REPORTS
IN
INFORMATICS

ISSN 0333-3590

Sensitivity Analysis Applied to the Pooling Problem

Lennart Frimannslund, Geir Gundersen
and Dag Haugland

REPORT NO 380 December 2008

Department of Informatics
UNIVERSITY OF BERGEN
Bergen, Norway

Sensitivity Analysis Applied to the Pooling Problem

Lennart Frimannslund, Geir Gundersen and Dag Haugland
Department of Informatics
University of Bergen
Norway
{Lennart.Frimannslund, Geir.Gundersen and Dag.Haugland}@ii.uib.no

December 22, 2008

Abstract

In this report we discuss sensitivity analysis applied to the pooling problem. We apply both classical sensitivity analysis and new results by Castillo et al [1] to get sensitivities for all the input parameters. In addition we discuss some of the results in Greenberg [2] which relates sensitivity analysis specifically to the pooling problem. We finally provide some illustrative and relevant case studies.

1 Introduction

If the variables and/or the optimal solution of a linear or nonlinear programming problem instance vary greatly as a consequence of small changes in the coefficients of the objective function or the constraints, we say that the instance is *sensitive* with respect to these parameters.

To find out which parameters the instance is sensitive to we could change the initial data and solve the model again. This is feasible if the model is small and there are not too many parameters. If the problem is large we have to resort to sensitivity analysis. Classical sensitivity analysis, for both linear and nonlinear problems, is covered in most optimization textbooks, see for example [8, pp. 111–115], [6, pp. 330–331] and [4, pp. 312–313].

We will in this note present results on sensitivity analysis from [1], which especially gives objective function sensitivities with respect to parameters. We will also discuss the results in [2], which relates sensitivity analysis specifically to the pooling problem. These results are important since the sensitivities are not only obtained in an elegant and neat manner but also at low computational costs. For other relevant articles on sensitivity analysis we refer to [1] and the references therein.

This is a technical note that requires the reader to have fundamental knowledge of linear and nonlinear programming, in addition to be familiar with the concept of sensitivity analysis. For the reader without this background we refer to the above mentioned textbooks.

2 Classical Sensitivity Analysis

Sensitivity analysis is the study of how the change or error in the input data or parameters can change the output of a model. It enables us to determine which parameters have the largest effect on the output. That is, if a small change in a parameter will create a large change in the output, then the model is sensitive to that parameter. Next we briefly discuss sensitivity analysis for linear programming (LP), then we discuss sensitivity analysis for nonlinear programming (NLP) in more details since the pooling problem is an NLP.

Consider the LP-problem

$$\min c^T x \tag{1}$$
$$s.t.\ \ Ax \ = \ b, \quad x \geq 0. \tag{2}$$

where $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ are input data and $x \in \mathbb{R}^n$ contains the unknowns. If we solve (1) we can easily obtain the dual variables (sensitivities), also known as Lagrange multipliers [1, among others]. Simply put, the dual variables measure how much the objective function will be influenced by a change in the right-hand side of the corresponding constraint.

Now, consider the NLP-problem

$$\min f(x;a) \tag{3}$$
$$s.t.\ \ h(x;a) \ = \ b \tag{4}$$
$$g(x;a) \ \leq \ c \tag{5}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^m$, $g : \mathbb{R}^n \to \mathbb{R}^p$, $x \in \mathbb{R}^n$, $a \in \mathbb{R}^t$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^p$. The parameter a represents any coefficients of the left-hand side of the constraints and in the objective function. Suppose that x^* is a locally optimal solution of (3)–(5). Define the Lagrangian function as

$$\mathcal{L}(x,\lambda,\mu) = f(x) + \sum_{i=1}^{m} \lambda_i (h_i(x;a) - b_i) + \sum_{j=1}^{p} \mu_j (g_j(x;a) - c_j), \tag{6}$$

where vector subscripts define the corresponding components. The set of constraints which are satisfied with equality is called the *active set*, or the set of *active constraints*. Let an asterisk superscript (e.g. x^*) denote the optimal value of a variable. If the gradients of the active constraints at x^* are linearly independent[1] then there exist unique Lagrange multiplier vectors λ^* and μ^*, such that the following conditions, the KKT conditions, are satisfied at (x^*, λ^*, μ^*):

$$\begin{array}{rcl} \nabla_x \mathcal{L}(x^*,\lambda^*,\mu^*) & = & 0, \\ h_i(x^*) - b_i & = & 0,\ i = 1,\ldots,m, \\ g_j(x^*) - c_i & \leq & 0,\ j = 1,\ldots,p, \\ \mu_j^*(g_j(x^*) - c_j) & = & 0,\ j = 1,\ldots,p. \\ \mu_j^* & \geq & 0,\ j = 1,\ldots,p. \end{array} \tag{7}$$

