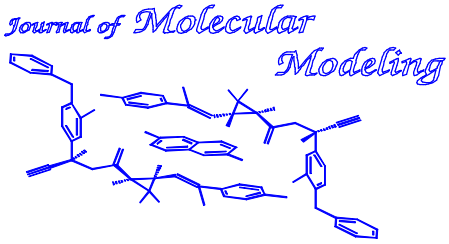
J. Mol. Model. **1995**, *1*, 161 –175



AdaptivSearch: A Novel Iterative Algorithm for the Generation of n-dimensional Hypersurfaces Using an Adaptive Numerical Strategy

Part 2 in the series "Novel methods in computational chemistry"; for part 1 see ref. Bringmann, G.; Güssregen, S.; Busse, H. *J.Comput.-Aided Mol. Design* **1992**, *6*, 3821.

Gerhard Bringmann*, Klaus-Peter Gulden, and Daniel Vitt

Institut für Organische Chemie, Universität Würzburg, Am Hubland, D–97074 Würzburg, Germany (gulden@chemie.uni-wuerzburg.de)

Klaus Birken*, and Clemens Helf

Rechenzentrum Universität Stuttgart, Abteilung für Numerische Höchstleistungsrechner, Allmandring 30, D–70550 Stuttgart, Germany (birken@rus.uni-stuttgart.de)

Received: 27 June 1995 / Accepted: 12 July 1995

Abstract

AdaptivSearch is the first adaptive strategy based algorithm for the rational and economi-cal construction of n-dimensional hypersurfaces. *AdaptivSearch* works iteratively: At each step it parcels out the definition range into several triangles, evaluates the worst according to a built–in error criterion, and refines the approximation to the unknown function by choosing the barycenter of this partial area as the node to be calculated next. Based upon the error criterion, *AdaptivSearch* selectively approaches those parts of the hypersurface in which the curvature exhibits the strongest changes. Some examples of *AdaptivSearch* applications for both analytical functions and chemical model surfaces are given in order to demonstrate the behavior of the algorithm. These show its broad applicability and the usefulness, especially for chemical problems.

Keywords: *AdaptivSearch*, construction of hypersurfaces, adaptive numerical strategy, iterative algorithm, parallel computing, Delauney triangulation

Introduction

Despite exponentially increasing computer resources and optimized software, there is a chronic shortage of CPU time – because simultaneously expectations and demands on the quality of the results and their presentation are also strongly increasing. Particularly, for scientific problems that depend on several parameters and that have to be solved by construction of multidimensional hypersurfaces *via* computer simulations, the required calculation time is usually the limiting factor.

The quality of the approximation to the unknown n–dimensional function depends on two factors: on the one hand, the exactness depends on the significance and quality of the calculated function value for a particular “node”[1] required for the generation of the function. This, *e.g.* in the case of chemical potential energy surfaces (PES), might be attained by using a larger basis set in quantum chemical procedures.

* *To whom correspondence should be addressed*

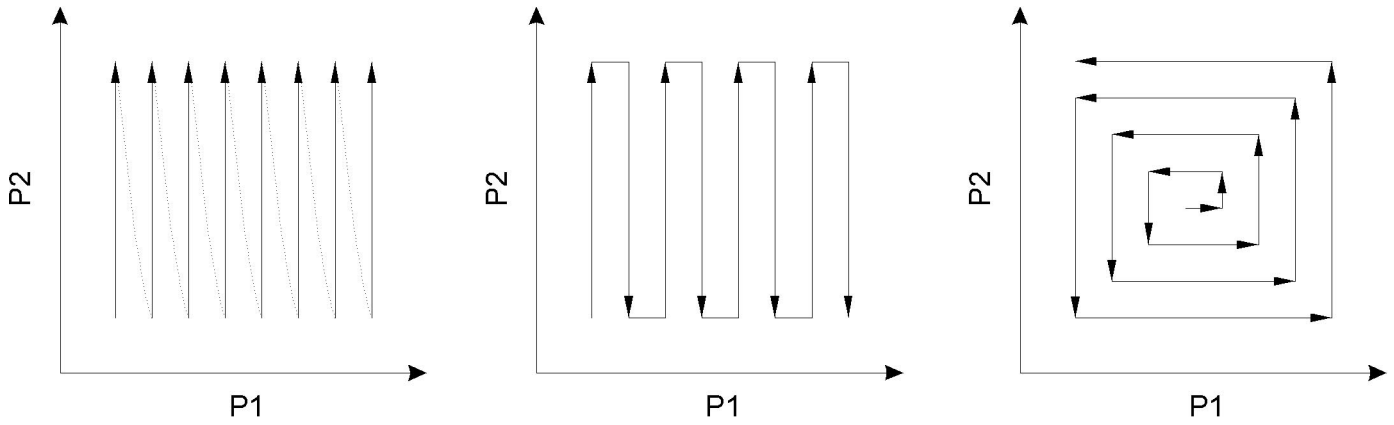


Figure 1: *Three versions of GridSearch to go through the definition area for the two-dimensional space defined by the parameters P1 and P2.*

On the other hand, it is important how well the "ensemble", *i.e.* the set of chosen points, represents the actual function.

A current and pragmatical method for the generation of such hypersurfaces is the commonly used "GridSearch" [2] algorithm. In this procedure, the range of definition is explored by scanning along parallels of the coordinate axes in equidistant steps. GridSearch, which is very easy to implement algorithmically, generates a regular grid over the interesting area and determines the function value at each lattice point (Figure 1).

The advantage of the procedure, to address the next node to be measured rapidly with a minimum decision overhead, is only relevant for problems for which the calculatory expense per calculated (or measured) node is very small. For nodes that consume a large amount of computational time, the strategy even turns out to have grave disadvantages: Only *after* having completed the scanning process does it become manifest whether unnecessary calculations were performed for some parts of the surface, or, *vice versa,* whether other parts were not sufficiently scanned.

In every day computer–chemistry, the demands on potential energy surfaces (PES) may vary greatly, depending on the particular problem: Whereas in some cases only energetic minima on the surface (*i.e.* molecular conformations) are important for further investigations, it is, in other cases, only the saddle points (*i.e.* the transition states between two stable ground state geometries) that are of interest. Still more demandingly, for the calculation of nuclear wave functions [3] a highly accurate PES is necessary.

In the literature [4], there is so far no specific method for the construction of PES that delivers a surface with a pre–defined exactness based on a minimum number of measured nodes, or, alternatively, that allows a stepwise enhancement of precision at any time of the process, either for parts or for the whole area of definition, relying on the already determined points.

In this paper, we present a fundamentally novel procedure, named *AdaptivSearch*, which is capable of adapting itself iteratively to the problem. It selectively explores the strategically important parts of the potential energy surfaces preferentially. The *AdaptivSearch* algorithm thus allows the most effective possible use of the given computational resources.

Fundamental principle and mathematical description of *AdaptivSearch*

We have developed the *AdaptivSearch* method for the fast and rational computation of multi-dimensional hypersurfaces by an adaptive approach. During a typical *AdaptivSearch* run, the set of nodes of the problem domain already computed is enhanced iteratively by calculating new nodes according to a problem-specific error criterion. Within each iteration, a Delauney triangulation [5] of the existing set of nodes is constructed. The error criterion can then be computed for each triangle.

The triangulation approximation problem (TAP)

Throughout this work the task of generating n-dimensional hypersurfaces will be treated as an iterative approximation problem that can be stated in a general way as follows:

Let $f : \Omega \rightarrow R$ be a function of n variables $x_1 ... x_n$ in the n-dimensional problem domain Ω. The triangulation approximation problem (TAP) is to determine a set of interpolation nodes $\vec{x}_i \in \Omega$ and a triangulation T based on these nodes, so that the piecewise linear, continuous approximation function g with $\forall i$:

$g(\vec{x}_j) \stackrel{!}{=} f(\vec{x}_j)$ is an approximation for f minimizing the global error with respect to a certain error norm.

Hereby the triangulation T of Ω is defined as a set of triangles underlying two conditions [6]:

1. Within each pair of triangles only one edge or one node overlaps.
2. The union of all triangles equals Ω.

A practical error norm definition is, *e.g.* the L^2 norm, where $\int || (f - g)^2 || dx$ is minimized during the iterative TAP solution process.

Assumptions and restrictions

Function f has two continuous derivatives. This can be assumed for physical reasons, as well as from a correct choice of the parametric space.

Two-dimensional problems. Although the *AdaptivSearch* method will work for approximation problems independent of their spatial dimension, the following explanations will be based on the two-dimensional case in order to keep the algorithm description simple and understandable. Some comments on problems of more than two dimensions will be given in the paragraph *"Multi–dimensional problems"*.

Costly function evaluation. The approximation of function f is necessary because of the computational expense of evaluating individual values of f . Taking this assumption into account, a small fraction of this computational time can be used for the *AdaptivSearch* strategy itself.

