a 3. és 4. tagját az **alpha** tömbbe. Ez a két tag egy-egy skalárszorozás, ami csak a vektorok betöltött indexeiben különbözik, ezért elegendő egyszer elvégezni és az eredményt a megfelelő helyre beírni

$$\mathbf{t}^{i\dagger} \mathbf{k}^{klj} \rightarrow \text{alpha}(i, j, k, l), \quad (2.58)$$

$$\mathbf{t}^{j\dagger} \mathbf{k}^{lki} \rightarrow \text{alpha}(i, j, k, l) \equiv \mathbf{t}^{i\dagger} \mathbf{k}^{klj} \rightarrow \text{alpha}(j, i, l, k) \quad (2.59)$$

módon.

- (f) Kiszámoljuk a (2.25) egyenlet első szummájának második tagját. Ezt közvetlenül a $\mathbf{V}^{ij}$ mátrixba számoljuk ki, azaz kellenek a

$$-\sum_k \mathbf{k}^{ijk} \mathbf{t}^{k\dagger} \rightarrow \mathbf{V}^{ij} \quad (2.60)$$

diadikus szorzatok és a

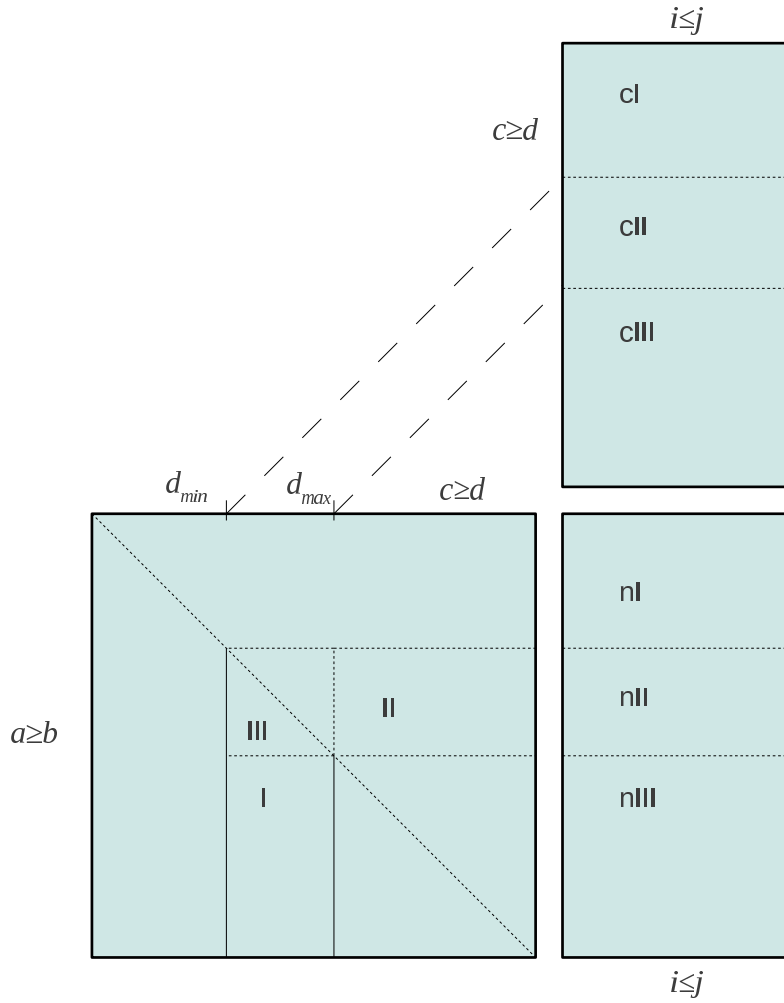
$$-\sum_k \mathbf{t}^k \mathbf{k}^{jik\dagger} \rightarrow \mathbf{V}^{ij} \quad (2.61)$$

diadikus szorzatok, amik az előzőek transzponáltjai $i \leftrightarrow j$ cserével.

- (g) A fenti pontokban felépített `alpha` tömböt felhasználva kiszámoljuk a (2.27) egyenletben a harmadik tagot a $\mathbf{V}^{ij}$ mátrixba. Itt is külön kell kezelnünk az $l > k$ és $l \leq k$ eseteket. Minden esetben felépítjük a megfelelő $\mathbf{C}^{kl}$ mátrixot és ennek az `alpha(i, j, k, l)`-szeresét hozzáadjuk a $\mathbf{V}^{ij}$ mátrixhoz.
3. (a) Kiírjuk a merevlemezre a `vv` tömböt.
- (b) Felépítjük a $\mathbf{C}(-)$ tömböt a (2.41) egyenlet alapján a `vv` helyére. Az $\mathbf{N}$ tömböt a memória végére építjük a következő lépésben.
- (c) Kiszámoljuk hány háromindexes blokk fér be a fennmaradó memóriába a (3d) pont figyelembevételével. Beolvasunk az $\langle ab||cd \rangle$ integrálokból ennyi háromindexes blokkot. Nevezzük d_{min} -nek az ebben az lépésben elsőnek beolvasott háromindexes blokk indexét. Nevezzük d_{max} -nak az utolsó ilyen.
- (d) A beolvasott integrállistát úgy tároljuk a memóriában, mintha az $ab \geq cd$ megkötést feloldottuk volna és az így kapott ab és cd hiperindexekkel indexelt kétindexes tömb egy része lenne. Ennek a részletnek a sorindexe $b = d_{min}$ -től $b = n_v$ -ig terjed, az oszlopindexe $d = d_{min}$ -től $d = d_{max}$ -ig.

Ezzel a módszerrel megkapjuk az $\langle ab||cd \rangle$ integrállista azon részét, ahol a sorindexekre a $d_{max} < b \leq n_v$, az oszlopindexekre a $d_{min} \leq d \leq d_{max}$ megszorítás érvényes. Nevezzük ezt I résznek. Az I rész transzponáltja megegyezik a $\langle ab||cd \rangle$ integrállista azon részével, ahol a sorindexekre a $d_{min} \leq d \leq n_v$, az oszlopindexekre a $d > d_{max}$ megszorítás érvényes. Nevezzük ezt II résznek. Azt a részt, ahol a sorindexekre a $b \leq d_{max}$, az oszlopindexekre a $d_{min} \leq d \leq d_{max}$ megszorítás érvényes, nevezzük III résznek. Ez egy szimmetrikus mátrix alsó háromszöge. A memóriában csak az I és a III részt tároljuk.

A $\mathbf{C}(-)$ tömböt szintén felfoghatjuk egy cd és ij hiperindexű kétindexes tömbnek. Azt a részét, ahol a sorindex $1 \leq d \leq d_{min}$ nevezzük cI-nek, azt, ahol $d_{min} < d < d_{max}$ cII-nek, azt, ahol $d_{max} < d < n_v$ cIII-nek. ($\mathbf{N}$ tömbnél ugyanígy: nI, nII, nIII.) Ezt a (2.1) ábrán szemléltetjük.



2.1. ábra. A mátrixszorzás elvégzése darabokban

- (e) Az $\mathbf{N}$ felépítése a következőképp zajlik. Az I tömböt meg kell szoroznunk cII tömbbel, ez nIII-ba kerül. A II tömb I transzponáltja, ezt cIII-mal szorozzuk meg, ez nII-be kerül. Ennél a két lépésnél a `dgemm` rutint használjuk. A III tömböt meg kell szoroznunk a cIII tömbbel,

az eredmény az nII részbe kerül. Ennél a két lépésnél a `dsymm` rutint használjuk. Ennek az az előnye, hogy csak az alsó háromszöget használja és emiatt nem kell a felső háromszögbe való transzponálást elvégezni. Abban az esetben, amikor $d_{max} = n_v$ csak ezt az utolsó műveletet kell alkalmaznunk. Így fel tudjuk építeni a teljes $\mathbf{N}$ tömböt, végighaladva a beolvasással az $\langle ab|cd \rangle$ integrállista darabjain.

- (f) Ezek után visszaolvassuk a helyükre a vv , t és tt tömböket. Feloldva az $a \geq b$ megkötést hozzáadjuk az $\mathbf{N}$ tömböt vv -hez, majd ezt ismét kiírjuk a merevlemezre.
 - (g) Felépítjük a $\mathbf{C}(+)$ tömböt a (2.42) egyenlet alapján a vv helyére. Az $\mathbf{M}$ tömböt a memória végébe fogjuk építeni.
 - (h) Ugyan úgy járunk el, mint a fentiekben $\mathbf{N}$ felépítésénél, $\mathbf{C}(-)$ -t $\mathbf{C}(+)$ -ra és $\langle ab|cd \rangle$ -t $\langle ab||cd \rangle$ -re cserélve. Ismét visszaolvassuk a helyükre a vv , t és tt tömböket. Feloldva az $a \geq b$ megkötést hozzáadjuk az $\mathbf{M}$ tömböt vv -hez.
4. (a) Felépítjük a teljes $\mathbf{C}$ tömböt megkötések nélkül a memória végére. Kihagyunk a kdo tömbnek megfelelő helyet a fennmaradó memória elején és a maradék helyre az $\langle ab|ci \rangle$ integrállistából annyi három virtuális indexes blokkot olvasunk be, amennyi belefér.
- (b) Kiszámoljuk a (2.32) egyenlet első és második tagját azokra az indexekre, amikre beolvastuk a memóriába az integrállistát, közvetlenül a vv tömbbe. A második tag egy

$$\sum_c \langle ab|cj \rangle t_c^i \rightarrow \mathbf{kdv}(\mathbf{a}, \mathbf{b}, \mathbf{i}, \mathbf{j}) \quad (2.62)$$

mátrix-vektor szorzás, a háromindexes blokkot felfoghatjuk egy ab hiperindexű és egy c normál indexű mátrixnak rögzített j -re, a t_c^i tömböt pedig egy c indexű vektornak rögzített i -re. Erre a `dgemv` rutint használjuk. Az első tagot írjuk át a

$$\sum_c \langle ab|ic \rangle t_c^j = \sum_c \langle ba|ci \rangle t_c^j = \sum_c \langle ca|bi \rangle t_c^j \rightarrow \mathbf{kdv}(\mathbf{a}, \mathbf{b}, \mathbf{i}, \mathbf{j}) \quad (2.63)$$

módon. Ez szintén egy mátrix-vektor szorzás, de itt a háromindexes blokkot úgy fogjuk fel, mint egy kétindexes mennyiség, amelynek az első (normál) indexe c , a második (hiper)indexe bc . Ezt transzponálni kell és a `dgemv` rutinnal el kell végezni a szorzást. A `dgemv` rutinnak megadhatjuk, hogy a mátrix transzponáltja van a kijelölt memóriahelyen, ezért ezzel nekünk nem kell foglalkoznunk. A $\mathbf{kdv}$ tömb helyett közvetlenül a vv tömbbe számoljuk ki ezeket a tagokat.

5. (a) Kiszámoljuk a (2.33) egyenlet harmadik tagját a memóriában lévő integrállista részlettel a **kd**-nek lefoglalt helyre. Ezt a tagot

$$\sum_{bc} \langle ak|bc \rangle C_{bc}^{ij} = \sum_{bc} \langle bc|ak \rangle C_{bc}^{ij} \rightarrow \mathbf{kdo}(\mathbf{a}, \mathbf{k}, \mathbf{i}, \mathbf{j}) \quad (2.64)$$

átrendezve felfoghatjuk mátrixszorzásnak. Az első mátrix egy ak sor-hiperindexű, bc oszlop-hiperindexű mátrix transzponáltja, a második egy bc sor-hiperindexű, ij oszlop-hiperindexű mátrix. Ezt a műveletet a **dgemm** rutinnal végezhetjük el. Ez után újabb részletet olvasunk be a memóriába az $\langle ab|ci \rangle$ integrállistából és végighaladunk a (4b)-(5a) lépéseken, amíg az egész integrállistát végig nem olvastuk.

- (b) Kiszámoljuk a (2.33) egyenlet első tagját. Az előző lépésekhez hasonlóan járunk el. Beolvasunk a memóriába annyi $n_v^2 n_o$ méretű blokkot a **kk** tömbből, ahány befér a memóriába. Minden lehetséges i, j, k indexre ez egy

$$\sum_b \langle ak|ib \rangle t_b^j = \sum_b \langle ab|ik \rangle t_b^j \rightarrow \mathbf{kdo}(\mathbf{a}, \mathbf{k}, \mathbf{i}, \mathbf{j}) \quad (2.65)$$

mátrix-vektor szorzás, amit a **dgemv** rutinnal elvégzünk. Ezután ismét beolvasunk annyi blokkot, amennyi befér a memóriába és ismételjük ezt a lépést, amíg az egész **kk** tömböt végig nem olvastuk.

- (c) Kiszámoljuk a (2.33) egyenlet második tagját. Most a **jj** tömböt olvassuk a merevlemezről a memóriába blokkonként. A műveletet

$$\sum_b \langle ak|bj \rangle t_b^i = \sum_b \langle ab|kj \rangle t_b^i \rightarrow \mathbf{kdo}(\mathbf{a}, \mathbf{k}, \mathbf{i}, \mathbf{j}) \quad (2.66)$$

szerint átírva adott j, k indexre egy a sor- és b oszlopindexű mátrixot szorzunk össze egy b sor- és i oszlopindexű mátrixszal. A **jj** tömböt az $a \geq b$ és $k \geq j$ megkötésekkel tároljuk, az elsőt a merevlemezről való olvasáskor feloldjuk, a másodikat figyelembe vesszük úgy, hogy ha $k < j$, akkor egy $k \leftrightarrow j$ cserét hajtunk végre. Ezzel felépítettük a teljes **kd** tömböt.

- (d) Kiszámoljuk a (2.20) egyenlet negyedik tagját a **v** tömbbe. Minden i, k indexre hozzáadjuk a

$$[2\mathbf{k}\mathbf{d}^{kik} - \mathbf{k}\mathbf{d}^{kki}] = 2\mathbf{kdo}(\cdot, \mathbf{k}, \mathbf{i}, \mathbf{k}) - \mathbf{kdo}(\cdot, \mathbf{k}, \mathbf{k}, \mathbf{i}) \quad (2.67)$$

vektorokat a $\mathbf{v}(\cdot, \mathbf{i})$ vektorhoz a **daxpy** rutinnal.

- (e) Kiszámoljuk a (2.25) egyenlet első szummájának első tagját közvetlenül a **vv** tömbbe. A (2.67) egyenlethez hasonlóan a **kdo**($\cdot$, **k**, **i**, **j**) vektorral elvégezzük a

$$\mathbf{kdo}(\cdot, \mathbf{k}, \mathbf{i}, \mathbf{j}) \mathbf{t}^{\mathbf{k}\dagger} \quad (2.68)$$

diadikus szorzatokat a **dger** rutinnal.

6. (a) Beolvassuk a merevlemezről a teljes **ls** tömböt, a **kdo** tömböt felülírjuk.
 (b) Kiszámoljuk a (2.20) egyenlet harmadik tagját közvetlenül a **v** tömbbe. Ez, ahogy az egyenletben szerepel, egy mátrix-vektor szorzás minden i, k, l indexre.
 (c) Kiszámoljuk a (2.18) egyenlet harmadik tagját. Ezt

$$\sum_l \mathbf{t}^{l\dagger} \mathbf{l}^{ki} = \sum_{al} \mathbf{t}(\mathbf{a}, l) \mathbf{ls}(\mathbf{a}, l, \mathbf{k}, \mathbf{i}) \rightarrow \mathbf{beta}(\mathbf{k}, \mathbf{i}) \quad (2.69)$$

módon kifejtve látszik, hogy egy mátrix-vektor szorzás. Az **ls** tömböt fogjuk fel egy al sor- és egy ki oszlop-hiperindexű mátrixnak, ennek a transzponáltját kell megszorozni a **t** al hiperindexű vektorral. Az eredmény a **beta** ki hiperindexű vektor. Ezt a **dgemv** rutinnal végezhetjük el.

