Combined-Multicore Parallelism for the UK electron-atom scattering Inner Region R-matrix codes on HECToR

Andrew Sunderland, Cliff Noble, Martin Plummer

Advanced Research Computing and Atomic and Molecular Physics Groups

STFC Daresbury Laboratory

Co-Investigators: HW van der Hart and MP Scott

The Queen’s University of Belfast

Summary

- Background to R-matrix approach
  - Electron atom collisions and other applications
  - R-matrix division of configuration space
- Background to PRMAT inner region codes
  - RMATRXII: PG Burke, VM Burke and KM Dunseath, J Phys B 27 (1994) 5341-5373: ‘BBD’)
  - Hamiltonian construction
- ANG, RAD and HAM
  - ANG developments
  - RAD developments
  - Xstream and filehand

Electron-Atom Collisions

Orion Nebula Mosaic HST • WFPC2

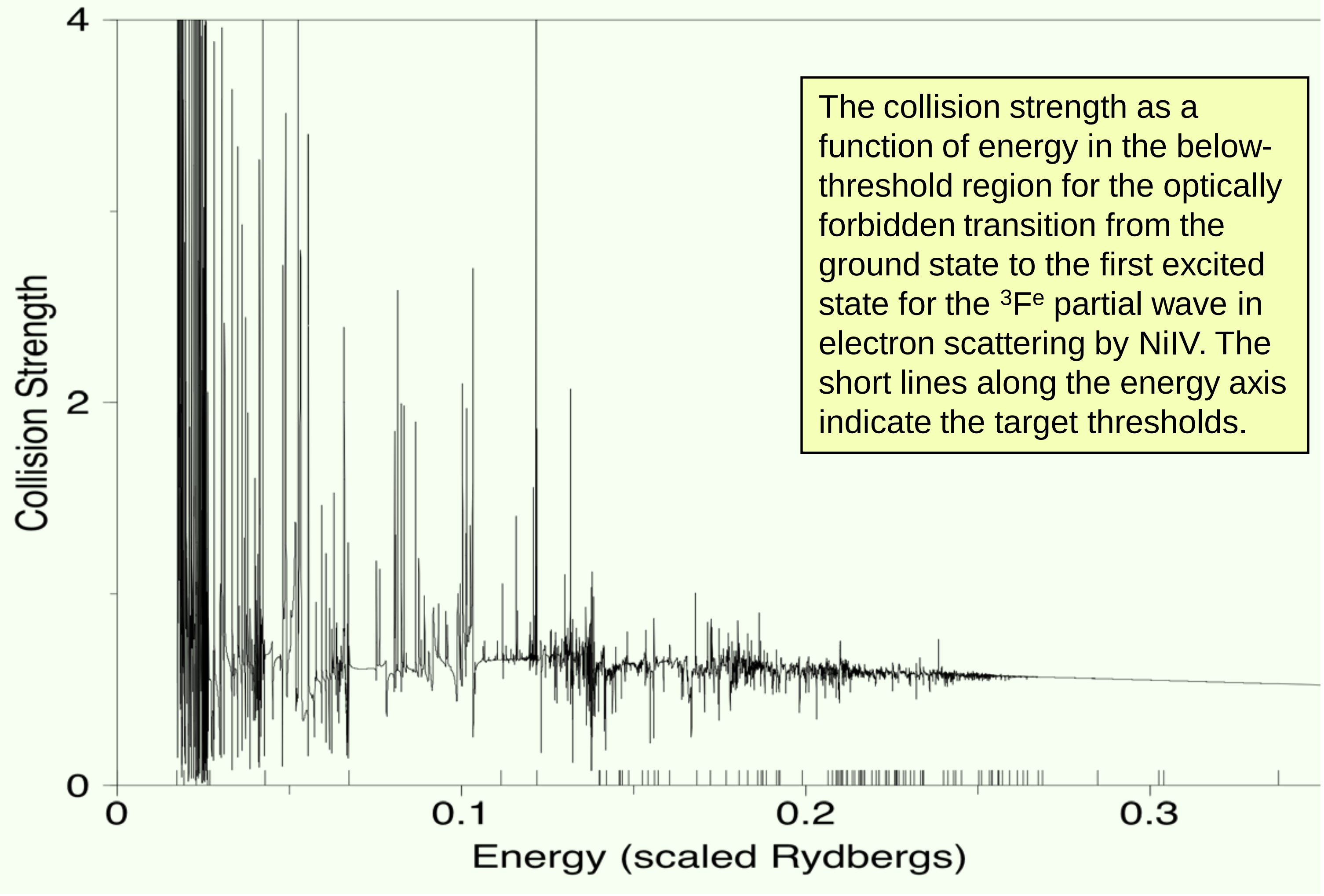
PRC95-45a • ST ScI OPO • November 20, 1995

C. R. O'Dell and S. K. Wong (Rice University), NASA

- Detailed electron-atom collision data is essential for understanding the behaviour of plasmas such as
  - Identifying forbidden lines such as those corresponding to the excitation of Ni^+ seen in observations of the Orion nebula (NGC 1976).
  - Plasma diagnostics of impurities in plasma fusion. JET, ITER: calculations of ionized tungsten underway.
  - Tin ions in next-generation nanolithography tools.

