

A Project on Circles in Space

Carl C. Cowen*
Purdue University

Preliminary Version
June 22, 1995

Abstract

This project asks students to decide if a collection of points in space do or do not lie on a circle. The project is accessible to linear algebra students who have studied Gram–Schmidt orthogonalization and the application of linear algebra to least squares regression. Several pedagogical aspects of the project are discussed including two successful approaches for the solution of the problem. Matlab code for generating data points is included.

The project of deciding if a collection of points in space do or do not lie on a circle has been very successful in my recent linear algebra classes. Students understand the problem and, given the background of the problem, the motivation for solving it. The project forces them to integrate their knowledge of several parts of the course and some earlier work in other courses. Moreover, since it sounds easier than it usually turns out to be, students get hooked on the problem. I believe they come to understand to some extent how the mathematics they know could be useful in a “real world” setting.

The project was inspired by a presentation given by Michael A. Lachance, University of Michigan at Dearborn, at the Tri-section Meeting of the Michigan, Illinois, and Indiana Sections of the Mathematical Association of America held at Saint Mary’s College (Notre Dame, Indiana) in the spring of 1993. Professor Lachance described this and other problems he had learned of in his consulting work with Ford Motor Company. I would like to thank Professor Lachance and Dr. Edward Moylan, Ford Motor Company, for their assistance in preparing this note.

*©Carl C. Cowen. Permission is granted for use of this material, with attribution, for instruction, but publication of this material requires prior written consent of the author.

1 The Project

In modern manufacturing plants, much of the work is done by robots. While robots are very efficient, they are less cognizant than humans of errors that have been made. To combat this, as part of the quality control process, measurements are taken to compare the actual outcome with the desired outcome. In this instance, the coordinates of ten points on the cut edge of the fuel tank filler tube (that is, a circular cylinder) have been measured. If the part is well made, these ten points will lie on a circle whose radius is the radius of the cylinder. If the part has been cut crookedly, these points are on an ellipse and if the piece has been bent, the points are on a bent circle. The question, therefore, is to determine whether ten points whose coordinates are given lie on a circle or whether they do not.

I have used this project in a second course in linear algebra for math majors, in a first course for sophomore engineering students, and in a graduate course for engineering students, a course that assumes no prior knowledge of linear algebra but moves quickly through material to culminate with quadratic forms and Jordan Canonical Form. In my class, each student receives her or his own data set but I encourage students to work together. Nevertheless, I ask that each problem be solved; that is, if Maria works alone, she needs to solve her own problem, but if Maria and John work together, they need to solve both Maria's problem and John's problem and they each get the average grade for the two submissions. The data sets I use are either circles, ellipses with eccentricity nearly 1, or circles that have been "bent", with a preponderance of circles.

Some kind of numerical software is essential for attacking this problem; students in my classes have access to MATLAB [1]. Ultimately, some numerical issues come up, in particular, the data points are given as numbers with four digit precision and it is quite unlikely that they lie on a circle to infinite precision. On this issue, I ask them to realize that in the "real world", they will be working with imprecise numbers and to use their own judgement as to whether the points "nearly" lie on a circle or whether they are "far" from being on a circle. We do not formally address the issue of "tolerance" to ask, for example, if the points are within .001" of being on a circle but I believe they could address that issue with more time. For the most part, however, the issues the students dwell on are the mathematical issues, not the numerical ones.

When the project is assigned, we will have covered least square approximations and the Gram-Schmidt orthogonalization algorithm and its relation

Circle Project

Decide if the following points in space lie on a circle. If they do, find the center, the radius, and describe the location of the circle in space. In any case, provide some supporting calculations and explain how your calculations justify your answer.

Data Set 38			
	x	y	z
point 1	0.4155	1.4804	-4.1936
point 2	2.5749	7.7438	2.6115
point 3	-0.8669	7.1880	1.0171
point 4	5.3092	6.7052	2.4255
point 5	1.4171	1.0525	-4.3084
point 6	6.4762	5.6856	1.7804
point 7	4.7269	7.0463	2.5828
point 8	-1.7171	6.4185	0.0029
point 9	1.7591	0.9468	-4.3078
point 10	6.4560	5.7093	1.7975

Figure 1: A typical assignment sheet for the circle project.

to QR-factorization. In more advanced courses, we also will have covered unitary transformations of \mathbf{R}^n and \mathbf{C}^n . I typically give the students two weeks to think about the problem and we discuss various issues as students have questions. On the class after the assignment is made, I ask the students to discuss the question “What is a circle?” among themselves. Frequently, this question is asked of me at the beginning of that class, but I do not answer it directly. The first major hurdle is passed when the students consciously state that a circle is a plane curve.

