Diferenças Computacionais entre Métodos de Resolução de Sistemas Lineares

Daniela do Nascimento Rodrigues (RA: 141314)
Davi Juliano Ferreira Alves (RA: 133595)
Dérick Alves de Jesus (RA: 155397)
Felipe Ikejiri Hilário (RA: 143417)

Resumo

Este projeto tem como objetivo investigar as diferenças de tempo e precisão do *software* Octave ao resolver sistemas lineares $Ax = b$, onde $A \in \mathbb{M}_n(\mathbb{R}), x, b \in \mathbb{R}^n$, operando com a inversa A^{-1} e com a fatoração LU de A, levando em conta a dimensão n da matriz e seu número de condicionamento k.

Palavras-chave: Sistemas Lineares, Fatoração LU, Octave, Inversa.

Conteúdo

1 Introdução

A resolução de sistemas lineares é um tema que, até hoje, é muito discutido e que tem suma importância na sociedade atual. Sistemas lineares são essenciais para entender diversas atividades que podem ser executadas sobre um determinado parâmetro e com uma simultaneidade ímpar, pois, com tal ferramenta, pode-se resolver uma quantidade de n problemas sobre um determinado espaço n.

De acordo com [2], a importância de sistemas lineares se dá pelo fato de poder modelar e resolver vários problemas de maneira efetiva. Esses problemas modelados vão dos mais fáceis, de duas incógnitas e duas equações, que podem ser facilmente resolvidos pelos métodos mais básicos, até os mais difíceis, de áreas científicas e tecnológicas, abrangendo uma grande quantidade de variáveis e necessitando de métodos elaborados e mais robustos para as resoluções. Além disso tudo, avançando no contexto de Álgebra Linear, encontra-se muita utilidade no estudo de sistemas lineares quando os mesmos estão intrinsecamente ligados com as transformações lineares, bem como com toda a teoria matricial. Portanto, de acordo com [2], deve-se aprofundar no estudo de sistemas lineares.

Dentro da teoria da resolução destes sistemas, existem diversos métodos eficientes que resolvem-nos. Dentre os quais, há os métodos iterativos, mas estes não serão abordados neste trabalho. No que se segue, iremos resolver sistemas lineares por meio da Fatoração LU e o método da matriz inversa.

O método da matriz inversa consiste em, dada uma matriz inversível, calcular o vetor solução como um produto de uma matriz (a inversa) com um vetor de valor inicial (geralmente denominado por b). Já o método da fatoração LU é mais sofisticado, pois consiste em, dada uma matriz quadrada qualquer, fatorá-la em um produto de uma matriz triangular inferior com uma triangular superior e, se necessário, uma (ou duas) matriz de permutação.

Neste trabalho, utilizaremos do Software Octave para exibir dois algoritmos, sendo um algoritmo o que descreve a Fatoração LU de uma matriz e, subsequentemente, a resolução do sistema linear por meio dessa, e o outro algoritmo aquele que descreve a resolução do sistema linear por meio do método da inversa da matriz. Dentre esses dois algoritmos, será feito um estudo sobre qual é mais computacionalmente viável, testando ambos em matrizes específicas.

2 Fundamentação Teórica

Primeiramente, será definido o que é um sistema linear e toda a base fundamental para que possamos avançar com os testes computacionais.

Definição 1 *(Sistema Linear) Um sistema de m equações lineares com n incógnitas ($m, n \geq 1$) é um conjunto de m equações lineares, cada uma delas com n incógnitas, consideradas simultaneamente, um sistema linear pode se apresentar da seguinte maneira:*

$$S : \begin{cases} \alpha_{11}x_1 + ... + \alpha_{1n}x_n = \beta_1 \\ \alpha_{21}x_1 + ... + \alpha_{2n}x_n = \beta_2 \\ ..... \\ \alpha_{m1}x_1 + ... + \alpha_{mn}x_n = \beta_m \end{cases}$$

Ou pode simplesmente ser apresentada da forma matricial: $Ax = b$, com A sendo a matriz dos coeficientes α_{mn}, b sendo o vetor de coeficientes β_m e x sendo o vetor de solução.

