Um Modelo Matemático para Análise de Batalhas Históricas Considerando Munição Limitada

Thales Luis Rodrigues Sabino

9 de dezembro de 2015

Relatório técnico do trabalho realizado na disciplina *Introdução a Modelagem Matemática (2015-3)* do **Programa de Pós-Graduação em Modelagem Computacional (PGMC) da Universidade Federal de Juiz de Fora (UFJF)**.

Professor: Rodrigo Weber dos Santos

Resumo

Este trabalho tem como objetivo propor um modelo matemático para representação de conflitos armados no contexto de batalhas históricas. O modelo proposto foi baseado nas equações de Lanchester porém a munição na linha de frente é limitada e deve ser reposta, devendo ser levada da retaguarda até a linha de frente pelo processo de difusão. Foi realizada uma simulação de uma batalha particular e também foi feita uma análise pela variação de diversos parâmetros que compõe o modelo, de forma a extrair informações sobre as condições que podem levar um exército a vitória ou derrota.

1 Introdução

Conflitos são parte da história da humanidade, tanto que encontram-se representados através da arte. Exemplo disso é a relíquia descoberta no século XIX na antiga Suméria, a cidade-estado de *Lagash* possui ruínas contendo registros dos vários aspectos da guerra travada com uma cidade-estado vizinha conhecida como *Umma*. Nas ruínas foram encontradas representações de soldados sendo devorados por abutres. A relíquia, hoje, encontra-se exposta no Museu do Louvre, em Paris [3].

Conflitos, armados ou não, sempre fizeram parte da história da humanidade. O orçamento dedicado a defesa de vários países corresponde a uma parte significativa do PIB gerado, tornando o aspecto bélico de extrema importância para a defesa e segurança de um país.

É evidente que uma parte significa do dinheiro investido nas forças militares é dedicada ao estudo de situações de combate, estratégias e maneiras eficientes de vencer as batalhas. O termo eficiente, nesse contexto, significa derrotar o inimigo sofrendo um número mínimo de baixas.

Não faz parte do escopo deste trabalho a realização de uma discussão sobre os aspectos positivos, negativos e impactos sociais causados por conflitos, somente uma análise de um modelo matemático que tenta capturar os aspectos de um conflito armado entre dois exércitos homogêneos.

1.1 Modelos de Combate do Tipo Lanchester

A análise de combates do ponto de vista matemático foi inicialmente desenvolvido por Frederick W. Lanchester (1914) (Figura 1), um engenheiro britânico que que desenvolveu a teoria dos combates baseado em conflitos aéreos da Primeira Guerra Mundial na tentativa de explicar porque a concentração de forças era útil em combates modernos. As *leis* de Lanchester são ensinadas e usadas em todos os colégios militares do mundo.

Figura 1: Frederick Willian Lanchester

Existem dois tipos de modelos a serem considerados, **homogêneos** e **heterogêneos**. No modelo homogêneo, um único escalar é utilizado para representar o poder de combate de uma unidade e ambos os lados do conflito são considerados equivalentes quanto a eficiência em combate. O modelo homogêneo pode ser considerado de cunho *acadêmico* e é aplicado na análise e revisão de *batalhas históricos*, não sendo considerado um bom modelo para descrição de *combates modernos* [1].

1.2 Batalhas Históricas

Não foi considerada nenhuma batalha específica no desenvolvimento deste trabalho, somente a forma como as batalhas eram efetuadas.

Antes da Primeira Guerra Mundial, conhecida como guerra das trincheiras, as batalhas entre dois países que estavam em guerra tinham um formato bem diferente dos combates modernos. As batalhas, normalmente, tinham um local e data pré-definidos, onde das duas forças se encontravam e faziam a disputa. A força que conseguisse incapacitar a maior parte da força do adversário era considerada a vencedora. Nesse tipo de conflito, os dois exércitos praticamente se alinhavam um de frente para o outro, em uma formação, normalmente em linha ou retangular até que fosse ordenado o ataque. A linha de frente de cada exército ia sendo substituída a medida que os soldados iam ficando sem munição ou era atingidos.

A **Batalha de Gettysburg** (Figura 3) é uma batalha famosa da Guerra Civil Americana conhecida por ser a batalha com maior número de vítimas de tal guerra. O **Assalto de Pickett** (2) foi o famoso assalto da infantaria Confederada ordenada pelo Gal. Robert E. Lee, em 3 de Julho de 1863, durante a Batalha de

Gettsburg. No entanto, esse assalto foi repelido com um grande número de baixas e decisão a favor da União.

Figura 2: Edwin Forbes. O Assalto de Pickett de uma posição na linha dos Confederados olhando para as linhas da União.

Figura 3: Thrue de Thulstrup. Batalha de Gettysburg 1887.

As **Equações de Lanchester** foram utilizas para modelar esse conflito e, portanto, representam um modelo válido para o estudo de conflitos clássicos.

O modelo matemático apresentado neste trabalho foi ligeiramente baseado no modelo de forças homogêneas de Lanchester, porém foi considerado que a munição disponível na linha de frente é limitada e deve ser reposta. Para modelar a reposição da munição na linha de frente foi utilizado o processo de difusão, onde a munição deve sair da retaguarda até chegar na linha de frente onde pode ser utilizada contra o exército inimigo.

2 Objetivo

O objetivo deste trabalho é o desenvolvimento e análise de um modelo matemático capaz de representar o embate entre duas forças homogêneas, porém com munição disponível limitada na linha de frente.

O modelo proposto foi baseado no modelo de Lanchester porém foram acrescentados outros parâmetros de forma a a permitir um controle mais refinado do desenrolar da batalha.

3 Modelo Matemático de Batalhas Históricas

Considere o embate entre duas forças homogêneas no contexto de uma batalha histórica, onde os exércitos se alinham um de frente para o outro em formação e mantém a posição atirando e substituindo a linha de frente a medida que esta é atingida pelo inimigo. Para o desenvolvimento do modelo, será considerado que a força dos exércitos é medida pelo número de soldados presentes em campo de um determinado instante de tempo. E (o exército sob comando) e a I (o exército inimigo) são as funções que representam essas quantidades. É desejável saber em que condições o exército E irá vencer ou perder a batalha. Outras perguntas se deseja responder são: Como o número de soldados irá cair com o tempo na batalha? Quantos sobreviventes o vencedor terá? O quanto a eficiência do exército é importante para garantir a vitória na batalha?