Students must eventually realize that in linear algebra we have thought only of planes as two dimensional subspaces, that is, we have thought only about planes through the origin. In particular, finding the rank of the matrix whose columns are the data points is not a good way to decide if the points lie on a plane! Eventually, student questions lead us to discuss ways to translate the data so that, if the points lie on a plane, the translated data will lie on a plane through the origin. We usually compare the merits of the most obvious technique of subtracting one of the points from each of the points with the more subtle technique of subtracting the average of the points. Additionally, I expect students to remember that planes in \mathbf{R}^3 have an equation of the form $ax + by + cz + d = 0$, to realize that this is too flexible a formulation, and to consider using a least squares technique to find the best plane of the form

$$ax + by + cz + 1 = 0$$

Students usually get this far with little assistance from me beyond providing time in class for the discussion.

The next step is frequently more problematic. Most students do not have a firm grasp on the idea that orthogonal coordinate systems are essential for distinguishing circles from ellipses. That is, to decide whether the points on the plane are on a circle, it is important to get an orthonormal basis for the plane, not just any basis, because circles are defined in terms of distances and coordinates with respect to an orthonormal coordinate system allow one to compute the distances with the new coordinates. To make this point, one can give examples of a circle described in a non-orthogonal basis giving coordinates that form an ellipse if interpreted as points in the plane in the usual way. This discussion is more easily carried out if the students have covered changing coordinates and especially orthogonal change of coordinates before the project. We will have done so for our more advanced students, but not for the sophomore engineering course. After describing the points in terms of an orthonormal coordinate system, I expect them to use least squares to find the circle that fits best.

Students find this a challenging but interesting project. In my experience, most students choose to work in small groups, and most groups are fairly successful in developing a suitable technique to attack the problem, although not all carry out the computations carefully enough to get correct answers. In the past, I have given some data sets whose points did not all lie in a plane, but students regarded this as unfair because it meant that those students had much less work to do to finish the project; in the future, all the data sets I use will have points that are coplanar. I regard this as one of the most successful projects I have included in courses because it asks students to use a variety of tools to attack an easily understood problem whose practical significance they can appreciate.

2 A Solution

The following is an outline of a solution to the problem in Figure 1 typical of the approach used in my more elementary class. It is based on translation of the data points to a plane through the origin and using an orthogonal coordinate system in that plane to fit a circle to the data. The calculations are from a MATLAB session on a Macintosh.

First, the data must be entered, most conveniently as the 3×10 matrix whose columns are the data points.

```
data =
Columns 1 through 7
    0.4155    2.5749   -0.8669    5.3092    1.4171    6.4762    4.7269
    1.4804    7.7438    7.1880    6.7052    1.0525    5.6856    7.0463
   -4.1936    2.6115    1.0171    2.4255   -4.3084    1.7804    2.5828
Columns 8 through 10
   -1.7171    1.7591    6.4560
    6.4185    0.9468    5.7093
    0.0029   -4.3078    1.7975
```

To move the plane to the origin, we will subtract from each point, the average of the points.

```
> avgpt=data*ones(10,1)/10
avgpt =
    2.6551
    4.9976
   -0.0592
```

```

> trdata=data-diag(avgpt)*ones(3,10)
trdata =
  Columns 1 through 7
  -2.2396   -0.0802   -3.5220    2.6541   -1.2380    3.8211    2.0718
  -3.5172    2.7462    2.1904    1.7076   -3.9451    0.6880    2.0487
  -4.1344    2.6707    1.0763    2.4847   -4.2492    1.8396    2.6420
  Columns 8 through 10
  -4.3722   -0.8960    3.8009
   1.4209   -4.0508    0.7117
   0.0621   -4.2486    1.8567

```

If the translated data points actually lie on a plane, then the rank of `trdata` should be 2. In fact, the rank is given as 3, but if `eps` is changed from the default value of 2×10^{-16} to about 10^{-6} , then the rank is reported as 2; in other words, the matrix is nearly rank 2, but not rank 2 to the accuracy MATLAB usually expects. Use of the QR-factorization makes the situation much clearer!

```

> [Q R]=qr(trdata)
Q =
  -0.3814    0.8996   -0.2125
  -0.5990   -0.4156   -0.6844
  -0.7041   -0.1338    0.6974
R =
  Columns 1 through 7
   5.8720   -3.4948   -0.7265   -3.7846    5.8271   -3.1647   -3.8775
    0.0000   -1.5707   -4.2229    1.3457    1.0943    2.9057    0.6590
    0.0000    0.0000   -0.0001    0.0000   -0.0001   -0.0000    0.0000
  Columns 8 through 10
   0.7728    5.7595   -3.1833
  -4.5323    1.4458    2.8754
   0.0000    0.0000   -0.0000