Definição 2 *(Matriz Inversa) Uma matriz A de ordem n se diz inversível se, e somente se, existe uma matriz B, também de ordem n tal que satisfaz:*

$$A \cdot B = B \cdot A = I_n, \text{ sendo } I_n, \text{ a matriz identidade de ordem } n.$$

Nesse caso, inversa de A é única e será representada por A^{-1}.

No que se segue, estamos usando espaços vetoriais sobre $\mathbb{R}$.

Definição 3 *(Norma) Dado um espaço vetorial X, uma função $\|\cdot\| : X \to \mathbb{R}^+$ é chamada de norma se, para quaisquer $x, y \in X$ e todo $\alpha \in \mathbb{R}$ satisfazer*

i) $\|x\| = 0 \Leftrightarrow x = 0$

ii) $\|\alpha x\| = \|\alpha\| \cdot \|x\|$

iii) $\|x + y\| \leq \|x\| + \|y\|$

Definição 4 *(Norma Matricial) Seja $\mathbb{R}^{n\times n}$ o espaço das matrizes $n \times n$. Uma norma matricial em $\mathbb{R}^{n\times n}$ é a aplicação*

$$\|\cdot\| : \mathbb{R}^{n\times n} \to \mathbb{R}$$

que satisfaz

i) $\|A\| \geq 0 \ \forall A \in \mathbb{R}^{n\times n}$

ii) $\|\lambda A\| = |\lambda| \cdot \|A\| \ \forall A \in \mathbb{R}^{n\times n}$ *e* $\forall \lambda \in \mathbb{R}$

iii) $\|A + B\| \leq \|A\| + \|B\| \ \forall A, B \in \mathbb{R}^{n\times n}$

iv) $\|AB\| \leq \|A\| \cdot \|B\| \ \forall A, B \in \mathbb{R}^{n\times n}$

Note que é possível definir a norma de uma matriz usando uma norma vetorial. Basta tomar $\|A\| := \max_{x\neq 0} \frac{\|Ax\|}{\|x\|}$. Neste caso dizemos que $\|\cdot\|$ é uma **norma matricial induzida**

No que se segue, estamos usando uma norma matricial induzida para definir o número de condicionamento.

Definição 5 *(Número de Condicionamento) O número de condicionamento k de uma matriz não-singular A é*

$$k(A) := \|A\| \cdot \|A^{-1}\| = \frac{\max\limits_{\|x\|=1} \|Ax\|}{\min\limits_{\|x\|=1} \|Ax\|}$$

O número de condicionamento nos diz sobre o erro que se propaga em um certo sistema linear.

Um número de condicionamento de uma matriz identidade $I_n = 1$, independente da norma, pois esta é sempre induzida. Entretanto, de forma geral, o número de condicionamento de uma matriz varia conforme a norma utilizada. O número de condição de uma matriz inversível é maior ou igual a 1, entretanto, se uma matriz A é singular definimos que $k(A) = \infty$.

Agora, apontamos um resultado importante sobre o número de condição e perturbações. Dada a equação $Ax = b$, sabemos que uma perturbação na solução x, chamada δx e no vetor b, chamado de δb estão relacionadas na seguinte equação.

$$\boxed{\frac{\|\delta x\|}{\|x\|} = K(A)\frac{\|\delta b\|}{\|b\|}}$$

Iremos, agora, definir a inversa de matrizes e a fatoração LU.

Método 1 *(Inversa) Sejam $b \in \mathbb{R}^n$ e $A \in \mathbb{R}^{n\times n}$ uma matriz inversível, ou seja, existe uma A^{-1}, tal que $AA^{-1} = A^{-1}A = I_n$. Dado o sistema $Ax = b$ temos*

$$Ax = b \Rightarrow A^{-1}Ax = A^{-1}b \Rightarrow x = A^{-1}b$$

Portanto, x pode ser obtido por meio do produto $A^{-1}b$.

O método da inversa de matrizes é extremamente simples e elegante na teoria. Entretanto, o método para obter a matriz inversa é computacionalmente custoso. Além de obter-se a matriz inversa, deve-se fazer o produto da mesma pelo vetor b, e isso torna mais robusta a complexidade no algoritmo.

Antes de caracterizar o método por fatoração LU precisamos definir os seguintes termos.