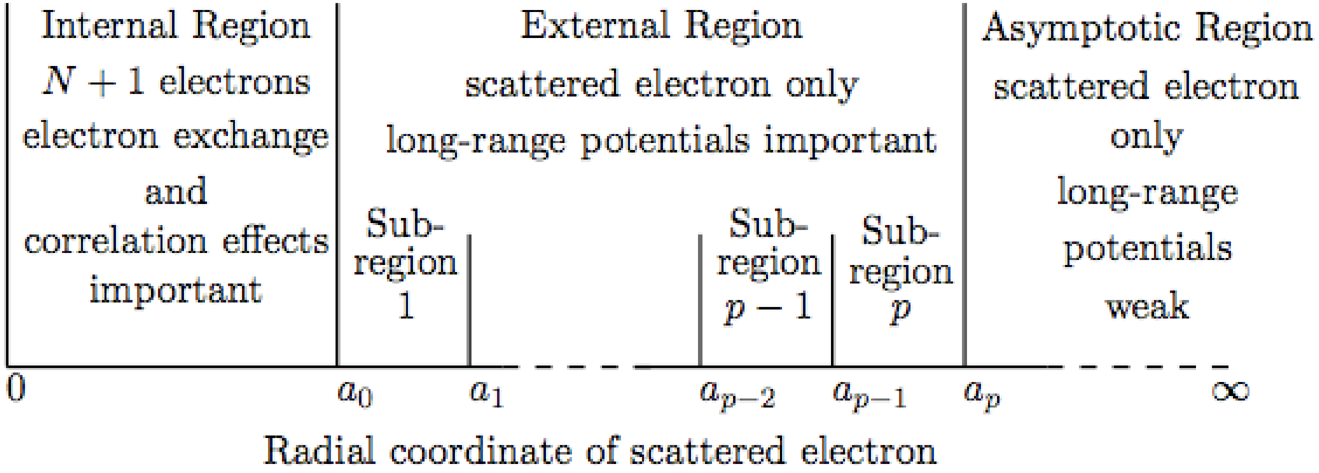
- R-matrix theory provides efficient computational methods for investigating electron-atom and electron-molecule collisions (cf talk by M Lysaght), also laser atom/molecule interactions (cf talk by L Moore)

Results: Collision Strengths



The collision strength as a function of energy in the below-threshold region for the optically forbidden transition from the ground state to the first excited state for the $^3F^e$ partial wave in electron scattering by NiIV. The short lines along the energy axis indicate the target thresholds.

Partition of Configuration Space



Internal Region

$N + 1$ electrons

electron exchange and correlation effects important

External Region

scattered electron only

long-range potentials important

Asymptotic Region

scattered electron only

long-range potentials weak

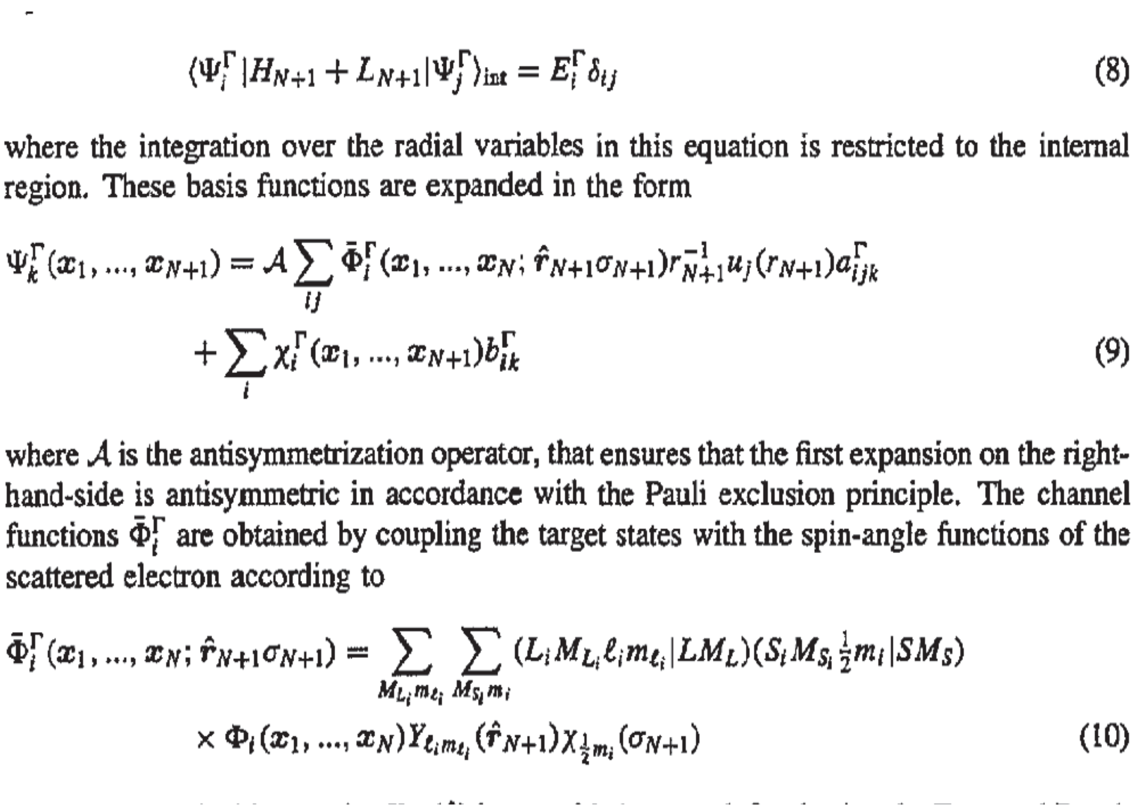
The parallelization of the overall code maps closely to this partitioning. A previous dCSE developed the outer region code PFARM: PFARM has since been very recently adapted for use by the molecular R-matrix codes UKRmol (work by UK-RAMP and DL core support)

In this dCSE we are developing/parallelizing the inner region codes. These are core codes which are needed by all the atomic R-matrix code-packages, for scattering and laser interactions (RMT, TDRM (RMF))

- Whereas PFARM was a high-scaling parallel code to begin with, the inner region codes are serial apart from an option for ScaLapack parallel diagonalization of the final Hamiltonian (in practice, a separate code: one of the aims of the current dCSE is to unify the diverse inner region sub-packages into a coherent ‘guaranteed’ package, in addition to parallelizing the Hamiltonian (and dipole matrix) construction.
- To understand the complexity of the codes, we quote liberally from ‘BBD’.