```

The columns of the matrix \mathbf{Q} are an orthonormal basis for \mathbf{R}^3 and $\mathbf{trdata} = \mathbf{QR}$. In particular, since the last row of \mathbf{R} is (nearly) zero, each of the columns of \mathbf{trdata} is (nearly) a linear combination of the first two columns of \mathbf{Q} . In other words, the first two columns of \mathbf{Q} form an orthonormal basis for the plane containing the data points. Moreover, since the j^{th} column of \mathbf{trdata} is $R_{1j}q_1 + R_{2j}q_2$ where q_1 and q_2 are the first two columns of \mathbf{Q} , the data points will lie on a circle in space if and only if the points of the plane whose coordinates are $\{(R_{1j}, R_{2j})\}$ lie on a circle.

Now points in the plane that lie on the circle with center (x_0, y_0) and radius r satisfy the equation $(x - x_0)^2 + (y - y_0)^2 = r^2$ or

$$ax + by + c = x^2 + y^2 \tag{1}$$

where $a = 2x_0$, $b = 2y_0$, and $c = r^2 - x_0^2 - y_0^2$. We find the best circle for the coordinate data by finding the least squares approximation for a , b , and c . Recall that in MATLAB, the `\` command gives the least squares solution to the system and that `v.^2` gives the vector of the same size as `v` whose entries are the squares of the entries of `v`, so the vector `coefs` below is the best set of coefficients (a, b, c) in Equation (1).

```
> coord=R(1:2,:)
coord =
  Columns 1 through 7
    5.8720    -3.4948    -0.7265    -3.7846    5.8271    -3.1647    -3.8775
         0    -1.5707    -4.2229     1.3457     1.0943     2.9057     0.6590
  Columns 8 through 10
    0.7728     5.7595    -3.1833
   -4.5323     1.4458     2.8754
> coefs=[coord' ones(10,1)]\'(coord(1,:).^2 +coord(2,:).^2)'
coefs =
    1.9969
    0.6968
   22.7537
```

This enables us to find the center and radius of the best circle in the plane and move this back to the original data.

```
> x0=coefs(1)/2
x0 =
    0.9985
> y0=coefs(2)/2
y0 =
    0.3484
> ctr=x0*Q(:,1)+y0*Q(:,2)+avgpt
ctr =
    2.5877
    4.2548
   -0.8088
> rad=sqrt(coefs(3)+x0^2+y0^2)
rad =
    4.8859
```

The center in the coordinate plane is (x_0, y_0) which corresponds to $x_0q_1 + y_0q_2$ in the translated plane and the center in the original plane is $x_0q_1 + y_0q_2 + \text{avgpt} = (2.5877, 4.2548, -0.8088)$. The radius, which is the same in each plane, is 4.8859. It is not difficult to check that the distance from each of the data points to the computed center is, to four places, 4.8859. This, together

with the QR-factorization showed the points lie (nearly) on a plane, and since the third column of Q is perpendicular to q_1 and q_2 , it must be a normal to the plane. Moreover, the point `avgpt` lies on the plane, so its equation is

$$-0.2125(x - 2.6551) - 0.6844(y - 4.9976) + 0.6974(z + 0.0592) = 0$$

Thus, these data points lie on a circle in this plane with radius 4.8859 and center (2.5877, 4.2548, -0.8088).

3 A Second Solution

The following is an outline of a solution to the problem in Figure 1 using a more sophisticated approach based on unitary coordinate transformations. The goal is to find the plane on which the data lies, then rotate the plane to be horizontal so that the x and y coordinates of the rotated data are coordinates in “the” x, y -coordinate system. The calculations are from a MATLAB session on a Macintosh.

Again, the data must be first be entered; it is stored as the 3×10 matrix `data` as before. We begin by fitting a plane to the data points by finding the best A , B , and C for the equation $Ax + By + Cz = 1$. Recalling that in MATLAB, `\` finds the least squares solution of a system, we compute

```
> N=data'\ones(10,1)
N =
    0.0528
    0.1700
   -0.1732
```

Checking, we get

```
> data'*N
ans =
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

Thus, the data points (nearly) lie on the plane $0.0528x + 0.1700y - 0.1732z = 1$ and N is a normal to this plane. Now, we want to rotate this plane in space, rigidly, so that it is horizontal, because, then, the usual x, y -coordinate system will apply. That is, we want to find a unitary U so that UN is vertical. Now, computing the QR-factorization of N will give a unitary V so that $V^*N = V^*VS = S$ is a multiple of $(1, 0, 0)$

```
> [V S]=qr(N)
V =
  -0.2125   -0.6844    0.6974
  -0.6844    0.6136    0.3937
   0.6974    0.3937    0.5989
S =
  -0.2484
     0
     0
```

and rearranging the rows of V^* gives a unitary that takes N to a vertical vector.

