Approximate Query Answering In Numerical Databases

Nabil Hachem, Chenye Bao, and Stephen Taylor
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609, USA
email: {hachem,staylor}@cs.wpi.edu
phone: (508) 831-{5409,5669,5449}
Fax: (508) 831-5776

Abstract

Scientific databases are usually large, distributed and dynamically changing. We address the problem of efficient processing of queries in scientific databases, especially in very large numerical databases. Previous work has focused on how to store the database and the design of index structures for the efficient access of data. Recently more and more statistical methods have been used in query optimization. Those methods essentially attempt to approximate the distribution of the attribute values in order to estimate the selectivity of query results.

We introduce a new methodology that uses regression techniques to approximate the actual attribute values. Through analysis of the data, one derives a set of characteristic functions to form a "regression database," a compressed image of the original database. Based on these functions, approximate answers to queries may be provided within a pre-specified tolerable error, but without the expensive search overhead usually inherent with the use of indexing techniques.

We propose a framework to build regression databases. An experimental prototype is implemented to evaluate the technique in terms of realizability, efficiency and practicality. The results demonstrate that our approach is complementary to conventional approaches and to statistical methods.

Contents

1 Introduction **1**

2 Regression Analysis and Curve Fitting **1**

3 Framework for Approximate Query Answering **4**

3.1 Framework 4

3.2 Issues 4

4 Implementation **5**

4.1 Algorithm-A: Naive Segmentation 6

4.2 Algorithm-B: Delayed Segmentation 6

4.3 Problems with LSR 7

4.4 Algorithm-C: Relaxation LSR 7

4.5 Query Processing 9

5 Analysis and Evaluation **9**

5.1 Test Data Sets and Evaluation Criteria 9

5.2 Tests with the synthetic database 10

5.3 Tests with real databases 11

5.4 Comparisons with other approaches 15

6 Conclusions and Future Work **17**

1 Introduction

Scientific databases are usually large, distributed and dynamically changing. The efficient processing of queries in database systems, and specifically in very large numerical databases is an important research problem [8, 13, 16, 17, 28]. When the precise attribute values are not significant in answering a query, using an approximate representation can reduce the storage space for loading the database and the time for searching the large amount of information. Through a good choice of representation, a class of queries including aggregate queries can be supported. We propose to use regression techniques to approximate the actual attribute values of the data.

Regression analysis is a statistical technique for investigating multidimensional/multivariate data. It provides a conceptually simple method for establishing a functional relationships among variables [2]. We applied Least Squares Regression, which minimizes the sum of squares of differences between the observed values and the corresponding approximate values, to compute the set of coefficients of the fitting function. We refer to databases that use regression functions to model numerical data as "regression databases." Queries on actual data can be mapped to queries on regression databases. Query answers, evaluated against the regression database, are an approximation of the exact answers against the actual data.

The contributions of this work include the development of a framework to build regression databases for approximate query answering: first we analyze the data to choose a function that best fits it and a regression technique to compute the set of coefficients of the fitting function. Second we derive the coefficients from the data; third we maintain the regression database using adaptive techniques. Different maintenance algorithms, that deal with different fitting functions and criteria, were developed. An experimental prototype was implemented to evaluate the approach in terms of realizability, and efficiency.

Section 2 introduces regression analysis and curve fitting. A general framework to build regression databases is presented in Section 3, while the details of algorithms which were developed and implemented are described in Section 4. An evaluation of these algorithms is then presented in Section 5. Finally Section 6 concludes this work and gives a direction for future research.

2 Regression Analysis and Curve Fitting

We overview linear regression models in regression analysis. The basic form of that model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + e$$

where $X_1, X_2, \cdots, X_p$ are *independent variables* or *explanatory variables*, and Y is the *dependent variable* or *response variable*. $\beta_0, \beta_1, \beta_2, \cdots, \beta_p$ are called the *regression coefficients* or *regression parameters* and determined from the data. The error e is assumed to be a random disturbance with mean 0 and a common variance denoted by σ^2.

When we say that a model is linear or nonlinear, we are referring to linearity or nonlinearity *in the parameters* [6]. The value of the highest power of a predictor variable in the model is the *order* of the model. For example,

$$Y = \beta_{10} + \beta_{11} X + \beta_{12} X^2 + e$$

is a second-order (in X) linear (in the β's) regression model.

X_i can be X_i^2, $1/X_i$, $X_i X_j$, X_i/X_j, or $\log X_i$, $\sin X_i$, X_i', or any other transforms of X_i, or combination of any of these transforms. These forms can be chosen into a regression model as the basis fitting function. Additional details can be found in [1, 2, 6, 14].

Regression Techniques	Domain of Optimality
Least Squares	when error distribution is exactly normal
Least-Absolute-Deviations	when error distribution has heavy tails and is effective at controlling bias
Huber M-estimate	when error distribution has heavy tails
Nonparametric	when error distribution has heavy tails
Bayesian	prior information are taken into account in constructing an estimate and produce direct probability statements about the parameters
Ridge	when there are collinearities among the independent variables

Table 1: Classification of regression techniques

The objective is to estimate the regression coefficients from the observed data so that the regression model best represents the set of observations. There are various regression methods that correspond to different definitions of what "best represents" stands for. Table 1 classifies some of the commonly used regression techniques and briefly describes under which conditions they are optimal. We discuss the Least Squares regression (LSR) method. Additional details on the other methods is found in [1, 4].

Consider a one-dimensional data set with n observations and p independent variables. Let $Y = X\beta + e$ where

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & & & \cdots & \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}.$$

The LS method minimizes the squares of the residuals, that is, minimizes $\sum_{i=1}^{n} e^2$. The LS estimate is the best linear unbiased estimate [1]. The disadvantage is that it assumes that the random errors are normally distributed and allows "outlying" data points to influence the final determination of the regression function parameters.

Before using regression techniques, we need to choose a set of basis functions that may fit the set of observations. The problem to be addressed relates to which functions and regression techniques to choose. The data distribution and the error distribution need to be analyzed. For example, if the data set has periods one may use trigonometric functions. If the error population is assumed to be normally distributed, Least Squares Regression may be used. In this work, we focused on two classes of basis functions: polynomial interpolation and Fourier Series.

Polynomial Interpolation

The most elementary notion of curve fitting is that of interpolation [14, 26]. The goal of interpolation is to derive coefficients of a polynomial function that exactly fits a given set of data points. For regression databases, we are interested in the case where there are more conditions to be satisfied than parameters to be adjusted. For example, we may be given

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6,$$

which has seven coefficients, and we are required to find the set of coefficients which best approximates a set of $i >> 7$ observations (x_i, f_i) (f_i denotes $f(x_i)$). we need to solve the matrix equation $XA + E = F$ where:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 & x_1^6 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & x_2^5 & x_2^6 \\ \vdots & & & & & & \\ 1 & x_n & x_n^2 & x_n^3 & x_n^4 & x_n^5 & x_n^6 \end{bmatrix} A = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_6 \end{bmatrix} E = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} F = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}.$$

It is well known that $A = (X^TX)^{-1}X^TF$ gives the coefficients that minimize the sum of the squares of differences between the observations and the approximations [24].
That is, with LSR, $\sum_{i=1}^n (\tilde{f}_i - f_i)^2 = \sum_{i=1}^n e_i^2$ is minimized, where

$$\tilde{f}_i(x) = \tilde{f}(x_i) = a_0 + a_1x_i + a_2x_i^2 + a_3x_i^3 + a_4x_i^4 + a_5x_i^5 + a_6x_i^6$$

and (x_i, f_i) are the observations, $i = 1, \cdots, n$.