Definição 6 *(Matriz de Permutação) $P \in \mathbb{R}^{n\times n}$ é dita matriz de permutação quando é uma matriz quadrada obtida ao permutar as linhas da matriz identidade. Esta tem o efeito de gerar uma permutação dos elementos de um vetor ou entre linhas ou colunas de uma matriz.*

Método 2 *(Fatoração LU) Sabemos que toda matriz A quadrada pode ser decomposta por um produto de matrizes triangulares e uma matriz de permutação. Seja A uma matriz quadrada, temos que existe uma fatoração LU. Ou seja, dada $A \in \mathbb{R}^{n\times n}$, existem $L, U, P \in \mathbb{R}^{n\times n}$ com L triangular inferior, U triangular superior e P uma matriz de permutação com $PA = LU$. Seja o sistema linear dado por $Ax = b$ temos:*

$$Ax = b \Rightarrow (LU)x = Pb \Rightarrow Ly = Pb \text{ e } Ux = y$$

Ou seja, a solução $x \in \mathbb{R}^n$ pode ser obtida pelo seguinte sistema M de matrizes:

$$M : \begin{cases} Ly = Pb \\ Ux = y \end{cases}$$

2.1 Software Octave

De acordo com [1], o GNU Octave é uma linguagem de alto nível, destinada principalmente a cálculos numéricos. Ele fornece uma interface de linha de comando conveniente para resolver problemas lineares e não lineares numericamente e para realizar outros experimentos numéricos, usando uma linguagem que é compatível com o Matlab. Ele também pode ser usado como uma linguagem orientada a lote.

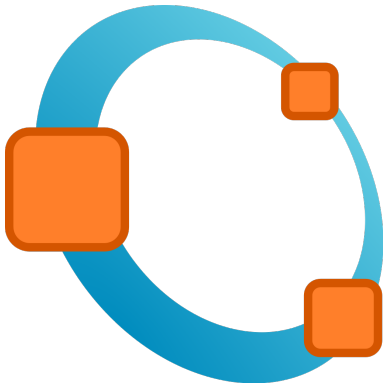

Figura 1: Logo da GNU Octave (Fonte: [1])

De acordo ainda com [1], o Octave é um software gratuito redistribuível. Pode-se redistribuir e/ou modificar sobre os termos da GNU General Public License que foi publicado pela Free Software Foundation.

3 Desenvolvimento

Definidos os métodos de resolução dos sistemas lineares e o software que será utilizado para executar os problemas, definiremos o algoritmo implementado que analisará a complexidade. Ambos os códigos podem ser encontrados no drive `https://drive.google.com/drive/folders/1IVqLkH46craMVFoUQjS3UGSdMlKEpW5t?usp=sharing`.

3.1 Nosso método

3.1.1 Uma estranheza

A executar os comandos com uma mesma matriz qualquer A percebemos que, **às vezes**, executar Inv(A) (lê-se inversa de A) primeiro tornava a fatoração LU extremamente rápida e fatorar em LU primeiro acelerava o processo da inversa.

Conjecturamos que isso se deve ao fato de que o Octave calcula a inversa fazendo uma fatoração LU e resolvendo n sistemas lineares do tipo $LU = e_i$ onde e_i é o vetor que a i-ésima coordenada vale um e as outras são nulas. Se tal fatoração for guardada, isso justificaria o processo acelerado do cálculo da LU e se o programa usar a LU já definida, isso justifica a aceleração do cálculo da inversa.

Mas também pode ser que estes testes tenham tido tais resultado por outro motivo, pois em testes posteriores isto não tornou a acontecer. Dado que aconteceu em mais de um computador e já tivemos todo este trabalho, continuaremos com tal método.

3.1.2 Duas formas de comparação

Para contornar o problema supracitado, resolvemos definir matrizes aleatórias distintas. Ou seja, comparar os tempos e erros dos métodos em conjuntos de matrizes distintos. Mas com mesma ordem e mesmo número de condição para uma comparação fidedigna.

Assim, iremos definir dois conjuntos de vetores e dois conjuntos de matrizes. Serão $\{b_i \in \mathbb{R}^n ; 1 \leq i \leq k\}$ e $\{b'_i \in \mathbb{R}^n ; 1 \leq i \leq k\}$ e $\{M_i \in \mathbb{R}^{n\times n}; 1 \leq i \leq k\}$ e $\{M'_i \in \mathbb{R}^{n\times n}; 1 \leq i \leq k\}$ com $K(M_i) = K(M'_i) = c \; \forall i$. Resolveremos as equações $M_i x = b_i$ com um método e $M'_i x = b'$ com outro.

