CONCAVE PROGRAMMING UNDER THE SIMPLIEST LINEAR BOUNDARIES

A.I. RUSAKOV

1. Problem formulation and the general frame of method

We present the program of solving the following concave problem:

$$\varphi(y) \longrightarrow \max; \tag{1}$$

$$\sum_{i=1}^{m} k_i y_i = A \tag{2}$$

$$\underline{y}_i \le y_i \le \bar{y}_i, \ i = 1 \div m. \tag{3}$$

where $y \in E^m$; $\varphi(y)$ is a continues convex function defined on E^m ; $\underline{y}_i < \bar{y}_i$ and $k_i \le 0$ for any i; parameters A, $\underline{y}_i$, $\bar{y}_i$, k_i are such that feasible set M, defined with restrictions (2–3), is not empty and not reduced to the point.

By the conversions:

$$x_i = k_i y_i \tag{4}$$

the problem (1–3) takes the following form investigated below:

$$f(x) \equiv \varphi(x_i/k_i \ , \ \dots \ , \ x_i/k_i) \longrightarrow \max; \tag{5}$$

$$\sum_{i=1}^{m} x_i = A \tag{6}$$

$$\underline{x}_i \le x_i \le \bar{x}_i, \ i = 1 \div m. \tag{7}$$

Any set in E^m, defined by (6–7), is denoted M as before. The denotation's meaning is made out by the context of derivation.

Let us consider following objects on the hyperplane: the feasible polyhedron M defined by restrictions (6-7); the vertex of the polyhedron M; the flat cone; the flat simplex; the rib of a polyhedron (and the rib of a cone in that number). All this objects may be adduced in the strict terms, but we shell define them through isomorphous objects in the $(m - 1)$-dimensional space, obtained by exclusion of the variable:

$$x_m = A - \sum_{i=1}^{m-1} x_i \ .$$

The presented description of the method is very rough; in particular, possi-

ble degeneration of vertices is not considered. The method is founded on the scheme of construction of the cutting simplices (Hoang Tuy, 1964). There will be under consideration coherently oriented (along the coordinate axes) hyperparallelepipeds (HP) defined by the restrictions likewise (7). The solution is obtained with successive diminishing of the size of the HP, restricting the region for seeking of maximum. The solution algorithm takes the generalized form as follows:

ALGORITHM 1

1. Make setting: Rec = $-\infty$.
2. Execute the cutting algorithm:

2.1. With a simplex-method obtain the supposed maximum of the function f (some vertex x_A of the feasible polyhedron M). If $f(x_A) >$ Rec then make settings: Rec = $f(x_A)$; $x_{\text{opt}} = x_A$.

2.2. Fix the guiding vectors of the cone ribs, diverging from the point x_A. Construct the hyperplane cutting on the cone some simplex for which the goal function has a maximum at the cone vertex. Test unexplored domain for one is not empty. If one is empty than the domain M has been explored: the point of maximum x_{opt} has been searched.

2.3. Construct HP having the shortest ribs and containing the unexplored region of feasible polyhedron. The constructed HP is given by lower $\underline{x}'_i$ and upper $\bar{x}'_i$ boundaries, $i = 1 \div m$.

3. Make settings: $\underline{x}_i = \underline{x}'_i$, $\bar{x}_i = \bar{x}'_i$, $i = 1 \div m$.

4. Make (if necessary) the bisection of HP and do the recursive call (see later).

5. Go to the item 2.

Operation 2.3, resulting in HP of smaller size, later is named the contraction of the initial HP. Operations 2.3 and 3 modify the boundaries in the inequalities (7) to get the unexplored part of the feasible polyhedron. The step 4 prevents the recycling if the contraction would slow down. With proper input data this procedures are not activated. The step 4 includes:

4.1. The monitoring of contraction speed. On the slow down process the following points 4.2 - 4.5 are carried out else going to the point 5 of algorithm 1.

4.2. The bisection of HP: the special choosing of hyperplane No. i defined with the equation:

$$x_{i_0} = \frac{1}{2}(\underline{x}_{i_0} + \bar{x}_{i_0}).$$

4.3. The choice of HP's half and solving the problem (5 - 7) for chosen half by the recursive call of algorithm 1. The optimization is resulted in the variable Rec1 and vector x^1_{opt}.