R-matrix theory (atomic)

We look for solutions of the TISE inside the sphere: we construct a Hamiltonian and diagonalize it. The basis functions include bound orbitals and continuum orbitals (non-zero on the boundary):



$$\langle \Psi_i^\Gamma | H_{N+1} + L_{N+1} | \Psi_j^\Gamma \rangle_{\rm int} = E_i^\Gamma \delta_{ij} \tag{8}$$

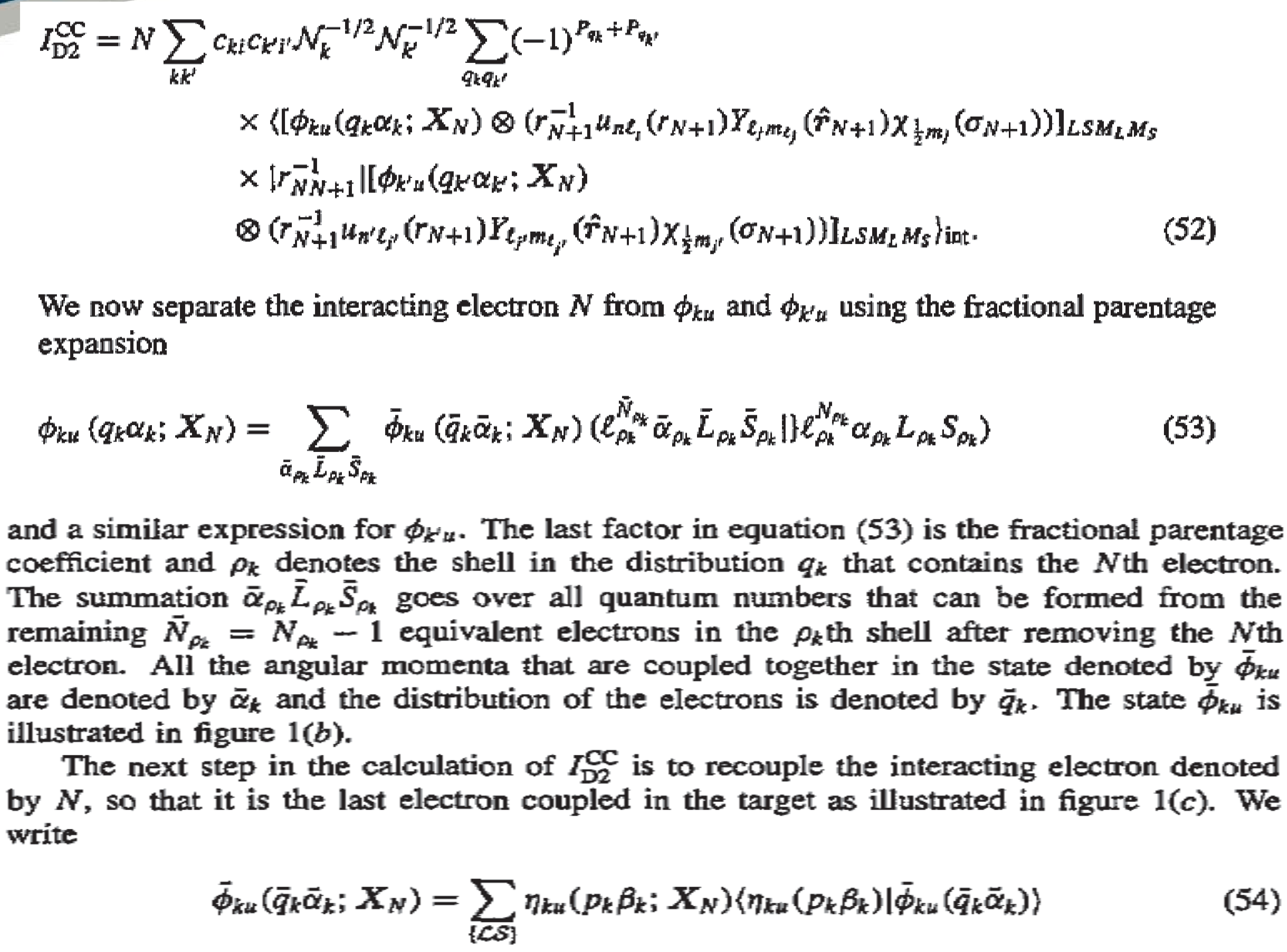
where the integration over the radial variables in this equation is restricted to the internal region. These basis functions are expanded in the form

$$\Psi_k^\Gamma(x_1, ..., x_{N+1}) = \mathcal{A} \sum_{ij} \bar{\Phi}_i^\Gamma(x_1, ..., x_N; \hat{r}_{N+1}\sigma_{N+1}) r_{N+1}^{-1} u_j(r_{N+1}) a_{ijk}^\Gamma + \sum_i \chi_i^\Gamma(x_1, ..., x_{N+1}) b_{ik}^\Gamma \tag{9}$$

where $\mathcal{A}$ is the antisymmetrization operator, that ensures that the first expansion on the right-hand-side is antisymmetric in accordance with the Pauli exclusion principle. The channel functions $\bar{\Phi}_i^\Gamma$ are obtained by coupling the target states with the spin-angle functions of the scattered electron according to

$$\bar{\Phi}_i^\Gamma(x_1, ..., x_N; \hat{r}_{N+1}\sigma_{N+1}) = \sum_{M_{L_i} m_{\ell_i}} \sum_{M_{S_i} m_i} (L_i M_{L_i} \ell_i m_{\ell_i} | L M_L)(S_i M_{S_i} \tfrac{1}{2} m_i | S M_S) \times \Phi_i(x_1, ..., x_N) Y_{\ell_i m_{\ell_i}}(\hat{r}_{N+1}) \chi_{\frac{1}{2} m_i}(\sigma_{N+1}) \tag{10}$$

- Note the two expansions, the second expansion appears because the continuum orbitals in the first expansion are orthogonal to the bound orbitals in the N-electron target states.
    - The N-electron target states are themselves full ‘CI’ expansions
- The Hamiltonian matrix consists of three distinct types, ‘CC’, ‘BC’ and ‘BB’
- Components are made up from 1-electron integrals and 2-electron integrals (1 / r_ij), by ‘Slater’s rules’. The 2-electron integrals have ‘direct’ and ‘exchange’ components.
- Because of the spherical symmetry, extensive use is made of spherical tensor theory.
    - (representations of the 3d rotation-inversion group combined with antisymmetry, transformations of spherical harmonics and spinors)



$$I^{\mathrm{CC}}_{\mathrm{D2}} = N \sum_{kk'} c_{ki} c_{k'i'} \mathcal{N}_k^{-1/2} \mathcal{N}_{k'}^{-1/2} \sum_{q_k q_{k'}} (-1)^{P_{q_k} + P_{q_{k'}}}$$
$$\times \langle [\phi_{ku}(q_k \alpha_k; \boldsymbol{X}_N) \otimes (r^{-1}_{N+1} u_{n\ell_i}(r_{N+1}) Y_{\ell_i m_{\ell_i}}(\hat{r}_{N+1}) \chi_{\frac{1}{2} m_i}(\sigma_{N+1}))]_{LSM_L M_S}$$
$$\times | r^{-1}_{N N+1} | [\phi_{k'u}(q_{k'} \alpha_{k'}; \boldsymbol{X}_N)$$
$$\otimes (r^{-1}_{N+1} u_{n'\ell_{i'}}(r_{N+1}) Y_{\ell_{i'} m_{\ell_{i'}}}(\hat{r}_{N+1}) \chi_{\frac{1}{2} m_{i'}}(\sigma_{N+1}))]_{LSM_L M_S} \rangle_{\mathrm{int}}. \tag{52}$$