If the LICQ condition does not hold, the Lagrange multipliers may still exist, but may not necessarily be unique. See for instance [5, Theorem 7.3.7].

Given a locally optimal solution x^*, the multipliers λ^* and μ^* indicate how sensitive the solution is to changes in the right-hand side of the constraints. One can imagine this as determining which constraints the objective function "pushes to" or "pulls from" the most. Assume x^* is the optimal solution to (3)–(5). If the solution is moved a distance ϵ, across an active constraint, say, $h_i(x;a) = 0$, then the rate of change in the optimal value is [6, equation 12.33]

$$\frac{df(x^*(\epsilon);a)}{d\epsilon} = -\lambda_i \|\nabla h_i(x^*;a)\|, \tag{8}$$

where $x^*(\epsilon)$ is the perturbed optimal solution. Since this is a derivative, it is valid as a first order approximation, and in addition *the set of active constraints is assumed to be the same as for the unperturbed solution.*

[1]This is called linear independence constraint qualification, or the LICQ condition.

Computing the multipliers If we have the value of x^* from solving (3)–(5), we can obtain the optimal values of the Lagrange multipliers in the following way. We have the linear system:

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = \nabla_x f(x^*; a) + \lambda^{*T} \nabla_x h(x^*; a) + \mu^{*T} \nabla_x g(x^*; a) = 0,$$

with the additional requirement that

$$\mu^* \geq 0,$$

and also that μ_j is zero for all j corresponding to inactive inequality constraints. This follows from the last equality in (7). The latter requirement can be formulated as a linear equation, so all in all we have to solve:

$$\begin{bmatrix} \nabla h_1 & \cdots & \nabla h_m & \nabla g_1 & \cdots & \nabla g_p \\ 0 & \cdots & 0 & \gamma_1 & \cdots & \gamma_p \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} -\nabla f(x^*) \\ 0 \end{bmatrix}, \tag{9}$$

subject to

$$\mu \geq 0. \tag{10}$$

Here γ_j is a vector of length p made up of zeros, except for an entry 1 in position j if g_j does not belong to the active set. This can for instance be solved with an LP solver, if we maximize an artificial objective function consisting only of zeros, subject to (9) and (10).

Not all of the sensitivities are obtained when we get the dual variables for LP and NLP, as the Lagrange multipliers give sensitivities with respect to b and c, but not to a. We will in the next section show how to obtain the parameter sensitivities related to a.

3 Objective function sensitivities with respect to parameters

Castillo et al in the paper [1] provide explicit formulas for sensitivities, among others for linear programming. These formulas provide sensitivities with respect to the variables, but also with respect to the input data.

The following Theorem from Castillo et al [1] gives the objective function sensitivities with respect to the parameter a:

Theorem 3.1. *The sensitivity of the objective function of the problem (3)–(5) with respect to the parameter a is given by*

$$\nabla_a \mathcal{L}(x^*, \lambda^*, \mu^*; a, b, c)$$

which is the partial derivative of its Lagrangian function

$$\mathcal{L}(x, \lambda, \mu; a, b, c) = f(x; a) + \lambda^T (h(x; a) - b) + \mu^T (g(x; a) - c)$$

with respect to a evaluated at the optimal solution x^*, λ^*, and μ^*.