7. (a) Feltöltjük nullákkal az **x** tömböt.
 (b) Beolvasunk a merevlemezről annyi háromindexes blokkot az **l1** tömbből, amennyi befér a memóriába.
 (c) A (2.19) egyenlet második tagját kiszámoljuk. Ez egy mátrix-vektor szorzás minden k, l indexre, amit a **dgemv** rutinnal végzünk el.
 (d) Kiszámoljuk a (2.18) egyenlet második tagját. Ezt kifejtve a

$$\sum_l \text{tr}(\mathbf{L}^{kl} \mathbf{C}^{li}) = \sum_{abl} L_{ab}^{kl} C_{ba}^{li} = \sum_{abl} L_{ab}^{kl} C_{ab}^{il} \rightarrow \mathbf{beta}(\mathbf{k}, \mathbf{i}) \quad (2.70)$$

módon látjuk, hogy ez egy skalárszorítás minden i, k, l indexre. Ezt a **ddot** rutinnal végezhetjük el.

- (e) A (2.21) egyenlet alapján kiszámoljuk az **aa** mátrixot és hozzáadjuk **x**-hez. Ez minden k, l indexre egy mátrixszorzás. Újból beolvasunk egy részletet a **l1** tömbből és elvégezzük a (7b)-(7e) lépéseket, amíg végig nem olvassuk az egész **l1** tömböt.
 8. (a) Bemásoljuk a **scrs** tömbbe az **x** tömb transzponáltját. Megszorozzuk **scrs** tömb minden elemét -1-gyel. Hozzáadjuk a **scrs** tömbhöz a **h** tömb két virtuális indexű részét [a (2.20) egyenlet második tagja].

- (b) A `dgemm` rutin hívásával a `scrs` tömböt megszorozzuk a `t` tömbbel, amit most egy $n_v \times n_o$ -s mátrixként fogunk fel, az eredményt a `v` tömbbe írjuk. Ezzel felépítettük a teljes $\mathbf{s}^i$ vektort, amit jelenleg a `v` helyén tárolunk.
 - (c) Kiszámoljuk a (2.25) egyenlet első tagját közvetlenül a `vv` tömbbe. Ez egy diadikus szorzás minden i, j indexre, amit a `dger` rutinnal végzünk el.
9. (a) Kiszámoljuk a (2.26) egyenlet második tagját. Ez két mátrix-vektor szorzás minden i, k indexre $\mathbf{T}^{ik}$, ill. $\mathbf{T}^{ki}$ mátrixszal és $\mathbf{r}^k$ vektorral.
- (b) Kiszámoljuk a (2.26) egyenlet utolsó tagját. Minden i, k indexre hozzá kell adnunk a $\mathbf{t}^k$ vektor $-\text{beta}(\mathbf{k}, \mathbf{i})$ -szeresét $\mathbf{v}(\cdot, \mathbf{i})$ vektorhoz. Ezt átírhatjuk úgy is, hogy a `t`-t mint $n_v \times n_o$ -s mátrixot szorozzuk meg a $\text{beta } n_o \times n_o$ -s mátrixszal. Ezt a `dgemm` rutinnal végezzük el.
10. Kiszámoljuk a (2.25) egyenlet harmadik tagját közvetlenül a `vv` tömbbe. Minden i, j, k -ra a $\mathbf{C}^{kj}$ mátrix $-\text{beta}(\mathbf{k}, \mathbf{i})$ -szorosát adjuk hozzá a $\mathbf{vv}(\cdot, \cdot, \mathbf{i}, \mathbf{j})$ mátrixhoz.
11. A (2.22) egyenlet alapján megszorozzuk az `x` mátrix elemeit -1 -gyel és hozzáadjuk az `fab` mátrixot az eredményt az `x` mátrixba írjuk.
12. Ugyanezen egyenlet utolsó tagját is kiszámoljuk és az `x` mátrixhoz adjuk. Ez egy diadikus szorzás minden k -ra, de átírhatjuk mátrixszorzásra úgy, hogy az `r` $n_v \times n_o$ -s mátrixot szorozzuk meg az `t` $n_v \times n_o$ -s mátrix transzponáltjával.
13. (a) Felépítjük a (2.8) egyenlet alapján a `je` tömböt. Ezt az előzőekhez hasonlóan úgy tesszük, hogy beolvassuk annyi n_v^3 méretű blokkot a memóriába az $\langle ab|ci \rangle$ integrállistából, amennyi befér, ezekkel kiszámoljuk a `je` tömb megfelelő részét. A memóriában (11|22) formátumban tároljuk ezeket a blokkokat. Egy ab sor-hiperindexű és c oszlopindexű mátrixot kell megszoroznunk a $\mathbf{t}^j$ vektorral. Ezt a `dgemv` rutinnal végezzük el. Ezután újra beolvassuk egy részletet az integrállistából és ezzel újra kiszámoljuk a megfelelő részét a `je` tömbnek.
- (b) A `je` tömböt a szabad memória elején tároljuk (nevezzük ezt most `z`-nek), majd amikor készen vagyunk kiírjuk a merevlemezre.
- (c) A memóriában lévő `je` tömbbel kiszámoljuk a (2.22) egyenlet harmadik tagját `x` mátrixba, azaz hozzáadjuk minden k -ra a $\mathbf{J}(E^{kk})$ mátrixot `x`-hez.
- (d) A (2.24) egyenlet alapján beolvassuk blokkonként a `jj` tömböt és hozzáadjuk a `z` tömbhöz.

- (e) Ugyanezen egyenletnek kiszámoljuk a harmadik és negyedik tagját. A harmadik tagban a $\mathbf{k}\mathbf{k}$ tömböt blokkonként olvassuk, majd minden j, k, l indexre elvégezzük a $\mathbf{K}^{lk}\bar{\mathbf{C}}^{jl}$ mátrixszorzást és levonjuk a $\mathbf{Z}^{kj}$ mátrixból. A negyedik taghoz beolvassuk blokkonként a $\mathbf{k}\mathbf{s}$ tömböt, majd minden k -ra megszorozzuk a $\mathbf{k}\mathbf{s}(\cdot, \cdot, \mathbf{k}, \mathbf{j})$ $n_v \times n_o$ -s mátrixot a $\mathbf{t}$ $n_v \times n_o$ -s mátrix transzponáltjával.
14. A (13) pontban elkészült $\mathbf{z}$ tömb segítségével kiszámoljuk a (2.25) egyenlet hatodik és hetedik tagját. Minden i, j, k indexre a $\mathbf{s}\mathbf{c}\mathbf{r}\mathbf{s}$ mátrixba kiszámoljuk a $\mathbf{T}^{ki}\mathbf{Z}^{kj}$ mátrixot. Ennek a transzponáltját a $\mathbf{w}$ tömbbe írjuk. Hozzáadjuk $\mathbf{v}\mathbf{v}(\cdot, \cdot, \mathbf{i}, \mathbf{j})$ mátrixhoz a $\mathbf{s}\mathbf{c}\mathbf{r}\mathbf{s}$ -1 -szeresét és a $\mathbf{w}$ $-1/2$ -szeresét.
15. (a) A (13) ponthoz hasonlóan felépítjük a $\mathbf{k}\mathbf{e}$ tömböt a (2.9) egyenlet alapján, ezt a memóriaterületet hívjuk most $\mathbf{y}$ -nak.
 (b) Kiszámoljuk a (2.22) egyenlet negyedik tagját a (13c) ponthoz hasonlóan.
 (c) Blokkonként olvasva hozzáadjuk a $\mathbf{j}\mathbf{j}$, $\mathbf{k}\mathbf{k}$, $\mathbf{j}\mathbf{e}$ tömböket az $\mathbf{y}$ tömbhöz a (2.23) egyenlet alapján.
 (d) Kiszámoljuk a (2.23) egyenlet utolsó tagjait. Ehhez 11 tömböt blokkonként olvassuk és minden j, k, l indexre kiszámoljuk a

$$2\bar{\mathbf{C}}^{lj} - \bar{\mathbf{C}}^{jl} \rightarrow \mathbf{s}\mathbf{c}\mathbf{r}\mathbf{s} \quad (2.71)$$

mátrixot, majd ezzel a

$$\frac{1}{2}\mathbf{L}^{kl}\mathbf{s}\mathbf{c}\mathbf{r}\mathbf{s} \quad (2.72)$$

mátrixszorzást, és az eredményt az $\mathbf{y}$ tömb megfelelő részéhez hozzáadjuk. Ezután blokkonként olvassuk az $\mathbf{l}\mathbf{s}$ tömböt és elvégezzük az

$$\frac{1}{2} \sum_l \mathbf{l}^{lkj} \mathbf{t}^{l\dagger} \rightarrow \mathbf{Y}^{kj} \quad (2.73)$$

diadikus szorzást. A

$$\mathbf{f}^k \mathbf{t}^{j\dagger} \rightarrow \mathbf{Y}^{kj} \quad (2.74)$$

diadikus szorzást elvégezzük minden j, k indexre.

16. A (2.25) egyenlet második szummájának első tagját kiszámoljuk $\mathbf{v}\mathbf{v}$ -be. Ehhez minden i, j, k indexre a $\mathbf{w}$ -be másoljuk a $\mathbf{T}^{ik}$ mátrixot, megszorozzuk minden elemét 2-vel. Az $\mathbf{s}\mathbf{c}\mathbf{r}\mathbf{s}$ tömbbe másoljuk a $\mathbf{T}^{ik}$ mátrix transzponáltját [ld. (2.29)], és ezt levonjuk a $\mathbf{w}$ tömbből. Elvégezzük a

$$\mathbf{w}\mathbf{Y}^{kj} \rightarrow \mathbf{v}\mathbf{v}(\cdot, \cdot, \mathbf{i}, \mathbf{j}) \quad (2.75)$$

mátrixszorzást a `dgemm` rutinnal.

17. Minden i, j indexre kiszámoljuk a (2.25) egyenlet negyedik tagját. Ez egy mátrixszorzás a `dgemm` rutinnal.
18. Kiszámoljuk az új amplitúdókat a

$$t_a^i = t_a^i - \frac{v_a^i}{f_{aa} - f_{ii}}, \quad (2.76)$$

$$T_{ab}^{ij} = T_{ab}^{ij} - \frac{V_{ab}^{ij}}{f_{aa} + f_{bb} - f_{ii} - f_{jj}} \quad (2.77)$$

képletekkel.

19. Lefuttatjuk a DIIS konvergenciagyorsító rutint [53].
20. Kiszámoljuk az energiát a (2.28) képlettel.

2.4. Módosítások a lokális számításához

A fent ismertetett algoritmus kis módosításokkal lokális CCSD számolásokra is alkalmazható. Lokális számítás esetén megoldjuk a CCSD egyenleteket a doménhez tartozó transzformált MO bázisban, majd a transzformáció mátrixának az adott pályához tartozó (sor)vektorával ($\mathbf{U}_i$) vissza kell transzformálnunk az energiát a

$$\delta E_i = 2 \sum_{akl} U_{ik} f_a^k U_{il} t_a^l + \sum_{abjkl} U_{ik} L_{ab}^{kj} U_{il} C_{ab}^{lj} \quad (2.78)$$

képlettel. Ehhez bevezetjük az

$$\text{INT1}(\mathbf{k}, 1) = 2 \sum_a f_a^k t_a^l + \sum_{abj} L_{ab}^{kj} C_{ab}^{lj} \quad (2.79)$$

intermediert. Így a (2.78) egyenlet a

$$\delta E_i = \sum_{kl} U_{ik} [\text{INT1}(\mathbf{k}, 1) U_{il}] \quad (2.80)$$

szerint módosul. Az első tag egy mátrixszorzás, az $\mathbf{fai}$ $n_v \times n_o$ -s mátrixot szorozzuk a $\mathbf{t}$ $n_v \times n_o$ -s mátrix transzponáltjával. A második tagál minden lj indexre felépítjük a $\mathbf{C}^{lj}$ mátrixot. Ezt felfoghatjuk úgy, mint egy n_v^2 hosszú vektort. Ezt skalárisan szorozzuk az $\mathbf{L}^{kj}$ -vel minden k indexre. Amikor készen van az INT1 mátrix, elvégezzük a (2.80) egyenletben szereplő mátrix-vektor szorzást majd, a skaláris szorzatot.

3. fejezet

A perturbatív (T) korrekció

3.1. Az egyenletek

A 54. referencia egyenleteiből indultunk ki. Úgy, mint a CCSD programhoz használt közleményben, elvileg itt is programozható formában adták meg az egyenleteket, amit hatékonyan tudunk programozni.

A (T) korrekciót zárt héjú rendszerre a

$$E_{(T)} = \frac{1}{3} \sum_{ijk} \sum_{abc} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \quad (3.1)$$

képlet adja meg. Itt bevezettük a

$$W_{abc}^{ijk} = P_{abc}^{ijk} \left(\sum_d (bd|ai)T_{cd}^{kj} - \sum_l (ck|jl)T_{ab}^{il} \right) \quad (3.2)$$

és

$$V_{abc}^{ijk} = W_{abc}^{ijk} + (bj|ck)t_a^i + (ai|ck)t_b^j + (ai|bj)t_c^k \quad (3.3)$$

intermediereket, ahol a P_{abc}^{ijk} a

$$P_{abc}^{ijk} = \begin{pmatrix} abc \\ ijk \end{pmatrix} + \begin{pmatrix} acb \\ ikj \end{pmatrix} + \begin{pmatrix} cab \\ kij \end{pmatrix} + \begin{pmatrix} cba \\ kji \end{pmatrix} + \begin{pmatrix} bca \\ jki \end{pmatrix} + \begin{pmatrix} bac \\ jik \end{pmatrix} \quad (3.4)$$

permutációs operátor. A energianevezőt a

$$D_{abc}^{ijk} = f_{ii} + f_{jj} + f_{kk} - f_{aa} - f_{bb} - f_{cc} \quad (3.5)$$

képlet definiálja. Ez azt jelenti, hogy a W_{abc}^{ijk} felépítéséhez 2×6 darab mátrix-szorzást kell elvégezni. Ha kihasználjuk a W_{abc}^{ijk} és V_{abc}^{ijk} permutációs szimmetriáját, azaz például

$$W_{abc}^{ijk} = W_{cab}^{kij} = W_{cba}^{kji}, \text{ stb.}, \quad (3.6)$$

megkötéseket vezethetünk be a szummákban belül az indexekre és ezzel csökkenthetjük a műveletigényt. A közleményben meg is adták a megfelelő képletet. Azonban ez hibás, ezért ezt levezettem és a javított

$$\begin{aligned} E_{(T)} = & 2 \sum_{i \geq j \geq k} \sum_{a \geq b \geq c} [(Y_{abc}^{ijk} - 2Z_{abc}^{ijk})(W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk}) \\ & + [(Z_{abc}^{ijk} - 2Y_{abc}^{ijk})(W_{acb}^{ijk} + W_{bac}^{ijk} + W_{cba}^{ijk}) + 3X_{abc}^{ijk}] \\ & / [D_{abc}^{ijk}(1 + \delta_{ijk})(1 + \delta_{abc})] \end{aligned} \quad (3.7)$$

képletet használtam a programban, ahol a δ_{ijk} -t, ill. a δ_{abc} -t a

$$\delta_{ijk} = \delta_{ij} + \delta_{jk}, \quad (3.8)$$

$$\delta_{abc} = \delta_{ab} + \delta_{bc} \quad (3.9)$$

módon, az X , Y , Z tömböket pedig a

$$\begin{aligned} X_{abc}^{ijk} = & W_{abc}^{ijk}V_{abc}^{ijk} + W_{bca}^{ijk}V_{bca}^{ijk} + W_{cab}^{ijk}V_{cab}^{ijk} \\ & + W_{acb}^{ijk}V_{acb}^{ijk} + W_{bac}^{ijk}V_{bac}^{ijk} + W_{cba}^{ijk}V_{cba}^{ijk}, \end{aligned} \quad (3.10)$$

$$Y_{abc}^{ijk} = V_{abc}^{ijk} + V_{bca}^{ijk} + V_{cab}^{ijk}, \quad (3.11)$$

$$Z_{abc}^{ijk} = V_{acb}^{ijk} + V_{bac}^{ijk} + V_{cba}^{ijk} \quad (3.12)$$

módon definiáljuk. A levezetés az A. függelékben található.

