Simulador Gráfico de Algoritmos Matemáticos

Luís M. Alves
Instituto Politécnico de Bragança
Bragança, Portugal
lalves@ipb.pt

Carlos Balsa
Instituto Politécnico de Bragança
Bragança, Portugal
balsa@ipb.pt

Maria João Varanda Pereira
Instituto Politécnico de Bragança
Bragança, Portugal
mjoao@ipb.pt

Resumo

O objetivo principal do trabalho de investigação foi desenvolver uma ferramenta computacional que possa ser usada por professores e alunos no ensino dos Métodos Numéricos. A ferramenta, designada por GraSMA (*Graphical Simulator of Mathematical Algorithms*), permite visualizar no ecrã a execução dos correspondentes algoritmos. Esta ferramenta integra vários softwares *open source* e baseia-se na anotação automática de código Octave com funções de inspeção que permitem captar a sucessão de valores e produzir uma animação do funcionamento do método. A validação do GraSMA como ferramenta de apoio à lecionação destes métodos foi feita através de uma experiência feita em contexto de sala de aula que obteve resultados muito positivos conforme será descrito neste artigo.

Palavras-chave: e-learning, representação gráfica, métodos numéricos, octave, XML.

1 Introdução

Os olhos e o córtex visual do cérebro formam um processador massivamente paralelo que fornece o canal de maior largura de banda em centros cognitivos humanos. Considerando este fato, podemos facilmente assumir que a representação visual dos dados amplia a cognição, alavancando a perceção e compreensão de ideias complexas com clareza, precisão e eficiência.

Como uma extensão a este conceito, é claro que a representação geométrica dos conceitos matemáticos irá fornecer um método valioso para ajudar a interpretar e compreendê-los. A partir desta perspetiva, os métodos numéricos não podem mais ser vistos como uma sequência de linhas de código ou um algoritmo a seguir, mas um rico conjunto de objetos gráficos em movimento.

Neste contexto, desenvolvemos uma ferramenta *open source*, a que atribuímos o nome GraSMA (*Graphical Simulator of Mathematical Algorithms*) que pode ser usada por professores e alunos em diferentes unidades curriculares que contenham métodos numéricos no seu programa curricular. O GraSMA irá ajudar a compreender vários conceitos como, solução aproximada, iteração, convergência, erros, etc.

Atualmente, existem várias aplicações informáticas na área da educação matemática. Algumas são comerciais e outras são livres. A grande maioria destas aplicações destinam-se ao ensino secundário. No entanto, aplicações informáticas que auxiliem o ensino de métodos numéricos no ensino superior são escassos. Neste contexto, destacamos o "*Interactive Educational Modules in Scientific Computing*" disponível *online* no site http://heath.cs.illinois.edu/iem/. Trata-se de um conjunto alargado de ferramentas que ilustram conceitos básicos e algoritmos da computação científica. Estas ferramentas foram desenvolvidas pelo *Department of Computer Science* da *University of Illinois* nos Estados Unidos da América. Cada módulo destas ferramentas é um *applet* Java que é acessível através de um navegador *web*. Para cada *applet*, podemos selecionar, a partir de uma lista, o algoritmo pretendido e inserir os dados de forma interativa e em seguida, receber feedback imediato sobre os resultados, tanto numérica como graficamente. A nossa abordagem difere desta porque é *open source* e genérica, aberta à inclusão de novos métodos matemáticos que possam ser ilustrados graficamente.

O GraSMA permite exibir no ecrã a execução de algoritmos correspondentes a métodos numéricos, codificados previamente em Octave (Eaton, Bateman, & Hauberg, 2008). Para o utilizador final, a ferramenta é, por conseguinte, uma sequência de algoritmos parametrizados cujos passos podem ser visualizados graficamente. Neste sentido, foi seguida uma estratégia de anotação automática do algoritmo original através de funções de inspeção. A automatização da inserção das funções de inspeção no código Octave envolveu a utilização de um processador de linguagem. Este processador de linguagem foi construído com base nas ferramentas Lex (*Lexical*) e YACC (*Yet Another Compiler-Compiler*).

A versão atual da ferramenta pode ser utilizada a uma classe particular de algoritmos numéricos que calcula os zeros de funções não lineares. No entanto, temos como objetivo estender a ferramenta a uma classe maior de métodos numéricos.

