Solution Strategies in GFLOW

Henk M. Haitjema

April 4, 2005

Introduction

The groundwater flow problem in GFLOW is formulated in terms of the analytic element method (AEM), which is a variant of the Boundary Integral Equation Method (BIEM). Within the AEM a solution is sought for the a priory unknown "strength" parameters of the analytic elements. The strength parameter of a line-sink is its extraction rate σ $[m^2/day]$, which is constant along the line-sink (Haitjema, 1995 section 5.1.2 and Strack and Haitjema, 1981a). The strength parameters for a line-doublet define the jump in the discharge potential s $[m^3/day]$ generated across the line-doublet (Haitjema, 1995 section 5.1.3 and Strack and Haitjema, 1981b). The strength parameter for a head specified well is its pumping rate Q $[m^3/day]$ (Haitjema, 1995 section 3.1.8), while the strength parameters for a partially penetrating well define the linear and singular sink densities along a three dimensional line-sink at the well axis (Haitjema, 1995 section 4.2.1). Most strength parameters are a priory unknown, but for each strength parameter a boundary condition can be formulated at a collocation point, usually selected at the line-sink or line-doublet itself. Each analytic element influences the discharge potential everywhere, linearly proportional to the value of its strength parameter or parameters. Consequently, the solution procedure for the unknown strength parameters x_i gives rise to set of linear equations:

$$a_{ij}x_j = b_i \tag{1}$$

by which the coefficient matrix a_{ij} contains the "influences" of all analytic elements on all collocation points and is *fully populated*. The vector x_i is the "solution vector" and contains the strength parameters of the analytic elements. The vector b_i is the "known vector" resulting from the boundary conditions formulated at the collocation points. In writing (1) the Einstein summation convention is adopted, implying summation over the dummy index j. In GFLOW each line-sink has a collocation point at its center, while each line-doublet has a collocation point at its center and at its endpoints. A head specified well has a collocation point at the well screen perimeter (to the right of the well axis) and a partially penetrating well has three collocation points along the well screen perimeter; two near the ends and one at the center of the well screen. Details on the form of the set of equations are found in section 5.1.5 of Haitjema (1995).

Since the coefficient matrix a_{ij} is fully populated most analytic element codes employ a *direct solution procedure*, such as Gauss elimination. However, each analytic element function may be formulated to generate its largest contribution to the discharge potential on its own collocation point. It appears that under those circumstances an iterative solution procedure such as Gauss Seidel will also converge to the desired solution. In GFLOW the direct solution method of LU decomposition is used employing the Fortran subroutines DECOMP and SOLVE published by Forsythe et al. (1977). Since all functions in GFLOW are forced to contribute zero or nearly zero at the (remote) reference point, the set of equations is also suitable for an iterative solution procedure (e.g. Gauss Seidel or Conjugate Gradient method).

Iterative Refinement

In GFLOW the equations are defined as follows:

$$a_{ij}\left(x_j^{(n)} - x_j^{(n-1)}\right) = b_i^{(n)} - b_i^{(n-1)} \tag{2}$$

where $x_i^{(n)}$ and $b_i^{(n)}$ are the solution vector and known vector, respectively, of the current solution step n, while $x_i^{(n-1)}$ and $b_i^{(n-1)}$ are the solution vector and known vector, respectively, at the previous solution step $n-1$. The formulation (2) has two advantages over (1). First of all, the numerical accuracy of the Gauss elimination procedure is applied to calculating the *change* in the strength parameters, rather than calculating the strength parameters themselves. Since GFLOW is coded in Fortran with REAL(8) and COMPLEX(8) (double precision), successive solutions are likely to lead to an iterative refinement of the strength parameters. A second advantage is that the analytic elements in GFLOW may be subdivided into two categories: elements with a priory known strength parameters (e.g. discharge specified wells and recharge only inhomogeneities) and elements with a priory unknown strength parameters (e.g. line-sinks representing streams and lakes). Only the latter elements are involved in the left-hand side of (1) or (2), while only the first (known) elements are involved in the right-hand side of (1). In writing (2) the elements with known strength parameters (that do not change between solutions) occur in both $b_i^{(n)}$ and $b_i^{(n-1)}$, thus cancelling each other. To illustrate this consider the equation for a head specified line-sink at the i^{th} collocation point:

$$F_{ij}\sigma_j + F_{ij}\sigma_j^* + C = \Phi_i \tag{3}$$

where σ_i and σ_i^* are the a priory unknown and known line-sink strength parameters, respectively, C is an unknown integration constant and Φ_i is the specified discharge potential at the i^{th} collocation point. It is assumed in (3) that there are only line-sinks in the model and that the Einstein summation convention applies. Hence, summation is implied over the dummy index j, summing over all unknown and known line-sink strengths. The function F_{ij} is the contribution

to the discharge potential Φ_i at the i^{th} collocation point due to the j^{th} line-sink with strength 1. F_{ij} for the unknown strength parameters are the matrix coefficients a_{ij} in (1). Writing (3) in the form of (1) yields:

$$a_{ij} = F_{ij} \;\; ; \;\; a_{iN} = 1.0 \;\; ; \;\; b_i = \Phi_i - F_{ij}\sigma_j^* \;\; ; \;\; x_i = \sigma_i \;\; ; \;\; x_N = C \qquad (4)$$

for the case that there are N line-sinks with unknown strengths σ, hence a system of N linear equations.

Writing (3) in the form of (2) implies subtracting the discharge potential $\Phi_i^{(n-1)}$ from both sides in (1). $\Phi_i^{(n-1)}$ is calculated as in (3), but with σ_i replaced by $\sigma_i^{(n-1)}$, which are the a priory unknown strength parameters as obtained from a previous solution. Thus:

$$F_{ij}\sigma_j^{(n)} + F_{ij}\sigma_j^* + C^{(n)} - (F_{ij}\sigma_j^{(n-1)} + F_{ij}\sigma_j^* + C^{(n-1)}) = \Phi_i - \Phi_i^{(n-1)} \qquad (5)$$

Note that the known strength parameters σ_i^* do not change between successive solutions and thus cancel in (5). Consequently, these known parameters do not have to be taken into account separately as in (4) when applying the iterative refinement formulation (2). When writing (2) with (5) in the form of (1) we have:

$$\begin{aligned} &a_{ij} = F_{ij} \;\; ; \;\; a_{iN} = 1.0 \;\; ; \;\; b_i = \Phi_i - \Phi_i^{(n-1)} \\ &x_i = \sigma_i^{(n)} - \sigma_i^{(n-1)} \;\; ; \;\; x_N = C^{(n)} - C^{(n-1)} \end{aligned} \qquad (6)$$

Solving a Non-Linear Problem

The solution procedure discussed sofar applies to *linear* equations, which means that the coefficient matrix a_{ij} does not depend on the solution. In other words, the coefficient matrix a_{ij} remains constant during the successive solutions suggested by (2). In practice, however, there are several conditions at collocation points that give rise to *non-linear* equations, which means that the coefficient matrix a_{ij} depends on the solution x_i. This circumstance gives a new meaning to the concept of iterative refinement as discussed above. Successive solutions are now required to adjust the coefficient matrix a_{ij} until the procedure converges on a single unique solution x_i, after which the matrix coefficients remain constant.

There are several features in GFLOW that give rise to non-linear equations. In nearly each case the non-linear behavior occurs only under *unconfined* flow conditions, except for non-linearities that arise from conjunctive surface water and groundwater flow solutions. The non-linear features in GFLOW are:

1. Line-sinks with resistance. These are line-sinks that represent surface water features that are separated from the aquifer by a resistance layer.

2. Inhomogeneity domains with a jump in the aquifer bottom. For details see the pdf document "Theory of Modeling Inhomogeneity Domains in GFLOW" in the documents folder of the GFLOW applications directory.

3. Horizontal barriers (slurry walls) that are partially penetrating the aquifer.

4. Line-sinks which strength is dictated by available streamflow (conjunctive surface water and groundwater flow solutions). This may occur under both confined and unconfined flow conditions.

