Manual do Usuário

ISOTHERMLIB

Versão 1.0

Aline Zuliani Lunkes
Iasmim Barboza Stork
Lara Botelho Brum
Luan Rodrigues Soares de Souza
João Flávio Vieira de Vasconcellos

31 de julho de 2024

Sumário

1 Apresentação da Biblioteca IsothermLib

O propósito deste manual é apresentar a biblioteca IsothermLib. Ela foi elaborada para auxiliar quem desenvolve programas na área de fenômenos de transporte e precisam que nos efeitos de adsorção/dessorção sejam considerados nos modelos de transporte de massa.

A adsorção é a adesão de átomos, íons ou moléculas de um gás, líquido ou sólido dissolvido a uma superfície. Este processo cria uma película do adsorvato na superfície do adsorvente e difere do processo conhecido como absorção, em que um fluido (o absorvente) é dissolvido por ou permeia um líquido ou sólido (o absorvente). A adsorção é um processo de superfície, enquanto a absorção envolve todo o volume do material. O termo sorção engloba ambos os processos, enquanto a dessorção é o inverso da adsorção [1].

Para descrever corretamente a adsorção e/ou dessorção nestes eventos, usa-se frequentemente o conceito de isotermas de adsorção, que são equações matemáticas usadas para modelar, em termos quantitativos, a adsorção ou dessorção de solutos por sólidos [2]. Tais funções procuram descrever como um adsorvente efetivamente adsorverá um soluto. Na revisão da literatura conduzida no âmbito deste trabalho foram encontrados cerca de 40 modelos diferentes e chegou-se a conclusão que elas poderiam ser escritas da seguinte forma:

$$q = f(C) \quad \text{ou} \quad q = f(C, T) \tag{1.1}$$

em que q é a quantidade de soluto retida no adsorvente em [M/M], C a concentração de equilíbrio em [M/L^3] e T a temperatura em [K].

Sendo assim, constatou-se a possibilidade de empregar os recursos de orientação a objetos e desenvolver na linguagem C++20, uma biblioteca denominada IsothermLib, cujo propósito é utilizar qualquer uma das isotermas catalogadas na forma da Eq. (1.1) de maneira bastante simples. Desta forma, pensou-se em escrever uma biblioteca computacional em que todos estes modelos estivessem presentes e fosse possível intercambiar o modelo de isoterma de uma maneira bastante simples sem que seja necessário realizar grandes modificações no código que utiliza estas isotermas.

Algumas das isotermas apresentada neste texto são do tipo $q = q(C, T)$ em que T é a temperatura do fluido em $[K]$. Neste texto, não consideramos que T seja um parâmetro da isoterma, como também não consideramos C. Ao longo do texto, exploraremos mais detalhadamente as características dessas isotermas e sua relevância para o estudo em questão.

2 Instalação da Biblioteca IsotermLib

2.1 Instalação

Nesta versão a biblioteca foi testada e preparada para ser utilizada em ambiente Linux. Contudo, como usa somente os recursos do C++20 padrão, com alguns pequenos ajustes, ela poderá ser utilizada em outros sistemas operacionais.

2.1.1 Instruções de Compilação para a Biblioteca C++

Este documento fornece instruções para compilar a biblioteca C++ usando o sistema de build CMake. Siga as etapas para compilar a biblioteca a partir do código-fonte. Requisitos

Antes de começar, certifique-se de que você tem as seguintes ferramentas instaladas:

1. CMake: Ferramenta de configuração e geração de projetos de compilação. Se ainda não o tem instalado é necessário que proceda a instalação do mesmo.
2. Make: Ferramenta para automatizar a construção de projetos de software. Normalmente já está disponível em sistemas Unix e Linux. Se ainda não o tem instalado é necessário que proceda a instalação do mesmo.
3. C++20: Certifique-se que a versão do compilador instalada permite compilar códigos escritos com comandos do C++20. No compilador GCC isto é possivel a partir da versão 10.1.

2.1.2 Passos para Compilar a Biblioteca

Siga as seguintes etapas para compilar a biblioteca.

1. Faça uma cópia da biblioteca que está armazenada no GitHub. Sugerimos o seguinte comando:

```
git clone https://github.com/IsothermLib/IsothermLib.git IsothermLib
```

   Este comando irá criar a pasta IsothermLib e fazer uma cópia da última versão da biblioteca que está armazenada no GitHub.Crie uma pasta build dentro da pasta

Crie uma pasta chamada build na raiz do diretório do projeto. Esta pasta será usada para armazenar todos os arquivos gerados durante o processo de compilação. mkdir build cd build cmake ../. Compile o projeto usando o Make. Este comando compila o código-fonte e gera a biblioteca e os arquivos executáveis. make Se desejar instalar a biblioteca no sistema, você poderá executar o comando make install. Você pode precisar de permissões de superusuário para esta etapa. sudo make install

3 Isoterma com um parâmetro

3.1 Isoterma de Henry

Na busca por modelos para elaborar esta biblioteca, encontramos uma única isoterma cujo modelo dependia somente de um parâmetro. Trata-se da isoterma de Henry [3]. As isotermas com dois ou mais parâmetros, provavelmente, modelam bem melhor o fenômeno.

