Universidade Federal de Pernambuco
Centro de Ciências Exatas e da Natureza
Programa de Pós Graduação em Matemática

Marcelo Pirôpo da Silva

SURFANDO EM SUPERFÍCIES

MODELAGEM MATEMÁTICA EM ESPAÇOS CURVOS E APLICAÇÕES

Recife

2019

Marcelo Pirôpo da Silva

SURFANDO EM SUPERFÍCIES

MODELAGEM MATEMÁTICA EM ESPAÇOS CURVOS E APLICAÇÕES

Tese apresentada ao Programa de Pós-graduação em Matemática da Universidade Federal de Pernambuco como requisito parcial para obtenção do título de Doutor em Matemática.

Área de Concentração: Geometria
Orientador: Dr. Fernando A. Nóbrega Santos
Coorientador: Dr. Fernando Jorge Sampaio Moraes

Recife

2019

Catalogação na fonte
Bibliotecária Mariana de Souza Alves CRB4-2106

S586s Silva, Marcelo Pirôpo da
Surfando em superfícies: modelagem matemática em espaços curvos e aplicações/ Marcelo Pirôpo da Silva – 2019.
76 f.: il., fig.

Orientador: Fernando A. Nóbrega Santos
Tese (Doutorado) – Universidade Federal de Pernambuco. CCEN, Matemática. Recife, 2019.
Inclui referências e apêndices.

1. Geometria. 2. Modelagem Matemática em espaços curvos. 3. Difusão em espaços curvos. 4. Biologia Matemática. I. Santos, Fernando A. Nóbrega (orientador). II. Título.

516 CDD (22. ed.) UFPE-MEI 2019-162

MARCELO PIRÔPO DA SILVA

SURFANDO EM SUPERFÍCIES: MODELAGEM MATEMÁTICA EM ESPAÇOS CURVOS E APLICAÇÕES

Tese apresentada ao Programa de Pós-graduação do Departamento de Matemática da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutorado em Matemática.

Aprovado em: 28/02/2019

BANCA EXAMINADORA

_____

Prof. Dr. Fernando Antônio Nóbrega Santos (Orientador)
Universidade Federal de Pernambuco

_____

Prof. Dr. Antônio Fernando Pereira de Sousa (Examinador Interno)
Universidade Federal de Pernambuco

_____

Prof. Dr. Jones Oliveira de Albuquerque (Examinador Externo)
Universidade Federal de Pernambuco

_____

Prof. Dr. Jonas Romero Fonseca de Lima (Examinador Externo)
Universidade Federal Rural de Pernambuco

_____

Prof.Dr. Antônio de Pádua Santos (Examinador Externo)
Universidade Federal Rural de Pernambuco

A quem se alegrar!

AGRADECIMENTOS

A Deus! A Ele agradeço por tudo. Aqui, evidencio minha gratidão por me dar serenidade para superar todos os momentos difíceis, quando muitas vezes me sentir cansado e incapaz, ajudou-me a não desanimar!

Agradeço a minha família, em especial a minha mãe, irmão e sobrinho, pelo apoio incondicional.

Agradeço aos meus amigos e irmãos em Cristo, em especial, Rubem, Felipe e Sidney.

Um agradecimento especial a Rubia que desde o início do mestrado passamos a estudar juntos, nos tornamos amigos e irmãos. Sou muito grato por nossa amizade e que possa sempre contar comigo.

Agradeço a todos os colegas de doutorado.

Agradeço a todos os professores que tive, pois foram realmente muito importantes. No entanto, deixo registrado aqui, meu agradecimento aos professores: Antonio Fernando (Tony), Henrique Araujo, Airton Temistocles e a Miguel Fidencio.

Agradeço ao professor Dr. Fernando Moraes pela coorientação fundamental para a conclusão deste trabalho.

Agradeço ao professor Fernando Antonio Nóbrega por ter me orientado e por todo apoio, motivação e por suas ideias, o que nos permitiu concluir este trabalho. Agradeço a CAPES e ao CNPq pelo auxílio financeiro.

"... uma imensa biblioteca, repleta de livros em muitas línguas... sabe que alguém deve ter escrito aqueles livros, mas não sabe como. Não compreende as línguas em que foram escritos... suspeita de que a disposição dos livros obedece a uma ordem misteriosa, mas não sabe qual ela é"

(Albert Einstein [1])

RESUMO

Estruturas curvas são onipresentes na natureza, particularmente em matemática aplicada. Os sistemas em que estão presentes incluem membranas, interfaces, espaços-tempo curvos, mecânica dos fluidos, etc. Em contraste, os aspectos exaustivos da implementação de quantidades geométricas tornam o uso de geometria diferencial muitas vezes inviáveis para modelar sistemas em espaços curvos, sendo as aplicações limitadas a superfícies triviais. Neste trabalho, apresentamos o Surf, um pacote de matemática simbólica que visa contornar essa dificuldade, com funções para facilitar a modelagem interdisciplinar em sistemas curvos. A idéia principal é ter uma superfície parametrizada como entrada e o modelo de interesse escrito no plano. Como saída, obtemos o sistema modelo na superfície curva desejada. Os operadores usuais para uma determinada superfície são implementados, de modo que um modelo arbitrário possa ser imediatamente mapeado do espaço plano para o espaço curvo. A simplicidade e utilidade de nossa abordagem é ilustrada com algumas aplicações em uma variedade de problemas de pesquisa interdisciplinares: difusão em um espaço curvo, deslocamento quadrático médio, mecânica quântica em superfícies e modelagem da trajetória do C. elegans. Em algumas das aplicações, estendemos resultados da literatura para domínios maiores. Esperamos que este pacote contribua para facilitar a implementação da modelagem matemática em superfícies, cuja demanda vem crescendo de maneira geral.

Palavras-chave: Modelagem Matemática em espaços curvos. Difusão em espaços curvos. Deslocamento Quadrático Médio. Biologia Matemática. Mecânica Quântica em Superfícies.

ABSTRACT

Curved structures are ubiquitous in nature, particularly in applied mathematics. The systems where they are present include membranes, interfaces, curved spacetimes, fluid mechanics, etc. In contrast, the exhaustive aspects of implementing geometric quantities make the use of differential geometry often infeasible for modeling systems in curved spaces, with applications limited to trivial surfaces. In this work, we present Surf, a symbolic mathematical package that aims to overcome this difficulty, with functions to facilitate interdisciplinary modeling in curved systems. The main idea is to have a parameterized surface as input and the model of interest written in the plane. As an output, we obtain the model system on the desired curved surface. The usual operators for a given surface are implemented, so that an arbitrary model can be immediately mapped from flat space to curved space. The simplicity and usefulness of our approach is illustrated with some applications in a variety of interdisciplinary research problems: diffusion in a curved space, mean square displacement, quantum mechanics in surfaces and modeling of the C. Elegans trajectory. In some applications, we extend literature results for larger domains. We hope that this package contributes to facilitating the implementation of mathematical modeling on surfaces, whose demand has been growing in general.

Keywords: Mathematical modeling in curved spaces. Diffusion in curved spaces. Mean square displacement. Mathematical Biology. Quantum Surface Mechanics.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Geometria diferencial tem aplicações em diversas áreas, incluindo mecânica quântica [2], difusão [3], biologia [4, 5], relatividade geral [6], etc. Em muitas áreas de conhecimento, um sistema modelo sobre um plano pode ser estendido a uma superfície curva com ajuda da geometria diferencial. Isso torna o modelo mais realista, já que sistemas naturais geralmente são corrugados. Superfícies simples como uma esfera, um toro ou um cilindro elíptico, são descritas diretamente por coordenadas curvilíneas e ortogonais. Por outro lado, superfícies mais complexas, por exemplo, a caixa de ovos, requerem coordenadas não ortogonais, o que pode dificultar o cálculo de operadores diferenciais.

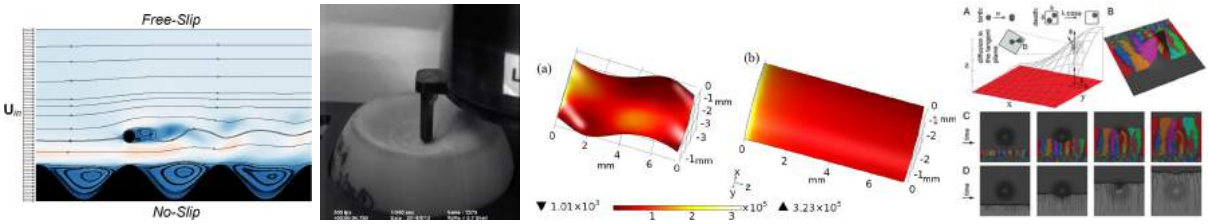
De fato, considerado o espaço-tempo curvo de Einstein, superfícies curvas estão presentes em todos os lugares. Por exemplo, no micro-mundo biológico, onde é possível observar as superfícies de membranas celulares e de músculos ventriculares (Figura 1)

Figura 1 – Exemplos de superfícies curvas (Biologia).

Fonte [7, 8].

Diante disso, em muitas pesquisas a influência de propriedades geométricas como a curvatura, são investigadas. Por exemplo (Figura 2), em trabalhos sobre: a amplitude da enrugues de uma parede sinusoidal as forças hidrodinâmicas exercidas em uma partícula fixada [9]; dobras criadas em superfícies curvas [10]; a transferência de calor em tubos corrugados [11] e sobre a evolução da expansão populacional em superfícies com curvatura [12].

Figura 2 – A influência da curvatura.



Fontes: [9–12].

Neste trabalho, apresentamos uma ferramenta computacional, chamada *Surf*, desti-

nada ao ensino e pesquisa da geometria diferencial de curvas e superfícies. Implementamos a ferramenta como um novo pacote para o sistema de álgebra computacional Maple.

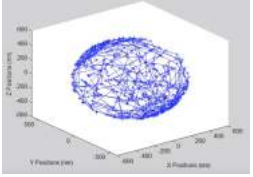
No pacote *Surf*, incluímos funções que calculam os principais operadores diferenciais sobre superfícies curvas: gradiente, rotacional, divergência, hessiana e o Laplaciano, sendo aplicados, dependendo de suas definições, em funções escalares, vetoriais ou matriciais. Além dessas, a partir de conceitos da geometria diferencial de curvas e superfícies, implementamos outras funções que calculam: o vetor normal a uma superfície; o comprimento do arco sobre superfícies; as curvaturas normal e geodésica; a curvatura de uma curva no plano, no espaço, ou em uma superfície; a torção (geodésica); o operador de Weingarten; as curvaturas gaussiana, média, e a segunda curvatura de Gauss; a primeira e a segunda forma fundamental, os símbolos de Christoffel e a equação da geodésica.

Além de uma linguagem de programação interna dedicada à matemática que permite criar funções, o Maple contém uma biblioteca com vários pacotes com funções para diversas áreas da matemática e suas aplicações. Em vista disso, com objetivo de aplicar o *Surf* a uma ampla gama de superfícies, escrevemos o código de modo que coordenadas não ortogonais possam ser usadas. Como aplicações das funções acima mencionadas, implementamos novos algoritmos tendo em mente dinâmica de fluidos, difusão, mecânica quântica e biologia. Esses algorítmos foram finalizados como novas funções para o *Surf*.

Para uma primeira aplicação, implementamos funções que calculam os operadores diferenciais que aparecem nas equações fundamentais da dinâmica de fluidos (as equações de movimento, energia e continuidade), incluindo uma função específica para o tensor de tensão viscosa.

Para difusão em superfícies curvas, que é essencial para processos industriais e biológicos, obtemos uma função que calcula a série para o deslocamento quadrático médio até um número arbitrário de termos obtidos primeiro por Castro-Villarreal [3], que calculou até três termos para uma esfera. Com a nossa função, apresentamos oito termos para a esfera, validando e ampliando o resultado da Ref. [3]. Desse modo, o *Surf* pode favorecer investigações teóricas e experimentais como realizada na referência [13], onde utilizaram a série para esfera obtida por [3], (Figura 3).

Figura 3 – Trajetória simulada de um objeto pontual que se difunde em uma gota esférica, obtida a partir da série do deslocamento quadrático médio.

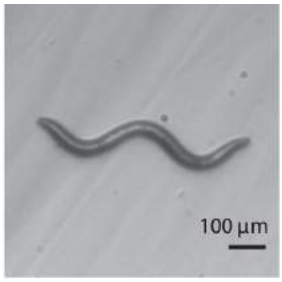


Fonte [13].

Para mecânica quântica, implementamos uma função que calcula o potencial geométrico obtido por da Costa em [2], e que tem sido usado em uma variedade de trabalhos (ver, por exemplo, [14–17]).

Finalmente, nossa aplicação biológica é sobre o Caenorhabditis elegans (C. elegans) que é um nematoide usado como sistema modelo ideal para investigações abrangentes de biologia (ver [5], [18]). Ressaltamos que o C. elegans (Figura 4) tem reconhecida importância para ciências biológicas, de modo que, por duas vezes esteve envolvido diretamente em estudos que resultaram no Prêmio Nobel de Fisiologia ou Medicina (2002 e 2006), além de ter sido utilizado no estudo que resultou no Premio Nobel de Química em 2008 [19].

Figura 4 – C. Elegans.



Fonte: [4].

Portanto, nosso exemplo de aplicação biológica é uma função que descreve as trajetórias (formas) do C. elegans, não só em duas dimensões como obtido em [4] a partir de sua curvatura, mas, considerando o avanço tecnológico de métodos como a microscopia que é utilizado em trabalhos sobre o C. elegans, implementamos uma função no pacote *Surf* para obter a trilha em três dimensões, isto supondo uma função a ser conhecida e que reflita a variação do nematoide no espaço.

No capítulo 2, revisamos alguns conceitos geométricos de curvas e superfícies, e então apresentamos as fórmulas que definem os operadores diferenciais para superfícies curvas que usamos para implementar o pacote *Surf*. No capítulo 3, apresentamos um resumo sobre sistemas de álgebra computacional. No capítulo 4, descrevemos a estrutura do pacote *Surf*, enfatizando as principais diferenças e novidades presentes no pacote, e os métodos/procedimentos utilizados para sua implementação, além de descrevermos todas as funções do *Surf*, enquanto o capítulo 5 contém as aplicações.

Finalmente, apresentamos nossas conclusões no capítulo 6, e incluímos um apêndice com algorítimos sobre o deslocamento quadrático médio e as trajetórias de Caenorhabditis elegans.

2 PRELIMINARES

Para apresentar o pacote *Surf*, recordamos, na primeira parte da próxima seção, vários conceitos e definições da geometria diferencial de curvas e superfícies. Estes, são bem conhecidos e podem ser encontrados em vários livros, como em [20–24]. No entanto, preferimos incluí-los para tornar esse texto autossuficiente.

2.1 Noções de geometria diferencial de curvas e superfícies

Um conjunto S de $\mathbb{R}^3$ é uma superfície regular se, para cada $p \in S$, existe uma vizinhança U de p em $\mathbb{R}^3$ e uma aplicação $\sigma : \mathcal{U} \subset \mathbb{R}^2 \to U \cap S$, onde $\mathcal{U}$ é um aberto de $\mathbb{R}^2$, tal que

1. $\sigma : \mathcal{U} \to S$ é diferenciável;
2. $\sigma : \mathcal{U} \to U \cap S$ é um homeomorfismo;
3. para cada $p \in \mathcal{U}$, a diferencial $d\sigma_p : \mathbb{R}^2 \to \mathbb{R}^3$ é injetiva.

A aplicação σ é dita uma parametrização da superfície S. Denotemos as funções coordenadas de σ por x, y, z, e escrevemos $\sigma(u, v) = [x(u, v), y(u, v), z(u, v)]$, $\sigma_u = \frac{\partial \sigma}{\partial u}$ e $\sigma_v = \frac{\partial \sigma}{\partial v}$.

O plano tangente a S em um ponto $p \in S$, denotado por T_pS, é o plano gerado por $\{\sigma_u, \sigma_v\}$ que contem p. A aplicação $<,>: T_pS \times T_pS \to \mathbb{R}$ dada por $< w_1, w_2 >= w_1 \cdot w_2$, em cada ponto $p \in S$, é uma métrica induzida pelo produto interno usual de $\mathbb{R}^3$ sobre S. As funções $g_{ij} =< w_i, w_j >$, $i, j = 1, 2$ são chamadas componentes (ou coeficientes) da métrica e são importantes para obter muitas das propriedades intrínsecas de uma superfície. A matriz da métrica é denotada por (g_{ij}), e sua inversa por (g^{ij}). Agora, sejam

$$E = ||\sigma_u||^2, \quad F =< \sigma_u, \sigma_v >, \quad G = ||\sigma_v||^2. \tag{2.1}$$

Em termos de diferenciais, a métrica tem uma expressão quadrática dada por

$$\mathbf{I} = Edu^2 + 2Fdudv + Gdv^2, \tag{2.2}$$

e é usualmente chamada de primeira forma fundamental da superfície S. Observe que, na maioria dos textos de física, esta é representada por ds^2.

Dos diferentes usos da primeira forma fundamental, seus coeficientes podem ser usados para determinar o comprimento de curvas sobre uma superfície S. Assim, dada

uma curva diferenciável $\gamma : [a, b] \subset \mathbb{R} \to \mathbb{R}^3$, dizemos que γ está sobre S se $\gamma(t) \in S$, onde $t \in [a, b]$. Neste caso, o comprimento exato de γ para $t \in [a, b]$ é dado por

$$L = \int_a^b \sqrt{E(u,v)\Big(\frac{du}{dt}\Big)^2 + 2F(u,v)\frac{du}{dt}\frac{dv}{dt} + G(u,v)\Big(\frac{dv}{dt}\Big)^2}. \tag{2.3}$$

Lembremos que o comprimento de arco de uma curva $\gamma(t)$ é a grandeza infinitesimal do vetor $\gamma'(t)$, isto é, $\int_a^b ||\gamma'(t)||dt$. Comparando com a fórmula para métrica induzida em uma superfície (2.2), ou seja, observando que $ds^2 = g_{ij}dudv = \mathbf{I}$, obtem-se a fórmula (2.3).

Dada uma curva diferenciável $\gamma : [a, b] \to \mathbb{R}^3$, a velocidade com que os vetores tangentes $\gamma'(t)$, $t \in (a, b)$, mudam de direção é chamada de curvatura da curva γ em t. Se γ é parametrizada pelo comprimento de arco, sua curvatura é dada por

$$k(t) = |\gamma''(t)|. \tag{2.4}$$

É fácil mostrar que se o traço de γ é uma reta, então sua curvatura é nula. Portanto, k é a função escalar que determina a extensão em que uma curva se desvia de uma linha reta. Mais geralmente, se $\gamma \in \mathbb{R}^3$ é uma curva diferenciável, não necessariamente parametrizada pelo comprimento de arco, a curvatura de γ pode ser obtida pela fórmula

$$k(t) = \frac{||\gamma'(t) \times \gamma''(t)||}{||\gamma'(t)||^3}. \tag{2.5}$$

Para uma curva regular em $\mathbb{R}^2$, sua curvatura é dada por

$$k(t) = \frac{\gamma''(t) \cdot J\gamma'(t)}{\| \gamma'(t) \|^3}, \tag{2.6}$$

onde

$$J = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},$$

é a estrutura linear complexa sobre um vetor (rotação por $\pi/2$ no sentido anti-horário).

Se a curva é um gráfico de uma função explícita $y = \varphi(x)$ sobre o plano, então

$$k(x) = \frac{|\varphi''(x)|}{(1 + \varphi'(x)^2)^{\frac{3}{2}}}. \tag{2.7}$$

Em coordenadas polares, com $\gamma(\theta) = \vec{r}(\theta)e^{i\theta}$, a curvatura da curva é

$$k(\theta) = \frac{2\vec{r}'(\theta)^2 - \vec{r}(\theta)\vec{r}''(\theta) + \vec{r}(\theta)^2}{(\vec{r}'(\theta)^2 + \vec{r}(\theta)^2)^{\frac{3}{2}}}. \tag{2.8}$$

Se uma curva $\gamma : [a, b] \to \mathbb{R}^3$ tem curvatura não nula, podemos definir o vetor normal unitário por

$$N(t) = \frac{\gamma''(t)}{k(t)}. \tag{2.9}$$

Para uma superfície S com parametrização σ, define-se o campo normal unitário por

$$N(u,v) = \frac{\sigma_u \times \sigma_v}{||\sigma_u \times \sigma_v||}, \tag{2.10}$$

e a segunda forma fundamental de S é escrita como

$$\mathbf{II} = edu^2 + 2fdudv + gdv^2, \tag{2.11}$$

onde $e =< \sigma_{uu}, N >,\ f =< \sigma_{uv}, N >$ e $\ g =< \sigma_{vv}, N >$, com $N := N(u,v)$.

Sendo N definido ao longo de uma curva γ sobre S, com $\gamma(0) = p$, a aplicação linear $W_p : T_pS \to T_pS$ definida por $W_p = -D_{\gamma'(0)}N$ nos diz como S se curva na direção $\gamma'(0)$, ou seja, nos diz sobre a forma da superfície S. W é chamada de aplicação de Weingarten ou operador de forma.