3.1 Modelagem de Formação de Batalha

Seja $E(t)$ e $I(t)$ as respectivas forças do exército e do inimigo. A taxa de variação do exército pode ser modelada da seguinte forma:

$$\frac{dE(t)}{dt} = -k_1 \alpha E(t) \beta I(t) \tag{1}$$

onde $0 < k_1 \leq 1$ é a constante que mede a eficiência do inimigo e $0 < \alpha, \beta \leq 1$ é a porcentagem do exército que irá compor as linhas de frente do exército e do inimigo, respectivamente, em um dado momento. A Equação 1 pode ser resumida como a modelagem do embate entre as duas linhas de frente.

De maneira análoga, a taxa com que o exército inimigo é cai é modelada como o encontro das duas linhas de frente:

$$\frac{dI(t)}{dt} = -k_2 \rho \mu(t, L) \alpha E(t) \beta I(t) \tag{2}$$

onde $0 < k_2 \leq 1$ é a constante que mede a eficiência do exército em relação ao inimigo $0 \leq \rho \leq 1$ é a taxa de disparos que o exército consegue efetuar e $\mu(t, L)$ é a quantidade de munição disponível na linha de frente no instante de tempo t. $L > 0$ é a extensão da formação. Quanto maior for L mais espaçado estarão os soldados relativos a linha de frente e mais tempo levará para a munição chegar na linha de frente.

A difusão da munição é modelada pela equação de difusão em 1D:

$$\frac{\partial \mu(t,x)}{\partial t} = D\frac{\partial^2 \mu(t,x)}{\partial x^2} \tag{3}$$

onde $0 < x \leq L$ e $D > 0$ é o coeficiente de difusão da munição pelo exército.

3.2 Condições Iniciais e de Contorno

Após a escolha dos parâmetros da batalha, as condições iniciais dizem respeito somente ao contingente inicial de cada exército e a quantidade de munição presente na retaguarda do exército no tempo $t = 0$:

$$\begin{aligned} E(0) &= \Upsilon \\ I(0) &= \Phi \\ \mu(0,0) &= \Delta \end{aligned} \tag{4}$$

onde $\Upsilon, \Phi, \Delta > 0$.

Para as condições de contorno é importante notar que na retaguarda ($x = 0$) não há fluxo de munição, porém na linha de frente a munição é utilizada a uma taxa proporcional a taxa de disparos que o exército consegue efetuar, logo:

$$\begin{aligned} \nabla \mu(t,L) \cdot \eta(x) &= -\frac{\rho \mu(t,L)}{D\alpha E(t)} \\ \nabla \mu(t,0) \cdot \eta(x) &= 0 \end{aligned} \tag{5}$$

3.2.1 Condição para Vitória

O exército vencedor de uma batalha é aquele que permanece com um número maior de soldados em campo quando a contagem de um dos lados chega a um determinado valor. No modelo proposto, para considerar que um dos exércitos perdeu a batalha, uma fração do número original de soldados deve ser atingida. Um exército é considerado perdedor se essa fração for atingida, declarando rendição ao adversário. Para implementar essa condição um valor $0 < \gamma \leq 1$ é escolhido de forma que se $E < \gamma E(t)$ considera-se derrota para o inimigo e se $I < \gamma I(t)$ considera-se vitória sobre o inimigo para algum instante de tempo $t > 0$

3.3 Discretização por Diferenças Finitas

Para solucionar o modelo, foi escolhido o método de diferenças finitas explicito para solução numérica das equações diferenciais que fazem parte do modelo.

Para as Equações 1 e 2 foi escolhido o esquema para frente no tempo, logo a discretização por diferenças finitas para a Equação 1 é dada por:

$$\frac{dE}{dt} = \frac{E^{t+1} - E^t}{\Delta t} = -k_1 \alpha E(t) \beta I(t) \tag{6}$$

isolando o termo E^{t+1} tem-se:

$$E^{t+1} = E^t - \Delta t \left[k_1 \alpha E(t) \beta I(t)\right] \tag{7}$$

De forma análoga, a discretização da Equação 2 depois de isolado o termo I^{t+i} é:

$$I^{t+1} = I^t - \Delta t \left[k_2 \rho \mu(t, L) \alpha E(t) \beta I(t)\right] \tag{8}$$

Para a discretização da equação de difusão foi escolhido o esquema de diferenças centradas no espaço e para frente no tempo, logo:

$$\frac{\partial \mu}{\partial t} = \frac{\mu_{t+1}^i - \mu_i^t}{\Delta t} = D\left(\frac{\mu_{i-1}^t - 2\mu_i^t + \mu_{i+2}^t}{(\Delta x)^2}\right) \tag{9}$$

isolando o termo μ_{i+1}^t tem-se que:

$$\mu_{i+1}^t = \mu_i^t + \frac{D\Delta t}{(\Delta x)^2}\left(\mu_{i-1}^t - 2\mu_i^t + \mu_{i+2}^t\right) \tag{10}$$

onde $\frac{D\Delta t}{(\Delta x)^2} < \frac{1}{2}$ é a condição CFL do modelo. Portanto, os valores de D, Δt e Δx devem ser escolhidos de forma adequada para não causar instabilidade na solução numérica do modelo.

3.4 Dados de Entrada

Nesta seção serão apresentados os dados de entradas escolhidos como dados de entradas.

Dados de Entrada			
Υ	500	Número inicial de soldados no exército	$\Upsilon > 0$
Φ	500	Número inicial de soldados inimigos	$\Phi > 0$
Δ	1000	Quantidade inicial de munição na retaguarda	$\Delta > 0$
γ	0.01	Fração que deve sobrar no campo de batalha para considerar derrota	$0 < \gamma \leq 1$
k_2	0.01	Constante que mede o poder de ataque e a eficiência do exército	$0 < k_2 \leq 1$
k_1	0.03	Constante que mede o poder de ataque e a eficiência do inimigo	$0 < k_1 \leq 1$
ρ	0.05	Medida da taxa de disparos feitas pelo exército por unidade de tempo	$0 < \rho \leq 1$
D	0.8	Coeficiente de difusão da munição pelo exército	$D > 0$
L	7	Tamanho da formação do exército	$L > 0$
α	0.1	Porcentagem do exército que ocupa a linha de frente	$0 < \alpha \leq 1$
β	0.1	Porcentagem do exército inimigo que ocupa a linha de frente	$0 < \beta \leq 1$
Δt	0.07	Discretização temporal	$\Delta t > 0$
Δx	0.6	Discretização espacial	$\Delta x > 0$

Tabela 1: Lista dos valores escolhidos para condição inicial, de contorno e parâmetros do modelo.

