University of Zurich - HS 2013

Computational Science I
Exercise notes: Markov and Newton-Cotes

Tobias Grubenmann

October 27, 2013

Exercise 1

The simulation of lighthouse flashes runs 500 times with $a = -0,5$ and $b = 0.7$. Only the values for x between -1 and 1 are stored into a list, all others are ignored.

After the simulation, the Metropolis algorithm runs with 1000 Iterations trying to recover a and b. For this purpose the joint probability for random values for a and b is calculated over all x. The new parameters a and b are accepted if the quotient between the new and the old joint probability is greater than a random value from a uniform $[0, 1]$ distribution:

```
import random
import pylab

def run():
    a = -0.5
    b = 0.7

    x = []

    for i in range(500):
        phi = random.random() * 2 * pi
        value = a + b * tan(phi)

        if value >= -1 and value <= 1:
            x += [value]

    u = [random.random()*2 - 1]
```

```
    v = [random.random()]

    for i in range(1000):
        u += [random.random()*2 - 1]
        v += [random.random()]
        r = random.random()

        jointDensity1 = reduce(lambda s, n: s + log(p(x[n],
            u[-1], v[-1])), range(len(x)), 0)
        jointDensity2 = reduce(lambda s, n: s + log(p(x[n],
            u[-2], v[-2])), range(len(x)), 0)

        if exp(jointDensity1 - jointDensity2) <= r:
            u[-1] = u[-2]
            v[-1] = v[-2]

    subplot(311)
    hist(x, bins=40, histtype='step', normed=True)
    subplot(312)
    hist(u, bins=40, histtype='step', normed=True)
    subplot(313)
    hist(v, bins=40, histtype='step', normed=True)

def p(x, a, b):

    return (1/(1 + ((x-a)/b)**2)*1/b) / (arctan((1-a)/b) -
       arctan((-1-a)/b))
```

The histogram of random flashes and of the parameters a and b are as following:

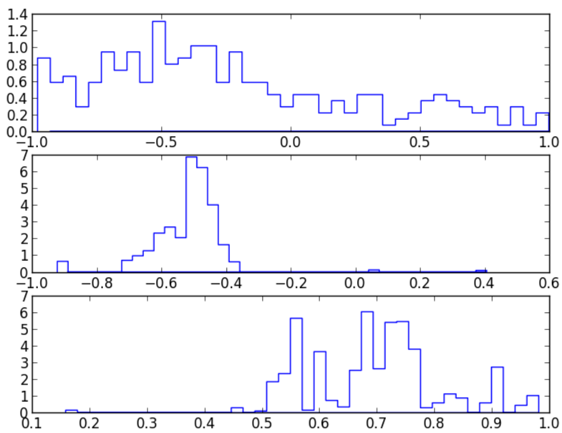

Figure 1: Lighthouse with 500 flashes, $a = -0.5$, $b = 0.7$

Exercise 1

By the symmetry of the integration rule the polynomials x,x^3 and x^5 are always integrated correctly. Therefore, we only have to consider the polynomials 1, x^2 and x^4. Now,

$$\begin{aligned}
\int_{-5}^{5} 1dx &= [x]_{-5}^{5} = 2 \cdot 5 \\
\int_{-5}^{5} x^2 dx &= \left[\frac{1}{3}x^3\right]_{-5}^{5} = 2 \cdot \frac{1}{3}5^3 \\
\int_{-5}^{5} x^4 dx &= \left[\frac{1}{5}x^5\right]_{-5}^{5} = 2 \cdot \frac{1}{5}5^5
\end{aligned}$$

With this values, we can build a system of linear equations:

$$\begin{pmatrix} \frac{1}{2} & 1 & 1 \\ 0 & 2^2 & 4^2 \\ 0 & 2^4 & 4^4 \end{pmatrix} \begin{pmatrix} c_0 \\ c_2 \\ c_4 \end{pmatrix} = \begin{pmatrix} 5 \\ \frac{1}{3}5^3 \\ \frac{1}{5}5^5 \end{pmatrix}$$

The following Python script solves the equation above and then derives the coefficients as fractions:

```
from scipy.linalg import solve

def getCoefficients():

    A = [[1./2, 1, 1],[0, 2**2, 4**2],[0, 2**4, 4**4]]

    B = [[5], [1./3*(5**3)], [1./5*(5**5)]]

    X = solve(matrix(A), matrix(B))

    for i in range(3):

        coeff = []

        while X[i] < pow(10,8):

            coeff += [floor(X[i])]
```

```
            X[i] = 1./(X[i]-floor(X[i]))

        numerator = coeff[-1]
        denominator = 1

        for j in range(len(coeff)-1):

            temp = denominator
            denominator = numerator
            numerator = temp
            numerator = numerator + coeff[-(j+2)]*
               denominator

        c = gcd(numerator[0], denominator[0])

        print numerator[0]/c, "/", denominator[0]/c

def gcd(a, b):
# returns the greatest common divisor of a and b
    if b == 0 :
        return a
    r = a%b
    return gcd(b, r)
```

The solution for the coefficients is:

$$
\begin{aligned}
c_0 &= \frac{335}{96} \\
c_2 &= \frac{125}{144} \\
c_4 &= \frac{1375}{576}
\end{aligned}
$$

Exercise 3

The function `fivepoint( f , a, b, B = 1)` approximates the integral of f between a and b using B blocks using the five point rule derived from the last exercise:

```
def fivepoint( f , a, b, B = 1) :
# Compute integral of f between a and b
# using B blocks of the 5-point integrator.

    integral = 0
```

```
    blockLength = (b - a)/(B*1.)

    for i in range(B):

        c = a + i * blockLength
        d = a + (i+1) * blockLength

        integral += (d-c)/10.*(1375./576. * f((d-c)/10.*-4.
             + (c+d)/2.) + 125/144. * f((d-c)/10.*-2. + (c+d
            )/2.) + 335/96. * f((c+d)/2) + 125./144. * f((d-
            c)/10.*2. + (c+d)/2.) + 1375./576. * f((d-c)
            /10.*4. + (c+d)/2.))

    return integral
```

To calculate $\int\limits_{-\infty}^{+\infty} e^{-x^2} dx$ I use the following transform of variables:

$$\begin{aligned} x &= \frac{t}{1-t^2} \\ \frac{dx}{dt} &= \frac{(1-t^2)+2t^2}{(1-t^2)^2} = \frac{1+t^2}{(1-t^2)^2} \end{aligned}$$

We can now rewrite the integral:

$$\int\limits_{-\infty}^{+\infty} e^{-x^2} dx = \int\limits_{-1}^{1} e^{-\left(\frac{t}{1-t^2}\right)^2} \frac{1+t^2}{(1-t^2)^2} dt$$

Using the above `fivepoint` function the integral evaluates to $1.7724538509059757 \approx \sqrt{\pi}$.