MODELAGEM HIDROLÓGICA SOB UMA ABORDAGEM BAYESIANA: COMPARAÇÃO DE ALGORITMOS MCMC E ANÁLISE DA INFLUÊNCIA DA FUNÇÃO VEROSSIMILHANÇA NA ESTIMATIVA DOS PARÂMETROS E DESCRIÇÃO DAS INCERTEZAS

CÁSSIO GUILHERME RAMPINELLI

DISSERTAÇÃO DE MESTRADO EM TECNOLOGIA AMBIENTAL E RECURSOS HÍDRICOS
DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

MODELAGEM HIDROLÓGICA SOB UMA ABORDAGEM BAYESIANA: COMPARAÇÃO DE ALGORITMOS MCMC E ANÁLISE DA INFLUÊNCIA DA FUNÇÃO VEROSSIMILHANÇA NA ESTIMATIVA DOS PARÂMETROS E DESCRIÇÃO DAS INCERTEZAS

CÁSSIO GUILHERME RAMPINELLI

ORIENTADOR: DIRCEU SILVEIRA REIS JUNIOR
COORIENTADOR: CARLOS HENRIQUE RIBEIRO LIMA

DISSERTAÇÃO DE MESTRADO EM TECNOLOGIA AMBIENTAL E RECURSOS HÍDRICOS

PUBLICAÇÃO: PTARH.DM-188/16
BRASÍLIA/DF: AGOSTO – 2016

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

MODELAGEM HIDROLÓGICA SOB UMA ABORDAGEM BAYESIANA: COMPARAÇÃO DE ALGORITMOS MCMC E ANÁLISE DA INFLUÊNCIA DA FUNÇÃO VEROSSIMILHANÇA NA ESTIMATIVA DOS PARÂMETROS E DESCRIÇÃO DAS INCERTEZAS

CÁSSIO GUILHERME RAMPINELLI

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISÍTOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM TECNOLOGIA AMBIENTAL E RECURSOS HÍDRICOS.

APROVADA POR:

Prof. Dirceu Silveira Reis Junior, PhD (ENC-UnB)

(Orientador)

Prof. Sérgio Koide, PhD (ENC-UnB)

(Examinador Interno)

Prof. Wilson dos Santos Fernandes, Doutor (UFMG)

(Examinador Externo)

BRASÍLIA/DF, 19 DE AGOSTO DE 2016

BRASÍLIA/DF, 19 DE AGOSTO DE 2016

FICHA CATALOGRÁFICA

RAMPINELLI, CÁSSIO GUILHERME

Modelagem hidrológica sob uma abordagem bayesiana: Comparação de algoritmos MCMC e análise da influência da função verossimilhança na estimativa dos parâmetros e descrição das incertezas [Distrito Federal] 2016.

xix, 188p., 210 x 297 mm (ENC/FT/UnB, Mestre, Tecnologia Ambiental e Recursos Hídricos, 2016).

Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Civil e Ambiental.

1.Análise Bayesiana 2.Calibração

3.Incertezas 4.Modelos hidrológicos

I. ENC/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

RAMPINELLI, C. G. (2016). *Modelagem hidrológica sob uma abordagem bayesiana: Comparação de algoritmos MCMC e análise da influência da função verossimilhança na estimativa dos parâmetros e descrição das incertezas*. Dissertação de Mestrado em Tecnologia Ambiental e Recursos Hídricos, Publicação PTARH.DM-188/16, Departamento de Engenharia Civil e Ambiental, Universidade de Brasília, Brasília, DF, 189p.

CESSÃO DE DIREITOS

AUTOR: Cássio Guilherme Rampinelli.

TÍTULO: Modelagem hidrológica sob uma abordagem bayesiana: comparação de algoritmos MCMC e a análise da influência da função de verossimilhança na estimativa dos parâmetros e na descrição das incertezas.

GRAU: Mestre ANO: 2016

_____

Cássio Guilherme Rampinelli
cassiorampinelli@gmail.com

À minha família

"All models are wrong but some are useful"

George Edward Pelham Box

AGRADECIMENTOS

Agradeço aos meus pais, Cássio S. Rampinelli e Maria das Graças, pela participação e motivação contínua na construção da minha formação educacional, sem a qual a realização deste trabalho não seria possível. Em especial, à minha querida mãe, que não mediu esforços para garantir educação de qualidade a seus filhos, mesmo em situações de dificuldade financeira, sempre colocou nossa educação em primeiro lugar.

À minha amada esposa, Ylmara, pela paciência, apoio e compreensão diante do longo tempo que precisei me dedicar ao desenvolvimento deste trabalho.

À minha querida irmã Elaine, parceira e amiga de toda vida e a todos os meus familiares por entenderem minhas freqüentes ausências nos encontros de família.

Ao professor Dirceu, pelo estímulo e motivação nas brilhantes sessões de orientação, pela paciência e compreensão diante das minhas falhas e deficiências que delongaram o término deste trabalho desafiador.

Ao professor Carlos pela predisposição em sempre me atender a qualquer momento que batia em sua porta para receber seus conselhos e sugestões a respeito do andamento da pesquisa. Aos demais professores do PTARH, especialmente, ao professor Sérgio Koide e a professora Yovanka, que me acompanham de perto ao longo desses anos de estudo e dedicação. Ao professor Lineu, grande incentivador,durante meus anos de graduação, para que desse prosseguimento à minha formação acadêmica.

Aos professores Afrânio Vieira e André Cançado do departamento de estatística da UnB por ajudarem na compreensão e nas aplicações de métodos estatísticos. À Josiane Cordeiro pelo apoio e compartilhamento dos códigos em R de seu trabalho, ao Francisco Eustáquio pela generosidade e explicações referentes ao algoritmo DREAM.

Aos amigos do PTARH pelas discussões que ajudaram a enriquecer este trabalho, em especial ao Rodrigo Otsuki e Sara Ferrigo pelos dados e informações compartilhadas e aos amigos Luis Hernandeze Ana Luisa, pelas dicas e conversas valiosas.

RESUMO

Pesquisas realizadas na última década tem mostrado a abordagem bayesiana como uma ferramenta promissora na avaliação de incertezas em modelos hidrológicos e no auxílio à tomada de decisões a partir de prognósticos desses modelos. A abordagem bayesiana com o emprego das denominadas cadeias de Markov via simulação Monte Carlo permite a obtenção das distribuições posteriores completas de todos os parâmetros do modelo e de qualquer função dos mesmos, incluindo a do hidrograma simulado. Contudo, ao se empregar essa abordagem faz-se necessário representar a natureza assumida para os erros do modelo por meio de uma função de verossimilhança. É comum, em diversos casos de simulação, assumir que os resíduos possam ser representados por uma distribuição normal com média zero e desvio padrão conhecido. Entretanto, diversos estudos tem revelado que essa premissa é frequentemente violada para a maioria dos casosem hidrologia.

Este trabalho se insere nesse contexto e apresenta o desenvolvimento ea aplicação dos algoritmos *Metropolis Hastings* e *Adaptive Metropolis* para a estimativa das incertezas referentes aos parâmetros do modelo hidrológico do tipo conceitual denominado *Soil Moisture Accounting Procedure* (SMAP), para passo de tempo mensal. Complementarmente, o algoritmo *Differential Evolution Adaptive Metropolis* (DREAM), amplamente testado em diversos trabalhos correlatos, é empregado com o propósito de validar os resultados obtidos com os algoritmos implementados, bem como avaliar o comportamento das estimativas dos erros e das distribuições posteriores quando se faz uso de uma função de verossimilhança generalizada, capaz de agregar a não normalidade, a heterocedasticidade e a correlação entre os resíduos. Com o propósito de melhor explorar o comportamento dos parâmetros e contrapor a abordagem bayesiana com os métodos de calibração determinísticos usualmente empregados, os parâmetros são também estimados com o algoritmo de buscaglobal PSO. Os resultados indicam o comportamento adequado dos algoritmos implementados, mostrando o potencial da abordagem bayesiana para a avaliação de incertezas bem como demonstram a influência da função de verossimilhança nas distribuições posteriores dos parâmetros e nas vazões simuladas. Com o intuito de corroborar para disseminação e uso dessa ferramenta os códigos implementados em linguagem R são disponibilizados ao final do trabalho para os principais casos estudados.

ABSTRACT

Researches conducted in the last decade has shown Bayesian approach as a promising tool in exploring uncertainty in hydrologic modeling and as an useful support in decision making process.The application of the Bayesian approach, implemented by Markov chains via Monte Carlo simulation, provides the full posterior distribution of the model parameters and any function of them, including the distribution of the simulated flows. However, when applying Bayesian approach the modeler should make assumptions about the errors nature by implementing an appropriate likelihood function. It is usual, in many cases, assuming that residuals can be described by a normal distribution with zero mean and a given standard deviation. However, several studies have shown that this assumption is often violated for most cases in hydrology.

This work corroborates to this context and presents Metropolis Hastings and Adaptive Metropolis algorithms implemented in R programming language in order to assess the uncertainty regarding the Soil Moisture Accounting Procedure (SMAP) parameters, a conceptual Rainfall-Runoff model, to monthly time steps. In addition, DREAM algorithm, widely applied in several related works, is used in order to validate the results obtained with the implemented algorithms. Furthermore, residuals behavior are analyzed by a Generalized Likelihood Function that better address the non-normality, heterocedasticity and correlation between errors. For the sake of comparison, parameters were also estimated with a global optimization procedure and with a Bayesian approach.Results illustrate the potential of the Bayesian approach in exploring the parameters uncertainties, and indicate the influence of the likelihood function in the parameter posterior distributions and simulated flow. In order to corroborate to the use and dissemination of Bayesian analyses in hydrologic modeling, programming codes implemented in R statistical language are available at the end of work for the main case studies.

SUMARIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE NOMENCLATURA E ABREVIAÇÕES

AM	*Adaptive Metropolis*
AR	*Modelo Autoregressivo*
ASCE	*American Society of Civil Engineers*
BaRE	*Bayesian Recursive Estimation Technique*
DE-MC	*Differential Evolution Markov Chain*
DREAM	*Differential Evolution Adaptive Metropolis*
DYNIA	*Dynamic Identifiability Analysis Framework*
FDP	Função Densidade de Probabilidade
GL	Função de Verossimilhança Generalizada
GLUE	*Generalized Likelihood Uncertainty Estimation*
IAHS	*International Association of Hydrological Sciences*
L	Verossimilhança
M	*Metropolis*
MCMC	*Markov Chain Monte Carlo*
MH	*Metropolis Hastings*
MLBMA	*The Maximum Likelihood Bayesian Averaging Method*
NS	Função objetivo - *Nash-Suticliffe*
PSO	*Particle Swarm Optimization*
PUB	*Predictions in Ungauged Basins*
SA	*Simulated Anneling*
SCEM	*Shuffled Complex Evolution Metropolis*
SCEM-UA	*Shuffled Complex Evolution Metropolis Algorithm*
SMAP	*Soil Moisture Accounting Procedure*
SODA	*Simultaneous Optimization and Data Assimilation Algorithm*
AWBM	*Water Balance Model*

LISTA DE SÍMBOLOS

α	Taxa de Probabilidade de Aceitação
B	Condições de Contorno do Sistema/ Variância entre Cadeias
u	*Input* - Entrada do Sistema
C_i	Matriz de variância-covariância no passo i
C_0	Matriz de variância-covariância adotada até o passo i_0
$Cov()$	Função que retorna a matriz de variância-covariância
d	Dimensão do vetor (número de elementos)
DA	*Data Assimilation*
ε	Parâmetro de valor pequeno escolhido para que $C_i$não se torne singular
ε_t	Erros associados ao modelo ao longo do tempo t
ε_t^*	Erros associados ao modelo ao longo do tempo t para variável transformada
$f(x_t;\theta)$	Função do modelo hidrológico com dados de entrada x_te parâmetros θ
$f(x_i;\mu)$	Função de densidade de probabilidade com parâmetro(s) μ aplicada a variável x_i
i	Passo corrente da iteração
i_0	Número pré-definido de iterações (ou passos)
I_d	Matriz identidade de dimensão d
θ	Vetor de Parâmetros do Modelo
$\bar{\theta}$	Média do Vetor de Parâmetros do Modelo
θ_0	Vetor de ParâmetrosIniciais do Modelo
$\theta^i;\theta^{i+1}$	Vetor de Parâmetros no passo corrente da iteração e no passo seguinte
$\bar{\theta};\bar{\theta}_i;\bar{\theta}_{i-1}$	Vetor de Parâmetros Médio no passo i e no passo i+1
θ_i^T	Transposta do vetor de parâmetros no passo i
$\overline{\theta}_i^T$; $\overline{\theta}_{i-1}^T$	Transposta do vetor de parâmetros médios no passo ie no passo i-1
θ^p	Vetor de Parâmetros Propostos
$L(\theta;x_1,\ldots,x_n)$	Verossimilhança de uma função de densidade de probabilidade f com parâmetros θ, aplicada sobre os dados x_1a x_n
$\lambda;\lambda_1;\lambda_2$	Parâmetro da transformação Box Cox
M	Algoritmo Metropolis

MH	Algoritmo Metropolis-Hastings
N	Número de cadeias
$N\sim(0,\sigma^2)$	Distribuição Normal com média 0 e variância σ^2
$N\sim(\theta,C)$	Distribuição Normal multivariada com média dada pelo vetor θ e matriz de covariância C
n_1	Número de vazões nulas com erros 0
n_2	Número de vazões nulas com erros diferentes de 0
n_3	Número de vazões não nulas
$P(Q\|\theta)$	Função de verossimilhança
$p(\theta)$	Distribuição *a priori*
$P(\theta\|Q)$	Função de densidade de probabilidade posterior do conjunto de parâmetros θ
$P(Q)$	Constante de proporcionalidade
ρ	Probabilidade de que o resíduo seja nulo para vazão nula
ρ_k	Autocorrelação na Defasagem (*lag*) k
q	Distribuição proposta
Q_{obs}	Vazão Observada
$\bar{Q}_{obs}$	Média das Vazões Observadas
Q_{sim}	Vazão simulada
R	Fator potencial de redução de escala de Gelaman-Rubin
s_d	Fator de escala que depende da dimensão d
σ^2	Variância Populacional
σ	Desvio Padrão Populacional/ Desvio Padrão do Erro do Modelo
T	Número total de passos de tempo
t	Passo de tempo corrente
u	*Input* - Entrada do Sistema
$\widehat{\mathrm{Var}}(\theta)$	Variância estimada para a distribuição estacionária
W	Variância dentro de uma única cadeia
x	Estado do Sistema
x_t	Dados de entrada do modelo ao longo do tempo t
x_0	Estados Iniciais do Sistema
y	*output* - Saída com série de vazões simulada pelo modelo

1 - INTRODUÇÃO

A utilização de modelos hidrológicos permite aos hidrólogos extrapolar o conhecimento adquirido por meio de observações, tanto em termos espaciais, particularmente úteis no caso de bacias não monitoradas, quanto temporais, quando o objetivo é extender o histórico de séries ou realizar previsões dos estados futuros de variáveis associadas ao fenômeno modelado (vazão, volume, nível do curso d'água, dentre outros) (Silva *et al.,* 2014).

Dessa forma, o emprego de modelos hidrológicos é último planejamento e gestão sustentável de recursos hídricos, sendo imprescindível para projetos e operações de sistemas de aproveitamento e controle de recursos hídricos, para o planejamento de drenagem urbana, gerenciamento de práticas agrícolas, mapeamento e planos de redução de riscos de inundações, obtenção de licenciamentos e outorgas de uso de recursos hídricos, dentre outros. Beven (2001) aponta ainda, que os modelos podem ser aplicados com fins científicos, com o propósito de simular processos que não podem ser observados diretamente, tais como a percolação da água do solo, por exemplo. Singh & Woolhiser (2002) discutem diversas aplicações dos modelos hidrológicos e apresentam uma perspectiva histórica da evolução dos modelos de simulação, em especial, dos modelos do tipo chuva-vazão.

Marshall (2005) destaca, contudo, que apesar dos diversos modelos hidrológicos que têm surgido, nenhum modelo se mostrou ideal para ser aplicado na ampla gama de situações de modelagem hidrológica que existem. Liu & Gupta (2007) ressaltam que, nas últimas décadas, o aumento considerável no poder de processamento dos computadores, assim como a ampliação da disponibilidade de dados hidrológicos observados e os avanços na compreensão da física e dinâmica dos sistemas hidrológicos têm contribuído para o desenvolvimento de modelos hidrológicos cada vez mais complexos. Paradoxalmente, ainda que a sofisticação desses modelos reflita em um aumento no conhecimento a respeito dos processos hidrológicos há, também, de se considerar um aumento associado ao grau de incerteza das simulações, seja em função da maior necessidade de dados para processamento ou do maior número de parâmetros para o modelo de interesse.

Outro aspecto relevante que tem motivado o desenvolvimento de técnicas para quantificação de incertezas, diz respeito à consideração dos riscos de tomadas de decisões a partir de prognósticos de modelos.

Portanto, a consideração de incertezas no emprego dos modelos hidrológicos é de relevante importância e tem sido tema de pesquisas cada vez mais recorrentes na comunidade científica (Beven, 2009; Beven, 2001; Marshall *et al.,* 2004; Reis & Stedinger, 2005; Bates & Campbell, 2001; Marshall, 2005; Wagener & Wheater, 2006; Montanari *et al.,* 2009; Wagener & Montanari, 2011; Liu & Gupta, 2007; Smith, 2012; Silva *et al.,* 2014; Silva, 2015).

Atualmente, a inferência bayesiana tem emergido como uma ferramenta potencial para avaliar as incertezas nas modelagens hidrológicas (Marshall, 2005). A abordagem bayesiana permite construir ferramentas para uma representação estocástica de todas as incertezas inerentes ao processo de modelagem hidrológica, incluindo aspectos referentes aos dados e à estimativa dos parâmetros, bem como da própria estrutura do modelo (Liu & Gupta, 2007). O produto final dessa abordagem consiste em uma distribuição probabilística (denominada distribuição posterior ou *a posteriori*) que abarca as incertezas referentes aos dados e a estrutura do modelo. Assim, os parâmetros do modelo hidrológico passam a ser vistos como distribuições probabilísticas e não como valores determinísticos.

No que concerne à definição da distribuição posterior, a complexa natureza não linear de grande parte dos modelos hidrológicos impossibilita a solução analítica das integrais necessárias para o cálculo das distribuições marginais o que, de certa forma, dificulta o emprego da inferência bayesiana.

Contudo, o uso dos métodos baseados nas denominadas Cadeias de Markov via simulação Monte Carlo (*Markov Chain Monte Carlo* - MCMC) tem permitido a aplicação da abordagem bayesiana com resultados promissores (Marshall *et al.,* 2004). Algoritmos MCMC operam construindo uma cadeia de Markov, cuja distribuição estacionária consiste na distribuição posterior dos parâmetros. Diversos trabalhos fizeram uso de métodos MCMC com o propósito de adotar a abordagem bayesiana para avaliação de incertezas em modelos hidrológicos (Kuczera & Parent, 1998; Bates & Campbell 2001; Vrugt *et*

*al.,*2003; Marshall *et al.,* 2004; Reis & Stedinger, 2005; Cordeiro, 2009; Kuczera *et al.,* 2010; Smith, 2012; Silva *et al.,* 2014).

Esta dissertação se propõe a avaliar diferentes algoritmos MCMC e entender como as premissas referentes à natureza dos resíduos afetam as estimativas dos parâmetros do modelo, assim como as incertezas das vazões simuladas, além de demonstrar o potencial existente no emprego da inferência bayesiana para a avaliação de incertezas associadas à modelos hidrológicos.

A estrutura deste documento está dividida em 8 capítulos, sendo o primeiro uma breve introdução ao contexto do trabalho já apresentada nesta seção. O capítulo 2 trata dos objetivos da dissertação. O capítulo 3 traz uma revisão bibliográfica referente aos principais temas envolvidos na pesquisa a saber: modelos hidrológicos, calibração de modelos hidrológicos, incertezas associadas à modelagem hidrológica e o emprego da abordagem bayesiana para a avaliação de incertezas. O capítulo 4 apresenta os algoritmos de interesse referentes ao uso de cadeias de Markov via Simulação Monte Carlo (MCMC), bem como aspectos relativos à métricas de avaliação de convergência e o emprego de funções de verossimilhança. O Capítulo 5 apresenta a metodologia proposta para o desenvolvimento do trabalho. O Capítulo 6 mostraos resultados obtidos e as discussões dos casos simulados. O Capítulo 7 apresenta as conclusões e recomendações para trabalhos futuros. Por fim, apresentam-se as referências bibliográficas que subsidiaram os estudos realizados.

2 - OBJETIVOS

2.1 - OBJETIVO GERAL

Empregar a abordagem bayesiana para avaliaçãode incertezas associadas aos parâmetros de um modelo hidrológico chuva-vazão do tipo conceitual, bem como verificar o reflexo dessas incertezas nas séries de vazões geradas pelo modelo, analisando conjuntamente o desempenho de diferentes algoritmos MCMC e a influência da função de verossimilhança na estimativa dos parâmetros do modelo.