Fourier Series

Polynomial interpolation techniques are not always satisfactory. They usually suffer from rounding error propagation problems. Another set of basis functions can be derived with Fourier Series of the form

$$F(x) = \frac{a_0}{2} + a_1 \cos nx + b_1 \sin nx + a_2 \cos 2nx + b_2 \sin 2nx$$
$$+a_2 \cos 3nx + b_2 \sin 3nx + \cdots$$

where $n = \frac{2\pi}{L}$ and L is the interval length $(0 \leq x \leq L)$. We have $XA \approx F$, with

$$X = \begin{bmatrix} 1 & \cos nx_1 & \sin nx_1 & \cos 2nx_1 & \sin 2nx_1 & \cos 3nx_1 & \sin 3nx_1 & \cdots \\ 1 & \cos nx_2 & \sin nx_2 & \cos 2nx_2 & \sin 2nx_2 & \cos 3nx_2 & \sin 3nx_2 & \cdots \\ \vdots & & & & & & & \\ 1 & \cos nx_m & \sin nx_m & \cos 2nx_m & \sin 2nx_m & \cos 3nx_m & \sin 3nx_m & \cdots \end{bmatrix},$$

$$A = \begin{bmatrix} a_0 \\ a_1 \\ b_1 \\ a_2 \\ b_2 \\ a_3 \\ b_3 \\ \vdots \end{bmatrix}, F = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix},$$

We still use LSR, with $A = (X^TX)^{-1}X^TF$.

Increasing the number of terms of the Fourier series does not suffer from the rounding error problem as much as polynomials. But simply increasing the number of terms will need additional storage space and may not improve the performance much.

3 Framework for Approximate Query Answering

In this section, we propose a general framework to build regression databases and discuss some issues associated with the framework.

3.1 Framework

Given a multidimensional/multivariate data set, the general framework for building and using regression databases includes three main steps:

1. **Data Analysis** which can be viewed as model construction. It is used to determine the characteristic of the data distribution. Within this step, a class of curve fitting functions as well as the regression technique that may be used to fit the data set are determined and selected.

2. **Regression Function Generation**. Based on the chosen regression technique and curve fitting methods, the set of coefficients such that the fitting function best fits the data set are computed. In our work we chose LSR coupled with the class of polynomial interpolation or Fourier Series as basis functions. We also propose and test a variation of LSR which is referred to as "relaxation" LSR.

   One must then check if the regression function "best" represents the data set. If the ***relative error*** between the actual value and the corresponding function value is within ε, the function thus derived is considered as fitting the data points.

3. **Regression Function Maintenance**. To maintain the regression database, space hypercubes are used as bounding boxes for data sets. A fitting function is associated with the region covered by a bounding hypercube. In response to an update, the objective is to adaptively modify the regression function associated with the bounding box which is the target of the update.

   For example, after a new data point is added to a region, one needs to determine the suitability of the regression function associated with that region. If the ***relative error*** between the new data point and the corresponding function value is greater than a prespecified ***tolerable error*** ε, a new function needs to be derived (Step 2). From time to time, recomputing a new function that satisfies ε may not be possible. In this case, the region is split into two subregions, by dividing the bounding hypercube along one dimension. Then a fitting function is derived for each region (Step 2). Optionally, one can just derive a new function for the region containing the new data point, and use the previous one for the other region. In some situations, it may be possible to switch to another class of basis fitting functions and regression technique, which implies restarting from Step 1.

3.2 Issues

There are some design and research issues associated with each step:

1. **Data Analysis**

   It is often the case that a single function is not sufficient to approximate the data set within ε. In many cases, the data set needs to be divided into several segments so that each segment can be accurately approximated.

   Furthermore for a set of data segments, different classes of basis functions and regression techniques may need to be associated with one or more segments. The choice of a suitable function depends on the data distribution of the segment. For example, if the segment has periods we may use trigonometric functions; if the segment has poles, we may use rational functions.

For the work reported here, we have analyzed the use of Fourier series and polynomials only. Although the appropriate regression technique depends on the nature of the data, and should be chosen dynamically, we used the LSR technique. Further investigations on this subject will be necessary.

Finally, the problem associated with the selection of a suitable data structure to hold the bounding boxes and associated fitting functions is within the field of study of indexing techniques, and out of our current scope. One can use current available access index structures.

2. **Regression Function Generation:**

   For this step, issues associated with efficiency, computation and storage overhead are considered.

   Using all data points in a segment to compute the coefficients may be slow. To speed up the computation, one might just want to use a subset of representative data points to compute the coefficients without losing much accuracy. Sampling techniques have been proposed for that purpose [5, 10, 11, 20, 21].

   The space required to store the computed matrix depends on the number of data points "n" as well as the number of coefficients of the fitting function. Increasing the number of coefficients may increase the accuracy of the fitting function, at the expense of an increase in storage overhead. Assume f_0 is the mean size of segments and d is the dimension of the data set. To store the whole segment we need $8f_0(d+1)$ bytes, assuming 8 bytes per point. For each segment, we need to record the regression technique and fitting function types (1 byte each), as well as the coefficients c_d of the function (8 bytes each)[1] . Therefore the ratio of the size of the regression database to the size of the original numerical database, referred to here as ***compression ratio***, is expressed as

$$\frac{8f_0(d+1)}{2+8c_d} \approx \frac{f_0(d+1)}{c_d}.$$

3. **Regression Function Maintenance**.

   Issues associated with this step include update propagation, split criteria and error.

   Immediate or delayed updates may be adopted. Two of the algorithms proposed and studied in this work use delayed update propagation with the use of overflow arrays. Whenever a data segment needs to be split into two new segments, different split criteria can be followed. For example an ***equal density*** criterion would result into each segment having the same number of data points, while an ***equal range*** criterion results into segments that cover equal absolute ranges of the dimension of the split. Another technique would be to chose the boundary between segments as the value of the newly inserted data point along the split. ***Relative error*** and ***absolute error*** both have advantages and disadvantages. Which should be chosen depends on the applications [4].

4 Implementation

We implemented three different algorithms within the framework. The implementation and evaluation parameters for each algorithm are tabulated in Table 2. The algorithms were tested using a 2-dimensional

[1]This estimate is pessimistic, as usually 8 bytes double-precision representations may not be needed. Furthermore, the storage of a point should include the X and Y values for a 2-dimensional data set.

