TCCM lectures – Advanced Computational Techniques

Peter Reinhardt

Laboratoire de Chimie Théorique, Sorbonne Université, 75252 Paris CEDEX 05,

Peter.Reinhardt@upmc.fr

Tuesday afternoon — I

Stability, precision, Fortran

- Language elements of FORTRAN
- Representing numbers in a computer
- Numerical precision and errors
- Numerical stability
- Interpolation
- Some examples

Language elements of FORTRAN

- FORTRAN = FORmula TRANslator
- FORTRAN IV $\rightarrow$ Fortran77 $\rightarrow$ F90 $\rightarrow$ ...
- free compilers available
- Simple structure, main program + subroutines + functions
- arguments passed via memory addresses
- no distinction of upper or lower case characters
- variable declaration recommended, but not mandatory
- array `a(n)` goes from `a(1)` to `a(n)`
- 2D array stored as `a(1,1), a(2,1), a(3,1)) ...`
- `do i=1,n ... end do`
- `if (condition) ... else if .... else ... end if`
- `do while (condition) ... end do`
- binary operators: `.and.`, `.or.`, `.eq.`, `.neq.`, `.gt.`, `.le.` etc

Language elements of FORTRAN

Input/Output

- file types: formatted, unformatted, direct-access
- read/write via `format` statements
- open/close of channels with numbers, 5 = stdin, 6 = stdout

```
open(unit=12,file='unit12',status='old',form='formatted')
read(12,*) i,j,k,(a(n),n=1,15)
close(12,status='delete')

open(unit=12,file='FILE12',status='unknown',form='unformatted')
write(12) i,j,k,(a(n),n=1,15)
close(12)

OPEN(UNIT=IUNIT1,FILE='DAFILE.TMP',STATUS='NEW',
-     ACCESS='DIRECT',RECL=NCDMX*8)
DO IA=1,NOCC
 DO IB=IA,NBAS
  IDREC=IDREC+1
  WRITE(IUNIT1,REC=IDREC) (H0(INDX0+NTRIA*(I-1)),I=1,NCDMX)
  INDX0=INDX0+1
 END DO
END DO
```

Language elements of FORTRAN

An example of F77 code

```
      PROGRAM MAIN
      IMPLICIT NONE
      COMMON /ABC/ ATABLE(10)
      REAL*8 ATABLE,PI,X,B,CALSIN
      INTEGER I

      PI=2.D0*ACOS(0.D0)
      DO I=1,10
       ATABLE=PI/DBLE(I)
      END DO
      CALL WSIN(ATABLE)
      CALL WWSIN
      X=ATABLE(2)
      B=CALSIN(X)
      WRITE(6,23) X,B
23    FORMAT(' X, SIN(X) = ',2E20.12)
      END

      SUBROUTINE WSIN(TT)
      IMPLICIT NONE
      REAL*8 TT(10)
      WRITE(6,*) ' WSIN ',TT(2),SIN(TT(2))
      RETURN
      END
```

```
      SUBROUTINE WWSIN
      IMPLICIT NONE
      COMMON /ABC/ TT(10)
      REAL*8 TT
      WRITE(6,9901) TT(2),SIN(TT(2))
9901  FORMAT(' WWSIN ',2F10.4)
      RETURN
      END

      REAL*8 FUNCTION CALSIN(X)
      IMPLICIT NONE
      REAL*8 X
      CALSIN=SIN(X)
      RETURN
      END
```