4.4. If Rec1 > Rec then set Rec = Rec1, $x_{\text{opt}} = x^1_{\text{opt}}$.

4.5. The boundaries $\underline{x}_i$, $\bar{x}_i$ are modified to specify another half for further exploration.

Obviously, after the first recursion it is possible the repeat of slowdown of contraction, which causes the bisection etc. The recursions are not deep through adjustments of algorithm; therefore process is quick, unrelated to the problem's dimension *m*.

To solve the problem (1 - 3) one has to set up:

the program module for the function $\varphi(y)$ calculation;

the parameters A, $\underline{y}_i$, $\bar{y}_i$, k_i;

the tuning parameters Δ_1, $\Delta\varphi_0$, named permissible errors for the argument and the function respectively, which satisfy to following demands:

1) almost at the whole feasible set the derivatives $\dfrac{\partial\varphi}{\partial y_i}$ is calculated as the ratio of increments $\dfrac{\Delta\varphi}{\Delta y^1_i}$ with a small error, where

$$\Delta y^1_i = \Delta_1(\bar{y}_i - \underline{y}_i); \tag{8}$$

2) for any generated vertex y^0 the component y^0_i belonging to the gap $(\underline{y}_i + \Delta y^1_i,\ \bar{y}_i - \Delta y^1_i)$ can't coincide with the boundaries $\underline{y}_i$, $\bar{y}_i$ on the chance of absolutely precise computations.

3) for the permissible optimization error $\Delta\varphi_{\max}$ next inequality takes place:

$$\Delta\varphi_{\max} \le \Delta\varphi_0 + \max_{y,\Delta y} |\Delta\varphi|,$$

where $\Delta\varphi$ is the finite increment of the function φ; maximum is searched for any vector $y \in M$ and vector Δy, such that $|\Delta y_i| \le \Delta y^1_i$, $i = 1 \div m$.

It is not recommended to take the considered parameters excessively small, for the latter can slow down the process. Setting this parameters to zero can produce abnormal termination;

the tuning parameter Δ_2, which means permissible relative error of argument optimization during calculations of the bounds of the truncated HP. Recommended values of this one are $\Delta_2 = 0.001 \div 0.00001$. Change this value only if some warn-

ing messages appeared one after the other, or program terminated abnormally.

2. The installation instructions and program interface

The package of programs was elaborated under MS-DOS. The package can be installed into the directory with any name: you have to copy there SFX-archives “conc-sfx.exe”, then unpack one. You’ll get following directories:

DATA – contains the input and output files;

LOAD – created to room the resulting optimization program “concave.exe”;

OM – serves to room the objective modules and libraries for linking of the resulting program, as well to include the linkage editor service, in particular, the command files "BAT";

SSM – serves to room the source-statement modules for the function calculation.

After unpacking the enumerated directories contains the set of files, related to the testing task, described below. Some data of this files, in particular, the optimization parameters from the file “sysin.cnc”, have to be used to solve user’s task.

The program “concave.exe” solves the problem (1 - 3) and interact with the user by means of the input file “sysin.cnc”, the output file “result.cnc”, the process monitoring on the screen, the objective module of the calculation of the function $\varphi(y)$. The input and output files are written in the ASCII-format. The separating agents are blank spaces (one or more) or <CR>. Comment lines may be used in the files. Non-normalized parameters, which were denoted above “y”, are denoted in comments like “X”.

The structure of the file “sysin.cnc”:

1-st line is a comment;

2-nd line is the problem dimension m;

3-rd line is a comment (the table’s heading);

following m lines have the form “$i \quad \underline{y}_i \quad \overline{y}_i$” and define the boundaries for the coordinates y_i;

(m + 4)-th line is a comment (the table’s heading);

following m lines have the form “$i \quad k_i$” and define the condition (2);

from the (2m + 5)-th line follow: the comment, the value A, the comment, the value Δ_1, the comment, the value $\Delta\varphi_0$, the comment, the value Δ_2 — all units are separated with <CR>.

The structure of the file “result.cnc”:

the beginning of the file is a list of input data with comments, then the line “THE RESULT OF OPTIMIZATION:”, then the table with coordinates of the optimal vector y, after that the optimal value of the function φ (the string “FUNC.VALUE = <value>”) and the algorithm setup parameters. The later para-

meters are: the summary number of recursions (SUM OF RECURSIONS = <value>); the maximum depth-level of recursions (DEPTH OF RECURSION = <value>); the summary number of failures while constructing the cones from the chosen vertices (SUM OF SUPPORT CONSTR.FAILURES = <value >), the timing of start and finish.

