Demo_2_Lapis_2019

April 8, 2019

```
In [2]: using PyPlot,LinearAlgebra
```

1 LAPIS 2019 - UNLP

1.1 Non-quadratic Regularization

We replace the ell-2 norm of the model by an ell-1 norm

$$J = \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2^2 + \mu\|\mathbf{W}\mathbf{m}\|_1$$

We will consider the case $\mathbf{W} = \mathbf{D}_1$ which I like to call Edge Preserving Regularization (EPR)

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

        N = 400
        M = 500


        m = zeros(M)
        m[30:50]     =   m[30:50]   .+1.0
        m[130:150]   = m[130:150]   .-2.0
        m[230:350]   = m[230:350]   .+2.0
        x_max = 100.
        x_min =   0.

        r_max = 100.
        r_min =   0.

        alpha = 0.1
        A = 0.01


        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]
```

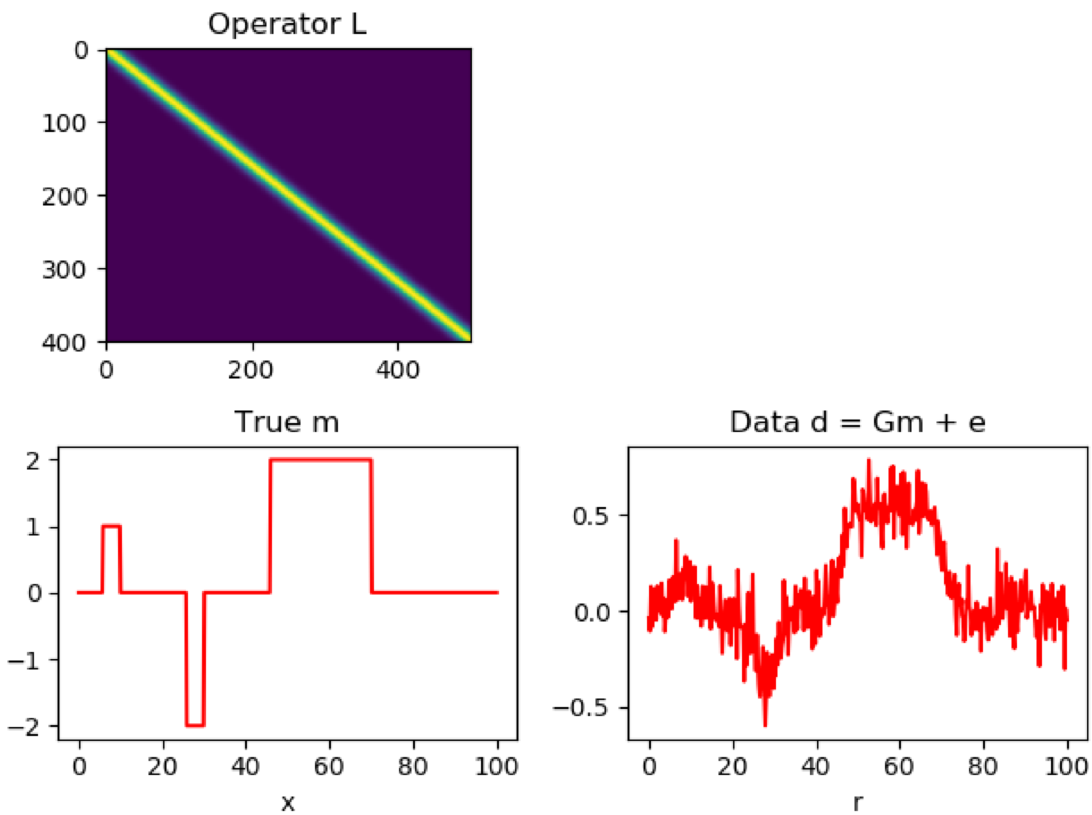
```
L = A*[exp(-alpha*(( x[i]-r[j] )^2)) for j in 1:N, i in 1:M]

# 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 = Gm + e");xlabel("r")

tight_layout()
```



In [5]:
```
function irls(G,d,mu,dx;order=1)

    # least-squares solution with smoothing
    # order = 1 -> First order quadratic regularization: D1
    # order = 2 -> Second order quadratic regularization: D2

    N,M = size(G);
    I = diagm(0=>ones(M))
    if order==1
```

```julia
            D = -diagm(0=>ones(M))+diagm(1=>ones(M-1))
        end

        if order==2
            D = (-diagm(-1=>ones(M-1))+2*diagm(0=>ones(M))-diagm(1=>ones(M-1)))/dx^2
        end

        m_sol = (G'*G +mu*I)\(G'*d)

        for k =1:100
            u = 1.0./(0.00001 .+ abs.(D*m_sol))
            Q = diagm(0=>u)
            m_sol = (G'*G +mu*D'*Q*D)\(G'*d)
        end

        d_pred = G*m_sol

        return m_sol, d_pred

    end

    function wls(G,d,mu,dx;order=1)

        # least-squares solution with smoothing
        # order = 1 -> First order quadratic regularization: D1
        # order = 2 -> Second order quadratic regularization: D2

        N,M = size(G);

        if order==1
            D = (-diagm(0=>ones(M))+diagm(1=>ones(M-1)))/dx
        end

        if order==2
            D = (-diagm(-1=>ones(M-1))+2*diagm(0=>ones(M))-diagm(1=>ones(M-1)))/dx^2
        end

        m_sol = (G'*G +mu*D'*D)\(G'*d)
        d_pred = G*m_sol

        return m_sol, d_pred

    end
```

Out[5]: wls (generic function with 1 method)

In [6]:
```julia
mu = .1
m_sol,d_pred= irls(L,d,mu,dx;order=1)
```

```
subplot(221); plot(x,m,"r",x,m_sol, "g"); xlabel("x"); title("True and Estimated Model
subplot(222); plot(r,d,"r",r,d_pred,"g"); xlabel("r"); title("Data and Predicted Data")

mu = 1.
m_sol,d_pred= wls(L,d,mu,dx;order=2)

subplot(223); plot(x,m,"r",x,m_sol, "g"); xlabel("x"); title(L"True and Estimated Model
subplot(224); plot(r,d,"r",r,d_pred,"g"); xlabel("r"); title("Data and Predicted Data")


tight_layout()
```