Convex approximations in stochastic programming by semidefinite programming

Tamás Terlaky
Lehigh University

Imre Pólik
Lehigh University

István Deák
Corvinus University

András Prékopa
Eötvös University

Report: 10T-002

Convex approximations in stochastic programming by semidefinite programming*

István Deák[†] Imre Pólik[‡] András Prékopa[§] Tamás Terlaky[¶]

April 8, 2010

Abstract

The following question arises in stochastic programming: how can one approximate a noisy convex function with a convex quadratic function that is optimal in some sense. Using several approaches for constructing convex approximations we present some optimization models yielding convex quadratic regressions that are optimal approximations in L_1, L_∞ and L_2 norm. Extensive numerical experiments to investigate the behaviour of the proposed methods are also performed.

1 Introduction

Let $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ be a convex function whose values can be evaluated only with some noise. Assume that for given points $\mathbf{x}_i, i = 1, \ldots, N$, instead of the function value $f_i = f(\mathbf{x}_i), i = 1, \ldots, N$, we can compute unbiased estimates, i.e., only the values $\varphi_i = f_i + \varepsilon_i$ are available, where the random variables ε_i are completely independent with $E(\varepsilon_i) = 0$ and $D^2(\varepsilon_i) = \sigma^2$, $i = 1, \ldots, N$. This relationship between the function value and its noisy approximation is denoted by $f_i \sim \varphi_i$. The noise in our problems arises because the function values are computed by simulation.

*The research of the first and third authors was supported by the National Scientific Research Fund (Hungary), grant number T047340. The research of the second and fourth authors was supported by start-up grants from Lehigh University.

[†]Computer Science Department, Corvinus University of Budapest, H-1093 Budapest, IX. Fővám tér 8., Hungary, (Corresponding author)

[‡]Department of Industrial & Systems Engineering, Lehigh University, Bethlehem, PA, USA

[§]Department of Operations Research, Eötvös University, Budapest, Hungary

[¶]Department of Industrial & Systems Engineering, Lehigh University, Bethlehem, PA, USA

The main question we consider in this paper is the following: given the values $\mathbf{x}_i, \varphi_i, i = 1, \ldots, N$, how can one determine an optimal quadratic approximation $d(\cdot)$ to the function $f(\cdot)$, where

$$d(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

and A is an $n \times n$ symmetric positive definite matrix. In the sequel this problem will be referred to as the quadratic regression problem.

This problem emerged while one of the authors constructed some numerical solution procedures, quite successful for solving two-stage problems [3, 5, 4], probabilistic constrained problems, and mixed stochastic programming problems [2]. Prékopa pointed out that the approximation used in these numerical procedures can be made convex by semidefinite optimization. Convex approximation also occurs, e.g., in [12], where the authors approximate convex density functions by convex, piecewise linear functions. Further, the problem of quadratic regression arises in several statistical problems, see [13] for some examples. It is also featured as one the applications of semidefinite optimization in [1], but no numerical experiments are given.)

The paper is structured as follows. In Section 2 we give a short overview of the method of successive regression approximations as applied to the two-stage problem of stochastic programming together with a naive approach, used in previous solution algorithms. In Section 2.1 we present three approaches to solve the quadratic regression problem, where the distance of the original function $f(\mathbf{x})$ and the approximating function $d(\mathbf{x})$ is optimal in L_1, L_∞, and L_2-norm, respectively. Extensive numerical experiments are presented in Section 3.

Throughout this paper we assume that the set $S_N = \{\mathbf{x}_i, \varphi_i\}_{i=1}^N$ of points and noisy function values is available.

2 Stochastic programming and successive regression approximations

The successive regression approximation (SRA) algorithm for solving stochastic programming problems is illustrated here by considering the application of SRA to one of the basic models of stochastic programming, the two-stage problem. Quadratic regressions may be very useful in this case. The first stage problem of the two-stage stochastic programming model has the following form:

$$\begin{aligned} \min\ \mathbf{c}^T\mathbf{x} + &E[Q(\mathbf{x}, \xi)] \\ A\mathbf{x} &\le \mathbf{b}, \\ \mathbf{x} &\ge 0. \end{aligned} \tag{St1}$$

The recourse function $\mathcal{Q}(\mathbf{x}, \xi)$ is defined as the optimal value of a linear programming problem, the so called second stage problem:

$$\begin{aligned} \mathcal{Q}(\mathbf{x}, \xi) = \min_{\mathbf{y}} \mathbf{q}^T\mathbf{y} \\ T\mathbf{x} + W\mathbf{y} = \xi, \\ \mathbf{y} \geq 0. \end{aligned} \tag{St2}$$

Assume that the distribution function of the random vector ξ is $F(\mathbf{x})$ and Ξ is its support. Then the expected recourse function can be given as

$$Q(\mathbf{x}) = E(\mathcal{Q}(\mathbf{x}, \xi)) = \int_{\Xi} \mathcal{Q}(\mathbf{x}, \xi) dF(\xi). \tag{2.1}$$

The function f described in the previous section is replaced now by $Q(\mathbf{x})$. We assume that only the right hand side vector ξ of the second stage problem is random, while all other quantities (A, b, q, T, W) are deterministic. Furthermore we assume that all linear programming problems involved can be solved with finite objective function values (fixed and complete recourse problem). For description of results concerning this stochastic programming model see [6, 8, 9, 10].

One of the main numerical difficulties in solving a two-stage problem is the evaluation of the expected recourse function in (2.1). In some nonlinear optimization algorithms we compute an approximate value of it by a Monte Carlo method [3, 5].

The SRA technique is applied the following way:

1. the computationally hard expected recourse function $Q(\mathbf{x})$ is replaced with a convex quadratic approximation $d(\mathbf{x})$, which is optimal in some sense for the given set of points and function values (the quadratic regression problem),

2. the resulting approximate problem is optimized, and

3. an optimal solution of the approximate problem and its "noisy" function value are attached to the set of points and function values over which the convex approximation is determined, and the procedure is repeated from the beginning.

So assuming that we already have N points and noisy function values in the set S_N we look for an approximating function of the form

$$d_N(\mathbf{x}) = \mathbf{x}^T A_N \mathbf{x} + \mathbf{b}_N^T \mathbf{x} + c_N, \tag{2.2}$$

where A_N is symmetric and positive semidefinite. If we have determined the convex approximation $d_N(\mathbf{x})$, then we are able to formulate Algorithm 2.1, a tentative successive regression approximation algorithm for solving the two-stage problem.

Algorithm 2.1 The *SRA* algorithm for two-stage problems

```
Input: A set of N points and function values $S_N = \{\mathbf{x}_i, \varphi_i\}_{i=1}^N$.
  Let $k = N$
  repeat
    Compute (by semidefinite optimization, or otherwise) the constants $A_k, \mathbf{b}_k, c_k$
    of the function $d_k(\mathbf{x})$ from $S_k$.
    Replace the original problem (St1) by an approximate one:
        $\min_{\mathbf{x}} \mathbf{c}^{\mathbf{T}}\mathbf{x} + d_k(\mathbf{x}),$
        $A\mathbf{x} \le \mathbf{b},$                                    (2.3)
        $\mathbf{x} \ge 0,$
    and denote an optimal solution of this approximate problem by $\mathbf{x}_{k+1}$.
    Compute the noisy function value $\varphi_{k+1} \sim Q(\mathbf{x}_{k+1})$, and
    add the new point $\mathbf{x}_{k+1}$ and the function value $\varphi_{k+1}$ to the set $S_k$:
        $S_{k+1} = S_k \cup \{\mathbf{x}_{k+1}, \varphi_{k+1}\}.$
    Set $k = k + 1$
  until $\mathbf{x}_k$ is "good enough"
Output: The constants $A_k, \mathbf{b}_k, c_k$ of the quadratic approximation function $d_k(\mathbf{x})$.
```

From this algorithm one can see that there are two parts of the quadratic regression problem. First from the set S_N the values of $A_N, \mathbf{b}_N, c_N$ of the initial convex approximation are to be determined, then in each iteration we have to recompute the same values over the extended set S_{N+1}, or use an updating procedure of the values. In this paper we only consider the first problem.

In the first attempt to solve the quadratic regression problem we simply solved the unconstrained optimization problem

$$\min_{A_N, \mathbf{b}_N, c_N} \sum_{i=1}^{N} \left[\varphi_i - d_N(\mathbf{x}_i)\right]^2 \tag{2.4}$$

for the unknowns $A_N, \mathbf{b}_N, c_N$. The first order necessary conditions of optimality yielded a large system of linear equations that could be solved. This algorithm of solving the quadratic regression problem will be called the least-squares (LS) approach. This method had some serious drawbacks:

- The system may be over or underdetermined. To remedy this a least-squares approach was used.

- Matrix A_k is not guaranteed to be positive semidefinite, in particular, if the points lie on an affine subspace (which is the case in our application, since we work with feasible solutions of the first stage problem), or if they are ill-conditioned.
- Even if the points are in general position and the function to approximate is convex, the approximation computed from system (2.4) may not be convex, see [1, Example 9.1] for an example.
- The presence of noise in the function values is not accounted for.

This experience leads us to force the convexity of the approximating quadratic function explicitly.

2.1 Semidefinite optimization

The quadratic regression problem can be formulated as the following optimization problem:

$$\min_{A \succeq 0, \mathbf{b}, c} \left\| \left[\varphi_i - \left(\mathbf{x}_i^T A \mathbf{x}_i + \mathbf{b}^T \mathbf{x}_i + c\right)\right]_{i=1}^N \right\|, \tag{2.5}$$

where $\|\cdot\|$ is some norm and for convenience the subscript N is dropped from coefficients of the approximating function. We consider three different norms. In all of these problems the number of unknowns in A, $\mathbf{b}$ and c altogether is $M = n(n+1)/2 + n + 1$, so for $n = 100$ this becomes $M = 5151$.

As in [1], there are three tractable choices for the norm:

2.1.1 L_1 minimum norm optimal convex approximation

Assuming the absolute value norm, the following problem has to be solved:

$$\begin{aligned} &\min \sum_{i=1}^N \left(\vartheta_i^+ + \vartheta_i^-\right) \\ \varphi_i - \left(\frac{1}{2}\mathbf{x}_i^T A \mathbf{x}_i + \mathbf{b}^T \mathbf{x}_i + c\right) &= \vartheta_i^+ - \vartheta_i^-, \ i = 1, \ldots, N, \\ &\vartheta_i^+, \vartheta_i^- \geq 0, \ i = 1, \ldots, N, \\ &A \succeq 0. \end{aligned} \tag{2.6}$$

This is a pure semidefinite optimization problem, with $2N$ nonnegative variables and N linear equalities.

2.1.2 L_∞ minimum norm optimal convex approximation

If the norm in (2.5) is the maximum distance norm, then the optimal approximation is a solution of the following problem:

$$\begin{aligned} &\min \vartheta \\ -\vartheta \le \varphi_i - \left(\frac{1}{2}\mathbf{x}_i^T A \mathbf{x}_i + \mathbf{b}^T \mathbf{x}_i + c\right) &\le \vartheta,\ i = 1,\dots,N, \\ \vartheta &\ge 0, \\ A &\succeq 0. \end{aligned} \tag{2.7}$$

This problem is again pure semidefinite optimization, with $2N$ linear inequalities and N nonnegative variables.

2.1.3 L_2 norm optimal convex approximation

Finally the least squares approximation can be derived as the solution of the problem:

$$\begin{aligned} &\min \sum_{i=1}^{N} \vartheta_i^2 \\ \varphi_i - \left(\frac{1}{2}\mathbf{x}_i^T A \mathbf{x}_i + \mathbf{b}^T \mathbf{x}_i + c\right) &= \vartheta_i,\ i = 1,\dots,N, \\ A &\succeq 0. \end{aligned} \tag{2.8}$$

An equivalent formulation of the previous problem can be given as:

$$\begin{aligned} &\min \vartheta \\ \sqrt{\sum_{i=1}^{N} \vartheta_i^2} &\le \vartheta \\ \varphi_i - \left(\frac{1}{2}\mathbf{x}_i^T A \mathbf{x}_i + \mathbf{b}^T \mathbf{x}_i + c\right) &= \vartheta_i,\ i = 1,\dots,N, \\ A &\succeq 0, \end{aligned} \tag{2.9}$$

which is a mixed second-order–semidefinite optimization problem with only one second-order cone and N linear equalities.

3 Numerical experiments

Some numerical problems have been solved to show the practical usefulness of these approaches and to test their behaviour under different conditions. All of the tests were run on a dual core AMD Turion laptop with 4GB ram, under Windows Vista (64 bit) using Matlab R2007b (also 64 bit).

3.1 Modelling

The numerical examples were solved with SeDuMi [11], version 1.2, available from `http://sedumi.ie.lehigh.edu`. To facilitate modelling we used the modelling language Yalmip [7] to model the problems with the following lines code:

```
Avar=sdpvar(n,n);
bvar=sdpvar(n,1);
cvar=sdpvar(1);

F=set(Avar>=0);
OBJ=norm(phi_x-(dot(x,Avar*x)'+x'*bvar+cvar),p);
options=sdpsettings('solver','sedumi','verbose',0);

solvesdp(F,OBJ,options);
```

Recall that n is the dimension of the space and N is the number of points. The value of p was 1, 2 or ∞. Yalmip then transforms the problem into the standard input format used by SeDuMi. This transformation takes relatively little time compared to the solution of the problem afterwards. We have also tested the simple least squares approach with iterative refinement for better accuracy.

3.2 Test sets

The following test cases were generated:

Cube Points are generated in general position. Uniform distribution over the unit-cube was used.

Ball Points are generated in general position. Uniform distribution over the unit-ball was used.

Ball-noisy Like the previous example, but a small percentage of the components of the vectors are zeroed out.

Degenerate The points span a lower dimensional subspace. The dimension of the subspace is roughly half of the dimension of the space.

Ill-conditioned The points span the full space, but they are ill-conditioned. These are generated from the degenerate test cases by a small perturbation of the points.

We started the problem exposition in this paper with applying the SRA method to the two-stage stochastic programming problem. This iterative technique generally produces a degenerate set of points, since the points produced during the iteration procedure are feasible solutions of the first stage, thus if among the inequalities $A\mathbf{x} \leq \mathbf{b}$ we have some equations, or near to the optimum some conditions are active, then degeneracy is certain. To counteract this feature during the SRA procedure we produced points by small perturbations of the feasible points of the first stage, too. This is why the degenerate and ill-conditioned sets of points are important in our application. Degenerate sets of points appear in using SRA for probabilistic and mixed stochastic programming problems as well, because of the same reasons described above.

The function to approximate is a convex quadratic function in each case. We used three different functions:

General A randomly generated positive semidefinite quadratic function. It may be close to being indefinite.

Definite A randomly generated convex quadratic function plus the norm function, thus this function is always strictly convex.

Deficient A convex, but not strictly convex quadratic function, thus matrix A is rank-deficient. An optional parameter controls the rank of A.

In all of the cases we have also varied the number of data points and the amount of noise added to the function values.

3.3 Results

3.3.1 General comments

As expected, the simple linear approximation fails miserably if there are too few data points. On the other hand, if it works, it is the fastest method.

