Optimizing the matrix computations in environmental models

1. *Types of the matrix computations that appear in environmental models*
2. *Matrix computations related to the pseudo-spectral method*
3. *Matrix computations related to finite elements and finite differences*
4. *Matrix computations related to the chemical sub-model*
5. *Dense matrix computations*
6. *Implementation of a special sparse matrix technique*
7. *Numerical results*
8. *Conclusions and open problems*

Types of matrix computations

- Using FFTs in the computations
- Computations with different banded matrices
- Computations with general sparse matrices
- Computations with dense matrices

1. A large part of the computations is related to matrices
2. The implementation of efficient techniques for optimizing the matrix computations is crucial

Matrix computations related to the pseudo-spectral method

1. FFTs - reduction of the number of computations from N^2 to $N \log(N)$
2. Development for different algorithms for different computers
3. Swarztrauber´s algorithm
4. Using multiple FFTs (Temperton´s algorithm)
5. NAG FFTs
6. Difficulties (sometimes transposed matrices have to be formed and used, special order is very often needed)

Matrix computations related to finite elements and finite differences

- Banded matrices, sometimes symmetric and positive definite, sometimes tri-diagonal
- Typical operations: matrix-vector multiplications, factorizations, back substitutions and inner products
- Standard libraries used: LAPACK, NAG Library, SCALAPACK
- Exploiting the symmetry: leads to reduction of the storage requirements, but degrades the speed of the computations
- Recursive algorithms (Andersen, Wasniewski and Gustavson, 2001)

Matrix computations related to the chemical sub-model

- QSSA - practically no matrix computations are used with this method
- Classical ODE methods: calculation of the Jacobian matrix, formation of the shifted Jacobian matrix, factorization, back substitution
- Partitioned methods: the same operations as above, but only for the diagonal blocks (the strong blocks)
- Using conjugate gradient type methods (matrix-vector multiplication and inner products). Preconditioning might be needed

Dense matrix computations

- Direct use of sparse matrix packages for general matrices (Duff, Erisman and Reid, 1986, Zlatev, 1991) is not efficient because the matrices are small and the number of non-zero elements is relatively high).
- The use of dense matrix techniques is more efficient than exploiting the sparsity by applying general-purpose sparse matrix software
- Used packages: LAPACK, BLAS, NAG Library
- Recursive algorithms can again be applied
- General properties of the dense matrix computations: (i) high performance can be achieved (regular structure), but (ii) the number of computations is high

Special sparse matrix techniques

Why is a special sparse matrix technique needed?

Disadvantages of the general sparse matrix techniques

- Implicit addressing
- Treatment of fill-ins
- Copies of rows and columns to the end of the ordered lists
- Garbage collections
- Finding a pivotal element
- Many short loops are to be carried out
- Use of many integer arrays

Removing the disadvantages

- **No pivoting for numerical stability: allows us to perform a preliminary reordering (Markowitz type of strategy for reducing the number of fill-ins)**
- **The positions of all fill-ins are determined and locations for the fill-ins are reserved**
- **A loop-free code for the calculation of the LU factorization is developed**
- **A loop-free code for the calculation of the back-substitution is developed**

The price that has to be paid

- The pivoting for numerical stability is sacrificed
- Loop-free codes tend to be long

Similar ideas:

Willoughby (1970)	IBM Center (Yorktown Heights)
Sandu et al. (1999)	Iowa University
Swart and Blom (1996)	CWI (Amsterdam)

Numerical results

Method	Computing time	
QSSA-1	12.85	
QSSA-2	11.72	
Euler	15.11	
Trapez.	15.94	
RK-2	28.49	
Part. dense	10.09	Based on Euler
Part. sparse	9.21	Based on Euler

Conclusions and open problems

- **The results can be improved by choosing the right way to handle the matrix computations**
- **Some new techniques (such as recursive computations) have not been tried yet**
- **Iterative methods with preconditioning might improve the performance (the problem of finding an optimal preconditioner is open)**