Algorithm	A: Naive A: Segmentation	B: Delayed B: Segmentation	C: Relaxation C: LSR			
Regression technique	Least Square Regression		Relaxed LSR			
Basis function	Polynomial interpolation				Fourier series	
ε	relative error				relative error	absolute error
Split criterion	equal density					
Test databases	synthetic	synthetic	synthetic	solar	solar	temperature

Table 2: Parameter sets for the three algorithms

synthetic database and two real databases (refer to Section 5). Each algorithm has two main parts that are briefly described: a pre-processing module and an update module. We also discuss how queries are processed.

4.1 Algorithm-A: Naive Segmentation

This is a basic naive algorithm that essentially keeps splitting the data into smaller segments until each individual data set is properly fitted. The two modules as as follows:

1. Pre-processing
   (a) Start with one bounding box;
   (b) Fit the original data with a set of coefficients;
   (c) For each bounding box
   { While there exists a point whose *relative error* is greater than ε, split the data set into two bounding boxes;
   for each bounding box repeat from step (b);}
   (d) Store the sets of coefficients with their corresponding bounding boxes and return;
2. Adding a new data element
   (a) Find the bounding box covering the new data element;
   (b) Using the corresponding fitting function, compute the approximate value of the new point;
   (c) If the *relative error* of the estimate to the true value is not less than ε
   {split the bounding box into two regions;
   for each bounding box, call the pre-processing module; }

4.2 Algorithm-B: Delayed Segmentation

This algorithm follows a "delayed commitment" strategy. The idea is to associate an "overflow" array with each bounding box. This structure can contain at most N points and is used to delay splitting or merging the data segments and thus recomputing the coefficients of fitting functions. The two modules as as follows:

1. Pre-processing
   (a) Start with one bounding box;
   (b) Fit the original data with a set of coefficients;
   (c) For each bounding box
   {While there exists a point whose ***relative error*** is greater than ε, store it in the corresponding overflow array;

      If the overflow structure is full, split the data set into two bounding boxes and independently restart from step (b);}
   (d) Store the coefficients with their corresponding bounding boxes and return;
2. Adding a new data element
   (a) Find the bounding box covering the new data element;
   (b) Using the corresponding fitting function, compute the approximate value of the new point;
   (c) If the ***relative error*** of the estimate to the true value is not less than ε
   {Store it in the corresponding overflow array. If the array is full,

      Split the bounding box into two bounding boxes;
      for each bounding box, call the pre-processing module;}

4.3 Problems with LSR

The main characteristic of LSR is that it advocates "egalitarianism" which makes "good" points and "bad" points all become "fair" points. In other words, it gives great emphasis to large errors, and little emphasis to small ones. For example, it prefers 10 errors of size 1 to one error of size 4 because $4^2 > \sum_{k=1}^{10} 1^2 = 10$.

Consequently, the presence of some "outlying" data points can distort the trend of the fitting curve. For example, using a synthetic database (Table 3), we observe that when using a 6th order polynomial fit and with ε=0.05, fitting the 1000 points in one bounding box results in only 518 "good" points whose relative errors are within 0.05, although there are 566 synthetically generated "good" points. This is because LSR advocates "egalitarianism" which makes "good" points and "bad" points all become "fair" points.

Therefore, to get a reasonable fitting curve which fits most data points, it is better to delete the "outlying" data points first, then fit the remaining data points. In Figure 1 the curve fitted by LSR is shown in solid line. If we delete the two "outlying" data points and fit the remaining data points, we get the more reasonable dashed line curve.

The objective is then to find the "best" function that can fit most data points within ε. In other words we want to minimize the the number of points in the overflow array. To solve this problem, we propose a heuristic with Algorithm-C which we refer to as the relaxation fit or relaxation LSR.

4.4 Algorithm-C: Relaxation LSR

This algorithm aims at fitting most data points by reducing the number of "outlying" points in the overflow array. This should result in an increase of the compression ratio.

If we delete all "bad" points at the same time, then many "good" points will also be deleted since they appear "bad" with respect to the wrong fitting curve. It is best to incrementally delete "bad" points, by deleting the "worst" points first. This technique is referred to as "relaxation LSR."

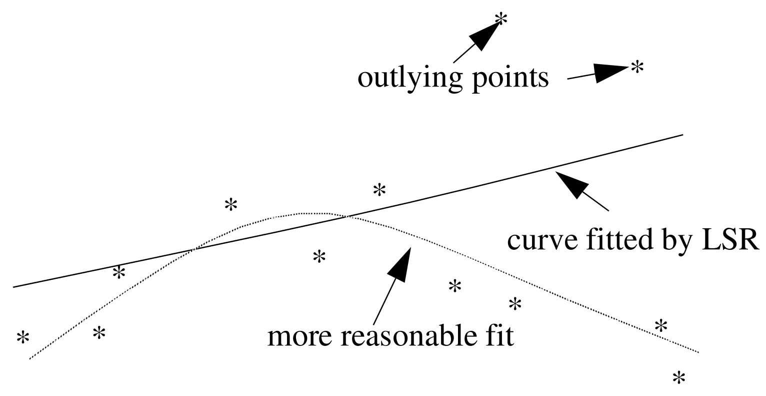

Figure 1: Effect of outliers on the goodness fit of LSR

The issue is to determine which points are "bad," which are "worse" and which are the "worst." This determines the width of ***error levels***. Our proposed algorithm is based on a heuristic observation: Because the "worse" the points are, the less the number of the points, we divide the "bad" points into different width levels. The largest error level is $\varepsilon + 10L$; the second largest one is $\varepsilon + 6L$; the third largest one is $\varepsilon + 3L$; the fourth largest one is $\varepsilon + L$; and the fifth largest one is ε. The widths between the two successive ***error levels*** are 4L, 3L, 2L, and L, respectively. One can change the number of levels in the algorithm. The more the number of the ***error levels***, the better the results; however, the longer the time it takes to iteratively fit the data. The two modules as as follows:

1. Pre-processing
   (a) Start with one bounding box;
   (b) Fit the original data with a set of coefficients;
   (c) If ***maximum relative error*** $\leq \varepsilon$, then store the set of coefficients and return; (This means there is no point whose relative error is larger than ε)
   (d) If ***maximum relative error*** $\leq i * \varepsilon, i = 2, 3, 4, 5, 6, 7, 8, 9, 10$, then $L = (i-1) * \varepsilon/10$; Else, $L = \varepsilon$.
      i. Divide the points into 5 levels with relative errors $\leq \varepsilon$, $\leq \varepsilon + L$, $\leq \varepsilon + 3L$, $\leq \varepsilon + 6L$ and $\leq \varepsilon + 10L$ respectively;
      ii. Iteratively delete the points with relative errors $> \varepsilon + 6L$, $> \varepsilon + 3L$ and $> \varepsilon + L$, and each time fit the remaining points with a set of coefficients;
      iii. Keep the set of coefficients which minimizes the number of points with relative errors $> \varepsilon$.
      iv. Store the points with relative errors $> \varepsilon$ in an overflow array;
2. Adding a new data element
   This module is similar to the one for Algorithm-B.

