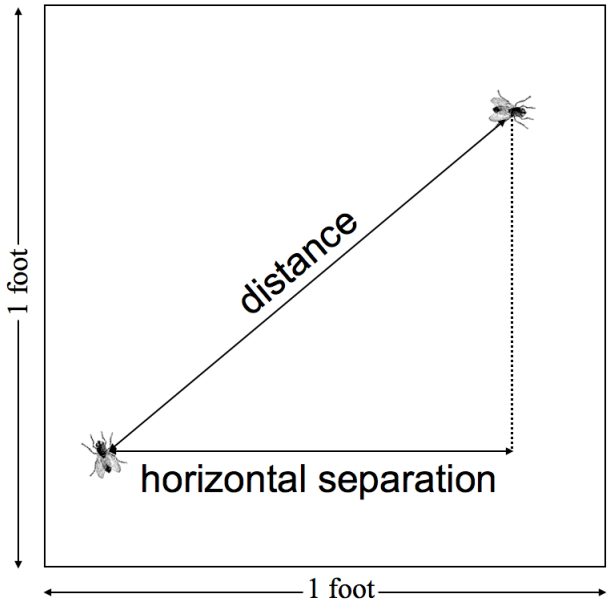
# MATLAB Demonstration

## MATH2070: Numerical Methods in Scientific Computing I

Location: http://people.sc.fsu.edu/~jburkardt/classes/math2070_2019/matlab_demo/matlab_demo.pdf

---




*Two flies land in the unit square. On average, how far apart are they?*

MATLAB lets you implement numerical algorithms, and interactively experiment with them. It has many numerical algorithms as built-in commands, and uses a linear-algebra notation that easily expresses vector and array operations. It includes extensive graphics. MATLAB "toolboxes" supply additional packages of very specialized algorithms for symbolic mathematics, machine learning, signal analysis, image processing, financial mathematics, and bioinformatics.

**MATLAB Demo**

*How can we use a scientific programming language like MATLAB to:*

- *create a mathematical model of a real-world problem;*
- *devise an algorithm that can simulate the problem;*
- *write a computer program to implement the algorithm;*
- *experimentally search for solutions;*
- *graphically illustrate the properties of the solution;*

Since this is a graduate class, and many of you have some experience with MATLAB already, we will simply start a MATLAB computation, explaining a few things as we go along, and then talk more about MATLAB after we get our result.

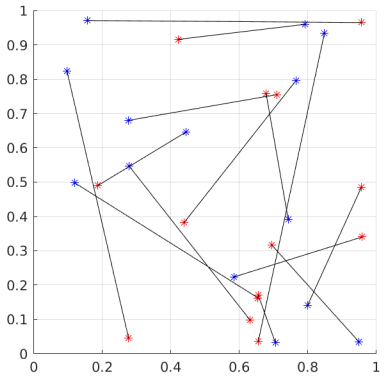
This particular fly-paper problem can be solved analytically, but we are interested in how MATLAB can simulate the problem, estimate the answer, and produce convincing tables and plots. We are studying a random process, and trying to say something (the average) about a huge number of cases. We begin by simulating a single case.

# 1 One simulation

Being mathematicians, we assume the paper is a $1 \times 1$ square. To simulate the randomly chosen position of the first fly, $p_1 = (x_1, y_1)$, we need to choose uniform random values between 0 and 1. Then we can place the second fly. Once we have their locations, we simply use the Pythagorean theorem to get the distance.

1. First fly: `p1 = rand(2,1)`
2. Second fly: `p2 = rand(2,1)`
3. Distance: `d = norm ( p1 - p2 )`
4. Plot:

```
    hold ( 'on' )
    plot ( p1(1), p1(2), 'r*' )
    plot ( p2(1), p2(2), 'b*' )
    plot ( [p1(1),p2(1)], [p1(2),p2(2)], 'k-' )
    axis ( [0,1,0,1] )
    hold ( 'off')
```




# 2 Many Simulations, Averaged

The average distance between random flies can be estimated by simulating many examples. In MATLAB, the *for* loop lets us repeat an operation many times. We can initialize a variable `average` to zero, loop over `n` simulations, adding each distance to `average`, and then dividing by `n`:

```
  average = 0.0;
  n = 10;
  for test = 1 : n
    p1 = rand(2,1);
    p2 = rand(2,1);
    d = vecnorm ( p1 - p2 );
    average = average + d;
  end
  average = average / n
```

How much variation do we see in our answers? Let's try `n = 100` and compare answers. What about `n = 1000`?

**How large a multiple of 10 do we need before everyone's answer agrees to three decimal places?**

## 3 Using Vector Notation

```
  n = 10;
  p1 = rand(2,n);
  p2 = rand(2,n);
  d = vecnorm ( p1 - p2 );
  average = mean ( d )
```

Notice that, when moving to the vector version of the calculation, each of the three ways of computing the distance `d` had to be adjusted in some way.
The advantages of writing a calculation in the vector version are that the code is somewhat more compact, it is easier to see that the computation could be done in parallel, and in particular, MATLAB is able to compute many vector operations very efficiently.