Algorithmic overview

Solving the TAP with *AdaptivSearch* is basically an iterative method, with the number of nodes increasing during the iteration process. Within each iteration, a Delauney triangulation [5] of the existing set of nodes is constructed. A local error criterion is then used to find the nodes to be evaluated next. Scheme 1 describes the *AdaptivSearch* algorithm in more detail.

In the case of a rectangular problem domain (*i.e.* a two-parameter problem), the starting set (line 01) may contain only the four nodes in the corners of the domain. As an alternative, a larger set of nodes from a prior computation may be used (*evaluation recycling*).

Scheme 1: *Description of the AdaptivSearch execution scheme in detail*

	Algorithm *AdaptivSearch*
01:	Determine set of starting nodes P_0
02:	Set iteration count $i = 0$
03:	While $P_i \neq \varnothing$
04:	For each $x_j \in P_i$
05:	Compute $f(x_j)$
06:	End For
07:	Create triangulation T_i on $\cup_i P_i$
08:	Compute local errors on T_i
09:	Determine set of new nodes P_{i+1}
10:	Next iteration: $i = i + 1$
11:	End While

During the main loop (lines 03-11), at first the new nodes are evaluated, which normally takes the largest part of the computational time. After this step (lines 04-06), the actual *AdaptivSearch* strategy is executed. For this purpose, a Delauney triangulation is constructed to obtain an approximation for f based on $\cup_i P_i$, which is the set of all nodes computed so far. The Delauney triangulation T_i is a triangulation (as defined above) based on an additional geometric condition [5] concerning the numerical stability.

In order to obtain a list of nodes to be computed next, the triangles have to be sorted according to an error criterion. This criterion is applied to each triangle to evaluate the quality of the current approximation of f (line 08; for details, see paragraph *"Error criteria"*).

As a last step during each iteration, the barycenters of the worst triangles are chosen as new nodes (line 09). In order to ensure consistency at problem domain boundaries, additional new nodes are generated at the boundary edge midpoints of all chosen triangles.

Finally, the next iteration begins with the computation of all new nodes P_{i+1} . The algorithm ends as soon as P_i is empty, *i.e.* when no further triangles are chosen or the upper limit of nodes has been reached.

The *AdaptivSearch* method is based on a triangulation discretization, which is similar to well–known *finite–element* methods used for the solution of partial differential equation problems (for general introductions, see [7, 8]; for an application in the computational chemistry field, see [9]). The kind of adaptive refinement being applied in each of the two methods, however, differs in its triangulation lifetime: during adaptive finite-element computations [10], a sequence of triangulations is constructed by refinement of certain triangles from the previous, coarser mesh. Because of the comparatively low computational effort per node during the iteration scheme, it is necessary to reuse existing grids. Thus, the majority of topological information describing the grids can be kept during the computation process. With *Adaptiv-Search*, the computational time per node is much higher, therefore one can afford to create all triangulations from scratch. Hence, in each *AdaptivSearch* iteration the optimal triangulation can be used.

Error criteria

Figure 2 shows the two main strategic aspects of *Adaptiv-Search*. The *local error criterion* is applied to each element of the triangulation as a measure for the local quality of the approximation. Afterwards, all elements are sorted according to this local quality measure. Finally, the *global strategy* decides how many new nodes will be chosen, starting from the top of the sorted list.

These two aspects are mutually dependent because they alternate during the iteration process. For a wide range of TAP problems (especially for those fromthe field of computational

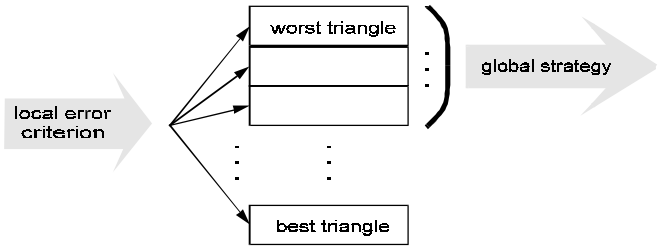


Figure 2: *Schematic outline for generating the sorted list of "worst" triangles; the user can determine how many triangles are to be refined.*

chemistry presented here), the following strategy has proved to be most effective and generally applicable:

Local error criterion:
For each triangle i the local error is computed using the neighbor gradients ∇j with $j \in nb_i$,where nb_i is the set of neighbors of i. With $\lambda(i)$ being the xy-projected volume of triangle i, the local error is given by

$$\text{error}_{local}(i) = \frac{\sum_{j \in nb_i} \left\| \nabla_i - \nabla_j \right\|}{\# nb_i} \lambda(i)$$

Global strategy:
Only the triangle with the worst local error is chosen for the generation of the next nodes. Its barycenter is the first new node. If the edge of the corresponding triangle is a boundary of Ω, its edge mid points are chosen as additional further nodes.

Parallel Processing

The current *AdaptivSearch* prototype consequently has been implemented in a modular manner using several standard software components. Furthermore, embedding it into a typical UNIX-compatible environment allows the usage of suitable network features including complete support for utilizing heterogeneous computing environments.

Three different scenarios representing possible types of parallelism may be used without modification of *AdaptivSearch*:

1. Fine-grain parallelism: Clusters of workstations as available in most computational chemistry labs may be used for the evaluation of single nodes: Programs with highly parallellized subroutines (*e.g.* for the solution of eigenvalue problems or for the filling of large matrices, etc.) serve as black–box evaluators for nodes instead of the sequential program versions on one single processor.
2. Node parallelism: The evaluation of all nodes in the current set P_i can be tackled com-pletely in parallel (*"Algorithmic Overview"*, line 04-06). Different global strategies (see paragraph above) for determining the sets *Pi* can be used in order to adjust the necessary degree of parallelism.
3. Task parallelism: For each single node, two or more evaluations may be computed with slightly different starting molecule geometries. This can be used for handling anomalies which occur when modeling high–dimensional problems with low–dimensional parameter sets [12]).

All three approaches maybe used simultaneously for distributed computations in a typical computer center environment. Thus, the central *AdaptivSearch* strategy manages the solution process by exploiting the power of high performance computing hardware combined with fast network connections.

Multi–dimensional problems

Most parts of the algorithm described above can be applied easily to the solution of problems in more than two spatial dimensions. Only the postprocessing facilities and the triangulation algorithm will not be usable for higher dimensional problems. Although visualization of solutions in three and more space dimensions is an important issue, it will not be considered further in this paper, but will be described later.

Triangulation algorithms for 3D data may be adapted from well-known grid generation research from Computational Fluid Dynamics. In this case *tetrahedra* are created covering the complete 3D problem domain.

Problems in four and more space dimensions requiring additional efforts for the triangulation [11] and visualization are subject to current research.

Application of the algorithm

As mathematically depicted above, *AdaptivSearch* differs distinctly from other methods: For the search of new nodes, it neither follows an anticipated and thus rigid strategy such as *e.g.* for "GridSearch", nor is the decision which node is to be calculated next left to chance as is typical for the RandomSearch approach [13]. Per each iteration step, *AdaptivSearch* discretizes the definition area into polyhedral partial areas according to the already calculated nodes and subsequently evaluates each of those polyhedra relative to the neighboring areas using the "gradients difference criterion" (see *"Error criteria"*). The specific shapes of these partial areas are triangles for two-dimensional space, tetrahedra for three-dimensional space, etc. According to its definition, the error criterion can be interpreted as a sort of discrete 2nd average derivative. Thus, with each refinement step, *AdaptivSearch* rapidly elaborates more and more detailed information of the behavior of the unknown hypersurface by advancing preferentially into those areas in which the curvature of the hypersurface changes strongly. Likewise, from small amounts of local errors, *AdaptivSearch*

is capable of recognizing in which parts of the definition range the gradient of the energy function is relatively constant and where consequently the function may feasibly be described by a piecewise linear approximation.

In order to demonstrate the scope and limitations of *AdaptivSearch*, we have chosen seven analytical model functions with different properties and have probed, *i.e.* scanned, them using *AdaptivSearch*. Exemplarily for the first three chosen functions (examples 1–3 in *"Analytical functions"*), we focus on the general behavioral patterns and characteristics of the algorithm. As *AdaptivSearch* was initially developed primarily for the investigation of chemically relevant potential energy surfaces (PES), we wish to show the broad applicability of this procedure for the solution of chemical problems exemplarily using the functional model surfaces 4-7 (*"Chemical model surfaces"*).