Como $W_p \in T_pS$, podemos escrever

$$W_p(\sigma_u) = a_1\sigma_u + a_2\sigma_v, \ \ e \ \ W_p(\sigma_v) = a_3\sigma_u + a_4\sigma_v. \tag{2.12}$$

. Logo,

$$\begin{aligned} e &= \mathbf{I}(W_p(\sigma_u), \sigma_u) = a_1E + a_3F, \\ g &= \mathbf{I}(W_p(\sigma_v), \sigma_u) = a_2F + a_4G, \\ f &= \mathbf{I}(W_p(\sigma_u), \sigma_v) = a_1F + a_3G, \\ f &= \mathbf{I}(W_p(\sigma_v), \sigma_u) = a_2E + a_4F. \end{aligned}$$

De onde, segue que

$$\begin{pmatrix} e & f \\ f & g \end{pmatrix} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4. \end{pmatrix} \tag{2.13}$$

Portanto, ao determinarmos os valores a_1, a_2, a_3 e a_4, obtemos que, em termos da base $\{\sigma_u, \sigma_v\}$, esse operador é dado por

$$[W_p] = \frac{1}{EG - F^2} \begin{bmatrix} fF - eG & eF - fE \\ gF - fG & fF - gE \end{bmatrix}. \tag{2.14}$$

Como W_p é um operador linear auto-adjunto em cada T_pS, existe uma base $\{w_1, w_2\}$ tal que $W_p(w_1) = \lambda_1 w_1$ e $W_p(w_2) = \lambda_2 w_2$. Os autovalores λ_1 e λ_2 são precisamente as curvaturas principais de S em p.

Denotando por T o vetor unitário γ', utilizamos os vetores N e T, calculados em um ponto sobre S, para obtermos o vetor binormal a γ dado por

$$B = T \times N. \tag{2.15}$$

Portanto, se a curvatura de γ é não nula, tem-se o referencial ortonormal $\{T, N, B\}$ chamado de Triedro de Frenet ao longo da curva γ.

Além disso, como B' é paralelo a N, defini-se o número real τ denominado torção da curva γ, pela equação

$$B'(t) = \tau(t)N(t). \tag{2.16}$$

Como uma curva contida em plano tem torção nula, segue que τ é a função escalar que determina a extensão em que uma curva não está contida em um plano. A torção pode ser calculada pela fórmula

$$\tau(t) = \frac{(\gamma'(t) \times \gamma''(t)) \cdot \gamma'''(t)}{||\gamma'(t) \times \gamma''(t)||^2}. \tag{2.17}$$

Derivando as funções vetoriais $T(s), N(s)$ e $B(S)$ conclui-se, a partir da equações (2.9),(2.15) e (2.16), que o triedro de Frenet $\{T(s), N(s), B(s)\}$ para uma curva γ, com curvatura $k(s)$ não nula, satisfaz o sistema de equações diferenciais

$$\begin{cases} T' = kN, \\ N' = -kT - \tau B, \qquad \text{(Equações de Frenet).} \\ B' = \tau N. \end{cases}$$

Seja $k(s) : I \to \mathbb{R}$ uma função diferenciável. O Teorema Fundamental das Curvas planas, mostra que existe exatamente uma curva $\gamma : I \to \mathbb{R}^2$, parametrizada pelo comprimento de arco s tal que a curvatura é $k(s)$. Um teorema análogo para curvas no espaço mostra que a curvatura $k(s) > 0$ junto com a torção $\tau(s) : I \to \mathbb{R}$ determinam uma única curva no espaço.

Denotando $\overline{B} = \gamma' \times N$, a torção geodésica τ_g da curva γ é número real dado por

$$\tau_g = \overline{B} \cdot N'. \tag{2.18}$$

Como γ'' tem uma componente paralela e outra perpendicular a S, escrevendo $\gamma''(t) = k_n N + k_g N \times \gamma'(t)$, os escalares k_n e k_g são chamados, respectivamente, curvatura normal e curvatura geodésica da curva γ, e são dados por

$$k_n = \gamma''(t) \cdot N(\gamma(t)), \qquad k_g = \frac{\gamma''(t) \cdot (N(\gamma(t)) \times \gamma'(t))}{\| \gamma'''(t) \|^3}. \tag{2.19}$$

Diferente da curvatura normal, que depende dos coeficientes da primeira e da segunda forma fundamental, a torção geodésica e a curvatura geodésica dependem apenas dos coeficientes da primeira forma, isto é, são intrínsecas a superfície. Quando γ está sobre S, a curvatura k_g mede como γ está se curvando sobre S.

Suponha que $\gamma(t) = \sigma(x^1(t), x^2(t))$ para $x^1, x^2 : I \to \mathbb{R}$. A curva γ é uma geodésica se, e só se

$$\frac{d^2x^k}{dt^2} + \Gamma_{ij}^k \sum_{i,j=1}^{2} \frac{dx^i}{dt}\frac{dx^j}{dt} = 0, \quad k = 1, 2, \tag{2.20}$$

onde $\Gamma_{ij}^m : \mathcal{U} \to \mathbb{R}$, $1 \leq i, j, k \leq 2$ são os simbolos de Christoffel (de segundo tipo), da parametrização $\sigma : \mathcal{U} \to U \cap S$ calculados em $(x^1(t), x^2(t))$, definidos em termos da métrica de S por

$$\Gamma_{ij}^m = \frac{1}{2} g^{km} \left\{ \frac{\partial g_{ik}}{\partial x^j} + \frac{\partial g_{jk}}{\partial x^i} - \frac{\partial g_{ij}}{\partial x^k} \right\}. \tag{2.21}$$

Uma propriedade extrínseca de S é a curvatura média, definida por

$$H = \frac{eG - fF + gE}{2(EG - F^2)}, \tag{2.22}$$

enquanto se olharmos, mais uma vez, para a geometria intrínseca de S, a curvatura de Gauss lê-se

$$K = \frac{eg - f^2}{EG - F^2}, \tag{2.23}$$

ou ainda, em termos dos coeficientes da primeira forma fundamental, K pode ser descrita por

$$\frac{1}{(EG - F^2)} \left\{ \begin{vmatrix} -\frac{1}{2}E_{vv} + F_{uv} - \frac{1}{2}G_{uu} & \frac{1}{2}E_u & F_u - \frac{1}{2}E_v \\ F_v - \frac{1}{2}G_u & E & F \\ -\frac{1}{2}G_v & F & G \end{vmatrix} - \begin{vmatrix} 0 & \frac{1}{2}E_v & \frac{1}{2}G_u \\ \frac{1}{2}E_v & E & F \\ \frac{1}{2}G_u & F & G \end{vmatrix} \right\}. \tag{2.24}$$

Podemos usar os dois conceitos anteriores para descrever as curvaturas principais de S como

$$k_1 = H + \sqrt{H^2 - K}, \qquad k_2 = H - \sqrt{H^2 - K}. \tag{2.25}$$

Da segunda forma fundamental, a segunda curvatura de Gauss de uma superfície S é definida por ([25, 26])

$$K_{II} = \frac{1}{(eg - f^2)^2} \left\{ \begin{vmatrix} -\frac{1}{2}e_{uu} + f_{uv} - \frac{1}{2}g_{vv} & \frac{1}{2}e_u - \frac{1}{2}e_u & \\ f_u - \frac{1}{2}g_u & e & f \\ \frac{1}{2}g_v & f & g \end{vmatrix} - \begin{vmatrix} 0 & -\frac{1}{2}e_v & \frac{1}{2}e]g_u \\ \frac{1}{2}e_v & e & f \\ \frac{1}{2}g_u & f & g \end{vmatrix} \right\}. \tag{2.26}$$

2.2 Operadores diferenciais sobre superfícies curvas

Como nosso principal objetivo é desenvolver uma ferramenta que facilite a tarefa de mapear sistemas modelos do espaço plano para um espaço curvo, um passo fundamental é obter os operadores diferenciais apropriados no espaço curvo. Embora essa etapa seja formalmente viável, a implementação analítica em uma superfície genérica pode ser bastante difícil, ou mesmo incômoda. Nesta seção, revisamos os principais operadores diferenciais em superfícies curvas que são implementados em nosso código (para mais detalhes veja [27–31]).

Gradiente

Com a mesma notação que usamos anteriormente, consideremos $U \subseteq \mathbb{R}^3$ um subconjunto aberto, com $U \cap S \neq \emptyset$.

Lembremos que um campo vetorial Y em um aberto $U \subseteq \mathbb{R}^3$ é uma função que atribui a cada ponto $p \in U$, um vetor tangente $Y \in T_pS$.

Seja $\varphi : U \subseteq \mathbb{R}^3 \to \mathbb{R}$ uma função escalar diferenciável. O gradiente de φ é o único campo vetorial diferenciável $\nabla\varphi$, definido sobre U, tal que

$$\nabla\varphi \cdot X = X(\varphi), \tag{2.27}$$

para todo campo vetorial X que opera em uma função escalar φ . O gradiente de φ pode ser calculado pela fórmula

$$\nabla\varphi = g^{ij}\frac{\partial\varphi}{\partial x^j}\partial_i. \tag{2.28}$$

Seja $V : U \subseteq \mathbb{R}^3 \to \mathbb{R}^3$ um campo vetorial (tensor de rank 1) definido por $V := V(v^1, v^2, v^3)$. O gradiente de V é a matriz (tensor de ordem 2) dada por

$$\nabla V = \nabla_j v^i g^j g_i, \tag{2.29}$$

onde $\nabla_j v^i = \partial_j v^i + \Gamma^i_{kj} v^k$, e g_i's são vetores básicos obtidos a partir da parametrização de S, não necessariamente unitários e mutualmente ortogonais. Enquanto a base dual é formada pelos vetores g^i's da matriz inversa de (g_i).

Divergência

A Divergência de um campo vetorial V, denotado por div V é a função escalar definida por

$$\text{div } V = \frac{1}{\sqrt{\mathbf{g}}}\frac{\partial}{\partial x_i}(v_i\sqrt{\mathbf{g}}), \tag{2.30}$$

onde $\mathbf{g}$ é o determinante da matriz (g_{ij}).

Seja $\mathcal{T}$ um tensor de ordem 2. A divergência covariante de $\mathcal{T}$ é o vetor cuja j-ésima coordenada é dada por

$$(\text{div } \mathcal{T})_j = \frac{\partial \mathcal{T}^{ij}}{\partial x^i} + \Gamma^i_{ki}\mathcal{T}^{kj} + \Gamma^j_{ki}\mathcal{T}^{ik}. \tag{2.31}$$

Rotacional (Curvilíneo)

O rotacional (ou curvilíneo) de um campo vetorial V é o campo dado por

$$\nabla \times V = \epsilon^{ijk}\nabla_i v_j g_k, \tag{2.32}$$

onde $\nabla_i v_j = \partial_i v_j - \Gamma^k_{ij} v_k$, e o símbolo de permutação (ou simbolo de Levi-Civita) ϵ^{ijk} é definido por: $\epsilon^{ijk} = \mathbf{g}^{-1/2}$, se (i, j, k) é uma permutação impar; $\epsilon^{ijk} = -\mathbf{g}^{-1/2}$ se é uma permutação par; ou $\epsilon^{ijk} = 0$, se dois dos índices são iguais, onde $i, j, k \in \{1, 2, 3\}$.

Laplace-Beltrami

Seja $\varphi : S \to \mathbb{R}$ um função diferenciável. O Laplaciano de φ é a função escalar $\Delta\varphi : S \to \mathbb{R}$, dada por

$$\Delta\varphi = \text{div}(\nabla\varphi). \tag{2.33}$$

ou seja, a divergência de um campo gradiente. Intuitivamente, nota-se que $\Delta\varphi$ é proporcional à diferença entre o valor médio de φ em uma vizinhança de um ponto p, e o valor $\varphi(p)$.

Em coordenadas, o Laplaciano de φ pode ser calculado pela fórmula

$$\Delta\varphi = \frac{1}{\sqrt{\mathbf{g}}}\frac{\partial}{\partial x_i}\Big(g^{ij}\sqrt{\mathbf{g}}\frac{\partial\varphi}{\partial x_j}\Big). \tag{2.34}$$

Se V é um campo vetorial, o Laplaciano de V é o campo vetorial dado por

$$\nabla^2 V = \nabla\Big(\nabla V\Big) - \nabla \times \Big(\nabla \times V\Big). \tag{2.35}$$

Hessiana

Dada uma função diferenciável $\varphi : S \to \mathbb{R}$, o Hessiano de φ é operador linear $(Hess\ \varphi)_p : T_pS \to T_pS$, $p \in S$, dado por

$$(Hess\ \varphi)_p(v) = \nabla_v \nabla\varphi. \tag{2.36}$$

onde ∇_v denota a derivada covariante. A matriz Hessiana de φ tem componentes

$$Hess(\varphi)_{ij} = g^{ij}\Big(\frac{\partial^2\varphi}{\partial x_i \partial x_j} - \Gamma^k_{ij}\frac{\partial\varphi}{\partial x_k}\Big). \tag{2.37}$$

Exemplo

Os operadores apresentados acima são bem conhecidos e frequentemente encontramos suas expressões para superfícies como esfera, cilindro, toro, etc. Aqui, vamos considerar um exemplo em coordenadas não ortogonais, o que pode ser útil para tornar esse texto autossuficiente. A princípio, notemos que os vetores de base covariantes e contravariantes g_i e g^i, não são necessariamente unitários. Sendo assim, considerando $|g_i| = \sqrt{g_i \cdot g_i}$ e $|g^i| = \sqrt{g^i \cdot g^i}$, temos os seguintes vetores unitários

$$\hat{g}_i \;=\; \frac{g_i}{\sqrt{g_{ii}}} \quad \text{e} \quad \hat{g}^i = \frac{g^i}{\sqrt{g^{ii}}}. \tag{2.38}$$

Logo, se escrevemos um vetor V nas bases dadas por g_i ou por g^i, isto é, $V = V^i g_i$ ou $V = V_i g^i$, obtemos, respectivamente $V = V^i\sqrt{g_{ii}}\hat{g}_i$ ou $V = V_i\sqrt{g^{ii}}\hat{g}^i$. Portanto, a respeito das componentes físicas covariantes e contravariantes de um vetor V e de um tensor $\mathcal{T}$, segue as seguintes relações ([28, 30–32]):

$$V^{(j)} \;=\; V^j|g^j| \quad \text{e} \quad V_{(j)} = \frac{V_j}{|g_{jj}|}, \tag{2.39}$$

$$\mathcal{T}^{(ij)} \;=\; \sqrt{g^{ii}}\sqrt{g^{jj}}\mathcal{T}^{ij}. \tag{2.40}$$

Como exemplo, seja $X(u, v, w) = [u, v, (uv)^2 + w]$. Se w é constante, $X := X(u, v)$ é uma parametrização da superfície (Figura 5) conhecida como calha cruzada [33].

Figura 5 – Calha cruzada.

Fonte: os autores (obtida com o sofware Maple).

Desde que w seja constante, obtemos $X_u = (1, 0, 2uv^2)$ e $X_v = (0, 1, 2vu^2)$. Logo, a matriz da primeira forma fundamental e sua inversa são facilmente calculadas:

$$(g_{ij}) = \begin{pmatrix} 1 + 4v^4u^2 & 4u^3v^3 \\ 4u^3v^3 & 1 + 4u^4v^2 \end{pmatrix} \quad \text{e} \quad (g^{ij}) = \begin{pmatrix} \frac{1+4v^2u^4}{1+4v^4u^2+4u^4v^2} & -\frac{4u^3v^3}{1+4v^4u^2+4u^4v^2} \\ -\frac{4u^3v^3}{1+4v^4u^2+4u^4v^2} & \frac{1+4u^2v^4}{1+4v^4u^2+4u^4v^2} \end{pmatrix}.$$

Seja $\varphi : U \cap S \to \mathbb{R}$ uma função escalar $\varphi := \varphi(u, v)$. Para calcular o gradiente de φ, utilizamos a equação (2.39) para as funções componentes $(\nabla\varphi)_i = \sqrt{g_{ii}}g^{ij}\frac{\partial\varphi}{\partial x^j}$. Logo,

$$\nabla\varphi = \begin{pmatrix} \frac{\sqrt{4\,u^2v^4+1}\left(4\,u^4v^2+1\right)\frac{\partial}{\partial u}\varphi(u,v)}{4\,u^4v^2+4\,u^2v^4+1} - \frac{4\sqrt{4\,u^2v^4+1}u^3v^3\frac{\partial}{\partial v}\varphi(u,v)}{4\,u^4v^2+4\,u^2v^4+1} \\ -\frac{4\sqrt{4\,u^4v^2+1}u^3v^3\frac{\partial}{\partial u}\varphi(u,v)}{4\,u^4v^2+4\,u^2v^4+1} + \frac{\sqrt{4\,u^4v^2+1}\left(4\,u^2v^4+1\right)\frac{\partial}{\partial v}\varphi(u,v)}{4\,u^4v^2+4\,u^2v^4+1} \end{pmatrix}. \tag{2.41}$$

Agora, supondo w variável, e derivando mais uma vez X, também com respeito a w, temos que $X_u = \langle 1, 0, 2uv^2 \rangle$, $X_v = \langle 0, 1, 2vu^2 \rangle$ e $X_w = \langle 0, 0, 1 \rangle$. Assim, a matriz da primeira forma de S passa a ser:

$$(g_{ij}) = \begin{pmatrix} 1 + 4v^4u^2 & 4u^3v^3 & 2uv^2 \\ 4u^3v^3 & 1 + 4u^4v^2 & 2vu^2 \\ 2uv^2 & 2vu^2 & 1 \end{pmatrix}. \tag{2.42}$$

Para obtermos a inversa (g^{ij}), denotemos por $L1, L2, L3$ os vetores linhas de (g_{ij}), e efetuamos as seguintes operações elementares: (1) $L1 := \frac{L1}{1+4v^4u^2}$, (2) $L2 := L2 - 4u^3v^3L1$, (3) $L3 := L3 - 2v^2uL1$, (4) $L2 := \frac{1+4v^4u^2}{4u^4v^2+4u^2v^4+1}L2$, (5)$L1 := L2 - \frac{4u^3v^3}{1+4v^4u^2}L2$, (6)$L3 := L3 - \frac{2vu^2}{4u^2v^4+1}L2$, (7)$L3 := (4u^4v^2 + 4u^2v^4 + 1)L3$, (8)$L2 := L2 - (\frac{2u^2v}{4u^4v^2+4u^2v^4+1})L3$, e por último (9) $L1 := L2 - (\frac{2uv^2}{4u^4v^2+4u^2v^4+1})L3$. Logo,

$$(g^{ij}) = \begin{pmatrix} 1 & 0 & -2uv^2 \\ 0 & 1 & -2vu^2 \\ -2uv^2 & -2vu^2 & 4u^4v^2 + 4u^2v^4 + 1 \end{pmatrix}. \tag{2.43}$$

É necessário calcularmos os símbolos de Christoffel para a superfície S. Assim,

$$\begin{aligned} \Gamma^3_{11} &= \Gamma^w_{uu} \\ &\quad -uv^2\frac{\partial}{\partial u}\left(4\,u^2v^4+1\right) - u^2v\left(2\,\frac{\partial}{\partial u}\left(4\,u^3v^3\right) - \frac{\partial}{\partial v}\left(4\,u^2v^4+1\right)\right) \\ &\quad +\frac{1}{2}\left(4\,u^4v^2+4\,u^2v^4+1\right)\left(2\,\frac{\partial}{\partial u}\left(2\,uv^2\right) - \frac{\partial}{\partial w}\left(4\,u^2v^4+1\right)\right) \\ &= 2v^2, \end{aligned}$$

$$\begin{aligned} \Gamma^3_{12} &= \Gamma^w_{uv} \\ &\quad -uv^2\frac{\partial}{\partial v}\left(4\,u^2v^4+1\right) - u^2v\frac{\partial}{\partial u}\left(4\,u^4v^2+1\right) \\ &\quad +2\left(\frac{\partial}{\partial u}\left(2\,u^2v\right) + \frac{\partial}{\partial v}\left(2\,uv^2\right) - \frac{\partial}{\partial w}\left(4\,u^3v^3\right)\right)\left(u^4v^2+u^2v^4+1/4\right) \\ &= 4uv, \\ \Gamma^3_{22} &= \Gamma^w_{vv} \\ &\quad -uv^2\left(2\,\frac{\partial}{\partial v}\left(4\,u^3v^3\right) - \frac{\partial}{\partial u}\left(4\,u^4v^2+1\right)\right) - u^2v\frac{\partial}{\partial v}\left(4\,u^4v^2+1\right) \\ &\quad +\frac{1}{2}\left(4\,u^4v^2+4\,u^2v^4+1\right)\left(2\,\frac{\partial}{\partial v}\left(2\,u^2v\right) - \frac{\partial}{\partial w}\left(4\,u^4v^2+1\right)\right) \\ &= 2u^2. \end{aligned}$$

Todos os demais símbolos Γ^3_{i3}, $\Gamma^3_{3,j}$, Γ^1_{ij} e Γ^2_{ij}, onde $(i, j = 1..3)$, são nulos.

Utilizamos as componentes da métrica da superfície S para obtermos os vetores $g^i = g^{ij}g_j$, $i, j = 1..3$.

$$\begin{aligned} g^1 &= g^{11}g_1 + g^{12}g_2 + g^{13}g_3 \\ &= \langle 1, 0, 0\rangle, \\ g^2 &= g^{21}g_1 + g^{22}g_2 + g^{23}g_3 \\ &= \langle 0, 1, 0\rangle, \\ g^3 &= g^{31}g_1 + g^{32}g_2 + g^{33}g_3 \\ &= \langle -2uv^2, -2vu^2, 1\rangle. \end{aligned} \tag{2.44}$$

Segue que $|g^1| = 1$, $|g^2| = 1$ e $|g^3| = \sqrt{4u^4v^2 + 4v^4u^2 + 1}$.