Com a possibilidade de testarmos a ferramenta em contexto de ensino-aprendizagem, algumas questões surgiram de imediato. Em concreto, a utilização da ferramenta desenvolvida reporta-nos à seguinte questão de investigação: a utilização da ferramenta em contexto de sala de aula melhorará o tempo de resolução, correção e compreensão do algoritmo numérico que implementa?

Para responder a esta questão testou-se o GraSMA, em de sala de aula, como meio auxiliar pedagógico. Avaliou-se a sua eficácia na lecionação da unidade curricular de Matemática Computacional, através de exercícios diagnósticos e de um inquérito.

Na secção 2 faz-se uma descrição da arquitetura do software GraSMA com a descrição dos seus vários componentes e da maneira como estão interligados. Segue-se, na secção 3, uma breve descrição da sua utilização. Apresenta-se na secção 4 um estudo que valida a sua utilização como ferramenta pedagógica de apoio à lecionação de métodos numéricos. Termina-se com a apresentação de algumas observações finais na secção 5.

2 Visão geral do GraSMA

O GraSMA é uma ferramenta com fins educativos cuja primeira versão data de 2010 (Balsa, Alves, Pereira, & Rodrigues, 2010) e que tem sido continuamente completada de forma a melhorá-la (Balsa, Alves, Pereira, Rodrigues, & Lopes, 2012). O GraSMA é um simulador gráfico de algoritmos numéricos que cria, automaticamente, gráficos, descrevendo o seu comportamento geométrico. Para este fim, a aplicação usa algoritmos instrumentados automaticamente com funções de inspeção que auxiliam na conversão dos ficheiros gerados pelo Octave. A estes ficheiros são adicionados as funções inspeção e posteriormente convertidos em ficheiros ".xml" pela ferramenta desenvolvida. Os ficheiros resultantes contêm nodos correspondentes a objetos matemáticos que o Java irá ler, criar instâncias desses objetos, armazená-los numa classe e posteriormente visualizá-los. Neste processo, é usado um *parser* e o OpenGL que permite visualizar os objetos matemáticos criados. No desenvolvimento das várias componentes da ferramenta, procurou-se a utilização, apenas, de software *open source*.

A aplicação foi desenvolvida em linguagem de programação Java (Cadenhead & Lemay, 2007) e os algoritmos matemáticos em Octave. Na codificação Java, usou-se o IDE (*Integrated*

Development Environment) Netbeans que contém o *Java Runtime Environment*. Várias bibliotecas do Java foram usadas, sendo as mais importantes, o *Swing*, o *OpenGL* (*Open Graphics Library*) e o AWT (*Abstract Window Toolkit*).

Para construir a informação essencial no sentido de animar as iterações do algoritmo optamos pela instrumentação do código (*Code Instrumentation*). Esta técnica é bem conhecida na área da compreensão de programas, ver, por exemplo (Berón, Henriques, Pereira, & Uzal, 2007) e (Cruz, Beron, Henriques, & Pereira, 2009), e, geralmente, é adotada quando o objetivo é visualizar programas desenvolvidos numa linguagem específica. A ideia principal por detrás deste processo é anotar o código-fonte com funções de inspeção. Isto permitirá a recuperação de informação estática e dinâmica da execução do programa.

Para armazenar esta informação, um *Document Type Definition* (DTD) foi criado por forma a gerar uma representação intermédia em XML (*eXtensible Markup Language*) (Ramalho & Henriques, 2006). O DTD descreve como representar a informação obtida na execução do algoritmo. Um dos primeiros desafios deste trabalho foi o de determinar o esquema apropriado para descrever um grande número de diferentes algoritmos matemáticos.

Finalmente, a fim de visualizar os algoritmos, a linguagem de programação Java e o OpenGL (Shreiner, 2009) foram utilizados.

A aplicação, baseada em Java e OpenGL, conta com o motor GNU Octave para cálculos numéricos. A seção de visualização é baseada em duas classes predominantes: o GLRenderer2D e o GLRenderer3D.