2.2 - OBJETIVOS ESPECÍFICOS

- Desenvolver código computacional em linguagem R que permita obter distribuições posteriores dos parâmetros do modelo SMAP mensal, assim como qualquer função dos mesmos, com base em três algoritmos MCMC conhecidos na literatura científica, a saber: *Metropolis* (M), *Metropolis Hastings* (MH) e *Adaptive Metropolis* (AM);
- Avaliar de forma comparativa o desempenho dos algoritmos *Metropolis* (M), *Metropolis Hastings* (MH) e *Adaptive Metropolis* (AM) na estimativa dos parâmetros do modelo SMAP mensal e na descrição das respectivas incertezas, empregando tanto dados hidrológicos sintéticos, quanto dados reais; e
- Avaliar a influência das premissas adotadas em relação à natureza dos erros do modelo no processo de calibração do modelo SMAP mensal, sintetizadas na escolha da função de verossimilhança, na estimativa dos parâmetros e na descrição das incertezas.

3 - REVISÃO DA LITERATURA

Esta seção apresenta uma revisão bibliográfica a respeito da definição, classificação e aplicação dos modelos hidrológicos (mais especificamente dos modelos do tipo chuva-vazão), além de uma apresentação geral das incertezas envolvidas nos processos de modelagem hidrológica e do emprego da abordagem bayesiana como ferramenta para avaliação das incertezas. Também se discorre brevemente a respeito da consideração dessas incertezas na tomada de decisão.

3.1 - MODELOS HIDROLÓGICOS DO TIPO CHUVA-VAZÃO

3.1.1 - Contextualização e Aplicações

Os modelos chuva-vazão consistem em uma forma de se estimar séries de vazões em locais desprovidos de dados hidrométricos de vazão. A partir de um modelo hidrológico pré-estipulado se estabelece uma relação entre a precipitação e o escoamento resultante.

Essa relação engloba diversos processos do ciclo hidrológico de tal forma que esses modelos buscam descrever o comportamento da vazão a partir de dados de precipitação e evapotranspiração, levando em consideração as perdas por interceptação, depressões do solo, o fluxo através do solo pela infiltração, percolação e água subterrânea, escoamento superficial, sub-superficial e na calha do rio.

Os modelos do tipo chuva-vazão possuem as mais variadas aplicações, seja em estudos de engenharia, investigações hidrológicas ou ciências ambientais. Wagener *et al.* (2004) destacam diversos empregos recorrentes desses modelos na literatura, dentre eles: extensão de séries de vazões no espaço e tempo, avaliação de estratégias de gerenciamento e/ou respostas de bacias hidrográficas relacionadas à variabilidade climática e de uso do solo, vazões de projetos, previsão de vazões em tempo real, ou ainda, fornecer condições de contorno para modelos de circulação atmosférica. Beven & Hall (2014) e Beven (2009) apresentam aplicações da avaliação de incertezas como ferramenta para suporte à tomada de decisões no campo do gerenciamento de riscos à inundações, além de aplicações em engenharia ambiental.

Uma vantagem dos modelos chuva-vazão na estimativa de séries de vazões em locais sem dados está no fato de que o tamanho da série gerada não se limita à série de vazão de um posto de referência, mas sim ao tamanho da série de precipitação da região. Como usualmente a extensão dos registros pluviométricos supera a cobertura de medições de vazões, com o emprego de modelos do tipo chuva-vazão é possível extrapolar séries de vazões em determinadas localidades que, eventualmente, estariam limitadas a extensão das séries dos postos de referência. Ademais, a partir das previsões de volumes ou índices pluviométricos para determinada região é possível estimar as correspondentes vazões esperadas, uma vez que os parâmetros do modelo tenham sido estimados.

Tucci (2005) elenca de forma sistemática diversos usos para os modelos hidrológicos, destacando-se além do uso em estudos, projetos, previsões de vazões e preenchimento de falhas, o emprego dos modelos hidrológicos como forma de melhor compreender os processos físicosdos fenômenos hidrológicos naturais.

Dessa forma, os modelos hidrológicos de bacias hidrográficas corroboram com diversos trabalhos relacionados aos recursos hídricos, seja em nível de projeto, planejamento ou gerenciamento, tendo como principal saída às vazões nos cursos de água, em resposta a séries contínuas de precipitação e evapotranspiração, ou a eventos isolados (Pinheiro, 2009).

3.1.2 - Classificações

Em uma percepção mais ampla, os modelos do tipo chuva-vazão constituem uma categoria dos modelos de simulação hidrológica. Singh & Woolhiser (2002) apresentam, de forma mais geral, classificações de modelos hidrológicos recorrentes na literatura e destacam que a estruturação dos modelos é determinada em função dos objetivos pelos quais o modelo é construído. Neste trabalho é apresentada uma classificação proposta por um dos autores: Singh (1995) e outra indicada pela Sociedade Americana de Engenheiros Civis (*American Society of Civil Engineers* – ASCE, 1996).

A primeira classificação categoriza os modelos hidrológicos de acordo com: (1) o processodescrito; (2) a escala de tempo; (3) a escala espacial; (4) as técnicas de solução; (5) o uso do solo e (6) o uso do modelo.

A segunda classificação citada (ASCE, 1996) divide os modelos hidrológicos da seguinte forma: (1) modelos baseados em eventos pontuais de precipitação e escoamento, (2) modelos contínuos de precipitação e escoamento; (3) modelos de escoamento em regime permanente; (4) modelos de escoamento em regime transiente; (5) modelos de regulação de reservatório e (6) modelos de análise de frequência de cheias.

Os modelos chuva-vazão tratados neste trabalho podem ser considerados integrantes da segunda categoria da classificação da ASCE (1996), pois o interesse está na simulação de séries de vazões contínuas a partir do histórico de chuva.

Tomando como referência os modelos do tipo chuva-vazão, Costa (2011) apresenta de forma mais detalhada uma classificação destacada por Pinheiro (2009), que divide os modelos de acordo com a escala temporal, escala espacial, natureza dos processos envolvidos e estrutura.

A escala temporal diz respeito à representação de comportamentos contínuos ou de eventos discretos. No primeiro caso, são empregadas séries contínuas de vazões, precipitações e/ou outras variáveis hidrológicas para se realizar as simulações desejadas vislumbrando o comportamento hidrológico por um aspecto mais geral. No segundo caso, busca-se a simulação da resposta da bacia para um evento pré-determinado, tal como uma chuva de projeto ou de um evento crítico do passado, por exemplo.

Com relação à escala espacial, os modelos podem ser concentrados ou distribuídos. Nos modelos concentrados, a bacia hidrográficaérepresentadade forma pontual, em que,em dado um ponto de referência, busca-se,por meio do modelo, contemplar as variações dos fenômenos distribuídosno espaço, focando-se no comportamento da variável de interesse (por exemplo, a vazão) ao longo do domínio do tempo. No modelo distribuído, avalia-se o comportamento do fluxo distribuído no domínio espaço-tempo na bacia.

Wagener *et al.* (2004) fazem ainda referência aos modelos semi-distribuídos. Tal como citado anteriormente, esses modelos consistem em uma simplificação do modelo distribuído, pois fazem uso da combinação de vários modelos concentrados que compartimentam uma bacia maior em sub-bacias menores que permitem a avaliação dos

fluxos em diversos pontos da bacia maior de referência. Ou seja, trata-se de modelos concentrados aplicados a subbacias.

Referente à natureza dos processos, os modelos podem ser classificados em estocásticos, determinísticos ou híbridos. Os modelos estocásticos são aqueles em que a teoria de probabilidades é empregada para a formulação do problema. Já os modelos determinísticos são regidos por leis físicas específicas. Os modelos híbridos combinam a abordagem de ambos: estocásticos e determinísticos.

Concernente à estrutura, segundo Wagener *et al.* (2004) os modelos se apresentam como empíricos (métricos ou *black box*), conceituais (paramétricos ou *grey box*) ou de base física. Os modelos empíricos usualmente utilizam os dados disponíveis das séries históricas para derivar os valores simulados a partir de funções que, em princípio, não possuem nenhuma relação com o comportamento da bacia ou o processo de escoamento. São, portanto, baseados unicamente nos dados disponíveis. Dessa forma, não são muito adequados para serem empregados em regiões com indisponibilidade de dados. Dentre vários recorrentes na literatura, destacam-se os modelos baseados em redes neurais artificiais ou em funções de transferências (McIntyre *et al.,* 2011; Young, 1988).

Os modelos conceituais, por sua vez, embora, também, se baseiem fortemente nos dados de séries históricas de saída do sistema – *output* (usualmente dados de vazão, para estimar os valores dos parâmetros a partir de calibração), tem sua estrutura concebida a partir de um entendimento do comportamento hidrológico do sistema. Utilizam usualmente estruturas de armazenamento e propagação, representadas por reservatórios fictícios que são abastecidos a partir de fluxos decorrentes da precipitação, infiltração ou percolação, e esvaziados a partir dos processos de evapotranspiração, escoamento, drenagem, etc. (Wagener *et al.,* 2011, Arnold *et al.,* 1998, Bergström, 1995, Collischonn *et al.,* 2007).

Os parâmetros do modelo representam aspectos como a capacidade ou a área dos reservatórios de armazenamento, a distribuição dos fluxos entre eles, a taxa de deplecionamento, dentre outros. Dessa forma, esses parâmetros agregam inúmeros processos que ocorrem no espaço e tempo e, usualmente, não são passíveis de serem obtidos a partir de medições em campo (não possuem uma interpretação física claramente definida, conforme descrito por Wagener *et al.,* 2004).

A maioria dos modelos conceituais considera a bacia como uma unidade homogênea. Contudo, uma abordagem relativamente comum é a segmentação da bacia em pequenas sub-bacias, na tentativa de se compor um modelo concentrado por subbacias. Os modelos conceituais são os mais utilizados em aplicações práticas, porém, a forte dependência de dados observados de vazão para calibração dos parâmetros desses modelos limita as suas aplicações a bacias com disponibilidade de dados. Contudo, estudos (Seibert, J., 2009; Fernandez, *et al*., 2000; Parajka, *et al*., 2005; Bárdossy, 2007) vêm sendo desenvolvidos na tentativa de se estabelecer relações estatísticas entre os parâmetros desses modelos e características das bacias o que permitiria o seu emprego em regiões com limitação ou indisponibilidade de dados.

A iniciativa lançada pela *International Association of Hydrological Sciences*-IAHS por meio do programa *Predictions in Ungauged Basins*-PUB (Sivapalan, *et al.*, 2003) revela a preocupação da comunidade científica em nível internacional com o fomento de pesquisas na área de previsão em bacias sem monitoramento.

