# U.S. Department Of Energy

In Situ Remediation Integrated Program
Office of Research and Development, EM-541

# MODLP

## Program Description:

## A Program for Solving Linear Optimal Hydraulic Control of Groundwater Contamination Based on MODFLOW Simulation

Version 1.0

University of Connecticut,
Research Center for Groundwater Remediation Design

University of Vermont,
Research Center for Groundwater Remediation Design



Lawrence Livermore National Laboratory
Environmental Restoration/Waste Management-Advanced Technologies Program

## DISCLAIMER

# MODLP Program Description: A program for Solving Linear Optimal Hydraulic Control of Groundwater Contamination Based on MODFLOW Simulation

**David P. Ahlfeld and David E. Dougherty**

Version 1.0
December 1993 (Last Revision January 1994)

---

MODLP was programmed by David P. Ahlfeld, Department of Civil Engineering, University of Connecticut, U-37, Storrs, CT 06269.

This work was supported by the Research Center for Groundwater Remediation Design at the University of Vermont and the University of Connecticut. Partial support was provided by the U.S. Department of Energy In Situ Remediation Integrated Program (ISR IP) under a project entitled "Optimal Remediation Design: Methodology and User-Friendly Software for Contaminated Aquifers", Subcontract B-239648 from the Lawrence Livermore National Laboratory.

## Abstract

MODLP is a computational tool that may help design capture zones for controlling the movement of contaminated groundwater. It creates and solves linear optimization programs that contain constraints on hydraulic head or head differences in a groundwater system. The groundwater domain is represented by USGS MODFLOW groundwater flow simulation model. This document describes the general structure of the computer program, MODLP, the types of constraints that may be imposed, detailed input instructions, interpretation of the output, and the interaction with the MODFLOW simulation kernel.

---

This work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. This work is funded by the Office of Technology Development within the Department of Energy's office of Environmental Management under the *In Situ* Remediation Integrated Program

## 1. INTRODUCTION

MODLP is a computational tool for use in groundwater remediation design. It is a FORTRAN program that incorporates the well known and widely used MODFLOW simulator to represent flow of water in a saturated natural porous medium. MODLP is designed to allow the user to create and solve optimization problems for hydraulic control in groundwater systems. This is accomplished by coupling the groundwater flow simulation MODFLOW with a linear programming (LP) optimization solver. Solving optimization problems involves two steps. First, the simulator is designed and calibrated to match the conditions in the system under study, using available data. The simulator is expected to provide an acceptable representation of the response of the field system to alternate pumping strategies. Second, LP optimization is used to solve for the set of pumping rates and locations that minimizes the weighted sum of pumping while simultaneously satisfying a number of user-specified constraints.

A description of MODFLOW [MacDonald and Harbaugh, 1984] or of its use is far beyond the scope of this document. Tersely, MODFLOW is a three-dimensional, transient or steady-state, saturated groundwater flow simulator developed by the U. S. Geological Survey. It may incorporate a single water table and a variety of commonly encountered, linear boundary and source conditions. The MODFLOW model does not represent solute transport, although it has been extended in a number of proprietary commercial codes to do so.

Users of MODLP are assumed to be familiar (and reasonably comfortable) with simulating groundwater flow using MODFLOW.

MODLP provides a "wrapper" around MODFLOW so that a calibrated groundwater flow model may be interrogated to assess the least cost solution to a groundwater cleanup problem that depends linearly upon the decisions to be taken and for which a set of constraints exist that are linear in the decisions. In its current form, MODLP accommodates vertical or horizontal injection or extraction wells (the wellbore physics are extremely simplified). It is applicable only to confined and leaky aquifer problems (phreatic surface problems are nonlinear and cannot be treated using the LP method).

Specification of the optimization problem requires that the user perform several steps:

1) A set of numerical discretization nodes to be considered as candidates for remediation pumping are selected.
2) The unit costs for injection and extraction pumping at each candidate well are specified.
3) Requirements on the hydraulic behavior of the groundwater system during remediation are selected from a set of possible constraints on head and pumping rates.

## 2. COUPLING WITH MODFLOW

As previously mentioned, it is assumed that the user of MODLP has already constructed, calibrated, and verified a site-specific model of groundwater flow. To successfully couple this model with MODLP, the MODFLOW must be restricted. The MODFLOW model may contain any of the features of MODFLOW with the following exceptions:

1) the model must contain one stress period (i.e., NPER=1).
2) the model must run to steady state (i.e., ISS=1).
3) the model must be of the constant transmissivity type (i.e., LAYCON=0 for all layers).