```
> U=[V(:,2:3) V(:,1)]'
U =
  -0.6844    0.6136    0.3937
   0.6974    0.3937    0.5989
  -0.2125   -0.6844    0.6974
> U*N
ans =
  -0.0000
   0.0000
  -0.2484
> U*U'
ans =
  1.0000    0.0000   -0.0000
  0.0000    1.0000   -0.0000
 -0.0000   -0.0000    1.0000
```

The final two calculations are, of course, unnecessary because we have constructed U to map N to a vertical vector and to be unitary, but I encourage my students to check their work constantly. When doing calculations in class (the class meets in an ordinary lecture room, but we have a portable computer and LCD plate for demonstrations), I use questions about checking as an opportunity to reinforce the theory by asking the students to tell me how we can check a calculation. For example, in this case, I might ask “How do we know that U is unitary?” and wait for the students to come up with the defining property that U^* is U^{-1} which we then check. Since U is unitary and maps N to a vertical vector, U is a rigid motion of space that should take the data points onto a horizontal plane.

```

> rotdata=U*data
rotdata =
  Columns 1 through 7
  -1.0268    4.0176    5.4046    1.4356   -2.0201   -0.2428    2.1054
  -1.6389    6.4082    2.8342    7.7948   -1.1776    7.8210    7.6172
  -4.0262   -4.0262   -4.0262   -4.0262   -4.0262   -4.0262   -4.0262
  Columns 8 through 10
    5.1151   -2.3188   -0.2077
    1.3310   -0.9803    7.8265
   -4.0262   -4.0261   -4.0262

```

The rotated data points (nearly) lie on the plane $z = -4.0262$ and the x, y -coordinates of these data points are

```

> Coord=rotdata(1:2,:)
Coord =
  Columns 1 through 7
  -1.0268    4.0176    5.4046    1.4356   -2.0201   -0.2428    2.1054
  -1.6389    6.4082    2.8342    7.7948   -1.1776    7.8210    7.6172
  Columns 8 through 10
    5.1151   -2.3188   -0.2077
    1.3310   -0.9803    7.8265

```

These points should lie on a circle in the x, y -plane; we fit a circle to these points as in the previous solution.

```

> Coefs=[Coord' ones(10,1)]\'(Coord(1,:).^2 +Coord(2,:).^2)\'
Coefs =
    1.0427
    5.9905
   14.6288
> X0=Coefs(1)/2
X0 =
    0.5214
> Y0=Coefs(2)/2
Y0 =
    2.9952
> Rad=sqrt(Coefs(3)+X0^2+Y0^2)
Rad =
    4.8859

```

Thus, $(0.5214, 2.9952, -4.0262)$ is the center of the fitted circle in the rotated plane; to find center in the original plane, we must rotate back by multiplying by $U^* = U^{-1}$.

```

> Ctr=U'*[X0 Y0 -4.0262]
Ctr =
    2.5877
    4.2548
   -0.8088

```

As before, we can check that the data points are indeed equidistant from `Ctr` and we find that the data points line on the circle with radius 4.8859 and center (2.5877, 4.2548, -0.8088) in the plane with equation $0.0528x + 0.1700y - 0.1732z = 1$.

4 Generating the Data

In order to individualize the assignment, I created the data using MATLAB programs. It is essential, unless you as the instructor wish to work 30 or 40 such problems, that the answer key be generated at the same time as the problems. The following MATLAB programs generate data and answers for each type of problem. In each case, the data points and the answers are stored in a single matrix with the top 10×3 submatrix being the data and the last few rows being the answer key. To produce the necessary output one only needs to turn the “diary” on and use a small loop to create the data. For example

