SVD Decomposition of a Surface

Rank 1 Approximation

The code for this example is here.

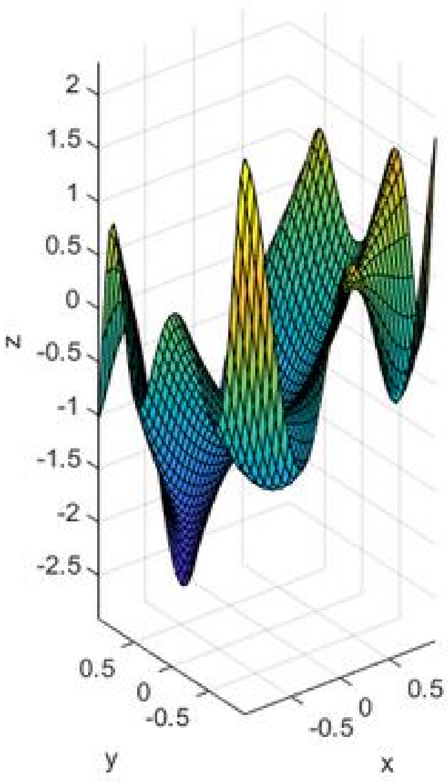
Here we want to consider a surface lying over a square patch, say the square in the x, y-plane given by $[-1, 1]\times[-1, 1]$. The z-coordinates produced by a function $z = f(x, y)$. We can almost view this as an infinite matrix, for example, at row $y = .25$, and column $x = -.15$, there is the entry $z = f(-.15, .25)$. If we in fact discretize $[-1, 1]$ by splitting it into 40 sub-intervals all of width $2/40 = -.05$ we get a 40×40 matrix of z values. Here $y_i = -1 + 0.05(i - 1)$, $x_j = -1 + 0.05(j - 1)$ and $z(i, j) = f(x_j, y_i)$ for $1 \leq i, j \leq 41$. (You actually need 41 points to make 40 intervals.)

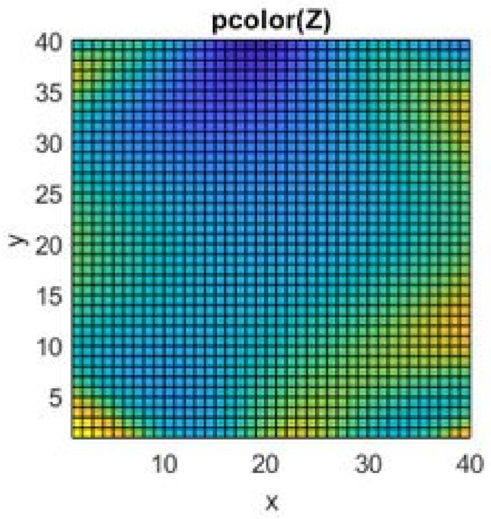
This is all very easy to achieve and plot in MATLAB. Here is code taken from the provided MATLAB live script SVDProject.mlx:

```
close
% This generates: -0.975 -0.925 .... 0.925 0.975, 40 points equally
    spaces
% in [-1,1] separated by a distance of 2/40 = 0.05
x = linspace(-0.975,0.975,40);
y = x;
% This makes the x,y - plane as far as MATLAB is concerned.
% X is a 40 x 40 matrix and (X(i,j), Y(i,j)) is what you
% would use as (x_i, y_j)
[X,Y] = meshgrid(x,y);
% Define the surface
Z = sin(2*pi*X.*Y.^2 - Y) - cos(4*pi*X.^2*Y + X) + X.^2 - Y.^3;
% Actually plot the surface

figure(1);
subplot(1,2,1);
surf(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('z');
axis([-1 1 -1 1 -3 2.5], 'equal');

subplot(1,2,2);
pcolor(Z);
```

```
title('pcolor(Z)');
xlabel('x');
ylabel('y');
axis('square');
```

The result is:





The second plot is simply a representation of Z using colors instead of numbers, but notice that the matrix is flipped vertically, so that the $(1, 1)$ value is in the lower left, not upper left. Here is part of the matrix Z. This helps us appreciate the visual representation.