The first two exceptions are to maintain computational tractability, while the third is to maintain the linear response of head required by the linear programming formulation.

MODLP contains a complete copy of MODFLOW embedded within it. Only the main program unit, or driver, of MODFLOW has been modified so that it can be invoked repeatedly by the optimization routine. Therefore, a practical application of MODLP follows the construction and calibration of a MODFLOW model. Once an acceptable design-worthy MODFLOW model has been completed, its input files can be used directly by MODLP without modification. A separate input file specific to MODLP must be created for the optimization runs. Note, however, that the MODFLOW input data files are not at all affected by use of MODLP, so they can be preserved for archival and recalibration purposes.

## 3. CONSTRAINTS AND OBJECTIVE FUNCTIONS IN MODLP

The optimization portion of MODLP always seeks the pumping strategy with the least "cost" associated with it. This cost is defined by the user by placing units costs for injection and extraction pumping at each candidate pumping well. The total cost that is minimized is measured as the sum of the products of the unit costs at each well and the flow rate at the well. Mathematically, the objective function is represented by the weighted sum of withdrawal and recharge rates as

$$\sum_{j \in J} \beta_j \, q_j \qquad (1)$$

where $\beta_j$ is the relative cost coefficient that related the rate of pumping to the cost of pumping at location j. Note that if injection or extraction are considered at a single spatial location, then there are two decision variables (and two terms in the objective function) corresponding to that one point in space. The use of this cost function presumes that the cost of operation of the hydraulic control system is linearly proportional to the rate of pumping.

Input requirements for each of these constraint types are described below.

MODLP can define two types of constraints on the hydraulic head and one kind of constraint on the pumping rates.

1. An absolute bound may be placed on heads at a point:

$$h_i \geq h^*_{i,l} \qquad (2)$$

or

$$h_i \leq h^*_{i,u} \qquad (3)$$

where $h^*_{i,l}$ and $h^*_{i,u}$ are specified lower and upper values and $h_i$ is the head at a node where control is to be imposed. Depending on the direction of the inequality, the type of constraint can be used to control excessive drawdown (equation 1a) or mounding (equation 1b) of the piezometric surface.

2. Requirements may be placed on the difference in heads:

$$h_{i1} - h_{i2} \geq d^*_i \qquad (4)$$

where $d^*_i$ is the specified bound on head differences, and $h_{i1}$ and $h_{i2}$ are the heads at (often adjoining) node points *i1* and *i2*, respectively. This constraint can be

used to force a gradient in the hydraulic flow field, either horizontally or vertically.

3. Constraints commonly are placed on withdrawal and recharge rates to express bounds on well yield or limitations on the rate at which water can be recharged. Bounds can be imposed on flux rates, either withdrawal or recharge, as follows

$$q_j \leq q_{jmax} \qquad (5)$$

where $q_j$ is an individual flux rate at location j and $q_{jmax}$ is the maximum recharge or withdrawal rate respectively that can be achieved at location j.

## 4. PROGRAM DISTRIBUTION

MODLP is distributed with all FORTRAN source code, together with a sample input set and the corresponding outputs. The source code has been successfully compiled and executed on SUN4, SGI, and IBM RS/6000 workstations and servers, and on DOS/Windows computers.

### 4.1 Program Structure

The MODLP package consists of a series of modules written in FORTRAN 77, each of which contains a number of subroutines. The module *modlp.f* contains the main program, the input and output subroutines, and the subroutines which construct the linear programming problem.

As part of the formulation of the linear programming coefficient matrix, MODFLOW is called repeatedly. Module *calmod.f* contains the modified main program of MODFLOW (which enables it to be called as a subroutine) and other utility subroutines related to the repeated call of MODFLOW. Modifications to the driver routine of MODFLOW are indicated by lower case text in *calmod.f*, while the original MODFLOW main routine appears in upper case. The remainder of MODFLOW is contained in module *modsub.f*. The subroutines contained in *modsub.f* are unmodified from the copy of MODFLOW distributed by the International Ground Water Modeling Center (Colorado School of Mines, Golden, CO). If an updated version of MODFLOW is incorporated into MODLP, the lower case lines in *calmod.f* indicate most of the changes that will be required to embed the new MODFLOW into MODLP; all other subprograms in the updated MODFLOW replace the current *modsub.f*.

A unique feature of MODLP is the capability for automatic generation of head difference constraints by specification of the outline of a capture zone. The translation of the capture zone definition to constraints is accomplished in module *mkpair.f*.