A equação para este modelo de isoterma é [3]:

$$q = K_1 C \quad \text{com} \quad K_1 >\geq 0 \tag{3.1}$$

4 Isoterma com dois parâmetros

4.1 Lista de isotermas com dois parâmetros

Uma lista dos modelos de isotermas que serão tratados neste capítulo está detalhada a seguir, incluindo as referências onde aspectos importantes de cada isoterma podem ser encontrados. Usou-se, preferencialmente, a referência original da isoterma que estivesse disponível na internet. Porém, isto nem sempre foi possível e, nestes casos, uma outra referência que pudesse substituir a original foi adotada.

As seguintes isotermas de dois parâmetros foram consideradas:

1. Isoterma de Dubinin-Radushkevich, na seção 4.1.1;
2. Isoterma de Elovich, na seção 4.1.2;
3. Isoterma de Freundlich, na seção 4.1.3;
4. Isoterma de Halsey, na seção 4.1.4;
5. Isoterma de Harkin-Jura, na seção 4.1.5;
6. Isoterma de Jovanovic, na seção 4.1.6;
7. Isoterma de Langmuir, na seção 4.1.7;
8. Isoterma de Temkin, na seção 4.1.8;

4.1.1 Isoterma de Dubinin-Radushkevich

Equação do Modelo

A equação para este modelo de isoterma é [4]:

$$\theta = e^{-K_1 \epsilon^2}, \tag{4.1}$$

em que $\theta = q/q_{max}$ e ϵ é o potencial de Polanyi definido como em [5]:

$$\epsilon = RT \ln\left(1 + \frac{C_0}{C}\right), \tag{4.2}$$

em que $R = 8{,}31446261815324$ $[\mathrm{J\,mol^{-1}\,K^{-1}}]$ é a constante universal dos gases, T a temperatura em [K] e $C_0 = 1$ $[\mathrm{mol\,dm^{-3}}]$.

Em diversas referências, como em Gautam e Chattopadhyaya [6], Ayawei, Ebelegi e Wankasi [7], a equação do potencial de Polanyi é definida como:

$$\epsilon = RT \ln\left(1 + \frac{1}{C}\right), \tag{4.3}$$

e, como bem alertou Zhou [5], esta Equação (4.3) está dimensionalmente errada e dependendo da unidade adotada para C o valor de ϵ será calculado com imprecisão.

Por esta razão, será utilizado a Equação (4.2) para o cálculo de ϵ e, sendo assim, deve-se observar se a unidade de C é dimensionalmente compatível com C_0 definido na unidade de $[\mathrm{mol\,dm^{-3}}]$, senão faz-se necessário uma conversão de unidades para que o valor de ϵ seja calculado corretamente.

Os dois parâmetros deste modelo são [8]:

q_{max} – Constante de equilíbrio de Dubinin-Radushkevich.

K_1 – Coeficiente associada com a energia de adsorção.

Deve ser observado que C_0, na Equação (4.2), não é um valor que possa ser alterado no código, por isto não está presente na lista de parâmetros do modelo.

Para a unidade de R padrão, o valor de K_1 deverá ser definido na unidade de $[\mathrm{mol^2\,J^{-2}}]$.

Restrições do Modelo

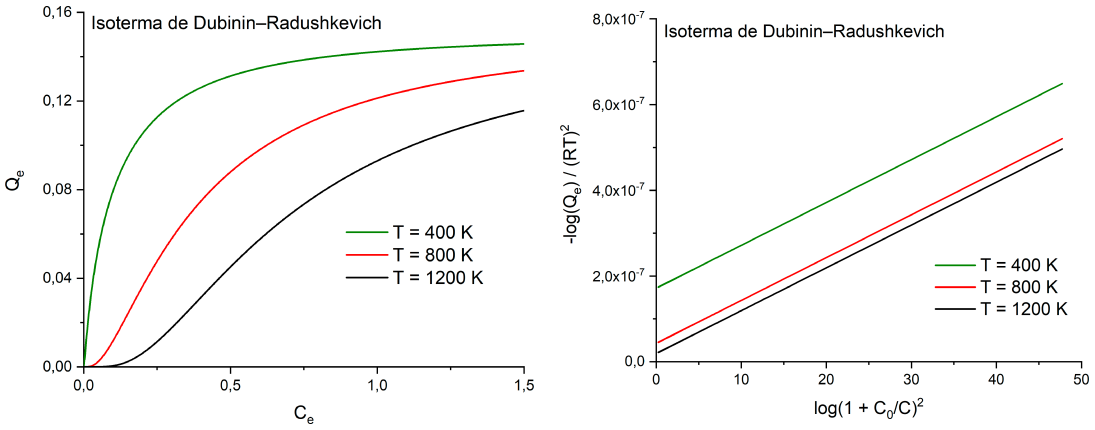
As seguintes condições deverão ser satisfeitas durante o uso desta isoterma:

$$\begin{aligned} &C > 0 \quad ; \\ &q_{max} > 0 \quad ; \\ &K_1 > 0 \quad ; \\ &T > 0 \quad \text{e}, \\ &R > 0. \end{aligned}$$

No caso de alguma destas restrições ser violada, o programa irá parar a sua execução e uma mensagem de erros irá ser emitida.