Agora, seja $V : U \subset \mathbb{R}^3 \to \mathbb{R}^3$ uma função vetorial, onde $V^i := V^i(u, v, w)$, $i = 1, 2, 3$. Para calcularmos o gradiente de V, utilizamos as equações (2.38) e (2.39) para cada componente da matriz ∇V e cada função coordenada de V. Com isso, temos que $(\nabla V)_{ij} = (\partial_j(V_i\sqrt{g_{ii}}) + \Gamma^i_{kj}V_k\sqrt{g_{kk}})|g^{jj}||g_{ii}|$. Segue que:

$$\begin{aligned} \nabla_1 V^1 &= (\partial_u V^1)\sqrt{1 + 4u^2v^4}, \\ \nabla_2 V^1 &= (\partial_v V^1)\sqrt{1 + 4u^2v^4}, \\ \nabla_3 V^1 &= (\partial_w V^1)\sqrt{1 + 4u^2v^4}\sqrt{1 + 4u^4v^2 + 4v^4u^2}, \\ \nabla_1 V^2 &= (\partial_u V^2)\sqrt{1 + 4u^4v^2}, \end{aligned}$$

$$\begin{aligned} \nabla_2 V^2 &= (\partial_v V^2)\sqrt{1 + 4u^4v^v}, \\ \nabla_3 V^2 &= (\partial_w V^2)\sqrt{1 + 4u^4v^2}\sqrt{1 + 4u^4v^2 + 4v^4u^2}, \\ \nabla_1 V^3 &= (\partial_u V^3)\sqrt{4u^4v^2 + 4v^4u^2 + 1} + V^3\frac{16u^3v^2 + 8v^4u}{2\sqrt{4u^4v^2 + 4v^4u^2 + 1}} + 2v^2V^1 + 4uvV^2, \\ \nabla_2 V^3 &= (\partial_v V^3)\sqrt{4u^4v^2 + 4v^4u^2 + 1} + V^3\frac{8u^4v + 16v^3u^2}{2\sqrt{4u^4v^2 + 4v^4u^2 + 1}} + 4uvV^1 + 2u^2V^2, \\ \nabla_3 V^3 &= (\partial_w V^3)(4u^4v^2 + 4v^4u^2 + 1). \end{aligned}$$

Para calcularmos a divergência de V utilizamos (2.39) e o determinante de (g_{ij}). Desta forma obtemos

$$\begin{aligned} \text{div } V &= \sum_{i=1}^{3}\frac{\partial}{\partial x_i}\left(\frac{V_i}{\sqrt{g_{ii}}}\right). \\ &= \frac{\partial V_1}{\partial u}\frac{1}{\sqrt{1 + 4v^4u^2}} - \frac{4uv^4V_1}{(1 + 4v^4u^2)^{3/2}} + \frac{\partial V_2}{\partial v}\frac{1}{\sqrt{1 + 4u^4v^2}} - \frac{4vu^4V_2}{(1 + 4u^4v^2)^{3/2}} + \frac{\partial V_3}{\partial w} \end{aligned}$$

Consideremos

$$\mathcal{T} = \begin{pmatrix} \mathcal{T}^{11}(u, v) & 0 & 0 \\ 0 & \mathcal{T}^{22}(u, v) & 0 \\ 0 & 0) & 1 \end{pmatrix}, \tag{2.45}$$

A fim de determinarmos a divergência do tensor de segunda ordem $\mathcal{T}$, utilizamos primeiramente a equação (2.40) para cada componente de $\mathcal{T}$. Por tanto, da definição (2.31) temos que $(\text{Div } \mathcal{T})_j = \frac{\partial(\mathcal{T}^{ij}\sqrt{g^{ii}}\sqrt{g^{jj}})}{\partial x^i} + \Gamma^i_{ki}\mathcal{T}^{kj}\sqrt{g^{kk}}\sqrt{g^{jj}} + \Gamma^j_{ki}\mathcal{T}^{ik}\sqrt{g^{ii}}\sqrt{g^{kk}}$. Em seguida, devemos utilizar a equação (2.39) para as funções coordenadas do vetor Div $\mathcal{T}$. Para concluir o cálculo da divergencia, lembramos que os únicos símbolos de Christoffel não nulos de S são $\Gamma^3_{11} = 2v^2$, $\Gamma^3_{12} = 4uv$ e $\Gamma^3_{22} = 2u^2$. Logo,

$$\text{div } \mathcal{T} = \begin{pmatrix} \frac{4u^2v^4\frac{\partial}{\partial u}\mathcal{T}^{11}(u,v)}{\sqrt{4u^2v^4+1}} + \frac{\frac{\partial}{\partial u}\mathcal{T}^{11}(u,v)}{\sqrt{4u^2v^4+1}} \\ \frac{4u^4v^2\frac{\partial}{\partial v}\mathcal{T}^{22}(u,v)}{\sqrt{4u^4v^2+1}} + \frac{\frac{\partial}{\partial v}\mathcal{T}^{22}(u,v)}{\sqrt{4u^4v^2+1}} \\ 2v^2\mathcal{T}^{11}(u,v) + 2u^2\mathcal{T}^{22}(u,v) \end{pmatrix}. \tag{2.46}$$

Em relação ao rotacional (2.32), como $det(g_{ij}) = 1$, temos que a permutação ϵ^{ijk} assumirá apenas os valores $0, 1$ ou -1. Portanto, basta utilizarmos a equação (2.39), no cálculo usual para o rotacional de V, para obtermos $\nabla \times V$, isto é,

$$\begin{bmatrix} \frac{\partial}{\partial v}V3(u,v)\sqrt{4u^2v^4+1} \\ -\frac{\partial}{\partial v}V3(u,v)\sqrt{4u^4v^2+1} \\ \frac{4\left(\frac{\partial}{\partial u}V2(u,v)\right)u^4v^2}{\sqrt{4\,u^4v^2+1}} + \frac{8V2(u,v)u^3v^2}{\sqrt{4\,u^4v^2+1}} + \frac{\frac{\partial}{\partial u}V2(u,v)}{\sqrt{4\,u^4v^2+1}} - \frac{4\left(\frac{\partial}{\partial v}V1(u,v)\right)u^2v^4}{\sqrt{4\,u^2v^4+1}} - \frac{8V1(u,v)u^2v^3}{\sqrt{4\,u^2v^4+1}} - \frac{\frac{\partial}{\partial v}V1(u,v)}{\sqrt{4\,u^2v^4+1}} \end{bmatrix}.$$

Os demais operadores, ou seja, Hessiana e Laplaciano de uma função escalar, seguem diretamente das respectivas definições, sem demais atribuições referentes as suas componentes. No caso do Laplaciano de uma função vetorial V, de acordo com a fórmula (2.35), para se obter as componentes do vetor ΔV, basta repetir os procedimentos já feitos para a divergência de V, obtendo assim uma função escalar, em seguida, calcular o gradiente, obtendo assim um vetor, do qual deve-se subtrair o vetor obtido pelo rotacional do rotacional de V.

3 ALGÉBRICA COMPUTACIONAL

Há décadas, que o uso de algoritmos algébricos tem atraído interesse nas áreas de ciência da computação, matemática e aplicações. Os primeiros softwares desenvolvidos para realizar cálculos simbólicos foram, possivelmente, descritos em 1953 [34, 35]. No entanto, muito antes disto, em 1921, cerca de cem anos antes da invenção do computador como conhecemos hoje, o cientista Charles Babbage, prestes a realizar uma apresentação científica, em meio a tabelas matemáticas, ao encontrar uma grande quantidade de erros, exclamou: "Desejo a Deus que esses cálculos tenham sido executados a todo vapor". Algum tempo depois, Babbage começou a idealizar o fim dos erros humanos com a primeira máquina de computação [36].

Em paralelo ao avanço da computação, teorias matemática foram bem desenvolvidas permitindo que objetos algébricos viessem a ser armazenados na memória de um computador, sendo utilizados para cálculos (simbólicos) sem perda de precisão e significância. Desse desenvolvimento, surge a algébrica computacional como "a parte da ciência da computação que projeta, analisa, implementa e aplica algoritmos algébricos" [37]. Consequentemente, o tratamento de expressões matemáticas por meio de algorismos computacionais torna-se amplamente utilizado em diversos campos da ciência [38–41].

Com passar dos anos, muitos softwares destinados a computação simbólica foram desenvolvidos e aprimorados tornando-se poderosas ferramentas para lidar com diversos tipos de problemas da matemática, engenharias e da ciência em geral. Conhecidos como sistemas de álgebra computacional, ou ainda sistema CAS, do inglês computer algebra system, esses softwares apresentam recursos para produzir gráficos e manipular expressões matemáticas em forma analítica (simbólica) e numérica [42–46].

Exemplos de sistemas CAS:

- REDUCE

  Sistema para cálculos algébricos cujo desenvolvimento iniciou-se em 1963 por Tony Hearn durante o pós-doutorado em Física Teórica na Universidade de Stanford. Naquele ano, durante trabalhos com diagramas de Feynman, Hearn imaginou se os cálculos que realizava poderiam ser feitos por computador. Em 1968, Tony Hearn disponibilizou para diversos pesquisadores cópias do software [47]. Em 2008, o Reduce passou a ser disponibilizado sob uma licença BSD modificada.

  O sistema fornece uma linguagem de programação completa, com uma sintaxe semelhante a outras linguagens de programação modernas.

  https://reduce-algebra.sourceforge.io/index.php

- DERIVE

  Foi um sistema CAS, desenvolvido pela Soft Warehouse em Honolulu Warehouse in Honolulu, Hawaii, agora de propriedade da Texas Instruments. O software foi implementado em muLISP. Em 2007 foi descontinuado em favor da calculadora TI-Nspire. A última versão disponibilizada foi Derive 6.1 para MS-Windows.

- MAXIMA

  É um software descendente de Macsyma, o célebre sistema de álgebra computacional desenvolvido por Joel Moses, William Martin e outros no Instituto de Tecnologia de Massachusetts (MIT) no final da década 60. O Macsyma foi revolucionário em sua época e muitas ideias incluídas nele são atualmente usadas em outros programas como o Maple e o Mathematica. Em 1982 o Macsyma teve seu desenvolvimento descontinuado. No entanto, William Schelter manteve uma versão adaptada a linguagem Lisp conhecida por Maxima para diferenciar do Macsyma. Em 1998, Schelter obteve permissão para distribuir Maxima sobre licença GNU GPL.

  http://maxima.sourceforge.net

- MAPLE

  É um sistema algébrico computacional comercial de uso genérico que tem sido utilizado por engenheiros, cientistas e matemáticos para resolver diversos problemas em muitos setores e disciplinas técnicas [48]. Seu desenvolvimento começou no início da década de 80 pelo grupo de computação simbólica na Universidade de Waterloo, no Canadá. Desde 1988, o Maple passou a ser desenvolvido e comercializado pela Maplesoft, uma companhia canadense fornecedora de softwares e serviços para educação, ciência, tecnologia, engenharia e matemática.

  A interface do software é bem amigável, ou seja, permite inserir expressões matemáticas de forma semelhante a notação matemática tradicional. O Maple é baseado em um kernel, escrito em C, que fornece a linguagem Maple. A maioria das funcionalidades do software é fornecida por bibliotecas que em geral são escritas na própria linguagem Maple.

  https://maplesoft.com

- MATLAB

  Comumente relacionado com sistemas de álgebra computacional, o Matlab é uma plataforma de programação comercial, projetada especificamente para engenheiros e cientistas. O software é baseado na linguagem Matlab, baseada em matriz que permite a expressão mais natural da matemática computacional. A linguagem Matlab permite analisar dados, desenvolver algoritmos, criar modelos e aplicativos.

O Matlab foi criado no final da década de 70 por Cleve Moler, no departamento de ciência da computação da Universidade do Novo México. Em 1984, foi reescrito em C por Moler e Steve Bangert, da Universidade de Stanford. No mesmo ano fundaram a MathWorks empresa que continua desenvolvendo o software.

https://www.mathworks.com/products/matlab.html?s_tid=hp_ff_p_matlab

- MATHEMATICA

  É um programa de computador que implementa um sistema de álgebra computacional. Foi desenvolvido por Stephen Wolfram em meados da década de 80, tendo sua primeira versão disponibilizada em 1988, e atualmente continua em desenvolvimento pela empresa Wolfram Research.

  Com pacotes para tratamento numérico, algébrico e gráfico, o Mathematica tem sido amplamente utilizado em diversas áreas, como: engenharia, ciências biológicas e exatas, economia e finanças, ensino da matemática, etc.

  https://www.wolfram.com/mathematica/

Alguns desses sistemas algébricos contém em sua estrutura, um núcleo, que é responsável em executar as operações em baixo nível, e uma biblioteca com a maior parte de suas funcionalidades, composta de funções destinadas a diversas áreas de aplicação. Em parte, o aprimoramento de sistemas CAS ocorre devido ao desenvolvimento de novos pacotes com novas funções podendo serem incluídas em suas bibliotecas.

Os pacotes são conjuntos de algorítimos (códigos) salvos em arquivos específicos de um CAS. Geralmente são desenvolvidos a partir das linguagens de programação dos próprios sistemas. Esses algorítimos, uma vez salvos como pacotes, podem ser acessados pelo Kernel, em momentos posteriores, disponibilizando aos usuários as funções implementadas em seu desenvolvimento sem a necessidade de reescrever os códigos para executa-los.

Muitos pesquisadores têm utilizado desses recursos computacionais para desenvolver pacotes a fim de tratar problemas diversos. Alguns exemplos podem ser encontrados nos sites dos sistemas CAS, ou em trabalhos publicados, como nas referências [49–54]. Quanto a literatura para o desenvolvimento de pacotes sugerimos [55–60], além dos manuais dos CAS.

4 O PACOTE SURF

4.1 Procedimentos

Com a finalidade de desenvolvermos uma ferramenta que possa facilitar a modelagem matemática de sistemas em espaço curvo visando possíveis aplicações reais, escolhemos o sistema de álgebra computacional Maple. Nossa escolha se deu pelo fato da Universidade Federal de Pernambuco (UFPE) ter obtido licenças ilimitadas para uso de alguns CAS, bem como por nossa afinidade com o sistema Maple.

Sobre o desenvolvimento da ferramenta, é útil enfatizar abordagens alternativas e inspiradoras disponíveis na literatura, como o livro de John Oprea [22] e o livro de Alfred Gray [24] que abordam a teoria da geometria diferencial com linguagens de programação dos softwares Maple e Mathematica, respectivamente.

Em relação ao Maple, ressaltamos que o software combina um eficiente mecanismo matemático com mais de 5000 funções destinadas a praticamente todas as áreas da matemática, com uma interface amigável para análise, exploração, visualização e solução de problemas matemáticos. Portanto, utilizamos a linguagem de programação Maple, versão 2016, e finalizamos nossa ferramenta como um pacote Maple, o qual denominamos *Surf*.

Na proxima seção, a fim de explicarmos a metodologia utilizada no desenvolvimento do *Surf*, optamos por exemplificar os procedimentos gerais utilizados para desenvolver um pacote no sistema Maple.

4.1.1 Um pacote Maple

No Maple, pacotes são módulos, isto é, comandos estruturados em blocos de códigos, com os quais se organizam e exportam as funções que constituem um pacote. Essas funções podem receber expressões arbitrárias em Maple, que geralmente são definidas como procedimentos. Por sua vez, um procedimento é uma expressão válida que pode ser atribuída a um nome.

A estrutura básica de um pacote escrito na linguagem Maple deve conter: inicialmente o comando *module()* o que indica o início do módulo; em seguida, a opção *option package* que informa ao Kernel do sistema Maple que o código escrito no módulo refere-se a um pacote; o comando *export* seguido dos nomes atribuídos aos procedimentos (funções), que poderão ser utilizados pelo usuário; e para concluír, o comando *end module:* que finaliza o módulo.

Esboço da estrutura do código de um pacote em Maple:

```
> NomedoPacote:=module()
> option package;
> export funcao1, funcao2;
>
> funcao1:=proc(argumento1,argumento2)
>   Algoritmo1;
>  end proc:
>
> funcao2:=proc(argumento1,argumento2,argumento3)
>   Algoritmo2;
>  end proc:
>
> funcao3:=proc(argumento1)
>   Algoritmo3;
>  end proc:
>
> end module:
```

Notemos que os argumentos argumento1, argumento2 e argumento3 recebem informações necessárias para que a função, na qual o argumento foi utilizado, seja executada. Além disso, as informações são restritas a cada respectivo algorítimo, como uma variável de valor fixado e local.

Sublinhamos que ao utilizarmos um contorno, como no esboço acima, estamos nos referindo a uma planilha de trabalho do software Maple.

Agora, apresentamos um exemplo, que possa ser verificado, de um pacote escrito no Maple. De fato, nesse exemplo, não pretendemos tratar da programação em Maple, mas apenas, uma introdução sobre o processo de criar, estruturar, salvar e executar um pacote no Maple. Lembramos, como mencionado no capítulo anterior, que na literatura específica de cada sistema CAS, existem materiais sobre programação nas suas linguagens específicas. Decerto, o Help do sistema Maple contém excelentes informações sobre o desenvolvimento de pacotes.

Nosso exemplo é um pacote que denominamos por *MeuPacote*, cujo código apresentamos na página 31. Nele, incluímos duas funções que disponibilizamos com o comando *export*.

A primeira função, denominada *funcao1*, contém apenas um argumento, *argumento1*. Nela implementamos três variáveis locais *text1,text2* e *text3*. Observe que ao definirmos uma variável local dentro de um procedimento, só podemos utiliza-la com o valor de

definição, dentro do respectivo procedimento. Em nossa função, enquanto atribuímos uma string na primeira variável *text1*, utilizamos a segunda para converter o *argumento1* em string, e a terceira *text3* para concatenar as outras duas.

A segunda função, *funcao2*, contém um único argumento, *argumento2*, e duas variáveis locais denominadas com os mesmos nomes *text1* e *text2*. No entanto, em *funcao2*, a variável *text1* chama a primeira função, *funcao1*, para o argumento dado em *argumento2* enquanto a variável *text2* converte *text1* para o tipo lista, e no comando *return*, a função nativa do Maple, *nops*, é utilizada para calcular quantos caracteres tem a saída de *funcao1*. Segue o código de nosso exemplo:

```
> MeuPacote:=module()
> option package;
> export funcao1, funcao2;
>
> funcao1:=proc(argumento1)
>  local text1,text2,text3;
>    text1:= "Um pacote no Maple, por: ";
>    text2:= convert(argumento1, string);
>    text3:=cat(text1,text2);
>    return(text3);
>  end proc:
>
> funcao2:=proc(argumento2)
> local text1,text2;
>    text1:=funcao1(argumento2);
>    text2:=convert(text1, list);
>    return(nops(text2));
> end proc:
>
> end module:
```

Para utilizar um pacote no Maple devemos usar o comando *with*, cuja sintaxe é acompanhada do nome do pacote entre parênteses. O Maple, uma vez identificado pelo kernel, que o nome corresponde a um pacote instalado em sua biblioteca, retorna as funções que foram indicadas no código do pacote com o comando *export*. No nosso caso, ainda não instalado o pacote, obtemos o mesmo resultado se utilizarmos o comando *with* na mesma planilha de trabalho que contém o código, e se este já tenha sido executado.

Assim, ao executar *Enter* em algum local do código MeuPacote, devemos obter

com o comando *with(MeuPacote)* os nomes das funções *funcao1* e *funcao2*.

Caso a planilha de trabalho do Maple seja fechada, mesmo tendo salvo o arquivo MeuPacote.mw com o código do pacote, ao reiniciar sistema, o Maple não identificará o novo pacote. Neste caso, se usarmos o comando with(MeuPacote), o sistema retornará uma mensagem de erro:

"Error, invalid input: with expects its 1st argument, pname, to be of type 'module', package, but received MeuPacote."

Para que o Maple identifique nosso pacote, é necessário salvar o código como um arquivo da biblioteca do Maple, cuja extensão é *.mla*, e executar sua instalação em uma pasta identificada como biblioteca (Library) para o sistema Maple.

Para isso, com o arquivo *MeuPacote.mw* aberto, vamos realizar as seguintes etapas:

1. Criar uma pasta que iremos utilizar para salvar os arquivos do pacote, ou utilizamos qualquer outra já existente. No nosso exemplo, criamos uma pasta com nome MinhaBib.

2. Em seguida, iniciamos o pacote:

```
> with(MeuPacote);
                              [funcao1, funcao2]
```

3. Definimos uma variável *libFilename*, com o caminho para a pasta MinhaBib onde iremos criar um repositório .mla:

```
> libFilename:="/Users/marcelopiropo/MinhaBib/MeuPacote.mla":
>
```

   Observe que o caminho */Users/marcelopiropo/MinhaBib/* depende de cada computador em que o Maple foi instalado, e do diretório onde pretende-se criar o pacote.

   Utilizamos o comando *savelib* para criar/salvar o arquivo MeuPacote.mla na pasta MinhaBib:

```
> savelib(MeuPacote, libraryFilename);
```

   Os argumentos do comando *savelib* são: o nome do pacote e a variável com o caminho para o repositório que contém o arquivo *.mla*.

   Após a etapa 4, o pacote já deverá está gravado no arquivo da biblioteca do Maple MeuPacote.mla, na pasta MinhaBib. Nesse momento o pacote está pronto para ser instalado em um sistema Maple de versão igual ou mais recente a que foi utilizada para obter o arquivo .mla.

Agora, tratamos da instalação do pacote no Maple. Das formas possíveis, vamos utilizar a que consideramos mais fácil e intuitiva, de modo a disponibilizar o pacote para outros usuários.

1. Utilizamos o pacote *InstallerBuilder* específico para criar instaladores:

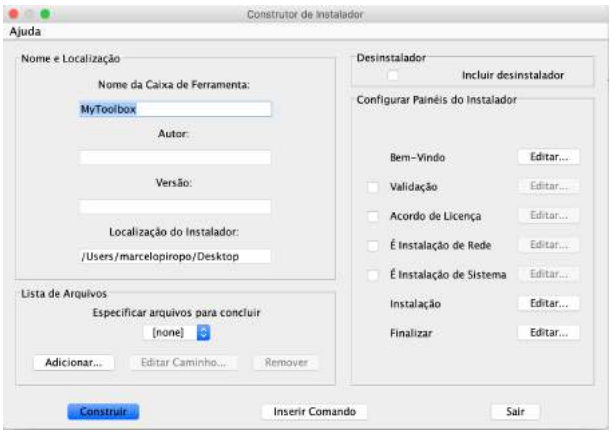
```
> with(InstallerBuilder);
                                [Build, Interactive]
```

2. Após executar o comando *Interactive*:

```
> Interactive();
```

É aberta uma interface (Figura 6) de construção de instaladores:

Figura 6 – Construtor de instalador.