Language elements of FORTRAN

another example

```
   50 CONTINUE
      NEWBAS=0
      IF (NCOVAR.EQ.0) LPQ=0
      DO 1000 I=1,NSYM
      IF (NCOVAR.EQ.0) LPQ=LPQ+NBCON(I)*(NBCON(I)+1)/2
 1000 NEWBAS=NEWBAS+NBCON(I)
      DO 1002 I=1,NEWBAS
 1002 IF(NROW(I).GT.MXCC) MXCC=NROW(I)
      IF(ITRAN(1).NE.0) GO TO 5
      JJ=1
      IV=0
      DO 4 I=1,NEWBAS
      J=(I-1)*MXCC+1
      KMIN=NROW(I)
      DO 4 K=1,KMIN
      IV=IV+1
      INVC(IV)=I
      ITRAN(J)=JJ
      J=J+1
      JJ=JJ+1
    4 CONTINUE
      IF(NCOVAR.NE.0) GO TO 1003
    5 WRITE (6,2004)
 2004 FORMAT ('0',5X,'CONTRACTION COEFFICIENTS',/,' ')
      KMIN=0
      DO 1004 I=1,NEWBAS
      KMAX=KMIN+NROW(I)
      KMIN=KMIN+1
      WRITE (6,7) I,(ITRAN(K),CTRAN(K),K=KMIN,KMAX)
 1004 KMIN=KMIN+MXCC-1
    7 FORMAT(6X,'*',I2,'*',5X,7(I3,F12.7,2X),/,(10X,5X,7(I3,F12.7,2X)))
      WRITE (*,'(/)')
                                                                  811,7
```

Language elements of FORTRAN

Fortran 77

- first 5 columns for labels, 6th column for continuation character
- 1st column with `C`: comment
- line length 72 characters
- data sharing via `COMMON` blocks
- static memory layout
- `RETURN` statement in subroutines and functions

Language elements of FORTRAN

same example as F90 code

```
module abc
  implicit none
  real*8 :: atable(10)
contains

  subroutine fill_atable
    implicit none
    integer :: i
    real*8 :: pi
    pi=2.D0*acos(0.D0)
    do i=1,10
     atable(i)=pi/dble(i)
    end do
  end subroutine fill_atable
end module abc

program main
  use abc
  implicit none
  real*8 :: b,x,calsin
  call fill_atable
  call wsin(atable)
  call wwsin
  x=atable(2)
  b=calsin(x)
  write(6,23) x,b
  23 format(' x sin(x) = ',2e20.12)
end program main
```

```
subroutine wsin(tt)
  implicit none
  real*8 :: tt(10)
  write(6,*) ' wsin ',tt(2),sin(tt(2))
end subroutine wsin

subroutine wwsin
  use abc
  implicit none
  write(6,9901) atable(2),sin(atable(2))
  9901 format(' wwsin ',2F10.4)
end subroutine wwsin

real*8 function calsin(x)
  implicit none
  real*8 :: x
  calsin=sin(x)
end function calsin
```

