

EVALUATION OF THE DIRECT LINEAR ALGORITHM FOR THE COMPUTATION
OF 2D HOMOGRAPHIES

by

Srinija Kalluri

ProQuest Number: 10781279

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10781279

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.

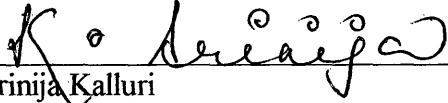
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.


ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

An Engineering Report submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of *Master of Engineering* (Engineering Systems)

Golden, Colorado

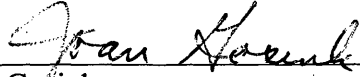
Date 11.10.2002

Signed: 
Srinija Kalluri

Approved: 
Dr. Christian H. Debrunner
Thesis Advisor

Golden, Colorado

Date 11/15/02


Dr. Joan Gosink
Division Head, Professor
Division of Engineering

ABSTRACT

The 2D projective transformation or homography is a compact description of image motion over time. Several algorithms have been developed to compute the projective transformation and we focus on the two, which are computed from point correspondences: the Direct Linear Transformation (DLT) Algorithm and the Gold Standard (GS) Algorithm. Building upon Hartley's work on the 2D projective transformations (18), we seek to improve the understanding of the DLT. We measure the accuracy and the robustness of the DLT over a wide range of homographies and over a varying number of point correspondences. We compare the results with the correct solution and the maximum likelihood solution obtained by the GS. The results show that the d parameter (the distance from origin to the first image which projects to a line in the second image) is the primary determinant that determines the accuracy and the robustness of the DLT algorithm.

TABLE OF CONTENTS

ABSTRACT.....	(iii)
LIST OF FIGURES.....	(v)
ACKNOWLEDGEMENTS.....	(vi)
CHAPTER 1. INTRODUCTION.....	1
CHAPTER 2. LITERATURE REVIEW.....	5
CHAPTER 3. ANALYSIS.....	7
3.1 Description of the problem.....	7
3.2 Description of the DLT algorithm.....	15
3.3 Cost functions.....	17
3.4 Optimal GS algorithm.....	20
3.5 Effect of normalization.....	21
CHAPTER 4. EXPERIMENTAL RESULTS.....	24
4.1 Plots.....	24
4.2 Choice of parameters of the decomposition of homography.....	26
4.3 Analytical Results.....	27
4.4 Effect of various parameters on average fit error and failure plots.....	29
CHAPTER 5. CONCLUSIONS AND FUTURE WORK.....	33
REFERENCES.....	34
APPENDIX A.....	37
APPENDIX B.....	50

LIST OF FIGURES

1. Projective transformation between two images.....	8
2. Mapping between points in two planes.....	8
3. Reprojection error in two images.....	20
4. Case 1 – Affine plots of DLT algorithm with point and error normalization.....	52
5. Case 2 – Non-affine with $d=100$ with DLT algorithm.....	62
6. Case 3 – Non-affine $d=1000$ with point and error normalization with DLT algorithm.....	67
7. Case 4 – Non-affine with $d=10000$ with point and error normalizations with DLT algorithm.....	72
8. Case 5 – Plots with correct solution.....	83
9. Case 6 – GS solution with LM minimization.....	94
10. Case7 – DLT algorithm without point normalization for affine case with d set to infinity.....	97

ACKNOWLEDGEMENTS

I express my deepest gratitude to my advisor, Dr. Chris Debrunner. With his enthusiasm, inspiration, and great efforts to explain things clearly and simply, he made computer vision easier and fun for me. I am also thankful to Dr. Hoff and Dr. Simoes for serving on the committee and for their valuable suggestions. Most of all, I thank my parents and sisters for their constant encouragement and support. I wouldn't have been able to do anything without the support of my husband, Dr. "J" (as his students call him) and without the smiles of baby Srineeth.

Chapter 1. Introduction

The 2D projective transformation or homography (also known as collineation and projectivity) is a compact description of image motion over time. For example, projective transformation provides a useful description of motion for pure camera rotation, and of planar surfaces (2D) under general motion. Determining the projective transformation, between regions of two images is an important problem in computer vision whether it is for tracking or registration. In addition to many applications in computer vision area, for example recognition and surveillance, it has a number of applications for computer graphics, such as image mosaicing, model-based compression, and so forth. Recovering the projective transformation is particularly important for applications that resynthesize the images, such as *image mosaicing* (1)(2)(3). Image mosaicing is a common and popular method of effectively increasing the field of view of a camera, by allowing several views of a scene to be combined into a single view. If we assume that either the camera is rotating in a stationary scene, or that the scene is approximately planar so that there is no parallax induced by the motion, the transformation between the views is projective. The aim is to recover the projective transformations or homographies that map images to each other, and hence allow the images to be transformed and combined in a single coordinate system. Another example where the computation of homographies is used is for video compression. Video compression is becoming increasingly important in the modern age where people all over the world are connected via the Internet, and video

the world are connected via the Internet, and video conferencing has become an important way to stay connected. However, to be able to do so, the video files should be small enough so that the transfer rate and the download time are less. In this technique, the key frame is divided into segments of a general shape so that motion of the each of segments is predicted well with the help of homography. Planar (2D) homography, which is used for motion prediction, gives us small errors between the predicted intermediate frames and the true frames. Other applications of homographies are for feature and region tracking. All these applications require an accurate transformation; the use of simplified models or inaccurate parameters may lead to undesirable visual artifacts. Even in cases where the transformation doesn't exactly describe the true and accurate motion, the projective transformation provides a better approximation to account for depth effects (4)(5) than simpler transformations. Projective transformations are a superset of other transformations such as affine, similarity, translational, Euclidean¹ and therefore can provide more accurate descriptions of motion. Therefore, there is a constant need to develop new methodologies and to improve the existing methods for determining the homography between images.

Several algorithms have been developed to compute the projective transformation and we focus on the Direct Linear Transformation (DLT) Algorithm and the Gold Standard (GS) Algorithm (11)(12)(18). The direct linear transformation (DLT) was originally developed by (6) and was revised by several authors, e.g. in (7). While these algorithms

¹ Please refer to Appendix B for complete list of the transformations and their geometric properties

of the homography, which is very critical in determining an optimal solution, there are tradeoffs associated with them. For example, the DLT algorithm is computationally very efficient because it is linear but does not provide an optimal solution and has stability problems without *normalization*. Normalization is a technique in which the data points are translated such that the centroid is at the origin and then they are scaled such that the average distance from the origin is $\sqrt{2}$; we discuss the normalization technique in more detail later. On the other hand, nonlinear iterative algorithms like the GS algorithm are very accurate in determining the maximum likelihood solution, but the convergence is not guaranteed under certain degenerate (a bad starting point) starting conditions. It is well known that the DLT algorithm is not optimal and can give severely biased estimates when the noise level is high. However, the results from the linear algorithms can be used as a starting point to initialize the search process of nonlinear algorithms. Due to their geometric simplicity, clearly understanding of the linear algorithms certainly helps in developing and understanding more sophisticated and efficient motion estimation schemes. While Hartley conducted some experiments in computing homographies with the DLT algorithm, he did these experiments over a limited set of homographies. Moreover, he did not study the effect of specific parameters of homography on the accuracy and robustness of the DLT. We believe that the stability of the DLT algorithm can be further improved. The goal of this thesis is to conduct experiments over a wide range of homographies, providing data for future improvements to the DLT algorithm.

Therefore, building upon Hartley's work on the 2D projective transformations, we seek to improved the understanding of the DLT by:

1. using an intuitive decomposition of the homography and studying the effects of the various parameters in this decomposition;
2. unlike in Hartley's experiments, characterizing the error level in the solutions and the failure rates of the algorithms under a variety of homographies and numbers of corresponding points;
3. compare the performance of the DLT algorithm to the GS solution and to the correct solution.

Chapter 2. Literature Review

Extensive work has been done in the area of computation of homographies for different applications in the area of computer vision. For example, computation of homographies is very important for *Image rectification*, an important component of stereo computer vision algorithms (14). Image rectification is the process of applying a pair of two-dimensional projective transformations to a pair of images whose epipolar geometry is known so that epipolar lines in the original images map to horizontally aligned lines in the transformed images.

Many new algorithms have been developed and existing algorithms have been modified to get a better estimate of projective transformation. The projective transformation for stereo rectification can be computed from fundamental matrix. Thus, the computation of fundamental matrix is a critical step for stereo rectification. For example, (19) in their recent paper proposed a new method for the computation of fundamental matrix from point correspondences. Their algorithm minimizes the algebraic distance as in the linear problem however they take into consideration the rank constraint. They show that the resulting least squares problem can be solved using convexification techniques. Their analysis shows that their method provides more accurate estimates of the fundamental matrix compared to those generated by the linear criterion in terms of the epipolar geometry. Hartley has been one of the pioneering researchers working to develop linear algorithms for computing homographies. In (13) he presented and evaluated a linear method for computing a projective reconstruction from a large number

of images. The algorithm that they use utilizes a Singular Value Decomposition (SVD) method to linearize the number of point matches used and hence it is computationally very efficient. In their recent paper Zelnik Manor and Irani (15) showed that for regions having at least two 3D planes, multiple view linear subspace constraints on their homographies could be derived. These constraints could be used to improve the current algorithms used for homography estimation. Another algorithm that received a lot of attention for the computation of homographies is the DLT algorithm. This algorithm minimizes the algebraic distance between the measured and estimated points in two images. As the name suggests, DLT is a linear algorithm and hence computationally efficient. This algorithm has been used by several researchers to compute 2D homographies between two matching regions in adjacent frames. Building upon previous research in this area, this project examines the accuracy and robustness of the DLT algorithm by characterizing the error in terms of parameters of the decomposition of the homography. Through the use of a wide range of errors – incremental and radical in magnitude—we establish the robustness of the algorithm. It also tests the algorithm over a wide range of homographies, which are characterized in an intuitive way. Using experiments that specially target the nature of this error the project extends prior research on DLT algorithm.

Chapter 3. Analysis procedure

3.1 Description of the problem

We have taken up the problem of estimation. Given a co-ordinate system, the points of real n -dimensional *projective space* \mathbb{IP}^n can be represented by $n+1$ -component real column vectors $(x_1, x_2, \dots, x_{n+1})^T$, with the stipulations that at least one coordinate must be non-zero and that the vectors $(x_1, x_2, \dots, x_{n+1})^T$ and $(\lambda x_1, \lambda x_2, \dots, \lambda x_{n+1})^T$ represent the same point of \mathbb{IP}^n for all $\lambda \neq 0$. The x_i are called *homogeneous coordinates* for the projective point i.e. if we consider a vector (or point) $(x, y)^T$ in the real plane, it can be represented as homogeneously as $(x, y, 1)^T$. A nonsingular projective mapping between two projective spaces is any mapping defined by multiplication of homogeneous coordinates by a full rank matrix. All projective mappings between the vector spaces can be represented by matrices. As with homogeneous coordinate vectors, these are only defined up to a non-zero scaling. Also, the product of a linear mapping, which is represented as a product of matrix and a vector, is also another vector.

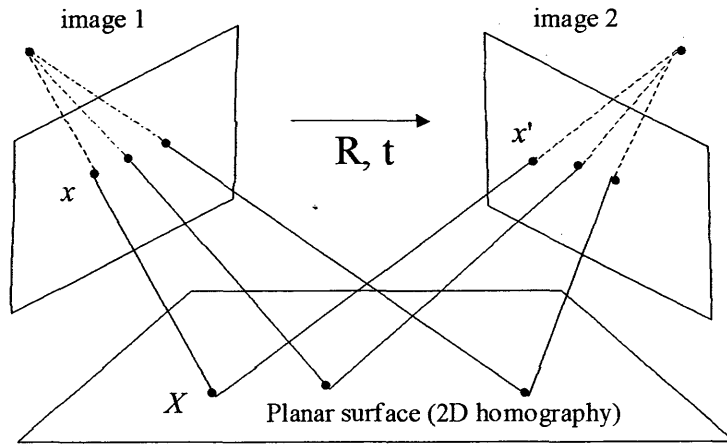


Figure1. Projective transformation between two images (Hartley and Zisserman (2000))

In Figure 1, A point, x in image 1 is related to another point, x' in image 2 that has been rotated and translated. (R, t in the Figure 1) via a homography, H such that $x' = Hx$. X is the world or 3D co-ordinate of the image points, x and x'

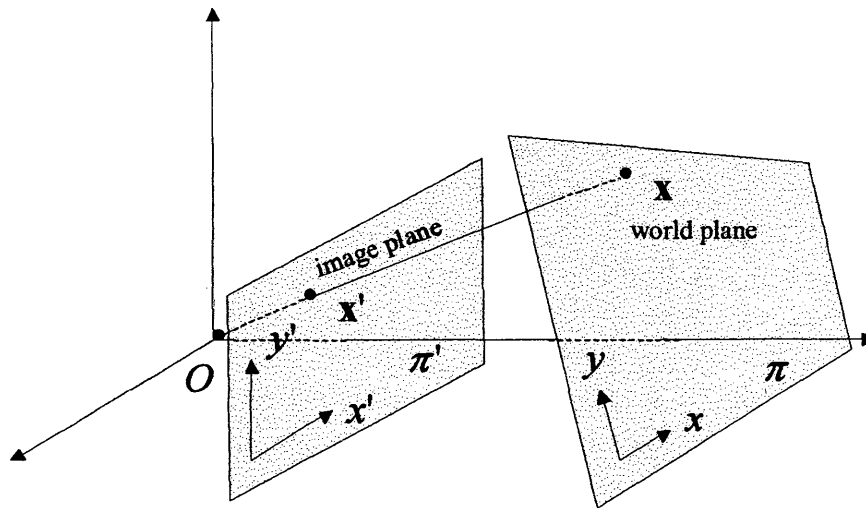


Figure2. Mapping between points in one plane to another (Hartley and Zisserman (2000))

The Figure 2 illustrates how images taken from a fixed camera location are related projectively. In the Figure 2, O can be thought of as a camera center and the line through both planes, π and π' in the Z-direction as the principal axis of the camera. The plane π is the world plane and π' is the image plane. The projective transformation maps points on one plane, π to another plane, π' such that the points both planes are related by a linear mapping such that $x' = Hx$.

The algorithm studied in this project is to compute projective mappings given set of points say X_i in IP^n and a corresponding set of points, X'_i in IP^n . These points could be the image points of any 3D point viewed by a camera. These two points, represented homogeneously, are related via a linear transformation given by

$$X'_i = HX_i$$

where H is linear mapping represented by a 3x3 matrix given by

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

The H matrix contains nine elements and to constrain H completely the first question that arises is how many point correspondences we need to consider to compute a homography. The number of correspondences required can be known by considering the number of constraints and the number of degrees of freedom. H has nine elements out of which we need to know eight ignoring the scale factor and for each 2D image point there

are two independent constraints corresponding to x and y components, giving rise to two equations in H . Therefore, we need at least four point correspondences to uniquely specify H , provided the four points are in “general position”, which means that no three points are collinear. More specifically, let us consider two homogeneous three-vectors given by

$$\begin{bmatrix} x_{11} \\ y_{11} \\ z_{11} \end{bmatrix}$$

and

$$\begin{bmatrix} x_{22} \\ y_{22} \\ z_{22} \end{bmatrix}$$

then the planar projective transformation between two vectors is given by

$$\begin{bmatrix} x_{22} \\ y_{22} \\ z_{22} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_{11} \\ y_{11} \\ z_{11} \end{bmatrix}$$

Now, as represented in Figure 2, let the inhomogeneous representation of the pair of matching points in the world and images plane be (x, y) and (x', y') . If we represent these image points inhomogeneously, since it is these inhomogeneous coordinates that are measured directly from the image, we can then write the homography, H between these points in inhomogeneous form as

$$x' = \frac{x_{22}}{z_{22}} = \frac{h_{11}x_{11} + h_{12}y_{11} + h_{13}}{h_{31}x_{11} + h_{32}y_{11} + h_{33}}$$

$$y' = \frac{y_{22}}{z_{22}} = \frac{h_{21}x_{11} + h_{22}y_{11} + h_{23}}{h_{31}x_{11} + h_{32}y_{11} + h_{33}}$$

As shown here each point correspondence generates two equations for the elements of the homography, which are linear in its elements after multiplying the denominator out. Four such point correspondences are needed to fully constrain the homography to solve for its eight unknown parameters (defined up to a scale).

Thus, four points give us an exact solution for H with zero error. If there is an error in the inputs, we still get a zero-error solution, however the solution will reflect the error in the inputs. With more than four point correspondences and no error we can also get a zero error solution assuming the points fit the model. There is always a noise component in the measurements, so for more than four point correspondences, in practice, there is never zero error solution for H . Therefore, given over-determined data we need to pick the “best” transformation, which is determined by some cost function that is minimized by this transformation. We use two error measures one for the DLT and another for the GS, which will be explained in detail in Section 3.3. In our project we generate synthetic data with errors drawn from a Gaussian (or normal) probability distribution. The project focuses on the propagation of the errors, i.e. to compute and measure the effect of various factors on the propagation of errors from the input data to the results. We also investigate the effect of prescaling as recommended by Hartley and Zisserman (18).

the results. We also investigate the effect of prescaling as recommended by Hartley and Zisserman (18).

We explore the various factors responsible for different effects on the propagation of errors using a decomposition of the projective transformation matrix. A homography can be decomposed into a chain of transformations where each matrix added to the chain creates a transformation that has more degrees of freedom than the previous one in the hierarchy². Writing down the homography (H_T) in the decomposition terms we have:

$$H_T = H_S H_A H_P \quad (1)$$

where

$$H_S = \begin{bmatrix} sR & t_{2 \times 1} \\ 0^T & 1 \end{bmatrix} \quad (2)$$

is a similarity transformation

$$H_A = \begin{bmatrix} K & 0_{2 \times 1} \\ 0^T & 1 \end{bmatrix} \quad (3)$$

and the product of $H_S H_A$ is an affine transformation

$$H_P = \begin{bmatrix} I & 0_{2 \times 1} \\ \mathbf{v}^T & \nu \end{bmatrix} \quad (4)$$

giving the complete projective transformation,

² Hierarchy of transformation as defined in [18]; with the most specialization transformation, isometries being the simplest in the chain of transformations and progressively generalizing to homographies

$$H_T = \begin{bmatrix} A & t_{2x1} \\ v^T & v \end{bmatrix}. \quad (5)$$

Here A is a non-singular matrix, $A = sRK + tv^T$ and K is upper triangular skew matrix normalized such that $\det K = 1$. The various parameters we are interested in, which affect the propagation of noise are:

- a. s , the scaling factor;
- b. the parameter, θ of the 2x2 rotation matrix, $R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$;
- c. the components, a and b , of K matrix given by $\frac{1}{\sqrt{a}} \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix}$; and
- d. d and ϕ , which specify $(v_1, v_2)^T$ as $v_1 = \frac{\cos(\phi)}{d}$ and $v_2 = \frac{\sin(\phi)}{d}$.

s accounts for isotropic scaling, a is a ratio of scales between x and y and b skews the image plane and v is fixed to be 1. The translation, t doesn't affect the solution because when we normalize the data we shift the centroid to the origin (as described in Section 3.5). The image plane is rotated by the angle given by θ and finally d is the distance of the line in image 1 that projects to the line at infinity in image 2, the normal to which is given by ϕ . To illustrate this last point, suppose a point

$$X_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

is projected into a point in the second image X_i' via the homography, H_T

Then,

$$X'_i = H_T X_i$$

where

$$H_T = \begin{bmatrix} a_{11} & a_{12} & t_1 \\ a_{21} & a_{22} & t_2 \\ \frac{\cos \phi}{d} & \frac{\sin \phi}{d} & 1 \end{bmatrix}$$

if we represent the elements of A of H_A as $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$. If the third element of X'_i is zero,

this indicates that it is a point on the line at infinity and if the third element of X'_i is zero,

we see that the last row of this matrix equation becomes,

$$x_1 \cos \phi + y_1 \sin \phi = -d$$

This line in the first image, whose distance from origin is d and whose normal is an angle ϕ , is projected to the line at infinity in the second image.

With only rotation, translation, scaling, and skew, the equations should produce the estimate, which minimizes the squared distance between the transformed points and the measured points. If we add the non-zero projective terms, v , however, the error minimized is no longer the least squares error. We then characterize the errors as a function of the "size" of the projective terms, the number of corresponding points, and the sigma of the added noise. In the next section we describe in detail the DLT algorithm and

also the optimal algorithm, GS for computing homography and we define the cost functions that are to be minimized.

3.2 Description of the DLT algorithm (18)

As the name suggests, DLT is a simple linear algorithm to compute homography. As discussed in the previous section, if H is the mapping between two points X and X' in the first and the second image respectively, then the point in the first image is related the second image point as

$$X' = HX \quad (6)$$

The elements of H can be written as

$$\begin{pmatrix} h^{1T} \\ h^{2T} \\ h^{3T} \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

such that $X' = HX$ can be rewritten as

$$X' = \begin{pmatrix} h^{1T} X_i \\ h^{2T} X_i \\ h^{3T} X_i \end{pmatrix} \quad (7)$$

Since the measured point position and the computed position are equal up to a scale, we note that their cross product is zero. Setting the co-ordinates of $X'_i = (x'_i, y'_i, z'_i)$ the cross product is given by

$$X_i' \times HX_i = \begin{pmatrix} y_i' h^{3T} X_i - z_i' h^{2T} X_i \\ z_i' h^{1T} X_i - x_i' h^{3T} X_i \\ x_i' h^{2T} X_i - y_i' h^{1T} X_i \end{pmatrix} = 0 \quad (8)$$

This equation can be again rewritten as

$$\begin{pmatrix} 0^T & -z_i' X_i^T & y_i' X_i^T \\ z_i' X_i^T & 0^T & -x_i' X_i^T \\ -y_i' X_i^T & x_i' X_i^T & 0^T \end{pmatrix} \begin{pmatrix} h^1 \\ h^2 \\ h^3 \end{pmatrix} = 0 \quad (9)$$

These equations have the form $A_i h = 0$, where A_i is a 3x9 matrix and h is a 9-vector consisting of elements of H . The equation $A_i h = 0$ is a linear equation in the unknown h . From observation, only two of the three equations of equation (9) are linearly dependent (third row can be obtained up to a scale, from the sum of x_i' times the first row and y_i' times the second row) which means we need only the first two rows making A_i a 2x9 matrix. Therefore, we can write

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \cdot \\ \cdot \\ \cdot \\ A_n \end{bmatrix}$$

and

solution for H and instead need to pick an optimal H that minimizes the cost function $\|Ah\|^2$ subject to the constraint $\|h\|^2 = 1$. The DLT algorithm is as follows:

Given four or more point correspondences:

Step1: Compute the matrix A_i and consider only the first two rows of A_i

Step2: Form a single $2n \times 9$ matrix, A from the n 2×9 matrices where n is the number of point correspondences under consideration

Step3: Obtain a singular value decomposition of A (SVD) (9). If $A = UDV^T$ where D contains entries (singular values) in the descending order along the diagonal, then h is the last column of V . H can be obtained by rearranging the elements of h into a 3×3 matrix. In the next section we discuss the cost function that is minimized in more detail.

3.3 Cost functions

Since we don't have an exact solution for H for an over determined system of equations, we need to select a cost function or an error function that is minimized by the optimal H which is our required solution. Describing the cost function for an algorithm is an integral part of the algorithmic solution for an over determined system. The importance of this can be better appreciated if we take the view that the parameters (or variables) in our system model correspond to unique and physically meaningful values. Then the algorithmic goal becomes that of parameter estimation (along with the associated covariance so that we can make practical use of the result). While DLT

algorithm minimizes the algebraic distance, the GS algorithm minimizes the geometric error (defined as the Euclidean image distance in second image between the measured point and $H\bar{x}$ where \bar{x} represents true value of the points). The DLT algorithm minimizes the norm $\|Ah\|$ of the vector $\varepsilon = Ah$ called the residual vector. From the equation $\varepsilon = Ah$, it is evident that the components of this vector are derived from the individual point correspondences that constitute the A matrix. Each point correspondence gives rise to a partial error, say $\varepsilon_i = A_i h$ and all the partial errors constitute full error, ε^3 . The error vector ε_i is known as the *algebraic error vector* for i^{th} point correspondence and homography. The norm of this vector is the *algebraic distance*. Generally stating, if x_1 and x_2 are any vectors, the algebraic distance is given by:

$$d_{alg}(x_1, x_2)^2 = x_1 \times x_2 \quad (10)$$

Now, if the homography we compute is affine, i.e. the last row of H , h_3 is constrained to be $[0 \ 0 \ 1]$, then, since $Z'_i = 1$, the algebraic error is given by

$$\left(X'_i - h_1 X_i\right)^2 + \left(Y'_i - h_2 X_i\right)^2$$

which is the distance in image 2 between the measured point and the predicted point HX_i . If the components of v , v_1 and v_2 , given in equation (4) are allowed to be non-zero then these errors are scaled by $h_3 X$ i.e. scaled by

³ Please refer to *Multiple view geometry in computer vision* by Hartley and Zisserman for more details

$$\left[\begin{array}{ccc} \cos \phi & \sin \phi & \\ \frac{1}{d} & \frac{1}{d} & 1 \end{array} \right] X$$

which is proportional to the distance from the line projecting to the line at infinity. If $h_3 X$ is approximately constant over all points (when d is large) this scale factor has little effect on the location of the minimum point. If the scale factor is different for each point we get uneven weighting of errors causing a bias in the solution. The solutions that minimize algebraic error may not be what we expect intuitively (10). However, with a good choice of normalization the DLT algorithm gives good results (the effect of normalization will be discussed in the next section). The results from the DLT algorithm can be used as the starting point for the non-linear minimization of the geometric distance such as Levenberg-Marquardt minimization technique used the GS algorithm.

To obtain the Maximum likelihood solution under an assumption of Gaussian error in the input, we need to estimate not only the correct homography but also the correct point locations. Then, the error we minimize becomes the deviation between the estimated correct point locations and the measured point locations in both images. This minimization must be carried out under the constraint that the point positions in the two images are related by a homography. This constrained problem is equivalent to estimating the point positions in image 1 and a homography and minimizing the distance between the estimated and the measured points in both images. In this case, however, the estimated points in image 2 are calculated from the estimated points in image 1 and the estimated homography. This error is known as *reprojection error* and is given by

$$\sum_i d(X_i, \hat{X}_i)^2 + d(X'_i, H\hat{X}_i)^2 \quad (11)$$

where X_i and \hat{X}_i represent the measured and estimated point positions in image 1, X'_i represents the measured point positions in image 2, \hat{H} represents the estimated homography and $d(\cdot, \cdot)$ stands for the point-to-point Euclidean image distance. Minimizing this error function involves determining both the estimates of the homography given by \hat{H} and also the point estimates given by \hat{X}_i .

Reprojection error is more sensible than the algebraic error because this estimate also gives us the optimal position estimate.

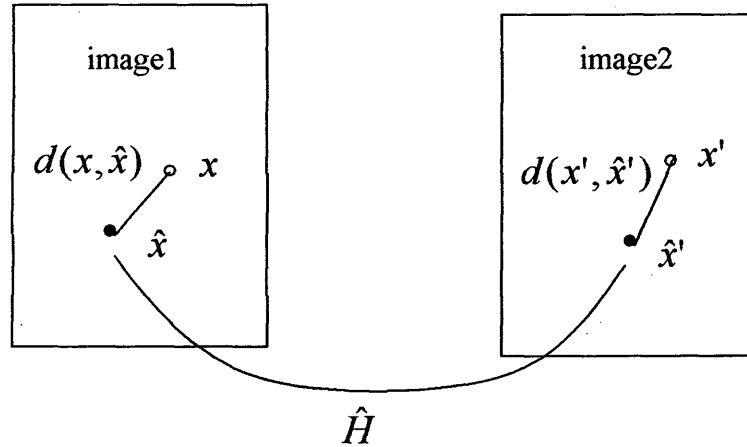


Figure 3. Reprojection error in two images

3.4 Optimal GS algorithm

As mentioned before we also performed the optimal algorithm for the computation of homographies so as to compare the accuracy and robustness of DLT algorithm. For more than four point correspondences, our objective is to obtain the Maximum likelihood

estimate of the homography. This is obtained by minimizing the error function (11).

Therefore, the method used in the GS algorithm consists of the following steps:

Step1: Compute the initial estimate using normalized DLT algorithm

Step2: Minimize the error function given in equation (11) using the Levenberg-

Marquardt algorithm over a suitable parameterization of \hat{H} and \hat{X}_i

We minimize this error function using Levenberg-Marquardt minimization algorithm.

If we consider the Figure 3 x and \hat{x} are the measured and estimated points in image 1 and x' and $H\hat{x}$ are the measured and estimated image points in image 2. Then, the problem is to reduce the reprojection error, the distance as defined in equation (11), between the measured and estimated co-ordinates in images 1 and 2. The initial image points x and x' are generated synthetically and Gaussian distributed noise is added from which the new estimate \hat{H} is calculated. From this new estimate of the homography we calculated the estimate \hat{x}' for the second image from the equation $\hat{H}\hat{x}$. This error function is iteratively minimized until the difference between the estimated and the measured values is less than a certain tolerance value that we defined in our LM minimization function.

3.5 Effect of normalization

As mentioned in the earlier section, the DLT algorithm is not invariant to the choice of co-ordinate frame or scaling of points. To eliminate this effect we have to carry out a very important step before the DLT is applied. This important step is *point normalization*

of the data. Suppose if our points X and X' are changed by some corresponding transformations say T and T' such that $\tilde{X} = TX$ and $\tilde{X}' = T'X'$ where T and T' are 3x3 homographies. Substituting $X' = HX$ we get $\tilde{X}' = T'HT^{-1}\tilde{X}$ so homography is $\tilde{H} = T'HT^{-1}$ for the \tilde{X} and \tilde{X}' point correspondences. Therefore, the homography matrix found by the above method for the transformed points, \tilde{X} and \tilde{X}' can be converted to a homography for the original points for this transformation. Now, the obvious question is that whether our choice of points and co-ordinate frame is invariant to the transformations T and T' . If we choose similarity transformations with equal scale factors for both T and T' then the results of the algorithm are invariant to these transformations provided the algorithm minimizes the geometric distance. We would expect the same for the DLT algorithm that minimize algebraic error, however it is not the case. The proof for this is given in p.89-p.91 of (18). This obviously suggests that the results of the DLT algorithm for computing homographies depends on the co-ordinate frame in which points are expressed and the results are not invariant to similarity transformations which means that some co-ordinate frames are better than the others for homography computations. In order to remove the affect of choice of co-ordinate frame and scaling we carry out point normalization to the data before the DLT is applied. This technique improves the accuracy of the results and the algorithm will be invariant to the choice of co-ordinate origin and scaling. The steps for normalization are:

Step1: The points are translated so that their centroid is at the origin i.e.

and scaling we carry out point normalization to the data before the DLT is applied. This technique improves the accuracy of the results and the algorithm will be invariant to the choice of co-ordinate origin and scaling. The steps for normalization are:

Step1: The points are translated so that their centroid is at the origin i.e.

$$\sum_i S_{IS}(X_i - t_R) = (0,0)^T . \text{ Solving this equation we get translation,}$$

$$t_R = \frac{\sum_i X_i}{N}$$

Step2: The points are then scaled so that the average distance from the origin is equal to $\sqrt{2}$ i.e.

$$1/N \left(\sum_i S_{IS}(X_i - t_R)^T S_{IS}(X_i - t_R) \right) = 2 . \text{ Solving this equation we}$$

get the scale factor,

$$S_{IS} = \sqrt{\frac{2N}{\sum_i (X_i - t_R)^T (X_i - t_R)}}$$

Step3: The above two steps will give the components of the similarity transformations. The above two steps are carried out for the two images independently.

In our project we also show the effect of point normalization and show that the point-normalized results are much better. In the next section we discuss experimental results.

Chapter 4. Experimental Results

4.1 Plots

We ran several sets of experiments to evaluate the DLT algorithm for the computation of 2D homographies. The sets of plots we are interested in are how the average fit error of the points is going to vary with the number of point correspondences and how frequently the algorithm fails. A failure is defined as a solution with fit error values, greater than $10 \times$ the standard deviation of the input noise. Measuring the frequency of these failures allows us to test the robustness of our algorithm, and to determine which parameter(s) have a greater influence on the robustness of our algorithm.

These plots are obtained for different cases as described below:

1. The affine DLT case where d (as described in Section 3.1) is set to ∞ and other parameters of the decomposition matrix of H set to Identity. By doing so, the parameters as we discussed in Section 3.1, v_1 and v_2 are set to zero so that the last row of the projective transformation, H in equation (1) is set to $[0 \ 0 \ 1]$ i.e. since we are assuming v to be 1).
 - a. We then changed the parameters s, a, b, θ and ϕ of the decomposition matrix so as see the effect of these on the error plots.
 - b. We also carried out the experiment with and without error normalization. In section 4.3 we derive analytically the expected error given the correct homography in terms of the parameters, s, a, b and θ . In order to

eliminate the effects of these parameters on the error we normalize the fit error by this term as given by equation (12). This process is known as error normalization.

2. Then, we checked for nonaffine cases with finite values of d . We used values of 100, 1000, 10000. Obviously, by changing d , the last row of H is changed and is no longer $[0 \ 0 \ 1]$. We are then introducing v_1 and v_2 terms in the last row such that the last row now becomes $\left[\frac{\cos \phi}{d} \quad \frac{\sin \phi}{d} \quad 1 \right]$. Obviously, with larger d we would expect the v_1 and v_2 be close to zero thereby giving us solutions similar to those in the affine case. These sets of experiments were done with point normalization.
3. Then, we also ran the algorithm on the data without point normalization for $d=1000$ and changed the parameters, s, a, b, θ and ϕ to see their effect on the error plots. Again, we ran the experiments using Gold Standard Algorithm where the minimization of the error function is done by Levenberg-Marquardt algorithm. The results of DLT algorithm are compared with the results of the GS algorithm. These give us a lower bound of the error we expect for the DLT algorithm. Note that the GS errors could be lower than the errors with the exact solution.

The DLT algorithm was coded in MATLAB and the plots were obtained. We used the built-in function for the Levenberg-Marquardt minimization of the error function as given in equation (11).

4.2 Choice of parameters of the decomposition of homography

We have chosen a 100x100 pixel image. The range of isotropic scale factor s is taken to be 0.5 to 2.0. For the scale factor a , which is the ratio of scales between x and y is also taken to be 0.5 to 2.0. The range for the skew factor, b is taken to be -1 to 1 and for θ and ϕ , the range is from zero to $2 * \pi$. In any matching regions in two adjacent frames the range of the scale factors, s and a or the range for the skew factor b and ranges for θ and ϕ are sufficiently large i.e. we don't expect variations in real imagery greater than the ranges already tested. We used zero mean Gaussian noise with σ (standard deviation) error of 2% and the threshold for the failures in the error plots is taken to be $10 * \text{input noise level}$, thereby making sure we aren't *throwing away* any valid solution (at least 99.5% of points assuming a Gaussian distribution) in our plots. As described in (18) we would expect that as the number of point correspondences increases the average fit error should increase initially with a linear trend and then as the point correspondences increase further the average fit error levels out. Essentially, in our experiment we generate synthetic image 1 points from a random number generator, transform them by a randomly selected homography, and add noise drawn from a Gaussian distribution to the points in image 1 and image 2, and finally calculate the estimated homography. We then compute the error in the second image given by

$$d(X'_i, HX_i)^2.$$

We can't use the reprojection error to quantify the accuracy of the DLT result since only the homography is estimated.

4.3 Analytical Results

The concept of error covariance is very important in statistics as it allows us to model linear correlations between parameters. We now derive the expected value and the covariance of the image 2 error for the case where d is set to ∞ (i.e. affine case) and the true homography is known as projection.

X, Y are measured image positions in image 1 and 2 respectively (two vectors). We assume that these quantities have independent zero mean Gaussian noise with a variance of σ^2

\hat{Y} is predicted image position predicted from H and X

\bar{Y} is the expected value of Y

$\hat{\bar{Y}}$ is the expected \hat{Y} . Here \bar{Y} and $\hat{\bar{Y}}$ are the same since if \bar{X} is the expected X , then $\bar{Y} = H\bar{X}$

\bar{X} is expected X and $\bar{Y} = H\bar{X}$

From equations (1) to (5) we have that

$$\hat{Y} = sRK\bar{X} + T.$$

Then, we can find the mean and covariance of \hat{Y} as

$$\begin{aligned} \bar{Y} = \hat{\bar{Y}} &= E(\hat{Y}) = E(sKR\bar{X} + T) = sKRE(X) + T = sKR\bar{X} + T \\ C_{\hat{Y}} &= E\left[(\hat{Y} - \bar{Y})(\hat{Y} - \bar{Y})^T\right] = E\left[sKR(X - \bar{X})(X - \bar{X})^T R^T K^T s\right] \\ &= sKRE\left[(X - \bar{X})(X - \bar{X})^T\right] R^T K^T s \\ &= sKR\sigma^2 IR^T K^T s \\ &= \sigma^2 s^2 KK^T \end{aligned}$$

$$K = \begin{pmatrix} 1 \\ \sqrt{a \ b} \\ \sqrt{0 \ 1} \end{pmatrix} \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix} = \begin{pmatrix} 1 \\ \sqrt{a} \\ 0 \end{pmatrix} \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \frac{1}{\sqrt{a}} \end{bmatrix}$$

We have

$$\begin{aligned} KK^T &= \begin{bmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \frac{1}{\sqrt{a}} \end{bmatrix} \begin{bmatrix} \sqrt{a} & 0 \\ \frac{b}{\sqrt{a}} & \frac{1}{\sqrt{a}} \end{bmatrix} \\ &= \begin{bmatrix} \left(a + \frac{b^2}{a}\right) & \frac{b}{a} \\ \frac{b}{a} & \frac{1}{a} \end{bmatrix} \end{aligned}$$

We want to know the error between the predicted and measured point positions in image 2 given by

$$\begin{aligned} E|\hat{Y} - Y|^2 &= E\left[\left((\hat{Y} - \bar{Y}) - (Y - \bar{Y})\right)^T \left((\hat{Y} - \bar{Y}) - (Y - \bar{Y})\right)\right] \\ &= E\left((\hat{Y} - \bar{Y})^T (\hat{Y} - \bar{Y})\right) - 2E\left((\hat{Y} - \bar{Y})^T (Y - \bar{Y})\right) + E\left((Y - \bar{Y})^T (Y - \bar{Y})\right) \end{aligned}$$

Expected value of $\left(|\hat{Y} - Y|^2\right)$

$$\begin{aligned} &= \text{trace}(C_{\hat{Y}}) + 2\sigma^2 \\ &= \sigma^2 s^2 \left(a + \frac{b^2}{a} + \frac{1}{a}\right) + 2\sigma^2 \end{aligned}$$

$$= \sigma^2 \left[s^2 \left(a + \left(\frac{b^2 + 1}{a} \right) \right) + 2 \right] \quad (12)$$

This analytical result is important because it tells us what error to expect in our solution for the affine transformation if we have the exact H . By scaling our errors by the analytical value we can study the effects of the d parameter somewhat independently of s, a, b, θ and ϕ .

4.4 Effect of various parameters on average fit error and failure plots

For the affine problem with point and error normalization we set $d = \infty$. For all these plots shown in Figure 3 we have taken the number of point correspondences to range from 4 to 40, noise level set to 2% and threshold for failures set to 10*input noise level. We used 1000 point sets. The least number of point correspondences needed is four, which uniquely specifies the homography. When all the parameters of decomposition of the homography are set to identity i.e. $s=1, b=0, a=1, \theta=0, \phi=0$ and $d = \infty$ the average fit error started at zero for four points. In Figure 4d, the average fit error started at zero, and increased for up to 10 points and started to level off and remained so after 20 points. We also observe that the standard deviation decreased steadily to around 0.2 at 40 points. The maximum percent failure as seen in Figure 4n is 0.2 at 5 points. When all parameters vary randomly we observe a slight increase in the standard deviation and maximum percent failure is 0.2 as depicted in 4b and 4l.

When a is varied from 0.5 to 2.0, analytically from the error normalization term given in equation (12) we see that the plots for $a=0.5$ and $a=2.0$ should be the same. These observations are also verified experimentally (Figures 4a and 4k). Obviously, if we set $a=1.0$ this again becomes an affine case (since all other parameters are set such that the decomposition matrix is set to Identity) and as we expected, the standard deviations are lowest for this value of a . The plots are similar to each other when $b=-1$ and $b=1$. This is as predicted analytically. Experimentally, by varying θ and ϕ the plots look more or less the same for different values of θ and ϕ for affine case with all other parameters of the decomposition of homography set to identity matrix (Figures 4i, 4j, 4s and 4t for θ and Figures 4e, 4f, 4o and 4p for ϕ). There is a 0.2% failure at five points when θ is varied randomly as shown in Figure 4o. We also observed that changing the scale factor, s from 0.5-2.0, the lowest standard deviation of average fit error is for $s=2.0$ (Figures 4q and 4r). ϕ has no effect if $d = \infty$.

For the non-affine problem setting $d=100$, all the plots for different parameter values are very noisy and the percent failures are very high with error and point normalization. This is true because if the line at infinity is closer to the center, the errors become highly biased. The standard deviations are very high except for $\phi = \pi$. We don't have an explanation why this is the case. When $d=1000$, the plots resemble the affine plots except the standard deviations are a bit higher (Figures 6a-6j). For $d=10000$, the plots resemble the plots for $d= \infty$, i.e. affine plots. There are no significant differences in the plots for

variation of s, a, b, θ and ϕ . The error normalization seems to compensate for the effects of s, a, b, θ and ϕ .