Ezt a képletet átalakíthatjuk úgy, hogy megcseréljük a szummázást a betöltött és a virtuális indexekre a

$$\begin{aligned} E_{(T)} = & 2 \sum_{a \geq b \geq c} \sum_{i \geq j \geq k} [(Y_{abc}^{ijk} - 2Z_{abc}^{ijk})(W_{abc}^{ijk} + W_{abc}^{kij} + W_{abc}^{jki}) \\ & + [(Z_{abc}^{ijk} - 2Y_{abc}^{ijk})(W_{abc}^{ikj} + W_{abc}^{jik} + W_{abc}^{kji}) + 3X_{abc}^{ijk}] \\ & / [D_{abc}^{ijk}(1 + \delta_{ijk})(1 + \delta_{abc})] \end{aligned} \quad (3.13)$$

módon, ahol most a permutációs szimmetriákat úgy használtuk, hogy a virtuális indexek sorrendje mindenhol abc legyen. Ekkor az X , Y , Z tömbök az

$$\begin{aligned} X_{abc}^{ijk} = & W_{abc}^{ijk}V_{abc}^{ijk} + W_{abc}^{kij}V_{abc}^{kij} + W_{abc}^{jki}V_{abc}^{jki} \\ & + W_{abc}^{ikj}V_{abc}^{ikj} + W_{abc}^{jik}V_{abc}^{jik} + W_{abc}^{kji}V_{abc}^{kji}, \end{aligned} \quad (3.14)$$

$$Y_{abc}^{ijk} = V_{abc}^{ijk} + V_{abc}^{kij} + V_{abc}^{jki}, \quad (3.15)$$

$$Z_{abc}^{ijk} = V_{abc}^{ikj} + V_{abc}^{jik} + V_{abc}^{kji} \quad (3.16)$$

szerint változnak. Ez hasznos lehet akkor, amikor a rendelkezésre álló memória nem elegendő a (3.7) képlet használatához, de erre a későbbiekben kitérünk.

3.2. Az algoritmus

3.2.1. A memória szervezése

A CCSD futás elején lefoglalt memóriát átrendezzük. A továbbra is szükséges tömbök új helyre kerülnek azonos néven. A (T) korrekció számításához szükséges további tömbökhöz hozzárendeljük a méretüknek megfelelő memóriaterületet. A tömbök sorrendjét a (3.1) táblázat tartalmazza. A W és V tömbökhöz két n_v^3 méretű memóriaterületet rendelünk a rögzített ijk indexekre [lsd. 42. old. 3. pont].

3.1. táblázat. Tömbök elnevezése és sorrendje a (T) korrekció számításakor

név az egyenletekben	név a kódban
-	scrs
-	scro
f_{ii}	fdo
f_{aa}	fdv
$\mathbf{t}$	t
$\mathbf{T}$	tt
$\langle ai jk \rangle$	aijk
$\langle ab ij \rangle$	abij
W	w
V	v

3.2.2. A számolás előtti műveletek

Mivel a (T) korrekció kiszámolásához csak a Fock-mátrix diagonális elemeire lesz szükségünk, kiválogatjuk ezeket az

$$\text{fdo}(\mathbf{i}) = \text{fij}(\mathbf{i}, \mathbf{i}), \quad (3.17)$$

$$\text{fdv}(\mathbf{a}) = \text{fab}(\mathbf{a}, \mathbf{a}) \quad (3.18)$$

egyenletek szerint. Átmásoljuk a $\mathbf{t}$ tömböt az új helyére. Felbontjuk az $i \geq j$ indexek megkötését a $\mathbf{tt}$ tömbben és az így kapott tömböt az új helyére helyezzük. Beolvassuk a merevlemezről a teljes $\langle ai|jk \rangle$ és $\langle ab|ij \rangle$ integrállistát a memóriába.

3.2.3. A (T) korrekció kiszámítása

A (3.7) képletet programoztuk be az alábbi lépések szerint.

1. Beolvasunk az $\langle ab|ck \rangle$ integrállistából annyi három virtuális indexes blokkot, amennyi befér a memóriába, ezek betöltött indexe $k = k_{min}, \dots, k_{max}$. Fenntartunk további két blokknak elegendő helyet.
2. Elindítunk három egymásba ágyazott ciklust k, j, i indexekre úgy, hogy a (3.7) képletben az első szummában lévő megkötések ($i \geq j \geq k$) és a $k_{min} \leq k \leq k_{max}$ teljesüljenek. Ha $i \notin [k_{min}, k_{max}]$ vagy $j \notin [k_{min}, k_{max}]$, akkor a két fenntartott helyre beolvassuk az $\langle ab|ci \rangle$, ill. $\langle ab|cj \rangle$ integrállista három virtuális indexes blokkját.
3. Rögzített ijk indexre kiszámoljuk az

$$\text{fkji} = \text{fdo}(\mathbf{k}) + \text{fdo}(\mathbf{j}) + \text{fdo}(\mathbf{i}), \quad (3.19)$$

$$\text{delta3} = 1 + \delta_{ijk} \quad (3.20)$$

értékeket, valamint a $W_{\dots}^{ijk}$ (**w**) és a $V_{\dots}^{ijk}$ (**v**) tömböket a (3.2), ill. a (3.3) egyenletek alapján. Először **w**-t számoljuk ki, ez kétszer hat darab művelet a következők szerint. Az első tagok járulékai:

(a)

$$\sum_d \langle ac|dk \rangle T_{bd}^{ji} = \sum_d \langle ac|dk \rangle T_{db}^{ij} \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.21)$$

Itt adott k indexhez tartozó integrálokból álló ac sor-hiperindexű és d oszlopindexű mátrixot megszorozunk egy adott ij indexhez tartozó d sorindexű, b oszlopindexű klaszteramplitúdókból álló mátrixszal a **dgemm** rutin segítségével. Az eredményt a **w**-ben tároljuk, majd felcseréljük az utolsó két indexét.

(b)

$$\sum_d \langle ba|di \rangle T_{cd}^{kj} = \sum_d \langle da|bi \rangle T_{dc}^{jk} \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.22)$$

Itt adott i indexhez tartozó d sorindexű, ab oszlop-hiperindexű mátrix transzponáltját szorozzuk meg egy adott jk indexhez tartozó d sorindexű, c oszlopindexű mátrixszal a **dgemm** rutin segítségével. Az eredményt hozzáadjuk **w**-hez.

(c)

$$\sum_d \langle ca|di\rangle T_{bd}^{jk} = \sum_d \langle ca|di\rangle T_{db}^{kj} \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.23)$$

Minden c -re, adott i indexhez tartozó a sorindexű, d oszlopindexű mátrixot szorozzuk meg az adott kj indexhez tartozó d sorindexű, b oszlopindexű mátrixszal a `dgemm` rutin segítségével. Az eredményt hozzáadjuk $\mathbf{w}(\cdot, \cdot, \mathbf{c})$ -hez.

(d)

$$\sum_d \langle bc|dk\rangle T_{ad}^{ji} = \sum_d T_{da}^{ji} \langle bc|dk\rangle \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.24)$$

Adott ji indexhez tartozó d sorindexű, a oszlopindexű mátrixot szorozzuk meg adott k indexhez tartozó bc sor-hiperindexű, d oszlopindexű mátrix transzponáltjával a `dgemm` rutin segítségével. Az eredményt hozzáadjuk $\mathbf{w}$ -hez.

(e)

$$\sum_d \langle cb|dj\rangle T_{ad}^{ik} = \sum_d T_{ad}^{ik} \langle db|cj\rangle \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.25)$$

Adott ik indexhez tartozó a sorindexű, d oszlopindexű mátrixot szorozzuk meg adott j indexhez tartozó d sorindexű, bc oszlop hiperindexű mátrixszal a `dgemm` rutin segítségével. Az eredményt hozzáadjuk $\mathbf{w}$ -hez.

(f)

$$\sum_d \langle ab|dj\rangle T_{cd}^{ki} = \sum_d \langle ab|dj\rangle T_{dc}^{ik} \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.26)$$

Adott j indexhez tartozó ab sor-hiperindexű, d oszlopindexű mátrixot szorozzuk meg adott ik indexhez tartozó d sorindexű, c oszlopindexű mátrixszal a `dgemm` rutin segítségével. Az eredményt hozzáadjuk $\mathbf{w}$ -hez.

A második tagok járulékai:

(a)

$$-\sum_l \langle cj|kl\rangle T_{ab}^{il} = -\sum_l \langle cl|kj\rangle T_{ab}^{il} \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.27)$$

Itt adott ijk indexekre, minden c és l indexre hozzáadjuk $\mathbf{w}(\cdot, \cdot, \mathbf{c})$ -hez `tt`($\cdot, \cdot, \mathbf{i}, 1$) az n_v^2 hosszúságú vektor $-\mathbf{aijk}(\mathbf{c}, \mathbf{l}, \mathbf{k}, \mathbf{j})$ -szeresét a `daxpy` rutin segítségével.

(b)

$$-\sum_l \langle bk|jl \rangle T_{ac}^{il} = -\sum_l T_{ac}^{il} \langle bl|jk \rangle \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.28)$$

Adott ijk indexekre, minden c és l indexre kiszámoljuk a $\mathbf{tt}(\cdot, \mathbf{c}, \mathbf{i}, \mathbf{l})$ vektor és az $\mathbf{aijk}(\cdot, \mathbf{l}, \mathbf{j}, \mathbf{k})$ vektor diadikus szorzatát a `daxpy` rutin segítségével, és levonjuk $\mathbf{w}(\cdot, \cdot, \mathbf{c})$ -ből.

(c)

$$-\sum_l \langle bi|jl \rangle T_{ca}^{kl} = -\sum_l T_{ca}^{lk} \langle bl|ji \rangle \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.29)$$

Adott ijk indexekre, minden c és l indexre kiszámoljuk a $\mathbf{tt}(\cdot, \mathbf{c}, \mathbf{l}, \mathbf{k})$ vektor és az $\mathbf{aijk}(\cdot, \mathbf{l}, \mathbf{j}, \mathbf{i})$ vektor diadikus szorzatát a `dger` rutin segítségével és levonjuk $\mathbf{w}(\cdot, \cdot, \mathbf{c})$ -ből.

(d)

$$-\sum_l \langle ak|il \rangle T_{bc}^{jl} = -\sum_l \langle al|ik \rangle T_{bc}^{jl} \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.30)$$

Adott ijk indexekre, minden l indexre kiszámoljuk az $\mathbf{aijk}(\cdot, \mathbf{l}, \mathbf{i}, \mathbf{k})$ vektor és a $\mathbf{tt}(\cdot, \cdot, \mathbf{l}, \mathbf{j})$ n_v^2 hosszúságú vektor diadikus szorzatát a `dger` rutin segítségével és levonjuk $\mathbf{w}$ -ből.

(e)

$$-\sum_l \langle aj|il \rangle T_{cb}^{kl} = -\sum_l \langle al|ij \rangle T_{cb}^{lk} \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.31)$$

Adott ijk indexekre kiszámoljuk az $\mathbf{aijk}(\cdot, \cdot, \mathbf{i}, \mathbf{j})$ a sorindexű, l oszlopindexű mátrix és a $\mathbf{tt}(\cdot, \cdot, \cdot, \mathbf{k})$ bc sor-hiperindexű, l oszlopindexű mátrix transzponáltjának szorzatát a `dgemm` rutin segítségével. Az eredményt levonjuk $\mathbf{w}$ -ből.

(f)

$$-\sum_l \langle ci|kl \rangle T_{ba}^{jl} = -\sum_l T_{ab}^{lj} \langle cl|ki \rangle \rightarrow \mathbf{w}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.32)$$

Adott ijk indexekre kiszámoljuk a $\mathbf{tt}(\cdot, \cdot, \cdot, \mathbf{j})$ ab sor-hiperindexű, l oszlopindexű mátrix és az $\mathbf{aijk}(\cdot, \cdot, \mathbf{i}, \mathbf{k})$ c sorindexű, l oszlopindexű mátrix transzponáltjának szorzatát a `dgemm` rutin segítségével. Az eredményt levonjuk $\mathbf{w}$ -ből.

Ezután $\mathbf{v}$ -t számoljuk ki.

- (a) Bemásoljuk $\mathbf{w}$ -t $\mathbf{v}$ -be.
- (b) Kiszámoljuk (3.3) második tagját

$$\langle bc|jk\rangle t_a^i \rightarrow \mathbf{v}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.33)$$

szerint. Adott i, j és k indexekre kiszámoljuk a $\mathbf{t}(\cdot, \mathbf{i})$ vektor és az $\mathbf{abij}(\cdot, \cdot, \mathbf{j}, \mathbf{k})$ vektor diadikus szorzatát a $\mathbf{dger}$ rutinnal és hozzáadjuk $\mathbf{v}$ -hez.

- (c) Kiszámoljuk (3.3) harmadik tagját

$$\langle ac|ik\rangle t_b^j \rightarrow \mathbf{v}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.34)$$

szerint. Adott i, j és k indexekre, minden c indexre kiszámoljuk az $\mathbf{abij}(\cdot, \mathbf{c}, \mathbf{i}, \mathbf{k})$ vektor és a $\mathbf{t}(\cdot, \mathbf{j})$ vektor diadikus szorzatát a $\mathbf{dger}$ rutinnal és hozzáadjuk $\mathbf{v}(\cdot, \cdot, \mathbf{c})$ -hez.

- (d) Kiszámoljuk (3.3) negyedik tagját

$$\langle ab|ij\rangle t_c^k \rightarrow \mathbf{v}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \quad (3.35)$$

szerint. Adott ijk indexekre kiszámoljuk az $\mathbf{abij}(\cdot, \cdot, \mathbf{i}, \mathbf{j})$ vektor és a $\mathbf{t}(\cdot, \mathbf{k})$ vektor diadikus szorzatát a $\mathbf{dger}$ rutinnal és hozzáadjuk $\mathbf{v}$ -hez.

4. Elindítunk egy ciklust a virtuális indexekre a (3.7) egyenlet szerint. Kiszámoljuk adott abc indexre az

$$\mathbf{fcb} = \mathbf{fdv}(\mathbf{c}) + \mathbf{fdv}(\mathbf{b}) + \mathbf{fdv}(\mathbf{a}), \quad (3.36)$$

$$\mathbf{delta} = 1 + \delta_{abc} \quad (3.37)$$

számokat valamint a szummában található tagokat, és hozzáadjuk a korrekciónak fenntartott változóhoz. Itt most a nevezőt a

$$D_{abc}^{ijk}(1 + \delta_{abc})(1 + \delta_{ijk}) = (\mathbf{fkji} - \mathbf{fcb}) \cdot \mathbf{delta} \cdot \mathbf{delta}^3 \quad (3.38)$$

képlet szerint számoljuk.

5. Ezután ismét beolvasunk annyi háromindexes blokkot a memóriába az $\langle ab|ck\rangle$ integrállistából, amennyi a memóriába fér, majd ismét végrehajtjuk a (2)-(5) lépéseket, amíg a teljes $\langle ab|ck\rangle$ integrállistát végig nem olvastuk.
6. A korrekciót megszorozzuk 2-vel.

3.3. Módosítások a lokális számításhoz

A lokális CCSD(T) számolásnál a (3.7) képletben lévő betöltött indexekre való szumma legkülső indexére nem kell összegezni, helyette transzformálni kell az energiaképletben szereplő összes tömb adott indexét. Ezt ebben a formában nem tudjuk elvégezni, mivel az indexekre megszorítás van érvényben és ezért nem tudjuk leválasztani a szummát a képletből. A (3.7) képletből kiindulva csak egy kedvezőtlen skálázódású, nagy memóriaigényű algoritmust tudnánk kidolgozni. Ennek az az oka, hogy amint később a (3.43)-(3.48) egyenleteknél látjuk, a W és V tömbök mindhárom betöltött indexét transzformálnunk kell.