A Tabela 1 lista todos os valores escolhidos para resolução numérica do modelo das batalhas. Note que a condição CFL $\left(\frac{D\Delta t}{(\Delta x)^2} < 0.15 < 1/2\right)$ foi atendida, logo, para os parâmetros escolhidos, o método explícito não apresenta instabilidades.

4 Análise do Modelo

O modelo de batalhas proposto, apresentado na Seção 3.1, é composto por três equações. Dessas, duas são compostas exclusivamente por termos de reação (Equações 1 e 2) e uma é composta exclusivamente por um termo de difusão (Equação 3.

Nesta seção será feita uma análise dos termos de reação do modelo proposto a fim de entender, de forma qualitativa, o comportamento das soluções.

4.1 Pontos de Equilíbrio e *Nullclines*

Os pontos de equilíbrio para os termos de reação do modelo de batalhas são obtidos ao fazer $\frac{dE}{dt} = 0$ e $\frac{dI}{dt} = 0$. É evidente que, para as Equações 1 e 2, existem infinitos pontos de equilíbrio localizados nos eixos x e y. A fim de visualizar essa informação, a Figura 4 mostra o gráfico com as *Nullclines* de $I(t)$ e $E(t)$. Como $\frac{dE}{dt} < 0$ e $\frac{dI}{dt} < 0$ para todos os instantes de tempo, todos os pontos de equilíbrios são sumidouros e tendem a levar a solução para a origem. Por razões obvias, valores negativos não foram considerados por se tratar do número de soldados em campo.

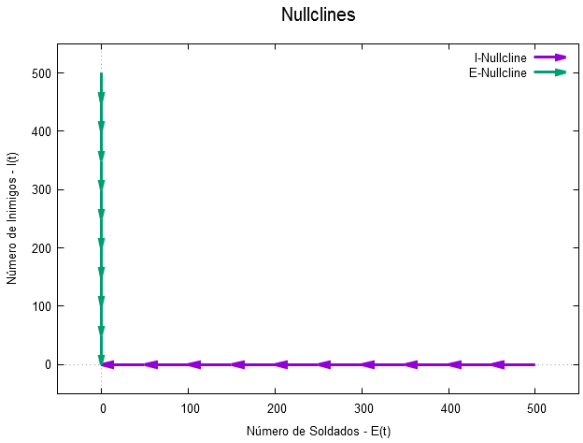

Figura 4: Gráfico com as *Nullclines* de $E(t)$ vs $I(t)$.

4.2 Plano de Fase

O comportamento das soluções do modelo descritas pelo gráfico de *Nullclines* é corroborado pelo plano de fase dos termos de reação do sistema. Como pode ser visto na Figura 5, construída como um campo vetorial de $I(t)$ vs $E(t)$, todas as soluções são levadas para algum dos eixos canônicos.

Pela Figura também é possível perceber que a taxa com que a quantidade de soldados caí no decorrer da batalha é maior quando existem muitos soldados em campo. O que faz sentido, dado que quanto mais soldados em batalha maior a chance de acontecer um acerto por parte do adversário. A medida que o número de soldados vai caindo, a taxa com que o número de soldados caí também é reduzida. Na Figura

5 foi escolhido o desenho do campo vetorial no lugar do campo de direções para evidenciar esse fato.

A Figura 5 também é composta pela solução numérica do modelo de acordo com os dados de entrada da Tabela 1. Os vetores da figura deveriam representar a direção da derivada em cada um dos pontos. Note que isso não acontece na Figura 5. Esse fato é decorrente da influência do termo de difusão na solução do sistema. O número de inimigos, para essa solução particular, decresce mais rapidamente que o número de soldados do exército quando a munição fica disponível na linha de frente. Essa disponibilidade é a influência do termo de difusão na solução.

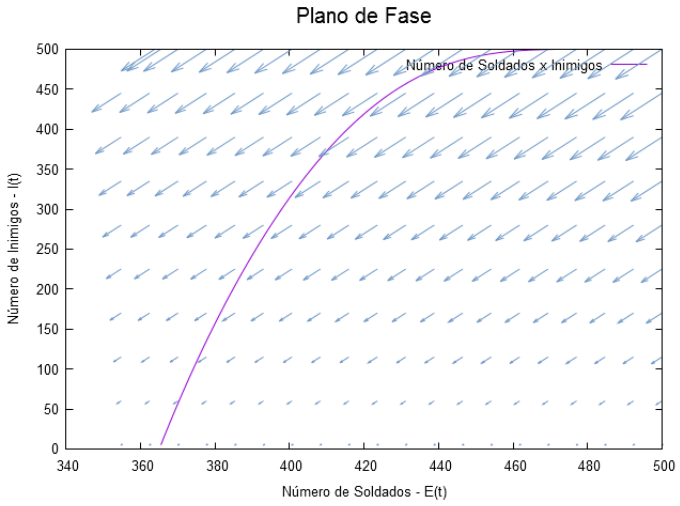

Figura 5: Resultado da batalha para os dados de entrada da Tabela 1. Nestas condições o exército sai vitorioso.

5 Resultados

A Figura 6 mostra o resultado da batalha descrita pelos dados de entrada mostrados na Tabela 1. É possível ver claramente que a estratégia de manter uma formação compacta, com uma pequena quantidade de soldados na linha-de-frente, é eficiente. É possível ver, também, o momento em que a munição começa a ser usada pela linha de frente do exército ($t \approx 2$). Neste momento o número de soldados inimigos começa a cair drasticamente.

A Figura 7 mostra a concentração de munição tanto na retaguarda quando na linha de frente para o exército sendo modelado. Note que, no inicio da batalha, o exército começa sofrente baixas, porém como a linha de frente é relativamente pequena em relação ao número de soldados, as baixas não representam perdas significativas do contingente de soldados. Em $t \approx 2$, quando a munição começa a chegar na linha de frente, é possível ver uma reviravolta drástica no destino da batalha. Como a perícia e o poder de fogo do exército são maiores em relação ao inimigo, quando a munição fica disponível na linha de frente, a queda no contingente inimigo é significativa levando-os a derrota.