The optimization process is monitored on the screen. At the beginning all input data from “sysin.cnc” are echoed, then in the first (upper) line is displayed the percentage of the explored feasible domain and maximum criterion’s value reached to the current time. Below there is the scale and the indicator of the current recursion level (the greatest depth-level is 40). After that the “least” of input data follow. On the abnormal operations the corresponding statements is appearing by scrolling under the scale. The number of failures while constructing the cone, if not zero one, is printed at the first line by the right.

If current recursion level is greater, in average, then 15÷20, then computations will be excessively long, and process-tuning parameters has to be corrected. The program can’t be interrupted from the console and therefore it is recommended to start one under WINDOWS.

The screen is cleared before termination and computational results are displayed just like they placed into the file “result.cnc”.

3. Function programming, linking and testing

The objective modules in the directory OM together with command files make useful the procedures of calculation of the function $\varphi(y)$ almost in any programming language under MS-DOS (it is recommended compilers by Microsoft corp. produced in 1985-1990). The directory SSM contains the source-statements modules of calculation of the testing function like follows:

$$\varphi(y) = \sum_{i=1}^{m} [(m - i + 1) y_i + \frac{\sqrt{i}}{m} {y_i}^2],$$

in the languages FORTRAN77 v3.31 (1985) and QuickBASIC v4.5 (1988). Given files are presented as the samples of programming of user’s functions. In the program under consideration the external reference to the procedure-function is defined as FUNC, the returning value is of float type, 4 bytes long. To transfer parameters into the subroutine FUNC it is applicable the “subroutine-link” being named RELAY, included into the main objective module “concave.obj”. Given procedure-function is called from the user’s procedure FUNC at the beginning of runtime and returns with 4 bytes floating point format: at the first call - the dimension m, at the following m calls - the coordinates y_i, $i=1 \div m$. Such the unwieldy transference of parameters makes procedure FUNC compatible with supplied objective modules indifferently to the language of programming of procedure

FUNC. After calculation of the function's value the return into the parental module is made in common technique - see the examples "func-f.for" (written in FORTRAN) and "func-b.bas" (written in QuickBASIC). To compile the source modules "func-b.bas" and "func-f.for" the default options for keys and libraries are applicable. The compiled modules have been placed into the directory OM ("func-b.obj" and "func-f.obj" correspondingly) and may be used for training in the assembling technique. Linking of the objective module "func-b.obj" can be done with command file "link-bas.bat", started from the current directory OM. The result of linkage is the file "concave.exe" that uses a coprocessor. Linking of the module "func-f.obj" can be done with starting either the file "link-f-e.bat" to create the file "concave.exe" that uses simulation of coprocessor 8087, or the file "link-f-m.bat" to create the similar file that uses a real coprocessor. All modules and libraries required for linkage have been placed into the directory OM. The resulting file "concave.exe" you have to displace into the directory LOAD and it's ready to start. (From this directory you can remove the demo-program with procedure FUNC compiled from file "func-b.bas"). If you replace in the OM the demo-objective module of computation of the function $\varphi(y)$ by yours one (saving, of course, names func-b.obj or func-f.obj) then by starting corresponding command files you can create the wanted optimization program. However, you have to utilize one of above-mentioned languages with a pointed version, otherwise you have to use another libraries. To do correct original linkage it is presented below a list of files in the directory OM and operation instructions for the program "link.exe".

Subdirectory OM includes:

objective modules of the elaborated program in QuickBASIC v4.5: "concave.obj" and "service.obj";

"link-bas.bat" that is the command file of linking of "clear" BASIC-programs;

"link-f-e.bat", "link-f-m.bat" that are the files of linking of mixed-languages programs using FORTRAN-procedure;

"func-b.obj" and "func-f.obj" that are objective modules of computation of testing function (see above).

In the starting command of linker you have to use the keys /NOE (ignore extensions of libraries dictionaries) and /NOD (ignore default libraries). Link together modules concave.obj + service.obj + noem.obj + {procedure-function with entry point FUNC}, include "bcom45.lib" and least number of standard libraries to resolve the references of user's function, moreover a good deal of references (may be all) are resolved with a library "bcom45.lib".

Remark: in the freeware version you are limited with dimensions $m \le 20$.

The theory of method is in the article:

Rusakov A.I. Concave Programming under Simplest Linear Constraints. Computational Mathematics and Mathematical Physics, Vol. 43, No. 7, 2003, pp. 908–917.

mailto:rusakov@rostel.ru
Phone 007-8632-63-58-32

Alexander Rusakov