Once the linear program is formulated, its solution is accomplished using the simplex algorithm [Luenberger, 1984]. Module *lpsub.f* contains an implementation of the simplex algorithm that does not contain many of the efficiency measures and other features present in sophisticated commercial simplex codes. This code is intended for research and educational purposes. Its use for other purposes should be considered with caution.

The program can also generate a standard MPS file which fully describes the LP and can be read by many simplex solution codes. This implementation has been successfully coupled with MINOS 5.1, a product of Stanford University, for example.

### 4.2 Installing MODLP

To install MODLP on a computer system, choose a directory in which to place the MODLP subdirectory. This directory will be referred to as *$MODLP_HOME* in this section. Set the current directory to *$MODLP_HOME* (*cd $MODLP_HOME*), create a subdirectory *modlp* (*mkdir modlp*), and move to it (*cd modlp*). Copy all files from the distribution diskette into the

directory. Note that the diskette is provided in DOS format; use on another type of computer may require a small change (usually automated on each type of computer) for file format conversion.

Included in the distribution is a file named *Makefile*. This file can be used on Unix systems to compile and link MODLP using the *make* utility. If the FORTRAN compiler is called *f77* and no special libraries are needed, simply type *make*. MODLP will be compiled and linked, and the resulting executable file named *modlp* will be created in the directory *$MODLP_HOME/modlp*. If the compiler has a different name (e.g., on IBM RS/6000s it is called *xlf*) or if some special library is required, then *Makefile* must be edited. At the top of the file are two lines, one starting with the string "*FC* = " and the other starting with "*LLIBS* = ". Append to these lines the compiler or library names (ask a local expert for assistance).

For DOS, Windows, or Windows NT systems *Makefile* may be useful, or it may not be useful. This depends on the compiler that has been installed and whether it uses a *make* or *make*-like utility.

If you intend to use MODLP repeatedly, it will be useful to modify the PATH of the computer to include the *$MODLP_HOME/modlp* directory.

### 4.3 Customizing for MODFLOW Application

Manipulation of the various input files required by MODFLOW is performed in subroutine *modio* within module *calmod.f*. Here, the user may introduce OPEN statements for file definition. If the solver within *lpsub.f* is used, then all files used by MODFLOW are rewound in this subroutine at the beginning of the second simulation of flow using the optimized pumping rates.

### 4.4 Dimensioning

All dimensioning is performed at the top of module MODLP using parameter statements. Internal comments describe the meaning of each parameter. Most of the dimensioning parameters have internal error checking to insure that array dimensions are sufficiently large.

## 5. INPUT STRUCTURE FOR MODLP-SPECIFIC DATA FILE

The MODFLOW simulator that is embedded in the MODLP package takes input through standard MODFLOW input files as described in the MODFLOW documentation. Code to open MODFLOW input files can be placed in subroutine *modio.f* in module *calmod.f*, although the current implementation assumes that these files will be opened externally. Because MODFLOW is called repeatedly, the MODFLOW files must be rewound to permit re-reading. This is accomplished in subroutine *modio.f* within module *calmod.f* and must be coded by the user.

The optimization input is all contained in a single input file which has the name *opt.in*. The contents of *opt.in* fully describe the optimization formulation. All items in *opt.in* are read in free (i.e., list-directed) format. No adherence to column location is necessary. The units of values in opt.in must be consistent with the units used in the user's MODFLOW data set.

The file *opt.in* is organized into five groups of data. No special symbols or delimiters are used within the file to mark the beginning or end of a group.

The first 9 lines of the file *opt.in* comprise GROUP A of the input file. Group A data describe the optimization formulation and directions for its solution. The first line is a title that identifies the problem. The remaining lines contain numerical values describing the optimization problem. These 9 lines are described in the table below.

Note that the coordinates described in line 6 of Group A is only relevant if capture zone lines are defined in Group E. If capture zone lines are not used, then arbitrary values suffice in Group A, line 6. The origin of the coordinate system used in line 6 is assumed to coincide with the external corner of cell (1,1) of the MODFLOW grid. An angle of rotation equal to zero implies that the $x$ and $y$ coordinates are aligned with the columns and rows, respectively, of the MODFLOW grid.

