In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import cauchy
from scipy.stats import norm
import seaborn as sn
import korali
from scipy.optimize import fsolve
from IPython.display import display, HTML
display(HTML("<style>.container { width:85% !important; }</style>"))
import random
```

AM207 Homework 4 - Bayesian inference

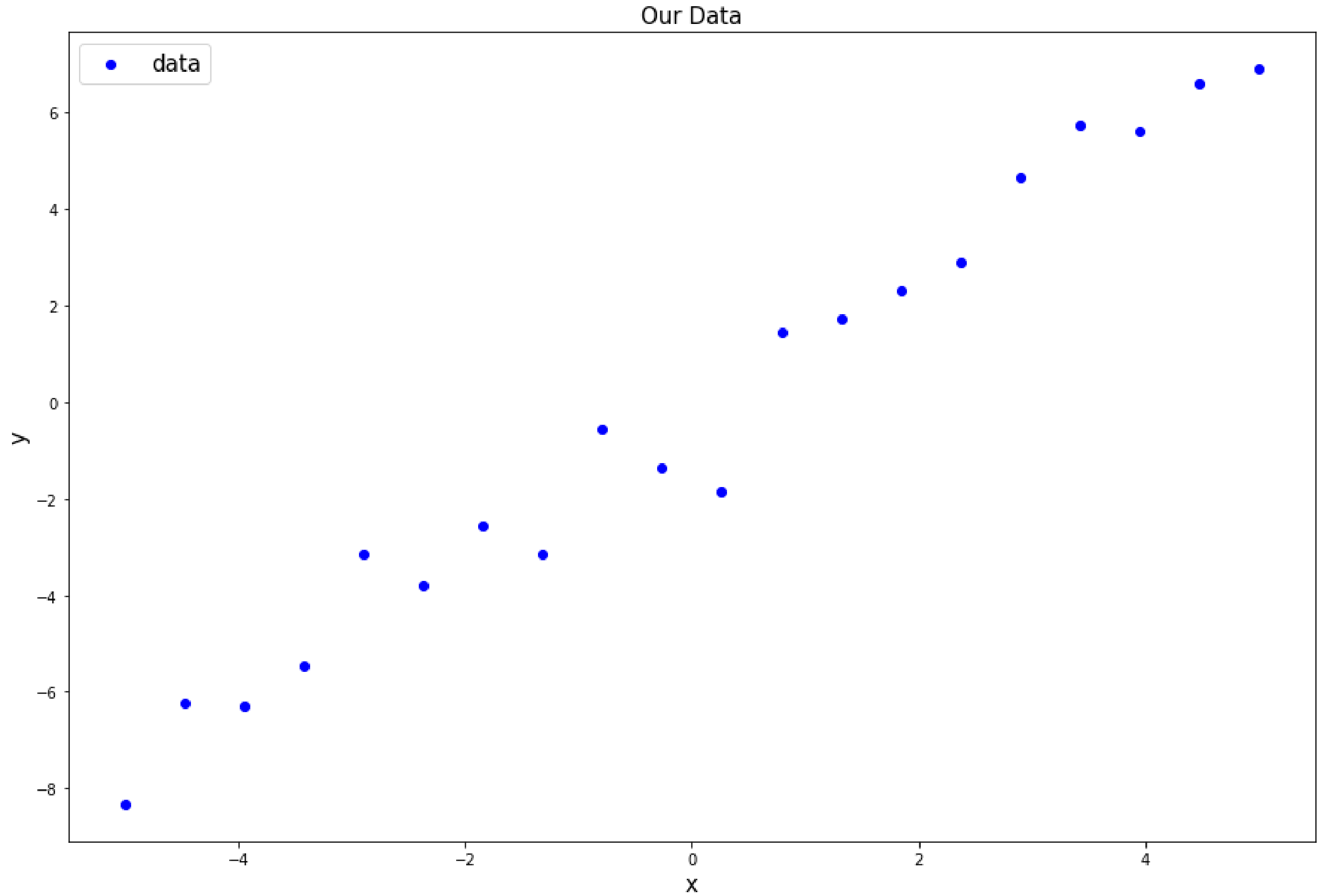
Author : Elie Attias

Date: 11/17/2022

Question 1: Bayesian inference of linear model

An experimentalist collected data $\{x_i, y_i\}_{i=1}^N$. A model of this data is given by $y_i = ax_i + \sigma\xi_i$ where a and σ are scalars and ξ_i are i.i.d. normally distributed random variables, $\xi_i \sim \mathcal{N}(0, 1)$. We denote our parameters as $\theta = [a, \sigma]^T \in \mathbb{R}^2$. We define $d_i := y_i - ax_i = \xi_i$ which is a random variable following $d_i \sim \mathcal{N}(0, \sigma^2)$. Indeed, $(d_i)_{1\leq i \leq N}$ are independent as for the random variables ξ_i.

In [2]:
```
df_1 = pd.read_csv('./1_laplace/data.csv')
X = df_1["x"].to_numpy()
Y = df_1["y"].to_numpy()
fig = plt.figure(figsize = (15,10))