Por outro lado, a classe OctaveCaller gera o ficheiro XML, com base na execução do algoritmo implementado em Octave. O processo das classes "Renderer" exibe no ecrã uma série de representações dos objetos matemáticos que representam cada passo ou iteração de um método numérico.

O algoritmo é representado, em Java, pela classe *Algorithm*, que descreve o algoritmo Octave através de uma lista de iterações (cada iteração é em si uma lista de objetos matemáticos a serem exibidos). Esta informação é armazenada numa lista de iterações e é obtida através da classe *Parser* que pode processar um ficheiro XML para recuperar os dados das iterações e, assim, colocá-los no campo correspondente na instância da classe *Algorithm*.

Para visualizar a execução do algoritmo dois painéis diferentes são apresentados: o primeiro apresenta alguns elementos padrão que estão sempre no ecrã (chamado os

elementos "globais"). O segundo chama alguns elementos que são visíveis apenas para a iteração que está atualmente em exibição. Esses elementos serão substituídos na próxima iteração. É por isso que é possível ver no diagrama de classes que a classe *Algorithm* está ligada à interface *MathObject* por duas ligações diferentes: uma *interactionList* (lista de iteração em que uma iteração é uma lista de *MathObject*), e uma lista global é apenas uma lista de *MathObject*.

A Figura 1 mostra todos os componentes arquiteturais do GraSMA, bem como a ligação entre eles. Como podemos observar as componentes de Octave e de Java são independentes. O Octave cria o ficheiro XML contendo todos os objetos matemáticos das iterações do algoritmo numérico e o Java apenas lê esse ficheiro XML e mostra os objetos.

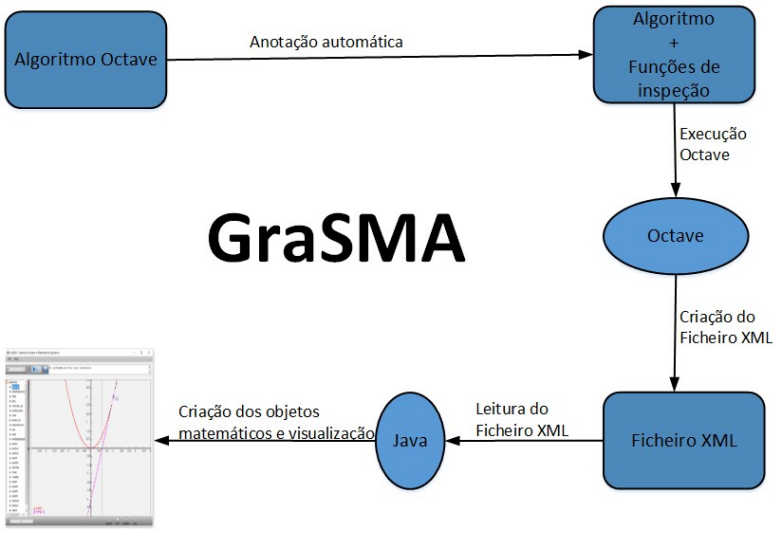

Figura 1 – Arquitetura do GraSMA

Inicialmente, o algoritmo é implementado na linguagem de programação disponibilizada pelo Octave. Como é sabido, o Octave é um ambiente de acesso livre similar ao MatLab, usado fundamentalmente em meios académicos. Ao código que implementa o algoritmo numérico são adicionadas funções de inspeção de forma automática através de um processador de linguagem construído com base nas ferramentas Lex e YACC. Essas funções permitem criar os ficheiros XML. Cada função cria uma *flag* XML que representa um objeto matemático com os seus atributos.

A nossa aplicação lê o ficheiro XML que contém *flags* que representam objetos matemáticos definidos por atributos diferentes. A seguir, a aplicação usa uma classe *Parser* para criar os objetos Java e uma instância da classe *Algorithm* contendo os dados dos ficheiros XML. Estes dados contém a informação dos objetos matemáticos globais, as suas iterações, os seus erros e

alguns dados úteis para a visualização. Finalmente, a aplicação lê a lista de objetos da iteração atual exibindo-os conjuntamente com os eixos do gráfico. O algoritmo numérico pode ser visualizado passo a passo ou, automaticamente, em sequência, bastando para isso, um simples clique.