Exemplo de Aplicação do Modelo

Nos gráficos mostrados na Figura 4.1 os valores utilizados em sua confecção foram gerados com o programa a seguir. No Código 4.1 na Figura 4.1a tem-se casos mostrando a influência da temperatura. Na Figura 4.1b tem-se os mesmos valores mas apresentados de forma linearizada.



(a) Gráfico $Q_e \times C_e$

(b) Gráfico $Q_e \times C_e$ – Linearizado

Figura 4.1: Isotermas de Dubinin-Radushkevich

Código 4.1: Código C++ empregado na geração das curvas da Figura 4.1

```
1  // include da lib c++
2  #include <cmath>
3  #include <iomanip>
4  #include <iostream>
5  #include <string>
6
7  // include da isotherm++
8  #include <Isotherm/Misc/Misc.h>
9  #include <Isotherm/Isotherm/TwoParameters/DubininRadushkevich.h>
10
11 int main(int argc, char** argv) {
12
13
14 // Definição dos parâmetros
15
16 const Real TEMP(400);      // Temperatura em kelvin
17 const Real CE(2e-3);       // Concentração de equilíbrio
18                            // do adsorvato no adsorvente em [mg / L]
19 const Real QMAX(15e-2);    // Quantidade de adosrvato em equilíbrio [mol/g]
20 const Real K1(1e-8);       // Constante de Dubinin-Radushkevich
21
22 const ist::DubininRadushkevich  isoterma(QMAX, K1);
```

```
23 const Real                          qe = isoterma.Qe(CE, TEMP);
24
25 // Impressão das informações sobre as constantes do modelo
26
27     std::cout << "\n";
28     PrintLine(std::cout);
29     std::cout << "Impressao de informacoes sobre as constantes\n";
30     PrintLine(std::cout);
31
32     std::cout   << isoterma
33                 << std::endl;
34
35     std::cout   << "Numero de constantes de variavel isoterma: "
36                 << isoterma.NumberConst()
37                 << std::endl;
38
39     std::cout << std::setfill(' ');
40     std::cout << "Impressao de informacoes da isoterma\n"
41               << std::setw(20)
42               << std::left
43               << "Parametro"
44               << std::setw(40)
45               << " Detalhe do parametro"
46               << std::right
47               << std::endl;
48
49     for (auto x : isoterma) {
50         std::cout   << std::setw(20)
51                     << std::left
52                     << x.first
53                     << std::setw(40)
54                     << x.second
55                     << std::right
56                     << "\n";
57     }
58
59
60     std::cout << "\n\nPara a temperatura: " << TEMP
61               << " e concentracao de equilibrio: " << CE
62               << " tem-se que qe = " << qe
63               << std::endl;
64
65 // Imprimindo a curva da isoterma
66
67 const Real CMIN(1E-3);     // valor minimo da concentração
68 const Real CMAX(1.5);      // valor máximo da concentração
69 const UInt NPTO(1001);     // numero de pontos gerados
70
71  // Função de linearização da isoterma no eixo X
```

```
72  auto EixoX = [](const Real& _qe, const Real&_ce)
73  {
74  Real auxi = log(1 + 1.0 / _ce);
75
76      return  auxi * auxi;
77  };
78
79  // Função de linearização da isoterma no eixo Y
80  auto EixoY = [isoterma, TEMP](const Real& _qe, const Real&_ce)
81  {
82
83      auto auxi =    1.0 / (isoterma.Rgas() * TEMP);
84      return - log( _qe ) * auxi * auxi;
85
86  };
87
88   //Arquivo onde que serão impressos os valores desta isoterma
89  const std::string FILENAME("CasoL400.dat");
90
91      std::cout << "\n";
92      PrintLine(std::cout);
93      std::cout   << "Impressao dos grafico no arquivo: "
94                  << FILENAME
95                  << "\n";
96
97      PrintLine(std::cout);
98
99      isoterma.PlotGraph  (   FILENAME
100                         ,   EixoX
101                         ,   EixoY
102                         ,   NPTO
103                         ,   CMIN
104                         ,   CMAX
105                         ,   TEMP);
106
107     return 0;
108 }
```

4.1.2 Isoterma de Elovich

A equação para este modelo pode ser encontrada em [9, 10]:

$$\theta = K_1 C \exp(-\theta), \tag{4.4}$$

em que $\theta = q/q_{max} > 0$. Os dois parâmetros deste modelo são [8]:

q_{max} – Capacidade máxima de adsorção.