Os modelos de base física são estruturados nas equações fundamentais de conservação da massa, momento e energia. Nesses modelos, os processos a serem modelados são usualmente descritos a partir de sistemas de equações diferenciais parciais não lineares, os quais, na maioria dos casos, não possuem soluções analíticas. Esses sistemas de equações buscam descrever matematicamente os diferentes e complexos processos envolvidos na natureza dos ciclos hidrológicos. O avanço na capacidade de processamento dos computadores, acompanhado da evolução das técnicas de soluções numéricas de sistemas de equações diferenciais, permitiu a aplicação prática desses modelos a partir da década de 80 (Freeze & Harlan, 1969; Abbott *et al.,* 1986; Calver, 1988).

A expectativa com os modelos de base física era a de que, devido ao fato de se basearem no comportamento físico dos processos, seus parâmetros estivessem diretamente relacionados com as características físicas da bacia modelada, tais como a condutividade hidráulica, coeficiente de atrito do escoamento, dentre outros. Dessa forma, a necessidade de calibração poderia ser eliminada, uma vez que seria possível medir esses parâmetros em campo. Contudo, a complexidade e a grande heterogeneidade dos sistemas reais exigem uma quantidade extrema de dados. Há ainda problemas relacionados a efeitos de escala que distorcem as relações entre os parâmetros medidos em campo e a representatividade desses

nos modelos empregados, além de uma série de outros complicadores que acabam fazendo com que mesmo os modelos de base física façam uso de calibração para alguns parâmetros (Beven, 2009). Esses tipos de modelos são mais empregados quando um alto nível de discretização espacial é importante para se estimar efeitos locais de erosão do solo, ou poluição de águas superficiais e subterrâneas (Wagener *et al.,* 2004). Todini (1996) ressalta que os modelos de base física são apropriados para avaliar os efeitos do uso do solo, erosões e interações entre águas superficiais e aquíferos subterrâneos. Há ainda modelos híbridos que mesclam as abordagens de modelos conceituais e empíricos ou ainda empíricos e físicos.

Neste trabalho serão abordados modelos do tipo conceituais concentrados que representam com performance satisfatória e com uma estrutura relativamente simples a previsão de séries de vazões em uma bacia para uma determinada seção de interesse.

3.1.3 - Procedimentos Típicos para Aplicação de Modelos Chuva-Vazão

Vários procedimentos são sugeridos para a utilização de modelos conceituais chuva-vazão. Esses procedimentos podem variar de acordo com cada caso em questão. Contudo, Wagener *et al.* (2004) destacam os passos mais comuns encontrados nas diversas aplicações desses modelos, são eles: seleção da estrutura do modelo, análise de sensibilidade, calibração, validação e previsão. Beven (2001) destaca ainda uma fase anterior a todas as etapas que consiste no denominado modelo perceptivo (*perceptual model*) que consiste basicamente em uma síntese de nossas percepções a respeito de como a bacia responderá aos eventos de chuva sob diferentes condições. Essa fase é fundamental quando se pretende implementar um novo modelo pois requer uma esquematização/diagramação dos diversos processos relacionados à conversão da chuva em escoamento superficial, sub-superficial e subterrâneo.

3.1.3.1. - Seleção da Estrutura do Modelo

A seleção da estrutura do modelo deve estar relacionada ao propósito pretendido com a aplicação (tipo de processo que deve ser melhor descrito para a representação de determinada bacia de interesse), ao histórico e aos tipos de dados disponíveis, à experiência do hidrólogo na utilização de determinado modelo, ao custo/tempo de se

desenvolver ou se familiarizar com determinada estrutura de modelo, entre outros fatores. Woolhiser & Brakensiek (1982) destacam que metodologias objetivas para definir qual o melhor modelo a ser empregado ainda não foram desenvolvidas e que a escolha ainda é, de certa maneira, subjetiva e parte da arte da modelagem hidrológica.

3.1.3.2. - Análise de Sensibilidade

A análise de sensibilidade consiste em uma avaliação para se verificar o quão sensível é a saída do modelo (*output*) à modificações nos valores dos parâmetros. Dessa forma, determinados parâmetros são fixados outrosvariadosexaustivamente observando-se as modificações na saída do modelo. Assim, pode-se observar quais parâmetros exercem maior influência na resposta do modelo. Tal análise pode ser realizada antes ou após a calibração com objetivos distintos. A análise de sensibilidade antes da calibração, busca, sobretudo, verificar o grau de influência de cada parâmetro na resposta do modelo e é usada para direcionar os espaços de busca dos parâmetros nos processos de otimização. Após a calibração, a análise de sensibilidade visa verificar o quão robusto é o modelo à alterações pequenas dos parâmetros calibrados. A análise de sensibilidade pode ser empregada ainda para direcionar a avaliação de incertezas na modelagem realizada (Trucano, *et al.,* 2006).

3.1.3.3. - Calibração

A calibração consiste no processo de seleção dos parâmetros de tal forma que os hidrogramas simulados e observados sejam os mais próximos possíveis. Esse grau de proximidade dos valores simulados e aqueles observados pode ser baseado em uma simples comparação visual dos hidrogramas simulados e observados ou a partir de outro tipo de medida (ou métrica) aplicada para avaliar a performance do modelo de forma mais objetiva. Esse segundo procedimento pode derivar de estatísticas ou aspectos do comportamento hidrológico comparativo compondo as funções objetivo. A função objetivo consiste em uma expressão matemática para medir a diferença entre os dados observados e aqueles simulados pelo modelo. Assim, o objetivo da calibração usualmente consiste em minimizar (ou maximizar dependendo da definição) o valor da função objetivo. Gupta *et al.* (1998) apresentam uma discussão a respeito da seleção de funções objetivo, bem como da realização de calibração multiobjectivo (Yapo *et al.,* 1998; Boyle, 2001; Madsen, 2000),

onde mais de uma função objetivo é considerada no procedimento de calibração. Por não ser escopo desta pesquisa esse tema, a função objetivo adotada se limitará à função (NS) apresentada por Nash & Sutcliffe (1970), dada pela seguinte expressão:

$$NS = 1 - \frac{\sum(Q_{obs} - Q_{sim})^2}{(Q_{obs} - \bar{Q}_{obs})^2} \quad (3.1)$$

em que, Q_{obs} é a vazão observada, $\bar{Q}_{obs}$é a média das vazões observadas e Q_{sim}é avazão simulada com base no modelo.

O procedimento de calibração pode ser realizado de forma manual ou automática. No primeiro caso, a partir de processos de tentativa e erro modificam-se os valores dos parâmetros do modelo observando as proximidades entre os hidrogramas simulados ou observando o comportamento da(s) função(ões) objetivo(s). A calibração automática faz uso de procedimentos computacionais de otimização para obtenção dos chamados parâmetros ótimos que otimizam a(s) função(ões) objetivo(s). Existem diversos algoritmos de otimização que se propõem a buscar os ótimos das funções objetivo, destacando-se os métodos de busca de ótimos locais e ótimos globais. Os algoritmos de busca local foram um dos primeiros métodos de otimização de funções não lineares aplicados aos procedimentos automáticos de calibração de modelos hidrológicos, tendo sido desenvolvidos a partir da idéia básica de fazer o algoritmo evoluir encontrando novos pontos situados em direções para as quais o valor da função decresça (ou cresça) em relação ao ponto corrente. Dentre os métodos de busca local podem-se citar os algoritmos de otimização baseados no método gradiente (Jameson, 1995) e no método de Newton (Murray, 2010), o algoritmo de Hooke & Jeeves (1961) e o método de Nelder & Mead (1965).

Contudo, o comportamento das funções objetivo aplicadas em modelos hidrológicos é conhecido por apresentar muitos pontos de ótimos locais. Esse aspecto passou a ser superado com emprego de algoritmos de busca global, tais como o algoritmo genético (Wang, 1991), o método *Simulated Annealing - SA* (Sumner, 1995), o algoritmo *Shuffled Complex Evolution Metropolis - SCEM-UA* (Duan *et al.,* 1993)*, o* algoritmo *Particle Swarm Optimization-PSO* (Eberhart & Kennedy, 1995) e métodos baseados em redes neurais artificiais (Hsu *et al.,* 1995). A associação de métodos de busca local com métodos

de busca global também surge como alternativa para aumentar o desempenho dos algoritmos (Franchini,1996). Zhang *et al.* (2009) compararam o desempenho de diferentes algoritmos de busca global. Dessa forma, a calibração de modelos hidrológicos constitui uma ampla linha de pesquisa dentro da temática de modelos de simulação hidrológica, Duan *et al.* (2002) reúnem uma série de publicações a respeito de métodos aplicados à calibração de modelos hidrológicos para previsão, projetos e gerenciamento de recursos hídricos.

3.1.3.4. - Validação

A etapa de validação ou verificação basicamente consiste na aplicação do modelo com os parâmetros ótimos definidos em uma amostra de dados que não fora utilizada no procedimento de calibração. Usualmente, se reparte a amostra em duas sub-amostras de períodos distintos. Na literatura encontram-se várias sugestões de procedimentos para efetuar a validação, contudo, a definição do que esta etapa realmente representa é bastante controversa (Wagener *et al.,* 2004).

3.1.3.5. - Aplicação

A última etapa consiste na aplicação do modelo para previsão ou geração da série sintética de vazões de forma a compor o hidrograma para o período desejado. Diversos estudos tem sido apresentados na tentativa de se incluir uma avaliação das incertezas associadas as previsões realizadas.

A Figura 3.1 resume as etapas básicas descritas para simulação hidrológica a partir de um modelo do tipo chuva-vazão.

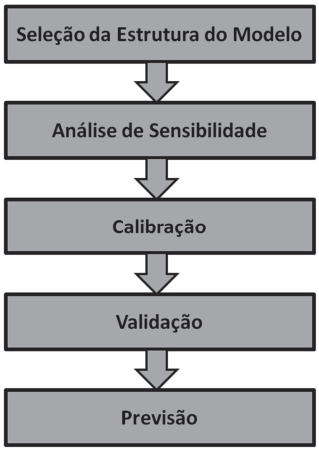

Figura 3.1 - Etapas para o emprego de modelos chuva-vazão (Adaptado de Wagener *et al.,* 2004)

3.2 - INCERTEZAS NOS MODELOS DE SIMULAÇÃO HIDROLÓGICA E A ABORDAGEM BAYESIANA

A consideração das incertezas no processo de modelagem hidrológica e o desenvolvimento de métodos concretos para lidar com essas incertezas nos modelos de simulação têm se tornado cada vez mais recorrentes na literatura, sendo observado um crescente número de publicações dedicadas a este tema, conforme apresentado por Silva *et al.* (2014).