Pode-se argumentar que não ter munição na linha de frente no início da

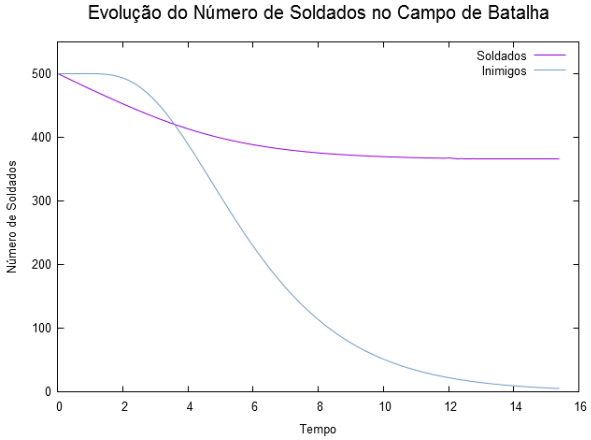

Figura 6: Resultado da batalha para os dados de entrada da Tabela 1. Nestas condições o exército sai vitorioso.

batalha ($t = 0a$) é uma falha do modelo, mas nesse ponto é razoável assumir que a munição presente na linha de frente não é suficiente para causar dano no inimigo, tornando a modelagem válida.

Uma análise interessante no contexto de batalhas entre exércitos é avaliar o impacto que a perícia do exército tem no resultado da batalha. Para fazer essa análise, o modelo foi utilizado para avaliar diversas batalhas onde a perícia do exército sofre variação ($0.001 \leq k_2 < 0.01$). De forma a ver o resultado das batalhas foi gerado o gráfico da Figura 8. Nessa figura é possível os diversos resultados. Note que a batalha foi perdida para $k_2 < 0.0064$, a causa da derrota não é somente a baixa perícia do exército. É possível ver que para $k_2 = 0.0055$ o número de baixas do inimigo deixa de aumentar. Isso acontece devido a munição da linha de frente ter se esgotado. A baixa perícia do exército faz com que a munição inicial não seja suficiente para vencer a batalha. É possível ver que mesmo com uma perícia inferior ao inimigo, o aumento do poder de fogo com a munição chegando na linha de frente é considerável.

A fim de ilustrar o que foi afirmado no parágrafo anterior, a quantidade inicial de munição foi aumentada em 10 vezes ($\mu(0,0) = 10000$) enquanto que o intervalo de variação da perícia do exército foi reduzido em 10 vezes ($0.0001 \leq k_2 < 0.001$). A Figura 9 mostra o resultado das batalhas com essa configuração. Note que a forma do gráfico não muda, o que comprova que o poder de fogo proporcionado pela munição na linha frente pode garantir a vitória mesmo com um exército de perícia mais baixa.

Outros resultados foram obtidos através da variação da porcentagem do exército que ocupa a linha de frente ($0.1 \leq \alpha < 1$) e variação do coeficiente de difusão de munição ($0.1 \leq D < 0.8$), Figuras 10 e 11, respectivamente. Os resultados deixam claro o que era esperado do modelo. Quanto maior a linha de frente maior é a probabilidade do exército sofrer uma baixa, algo similar com a área de contato entre os exércitos ser maior. No caso do coeficiente de difusão, quanto mais rápido a munição chega na linha de frente, mais rápido a batalha é vencida.

Por último, foi feita a variação no espaçamento da formação ($2 \leq L < 20$). Esse parâmetro indica qual a distância entre a retaguarda e a linha de frente. Os resultados da variação desse parâmetro são mostrados no gráfico da Figura 12.

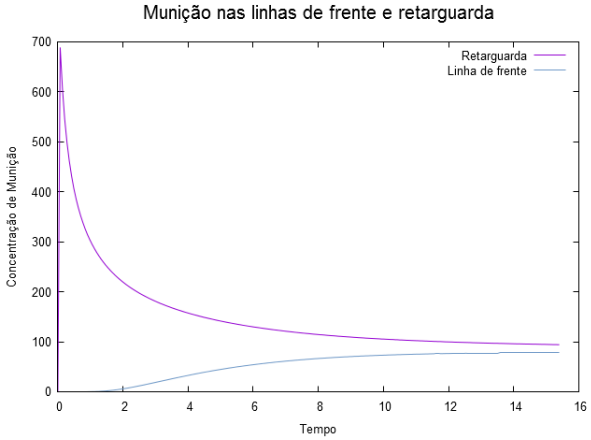

Figura 7: Difusão da munição a partir da retaguarda até a linha de frente. Como a munição se concentra totalmente na retaguarda, ela rapidamente chega a linha de frente.

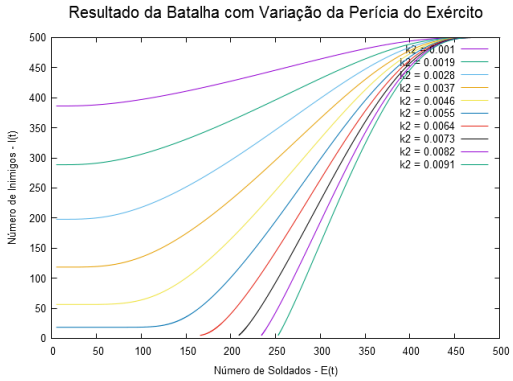

Figura 8: Resultado de batalhas com a variação da perícia do exército ($0.001 \leq k_2 < 0.01$).

Novamente, os resultados estão de acordo com a proposta do modelo. Quanto mais compacta a formação mais rápido a munição chega na linha de frente, e quanto mais espaçada, mais tempo levará para que a munição esteja disponível para uso. Pode-se perceber que se a formação estiver muito espaçada ($L > 12.8$), o tempo para que a munição chegue na linha de frente se torna tão grande que a batalha é perdida.

6 Conclusão

Neste trabalho foi apresentado um modelo matemático que descreve a evolução de uma batalha entre duas forças homogêneas com munição limitada na linha de frente. O modelo considera que a munição que é difundida pelo exército de forma que ela deve ser levada da retaguarda para a linha de frente para que a mesma possa causar impacto no inimigo.

