1. Rosenbrock 函数（香蕉函数）

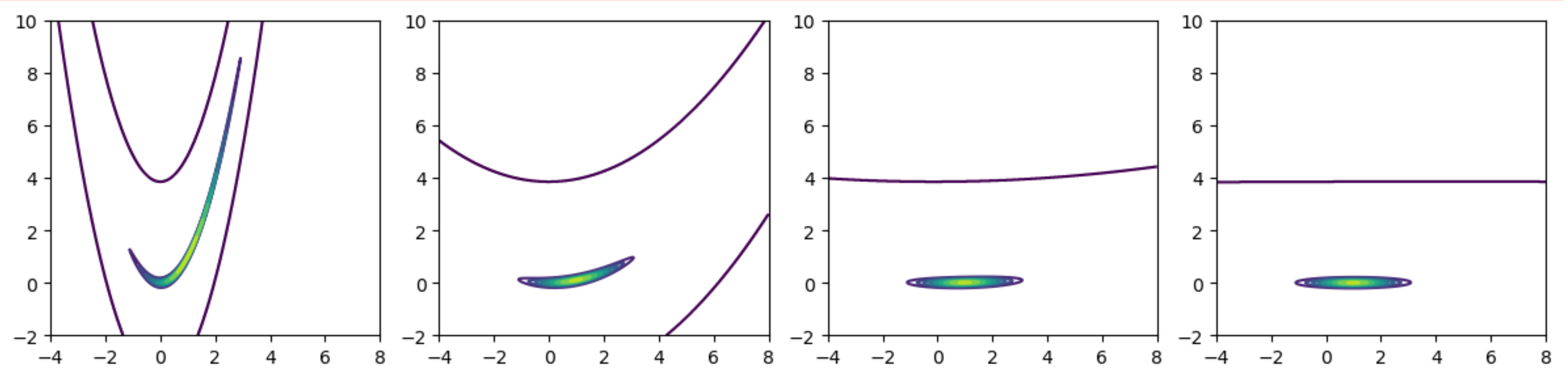
我们要采样的后验分布满足

$$
\begin{aligned}
\rho_{\text{post}}(\theta) &= e^{-\Phi_R(\theta)} \\
\Phi_R &= \frac{1}{2}\Big((y-\mathcal{G}(\theta))^T\Sigma_\eta^{-1}(y-\mathcal{G}(\theta)) + (\theta-r_0)^T\Sigma_0^{-1}(\theta-r_0)\Big) \\
&= \frac{1}{2}\Big(\frac{100(\theta_2-c_1\theta_1^2)^2}{c_2} + \frac{(1-\theta_1)^2}{c_2} + \frac{\theta_1^2}{100} + \frac{\theta_2^2}{100}\Big)
\end{aligned}
$$