Sketch of proof: The following formulation, where $\tilde{a} \in R^t$ is a decision variable, is equivalent to (3)–(5)

$$\begin{aligned} \min_x Z_p &= f(x; \tilde{a}) & (11) \\ s.t. \quad h(x; \tilde{a}) &= b, & (12) \\ g(x; \tilde{a}) &\leq c, & (13) \\ \tilde{a} &= a, & (14) \end{aligned}$$

with Lagrange multipliers λ, μ and η corresponding to (12), (13) and (14) respectively, we get we get the following formula:

$$\begin{aligned}\nabla_{\tilde{a}}\mathcal{L}(x^*,\lambda^*,\mu^*;\tilde{a},b,c) &= \nabla_{\tilde{a}}f(x^*;\tilde{a})+\lambda^{*T}\nabla_{\tilde{a}}h(x^*;\tilde{a}) \\ &\quad +\mu^{*T}\nabla_{\tilde{a}}g(x^*;\tilde{a}) \qquad (15)\\ &= -\eta^*, \qquad (16)\end{aligned}$$

where η contains the sensitivities for the parameters. □
Note that the formula gives the negative of the sensitivities.

There should be at least two ways to get these parameter sensitivities:

1. After x^* is obtained we get the parameter sensitivities a, together with the dual variables λ^* and μ^* using LP, or
2. after x^*, λ^* and μ^* are obtained we get the sensitivities by using analytical expressions.

In the next section we show by an example how to apply the latter approach for the pooling problem.

4 The Pooling Problem

The pooling problem is a network flow model where the raw materials are transformed to end products by blending at the sink nodes and in pools between sources and sinks The products are mixed to satisfy some given requirement at the network sinks. This problem is important for both the agricultural and petroleum industry. We can model the pooling problem through (bilinear) nonconvex quadratic programming, as an extension of the minimum cost flow problem. The following definition of the pooling problem is taken from [7] (mostly verbatim). Note that x here denotes a subset of the variables, as opposed to *all* the variables. We will use both meanings of x depending on the context.

Indices	i	sources where raw materials are applied
	j	sinks where end products are delivered
	k	quality attributes
	l	pools where entering flow streams are blended
	I	is the number of sources
	J	is the number of sinks
	L	is the number of qualities
	K	is the number of pools
Variables	p_{lk}	k^{th} quality of pool l from pooling of raw materials
	x_{il}	flow from i^{th} source into pool l
	y_{jk}	total flow from pool j to sink k
	z_{ij}	direct flow of source i to product j
Parameters	c_i	unit cost of the i^{th} raw material
	d_j	price of the j^{th} product
	A_i	availability of i^{th} raw material
	C_{ik}	k^{th} quality of raw material supplied at source i
	D_j	demand of the j^{th} product
	P_{jk}^U	upper bound on k^{th} quality of j^{th} product
	S_l	l^{th} pool capacity

Table 1: Indices, variables and parameters for the P-formulation.

In Table 1, we define the indices, variables and parameters for the so-called P-formulation of the pooling problem, which reads:

$$\text{(P)}\ \min\sum_{i=1}^{I}c_i\sum_{l=1}^{L}x_{il}-\sum_{j=1}^{J}d_j\sum_{l=1}^{L}y_{lj}-\sum_{i=1}^{I}\sum_{j=1}^{J}(d_j-c_i)z_{ij} \qquad (17)$$

$$\begin{aligned}
\text{s.t.} \sum_{l=1}^{L} x_{il} + \sum_{j=1}^{J} z_{ij} &\leq A_i, \quad i = 1, ..., I && (18)\\
\sum_{i=1}^{I} x_{il} - \sum_{j=1}^{J} y_{lj} &= 0, \quad l = 1, ..., L && (19)\\
\sum_{i=1}^{I} x_{il} &\leq S_l, \quad l = 1, ..., L && (20)\\
\sum_{l=1}^{L} y_{lj} + \sum_{i=1}^{I} z_{ij} &\leq D_j, \quad j = 1, ..., J && (21)\\
\sum_{l=1}^{L} (p_{lk} - P_{jk}^U) y_{lj} + \sum_{i=1}^{I} (C_{ik} - P_{jk}^U) z_{ij} &\leq 0, \quad j = 1, ..., J, k = 1, ..., K && (22)\\
\sum_{i=1}^{I} C_{ik} x_{il} - p_{lk} \sum_{j=1}^{J} y_{lj} &= 0, \quad l = 1, ..., L, k = 1, ..., K && (23)
\end{aligned}$$

$$x_{il} \geq 0 \ \forall(i,l), \ y_{lj} \geq 0 \ \forall(l,j), \ z_{ij} \geq 0 \ \forall(i,j), \ p_{lk} \geq 0 \ \forall(l,k).$$