K_1 – Constante da isoterma de Elovich.

A determinação de $\theta = f(C)$ deve ser feita empregando o método de Newton-Raphson [11]. A função a ser utilizada é:

$$f(\theta) = \theta - K_1 C \exp(-\theta) = 0 \quad \text{com} \quad 0 \leq \theta \leq 1 \quad \text{e} \quad C \geq 0 \tag{4.5}$$

Após a determinação de θ chega-se ao valor de q utilizando-se a relação $\theta = q/q_{max}$. Caso os critérios de convergência não sejam alcançados o programa irá parar a sua execução e uma mensagem de erro será emitida.

Restrições do Modelo

As seguintes condições deverão ser satisfeitas durante o uso desta isoterma:

$$\begin{aligned} &C \geq 0 \quad ; \\ &q_{max} > 0 \quad \text{e}, \\ &K_1 > 0. \end{aligned}$$

No caso de alguma destas restrições ser violada, o programa irá parar a sua execução e uma mensagem de erro irá ser emitida.

Exemplo de Aplicação do Modelo

Código 4.2: Código C++ empregado na geração das curvas da Figura 4.1

```
1  // include da lib c++
2  #include <iomanip>
3  #include <iostream>
4  #include <string>
5
6
7  // include da isotherm++
8  #include <Isotherm/Misc/Misc.h>
9  #include <Isotherm/Isotherm/TwoParameters/Elovich.h>
10
11 int main(int argc, char** argv) {
12
13 // Definição dos parâmetros
14
15 const Real CE(2e-3);       // Concentração de equilíbrio
16                             // do adsorvato no adsorvente em [mg / L]
17 const Real QMAX(150e-3);   // Quantidade de adosrvato em equilíbrio [mol/g]
18 const Real K1(1e-8);       // Constante de Elovich
19
20 const ist::Elovich isoterma(QMAX, K1);
21 const Real qe = isoterma.Qe(CE);
22
23 // Impressão das informações sobre as constantes do modelo
24
```

```
25    std::cout << "\n";
26    PrintLine(std::cout);
27    std::cout << "Impressao de informacoes sobre as constantes\n";
28    PrintLine(std::cout);
29
30    std::cout  << isoterma
31          << std::endl;
32
33    std::cout  << "Numero de constantes de variavel isoterma: "
34          << isoterma.NumberConst()
35          << std::endl;
36
37    std::cout << std::setfill(' ');
38    std::cout << "Impressao de informacoes da isoterma\n"
39         << std::setw(20)
40         << std::left
41         << "Parametro"
42         << std::setw(40)
43         << " Detalhe do parametro"
44         << std::right
45         << std::endl;
46
47    for (auto x : isoterma) {
48      std::cout  << std::setw(20)
49            << std::left
50            << x.first
51            << std::setw(40)
52            << x.second
53            << std::right
54            << "\n";
55    }
56
57
58    std::cout << "Concentracao de equilibrio: " << CE
59         << " tem-se que qe = " << qe
60         << std::endl;
```

4.1.3 Isoterma de Freundlich

A isoterma de Freundlich [12] possui a seguinte equação:

$$q = K_1 C^{1/K_2}, \tag{4.6}$$

Foi o primeiro modelo a descrever a adsorção de um gás em um soluto, porém a grande restrição deste modelo é não possuir um limite máximo para adsorção em sua equação. Os parâmetros e restrições

Restrições do Modelo

As seguintes condições deverão ser satisfeitas durante o uso desta isoterma [13]:

$$\begin{aligned} K_1 &> 0 \quad ; \\ K_2 &> 0 \quad \text{e}, \\ C &> 0. \end{aligned}$$

No caso de alguma destas restrições ser violada, o programa irá parar a sua execução e uma mensagem de erros irá ser emitida.

Exemplo de Aplicação do Modelo

Código 4.3: Código C++ empregado na geração das curvas da Figura 4.1

```
1  // include da lib c++
2  #include <iomanip>
3  #include <iostream>
4  #include <string>
5
6  // include da isotherm++
7  #include <Isotherm/Misc/Misc.h>
8  #include <Isotherm/Isotherm/TwoParameters/Freundlich.h>
9
10 int main(int argc, char** argv) {
11
12 // Definição dos parâmetros
13
14 const Real CE(2e-3);          // Concentração de equilíbrio
15                                // do adsorvato no adsorvente em [mg / L]
16
17 const Real K1(1e-8);          // Constante 1 de Freundlich
18 const Real K2(150e-3);        // Constante 2 de Freundlich
19
20 const ist::Freundlich    isoterma(K1, K2);
21 const Real               qe = isoterma.Qe(CE);
22
23 // Impressão das informações sobre as constantes do modelo
24
25     std::cout << "\n";
26     PrintLine(std::cout);
27     std::cout << "Impressao de informacoes sobre as constantes\n";
28     PrintLine(std::cout);
29
30     std::cout   << isoterma
31                 << std::endl;
32
33     std::cout   << "Numero de constantes de variavel isoterma: "
34                 << isoterma.NumberConst()
```

```
35                  << std::endl;
36
37      std::cout << std::setfill(' ');
38      std::cout << "Impressao de informacoes da isoterma\n"
39                << std::setw(20)
40                << std::left
41                << "Parametro"
42                << std::setw(40)
43                << " Detalhe do parametro"
44                << std::right
45                << std::endl;
46
47      for (auto x : isoterma) {
48          std::cout   << std::setw(20)
49                      << std::left
50                      << x.first
51                      << std::setw(40)
52                      << x.second
53                      << std::right
54                      << "\n";
55      }
56
57
58      std::cout << " Concentracao de equilibrio: " << CE
59                << " tem-se que qe = " << qe
60                << std::endl;
```

