

SDA/PH4515 Discussion Notes Week 7

Problem sheet 4 solutions:

$$1a) \quad x_N = \sum_{i=1}^N r_i, \quad r_i \text{ indep. } \& \sim U[0,1]$$

$$E[x_N] \equiv \mu_N = E\left[\sum_{i=1}^N r_i\right]$$

$$= \sum_{i=1}^N \underbrace{E[r_i]}_{\substack{= \\ \frac{1}{2} \text{ from lect. notes}}} = \sum_{i=1}^N \frac{1}{2} = \underline{\underline{\frac{N}{2}}}$$

$$V[x_N] \equiv \sigma_N^2 = V\left[\sum_{i=1}^N r_i\right]$$

$$= \sum_{i=1}^N \underbrace{V[r_i]}_{\substack{= \\ \frac{1}{12} \text{ from lect. notes}}} \quad (\text{since all } r_i \text{ indep.})$$

$$= \frac{N}{12}$$

$$\Rightarrow \underline{\underline{\sigma_N = \sqrt{\frac{N}{12}}}}$$

 $\Rightarrow$ By construction

$$y_N = \frac{x_N - \mu_N}{\sigma_N} \quad \text{is "standardized": } E[y_N] = 0$$

$$V[y_N] = 1$$

$$= \sqrt{\frac{12}{N}} \left(\sum_{i=1}^N r_i - \frac{N}{2} \right)$$

$$2) \quad f(x) = 3x^2, \quad 0 \leq x \leq 1$$

2a) Transformation method

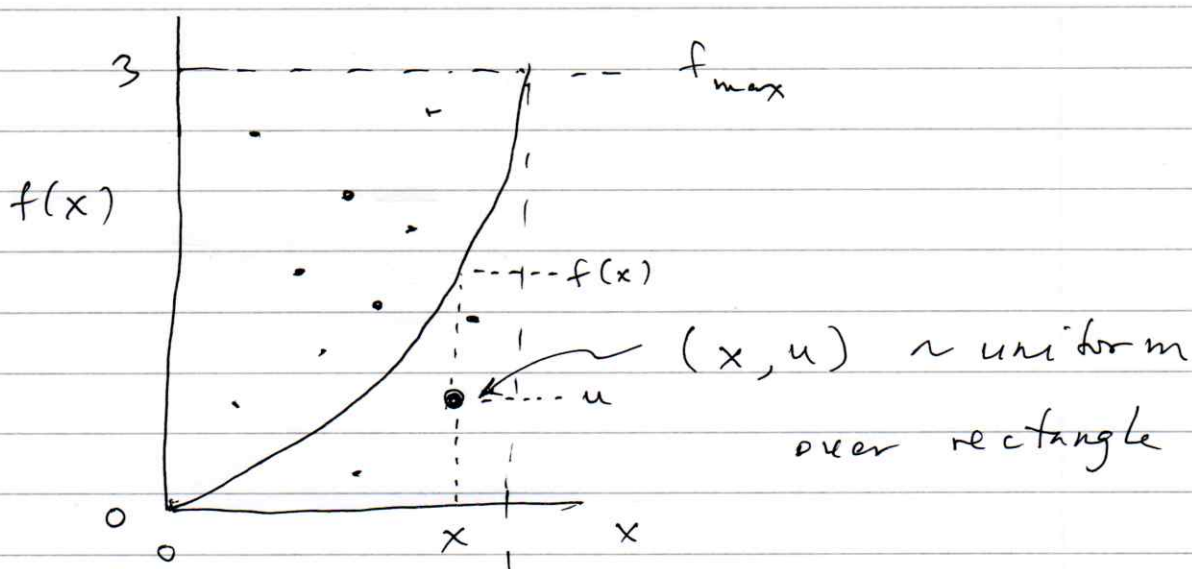
First find cumulative distribution

$$F(x) = \int_0^x 3x'^2 dx' = x^3$$

$$\text{Set } F(x) = x^3 = r \quad \leftarrow r \sim U[0,1]$$

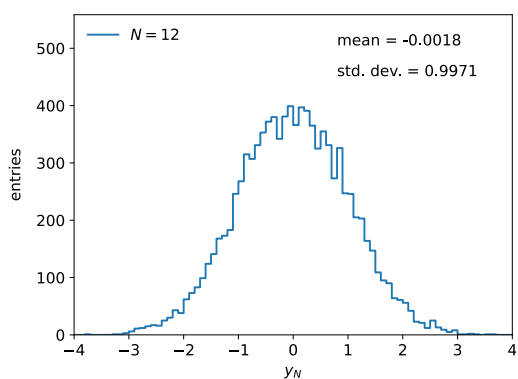
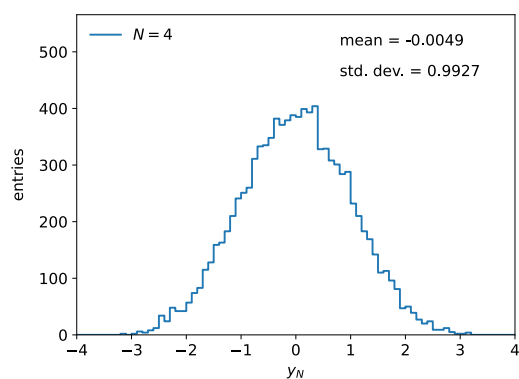
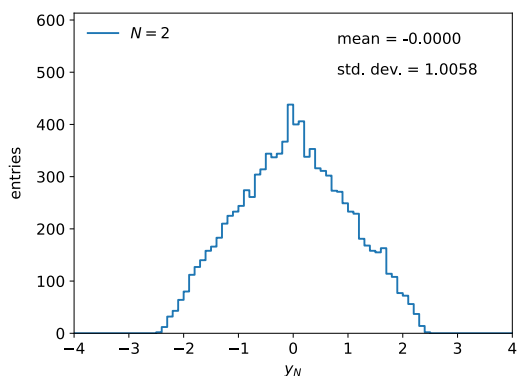
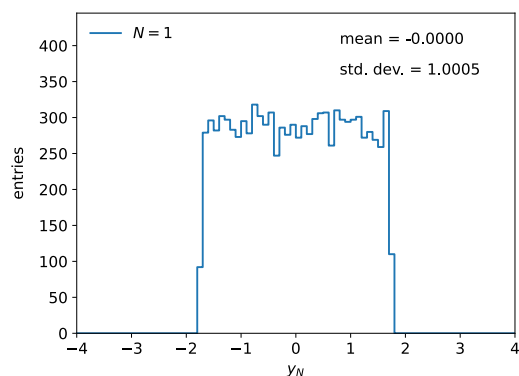
$$\Rightarrow \underline{x(r) = r^{1/3}}$$