Fonte: print screen da janela de Construtor de instalador do software Maple.

Preenchemos os dados do construtor de instalador:

a) Nome da Caixa de Ferramentas: *MeuPacote*. Esse nome será usado para o instalador (extensão .mla) e para a pasta de instalação;

b) Autor: Marcelo;

c) Versão: Maple xx;

d) Localização do Instalador: */Users/marcelopiropo/Desktop*. Esse é local onde o *InstallerBuilder* irá criar o instalador, não tendo qualquer influência quando o pacote for instalado posteriormente. Note que utilizamos um local diferente de onde está o arquivo de biblioteca do Maple (o pacote), pois nosso instalador (.mla) será criado com mesmo nome;

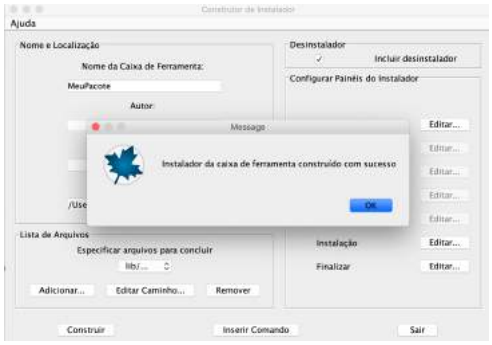
e) Lista de arquivos: Aqui, devemos selecionar o arquivo MeuPacote.mla;

f) Selecionamos a opção *Desinstalador*. Isso faz com que o instalador inclua um arquivo *MeuPacote_Unistall.mla* durante a instalação do pacote;

g) Deixamos para leitor a escolha para a seção Configurar Painéis do Instalador.

3. Após clicar em *Construir*, o sistema abre uma janela de confirmação. Basta clicar em *ok* (Figura 7).

Figura 7 – Finalização do construtor de instalador.

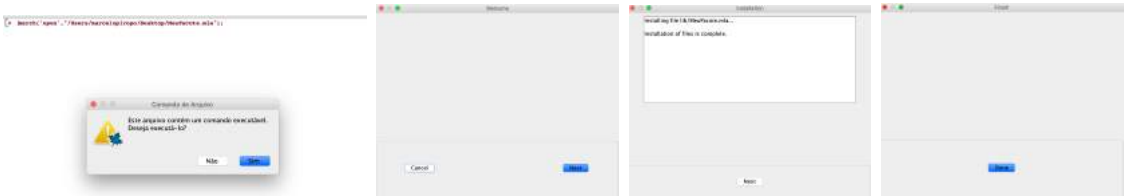


Fonte: print screen da aplicação ao finalizar o construtor de instalador.

Nesta etapa, o instalador (um arquivo MeuPacote.mla) deve ter sido construído com sucesso pelo sistema. Sugerimos modificar o nome desse arquivo, por exemplo, MeuPacoteInstalador.mla.

4. Para instalar o pacote no sistema Maple, basta clicar duas vezes no arquivo MeuPacoteInstalador.mla e seguir o processo de instalação clicando em *Sim*, *next*, *next* e *Done*, nas janelas que serão abertas pelo sistema (Figura 8).

Figura 8 – Etapas da instalação de um pacote.

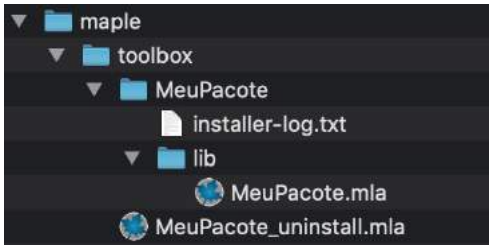


Fonte: print screen das aplicações durante o processo de instalação.

Após a conclusão do processo de instalação, deverá ser criado no diretório raiz do Maple três pastas: *toolbox*,*MeuPacote* e *lib*, contendo os arquivos MeuPacote_unistall.mla, installer-log.txt e MeuPacote.mla (Figura 9).

Ressaltamos que há outras formas de instalação, inclusive utilizando linhas de comando, com a possibilidade de criar diretórios personalizados. No entanto, optamos pela forma que consideramos mais intuitiva, sendo de fácil instalação até para um recente usuário do Maple.

Figura 9 – Diretório de instalação.



Fonte: print screen do diretório de instalação no sistema operacional macOS.

Banco de dados de ajuda

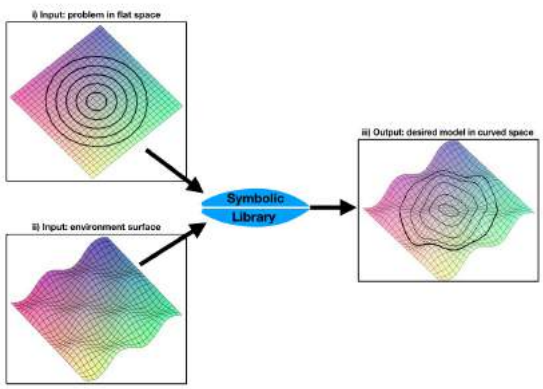
Um banco de dados de ajuda do Maple é um arquivo de extensão *.help* que armazena uma coleção de planilhas (*.mw*). Planilhas *.mw* podem ser convertidas para arquivos de ajuda e, uma vez implementadas em um arquivo *.help*, podemos adiciona-lo ao índice do sistema de ajuda do Maple.

No próprio sistema de ajuda do Maple podemos encontrar informações para criação de páginas de ajuda. Os procedimentos necessários podem ser obtidas, por exemplo, procurando por *Overview of Creating Help Pages in Maple na ajuda disponibilizada no sistema.*

4.2 O pacote Surf

O *Surf* é um pacote Maple desenvolvido para o ensino e pesquisa da geometria diferencial em superfícies curvas, com aplicações em dinâmica dos fluidos, difusão em superfícies, mecânica quântica e biologia. A ideia principal é considerar um sistema modelo no espaço plano, e a partir da parametrização de uma superfície curva, obter o sistema modelo no espaço curvo desejado (Figura 10).

Figura 10 – Diagrama da ideia principal do pacote **Surf**.



Fonte: os autores.

Nesse sentido, referimos ao tratamento simbólico de problemas aplicados do plano para espaços curvos, o que passa necessariamente pelo cálculo de operadores diferenciais, tais como: gradiente, divergência, Laplaciano, rotacional e Hessiana. Assim, o pacote *Surf* foi iniciado com a implementação de funções para o cálculo simbólico desses operadores diferenciais que são fundamentais tanto para avaliação como compreensão teórica de diversos campos da ciência.

Antes de continuarmos com *Surf*, convém fazermos uma observação a respeito do cálculo desses operadores com algumas funções nativas do Maple. Primeiro destacamos que os pacotes Physics[Vectors], VectorCalculus e DifferentialGeometry, nativos da biblioteca do Maple, têm funções para calcular os operadores diferenciais *Gradiente*, *Divergencia*, *Rotacional* e *Laplaciano*.

O primeiro, *Physics[Vectors]*, permite o cálculo dos operadores: gradiente para uma função escalar, o rotacional (Curl) e a divergência (Divergence) para uma função vetorial, e laplaciano (Laplacian) de funções escalares ou vetoriais. Esse operadores são calculados no *Physics[Vectors]* em coordenadas cartesianas, cilíndricas e esféricas denotadas por $\{\phi, r, \rho, \theta, x, y, z\}$.

O pacote *VectorCalculus*, por sua vez, tem funções para o cálculo dos mesmos operadores calculados pelo *Physics[Vectors]*, no entanto, em *VectorCalculus* é possível calcula-los em outras coordenadas curvilíneas ortogonais predefinidas, sendo um total de 13 sistemas em duas dimensões e 24 em três. Além disso, é possível utilizar o comando *AddCoordinates* para incluir um novo sistema de coordenadas. No entanto, quando inserimos a informação de um sistema não ortogonal, o Maple retorna a seguinte mensagem:

Warning, the unit Vectors in the new coordinate system are not orthogonal, only added to global coordinate systems.

Além desses dois pacotes, destacamos o *DifferentialGeometry* por ter recursos notáveis dedicados a diferentes áreas, tais como, variedades, análise tensorial, álgebra de lie e assim por diante. Neste pacote, o Laplaciano pode ser calculado em coordenadas curvilíneas para uma dada métrica de uma superfície. No entanto, sua implementação pode ser bastante abstrata para problemas práticos, como modelar sistemas em espaços curvos com um ponto de vista experimental e fenomenológico [61].

No pacote *Surf*, todas as funções foram implementadas com algoritmos baseados na métrica de uma superfície. Assim, diferente das funções nativas do Maple, os operadores podem ser obtidos em coordenadas curvilíneas não necessariamente ortogonais. Além da possibilidade de serem aplicados à funções escalares, vetoriais e matriciais, dependendo de suas definições. Além disso, outro aspecto importante é que a sintaxe das funções do *Surf* tem como argumento central a parametrização de uma superfície, o que o torna uma ferramenta fácil de usar para cientistas e estudantes de diversas áreas de pesquisa e ensino.

Na próxima seção, apresentamos cada função do *Surf*, ou seja, os operadores diferenciais e as demais funções sobre a geometría de curvas e superfícies, para em seguida, no capitulo 5 mostrarmos nossas aplicações.

4.3 Funções do pacote Surf

Iniciamos essa seção apresentando os argumentos mais comuns utilizados como entrada às funções do *Surf*.

Argumentos de entrada:
metric: Argumento opcional usado para modificar o tipo de entrada para o argumento *param*.
param: Esse argumento aceita, dependendo da função, uma parametrização de uma superfície regular. Exemplo:

$$[r * cos(u), r * sin(u), v].$$

Ou a expressão da métrica quando o argumento **metric** é usado. Exemplo:

$$a * dr^2 + b * drdu + 2c * du^2 + dv^2.$$

vars: Usado para incluir as variáveis independentes da parametrização dada em *param*. Exemplo: $[r, u, v]$.
paramcurv: Substitui o argumento *param*, quando a parametrização de uma curva é esperada. Exemplo: $[u(t), v(t), z(t)]$.
var: Uma variável independente para uma curva dada no argumento *paramcurv*. Exemplo: t;
point: Usado para incluir dois elementos numéricos, coordenadas de um ponto. Exemplo: $[0, \pi]$;
func: Para incluir uma função escalar ou vetorial. Exemplo: $f(u, v)$.
vecfunc: Para incluir uma função vetorial. Exemplo: $< f(u, v), g(u, v), h(u, v) >$.
int: Ponto inicial e final de um intervalo. Exemplo: $[0, 1]$.
output: é usado para modificar a saída de algumas funções.

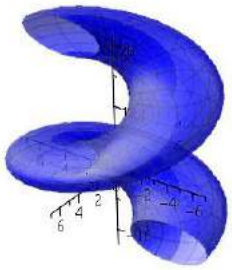
Funções básicas (curvas e superfícies)

Antes de introduzirmos as funções básicas do pacote, esclarecemos que para facilitar o uso dos comandos (funções), adicionamos um "s" ao início dos nomes das funções, diferenciando daquelas que possuem nomes correspondentes no Maple. Mais detalhes e outros exemplos de parametrizações de superfícies conforme utilizadas neste texto, podem ser obtidas em [33]. As figuras apresentadas nos exemplos foram construídas no software Maple com os comandos *plot3d*, *spacecurve* e *intersectplot*, sendo os dois últimos funções do pacote *plot*.

4.3.1 sNormalVector

A função *sNormalVector(param, vars)* calcula o vetor unitário normal (2.10) a uma superfície com parametrização dada em *param*. Quando o argumento opcional *point* é usado, a função *sNormalVector(param, vars, point)* calcula o vetor unitário normal a superfície no ponto indicado em *point*.

Figura 11 – Superfície helicoidal tubular.



Fonte: os autores (obtida com o sofware Maple).

Exemplo

- Vetor normal a uma superfície helicoidal tubular.

Começamos com uma superfície helicoidal tubular $\mathfrak{T}(u, v)$ (Figure 11):

$$tubularsurf := [(a + r * \sin(u)) * \cos(v), (a + r * \sin(u)) * \sin(v), p * v + r * \cos(u)]. \quad (4.1)$$

Para parâmetros genéricos (u, v), o vetor normal é

```
> sNormalVector(tubularsurf, [u, v]);
```

$$\begin{bmatrix} \frac{\left(-r(\cos(u))^2+\sin(u)a+r\right)\cos(v)+\cos(u)\sin(v)p}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}} \\ \frac{\left(-r(\cos(u))^2+\sin(u)a+r\right)\sin(v)-\cos(u)\cos(v)p}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}} \\ \frac{\cos(u)(a+r\sin(u))}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}} \end{bmatrix}.$$

No ponto $(u, v) = (\frac{\pi}{4}, \frac{\pi}{4})$, o vetor normal é

```
> sNormalVector(tubularsurf, [u, v], [Pi/4, Pi/4])
```

$$\begin{bmatrix} \frac{2\,a+r\sqrt{2}+2\,p}{2\sqrt{2\,p^2+2\,r^2+4\,\sqrt{2}ar+4\,a^2}} \\ \frac{2\,a+r\sqrt{2}-2\,p}{2\sqrt{2\,p^2+2\,r^2+4\,\sqrt{2}ar+4\,a^2}} \\ \frac{r\sqrt{2}+2\,a}{2\sqrt{2\,\sqrt{2}ar+2\,a^2+p^2+r^2}} \end{bmatrix}$$

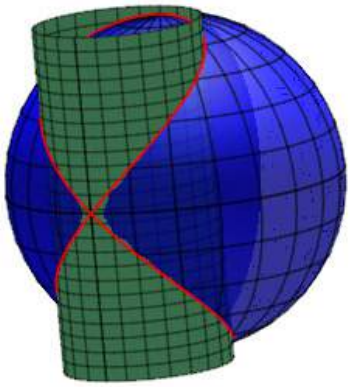
4.3.2 sCurveLength

A função *sCurveLength(paramcurv, var, int)* calcula o comprimento de arco de uma curva em $\mathbb{R}^3$ com parametrização dada pelo argumento *paramcurv*. Omitindo o argumento *int*, temos a função comprimento. Incluindo os argumentos *int*, *param* e *vars*, a função *sCurveLength(paramcurv, var, int, param, vars)* calcula o comprimento de arco da curva sobre a superfície. Eq. (veja Eq. 2.3).

Exemplo

• O comprimento da curva de Viviani
A curva de Viviani é a interseção de uma esfera com um cilindro (Figura 12).

Figura 12 – Curva de Viviani.



Fonte: os autores (obtida com o sofware Maple).

Fórmulas paramétricas:

$$vivianicurve := [r * (1 + cos(t)), r * sin(t), 2 * r * sin((t/2))]. \tag{4.2}$$

$$sphere := [2 * r * cos(v) * cos(u), 2 * r * cos(v) * sin(u), 2 * r * sin(v)]. \tag{4.3}$$

$$cylinder := [r * (1 + cos(u)), r * sin(u), v]. \tag{4.4}$$

Note que o comprimento de arco da curva de Viviani sem considera-la incorporada a uma superfície é

$> \ sCurveLength(vivianicurve, t)$

$$r(\int_{p_1}^{p_2} \sqrt{(\cos(t/2))^2 + 1}\, dt)$$

Para obter o comprimento total, fazemos

$> \ sCurveLength(vivianicurve, t, [-2 * Pi, 2 * Pi])$

$$8\sqrt{2} csgn(r)\, rEllipticE\left(\frac{\sqrt{2}}{2}\right)$$

Considerando a curva sobre um cilindro, a função comprimento de arco passa a ser

$$> \; sCurveLength(vivianicurve, t, [-2*\pi, 2*\pi], cylinder, [u, v], symbolic)$$

$$r \int_{-2\,\pi}^{2\,\pi} \sqrt{-r^2 \left(\cos\left(t\right)\right)^2 + \left(\cos\left(t\right)\right)^2 + r^2}\, dt$$

A solução é

$$> \; sCurveLength(vivianicurve, t, [-2*\pi, 2*\pi], cylinder, [u, v])$$

$$8\, csgn\left(\frac{1}{r}\right) r^2 EllipticE\left(\frac{\sqrt{r^2-1}}{r}\right)$$

Repetindo este procedimento, considerando a curva na esfera, temos que

$$> \; sCurveLength(vivianicurve, t, [-2*Pi, 2*Pi], sphere, [u, v], symbolic)$$

$$2r^2 \int_{-2\,\pi}^{2\,\pi} \sqrt{-\left(\cos\left(t\right)\right)^2 \left(\cos\left(v\right)\right)^2 + \left(\cos\left(v\right)\right)^2}\, dt$$

Calculando

$$> \; sCurveLength(vivianicurve, t, [-2*Pi, 2*Pi], sphere, [u, v])$$

$$16\, r^2 EllipticE\left(\frac{i\sin(v)}{\cos(v)}\right) csgn\left(\frac{1}{\cos(v)}\right) \cos\left(v\right)$$

4.3.3 sCurveCurvature

A função *sCurveCurvature(paramcurv,var,param,vars)* calcula a curvatura (2.5) de uma curva diferenciável definida no plano, no espaço tri-dimensional, ou sobre uma superfície.

Observação: se incluirmos os argumentos *param* e *vars*, então o argumento *paramcurv* deve ser: a parametrização de uma curva, se tem três elementos, ou funções coordenadas da curva que serão aplicadas a parametrizaçao da superfície, se tem dois elementos.

Exemplo

• Para uma curva explícita, fazemos

$$> \; sCurveCurvature(a*cos(t) + a*t*sin(t), t)$$

$$\frac{|a(-\cos(t)+t\sin(t))|}{\left(1+a^2t^2(\cos(t))^2\right)^{3/2}}$$

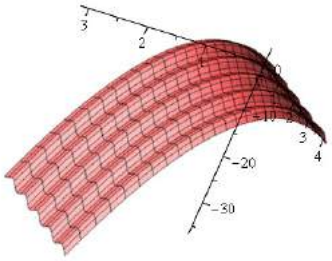
• Para uma curva plana

```
> sCurveCurvature([a * cos(t), b * sin(t)], t)

                              ab
                  ((-a^2+b^2)(cos(t))^2+a^2)^(3/2)
```

Exemplo

Figura 13 – Translação de um sinusoide ao longe de uma parábola.



Fonte: os autores (obtida com o sofware Maple).

• Considere uma superfície gerada pela translação de um sinusoide ao longe de uma parábola (Figura 13):

$$sinusoidsurface := [u, v, -a\,(u-b)^2 - c\cos\left(\frac{n\pi\,v}{d}\right) + ab^2 + c]. \tag{4.5}$$

Agora, considere uma curva sobre a superfície acima, assumindo u, a, b, c, d, n fixos, com $v = t$.

Sua curvatura em $\mathbb{R}^3$ é

```
> sCurveCurvature([u, t, -a (u - b)^2 - c cos(n*pi*t/d) + ab^2 + c], t)

                          cn^2 pi^2 d cos(n pi t/d)
          (-c^2 n^2 pi^2 (cos(n pi t/d))^2 + c^2 n^2 pi^2 + d^2)^(3/2)
```

A curvatura da curva sobre a superfície é

```
> sCurveCurvature(sinusoidcurve, t, sinusoidsurface, [u, v])

                          2 cn^2 pi^2 cos(n pi v/d) a(-u+b)
          d sqrt(-c^2 n^2 pi^2 (cos(n pi v/d))^2 + (1+4 a^2 (-u+b)^2) d^2 + c^2 n^2 pi^2)
```

4.3.4 sTorsion

A função *sTorsion(paramcurv,var)* calcula a torção (2.17) de uma curva com parametrização dada por *paramcurv*. Quando o argumento opcional *geodesic* é usado, a função *sTorsion(paramcurv,var,geodesic)* calcula a torção geodésica de uma curva (2.18).

Exemplo

• A torção da curva de Viviani é

```
> sTorsion(vivianicurve, t)
```

$$\frac{3\cos\left(\frac{t}{2}\right)}{\left(3\,\cos(\frac{t}{2})^2+5\right)r}$$

A torção geodésica

```
> sTorsion(vivianicurve, v, geodesic)
```

$$-\frac{3r\cos\left(\frac{t}{2}\right)}{4\,\cos(\frac{t}{2})^2+4}$$

4.3.5 sNormalCurvature

A função *sNormalCurvature(param,vars,paramcurv,var)* calcula a curvatura normal (2.19) de uma curva dada em *paramcurv* sobre a superfície especificada em *param*.

Exemplo

• A curvatura normal de uma curva dada pela parametrização da superfície *sinusoid* (4.5), considerando $v = t$ e u constantes.

$$\gamma(t) := [u, t, -a\,(u-b)^2 - c\cos\left(\frac{n\pi\,t}{d}\right) + ab^2 + c]. \tag{4.6}$$

```
> sNormalCurvature(sinusoidsurface, [u, v], γ(t), t)
```

$$\frac{\left(cn^2\pi^2 d\cos\left(\frac{n\pi\,t}{d}\right)\right)\left(-c^2n^2\pi^2\left(\cos\left(\frac{n\pi\,t}{d}\right)\right)^2+c^2n^2\pi^2+d^2\right)^{-1}}{\sqrt{-c^2n^2\pi^2\left(\cos\left(\frac{n\pi\,v}{d}\right)\right)^2+\left(1+4\,a^2(-u+b)^2\right)d^2+c^2n^2\pi^2}}$$

4.3.6 sGeodesicCurvature

A função *sGeodesicCurvature(param,vars,paramcurv,var)* calcula a curvatura geodésica (2.19) de uma curva sobre uma superfície .