plt.scatter(X,Y, c = 'b', label = 'data')
plt.xlabel('x', fontsize = 15)
plt.ylabel('y', fontsize = 15)
plt.title('Our Data', fontsize = 15)
plt.legend(loc = 'upper left', fontsize = 15)
plt.show()
```



a)

Let $i \in \{1, \cdots, N\}$. We have that the likelihood of each d_i given parameters $\theta = (a, \sigma)$ is :

$$p(d_i \mid \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - ax_i)^2}{2\sigma^2}\right)$$

Thus the likelihood of $d_1, \cdots, d_n$ is the product of each likelihood $p(d_i \mid \theta)$ since we assume each response d_i to be independent. Hence, we have that the likelihood of the data is :

$$p(D \mid \theta) = \prod_{i=1}^{N} p(d_i \mid \theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - ax_i)^2}{2\sigma^2}\right)$$

where we define by D our data $(\{x_i\}_{1 \leq i \leq N}, \{y_i\}_{1 \leq i \leq N})$.

Finally, we have that the log-likelihood of the data is :

$$
\begin{aligned}
\mathcal{L} &= \log(p(D \mid \theta)) \\
&= \log\left(\prod_{i=1}^{N} p(d_i \mid \theta)\right) \\
&= \sum_{i=1}^{N} \log[p(d_i \mid \theta)] \\
&= \sum_{i=1}^{N} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - ax_i)^2}{2\sigma^2}\right)\right] \\
&= N \log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right] - \sum_{i=1}^{N} \frac{(y_i - ax_i)^2}{2\sigma^2} \\
&= -\frac{N}{2}\log(2\pi\sigma^2) - \sum_{i=1}^{N} \frac{(y_i - ax_i)^2}{2\sigma^2} \\
&= -\frac{N}{2}\log(2\pi) - N\log(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - ax_i)^2
\end{aligned} \tag{1}
$$

Let us now find the parameters $\theta_0 = (a_0, \sigma_0)$ that maximise the log-likelihood. To do so, let's differentiate the log likelihood function with respect to (a, σ) and find the roots of the loglikelihood's derivative.

$$
\begin{aligned}
0 = \frac{\partial \mathcal{L}}{\partial a} \,|_{\theta_0} &= -\frac{1}{2\sigma_0^2}\sum_{i=1}^{N}(2a_0x_i^2 - 2y_ix_i) \implies a_0 = \frac{\sum_{i=1}^{N} x_iy_i}{\sum_{i=1}^{N} x_i^2} \\
0 = \frac{\partial \mathcal{L}}{\partial \sigma} \,|_{\theta_0} &= -\frac{N}{\sigma_0} + \frac{1}{\sigma_0^3}\sum_{i=1}^{N}(y_i - ax_i)^2 \\
\implies \sigma_0^2 &= \frac{1}{N}\sum_{i=1}^{N}(y_i - a_0x_i)^2 \text{ since } \sigma_0 \neq 0 \\
\implies \sigma_0^2 &= \frac{1}{N}\sum_{i=1}^{N}\left(y_i - x_i\frac{\sum_{j=1}^{N} x_jy_j}{\sum_{j=1}^{N} x_j^2)}\right)^2
\end{aligned} \tag{2}
$$

In order to check whether or not the likelihood is maximal at (a_0, σ_0), we must check that the Hessian of the log likelihood is negative definite.

$$
\begin{aligned}
\frac{\partial^2 \mathcal{L}}{\partial a^2}|_{\theta_0} &= -\frac{1}{\sigma_0^2}\sum_{i=1}^{N} x_i^2 \leq 0 \\
\frac{\partial^2 \mathcal{L}}{\partial \sigma^2}|_{\theta_0} &= \frac{N}{\sigma_0^2} - \frac{3}{\sigma_0^4}\sum_{i=1}^{N}(y_i - a_0x_i)^2 = \frac{N}{\sigma_0^2} - \frac{3N}{\sigma_0^2} = \frac{-2N}{\sigma_0^2} \leq 0 \\
\frac{\partial^2 \mathcal{L}}{\partial a \partial \sigma}|_{\theta_0} &= \frac{-2}{\sigma_0^3}\sum_{i=1}^{N}(y_ix_i - a_0x_i^2) = 0 \\
\frac{\partial^2 \mathcal{L}}{\partial \sigma \partial a}|_{\theta_0} &= \frac{2}{\sigma_0^3}\sum_{i=1}^{N}(a_0x_i^2 - y_ix_i) = 0
\end{aligned} \tag{3}
$$

Hence the hessian $\mathbf{H}$ of the log-likelihood evaluated at θ_0 is diagonal :

$$\mathbf{H}_{\theta_0} = \begin{pmatrix} \frac{\partial^2 \mathcal{L}}{\partial a^2} & \frac{\partial^2 \mathcal{L}}{\partial a \partial \sigma} \\ \frac{\partial^2 \mathcal{L}}{\partial \sigma \partial a} & \frac{\partial^2 \mathcal{L}}{\partial \sigma^2} \end{pmatrix} (\theta_0) = \begin{pmatrix} \frac{\partial^2 \mathcal{L}}{\partial a^2} & 0 \\ 0 & \frac{\partial^2 \mathcal{L}}{\partial \sigma^2} \end{pmatrix} (\theta_0)$$

and negative definite as we have that for all $\theta^T = [a, \sigma]^T \in \mathbb{R}^2$:

$$\theta^T \mathbf{H}_{\theta_0} \theta = a^2 \frac{\partial^2 \mathcal{L}}{\partial a^2}(a_0, \sigma_0) + b^2 \frac{\partial^2 \mathcal{L}}{\partial \sigma^2}(a_0, \sigma_0) \leq 0$$

Hence, we have that that the loglikelihood and the likelihood are maximal at the point $\theta_0 = (a_0, \sigma_0)$.

In [3]:
```
N = len(X)
a_0 = np.sum(X*Y)/np.sum(X**2)
sigma_0 = np.sqrt((1/N)*np.sum((Y - a_0*X)**2))
theta_0 = np.array([a_0, sigma_0])
print(f'The point at which the likelihood and log-likelihood function is maximal is : (a
```

