OPTIMIZATION OF NON-LINEAR LARGE SCALE SYSTEMS WITH LINEAR DYNAMICS - AN APPLICATION TO LOAD SCHEDULING

A. Friedlander - C. Lyra** - H. Tavares***

ABSTRACT. This paper presents an algorithm for optimization of large scale non-linear dynamical problems with linear constraints. The approach was devised with the aim of solving deterministic scheduling problems of hydrothermal power systems. The method has a conception based on the overall structure of the reduced gradient method, more specifically it is based on the implementation of this method by Murtagh and Saunders (1978). The dynamical characteristic of the problem leads to a constraints matrix with staircase structure. Skillfull use of this feature in storing and computations is mandatory for large scale problems. The staircase structure is considered in the L - U decomposition of the constraints matrix and in the updating scheme which is based in the classic paper of Bartels-Golub.

The algorithm was implemented in a computer program and an application to load scheduling is presented.

---

* Departamento de Matemática Aplicada - IMECC - UNICAMP.

** FEE - UNICAMP.

1. INTRODUCTION

Optimization of non-linear dynamic systems with linear dynamics arise, frequently, in control and planning of engineering systems. The main feature of the problems is that their constraints are represented by sparse matrices with staircase structure.

Linear dynamic programming has been the object of many earlier research. Krivonozhko and Chebotarev (1976) and Propoi and Krivonozhko (1978) developed an algorithm to solve a linear dynamic problem. This algorithm adapted the Simplex method for constraints matrices with staircase structure and used the Forrest - Tomlin (1972) updating scheme, which has good properties related with sparsity preservation but is not numerically stable. Fourier (1979, 1982, 1983, 1984) surveys the properties of staircase matrices and the varieties of Gaussian elimination for these matrices. Friedlander, Medina and Tavares (1982) presented an algorithm to solve a linear dynamic programming problem, establishing a compromise between the preservation of staircase structure and numerical stability. Friedlander (1986) implemented a computer code for this algorithm and showed the viability of extending it to non-linear problems.

In this paper we solve the dynamic problem with non-linear objective function and linear restrictions. The algorithm is based on Murtagh and Saunders (1978) implementation of the reduced gradient method. The basic constraints matrices are factorized in the LU form. When a change of basis is necessary, the factors are updated by a procedure based on the Bartels and Golub's Scheme (1969). The decomposition and the updating process take advantage of the staircase structure. During the whole process a compromise is established between structure preservation and numerical stability.

2. STATEMENT OF THE PROBLEM

The non-linear programming problem with linear dynamics has the form:

$$\text{minimize} \quad f(x,u) \tag{1}$$

subject to

$$x(t+1) = A(t)x(t) + B(t)u(t) \tag{2}$$

$$x(0) = x^{o} \tag{3}$$

$$C(t)x(t) + D(t)u(t) = g(t) \tag{4}$$

$$\underline{x} \leq x(t+1) \leq \bar{x} \tag{5}$$

$$\underline{u} \leq u(t) \leq \bar{u} \qquad t = 0,1,\ldots,T-1 \tag{6}$$

where $f(x,u) \in \mathbb{C}^1$

$x(t) \in \mathbb{R}^n$, state vector

$u(t) \in \mathbb{R}^r$, control vector

$g(t) \in \mathbb{R}^m$, $m \leq r$, given resource vector

$x(0) = x^{o}$, initial state vector

$A(t) \in \mathbb{R}^{n\times m}$, $B(t) \in \mathbb{R}^{n\times r}$, $C(t) \in \mathbb{R}^{m\times n}$, $D(t) \in \mathbb{R}^{m\times r}$

T, number of time periods, fixed.