The indices only run over those values for which there is a corresponding arc in the network. The constraints (22)–(23) can be explained as follows: The quality p_{lk} is defined as weighted average of the qualities of raw materials entering pool l, where the flow constitutes the weights. That is,

$$p_{lk} = \frac{\sum_i^I C_{ik} x_{il}}{\sum_i^I x_{il}}, \quad l = 1, ..., L, k = 1, ..., K.$$

The objective function gives the difference between the cost of purchasing the input raw materials and the returns from selling the end products. Other objective functions can also be used. The constraints are raw material availabilities (18), mass balance at pools (19), pool capacities (20), product demands (21), product quality requirements (22), and quality balances at pools (23), respectively.

An instance of the pooling problem is shown in Figure 1, which is an often-cited example due to Haverly [3].

4.1 The sensitivities w.r.t. the parameters of the Pooling Problem

As mentioned in the introduction, there exist unique Lagrange multipliers if the LICQ condition on the constraint gradients holds. As we will see, this condition does not necessarily hold for the pooling problem. However, it turns out that for the (single-quality) pooling problem instances we have tested, the Lagrange multipliers can be found and do have meaningful values, even though the constraint gradients are not linearly independent. This is not a violation of the theory, but a something which we would like to investigate in the future. Specifically, an interesting question is if there is a constraint qualification different from the LICQ condition, which does hold. See [5, Theorem 7.3.7] for details.

Since the pooling problem is an instance of an NLP, the sensitivities with respect to the right hand side coefficients b and c are simply λ and μ. We now give the sensitivities with respect to the parameters for the P-formulation, that is, the components of $-\eta$. For simplicity we adopt the somewhat unorthodox notation that $\mu_{(18)}$ is the multiplier which corresponds to constraint (18), and similarly for the other constraints and for the λ's. Since e.g. the expression

(18) may correspond to more than one constraint because of the index i, it follows with this notation that e.g. $\mu_{(18)}$ may also correspond to more than one multiplier.

$$
\begin{aligned}
\frac{\partial \mathcal{L}(x,y,\lambda,\mu)}{\partial c_i} &= \sum_{l=1}^{L} x_{il}^* + \sum_{j=1}^{J} z_{ij}^*, \quad i=1,...,I \\
\frac{\partial \mathcal{L}(x,y,\lambda,\mu)}{\partial d_j} &= -\sum_{l=1}^{L} y_{lj}^* - \sum_{i=1}^{I} z_{ij}^*, \quad j=1,...,J \\
\frac{\partial \mathcal{L}(x,y,\lambda,\mu)}{\partial A_i} &= -\mu_{(18)}^*, \quad i=1,...,I \\
\frac{\partial \mathcal{L}(x,y,\lambda,\mu)}{\partial S_l} &= -\mu_{(20)}^*, \quad l=1,...,L \\
\frac{\partial \mathcal{L}(x,y,\lambda,\mu)}{\partial D_j} &= -\mu_{(21)}^*, \quad j=1,...,J \\
\frac{\partial \mathcal{L}(x,y,\lambda,\mu)}{\partial C_{ik}} &= \sum_{j=1}^{J} \mu_{(22)}^* z_{ij}^* + \sum_{l=1}^{L} \lambda_{(23)}^* x_{il}^*, \quad i=1,...,I, k=1,...,K \\
\frac{\partial \mathcal{L}(x,y,\lambda,\mu)}{\partial P_{jk}^U} &= -\mu_{(22)}^* (\sum_{l=1}^{L} y_{lj}^* + \sum_{i=1}^{I} z_{ij}^*), \quad j=1,...,J, k=1,...,K
\end{aligned}
$$

If there is no direct flow from sources to sinks we have the following sensitivity for the parameter C_{ik}

$$\frac{\partial \mathcal{L}(x,y,\lambda,\mu)}{\partial C_{ik}} = \lambda_{(23)}^* \sum_{l=1}^{L} x_{il}^*, \quad i=1,...,I, k=1,...,K.$$

We see that the sensitivities of A_i, S_l and D_j are equal (apart from sign) to a multiplier, which is what we would expect since these are on the right-hand side of their respective constraints, and the sensitivity formula for the parameters a gives $-\eta$, not η. (See the proof of Theorem 3.1.)

