Prof. C. Naaktgeboren, PhD
https://github.com/CNThermSci/ApplThermSci

A(5)(3)-pt – Modelagem de Dados via Mínimos Quadrados

Introdução

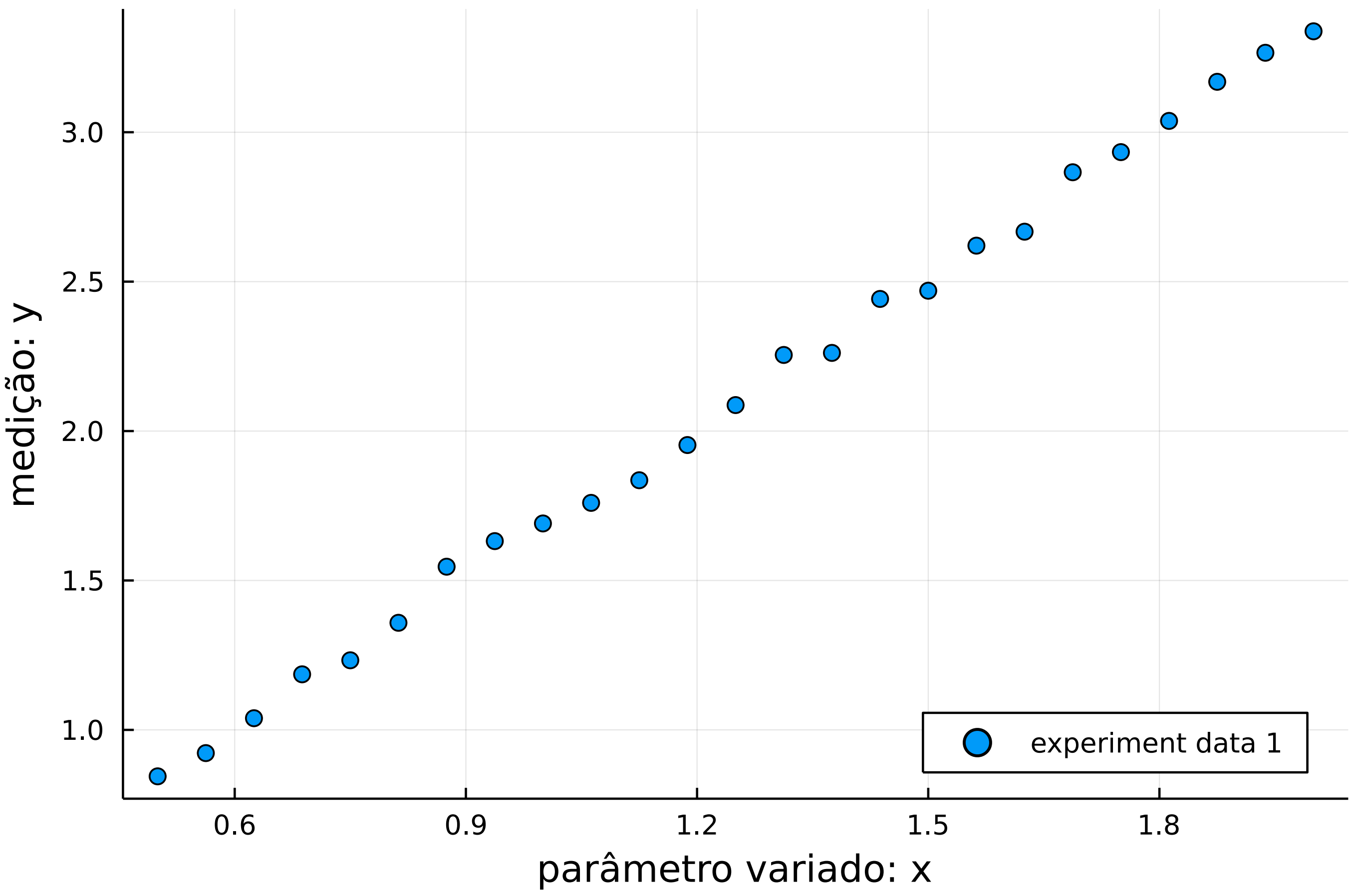
Conjuntos de experimentos com *variação de parâmetro(s)* frequentemente são feitos objetivando determinar o *impacto* do(s) parâmetro(s) variado(s) no(s) fenômeno(s) testado(s).