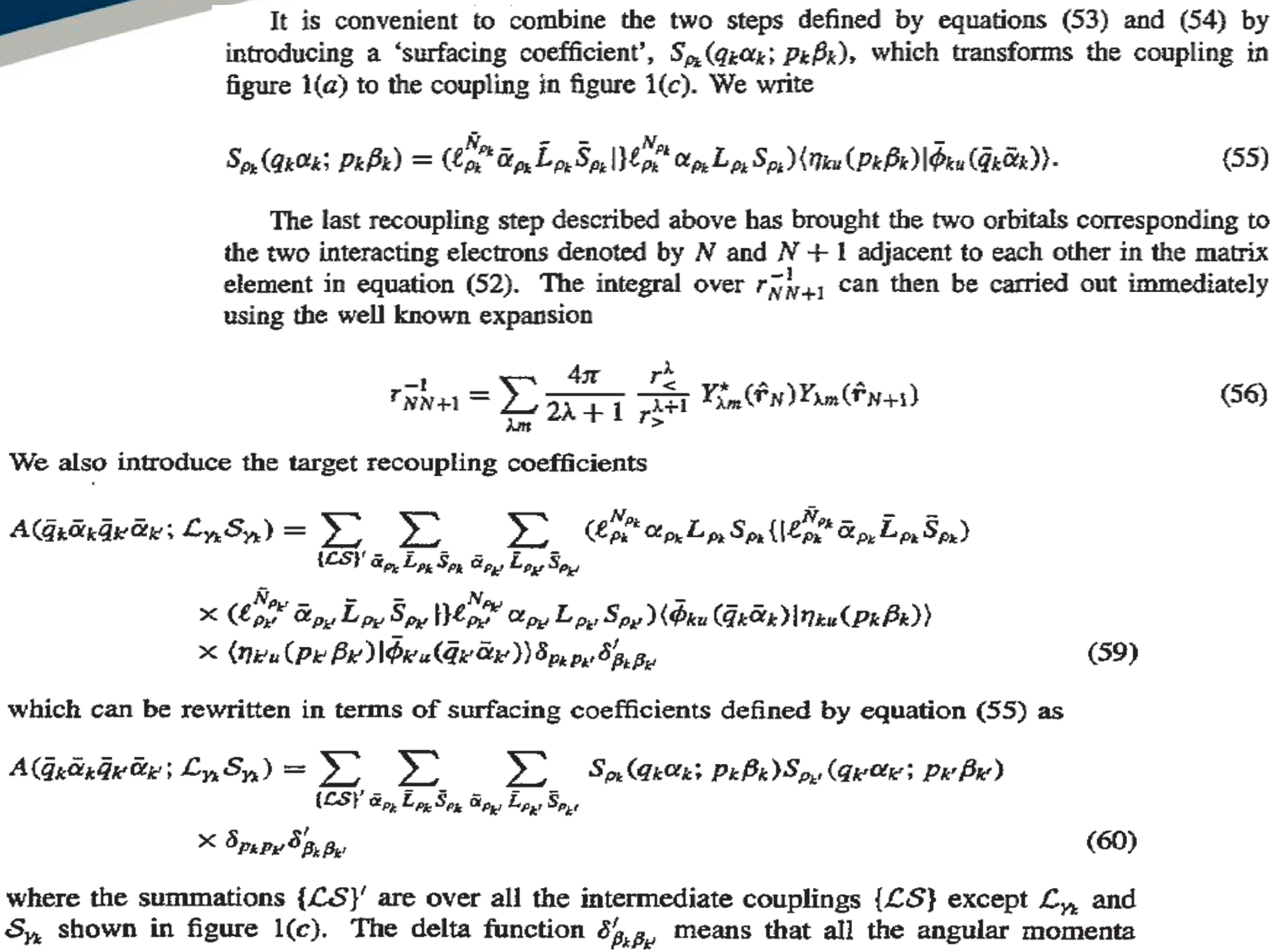
We now separate the interacting electron N from ϕ_{ku} and $\phi_{k'u}$ using the fractional parentage expansion

$$\phi_{ku}\,(q_k \alpha_k; \boldsymbol{X}_N) = \sum_{\bar{\alpha}_{\rho_k} \bar{L}_{\rho_k} \bar{S}_{\rho_k}} \bar{\phi}_{ku}\,(\bar{q}_k \bar{\alpha}_k; \boldsymbol{X}_N)\,(\ell_{\rho_k}^{\bar{N}_{\rho_k}} \bar{\alpha}_{\rho_k} \bar{L}_{\rho_k} \bar{S}_{\rho_k} | \} \ell_{\rho_k}^{N_{\rho_k}} \alpha_{\rho_k} L_{\rho_k} S_{\rho_k}) \tag{53}$$

and a similar expression for $\phi_{k'u}$. The last factor in equation (53) is the fractional parentage coefficient and ρ_k denotes the shell in the distribution q_k that contains the Nth electron. The summation $\bar{\alpha}_{\rho_k} \bar{L}_{\rho_k} \bar{S}_{\rho_k}$ goes over all quantum numbers that can be formed from the remaining $\bar{N}_{\rho_k} = N_{\rho_k} - 1$ equivalent electrons in the ρ_kth shell after removing the Nth electron. All the angular momenta that are coupled together in the state denoted by $\bar{\phi}_{ku}$ are denoted by $\bar{\alpha}_k$ and the distribution of the electrons is denoted by $\bar{q}_k$. The state $\bar{\phi}_{ku}$ is illustrated in figure 1(*b*).

