Table-driven implementation of the multi-reference perturbation theories at second order

Alexander A. Granovsky

Laboratory of Chemical Cybernetics, M.V. Lomonosov Moscow State University, Moscow, Russia

November 1, 2006

Canonical single-reference MP

- MP2:

$$E_{mp2} = \frac{1}{4} \sum_{ijab} \frac{|< ij \| ab >|^2}{\Delta_{ijab}}$$

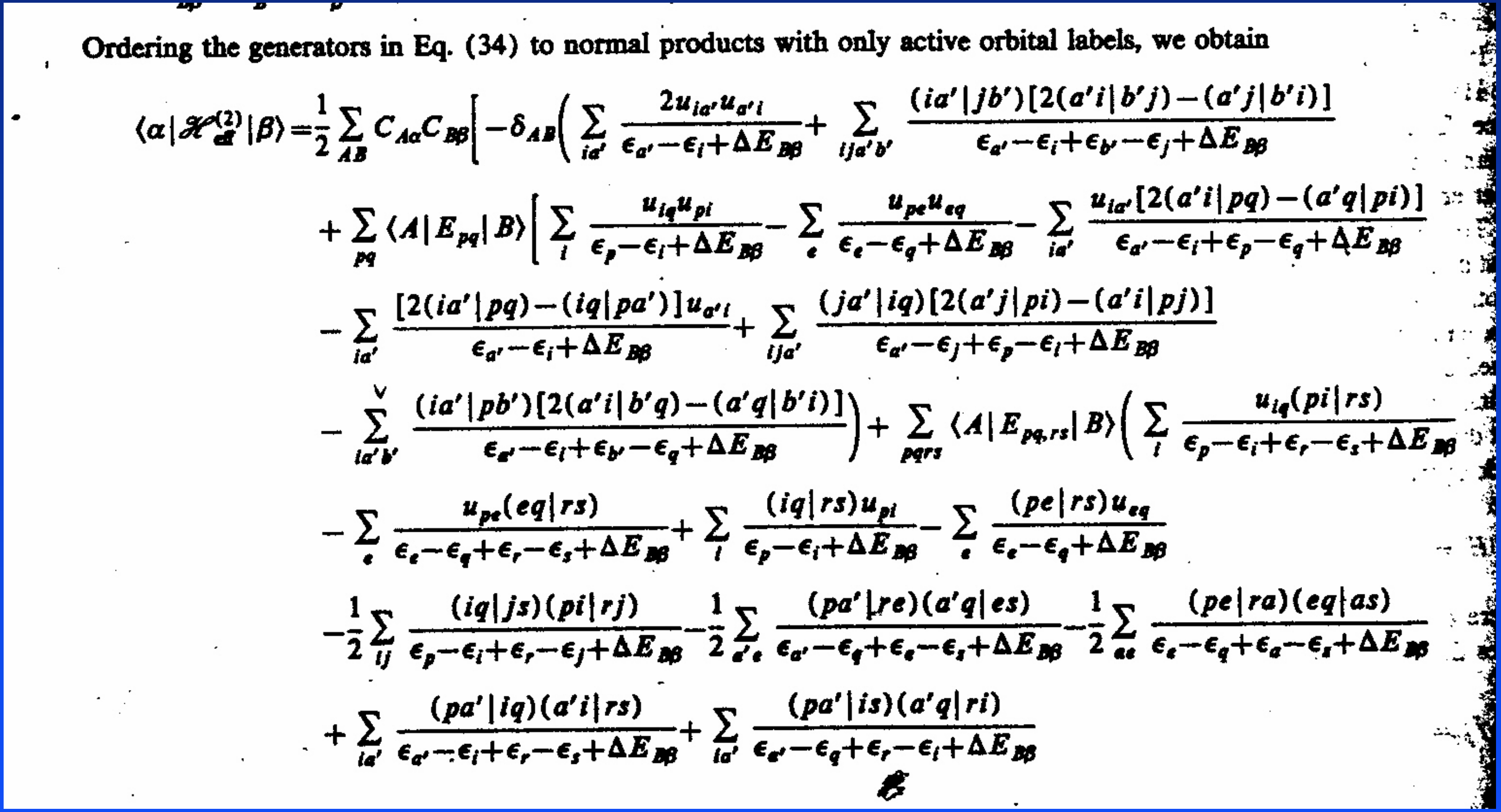
  - Parameters: N, N_{occ} *(i,j…)*, N_{virt}*(a,b…)*
  - Integral transformation - N^5 step
  - Only minor overhead due to PT power series summation itself (N^4 step)