3 Utilização do GraSMA

Nesta secção iremos abordar os principais aspetos da utilização do GraSMA. A Figura 2 mostra o *layout* do GraSMA correspondente a uma iteração do método iterativo Newton-Raphson. Nesta imagem incluem-se também legendas que explicam as componentes da aplicação.

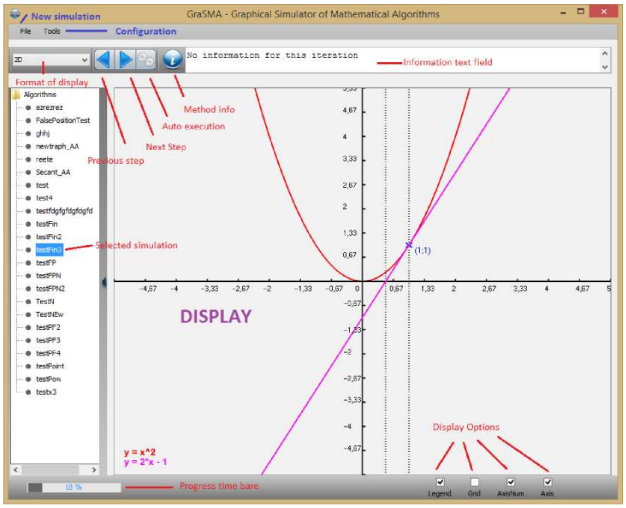

Figura 2 – *Layout* da janela de execução do GraSMA

Para executar uma simulação no GraSMA podemos fazê-lo de duas maneiras diferentes: criando um novo algoritmo ou, em alternativa, importando um algoritmo XML previamente criado (menu *File*, subopção *Import*). Neste último caso, o algoritmo é colocado na lista de algoritmos (ver coluna de algoritmos da Figura 2) para posteriormente ser visualizado.

No caso da criação de um novo algoritmo, temos de selecionar a opção *File*, subopção *Menu*, e preencher a janela exibida com os parâmetros necessários para executar o método numérico escolhido. A Figura 3 mostra a parametrização necessária para executar o método numérico da Falsa Posição. No passo 1 (*step 1)* devemos selecionar o algoritmo Octave anotado e dar-lhe um nome à nossa escolha no passo 2 (*step 2*). Finalmente no passo 3 (*step 3*) devemos indicar os parâmetros correspondentes ao problema a resolver, nomeadamente a função a anular, o intervalo onde está a solução assim como os parâmetros de convergência do método.

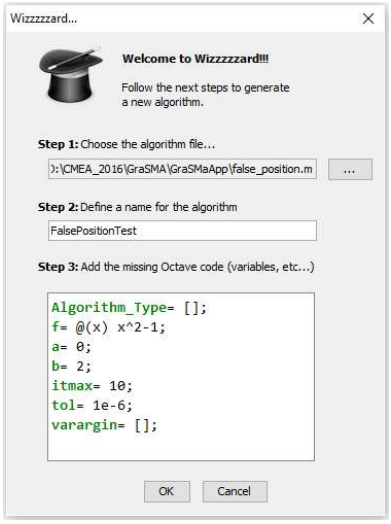

Figura 3 – Exemplo da criação de um novo algoritmo

A Figura 3 mostra a invocação do método da Falsa Posição para encontrar os zeros da função $f(x)=x^2-1$, dado um intervalo inicial [0;2], considerando um número máximo de 10 iterações e um critério de convergência inferior a 10^{-6}.

Após a inserção e validação desta informação, a representação visual do algoritmo pode ser obtida através de duplo clique efetuada no respetivo algoritmo apresentado na coluna mais à esquerda da janela de visualização (ver coluna de algoritmos da Figura 2).

O interface com o utilizador é bastante simples e contém os seguintes botões: ir para a próxima iteração (*Next Step)*; ir para a iteração anterior (*Previous Step*); visualizar automaticamente todas as iterações (*Auto execution*). O progresso da simulação do algoritmo é mostrado na barra inferior da janela de visualização (*Progress time bare*). Caso ocorra alguma informação relevante durante a iteração corrente essa informação é mostrada na caixa de texto na parte superior da janela de visualização (*Information text field*).