4.5 Query Processing

In the case of Algorithm A, a query is processed by applying it against the regression database returning an approximate answer. For Algorithms B and C, the overflow structures are used first. If the query can be answered then the answer is returned. If not, then the query is processed against the regression database, returning an approximate answer.

5 Analysis and Evaluation

5.1 Test Data Sets and Evaluation Criteria

The different algorithms were evaluated with 3 different data sets

Synthetic data set

The synthetic data set consists of 1,000 data points, generated from a 6-th order polynomial function perturbed by multiplying by a uniform random variable, keeping the average relative error within 5%.

The second row in Table 3 shows the distribution of perturbations of the synthetic data set. The third and fourth rows show the distribution of the relative errors of approximate values fitted with a 6-degree and an 8-degree polynomial respectively; with respect to the corresponding actual values.

	$\leq .05$	.05-.10	.10-.15	.15-.20	.20-.25	.25-.30	$> .30$
Synthetic	566	327	93	13	1	1	0
Degree=6	518	318	95	29	16	22	18
Degree=8	518	306	101	34	13	19	22

Table 3: Data points distribution as a function of the error level for the synthetic data set

Solar data set

The ERB-SOLI data set from NASA's GEDEX CD-ROM [22] was used. It consists of 4,794 data points on daily averages of solar irradiance, collected by the NIMBUS-7 satellite between 1978 and 1991.

Temperature data set

The temperature data set was extracted from the GEDEX CD-ROM as well. It consists of a more restricted set of average monthly temperatures over a 20 year period for a specific spatial location.

Evaluation Criteria

Some of the criteria which can be used to evaluate the performance of regression database are:

1. *Compression ratio.* This parameter is a measure of how many data points can be mapped to a single function in practical situations.

2. *Precision.* The accuracy of answers provided by the regression database to a class of queries including aggregate queries (e.g. COUNT, RANGE, AVERAGE).

3. ***Robustness.*** This test evaluates the behavior of the method in the presence of missing information from the original database..

4. ***Query speed.*** The response time in answering a query by the regression database.

5. ***Maintenance Efficiency.*** This relates to the computational as well as the I/O overhead associated with the adaptive maintenance techniques.

Our current evaluation focused on the compression ratio, precision and robustness of the techniques.

5.2 Tests with the synthetic database

• Algorithm-A: We observed that, when attempting to fit the data with a 6th degree polynomial, the data set is split repetitively and 153 bounding boxes are generated. The algorithm is essentially free running with no control on the number of splits. As expected, it results in low compression ratios, due to its high susceptibility to "outlier" points.

• Algorithm-B:

Table 5.2 illustrates the storage overhead required as a function of the average overflow length. The test case was with a 6-degree polynomial fit and for $\varepsilon = 0.05$.

# of segments	Storage overhead (words)	Average "outlying" points/box
31	1457	11.65
31	1519	11.65
28	1428	12.93
24	1272	15.46
22	1210	17.00
21	1197	18.05
19	1121	20.05
19	1159	20.05
19	1197	20.05
19	1235	20.05
17	1139	23.00

Table 4: Storage overhead correlated to the average number of "outlying" points in each bounding box. (Algorithm-B, polynomial fit of degree 6, and $\varepsilon = 0.05$)

As expected, we observe that the number of bounding boxes decreases as we increase the capacity of the overflow arrays. The resulting curves can be used for interpolation, but the storage overhead associated with this algorithm is still unacceptable.

• Algorithm-C:

Table 5 shows data points distributions with respect to the relative error levels, when fitting the synthetic data set with a 6-degree polynomial using Algorithm-C. One can observe that there is a limit to deleting "bad" points. If this limit is exceeded, the performance will degrade. For example consider the number of points whose relative error is with 0.05. Deleting points whose relative errors are greater than 0.1 (iteration 4), results in a 522 points instead of the 544 points when deleting the points with relative error greater than 0.2 (iteration 3). Algorithm-C adjusts by picking the best fit (iteration 5). Furthermore, due to the

Deletion iteration	# points deleted	relative errors distribution					
		≤ 0.05	≤ 0.1	≤ 0.2	≤ 0.35	≤ 0.55	> 0.55
#1	0	518	318	124	23	6	11
#2	17	517	319	124	23	0	0
#3	23	544	300	115	1	0	0
#4	116	522	228	42	2	0	0
#5	0	544	301	128	16	3	8

Table 5: Fit of synthetic data set with 6-degree polynomial using Algorithm-C

use of a polynomial fitting function, the compression ratio is still unacceptable in most situations and was noted to be around 2.18 for ($\varepsilon = 0.05$).

We observed that when x is large, the 6-degree polynomial is dominated by c_6. Then it is not worth calculating c_0 to c_5. On the other hand, if a 5-degree polynomial can fit the data set well, there is no gain achieved by increasing the degree to 6 because c_6 will turn out to be close to zero. Furthermore, polynomial curve fitting functions have a rounding error problem. The errors may propagate as the degree is increased, which deteriorates the accuracy of the result. For the synthetic data used, it is expected that this phenomenon occurs if the order of the polynomial fitting function is increased from 6 to 8.

Fourier series are often useful when polynomial interpolation techniques are not [2]. Increasing the number of terms of the Fourier series does not suffer from the rounding error problem as much as polynomials. But simply increasing the number of terms will need additional storage space and may not improve the performance in a noticeable way. We illustrate the use of Fourier series with real data sets evaluations.

5.3 Tests with real databases

Fourier Series versus Polynomial Interpolation

degree /term	Polynomial		Fourier Series	
	maximum	average	maximum	average
6	0.002136	0.001328	0.002197	0.001341
8	0.002182	0.001311	0.002145	0.001331
10	0.002146	0.001299	0.002077	0.001314
12	0.002143	0.001300	0.002090	0.001307
14	0.002228	0.001327	0.002054	0.001300
16	0.002169	0.001301	0.002052	0.001298
18	0.002203	0.001317	0.002029	0.001296
20	0.002145	0.001296	0.001976	0.001294
30	0.002277	0.001400	0.001878	0.001251
40	0.002108	0.001279	0.001864	0.001229
50	cannot calculate		0.001855	0.001215

Table 6: Relative errors for solar data set fit with polynomial versus Fourier series

The objective here is to use algorithm C and find a polynomial function or a Fourier series of n terms that fit the Gedex solar data within $\varepsilon = 0.002$. In this test, we attempted to fit the data in one box and

[2] One could actually evaluate other techniques such as Fourier Transforms

computed the relative errors. Table 5.3 shows maximum relative errors and average relative errors when fitting with polynomial versus Fourier series while the degree of the polynomial and the number of terms of Fourier series are increased. The maximum relative error was set to ≤ 0.002. One can observe that using a 6-degree polynomial to fit the solar data set outperforms the corresponding 6-term (and even the 8-term) Fourier series (Figures 2 and 3). On the other hand, neither of them satisfies the requirement. Increasing the degree of the polynomial does not improve the performance; the maximum relative errors are worse and the average relative errors are not stable. A 50-degree polynomial cannot get the precise coefficients due to the propagation of rounding errors. On the other hand when we increase the number of terms of the Fourier series, the maximum relative errors and the average relative errors decrease. A 20-term Fourier series satisfies the requirement (Figures 4 and 5).

