J Glob Optim (2006) 36:461–469
DOI 10.1007/s10898-006-9021-4

ORIGINAL ARTICLE

Approximating data in $\Re^n$ by a quadratic underestimator with specified Hessian minimum and maximum eigenvalues

J. B. Rosen · John Glick

Received: 8 March 2006 / Accepted: 8 March 2006 / Published online: 15 June 2006

Abstract The problem of approximating m data points (x_i, y_i) in $\Re^{n+1}$, with a quadratic function $q(x, p)$ with s parameters, $m \geq s$, is considered. The parameter vector $p \in \Re^s$ is to be determined so as to satisfy three conditions: (1) $q(x, p)$ must underestimate all m data points, i.e. $q(x_i, p) \leq y_i, i = 1, \ldots, m$. (2) The error of the approximation is to be minimized in the $L1$ norm. (3) The eigenvalues of H are to satisfy specified lower and upper bounds, where H is the Hessian of $q(x, p)$ with respect to x. This is called the Quadratic Underestimator with Bounds on Eigenvalues (QUBE) problem. An algorithm for its solution (QUBE algorithm) is given and justified, and computational results presented. The QUBE algorithm has application to finding the global minimum of a basin (or funnel) shaped function with a large number of local minima. Such problems arise in computational biology where it is desired to find the global minimum of an energy surface, in order to predict native protein-ligand docking geometry (drug design) or protein structure. Computational results for a simulated docking energy surface, with $n = 15$, are presented. It is shown that specifying a small condition number for H improves the ability of the underestimator to correctly predict the global minimum point.

Keywords Underestimating approximation · eigenvalue bounds · global minimization · protein docking

1 Introduction

There are a number of applications in computational biology where it is desired to approximate data in $\Re^n$ by a quadratic function with specified properties. Typically, we want a quadratic approximation which is convex, and which underestimates all the data points

J. B. Rosen (✉)
Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093, USA
e-mail: jbrosen@cs.ucsd.edu

J. Glick
Department of Mathematics and Computer Science, University of San Diego, San Diego, CA 92110, USA
e-mail: glick@sandiego.edu

(Dill et al. 1997; Rosen and Marcia 2004). This convex underestimator is used as part of an iterative algorithm to determine the global minimum of a basin (or funnel) shaped energy function with a large number of local minima. Such problems arise in computational biology where it is desired to find the global minimum of an energy surface, in order to predict native protein-ligand docking geometry (drug design) or protein structure (Mitchell et al. 1999; Powers et al. 2002; Wei et al. 2004; Marcia et al. 2005).

The quadratic underestimator in $\Re^n$ is represented by:

$$q(x, p) = \alpha + c^T x + \tfrac{1}{2} x^T H x, \quad x \in \Re^n, \tag{1}$$

where $p \in \Re^s$ represents the $s = \frac{1}{2}(n+1)(n+2)$ parameters $\alpha, c \in \Re^n$, and the elements of the symmetric matrix H. The parameters p are determined so as to minimize the $L1$ approximation error in the fit of $q(x, p)$ to m data points $(x_i, y_i), i = 1, \ldots, m$, where $m \geq s$. In typical applications, the data points (x_i, y_i) are local minima of a function $f(x)$ which represents a basin-shaped energy surface in $\Re^n$. The function $f(x)$ will typically have a very large number of local minima and will be computationally expensive to evaluate.

Here, we consider a somewhat more general quadratic underestimating function, which includes the convex quadratic as a special case. Two bounds are specified (μ_1 and μ_n), representing the minimum and maximum eigenvalues of the symmetric Hessian matrix H. The algorithm, which we summarize next, guarantees that the elements of H are determined so that the eigenvalues of H satisfy these bounds, and the $L1$ approximation error is minimized. We call this the Quadratic Underestimator with Bounds on Eigenvalues (QUBE) problem. This differs from the problem considered by Dill et al. (1997), which requires the Hessian matrix to be diagonal with positive entries, and by Rosen and Marcia (2004), which does not allow for upper and lower bounds on the eigenvalues of the Hessian matrix.

In the QUBE problem, if the lower bound is non-negative, we have the convex case. However, for some applications an indefinite quadratic, with specified minimum and maximum eigenvalues, may be useful.