We will use these formulas in our case studies, but first we give an outline of a fundamentally different approach to computing sensitivities.

4.2 An alternative approach to sensitivity analysis

In [2] a novel geometric approach to preprocessing and sensitivity analysis is presented. The paper looks at pooling problems where the demand at the endpoints is required to be satisfied with equality. Consequently, if the demand cannot be met then the problem instance is infeasible. By using the proposed geometric technique one can for example determine the range of the qualities at the source nodes for which the problem remains feasible. One can also determine *essential* pools and sources, that is, nodes which can be removed without making the problem infeasible. The notion of being essential is relative, so if several nodes are unessential all may not necessarily be removed at the same time, but removing one at a time will not destroy feasibility.

The paper also gives two Lagrange multipliers of the pooling problem. The first observation is that the flow weighted average cost (through pool p) equals the Lagrange multiplier, say, λ_l, associated with the flow balance equation of the pool:

$$\lambda_l = \frac{\sum_i x_{il}}{\sum_i c_i x_{il}} \quad l=1,...,L$$

The second observation is that the flow weighted average cost (of product j) equals the Lagrange multiplier for the demand constraint of sink j, say, λ_j, provided that the constraint is written with equality:

$$\lambda_j = \frac{\sum_l \beta_l y_{lj}}{\sum_l y_{lj}} \quad j = 1, ..., J$$

That these formulas for the Lagrange multipliers are given explicitly when x^* is obtained, can be advantageous since we can avoid expensive computations.

An open question concerning the pooling problem is whether more dual variables, or even all of them can be obtained by analytical calculations, when x^* is given.

5 Case studies

In this section we show how to get the sensitivities of the Haverly 1 problem instance [3]. For simplicity, we eliminate the use of asterisks (*) to denote optimal variable values. The Haverly 1 instance is visualized in Fig. 1:

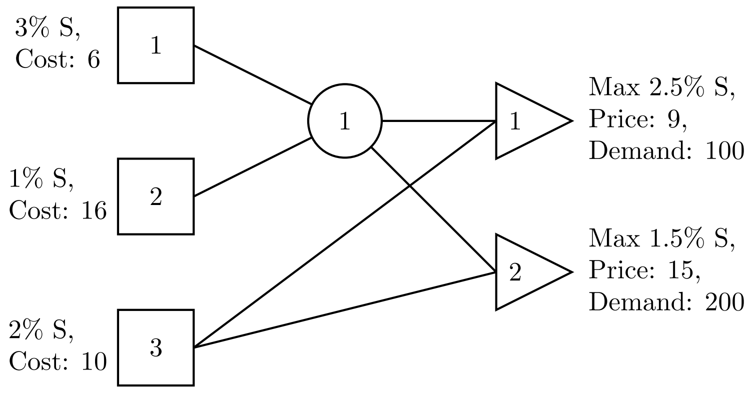

Figure 1: The Haverly 1 pooling problem.

As the figure shows, we have three sources, (the squares on the left) which provide raw materials at three different prices, and with three different levels of contaminant S (quality) (sulphur in Haverly's original work). We have one pool (the circle), and two terminals (the triangles), which have bounds on both the quality of the final product they receive, a certain price that they will pay, and an upper bound on the amount of product they want.

