# 1. Rosenbrock 函数（香蕉函数）

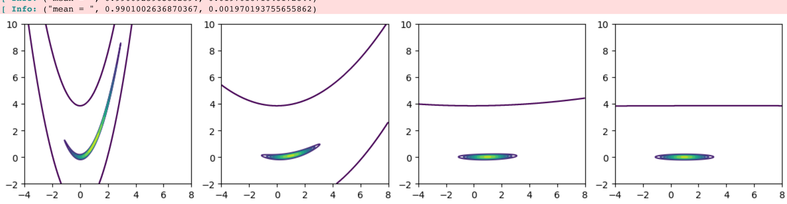
我们要采样的后验分布满足

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

# 暴力网格搜索

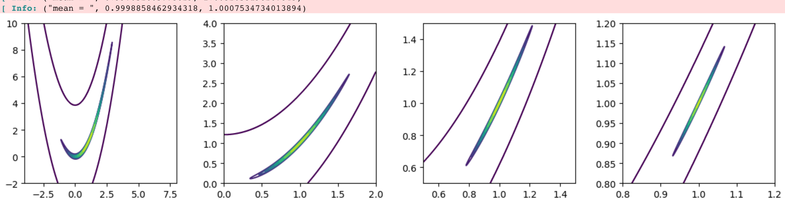
In [3]:
```
using PyPlot

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]:
```
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, 1.0923027145862857)
[ Info: ("mean = ", 0.9970526057738291, 1.0032053198267843)
[ Info: ("mean = ", 0.9998858462934318, 1.0007534734013894)
```



# 2. Bernstein Von Mises 理论

对于后验分布，我们有

$$
\begin{aligned}
\rho_{\text{post}}(\theta) &= e^{-\Phi_R(\theta)} \qquad \Phi_R = \frac{1}{2}\Big( \frac{1}{\gamma^2}\Phi(\theta) + R(\theta) \Big) \\
\Phi(\theta) &= (y-\mathcal{G}(\theta))^T \Sigma_\eta^{-1} (y-\mathcal{G}(\theta)) \qquad R(\theta) = (\theta-r_0)^T \Sigma_0^{-1} (\theta-r_0)
\end{aligned}
$$

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

$$
\frac{\rho_{\text{post}}(\theta^\dagger)}{\rho_{\text{post}}(\theta)} = e^{\frac{1}{\gamma^2}(\Phi(\theta)-\Phi(\theta^\dagger))+R(\theta)-R(\theta^\dagger)} \to 0
$$

质量将集中在 $\theta^\dagger$ 附近，我们可以线性化，转化为线性贝叶斯问题处理。

数据足够多，也可以转化成误差足够小。

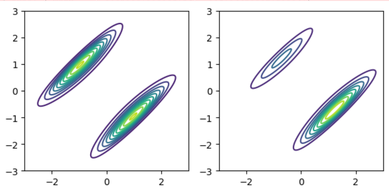
# 2. 多峰函数

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]:
```
using PyPlot

using PyPlot

function  Phi_R_Multimodal(θ₁, θ₂, c)
    return ( ( 4 - (θ₁- θ₁)^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 [ ]:
```

```