The constraint matrix of this problem has the form shown below

$$A = \begin{array}{c} \begin{matrix} u(0) & x(1) & u(1) & x(2) & \cdots & x(T-1) & u(T-1) & x(T) \\ r & n & r & n & & & & \end{matrix} \\ \begin{pmatrix} D(0) & & & & & & & \\ B(0) & -\mathbb{I} & & & & & & \\ C(1) & D(1) & & & & & & \\ A(1) & B(1) & -\mathbb{I} & & & & & \\ & & \ddots & & & & & \\ & & & C(T-1) & D(T-1) & & & \\ & & & A(T-1) & B(T-1) & -\mathbb{I} & & \end{pmatrix} \end{array}$$

Fig. 1

where $\mathbb{I}$ is a $(n \times n)$, identity matrix.

3. SUMMARY OF THE MURTAGH AND SAUNDERS VERSION OF THE REDUCED GRADIENT METHOD

We wish to

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \tag{1}$$

subject to

$$Ax = b \tag{2}$$

$$\underline{x} \leq x \leq \bar{x} \tag{3}$$

where $A \in \mathbb{R}^{m \times n}$, $m \leq n$.

Given x^k, a vector that satisfies (2) and (3), $\varepsilon > 0$ and a partition of the set of general constraints (2) as follows

$$Ax = \begin{array}{|c|c|c|} \hline B & S & N \\ \hline \end{array} \begin{pmatrix} x_B \\ x_S \\ x_N \end{pmatrix} = b.$$

basics; super basics; non-basics

The matrix B is square and non-singular as in the Simplex method, $S \in \mathbb{R}^{m \times s}$ with $0 \leq s \leq n - m$ and N is the matrix formed by the remaining columbs of A. The associated variables x_B, x_S, x_N are called the basics, superbasics and non-basics, respectively. Then if we fix x_N, and let x_B and x_S free to vary between their bounds, from (2) and (3) we obtain

$$x_B = B^{-1}b - B^{-1}Sx_S - B^{-1}Nx_N .$$

Define, $h(x_S) = f(x_B, x_S, x_N)$.

STEP 1. Compute the gradient vector $\nabla f(x^k)$.

STEP 2. Solve $B^T\pi^k = \nabla_{x_B} f(x^k)$.

STEP 3. Compute the reduced gradient

$$\nabla x_S h(x_S^k) = \nabla x_S f(x^k) - S^T \pi^k.$$

If $s = 0$ or $\| \nabla x_S h(x_S^k) \| \leq \varepsilon$ go to Step 8.

Otherwise go to Step 4.

STEP 4. Compute a search direction Δx_S^k.

STEP 5. Solve $B\Delta x_B^k = - S\Delta x_S^k$.

Set

$$\Delta x^k = \begin{pmatrix} \Delta x_B^k \\ \Delta x_S^k \\ 0 \end{pmatrix}.$$

STEP 6. Find $\alpha^* \geq 0$, the greatest value of α for which $x^k + \alpha\Delta x^k$ is feasible.

STEP 7. Find $\gamma \in [0,\alpha^*]$ so that

$$f(x^k + \gamma\Delta x^k) \leq f(x^k).$$

Define $x^{k+1} = x^k + \alpha\Delta x^k$.

If $\gamma < \alpha^*$ make $k = k+1$, go to Step 1.

If $\gamma = \alpha^*$, we conclude that a basic or a superbasic variable hit a bound. If it is a superbasic, change it to non-basic make $s = s - 1$, $k = k+1$ and go to Step 1.

If it is basic, a change of basis is necessary. Find a superbasic variable to enter the basis and declare the basic variable that hit the bound non-basic. Make $s = s - 1$, $k = k+1$ and go to Step 1.

STEP 8. ("Price", i.e., estimate Lagrange multipliers). Compute

$$\lambda = \nabla x_N f(x^k) - N^T \pi^k.$$

If the Kuhn - Tucker necessary conditions of optimality are satisfied, x^k is an optimal solution of our problem. Then STOP.

Otherwise, choose a non-basic variable not satisfying the optimality conditions, to release it from its bound. Declare it superbasic, make $s = s + 1$ and go to Step 4.

The next sections show the basis factorization and the updating procedure used for the matrices with staircase structure.