$$Z = \begin{bmatrix} 2.3070 & 2.1675 & 1.9471 & 1.6540 & \cdots \\ 2.1036 & 1.8324 & 1.5112 & 1.1539 & \cdots \\ 1.6833 & 1.3435 & 0.9860 & 0.6241 & \cdots \\ 1.1526 & 0.8045 & 0.4634 & 0.1390 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

It is important to understand how the indices in the Z-matrix correspond to the x, y-plane so here is $X(1:4, 1:4)$ and $Y(1:4, 1:4)$:

$$X = \begin{bmatrix} -0.9750 & -0.9250 & -0.8750 & -0.8250 & \cdots \\ -0.9750 & -0.9250 & -0.8750 & -0.8250 & \cdots \\ -0.9750 & -0.9250 & -0.8750 & -0.8250 & \cdots \\ -0.9750 & -0.9250 & -0.8750 & -0.8250 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

and

$$Y = \begin{bmatrix} -0.9750 - 0.9750 - 0.9750 - 0.9750 & \cdots & & & & \\ -0.9250 - 0.9250 - 0.9250 - 0.9250 & \cdots & & & & \\ -0.8750 - 0.8750 - 0.8750 - 0.8750 & \cdots & & & & \\ -0.8250 - 0.8250 - 0.8250 - 0.8250 & \cdots & & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

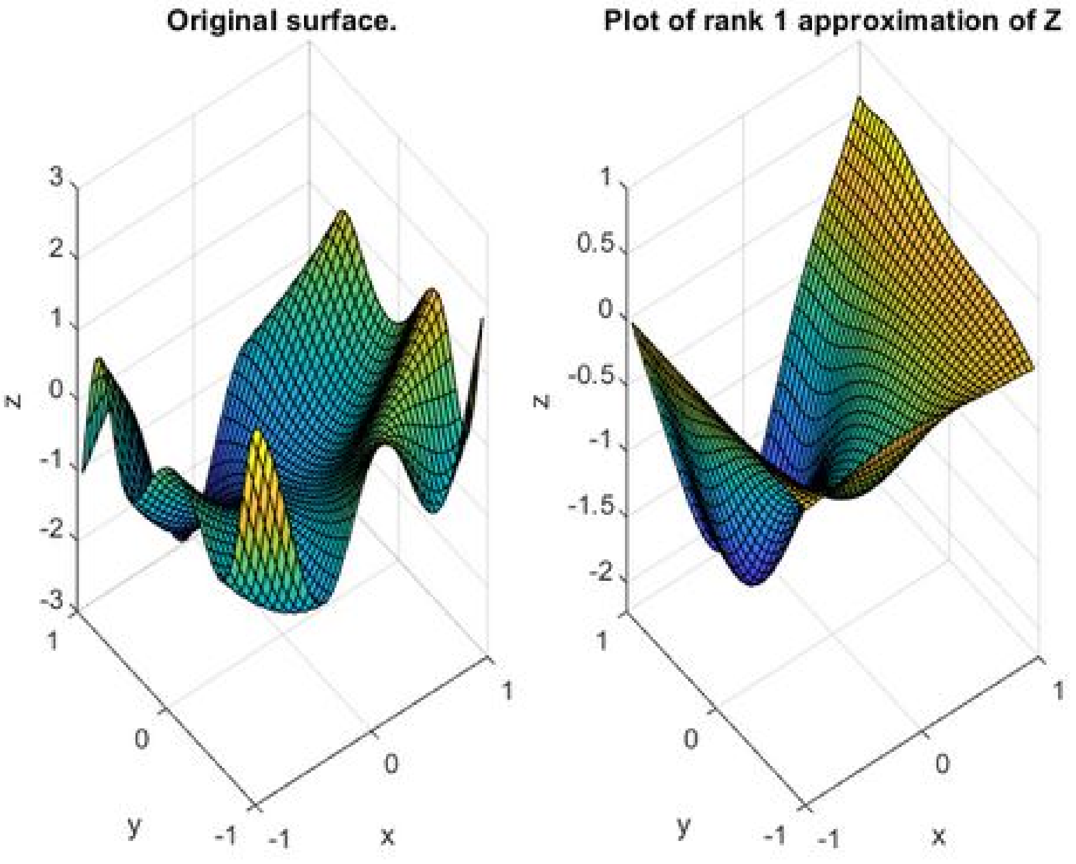
So $X(i,j) = x_j$ and $Y(i,j) = y_i$ so $z(x_j, y_i) = z(X(i,j), Y(i,j)) = Z(i,j)$.