The algorithm will now be summarized. Determination of the parameters p, so as to minimize the $L1$ error, is formulated as a sequence of major iterations, each of which is a linear program. After each major iteration, the eigenvalues of H are computed. If the eigenvalues of H satisfy the specified bounds, the desired approximation has been obtained, and the algorithm terminates. Otherwise, at least one new linear inequality is added and the next major iteration carried out. Convergence of this algorithm is shown, and computational tests presented showing that typically only a small number of major iterations are required.

Each new linear inequality constraint is in fact a cutting plane which excludes a finite region of the parameter space where the eigenvalues of H violate the specified bounds. More specifically, consider only the minimum eigenvalue of H. Let μ_1 be the specified lower bound, and $\lambda_1(H)$ the minimum eigenvalue of H, with $u_1(H)$ the corresponding eigenvector, where $\|u_1\| = 1$. Since $Hu_1 = \lambda_1 u_1$, we have $u_1^T H u_1 = \lambda_1$. Suppose $\lambda_1 < \mu_1$. Then the constraint $u_1^T H u_1 \geq \mu_1$ will cut off a finite region of parameter space where $\lambda_1(H) < \mu_1$. Note that the constraint $u_1^T H u_1 \geq \mu_1$, is linear in the elements of H. The algorithm is described in detail in Sect. 3.

For the special case where $\mu_1 = 0$ and $\mu_n = \infty$, the problem can be formulated as a positive semi-definite program (Alizadeh et al. 1997). However, to our knowledge, no method for the more general case considered here, has been presented.

2 Determination of quadratic underestimator by a cutting plane algorithm

We consider the problem of determining the parameters $p \in \Re^s$ of the quadratic function $q(x, p)$ as given by (1), with specified bounds on the eigenvalues of its Hessian H, so that it underestimates a set of $m \geq s$ data points in $\Re^n$, and minimizes the error of approximation in the $L1$ norm. We show how this can be done efficiently by a sequence of linear programs, with one (or more) cutting plane added to each new linear program.

The s parameters to be determined are the elements h_{ij} of the symmetric matrix H (specified by $h \in \Re^t$), the vector $c \in \Re^n$ and the scalar α, as given by (1). We impose a large bound ρ on the absolute value of each parameter, so that p is contained in a hypercube in $\Re^s$. Thus, there are a total of $s = t + n + 1$ parameters to be determined, where $t = \frac{1}{2}n(n+1)$ is the number of parameters in h. We denote by $\lambda_1(h)$ and $\lambda_n(h)$ the minimum and maximum eigenvalues of $H(h)$. Also let $u_1(h)$ and $u_n(h)$ be the corresponding eigenvectors, with $\|u_1\| = \|u_n\| = 1$. We specify a lower bound μ_1 for $\lambda_1(h)$, and an upper bound μ_n for $\lambda_n(h)$. To simplify the presentation, we consider only the lower bound. The upper bound is imposed in an obvious manner similar to the lower bound (see QUBE Algorithm).

We consider the hypercube $\Omega \in \Re^t$, where $|h_{ij}| \leq \rho$. For each point $h \in \Omega$, the Hessian has a minimum eigenvalue $\lambda_1(h)$. Given a lower bound μ_1, the requirement $\lambda_1(h) = \mu_1$ defines a surface in Ω, which partitions Ω into two subdomains, $\Omega_+ : \lambda_1(h) \geq \mu_1$, and $\Omega_- : \lambda_1(h) \leq \mu_1$. Ω_+ and Ω_- are each closed and bounded.

Theorem 2.1 *The subdomain Ω_+ is convex.*

Proof Let h_1 and h_2 be any two points in Ω_+. We have $\lambda_1(h_1) \geq \mu_1$, and $\lambda_1(h_2) \geq \mu_1$. Now consider the point $\bar{h} = \frac{1}{2}(h_1 + h_2)$. We show that $\lambda_1(\bar{h}) \geq \mu_1$. We have $H(\bar{h}) = \frac{1}{2}(H(h_1) + H(h_2))$. Note that

$$\lambda_1(h_1) = \min_{\|z\|=1} (z^T H(h_1) z),$$
$$\lambda_1(h_2) = \min_{\|z\|=1} (z^T H(h_2) z) \quad \text{and}$$
$$\lambda_1(\bar{h}) = \min_{\|z\|=1} (z^T H(\bar{h}) z),$$

for $z \in \Re^n$. Then

$$\begin{aligned}\lambda(\bar{h}) = \frac{1}{2} \min_{\|z\|=1} [z^T (H(h_1) + H(h_2)) z] &\geq \frac{1}{2}(\lambda_1(h_1) + \lambda_1(h_2)) \\ &\geq \mu_1.\end{aligned}$$ □

We note that the hypersurface $\lambda_1(h) = \mu_1$ in Ω is the boundary of the convex subdomain Ω_+.

