Nathan Yee

Quantitative Engineering Analysis

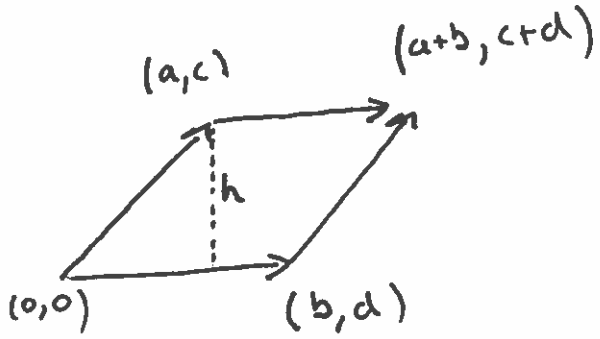
Version: 2016-03-21

Vectors, Matrices, and Data Arrays

QEA

Spring 2016

We shall start by completing the exercises that we did not get to in class on Day 3. Please skip any of the following exercises if you completed it in class.



Determinants

The determinant of a two by two matrix $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is defined by $det(\mathbf{A}) = ad - bc$.

1. Consider two vectors u and v in 2D which do not lie on the same line. These two vectors form the side of a parallelogram. Show that the area of the parallelogram is equal to the magnitude of the cross-product of the two vectors.

$A = \beta h$

$h = c$

$\beta = b$

$A = cb$

2. Show that the magnitude of $det(\mathbf{A})$ is equal to the area of a parallelogram formed by the column vectors of the matrix A.

$|det(A)| = \sqrt{(ad-bc)^2}$

$|det(A)| = \sqrt{(-bc)^2}$

$|det(A)| = bc$

3. What is the determinant of the rotation, reflection, scaling, and shearing matrices that we have already defined? Does this agree with your earlier findings on areas?

4. Explain the result that $det(\alpha\mathbf{A}) = \alpha^2\mathbf{A}$ where A is a two by two matrix, and α is a real number.

5. Explain the result that $det(\mathbf{AB}) = det(\mathbf{A})det(\mathbf{B})$, where A and B are two by two matrices.

6. Explain the result that $det(\mathbf{A}) = det(\mathbf{A}^T)$ where A is a two by two matrix.

Inverses

The inverse of the general two by two matrix $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is

$$\mathbf{A}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

1. Review the inverse matrices for rotation, scaling, etc and check that this result agrees with your previous findings.

2. Explain the statement *a matrix is invertible if and only if its determinant is non-zero.*

3. Explain the result that $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$, and confirm it for general two-by-two matrices.

4. Explain the result that $det(\mathbf{A}^{-1}) = \frac{1}{det(\mathbf{A})}$ where $\mathbf{A}$ is a two by two matrix.

$$\det(A^{-1})\det(A) = 1$$

$$\det(A^{-1}A) = 1$$

Orthogonal Matrices

A MATRIX IS *orthogonal* IF $A^{-1} = A^T$.

1. Which of our transformation matrices are orthogonal?

Scale => False

Rotation => True +1

Sheer => False

Translation => False

Identity => True +1

Reflection => True -1

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \neq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

2. Explain the result that the row vectors and column vectors of A are *orthonormal*.

Orthonormal:

Every set of vectors magnitude = 1

Set of vectors are orthagonal

$2D \Rightarrow 2x^2$

Geometric

$det(A) = det(A^T)$

3. Explain the result that $det(A) = \pm 1$ if A is orthogonal. If $det(A) = +1$ what kind of matrix is A? If $det(A) = -1$ what kind of matrix is A?

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$det(A^T A) = det(A^T)\, det(A)$

$det(I) = (det(A))^2$

$1 = (det(A))^2$

$det(A) = \pm 1$

THESE EXERCISES require a number of data files, which are located on the website: `1998DailyTempBos.csv`, `english_horn.wav`, and `house.png`. You'll want to put them in the MATLAB directory that you're using.

Vectors and Matrices as Data

So far, you have primarily thought of vectors as representations of points in space, and matrices that transform these vectors. Vectors and matrices are also extremely helpful when we deal with data that is not spatial, where the data can come from any number of sources: audio data, images, temperature data, stock prices, etc. We often think of data as being represented in arrays which can have multiple dimensions, much like matrices and vectors. For instance one may have a 1-dimensional array **a** which contains samples from an audio segment, with each sample corresponding to the instantaneous pressure associated with the sound.

We denote arrays and vectors/matrices in the same way, and whether we treat them as arrays or vectors depends on the context, which hopefully should be clear.