- MP3 and above:
  - Integral transformation - N^5 step
  - Intermediate quantities (amplitudes entering into numerators of the individual terms of the PT series) calculations - N^6 and above
  - As in the case of MP2, PT summation itself has better scaling (e.g., N^4 for MP3)

Multi-reference (MR) MBPT theories

- Additional parameters:
  - $N_{act}(p,q,r,s,...)$, N_{det} (N_{csf}), N_{Heff}
- More complex expressions both for energy correction itself and for computational costs
  - Third and higher orders of various formulations of the multi-reference (MR) MBPT
    - Calculation of various intermediates is the most computationally-demanding stage
  - Non-contracted and partially contracted MR-MBPT theories at second order
    - Most of the computational efforts are typically due to summation of the individual terms of the PT series themselves, especially in the case of large active spaces

MCQDPT2 example



Ordering the generators in Eq. (34) to normal products with only active orbital labels, we obtain

$$
\begin{aligned}
\langle\alpha|\mathscr{H}_{\mathrm{eff}}^{(2)}|\beta\rangle = \frac{1}{2}\sum_{AB} C_{A\alpha}C_{B\beta}\Bigg[-\delta_{AB}\Bigg(\sum_{ia'} \frac{2u_{ia'}u_{a'i}}{\epsilon_{a'}-\epsilon_i+\Delta E_{B\beta}} + \sum_{ija'b'} \frac{(ia'|jb')[2(a'i|b'j)-(a'j|b'i)]}{\epsilon_{a'}-\epsilon_i+\epsilon_{b'}-\epsilon_j+\Delta E_{B\beta}} \\
+ \sum_{pq}\langle A|E_{pq}|B\rangle \Bigg[\sum_i \frac{u_{iq}u_{pi}}{\epsilon_p-\epsilon_i+\Delta E_{B\beta}} - \sum_e \frac{u_{pe}u_{eq}}{\epsilon_e-\epsilon_q+\Delta E_{B\beta}} - \sum_{ia'} \frac{u_{ia'}[2(a'i|pq)-(a'q|pi)]}{\epsilon_{a'}-\epsilon_i+\epsilon_p-\epsilon_q+\Delta E_{B\beta}} \\
- \sum_{ia'} \frac{[2(ia'|pq)-(iq|pa')]u_{a'i}}{\epsilon_{a'}-\epsilon_i+\Delta E_{B\beta}} + \sum_{ija'} \frac{(ja'|iq)[2(a'j|pi)-(a'i|pj)]}{\epsilon_{a'}-\epsilon_j+\epsilon_p-\epsilon_i+\Delta E_{B\beta}} \\
- \sum_{ia'b'}^{\vee} \frac{(ia'|pb')[2(a'i|b'q)-(a'q|b'i)]}{\epsilon_{a'}-\epsilon_i+\epsilon_{b'}-\epsilon_q+\Delta E_{B\beta}} \Bigg) + \sum_{pqrs}\langle A|E_{pq,rs}|B\rangle \Bigg(\sum_i \frac{u_{iq}(pi|rs)}{\epsilon_p-\epsilon_i+\epsilon_r-\epsilon_s+\Delta E_{B\beta}} \\
- \sum_e \frac{u_{pe}(eq|rs)}{\epsilon_e-\epsilon_q+\epsilon_r-\epsilon_s+\Delta E_{B\beta}} + \sum_i \frac{(iq|rs)u_{pi}}{\epsilon_p-\epsilon_i+\Delta E_{B\beta}} - \sum_e \frac{(pe|rs)u_{eq}}{\epsilon_e-\epsilon_q+\Delta E_{B\beta}} \\
-\frac{1}{2}\sum_{ij} \frac{(iq|js)(pi|rj)}{\epsilon_p-\epsilon_i+\epsilon_r-\epsilon_j+\Delta E_{B\beta}} - \frac{1}{2}\sum_{a'e} \frac{(pa'|re)(a'q|es)}{\epsilon_{a'}-\epsilon_q+\epsilon_e-\epsilon_s+\Delta E_{B\beta}} - \frac{1}{2}\sum_{ae} \frac{(pe|ra)(eq|as)}{\epsilon_e-\epsilon_q+\epsilon_a-\epsilon_s+\Delta E_{B\beta}} \\
+ \sum_{ia'} \frac{(pa'|iq)(a'i|rs)}{\epsilon_{a'}-\epsilon_i+\epsilon_r-\epsilon_s+\Delta E_{B\beta}} + \sum_{ia'} \frac{(pa'|is)(a'q|ri)}{\epsilon_{a'}-\epsilon_q+\epsilon_r-\epsilon_i+\Delta E_{B\beta}}
\end{aligned}
$$