The next step in the calculation of $I^{\mathrm{CC}}_{\mathrm{D2}}$ is to recouple the interacting electron denoted by N, so that it is the last electron coupled in the target as illustrated in figure 1(*c*). We write

$$\bar{\phi}_{ku}(\bar{q}_k \bar{\alpha}_k; \boldsymbol{X}_N) = \sum_{\{\mathcal{LS}\}} \eta_{ku}(p_k \beta_k; \boldsymbol{X}_N) \langle \eta_{ku}(p_k \beta_k) | \bar{\phi}_{ku}(\bar{q}_k \bar{\alpha}_k) \rangle \tag{54}$$



It is convenient to combine the two steps defined by equations (53) and (54) by introducing a ‘surfacing coefficient’, $S_{\rho_k}(q_k\alpha_k;\, p_k\beta_k)$, which transforms the coupling in figure 1(a) to the coupling in figure 1(c). We write

$$S_{\rho_k}(q_k\alpha_k;\, p_k\beta_k) = (\ell_{\rho_k}^{\bar{N}_{\rho_k}}\bar{\alpha}_{\rho_k}\bar{L}_{\rho_k}\bar{S}_{\rho_k}|\}\ell_{\rho_k}^{N_{\rho_k}}\alpha_{\rho_k}L_{\rho_k}S_{\rho_k})\langle\eta_{ku}(p_k\beta_k)|\bar{\phi}_{ku}(\bar{q}_k\bar{\alpha}_k)\rangle. \qquad (55)$$

The last recoupling step described above has brought the two orbitals corresponding to the two interacting electrons denoted by N and $N+1$ adjacent to each other in the matrix element in equation (52). The integral over $r_{N\,N+1}^{-1}$ can then be carried out immediately using the well known expansion

$$r_{N\,N+1}^{-1} = \sum_{\lambda m}\frac{4\pi}{2\lambda+1}\,\frac{r_<^{\lambda}}{r_>^{\lambda+1}}\,Y_{\lambda m}^{*}(\hat{\mathbf{r}}_N)Y_{\lambda m}(\hat{\mathbf{r}}_{N+1}) \qquad (56)$$

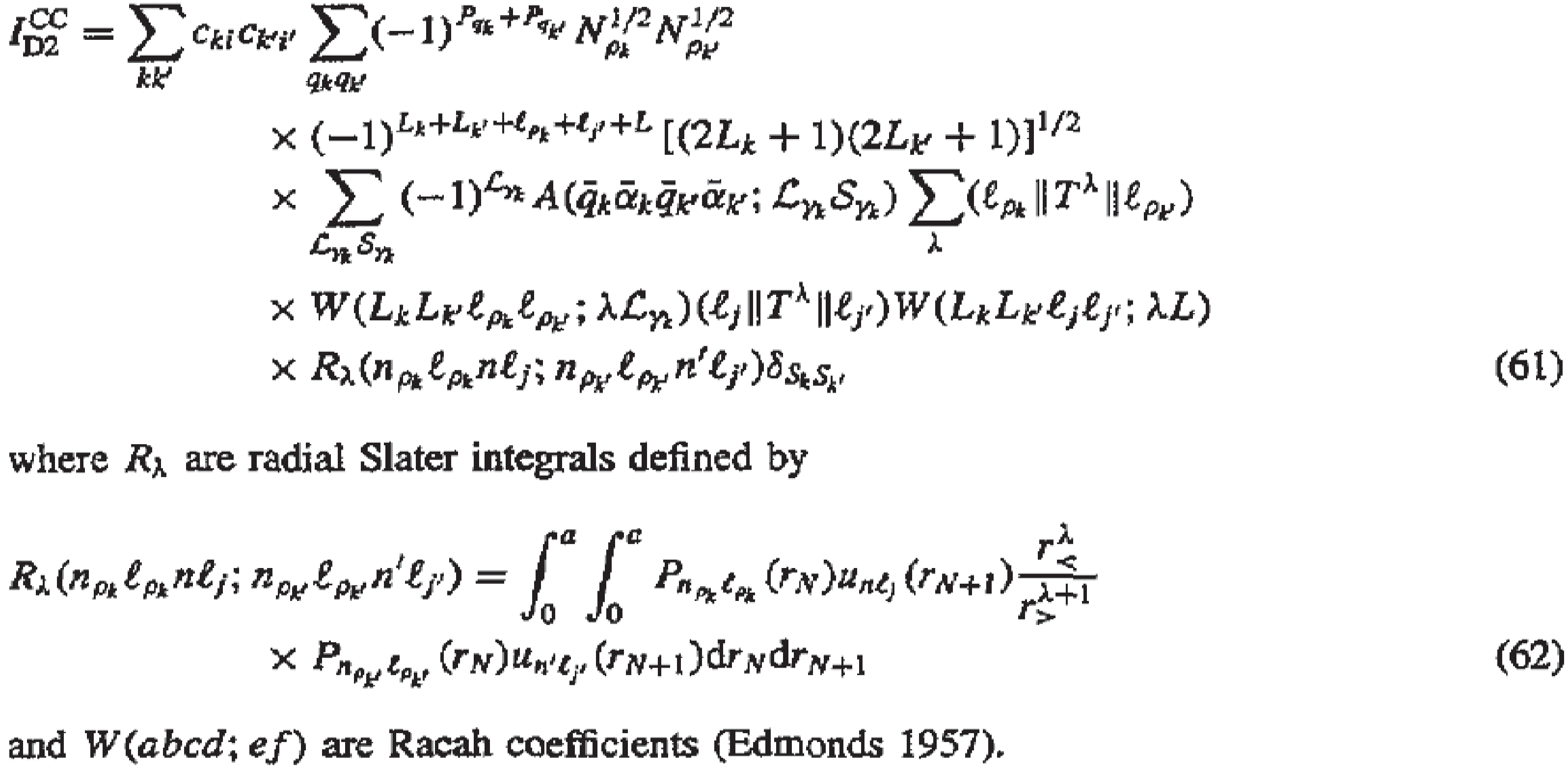
We also introduce the target recoupling coefficients