| GROUP A LINE NUMBER | ENTRIES | DESCRIPTION |
|---|---|---|
| 1 | <s1> | String title/description line |
| 2 | <n1> | Integer number of candidate wells |
| 3 | <n1> | Integer number of head bound constraints |
| 4 | <n1> | Integer number of head difference constraints |
| 5 | <n1> | Integer number of lines of capture zone information |
| 6 | <r1> <r2> <r3> | <r1> = real $x$ coordinate of origin<br><r2> = real $y$ coordinate of origin<br><r3> = real angle of rotation of capture zone point coordinates relative to the grid |
| 7 | <n1> | Integer flag to calculate/reuse LP coefficient matrix (or response matrix).<br><br>A value of 1 means compute matrix from scratch using perturbation producing the file *jacob*.<br><br>A value of 0 means to use the existing matrix defined in the file *jacob*. This capability allows the user to repeatedly solve optimization problems with changes to constraint bounds or cost coefficients without recomputing the coefficient matrix.<br><br>If the number, order, or location of constraints or candidate wells changes, the coefficient matrix must be recomputed from scratch. |
| 8 | <n1> | Integer flag to select LP solver.<br><br>1 generates MPS deck for use with commercial solver.<br>2 uses the embedded LP solver *lpsub*. |
| 9 | <n1> | Integer maximum number of LP iterations. |

The next set of lines within *opt.in*, GROUP B, describes the candidate wells. The first line in the set is a descriptive, but unused, label. The remaining lines describe each candidate well. The values listed are row number, column number, layer number, flux direction index (1=injection, 2=extraction), upper bound on flux ($q_{i,u}$), and cost coefficient ($\beta_j$) used in (1). A lower bound of zero is assumed. If both injection and extraction are to considered at the same well, then two lines of input are needed, one for each sense. The upper bound on flux may be set to zero to "turn off" the candidate well during successive optimization runs while using the same LP coefficient matrix.

| GROUP B<br>LINE NUMBER | ENTRIES | DESCRIPTION |
|---|---|---|
| 1 | <s1> | Descriptive string |
| 2...<br>...# of<br>candidate wells<br>+ 1 | <n1> <n2> <n3> <n4> <r1> <r2> | <n1> integer row number<br><n2> = integer column number<br><n3> = integer layer number<br><n4> = flag for sense of pumping<br>1=injection<br>2=extraction<br><r1> = upper bound on flux ($q_{i,u}$)<br><r2> = cost coefficient ($\beta_j$) |

The next set of lines, GROUP C, describes the head bound constraints (2) and (3). The first card in the set is a dummy label. Each successive card describes a constraint. The first three values give the row, column, and layer location of the constraint. The absolute value of the last value gives the bounding head value. The sign of the last value has significance. If the sign is negative, the constraint (3) is implied; if the sign is positive then constraint (2) is implied. Thus, this implementation cannot impose bounds on heads below zero.

| GROUP C<br>LINE NUMBER | ENTRIES | DESCRIPTION |
|---|---|---|
| 1 | <s1> | String descriptor |
| 2...<br>...# of head constraints + 1 | <n1> <n2> <n3> <r1> | <n1> integer row number<br><n2> = integer column number<br><n3> = integer layer number<br><r1> = real head bounding value with sign used as flag.<br><br>If <r1> is negative, constraint (3) is implied.<br><br>If <r1> is positive, constraint (2) is implied. |

The next set of lines, GROUP D, describes the generalized head difference constraints (4). The first card in the set is a dummy label. Each successive card describes a constraint. The first three values give the row, column, and layer location of cell *i2* in (4), while the second three values give the row, column, and layer location of cell *i1* in (4). The final value is *d*. A value of -999 for *d* will deactivate the constraint. This may be useful during successive runs using the same LP coefficient matrix.

| GROUP D<br>LINE NUMBER | ENTRIES | DESCRIPTION |
|---|---|---|
| 1 | <s1> | String descriptor |
| 2...<br>...# of head difference constraints + 1 | <n1> <n2> <n3> <n4> <n5> <n6> <r1> | For node *i2* in (4):<br><n1> integer row number<br><n2> = integer column number<br><n3> = integer layer number<br><br>For node *i1* in (4):<br><n4> integer row number<br><n5> = integer column number<br><n6> = integer layer number<br><br><r1> = real head difference *d* in (4)<br><br>A value of -999 for *d* will deactivate the constraint. |