5.1 Min cost formulation

First we look at an objective function that represents the cost (or equivalently, the negative of the profit) of sending flow, which we want to minimize. We get the following model using the P-formulation:

$$\min c_1 x_{11} + c_2 x_{21} + c_3(z_{31} + z_{32}) - d_1(y_{11} + z_{31}) - d_1(y_{12} + z_{32}), \tag{24}$$

subject to

$$\begin{aligned} x_{11} + x_{21} - y_{11} - y_{12} &= 0, && (25)\\ C_1 x_{11} + C_2 x_{21} - p_1(y_{11} + y_{12}) &= 0, && (26)\\ p_1 y_{11} + C_3 z_{31} - P_1^U(y_{11} + z_{31}) &\leq 0, && (27)\\ p_1 y_{12} + C_3 z_{32} - P_2^U(y_{12} + z_{32}) &\leq 0, && (28)\\ y_{11} + z_{31} &\leq D_1, && (29)\\ y_{12} + z_{32} &\leq D_2. && (30) \end{aligned}$$

Here we have:

$c_1 = 6$	$d_2 = 15$	$D_1 = 100$
$c_2 = 16$	$C_1 = 3$	$D_2 = 200$
$c_3 = 10$	$C_2 = 1$	$P_1^U = 2.5$
$d_1 = 9$	$C_3 = 2$	$P_2^U = 1.5$

In addition we have nonnegativity constraints on all the flow variables, and a requirement that $p_1 \leq 100$. Note that since there is only one quality attribute, we use single indices on p, P and C. Note also that the parameters P and C appear in the left-hand side of the constraints and are as such not accessible for traditional sensitivity analysis.

The globally optimal solution to this problem is:

$$\begin{aligned}
x_{11} &= 0, \\
x_{21} &= 100, \\
z_{31} &= 0, \\
z_{32} &= 100, \\
y_{11} &= 0, \\
y_{12} &= 100, \\
p_1 &= 1,
\end{aligned}$$

which gives a cost of -400. The LICQ condition does *not* hold for this solution, but fortunately we are still able to compute the Lagrange multipliers. Using the equation numbers as a subscript once again, the multipliers are:

Multiplier	Value
$\lambda_{(25)}$	22
$\lambda_{(26)}$	6
$\mu_{(27)}$	0.13
$\mu_{(28)}$	6
$\mu_{(29)}$	0
$\mu_{(30)}$	2

There are also multipliers for the nonnegativity constraints, but these are not needed in the subsequent analysis. We now apply the sensitivity formulas from section 4.1. We then get:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial c_1} &= x_{11} = 0, \\
\frac{\partial \mathcal{L}}{\partial c_2} &= x_{21} = 100, \\
\frac{\partial \mathcal{L}}{\partial c_3} &= z_{31} + z_{32} = 100, \\
\frac{\partial \mathcal{L}}{\partial d_1} &= -(z_{31} + y_{11}) = 0, \\
\frac{\partial \mathcal{L}}{\partial d_2} &= -(z_{32} + y_{12}) = -200, \\
\frac{\partial \mathcal{L}}{\partial D_1} &= -\mu_{(29)} = 0, \\
\frac{\partial \mathcal{L}}{\partial D_2} &= -\mu_{(30)} = -2, \\
\frac{\partial \mathcal{L}}{\partial P_1^U} &= -\mu_{(27)}(y_{11} + z_{31}) = 0, \\
\frac{\partial \mathcal{L}}{\partial P_2^U} &= -\mu_{(28)}(y_{12} + z_{32}) = -1200, \\
\frac{\partial \mathcal{L}}{\partial C_1} &= \lambda_{(26)} x_{11} = 0, \\
\frac{\partial \mathcal{L}}{\partial C_2} &= \lambda_{(26)} x_{21} = 600, \\
\frac{\partial \mathcal{L}}{\partial C_3} &= \mu_{(27)} z_{31} + \mu_{(28)} z_{32} = 600.
\end{aligned}$$

To verify the results, and to see what information they provide, we solve the Haverly 1 instance again for slightly different parameter values, and compare the result with what is predicted by the derivative (8). Note that the constraint norm in (8) corresponding to these sensitivities (η) is 1 and that the above derivatives are equal to $-\eta$. (See the proof of Theorem 3.1.)

Parameter change	Prediction	Result
$\Delta P_2^U = +0.10$	$f^* = -520$	$f^* = -520$
$\Delta P_2^U = -0.10$	$f^* = -280$	$f^* = -280$
$\Delta P_2^U = +0.01$	$f^* = -412$	$f^* = -412$
$\Delta P_2^U = -0.01$	$f^* = -388$	$f^* = -388$

As the table shows, the results exactly match the prediction. The reason for this is that the objective function is linear, and that the active constraints at the perturbed solution are the same as at the original solution. However, a linear objective function is not in itself a guarantee for exact predictions, as we will see in the next section. In addition, the active set will not remain the same for all possible perturbations.

