MS211 - Cursão - 2007

Terceiro Exercício Programa

EP de Páscoa

Resumo

Os dois problemas propostos neste EP ilustram muito bem os principais conceitos expostos até agora no curso. O primeiro problema, o estudo das bacias de atração do método de Newton, requer a programação de um método iterativo para solução de sistemas não lineares. Há vários pontos *propositalmente* ambíguos no enunciado. Espera-se que cada grupo faça suas escolhas (por exemplo, que sistema usar, que critério de parada utilizar, que método usar na solução dos sistemas lineares, etc...). Faremos uma outra galeria com os resultados finais... O segundo problema trata do problema das n rainhas. É um problema de enunciado simples, mas de solução muito complexa. Veremos que se trata de um problema NP (*Non-deterministic Polynomial time*), com muitas implicações interessantes.

1 Bacias de atração fractais e o método iterativo Newton

Como vimos, o método de Newton para a solução de sistemas não-lineares $F : R^n \to R^n$ consiste num algorítmo que, dado um ponto inicial $R^n \ni X_0 = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$, retorna uma sequência $\{X_k\} \in R^n$ que converge para a solução do sistema linear,

$$\lim_{k\to\infty} ||X_k - \bar{X}|| \to 0, \quad F(\bar{X}) = 0.$$

A sequência é dada por $X_{k+1} = X_k + S_k$, sendo S_k a solução do sistema linear

$$J(X_k)S_k = -F(X_k).$$

A matriz $J(X_k)$ é a matriz Jacobiana $\nabla F = \left(\frac{\partial f_i}{\partial x_k}\right)^t$ calculada no ponto X_k. Consideremos apenas os casos bi-dimensionais ($n = 2$).

Funções complexas podem dar exemplos bi-dimensionais interessantes para o método de Newton. Consideremos, por exemplo, o caso discutido em aula da função $f : \mathbb{C} \to \mathbb{C}$ dada por $f(z) = z^4 - 1$. Lembrando que um número complexo é composto por dois reais $z = x + iy$, pode-se obter $z^4 = (x^2 - y^2)^2 - 4x^2y^2 + 4ixy(x^2 - y^2)$. As soluções de $f(z) = 0$, portanto, correspondem as soluções do sistema não-linear

$$\begin{cases} (x^2 - y^2)^2 - 4x^2y^2 - 1 = 0, \\ 4xy(x^2 - y^2) = 0. \end{cases}$$

É claro que não é necessário usar o Método de Newton para se resolver esse sistema. Suas soluções correspondem às quatros raizes de $z^4 = 1$: $z = 1$, ou $(x, y) = (1, 0)$; $z = i$, ou $(x, y) = (0, 1)$; $z = -1$, ou $(x, y) = (-1, 0)$; e finalmente $z = -i$, ou $(x, y) = (0, -1)$. O ponto interessante aqui não é encontrar essas raizes, mas determinar para qual raiz convergirá uma sequência gerada pelo método de Newton para um dado ponto inicial $X_0 \in R^2$. A matriz Jacobiana para este problema é

$$J(X_k) = 4 \begin{pmatrix} (x_k^2 - y_k^2)x_k - 2x_k y_k^2 & -(x_k^2 - y_k^2)y_k - 2x_k^2 y_k \\ (x_k^2 - y_k^2)y_k + 2x_k^2 y_k & (x_k^2 - y_k^2)x_k - 2x_k y_k^2 \end{pmatrix}$$

As iterações do método de Newton para este caso serão:

$$\begin{cases} x_{k+1} = x_k + s_k, \\ y_{k+1} = y_k + p_k, \end{cases}$$

sendo que (s_k, p_k) satisfazem

$$J(X_k) \begin{pmatrix} s_k \\ p_k \end{pmatrix} = \begin{pmatrix} -(x_k^2 - y_k^2)^2 + 4x_k^2 y_k^2 + 1 \\ -4x_k y_k (x_k^2 - y_k^2) \end{pmatrix}.$$

Com excessão da origem, onde a matriz Jacobiana J é singular, qualquer outro ponto pode ser usado como condição inicial para o método de Newton. Já sabemos que as seqüências geradas devem convergir para uma das raízes citadas acima.