Suponha que um *conjunto de experimentos* tenha resultado nos **dados abaixo**, no qual o primeiro grupo de dados representa o parâmetro variado, e o segundo, o correspondente valor medido (nominal) à partir do experimento:

```
expData1 =
▸(Float64[0.5, 0.5625, 0.625, 0.6875, 0.75, 0.8125, 0.875, 0.9375, 1.0, ⋯ more ,2.0], Floa
```

O correspondente *gráfico de dispersão* é mostrado abaixo, one as abscissas — as quais chamaremos de x — são os parâmetros variados e as ordenadas — as quais chamaremos de y — são as medições (nominais) de interesse:

```
sca1 =
```



A aparência do gráfico sugere a existência de uma **relação funcional** entre x e y, com uma certa *sobreposição de ruído*, o qual pode ser oriúndo dos erros aleatórios associados ao experimento e às medições.

Ainda, a aparência do gráfico sugere uma *relação linear* entre y e x, ou seja: $y = a_0 + a_1 x$.

Definição:

A **modelagem de dados** é o procedimento que visa determinar *quantitativamente* **coeficientes** para certas *funções* **propostas**, tais que minimizem, de alguma forma, as diferenças entre a *relação funcional proposta* e os dados sob modelagem.

Tal procedimento é conhecido como *regressão* em português, ou *fit*, em inglês.

Neste exemplo, foi proposta:

- Uma função *linear* entre y e x, a saber: $y = a_0 + a_1 x$;
- A *modelagem de dados* visa determinar os coeficientes a_0 e a_1.

Ainda, cabe observar que:

- A modelagem **não** tem a função de *propor* a *relação funcional*!
- Este é o papel do(a) *analista*!

Exemplo - Regressão Linear Manual

Em *regressão linear* (ou regressão afim), ajusta-se dois parâmetros da função linear (afim) proposta, a saber: a_0 e a_1, tal que a função resultante melhor se aproxime dos dados modelados.

Matematicamente, pode ser demonstrado que a "melhor" aproximação é aquela que **minimiza** a soma das diferenças (entre modelo e dados) **ao quadrado**, tal que desvios positivos não cancelem desvios negativos.

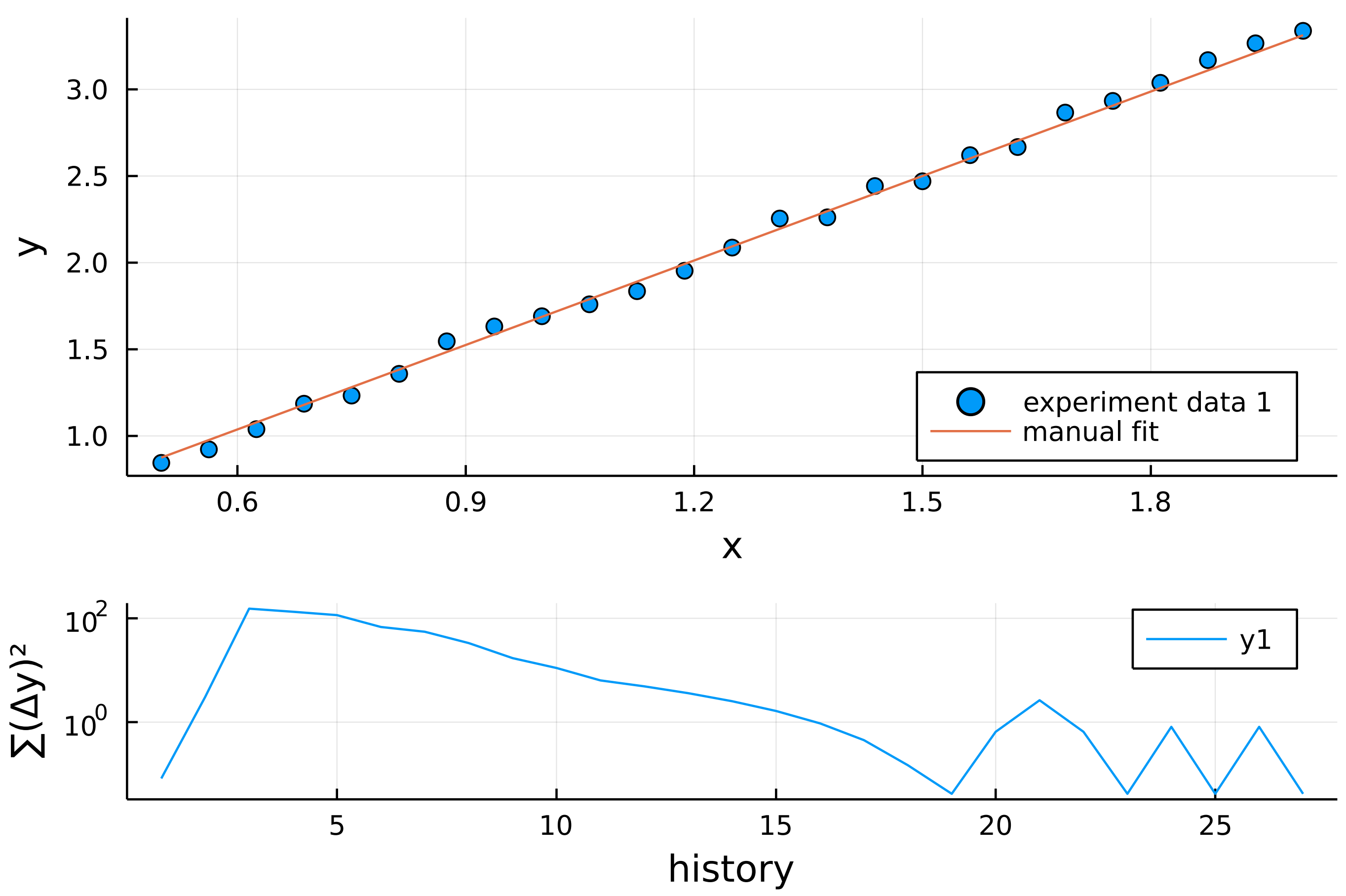
Na ilustração abaixo, os valores de a_0 e a_1 estão associados ao *sliders*, podendo ser ajustados manualmente:

0.0625

```
@bind manu_a0 Slider(-0.5:0.0625:+0.5, show_value=true)
```

1.625

```
@bind manu_a1 Slider(0.0:0.125:2.5, show_value=true)
```




```
manu_model (generic function with 1 method)
```

```
manu_soma (generic function with 1 method)
```

```
manu_soma_log = ▸[]
```

```
▸[0.0821288, 3.06312, 153.515, 133.666, 115.195, 68.0611, 55.1085, 33.3414, 17.0919, 11.036
```

Mínimos Quadrados Genérico em Julia

Considere o seguinte modelo **genérico** $y(x)$:

$$y(x) = \sum_{k=0}^{M-1} a_k X_k(x),$$

onde $X_k(x), 0 \leqslant k \leqslant M-1$ são M **funções-base** do modelo linearmente independentes entre si para os N valores de y_i obtidos experimentalmente. O objetivo da regressão é determinar os valores dos M coeficientes a_k.

Seja $\mathbf{X}_{ij}$ uma matriz $M \times N$ com componentes construídos conforme as M funções-base aplicadas aos N valores do parâmetro x variado nos experimentos, da seguinte forma:

$$\mathbf{X}_{ij} = X_j(x_i).$$

Ainda, o vetor $\mathbf{y}_i$, de N componentes:

$$\mathbf{y}_i = y_i.$$

O vetor-coeficientes $\mathbf{a}_j$ que minimiza $||\mathbf{X}x - \mathbf{y}||^2$ é dado por:

$$\mathbf{a} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y},$$

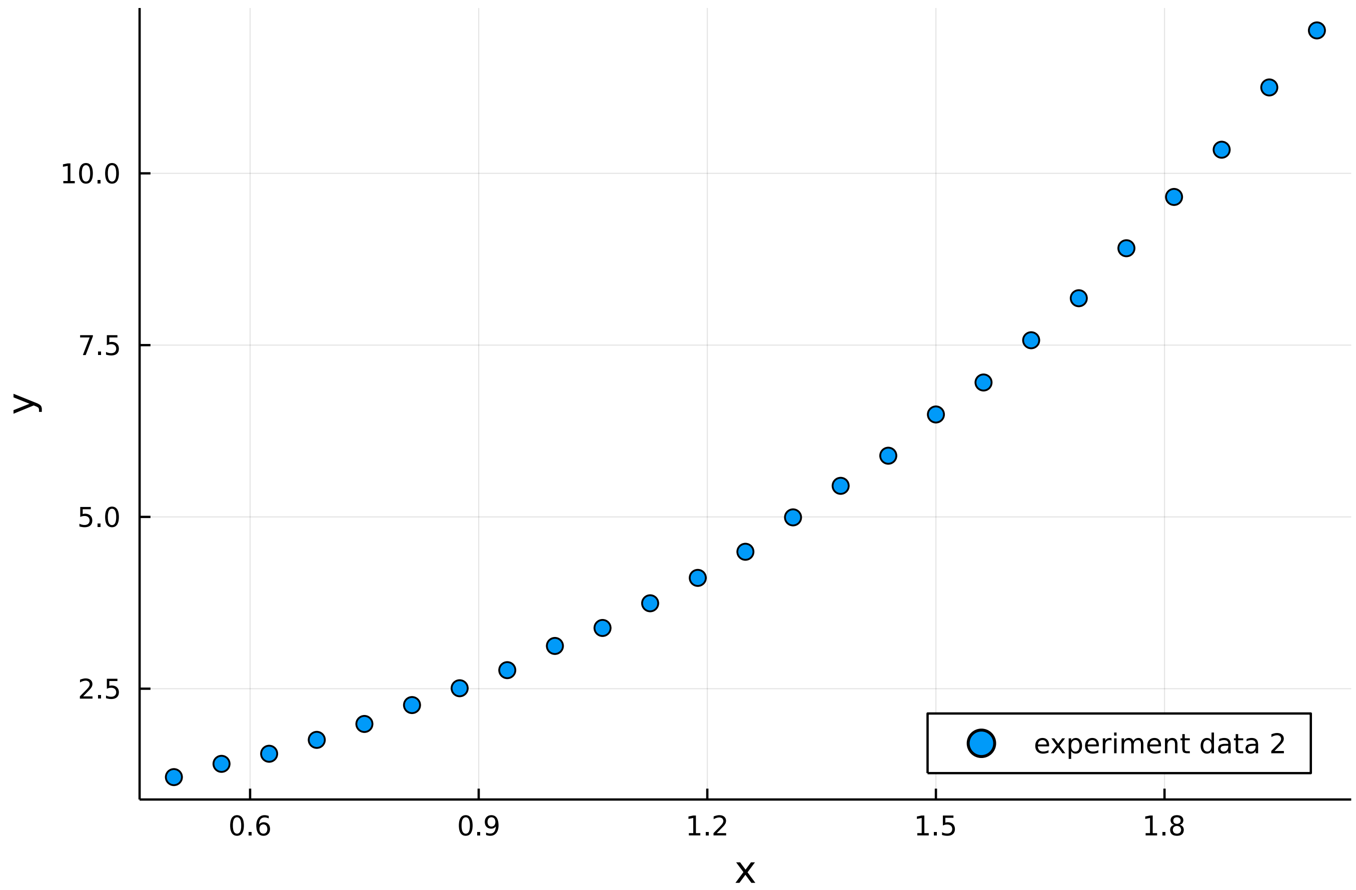
que em `Julia` é implementado pelo operador `\`:

```
a = X \ y
```

Exemplo: Mínimos Quadrados Genéricos em Julia:

Considere o novo conjunto de dados abaixo:

```
expData2 =
▸([0.5, 0.5625, 0.625, 0.6875, 0.75, 0.8125, 0.875, 0.9375, 1.0, ⋯ more ,2.0], [1.21336, 1.
```



Propondo um polinômio do segundo grau como modelo, isto é:

$$y(x) = a_0 x^0 + a_1 x + a_2 x^2,$$

tem-se: $X_0(x) = 1$, $X_1(x) = x$ e $X_2(x) = x^2$, e a matriz $\mathbf{X}$ é construída:

```
MODELS =
▶Dict("linear+sin" ⇒ [#180, #181, #182], "cubic" ⇒ [#176, #177, #178, #179], "inverse" ⇒
```

```
MODELS = Dict(
    "quadratic" => [
        x -> one(x),
        x -> x,
        x -> x^2,
        ],
    "cubic" => [
        x -> one(x),
        x -> x,
        x -> x^2,
        x -> x^3,
        ],
    "linear+sin" => [
        x -> one(x),
        x -> x,
        x -> sin(2π*x),
        ],
    "inverse" => [
        x -> one(x),
        x -> inv(x),
        ],
    "exponential" => [
        # x -> one(x),
        x -> exp(x),
        ],
)
```

```
leastSq (generic function with 1 method)
```