Let $Z = U\Sigma V^T$ be the SVD decomposition of Z so that $Z = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$, where $k = \text{rank}(Z)$.

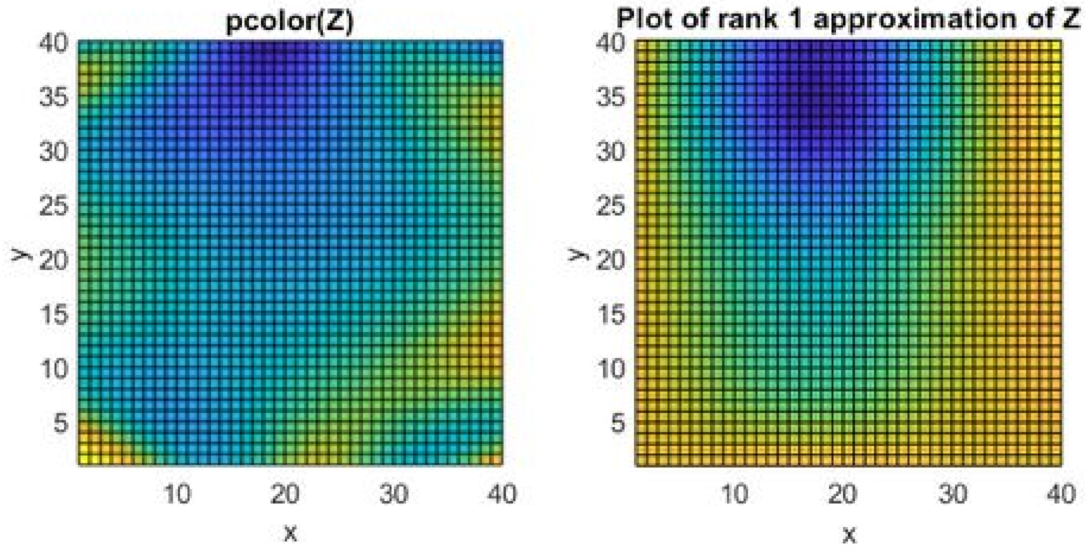
Let $Z(r)$ be the "best" rank r approximation of Z, this is given by