Exercises

1. In MATLAB, load two 1-dimensional arrays of data, `t` and `x`, using the following commands:

```
>> t = csvread('1998DailyTempBos.csv');
>> [x, fs] = audioread('english_horn.wav');
```

   Create the implied time vectors associated with these two arrays, and then plot the values of each array versus time. Label the axes of your plot appropriately. Note: you'll have to look up some things about the matlab commands and about Boston to make appropriate plots.

Array and Vector Operations

Now, once you have a set of data in an array, you can perform a number of operations on the data. **Array operations** are done on an element, by element basis. Consider the following two arrays of data

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

If we let c denote the element-by-element wise product of a and b, we have the following

$$\mathbf{c} = \begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ \vdots \\ a_m b_m \end{bmatrix}$$

Element-by-element operations are denoted in a number of different ways; a not uncommon notation for it is $\mathbf{a} \odot \mathbf{b}$. In MATLAB, element-by-element operations are denoted with a period: `a.*b` is the array multiplication of the vectors a and b, where as `a*b` is the vector multiplication of a and b.

Now, if we view a and b as vectors, given the fact that they are both $m \times 1$, their product is not defined. However, if we write a new $m \times m$ matrix D which has zeros on all its off-diagonal elements and $b_1, b_2, \cdots b_m$ on its diagonal, i.e.

$$\mathbf{D} = \begin{bmatrix} b_1 & 0 & \cdots & 0 \\ 0 & b_2 & 0 \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & b_m \end{bmatrix},$$

then the product of the matrix D with a results in c. Note that for one-dimensional arrays, operations which can be performed using element-by-element operations can be performed using appropriate matrix operations. Later, when we look at 2-dimensional arrays (in particular for image processing), we shall encounter operations which cannot be conveniently represented as matrix operations (or at least not without first changing the 2D array into a 1D array!).

Exercises

2. Consider the following matrices:

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

What do the following operations generate?

(a) $\mathbf{b} \odot \mathbf{a}$

(b) $\mathbf{ab}$

(c) $\mathbf{ab}^T$

(d) $\mathbf{Da}$

(e) $\mathbf{Daa}^T$

3. Consider an $m \times m$ matrix $\mathbf{D}$ defined as follows.

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \cdots & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & \cdots & \cdots & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & \cdots & \cdots & 0 & 0 & 0 & 1 \end{bmatrix}$$

In words, describe the vector that results when you take the product of the matrix $\mathbf{D}$ and an $m \times 1$ vector $\mathbf{v}$, whose i-th entry is v_i. How many additions and multiplications are required to compute the product $\mathbf{Dv}$?

$m \times m \cdot m \times 1 = m \times 1$

Resulting vector contains values equal to $\frac{1}{3}$ weights of next three numbers

m^2 additions + multiplicat

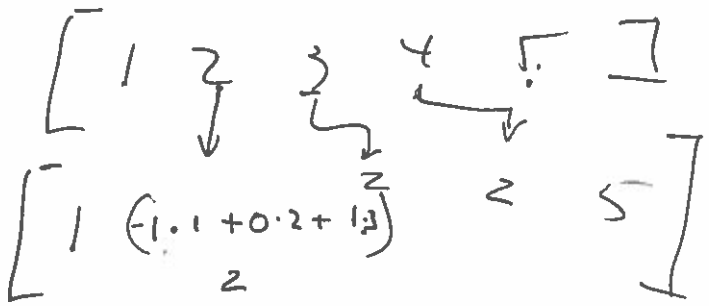
4. Create a 365×365 version of the matrix $\mathbf{D}$ in MATLAB. You will find it useful to exploit the command `diag`. Then operate with `D` on the vector `t` that you loaded above for the temperature in Boston. What do you expect the vector `Dt` to look like? Plot the results to make sure the answer makes sense.

Smooths out graph

Kernel operations

One can also perform what are known as kernel operations on arrays of data. We shall first consider one-dimensional arrays and move to two dimensions later. A 1-D kernel operation takes a 1-D data array and a kernel, which is another 1-D array with an odd number of elements (it's not strictly necessary for this to be odd), and produces a new 1-D array. The new array has the same number of elements as the original data array. The elements of the new array comprise weighted averages of adjacent elements in the original data array. The kernel determines the weights used in the weighted average.

There is no standard approach of what to do at the edges of the arrays where there are no adjacent samples to use in the weighted average. In this course, the edge elements of the new array will be equal to the corresponding edge elements of the original array. This idea is best illustrated by an example.