```
The point at which the likelihood and log-likelihood function is maximal is : (a,sigma)
= (1.4864344460664443, 0.7795182956841904).
```

b)

Let us now express the posterior distribution of the parameters. We assume uniform priors with large enough bounds for a and σ, respectively a_1, a_2 and σ_1, σ_2. We define $I := [a_1, a_2] \times [\sigma_1, \sigma_2] \subset \mathbb{R}^2$. As our parameters follow a uniform distribution, their pointwise density is a constant which we denote as $p(\theta) := \frac{1}{C} \in \mathbb{R}$.

By Baye's Theorem, we have that :

$$\begin{aligned} p(\theta \mid D) &= \frac{p(D \mid \theta) p(\theta)}{p(D)} \\ &= \frac{p(D \mid \theta)}{C p(D)} \\ &= \frac{p(D \mid \theta)}{C \int_I p(D \mid \theta) p(\theta) d\theta} \\ &= \frac{p(D \mid \theta)}{C \int_I p(D \mid \theta) \frac{1}{C} d\theta} \\ &= \frac{p(D \mid \theta)}{\int_I p(D \mid \theta) d\theta} \\ &\propto p(D \mid \theta) \text{ since } \int_I p(D \mid \theta) d\theta \text{ is a constant} \end{aligned} \tag{4}$$

The log-Laplace approximation for our log posterior distribution around the equilibria $\theta_0 = (\sigma_0, a_0)^T$ is:

$$
\begin{aligned}
\log p(\theta \mid D) &\propto \mathcal{L}(D \mid \theta) \\
&\approx \mathcal{L}(D \mid \theta_0) + \nabla_\theta \mathcal{L}(D \mid \theta_0)^T(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^T \mathbf{H}_{\theta_0}(\theta - \theta_0) \\
&= \mathcal{L}(D \mid \theta_0) + \frac{1}{2}(\theta - \theta_0)^T \mathbf{H}_{\theta_0}(\theta - \theta_0) \text{ since the log-likelihood is maximal at } \theta_0 \\
&= \mathcal{L}(D \mid \theta_0) + \frac{\partial^2 \mathcal{L}(D \mid \theta)}{\partial a^2}|_{\theta_0}(a - a_0)^2 + \frac{\partial^2 \mathcal{L}(D \mid \theta)}{\partial \sigma^2}|_{\theta_0}(\sigma - \sigma_0)^2
\end{aligned}
$$

c)

We have that our target distribution is $p(\theta \mid D)$. We form the following proposal distributions from two independent Cauchy distributions with median at the previous sample value and with a scale parameter $\gamma : Q_1(\sigma_1, \sigma_0, \gamma) = \frac{1}{\pi\gamma}\left[\frac{\gamma^2}{(\sigma_1 - \sigma_0)^2 + \gamma^2}\right], Q_2(a_1, a_0, \gamma) = \frac{1}{\pi\gamma}\left[\frac{\gamma^2}{(a_1 - a_0)^2 + \gamma^2}\right]$. We note that since the proposal cauchy distributions are symmetric, the acceptance rate becomes: $p(\theta_{proposed} \mid D)/p(\theta_{current} \mid D)$

In [4]:
```
def likelihood(theta):
    'returns likelihood of the data D = (X, Y) given parameters theta'
    a = theta[0]
    sigma = theta[1]
    coef = 1/((2*np.pi*(sigma**2))**(N/2))
    S = np.sum((Y - a*X)**2)
    S_exp  = np.exp(-S/(2*sigma**2))
    return coef*S_exp
```

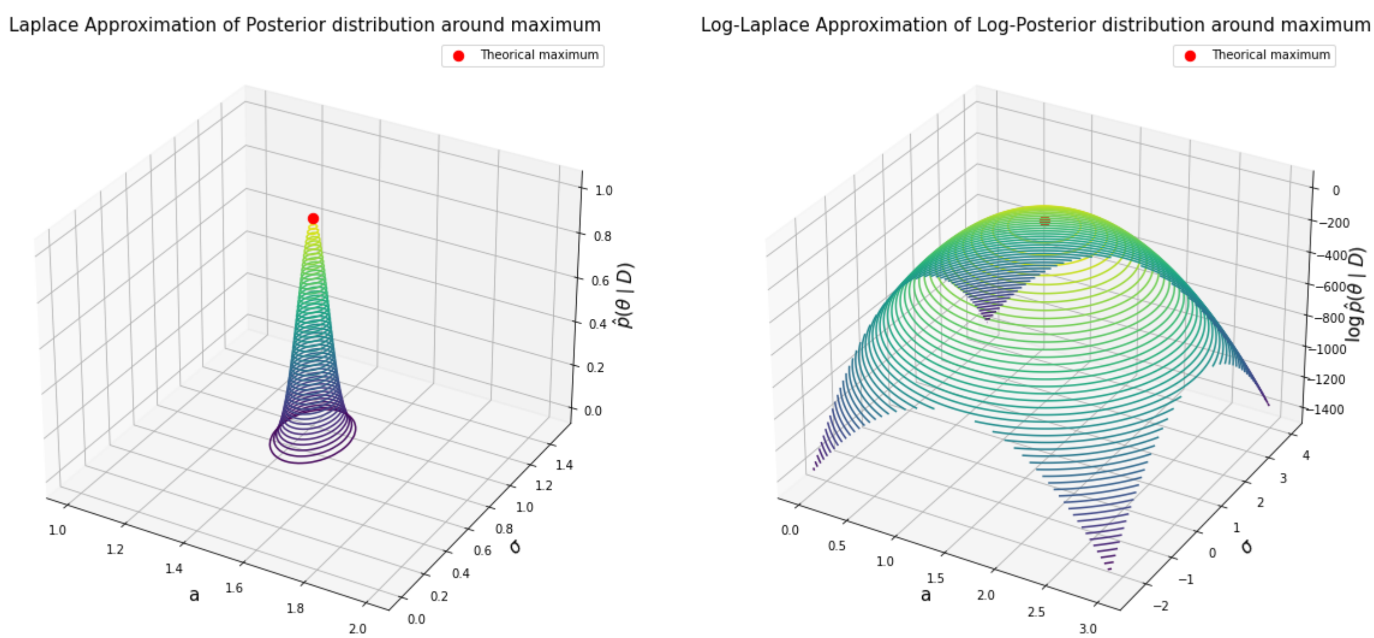
In [5]:
```
###first derivatives
partial_a_theta_0 = np.sum(X*Y - a_0*X**2) ## this is 0
partial_s_theta_0 = (-N/sigma_0 + np.sum((Y - a_0*X)**2)/(sigma_0**3)) ## this is 0
###double derivatives
partial_aa_theta_0 = -1/(sigma_0**2)*np.sum(X**2)
partial_ss_theta_0 = -2*N/sigma_0**2
print(f'log likelihood double derivative with respect to a at theta_0 is : {partial_aa_t
```

```
log likelihood double derivative with respect to a at theta_0 is : -303.15296336969317
log likelihood double derivative with respect to sigma at theta_0 is : -65.8275006174190
8
```

In [6]:
```
def posterior(theta):
    'returns posterior up to the normalisation constant'
    return likelihood(theta)

def log_posterior(theta):
    'returns the log of the exact posterior'
    return np.log(posterior(theta))

def posterior_log_laplace(theta):
    'returns the log-laplace approximation of log-posterior density'
    a = theta[0]
    sigma = theta[1]
    return  likelihood(theta_0) + partial_aa_theta_0 * (a - a_0)**2 + partial_ss_theta_0

def posterior_laplace(theta):
    'returns the exponential of the log laplace approxmation of the log posterior'
    return np.exp(posterior_log_laplace(theta))
```

In [7]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, subplot_kw=dict(projection='3d'), figsize = (20,20)
x = np.linspace(1, 2, 100)
y = np.linspace(0, 1.5, 100)
```

```
X_val, Y_val = np.meshgrid(x, y)
Z = posterior_laplace([X_val, Y_val])

ax1.contour3D(X_val, Y_val, Z, 50, cmap='viridis') #cmap = binary, viridis
ax1.scatter([a_0],[sigma_0], [posterior_laplace(theta_0)], s = 70, c = 'r', label = 'The
ax1.legend()
ax1.set_xlabel('a', fontsize = 15)
ax1.set_ylabel('$\sigma$', fontsize = 15)
ax1.set_zlabel(r'$\hat{p}(\theta \mid D$)', fontsize = 15);
ax1.set_title('Laplace Approximation of Posterior distribution around maximum', fontsize


x = np.linspace(0., 3, 100)
y = np.linspace(-2.5, 4, 100)

X_val, Y_val = np.meshgrid(x, y)
Z = posterior_log_laplace([X_val, Y_val])

ax2.contour3D(X_val, Y_val, Z, 50, cmap='viridis') #cmap = binary, viridis
ax2.scatter([a_0],[sigma_0], [posterior_log_laplace(theta_0)], s = 70, c = 'r', label =
ax2.legend()
ax2.set_xlabel('a', fontsize = 15)
ax2.set_ylabel('$\sigma$', fontsize = 15)
ax2.set_zlabel(r'$\log \hat{p}(\theta \mid D$)', fontsize = 15);
ax2.set_title('Log-Laplace Approximation of Log-Posterior distribution around maximum',
plt.show()
```



where we denote in the z axis $\hat{p}$ as the laplace approximation of the posterior.

