#6119

Solution methods for the air balance in multizone buildings

Magnus K. Herrlin*

Department for Building Services Engineering, Royal Institute of Technology, Stockholm (Sweden)

Francis Allard

Centre de Thermique, INSA de Lyon, URA CNRS 1372, Bat 307, INSA, F-69621 Cédex, Villeurbanne (France)

(Received March, 1990; accepted January, 1991; revised paper received February, 1991)

Abstract

Air infiltration programs establish infiltration and ventilation rates in a building by the solution of a nonlinear system of equations. This paper discusses various modifications of the Newton–Raphson method and the special characteristics of the resulting linear system of equations. The aim of the paper is selecting efficient and robust methods to solve the system of equations representing the airflow distribution in multizone buildings.

Taking into account the special characteristics of the system of equations, we recommend a skyline-Cholesky's method for the linear solver. This method can be used in a general way for these problems and it appears to be very efficient in avoiding unnecessary operations on zero elements.

The choice of an under-relaxation process to ensure the convergence and efficiency is not obvious. Two different methods are used to find the under-relaxation coefficients. The extrapolated method uses two steps of fixed-point iteration to calculate the starting value for the next step of the process. The optimized method uses a search routine where the direction of search is determined by Newton–Raphson and the distance of movement is determined by minimization of a related one-dimensional function. In studying the methods on the same sample of test cases, we found that they are very similar regarding CPU time. However, the optimized method appears to be safer and more efficient in the resolution of the nonlinear system. We, therefore, recommend using the optimized method for the nonlinear solver.

1. Introduction

Infiltration programs establish the infiltration and ventilation rates in a building by the solution of a nonlinear system of equations. An iterative method can be used in which a linear system of equations is solved in each step of the process.

The Newton–Raphson method is often used for this kind of problem and usually it works satisfactorily although the convergence is sometimes slow. The approach here is to use Newton–Raphson but to modify the method in exceptional cases.

This paper discusses various modifications of the Newton–Raphson method and the special characteristics of the resulting linear system of equations. Two modifications are outlined, coded, and timed. Some direct linear methods are described and several of these are also coded and timed.

2. System of equations

The network consists of pressure nodes and links. A mass flow balance must exist in each node as described by the following flow balance equation,

$$f(p)=\sum_i \dot{m}_i=0 \qquad (1)$$

and in vector form for all nodes,

$$\boldsymbol{f}(\boldsymbol{p})=0 \qquad (2)$$

An appropriate function describes the flow rate as a function of pressure difference for each link. Nonlinear expressions of the following type are the predominant,

$$\dot{m}=C(\Delta p)^n \qquad (3)$$

or more precisely,

$$\dot{m}=C\ sign(\Delta p)|\Delta p|^n \qquad (4)$$

where $0.5 \leqslant n \leqslant 1.0$.

*Visiting Scientist with the Energy Performance of Buildings Group, Lawrence Berkeley Laboratory, Berkeley, USA.

0378-7788/92/$5.00

Equation (2) is obviously a nonlinear system of equations.

The following analysis is, in general, not limited to this particular function although airflow functions are always nonlinear. A nonlinear system of equations is usually solved with an iterative method that uses a linear process in each step.

The system has the following general structure for two zones (Fig. 1), four fixed pressures, and five links,

$$f_a(\boldsymbol{p}) = C_{a1}(p_a - p_1)^{n_{a1}} + C_{a2}(p_a - p_2)^{n_{a2}} + C_{ab}(p_a - p_b)^{n_{ab}} = 0 \quad (5)$$

$$f_b(\boldsymbol{p}) = C_{ab}(p_b - p_a)^{n_{ab}} + C_{b4}(p_b - p_4)^{n_{b4}} + C_{b5}(p_b - p_5)^{n_{b5}} = 0 \quad (6)$$

The equations are formulated systematically. A flow out from a zone is positive. Again the system can be written more conveniently as vectors,

$$\boldsymbol{f}(\boldsymbol{p}) = 0 \quad (7)$$

The Newton–Raphson method, which is the most fundamental and widely used, finds the next approximation in the one-dimensional case through the following iteration function $h(p_n)$,

$$p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)} \quad (8)$$

where f' is the derivative of f.

This disregards everything except the constant and linear terms in the Taylor expansion. A multi-dimensional case is similar,

$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_n - \frac{\boldsymbol{f}(\boldsymbol{p}_n)}{\mathbf{J}(\boldsymbol{p}_n)} \quad (9)$$

where **J** is the Jacobian matrix.

The Jacobian matrix is obviously similar to f' in the one-dimensional case; the matrix consists of the partial derivatives of all the flow balance equations $\boldsymbol{f}$ regarding all pressures $\boldsymbol{p}$.

Equation (9) can be rewritten in the following way,

$$\mathbf{J}(\boldsymbol{p}_n)\boldsymbol{corr} = -\boldsymbol{f}(\boldsymbol{p}_n) \quad (10)$$

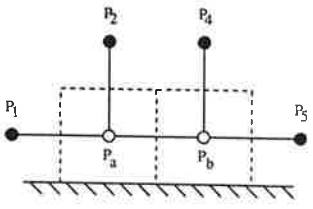


Fig. 1. A two-zone building.

where ***corr*** is the Newton–Raphson corrections of the pressures. Note that this system of equations is linear. From now on, in order to simplify, the $(\boldsymbol{p})$ in $\mathbf{J}(\boldsymbol{p})$ and $\boldsymbol{f}(\boldsymbol{p})$ will occasionally be dropped.

3. Special characteristics of the Jacobian matrix

Some of the most frequently cited books in Linear Algebra have been reviewed for the preparation of this Section [1–6].

The Jacobian matrix has, as a rule, two important features in airflow problems – it is symmetric and positive definite. To prove the latter it is necessary to look at determinants and eigenvalues.