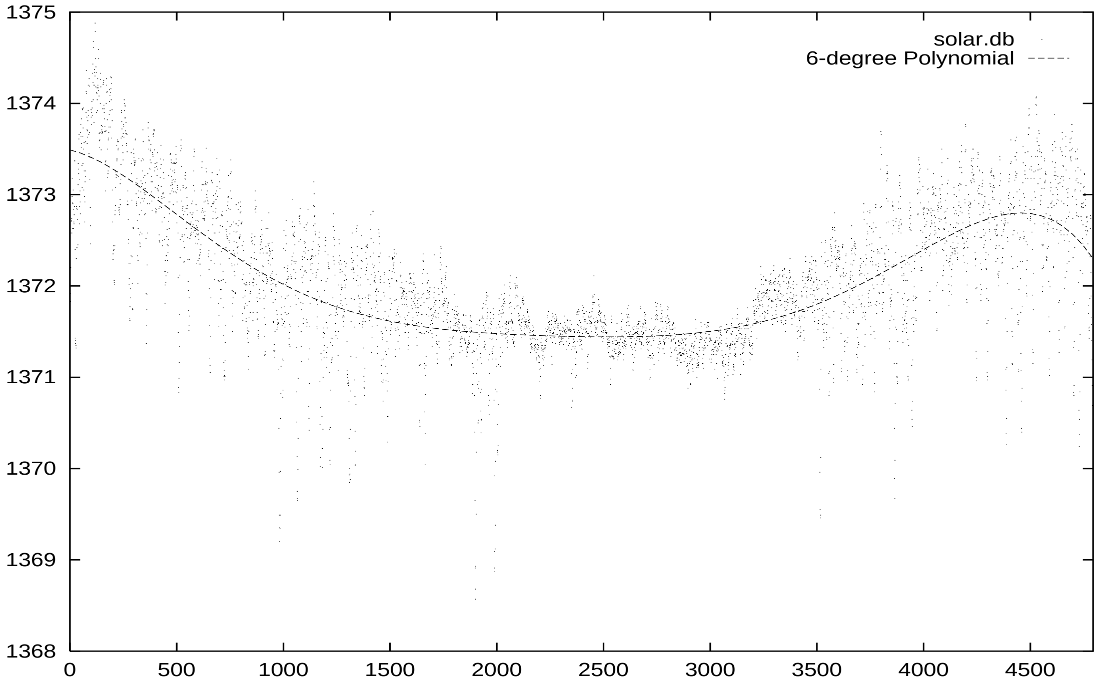

Figure 2: Fit of solar data set with 6-degree polynomial

Fourier Series with Algorithm-C

Deletion iteration	# points deleted	relative errors distribution					
		$\leq .0005$	$\leq .00065$	$\leq .00095$	$\leq .0014$	$\leq .002$	$> .002$
#1	0	4157	301	198	107	31	0
#2	31	4186	267	198	109	3	0
#3	112	4230	210	199	12	0	0
#4	211	4229	183	28	0	0	0
#5	0	4230	210	199	107	48	0

Table 7: Fit of the solar data set with a 50-term Fourier Series using Algorithm-C

Table 7 shows the distribution of the data points with respect to the relative error levels, when fitting

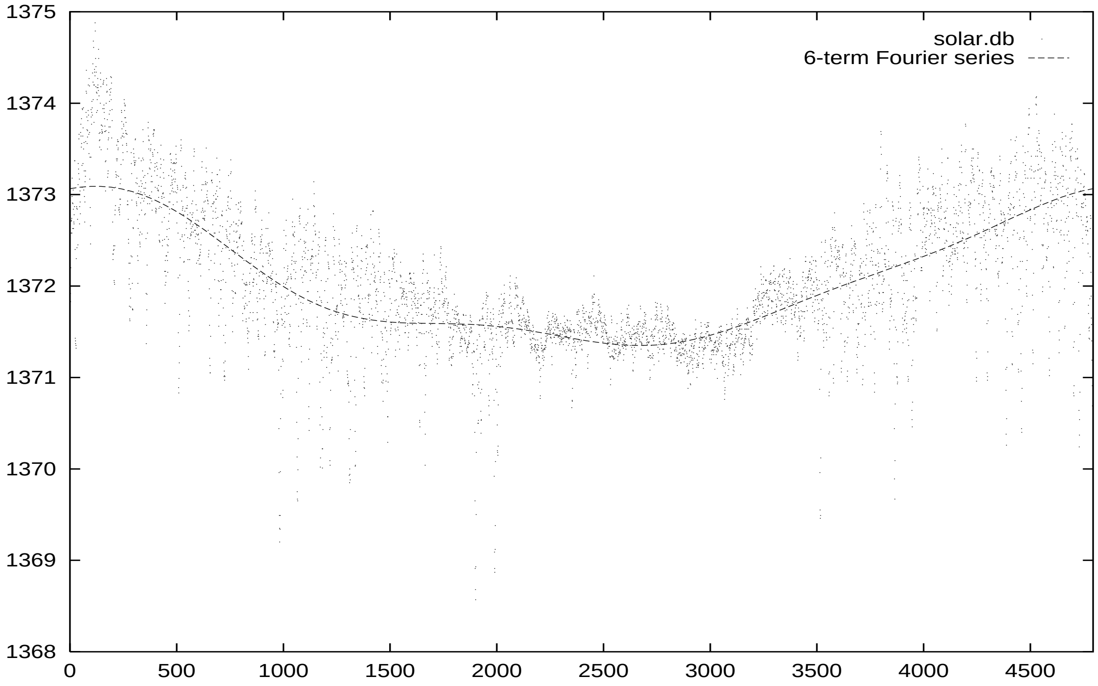

Figure 3: Fit of solar data set with 6-term Fourier series

the solar data set with a 50-term Fourier Series using Algorithm-C. If ε is set to 0.002, the ***compression ratio*** is 188:1, which is a practically useful ratio (Refer to Table 8). If ε is set to 0.0005 (illustrated in Figure 6), the compression ratio drops to 8.16:1. Obviously, lower precision imply higher compression ratios. We observe that for both test cases, the precision is high and the query speed is negligible. On the other hand, the off-line processing overhead is noticeable, but is acceptable.

ε (***relative error***)	0.0005	0.002
compression ratio	8.160	188.0
query speed	negligible	
off-line preprocessing time running on Sun Sparc 10	1 min 25 sec	20 sec

Table 8: Fit of solar data set with 50-term Fourier series

We further fitted the, GEDEX extracted, temperature data set with a 40-term Fourier series using Algorithm-C, setting (***absolute error***) $\varepsilon = 6^oC$ (Table 9 and Figure 7). We observed high precision in the results and negligible query speed. The off-line algorithm needed 4 seconds to preprocess. On the other hand, the compression ratio was low and is attributed to the small data set under consideration. We can expect that increasing the number of data points while keeping the number of terms of the Fourier series constant, we can achieve larger compression ratio.