Também iremos comparar ambos os métodos com o mesmo conjunto de matrizes, caso o problema apontado na seção 3.1.1 seja um acaso.

Portanto, dados um conjunto de vetores $\{b_i \in \mathbb{R}^n; 1 \leq i \leq k\}$ e um conjunto de matrizes aleatórias $\{M_i \in \mathbb{R}^{n\times n}; 1 \leq i \leq k\}$ com $K(M_i)$ e ordem de A pré definidos, resolveremos as equações $M_i x = b_i$ com ambos os métodos.

3.1.3 Controlando o número de condição

Para tanto, definimos duas matrizes aleatórias de ordem $n \times n$ A e B e em seguida fizemos a fatoração QR de ambas, definindo assim, as matrizes Q_A, Q_B, R_A, R_B com $Q_A R_A = A$ e $Q_B R_B = B$.

Após, ao pegar um número de condição inteiro qualquer m, definimos uma matriz diagonal D com valores em $\mathbb{Z}^+$, que variam de 1 a $m - 1$.

Deste modo, ao definir a matriz $M = Q_A D Q_B^T$, temos uma matriz quadrada invertível de ordem n e número de condição $= m$.

Neste caso, torna-se simples definir matrizes bem ou mal condicionadas, bem como gerar matrizes simétricas, basta tomar $A = B$.

3.1.4 As funções

Para o método da Fatoração LU, utilizamos de funções dadas pelo GNU Octave para determinar duas funções. Elaboramos funções que geram matrizes dados um número de condicionamento c e uma dimensão n e, em seguida, elabora e resolve o sistema pelo método em questão. Assim, as funções geram o tempo e o erro com a fatoração LU (temperrolu) e com a inversa (temperroinv).:

```
function [t1,e1] = temperroinv(c, n, A, B, X);
[Qa,Ra] = qr(A);
[Qb,Rb] = qr(B);
d = [c randi([1 c-1], 1, n-2) 1];
M = Qa*diag(d)*Qb;
b = M*X;
tic;
N = inv(M);
x = N*b;
t1 = toc;
e1 = norm(X-x);
endfunction
```

```
function [t2,e2] = temperrolu(c, n, A, B, X)
[Qa,Ra] = qr(A);
[Qb,Rb] = qr(B);
d = [c randi([1 c-1], 1, n-2) 1];
M = Qa*diag(d)*Qb;
b= M*X;
tic;
[L,U,P] = lu(M);
y = L\P*b;
x = U\y;
t2 = toc;
e2 = norm(X-x);
endfunction
```

De acordo com [1], ao usar a barra invertida "\ " (mldivide), realizamos o melhor procedimento para resolver o sistema. Então, a máquina identifica qual o formato da matriz (se é quadrada, triangular, definida positiva, hermitiana ou Hessenberg) e executa o método adequado.

Ou seja, ao fazer a fatoração LU e em seguida $L\backslash b$, o programa identifica que L é triangular (assim como U) e resolve diretamente.

3.2 Aplicando ambos os métodos nas mesmas matrizes

Para tanto, determinamos previamente o número de condição c, a dimensão n e o número de iterações k. Em seguida, aplicamos o seguinte código:

```
tempo = 1:200;
erro = 1:200;
tempo1 = 1:200;
erro1 = 1:200;
m = 50;
n = 50;
j = 1;
c = m;


for i = 1:200;
  A = rand(n);
  B = rand(n);
  X = rand(n,1);
  [Qa,Ra] = qr(A);
  [Qb,Rb] = qr(B);
  d = [c randi([1 c-1], 1, n-2) 1];
  M = Qa*diag(d)*Qb;

  [tempo(i), erro(i)] = temperroinv1(c, n, A, B, X, M);
  [tempo1(j), erro1(j)] = temperrolu1(m, n, A, B, X, M);

  i++;
  j++;
endfor

devpd_lut= std(tempo1);
devpd_lue = std(erro1);
med_lut = mean(tempo1);
med_lue = mean(erro1);
devpd_invt = std(tempo);
devpd_inve = std(erro);
med_invt = mean(tempo);
med_inve = mean(erro);

disp("Desvio padrao do tempo do LU:"), disp(devpd_lut)
disp("\nDesvio padrao do erro do LU:"), disp(devpd_lue)
disp("\nMedia do tempo do LU:"), disp(med_lut)
disp("\nMedia do tempo do LU:"), disp(med_lue)
disp("\nDesvio padrao do tempo do Inv:"), disp(devpd_invt)
disp("\nDesvio padrao do erro do Inv:"), disp(devpd_inve)
disp("\nMedia do tempo do Inv:"), disp(med_invt)
disp("\nMedia do erro do Inv:"), disp(med_inve)
```

