Demo_4_Lapis_2019

April 8, 2019

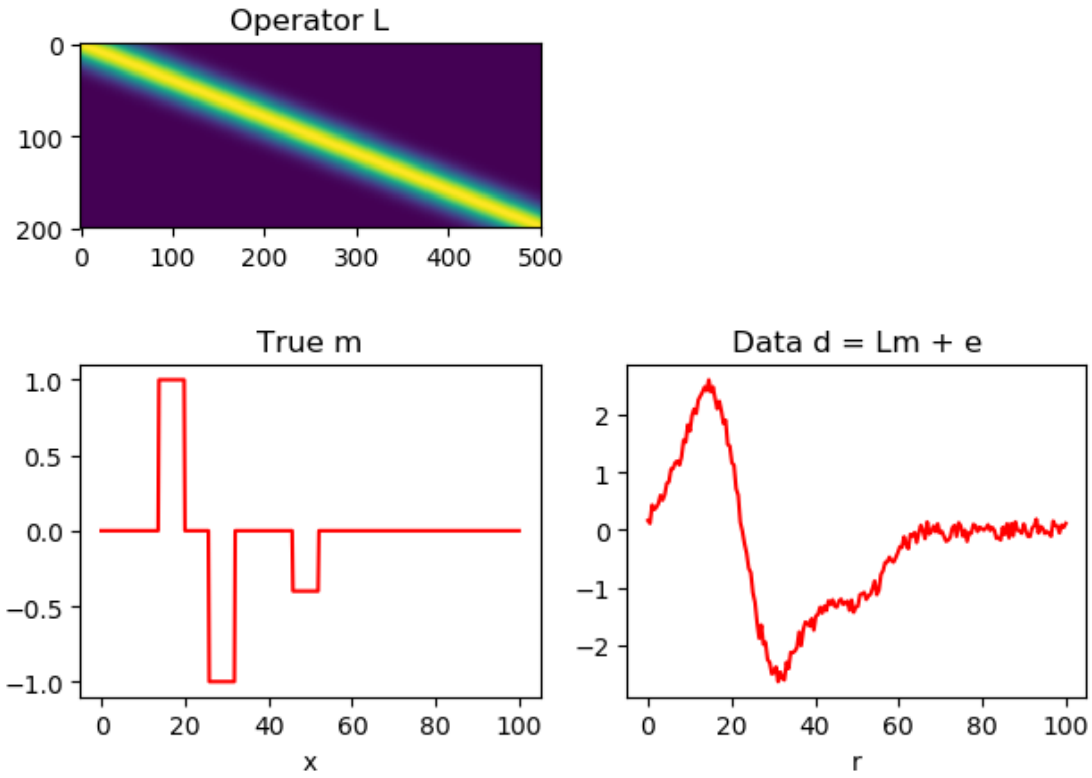
In [3]:
```
using PyPlot, LinearAlgebra
```

1 LAPIS 2019 - UNLP

Use SD to solve sparse inversion problem with Preconding. Rather that operator **W** I will use a low-pass operator **P**

In [4]:
```
# Set forward problem and compute mock (sythetic) data

        N = 200
        M = 500


        # Make a sparse signal

        m = zeros(M)
        m[70:100]= m[70:100] .+1.0
        m[130:160]= m[130:160] .-1.0
        m[230:260]= m[230:260] .-0.4
        x_max = 100.
        x_min =   0.

        r_max = 100.
        r_min =   0.

        alpha = 0.01
        A = .1

        dx = (x_max-x_min)/(M-1)
        dr = (r_max-r_min)/(N-1)

        x = [x_min+dx*(i-1) for i in 1:M]
        r = [r_min+dr*(i-1) for i in 1:N]

        L = A*[exp(-alpha*(( x[i]-r[j] )^2)) for j in 1:N, i in 1:M]

        # Make the synthetic data and add noise to it
```

```
d = L*m + 0.1*randn(N)

subplot(221); imshow(L);      title("Operator L");
subplot(223); plot(x,m,"r"); title("True m"); xlabel("x")
subplot(224); plot(r,d,"r"); title("Data d = Lm + e"); xlabel("r")

tight_layout()
```



In [10]:
```
# Solve via SD

a = 0.01
mu = 0.01

# Low pass-filter
P =  0.25* diagm(-1=> ones(M-1)) + 0.5*diagm(0=> ones(M)) + 0.25*diagm(1=> ones(M-1))


# SD method

m_sol = zeros(M)
u = zeros(M)
```

```
for k = 1:300
    u= u - a*(P'*L'*(L*P*u-d)+mu*u)
    m_sol = P*u

end

d_pred = L*m_sol
subplot(221); plot(x,m,"r", x,m_sol,  "g"); xlabel("x"); title("True and Estimated Mo
subplot(222); plot(r,d,"r", r,d_pred, "g"); xlabel("r"); title("Observed and Predicte
tight_layout()
```