Fica evidente que o impacto que a munição tem ao decidir o destino da

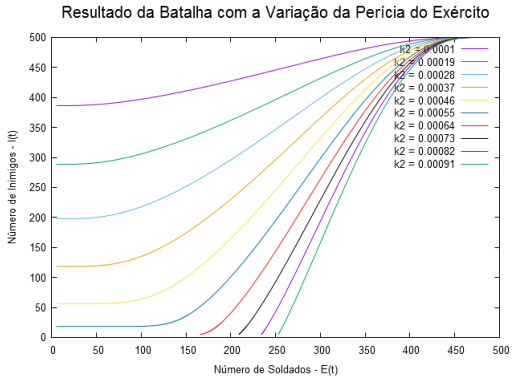

Figura 9: Resultado de batalhas com a variação da perícia do exército ($0.0001 \leq k_2 < 0.001$).

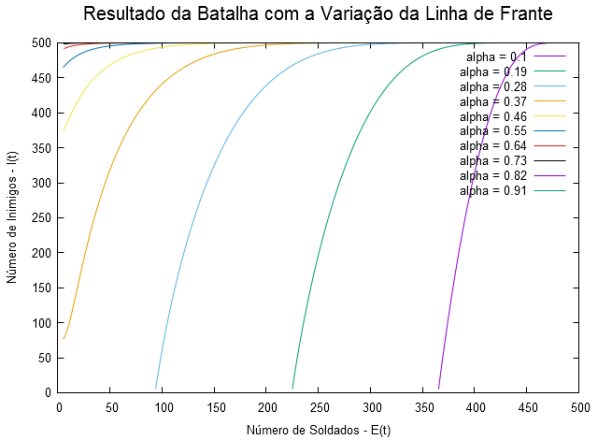

Figura 10: Resultado de batalhas com a variação da porcentagem do exército na linha de frente (α).

batalha é significativo. Manter uma fração do exército na linha de frente contando com substituição contínua dos soldados impacta no tempo total da batalha, porém uma linha de frente menor significa menos mortes por unidade de tempo, já que a probabilidade do inimigo acertar um disparo é proporcional ao tamanho da linha de frente.

O modelo descrito, apesar de simples, pode ser utilizado para descrever o comportamento de uma batalha entre duas forças homogêneas e pode ser uma boa ferramenta para entender o desfecho de batalhas desse tipo que ocorriam antes dos conflitos armados modernos.

As análises de *Nullclines* e Plano de Fase foram baseadas na teoria apresentada em [2].

O código fonte com a implementação do modelo está disponível em `https://github.com/mcleary/GentlemansBattle`.

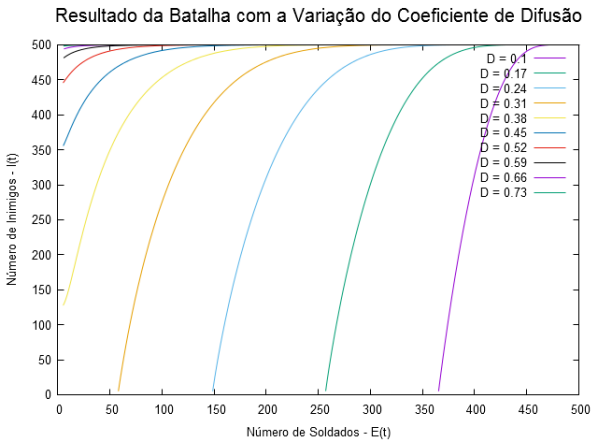

Figura 11: Resultado de batalhas com a variação do coeficiente de difusão de munição pelo exército ($0.1 \leq D < 0.8$).

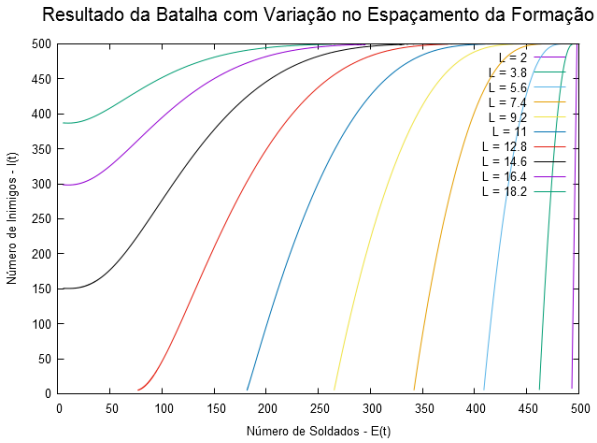

Figura 12: Resultado de batalhas com a variação do tamanho da formação ($2 \leq L < 20$).

Referências

[1] Giordana, F. R., Weir, M. D., and Fox, W. P. *A First Course in Mathematical Modelling*. Thomson Learning, 2003.

[2] Hirsch, M. W., Smale, S., and Devaney, R. L. *Differential equations, dynamical systems, and an introduction to chaos*. Academic Press, Waltham (Mass.), 2013.

[3] Navarro, R. Qual foi a primeira guerra da história? `http://mundoestranho.abril.com.br/materia/qual-foi-a-primeira-guerra-da-historia`. [Online; acessado em 2 de dezembro de 2015].

A Código Fonte com a Implementação do Modelo

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <string>

struct ModelInput
{
    double start_army_size;              // > 0          Tamanho inicial
    do exÅrcito
    double start_enemy_size;             // > 0          Tamanho inicial
    do exÅrcito inimigo
    double loose_battle_fraction;        // 0 < x < 1:   Porcentagem do
    exÅrcito que define derrota
    double army_skill;                   // 0 < x <= 1   PerÅcia do exÅ
    rcito em eliminar inimigos
    double enemy_skill;                  // 0 < x <= 1   PerÅcia do
    inimigo em eliminar o exÅrcito
    double start_ammo;                   // > 0          Quantidade de
    muniÅÃo que estarÅ disponÅvel durante a batalha
    double army_fire_rate;               // 0 < x <= 1   Taxa com que o
    exÅrcito consegue atirar a muniÅÃo disponÅvel
    double ammo_diffusion_coeffient;     // > 0          Velocidade com o
     que a muniÅÃo Å distribuÅda para o exÅrcito
    double formation_size;               // > 0          Tamanho da forma
    ÅÃo utilizada na batalha pelo exÅrcito
    double front_line_fraction;          // 0 < x <= 1   Porcentagem do
    exÅrcito que atuarÅ na linha de frente
    double enemy_front_line_fraction;    // 0 < x <= 1   Porcentagem do
    exÅrcito inimigo que atuarÅ na linha de frente

    double delta_time;                   // > 0
    double delta_x;                      // > 0