$$Z(r) = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

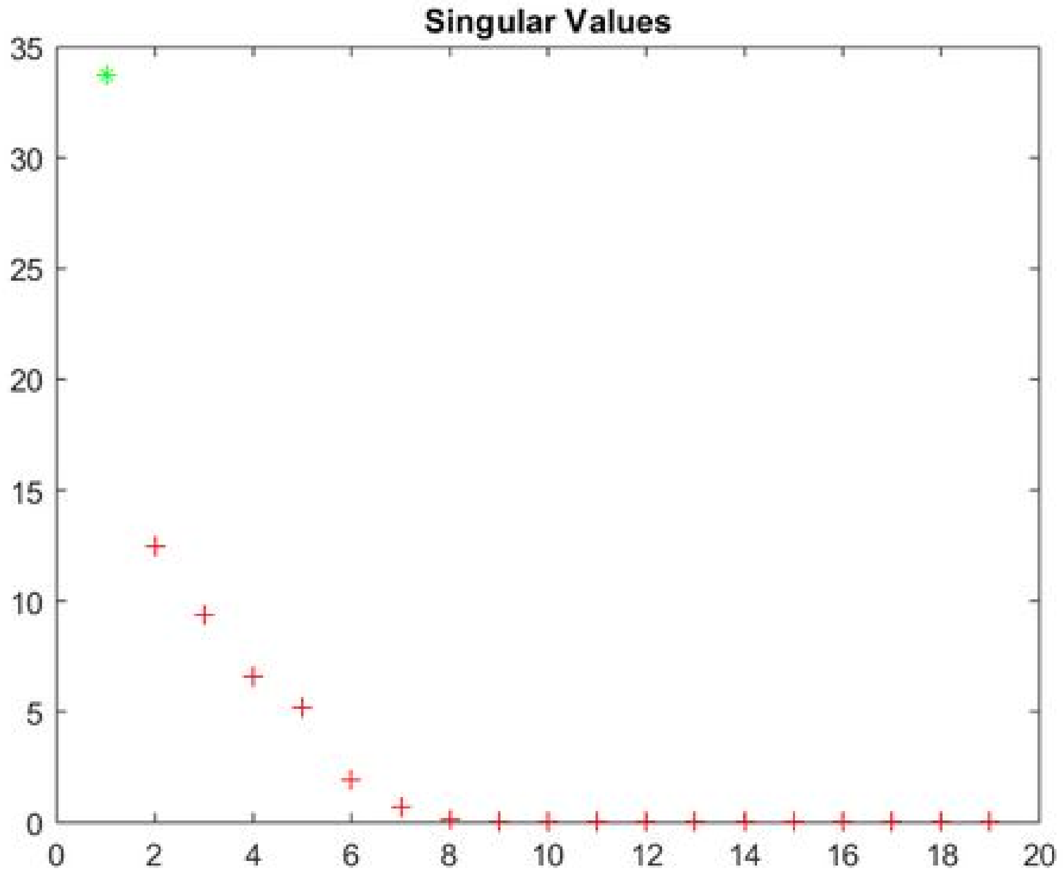
Think about the interpretation of $\boldsymbol{u}_1 \boldsymbol{v}_1^T$ here. This is a rank 1 approximation of the matrix Z and as a matrix can be viewed itself as a surface. The vector $\boldsymbol{u}_1$ generates the surface as follows: At the i^{th} x-value, x_i we generate the i^{th} slice parallel to the y-axis given by $\sigma_1 v_1(i) \boldsymbol{u}_1$. Stacking all these slices produces the surface. Similarly $\boldsymbol{v}_1$ generates the surface as slices in the x-direction given by v_1^T with weight $\sigma_1 u_1(i)$.



Original surface compared to the rank 1 approximation $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T$

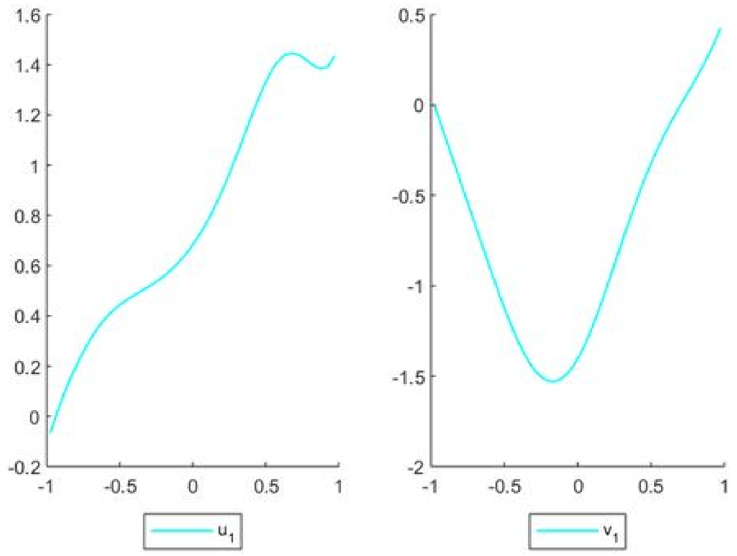


Coloring of the two matrices Z and $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T$ (the "best" rank 1 approximation to Z).



Graph of $\boldsymbol{u}_1$ (the "slices") and $\boldsymbol{v}_1$ (the "weights").

The following plots the singular values, you can see the relative weights contributed by each rank 1 approximation $\boldsymbol{u}_i\boldsymbol{v}_i$, there is a singular value for each i, $1 \leq i \leq \text{rank}(Z)$.



