Development and Analysis of Sparse Matrix Concepts for Finite Element Approximation on general Cells

T. Pick, Institut für Bauinformatik, Universität Hannover, Germany
pick@bauinf.uni-hannover.de

Heimsund, Department of Mathematics, Bergen, Norway
bjornoh@math.uib.no

P.Milbradt, Institut für Bauinformatik, Universität Hannover, Germany
milbradt@bauinf.uni-hannover.de

Summary

In engineering and computing, the finite element approximation is one of the most well-known computational solution techniques. It is a great tool to find solutions for mechanic, fluid mechanic and ecological problems. Whoever works with the finite element method will need to solve a large system of linear equations. There are different ways to find a solution. One way is to use a matrix decomposition technique such as LU or QR. The other possibility is to use an iterative solution algorithm like Conjugate Gradients, Gauß-Seidel, Multigrid Methods, etc. This paper will focus on iterative solvers and the needed storage techniques.

The Institute of Computer Science in Civil Engineering of the University of Hannover, Germany has developed a finite element method based on general cells. The paper will give a short introduction to these cells. This results in general in a finite element mesh which contains a lot of differently shaped cells. Each node is typically connected to three other nodes, and leads to the idea to use band matrices. Band matrices are a special form of sparse matrices. The paper will show that band structure is not needed for this kind of Problem.

There is a new flexibility on our cells, but efficient solvers of the linear equations are recommended. The information of the problem needs to be stored in matrices. The computing power has increased very rapidly over the last years, but even today the limits are reached very fast when wrong or ineffective algorithms are used. In the future, we expect more and more time intensive operation will be done on parallel computers. The paper will end with an case study of the shallow water equation.

1 Introduction

Civil engineers need numerical methods to compute complex building structures, the automobile industry use simulation modells to design new cars and crash tests, and in fluid mechanics the simulation of transportation problems support a better understanding of natural processes. Engineering problems like these can be described with partial differential equations (PDE). Unfortunately, only in rare cases an analytic solution of these PDEs exists. This is why numeric methods are used to generate an approximate solution. Typical numeric methods are the Finite Element Method (FEM), the Finite Volume Method (FVM), and the Finite Differential Method (FDM) [Ka04]. An essential task of these numeric methods is to solve large algebraic systems of equations with thousands of rows and columns.

At the Institute of Computer Science in Civil Engineering of the University in Hannover, Germany a new method was developed, which differs significantly from the common FEM. The finite elements used in this new method are not necessarily triangles or quadrangles. Any convex, non convex, or parametric cell is a possible basis for the finite elements. This is new and leads to the question of what kind of matrix will arise form these new finite elements and decompositions. They may be band structured or sparse.

A matrix A of the order n is said to be sparse if it contains a small number of nonzero elements compared to its size. This paper will show how to develop the matrix A using FEM and how the resulting linear system of the following form can be solved.

$$Ax = b$$

Sparse systems arise in many contexts – fluid dynamics, structural engineering, linear programming, economic models, and electrical circuits, just to name a few. In the last two decades much research was done on sparse systems. This paper will give an overview over what can be done very easily without knowing too much of the material but be sensible enough to use the right black box. The simplest way for decomposition we know is the LU decomposition. The LU decomposition is done with an effort proportional to n^3. In this case the matrix A is written as follows:

$$A = LU$$

L is lower and U is upper triangular. The system $Ax = b$ can then be solved in two stages

1. $Ly = b$
2. $Ux = y$

Since L and U are triangular, the system can be solved efficiently by standard algorithms. Two problems arise when using this kind of decomposition. For large algebraic systems of equations computation time and storage space are critical.

For example, if it takes 10 seconds to compute an equation with 1.000 unknowns it will last 2,77 hours to compute an equation with 10.000 unknowns. This ratio shows that the method is not very effective. Hence, other ways to solve this equation have to be found. Possible algorithms will be shown in section 3.