4. BASIS FACTORIZATION

Any submatrix B of A satisfies

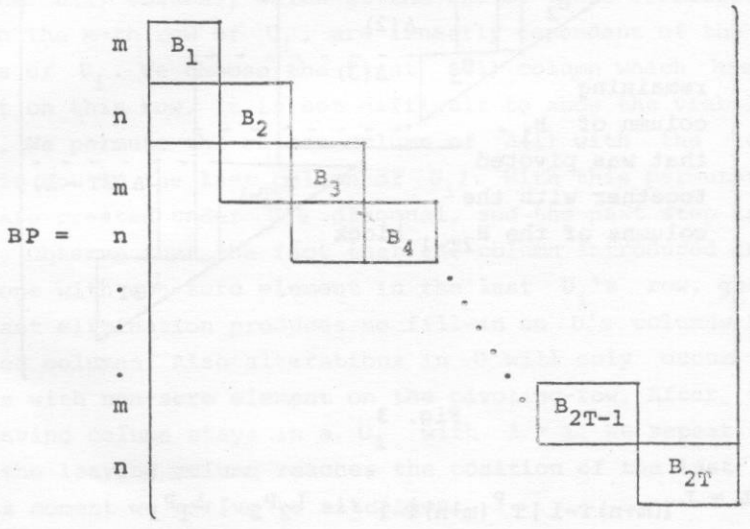

Fig. 2

for some permutation matrix P.

Let B be a basis, then the block B_1 must have m independent columns. We perform a stable row-pivoting within this (m × m) submatrix of block B_1. As the number of columns of B_1 can be greater than m, after this process we may have some columns of B_1, that remain unpivoted.

Let $\Delta(1)$ be the submatrix formed by the remaining columns of B_1, where we take just the first m-rows. To continue the factorization,

consider the submatrix whose columns are the remaining columns of B_1 and the columns of B_2, and whose rows are the first n-rows after the last pivoted row. Once again the fact that B is a basis guarantees the performance of a stable row-pivoting within this submatrix. We continue this process until the whole matrix B is factorized. Finally we have

$$L^{-1}BQ = U$$

where

$U =$

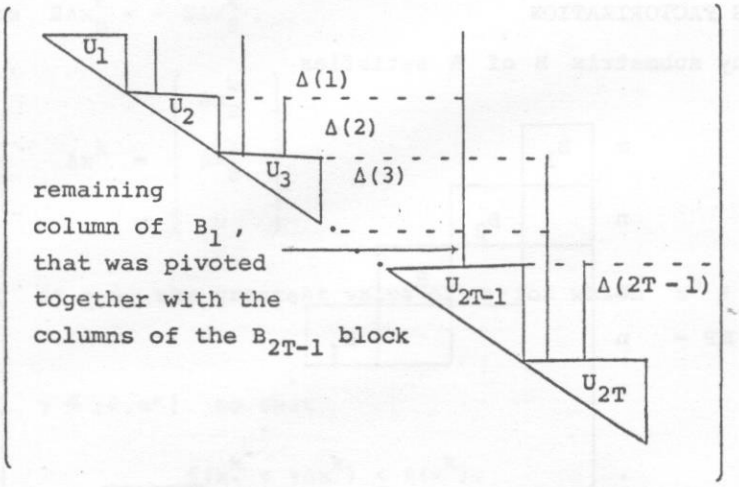

Fig. 3

$$L = L_{[(m+n)T-1]}\, T^{P}_{(m+n)T-1} \cdots L_2 P_2 \cdot L_1 P_1 ,$$

the L_i are elementary matrices, and the P_i elementary permutation matrices. Q is a permutation matrix. $\Delta(i)$ is the matrix whose columns are the remaining ones and whose rows are the corresponding rows of U_i, $1 \leq i \leq 2T-1$. The whole process will not produce fill-in outside the staircase except for the remaining columns. Note that the columns of any U_i, $0 \leq i \leq 2T$ may represent state or control variables from any previous stage.