We can measure the percentage of the "information" in Z captured by our rank r approximation $Z(r)$ as

$$\frac{\sum_{i=1}^{r}\sigma_i^2}{\sum_{i=1}^{k}\sigma_i^2}$$

For this rank 1 approximation we have

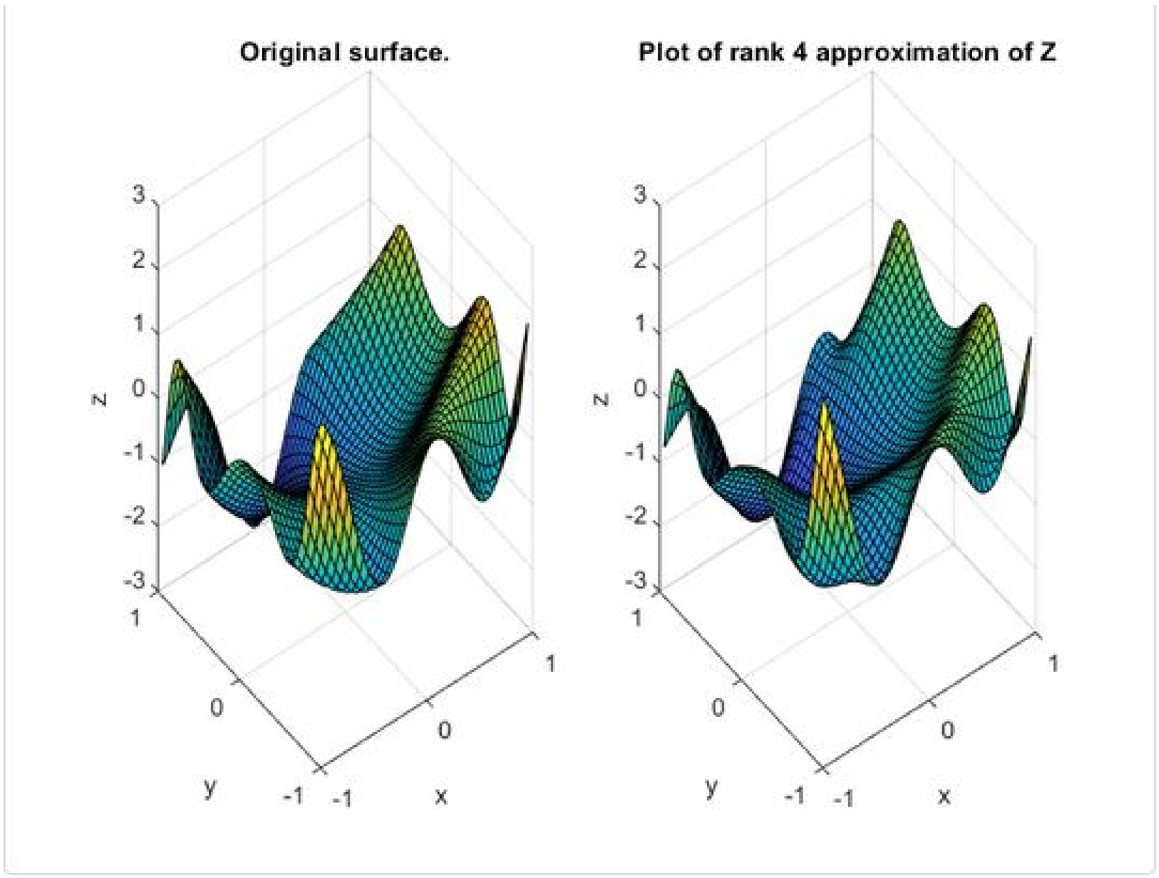
$$\frac{\sigma_1^2}{\sum_{i=1}^{k}\sigma_i^2} = 78.07\%$$

so about 78.07% of the "information" in the original surface is captured by this rank 1 approximation.

Rank N approximation.