No caso, neste exemplo seriam 200 testes de matrizes de ordem 50 e condicionamento = 50. Esse código é aplicado para a mesma matriz M que passou pelo processo já citado em 3.1.3. Como queremos verificar se 3.1.1 é realmente um acaso, fizemos os testes usando do código acima com relação ao código

das matrizes M distintas e inicializadas dentro de cada função. Se gerar uma certa discrepância, a conclusão gera que 3.1.1 não é só um acaso, mas há algo na programação do Octave que permite o aprendizado contínuo do software de acordo com os comandos passados.

4 Resultados

Depois de definir as funções temperroinv e temperrolu nós modificamos alguns parâmetros, inclusive a solução, e a matriz M.

Para nossa análise trouxemos as médias do tempo e erro ($\bar{x}$ e σ. Buscamos gerar um bom número de iterações no For, mas por limites da máquina precisamos usar valores mais limitados. Nossa maior empreitada no quesito número de iterações foi ao analisar os tempos e a norma dos resíduos em ambos os métodos. Usamos 100.000 iterações. Os dados desta tentativa estão na tabela abaixo. No caso usamos as mesmas matrizes para os dois métodos.

Tabela 1: Teste dos resíduos

M = randomizada		$K(M) = 500$	$n = 100$	$IT = 10^5$
	$\bar{x}$(tempo)	σ(tempo)	$\bar{x}$(resíduo)	σ(resíduo)
LU	$4.931 \cdot 10^{-4}$	$4.556 \cdot 10^{-5}$	$9.088 \cdot 10^{-14}$	$6.489 \cdot 10^{-14}$
Inversa	$1.3863 \cdot 10^{-3}$	$4.7255 \cdot 10^{-4}$	$2.1530 \cdot 10^{-13}$	$6.4696 \cdot 10^{-14}$

4.1 Mesmas matrizes métodos diferentes

Usando o For definido acima conseguimos os seguintes resultados:

Tabela 2: Teste com matrizes de ordem 50

M = randomizada		$K(M) = 100$	$n = 50$	$IT = 10000$
	$\bar{x}$(tempo)	σ(tempo)	$\bar{x}$(erro)	σ(erro)
LU	$1.4967 \cdot 10^{-4}$	$3.8377 \cdot 10^{-5}$	$9.7186 \cdot 10^{-15}$	$4.9728 \cdot 10^{-15}$
Inversa	$3.9725 \cdot 10^{-4}$	$7.8081 \cdot 10^{-5}$	$8.0672 \cdot 10^{-15}$	$2.6349 \cdot 10^{-15}$

4.2 Matrizes diferentes, mesmas condições

Aqui apresentaremos testes com matrizes randômicas distintas na fatoração LU e no cálculo da inversa. Isso foi feito definindo as matrizes A e B dentro dos ambientes function definidos para os métodos (temperrolu e temperroinv).

4.3 Estimulando o erro

Dado que estamos usando uma matriz $M = Q_A D Q_B$ com $D(1,1) = \max D = K(M)$ e $D(n,n) = \min D = 1$ podemos notar que o vetor $v_1 = Q_B(1, 1:n)'$ e $v_n = Q_B(n, 1:n)'$ são os vetores com máxima e mínima distorção pela matriz M; pois:

$$Mv_1 = Q_A D Q_B Q_B(1, 1:n)' = Q_A D e_1 = K(M) Q_A e1 \qquad \|K(M) Q_A e1\| = K(M)$$

E de modo análogo vemos que $\|Mv_n\| = 1$.

