

# Um Modelo Matemático para Análise de Batalhas Históricas Considerando Munição Limitada

Thales Luis Rodrigues Sabino

8 de dezembro de 2015

Relatório técnico do trabalho realizado na disciplina *Instrução 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

Abstract text

## 1 Introdução

Descoberta no século 19 na antiga Suméria, a cidade-estado de Lagash possui ruínas que contém registros que mostram 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].

Fica claro que 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 (procurar referência).

É evidente que uma parte significativa 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 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 históricas*, 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 Gettysburg. No entanto, esse assalto foi repellido com um grande número de baixas e decisão a favor da União.

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



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



Figura 3: Pintura da Batalha de Gettysburg

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 acres-

centados 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 a força  $E$  (o exército sob comando) e a  $I$  (o exército inimigo). 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:

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

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 em um dado momento. A Equação 1 pode ser resumida como a modelagem da batalha das duas linhas de frente por unidade de tempo.

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

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

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:

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

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 a maior parte de seus soldados em campo. 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.

## 3.3 Discretização por Diferenças Finitas

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

Para as Equações 1 e 2 foi escolhida 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:

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

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

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

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

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

isolando o termo  $\mu_{i+1}^t$  tem-se que:

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

onde  $\frac{D\Delta t}{(\Delta x)^2} < \frac{1}{2}$  é a condição CFL do modelo. Portanto, os valores de  $D$ ,  $\Delta t$  e  $\Delta x$  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			
$\Upsilon$	500	Número inicial de soldados no exército	$\Upsilon > 0$
$\Phi$	500	Número inicial de soldados inimigos	$\Phi > 0$
$\Delta$	1000	Quantidade inicial de munição na retaguarda	$\Delta > 0$
$\gamma$	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	$0 < k_1 \leq 1$
$\rho$	0.05	Medida da taxa de disparos feitas pelo exército por unidade de tempo	$0 < \rho \leq 1$
$D$	0.8	Coefficiente de difusão da munição pelo exército	$D > 0$
$L$	7	Tamanho da formação do exército	$L > 0$
$\alpha$	0.1	Porcentagem do exército que ocupa a linha de frente	$0 < \alpha \leq 1$
$\beta$	0.1	Porcentagem do exército inimigo que ocupa a linha de frente	$0 < \beta \leq 1$
$\Delta t$	0.07	Discretização temporal	$\Delta t > 0$
$\Delta x$	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 do modelo.

#### 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.

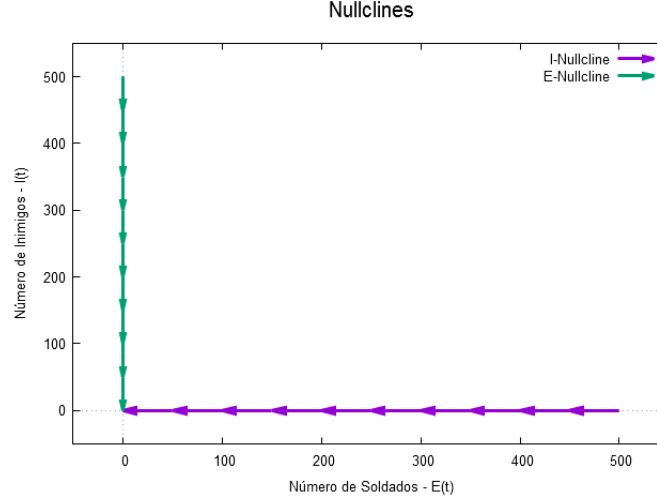


Figura 4: Gráfico com as *Nullclines* de  $E(t)$  vs  $I(t)$ .

#### 4.2 Plano de Fase

O comportamento das soluções do modelo descritos 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 cai 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 cai 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.

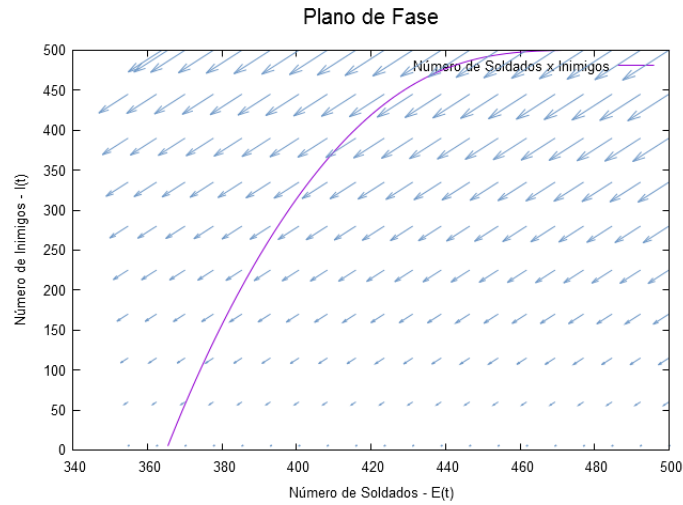


Figura 5: Resultado da batalha para os dados de entrada da Tabela 1. Nestas condições o exército sai vitorioso.