The FEM matrices used by the Institute of Computer Science in Civil Engineering of the University in Hannover, Germany are often very large. Matrices with over 100.000 unknowns are not unusual. If such a matrix was dense 10.000.000.000 floting point numbers would have to be stored. To compute these matrices they would have to be stored on physical storage medium. A matrix of the order 10.000 requires eight hundred megabytes, which is not typical for a personal computer or workstation. Thus, the size of the matrices would be a problem. Therefore, we use sparse matrix storage techniques to reduce memory space and computing time significantly. How and why this is possible will be shown in the following section 4.

2 Building the matrices using the FEM

Everything begins with the problem which is to be solved. Regularly, this is a PDE (Partial Differential Equation) of the following form

$$\Delta U = f \ .$$

This PDE is fulfilled within the domain of investigation Ω. It describes a function $U(\vec{x})$, which is unknown and a known function f on the right side. The solution is defined in continuous space. The Finite Element Method (FEM) transforms the problem to a discrete space. To find an approximation $\widetilde{U}$ for the unknown solution U it is assumed that $\widetilde{U}$ has the following form

$$\widetilde{U}(\vec{x}) \cong \sum_{i=1}^{n} u_i \cdot \phi_i(\vec{x})$$

The continuous area is split in finite elements. All nodes of this area have a value u_i, which is unknown. They are connected with a basic function $\phi_i(\vec{x})$, which is known, because it can be chosen by the developer of the FEM. After a few transformation steps [Pi02] the linear equation has the following form:

$$\frac{1}{\Delta t}\overline{M}_k \vec{c}_k^{(t+\Delta t)} = \left(\frac{1}{\Delta t}\overline{M}_k - K\overline{D}_k\right)\vec{c}_k^{(t)}$$

$$m_{ij} = \int_{\Omega_k} \phi_{k(i)}\phi_{k(j)}\, d\Omega \qquad d_{ij=}\int_{\Omega_k} \frac{\partial\phi_{k(i)}}{\partial x}\frac{\phi_{k(j)}}{\partial x}d\Omega + \int_{\Omega_k} \frac{\partial\phi_{k(i)}}{\partial y}\frac{\phi_{k(j)}}{\partial y}d\Omega$$

For each node chosen to represent the area the matrix gets one row and one column, therefore a quadratic matrix is build. It can be shown that the matrix is symmetric and positive definite.

2.1 Element forms

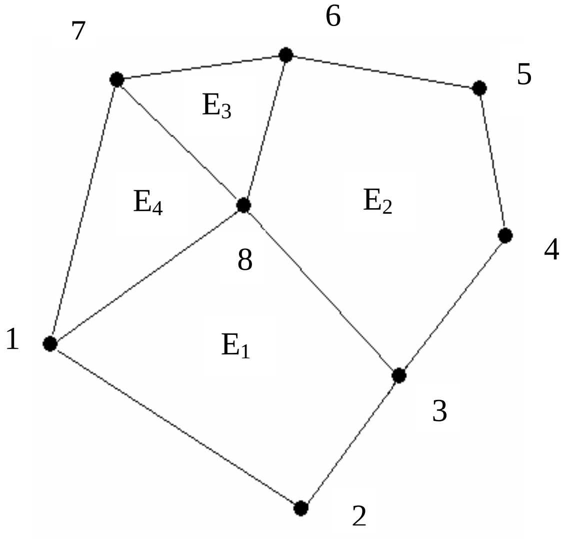
The geometric basis of our finite elements are convex, non convex, or parametric polytrope cells. They do not need to be triangles or quadrangles. The decomposition can be formed through n- dimensional convex cells, which are described by their nodes. Two nodes build the facet for the 2- dimensional elements and on average one node is connected to 3.08 other nodes. Entries in the column n of the matrix A are generated for each node, which belongs to an element connected to E_n. The reason for the complexity of how to get a band matrix can be shown here. The numbering of the nodes is chosen by the developer. The problem is that only with the optimal numbering you get the optimal band matrix; otherwise it is just a sparse matrix. It is important for the band matrix that the range between all nodes connected with other nodes (via the elements) is short for all nodes. An algorithm to build a band matrix through renumbering is very extensive. This paper will show that it is not necessary to do so if an iterative solver is used.