Pontos para os quais uma seqüência do tipo das geradas pelo método de Newton podem convergir recebem o nome de pontos atratores, ou, simplesmente, atratores. Para o nosso caso específico, há quatro atratores que

correspondem aos seguintes pontos de R^2: $(1,0)$, $(0,1)$, $(-1,0)$ e $(0,-1)$. Uma outra definição útil é o da bacia de atração $B(\bar{X})$ de um ponto atrator $\bar{X}$:

$$B(\bar{X}) = \left\{ X_0 \in R^n \,\middle|\, \lim_{k\to\infty} ||X_k - \bar{X}|| \to 0 \right\},$$

i.e., o conjunto de pontos iniciais X_0 que implicam na convergência da seqüência para o atrator $\bar{X}$. A bacia de atração correspondente as raízes de um sistema não linear nos reserva algumas surpresas. Vamos, neste nosso caso das raizes de $z^4 = 1$, identificar as respectivas bacias de atração das quatro raizes por cores. Caso um ponto pertença à bacia de atração da raiz $(1,0)$, vamos pintá-lo de amarelo. Se pertencer à bacia da raiz $(0,1)$, de azul. Usaremos o vermelho e o verde, respectivamente, para os pontos correspon-

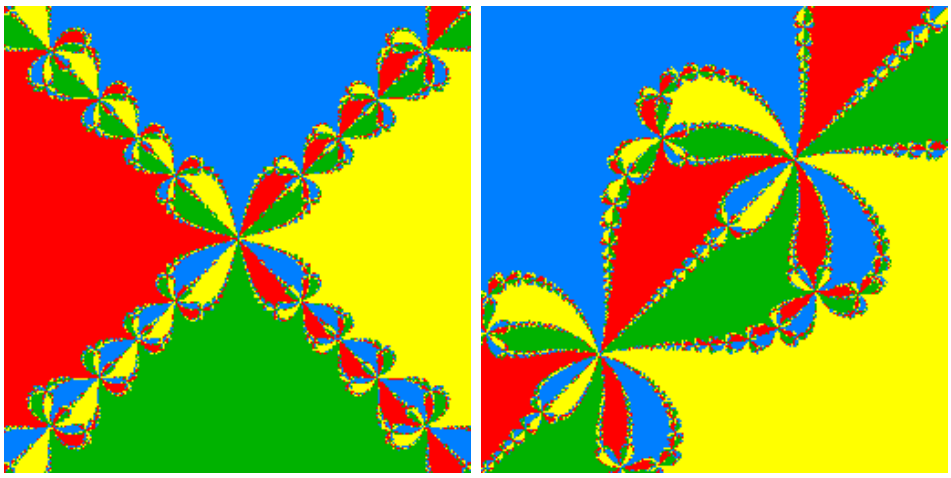

Figura 1: Bacias de atração para o método de Newton aplicado para a função complexa $z^4 = 1$. Esquerda: bacia correspondente à região $-2 \leq x \leq 2$ e $-2 \leq y \leq 2$. Direita: ampliação de um dos detalhes. *(Imagens retiradas do site de Simon Tatham.)*

dentes às raízes $(-1,0)$ e $(0,-1)$. Após diversas simulações, uma para cada condição inicial, obtem-se algo como na Fig. 1. Trata-se de um fractal. A fronteira na região das bissetrizes é extremamente complicada. Pontos arbitrariamente próximos podem levar a seqüências que convergem para raízes diferentes, comportamento diferente do verificado nas regiões distantes das bissetrizes. Se, alem da cor associada à raiz, usarmos uma tonalidade para indicar o número de passos necessários para se atingir a raiz com uma certa

precisão pré-estabelecida, obtem-se algo como na Fig. 2.

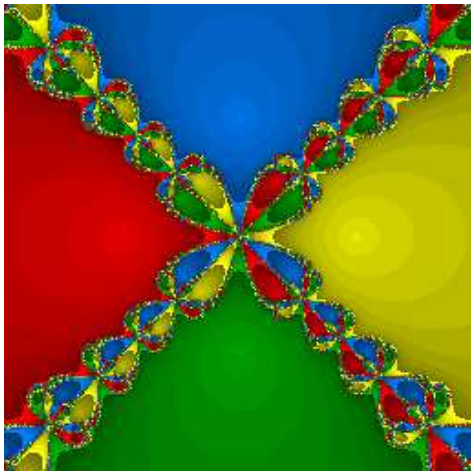

Figura 2: Bacias de atração para o método de Newton aplicado para a função complexa $z^4 = 1$, com indicações do número de passos necessários para se atingir a solução. As regiões mais claras correspondem às de convergência mais rápida. *(A impressão pode comprometer a qualidade da imagem, também retirada do site de Simon Tatham.)*

Item a: O primeiro item deste EP é fazer um programa capaz de gerar bacias de atração para sistemas bi-dimensionais arbitrários. Já adianto que os efeitos mais interessantes são conseguidos com sistemas simples, com poucas raízes (e, portanto, poucas cores). Pode-se usar outros polinômios complexos. As bacias podem ser gravadas diretamente num arquivo PPM, vocês já têm essa tecnologia! Não é necessário criar grandes figuras. Para referência, as mostradas aqui têm 256×256 pixels. Vocês não devem fazer nada que não caiba numa folha A4.