Beven (2001) adota o termo equifinalidade para reconhecer que não há um conjunto único de parâmetros ótimos e que diferentes conjuntos de parâmetros podem levar a resultados aceitáveis de performance do modelo. Dessa forma, o emprego de métodos determinísticos para a definição de parâmetros ótimos traz certa limitação, abrindo espaço para propostas de metodologias mais robustas que descrevem o comportamento dos parâmetros agregando as incertezas associadas.

Marshall (2005) relembra que as incertezas dos modelos podem ser compreendidas a partir das incertezas da própria estrutura do modelo (inclui a concepção ou mecanismo de representação dos processos físicos representados e a forma de especificação de seus parâmetros) e das incertezas advindas da observação (que incluem aquelas oriundas dos dados de entrada, dos processos de calibração, dos contornos do modelo e das condições iniciais).

No campo das incertezas oriundas das observações ou dos dados, destacam-se as diversas limitações nos sistemas de medições e observação dos processos hidrológicos que ocorrem na bacia, a saber: limitações tecnológicas (os equipamentos utilizados não possuem tecnologia suficiente para captar todos os processos naturais envolvidos), limitações econômicas (não é possível monitorar todos os pontos da bacia o que pode comprometer a representatividade espacial dos dados) e restrições físicas (observações ou experimentos efetuados em laboratório podem não representar exatamente as condições existentes em campo ou no meio ambiente natural).

3.2.1 - Incertezas a Partir de Uma Visão Sistêmica

Liu & Gupta (2007) debatem mais a fundo aspectos relacionados às fontes de incerteza e métodos recorrentes na literatura empregados para tratar as incertezas associadas aos modelos de simulação hidrológica e de previsão. Os autores destacam três principais aspectos a serem considerados ao se lidar com as incertezas:

- a compreensão das fontes ou origens das incertezas;
- a quantificação das incertezas, e
- a redução das incertezas.

Com base na proposta dos autores, o modelo deve ser entendido como formado por múltiplos componentes os quais possuem erros associados que se propagam nas simulações efetuadas aumentando as incertezas nas saídas das simulações.

Os autores consideraram um modelo genérico formado por sete diferentes componentes: contornos do sistema (B), entradas do sistema –*input* (*u*), estados iniciais no sistema (x_0), parâmetros (θ), estrutura do modelo (M), estado (x) e saídas do modelo –*output* (y). A título de ilustração, *u* pode se referir, por exemplo, àsérie de precipitação e evapotranspiração sobre a bacia hidrográfica (delimitada pelo contorno B do sistema de interesse), y pode representar a distribuição das vazões ao longo do tempo em um dado ponto de interesse da bacia, x pode se referir à distribuição variável ao longo do tempo da umidade armazenada no interior dos contornos do sistema, por exemplo, e θ pode representar o conjunto de parâmetros do modelo M. Os parâmetros da bacia são usualmente considerados invariáveis ao longo do tempo, ou a taxa de variação é admitida

muito lenta quando comparada com a variação dos estados do modelo. Contudo, esse tema não será aprofundado neste texto, por não se tratar do foco deste trabalho. A Figura 3.2 representa esquematicamente os componentes básicos do modelo aqui apresentado. Maiores detalhes a respeito podem ser consultados em Liu & Gupta (2007).

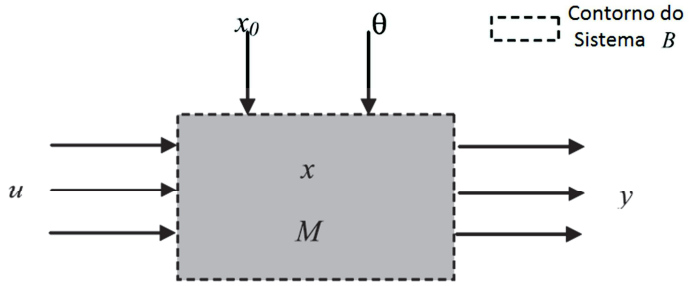

Figura 3.2 - Diagrama esquemático dos componentes de um modelo a partir de uma perspectiva sistêmica (Adaptado de Liu & Gupta, 2007)

Tomando como referência os sete componentes do modelo genérico indicado, cinco deles (B,u,x_0,θ e M) precisam ser especificados, estimados ou definidos antes que o modelo seja processado, enquanto dois deles (x e y) são computados ao se processar o modelo. Cada um dos cinco componentes predefinidos pode trazer algum tipo de incerteza para modelagem que se propagará para os estados e saídas processados. Portanto, os dados de entrada, os parâmetros, a estrutura do modelo, as condições iniciais e os contornos do sistema representam as cinco maiores fontes de incertezas na modelagem hidrológica. Em uma visão mais geral, pode-se considerar o estado inicial (x_0) como parâmetro do modelo e o contorno como componente da estrutura modelo, dessa forma, as incertezas podem ser reduzidas a três tipos primários na modelagem hidrológica: erros associados a estrutura, aos parâmetros e aos dados.

Enquanto a ciência hidrológica tem testemunhado crescentes avanços na disponibilidade de dados de diversas fontes (dados hidrométricos e meteorológicos oriundos de levantamentos de campo e radares meteorológicos) e na complexidade e representatividade dos modelos hidrológicos, ainda existe uma necessidade latente de técnicas que efetivamente consigam assimilar a importância de todas informações contidas nos dados disponíveis de forma a aprimorar as previsões e resultados dos modelos de simulação hidrológica (Liu & Gupta, 2007).

A estimativa dos parâmetros busca valores apropriados para os parâmetros do modelo a partir dos dados disponíveis de forma a aprimorar a resposta dada pelas simulações em relação ao comportamento real do sistema representado. Neste campo, tradicionalmente, os estudos mais recorrentes na literatura referem-se a abordagens determinísticas de calibração (tal como comentado anteriormente) a partir de técnicas de calibração manual ou automática que apresentam uma abordagem simplificada em relação aos erros relativos à modelagem hidrológica. Os procedimentos tradicionais de estimativa dos parâmetros do modelo geralmente admitem que os resíduos da simulação apresentam comportamento estatístico similar a uma distribuição normal com média 0 e desvio padrão σ. Contudo, diversos trabalhos tem mostrado que esta premissa é frequentemente violada nos casos práticos de simulação hidrológica (Bates & Campbell, 2001; Schoups & Vrugt, 2010; Smith, *et al.,* 2015; Silva, *et al.,* 2014). Assim, tem crescido o emprego de métodos que empregam a inferência bayesiana para a estimativa de parâmetros de modelos hidrológicos, inserindo-se neste campo o escopo deste trabalho.

O problema de identificação dos sistemas ou modelos constitui, segundo Liu & Gupta (2007), no mais importante, porém, também, no de abordagem mais difícil. Esse tipo de problema envolve a seleção de estruturas apropriadas de modelos que buscam representar os sistemas reais. Dessa forma, buscam-se construir representações matemáticas adequadas, a partir de equações que representem as relações entre entrada de dados no modelo, parâmetros, estados e saída de dados.

Liu & Gupta (2007) apresentam uma revisão de métodos tipicamente utilizados em modelagem hidrológica para a estimativa de parâmetros, de estado e identificação de sistemas considerando incertezas envolvidas nas estimativas. A seguir, listam-se os principais métodos elencados pelos referidos autores que, por não constituírem o foco deste trabalho, serão apenas listadas a seguir. Maiores detalhes podem ser consultados em Liu & Gupta (2007) ou a partir dos trabalhos dos autores que desenvolveram as referidas metodologias, listadas a seguir:

- GLUE – Generalized Likelihood Uncertainty Estimation (Beven& Binley, 1992);
- BaRE – Bayesian Recursive Estimation Technique (Thiemann *et al.,* 2001);

- SCEM – Shuffled Complex Evolution Metropolis Algorithm (Vrugt *et al.,* 2003);
- DYNIA- Dynamic Identifiability Analysis Framework (Wagener *et al.,* 2003);
- MLBMA – The Maximum Likelihood Bayesian Averaging Method (Neuman, 2003); e
- SODA – Simultaneous Optimization and Data Assimilation Algorithm (Vrugt *et al.,* 2005).

Dessa forma, se considerarmos os dados de vazão observados ao longo de um período t como $Q_{obs,t}$, a estimativa desses valores a partir de um modelo definido por uma função f pode ser expressa, conforme descrito em Marshall (2005) como:

$$Q_{obs,t} = f(x_t, \theta) + \varepsilon_t \quad (3.2)$$

em que: $f(x_t, \theta)$ representa a saída do modelo ao longo do tempo t, x_t consiste nos dados de entrada do modelo ao longo do tempo de simulação t, θ equivale ao conjunto de parâmetros do modelo e ε_t representa os erros associados ao modelo, que possuem uma dada distribuição de probabilidades.

3.2.2 - Inferência bayesiana e as incertezas nos modelos hidrológicos

O termo inferência (ou indução) refere-se ao processo de raciocínio pelo qual, partindo-se do conhecimento de uma parte, se procura tirar conclusões sobre o todo. A partir da década de 1930, a inferência estatística experimentou um marcante surto de desenvolvimento, porém, baseando-se em certos princípios restritivos e hipóteses então formulados. Ao corpo de resultados decorrentes desse movimento dá-se o nome de Estatística Clássica (Bekman & Neto, 2009).

A partir da década de 50, surge uma nova corrente de pensamento que, de certa forma, se contrapõe aos princípios e hipóteses do enfoque clássico. Esse movimento representa um retorno às origens da inferência e o abandono dos princípios restritivos referentes à corrente clássica, sendo denominado Estatística Bayesiana, em homenagem ao reverendo

Thomas Bayes (1702-1761), pela interpretação que se deu ao teorema que leva seu nome (Bekman & Neto, 2009) e que é sucintamente apresentado a seguir, em termos de parâmetros θ e vazões associados a um modelo hidrológico:

$$P(\theta \mid Q) = \frac{p(\theta) \cdot P(Q \mid \theta)}{P(Q)} \tag{3.3}$$

Em que: P(Q|θ) é a função de verossimilhança, P(Q) é uma constante de proporcionalidade, *p(θ)* representa o conhecimento inicial existente a respeito dos parâmetros, dado por uma distribuição de probabilidades denominada distribuição *a priori* e P(θ|Q) representa a distribuição posterior do parâmetro. Na maioria dos casos, a distribuição posterior não pode ser obtida analiticamente, sendo necessário recorrer a técnicas MCMC para obtê-la por amostragem.

