Hierarchic Data Structures for Sparse Matrix Representation in Large-scale DFT/HF Calculations

Paweł Sałek
Theoretical Chemistry, KTH, Stockholm

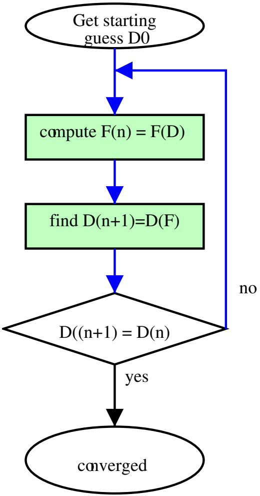
16-18 October 2007, Linköping

Talk Outline

- Running large-scale ab-initio calculations – scaling.
- Solving problems with sparse matrices.
- OpenMP parallelization – problems.

Performance of the SCF Cycle

- Computation of Kohn-Sham matrix F is time-consuming.
- For really large systems, density evaluation ($F \rightarrow D$) is time-consuming as well.
- Matrix memory usage grows quadratically.
- Local basis set – basis functions localized on atoms.



Research Group

- Elias Rudberg: Interaction evaluation (PhD in December).
- Emanuel Rubensson: Sparse matrices.

$F \longrightarrow D$ Step

Density traditionally obtained via diagonalization and *aufbau* principle:

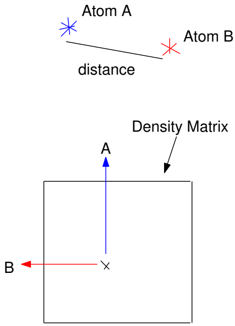
$$FC = \epsilon SC \qquad D = C_{\mathrm{occ}} C_{\mathrm{occ}}^T$$

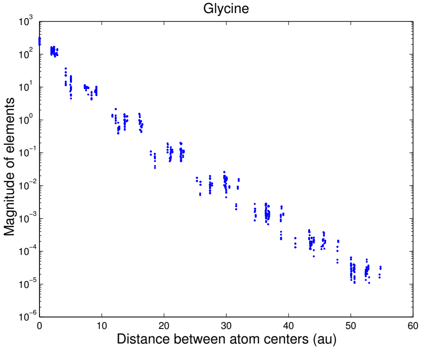
- Diagonalization does not scale linearly.
- Density optimization and purification algorithms scale linearly when sparsity is used.

Sparse Matrices in Quantum Chemistry

- Matrices are used to represent operators D and F.
- Overlap matrix, Density matrix, Fock Matrix, Kohn-Sham matrix.
- Matrices must be represented in such a way that common operations are fast.
- Sparsity appears only for larger molecules ($>$ 50 atoms).

Sparsity Patterns

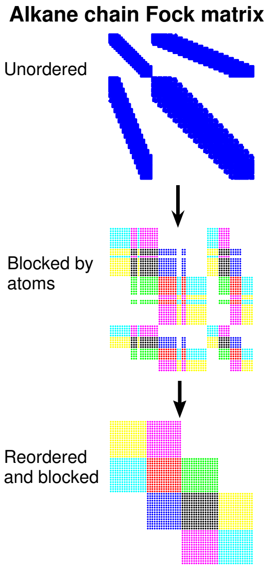




Matrix sparsity depends on the basis set and geometry and to some extend on the band gap.

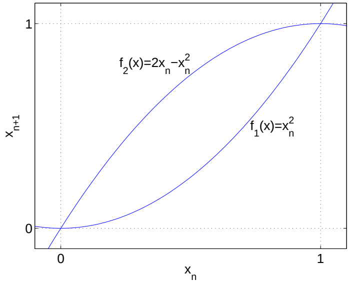
Taking Advantage of Sparsity Patterns

- Sparsity appears in blocks.
- Reorder atoms to merge the atom blocks in larger ones.
- Use BLAS for operations on blocks and Compressed-Sparse Row (CSR) format for block storage.
- Enforce sparsity by small element truncation.



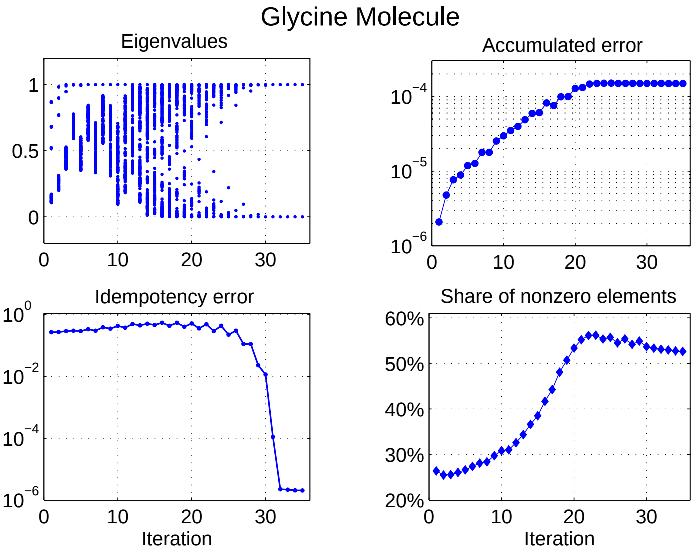
$F \rightarrow D$ Step With Sparse Matrices