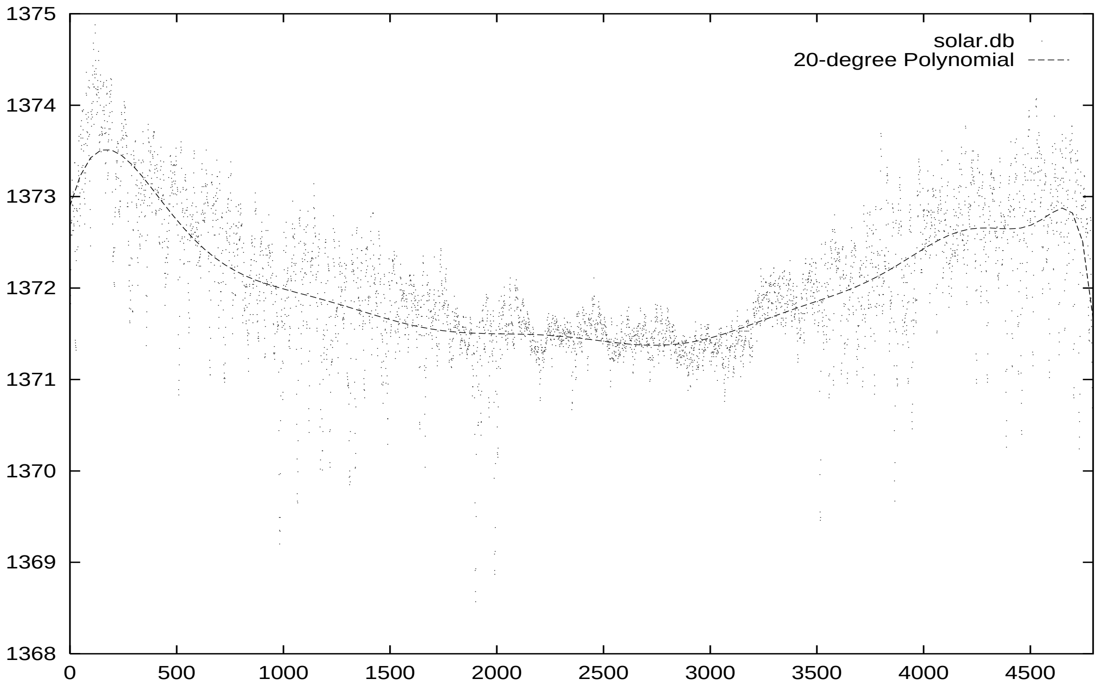

Figure 4: Fit of solar data set with 20-degree polynomial

ε (*absolute error*)	6	11
compression ratio	7.377	10.975
query speed	negligible	
off-line preprocessing time running on Sun Sparc 10	4 sec	negligible

Table 9: Fit of temperature data set with 40-term Fourier series

Robustness tests

Robustness tests were performed on both Algorithm B and C, using either a 6-degree polynomial or a 50-term Fourier series.

Algorithm-B on the synthetic data set

For this case we fitted the data points with a 6-degree polynomial function. We used the fitted curve to compute the average by integration and compared it to the true average (which is 11.504342). We then successively deleted 10, 20, 50, 100 data points. Each test was repeated 10 times and the results averaged. Relative errors are tabulated in Table 10 It is observed that the relative errors are stable and thus the function is reliable.

Algorithm-C on the solar data set

Algorithm-C was tested using a 50-term Fourier series. The data set included 4,794 data points. We performed the test over intervals of different widths over the range of data values. For each widths of an

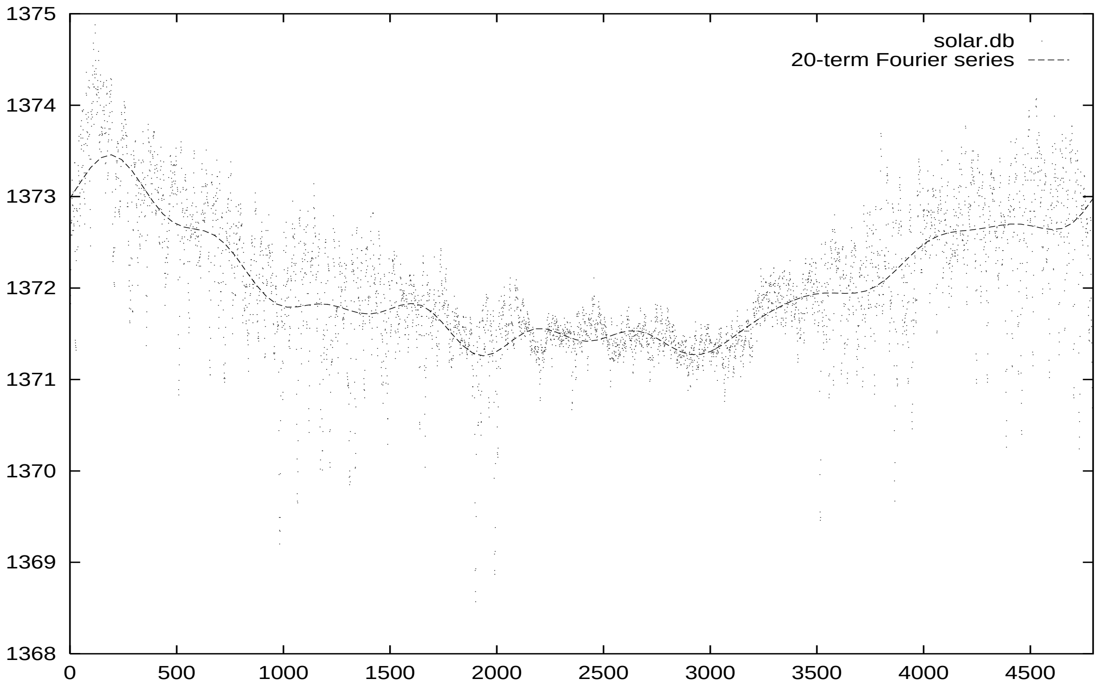

Figure 5: Fit of solar data set with 20-term Fourier series

# of points	1,000	990	980	970	960
Average by Integration (6-th order polyn.)	13.179896	13.188515	13.191811	13.226879	13.174106
Relative Error	.145645	.146394	.146681	.149729	.145142

Table 10: Robustness of Algorithm-B for an AVERAGE Query on the synthetic database

interval (referred to as a sliding window), th test was repeated 10 times and the results averaged. The sliding window width were 5%, 10%, and 20% respectively. The average over each interval was computed using the real data set, and then integration over the fitting function.

Finally, we randomly deleted 5% (and 10%) of the data set and fitted the remaining points. We then repeated the tests with the sliding windows. The results are tabulated in Table 11. One can truly observe that the relative errors are very small and stable. For example, even when deleting 10% of the data, the fitting function still reports stable and accurate query results. The average of the relative errors is 10^{-6}. This would be extremely valuable in noisy environments.