We now describe and justify the use of a cutting plane algorithm to determine the optimum parameter vector $p \in \Re^s$, such that $q(x, p)$ minimizes the approximation error, and its Hessian H has a minimum eigenvalue $\geq \mu_1$.

Consider any point $\bar{h} \in \Re^t$, such that $\lambda_1(\bar{h}) = \bar{\lambda}_1$. If $\bar{u}$ is the corresponding eigenvector to this minimum eigenvalue then $H(\bar{h})\bar{u} = \bar{\lambda}_1 \bar{u}$, or $\bar{u}^T H(\bar{h}) \bar{u} = \bar{\lambda}_1$, since $\|\bar{u}\| = 1$. But $\bar{u}^T H(h) \bar{u}$ is linear in the elements of h. Specifically, let

$$\begin{aligned} h^T &= (h_{11}, h_{22}, \ldots, h_{nn}, h_{12}, h_{13}, \ldots, h_{n-1,n}) \in \Re^t \quad \text{and} \\ g^T(u) &= (u_1^2, u_2^2, \ldots, u_n^2, u_1 u_2, u_1 u_3, \ldots, u_{n-1} u_n) \in \Re^t, \end{aligned} \tag{2}$$

where u_i is the ith element of u. Then $g^T(\bar{u})h = \bar{u}^T H(h)\bar{u} = \lambda_1(\bar{h})$ is a supporting hyperplane to the surface $\lambda_1(h) = \bar{\lambda}_1$ at $\bar{h}$. Note that $g(\bar{u})$ is the gradient of $\lambda_1(h)$ at $\bar{h}$.

Now, consider any point $\hat{h}$ in the interior of Ω_-. Then $\lambda_1(\hat{h}) = \hat{\lambda}_1 < \mu_1$, with corresponding eigenvector $\hat{u}$. The linear equality $g^T(\hat{u})h = \hat{u}^T H(\hat{h})\hat{u} = \mu_1$, will strictly separate $\hat{h}$ and the convex set Ω_+. Furthermore, the inequality

$$g^T(\hat{u})h \geq \mu_1 \tag{3}$$

will eliminate a finite region of Ω_- from the feasible set. Therefore, the inequality (3) is a cutting plane with respect to $\hat{h}$ and Ω_+.

The algorithm can now be summarized as follows. We first solve a linear program (LIP), which minimizes the approximation error in $q(x, p)$ in the $L1$ norm, subject to $3m$ linear inequality constraints, which define a polyhedral set $\hat{\Omega}$, which is an approximation to Ω_+ (see Sect. 3). The (LIP) gives a point $\hat{p} \in \Re^s$, and $\hat{h} \in \Re^t$. If $\lambda_1(\hat{h}) \geq \mu_1$, (i.e., $\hat{h} \in \Omega_+$), then $\hat{p}$ is the optimal solution to the problem. Otherwise, we add a cutting plane (3) corresponding to each eigenvalue of $H(\hat{h}) < \mu_1$. The (LIP) with the additional inequality constraints (3) is again solved. This is repeated until $\lambda_1(\hat{h}) \geq \mu_1 - \epsilon$, for some small tolerance ϵ. This will give the underestimator $q(x, p)$ with the minimum $L1$ norm approximation to the data, and $\lambda_1(\hat{h}) \geq \mu_1 - \epsilon$. We call this an ϵ-optimal solution to the QUBE problem.

In Sect. 3, we give the linear program formulation and the QUBE algorithm. In Sect. 4, we show that the algorithm converges in a finite number of steps, to an optimum point $\hat{p}$ in the parameter space. In Sect. 5, we give the computational results for problems with $n = 15$. Similar computational results were obtained for $n = 5$, 10 and 20. In all these computational results the optimal $\hat{p}$ was obtained with no more than six major iterations of the QUBE algorithm.

3 Linear program formulation and algorithm

Initially, we solve a linear program, which determines the parameter vector $p \in \Re^s$, so as to minimize the $L1$ norm of the error of the quadratic underestimator $q(x, p)$ in approximating the m data points (x_i, y_i). This LP is given by:

$$\min_{p \in \Re^s} \sum_{i=1}^{m} [y_i - q(x_i, p)] \tag{4}$$

(L1P) subject to

$$\begin{aligned} & y_i - q(x_i, p) \geq 0, \quad i = 1, \ldots, m, \\ & \mu_1 \|x_i\| \leq x_i^T H x_i \leq \mu_2 \|x_i\|, \quad i = 1, \ldots, m, \\ & -\rho \leq p_j \leq \rho, \quad j = 1, \ldots, s, \end{aligned} \tag{5}$$

where $q(x, p)$ is given by (1).

This LP has $s = O(n^2)$ bounded variables, and $3m + 2s$ inequality constraints, which represent the polyhedral set $\hat{\Omega}$.

The result of (L1P) is a parameter vector $\hat{p}$, and corresponding Hessian parameter vector $\hat{h}$. If $\lambda_1(\hat{h}) \geq \mu_1$, then $q(x, \hat{p})$ is the optimal solution to the underestimator approximation problem. Otherwise add one or more cutting plane inequalities and continue. Details are given in the following QUBE algorithm, which computes an ϵ-optimal solution to the QUBE problem.

Algorithm: Quadratic Underestimator with Bounds on Eigenvalues (QUBE)

```
1. Solve initial L1P to get $H$. Compute eigenvalues of $H$.
2. While ($\exists$ eigenvalues of $H \leq \mu_1 - \epsilon$ OR $\geq \mu_2 + \epsilon$)
    (a) Add cutting plane inequality constraints to L1P.
    (b) Solve L1P to get $H$. Compute eigenvalues of $H$.
```

Add cutting plane constraints:

- For each eigenvalue violating bounds, add a cutting plane inequality constraint.
- For eigenvalue $\lambda_j(H) \leq \mu_1 - \epsilon$, and u_j the corresponding eigenvector, add constraint $u_j^T H u_j \geq \mu_1$.
- For $\lambda_k(H) \geq \mu_2 + \epsilon$, add $u_k^T H u_k \leq \mu_2$.

4 Convergence of QUBE algorithm

Since, the QUBE algorithm is a cutting plane algorithm, the proof of its convergence follows that of a general cutting plane algorithm. (see, e.g., Bazarra and Shetty 1979, p 248]. We first prove several properties of the QUBE algorithm, and then show convergence of the algorithm. As we did in Sect. 2, to simplify the discussion, we assume that there is only a lower bound on the eigenvalues of H.

Lemma 4.1 *The sequence of matrices H generated by the iterations of the QUBE algorithm (recall that the solution p of LIP contains the entries of H) are contained in a compact set. Also, the eigenvectors u_j of the out-of-bounds eigenvalues (that are used to define additional cutting plane constraints) are also contained in a compact set.*

Proof These follow directly from the bound constraints placed on the elements of p in the linear program L1P, and from the continuity of the elements of eigenvectors with respect to the corresponding eigenvalues. □

Lemma 4.2 *At each iteration of the QUBE algorithm, the matrix H satisfies all of the cutting plane constraints so far generated by the algorithm.*

Proof This follows from the fact that the cutting planes are constraints of the linear program part of whose solution is H, and by the fact that no cutting plane constraints are ever removed from the linear program. □

Lemma 4.3 *The mapping from an out-of-bounds eigenvalue of H (i.e., $\lambda_j(H) < \mu_1$) to its corresponding eigenvector (u_j) is closed.*

Proof This follows from the continuity of the eigenvalues of a matrix with respect to the entries in the matrix, and from the continuity of the elements of eigenvectors with respect to the corresponding eigenvalues. □

Lemma 4.4 *In step 2 of the QUBE algorithm, if H has an out-of-bounds eigenvalue and a new cutting plane constraint is added ($u_j^T H u_j \geq \mu_1$), then the current value of H violates this constraint.*