- Use Trace-Correcting Purification – a series of spectral transformations.
- Performance limited by the sparse matrix multiplication speed.




```
compute P = (lmax I-F)/(lmax-lmin)
while abs(trace(P)-N)>threshold
  if(trace(P)>N) then
     P := P*P
  else
     P := 2*P-P*P
end while
```

Example TC2 Application



Problems with TC2

- Number of TC2 iterations depends the bandgap.
- Control the error: TC2 error grows exponentially with the number of iterations.

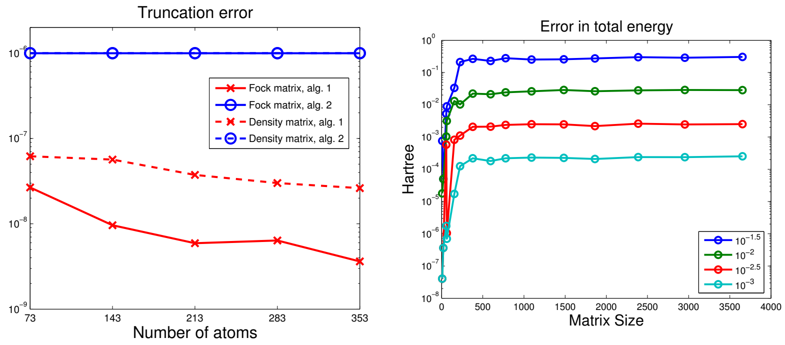
- More flexible representation than Compressed Sparse Row is needed for easy implementation of other algorithms.

Systematic Small-Submatrix Selection Algorithm

- Error introduced by truncation of small elements.
- First approaches considered only distance between atoms and empirical threshold factors – unreliable!
- More advanced approaches look at the norms of neglected blocks – more reliable but strict error control still impossible.
- SSSA looks at the error of the entire matrix. Provides strict error control.

The Effect of SSSA on Total Energy Error

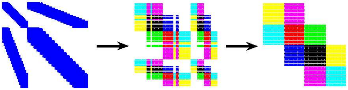
Benchmark on water clusters with Hartree-Fock and STO-3G. alg. 1: threshold based filtering, alg. 2: SSSA.

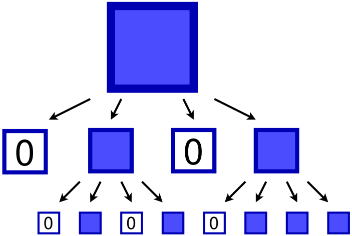


SSSA provides rigorous error control. → Saves time and gives trustworthy results. Energy extrapolation possible.

Hierarchic Matrix Library (C++)

HML allows for low-overhead random element access.





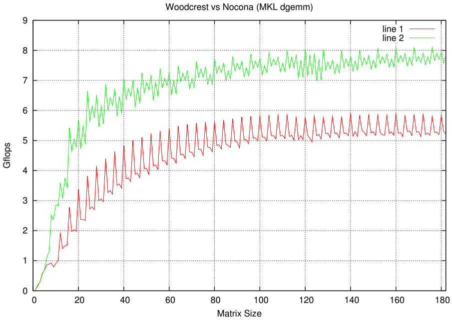

```
typedef Matrix<Matrix<Matrix<double> >
> MyMatrixType;
typedef Matrix<Matrix<Matrix<long double> >
> MyAccurateMatrixType;
```

HML Features

- Easy to code and maintain.
- Block size determined by the architecture performance, not chemistry.
- Low overhead random element access.
- Blocked algorithms easy to express:
  1. Matrix multiplication, also by transposed matrices.
  2. Use of matrix symmetry.
  3. INverse CHolesky factorisation (INCH).

Block Size Tradeoff

- Smaller block size → more opportunity for screening.
- Larger block size → better block-block multiplication performance.