4.1.4 Isoterma de Halsey

A equação para este modelo é dada em [14]:

$$\ln(q) = \frac{1}{K_2}\ln(K_1) - \frac{1}{K_2}\ln(C), \tag{4.7}$$

Os parâmetros e as restrições deste modelo são:

$K_1 > 0$ – Constante da isoterma de Halsey.

$K_2 > 0$ – Constante de ligação de equilíbrio isotérmica.

Restrições do Modelo

As seguintes condições deverão ser satisfeitas durante o uso desta isoterma [13]:

$$K_1 > 0 \quad \text{e},$$
$$K_2 > 0,$$

No caso de alguma destas restrições ser violada, o programa irá parar a sua execução e uma mensagem de erros irá ser emitida.

Exemplo de Aplicação do Modelo

Código 4.4: Código C++ empregado na geração das curvas da Figura 4.1

```
1  // include da lib c++
2  #include <cmath>
3  #include <iomanip>
4  #include <iostream>
5  #include <string>
6
7  // include da isotherm++
8  #include <Isotherm/Misc/Misc.h>
9  #include <Isotherm/Isotherm/TwoParameters/Halsey.h>
10
11 // Definição dos parâmetros
12
13 const Real CE(2e-3);          // Concentracao de equilibrio
14                               // do adsorvato no adsorvente em [mg / L]
15
16 const Real K1(1e-8);          // Constante 1 de Halsey
17 const Real K2(150e-3);        // Constante 2 de Halsey
18
19 const ist::Halsey        isoterma(K1, K2);
20 const Real               qe = isoterma.Qe(CE);
21
22 // Impressão das informações sobre as constantes do modelo
23
24     std::cout << "\n";
25     PrintLine(std::cout);
26     std::cout << "Impressao de informacoes sobre as constantes\n";
27     PrintLine(std::cout);
28
29     std::cout   << isoterma
30                 << std::endl;
31
32     std::cout   << "Numero de constantes de variavel isoterma: "
33                 << isoterma.NumberConst()
34                 << std::endl;
35
36     std::cout << std::setfill(' ');
37     std::cout << "Impressao de informacoes da isoterma\n"
38               << std::setw(20)
39               << std::left
40               << "Parametro"
41               << std::setw(40)
42               << " Detalhe do parametro"
43               << std::right
44               << std::endl;
45
46     for (auto x : isoterma) {
```

```
47          std::cout   << std::setw(20)
48                      << std::left
49                      << x.first
50                      << std::setw(40)
51                      << x.second
52                      << std::right
53                      << "\n";
54      }
55
56
57      std::cout << " Concentracao de equilibrio: " << CE
58                << " tem-se que qe = " << qe
59                << std::endl;
```

4.1.5 Isoterma de Harkins-Jura

A equação para este modelo encontra-se em [15]:

$$\frac{1}{q^2} = \frac{K_2}{K_1} - \frac{1}{K_1}\log(C) \quad \text{com} \quad q > 0 \quad \text{e} \quad \log(C) < K_2, \tag{4.8}$$

Os parâmetros e as restrições deste modelo são:

$K_1 > 0$ – Constante da isoterma de Harkins-Jura.

$K_2 \geq 0$ – Constante da isoterma de Harkins-Jura.

Restrições do Modelo

As seguintes condições deverão ser satisfeitas durante o uso desta isoterma [13]:

$$\begin{aligned} &K_1 > 0 \quad ; \\ &K_2 > 0 \quad \text{e}, \\ &C > 0. \end{aligned}$$

No caso de alguma destas restrições ser violada, o programa irá parar a sua execução e uma mensagem de erros irá ser emitida.