## 4 Timing Comparison

Let's compare the scalar and vector codes when `n` is large.
First, let's rewrite our scalar calculation as an *M-file*, which we will call *scalar_code.m*, with the value of `n` an input parameter.

```
function average = scalar_code ( n )
  average = 0.0;
  for test = 1 : n
    p1 = rand(2,1);
    p2 = rand(2,1);
    d = vecnorm ( p1 - p2 );
    average = average + d;
  end
  average = average / n
  return
end
```

Doing this means we have created our own MATLAB command, which can be executed with a single line:

```
  average = scalar_code ( n )
```

We can similarly create *vector_code.m*. Now we could compare the speed of the two methods by simply waiting for each to complete, or using a stopwatch, but instead, we will use the MATLAB commands `tic` and `toc`:

```
  for logn = 1 : 6
    n = 10^logn;
    tic;
    scalar_code ( n );
    t1 = toc;
    tic;
    vector_code ( n );
    t2 = toc;
    fprintf ( 1, '%d  %g  %g  %g\n', n, t1, t2, t1/t2 );
  end
```

## 5 A convergence table

The average is an average over an infinite number of cases. We are estimating that average using `n` cases. How do we convince ourselves that the estimate is a good approximation? One way is to repeat the calculation for larger values of `n` and make a table. If the estimates seem to converge, then this suggests that our many choices of `n` seem to point towards a common answer *although we can't be sure this answer is right!*.

```
  average_old = 0.0;
  for logn = 1 : 6
    n = 10^logn;
    average = vector_code ( n );
    fprintf ( 1, '%7d  %14.6f  %14.6f\n', n, average, average - average_old );
    average_old = average;
  end
```

Our results show that even a value of `n=1,000,000` only gives us confidence in the first two decimals of our estimate.

```
>> converge
      10        0.476492        0.476492
     100        0.535507        0.059016
    1000        0.518386       -0.017121
   10000        0.523612        0.005226
  100000        0.521069       -0.002543
 1000000        0.521247        0.000178
```

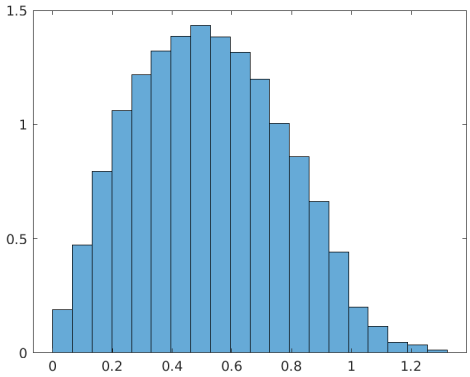
# 6 A histogram of the results

In the problem we are thinking about, possible distances $d$ range between 0 and $\sqrt{(2)}$. Some distances are more likely than others, as described by a probability density function. If we choose `n` large enough, we can make a histogram of our results that will suggest what the true PDF looks like.
The basic command is simply

```
  histogram ( d )
```

but I want to specify the number of bins to be 20, and I want the vertical $y$ axis to be scaled like a PDF, rather than a frequency count.

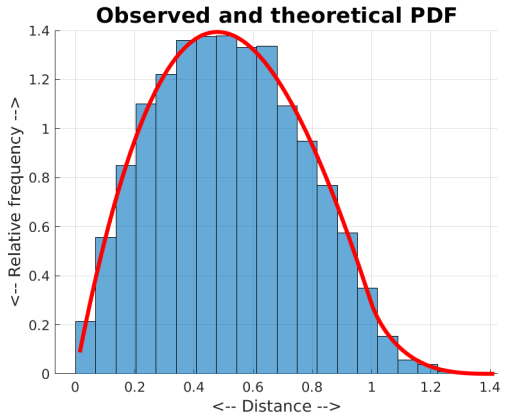
```
  n = 10000;
  p1 = rand(2,n);
  p2 = rand(2,n);
  d = vecnorm ( p1 - p2 );
  bins = 20;
  histogram ( d, bins, 'Normalization', 'pdf' )
```



# 7 Histogram versus PDF

For this particular problem, there happens to be an exact formula for the PDF. We can plot it together with the histogram, to see how we did. The formula for the exact PDF is a little messy:

$$\text{pdf}(d) = \begin{cases} 2d(d^2 - 4d + \pi) & \text{if } d <= 1.0 \\ 2d(4\sqrt{d^2 - 1.0} - (d^2 + 2 - \pi) - 4\arctan(\sqrt{d^2 - 1})) & \text{if } 1.0 < d \end{cases}$$




# 8 Challenge:

Consider a new problem: *Two flies land inside the unit* **disk**. *On average, how far apart are they?*

To pick a random point $(r, \theta)$ in the unit circle:

1. pick a random number $r_1$, and set $\theta = 2\pi r_1$;
2. pick a random number $r_2$, and set $r = \sqrt{r_2}$;

If you are surprised by the computation of $r$, plot 1,000 random points this way, and than compare with 1,000 random points computed by $r = r_2$.

Generate a lot of pairs (the right way!), save the distances in a vector $d$, and then histogram the results. Compare your histogram to a plot of the exact PDF for the distance $0 \leq d \leq 2$:

$$pdf(d) = \frac{d}{\pi}(4\arccos(\frac{d}{2}) - d\sqrt{4 - d^2})$$