If we perturb more than one variable at a time we will need to deduce a directional derivative counterpart of (8), using the same machinery as was used to deduce (8) in [6, equation 12.33]. A reasonable question and direction of further research is to how small/large P_2^U, or indeed any parameter can be without changing the set of active constraints.

5.2 Maximum flow formulation

Instead of using the objective function (24), we can maximize the flow to the terminals instead. For technical reasons, let us formulate this as minimizing the negative of the flow, that is:

$$\min -\big(y_{11} + y_{12} + z_{31} + z_{32}\big) \tag{31}$$

The constraints for the problem are the same as for the minimum cost formulation, but in order to make the problem more interesting we add upper bounds on the flow from the sources:

$$\begin{aligned} x_{11} &\leq A_1, \\ x_{21} &\leq A_2, \\ z_{31} + z_{32} &\leq A_3, \end{aligned}$$

with $A_1 = A_2 = 80$, and $A_3 = 250$. The optimal solution is then a flow of 260. This is lower than the total demand of 300. We can then perform exactly the same analysis as we did for the minimum cost case, and perform similar predictions. If we alter P_2^U as we did for the minimum cost case, we get:

Parameter change	Prediction	Result
$\Delta P_2^U = +0.10$	$f^* = -292$	$f^* = -300$
$\Delta P_2^U = -0.10$	$f^* = -228$	$f^* = -233.3333$
$\Delta P_2^U = +0.01$	$f^* = -263.2$	$f^* = -263.26$
$\Delta P_2^U = -0.01$	$f^* = -256.8$	$f^* = -256.86$

As we can see, the predictions are in this case good for small perturbations, but not as good for larger perturbations, which is what we would usually expect from a derivative. For the case $\Delta P_2^U = +0.10$ with solution -300 the active set has changed, since there is a constraint on the total demand of 300 which has become active. Consequently the prediction is not meaningful in this case.

6 Software

In this section we briefly present some of the software packages which can be used for solving pooling problem instances. In addition, we mention whether the Lagrangian multipliers are provided when solving the original problem. Even if these codes can solve the same problem, they utilizes different algorithms/methods to find a solution.

GAMS/SNOPT is a solver for sparse nonlinear constrained optimization problems. It utilizes a sequential quadratic programming (SQP) method for solving constrained optimization problems. It finds a locally optimal solution. Lagrange multipliers are given when solving the primal problem.

GAMS/CONOPT and GAMS/CONOPT2 is a solver for nonlinear constrained optimization problems. GAMS/CONOPT2 is especially suited for very nonlinear constraints. It finds a locally optimal solution. GAMS/CONOPT2 has been designed for large and sparse models. Lagrange multipliers are given when solving for the primal problem.

GAMS/MINOS is a solver for large-scale linear and nonlinear optimization problems. It finds a locally optimal solution. Note that integer restrictions cannot be imposed directly. Lagrange multipliers are given when solving the primal problem.

MATLAB has the possibility to solve nonlinear constrained optimization problems. It finds a locally optimal solution. Lagrange multipliers are given when solving the primal problem.

GAMS/BARON is a solver for both nonlinear and mixed-integer nonlinear problems. It finds a globally optimal solution. Whether the Lagrange multipliers are available when using GAMS/BARON is of the time of writing not known.

7 Extension to Production Availability

Consider the case where one wants to investigate the *production availability* (PA) of a certain network, over its life span. One way to do this is to perform a Monte-Carlo simulation over several life cycles, where each life cycles contains a number of randomly generated events. Each event can be a change in demand, a change in supply, the shutdown of a pipeline, the repair of a damaged pipeline, etc. Each event puts the network in a certain state, and for each state we can solve a max-flow problem. For each state we can also compute the ratio:

$$\frac{\text{Actual flow}}{\text{Desired flow}}, \tag{32}$$

where the “flow” typically refers to the total flow entering one or more sinks. The PA is then a time-weighted average of this fraction:

$$PA = \frac{1}{t_N - t_0} \sum_{i=1}^{N} \Big(\frac{\text{Actual flow}}{\text{Desired flow}} \cdot (t_i - t_{i-1}) \Big), \tag{33}$$

where the network is considered from time t_0 to time t_N, and the events that change the state of the network occur instantaneously at the points $t_1, t_2, \ldots, t_N$ in time. The output of the Monte-Carlo simulation is the average PA over many life cycles.