暴力网格搜索

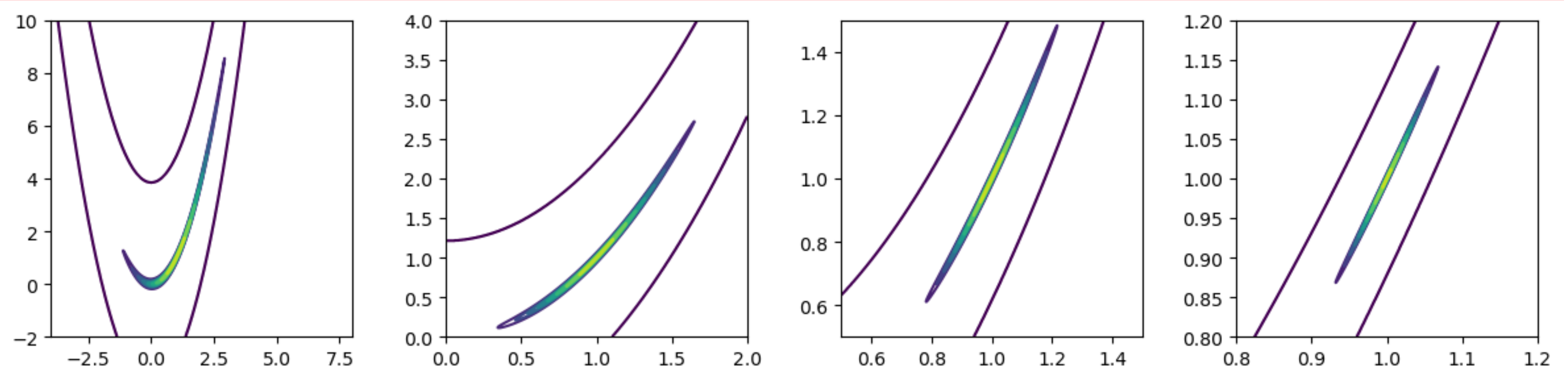
In [3]:
```julia
using PyPlot

function  Phi_R_Rosenbrock(θ₁, θ₂, c₁, c₂)
    return (100*(θ₂ - c₁*θ₁^2)^2/c₂ + (1.0 - θ₁)^2/c₂  + θ₁^2/100 + θ₂^2/100)/2.0
end

Lx , Ux = -4.0,  8.0
Ly , Uy = -2.0,  10#3.5
N = 1000
X = zeros(N, N)
Y = zeros(N, N)
ρ = zeros(N, N)
for i = 1:N
    for j = 1:N
        X[i, j], Y[i, j] = Lx + (Ux - Lx) * (i-1)/(N-1), Ly + (Uy - Ly) * (j-1)/(N-1)
    end
end
Δx = Δy = 1/(N-1)

fig, ax = PyPlot.subplots(ncols=4, nrows=1, sharex=false, sharey=false, figsize=(12,3))

for k = 1:4
    c₁, c₂=10^(-(k-1.0)), 1.0
    for i = 1:N
        for j = 1:N
            ρ[i, j] = Phi_R_Rosenbrock(X[i, j], Y[i, j], c₁, c₂)
        end
    end

    ρ .= exp.(-ρ)
    Z = sum(ρ)
    ρ ./= Z
    ρ ./= (Δx * Δy)
    ax[k].contour(X, Y, ρ, 10)
    @info "mean = ", sum(X.*ρ)/sum(ρ), sum(Y.*ρ)/sum(ρ)
end

fig.tight_layout()
fig.savefig("Rosenbrock_1.pdf")
```

```
[ Info: ("mean = ", 0.9083223721468913, 1.686437532612661)
[ Info: ("mean = ", 0.9893276577929085, 0.19675099271002844)
[ Info: ("mean = ", 0.9900925963862894, 0.019701671046172844)
[ Info: ("mean = ", 0.9901002636870367, 0.001970193755655862)
```



In [4]:
```julia
Lx , Ux = [-4.0;0.0;0.5;0.8] , [8.0;2.0;1.5;1.2]
Ly , Uy = [-2.0;0.0;0.5;0.8] , [10.0;4.0;1.5;1.2]
N = 1000
X = zeros(N, N)
Y = zeros(N, N)
ρ = zeros(N, N)
for i = 1:N
    for j = 1:N

    end
end

fig, ax = PyPlot.subplots(ncols=4, nrows=1, sharex=false, sharey=false, figsize=(12,3))

for k = 1:4
    c₁, c₂= 1.0, 10^(-(k-1.0))
    for i = 1:N
        for j = 1:N
            X[i, j], Y[i, j] = Lx[k] + (Ux[k] - Lx[k]) * (i-1)/(N-1), Ly[k] + (Uy[k] - Ly[k]) * (j-1)/(N-1)
            ρ[i, j] = Phi_R_Rosenbrock(X[i, j], Y[i, j], c₁, c₂)
        end
    end
    ρ .= exp.(-ρ)
    Z = sum(ρ)
    ρ ./= Z
    ρ ./= (Δx * Δy)
    ax[k].contour(X, Y, ρ, 10)
    @info "mean = ", sum(X.*ρ)/sum(ρ), sum(Y.*ρ)/sum(ρ)
end

fig.tight_layout()
fig.savefig("Rosenbrock_2.pdf")
```

```
[ Info: ("mean = ", 0.9083223721468913, 1.686437532612661)
[ Info: ("mean = ", 0.9978348801870758, 1.092302714586285)
[ Info: ("mean = ", 0.997052605773829, 1.0032053198267843)
[ Info: ("mean = ", 0.9998858462934318, 1.0007534734013894)
```



2. Bernstein Von Mises 理论

小噪音情况

对于后验分布，我们有

$$\rho_{\text{post}}(\theta) = e^{-\Phi(\theta)}\rho_{\text{prior}}(\theta) \qquad \Phi(\theta) = \frac{1}{2\gamma^2}(y-\mathcal{G}(\theta))^T\Sigma_\eta^{-1}(y-\mathcal{G}(\theta))$$

假设 $\Phi(\theta)$ 的极小值是 $\theta^\dagger$， 那么当$\gamma \rightarrow 0$

$$\frac{\rho_{\text{post}}(\theta^\dagger)}{\rho_{\text{post}}(\theta)} = e^{\Phi(\theta)-\Phi(\theta^\dagger)}\frac{\rho_{\text{prior}}(\theta^\dagger)}{\rho_{\text{prior}}(\theta)} \rightarrow \infty$$