Conforme discutido em Silva *et al.* (2014), enquanto no contexto clássico, a medida da probabilidade capta a variabilidade inerente ao processo, no contexto bayesiano, tal medida captura o desconhecimento do indivíduo sobre a variável em estudo. Dessa forma, os frequentistas vêem o parâmetro θ como um valor fixo e tentam estimar este valor desconhecido maximizando uma função de verossimilhança, enquanto os bayesianos o consideram um parâmetro aleatório e buscam a sua modelagem a partir de uma distribuição de probabilidades, ou função de densidade de probabilidade - FDP, a qual resume o conhecimento que se tem sobre essas quantidades (Fernandes, 2009).

Portanto, o processo de inferência se inicia pela atribuição de uma distribuição de probabilidades sobre o parâmetro a ser estimado. Essa distribuição deve espelhar da melhor forma possível nosso conhecimento prévio a respeito do parâmetro em questão.

Procedida a amostragem, obtêm-se uma distribuição posterior (ao experimento) aplicando-se o teorema de Bayes à distribuição prévia. Essa metodologia de atualização de probabilidades é habitualmente denominada inferência bayesiana, para diferenciá-la de outros métodos mais tradicionais que tentam excluir da análise qualquer conhecimento prévio a respeito do parâmetro (Bekman & Neto, 2009).

Silva *et al.* (2014) destacam que a função de verossimilhança P(Q|θ) desempenha um importante papel no teorema de Bayes. É através dela que os dados observados transformam o conhecimento *a priori* sobre os parâmetros θ, ou seja, por meio da função de verossimilhança é possível atualizar uma informação prévia sobre θ continuamente, à medida que novas observações são realizadas. A Figura 3.3 apresenta uma ilustração esquemática desse conceito.

Nas aplicações da análise bayesiana em modelos hidrológicos, a função de verossimilhança é empregada para avaliação da performance da simulação. O objetivo, em geral, consiste em maximizar (ou minimizar) seu valor a partir da comparação das séries simuladas e observadas. Ressalta-se que a aplicação do método se condiciona, em regra, ao ajuste de um modelo probabilístico para os resíduos.

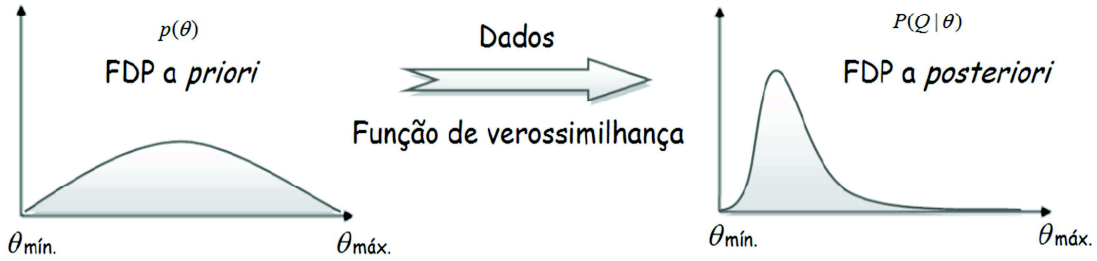

Figura 3.3 - Representação gráfica do teorema de Bayes (adaptado de Silva *et al.*,2014).

Portanto, é comum observar na literatura técnica relativa à calibração de modelos, algoritmos de calibração desenvolvidos com o intuito de minimizar o erro quadrático do modelo (Silva *et al.,* 2014).

A abordagem tradicional assume basicamente duas hipóteses: a homocedasticidade e a independência dos erros. Isto implica em assumir que a variância do erro do modelo (σ^2) é constante, admitindo-se que os resíduos (ε) podem ser representados por uma distribuição normal com média 0 e uma variância σ^2, dado por $\varepsilon \sim N(0,\sigma^2)$. Contudo, tal como comentado anteriormente, constatam-se na literatura trabalhos que indicam que tal premissa é frequentemente violada.

Embora na abordagem bayesiana, muitas vezes se assumam essas hipóteses, ao se admitir que os parâmetros sejam variáveis aleatórias que seguem um modelo probabilístico, os

resultados advindos da distribuição posterior permitem a verificação das hipóteses assumidas para o comportamento dos erros do modelo.

O emprego de métodos bayesianos em modelos complexos, tal como os que se encontram em simulação hidrológica, podem requerer um enorme esforço computacional, principalmente, para a estimação das distribuições posteriores que usualmente não possuem soluções analíticas. Os avanços crescentes na capacidade de processamento e a disseminação do uso dos computadores, bem como avanços nas implementações dos algoritmos computacionais relacionados à abordagem bayesiana (Marshall, 2005) têm aumentado a recorrência de estudos nessa área. O aperfeiçoamento dos algoritmos se referem ao desenvolvimento de metodologias baseadas em simulações Monte Carlo via cadeias de *Markov* ou *Markov Chain Monte Carlo* (MCMC), que envolvem amostragens da distribuição posterior via simulação Monte Carlo (Marshall, 2005).

4 - CADEIAS DE MARKOV VIA AMOSTRAGEM COM SIMULAÇÃO MONTE CARLO E MÉTODOS BAYESIANOS

A complexa natureza não linear de grande parte dos modelos hidrológicos impossibilita a solução analítica das integrais necessárias para o cálculo das distribuições marginais dos parâmetros do modelo, o que, de certa forma, dificulta o emprego da inferência bayesiana.

O uso de métodos MCMC tem permitido a aplicação da abordagem bayesiana com resultados favoráveis. Algoritmos MCMC operam construindo uma cadeia de Markov,cuja distribuição estacionária consiste na distribuição posterior dos parâmetros.

De acordo com Gelman *et al.* (2003), um método MCMC é qualquer método que vise a simulação de uma determinada distribuição alvo (no caso a distribuição referenciada como posterior) baseado em uma amostragem aleatória de valores da distribuição alvo, a partir de uma distribuição aproximada, cujos valores amostrados são "corrigidos" de forma a melhorar a convergência para a distribuição alvo, ou a denominada distribuição estacionária.

Diversos trabalhos fizeram uso de métodos MCMC com o aplicações da abordagem bayesiana para avaliação de incertezas em modelos hidrológicos (Kuczera & Parent,1998; Bates & Campbell 2001;Vrugt *et al.,* 2003; Marshall *et al.,* 2004; Cordeiro, 2009; Kuczera *et al.,* 2010; Schoups & Vrugt, 2010; Silva *et al.,* 2014).

Embora existam diferentes algoritmos MCMC de amostragem da distribuição posterior, neste trabalho serão empregados algoritmos do tipo Metropolis, que, de uma forma geral, pode-se resumir no seguinte procedimento:

i. Inicia-se a simulação (na iteração i=1) com um conjunto inicial de parâmetros dado por um vetor $\theta^0 = (\theta_1^0; \theta_2^0; \theta_3^0 ; \ldots ; \theta_m^0)$, em que no elemento θ_j^0 , "0" refere-se ao vetor inicial de parâmetros indexados pelo índice j, variando de 1 até "m" parâmetros;

ii. Geram-se valores propostos $\theta^p = (\theta_1^p; \theta_2^p; \theta_3^p; \ldots; \theta_m^p)$ para o conjunto de parâmetros do modelo a partir de uma distribuição proposta q que depende dos valores dos parâmetros θ^i no passo corrente i;

iii. Calcula-se uma probabilidade de aceitação α que depende dos valores propostos e dos valores dos parâmetros θ^i no passo corrente. Essa probabilidade de aceitação define se os valores propostos serão aceitos ou não para compor o vetor de parâmetros do passo seguinte da cadeia θ^{i+1};

iv. Com base na probabilidade de aceitação, pode-se aceitar os valores propostos, ou seja, faz-se $\theta^{i+1}=\theta^p$, ou mantêm-se os valores correntes para o próximo passo da cadeia $\theta^{i+1}=\theta^i$; e

v. Incrementa-se o valor i do passo da iteração e reinicia-se no passo ii.

Após iniciar as iterações, a dificuldade seguinte surge no estabelecimento do critério de parada. A avaliação do desempenho do método associa-se à capacidade de convergência do algoritmo. Uma maneira relativamente simples de se avaliar a convergência das cadeias geradas consiste na análise do traço das cadeias com base na avaliação gráfica dos valores dos parâmetros ao longo das iterações. A questão da convergência será melhor discutida em seção posterior, onde serão apresentadas algumas métricas ou critérios considerados para definição da convergência da cadeia. A seguir, são apresentados os algoritmos MCMC que serão utilizados neste trabalho.

A Figura 4.1 apresenta um resumo geral dos passos relativos ao emprego de métodos MCMC para a avaliação de incertezas em modelos de simulação hidrológica.

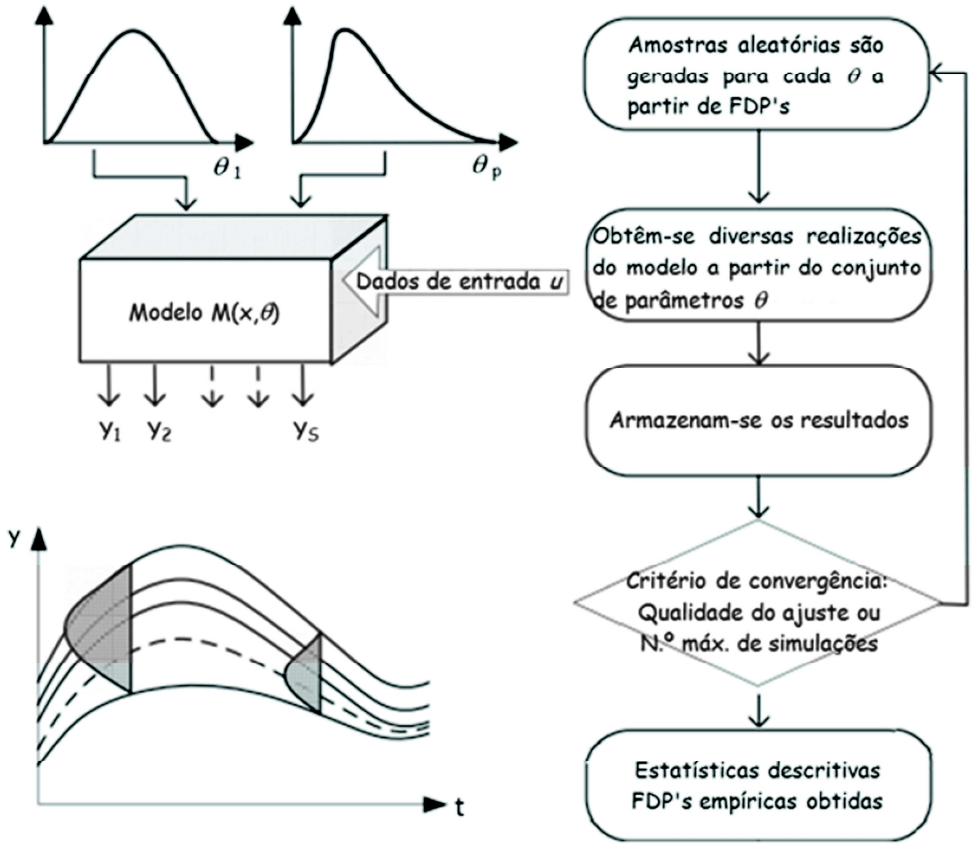

Figura 4.1 - Representação esquemática do emprego do método MCMC para avaliação de incertezas (Silva *et al.*,2014).

