- SVD computation
- Image compression and analysis
- Model reduction

- From $A = U\Sigma V^T$ deduce $AA^T = U\Sigma^2 U^T$, $A^TA = V\Sigma^2 V^T$, hence U is the eigenvector matrix of AA^T, and V is the eigenvector matrix of A^TA
- SVD computation is carried out by solving eigenvalue problems

```
octave> A=[2 -1; -3 1]; [U S2]=eig(A*A'); [V S2]=eig(A'*A); S=sqrt(S2); disp([ U S V']);
```

```
-0.81742  -0.57605   0.25878   0.00000  -0.36060  -0.93272
-0.57605   0.81742   0.00000   3.86433  -0.93272   0.36060
```

- The Octave/Matlab `svd` function carries out both eigenvalue problems efficiently

```
octave> [U S V]=svd(A); disp([U S V']);
```

```
-0.57605   0.81742   3.86433   0.00000  -0.93272   0.36060
 0.81742   0.57605   0.00000   0.25878  -0.36060  -0.93272
```

```
octave> disp([U*U' U'*U V*V' V'*V]);
```

```
1  0  1  0  1  0  1  0
0  1  0  1  0  1  0  1
```

```
octave>
```

- The SVD can be expressed as a weighted sum of "rank-1 updates" $\boldsymbol{u}_i \boldsymbol{v}_i^T$ with weights σ_i

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

- By construction of the SVD $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_r > 0$, and very often $\sigma_1 \gg \sigma_2 \gg \ldots \gg \sigma_r > 0$,

$$\boldsymbol{A} \cong \sum_{i=1}^{p} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i, p \ll r$$

```
octave> function img=readface(n,type)
          fhead = "/home/student/courses/MATH547/lessons/yalefaces/subject";
          fnr = strcat(num2str(n,"%2.2d"),"."); fname = strcat(fhead,fnr,type);
          [img,map,alpha] = imread(fname);
        endfunction;
```

```
octave> f01n=readface(1,"normal");
```

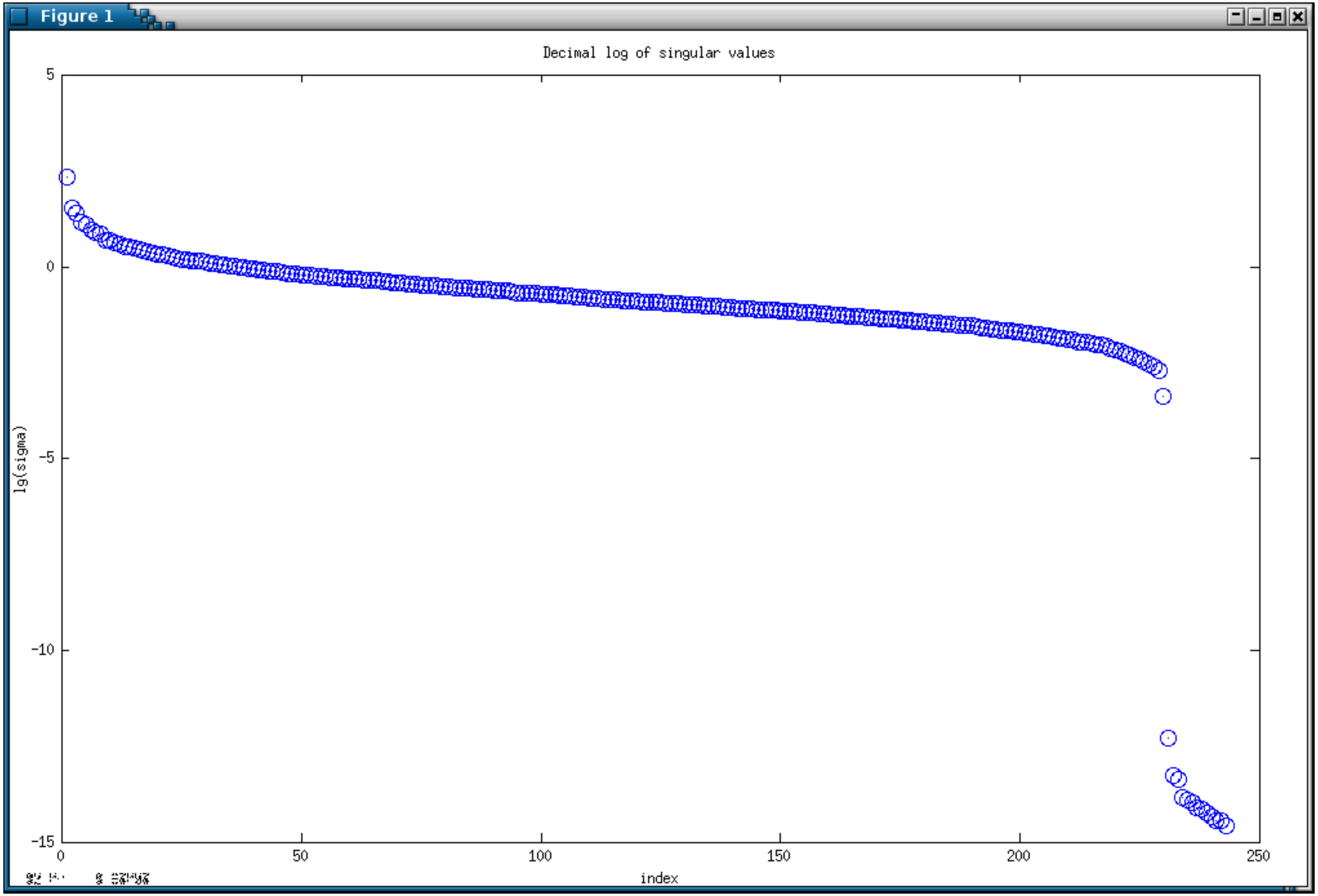
```
octave> function mat=img2mat(img)
          [h,w]=size(img); mat=zeros(h,w); mat=double(img)/255.;
        endfunction;
```

```
octave> f01n=readface(1,"normal"); m01n=img2mat(f01n);
```

```
octave>
```