Consider an array of numbers a and a kernel k as follows

$$\mathbf{a} = \begin{bmatrix} 1 \\ 3 \\ 2 \\ -1 \\ 3 \\ 1 \end{bmatrix}, \quad \mathbf{k} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{3} \\ \frac{1}{4} \end{bmatrix}$$

When the kernel k is applied to a, the following array results

$$\begin{bmatrix} 1 \\ \frac{1}{2}\cdot 1 + \frac{1}{3}\cdot 3 + \frac{1}{4}\cdot 2 \\ \frac{1}{2}\cdot 3 + \frac{1}{3}\cdot 2 + \frac{1}{4}\cdot(-1) \\ \frac{1}{2}\cdot 2 + \frac{1}{3}\cdot(-1) + \frac{1}{4}\cdot 3 \\ \frac{1}{2}\cdot(-1) + \frac{1}{3}\cdot 3 + \frac{1}{4}\cdot 1 \\ 1 \end{bmatrix} \tag{1}$$

Notice that the edge points of the resulting array are identical to the original array. The other points are weighted sums of the corresponding point in the original array (with a weight of $\frac{1}{3}$), the point just before (with a weight of $\frac{1}{2}$), and the point just after (with a weight of $\frac{1}{4}$).

Exercises

5. With these definitions, find a kernel which when operating on a 1-D array has the same effect as the matrix multiplication by D given above. How many additions and multiplications are required to compute the result of applying this kernel to a data array of length m. Compare this with your answer to the corresponding question in Exercise 3.

6. Consider the following kernels. What does each do to a 1-D data array, conceptually speaking? To answer this question, think about what the ith element of the resulting array would be in terms of the elements of the input array.

$$\mathbf{d} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Convolution

The kernel operation above is closely related to the convolution operation, which is an operation used for filtering signals. Suppose that we have two arrays, an $m \times 1$ array x and an $n \times 1$ array

X: m×1 n≤m
h: n×1

h with $n \leq m$. Then the convolution of x and h produces an $(m + n - 1) \times 1$ dimensional array y. If we denote the i-th entry of the array x as x_i, and use the same notation to describe the entries of the other vectors, then we have

$$y_i = x_1 h_{i-1} + x_2 h_{i-2} + \cdots + x_m h_{i-m} \tag{2}$$

where we set $h_k = 0$, if $k < 1$ or $k > n$. You can compute the convolution of two arrays/vectors in MATLAB using the conv command. Convince yourself that except at the edge points, applying the length-m kernel k to an array x is equivalent to convolving x with h, where $h_i = k_{m-i}$. In other words, h is a flipped version of k. It may help to consider an example such as the following:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 3 \\ -1 \\ -2 \\ 3 \\ 1 \\ 2 \end{bmatrix}, \mathbf{k} = \begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix}, \mathbf{h} = \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix}$$

Note that when filtering is performed in software, it is most often done using a convolution operation, although one could in principle use matrix and kernel operations with appropriately defined matrices and kernels.

7. Load the wave file: english_horn.wav in MATLAB. You can use the following code snippet:

   ```
   >> [x, fs] = audioread('english_horn.wav');
   ```

   The vector x contains the samples of an audio segment, and fs is the sample rate in Hz, at which the samples were taken. Play this sound using the following command

   ```
   >> sound(x, fs);
   ```

8. What happens if you treat x as a vector and try to multiply it with the matrix D as defined above in MATLAB? Instead of using a matrix multiply, please try to accomplish the same thing by applying an appropriate kernel to the array of sound samples. Note that conceptually, the simplest thing to do is to use a for loop here. Now, convolve the array of sound samples with the kernel you just defined. Using the sound command, play the resulting data array on your computer and compare it to the original, unfiltered sound data. Please explain any differences. On the same axes, plot

graphs of the sound samples before and after processing. Please explain the differences in the graphs.

9. Now, define the following $m \times m$ matrix **H**:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & -1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & 1 & -1 & 0 & \cdots & \cdots & 0 \\ \vdots & \cdots & 0 & \ddots & \ddots & 0 & \cdots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & -1 & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 & -1 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix}.$$

Treating the sound data as being contained in a vector **x**, what is the result of multiplying the matrix **H** with the vector **x**? Note that you will likely have to define a kernel that accomplishes the same thing, and apply the kernel to the data array. Repeat the operation using a convolution with an appropriately defined array. What does the resulting data sound like compared to the original? Is the difference between the original, the result of the kernel operation, and the result of the convolution what you expect?