4.1 - ALGORITMO METROPOLIS (M)

Suponha que queremos obter amostras da distribuição posterior dos parâmetros $\theta = (\theta_1, \theta_{2,\ldots,} \theta_m)$ de um modelo hidrológico chuva-vazão do tipo conceitual, em que m representa o total de parâmetros do modelo. Utiliza-se ainda a notação θ^i para denotar o vetor de parâmetros $\theta^i = (\theta_1^i, \theta_2^i, \ldots, \theta_m^i)$no passo i da iteração e que armazenará valores selecionados para o vetor de parâmetros da distribuição posterior a partir de um vetor inicialmente proposto de parâmetros $\theta^p = (\theta_1^p, \theta_2^p, \ldots, \theta_m^p)$, que poderá ou não ser aceito. Os passos resumidos a seguir exemplificam a estrutura geral do algoritmo proposto por Metropolis *et al.* (1953).

i. Inicialize i=1;
ii. Arbitre valores iniciais para o vetor de parâmetros $\theta^{i=1}=\theta^0$;
iii. Aplique o modelo hidrológico $f(x_t; \theta^i)$, para t=1,2,...T, em que T representa o número total de passos de tempo;

iv. Amostre valores propostos θ^p para o vetor de parâmetros com base na distribuição proposta escolhida, q;

v. Aplique o modelo hidrológico $f(x_t; \theta^p)$ para t=1,2...T;

vi. Calcule a razão de aceitação

$$\alpha = \frac{P(f(x_t; \theta^p)|\theta^p) \cdot P(\theta^p)}{P(f(x_t; \theta^i)|\theta^i) \cdot P(\theta^i)} \quad (4.1)$$

vii. Amostre u ~ Uniforme[0,1];

viii. Se (u<min(1,α))

Então $\theta^{i+1} = \theta^p$ (aceita-se o vetor de parâmetros propostos)

Senão $\theta^{i+1} = \theta^i$ (mantém-se no passo seguinte da cadeia o vetor de parâmetros do passo anterior)

ix. i=i+1 e volte ao passo iii até que i>número total de iterações.

4.2 - ALGORITMO METROPOLIS-HASTINGS (MH)

Hastings (1970) apresentou uma versão geral do algoritmo de *Metropolis et al.* (1953). O algoritmo de *Metropolis* consiste, na verdade, em um caso particular do algoritmo de *Metropolis-Hastings*, quando o valor da distribuição proposta, com os parâmetros fixados nos valores propostos, aplicada nos valores dos parâmetros no passo corrente $q(\theta^i|\theta^p)$ equivale ao valor da distribuição proposta, com os parâmetros fixados nos valores no passo corrente, aplicada nos valores propostos $q(\theta^p|\theta^i)$, ou seja, quando essas distribuições são simétricas ($q(\theta^i|\theta^p) = q(\theta^p|\theta^i)$). A seguir, apresenta-se o algoritmo MH que se assemelha ao algoritmo M, com diferença apenas no cálculo da razão de aceitação α.

i. Inicialize i=1;

ii. Arbitre valores iniciais para o vetor de parâmetros θ^0 e atribua este vetor ao primeiro passo da cadeia;

iii. Aplique o modelo hidrológico $f(x_t; \theta^i)$, para t=1,2,...T, em que T representa o número total de passos de tempo;

iv. Amostre valores propostos θ^p para o vetor de parâmetros, da distribuição proposta escolhida, q;

v. Aplique o modelo hidrológico $f(x_t; \theta^p)$ para t=1,2...T;

vi. Calcule a razão de aceitação

$$\alpha = \frac{P(f(x_t; \theta^p)|\theta^p) \cdot P(\theta^p) \cdot q(\theta^i|\theta^p)}{P(f(x_t; \theta^i)|\theta^i) \cdot P(\theta^i) \cdot q(\theta^p|\theta^i)} \quad (4.2)$$

vii. Amostre u ~ Uniforme[0,1];

viii. Se (u<min(1,α))

Então $\theta^{i+1} = \theta^p$ (aceita-se o vetor de parâmetros propostos)

Se não $\theta^{i+1} = \theta^i$ (mantém-se no passo seguinte da cadeia o vetor de parâmetros do passo anterior)

ix. i=i+1 e volte ao passo iii até que i > número total de iterações.

4.3 - ALGORITMO ADAPTIVE METROPOLIS (AM)

O algoritmo *Adaptive Metropolis* (AM) proposto por Haario *et al.* (2001) consiste em uma variação do algoritmo *Metropolis*. O algoritmo AM caracteriza-se por uma distribuição proposta tomada como uma distribuição normal multivariada de covariânciabaseada nas estimativas realizadas para os parâmetros ao longo dos passos da cadeia. A cada passo i da iteração, o valor proposto dos parâmetros é obtido com base em uma distribuição normal multivariada com média dada pelos valores dos parâmetros no passo corrente e matriz de variância-covariância C_i. A matriz C_i assume um valor fixo C_0 até um número inicial de iterações i_0, e é atualizada sequencialmente conforme a expressão:

$$C_i = \begin{cases} C_0 & ; para\ i \leq i_0 \\ s_d Cov((\theta_0, \dots, \theta_{i-1}) + s_d \varepsilon I_d) & ; para\ i > i_0 \end{cases} \quad (4.3)$$

em que ε é um parâmetro de valor pequeno escolhido para garantir que C_i não se torne singular; I_d é a matriz identidade; e s_d é um fator de escala que depende da dimensão d do vetor θ para garantir taxas de aceitação razoáveis para os valores propostos. Como uma diretriz geral Haario *et al.* (2001) sugerem que a escolha do fator s_d para um modelo de dimensão d seja dada por:

$$s_d = \frac{(2,4)^2}{d} \tag{4.4}$$

A matriz de covariância na iteração i+1 é obtida de forma a satisfazer a seguinte equação:

$$C_{i+1} = \frac{i-1}{i} C_i + \frac{s_d}{i} (i\overline{\theta}_{i-1}\overline{\theta}_{i-1}^T - ((i+1))\overline{\theta}_i\overline{\theta}_i^T + \theta_i\theta_i^T + \varepsilon I_d) \tag{4.5}$$

A estimativa da matriz de covariância requer a definição de uma valor inicial para C_0. Marshall (2005) e Haario *et al.* (2001) propõem que a matriz inicial de variância-covariância seja obtida a partir das distribuições *a priori* dos parâmetros. Os passos referentes ao algoritmo AM são resumidos a seguir.

i. Inicialize i=1;
ii. Determine C_i para o passo corrente da iteração;
iii. Amostre valores propostos θ^p, tomando $\theta^p \sim N(\theta^i, C_i)$;
iv. Aplique o modelo hidrológico $f(x_t; \theta^p)$ para t=1,2...T;
v. Calcule a razão de aceitação

$$\alpha = \frac{P(f(x_t; \theta^p)|\theta^p) \cdot P(\theta^p)}{P(f(x_t; \theta^i)|\theta^i) \cdot P(\theta^i)} \tag{4.6}$$

vi. Amostre u ~ Uniforme[0,1];
vii. Se (u<min(1,α))
Então $\theta^{i+1} = \theta^p$(aceita-se o vetor de parâmetros propostos)

Se não $\theta^{i+1} = \theta^i$(mantém-se no passo seguinte da cadeia o vetor de parâmetros do passo anterior)

viii. i=i+1 e volte ao passo ii até que i > número total de iterações.

Conforme destacado por Marshall (2005), ressalta-se que a configuração do algoritmo AM que estabelece a dependência da distribuição proposta a todo histórico da cadeia ao invés estabelecer os valores propostos apenas com referência ao passo imediatamente anterior consiste em um fator que descaracteriza a ocorrência de um processo de Markov. Contudo, Haario *et al.* (2001) demostraram a validade do algoritmo no que se refere a convergência para a distribuição posterior.

Marshall (2005) relata que embora o algoritmo AM tenha sido aplicado em modelos com mais de 200 parâmetros, podem ser necessárias modificações para melhorar as taxas de aceitação nesses casos, uma vez que a atualização da cadeia é feita com todo o conjunto de parâmetros simultaneamente, conforme proposto por Haario *et al.* (2001).

4.4 - ALGORITMO DIFFERENTIAL EVOLUTION ADAPTIVE METROPOLIS (DREAM)

O Algoritmo DREAM (*Differenatial Evolution Adaptive Metropolis*) foi apresentado inicialmente por (Vrugt *et al*., 2008, 2009) e consiste em uma adaptação do algoritmo *Shuffled Complex Evolution Metropolis* (SCEM-UA) (Vrugt *et al*., 2003). O algoritmo simula múltiplas cadeias simultaneamente e efetua ajustes automáticos nas distribuições propostas durante a evolução dos passos da cadeia.

A base do algoritmo DREAM deriva do algoritmo *Differential Evolution Markov Chain* (DE-MC), proposto por ter Braak (2006). O método DE-MC consiste em um algoritmo genético com critério de seleção similar ao do algoritmo Metropolis. N diferentes cadeias são simuladas simultaneamente e as populações de parâmetros θ são armazenadas em uma matriz N X d, em que d refere-se à dimensão do problema (no caso, o número de parâmetros a ser estimado). Propostas multivariadas são geradas a partir de um número fixo de diferenças entre elementos de duas cadeias aleatoriamente selecionadas sem reposição com os índices r_1 e r_2, tal como descrito a seguir:

$$\theta^{p} = \gamma_{d}(\theta_{i-1}^{r1} - \theta_{i-1}^{r2}) + e_{d}\ ; \text{com } r_1 \neq r_2 \neq i \qquad (4.7)$$

Em que γ_d é um fator dado por $\gamma_d = 2{,}38/\sqrt{2d}$ e e_dé uma amostra de uma distribuição normal com média 0 e variância pequena (10^{-6}). A taxa de aceitação de Metropolis é utilizada para determinar se os parâmetros propostos são aceitos ou rejeitados. Maiores detalhes a respeito do algoritmo podem ser consultados em ter Braak (2006) ou no Manual do método DREAM (Vrugt, 2015).