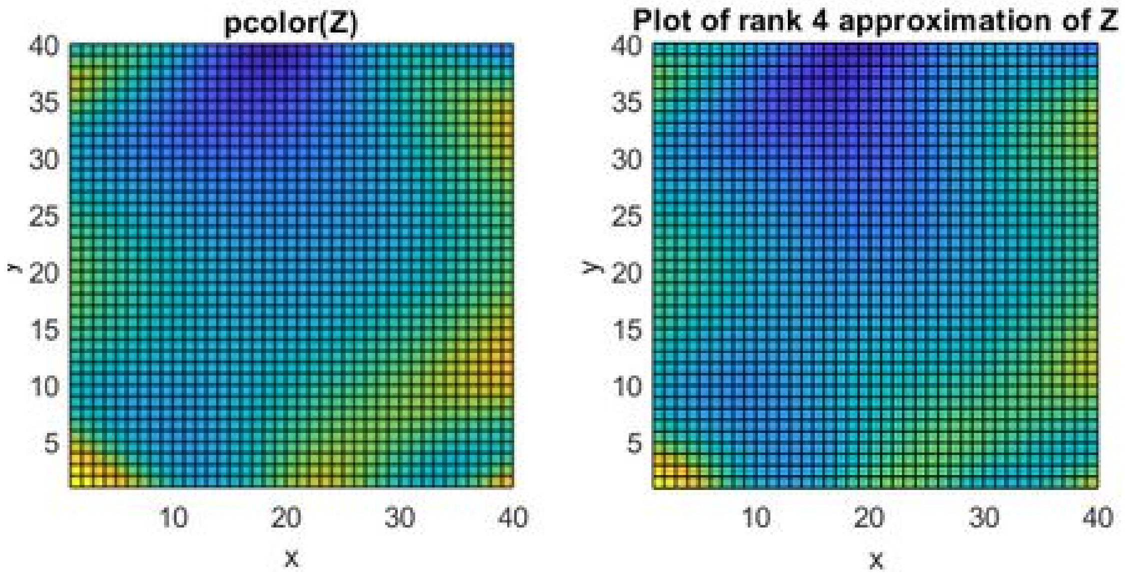
Here we take $N = 4$ to be specific. The rank 4 approximation $Z(4) = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^T + \sigma_3 \boldsymbol{u}_3 \boldsymbol{v}_3^T + \sigma_4 \boldsymbol{u}_4 \boldsymbol{v}_4^T$ captures about 97.736% of the information! Here are plots of the rank 4 approximation of Z vs Z itself.

```
% Choose what rank approximation to investigate

N = 4;

% This tiny bit of code does all the work
 [U,S,V] = svd(Z);
% Here is the rank N approximation
ZZ = U(:,1:N)*S(1:N,1:N)*V(:,1:N)';
```



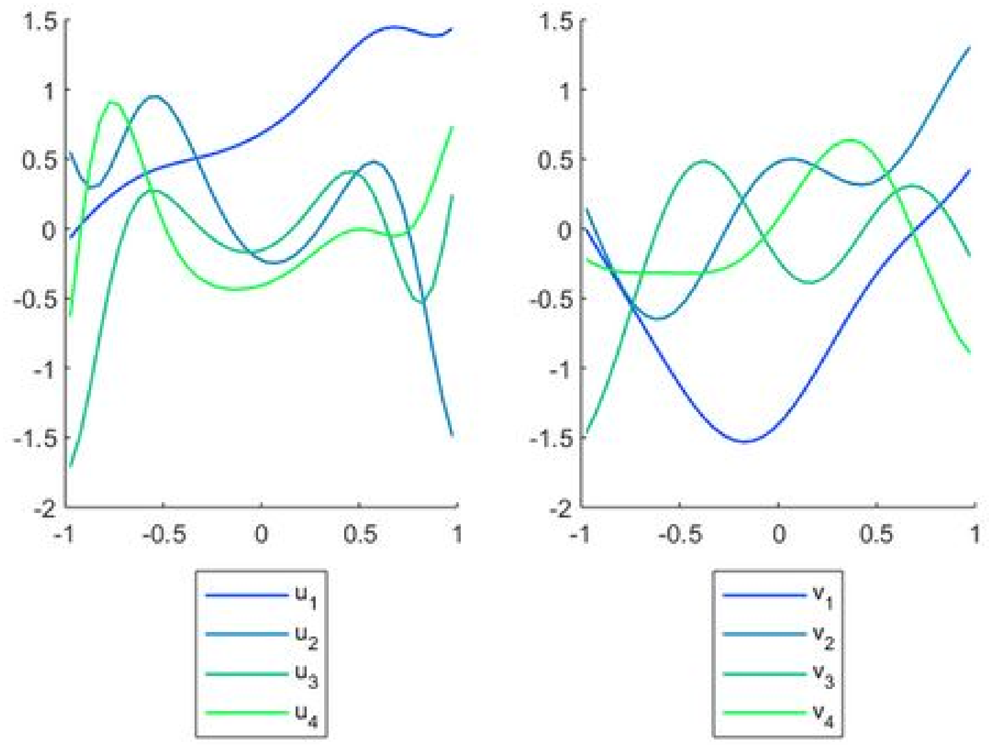
Original surface compared to $Z(4)$ (the rank 4 approximation)

```
close
% Build a title that depends on N
str = ['Plot of rank ', num2str(N), ' approximation of Z'];

% This plots the actual surfaces Z and Z(N)
f3 = figure(3);
subplot(1,2,1);
surf(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('z');
title("Original surface.")


subplot(1,2,2);
surf(X,Y,ZZ);
xlabel('x');
ylabel('y');
zlabel('z');
title(str);
```