By adding the error values over all partial areas and dividing by the number of triangles, one obtains a value for the average error of the approximation, in the following denoted as error*avr*. With increasing number of nodes, the quality of the approximation to the function is enhanced and the error*avr* decreases. Interestingly, for *AdaptivSearch* the decrease of $error_{avr}$ can be described by a polynomial relation. This becomes evident when plotting $error_{avr}$ against the number of calculated nodes in a doubly logarithmic diagram, which results in a straight line that may be formulated as

$$p = a \cdot q + b;$$

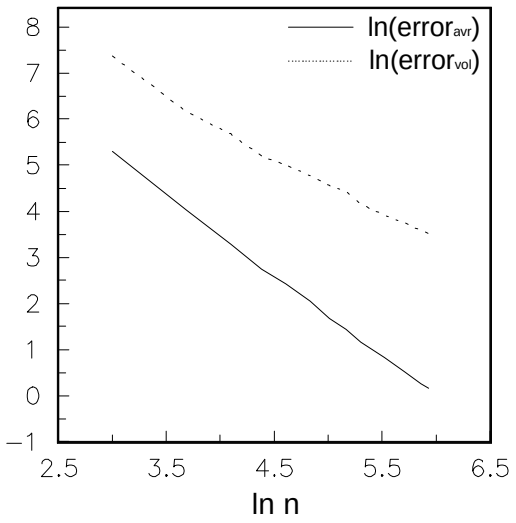


Figure 3. *Plotting the error(avr) value (full line) against the number of calculatednodes one gets a straight line. This is shown exemplarily for the example function no. 1. When the error(vol) is plotted in the same way (dotted line), one also gets a straight line, but with a slightly smaller gradient.*

from which, by simple transformations ($p \rightarrow \ln(error_{avr})$, $q \rightarrow \ln n$),

$$\ln(error_{avr}) = a \cdot \ln n + b$$

$$error_{avr} = e^{a \cdot \ln n + b} = e^{b} \cdot e^{a \ln} = e^{b} \cdot e^{\ln n^{a}}$$

the hyperbolic relation

$$error_{avr} = e^{b} + n^{a}$$

can be obtained. Figure 3 illustrates this relation for the example function no. 1. The coefficients a and b, *i.e.* the gradient of the straight line and the intersection point with the y-axis, can be read from a regression line, which can be obtained after calculation of only a few (15–20) nodes. This allows the straightforward possibility of easily predicting the number of further iteration steps required to reach a desired quality of the hypersurface investigated relative to the built–in error criteria.

We show by means of the 7 example functions how well this built–in error measurement corresponds to the real error of approximation. For this purpose, it is useful to regard the difference volume between the approximation and the actual function as a measure for the actual error, which is denoted as error*vol* in the following. When $error_{vol}$ is plotted against the number of calculated points in the manner described above, astonishingly the following empirical relation was found (see

Table 1. *Gradients of the regression lines obtained by plotting $error_{avr}$ and $error_{vol}$ against the number of calculated nodes n in a doubly logarithmic diagram. Whereas the absolute values of the gradients expressing the expenditure for AdaptivSearch may vary greatly, the difference between respective values (last column) is constant.*

	gradients		
function no.	**$error_{avr}$**	**$error_{vol}$**	**Δ**
1	-1.748	-1.264	0.484
2	-1.357	-0.876	0.481
3	-1.503	-0.997	0.506
4	-1.542	-1.058	0.484
5	-1.619	-1.080	0.539
6	-1.545	-1.034	0.511
7	-1.611	-1.071	0.540
			$\bar{n}$ = 0.506

Figure 3, for a complete list of all example functions, see Table 1).

$$\text{error}_{vol} = const \cdot n^{a+0.5}$$

where n denotes the number of calculated nodes and a is the gradient of the regression line from the doubly logarithmic diagram of error_{avr}.

For the deduction of this *con*st. factor, which is needed for the calculation of the **absolute** amount of the error_{vol}, we start with formula (1):

$$\text{error}_{vol} = \left|V(n) - V_{exact}\right| = const \cdot n^{a+0.5} \qquad (1)$$

$V(n)$ is the volume at the actual approximation level. In order to eliminate the exact volume V_{exact}, which is unknown in real cases, we differentiate (1) by n and get:

$$\frac{d}{dn}\left|V(n) - V_{exact}\right| = \frac{d}{dn}\left(const \cdot n^{a+0.5}\right)$$

$$\left|\frac{dV(n)}{dn} - 0\right| = const \cdot (a+0.5) \cdot n^{a-0.5}$$

$$\Rightarrow \quad \left|\frac{V(n) - V(n+\Delta n)}{\Delta n}\right| = const \cdot (a+0.5) \cdot n^{a-0.5} \qquad (2)$$

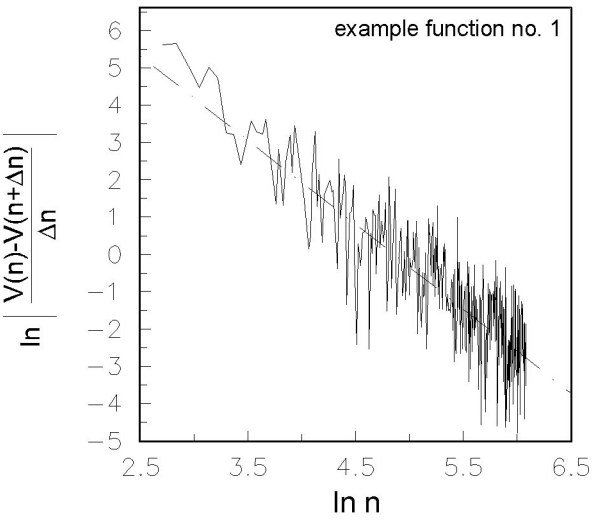


Figure 4. *In order to obtain information on the absolute error of the approximation, one has to plot the discrete derivative [see equation (2)] against the number of calculated nodes in a doubly logarithmic diagram. This simultaneously again proves the polynomial relation between error(vol) and the calculated number of nodes.*

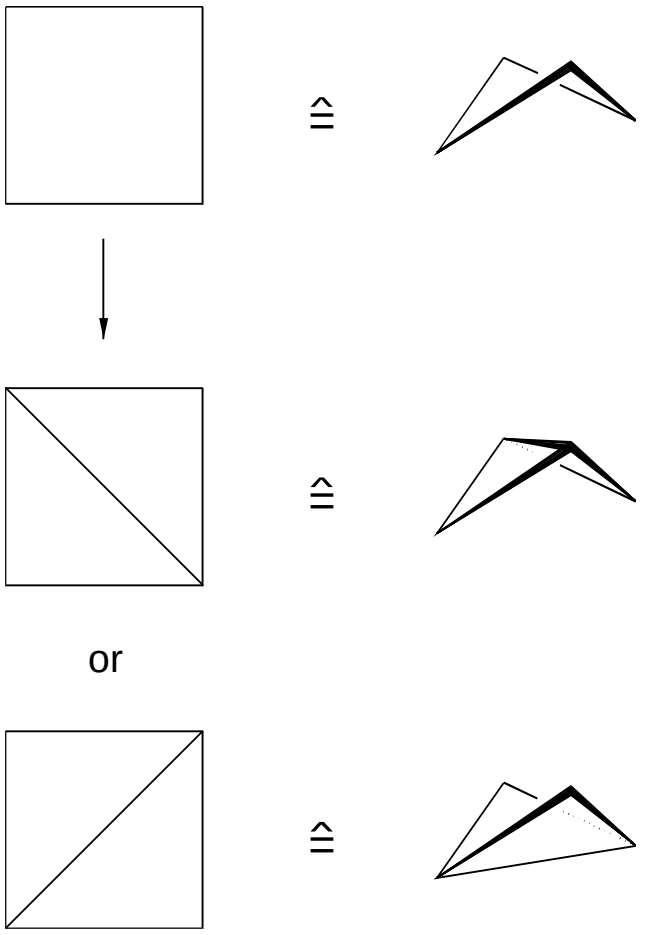


Figure 5. *Subdivision of the local area elements delivered by Grid Search leading to two different (since concave/ convex) approximations to the unknown function.*

By the last step, the infinitesimal derivative is converted to a discrete one, which is accessible at any time of an *AdaptivSearch* run. Again by plotting the term on the left side against the numer of calculated nodes (shown for example function no. 1 in Figure 4) in a doubly logarithmic diagram, one can extract the missing factor.

Thus at a very early point of the investigation this approach allows a concrete prediction of theexpenditurerequired for *AdaptivSearch* for the exploration of the function up to a needed or defined accuracy – an inestimably valuable help for the user, especially for costly nodes.

Although in the following examples the efficiency of *AdaptivSearch* is discussed relative to that of the conventional GridSearch method, a direct comparison of the two algorithms is not possible, as seen for the two aspects described below, which constitute both distinct arguments for *AdaptivSearch* and against GridSearch.