10. The filtering operation in Exercise 9 is a simple high pass filter. Since the operation takes the difference between successive samples in the data array, whenever there is a small change from one sample to the next sample, we end up with a small value. As a result, low frequency components in the signal are attenuated, and the audio sample sounds higher pitched. To illustrate this point further, define two cosine waves in MATLAB, one with high frequency and one with a much lower frequency as follows:

```
≫ t = linspace(0,10, 10000); % generate an array of 10000
values, linearly spaced from 0 to 10.  Use this as the
time index
≫ x_hf = cos(1000*t); % generate a high frequency cosine
≫ x_lf = cos(10t); % generate a low frequency cosine
```

Perform the same high-pass operation that you did to the data in Exercise 9. You can either use a convolution, or a kernel operation or both here. On the same axes, plot the graphs associated with both cosines. Zoom in to compare the two graphs.

11. Repeat the previous exercise using the following kernel.

$$\mathbf{k} = \begin{bmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix}$$

Low pass filter - ish

What does this kernel do to the high and low frequency cosines?

12. By generating and applying an appropriate kernel and/or convolution, generate a version of the audio data from Exercise 7, that has an audible echo. You will have to experiment a little with this, so use headphones! Also, note that if the sampling frequency is f_s, then, successive samples in the data array were taken $1/f_s$ time units apart.

$\begin{bmatrix} \frac{1}{2} \\ 0 \\ \vdots \\ 0 \\ \frac{1}{2} \end{bmatrix}$

13. For our next example, we shall use the average daily temperature in Boston in the year 1998. This data is provided in a file of 365 comma-separated values (CSVs), from the Average Daily Temperature Data Archive provided by the University of Dayton at

    `http://academic.udayton.edu/kissock/http/Weather/`

    . You can read the data in MATLAB using the following command

    ```
    >> t = csvread('1998DailyTempBos.csv');
    ```

    Find a matrix A such that the matrix product At results in a 12 × 1 vector with the average temperature for each month of the year. Is it possible to do this calculation using a convolution, or a kernel operation?

31 365-31

Jan $\begin{bmatrix} \frac{1}{31} \cdots \frac{1}{31}, 0000\ldots \\ \text{Feb } 0,0,0,0,\ldots, \frac{1}{29}, \frac{1}{29} \cdots \frac{1}{29}, \\ \vdots \\ \text{Dec } 0,0,0,\ldots\ldots \frac{1}{31}, \frac{1}{31} \cdots \end{bmatrix}$

$12 \times 365 \quad 365 \times 1$

12×1

convolutions produce $(m+n-1) \times 1$ arrays

So we must satisfy

$365 + n - 1 = 12$

$n = -352$

-352×1 ∴

14. Find a kernel k which can be used to compute 5-day moving averages of the temperature. Use this kernel to plot a smoother version of the temperature vs. day graph.

$k = \begin{bmatrix} 1/5 \\ 1/5 \\ 1/5 \\ 1/5 \\ 1/5 \end{bmatrix}$

15. Consider an alternative kernel k defined below

$$\mathbf{k} = \begin{bmatrix} 0.036 \\ 0.241 \\ 0.446 \\ 0.241 \\ 0.036 \end{bmatrix}.$$

Apply this kernel to the vector of temperature values and plot the resulting data. How does this compare to the graph in the previous question. Describe the qualitative difference between the averaging operation resulting from this kernel and the kernel from the previous part.

Kernal similar

K gives more weight to the middle values for each operation

16. Apply the kernel from Exercise 9 to the temperature data and plot the resulting array. In words, describe what the large positive and negative spikes in the data correspond to.

Large pos + neg will lead to the peaks in the filtered data. This is looking at the difference between large and small values.

Kernels in two dimensions (Optional)

Thus far we've thought about data arrays that are one dimensional. But of course, data can come in more than one dimension. Grayscale images, for example, are represented by two dimensional data arrays. The ideas of kernels and convolution translate easily to two dimensions (and higher dimensions).

Recall that in 1D, the application of the kernel **b** of length 5 to a 1D array **a** of length m gave us a resulting array **c** where

$$c_i = \sum_{j=1}^{5} b_j a_{i+j-3}$$

Note that this definition only applies for values of $2 < j < m$.

We can construct a similar definition of a kernel operation on a 2D array. The 2-D kernel operation takes an array of numbers and a 2-D kernel. The kernel is typically square and has an odd number of rows and columns. The kernel operation replaces each element of the original array by a weighted average of itself and its nearby points. As in the 1-D case, the kernel defines the weights applied to each of the points in the weighted average. For instance, the following kernel

$$\mathbf{K} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

replaces each point in an array with an average of itself and the eight points adjacent to it. A natural question to ask is what happens to the points at the edge or corner of an array. There isn't a standard way of handling this, but commonly used options include, not applying the kernel operations to those points, and wrapping around. In this course, we shall default to not doing the kernel operations on the edges of arrays.

