Problem (b)

```
function p = variancePercentage(s)

v = s.^2;
p = cumsum(100*v/sum(v));
```

The first two calls (imagePCA and variancePercentage) took 6.681431 and 0.006491 seconds, respectively. The compression achieved is $\rho = 19.6$, preserving 79.421582% of the total data variance.

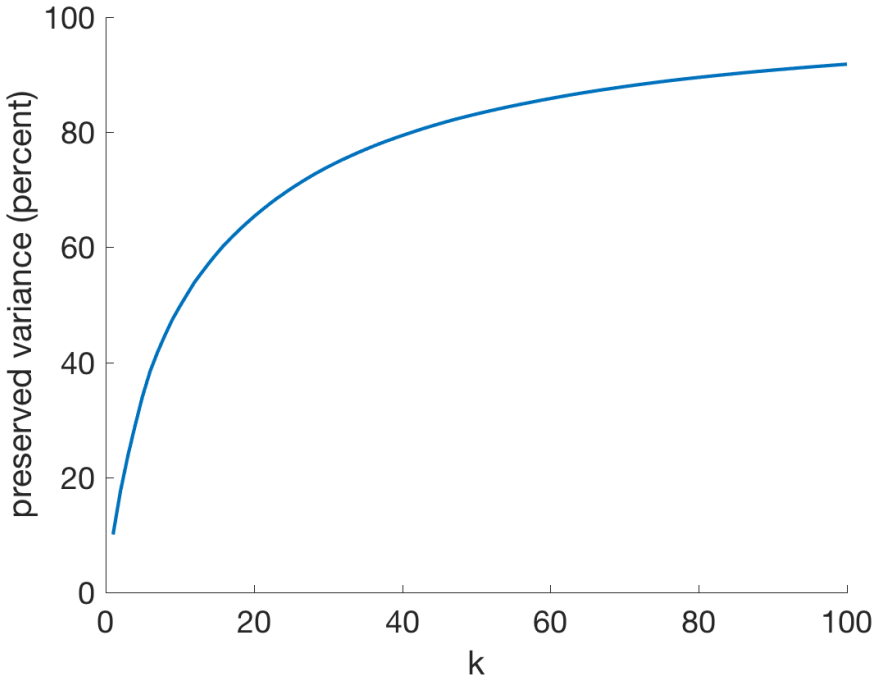

Figure 2: Preserved variance for kth singular values

Problem (e)

```
function [code, p] = digitPCA(data, k)

digit = 0:9;
N = length(digit);
code = cell(1, N);

[r,c,n] = size(data.image);
m = r*c;
p = zeros(min(m, n), N);

% Compute PCA for each digit
for d=1:N
    I = data.image(:, :, data.label == digit(d));
    [code{d}, s] = imagePCA(I, k);
    p(:, d) = variancePercentage(s);
end
```

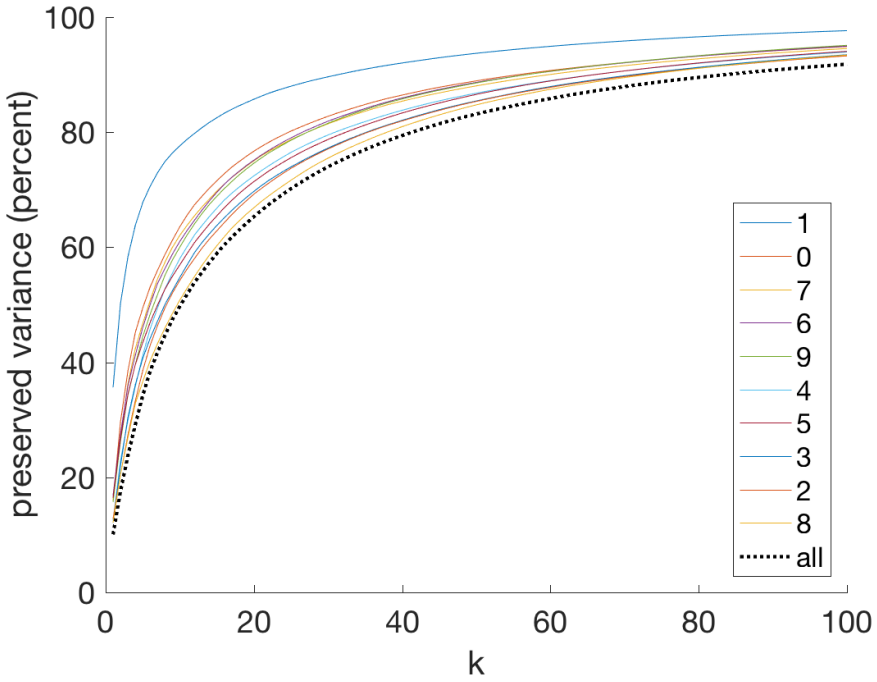

Figure 3: Preserved variance for singular values using digit-specific encoding

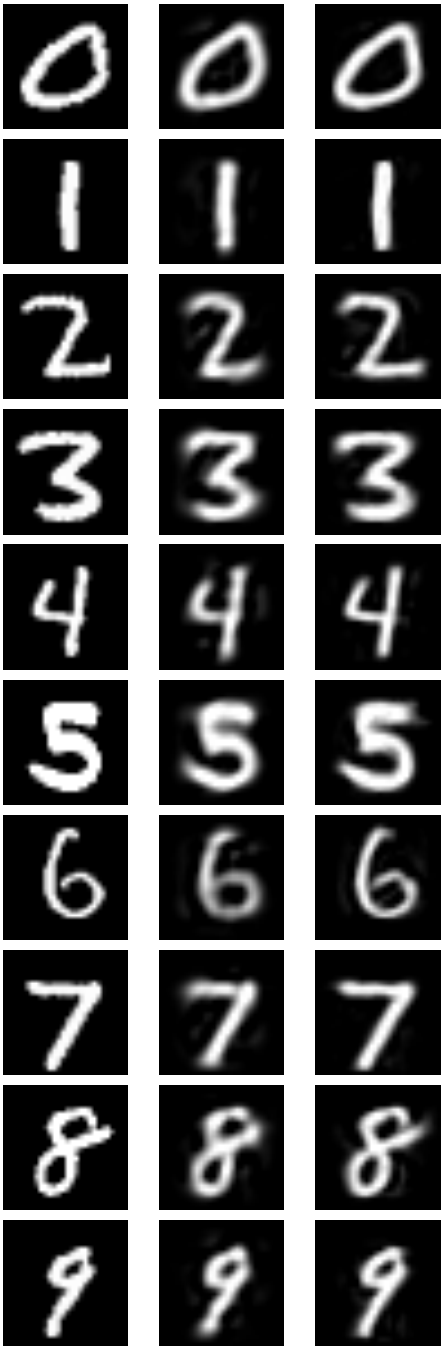

Figure 4: *Left:* Original. *Middle:* Generic PCA code. *Right:* Digit-specific PCA code.