$$\begin{aligned}A(\bar{q}_k\bar{\alpha}_k\bar{q}_{k'}\bar{\alpha}_{k'};\, \mathcal{L}_{\gamma_k}\mathcal{S}_{\gamma_k}) = &\sum_{\{\mathcal{LS}\}'}\sum_{\bar{\alpha}_{\rho_k}\bar{L}_{\rho_k}\bar{S}_{\rho_k}}\sum_{\bar{\alpha}_{\rho_{k'}}\bar{L}_{\rho_{k'}}\bar{S}_{\rho_{k'}}}(\ell_{\rho_k}^{N_{\rho_k}}\alpha_{\rho_k}L_{\rho_k}S_{\rho_k}\{|\ell_{\rho_k}^{\bar{N}_{\rho_k}}\bar{\alpha}_{\rho_k}\bar{L}_{\rho_k}\bar{S}_{\rho_k})\\ &\times(\ell_{\rho_{k'}}^{\bar{N}_{\rho_{k'}}}\bar{\alpha}_{\rho_{k'}}\bar{L}_{\rho_{k'}}\bar{S}_{\rho_{k'}}|\}\ell_{\rho_{k'}}^{N_{\rho_{k'}}}\alpha_{\rho_{k'}}L_{\rho_{k'}}S_{\rho_{k'}})\langle\bar{\phi}_{ku}(\bar{q}_k\bar{\alpha}_k)|\eta_{ku}(p_k\beta_k)\rangle\\ &\times\langle\eta_{k'u}(p_{k'}\beta_{k'})|\bar{\phi}_{k'u}(\bar{q}_{k'}\bar{\alpha}_{k'})\rangle\delta_{p_kp_{k'}}\delta'_{\beta_k\beta_{k'}} \qquad (59)\end{aligned}$$

which can be rewritten in terms of surfacing coefficients defined by equation (55) as

$$\begin{aligned}A(\bar{q}_k\bar{\alpha}_k\bar{q}_{k'}\bar{\alpha}_{k'};\, \mathcal{L}_{\gamma_k}\mathcal{S}_{\gamma_k}) = &\sum_{\{\mathcal{LS}\}'}\sum_{\bar{\alpha}_{\rho_k}\bar{L}_{\rho_k}\bar{S}_{\rho_k}}\sum_{\bar{\alpha}_{\rho_{k'}}\bar{L}_{\rho_{k'}}\bar{S}_{\rho_{k'}}}S_{\rho_k}(q_k\alpha_k;\, p_k\beta_k)S_{\rho_{k'}}(q_{k'}\alpha_{k'};\, p_{k'}\beta_{k'})\\ &\times\delta_{p_kp_{k'}}\delta'_{\beta_k\beta_{k'}} \qquad (60)\end{aligned}$$

where the summations $\{\mathcal{LS}\}'$ are over all the intermediate couplings $\{\mathcal{LS}\}$ except $\mathcal{L}_{\gamma_k}$ and $\mathcal{S}_{\gamma_k}$ shown in figure 1(c). The delta function $\delta'_{\beta_k\beta_{k'}}$ means that all the angular momenta



$$
\begin{aligned}
I_{\mathrm{D2}}^{\mathrm{CC}} = \sum_{kk'} c_{ki} c_{k'i'} & \sum_{q_k q_{k'}} (-1)^{P_{q_k}+P_{q_{k'}}} N_{\rho_k}^{1/2} N_{\rho_{k'}}^{1/2} \\
& \times (-1)^{L_k+L_{k'}+\ell_{\rho_k}+\ell_{j'}+L} [(2L_k+1)(2L_{k'}+1)]^{1/2} \\
& \times \sum_{\mathcal{L}_{\gamma_k} \mathcal{S}_{\gamma_k}} (-1)^{\mathcal{L}_{\gamma_k}} A(\bar{q}_k \bar{\alpha}_k \bar{q}_{k'} \bar{\alpha}_{k'}; \mathcal{L}_{\gamma_k} \mathcal{S}_{\gamma_k}) \sum_{\lambda} (\ell_{\rho_k} \| T^{\lambda} \| \ell_{\rho_{k'}}) \\
& \times W(L_k L_{k'} \ell_{\rho_k} \ell_{\rho_{k'}}; \lambda \mathcal{L}_{\gamma_k}) (\ell_j \| T^{\lambda} \| \ell_{j'}) W(L_k L_{k'} \ell_j \ell_{j'}; \lambda L) \\
& \times R_{\lambda}(n_{\rho_k} \ell_{\rho_k} n \ell_j; n_{\rho_{k'}} \ell_{\rho_{k'}} n' \ell_{j'}) \delta_{S_k S_{k'}}
\end{aligned}
\qquad (61)
$$