Durante a visualização da simulação do algoritmo, podemos alterar algumas opções, nomeadamente, visualizar a legenda, a grelha vertical e horizontal, os números dos eixos e os próprios eixos (*Display Options*). Temos ainda acesso, a informações sobre o método numérico escolhido, clicando no botão azul de informações (*Method info*). Podemos ainda interagir com a aplicação aumentando e diminuindo a visualização usando a roda do rato ou simplesmente clicando sobre o gráfico. Podemos ainda fazer deslocações do gráfico clicando sobre ele e arrastando-o para a zona pretendida.

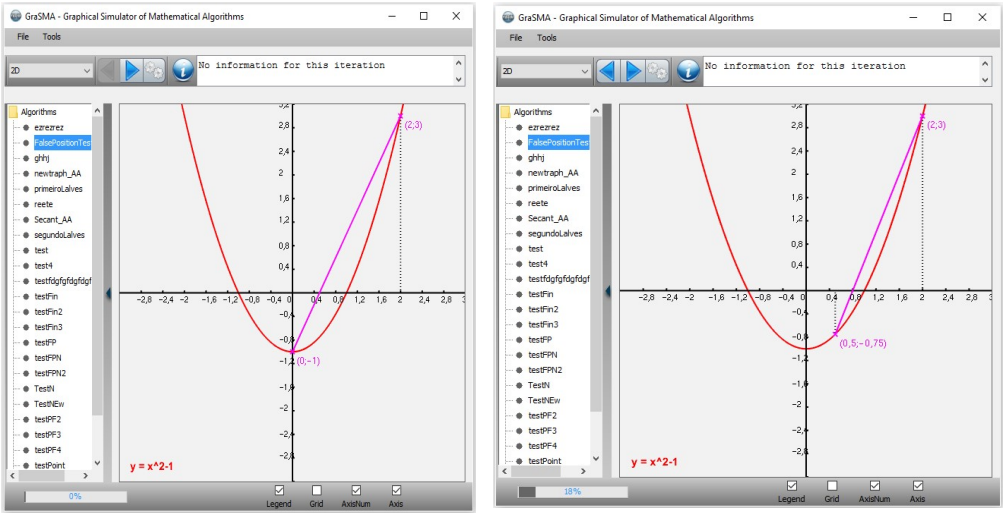

Figura 4 – Primeira e segunda iteração do método da Falsa Posição

A Figura 4 mostra as duas primeiras iterações do método da Falsa Posição após inserção dos parâmetros apresentados na Figura 3. Neste método o zero da função é aproximado através da reta secante à função (reta a roxo na figura) nos pontos *a* e *b.*

4 Validação do GRASMA

Nesta secção descreve-se uma experiência de validação do GraSMA quando utilizado pelo professor como ferramenta pedagógica. A experiência decorreu nas aulas de Matemática Computacional cujo conteúdo programático consiste no ensino de métodos numéricos para a resolução de problemas formulados matematicamente. As aulas são lecionadas em salas equipadas com computadores de forma a permitir a implementação real dos algoritmos em linguagem Octave em vez da habitual exposição teórica dos métodos.

A ferramenta GraSMA foi utilizada para o capítulo sobre a resolução de equações não lineares através do método de Newton-Raphson. O método de Newton-Raphson é um método numérico iterativo que aproxima o zero da função $f(\mathrm{x})$ através de aproximações sucessivas, designadas por iterações. Em cada iteração repete-se o mesmo procedimento. Se designarmos o valor aproximado da solução na iteração *k* por x_k, a solução aproximada de $f(\mathrm{x})=0$ obtida na iteração seguinte (x_{k+1}) é dada pela seguinte fórmula de recorrência:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad \text{para} \quad k = 0,1,2\ldots \tag{1}$$

Em que $f'(\mathrm{x})$ representa a derivada de $f(\mathrm{x})$. A aplicação desta fórmula só é possível se estiver disponível uma aproximação inicial à solução x_0 que tem de ser arbitrado. A sua escolha tem uma importância fundamental pois pode definir logo à partida se o método vai convergir para a solução ou não.

O método introduzido desta forma não passa de mais uma fórmula que o aluno assimila sem compreender o seu significado. Esta situação é habitualmente desmotivante para o aluno e para o professor. Contudo se analisarmos este método do ponto de vista geométrico verifica-se que a ideia subjacente é extremamente simples. Em cada iteração k, o método de Newton-Raphson consiste em substituir a função pela reta tangente no ponto x_k e considerar o zero dessa reta como aproximação do zero da função, tal como representado na Figura 5.