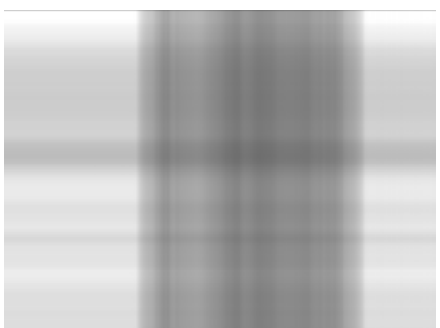
Image compression

```
octave> [U S V]=svd(m01n); plot(log10(diag(S)),'o');
octave> xlabel('index'); ylabel('lg(sigma)'); title('Decimal log of singular values');
octave> print -deps /tmp/temp.eps;
```

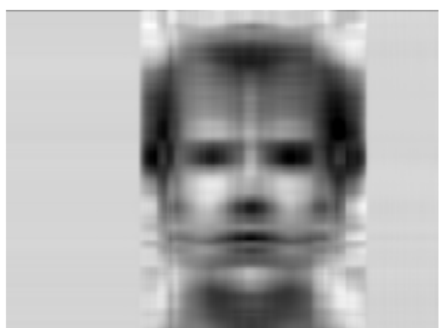


Build sequence of image approximations

```
octave> function cimg=cmprim(img,p)
          [h,w]=size(img); mat=zeros(h,w); mat=double(img)/255.;
          [U S V]=svd(img); s=diag(S); cmat=zeros(h,w);
          for i=1:p
            cmat = cmat + s(i)*U(:,i)*V(:,i)';
          end;
          cmn=min(min(cmat)); cmat=cmat+cmn*ones(h,w);
          cmx=max(max(cmat)); sc=255./cmx; cmat=sc*cmat; cimg=uint8(cmat);
          fname=strcat("/home/student/courses/MATH547/lessons/cimg",num2str(p),".png");
          imwrite(cimg,fname);
        endfunction;
octave> f01n1=cmprim(f01n,1); f01n4=cmprim(f01n,4); f01n16=cmprim(f01n,16);
octave> f01n64=cmprim(f01n,64);
octave>
```



$p = 1$



$p = 4$

$p = 16$

$p = 64$

- Mass + spring systems lead to ODE system, $\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = \boldsymbol{f}$, with $\boldsymbol{M} = \mathrm{diag}(m_1, \ldots, m_n)$, $\boldsymbol{K} \in \mathbb{R}^{n \times n}$, $\boldsymbol{x}(t) : \mathbb{R} \to \mathbb{R}^n$ the trajectories of the point masses
- The point masses usually move in some correlated fashion, e.g., eigenmodes
- Construct a reduced model $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{y}$, $\boldsymbol{B} \in \mathbb{R}^{m \times p}$, $p \ll n$, $\boldsymbol{B}$ orthonormal
- Take projection of system on $C(\boldsymbol{B})$ (multiply by projection matrix $\boldsymbol{P} = \boldsymbol{B}\boldsymbol{B}^T$)

$$(\boldsymbol{B}^T\boldsymbol{M}\boldsymbol{B})\,\ddot{\boldsymbol{y}} + (\boldsymbol{B}^T\boldsymbol{K}\boldsymbol{B})\boldsymbol{y} = \boldsymbol{B}^T\boldsymbol{f} \Leftrightarrow \boldsymbol{N}\,\ddot{\boldsymbol{y}} + \boldsymbol{L}\boldsymbol{y} = \boldsymbol{g}, \boldsymbol{N}, \boldsymbol{L} \in \mathbb{R}^{p \times p}, \boldsymbol{y}, \boldsymbol{g} \in \mathbb{R}^p$$