1. By the scanning process, GridSearch does not deliver direct and unambiguous information about the ensemble of elemental areas. This can easily be explained for two–dimen-

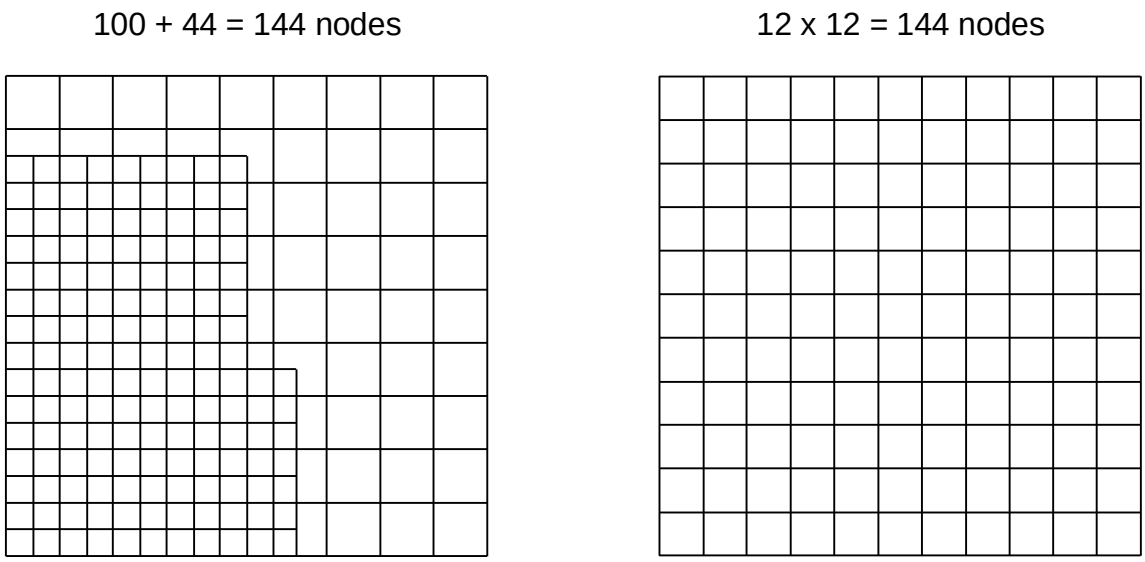


Figure 6. *Applying GridSearch, one obtains totally different discretenesses of the definition area, although an identical number of nodes have been calculated (here 144). Left: Arbitrarily refinement of a 10 x 10 lattice by additional 44 nodes; right: a regular grid of 12 x 12 nodes.*

sional space: The smallest, *i.e.* fundamental spatial elements as obtained by GridSearch are quadrangles (frequently squares), as defined by the 2D projection of four directly neighboring nodes of the surface. Still, since four nodes in the 3D space do not unambiguously define a plane, the quadrangles have to be subdivided into two triangles each. As Figure 5 illustrates, this can be done in two ways (that are degenerate in the 2D projection), which, however, may deliver dramatically different approximations to the functions.

Concretely speaking, the upper solution in Figure 5 (if evaluated from above) would give rise to a convex curvation of the approximation surface, whereas the lower one would give a concave course for this local area element.

Alternatively, by application of an interpolation algorithm [14] to the resulting grid or by fitting the set of all nodes to a spline function [15], it is also possible to effect an unequivocal, *i.e.* unique (unambiguous), record from all nodes calculated so far. However, one risks ending up with a misleading, since arbitrary interpretation of the ensemble as obtained by GridSearch.

This is due to unwanted fluctuations between the data points or inadequate accuracy of the approximation.

2. For the GridSearch strategy, the calculatory expense is set from the beginning of the calculation by the definition of the number of the scanning steps in each direction. An *a posteriori* increase of exactness by including further nodes is problematic. This can meaningfully only be done by a renewed regular subdivision of the established grid, since for the GridSearch algorithm all partial areas have to be treated identically. Consequently, this already existing lattice can be further refined only by a fixed (and thus rigid) quantity of additional nodes. If, nonetheless, by an *a posteriori* refinement of the discreteness less nodes are chosen, this results in an arbitrarily inhomogeneous subdivision of the definition area (Figure 6). Furthermore, the discreteness of a definition area, *e.g.* 144 nodes by adding 44 additional nodes to a preformed 10 x 10 lattice, looks completely different from a lattice that has been calculated from 12 x 12 nodes from the beginning (Figure 6).

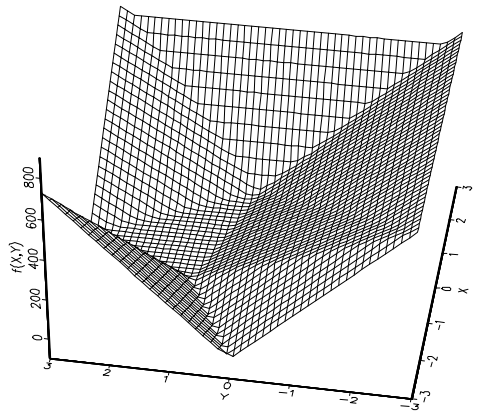
It is already clear at this point that *AdaptivSearch* will by no means be restricted to the solution of chemical problems, but should also be widely applicable to the investigation of completely different surfaces.

Analytical functions

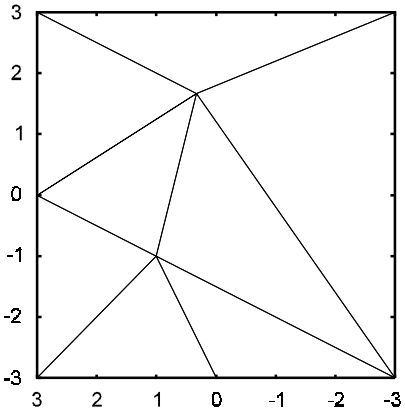
1. $f(x; y) = x^2 \cdot y^4$ with $x \in \{-3{:}0; 3{:}0\}$, $y \in \{-3{:}0; 3{:}0\}$:

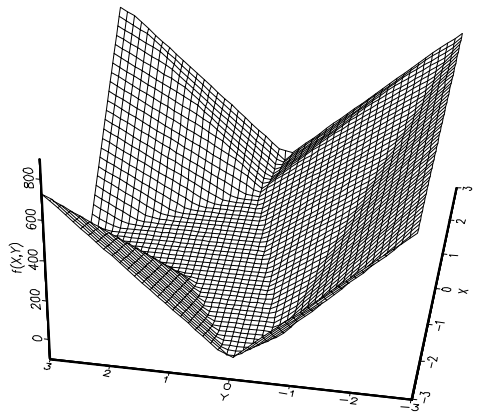
By this exemplary function, we illustrate exhaustively how a typical *AdaptivSearch* run looks in practice and what profit a user can draw from the data and the additional information that the *AdaptivSearch* program package delivers during the search process. While *Adaptiv-Search* is directing the corresponding calculating program to determine the value of the next node, the available data are evaluated and stored as data files. The filed information can be visualized graphically using standard visualization programs such as GNUPLOT,

Figure 7. (page 8 and 9) *Evolution of the approximation performed by AdaptivSearch to the example function no. 1; left side: 3D plot;*
right side: triangulation of the actual approximation state.