2b) Acceptance - rejection method

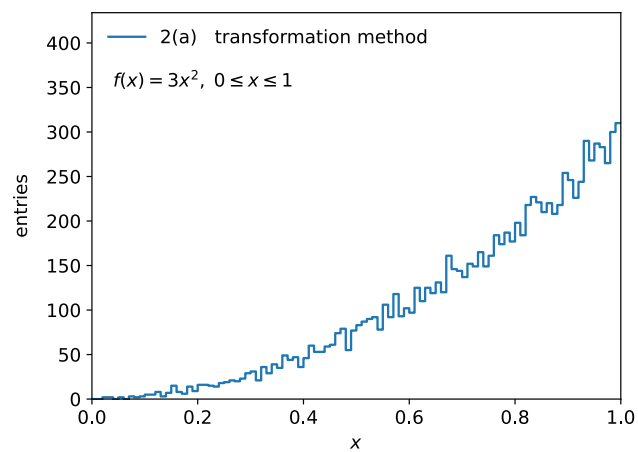


If $u < f(x)$, accept x

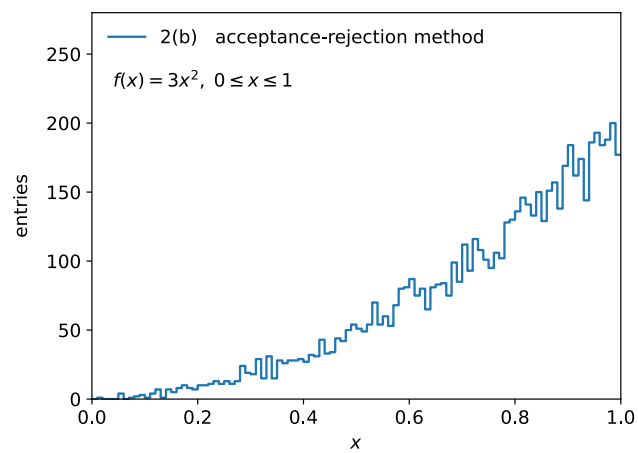
1(b) [7 marks] (code for Exercises 1 and 2 at end). Histogram of y_N for $N = 1, 2, 4, 12$. The means and standard deviations are indicated on the plots. As shown, they are close to $\mu = 0$ and $\sigma = 1$, as must emerge by construction for the standardized variable y_N .



Histogram of values generated from the transformation method:



2(b) [5 marks] Histogram of values generated with the acceptance-rejection method (code at end):



Python code for questions 1 and 2:

```
# simpleMC.py -- simple Monte Carlo program to make histogram of uniformly
# distributed random values and plot
# G. Cowan, RHUL Physics, November 2022

import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# Solution to problem sheet 4

# First make a simple routine to plot histograms
def plotHist (hist, bin_edges, xLabel, yLabel, plotLabel):
    matplotlib.rcParams.update({'font.size':12})      # set all font sizes
    xMin = bin_edges[0]
    xMax = bin_edges[len(bin_edges)-1]
    yMin = 0.
    yMax = np.max(hist)*1.4
    binLo, binHi = bin_edges[:-1], bin_edges[1:]
    xPlot = np.array([binLo, binHi]).T.flatten()
    yPlot = np.array([hist, hist]).T.flatten()
    fig, ax = plt.subplots(1,1)
    plt.gcf().subplots_adjust(bottom=0.15)
    plt.gcf().subplots_adjust(left=0.15)
    ax.set_xlim((xMin, xMax))
    ax.set_ylim((yMin, yMax))
    plt.xlabel(xLabel, labelpad=5)
    plt.ylabel(yLabel, labelpad=10)
    plt.plot(xPlot, yPlot, label=plotLabel)
    ax.legend(loc='upper left', frameon=False)
    plt.show()

# The function for yN returns an array of values
def yArr(N, numVal):
    rSum = np.zeros(numVal)
    for i in range (N):
        rSum += np.random.uniform(0., 1., numVal)
    return np.sqrt(12./N)*(rSum - N/2.)

numVal = 10000
nBins = 80
yMin = -4.
yMax = 4.

for N in [1, 2, 4, 12]:
    y = yArr(N, numVal)
    histLabel = '$N = $' + str(N)
    yHist, bin_edges = np.histogram(y, bins=nBins, range=(yMin, yMax))
    plotHist(yHist, bin_edges, r'$y_{N}$', r'entries', histLabel)
    plt.figtext(0.6, 0.81, f'mean = {y.mean():.4f}')
    plt.figtext(0.6, 0.74, f'std. dev. = {y.std():.4f}')

# 2(a) Generate random values according to  $f(x) = 3x^2$  ( $0 < x < 1$ )
# with transformation method; here  $x = r*(1./3.)$ 
numVal = 10000
nBins = 100
rData = np.random.uniform(0., 1., numVal)
xMin=0.
xMax=1.
xData = pow(rData, 1./3.)
xHist, bin_edges = np.histogram(xData, bins=nBins, range=(xMin, xMax))
plotHist(xHist, bin_edges, r'$x$', r'entries', r'2(a) transformation method')
plt.figtext(0.18, 0.74, r'$f(x) = 3x^{\{2\}}, \; ; \; 0 \leq x \leq 1$')
```

```

plt.show()

# 2(b) Repeat using the acceptance rejection method
numVal = 20000
nBins = 100

def f(x):
    return 3. * x**2

xMin = 0.
xMax = 1.
fMax = 3.
r1 = np.random.uniform(0., 1., numVal)
x = xMin + r1*(xMax - xMin)
r2 = np.random.uniform(0., 1., numVal)
u = fMax*r2
xDat = x[u<f(x)]
nBins = 100
xHist, bin_edges = np.histogram(xDat, bins=nBins, range=(xMin, xMax))
plotHist(xHist, bin_edges, r'$x$', r'entries', r'2(b) acceptance-rejection method')
plt.figtext(0.18, 0.74, r'$f(x) = 3x^2$, \; 0 \leq x \leq 1$')
plt.show()