Exemplo de Aplicação do Modelo

Código 4.5: Código C++ empregado na geração das curvas da Figura 4.1

```
1  // include da lib c++
2  #include <cmath>
3  #include <iomanip>
4  #include <iostream>
5  #include <string>
6
```

```
7  // include da isotherm++
8  #include <Isotherm/Misc/Misc.h>
9  #include <Isotherm/Isotherm/TwoParameters/HarkinsJura.h>
10
11 // Definição dos parâmetros
12
13 const Real CE(2e-3);          // Concentracao de equilibrio
14                               // do adsorvato no adsorvente em [mg / L]
15
16 const Real K1(1e-8);          // Constante 1 de Harkins-Jura
17 const Real K2(150e-3);        // Constante 2 de Harkins-Jura
18
19 const ist::HarkinsJura    isoterma(K1, K2);
20 const Real                qe = isoterma.Qe(CE);
21
22 // Impressão das informações sobre as constantes do modelo
23
24     std::cout << "\n";
25     PrintLine(std::cout);
26     std::cout << "Impressao de informacoes sobre as constantes\n";
27     PrintLine(std::cout);
28
29     std::cout   << isoterma
30                 << std::endl;
31
32     std::cout   << "Numero de constantes de variavel isoterma: "
33                 << isoterma.NumberConst()
34                 << std::endl;
35
36     std::cout << std::setfill(' ');
37     std::cout << "Impressao de informacoes da isoterma\n"
38               << std::setw(20)
39               << std::left
40               << "Parametro"
41               << std::setw(40)
42               << " Detalhe do parametro"
43               << std::right
44               << std::endl;
45
46     for (auto x : isoterma) {
47         std::cout   << std::setw(20)
48                     << std::left
49                     << x.first
50                     << std::setw(40)
51                     << x.second
52                     << std::right
53                     << "\n";
54     }
55
```

```
56
57      std::cout << " Concentracao de equilibrio: " << CE
58                << " tem-se que qe = " << qe
59                << std::endl;
```

4.1.6 Isoterma de Jovanović

A equação para este modelo é [16]:

$$\ln(q) = \ln(q_{max}) - K_1 C \quad \text{com} \quad q > 0 \quad \text{e} \quad C \geq 0, \tag{4.9}$$

Os parâmetros e as restrições deste modelo são [17]:

$q_{max} > 0$ – Capacidade máxima de adsorção.

$K_1 > 0$ – Constante da isoterma de Jovanovic.

Exemplo de Aplicação do Modelo

Código 4.6: Código C++ empregado na geração das curvas da Figura 4.1

```
1  // include da lib c++
2  #include <cmath>
3  #include <iomanip>
4  #include <iostream>
5  #include <string>
6
7  // include da isotherm++
8  #include <Isotherm/Misc/Misc.h>
9  #include <Isotherm/Isotherm/TwoParameters/Jovanovic.h>
10
11 // Definição dos parâmetros
12
13 const Real CE(2e-3);          // Concentracao de equilibrio
14                               // do adsorvato no adsorvente em [mg / L]
15
16 const Real K1(1e-8);          // Constante 1 de Jovanovic
17 const Real K2(150e-3);        // Constante 2 de Jovanovic
18
19 const ist::Jovanovic     isoterma(K1, K2);
20 const Real               qe = isoterma.Qe(CE);
21
22 // Impressão das informações sobre as constantes do modelo
23
24      std::cout << "\n";
25      PrintLine(std::cout);
26      std::cout << "Impressao de informacoes sobre as constantes\n";
27      PrintLine(std::cout);
28
```

```
29      std::cout   << isoterma
30                  << std::endl;
31
32      std::cout   << "Numero de constantes de variavel isoterma: "
33                  << isoterma.NumberConst()
34                  << std::endl;
35
36      std::cout << std::setfill(' ');
37      std::cout << "Impressao de informacoes da isoterma\n"
38                << std::setw(20)
39                << std::left
40                << "Parametro"
41                << std::setw(40)
42                << " Detalhe do parametro"
43                << std::right
44                << std::endl;
45
46      for (auto x : isoterma) {
47          std::cout   << std::setw(20)
48                      << std::left
49                      << x.first
50                      << std::setw(40)
51                      << x.second
52                      << std::right
53                      << "\n";
54      }
55
56
57      std::cout << " Concentracao de equilibrio: " << CE
58                << " tem-se que qe = " << qe
59                << std::endl;
```

4.1.7 Isoterma de Langmuir

A isoterma de Langmuir [18] é provavelmente a mais utilizada entre todas os modelos de isotermas. Sua equação é:

$$\frac{C}{q} = \frac{1}{q_{max}K_1} + \frac{C}{q_{max}}, \tag{4.10}$$

logo,

$$q = q_{max}\frac{K_1 C}{1 + K_1 C} \quad \text{com} \quad q \geq 0 \quad \text{e} \quad C \geq 0, \tag{4.11}$$

Os parâmetros e restrições deste modelo são [19]:

$q_{max} > 0$ – Capacidade máxima de adsorção em $[M/M]$.

$K_1 > 0$ – Constante de equilíbrio de Langmuir em $[L^3/M]$.