There is an important observation here. If the linear approximation returns a non-convex quadratic function, then in the norm-optimization problems the semidefiniteness constraint will be active (otherwise we could have ignored it), meaning the function we get will be convex, but not strictly convex, i. e., matrix A will be positive semidefinite, but not positive definite. Now, due to numerical problems and rounding inaccuracies, the result we get from an algorithm may be slightly indefinite, the smallest eigenvalue can be around -10^{-9}. To counter this we need to use a constraint such as $A \succeq \alpha I$ with some $\alpha > 0$ or add a small regularization term βI to A to force A to be positive definite.

3.3.2 Solution quality

It is particularly tricky to measure the quality of the solutions. Obviously, if A is not positive semidefinite, then the approximation failed, but as we explained above, this may be a normal occurrence. Comparing against the original quadratic function only makes sense if the approximation is unique, in which case all the methods would return the same result. The presence of noise in the sample values introduces another challenge. Also, there is a more philosophical observation here. In practical applications, it only makes sense to use a convex quadratic approximation if the function to approximate is (nearly) convex.

The advantage of norm-minimization approaches is that their accuracy can be controlled. To compensate for this, iterative refinement (3 iterations) was used to improve the accuracy of the least-squares solution.

For these reasons we focus on the computational behaviour of the algorithms instead of the solution quality. As we will see, this is a valid approach in our application.

3.3.3 Solution time

Tables 1 and 2 contain some typical running times for test problems of different sizes.

n	N	LS	1-norm	2-norm	inf-norm
2	6	0.001	0.335	0.330	0.307
4	15	0.001	0.346	0.342	0.314
8	45	0.005	0.396	0.389	0.375
16	153	0.086	1.079	0.684	0.983
32	561	4.534	36.031	12.401	28.355
64	2145	253.697	2019.720	680.395	1681.470
70	2556	448.088	m	1166.570	m
75	2926	652.571	m	1591.130	m
80	3321	938.302	m	m	m
85	3741	1902.000	m	m	m
90	4186	2623.000	m	m	m
95	4656	3879.000	m	m	m

Table 1: Running times (in seconds) on problems of different sizes, when n is increasing and N is set such that we have just enough points. m denotes "Out of memory"

As expected, the least-squares approach is the fastest – if it works. Also, its performance is very predictable, as it reduces to solving a linear system. Exploiting

n	N	LS	1-norm	2-norm	inf-norm
32	561	4.535	36.030	12.401	28.355
32	1000	5.344	61.986	15.912	41.040
32	1500	7.705	109.679	25.110	56.383
32	2000	8.770	192.197	36.493	75.811
32	2500	8.942	290.672	42.012	97.306
32	3000	10.899	439.208	52.648	114.929

Table 2: Running times (in seconds) on problems of different sizes, when n is fixed and N is increasing.

sparsity in the data is quite immediate. Overall, all four approaches are much less sensitive to the increase of the number of points, than the increase of the dimension. Among the norm-optimization approaches the 2-norm minimization behaves better than the other two. This is probably due to the fact that it results in the smallest model, as it does not have linear inequalities. This explains both the low memory requirement and the fast execution.

3.3.4 Problem sizes

It is obvious from the solution times that the approaches differ in what size of problems they can handle. Moreover, the memory requirements further limit the size of tractable problems. We have discussed briefly that out of the three norm-minimization approaches, the 2-norm results in the smallest model and uses the less memory. This is of course much more than the memory requirement of the least-squares approach, which can solve problems up to $n = 250$ on a modern PC.

3.3.5 Datasets

We tested all four methods on all five datasets. We choose $n = 10$ and $N = 66$. The quadratic functions in this experiment were strictly convex. As all the methods yielded a very good representation error, we measured solution quality in terms of the convexity of the approximation. As we discussed earlier, interior-point methods sometimes return slightly indefinite results, which can be countered easily by adding a small diagonal matrix to A. We accepted a solution if the smallest eigenvalue of A was not less than -10^{-6}. We have also tried other thresholds, ranging from -10^{-8} to -10^{-2}, but it had only a minor effect on the overall percentages, since the function to approximate was strictly convex. As a comparison, the success rate of the least-squares approach on the fourth dataset dropped to 17% if we required the smallest eigenvalue to be nonnegative. Table 3 summarizes the results.

Dataset	LS	1-norm	2-norm	inf-norm
Cube	100%	100%	100%	100%
Ball	100%	100%	100%	100%
Ball-noisy	100%	100%	100%	100%
Degenerate	41%	100%	100%	100%
Ill-conditioned	<0.01%	100%	100%	100%

Table 3: Success rates for problems from different datasets. The size of the problems is fixed, $n = 10$, $N = 66$. The fourth datasets contains points in a lower dimensional subspace, while the fifth one is a perturbed version of that. The table is based on 1000 runs.

All four methods are quite reliable if the data points span the whole space and are well-poised. On the other hand, the least-squares method struggles to yield a convex approximator if the points are in a lower dimensional space. This is not surprising, as in that case the best "quadratic" approximation is a linear function, apart from some rounding error, which may make the approximation indefinite. This might be circumvented by finding the subspace spanned by the points and applying the least-squares procedure in that subspace. The ill-conditioned sets, however, pose a bigger challenge, and the least-squares approximation is rarely convex. This is a direct consequence of the geometry of the problem.

3.3.6 Different functions

So far all the tests have been run on generic quadratic functions. To better understand the strengths and weaknesses of the four approaches we tested them with strictly positive definite and rank-deficient quadratic functions. Once again, we checked if the smallest eigenvalue of A is greater than -10^{-6}. The results are summarized in Table 4.

Function	LS	1-norm	2-norm	inf-norm
General	100%	100%	100%	100%
Definite	100%	100%	100%	100%
Deficient	97%	100%	100%	100%

Table 4: Success rates on problems with different functions over points distributed on the unit ball. The size of the problems is fixed, $n = 10$, $N = 66$. The table is based on 1000 runs.

The points were distributed uniformly over the unit ball. As expected, the rank-deficient functions posed some challenge to the least-squares method, but overall, it performed quite well. We have to add that the least-squares method almost never returned a convex approximation, some very small eigenvalues were always present, and in 3% of the cases, some of these eigenvalues were more negative than -10^{-6}. The generic and the strictly convex quadratic functions were handled without any difficulty by all the methods.

3.3.7 Noise-tolerance

To test the noise-tolerance of the methods we chose the combination of a strictly convex quadratic function over a full-dimensional, well-poised dataset. We added a normally distributed relative noise to the function values. In this experiment we made sure that the number of data points is greater than what would be needed to uniquely define a quadratic function, otherwise a perfect fit could be found for the perturbed problem. The results are summarized in Table 5.

σ	LS	1-norm	2-norm	inf-norm
0.000	100%	100%	100%	100%
0.001	99.34%	100%	100%	100%
0.002	97.04%	100%	100%	100%
0.004	92.51%	100%	100%	100%
0.008	91.14%	100%	100%	100%
0.016	81.64%	100%	100%	100%
0.032	76.74%	100%	100%	100%
0.064	70.34%	100%	100%	100%
0.128	65.50%	100%	100%	100%
0.256	52.48%	100%	100%	100%

Table 5: Success rates for problems with added noise. The size of the problems is fixed, $n = 3$, $N = 15$. The last row represents a 25% noise/signal ratio. The table is based on 1000 runs.

As the noise increases, the best quadratic approximator to the perturbed problem is becoming nonconvex, thus the least-squares method will struggle to find a convex approximation. The norm-minimization methods do not exhibit this problem.

As a second test, we looked at the noise-filtering effect of the methods. In other words, as the noise is increasing, how far the approximation will be from the true coefficients of the quadratic function. We measured the distance in 2-norm, and consequently used only the 2-norm minimization approach. The results are in Table 6.

Only the instances where the minimum eigenvalue of A was greater than -10^{-6} were included in the calculations. We can see that the 2-norm minimization has

σ	LS	2-norm
0.000	6.2443×10^{-14}	6.4069×10^{-13}
0.001	1.0687×10^{-2}	1.0626×10^{-2}
0.002	2.1039×10^{-2}	2.1193×10^{-2}
0.004	4.0060×10^{-2}	4.0791×10^{-2}
0.008	8.2280×10^{-2}	8.1017×10^{-2}
0.016	1.5711×10^{-1}	1.5625×10^{-1}
0.032	3.3460×10^{-1}	3.3080×10^{-1}
0.064	6.2072×10^{-1}	5.9699×10^{-1}
0.128	1.2723×10^{0}	1.2312×10^{0}
0.256	2.3891×10^{0}	2.2555×10^{0}

Table 6: The noise-filtering effect of the methods. The difference between the true coefficients and the approximations to the noisy problems, measured in 2-norm. The table is based on 1000 runs.

a slightly better noise-filtering effect, even though we did not use any regularization. The advantage becomes more pronounced as the noise increases. We can also conclude (see the first row) that the least-squares approach achieves slightly better accuracy in recovering the quadratic function for the unperturbed problem.

4 Conclusions

As we can see, the least-squares approximation works well only if the data points are well-poised, the quadratic function is strictly convex and the noise is fairly small. Problems arising from our application rarely fall into this category, as they are numerically degenerate and may have too much noise in them. In these cases the convexity of the approximation has to be enforced explicitly using one of the three methods discussed in this paper.

Our numerical experiments indicate that the most efficient method in terms of solution time and memory usage is the L_2-norm approximation, which leads to a mixed second-order–semidefinite optimization problem. Our models are solvable on a modern computer for up to $N = 3000$ points in an $n = 50$ dimensional space, which are practically useful problem sizes.

References

[1] E. de Klerk. *Aspects of Semidefinite Programming: Interior Point Algorithms and Selected Applications*, volume 65 of *Applied Optimization*, chapter 9, pages

149–155. Kluwer Academic Publishers, 2002.

[2] I. Deák. Solving stochastic programming problems by successive regression approximations – numerical results. In K. Marti, Y. Ermoliev, and G. Pflug, editors, *Dynamic Stochastic Optimization*, number 532 in Lecture Notes in Economics and Mathematical Systems, pages 209–224. Springer, 2003.

[3] I. Deák. Two-stage stochastic problems with correlated normal variables: computational experiences. *Annals of Operations Research*, 142(1):79–97, 2006.

[4] I. Deák. Convergence of successive regression approximations for solving equations with noisy function values. In B. Topping, editor, *Proceedings of the CST2010 The Tenth International Conference on Engineering Computational Technology, Valencia, Spain.* 2010. Submitted.

[5] I. Deák. Testing successive regression approximations by large-scale two-stage problems. *Annals of Operations Research*, 2010. To appear.

[6] Cs. I. Fábián and Z. Szőke. Solving two-stage stochastic programming problems with level decomposition. *Computational Management Science*, 4(1):313–353, 2007.

[7] J. Löfberg. YALMIP: a toolbox for modeling and optimization in MATLAB. In *Proceedings of the 2004 IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, 2004.

[8] J. Mayer. *Stochastic Linear Programming Algorithms.* Gordon and Breach Science Publishers, 1998.

[9] A. Prékopa. *Stochastic Programming.* Number 324 in Mathematics and its Applications. Kluwer Academic Publishers, 1995.

[10] A. Ruszczyński and A. Shapiro, editors. *Stochastic Programming*, volume 10 of *Handbooks in Operation Research and Management Science.* Elsevier, 2003.

[11] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[12] T. Terlaky and J-Ph. Vial. Computing maximum likelihood estimators of convex density functions. *SIAM J. on Scientific Computing*, 19:675–694, 1998.

[13] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.