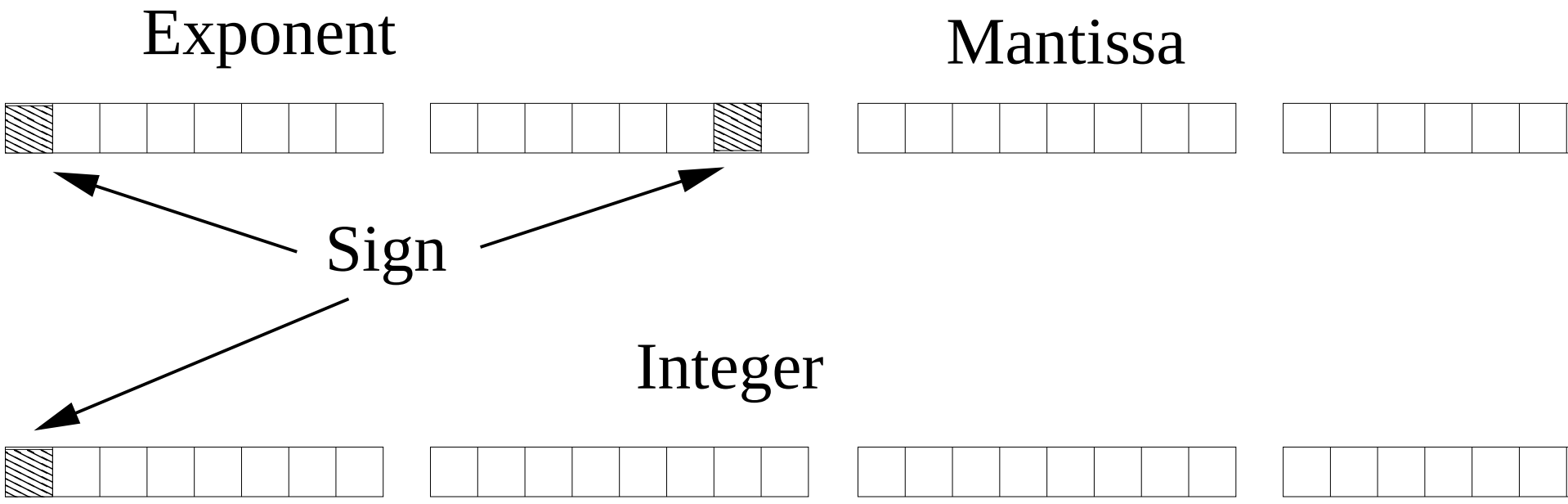
Language elements of FORTRAN

Fortran 90

- free format, continuation via & at the end of a line
- comments via `!` anywhere in a line (outside a string)
- data sharing via modules
- dynamic memory layout via `allocate`, `deallocate`
- no `RETURN` statement in subroutines and functions needed
- a first compilation produces the `.mod` files with the modules
- a second compilation uses the modules and produces the executable

Representing numbers in a computer

- A computer knows about 0 and 1 (bits)
- 8 bits are a byte, thus a number between `0000 0000` and `1111 1111` or in hexadecimal notation `00` to `FF`, or $255 = 16 * 16 - 1$.



Representing numbers in a computer

- Today a computer runs on 64-bit words — we can count from zero to $2^{64} - 1 = 2^4 \times \left(2^{10}\right)^6 - 1 \approx 16 \times 1000^6 \approx 1.6 \times 10^{19}$
- The sign is one bit less, or a factor of two: 10^{19}
- For a float we have to reserve digits for the sign, the sign of the exponent, the exponent, and the mantissa.
- IEEE = I-triple E standard: 1 bit for sign, 11 bits for exponent, 52 bits for mantissa = 16 decimal digits precision (`real*8` = double precision), numbers ranging from 10^{-1023} to 10^{+1023}.
- FORTRAN knows about
  - Single precision (`real`) = 4 bytes = 32 bit
  - Double precision (`real*8`) = 8 bytes = 64 bit
  - Quadruple precision (`real*16`) = 16 bytes = 128 bit
  - Integer, Integer*8
  - Complex, Complex*16
  - Logical, Character

Representing numbers in a computer

- CRAY T90 computers: 2 bits for signs, 48 bit for the mantissa, and 14 bit for the exponent
- Real numbers can range from 1.0×10^{-16384} to $1.0 \times 10^{+16384}$
- Double precision adds 64 bit to the mantissa
- Precision means what you may distinguish $\neq$ number range
- With a mantissa of 48 bits you can distinguish $2^{48} - 1 = 281474976710655$ from $2^{48} - 2 = 281474976710654$, i.e. a relative precision of 3.5×10^{-15}.

Why do we rarely talk about numerical errors in quantum chemistry?

- We are already happy to devise methods and write the code for these, i.e. calculate "correctly" a number
- We are chemists, not mathematicians
- Linear algebra and iterative solutions of equations means $\phi_k = \sum_i c_{ik}\varphi_i$ with MANY operations. Assigning a fixed error to every operation yields tremendous error bars.
- But these are worst-case scenarios
- Instead we can extrapolate to "Complete Basis Set" (CBS) limits.

Numerical precision and errors

- Every operation produces a round-off error
- $(1 + 10^{-15})(1 + 10^{-15}) = 1 + 2 \cdot 10^{-15} + 10^{-30}$
- Same for addition: $(1 + 10^{-15}) + (1 + 10^{-15}) = 2 + 2 \cdot 10^{-15}$
- 10 multiplications or additions $\rightarrow 10 \times 10^{-15} = 10^{-14}$ as precision.
- 10 000 multiplications or additions $\rightarrow 10000 \times 10^{-15} = 10^{-11}$ as precision.
- Loop over increments $r_n = n \cdot \Delta$: never increment via $r_n = r_{n-1} + \Delta$.
- Order of operations: $(-1 + 1) + \epsilon \neq -1 + (1 + \epsilon)$
- Try the powers of the "Golden Mean" $\phi = \frac{\sqrt{5}-1}{2}$. We may use two different ways, ϕ^n or the recursion $\phi^{n+1} = -\phi^n + \phi^{n-1}$. The latter goes astray beyond $n = 36$.