2 O problema das N rainhas

Diz-se que um problema pertence a classe de complexidade P se ele pode ser resolvido em um tempo que depende polinomialmente de algum parâmetro do problema. Pode-se mostrar, por exemplo, que no algorítmo da triangularização de Gauss de uma matriz $n \times n$, o número de operações de ponto-flutuante realizadas é da ordem de n^3. Portanto, supondo que o tempo de

execução de um dado algorítmo é proporcional ao número de operações de ponto-flutuante realizadas, concluí-se que a triangularização de Gauss é um problema da classe P. Uma outra classe de complexidade muito importante é a chamada NP (Non-deterministic Polynomial time). Pertencem a esta classe os problemas *cujas soluções* podem ser verificadas em um tempo polinomial. Um exemplo clássico é o dos subconjuntos de inteiros de soma zero: dado o conjunto de inteiros $I = (-10, 3, 4, 6, 1, 2, -3, 9)$, vemos que o subconjunto $(-10, 4, 6)$, por exemplo, tem soma zero, e isto pode ser verificado em um tempo polinomial. Porém, encontrar todos os suconjuntos de I que somem 0 envolve a construção de arranjos e combinações que podem exigir um número de operações muito grande, um número *superpolinomial*. Um dos grandes problemas em aberto da matemática é o da equivalência entre as classes P e NP. Seria possivel resolver em tempo polinomial os problemas cujas soluções podem ser verificadas em tempo polinomial? Essa é uma das perguntas de US $ 1.000.000,00.

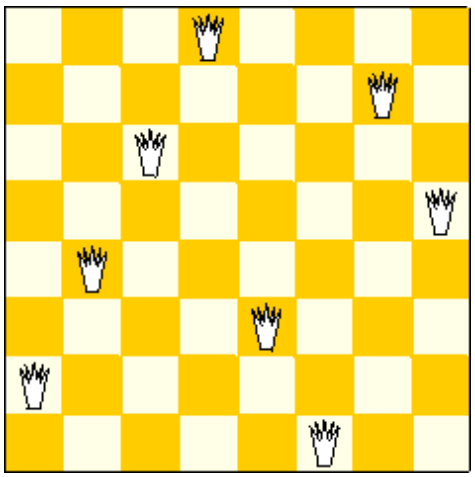

Figura 3: Uma das 92 soluções do problema das 8 rainhas.

O problema das N rainhas pertence à classe NP[1]. Consiste na generalização do clássico problema de xadrez: de quantas maneiras distintas 8 rainhas podem ser dispostas num tabuleiro de xadrez, inicialmente vazio, de maneira que nenhuma delas esteja sob qualquer ataque? A resposta é surpreendente para os que já tentaram encontrar alguma dessas soluções. São 92

[1]Na verdade, é NP-complete e NP-hard.

maneiras diferentes... Uma delas está na Fig. 3. O problema das N rainhas consiste em encontrar o número total de possibilidades de se dispor N rainhas em um tabuleiro $N \times N$, sem que nenhuma esteja sob qualquer ataque. Dada uma solução do problema, pode-se facilmente, em tempo polinomial, verificar que as rainhas estão, de fato, livre de ataques. Encontrar todas as soluções, por outro lado, é muito mais difícil.

Há um vasto material sobre o assunto na Internet, inclusive muitas fontes com os códigos já prontos. Obviamente, espera-se que vocês pensem no problema e proponham suas próprias soluções. Poderão surgir agradáveis surpresas... Um link muito interessante, com um programa em Java para solução interativa é este. Vale a pena conferir.

Item b: Neste segundo item, vocês devem fazer um programa capaz de resolver o problema das N-rainhas. O programa deve responder o número de soluções e permitir também que o usuário opte por ver algumas delas (ou todas). Para isto, pode-se usar, por exemplo, uma representação matricial do tipo

```
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
```

que corresponderia a solução da Fig. 3. Vocês devem construir uma tabela fornecendo, para cada N, o número de de soluções e uma medida da dificuldade (número de iterações necessárias, por exemplo) para encontrar a primeira delas. Para discussão nas aulas exploratórias (não é necessário escrever o código), vocês devem pensar no mesmo problema, porém envolvendo super-rainhas, lembrando que uma super-rainha é uma peça que pode se mover como rainha ou como cavalo.

Agradecimentos

Olívia, Tim e R.E.M.