In case 1 the change in a_{ij} constitutes a modification of the diagonal element a_{ii} (see equation 5.69 of Haitjema, 1995), while for the cases 2 and 3 several elements near the diagonal may also need to be modified. In all cases the modifications result from a change in the head at the collocation points between two successive solution steps.

In case 4, and under some circumstances in case 1, the strength parameter of one or more line-sinks becomes dictated by limitations imposed by the surface water flow solution (Haitjema, 1995 section 5.2.4) or in case 1 by the head falling below the resistance layer (Haitjema, 1995 section 5.2.3 "Percolating Surface Waters"). For these cases the equation for the boundary condition at the associated collocation point must be removed from the set of equations. In matrix jargon: for each i^{th} equation to be removed the i^{th} row and i^{th} column of the coefficient matrix are struck and the i^{th} element of the known vector is eliminated. This, of course, reduces the system of equations to be solved.

The non-linear aspect of the equations to be solved in GFLOW is the primary motivation for the iterative refinement formulation, whereby after each solution the newly obtained strength parameters are used to update the heads at collocation points and calculate a new surface water flow solution. Based on this new solution a new coefficient matrix is being generated, with several new elements and several equations removed or added! The solution procedure is summarized in the following steps:

1. Generate a complex vector z_i of collocation points and an associated integer vector t_i with a code that defines the type of boundary condition to be required at the corresponding collocation point. In some cases a second point is associated with the collocation point, which is stored in a separate, but parallel vector. After the first iteration (cycle through these steps), hence after a preliminary solution exists, line-sinks with resistance are checked for percolating conditions. If present, the line-sink strength is calculated and considered known. Hence no collocation point is added to z_i, reducing the set of equations by one equation for each percolating line-sink.

2. Generate the coefficient matrix a_{ij}. The type of matrix coefficient to be generated for row i is based on the value of the equation type vector element t_i. For instance, each analytic element may be asked to contribute to the discharge potential at z_i, or to contribute to the flow between the points z_i and some other point. In calculating the diagonal elements or near diagonal elements for non-linear equations, use the heads based on the previous solution.

3. Generate the known vector b_i, using the equation type vector t_i to define the type of boundary condition requested at the i^{th} collocation point. When relevant, the discharge potential calculated with the strength parameters obtained from the previous solution is subtracted from the specified potential.

4. Decompose the coefficient matrix using LU-decomposition. At this point the original coefficient matrix a_{ij} has been replaced by the lower triangular matrix L_{ij} and upper triangular matrix U_{ij}. In GFLOW the routine DECOMP is used, see Forsythe et al. (1977). Note, DECOMP employs *partial pivoting*, which constitutes a reordering of the equations to maximize the so-called "pivots", which results in minimizing the numerical errors during the next (solution) step. The pivoting information (order of the equations in the decomposed matrix) is stored in the vector p_i.

5. Solve for the solution vector x_i using forward elimination and back substitution. In GFLOW the routine SOLVE is used, which relies on the decomposed matrix and pivot vector obtained from DECOMP (Forsythe et al., 1977).

6. Substitute the solution vector into the strength parameter arrays of the various analytic elements.

7. Calculate the errors in the specified conditions at all collocation points. In other words, check if the solution is accurate.

8. Solve the surface water flow problem using the new groundwater flow solution for an updated baseflow component. At this step it is decided which line-sinks should become "discharge specified," hence removed from the set of equations, and which line-sinks that had been removed during an earlier iteration should be considered of unknown strength, hence added again to the set of equations.

The steps 1 through 7 form one groundwater solution in a series of solutions that are to converge on the true values x_i. For the case of conjunctive surface water and groundwater flow solutions step 8 is also part of each iteration. In GFLOW the number of iterations required for convergence is estimated in advance and specified as the "number of iterations (inner loop)." Note, "outer loop iterations" are applied to reach convergence on lake water balances, see the pdf document "Modeling Lake-Groundwater Interactions in GFLOW" in the documents folder in the GFLOW applications directory.