Exemplo

• A curvatura geodésica de uma curva dada pela equação (4.6)

$$> \quad sGeodesicCurvature(sinusoidsurface, [u, v], \gamma(t), t)$$

$$\frac{2\,cn^2\pi^2\cos\left(\frac{n\pi\,t}{d}\right)a(-u+b)}{d\sqrt{-c^2n^2\pi^2\left(\cos\left(\frac{n\pi\,v}{d}\right)\right)^2+\left(1+4\,a^2(-u+b)^2\right)d^2+c^2n^2\pi^2}}$$

4.3.7 sSurfaceCurvature

A função *sSurfaceCurvature(param,vars)* calcula por padrão a curvatura de Gauss (2.23) da superfície dada no argumento *param*. Quando *output = mean*, a função *sSurfaceCurvature(param,vars,opt=mean)* calcula a curvatura média (2.22). Se *output = secondgauss*, então a função GaussCurvature(param,vars,output = secondgauss) calcula a segunda curvatura de Gauss (2.26). Quando *output = mean*, a função *sSurfaceCurvature* calcula as curvaturas principais.

Exemplo

• As curvaturas de um sinusoid ao longo de uma parábola:

Lembrando que o *sinusoid* ao longo de uma parábola é dado por:

$$sinusoidsurface := [u, v, -a\,(u-b)^2 - c\cos\left(\frac{n\pi\,v}{d}\right) + ab^2 + c] :$$

A curvatura de Gauss:

$$> \quad sSurfaceCurvature(sinusoidsurface, [u, v])$$

$$-\frac{acn^2\pi^2\cos\left(\frac{n\pi\,v}{d}\right)d^2}{8\left(\frac{-c^2n^2\pi^2\left(\cos\left(\frac{n\pi\,v}{d}\right)\right)^2}{4}+d^2\left(\frac{1}{4}+a^2(-u+b)^2\right)+\frac{c^2n^2\pi^2}{4}\right)^2}$$

A curvatura média:

$$> \quad sSurfaceCurvature(sinusoidsurface, [u, v], output = mean)$$

$$\frac{2d\left(\frac{\pi^2\left(\cos\left(\frac{n\pi\,v}{d}\right)\right)^2ac^2n^2}{2}+\pi^2n^2\left(\frac{1}{4}+a^2(-u+b)^2\right)c\cos\left(\frac{n\pi\,v}{d}\right)-\frac{a\left(c^2n^2\pi^2+d^2\right)}{2}\right)}{\left(-c^2n^2\pi^2\left(\cos\left(\frac{n\pi\,v}{d}\right)\right)^2+\left(1+4\,a^2(-u+b)^2\right)d^2+c^2n^2\pi^2\right)^{3/2}}$$

A segunda curvatura de Gauss (por simplicidade, calculamos esta para a esfera):

$$sphere := [rcos(v)sin(u), rsin(v)sin(u), rcos(u)].$$

```
> sSurfaceCurvature(sphere, [u, v], output = secondgauss)
```

$$-\frac{1}{r(\sin(u))^4}$$

Exemplo:

• Para uma superfície de revolução, as curvaturas principais são obtidas:

```
> sCurvatureSurface([F(u) cos(v), F(u) sin(v), H(u)], [u, v], output = principal)
```

$$\left\{ \frac{\frac{d}{du}H(u)}{F(u)\sqrt{\left(\frac{d}{du}F(u)\right)^2+\left(\frac{d}{du}H(u)\right)^2}}, \frac{\left(\frac{d}{du}F(u)\right)\frac{d^2}{du^2}H(u)-\left(\frac{d}{du}H(u)\right)\frac{d^2}{du^2}F(u)}{\left(\left(\frac{d}{du}F(u)\right)^2+\left(\frac{d}{du}H(u)\right)^2\right)^{3/2}} \right\}$$

Primeira e segunda forma fundamental

4.3.8 sFirstForm

A função *sFirstForm(param,vars)* calcula a primeira forma fundamental (2.1) de uma superfície com parametrização dada por *param*.

Por padrão, a saída da função *sFirstForm* é a forma em sí. Uma observação: quantidades relacionadas que podem ser úteis para modelar sistemas no espaço curvo podem ser obtidas especificando a saída desejada. Se *output = matrix*, ou *inverse*, ou *det*, ou *indet*, ou *coeff*, obtem-se a matriz da primeira forma, sua inversa, seu determinante, o determinante da matriz inversa, ou uma lista com os coeficientes da primeira forma, respectivamente.

Exemplo

• A primeira forma fundamental da superfície tubular.

A expressão da métrica

```
> sFirstForm(tubularsurf, [u, v])
```

$$r^2du^2 - 2\,r\sin(u)\,pdu\,dv + \left(-(\cos(u))^2\,r^2 + 2\,\sin(u)\,ar + a^2 + p^2 + r^2\right)dv^2$$

A matriz da forma

```
> sFirstForm(tubularsurf, [u, v], output=matrix)
```

$$\begin{bmatrix} r^2 & -r\sin(u)\,p \\ -r\sin(u)\,p & -(\cos(u))^2\,r^2 + 2\sin(u)ar + a^2 + p^2 + r^2 \end{bmatrix}$$

O determinante

```
> sFirstForm(tubularsurf, [u, v], output=det)
```

$$-\cos(u)^2 r^4 - r^2 \sin(u)^2 p^2 + 2\sin(u) a r^3 + a^2 r^2 + p^2 r^2 + r^4$$

A matriz inversa

```
> sFirstForm(tubularsurf, [u, v], output=inverse)
```

$$\begin{bmatrix} \frac{(\cos(u))^2 r^2 - 2\,\sin(u) a r - a^2 - p^2 - r^2}{r^2\left((\cos(u))^2 r^2 + (\sin(u))^2 p^2 - 2\sin(u) a r - a^2 - p^2 - r^2\right)} & \frac{\sin(u) p}{r\left((\cos(u))^2 r^2 + (\sin(u))^2 p^2 - 2\,\sin(u) a r - a^2 - p^2 - r^2\right)} \\ -\frac{\sin(u) p}{r(\cos(u)^2 r^2 + \sin(u)^2 p^2 - 2\sin(u) a r - a^2 - p^2 - r^2)} & -\left((\cos(u))^2 r^2 + (\sin(u))^2 p^2 - 2\,\sin(u) a r - a^2 - p^2 - r^2\right)^{-1} \end{bmatrix}$$

O determinante da matriz inversa

```
> sFirstForm(tubularsurf, [u, v], output=indet)
```

$$-\frac{1}{r^2\left((\cos(u))^2 r^2 + (\sin(u))^2 p^2 - 2\,\sin(u) a r - a^2 - p^2 - r^2\right)}$$

Uma lista com os coeficientes da métrica

```
> sFirstForm(tubularsurf, [u, v], output=coeff)
```

$$[r^2, -r\sin(u)\, p, -(\cos(u))^2 r^2 + 2\,\sin(u)\, a r + a^2 + p^2 + r^2]$$

4.3.9 sChristoffelSymbols

Quando o argumento *vars* é uma lista com duas variáveis, a função *sChristoffelSymbols(param,vars)* calcula os símbolos de Christoffel (2.21) de uma superfície mergulhada (de variáveis *vars*) in $\mathbb{R}^3$. Se a lista *vars* tem três variáveis, a função *sChristoffelSymbols* calcula os símbolos de Christoffel no espaço em coordenadas dadas por *param*.

Por padrão, a função retorna matrizes com os símbolos de Christoffel. Nesse caso, os símbolos são ordenados da seguinte forma

$$\Gamma^k_{ij} = \begin{pmatrix} \Gamma^k_{11} & \Gamma^k_{12} \\ \Gamma^k_{21} & \Gamma^k_{22} \end{pmatrix}, \tag{4.7}$$

se calculados em uma superfície. Ou se consideradas três coordenadas curvilíneas,

$$\Gamma^k_{ij} = \begin{pmatrix} \Gamma^k_{11} & \Gamma^k_{12} & \Gamma^k_{13} \\ \Gamma^k_{21} & \Gamma^k_{22} & \Gamma^k_{23} \\ \Gamma^k_{31} & \Gamma^k_{32} & \Gamma^k_{33} \end{pmatrix}. \tag{4.8}$$

Quando o argumento opcional *output=list* é usado, a função *sChristoffelSymbols* retorna uma lista com os símbolos não nulos e sem repetir os simétricos:

$$[\Gamma^1_{11}, \Gamma^1_{12}, \Gamma^1_{21}, \Gamma^1_{22}, \Gamma^2_{11}, \Gamma^2_{12}, \Gamma^2_{21}, \Gamma^2_{22}].$$

Quando a entrada para o argumento *param* é uma métrica, deve-se incluir o argumento *metric*, isto é, *sChristoffelSymbols(param,vars,metric)*. Neste caso, *param* deve ser a forma quadrada de uma métrica.

Por padrão, *sChristoffelSymbols* calcula os símbolos de segundo tipo. Para calcular os símbolos de primeiro tipo, devemos utilizar o argumento *firstkind*.

Exemplo

• Os símbolos de Christoffel sobre uma superfície tubular, considerando $a = 0, r = 1, p = 1$, dada por:

$$modtubular := [(sin(u)) * cos(v), (sin(u)) * sin(v), v + cos(u)]. \tag{4.9}$$

> *sChristoffelSymbols*(*modtubular*, $[u, v]$)

$$\Gamma^1_{ij} = \begin{bmatrix} -\sin(u)\cos(u) & (\sin(u))^2\cos(u) \\ (\sin(u))^2\cos(u) & \left((\cos(u))^2 - 2\right)\sin(u)\cos(u) \end{bmatrix}$$

$$\Gamma^2_{ij} = \begin{bmatrix} -\cos(u) & \sin(u)\cos(u) \\ \sin(u)\cos(u) & -(\sin(u))^2\cos(u) \end{bmatrix}$$

Exemplo

• A partir da expressão da métrica de uma superfície:

> *sChristoffelSymbols*($dr^2 + r^2du^2 + r^2\sin(u)^2dv^2, [r, u, v]$, output=list)

$$\left[\Gamma^1_{22} = -r, \Gamma^1_{33} = -r\sin(u)^2, \Gamma^2_{12} = \tfrac{1}{r}, \Gamma^2_{33} = -\sin(u)\cos(u), \Gamma^3_{13} = \tfrac{1}{r}, \Gamma^3_{23} = \tfrac{\cos u}{\sin u}\right]$$

Exemplo

• Símbolos de Christoffel de primeiro tipo dados por uma métrica:

> *sChristoffelSymbols* $\left(r^2du^2 + dv^2, [r, u, v], metric, \text{output=list}, firstkind\right)$

$$\left[\Gamma_{111} = \tfrac{u}{2} - \tfrac{r}{2}, \Gamma_{112} = \tfrac{r}{2}, \Gamma_{211} = \tfrac{u}{2} - \tfrac{r}{2}, \Gamma_{212} = \tfrac{r}{2}, \Gamma_{311} = \tfrac{u}{2} - \tfrac{r}{2}, \Gamma_{312} = \tfrac{r}{2}\right]$$

4.3.10 sGeodesicEquations

A função *sGeodesicEquations(param, var, paramcurve, var)* calcula as equações da geodésica de uma curva sobre uma superfície.

Exemplo

As equações diferenciais das geodésicas sobre uma superfície definida por $\sigma(u,v) = [u, v, u^2 + v^2]$.

```
> sGeodesicEquations([u, v, u^2 + v^2], [u, v], [u(t), v(t)], t)
```

$$\left[\begin{array}{l} \frac{d^2}{dt^2}u\left(t\right) + \frac{4u\left(\frac{d}{dt}u(t)\right)^2}{4\,u^2+4\,v^2+1} + \frac{4u\left(\frac{d}{dt}v(t)\right)^2}{4\,u^2+4\,v^2+1} = 0 \\ \frac{d^2}{dt^2}v\left(t\right) + \frac{4v\left(\frac{d}{dt}dtu(t)\right)^2}{4\,u^2+4\,v^2+1} + \frac{4v\left(\frac{d}{dt}v(t)\right)^2}{4\,u^2+4\,v^2+1} = 0 \end{array}\right]$$

4.3.11 sSecondForm

A função *sSecondForm(param,vars)* calcula a segunda forma fundamental (2.11) de uma superfície com parametrização dada em *param*.

A função *sSecondForm* retorna por padrão a forma, e pode ser modificada com procedimento análogo a função *sFirstForm*. Assim, usando as opções *matrix*, *inverse*, *det*, *indet*, *coeff*, tem-se: a matriz da forma; a inversa, os determinantes, ou uma lista com coeficientes, respectivamente.

Exemplo

• A segunda forma de uma superfície tubular é obtida por :

```
> sSecondForm(tubularsurf, [u, v])
```

$$-\frac{r(a+r\sin(u))du^2}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}} - \frac{(2\cos(u))^2rpdu\,dv}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}}$$

$$-\frac{\sin(u)\left(-(\cos(u))^2r^2+2\,\sin(u)ar+a^2+r^2\right)dv^2}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}}$$

• E sua representação matricial:

```
> sSecondForm(tubularsurf, [u, v], output = matrix)
```

$$\left[\begin{array}{ll} -\frac{r(a+r\sin(u))}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}} & -\frac{(\cos(u))^2rp}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}} \\ -\frac{(\cos(u))^2rp}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}} & -\frac{\sin(u)\left(-(\cos(u))^2r^2+2\,\sin(u)ar+a^2+r^2\right)}{\sqrt{(p^2-r^2)(\cos(u))^2+2\,\sin(u)ar+a^2+r^2}} \end{array}\right]$$

Operadores diferenciais sobre superfícies curvas

Os operadores diferenciais podem ser calculados sobre qualquer superfície curva (com métrica não singular), seja em coordenadas curvilíneas ortogonais ou não-ortogonais.

Adicionando o argumento opcional *symbolic*, a função não calcula as derivadas explicitamente. Com isso, a expressão do operador pode se assemelhar à fórmula padrão da função. Em alguns casos com o argumento *symbolic*, o operador não atuará em uma função constante, não retornando o resultado nulo desejado para tal.

4.3.12 sGradient

A função *sGradient(param,func,vars)* calcula o vetor gradiente (2.28) de uma função escalar dada em $func$, sobre uma superfície dada pela parametrização, *param*. O argumento *var* aceita duas ou três variáveis. Se forem especificadas três variáveis, o operador calcula o gradiente em três dimensões.

Quando *func* é uma função vetorial, a função *sGradient* retorna uma matriz representação do tensor de segunda ordem dada por Eq. (2.29).

No argumento *param* podemos inserir termos da forma quadrática de uma métrica (2.2) se $func$ é uma função escalar e o argumento opcional *metric* é incluído.

Exemplo

• O gradiente de uma função escalar sobre uma uma superfície helicoidal tubular:

$> \; sGradient\,(tubularsurf, f(u,v), [u,v])$

$$\begin{bmatrix} \frac{\left(-(\cos(u))^2r^2+2\sin(u)ar+a^2+p^2+r^2\right)\frac{\partial}{\partial u}f(u,v)}{r\left((\cos(u))^2p^2-(\cos(u))^2r^2+2\sin(u)ar+a^2+r^2\right)}+\frac{\sin(u)p\frac{\partial}{\partial v}f(u,v)}{(p^2-r^2)((\cos(u))^2+2\sin(u)ar+a^2+r^2)} \\ \frac{\sqrt{-(\cos(u))^2r^2-2\sin(u)ar+a^2+p^2+r^2}p\sin(u)\frac{\partial}{\partial u}f(u,v)}{r\left((p^2-r^2)(\cos(u))^2+2\sin(u)ar+a^2+r^2\right)}+\frac{\sqrt{-(\cos(u))^2r^2+2\sin(u)ar+a^2+p^2+r^2}\frac{\partial}{\partial v}f(u,v)}{(p^2-r^2)(\cos(u))^2+2\sin(u)ar+a^2+r^2} \end{bmatrix}$$

Exemplo

• O operador gradiente de uma função escalar $f(u,v)$ sobre uma superfície de métrica

$$mtrc = \text{a*du**2-b*dudv+c*dv**2,}$$

é obtido por

$> \; sGradient\,(mtrc, f(u,v), [u,v], metric)$

$$\begin{bmatrix} \frac{4\sqrt{a}c\frac{\partial}{\partial u}f(u,v)}{4\,ac-b^2}+\frac{2\sqrt{a}b\frac{\partial}{\partial v}f(u,v)}{4\,ac-b^2} \\ \frac{2\sqrt{c}b\frac{\partial}{\partial u}f(u,v)}{4\,ac-b^2}+\frac{4\sqrt{c}a\frac{\partial}{\partial v}f(u,v)}{4\,ac-b^2} \end{bmatrix}$$

Exemplo

• O gradiente de uma função vetorial sobre uma esfera:

$$> \ sGradient(sphere, < r\sin(\frac{\pi}{2}\cos(v)), r\tfrac{\pi}{2}\sin(v), rcos(v) >, [r,u,v]) :$$

$$\begin{bmatrix} \sin(\frac{r\cos(v)}{2}) & -\frac{\pi\sin(v)}{2} & \frac{-\frac{r\pi\sin(v)\cos(\frac{\pi\cos(v)}{2})}{2}-r\cos(v)\sin(u)}{r\sin(u)} \\ \frac{\pi\sin(v)}{2} & \sin(\frac{\pi\cos(v)}{2}) & \frac{\frac{\pi\cos(v)}{2}-\cos(v)\sin(u)}{\sin(u)} \\ \cos(v) & 0 & \frac{\sin(v)}{\sin(u)}+\sin(\frac{\pi\cos(v)}{2})+\frac{\pi\sin(v)\cos(u)}{2\sin(u)} \end{bmatrix}$$

4.3.13 sDivergence

A função *sDivergence(param,func,vars)* calcula a divergência (2.30) de uma função vetorial dada no argumento *func*. Neste caso, a *sDivergence* é uma função com valores reais. Quando *func* é uma matriz com componentes de um tensor de segunda ordem, *sDivergence(param,func,vars)* é uma aplicação com valor vetorial em $\mathbb{R}^3$ como dado por (2.31) .

Se o argumento opcional *metric* é incluído e *func* é um vetor, no argumento *param* podemos inserir uma métrica (2.2).

Exemplo

• A divergência de uma função vetorial definida sobre uma tira helicoidal cilíndrica:

$$> \ sDivergence(cylindricalstrip, < V1(u,v), V2(u,v), V3(u,v) >, [r,u,v])$$

$$\frac{V1(u,v)}{r}+\frac{\frac{\partial}{\partial u}V2(u,v)}{\sqrt{c^2+r^2}}+\frac{\partial V3(u,v)}{\partial v}$$

Exemplo

• A divergência sobre uma superfície com métrica *mtrc*, dada por

dr**2+u*du**2+dudv+dv**2.

$$> \ sDivergence(mtrc, < V1(u,v), V2(u,v), V3(u,v) >, [r,u,v], metric)$$

$$\frac{2\left(\frac{\left(\frac{\partial}{\partial u}V2(u,v)\right)\sqrt{4u-1}}{2\sqrt{u}}-\frac{V2(u,v)\sqrt{4u-1}}{4u^{3/2}}+\frac{V2(u,v)}{\sqrt{u}\sqrt{4u-1}}\right)}{\sqrt{4u-1}}+\frac{\partial}{\partial v}V3(u,v)$$

Exemplo

• Divergência de uma matriz sobre uma tira helicoidal cilíndrica:

$$Tij := \begin{bmatrix} T_{11}(u,v) & T_{12}(u,v) & T_{13}(u,v) \\ T_{12}(u,v) & T_{22}(u,v) & T_{23}(u,v) \\ T_{13}(u,v) & T_{23}(u,v) & T_{33}(u,v) \end{bmatrix}$$

$$> \ sDivergence(cylindricalstrip, Tij, [r,u,v])$$

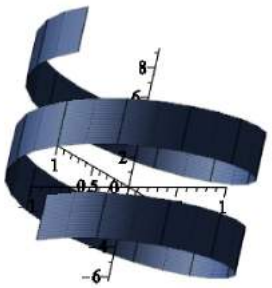
$$\begin{bmatrix} \frac{\frac{\partial}{\partial u}T_{21}(u,v)}{r} + \frac{T_{11}(u,v)}{r} - \frac{T_{22}(u,v)}{r} + \frac{\left(\frac{\partial}{\partial v}T_{31}(u,v)\right)\sqrt{c^2+r^2}}{r} \\ \frac{\left(\frac{\partial}{\partial v}T_{32}(u,v)\right)c^2}{r^2} + \frac{\partial}{\partial v}T_{32}(u,v) + \frac{\left(\frac{\partial}{\partial u}T_{22}(u,v)\right)c^2}{\sqrt{c^2+r^2}r^2} + \frac{\frac{\partial}{\partial u}T_{22}(u,v)}{\sqrt{c^2+r^2}} + \frac{T_{12}(u,v)c^2}{\sqrt{c^2+r^2}r^2} + \frac{T_{12}(u,v)}{\sqrt{c^2+r^2}} + \frac{T_{21}(u,v)c^2}{\sqrt{c^2+r^2}r^2} + \frac{T_{21}(u,v)}{\sqrt{c^2+r^2}} \\ \frac{\left(\frac{\partial}{\partial v}T_{33}(u,v)\right)c^2}{r^2} + \frac{\partial}{\partial v}T_{33}(u,v) + \frac{T_{13}(u,v)}{\sqrt{c^2+r^2}} - \frac{cT_{12}(u,v)}{r^2} + \frac{\left(\frac{\partial}{\partial u}T_{23}(u,v)\right)c^2}{\sqrt{c^2+r^2}r^2} + \frac{\frac{\partial}{\partial u}T_{23}(u,v)}{\sqrt{c^2+r^2}} - \frac{cT_{21}(u,v)}{r^2} \end{bmatrix}$$

4.3.14 sCurl

A função *sCurl(param,funcvec,vars)* calcula o rotacional (curvilineo) (2.32) de uma campo vetorial dado por *funcvec*.