5.4 Comparisons with other approaches

No existing methods that we know of supports the notions of approximate query answering based on regression techniques, similar to the ones presented in this work.

Previous work on query optimization concentrated on designing fast index structures. Our approach can quickly respond to queries because we avoid searching the index, but simply calculate the fitting

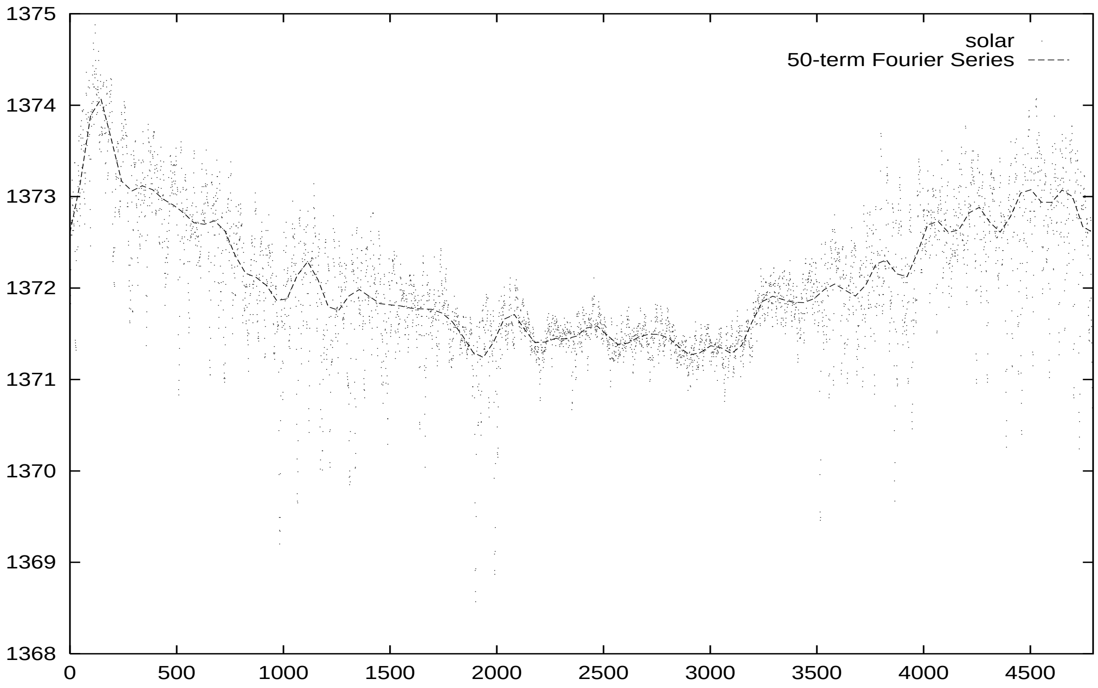

Figure 6: Fit of solar data with 50-term Fourier series

function and return the approximating value [3]. Meanwhile, we do not need to store the index structure in addition to the original database. We only need to store the regression database which includes the sets of coefficients of the fitting functions and the overflow arrays. Yet, our approach is achieved at the cost of the accuracy of the query results.

Recent work on query optimization essentially attempt to map the distribution of the actual data by statistical methods [7, 9, 12, 15, 18, 19, 20, 23, 25, 27]. We intend to map the actual data directly against regression functions. With histogram methods [18, 19, 23, 25], one needs to store the detailed statistics about the database. Parametric methods have a problem [7]: if no known statistical model fits the actual distribution, any attempt to approximate the distribution will be in vain. Sampling methods are rather costly due to run-time disk I/O [9, 12, 15, 20]. Only curve fitting methods are similar to our approach [27]. Our approach can provide accurate approximating functions to any kind of data set since the functions are derived from the characteristics of the actual data.

Chen and Roussopoulos proposed a method of approximating the attribute value distribution by a polynomial function using a query feedback mechanism [3]. Their approach is useful when queries cover a small range of the database. By contrast, our approach provides approximate query answers over the full range and is based on the actual data values.

[3] When there are more than one bounding box associated with a fitting function in our regression database, we first search into which box the queried data point is falling, then we can use the corresponding fitting function to compute the approximating value. However, searching for the box within some boxes is much faster than searching for a data point within thousands of data points.

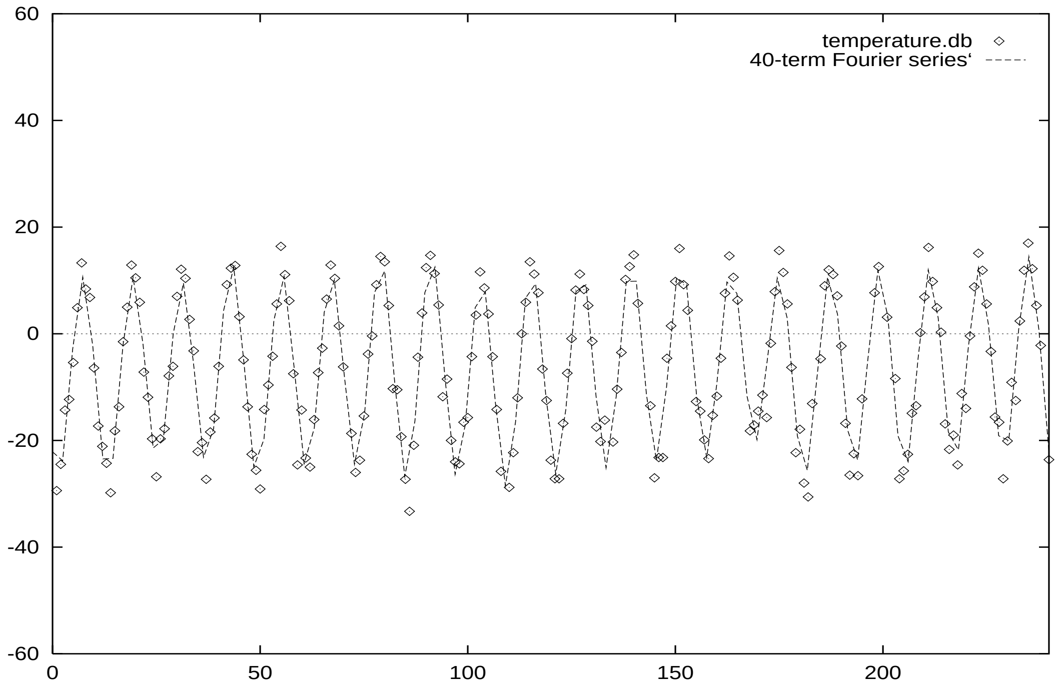

Figure 7: Fit of temperature data with 40-term Fourier series

6 Conclusions and Future Work

A general framework for building regression databases was proposed. Regression databases can be used for approximate but efficient query processing in large scientific and numerical databases. Queries on databases can be mapped to queries on the associated regression functions. Answers from fitting functions are used as an approximation of the exact answer against the actual data.