Itt ezt egyrészt úgy tehetnénk meg, hogy a k indexre feloldjuk a megkötéseket és leválasztjuk az erre futó szummát. Az indexek transzformációjához a fennmaradó szummákat ($i \geq j$ -re és $a \geq b \geq c$ -re) el kéne végezni W és V egy-egy indexére, azaz n_o^2 -szer, ennek a skálázódása n^9 -es. Azaz az így kapható módszer rosszabbul fog skálázódni a domén méretével, mint a hagyományos (T) korrekció számítása.

Másrészt azt is megtehetjük, hogy az $i \geq j$ indexekre futó cikluson belül nem három, hanem négy indexes ($n_o n_v^3$ méretű) tömböket vezetünk be W és V tárolására. Ezekből három-három darabot kéne a memóriában tartanunk, holott általában egy sem fér be belőlük a memóriába.

Kiindulhatnánk a (3.13) egyenletből is, azonban itt is megszorítás van érvényben a legkülső betöltött indexre. Ezért ahelyett, hogy feloldanánk az indexek megszorítását valamilyen módon, új képletet vezettünk le a (3.1) egyenletből úgy, hogy a lehető legtöbb megszorítást alkalmazhassuk az indexekre. A levezetést a B. függelék tartalmazza. A (T) korrekciót az

$$\begin{aligned}
E_{(T)} = \sum_{a \geq b \geq c} \left[\sum_{i > j} \sum_k [& W_{abc}^{ijk} (2V_{abc}^{ijk} - V_{abc}^{ikj} - V_{abc}^{kji}) + W_{abc}^{kij} (2V_{abc}^{kij} - V_{abc}^{ikj} - V_{abc}^{jik}) \\
& + W_{abc}^{jki} (2V_{abc}^{jki} - V_{abc}^{jik} - V_{abc}^{kji}) + W_{abc}^{ikj} (2V_{abc}^{ikj} - V_{abc}^{kij} - V_{abc}^{ijk}) \\
& + W_{abc}^{jik} (2V_{abc}^{jik} - V_{abc}^{jki} - V_{abc}^{kij}) + W_{abc}^{kji} (2V_{abc}^{kji} - V_{abc}^{ijk} - V_{abc}^{jki}) \\
& + \frac{2}{3} (W_{abc}^{ijk} + W_{abc}^{kij} + W_{abc}^{jki} - W_{abc}^{ikj} + W_{abc}^{jik} + W_{abc}^{kji}) \\
& \times (V_{abc}^{ijk} + V_{abc}^{kij} + V_{abc}^{jki} - V_{abc}^{ikj} + V_{abc}^{jik} + V_{abc}^{kji})] / D_{abc}^{ijk} \\
& + \frac{1}{3} \sum_{ik} [W_{abc}^{iik} (3V_{abc}^{iik} - \bar{Y}_{abc}^{ik}) + W_{abc}^{iki} (3V_{abc}^{iki} - \bar{Y}_{abc}^{ik}) \\
& + W_{abc}^{kii} (3V_{abc}^{kii} - \bar{Y}_{abc}^{ik})] / D_{abc}^{iik} \Big] / (1 + \delta_{abc}) \quad (3.39)
\end{aligned}$$

képlet adja meg, ahol bevezettük az

$$\bar{Y}_{abc}^{ik} = V_{abc}^{iik} + V_{abc}^{iki} + V_{abc}^{kii} \quad (3.40)$$

menyiséget. Az energianevezővel leosztjuk a W tömb elemeit a

$$\tilde{W}_{abc}^{ijk} = W_{abc}^{ijk} / D_{abc}^{ijk} \quad (3.41)$$

képlet szerint. Így az

$$\begin{aligned} E_{(T)} = \sum_{a \geq b \geq c} & \left[\sum_{i > j} \sum_k [\tilde{W}_{abc}^{ijk} (2V_{abc}^{ijk} - V_{abc}^{ikj} - V_{abc}^{kji}) + \tilde{W}_{abc}^{kij} (2V_{abc}^{kij} - V_{abc}^{ikj} - V_{abc}^{jik}) \right. \\ & + \tilde{W}_{abc}^{jki} (2V_{abc}^{jki} - V_{abc}^{jik} - V_{abc}^{kji}) + \tilde{W}_{abc}^{ikj} (2V_{abc}^{ikj} - V_{abc}^{kij} - V_{abc}^{ijk}) \\ & + \tilde{W}_{abc}^{jik} (2V_{abc}^{jik} - V_{abc}^{kji} - V_{abc}^{kij}) + \tilde{W}_{abc}^{kji} (2V_{abc}^{kji} - V_{abc}^{ijk} - V_{abc}^{jki}) \\ & + \frac{2}{3} (\tilde{W}_{abc}^{ijk} + \tilde{W}_{abc}^{kij} + \tilde{W}_{abc}^{jki} - \tilde{W}_{abc}^{ikj} - \tilde{W}_{abc}^{jik} - \tilde{W}_{abc}^{kji}) \\ & \times (V_{abc}^{ijk} + V_{abc}^{kij} + V_{abc}^{jki} - V_{abc}^{ikj} - V_{abc}^{jik} - V_{abc}^{kji})] \\ & + \frac{1}{3} \sum_{ik} [\tilde{W}_{abc}^{iik} (3V_{abc}^{iik} - \bar{Y}_{abc}^{ik}) + \tilde{W}_{abc}^{iki} (3V_{abc}^{iki} - \bar{Y}_{abc}^{ik}) \\ & \left. + \tilde{W}_{abc}^{kii} (3V_{abc}^{kii} - \bar{Y}_{abc}^{ik})] \right] / (1 + \delta_{abc}) \quad (3.42) \end{aligned}$$

képletet kapjuk az energiára. A $\tilde{W}_{abc}^{ijk}$ és a V_{abc}^{ijk} tömbök k indexét transzformáljuk rögzített abc indexekre

$$v1_{abc}^{ij} = \sum_k U_{lk} V_{abc}^{kij}, \quad (3.43)$$

$$v2_{abc}^{ij} = \sum_k U_{lk} V_{abc}^{ikj}, \quad (3.44)$$

$$v3_{abc}^{ij} = \sum_k U_{lk} V_{abc}^{ijk}, \quad (3.45)$$

$$w1_{abc}^{ij} = \sum_k U_{lk} \tilde{W}_{abc}^{kij}, \quad (3.46)$$

$$w2_{abc}^{ij} = \sum_k U_{lk} \tilde{W}_{abc}^{ikj}, \quad (3.47)$$

$$w3_{abc}^{ij} = \sum_k U_{lk} \tilde{W}_{abc}^{ijk} \quad (3.48)$$

szerint. Ezeket behelyettesítve a (3.42) képletbe a l . parciális energiára

$$\begin{aligned} \delta E_{(T)}^l = & \sum_{a \geq b \geq c} \left[\sum_{i > j} [w 3_{abc}^{ij} (2v 3_{abc}^{ij} - v 2_{abc}^{ij} - v 1_{abc}^{ji}) + w 1_{abc}^{ij} (2v 1_{abc}^{ij} - v 2_{abc}^{ij} - v 3_{abc}^{ji}) \right. \\ & + w 2_{abc}^{ji} (2v 2_{abc}^{ji} - v 3_{abc}^{ji} - v 1_{abc}^{ji}) + w 2_{abc}^{ij} (2v 2_{abc}^{ij} - v 1_{abc}^{ij} - v 3_{abc}^{ji}) \\ & + w 3_{abc}^{ji} (2v 3_{abc}^{ji} - v 2_{abc}^{ji} - v 1_{abc}^{ij}) + w 1_{abc}^{ji} (2v 1_{abc}^{ji} - v 3_{abc}^{ij} - v 2_{abc}^{ji}) \\ & + \frac{2}{3} (w 3_{abc}^{ij} + w 1_{abc}^{ij} + w 2_{abc}^{ji} - w 2_{abc}^{ij} - w 3_{abc}^{ji} - w 1_{abc}^{ji}) \\ & \times (v 3_{abc}^{ij} + v 1_{abc}^{ij} + v 2_{abc}^{ji} - v 2_{abc}^{ij} - v 3_{abc}^{ji} - v 1_{abc}^{ji})] \\ & + \frac{1}{3} \sum_i [w 3_{abc}^{ii} (3v 3_{abc}^{ii} - \bar{y}_{abc}^i) + w 3_{abc}^{ii} (3v 2_{abc}^{ii} - \bar{y}_{abc}^i) \\ & \left. + w 1_{abc}^{ii} (3v 1_{abc}^{ii} - \bar{y}_{abc}^i)] \right] / (1 + \delta_{abc}) \end{aligned} \quad (3.49)$$

képletet kapjuk, ahol most $\bar{y}_{abc}^i$ -t az

$$\bar{y}_{abc}^i = v 1_{abc}^{ii} + v 2_{abc}^{ii} + v 3_{abc}^{ii} \quad (3.50)$$

módon definiáltuk.

3.4. Algoritmus kevés memória esetén és lokális számoláshoz

Ha a rendelkezésre álló memória akkora, hogy nem fér bele öt darab n_v^3 méretű tömb, ill. ha lokális számítást végzünk nem használhatjuk a (3.2) pontban részletezett algoritmust. Ennek elkerülése érdekében egy másik, kisebb memóriaigényű algoritmusra van szükségünk. Mindkét esetben ugyanúgy indul az algoritmus, az alapvető különbség, hogy az energiaképletben (3.13, 3.49) a virtuális indexekre futó szumma megelőzi a betöltöttökre futót. Azaz most az a , b és c indexekre futó ciklusok vannak kívül és az adott abc indexre a belső ciklusokban építjük fel a megfelelő tömbök összes ijk indexű elemét. Így csak n_o^3 méretű tömböket kell tárolnunk a memóriában. Lokális számításnál ezeknek a tömböknek kell az egyik indexét transzformálni, ami így kézenfekvő.

3.4.1. A memória szervezése

A memóriát a (3.2.1) pont szerint rendezzük át azzal a különbséggel, hogy a v és a w tömbökhöz egy-egy n_o^3 méretű memóriarészt rendelünk hozzá. Lokális számolás esetén a (3.43)-(3.48) tömbök számára lefoglalunk egy-egy n_o^2 méretű memóriarészt.

3.4.2. A számolás előtti lépések

Átrendezzük a $\mathbf{tt}$ tömb elemeit úgy, hogy a betöltött indexek legyenek elől és a virtuálisak hátul, azaz

$$\mathbf{tt}(\mathbf{a}, \mathbf{b}, \mathbf{i}, \mathbf{j}) \rightarrow \mathbf{tt}(\mathbf{i}, \mathbf{j}, \mathbf{a}, \mathbf{b}). \quad (3.51)$$

Átrendezzük az $\langle ab|ci \rangle$ integrállistát úgy, hogy a betöltött index legyen elől és a virtuálisak hátul, azaz

$$\langle ab|ci \rangle \rightarrow \langle ia|bc \rangle, \quad (3.52)$$

és az új integrállistát a merevlemezre írjuk. Ilyen módon be tudjuk olvasni tetszőleges c indexre az $\langle ia|bc \rangle$ integrállista teljes három indexes (iab) blokkját a memóriába.

Beolvassuk a memóriába a teljes $\langle ab|ij \rangle$ integrállistát az $\mathbf{abij}$ tömbbe. Beolvassuk a memóriába a teljes $\langle ai|jk \rangle$ integrállistát az $\mathbf{ijka}$ tömbbe, úgy, hogy az utolsó helyen álljon a virtuális index, azaz

$$\langle ai|jk \rangle \rightarrow \mathbf{ijka}(\mathbf{i}, \mathbf{j}, \mathbf{k}, \mathbf{a}) \quad (3.53)$$

3.4.3. A (T) korrekció kiszámítása

A (3.13), ill. a (3.49) képleteket programoztuk be az alábbi lépések szerint.

1. Beolvasunk az $\langle ia|bc \rangle$ integrállistából annyi három indexes blokkot, amennyi befér a memóriába, ezek betöltött indexe $c = c_{min}, \dots, c_{max}$. Fenntartunk további két blokknak elegendő helyet.
2. Elindítunk három egymásba ágyazott ciklust c, b, a indexekre úgy, hogy az energiaképletekben az első szummában lévő megkötések ($a \geq b \geq c$) és a $c_{min} \leq c \leq c_{max}$ teljesüljenek. Ha $a \notin [c_{min}, c_{max}]$ vagy $b \notin [c_{min}, c_{max}]$, akkor a két fenntartott helyre beolvassuk az $\langle ia|bd \rangle$, ill. $\langle ia|bd \rangle$ integrállista három virtuális indexes blokkját, ahol most az utolsó index a , ill. c .
3. A rögzített abc indexre kiszámoljuk az

$$\mathbf{fcba} = \mathbf{fdv}(\mathbf{c}) + \mathbf{fdv}(\mathbf{b}) + \mathbf{fdv}(\mathbf{a}), \quad (3.54)$$

$$\mathbf{delta3} = 1 + \delta_{abc} \quad (3.55)$$

értékeket, valamint a $W_{\dots}^{abc}(\mathbf{w})$ és a $V_{\dots}^{abc}(\mathbf{v})$ tömböket a (3.2), ill. a (3.3) egyenletek alapján. Először $\mathbf{w}$ -t számoljuk ki, ez kétszer hat darab művelet a következők szerint. Az első tagok járulécai:

(a)

$$\sum_d \langle ba|di\rangle T_{cd}^{kj} = \sum_d \langle id|ab\rangle T_{dc}^{jk} = \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.56)$$

Itt adott ab indexhez tartozó i sorindexű, d oszlopindexű integrálokból álló mátrixot szorozzuk meg egy adott c indexhez tartozó jk sor-hiperindexű, d oszlopindexű klaszteramplitúdókból álló mátrix transzponáltjával a **dgemm** rutin segítségével. Az eredményt **w** tömbbe írjuk.

(b)

$$\sum_d \langle ca|di\rangle T_{bd}^{jk} = \sum_d \langle id|ac\rangle T_{db}^{kj} = \mathbf{scro}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.57)$$

Itt adott ac indexhez tartozó i sorindexű, d oszlopindexű mátrixot szorozzuk meg egy adott b indexhez tartozó kj sor-hiperindexű, d oszlopindexű mátrix transzponáltjával a **dgemm** rutin segítségével. Az eredményt **scro** tömbbe írjuk. Azért nem **w** tömbhöz adjuk hozzá az eredményt, mert a fenti mátrixszorzás eredményeképpen kapott tömb utolsó két indexe fel van cserélve. Ezeket a következő lépés után cseréljük meg.

(c)

$$\sum_d \langle ab|dj\rangle T_{cd}^{ki} = \sum_d T_{dc}^{ik} \langle jd|ba\rangle \rightarrow \mathbf{scro}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.58)$$

Adott c indexhez tartozó ik sor-hiperindexű, d oszlopindexű mátrixot szorozzuk meg adott ba indexhez tartozó j sorindexű, d oszlopindexű mátrix transzponáltjával a **dgemm** rutin segítségével. Az eredményt hozzáadjuk **scro**-hoz. Az **scro** tömb utolsó két indexét felcseréljük és hozzáadjuk **w**-hez, azaz

$$\mathbf{scro}(\mathbf{i}, \mathbf{k}, \mathbf{j}) \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}). \quad (3.59)$$

(d)

$$\sum_d \langle ac|dk\rangle T_{bd}^{ji} = \sum_d T_{db}^{ij} \langle kd|ca\rangle \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.60)$$

Itt adott b indexhez tartozó ij sor-hiperindexű, d oszlopindexű mátrixot adott ca indexhez tartozó k sorindexű, d oszlopindexű mátrix transzponáltjával szorozzuk meg a **dgemm** rutin segítségével. Az eredményt hozzáadjuk **w**-hez.