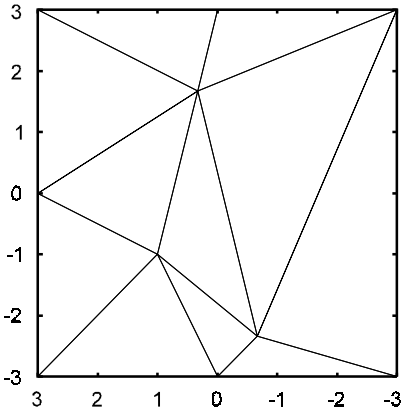


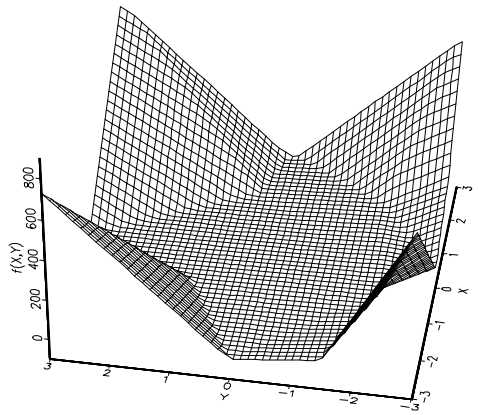
8 points



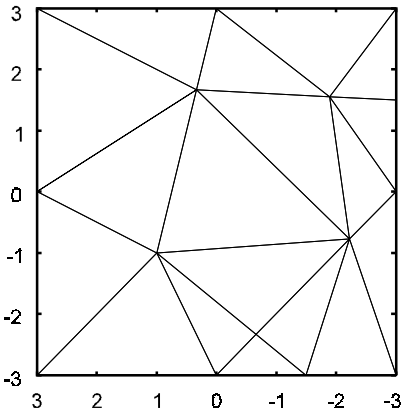


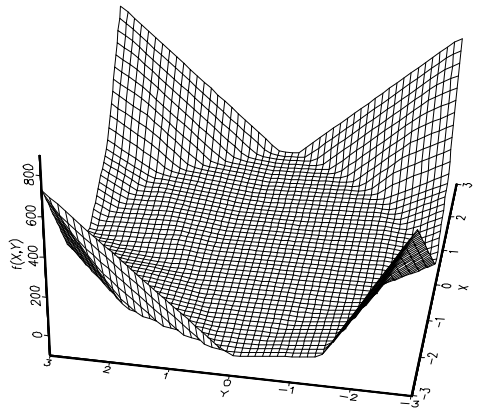
10 points



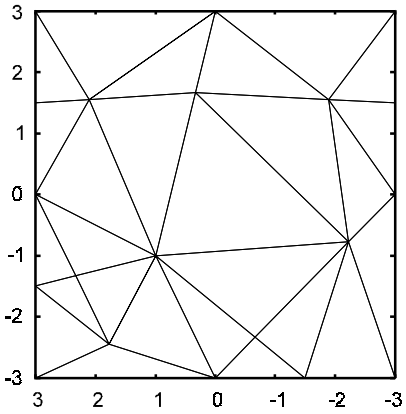


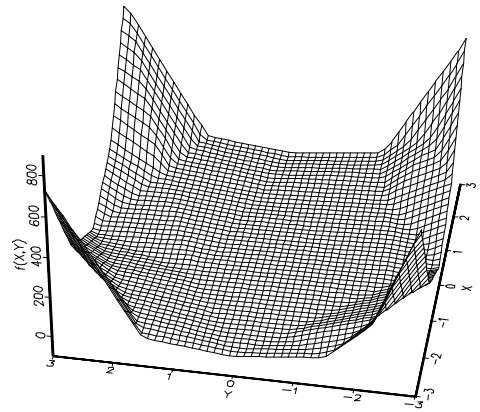
15 points



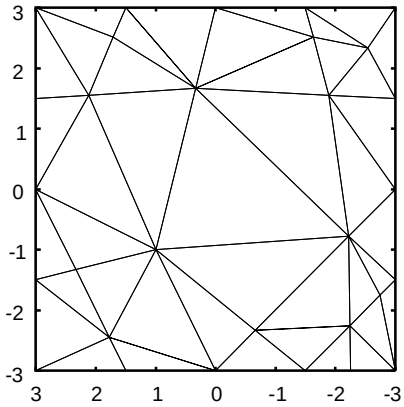


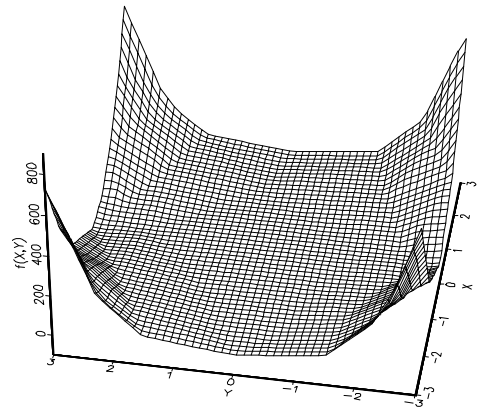
20 points



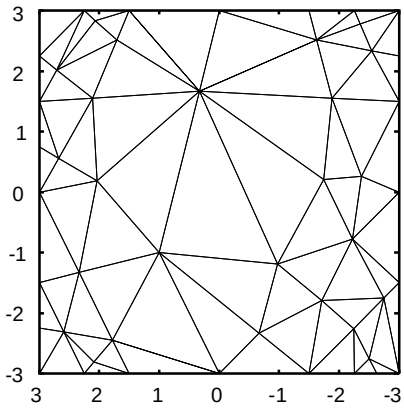


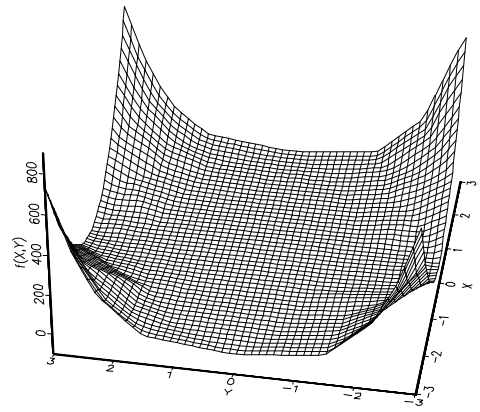
30 points



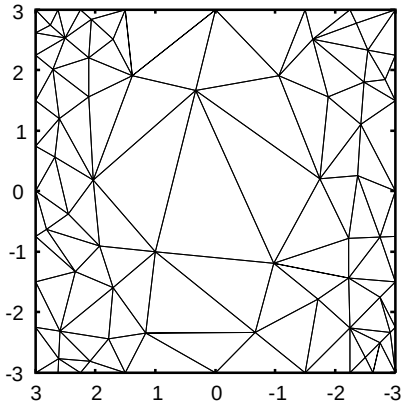


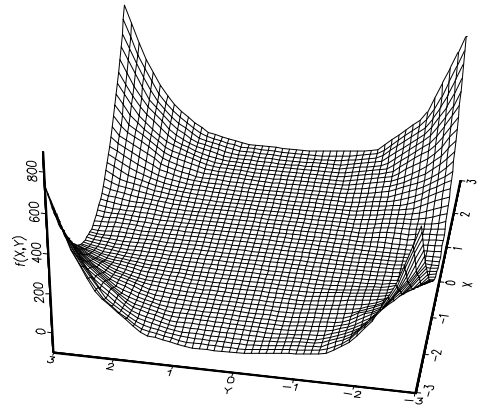
50 points



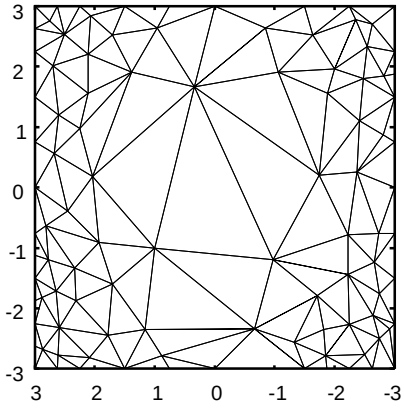


75 points





100 points



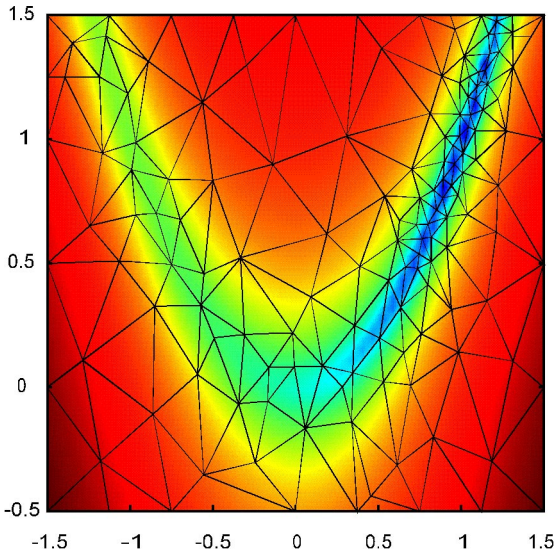


Figure 8. *A colored contour plot of the Rosenbruck function (example no. 2) and the AdaptivSearch triangulation after 200 calculated nodes.*

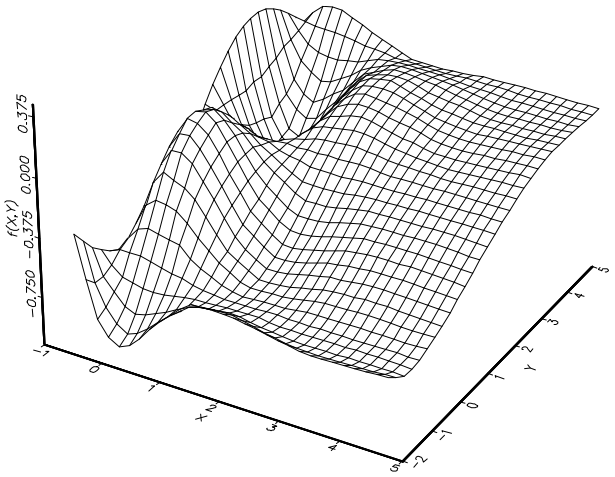


Figure 10a. *An arbitrary chemical model function (example no. 4)*

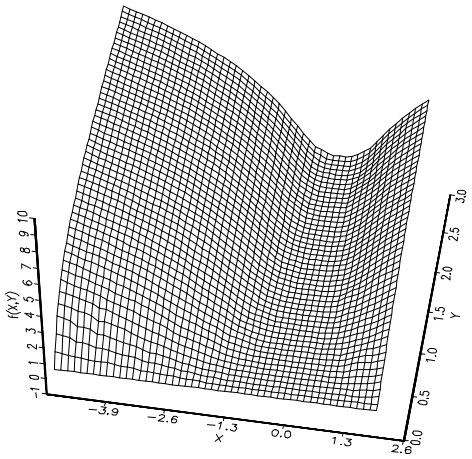
AXUM, or NCSA–Collage, so that the user can assess of the actual state at any point of the process.