Proof This is proven in Sect. 2, where we show any point in Ω_- (i.e., those H that have out-of-bounds eigenvalues) are strictly separated from Ω_+ by the added cutting plane constraint. □

Convergence Theorem *The QUBE algorithm converges in a finite number of steps to an ϵ-optimal solution, with the parameter vector p satisfying (L1P), and the eigenvalues of H between $\mu_1 - \epsilon$ and $\mu_n + \epsilon$, for any $\epsilon > 0$.*

Proof Consider a sequence of iterations of the QUBE algorithm. Without loss of generality, we can assume that at most one cutting plane constraint is added with each iteration. Then let $\{H_k\}$ be the sequence of matrices H generated by the iterations of the QUBE algorithm, and let $\{u_k\}$ be the sequence eigenvectors corresponding to the out-of-bounds eigenvalues. Lemma 4.1 implies that both $\{H_k\}$ and $\{u_k\}$ must have convergent subsequences. Let $\{H_k\}_\kappa$ and $\{u_k\}_\kappa$ be these subsequences, and let H and u be their limits. By Lemma 4.2, we can say that for any k, $u_k^T H_l u_k \geq \mu_1$ for all $l > k$. Because the mapping from H to its out-of-bounds eigenvalue is closed (Lemma 4.3), we can conclude that in the limit this inequality is also true; that is, $u^T H u \geq \mu_1$. H, however, cannot have any out-of-bounds eigenvalues, because otherwise this inequality violates Lemma 4.4. This in turn implies that the vector p that contains Hmust be optimal, because p is the solution to (LIP), and the eigenvalues of H are within its bound. Convergence in a finite number of steps is achieved by requiring only that the eigenvalues are not out-of-bounds by more than ϵ. □

5 Computational results

The QUBE algorithm has been tested on a range of problems in n-dimensional space, with $n = 5$ up to $n = 20$. The data points to be approximated by the quadratic underestimator were generated by simulating the energy surface for a protein-ligand docking problem (Mitchell et al. 1999; Rosen and Marcia 2004; Marcia et al. 2005). Each of the m points represents a local minimum of the energy surface, which is assumed to be basin-shaped with many local minima. We used $m = 2s$, and for $n = 15$, this gives $t = \frac{1}{2}n(n+1) = 120$, and $s = t + n + 1 = 136$, so that $m = 272$. The energy surface function was defined in a hypercube with edges of length 10. The basin-shaped function was represented by perturbing a convex quadratic function $f(x)$, where $f(x) = b^T x + \frac{1}{2}x^T Qx$. More specifically, each of the local minimum points (x_i, y_i), $i = 1, \ldots, m$ was generated by randomly choosing each x_i in the hypercube interior, and then letting $y_i = f(x_i) + \eta$, where η is the perturbation, randomly chosen in $[-\gamma, \gamma]$. Thus each data point (x_i, y_i) is assumed to be a local minimum of the energy surface, and the minimum $x_p = -H^{-1}c$ of the quadratic underestimator $q(x, p)$ is used as a prediction of the true global minimum $x_{g\min}$ of the simulated energy surface. As is explained below, this global minimum point is chosen so as to be contained in the hypercube. A successful quadratic underestimator will typically be a good approximation to $f(x)$, so that $\|x_p - x_{g\min}\|$ should be relatively small. This distance will of course increase with increasing γ.

The matrix Q used to construct the simulated energy surface $f(x)$ is generated as follows. An $n \times n$ matrix A with random elements in [0, 9] is first formed. Then the symmetric, positive definite, matrix $Q = A^T A + \delta I$, $\delta > 0$, is computed. The value of δ is specified so as to impose a reasonable upper bound on the condition number of Q. The vector b is determined by $x_{g\min} = -Q^{-1}b$, where $x_{g\min}$ is chosen as a strictly interior point of the initial hypercube. Suppose that the imposed lower and upper bounds μ_1 and μ_n, on the eigenvalues of H, are such that $\mu_1 \leq \lambda_1(Q)$ and $\mu_n \geq \lambda_n(Q)$, where $\lambda_1(Q)$ and $\lambda_n(Q)$ are the minimum and maximum eigenvalues of Q, and the condition number of Q, cond(Q), is given by $\lambda_n(Q)/\lambda_1(Q)$. Then if $\gamma = 0$, the solution to L1P will give $q(x, p) = f(x)$, with zero error, and all m data points interpolated. We will also have $H = Q$, and $c = b$, so that $x_p = x_{g\,\min}$. For $\gamma > 0$, we will get $x_p \neq x_{g\,\min}$, and only some of the data points will be interpolated (usually approximately half of them). On the other hand, if one or both of the inequalities for μ_1 and μ_n are not satisfied then even for $\gamma = 0$ we cannot get $H = Q$, so $\|x_p - x_{g\,\min}\| > 0$. However, for $\gamma > 0$ the value of $\|x_p - x_{g\,\min}\|$ may actually be smaller, for any specific $\gamma > 0$, when $\mu_1 > \lambda_1(Q)$ and $\mu_n < \lambda_n(Q)$, so that the condition number of H is less than that of Q. To investigate this, we tested the two possibilities:

$$\text{Type A bounds}: \mu_1 = \lambda_1(Q) - \sigma, \quad \mu_n = \lambda_n(Q) + \sigma,$$
$$\text{Type B bounds}: \mu_1 = \lambda_1(Q) + \sigma, \quad \mu_n = \lambda_n(Q) - \sigma,$$

where σ is a specified positive constant. Typically, we get $\lambda_1(H) \cong \lambda_1(Q)$ and $\lambda_n(H) \cong \lambda_1(Q)$, for type A bounds, and $\lambda_n(H) \cong \lambda_1(Q) + \sigma$ and $\lambda_n(H) \cong \lambda_n(Q) - \sigma$, for type B bounds. Therefore, cond(H) will be smaller for type B than for type A bounds.

We now summarize the results of the computational tests carried out. Several hundred individual runs were made, using values of $n = 5,10,15$ and 20. A different random seed was used for each run, so that the matrix A (and therefore, Q) was different for each run. The results obtained for $n = 15$ are given in Table 1. The results for other values of n were similar. For the results presented the parameter values $\delta = 1.5$, $\sigma = 1.0$ and $\epsilon = 0.1$ were used. The minimum eigenvalue of Q was in the range [1.50, 1.74], the maximum in the range [42.9, 61.0] and the condition number of Q was in the range [28.5, 39.9]. The number of data points used was 272, and the number of parameters, $s = 136$, so for this overdetermined system there will always be an error in the underestimating approximation, except for the special case of type A bounds and $\gamma = 0$. In general, about 50% of the data points will be interpolated.

Table 1 shows, for each value of γ and each bound type, the following results: (1) Number of initial violations of the imposed lower and upper bounds. (2) Number of iterations required to satisfy the termination test. (3) Number of data points interpolated. (4) Average CPU time required to solve. (5) Edge length of reduced hypercube known to contain both x_p and $x_{g\,\min}$. For (1), (2), (3) and (5) the minimum and maximum value of each quantity, and for (4), the average time, is shown. The minimum and maximum values, and the average, are over all runs made for the specified bound type and γ value. Typically, for bound type B, both lower and upper bounds were initially violated.

Table 1 Table of results for $n = 15$

Bound type	γ	# Init. violats		# Itns.		# Interp. Points		CPU Time(s)	Hypercube edge len.	
		Min	Max	Min	Max	Min	Max	Avg	Min	Max
A	0	0	0	1	1	272	272	151	0	0
A	5	0	0	1	1	136	136	155	0.036	0.050
A	10	0	0	1	1	136	136	178	0.110	0.156
A	20	0	0	1	1	136	136	183	0.118	0.360
A	40	0	1	1	3	134	136	167	0.378	0.790
B	0	3	4	4	4	131	131	239	0.104	0.110
B	5	3	4	2	5	129	133	191	0.100	0.120
B	10	3	4	2	5	129	133	189	0.132	0.186
B	20	3	4	3	5	131	132	189	0.170	0.244
B	40	3	3	5	6	131	131	169	0.328	0.372

$n = 15$, $m = 272$, $\delta = 1.5$, $\sigma = 1.0$ and $\epsilon = 0.1$

The solution time depends primarily on the time required to solve the initial (LIP). Since, the number of both variables and inequality constraints is $O(n^2)$, it is reasonable to assume that the time dependence of the initial linear program will be $O(n^4)$. This is what we observed for $n = 5$,10,15 and 20. The total solution time for $n = 15$ is seen to be approximately 3 min. The total solution time for $n = 5$ was approximately 1 s. No significant dependence of total solution time on the number of iterations required is seen, at least for the number of iterations ≤ 6. This is because we add only one inequality constraint to the primal (LIP) for each violated bound. The resulting LP is then solved as the dual problem, starting with the optimal basis from the initial (LIP). Since each violated bound simply adds a column to the dual tableau, this only requires a few basis changes to achieve optimality. Therefore, the additional time required is negligible, unless many bounds are violated.