GROUP E, the final set of lines, describes the capture zone. MODLP translates the series of lines described here into head difference constraints of the form of (4). Thus, it automatically generates the head difference constraints for all cells along the edge of the capture zone. Input in this section again begins with a dummy card. Each card after this contains three values. If the first value is -9999, then this card describes the beginning of a new capture zone line. The layer in which the line lies is given by the value immediately after the -9999 value. Each capture zone line is described as a series of points. These points are given by the lines that follow the -9999 line and continue until a new -9999 line is reached or all capture zone lines are read. Each line that describes a point has three values. The first two are the x,y coordinates of the point. The last value gives the required gradient along the line formed by the current point and the next point. The gradient value of the last point in the line is ignored. The "inside" of the capture zone is defined to be the left side of the line when one is moving from one point to the next in the list.

| GROUP E<br>LINE NUMBER | ENTRIES | DESCRIPTION |
|---|---|---|
| 1 | <s1> | String description |
| 2...<br>...# of capture zone description lines + 1 | <g1> <g2> <g3> | If <g1> = -9999:<br><g1> = flag indicating start of new capture zone line<br><g2> = integer layer number in which capture zone lies<br><g3> = insignificant<br><br>If <g1> ≠ -9999:<br><g1> = real *x* coordinate of point on current capture zone line<br><g2> = real *y* coordinate of point on current capture zone line<br><g3> = real required gradient along the line formed between the current point and the next point |

## 6. OUTPUT INTERPRETATION

MODLP produces up to 4 output streams depending on the LP solver used. Input echo and information describing the generation of the coefficients are written to standard (usually terminal) output.

The output produced by MODFLOW is directed to a file called *modout*. This file will contain the initial MODFLOW solution, a convergence report on each solution used for generating the linear program coefficient matrix, and the final MODFLOW solution.

If the LPSUB solver is selected, then 2 files of output are produced. The main results file is named *solution*. It reports on the solution status and gives the binding constraints and their shadow prices. It also reports the active pumping locations and rates. This output is printed from subroutine OPTOUT within module MODLP and can be modified by the user. The output produced by the formulation solver is written to a file called *solveout*. Its contents give information on the changes to basis that occur at each iteration in the simplex algorithm.

If the MPS solver option is selected, then a file *mpsinput* is created. This file can be directly supplied to a commercial LP solver of the user's choice.

## 7. DISTRIBUTION TEST PROBLEM

A test problem for MODLP is included with the distribution. A unequally spaced mesh comprising 20 rows, 20, columns, and 2 layers is employed in steady state mode. The strongly implicit procedure (SIP) is used to solve the flow equations. Uniform recharge enters the top layer and head-dependent boundary fluxes occur on two opposing vertical faces of the model region. Wells are allowed at nine locations; five of them may be operated as injection or extraction wells, whereas the remaining four wells can be used for extraction only. The cost of operating each well per unit discharge rate is identical. The optimization problem is to minimize the total pumping rate from the nine wells, each of which has limits on pumping rate, such that a capture zone is created. In addition, there are three constraints to insure that heads remain below 35 feet at three spatial locations and three locations at which head differences (discrete versions of gradient constraints) are imposed. The input data are contained in file *sam.in*.

The origin of the coordinate system is in the lower left corner of the grid, in accordance with usual Cartesian coordinates. The indices of rows and columns of the grid also begin in that corner. The locations of constraints and candidate wells are given in the file *sam.in*. Note that the third line of input in the capture zone line group indicates a gradient requirement of -10000. This value is easily satisfied for the sample problem configuration. It effectively turns off the constraints along the line between points (3950, 2950) and (4750,2950).

Execution of the sample problem requires files defined with default name *fort*.n where n has values 1, 8, 11, 14, 15, 19, and 23. Execution produces output to the screen and to files *modout*, *solveout*, and *solution*. In addition, the program produces *jacob*, an unformatted file containing the linear programming coefficient matrix. This file is used for input to subsequent runs if the coefficient equations do not change. The program also produces *solution.all*, which shows more details of the LP solution. To insure proper execution of the program, the user should run MODLP with test problem described by *sam_opt.in*. The distribution files *solution.sam*, *solveout.sam*, and *modout.sam* should be identical (barring roundoff error due to compiler and hardware differences) to the corresponding files produced by such a test run.

The output from the sample problem indicates that only two of the constraints are binding at the optimal solution. All other constratints are satisfied as inequalities. The sample output also shows that only two of the 14 candidate wells have been selected and that both of them are operated as extraction wells.

## 8. REFERENCES

Luenberger, D. G., *Linear and Nonlinear Programming*, 2nd edition, Addison-Wesley, Reading, MA, 1984.

MacDonald, M.G., and A. W. Harbaugh, *A Modular Three-Dimensional Finite Difference Groundwater Flow Model (MODFLOW)*, U.S. Geological Survey, Reston, VA, 1984.