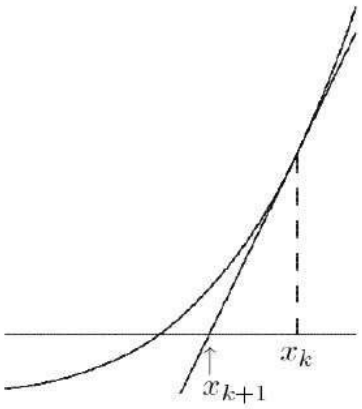

Figura 5 – Ilustração do método de Newton-Raphson

A ferramenta GRASMA permite fazer a representação gráfica da execução do método de Newton-Raphson assim como de outros métodos de resolução de equações não lineares. Tal como descrito na secção anterior, após a definição do problema que se quer resolver o GraSMA representa graficamente a função a anular assim como as aproximações obtidas pelo método iteração a iteração. Desta forma é possível verificar visualmente se há convergência da sequência produzida pelo método ou se, pelo contrário, se há divergência. Com base nestas observações será mais fácil arbitrar um ponto de partida x_0 para iniciar o método num próximo problema.

4.1 Experiência do GRASMA em sala de aula

Para testar a eficácia da ferramenta GraSMA no ensino dos métodos numéricos dividiu-se a experiência em duas partes: numa primeira sessão de quatro horas foi feita a introdução do

método de Newton-Raphson e foi proposto aos alunos um exercício sem utilização do GraSMA. Numa segunda sessão de quatro horas (uma semana depois da primeira) foi efetuado um segundo exercício diagnóstico a resolver usando o GraSMA. Os dois exercícios diagnósticos são diferentes mas incidem sobre o mesmo problema de resolver uma equação não linear pelo método de Newton-Raphson. Nestes, procura-se saber se o aluno compreende o funcionamento do método e como tal, se sabe arbitrar uma estimativa inicial da solução x_0 e, no caso de esta ser dada, saber se a sequência produzida pelo método se aproxima da solução. Avalia a capacidade de aplicar a fórmula de recorrência dada pela equação (1). No final da experiência obtivemos a informação necessária para fazer uma análise comparativa do desempenho do aluno nos dois testes propostos.

O segundo processo de avaliação da ferramenta consistiu num inquérito distribuído aos alunos onde se procura saber a opinião destes em relação à ferramenta GraSMA. Este questionário foi distribuído após os alunos terem entregado a sua resolução do segundo exercício diagnóstico. O questionário proposto foi o seguinte:

1- Classifique, numa escala de 1 a 5 (1- Discordo totalmente; 2- Discordo; 3- Nem concordo nem discordo; 4- Concordo e 5- Concordo totalmente), as seguintes afirmações:

   a. A visualização gráfica dos métodos é uma ajuda fundamental para a sua compreensão. ____

   b. A animação da execução dos métodos traz mais-valias em relação à sua visualização estática. ____

   c. O acompanhamento da animação passo a passo ajudou a resolver os problemas propostos. ____

2- Quanto tempo demorou a resolver os exercícios?

3- Tem alguma sugestão de melhoria relativamente à animação gerada?

5 Análise dos resultados

Os resultados dos dois exercícios diagnósticos estão apresentados na Tabela 1. A avaliação foi feita numa escala de zero a cem. Participaram no primeiro exercício 13 alunos e no segundo participaram 10. Verifica-se claramente que há melhores resultados no segundo exercício. No segundo exercício a média dos resultados é de 87 enquanto no primeiro é de 49. No segundo há

9 alunos com nota superior a 50, correspondendo a 90% das respostas, enquanto no primeiro há apenas 7 (54% das respostas). As notas do segundo exercício são mais dispersas porque houve um aluno que tirou apenas 20 enquanto os outros tiraram notas superiores a 50.