质量将集中在 $\theta^\dagger$ 附近 $|\theta-\theta^\dagger| < \gamma$，我们可以线性化，

$$\Phi(\theta) \approx \frac{1}{2\gamma^2}(y-\mathcal{G}(\theta^\dagger) - \nabla\mathcal{G}(\theta^\dagger)^T(\theta-\theta^\dagger))\Sigma_\eta^{-1}(y-\mathcal{G}(\theta^\dagger) - \nabla\mathcal{G}(\theta^\dagger)(\theta-\theta^\dagger)) = \Phi(\theta^\dagger) + \frac{1}{2\gamma^2}(\theta-\theta^\dagger)^T\nabla\mathcal{G}(\theta^\dagger)^T)\Sigma_\eta^{-1}\nabla\mathcal{G}(\theta^\dagger)(\theta-\theta^\dagger)$$

转化为线性贝叶斯问题处理。

大量数据情况

数据足够多，也可以转化成误差足够小。对于数据足够多的情况，后验分布满足

$$\rho_{\text{post}}(\theta) = e^{-\sum_{i=1}^N \Phi(\theta;y_i)}\rho_{\text{prior}}(\theta) \qquad \Phi(\theta;y) = \frac{1}{2}(y-\mathcal{G}(\theta))^T\Sigma_\eta^{-1}(y-\mathcal{G}(\theta))$$

假设 $y_i = \mathcal{G}(\theta^*) + \eta_i$

$$\sum_{i=1}^N \Phi(\theta;y_i) = \frac{1}{2}\sum_{i=1}^N(\mathcal{G}(\theta^*)+\eta_i-\mathcal{G}(\theta))^T\Sigma_\eta^{-1}(\mathcal{G}(\theta^*)+\eta_i-\mathcal{G}(\theta))$$

在一些假设下，最大似然估计 $\theta^\dagger \rightarrow \theta^*$, ρ_{post} 质量几乎在最大似然附近, 二阶导数为$\mathcal{O}(N)$. 在它附近展开 $\mathcal{G}(\theta) \approx \mathcal{G}(\theta^*) + \nabla\mathcal{G}(\theta^*)(\theta-\theta^*)$, 我们有近似

$$\sum_{i=1}^N \Phi(\theta;y_i) \approx \frac{N}{2}(\theta-\theta^*)^T\nabla\mathcal{G}(\theta^*)\Sigma_\eta^{-1}\nabla\mathcal{G}(\theta^*)(\theta-\theta^*)$$

更一般情况下，后验分布满足

$$\rho_{\text{post}}(\theta) = e^{\sum_{i=1}^N \log P_\theta(y_i)}\rho_{\text{prior}}(\theta)$$

定义

$$\Phi(\theta) = -\sum_{i=1}^N \log P_\theta(y_i)$$

在一些假设下，最大似然估计 $\theta^\dagger \rightarrow \theta^*$, ρ_{post} 质量几乎在最大似然附近, 二阶导数为

$$\nabla_\theta^2\Phi(\theta^*) = -\sum_{i=1}^N \nabla_\theta^2 \log P_{\theta^*}(y_i) \approx -N\mathbb{E}_{P_{\theta^*}(y)}\Big[\nabla_\theta^2 \log P_{\theta^*}(y)\Big] \approx N\mathbb{E}_{P_{\theta^*}(y)}\Big[\frac{\nabla_\theta P_{\theta^*}(y)\nabla_\theta P_{\theta^*}(y)^T}{P_{\theta^*}(y)^2}\Big]$$

为$\mathcal{O}(N)$. 在它附近展开 $\log P_\theta(y) \approx \log P_{\theta^*}(y) + \nabla\log P_{\theta^*}(y)(\theta-\theta^*) + \frac{1}{2}(\theta-\theta^*)^2\nabla^2\log P_{\theta^*}(y)(\theta-\theta^*)$, 我们有近似

$$
\begin{aligned}
\Phi(\theta) &\approx -N\mathbb{E}_{P_{\theta^*}(y)}\Big[\nabla_\theta \log P_{\theta^*}(y)\Big](\theta-\theta^*) - \frac{N}{2}(\theta-\theta^*)^T\mathbb{E}_{P_{\theta^*}(y)}\Big[\nabla_\theta^2 \log P_{\theta^*}(y)\Big](\theta-\theta^*) \\
&\approx \frac{N}{2}(\theta-\theta^*)^T I(\theta^*)(\theta-\theta^*)
\end{aligned}
$$

