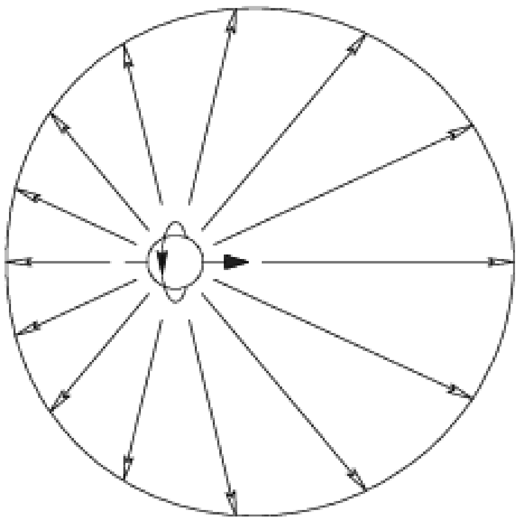
Exercise 4: Angular Distribution in Muon Decay

The positive muon decays via the parity-violating weak interaction to produce an energetic positron and two neutrinos:

$$\mu^+ \to e^+ + \nu_e + \bar{\nu}_\mu \,.$$

The angle θ at which the positrons are emitted is preferentially in the direction of the muon spin (polarization) at the instant of decay.



Schematic illustration of the angular distribution of positrons from muon decay. The length of the arrow represents the relative probability of positron emittance in the direction of the arrow. The angle θ is with respect to the direction of the muon spin indicated by the solid arrow, i.e., in the direction of the positive x-axis.

More specifically, the probability density of a positron being emitted in a direction at an angle θ to the spin is given by

$$f_{\cos\theta}(\cos\theta|\alpha) = \frac{1+\alpha\cos\theta}{2} \,. \tag{1}$$

Here α is called the "asymmetry".

More information about muon decay and the (experimental) difficulties involved can be found on `http://musr.physics.ubc.ca/theses/Morris/node8.html` and `http://phycomp.technion.ac.il/~zaher/proposal/node19.html`. Here we will consider an artificial data set, drawn from the distribution (1).

a. Equation (1) gives the probability density function in terms of $\cos\theta$. What is the probability density function in terms of θ itself?

It will be most convenient, however, to stick to the density function (1) in terms of $x = \cos\theta$.

b. Build a function `randangle` that takes parameters `alpha`, `m`, and `n`, and returns the matrix `x` of size `m` $\times$ `n` with (pseudo-)random values drawn according to the distribution (1).

```
function x = randangle(alpha,m,n);
```

Try to use MATLAB's matrix routines as much as you can. Check your implementation by making a histogram of many randomly drawn values.

Our (artificial) data set consists of the following $n = 100$ observations:

$$\underline{\theta} = [\begin{array}{rrrrrrrrrr}
2.6764 & 0.8894 & 1.4354 & 1.5006 & 0.9046 & -1.3146 & 1.6474 & -0.1975 & -1.2747 & -0.9988 \\
1.1555 & -1.6281 & 2.1967 & -1.0610 & 1.2414 & 0.6162 & -2.9699 & 1.5531 & 2.5216 & 2.1182 \\
-1.8529 & -1.4210 & 1.9418 & 1.4794 & -1.7858 & 1.8154 & 2.8942 & -0.9977 & -0.5202 & -2.0464 \\
1.5976 & 1.7503 & 2.2237 & -2.0187 & -0.3506 & 0.7647 & -2.4849 & 1.1087 & -0.8247 & 0.8363 \\
-0.4932 & -0.7828 & 2.3109 & 0.5433 & -0.0738 & 2.4771 & 2.4685 & 1.3219 & -1.2041 & -1.2495 \\
-0.1149 & -1.3569 & -1.7768 & -1.3162 & -1.3235 & 0.4888 & 0.4864 & 1.9602 & -2.7259 & -1.3423 \\
0.8733 & 2.2035 & -1.8747 & 1.0333 & -0.8709 & 2.0805 & 2.8855 & -3.0309 & 1.7713 & 0.5745 \\
1.0792 & 0.7908 & -1.6818 & 2.0775 & 0.2353 & 1.5304 & -1.8393 & -1.7928 & -1.9249 & 2.4183 \\
0.8377 & -0.4723 & -0.1739 & -1.3559 & 1.8984 & 1.4318 & 1.1355 & -1.0589 & 0.9312 & 2.7828 \\
-1.2460 & 2.6038 & 1.6926 & 0.9147 & 2.3276 & -1.1876 & 0.8835 & -2.1147 & 1.5013 & -2.7682
\end{array}];$$

This data can also be found on the website `http://www.mbfys.kun.nl/~tom/teaching.html` or downloaded directly from `http://www.mbfys.kun.nl/~tom/files/angulardata.dat`.

c. Use the method of moments to estimate α from this data.

d. Use the parametric bootstrap method to generate $B = 10000$ samples from the (approximate) sampling distribution of the method of moments estimate $\hat{\alpha}$. Draw a histogram of these samples.

e. Compute the mean and standard deviation of the sampling distribution, both from your bootstrap samples and analytically. Are these consistent? Is the method of moment estimate unbiased in this case? How would you report your estimate of the parameter α given these results?

f. Plot the log likelihood for different values of α. Give an analytic expression for the maximum likelihood estimate. Use either MATLAB's `fzero` or one of the routines you yourself constructed in Exercise 1 (e.g., the bisection routine) to find the maximum likelihood estimate for the given data set.

g. Use the parametric bootstrap to generate 10000 samples of the approximate sampling distribution of the maximum likelihood estimate. Is the maximum likelihood estimate in theory biased or unbiased? What do the bootstrap samples suggest? Compare the variance of the "maximum likelihood sampling distribution" with the "method of moments sampling distribution".

h. (optional) Find a nice way to visualize your result.

Hints for Exercise 4.

a. See the syllabus, section 2.3.3. Check that

$$\int_{-1}^{1} d\cos\theta \, f_{\cos\theta}(\cos\theta|\alpha) = 1$$

and also

$$\int_{-\pi}^{\pi} d\theta \, f_\theta(\theta|\alpha) = 1 \, .$$

Note that the lengths of the arrows in the figure are proportional to $f_{\cos\theta}(\cos\theta|\alpha)$. Whereas $f_{\cos\theta}$ has a mode at 0, f_θ does not.

b. There are several ways of doing this, a straightforward approach uses the following proposition:

Proposition 1 *Let U be uniform on $[0,1]$, and let $X = F^{-1}(U)$. Then the cumulative distribution function of X is F.*

Proof. $P(X \le x) = P(F^{-1}(U) \le x) = P(U \le F(x)) = F(x)$ $\square$

1. MATLAB's `rand` generates uniform (pseudo) random values x between 0 and 1.
2. Compute $F(x)$ and $F^{-1}(x)$ for the density function (1). Although in principle F does not have a unique inverse you can reason which one is appropriate.
3. Although you can implement the routine using for-loops, MATLAB works a lot faster if you manipulate entire matrices instead of individual entries in for-loops. You can e.g. use `rand` to generate a matrix of (pseudo) random numbers, and make F^{-1} work on matrices.

c. Store the data from the website locally and read it into MATLAB using `load angulardata.dat`. This gives you a variable named `angulardata` with angles θ.

Express the mean of the density (1) in terms of the parameter α. Transform the angles θ into $x = \cos\theta$, compute the sample average, and deduce $\hat{\alpha}$. See section 4.3 in the syllabus (be careful: in the syllabus θ stands for the parameter(s) of the probability model, not for the data).

d. The parametric bootstrap is described in section 4.6 of the syllabus and should be relatively straightforward to implement given this description:

- Generate many different samples $\underline{x}_b$ of length $n = 100$ given the method of moment estimate $\hat{\alpha}$ and using the routine `randangle` that you have constructed before.
- Compute a new method of moment estimate $\hat{\alpha}_b$ for each of these samples.

Be careful: MATLAB also has a bootstrap function (called `bootstrp`), but this is an implementation of the so-called nonparametric bootstrap method that will be discussed in chapter 7.

e. Following example 4.1 on the sampling distribution for the Poisson distribution in section 4.4 of the syllabus, you should be able to compute the mean and variance of the sampling distribution for the method of moment estimate in this case. Note that computing the exact sampling distribution is much more complicated here. The notion of unbiasedness is discussed in both section 4.4 and section 4.8.

f. Maximum likelihood estimation is explained in section 4.7 of the syllabus. Taking the derivative of the log likelihood with respect to α, you will find a nonlinear equation for α that must be solved iteratively. To this end, you can use MATLAB's `fzero`. The most elegant way is to first construct an m-file that implements the derivative of the log likelihood with respect to the parameter α and takes as inputs this parameter α and the data vector $\underline{x}$. The name of this function, e.g., `gradangular` is then given to `fzero` (see `help fzero` for the correct syntax; note that the name of the function should be between ‘’). For example, a call could look like `alpha = fzero('gradangular',alpha0,[],x);`, with `alpha0` an initial estimate (e.g., the method of moment estimate) and `x` the data vector $\underline{x}$. Another alternative is to minimize "minus the log likelihood" with MATLAB's `fminunc`. However, although probably a little faster especially when gradient information is included, the syntax is somewhat more complicated. For example, when incorporating gradient information the function to be minimized (minus the log likelihood) should return the function value itself as its first argument and its gradient as the second argument. Furthermore, options have to be included that can be set with something like `options = optimset('GradObj','on');`.

g. Bootstrapping proceeds as in **d.**, but is now slightly more complicated since we have to use some iterative procedure as a subroutine to find the maximum likelihood estimate $\hat{\alpha}_b$ given bootstrap sample $\underline{x}_b$. Drawing 10000 samples might take some time, so perhaps you want to add some print statements using `disp` or `fprintf` to keep track of the progress.

Note that the question is whether the maximum likelihood estimate is biased or unbiased in this specific finite-sample case, i.e., not whether it is asymptotically unbiased.

h. See what this does:

```
clf
hold on     % little trick, gridlines appear when left out
alpha = 0.6;
theta = -pi:0.01:pi;
f = (1 + alpha*cos(theta))/2;
polar(theta,f,'--');
axis off
```