The effectiveness of the QUBE algorithm in predicting $x_{g\,\min}$ is measured as follows. Initially, we know that $x_{g\,\min}$ lies in a hypercube in $\Re^n$ with edge length 10, so its volume is 10^n, and this is the initial search domain. For each run we determined the smallest hypercube, with x_p at its center, that contained $x_{g\min}$. Since both x_p and $x_{g\min}$ are known, this is easily determined. We denote by d the length of an edge of this reduced hypercube. The reduced hypercube will have a volume of d^n, so that after applying the QUBE algorithm, the search domain volume will have been reduced by a factor of $(10/d)^n$. For $n = 15$, the maximum $d \leq 0.79$, and the minimum $d \leq 0.38$. Therefore, the hypercube volume was always reduced by a factor of at least $(10/0.79)^{15} = 34.3 \times 10^{15}$, for the $n = 15$ tests.

We note from Table 1 that the size of the reduced volume is smaller for type B bounds (and larger values of γ). We also noted earlier that cond(H) is smaller for type B bounds. This result is also confirmed by other tests. This suggests that a good general strategy in typical docking applications (when Q is not known) is to choose the bounds μ_1 and μ_n so that cond (H) is relatively small, say ≤ 30. This is easily done by letting, for example, $\mu_1 = 1.0$ and $\mu_n = 30$.

In a realistic model of a docking energy surface, the true global minimum, $x_{g\,\min}$, is of course not known. The assumption is that the predicted global minimum, x_p, is close to $x_{g\,\min}$, and that $x_{g\,\min}$ is contained in the reduced hypercube volume centered at x_p, as in the test cases. To actually locate $x_{g\,\min}$ more accurately in a real docking application, a final search in the reduced hypercube is carried out. Typically, this would consist of local minimizations from randomly chosen initial points in the reduced hypercube. The local minimum point with the lowest energy value would then be chosen as $x_{g\,\min}$.

Acknowledgement The work of the first author was supported in part by the National Science Foundation grant 0082146.

References

Alizadeh, F., Haeberly, J.P.A., Overton, M.L.: Complementary and non-degeneracy in semidefinite programming. *Math. Prog.* **77**, 111–128 (1997)

Bazarra, M.S., Shetty, C.M.: Nonlinear Programming, pp. 248–250. Wiley, New York (1979)

Dill, K.A., Phillips, A.T., Rosen, J.B.: Cgu: an algorithm for molecular structure prediction. In: Large-scale Optimization with Applications, Part III, vol. 94 of IMA Vol. Math. Appl. pp. 1–21. Springer, Berlin (1997)

Marcia, R.F., Mitchell, J., Rosen, J.B.: Iterative convex quadratic approximation for global optimization in protein docking. Comput Optim Appl **32**, 285–297 (2005)

Mitchell, J.C., Rosen, J.B., Phillips, A.T., Ten Eyck, L.F.: Coupled optimization in protein docking. In: Proceedings of the Third Annual International Conference on Computational Molecular Biology, pp. 280–284. ACM Press, New York (1999)

Powers, R.A., Morandi, F., Shoichet, B.K.: Structure-based discovery of a novel, non-covalent inhibitor of ampc beta-lactamase. Structure **7**, 1013–1023 (2002)
Rosen, J.B., Marcia, R.F.: Convex quadratic approximation. Comput Optim Appl **28**, 173–184 (2004)
Veinott, A.F.: The supporting hyperplane method for unimodal programming. Oper Res **15**, 147–152 (1967)
Wei, B.Q., Weaver, L., Ferrari, A.M., Matthews, B.M., Shoichet, B.K.: Testing a flexible-receptor docking algorithm in a model binding site. J Mol Biol **5**, 1161–1182 (2004)