## 5 Resultados

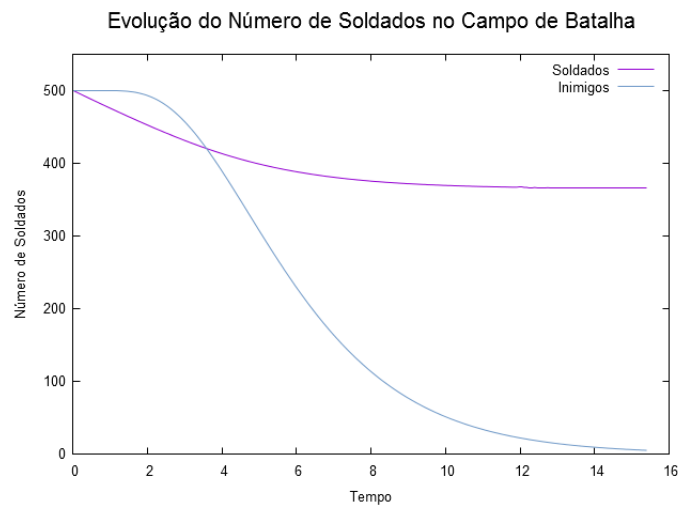


Figura 6: Resultado da batalha para os dados de entrada da Tabela 1. Nestas condições o exército sai vitorioso.

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 claramente o momento em que a munição começa a

A Figura 7 mostra a concentração de munição tanto na retaguarda quando na linha de frente para o exército sendo modelado. Note que, no início da batalha, o exército começa sofrendo baixas, porém como a linha de frente é relativamente pequena



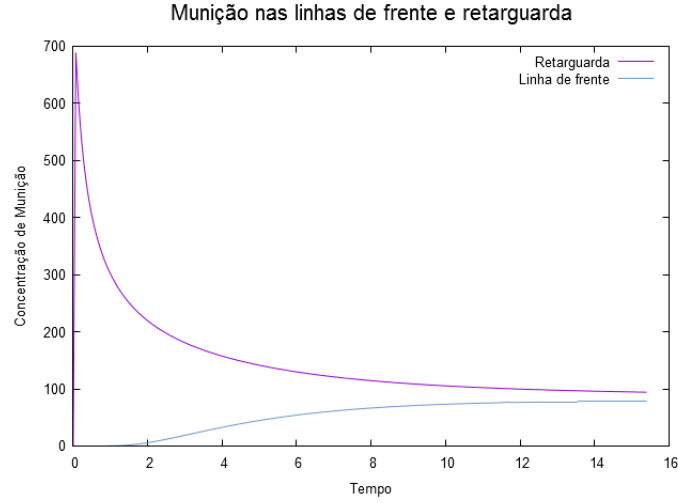


Figura 7: Difusão da munição a partir da retaguarda até a linha de frente. Como a munição se concentra totalmente na retaguarda, ela rapidamente chega a linha de frente.

em relação ao número de soldados, as baixas não representam perdas significativas do contingente de soldados. Por volta de  $t = 2$ , 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 vitória.

Pode-se argumentar que não ter munição na linha de frente 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.

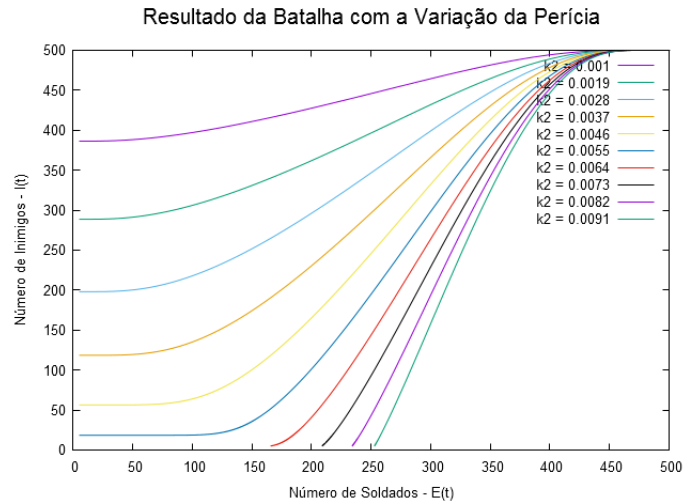


Figura 8: Resultado de batalhas com a variação da perícia do exército ( $k_2$ ).