2.2 Nonzero numbers in the matrix from FEM

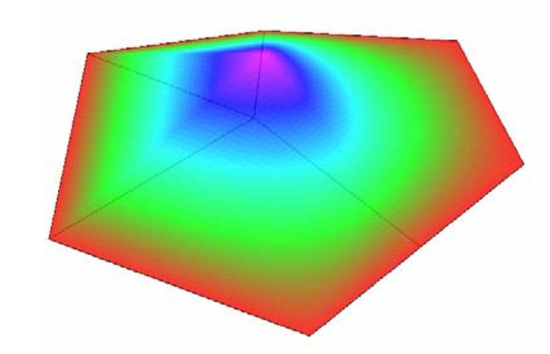
As an example following PDE $\triangle U = f$ is given the on the domain Ω in particular $\mathbb{R}^2$. In most cases an analytic solution does not exit. The equation for this example, which has to be solved, has the following form:

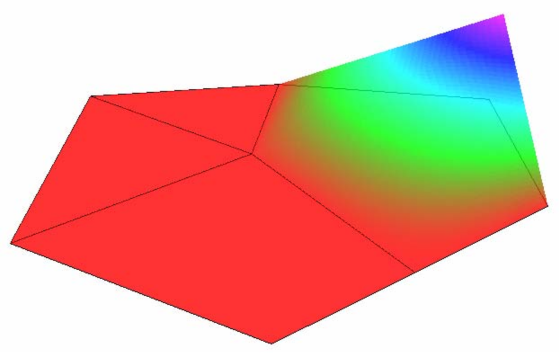
$$\frac{\partial c(x,y,t)}{\partial t} = K\left(\frac{\partial^2 c(x,y,t)}{\partial x^2} + \frac{\partial^2 c(x,y,t)}{\partial y^2}\right)$$

The equation describes the diffusive flow of a concentration c within a fluid over time. This is the simplest model concept possible, because no advection is present. Transportation of the concentration is done through the Brownian movement (which exists wherever temperature is above zero Kelvin). The basis functions are constructed in a way that they are nonzero only in the elements to which they belong.



The picture shows a selected area of the domain $\mathbb{R}^2$. It is just a short part of the whole net and shows all elements connected to the node 8. Around this part are arbitrary more elements. In this case four elements are connected to centre node 8. As described above each node has its own basis function. The basis functions are equal to one at the nodes to which they belong and zero on all other nodes. The basis function is nonzero within the connected elements E_k. In this case the basis function $\phi_8(\vec{x})$ is nonzero within the elements E_1, E_2, E_3 and E_4.





In these pictures you can see the basis function for node 8 and the basis function for node 5. You got an entry in the matrix A at row 8 column 5, because both function are defined in element E_2 and therefore the integral of the product of these function is nonzero.

3 Algorithms to solve sparse linear systems

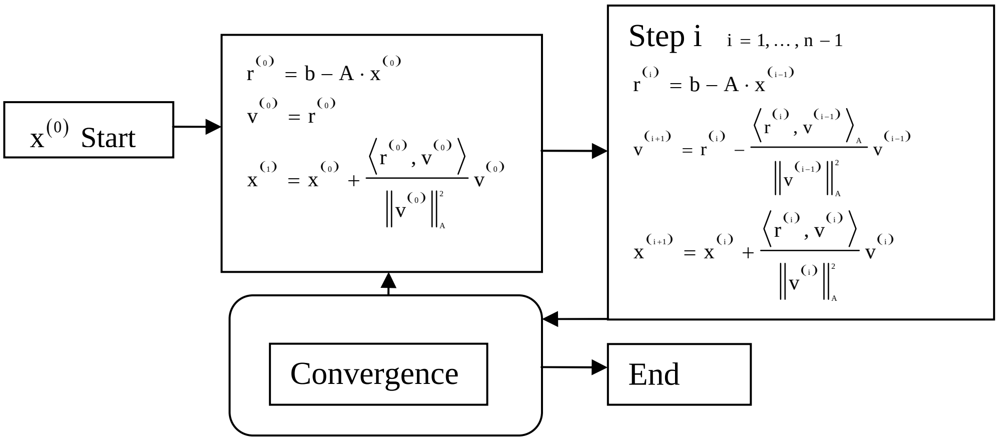
Heimsund [Hei04] developed a Sparse Matrix Collection. The Sparse Matrix Collection offers general, unstructured sparse matrices with iterative solvers and preconditioners. It provides algorithms for solving large matrix problems arising from PDE problems. It features iterative solvers from Templates [Tem94]. These include

- Conjugate Gradients
- Generalized Minimal Residuals (GMRES), restarted version
- Bi Conjugate Gradients
- Quasi Minimal Residuals
- Conjugate Gradients Squared
- Bi Conjugate Gradients Stabilized
- Chebyshev

All these solvers are nonstationary iterative methods. They differ from stationary iterativ solvers like Jakobi, Gauß Seidel, SOR, and SSOR since the computations involve information which changes at each iteration. Typically, constants are computed by taking inner products of residuals or other vectors arising from the iterative method. The solvers used in the Sparse Matrix Collection are best discribed in [Tem94] and will be discussed in the following subsection.

3.1 Conjugate Gradient Methode

The Conjugate Gradient method (CG) derives its name from the fact that it generates a sequence of conjugate or orthogonal vectors. These vectors are the residuals of the iterates. They are also the gradients of a quadratic functional, the minimization of which is equivalent to solving the linear system. CG is an extremely effective method if the coefficient matrix is symmetric positive definite since storage for only a limited number of vectors is required.


$x^{(0)}$ Start
$r^{(0)} = b - A \cdot x^{(0)}$
$v^{(0)} = r^{(0)}$
$x^{(1)} = x^{(0)} + \frac{\langle r^{(0)}, v^{(0)} \rangle}{\| v^{(0)} \|_A^2} v^{(0)}$
Step i $i = 1, \dots, n-1$
$r^{(i)} = b - A \cdot x^{(i-1)}$
$v^{(i+1)} = r^{(i)} - \frac{\langle r^{(i)}, v^{(i-1)} \rangle_A}{\| v^{(i-1)} \|_A^2} v^{(i-1)}$
$x^{(i+1)} = x^{(i)} + \frac{\langle r^{(i)}, v^{(i)} \rangle}{\| v^{(i)} \|_A^2} v^{(i)}$
Convergence
End

3.2 Minimal Residuals (MINRES)

The Minimal Residuals method (MINRES) is a computation alternative for CG. It is used for coefficient matrices which are symmetric but possible indefinite. It will generate the same solution as CG if the coefficient matrix is symmetric positive definite.

3.3 GEMRES

The Generalized Minimal Residual method (GEMRES) computes a sequence of orthogonal vectors (like MINRES) and combines these through a least-squares solve and update. However, unlike MINRES (and CG) it requires storing the whole sequence, so that a large amount of storage is needed. For this reason, restarted versions of this method are used. In restarted versions, computation and storage costs are limited by specifying a fixed number of vectors to be generated. This method is useful for general non symmetric matrices.

3.4 BiConjugate Gradient

The Conjugate Gradient method is not suitable for non symmetric systems because the residual vectors cannot be made orthogonal with short recurrences. The GMRES method retrains orthogonality of the residuals by using long recurrences, at the cost of a larger storage demand. The BiConjugate Gradient method (BiCG) takes another approach, replacing the orthogonal sequence of residuals by two mutually orthogonal sequences, at the price of no longer providing a minimization.

3.5 Quasi Minimal Residual (QMR)

The Quasi-Minimal Residual method (QMR) applies a least-squares solve and update to the BiCG residuals, thereby smoothing out the irregular convergence behaviour of BiCG, which may lead to more reliable approximations. In full glory, it has a look ahead strategy built in, which avoids the BiCG breakdown. Even without look ahead, QMR largely avoids breakdowns, which can occur in BiCG. On the other hand, it does not effect a true minimization of either the error or the residual, and while it converges smoothly, it often does not improve on the BiCG in terms of the number of iterations steps.

3.6 CGS

The Conjugate Gradient Squared method is a variant of BiCG, that applies the updating operations for the A-sequence and the A^T -sequence both to the same vectors. Ideally, this would double the convergence rate, but in practise convergence may be much more irregular than for BiCG. Sometimes, this may lead to unreliable results. A practical advantage of the method is that it does not need the multiplications with the transpose of the coefficient matrix.

3.7 BiCG-stab

The BiConjugate Gradient Stabilized method (BiCG-stab) is a variant of BiCG, like CGS, but using different updates for the A^T -sequence in order to obtain smoother convergence than CGS.

3.8 Chebyshev Iteration

The Chebyshev Iteration recursively determines polynomial with coefficients chosen to minimize the norm of the residual in a min-max sense. The coefficient matrix must be positive definite and knowledge of the extremal eigenwert is required. This method has the advantage of requiring no inner product.

These methods provide a lot of iterative solvers, which can be used to solve linear systems. Since our matrices arise from FEM and are in most cases regular positive definite or at least symmetric most of these methods can be used. BiCG-stab is a bit slower than the other computation methods but it is the most reliable.

4 Concept of matrix storage

The aim of a Sparse Matrix package is to achieve a better performance. One way to realize this is to use a fast solver. The other important impact on performance is caused by the storage technique. There are several possibilities to store a matrix. The key is not to store unnecessary data. Each technique has its advantages and disadvantages. The easiest possible storage scheme records all values in a row. If it is known that the matrix is symmetric no more information is needed. For dense matrices this is the best technique. For sparse matrix it is not, because all zero numbers would be stored and thus consume the short-handed storage medium.

Dense matrix storage increases quadratic with the numbers of nodes This is because for each node chosen to represent the area, the matrix grows by one row and one column. An example: an area represented by 1000 nodes needs a matrix for which 1.000.000 double values have to be stored. This is the point where the first huge increasing factor can be found, because storage medium increases linear with sparse matrix storage techniques (number of nodes multiplied with average nonzero numbers per row). The question is how this can be done most economically.

Special forms of sparse storage techniques are band storage or skyline techniques. The disadvantages of these techniques are that within the band zeros will be stored. A special numbering for these nodes is needed to find an optimal or near optimal solution to keep the band of the matrix short. But an optimal numbering is a complex and time intensive operation. An advantage is that there is only a fill in within the band while building the inverse.

4.1 Row oriented sparse matrix storage

Sparse matrix storage can be done with tree vectors. Using the row oriented method the first vector (data vector) is filled with all non zero values |g|, beginning with the fist row’s first entry and ending the last row’s last entry (n rows). With this kind of storage all information of the position in the matrix is lost. To keep them you need two more vectors. The second vector has the same size as the first vector and is filled with the number of the column from the associated value (column indices). The third vector is smaller; its size is the number of rows plus one (row pointer). It gives the information at which positions in Vector one (two) the rows begin.

$$A = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 \\ 3 & 9 & 0 & 0 & 9 \\ 0 & 7 & 8 & 7 & 0 \\ 3 & 0 & 8 & 7 & 5 \\ 0 & 8 & 0 & 9 & 9 \end{pmatrix}$$

Data	10	-2	3	9	9	7	8	7	3	8	7	5	8	9	9
Column Indices	0	4	0	1	4	1	2	3	0	2	3	4	1	3	4

Row Pointer	0	2	5	8	12	15

There are

$$2*g+n+1$$

values to be stored. Clearly, it is not an effective method if the matrix is filled with nonzero variables.

$$2*n*n+n+1$$

In this case more than twice the storage medium is needed. But storage medium will be saved if the following condition is fulfilled:

$$2*g+n+1<n^2.$$

This is the basis for sparse matrix storage. In this way the medium needed is reduced to a minimum. To solve the linear equation an algorithm is needed with an optimal usage of the storage medium and high effectiveness in computing.

4.2 Column oriented sparse matrix storage

The column oriented sparse matrix storage is also based on three vectors. This time the first vector is filled with all non zero values, beginning with the fist column's first entry and ending the last column's last entry. The second and third vector are filled following the same method as above.

Data	10	3	3	9	7	8	8	8	7	7	9	-2	9	5	9
Row Indices	0	1	3	1	2	4	2	3	2	3	4	0	1	3	4

Column Pointer	0	3	6	8	11	15

The row oriented storage is optimized for the multiplication $A \cdot x$ which is optimal for pre-multiplication. The column oriented sparse matrix storage is optimized for $x^T \cdot A$ and $A^T \cdot x$, because it has a fast column access. If both accesses are regularly used within an algorithm, it leads to the idea to store the matrix in two ways. This doubles the memory usage but reduces the computing time needed dramatically. These approaches seem to be convenient for the multi grid method which is topic of further research.

5 Case Study

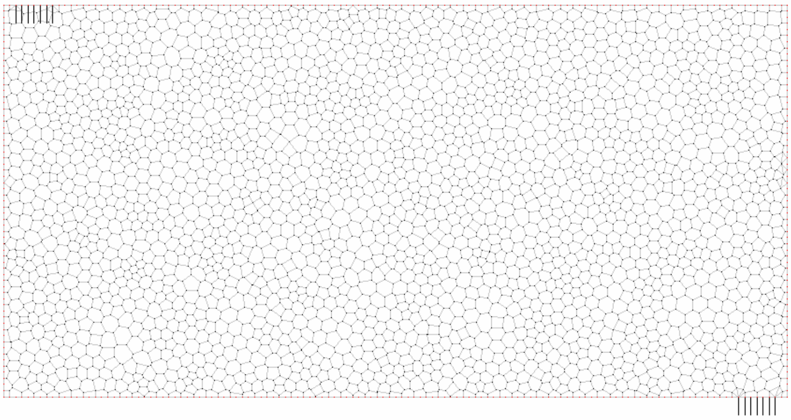
This Case Study is based on a FEM example from fluid mechanics. The approximation of the instantaneous simplified shallow water equation in a rectangular basin is used as a test case.

$$\frac{\partial U_x}{\partial t} = -U_x \frac{\partial U_x}{\partial x} - U_y \frac{\partial U_x}{\partial y} - g \frac{\partial \eta}{\partial x}$$

$$\frac{\partial U_y}{\partial t} = -U_x \frac{\partial U_y}{\partial x} - U_y \frac{\partial U_y}{\partial y} - g \frac{\partial \eta}{\partial x}$$

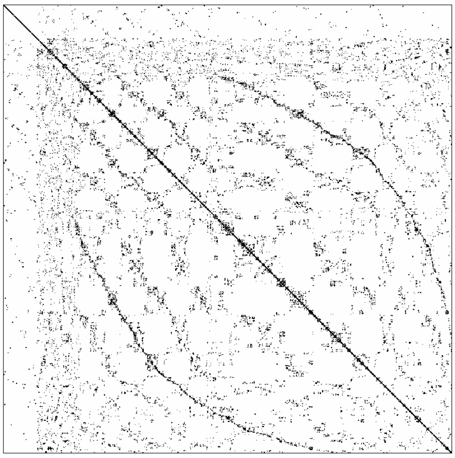
$$\frac{\partial \eta}{\partial t} = -\frac{\partial U_x (\eta - h)}{\partial x} - \frac{\partial U_y (\eta - h)}{\partial y}$$

η is the instantaneous water surface amplitude, U_x and U_y representing the velocities in x- and y-direction, h is the mean water depth.

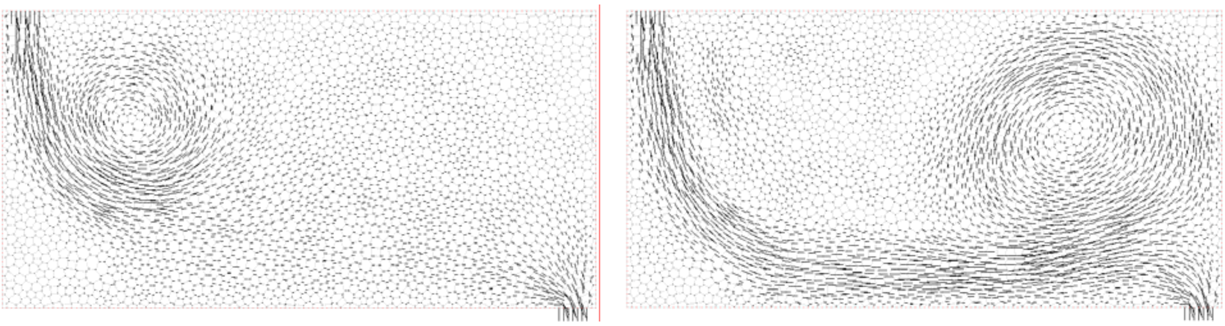


This example shows [Schw04] a net where water is filled in steadily from the entrance in the upper left corner and moves into the lower right corner. Obviously, this net does not consist of triangles or quadrangles, but it is a decomposition of any convex polytope cells. The net was generated by a voronoi decomposition and voronoi cells are used as the basis for the FEM.

The numeric approximation was done with stabilized FEM. The structure of the matrix resulting from this stabilized method changes marginally. The resulting matrix has the following form.



What can be seen here? The matrix for this decomposition of the basin has 15.201 nodes and 572561 nonzero entries, which is an average of 37.66 entries per row. Why this matrix is symmetric was explained above. Significant is the small band on the left side. These are the border nodes of the basin. They have the smallest numbers. This was done for reasons of easier access to set boundary conditions. These nodes have fewer neighbours than the others. Obvious is the fact that there is no band structure, the entries are distributed sparse all over the rows of the matrix. Further identification of structures is topic of current research.



These pictures show the results of the FEM simulation, which was done with n-dimensional convex polytope cells. The computation of the linear system was done with the sparse matrix pakage of Heimsund. After 8 minutes a whirl appeared, which moved to the right side (150 minutes) to build a quasi stationary whirl. The perspective is from above. This is part the current research at the Institute of Computer Science in Civil Engineering in Hannover, Germany.

6 Summary and Outlook

The paper has shown that, with the right storage technique and the right iterative solver for sparse filled matrices efficient and proper solutions are generated. The basis for the new FEM is a decomposition of any convex, non convex, or parametric polytrope cells. The matrix resulting from this new FEM is sparse and not band structured. A few years ago it was popular to generate an optimal numbering to obtain a band matrix, but as this paper has shown, this is not necessary. The access via these compact storage techniques is optimal hence there is no zero entry used and therefore no unnecessary computation is done. The benefit of the band matrix is that its inverse matrix is easily build in comparison with the sparse matrix (fill in). But the advantage of the iterative algorithms is that no inverse matrix has to be build. All operations are matrix vector products and straight forward.

The shown iterative solvers for sparse filled matrices were used within the scope of the shallow water equation. The new FEM in combination with the iterative sparse solvers build the basis for further research.

7 Acknowledgement

We want to thank our colleagues and especially Axel Schwöppe for his assistance and encouragement with the compilation of the shown case study. The case study is part of his current research.

8 Referenzes

[Hei04] Heimsund, B.-O.; SMT - *A sparse matrix toolkit for Java*; Centre of Integrated Petroleum Resarch, Bergen, Norway; http://www.mi.uib.no/~bjornoh/mtj

[Ka04] Kaapke; *Voronoi-based finite volume method for transport problems*; The ICCCBE-X-Conference Proceedings publication; Bauhaus-Universität Weimar. Germany; 2004

[Mat04] Sparse Matrix Algorithms Research at the University of Florida http://www.cise.ufl.edu/research/sparse

[Pi02] Pick, Tobias; *Finite Element Approximation auf der Basis von Zellen*; Institut für Bauinformatik Uni. Hannover; Diplomarbeit 2002

[Schw02] Schwöppe, A. | Milbradt, P.; *A Class of Convex, Polygonal Bounded Finite Elements*; Paper für Fifth World Congress on Mechanics 2002 (Wien)

[Schw04] Schwöppe, A; *Finite Element Approximation auf der Basis geomtrischer Zellen*; Dissertation Hannover;2004; noch unveröffentlich

[Tem94] Richard Barrett, et al; *Templates fot the Solution of Linear Systems: Building Blocks for Iterative Methods*; SIAM; 1994; http://www.netlib.org/templates

[Wei90] J. Weissinger; *Spärlich Besetzte Gleichungssysteme*; BI Wissenschaftsverlag; Mannheim/Wien/Zürich; 1990

[Zur86] Zurmühl, Falk; *Matrizen und ihre Anwendungen*, Teil 2: Numerische Methoden; Fünfte Auflage; Springer Verlag; 1986