Figures 7n –7u are the plots without error normalization. There are 1 and 0.2 percent failures at five and six points when $d = 10000$ with $s=1, a=1, b=0, \theta=0$ and $\phi = 0$. The average fit error increased to 3.4 from 0.2 without error normalization. Note that the failures are removed after the error normalization. Though there aren't significant failures when s, a, b, θ and ϕ are varied, the average fit errors increased significantly in accordance with equation (12). This shows that the DLT algorithm errors agree well with predicted error for affine as used in error normalization. Figures 10a to 10n show that without point normalization, there are significant failures when all parameters are varied showing that DLT algorithm fails without point normalization. The plots for the average fit error also look noisy. The failure rates are low if at all, in the order of 0.2%-0.4% for all cases except for $d = 100$ and for un-normalized data plots. There are no failures for $d=1000$ and for $d=10000$ except a failure of 0.2% when $s=2.0$ for $d=1000$ and for $a=2.0$ when $d=10000$.

With the correct solution and with error normalization, the expected average fit error is 1.0 whereas experimentally we obtained an error of 0.9. The differential appears to be due to an inherent bias in MATLAB random number generator. There are no failures except when all parameters are varied when $d=100$. In certain cases, the errors propagated from the source image are so high that the resulting error level was above our failure threshold. The error normalization term given in equation (12) is derived for the

affine case, so when $d=100$ the propagated errors appear to be much higher than their affine case. This also suggests that the parameter that affects our results significantly is the d parameter. With the GS algorithm with LM minimization, the expected error was around 2.8 however; we got an error of 2.5 apparently due to bias in the MATLAB random number generator. Some reduction in error at small number of points could be because the ratio of number of constraints to the degrees of freedom increases with number of points. This could be because we have more number of constraints than the number of variables and hence we could be getting better results. The average fit error with the GS is less than that for the DLT without error normalization. There are no failures in the results of GS.

Chapter 5. Conclusion and future work

In conclusion, we have evaluated the DLT algorithm and compared its performance with the optimal GS algorithm. The work on this project was based on the Hartley's experiments as specified in (18). Unlike in Hartley's experiments we have also characterized the errors in terms of the parameters of the decomposition of the homography. Our experiments also support Hartley's claim that the DLT algorithm works well with point normalization and fails for the un-normalized data. As we have seen from our results, for the DLT with point normalization the primary parameter effecting the accuracy and robustness of the algorithm is d . To get solutions with small probability of failures we need at least eight points. It also compares well with the optimal GS algorithm for larger d . Based on the error characterization in terms of parameters of decomposition of homography, further work can be done to revise and improve the DLT algorithm to account for the minimization of the average fit error.

References

- (1) R. Szeliski, Image mosaicing for tele-reality applications, *Workshop on Applications of Computer Vision*, 1994.
- (2) P. Heckbert, Fundamentals of texture mapping and image warping, *Master's thesis*, U. C. Berkeley, June 1989.
- (3) S. E. Chen, QuickTime VR – an image-based approach to virtual environment navigation, *Computer Graphics (SIGGRAPH'95 Proceedings)*, pages 29–38, August 1995.
- (4) S. Mann and R. Picard, Virtual bellows: Constructing high quality stills from video, *IEEE Conference on Image Processing*, November 1994.
- (5) Anders Heyden, Reconstruction from Multiple Images by means of using Relative Depths, *International Journal of Computer Vision*, Volume 24, no. 2, p. 155-161, 1997
- (6) Abdel-Aziz, Y. I. & Karara, H. M. (1971), Direct Linear transformation into object space coordinates in close-range photogrammetry, *Proc. Symposium on Close-Range Photogrammetry*, Urbana, Illinois, p. 1-18.
- (7) Faugeras, O. D. & Toscani, G. (1987), Camera calibration for 3D computer vision, *Proc. International Workshop on Industrial Applications of Machine Vision and Machine Intelligence*, Silken, Japan, p.240-247.

- (8) Richard I. Hartley, "In defense of Eight Point Algorithm", *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 6, June 1997
- (9) R.I. Hartley, "Minimizing Algebraic Distance," *Proc. DARPA Image Understanding Workshop*, 1997.
- (10) F.L. Bookstein, "Fitting Conic Sections to Scattered Data," *Computer Graphics and Image Processing*, vol. 9, pp. 56–71, 1979.
- (11) Hatze, H. (1988). High-precision three-dimensional photogrammetric calibration and object space reconstruction using a modified DLT-approach. *J. Biomechanics* 21, 533-538.
- (12) Marzan, G.T. & Karara, H.M. (1975), A computer program for direct linear transformation solution of the collinearity condition, and some applications of it, *Proceedings of the Symposium on Close-Range Photogrammetric Systems*, pp. 420-476, Falls Church, VA: American Society of Photogrammetry.
- (13) Robert Kaucic, Nicolas Dano and Richard Hartley, Plane-based Projective Reconstruction, *International Conference on Computer Vision*, p.420-427, 2001
- (14) Charles Loop and Zhengyou Zhang, Computing Rectifying Homographies for Stereo Vision, *Computer Vision and Pattern Recognition-Volume 1*, p.1125, 1999
- (15) Lihi Zelnik-Manor, Michal Irani, Multiview constraints on homographies, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 24, Issue 2, February 2002

- (16) W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, England: Cambridge Univ. Press, 1988.
- (17) K.E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed. New York: John Wiley and Sons, 1989.
- (18) Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge University, 2000
- (19) Graziano Chesi, Andrea Garulli, Antonio Vicino, Roberto Cipolla, Estimating the Fundamental Matrix via a constrained Least-Squares: a Convex approach, *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, March 2002

Appendix A

Computer code

1. Main program: Datary

```
function [Errors,Esumsq, Counts, OLcounts] = datary(varargin);

%The objective of this program is to first compute the homography between point
features in two images and then to test the solution accuracy by varying the amount of
noise, number of point correspondences

count = 0;
outliercount = 0;
total =0;
if (length(varargin)>0)
    points=varargin{1};
else
    points= 4:1:80;
end
if (length(varargin) > 1)
    length = varargin{2};
else
    levels= 2;
end
totalerrsquare = 0;
global MULT N NLEVEL

Errors= [];
Esumsq=[];
Counts = [];
OLcounts = [];
MULT= 100;

for N = points
    for NLEVEL = levels
        for q = 1:l

            [homog,decompvals] = decomposition(1000);
            s = decompvals{2};
```

```

b = decompvals{4};

for w = 1:100

    %Calculate random image points
    [X1,Y1] = randata(homog);

    %Obtain noisy image points
    [X1n,Y1n]= datanoise(X1,Y1);

    %Scale the noisy image points
    [XS,YS,T,T11] = scalemattry(X1n,Y1n);

    %Calculate noisy homography
    Hnoi=prehomog(XS,YS,T,T11);

    %Calculate the noisy image point from the noisy homography
    Ynoi=Hnoi*reshape(X1n,3,N);

    %Obtain the inhomogenous noisy and random image points
    Ynoiimage = Ynoi(1,:)./Ynoi(3,:);Ynoi(2,:)./Ynoi(3,:);
    Yimage = [Y1(1,:)./Y1(3,:);Y1(2,:)./Y1(3,:)];

    %Calculate the error level between these image points
    errlevel = (sum(sqrt(sum([(Ynoiimage-Yimage).^2]))))/N;
    errlevel = errlevel / (NLEVEL*s*sqrt(b^2+1));
    if (errlevel > 10 * NLEVEL)
        outliercount = outliercount + 1;
    else
        totalerrsquare = totalerrsquare + errlevel^2;
        total = total + errlevel;
    end
    count = count +1;

end % for all random point sets
end % for all choices of H
%save the errors and counts
Counts(find(levels==NLEVEL),find(points==N))= count;
OLcounts(find(levels==NLEVEL),find(points==N))= outliercount;
Errors(find(levels==NLEVEL),find(points==N))= total;
Esumsq(find(levels==NLEVEL),find(points==N))= totalerrsquare;

```

```

    total = 0;
    totalerrsquare=0;
    count = 0;
    outliercount = 0;
end      % for choices of noise levels

end      % for choices of N

%Plot noise level vs the average error
figure(1)
gcounts = Counts(:,1) - OLcounts(:,1);
ebar = sqrt((Esumsq(:,1) - (Errors(:,1).^2)./gcounts)./gcounts);
errorbar(levels,Errors(:,1)./gcounts,ebar,'-b*')
title('Plot of the average positional error vs noiselevel','FontSize',12)
hold on

figure(2)
gcounts = Counts(1,:) - OLcounts(1,:);
ebar = sqrt((Esumsq(1,:) - (Errors(1,:).^2)./gcounts)./gcounts);
errorbar(points,Errors(1,:)/gcounts,ebar,'-r*')
title('Plot of the average positional error vs number of points','FontSize',12)
hold on

figure(3)
plot(points,100*OLcounts(1,:)/Counts(1,:),'-r*')
title('Plot of the percent outliers vs number of points','FontSize',12)
hold on

```

2. Decomposition

```

function [homog, vals] = decomposition(d)

global MULT
T=[1 0 (MULT/2) ; 0 1 (MULT/2) ; 0 0 1];
Tinv=inv(T);

theta = rand(1)*2*pi;
R=[cos(theta) -sin(theta);sin(theta) cos(theta)];
s = 1.5*rand(1) + 0.5;

```

```

%a = (1.5*rand(1) + 0.5);
a=1;
%b = (2*rand(1) - 1);
b=0;
tr = [MULT*(rand(1) - 0.5) ; MULT*(rand(1) - 0.5)];
K = [a b ; 0 1];
K = K/sqrt(det(K));
phi = rand(1)*2*pi;
v1= -cos(phi)/d;
v2= -sin(phi)/d;
%v1=0;
%v2=0;
v=1;

homog = Tinv * [s*R tr;0 0 1] * [K [0;0] ; 0 0 1] * [1 0 0; 0 1 0; v1 v2 v] * T;

vals = {theta,s,a,b,phi};

```

3. Randata

```

function [X1,Y1] = randata(homog)
global N MULT

X1 = reshape([MULT*rand(2,N);ones(1,N)],3*N,1);
Y11 = homog*reshape(X1,3,N);
Y1 = [Y11(1,:)./Y11(3,:);Y11(2,:)./Y11(3,:); ones(1,N)];

```

4. Datanoise

```

function [X1n,Y1n]= datanoise(X,Y)
%To call the 2 sets of image points from data.m file
global N NLEVEL

%Noisy image points

a11=[NLEVEL*randn(2,N);zeros(1,N)];
b11=[NLEVEL*randn(2,N);zeros(1,N)];

X1n = X + reshape(a11,3*N,1);
Y1n = reshape(reshape(Y,3,N)+b11,3*N,1);

```

5. Scalemattry

```
function [XS,YS, T, T11] = scalemattry(X,Y)

%Normalize the 2D points
%The centroid of the points xi should be at origin, which will give 't'
%The average distance from origin is Sqrt(2), which gives 's'

global N
%call data function
%[X,Y]=data

%Sum(s*(xi-t)=(0,0)^T
%t = (sum(X'))/N

trone = (1/N)*(sum((reshape(X, 3, N))))';
trtwo = (1/N)*(sum((reshape(Y, 3, N))))';

ts=trone*ones(1,N);
tp=trtwo*ones(1,N);

%1/N*sum(s*(xi-t)^T*s*(xi-t))

P = sum((sum(((reshape(X, 3,N)-ts).^2))))';
P11 = sum((sum(((reshape(Y, 3,N)-tp).^2))))');

s=sqrt(2*N/P);

s11=sqrt(2*N/P11);

R=[1 0;0 1];
T = [s*R -s*[ts(1,1);ts(2,1)];0 0 1];
T11=[s11*R -s11*[tp(1,1);tp(2,1)];0 0 1];

%Scaled X and Y
XS=reshape(T*reshape(X,3,N), 3*N,1);
YS=reshape(T11*reshape(Y,3,N),3*N,1);
```

6. Prehomog

```

function H = prehomog(X,Y,T,T11)

global N
% TO TEST SOLUTION ACCURACY BY VARYING AMOUNT OF NOISE FOR
4PT CORRESPONDENCES WITH PRESCALING

%Apply Direct Linear Algorithm ie calculate Ai for each pt correspondence

for k = 1 : N
    A((2*k-1):(2*k),:)= [0 0 0 -1*[X((k+2*(k-1)),1); X((k+2*(k-1)+1),1); X((k+2*(k-1)+2),1)]' Y((k+2*(k-1)+1),1)*[X((k+2*(k-1)),1); X((k+2*(k-1)+1),1); X((k+2*(k-1)+2),1)]'];
    1*[X((k+2*(k-1)),1); X((k+2*(k-1)+1),1); X((k+2*(k-1)+2),1)]' 0 0 0 -Y((k+2*(k-1)+1),1)*[X((k+2*(k-1)),1); X((k+2*(k-1)+1),1); X((k+2*(k-1)+2),1)]'];
end

[m,n] = size(A) ;

if (m<n)

%Since m < n in A,add one row of zeros so that An is a 9x9 matrix
A=[A;zeros(1,9)];

%An_square=udv^T
[U,D,V] = svd(A);

else

    %An_square=udv^T
    [U,D,V] = svd(A);
end

%Hn is the last column of transpose of V
O=V;
h=O(:,9);

%fprintf(1,'D(9,9,) is %f\n',D(9,9))

Hn_invert = reshape(h,3,3);
Hn = Hn_invert';

```

```
%Denormalize Hn
H = inv(T11)*Hn*T;
```

4. Datetrywithgs

```
function [Errors,Esumsq, Counts, OLcounts] = datatrywithgs(varargin);
```

```
%The objective of this program is to first compute the homography between point
features
%in two images and then to test the solution accuracy by varying the amount of noise,
number
%of point correspondences and also with and without prescaling
```

```
count = 0;
outliercount = 0;
total =0;
if (length(varargin)>0)
    points=varargin{1};
else
    points= 4:1:40;
end
if (length(varargin) > 1)
    levels = varargin{2};
else
    levels= 2;
end
totalerrsquare = 0;
global MULT N NLEVEL
```

```
Errors= [];
Esumsq=[];
Counts = [];
OLcounts = [];
MULT= 100;
```

```
for N = points
    for NLEVEL = levels
        for q = 1:10
```

```
            [homog,decompvals] = decomposition(1000);
            s = decompvals{2};
```

```

b = decompvals{4};
a = decompvals{3};

for w = 1:1

    %Calculate random image points
    [X1,Y1] = randata(homog);

    %Obtain noisy image points
    [X1n,Y1n]= datanoise(X1,Y1);

    %Scale the noisy image points
    [XS,YS,T,T11] = scalemattry(X1n,Y1n);

    %Calculate noisy homography
    Hnoi=prehomog(XS,YS,T,T11);

    state = [reshape(Hnoi,9,1) ; X1n(1:(2*N))];

    objnew = errorfun(state,X1n,Y1n);
    state = lsqnonlin('errorfun', state,[],[],...
        optimset('LevenbergMarquardt','on'),X1n,Y1n);
    options = optimset('TolX',0.0001)
    optnew = optimset(options,'TolX',0.0001);
    res = errorfun(state,X1n,Y1n);

    %Calculate the error level between these image points
    errlevel = sqrt(sum(res.^2))/N;

    if (errlevel > 10* NLEVEL)
        outliercount = outliercount + 1;
    else
        totalerrsquare = totalerrsquare + errlevel^2;
        total = total + errlevel;
    end
    count = count +1;

end % for all random point sets
end % for all choices of H
%save the errors and counts
Counts(find(levels==NLEVEL),find(points==N))= count;

```

```

    OLcounts(find(levels==NLEVEL),find(points==N))= outliercount;
    Errors(find(levels==NLEVEL),find(points==N))= total;
    Esumsq(find(levels==NLEVEL),find(points==N))= totalerrsquare;
    total = 0;
    totalerrsquare=0;
    count = 0;
    outliercount = 0;
end      % for choices of noise levels

end      % for choices of N

%Plot noise level vs the average error
figure(1)
gcounts = Counts(:,1) - OLcounts(:,1);
ebar = sqrt((Esumsq(:,1) - (Errors(:,1).^2)./gcounts)./gcounts);
errorbar(levels,Errors(:,1)./gcounts,ebar,'-b*')
title('Plot of the average positional error vs noiselevel','FontSize',12)
hold on

figure(2)
gcounts = Counts(1,:) - OLcounts(1,:);
ebar = sqrt((Esumsq(1,:) - (Errors(1,:).^2)./gcounts)./gcounts);
errorbar(points,Errors(1,:)/gcounts,ebar,'-r*')
title('Plot of the average positional error vs number of points','FontSize',12)
hold on

figure(3)
plot(points,100*OLcounts(1,:)/Counts(1,:),'-r*')
title('Plot of the percent outliers vs number of points','FontSize',12)
hold on

```

5. Prehomnosc

```
function H = prehomognosc(X,Y)
```

```
global N
```

```

%CASE 1: TO TEST SOLUTION ACCURACY BY VARYING AMOUNT OF NOISE
FOR 4PT CORRESPONDENCES
%WITH PRESCALING
%[X,Y]=data

```

```

%[T,T11, XS,YS] = scalemattry[X,Y]

%Apply Direct Linear Algorithm ie calculate Ai for each pt correspondence
for k = 1 : N
    A((2*k-1):(2*k),:)= [0 0 0 -1*[X((k+2*(k-1)),1); X((k+2*(k-1)+1),1); X((k+2*(k-1)+2),1)]'
    Y((k+2*(k-1)+1),1)*[X((k+2*(k-1)),1); X((k+2*(k-1)+1),1); X((k+2*(k-1)+2),1)]';
    1*[X((k+2*(k-1)),1); X((k+2*(k-1)+1),1); X((k+2*(k-1)+2),1)]' 0 0 0 -Y((k+2*(k-1)+1),1)*[X((k+2*(k-1)),1); X((k+2*(k-1)+1),1); X((k+2*(k-1)+2),1)]';
end

[m,n] = size(A) ;

if (m<n)

%Since m < n in A,add one row of zeros so that An is a 9x9 matrix
A=[A;zeros(1,9)];

%An_square=udv^T
[U,D,V] = svd(A);

else

    %An_square=udv^T
    [U,D,V] = svd(A);
end

%Hn is the last column of transpose of V
O=V;
h=O(:,9);

Hn_invert = reshape(h,3,3);
H = Hn_invert';

```

6. Datatrjnps

```
function [Errors,Esumsq, Counts, OLCcounts] = datatrjnps(varargin);
```

```
%The objective of this program is to first compute the homography between point features
```

%in two images and then to test the solution accuracy by varying the amount of noise,
number
%of point correspondences and also with and without prescaling

```
count = 0;  
outliercount = 0;  
total = 0;  
if (length(varargin)>0)  
    points=varargin{1};  
else  
    points= 4:1:40;  
end  
if (length(varargin) > 1)  
    length = varargin{2};  
else  
    levels= 2;  
end  
totalerrsquare = 0;  
global MULT N NLEVEL
```

```
Errors= [];  
Esumsq=[];  
Counts = [];  
OLcounts = [];  
MULT= 100;
```

```
for N = points  
    for NLEVEL = levels  
        for q = 1:500
```

```
            [homog,decompvals] = decomposition(1/0);  
            s = decompvals{2};  
            b = decompvals{4};  
            a = decompvals{3};
```

```
            for w = 1:1
```

```
                %Calculate random image points  
                [X1,Y1] = randata(homog);
```

```
                %Obtain noisy image points
```

```

[X1n,Y1n] = datanoise(X1,Y1);

%Calculate homography without prescaling
Hnoi= prehomognosc(X1n,Y1n);

%Calculate the noisy image point from the noisy homography
Ynoi=Hnoi*reshape(X1n,3,N);

%Obtain the inhomogenous noisy and random image points
Ynoiimage = [Ynoi(1,:)/Ynoi(3,:);Ynoi(2,:)/Ynoi(3,:)];
Y1nm = reshape(Y1n,3,N);
Yimage = [Y1nm(1,:)/Y1nm(3,:);Y1nm(2,:)/Y1nm(3,:)];

%Calculate the error level between these image points
errlevel = (sum(sqrt(sum([(Ynoiimage-Yimage).^2]))))/N;
errlevel = errlevel / (NLEVEL*s*sqrt(a+(b^2+1)/a +2));

if (errlevel > 10 * NLEVEL)
    outliercount = outliercount + 1;
else
    totalerrsquare = totalerrsquare + errlevel^2;
    total = total + errlevel;
end
count = count +1;

end % for all random point sets
end % for all choices of H
%save the errors and counts
Counts(find(levels==NLEVEL),find(points==N))= count;
OLcounts(find(levels==NLEVEL),find(points==N))= outliercount;
Errors(find(levels==NLEVEL),find(points==N))= total;
Esumsq(find(levels==NLEVEL),find(points==N))= totalerrsquare;
total = 0;
totalerrsquare=0;
count = 0;
outliercount = 0;
end % for choices of noise levels

end % for choices of N

%Plot noise level vs the average error

```

```

figure(1)
gcounts = Counts(:,1) - OLcounts(:,1);
ebar = sqrt((Esumsq(:,1) - (Errors(:,1).^2)./gcounts)./gcounts);
errorbar(levels,Errors(:,1)./gcounts,ebar,'-b*')
title('Plot of the average positional error vs noiselevel','FontSize',12)
hold on


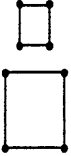


%Plot of the average positional error vs number of points
figure(2)
gcounts = Counts(1,:) - OLcounts(1,:);
ebar = sqrt((Esumsq(1,:) - (Errors(1,:).^2)./gcounts)./gcounts);
errorbar(points,Errors(1,:)./gcounts,ebar,'-r*')
title('Plot of the average positional error vs number of points','FontSize',12)
hold on

%Plot of the percent outliers vs number of points
figure(3)
plot(points,100*OLcounts(1,:)./Counts(1,:),'-r*')
title('Plot of the percent outliers vs number of points','FontSize',12)
hold on

```

APPENDIX B

Frequently occurring planar transformations and their geometric invariant properties

Group	Matrix	Distortion	Invariant properties
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratios of lengths, single. The circular points I, J
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g., centroids). The line at infinity, l_∞
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, order of contact: intersection (1 pt contact); tangency (2 pt contact); inflection (3 pt contact); tangent discontinuities and cusps, cross ratio (ratio of ratio of lengths)

source: Hartley and Zisserman (2000)

The table shown here depicts hierarchy of transformations, with Euclidean transformation being the simplest in terms of the degrees of freedom. A planar Euclidean transformation

has three degrees of freedom, one for rotation and two for translation. Therefore, we need at least two point correspondences to compute this transformation. In the similarity transformation there is an extra degree of freedom from isotropic scale factor, s . This transformation is also known as *equi-form* transform as it preserves “shape”. An affine transformation is a non-singular linear transformation followed by translation. A planar affine transformation has six degrees of freedom. Because affine transformation includes non-isotropic scaling, the similarity invariants of length ratios and angles between the lines are not preserved. Projectivities have eight degrees of freedom. The difference between the affine transformation and projective transformation is that the vector \mathbf{v} is not null for projectivity. This causes the non-linear effects of the projectivity.

Figure 4. Case 1 – Affine plots of DLT algorithm with point and error normalization