where R_λ are radial Slater integrals defined by

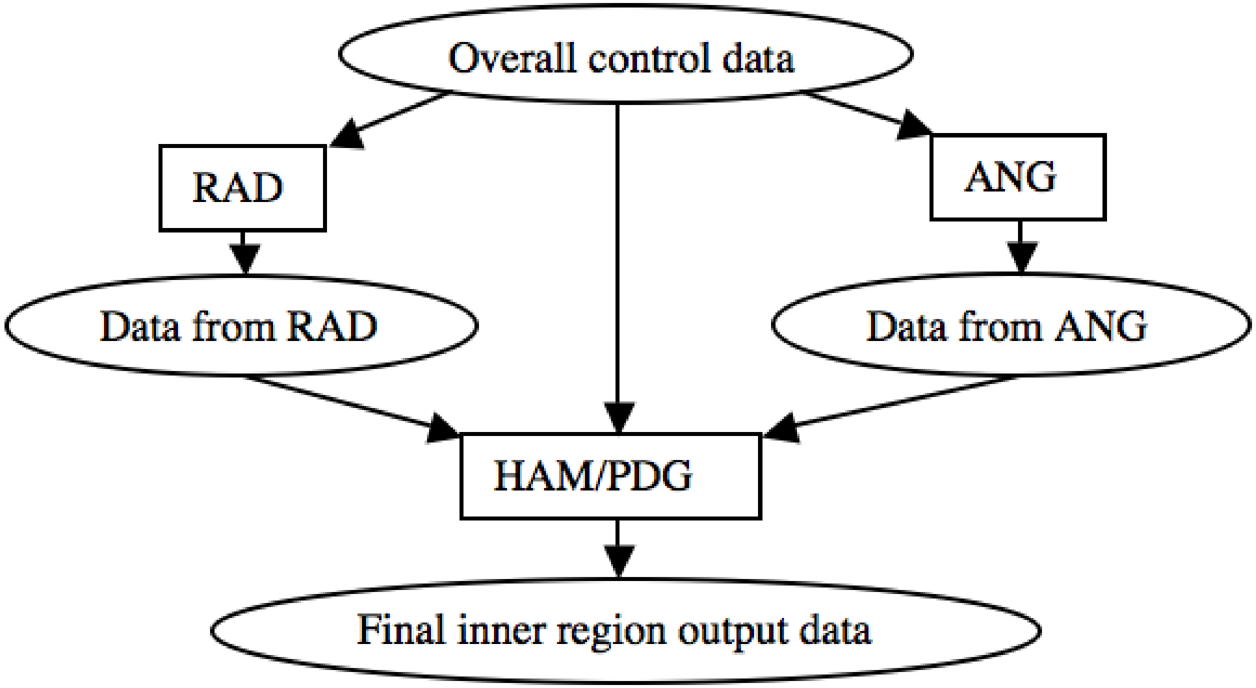
$$
\begin{aligned}
R_{\lambda}(n_{\rho_k} \ell_{\rho_k} n \ell_j; n_{\rho_{k'}} \ell_{\rho_{k'}} n' \ell_{j'}) = & \int_0^a \int_0^a P_{n_{\rho_k} \ell_{\rho_k}}(r_N) u_{n\ell_j}(r_{N+1}) \frac{r_<^{\lambda}}{r_>^{\lambda+1}} \\
& \times P_{n_{\rho_{k'}} \ell_{\rho_{k'}}}(r_N) u_{n'\ell_{j'}}(r_{N+1}) \mathrm{d}r_N \mathrm{d}r_{N+1}
\end{aligned}
\qquad (62)
$$

and $W(abcd; ef)$ are Racah coefficients (Edmonds 1957).

Note: ‘R’ is purely radial, the rest are angular factors

- There are similar but more complicated expressions for the exchange CC integrals (eg involving '9j' symbols), and similar expressions for the BC and BB integrals, and corresponding dipole matrices.
- The introduction of the 'surfacing coefficients' by BBD was a major performance enhancement for the CC integrals, as angular integrals are recoupled to minimize recalculation of existing components.
- The BB integrals are speeded up to a lesser extent by the surfacing coefficients: for modern calculations with large sets of bound orbitals they have become the most time consuming part of the angular part of the program
- Hamiltonian construction is thus split into three separate stages: ANG, RAD and HAM.
- HAM may contain the diagonalization, otherwise PDG is used.

Package structure



ANG

- Plan: parallelize construction of ‘surfacing tables’ using shared memory. Then parallelize ‘angbb’ in particular (and angcc, angbc etc).
- eg: a serial ANG run for a complex Fe+ calculation took ~16 hours on an IBM Power7 of which angbb took up ~80%: in general angbb and surfacing tables dominate timing.
- At the same time improve commenting in the code so that it clearly relates to the BBD and other references.
- New coding in F95+F2003 (generally following ‘Coding Standards for the UKRmol Project’ by MP, JM Carr and JD Gorfinkiel) with low-lying C modules:
- Parallelization to be attempted using both MPI with shared memory segments for the surfacing tables and OpenMP.

- ANG work performed by CJN (~4.5 months effort over 9 months)
- New multi-node MPI code with sms written (to ‘beta+’ standard) and currently under a severe testing/ bug-fixing regime to reach ‘release’ standard: inter-node and intra-node communicators defined and introduced.
- A relatively compact sms-communicator derived-type (very suitable for ANG) is being checked against the NAG-developed software used in CASTEP.
- Technical difficulty: we don’t know the size of the surfacing table in advance: it is calculated by constructing the coefficients locally and storing them in linked lists until the final size is known by a ‘group leader’. We can then form a shared array of coefficients and a shared array of indices.
  - Some re-writing for sms required (Fortran->C does ‘not’ allow allocatable arrays to be C structures)
- Performance results to follow once code is declared ‘fully’ bug-free.

New ‘sms’ type

```
type sms                ! shared memory segment type
     private
     integer(c_int)        :: keygen_id = -1 ! project id of segment
     integer(c_int)        :: id = -999      ! segment IPC id
     type(c_ptr)           :: ptr            ! c-pointer to segment
     integer(c_int)        :: num_els        ! # datatype elements
     integer(c_int)        :: datatype       ! type of data
     integer, pointer      :: iptr(:)     ! integer fortran ptr to SMS
     real(wp), pointer     :: rptr(:)     ! real fortran ptr to SMS
   contains
     procedure :: attach => attach_sms
     procedure :: kid => getId
     procedure :: shm => getShmid
     procedure :: detach => detach_sms
     procedure :: getiValue
     procedure :: getrValue
     generic   :: get => getiValue, getrValue
  end type sms

  interface sms
     procedure constructor
  end interface
```

- OpenMP tests
  - An OpenMP coding of surfacing coefficients and angbb existed which apparently worked on HPCx (factor of 4 performance for 8 threads) with coarse-grained parallelism (ie upper loops). However it seems that IBM’s interpretation of OpenMP was ‘faulty’.
  - Testing on HECToR shows evidence of a data race in ‘surfce’. The likely causes of this are under investigation but hampered by the lack of thread-checking tools on HECToR: the F95 coding with ‘out-of-scope’ module variables is very complex to analyze otherwise.
  - OpenMP allows orphaned threadprivate directives. One cannot declare out of scope variables to be ‘shared’: out-of-scope . A restructuring of the code is possible but would undermine the logical modular structure of the code and future maintenance. It’s possible to put in simple fine grained OpenMP at lower levels of the structure, but the MPI/FIPC/sms approach is more wide-ranging and exciting.
  - Tools such as ‘Intel Parallel Inspector’ would be very helpful.
  - Temporary conclusion: limited potential gains of OpenMP parallelism should be bypassed with combined MPI/FIPC approach. The coarse grain parallelism of the earlier OpenMP code is better handled by high-level MPI communicators which are fitting ‘naturally’ onto the finer grained sms communicator.
  - If necessary, we’ll remove ‘surfce’ OpenMP and continue with angbb etc.