For example, for a 3×3 kernel **b**, and a $n \times p$ data array **a**, the resulting array **c** is

$$c_{i,j} = \sum_{k=1}^{3} \sum_{m=1}^{3} b_{k,m} a_{i+k-2,j+m-2}$$

except, of course, at the edges of the array.

More generally consider the following array

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \cdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix}$$

and the following 3×3 kernel

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{21} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$$

The ij-th entry of the array that results from applying the kernel $\mathbf{K}$ to the array $\mathbf{A}$ is as follows

$$\begin{aligned} b_{ij} = {} & k_{11}a_{(i-1)(j-1)} + k_{12}a_{(i-1)j} + k_{13}a_{(i-1)(j+1)} \\ & + k_{21}a_{i(j-1)} + k_{22}a_{ij} + k_{23}a_{i(j+1)} \\ & + k_{31}a_{(i+1)(j-1)} + k_{32}a_{(i+1)j} + k_{33}a_{(i+1)(j+1)} \end{aligned} \tag{3}$$

The following figure illustrates a 3×3 kernel applied to an array of numbers.

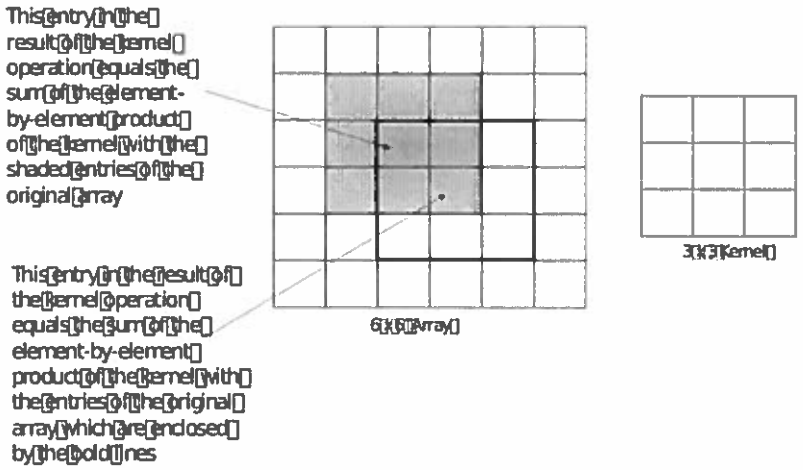


Figure 1: Illustration of a 3×3 kernel operation on an array.

For a more specific example, consider the following array of numbers and kernel

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & -3 & 1 & 2 \\ 2 & -2 & 4 & 3 & 1 \\ 3 & -4 & 5 & 1 & -1 \\ 4 & -4 & 3 & 1 & -1 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} -1 & 1 & 2 \\ -2 & 1 & 3 \\ 1 & 3 & 1 \end{bmatrix}$$

The resulting array of number B when the kernel K is applied to the array A is

$$\mathbf{B} = \begin{bmatrix} 1 & 2 & -3 & 1 & 2 \\ 2 & (-1+2-6-4-2+12+3-12+5) & 26 & 13 & 1 \\ 3 & 4 & 34 & -6 & 1 \\ 4 & -4 & 3 & 1 & -1 \end{bmatrix}.$$

The term in the parenthesis above is provided to help you understand what operations are used to perform the kernel operation.

Exercises

17. For each of the following kernels, discuss what you *expect* would happen to a grayscale image if you applied the kernel to the image array.

$$\mathbf{A} = \frac{1}{11}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{Blur}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{Invert}$$

$$\mathbf{C} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{Sharpen}$$

Blur

Edge Detection

Sharpen

18. For this exercise, we'd like you to write a short MATLAB code that calculates the resulting 2D array produced by operating on a grayscale image with a 2D kernel, and use it to test your intuition on the kernels above.
    (a) Load the 2D array that is a grayscale image of a house using the command `» A=imread('house.png')`
    (b) Using a nested `for` loop, calculate the resulting image for a given kernel. Your code should be well-documented and written in a way that allows you easily to change the kernel.
    (c) Display the original image and the resulting image for each of the kernels above. Was your intuition right?
19. How does convolution in 2D work? Look it up, and write an expression for the i,jth element that results from the convolution of a matrix A with a matrix H. Compare to the kernel operation defined above.
20. Repeat the image operations above using `conv2`.

101 & 111

101 →

T T
T F
F T
F F