$$\phi^{n+1} = \left(\frac{\sqrt{5}-1}{2}\right)^2 \phi^{n-1} = \left(\frac{6-2\sqrt{5}}{4}\right) \phi^{n-1} = \phi^{n-1} - \phi^n$$

- Maybe useful to know: $0.5 \exp(3) \approx \pi^2 \approx 10$

Numerical precision and errors

Data carry errors:

- Conversion factors between units:
  - Ångström and atomic units
  - eV, kcal, Hartree
- Stored data:
  - Basis set libraries
  - Cartesian basis functions or spherical harmonics
  - Use of molecular symmetry
  - Definition of core orbitals
  - Pseudopotentials
- Cut-off thresholds
- Convergence limits

Stability

McWeeny purification scheme for (idempotent) density matrices

$$D_{n+1} = 3D_n^2 - 2D_n^3$$

as

$$\begin{aligned} 3(P+\epsilon)^2 - 2(P+\epsilon)^3 &= 3\epsilon^2 - 2\epsilon^3 + 6\epsilon P - 6\epsilon^2 P + 3P^2 - 6\epsilon P^2 - 2P^3 \\ &= 3\epsilon^2 - 2\epsilon^3 + 6\epsilon P - 6\epsilon^2 P + 3P - 6\epsilon P - 2P \\ &= P - 6\epsilon^2 P + 3\epsilon^2 - 2\epsilon^3 \end{aligned}$$

with $P^3 = P^2 = P$ and a small perturbation ϵ

- may be applied several times until convergence
- P is typically a projection matrix

3 iterations to

$$\begin{pmatrix} 1+\alpha & 0 & 0 \\ 0 & 1+\beta & 0 \\ 0 & 0 & \gamma \end{pmatrix} \longrightarrow \begin{pmatrix} 1+o(\alpha^8) & 0 & 0 \\ 0 & 1+o(\beta^8) & 0 \\ 0 & 0 & o(\gamma^8) \end{pmatrix}$$

Stability

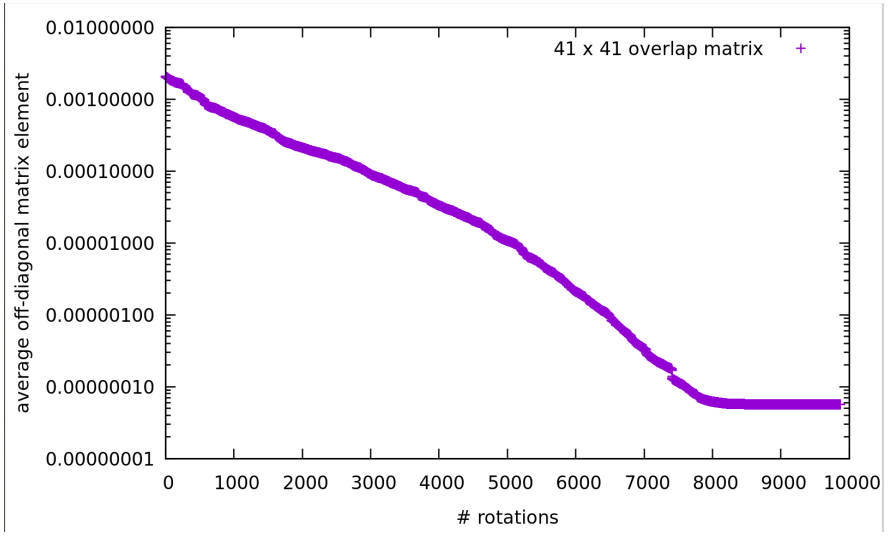
example: 1/2 of H_2O density matrix, eigenvalues

```
        23   2.1541300559071860E-016  -->    9.9318671246113933E-017
        24   5.0814749914346088E-016  -->    1.7354262068732642E-016
        25   9.7417453411357809E-016  -->    1.8597463649903488E-016
        26   1.6858498097574653E-015  -->    2.2024960136330725E-016
        27   2.7178916854748376E-015  -->    2.4129508929143867E-016
        28   3.1565818622118478E-015  -->    2.7148294596172379E-016
        29   3.4178123987942665E-015  -->    3.2385218542839834E-016
        30   4.5109609503031207E-015  -->    3.6023808219162995E-016
        31   5.8292291526280377E-015  -->    4.3348653584837588E-016
        32   8.6531937921240906E-015  -->    7.2090641701219215E-016
        33   1.2690033109608699E-014  -->    1.2389115675866279E-015
        34   1.6530983408775673E-014  -->    1.4108730269861222E-015
        35   8.8502215903146265E-014  -->    2.4093126048583025E-015
        36   1.0885367705979543E-013  -->    6.1964343448115541E-015
        37  0.9999999868536371        -->   0.9999999999999734
        38  0.9999999983316923        -->   0.9999999999999845
        39   1.0000000001859897       -->   0.9999999999999967
        40   1.0000000005100176       -->   0.9999999999999989
        41   1.0000000009090897       -->    1.0000000000000002
 average errors
  4.1371898568132271E-011  -->    2.2849664574668 46E-016
```