No argumento *param* podemos inserir uma métrica (2.2) se o argumento opcional *metric* é incluído.

Figura 14 – Tira helicoidal cilíndrica.



Fonte: os autores (obtida com o sofware Maple).

Exemplo

• Rotacional de uma função vetorial definida sobre uma superfície (tira) helicoidal cilíndrica (Figura 14) dada pela parametrização:

$$cylindricalstrip := [r\cos(u), r\sin(u), cu + v]. \tag{4.10}$$

> $sCurl(cylindricalstrip, < V1(u,v), V2(u,v), V3(u,v) >, [r,u,v])$

$$\begin{bmatrix} \frac{\frac{\partial}{\partial u}V3(u,v)}{r} - \frac{\left(\frac{\partial}{\partial v}V2(u,v)\right)\sqrt{c^2+r^2}}{r} \\ \frac{\left(\frac{\partial}{\partial v}V1(u,v)\right)\sqrt{c^2+r^2}}{r} \\ -\frac{\frac{\partial}{\partial u}V1(u,v)}{r} + \frac{V2(u,v)}{\sqrt{c^2+r^2}} \end{bmatrix}$$

Exemplo

• O rotacional de uma função vetorial sobre uma superfície de métrica

$$mtrc := a(u,v) * drdr + b(u,v) * du^2 + c(u,v) * dv^2.$$

> $sCurl(mtrc, < f1(u,v), f2(u,v), f3(u,v) >, [r,u,v], metric)$

$$\begin{bmatrix} \frac{V3(u,v)\frac{\partial}{\partial u}c(u,v)}{2c(u,v)\sqrt{b(u,v)}} + \frac{\frac{\partial}{\partial u}V3(u,v)}{\sqrt{b(u,v)}} - \frac{V2(u,v)\frac{\partial}{\partial v}b(u,v)}{2\sqrt{c(u,v)}b(u,v)} - \frac{\frac{\partial}{\partial v}V2(u,v)}{\sqrt{c(u,v)}} \\ \frac{\frac{\partial}{\partial v}V1(u,v)}{\sqrt{c(u,v)}} + \frac{V1(u,v)\frac{\partial}{\partial v}a(u,v)}{2a(u,v)\sqrt{c(u,v)}} \\ -\frac{\frac{\partial}{\partial u}V1(u,v)}{\sqrt{b(u,v)}} - \frac{V1(u,v)\frac{\partial}{\partial u}a(u,v)}{2a(u,v)\sqrt{b(u,v)}} \end{bmatrix}$$

4.3.15 sLaplacian

A função *sLaplacian(param,func,var)* calcula o operador Laplace-Beltrami de uma função escalar ou vetorial (2.34).

Se o argumento opciona *metric* é incluído e *func* é uma função escalar, no argumento *param* podemos inserir uma métrica (2.2).

Exemplo

• O Laplaciano de uma função escalar sobre a superfície helicoidal tubular:

> $sLaplacian(tubularsurf, f(u,v), [u,v], \text{symbolic})$

$$\frac{\frac{\partial}{\partial u}\left(\frac{\left(-r^2(\cos(u))^2+2\sin(u)ar+a^2+p^2+r^2\right)\frac{\partial}{\partial u}f(u,v)+\sin(u)p\left(\frac{\partial}{\partial v}\frac{\partial}{\partial u}f(u,v)\right)r}{r\sqrt{2\sin(u)ar+(p^2-r^2)(\cos(u))^2+a^2+r^2}}\right)+\frac{\partial}{\partial v}\left(\frac{\sin(u)p\frac{\partial}{\partial u}f(u,v)+\left(\frac{\partial}{\partial v}f(u,v)\right)r}{\sqrt{2\sin(u)ar+(p^2-r^2)(\cos(u))^2+a^2+r^2}}\right)}{r\sqrt{2\sin(u)ar+(p^2-r^2)(\cos(u))^2+a^2+r^2}}$$

Exemplo

• Para o hiperbolóide de revolução de uma folha, com métrica

$$mtrc := a^2 * (v^2 + 1) * du^2 - 2 * a^2 * du * dv + (a^2) * dv^2,$$

o Laplaciano é obtido por:

```
> sLaplacian(mtrc, f(u,v), [u,v], metric)
```

$$\frac{(v^3+v)\frac{\partial^2}{\partial v^2}f(u,v)+(\frac{\partial}{\partial v}f(u,v))v^2+(\frac{\partial^2}{\partial u^2}f(u,v))v+(\frac{\partial^2}{\partial v\partial u}f(u,v))v-\frac{\partial}{\partial u}f(u,v)-\frac{\partial}{\partial v}f(u,v)}{a^2v^3}$$

Exemplo

• O Laplaciano de uma função vetorial sobre um cilindro:

```
> sLaplacian([r*cos(u), r*sin(u), v], <V1(u,v), V2(u,v), V3(u,v)>, [r,u,v])
```

$$\begin{bmatrix} -\frac{V1(u,v)}{r^2} - \frac{2\frac{\partial}{\partial u}V2(u,v)}{r^2} + \frac{\partial^2}{\partial v^2}V1\,(u,v) + \frac{\frac{\partial^2}{\partial u^2}V1(u,v)}{r^2} \\ \frac{\partial^2}{\partial v^2}V2\,(u,v) + \frac{\frac{\partial^2}{\partial u^2}V2(u,v)}{r^2} - \frac{V2(u,v)}{r^2} + \frac{}{2\frac{\partial}{\partial u}V1(u,v)}r^2 \\ \frac{\partial^2}{\partial v^2}V3\,(u,v) + \frac{\frac{\partial^2}{\partial u^2}V3(u,v)}{r^2} \end{bmatrix}$$

4.3.16 sHessian

A função *sHessian(param,func,vars)* calcula a matriz Hessiana de uma função escalar (2.37) .

Se o argumento opcional *metric* é incluído, isto é, *sHessian(param,func,vars,metric)*, a função *sHessian* requer que no argumento *param*, seja inserida a expressão da métrica de uma superfície (2.2).

Exemplo

• A matriz hessiana de uma função definida sobre uma tira de um helicoidal cilíndrico :

```
> sHessian(cylindricalstrip, f(u,v), [r,u,v])
```

$$\begin{bmatrix} 0 & -\frac{\frac{\partial}{\partial u}f(u,v)}{r} + \frac{c\frac{\partial}{\partial v}f(u,v)}{r} & 0 \\ -\frac{\frac{\partial}{\partial u}f(u,v)}{r} + \frac{c\frac{\partial}{\partial v}f(u,v)}{r} & \frac{\partial^2}{\partial u^2}f\,(u,v) & \frac{\partial^2}{\partial v\partial u}f\,(u,v) \\ 0 & \frac{\partial^2}{\partial v\partial u}f\,(u,v) & \frac{\partial^2}{\partial v^2}f\,(u,v) \end{bmatrix}$$

Exemplo

• A hessiana sobre um cone:

```
> sHessian(dr^2 + (4 * r^2 * sin(v)^2 * du^2/)a^2 + r^2 * dv^2, f(u,v), [u,v], metric)
```

$$\begin{bmatrix} \frac{\partial^2}{\partial u^2} f(u,v) + 4\left(\frac{\partial}{\partial v} f(u,v)\right)\sin(v)\cos(v) & \frac{\partial^2}{\partial v \partial u} f(u,v) - \frac{\cos(v)\frac{\partial}{\partial u} f(u,v)}{\sin(v)} \\ \frac{\partial^2}{\partial v \partial u} f(u,v) - \frac{\cos(v)\frac{\partial}{\partial u} f(u,v)}{\sin(v)} & \frac{\partial^2}{\partial v^2} f(u,v) \end{bmatrix}$$

4.3.17 sWeingarten

A função *sWeingarten*(param,vars) calcula o operador de Weingarten de uma superfície curva dada em *param*.

Exemplo

• Sobre a superfície dada pela parametrização (4.5)

```
> sWeingarten((sinusoidsurface), [u,v])
```

$$\begin{bmatrix} -\frac{r\cos(u)\sin(v)\left((\cos(v))^2 r^2+1\right)}{\left(2(\cos(u))^2 r^2-r^2\right)(\cos(v))^2-(\cos(u))^2 r^2+r^2+1)^{3/2}} & \frac{r\sin(u)\cos(v)\left((\cos(v))^2 r^2-r^2-1\right)}{\left(2(\cos(u))^2 r^2-r^2\right)(\cos(v))^2-(\cos(u))^2 r^2+r^2+1)^{3/2}} \\ -\frac{r\sin(u)\cos(v)\left((\cos(u))^2 r^2+1\right)}{\left(2(\cos(u))^2 r^2-r^2\right)(\cos(v))^2-(\cos(u))^2 r^2+r^2+1)^{3/2}} & \frac{r\cos(u)\sin(v)\left((\cos(u))^2 r^2-r^2-1\right)}{\left(2(\cos(u))^2 r^2-r^2\right)(\cos(v))^2-(\cos(u))^2 r^2+r^2+1)^{3/2}} \end{bmatrix}$$

• Sobre a superfície tubular dada pela parametrição (4.1):

```
> sWeingarten((tubularsurface), [u,v])
```

$$\begin{bmatrix} \frac{\left(\left(-p^2 r+r^3\right)(\cos(u))^2-r^3+\left(-3a^2-p^2\right)r\right)\sin(u)-a\left(-3(\cos(u))^2 r^2+a^2+p^2+3r^2\right)}{((p^2-r^2)(\cos(u))^2+2\sin(u)ar+a^2+r^2)^{3/2} r} & \frac{p(r+\sin(u)a)}{((p^2-r^2)(\cos(u))^2+2\sin(u)ar+a^2+r^2)^{3/2}} \\ -\frac{p}{\sqrt{(p^2-r^2)(\cos(u))^2+2\sin(u)ar+a^2+r^2}\,r} & -\frac{\sin(u)}{\sqrt{(p^2-r^2)(\cos(u))^2+2\sin(u)ar+a^2+r^2}} \end{bmatrix}$$

5 APLICAÇÕES

Neste capítulo, ilustramos a diversidade de áreas onde as aplicações do pacote *Surf* podem ser implementadas. Mostramos como resultados atuais da literatura poderiam ser facilmente re-obtidos e às vezes até melhorados com o uso deste pacote. Apresentamos abaixo, algumas funções adicionais que implementamos no *Surf* para aplicações específicas nas áreas de mecânica dos fluidos, mecânica quântica, difusão e biologia.

5.1 Mecânica dos fluidos no espaço curvo: continuidade, tensão viscosa, energia e fluxo de fluidos

Como exemplo de uma possível aplicação de mecânica de fluidos em superfícies curvas, podemos mencionar o estudo de instabilidades de interface em células curvas de Hele-Shaw [62]. A influência da curvatura da superfície na dinâmica do fluido foi explorada com o objetivo de controlar geometricamente algumas das propriedades do fluxo. Isso foi feito em sistemas onde a implementação geométrica era analiticamente viável, como esfera, cone e uma superfície de curvatura negativa constante, por exemplo [63–69].

Aqui enfatizamos três equações básicas da mecânica dos fluidos em sua forma diferencial, que são descritas em termos dos operadores diferenciais vistos no capítulo anterior ([32, 70–73]). A primeira delas, a *equação da continuidade*, pode ser expressa por

$$\frac{\partial \rho}{\partial t} + div(\rho \vec{V}) = 0, \tag{5.1}$$

onde ρ é a densidade do fluido e $\vec{V}(u, v)$ sua velocidade no ponto (u, v).

5.1.1 sContinuityEquation

Usando a função *sDivergence* para uma função vetorial, obtemos diretamente a nova função *sContinuityEquation* que determina a equação da continuidade para um fluxo de um fludo sobre uma superfície curva.

Em *sContinuityEquation(param,vecfunc,vars)*, a função densidade é, por padrão, constante. Para modificar esta condição, podemos incluir a opção *density = function*, onde *function* deve ser uma função com as variáveis desejadas.

Exemplo:

• Seja $V =< V_1(u, v), V_2(u, v), V_3(u, v) >$. Supondo que a função densidade depende da variável t, a equação da continuidade sobre uma esfera é:

$$> \; sContinuityEquation(sphere, V, [r,u,v], density = rho(t), symbolic) :$$

$$\frac{d}{dt}\rho\left(t\right) + \frac{\frac{\partial}{\partial r}\left(\rho(t)V1(u,v)r^2\sin(u)\right)}{r^2\sin(u)} + \frac{\frac{\partial}{\partial u}(\rho(t)V2(u,v)r\sin(u))}{r^2\sin(u)} + \frac{\frac{\partial}{\partial v}(\rho(t)V3(u,v)r)}{r^2\sin(u)} = 0$$

5.1.2 sViscousStress

Usando principalmente *sFirstForm* do pacote *Surf*, obtemos uma função *sViscousStress* que determina o tensor de tensão viscosa

$$\pi^{ij} = (\eta - \frac{2}{3}\sigma)g^{ij}V^m_{,m} + \sigma(g^{im}V^j_{,m} + g^{jm}V^i_{,m}), \tag{5.2}$$

sobre uma superfície curva, onde η e σ são os coeficientes de viscosidade de bulk e shear, respectivamente.

A função *sViscousStress(param,func,vars)* retorna o tensor de tensão viscosa, denotado por π^{ij}.

Incluindo o argumento opcional *Stokes*, a função *sViscousStress* retorna os componentes do tensor de tensão viscosa considerando a hipótese de Stokes para o segundo coeficiente de viscosidade.

Exemplo

• O tensor de tensão viscosa em uma tira cilíndrica (Figura 14) , assumindo a hipótese de Stokes:

$$> \; sViscousStress(cylindricalstrip, < V1(u,v), V2(u,v), V3(u,v) >, [r,u,v], Stokes)$$

$$\pi^{11} = 0$$

$$\pi^{12} = \sigma\left(-\frac{V2(u,v)r}{(c^2+r^2)^{3/2}} + \frac{\frac{\partial}{\partial u}V1(u,v)}{r^2} - \frac{c\frac{\partial}{\partial v}V1(u,v)}{r^2}\right)$$

$$\pi^{13} = \sigma\left(-\frac{c\frac{\partial}{\partial u}V1(u,v)}{r^2} + \frac{\left(c^2+r^2\right)\frac{\partial}{\partial v}V1(u,v)}{r^2}\right)$$

$$\pi^{21} = \sigma\left(-\frac{V2(u,v)r}{(c^2+r^2)^{3/2}} + \frac{\frac{\partial}{\partial u}V1(u,v)}{r^2} - \frac{c\frac{\partial}{\partial v}V1(u,v)}{r^2}\right)$$

$$\pi^{22} = \sigma\left(\frac{2\frac{\partial}{\partial u}V2(u,v)}{r^2\sqrt{c^2+r^2}} - \frac{2c\frac{\partial}{\partial v}V2(u,v)}{r^2\sqrt{c^2+r^2}} + \frac{2V1(u,v)}{r^3}\right)$$

$$\pi^{23} = \sigma\left(-\frac{2cV1(u,v)}{r^3} + \frac{\frac{\partial}{\partial u}V3(u,v)}{r^2} - \frac{c\frac{\partial}{\partial v}V3(u,v)}{r^2}\right) - \sigma\left(\frac{c\frac{\partial}{\partial u}V2(u,v)}{r^2\sqrt{c^2+r^2}} + \frac{\sqrt{c^2+r^2}\frac{\partial}{\partial v}V2(u,v)}{r^2}\right)$$

$$\pi^{31} = \sigma\left(-\frac{c\frac{\partial}{\partial u}V1(u,v)}{r^2} + \frac{\left(c^2+r^2\right)\frac{\partial}{\partial v}V1(u,v)}{r^2}\right)$$

$$\pi^{32} = \sigma\left(-\frac{2cV1(u,v)}{r^3} + \frac{\frac{\partial}{\partial u}V3(u,v)}{r^2} - \frac{c\frac{\partial}{\partial v}V3(u,v)}{r^2}\right) - \sigma\left(\frac{c\frac{\partial}{\partial u}V2(u,v)}{r^2\sqrt{c^2+r^2}} + \frac{\sqrt{c^2+r^2}\frac{\partial}{\partial v}V2(u,v)}{r^2}\right)$$

$$\pi^{33} = \sigma\left(\frac{2c^2V1(u,v)}{r^3} - \frac{2c\frac{\partial}{\partial u}V3(u,v)}{r^2} + \frac{2\left(c^2+r^2\right)\frac{\partial}{\partial v}V3(u,v)}{r^2}\right)$$

5.1.3 sEnergyEquation

Agora, com as funções *sFirstForm* e *sViscousStress*, nós ganhamos a função *sEnergyEquation(param,funcvec,vars)* que determina a equação da energia

$$\rho\Big[\frac{\partial}{\partial t}\left(h+\frac{1}{2}V_iV^j\right)+V^j(h+\frac{1}{2}V_iV^j)_{,j}\Big]=\frac{\partial p}{\partial t}+(V_i\pi^{ij})_{,j}+\rho V_iF^iq^i_{,i}. \tag{5.3}$$

onde h é a entalpia, p é a pressão, F^i é a força de campo (corporais) body force, e q^i é a componente do vetor de fluxo de calor.

Por padrão, na função *sEnergyEquation(param,funcvec,vars)*, as forças body e Heat flux são representadas simbolicamente por $[F1, F2, F3]$ e $[q1, q2, q3]$, respectivamente. Para atribuir novas funções coordenadas, devemos usar os argumentos *bodyforce* and *heatflux* igualando-os a funções desejadas.

Também por padrão, o tensor de tensão viscosa é representado na equação de energia simbolicamente por π^{ij}. O argumento opcional *viscousstress* (ou apenas stress) usa a função *sViscousStress* para calcular o tensor π^{ij} e substituir na saída da função *sEnergyEquation*.

Exemplo

• A equação da energia em uma tira cilíndrica:

$$\begin{aligned}
&> \ sEnergyEquation(cylindricalstrip, < V1(u,v), V2(u,v), V3(u,v) >, [r,u,v])\\
&\\
&\rho\left(\frac{V2(u,v)\left(V1(u,v)\frac{\partial}{\partial u}V1(u,v)+V2(u,v)\frac{\partial}{\partial u}V2(u,v)+V3(u,v)\frac{\partial}{\partial u}V3(u,v)\right)}{\sqrt{c^2+r^2}}\right)\\
&+\rho\left(V3\,(u,v)\left(V1\,(u,v)\,\frac{\partial}{\partial v}V1\,(u,v)+V2\,(u,v)\,\frac{\partial}{\partial v}V2\,(u,v)+V3\,(u,v)\,\frac{\partial}{\partial v}V3\,(u,v)\right)\right)\\
&=\frac{V1(u,v)\widehat{\pi}^{11}+r^3(\frac{\partial}{\partial u}V1(u,v))\widehat{\pi}^{12}+\frac{r^3(\frac{\partial}{\partial v}V2(u,v))\widehat{\pi}^{23}}{c^2+r^2}}{r}\\
&+\rho(t)(V1(u,v)\widehat{F1}+V2(u,v)\widehat{F2}+V3(u,v)\widehat{F3})-\frac{\widehat{q}_r(r,u,v)+r(\frac{\partial}{\partial r}\widehat{q}_r(r,u,v))}{r}\\
&-r^3(\frac{\partial}{\partial u}\widehat{q}_u(r,u,v))-\frac{r^3(\frac{\partial}{\partial v}\widehat{q}_v(r,u,v))}{c^2+r^2}
\end{aligned}$$

5.1.4 sMotionEquation

A terceira e última das equações básicas da mecânica dos fluidos que incluímos no pacote *Surf* é a equação do movimento,

$$\rho(f_i-F_i)=g^{jk}\Gamma_{ij,k}, \tag{5.4}$$

onde $\Gamma_{ij}=-pg_{ij}+\pi_{ij}$ são as forças de viscosidade e pressão. Desta fórmula, nós construímos a função *sMotionEquation*.