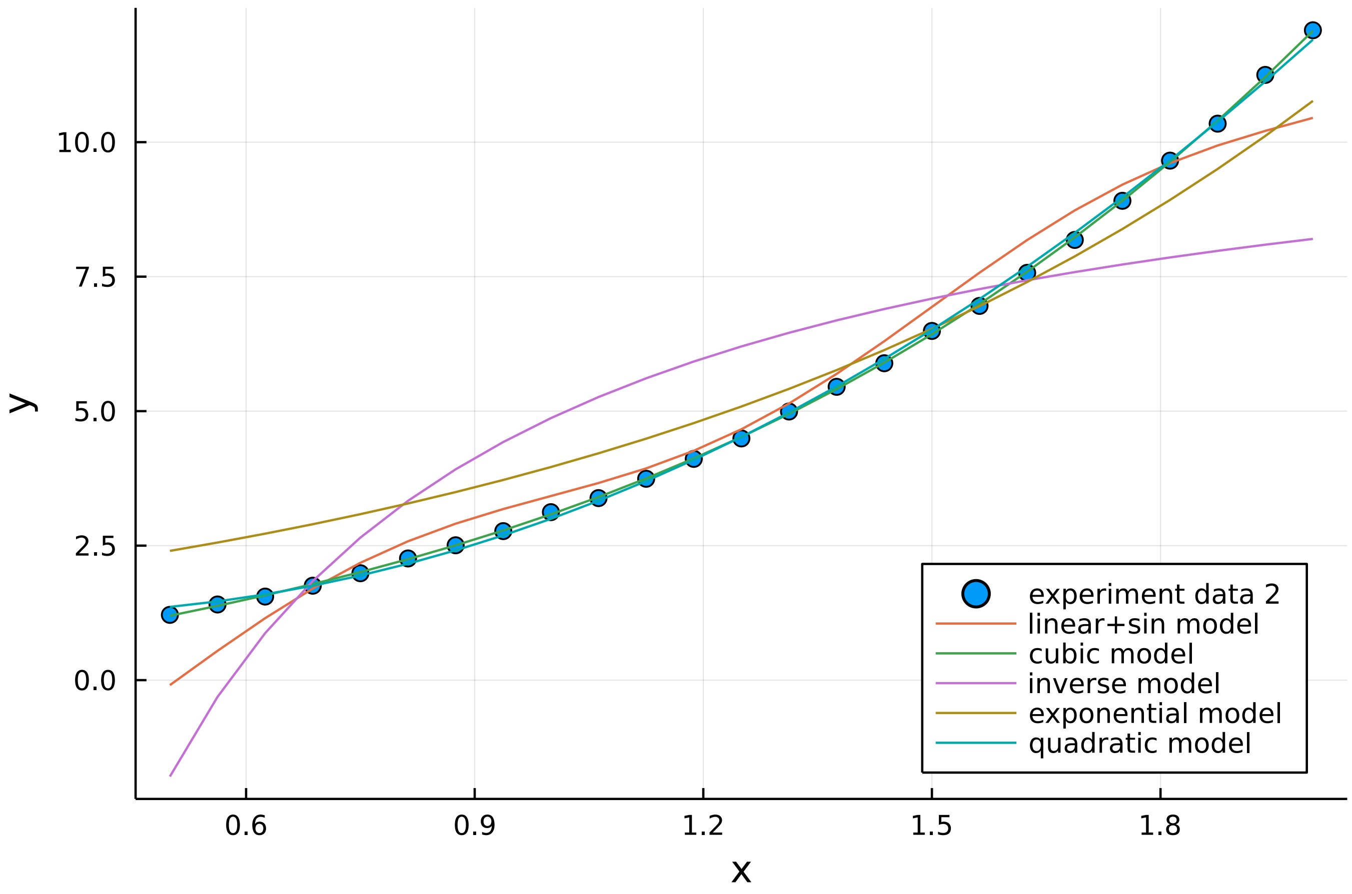
```
function leastSq(the_x, the_y, MODEL)
    X = hcat([ F.(the_x) for F in MODEL ]...)
    y = copy(the_y)
    a = X \ y
end
```

```
model (generic function with 1 method)
```

```
model(MOD, a, x) = sum(a .* [F.(x) for F in MOD])
```

```
sols = Tuple{String, Vector{Float64}}[
        1: ("linear+sin", [-3.60717, 7.02937, -0.516411])
        2: ("cubic", [-0.0558713, 2.42737, -0.397387, 1.1075])
        3: ("inverse", [11.5328, -6.66243])
        4: ("exponential", [1.45717])
        5: ("quadratic", [1.60213, -2.35997, 3.75574])
     ]
```

```julia
sols = collect((key.first, leastSq(expData2..., key.second)) for key in MODELS)
```



Bibliotecas e Demais Recursos

Bibliotecas

```julia
begin
    using PlutoUI
    using Random
    using Plots
end
```

```
0.25:0.03125:0.5
```

```
newSet (generic function with 1 method)
```