In [8]:

```
def MH(max_iter, gamma, theta):
    '''x represents a; y represents sigma ; this function implements Metropolis Hasting
       to sample parameters (a, sigma) from the log posterior density'''

    t, x, y = 0, theta[0], theta[1]
    Sx, Sy = [], []
    p = log_posterior(np.array([x, y]))
    accept, reject, dx, dy = 0, 0, 0, 0
    Ra, Rs = [], []

    while True:
        Sx.append(x)
        Sy.append(y)
        t += 1
```

```python
        if t>=max_iter:
            break
        dx = cauchy.rvs(scale = gamma)
        dy = cauchy.rvs(scale = gamma)

        xp, yp = x + dx, y + dy
        thetap = np.array([xp,yp])
        pp = log_posterior(thetap)
        u = np.random.uniform(low=0., high=1.0, size = 1)[0]
        if (pp > p or pp >= np.log(u) + p) and  y > 0 and yp > 0: # y,yp > 0 to enforce
            x, y = xp, yp
            p = pp
            accept += 1

        else:
            Ra.append(xp)
            Rs.append(yp)
            reject += 1
    print(f"Rejection rate for \u03B3 = {gamma} is :", reject/(accept + reject))
    return np.array(Sx), np.array(Sy), np.array(Ra), np.array(Rs), reject/(accept + reje
```

In [9]:

```python
gamma = 0.1
max_iter= 1000
theta_start = [0.5, 1]
print('We chose gamma in order to have a rejection rate close to 0.7: \n')
Sx1, Sy1, Ra1, Rs1, R1 = MH(max_iter, gamma,theta_start)
```

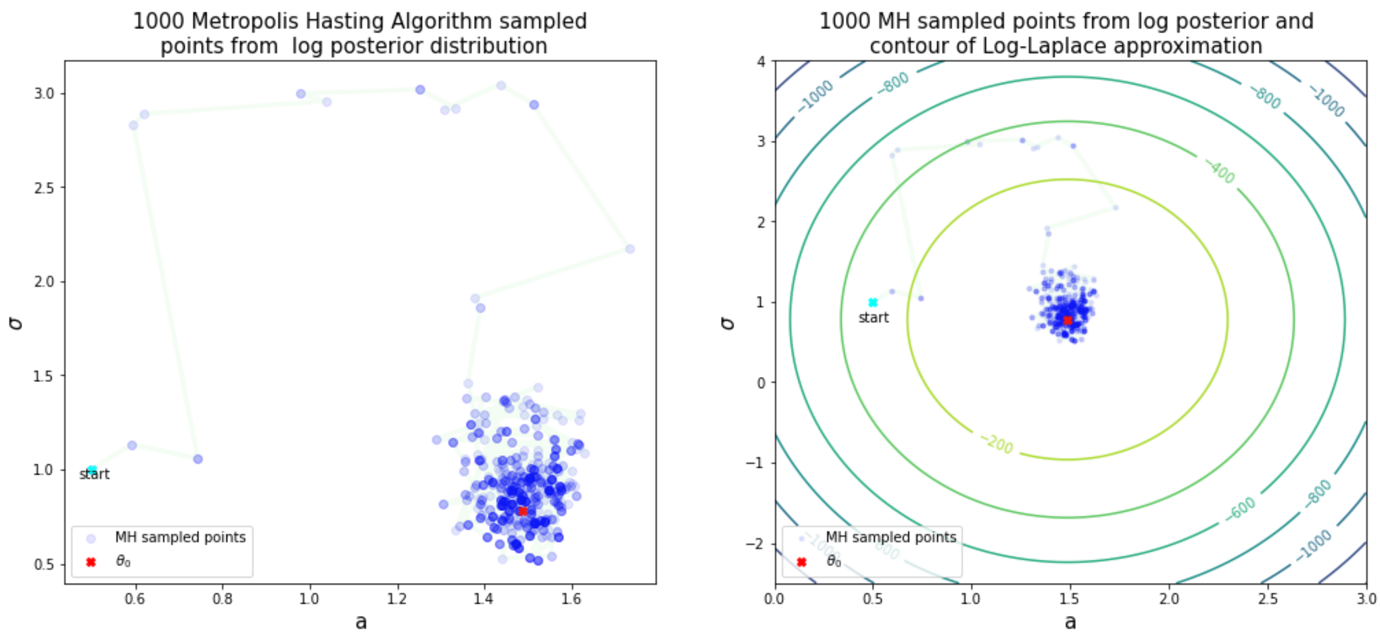
```
We chose gamma in order to have a rejection rate close to 0.7:

Rejection rate for γ = 0.1 is : 0.6996996996996997
```

```
/Users/elieattias/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7: Runtime
Warning: divide by zero encountered in log
  import sys
```

In [10]:

```python
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.set_figheight(7)
fig.set_figwidth(17)

ax1.plot(Sx1, Sy1, alpha=0.1, c = 'lightgreen', lw=3)
ax1.scatter(Sx1, Sy1, alpha=0.1, c = 'b', label = 'MH sampled points')
ax1.scatter([a_0], [sigma_0], c = 'r', label = r'$\theta_0$', marker = 'X')
ax1.scatter([theta_start[0]], [theta_start[1]], c = 'cyan', marker = 'X')
ax1.text(x = theta_start[0 ] - 0.03, y = theta_start[1] - 0.05, s = 'start')
ax1.legend(loc = 'lower left',fontsize = 10)
ax1.set_xlabel('a', fontsize = 15)
ax1.set_ylabel(f'$\sigma$' , fontsize = 15)
ax1.set_title(f' 1000 Metropolis Hasting Algorithm sampled \npoints from  log posterior

## contour plot
x = np.arange(np.min(Sx1) - 0.5, np.max(Sx1) + 0.5, 0.1)
y = np.arange(np.min(Sy1) - 0.5, np.max(Sy1) + 0.5, 0.1)

x = np.linspace(0., 3, 100)
y = np.linspace(-2.5, 4, 100)

x_log = np.linspace(0., 3, 100)
y_log =  np.linspace(-2.5, 4, 100)

X_plot, Y_plot = np.meshgrid(x, y)
X_plot, Y_plot = np.meshgrid(x_log, y_log)
Z_plot = np.array([[posterior_log_laplace(np.array([X_plot[i,j],Y_plot[i,j]])) for j in

CS = plt.contour(X_plot, Y_plot, Z_plot)
ax2.plot(Sx1, Sy1, alpha=0.1, c = 'lightgreen', lw=3)
ax2.scatter(Sx1, Sy1, alpha=0.1, c = 'b', label = 'MH sampled points', s = 10)
```

```
ax2.scatter([a_0], [sigma_0], c = 'r', label = r'$\theta_0$', marker = 'X' )

ax2.scatter([theta_start[0]], [theta_start[1]], c = 'cyan', marker = 'X')
ax2.text(x = theta_start[0 ] - 0.08, y = theta_start[1] - 0.25, s = 'start')
ax2.clabel(CS, inline=1, fontsize=10)
ax2.legend(loc='lower left', fontsize = 10)
ax2.set_xlabel('a', fontsize=15)
ax2.set_ylabel('$\sigma$', fontsize=15)
ax2.set_title('1000 MH sampled points from log posterior and \ncontour of Log-Laplace ap

plt.show()
```



d)

Now we have $K = 1000$ samples points $\{a_k, \sigma_k\}$. For each sample pair (a_k, σ_k), we can hence make k predictions on y given x. A prediction y given a sample item $\theta_k = (a_k, \sigma_k)$ follows a normal distribution i.e :$y \mid x \sim \mathcal{N}(a_k x, \sigma_k^2)$ where $k \in \{1, \cdots, 1000\}$.

Hence, for any x, y, we have 1000 predictions for $p(y \mid D, x)$. We approximate such prediction by the mean :

$$p(y \mid D; x) \approx \frac{1}{K} \sum_{k=1}^{K} \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(y - a_k x)^2}{2\sigma_k^2}\right)$$

We then approxmiate the cdf by the cumulative sum with respect to y:

$$C(y) = \sum_{i=min(Y)}^{y} p(i \mid D; x)$$

Finally, to find the quantile y_q, we compute the following:

$$y_q = \operatorname{argmin}_y \left(|C(y) - q|\right)$$

These are shown for $x = -4$ and $x = 3$ in the examples here-under.