Stability

successive 2×2 rotations to diagonalize a matrix

- Iterative procedure starting from a set of orthogonal vectors
- $N(N-1)/2$ rotations to apply per sweep
- each rotation produces a zero off-diagonal element, but changes all elements of a row/column
- On average off-diagonal elements become smaller
- Approximate sin and cos by low-order expansions
- Upon convergence errors vanish
- Sequential rotations difficult to parallelize on many processors



Matrix-dressing techniques

Start with CI of Singles and Doubles

$$(\mathbf{H} - \mathbf{E_0}).\Psi = E_{\mathrm{corr}}..\Psi$$

- 1st row/column: $\langle \Phi_0 | \hat{H} | \Phi_I \rangle$, $\Phi_0 = \Phi_{\mathrm{HF}}$, Φ_I= di-excited determinant
- Add effects of higher excitations to the diagonal: $H_{ii} \rightarrow H_{ii} + \Delta_i$
- $\Delta_i = E_{\mathrm{corr}}$: CEPA–0 (Coupled electron pair approximation)
- $\Delta_i = E_{\mathrm{corr}}\,(1 - 2/n)$: ACPF (Averaged Coupled Pair Functional, n = number of correlated electrons)
- $\Delta_i = E_{\mathrm{corr}} - \sum_{J \in \mathrm{EPV}} \langle \Phi_I | \hat{H} | \Phi_{I+J} \rangle$: CEPA–2, CEPA–3, Full CEPA=(SC)^{2}CI
- As $\langle \Phi_I | \hat{H} | \Phi_{I+J} \rangle = \langle \Phi_0 | \hat{H} | \Phi_I \rangle$, no additional cost
- Restores size-consistence of the CISD equations

Matrix-dressing techniques

In practice:

- Start with an approximate CI vector $|\Psi\rangle$, e.g. MP2 solution
- Modify Hamiltonian with approximate E_{Corr} on the diagonal
- Find a better approximation to the eigenvector
- Modify Hamiltonian, etc, continue until convergence
- no need to diagonalize exactly the Hamiltonian, calculate dressing, rediagonalize exactly etc
- Solution for the dressed problem is found “on the fly”
- Inclusion of infinite summations of higher-order excitations

Matrix-dressing techniques

Coupled-Cluster equations

$$\langle \Phi_I \,|\, \hat{H} \left(1 + \hat{T}_2 + \frac{1}{2}\,\hat{T}_2^2 \right) \,|\, \Phi_0 \,\rangle = (E_{\rm HF} + E_{\rm corr})\, c_I$$

with $\hat{T}_2\hat{T}_2|\Phi_0\rangle = 2\sum_{klcd}(c_{ij}^{ab} * c_{kl}^{cd})\,|\Phi_{ijkl}^{abcd}\rangle$ and