The Jacobian matrix in our example is,

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f_a}{\partial p_a} & \dfrac{\partial f_a}{\partial p_b} \\[2ex] \dfrac{\partial f_b}{\partial p_a} & \dfrac{\partial f_b}{\partial p_b} \end{bmatrix} \quad (11)$$

Remembering the limits of n the derivatives of the first row are,

$$\frac{\partial f_a}{\partial p_a} = C_{a1}n_{a1}|p_a - p_1|^{n_{a1}-1} + C_{a2}n_{a2}|p_a - p_2|^{n_{a2}-1} + C_{ab}n_{ab}|p_a - p_b|^{n_{ab}-1} \quad (12)$$

$$\frac{\partial f_a}{\partial p_b} = -C_{ab}n_{ab}|p_a - p_b|^{n_{ab}-1} \quad (13)$$

where the first equation represents one diagonal element and the second equation one non-diagonal element in the matrix, which means that the elements of the matrix are negative except those on the diagonal.

3.1. *Symmetry*

A symmetric matrix **A** is a matrix with the property $A(i, j) = A(j, i)$ or $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$. In our case this is the same as saying that the partial derivative of zone i with reference to zone j is equal to the partial derivative of zone j with reference to zone i.

Formulating the equations in the same way as in the preceding Section we have,

$$J(i, j) = \frac{\partial}{\partial p_j} C_{ij}(p_i - p_j)^{n_{ij}} = -C_{ij}n_{ij}|p_i - p_j|^{n_{ij}-1} \quad (14)$$

$$J(j,\ i)=\frac{\partial}{\partial p_i}\ C_{ij}(p_j-p_i)^{n_{ij}}=-C_{ij}n_{ij}|p_j-p_i|^{n_{ij}-1} \tag{15}$$

These derivatives are in general not defined for the case $p_i=p_j$. Except in this exceptional case the Jacobian matrix is symmetric. Non-defined derivatives are a potential problem but one which can be avoided in several ways. The airflow function can be linearized between zero and a certain very small pressure difference which also is physically consistent with flow behaviour. It is also possible to disregard every link with a very small pressure difference which will not change the result significantly.

If no leak is defined between two zones i and j, the element $J(i,\ j)=0$ because the flow balance equation for zone i does not include zone j and, in consequence, the derivative of i with reference to zone j is 0. This is also the case when a fixed flow is defined between two zones.

Again, the elements $J(i,\ i)$ are all positive which follows from the way we formulated our equations,

$$J(i,\ i)=\sum_{j\epsilon i}\frac{\partial}{\partial p_i}\ C_{ij}(p_i-p_j)^{n_{ij}}=\sum_{j\epsilon i}C_{ij}n_{ij}|p_i-p_j|^{n_{ij}-1} \tag{16}$$

where j are zones or fixed pressures with a connection to zone i. Equations (14), (15), and (16) give directly,

$$J(i,\ i)\geqslant\sum_{j,j\neq i}|J(i,j)| \tag{17}$$

with equality only when zone i does not have any connection to a zone with fixed pressure.

The last equation is important when we want to show that the Jacobian matrix is positive definite.

3.2. *Positive definite*

Gerschgorin's theorem can be used to locate the eigenvalues. This theorem states that every eigenvalue of the matrix lies in at least one of the circular discs, in the complex domain, with centres $J(i,\ i)$ and radii $\Sigma|J(i,\ j)|$,

$$|z-J(i,\ i)|\leqslant\sum_{j,j\neq i}|J(i,\ j)| \tag{18}$$

We know that,

$$J(i,\ i)\geqslant\sum_{j,j\neq i}|J(i,j)|\geqslant|z-J(i,\ i)| \tag{19}$$

that is, all values on the disc are positive or zero, i.e., all eigenvalues $\geqslant 0$. The matrix is said to be positive semi-definite. A necessary and sufficient condition for a matrix to be positive definite is that all the eigenvalues of the matrix should be >0.

Every general non-singular matrix can be transformed into a triangular matrix and every symmetric matrix into a diagonal one. These similarity transformations play an important role in the computation of eigenvalues because they leave the eigenvalues of a matrix unchanged. The diagonal elements in these matrices are the eigenvalues of the matrices. This follows from the definition of eigenvalues and expansion by minors.

The definition of eigenvalues **e** of a matrix **A** is,

$$\det(\mathbf{A}-\mathbf{e}\mathbf{I})=0 \tag{20}$$

where **I** is the unity matrix.

It follows directly from the definition of determinants that the value of the determinant for a diagonal or triangular matrix is the product of the diagonal elements, i.e., the determinant of a matrix **J** is the product of its eigenvalues, i.e., $\det(\mathbf{J})=\Pi$ eigenvalues.

$\det(\mathbf{J})=0$ means that the matrix is singular by definition. No unique solution exists, possibly due to modelling or round-off errors. If one or several nodes are completely isolated from other nodes and constant pressures, the system of equations is singular.

The conclusion is that the Jacobian matrix must be positive definite except when no unique solution exists.

A necessary but not sufficient condition for $\det(\mathbf{J})=0$ is,

$$J(i,\ i)=\sum_{j,j\neq i}|J(i,\ j)| \tag{21}$$

for one or several rows in the matrix. Gerschgorin's theorem states merely that the eigenvalues must be on the discs — not exactly where.

Any symmetric positive definite matrix **A** has a unique decomposition in the form,

$$\mathbf{A}=\mathbf{G}\mathbf{G}^{\mathrm{T}} \tag{22}$$

where **G** is a triangular matrix. The consequences of this will be discussed in the next Section.

4. Linear solvers

This Section is mainly based on refs. 1 and 7.