In [11]:
```
L = 50
K = 1000
```

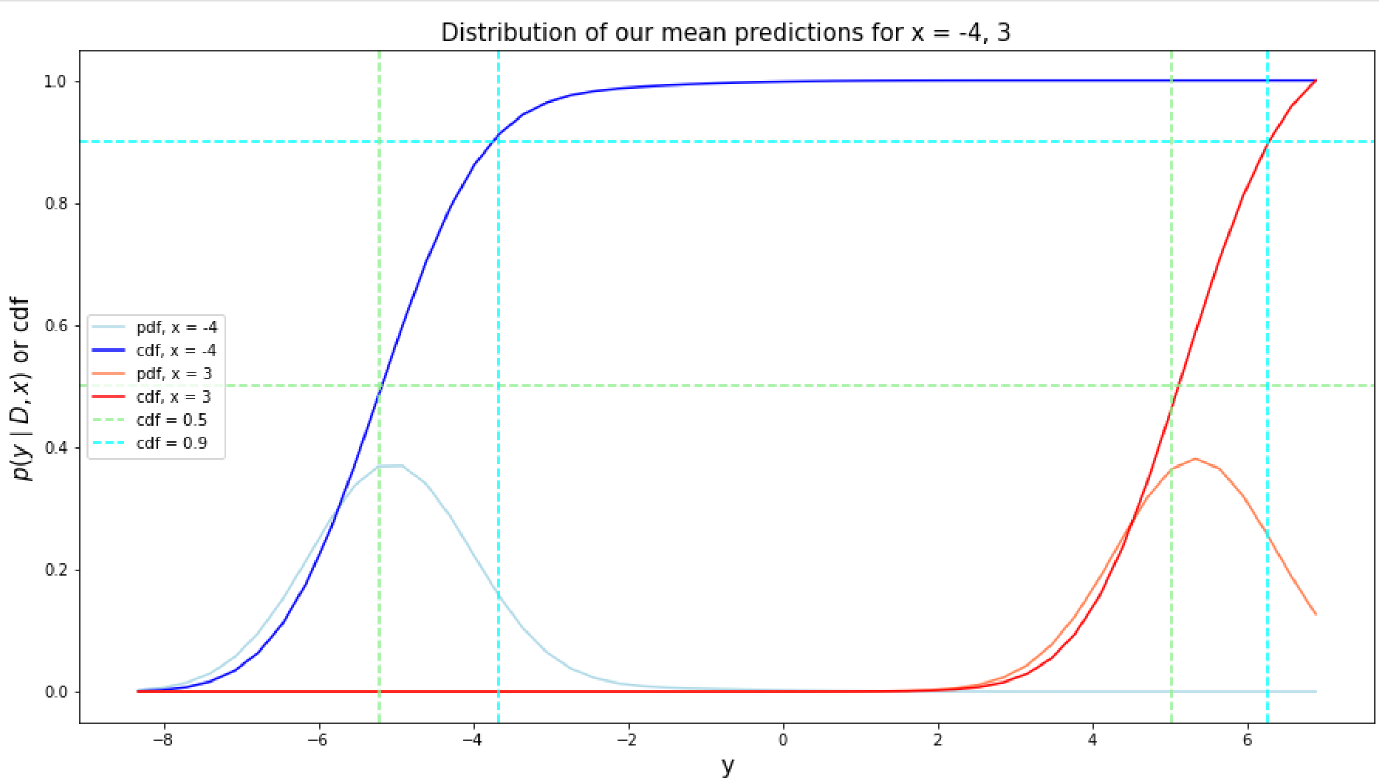
```
y_val = np.linspace(np.min(Y), np.max(Y), L)
x_val = np.linspace(np.min(X), np.max(X), L)
```

In [12]:

```
x1 = -4
pdfs_2 = np.array([[norm.pdf(y_val[k] -  x1* Sx1[i], Sy1[i]) for k in range(L)] for i in
pdf_mean_2 = np.mean(pdfs_2, axis = 0)
cdf_2 = np.cumsum(pdf_mean_2)
cdf_2 = cdf_2/np.max(cdf_2)
q1 = y_val[np.argmin(np.abs(cdf_2 - 0.5))]
q2 = y_val[np.argmin(np.abs(cdf_2 - 0.9))]

x2 = 3
pdfs_3 = np.array([[norm.pdf(y_val[k] -  x2* Sx1[i], Sy1[i]) for k in range(L)] for i in
pdf_mean_3 = np.mean(pdfs_3, axis = 0)
cdf_3 = np.cumsum(pdf_mean_3)
cdf_3 = cdf_3/np.max(cdf_3)
q3 = y_val[np.argmin(np.abs(cdf_3 - 0.5))]
q4 = y_val[np.argmin(np.abs(cdf_3 - 0.9))]
```

In [13]:

```
fig = plt.figure(figsize = (15,8))
plt.plot(y_val, pdf_mean_2, c ='lightblue', label= f'pdf, x = {x1}')
plt.plot(y_val, cdf_2/np.max(cdf_2), c = 'b', label = f'cdf, x = {x1}')

plt.plot(y_val, pdf_mean_3, c ='coral', label= f'pdf, x = {x2}')
plt.plot(y_val, cdf_3/np.max(cdf_3), c = 'r', label = f'cdf, x = {x2}')
plt.axhline(0.5, c = 'lightgreen', linestyle = 'dashed', label = 'cdf = 0.5')
plt.axhline(0.9, c = 'cyan', linestyle = 'dashed', label = 'cdf = 0.9')
plt.axvline(q1, c = 'lightgreen', linestyle = 'dashed')
plt.axvline(q2, c = 'cyan', linestyle = 'dashed')
plt.axvline(q3, c = 'lightgreen', linestyle = 'dashed')
plt.axvline(q4,  c = 'cyan', linestyle = 'dashed')
plt.xlabel('y', fontsize = 15, c = 'black')
plt.ylabel(r'$p(y\mid D, x)$ or cdf', fontsize = 15, c = 'black')
plt.title(f'Distribution of our mean predictions for x = {x1}, {x2}', fontsize = 15, c =
plt.legend(loc = 'center left')
plt.show()
```