Figure 4a. Plot of the average fit error vs number of points

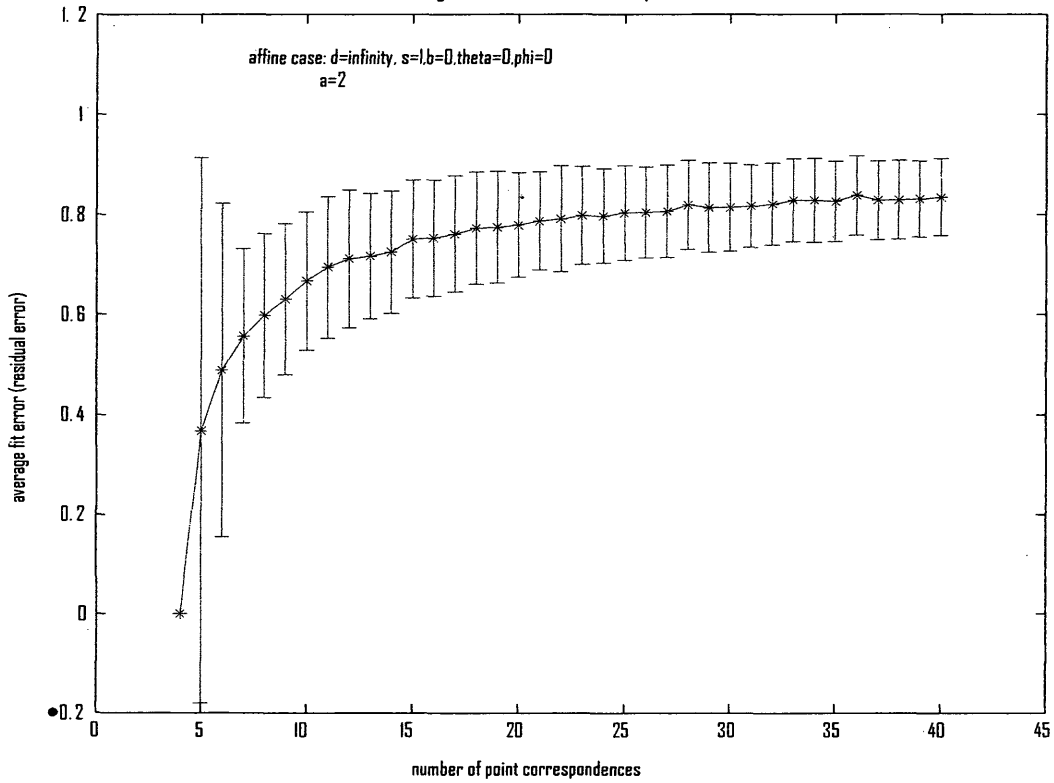


Figure 4b. Plot of the average fit error vs number of points

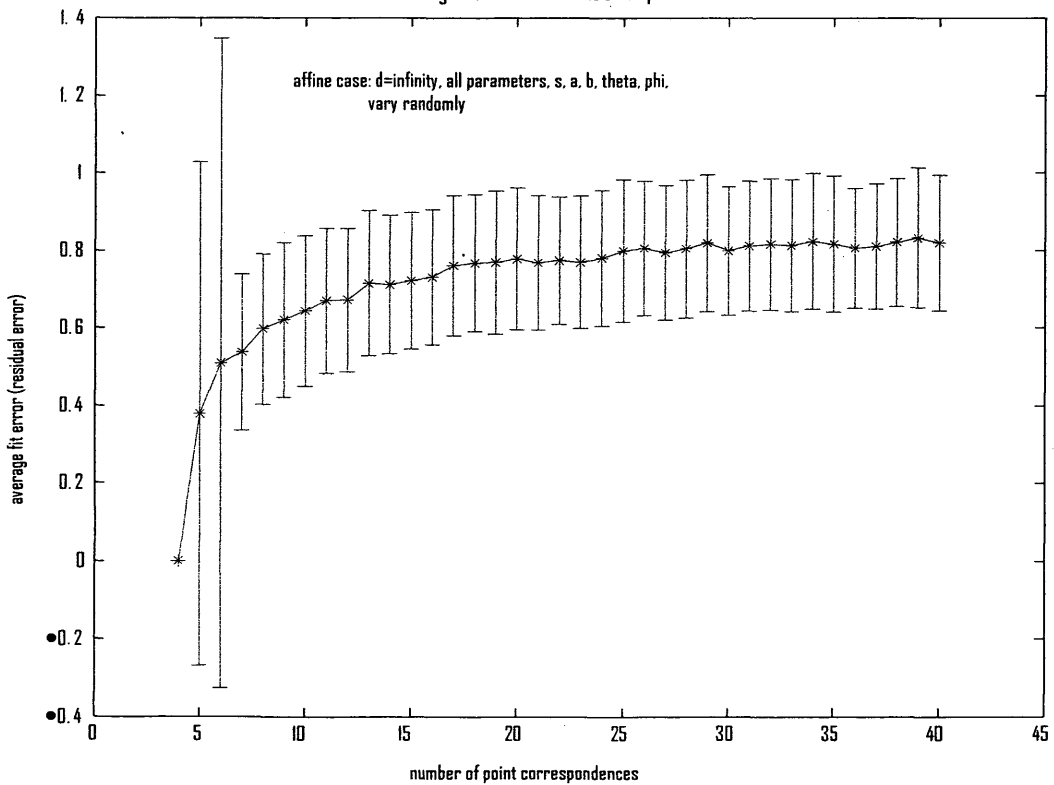


Figure 4c. Plot of the average fit error vs number of points

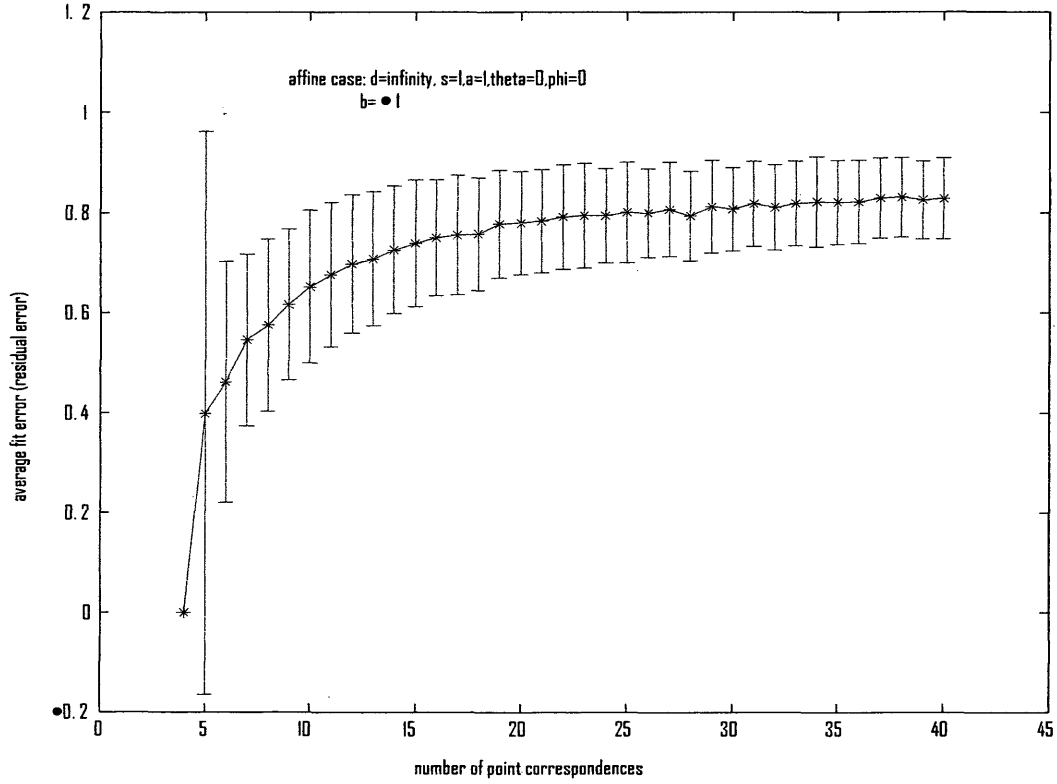


Figure 4d. Plot of the average fit error vs number of points

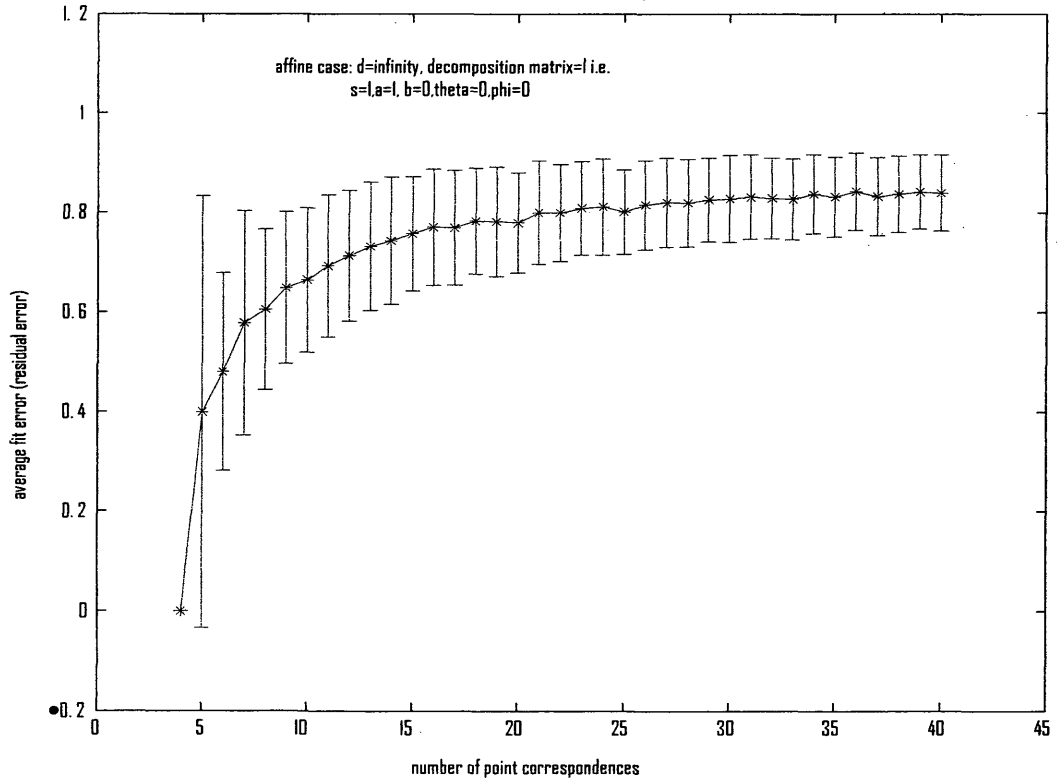


Figure 4e. Plot of the average fit error vs number of points

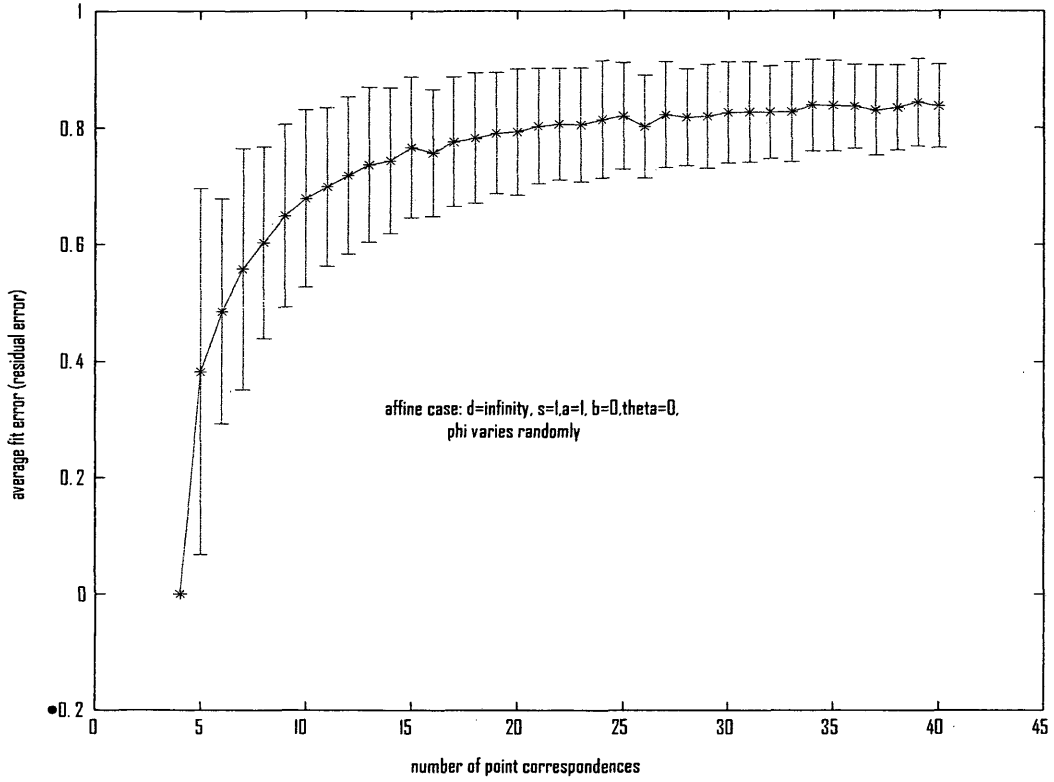


Figure 4f. Plot of the average fit error vs number of points

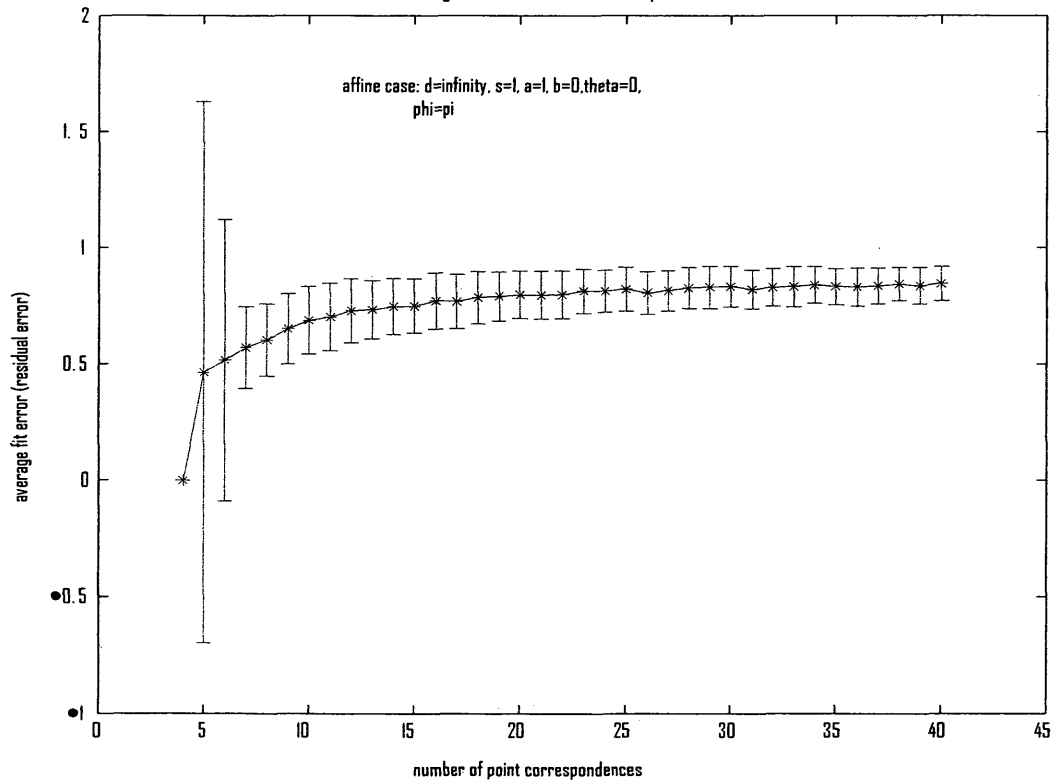


Figure 4g. Plot of the average fit error vs number of points

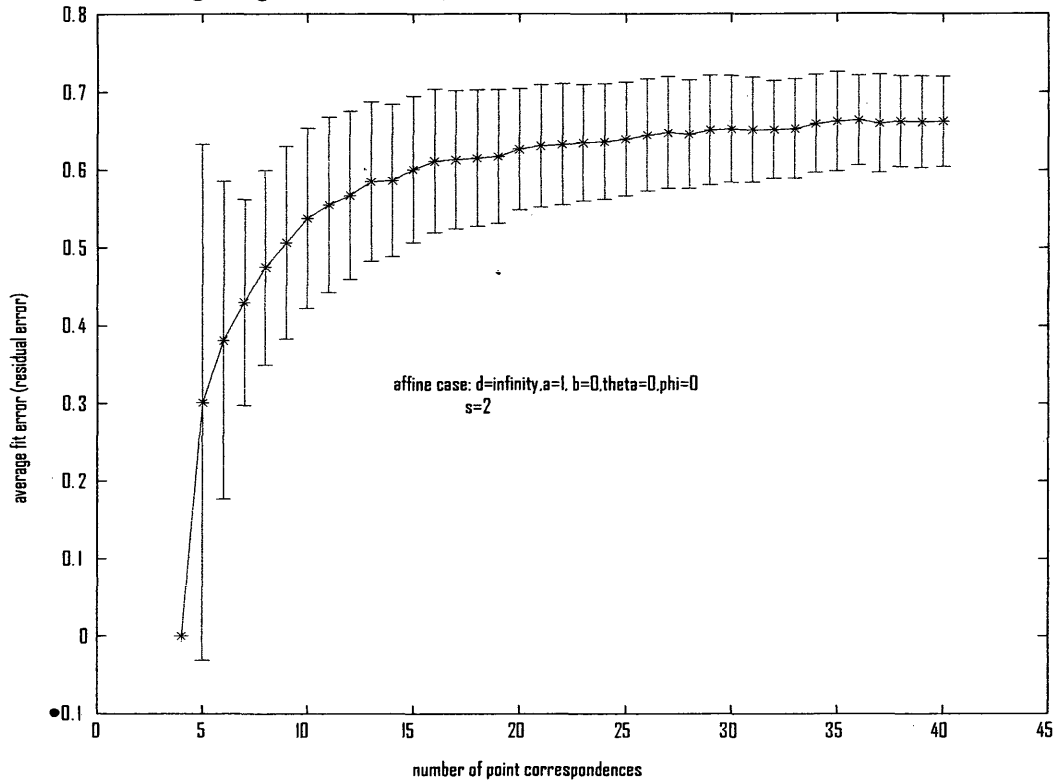


Figure 4h. Plot of the average fit error vs number of points

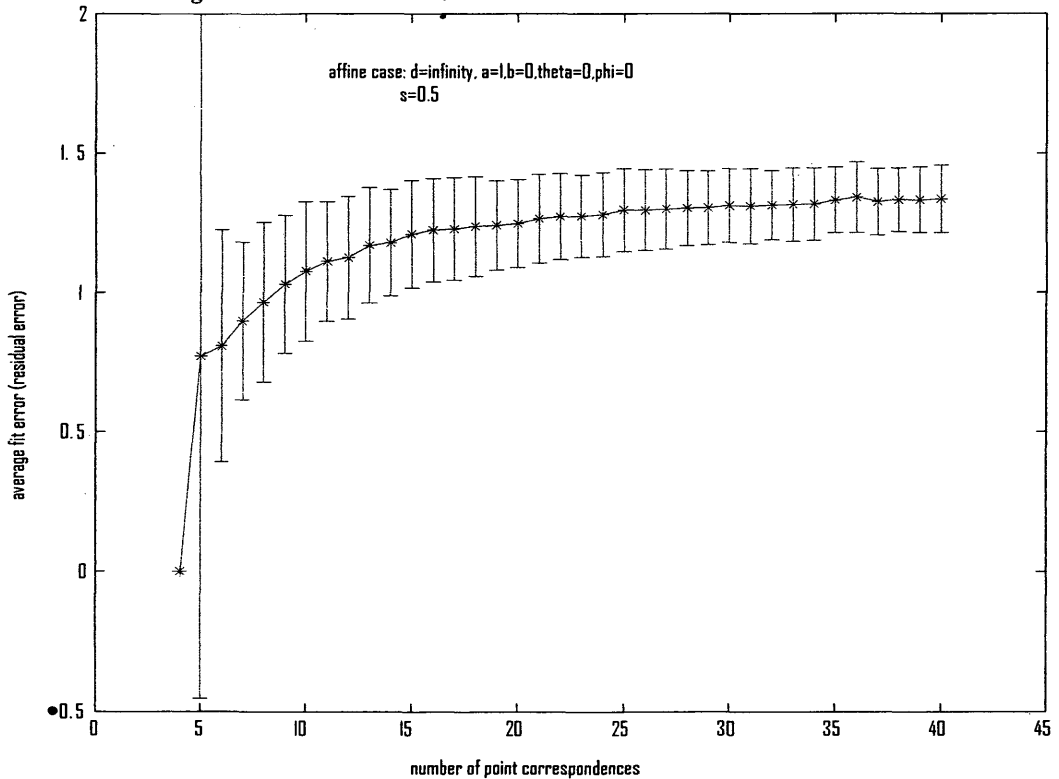


Figure 4i. Plot of the average fit error vs number of points

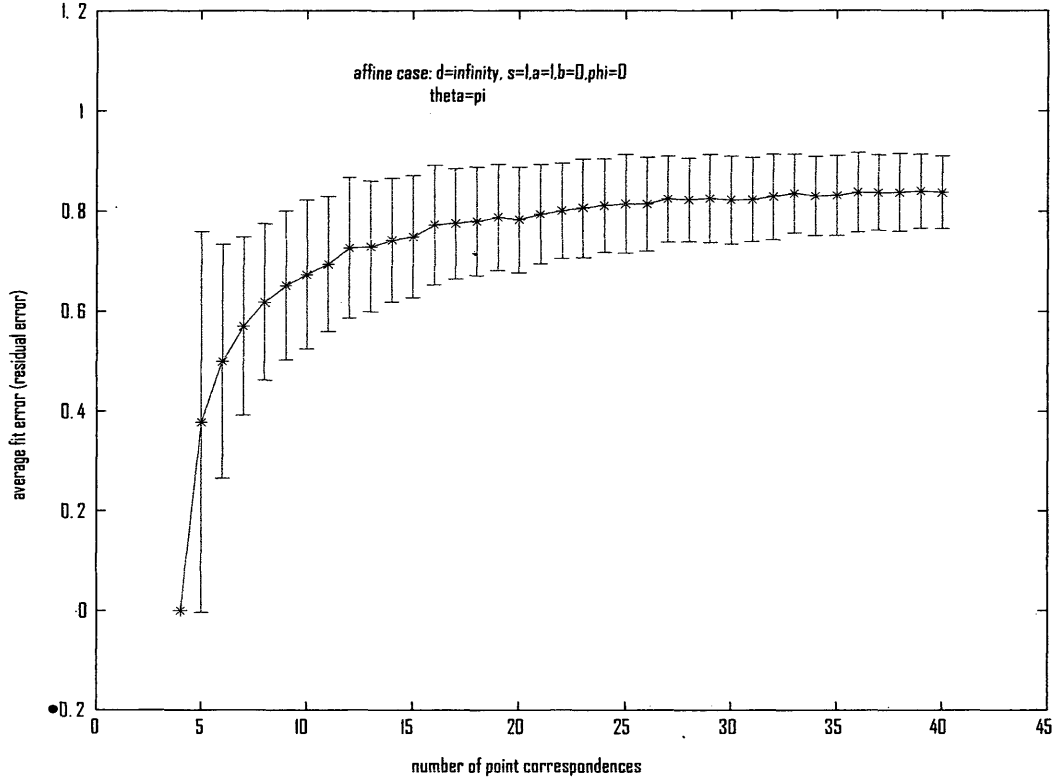


Figure 4j. Plot of the average fit error vs number of points

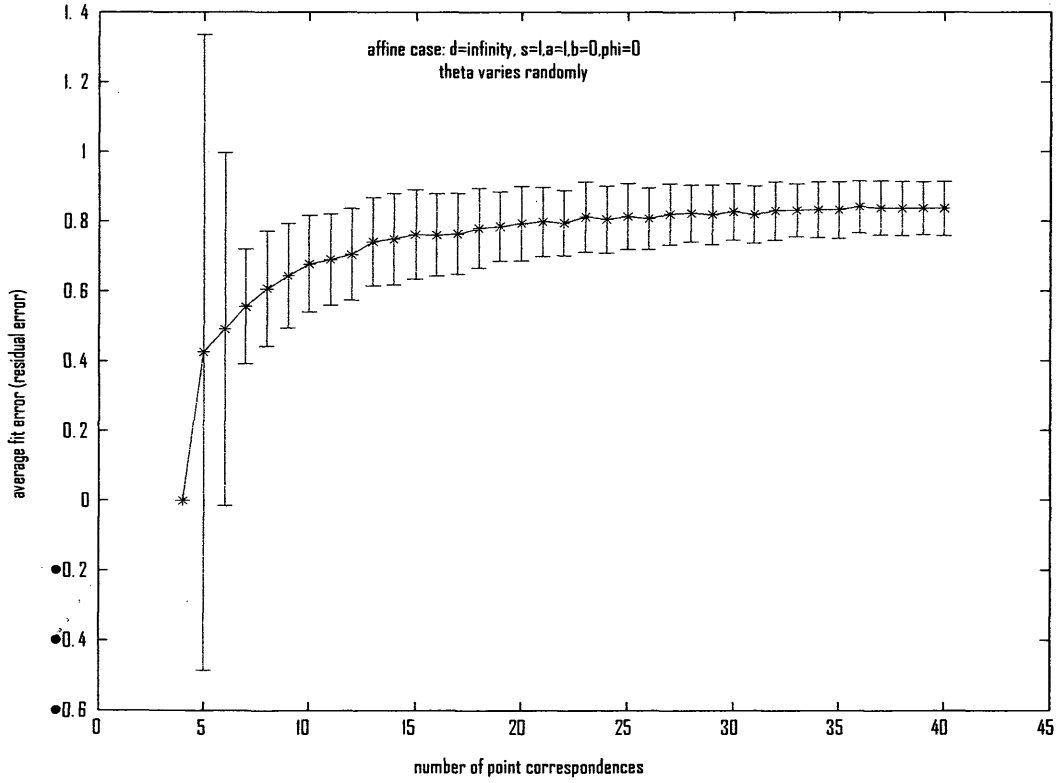


Figure 4k. Plot of the percent failures vs number of points

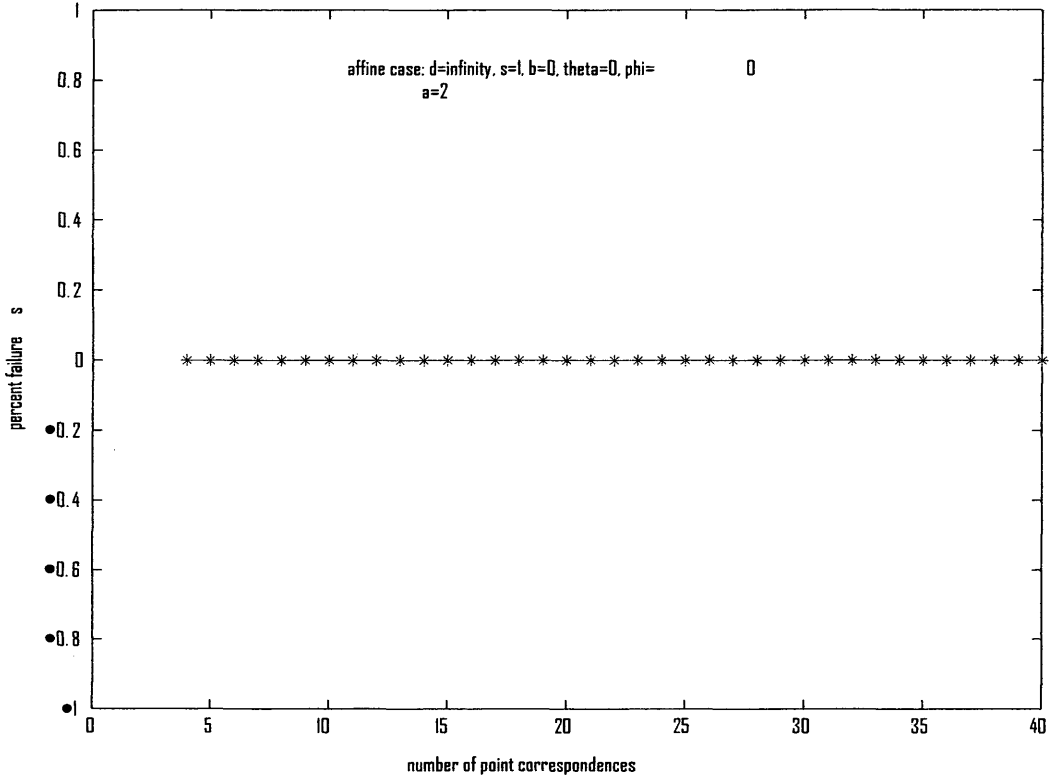


Figure 4l. Plot of the percent failures vs number of points

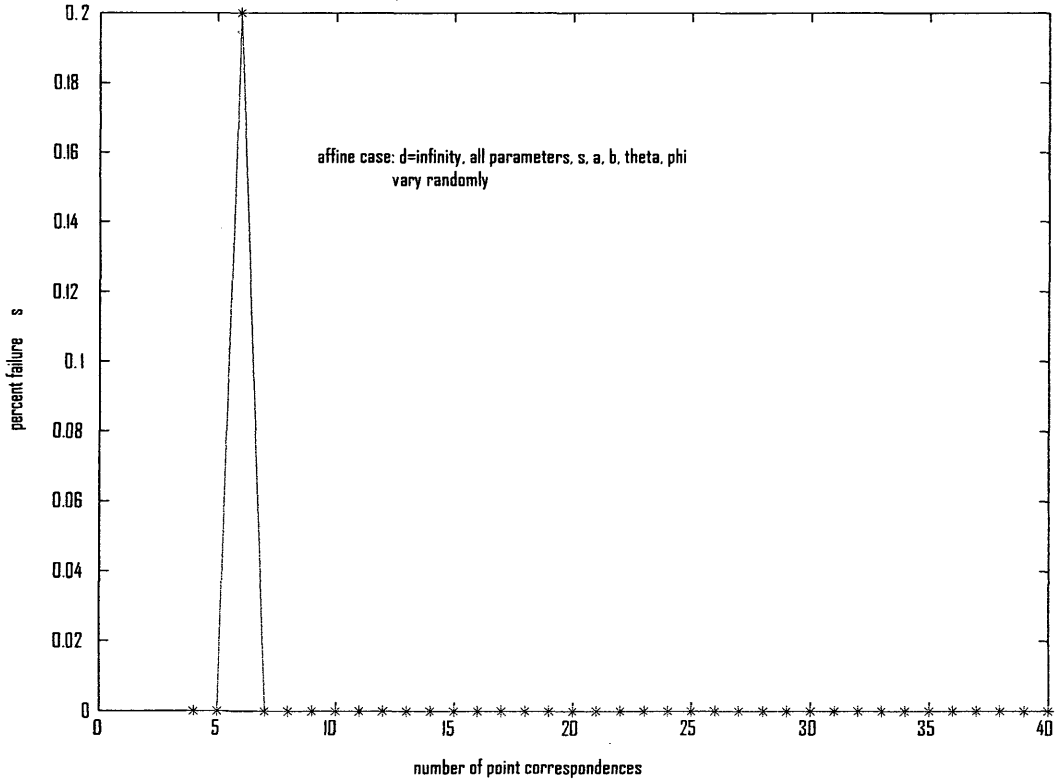


Figure 4m. Plot of the percent failures vs number of points

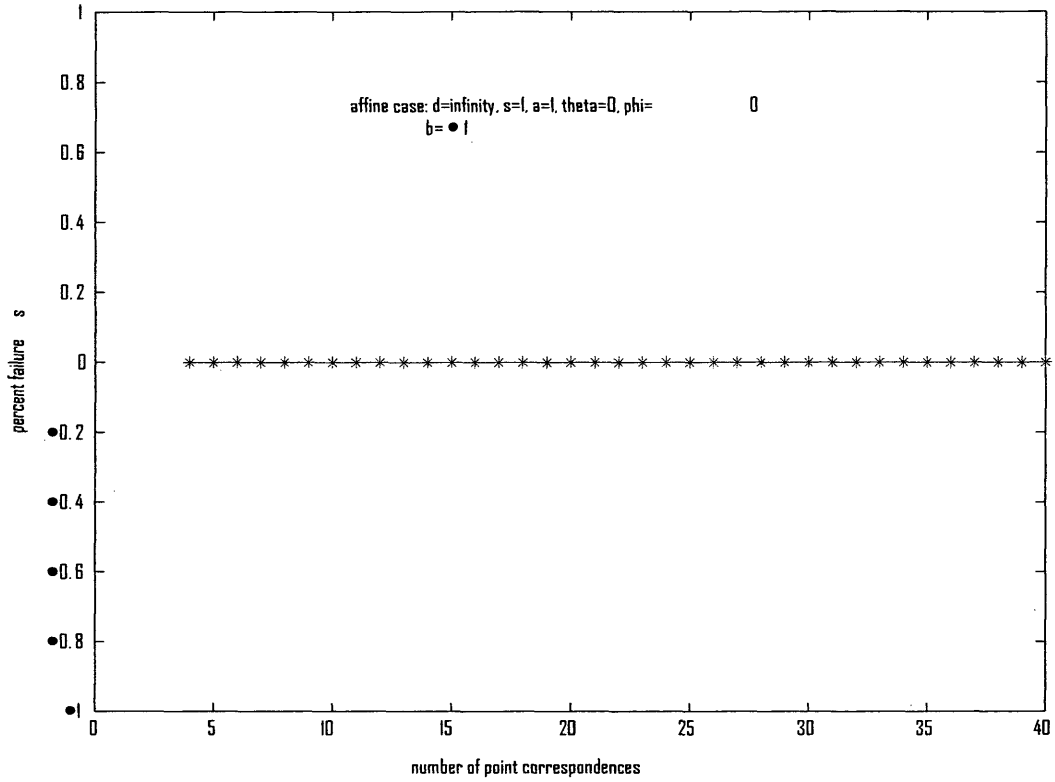


Figure 4n. Plot of the percent failures vs number of points

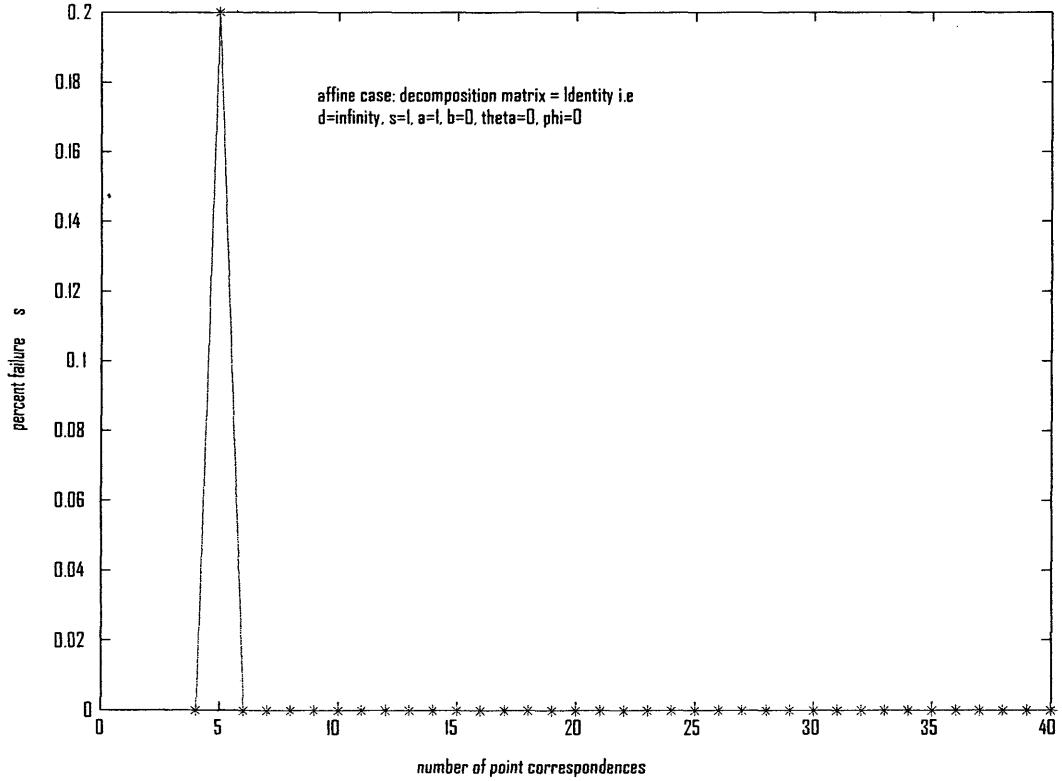


Figure 4o. Plot of the percent failures vs number of points

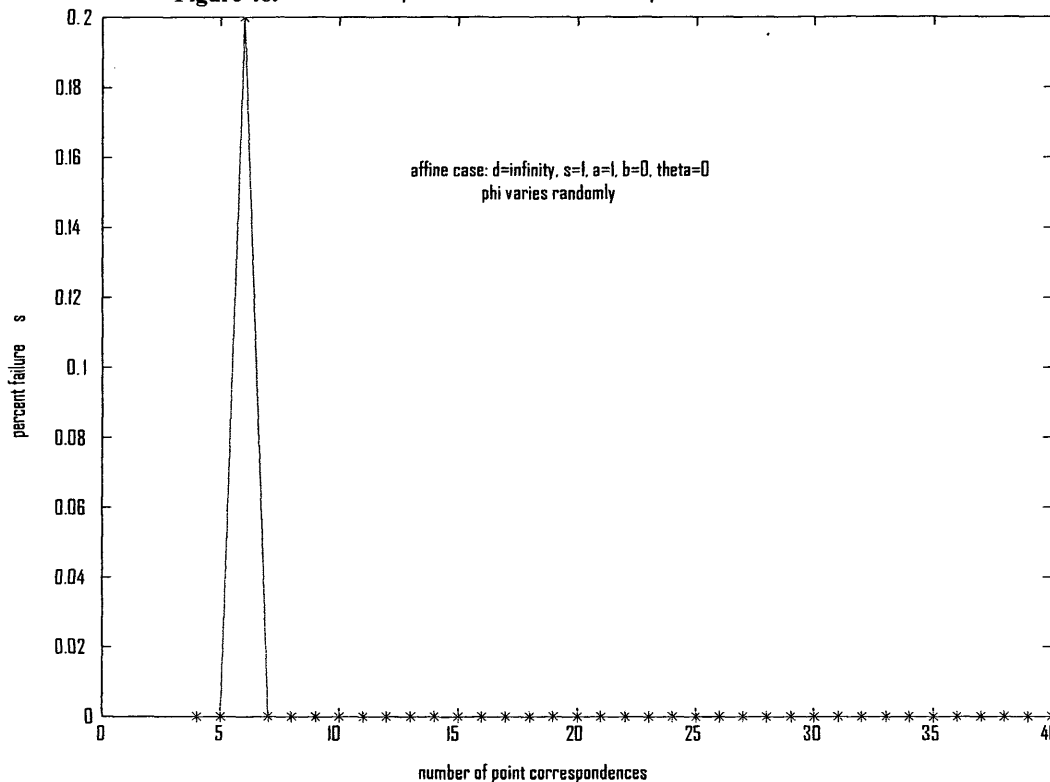


Figure 4p. Plot of the percent failures vs number of points

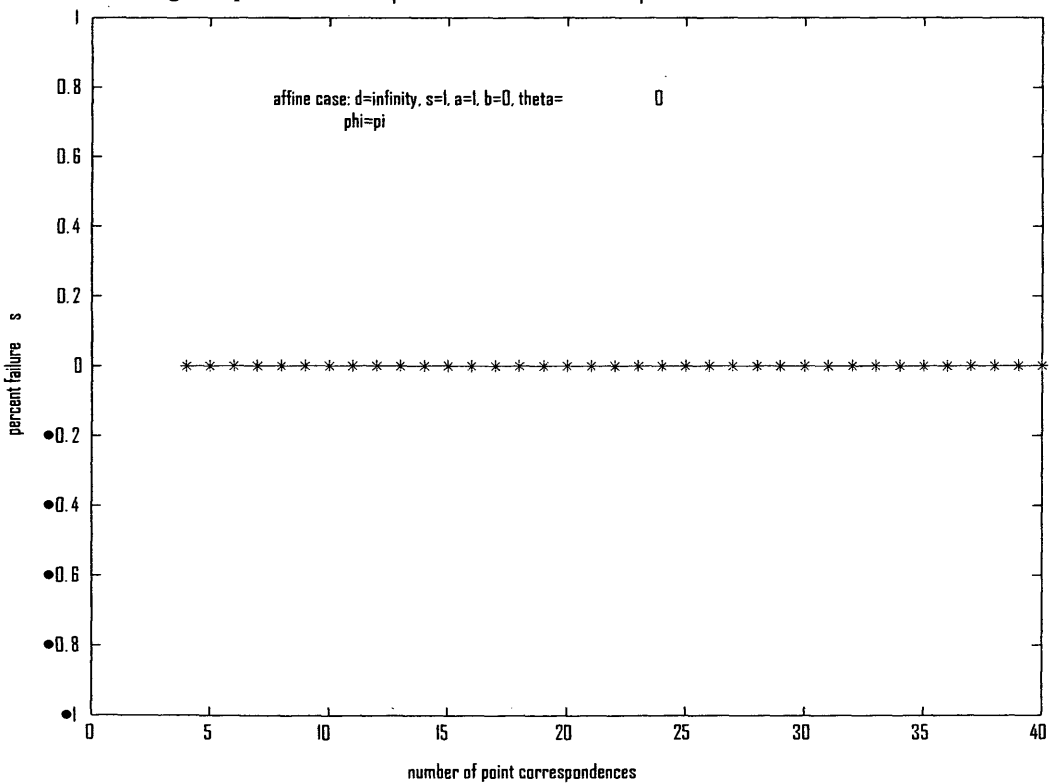


Figure 4q. Plot of the percent failures vs number of points

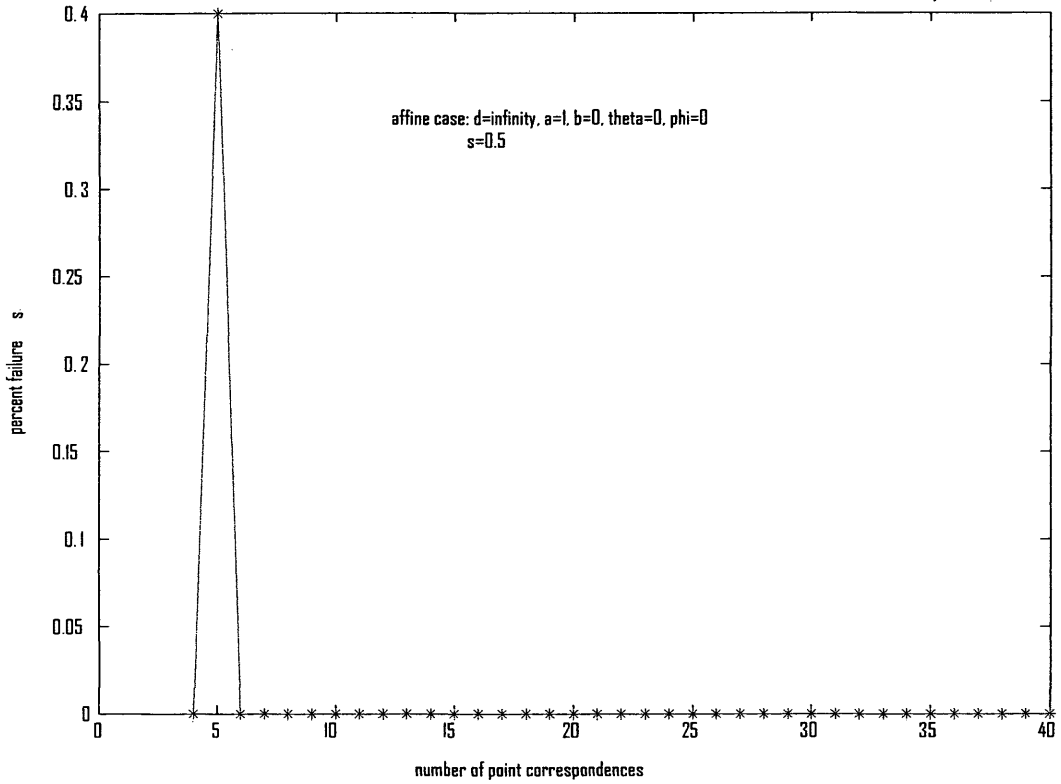


Figure 4r. Plot of the percent failures vs number of points

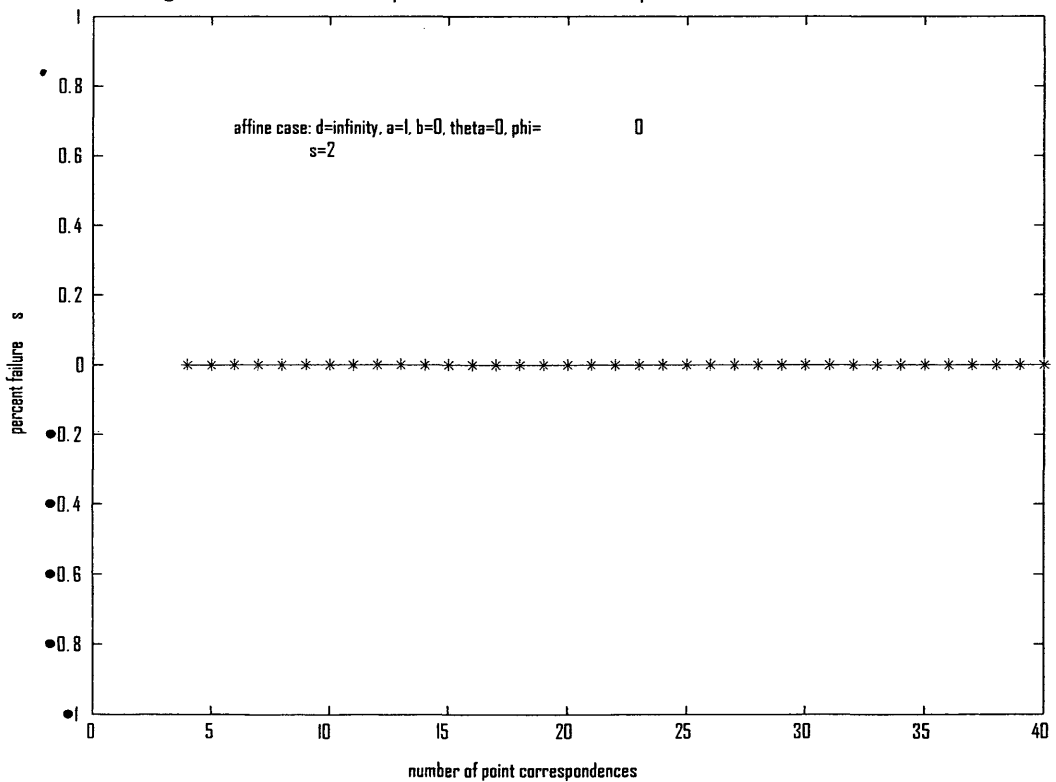


Figure 4s. Plot of the percent failures vs number of points

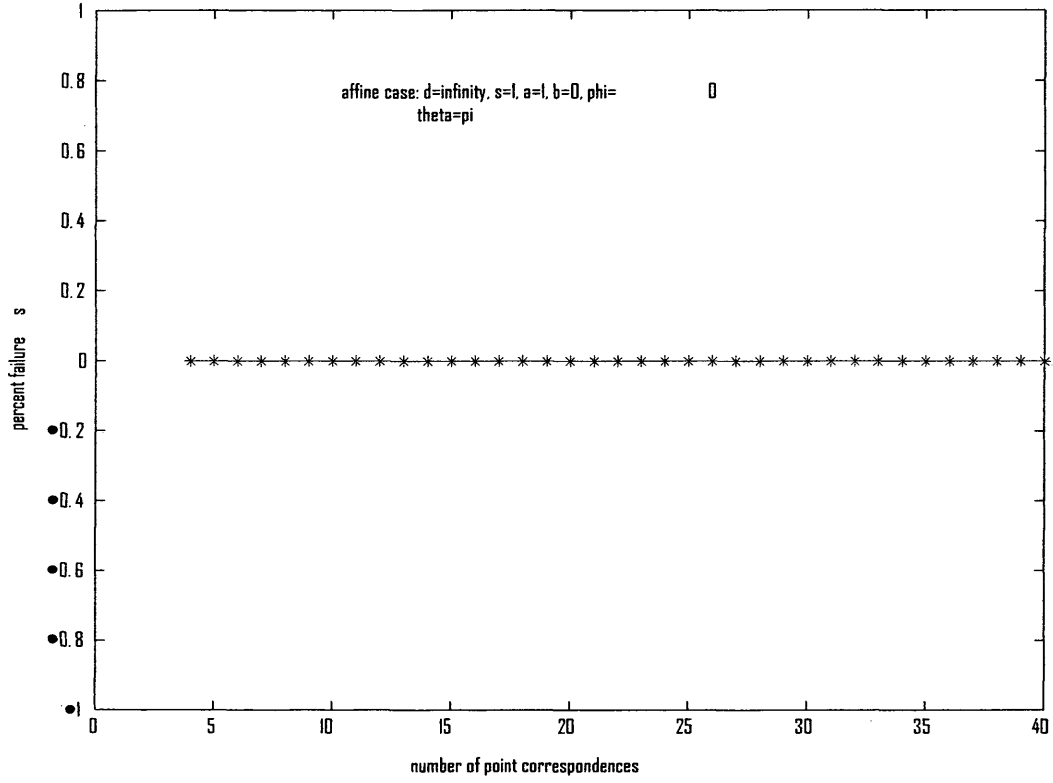


Figure 4t. Plot of the percent failures vs number of points

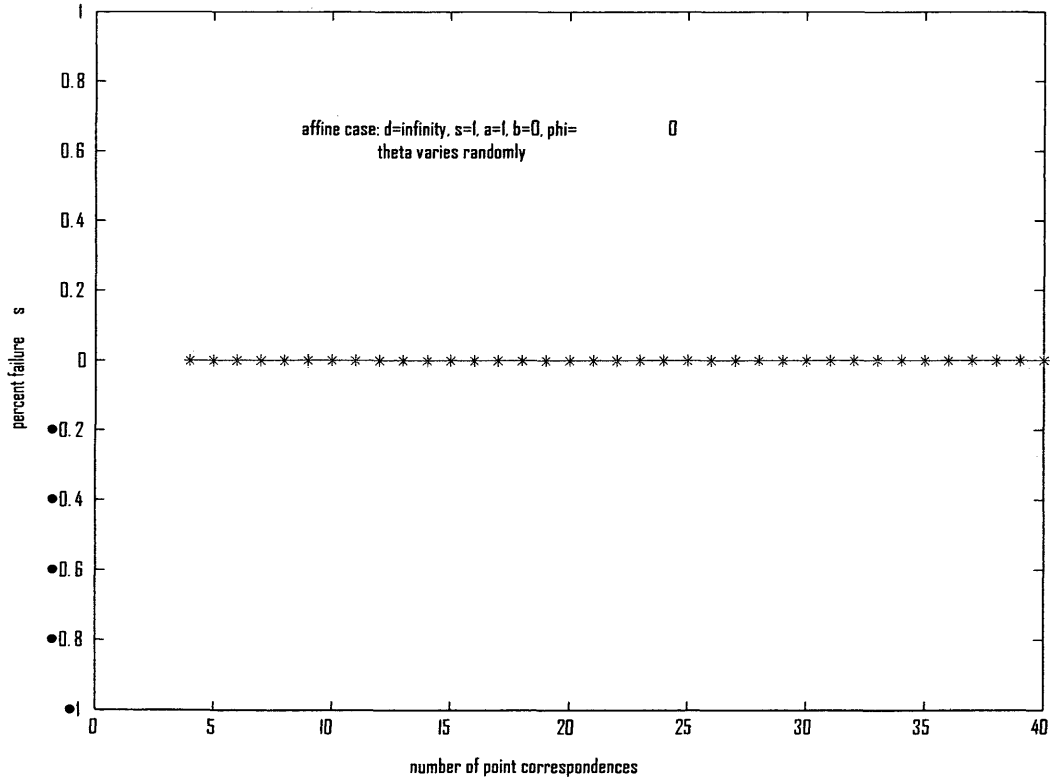


Figure 5. Case 2 -- Non-affine with $d=100$ with DLT algorithm

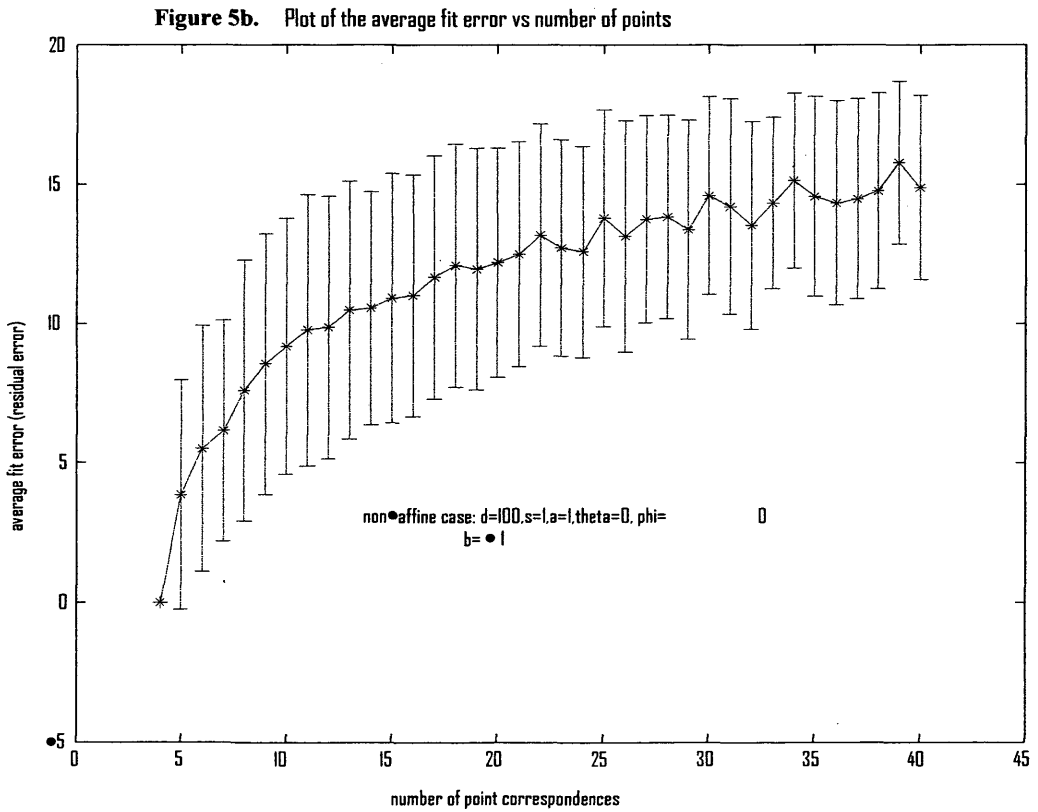
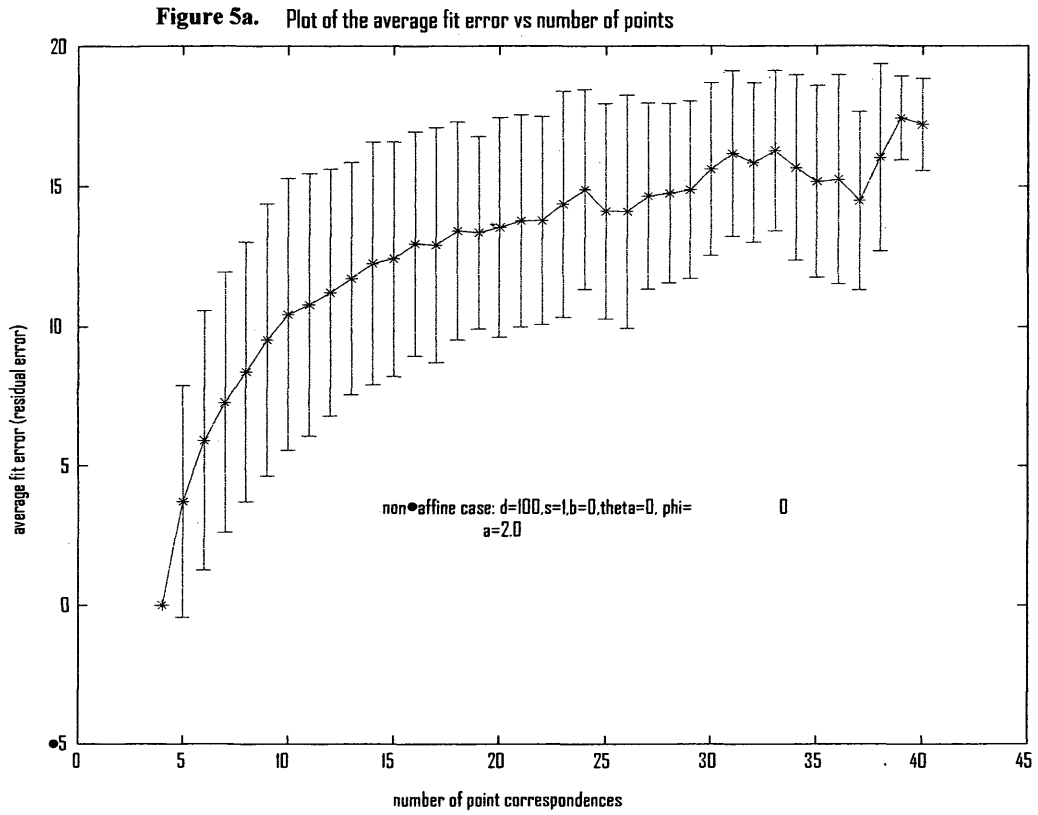