(e)

$$\sum_d \langle bc|dk\rangle T_{ad}^{ij} = \sum_d T_{da}^{ji} \langle kd|cb\rangle = \text{scro}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.61)$$

Adott a indexhez tartozó ji sor-hiperindexű, d oszlopindexű mátrixot szorozzuk meg adott cb indexhez tartozó k sorindexű, d oszlopindexű mátrix transzponáltjával a **dgemm** rutin segítségével. Az eredményt az **scro** tömbbe írjuk. Az **scro** első két indexét felcseréljük és hozzáadjuk **w**-hez, azaz

$$\text{scro}(\mathbf{j}, \mathbf{i}, \mathbf{k}) \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}). \quad (3.62)$$

(f)

$$\sum_d \langle cb|dj\rangle T_{ad}^{ik} = \sum_d T_{da}^{ki} \langle jd|bc\rangle = \text{scro}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.63)$$

Adott a indexhez tartozó ki sor-hiperindexű, d oszlopindexű mátrixot szorozzuk meg adott bc indexhez tartozó j sorindexű, d oszlopindexű mátrix transzponáltjával a **dgemm** rutin segítségével. Az eredményt az **scro** tömbbe írjuk. **scro** első két indexét felcseréljük a harmadikkal és hozzáadjuk **w**-hez, azaz

$$\text{scro}(\mathbf{k}, \mathbf{i}, \mathbf{j}) \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}). \quad (3.64)$$

A második tagok járulékai:

(a)

$$-\sum_l \langle cj|kl\rangle T_{ab}^{il} = -\sum_l T_{ab}^{il} \langle jk|lc\rangle \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.65)$$

Adott ab indexhez tartozó i sorindexű, l oszlopindexű mátrixot megszorozzuk adott c indexhez tartozó jk sor-hiperindexű, l oszlopindexű mátrix transzponáltjával a **dgemm** rutin segítségével. Az eredményt hozzáadjuk **w**-hez.

(b)

$$-\sum_l \langle bk|jl\rangle T_{ac}^{il} = -\sum_l T_{ac}^{il} \langle lj|kb\rangle \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.66)$$

Adott ac indexhez tartozó i sorindexű, l oszlopindexű mátrixot megszorozzuk adott b indexhez tartozó l sorindexű, jk oszlop-hiperindexű mátrixszal a **dgemm** rutin segítségével. Az eredményt hozzáadjuk **w**-hez.

(c)

$$-\sum_l \langle bi|jl \rangle T_{ca}^{kl} = -\sum_l \langle ij|lb \rangle T_{ca}^{kl} \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.67)$$

Adott b indexhez tartozó ij sor-hiperindexű, l oszlopindexű mátrixot megszorozzuk az adott ca indexhez tartozó k sorindexű, l oszlopindexű mátrix transzponáltjával a **dgemm** rutin segítségével. Az eredményt hozzáadjuk $\mathbf{w}$ -hez.

(d)

$$-\sum_l \langle aj|il \rangle T_{cb}^{kl} = -\sum_l \langle li|ja \rangle T_{bc}^{lk} \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.68)$$

Minden k indexre, az adott a indexhez tartozó l sorindexű, ij oszlop-hiperindexű mátrix transzponáltját megszorozzuk az adott bc indexhez tartozó $\mathbf{tt}(\cdot, \mathbf{k}, \mathbf{b}, \mathbf{c})$ vektorral a **dgemv** rutin segítségével. Az eredményt hozzáadjuk $\mathbf{w}(\cdot, \cdot, \mathbf{k})$ -hoz.

(e)

$$-\sum_l \langle ak|il \rangle T_{bc}^{jl} = -\sum_l \langle li|ka \rangle T_{bc}^{jl} \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.69)$$

Minden k indexre, az adott a indexhez tartozó $\mathbf{ijka}(\cdot, \cdot, \mathbf{k}, \mathbf{a})$ $n_o \times n_o$ -s mátrix transzponáltját megszorozzuk az adott bc indexhez tartozó $\mathbf{tt}(\cdot, \cdot, \mathbf{b}, \mathbf{c})$ $n_o \times n_o$ -s mátrixszal a **dgemm** rutin segítségével. Az eredményt hozzáadjuk $\mathbf{w}(\cdot, \cdot, \mathbf{k})$ -hoz.

(f)

$$-\sum_l \langle ci|kl \rangle T_{ba}^{jl} = -\sum_l T_{ba}^{jl} \langle ik|lc \rangle \rightarrow \mathbf{w}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.70)$$

Minden l és k indexre, kiszámoljuk az adott c indexhez tartozó n_o hosszúságú $\mathbf{ijka}(\cdot, \mathbf{k}, \mathbf{l}, \mathbf{c})$ vektor és az adott ba indexhez tartozó n_o hosszúságú $\mathbf{tt}(\cdot, \mathbf{l}, \mathbf{b}, \mathbf{a})$ vektor diadikus szorzatát a **dger** rutin segítségével. Az eredményt hozzáadjuk $\mathbf{w}(\cdot, \cdot, \mathbf{k})$ -hoz.

Ezután $\mathbf{v}$ -t számoljuk ki.

(a) Bemásoljuk $\mathbf{w}$ -t $\mathbf{v}$ -be.

(b) Kiszámoljuk (3.3) második tagját

$$\langle bc|jk \rangle t_a^i \rightarrow \mathbf{v}(\mathbf{i}, \mathbf{j}, \mathbf{k}) \quad (3.71)$$

szerint. Az adott abc indexekre kiszámoljuk a $\mathbf{t}(\mathbf{a}, \cdot)$ vektor és az $\mathbf{abij}(\mathbf{b}, \mathbf{c}, \cdot, \cdot)$ vektor diadikus szorzatát a **dger** rutinnal és hozzáadjuk $\mathbf{v}$ -hez.

- (c) Kiszámoljuk (3.3) harmadik tagját

$$\langle ac|ik\rangle t_b^j \rightarrow v(i, j, k) \quad (3.72)$$

szerint. Feltöltjük az `scro` tömb elemeit nullákkal. Adott abc indexekre kiszámoljuk a $t(\cdot, j)$ vektor és az $abij(\cdot, c, i, k)$ vektor diadikus szorzatát a `dger` rutinnal és az eredményt `scro` tömbbe írjuk. Felcseréljük `scro` első két indexét és hozzáadjuk v -hez, azaz

$$scro(j, i, k) \rightarrow v(i, j, k). \quad (3.73)$$

- (d) Kiszámoljuk (3.3) negyedik tagját

$$\langle ab|ij\rangle t_c^k \rightarrow v(i, j, k) \quad (3.74)$$

szerint. Adott abc indexekre kiszámoljuk az $abij(a, b, \cdot, \cdot)$ vektor és a $t(c, \cdot)$ vektor diadikus szorzatát a `dger` rutinnal és hozzáadjuk v -hez.

4. Amikor a hagyományos (T) korrekciót a (3.13) egyenlet szerint számoljuk, a következő lépésekkel folytatjuk az algoritmust.

- (a) Elindítunk egy ciklust a betöltött indexekre a (3.13) egyenlet szerint. Kiszámoljuk adott ijk indexre az

$$fkji = fdo(k) + fdo(j) + fdo(i), \quad (3.75)$$

$$delta = 1 + \delta_{ijk} \quad (3.76)$$

számokat valamint a szummában található tagokat, és hozzáadjuk a korrekciónak fenntartott változóhoz. Itt most a nevezőt a

$$D_{abc}^{ijk}(1 + \delta_{abc})(1 + \delta_{ijk}) = (fkji - fcba) \cdot delta \cdot delta3 \quad (3.77)$$

képlet szerint számoljuk.

- (b) Ezután ismét beolvasunk annyi háromindexes blokkot a memóriába az $\langle ia|bc\rangle$ integrállistából, amennyi a memóriába fér, majd végrehajtjuk a (2)-(4b) lépéseket, amíg a teljes $\langle ia|bc\rangle$ integrállistát végig nem olvastuk.
- (c) A korrekciót megszorozzuk 2-vel.
5. Lokális számolás esetén a a (3.49) egyenlet szerint számoljuk a $\delta E_{(T)}^l$ parciális energiát, a következő lépésekkel folytatjuk az algoritmust.
- (a) Leosztjuk w elemeit az energianevezővel a (3.41) egyenlet szerint.

- (b) Kiszámoljuk a transzformált mennyiségeket a (3.43)-(3.48) egyenletek szerint a `dgemm` rutin segítségével.
- (c) Elindítunk egy ciklust a betöltött indexekre a (3.49) egyenlet szerint, azaz $i > j$, a külső ciklus j -re fut, a második szummát $i \leftrightarrow j$ indexcserével végezzük el.
- (d) Kiszámoljuk a szummákban található tagokat, és hozzáadjuk ennek az `1/delta3`-szorosát a korrekciónak fenntartott változóhoz.
- (e) Ezután ismét beolvasunk annyi háromindexes blokkot a memóriába az $\langle ia|bc \rangle$ integrállistából, amennyi a memóriába fér, majd végrehajtjuk a (2)-(5e) lépéseket, amíg a teljes $\langle ia|bc \rangle$ integrállistát végig nem olvastuk.

4. fejezet

Tesztszámolások

4.1. Általános megjegyzések a teszt számolásokhoz

A programok futásához 7500MB memóriát használtunk, a számolásokat Intel Xeon E3110 3.00GHz processzorokon végeztük. A molekulákban jelen levő atomok törzselektronjait befagyasztottuk, és Dunning-féle cc-pVDZ, cc-pVTZ, cc-pVQZ, ill. cc-pV5Z bázisokat használtunk [55]. A molekulák geometriáját a C függelék tartalmazza.

4.2. Teszt számolások a CCSD(T) programra

Annak érdekében, hogy megbizonyosodjunk arról, hogy a program megfelelően gyorsan számol a feltételezett méretű (100-300 bázisfüggvény) doméneken, teszt számolásokat (benchmarking) végeztünk különböző molekulákra az általam írt programmal és a futási időket összevetettük a napjainkban használatos leggyorsabb molekulapálya integrálokat használó programmal (CFOUR, [56]). A futási eredményeket a (4.1) és a (4.2) táblázatok tartalmazzák (a bázis méretét és típusát a "Bázis" oszlop tartalmazza).

A (4.1) táblázat alapján látjuk, hogy a valós futási ideje egy CCSD iteráció alatt az általam írt programnak minden esetben kisebb, legtöbbször jóval kisebb (fele, tizede, átlagosan kb. ötöde) a CFOUR valós futási idejének. A CPU idő szintén jóval kisebb az általam írt programnál. A valós futási idő / CPU időarány általában jóval kisebb az én programom esetén. Ez azt jelenti, hogy jóval kevesebb idő fordítódik I/O műveletekre, amit végig szem előtt tartottunk a program írásakor.

A (4.2) táblázatnál látszik, hogy ugyan a (T) számítás CPU ideje nem, de

4.1. táblázat. CPU és valós futási idők (wall time, WALL) percben egy iteráció alatt hagyományos CCSD számításokra

Molekula	Bázis		Saját program		CFOUR	
			CPU	WALL	CPU	WALL
CH ₄	311	cc-pV5Z	0.85	7.55	2.10	15.4
Cl ₂	190	cc-pV5Z	0.15	0.18	0.42	2.07
F ₂	182	cc-pV5Z	0.15	0.17	0.42	2.07
H ₂ O ₂	292	cc-pV5Z	1.01	8.86	2.62	13.8
HF	146	cc-pV5Z	0.04	0.04	0.08	0.08
NH ₃	256	cc-pV5Z	0.39	4.70	0.98	6.89
C ₂ H ₆	290	cc-pVQZ	0.98	6.12	1.95	13.1
H ₂ O ₂	170	cc-pVQZ	0.12	0.12	0.25	1.30

4.2. táblázat. CPU és valós futási idők (wall time, WALL) percben hagyományos (T) korrekció számításakor

Molekula	Bázis		Saját program		CFOUR	
			CPU	WALL	CPU	WALL
CH ₄	311	cc-pV5Z	4.08	4.09	3.23	8.67
Cl ₂	190	cc-pV5Z	2.21	2.21	1.99	4.13
F ₂	182	cc-pV5Z	2.20	2.20	1.99	8.32
H ₂ O ₂	292	cc-pV5Z	13.6	13.6	12.3	103
HF	146	cc-pV5Z	0.23	0.23	0.17	0.17
NH ₃	256	cc-pV5Z	1.92	1.92	1.43	5.43
C ₂ H ₆	290	cc-pVQZ	13.8	13.8	11.5	11.8
H ₂ O ₂	170	cc-pVQZ	1.69	1.69	1.45	4.58

valós futási ideje általában jóval kisebb (kb. fele) az általam írt programnak, mint a CFOUR-nak. Számottevő eltérés csak az C₂H₆-nál lép fel, de itt is csak kb. 14%-kal tér el és 290 bázisfüggvény már nem igen fordul elő lokális számítások esetén egy doménben. A bázis mérete kb. megegyezik H₂O₂-nál cc-pV5Z bázisban és a C₂H₆-nál cc-pVQZ bázisban (292, ill. 290), de a CFOUR valós futási ideje jóval nagyobb az előbbinél, ami azért lehet, mert az előbbinél kb. kétszer annyi betöltött pálya van, mint az utóbbinál (13, ill. 7). Emiatt megnő az integrálisták mérete és több időt vesz igénybe a beolvasásuk. Ez a probléma az általam írt kódban láthatóan nem merül fel.

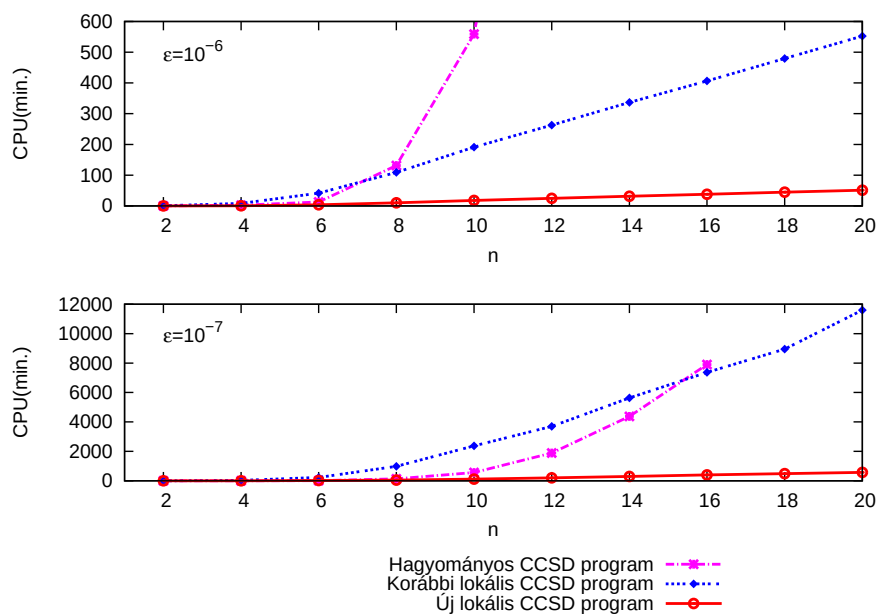
4.3. Tesztszámolások a lokális CCSD(T) programra

A tesztszámolásokat a lokális CCSD(T) módszerre a C_nH_{2n+2} ($n=2,4,6,\dots$) összegképletű normál alkánokon végeztük cc-pVDZ, ill. cc-pVTZ bázisokban, a lokalizációhoz használt paraméter $\varepsilon = 10^{-6}$, ill. $\varepsilon = 10^{-7}$ volt (lsd.: 16. old.). A CCSD számításra és a (T) korrekció számítására kapott CPU időket (amelyek gyakorlatilag megegyeztek a valós futási időkkel) ábráztuk n függvényében mind a nyolc esetben módszer és bázis szerint csoportosítva [(4.1), (4.2), (4.3), (4.4) ábrák]. Ugyanezek az ábrák ábráztuk a korábbi lokális számítások CPU idejét és a hagyományos CCSD számítások CPU idejét is [52].