Previous presentation goals

- Remove inefficient divide operations from inner loops
- Construct cache-friendly algorithm

Cache-friendly code sample

$$S = \sum_B C_{B\alpha} C_{B\beta} \sum_{ijab} \frac{(ia\,|\,jb)[2(ia\,|\,jb) - (ja\,|\,ib)]}{\varepsilon_a + \varepsilon_b - \varepsilon_i - \varepsilon_j + \Delta E_{B\beta}}$$

```
Loop over i
    Loop over j
        Loop over a
            Calculate $t_b = (ia|jb)[2(ia|jb) - (ja|ib)]$
            Loop over B
                Calculate $\Delta = \varepsilon_a - \varepsilon_i - \varepsilon_j + \Delta E_{B\beta}$
                Special sum over b: $W = W + \sum_b \frac{t_b}{\varepsilon_b + \Delta}$
                Accumulate
            End loop over B
        End loop over a
    End loop over j
End loop over i
```

Present work goals

- Further reduce computational costs
- Reduce algorithmic complexity

Formal analysis of the computational costs

- Each particular type of terms results in computational costs of a generic form:

$$N_{occ}^{k} \cdot N_{virt}^{l} \cdot N_{act}^{m+n} \cdot N_{csf} \cdot N_{Heff}$$

where k, l, m, n are some integers ≥ 0

- For the particular example above, *k*=2, *l*=2, *m*=*n*=0

Reminder - initial code version

$$S = \sum_B C_{B\alpha} C_{B\beta} \sum_{ijab} \frac{(ia \mid jb)[2(ia \mid jb) - (ja \mid ib)]}{\varepsilon_a + \varepsilon_b - \varepsilon_i - \varepsilon_j + \Delta E_{B\beta}}$$

- Loop over B
  - Loop over i
    - Loop over j
      - Loop over a
        - Sum over b: $$T = T + \sum_b \frac{(ia \mid jb)[2(ia \mid jb) - (ja \mid ib)]}{\varepsilon_a + \varepsilon_b - \varepsilon_i - \varepsilon_j + \Delta E_{B\beta}}$$
      - End loop over a
    - End loop over j
  - End loop over i
  - Accumulate S
- End loop over B

Code modification

$$S\left(\Delta E_{B\beta}\right) = \sum_{ijab} \frac{(ia\,|\,jb)[2(ia\,|\,jb)-(ja\,|\,ib)]}{\varepsilon_a+\varepsilon_b-\varepsilon_i-\varepsilon_j+\Delta E_{B\beta}}$$

$$S = \sum_{B} C_{B\alpha} C_{B\beta} S\left(\Delta E_{B\beta}\right)$$

```
Loop over B
Loop over i
    Loop over j
        Loop over a
            Sum over b:   $T = T + \sum_{b} \frac{(ia|jb)[2(ia|jb)-(ja|ib)]}{\varepsilon_a+\varepsilon_b-\varepsilon_i-\varepsilon_j+\Delta E_{B\beta}}$
        End loop over a
    End loop over j
End loop over i, accumulate   $S\left(\Delta E_{B\beta}\right)$
End loop over B
Loop over B: accumulate S:   $S = S + C_{B\alpha} C_{B\beta} S\left(\Delta E_{B\beta}\right)$
```