The shape of the function corresponds to that of a four–pointed crown. Due to the even exponents of the parameters *x* and *y*, the function is symmetric both relative to the *xz* and to the *yz* plane, but not relative to the bisectors of the quadrants. When dissecting the function to the axes in parallel, one obtains parabolic curves, which, because of the y^4 term, are more strongly curved and become steeper than those that run in the *x* direction.

Figure 7 (left side) illustrates the development of the *AdaptivSearch* approximation to the function based on the nodes already calculated or measured. Figure 7 (right side) shows the triangulation at the corresponding state of the approximation. After a small number of measured nodes (20-25), the gross morphology of the surface is evident, the subsequent iteration steps only further "polish" the functional graph.

From the triangulation pattern of the last approximation step it becomes evident which partial areas have required

Figure 10b. (next page) *Evolution of the approximation to the arbitrarily constructed chemical model function (example no. 4).*



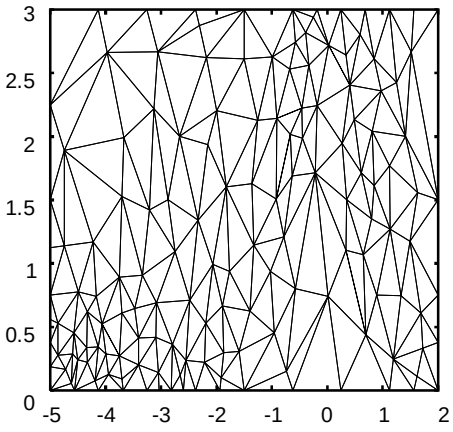
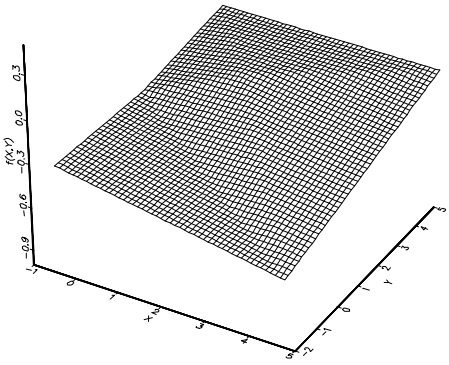
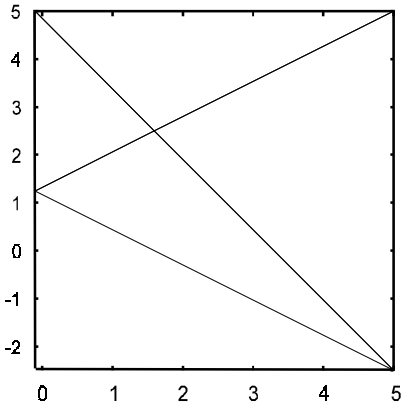
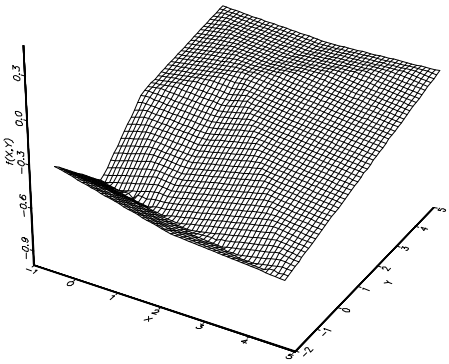


Figure 9. *Left: The "genuine" example function no. 3; right: The triangulation of the definition area at the stage of 200 points.*

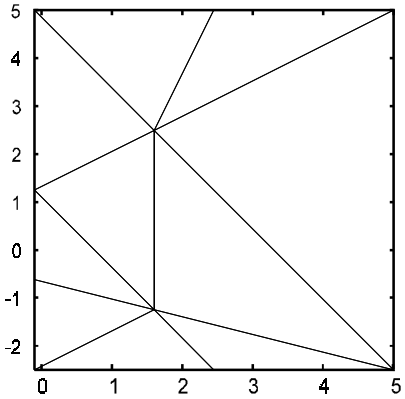


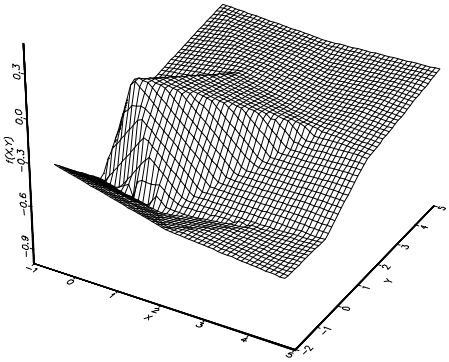
6 points



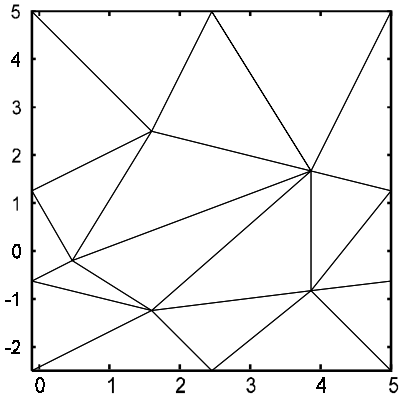


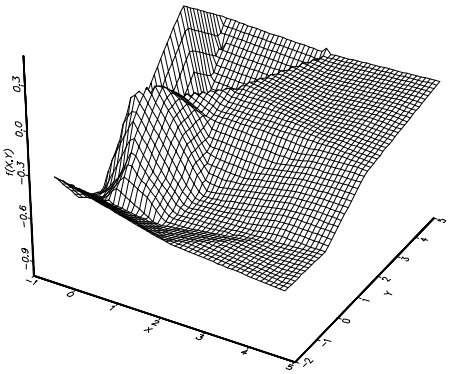
10 points



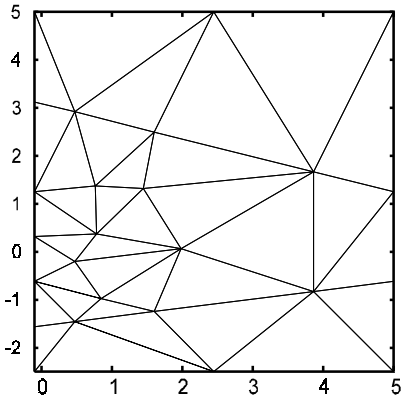


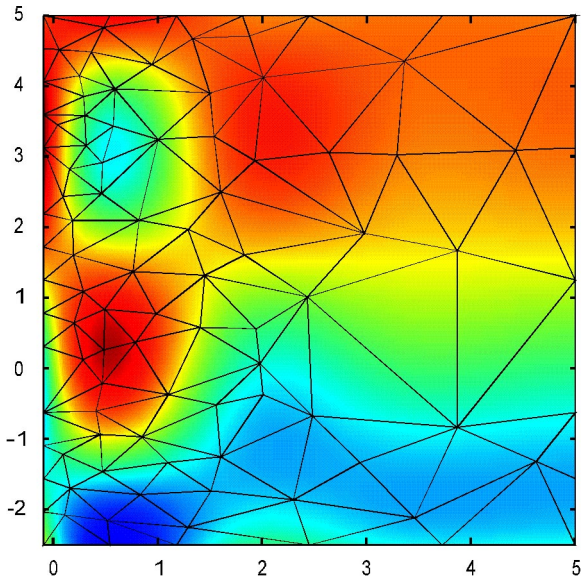
15 points





25 points





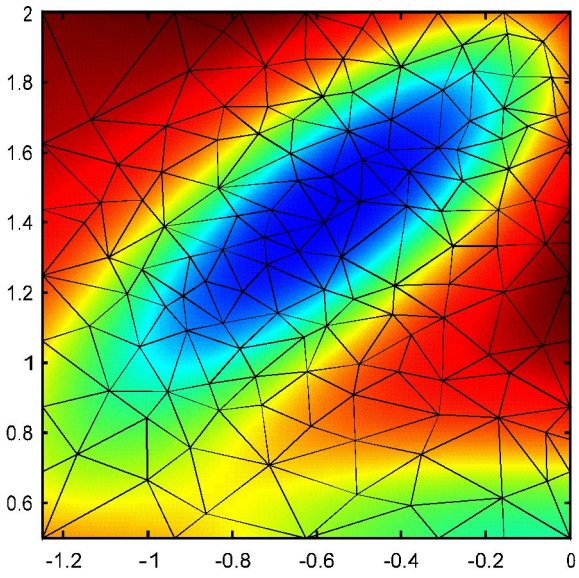


Figure 10c. *Triangulation of the arbitrarily constructed model function (example no. 4) on the level of 100 calculated nodes.*

the most intensive work. This discretization is particularly detailed at the edges of the definition range since here the function values grow rapidly because of the square and fourth power dependence of the variables. In contrast, *AdaptivSearch* does not refine the region around $x \approx y \approx 0$ intensively, because the function is relatively constant in this area (or, more humanly spoken, "boring") compared with the rest of the definition area.