In [15]:

```
quantiles_50 = np.zeros(L)
quantiles_90 = np.zeros(L)
```

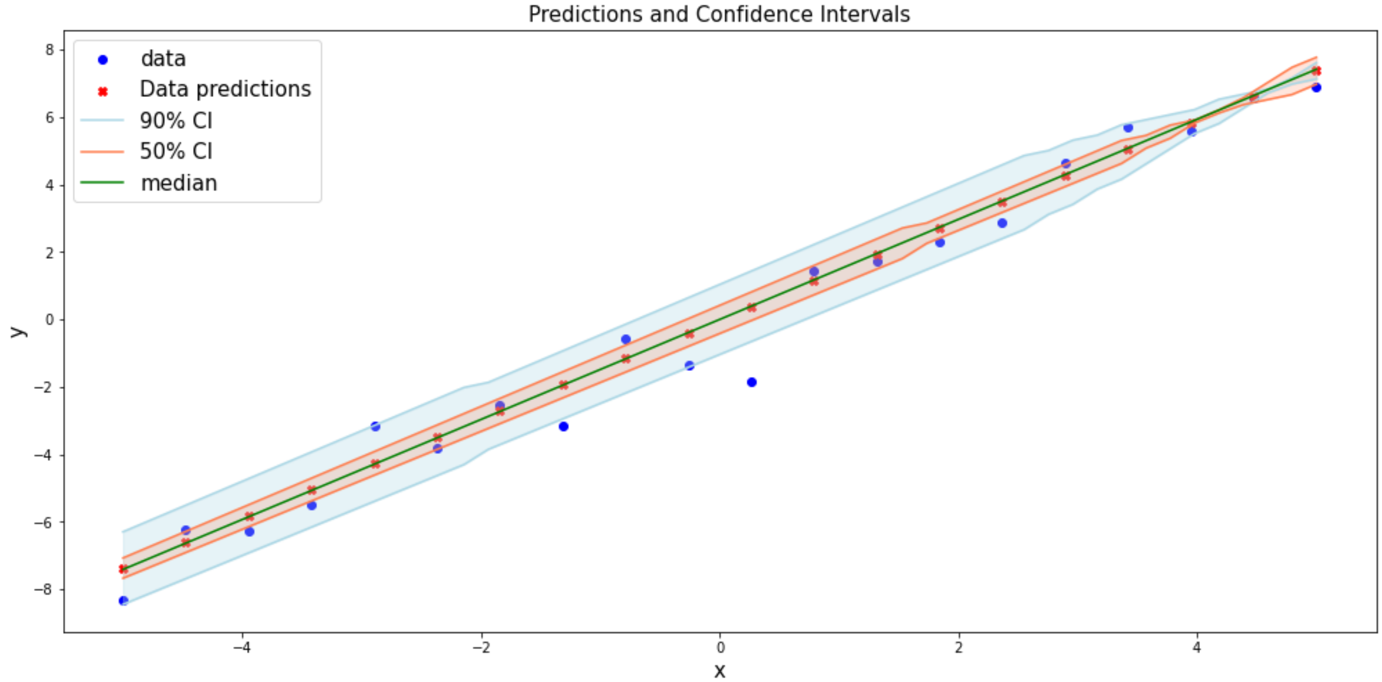
```
medians = np.zeros(L)

for c, x in enumerate(x_val):
    predict_y = x*Sx1
    predict_y.sort()
    median = predict_y[K//2] #since K = 1000 is even

    pdfs = np.array([[norm.pdf(y_val[l] - x * Sx1[k], Sy1[k]) for l in range(L)] for k i
    pdf_mean = np.mean(pdfs, axis = 0)
    cdf = np.cumsum(pdf_mean)
    cdf = cdf/np.max(cdf)
    y_50 = y_val[np.argmin(np.abs(cdf - 0.5))] #we can take the argmin of the abs becaus
    y_90 = y_val[np.argmin(np.abs(cdf - 0.9))]

    medians[c] = median
    quantiles_50[c] = y_50
    quantiles_90[c] = y_90
```

In [16]:

```
predictions = x_val*np.mean(Sx1)
data_predictions = X*np.mean(Sx1)
r_50 = np.abs(predictions - quantiles_50)
r_90 = np.abs(predictions - quantiles_90)
y_low_50 = predictions - r_50/2
y_high_50 = predictions + r_50/2
y_low_90 = predictions - r_90/2
y_high_90 = predictions + r_90/2
```

In [17]:

```
fig = plt.figure(figsize = (17,8))

plt.scatter(X, Y, c = 'b', label = 'data')
plt.scatter(X, data_predictions, c = 'r', label = 'Data predictions', marker = 'X')

plt.fill_between(x_val, y_low_90, y_high_90, color = 'lightblue', alpha = 0.3)
plt.fill_between(x_val, y_low_50, y_high_50, color = 'coral', alpha = 0.2)

plt.plot(x_val, y_low_90, c = 'lightblue', label = '90% CI')
plt.plot(x_val, y_high_90, c = 'lightblue')

plt.plot(x_val, y_low_50, c = 'coral', label = '')
plt.plot(x_val, y_high_50, c = 'coral',label = '50% CI')

plt.plot(x_val, medians, c = 'g' , label = 'median')
plt.legend(loc = 'upper left', fontsize = 15)
plt.xlabel('x', fontsize= 15)
plt.ylabel('y', fontsize= 15)
plt.title('Predictions and Confidence Intervals', fontsize= 15)
plt.show()
```



Question 2: Where is the beacon?

A beacon positioned at unknown location (a, b) emits light in random directions $\theta \sim \mathcal{U}(-\pi/2, \pi/2)$. The light is detected by sensors located on the shore $(y = 0)$. The light detections are recorded at coordinates x_i, $i = 1, 2, \ldots, N$.

a)

We define g such that $x = g(\theta) := a - b\tan\theta$. We are given that the density of θ is uniform such that $p_\theta(\theta_i) = \dfrac{1}{\pi}$. We note that g and all our random variables X, θ are continuous. In addition, g is a bijective continuous function and is hence invertible. Its inverse is : $g^{-1}(x) = \arctan\left(\frac{a-x}{b}\right)$. We have by transformation of continuous random variables that the likelihood of each x given θ and parameters $\omega = (a, b)$ is :

$$\begin{aligned} p_X(x \mid \omega) &= p_\theta(g^{-1}(x) \mid \omega)\left|\frac{d}{dx}g^{-1}(x)\right| \\ &= \frac{1}{\pi}\left|\frac{d}{dx}\arctan\left(\frac{a-x}{b}\right)\right| \\ &= \frac{1}{\pi}\left|\frac{1}{1+\frac{(a-x)^2}{b^2}} \times \left(\frac{-1}{b}\right)\right| \\ &= \frac{1}{\pi}\frac{b}{b^2+(a-x)^2} \end{aligned} \tag{6}$$

which is a Cauchy distribution.

Thus the likelihood of $x_1, \cdots, x_n$ is the product of each likelihood $p(x_i \mid \omega)$ since we assume each response x_i to be independent. Hence, we have that the likelihood of the data is :

$$P(X \mid \omega) = \prod_{i=1}^{N} p_X(x_i \mid \omega) = \prod_{i=1}^{N} \frac{1}{\pi} \frac{b}{b^2 + (a - x_i)^2}$$

where X denotes $(x_i)_{1 \leq i \leq N}$.