$$\begin{aligned} c_{ij}^{ab} * c_{kl}^{cd} &= c_{ij}^{ab}\, c_{kl}^{cd} - \langle c_{ij}^{ab} * c_{kl}^{cd} \rangle \\ &= c_{ij}^{ab}\, c_{kl}^{cd} - c_{ik}^{ab}\, c_{jl}^{cd} + c_{il}^{ab}\, c_{jk}^{cd} - c_{ij}^{ac}\, c_{kl}^{bd} + c_{ik}^{ac}\, c_{jl}^{bd} - c_{il}^{ac}\, c_{jk}^{bd} \\ &\quad + c_{ij}^{ad}\, c_{kl}^{bc} - c_{ik}^{ad}\, c_{jl}^{bc} + c_{il}^{ad}\, c_{jk}^{bc} + c_{ij}^{cd}\, c_{kl}^{ab} - c_{ik}^{cd}\, c_{jl}^{ab} + c_{il}^{cd}\, c_{jk}^{ab} \\ &\quad - c_{ij}^{bd}\, c_{kl}^{ac} + c_{ik}^{bd}\, c_{jl}^{ac} - c_{il}^{bd}\, c_{jk}^{ac} + c_{ij}^{bc}\, c_{kl}^{ad} - c_{ik}^{bc}\, c_{jl}^{ad} + c_{il}^{bc}\, c_{jk}^{ad} \end{aligned}$$

- Rearrange as

$$(H_{0I} - \sum_J H_{0J}\,\langle c_I * c_J \rangle) + \langle \Phi_I | \hat{H} - E_{HF} | \Phi_I \rangle\, c_I + \sum_{J \neq I} H_{IJ}\, c_J = 0$$

- Dressing of the first column, scales as N^8 $(ijkl\,abcd)$

Some examples

Summing a polynomial

- naively: $P(x) = \sum_i a_i x^i$ or $P_i(x) = P_{i-1}(x) + a_i x^i$

```
P=a(0)
Xcumul = x
do i = 1,n
 P = P + a(i) * Xcumul
 Xcumul = Xcumul * x
end do
```

- savely (?) via recursion: $P_i(x) = x\, P_{i-1}(x) + a_{n-i}$

```
P = a(n)
D = 0.D0
do i = n-1, 0, -1
 P = x*P + a(i)
 D = x*D + (i+1)*a(i+1)
end do
```

Less operations, and terms of same order of magitude to handle

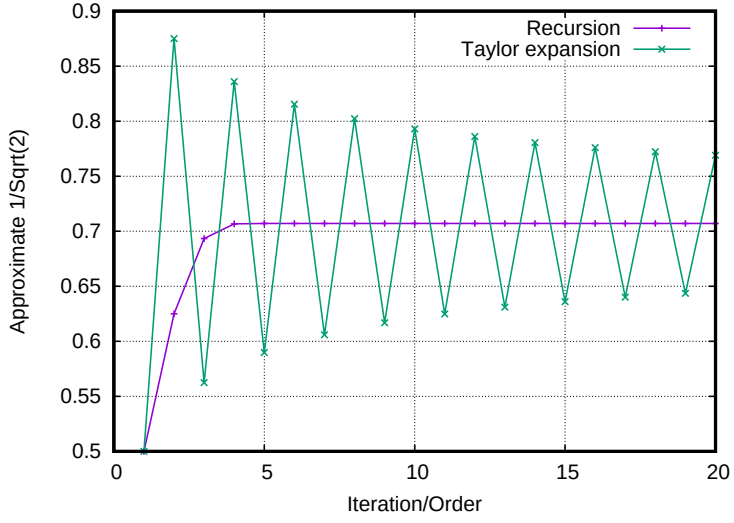
Some examples

Matrix operations: $S^{-1/2}$

- Taylor expansion converges slowly

$$S^{-1/2} = (1 + \underbrace{(S-1)}_{X})^{-1/2} \approx 1 - \frac{1}{2}X + \frac{3}{8}X^2 - \frac{5}{16}X^3 \pm \ldots$$

- Recursion $A_0 = 1$, $A_n = (3/2 - 1/2(A_{n-1}.S.A_{n-1}).A_{n-1})$ converges faster, but ...
- Convergence limited, $S = 2$, $X = 1$, $S_1 = 1/2$, $A_1 = 1/2$



Some more examples

Rayleigh-Schrödinger Perturbation theory for non-degenerate ground states:
Schrödinger equation $\mathbf{H}|\Psi\rangle = E_0\,|\Psi\rangle$:

$$\left(\hat{H}_0 + \lambda\hat{V}\right)\sum_{n=0}^{\infty}\lambda^n\,|\Psi^{(n)}\rangle = \sum_{m=0}^{\infty}\lambda^m\,E_0^{(m)}\sum_{k=0}^{\infty}\lambda^k\,|\Psi^{(k)}\rangle$$

