UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE MINAS, METALÚRGICA E MATERIAIS

David Alvarenga Drumond

Desenvolvimento de algoritmos para análise e modelagem variográfica

Porto Alegre
2016

David Alvarenga Drumond

Desenvolvimento de algoritmos para análise e modelagem variográfica

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica e Materiais da Escola de Engenharia da UFRGS, como quesito parcial para obtenção ao título de Mestre em Engenharia.

Orientador: João Felipe Coimbra Leite Costa

Porto Alegre

2016

David Alvarenga Drumond
Desenvolvimento de algoritmos para análise e modelagem variográfica/ David Alvarenga Drumond. – Porto Alegre, 2016-
167 p. : il. (algumas color.) ; 30 cm.

Orientador: João Felipe Coimbra Leite Costa

Dissertação (Mestrado) – UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE MINAS, METALÚRGICA E MATERIAIS , 2016.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

David Alvarenga Drumond

Desenvolvimento de algoritmos para análise e modelagem variográfica

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Minas, Metalúrgica e Materiais da Escola de Engenharia da UFRGS, como quesito parcial para obtenção ao título de Mestre em Engenharia.

Trabalho aprovado. Porto Alegre, 08 de junho de 2016:

João Felipe Coimbra Leite Costa
Orientador

Professor Áttila Leães Rodrigues
Convidado 1

Professor Diego Machado Marques
Convidado 2

Professora Vanessa Cerqueira Koppe
Convidado 3

Porto Alegre
2016

Dedico esta obra aos meus pais que tanto se esforçaram para que eu tivesse condições para o estudo. Ao meu irmão pela mão amiga sempre nas dificuldades do desenvolvimento desta dissertação e aos colegas que auxiliaram na minha caminhada. .

Agradecimentos

Ao Programa de Pós-Graduação em Engenharia Metalúrgica, Materiais e Minas, PPGEM, pela oportunidade de realização de trabalhos em minha área de pesquisa. Ao orientador João Felipe Coimbra Leite Costa pela oportunidade de estudo e da realização dos meus trabalhos. Aos colegas do PPGEM pelo seu auxílio nas tarefas desenvolvidas durante o curso e apoio na revisão deste trabalho. Em especial para o M.Sc. Rafael Moniz Caixeta pelo auxílio nos desenvolvimentos nas rotinas em C++ , ao Dr. Áttila Leães Rodrigues e ao M.Sc. Péricles Lopes Machado pelos desenvolvimentos em Python. A CAPES pela provisão da bolsa de mestrado.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

A análise da continuidade espacial inclui uma série de ferramentas para estimar e modelar a continuidade de variáveis aleatórias regionalizadas. Ela é a base para muitas das avaliações de depósitos minerais baseadas na geoestatísitca. O modelo ajustado é de grande importância e influencia nos resultados em vários algoritmos de krigagem e simulações subsequentes. Tanto os softwares acadêmicos e comerciais podem melhorar no desenvolvimento dos gráficos, na interatividade com o usuário e no uso de formas automáticas de modelagem.

O SGeMS (Stanford Geoestatistical Modeling Software) é um programa gratuito usado entre a comunidade de geoestatísticos ao qual tem um grande potencial de desenvolvimento, mas que, no entanto, ainda não possui todas as ferramentas de análise da continuidade espacial incorporadas. Diferentemente do SGeMS, o GSLIB é uma boa biblioteca gratuita para análise geoestatística e é mais completa, mas as estradas do programa são modificadas pela edição de arquivos .txt e usando linhas de comando o que torna a utilização do software pouco amigável com o usuário, apesar da robustez e qualidade dos programas da biblioteca. Dada as limitações dos mais usados e completos softwares gratuitos de geoestatística, essa dissertação objetiva a transcrição e adaptação do algoritmo do GSLIB (GamV .f) para o software SGeMS, modificando a lógica de programação para criar diferentes ferramentas auxiliares como h-scatterplots e mapas de variograma e covariograma.

Os resultados demonstraram que a adaptação de algoritmos antigos leva a uma solução gratuita. Além disso, um algoritmo para a otimização da modelagem de variogramas pelo método dos mínimos quadrados foi desenvolvido. As rotinas foram desenvolvidas ambas em C++ e em Python. Os algoritmos foram validados com os valores obtidos pelo software GSLIB. Todos os desenvolvimentos dos plug-ins foram testados e validados usando dois casos ilustrativos: um depósito de ferro e um caso polimetálico. Os resultados provaram ser consistentes e similares com aqueles obtidos com softwares comerciais e renomados.

Palavras-chaves: Variogramas, GSLib, Análise Estrutural, GamV, H-Scatter Plot, Mapas de Variograma.

Abstract

The spatial continuity analysis includes a serie of tools to estimate and model the continuity of regionalized random variables. It is the basics for many mineral deposit evaluation methods based on geostatistics. The model adjusted is of paramount importance and influences the results in many subsequent kriging and simulation algorithms. Both commercial and academic softwares can be improved in graphics, users interactivity with and automated tools for modeling spatial continuity.

SGeMS (Stanford Geoestatistical Modeling Software) is a freeware program used among the geostatistical community which has an extremely potential for development however it does not have enough variographic or graphical tools. Unlike SGeMS, GSLIB is a good and more complete free library for geostatistical analysis, however the program inputs are modified by editing of .txt files and uses DOS command lines. This makes the software less user friendly, despite its robustness and quality. Given the limitation on both most used and complete freeware geostatistical softwares, this dissertation aims at transcripting and adpating an algorithm from GSLIB(GamV.f) into SGeMS software, handling the programming logic to create different auxiliary tools as h-scatterplot and variomaps.

The results demonstrated that the adaptation of the old and stable algortihms lead to an inexpensive solution. Futhermore, an algorithm was developed for optimizing variogram modeling based on weigthed least squares method. The routines were developed in both C++ and Phyton. The algorithms were validated against actual values generated by GSLIB. All developed of plug-ins were tested and validated using two illustration studies: an iron ore deposit and a polymetallic one. The results proved to be consistent and similar to the ones obtained by commercial well known sofwares.

Key-words: Variograms, GSLib, Structural Analysis, GamV, H-Scatter Plot, Variomaps

Lista de ilustrações

1.1 Pirâmide de hierarquia das metodologias geoestatísticas. 21
1.2 Continuidade espacial como uma analogia de perfis topográficos. 22
3.1 Parâmetros de entrada do VarMap – GSLib. (DEUTSCH; JOURNEL et al., 1998) 25
3.2 Variografia no SGeMS. 26
7.1 Notação geoestatística para a definição de um lag h em uma direção dentro de um domínio D de uma amostra com suporte x. Dois pares de pontos, um considerado head value, ou ponta do vetor e outro considerado tail value, ou início do vetor. Figura modificada (Wackernagel, 1995). 32
7.2 Funções de continuidade espacial como uma interpretação de vetores. a) Variograma como uma diferença de vetores b) Covariograma como uma projeção de vetores. 33
7.3 H-scatterplots de uma variável para lags de 0,1m , 0,2m, 0,3m e 0,4m. Aumento da descorrelação de acordo com o aumento do comprimento dos lags. Figura retirada de (ISAAKS et al., 1989). 34
8.1 Interpretação geométrica do semi-variograma como sendo a distância de um ponto em relação a reta x=y. 37
8.2 Cálculo de variogramas experimentais segundo um lag unitário na direção Leste-Oeste. 39
8.3 Cálculo de variogramas experimentais segundo o dobro do lag na direção leste-oeste. 39
8.4 Função variograma experimental para os cálculos nas Figuras 9 e 10 e ajuste em um modelo hipotético. P1 e P2 representam os pontos para um lag unitário e o dobro do lag. 40
8.5 Busca de pontos da função de continuidade para amostras irregularmente espaçadas. 41
8.6 Busca de pares de pontos em uma direção tridimensional. Busca de pares elíptica utilizada no SGeMS e em caixa utilizada no GSLib. 41
9.1 Parâmetros do variograma. 44
9.2 Parâmetros da função covariograma. 44
9.3 Representação do modelo de anisotropia geométrico. Elipsóide com valores de alcance mínimo médio e alcance máximo. 46
9.4 Anisotropia zonal representada nos variogramas a) Diferença entre as direções dos variogramas b) Representação da anisotropia por variogramas com patamares diferentes. Figura modificada(GOOVAERTS, 1997). 47

10.1 Teste geométrico do par de amostras segundo o critério de aceitação do azimute. P1 e P2 são os pares de amostras separados por uma projeção XY no plano horizontal. X e Y são as diferenças entre cotas e Projeção azimutal é a projeção dos pontos amostrais na direção do azimute. 53
10.2 Algoritmo das distâncias euclidianas em Python para aceitação do par permissível. 54
10.3 Algoritmo da busca de pares. 55
10.4 Algoritmo da busca de pares em pseudocódigo. 56
11.1 Gráficos de hscatterplot para uma direção específica a) H-scatterplot para um lag de 1500m com correlação 0,59 b) H-scatterplot para um lag de 3000m com correlação 0,49 c) H-scatterplot para um lag de 6000m com correlação 0,15. 57
11.2 Interface do usuário do programa hscatter. I – caixa de definição das propriedades da ponta do vetor, II – caixa de definição das propriedades do início do vetor, III – Caixa da direção dos h-scatterplots, IV- Parâmetros de busca do variograma. 59
11.3 Parâmetros do mapa de variogramas. I – Caixa para definição da propriedade da cabeça do vetor, II- Caixa para definição de propriedades do início do vetor, III – Seleção da função para medida da continuidade espacial, IV – Direção do plano para o mapa de variogramas, orientação azimutal em relação ao norte V- Parâmetros do variograma experimental. 61
11.4 Output do plug-in de mapas de variogramas. Mapa criado a partir de um corpo geológico de minério de ferro plotado no plano horizontal. Maiores valores de variograma no vermelho e menores no azul. Direção de maior continuidade em 52°NE. 62
11.5 Demonstração do arquivo de report do plug-in. 63
11.6 Demonstração do arquivo de saída xml do plug-in. 63
11.7 Parâmetros do plug-in de variogramas experimentais. I – Caixa para definição das propriedades da ponta do vetor, II – Caixa para definição das propriedades do início do vetor, III- Caixa de salvamento do arquivo de report a ser utilizado no plug-in de ajuste automático, IV- Caixa de salvamento do arquivo de relatório do variograma, V- Caixa de seleção da função de continuidade espacial , VI- Caixa para definir as direções de cálculo do variograma , VII - Caixa de parâmetros do variograma experimental, VIII -Caixa de inversão do eixo do correlograma. 65

11.8 Parâmetros do plug-in de variogramas experimentais. I – Caixa para definição das propriedades da ponta do vetor, II – Caixa para definição das propriedades do início do vetor, III- Caixa de salvamento do arquivo de report a ser utilizado no plug-in de ajuste automático, IV- Caixa de salvamento do arquivo de relatório do variograma, V- Caixa de seleção da função de continuidade espacial , VI- Caixa para definir as direções de cálculo do variograma no plano. Define-se o azimute, o dip da reta de máxima continuidade do plano e o número de direções. , VII - Caixa de parâmetros do variograma experimental, VIII -Caixa de inversão do eixo do correlograma. 66
11.9 Arquivo de entrada do otimizador do ajuste variográfico. 67
11.10 Eixos da continuidade espacial. 70
11.11 Parâmetros do otimizador da modelagem. I- Endereço do arquivo de input , II – Inserção do número de interações, III- Inserir o número de variogramas no arquivo, IV – Inserir o número de variáveis, V- Inserir o número de lags, VI-Inserir o número de estruturas a serem modeladas, VII- Inserir o mínimo número de pares para um dado lag, VIII- Inserir as direções de ajuste da modelagem , IX – Inserir as restrições do modelo. 71
12.1 Validação do número de pares do variograma experimental. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLib na direção de 157°N. 72
12.2 Validação Pairwise. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLib na direção de 157°N. 73
12.3 Validação variograma. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N. 73
12.4 Validação variograma relativo. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N. 73
12.5 Validação madograma. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N. 74
12.6 Validação covariograma. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N. 74
12.7 Validação correlograma. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N. 74
12.8 Visão em planta de uma simulação gaussiana com máxima continuidade na direção N45, mergulho 45NE. 75
12.9 Visão em seção vertical frontal E-W de uma simulação gaussiana com direção de maior continuidade N45 e mergulho 45NE. 76
12.10 Visão em seção vertical frontal E-W de uma simulação gaussiana com direção de maior continuidade N45 e mergulho 45NE. 77

12.11 Mapa de variogramas realizado no plano vertical com strike na direção 45ºN. Maior continuidade do modelo simulado demonstrado no perfil abaixo. 78
12.12 Comparação entre os mapas de variograma do ISATIS e do plugin. A) Mapa do software ISATIS e B) Mapa do software SGeMS. Similaridade da continuidade segundo a direção 157ºN. Mapa no plano horizontal 79
13.1 Depósito de ferro do tipo Lago Superior da mina de Conceição, Itabira, Minas Gerais. Fonte: http://www.zjmineracao.com.br/noticia.php?id=578 acessado 17/03/2016. 82
13.2 Mapa horizontal do depósito de ferro. Abaulamento no terceiro quadrante. 83
13.3 Mapa do depósito de ferro com perfil vertical. Mergulho característico na direção NE. 83
13.4 Histograma da distribuição dos dados amostrais do depósito de Ferro. . . 84
13.5 Mapa de variogramas no plano horizontal do depósito de Ferro. A) Variomap calculado pelo plugin e B) Variomap calculado pelo Isatis. Continuidade característica na direção N52 e N22,5. 85
13.6 Varmap no plano com direção do strike N52, mergulho 90 a) Mapa de variogramas realizado no plugin. b) Mapa de variogramas realizado no ISATIS. 86
13.7 Varmap no plano com direção do strike N322, mergulho 38SW. a) Mapa de variogramas realizado no plugin. b) Mapa de variogramas realizado no ISATIS. 86
13.8 Direção de maior continuidade no plano horizontal. Variograma mais contínuo na direção de azimute N52. 87
13.9 Direções de maior continuidade na direção vertical. Variograma mais contínuo na direção do mergulho 52NE. 88
13.10 Direções de maior continuidade nas direções perpendiculares à de maior continuidade espacial. Variograma mais contínuo na direção $N232,38SW$. 89
13.11 Variograma da direção principal automatizado.. Variograma mais contínuo na direção $N52,52NE$. 90
13.12 Variograma da direção secundária. Variograma mais contínuo na direção $N142,0SE$. 90
13.13 Variograma da direção intermediária. Variograma mais contínuo na direção $N232,38SW$. 91
14.1 Mapa de disposição das amostras de Pb. Dados isotópicos para todas as variáveis, Ni,Cu e Pb. 92
14.2 Correlação entre a variável Ni e Cu. 93
14.3 Correlação entre a variável Pb e Cu. 93
14.4 Correlação entre a variável Pb e Ni. 94
14.5 Mapa da continuidade espacial da variável Cu. 94

14.6 Variogramas da variável níquel. Caso omnidirecional. 95
14.7 Variogramas da variável chumbo. Caso omnidirecional. 96
14.8 Variogramas da variável cobre. Caso omnidirecional. 96
14.9 Variogramas dos dados de Jura. Variável 1- Cu , 2-Ni e 3-Pb. Valores de índices iguais representam variogramas diretos. Valores de índices diferentes representam variogramas cruzados. 97

Lista de tabelas

3.1 Softwares de geoestatística gratuitos. 25
3.2 Ferramentas de auxílio à variografia 26
8.1 Funções de continuidade espaciais experimentais 38
13.1 Parâmetros experimentais dos variogramas modelados automaticamente . 89
14.1 Parâmetros do variograma experimental. 95
14.2 Tabela dos valores modelados dos variogramas diretos e cruzados do depósito de Jura segundo um modelo linear de corregionalização 98

Lista de abreviaturas e siglas

GSLIB	Geostatistical Software Library
SGeMS	Stanford Geostatistical Modeling Software

Lista de símbolos

NE	Nordeste
NW	Noroeste
SE	Sudeste
SW	Sudoeste
h	Vetor segundo uma direção no espaço
$\gamma(h)$	Função variograma para uma distância h.
$C(h)$	Função covariograma para uma distância h.
$C(0)$	Variância à priori dos dados.
$\rho(h)$	Função correlograma para uma distância h.
$Gauss(\frac{a}{d})$	Modelo gaussiano para um alcance a em uma direção d.
$Exp(\frac{a}{d})$	Modelo exponencial para um alcance a em uma direção d.
$Sph(\frac{a}{d})$	Modelo esférico para um alcance a em uma direção d.
$\gamma(h) = c + C.model(\frac{a}{d})$	Modelo de variograma com um efeito pepita c, contribuição C e uma função variograma com alcance a e direção d.

Sumário

I ESTADO DA ARTE 19

1 INTRODUÇÃO 20

2 JUSTIFICATIVA 24

3 APRESENTAÇÃO DO PROBLEMA 25

4 META E OBJETIVOS 28

5 METODOLOGIA 29

6 OGANIZAÇÃO DA DISSERTAÇÃO 30

II REFERENCIAIS TEÓRICOS 31

7 DEFINIÇÕES 32
7.1 Definição de continuidade espacial e variografia 32
7.2 Dependência espacial 33
7.3 Hipótese de estacionaridade 34

8 FUNÇÕES EXPERIMENTAIS DE CONTINUIDADE ESPACIAL . . 35
8.1 Efeito dos dados sobre os valores experimentais 35
8.2 Funções de continuidade espacial mais comuns 35
8.3 Outras funções experimentais 37
8.4 Parâmetros de busca 40

9 MODELAGEM DE FUNÇÕES DE CONTINUIDADE ESPACIAL . . 43
9.1 Modelos de variogramas permissíveis 43
9.2 Parâmetros das funções de continuidade 43
9.3 Modelos de continuidade espacial mais comuns 44
9.4 Anisotropia 45
9.5 Funções de continuidade espacial cruzadas 47
9.6 Modelo linear de corregionalização 48
9.7 Modelagem automática de variogramas 49

III DESENVOLVIMENTO 51

10 BUSCA DE PARES DE PONTOS 52

11 ALGORITMOS 57
11.1 Algoritmo do hscatterplot 57
11.2 Algoritmo do mapa de variogramas 59
11.3 Algoritmo de variogramas experimentais 62
11.4 Algoritmo para modelagem automática de variogramas 66

12 VALIDAÇÃO DOS ALGORITMOS DE VARIOGRAMAS 72
12.1 Validação dos variogramas experimentais 72
12.2 Validação do plug-in do mapa de variogramas 75

IV ESTUDO DE CASO 80

13 ESTUDO DE CASO DEPÓSITO DE FERRO (UNIVARIADO) . . . 82

14 ESTUDO DE CASO DO DEPÓSITO DE JURA (CASO MULTIVARIADO) 92

15 RESULTADOS E DISCUSSÃO 99

V CONCLUSÃO E TRABALHOS FUTUROS 100

16 CONCLUSÃO 101

17 TRABALHOS FUTUROS 102

REFERÊNCIAS 103

APÊNDICES 105

APÊNDICE A – ALGORITMO DO HSCATTERPLOT 106

APÊNDICE B – ALGORITMO DO VARMAP 115

APÊNDICE C – ALGORITMO DOS VARIOGRAMAS EXPERIMENTAIS 127

APÊNDICE D – ALGORITMO DA MODELAGEM AUTOMÁTICA 146

Parte I

Estado da arte

1 Introdução

A geoestatística é o conjunto de técnicas fundamentadas na teoria das variáveis regionalizadas com o objetivo de analisar variáveis aleatórias no espaço. Possui vigor na modelagem de depósitos minerais, petróleo, agricultura, engenharia civil e demais disciplinas que se caracterizam por amostragens de alto custo, informações escassas e erráticas. A análise espacial procede com três grandes domínios que são o alicerce dos métodos geoestatísticos (WACKERNAGEL, 2013) sendo eles:

1. Descrição dos dados: O objetivo inicial da geoestatística é verificar a disposição espacial das amostras em campo, observar seus valores e tirar conclusões acerca das propriedades físico-químicas analisadas do depósito mineral. Caso a descrição dos dados demonstrar momentos estatísticos de primeira e segunda ordem abaixo de valores pré-estabelecidos, a chance de se obter uma jazida mineral reduz. A descrição dos dados caracteriza o comportamento global das variáveis aleatórias pela utilização de estatística clássica, mapas das amostras e estimativa da continuidade espacial do fenômeno. A análise variográfica mapeia a continuidade espacial do fenômeno, e influencia na decisão de usar ou não um maior número de informações para uma estimativa local, no raio de busca utilizado na estimativa, na variância do erro e nos pesos da krigagem. A descrição estatística dos dados é o primeiro passo para resumir as informações em um domínio espacial e a primeira ferramenta de decisão na avaliação dos recursos.

2. Interpretação: Os gráficos obtidos pela informação numérica são avaliados tendo em conta a experiência, dados similares e fatos científicos relacionados com a variável em estudo. A interpretação da estrutura espacial, as suas associações e causas, levam a um modelo de continuidade espacial que é ajustado aos dados.

3. Estimativa: Nesta fase estimam-se os valores esperados e suas variações em regiões do domínio físico onde não há informação. A krigagem é o método usado para interpolar valores precisos com o menor erro quadrático usando um conjunto de amostras localizadas no entorno do local estimado. A estimativa e a maioria dos algoritmos de simulação são baseadas nos modelos de continuidade espacial, que influenciam nos pesos dados às amostras usadas para estimar em uma região onde não há informações.

A geoestatística é uma ciência que provê o conjunto de ferramentas que garantem a melhor estimativa quando se lida com variáveis regionalizadas. Nota-se que os modelos de continuidade espacial são o alicerce do desenvolvimento dos métodos geoestatísticos

como demonstrado na pirâmide hierárquica da Figura 1.1. Como é um modelo de base conclui-se que a sua infidelidade pode afetar os resultados na análise do depósito.

Figura 1.1 – Pirâmide de hierarquia das metodologias geoestatísticas.

A análise da continuidade espacial mede a correlação ou descorrelação entre pares de dados dispostos em uma ou várias direções . Por isso necessita-se de um número amplo de pares alinhados em várias direções possíveis para uma análise completa no espaço. Os resultados desse modelo de continuidade espacial são posteriormente simplificados e representados por um elipsoide de anisotropia.

O conceito de continuidade espacial pode ser entendido graficamente usando os perfis topográficos como na Figura 1.2, que representa a suavização das cotas do terreno segundo uma direção. O gráfico da esquerda é mais suave do que o da direita à medida que para uma pequena variação da distância d temos pequenas variações para a distância H. Nota-se que estas representações são uma demonstração de um fenômeno estacionário, tal que os valores flutuam em meio a um valor médio constante. Um perfil topográfico de uma depressão não geraria um modelo permissível de variograma, apresentando uma tendência constante ao longo da distância.

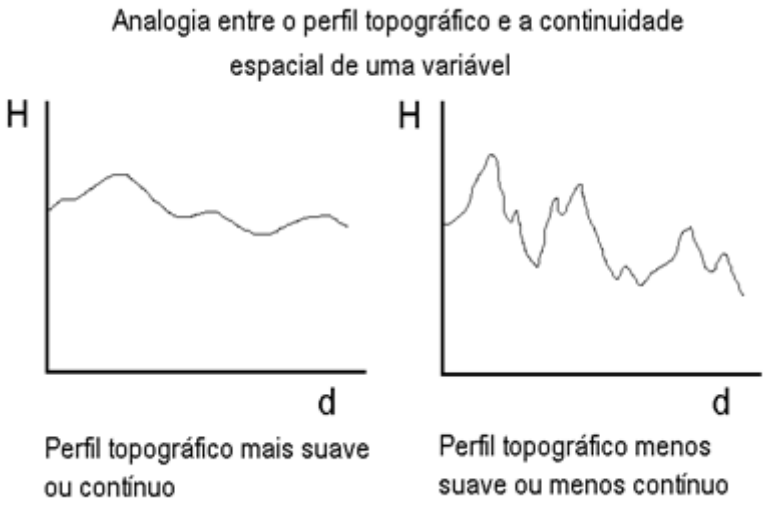

Figura 1.2 – Continuidade espacial como uma analogia de perfis topográficos.

Admitir uma maior continuidade espacial para uma variável aleatória, significa considerar que seus valores são mais similares entre si quão mais proximais. Se uma variável apresentar efeito pepita puro, as técnicas de estatística clássica seriam mais recomendáveis que a geoestatística. A existência da continuidade espacial é o que define os métodos estocásticos e os distinguem das técnicas clássicas de interpolação.

Como a variografia impacta sobre os resultados das estimativas, torna-se imperioso que seja bem calculada e modelada. Sabe-se que existem dificuldades relacionadas à sua determinação experimental como valores outliers, heterocedasticidade ou agrupamento de dados.

A continuidade espacial é uma estimativa realizada a partir das operações que medem as diferenças ou similaridade entre amostras segundo direções escolhidas. Se as amostras são esparsas, ou estejam agrupadas preferencialmente, ou a variabilidade do fenômeno a ser modelado for muito alta, os modelos podem à posteriori ser mal estimados. Determinar e interpretar modelos com dados esparsos ou muito variáveis torna a tarefa da variografia problematizada, principalmente em depósitos de petróleo que apresentam poucos poços (GRINGARTEN; DEUTSCH, 2001).

A modelagem da continuidade espacial também apresenta condicionantes. Um modelo conciso envolve variogramas em várias direções e muitas vezes ajustados por critérios visuais e empíricos sobre os pontos experimentais. A prática da modelagem muitas vezes é realizada pelo ajuste visual, o que se confirma até nos dias atuais (CRESSIE, 1985). A visualização de um número extenso de variogramas prejudica a obtenção de um melhor ajuste. Para fins ilustrativos, é demonstrada a Equação 1.1 que determina o número mínimo de variogramas para a caracterização da continuidade espacial de um caso

multivariado em uma dimensão (LARRONDO; NEUFELD; DEUTSCH, 2003):

$$N = \frac{n(n+1)}{2} \tag{1.1}$$

Em que N é o número mínimo de variogramas para um número n de variáveis associadas ao problema. A Equação 1.1 é análoga para o número de dimensões consideradas. Incorporando as duas propriedades em uma única função, determina-se a Equação 1.2:

$$N = \frac{d(d+1)}{2}\frac{n(n+1)}{2} \tag{1.2}$$

Em que N é o número mínimo de variogramas para um número n de variáveis e para um número d de dimensões. Por exemplo, para o caso de uma análise da continuidade espacial em duas dimensões e uma variável seriam necessários um número mínimo de três variogramas para determinar a continuidade espacial do fenômeno. A Equação 1.2 considera apenas a determinação de um número mínimo de variogramas, porém em prática utiliza-se o máximo de direções possíveis na análise. Convenciona-se que para uma variável são necessários apenas 1 variograma em uma dimensão, 8 para duas dimensões e 24 para as três dimensões de maior, média e menor continuidade.

Estabelecer um modelo de continuidade espacial válido não é apenas um problema teórico, mas também prático e operacional. Conjugar diferentes ferramentas matemáticas que possam auxiliar no reconhecimento geológico, permite ao avaliador do depósito reduzir os riscos associados ao desenvolvimento do empreendimento mineral e garantir a redução dos erros na avaliação.

2 Justificativa

Dada à importância da variografia nos métodos geoestatísticos a utilização de ferramentas de auxílio na análise da continuidade espacial é significativa para uma melhor compreensão do fenômeno a ser estudado. O desenvolvimento de metodologias mais acessíveis requer mais recursos computacionais (TAHMASEBI; HEZARKHANI; SAHIMI, 2012). Por vezes é necessário simplificar os métodos evitando a exigência dos processadores, tal como nos modelos de Markov (MA; JOURNEL, 1999) e em outros modelos multivariados. Essas simplificações advêm do estabelecimento de modelos linearmente dependentes e proporcionais que evitam a análise convencional laboriosa, tal como no modelo linear de corregionalização usado na cokrigagem. A modelagem de variogramas é facilitada com um conhecimento à priori da continuidade espacial do fenômeno, que pode ser obtido utilizando-se mapas variográficos ou funções de estimativa espaciais mais robustas, que evitem problemas advindos pela presença de valores outliers. Com isso, podem-se obter avaliações mais rápidas da continuidade espacial de forma a simplificar o processo de obtenção dos modelos utilizados na estimativa.

A complexidade da modelagem variográfica em problemas reais é sentida tanto no meio acadêmico como na indústria. Tarefas que poderiam ser simplificadas pela utilização de recursos simples, se tornam laboriosas principalmente quando lidam com múltiplas variáveis em três dimensões. O desenvolvimento de softwares livres permite o uso mais intenso de ferramentas matemáticas na avaliação de depósitos minerais principalmente no alicerce da geoestatística e na variografia.

3 Apresentação do problema

A Tabela 3.1 apresenta os principais desenvolvimentos de softwares de domínio público. Apesar de alguns programas apresentarem rotinas de auxílio à variografia tal como o GSLib, muitas vezes possuem interfaces pouco amigáveis, o que torna a sua utilização não tão recorrente no meio profissional.

Tabela 3.1 – Softwares de geoestatística gratuitos.

Software	**Plataforma**	**Linguagem**
GSLIB	Multiplataforma	Fortran
GSTL	Multiplataforma	C++
HPGL	Multiplataforma	Python
mGstat	Multiplataforma	MatLab

A Figura 3.1 representa os parâmetros de entrada de um programa da biblioteca GSLib. A entrada de dados é realizada pela edição de arquivos do tipo .txt e processamento em janela DOS, apesar de todos os algoritmos serem robustos e estáveis. Além disso, é necessário ao usuário deter bom conhecimento dos parâmetros do programa. Apesar do processamento numérico do software ser inegavelmente eficiente, a visualização gráfica e o manuseio não são a prioridade para este conjunto de softwares.

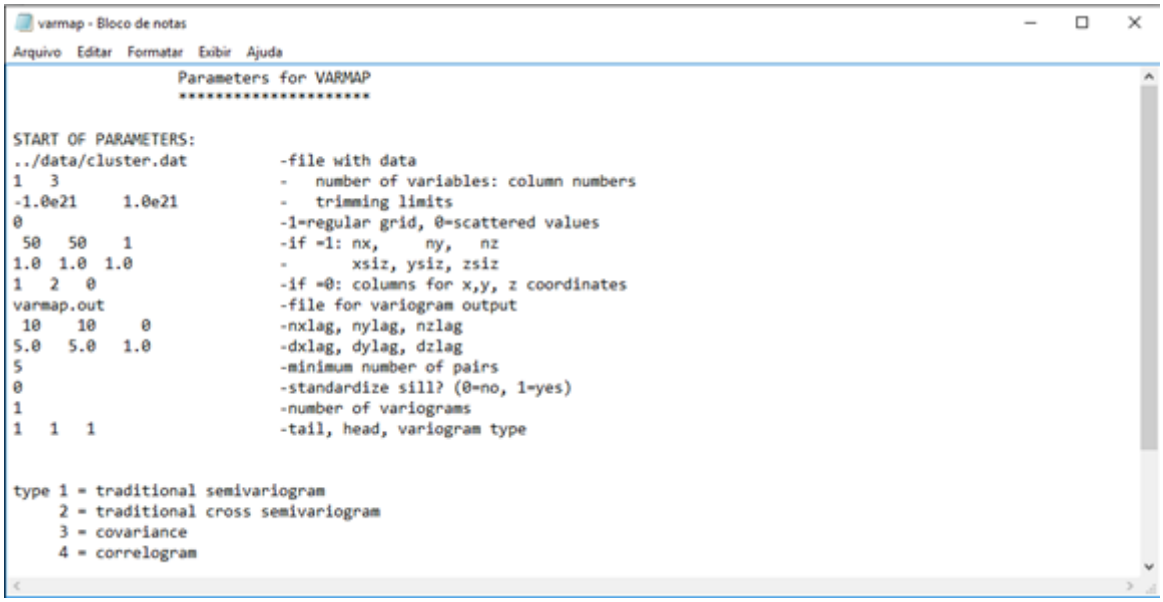

Figura 3.1 – Parâmetros de entrada do VarMap – GSLib. (DEUTSCH; JOURNEL et al., 1998)

O programa SGeMS apresenta uma interface gráfica interativa que melhor se

adéqua às necessidades do usuário e facilita a modelagem da continuidade. No entanto, o software não apresenta toda a gama de funções de continuidade espacial, nem algumas funcionalidades ainda necessárias para uma análise variográfica mais completa. A Figura 3.2 demonstra a variografia como realizada atualmente no SGeMS. Ainda não há ferramentas de suporte para variografia no software.

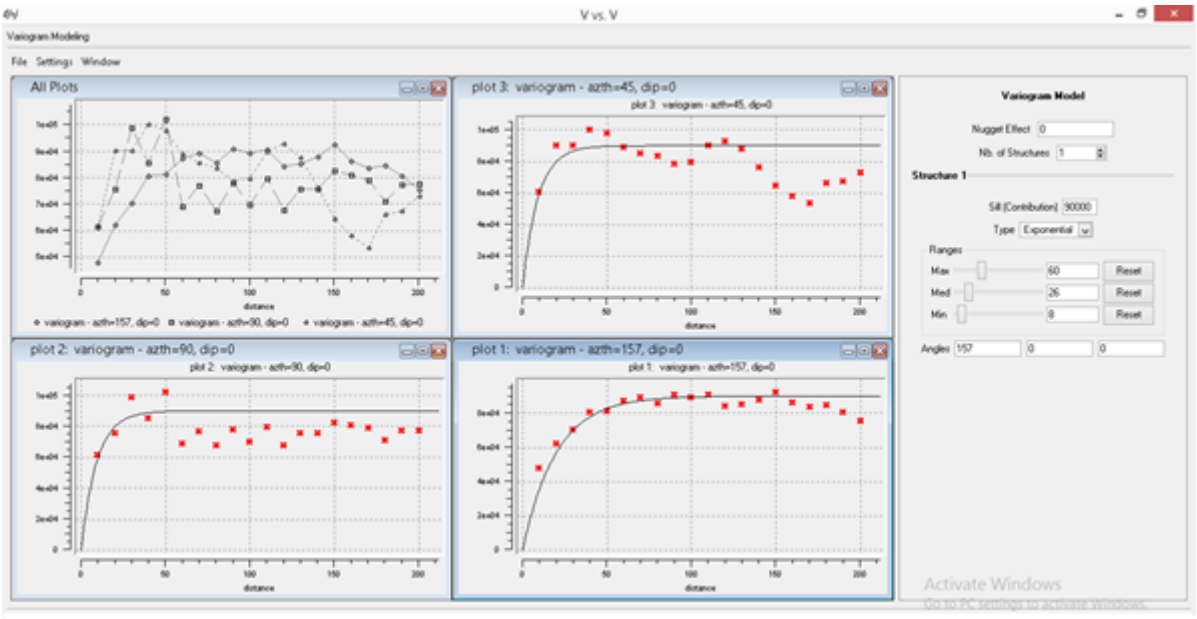

Figura 3.2 – Variografia no SGeMS.

A ausência de um programa que aglomere todas as funções matemáticas e funcionalidades práticas é clara. Existem diversas funcionalidades ou ferramentas que auxiliam na variografia e no reconhecimento da continuidade espacial do fenômeno geológico. A Tabela 3.2 apresenta as ferramentas principais e mais usuais da análise variográfica.

Tabela 3.2 – Ferramentas de auxílio à variografia

Ferramenta	**Objetivo**
Perfis da variável aleatória	Observação da estacionaridade em uma direção
h-scatterplot	Verificação de valores outliers
Estimativas robustas de variograma	Funções de continuidade espacial menos susceptíveis à valores extremos e efeito proporcional
Mapas de variograma	Verificação da continuidade espacial e anisotropia em um plano qualquer do espaço
Modelagem automática	Retirada da interferência humana na modelagem geológica

Todos os métodos possuem objetivos diferentes procurando ajudar ao final na obtenção de modelos mais robustos de continuidade espacial:

1. A ferramenta de h-scatterplot auxilia na verificação de valores extremos das amostras, que resultam em ruídos que interferem no ajuste das funções.

2. As funções de continuidade espacial robustas, como o variograma relativo, também permitem eliminar os efeitos causados pela presença de valores extremos, sendo os resultados menos sensíveis aos dados.

3. Os mapas de variograma permitem a verificação de anisotropias no espaço e auxiliam determinar das direções de menor, intermediária e maior continuidade.

4. A modelagem automática é o recurso utilizado após a obtenção dos valores experimentais, buscando reduzir a interferência pessoal na modelagem.

Procurando unir a funcionalidade dos dois programas, optou-se por desenvolver algoritmos para o software SGeMS que incorporariam as ferramentas de variografia necessárias, unindo a robustez dos algoritmos do software GSLIB e a funcionalidade do software SGeMS.

4 Meta e objetivos

A meta deste trabalho é desenvolver e melhorar os mecanismos de mapeamento e modelagem da continuidade espacial utilizando o software gratuito SGeMS.

O objetivo específico é desenvolver novas metodologias de modelagem da continuidade espacial e incoporá-las no software SGeMS que contenham:

1. mapas de variograma tridimensionais.
2. funções de continuidade espacial robustas.
3. plotagem do h-scatterplot.
4. modelagem automática das funções experimentais.

Alguns destes plug-ins terão como referência a biblioteca de algoritmos do Gslib em Fortran 90 e serão transcritos para linguagem C++ e Python. Após as rotinas serem validadas em um projeto real, serão demonstrados os benefícios da utilização dessas ferramentas.

5 Metodologia

A dissertação será desenvolvida adaptando algoritmos da biblioteca do GSLib 90, escritos em Fortran 90, para a linguagem de programação em C++ e Python utilizada no SGeMS. Por uma questão técnica e operacional haverá modificações nas linhas do código portado para que atenda os requisitos do software SGeMS. Os procedimentos e lógica deverão se manter fiéis à programação inicial.

O primeiro plug-in envolverá a adaptação do programa GamV (DEUTSCH; JOURNEL et al., 1998) para o SGeMS, com a mesma lógica da busca de pares de amostras, com parâmetros de entrada semelhantes e com as mesmas funções de continuidade espacial demonstradas no GSLib. Haverá a possibilidade de exportar os valores das funções para um arquivo xml que poderá ser utilizado na modelagem contida na aba Variogram do software SGeMS

O segundo plug-in permitirá a plotagem dos valores das funções de continuidade espacial no espaço criando um mapa de isovalores das funcões variograma ou covariograma, essenciais para identificar a possível anisotropia do fenômeno. Um terceiro algoritmo será desenvolvido a partir da rotina padrão do GamV para produzir um gráfico h-scatterplot para cada lag do variograma.

O quarto algoritmo propõe o ajuste automático de variogramas por metodologia de mínimos quadrados. O procedimento da escolha dos parâmetros será baseado no método de Monte Carlo, adaptando-o para a condição do SGeMS. O algoritmo também incorporará ajustes automáticos do variograma utilizando um modelo linear de corregionalização. Todos os cálculos do variograma experimental serão validados segundo o programa GSLIB.

6 Oganização da dissertação

A estrutura da dissertação estará organizada em cinco capítulos, sendo que o primeiro se refere ao estado da arte. O segundo é uma revisão bibliográfica que demonstrará os principais desenvolvimentos na variografia (MATHERON, 1963), até os modelos robustos atuais e técnicas de modelagem automática.

O capítulo 3 será a apresentação dos programas desenvolvidos em Python, desde a estrutura da interface dos plug-ins até a sua aplicação, além dos códigos propriamente ditos. Serão validados os plug-ins pelos resultados análogos do software GSLib.

O capítulo 4 consistirá na aplicação das rotinas para um banco de dados tridimensional e verificação das ferramentas em um problema real.

O capítulo 5 apresenta as conclusões do estudo e propõe trabalhos futuros.

Parte II

Referenciais teóricos

7 Definições

Este capítulo apresenta uma breve revisão teórica sobre variografia e os conceitos relevantes para investigar o comportamento espacial do fenômeno geológico. Dentre eles destacam-se as funções experimentais, os modelos de continuidade espacial, a definição de anisotropia, os modelos lineares de corregionalização e as diferentes técnicas utilizadas para o ajuste automático de variogramas. Toda a conceituação teórica envolvida neste capítulo será de relevância para o desenvolvimento dos plug-ins. Ela aborda os principais desenvolvimentos de suas bases (MATHERON, 1963), até as funções robustas (CRESSIE; HAWKINS, 1980) e modelos de ajuste automático (CRESSIE, 1985).

7.1 Definição de continuidade espacial e variografia

A continuidade espacial define o comportamento médio da variável regionalizada para direções determinadas(GOOVAERTS, 1997). Ela é uma propriedade intrínseca do fenômeno estudado e caracteriza sua organização espacial. A variografia utiliza de funções estatísticas para reconhecer a continuidade espacial da variável aleatória. São formulações bi-pontuais que requerem a disposição espacial das amostras, conjugando sempre pares de valores para um dado espaçamento. A Figura 7.1 demonstra o posicionamento de amostras dentro de um domínio D e a diferença vetorial de espaçamento h.

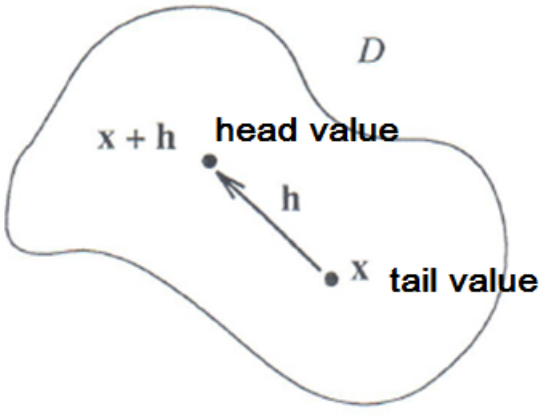

Figura 7.1 – Notação geoestatística para a definição de um lag h em uma direção dentro de um domínio D de uma amostra com suporte x. Dois pares de pontos, um considerado head value, ou ponta do vetor e outro considerado tail value, ou início do vetor. Figura modificada (Wackernagel, 1995).

As funções de continuidade espacial medem comportamentos diferentes da variável

aleatória: ou elas determinam a similaridade dos valores, ou determinam suas dissimilaridades (WACKERNAGEL, 2013). Essas propriedades são análogas às projeções vetoriais. A Figura 7.2a demonstra a dissimilaridade como uma diferença de dois vetores, um vetor considerando os dados in situ e outro com os dados deslocados em um direção h. A diferença entre os vetores é analogamente comparada à função variograma em que são calculadas as médias das diferenças entre dados. Quanto maior for a discrepância entre os dados, maior será o vetor da diferença entre as amostras. A Figura 7.2b também demonstra a similaridade dos dados como uma projeção vetorial, em que um conjunto mais similar apresenta maiores projeções. Nesse caso a similaridade analogamente comparada com o covariograma pode ser comparada com uma projeção de um vetor sobre outro ou com o produto escalar entre o vetor Z(x+h) e o vetor Z(x).

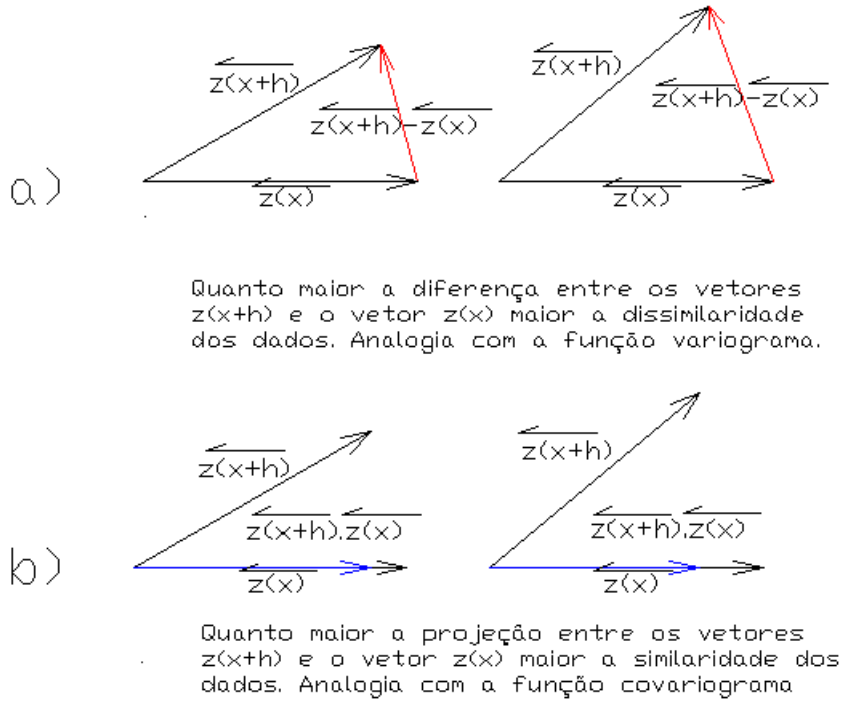

Figura 7.2 – Funções de continuidade espacial como uma interpretação de vetores. a) Variograma como uma diferença de vetores b) Covariograma como uma projeção de vetores.

7.2 Dependência espacial

Para os fenômenos geológicos, é de se esperar que as amostras mais próximas apresentem maior similaridade de valores amostrados (GRINGARTEN; DEUTSCH, 2001). A Figura 7.3 é uma representação gráfica desta característica comentada em um gráfico h-scatter para valores de lag crescentes. Quanto maior for a distância entre as amostras, mais estes pares de valores são dissimilares ou descorrelacionados (ISAAKS et al., 1989).

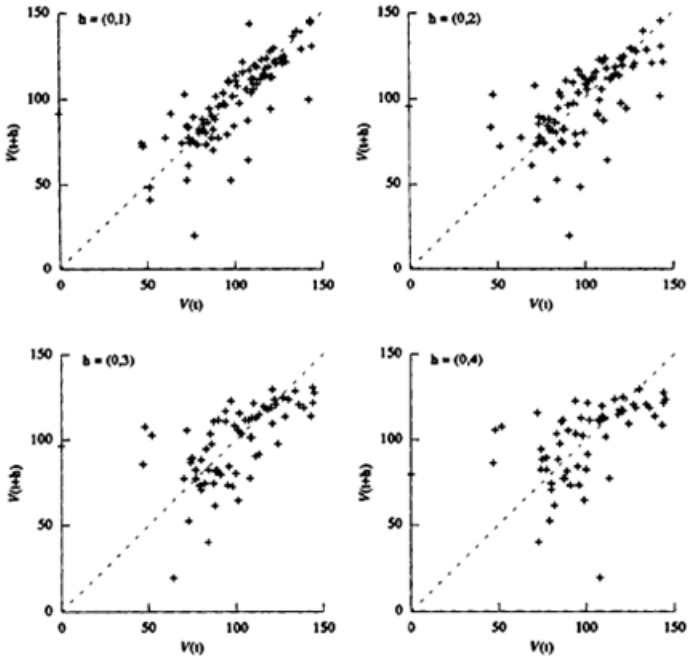

Figura 7.3 – H-scatterplots de uma variável para lags de 0,1m , 0,2m, 0,3m e 0,4m. Aumento da descorrelação de acordo com o aumento do comprimento dos lags. Figura retirada de (ISAAKS et al., 1989).

7.3 Hipótese de estacionaridade

As funções de continuidade espacial requerem que o modelo ajustado não seja afetado pela translação. A hipótese estacionaridade também pode ser denominada de invariância de translação (TAHMASEBI; HEZARKHANI; SAHIMI, 2012). Alguns tipos de funções exigem hipóteses mais fortes, tal como a de estacionaridade segunda ordem. Esta admite que todas as distribuições das variáveis aleatórias no espaço possuam médias e variância iguais. Um exemplo da necessidade da estacionaridade de segunda ordem é a portabilidade das funções variograma e covariograma. As funções variograma e covariograma podem estão relacionadas quando estabelecido um regime estacionário de segunda ordem, resultando na Equação 7.1:

$$\gamma(h) = C(0) - C(h) \tag{7.1}$$

Em que $\gamma(h)$ é o semi-variograma, $C(0)$ é a variância à priori dos dados e $C(h)$ o valor do covariograma. No entanto, a presença de tendência nos dados mostra que o covariograma e o variograma não se convertem em uma relação direta, pois a média da variável aleatória não é constante no domínio e por isso não implica em uma hipótese de estacionaridade de segunda ordem.

8 Funções experimentais de continuidade espacial

8.1 Efeito dos dados sobre os valores experimentais

As funções clássicas de continuidade espacial são afetadas por valores extremos, esparcidade dos dados e valores clusterizados o que levou à investigação de funções de estimativas robustas (CRESSIE; HAWKINS, 1980). Leva-se em consideração que a continuidade espacial é uma propriedade do domínio e não das amostras. No entanto, pela escassez de informação, ela é inferida a partir de uma quantidade limitada de dados. Se as observações não cobrirem as dimensões do objeto de estudo, devido a um número pequeno de amostras com dados esparsos ou agrupados, a estimativa pode não representar a continuidade do fenômeno. Pode-se demonstrar a conexão entre o variograma experimental e a amostragem, tal que as amostras devem respeitar as seguintes definições (DAVID, 1977):

1. As amostras devem estar contidas na mineralização do depósito.
2. Os corpos de minério devem ser tratados de forma diferenciada.
3. Todas as amostras devem ter o mesmo suporte.

Um número crescente de amostragens pode nem sempre resultar em um benefício da informação sobre a continuidade espacial. Como as funções de continuidade espacial são valores médios de uma gama de pares de amostras no espaço, e os valores médios tendem a suavizar o efeito das informações, o acréscimo de pares de informações redundantes pode não alterar as funções de continuidade espacial. O reconhecimento da continuidade exige uma estratégia de amostragem adequada e depende complexidade do depósito mineral. Corpos com baixa continuidade espacial podem ser analisados com malhas adensadas em contrapartida de depósitos com alta continuidade espacial que podem ser analisados com malhas menos adensadas.

8.2 Funções de continuidade espacial mais comuns

(MATHERON, 1963) desenvolveu as principais funções de estimativa da continuidade espacial para entender o comportamento das variáveis aleatórias regionalizadas. Duas destas são consideradas as mais tradicionais definidas inicialmente pelo autor. O

covariograma e o variograma medem respectivamente a similaridade e dissimilaridade dos dados. Define-se a função covariograma pela Equação 8.1:

$$C(h) = E\left[\left(Z_{i+h} - m_{i+h}\right)\left(Z_i - m_i\right)\right] \tag{8.1}$$

Em que Z_i é o valor da variável aleatória no suporte i, Z_{i+h} é o valor da variável aleatória transladada por um vetor h e m_{i+h} o valor médio da variável transladada. Sob a hipótese de estacionaridade de segunda ordem os valores da média são constantes tal que $m_{i+h} = m_i$ e a Equação 8.1 pode ser traduzida pela Equação 8.2 por uma transformação algébrica:

$$C(h) = E\left(Z_{i+h}Z_i\right) - m^2 \tag{8.2}$$

A função variograma pode ser representada pela Equação 8.3:

$$2\gamma(h) = E\left[\left(Z_{i+h} - Z_i\right)^2\right] \tag{8.3}$$

Em que $\gamma(h)$ é também denominado de semi variograma. Na literatura, é comum a utilização ambígua dos termos, referindo-se ao valor de semi variograma como a função variograma. A função semi variograma pode ser representada como uma distância da dispersão de pontos em relação à reta Y=X, em um gráfico h-scatterplot (JOURNEL; HUIJBREGTS, 1978). A Figura 8.1 é uma demonstração gráfica da interpretação do semivariograma como um valor médio das distâncias das variáveis Zu e Zu+h, com esses valores em x e y e com a reta de correlação máxima. A demonstração da Figura 8.1 em termos matemáticos está descrito na Equação 8.4:

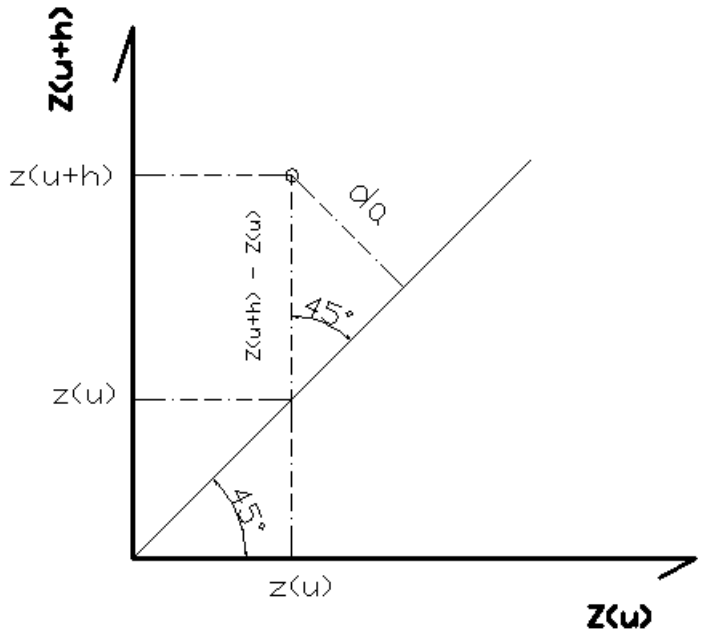

Figura 8.1 – Interpretação geométrica do semi-variograma como sendo a distância de um ponto em relação a reta x=y.

$$\frac{1}{n}\sum_{i=1}^{n} da^2 = \frac{1}{n}\sum_{i=1}^{n} sen^2(45°)\,(z_i - z_{i+h})^2 = \frac{1}{2n}\sum_{i=1}^{n}(z_i - z_{i+h})^2 \tag{8.4}$$

8.3 Outras funções experimentais

Várias são as funções de continuidade espacial utilizadas na bibliografia, principalmente as mais clássicas desenvolvidas por (MATHERON, 1963). No entanto, a busca de estimativas cada vez menos sensíveis aos valores extremos levou ao desenvolvimento de modelos robustos. Entre eles podemos citar o variograma relativo e o pairwise (GOOVAERTS, 1997). Há também uma classe de diferentes tipos de estimativas para a função variograma, utilizando uma série de médias com limites aparados e dentre elas a própria mediana para poucos valores de pares (CRESSIE, 1985). Modelos mais robustos de variograma estão desenvolvidos ao longo da bibliografia (CRESSIE; HAWKINS, 1980). A Equação 8.5 demonstra a relação determinada pelos autores, que garante menores efeitos de valores extremos ao contrário do variograma tradicional proposto por Matheron, também denominada de Rodograma:

$$\gamma(h) = \frac{1}{2n}\sum_{i=1}^{n} |Z_i - Z_{i+h}|^{\frac{1}{2}} \tag{8.5}$$

A Tabela 8.1 é uma representação das principais estimativas de funções de continuidade espacial (GOOVAERTS, 1997). Alguns tipos tem maior recorrência que as demais

como o variograma tradicional e a covariância, propostos inicialmente por Matheron:

Tabela 8.1 – Funções de continuidade espaciais experimentais

Função experimental	**Medida de**	**Equação**
Semi-variograma	dissimilaridade	$\sum_{i=0}^{n} \frac{(Z_i - Z_{i+h})^2}{2n}$
Covariograma	similaridade	$\frac{1}{n} \sum_{i=0}^{n} (Z_i - m_i).(Z_{i+h} - m_{i+h})$
Correlograma	similaridade	$\frac{1}{n} \sum_{i=0}^{n} \frac{(Z_i - m_i).(Z_{i+h} - m_{i+h})}{\sigma_i \sigma_{i+h}}$
Pair-Wise	dissimilaridade	$\frac{1}{n} \sum_{i=0}^{n} \frac{(Z_i - Z_{i+h})^2}{\left(\frac{Z_i + Z_{i+h}}{2}\right)^2}$
Madograma	dissimilaridade	$\frac{1}{n} \sum_{i=0}^{n} \lvert Z_i - Z_{i+h} \rvert$
Variograma Relativo	dissimilaridade	$\frac{1}{n} \sum_{i=0}^{n} \frac{(Z_i - Z_{i+h})^2}{\left(\frac{m_i + m_{i+h}}{2}\right)^2}$

Em que m_i e m_{i+h} são os valores médios determinados no início e na ponta do vetor ($E(Z_i)$ e $E(Z_{i+h})$) e σ_i^2 e σ_{i+h}^2 são as variâncias no início e na ponta do vetor. No caso de estacionaridade de segunda ordem $m_i = m_{i+h}$ e $\sigma_i^2 = \sigma_{i+h}^2$, no entanto as funções experimentais calculam à priori estes valores de acordo com as distribuições amostrais do tail e do head para cada diferença de lag. A obtenção dos valores experimentais das funções de continuidade espacial é a etapa inicial, que é seguida pela modelagem variográfica e pelas etapas posteriores de estimativa e simulações.

O cálculo dos valores experimentais é realizado segundo uma direção vetorial. A Figura 8.2 demonstra os valores de amostras aceitáveis como pares de pontos permissíveis. Para um lag unitário, somente os valores adjacentes no grid estão disponíveis. A direção escolhida para o cálculo do variograma experimental é leste-oeste.

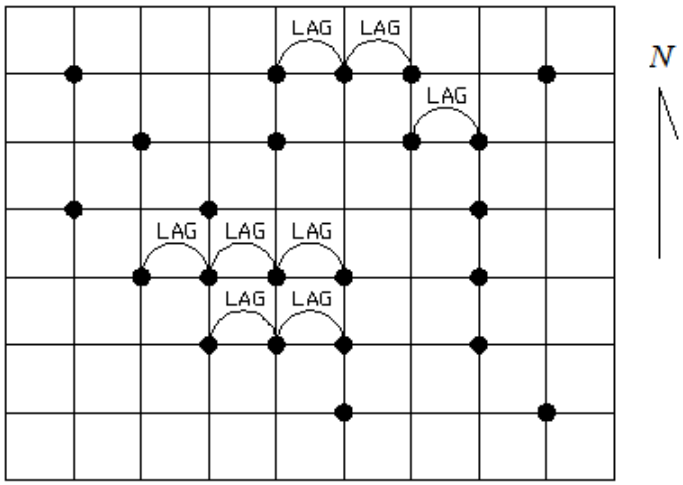

Figura 8.2 – Cálculo de variogramas experimentais segundo um lag unitário na direção Leste-Oeste.

O mesmo pode ser representado na Figura 8.3, em que os valores disponíveis como pares para o cálculo são efetuados em dois nós do grid consecutivos.

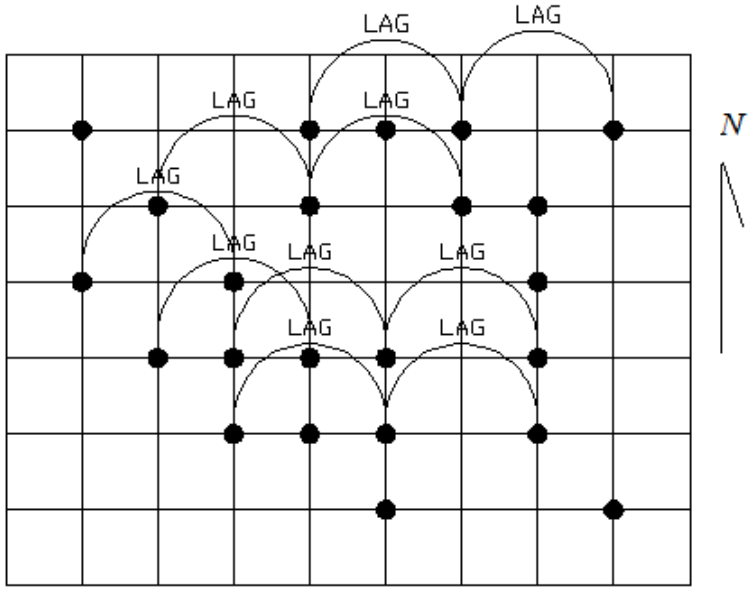

Figura 8.3 – Cálculo de variogramas experimentais segundo o dobro do lag na direção leste-oeste.

Os valores calculados do variograma nas Figuras 8.2 e 8.3 estão demonstrados em um gráfico na Figura 8.4 de forma ilustrativa como dois pontos consecutivos, P1 e P2 ligados por um modelo hipotético como forma ilustrativa. Cada diferença de lag representará um valor de variograma associado. Exemplos do cálculo de variogramas

experimentais podem ser encontrados em diversas bibliografias(WACKERNAGEL, 2013) , (JOURNEL; HUIJBREGTS, 1978), (ISAAKS et al., 1989).

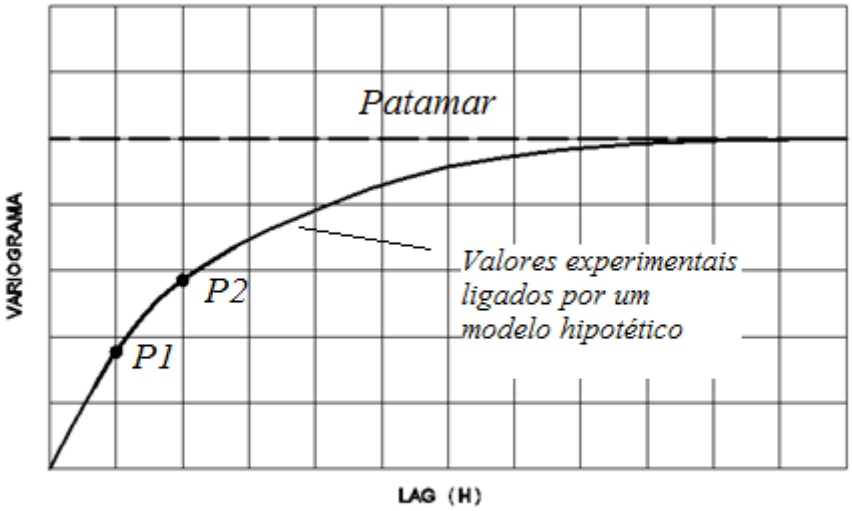

Figura 8.4 – Função variograma experimental para os cálculos nas Figuras 9 e 10 e ajuste em um modelo hipotético. P1 e P2 representam os pontos para um lag unitário e o dobro do lag.

8.4 Parâmetros de busca

Nas Figuras 8.2 e 8.3, a direção escolhida para o cálculo da função variograma permite medidas regulares. No entanto, a maioria dos casos relacionados à mineração é caracterizada por disposições irregulares das amostras (ISAAKS et al., 1989). Neste caso, o variograma direcional não é mais calculado em uma direção absoluta, mas apresenta uma região de incerteza no alinhamento das amostras. A Figura 8.5 é uma representação da busca de pares irregularmente espaçados.

Para o problema bidimensional, são consideradas 3 variáveis geométricas de incerteza e o lag do vetor propriamente dito. As geometrias variáveis são:

1. Tolerância angular = Desvio angular da direção nos lags de menor tamanho.
2. Banda = Desvio lateral da busca.
3. Tolerância linear = Desvio longitudinal da busca.

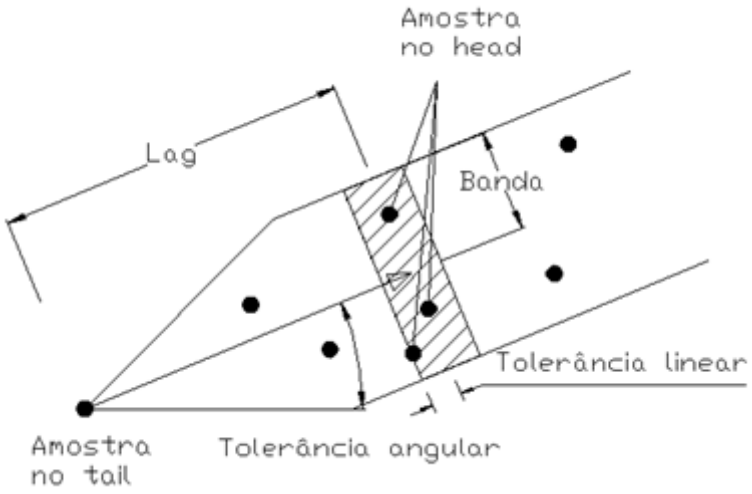

Figura 8.5 – Busca de pontos da função de continuidade para amostras irregularmente espaçadas.

No caso tridimensional, a busca de pares pode se realizar de duas formas diferentes, sob uma perspectiva elíptica ou prismática. A Figura 8.6 é uma representação da busca de pares nas duas formas geométricas.

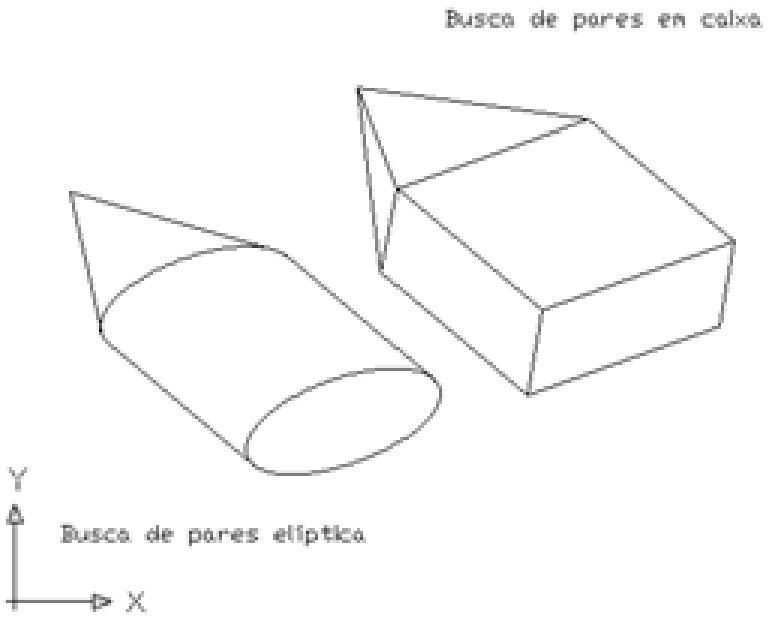

Figura 8.6 – Busca de pares de pontos em uma direção tridimensional. Busca de pares elíptica utilizada no SGeMS e em caixa utilizada no GSLib.

A procura pela metodologia em caixa é utilizada no software Gslib (DEUTSCH; JOURNEL et al., 1998), em que são estipuladas não somente a tolerância horizontal e

a banda horizontal, como também a tolerância vertical e a banda vertical. A alternativa de caixa é preferencial à busca de pontos cilíndrica, pois em casos onde depósitos minerais apresentem estratigrafia característica, as bandas verticais e horizontais podem ser utilizadas para delimitação de amostras pertencentes ao mesmo nível de formação geológica.

9 Modelagem de funções de continuidade espacial

9.1 Modelos de variogramas permissíveis

Após a estimativa dos valores experimentais, a análise variográfica procede com a modelagem de funções permissíveis e estabelecimento de um modelo simplificado de regionalização. Um modelo permissível de variograma deve possuir as seguintes características (SHAPIRO; BOTHA, 1991):

1. O modelo deve ser uma função par $\gamma(h) = \gamma(-h)$.
2. O modelo deve ser uma função positiva definida tal que qualquer combinação linear dos seus valores deve ser maior ou igual a zero, como demonstrado na Equação 9.2.

$$\sum_{i=0}^{n}\sum_{j=0}^{n}\lambda_i\lambda_j\gamma(x_i - x_j) \geq 0 \tag{9.1}$$

   Em que λ_i é uma constante de proporcionalidade e x_i e x_j são as diferenças das amostras em um suporte i e j qualquer.

3. Modelo deve ser limitado por um valor limite, geralmente caracterizado como a variância à priori do fenômeno.

9.2 Parâmetros das funções de continuidade

O conjunto de variogramas transitivos, ou seja, que apresentam um patamar possuem parâmetros característicos. A Figura 9.1 é uma representação de um modelo de variograma. Os parâmetros da função são:

1. Efeito pepita: Caracteriza a dispersão dos valores para um lag imediatamente maior que zero. O Efeito pepita representa, além da variabilidade de escala, os erros associados à amostragem.
2. Range ou alcance: Máxima distância de influência da correlação. A partir do range não mais existe correlação entre os pares de valores da variável aleatória e estes podem ser ditos independentes.
3. Patamar: O patamar representa o estabelecimento da máxima dispersão admissível. Nas funções em que a similaridade é a propriedade caracterizada, o patamar assume

valor nulo tal como na Figura 9.2. A melhor estimativa para o patamar pode não ser a variância das amostras, mas deve-se considerar o posicionamento espacial destas pela utilização da variância de dispersão, da declusterização dos pesos, além do tratamento de valores extremos (GRINGARTEN; DEUTSCH, 2001).

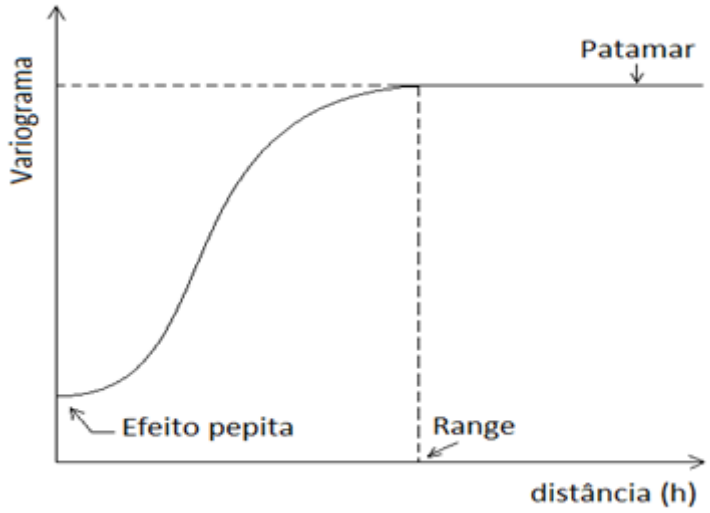

Figura 9.1 – Parâmetros do variograma.

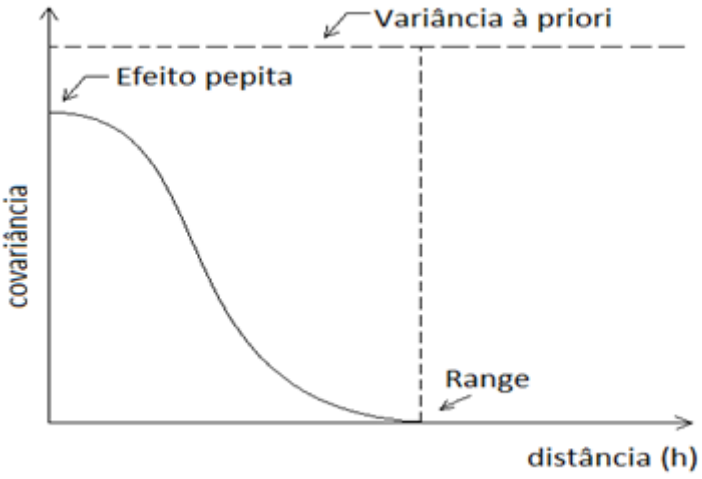

Figura 9.2 – Parâmetros da função covariograma.

9.3 Modelos de continuidade espacial mais comuns

Dentre os modelos de covariância mais comuns podemos citar (Goovaerts, 1997):

Efeito de pepita puro: Considera o efeito de dispersão puro. Não representa nenhuma conectividade dos dados e a probabilidade de ocorrência de um determinado valor é

caracterizada por uma distribuição uniforme. O efeito pepita pode ser caracterizado como uma percepção não linear do fenômeno em uma escala considerada (GRINGARTEN; DEUTSCH, 2001). A Equação 9.2 demonstra o modelo de variograma com efeito de pepita puro.

$$\gamma(h) = \begin{cases} 0 & , h = 0 \\ 1 & , \text{ao contrário} \end{cases} \tag{9.2}$$

Modelo exponencial: O modelo representa o valor de variabilidade com decaimento exponencial. Apresenta-se assíntota no patamar e o range é caracterizado por um valor prático que ocupa 95% da variância a priori quando h = 3a, sendo "a" o alcance prático. A Equação 9.3 demonstra o modelo de variograma exponencial.

$$\gamma(h) = 1 - \exp^{\frac{-h}{a}} \tag{9.3}$$

Modelo Gaussiano: O modelo representa o valor de variabilidade de decrescimento exponencial quadrático. Dentre as funções, é a que apresenta maior suavização próxima da origem. Apresenta também um range prático tal que $h = a\sqrt{3}$. (JOURNEL; HUIJBREGTS, 1978). A Equação 9.4 demonstra o modelo de variograma gaussiano.

$$\gamma(h) = 1 - \exp^{-\frac{h^2}{a^2}} \tag{9.4}$$

Modelo Esférico: A Equação 9.5 é a representação de um modelo esférico. Apesar de constituir uma função de terceira ordem, que feriria os princípios de positiva definida, o modelo esférico é limitado pelo alcance da função, e a partir daquele valor é substituído pelo patamar.

$$\gamma(h) = \begin{cases} \left(\frac{3h}{2a} - \frac{h^3}{2a^3}\right) & , h < a \\ 1 & , h >= a \end{cases} \tag{9.5}$$

Como todos os modelos prescritos são permissíveis então qualquer combinação destes também resulta em um modelo permissível.

9.4 Anisotropia

A anisotropia é a mudança de comportamento das propriedades do variograma por rotação. Os fenômenos geológicos podem permitir a gênese diferenciada dos litotipos à partir de controles e enriquecimentos em sentidos distintos (JOURNEL; HUIJBREGTS, 1978). Dois casos são recorrentes na literatura e envolvem a forma geométrica e zonal.

O caso geométrico delimita alcances diferentes para um mesmo patamar. A anisotropia geométrica pode ser resumida em um modelo de elipsóide em que haverá eixos de máximo, médio e mínimo alcance. A Figura 9.3 é uma representação da anisotropia geométrica.

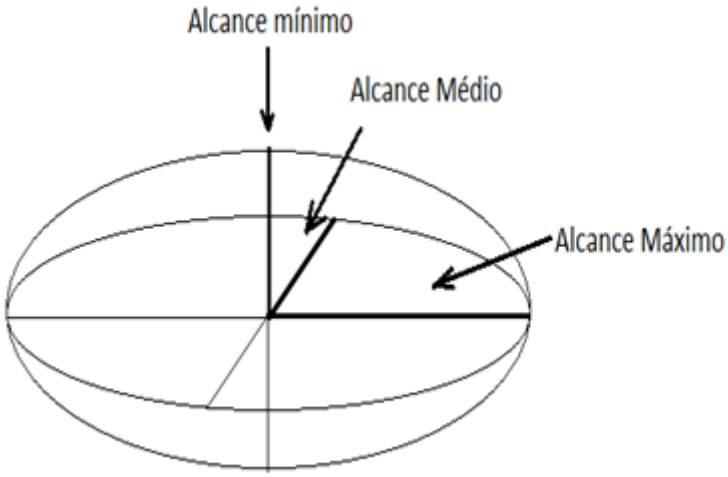

Figura 9.3 – Representação do modelo de anisotropia geométrico. Elipsóide com valores de alcance mínimo médio e alcance máximo.

Os alcances em qualquer direção podem ser derivados de um modelo isotrópico unitário a partir de operações lineares, resultando em um novo sistema de coordenadas (WACKERNAGEL, 2013). A Equação 9.6 representa a matriz de rotação das coordenadas para os eixos de referência.

$$Q = \begin{bmatrix} cos\theta_3 & sin\theta_3 & 0 \\ -sin\theta_3 & cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ cos\theta_2 & sin\theta_2 & 0 \\ -sin\theta_2 & cos\theta_2 & 0 \end{bmatrix} \begin{bmatrix} cos\theta_1 & sin\theta_1 & 0 \\ -sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.6}$$

Em que θ_3 consiste no ângulo de rotação no eixo z, θ_2 o ângulo de rotação no eixo y e θ_1 a rotação do ângulo no eixo x. Os valores do vetor unitário são então redimensionados segundo a matriz de dilatação da Equação 9.7.

$$D = \begin{bmatrix} l_1 & 0 & 0 \\ 0 & l_2 & 0 \\ 0 & 0 & l_3 \end{bmatrix} \tag{9.7}$$

Em que l_1, l_2 e l_3 são os comprimentos dos eixos de máximo, médio e mínimo alcance. Tais matrizes são utilizadas para se construir o modelo do elipsoide de anisotropia e determinar os alcances em qualquer direção possível.

O caso zonal consiste em variações de patamares ao longo de direções diferentes. Em (GOOVAERTS, 1997) demonstra-se o exemplo da anisotropia zonal. Observa-se na Figura 9.4 que a diferença de azimute pelo ângulo θ leva a uma diferença de patamares de g1 para g1 + g2. A anisotropia zonal é característica em alguns tipos de depósitos divididos em estratos, ao qual se verifica diferenças litológicas nas diversas camadas. A variabilidade na direção perpendicular aos estratos tende a ser diferente da direção paralela, que tende a ser mais contínua pelo princípio de sedimentação.

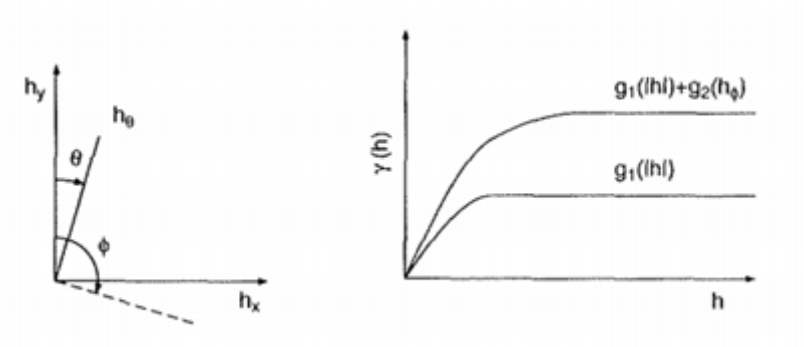

Figura 9.4 – Anisotropia zonal representada nos variogramas a) Diferença entre as direções dos variogramas b) Representação da anisotropia por variogramas com patamares diferentes. Figura modificada(GOOVAERTS, 1997).

A modelagem de anisotropias zonais envolve a utilização de estruturas distintas de variograma que combinadas representarão o conjunto total. A anisotropia zonal também pode ser caracterizada como uma mudança de fenômeno em larga escala (GRINGARTEN; DEUTSCH, 2001).

A anisotropia é uma flexibilização do modelo de variograma para atender às necessidades de depósitos mais complexos, que podem apresentar características diferenciadas segundo diversas direções.

9.5 Funções de continuidade espacial cruzadas

Na modelagem espacial de múltiplas variáveis, tal como os modelos de cokrigagem ou markovianos, é necessário determinar funções de continuidade cruzadas além das diretas. Estas não estão submetidas às mesmas condições de contorno das funções diretas. Primeiramente, porque o valor de patamar de uma estatística cruzada é sempre menor ou igual a das estatísticas diretas e está ligado à correlação entre as variáveis utilizadas para a modelagem.

O covariograma cruzado pode ser representada pela Equação 9.8 em que Zi e Zj

são variáveis distintas e mi e mj são suas respectivas médias:

$$C_{ij}(h) = \frac{1}{2} E\left[\left(Z_i(x+h) - m_i\right)\left(Z_j(x) - m_j\right)\right] \tag{9.8}$$

O covariograma direto é uma função par e unicamente limitada por um valor de patamar. As funções cruzadas, no entanto, podem apresentar efeitos de retardo e não se comportarem como uma função par, tal que $C_{ij}(h) \neq C_{ij}(-h)$, e que i e j são variáveis aleatórias diferentes entre si (WACKERNAGEL, 2013). Segundo (WACKERNAGEL, 2013), toda função pode ser descrita como uma combinação de funções pares e ímpares. A covariância cruzada pode ser decomposta tal como na Equação 9.9:

$$C_{ij}(h) = \frac{1}{2}\left(C_{ij}(h) + C_{ij}(-h)\right) + \frac{1}{2}\left(C_{ij}(h) - C_{ij}(-h)\right) \tag{9.9}$$

Em que $\frac{1}{2}\left(C_{ij}(h) + C_{ij}(-h)\right)$ representa o termo par da função e $\frac{1}{2}\left(C_{ij}(h) - C_{ij}(-h)\right)$ o termo ímpar. Segundo (WACKERNAGEL, 2013), há um sério problema em se definir a matriz de covariâncias, pois geralmente para um dado lag ela não poderá ser considerada nem positiva definida ou negativa definida. Os efeitos produzidos pelo retardo não permitem a utilização de funções assimétricas na resolução dos sistemas de krigagem utilizados a posteriori. Segundo o mesmo autor, a dificuldade de caracterização da covariância no espaço de valores reais leva a utilização em números complexos.

O variograma cruzado, no entanto, não está sujeito aos efeitos do retardo tal como a covariância e apresenta unicamente um termo par. A Equação 9.10 expressa a fórmula da função:

$$\gamma_{ij}(h) = \frac{1}{2} E\left[\left(Z_i(x+h) - Z_i(x)\right)\left(Z_j(x+h) - Z_j(x)\right)\right] \tag{9.10}$$

9.6 Modelo linear de corregionalização

Segundo (GOOVAERTS, 1997), o modelo linear de corregionalização implica que uma variável aleatória deve ser escrita como uma combinação linear de funções aleatórias independentes. Isso significa que para qualquer variável i e j, o modelo estrutural deve ser o mesmo, tal que $C(h) = \sum_{i=1}^{n} b_i \rho(h)$ e que $\rho(h)$ é um modelo único de correlograma e b_i é a contribuição para cada variável considerada. Para ser considerado um modelo permissível, o traço da matriz de covariância deve ser maior que a soma de qualquer coluna ou linha, ou que o determinante deva ser maior ou igual a zero. Os modelos lineares de corregionalização devem satisfazer a condição de matrizes positiva definidas para a resolução dos casos multivariados (LARRONDO; NEUFELD; DEUTSCH, 2003). A dificuldade de se estabelecerem modelos segundo os critérios necessários, levou à simplificações das krigagens colocadas e de modelos Markovianos.

9.7 Modelagem automática de variogramas

Na tentativa de minimizar o trabalho do avaliador na modelagem de funções cada vez mais complexas, a modelagem semiautomática também é uma alternativa para reduzir o erro do ajuste do modelo. Em 1985, já havia se iniciado a tentativa de modelagens automáticas por meio de mínimos quadrados ponderados (CRESSIE, 1985). Em 1988, optou-se por utilizar alternativas não paramétricas no desenvolvimento de variogramas por transformadas de Fourier (GENTON, 1998). A alternativa não paramétrica auxilia na obtenção rápida de mapas de variograma que representam a continuidade em um domínio espacial.

A necessidade de análises rápidas e eficientes aproximou a geoestatística cada vez mais da computação e dos algoritmos numéricos. O Varfit, um programa de uso livre para variogramas automáticos, constitui até hoje uma base de desenvolvimento para os softwares de modelagem automática em geoestatística (LARRONDO; NEUFELD; DEUTSCH, 2003). Houve modificações no programa para atender às necessidades do operador para pontos de âncora no variograma experimental (PARDO-IGÚZQUIZA, 1999). Estes pontos de âncora são valores do variograma experimental que possuem o ajuste coincidente com o seu valor naquele local.

Trabalhos mais atuais demonstram que a geoestatística preocupa cada vez mais em análises rápidas e menos laboriosas, tal como a utilização de variogramas automáticos juntamente com krigagem processada em múltiplos processadores em paralelo (PESQUER; CORTÉS; PONS, 2011). Além disso, há a proposição de algoritmos interativos para a variografia (EMERY, 2010).

O desenvolvimento dos recursos computacionais e de uma teoria mais abrangente do que a inicialmente proposta por (MATHERON, 1963) permitiram o desenvolvimento de estudos em diversas áreas tal como na biologia (BULIT, 2014),metalurgia (SADAWY; ISMAEL; GOUDA, 2015) entre outras áreas tais como também hidrogeologia, engenharia civil e ambiental.

O objetivo da modelagem automática de variogramas é criar um modelo consistente que envolva as principais características do fenômeno descrito, tais como anisotropia e comportamentos próximos da origem, sem a necessidade da interferência manual. A modelagem puramente computacional, sem interferência parcial do operador, leva à criação de continuidades artificiais pouco representativas do fenômeno (LARRONDO; NEUFELD; DEUTSCH, 2003). A proposta semi-automática é então indicada, aos quais os eixos de maior, menor e média continuidade são definidos primordialmente.

Duas vertentes dos processos de otimização são descritas na bibliografia e se dividem em uma abordagem paramétrica e uma abordagem não paramétrica. Na primeira alternativa, propõem-se a otimização de funções já conhecidas e permissíveis, em contrapartida

da segunda aos quais o ajuste é numérico e não é estipulada uma função propriamente dita.

As propostas desenvolvidas a partir da década de setenta constam desde a metodologia de mínimos quadrados (DAVID, 1977), pela utilização de valores ponderados (CRESSIE, 1985) , ou por métodos que envolvam hipótese de multi-gaussianidade (GENTON, 1998). Na sua grande maioria, os métodos de modelagem automática são definidos pelo modelo que levar ao menor desvio médio quadrático.

Há a necessidade da utilização de ponderadores para os diversos pontos experimentais para o ajuste de variogramas, à medida que para distâncias mais curtas é necessário um melhor ajuste (PARDO-IGÚZQUIZA, 1999). As alternativas propostas indicam a utilização do número de pares da estatística, o inverso da distância e do valor do variograma experimental como medidas de ajuste. Mesmo definindo pesos para os valores experimentais, a modelagem automática ainda pode requerer intervenção do operador.

Em todos os modelos de otimização do ajuste de variogramas, é necessário construir uma função objetivo que é responsável pela aproximação dos valores estimados e dos experimentais (CRESSIE, 1985). Geralmente, procura-se otimizar a dissimilaridade entre os valores conjugados. A Equação 9.11 demonstra a relação de dissimilaridade entre o modelo e os variogramas experimentais:

$$\psi = \sum_{i=0}^{n} \rho_i(\gamma_i - \gamma_i^*) \tag{9.11}$$

Em que ψ é a equação objetivo, γ_i são os valores experimentais e γ_i^* são os valores de um modelo a ser ajustado, para uma função de ajuste ρ.

Parte III

Desenvolvimento

10 Busca de pares de pontos

O desenvolvimento da parte experimental consiste na programação e validação de três plugins para auxílio à análise da continuidade espacial, todos desenvolvidos em linguagem python e adaptados segundo o software gamV do GSLIB. Adicionalmente, são propostos algoritmos para modelagem automática de variogramas diretos e cruzados utilizando modelo linear de corregionalização. Os plug-ins são:

1. Programa para o cálculo das funções de continuidade espacial em várias direções.
2. Modelagem semi-automática de variogramas.
3. Mapa de variogramas e covariogramas.
4. H-scatterplots em uma direção.

A busca de pares de pontos admissíveis é o primeiro passo para o cálculo das funções de continuidade espacial. O algoritmo utilizado foi o mesmo do programa GamV do GSLIB . Duas etapas são preponderantes na busca de pares, a primeira que calcula as distâncias euclidianas e a segunda que testa os valores das separações segundo critérios geométricos.

A Figura 10.1 demonstra como é feita a aceitação geométrica de par de amostras no espaço P1 e P2. A distância XY representa a projeção da distância dos pontos amostrais no plano XY e Proj é a projeção da distância dos pontos sobre o vetor da direção do azimute. O ângulo entre o eixo X com a projeção também é chamado de azimute matemático. X e Y são as diferenças entre as posições X e Y dos dois pontos. Se o cosseno da projeção sobre a direção do azimute for maior que o cosseno da tolerância angular o ponto é dito como aceitável sobre o critério da direção horizontal. A Equação 10.1 demonstra o critério matemático de aceitação do par de amostras como permissível utilizando seus parâmetros geométricos. Não somente são testadas as tolerâncias angulares, mas também as bandas verticais e horizontais. Além disso, o GSLIB prescreve uma distância máxima da busca de pares, que pode ser avaliada como multiplicação do número de lags pelo tamanho do lag. Qualquer par de pontos que se situar fora destes limites é descartado do conjunto.

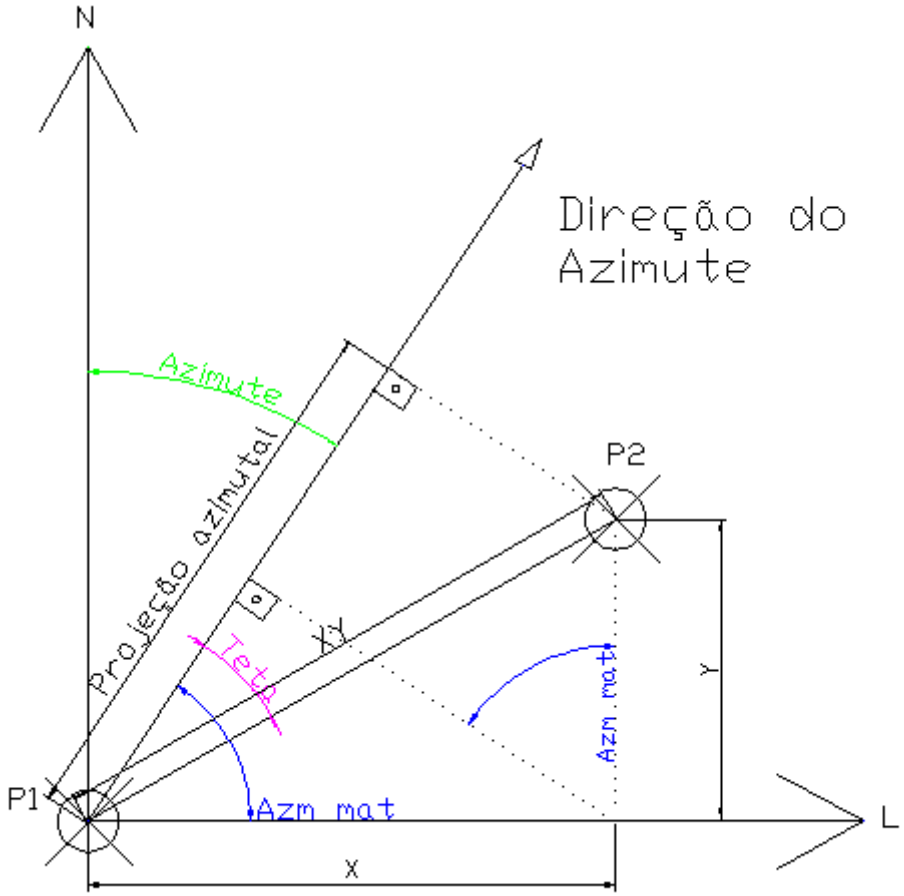

Figura 10.1 – Teste geométrico do par de amostras segundo o critério de aceitação do azimute. P1 e P2 são os pares de amostras separados por uma projeção XY no plano horizontal. X e Y são as diferenças entre cotas e Projeção azimutal é a projeção dos pontos amostrais na direção do azimute.

$$\begin{aligned} &cos(\theta) > cos(\text{tol angular}) \\ &Proj/XY > cos(\text{tol angular}) \\ &y.sen(Azimute_mat) + x.cos(Azimute_mat)/XY > cos(\text{tol angular}) \end{aligned} \tag{10.1}$$

A estratégia da busca de pares do programa GamV é modular para aumentar a velocidade de processamento dos cálculos. A Figura 10.2 demonstra o trecho do código para o cálculo das distâncias entre todos os pontos. Os testes geométricos não são realizados em conjunto com o cálculo dos variogramas para não aumentar a complexidade do algoritmo. Existem alternativas de códigos mais simples para o reconhecimento dos pares de amostras permissíveis, no entanto, não possuem os mesmos critérios geométricos do software GamV.

```
# DEFINA TODAS AS DISTANCIAS EUCLIDIANAS POSSIVEIS

cabeca = []
rabo = []
distanciah =[]
distanciaxy =[]
distanciax =[]
distanciay = []
distanciaz = []
for t in range(0, len(yt)):
    for h in range(t, len(yh)):
        cabeca.append(vh[h])
        rabo.append(vt[t])
        dx = xh[h]-xt[t]
        dy= yh[h]-yt[t]
        dz = zh[h]-zt[t]
        if (distanciah > 0):
            distanciay.append(dy)
            distanciax.append(dx)
            distanciaz.append(dz)
            distanciah.append(math.sqrt(math.pow(dy,2) + math.pow(dx,2) + math.pow(dz,2)))
            distanciaxy.append(math.sqrt(math.pow(dy,2) + math.pow(dx,2)))
```

Figura 10.2 – Algoritmo das distâncias euclidianas em Python para aceitação do par permissível.

A Figura 10.3 demonstra o trecho da busca de pares utilizado no programa Varmap. No cálculo dos mapas de variograma, os valores são iterados não apenas para uma direção, mas para 18 direções com variações de 20 em 20°. A adaptação do algoritmo do GSLib para aplicações no mapa aumentou a ordem da complexidade do algoritmo.

```
# CALCULE OS PONTOS ADMISSIVEIS
for a in range(0,36):
    azm = math.radians(a*10)
    if (dip != 90 and dip != 180):
        if (a != 0 and a != 180):
            dipatua = math.radians(90 - math.atan(math.tan(math.radians(dip))/math.sin(math.radians(a))))
        else:
            dipatua = 0
    else:
        dipatua = math.radians(90)
    for l in range(0, nlags):
        valores_admissiveis_h = []
        valores_admissiveis_t = []
        distancia_admissivel = []
        azimute_admissivel =[]
        dip_admissivel=[]
        lag= lagdistance*(l+1)
        par= 0
        for p in range(0,len(distanciah)):
            if (distanciah[p] < dmaximo):
                limitemin = lag - lineartolerance
                limitemax = lag + lineartolerance
                if (distanciah[p] > limitemin and distanciah[p] < limitemax):
                    if (distanciaxy[p] > 0.000):
                        check_azimute = (distanciax[p]*cos_Azimute[a] +distanciay[p]*sin_Azimute[a])/distanciaxy[p]
                    else:
                        check_azimute = 1
                    check_azimute = math.fabs(check_azimute)
                    if (check_azimute >= htolerance):
                        check_bandh = (cos_Azimute[a]*distanciay[p]) - (sin_Azimute[a]*distanciax[p])
                        check_bandh = math.fabs(check_bandh)
                        if (check_bandh < hband):
                            if(distanciah[p] > 0.000):
                                check_dip = (math.fabs(distanciaxy[p])*sin_Dip[a] + distanciaz[p]*cos_Dip[a])/distanciah[p]
                            else:
                                check_dip = 0.000
                            check_dip = math.fabs(check_dip)
                            if (check_dip >= vtolerance):
                                check_bandv = sin_Dip[a]*distanciaz[p] - cos_Dip[a]*math.fabs(distanciaxy[p])
                                check_bandv = math.fabs(check_bandv)
                                if (check_bandv < vband):
                                    valores_admissiveis_h.append(cabeca[p])
                                    valores_admissiveis_t.append(rabo[p])
                                    distancia_admissivel.append(distanciah[p])
                                    par = par + 1
                                    azimute_admissivel.append(azm)
                                    dip_admissivel.append(dipatua)
        if (len(valores_admissiveis_h) > 0 and len(valores_admissiveis_t) > 0):
            v_valores_admissiveis_h.append(valores_admissiveis_h)
            v_valores_admissiveis_t.append(valores_admissiveis_t)
            distancias_admissiveis.append(distancia_admissivel)
            pares.append(par)
            azimute_admissiveis.append(azimute_admissivel)
            dip_admissiveis.append(dip_admissivel)
```

Figura 10.3 – Algoritmo da busca de pares.

A Figura 10.4 é uma representação do algoritmo em pseudocódigo.

Procedimento da estratégia de busca de pares
Defina para cada direção

1. Distância máxima $= \left((\text{number of lags} + 0.5) * \text{lag}\right)^2$
2. $\text{Mat}_{\text{azimth}} = (90 - \text{azimth}) * \frac{\text{pi}}{180}$
3. $\text{Mat}_{\text{dip}} = (90 - \text{dip}) * \frac{\text{pi}}{180}$

Defina para cada par de ponto

4. $\Delta x = x_{\text{cabeça}} - x_{\text{rabo}}$
5. $\Delta y = y_{\text{cabeça}} - y_{\text{rabo}}$
6. $\Delta z = z_{\text{cabeça}} - z_{\text{rabo}}$
7. $h = \Delta x^2 + \Delta y^2 + \Delta z^2$
8. $\Delta xy = \sqrt{\Delta x^2 + \Delta y^2}$
9. Recuse o ponto se estiver fora da distância máxima
   $h >$ Distância máxima
10. $h \leftarrow \sqrt{h}$
11. Recuse o ponto se estiver fora dos limites do lag
    $(h \geq \text{lag} * n_0\text{lag} - \text{tolerância linear E } h \leq \text{lag} * n_o\text{lag} + \text{tolerância linear})$
12. Critério da banda horizontal $= \cos(\text{Mat}_{\text{azimth}})\Delta y - \sin(\text{Mat}_{\text{azimth}})\Delta x$
13. Recuse o ponto se estiver fora do critério da banda horizontal
    Band h criteria $> bandh$
14. Projeção do azimute $= \dfrac{(\Delta xy.\cos(\text{Mat}_{\text{azimth}}) + \Delta y.\sin\ \sin(\text{Mat}_{\text{azimth}}))}{\Delta xy}$
15. Recuse os pontos se estiverem fora dos limites da direção
    Projeção do azimute $< \cos(\text{Mat}_{\text{azimth}})$
16. Se a projeção do azimute < 0 , então $\Delta xy \leftarrow -\Delta xy$
17. Projeção do mergulho $= \dfrac{(\Delta xy.\cos(\text{Mat}_{\text{dip}}) + \Delta z.\sin\ \sin(\text{Mat}_{\text{dip}}))}{h}$
18. Recuse os pontos se estiverem fora dos limites do mergulho
    Projeção do mergulho $< \cos(\text{Mat}_{\text{dip}})$
19. Critério da banda vertical $= \cos(\text{Mat}_{\text{dip}})\Delta z - \sin(\text{Mat}_{\text{dip}})\Delta xy$
20. Recuse o ponto se estiver fora do critério da banda vertical
    Critério da banda vertical $> bandv$
21. Troque os valores do tail e head se as projeções do mergulho e da direção forem < 0
22. Guarde os pares se forem aceitos

Fim para
Fim para

Figura 10.4 – Algoritmo da busca de pares em pseudocódigo.

Em seguida, o GSLIB seleciona a função de interesse para cálculo da continuidade espacial e calcula individualmente as estruturas segundo os parâmetros de entrada. A busca de pares é o primeiro procedimento que deve estar adequado com a situação prescrita nas linguagens de programação em Fortran 90 e no phyton. Ao definir a lógica da estratégia de busca, todos os procedimentos de cálculo para h-scatterplots e mapas de variogramas tornam-se possíveis. Cada software pode escolher algoritmos de busca com características diferenciadas. A metodologia prescrita no software SGeMS utiliza uma busca de amostras cilíndrica, que consiste em comprimentos de bandas iguais horizontais e verticais. O programa GSLIB utiliza tolerâncias horizontais e verticais para cada direção selecionada e assim permite que as estatísticas sejam calculadas com melhor definição de direção pelo usuário. Isso facilita ao usuário calcular variogramas para estratos ou regiões definidas.

11 Algoritmos

11.1 Algoritmo do hscatterplot

Este plug-in utiliza da busca de pares para separar os pares permissíveis para cada lag considerado e plotá-los em gráficos de dispersão, calcular a regressão linear dos valores e demonstrar o nível de correlação entre os pares de amostras. O gráfico de h-scatterplot facilita a identificação da presença de valores discrepantes, sendo representados por pares de pontos que fogem de uma correlação linear das amostras. A Figura 11.1a , 11.1b e 11.1c demonstram os gráficos de saída do plug-in para os lags de 1500, 3000 e 6000m.

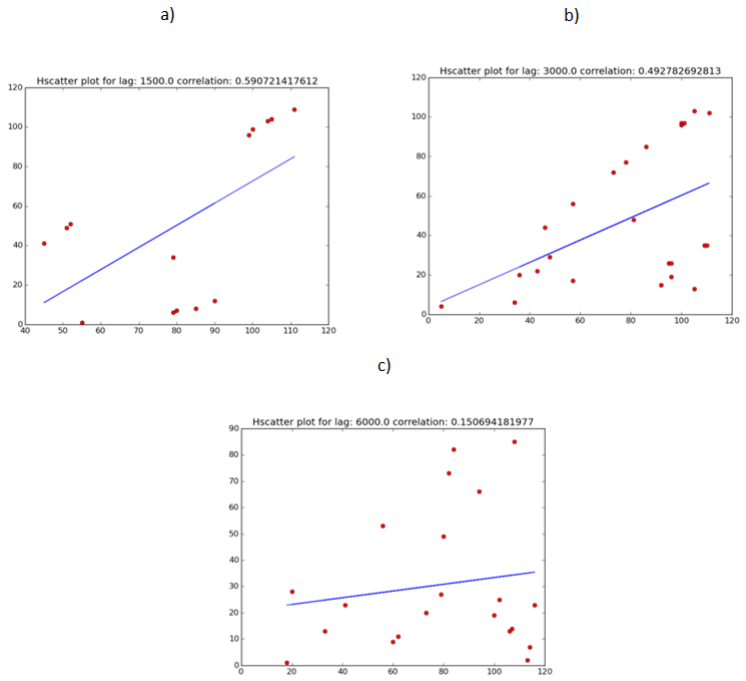

Figura 11.1 – Gráficos de hscatterplot para uma direção específica a) H-scatterplot para um lag de 1500m com correlação 0,59 b) H-scatterplot para um lag de 3000m com correlação 0,49 c) H-scatterplot para um lag de 6000m com correlação 0,15.

A Figura 11.2 demonstra a interface do usuário para o plug-in do h-scatterplot:

1. Caixa das propriedades da ponta do vetor: Consiste na propriedade que será inserida como valor de busca na extremidade da ponta do vetor.

2. Caixa das propriedades do início do vetor: Consiste na propriedade que será inserida como valor de busca no rabo do vetor.

3. Direção do vetor: Direção onde serão calculados os hscatterplots para o variograma considerado. Essa direção é dada como a rotação do azimute em relação ao eixo Z e a rotação do mergulho em relação ao eixo X.

4. Parâmetros do variograma: São todos os parâmetros que definem a busca do variograma experimental, ao qual os valores da ponta e do início do vetor serão plotados em um gráfico e realizada uma regressão linear. Entre eles temos o número de lags, o comprimento do lag, a tolerância linear, as tolerâncias angulares vertical e horizontal e os comprimentos das bandas.

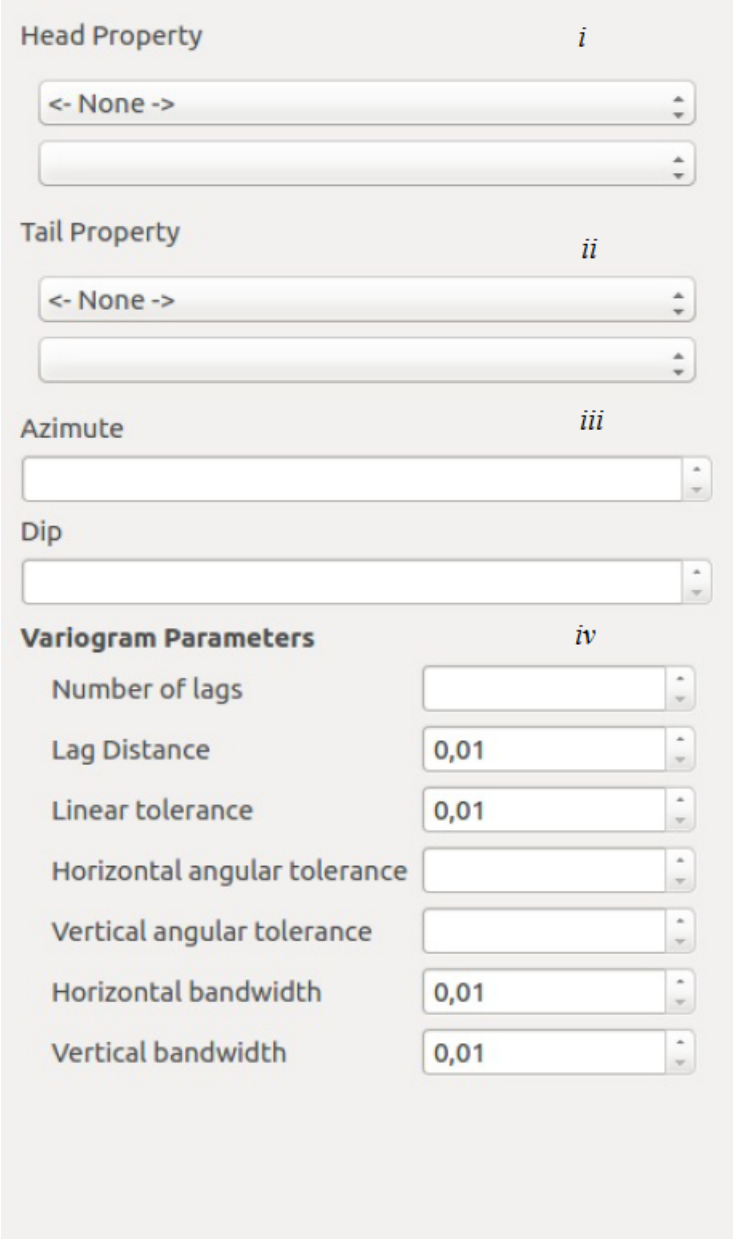

Figura 11.2 – Interface do usuário do programa hscatter. I – caixa de definição das propriedades da ponta do vetor, II – caixa de definição das propriedades do início do vetor, III – Caixa da direção dos h-scatterplots, IV- Parâmetros de busca do variograma.

11.2 Algoritmo do mapa de variogramas

Este plug-in utiliza a lógica da busca de pares e calcula o valor do variograma ou do covariograma, gerando um mapa dessas medidas de continuidade espacial para uma direção do plano no espaço. O algoritmo rotaciona os valores dos dados para que coincidam com o plano do mapa de variogramas e realiza os cálculos necessários. Para isso convenciona-se que a direção do azimute é horária em relação ao eixo Z e do mergulho com rotação horária do eixo X. A Figura 11.3 demonstra o trecho do código com o respectivo

cálculo dos variogramas. O valor de saída é uma lista que contém os valores admissíveis para o lag médio, o azimute médio, o mergulho médio de todos os pares de pontos da estatística e o valor da função de continuidade espacial:

1. Definição das propriedades da ponta do vetor: Consiste na variável que será inserida como valor de busca na extremidade da pontado vetor.

2. Definição das propriedades do início do vetor: Consiste na variável que será inserida como valor de busca no início do vetor.

3. Seleção da função espacial: Os mapas podem ser realizados tanto para variogramas ou covariâncias.

4. Seleção da direção do mapa de variogramas. A referência do norte do mapa é modificada pela variação do azimute no sentido horário do eixo Z, enquanto a declividade em relação ao eixo X é modificada pelo mergulho, no sentido descendente.

5. Parâmetros do variograma: São todos os parâmetros que definem o variograma experimental ao qual o mapa irá interpolar e plotar. O algoritmo calcula valores experimentais do variograma em 18 direções e esses valores são interpolados e plotados em mapa. Entre eles temos o número de lags, o comprimento do lag, a tolerância linear, as tolerâncias angulares vertical e horizontal, além dos comprimentos das bandas.

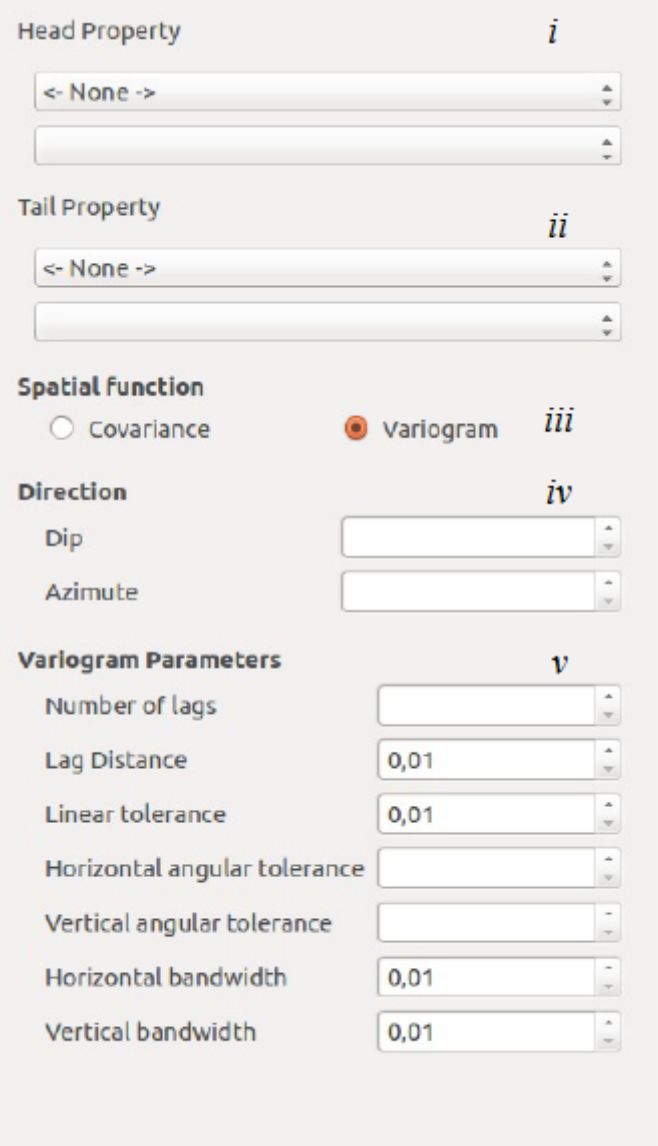

Figura 11.3 – Parâmetros do mapa de variogramas. I – Caixa para definição da propriedade da cabeça do vetor, II- Caixa para definição de propriedades do início do vetor, III – Seleção da função para medida da continuidade espacial, IV – Direção do plano para o mapa de variogramas, orientação azimutal em relação ao norte V- Parâmetros do variograma experimental.

O arquivo de saída do programa é um mapa de variogramas como demostrado na Figura 11.4. O algoritmo depois de calcular os valores do variograma no espaço, interpola-os segundo uma função linear, gerando um mapa de valores no plano. Neste caso, as variações próximas do espectro azul correspondem a menores valores do variograma e as variações próximas do espectro vermelho correspondem a maiores valores do variograma.

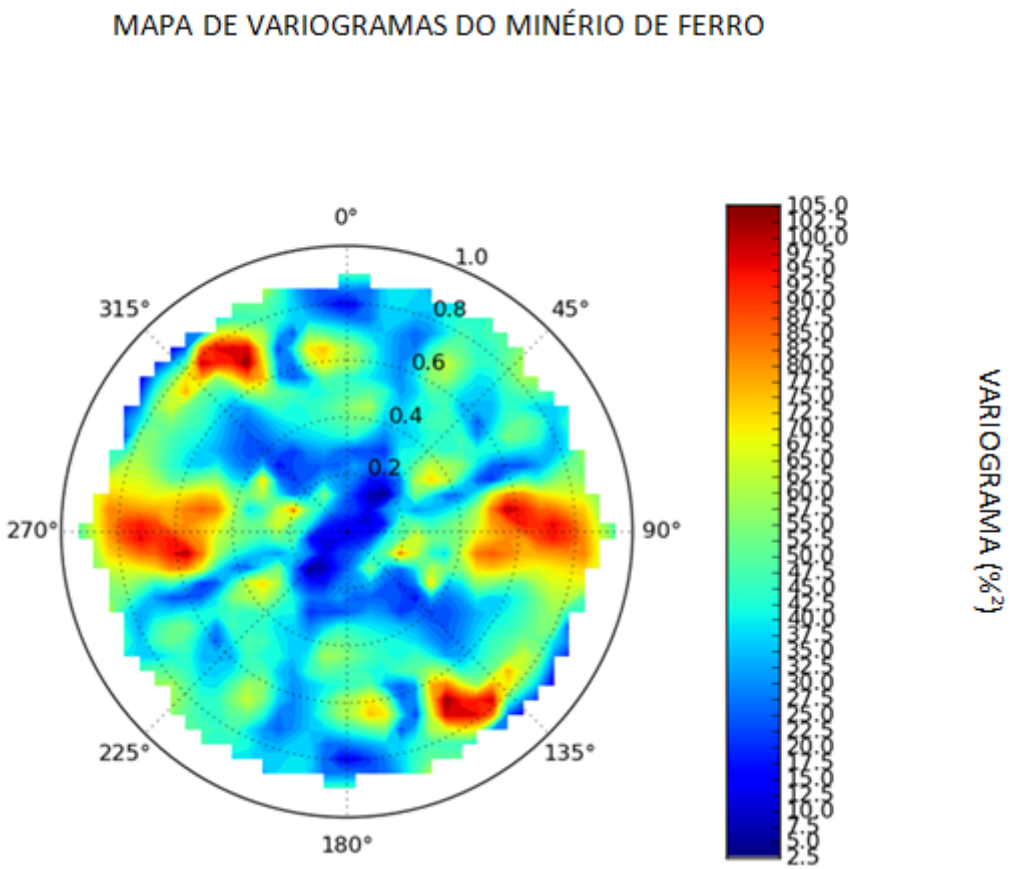

Figura 11.4 – Output do plug-in de mapas de variogramas. Mapa criado a partir de um corpo geológico de minério de ferro plotado no plano horizontal. Maiores valores de variograma no vermelho e menores no azul. Direção de maior continuidade em 52°NE.

11.3 Algoritmo de variogramas experimentais

O próximo plug-in permite calcular diversos tipos funções de continuidade espacial e que após serem calculadas podem ser importadas para modelagem do Sgems. Além disso, há uma opção de criar um arquivo de saída para ser utilizado no ajuste automático de variogramas com um modelo linear de regionalização de acordo com o que será explanado no próximo tópico. Cada vez que é gerada uma nova realização do variograma, o arquivo de report anterior é aberto e é acrescentada a função desejada. Para isso, são atribuídos índices, tal que índices iguais representam um variograma direto e índices diferentes representam os variogramas cruzados. Os valores de variograma são acumulados em cada inicialização do programa no mesmo arquivo. Para constituir em um modelo linear de corregionalização completo, o operador deve proceder o cálculo de todos os variogramas diretos e cruzados do problema.

A Figura 11.5 demonstra o arquivo de saída do report. As duas primeiras linhas

constituem as variáveis de índice na matriz de variogramas (no caso variável 1 com 1 – variograma direto). A terceira linha é um cabeçalho com os valores do variograma contendo o azimute, o mergulho, o lag, o valor do variograma e o número de pares utilizado para o cálculo.

```
1
1
Azimuth Dip lag variogram   pares
157.0   0.0 11.391987159    0.406900936635  257
157.0   0.0 21.5148161393   0.281069643454  227
157.0   0.0 31.4843132561   0.161386042993  161
157.0   0.0 41.5164631  0.201853601426  151
157.0   0.0 50.6695707744   0.0541976578609 153
157.0   0.0 59.8895084996   0.0305555607533 164
157.0   0.0 69.4118267982   0.0909284025784 145
157.0   0.0 78.6585911481   0.265390149326  127
157.0   0.0 88.2302294486   0.138487511141  115
157.0   0.0 98.5100862706   -0.0855394842491    106

```

Figura 11.5 – Demonstração do arquivo de report do plug-in.

A Figura 11.6 mostra o arquivo de saída do tipo SGeMS para os valores experimentais gerados pelo plug-in. Além do arquivo de report, o plug-in permite o cálculo das funções de continuidade espacial que podem ser importados em um arquivo do tipo xml para o SGeMS. Os valores das funções podem então ser visualizados e modelados da forma tradicional pelo programa.

```
<experimental_variograms>
  <variogram>
    <title>variogram - azth=0, dip=0</title>
    <direction>6.12323e-017 1 0 </direction>
    <x>10 20 30 40 50 60 70 80 90 100 110 120 130 </x>
    <y>47269.6 58396.9 76246 76557.6 87014.7 85063.8 94962 84258.7 91352.9 88263.9 91812.1 88913.1 95804.9  </y>
    <pairs>378 606 569 755 771 1057 948 1167 945 1112 967 1082 973 </pairs>
  </variogram>
  <variogram>
    <title>variogram - azth=45, dip=0</title>
    <direction>0.707107 0.707107 0 </direction>
    <x>10 20 30 40 50 60 70 80 90 100 110 120 130 </x>
    <y>61165 90325.2 90205.1 100184 97945.8 89053.7 85506.5 83763.8 78781.6 79745.6 89975.6 93041.4 87755.7 </y>
    <pairs>347 520 605 725 695 651 875 687 849 822 711 689 796  </pairs>
  </variogram>
</experimental_variograms>
```

Figura 11.6 – Demonstração do arquivo de saída xml do plug-in.

Os parâmetros do plug-in podem ser representados na Figura 11.7:

1. Caixa de seleção das propriedades da ponta do vetor do variograma.
2. Caixa de seleção das propriedades do início do vetor do variograma.
3. Endereço de salvamento do arquivo de report: algoritmo gera um arquivo de report para leitura rápida dos valores experimentais no formato txt. O endereço do arquivo

deve ser informado na forma de string. O número associado ao tail e ao head são de auxílio no programa de modelagem automática descrito posteriormente.

4. Endereço de salvamento do arquivo de saída SGeMS: O plug-in gera um arquivo salvo do tipo de importação pelo modeller do SGeMS no formato txt. O endereço do arquivo deve ser informado em string da mesma forma que no item anterior.

5. Seleção da função de continuidade experimental desejada: A função desejada deve ser escolhida dentro de uma série definida à partir de um pushbutton. Apenas uma função deve ser informada de cada vez. As funções permissíveis são o variograma, o variograma relativo, o madograma, o correlograma, o covariograma e o variograma par-a-par relativo.

6. Caixa de direções: Na caixa de direções, são informadas todas as direções desejadas para o cálculo dos variogramas experimentais. Os variogramas são variados em suas direções segundo uma função linear ao qual se tem o primeiro valor, o ângulo da variação e o número de variações:

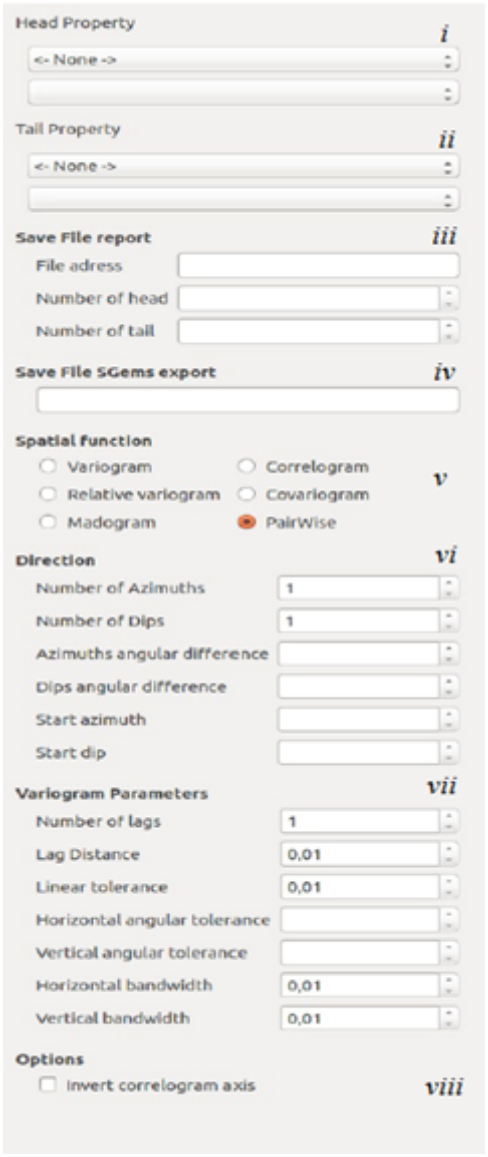

Figura 11.7 – Parâmetros do plug-in de variogramas experimentais. I – Caixa para definição das propriedades da ponta do vetor, II – Caixa para definição das propriedades do início do vetor, III- Caixa de salvamento do arquivo de report a ser utilizado no plug-in de ajuste automático, IV- Caixa de salvamento do arquivo de relatório do variograma, V- Caixa de seleção da função de continuidade espacial , VI- Caixa para definir as direções de cálculo do variograma , VII - Caixa de parâmetros do variograma experimental, VIII -Caixa de inversão do eixo do correlograma.

Uma variação dos cálculos de variogramas experimentais para um plano está demonstrada na Figura 11.8. Nesta versão define-se a direção da reta de máximo declive em um dado plano e o número de variogramas a ser calculado nas direções daquele plano.

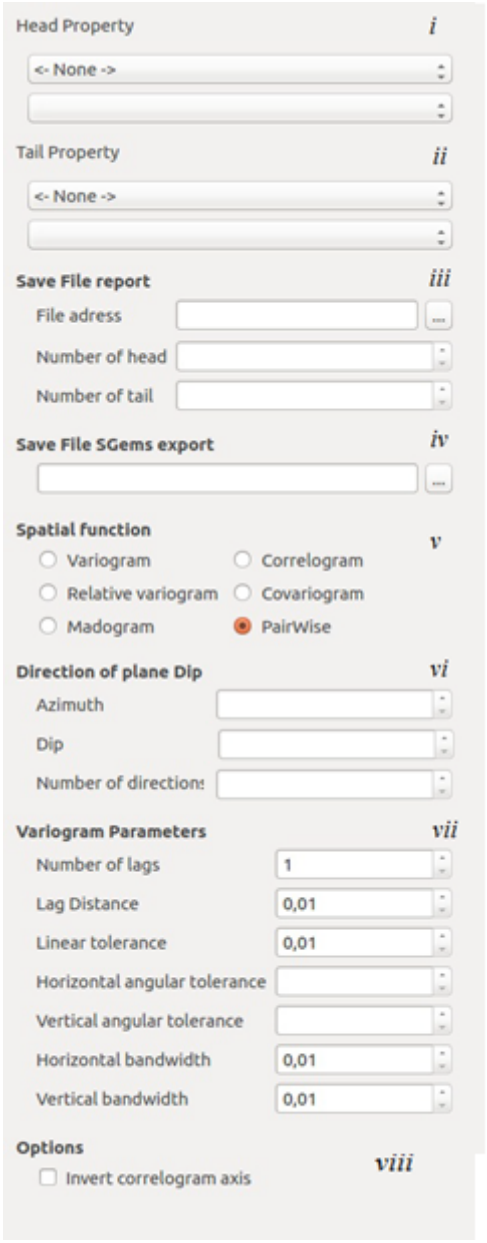

Figura 11.8 – Parâmetros do plug-in de variogramas experimentais. I – Caixa para definição das propriedades da ponta do vetor, II – Caixa para definição das propriedades do início do vetor, III- Caixa de salvamento do arquivo de report a ser utilizado no plug-in de ajuste automático, IV- Caixa de salvamento do arquivo de relatório do variograma, V- Caixa de seleção da função de continuidade espacial , VI- Caixa para definir as direções de cálculo do variograma no plano. Define-se o azimute, o dip da reta de máxima continuidade do plano e o número de direções. , VII - Caixa de parâmetros do variograma experimental, VIII -Caixa de inversão do eixo do correlograma.

11.4 Algoritmo para modelagem automática de variogramas

Para a utilização do plug-in de modelagem automática, é necessária um pré-processamento dos dados no plug-in de variogramas do tópico anterior, que gera um

relatório com os dados para o ajuste automático. O arquivo de entrada do programa deve ser do tipo formulário como demonstrado na Figura 11.9.

```
1
1
 Azimuth    Dip lag variogram   pairs
52.0    52.0    58.1545705606   9.24916637421   6
52.0    52.0    116.57300937    21.2179040047   513
52.0    52.0    145.511340995   29.554942594    1319
52.0    52.0    197.025336662   35.7603360267   521
52.0    52.0    256.302051659   50.1442324991   228
52.0    52.0    296.095852557   34.6298718414   391
52.0    52.0    343.511222479   48.8499528159   112
52.0    52.0    391.829489694   31.8268508629   27
52.0    52.0    451.015852412   36.0015474806   29
52.0    52.0    494.202696064   6.06060007324   25
142.0   0.0 26.9986893685   16.0213040128   46
142.0   0.0 52.4256353591   20.997214893    88
142.0   0.0 77.4004372674   26.61251121 221
142.0   0.0 100.385019609   27.2642445317   1038
142.0   0.0 123.674675849   27.7086215269   156
142.0   0.0 143.618582892   21.7021290419   54
142.0   0.0 176.775139503   19.0453477993   43
142.0   0.0 193.450466109   55.0124522048   51
142.0   0.0 225.774319131   72.8389951986   15
142.0   0.0 244.069037124   32.7760599957   33
52.0    142.0   97.4838048941   28.7166246092   311
52.0    142.0   133.044757122   36.1656242728   3067
52.0    142.0   218.894304164   35.3455600894   2537
52.0    142.0   273.036004923   39.4407207622   2711
52.0    142.0   354.154508621   36.8044364919   2044
```

Figura 11.9 – Arquivo de entrada do otimizador do ajuste variográfico.

O relatório contém os índices que indicam se os variogramas são diretos ou cruzados, além dos valores médios dos lags e do número de pares que são utilizados como ponderadores no ajuste automático. Os variogramas são importados e ajustados segundo os mínimos quadrados ponderados. A Equação 11.1 demonstra a equação objetivo utilizada no plug-in:

$$\psi = \sum_{i=0}^{n} \delta_d^i \delta_p^i \left(\gamma_i - \gamma_i^*\right)^2 \tag{11.1}$$

Em que γ_i são os valores dos variogramas experimentais, γ_i^* são os valores do modelo ajustado e δ_d^i e δ_p^i são os pesos do ajuste para os comprimentos dos lags e do número de pares. Cada peso do comprimento dos lags pode ser descrito na Equação 11.2

$$\delta_d^i = \frac{\frac{1}{lag_i}}{\sum_{i=0}^{i=n} \frac{1}{lag_i}} \tag{11.2}$$

Em que lagi é o valor do lag para cada valor do variograma experimental. Cada peso do número de pares do variograma experimental pode ser descrito pela Equação 11.3

$$\delta_p^i = \frac{n_i}{\sum_{i=0}^{i=n} n_i} \tag{11.3}$$

Em que n_i é o número de pares para cada valor do variograma experimental. O programa desenvolvido utiliza a metodologia de Monte Carlo, isto significa que escolherá aleatoriamente um parâmetro para ser mudado, alterando-o para mais ou para menos em uma porcentagem fixa. O usuário poderá escolher um número de estruturas para o ajuste, mas os tipos de modelos e seus parâmetros são sorteados aleatoriamente para minimizar o resíduo total. Os procedimentos do programa são:

1. Inicie o procedimento com a direção principal.
2. Determine o modelo inicial:
   a) efeito pepita zero.
   b) range inicial igual a metade os valores mínimos e máximos dos lags do variograma experimental.
   c) modelo de variograma igual ao esférico.
   d) contribuição iguais de cada estrutura cuja a soma é o valor médio dos variogramas experimentais.
3. Calcule o resíduo do variograma.
4. A cada interação sorteie um parâmetro do variograma para modificação. Ou selecione o modelo de variograma, ou o alcance de cada estrutura, ou as contribuições de cada uma das estruturas, ou o efeito pepita.
5. O range do variograma é modificado em $\pm 7,5\%$ do valor inicial da estrutura, assim como a contribuição de cada estrutura e do efeito pepita. O modelo de variograma é selecionado aleatoriamente entre 3 modelos sendo o índice 0 para o variograma esférico, 1 para o variograma exponencial e 2 para o variograma gaussiano.
6. Calcule o resíduo com o parâmetro modificado.
7. Se o resíduo modificado for menor que o resíduo inicial, substitua os parâmetros anteriores, sendo os novos, agora, o valor inicial do modelo.
8. Ao final das interações, teste se o ajuste é um modelo linear de corregionalização montando a matriz de contribuições, a matriz dos efeitos pepitas e calcule se o determinante é maior que zero. Se não for um modelo linear de corregionalização, inicie o procedimento de novo no máximo de 100 vezes. Indique se o modelo ajustado foi um modelo linear de corregionalização.
9. O range de cada direção é calculado como o range médio de todas as variáveis ajustadas, com seus variogramas diretos e cruzados.
10. Fixe as contribuições e efeito pepita para as direções secundária e vertical.

11. Determine o range nas direções secundária e vertical.

O aumento do número de interações facilita o algoritmo convergir para o resultado do ajuste. No caso de mais de uma estrutura variográfica, o algoritmo realiza a soma total dos desvios quadráticos do erro de cada estrutura. No final, o programa lança ao usuário na área de comando do SGeMS, o valor dos parâmetros do variograma ajustados e os gráficos dos variogramas para as direções de máxima, mínima e vertical.

A Figura 11.11 demonstra os inputs do programa segundo a caixa de diálogo:

1. Caixa de inserção do endereço do arquivo SGeMS. Aqui deve ser digitado o endereço do arquivo do tipo hml para a importação dos valores dos variogramas experimentais. O endereço do arquivo deve ser exatamente àquele requisitado, podendo variar de tipo de acordo com o sistema operacional.

2. Na caixa do número de interações, indica o número máximo de interações do algoritmo até obter o melhor ajuste possível.

3. Nesta caixa, estão colocados os números de direções dos variogramas experimentais no ARQUIVO. Mesmo que o processo de ajuste apenas necessite de três, coloque o número de TODAS as direções dos variogramas contidos neste arquivo.

4. Nesta caixa, são inseridos o número de variáveis do projeto, isso significa que devem ser colocados o número de variáveis diretas + variáveis cruzadas do problema.

5. Delimite o número de estruturas a serem modeladas no programa. Não adicione muitas estruturas, pois o processamento pode ser demorado e não haver convergência para a solução.

6. Nesta caixa, coloque as direções do variograma associadas a máxima direção, mínima direção e a direção vertical. Os pontos experimentais devem coincidir com os valores estipulados nesta caixa. O ajuste do variograma é apenas feito se estes valores forem contidos no arquivo. A Figura 11.10 demonstra os eixos da continuidade espacial. A rotação dos eixos de continuidade espacial são realizados sempre no sentido horário olhando para a origem dos eixos. O mergulho é positivo como na convenção geológica.

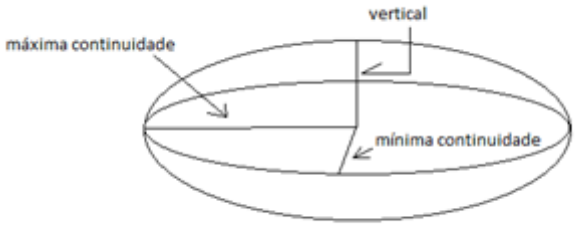

Figura 11.10 – Eixos da continuidade espacial.

7. Nesta caixa, coloque as restrições utilizadas para o ajuste do variograma, que implicam em uma contribuição máxima de sill e um efeito pepita mínimo.

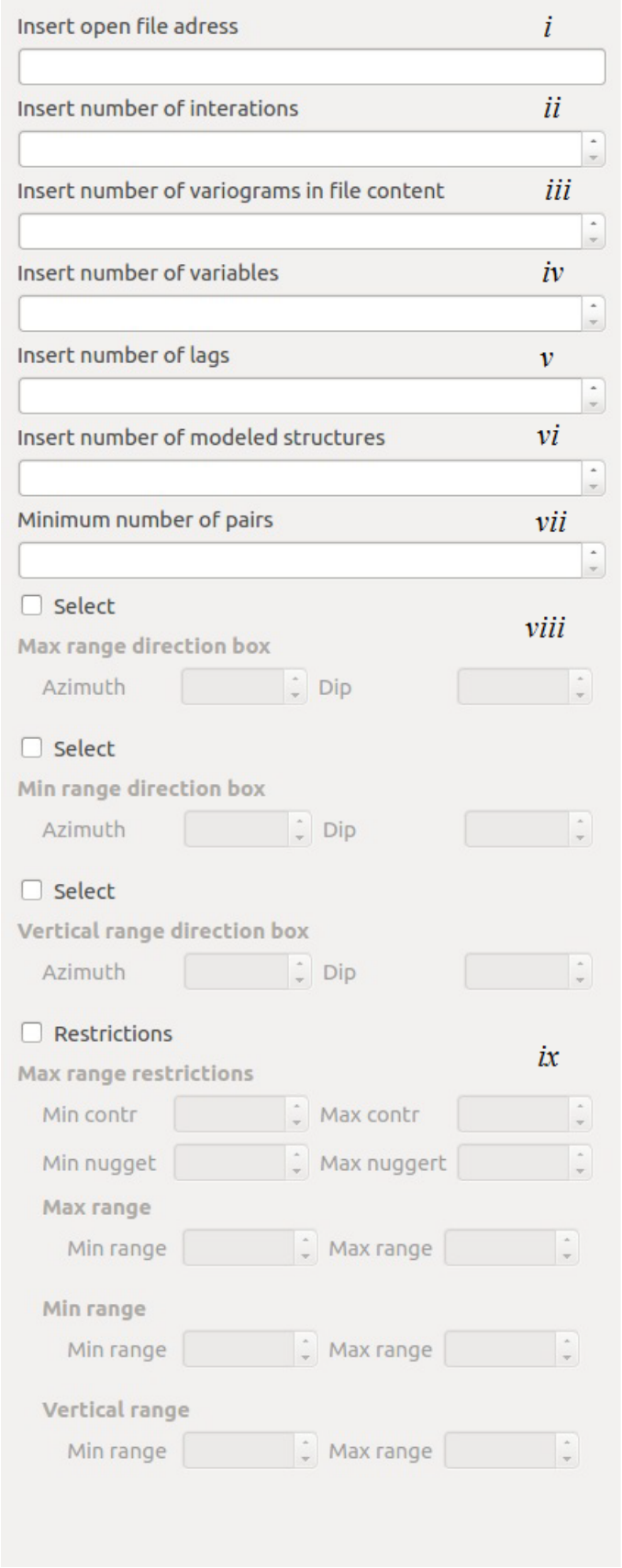

Figura 11.11 – Parâmetros do otimizador da modelagem. I- Endereço do arquivo de input , II – Inserção do número de interações, III- Inserir o número de variogramas no arquivo, IV – Inserir o número de variáveis, V- Inserir o número de lags, VI-Inserir o número de estruturas a serem modeladas, VII- Inserir o mínimo número de pares para um dado lag, VIII- Inserir as direções de ajuste da modelagem , IX – Inserir as restrições do modelo.

12 Validação dos algoritmos de variogramas

12.1 Validação dos variogramas experimentais

Os plug-ins foram validados a partir de comparações entre os valores calculados com o plug-in construído e os valores de saída do GamV. O banco de dados utilizado foi uma partição do Walker Lake (Isaaks & Srivastava, 1989). O banco de dados foi obtido medindo-se a elevação do terreno em uma área de Nevada nos Estados Unidos.

Os parâmetros do variograma experimental utilizados foram 10 lags, de comprimento 10m, tolerância linear de 5m, tolerâncias angulares de 22° e bandas horizontal e vertical de 5m. A validação da busca de pares está demonstrado na Figura 12.1

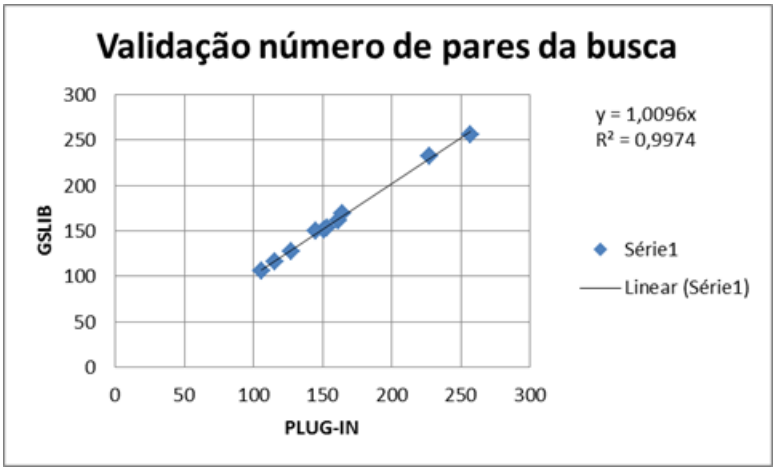

Figura 12.1 – Validação do número de pares do variograma experimental. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLib na direção de 157°N.

Nota-se que a busca de pares de pontos usados no cálculo do variograma não coincide em 100% dos valores. Desta forma, os valores das funções de continuidade espacial também são ligeiramente diferentes entre os plug-ins e o GSLib como demonstrado nas Figuras 34 a 40, pois não há concordância exata dos algoritmos. Ambos resultados serão bem correlacionados, não apresentando mudanças extremas.

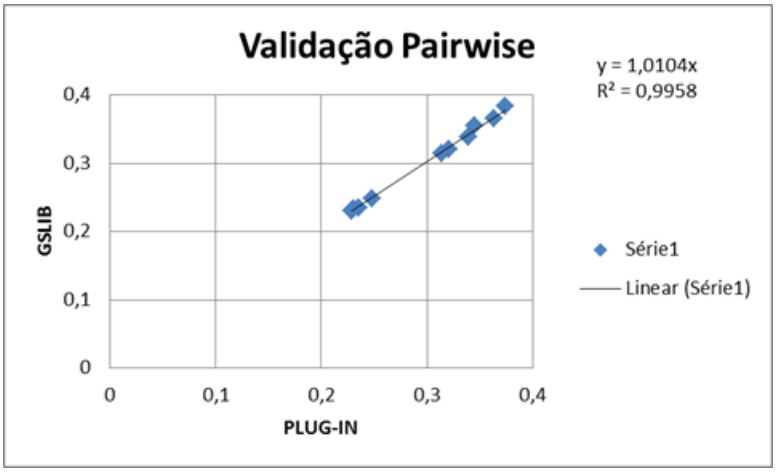

Figura 12.2 – Validação Pairwise. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLib na direção de 157°N.

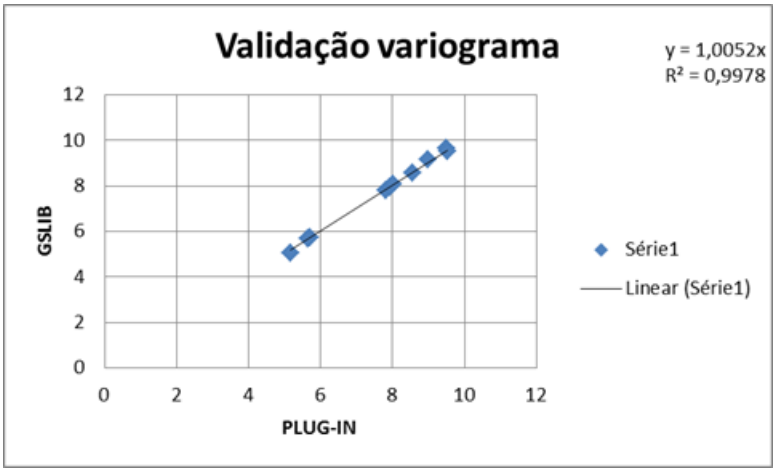

Figura 12.3 – Validação variograma. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N.

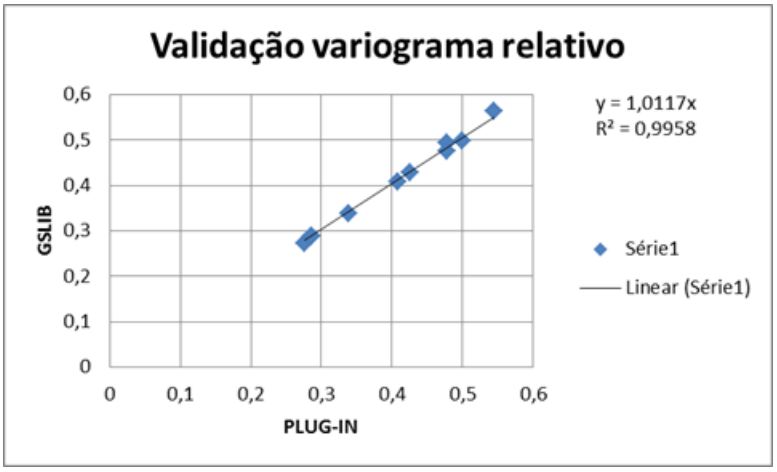

Figura 12.4 – Validação variograma relativo. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N.

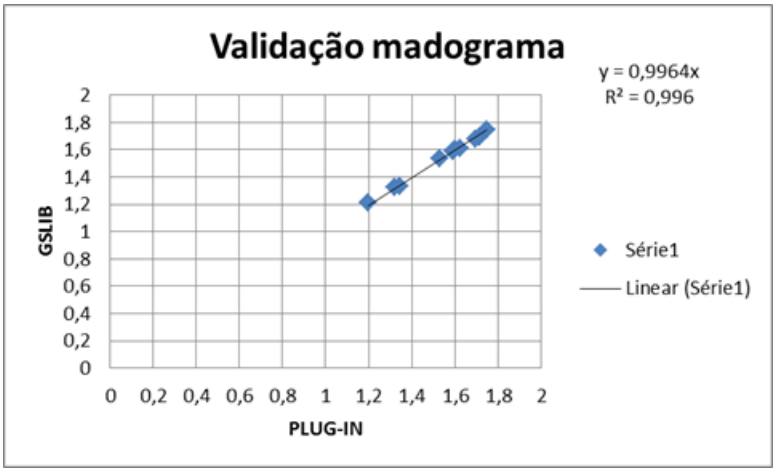

Figura 12.5 – Validação madograma. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N.

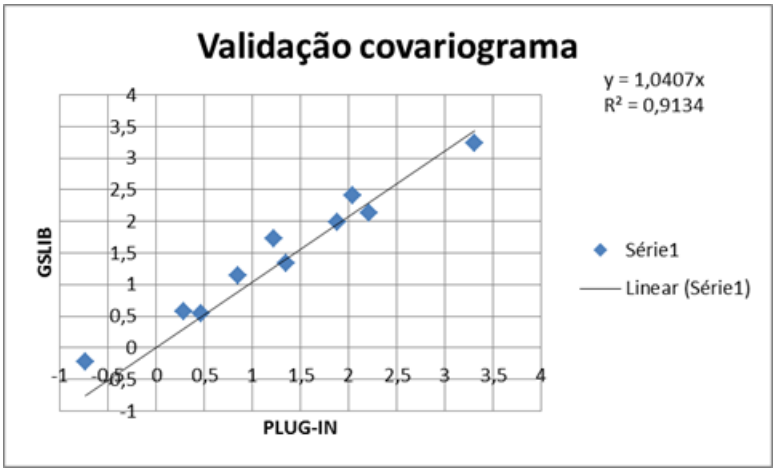

Figura 12.6 – Validação covariograma. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N.

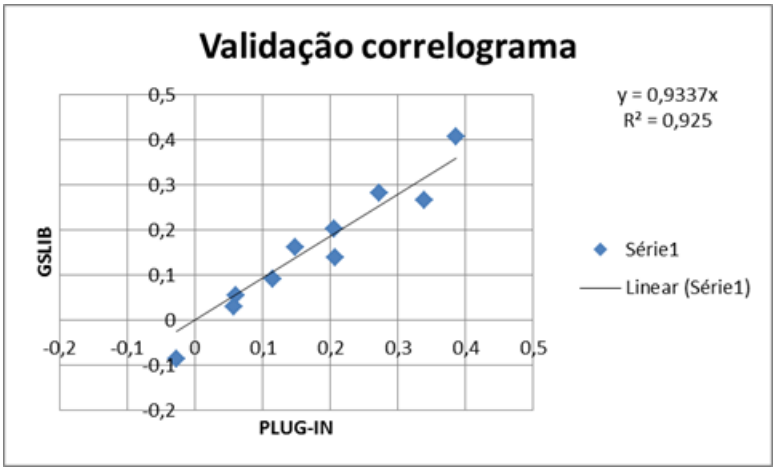

Figura 12.7 – Validação correlograma. Valores de saída do Walker Lake para o Plug-in desenvolvido e para o GSLIB na direção de 157°N.

12.2 Validação do plug-in do mapa de variogramas

Para validar o plugin de mapas de variograma, foi criada uma simulação gaussiana não condicional com modelo de anisotropia característico na direção N45, 45NE , com um modelo gaussiano e com um grid demonstrado na Figura 12.8, 12.9 e 12.10. O grid formado foi composto de 8x8x8 blocos, cada um com tamanho regular unitário 1x1x1 preservando uma quantidade limite de dados a serem incorporados na análise. O mapa de variograma foi criado no plano vertical que contém a anisotropia de N45, 45NE imposta no banco de dados. A Figura 12.11 demonstra a direção de máxima continuidade como imputado os valores no bloco.

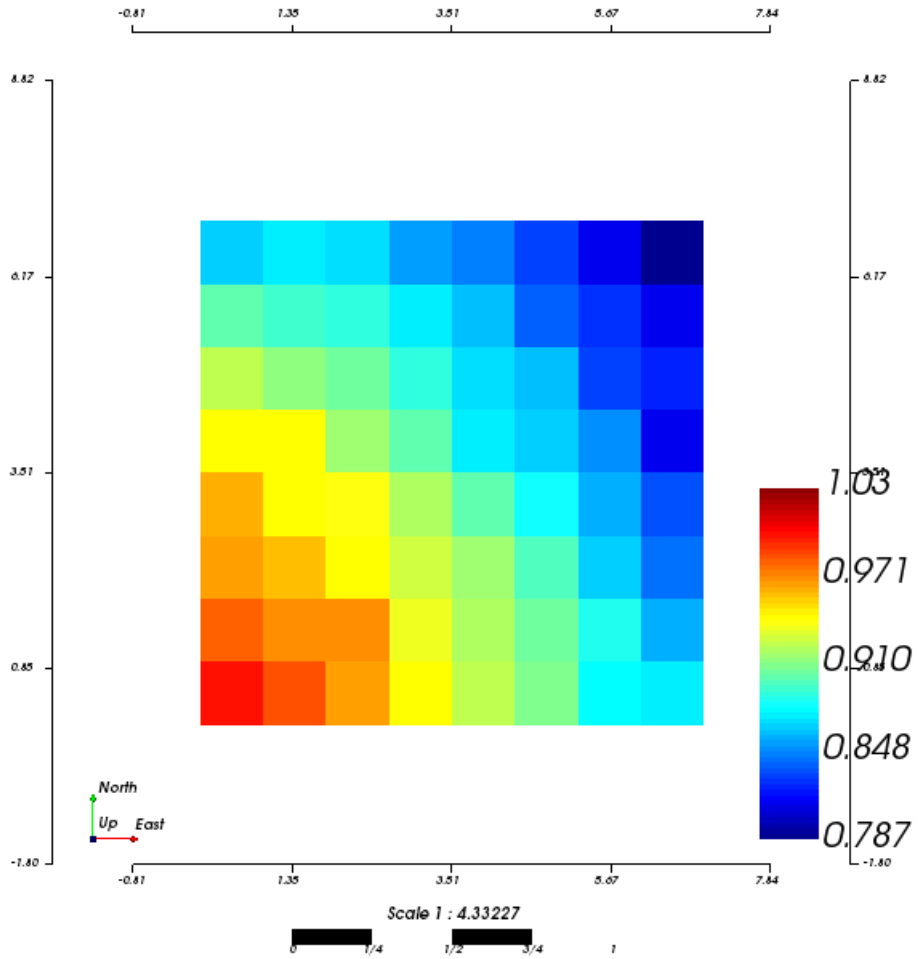

Figura 12.8 – Visão em planta de uma simulação gaussiana com máxima continuidade na direção N45, mergulho 45NE.

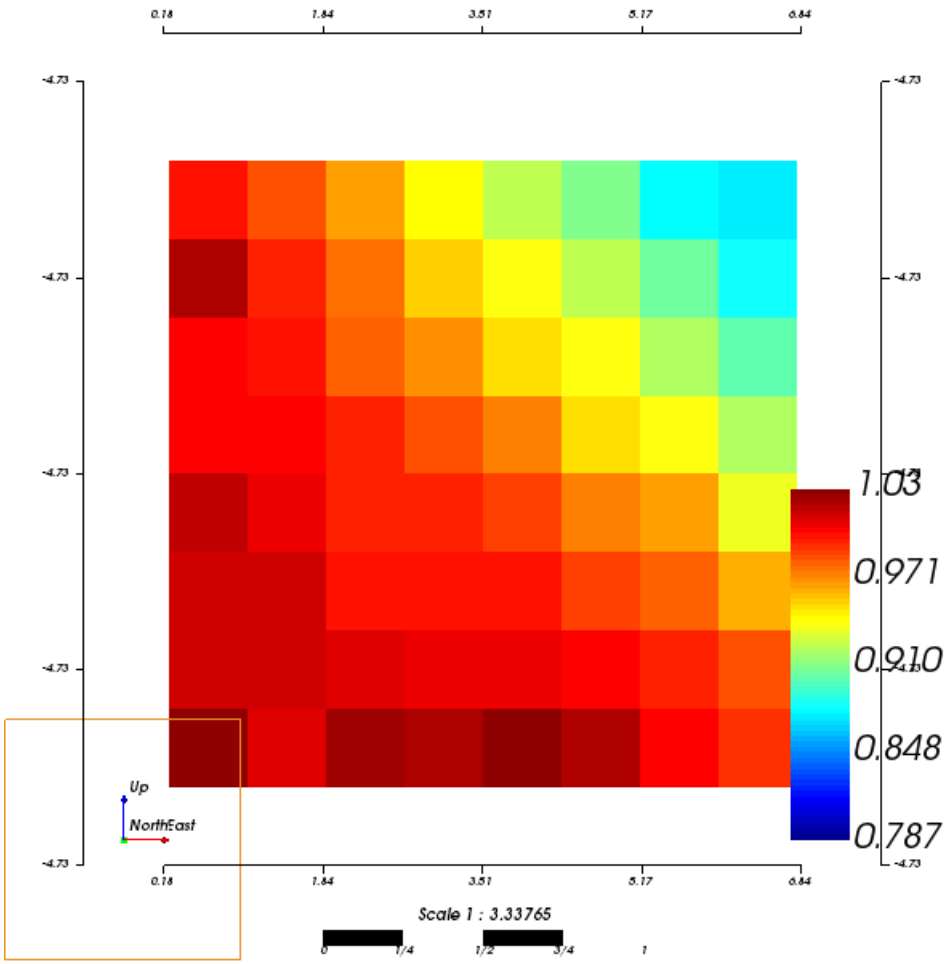

Figura 12.9 – Visão em seção vertical frontal E-W de uma simulação gaussiana com direção de maior continuidade N45 e mergulho 45NE.

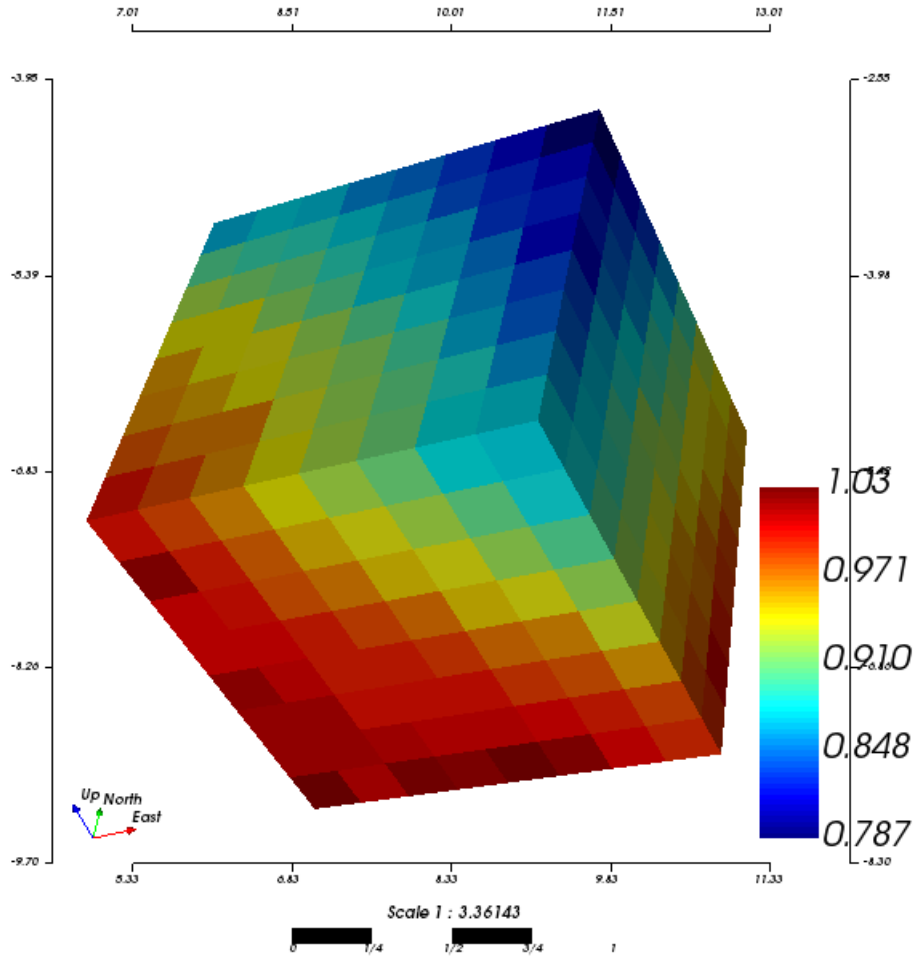

Figura 12.10 – Visão em seção vertical frontal E-W de uma simulação gaussiana com direção de maior continuidade N45 e mergulho 45NE.

O mapa gerado na Figura 12.11 pelo plugin demonstra a mesma continuidade espacial imputada pelos dados à priori. Para o cálculo do mapa de variogramas foi utilizado dez lags com tamanho igual a 1, tolerância linear igual a 0,5, tolerância angular igual a 22,5 tanto horizontal como vertical e banda igual a 0,5 tanto vertical como horizontal.

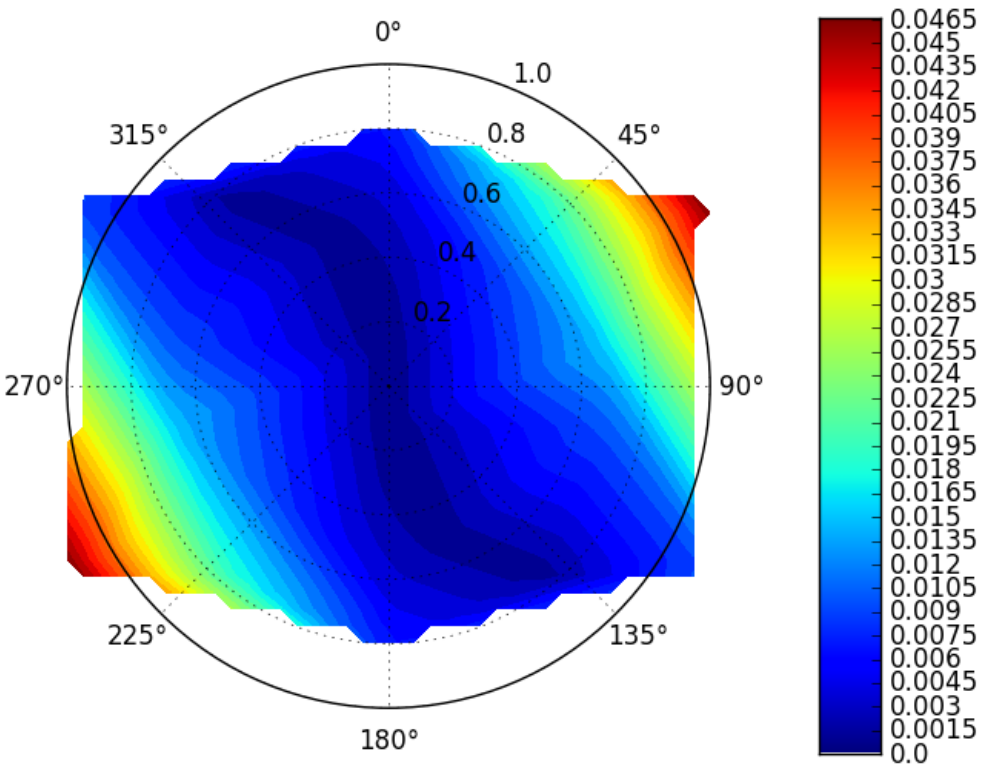

Figura 12.11 – Mapa de variogramas realizado no plano vertical com strike na direção 45°N. Maior continuidade do modelo simulado demonstrado no perfil abaixo.

Uma comparação foi realizada entre os gráficos gerados pelo plugin do Isatis e pelo plugin de variogramas utilizando o banco de dados do Walker Lake (Isaaks & Srivastava, 1989). Nota-se que ambos os programas geram padrões de continuidade semelhantes. A Figura 12.12 demonstra a comparação entre o varmap gerado pelo plugin e pelo Isatis.

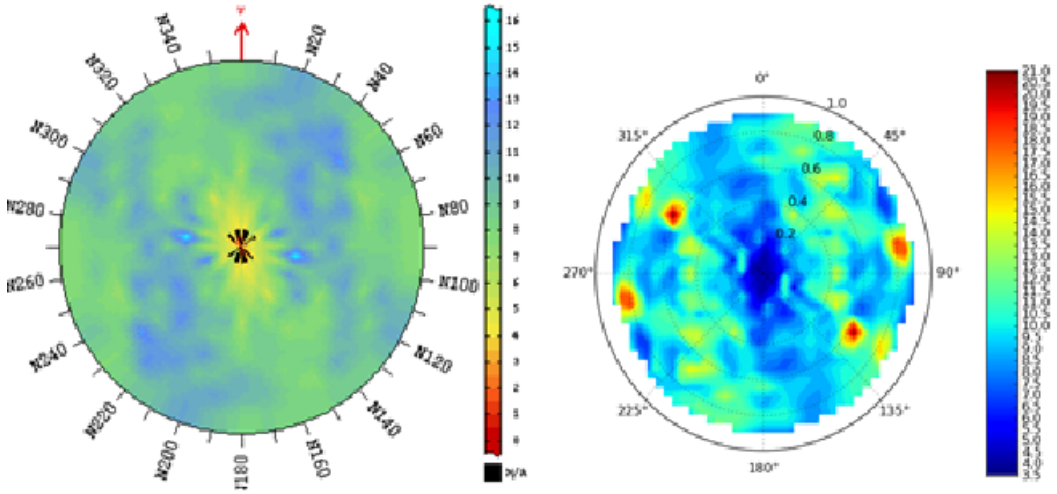

Figura 12.12 – Comparação entre os mapas de variograma do ISATIS e do plugin. A) Mapa do software ISATIS e B) Mapa do software SGeMS. Similaridade da continuidade segundo a direção 157°N. Mapa no plano horizontal

Parte IV

Estudo de caso

Nesta seção serão apresentados dois casos da utilização dos plug-ins. O primeiro caso é relativo a um depósito de ferro com amostras distribuídas tridimensionalmente, para ajuste automático de um modelo, e o segundo caso é o ajuste para a modelagem de um problema multivariado com uso do modelo linear de corregionalização para o os dados de Jura apresentado em Goovaerts (1997).

13 Estudo de caso depósito de ferro (univariado)

Depósitos de ferro sedimentares, provenientes de exalações vulcânicas são característicos de dois períodos distintos de formação: depósitos formados no Fanerozóico, que apresentam hábito oolítico, são pouco bandados e apresentam baixa quantidade de chert e depósitos formados no Pré-Cambriano que se distinguem por laminações por intercalações de ferro e chert, geralmente não oolíticos (MISRA, 2012). O depósito subsequente é uma representação do segundo tipo de depósito também denominado de Lago Superior. Os litotipos de ferro estão intercalados com rochas constituídas de sílica sedimentar e são característicos de bandamentos, também chamados de BIFs (Banded Iron Formations). A Figura 13.1 é um exemplo da mina de Conceição, MG com gênese similar ao depósito das amostras.

Figura 13.1 – Depósito de ferro do tipo Lago Superior da mina de Conceição, Itabira, Minas Gerais. Fonte: http://www.zjmineracao.com.br/noticia.php?id=578 acessado 17/03/2016.

O depósito de ferro selecionado apresenta comportamento semelhante ao depósito da mina de Conceição, em que podemos notar uma estrutura geológica com intercalações dos teores de ferro e quartzo e geometria do depósito semelhante a uma sinclinal, formada a partir da metamorfização do depósito sedimentar. O corpo mineral apresenta alongamento na direção NE, com abaulamento no terceiro quadrante. A Figura 13.2 mostra a morfologia do depósito em um mapa de perfil horizontal.

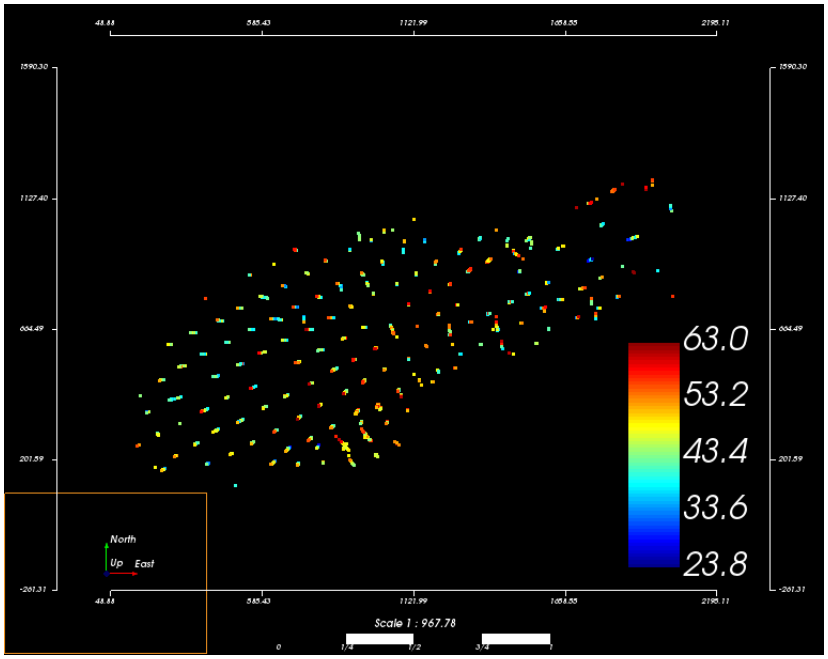

Figura 13.2 – Mapa horizontal do depósito de ferro. Abaulamento no terceiro quadrante.

O corpo do minério de ferro apresenta mergulho característico na direção NE como demonstrado na Figura 13.3:

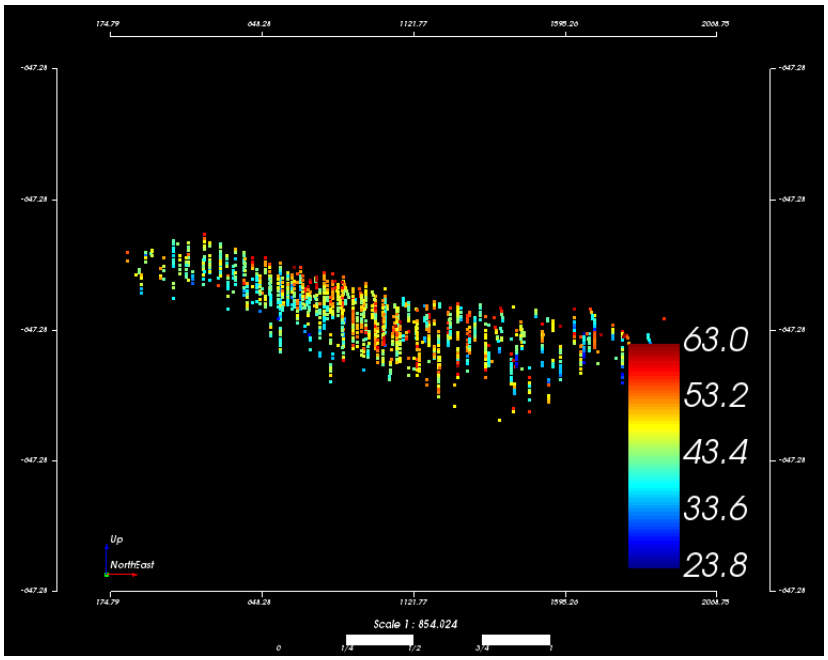

Figura 13.3 – Mapa do depósito de ferro com perfil vertical. Mergulho característico na direção NE.

O depósito é composto de 1195 amostras com padrão de distribuição simétrico, valor médio 46% e variância $44\%^2$. O valor máximo de teor é de 63% e o valor mínimo de 24%. A Figura 13.4 representa um histograma da distribuição das amostras. O coeficiente de

variação das amostras é de apenas $0,14$, o que não demonstra haver variações significativas entre os valores lidos.

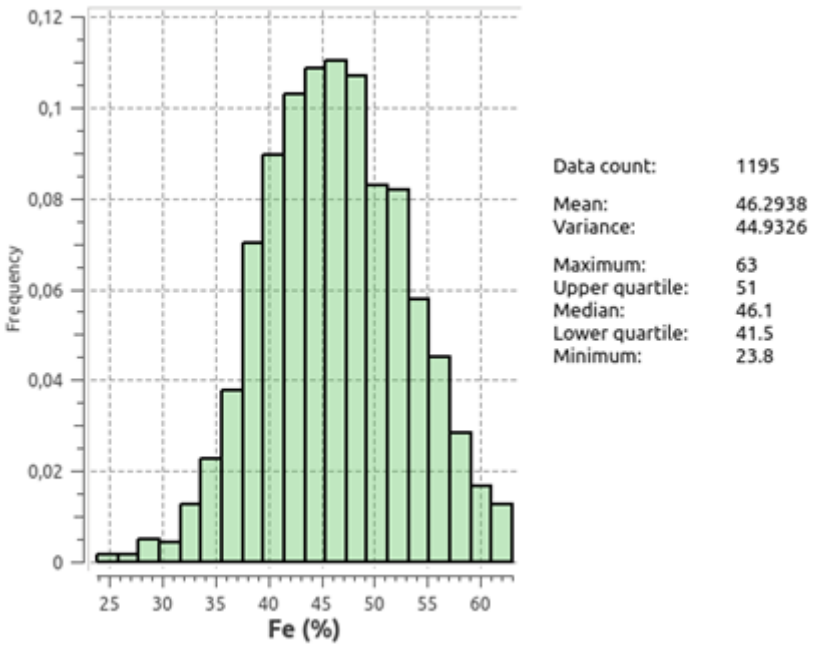

Figura 13.4 – Histograma da distribuição dos dados amostrais do depósito de Ferro.

Para a determinação da continuidade espacial do fenômeno foram criados primordialmente os mapas de variogramas na direção horizontal. A Figura 13.5 demonstra o mapa de variogramas na direção do plano horizontal apresentados tanto pelo plugin quanto pelo Isatis. Notam-se aparentemente duas estruturas mais contínuas no mapa, uma na direção N52 e uma na direção N22,5.

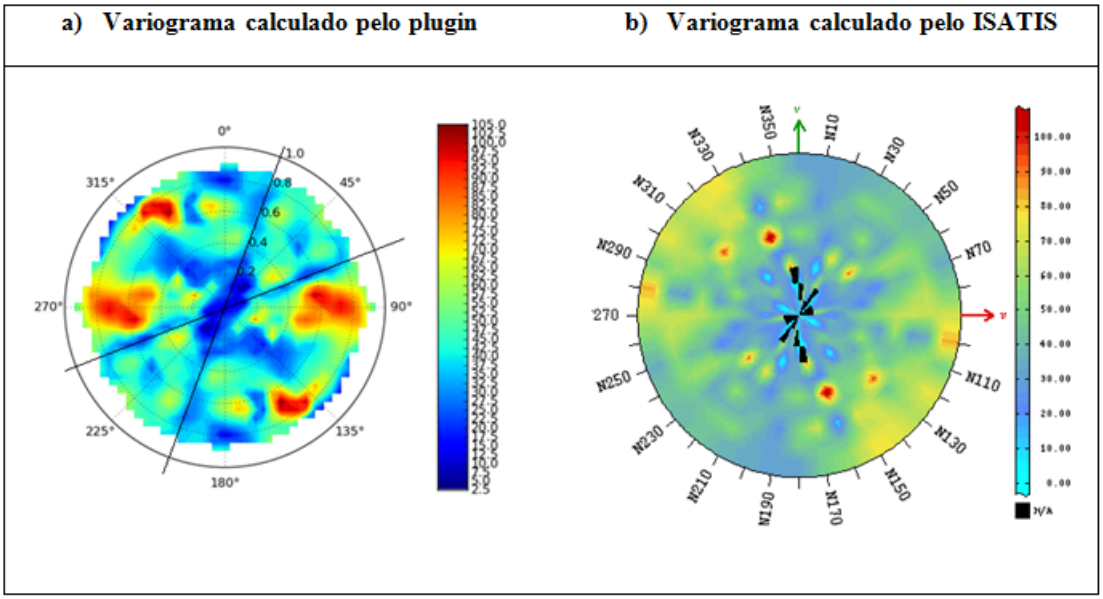

Figura 13.5 – Mapa de variogramas no plano horizontal do depósito de Ferro. A) Variomap calculado pelo plugin e B) Variomap calculado pelo Isatis. Continuidade característica na direção N52 e N22,5.

Os valores gerados pelo plugin e pelo Isatis são diferentes pelo procedimento de cálculo e tipo de interpolação. Os valores do programa são, no entanto, mais suavizados que os valores encontrados pelo plugin.

Nota-se que a continuidade espacial apresenta direção de maior continuidade nesse plano no mergulho de 52NE. Outro mapa foi feito no plano com direção do strike de N52 com mergulho de 90 para a determinação do mergulho da direção de maior continuidade. Podemos notar na Figura 13.6 que o mergulho da direção de maior continuidade está na banda azul que consiste em um mergulho de aproximadamente N52 e que serão confirmados com os cálculos dos variogramas experimentais da Figura 13.9, demonstrando uma continuidade em N52,52NE. Em seguida, para a determinação da continuidade espacial dos demais eixos foi realizado um mapa ortogonal à direção de máxima continuidade como demonstrado na Figura 13.7.

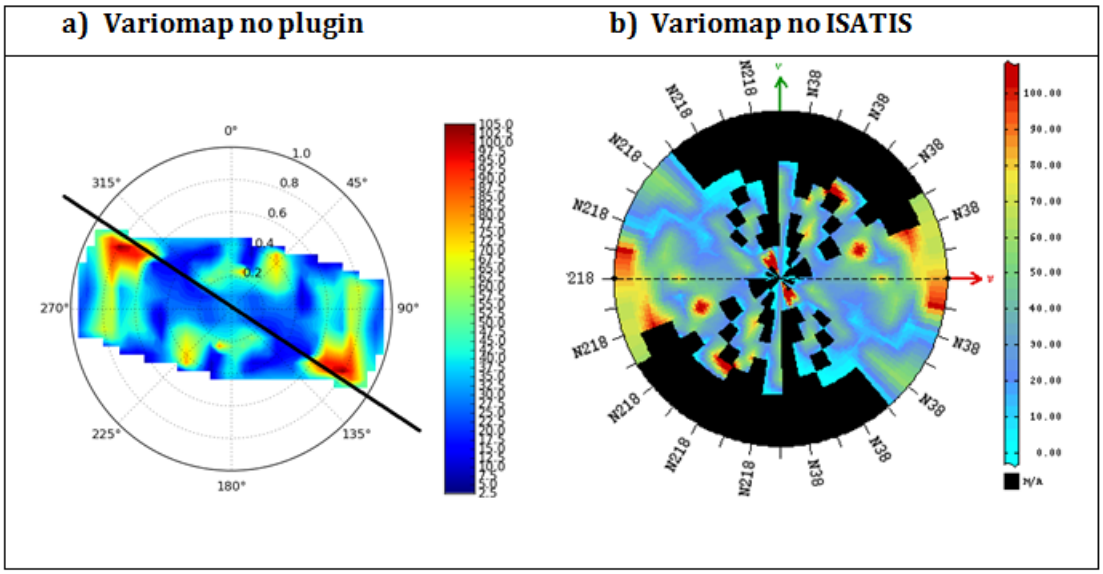

Figura 13.6 – Varmap no plano com direção do strike N52, mergulho 90 a) Mapa de variogramas realizado no plugin. b) Mapa de variogramas realizado no ISATIS.

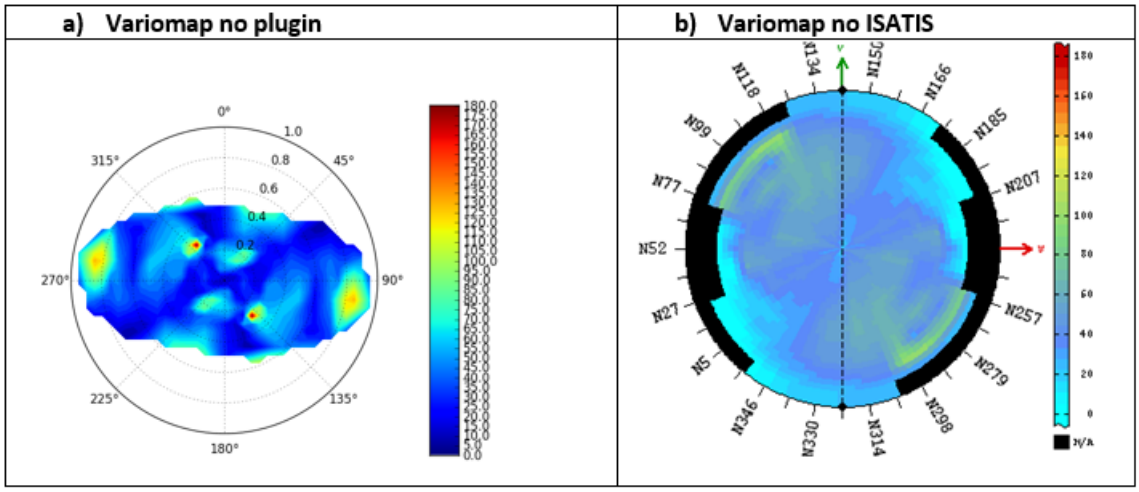

Figura 13.7 – Varmap no plano com direção do strike N322, mergulho 38SW. a) Mapa de variogramas realizado no plugin. b) Mapa de variogramas realizado no ISATIS.

A Figura 13.8 representa os variogramas experimentais na direção horizontal, constituindo a primeira etapa de verificação da continuidade espacial dos dados. Todos os variogramas foram gerados pelo plugin e são funções pairwise. Nota-se que a continuidade espacial no plano horizontal está na direção N52.

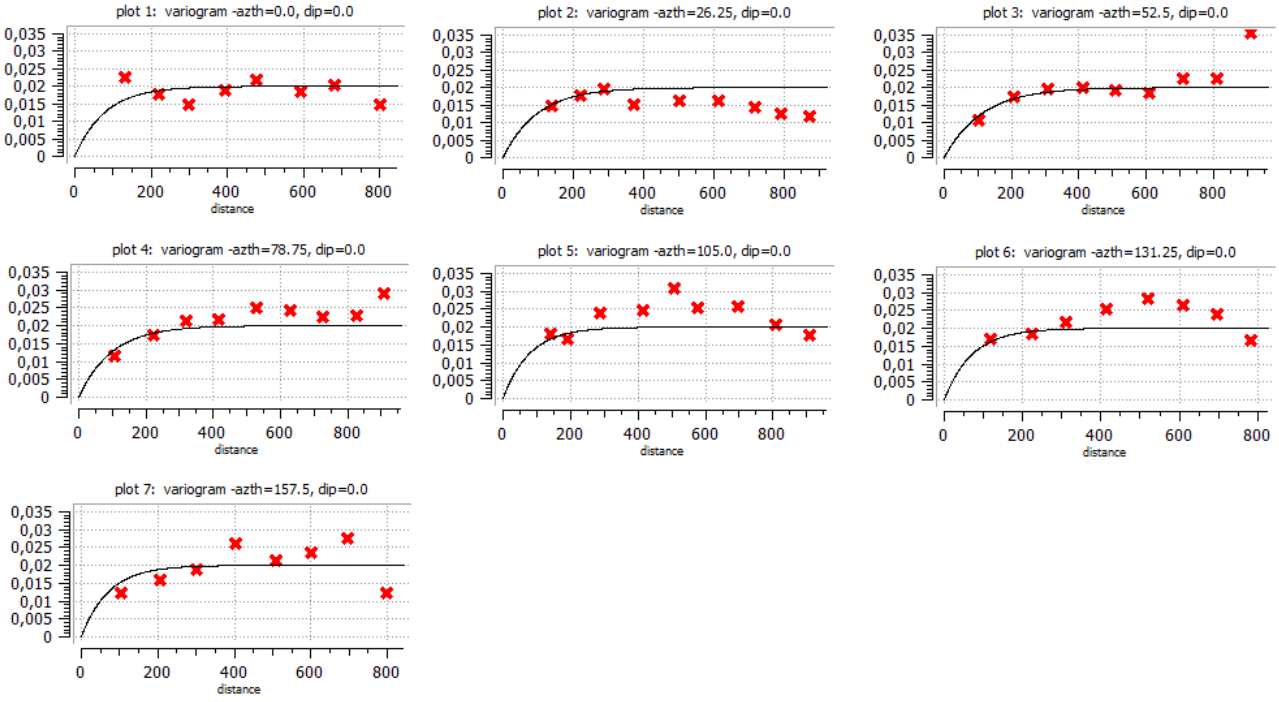

Figura 13.8 – Direção de maior continuidade no plano horizontal. Variograma mais contínuo na direção de azimute N52.

A Figura 13.9 demonstra a direção de maior continuidade no plano vertical à direção de máxima continuidade no plano horizontal. Com isso determina-se o eixo de maior continuidade como sendo N52, 52NE.

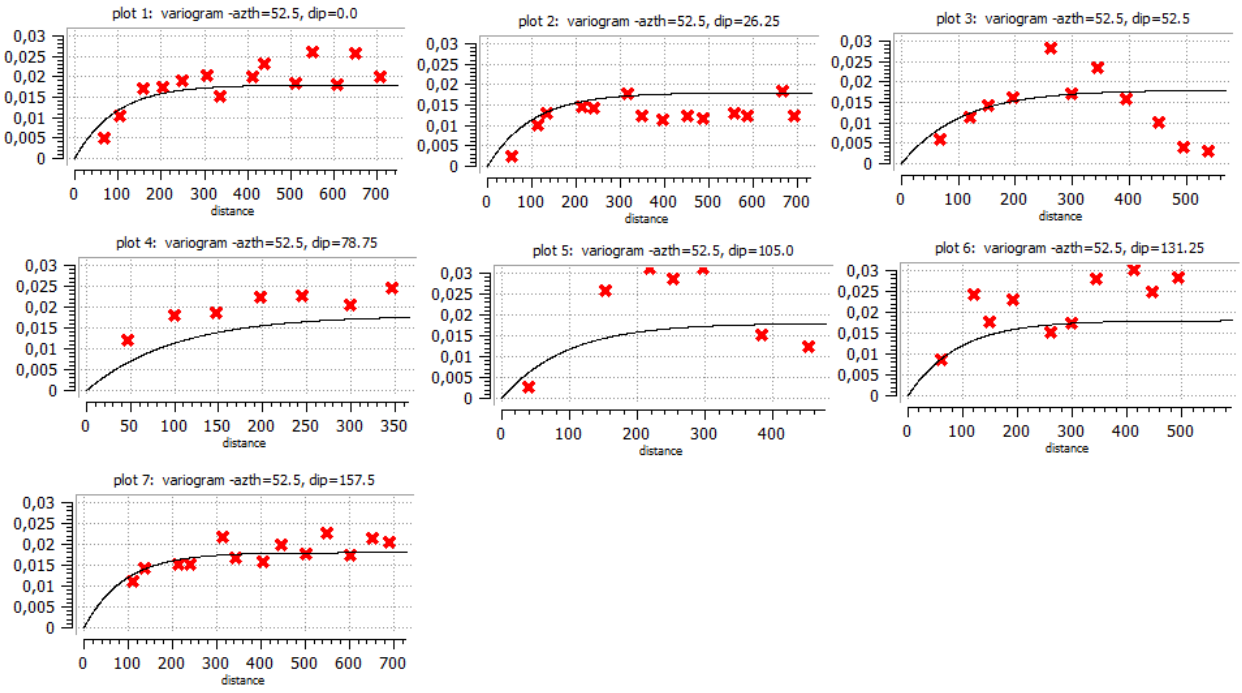

Figura 13.9 – Direções de maior continuidade na direção vertical. Variograma mais contínuo na direção do mergulho 52NE.

Para determinar os eixos de continuidade secundário e intermediário foi realizado os variogramas no plano perpendicular à direção principal. Notou-se anisotropia na direção 232N 38SW. A Figura 13.10 demonstra os dados experimentais. Optou-se por utilizar como continuidade média a reta orientada na horizontal e a reta de menor continuidade como a perpendicular aos demais eixos.

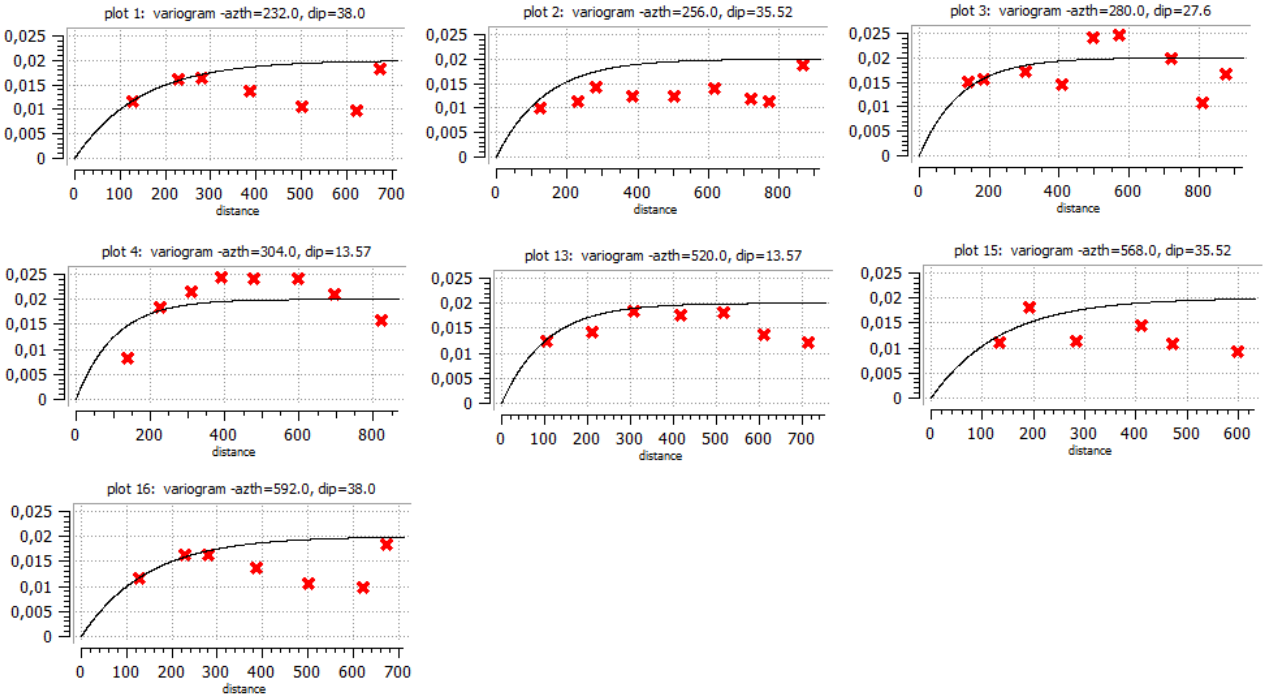

Figura 13.10 – Direções de maior continuidade nas direções perpendiculares à de maior continuidade espacial. Variograma mais contínuo na direção $N232,38SW$.

Para o cálculo dos valores dos variogramas experimentais utilizados para a modelagem automática foram utilizados os seguintes parâmetros:

Tabela 13.1 – Parâmetros experimentais dos variogramas modelados automaticamente

Direção	**Parâmetros**	**Valores**
Principal, Secundária, Intermediária	Número de lags	10
Principal, Secundária, Intermediária	Tamanho do lag	25m
Principal, Secundária, Intermediária	Tolerância linear	12,5m
Principal, Secundária, Intermediária	Tolerância angular horizontal	$22,5°$
Principal, Secundária, Intermediária	Tolerância angular vertical	$22,5°$
Principal, Secundária, Intermediária	Banda horizontal	25m
Principal, Secundária, Intermediária	Banda vertical	25m

Os valores foram automatizados para os três eixos da direção de máxima, média e mínima segundo as Figuras 13.11, 13.12 e 13.13.

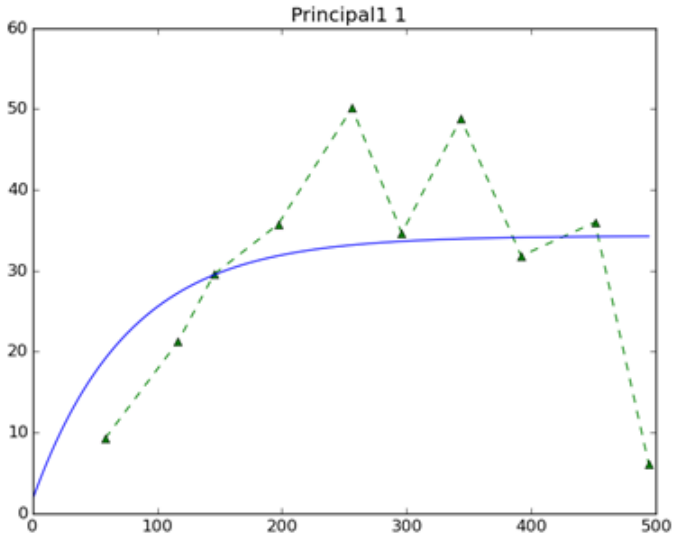

Figura 13.11 – Variograma da direção principal automatizado.. Variograma mais contínuo na direção $N52,52NE$.

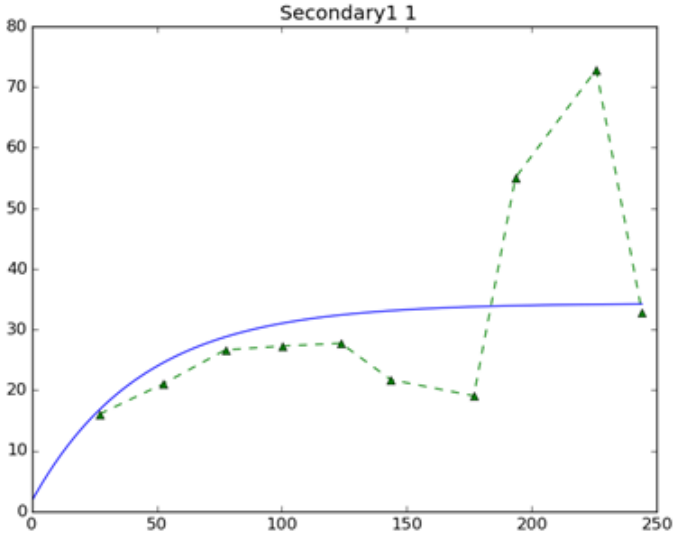

Figura 13.12 – Variograma da direção secundária. Variograma mais contínuo na direção $N142,0SE$.

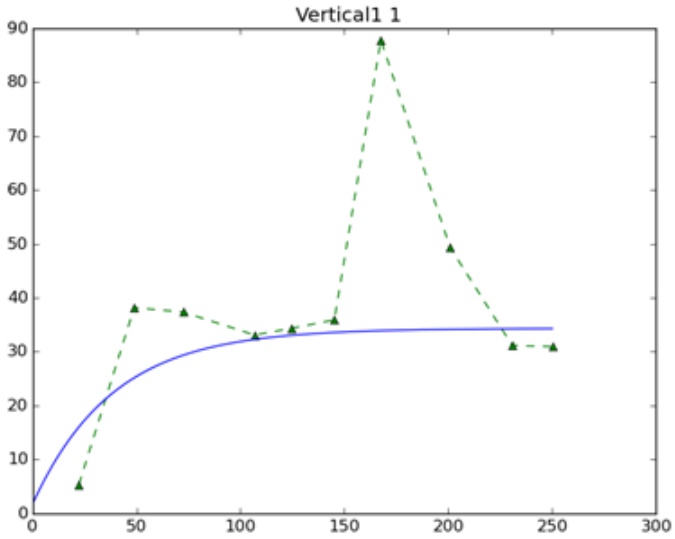

Figura 13.13 – Variograma da direção intermediária. Variograma mais contínuo na direção $N232,38SW$.

O modelo de variograma encontrado é Exponencial (Exp()), com efeito pepita 1,70, contribuição igual 32,60, direções do eixo de anisotropia na N52,52 NE, N142,0 SE, N232,38SW com valores de alcance iguais a 228m, 131m e 115m. A Equação 13.1 demonstra o modelo de variograma representando os eixos de anisotropia:

$$\gamma(h) = 1,70 + 32,60\text{Exp}\left(\frac{228m}{N52,52NE}, \frac{131m}{N142,0SE}, \frac{115m}{N232,38SW}\right) \tag{13.1}$$

14 Estudo de caso do depósito de Jura (Caso multivariado)

O banco de dados Jura provém de uma região formada por molassas no oeste dos Alpes. As rochas são formadas por material do período Jurássico e apresentam a maior quantidade de sedimentos do local. O local de estudo apresenta aproximadamente 400km de comprimento e um arco interno de aproximadamente 340km (SOMMARUGA, 1996). A área amostrada apresenta estrutura em forma de triângulo característico de depósitos sedimentares de molassas. A Figura 14.1 demonstra a disposição das amostras em mapa.

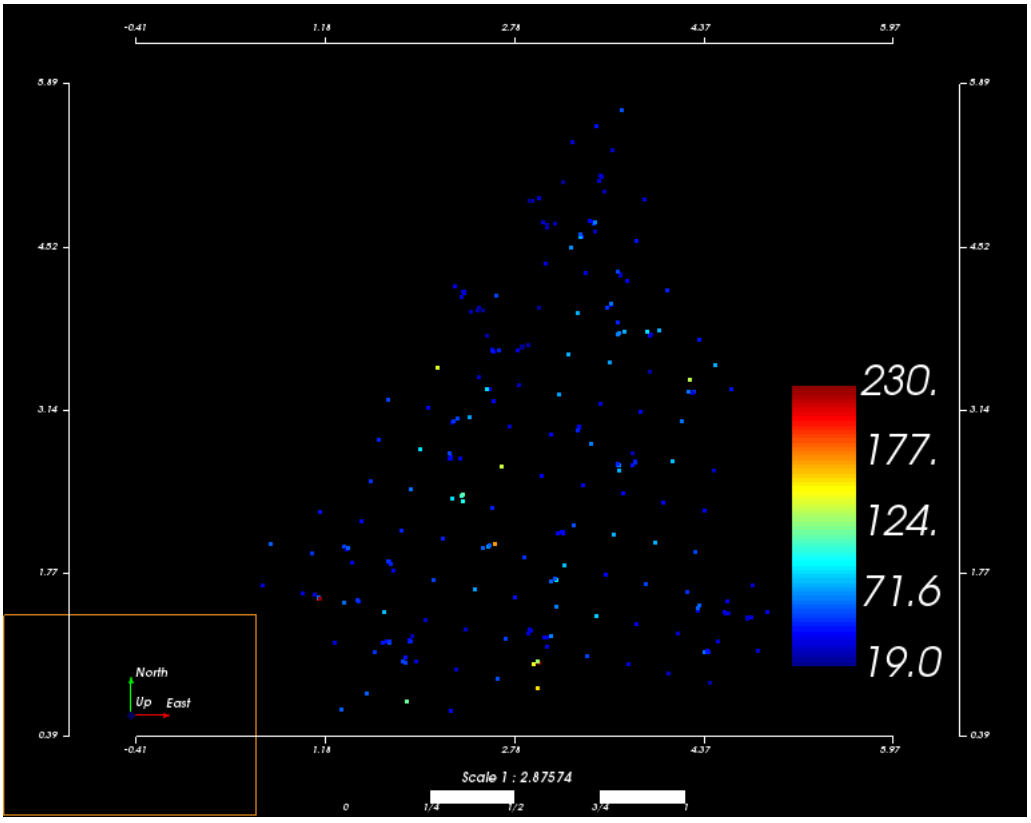

Figura 14.1 – Mapa de disposição das amostras de Pb. Dados isotópicos para todas as variáveis, Ni,Cu e Pb.

O banco de dados Jura corresponde a 259 amostras de concentrações de sete metais pesados: cádimo, cobalto, cromo, cobre, níquel, zinco, cobalto e chumbo.

Neste trabalho, foram escolhidas três variáveis correlacionadas: níquel, cobre e chumbo. As Figuras 14.2 a 14.4 demonstram a correlação entre as variáveis de 0,22 entre NixCu, 0.77 entre NixPb e 0,31 entre CuxPb .

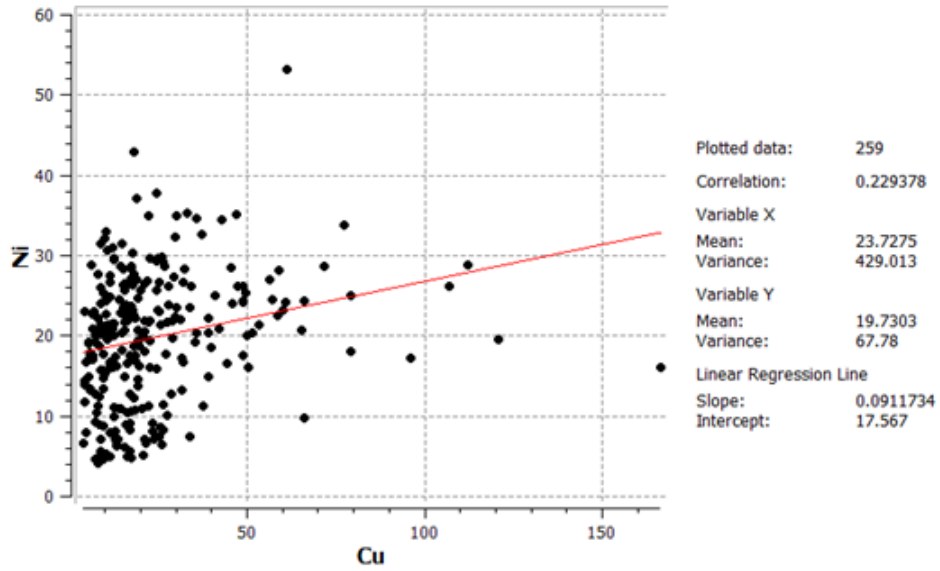

Figura 14.2 – Correlação entre a variável Ni e Cu.

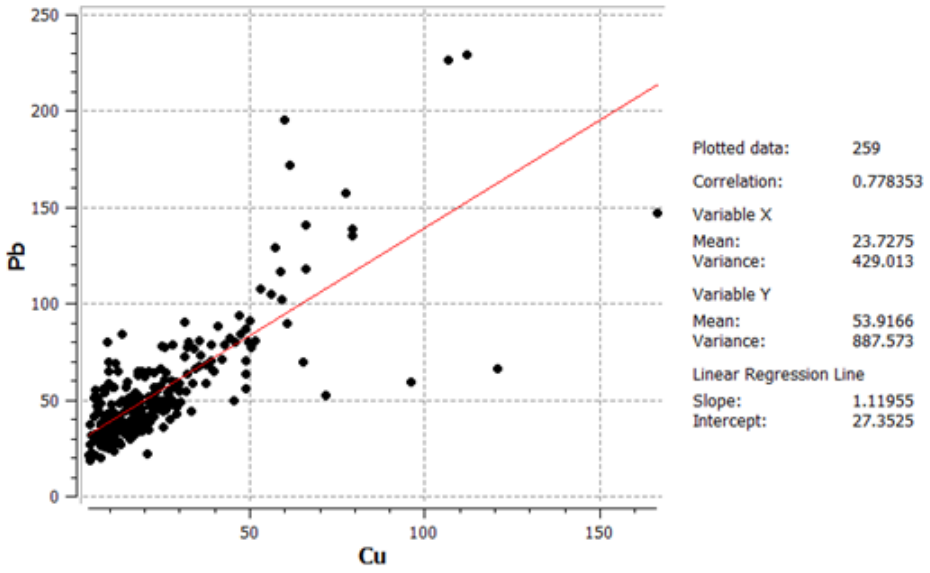

Figura 14.3 – Correlação entre a variável Pb e Cu.

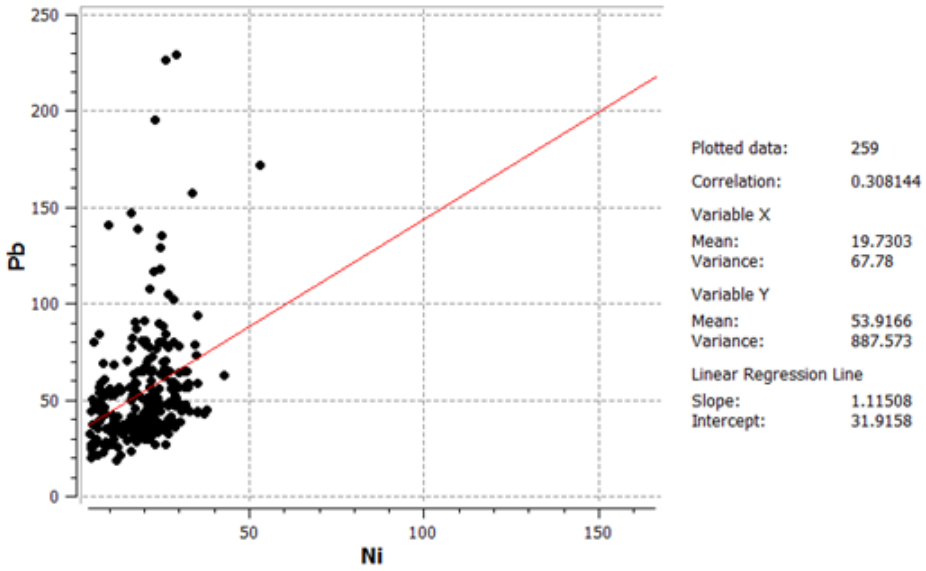

Figura 14.4 – Correlação entre a variável Pb e Ni.

A Figura 14.5 mostra o mapa da continuidade espacial da variável Cu. Nota-se, que a variável apresenta isotropia, tal como todas as outras variáveis.

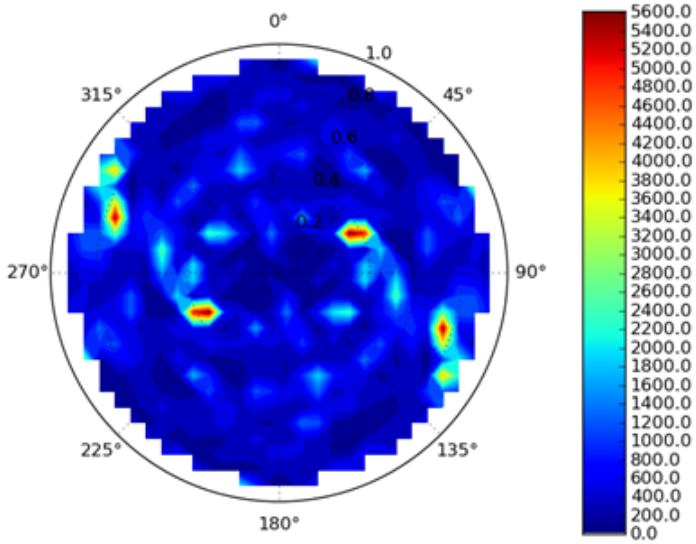

Figura 14.5 – Mapa da continuidade espacial da variável Cu.

Os valores dos variogramas experimentais foram calculados segundo a Tabela 14.1:

Tabela 14.1 – Parâmetros do variograma experimental.

Parâmetro do variograma	**Valor**
Número de lags	20
Tamanho do lag	0,1 km
Tolerância linear	0,05 km
Tolerância angular horizontal	11,25 graus
Tolerância angular vertical	11,25 graus
Banda horizontal	0,05 km
Banda vertical	0,05 km

A análise espacial foi realizada utilizando os variogramas pairwise para as diferentes variáveis, buscando verificar a condição de anisotropia. As Figuras 14.6, 14.7 e 14.8 demonstram os variogramas pairwise para as variáveis Ni, Pb e Cu. Nota-se que as variáveis podem ser consideradas isotrópicas segundo a modelagem dos dados.

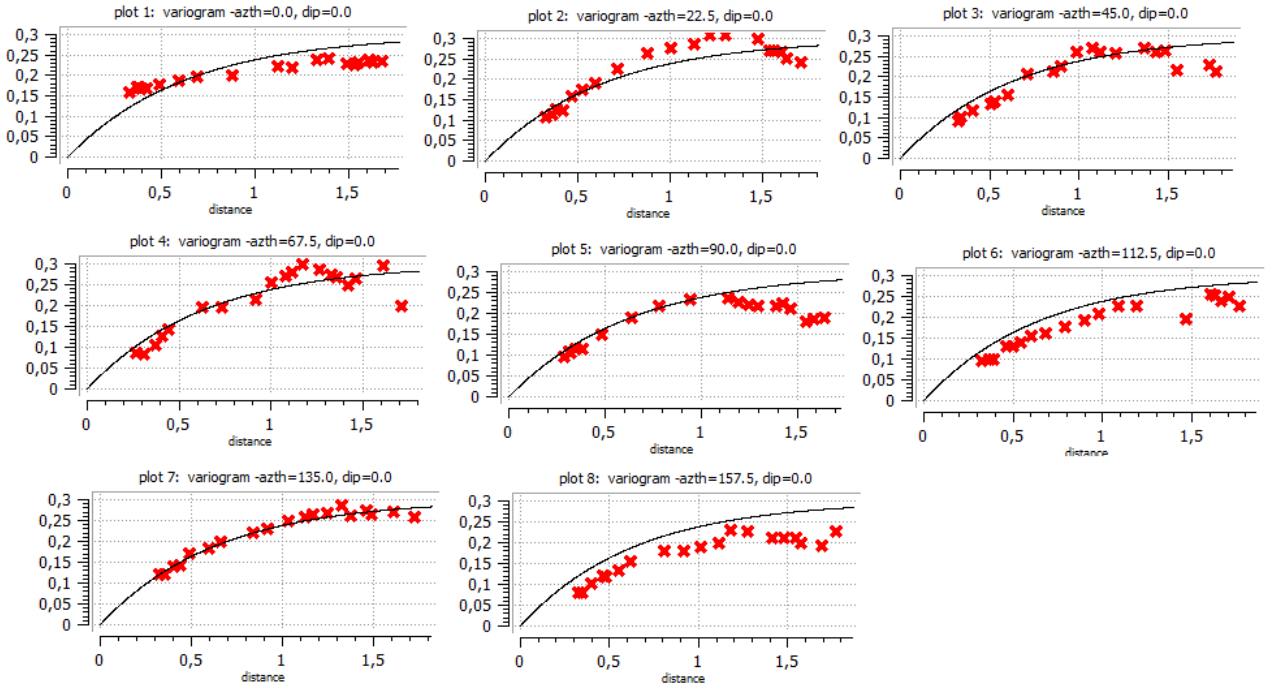

Figura 14.6 – Variogramas da variável níquel. Caso omnidirecional.

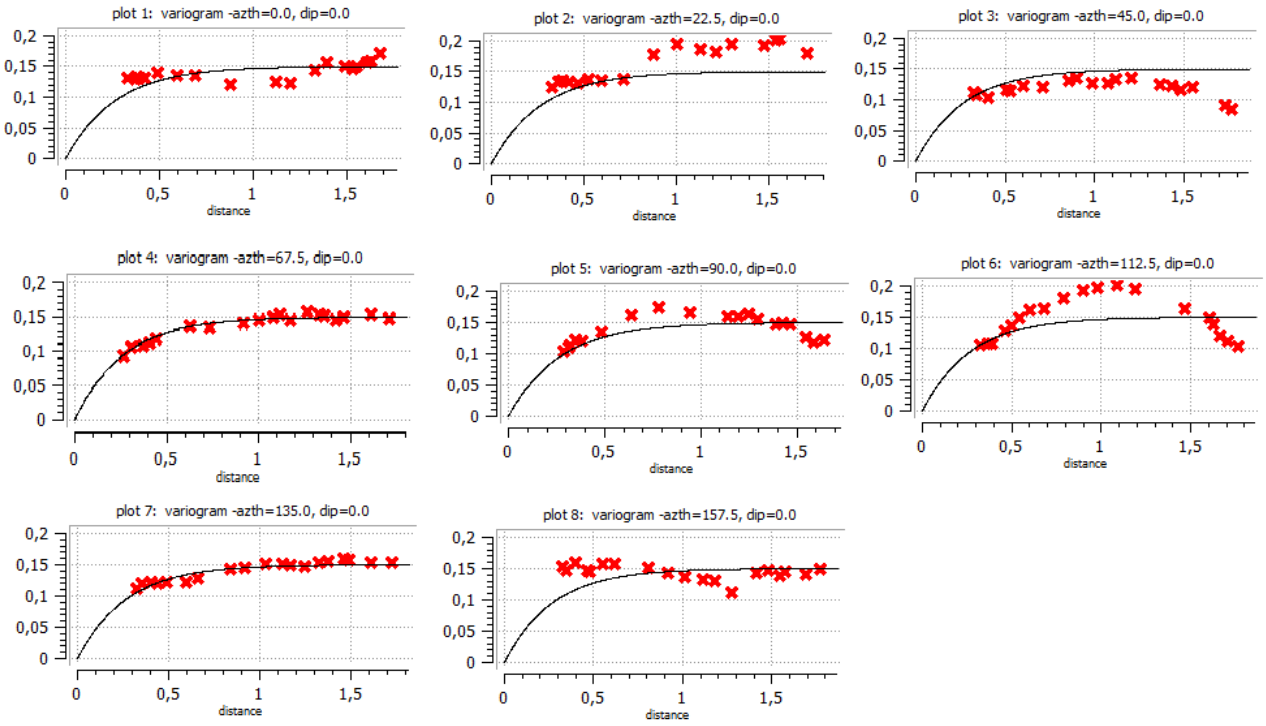

Figura 14.7 – Variogramas da variável chumbo. Caso omnidirecional.

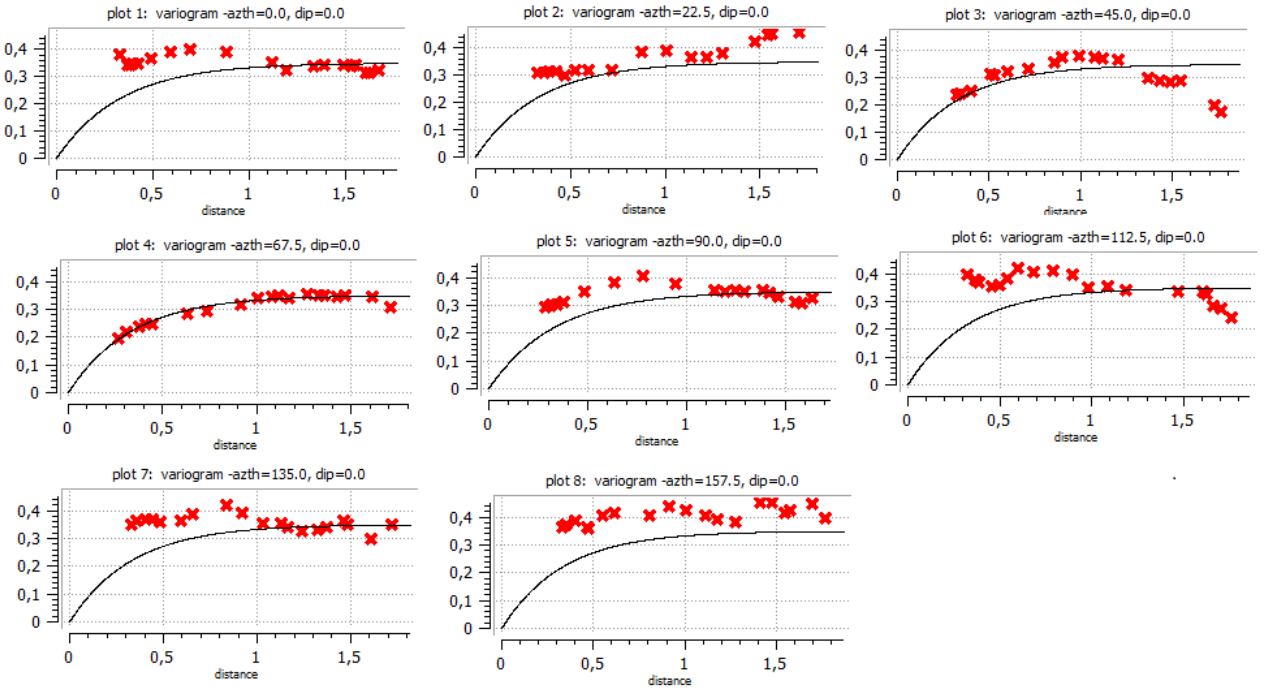

Figura 14.8 – Variogramas da variável cobre. Caso omnidirecional.

Um modelo linear de corregionalização foi construído com os algoritmos de ajuste

automático desenvolvido. Os variogramas diretos e cruzados ajustados automaticamente na direção principal e secundária demonstram-se na Figura 14.9:

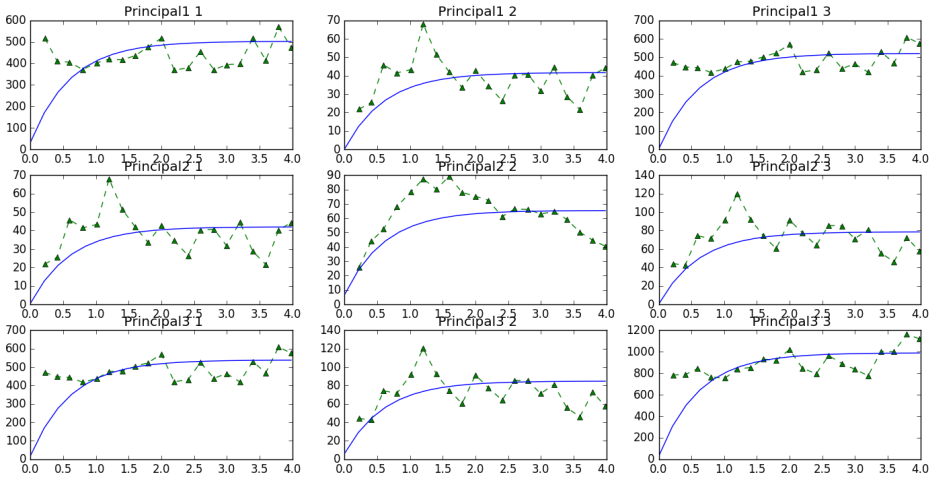

Figura 14.9 – Variogramas dos dados de Jura. Variável 1- Cu , 2-Ni e 3-Pb. Valores de índices iguais representam variogramas diretos. Valores de índices diferentes representam variogramas cruzados.

O resultado do modelo ajustado para o modelo linear de corregionalização está descrito segundo as equações 14.1 a 14.6, sendo o índice 1 do variograma respectivo à variável Cu, o índice 2 respectivo à variável Ni e o índice 3 respectivo à variável Pb:

$$\gamma_{11}(h) = 39.85 + 468.37\left(1 - \exp\left(\frac{-h}{1.84}\right)\right) \tag{14.1}$$

$$\gamma_{12}(h) = 0.88 + 41.39\left(1 - \exp\left(\frac{-h}{1.84}\right)\right) \tag{14.2}$$

$$\gamma_{13}(h) = 15.36 + 518.25\left(1 - \exp\left(\frac{-h}{1.84}\right)\right) \tag{14.3}$$

$$\gamma_{22}(h) = 4.55 + 68.12\left(1 - \exp\left(\frac{-h}{1.84}\right)\right) \tag{14.4}$$

$$\gamma_{23}(h) = 1.01 + 78.36\left(1 - \exp\left(\frac{-h}{1.84}\right)\right) \tag{14.5}$$

$$\gamma_{33}(h) = 28.27 + 960.51\left(1 - \exp\left(\frac{-h}{1.84}\right)\right) \tag{14.6}$$

A Tabela 14.2 resume os parâmetros encontrados na modelagem automática do depósito de Jura:

Tabela 14.2 – Tabela dos valores modelados dos variogramas diretos e cruzados do depósito de Jura segundo um modelo linear de corregionalização

Variável head	**Variável no tail**	**Efeito pepita**	**Contribuição**	**Alcance**
Cobre	Cobre	39.85 %^{2}	468,37 %^{2}	1,84 km
Cobre	Níquel	0,88 %^{2}	41,39 %^{2}	1,84 km
Cobre	Chumbo	15.36 %^{2}	518,25 %^{2}	1,84 km
Níquel	Níquel	4.55 %^{2}	68,12 %^{2}	1,84 km
Níquel	Chumbo	1.01 %^{2}	78,36 %^{2}	1,84 km
Chumbo	Chumbo	28.27 %^{2}	960,51 %^{2}	1,84 km

15 RESULTADOS E DISCUSSÃO

Os plug-ins foram testados em dois estudos de caso, sendo o primeiro relacionado com um depósito de ferro do tipo Lago Superior e o segundo com um depósito polimetálico de Jura. A continuidade espacial do primeiro depósito se encontra aproximadamente em N52 52NE, com ranges (228, 131, 115) m, efeito pepita 1,70 e contribuição de 32,60 com um modelo exponencial em relação à variável ferro. O segundo depósito é um depósito polimetálico de Jura em que foi ajustado um modelo linear de corregionalização com três variáveis, cobre, chumbo e níquel apresentando um modelo básico exponencial, isotrópico e com range igual a 0,69. As ferramentas utilizadas para auxílio na variografia permitiram agilizar o reconhecimento da continuidade espacial pela utilização de mapas de variograma além de um ajuste mais rápido pela modelagem automática.

Parte V

CONCLUSÃO E TRABALHOS FUTUROS

16 CONCLUSÃO

A variografia, apesar de ser uma etapa inicial no processo de estimativa, ainda consiste nos procedimentos mais trabalhosos dentro das técnicas geoestatísticas. As diversas funções de continuidade espacial robustas, os modelos de ajuste automático e a utilização de mapas de variograma ajudam a reduzir o esforço do modelador. Os depósitos minerais complexos apresentam estruturas complexas que não são triviais de se avaliar, o que torna a utilização de ferramentas auxiliares muito mais necessárias. Nesta dissertação, optou-se por desenvolver plug-ins de auxílio à continuidade espacial que pudessem ser incorporados no SGeMs como ferramentas auxiliares. Utilizou-se como base da estrutura dos algoritmos o software GamV do GSLib. Os valores gerados pelo plug-in e pelo programa original obtiveram alta correlação. Assim, foram feitas quatro rotinas, três baseadas no GamV, que consistem em mapas de variogramas, hscatterplots e cálculo das funções de continuidade, e um plug-in de modelagem automática baseada no modelo de Monte Carlo e que valida os dados para um modelo linear de corregionalização. Os programas foram testados em dois bancos de dados, um depósito de ferro e outro polimetálico. Os plug-ins criados se provaram de mais rápida utilização e interpretação dos dados do que a utilização dos executáveis do GSLIB. Os gráficos gerados automaticamente facilitam ao operador analisar a continuidade espacial.

17 TRABALHOS FUTUROS

Como trabalhos futuros propõem-se desenvolver as ferramentas em um ambiente de execução personalizado, com a inserção de botões, geração automática de gráficos e demais ferramentas gráficas para facilitar ainda mais a utilização dos algoritmos. Propõe-se a utilização de variogramas ponderados, e também ajustes de funções por máxima verossimilhança e transformada de Fourier.

Referências

BULIT, C. Good reasons and guidance for mapping planktonic protist distributions. *Acta Protozoologica*, Jagiellonian University-Jagiellonian University Press, v. 53, n. 1, p. 13, 2014. Citado na página 49.

CRESSIE, N. Fitting variogram models by weighted least squares. *Journal of the International Association for Mathematical Geology*, Springer, v. 17, n. 5, p. 563–586, 1985. Citado 5 vezes nas páginas 22, 32, 37, 49 e 50.

CRESSIE, N.; HAWKINS, D. M. Robust estimation of the variogram: I. *Journal of the International Association for Mathematical Geology*, Springer, v. 12, n. 2, p. 115–125, 1980. Citado 3 vezes nas páginas 32, 35 e 37.

DAVID, M. *Geostatistical ore reserve estimation*. [S.l.]: Elsevier, 1977. 364 p. Citado 2 vezes nas páginas 35 e 50.

DEUTSCH, C. V.; JOURNEL, A. et al. *Geostatistical software library and user's guide*. [S.l.: s.n.], 1998. Citado 4 vezes nas páginas 9, 25, 29 e 41.

EMERY, X. Iterative algorithms for fitting a linear model of coregionalization. *Computers & Geosciences*, Elsevier, v. 36, n. 9, p. 1150–1160, 2010. Citado na página 49.

GENTON, M. G. Variogram fitting by generalized least squares using an explicit formula for the covariance structure. *Mathematical Geology*, Springer, v. 30, n. 4, p. 323–345, 1998. Citado 2 vezes nas páginas 49 e 50.

GOOVAERTS, P. *Geostatistics for natural resources evaluation*. [S.l.]: Oxford University Press on Demand, 1997. Citado 5 vezes nas páginas 9, 32, 37, 47 e 48.

GRINGARTEN, E.; DEUTSCH, C. V. Teacher's aide variogram interpretation and modeling. *Mathematical Geology*, Springer, v. 33, n. 4, p. 507–534, 2001. Citado 5 vezes nas páginas 22, 33, 44, 45 e 47.

ISAAKS, E. H. et al. *Applied geostatistics*. [S.l.]: Oxford University Press, 1989. Citado 4 vezes nas páginas 9, 33, 34 e 40.

JOURNEL, A. G.; HUIJBREGTS, C. J. *Mining geostatistics*. [S.l.]: Academic press, 1978. Citado 3 vezes nas páginas 36, 40 e 45.

LARRONDO, P. F.; NEUFELD, C. T.; DEUTSCH, C. V. Varfit: A program for semi-automatic variogram modeling. *Fifth Annual Report of the Centre for Computational Geostatistics, University of Alberta, Edmonton*, 2003. Citado 3 vezes nas páginas 23, 48 e 49.

MA, X.; JOURNEL, A. G. An expanded gslib cokriging program allowing for two markov models. *Computers & Geosciences*, Elsevier, v. 25, n. 6, p. 627–639, 1999. Citado na página 24.

MATHERON, G. Principles of geostatistics. *Economic geology*, Society of Economic Geologists, v. 58, n. 8, p. 1246–1266, 1963. Citado 5 vezes nas páginas 30, 32, 35, 37 e 49.

MISRA, K. *Understanding mineral deposits*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 82.

PARDO-IGÚZQUIZA, E. Varfit: a fortran-77 program for fitting variogram models by weighted least squares. *Computers & Geosciences*, Elsevier, v. 25, n. 3, p. 251–261, 1999. Citado 2 vezes nas páginas 49 e 50.

PESQUER, L.; CORTÉS, A.; PONS, X. Parallel ordinary kriging interpolation incorporating automatic variogram fitting. *Computers & geosciences*, Elsevier, v. 37, n. 4, p. 464–473, 2011. Citado na página 49.

SADAWY, M.; ISMAEL, A.; GOUDA, M. Geostatistical analysis for corrosion in oil steel tank. *American Journal of Science and Technology*, v. 2, n. 2, p. 38–42, 2015. Citado na página 49.

SHAPIRO, A.; BOTHA, J. D. Variogram fitting with a general class of conditionally nonnegative definite functions. *Computational Statistics & Data Analysis*, Elsevier, v. 11, n. 1, p. 87–96, 1991. Citado na página 43.

SOMMARUGA, A. *Geology of the central Jura and the molasse basin*. Tese (Doutorado) — Université de Neuchâtel, 1996. Citado na página 92.

TAHMASEBI, P.; HEZARKHANI, A.; SAHIMI, M. Multiple-point geostatistical modeling based on the cross-correlation functions. *Computational Geosciences*, Springer, v. 16, n. 3, p. 779–797, 2012. Citado 2 vezes nas páginas 24 e 34.

WACKERNAGEL, H. *Multivariate geostatistics: an introduction with applications*. [S.l.]: Springer Science & Business Media, 2013. Citado 5 vezes nas páginas 20, 33, 40, 46 e 48.

Apêndices

APÊNDICE A – Algoritmo do hscatterplot

Todos os algoritmos encontram-se disponíveis no repositório:

https://github.com/LPM-UFRGS/lpm-python-course-projects/tree/master/dez-2015/varmap-plugin

```
#!/bin/python
import sgems
import matplotlib.pyplot as plt
from scipy import stats
from matplotlib import colors,ticker,cm
from scipy.interpolate import griddata
from scipy import interpolate
import numpy as np
import math
import random
import os

#.....................................................
def retire_nan (x,y,z,v):
 xn =[]
 yn =[]
 zn =[]
 vn =[]
 for i in range(0,len(v)):
  if (v[i] > -99999999999999999):

   xn.append(x[i])
   yn.append(y[i])
   zn.append(z[i])
   vn.append(v[i])
 return xn, yn, zn, vn
#.....................................................

class   hscatter:
 def _init_(self):
  pass
 def initialize(self,params):
```

```
    self.params = params
    return True
def execute(self):

'''
HSCATTERPLOT
-----------------------------------------------------------------------

This is a template for SGems program variogram and covariance maps.
The user have to fill the parameters with a ui.file with same name
of this python file before start program. The QT userinterface connect
SGems data with this routine
using the class params.

The output of this program is a graphic demonstranting interpolated
variogram values along one plane

AUTHOR: DAVID ALVARENGA DRUMOND  2016

'''

'''
INITIAL PARAMETERS
-----------------------------------------------------------------------

All initial parameters are transcribe by ui.file interface. Variables
are choose among common variables. The follow variables are:

COMMON VARIABLES:

head_property= head property
head_grid = head grid
tail_property = tail property
tail_grid= tail grid
C_variogram = check box for variogram maps
C_covariance= check box for covariance maps
nlags= number of lags in experimental variogram
lagdistance = lag length for this experimental variograms
lineartolerance= linear tolerance for this map
```

```
htolerance = horizontal tolerance for this map
vtolerance = vertical tolerance for this lag
hband = horizontal band
vband = vertical band
dip = dip of the rake of variogram map
azimute = azimuth of the rake of variogram map
gdiscrete = number of cells to discretize map
ncontour = number of contourn lines to interpolate in map


'''


# Stabilish initial parameters

head_property= self.params['head_prop']['property']
head_grid = self.params['head_prop']['grid']
tail_property = self.params['tail_prop']['property']
tail_grid= self.params['tail_prop']['grid']
nlags= int(self.params['nlags']['value'])
lagdistance = float(self.params['lagdistance']['value'])
lineartolerance= float(self.params['lineartolerance']['value'])
htolerance = float(self.params['htolangular']['value'])
vtolerance = float(self.params['vtolangular']['value'])
hband = float(self.params['hband']['value'])
vband = float(self.params['vband']['value'])
Dip = float(self.params['Dip']['value'])
Azimute= float(self.params['Azimute']['value'])

print (self.params)

# Get values from SGems and exclude nan values

xh = []
yh = []
zh = []
vh = []

xh1 = sgems.get_property(head_grid, "_X_")
```

```
yh1 = sgems.get_property(head_grid, "_Y_")
zh1 = sgems.get_property(head_grid, "_Z_")
vh1 = sgems.get_property(head_grid, head_property)

xh, yh, zh, vh = retire_nan(xh1,yh1,zh1,vh1)


xt = []
yt = []
zt = []
vt = []

xt1 = sgems.get_property(tail_grid, "_X_")
yt1 = sgems.get_property(tail_grid, "_Y_")
zt1 = sgems.get_property(tail_grid, "_Z_")
vt1 = sgems.get_property(tail_grid, tail_property)

xt, yt, zt, vt = retire_nan(xt1,yt1,zt1,vt1)

# Verify number of dimensions of problem

# Verify dimensions of head

cdimensaoxh = False
cdimensaoyh = False
cdimensaozh = False

for x in range(0,len(xh)):
 if (xh[x] != 0):
  cdimensaoxh = True


for y in range(0,len(yh)):
 if (yh[y] != 0):
  cdimensaoyh = True


for z in range(0,len(zh)):
 if (zh[z] != 0):
```

```
    cdimensaozh = True

 # Verify dimensions of tail

 cdimensaoxt = False
 cdimensaoyt = False
 cdimensaozt = False

 for x in range(0,len(xt)):
  if (xt[x] != 0):
   cdimensaoxt = True

 for y in range(0,len(yt)):
  if (yt[y] != 0):
   cdimensaoyt = True


 for z in range(0,len(zt)):
  if (zt[z] != 0):
   cdimensaozt = True

 if ((cdimensaoxt == cdimensaoxh) and (cdimensaoyt == cdimensaoyh)
and (cdimensaozt == cdimensaozh)):



  # Define maximum distance permissible

  dmaximo = (nlags + 0.5)*lagdistance

  # Define tolerances if they are zero

  if (htolerance == 0):
   htolerance= 45
  if (vtolerance == 0):
   vtolerance = 45
  if (hband == 0):
   hband = lagdistance/2
  if (vband == 0):
```

```
 vband = lagdistance/2
if (lagdistance == 0):
 lagdistance = 100
if (lineartolerance == 0):
 lineartolerance = lagdistance/2




# Convert tolerances in radians

htolerance = math.radians(htolerance)
vtolerance = math.radians(vtolerance)

# Determine dip and azimuth projections
htolerance = math.cos(htolerance)
vtolerance = math.cos(vtolerance)

# Define all euclidian distances permissible

cabeca = []
rabo = []
distanciah =[]
distanciaxy =[]
distanciax =[]
distanciay = []
 distanciaz = []
for t in range(0, len(yt)):
 for h in range(t, len(yh)):
  cabeca.append(vh[h])
  rabo.append(vt[t])
  dx = xh[h]-xt[t]
  dy= yh[h]-yt[t]
  dz = zh[h]-zt[t]
  if (distanciah > 0):
   distanciay.append(dy)
   distanciax.append(dx)
   distanciaz.append(dz)
   distanciah.append(math.sqrt(math.pow(dy,2) + math.pow(dx,2)
```

```
+ math.pow(dz,2)))
      distanciaxy.append(math.sqrt(math.pow(dy,2) + math.pow(dx,2)))

  # Calculate all cos and sin values

  cos_Azimute = math.cos(math.radians(90-Azimute))
  sin_Azimute = math.sin(math.radians(90-Azimute))
  sin_Dip = math.sin(math.radians(90-Dip))
  cos_Dip = math.cos(math.radians(90-Dip))

  v_valores_admissiveis_h =[]
  v_valores_admissiveis_t =[]


  # Calculate admissible points


  for l in range(0, nlags):
   valores_admissiveis_h = []
   valores_admissiveis_t = []
   lag= lagdistance*(l+1)
   for p in range(0,len(distanciah)):
    if (distanciah[p] < dmaximo):
     limitemin = lag - lineartolerance
     limitemax = lag + lineartolerance
     if (distanciah[p] > limitemin and distanciah[p] < limitemax):
      if (distanciaxy[p] > 0.000):
       check_azimute = (distanciax[p]*cos_Azimute
+distanciay[p]*sin_Azimute)/distanciaxy[p]
      else:
       check_azimute = 1
      check_azimute = math.fabs(check_azimute)
      if (check_azimute >= htolerance):
       check_bandh = (cos_Azimute*distanciay[p]) -
(sin_Azimute*distanciax[p])
       check_bandh = math.fabs(check_bandh)
       if (check_bandh < hband):
        if(distanciah[p] > 0.000):
         check_dip = (math.fabs(distanciaxy[p])*sin_Dip
```

```
+ distanciaz[p]*cos_Dip)/distanciah[p]
        else:
         check_dip = 0.000
        check_dip = math.fabs(check_dip)
        if (check_dip >= vtolerance):
         check_bandv = sin_Dip*distanciaz[p] -
cos_Dip*math.fabs(distanciaxy[p])
         check_bandv = math.fabs(check_bandv)
         if (check_bandv < vband):
          if (check_dip <0 and check_azimute <0):
           valores_admissiveis_h.append(rabo[p])
           valores_admissiveis_t.append(cabeca[p])
          else:
           valores_admissiveis_h.append(cabeca[p])
           valores_admissiveis_t.append(rabo[p])
   if (len(valores_admissiveis_h) > 0 and
len(valores_admissiveis_t) > 0):
    v_valores_admissiveis_h.append(valores_admissiveis_h)
    v_valores_admissiveis_t.append(valores_admissiveis_t)



  for i in range(0,len(v_valores_admissiveis_h)):
   vh = []
   vt = []
   vh = np.array(v_valores_admissiveis_h[i])
   vt = np.array(v_valores_admissiveis_t[i])
   arquivo2 = open("saida_hscatter.txt","w")
   arquivo2.write(str(i)+"\n")
   arquivo2.write(str(lagdistance)+"\n")
   for j in range(0,len(vh)):
    arquivo2.write(str(vh[j])+" "+str(vt[j])+"\n")
   arquivo2.close()
   os.system("python plot_hscatter.py")
```

```
    return True

  def finalize(self):
    print "Finalize program"
    return True
  def name(self):
    return "hscatter"
##############################################################
def get_plugins():
  return["hscatter"]
```

APÊNDICE B – Algoritmo do varmap

```
!/bin/python
import sgems
import matplotlib.pyplot as plt
from matplotlib import colors,ticker,cm
from scipy.interpolate import griddata
from scipy import interpolate
import numpy as np
import math
import random
import os



"""
VARIOMAP
"""

class    vario_map:
 def _init_(self):
  pass
 def initialize(self,params):
  self.params = params
  return True
 def execute(self):
  print self.params
  # ESTABELECA OS PARAMETROS INICIAIS
  head_property= self.params['head_prop']['property']
  head_grid = self.params['head_prop']['grid']
  tail_property = self.params['tail_prop']['property']
  tail_grid= self.params['tail_prop']['grid']
  C_variogram = int(self.params['C_variogram']['value'])
  C_covariance= int(self.params['C_covariance']['value'])
  nlags= int(self.params['nlags']['value'])
  lagdistance = float(self.params['lagdistance']['value'])
  lineartolerance= float(self.params['lineartolerance']['value'])
```

```
htolerance = float(self.params['htolangular']['value'])
vtolerance = float(self.params['vtolangular']['value'])
hband = float(self.params['hband']['value'])
vband = float(self.params['vband']['value'])
dip = float(self.params['Dip']['value'])
azimute = float(self.params['Azimute']['value'])
gdiscrete = int(self.params['Gdiscrete']['value'])
ncontour = int(self.params['ncontour']['value'])
Remove = float(self.params['Remove']['value'])

def retire_nan (x,y,z,v):
  xn =[]
  yn =[]
  zn =[]
  vn =[]
  for i in range(0,len(v)):
    if (v[i] > -99999999999999999):

      xn.append(x[i])
      yn.append(y[i])
      zn.append(z[i])
      vn.append(v[i])
  return xn, yn, zn, vn



xh = []
yh = []
zh = []
vh = []

xh1 = sgems.get_property(head_grid, "_X_")
yh1 = sgems.get_property(head_grid, "_Y_")
zh1 = sgems.get_property(head_grid, "_Z_")
vh1 = sgems.get_property(head_grid, head_property)

xh, yh, zh, vh = retire_nan(xh1,yh1,zh1,vh1)
```

```
xt = []
yt = []
zt = []
vt = []

xt1 = sgems.get_property(tail_grid, "_X_")
yt1 = sgems.get_property(tail_grid, "_Y_")
zt1 = sgems.get_property(tail_grid, "_Z_")
vt1 = sgems.get_property(tail_grid, tail_property)

xt, yt, zt, vt = retire_nan(xt1,yt1,zt1,vt1)

# VERIFIQUE O NUMERO DE DIMENSOES DO PROBLEMA

# VERIFICACAO DAS DIMENSOES DO HEAD

cdimensaoxh = False
cdimensaoyh = False
cdimensaozh = False

for x in range(0,len(xh)):
  if (xh[x] != 0):
    cdimensaoxh = True


for y in range(0,len(yh)):
  if (yh[y] != 0):
    cdimensaoyh = True


for z in range(0,len(zh)):
  if (zh[z] != 0):
    cdimensaozh = True

# VERIFICACAO DAS DIMENSOES DO TAIL

cdimensaoxt = False
cdimensaoyt = False
cdimensaozt = False
```

```
  for x in range(0,len(xt)):
    if (xt[x] != 0):
      cdimensaoxt = True

  for y in range(0,len(yt)):
    if (yt[y] != 0):
      cdimensaoyt = True


  for z in range(0,len(zt)):
    if (zt[z] != 0):
      cdimensaozt = True

  if ((cdimensaoxt == cdimensaoxh) and (cdimensaoyt == cdimensaoyh)
and (cdimensaozt == cdimensaozh)):



    # DEFINA A MAXIMA DISTANCIA PERMISSIVEL

    dmaximo = (nlags + 0.5)*lagdistance

    #CONVERTA TOLERANCIAS SE ELAS FOREM IGUAIS A ZERO
    if (htolerance == 0):
      htolerance= 45
    if (vtolerance == 0):
      vtolerance = 45
    if (hband == 0):
      hband = lagdistance/2
    if (vband == 0):
      vband = lagdistance/2
    if (lagdistance == 0):
      lagdistance = 100
    if (lineartolerance == 0):
      lineartolerance = lagdistance/2

    #CONVERTA OS VALORES DE TOLERANCIA EM RADIANOS
    htolerance = math.radians(htolerance)
```

```
    vtolerance = math.radians(vtolerance)

    # DETERMINE AS PROJECOES DO DIP E DO AZIMUTE
    htolerance = math.cos(htolerance)
    vtolerance = math.cos(vtolerance)

    # TRANSFORME O DIP EM VALOR NEGATIVO PARA A ROTACAO DO PLANO
    dip = - dip

    xhrot =[]
    yhrot =[]
    zhrot =[]
    xttrot =[]
    yttrot=[]
    zttrot=[]

    for i in range(0, len(xh)):

     # ROTACIONE PRIMEIRAMENTE NO AZIMUTE

      xrot = math.cos(math.radians(azimute))*xh[i] -
math.sin(math.radians(azimute))*yh[i]
      yrot = math.sin(math.radians(azimute))*xh[i] +
math.cos(math.radians(azimute))*yh[i]
      yrot2 = math.cos(math.radians(dip))*yrot -
math.sin(math.radians(dip))*zh[i]
      zrot = math.sin(math.radians(dip))*yrot +
math.cos(math.radians(dip))*zh[i]

      xhrot.append(xrot)
      yhrot.append(yrot2)
      zhrot.append(zrot)

      # ROTACIONE EM SEGUIDA NO MERGULHO

      xtrot = math.cos(math.radians(azimute))*xt[i] -
math.sin(math.radians(azimute))*yt[i]
      ytrot = math.sin(math.radians(azimute))*xt[i] +
math.cos(math.radians(azimute))*yt[i]
```

```
      ytrot2 = math.cos(math.radians(dip))*ytrot -
math.sin(math.radians(dip))*zt[i]
      ztrot = math.sin(math.radians(dip))*ytrot +
math.cos(math.radians(dip))*zt[i]

      xttrot.append(xtrot)
      yttrot.append(ytrot2)
      zttrot.append(ztrot)


    #CONVERTA OS VALORES DE TOLERANCIA EM RADIANOS
    htolerance = math.radians(htolerance)
    vtolerance = math.radians(vtolerance)

    # DETERMINE AS PROJECOES DO DIP E DO AZIMUTE
    htolerance = math.cos(htolerance)
    vtolerance = math.cos(vtolerance)

    # DEFINA TODAS AS DISTANCIAS EUCLIDIANAS POSSIVEIS

    cabeca = []
    rabo = []
    cabeca2 =[]
    rabo2 =[]
    distanciah =[]
    distanciaxy =[]
    distanciax =[]
    distanciay = []
    distanciaz = []
    for t in range(0, len(yt)):
     for h in range(t, len(yh)):


       dx = xhrot[h]-xttrot[t]
       dy= yhrot[h]-yttrot[t]
       dz = zhrot[h]-zttrot[t]
       dh = math.sqrt(math.pow(dy,2) + math.pow(dx,2) +
math.pow(dz,2))
```

```
        if (dh <dmaximo):
          cabeca.append(vh[h])
          cabeca2.append(vt[h])
          rabo2.append(vt[t])
          rabo.append(vh[t])

          distanciay.append(dy)
          distanciax.append(dx)
          distanciaz.append(dz)
          distanciah.append(dh)
          distanciaxy.append(math.sqrt(math.pow(dy,2) +
math.pow(dx,2)))

    # CALCULE TODOS OS VALORES DE COS E SENO ADMISSIVEIS AO AZIMUTE

    cos_Azimute = []
    sin_Azimute = []
    flutuante_d = 0


    for a in range(0,360,20):
      diferenca = a
      cos_Azimute.append(math.cos(math.radians(90-diferenca)))
      sin_Azimute.append(math.sin(math.radians(90-diferenca)))
      cos_Dip = math.cos(math.radians(90))
      sin_Dip = math.sin(math.radians(90))

    distancias_admissiveis = []
    azimute_admissiveis = []
    dip_admissiveis =[]
    v_valores_admissiveis_h =[]
    v_valores_admissiveis_t =[]
    v_valores_admissiveis_h2 =[]
    v_valores_admissiveis_t2 =[]
    pares = []



    # CALCULE OS PONTOS ADMISSIVEIS
```

```
    for a in xrange(0,18):
      # reestabelca o valor do norte retirando novamente o azimute
      azm = math.radians(a*20)
      for l in xrange(0, nlags):
        valores_admissiveis_h = []
        valores_admissiveis_t = []
        valores_admissiveis_h2 =[]
        valores_admissiveis_t2 =[]
        distancia_admissivel = []
        azimute_admissivel =[]
        dip_admissivel=[]
        lag= lagdistance*(l+1)
        par= 0
        limitemin = lag - lineartolerance
        limitemax = lag + lineartolerance
        for p in xrange(0,len(distanciah)):
          if (distanciah[p] > limitemin and distanciah[p] < limitemax):
            if (distanciaxy[p] > 0.000):
              check_azimute = (distanciax[p]*cos_Azimute[a]
+distanciay[p]*sin_Azimute[a])/distanciaxy[p]
            else:
              check_azimute = 1
            check_azimute = math.fabs(check_azimute)
            if (check_azimute >= htolerance):
              check_bandh = (cos_Azimute[a]*distanciay[p])
- (sin_Azimute[a]*distanciax[p])
              check_bandh = math.fabs(check_bandh)
              if (check_bandh < hband):
                if(distanciah[p] > 0.000):
                  check_dip = (math.fabs(distanciaxy[p])*sin_Dip
+ distanciaz[p]*cos_Dip)/distanciah[p]
                else:
                  check_dip = 0.000
                check_dip = math.fabs(check_dip)
                if (check_dip >= vtolerance):
                  check_bandv = sin_Dip*distanciaz[p]
- cos_Dip*math.fabs(distanciaxy[p])
                  check_bandv = math.fabs(check_bandv)
                  if (check_bandv < vband):
```

```
                        valores_admissiveis_h.append(cabeca[p])
                        valores_admissiveis_t.append(rabo[p])
                        valores_admissiveis_h2.append(cabeca2[p])
                        valores_admissiveis_t2.append(rabo2[p])
                        distancia_admissivel.append(distanciah[p])
                        par = par + 1
                        azimute_admissivel.append(azm)
        if (len(valores_admissiveis_h) > 0 and
len(valores_admissiveis_t) > 0):
          if (par > 0):
            v_valores_admissiveis_h.append(valores_admissiveis_h)
            v_valores_admissiveis_h2.append(valores_admissiveis_h2)
            v_valores_admissiveis_t2.append(valores_admissiveis_t2)
            v_valores_admissiveis_t.append(valores_admissiveis_t)
            distancias_admissiveis.append(distancia_admissivel)
            pares.append(par)
            azimute_admissiveis.append(azimute_admissivel)


    # CALCULE AS FUNCOES DE CONTINUIDADE ESPACIAL SEGUNDO
OS VALORES ADMISSIVEIS

    # CALCULE O VARIOGRAMA

    if (C_variogram == 1):
      continuidade =[]
      lag_adm =[]
      azimute_adm =[]
      for i in xrange(0,len(v_valores_admissiveis_h)):
        flutuantet =[]
        flutuanteh =[]
        flutuanteh2 = []
        flutuantet2 =[]
        flutuanted =[]
        flutuantea=[]
        flutuantedip=[]
        par_var = 0
        flutuanteh = v_valores_admissiveis_h[i][:]
        flutuanteh2= v_valores_admissiveis_h2[i][:]
```

```
        flutuantet = v_valores_admissiveis_t[i][:]
        flutuantet2 = v_valores_admissiveis_t2[i][:]
        flutuanted = distancias_admissiveis[i][:]
        flutuantea= azimute_admissiveis[i][:]
        par_var= pares[i]
        soma = 0
        lagmedio =0
        agmedio =0
        dgmedio = 0
        for j in xrange(0, len(flutuanteh)):
          soma = soma + (flutuanteh[j] - flutuantet[j])*
(flutuanteh2[j]-flutuantet2[j])/(2*pares[i])
        for z in xrange(0, len(flutuanted)):
          lagmedio = lagmedio + flutuanted[z]/len(flutuanted)
        for g in xrange(0, len(flutuantea)):
          agmedio = agmedio + flutuantea[g]/len(flutuantea)
        if (soma <= Remove):
          continuidade.append(soma)
          azimute_adm.append(agmedio)
          lag_adm.append(lagmedio)

    # CALCULE O COVARIOGRAMA
    if (C_covariance == 1):
      continuidade =[]
      lag_adm =[]
      azimute_adm =[]
      for i in xrange(0,len(v_valores_admissiveis_h)):
        flutuantet =[]
        flutuanteh =[]
        flutuanted =[]
        flutuantea=[]
        par_var = 0
        flutuanteh = v_valores_admissiveis_h[i]
        flutuantet = v_valores_admissiveis_t[i]
        flutuanted = distancias_admissiveis[i]
        flutuantea= azimute_admissiveis[i]
        par_var= pares[i]
        soma = 0
        lagmedio =0
```

```
        agmedio =0
        dgmedio = 0
        somah = 0
        somat = 0
        mediah = 0
        mediat =0
        for d in range (0, len(flutuanteh)):
          somah = somah + flutuanteh[d]
        mediah = float(somah/len(flutuanteh))
        for t in range (0, len(flutuantet)):
          somat = somat + flutuantet[t]
        mediat = float(somat/len(flutuantet))
        for j in xrange(0, len(flutuanteh)):
            soma = soma + float(((flutuanteh[j] - mediah)*
(flutuantet[j] - mediat))/(par_var))
        for z in xrange(0, len(flutuanted)):
          lagmedio = lagmedio + flutuanted[z]/len(flutuanted)
        for g in xrange(0, len(flutuantea)):
          agmedio = agmedio + flutuantea[g]/len(flutuantea)
        if (soma <= Remove):
          continuidade.append(soma)
          azimute_adm.append(agmedio)
          lag_adm.append(lagmedio)
    print ("OK")

    # RECORD FILE


    arquivo2 = open(".saida_varmap.txt","w")
    arquivo2.write(str(gdiscrete)+"\n")
    arquivo2.write(str(ncontour)+"\n")
    for i in range(0, len(azimute_adm)):
      arquivo2.write(str(azimute_adm[i])+"  "+str(lag_adm[i])
+ "  "+str(continuidade[i]) + "\n")
    arquivo2.close()


    # PLOTE O MAPA DE VARIOGRAMAS INTERPOLADO
```

```
    os.system("python plot_varmap.py")

 def finalize(self):
  print "Finalize program"
  return True
 def name(self):
  return "vario_map"
#####################################################
def get_plugins():
 return["vario_map"]
```

APÊNDICE C – Algoritmo dos variogramas experimentais

```
#!/bin/python
import sgems
import matplotlib.pyplot as plt
from matplotlib import colors,ticker,cm
from scipy.interpolate import griddata
from scipy import interpolate
import numpy as np
import math
import random


#.............................................................
def retire_nan (x,y,z,v):
 xn =[]
 yn =[]
 zn =[]
 vn =[]
 for i in range(0,len(v)):
  if (v[i] > -99999999999999999):

   xn.append(x[i])
   yn.append(y[i])
   zn.append(z[i])
   vn.append(v[i])
 return xn, yn, zn, vn
#.............................................................




class    variogram:
    def _init_(self):
        pass
```

```
    def initialize(self,params):
        self.params = params
        return True
    def execute(self):
print(self.params)

'''
EXPERIMENTAL VARIOGRAMS CALCULATION
-----------------------------------------------------------------------

This is a template for SGems program for calculating experimental
variograms.The user have to fill the parameters with a ui.file
with same name of this python file before start program.
The QT userinterface connect SGems data with this routine
using the class params.

The output of this program is two files, one relatory with all
experimental points used in Automatic fitting procedure and other
with a hml file used to import experimental variograms in SGems

AUTHOR: DAVID ALVARENGA DRUMOND  2016

'''

'''
INITIAL PARAMETERS
-----------------------------------------------------------------------

All initial parameters are transcribe by ui.file interface.
Variables are choose among common variables. The follow variables are:

COMMON VARIABLES:

head_property= adress of head property
head_grid = adress of head grid
tail_property = adress of tail property
tail_grid= adress of tail grid
C_variogram = check for variogram function calculation
C_covariance= check for covariance function calculation
```

```
C_relative_variogram = check for relative variogram calculation
C_madogram = check for madogram calculation
C_correlogram = check for correlogram calculation
C_PairWise = check for pairwise calcluation
nlags= number of lags in experimental variogram
lagdistance = lenght of lag in experimental variogram
lineartolerance= linear tolerance of experimental variogram
htolerance = horizontal angular tolerance
vtolerance = vertical angular tolerance
hband = horizontal bandwidth
vband = vertical bandwidth
nAzimuths = number of azimuths
NDips = number of dips
Azimth_diference = angular diference between azimuths
Dip_difference = angular diference between dips
sAzimuth = initial azimuth value
sDip = initial dip value
inv = choose of invert correlogram axis
save = save adress of relatory file
save2 = save adress of Sgems variogram file
nhead = number of head to print in relatory file
ntail = number of tail to print in relatory file


'''

# Stabilish initial parameters

head_property= self.params['head_prop']['property']
head_grid = self.params['head_prop']['grid']
tail_property = self.params['tail_prop']['property']
tail_grid= self.params['tail_prop']['grid']
C_variogram = int(self.params['C_variogram']['value'])
C_covariance= int(self.params['C_covariance']['value'])
C_relative_variogram = int(self.params['C_relative_variogram']['value'])
C_madogram = int(self.params['C_madogram']['value'])
C_correlogram = int(self.params['C_correlogram']['value'])
C_PairWise = int(self.params['C_PairWise']['value'])
nlags= int(self.params['nlags']['value'])
```

```
lagdistance = float(self.params['lagdistance']['value'])
lineartolerance= float(self.params['lineartolerance']['value'])
htolerance = float(self.params['htolangular']['value'])
vtolerance = float(self.params['vtolangular']['value'])
hband = float(self.params['hband']['value'])
vband = float(self.params['vband']['value'])
nAzimuths = int(self.params['nAzimuths']['value'])
NDips = int(self.params['NDips']['value'])
Azimth_diference = float(self.params['Azimth_diference']['value'])
Dip_difference = float(self.params['Dip_difference']['value'])
sAzimuth = float(self.params['sAzimuth']['value'])
sDip = float(self.params['sDip']['value'])
inv = int(self.params['Inv']['value'])
save = self.params['Save']['value']
save2 = self.params['Save2']['value']
nhead = self.params['NHEAD']['value']
ntail = self.params['NTAIL']['value']

# Obtain properties in SGems


xh = []
yh = []
zh = []
vh = []

xh1 = sgems.get_property(head_grid, "_X_")
yh1 = sgems.get_property(head_grid, "_Y_")
zh1 = sgems.get_property(head_grid, "_Z_")
vh1 = sgems.get_property(head_grid, head_property)

xh, yh, zh, vh = retire_nan(xh1,yh1,zh1,vh1)


xt = []
yt = []
zt = []
vt = []
```

```
xt1 = sgems.get_property(tail_grid, "_X_")
yt1 = sgems.get_property(tail_grid, "_Y_")
zt1 = sgems.get_property(tail_grid, "_Z_")
vt1 = sgems.get_property(tail_grid, tail_property)

xt, yt, zt, vt = retire_nan(xt1,yt1,zt1,vt1)

# Verify number of dimensions in the problem

# Verify number of dimensions in head

cdimensaoxh = False
cdimensaoyh = False
cdimensaozh = False

for x in range(0,len(xh)):
 if (xh[x] != 0):
  cdimensaoxh = True


for y in range(0,len(yh)):
 if (yh[y] != 0):
  cdimensaoyh = True


for z in range(0,len(zh)):
 if (zh[z] != 0):
  cdimensaozh = True

# Verify number of dimensions in tail

cdimensaoxt = False
cdimensaoyt = False
cdimensaozt = False

for x in range(0,len(xt)):
 if (xt[x] != 0):
  cdimensaoxt = True
```

```
 for y in range(0,len(yt)):
  if (yt[y] != 0):
   cdimensaoyt = True


 for z in range(0,len(zt)):
  if (zt[z] != 0):
   cdimensaozt = True

 if ((cdimensaoxt == cdimensaoxh) and (cdimensaoyt == cdimensaoyh)
and (cdimensaozt == cdimensaozh)):



  # Define maximum permissible distance

  dmaximo = (nlags + 0.5)*lagdistance

  #Transform tolerances if they are zero

  if (htolerance == 0):
   htolerance= 45
  if (vtolerance == 0):
   vtolerance = 45
  if (hband == 0):
   hband = lagdistance/2
  if (vband == 0):
   vband = lagdistance/2
  if (lagdistance == 0):
   lagdistance = 100
  if (lineartolerance == 0):
   lineartolerance = lagdistance/2


  #Convert tolerances in radians
  htolerance = math.radians(htolerance)
  vtolerance = math.radians(vtolerance)

  # Determine projections of Dip and Azimuth
```

```
  htolerance = math.cos(htolerance)
  vtolerance = math.cos(vtolerance)

  # Define all direction distances

  cabeca = []
  rabo = []
  cabeca2 =[]
  rabo2 = []
  distanciah =[]
  distanciaxy =[]
  distanciax =[]
  distanciay = []
   distanciaz = []
  for t in range(0, len(yt)):
   for h in range(t, len(yh)):
    cabeca.append(vh[h])
    rabo.append(vh[t])
    cabeca2.append(vt[h])
    rabo2.append(vt[t])
    dx = xh[h]-xt[t]
    dy= yh[h]-yt[t]
    dz = zh[h]-zt[t]
    if (distanciah > 0):
     distanciay.append(dy)
     distanciax.append(dx)
     distanciaz.append(dz)
     distanciah.append(math.sqrt(math.pow(dy,2)
+ math.pow(dx,2) + math.pow(dz,2)))
     distanciaxy.append(math.sqrt(math.pow(dy,2) + math.pow(dx,2)))

  # Calculate all sin and cosine functions of Dip and Azimuth



  azimute = []
  fAzimuth = float(Azimth_diference*(nAzimuths-1)) + sAzimuth
  azimute = np.linspace(sAzimuth,fAzimuth,nAzimuths)
```

```
dip = []
fDip = Dip_difference*(NDips-1) + sDip
dip = np.linspace(sDip,fDip,NDips)


cos_Azimute=[]
sin_Azimute=[]
for a in azimute:
 cos_Azimute.append(math.cos(math.radians(90-a)))
 sin_Azimute.append(math.sin(math.radians(90-a)))
cos_Dip =[]
sin_Dip =[]
for a in dip:
 cos_Dip.append(math.cos(math.radians(90-a)))
 sin_Dip.append(math.sin(math.radians(90-a)))




distancias_admissiveis = []
azimute_admissiveis = []
dip_admissiveis =[]
v_valores_admissiveis_h =[]
v_valores_admissiveis_t =[]
v_valores_admissiveis_h2 =[]
v_valores_admissiveis_t2 =[]
pares = []

# Calculate admissible pairs
for a in xrange(0,len(azimute)):
 azm = azimute[a]
 for d in xrange(0, len(dip)):
  dipv = dip[d]
  for l in range(0, nlags):
   valores_admissiveis_h = []
   valores_admissiveis_t = []
   valores_admissiveis_h2 =[]
   valores_admissiveis_t2 =[]
```

```
      distancia_admissivel = []
      azimute_admissivel =[]
      dip_admissivel=[]
      lag= lagdistance*(l+1)
      par= 0
      for p in xrange(0,len(distanciah)):
       if (distanciah[p] < dmaximo):
        limitemin = lag - lineartolerance
        limitemax = lag + lineartolerance
        if (distanciah[p] > limitemin and distanciah[p] < limitemax):
         if (distanciaxy[p] > 0.000):
          check_azimute = (distanciax[p]*cos_Azimute[a]
+distanciay[p]*sin_Azimute[a])/distanciaxy[p]
         else:
          check_azimute = 1
         check_azimute = math.fabs(check_azimute)
         if (check_azimute >= htolerance):
          check_bandh = (cos_Azimute[a]*distanciay[p])
- (sin_Azimute[a]*distanciax[p])
          check_bandh = math.fabs(check_bandh)
          if (check_bandh < hband):
           if(distanciah[p] > 0.000):
            check_dip = (math.fabs(distanciaxy[p])*sin_Dip[d]
+ distanciaz[p]*cos_Dip[d])/distanciah[p]
           else:
            check_dip = 0.000
           check_dip = math.fabs(check_dip)
           if (check_dip >= vtolerance):
            check_bandv = sin_Dip[d]*distanciaz[p]
- cos_Dip[d]*math.fabs(distanciaxy[p])
            check_bandv = math.fabs(check_bandv)
            if (check_bandv < vband):
             valores_admissiveis_h.append(rabo[p])
             valores_admissiveis_t.append(cabeca[p])
             valores_admissiveis_h2.append(rabo2[p])
             valores_admissiveis_t2.append(cabeca2[p])
             distancia_admissivel.append(distanciah[p])
             par = par + 1
             azimute_admissivel.append(azm)
```

```
                dip_admissivel.append(dipv)
        if (len(valores_admissiveis_h) > 0 and len(valores_admissiveis_t) >
         v_valores_admissiveis_h.append(valores_admissiveis_h)
         v_valores_admissiveis_h2.append(valores_admissiveis_h2)
         v_valores_admissiveis_t2.append(valores_admissiveis_t2)
         v_valores_admissiveis_t.append(valores_admissiveis_t)
         distancias_admissiveis.append(distancia_admissivel)
         pares.append(par)
         azimute_admissiveis.append(azimute_admissivel)
         dip_admissiveis.append(dip_admissivel)



# Calculate continuity functions

# Variogram

if (C_variogram == 1):
 continuidade =[]
 lag_adm =[]
 azimute_adm =[]
 dip_adm =[]
 for i in range(0,len(v_valores_admissiveis_h)):
  flutuantet =[]
  flutuanteh =[]
  flutuanteh2 = []
  flutuantet2 =[]
  flutuanted =[]
  flutuantea=[]
  flutuantedip=[]
  par_var = 0
  flutuanteh = v_valores_admissiveis_h[i][:]
  flutuanteh2= v_valores_admissiveis_h2[i][:]
  flutuantet = v_valores_admissiveis_t[i][:]
  flutuantet2 = v_valores_admissiveis_t2[i][:]
  flutuanted = distancias_admissiveis[i][:]
  flutuantea= azimute_admissiveis[i][:]
  flutuantedip = dip_admissiveis[i][:]
  par_var= pares[i]
```

```
    soma = 0
    lagmedio =0
    agmedio =0
    dgmedio = 0
    for j in range(0, len(flutuanteh)):
     soma = soma + (flutuanteh[j] - flutuantet[j])*
(flutuanteh2[j]-flutuantet2[j])/(2*pares[i])
    continuidade.append(soma)
    for z in range(0, len(flutuanted)):
     lagmedio = lagmedio + flutuanted[z]/len(flutuanted)
    lag_adm.append(lagmedio)
    for g in range(0, len(flutuantea)):
     agmedio = agmedio + flutuantea[g]/len(flutuantea)
    azimute_adm.append(agmedio)
    for t in range(0, len(flutuantedip)):
     dgmedio = dgmedio + flutuantedip[t]/len(flutuantedip)
    dip_adm.append(dgmedio)



  # Covariogram
  if (C_covariance == 1):
   continuidade =[]
   lag_adm =[]
   azimute_adm =[]
   dip_adm =[]
   for i in range(0,len(v_valores_admissiveis_h)):
    flutuantet =[]
    flutuanteh =[]
    flutuanted =[]
    flutuantea=[]
    flutuantedip=[]
    par_var = 0
    flutuanteh = v_valores_admissiveis_h[i]
    flutuantet = v_valores_admissiveis_t[i]
    flutuanted = distancias_admissiveis[i]
    flutuantea= azimute_admissiveis[i]
    flutuantedip = dip_admissiveis[i]
    par_var= pares[i]
```

```
    soma = 0
    lagmedio =0
    agmedio =0
    dgmedio = 0
    somah = 0
    somat = 0
    mediah = 0
    mediat =0
    for d in range (0, len(flutuanteh)):
     somah = somah + flutuanteh[d]
    mediah = float(somah/len(flutuanteh))
    for t in range (0, len(flutuantet)):
     somat = somat + flutuantet[t]
    mediat = float(somat/len(flutuantet))
    for j in range(0, len(flutuanteh)):
     soma = soma + float(((flutuanteh[j] - mediah)*
(flutuantet[j] - mediat))/(par_var))
    continuidade.append(soma)
    for z in range(0, len(flutuanted)):
     lagmedio = lagmedio + flutuanted[z]/len(flutuanted)
    lag_adm.append(lagmedio)
    for g in range(0, len(flutuantea)):
     agmedio = agmedio + flutuantea[g]/len(flutuantea)
    azimute_adm.append(agmedio)
    for y in range(0, len(flutuantedip)):
     dgmedio = dgmedio + flutuantedip[y]/len(flutuantedip)
    dip_adm.append(dgmedio)

  # Correlogram
  if (C_correlogram == 1):
   continuidade =[]
   lag_adm =[]
   azimute_adm =[]
   dip_adm =[]
   for i in range(0,len(v_valores_admissiveis_h)):
    flutuantet =[]
    flutuanteh =[]
    flutuanted =[]
    flutuantea=[]
```

```
    flutuantedip=[]
    par_var = 0
    flutuanteh = v_valores_admissiveis_h[i]
    flutuantet = v_valores_admissiveis_t[i]
    flutuanted = distancias_admissiveis[i]
    flutuantea= azimute_admissiveis[i]
    flutuantedip = dip_admissiveis[i]
    par_var= pares[i]
    soma = 0
    lagmedio =0
    agmedio =0
    dgmedio = 0
    somah = 0
    somat = 0
    diferencah =0
    diferencat =0
    mediah = 0
    mediat =0
    varh = 0
    vart = 0
    for d in range (0, len(flutuanteh)):
     somah = somah + flutuanteh[d]
    mediah = float(somah/len(flutuanteh))
    for t in range (0, len(flutuantet)):
     somat = somat + flutuantet[t]
    mediat = float(somat/len(flutuantet))
    for u in range(0, len(flutuanteh)):
     diferencah = flutuanteh[u]**2 + diferencah
    varh = math.sqrt(float(diferencah)/len(flutuanteh)-mediah**2)
    for m in range(0, len(flutuanteh)):
     diferencat = flutuantet[m]**2 + diferencat
    vart = math.sqrt(float(diferencat)/len(flutuantet)-mediat**2)
    for j in range(0, len(flutuanteh)):
      if (vart > 0 and varh >0 ):
       soma = soma + (flutuanteh[j] - mediah)*
(flutuantet[j] - mediat)/(par_var*vart*varh)
    if (inv ==1):
     continuidade.append(1-soma)
    else:
```

```
      continuidade.append(soma)
     for z in range(0, len(flutuanted)):
      lagmedio = lagmedio + flutuanted[z]/len(flutuanted)
     lag_adm.append(lagmedio)
     for g in range(0, len(flutuantea)):
      agmedio = agmedio + flutuantea[g]/len(flutuantea)
     azimute_adm.append(agmedio)
     for y in range(0, len(flutuantedip)):
      dgmedio = dgmedio + flutuantedip[y]/len(flutuantedip)
     dip_adm.append(dgmedio)

  # PairWise
  if (C_PairWise == 1):
   continuidade =[]
   lag_adm =[]
   azimute_adm =[]
   dip_adm =[]
   for i in range(0,len(v_valores_admissiveis_h)):
    flutuantet =[]
    flutuanteh =[]
    flutuanted =[]
    flutuantea=[]
    flutuantedip=[]
    par_var = 0
    flutuanteh = v_valores_admissiveis_h[i][:]
    flutuantet = v_valores_admissiveis_t[i][:]
    flutuanted = distancias_admissiveis[i][:]
    flutuantea= azimute_admissiveis[i][:]
    flutuantedip = dip_admissiveis[i][:]
    par_var= pares[i]
    soma = 0
    lagmedio =0
    agmedio =0
    dgmedio = 0
    for j in range(0, len(flutuanteh)):
     if (flutuanteh[j] + flutuantet[j] > 0):
      s = ((flutuanteh[j] - flutuantet[j])/
(flutuanteh[j]+flutuantet[j]))**2
      soma = soma + 2*s/(1.0*par_var)
```

```
    continuidade.append(soma)
    for z in range(0, len(flutuanted)):
     lagmedio = lagmedio + flutuanted[z]/len(flutuanted)
    lag_adm.append(lagmedio)
    for g in range(0, len(flutuantea)):
     agmedio = agmedio + flutuantea[g]/len(flutuantea)
    azimute_adm.append(agmedio)
    for t in range(0, len(flutuantedip)):
     dgmedio = dgmedio + flutuantedip[t]/len(flutuantedip)
    dip_adm.append(dgmedio)


# Relative Variogram
if (C_relative_variogram == 1):
 continuidade =[]
 lag_adm =[]
 azimute_adm =[]
 dip_adm =[]
 for i in range(0,len(v_valores_admissiveis_h)):
  flutuantet =[]
  flutuanteh =[]
  flutuanteh2 = []
  flutuantet2 =[]
  flutuanted =[]
  flutuantea=[]
  flutuantedip=[]
  par_var = 0
  flutuanteh = v_valores_admissiveis_h[i]
  flutuantet = v_valores_admissiveis_t[i]
  flutuanteh2 = v_valores_admissiveis_h2[i]
  flutuantet2 = v_valores_admissiveis_t2[i]
  flutuanted = distancias_admissiveis[i]
  flutuantea= azimute_admissiveis[i]
  flutuantedip = dip_admissiveis[i]
  par_var= pares[i]
  soma = 0
  lagmedio =0
  agmedio =0
  dgmedio = 0
```

```
    somah = 0
    somat = 0
    mediah = 0
    mediat =0
    for d in range (0, len(flutuanteh)):
     somah = somah + flutuanteh[d]
    mediah = float(somah/len(flutuanteh))
    for t in range (0, len(flutuantet)):
     somat = somat + flutuantet[t]
    mediat = float(somat/len(flutuantet))
    for j in range(0, len(flutuanteh)):
      soma = soma + float((flutuanteh[j] -flutuantet2[j])*
(flutuanteh2[j] - flutuantet2[j])/(par_var*(mediat+mediah)**2/4))
    continuidade.append(soma)
    for z in range(0, len(flutuanted)):
     lagmedio = lagmedio + flutuanted[z]/len(flutuanted)
    lag_adm.append(lagmedio)
    for g in range(0, len(flutuantea)):
     agmedio = agmedio + flutuantea[g]/len(flutuantea)
    azimute_adm.append(agmedio)
    for y in range(0, len(flutuantedip)):
     dgmedio = dgmedio + flutuantedip[y]/len(flutuantedip)
    dip_adm.append(dgmedio)

  # Madogram
  if (C_madogram == 1):
   continuidade =[]
   lag_adm =[]
   azimute_adm =[]
   dip_adm =[]
   for i in range(0,len(v_valores_admissiveis_h)):
    flutuantet =[]
    flutuanteh =[]
    flutuanted =[]
    flutuantea=[]
    flutuantedip=[]
    par_var = 0
    flutuanteh = v_valores_admissiveis_h[i]
    flutuantet = v_valores_admissiveis_t[i]
```

```
    flutuanted = distancias_admissiveis[i]
    flutuantea= azimute_admissiveis[i]
    flutuantedip = dip_admissiveis[i]
    par_var= pares[i]
    soma = 0
    lagmedio =0
    agmedio =0
    dgmedio = 0
    for j in range(0, len(flutuanteh)):
      soma = soma + float(math.fabs(flutuanteh[j] - flutuantet[j])
/(2*par_var))
    continuidade.append(soma)
    for z in range(0, len(flutuanted)):
     lagmedio = lagmedio + flutuanted[z]/len(flutuanted)
    lag_adm.append(lagmedio)
    for g in range(0, len(flutuantea)):
     agmedio = agmedio + flutuantea[g]/len(flutuantea)
    azimute_adm.append(agmedio)
    for t in range(0, len(flutuantedip)):
     dgmedio = dgmedio + flutuantedip[t]/len(flutuantedip)
    dip_adm.append(dgmedio)


  # Write experimental variograms in relatory


  save= open(save,"a")
  save.write(nhead + "\n")
  save.write(ntail + "\n")
  save.write(" Azimuth Dip lag variogram pairs \n")
  for i in range(0, len(continuidade)):
   save.write(str(azimute_adm[i]) + " " +str(dip_adm[i]) + " "
+ str(lag_adm[i]) + " " + str(continuidade[i])+ " " + str(pares[i]) + "\n

  # Separate experimental variograms according prescribe direction

  posicao =[]
  posicao.append(-1)
  for i in range(1, len(azimute_adm)):
```

```
    if (round(azimute_adm[i],2) != round(azimute_adm[i-1],2) or
round(dip_adm[i],2) != round(dip_adm[i-1],2)):
     posicao.append(i-1)
  posicao.append(len(azimute_adm)-1)

  azimutes = []
  dips = []
  lags =[]
  continuidades =[]
  paresv =[]

  for i in range(1,len(posicao)):
   azimutes.append(azimute_adm[posicao[i-1]+1:posicao[i]])
   dips.append(dip_adm[posicao[i-1]+1:posicao[i]])
   lags.append(lag_adm[posicao[i-1]+1:posicao[i]])
   continuidades.append(continuidade[posicao[i-1]+1:posicao[i]])
   paresv.append(pares[posicao[i-1]+1:posicao[i]])

  # Write hml file for experimental variograms

  save2= open(save2,"w")
  save2.write("    <experimental_variograms> \n")

  for i in range(0, len(azimutes)):
   azim = []
   Dip =[]
   lagv =[]
   cont =[]
   par =[]
   azim = azimutes[i]
   Dip = dips[i]
   lagv = lags[i]
   cont =continuidades[i]
   par = paresv[i]
   save2.write("  <variogram> \n")
   save2.write("  <title>variogram -azth="+
str(round(azim[0],2))+", dip="+str(round(Dip[0],2))+"</title> \n")
   direction1 = math.cos(math.radians(Dip[0]))*math.sin(math.radians(azim
   direction2 = math.cos(math.radians(Dip[0]))*math.cos(math.radians(azim
```

```
    direction3 = -math.sin(math.radians(Dip[0]))
    save2.write("<direction>" +str(direction1) + " "
+ str(direction2) +" "+ str(direction3)+ "</direction> \n")
    save2.write("    <x>")
    for j in range(0,len(lagv)):
     save2.write(str(lagv[j]) + " ")
    save2.write("</x> \n")
    save2.write("    <y>")
    for p in range(0,len(cont)):
     save2.write(str(cont[p])+ " ")
    save2.write("</y> \n")
    save2.write("    <pairs>")
    for h in range(0,len(cont)):
     save2.write(str(par[h])+ " ")
    save2.write("</pairs> \n")
    save2.write("  </variogram> \n")
   save2.write("    </experimental_variograms> \n")
   save2.close()



  print("Finish")

      return True

     def finalize(self):
         print "Finalize program"
         return True
     def name(self):
         return "variogram"
##################################################
def get_plugins():
     return["variogram"]
```

APÊNDICE D – Algoritmo da modelagem automática

```
#!/bin/python
import sgems
import matplotlib.pyplot as plt
import numpy as np
import math
import random
import os



"""
VARIOGRAM OPTIMIZATION
"""

class    automatic_fitting2:
 def _init_(self):
  caminho = "DADOS"
  numero_variogramas = 1
  numero_estruturas = 1
  azimute_direcao_principal = 0
  dip_direcao_principal = 0
  numero_interacoes = 0
  pass
 def initialize(self,params):
  self.params = params
  return True
 def execute(self):

  # DEFINA AS VARIAVEIS DA ROTINA
  caminho = str(self.params['endereco_arquivo']['value'])
  numero_variogramas = int(self.params['N_variogramas']['value'])
  numero_estruturas = int(self.params['N_estruturas']['value'])
  numero_interacoes =int(self.params['N_interacoes']['value'])
```

```
  set_maximumrange = int(self.params['set_maximumrange']['value'])
  set_minrange = int(self.params['set_minrange']['value'])
  set_verticalrange = int(self.params['set_verticalrange']['value'])

  if (set_maximumrange ==1):
    azimute_direcao_principal =
float(self.params['Azimute_dp']['value'])
    dip_direcao_principal = float(self.params['Dip_dp']['value'])

  if (set_minrange == 1):
    azimute_direcao_secundaria =
float(self.params['Azimute_dm']['value'])
    dip_direcao_secundaria = float(self.params['Dip_dm']['value'])

  if (set_verticalrange == 1):
    azimute_direcao_vertical =
float(self.params['Azimute_dv']['value'])
    dip_direcao_vertical = float(self.params['Dip_dv']['value'])

  restriction = int(self.params['restriction']['value'])
  nlags = int(self.params['nlags']['value'])
  nvariables = int(self.params['nvariables']['value'])
  min_npairs = int(self.params['min_npairs']['value'])

  if (restriction == 1):
    min_contribution =
float(self.params['min_contribution']['value'])
    max_contribution =
float(self.params['max_contribution']['value'])
    min_nugget = float(self.params['min_nugget']['value'])
    max_nugget = float(self.params['max_nugget']['value'])
    if (set_maximumrange == 1):
      min_range_max = float(self.params['min_range_max']['value'])
      max_range_max = float(self.params['max_range_max']['value'])
    if (set_minrange ==1):
      min_range_min = float(self.params['min_range_min']['value'])
      max_range_min = float(self.params['max_range_min']['value'])
    if (set_verticalrange == 1):
```

```
        min_range_vert = float(self.params['min_range_vert']['value'])
        max_range_vert = float(self.params['max_range_vert']['value'])

    else:
        min_contribution = 0
        max_contribution = 0
        min_nugget = 0
        max_nugget =0
        min_range_max = 0
        max_range_max = 0
        min_range_min = 0
        max_range_min = 0
        min_range_vert = 0
        max_range_vert = 0

    # ROTINA PARA LEITURA DE DADOS DO ARCHIVO XML DO GSLIB

    # INICIO DA LEITURA
    try:
        arquivo = open (caminho, "r")
    except:
        print("Could not open the file! Please try again!")

    size_of_matriz = math.sqrt(nvariables)



    azimuteVV =[]
    dipVV =[]
    variogramasV =[]
    dipsV =[]
    lagsV =[]
    paresV=[]
    indice_head =[]
    indice_tail =[]
    tipo = []

    for p in range(0,nvariables):
```

```
    # LEIA OS CABECALHOS DESNECESSARIOS
    # LEIA PRIMEIRA VARIAVEL
    linha = arquivo.readline()
    indice_head.append(int(linha))
    # LEIA SEGUNDA VARIAVEL
    linha = arquivo.readline()
    indice_tail.append(int(linha))


    # LEIA O CABECALHO
    linha = arquivo.readline()
    # DEFINA O VETOR DO AZIMUTE E DO DIP E OS PARAMETROS
DOS VARIOGRAMAS VARIOGRAMAS
    azimuteV = []
    dipV = []
    variogramas =[]
    dips = []
    lags = []
    pares =[]
    # PARA UM NUMERO DE VARIOGRAMAS NO ARCHIVO FACA
    for i in range(0,numero_variogramas):
     variograma =[]
     lag = []
     par =[]
     for j in range(0, nlags):
       linha = arquivo.readline()
       linhasplit = linha.split()
      if(int(linhasplit[4]) >= min_npairs):
         if (j ==0):
          azimuteV.append(float(linhasplit[0]))
          dipV.append(float(linhasplit[1]))
         lag.append(float(linhasplit[2]))
         variograma.append(float(linhasplit[3]))
         par.append(int(linhasplit[4]))
     variogramas.append(variograma)
     lags.append(lag)
     pares.append(par)
    azimuteVV.append(azimuteV)
    dipVV.append(dipV)
```

```
    variogramasV.append(variogramas)
    lagsV.append(lags)
    paresV.append(pares)

  arquivo.close()

  # DETERMINE O INDICE DA DIRECAO PRINCIPAL SEGUNDO
OS VETORES INFORMADOS
  I_dir_principal = -0.5
  I_dir_secundario = -0.5
  I_dir_vertical = --0.5

  informado_principal = False
  informado_secundario = False
  informado_vertical = False

  indice = 0
  interacoes = len(azimuteV)
  for i in range(0,interacoes):
   if (set_maximumrange ==1):
     if (azimuteV[i] == azimute_direcao_principal and dipV[i]
 == dip_direcao_principal):
       I_dir_principal = indice
      informado_principal = True
   if (set_minrange == 1):
     if (azimuteV[i] == azimute_direcao_secundaria and dipV[i]
== dip_direcao_secundaria):
       I_dir_secundario = indice
      informado_secundario = True
   if (set_verticalrange == 1):
     if (azimuteV[i] == azimute_direcao_vertical and dipV[i]
 == dip_direcao_vertical):
       I_dir_vertical = indice
      informado_vertical = True
   indice = indice + 1



  #.....
```

```
  def pesos_distancia(distancia):
   #DEFINA A SOMA ACUMULADA DO INVERSO DA DISTANCIA
   soma = 0
   pesos =[]
   for i in distancia:
    inverso = 1/i
    soma = soma + inverso
   for i in distancia:
    pesos.append((1/i)/soma)
   return pesos




  #.....................................................
  # Funcoes para o calculo dos pesos dos pares
  def pesos_pares(par):
   #DEFINA A SOMA ACUMULADA DOS PARES)
   soma =0
   pesos = []
   for i in range(0,len(par)):
    soma = soma + par[i]
   for j in range(0, len(par)):
    pesos.append(float(par[j])/soma)
   return pesos


  # .....................................................

  # FUNCAO PARA O CALCULO DOS VALORES VARIOGRAMAS PARA CADA TIPO
  def modelo_variogama(range_estruturas, sill_estruturas,
modelos_estruturas, lags, tamanho_lags, tamanho_estruturas):
   valor_variograma_estrutura = []
   n_estruturas = len(modelos_estruturas)
   n_lags = len(lags)
   for i in range(0, n_estruturas):
    valor_variograma = []
    for j in range(0, n_lags):
     if modelos_estruturas[i] == 0:
```

```
        if lags[j] < range_estruturas[i]:
         valor_variograma.append(sill_estruturas[i]*(1.5*lags[j]
-0.5*pow((lags[j]/range_estruturas[i]),3)))
        else:
         valor_variograma.append(sill_estruturas[i])
       elif modelos_estruturas[i] == 1:
        valor_variograma.append(sill_estruturas[i]*
(1-math.exp(-3*lags[j]/range_estruturas[i])))
       elif modelos_estruturas[i] == 2:
        valor_variograma.append(sill_estruturas[i]*
(1-math.exp(-3*pow((lags[j]/range_estruturas[i]),2))))
     valor_variograma_estrutura.append(valor_variograma)
    return valor_variograma_estrutura

   #.....................................................
  # FUNCAO PARA O DESVIO MEDIO QUADRADICO
   def desvio_quad(variograma, modelo_variograma,
tamanho_variograma,tamanho_estruturas, efeito_pepita,
pesos_distancia, pesos_pares):
    soma_desvio = 0
    for i in range(0, tamanho_estruturas):
     diferenca = 0
     m = list(modelo_variograma[i])
     for j in range(0, tamanho_variograma):
      diferenca = pesos_distancia[j]*pesos_pares[j]*math.pow
((m[j]-variograma[j]-efeito_pepita),2)/
tamanho_variograma + diferenca
     soma_desvio = soma_desvio + diferenca
     soma_desvio = math.fabs(soma_desvio)
    return soma_desvio
   #.....................................................................

  def otimizacao(I_dir_principal, variogramasV, lagsV,
paresV, nvariables, tipo, numero_estruturas,restriction,
min_contribution,max_contribution,min_nugget,max_nugget,
min_range,max_range, modeling, direcao, pepm, sillm):
     count = 0
     check_LMC = False
     check_LMC2 = False
```

```
    while (check_LMC == False or check_LMC2 == False):
      check_LMC = False
      check_LMC2 = False
      residuoV =[]
      rangeV =[]
      sillV =[]
      testerV =[]
      pepV = []

      for n in range(0,nvariables):

        variogramas = variogramasV[n]
        pares = paresV[n]
        lags = lagsV[n]

        # DEFINA O PESO PARA OS PARES DE AMOSTRAS

        p_pares = []
        p_pares = pesos_pares(pares[I_dir_principal][:])


        # DEFINA O PESO PARA AS DISTANCIAS
        p_distancia =[]
        p_distancia =
pesos_distancia(lags[I_dir_principal][:])

        # DETERMINACAO DOS PARAMETROS INICIAIS
        # 1) SILL DE CADA ESTRUTURA
        # 2) EFEITO PEPI
        # 3) RANGE DE CADA ESTRUTURA
         #.................................
        #1) SILL DE CADA ESTRUTURA
        sill_estruturas = []
        sill_medio = 0
        soma = 0
        n = 0
        resultado_pepita = 0

        # FACA PARA A DIRECAO PRINCIPAL
```

```
        for i in variogramas[I_dir_principal]:
         soma = soma + i
         n = n + 1
        sill_medio = float(soma/(n*numero_estruturas))
        if (count > 1):
          sill_medio = float(soma/(n*numero_estruturas))

        # SE AS RESTRICOES ESTIVEREM SELECIONADAS FACA O SILL MEDIO
COMO UMA DIFERENCA DA MAX E MIN CONTR
        if (restriction == 1):
          sill_medio = (max_contribution -
min_contribution)/numero_estruturas
        for i in range(0,numero_estruturas):
         sill_estruturas.append(sill_medio )
        #2) EFEITO PEPITA
        resultado_pepita = 0.1*sill_medio

        # SE AS RESTRICOES ESTIVEREM SELECIONADAS FACA
O EFEITO PEPITA IGUAL AO MIN NUGGET
        if(restriction == 1):
          resultado_pepita = min_nugget
        #3) RANGE DE CADA ESTRUTURA
        range_estruturas = []
        max_range_2 = float(max(lags[I_dir_principal])/2)

        for i in range(0, numero_estruturas):
         range_estruturas.append(float(max_range_2
/numero_estruturas)*(i+1))

        # SE AS RESTRICOES ESTIVEREM SELECIONADAS FACA O RANGE INICIAL
DENTRO DOS LIMITES
        if (restriction ==1):
          max_range_2 = (max_range - min_range)/numero_estruturas
          range_estruturas = []
          for i in range(0, numero_estruturas):
            range_estruturas.append(float(max_range_2*(i+1)
+ min_range))
```

```
#4) TIPO DE CADA ESTRUTURA
tipo_estrutura = []
for i in range(0,numero_estruturas):
 tipo_estrutura.append(0)
n = 0

#.....
# INICIE O PROCEDIMENTO DE OTIMIZACAO
# DEFINA OS VALORES DE VARIOGRAMA PARA AS DIRECOES NECESSARIAS

variograma = []
variograma = variogramas[I_dir_principal]
# DEFINA OS VALORES DE LAG PARA A DIRECAO DE MAXIMA CONTINUIDADE

lag = []
lag = lags[I_dir_principal]

# DEFINA O NUMERO DE PONTOS DO VARIOGRAMA

numero_variogramas = len(variograma)

# DEFINA O MAXIMO PATAMAR

max_patamar = max(variograma)

# DEFINA O MINIMO PATAMAR

min_patamar = min(variograma)

# DEFINA O MAXIMO LAG

max_lag = max(lag)

#  DEFINA O MINIMO LAG

min_lag = min(lag)

# DEFINA O NUMERO DE LAGS DO VARIOGRAMA
```

```
        numero_lags = len(lag)


        # DEFINA NUMERO DE PARAMETROS
        # N SILL DAS ESTRUTURAS, N RANGES DAS ESTRUTURAS,
MODELO DAS ESTRUTURAS, EFEITO PEPITA
        numero_de_parametros = numero_estruturas*4
        # CALCULE O PRIMEIRO VALOR DE RESIDUO BASEADO NOS
IMPUTS PRIMARIOS
        # DEFINA O MODELO DAS ESTRUTURAS

        modelo_estruturas = []
        modelo_estruturas = modelo_variogama(range_estruturas[:],
sill_estruturas[:], tipo_estrutura[:], lag[:], numero_lags,
 numero_estruturas)
        #DEFINA O PRIMEIRO RESIDUO

        residuo = 0
        residuo = desvio_quad(variograma, modelo_estruturas[:],
numero_lags, numero_estruturas, resultado_pepita,p_distancia[:]
,p_pares[:])

        # DEFINA A LISTA DE PARAMETROS MODIFICADA

        range_estruturas_modficado = []
        sill_estruturas_modficado = []
        tipo_estruturas_modificado =[]
        pepita_modificado = 0


        # COPIE O VALOR INICIAL

        range_estruturas_modificado = range_estruturas[:]
        sill_estruturas_modificado = sill_estruturas[:]
        tipo_estruturas_modificado = tipo_estrutura[:]
        pepita_modificado = resultado_pepita

        # FACA UMA COPIA DOS VALORES MEDIOS COMO UM VALOR DE RETORNO
        range_estruturas_inicial = []
```

```
        sill_estruturas_inicial = []
        tipo_estruturas_inicial =[]
        pepita_inicial = 0

        range_estruturas_inicial = range_estruturas[:]
        sill_estruturas_inicial = sill_estruturas[:]
        tipo_estruturas_inicial = tipo_estrutura[:]
        pepita_inicial = resultado_pepita

        # DEFINA AS VARIAVEIS DE SAIDA

        resultado_residuo = 0
        resultado_range = []
        resultado_estruturas =[]
        resultado_sill = []

        # DEFINA O PARAMETRO DE MODIFICACAO

        range_estrutura = 0
        sill_estrutura  =0
        modelo_estrutura =0

        # CONDICAO DE SOLUCAO ENCONTRADA

        convergencia_solucao = False

        # OTIMIZE A DIRECAO PRINCIPAL
        #.....
        # FACA PARA O NUMERO DE INTERACOES ESTIPULADO
        for i in range(0,numero_interacoes):
         # DEFINA UM NUMERO ALEATORIO PARA O PARAMETRO DO
VARIOGRAMA A SER MODIFICADO
          aleatorio_parametro = random.randint
(0,numero_de_parametros)
         # DEFINA SE O INCREMENTO SERA POSITIVO OU NEGATIVO
         aleatorio_sinal  = random.choice([-1,1])
         # SEGUNDO O NUMERO DO PARAMETRO ALEATORIO
MODIFIQUE EM 7.5% O VALOR INICIAL
          if (aleatorio_parametro < numero_estruturas):
```

```
            range_estrutura = range_estruturas[aleatorio_parametro]
            range_estrutura = range_estrutura + 0.075*
aleatorio_sinal*range_estrutura
            if(restriction == 1):
              if (range_estrutura < min_range):
                range_estrutura = min_range
              if (range_estrutura > max_range):
                range_estrutura = max_range
            range_estruturas_modificado[aleatorio_parametro]
= range_estrutura
          if (aleatorio_parametro >= numero_estruturas and
aleatorio_parametro < 2*numero_estruturas):
            sill_estrutura = sill_estruturas[(aleatorio_parametro
-numero_estruturas)]
            sill_estrutura = sill_estrutura + 0.075*
aleatorio_sinal*sill_estrutura
            if (restriction == 1):
              if(sill_estrutura > max_contribution):
                sill_estrutura = max_contribution
              if(sill_estrutura < min_contribution):
                sill_estrutura = min_contribution
            sill_estruturas_modificado [(aleatorio_parametro
- numero_estruturas)]
= sill_estrutura
          # DEFINA UM MODELO ALEATORIO PARA A ESTRUTURA
          if (aleatorio_parametro >= 2*numero_estruturas
and aleatorio_parametro
< 3*numero_estruturas):
            aleatorio_modelo = random.randint(0,2)
            tipo_estruturas_modificado[aleatorio_parametro
- 2*numero_estruturas]
= aleatorio_modelo



          # DEFINA O SILL TOTAL
          sill_total = 0
          for j in range(0, numero_estruturas):
            sill_total = sill_total + sill_estruturas_modificado[j]
```

```
          #  DEFINA UM EFEITO PEPITA
          if (aleatorio_parametro >= 3*numero_estruturas
 and aleatorio_parametro
< 4*numero_estruturas):
           pepita_modificado = math.fabs(pepita_modificado +
0.075*aleatorio_sinal*pepita_modificado)
          if (restriction == 1):
            if (pepita_modificado < min_nugget):
              pepita_modificado = min_nugget
            if (pepita_modificado > max_nugget):
              pepita_modificado = max_nugget

          # CALCULE O SILL TOTAL VERDADEIRO

           sill_total = sill_total + pepita_modificado

          range_estruturas_modificado.sort()
             sill_estruturas_modficado.sort()
          # DEFINA OS MODELOS PARA AS ESTRUTURAS VIGENTES
          modelo_estruturas = modelo_variogama
(range_estruturas_modificado[:],sill_estruturas_modificado[:],
 tipo_estruturas_modificado[:], lag[:],
numero_lags, numero_estruturas)
          residuo_modificado = desvio_quad(variograma,modelo_estruturas,
numero_lags, numero_estruturas, pepita_modificado, p_distancia, p_pares)
          #  ACEITAR OU REJEITAR MODIFICACAO BASEADO NA DIMINUICAO
DO RESIDUO MEDIO QUADRADICO
          if (residuo_modificado < residuo):
           if ((sill_total < max_patamar) and
(sill_total > min_patamar)):
             convergencia_solucao = True
             r_range = []
             r_sill = []
             r_testr = []
             r_residuo = residuo_modificado
             r_range = range_estruturas_modificado[:]
             r_sill = sill_estruturas_modificado[:]
             r_testr = tipo_estruturas_modificado[:]
             r_pep = pepita_modificado
```

```
            residuo = residuo_modificado

        if (convergencia_solucao == True):
         residuoV.append(r_residuo)
         rangeV.append(r_range)
         sillV.append(r_sill)
         testerV.append(r_testr)
         pepV.append(r_pep)


        #.....................................................................
```

```
    # TESTE DO LMC ---- SOMA DOS VARIOGRAMAS CRUZADOS
DEVE SER MENOR QUE A SOMA DOS VARIOGRAMAS DIRETOS
    # TESTE EQUIVALENTE AO DETERMINANTE

    soma_diretos =0
    soma_cruzados = 0

    #CRIE A MATRIZ DE CONTRIBUICOES
    indice = int(math.sqrt(nvariables))

    matriz_contr = []
    determ = []
    sillV_new = []
    for p in range(0,numero_estruturas):
      matriz = np.zeros((indice,indice))
      for x in range(0,len(sillV)):

        sill_t = []
        sill_v = []
        sill_t = sillV[x]
        sill_v = sill_t[p]

        i = indice_head[x]
```

```
      j = indice_tail[x]
      i = i - 1
      j = j - 1

      matriz[i][j] = sill_v

    #REGULARIZE A MATRIZ DE DE CONTRIBUICOES

    matriz_new = np.zeros((indice,indice))
    for i in range(0,int(math.sqrt(nvariables))):
      for j in range(0,int(math.sqrt(nvariables))):
        if (i==j):
          matriz_new[i][j] = matriz[i][j]
        else:
          matriz_new[i][j] = (matriz[i][j] + matriz[j][i])/2

    #DEFINA A CONTRIBUICAO PARA A ESTRUTURA
    sill_t_new =[]
    for i in range(0,int(math.sqrt(nvariables))):
        for j in range(0,int(math.sqrt(nvariables))):
          sill_t_new.append(matriz_new[i][j])
    sillV_new.append(sill_t_new)
    determinante = np.linalg.det(matriz_new)
    determ.append(determinante)
    matriz_contr.append(matriz_new)



#DEFINA AS ESTRUTURAS PARA CADA CONTRIBUICAO


for p in range(0,numero_estruturas):
  for x in range(0,nvariables):
    sillV[x][p] = sillV_new[p][x]


def verif(v):
  for i in v:
    if i>0:
```

```
        pass
      else:
        return False
    return True

if (determ > 0):
  check_LMC = verif(determ)

#NORMALIZE OS EFEITOS PEPITAS -> VARIAVEL 12 = VARIAVEL 21


#CRIE A MATRIZ DE EFEITOS PEPITA

matriz2 = np.zeros((indice,indice))
for x in range(0,len(pepV)):

  pep_t = []
  pep_v = []
  pep_v = pepV[x]


  i = indice_head[x]
  j = indice_tail[x]
  i = i - 1
  j = j - 1

  matriz2[i][j] = pep_v

#NORMALIZE OS EFEITOS PEPITAS -> VARIAVEL 12 = VARIAVEL 21

matriz2_new = np.zeros((indice,indice))
for i in range(0,int(math.sqrt(nvariables))):
  for j in range(0,int(math.sqrt(nvariables))):
    if (i==j):
      matriz2_new[i][j] = matriz2[i][j]
    else:
      matriz2_new[i][j] = (matriz2[i][j] + matriz2[j][i])/2

#DEFINA O EFEITO PEPITA NORMALIZADO
```

```
    pep_v_new =[]
    for i in range(0,int(math.sqrt(nvariables))):
      for j in range(0,int(math.sqrt(nvariables))):
        pep_v_new.append(matriz2[i][j])


    determ2 = np.linalg.det(matriz2_new)

    if (determ2 > 0):
      check_LMC2 = True

    count = count + 1
    if (count == 200):
      print("LMC not found")
      check_LMC = True
      check_LMC2 = True

  # FACA O RANGE MEDIO DAS VARIAVEIS


  rangemedio = [0 for i in range(0,len(rangeV[0]))]
  for m in rangeV:
    for i in range(0,len(m)):
      rangemedio[i] = rangemedio[i] + m[i]/len(rangeV)



  # ADOTE O MODELO DA VARIAVEL PRIMARIA PARA TODOS OS TIPOS

  modelo_adotado = []
  modelo_adotado = testerV[0]

  for i in range(0, len(testerV)):
    testerV[i] = modelo_adotado

  # ADOTE A MESMA CONTRIBUICAO E PEPITA E MODELO DA
DIRECAO PRINCIPAL
```

```
    if (direcao > 1):
      pepV = pepm
      testerV = modeling
      sillV = sillm

    if (direcao ==1):
      print("Variogram parameters")
      print ("..................................................
.....................")
      print("model = 0 -> spherical, model = 1 -> exponencial,
model =2 -> gaussian")
      print ("..................................................
.....................")
      print("sills:" + str (matriz_contr))
      print("models:" + str(testerV[0]))
      print("nugget:" + str(matriz2_new))
      print("range: da direcao "+str(direcao) +"  "+
str(rangemedio))
    else:
      print("range: da direcao "+str(direcao) +"  "+
str(rangemedio))
      print ("..................................................
.....................")


    # PLOTAGEM DE VERIFICACAO
    #CRIAR FUNCAO PARA PLOTAGEM


    arquivo2 = open("saida_automatic_fitting.txt","w")
    arquivo2.write(str(direcao)+"\n")
    arquivo2.write(str(nvariables)+"\n")
    arquivo2.write(str(nlags)+"\n")
    for n in range(0,nvariables):
      arquivo2.write(str(indice_head[n]) + "  "+
str(indice_tail[n])+"\n")
```

```
    for n in range(0,nvariables):
      r_sill = sillV[n]
      r_range = rangemedio[0]
      r_pep = pepV[n]
      r_testr = testerV[n]
      variogramas = variogramasV[n]
      lags = lagsV[n]
      variograma = variogramas[I_dir_principal]
      lag = lags[I_dir_principal]

      funcao = []
      for i in np.linspace(0, max(lag), len(lag)):
         valor_variograma = 0
         for j in range(0, numero_estruturas):
          if (r_testr[j] == 0):
             if (i <= r_range):
              valor_variograma = valor_variograma + r_sill[j]
*(1.5*i/r_range-0.5*math.pow((i/r_range),3))
             else:
              valor_variograma = valor_variograma + r_sill[j]
          elif (r_testr[j] == 1):
             valor_variograma = valor_variograma + r_sill[j]
*(1-math.exp(-3*i/r_range))
          elif (r_testr[j] == 2):
             valor_variograma = valor_variograma + r_sill[j]
*(1-math.exp(-3*pow(i/r_range,2)))
         valor_variograma = valor_variograma + r_pep
         funcao.append(valor_variograma)
      limite = np.linspace(0, max(lag),len(lag))

      # SALVAR ARQUIVO EM TXT COM OS DADOS

      for i in range(0,len(lag)):
        arquivo2.write(str(lag[i])+"  "+str(variograma[i])+"  "
+str(limite[i])+"  "+str(funcao[i]) +"\n")
    arquivo2.close()
    os.system("python plot_automatic_fitting.py")

    return testerV, pepV, sillV
```

```
  if (set_maximumrange == 1 and set_minrange == 0 and
set_verticalrange == 0):
    pepV =[]
    saida, pepV, sillV = otimizacao(I_dir_principal,variogramasV,
lagsV, paresV, nvariables, tipo, numero_estruturas,
restriction,min_contribution,max_contribution,min_nugget,
max_nugget,min_range_max,max_range_max, -1, 1, -1, [] )
  if (set_maximumrange == 1 and set_minrange == 1 and
set_verticalrange == 0):
    pepV =[]
    saida, pepV, sillV = otimizacao(I_dir_principal,variogramasV
, lagsV, paresV, nvariables, tipo, numero_estruturas,
 restriction,min_contribution,max_contribution,min_nugget,
max_nugget,min_range_max,max_range_max, -1, 1, -1, [] )
    saida2, pepd, sillV = otimizacao(I_dir_secundario,variogramasV
, lagsV, paresV, nvariables, tipo, numero_estruturas
, restriction,min_contribution,max_contribution,min_nugget,
max_nugget,min_range_min,max_range_min, saida, 2, pepV, sillV )
  if (set_verticalrange == 1 and set_maximumrange ==1
and set_minrange == 1 ):
    pepV =[]
    saida, pepV, sillV = otimizacao(I_dir_principal,variogramasV,
 lagsV, paresV, nvariables, tipo, numero_estruturas,
restriction,min_contribution,max_contribution,min_nugget,
max_nugget,min_range_max,max_range_max, -1, 1, -1, [] )
    saida2, pepd, sillV = otimizacao(I_dir_secundario,variogramasV
, lagsV, paresV, nvariables, tipo, numero_estruturas,
 restriction,min_contribution,max_contribution,min_nugget,
max_nugget,min_range_min,max_range_min, saida, 2, pepV, sillV )
    saida3, pepe, sillV = otimizacao(I_dir_vertical,variogramasV
, lagsV, paresV, nvariables, tipo, numero_estruturas,
 restriction,min_contribution,max_contribution,min_nugget,
max_nugget,min_range_vert,max_range_vert, saida, 3, pepV, sillV )

 def finalize(self):
  print "Finalize optimizing_variogram"
  return True
```

```
 def name(self):
  return "automatic_fitting2"
##################################################
def get_plugins():
 return["automatic_fitting2"]
```