Here are the coloring of the two matrices Z and $Z(4)$ (the "best" rank 4 approximation to Z).

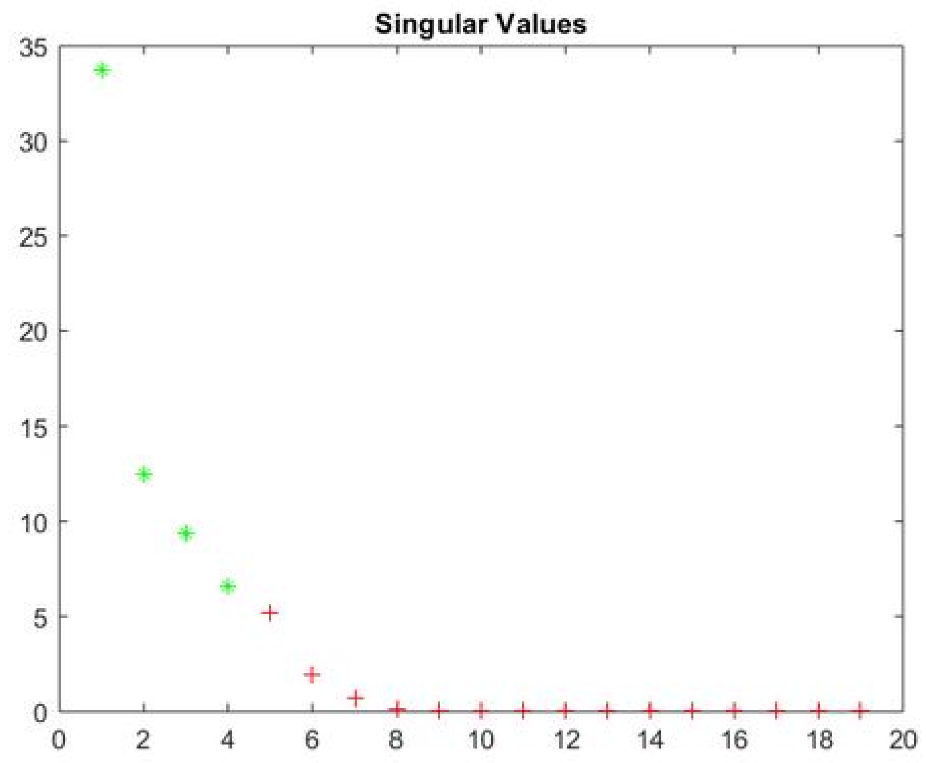
```matlab


% This shows the matrices with values color-coded.
f4 = figure(4);

subplot(1,2,1);
pcolor(Z);
title('pcolor(Z)');
xlabel('x');
ylabel('y');
axis('square');

subplot(1,2,2);
pcolor(ZZ);
title(str);
xlabel('x');
ylabel('y');
axis('square');
```



