Uma Plataforma de Software para o Estudo Interativo de Métodos e Algoritmos Econométricos

Carlos Duarte do Nascimento

Instituto de Matemática e Estatística
Universidade de São Paulo

4 de Março de 2009

Introdução

Apresentações

Banca Avaliadora

- Prof. Cicely Moitinho Amaral (orientador)
- Prof. Claudio Possani
- Prof. Sergio Muniz Oliva Filho

O que este trabalho *não* é

- Análise de um Problema Matemático / Econométrico
- Um Estudo Aprofundado de Métodos Numéricos
- Apologia (ou Crítica) do Ensino à Distância

O Problema

O Ensino de Econometria

- Teoria + Prática: É importante experimentar!
- Opções para experimentar:
  - Softwares específicos de Estatística/Econometria (EViews, SPSS, Stata)
  - Pacotes Matemáticos "puros"(Mathematica, Gnu R, Matlab, Octave)
  - Linguagens de Programação (C, Java, Pascal, Fortran, etc.)
    - Desenvolvimento pelo Professor
    - Desenvolvimento pelo Aluno

Um Problema Econométrico

(Judge) Estimação de Parâmetros no Modelo:

$$y_t = \theta_1 + \theta_2 x_{t2} + (\theta_2)^2 x_{t3} + e_t, t = 1, 2, ..., 20$$

$$y = f(\theta) + e$$

$$f(\theta) = \begin{pmatrix} \theta 1 + \theta_2 x_{12} + \theta_2^2 x_{13} \\ \theta 1 + \theta_2 x_{22} + \theta_2^2 x_{23} \\ . \\ . \\ . \\ \theta 1 + \theta_2 x_{20,2} + \theta_2^2 x_{20,3} \end{pmatrix}$$

Função objetivo (soma quadrática do erro):

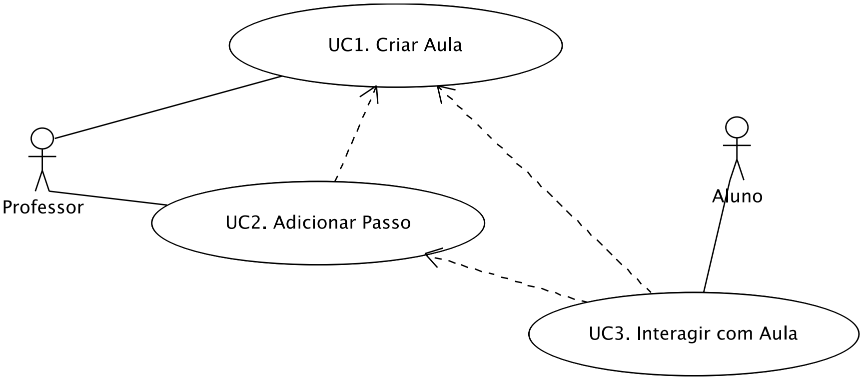
$$H(\theta) = [y - f(\theta)]'[y - f(\theta)] \tag{1}$$

A Plataforma

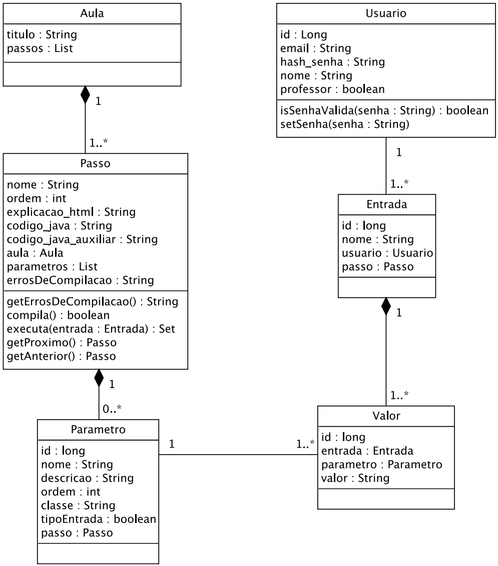
Proposta Funcional

- Duas Categorias de Usuários:
  - Professores (cadastram aulas)
  - Alunos (interagem com aulas)
- Aulas Divididas em Passos
- Passos Divididos em:
  - Parte Teórica: Texto/HTML
  - Parte Prática: Algoritmo interativo

Casos de Uso


UC1. Criar Aula
Professor
UC2. Adicionar Passo
Aluno
UC3. Interagir com Aula

Diagramas de Classe



Arquitetura de Software

Escolha da Linguagem

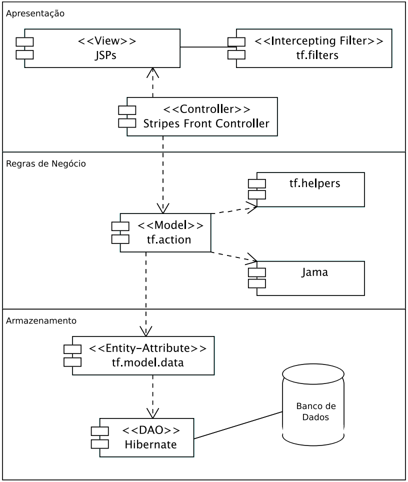
Cada uma apresenta suas vantagens:

- C/C++: Performance
- Pascal: Simplicidade
- Fortran: Material Acadêmico
- Java: Equilíbrio destes fatores; facilidade para compilação dinâmica; JAMA

Padrões de Projeto

- Mapeamento Objeto-Relacional
- Model / View / Controller
- Inversão de Controle / Injeção de Dependências

Componentes



Compilação Dinâmica de Algoritmos

- Idéia: usar a própria linguagem para executar os algoritmos
- Cadastro do Algoritmo (ex.: Cálculo de Juros):
  - Parâmetros de Entrada
    (taxa, valor)
  - Parâmetros de Saída
    (juros)
  - Algoritmo
    *(juros = valor*taxa; valor = valor + juros)*
  - Código Auxiliar (opcional)
    (função para Tabela Price)
- O sistema monta o código em tempo real usando estes elementos, e o *javac* se encarregad o resto.

Demonstração

Uma Aula Prática

Um Problema Econométrico (retomando)

Modelo:

$$y_t = \theta_1 + \theta_2 x_{t2} + (\theta_2)^2 x_{t3} + e_t, t = 1, 2, ..., 20$$

$$y = f(\theta) + e$$

Função-Objetivo:

$$H(\theta) = [y - f(\theta)]'[y - f(\theta)]$$

Métodos de Gradiente

Idéia Geral

$\theta_{n+1} = \theta_n - t_n P_n \gamma_n$

- P_n: direção
- t_n: "distância"
- γ_n: gradiente de H

Condições de Parada

1. $(\theta_{n+l} - \theta_n)'(\theta_{n+l} - \theta_n) < \epsilon$
2. $H(\theta_n) - H(\theta_{n+l}) < \epsilon$
3. $[\frac{\partial H}{\partial \theta}|_{\theta_n}]'[\frac{\partial H}{\partial \theta}|_{\theta_n}] < \epsilon$

Newton-Rhapson

Usar o inverso da matriz Hessiana para especificar a direção do passo em cada iteração, ajustando-o pelo gradiente, isto é:

$$\theta_{n+1} = \theta_n - \mathcal{H}_n^{-1}\gamma_n$$

$\mathcal{H}_n$: Hessiano de $H(\theta)$ aplicado em θ_n, isto é:

$$\mathcal{H}_n = \left[\left. \frac{\partial^2 H}{\partial\theta\partial\theta'} \right|_{\theta_n} \right]$$

Newton-Rhapson: implementando

```
double[][] hess = hess(theta1, theta2, y, x2, x3);
Matrix invHess = new Matrix(hess).inverse();
Matrix grad = new Matrix(
    gradH(theta1, theta2, y, x2, x3), 2);
Matrix passo = invHess.times(grad);
```

Gauss-Newton

Definindo $Z(\theta) = [\partial f / \partial \theta' |_{\theta}]$, o passo é dado por:

$$\theta_{n+1} = \theta_n + [Z(\theta_n)' Z(\theta_n)]^{-1} Z(\theta_n)' [y - f(\theta_n)]$$

que é o EMQ para o modelo:

$$\bar{y}(\theta_n) = Z(\theta_n)\theta + e$$

Caracterísitcas

- Funciona como uma sequência de regressões lineares
- Restrito a funções-objetivo que são somas de quadrados *ótimo:* $H(\theta) = [y - f(\theta)]'[y - f(\theta)] = e(\theta)' e(\theta)$

Gauss-Newton: implementando

```
Matrix Z = Z(theta1, theta2, x2, x3);
Matrix Zt = Z.transpose();
Matrix f = f(theta1, theta2, x2, x3);
Matrix passo = Zt.times(Z).inverse()
    .times(Zt)
    .times(y.minus(f)).times(-1);
```

Demonstração

Conclusão

Lições Aprendidas

- Não subestimar o aspecto educacional/didático
- É difícil conciliar simplificação e funcionalidades - mas coompensa

Sugestões para Continuidade

- Cadastro de usuários (possivelmente vinculado a algum sistema de matrícula);
- Possibilidade de salvar e recuperar dados;
- Permitir ao código decidir o próximo passo a ser executado;
- Uma interface mais amigável para o professor (em particular na visualização do texto das aulas e da depuração do código dos algoritmos);
- Possibilidade de usar outros sistemas de codificação, como LaTeX/MathML na composição da parte teórica;
- Conversão semi-automática de algoritmos em outras linguagens (ex.: FORTRAN).

Obrigado!