Computational Effort

The solution procedure outlined above is computationally intens. Due to the non-linear nature of the equations to be solved, it is necessary to regenerate the coefficient matrix a_{ij} (step 2) and apply LU-decomposition to a_{ij} (step 4) for every iteration. From all 8 steps that comprise one iteration, these two

steps (2 and 4) are the most computationally intensive (use the most processor time). This is particularly true if the set of equations becomes large, say over 1000 equations. While the coefficient matrix could be saved to disk and then *modified* at each iteration, instead of completely regenerated from scratch, the LU decomposition step cannot be avoided and for large systems of equations it requires the most processor time.

Below a strategy is outlined to significantly save on computational effort by avoiding both the regeneration and decomposition steps of the coefficient matrix.

Using the Sherman-Morrison Formula

It is possible to change matrix coefficients and even eliminate entire equations from an existing set of equations by modifying the solution vector x_i for the original set of equations into a new solution vector for the modified set of equations. To do so we make use of the Sherman-Morrison formula (Gill et al., 1974; Press et al., 1992; Golub and Loan, 1996), which written in index notation is:

$$\left(a_{ij} - u_i v_j\right)^{-1} = a_{ij}^{-1} + \frac{a_{ik}^{-1} u_k v_l a_{lj}^{-1}}{1 - \lambda} \tag{7}$$

with

$$\lambda = v_k a_{kl}^{-1} u_l \tag{8}$$

The vector product $u_i v_j$ forms a "correction matrix" for the original coefficient matrix a_{ij}. The Sherman-Morrison formula (also known as the Bartlett-Sherman-Morrison-Woodburry formula) provides an expression for the modification of the inverse of the original coefficient matrix based on this correction. In practice, however, the inverse matrix is seldom calculated. Therefore, expression (7) with (8) will be applied to the set of equations (1) to obtain a correction of its solution x_i based on the modification of the coefficient matrix as shown on the left-hand side of (7).

We write the solution x_i as:

$$x_i = a_{ij}^{-1} b_j \tag{9}$$

We write the solution x_i^* to the modified set of equations as:

$$x_i^* = \left(a_{ij} - u_i v_j\right)^{-1} b_j \tag{10}$$

In writing (9) and (10) it is implied that the known vector b_i belongs to the modified set of equations! Applying (7) yields:

$$x_i^* = a_{ij}^{-1} b_j + \frac{a_{ik}^{-1} u_k v_l a_{lj}^{-1} b_j}{1 - \lambda} \tag{11}$$

where λ is defined by (8). Expression (11) may be rewritten by use of (9) as:

$$x_i^* = x_i + \frac{u_i' v_l x_l}{1 - \lambda} \tag{12}$$

where u'_i is defined as

$$u'_i = a_{ij}^{-1} u_j \tag{13}$$

with which λ becomes:

$$\lambda = v_k u'_k \tag{14}$$

In summary, the computation of the solution x_i^* for the modified set of equations requires the following steps:

1. Calculate an initial solution x_i based on the *original* coefficient matrix and the known vector b_i for the *modified* set of equations.
2. Calculate the vector u'_i, which is the solution to a system of equations with the original coefficient matrix and the vector u_i as known vector.
3. Calculate the new solution vector x_i^* by use of (12) with (14).

Steps 1 and 2 can be accomplished by *any solution method*, including the LU decomposition, forward elimination and back substitution employed in GFLOW. Step 3 (equation (12) with equation (14)) makes use of the results from step 1 and 2, and is also independent of the solution method.

Eliminating one equation

Once a line-sink strength parameter changes its status from unknown to known, its corresponding equation must be removed from the original set of equations. Instead of actually *removing* an equation, however, we will force the solution to contain the specified value for the line-sink strength parameter. This is done as follows. Assume that x_n is the element of the solution vector that corresponds to the strength parameter that has become a "known" strength. If the known vector b_i is calculated with this known strength parameter than the value of x_n must become zero, as may be concluded from (2). To obtain this result ($x_n = 0$) we want to replace the original n^{th} equation by:

$$0.x_1 + 0.x_2 + 0.x_3 + ... + a_{nn}.x_n + ... + 0.C = 0 \tag{15}$$

where a_{nn} may be any non-zero number, but is for numerically reasons best selected very large. All other equations remain unaltered. Expression (15) implies that all matrix coefficients in row n must be set to zero, except a_{nn}, which must be set to a large number.

In the Sherman-Morrison formula (7) this is accomplished by selecting the vectors u_i and v_i as follows:

$$u_i = 0 \quad (n \neq i) \; ; \quad u_n = 1 \tag{16}$$

and

$$v_i = a_{ni} \quad (n \neq i) \; ; \quad v_n = 10^{100} \tag{17}$$

The choice for v_n is arbitrary.

Eliminating N equations

If more than one equation is to be removed, which is usually the case, the new set of equations may be written in the form:

$$\left(a_{ij} - \sum_{n=1}^{N} \overset{n}{u}_i \overset{n}{v}_j \right) x_i = b_i \tag{18}$$

The solution x_i to (18) may be obtained by repeated application of the Sherman-Morrison formula. To develop expressions for this procedure we define the coefficient matrix $\overset{m}{a}_{ij}$ as the original matrix with m modifications:

$$\overset{m}{a}_{ij} = \left(a_{ij} - \sum_{n=1}^{m} \overset{n}{u}_i \overset{n}{v}_j \right) \tag{19}$$

We further define $\overset{m}{x}_i$ as

$$\overset{m}{x}_i = \left(\overset{m-1}{a}_{ij} \right)^{-1} b_j \tag{20}$$

and similarly

$$\overset{m}{\overset{n}{u'}}_i = \left(\overset{m-1}{a}_{ij} \right)^{-1} \overset{n}{u}_j \tag{21}$$

The solution vector $\overset{m}{x}_i$ is obtained by solving the original system of equations with m equations modified and the vector b_i as known vector. The "solution vector" $\overset{m}{\overset{n}{u'}}_i$ is obtained by solving the original system of equations with m equations modified and the vector $\overset{n}{u}_i$ as known vector. Successively applying the Sherman-Morrison formula leads to the following recursion formula for $\overset{m}{x}_i$:

$$\overset{m+1}{x}_i = \overset{m}{x}_i + \frac{\overset{m}{\overset{m}{u'}}_i \overset{m}{v}_k \overset{m}{x}_k}{1 - \overset{m}{\lambda}} \tag{22}$$

where

$$\overset{m}{\lambda} = \overset{m}{v}_k \overset{m}{\overset{m}{u'}}_k \tag{23}$$

and where

$$\overset{1}{x}_i = a_{ij}^{-1} b_j \tag{24}$$

Evaluation of (22) with (23) requires the calculation of $\overset{m}{\overset{m}{u'}}_i$ by use of the following recursion formula:

$$\overset{n+1}{\overset{m}{u'}}_i = \overset{n}{\overset{m}{u'}}_i + \frac{\overset{n}{\overset{n}{u'}}_i \overset{n}{v}_k \overset{n}{\overset{m}{u'}}_k}{1 - \overset{n}{\lambda}} \tag{25}$$

where $\overset{n}{\lambda}$ is given by

$$\overset{n}{\lambda} = \overset{n}{v}_k \overset{n}{u'}_k \tag{26}$$

and where

$$\overset{1}{u'}_i = a_{ij}^{-1} \overset{m}{u}_j \tag{27}$$

Modifying individual matrix coefficients

The same procedure outlined above may also be used to modify only one or just a few matrix coefficients per row of equations to be modified. To modify the m^{th} coefficient in the n^{th} equation the vectors u_i and v_i are defined as follows:

$$u_i = 0 \quad (n \neq i) \; ; \quad u_n = 1 \tag{28}$$

and

$$v_i = 0 \quad (m \neq i) \; ; \quad v_m = a_{nm} - a^*_{nm} \tag{29}$$

where a^*_{nm} is the new coefficient that replaces the original coefficient a_{nm}. If more coefficients are to be modified in the same row it is sufficient to adjust v_i by replacing additional matrix coefficients as done in (29).

Implementation in GFLOW

In order to execute $\overset{m}{x}_i$ in (22) the recursion formula (25) must be evaluated for $n = 1, 2, ..., m$. This implies only 1 step, the evaluation of (27) for the vector $\overset{1}{u}_i$, up to m steps for $\overset{m}{u}_i$. In GFLOW the vectors $\overset{m}{u}_i$ $(m = 1, N)$ are each stored in an array and successively replaced by the vectors $\overset{m}{u'}_i$ $(m = 1, N)$ as they result from (25) with (26). These calculations are done in parallel with the evaluation of (22) with (23).

The Sherman-Morrison formula is currently only used to remove equations from the original set of equations. At each iteration step the existing LU decomposed matrix is loaded from disk and the solution is adjusted for the N equations that are to be removed at that step. The reintroduction of equations, when a known line-sink strength parameter becomes an unknown again, is thus accomplished by not eliminating it. After the decomposed matrix has been loaded from disk equation (24) and all N equations (27) are executed and the solutions stored. To generate the vectors v_i with (17) the original coefficient matrix is loaded from disk and stored in the array used for the decomposed matrix.

Using linearized equations

As stated, currently no matrix coefficients are updated in GFLOW, hence the corrections defined by (28) and (29) are not implemented in GFLOW. Consequently, the set of equations is de facto linearized, except for the fact that some

equations are "removed" as discussed above. To understand the implication of the linearization we write the i^{th} equation in (2) as:

$$\left(a_{ij} + \Delta \overset{n}{a_{ij}}\right)\left(x_j^{(n)} - x_j^{(n-1)}\right) = b_i^{(n)} - b_i^{(n-1)} \tag{30}$$

where $\Delta \overset{n}{a_{ij}}$ is the deviation of the initial matrix coefficient a_{ij} for the n^{th} iteration. We may rewrite (30) as

$$a_{ij}\left(x_j^{(n)} - x_j^{(n-1)}\right) = b_i^{(n)} - b_i^{(n-1)} - \Delta \overset{n}{a_{ij}}\left(x_j^{(n)} - x_j^{(n-1)}\right) \tag{31}$$

whereby the change in the matrix coefficient times the difference in successive values of x_i (strength parameters) is moved from the left-hand side to the right-hand side. This leaves us with the original matrix coefficients for all iterations. However, the term cannot actually be added to the right-hand side (the known vector), since the new solution $x^{(n)}$ is not yet know. On the other hand, as the solution process converges, the difference in two successive values for the strength parameters becomes zero, hence the last term in (31) will vanish and thus, in the end, does not have to be added to the known vector. It has been found, however, that under certain circumstances and for large values of $\Delta \overset{n}{a_{ij}}$ (for instance large resistances for line-sinks), omitting this term in the known vector may lead to a slow convergence.

Performance

It is noted that the evaluation of (27) for each of the N vectors u_i involves mostly multiplications by zero in the forward elimination process that is associated with the LU decomposition solution method. A rewrite of the routine SOLVE for the evaluation of (27) may yield some additional performance enhancement. In the event that individual matrix coefficients are adjusted , see (29), most elements of the vector v_i are zero as well. This reduces the scalar products with v_i in (22) and (23) to one or only a few multiplications if multiplications by zero are omitted.

References

Forsythe, G. E., Malcolm, M. A., & Moler, C. B. (1977). *Computer Methods for Mathematical Computations*. Prentice Hall, Inc.

Gill, P. E., Golub, G. H., Murry, W., & Saunders, M. A. (1974). Methods for modifying matrix factorizations. *Mathematics of Computationn*, 28:505 –535.

Golub, G. H. & Loan, C. F. (1996). *Matrix Computations*, (third ed.). John Hopkins.

Haitjema, H. M. (1995). *Analytic Element Modeling of Groundwater Flow*. Academic Press, Inc.

Press, W. H., Flannery, B. P., Tenkolsky, S. A., & Vetterling, W. T. (1992). Sherman Morrison Formula. In: *Numerical recipes in Fortran: The art of scientific computing*, (second ed.), pages 65–67. Cambridge University Press.