```

C++ code for questions 1 and 2:

```

// Solution to Computing and Statistics Monte Carlo Problem
// Glen Cowan, RHUL, Physics, November 2022

#include <iostream>
#include <cmath>
#include <TH1D.h>
#include <TFile.h>
#include <TRandom3.h>

using namespace std;

// Define the function yN
double y(int N, TRandom3 ran){
    double rSum = 0.;
    for(int i=0; i<N; i++){
        rSum += ran.Rndm();
    }
    return sqrt(12./N)*(rSum - N/2.);
}

int main(){

// Open output file (apparently needs to be done before booking)
    TFile file("simpleMC.root", "recreate");

// Book histograms
    TH1D h_y1("h_y1", "N = 1", 100, -4., 4.);
    TH1D h_y2("h_y2", "N = 2", 100, -4., 4.);
    TH1D h_y4("h_y4", "N = 4", 100, -4., 4.);
    TH1D h_y12("h_y12", "N = 12", 100, -4., 4.);
    TH1D h_trans("h_trans", "x", 100, 0., 1.);
    TH1D h_accrej("h_accrej", "x", 100, 0., 1.);

```

```

// Create a TRandom3 object to generate random numbers

int seed = 12345;
TRandom3 ran(seed);

// Generate values and fill histograms

const int numValues = 10000;
for (int i=0; i<numValues; ++i){
    double y1 = y(1, ran);
    double y2 = y(2, ran);
    double y4 = y(4, ran);
    double y12 = y(12, ran);
    h_y1.Fill(y1);
    h_y2.Fill(y2);
    h_y4.Fill(y4);
    h_y12.Fill(y12);
}

// Generate random values according to  $f(x) = 3x^2$  ( $0 < x < 1$ )
// with transformation method; here  $x = r^{(1./3.)}$ .

for (int i=0; i<numValues; i++){
    double r = ran.Rndm();
    double x = pow(r, 1./3.);
    h_trans.Fill(x);
}

// and again using acceptance rejection method

const double xMin = 0.;
const double xMax = 1.;
const double fMax = 3.;
int numAccepted = 0;
while ( numAccepted < numValues ){
    double x = (xMax - xMin)*ran.Rndm() + xMin;
    double u = fMax * ran.Rndm();
    double f = 3.*pow(x,2);
    if ( u < f ){
        numAccepted++;
        h_accrej.Fill(x);
    }
}

// Store all histograms in the output file and close up

file.Write();
file.Close();

return 0;
}

```

Example of MLE

Consider $f(x) = (1+\theta)x^\theta$, $0 \leq x \leq 1$

with i.i.d. sample $x_1, \dots, x_n$

$$L(\theta) = \prod_{i=1}^n f(x_i; \theta) = \prod_{i=1}^n (1+\theta)x_i^\theta$$