Figure 5c. Plot of the average fit error vs number of points

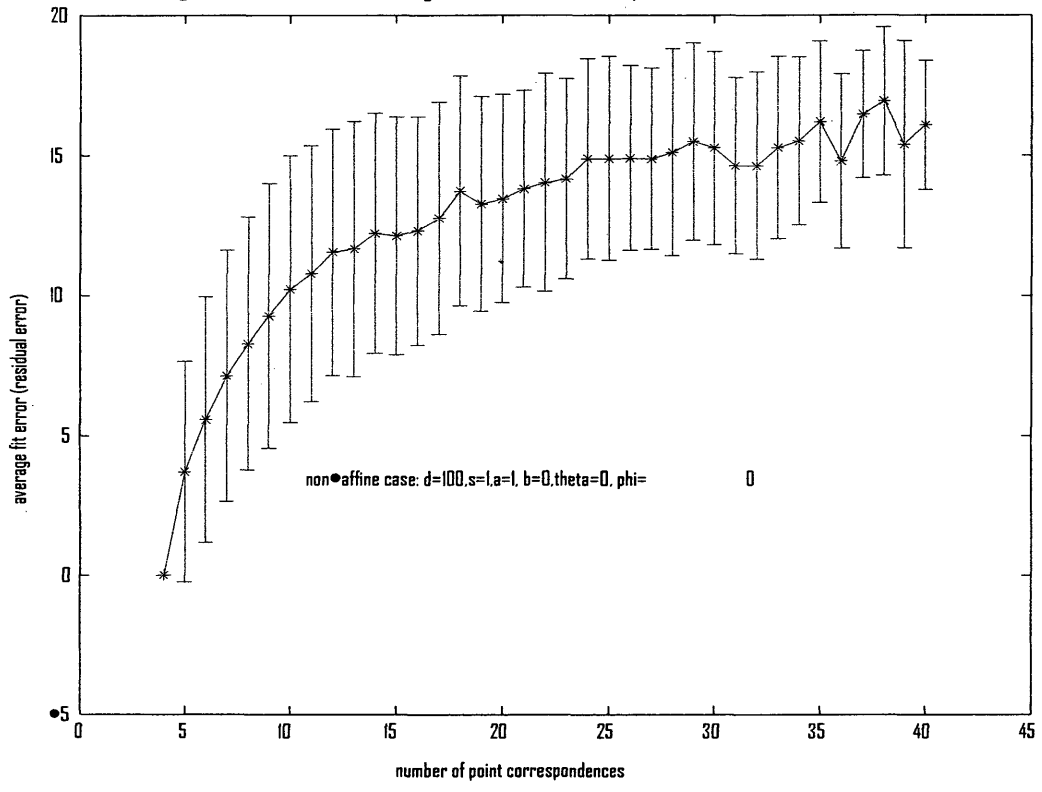


Figure 5d. Plot of the average fit error vs number of points

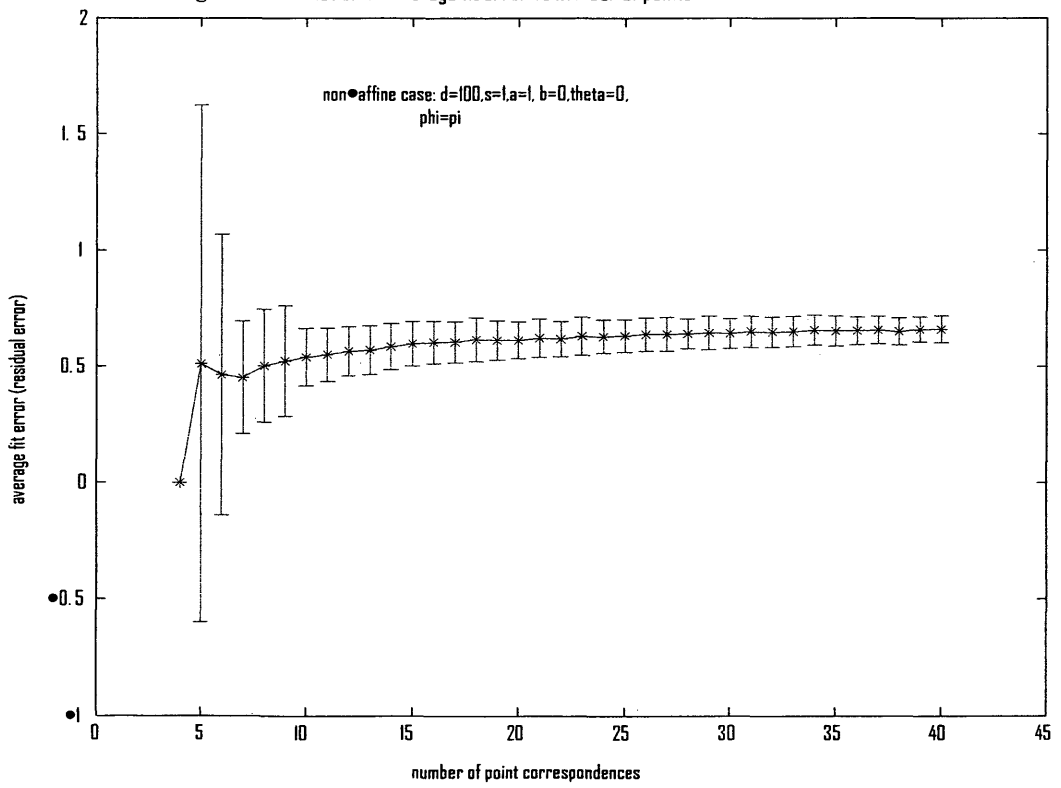


Figure 5e. Plot of the average fit error vs number of points

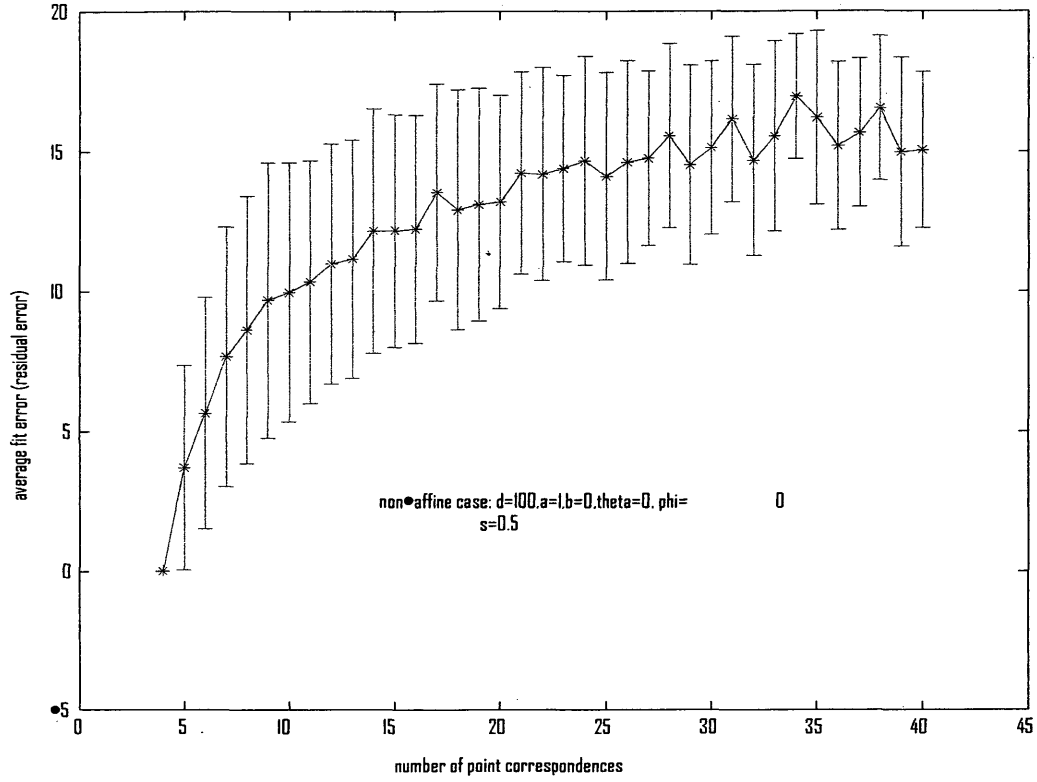


Figure 5f. Plot of the average fit error vs number of points

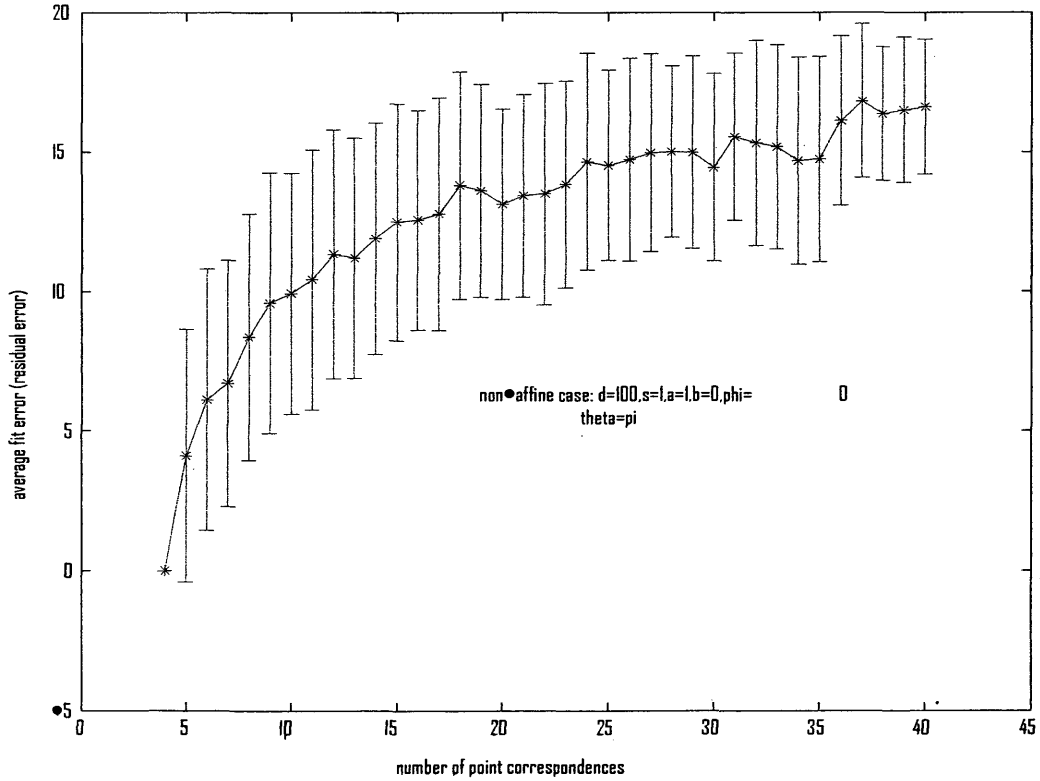


Figure 5g. Plot of the percent failures vs number of points

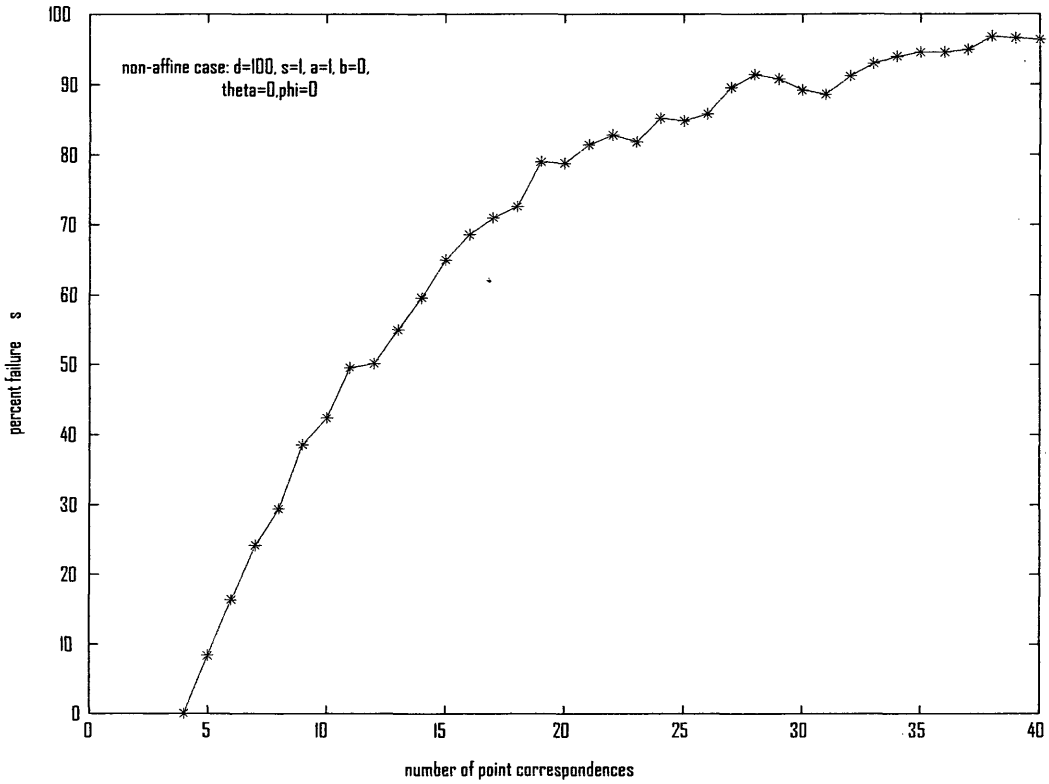


Figure 5h. Plot of the percent failures vs number of points

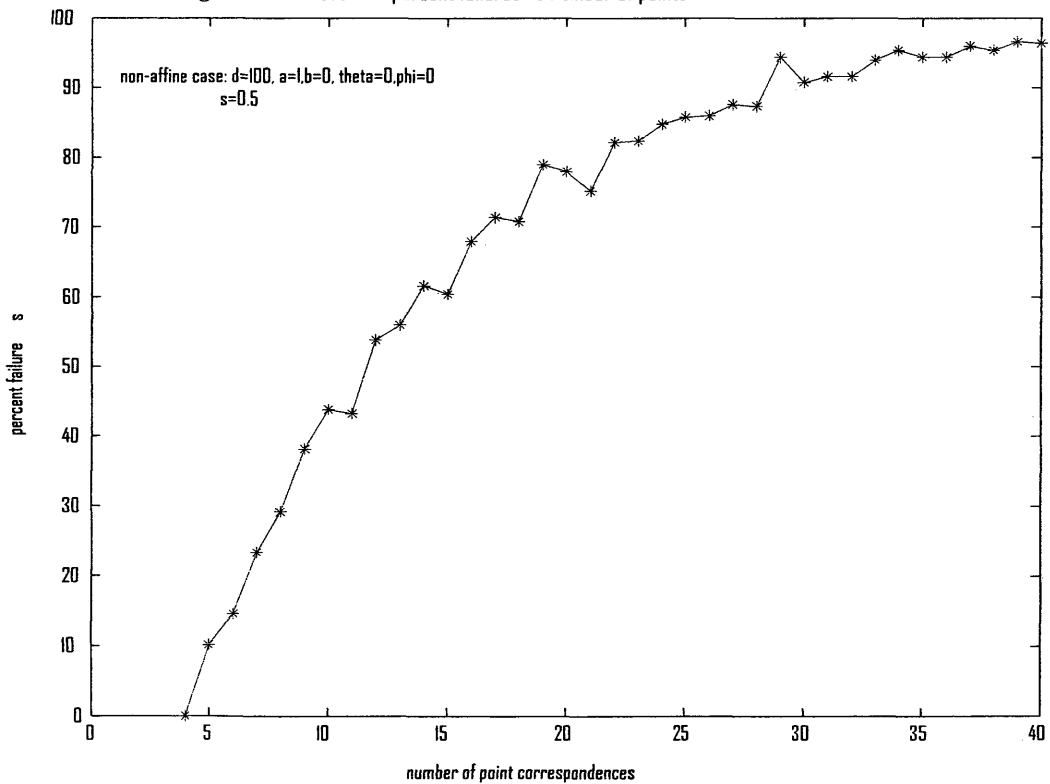


Figure 5i. Plot of the percent failures vs number of points

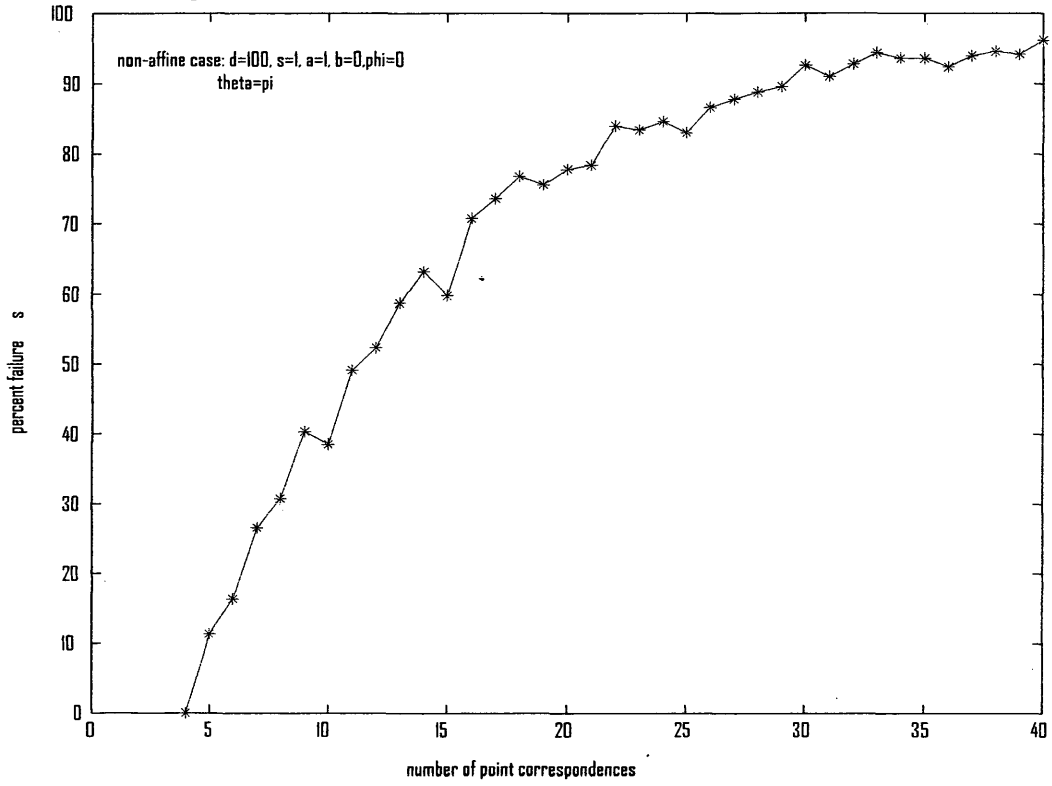


Figure 6. Case 3 – Non-affine $d=1000$ with point and error normalization with DLT algorithm

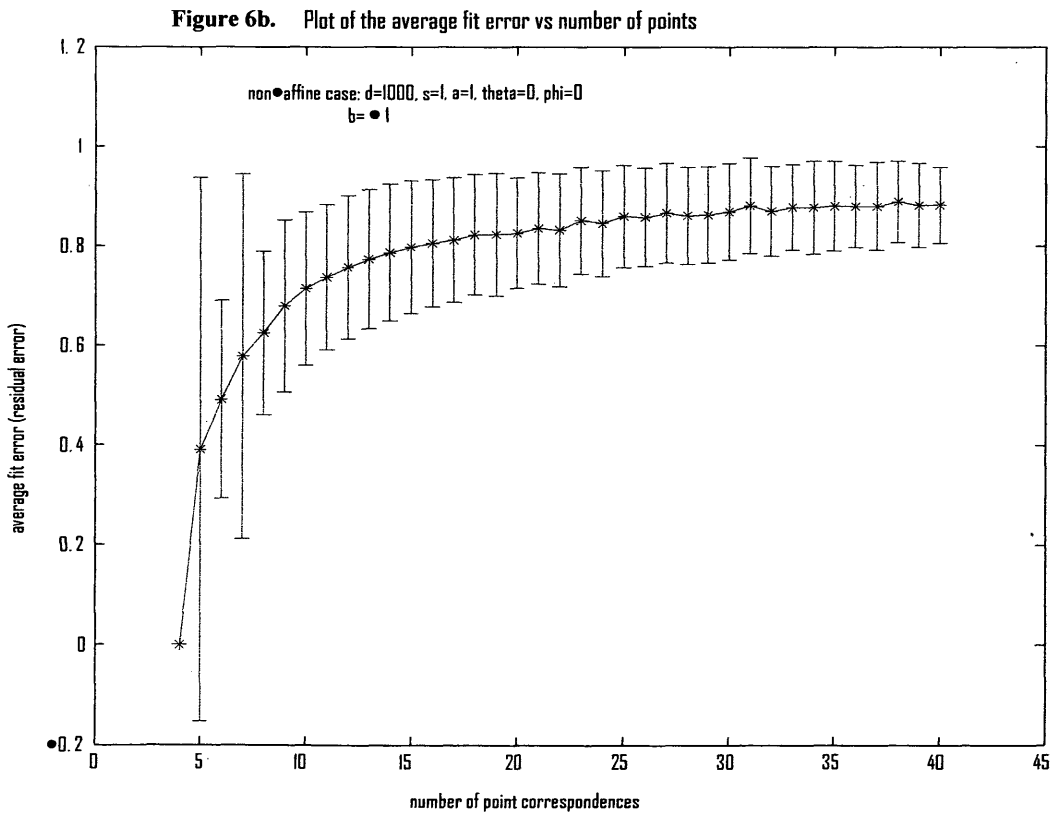
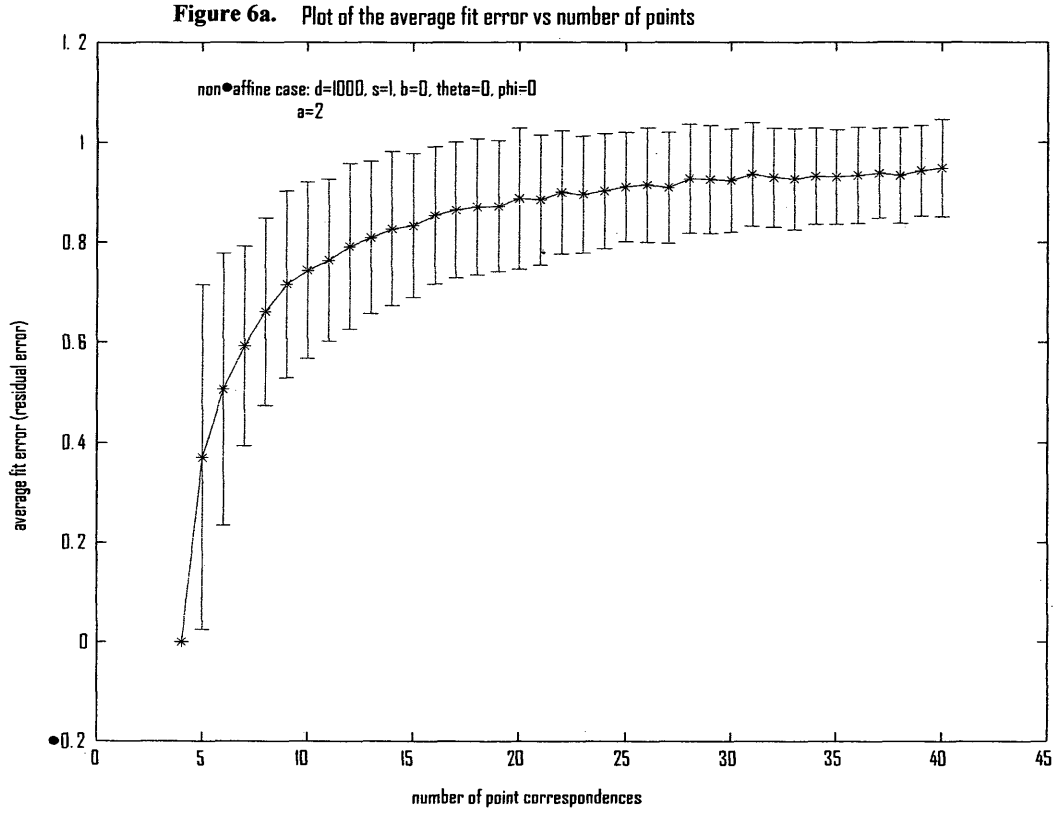


Figure 6c. Plot of the average fit error vs number of points

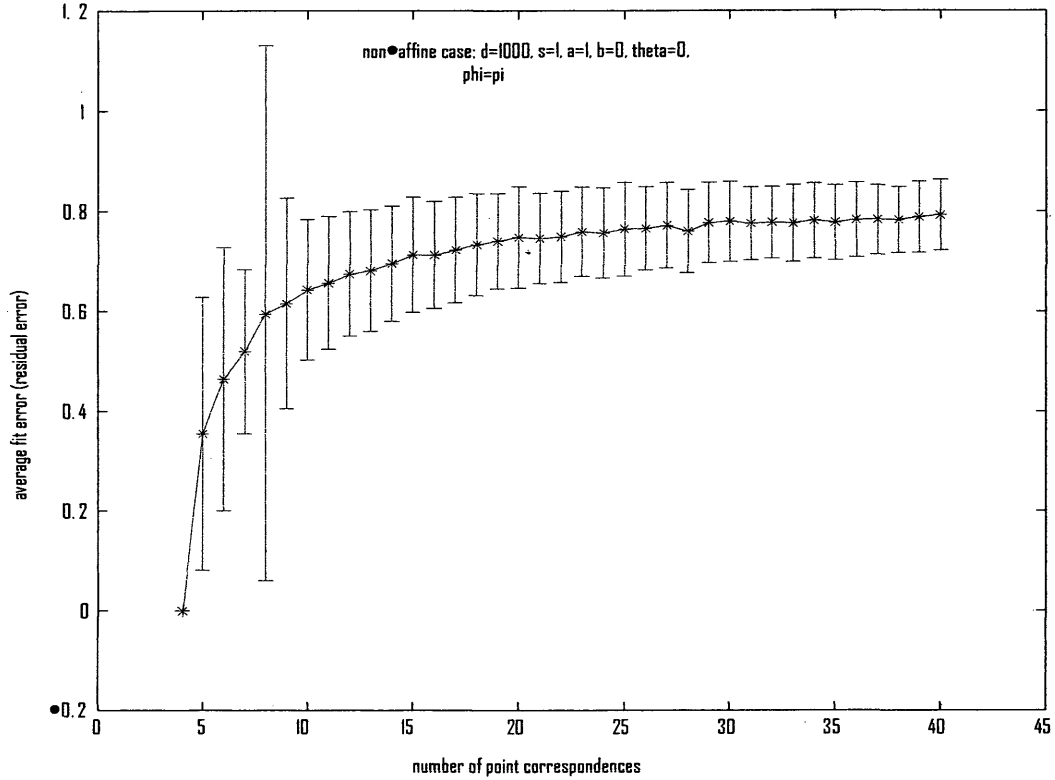


Figure 6d. Plot of the average fit error vs number of points

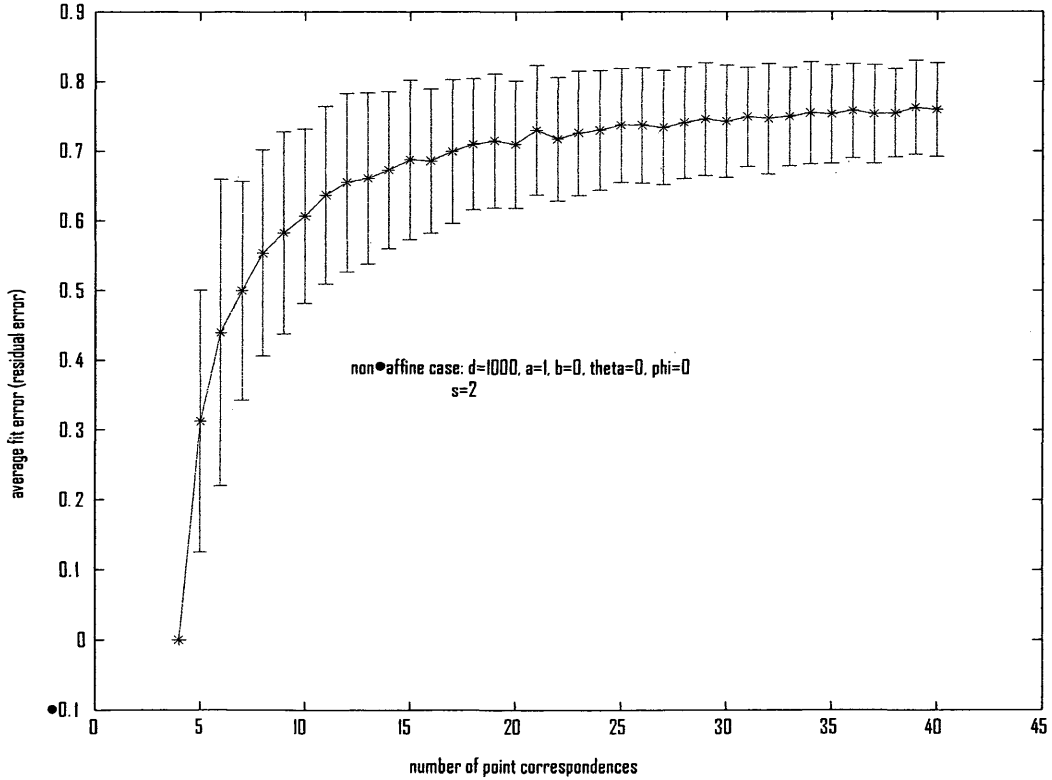


Figure 6e. Plot of the average fit error vs number of points

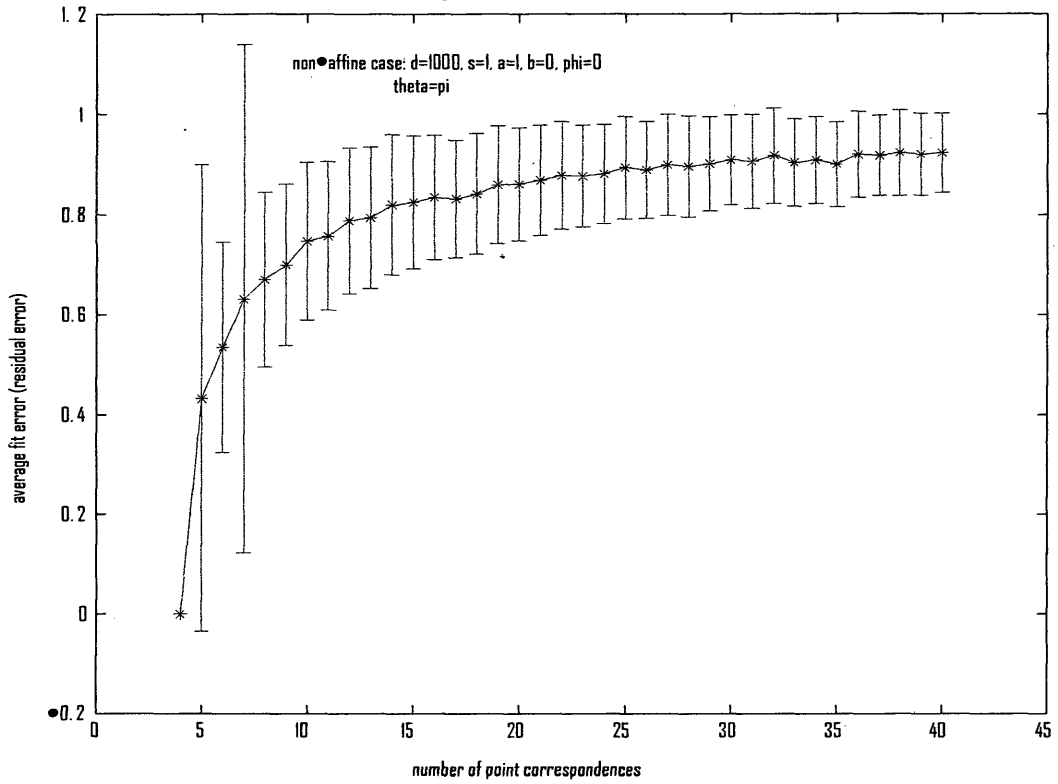


Figure 6f. Plot of the percent failures vs number of points

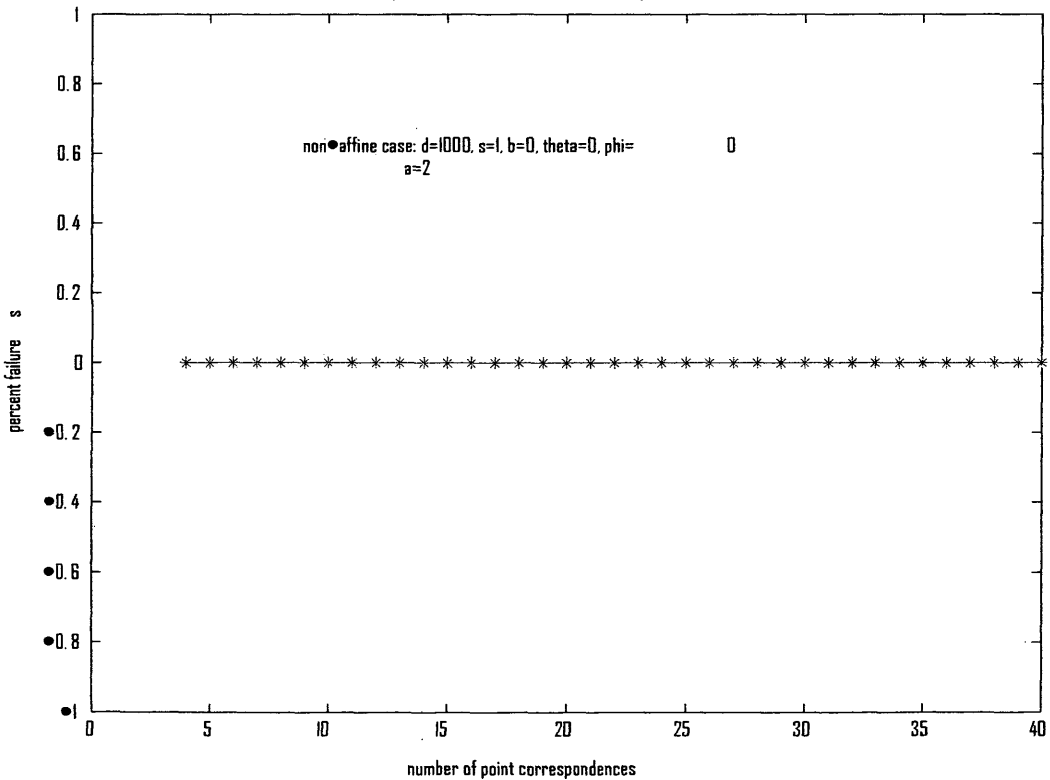


Figure 6g. Plot of the percent failures vs number of points

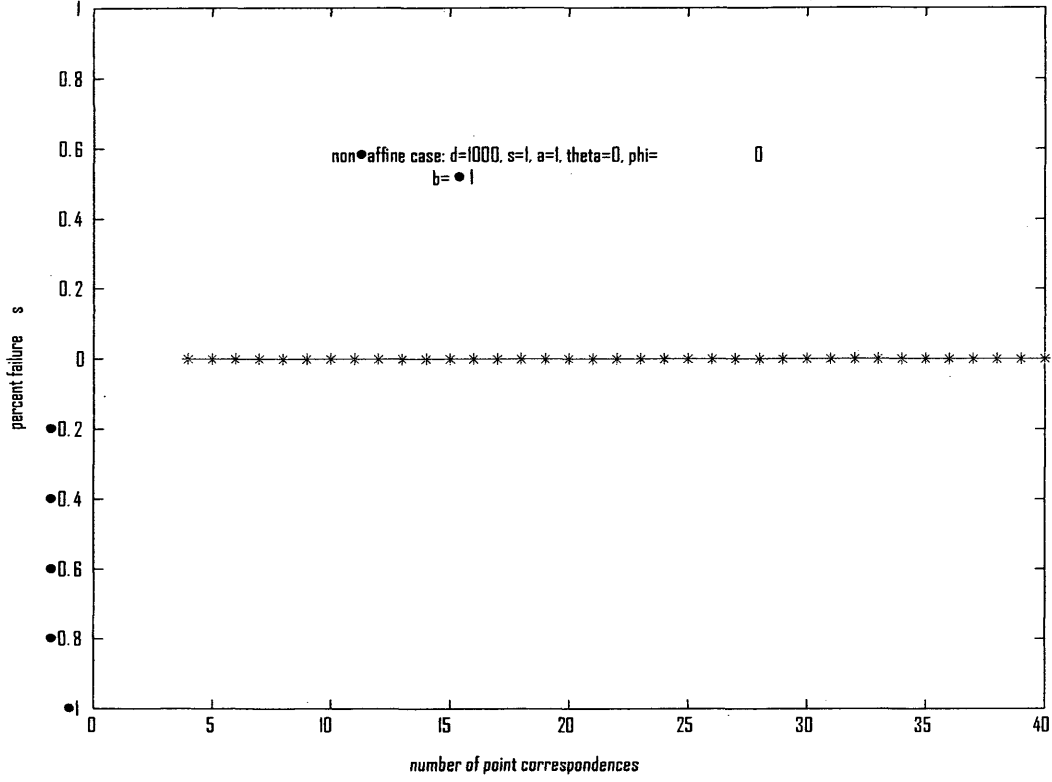


Figure 6h. Plot of the percent failures vs number of points

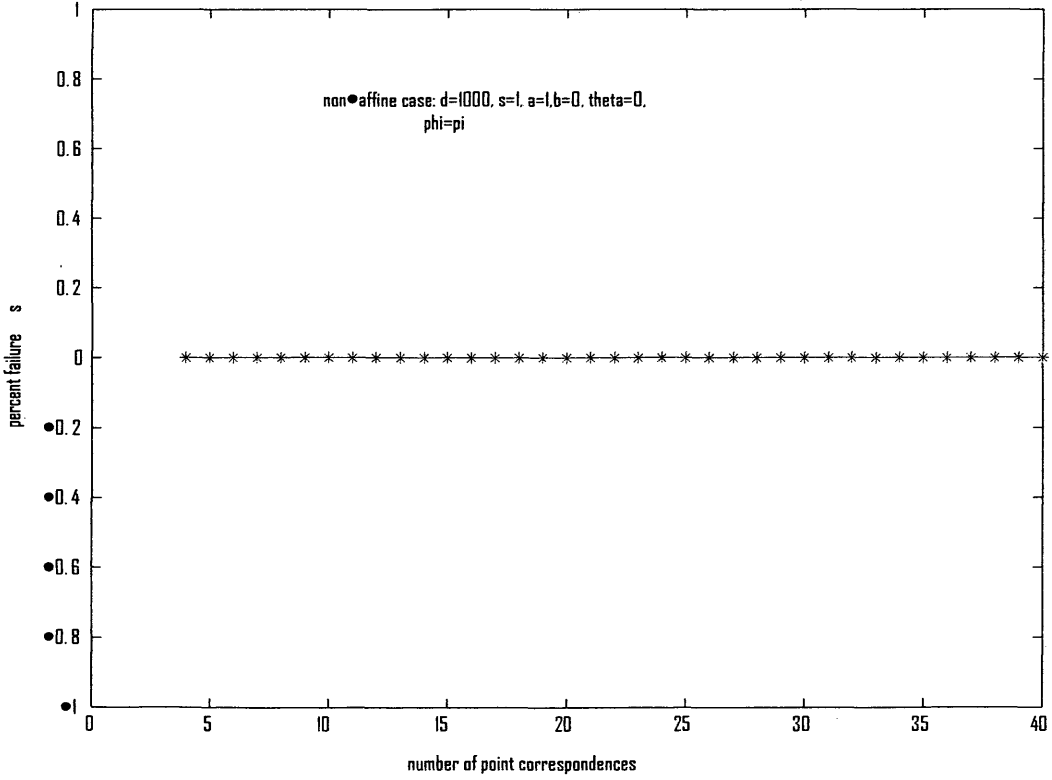


Figure 6i. Plot of the percent failures vs number of points

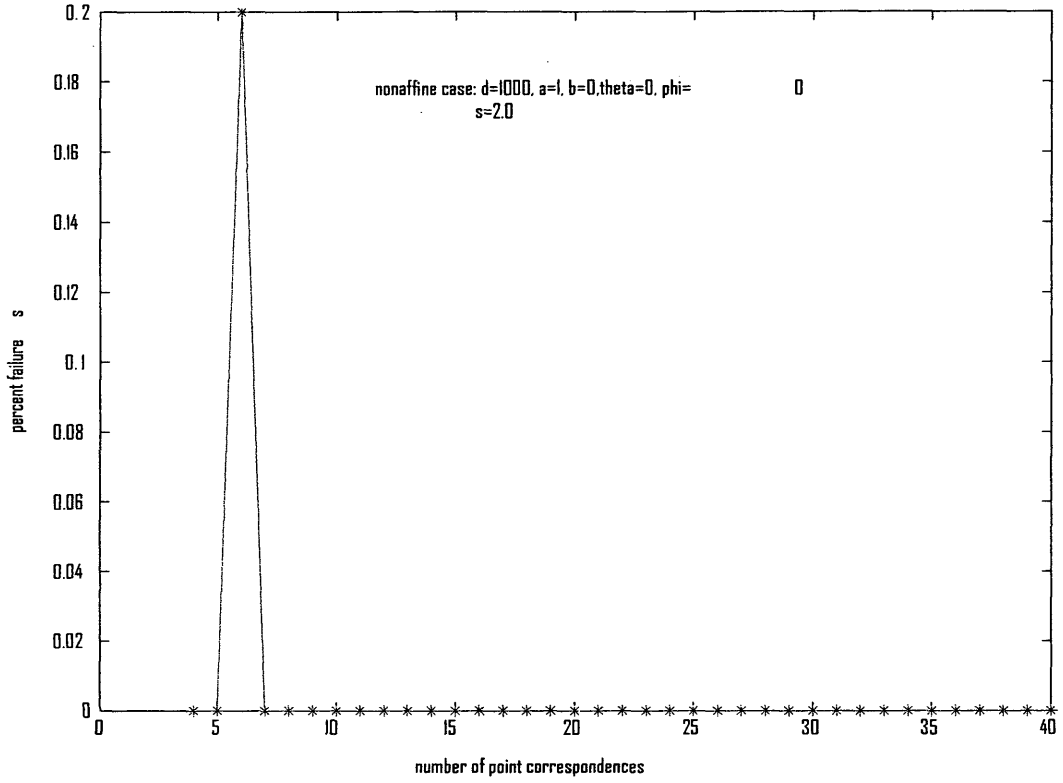


Figure 6j. Plot of the percent failures vs number of points

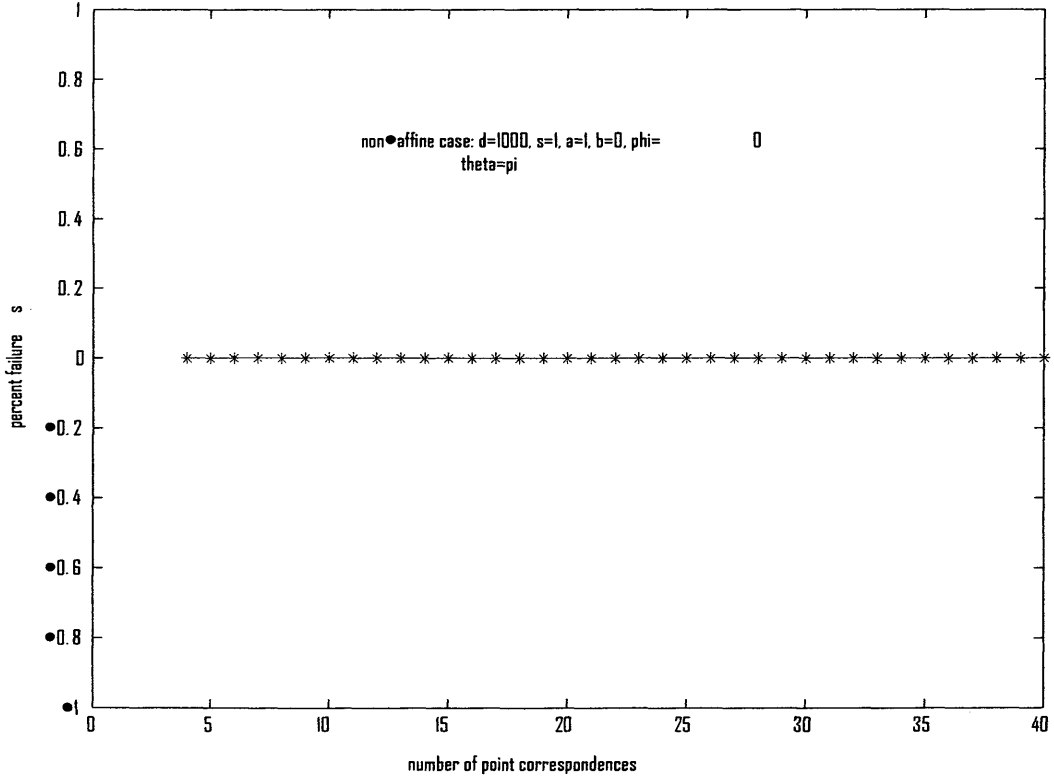


Figure 7. Case 4-- Non-affine with $d=10000$. The plots are with point and error normalizations with DLT algorithm.

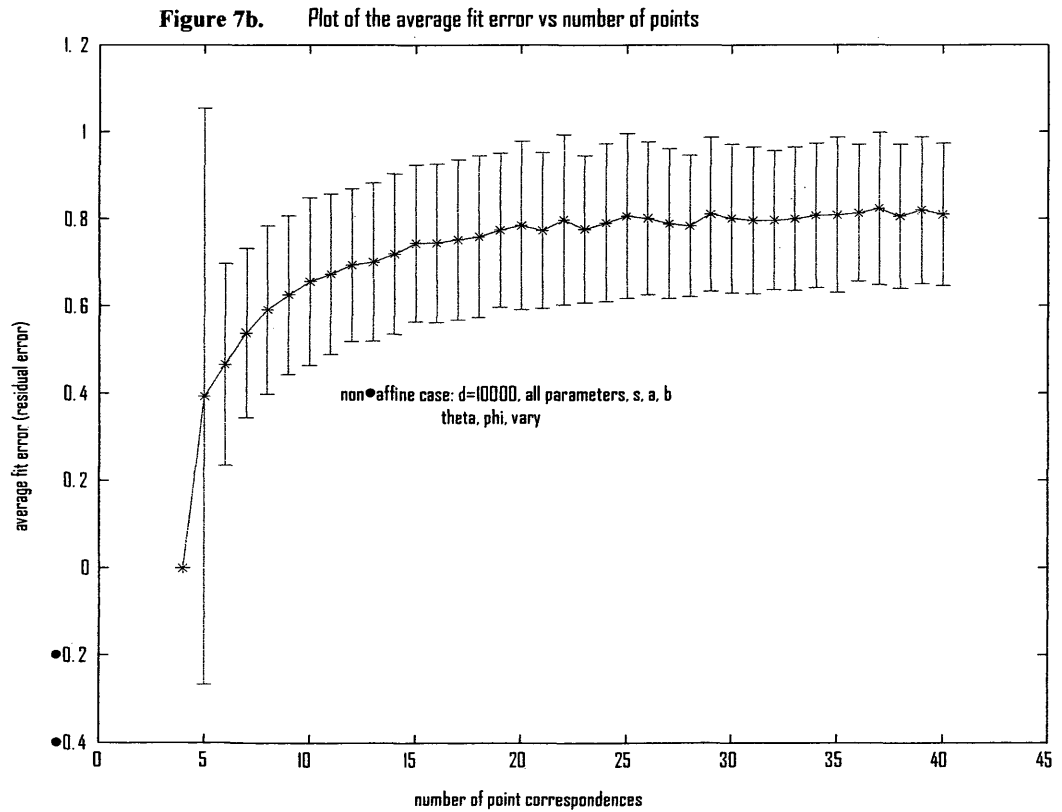
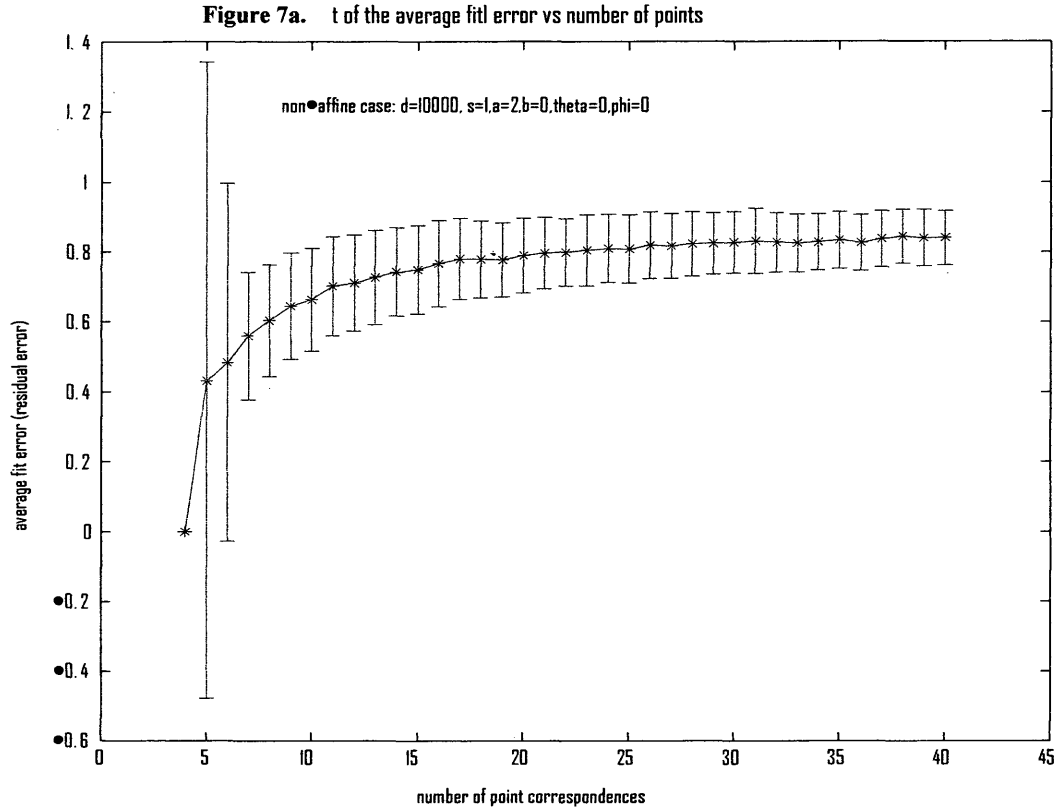


Figure 7c. Plot of the average fit error vs number of points

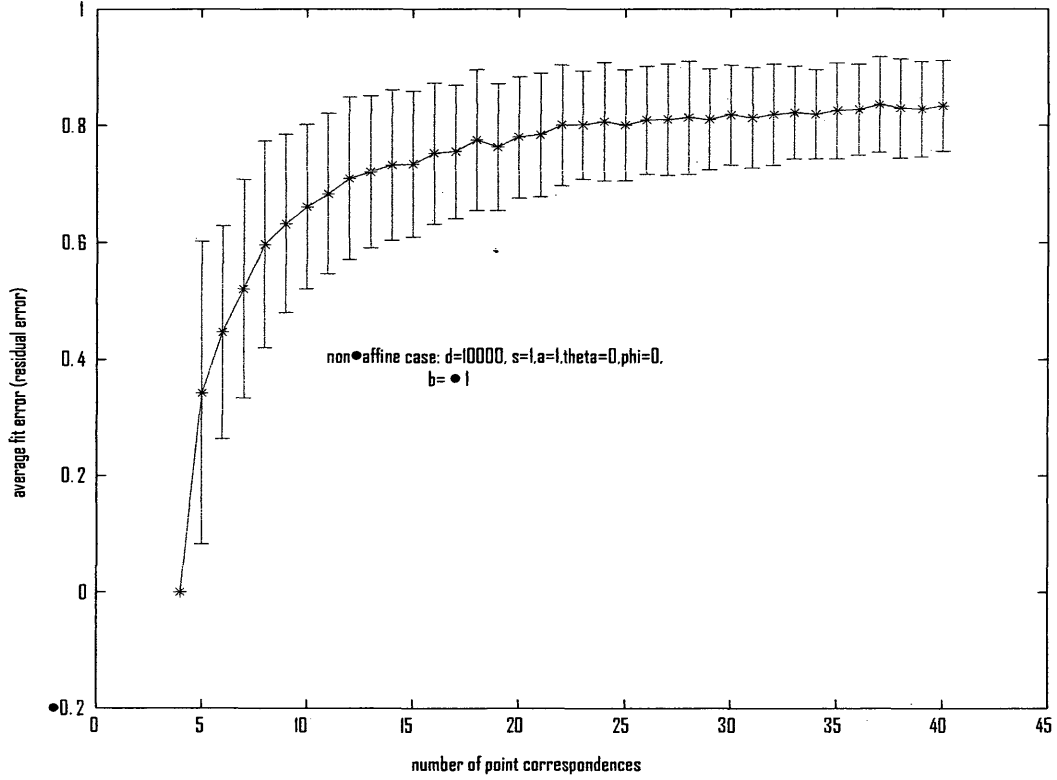


Figure 7d. Plot of the average fit error vs number of points

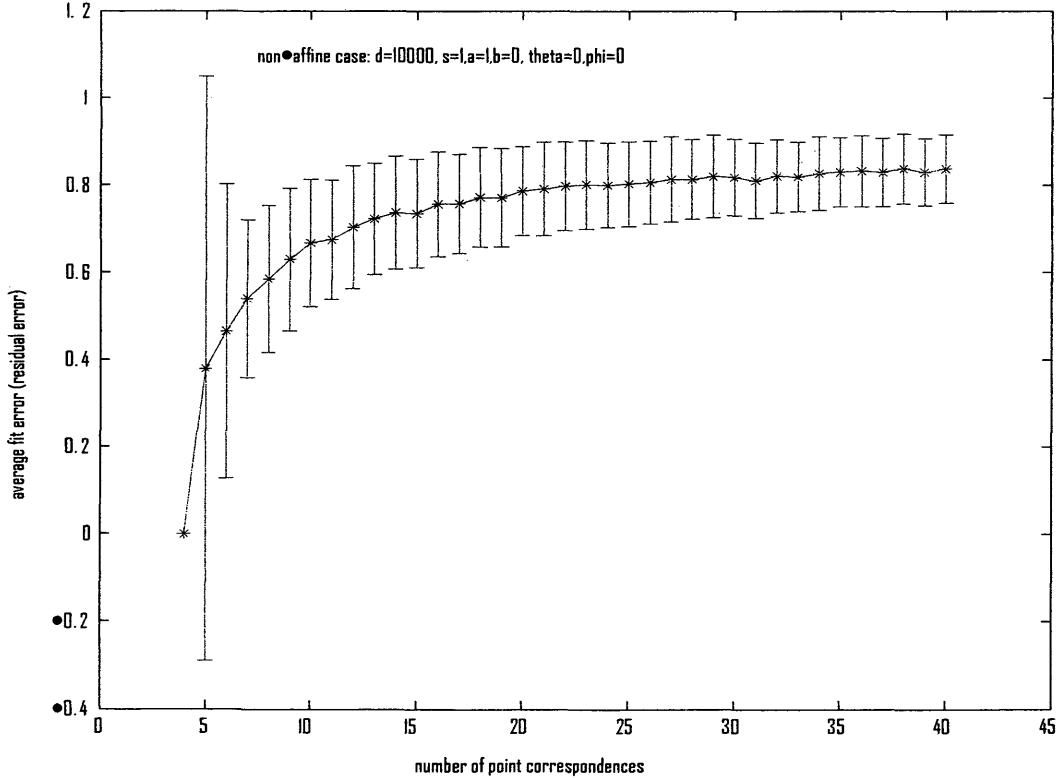


Figure 7e. Plot of the average positional error vs number of points

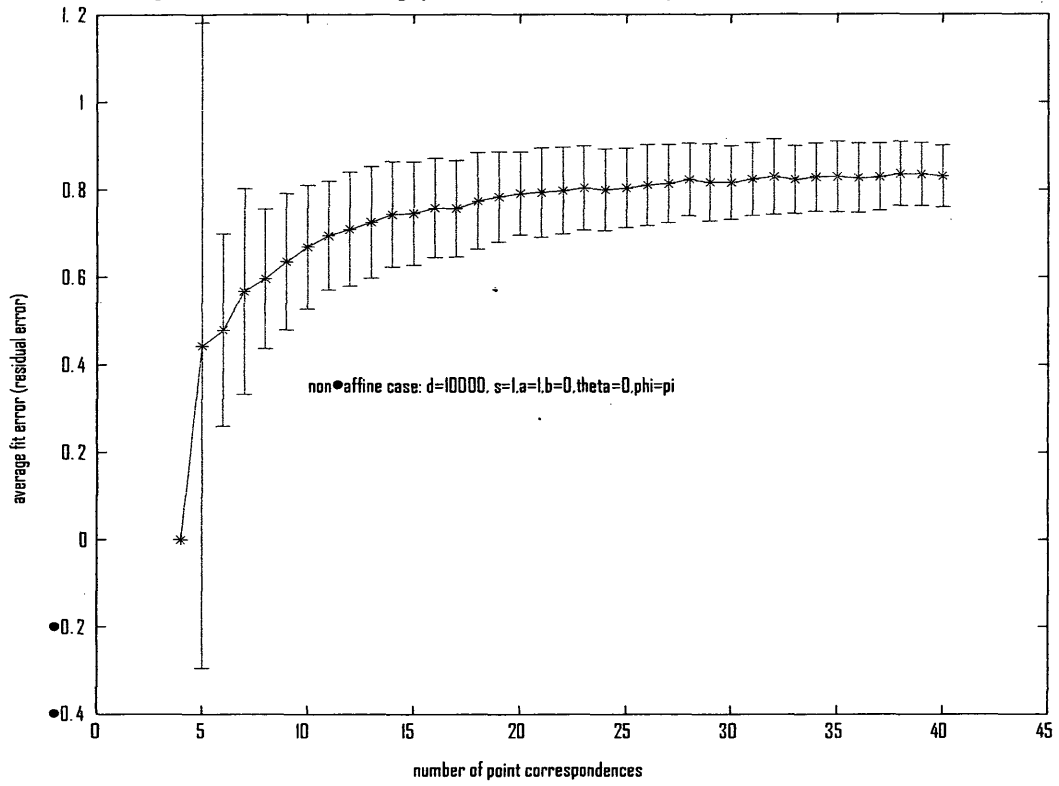


Figure 7f. Plot of the average fit error vs number of points

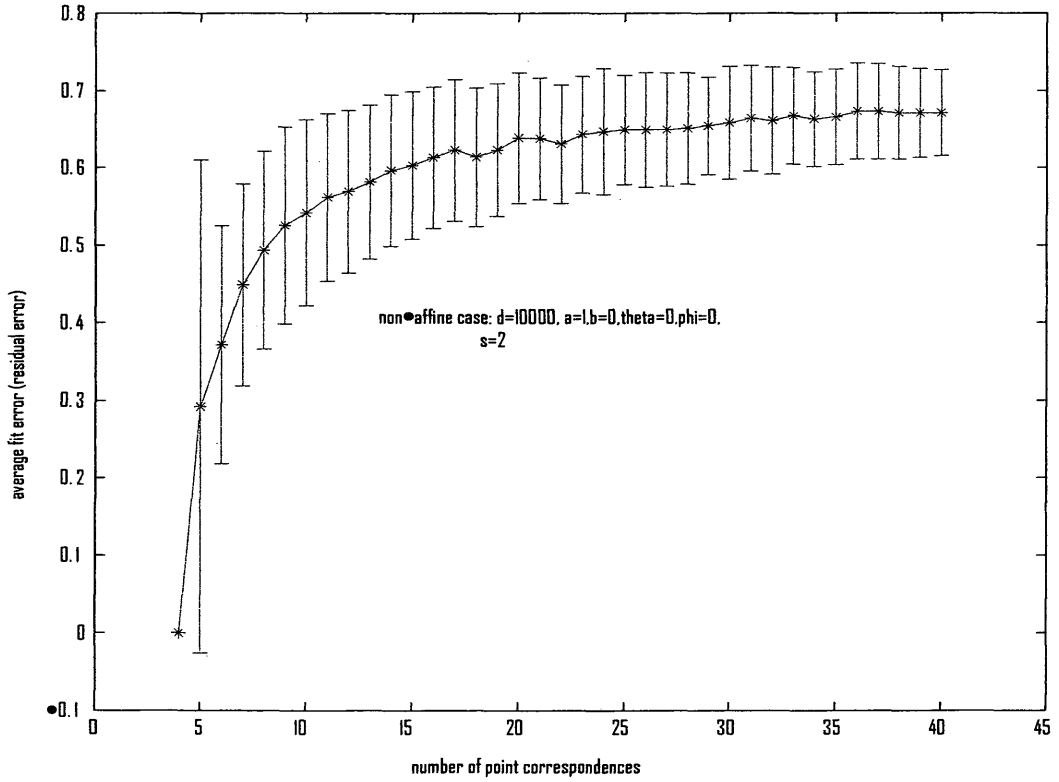


Figure 7g. Plot of the average fit error vs number of points

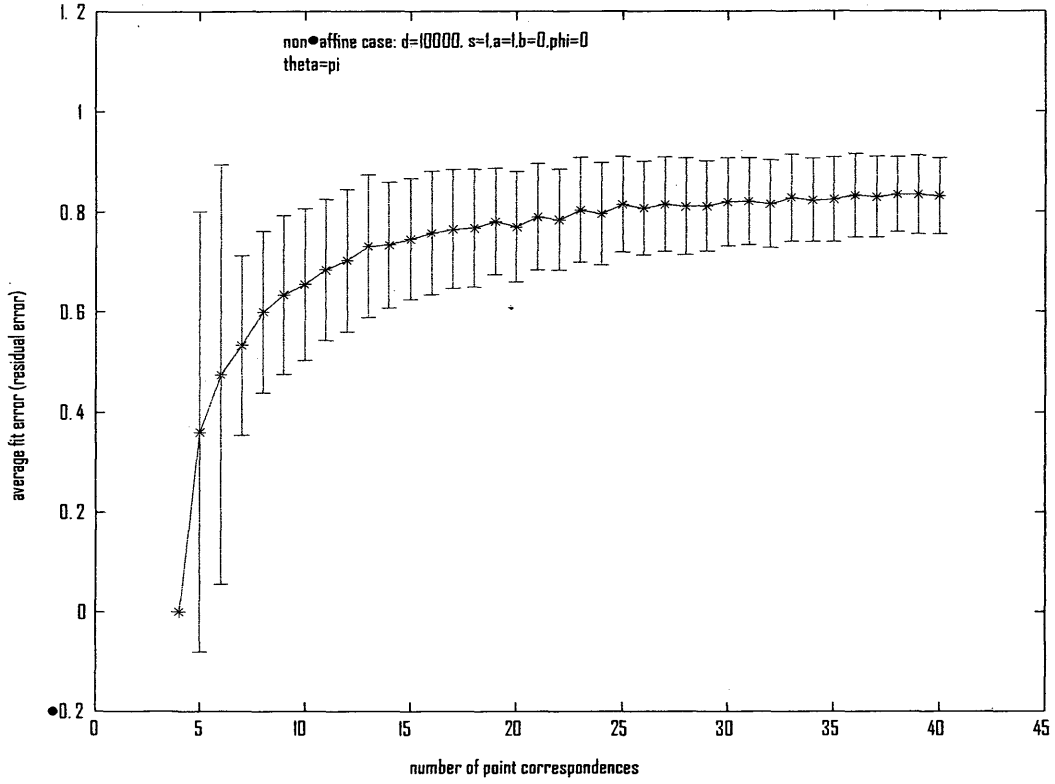


Figure 7h. Plot of the percent failures vs number of points

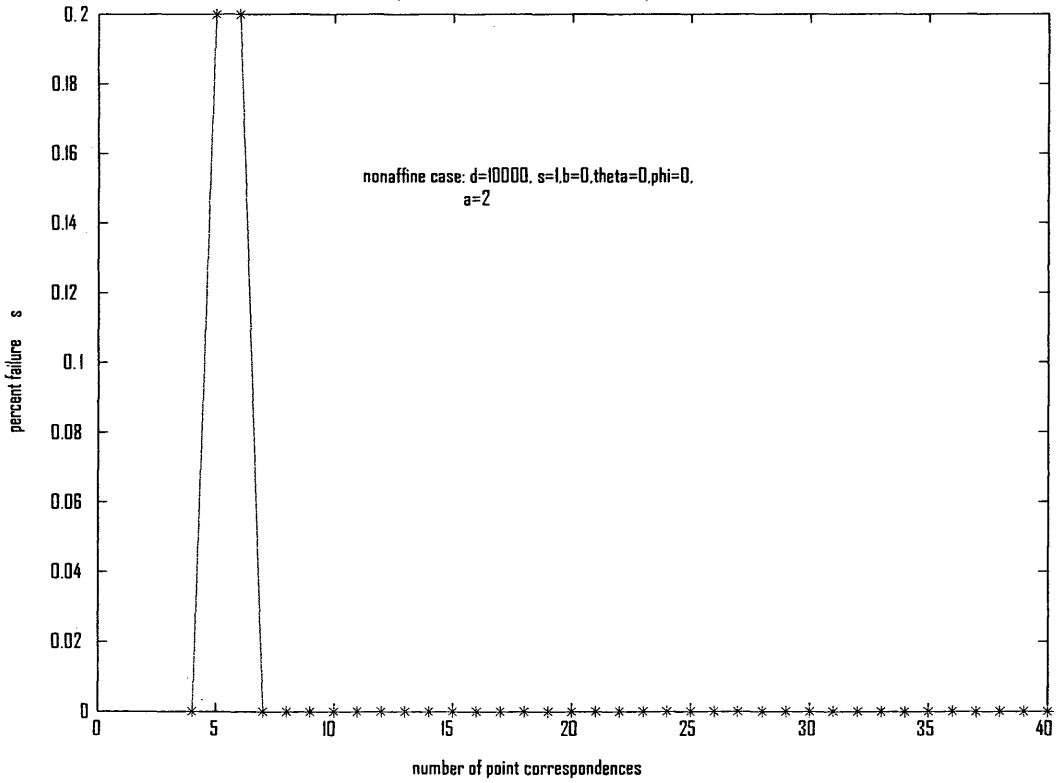


Figure 7i. Plot of the percent failures vs number of points

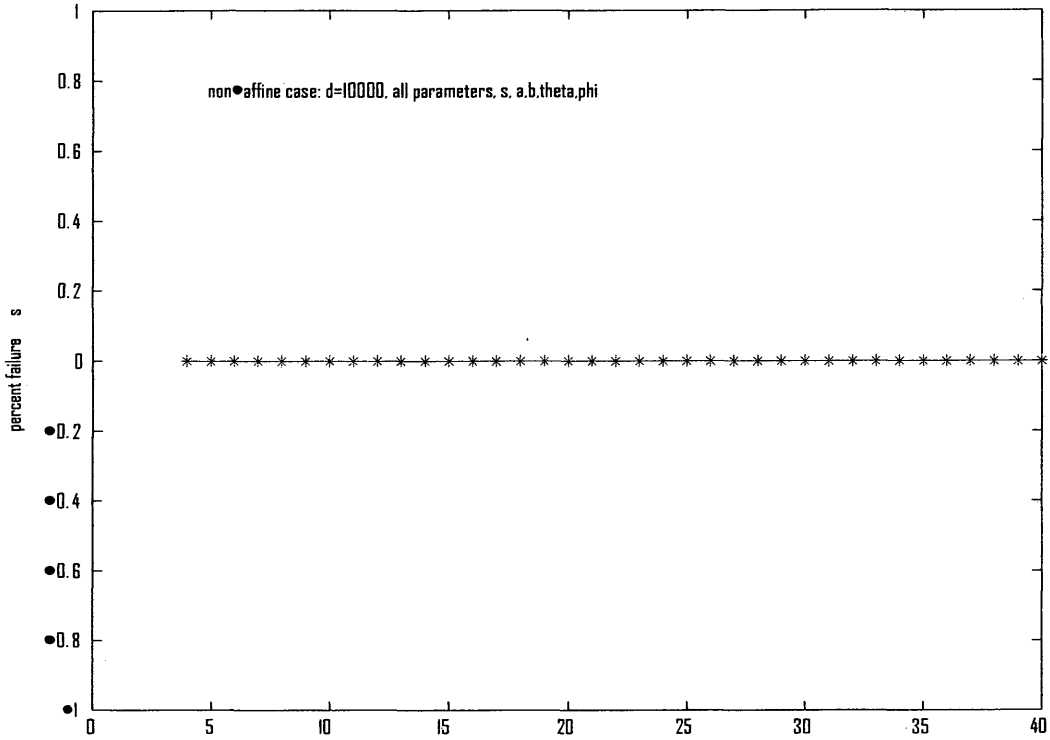


Figure 7j. Plot of the percent failures vs number of points

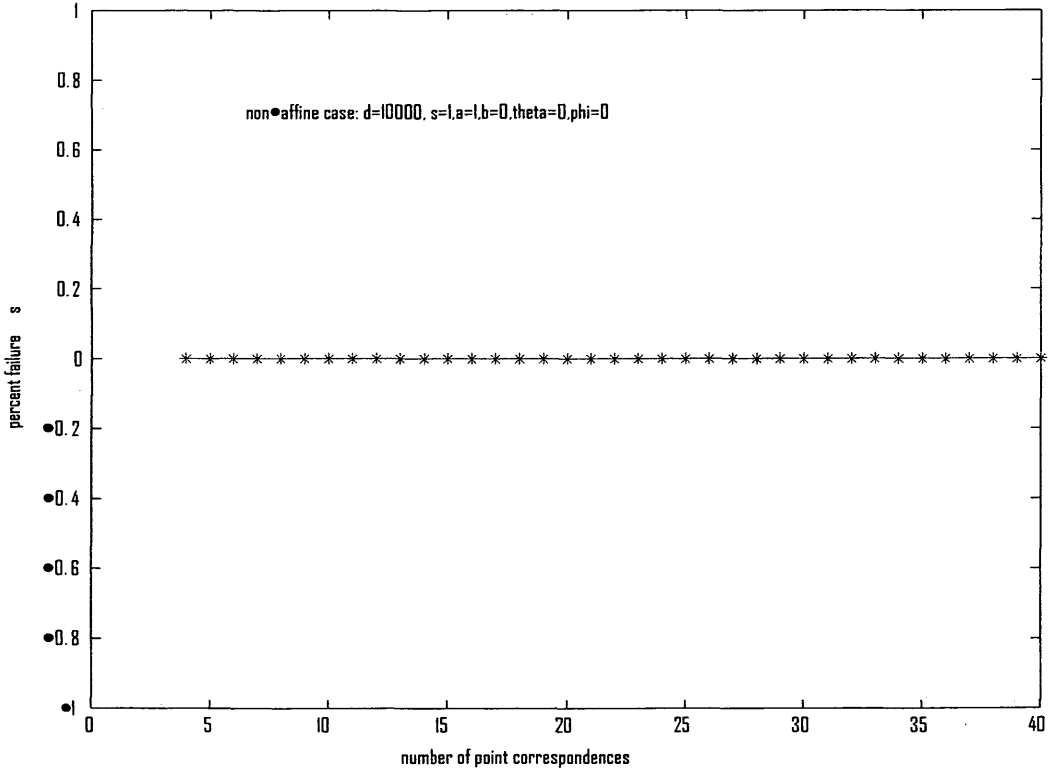


Figure 7k. Plot of the percent failures vs number of points

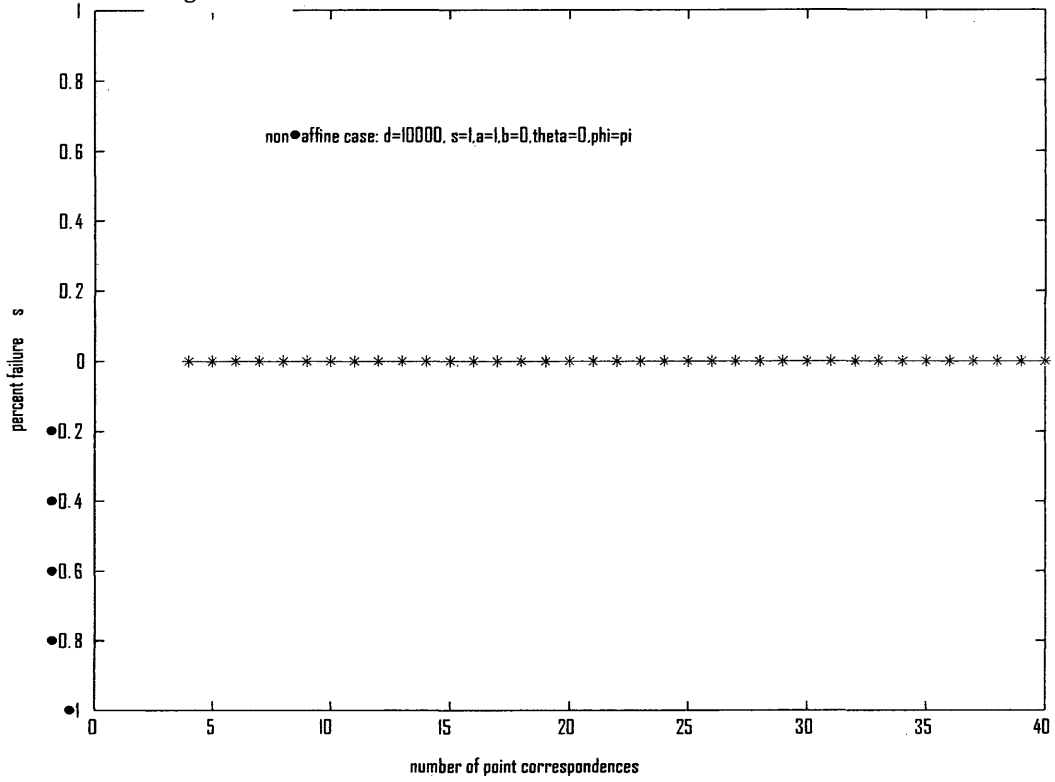


Figure 7l. Plot of the percent outliers vs number of points

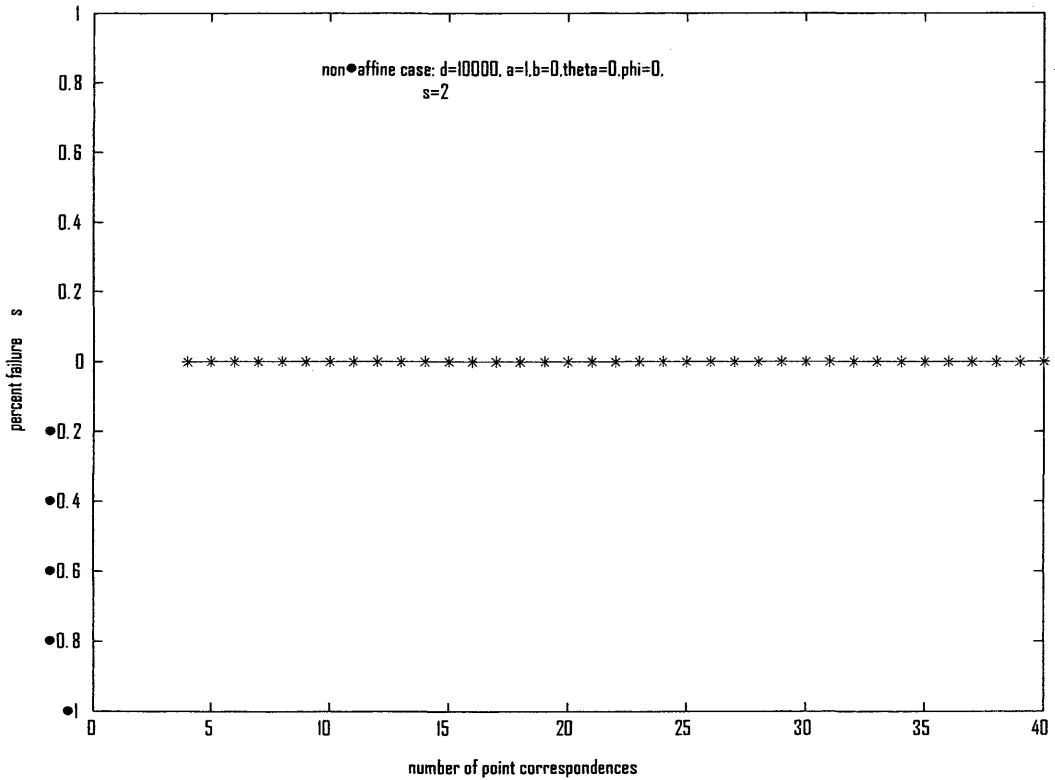


Figure 7m. Plot of the percent failures vs number of points

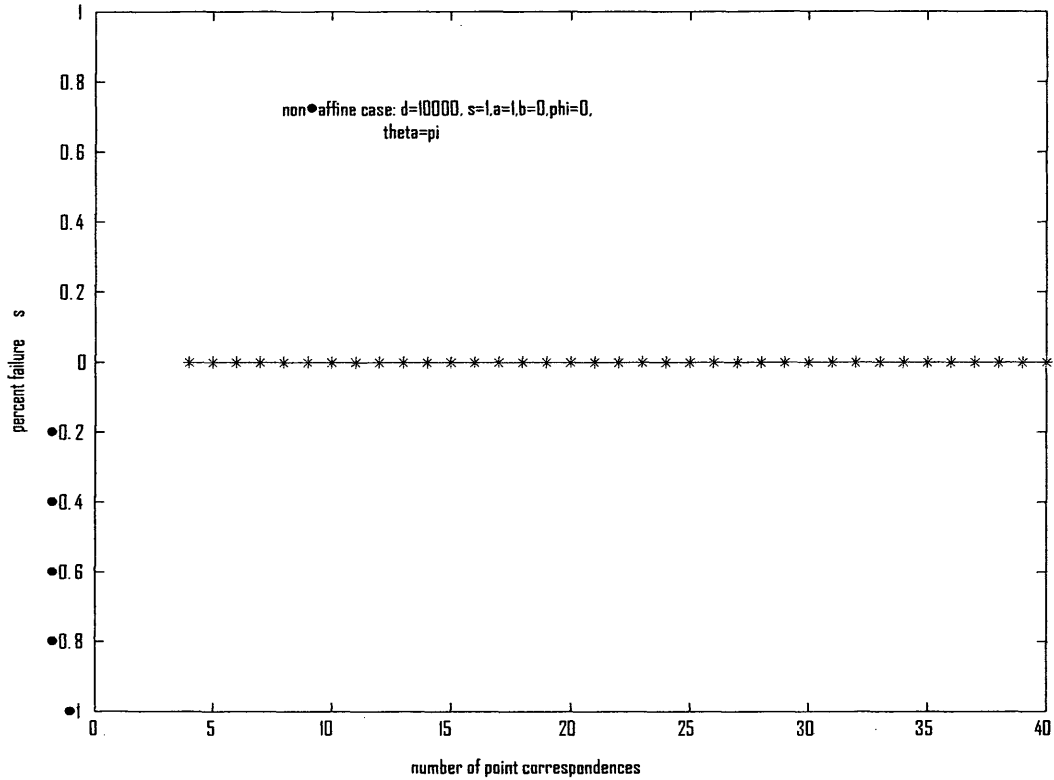


Figure 7n. Plot of the average fit error vs number of points

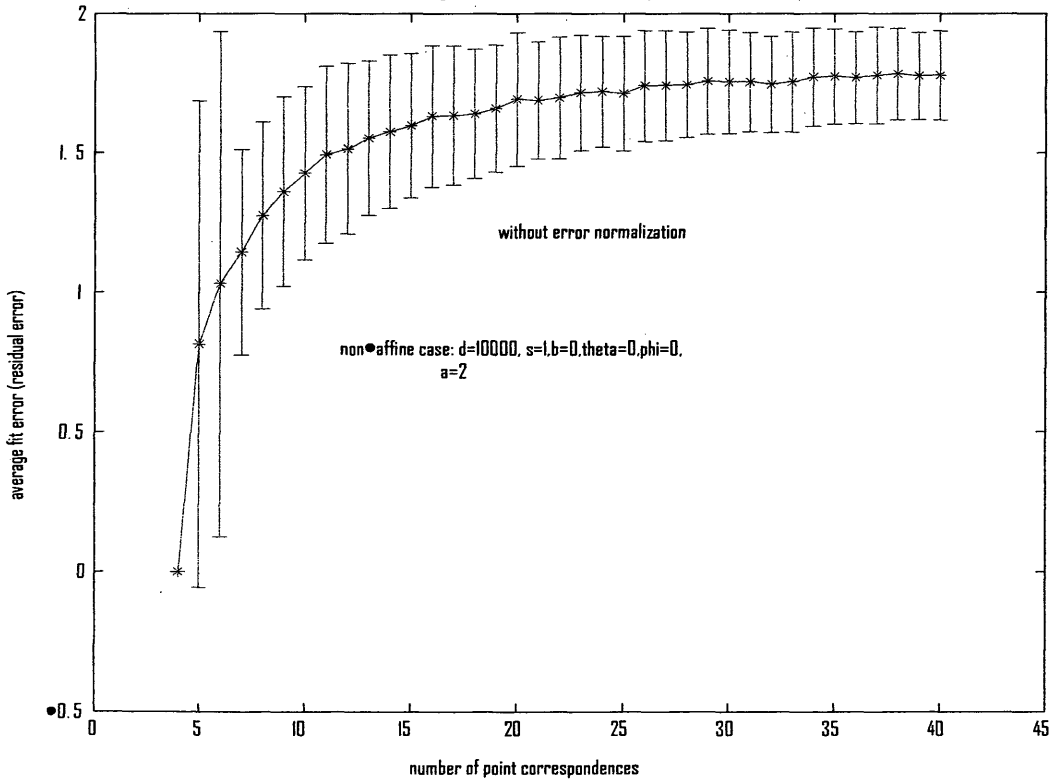


Figure 7o. Plot of the average fit error vs number of points

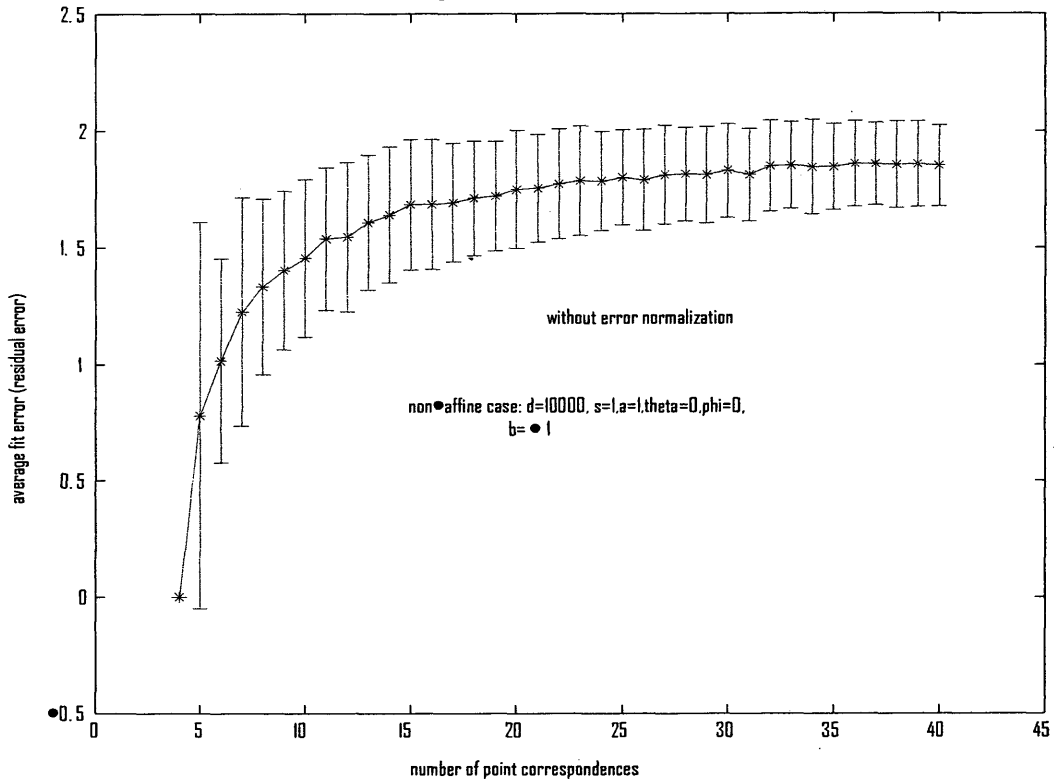


Figure 7p. Plot of the average fit error vs number of points

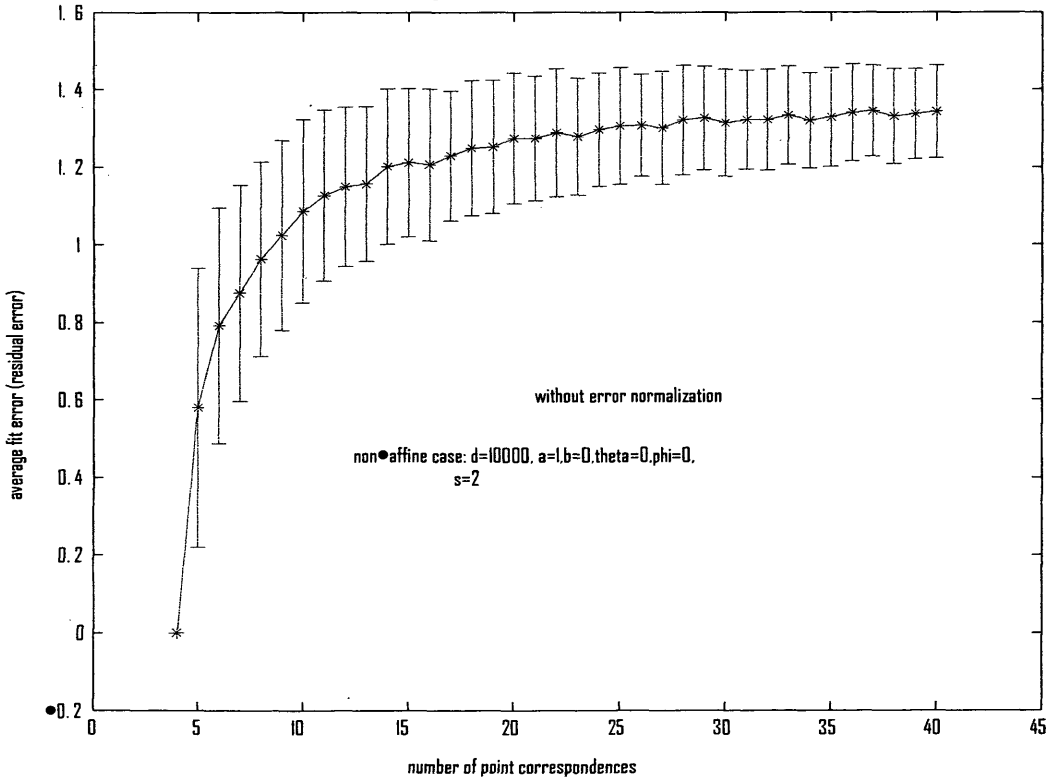


Figure 7q. Plot of the average fit error vs number of points

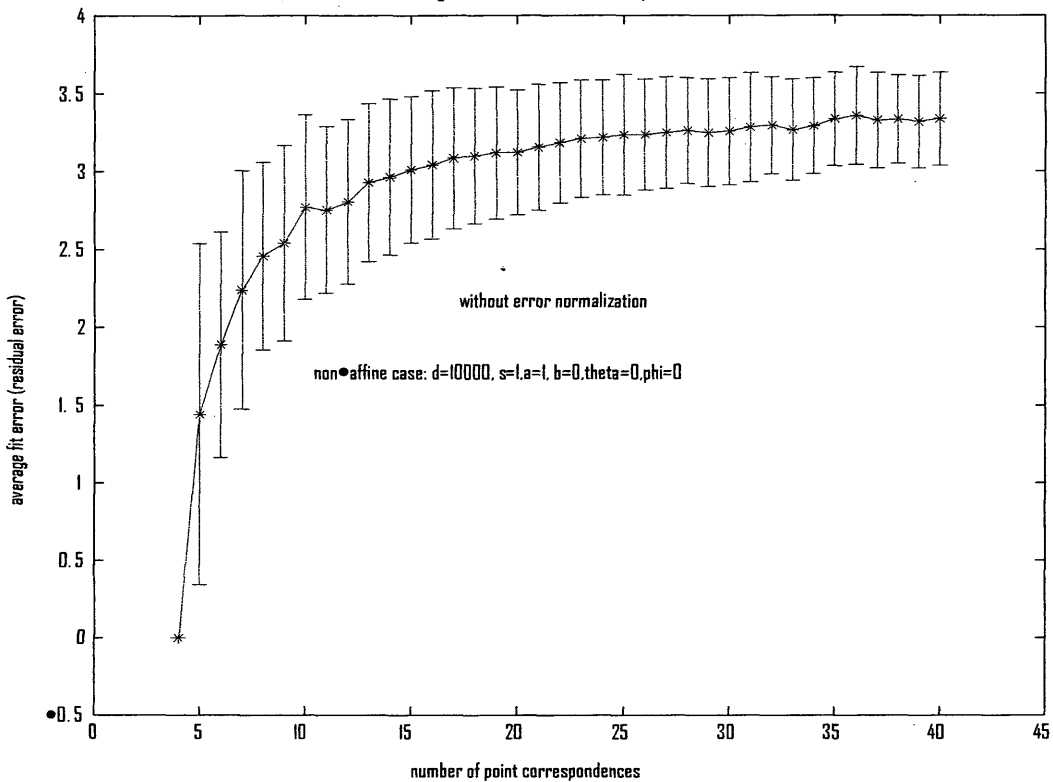


Figure 7r. Plot of the percent failures vs number of points

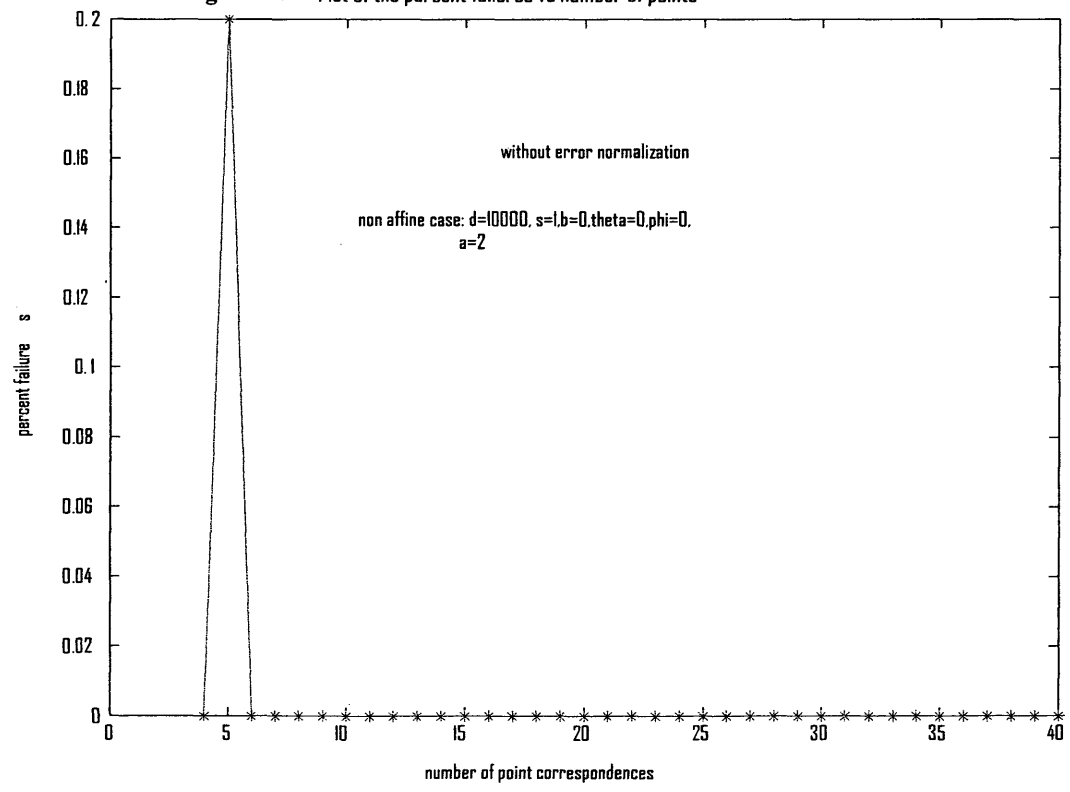


Figure 7s. Plot of the percent failures vs number of points

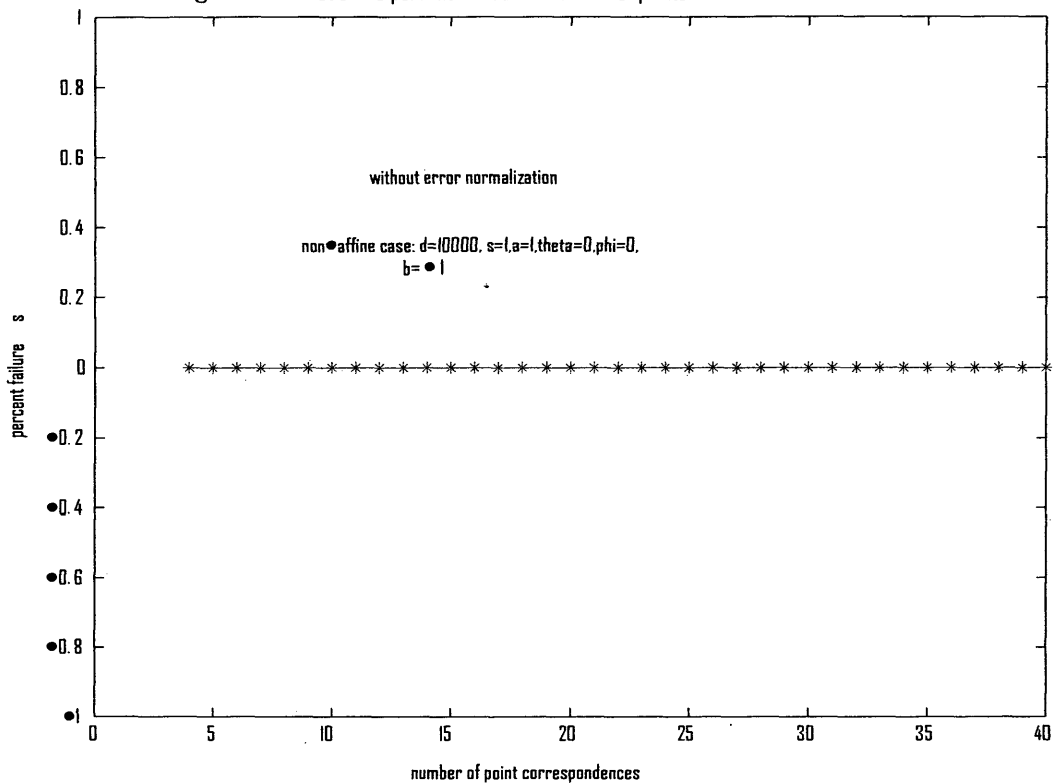


Figure 7t. Plot of the percent failures vs number of points

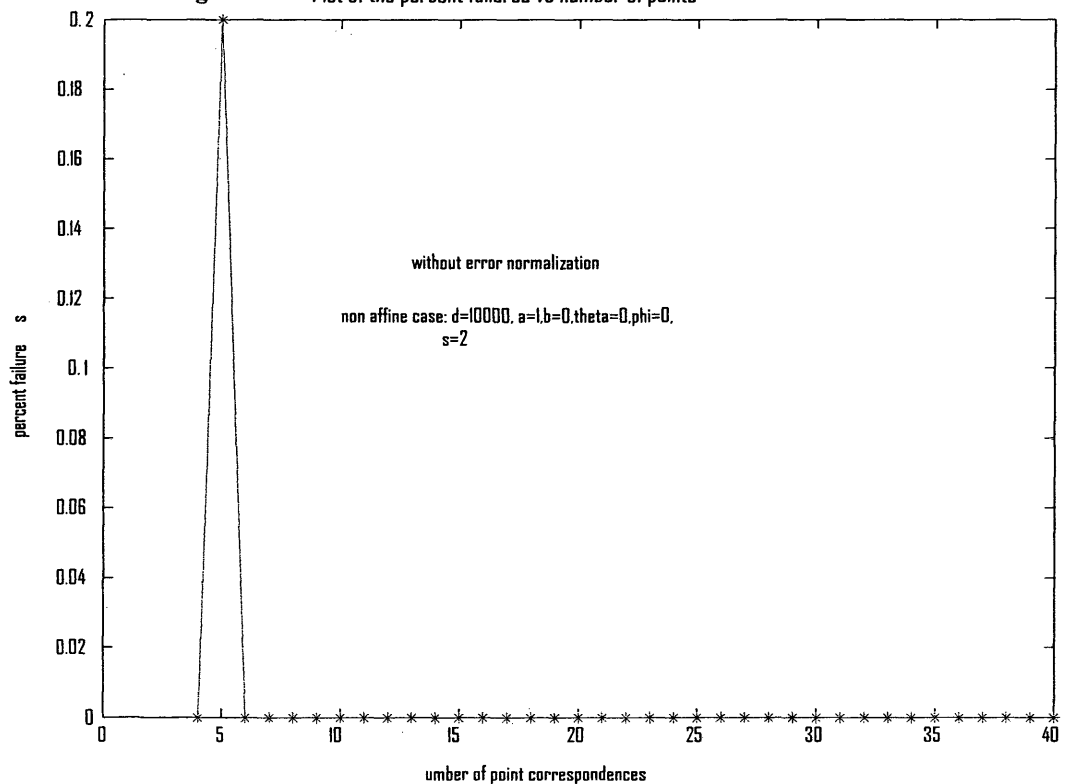


Figure 7u. Plot of the percent failures vs number of points

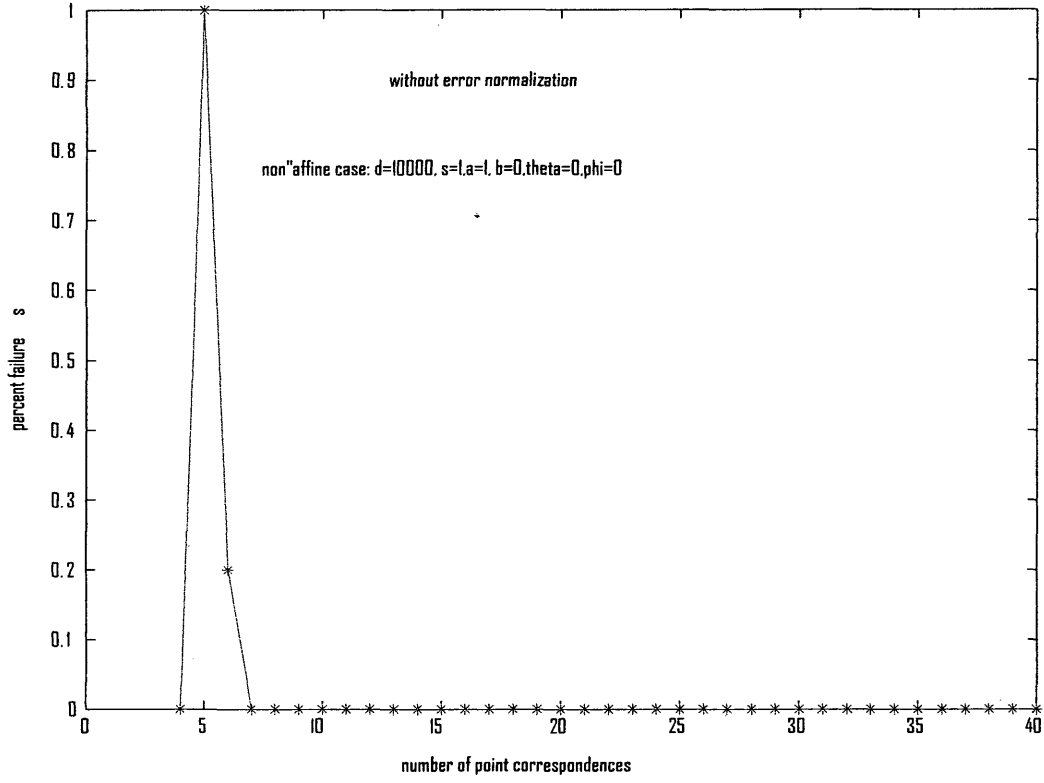


Figure 8. Case 5 -- Plots with correct solution

Figure 8a. Plot of the average fit error vs number of points

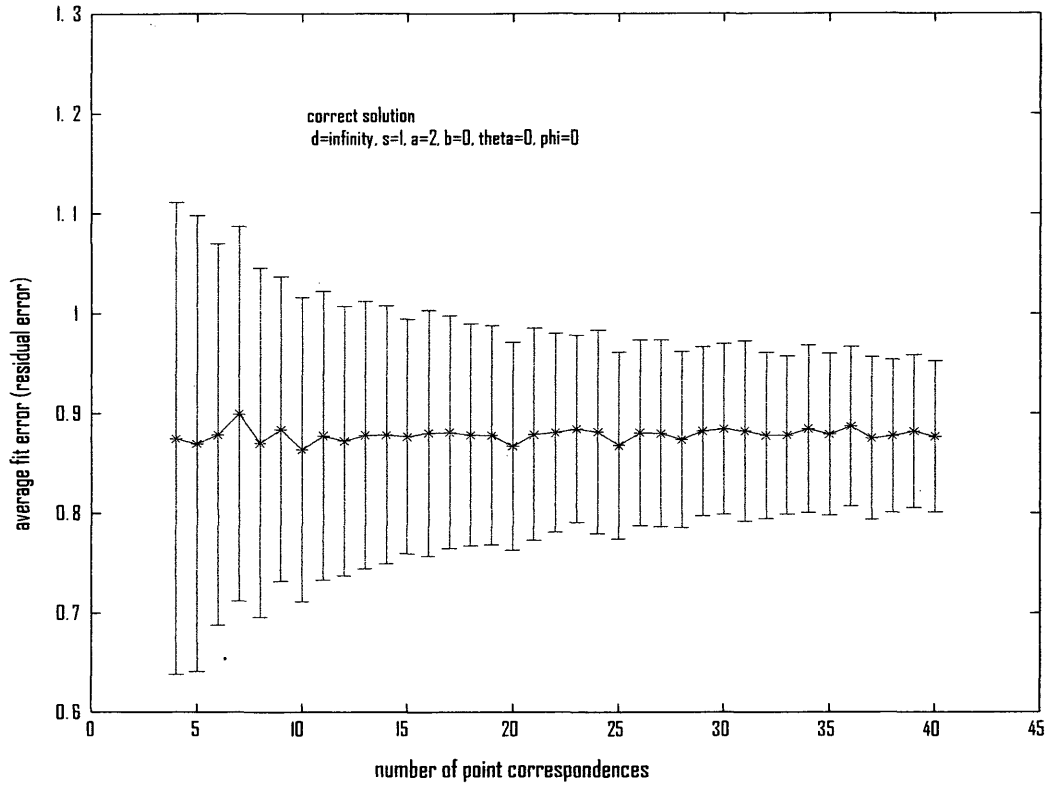


Figure 8b. Plot of the average fit error vs number of points

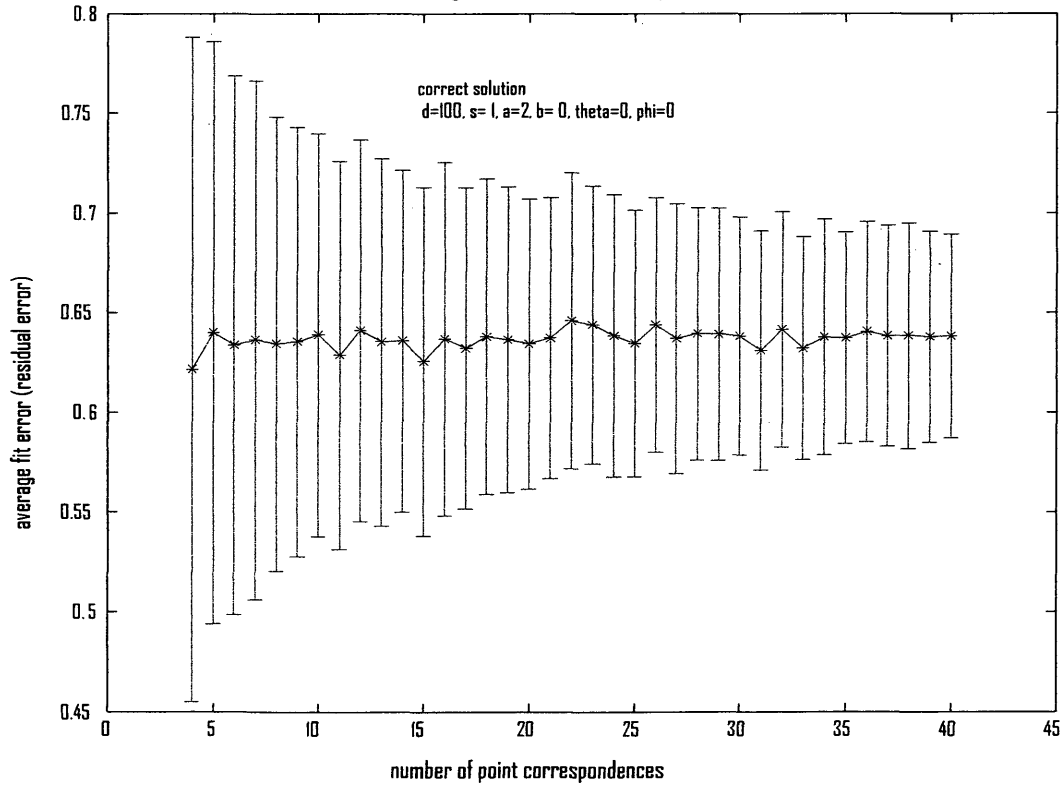


Figure 8c. Plot of the average fit error vs number of points

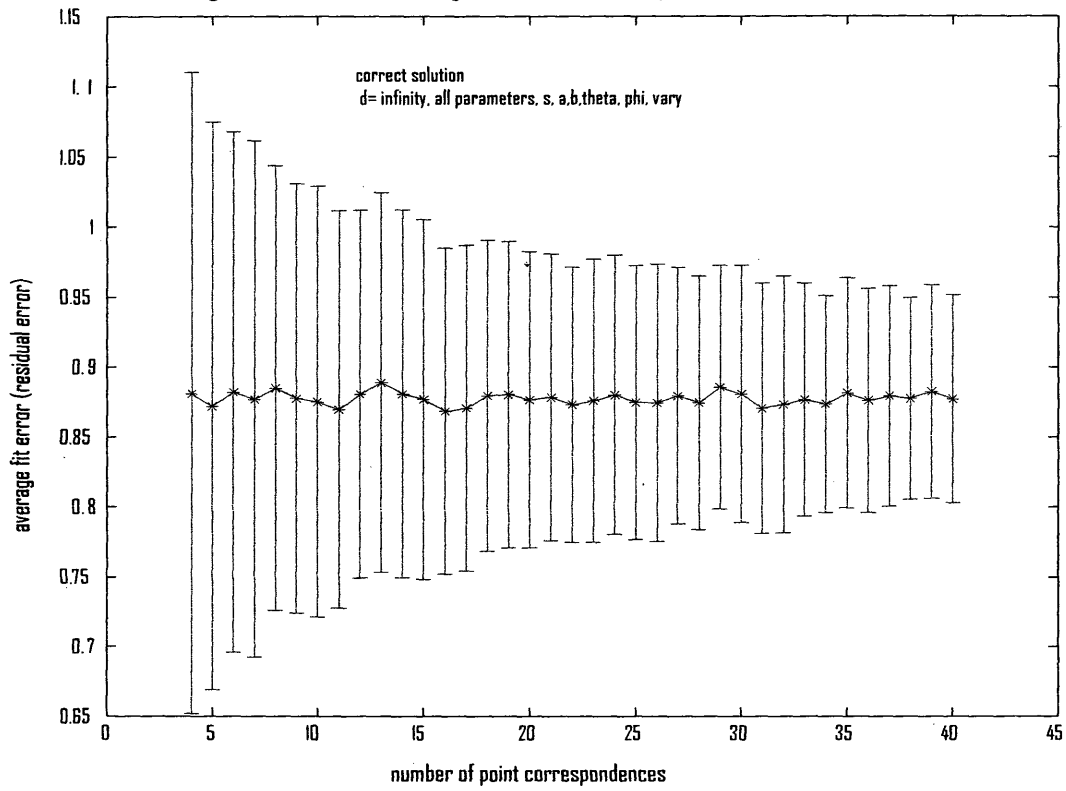


Figure 8d. Plot of the average fit error vs number of points

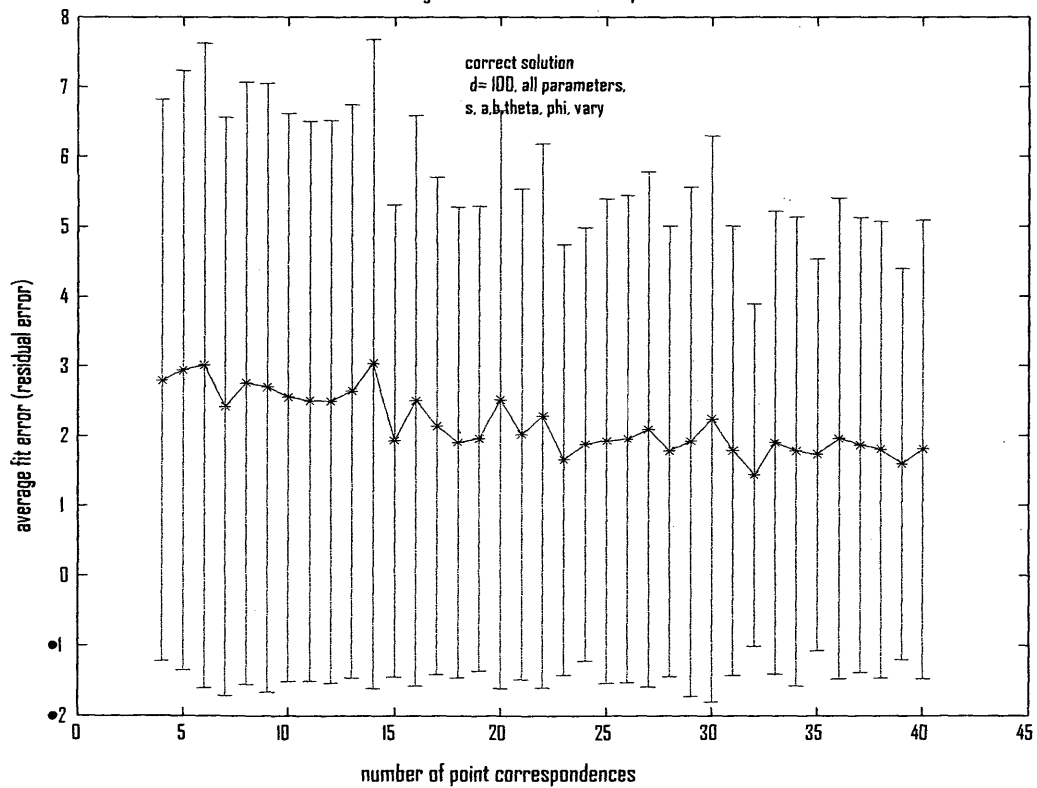


Figure 8e. Plot of the average fit error vs number of points

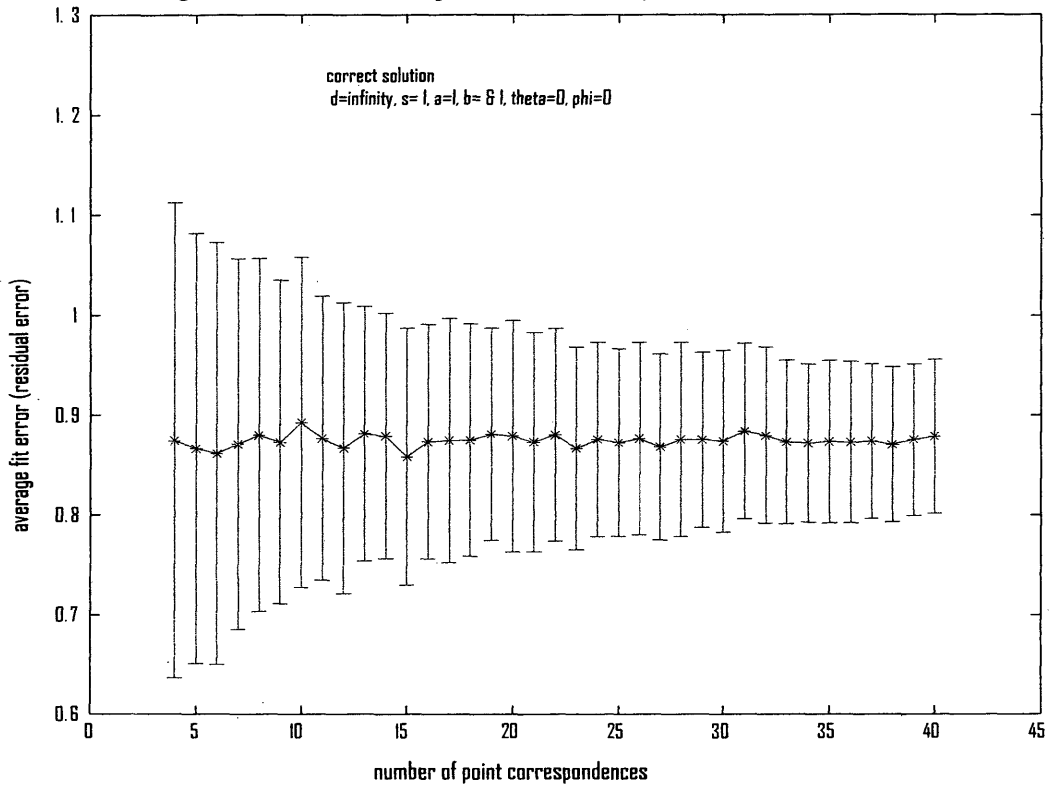


Figure 8f. Plot of the average fit error vs number of points

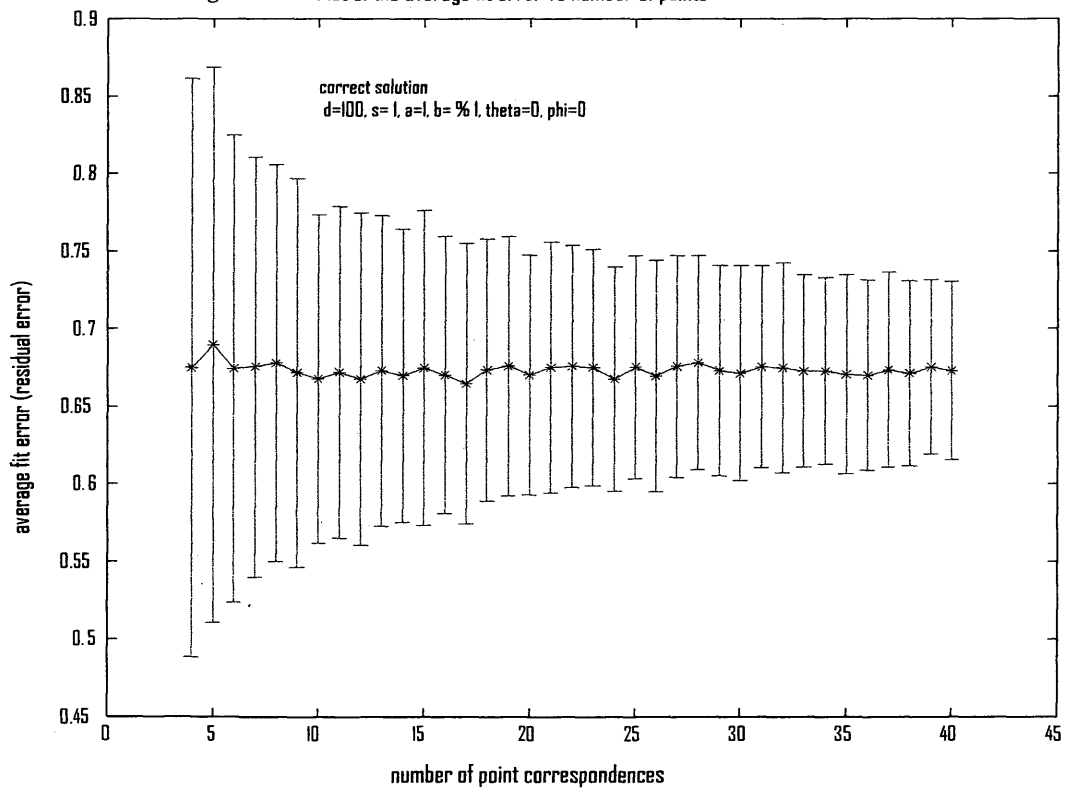


Figure 8g. Plot of the average fit error vs number of points

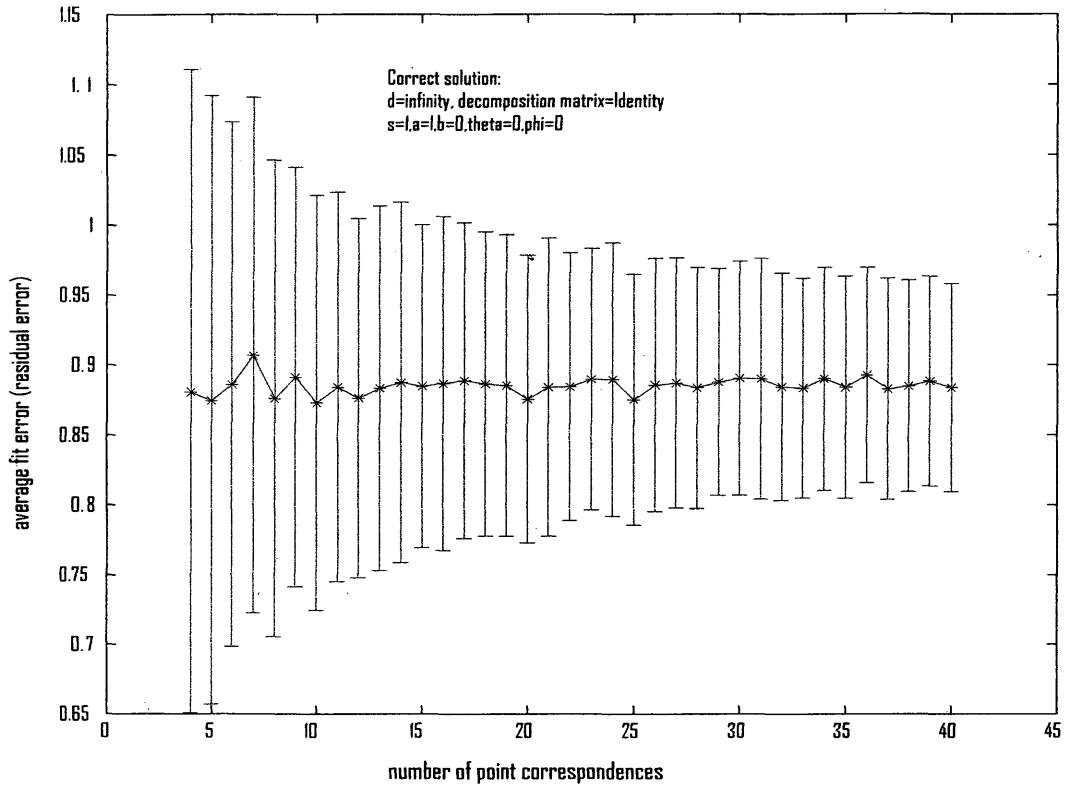


Figure 8h. Plot of the average fit error vs number of points

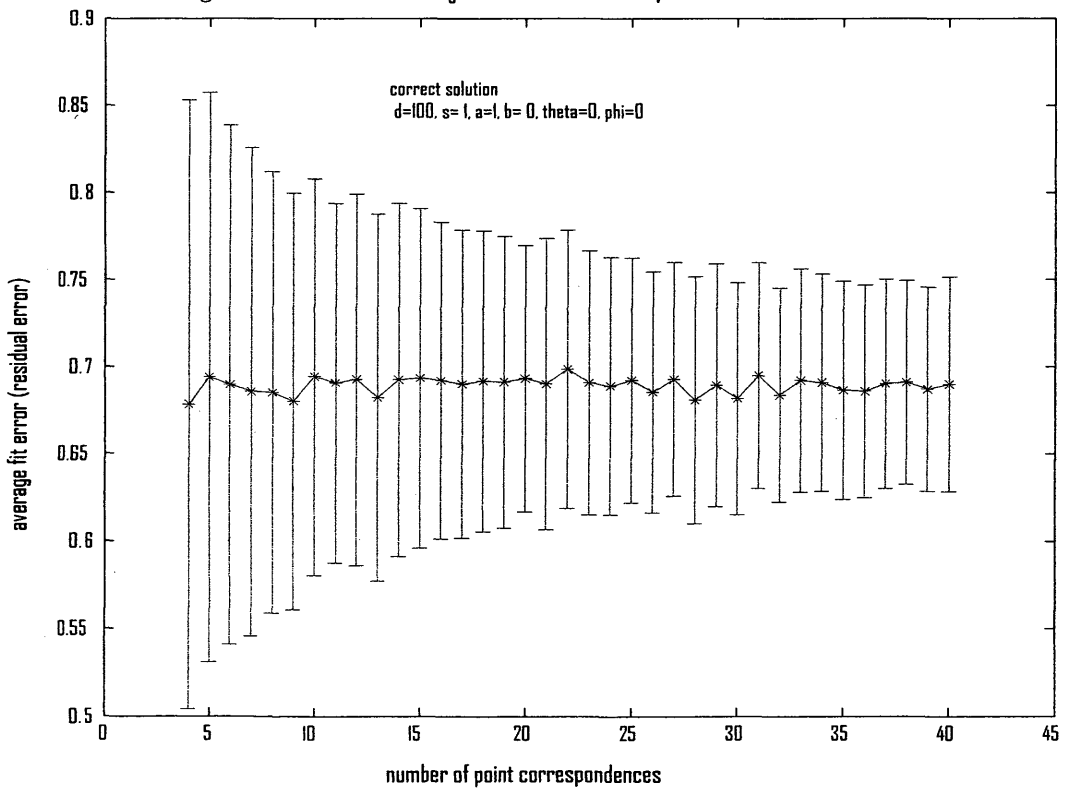


Figure 8i. Plot of the average fit error vs number of points

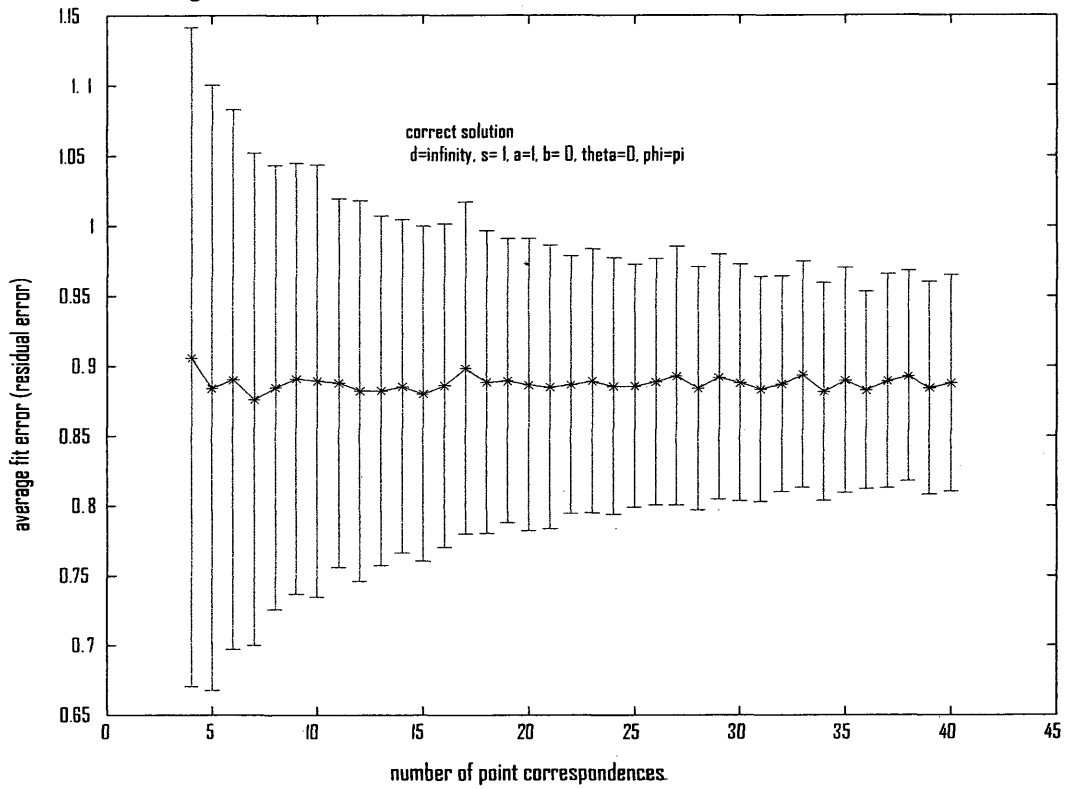


Figure 8j. Plot of the average fit error vs number of points

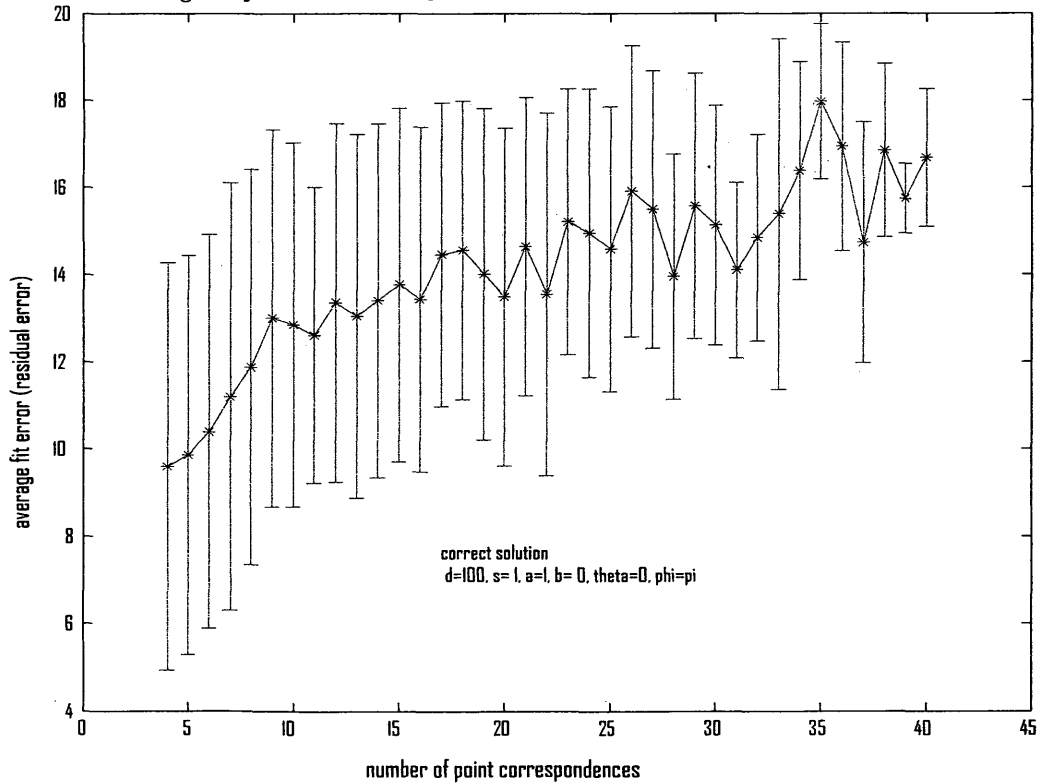


Figure 8k. Plot of the average fit error vs number of points

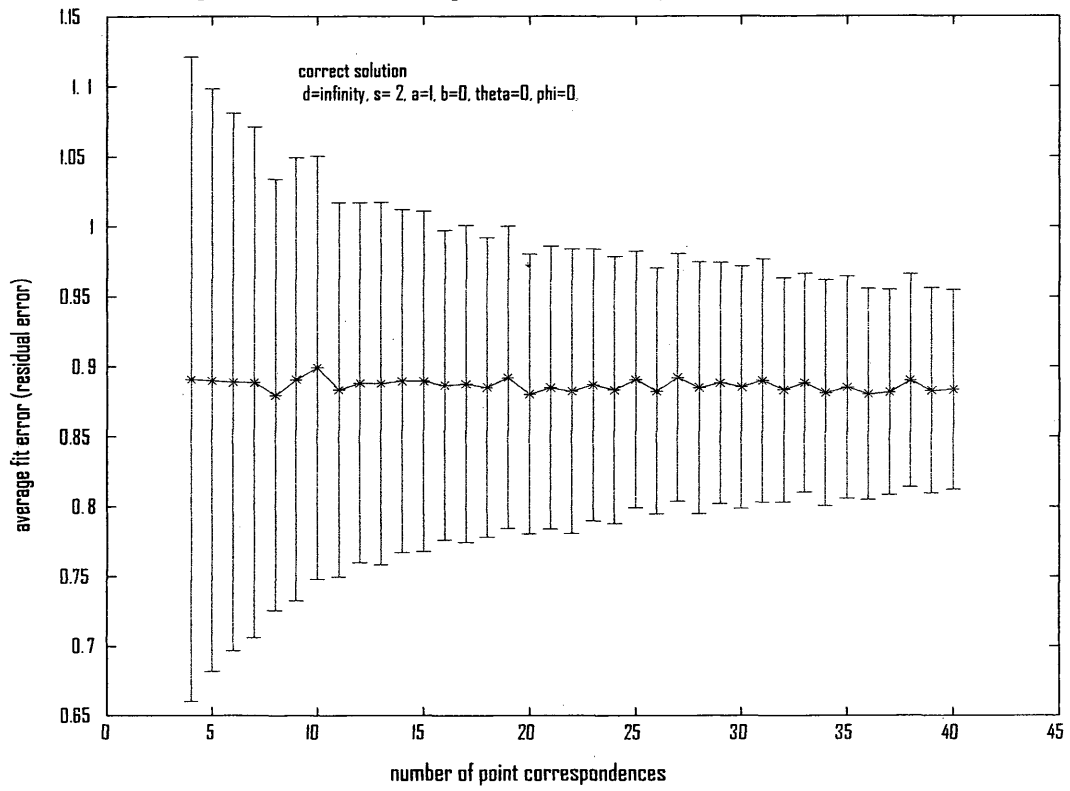


Figure 8l. Plot of the average fit error vs number of points

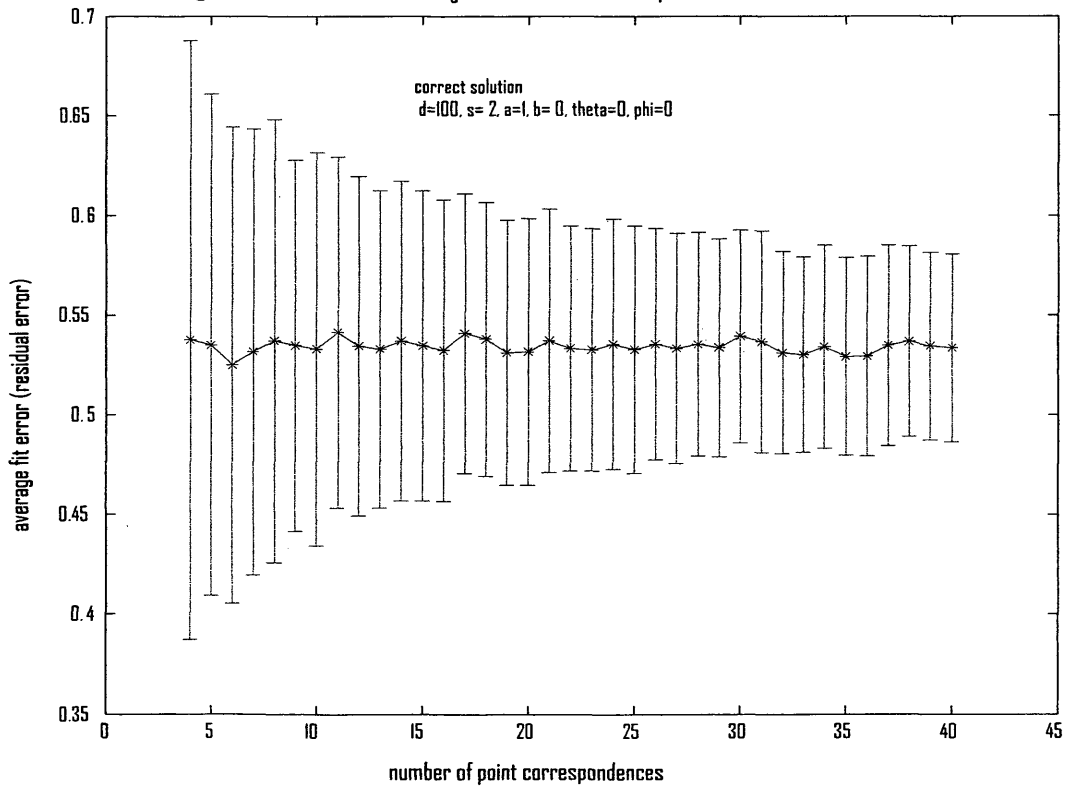


Figure 8m. Plot of the average fit error vs number of points

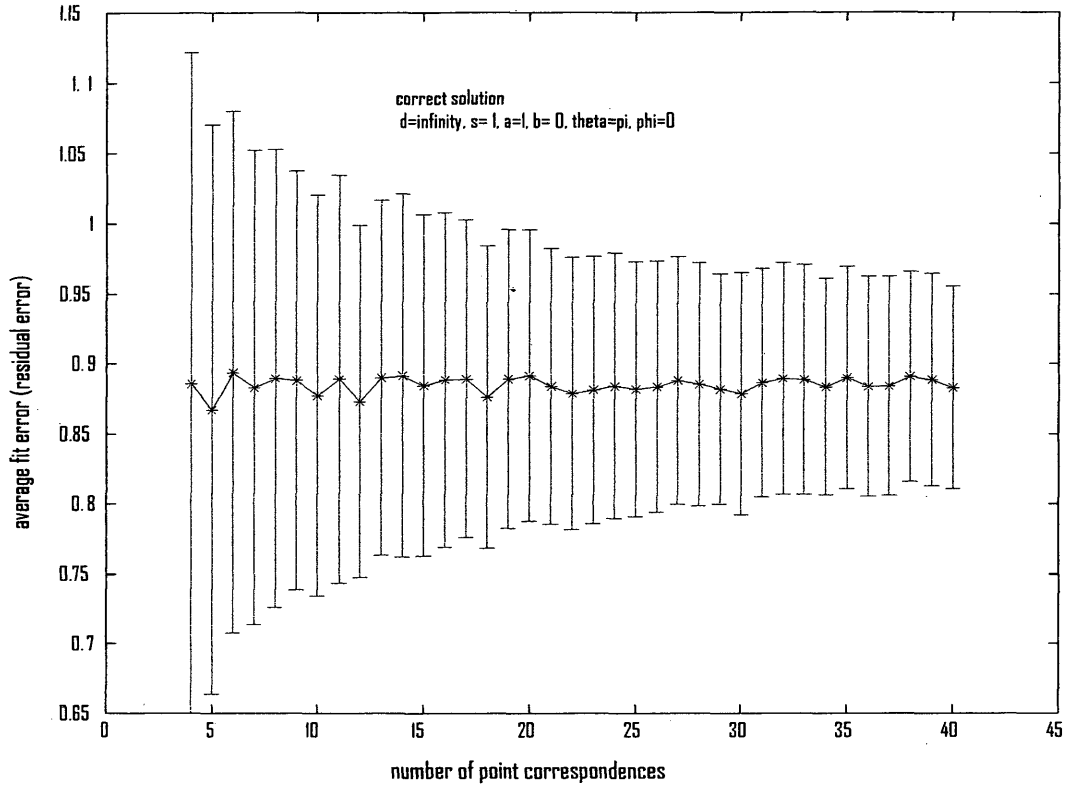


Figure 8n. Plot of the average fit error vs number of points

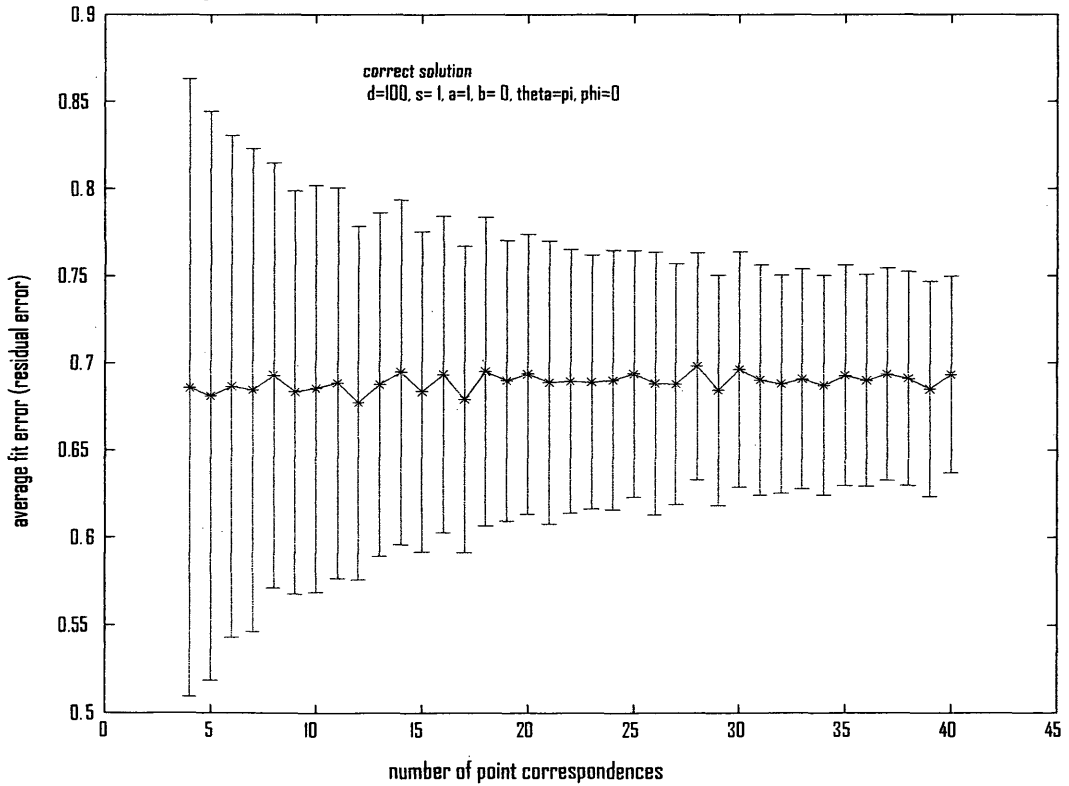


Figure 8o. Plot of the percent failures vs number of points

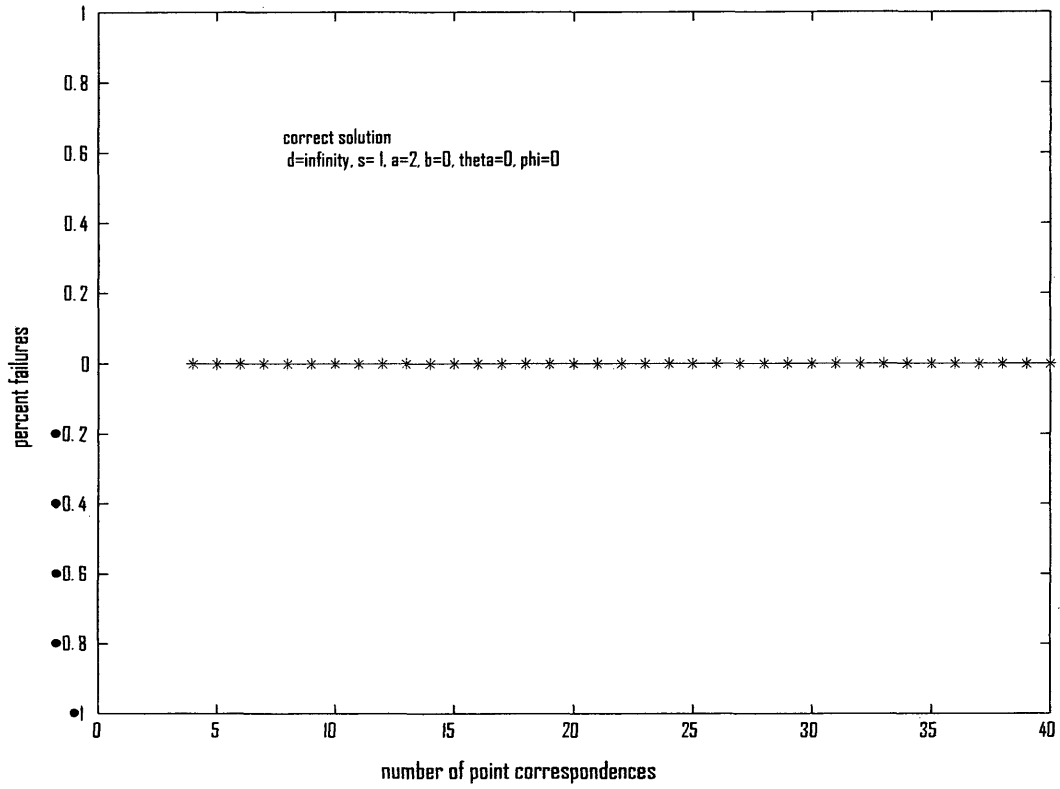


Figure 8p. Plot of the percent failures vs number of points

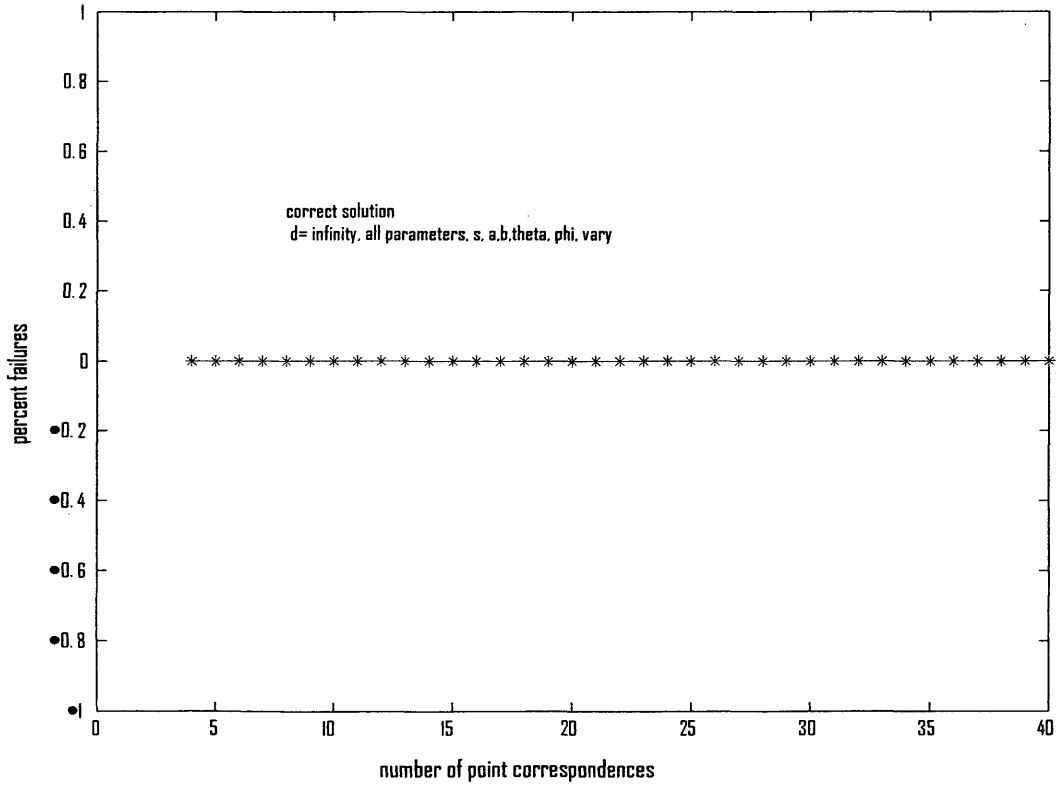


Figure 8q. Plot of the percent failures vs number of points

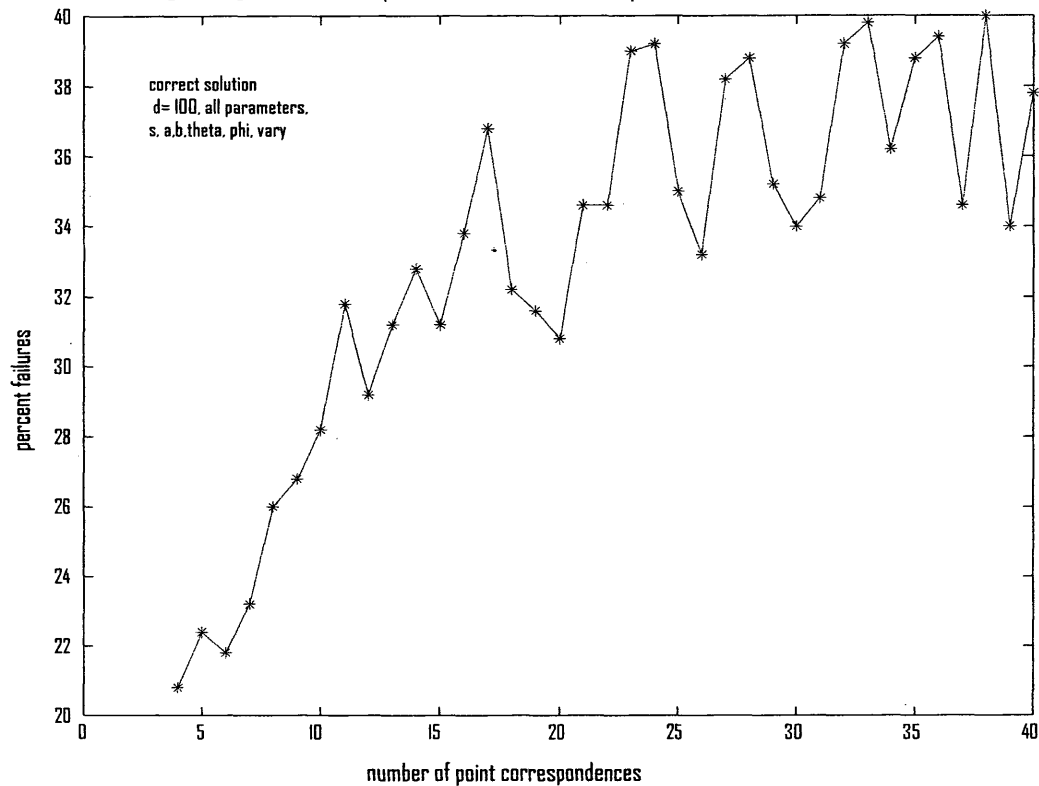


Figure 8r. Plot of the percent failures vs number of points

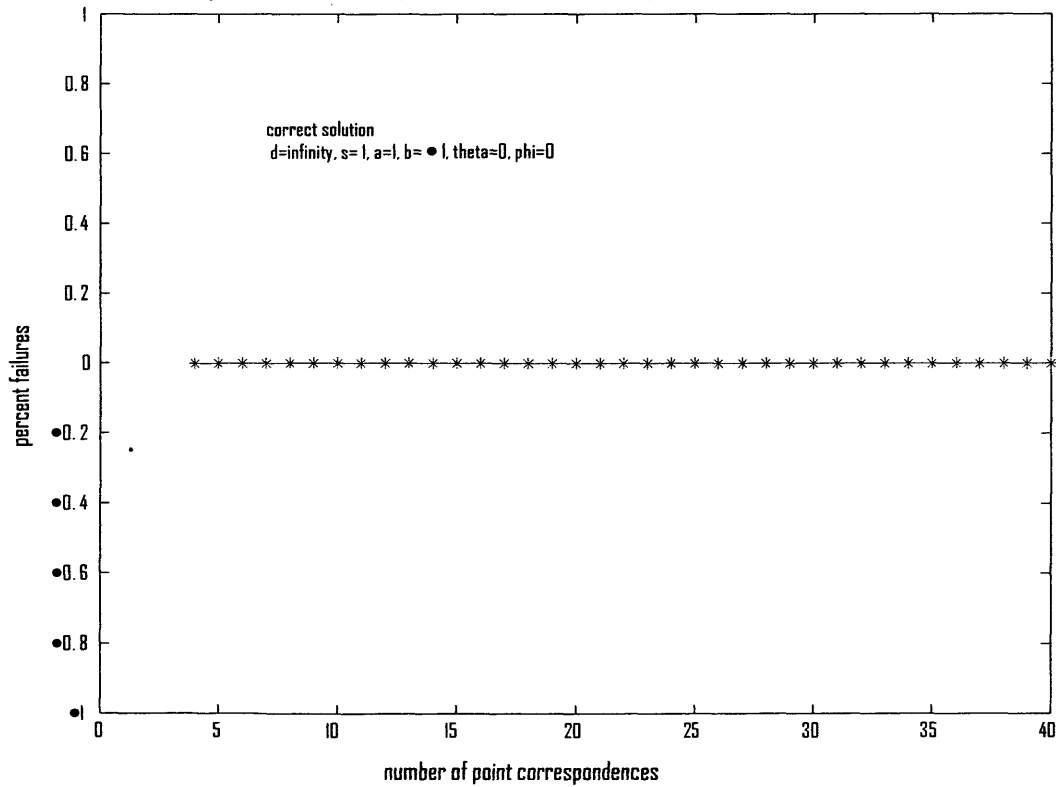


Figure 8s. Plot of the percent failures vs number of points

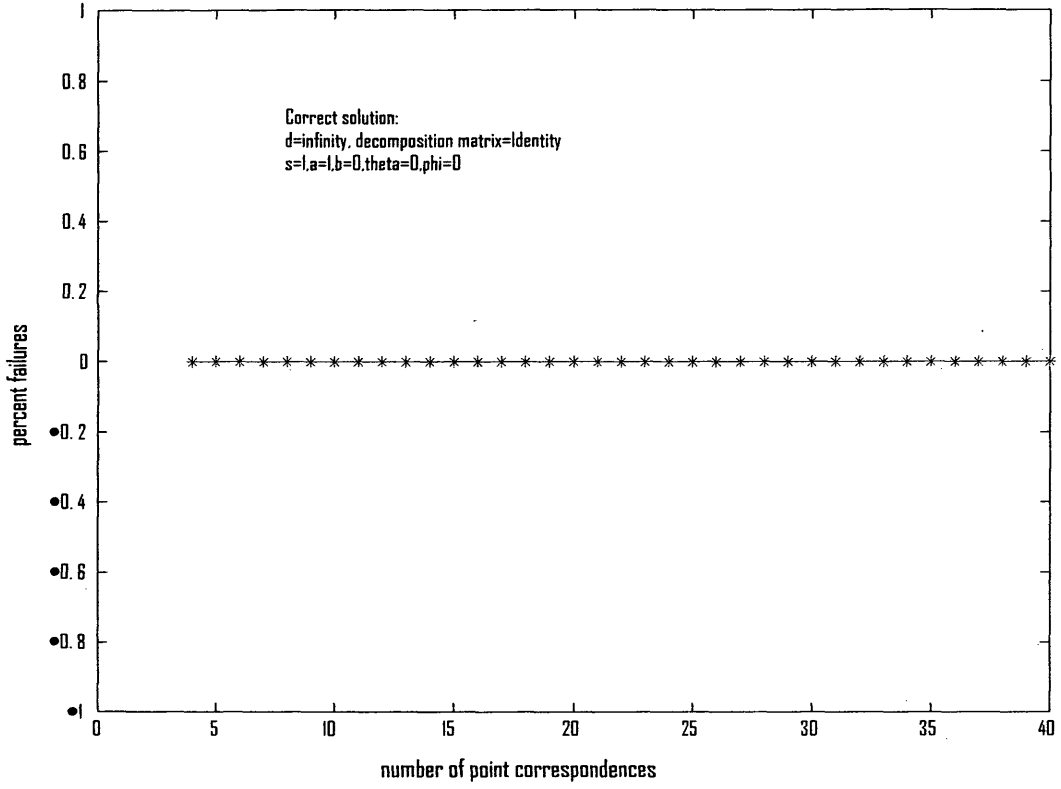


Figure 8t. Plot of the percent failures vs number of points

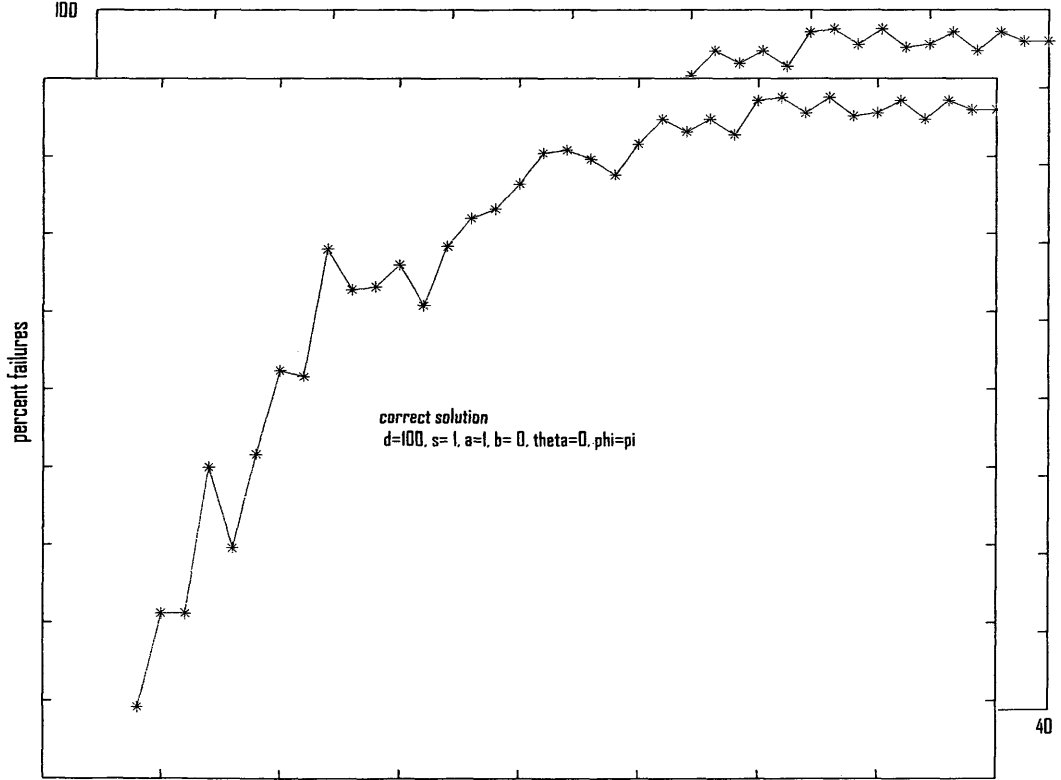


Figure 8u. Plot of the percent failures vs number of points

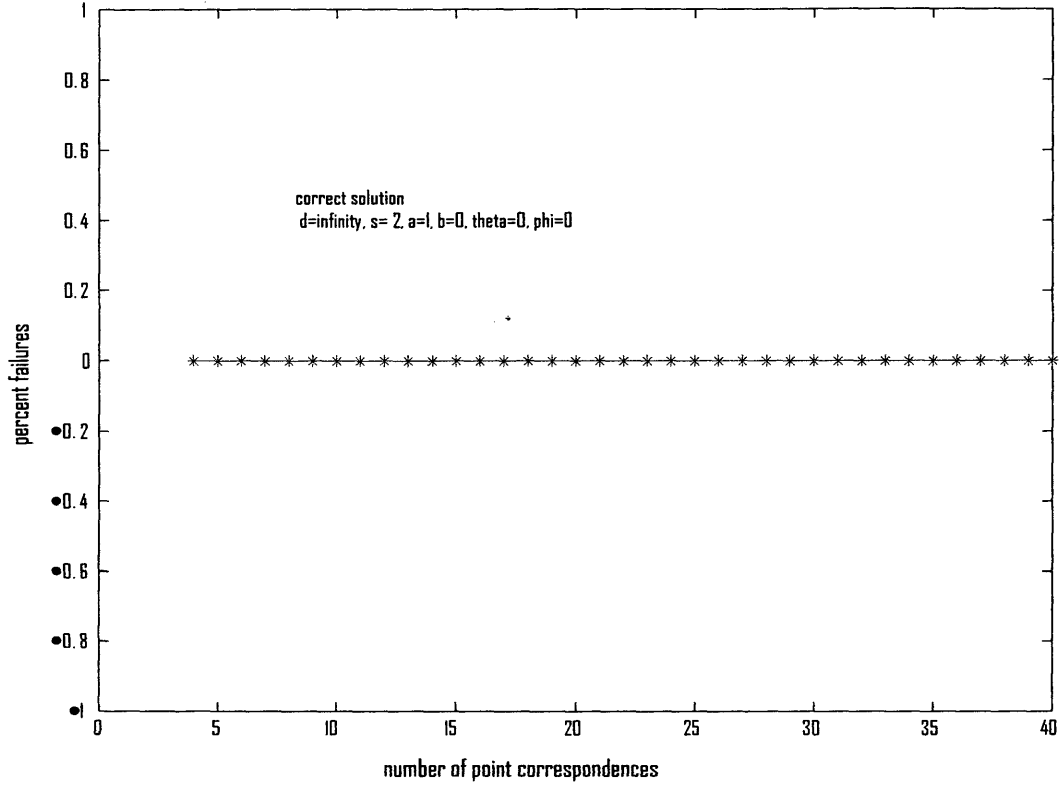


Figure 8v. Plot of the percent failures vs number of points

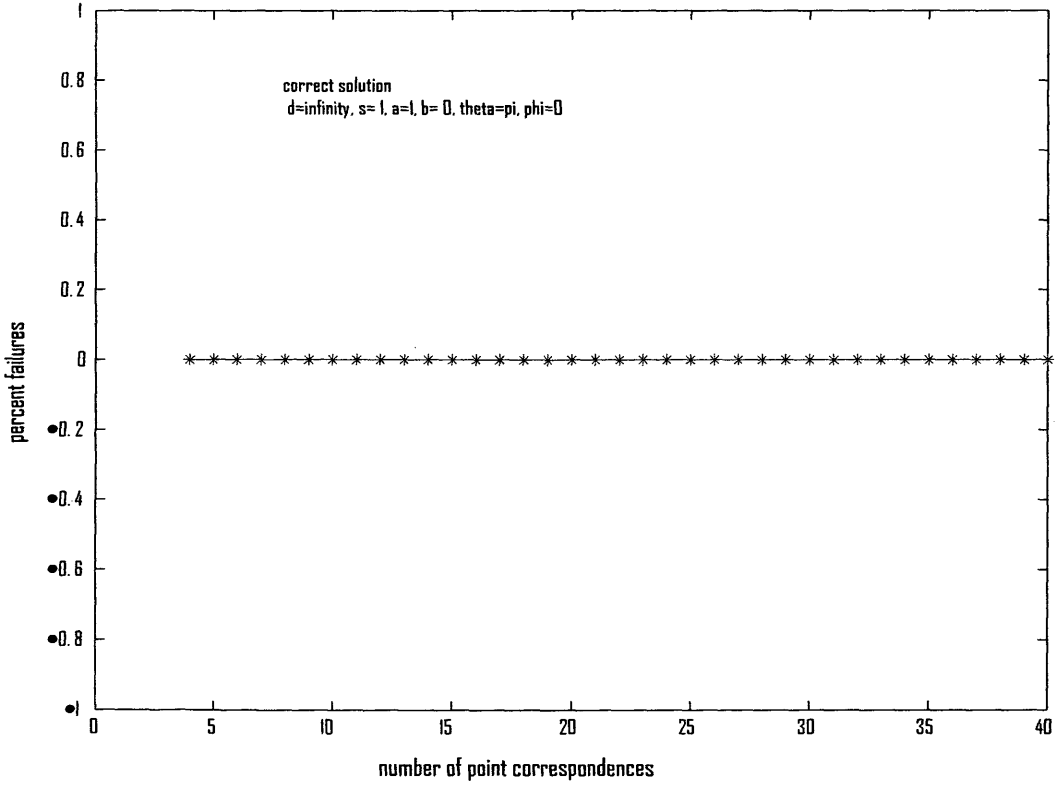


Figure 9. Case 6 -- GS solution with LM minimization