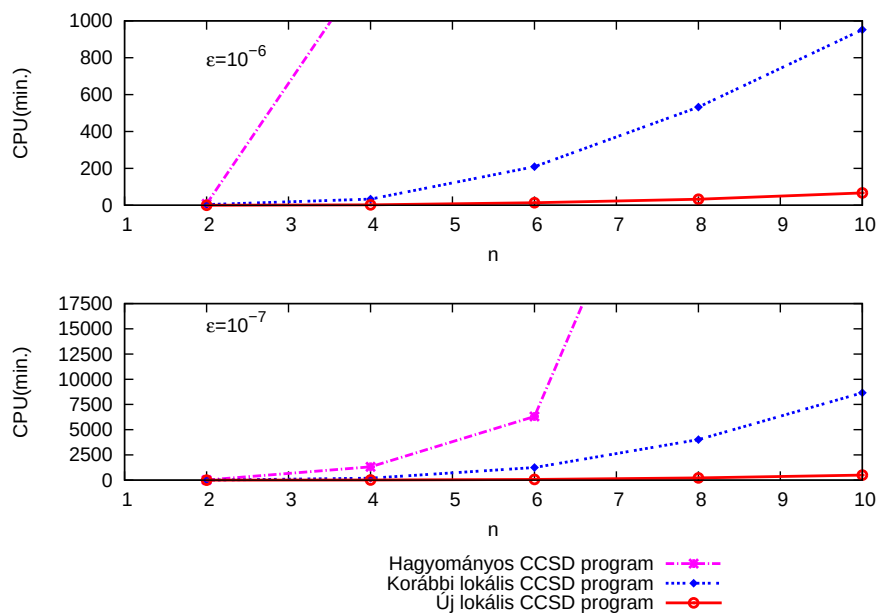
Az ábrákön látható, hogy a hagyományos számítások CPU ideje meredeken nő n -nel (azaz a molekula méretével és a bázisfüggvények számával), a korábbi lokális számítás CPU ideje már lineárisan skálázódik, az általam írt program CPU ideje szintén, viszont ez a korábbihoz képest drasztikusan csökken. Megfigyelhető, hogy a korábbi és az új CPU idők aránya a rendszer méretével nő minden esetben. Az itt bemutatott futási idők közül a legnagyobb eltérés legnagyobb molekulákra, a $C_{20}H_{42}$ molekulára cc-pVDZ bázisban, ill. a $C_{10}H_{22}$ molekulára cc-pVTZ bázisban volt. A CPU idők arányát ezen számításokra a (4.3) táblázat tartalmazza. Ez alapján várhatóan nagyobb bázis használatával még jobban fog csökkenni a CPU idő.

4.3. táblázat. CPU idők aránya (régi/új)

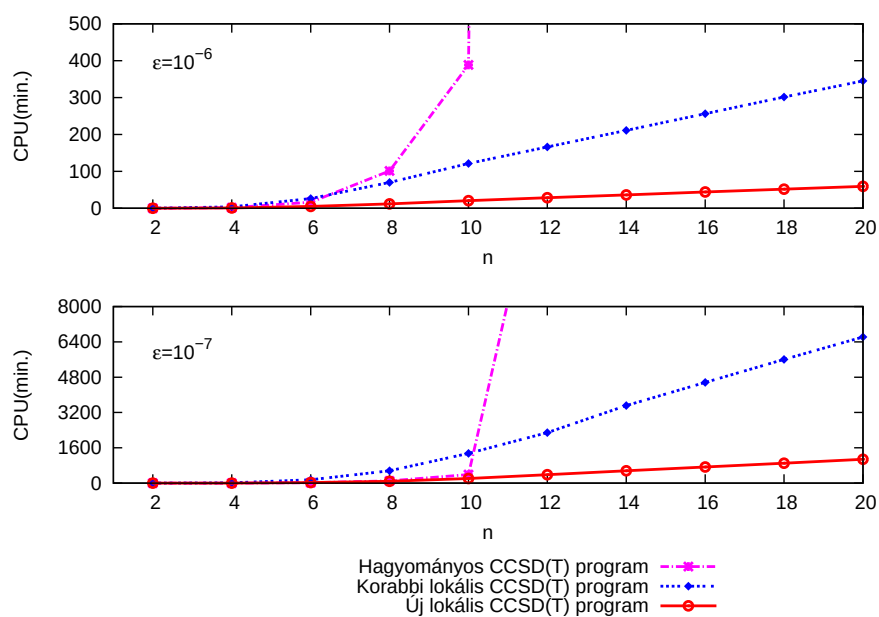
ε	$C_{20}H_{42}$		$C_{10}H_{22}$	
	10^{-6}	10^{-7}	10^{-6}	10^{-7}
CCSD	11	20	14	34
(T)	6	6	6	7



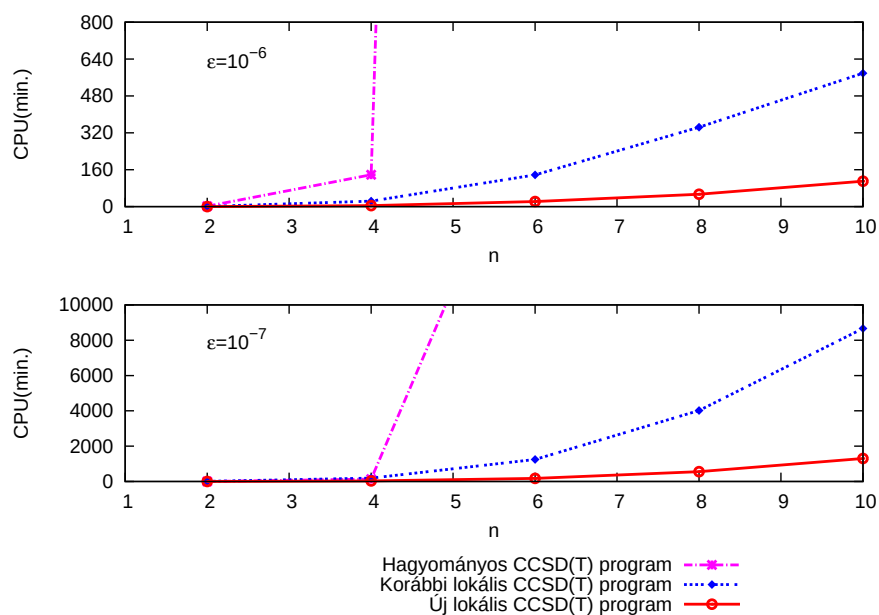
4.1. ábra. A lokális CCSD számítás skálázódása a korábbi lokális és a hagyományos CCSD számításokhoz képest cc-pVDZ bázisban



4.2. ábra. A lokális CCSD számítás skálázódása a korábbi lokális és a hagyományos CCSD számításokhoz képest cc-pVTZ bázisban



4.3. ábra. A lokális (T) korrekció számításának skálázódása a korábbi lokális és a hagyományos (T) korrekció számításokhoz képest cc-pVDZ bázisban



4.4. ábra. A lokális (T) korrekció számításának skálázódása a korábbi lokális és a hagyományos (T) korrekció számításokhoz képest cc-pVTZ bázisban

5. fejezet

Összefoglalás, konklúziók

Célunk a CCSD, ill. CCSD(T) számítások skálázódásának csökkentése volt azért, hogy nagy rendszerekre gyorsan és pontosan megkapjuk a CC korrelációs energiát. Ennek érdekében lokális közelítést alkalmaztunk, ahol kisebb rész-rendszerekre (doménekre) kell csak megoldanunk a CC egyenleteket. Ahhoz, hogy ezt hatékonyan tehessük meg egy olyan programot fejlesztettünk, ami optimálisan számolja a CCSD és CCSD(T) korrelációs energiát a feltételezett, legfeljebb 300 bázisfüggvény méretű doménen. Átírtuk a CC egyenleteket, hogy vektorizáltan, BLAS rutinok segítségével hatékonyan számolhassuk az intermediereket, és minimalizáltuk az I/O műveletek számát. A lokális számításokhoz új energiaképleteket vezettünk le és programoztunk be.

Tesztszámolásokat végeztünk hagyományos CCSD(T) és lokális CCSD(T) számításokra. A hagyományos számítások eredményei jól mutatják, hogy a program a tervezett doménméretre rendkívül gyors, a CFOUR-nál több mint kétszer gyorsabb minden esetben a CCSD módszer esetén és általában a (T) korrekció számításánál is. Az általam írt program valós futási idő / CPU idő aránya sokkal jobb, mint a CFOUR-é, azaz jóval kevesebb I/O műveletet végez, ami célunk volt a program írásakor. A lokális számításoknál a futási idők – a várakozásainknak megfelelően – drámaian csökkentek, ezzel lehetővé téve a korrelációs energia pontos és hatékony számítását akkora rendszerekre, amelyekre eddig még nem volt lehetőség.

A. Függelék

A (T) korrekció energiaképletének levezetése

Induljunk ki az eredeti (3.1) egyenletből. Bontsuk fel a szummákat a betöltött indexekre a

$$\begin{aligned} E_{(T)} = \frac{1}{3} \sum_{abc} \bigg[& \sum_{i \geq j \geq k} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \\ & + \sum_{i \geq k > j} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \\ & + \sum_{k > i \geq j} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \\ & + \sum_{j > i \geq k} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \\ & + \sum_{j \geq k > i} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \\ & + \sum_{k > j > i} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \bigg] \quad (A.1) \end{aligned}$$

módon. Hajtsuk végre az i, j, k indexek cseréjét az

$$\begin{aligned}
E_{(T)} = \frac{1}{3} \sum_{abc} \left[\sum_{i \geq j \geq k} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \right. \\
+ \sum_{i \geq j > k} (4W_{abc}^{ikj} + W_{bca}^{ikj} + W_{cab}^{ikj})(V_{abc}^{ikj} - V_{cba}^{ikj})/D_{abc}^{ikj} \\
+ \sum_{i > j \geq k} (4W_{abc}^{jki} + W_{bca}^{jki} + W_{cab}^{jki})(V_{abc}^{jki} - V_{cba}^{jki})/D_{abc}^{jki} \\
+ \sum_{i > j > k} (4W_{abc}^{jik} + W_{bca}^{jik} + W_{cab}^{jik})(V_{abc}^{jik} - V_{cba}^{jik})/D_{abc}^{jik} \\
+ \sum_{i \geq j > k} (4W_{abc}^{kij} + W_{bca}^{kij} + W_{cab}^{kij})(V_{abc}^{kij} - V_{cba}^{kij})/D_{abc}^{kij} \\
\left. + \sum_{i > j > k} (4W_{abc}^{kji} + W_{bca}^{kji} + W_{cab}^{kji})(V_{abc}^{kji} - V_{cba}^{kji})/D_{abc}^{kji} \right] \quad (A.2)
\end{aligned}$$

egyenlet szerint. Vegyük észre, hogy D_{abc}^{ijk} triviálisan invariáns az indexek bármilyen cseréjére. Használjuk ki a W_{abc}^{ijk} és a V_{abc}^{ijk} permutációs szimmetriáit úgy, hogy a betöltött indexekre mindenhol ijk legyen a sorrend

$$\begin{aligned}
E_{(T)} = \frac{1}{3} \sum_{abc} \left[\sum_{i \geq j, j \geq k} (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk})/D_{abc}^{ijk} \right. \\
+ \sum_{i \geq j > k} (4W_{acb}^{ijk} + W_{bac}^{ijk} + W_{cba}^{ijk})(V_{acb}^{ijk} - V_{cab}^{ijk})/D_{abc}^{ijk} \\
+ \sum_{i > j \geq k} (4W_{cab}^{ijk} + W_{abc}^{ijk} + W_{bca}^{ijk})(V_{cab}^{ijk} - V_{acb}^{ijk})/D_{abc}^{ijk} \\
+ \sum_{i > j > k} (4W_{bac}^{ijk} + W_{cba}^{ijk} + W_{acb}^{ijk})(V_{bac}^{ijk} - V_{bca}^{ijk})/D_{abc}^{ijk} \\
+ \sum_{i \geq j > k} (4W_{bca}^{ijk} + W_{cab}^{ijk} + W_{abc}^{ijk})(V_{bca}^{ijk} - V_{bac}^{ijk})/D_{abc}^{ijk} \\
\left. + \sum_{i > j > k} (4W_{cba}^{ijk} + W_{acb}^{ijk} + W_{bac}^{ijk})(V_{cba}^{ijk} - V_{abc}^{ijk})/D_{abc}^{ijk} \right], \quad (A.3)
\end{aligned}$$

így összevonhatjuk a szummákat a betöltött indexekre

$$\begin{aligned}
E_{(T)} = \frac{1}{3} \sum_{abc} \sum_{i \geq j \geq k} [& (4W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk})(V_{abc}^{ijk} - V_{cba}^{ijk}) \\
& + (4W_{acb}^{ijk} + W_{bac}^{ijk} + W_{cba}^{ijk})(V_{acb}^{ijk} - V_{cab}^{ijk}) \\
& + (4W_{cab}^{ijk} + W_{abc}^{ijk} + W_{bca}^{ijk})(V_{cab}^{ijk} - V_{acb}^{ijk}) \\
& + (4W_{bac}^{ijk} + W_{cba}^{ijk} + W_{acb}^{ijk})(V_{bac}^{ijk} - V_{bca}^{ijk}) \\
& + (4W_{bca}^{ijk} + W_{cab}^{ijk} + W_{abc}^{ijk})(V_{bca}^{ijk} - V_{bac}^{ijk}) \\
& + (4W_{cba}^{ijk} + W_{acb}^{ijk} + W_{bac}^{ijk})(V_{cba}^{ijk} - V_{abc}^{ijk})] \\
& / (D_{abc}^{ijk}(1 + \delta_{ij} + \delta_{jk})). \tag{A.4}
\end{aligned}$$