AdaptivSearch recognizes the symmetry of the function (Figure 7, last picture), although this is not explicitely implemented in the algorithm. Besides the high density of triangles of those parts of the hypersurface that correspond to the points of the crown, the triangulation of the intermediate areas along the border lines is also interesting. It is clearly to be seen that, because of the different curvatures of the dissected parabolic curves mentioned above, the algorithm indeed triangulates with a higher density in the *y* direction than in the *x* direction (last picture in Figure 7).

2. $f(x; y) = \ln((1 - x)^2 + 100 \cdot (y - x^2)^2 + 0.001)$
with $x \in \{-1{:}5; 1{:}5\}$, $y \in \{-0{:}5; 1{:}5\}$

also known as the Rosenbruck function [16]:

Morphologically conspicuous is the ditch, which like a crescent–shaped riverbed runs through the definition area. Whereas this ditch initially (*x* negative, *y* positive) is still relatively broad and shallow, it gets narrow at its end (*x* positive, *y* positive) and becomes distinctly deeper. In this region the function starts to vary strongly from node to node: the ground of the riverbed is no longer homogeneous, but partially very deep craters are found to occur, which is clearly to be seen

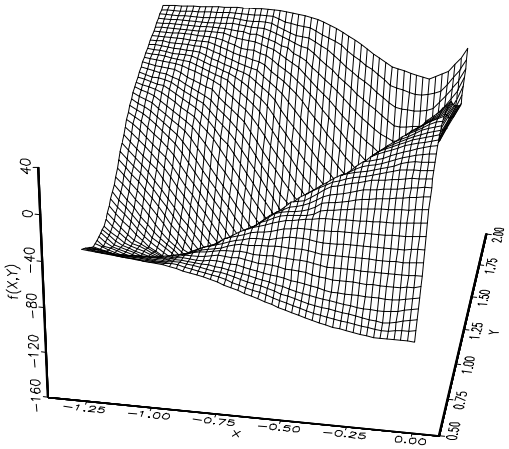


Figure 11. *Example function no. 5: The Müller-Brown surface and the triangulation at a stage of 200 calculated nodes.*

from the dark blue spots in the colored contour plots (Figure 8).

This function is a hard and thus ideal test case for any algorithm that has particularly been trained to reveal all the characteristic features of a hypersurface and to adapt its advancing to the unknown function. Although the calculation started by giving *AdaptivSearch* just the four points that limit the definition area, the algorithm rapidly and unbiasedly traces up the ditch and its fine contouration. The triangulation shown in Figure 8 represents the state of refinement after 200 measured nodes. The efficiency of the *AdaptivSearch* algorithm is convincingly demonstrated by the fact that the triangulation fully reflects the apparent description of the functional graph. Furthermore, even in the numerically critical fields (dark blue areas in Figure 8), the

algorithm does not loose its stability. Even for a higher number of calculated nodes (not shown), *AdaptivSearch* "keeps in mind" and investigates the entire definition area and does not get stuck in partial areas.

3. $f(x; y) = \ln(x^4 \cdot y^2 + 2)$
with $x \in \{-5{:}0; 2{:}0\}$, $y \in \{0.0; 3{:}0\}$.

The series of purely analytical functions is closed by a hypersurface in which the strongly curved and the planar parts are clearly separated from each other. This is very nicely elaborated by *AdaptivSearch*. The resulting surface and its triangulation are depicted in Figure 9.

Chemical model surfaces

4. A model surface arbitrarily constructed for a likewise imaginary chemical reaction system:

Even nowadays, experience and chemical intuition play an important role in the computer chemical analysis of chemical processes. Still, by just relying on one's intuition, one runs the risk of overlooking early interactions that may lead to relevant reaction pathways. In order to avoid this, one sometimes chooses the area to be probed larger than experience would suggest. Thus, one has to accept the fact that possibly also those areas are expensively investigated that are not actually relevant for the reaction considered. Our arbitrarily constructed model surface represents such a case for which, due to this precaution, the definition area was chosen too large (Figure 10a). For this hypersurface, the mathematical equation reads:

$$E = \exp(-x)(\sin 2x \cdot \cos y + 1) - 0.025\,(y - 4)^2 + 0.05 \exp(y)$$

Figure 10b shows the way how *AdaptivSearch* explores the surface. After 25 nodes already, all essential characteristics are apparent: 2 maxima, 2 minima, and 3 saddle points, *i.e.* transition states. Hence, by using *AdaptivSearch*, one can recognize very early which parts of the area are of interest for the reaction.

At this point a scanning process as performed using GridSearch would normally be stopped. Instead, a new search would be started within a strongly reduced definition area excluding the planar ("boring") regions.

By contrast, there is no need for a *AdaptivSearch* run to be interrupted since selectively only those partial areas are further elaborated in which chemical interactions cause the strongest contour changes, thus hinting at chemically interesting parts of the hypersurface. Unexpected, but possibly nonetheless existing energetic interactions in more planar, "calm" areas would still not get lost and would likewise be evaluated.

Figure 10c shows the triangulation after 100 nodes, the hypersurface is portrayed as a colored contour plot.

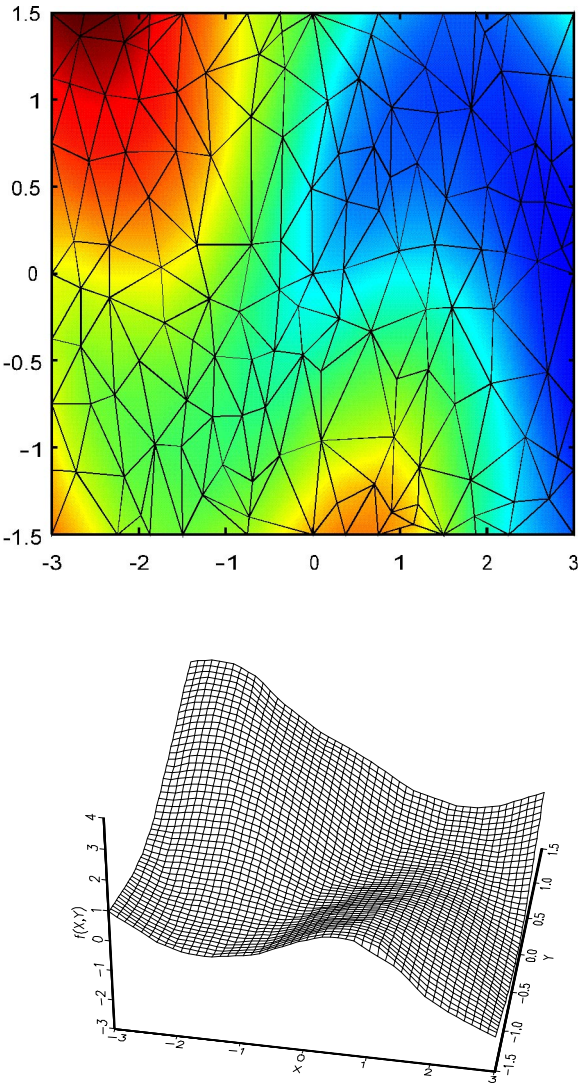


Figure 12. *Example function no. 6: The Schlegel-Gonzales surface and its triangulation (200 calculated nodes).*

5./6. Model surfaces containing one reaction path:

The model hypersurfaces described in the literature are normally used as test hypersurfaces for novel reaction path following algorithms. The definition areas of the functions are adjusted such that only the essential feature, namely the chemical reaction pathway, is mathematically modelled. For our *AdaptivSearch* investigations, we have chosen two standard representatives: the Müller–Brown surface (example no. 5) [17] and the Gonzales–Schlegel surface (example no. 6) [18]. Figures 11 and 12 show the results of the *AdaptivSearch* directed investigation after 200 calculated nodes.

As clearly to be seen, *AdaptivSearch* is capable of elaborating reaction pathways with particularly high accuracy. The slopes that parallel the reaction pathways have more constant gradients than the valley bottom and thus can be approximated linearly by larger triangles.