Exemplo

• A equação do movimento sobre um cilindro:

$$> \; sMotionEquation(cylinder, < V1(u,v), V2(u,v), V3(u,v) >, [r,u,v])$$

$$\begin{aligned}
&\tfrac{V2(u,v)\frac{\partial}{\partial u}V1(u,v)}{r} - \tfrac{(V2(u,v))^2}{r} + V3\,(u,v)\,\tfrac{\partial}{\partial v}V1\,(u,v) \\
&= -\tfrac{\partial}{\partial r}P(r,u,v) + \tfrac{\widehat{\pi}_{11}(r,u,v)}{r} + r\left(\tfrac{\partial}{\partial r}\widehat{\pi}_{11}(r,u,v)\right) \\
&+\tfrac{\partial}{\partial u}\widehat{\pi}_{12}(r,u,v) + r\left(\tfrac{\partial}{\partial v}\widehat{\pi}_{13}(r,u,v)\right) - \tfrac{\widehat{\pi}_{22}(r,u,v)}{r}
\end{aligned}$$

$$\begin{aligned}
&\tfrac{V1(u,v)V2(u,v)}{r} + \tfrac{V2(u,v)\frac{\partial}{\partial u}V2(u,v)}{r} + V3\,(u,v)\,\tfrac{\partial}{\partial v}V2\,(u,v) \\
&= \tfrac{1}{r}\left(- \tfrac{\partial}{\partial u}P(r,u,v) + 2\widehat{\pi}_{21}(r,u,v) + r^2\left(\tfrac{\partial}{\partial r}\widehat{\pi}_{21}(r,u,v)\right)\right) \\
&+\tfrac{1}{r}\left(r\left(\tfrac{\partial}{\partial u}\widehat{\pi}_{22}(r,u,v)\right) + r^2\left(\tfrac{\partial}{\partial v}\widehat{\pi}_{23}(r,u,v)\right)\right)
\end{aligned}$$

$$\begin{aligned}
&\tfrac{V1(r,u,v)V3(r,u,v)}{r} + V1\,(r,u,v)\,\tfrac{\partial}{\partial r}V3\,(r,u,v) \\
&+V1\,(r,u,v)\,\tfrac{\partial}{\partial u}V3\,(r,u,v) + \tfrac{V3(r,u,v)\frac{\partial}{\partial v}V3(r,u,v)}{r} \\
&= -\tfrac{1}{r}\left(\tfrac{\partial}{\partial v}P\,(r,u,v) + 2\,\pi_{31}\,(r,u,v) + r^2\,\tfrac{\partial}{\partial r}\pi_{31}\,(r,u,v)\right) \\
&= \tfrac{1}{r}\left((r^2\,\tfrac{\partial}{\partial u}\pi_{32}\,(r,u,v) + r\,\tfrac{\partial}{\partial v}\pi_{33}\,(r,u,v)\right)
\end{aligned}$$

5.2 Deslocamento quadrático médio

Seja $P : S \times S \times \mathbb{R}^+ \to \mathbb{R}^+$ a função densidade de probabilidade para uma partícula sobre uma superfície S, isto é, $P(\sigma, \sigma', t)$ é a probabilidade de encontrar uma partícula em um elemento de volume dv movendo-se σ' em um tempo t.

Supomos que o processo de difusão seja governado pela equação de difusão

$$\frac{\partial P(\sigma,\sigma',t)}{\partial t} \;=\; D\Delta_g P(\sigma,\sigma',t), \tag{5.5}$$

onde Δ_g é o operador Laplaciano-Beltrami (equação 2.34) em termos de $\sigma = (u, v)$, e D o coeficiente de difusão. Sua solução formal é dada pela evolução do núcleo do calor

$$P(\sigma,\sigma',t) = \exp(Dt\Delta_g)P(\sigma,\sigma',0). \tag{5.6}$$

A expansão de Taylor do operador exponencial na equação acima é dada por

$$\exp(Dt\Delta_g) = \sum_{n=0}^{\infty} \frac{(Dt)^n}{n!}(\Delta_g)^n. \tag{5.7}$$

Note que, considerando $\Omega : S \to \mathbb{R}$ um observável físico definido sobre a superfície S, e supondo que Ω seja bem comportada sob ação do operador Laplaciano Δ_g, a média (ou valor médio esperado) de Ω no tempo t é dada por

$$\langle \Omega(\sigma) \rangle_t = \int_S P(\sigma, \sigma', t) f(\sigma') dv.$$

Utilizando a equação de difusão (5.5) e a identidade

$$\int_S \Big(\Omega(\sigma) \Delta P(\sigma, \sigma', t) - P(\sigma -, \sigma', t) \Delta \Omega(\sigma) \Big) = \int_{\partial S} \Big(\Omega(\sigma) \frac{\partial P}{\partial \mu} - P \frac{\partial \Omega(\sigma)}{\partial \mu} \Big), \tag{5.8}$$

obtém-se

$$\frac{\partial^k \langle \Omega(\sigma) \rangle_t}{\partial t^k} = D^k \int_S \Omega(\sigma) \Delta_g^k P(\sigma, \sigma', t) dv. \tag{5.9}$$

Segue que

$$\left. \frac{\partial^k \langle \Omega(\sigma) \rangle_t}{\partial t^k} \right|_{t=0} = D^k \Delta_g^k \Omega(\sigma'). \tag{5.10}$$

Conclui-se que o valor esperado para o observável físico é dado pela série formal [74]

$$\langle \Omega(\sigma) \rangle_t = \sum_{n=0}^{\infty} \frac{(-Dt\Delta_g)^n}{n!} \Omega(\sigma) \Big|_{\sigma=\sigma'}. \tag{5.11}$$

Da equação (5.11), obtemos a seguinte função implementada no pacote *Surf*.

5.2.1 sMeanSquareDisplacement

A função *sMeanSquareDisplacement* calcula a expansão do deslocamento quadrático médio (em tempo curto) para uma superfície curva. Com a sintaxe *sMeanSquareDisplacement(param,func,vars,n)*, a função retorna o nth termo da expansão; com o argumento opcional *expansion*, a saída é a soma dos termos até a ordem nth. O argumento *func* deve ser uma função que forneça a distância geodésica da origem ao ponto de variáveis dada em $vars$, ou até mesmo a distância geodésica projetada que pode ser mais apropriada em medições de microscopia [75].

Exemplo:

• Expansão do deslocamento quadrático médio para uma esfera:

$>$ *sMeanSquareDisplacement*(sphere, r^2u^2, $[u, v]$, 8, expansion)

$$4\,Dt - \frac{4D^2t^2}{3r^2} - \frac{8\,D^3t^3}{45\,r^4} - \frac{44\,D^4t^4}{315\,r^6} - \frac{248\,D^5t^5}{1575\,r^8} - \frac{12296\,D^6t^6}{51975\,r^{10}} - \frac{422992\,D^7t^7}{945945\,r^{12}} - \frac{4830676\,D^8t^8}{4729725\,r^{14}}$$

Observe que, na ref. [3], a expansão foi obtida analiticamente para uma esfera com até três termos apenas.

Exemplo:

• Expansão do deslocamento quadrático médio para um plano corrugado (ver figura 10), considerando a distância geodésica projetada:

Considere a superfície corrugada definida por:

$$boxeggs := [u, v, z * cos(m * u)sin(m * v)].$$

Para $z = 0$ obtemos o resultado no plano uv:

$> \ sMeanSquareDisplacement(\text{subs}(z = 0, \text{boxeggs}), u * u + v * v, [u, v], 1)$

$$4Dt$$

que é o resultado clássico para o deslocamento quadrático médio no plano.

• Agora, na superfície $boxeggs$, obtemos o primeiro termo:

$> \ sMeanSquareDisplacement(\text{boxeggs}, u * u + v * v, [u, v], 1)$

$$\frac{2\,Dm^2r^2t+4\,Dt}{r^2m^2+1}$$

O segundo termo:

$> \ sMeanSquareDisplacement(\text{boxeggs}, u * u + v * v, [u, v], 2)$

$$\frac{4\,D^2t^2m^6r^4+8\,D^2t^2m^4r^2}{m^6r^6+3\,m^4r^4+3\,r^2m^2+1}$$

E o terceiro termo

$> \ sMeanSquareDisplacement(\text{boxeggs}, u * u + v * v, [u, v], 3)$

$$\frac{32D^3t^3m^{12}r^8+72D^3t^3m^{10}r^6+32D^3t^3m^8r^4-96D^3t^3m^6r^2}{3\,m^{10}r^{10}+15\,m^8r^8+30\,m^6r^6+30\,m^4r^4+15\,r^2m^2+3}$$

5.3 Potencial geométrico da Costa

Dado um ponto p na superfície S, seja R uma vizinhança de p em $\mathbb{R}^3$ parametrizada por

$$R(q_1, q_2, q_3) = \sigma(q_1, q_2) + q_3N(q_1, q_2), \tag{5.12}$$

onde $N(q_1, q_2)$ é o vetor unitário normal a S em $p = (q_1, q_2)$. Denotando $R = R(q_1, q_2)$ e $N = N(q_1, q_2)$, segue de [2] que

$$\frac{\partial R}{\partial q_i} = \frac{\partial r}{\partial q_i} + q_3 \frac{\partial N}{\partial q_i}. \tag{5.13}$$

Como $\frac{\partial N}{\partial q_i}$ está em T_pS,

$$\frac{\partial N}{\partial q_i} = \sum_{j=1}^{2} \alpha_{ij} \frac{\partial r}{\partial q_j}, \tag{5.14}$$

onde α_{ij} são dados pela equação de Weingarten.

$$\begin{aligned} \alpha_{11} &= \frac{1}{g}(g_{12}h_{21} - g_{22}h_{11}), \quad \alpha_{12} = \frac{1}{g}(g_{21}h_{11} - g_{11}h_{21}). \\ \alpha_{21} &= \frac{1}{g}(g_{12}h_{22} - g_{22}h_{12}), \quad \alpha_{22} = \frac{1}{g}(g_{12}h_{21} - g_{11}h_{22}). \end{aligned}$$

Segue que

$$\begin{aligned} \frac{\partial R}{\partial q_i} &= \frac{\partial r}{\partial q_i} + q_3 \frac{\partial N}{\partial q_i} \\ &= \Big(\delta_{ij} + \alpha_{ij} q_3\Big) \frac{\partial r}{\partial q_j} \\ \frac{\partial R}{\partial q_3} &= N(q_1, q_2). \end{aligned} \tag{5.15}$$

As componentes do tensor métrico na vizinhança R são dadas por

$$G_{ij} = \frac{\partial R}{\partial q_i} \frac{\partial R}{\partial q_j}, \qquad i, j = 1, 2, 3, \tag{5.16}$$

Utilizando a equação (5.15), as seguintes igualdades foram obtidas por [2].

$$\begin{aligned} G_{ij} &= g_{ij} + [\alpha g + (\alpha g)^T]_{ij} q_3 + (\alpha g \alpha^T)_{ij} q_3^2, \\ G_{i3} &= G_{3i} = 0, \;\; i = 1, 2; \;\; G_{33} = 1, \end{aligned}$$

e

$$dV = f(q_1, q_2, q_3) dq_1 dq_2 dq_3, \tag{5.17}$$

onde $f(q_1, q_2, q_3) = 1 + tr(\alpha_{ij}) q_3 + det(\alpha_{ij}) q_3^2$

A equação de Schrödinger em coordenadas curvilíneas dadas pela métrica G_{ij} é

$$\frac{\hbar^2}{2m} \sum_{i,j=1}^{3} \frac{1}{\sqrt{G}} \frac{\partial}{\partial q_i} (\sqrt{G} G^{ij} \frac{\partial \psi}{\partial q_j}) + V(q_3)\psi = i\hbar \frac{\partial \psi}{\partial t}, \tag{5.18}$$

onde $G = det(G_{ij})$.

Da equação (5.18), obtêm-se o potencial geométrico da Costa definido por

$$V = -\frac{\hbar^2}{2m}(M^2 - K), \tag{5.19}$$

onde $M := M(u, v)$ and $K := K(u, v)$ denotam, respectivamente, as curvaturas média e Gaussiana.

5.3.1 sVdaCosta

A função *sVdaCosta* calcula o potencial geométrico de energia sobre uma suerfície curva.

Exemplo:

- Consideremos a superfície conica dada por:

$$conicalsurface := [v * S(u) * cos(u), v * S(u) * sin(u), v] : \tag{5.20}$$

onde $S(u) = 1 + \mu \sin(mu)$.

O potencial geométrico de energia sobre uma superfície conica:

```
> sVdaCosta(conicalsurface, [u, v])
```

$$\frac{-h^2\left(2\left(\frac{d}{du}S(u)\right)^2+(S(u))^2-S(u)\frac{d^2}{du^2}S(u)\right)^2\left(1+(S(u))^2\right)^2}{8\,mv^2\left((S(u))^4+\left(\frac{d}{du}S(u)\right)^2+(S(u))^2\right)^3}$$

Figura 15 – (a) Superfície cônica com a curva da diretriz na forma de sinusoide circular; (b) Superfície ondulada de revolução de uma sinusoide.

(a) (b)

Fonte: os autores (obtida com o sofware Maple).

Exemplo:

• O potencial geométrico de energia sobre uma superfície corrugada de revolução de um sinusoide (Figura 15b) :
Seja

$$sinusoidcorrug := [a\sin\left(\frac{n\pi z}{b}\right)\cos(v), a\sin\left(\frac{n\pi z}{b}\right)\sin(v), z]. \tag{5.21}$$

```
> sVdaCosta(sinusoidcorrug, [z, v])
```

$$\frac{-\hbar^2 b^2\left(2\,a^2\left(\cos\left(\frac{n\pi z}{b}\right)\right)^2\pi^2 n^2-\pi^2 a^2 n^2+b^2\right)^2}{8\,ma^2\left(\sin\left(\frac{n\pi z}{b}\right)\right)^2\left(a^2\left(\cos\left(\frac{n\pi z}{b}\right)\right)^2\pi^2 n^2+b^2\right)^3}$$

Exemplo:

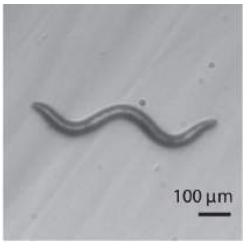
• O potencial geométrico de energia sobre um catenoide, como obtido na referência [76]:

```
> sVdaCosta([R*cosh(z/R)*cos(u), R*cosh(z/R)*sin(u), z], [u, z])
```

$$-\frac{\hbar^2}{2m\cosh\left(\frac{z}{R}\right)^4 R^2}$$

5.4 Caenorhabditis Elegans

Figura 16 – C. Elegans.



Fonte: [4].

O Caenorhabditis Elegans (C. elegans) é um nematódeo considerado um sistema modelo em biologia por ser um dos organismos mais simples com sistema nervoso (Figura 16).

Dada sua forma trasparente, os caminhos (ou formas) do *C. elegans* são rastreados a um nível macroscópio com uso de ferramentas microscopia de video, em um dispositivo que hoje é conhecido como worm tracker [77–79]. Todo o seu genoma e conectoma foram sequenciados [80, 81] para que se possa correlacionar a variabilidade no comportamento motor de *C. elegans* com suas propriedades microscópicas. Portanto, há uma grande

expectativa de que, ao entender profundamente o comportamento macro e microscópico do *C. elegans*, que tem apenas 302 neurônios, pode-se entender princípios fundamentais de outros organismos vivos.

Na ref. [5], métodos de geometria diferencial de curvas em $\mathbb{R}^2$ foram usados em conjunto com técnicas de análise de componentes principais entre outros para rastrear automaticamente o caminho de *C. elegans*. De fato, pode-se recuperar experimentalmente a curvatura do *C. elegans* em séries de Fourier, usando o teorema fundamental para curvas. Notavelmente, com apenas 4 modos de Fourier, é possível encontrar até 95% de suas trajetórias.

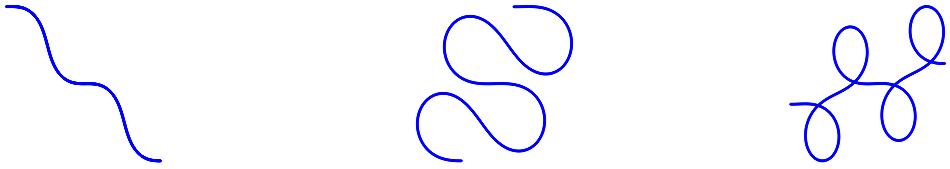
Posteriormente, em [4], foi mostrado que se alguém considerar a curvatura do *C. elegans* como uma função harmônica, pode-se recuperar a maioria de suas formas.

Aqui, podemos recuperar os resultados de [4], integrando a curvatura

$$k(s) = A\sin(qs + \phi), \tag{5.22}$$

pelo teorema fundamental das curvas em $\mathbb{R}^2$ (pag. 18), onde A é a amplitude da variação do corpo, q é o wavevector, ou seja, é a magnitude do vetor de onda (forma), e ϕ é a fase da onda harmônica. Assim, ilustramos as formas de um C. elegans (ver em [4]) , com $q = 0.7$ e $\phi = 3$ fixados, sendo $A = 0.42$, $A = 1.4$ e $A = 2.09$, respectivamente (Figuras 17a,17b e 17c).

Figura 17 – Formas de C. elegans no plano.



(a) (b) (c)

Fonte: os autores (obtida com o pacote *Surf*).

Apesar do enorme sucesso de rastrear o comportamento de *C. elegans* através desta estratégia, pode-se ter novos insights se o rastreamento pudesse ser feito em microscopia tridimensional. A microscopia tridimensional está se tornando mais presente na modelagem de sistemas vivos. Portanto, em algum momento, será necessário desenvolver ferramentas que possam estender o formalismo Eingenworms para microscopia tridimensional. Os aspectos da geometria diferencial de tal extensão são possíveis integrando a equação de Frenet (2.1), que é o chamado teorema fundamental para curvas em $\mathbb{R}^3$ (Pag. 18). Em contraste com a modelagem em $\mathbb{R}^2$, onde apenas a curvatura era o ingrediente necessário para inferir a forma do verme, aqui a curvatura e a torção da curva definem a forma de um *C . elegans* Nas figuras (18a) e (18b) ilustramos formas (faixas) de C. elegans no espaço.

Figura 18 – Formas de C. elegans no espaço.

(a) (b)

Fonte: os autores (obtida com o pacote *Surf*).

O atual, estado da arte do rastreamento de *C. elegans* baseia-se no pressuposto de que é modelado como um objeto unidimensional, isto é, uma curva. Podemos facilmente incluir espessura na modelagem deste nematoide usando ferramentas de geometria diferencial para melhor ilustrar o *C. elegans* em três dimensões. Na verdade, um tubo pode ser parametrizado da seguinte forma [24].

5.4.1 sPlotWorm

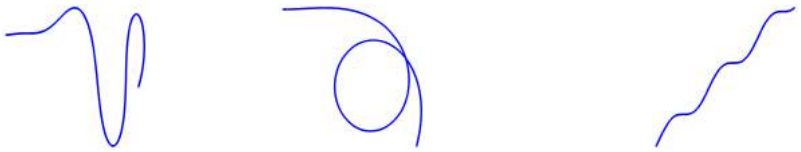
A função *sPlotWorm*(wormparam,var,initialpoint, endpoint) retorna a trilha ou forma do corpo de um verme C. elegans no plano e no espaço.

Quando o argumento *wormparam* é uma função escalar da curvatura, o *sPlotWorm* retorna a forma de C. elegans no plano. Quando o argumento *wormparam* é uma lista com duas entradas, ou seja, uma função curvatura e outra torção, *sPlotWorm* retorna a forma do germe no espaço. Se o argumento *wormparam* é uma lista com três entradas, que descreva uma curva parametrizada, a função *sPlotWorm* retorna uma ilustração de um C. elegans com um tubo parametrizado.

Exemplo:

• C. Elegans no plano.

$> \ \text{sPlotWorm}(0.4 * t * \sin((t/.7) + 1), t, -1, 8);$
$> \ \text{sPlotWorm}(2.09 * \sin(0.5 * t + 3), t, 0, 6);$
$> \ \text{sPlotWorm}(0.4 * \sin((t/2) + 3t), t, -1, 4);$

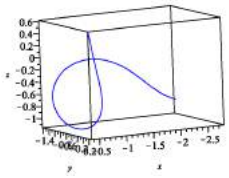


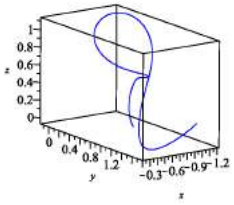
.

Exemplo:

• C. elegans no espaço:

> sPlotWorm($[1.84 * sin(.7 * t + .5), cos(t)], t, 3, 9$)
> sPlotWorm($[3 * cos(t), .3 * t * sin(3 * t)], t, 3, 9$)



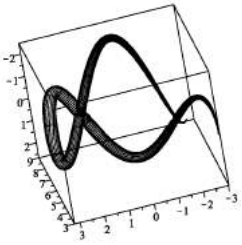


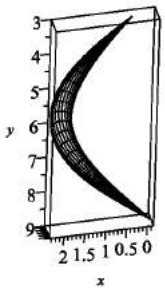
.

Exemplo:

• Ilustração da forma de um C. elegans com um tubo parametrizado.

> sPlotWorm($[3 * cos(t), .3 * t * sin(3 * t), t], t, 3, 9$)
> sPlotWorm($[2.09 * sin(.5 * t + 5), t, 0], t, 3, 9$)





.

6 CONCLUSÃO

Neste trabalho utilizamos o sistema de algébra computacional Maple para desenvolver o novo pacote denominado *Surf*, para o tratamento da geometria diferencial de curvas e superfícies. Ressaltamos que todas as funções do *Surf* foram implementadas com algorítimos baseados na métrica de uma superfície, o que nos permite obter resultados em superficies curvas com coordenadas não necessariamente ortogonais.

Dividimos as funções do *Surf* em três grupos. O primeiro grupo contém funções envolvendo curvas e superfícies que incluímos com o propósito de preencher lacunas que podem surgir durante estudos de superfícies curvas. Tais funções calculam: o vetor normal em um ponto determinado de uma superfície; a curvatura de uma curva na superfície; seu comprimento de arco; a torção (geodésica); as curvaturas normal e geodésica de uma curva; a equação da geodésica; o operador Weingarten; a curvatura de Gaussiana; a segunda curvatura de Gauss e a curvatura média.

O segundo grupo contém as principais funções do pacote *Surf*, que são: a primeira e a segunda forma fundamental; os símbolos de Christoffel; o gradiente de funções escalares ou vetoriais; a divergência de uma função vetorial ou matricial; o rotacional; a Hessiana; e o Laplace-Beltrami de funções escalares ou vetoriais.

Com essas funções, podemos estudar exemplos não triviais e não comuns na literatura, e construir novas funções para o pacote *Surf*. Dessa forma, obtemos o terceiro grupo de funções com aplicações dos outros dois grupos nas diferentes areas de conhecimento: dinâmca dos fluidos, difusão em superfícies, mecânica quântica e biologia.

Para dinâmca dos fluidos, obtemos funções que produzem as equações da continuidade, da energia, e do fluxo de um fluido, além de uma função específica para o cálculo do tensor de tensões viscosas.