Hence, the log likelihood is :

$$\mathcal{L}(X \mid \omega) = \log[P(X \mid \omega)] = \sum_{i=1}^{N} \log\left(\frac{1}{\pi} \frac{b}{b^2 + (a - x_i)^2}\right)$$

b)

Let us express the posterior distribution of $\omega = (a, b)$ where both a and b are unknown. We assume that the priors are independent -i.e that $P(\omega \mid X) = P(a \mid X)P(b \mid X)$. Hence the posterior distribiution for ω is:

$$\begin{aligned} p(\omega \mid X) &= \frac{P(X \mid \omega)P(\omega)}{P(X)} \\ &= \frac{P(X \mid \omega)P(a \mid X)P(b \mid X)}{P(X)} \\ &\propto P(X \mid \omega)P(a \mid X)P(b \mid X) \end{aligned} \tag{7}$$

where we assume that our priors are such that $a \mid X \sim \mathcal{U}(-20, 20)$, and $b \mid X \sim \mathcal{U}(0, 20)$. Indeed, considering that the impact points are mostly distributed around 0, it makes sense to take a prior for a symmetric around 0. Then, as understood by the problem set, the beacon's y coordinate (b) is positive.

c)

Using Korali, we sampled the posterior distribution of $\omega = (a, b)$ given the data X. The mean of the parameters a, b is $a = 14.76$ and $b = 4.52$. The posterior density of the beacon's position is shown here-under.

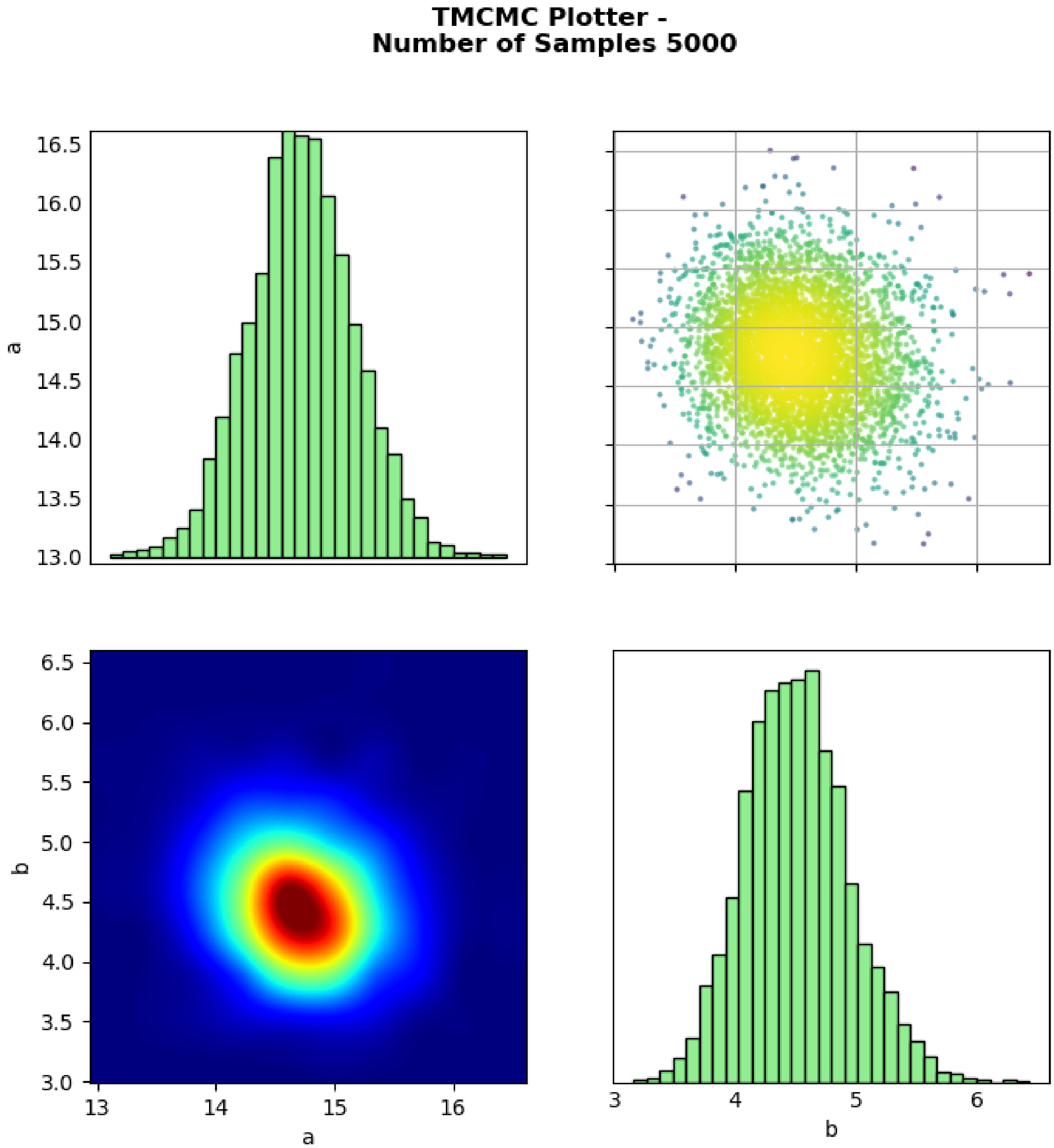
In [20]:

```
df = pd.read_csv('2_lighthouse/data.csv')
x = df["x"].to_numpy()
n = len(x)

def log_likelihood(ks):
    a,b =  ks["Parameters"]
    log_like = np.sum(np.log(p_x(a,b)))
    ks["logLikelihood"] = log_like

def p_x(a,b):
    'pointwise likelihood'
    return (1/np.pi) * (b/(b**2 + (a - x)**2))

e = korali.Experiment()

e["Problem"]["Type"] = "Bayesian/Custom"
e["Problem"]["Likelihood Model"] = log_likelihood

e["Solver"]["Type"] = "Sampler/TMCMC"
e["Solver"]["Population Size"] = 5000
e["Solver"]["Target Coefficient Of Variation"] = 1.0
e["Solver"]["Covariance Scaling"] = 0.2
```

```python
e["Distributions"] = [
    {"Name": "Prior a",
        "Type": "Univariate/Uniform",
        "Minimum": -20,
        "Maximum": 20},

    {"Name": "Prior b",
        "Type": "Univariate/Uniform",
        "Minimum": 0,
        "Maximum": 20}
]


e["Variables"] = [
    {"Name": "a", "Prior Distribution": "Prior a"},
    {"Name": "b", "Prior Distribution": "Prior b"}
]

e["Store Sample Information"] = True

e["Console Output"]["Verbosity"] = "Detailed"
k = korali.Engine()
k.run(e)
```

```
[Korali] --------------------------------------------------------------------
[Korali] Current Generation: #1
[Korali] Annealing Exponent:          0.000e+00.
[Korali] Acceptance Rate (proposals / selections): (100.00% / 48.16%)
[Korali] Coefficient of Variation: 100.00%
[Korali] log of accumulated evidence: -8.969
[Korali] max logLikelihood: -804.029
[Korali] Number of finite Evaluations (prior / likelihood): (5000 / 5000)
[Korali] Sample Mean:
 a = +1.268e+01
 b = +8.889e+00
[Korali] Sample Covariance:
   | +5.336e+00       -        |
   | -1.383e+00  +4.396e+00    |
[Korali] Experiment: 0 - Generation Time: 4.548s
[Korali] --------------------------------------------------------------------
[Korali] Current Generation: #2
[Korali] Annealing Exponent:          9.453e-03.
[Korali] Acceptance Rate (proposals / selections): (79.70% / 48.10%)
[Korali] Coefficient of Variation: 100.00%
[Korali] log of accumulated evidence: -29.977
[Korali] max logLikelihood: -804.030
[Korali] Number of finite Evaluations (prior / likelihood): (4511 / 4511)
[Korali] Sample Mean:
 a = +1.465e+01
 b = +6.131e+00
[Korali] Sample Covariance:
   | +1.037e+00       -        |
   | -1.353e-01  +1.375e+00    |
[Korali] Experiment: 0 - Generation Time: 4.604s
[Korali] --------------------------------------------------------------------
[Korali] Current Generation: #3
[Korali] Annealing Exponent:          3.395e-02.
[Korali] Acceptance Rate (proposals / selections): (81.66% / 47.10%)
[Korali] Coefficient of Variation: 100.00%
[Korali] log of accumulated evidence: -84.031
[Korali] max logLikelihood: -804.030
[Korali] Number of finite Evaluations (prior / likelihood): (4892 / 4892)
[Korali] Sample Mean:
 a = +1.477e+01
 b = +4.932e+00
```

```
[Korali] Sample Covariance:
   | +2.862e-01       -       |
   | -1.859e-02  +2.877e-01   |
[Korali] Experiment: 0 - Generation Time: 4.598s
[Korali] --------------------------------------------------------------------
[Korali] Current Generation: #4
[Korali] Annealing Exponent:          9.965e-02.
[Korali] Acceptance Rate (proposals / selections): (82.42% / 47.52%)
[Korali] Coefficient of Variation: 100.00%
[Korali] log of accumulated evidence: -264.372
[Korali] max logLikelihood: -804.030
[Korali] Number of finite Evaluations (prior / likelihood): (4998 / 4998)
[Korali] Sample Mean:
 a = +1.477e+01
 b = +4.590e+00
[Korali] Sample Covariance:
   | +7.844e-02       -       |
   | -5.536e-03  +7.250e-02   |
[Korali] Experiment: 0 - Generation Time: 4.626s
[Korali] --------------------------------------------------------------------
[Korali] Current Generation: #5
[Korali] Annealing Exponent:          3.224e-01.
[Korali] Acceptance Rate (proposals / selections): (82.88% / 48.88%)
[Korali] Coefficient of Variation: 106.63%
[Korali] log of accumulated evidence: -810.277
[Korali] max logLikelihood: -804.027
[Korali] Number of finite Evaluations (prior / likelihood): (5000 / 5000)
[Korali] Sample Mean:
 a = +1.477e+01
 b = +4.497e+00
[Korali] Sample Covariance:
   | +2.472e-02       -       |
   | -9.305e-04  +2.203e-02   |
[Korali] Experiment: 0 - Generation Time: 4.672s
[Korali] --------------------------------------------------------------------
[Korali] Current Generation: #6
[Korali] Annealing Exponent:          1.000e+00.
[Korali] Acceptance Rate (proposals / selections): (83.06% / 62.90%)
[Korali] Coefficient of Variation: 200.00%
[Korali] log of accumulated evidence: -810.277
[Korali] max logLikelihood: -804.027
[Korali] Number of finite Evaluations (prior / likelihood): (5000 / 5000)
[Korali] Sample Mean:
 a = +1.476e+01
 b = +4.520e+00
[Korali] Sample Covariance:
   | +4.214e-02       -       |
   | -2.790e-03  +3.844e-02   |
[Korali] Experiment: 0 - Generation Time: 4.750s
[Korali] --------------------------------------------------------------------
[Korali] sampler/TMCMC finished correctly.
[Korali] Termination Criterion Met: TMCMC['Target Annealing Exponent'] = 1.000000.
[Korali] Final Generation: 6
[Korali] Elapsed Time: 28.102s
```