Since the PA or even the output of the entire simulation is just a long sum mainly consisting of max-flow solutions we can, if we are able to store the Lagrange multipliers and constraint norms of each flow solution, compute its rate of change to a change in one of the constraints. This technique should be used with care, however, since as before the predicted change from the sensitivity analysis is valid only if the set of active constraints remains the same for each max-flow instance.

8 Extension to Gas Flow

So far we have considered a flow model where flow capacities are fixed exogenous data. When modelling the flow of gas the picture is more complicated, since gas is compressible and flows from high pressure to low pressure. One way to model gas flow is to investigate the underlying differential equations. In a steady state analysis, it is common practice to use a simplification of these equations, namely the Weymouth equation:

$$v^2 = W(p^2 - q^2), \tag{34}$$

where v is the flow through the pipeline, W is a constant which represents pipe characteristics such as its dimension and internal friction, p is the inlet (starting point for the flow in the pipe) pressure and q is the outlet (ending point for flow in the pipe) pressure. This is a non-convex constraint, but if we are considering a max-flow problem, it can be replaced by the convex constraint

$$v^2 \leq W(P - Q), \tag{35}$$

where P and Q are simply the squares of p and q and replace these as the variables. Since we are maximizing flow the constraint will be satisfied with equality at the optimal solution. It can either be used as written, or it can be replaced with a linear approximation by using multiple linear constraints, effectively replacing the curve (34) with many line segments. This only introduces a small error in the final solution.

A second complicating factor in the case of gas flow is the fact that when two pipelines meet at a junction, their outlet pressures must be equal. This can be modelled by introducing binary (0/1) variables, which enforce an equality constraint on the outlet pressures if and only if there is flow through both pipes.

The Lagrange multiplier framework is not suitable for binary variables, at least not in the form presented in this report. However, as long as the parameters of the problem instance are not changed to such a degree that the (0/1) variables have a different value in the new optimal solution, the framework can be used. It should be noted that if the constraint (35) is replaced by a linearization, then the framework will not be suitable if the parameters are changed such that the perturbed solution lies on a line segment different than the original one.

9 Conclusions

This note will hopefully give some ideas on how to efficiently implement the formulas for the sensitivities for the pooling problem. We have applied both traditional sensitivity analysis as well as recent ideas in the field. The machinery of Lagrange multipliers seems to be applicable to the problem, for both compressible and non-compressible flow, but some theoretical issues, notably related to the linear (in)dependence of the constraint gradients remain and should be investigated further. The case of multiple qualities should also be tested.

We have only considered the P-formulation in this note but it should be straightforward to get the sensitivity formulas for the other formulations such as the Q and PQ-formulation [7].

Acknowledgement

This work was financially supported by StatoilHydro, Gassco and the Research Council of Norway under contract 175967/S30. We would like to thank Bjørgulf H. Eidesen and Lars Hagesæther of StatoilHydro, as well as Sveinung Erland of Gassco, for interesting conversations which motivated this work.

References

[1] E. Castillo, A. J. Conejo, R. Minguez, and C. Castillo. A closed formula for local sensitivity analysis in mathematical programming. *Engineering Optimization*, 38(1):93–112, 2006.

[2] H. J. Greenberg. Analyzing the pooling problem. *ORSA J. Comput.*, 7(2):205–217, 1995.

[3] C. A. Haverly. Studies of the behavior of recursion for the pooling problem. *SIGMAP Bull.*, 25:19–28, 1978.

[4] D. Luenberger. *Linear and Nonlinear Programming, 2nd Ed.* Addison-Wesley, 2003.

[5] Olvi L. Mangasarian. *Nonlinear Programming.* McGraw–Hill Book Company, 1969.

[6] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 1999.

[7] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[8] R. Vanderbei. *Linear Programming: Foundations and Extensions.* Addison-Wesley, 1996.