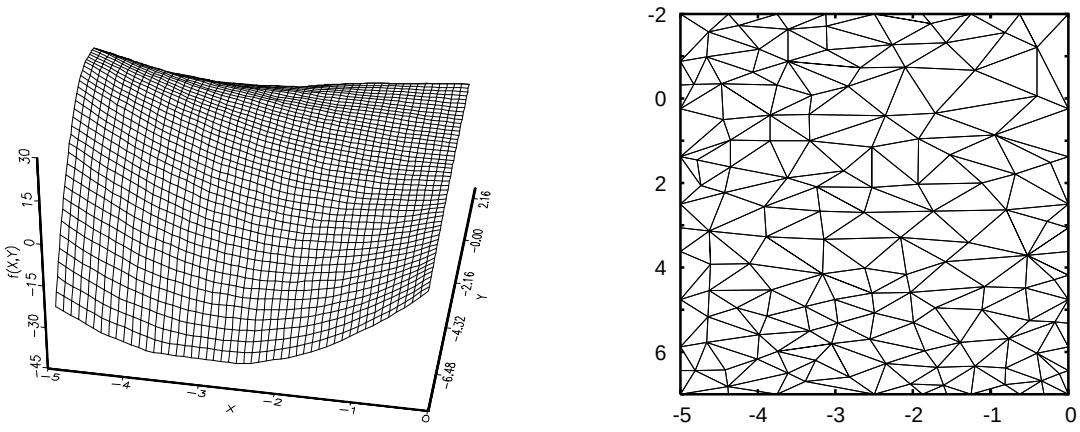


Figure 13. *The Ruedenberg function (example no. 4): The 3D plot of the model surface and its triangulation after 150 nodes.*

7. The Ruedenberg function [19]:

Like the Müller–Brown and the Schlegel–Gonzales surfaces, the Ruedenberg function is also an established model function for testing novel reaction path finders. The equation for the hypersurface reads:

$$E = ½ (xy^2 y^2 x + x^2 + 2y - 3)$$

In this polynomial function, the highest powers are squares. Since parabolic curves always have a constant 2nd derivative, they have uniform curvature. It is small wonder that the triangulation of the definition area is very homogeneous except for the margins of the upper right quadrant.

One is nearly tempted to consider GridSearch as a special case of *AdaptivSearch* for calm ("boring") hypersurfaces. Indeed, *AdaptivSearch*, with its remarkable property to approach inhomogeneous areas selectively, cannot display this advantage for such relatively homogeneous hypersurfaces. In the Ruedenberg function all partial areas are of similar interest for the algorithm. In this particular case, the use of GridSearch would also be justified. On the other hand, it is the characteristic strength of *AdaptivSearch* to elaborate that it is indeed a uniform function – information that is avalaible only by *AdaptivSearch*.

Summary and Outlook

We have shown *AdaptivSearch* to be a procedure that is capable of probing two–dimensional hypersurfaces of different types. *AdaptivSearch* delivers reliable and exact results – regardless whether the investigated problem is homogeneous, *i.e.* exhibiting an uniform course of the function, or very inhomogeneous, *i.e.* consisting of differently curved regions. Especially for the latter cases, which occur mostly when investigating potential energy surfaces of chemical reactions, *AdaptivSearch* predominantly probes those regions in which the curvature changes strongly. Due to its iterative execution scheme, it is possible for *AdaptivSearch* to inform the user about the details of each refinement step: During the investigation, the internal average error and the distribution of the local errors on the definition area are quantified and can be visualized continuously. *AdaptivSearch* can be stopped at any time of the scanning process, and may be easily restarted from the last point of interruption without loss of data.

Thus, the various demands and expectations of computer chemists towards chemically relevant hypersurfaces are ideally fulfilled by the *AdaptivSearch* algorithm.

• One obtains the gross topology of the hypersurface after only a few calculated nodes.

• Additionally, one can refine a hypersurface further – even a hypersurface that has been obtained by conventional GridSearch! – up to a defined accuracy with a minimum number of nodes.

In computational chemistry, fields of application for *AdaptivSearch* will mainly be the probing of energy potential surfaces, conformational analyses, and the calculation of electrostatic potential molecular surfaces (EPM), as they are widely required, *e.g.* in pharmaceutical research for the determination of the interactions between a substrate and the active site of an enzyme.

AdaptivSearch has already been realized for two-dimensional cases, the implementation of treating three-dimensional problems is drawing to a close. All program routines are written in C using standard components. Thus, the program will easily be ported to other UNIX platforms. Stable und tested program versions exist under IRIX5.2 (Silicon Graphics) and LinuX (UNIX similar operating system for IBM compatibles distributed under the GNU Public License, copyright by Linus Thorvalds).

Basically, *AdaptivSearch* was designed as a "black box" algorithm that is entirely independent from the corresponding measuring program. At this time, we have implemented several interfaces to the most frequently used quantum chemical program packages such as VAMP5.5 [20], MOPAC6.0 [2], GAMESS [21], and GAUSSIAN92 [22].

Acknowledgements This work was financially supported by the Deutsche Forschungsgemeinschaft (Sonderforschungsbereich No. 347 "Selective Reaction of Metal-activated molecules") and the Fonds der Deutschen Chemischen Industrie. We are especially indebted to U. Küster (Rechenzentrum, Universität–Stuttgart) for the initial impetus and T. Clark (Computational Chemistry Center, Universität Erlangen) for their kind help and fruitful discussions. Furthermore, we wish to thank M. Stahl, U. Dauer, and R. Stowasser for valuable suggestions and technical assistance.

References

1. In the paper, nodes are defined here as calculated or measured values at a discrete point on the n–dimensional surface being considered.
2. GridSearch as implemented in SYBYL: Tripos Associates, 1699 St. Hanley Road, Suite 303, St. Louis, MO, 63144; or MOPAC 6.0: Stewart, J.J.P., QCPE 455, 1990.
3. Meyer R. *J. Mol. Spec.* **1979**, *76*, 266.
4. CAS–Online search from May 1995 (Keywords: hypersurf*, algorithm) yielded 177 hits. None of the citations found deals with an adaptive approach for the construction of hypersurfaces.
5. Joe, B. *SIAM J. Sci. Stat. Comput.* **1986**, *7*, 514.
6. Braess, D. *Finite Elemente*, Springer-Verlag, Heidelberg, 1992.
7. Hirsch, C. *Numerical Computation of Internal and External Flows – Volume 1: Fundamentals of Numerical Discretization*, Wiley & Sons, New York, 1991.
8. Axelsson, O.; Barker, V.A. *Finite Element Solution of Boundary Value Problems*, Academic Press, New York, 1984.
9. Linderberg, J.; Padkjaer, S.B.; Öhrn, Y.; Vessal, B. *J. Chem. Phys* **1989**, *90*, 6254.
10. Deuflhard, P.; Leinen, P.; Yserentant, H. *IMPACT of Computing in Science and Engineering* **1989**, *1*,3.
11. Joe, B. *SIAM J. Sci. Stat. Comput.* **1993**, *14*, 1415.
12. Bringmann, G.; Güssregen, S.; Vitt, D.; Gulden, K.–P. unpublished results.
13. Hutter, M.; Clark, T. personal communication.
14. NAG Fortran Library, Introductory Guide, Mark 16, Chapters E01/E02, Oxford, 1993; Handbook of Mathematical functions, pages 878–913, Dover Publications, Inc., New York, 1965.
15. Boor, C. de *A Practical Guide to Splines* Springer-Verlag, New York, 1978.
16. Gnuplot 3.5: Williams, T.; Kelley, C.; A 3D graphic program distributed under GNU Public License, 1986–1994.
17. Müller, K.; Brown, L.D. *Theor. Chim. Acta* **1953**, *53*, 75.
18. Gonzalez, C.; Schlegel, H.B. *J. Chem. Phys.* **1991**, *95*, 5853.
19. Nord, R.S.; Ruedenberg, K. *Theor. Chim. Acta* **1969**, *69*, 265.
20. Rauhut, G.; Chandrasekhar, J.; Alex, A.; Beck, B.; Sauer, W.; Clark, T.; VAMP 5.5, available from Oxford Molecular Limited, The Magdalen Centre, Oxford Science Park, Sandford on Thames, Oxford OX4 4GA, England.
21. Dupuis, M.; Spangler, D.; Wendoloski, J.J.; GAMESS, National Resource for Compu-tations in Chemistry, Software Catalog, University of California, Berkeley, CA, USA, **1980**, Program QG01; Schmidt, M.W.; Baldridge, K.K.; Boatz, J.A.; Jensen, J.H.; Ko-seki, S.; Gordon, M.S.; Nguyen, K.A.; Windus, T.L.; Elbert, S.T.; *QCPE Bulletin* **1990**, *10*, 52.
22. Frisch, M.J.; Trucks, G.W.; Head-Gordon, M.; Gill, P.M.W.; Wong, M.W.; Foresman, J.B.; Johnson, B.G.; Schlegel, H.B.; Robb, M.A.; Replogle, E.S.; Gomperts, R.; Andres, J.L.; Raghavachari, K.; Binkley, J.S.; Gonzalez, C.; Martin, R.L.; Fox, D.J.; Defrees, D.J.; Baker, J.; Martin, R.; Kahn, L.R.; Stewart, J.J.P.; Pople, J.A. GAUSSIAN 92; Revision C, Gaussian, Inc., Pittsburgh PA, 1992.