    /**
     * @brief ModelInputData default input data
     */
    ModelInput()
    {
        start_army_size             = 500.0;
        start_enemy_size            = 500.0;
        loose_battle_fraction       = 0.01;
        army_skill                  = 0.03;
        enemy_skill                 = 0.01;
        start_ammo                  = 1000;
        army_fire_rate              = 0.05;
        ammo_diffusion_coeffient    = 0.8;
        formation_size              = 7;
        front_line_fraction         = 0.1;
        enemy_front_line_fraction   = 0.1;

        delta_time                  = 0.07;
        delta_x                     = 0.6;

        std::cout << "CFL = " << ammo_diffusion_coeffient * delta_time /
     (delta_x * delta_x) << std::endl;
    }


    friend std::ostream& operator<< (std::ostream& out, const ModelInput
    & input_data)
```

```
    {
        out << "#-------------------------------------------------------" <<
    std::endl;
        out << "#--- GentlesmanBattle Input Data" << std::endl;
        out << "#-------------------------------------------------------" <<
    std::endl;
        out << "#- Start Army Size     : " << input_data.start_army_size
    << std::endl;
        out << "#- Start Enemy Size    : " << input_data.start_enemy_size
     << std::endl;
        out << "#- Army Skill          : " << input_data.army_skill <<
    std::endl;
        out << "#- Enemy Skill         : " << input_data.enemy_skill <<
    std::endl;
        out << "#- Start Ammo          : " << input_data.start_ammo <<
    std::endl;
        out << "#- Ammo Diffusion Coef: " << input_data.
    ammo_diffusion_coeffient << std::endl;
        out << "#- Battle Field Size   : " << input_data.formation_size
    << std::endl;
        out << "#- Front Line Fraction: " << input_data.
    front_line_fraction << std::endl;
        out << "#- Delta Time          : " << input_data.delta_time <<
    std::endl;
        out << "#- Delta X             : " << input_data.delta_x << std::
    endl;
        out << "#-------------------------------------------------------" <<
    std::endl;

        return out;
    }
};

struct ModelInfo
{
    double time             = 0.0;

    double new_army_size    = 0.0;
    double old_army_size    = 0.0;

    double new_enemy_size   = 0.0;
    double old_enemy_size   = 0.0;

    std::vector<double> new_ammo_amount;
    std::vector<double> old_ammo_amount;

    const ModelInput& model_input;

    double CFL = 0.0;

    ModelInfo(const ModelInput& input_data) :
        model_input(input_data)
    {
    }

    void update_input()
    {
        // Setting up the mesh for the ammo diffusion
        new_ammo_amount.resize(static_cast<int>(model_input.
    formation_size / model_input.delta_x));
        old_ammo_amount.resize(new_ammo_amount.size());

        // Initial condition
        old_army_size = model_input.start_army_size;
```

```
    old_enemy_size = model_input.start_enemy_size;

    // Ammot starts at rear-line
    old_ammo_amount[1] = model_input.start_ammo;

    CFL = model_input.ammo_diffusion_coeffient * model_input.
delta_time / (model_input.delta_x * model_input.delta_x);
}

void advance_time()
{
    // Calculates the fraction of the army and enemies currently
standing in the front-line
    double front_line_size = model_input.front_line_fraction *
old_army_size;
    double enemy_front_line_size = model_input.front_line_fraction *
 old_enemy_size;

    // Army
    new_army_size = old_army_size - model_input.delta_time *
model_input.enemy_skill * front_line_size * enemy_front_line_size;
    old_army_size = new_army_size;

    // Enemy
    double shoots_fired = old_ammo_amount.back() * model_input.
army_fire_rate;
    new_enemy_size = old_enemy_size - model_input.delta_time *
model_input.army_skill * shoots_fired * front_line_size *
enemy_front_line_size;
    old_enemy_size = new_enemy_size;

    // Ammo Diffusion
    for(size_t i = 1; i < new_ammo_amount.size() - 1; ++i)
    {
        new_ammo_amount[i] = old_ammo_amount[i] + CFL * (
old_ammo_amount[i-1] - 2.0 * old_ammo_amount[i] + old_ammo_amount[i
+1]);
    }

    //
    // Ammo boundary conditions
    //

    // At x=0 there is no flow.
    new_ammo_amount[0] = new_ammo_amount[1];

    // At x=L the ammo is being used by the soldiers
    // Calculates the percentage of ammo used at the frontline.
    double ammo_usage_ratio = 1.0 - (1.0 / front_line_size);
    new_ammo_amount.back() = new_ammo_amount[new_ammo_amount.size()
- 2] * ammo_usage_ratio;

    // Swap vectors for next time step
    new_ammo_amount.swap(old_ammo_amount);

    // Finally, advance the time
    time += model_input.delta_time;
}

/**
 * @brief True if the battle has came to an end
 *
 * The condition for the battle to stop is when the army size
reaches a fraction defined
```

```
     * by @ref ModelInputData::loose_battle_fraction. The condition is
    applied for both the army
     * and the enemies.
     *
     * @return True if the battle has came to an end false otherwise
     */
    bool should_stop() const
    {
        return new_army_size <= model_input.start_army_size *
    model_input.loose_battle_fraction ||
               new_enemy_size <= model_input.start_enemy_size *
    model_input.loose_battle_fraction;