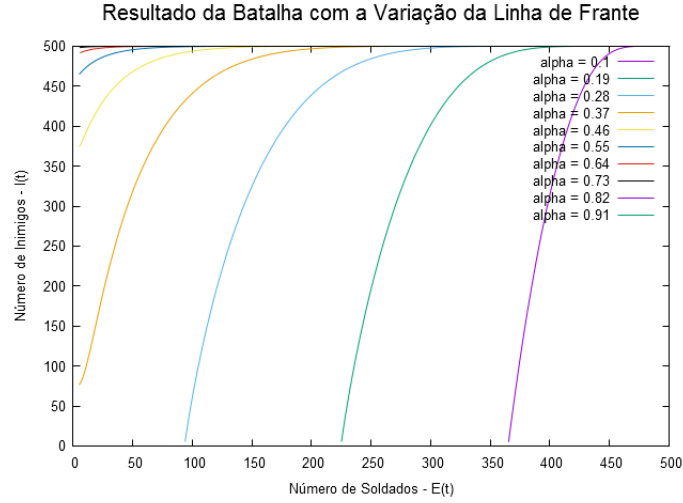


Figura 9: Resultado de batalhas com a variação da porcentagem do exército na linha de frente ( $\alpha$ ).

## 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 batalha é significativo. Manter uma fração do exército na linha de frente contando com substituição contínua dos soldados impacta no tempo total da batalha.

O modelo descrito, apesar de simples, pode ser utilizado para descrever o comportamento de uma batalha entre duas forças homogêneas e pode ser uma 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].

## 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].



Figura 10: Resultado de batalhas com a variação do coeficiente de difusão de munição pelo exército ( $D$ ).

## 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 exército
    double start_enemy_size;          // > 0           Tamanho inicial
    do exército inimigo
    double loose_battle_fraction;     // 0 < x < 1:    Porcentagem do
    exército que define derrota
    double army_skill;                // 0 < x <= 1    Perícia do exército
    rcito em eliminar inimigos
    double enemy_skill;               // 0 < x <= 1    Perícia do
    inimigo em eliminar o exército
    double start_ammo;                // > 0           Quantidade de
    munição que estará disponível durante a batalha
    double army_fire_rate;            // 0 < x <= 1    Taxa com que o
    exército consegue atirar a munição disponível
    double ammo_diffusion_coefficient; // > 0           Velocidade com o
    que a munição é distribuída para o exército
    double formation_size;            // > 0           Tamanho da forma
    ão utilizada na batalha pelo exército
    double front_line_fraction;       // 0 < x <= 1    Porcentagem do
    exército que atuará na linha de frente
    double enemy_front_line_fraction; // 0 < x <= 1    Porcentagem do
    exército inimigo que atuará 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_coefficient = 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_coefficient * delta_time /
(delta_x * delta_x) << std::endl;
}

friend std::ostream& operator<< (std::ostream& out, const ModelInput
& input_data)
{
    out << "#-----" <<
std::endl;
    out << "#--- GentlemanBattle 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_coefficient << 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_coefficient * 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 come 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 come 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 << "#— GentlemanBattle 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 'Número 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 'Evolução do Número 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' fontsize 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 'Concentração de Muni
ção'" << std::endl;
            gnuplot_script_file << "set zeroaxis" << std::endl;

```



```

        gnuplot_script_file << "set title 'Muni o 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 '#6f9c8' " <<
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 ' mero de Soldados -
E(t)'" << std::endl;
        gnuplot_script_file << "set ylabel ' mero 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 ' mero de Soldados x Inimigos'," <<
        " '++' u 1:2:(vx($1,$2)):(vy($1,
$2)) with vectors notitle linetype rgb '#6f9c8' " <<
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' fontsize 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 'N mero de Soldados - E(t"
        )" << std::endl;
        gnuplot_script_file << "set ylabel 'N mero 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' fontsize 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 GentlemanBattleModel
{
    ModelInput input;
    ModelOutput output;
    ModelInfo info;
};

```

```

GentlemanBattleModel(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 = 10;
    std::string param_name = "D";

    double param_min = 0.1;
    double param_max = 0.8;

    if(num_executions > 1)
    {
        ModelCondensedOutput condensend_output(num_executions,
                                                param_name,
                                                "Resultado da Batalha com
a Varia  o do Coeficiente ",
                                                "ammo_diffusion_variation
");

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

            model.input.front_line_fraction = param_value;

            model.run(false);

            condensend_output.add_param_value(model.input.
front_line_fraction);
        }
        condensend_output.show_condensed_plot();
    }
    else

```

```
{  
    GentlesmanBattleModel model;  
    model.run();  
}  
  
return EXIT_SUCCESS;  
}
```