$\Delta E_{B\beta}$ expression

$$\Delta E_{B\beta} = E_B - E_\beta$$

$$E_B = E_{core} + \sum_{\substack{active\\orbitals}} n_i(B)\varepsilon_i$$

$$E_\beta = \sum_B \left|C_B^\beta\right|^2 E_B$$

Conclusion: $\Delta E_{B\beta}$ vary over **known** limited range of energies defined only by the active orbitals energy differences and number of electrons in active space.

Next steps, key ideas

- Let us approximate $S\left(\Delta E_{B\beta}\right)$ using table-driven interpolation
  - Introduce intermediate equally-spaced helper grid of Δ_λ , λ=1..N_{grid}
  - Calculate $S\left(\Delta_\lambda\right)$, λ=1..N_{grid} using the definition given above and previously described efficient approach
  - Fill in interpolation tables
  - Calculate contributions to S using interpolated values of $S\left(\Delta E_{B\beta}\right) \simeq S\left(\Delta E_{B\beta}, \textit{interpolation tables}\right)$

Resulting code

```
Loop over i
    Loop over j
        Loop over a
            Calculate $t_b = (ia|jb)[2(ia|jb) - (ja|ib)]$
            Loop over λ
                Calculate $\Delta = \varepsilon_a - \varepsilon_i - \varepsilon_j + \Delta_\lambda$
                Special sum over b: $W = W + \sum_b \frac{t_b}{\varepsilon_b + \Delta}$
                Accumulate $S(\Delta_\lambda)$
            End loop over λ
        End loop over a
    End loop over j
End loop over I
Fill in interpolation tables
Loop over B: accumulate S: $S = S + C_{B\alpha} C_{B\beta} \cdot Interp(\Delta E_{B\beta})$
```

Formal analysis of the computational costs, new code

- Original code:

$$costs = N_{occ}^{2} \cdot N_{virt}^{2} \cdot N_{csf} \cdot N_{Heff}$$

- New code:

$$costs = N_{occ}^{2} \cdot N_{virt}^{2} \cdot N_{grid} + C \cdot N_{csf} \cdot N_{Heff}$$

Formal analysis of the computational costs, generic case

- Main result:

$$N_{occ}^{k} \cdot N_{virt}^{l} \cdot N_{act}^{m+n} \cdot N_{csf} \cdot N_{Heff}$$

- is replaced by:

$$N_{occ}^{k} \cdot N_{virt}^{l} \cdot N_{act}^{m+n} \cdot N_{grid} + C \cdot N_{act}^{n} \cdot N_{csf} \cdot N_{Heff}$$

- n = 0 for zero-body, 2 for one-body, 4 for two-body, and 6 for three-body terms.

Conclusions

- N_{grid} does not depend on the number of CSF and is defined only by the desired precision and by the structure of the active space
  - Computational costs dependence on the number of CSF is now efficiently decoupled from the dependence on the number of orbitals
  - Improved algorithmic complexity
  - Computed energies are smooth functions of external parameters
  - No more need to store transformed integrals, only interpolation tables need to be computed
  - Much faster calculations for large systems!