Plot of $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{u}_4$ (the "slices") and $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$ (the "weights").

```
close
f6 = figure(6);

% Have to do some non-sense to get the right coloring/labels
lgu = [];
lgv = [];
for i = 1:N

    su = ['"', 'u_', num2str(i), '"'];
    sv = ['"', 'v_', num2str(i),'"'];
    [lgu_1,lgu_2] = size(lgu);
    if (lgu_2) == 0
        lgu = [su];
        lgv = [sv];
    else
        lgu = [lgu, ',', su];
        lgv = [lgv, ',', sv];
    end

    subplot(1,2,1);
    hold on;
    plot(Y(:,1), sqrt(B(i))*U(:,i), 'color', [0, i/N, 1 - (i-1)/N],
        'linewidth', 1.0);

    subplot(1,2,2);
    hold on;
    plot(X(1,:), sqrt(B(i))*V(:,i)', 'color', [0,  i/N, 1 - (i-1)/N
        ],'linewidth',1.0);
end

subplot(1,2,1);
eval(['legend(', lgu, ', "location", "southoutside")']);
legend('boxon');
hold off;

subplot(1,2,2);
eval(['legend(', lgv, ', "location", "southoutside")']);
legend('boxon');
hold off;
```

The following is again the plot of the singular values showing the relative weights of the 4 singular values used.



```
% This plots the singular values
figure(5)

plot(diag(S)(1:rank(S)),'r+', diag(S)(1:N),'g*');
```

```
title("Singular Values")
```

$$\frac{\sigma_1^2+\sigma_2^2+\sigma_3^2+\sigma_4^2}{\sum_{i=1}^k \sigma_i^2} = 0.9785 = 97.85\%$$

So about 97.85% of the original information from the surface is contained in the rank 4 approximation.

There are other measures of how closely the approximation fits the original matrix, you can use the sum of the squares differences between the individual points in the matrix, this is just the L^2-norm we use on the vector space $\mathbb{R}^{41\times 41}$. The formula for this is

$$\sum_{\substack{1\le i\le 41\\ 1\le j\le 41}} (Z(i,j) - Z(4)(i,j))^2 = 31.31$$

This is closely related to the integral of the square of the distance between the two surfaces and since each little square is 0.05×0.05 in dimension, $dx\,dy = dA = 0.025$ and we get the approximate integral $\int_{[-1,1]\times[-1,1]}(Z(4) - Z)^2\,dx\,dy$ as

$$\sum_{\substack{1\le i\le 41\\ 1\le j\le 41}} (Z(i,j) - Z(4)(i,j))^2\,dA = 0.07828$$

Here is code for these calculations

```
variance_accounted_for = sum(diag(S(1:N,1:N)).^2)/sum(diag(S(1:rank
    (Z),1:rank(Z))).^2);

error = sum(sum((ZZ - Z).^2));

dA = (X(1,2)-X(1,1))*(Y(2,1)-Y(1,1));

disp(['variance accounted for = ', num2str(round(
    variance_accounted_for*100,2)),'%'])
fprintf("L^2 error = %.2f", error)
disp(['Squared area between surfaces = ', num2str(error*dA)])
```

Project

A surface can simply be interpreted as a matrix of numbers, similarly a gray image can be interpreted as a matrix of "gray values." The file "SVDImRed.mlx" has code that reads in an image of a cat converts this to a gray image which is just a matrix like Z above, in this case the matrix is called "GrayImg." An SVD decomposition is created GrayImg $= USV^T$. You can view the image reconstructed exactly as described above for the reconstruction of the surface by using the command "show(M,N)" which will reconstruct the image using the M^{th} through

N^{th} singular values/singular vectors, i.e.,

$$\text{GrayImg}(M, N) = \sum_{i=M}^{N} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

You can see the original grayscale and the reconstructed image side by side.

- Look at the first 10 rank 1 reconstructions, that is run "show(i,i)" for $i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$, then look at the rank 10 reconstruction $show(1, 10)$. Comment on what you notice.
- How much data is in the original image? Each gray value is a 1 byte = 8 bit positive integer value between 0 and 255. So each pixel requires 1 byte of information.
- How much data must be stored for the rank 10 reconstruction of the image? What is the percent reduction in data storage?
- What percentage of the variance is accounted for in the rank 10 reconstruction?
- What rank reconstruction is required to capture 99% of the initial variance (information)? What is the percent reduction in storage for this rank?

Recall that the the variance explained is

$$\text{variance explained} = \frac{\sum_{i=M}^{N} \sigma_i^2}{\sum_{i=1}^{k} \sigma_i^2},$$

where k is the rank of the matrix corresponding to the original image.