Exemplo de Aplicação do Modelo

Código 4.7: Código C++ empregado na geração das curvas da Figura 4.1

```
1  // include da lib c++
2  #include <cmath>
3  #include <iomanip>
4  #include <iostream>
5  #include <string>
6
7
8  // include da isotherm++
9  #include <Isotherm/Misc/Misc.h>
10 #include <Isotherm/Isotherm/TwoParameters/Languimuir.h>
11
12 int main(int argc, char** argv) {
13
14 // Definição dos parâmetros
15
16 const Real CE(2e-3);          // Concentração de equilíbrio
17                               // do adsorvato no adsorvente em [mg / L]
18 const Real QMAX(150e-3);      // Quantidade de adosrvato em equilíbrio [mol/g]
19 const Real K1(1e-8);          // Constante de Languimuir
20
21 const ist::Elovich isoterma(QMAX, K1);
22 const Real qe = isoterma.Qe(CE);
23
24 // Impressão das informações sobre as constantes do modelo
25
26     std::cout << "\n";
27     PrintLine(std::cout);
28     std::cout << "Impressao de informacoes sobre as constantes\n";
29     PrintLine(std::cout);
30
31     std::cout   << isoterma
32                 << std::endl;
33
34     std::cout   << "Numero de constantes de variavel isoterma: "
35                 << isoterma.NumberConst()
36                 << std::endl;
37
38     std::cout << std::setfill(' ');
39     std::cout << "Impressao de informacoes da isoterma\n"
40               << std::setw(20)
41               << std::left
42               << "Parametro"
43               << std::setw(40)
44               << " Detalhe do parametro"
45               << std::right
46               << std::endl;
```

```
47
48      for (auto x : isoterma) {
49          std::cout   << std::setw(20)
50                      << std::left
51                      << x.first
52                      << std::setw(40)
53                      << x.second
54                      << std::right
55                      << "\n";
56      }
57
58
59      std::cout << "Concentracao de equilibrio: " << CE
60                << " tem-se que qe = " << qe
61                << std::endl;
```

4.1.8 Isoterma de Temkin

A equação para este modelo é dada por [20]:

$$q = \frac{RT}{K_2} \ln(K_1 C) \quad \text{com} \quad q \geq 0 \quad , \quad K_1 C \geq 1 \quad \text{e} \quad T > 0, \tag{4.12}$$

em que R foi definido na seção 4.1.1.

Os parâmetros e as restrições deste modelo são [21]:

$K_1 > 0$ – Constante da isoterma de Temkin.

$K_2 > 0$ – Constante de ligação de equilíbrio isotérmica.

Exemplo de Aplicação do Modelo

Código 4.8: Código C++ empregado na geração das curvas da Figura 4.1

```
1  // include da lib c++
2  #include <cmath>
3  #include <iomanip>
4  #include <iostream>
5  #include <string>
6
7  // include da isotherm++
8  #include <Isotherm/Misc/Misc.h>
9  #include <Isotherm/Isotherm/TwoParameters/Temkin.h>
10
11 int main(int argc, char** argv) {
12
13
14 // Definição dos parâmetros
15
```

```
16 const Real TEMP(400);     // Temperatura em kelvin
17 const Real CE(2e-3);      // Concentração de equilíbrio
18                           // do adsorvato no adsorvente em [mg / L]
19 const Real QMAX(15e-2); // Quantidade de adosrvato em equilíbrio [mol/g]
20 const Real K1(1e-8);      // Constante de Temkin
21
22 const ist::Temkin                  isoterma(QMAX, K1);
23 const Real                         qe = isoterma.Qe(CE, TEMP);
24
25 // Impressão das informações sobre as constantes do modelo
26
27     std::cout << "\n";
28     PrintLine(std::cout);
29     std::cout << "Impressao de informacoes sobre as constantes\n";
30     PrintLine(std::cout);
31
32     std::cout   << isoterma
33                 << std::endl;
34
35     std::cout   << "Numero de constantes de variavel isoterma: "
36                 << isoterma.NumberConst()
37                 << std::endl;
38
39     std::cout << std::setfill(' ');
40     std::cout << "Impressao de informacoes da isoterma\n"
41               << std::setw(20)
42               << std::left
43               << "Parametro"
44               << std::setw(40)
45               << " Detalhe do parametro"
46               << std::right
47               << std::endl;
48
49     for (auto x : isoterma) {
50         std::cout   << std::setw(20)
51                     << std::left
52                     << x.first
53                     << std::setw(40)
54                     << x.second
55                     << std::right
56                     << "\n";
57     }
58
59
60     std::cout << "\n\nPara a temperatura: " << TEMP
61               << " e concentracao de equilibrio: " << CE
62               << " tem-se que qe = " << qe
63               << std::endl;
```

Referências Bibliográficas

[1] L. B. Brum, L. Rodrigues, A. J. Silva Neto, J. F. Vasconcellos em XXIII Encontro nacional de modelagem computacional, Palmas, **2020**.

[2] L. R. F. Alleoni, O. A. Camargo, J. C. Casagrande, "Isotermas de Langmuir e de Freundlich na descrição da adsorção de boro em solos altamente intemperizados", *Scientia Agricola* **1998**, *55*, 379–387, DOI 10.1590/S0103-90161998000300005.