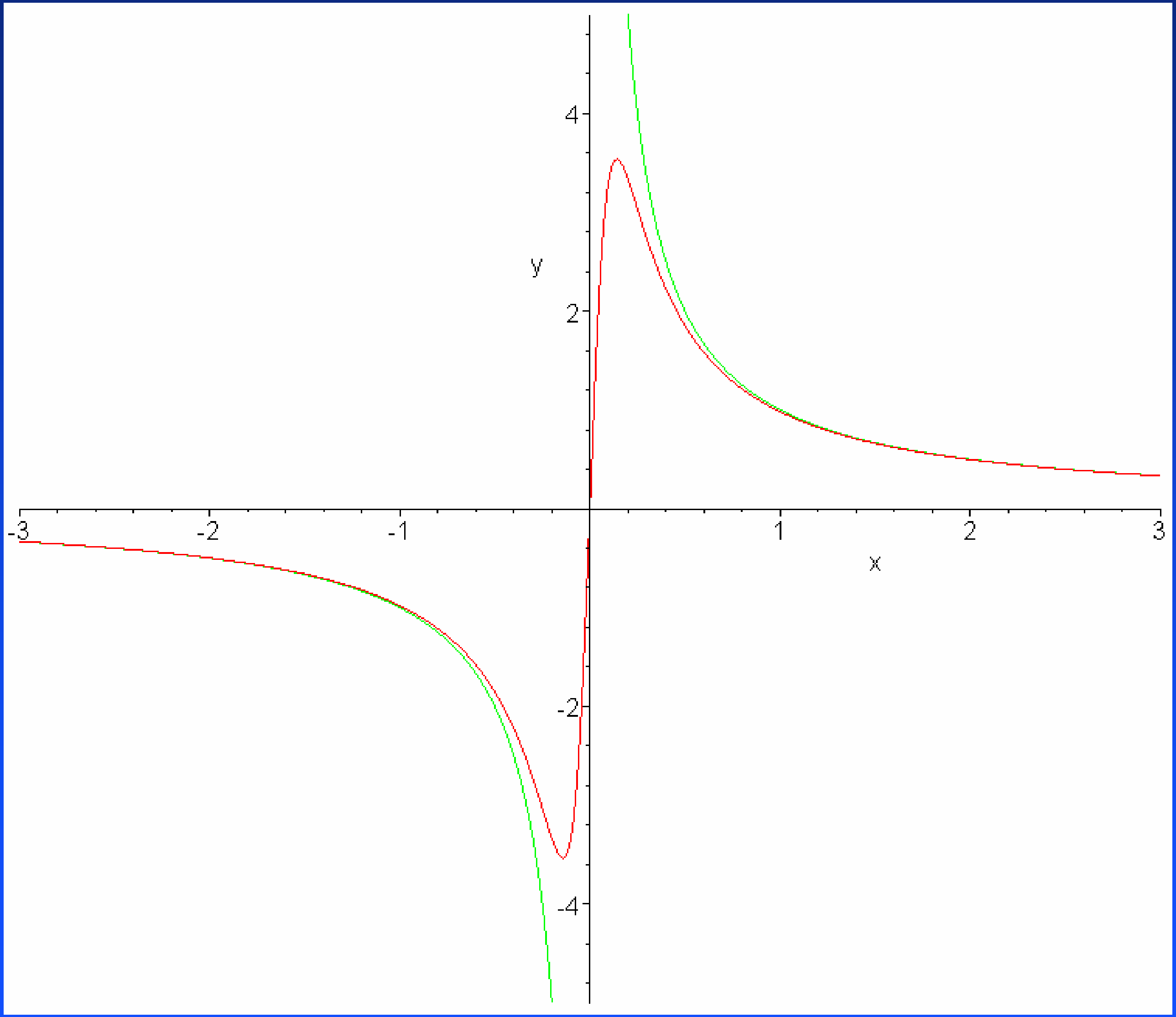
Main problem

- Q: How to interpolate $s(\Delta E_{B\beta})$ which can have multiple singularities?
- A: Actually, we always use ISA (Intruder State Avoidance) or some other energy denominators shift technique to avoid singularities. In the case of ISA, denominators are transformed as follows:

$$\frac{a}{b} \to \frac{a}{b+\frac{\alpha}{b}}$$

so that $s(\Delta E_{B\beta})$ is infinitely smooth function

Singularity removal by use of the ISA technique

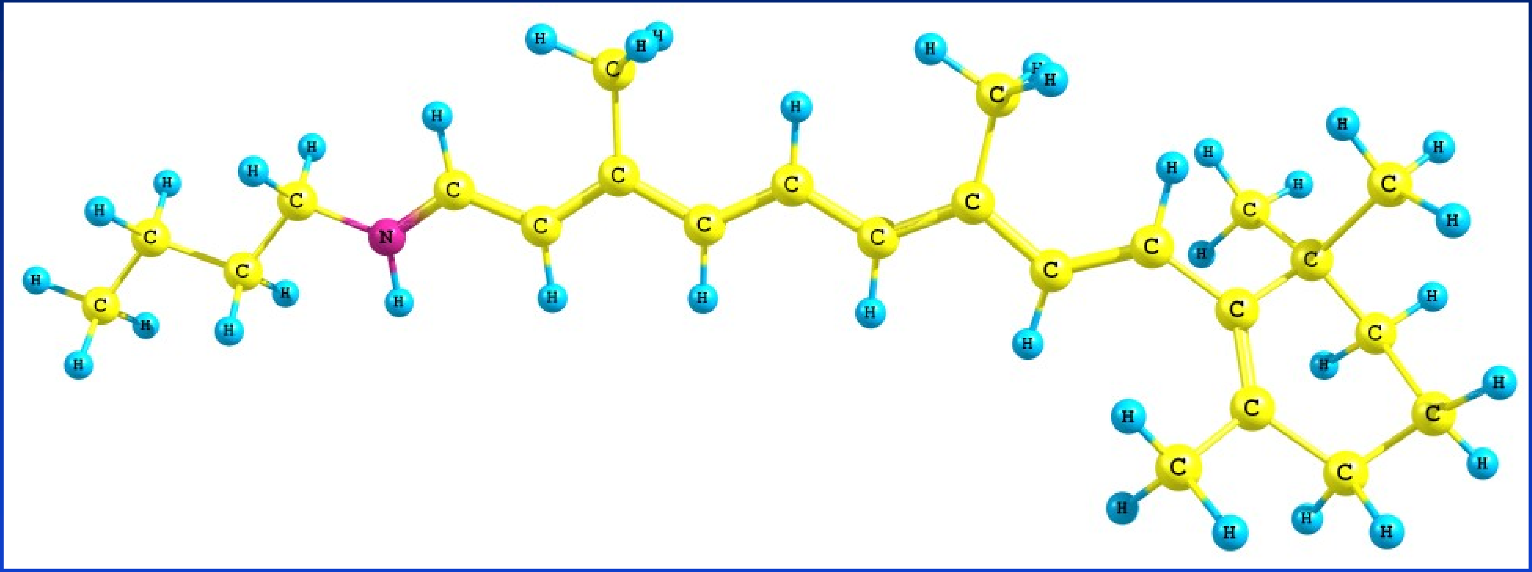


Practical example

Practical experience

- Seven-point polynomial interpolation seems to be optimal
- N_{grid} of ca. 200÷400 (ΔE of ca. 0.05 a.u.) seems to be enough to get cumulative absolute errors less than 10^{-8} a.u., which is for large problems in any case a way smaller than the round-off errors and errors introduced by the use of non-completely converged CASSCF orbitals.

Sample calculation



- Retinal molecule
- cc-pVTZ basis set, 1465 cartesian/1298 spherical basis functions
- CAS(12/12), 226512 CSF
- N_{Heff}=15

PC GAMESS, standard vs. table-driven approach

- January 2006
- Intel Xeon Dempsey 3.2 GHz
- CSF selection, 24709 CSF selected
- 95546 minutes of CPU time for PT
- E=-989.7676871075

- October 2006, pilot code (production code will be much faster)
- Intel Xeon Woodcrest 2.67 GHz
- No CSF selection, all 226512 are used
- 8650 minutes of CPU time for PT (5293 minutes with CSF selection)
- E=-989.7676871132

Thank you for your attention!