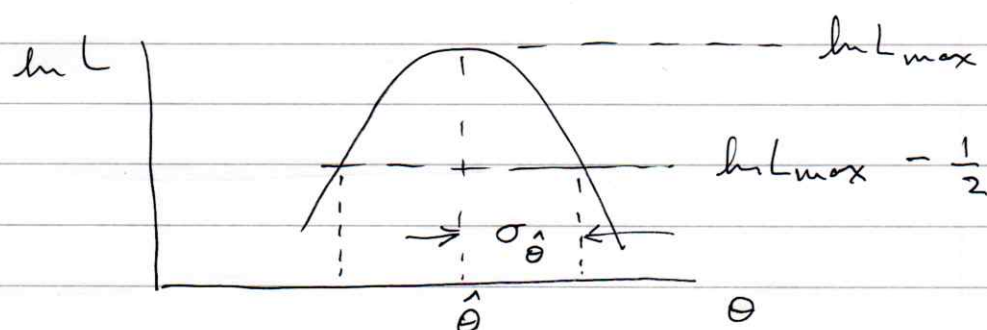
$$\Rightarrow \ln L(\theta) = \sum_{i=1}^n [\ln(1+\theta) + \theta \ln x_i]$$

To find MLE,

$$\frac{\partial \ln L}{\partial \theta} = \frac{n}{1+\theta} + \sum_{i=1}^n \ln x_i \stackrel{\text{set}}{=} 0$$

$$\Rightarrow \hat{\theta} = -1 - \frac{n}{\sum_{i=1}^n \ln x_i}$$

For variance (graphical method)



$\sigma_{\hat{\theta}}$ found by moving θ away from $\hat{\theta}$ until $\ln L$ decreases by $\frac{1}{2}$ from $\ln L_{\max}$.

Variance from asymptotic properties

Assume large n ,

$$V[\hat{\theta}] \approx \text{MVB} = - \frac{\left(1 + \frac{\partial b}{\partial \theta}\right)^2}{E\left[\frac{\partial^2 \ln L}{\partial \theta^2}\right]} \xrightarrow{n \rightarrow \infty} \approx 0$$

$$\frac{\partial^2 \ln L}{\partial \theta^2} = - \frac{n}{(1+\theta)^2} \quad \text{or indep. of data } x_i$$

$$- E\left[\frac{\partial^2 \ln L}{\partial \theta^2}\right] = \frac{n}{(1+\theta)^2}$$

$$\Rightarrow V[\hat{\theta}] \approx \frac{(1+\theta)^2}{n}$$

$$\sigma_{\hat{\theta}} \approx \frac{1+\theta}{\sqrt{n}}$$

MLE Example (cont.)

Now suppose problem is parametrized

using $\lambda = \exp\left[\frac{-1}{1+\theta}\right]$ find MLE of λ .

Use $\hat{\lambda} = \lambda(\hat{\theta})$

$$\Rightarrow \hat{\lambda} = \exp\left[-\frac{1}{1 + \left(-1 - \frac{n}{\sum_{i=1}^n \ln x_i}\right)}\right]$$

$$= \exp\left[\frac{1}{n} \sum_{i=1}^n \ln x_i\right] = \exp\left[\sum_{i=1}^n \ln x_i \cdot \frac{1}{n}\right]$$

$$= \prod_{i=1}^n x_i^{\frac{1}{n}}$$

or the hard way: $\theta = -1 - \frac{1}{\ln \lambda}$

$$f(x; \lambda) = -\frac{1}{\ln \lambda} x^{(-1 - \frac{1}{\ln \lambda})}$$

$$L(\lambda) = \prod_{i=1}^n f(x_i; \lambda)$$

$\rightarrow \ln \lambda \rightarrow \frac{\partial \ln L}{\partial \lambda} = 0 \rightarrow$ same $\hat{\lambda}$ as above.