    }

    /**
     * @brief Returns true if the number of soldiers is bigger than the
    number of enemies soldiers at this moment
     */
    bool is_army_winning() const
    {
        return old_army_size > old_enemy_size;
    }

    friend std::ostream& operator<< (std::ostream& out, const ModelInfo&
     info)
    {
        out << "#------------------------------------------------------" <<
    std::endl;
        out << "#--- GentlesmanBattle Execution Summary" << std::endl;
        out << "#------------------------------------------------------" <<
    std::endl;
        out << "# Number of Iterations      : " << info.time / info.
    model_input.delta_time << std::endl;
        out << "# Total time                : " << info.time << std::
    endl;
        out << "# Soldiers Count            : " << info.new_army_size <<
     std::endl;
        out << "# Enemies Count             : " << info.new_enemy_size
    << std::endl;
        out << "# Available Ammo at Fronline: " << info.new_ammo_amount.
    back() << std::endl;
        out << "# Available Ammo at Rearline: " << info.new_ammo_amount.
    front() << std::endl;
        out << "#------------------------------------------------------" <<
    std::endl;
        out << "# Battle Result: ";

        if(info.is_army_winning())
        {
            out << "YOU WIN!" << std::endl;
        }
        else
        {
            out << "YOU LOOSE!" << std::endl;
        }

        out << "#------------------------------------------------------" <<
    std::endl;

        return out;
    }
};
```

```
struct ModelOutput
{
    std::fstream output_file;
    std::fstream phase_plane_file;

    const ModelInput& model_input;
    const ModelInfo& model_info;

    const std::string prefix;

    ModelOutput(const ModelInfo& _model_info, const ModelInput&
    _model_input, const std::string& _prefix = "")
        : model_input(_model_input),
          model_info(_model_info),
          prefix(_prefix)
    {
    }

    void start_execution()
    {
        output_file.open(prefix + "_gentlemans_battle.dat", std::ios::
    out);
        output_file << model_input << std::endl << std::endl;
        output_file << "# Time   Army    Enemy   Rearguard   Frontline  "
     << std::endl;
    }

    void write_output_step()
    {
        output_file << std::setw(10) << std::setprecision(10) << std::
    fixed << std::setfill('0') <<
                       model_info.time << " " <<
                       model_info.old_army_size << " " <<
                       model_info.old_enemy_size << " " <<
                       model_info.old_ammo_amount.front() << " " <<
                       model_info.old_ammo_amount.back() << " " <<
                       std::endl;
    }

    void stop_execution(bool b_show_plots = true)
    {
        // Write Execution Summary into the output file
        output_file << model_info << std::endl;

        if(output_file.is_open())
        {
            output_file.close();
        }

        if(b_show_plots)
        {
            std::string output_filename_quotes = "'" + prefix + "
    _gentlemans_battle.dat'";
            std::string gnuplot_reaction_script = prefix + "
    _gentlemans_battle_result.gnu";
            std::string gnuplot_phase_plane_script = prefix + "
    _gentlemans_battle_phase_plane.gnu";
            std::string gnuplot_diffusion_script = prefix + "
    _gentlemans_battle_diffusion.gnu";

            const int title_font_size = 15;

            {
```

```
            std::fstream gnuplot_script_file(gnuplot_reaction_script
, std::ios::out);

            gnuplot_script_file << "set terminal 'wxt'" << std::endl
;
            gnuplot_script_file << "set xlabel 'Tempo'" << std::endl
;
            gnuplot_script_file << "set ylabel 'NÅmero de Soldados'"
 << std::endl;
            gnuplot_script_file << "set zeroaxis" << std::endl;
            gnuplot_script_file << "set yrange [0:550]" << std::endl
;
            gnuplot_script_file << "set title 'EvoluÅÃo do NÅmero de
 Soldados no Campo de Batalha' font 'Arial, " << title_font_size <<
"'" << std::endl;
            gnuplot_script_file << "plot " <<
                                   output_filename_quotes << " using
 1:2 with lines title 'Soldados'," <<
                                   output_filename_quotes << " using
 1:3 with lines title 'Inimigos' linetype rgb '#6f99c8'" << std::
endl;
            gnuplot_script_file << "set output 'report/figs/
battle_reaction.png'" << std::endl;
            gnuplot_script_file << "set terminal pngcairo enhanced
font 'arial,10' fontscale 1.0" << std::endl;
            gnuplot_script_file << "replot" << std::endl;

        }
        {
            std::fstream gnuplot_script_file(
gnuplot_diffusion_script, std::ios::out);

            gnuplot_script_file << "set terminal 'wxt'" << std::endl
;
            gnuplot_script_file << "set xlabel 'Tempo'" << std::endl
;
            gnuplot_script_file << "set ylabel 'ConcentraÅÃo de Muni
ÅÃo'" << std::endl;
            gnuplot_script_file << "set zeroaxis" << std::endl;

            gnuplot_script_file << "set title 'MuniÅÃo nas linhas de
 frente e retarguarda' font 'Arial, " << title_font_size << "'" <<
std::endl;
            gnuplot_script_file << "plot " <<
                                   output_filename_quotes << " using
 1:4 with lines title 'Retarguarda'," <<
                                   output_filename_quotes << " using
 1:5 with lines title 'Linha de frente' linetype rgb '#6f99c8'" <<
std::endl;
            gnuplot_script_file << "set output 'report/figs/
battle_ammo_diffusion.png'" << std::endl;
            gnuplot_script_file << "set terminal pngcairo enhanced
font 'arial,10' fontscale 1.0" << std::endl;
            gnuplot_script_file << "replot" << std::endl;
        }
        {
            const ModelInput& input = model_info.model_input;

            std::fstream gnuplot_script_file(
gnuplot_phase_plane_script, std::ios::out);

            gnuplot_script_file << "set terminal 'wxt'" << std::endl
;
```

```
                gnuplot_script_file << "k1 = " << input.enemy_skill <<
    std::endl;
                gnuplot_script_file << "k2 = " << input.army_skill <<
    std::endl;
                gnuplot_script_file << "alpha = " << input.
    front_line_fraction << std::endl;
                gnuplot_script_file << "beta = " << input.
    enemy_front_line_fraction << std::endl;
                gnuplot_script_file << "vec_scale = 0.5" << std::endl;

                gnuplot_script_file << "dEdt(I,E) = -k1 * alpha * E *
    beta * I" << std::endl;
                gnuplot_script_file << "dIdt(I,E) = -k2 * alpha * E *
    beta * I" << std::endl;

                gnuplot_script_file << "vx(x,y) = dEdt(x,y) * vec_scale
    # * (1 / sqrt(dEdt(x,y)**2 + dIdt(x,y)**2))" << std::endl;
                gnuplot_script_file << "vy(x,y) = dIdt(x,y) * vec_scale
    # * (1 / sqrt(dEdt(x,y)**2 + dIdt(x,y)**2))" << std::endl;

