A `for`–loop and a `break` statement

Consider a Newton iteration for $f(x) = x^2 - 2$,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - 2}{2x_n}, \quad n = 0, 1, 2, \ldots$$

```
x=1;
for n=1:20
  d=(x^2-2)/(2*x);
  x=x-d;
  if abs(d)<1e-12
      break
  end
end
```

Here if the test involving `d` is true then the `break` statement is executed and the you leave the loop.

A `for`–loop and a `continue` statement

On pages 13 and 14 the structure of an example is as follows.

```
% ...
count=0;
for k=1:500
  % ..randomly generate a matrix and get the eigenvalues

  if  some_tests_on_the_eigenvalues
    continue;
  end

  % ..display something
  count=count+1;
  if count==3
    break;
  end
end
```

Matrix operations

```
A=[1  2  3;
   3  2  1;
   4  0  2];

x=[1  1  1]';

b=A*x

r=(x'*b)/(x'*x)

C=A^2
```

Here * means matrix multiplication and similarly `A^2` is `A*A`.

In the statement starting with `x=` we get a column vector by transposing a row vector.

Using the colon notation with matrices

```
A=[  1    4    8    9    2;
     0    1    9    8    7;
     3    1    4    5    2;
     1    9    7    5    2;
     0    1    8    6    4];

r3=A(3, :)

c2=A(:, 2)

A3=A(2:4, 2:4)

E=A(4:end, 4:end)
```

`r3` is row 3 of `A`, `c2` is column 2 of `A`, `A3` is a 3×3 sub-matrix and as `A` is 5×5 `E` is a 2×2 sub-matrix.

The output

```
r3 =
     3     1     4     5     2
c2 =
     4
     1
     1
     9
     1
A3 =
     1     9     8
     1     4     5
     9     7     5
E =
     5     2
     6     4
```

Entry-wise operations

`.*`, `./` and `.^` are entry-wise operations. An example which also uses standard functions in a vectorised way is as follows.

Suppose we want to plot

$$f(t) = \sin(t) + 0.3\exp(-0.1t^2)\sin(10t).$$

```
t=linspace(0, 2*pi, 500);
y=sin(t)+0.3*exp(-0.1*t.^2).*sin(10*t);
figure(2)
plot(t, y)
```

The one statement involving `y=` achieves what the following 4 statements do in creating the vector `z`.

```
z=zeros(1, 500);
for k=1:500
  z(k)=sin(t(k))+0.3*exp(-0.1*t(k)^2)*sin(10*t(k));
end
```

Matrix operation/entry-wise operation comparison

```
A=[4  1  1;
   1  4  1;
   1  1  4];
A2=A*A
E2=A.*A
```

This creates the following.

```
A2 =
    18      9      9
     9     18      9
     9      9     18
E2 =
    16      1      1
     1     16      1
     1      1     16
```

A function file: the quadratic formula

$$x_{1,2} = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}.$$

```
function [x1, x2]=solve_quad(a, b, c)

d=b^2-4*a*c;
sd=sqrt(abs(d));
if d>=0
    x1=(-b-sd)/(2*a);
    x2=(-b+sd)/(2*a);
else
    x1=(-b-1i*sd)/(2*a);
    x2=(-b+1i*sd)/(2*a);
end
```

Store as `solve_quad.m`, 3 input arguments, 2 output arguments.