Max Kramer

Table of Contents

I affirm that I have adhered to the honor code on this assignment.

Hello again, scientist! I'll write in italics, and problems for you will always be in ***bold****. As a general rule, I expect you to do at least as much writing as I do. Code should be part of your solution, but I expect variables to be clear and explanation to involve complete sentences. Cite your sources; if you work with someone in the class on a problem, that's an extremely important source.*

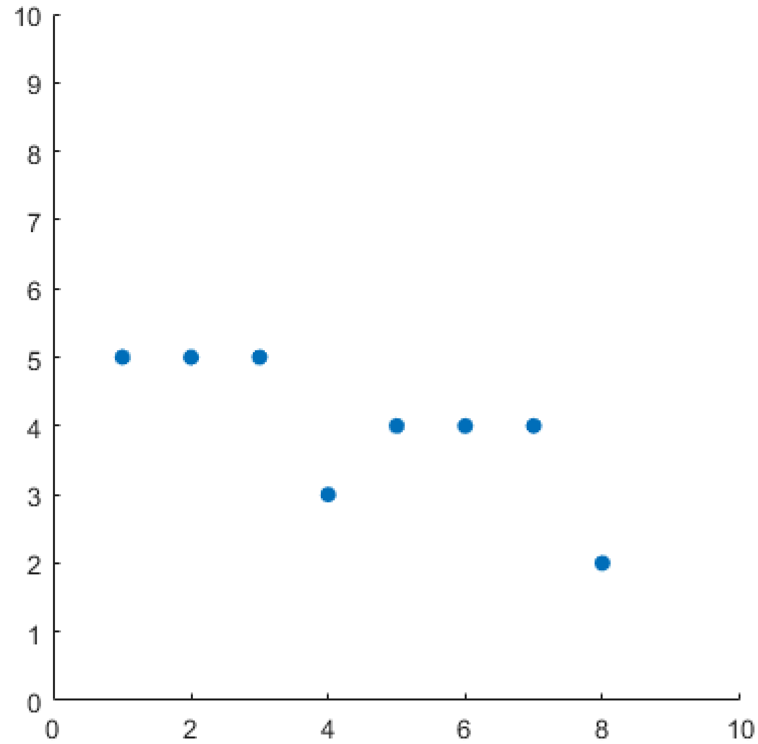
Problem 5.1.a.

It's 2020, right? We have super-powerful computers. Why don't we just fit curves exactly? That's a great rhetorical question, me! Let's find out.

```
B = [1 5; 2 5; 3 5; 4 3; 5 4; 6 4; 7 4; 8 2];
```

B is a list of 8 points in R^2. ***Show me a scatter plot of those 8 points. Explain why B stands for "Beethoven."***

```
scatter(B(:,1),B(:,2),'filled')
axis square; axis([0 10 0 10]);
```



DUN DUN DUN DUUUUN....DUN DUN DUN DUNNNN.

There is a unique degree 7 polynomial that interpolates these 8 points, and you're about to find it. ***Read this:*** *https://en.wikipedia.org/wiki/Polynomial_interpolation#Constructing_the_interpolation_polynomial*

You can create a Vandermonde matrix with vander(). ***Do that for the first column of B, and call the matrix A.***

```
A = vander(B(:,1))
```

```
A =

  Columns 1 through 6

         1           1           1           1           1
  1
       128          64          32          16           8
  4
      2187         729         243          81          27
  9
     16384        4096        1024         256          64
 16
     78125       15625        3125         625         125
 25
```

```
     279936        46656         7776         1296          216
    36
     823543       117649        16807         2401          343
    49
    2097152       262144        32768         4096          512
    64

  Columns 7 through 8

          1            1
          2            1
          3            1
          4            1
          5            1
          6            1
          7            1
          8            1
```

Once you've done that, uncomment the following. Then explain the output.

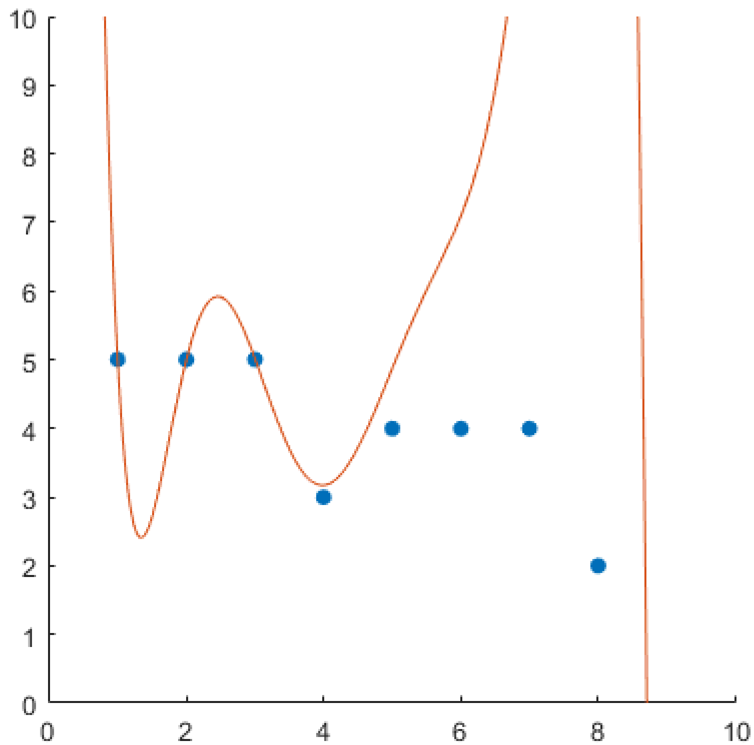
```
R = rref([A B(:,2)])
p = poly2sym(R(:,9))
```

```
R =

  Columns 1 through 7

    1.0000         0         0         0         0         0         0
         0    1.0000         0         0         0         0         0
         0         0    1.0000         0         0         0         0
         0         0         0    1.0000         0         0         0
         0         0         0         0    1.0000         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0    1.0000
         0         0         0         0         0         0         0

  Columns 8 through 9

         0   -0.0103
         0    0.3306
         0   -4.3306
         0   29.8056
         0 -115.0476
         0  245.3636
         0 -263.1119
    1.0000  112.0000


p =

- x^7/97 + (40*x^6)/121 - (524*x^5)/121 + (1073*x^4)/36 -
 (2416*x^3)/21 + (2699*x^2)/11 - (37625*x)/143 + 112
```

The first line of code creates the matrix R by creating an augmented matrix from the second comlumn of the matrix B and its Vandermode matrix A. The poly2sym() command then takes the 9th column of the resulting matrix and expresses it as a polynomial equation in the variable x.

You did it! Now ***plot B and p on the same axes; choose reasonable bounds.***

```
figure;
hold on;
scatter(B(:,1),B(:,2),'filled');
fplot(p,[0 10]);
axis square; axis([0 10 0 10]);
hold off;
```



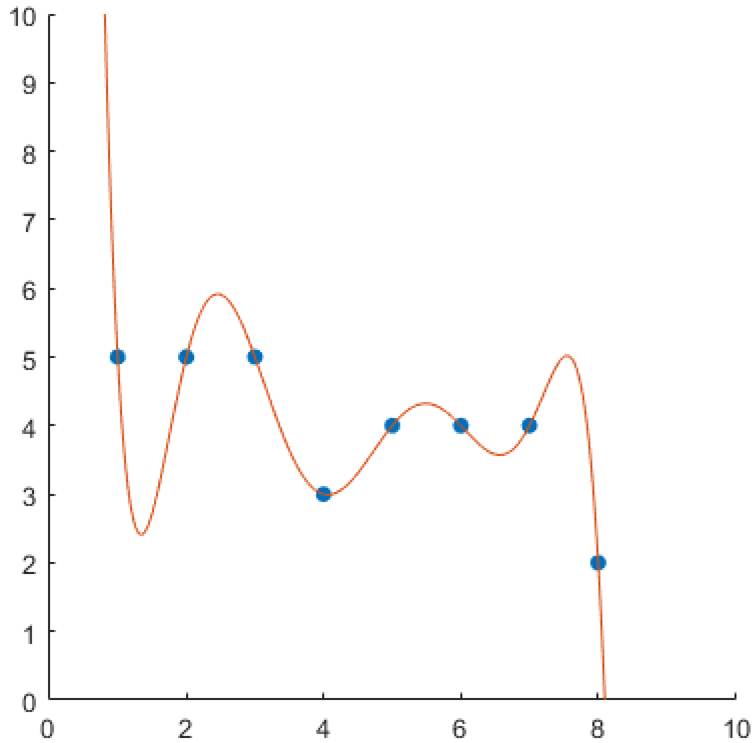
If you did this right, your polynomial should go pretty much exactly through the first three or four points and then start to freak out. That's intentional; as you've already seen, rref() is awful. Sorry.

Problem 5.1.b.

Okay, well, we're not going to give up that easily. MATLAB is much, much better at inverting matrices than it is at rref(), for reasons that we'll see later. This time, ***find the interpolating polynomial using matrix inversion and plot it and the points of B on the same set of axes.*** *If all goes well, your polynomial should hit every point.*

```
Q = A \ B(:,2);
```

```
t = poly2sym(Q);

figure;
hold on;
scatter(B(:,1),B(:,2),'filled');
fplot(t,[0 10]);
axis square; axis([0 10 0 10]);
hold off;
```

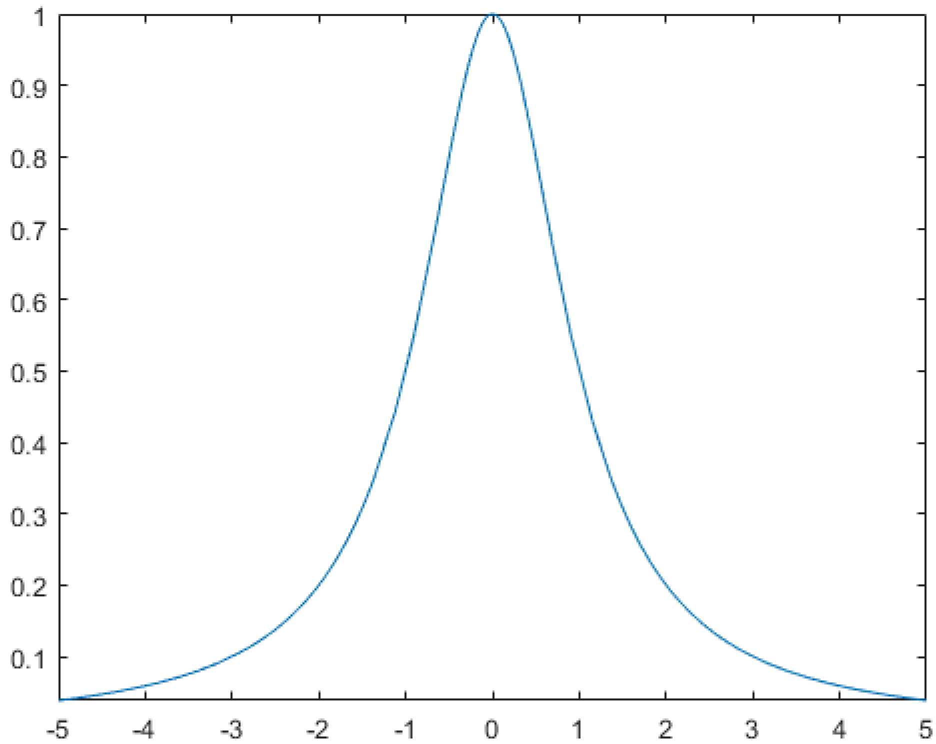


The vector Q is produced by solving the equation Ax=b by multiplying both sides by the matrix A^-1, represented here by A \ B(:,2). When that resultant vector is supplied to poly2sym(), a polynomial is generated that interpolates through all 8 points of B.

Problem 5.1.c.

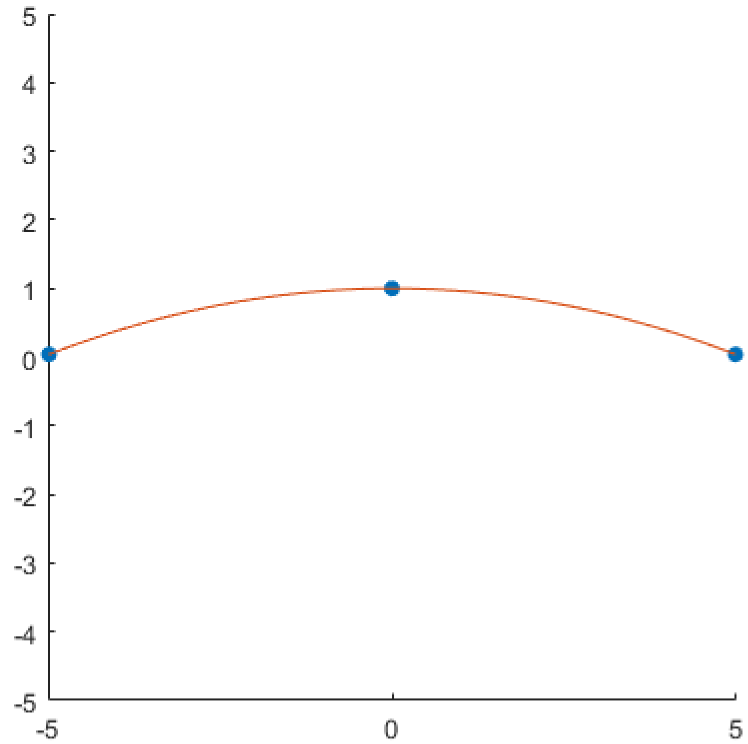
Now here's the real problem. Even when MATLAB is working perfectly, polynomial interpolation is extremely sensitive to "ringing," or what's called Runge's phenomenon. ("Runge" is pronounced ROON-geh.)

```
syms x;
runge = 1/(1+x^2);
figure;
fplot(runge,[-5,5]); % uncomment this when you're ready
```



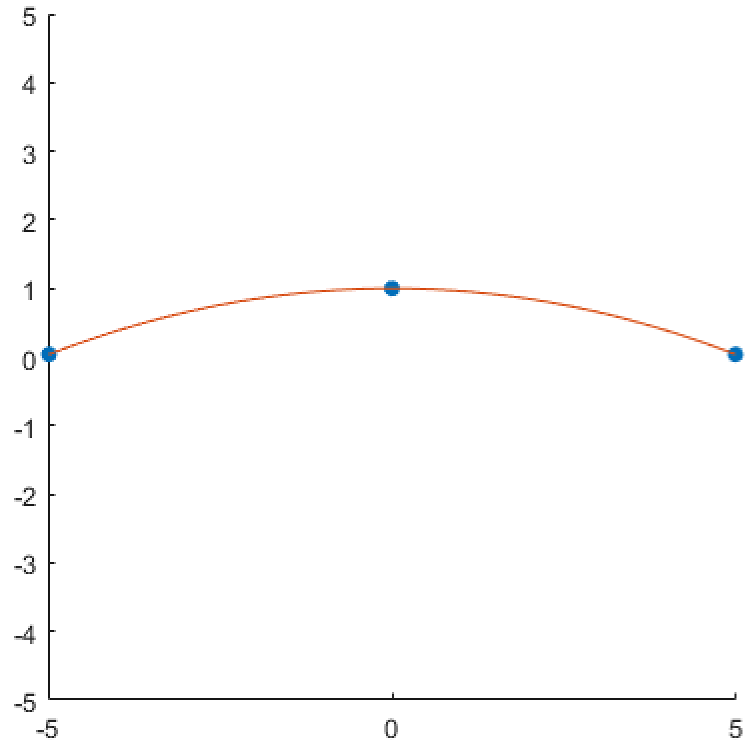
By using an appropriate Vandermonde matrix and matrix inversion, ***find and plot the interpolating polynomial for runge(x) on the interval [-5,5] using three equally-spaced points.***

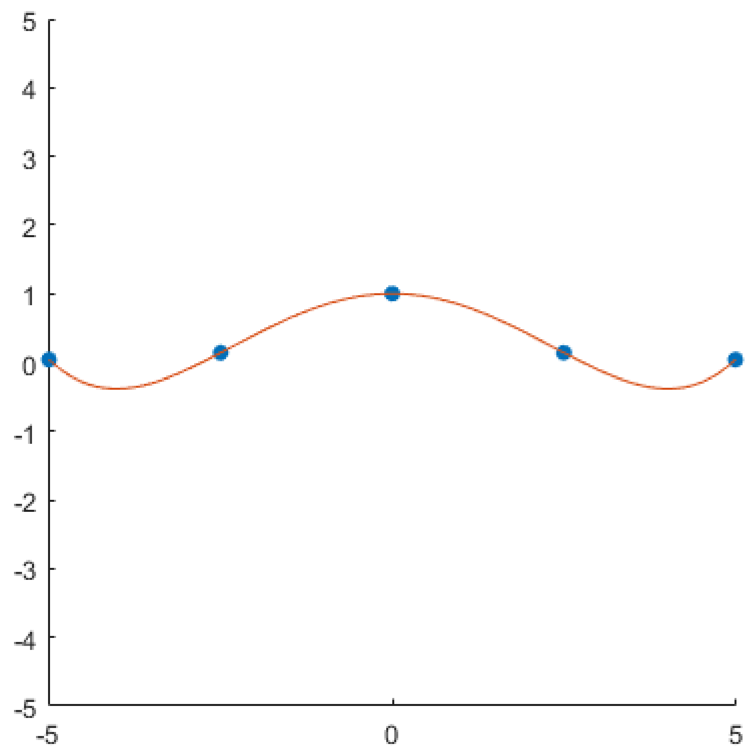
```
runge_vec_3 = [-5 1/26; 0 1; 5 1/26];
vand = vander(runge_vec_3(:,1));
V = rref([vand runge_vec_3(:,2)]);
threevand = poly2sym(V);
inv = vand \ runge_vec_3(:,2);
threeinv = poly2sym(inv);

figure;
hold on;
scatter(runge_vec_3(:,1),runge_vec_3(:,2),'filled');
fplot(threeinv,[-5 5]);
axis square; axis([-5 5 -5 5]);
hold off;
```



Five points.

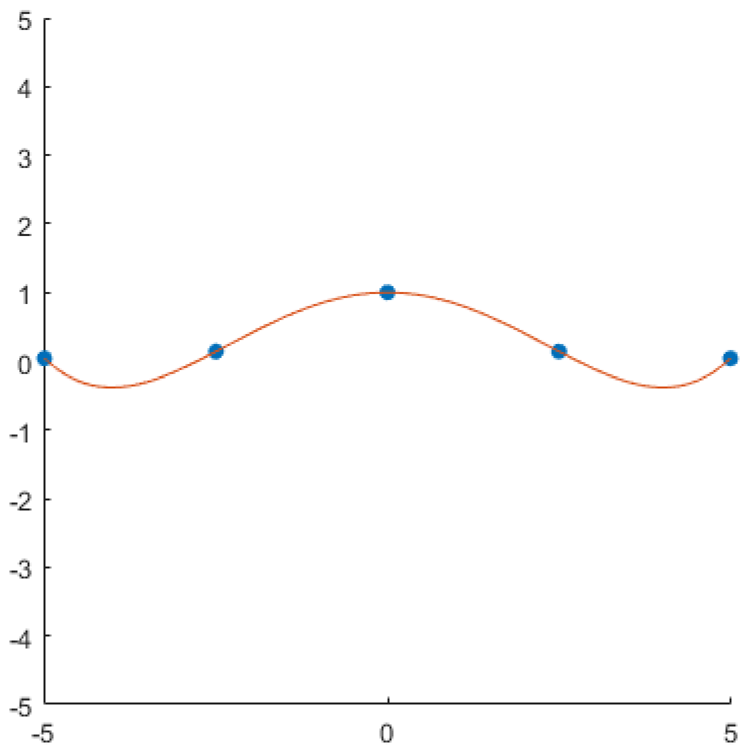
```
runge_vec_5 = [-5 1/26; -2.5 4/29; 0 1; 2.5 4/29; 5 1/26];
vand = vander(runge_vec_5(:,1));
V = rref([vand runge_vec_5(:,2)]);
fivevand = poly2sym(V);
inv = vand \ runge_vec_5(:,2);
fiveinv = poly2sym(inv);

figure;
hold on;
scatter(runge_vec_5(:,1),runge_vec_5(:,2),'filled');
fplot(fiveinv,[-5 5]);
axis square; axis([-5 5 -5 5]);
hold off;
```

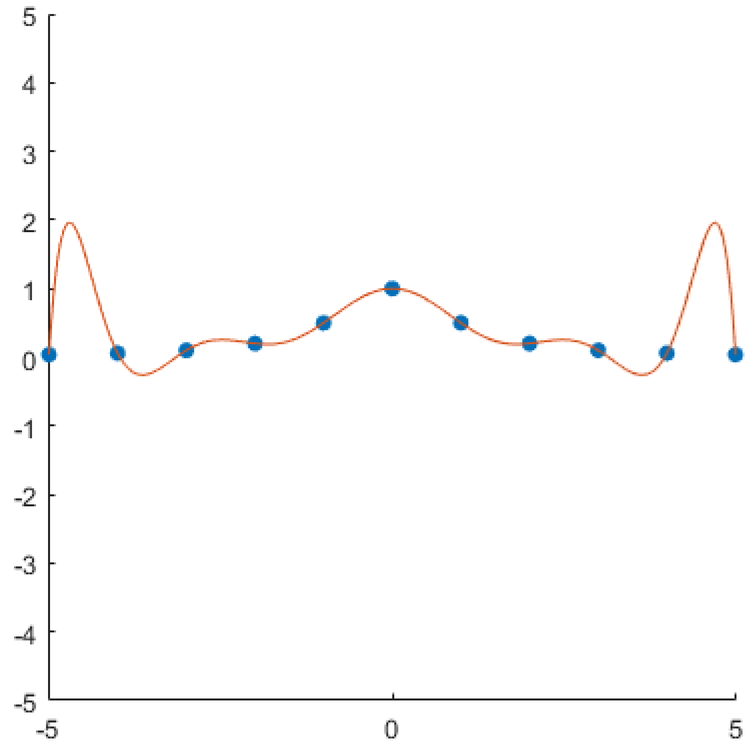

5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5


5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5

Eleven points.

```
xs = -5:5;
xs = xs';
ys = double(subs(runge,x,-5:5))';
runge_vec_11 = horzcat(xs,ys);
vand = vander(runge_vec_11(:,1));
V = rref([vand runge_vec_11(:,2)]);
elevenvand = poly2sym(V);
inv = vand \ runge_vec_11(:,2);
eleveninv = poly2sym(inv);

figure;
hold on;
scatter(runge_vec_11(:,1),runge_vec_11(:,2),'filled');
fplot(eleveninv,[-5 5]);
axis square; axis([-5 5 -5 5]);
hold off;
```

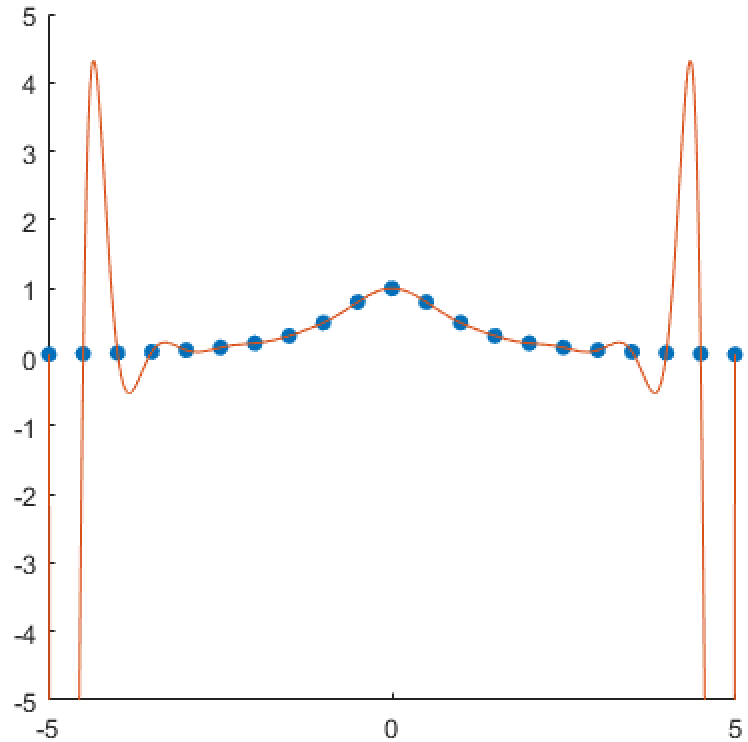




Twenty-one points.

```
xs_twentyone = -5:0.5:5;
xs_twentyone = xs_twentyone';
ys_twentyone = double(subs(runge,x,-5:0.5:5))';
runge_vec_21 = horzcat(xs_twentyone,ys_twentyone);
vand = vander(runge_vec_21(:,1));
V = rref([vand runge_vec_21(:,2)]);
twentyonevand = poly2sym(V);
inv = vand \ runge_vec_21(:,2);
twentyoneinv = poly2sym(inv);

figure;
hold on;
scatter(runge_vec_21(:,1),runge_vec_21(:,2),'filled');
fplot(twentyoneinv,[-5 5]);
axis square; axis([-5 5 -5 5]);
hold off;
```

```
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  1.452073e-16.
```



The discovery of this example (1901), along with the similar Gibbs phenomenon for Fourier series (1899), was a shock to the scientific community at the time. (The phenomenon was actually discovered earlier by physicists, but they assumed that it was just measurement error.) Designing high-quality interpolations that minimize ringing remains a major challenge in the field of signal processing.

Published with MATLAB® R2019b