Energies:

$$E_0^{(n)} \;=\; \langle\,0\,|\,V\,|\,n-1\,\rangle$$

Coefficients:

$$c_k^{(n)} \;=\; \langle\Phi_k|\Psi^{(n)}\rangle \;=\; \frac{1}{E_0^{(0)} \;-\; E_k^{(0)}}\,\big[\langle\,k\,|\,V\,|\,n-1\,\rangle$$

$$-E_0^{(1)}\,c_k^{(n-1)} \;-\; E_0^{(2)}\,c_k^{(n-2)} - \ldots - E_0^{(n-1)}c_k^{(1)}\Big]$$

Some more examples

Convergent or divergent? Calculate matrix elements via the Full-CI matrix elements $\langle\, I \,|\, \hat{H} \,|\, J \,\rangle - \langle\, I \,|\, \hat{H}_0 \,|\, J \,\rangle$, and denominators $E_0^{(0)} - E_k^{(0)}$

- Slater-Condon rules
- 1-electron, 2-electron integrals
- Diagonal elements of the Fock matrix
- Calculate energy and coefficients order by order, in alternance
- One may compare to results of Wigner's $2n+1$ rule

$$E_0^{(2n+1)} = \langle\, \Psi^{(2n)} \,|\, \hat{V} \,|\, 0 \,\rangle = \langle\, \Psi^{(n)} \,|\, \hat{V} \,|\, \Psi^{(n)} \,\rangle + \text{lower-order terms}$$

$$E_0^{(3)} = \langle\, \Psi^{(1)} \,|\, \hat{V} \,|\, \Psi^{(1)} \,\rangle$$

Some more examples

Example Be atom in a near-minimal Slater basis, MP series, 2 electrons correlated

```
0 -14.163174203154 -14.163174203154
1 -0.382190298908 -14.545364502062 (HF = -14.545364502062)
2 -0.020687496606 -14.566051998668
3 -0.012354937245 -14.578406935913
4 -0.006842089842 -14.585249025755
5 -0.003445406540 -14.588694432295
6 -0.001511429235 -14.590205861530
7 -0.000512011616 -14.590717873146
8 -0.000061980390 -14.590779853536
9 0.000095461250 -14.590684392286
10 0.000116894077 -14.590567498210
11 0.000088618926 -14.590478879284
12 0.000052254644 -14.590426624640
13 0.000023828191 -14.590402796449
14 0.000006508974 -14.590396287475
15 -0.000001779350 -14.590398066825
16 -0.000004360926 -14.590402427751
...
25 0.000000104127 -14.590410663845
```