Belátható, hogy az $(1 + \delta_{ij} + \delta_{jk})$ a nevezőben azért kell, mert az összevonás előtt nem mindenhol volt megengedve az egyenlőség az indexekre, ezért volt, ahol többször számoltuk valamelyik tagot. Ha csak az $i = j$ vagy a $j = k$, akkor minden tagot kétszer számolunk, ha az $i = j = k$, akkor minden tagot háromszor számolunk. Bontsuk fel a zárójeleket

$$\begin{aligned}
E_{(T)} = \frac{1}{3} \sum_{abc} \sum_{i \geq j \geq k} [& + 4W_{abc}^{ijk} V_{abc}^{ijk} + W_{bca}^{ijk} V_{abc}^{ijk} + W_{cab}^{ijk} V_{abc}^{ijk} \\
& - 4W_{abc}^{ijk} V_{cba}^{ijk} - W_{bca}^{ijk} V_{cba}^{ijk} - W_{cab}^{ijk} V_{cba}^{ijk} \\
& + 4W_{acb}^{ijk} V_{acb}^{ijk} + W_{bac}^{ijk} V_{acb}^{ijk} + W_{cba}^{ijk} V_{acb}^{ijk} \\
& - 4W_{acb}^{ijk} V_{cab}^{ijk} - W_{bac}^{ijk} V_{cab}^{ijk} - W_{cba}^{ijk} V_{cab}^{ijk} \\
& + 4W_{cab}^{ijk} V_{cab}^{ijk} + W_{abc}^{ijk} V_{cab}^{ijk} + W_{bca}^{ijk} V_{cab}^{ijk} \\
& - 4W_{cab}^{ijk} V_{acb}^{ijk} - W_{abc}^{ijk} V_{acb}^{ijk} - W_{bca}^{ijk} V_{acb}^{ijk} \\
& + 4W_{bac}^{ijk} V_{bac}^{ijk} + W_{cba}^{ijk} V_{bac}^{ijk} + W_{acb}^{ijk} V_{bac}^{ijk} \\
& - 4W_{bac}^{ijk} V_{bca}^{ijk} - W_{cba}^{ijk} V_{bca}^{ijk} - W_{acb}^{ijk} V_{bca}^{ijk} \\
& + 4W_{bca}^{ijk} V_{bca}^{ijk} + W_{cab}^{ijk} V_{bca}^{ijk} + W_{abc}^{ijk} V_{bca}^{ijk} \\
& - 4W_{bca}^{ijk} V_{bac}^{ijk} - W_{cab}^{ijk} V_{bac}^{ijk} - W_{abc}^{ijk} V_{bac}^{ijk} \\
& + 4W_{cba}^{ijk} V_{cba}^{ijk} + W_{acb}^{ijk} V_{cba}^{ijk} + W_{bac}^{ijk} V_{cba}^{ijk} \\
& - 4W_{cba}^{ijk} V_{abc}^{ijk} - W_{acb}^{ijk} V_{abc}^{ijk} - W_{bac}^{ijk} V_{abc}^{ijk}] \\
& / (D_{abc}^{ijk}(1 + \delta_{ijk})) \tag{A.5}
\end{aligned}$$

szerint, majd rendezzük át az

$$\begin{aligned}
E_{(T)} = \frac{1}{3} \sum_{abc} \sum_{i \geq j \geq k} [& +4W_{abc}^{ijk} V_{abc}^{ijk} + W_{abc}^{ijk} V_{cab}^{ijk} + W_{abc}^{ijk} V_{bca}^{ijk} \\
& -4W_{abc}^{ijk} V_{cba}^{ijk} - W_{abc}^{ijk} V_{acb}^{ijk} - W_{abc}^{ijk} V_{bac}^{ijk} \\
& +4W_{bca}^{ijk} V_{bca}^{ijk} + W_{bca}^{ijk} V_{abc}^{ijk} + W_{bca}^{ijk} V_{cab}^{ijk} \\
& -4W_{bca}^{ijk} V_{bac}^{ijk} - W_{bca}^{ijk} V_{cba}^{ijk} - W_{bca}^{ijk} V_{acb}^{ijk} \\
& +4W_{cab}^{ijk} V_{cab}^{ijk} + W_{cab}^{ijk} V_{abc}^{ijk} + W_{cab}^{ijk} V_{bca}^{ijk} \\
& -4W_{cab}^{ijk} V_{acb}^{ijk} - W_{cab}^{ijk} V_{cba}^{ijk} - W_{cab}^{ijk} V_{bac}^{ijk} \\
& +4W_{acb}^{ijk} V_{acb}^{ijk} + W_{acb}^{ijk} V_{bac}^{ijk} + W_{acb}^{ijk} V_{cba}^{ijk} \\
& -4W_{acb}^{ijk} V_{cab}^{ijk} - W_{acb}^{ijk} V_{bca}^{ijk} - W_{acb}^{ijk} V_{abc}^{ijk} \\
& +4W_{bac}^{ijk} V_{bac}^{ijk} + W_{bac}^{ijk} V_{acb}^{ijk} + W_{bac}^{ijk} V_{cba}^{ijk} \\
& -4W_{bac}^{ijk} V_{bca}^{ijk} - W_{bac}^{ijk} V_{cab}^{ijk} - W_{bac}^{ijk} V_{abc}^{ijk} \\
& +4W_{cba}^{ijk} V_{cba}^{ijk} + W_{cba}^{ijk} V_{bac}^{ijk} + W_{cba}^{ijk} V_{acb}^{ijk} \\
& -4W_{cba}^{ijk} V_{abc}^{ijk} - W_{cba}^{ijk} V_{bca}^{ijk} - W_{cba}^{ijk} V_{cab}^{ijk}] \\
& / (D_{abc}^{ijk} (1 + \delta_{ijk}))
\end{aligned} \tag{A.6}$$

szerint. Az

$$Y_{abc}^{ijk} = V_{abc}^{ijk} + V_{bca}^{ijk} + V_{cab}^{ijk}, \tag{A.7}$$

$$Z_{abc}^{ijk} = V_{acb}^{ijk} + V_{bac}^{ijk} + V_{cba}^{ijk}, \tag{A.8}$$

$$W1_{abc}^{ijk} = W_{abc}^{ijk} + W_{bca}^{ijk} + W_{cab}^{ijk}, \tag{A.9}$$

$$W2_{abc}^{ijk} = W_{acb}^{ijk} + W_{bac}^{ijk} + W_{cba}^{ijk} \tag{A.10}$$

és az

$$\begin{aligned}
X_{abc}^{ijk} = & W_{abc}^{ijk} V_{abc}^{ijk} + W_{bca}^{ijk} V_{bca}^{ijk} + W_{cab}^{ijk} V_{cab}^{ijk} \\
& + W_{acb}^{ijk} V_{acb}^{ijk} + W_{bac}^{ijk} V_{bac}^{ijk} + W_{cba}^{ijk} V_{cba}^{ijk},
\end{aligned} \tag{A.11}$$

$$\begin{aligned}
\bar{X}_{abc}^{ijk} = & W_{abc}^{ijk} V_{cba}^{ijk} + W_{bca}^{ijk} V_{bac}^{ijk} + W_{cab}^{ijk} V_{acb}^{ijk} \\
& + W_{acb}^{ijk} V_{cab}^{ijk} + W_{bac}^{ijk} V_{bca}^{ijk} + W_{cba}^{ijk} V_{abc}^{ijk}
\end{aligned} \tag{A.12}$$

mennyiségeket bevezetve, az

$$\begin{aligned}
E_{(T)} = \frac{1}{3} \sum_{abc} \sum_{i \geq j \geq k} [& 3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk} (Y_{abc}^{ijk} - Z_{abc}^{ijk}) \\
& + W2_{abc}^{ijk} (Z_{abc}^{ijk} - Y_{abc}^{ijk})] / (D_{abc}^{ijk} (1 + \delta_{ijk}))
\end{aligned} \tag{A.13}$$

képletet kapjuk. Látjuk, hogy X invariáns a virtuális indexek tetszőleges cseréjére. Ha kicserélünk két virtuális indexet Y -ban vagy Z -ben, akkor Z -t, ill. Y -t kapjuk. Ugyanez igaz $W1$ -re és $W2$ -re. Ez viszont azt jelenti, hogy ha a

$$W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk}) \quad (\text{A.14})$$

tagban kicsirélünk két tetszőleges virtuális indexet, a tag értéke nem változik. Most bontsuk fel a szummát a virtuális indexekre

$$\begin{aligned} E_{(T)} = \frac{1}{3} \sum_{i \geq j \geq k} \frac{1}{(1 + \delta_{ijk})} & \left[\sum_{a \geq b \geq c} 3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) \right. \\ & + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})/D_{abc}^{ijk} \\ & \sum_{a \geq c > b} 3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) \\ & + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})/D_{abc}^{ijk} \\ & \sum_{c > a \geq b} 3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) \\ & + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})/D_{abc}^{ijk} \\ & \sum_{b > a \geq c} 3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) \\ & + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})/D_{abc}^{ijk} \\ & \sum_{b \geq c > a} 3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) \\ & + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})/D_{abc}^{ijk} \\ & \sum_{c > b > a} 3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) \\ & \left. + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})/D_{abc}^{ijk} \right] \quad (\text{A.15}) \end{aligned}$$

szerint. Hajtsuk végre az a, b, c indexek cseréjét

$$\begin{aligned}
E_{(T)} = \frac{1}{3} \sum_{i \geq j \geq k} \frac{1}{(1 + \delta_{ijk})} & \left[\sum_{a \geq b \geq c} 3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) \right. \\
& + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})/D_{abc}^{ijk} \\
& \sum_{a \geq b > c} 3X_{acb}^{ijk} - 3\bar{X}_{acb}^{ijk} + W1_{acb}^{ijk}(Y_{acb}^{ijk} - Z_{acb}^{ijk}) \\
& + W2_{acb}^{ijk}(Z_{acb}^{ijk} - Y_{acb}^{ijk})/D_{abc}^{ijk} \\
& \sum_{a > b \geq c} 3X_{bca}^{ijk} - 3\bar{X}_{bca}^{ijk} + W1_{bca}^{ijk}(Y_{bca}^{ijk} - Z_{bca}^{ijk}) \\
& + W2_{bca}^{ijk}(Z_{bca}^{ijk} - Y_{bca}^{ijk})/D_{abc}^{ijk} \\
& \sum_{a > b > c} 3X_{bac}^{ijk} - 3\bar{X}_{bac}^{ijk} + W1_{bac}^{ijk}(Y_{bac}^{ijk} - Z_{bac}^{ijk}) \\
& + W2_{bac}^{ijk}(Z_{bac}^{ijk} - Y_{bac}^{ijk})/D_{abc}^{ijk} \\
& \sum_{a \geq b > c} 3X_{cab}^{ijk} - 3\bar{X}_{cab}^{ijk} + W1_{cab}^{ijk}(Y_{cab}^{ijk} - Z_{cab}^{ijk}) \\
& + W2_{cab}^{ijk}(Z_{cab}^{ijk} - Y_{cab}^{ijk})/D_{abc}^{ijk} \\
& \sum_{a > b > c} 3X_{cba}^{ijk} - 3\bar{X}_{cba}^{ijk} + W1_{cba}^{ijk}(Y_{cba}^{ijk} - Z_{cba}^{ijk}) \\
& \left. + W2_{cba}^{ijk}(Z_{cba}^{ijk} - Y_{cba}^{ijk})/D_{abc}^{ijk} \right] \quad (A.16)
\end{aligned}$$

szerint, majd összevonhatjuk a szummákat. Az első és az utolsó szummában az $\bar{X}$ tag megegyezik, úgy mint a második és harmadik szummában és a negyedik és ötödik szummában. Számoljuk ki ezeknek az összegét:

$$\begin{aligned}
& \bar{X}_{abc}^{ijk} + \bar{X}_{acb}^{ijk} + \bar{X}_{bca}^{ijk} + \bar{X}_{bac}^{ijk} + \bar{X}_{cab}^{ijk} + \bar{X}_{cba}^{ijk} = \\
& 2(W_{abc}^{ijk}V_{cba}^{ijk} + W_{bca}^{ijk}V_{bac}^{ijk} + W_{cab}^{ijk}V_{acb}^{ijk} \\
& + W_{acb}^{ijk}V_{cab}^{ijk} + W_{bac}^{ijk}V_{bca}^{ijk} + W_{cba}^{ijk}V_{abc}^{ijk} \\
& + W_{bac}^{ijk}V_{cab}^{ijk} + W_{acb}^{ijk}V_{abc}^{ijk} + W_{cba}^{ijk}V_{bca}^{ijk} \\
& + W_{bca}^{ijk}V_{cba}^{ijk} + W_{abc}^{ijk}V_{acb}^{ijk} + W_{cab}^{ijk}V_{bac}^{ijk} \\
& + W_{acb}^{ijk}V_{bca}^{ijk} + W_{cba}^{ijk}V_{cab}^{ijk} + W_{bac}^{ijk}V_{abc}^{ijk} \\
& + W_{abc}^{ijk}V_{bac}^{ijk} + W_{cab}^{ijk}V_{cba}^{ijk} + W_{bca}^{ijk}V_{acb}^{ijk}) = \\
& 2(W2_{abc}^{ijk}Z_{abc}^{ijk} + W1_{abc}^{ijk}Y_{abc}^{ijk}) \quad (A.17)
\end{aligned}$$

A fenti megfontolásokat is figyelembevéve Y -ról, Z -ről, $W1$ -ről és $W2$ -ről, a szummákat összevonva kapjuk az

$$E_{(T)} = \frac{1}{3} \sum_{i \geq j \geq k} \frac{1}{(1 + \delta_{ijk})} \sum_{a \geq b \geq c} [18X_{abc}^{ijk} - 6(W2_{abc}^{ijk}Z_{abc}^{ijk} + W1_{abc}^{ijk}Y_{abc}^{ijk}) + 6W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) + 6W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})] / [D_{abc}^{ijk}(1 + \delta_{ab} + \delta_{bc})] \quad (A.18)$$

képletet, amiből a (3.7) egyenlet következik. Itt is hasonlóan, mint a betöltött indexekre való szummánál, figyelembe kell vennünk, hogy ha csak az $a = b$ vagy $b = c$, akkor minden tagot kétszer számolunk, amikor $a = b = c$ mindent háromszor számolunk.

B. Függelék

A lokális (T) korrekció energiaképletéhez szükséges levezetés

Ha tehát kiindulunk a (3.1) képletből és ugyanúgy járunk el, mint amikor a betöltött indexekre róttuk ki a megkötéseket a szummákban, csak most a virtuális indexekre tesszük meg ugyanezt, akkor az (A.13) egyenletet kell csak módosítanunk a

$$E_{(T)} = \frac{1}{3} \sum_{ijk} \sum_{a \geq b \geq c} [3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})] / (D_{abc}^{ijk}(1 + \delta_{abc})) \quad (B.1)$$

módon, ahol az X , $\bar{X}$, Y , Z , $W1$, $W2$ -t a fentiekhez hasonlóan

$$X_{abc}^{ijk} = W_{abc}^{ijk}V_{abc}^{ijk} + W_{abc}^{kij}V_{abc}^{kij} + W_{abc}^{jki}V_{abc}^{jki} + W_{abc}^{ikj}V_{abc}^{ikj} + W_{abc}^{jik}V_{abc}^{jik} + W_{abc}^{kji}V_{abc}^{kji}, \quad (B.2)$$

$$\bar{X}_{abc}^{ijk} = W_{abc}^{ijk}V_{abc}^{kji} + W_{abc}^{jki}V_{abc}^{jik} + W_{abc}^{kij}V_{abc}^{ikj} + W_{abc}^{ikj}V_{abc}^{kij} + W_{abc}^{jik}V_{abc}^{jki} + W_{abc}^{kji}V_{abc}^{ikj}, \quad (B.3)$$

$$Y_{abc}^{ijk} = V_{abc}^{ijk} + V_{abc}^{kij} + V_{abc}^{jki}, \quad (B.4)$$

$$Z_{abc}^{ijk} = V_{abc}^{ikj} + V_{abc}^{jik} + V_{abc}^{kji}, \quad (B.5)$$

$$W1_{abc}^{ijk} = W_{abc}^{ijk} + W_{abc}^{kij} + W_{abc}^{jki}, \quad (B.6)$$

$$W2_{abc}^{ijk} = W_{abc}^{ikj} + W_{abc}^{jik} + W_{abc}^{kji} \quad (B.7)$$

módon definiáljuk. Ez jelképesen az $i \leftrightarrow a$, a $j \leftrightarrow b$ és a $k \leftrightarrow c$ indexcserét jelenti. Bontsuk fel a szummát a betöltött indexekre

$$E_{(T)} = \frac{1}{3} \sum_{a \geq b \geq c} \frac{1}{1 + \delta_{abc}} \left[\sum_{i \geq j} \sum_k [3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})]/D_{abc}^{ijk} \right. \\ \left. \sum_{i < j} \sum_k [3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})]/D_{abc}^{ijk} \right] \quad (B.8)$$

szerint most csak az i és a j indexre. Felcseréljük az i és a j indexeket a második összegben az

$$E_{(T)} = \frac{1}{3} \sum_{a \geq b \geq c} \frac{1}{1 + \delta_{abc}} \left[\sum_{i \geq j} \sum_k [3X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} + W1_{abc}^{ijk}(Y_{abc}^{ijk} - Z_{abc}^{ijk}) + W2_{abc}^{ijk}(Z_{abc}^{ijk} - Y_{abc}^{ijk})]/D_{abc}^{ijk} \right. \\ \left. \sum_{j < i} \sum_k [3X_{abc}^{jik} - 3\bar{X}_{abc}^{jik} + W1_{abc}^{jik}(Y_{abc}^{jik} - Z_{abc}^{jik}) + W2_{abc}^{jik}(Z_{abc}^{jik} - Y_{abc}^{jik})]/D_{abc}^{jik} \right] \quad (B.9)$$