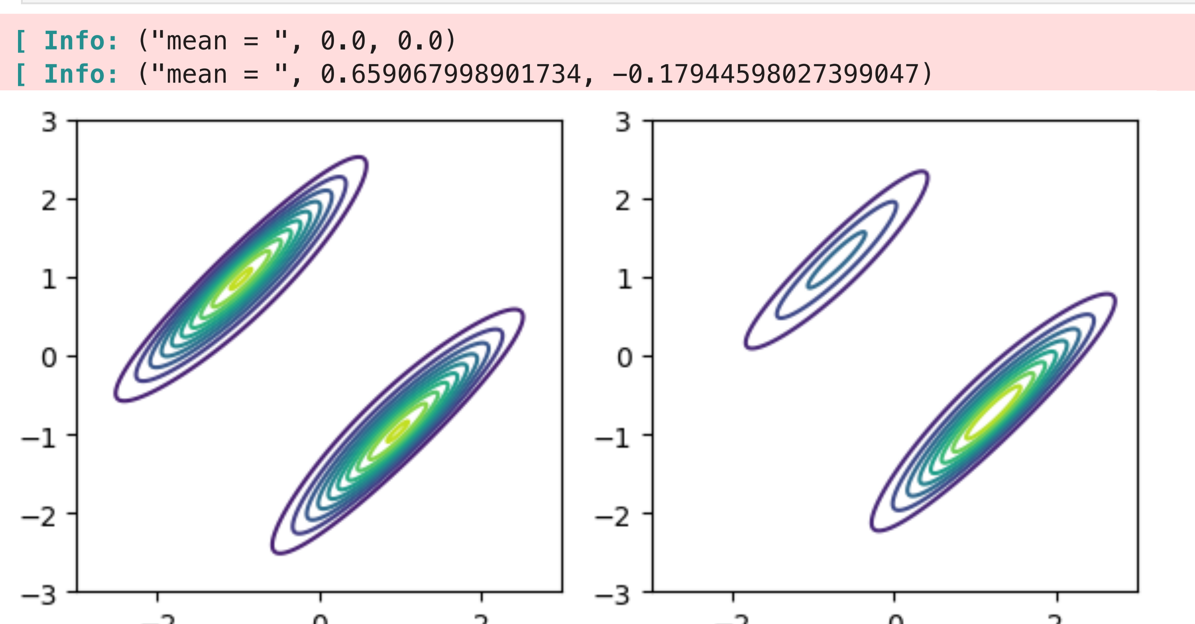
2. 多峰函数

我们要采样的后验分布满足

$$
\begin{aligned}
\rho_{\text{post}}(\theta) &= e^{-\Phi_R(\theta)} \\
\Phi_R &= \frac{1}{2}\Big((y-\mathcal{G}(\theta))^T\Sigma_\eta^{-1}(y-\mathcal{G}(\theta)) + (\theta-r_0)^T\Sigma_0^{-1}(\theta-r_0)\Big) \\
&= \frac{1}{2}\Big(\big(4-(\theta_2-\theta_1)^2\big)^2 + (c-\theta_1)^2 + \theta_2^2\Big)
\end{aligned}
$$

In [5]:
```julia
using PyPlot

using PyPlot

function  Phi_R_Multimodal(θ₁, θ₂, c)
    return ( ( 4 - (θ₁- θ₂)^2 )^2  + (θ₁ - c)^2 + θ₂^2)/2.0
end

Lx , Ux = -3.0,  3.0
Ly , Uy = -3.0,  3.0
N = 1000
X = zeros(N, N)
Y = zeros(N, N)
ρ = zeros(N, N)
for i = 1:N
    for j = 1:N
        X[i, j], Y[i, j] = Lx + (Ux - Lx) * (i-1)/(N-1), Ly + (Uy - Ly) * (j-1)/(N-1)
    end
end

fig, ax = PyPlot.subplots(ncols=2, nrows=1, sharex=false, sharey=false, figsize=(6,3))

for k = 1:2
    c = 0.5*(k-1)
    for i = 1:N
        for j = 1:N
            ρ[i, j] = Phi_R_Multimodal(X[i, j], Y[i, j], c)
        end
    end
    ρ .= exp.(-ρ)
    Z = sum(ρ)
    ρ ./= Z
    ρ ./= (Δx * Δy)
    ax[k].contour(X, Y, ρ, 10)
    @info "mean = ", sum(X.*ρ)/sum(ρ), sum(Y.*ρ)/sum(ρ)
end

fig.tight_layout()
fig.savefig("Multimodal.pdf")
```

```
[ Info: ("mean = ", 0.0, 0.0)
[ Info: ("mean = ", 0.659067998901734, -0.17944598027399047)
```



In [ ]: