# Parallelization and porting of UKRMol-in codes

Michael Lysaght and Jimena Gorfinkiel

[1]The Open University

HECToR dCSE Technical Meeting 2011

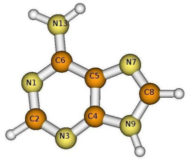
## Project duration

- 12 month dCSE project to parallelize diagonalization routines in UKRMOL-in.
- dCSE contract awarded 10th April 2011.
- dCSE contract ends 10th April 2012.

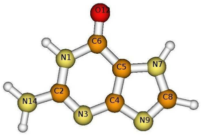
## The Science

- UKRMol polyatomic suite of codes are used to model electron- and positron- molecule scattering processes.
- Processes are fundamental to astrophysics, plasma physics, damage process in biological environments.
- UKRMol currently being used to study mechanisms of DNA strand breaking caused by low-energy electron collisions

## Current capability



Adenine



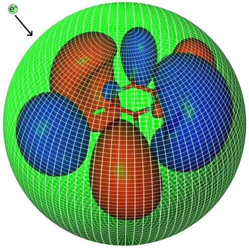
Guanine

- Diverse calculations in academia and industry
- Scattering from biomolecules
- Positron scattering and annihilation - $C_2H_2$

- Electronic excitation - $CH_4$ $H_2O$ $C_4N_2H_4$
- Collisions with small molecular clusters $(H_2O)_2$

- Currently don't run calculations that they would like, only within current capability.

## Underlying theory



- The UKRMol suite based on the time-independent R-matrix theory of electron scattering.
- Ab initio method for solving Scrödinger eqn.
- Based on division of space concept.

## dCSE and R-matrix codes

- The atomic versions of the R-matrix codes (PRMAT) have been ported and optimized on HECToR (Dr. M Plummer and Dr. A G Sunderland) with dCSE support.
- Current goal is to interface the PFARM part of PRMAT with the UKRMOL-in suite of codes (Dr M Plummer).
- Current dCSE project to parallelize the construction of the atomic Hamiltonian (Dr M Plummer).

## UKRMol codes - background

- The UKRMol suite developed with CCP2 support mainly by Prof. J. Tennyson's group at UCL over the last three decades (More recently by Dr. J Gorfinkiel at The OU).
- The codes were originally developed using F77. Parts of the suite remain as F77 legacy code.
- The codes are available to UK academics and non-UK scientists through the CCPForge website.
- UKRMol is currently used by groups in USA, India, Japan, France and Canada.

## UK-RAMP project

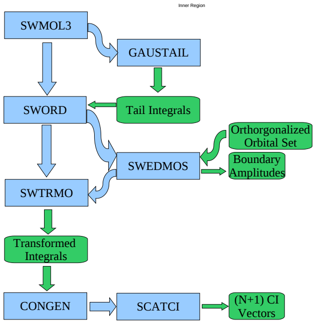
- Awarded to QUB, UCL, OU, and STFC CSED by EPSRC to bring together UK expertise in electron-atom/molecule and laser-atom/molecule interactions.
- Central drive is to combine time-independent atomic and molecular R-matrix codes with the time-dependent laser-atom methods of the HELIUM code (Prof. K T Taylor, QUB).
- Supported by dCSE: Atomic RMT code recently developed and ported to HECToR (see next talk by Dr. L Moore).
- Major UK-RAMP goal: to do the same for molecules using the UKRMoL suite – RMT-Mol.

## UKRMol within UK-RAMP

- Significant improvements were made to UKRMol in the first year of UK-RAMP (Dr. J Carr):
- Conversion of many modules to the F95 standard.
- Code reuse and encapsulation.
- Adaptation of diagonalizers to take better advantage of the ‘partitioned’ R-matrix method.

## The Inner Region codes (UKRMol-in)

- Contains a series of programs which:
- Calculate integrals over target and continuum electron orbitals along with orthogonalization (SW-)
- Construction and diagonalization of the Inner Region Hamiltonian (CONGEN and SCATCI)



## UKRMol-in

- The suite also contains modules to generate Hartree-Fock-SCF or pseudonatural orbitals and the basis sets for the description of the continuum.
- Being superseded by the use of standard Quantum Chemistry codes to generate more sophisticated orbitals.
- The solution of the Outer Region part of the problem is carried out in UKRMol-out. UKRMol-out will be based on PFARM.

## Bottleneck in UKRMol-in

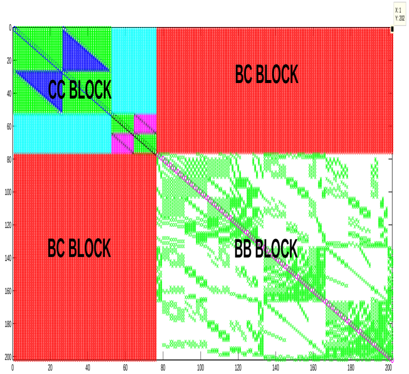
- Limiting factor in the calculations is the construction and diagonalization of the target+electron (N+1) Hamiltonian (SCATCI) $\sim 90\%$ of time spent here.
- Hamiltonian is highly sparse ($\sim 99\%$) – can take advantage of Arnoldi-based diagonalization methods.
- Using a “partitioned” R-matrix method, only $\sim$ 5-10$\%$ of eigenpairs are required from diagonalization of large Hamiltonian matrices.

## The Hamiltonian matrix

- Real and symmetric.
- Lower triangle of matrix written to H file.
- Stored in (unordered) (symmetric) COO sparse format.
- Largest H file size to date $\sim$ 60 GB

## The form of the Hamiltonian matrix

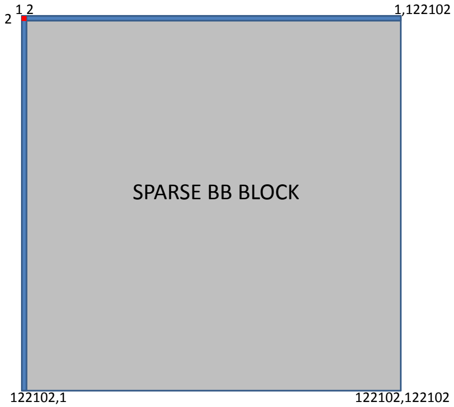
- CC and BC blocks are unordered dense and dimensions are known before construction.
- BB block is the sparse part of the matrix and by far the largest part of the matrix.
- BB sparsity structure is unpredictable and changes from molecule to molecule.



- BB block is constructed in order.

## A more realistic large problem

- Dimensions are for Phosphate calculation, presented later



## A parallel Eigensolver

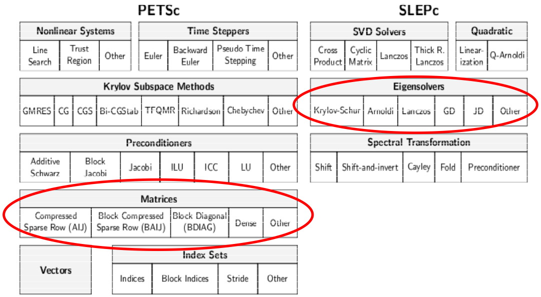
- SLEPc is the Scalable Library for Eigenvalue Problem computations.
- SLEPc is built on top of PETSc (available on HECToR) and is considered an extension of PETSc.
- Enforces the same programming paradigm as PETSc.
- It is being developed by the High Performance Networking and Computing Group (GRyCAP) of Universidad Politecnica de Valencia.

## Some of the benefits of SLEPc

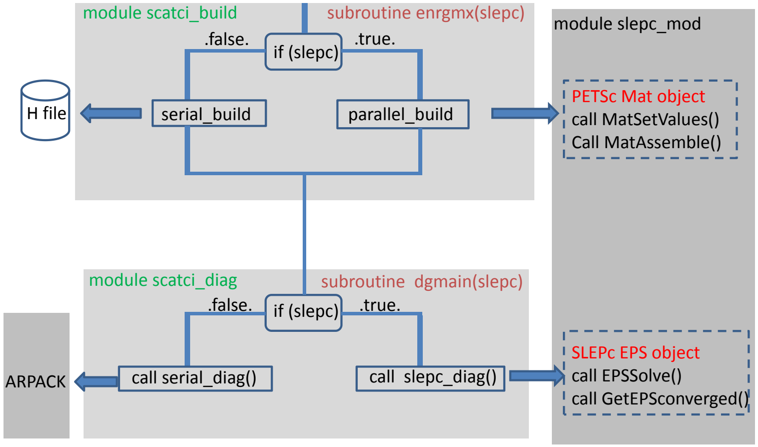
- Data-structure neutral implementation. Problems can be solved with matrices stored in parallel and serial, sparse and dense formats.
- Run-time flexibility, giving control over the solution process.
- Portability to a wide range of parallel platforms.
- Usable from code written in C, C++, F77 and F90.
- Extensive documentation – users manual, example programs, online manual for subroutines.
- Seamless integration with well-established packages such as ARPACK.
- Online Support through PETSc-users maintenance email.

## Scheme of SLEPc classes

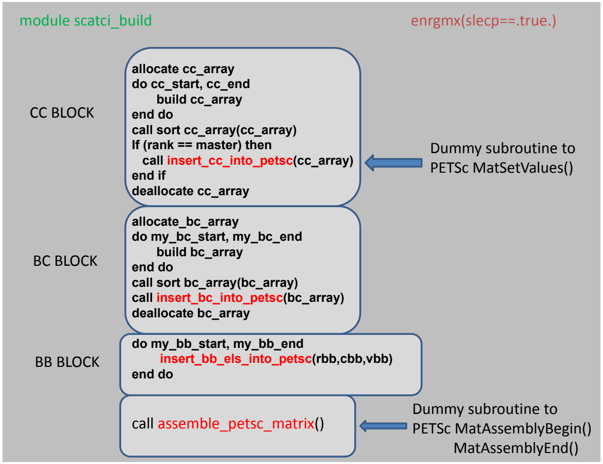
- Eigenproblem solver (EPS) is one of four objects provided by SLEPc.
- Provide a level of abstraction similar to PETSc solvers and use low level PETSc infrastucture such as Mat and Vec.



## Code modifications



## Parallel Build


module scatci_build
enrgmx(slecp==.true.)
CC BLOCK
BC BLOCK
BB BLOCK
Dummy subroutine to PETSc MatSetValues()
Dummy subroutine to PETSc MatAssemblyBegin() MatAssemblyEnd()

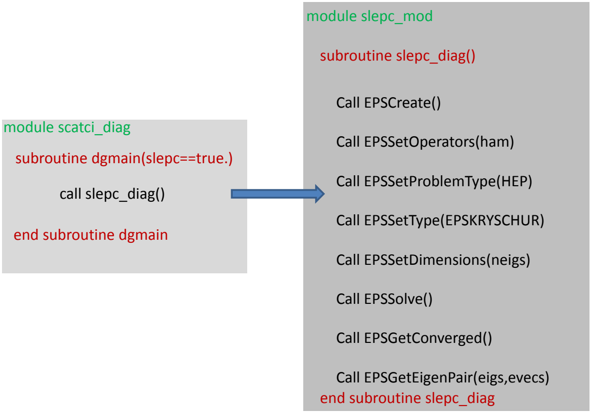
```
allocate cc_array
do cc_start, cc_end
      build cc_array
end do
call sort cc_array(cc_array)
If (rank == master) then
   call insert_cc_into_petsc(cc_array)
end if
deallocate cc_array
```

```
allocate_bc_array
do my_bc_start, my_bc_end
      build bc_array
end do
call sort bc_array(bc_array)
call insert_bc_into_petsc(bc_array)
deallocate bc_array
```

```
do my_bb_start, my_bb_end
       insert_bb_els_into_petsc(rbb,cbb,vbb)
end do
```

```
call assemble_petsc_matrix()
```

## Parallel Diagonalization



```
module scatci_diag
  subroutine dgmain(slepc==true.)
          call slepc_diag()
  end subroutine dgmain
```

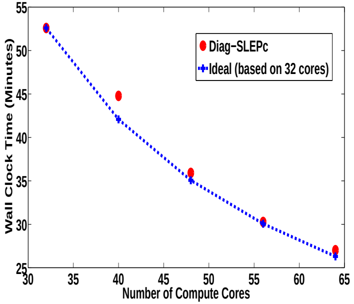
```
module slepc_mod
  subroutine slepc_diag()
     Call EPSCreate()
     Call EPSSetOperators(ham)
     Call EPSSetProblemType(HEP)
     Call EPSSetType(EPSKRYSCHUR)
     Call EPSSetDimensions(neigs)
     Call EPSSolve()
     Call EPSGetConverged()
     Call EPSGetEigenPair(eigs,evecs)
  end subroutine slepc_diag
```

## Early Stage Scaling on HECToR

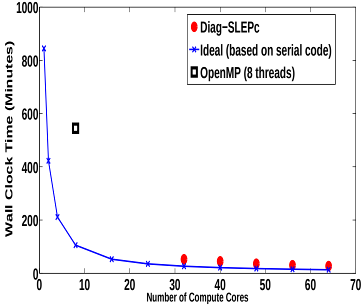
- New code compiled using PGI (no optmization flags)
- Results in excellent agreement with serial code.
- Test carried out for a Phosphate calculation (the DNA backbone is sugar molecules linked by phosphate groups).
- Hamiltonian matrix order = 122102,
  no. of non-zero elements = $1.6 \times 10^8$

## Scaling of the Diagonalization Phase (Diag-SLEPc)

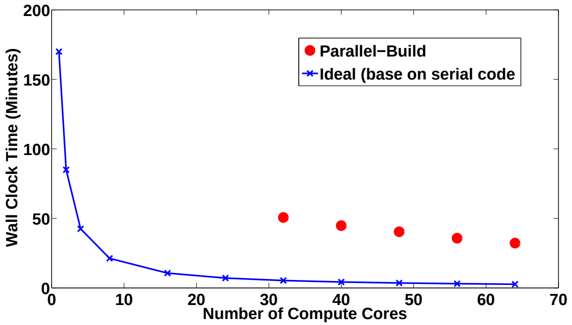
### Scaling of Diag-SLEPc



### Comparison to serial diag



## Scaling of the Build



- Build is poorly load-balanced mainly due to lower-triangle-only build.

## Things to do...

- Test code on bigger Hamiltonians ( $> 600{,}000 \times 600{,}000$).
- Improve load-balancing with regards to the BC BLOCK.
- Reduce the over-head of MatSetValues (using buffers).

## Open questions

- Can we make use of parallel graph-partitioning software (e.g ParMETIS, SCOTCH, PaToH,...)?
- Improve Memory Bottlenecks: Can I/O be improved (e.g. re-engineer SWORD)?
- SWORD calculates integrals and and writes integral array to disk (size $\sim$ 1 GB).
- Integral arrays are unordered and therefore each core needs to read array into RAM.

## Conclusion

- SCATCI has been parallelized and ported to HECToR.
- Makes use of more than one node for the first time.
- SCATCI has been interfaced with SLEPc and results are in excellent agreement with serial code.
- First test calculations have seen scaling to 64 compute cores on HECToR (small problem size).
- More investigations under way to improve load-balancing and to reduce memory bottlenecks.