Deste modo podemos definir $x = v_1 + v_n$ Como resultado, obtivemos a seguinte tabela que descreve os resultados:

```
function [t1,e1] = temperroinvmod(c, n, A, B);
[Qa,Ra] = qr(A);
[Qb,Rb] = qr(B);
d = [c randi([1 c-1], 1, n-2) 1];
X = Qb(1,1:n)'+Qb(n,1:n)';
M = Qa*diag(d)*Qb;
b = M*X;
tic;
N = inv(M);
x = N*b;
t1 = toc;
e1 = norm(X-x);
endfunction
```

Tabela 3: Primeiro teste

M = randomizada		$K(M) = 1.000$	$n = 1.000$	$IT = 500$
	$\bar{x}$(tempo)	σ(tempo)	$\bar{x}$(erro)	σ(erro)
LU	0.1022	0.03613	$4.114 \cdot 10^{-13}$	$1.6553 \cdot 10^{-13}$
Inversa	0.1042	0.0467	$6.970 \cdot 10^{-13}$	$1.7977 \cdot 10^{-13}$

Tabela 4: Segundo teste

M = randomizada		$K(M) = 10^6$	$n = 500$	$IT = 500$
	$\bar{x}$(tempo)	σ(tempo)	$\bar{x}$(erro)	σ(erro)
LU	0.01505	$4.063 \cdot 10^{-3}$	$1.479 \cdot 10^{-10}$	$1.156 \cdot 10^{-10}$
Inversa	0.0148	$8.499 \cdot 10^{-3}$	$2.783 \cdot 10^{-10}$	$4.0594 \cdot 10^{-11}$

4.4 Breve relato

Houve uma tentativa de rodar uma matriz randômica M com condição $K(M) = 100$, tamanho $n = 10.000$ por 50 iterações em um computador com a capacidade de processamento dada por um processador Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz de décima geração de processadores com capacidade de RAM dada por 16 GB, e o computador, em 20 minutos, foi capaz de realizar apenas 4 iterações das 50.

5 Conclusão

A necessidade do nosso trabalho se deu pela formalização dos conceitos computacionais de dois métodos resolutivos de sistemas lineares, avaliando-os e fazendo, analiticamente, testes para verificar a complexidade algorítmica de ambos. Os métodos testados foram o da decomposição LU e o método de cálculo da inversa.

Em geral, temos que, para valores razoavelmente pequenos de dimensão da matriz, os valores da LU são superiores, até mesmo em diferentes ordens de grandeza, para uma quantidade alta de iterações. Para matrizes com dimensão relativamente grande, temos que algumas vezes a LU possui uma complexidade inferior, entretanto, o erro da LU é menor em ambos os testes. Também, pelo breve relato, teve-se uma maior compreensão dos problemas matriciais com relação ao processamento real de dados, sendo o problema muito mais complexo computacionalmente do que havíamos compreendido. Ou seja, podemos falar que a dimensão $n = 10.000$ é quase uma cota superior para problemas de LU e Inversa para o computador citado.

Nas entrelinhas das análises matriciais, houve um pensamento sobre qual fator gerado seria aquele que predominaria sobre a complexidade do algoritmo, e, de acordo com os testes, o fator predominante do tempo é a dimensão das matrizes. O número de condição, de fato, é algo que alterou significantemente mais no desvio padrão do que no tempo. Portanto, a conclusão dos fatores que predominaram sobre a análise matricial é de que a dimensão das matrizes controla a quantidade de operações que o Octave necessita fazer, entretanto, o desvio padrão de cada conta é gerada, em grande parte, pelo número de condição.

Em geral, alcançamos nosso propósito de testar os métodos computacionais dados pelas funções da Sessão 3.1.4 e obtivemos um relativo domínio da fatoração LU sobre o método das Inversas. Entretanto, em um teste com $n = 500$, houve um domínio do método das Inversas. Portanto, a conclusão demonstrada é a de que ambos os métodos são eficazes e eficientes, entretanto, sobre uma determinada ordem de grandeza, ou até mesmo sobre alguns poucos milissegundos, o método LU foi superior ao método das Inversas nos quesitos de tempo e de desvio padrão das soluções.

Referências

[1] John W. Eaton, David Bateman, Søren Hauberg, and Rik Wehbring. *GNU Octave version 6.1.0 manual: a high-level interactive language for numerical computations*, 2020.

[2] Ana Eliza Gonçalves Ferreira et al. A importância dos sistemas lineares no ensino médio e a contribuição para a matemática e suas aplicações. 2013.