Demo_1_Lapis_2019

April 8, 2019

```
In [6]: using PyPlot,LinearAlgebra
```

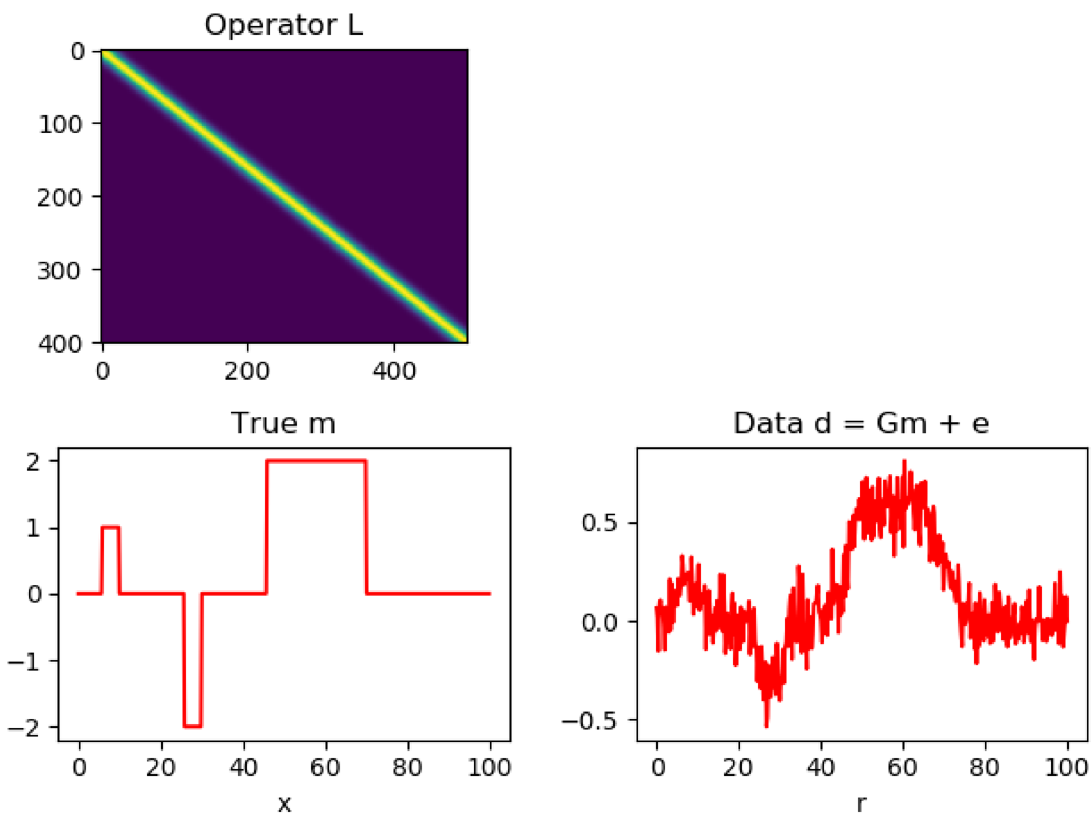
1 LAPIS 2019 - UNLP

Make a forward problem to generate data and solve it with DLS etc

$$d(r_j) = \int_{x_{min}}^{x_{max}} Ae^{-\alpha(r_j - x)^2} m(x) dx + e(r_j)$$

```
In [16]: # 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()
```



1.0.1 Damped least squares

Our first attempt to solve the inverse problem involves minizing the cost function

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

The solution is given by

$$m_{sol} = (\mathbf{G}'\,\mathbf{G} + \mu\mathbf{I})^{-1}\,\mathbf{G}'\mathbf{d}\,.$$

Similarly, you can use the solution to predict the data

$$\mathbf{d}_{pred} = \mathbf{G}\,\mathbf{m}_{sol}.$$

Last point is important, you should compare observation to predictions to access fit.

1.0.2 Least squares with smoothing constraints

Our second attempt is to minimize

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

that leads to

$$\mathbf{m}_{sol} = (\mathbf{G}'\mathbf{G} + \mu\mathbf{W}'\mathbf{W})^{-1}\mathbf{G}'\mathbf{d}$$

with - $\mathbf{W} = \mathbf{D}_1$ (First order quadratic regularization) - $\mathbf{W} = \mathbf{D}_2$ (Second order quadratic regularization)

In [17]:
```
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

function dls(G,d,mu)

    # least-squares solution with damping


    N,M = size(G);

        I = diagm(0=>ones(M))

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

    return m_sol, d_pred

end
```

```
Out[17]: dls (generic function with 1 method)
```

```
In [19]: mu = 0.001
         m_sol,d_pred = dls(L,d,mu)
         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 Mode
         subplot(224); plot(r,d,"r",r,d_pred,"g"); xlabel("r"); title("Data and Predicted Data'

         tight_layout()
```