5. UPDATING THE LU FACTORIZATION

To describe the updating process, we have to distinguish the following two situations:

i) The leaving column stays in U_i, and the entering column is associated with U_j, with $j > i$. Let k be the position of the leaving column relative to U_i, which we suppose of dimension m. We bring the columns $(k+1, k+2, \ldots, m)$ of U_i back to positions $(k, k+1, \ldots, m-1)$ respectively, and column k is put in place of the m-th column of U_i. This permutations transform the lower-triangular U_i in a Hessenberg matrix H_i. The H_i matrix is pivoted, in order to eliminate the sub-diagonal, choosing the element with largest absolute value between the diagonal and sub-diagonal in each column as pivot (Bartels-Golub, (1969)). These operations increase the number of elementary L_i that define L^{-1}. Clearly, these operations involve only the rows of B, corresponding to U_i. The $\Delta(i)$ columns, which at the end of these eliminations have a zero in the m-th row of U_i, are linearly dependent of the first $(m-1)$ columns of U_i. We choose the first $\Delta(i)$ column which has a non-zero element on this row. It is not difficult to show the viability of this choice. We permute the chosen column of $\Delta(i)$ with the leaving column (that is now in the last column of U_i). With this permutation, some elements are created under U's diagonal, and the next step is their elimination. Observe that the fact that the column introduced in U_i, was the first one with non-zero element in the last U_i's row, guarantees that this last elimination produces no fill-in on U's columns between the two permuted columns. Also alterations in U will only occur on the $\Delta(i)$ columns with non-zero element on the pivoting-row. After this process the leaving column stays in a U_ℓ with $\ell > i$. We repeat this procedure until the leaving column reaches the position of the last U_j's column. At this moment we arrive to situation:

ii) The leaving column is in U_i and the entering column is associated with U_j, $j \leq i$. We proceed just as in situation i. The difference is that, in this case, it may happen that the whole $\Delta(i)$ row, in which we look for a non-zero element, is null. If this is the case, the leaving column is deffinitely retired from U and replaced by the updated entering column. Once again, this change of columns, creates elements below U's diagonal, but the fact that the whole $\Delta(i)$'s pivoting row is null, guarantees that no fill-in is produced in U when those elements are eliminated. If $\Delta(i)$ has a non-zero element on the pivoting row we proceed as in i), repeating the process until for some $\ell \geq i$, $\Delta(\ell)$'s pivoting row is null or $\ell = 2T$. Then we may effectivize the change of

columns. It is clear that the whole process preserves the staircase structure with exception of the "remaining columns" of the factorization, and some new $\Delta(i)$ type columns that may be introduced during the updating process. Also some of these $\Delta(i)$ columns may be deleted during this process.

6. COMPUTATIONAL EXPERIENCE

The algorithm presented in the previous sections was implemented in a Fortran program and has been tested in an application to the optimal scheduling of the hydrothermal power system of the São Francisco River in Brazil. This system consists of four hydraulic turbines, two of them with reservois, Sobradinho and Moxotó. The others are Paulo Afonso I, II, III (P. A. I, II, III) and Paulo Afonso IV (P.A. IV).

The variables considered in the modelling of this system are:

x_i^t : storage of reservoir i at stage t;

y_i^t : natural inflow of reservoir i at stage t (known constant);

u_1^t : water released from reservoir 1 (Sobradinho) at stage t directed to the power plant to produce electricity;

v_1^t : water spilt from reservoir 1 at stage t;

u_2^t : water released from reservoir 2 (Moxotó) directed to the turbines of Moxotó and P.A. I, II, III to produce electricity;

v_2^t : water spilt from reservoir 2 and P.A. I, II, III;

u_3^t : water released from reservoir 2 directed to P.A. IV to produce electricity;

v_3^t : water spilt from P.A. IV;

τ : delay in the water flow from reservoir 1 to 2;

T : final stage.