Com a função que implementamos para o cálculo da expansão do deslocamento quadrático médio, em tempo curto, reproduzimos os três termos da expansão para a esfera obtida inicialmente por [3]. De fato, apresentamos oito termos da expansão, validando e expandindo o resultado. Além disso, como o *Surf* permite obter o deslocamento quadrático médio para qualquer superfície curva, nossa função pode ser útil para estudos teóricos e experimentais como realizado em [13], quando utilizaram a expansão obtida em [3] para medir trajetorias de nanoparticulas.

Para mecânica quântica, obtemos uma função para calcular o potencial geométrico quântico com base na teoria proposta em [2]. Com nossa função, reproduzimos a expressão do potencial para o catenoide obtido em [76].

Em nossa última aplicação, obtemos uma função que descreve as trilhas do nematoide C. Elegans. Com nossa função, além de reproduzimos as trilhas no plano, como obtido por [4], podemos obter trilhas de um C. elegans no espaço tridimencional.

Para concluir, é oportuno informar que o pacote *Surf* será disponibilizado na plataforma github, no endereço eletrônico https://github.com/marcelopiropo/Surf, com licença CC0 1.0, tendo sido submetido para publicação em [82].

REFERÊNCIAS

[1] Max Jammer. Einstein and religion: physics and theology, 2000. Citado na página 6.

[2] R. C T Da Costa. Quantum mechanics of a constrained particle. *Physical Review A*, 23(4):1982–1987, 1981. Citado 4 vezes nas páginas 12, 14, 60 e 66.

[3] Pavel Castro-Villarreal. Brownian motion meets riemann curvature. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(08):P08006, 2010. Citado 4 vezes nas páginas 12, 13, 58 e 66.

[4] Venkat Padmanabhan, Zeina S. Khan, Deepak E. Solomon, Andrew Armstrong, Kendra P. Rumbaugh, Siva A. Vanapalli, and Jerzy Blawzdziewicz. Locomotion of C. elegans: A Piecewise-Harmonic Curvature Representation of Nematode Behavior. *PLoS ONE*, 7(7):e40121, 7 2012. Citado 5 vezes nas páginas 12, 14, 62, 63 e 67.

[5] Greg J Stephens, Bethany Johnson-Kerner, William Bialek, and William S Ryu. Dimensionality and dynamics in the behavior of c. elegans. *PLoS computational biology*, 4(4):e1000028, 2008. Citado 3 vezes nas páginas 12, 14 e 63.

[6] M. Cahen and M. Flato, editors. *Differential Geometry and Relativity*. Springer, Dordrecht, Netherlands, 1976. Citado na página 12.

[7] Omid Bavi, Charles Cox, Manouchehr Vossoughi, Reza Naghdabadi, Yousef Jamali, and Boris Martinac. Influence of global and local membrane curvature on mechano-sensitive ion channels: A finite element approach. *Membranes*, 6(1):14, 2016. Citado na página 12.

[8] Mohammad EE El-Deeb and Amal AA Abd-El-Hafez. Can vitamin c affect the kbro3 induced oxidative stress on left ventricular myocardium of adult male albino rats? a histological and immunohistochemical study. *Journal of microscopy and ultrastructure*, 3(3):120–136, 2015. Citado na página 12.

[9] Ahad Zarghami, Hamid Reza Ashorynejad, and Johan T Padding. Hydrodynamics forces on a circular particle near a sinusoidal corrugated wall. *Powder Technology*, 342:789–800, 2019. Citado na página 12.

[10] Nakul Prabhakar Bende, Arthur A Evans, Sarah Innes-Gold, Luis A Marin, Itai Cohen, Ryan C Hayward, and Christian D Santangelo. Geometrically controlled snapping transitions in shells with curved creases. *Proceedings of the National Academy of Sciences*, 112(36):11175–11180, 2015. Citado na página 12.

[11] Kristina Navickaitė, Luca Cattani, Christian RH Bahl, and Kurt Engelbrecht. Elliptical double corrugated tubes for enhanced heat transfer. *International Journal of Heat and Mass Transfer*, 128:363–377, 2019. Citado na página 12.

[12] Daniel A Beller, Kim MJ Alards, Francesca Tesser, Ricardo A Mosna, Federico Toschi, and Wolfram Möbius. Evolution of populations expanding on curved surfaces (a). *EPL (Europhysics Letters)*, 123(5):58005, 2018. Citado na página 12.

[13] Yaning Zhong, Luyang Zhao, Paul M Tyrlik, and Gufeng Wang. Investigating diffusing on highly curved water–oil interface using three-dimensional single particle tracking. *The Journal of Physical Chemistry C*, 121(14):8023–8032, 2017. Citado 2 vezes nas páginas 13 e 66.

[14] Fernando Santos, Sébastien Fumeron, Bertrand Berche, and Fernando Moraes. Geometric effects in the electronic transport of deformed nanotubes. *Nanotechnology*, 27(13):135302, 2016. Citado na página 14.

[15] Luiz CB da Silva, Cristiano C Bastos, and Fábio G Ribeiro. Quantum mechanics of a constrained particle and the problem of prescribed geometry-induced potential. *Annals of Physics*, 379:13–33, 2017. Citado na página 14.

[16] Carmine Ortix and Jeroen van den Brink. Effect of curvature on the electronic structure and bound-state formation in rolled-up nanotubes. *Physical Review B*, 81(16):165419, 2010. Citado na página 14.

[17] Julio Brandão, Cleverson Filgueiras, Moises Rojas, and Fernando Moraes. Inertial and topological effects on a 2d electron gas. *Journal of Physics Communications*, 1(3):035004, 2017. Citado na página 14.

[18] Christopher Fang-Yen, Matthieu Wyart, Julie Xie, Risa Kawai, Tom Kodger, Sway Chen, Quan Wen, and Aravinthan DT Samuel. Biomechanical analysis of gait adaptation in the nematode caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 107(47):20323–20328, 2010. Citado na página 14.

[19] Nobelprize.org. The nobel prize. , Nobel Media AB. Disponivel em *https://www.nobelprize.org/prizes/*. Acesso em 25 Ago. 2019. Citado na página 14.

[20] Andrew Pressley. *Elementary Differential Geometry*. Springer Undergraduate Mathematics Series. Springer, London, 2010. Citado na página 15.

[21] Manfredo Perdigão Do Carmo. *Geometria diferencial de curvas superfícies*. Socie Brasileira de Matemática, 1990. Citado na página 15.

[22] John Oprea. *The mathematics of soap films: explorations with Maple*, volume 10. American Mathematical Society, Providence, USA, 2000. Citado 2 vezes nas páginas 15 e 29.

[23] John Oprea. *Differential geometry and its applications*. Mathematical Association of America, Washington, D.C., 2007. Citado na página 15.

[24] Elsa Abbena, Simon Salamon, and Alfred Gray. *Modern differential geometry of curves and surfaces with Mathematica*. Chapman and Hall/CRC, London, 2017. Citado 3 vezes nas páginas 15, 29 e 64.

[25] Dae Won Yoon. On the second gaussian curvature of ruled surfaces in euclidean 3-space. *Tamkang Journal of Mathematics*, 37(3):221–226, 2006. Citado na página 19.

[26] D E Blair. Ruled Surfaces with Vanishing Second Gaussian Curvature. *Monatshefte für Mathematik*, 181:177–181, 1992. Citado na página 19.

[27] Tai L. Chow. *Gravity, black holes, and the very early universe: an introduction to general relativity and cosmology*. Springer, 2010. Citado na página 20.

[28] Aleksandr Ivanovich Borisenko et al. *Vector and tensor analysis with applications*. Dover, New York, 1979. Citado 2 vezes nas páginas 20 e 22.

[29] BA Dubrovin, AT Fomenko, and SP Novikov. *Modern geometry methods and applications. Part I. The geometry of surfaces, transformation groups, and fields.*, volume 93. Springer-Verlag, New York, 1984. Citado na página 20.

[30] David Lovelock and Hanno Rund. *Tensors, differential forms, and variational principles*. Courier Corporation, 1989. Citado 2 vezes nas páginas 20 e 22.

[31] James G Simmonds. *A brief on tensor analysis*. Undergraduate Texts in Mathematics. Springer Science & Business Media, New York, NY, 2012. Citado 2 vezes nas páginas 20 e 22.

[32] Rutherford. Aris. *Vectors, tensors, and the basic equations of fluid mechanics*. Dover Publications, New York, 1989. Citado 2 vezes nas páginas 22 e 54.

[33] Sergey Krivoshapko. *Encyclopedia of Analytical Surfaces*. Springer, 2016. Citado 2 vezes nas páginas 22 e 37.

[34] Harry G Kahrimanian. Analytical differentiation by a digital computer. *MA Thesis, Temple University*, 1953. Citado na página 26.

[35] John F Nolan. *Analytical differentiation on a digital computer*. PhD thesis, Massachusetts Institute of Technology, 1953. Citado na página 26.

[36] Doron Swade. *The difference engine: Charles Babbage*. Viking Adult, 1 amer ed edition, 2001. Citado na página 26.

[37] Bruno Buchberger, George E Collins, Rüdiger Loos, and Rudolph Albrecht. Computer algebra symbolic and algebraic computation. *ACM SIGSAM Bulletin*, 16(4):5–5, 1982. Citado na página 26.

[38] CT Lee and CC Lee. Symbolic computation on a second-order kdv equation. *Journal of Symbolic Computation*, 74:70–95, 2016. Citado na página 26.

[39] Douglas Navarro Guevara. Primitive transcendental functions and symbolic computation. *Journal of Symbolic Computation*, 87:28–53, 2018. Citado na página 26.

[40] OJ Falcón, Raúl M Falcón, and Juan Núñez. Algebraic computation of genetic patterns related to three-dimensional evolution algebras. *Applied Mathematics and Computation*, 319:510–517, 2018. Citado na página 26.

[41] Bo Li, Jianping Zhao, Aimin Pan, Mohammad Mirzazadeh, Mehmet Ekici, Qin Zhou, and Wenjun Liu. Stable propagation of optical solitons in fiber lasers by using symbolic computation. *Optik*, 178:142–145, 2019. Citado na página 26.

[42] George Labahn Keith O. Geddes, Stephen R. Czapor. *Algorithms for Computer Algebra*. Kluwer, 1 edition, 1992. Citado na página 26.

[43] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013. Citado na página 26.

[44] Joel S Cohen. *Computer algebra and symbolic computation: Mathematical methods*. AK Peters/CRC Press, 2003. Citado na página 26.

[45] JA Van Hulzen and J Calmet. Computer algebra systems. In *Computer Algebra*, pages 221–243. Springer, 1983. Citado na página 26.

[46] Michael J Wester. *Computer algebra systems: a practical guide*. Wiley, 1999. Citado na página 26.

[47] AC Hearns. Reduce: the first forty years. In *Invited paper presented at the A3L Conference in Honor of the 60th Birthday of Volker Weispfenning*, 2005. Citado na página 26.

[48] TA Maple. Maplesoft. *Waterloo Maple Inc*, 2019. Citado na página 27.

[49] Huy Nguyen, Hao Shi, Jie Xu, and Shiwei Zhang. Cpmc-lab: A matlab package for constrained path monte carlo calculations. *Computer Physics Communications*, 185(12):3344 – 3357, 2014. Citado na página 28.

[50] Saeed Vatankhah, Vahid Ebrahimzadeh Ardestani, Susan Soodmand Niri, Rosemary Anne Renaut, and Hojjat Kabirzadeh. Igug: A matlab package for 3d inversion of gravity data using graph theory. *Computers & Geosciences*, 128:19 – 29, 2019. Citado na página 28.

[51] Bruno Salvy and Paul Zimmermann. *Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable.* PhD thesis, INRIA, 1992. Citado na página 28.

[52] Solange F. Rutz and Renato Portugal. Finsler: A computer algebra package for finsler geometries. *Nonlinear Analysis: Theory, Methods & Applications*, 47(9):6121 – 6134, 2001. Proceedings of the Third World Congress of Nonlinear Analysts. Citado na página 28.

[53] KG Chetyrkin, Johann H Kühn, and Matthias Steinhauser. Rundec: A mathematica package for running and decoupling of the strong coupling and quark masses. *Computer Physics Communications*, 133(1):43–65, 2000. Citado na página 28.

[54] Tobias Huber and Daniel Maitre. Hypexp, a mathematica package for expanding hypergeometric functions around integer-valued parameters. *Computer physics communications*, 175(2):122–144, 2006. Citado na página 28.

[55] Michael B Monagan, Keith O Geddes, K Michael Heal, George Labahn, SM Vorkoetter, James McCarron, and Paul DeMarco. Maple 9: Advanced programming guide. 2003. Citado na página 28.

[56] Michael B Monagan, Keith O Geddes, K Michael Heal, George Labahn, Stefan M Vorkoetter, JS Devitt, ML Hansen, D Redfern, KM Rickard, et al. *Maple V Programming Guide: For Release 5.* Springer Science & Business Media, 2012. Citado na página 28.

[57] Roman E Maeder. *Programming in mathematica.* Addison-Wesley Longman Publishing Co., Inc., 1991. Citado na página 28.

[58] Martha L. Abell and James P. Braselton. Chapter 7 - introduction to mathematica packages. In Martha L. Abell and James P. Braselton, editors, *Mathematica by Example*, pages 422 – 496. Academic Press, 1992. Citado na página 28.

[59] Desmond J Higham and Nicholas J Higham. *MATLAB guide*, volume 150. Siam, 2016. Citado na página 28.

[60] Brian R Hunt, Ronald L Lipsman, and Jonathan M Rosenberg. *A guide to MATLAB: for beginners and experienced users.* Cambridge university press, 2014. Citado na página 28.

[61] David R Nelson. The statistical mechanics of membranes and interfaces. In *Statistical Mechanics of Membranes and Surfaces*, pages 1–17. World Scientific, 2004. Citado na página 36.

[62] Jacob. Bear. *Dynamics of fluids in porous media*. Dover, 1988. Citado na página 54.

[63] Fernando Parisio, Fernando Moraes, José A Miranda, and Michael Widom. Saffman-taylor problem on a sphere. *Physical Review E*, 63(3):036307, 2001. Citado na página 54.

[64] Luciano Dos Reis and José A Miranda. Controlling fingering instabilities in nonflat hele-shaw geometries. *Physical Review E*, 84(6):066313, 2011. Citado na página 54.

[65] José A Miranda. Analytical approach to viscous fingering in a cylindrical hele-shaw cell. *Physical Review E*, 65(2):026303, 2002. Citado na página 54.

[66] Rodolfo Brandão, João V Fontana, and José A Miranda. Suppression of viscous fingering in nonflat hele-shaw cells. *Physical Review E*, 90(5):053003, 2014. Citado na página 54.

[67] José A Miranda. Nonlinear effects due to gravity in a conical hele-shaw cell. *Physical Review E*, 65(3):036310, 2002. Citado na página 54.

[68] Rodolfo Brandão and José A Miranda. Capillary and geometrically driven fingering instability in nonflat hele-shaw cells. *Physical Review E*, 95(3):033104, 2017. Citado na página 54.

[69] Eduardo SG Leandro, José A Miranda, and Fernando Moraes. Symmetric flows and darcy's law in curved spaces. *Journal of Physics A: Mathematical and General*, 39(7):1619, 2006. Citado na página 54.

[70] James Serrin. Mathematical principles of classical fluid mechanics. In *Fluid Dynamics I/Strömungsmechanik I*, pages 125–263. Springer, New York, 1959. Citado na página 54.

[71] Argyris G Panaras. Boundary-layer equations in generalized curvilinear coordinates, technical report technical memorandum 100003. Technical report, NASA, 1987. Citado na página 54.

[72] Bradley Sutter. The equations of fluid mechanics expressed in curvilinear coordinates. Technical report, Air Force Inst of Tech Wright-Patterson AFB OH, 1962. Citado na página 54.

[73] Hermann Schlichting and Klaus Gersten. *Boundary-layer theory*. Springer, New York, 2016. Citado na página 54.

[74] Alexander Grigor'yan. Estimates of heat kernels on riemannian manifolds. *London Math. Soc. Lecture Note Ser*, 273:140–225, 1999. Citado na página 58.

[75] Pavel Castro-Villarreal. Intrinsic and extrinsic measurement for brownian motion. *Journal of Statistical Mechanics: Theory and Experiment*, 2014, 2014. Citado na página 58.

[76] Rossen Dandoloff, Avadh Saxena, and Bjørn Jensen. Geometry-induced potential on a two-dimensional section of a wormhole: Catenoid. *Physical Review A*, 81(1):014102, 2010. Citado 2 vezes nas páginas 62 e 66.

[77] Steven J Husson, Wagner Steuer Costa, Cornelia Schmitt, and Alexander Gottschalk. Keeping track of worm trackers. *Wormbook, 2005*. Citado na página 62.

[78] Daniel Ramot, Brandon E Johnson, Tommie L Berry Jr, Lucinda Carnell, and Miriam B Goodman. The parallel worm tracker: a platform for measuring average speed and drug-induced paralysis in nematodes. *PloS one*, 3(5):e2208, 2008. Citado na página 62.

[79] Eviatar Yemini, Tadas Jucikas, Laura J Grundy, André EX Brown, and William R Schafer. A database of caenorhabditis elegans behavioral phenotypes. *Nature methods*, 10(9):877, 2013. Citado na página 62.

[80] William R Schafer. The worm connectome: Back to the future. *Trends in neurosciences*, 41(11):763–765, 2018. Citado na página 62.

[81] The C. elegans Sequencing Consortium. Genome sequence of the nematode c. elegans: a platform for investigating biology. *Science*, pages 2012–2018, 1998. Citado na página 62.

[82] Marcelo Piropo, Fernando Moraes, and Fernando AN Santos. Surfing on curved surfaces—the maple package surf. *Submetido para publicação para a Computer Physics Communications em 21/01/2019.* Citado na página 67.

APÊNDICE A - ALGORITMOS

Algoritmo 1: Expansão do núcleo do calor

input : X Parametrização de uma superfície;
$vars$ variáveis independentes da superfície;
f função escalar;
n Ordem n.

output : Termo de n=ésima ordem da expansão do núcleo do calor em tempo curto em X

```
1  for k = 1 .. n do
2      L[k] = ((-i)^k (Dt)^k Δ_g^k f) / k! ;
3      nL[k] = numer(L[k])
4      dL[k] = denom(L[k])
5      for j = 1..2 do
6          if lim_{q^j→0} nL[k] = 0 and lim_{q^j→0} dL[k] = 0 then
7              m=1
8              for i = 1..m do
9                  if lim_{q^j→0} ∂^i nL[k]/∂q^j = 0 then
10                     m=m+1
11                 else
12                     nL[k] := ∂^i nL[k]/∂q^j
13                 end
14             end
15             m=1
16             for i = 1..m do
17                 if lim_{q^j→0} ∂^i dL[k]/∂q^j = 0 then
18                     m=m+1
19                 else
20                     dL[k] := ∂^i dL[k]/∂q^j
21                 end
22             end
23             s[k] = nL[k]/dL[k]
24         else
25             s[k] = lim_{u=o} L[k]
26         end
27         nL[k] = numer(s[k])
28         dL[k] = denom(s[k])
29     end
30     return s[k]
31 end
```

Algoritmo 2: C. ELEGANS

input : X. Função curvatura; ou lista com curvatura e torção

v Variável

$s0$ valor inicial para v

sf valor final para v

output : Traço (ou corpo) de C. elegans no plano ou no espaço

if $numelems(X) = 1$ **then**
- $eq1 \leftarrow \frac{\partial x(s)}{\partial s} = \cos(\theta(s))$
- $eq2 \leftarrow \frac{\partial x(s)}{\partial s} = \sin(\theta(s))$
- $eq3 \leftarrow \frac{\partial \theta(s)}{\partial s} = X$
- sys $\leftarrow \{eq1, eq2, eq3\}$
- ci $\leftarrow [\theta(0) = 0, x(0) = 0, y(0) = 0]$

else if $numelems(X) = 2$ **then**
- $eq1 \leftarrow \frac{\partial x(v)}{\partial v} = t1(s)$;
- $eq2 \leftarrow \frac{\partial y(v)}{\partial v} = t2(s)$;
- $eq3 \leftarrow \frac{\partial z(v)}{\partial v} = t3(s)$;
- $eq4 \leftarrow \frac{\partial t1(v)}{\partial v} = ck * N1(s)$;
- $eq5 \leftarrow \frac{\partial t2(v)}{\partial v} = ck * N2(s)$;
- $eq6 \leftarrow \frac{\partial t3(v)}{\partial v} = ck * N3(s)$;
- $eq7 \leftarrow \frac{\partial N1(v)}{\partial v} = -ck * t1(v) + \text{ctau} * b1(v)$;
- $eq8 \leftarrow \frac{\partial N2(v)}{\partial v} = -ck * t2(v) + \text{ctau} * b2(v)$;
- $eq9 \leftarrow \frac{\partial N3(v)}{\partial v} = -ck * t3(v) + \text{ctau} * b3(v)$;
- $eq10 \leftarrow \frac{\partial b1(v)}{\partial v} = -\text{ctau} * N1(v)$;
- $eq11 \leftarrow \frac{\partial b2(v)}{\partial v} = -\text{ctau} * N2(v)$;
- $eq12 \leftarrow \frac{\partial b3(v)}{\partial v} = -\text{ctau} * N3(v)$;
- $sys \leftarrow \{eq1, eq2, ..., eq12\}$;
- $ic \leftarrow [x(0) = 0, y(0) = 0, z(0) = 0, t1(0) = 1, t2(0) = 0, t3(0) = 0, N1(0) = 0, N2(0) = 1, N3(0) = 0, b1(0) = 0, b2(0) = 0, b3(0) = 1]$;

end

$sol \leftarrow$ solução numérica do sistema sys com condições ci;

Plotar sol para $s0 \leq v \leq sf$