Some more examples

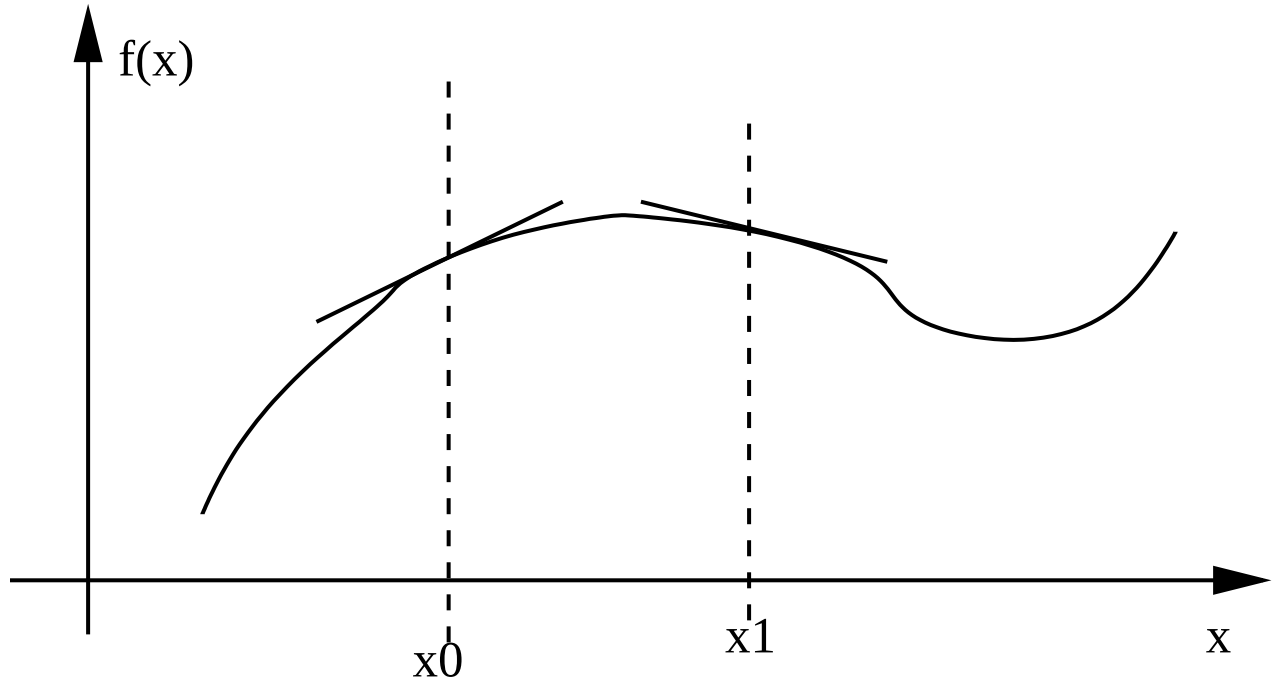
Change summation strategy: Shanks transformation

- Slowly convergent series: $S_N := \sum_{i=0}^{N} a_i$
- suppose $S_N \approx S + a\, b^N$ with 3 parameters, S, a and b
- take S_{N+1}, S_N, S_{N-1}, thus
  $b = (S_{N+1} - S)/(S_N - S) = (S_N - S)/(S_{N-1} - S)$
- $$S = \frac{S_{N+1}S_{N-1} - S_N^2}{S_{N+1} + S_{N-1} - 2S_N}$$

```
1 0.0000000000000000
2 -14.596725974242620
3 -14.593740861277285
4 -14.592189260977946
5 -14.591387063773329
6 -14.590980181805527
```

Interpolation

- function $f(x)$ and derivative $f'(x)$ known at grid-points x_i
- What is the best polynomial to interpolate between x_n and x_{n+1}?



Cubic polynomial $f(x) = a + bx + cx^2 + dx^3 = a + x(b + x(c + xd))$

- Suppose $x_n = 0$ and $x_{n+1} = 1 \rightarrow$ scale f' with $x_{n+1} - x_n$

Interpolation

System of equations

$$\begin{aligned} f(0) = f_0 &= a \\ f(1) = f_1 &= a + b + c + d \\ f'(0) = f'_0 &= b \\ f'(1) = f'_1 &= b + 2c + 3d \end{aligned}$$

with solution

$$a = f_0\ ,\ \ b = f'_0\ ,\ \ c = -3f_0 + 3f_1 - 2f'_0 - f'_1\ ,\ \ d = 2f_0 - 2f_1 + f'_0 + f'_1$$

Re-insert in the polynomial

$$f(x) = f_0(1 - 3x^2 + 2x^3) + f_1(3x^2 - 2x^3) + f'_0(x - 2x^2 + x^3) + f'_1(x^3 - x^2)$$

Interpolation

Interpolation scheme with tabulated $f(x)$ and $f'(x)$

1. is x within the limits of the table?
2. which are the two grid points with $x_n < x < x_{n+1}$?
3. get $f(x_n)$ and $f(x_{n+1})$, scale derivatives $f'(x_n)$ and $f'(x_{n+1})$
4. put data in a vector
5. determine relative coordinate $h = (x - x_n)/(x_{n+1} - x_n)$
6. evaluate the 4 polynomials at h: $u_1(h) = 1 - 3h^2 + 2h^3$, $u_2(h) = -1 - u_1(h)$, $u_3(h) = h(1 - 2h + h^2)$, $u_4(h) = h^2(h - 1)$
7. assemble the polynomial $f(x)$ as scalar product of the data and the vector of the polynomials $u_i(h)$