                gnuplot_script_file << "set samples 20" << std::endl;
                gnuplot_script_file << "set zeroaxis" << std::endl;

                gnuplot_script_file << "set xlabel 'NÅmero de Soldados -
     E(t)'" << std::endl;
                gnuplot_script_file << "set ylabel 'NÅmero de Inimigos -
     I(t)'" << std::endl;
                gnuplot_script_file << "set title 'Plano de Fase' font '
    Arial, " << title_font_size << "'" << std::endl;

                gnuplot_script_file << "plot '_gentlemans_battle.dat'
    using 2:3 with lines title 'NÅmero de Soldados x Inimigos'," <<
                                       " '++' u 1:2:(vx($1,$2)):(vy($1,
    $2)) with vectors notitle linetype rgb '#6f99c8'" <<
                                       std::endl;

                gnuplot_script_file << "set output 'report/figs/
    battle_phase_plane.png'" << std::endl;
                gnuplot_script_file << "set terminal pngcairo enhanced
    font 'Arial,10' fontscale 1.0" << std::endl;
                gnuplot_script_file << "replot" << std::endl;
            }

            // show reaction plot
            std::string plot_command = "gnuplot -p " +
    gnuplot_reaction_script + " > battle_reaction.png";
            std::cout << plot_command << std::endl;
            system(plot_command.data());

            // show diffusion plot
            plot_command = "gnuplot -p " + gnuplot_diffusion_script + "
    > battle_ammo_diffusion.png";
            std::cout << plot_command << std::endl;
            system(plot_command.data());

            // show phase plane plot
            plot_command = "gnuplot -p " + gnuplot_phase_plane_script +
    " > battle_phase_plane.png";
            std::cout << plot_command << std::endl;
            system(plot_command.data());
        }
    }
};
```

```
struct ModelCondensedOutput
{
    int num_executions;
    std::string param_name;
    std::string graph_title;
    std::string output_filename;
    std::vector<double> param_value_list;

    ModelCondensedOutput(int _num_executions,
                         const std::string& _param_name,
                         const std::string& _graph_title,
                         const std::string& _output_filename
                         ) :
        num_executions(_num_executions),
        param_name(_param_name),
        graph_title(_graph_title),
        output_filename(_output_filename)
    {
        param_value_list.reserve(num_executions);
    }

    void add_param_value(double param_value)
    {
        param_value_list.push_back(param_value);
    }

    void show_condensed_plot()
    {
        std::string script_filename = "_condenser.gnu";

#if 0
        {
            std::fstream gnuplot_script_file(script_filename, std::ios::
    out);

            gnuplot_script_file << "set terminal 'wxt'" << std::endl;
            gnuplot_script_file << "set xlabel 'Tempo'" << std::endl;

            gnuplot_script_file << "plot ";

            for(int i = 0; i < num_executions; ++i)
            {
                gnuplot_script_file << "'" << i << "_gentlemans_battle.
    dat' using 1:2 with lines linetype rgb '#e800" << i << i << "' title
     '" << param_name << " = " << param_value_list[i] << "',"
                                    << "'" << i << "_gentlemans_battle.
    dat' using 1:3 with lines linetype rgb '#21eb12' notitle '" <<
    param_name << " = " << param_value_list[i] << "',";
            }

            gnuplot_script_file.close();

            std::string plot_command = "gnuplot -p " + script_filename;
            std::cout << plot_command << std::endl;

            system(plot_command.data());
        }
#endif
        {
            std::fstream gnuplot_script_file(script_filename, std::ios::
    out);

            gnuplot_script_file << "set terminal 'wxt'" << std::endl;
```

```
            gnuplot_script_file << "set xlabel 'NÅmero de Soldados - E(t
    )'" << std::endl;
            gnuplot_script_file << "set ylabel 'NÅmero de Inimigos - I(t
    )" << std::endl;
            gnuplot_script_file << "set title '" << graph_title << "'
    font 'Arial, 15'" << std::endl;

            gnuplot_script_file << "plot ";

            for(int i = 0; i < num_executions; ++i)
            {
                gnuplot_script_file << "'" << i << "_gentlemans_battle.
    dat' using 2:3 with lines title '" << param_name << " = " <<
    param_value_list[i] << "',";
            }
            gnuplot_script_file << std::endl;

            gnuplot_script_file << "set terminal pngcairo enhanced font
    'Arial,10' fontscale 1.0" << std::endl;
            gnuplot_script_file << "set output 'report/figs/battle_" <<
    output_filename << ".png'" << std::endl;
            gnuplot_script_file << "replot" << std::endl;

            gnuplot_script_file.close();

            std::string plot_command = "gnuplot -p " + script_filename;
            std::cout << plot_command << std::endl;

            system(plot_command.data());

        }
    }
};

struct GentlesmanBattleModel
{
    ModelInput input;
    ModelOutput output;
    ModelInfo info;

    GentlesmanBattleModel(const std::string& prefix = "") :
        output(info, input, prefix), info(input)
    {
    }

    void run(bool b_show_plots = true)
    {
        // Print parameters information
        std::cout << input << std::endl;

        info.update_input();

        output.start_execution();

        do
        {
            output.write_output_step();
            info.advance_time();
        }
        while(!info.should_stop());

        // Print execution summary
        std::cout << info << std::endl;
```

```
        output.stop_execution(b_show_plots);
    }
};

int main()
{
    const int num_executions = 1;
    std::string param_name = "L";

    double param_min = 2;
    double param_max = 20;

    if(num_executions > 1)
    {
        ModelCondensedOutput condensend_output(num_executions,
                                               param_name,
                                               "Resultado da Batalha com
 VariaÅÅo no EspaÅamento da FormaÅÅo",
                                               "formation_size_variation
");

        for(int i = 0; i < num_executions; ++i)
        {
            GentlesmanBattleModel model(std::to_string(i));
            double param_value = param_min + (param_max - param_min) * (
i / static_cast<double>(num_executions));

            model.input.formation_size = param_value;

            model.run(false);

            condensend_output.add_param_value(model.input.formation_size
);
        }
        condensend_output.show_condensed_plot();
    }
    else
    {
        GentlesmanBattleModel model;
        model.run();
    }

    return EXIT_SUCCESS;
}
```