To evaluate the ideas, we implemented three different algorithms: a naive non-controlled algorithm, a controlled split algorithm and a third algorithm that uses a "relaxation" of LSR to cope with outlier data points. We performed various experiments using a synthetic as well as real data sets from the GEDEX database [22]. Experimental results indicate that our approach is promising in that it can achieve a 188:1 compression ratio over the original database while at the same time being able to provide answers to queries within a relative tolerable error as low as 0.002.

In situations where precise values are not necessary (for example, to estimate temperature and rainfall), our approach is useful. In addition, our approach is robust when the database suffers from loss of information. Moreover, the derived regression functions can be used for extrapolation, providing forecast for future. This would be part of future research on extrapolation techniques.

To reduce the space overhead of the regression database while at the same time providing accurate answers to queries, the regression functions should be able to fit most data points in the original database. However no known regression technique exists for this purpose. We used LSR as a starting point in our framework, and proposed a technique called "relaxation LSR" to overcome the disadvantages of LSR.

Several topics exist for future research. Many were outlined in Section 3. Furthermore, we are planning to address other topics including investigating the use of sampling techniques in order to speed up the computation of regression coefficients; further evaluation of different adaptive algorithms for curve fitting

#points	interval	AVERAGE by sum.	AVERAGE by int.	rel.err
4794	5% of 0-4794	1371.910162	1371.894578	-0.000011
	10% of 0-4794	1371.846219	1371.844480	-0.000001
	20% of 0-4794	1371.773551	1371.772805	-0.000001
4554	5% of 0-4794	1372.160800	1372.165089	0.000003
	10% of 0-4794	1372.079334	1372.081469	0.000002
	20% of 0-4794	1371.950266	1371.949222	-0.000001
4314	5% of 0-4794	1372.230063	1372.241515	0.000008
	10% of 0-4794	1372.147216	1372.148703	0.000001
	20% of 0-4794	1372.023440	1372.021121	-0.000002

Table 11: Robustness of Algorithm-C for an AVERAGE query

very large numerical data sets, and expanding on the prototype. Finally we will be investigating the use of such techniques to extend current data models to include interpolation, curve fitting and extrapolation techniques as basic constructs in numerical databases. Other areas of application, such as use for multimedia audio and video data signatures will be investigated.

References

[1] D. Birkes and Y. Dodge, ***Alternative Methods of Regression***, John Wiley & Sons, 1993.

[2] S. Chatterjee and B. Price, ***Regression Analysis by Example***, John Wiley & Sons, 1977.

[3] C. M. Chen and N. Roussopoulos, ***Adaptive Selective Estimation Using Query Feedback***, In Proceeding of ACM-SIGMOD International Conference on Management of Data, 1994, 1-20.

[4] B. Chenye, ***Approximate Query Answering in Numerical Databases***, MS thesis, WPI CS dept., Worcester, MA, 1995.

[5] W. Cochran, ***Sampling Techniques***, John Wiley & Sons, 1977.

[6] N. R. Draper and H. Smith, ***Applied Regression Analysis***, John Wiley & Sons, 1966.

[7] J. Fedorowicz, ***Database Evaluation Using Multiple Regression Techniques***, In Proceedings ACM-SIGMOD International Conference on Management of Data, Boston, MA, 1984, 70-76.

[8] G. Graefe, ***Query Evaluation Techniques for Large Databases***, ACM Computing Survey Vol 25 No 2, June 1993, 73-170.

[9] P. J. Haas and A. N. Swami, ***Sequential Sampling Procedures for Query Size Estimation***, In Proceeding of ACM-SIGMOD International Conference on Management of Data, San Diego, CA, 1992, 341-350.

[10] W. Hou, G. Ozsoyoglu and B.K. Taneja, ***Statistical Estimators for Relational Algebra Expressions***, In Proceeding of ACM SIGMOD, Austin, TX, 1988, 276-287.

[11] W. Hou, G. Ozsoyoglu and B.K. Taneja, ***Processing Aggregate Relational Queries with Hard Time Constraints***, In Proceeding of ACM SIGMOD, Portland, OR, 1989, 68-77.

[12] W. Hou, G. Ozsoyoglu and E. Dogdu, ***Error-Constrained Count Query Evaluation in Relational Databases***, In Proceeding of ACM SIGMOD, 1991, 278-287.

[13] M. Jarke and J. Koch, ***Query Optimization in Database System***, ACM Computing Surveys, Sept. Vol. 16, 1984, 111-152.

[14] P. Lancaster and K. Salkauskas, ***Curve and Surface Fitting***, Academic Press, 1986.

[15] R. J. Lipton, J. F. Naughton and D. A. Schneider, ***Practical Selectivity Estimation through Adaptive Sampling***, ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, 1990, 1-12.

[16] W.S. Luk and P.A. Black, ***On Cost Estimation in Processing a Query in a Distributed Database System***, In Proceeding of the IEEE 5th International Computer Software and Application Conference, Chicago, IL, Nov. 1981, 18-22. IEEE, New York, 24-32.

[17] M. V. Mannino, P. Chu and T. Sager, ***Statistical Profile Estimation in Database Systems***, ACM Computing Surveys, Sept. Vol. 20, No. 3, 1988, 191-221.

[18] T.H. Merrett and E. Otoo, ***Distribution Models of Relations***, In Proceedings 5th VLDB Conference, Rio De Janero, Brazil, 1979, 418-425.

[19] M. Muralikrishma and D. J. DeWitt, ***Equi-depth Histograms for Estimating Selectivity Factors for Multi-dimensional Queries***, In Proceedings ACM-SIGMOD Conference on management of Data, Chicago, Illinois, 1988, 28-36.

[20] F. Olken and D. Rotem, ***Simple Random Sampling for Relational Databases***, In Proceeding 12th VLDB, Kyoto, August, 1986, 160-169.

[21] F. Olken and D. Rotem, ***Random Sampling from B+ Trees***, In Proceeding 15th VLDB, Amsterdam, 1989, 269-278.

[22] L. M. Olsen and A. Warnock III, ***GEDEX: Selected Data Sets for Greenhouse Effect Detection Experiment***, The Space Agency Forum on The International Space Year, NASA, 1992.

[23] G. Piatetsky-Shapiro and C. Connell, ***Accurate Estimation of the Number of Tuples Satisfying a Condition***, In Proceedings ACM-SIGMOD International Conference on Management of Data, Boston, MA, 1984, 256-275.

[24] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, ***Numerical Recipes in C***, Cambridge University Press, 1992.

[25] P.G. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie and T.G Price, ***Access Path Selection in a Relational Database Management System***, In Proceedings ACM-SIGMOD, San Jose, CA, 1979, 23-34.

[26] J.Stoer and R.Bulirsch, ***Introduction to Numerical Analysis***, Springer-Verlag, 1993.

[27] W. Sun, Y. Ling, N. Rishe and Y. Deng, ***An Instant and Accurate Size Estimation Method for Joins and Selection in a Retrieval-Intensive Environment***, ACM SIGMOD International Conference on Management of Data, Washington, DC, 1993, 79-88.

[28] C. Yu and C. Chang, ***Distributed Query Processing***, ACM Computing Surveys, 1984, 399-433.