egyenletnek megfelelően. Érvényesek az

$$X_{abc}^{jik} = X_{abc}^{ijk}, \quad (B.10)$$

$$Y_{abc}^{jik} = Z_{abc}^{ijk}, \quad (B.11)$$

$$Z_{abc}^{jik} = Y_{abc}^{ijk}, \quad (B.12)$$

$$W1_{abc}^{jik} = W2_{abc}^{ijk}, \quad (B.13)$$

$$W2_{abc}^{jik} = W1_{abc}^{ijk} \quad (B.14)$$

azonosságok, azaz ha $i = j$, akkor a $W1$ -et és $W2$ -et tartalmazó tagok eltűnnek és csak a

$$\begin{aligned}
\sum_{i=j} \sum_k (X_{abc}^{ijk} - \bar{X}_{abc}^{ijk}) &= \sum_{ik} (W_{abc}^{iik} V_{abc}^{iik} + W_{abc}^{kii} V_{abc}^{kii} + W_{abc}^{iki} V_{abc}^{iki} \\
&\quad + W_{abc}^{iki} V_{abc}^{iki} + W_{abc}^{iik} V_{abc}^{iik} + W_{abc}^{kii} V_{abc}^{kii} \\
&\quad - W_{abc}^{iik} V_{abc}^{kii} - W_{abc}^{iki} V_{abc}^{iik} - W_{abc}^{kii} V_{abc}^{iki} \\
&\quad - W_{abc}^{iki} V_{abc}^{kii} - W_{abc}^{iik} V_{abc}^{iki} - W_{abc}^{kii} V_{abc}^{iik}) \\
&= \sum_{ik} [W_{abc}^{iik} (2V_{abc}^{iik} - V_{abc}^{kii} - V_{abc}^{iki}) \\
&\quad + W_{abc}^{iki} (2V_{abc}^{iki} - V_{abc}^{kii} - V_{abc}^{iik}) \\
&\quad + W_{abc}^{kii} (2V_{abc}^{kii} - V_{abc}^{iki} - V_{abc}^{iik})] \\
&= \sum_{ik} [W_{abc}^{iik} (3V_{abc}^{iik} - \bar{Y}_{abc}^{ik}) \\
&\quad + W_{abc}^{iki} (3V_{abc}^{iki} - \bar{Y}_{abc}^{ik}) \\
&\quad + W_{abc}^{kii} (3V_{abc}^{kii} - \bar{Y}_{abc}^{ik})]
\end{aligned} \tag{B.15}$$

tagok maradnak meg, ahol bevezettük az

$$\bar{Y}_{abc}^{ik} = V_{abc}^{iik} + V_{abc}^{iki} + V_{abc}^{kii} \tag{B.16}$$

mennyiséget. Ezek alapján az

$$\begin{aligned}
E_{(T)} &= \frac{1}{3} \sum_{a \geq b \geq c} \frac{1}{1 + \delta_{abc}} \left[\sum_{i > j} \sum_k [6X_{abc}^{ijk} - 3\bar{X}_{abc}^{ijk} - 3\bar{X}_{abc}^{jik} \right. \\
&\quad \left. + 2(W1_{abc}^{ijk} - W2_{abc}^{ijk})(Y_{abc}^{ijk} - Z_{abc}^{ijk})] / D_{abc}^{ijk} \right. \\
&\quad \left. + \sum_{ik} [W_{abc}^{iik} (3V_{abc}^{iik} - \bar{Y}_{abc}^{ik}) + W_{abc}^{iki} (3V_{abc}^{iki} - \bar{Y}_{abc}^{ik}) \right. \\
&\quad \left. + W_{abc}^{kii} (3V_{abc}^{kii} - \bar{Y}_{abc}^{ik})] / D_{abc}^{iik} \right]
\end{aligned} \tag{B.17}$$

képlet adja meg a korrekciót. Elvégezve a behelyettesítéseket megkapjuk a (3.39) képletet.

C. Függelék

Molekulák geometriái

Az alábbi felsorolásban megadjuk az egyes molekulák geometriáit. Itt szerepelnek az egyes atommagok távolságai (kötéshossz), kötésszögei.

CH₄

A C-H kötések hossza 1.09 Å és a H-C-H kötésszögek 109.47122°.

Cl₂

A Cl-Cl kötéshossz 1.99984 Å.

F₂

A F-F kötéshossz 1.41112 Å.

H₂O₂

A H-O kötések hossza 0.96096 Å, az O-O kötések hossza 1.44952 Å, A H-O-O kötésszög 100°, A H-O-O atommagok által kifeszített sík és a H-O kötés által bezárt szög 112.44°.

HF

A H-F kötés hossza 0.91516 Å.

NH₃

A H-N kötések hossza 1.01014 Å, a H-N-H kötésszögek 106.36173°, a H-N-H atommagok által kifeszített sík és a H-N kötés által bezárt szög 113.09001°.

A lokális számításokhoz használt normál alkán molekulák geometriáját ún. Z-mátrix formátumban adjuk meg. Példaképpen bemutatjuk a C₄H₁₀ Z-mátrixát a (C.1) táblázatban. A Z-mátrixot a következőképpen értelmezzük.

1. Lerakjuk az első sorban található atommagot az origóba, ez most egy C atommag.

C.1. táblázat. A C_4H_{10} Z-mátrixa

C							
H	1	r_{CH}					
C	1	r_{CC}	2	α			
H	1	r_{CH}	2	α	3	δ_4	
C	1	r_{CC}	2	α	3	δ_5	
H	5	r_{CH}	1	α	2	δ_6	
C	5	r_{CC}	1	α	6	δ_4	
H	5	r_{CH}	1	α	6	δ_8	
H	7	r_{CH}	5	α	1	δ_6	
H	7	r_{CH}	5	α	9	δ_4	
H	7	r_{CH}	5	α	9	δ_8	
H	3	r_{CH}	1	α	2	δ_6	
H	3	r_{CH}	1	α	12	δ_4	
H	3	r_{CH}	1	α	12	δ_8	

2. A második sorban lévő atomot, ami most egy H atommag, lerakjuk az első sorban lévő atommagtól – ezt jelöli a második sor második eleme (itt 1) – r_{CH} távolságba.
3. A harmadik sort úgy értelmezzük, hogy az első elem a következő atommag típusát jelöli – ez ismét egy C atommag, a második (itt 1), hogy melyik atomtól vett távolságát adjuk meg a harmadik elemmel (itt r_{CC}), a negyedik elem megadja, hogy melyik sorban megadott kötéssel (itt 2) az ötödik elemmel megadott szöget – itt α – zárja be a jelenlegi sorban megadott kötés.
4. A negyedik sort (és a további sorokat) úgy értelmezzük, hogy
 - az első elem megadja a következő atommag típusát (itt H), jelöljük ezt most M_1 -gyel,
 - a második elem kijelöli, hogy melyik sorban lévő atommaghoz kapcsolódik (itt 1), jelöljük ezt most M_2 -vel,
 - a harmadik elem megadja, hogy ettől mekkora távolságra van (itt r_{CH}), azaz az M_1 - M_2 kötéshosszat,
 - a negyedik elem kijelöl egy előzőleg megadott atommagot (itt 2), jelöljük ezt most M_3 -mal,
 - az ötödik elem megadja az M_1 - M_2 - M_3 kötésszöget (itt α),

- a hatodik elem kijelöl egy másik, előzőleg megadott atommagot (itt 3), jelöljük ezt most M_4 -gyel,
- a hetedik elem megadja, hogy a jelenlegi sorban megadott kötés mekkora szöget zár be az M_2 - M_3 - M_4 atomok által kifeszített síkkal (itt δ_4).

Az összes itt használt normál alkánt ezzel a technikával írjuk le úgy, hogy a szén atommagok a (C.1) ábra szerint legyenek egy síkban. Az összes molekulánál azonosak az egyes távolságok és szögek:

$$\begin{aligned}
 r_{CH} &= 1.09\text{\AA}, \\
 r_{CC} &= 1.55\text{\AA}, \\
 \alpha &= 109.47122^\circ, \\
 \delta_4 &= 120^\circ, \\
 \delta_5 &= -120^\circ, \\
 \delta_6 &= 180^\circ, \\
 \delta_8 &= 240^\circ.
 \end{aligned}$$



C.1. ábra. A szén atommagok elhelyezkedése (minden csúcson és vonalvégen egy-egy)

Irodalomjegyzék

- [1] F. Coester and H. Kümmel, *Nuclear Physics* **17**, 477 (1960).
- [2] J. Čížek, *J. Chem. Phys.* **45**, 4256 (1966).
- [3] R. J. Bartlett, in *Modern Electronic Structure Theory*, edited by D. R. Yarkony, volume 2, World Scientific, Singapore, 1995. p. 1047.
- [4] J. Gauss, in *Encyclopedia of Computational Chemistry*, edited by P. R. Schleyer, W. L. Jorgensen, H. F. Schaefer III, P. R. Schreiner, and W. Thiel, Wiley, New York, 1998, p. 615.
- [5] T. D. Crawford and H. F. Schaefer III, *Rev. Comp. Chem.* **14**, 33 (1999).
- [6] G. D. Purvis III and R. J. Bartlett, *J. Chem. Phys.* **76**, 1910 (1982).
- [7] Y. S. Lee, S. A. Kucharski, and R. J. Bartlett, *J. Chem. Phys.* **81**, 5906 (1984).
- [8] J. Noga and R. J. Bartlett, *J. Chem. Phys.* **86**, 7041 (1987).
- [9] S. A. Kucharski and R. J. Bartlett, *Theor. Chim. Acta* **80**, 387 (1991).
- [10] S. A. Kucharski and R. J. Bartlett, *J. Chem. Phys.* **97**, 4282 (1992).
- [11] N. Oliphant and L. Adamowicz, *J. Chem. Phys.* **94**, 1229 (1991).
- [12] M. Musiał, S. A. Kucharski, and R. J. Bartlett, *Chem. Phys. Lett.* **320**, 542 (2000).
- [13] J. Paldus, J. Čížek, and I. Shavitt, *Phys. Rev. A* **5**, 50 (1972).
- [14] R. J. Bartlett, *Ann. Rev. Phys. Chem.* **32**, 359 (1981).
- [15] R. J. Bartlett, H. Sekino, and G. D. Purvis III, *Chem. Phys. Lett.* **98**, 66 (1983).
- [16] M. Urban, J. Noga, S. J. Cole, and R. J. Bartlett, *J. Chem. Phys.* **83**, 4041 (1985).

- [17] K. Raghavachari, G. W. Trucks, J. A. Pople, and M. Head-Gordon, *Chem. Phys. Lett.* **157**, 479 (1989).
- [18] J. Gauss, W. J. Lauderdale, J. F. Stanton, J. D. Watts, and R. J. Bartlett, *Chem. Phys. Lett.* **182**, 207 (1991).
- [19] G. E. Scuseria, *Chem. Phys. Lett.* **176**, 27 (1991).
- [20] J. D. Watts, J. Gauss, and R. J. Bartlett, *J. Chem. Phys.* **98**, 8718 (1993).
- [21] S. A. Kucharski and R. J. Bartlett, *J. Chem. Phys.* **108**, 5243 (1998).
- [22] S. A. Kucharski and R. J. Bartlett, *J. Chem. Phys.* **108**, 5255 (1998).
- [23] P. Pulay, *Chem. Phys. Lett.* **100**, 151 (1983).
- [24] S. F. Boys, *Rev. Mod. Phys.* **32**, 296 (1960).
- [25] J. Pipek and P. Mezey, *J. Chem. Phys.* **90**, 4916 (1989).
- [26] P. Pulay and S. Saebø, *Theor. Chim. Acta* **69**, 357 (1986).
- [27] S. Saebø and P. Pulay, *J. Chem. Phys.* **86**, 914 (1987).
- [28] P. Y. Ayala and E. Scuseria, *J. Chem. Phys.* **110**, 3660 (1999).
- [29] E. Scuseria and P. Y. Ayala, *J. Chem. Phys.* **111**, 8330 (1999).
- [30] P. E. Maslen and M. Head-Gordon, *Chem. Phys. Lett.* **283**, 102 (1998).
- [31] C. Hampel and H.-J. Werner, *J. Chem. Phys.* **104**, 6286 (1996).
- [32] M. Schütz, G. Hetzer, and H.-J. Werner, *J. Chem. Phys.* **111**, 5691 (1999).
- [33] M. Schütz, *J. Chem. Phys.* **113**, 9986 (2000).
- [34] M. Schütz, *J. Chem. Phys.* **116**, 8772 (2002).
- [35] W. Förner, J. Ladik, P. Otto, and J. Čížek, *Chem. Phys.* **97**, 251 (1985).
- [36] H. Stoll, *Phys. Rev. B* **46**, 6700 (1992).
- [37] K. Rościszewski, K. Doll, B. Paulus, P. Fulde, and H. Stoll, *Phys. Rev. B* **57**, 14667 (1998).
- [38] H. Stoll, *Mol. Phys.* **108**, 243 (2010).
- [39] W. Li and S. Li, *J. Chem. Phys.* **121**, 6649 (2004).
- [40] M. Kobayashi and H. Nakai, *J. Chem. Phys.* **129**, 044103 (2008).

- [41] M. Ziółkowski, B. Jansík, T. Kjærgaard, and P. Jørgensen, *J. Chem. Phys.* **133**, 014107 (2010).
- [42] N. Flocke and R. J. Bartlett, *J. Chem. Phys.* **121**, 10935 (2004).
- [43] J. P. Foster and F. Weinhold, *J. Am. Chem. Soc.* **102**, 7211 (2004).
- [44] S. Li, J. Ma, and Y. Jiang, *J. Comput. Chem.* **23**, 237 (2002).
- [45] W. Li, P. Piecuch, J. R. Gour, and S. Li, *J. Chem. Phys.* **131**, 114109 (2009).
- [46] F. Neese, F. Wennmohs, and A. Hansen, *J. Chem. Phys.* **130**, 114108 (2009).
- [47] F. Neese, A. Hansen, and D. G. Liakos, *J. Chem. Phys.* **131**, 064103 (2009).
- [48] R. J. Bartlett and M. Musiał, *Rev. Mod. Phys.* **79**, 291 (2007).
- [49] I. Shavitt and R. J. Bartlett, *Many-Body Methods in Chemistry and Physics*, Cambridge University Press, Cambridge, 2009.
- [50] C. Hampel, K. A. Peterson, and H.-J. Werner, *Chem. Phys. Lett.* **190**, 1 (1992).
- [51] J. F. Stanton, *Chem. Phys. Lett.* **281**, 130 (1997).
- [52] Z. Rolik and M. Kállay, *J. Chem. Phys.* **135**, 104111 (2011).
- [53] G. E. Scuseria, T. J. Lee, and H. F. Schaefer III, *Chem. Phys. Lett.* **130**, 236 (1986).
- [54] A. P. Rendell, T. J. Lee, and A. Komornicki, *Chem. Phys. Lett.* **178**, 462 (1991).
- [55] T. H. Dunning Jr., *J. Chem. Phys.* **90**, 1007 (1989).
- [56] CFOUR, a quantum chemical program package written by J. F. Stanton, J. Gauss, M. E. Harding, P. G. Szalay with contributions from A. A. Auer, R. J. Bartlett, U. Benedikt, C. Berger, D. E. Bernholdt, Y. J. Bomble, O. Christiansen, M. Heckert, O. Heun, C. Huber, T.-C. Jagau, D. Jonsson, J. Jusélius, K. Klein, W. J. Lauderdale, D. A. Matthews, T. Metzroth, D. P. O'Neill, D. R. Price, E. Prochnow, K. Ruud, F. Schiffmann, S. Stopkiewicz, J. Vázquez, F. Wang, J. D. Watts and the integral packages MOLECULE (J. Almlöf and P. R. Taylor), PROPS (P. R. Taylor), ABACUS (T. Helgaker, H. J. Aa. Jensen, P. Jørgensen, and J. Olsen), and ECP routines by A. V. Mitin and C. van Wüllen. For the current version, see <http://www.cfour.de>.