$\underline{x}_i$, $\bar{x}_i$: lower and upper bounds on x_i^t $\quad i = 1,2;\ t = 1, \ldots, T$

$\underline{u}_i$, $\bar{u}_i$: lower and upper bounds on u_i^t $\quad i = 1,2;\ t = 0, \ldots, T-1$

d^t : energy demand to be met at stage t.

For this system we obtained the total energy production at a given stage as a quadratic function of the flows u_i^t.

$$g(u^t) = -0.00853(u_1^t)^2 + 8.3u_1^t + 32.98u_2^t + 34.93u_3^t .$$

The optimization problem we formulated is:

$$\min \sum_{t=0}^{T-1} (g(u^t) - d^t)^2 \tag{1}$$

subject to

$$x_1^{t+1} = x_1^t + y_1^t - u_1^t - v_1^t \tag{2}$$

$$x_2^{t+1} = x_2^t + y_2^t + u_1^{t-\tau} + v_1^{t-\tau} - u_2^t - u_3^t - v_2^t - v_3^t \tag{3}$$

(P)
$$\underline{x}_i \le x_i^t \le \bar{x}_i \qquad i = 1,2;\ t = 1,\ldots,T \tag{4}$$

$$\left.\begin{aligned} \underline{u}_i \le u_i^t - \bar{u}_i \\ v_i^t \ge 0 \end{aligned}\right\} \quad i = 1,2;\ t = 0,1, \ldots, T-1 \tag{5, 6}$$

$$x_i^o \quad \text{known constants} \tag{7}$$

An initial feasible solution was obtaned solving the linear programming problem

$$\min \sum_{t=0}^{T-1} \Delta^t \tag{8}$$

subject to (2), (3), (4), (5), (6), (7) and

$$\bar{g}(u^t) + \Delta^t - d^t = 0 \tag{9}$$

$$\Delta^t \ge 0 \tag{10}$$

where $\bar{g}(u^t) = 8.26u_1^t + 32.98u_2^t + 34.93u_3^t$, is a linear approximation of $g(u^t)$ obtained by fitting it's non-linear part with 1000 points. Δ^t may be interpreted as energy produced by other turbines than the hydraulic or as energy shortage.

The problem (P) was run on a VAX/785 with VMS operational system, for a two-week horizon discretized in periods of 8 hours, thus T=42. It converged to an optimal solution in 120 iterations consuming 0.7 CPU seconds iteration.

REFERENCES

BARTELS, R. H., and GOLUB, G. H. (1969). The simplex method using LU decomposition, Comm. ACM, 12, pag. 266 - 268.

FRIEDLANDER, A., MEDINA, E. L., TAVARES, M. M. F. (1982). A method for solving Linear Dynamic Programming Problems, XI International Symposium on Mathematical Programming, Bonn.

FORREST, J. J. H., and TOMLIN, J. A. (1972). Updating triangular factors of the basis to mantain sparsity in the product form simplex method, Mathematical Programming, 2, pag. 263 - 278.

FOURER, R. (1979). Sparse Gaussian elimination of staircase linear systems, Technical Report SOL 79 - 17, Systems Optimization Laboratory, Dept. of Operations Research, Stanford University.

FOURER, R. (1982). Solving staircase linear programs by the simplex method, 1: inversion, Mathematical Programming 23, pag. 274 - 313.

FOURER, R. (1983). Solving staircase linear programs by the simplex method, 2: Pricing, Mathematical Programming 25, pag. 251 - 292.

FOURER, R. (1984). Staircase Matrices and Systems, SIAM Review, Vol. 26, Nº 1, pag. 1 - 70.

KRIVONOZHKO, V. E. and CHEBOTAREV, S. P. (1976). Factorization method for linear dynamic programming problems developening systems. Automation and Remote Control, Nº 7.

MURTAGH, B. A. and SAUNDERS, M. A. (1978). Large-scale linearly constrained optimization, Mathematical Programming 14, pag. 41 - 72.

PROPOI, A. and KRIVONOZHKO, V. E. (1978). The simplex method for Dynamic Linear Programs - Report RR - 78 - 14 - Int. IASA, Laxenburg, Austria.