[3] W. Henry, "Experiments on the quantity of gases absorbed by water, at different temperatures, and under different pressures", *Philosophical Transactions of the Royal Society of London* **1803**, *93*, 29–274, DOI 10.1098/rstl.1803.0004.

[4] M. M. Dubinin, "The potential theory of adsorption of gases and vapors for adsorbents with energetically nonuniform surfaces.", *Chemical Reviews* **1960**, *60*, 235–241, DOI 10.1021/cr60204a006.

[5] X. Zhou, "Correction to the calculation of Polanyi potential from Dubinnin-Radushkevich equation", *Journal of Hazardous Materials* **2020**, *384*, 121101, DOI 10.1016/j.jhazmat.2019.121101.

[6] R. K. Gautam, M. C. Chattopadhyaya, "Kinetics and equilibrium isotherm modeling: Graphene-based nanomaterials for the removal of heavy metals from water" em *Nanomaterials for Wastewater Remediation*, Elsevier, **2016**, pp. 79–109, DOI 10.1016/B978-0-12-804609-8.00005-4.

[7] N. Ayawei, A. N. Ebelegi, D. Wankasi, "Modelling and interpretation of adsorption isotherms", *Journal of Chemistry* **2017**, *2017*, 1–11, DOI 10.1155/2017/3039817.

[8] Q. Hu, Z. Zhang, "Application of Dubinin–Radushkevich isotherm model at the solid/solution interface: A theoretical analysis", *Journal of Molecular Liquids* **2019**, *277*, 646–648, DOI 10.1016/j.molliq.2019.01.005.

[9] S. Y. Elovich, O. G. Larionov, "Theory of adsorption from nonelectrolyte solutions on solid adsorbents", *Bulletin of the Academy of Sciences of the USSR Division of chemical science* **1962**, *11*, 191–197, DOI 10.1007/BF00908016.

[10] C. Aharoni, F. Tompkins, "Kinetics of adsorption and desorption and the Elovich equation" em, **1970**, pp. 1–49, DOI 10.1016/S0360-0564(08)60563-5.

[11] B. Carnahan, J. O. Wilkes, *Digital computing and numerical methods*, John Wiley & Sons, New York, **1973**.

[12] H. Freundlich, "Über die Adsorption in Lösungen", *Zeitschrift für Physikalische Chemie* **1907**, *57U*, 1454, DOI 10.1515/zpch-1907-5723.

[13] D. L. Sparks, "Sorption phenomena on soils" em *Environmental Soil Chemistry*, Elsevier, **2003**, pp. 133–186, DOI 10.1016/B978-012656446-4/50005-0.

[14] G. Halsey, "Physical adsorption on non-uniform surfaces", *The Journal of Chemical Physics* **1948**, *16*, 931–937, DOI 10.1063/1.1746689.

[15] W. D. Harkins, G. Jura, "Surfaces of solids. XIII. A vapor adsorption method for the determination of the area of a solid without the assumption of a molecular area, and the areas occupied by nitrogen and other molecules on the surface of a solid", *Journal of the American Chemical Society* **1944**, *66*, 1366–1373, DOI 10.1021/ja01236a048.

[16] D. S. Jovanović, "Physical adsorption of gases", *Kolloid-Zeitschrift & Zeitschrift für Polymere* **1969**, *235*, 1214–1225, DOI 10.1007/BF01542531.

[17] M. A. Aly-Eldeen, A. A. M. El-Sayed, D. M. S. A. Salem, G. M. El Zokm, "The uptake of Eriochrome Black T dye from aqueous solutions utilizing waste activated sludge: Adsorption process optimization using factorial design", *The Egyptian Journal of Aquatic Research* **2018**, *44*, 179–186, DOI 10.1016/j.ejar.2018.09.001.

[18] I. Langmuir, "The adosrption of gases on plane surfaces of glass, mica and platinum", *Journal of the American Chemical Society* **1918**, *40*, 1361–1403, DOI 10.1021/ja02242a004.

[19] X. Guo, J. Wang, "Comparison of linearization methods for modeling the Langmuir adsorption isotherm", *Journal of Molecular Liquids* **2019**, *296*, 111850, DOI 10.1016/j.molliq.2019.111850.

[20] R. D. Johnson, F. H. Arnold, "The temkin isotherm describes heterogeneous protein adsorption", *Biochimica et Biophysica Acta (BBA) - Protein Structure and Molecular Enzymology* **1995**, *1247*, 293–297, DOI 10.1016/0167-4838(95)00006-G.

[21] C. S. T. Araújo, I. L. S. Almeida, H. C. Rezende, S. M. L. O. Marcionilio, J. J. L. Léon, T. N. Matos, "Elucidation of mechanism involved in adsorption of Pb(II) onto lobeira fruit (Solanum lycocarpum) using Langmuir, Freundlich and Temkin isotherms", *Microchemical Journal* **2018**, *137*, 348–354, DOI 10.1016/j.microc.2017.11.009.