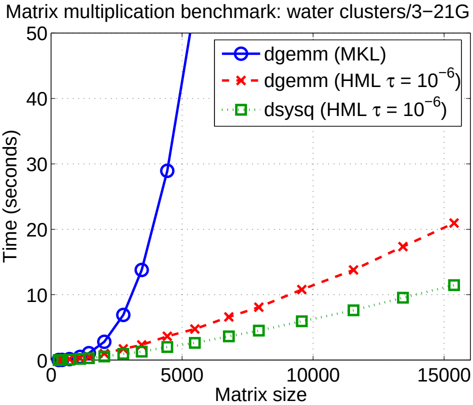
Example Implementation of C := beta*C + A*B


```
static void multiply(const Matrix<Telement>& A,
                     const Matrix<Telement>& B,
                     Matrix<Telement>& C, double beta) {
  for (int colC = 0; colC < C.ncols; colC++)
    for (int rowC = 0; rowC < C.nrows; rowC++) {
      Telement::multiply(A(rowC, 0), B(0, colC),
                         C(rowC, colC), beta);
      for (int colA = 1; colA < A.ncols; colA++)
        Telement::multiply(A(rowC, colA), B(colA, colC),
                           C(rowC, colC), 1);
    }
}
```

- Lowest level (block) multiplication expressed in terms of BLAS calls.
- Template expansion will generate (*instantiate*) code for all the remaining hierarchy levels.

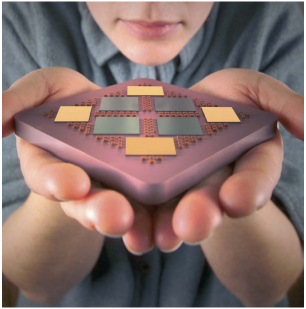
Intel MKL vs HML Benchmark

- HML design allows for easy implementation of symmetric matrix multiplication (sysq: $S = \alpha T^2 + \beta S$) as needed by TC2:
- sysq twice faster than general sparse multiplications.



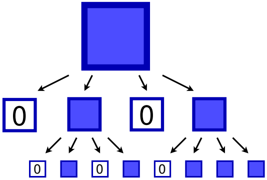
OpenMP Parallelization

- Parallel programs necessary to efficiently use modern multi-core hardware.
- OpenMP less *invasive* and easier to load-balance.
- Problems with scaling and. . . compiler support.
- Poor compiler support! GNU gcc is the only reliable, OpenMP-enabled compiler known to us so far.



Details of OpenMP Parallelization

- Pick a level in the hierarchy, run a parallel loop with dynamic scheduling over it.
- Approach trivial to implement.
- Higher levels: coarse load distribution.
- Lower levels: thread startup overhead.



Exceptions and OpenMP

- OpenMP and C++ exceptions *do* interact.
- Threads must `catch` any exceptions that are generated. The behavior is undefined otherwise.
- We do the *right* thing (in case you ask).

```
#pragma omp parallel for
for (int i = 0; i < MAX; i++) {
  try {
   // Heavy lifting here
  } catch (...) { /* Handle it nicely. */ }
}
```

Compiler Problems

GNU C++ OpenMP support since 4.1(?). No problems found. Sequential performance lower than its competitors.

Portland C++ fairly warns that it cannot handle exceptions and OpenMP at the same time. A honest warning but...

Intel C++ 3 versions tried. *All* of them had bugs either in sequential code or in OpenMP parallelization.
8.1 fails to generate correct sequential code; miscompiles OpenMP code as well.
9.1 works sequentially; compiler crashes with executed with `-openmp` flag.
10.0 fails to generate correct sequential code. Support tickets with Intel are open.

OpenMP Speedup

- Timings taken on 1.5GHz Itanium2, 4 CPU (luc2, PDC), 4 threads.
- Glycine-Alanine chain with 1600+ atoms, HF method. GNU C++.

Operation	CPU time [s]	Wall time [s]	Speedup
FDS-SDF	133.54	53	2.66
Purification	947.59	454	2.13

- Acceptable multiplication load balancing (3.5/4.0) but serial data management has negative impact on scalability.
- Additionally, purification involves serial error estimation routines.

Summary & Outlook

- SSSA – for strict error control.
- HML – flexible sparse matrix representation.
- A number of algorithms (arbitrary MxM multiplications, inverse Cholesky factorisation) already implemented.
- OpenMP parallelization.

Outlook

- Analyse the sparsity in the QM methods beyond the algorithms relevant for SCF: Linear response for calculation of molecular properties.

Small Submatrix Selection Algorithm (SSSA)

Given a matrix norm $\|\cdot\|$ and an error limit ε we want to find a sparse approximation $\widetilde{A}$ of A so that $\|A - \widetilde{A}\| < \varepsilon$.

SSSA:

1. Compute the Frobenius norm of each submatrix.
2. Sort the values in descending order.
3. Remove submatrices from the end as long as the error is within desired accuracy.

$\Longrightarrow$ Error very close to the requested value in the Frobenius norm.