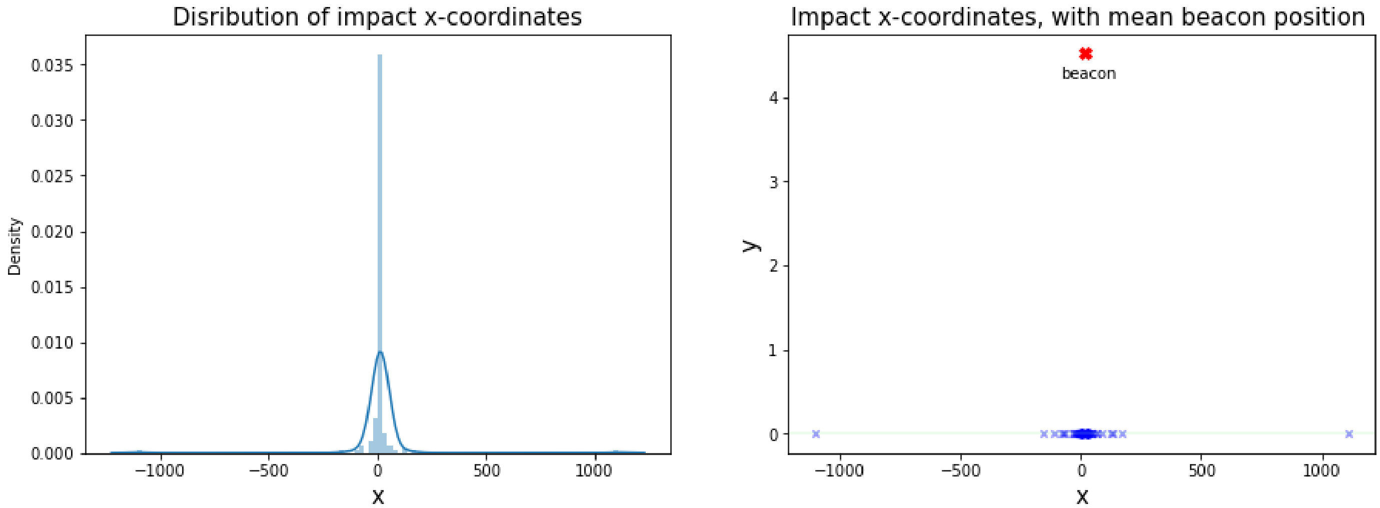


In [21]:

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (15, 5))
df = pd.read_csv('./2_lighthouse/data.csv')
x = df["x"].to_numpy()

sn.distplot(x,bins=100, ax = ax1)
ax2.scatter(x,[0]*len(x), c = 'b',alpha = 0.4, s = 20, marker = 'x')

ax1.set_xlabel('x', fontsize = 15)
ax2.set_xlabel('x', fontsize = 15)
ax2.set_ylabel('y', fontsize = 15)

a = 14.76
b = 4.52
ax2.scatter([a], [b], c = 'r', s = 60, marker = 'X')
ax2.axhline(xmin = -1000, xmax =1000, y = 0, alpha = 0.2, c = 'lightgreen')
ax2.text(x = a-100, y = b - 0.3,s = 'beacon')

ax1.set_title('Disribution of impact x-coordinates', fontsize = 15)
ax2.set_title('Impact x-coordinates, with mean beacon position ', fontsize = 15)
plt.show()
```

```
/Users/elieattias/anaconda3/lib/python3.7/site-packages/seaborn/distributions.py:2619: F
```

```
utureWarning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
In [ ]:
```