```

diary ClassData
for j=1:10
    ['Data Set' int2str(j)]
    CirclePts
    ['   ']
end
diary

```

will create ten data sets labeled “Data Set 1”, “Data Set 2”, etc. and put them in the file “ClassData” on your disk (the [' '] puts some blank space after each data set). You can then print the contents of the diary twice, once for your files and once with the answers cut off of the bottom of the matrices for the students. A somewhat more elegant way of handling the data sets is to import the information from the diary file into a database then format the output more neatly (one way with answers for yourself and another way without answers for the students), add instructions for the assignment, add the students’ names, and so forth, before printing.

The sizes and positions of the figures are randomized within certain limits. In MATLAB, random numbers are generated by the commands `rand` and `randn`

where the first command produces uniformly distributed numbers from the interval $[0, 1]$ and the second produces random numbers that are normally distributed with mean 0 and standard deviation 1. Both depend on a ‘seed’ to create the numbers and at startup, at least on my machine, MATLAB begins with the same seed each time. Thus, if you create half of your data on Tuesday and half of your data on Wednesday, the data sets should be the same both days. The scripts first produce random points in the interval $[0, 2\pi]$, then get points on a circle or ellipse using cosine and sine, a random multiplier for the radius, and a random translation for the center. The matrix `rot` is a random unitary that moves the data points out of a horizontal plane into general position.

There is a small probability that the data created by these scripts will be difficult to work with because all the points might end up in a small sector; I have not done anything special to prevent that from happening but I have also not experienced difficulty because of such an occurrence.

Random Circle in Space

```
%
% CirclePts.m
% This script generates 10 points on a random circle in space;
% the points, the radius and center of the circle, and the normal to
% the plane of the circle are saved in the 13 x 3 matrix CircleData.
% The points are arranged by rows in the top 10 x 3 submatrix,
% the center is the 11th row, the normal to the plane of the circle
% is the 12th row, and [radius 0 0] is the last row.
% The radius is between 2 and 5 and each component of the center
% is a random variable with mean 1 and standard deviation 3.
%
theta=6.2832*rand(1,10);
rad=(3*rand(1)+2);
circ=rad*[cos(theta); sin(theta); zeros(1,10)];
[rot s]=qr(randn(3,3));
ctr=3*rand(3,1) + ones(3,1);
CircleData=(rot*[circ [0 0 0;0 0 0;0 1 0]]+[diag(ctr)*ones(3,11) ...
    [0 rad;0 0;0 0]]'
```

Random Ellipse in Space

```
%
% EllipsePts.m
% This script generates 10 points on a random ellipse in space;
% the points, the center of the ellipse, and the normal to the
% plane of the ellipse are saved in the 12 x 3 matrix EllipseData.
% The points are arranged by rows in the top 10 x 3 submatrix,
% the center is the 11th row, and the normal to the plane of the
% ellipse is the 12th row. The length of the major axis of the
% ellipse is between 4 and 10 and each component of the center
% is a random variable with mean 1 and standard deviation 3.
%
theta=6.2832*rand(1,10);
SMaxis=(3*rand(1)+2);
ellps=SMaxis*[cos(theta); .85*sin(theta); zeros(1,10)];
[rot s]=qr(randn(3,3));
ctr=3*rand(3,1) + ones(3,1);
EllipseData=(rot*[ellps [0 0;0 0;0 1]]+[diag(ctr)*ones(3,11) ...
    [0;0;0]])'
```

Random Bent Circle in Space

```
%
% BentPts.m
% This script generates 8 points on a random circle in space
% and 2 points that are nearly on, but not on, the circle;
% the points, the radius and center of the circle, and the normal to
% the plane of the circle are saved in the 13 x 3 matrix BentData.
% The points are arranged by rows in the top 10 x 3 submatrix,
% the center is the 11th row, the normal to the plane of the circle
% is the 12th row, and [radius 0 0] is the last row.
% The radius is between 2 and 5 and each component of the center
% is a random variable with mean 1 and standard deviation 3.
%
theta=5*rand(1,8);
rad=(3*rand(1)+2);
bent1=rad*[cos(theta(1,1:3)); sin(theta(1,1:3)); zeros(1,3)];
bent2=.85*rad*[cos(5.5) cos(6); sin(5.5) sin(6); 0 0];
bent3=rad*[cos(theta(1,4:8)); sin(theta(1,4:8)); zeros(1,5)];
bent=[bent1 bent2 bent3];
[rot s]=qr(randn(3,3));
ctr=3*rand(3,1) + ones(3,1);
BentData=(rot*[bent [0 0 0;0 0 0;0 1 0]]+[diag(ctr)*ones(3,11) ...
    [0 rad;0 0;0 0]])'
```

Bibliography

- [1] THE MATHWORKS, INC., MATLAB, Version 4.2c, Natick, Massachusetts, 1995.

Department of Mathematics
Purdue University
West Lafayette IN 47907-1395
email: cowen@math.purdue.edu
Web: <http://www.math.purdue.edu/~cowen>