Figure 9a. Plot of the average fit error vs number of points

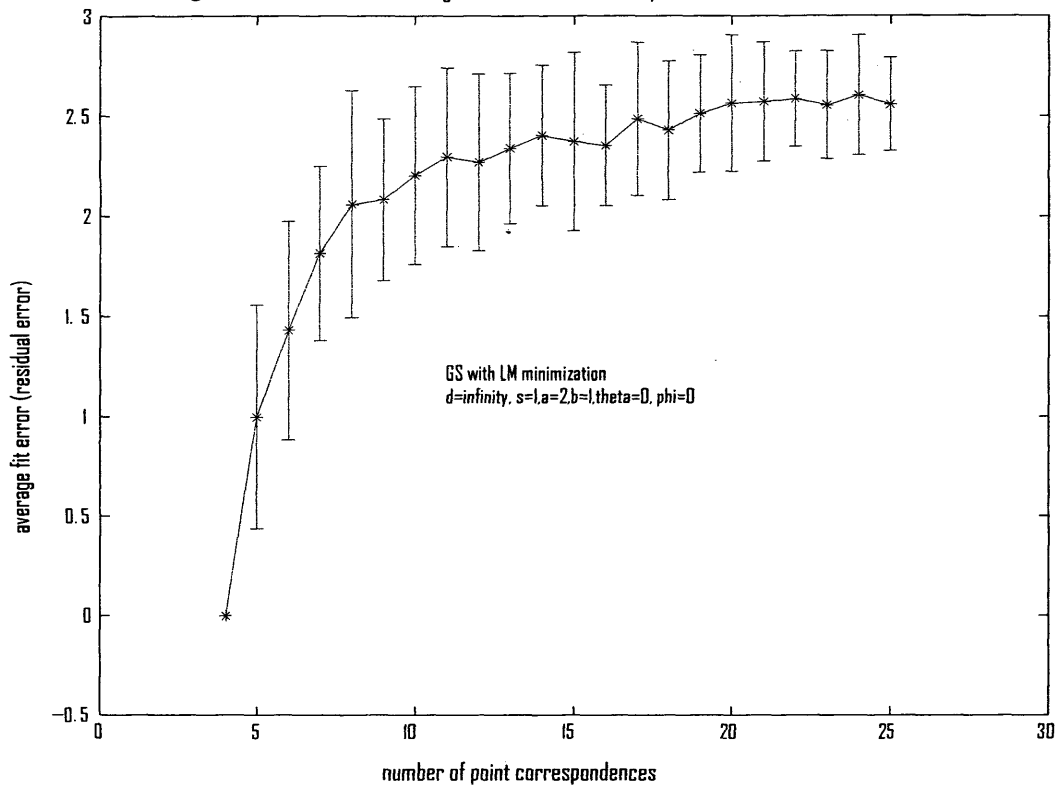


Figure 9b. of the average fit error vs number of points

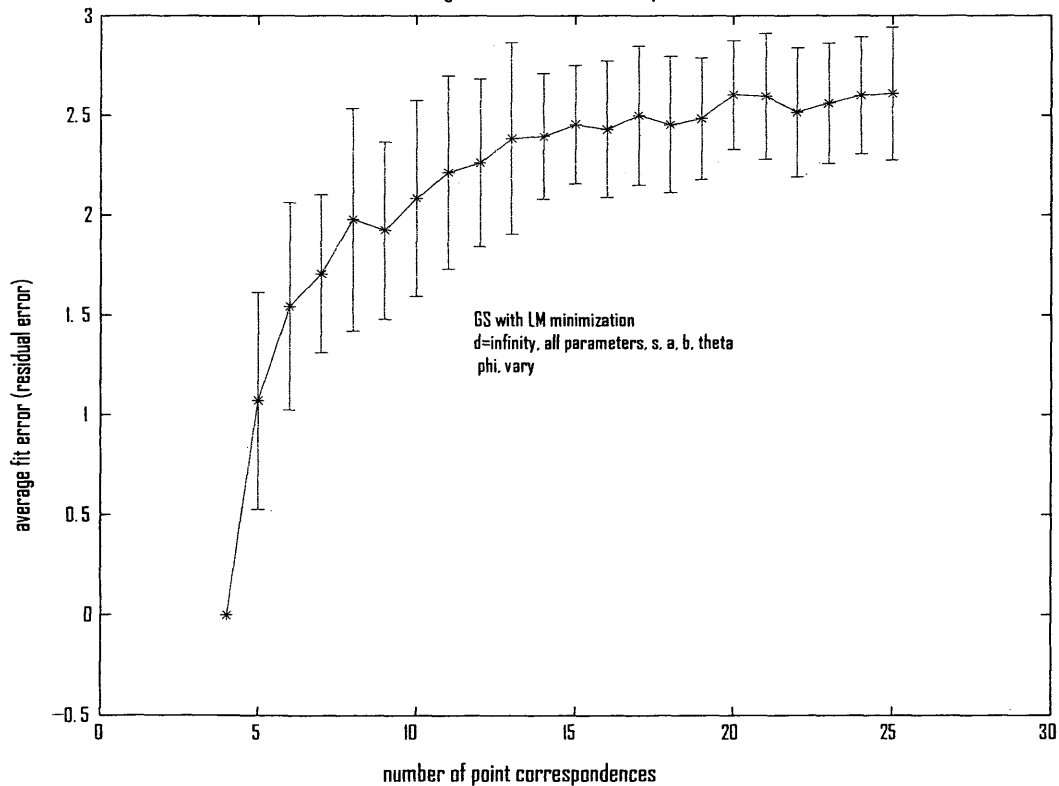


Figure 9c. Plot of the average fit error vs number of points

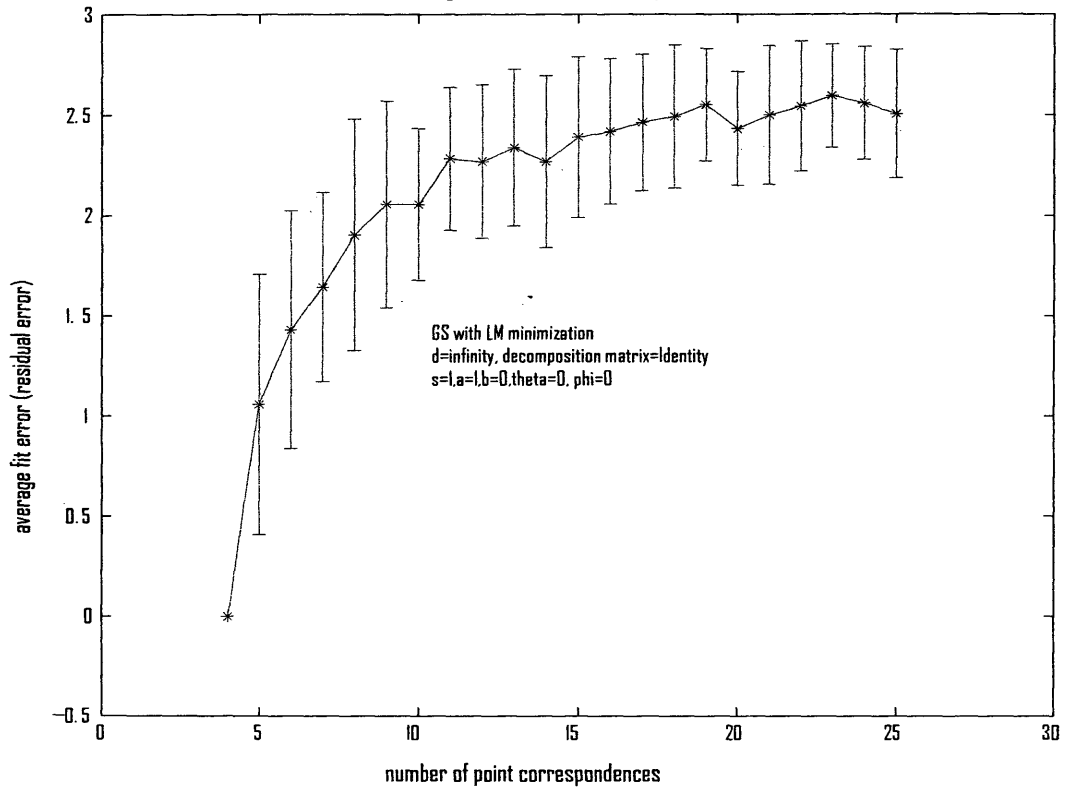


Figure 9d. Plot of the average fit error vs number of points

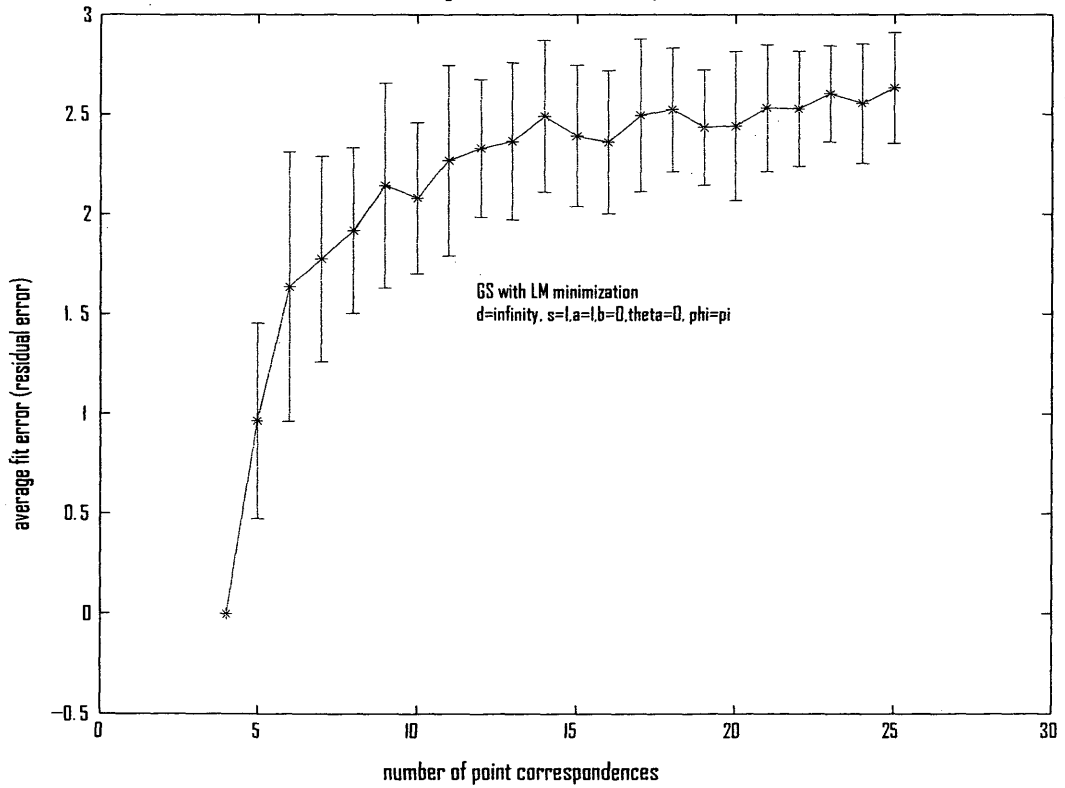


Figure 9e. Plot of the average fit error vs number of points

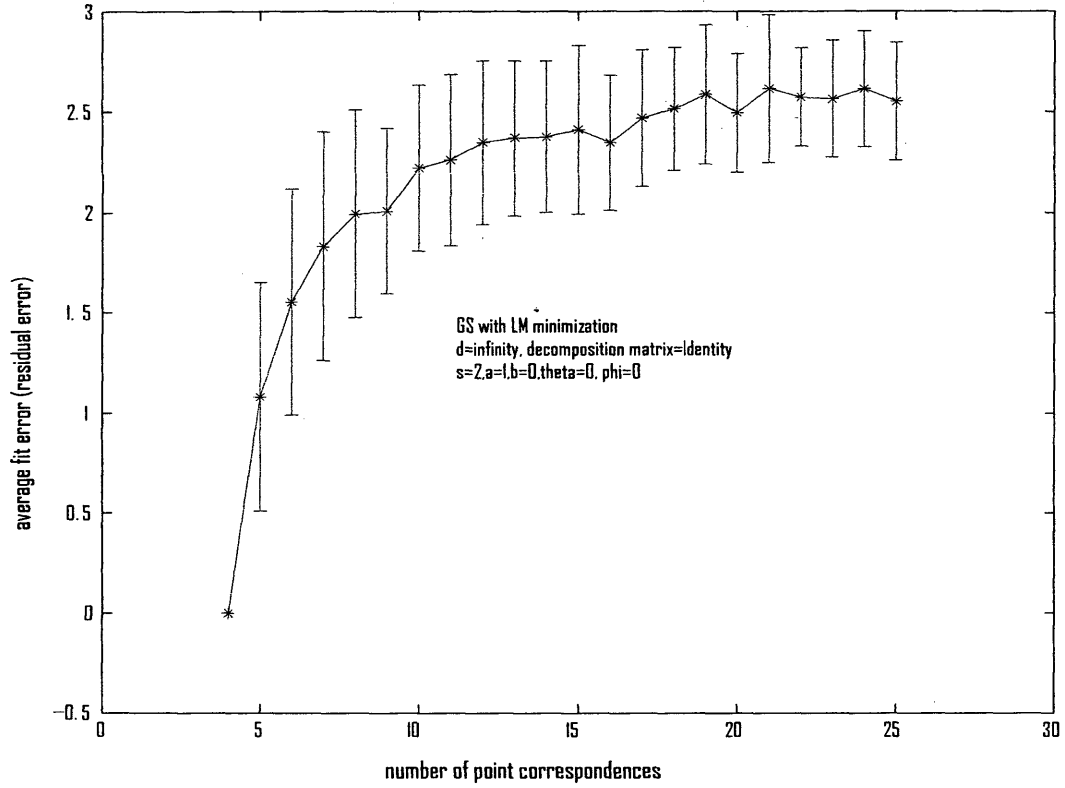


Figure 9f. Plot of the average fit error vs number of points

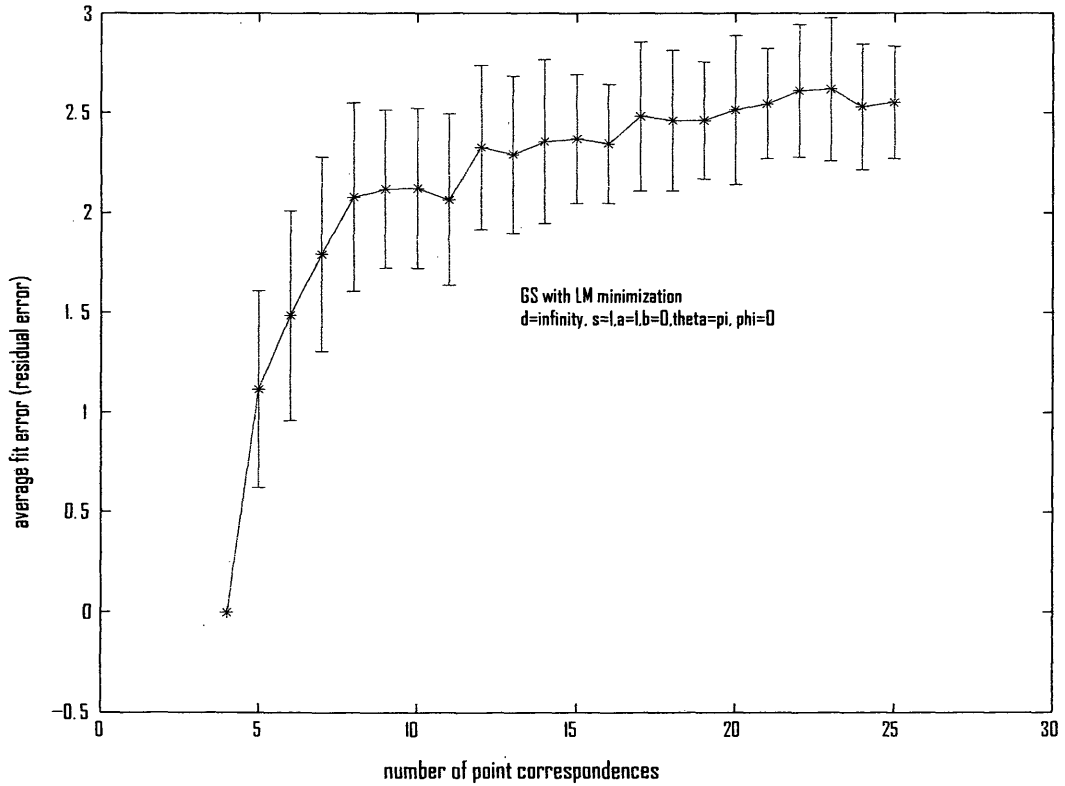


Figure 10. Case7 -- DLT algorithm without point normalization for affine case with d set to infinity

Figure 10a. Plot of the average fit error vs number of points

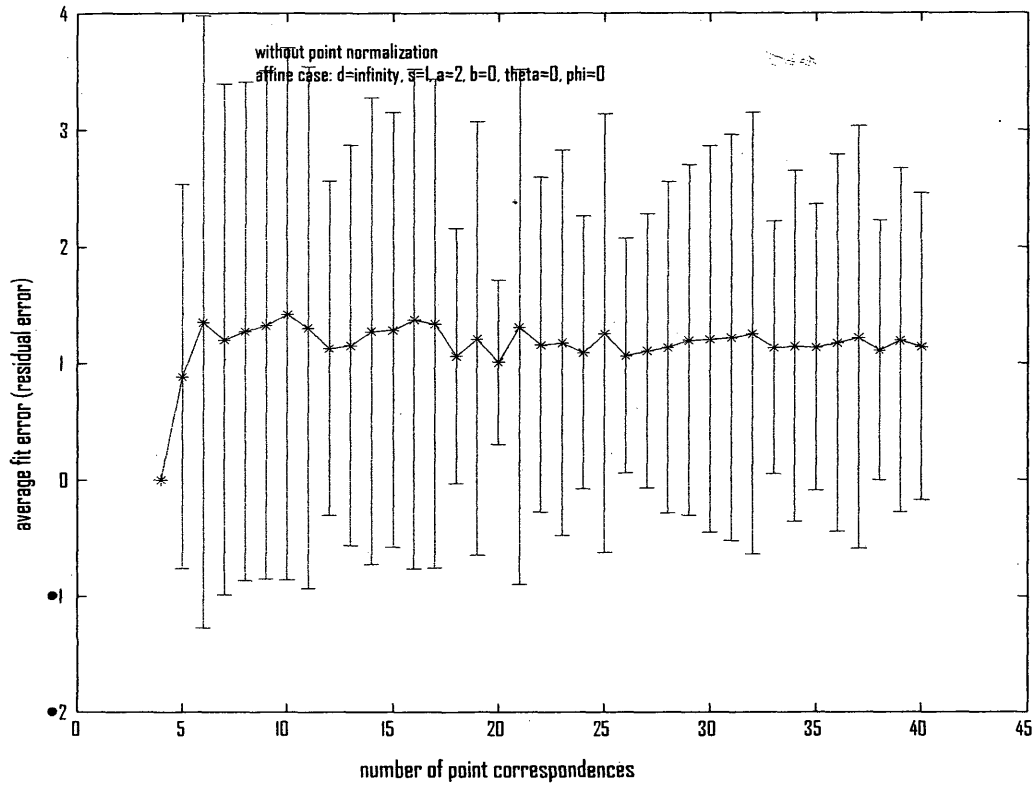


Figure 10b. Plot of the average fit error vs number of points

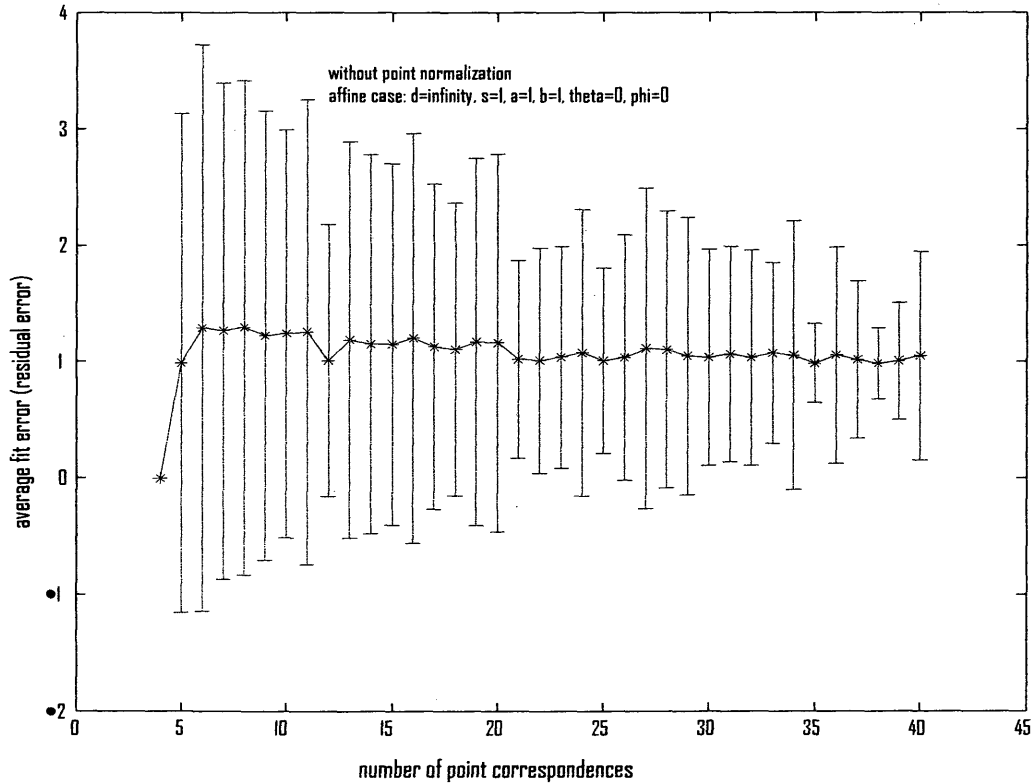


Figure 10c. Plot of the average fit error vs number of points

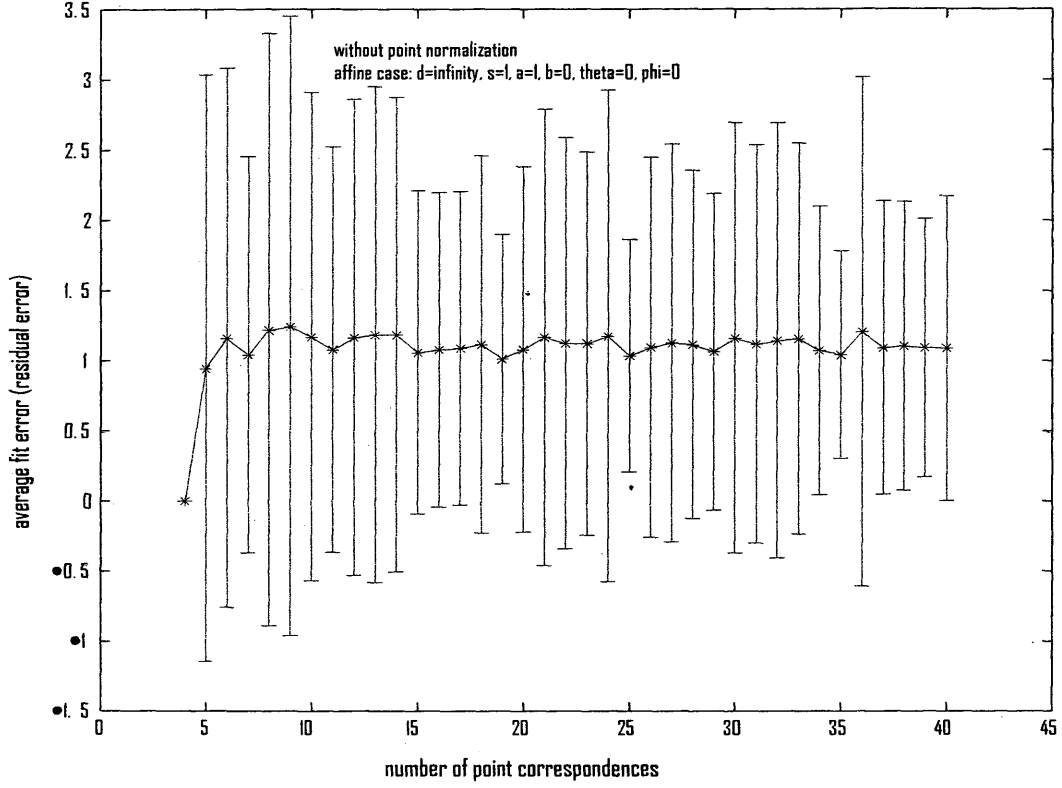


Figure 10d. Plot of the average fit error vs number of points

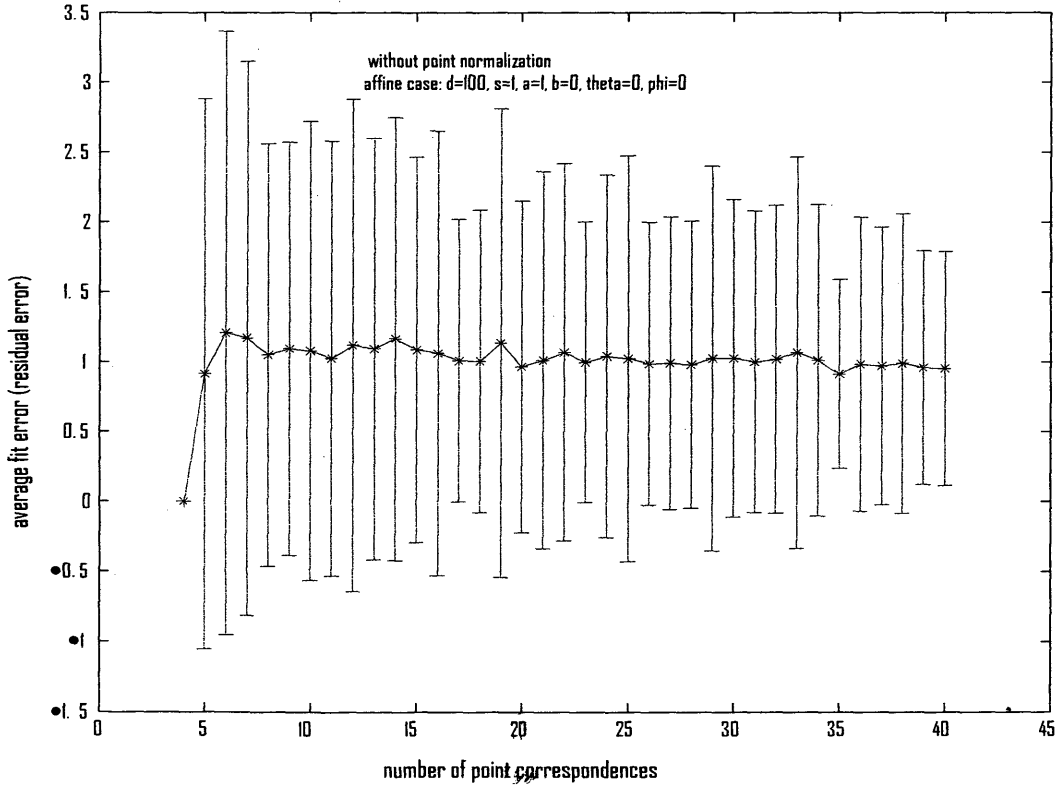


Figure 10e. Plot of the average fit error vs number of points

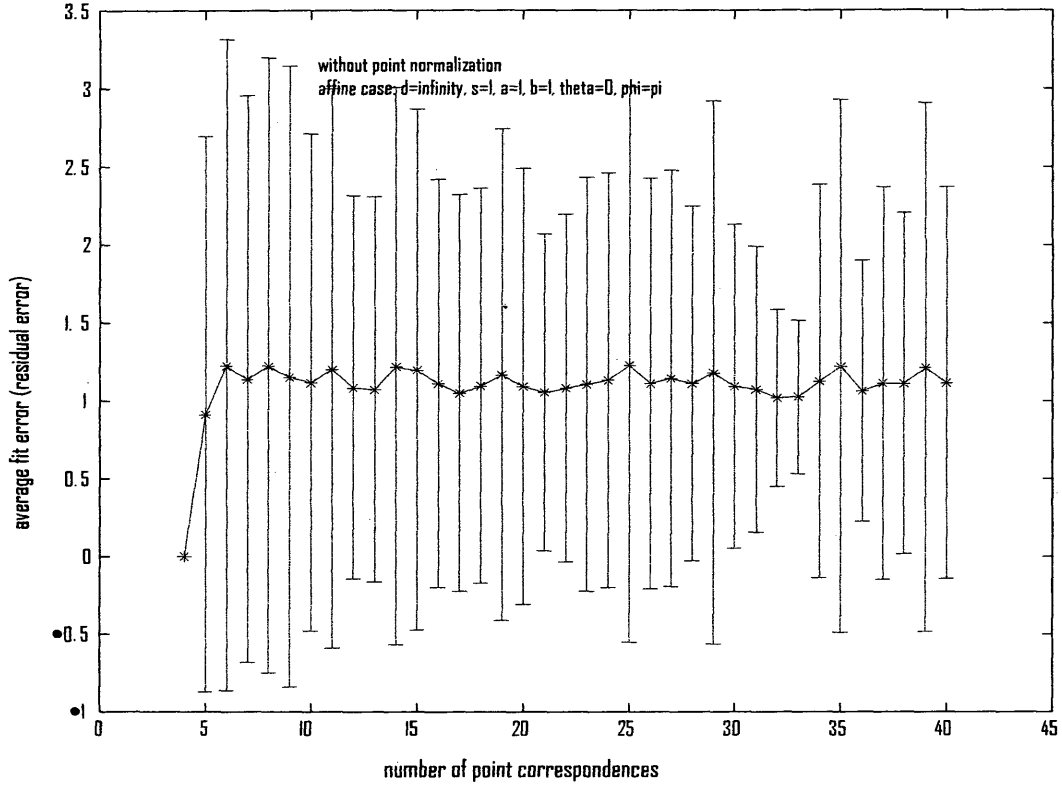


Figure 10f. Plot of the average fit error vs number of points

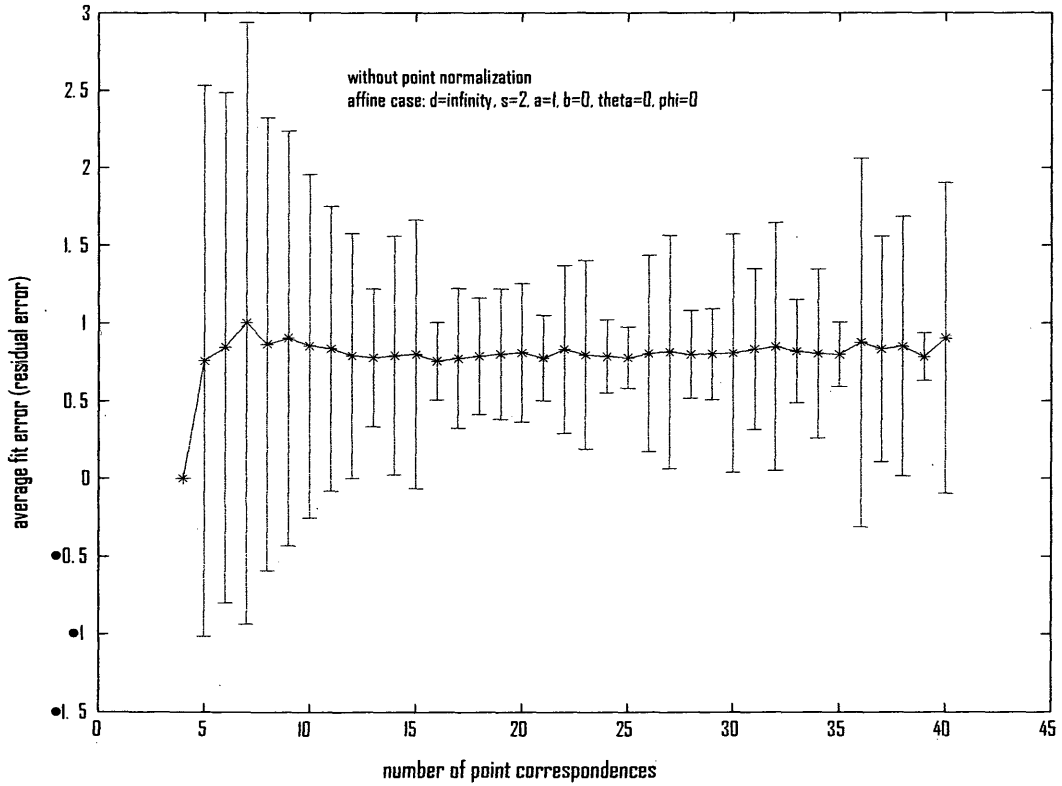


Figure 10g. Plot of the average fit error vs number of points

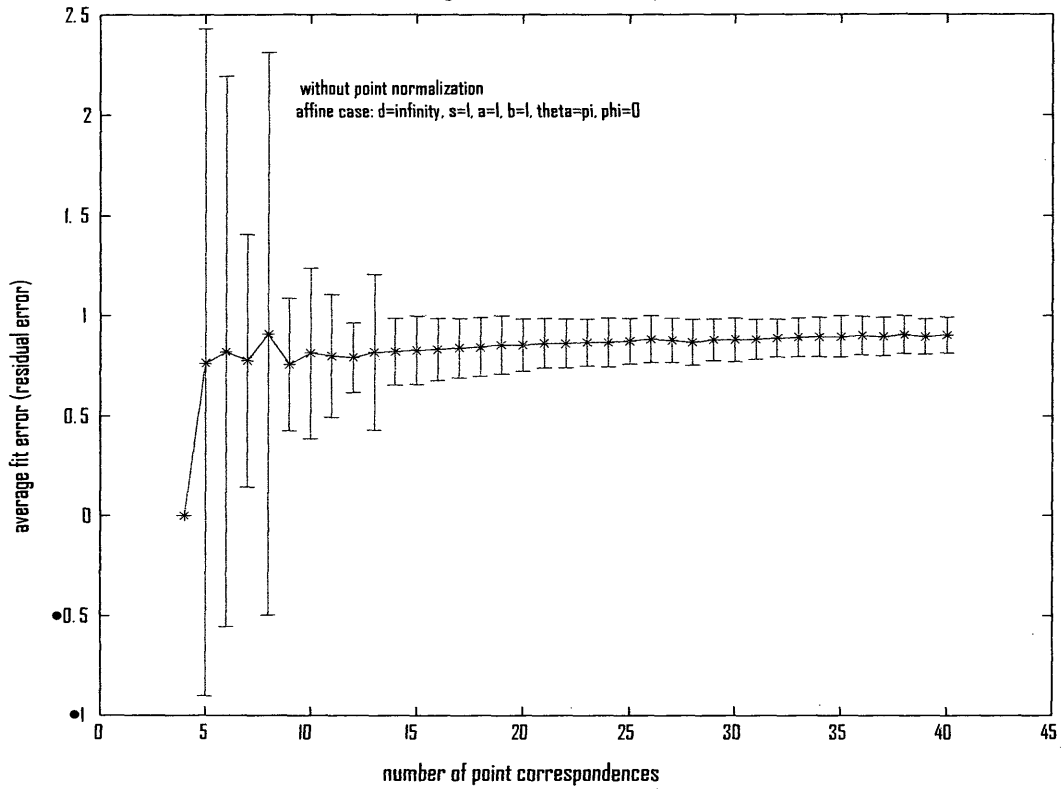


Figure 10h. Plot of the percent failures vs number of points

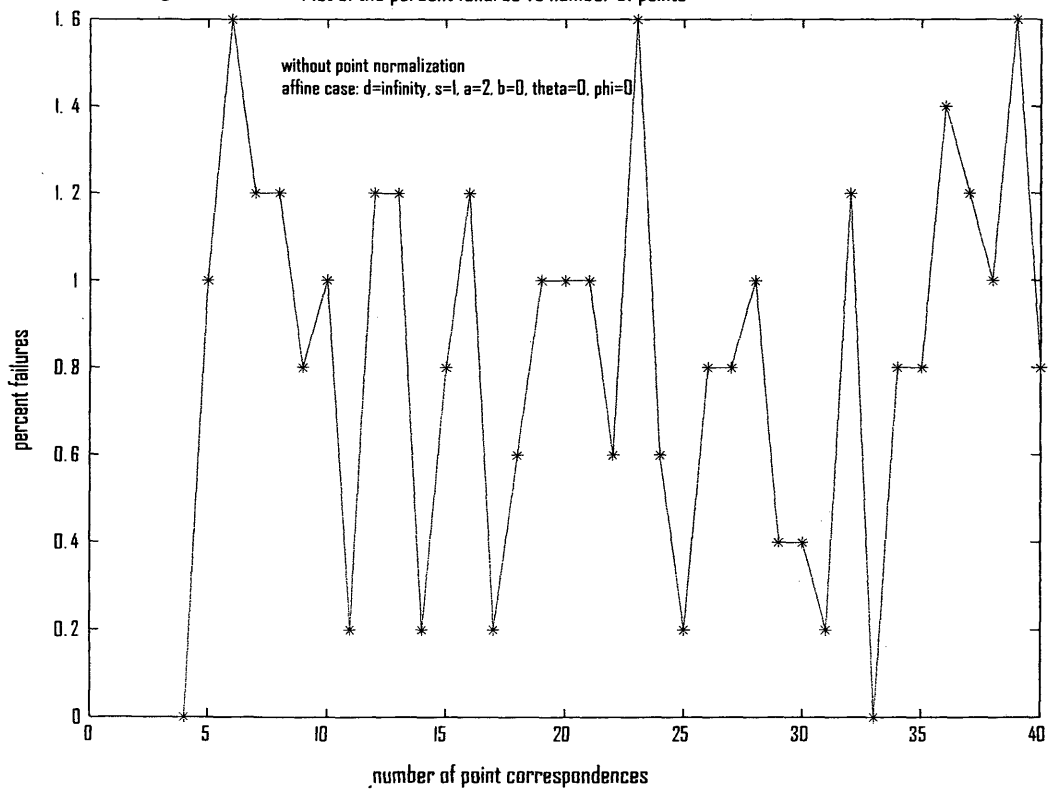


Figure 10i. Plot of the percent failures vs number of points

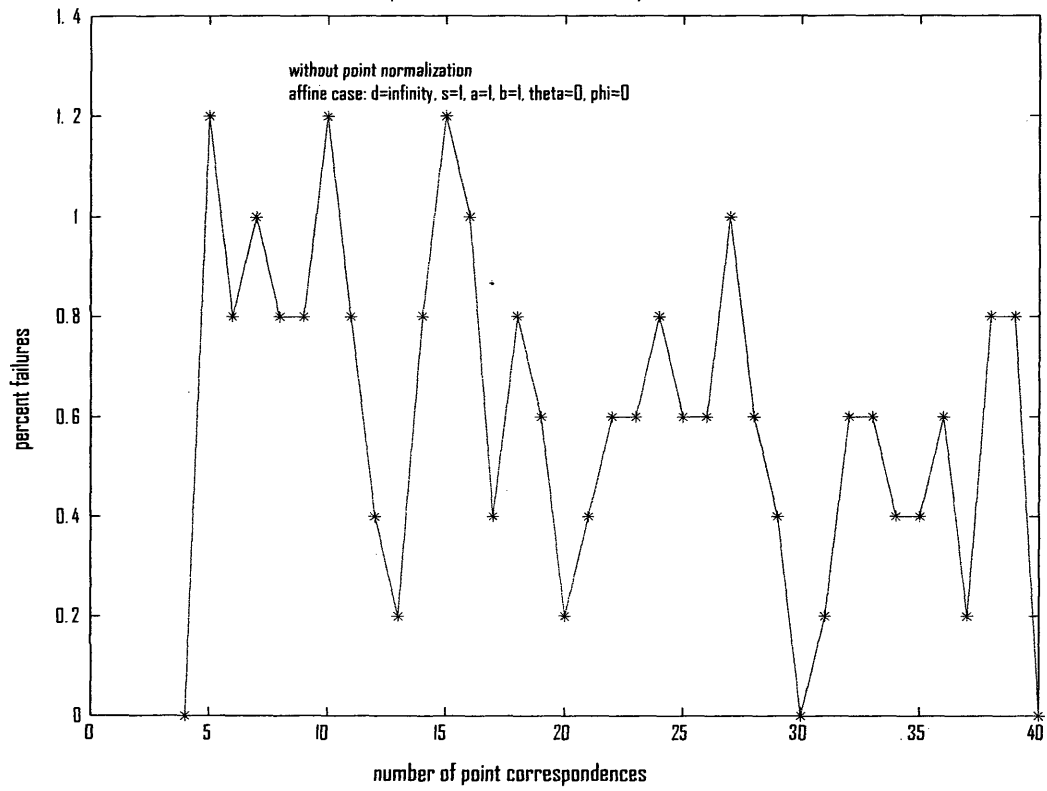


Figure 10j. Plot of the percent failures vs number of points

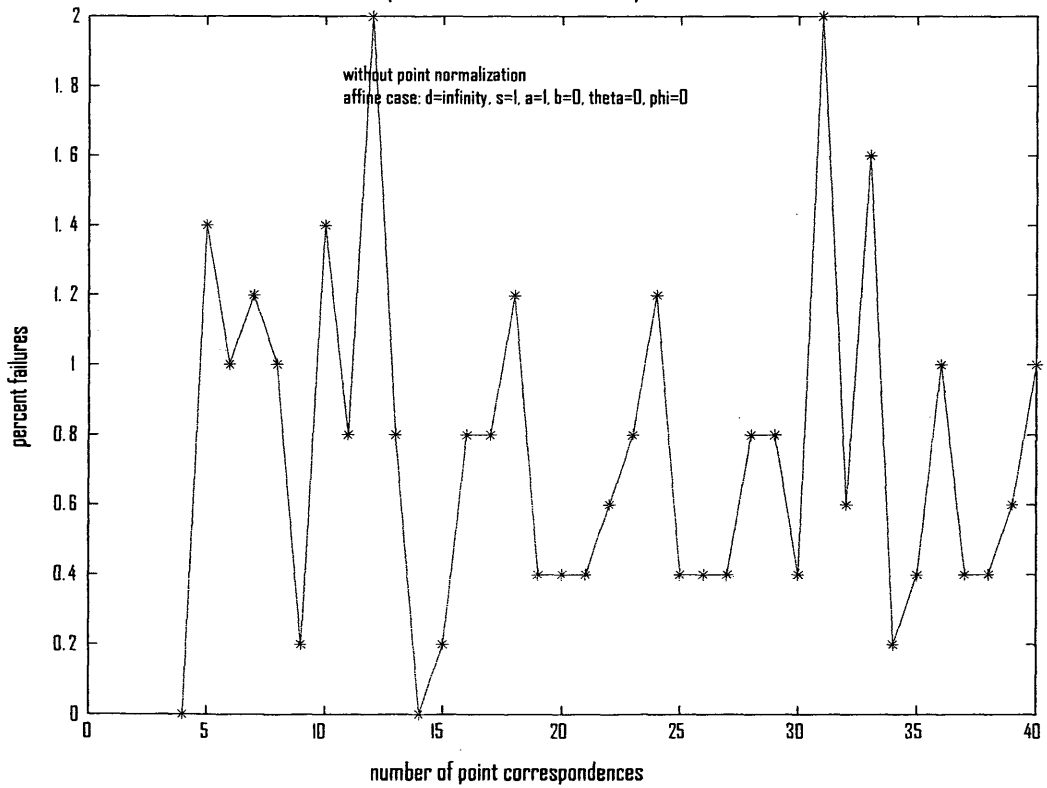


Figure 10k. Plot of the percent failures vs number of points

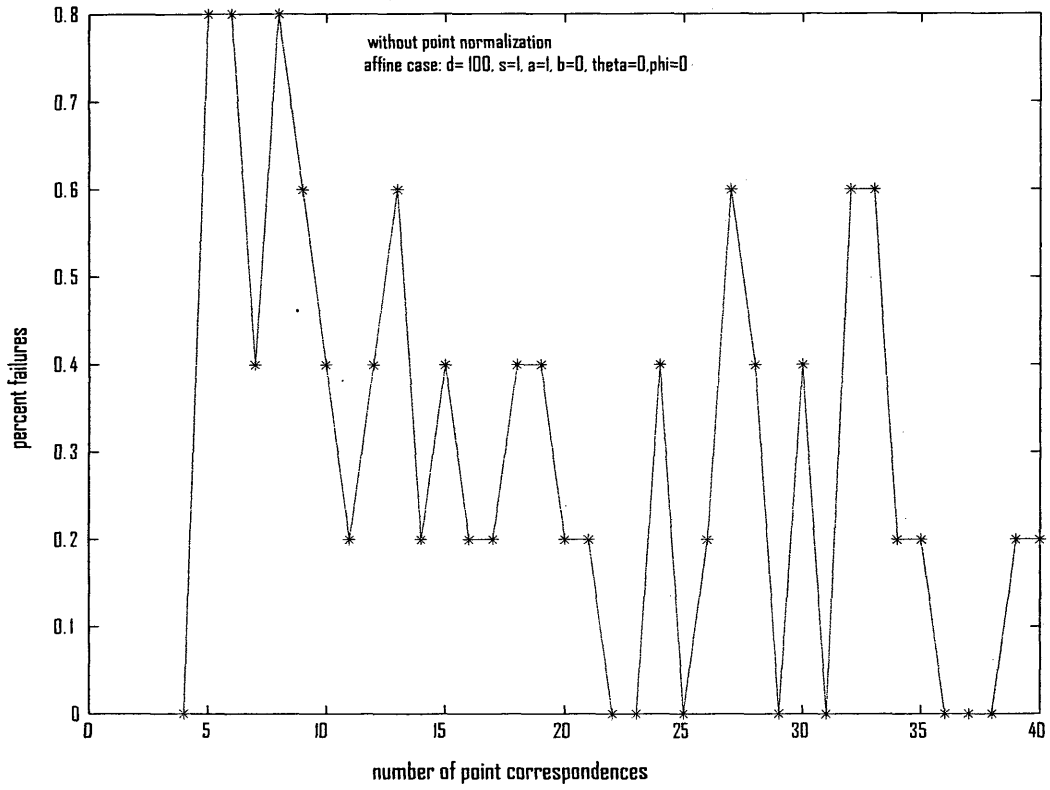


Figure 10l. Plot of the percent failures vs number of points

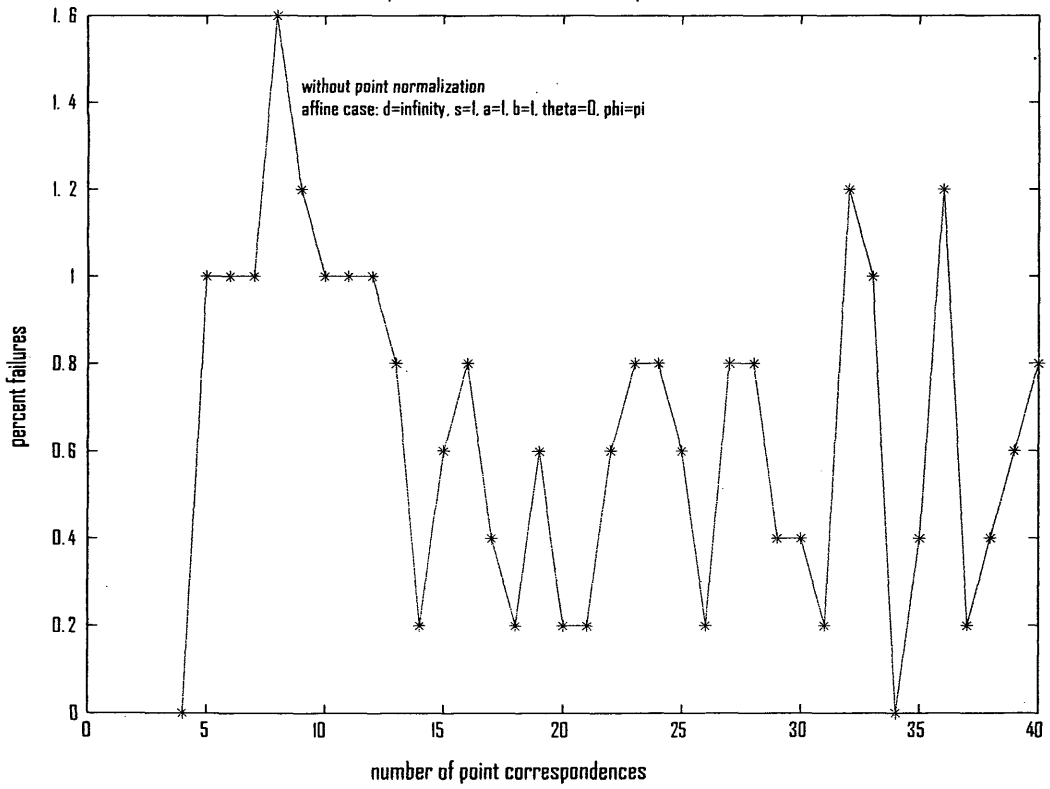


Figure 10m. Plot of the percent failures vs number of points

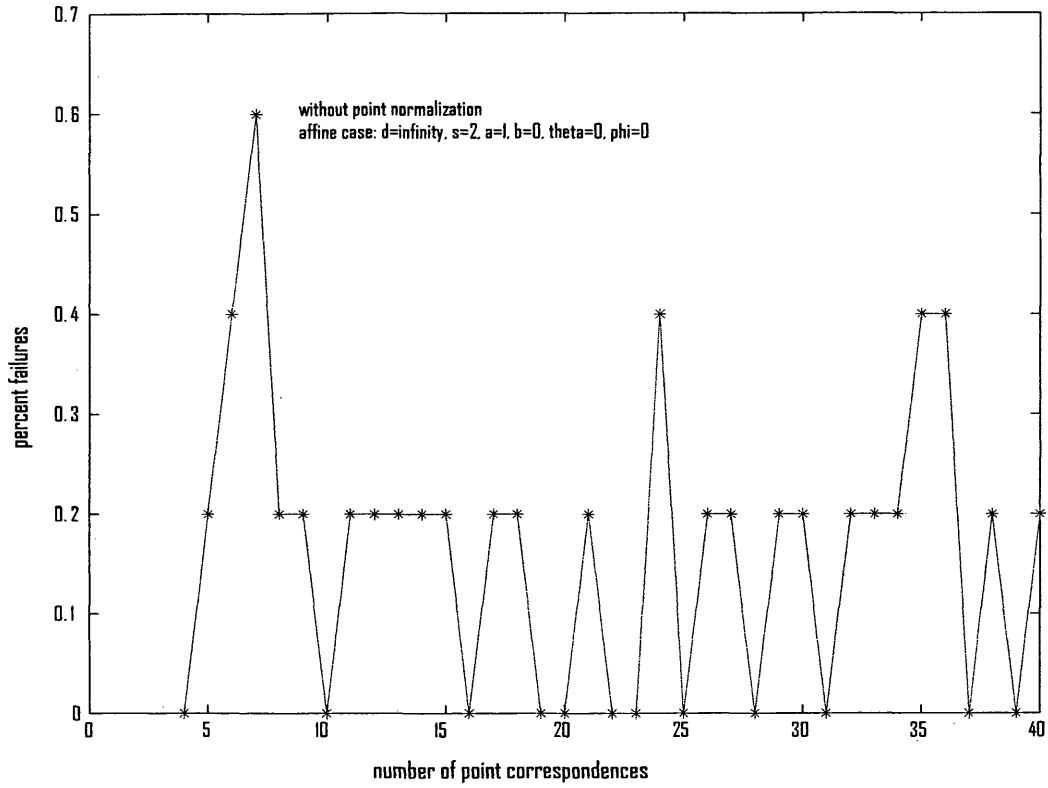


Figure 10n. Plot of the percent failures vs number of points