Tabela 1 – Resultados dos exercícios diagnósticos

	Exercício 1	Exercício 2
Avaliados	13	10
Nota Max	80	100
Nota Min	20	20
Nota Média	49	87
# notas > 50	7	9

A figura 6 mostra os resultados obtidos nas alíneas a), b) e c) da primeira pergunta do inquérito. Os resultados mostram que há uma avaliação positiva do contributo do GraSMA. Na primeira pergunta (Q1) a média das respostas é 4,6, na segunda (Q2) a média é 4,5 e na terceira (Q3) a média é 4,4.

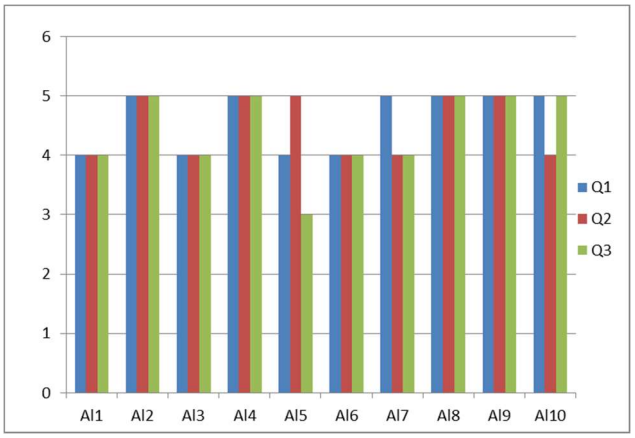

Figura 6 – Resultados dos exercícios diagnósticos

Relativamente à segunda pergunta do questionário, os alunos demoraram menos de 20 minutos contudo em geral não souberam dizer se demoram menos ou mais do que no primeiro exercício, apenas dois disseram que demoraram menos. Na terceira pergunta apenas quatro alunos fizeram uma sugestão. Esses quatro alunos mencionaram que gostariam de ter acesso ao GraSMA.

6 Conclusão

A ferramenta GraSMA permite ilustrar graficamente métodos numéricos destinados a resolução de equações não lineares. Verifica-se que a sua utilização para lecionar os conteúdos associados

a estes métodos constitui uma mais-valia. Os resultados diagnósticos do exercício diagnóstico mostram que há uma melhor compreensão do método por parte dos alunos quando o GraSMA é utilizado.

Em geral a avaliação dos alunos quanto à utilidade de visualização, animação e acompanhamento passo a passo disponibilizada pelo GraSMA foi muito positiva. Demoraram menos tempo na resolução do exercício e são de opinião que será desejável a disponibilização deste software a todos os alunos durante a resolução deste tipo de exercícios. Não apontaram nenhuma melhoria concreta relativa ao software em si, apenas indicaram que o uso do mesmo a uma maior escala seria desejável.

7 Referências

Balsa, C., Alves, L., Pereira, M. J., & Rodrigues, P. J. (2010, Nov 29, Dec 01). *Graphical Simulator of Mathematical Algorithms (GraSMA).* Paper presented at the IASK International Conference Teaching and Learning (TL2010), Seville, Spain.

Balsa, C., Alves, L., Pereira, M. J., Rodrigues, P. J., & Lopes, R. P. (2012, 16 April to 18 April). *Graphical simulation of numerical algorithms: An aproach based on code instrumentation and Java technologies.* Paper presented at the Proceedings of the 4th International Conference on Computer Supported Education (CSEDU 2012), Porto, Portugal.

Berón, M., Henriques, P. R., Pereira, M. J. V., & Uzal, R. (2007). *Static and dynamic strategies to understand C programs by code annotation.* Paper presented at the 1st Int. Workshop on Fondations and Techniques for Open Source Software Certification (OpenCert'07), Braga, Portugal.

Cadenhead, R., & Lemay, L. (2007). *Teach Yourself Java 6 in 21 Days* (5th Edition ed.). Sams Publishing.

Cruz, D. d., Beron, M., Henriques, P. R., & Pereira, M. J. V. (2009). Code Inspection Approaches for Program Visualization. *Acta Electrotechnica et Informatica, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Y*(X).

Eaton, John W., Bateman, D., Hauberg, S. (2008). GNU Octave Manual Version 3. Boston: Network Theory Limited

Ramalho, J. C., & Henriques, P. (2006). *XML & XSL: da teoria à prática*: FCA - Editora Informática.

Shreiner, D. (2009). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1* (7th Edition ed.): Addison-Wesley Professional.