A direct method, compared with an iterative method, solves a system of equations in a certain finite number of steps. Direct methods are usually the most efficient methods for small- to medium-size full systems. If special characteristics of the matrix are taken into consideration direct methods can also be efficient for larger systems. Very large and spare systems are best solved with iterative methods. The number of operations using a direct method is proportional to N^3 and, using an iterative method, N^2 where N is the order of the matrix. It is easy to show that if more than N iterations are needed the iterative method is not appropriate. Obviously the iterative methods become more attractive for a larger matrix. However these methods will not be discussed here.

Our concern is with matrices solved by Gaussian elimination, LU-factorization, Cholesky's method, and the inverse. This includes studies of general, symmetric, and positive definite matrices. Band and skyline methods are used to improve the solvers.

The number of operations required for some methods has been estimated with finite sums. "Operation" is defined here as one multiplication and addition – a useful measure.

4.1. Gaussian elimination with back substitution

A triangular matrix can easily be solved with backward or forward substitution depending on whether the matrix has an upper or lower triangular form. A general matrix can therefore be solved if it is first reduced to a triangular form.

The Gaussian elimination method reduces a general matrix to a triangular form. The technique is to eliminate $corr_1$ of the system $\mathbf{J}\ \boldsymbol{corr} = -\boldsymbol{f}$ from the last $N-1$ equations by subtracting multiples m of the first equation. Thereafter $corr_2$ is eliminated in the same way from the last $N-2$ equations. This procedure is repeated until we have an upper triangular form with elements u. The diagonal elements are called pivotal elements.

The compact scheme for Gaussian elimination can be written,

$$u_{kj}=j_{kj}-\sum_{p=1}^{k-1} m_{kp}u_{pj} \quad k=1,\ \ldots,N \quad j=k,\ k+1,\ldots,N. \tag{23}$$

$$m_{ik}=\frac{j_{ik}-\sum\limits_{p=1}^{k-1} m_{ip}u_{pk}}{u_{kk}} \qquad i=k+1,\ \ldots,N. \tag{24}$$

where j, m, and u are the elements in $\mathbf{J}$, $\mathbf{L}$, and $\mathbf{U}$ respectively.

The backward substitution can be written in compact form,

$$corr_i=\frac{-f_i-\sum\limits_{k=i+1}^{N} u_{ik}corr_k}{u_{ii}} \qquad i=N,N-1,\ \ldots,\ 1 \tag{25}$$

Forward substitution is done in a similar way.

The total number of operations for large N and one right-hand side is approximately $N^3/3+N^2/2$.

4.2. Pivoting

If a pivotal element becomes zero the Gaussian method obviously does not work since division by zero is not a defined mathematical operation. To solve the problem we can interchange rows or columns. In this way it is possible to find a non-zero pivotal element as long as the matrix is non-singular. Any non-singular system can thus be reduced by Gaussian elimination with pivoting. Pivoting requires a large number of logical operations.

4.3. LU-factorization

The LU-factorization decomposes a matrix $\mathbf{A}$ into a lower $\mathbf{L}$ and an upper $\mathbf{U}$ triangular matrix,

$$\mathbf{A}=\mathbf{LU} \tag{26}$$

Then the system $\mathbf{Ax}=\mathbf{b}$ is equivalent to $\mathbf{LUx}=\mathbf{b}$ which can be solved as two triangular systems $\mathbf{Ly}=\mathbf{b}$ and $\mathbf{Ux}=\mathbf{y}$. For every non-singular matrix a LU-factorization exists. In fact there is an equivalence between Gaussian elimination and LU-factorization. The elements in $\mathbf{L}$ are the multipliers m and the matrix $\mathbf{U}$ is the upper triangular matrix. Equations (23) and (24) are therefore also valid for LU-decomposition.

The LU-decomposition requires $N^3/3-N^2/2$ operations and the two triangular systems N^2 operations, i.e., the same as Gaussian elimination.

4.4. Cholesky's method

For a positive definite matrix, Gaussian elimination without pivoting is always possible, i.e., no pivots are zero as follows from $\mathbf{e}>0$. We can show that if Gaussian elimination can be carried out without pivoting for a symmetric matrix then there exist transformed elements m and u which form a symmetric matrix. This follows from eqn. (22).

Thus, if the matrix is symmetric positive definite we have only to compute $\mathbf{L}$ and the number of operations is approximately halved to $N^3/6$ compared with the general Gaussian elimination without pivoting. A symmetric positive definite matrix does not require pivoting to control round-off errors.

Equations (23) and (24) are now simplified to,

$$m_{kk}=(j_{kk}-\sum_{p=1}^{k-1}m_{kp}{}^{2})^{0.5}\quad k=1,\ \ldots,N \qquad (27)$$

$$m_{ik}=\frac{j_{ik}-\sum_{p=1}^{k-1}m_{ip}\,m_{kp}}{m_{kk}}\quad i=k+1,\ \ldots,N. \qquad (28)$$

This is Cholesky's method. The first equation calculates the diagonal elements and the second calculates the elements of the lower triangle.

4.5. Band matrices

The non-zero elements in a band matrix are arranged in a diagonal band of width $2M+1$ where M is the half band width.

In Fig. 2, the LU-factorization method has been adjusted to various matrices and a comparison made of the number of operations involved solving the system. The curves have been estimated with finite sums.

The number of operations for the symmetric case is approximately $NM^2/2-M^3/3$. Note that the number of operations is based on the decomposition only – backward and forward substitutions are not included although in this case it is just a fraction of the total number of operations. Administration in the program also is not included so that these curves are very much simplified. Nevertheless we can expect a large reduction in execution time if we take the special characteristics of the matrix into account.

The lower limits of the sums in eqns. (27) and (28) are increased according to the band width. The lower limit will not be 1, but instead where the band starts for the specific row. This explains the reduction in time.

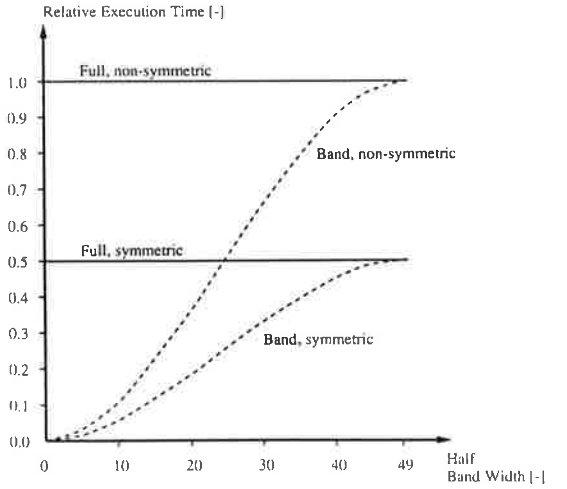


Fig. 2. Relative time for LU-decomposition on different matrices.

4.6. Skyline

The skyline method takes into account variable lengths of non-zero rows and columns in a matrix. All the elements below a non-zero element in the upper triangular part are taken into consideration, these non-zero elements forming a skyline profile. The lower triangular part is treated in a similar way, all elements to the right of a non-zero element being taken into consideration. Zero-elements outside the profile are not considered while zero-elements inside are. In this sense the skyline can be thought of as an irregular band. If there is one zone that has connections to almost all other zones, the best skyline is achieved if this common zone is placed as the last zone.

For a symmetric matrix we need only to handle the lower (or upper) part of the matrix and the diagonal. The skyline is therefore defined by the array of row lengths in the lower part of the matrix.

Again, the lower limits of the sums are changed in eqns. (27) and (28). The lower limit will follow the skyline. This means that the skyline method always requires equal or less number of operations than to consider the matrix as a band matrix. The execution time depends on sparsity and cannot easily be generalized as in the case of the band matrices above. However, some examples are presented in Section 6.3.

4.7. Inverse

The system of equations $\mathbf{J}\,\boldsymbol{corr}=-\boldsymbol{f}$ can be solved directly as $\boldsymbol{corr}=-\mathbf{J}^{-1}\boldsymbol{f}$. This requires the calculation of the inverse of the matrix $\mathbf{J}$ followed by a multiplication.

The calculation of the inverse requires N^3 operations and the multiplication N^2, in total N^3+N^2. This should be compared with the fewer iterations of the preceding methods. The calculation of the inverse is therefore not recommended when solving systems of equations.

Another disadvantage is that the inverse of a band matrix does not in general have a band structure. Gaussian elimination and LR-factorization on the other hand preserve the band structure.

5. Non-linear solvers

This section is based on refs. 8–11.

The Newton–Raphson method can be modified to avoid occasional problems. In principle, most modifications deal with finding an appropriate over- or under-relaxation coefficient λ for eqn. (9),

$$\boldsymbol{p}_{n+1}=\boldsymbol{p}_n-\lambda\frac{\boldsymbol{f}(\boldsymbol{p}_n)}{\mathbf{J}(\boldsymbol{p}_n)} \tag{29}$$

The Newton–Raphson correction corresponds to $\lambda=1.0$.

Our links have a type of flow function that, under certain conditions, creates very slow convergence. If Newton–Raphson converges a particular function and both the first and second derivatives are continuous then it is a second order method, i.e., the asymptotic error constant is 2. The method does not converge at all if the flow balance equation is symmetric and has a flow exponent of 0.5. The tangent of the function has, in this case, the same value in each step of the iteration process and consequently no convergence can be expected.

Two different methods will be discussed that, at times, can accelerate the rate of convergence of Newton–Raphson.

5.1. *Extrapolated relaxation coefficients*

The rate of convergence of Newton–Raphson can sometimes be accelerated. Steffensen iteration consists of two steps of fixed-point iteration and thereafter the use of the three available iterates p_{n-1}, p_n, and p_{n+1} to get the starting value for the next step.

A linear interpolant to the iteration function $h(p)$ in p_{n-1} and p_n, i.e., the secant, should be a good approximation to $h(p)$ in the interval if the slope of $h(p)$ varies little close to the root. Therefore, the iteration function given by the secant can be expected to give a good approximation of the root.

$$p_{extrapolated}=p_{n+1}+\frac{p_{n+1}-p_n}{\left(\dfrac{p_n-p_{n-1}}{p_{n+1}-p_n}\right)-1} \tag{30}$$

The method studied here is closely related to this method. The approach above being applied to each node and each node being treated independently of the other nodes. There are therefore, as many under-relaxation coefficients as there are nodes.

5.2. *Optimized relaxation coefficients*

To determine the root in a one-dimensional case to the function f is the same as determining the minimum of the function F defined as,

$$f=F' \tag{31}$$

where

$$F'=dF/dp \tag{32}$$

To determine the roots in a multidimensional case to the function $\boldsymbol{g}$ is the same as determining the minimum of the scalar function G defined as,

$$\boldsymbol{g}=\left(\frac{\partial \mathrm{G}}{\partial \mathrm{p}_{\mathrm{A}}},\frac{\partial \mathrm{G}}{\partial \mathrm{p}_{\mathrm{B}}},\ldots\right)^{\mathrm{T}} \tag{33}$$

G can be expressed as a function in the variable λ only. This means that $G(\lambda)$ is a one-dimensional function. It can be shown that this function has a unique minimum.

The minimum cannot be determined analytically but the function $G(\lambda)$ can be approximated with a polynomial,

$$P(\lambda)=a+b\lambda+c\lambda^2 \tag{34}$$

which, for example, interpolates $G(0)$, $G(0.5)$, and $G(1.0)$. The minimum, $\lambda_{\min}$, of this polynomial is easily determined and can represent the minimum of G. The next approximation is taken as,

$$\boldsymbol{p}_{n+1}=\boldsymbol{p}_n-\lambda_{min,\,n}\frac{\boldsymbol{f}(\boldsymbol{p}_n)}{\mathbf{J}(\boldsymbol{p}_n)} \tag{35}$$

Note that $\lambda_{\min}$ has to be determined in each iteration step n. This means that in each iteration step a Newton–Raphson solution and three evaluations of G are required.

The function F of a one-dimensional function,

$$f=\sum_j C_j\, sign(p-p_j)|p-p_j|^{n_j} \tag{36}$$

is,

$$F=\sum_j\frac{C_j}{n_j+1}\,|p-p_j|^{n_j+1} \tag{37}$$

where j are zones with fixed pressures and have connections to the zone.

A multidimensional case is similar. The function G of the vector function $\boldsymbol{g}$ is,

$$G=\sum_i\sum_{j\epsilon i}\frac{C_{ij}}{n_{ij}+1}\,|p_i-p_j|^{n_{ij}+1} \tag{38}$$

where j are zones or fixed pressures with a connection to zone i.

The example given in eqns. (5) and (6) is,

$$\begin{aligned}G=&\frac{C_1}{n_1+1}\,|p_a-p_1|^{n_1+1}+\frac{C_2}{n_2+1}\,|p_a-p_2|^{n_2+1}\\&+\frac{C_3}{n_3+1}\,|p_a-p_b|^{n_3+1}+\frac{C_4}{n_4+1}\,|p_b-p_4|^{n_4+1}\\&+\frac{C_5}{n_5+1}\,|p_b-p_5|^{n_5+1}\end{aligned} \tag{39}$$

where p_a and p_b can be expressed in λ.

Determining this function involves about the same amount of work as determining the derivatives. It can be important to use "double precision" when implementing this algorithm as well as in the rest of the solver.

The function G is obviously a sum of non-negative terms. The individual terms are strictly convex functions as long as the matrix is not singular but a sum of strictly convex functions is also a strictly convex function, i.e., G.

A strictly convex function has a unique minimizer. This in turn means that the equation $\boldsymbol{g}=0$ has a unique solution.

6. Timing of solvers

The aim here is to provide an overview of the behaviour of various solvers in typical airflow examples.

The solvers were timed and tested in two programs, AIRNET [12] and MOVECOMP-PC [13], with a special analyzer or profiler, gprof, in the ULTRIX 32 library. The tests were made on a Micro-VAX system. As two programs were used a significant amount of work was done to ensure that all input was identical.

6.1. Chosen networks

Representative networks were selected for the tests which were essentially based on discussions with George Walton at the National Institute of Standards and Technology. The input to AIRNET and MOVECOMP-PC was carefully checked so as to be identical and the output was identical for the examples used.

(1) 02-Network

A 2-node network with three links, according to Fig. 3, is convenient to use in showing the classical difficulties with Newton–Raphson for particular functions. All flow exponents are one-half. The ratios between inner and outer links are varied as 10 000, 1000, 100, 10, 1, 0.1, 0.01, 0.001, and 0.0001. The smallest link (s) has the flow coefficient 0.001 kg/s, Pa^n. Outer pressures are 50 and −50 Pa.

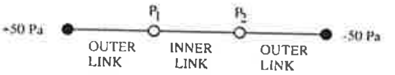


Fig. 3. A 02-Network.

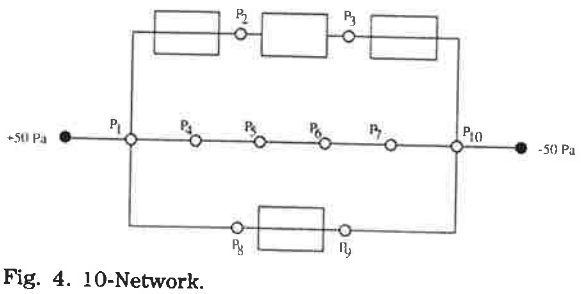


Fig. 4. 10-Network.

(2) 10-Network

This example has been constructed especially to show problems in different nonlinear solvers. All flow exponents are one-half. The sizes of the openings are a mixture of large and small with a maximum ratio of 40 000. An additional example, with all links equal to 0.001 kg/s Pa^n, was also used. Outer pressures are 50 and −50 Pa. Figure 4 shows the network.

(3) 45/41- and 27/25-Networks

These buildings consist of three and five floors with eight rooms per floor. These examples enable us to study the characteristics of different solvers when the number of nodes is increasing.

The centre shaft is simulated in two ways – as one node at middle-floor level (41 and 25) and as one node per floor (45 and 27). The latter takes into account friction losses in the shaft. The connections between the floors were first arranged in two ways – no connections and maximum possible number of connections, i.e., tight and untight floors. This created four different networks with different degrees of band structure and could therefore be used to show the differences between various linear solvers.

One study, however, proved that the most interesting cases are those with tight floors because they show the greatest difference between a band solver and a skyline approach.

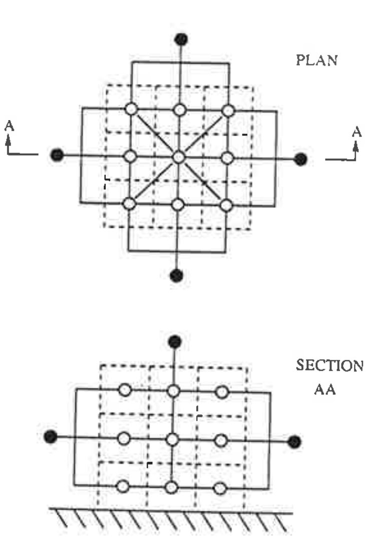


Fig. 5. 27-Network.

The buildings contain a mixture of large and small openings. Outer surfaces 0.001 kg/s, Pa^n; inner surfaces 0.01; inner doors 1, and shaft 10. In addition, all flow coefficients are set to 0.001 kg/s, Pa^n.

6.2. Other assumptions

The criteria for convergence has to be well specified – a relative limit has some advantages. However, in this comparison we use an absolute flow balance limit of E-6 kg/s or 0.0036 kg/h which, for our examples, seems to be appropriate.

At a very low pressure difference, a very steep gradient and large derivative can occur. A leak is therefore linearized between 0 and 10^{-5} Pa when the pressure difference is below 10^{-5} Pa.

All air densities were set to 1.20 kg/m^3.

The sizes of the links, i.e., the flow coefficients, are in the range 0.001–10 kg/s, Pa^n or 3.6–36 000 kg/h, Pa^n. The maximum ratio is therefore 10 000. The largest link represents a very large opening similar to an open garage door. The flow exponents are 0.5.

6.3. Linear solvers

Four methods were tested: a Gaussian with complete pivoting, a band-Cholesky, a skyline-Cholesky, and an iterative method. All runs were made with MOVECOMP–PC, simply exchanging the linear solvers. The iterative method, Gauss–Seidel, created problems. Although the cause is unclear, too many iterations are needed. Further comparison with this routine is not included here.

A sparse matrix can be organized to limit the amount of work during the solving procedure. To create a narrow band or a low skyline are examples of this. Simple automatic procedures in choosing the numbering of the nodes might be possible although this has not been done in this comparison.

The band routine is based on the Cholesky's method as described in Sections 4.4 and 4.5. To avoid operating on zeros the algorithm has been modified to take band structures into account. Zeros outside the band are not involved in the calculations which are done through the limits of the summations. This can reduce the execution time considerably as has been shown above and will be shown below. The routine determines the band width and thereafter operates on the band only, or more precisely, half the band because the Jacobian matrix is symmetric.

Two features were tested. First, the execution time as a function of the order of the matrix and second, the execution time as a function of the half band width. As a comparison an identical routine, but one with no band features, and two routines from LINPACK [14] were timed.

The result of the influence of the order of matrix is shown in Fig. 6.

Two cases are shown, full matrix and diagonal matrix. This gives the full range of all matrices. As can be seen, the benefits of taking the band structure into consideration increase with the order of matrix. At order 40 a diagonal matrix is solved in 11% of a full matrix. At order 20 the corresponding number is 23%.

The result of the influence of the half band width for a matrix of order 20 is shown in Fig. 7.

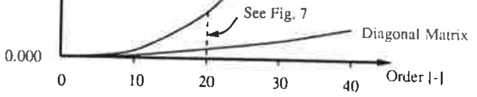
The drop in execution time for the band solver with decreasing band width is similar to the curves in Fig. 2. The non-band solver is identical to the band solver except that the band features are removed, i.e., this routine operates on the full matrix. We can see that the administration of the programs is distorting the results.

The dotted lines show the LINPACK routines. LINPACK(B) is a routine working on a band matrix stored in packed form. Time taken to store the data in packed form is not included here. LINPACK(NB) is a similar routine without the band features. These routines are generally more time-consuming.

It is not certain that the matrix has a well-developed band structure in airflow applications. The skyline approach is therefore interesting.

The band solver and the skyline solver were timed on all networks. As has been mentioned, these examples were chosen because they give as large a difference as possible between these methods. The difference between them is therefore the maximum range that is likely.

The figures are determined by one run per matrix and then divided by the number of iterations. The calculations for the matrix sizes 10 and 02 are forced to 200 iterations for purposes of accuracy. Since it seems to represent a good average the



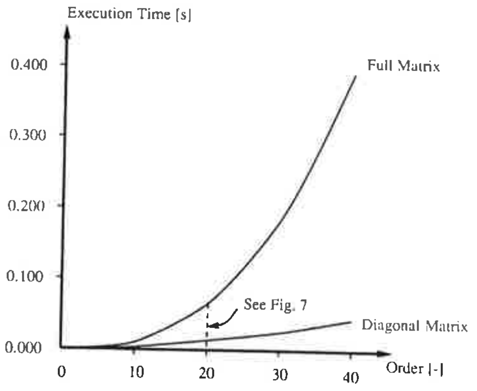


Fig. 6. The influence of the order of the matrix on the execution time for the band solver.

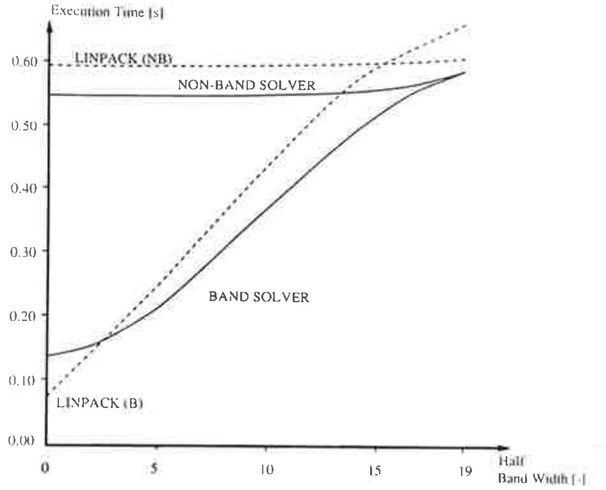


Fig. 7. The influence of the half band width on the execution time for the band routine and LINPACK.

determination of the half band width and the skyline are, in these cases, assumed to be done once for every sixth iteration. This determination requires an insignificant amount of time for any of the matrix sizes used here.

In Table 1 the "Half band width" column does not give any information about how well the band is organized. The "Ratio-Elem" column contains this information. The ratio is defined as the ratio between the number of elements in the lower triangular matrix, including the diagonal, that the solver will operate on for the skyline and band methods respectively. The lower the ratio the less organized the band structure.

It can be seen that the increase of time for the Gaussian method is close to the cube of the order of the matrix as predicted in Section 4.1. The complete pivoting is a N^3 process in logical operations which explains the substantial difference between the Gaussian method and the band approach. The numbers within parentheses represent the time for the skyline compared with that for the band solver. As expected, the larger the matrix the more efficient the skyline approach. The skyline does not require significantly more time even for a perfect band or full matrix.

The examples chosen have very poor band structure, the intention being to show the difference between the band and skyline solvers. The 10-matrix has almost as bad a band structure as the two larger networks. For more realistic examples, the difference between the band and the skyline methods is likely to be smaller although the skyline approach seems always to equal or be faster than the band method. The skyline-Cholesky's method is therefore considered to be a better choice for an efficient solver.

6.4. Non-linear solvers

Relaxation coefficients were selected according to three methods: Newton–Raphson, extrapolated coefficients, and an optimized method. All methods were initialized with two conventional Newton–Raphson iterations.

The Newton–Raphson and the optimized method were timed in MOVECOMP–PC. The number of iterations for the extrapolated method was determined in AIRNET. This method was thereafter assumed to require the same time per iteration as Newton–Raphson.

It was expected that the initialization might influence the number of iterations and it is therefore important that the same initialization is made for all test runs. If a direct linearization of the original problem is performed only the exponents are changed to unity. This means, in fact, that the function is linearized through 0 and 1 Pa pressure differences. Such a direct linearization was chosen.

It is preferable, of course, to linearize each flow equation as close to the solution as possible — one Pascal may seem to be too low and a factor of ten higher is many times more likely. It is best to chose the average pressure difference of all links. However, this average value varies considerably from network to network.

The 27/25 matrices have a similar number of iterations as the 45/41 matrices for all three methods. This is a consequence of the similarity of the

TABLE 1. Execution time per iteration for the linear solvers (s)

Matrix size	Gauss	Band	Skyline	Half band width	Ratio-Elem
45	6.1678	0.1437	0.0867 (60%)	9	175/405 (0.43)
41	5.2004	0.2753	0.1354 (49%)	20	305/651 (0.47)
27	1.4086	0.0688	0.0400 (58%)	9	103/225 (0.46)
25	1.2342	0.0748	0.0494 (66%)	12	136/247 (0.55)
10	0.0839	0.0123	0.0096 (78%)	7	33/52 (0.63)
02	0.0013	0.0010	0.0010 (100%)	1	1/1 (1.00)

matrices except their sizes. The Newton–Raphson method has problems, as expected, at high and low ratios and therefore is less useful.

The number of iterations for the two other methods are far less than for Newton–Raphson. The optimized method requires, on average 46% less iterations than the extrapolated method. For this comparison, the selected examples are shown in bold in Table 2. There is a linearization of some links at the highest ratios for each matrix size owing to extremely small pressure differences.

The results from some selected examples of a calculation with all links initially linearized at 0,0.1; 0,1; 0,10; 0,20; 0,30; and 0,40 Pa are given in Table 3. The exponents in the modified examples have been given different values in the range 0.5–0.9, the larger opening the lower value. If all exponents are equal the pressure where the initialization is done will not change the result as shown below. The system of equations is unchanged owing to the fact that the Jacobian matrix and the right hand vector are multiplied by the same constant. It is not trivial to compare the results from the modified and the unmodified examples since the modified examples are, in many ways, different from the original examples. However the column for 1 Pa shows, to some extent, the range for the number of iterations when the exponents are given more realistic values. It is of interest to note the changes, with initialization, for each modified example. The pattern is, although not clear because of linearizations, that more realistic values will reduce the number of iterations. For this comparison the optimized method was used.

TABLE 2. Number of iterations for the non-linear solvers

Matrix size	Newton	Extrapolated	Optimized
45–10000	>1500	15	8
45–1	24	9	6
41–1000	>1500	15	8
41–1	21	9	6
27–10000	>1500	15	8
27–1	25	9	6
25–1000	>1500	15	8
25–1	22	9	6
10–40000	>1500	19	7
10–1	8	8	6
02–10000	>1500	11	5
02–1000	946	11	7
02–100	86	9	6
02–10	8	8	5
02–1	1	2	1
02–0.1	10	8	5
02–0.01	91	9	6
02–0.001	964	11	7
02–0.0001	>1500	11	5

TABLE 3. Influence of initialization on number of iterations

Matrix size	0.1 Pa	1 Pa	10 Pa	20 Pa	30 Pa	40 Pa
45–10000–mod	8	8	8	8	8	10
45–10000	–	8	–	–	–	–
45–1–mod	5	4	4	4	4	4
45–1	–	6	–	–	–	–
10–40000–mod	7	7	8	9	9	9
10–40000	–	7	–	–	–	–
10–1–mod	4	4	4	4	4	4
10–1	–	6	–	–	–	–
02–10000–mod	6	7	7	7	7	7
02–10000	–	5	–	–	–	–
02–10–mod	4	4	3	3	3	3
02–10	–	5	–	–	–	–

Networks with high ratios require fewer iterations when the initialization pressure is lowered. The opposite is the case for the networks with low ratios. The latter networks have higher pressure differences across the openings with exponent 0.5 than the other networks. The 0.5 openings are the most important because these will have the largest approximation when they are linearized. This verifies that the closer we can linearize the flow equations to the final solution the fewer iterations are required. Nevertheless it seems that initialization is not critical for these examples and that the choice with a direct linearization, i.e., 0,1, is a reasonably good choice. Other types of linearizations of course are possible and would probably have given other results.

The figures in Table 4 are based on one run per example but are forced to 50 iterations and one Newton–Raphson to start with. This procedure was selected to avoid interference from some subroutines and the first Newton–Raphson iteration. All calculations were done with the skyline solver.

The numbers in parentheses are the contributions from the linear solver; compare with Table 1.

The increased time for the optimized method depends almost entirely on the power functions,

TABLE 4. Execution time per iteration for complete solvers (s)

Matrix size	Newton	Optimized	Ratio: Optimized/Newton
45	0.373 (23%)	0.957 (9%)	2.6
41	0.392 (35%)	0.955 (14%)	2.4
27	0.216 (19%)	0.559 (7%)	2.6
25	0.222 (22%)	0.561 (9%)	2.5
10	0.075 (13%)	0.168 (6%)	2.2
02	0.036 (3%)	0.060 (2%)	1.7

TABLE 5. Execution time for complete solvers (*s*)

Matrix size	Extrapolated	Optimized	Difference (%)
45–10000	5.707	5.904	−3
41–1	3.599	4.041	−11
27–1	1.983	2.325	−15
25–1000	3.397	3.471	−2
10–40000	1.454	0.897	+62
10–1	0.612	0.636	−4
02–10000	0.404	0.252	+60
02–10	0.294	0.252	+17
Total	17.450	17.778	

eqns. (37) and (38), that must be evaluated in the optimizer.

According to the figures within parentheses the contribution of the linear solver is 2–35% of the total time in the solver.

To find the time difference between the complete solvers the number of iterations has to be multiplied by the time for each iteration — remembering that the two first iterations are Newton–Raphson iterations. The optimized method switches to Newton–Raphson if $\lambda > 0.95$. This has two purposes, first, it prevents unnecessary optimization and second, it eliminates the influence of round-off errors in the optimizer.

If we assume that the extrapolated method does not require any significant extra time relative to Newton–Raphson, then the methods are quite equivalent. In fact the extrapolation method requires about 2% more time than Newton–Raphson. This has been included in Table 5.

The choice of a non-linear method is not as obvious as in the case of the linear solver. The optimized method always finds the solution in fewer iterations than the extrapolated method which indicates that the optimized method is more efficient in finding the appropriate under-relaxations. On the other hand, the process of finding the relaxation coefficients is so time-consuming that the methods have almost the same execution time for the test cases based on total time for all examples. The main argument for recommending the optimized method for incorporation in infiltration programs is that the method itself is more efficient and stable. The time-consuming process of finding the relaxation coefficients is a different matter, a pure numerical or mathematical problem.

7. Conclusion

The aim of the present study is selecting efficient and robust methods to be used in infiltration programs to solve the nonlinear system of equations representing the airflow distribution in multizone buildings.

Taking into account the special characteristics of the Jacobian matrix, we select a skyline-Cholesky's method for the linear solver. This method can be used in a general way for the problem we are dealing with and it appears to be the more efficient one.

The task is more difficult with regard to the nonlinear solver. Even if a generalized Newton–Raphson method is selected, the choice of an under-relaxation process to ensure the convergence and efficiency is not obvious. We tested two different methods, an extrapolated method and an optimized method, of finding the under-relaxation coefficients. Although the latter method is more time-consuming in finding the relaxation coefficient in each iteration, it appears to be safer and more efficient in the resolution of the nonlinear system. In testing these two methods on the same sample of test cases we found that they are very similar regarding total time. We suggest the optimized method to be used for the nonlinear solver.

Nomenclature

$\mathbf{A}$	matrix, general
C	flow coefficient
corr	correction vector
e	eigenvalue
f	flow balance function
F	$F' = f$
g	flow balance function
G	scalar function
$\mathbf{G}$	matrix, triangular
h	iteration function
i	node number
$\mathbf{I}$	matrix, unity
j	node number, element in $\mathbf{J}$
$\mathbf{J}$	Jacobian matrix
$\mathbf{L}$	matrix, lower triangular
m	element in $\mathbf{L}$
$\dot{m}$	mass flow
M	half band width
n	flow exponent
N	order of matrix
$\mathbf{U}$	matrix, upper triangular
u	element in $\mathbf{U}$
p	pressure
P	polynomial
z	complex number
λ	relaxation coefficient

Super- and subscripts

a, b	zones
n	iteration number
T	transpose
i, j, k, p	elements in **J**, **L**, and **U**

References

1 G. Dahlquist and A. Bjork, *Numerical Methods, Series in Automatic Computation,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974, 573 pp.

2 D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra,* W. H. Freeman and Co., San Francisco, 1963, 621 pp.

3 G. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems, Series in Automatic Computation,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1967, 148 pp.

4 H. R. Schwarz *et al.*, *Numerical Analysis of Symmetric Matrices, Series in Automatic Computation,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973, 276 pp.

5 J. H. Wilkinson and C. Reinsch, *Linear Algebra, Handbook for Automatic Computation, Vol 2,* Springer-Verlag, New York, 1971, 439 pp.

6 G. Golub and C. Van Loan, *Matrix Computations,* North Oxford Academic, 1983.

7 G. Dhatt, G. Touzot and G. Cantin, *The Finite Element Method Displayed,* John Wiley & Sons, New York, 1984.

8 S. D. Conte and C. de Boor, *Elementary Numerical Analysis – An algorithmic Approach,* McGraw-Hill Book Company, New York, 1972, 395 pp.

9 J. M. Ortega and W. C. Rheinbolt, *Iterative Solution of Non-linear Equations in Several Variables,* Academic Press, New York, 1970.

10 R. T. Rockafellar, *Convex Analysis,* Princeton University Press, 1972.

11 P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization,* Academic Press, New York, 1981, 392 pp.

12 G. N. Walton, *AIRNET – A Computer Program for Building Airflow Network Modeling,* US Department of Commerce, 1989.

13 M. K. Herrlin, Luftstromning i byggnader – en beraknings-modell, *Licentiate Thesis,* Division of Building Services Engineering, Royal Institute of Technology, Stockholm, Sweden, 1987, 125 pp.

14 J. J. Dongarra *et al.*, *LINPACK User's Guide*, Society for Industrial and Applied Mathematics, 1979.