

MATLAB Lecture 6. Images and ODEs in MATLAB

Images are represented as three dimensional matrices: for every point we store with its coordinates (x, y) the red, green, and blue intensities.

6.1 Images in MATLAB

From the website of this course, you can download the image “Library.jpg”, using the web address <http://www.math.uic.edu/~jan/mcs320/Library.jpg> To load and display an image in MATLAB, we type the following sequence of commands:

```
>> x = imread('h:\Library.jpg'); % image saved on drive h:
>> image(x)
>> axis off
```

If we type `size(x)`, we see three numbers, the dimensions of the three dimensional matrix. In particular, x is a $m \times n \times 3$ matrix, the elements in $x(:, :, 1)$ are the red intensities, $x(:, :, 2)$ are the green intensities, and $x(:, :, 3)$ are the blue intensities. For example, to remove the blue intensities, we can do the following :

```
>> noblue = x;
>> noblue(:, :, 3) = zeros(size(x,1),size(x,2));
>> image(noblue);
```

The entries in x are unsigned 8-bit integers, whereas the operations for MATLAB are mainly done with doubles. Therefore, we need to convert to doubles before we manipulate the images:

```
>> dx = double(x) % convert to double
>> y = uint8(dx) % convert to unsigned 8-bit integer
>> image(y) % same image as before
```

We can make the picture more blue by multiplying the blue intensities with the factor 1.5:

```
>> moreblue(:, :, 1) = x(:, :, 1);
>> moreblue(:, :, 2) = x(:, :, 2);
>> moreblue(:, :, 3) = dx(:, :, 3)*1.5;
>> image(uint8(moreblue));
```

Assigning the red, green, and blue intensities to their average, we obtain a gray scale picture:

```
>> gy = (dx(:, :, 1) + dx(:, :, 2) + dx(:, :, 3))/3;
>> y(:, :, 1) = gy;
>> y(:, :, 2) = gy;
>> y(:, :, 3) = gy;
>> image(uint8(y));
```

By selection of submatrices and the proper choice of axes, we can zoom the picture.

We can encrypt a message by multiplying the red, green, and blue intensities with a random matrix. To decrypt we then multiply with the inverse of that random matrix. Assuming we loaded the image into x , we generate a blurring matrix as follows:

```
>> blur = randn(size(x,1)); % matrix to blur
>> key = inv(blur); % matrix to reconstruct image
```

To encrypt or decrypt we can use the m-file `crypto`:

```
function y = crypto ( c, i )
% returns either blurred or reconstructed image
y(:, :, 1) = c*i(:, :, 1);
y(:, :, 2) = c*i(:, :, 2);
y(:, :, 3) = c*i(:, :, 3);
```

Here is how we use `crypto`:

```
>> bx = crypto(blur,double(x)); % bx is matrix of doubles
>> y = crypto(key,bx); % restore original image x
>> image(uint8(y)); % display to verify
```

6.2. Solving ODEs in MATLAB and Octave

A first important difference between MATLAB and Octave is in the setup:

$$\text{MATLAB: } \frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0, \quad \text{Octave: } \frac{dy}{dx} = f(y, x), \quad y(x_0) = y_0.$$

In MATLAB, the independent variable x is the first argument of f , whereas in Octave, the dependent variable y is the first argument of f . Our test problem is $f = y$, $y_0 = 1$, for $x \in [0, 1]$.

In MATLAB, we use the command **ode45**:

```
matlab >> xspan = [0 1]; yzero = 1;           % x in interval [0,1], y(0) = 1
matlab >> f = inline('y','x','y');           % define f(x,y) = y, solve dy/dx = f
matlab >> [x y] = ode45(f,xspan,yzero);      % solve initial value problem
matlab >> y(end) - exp(1)
```

The output of the last command show the accuracy of the computed solution, as the exact solution for the test problem is the exponential function e^x . To plot, do `plot(x,y)`.

In Octave, the command to solve ODEs is **lsode**:

```
octave >> f = inline('y','y','x');           % dependent variable comes first!
octave >> xspan = [0 1]; yzero = 1;           % range and initial condition
octave >> y = lsode(f,yzero,xspan);          % initial condition before range
octave >> y(end) - exp(1)
```

Consistent with swapping the order of dependent and independent variables in the definition of f , in **lsode**, the initial condition comes before the range of x . In the output of **lsode** (in contrast to the output of **ode45**), the range of y is always the same as the range of **xspan**. If we would like to plot the solution for x in the range `0:0.01:1`, then we must define **xspan** also as `0:0.01:1`.

A simple model of a pendulum is given by the following second-order equation:

$$\frac{d^2}{dt^2}\theta(t) + \sin(\theta(t)) = 0 \quad \text{equivalent to} \quad \begin{cases} \frac{d}{dt}y_1(t) = y_2(t) \\ \frac{d}{dt}y_2(t) = -\sin(y_1(t)) \end{cases} \quad \text{where } y_1 = \theta.$$

The equivalent system at the right is the input to MATLAB or Octave.

Solving this problem in MATLAB for initial conditions $y_1(0) = 1$ and $y_2(0) = 0$ goes as:

```
matlab >> f = inline('[y(2); -sin(y(1))]', 't', 'y'); % define right-hand side f
matlab >> yzero = [1 0]; tspan = [0 10];           % initials and range
matlab >> [t y] = ode45(f,tspan,yzero);           % solve the ODE
matlab >> plot(t,y(:,1))                           % plot the values for theta
```

In Octave, the corresponding command sequence is

```
octave >> f = inline('[y(2); -sin(y(1))]', 'y', 't'); % y comes before t
octave >> yzero = [1 0]; tspan = [0 10];           % initials and range
octave >> y = lsode(f,yzero,tspan)                 % solve the ODE
```

If the function f is defined in a MATLAB or Octave script, then we must put quotes around f , that is: `'f'` is the first argument of both **ode45** and **lsode**.

6.3 Assignments

1. The image “easter-egg.jpg” (available at <http://www.math.uic.edu/~jan/mcs320/easter-egg.jpg>) shows an egg in a nest. Load the image into MATLAB and manipulate the image so that it looks as if the egg has been removed from the nest. To create this impression, replace the color encoding at the coordinates of the egg by the color encoding of pixels of the surrounding nest.
2. The contrast in an image can be enhanced by giving low intensity pixels an even lower intensity and increasing the intensity of the high intensity pixels. One way to achieve this, is the *gamma correction*, defined by $y_{ij} = [W \cdot (x_{ij})^\gamma / W]$, where W equals the maximal value corresponding to the resolution, and for some constant γ . The x_{ij} represents either the grayscale encoding of the original picture at pixel (i, j) , or is one of the RGB intensities at (i, j) .

Take your favorite picture (or use the “easter-egg.jpg”) and try to find a good value of γ to enhance the contrast.