RAD

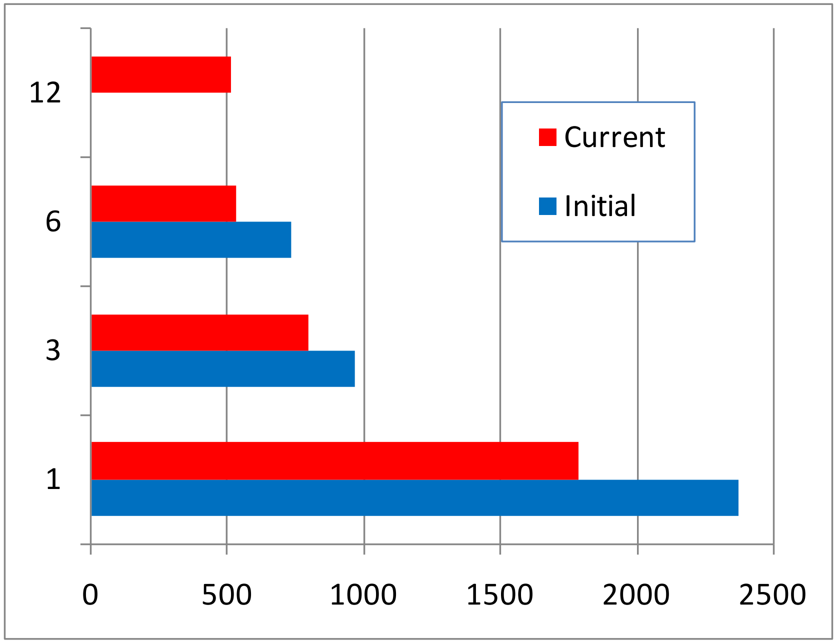
- Essentially RAD constructs the ‘*R*’ and other 1-2 electron radial integrals for bound and continuum orbitals.
- Again there are two parts: it first ‘creates’ the continuum orbitals, then forms the integrals.
- Two distinct types of continuum orbitals:
  - ‘Traditional’, generated numerically and iteratively on a grid of radial values, as consecutive solutions of a model potential problem (and orthogonal to the bound orbitals) with a fixed boundary condition
  - There are typically 20-30 ‘trad’ orbitals per ‘*l*’, but a ‘Buttle’ correction is needed: the continuum orbitals are ‘accurate’ representations of ‘real’ low-lying continuum orbitals.
  - ‘B-spline’: constructed from a large set of B-splines, orthogonal to the bound orbitals (which are also represented with the B-spline basis) and diagonalized on the model potential, free boundary conditions.
  - There may be 180-200 B-spline orbitals per ‘*l*’, but they span the continuum: Buttle corrections are not needed.

- The larger number of spline continuum orbitals has a knock-on effect on both the integral generation in RAD and on the Hamiltonian formation in HAM.
- Integral evaluation: multiple do-loops over labels with either fine-grid Simpson’s rule integrations (trad) or B-spline integrations as innermost.
- However the 2-electron integral loops are (mostly) cunningly written so that the inner integral can be saved across many integrals.
- (nb a third system generates ‘Buttle-free’ continuum orbitals iteratively: 30-40 needed to ‘span’ the continuum: good for HAM, but the RAD time-saving trick could not be used: this system (Plummer and Noble 1999) may be reintroduced as complementary work by MP.)

Plan for RAD: AG Sunderland, 3 months over ~7.5 months

- Merge the B-spline and traditional RAD codes into a single code
  - Done with a few ‘final refinements’ left for nearer end of project: input parameter chooses which version.
- Extend an OpenMP treatment by H van der Hart (QUB) of B-spline generation and CC integral generation to the rest of the (spline) code integral generation.
  - Done: the !$OMP commands have also been moved from the innermost loop (of 7-8) to 2-3 loops from innermost, with some loops condensed. Needed some THREADPRIVATE definitions to work.
  - A major performance enhancement for the exchange integrals was achieved by loop re-ordering, to make much better use of the ‘saved’ inner integral.
  - Nb: the integrals formation section takes >~0.8 of the time for 180 B-splines per ‘l’.
    - Stage 1 Time (Orbital Generation) = 182.1008 secs
    - Stage 2 Time (Integral Evaluation) = 1607.0104 secs
    - Elapsed time = 1789.1112 secs,
    - (‘trad’ serial takes 3+7=9 seconds for 30 B-splines per ‘l’)

OpenMP Oxygen test timings in seconds:
(the orbital-integral time ratio is maintained)



- Still to finish for RAD: add in MPI communicators over outer loops for integral generation: maybe add in sms code (as an exercise? Here the mixed mode formulation works well).
- This will automatically introduce a test for parallel I/O which will be important for the final ANG modifications and for HAM.
  - (generally speaking RAD is a good introduction to the package before serious work on HAM)
- Work for HAM (~4.5 months effort, ~3 months AGS, ~1.5 CJN): introduce MPI/IPC/sms code into HAM, parallel read, standardised choices of format for main output (XDR for portability and PDG or binary if preferred, using ‘xstream’ ideas).

- XStream and Filehand I/O handling:
  - PFARM uses XDR files to read inner region data and between stages. MPI-IO files are more efficient in parallel but not portable.
  - Xstream (CJN) provides a wrapper to allow either option at any given file read/write.
  - Introduced (for PFARM) during the previous dCSE.
  - Filehand is a double-buffered I/O module written by VM Burke for intermediate data in the inner region codes using direct access files and (being) parallelized by CJN. Initially devised to avoid continuous I/O, current tests show that the buffer size (hence record length) is fairly performance insensitive on HECToR.
  - The parallel Filehand and xstream will be compared (and combined).
    - (parallel reading from direct access files is straightforward(?))