Vrugt *et al.* (2009) ressaltam a eficiência do algoritmo DE-MC quando comparado a outros métodos que trabalham com apenas uma cadeia. O ajuste da distribuição proposta a partir de cadeias múltiplas permite uma busca ampla e variada do espaço amostral. A título de demonstração, o referido autor comparou os métodos AM e o DE-MC com base nos histogramas das distribuições posteriores obtidas para uma distribuição alvo relativamente simples composta por uma mistura de duas distribuições normais. O algoritmo AM produz uma aproximação expúria da distribuição alvo bimodal. A variância da distribuição proposta é insuficiente para permitir uma amostragem adequada dos dois modos da distribuição alvo. Uma alternativa para este problema é aumentar a variância da distribuição proposta de tal forma a permitir que o algoritmo AM realize passos mais largos na cadeia alternando entre os dois modos da distribuição alvo. Contudo, isto pode reduzir significativamente a taxa de aceitação e a eficiência do método. Já o algorimo DE-MC para o caso citado consegue explorar adequadamente os dois modos da distribuição alvo e converge rapidamente para distribuição posterior. A Figura 4.2 apresenta o resultado do caso exemplificativo citado.

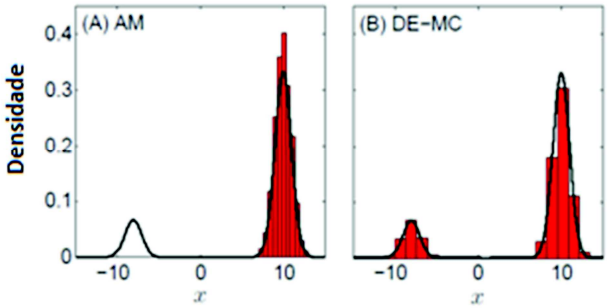

Figura 4.2 - Histograma da distribuição posterior obtida a partir do algoritmo AM (cadeia simples) (A) e DE-MC (cadeias múltiplas). A linha sólida escura representa a distribuição alvo verdadeira (adaptado de Vrugt, 2015).

Dessa forma, segundo Vrugt *et al.* (2009) e conforme descrito em Silva (2015) o algoritmo DE-MC realiza uma seleção automática da escala e da orientação da distribuição proposta, permitindo que distribuições multimodais e de caudas pesadas sejam eficientemente

exploradas. O altoritmo utiliza a posição atual na cadeia em contraposição à covariância adaptativa para gerar pontos candidatos e favorecer movimentos diretos entre as modas da distribuição alvo.

O algoritmo DREAM possui suas raízes no algoritmo DE-MC, contudo, utiliza subespaços de amostragem e correções de *outliers* gerados na cadeia para acelerar a convergência para a distribuições posterior. A atualização dos pontos candidatos é realizada por meio de cruzamento binomial que também aumenta a convergência do algoritmo para problemas com muitas dimensões. Equanto no algoritmo DE-MC os valores propostos são obtidos a partir de dois membros de N cadeias consideradas, no algoritmo DREAM os pontos candidatos incluem um número elevado de combinações entre as N cadeias, aumentando a busca no espaço amostral.

O algoritmo utilizado neste trabalho consiste em uma versão convertida para o software R (R Core Team, 2013), implementada em uma biblioteca por Guillaume & Andrews (2012). A limitação desta versão em R em relação a original consiste na distribuição a priori que é admitida uniforme. Detalhes específicos do algoritmo podem ser consultados nos arquivos disponibilizados nas bibliotecas do pacote. A Tabela 4.1 resume os argumentos de entrada da função DREAM.

Tabela 4.1- Argumentos de entrada da função DREAM no R

Argumento	**Descrição**
func.type	Função de verossimilhança adotada
pars	Lista com os nomes e intervalos de busca dos parâmetros
nseq	Número de cadeias avaliadas simulataneamente
ndraw	Número máximo de iterações
burn-in	Número de iterações desconsideradas no período de aquecimento do modelo
thin.t	Intervalos entre as avaliações da função de verossimilhança utilizados na composição da amostra
Rthres	Diagnóstico de Gelman e Rubin

4.5 - AVALIAÇÃO DA CONVERGÊNCIA

A avaliação da convergência, ou a definição de quando interromper o processamento da cadeia é um tópico de grande importância e tem sido foco de pesquisas no âmbito de métodos MCMC (Martinez & Martinez, 2002). Como o maior interesse está em obter amostras das distribuições posteriores e explorar as suas estatísticas, se a série obtida não tiver convergido para distribuição posterior, todas as estimativas e inferências realizadas serão inconsistentes.

Martinez & Martinez (2002) comentam que alguns métodos para avaliar a convergência são aplicáveis apenas a algoritmos específicos, tais como o Algoritmo MH ou ainda o algoritmode Gibbs. Cowles & Carlin (1996) apresentaram um comparativo entre diferentes métodos estatísticos de diagnóstico de convergência de cadeias de Markov. Neste trabalho, serão descritas, a seguir, algumas alternativas para avaliação da convergência, dentre elas o método proposto por Gelman & Rubin (1992) que pode ser aplicado em qualquer algoritmo MCMC (Martinez & Martinez, 2002).

4.5.1 - Avaliação visual da convergência

Uma forma relativamente intuititva de avaliação da convergência consiste em uma aferição visual do comportamento do traço da cadeia. A maneira como os valores da cadeia estão se movendo ou se misturando ao longo do espaço de domínio do parâmetro consiste em um fator de interesse que pode fornecer informações importantes a respeito da convergência da cadeia. A partir de uma análise visual pode-se buscar inferir o momento a partir do qual a cadeia atinge uma estabilidade e passa a oscilar entorno de determinados valores extremos.

Outra forma visual, porém, baseada em uma métrica definida, de se verificar a convergência consiste na visualização dos valores da média ou variância da cadeia ao longo dos passos de iteração. Espera-se que em situações onde a cadeia atingiu a convergência a média ou a variância se estabilizem entorno de um valor.

4.5.2 - Autocorrelação - Lag

Outro procedimento importante para avaliar a convergência consiste na análise da autocorrelação dos valores da cadeia da distribuição posterior. É esperado que toda cadeia de Markov gerada apresente autocorrelação, contudo, a amostra da distribuição posterior não deve apresentar correlação. Dessa forma, verificar a autocorrelação consiste em procedimento importante quando se empregam as cadeias de Markov. A título de exemplificação mostra-se a seguir o procedimento usualmente empregado para verificação da autocorrelação em uma cadeia de Markov.

Consideram-se os valores de parâmetro θ gerados por uma cadeia ao longo dos passos i, conforme indicado na Tabela 4.2, que apresenta um caso hipotético de uma cadeia com 6 passos. As duas últimas colunas da direita da tabela mostram o atraso ou defasagem de um passo (lag1) e dois passos (lag 2).

A autocorrelação entre os valores da cadeia na sequência natural e aqueles com o atraso de um passo (autocorrelação com *lag*1) indicará como os valores gerados pela cadeia estão relacionados com seus valores imediatamente anteriores, enquanto a autocorrelação entre os valores da cadeia na sequência natural e aqueles com um atraso de dois passos (autocorrelação com *lag2*) fornecerá a relação entre os valores da série com aqueles valores posicionados a dois passos anteriores na cadeia.

Dessa forma, o coeficiente de autocorrelação ρ_k, com *lag* k, refere-se à correlação entre cada amostra da cadeia e aquela amostra com um atraso ou defasagem (*lag*) k, ou seja:

$$\rho_k = \frac{\sum_{i=1}^{n-k}((\theta_i - \bar{\theta}))(\theta_{i+k} - \bar{\theta})}{\sum_{i=1}^{n}(\theta_i - \bar{\theta})^2} \tag{4.8}$$

Tabela 4.2- Esquema para exemplificar a avaliação de autocorrelação entre os valores da cadeia

Passo na cadeia(i)	Valor na Cadeia	Valor com lag 1	Valores com lag 2
1	θ_1	-	-
2	θ_2	θ_1	-
3	θ_3	θ_2	θ_1
4	θ_4	θ_3	θ_2
5	θ_5	θ_4	θ_3
6	θ_6	θ_5	θ_4

Espera-se que o valor da autocorrelação diminua a medida que o valor de k aumente. Caso o valor da autocorrelação ρ_k se mantenha elevado (próximos a unidade), mesmo para valores altos de k, há uma indicação de alto grau de correlação entre os valores da cadeia e, portanto, uma mistura inadequada dos valores da cadeia. A Figura 4.3 indica o comportamento esperado para os valores de correlação para diferentes defasagens (lags) para uma cadeia hipotética com sinais de convergência.

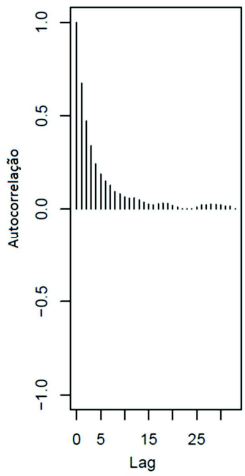

Figura 4.3 - Valores obtidos para a autocorrelação para diferentes defasagens (lags) k , indicando mistura adequada dos valores da cadeia

4.5.3 - Diagnóstico de Gelman-Rubin

O diagnóstico de convergência de Gelman-Rubin (Gelman, 1996) necessita da simulação de múltiplas cadeias. O método se baseia na ideia de que a variância dentro de uma única cadeia será menor do que a variância nas sequências de cadeias combinadas, caso a convergência não tenha sido atingida (Martinez & Martinez, 2002).

Os passos para a aplicação do método podem ser resumidos na sequência descrita a seguir:

i. Processam-se $m \geq 2$ cadeias com comprimento 2n, a partir de diferentes valores de partida para a cadeia;
ii. Descartam-se os primeiros n valores de cada cadeia (*burn in*);
iii. Calcula-se média das variâncias de cada cadeia (W) e entre cadeias (B);
iv. Calcula-se a variância estimada para a distribuição estacionária do parâmetro $\widehat{\mathrm{Var}}(\theta)$ como uma soma ponderada das variâncias médias dentro das cadeias (W) e entre as cadeias (B/n).
v. Calcula-se o fator potencial de redução de escala (R).

A média das variâncias dentro das cadeias (W) refere-se à média das variâncias de cada cadeia (s_j^2), a seguir, apresentam-se as expressões referentes a esses parâmetros, em que i refere-se a posição dentro de uma cadeia j refere-se ao índice que representa o número da cadeia de interesse, n refere-se a extensão de uma cadeia e m o total de cadeias consideradas.

$$W = \frac{1}{m}\sum_{j=1}^{m} s_j^2 \tag{4.9}$$

$$s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\theta_{ij} - \bar{\theta}_j)^2 \tag{4.10}$$

A variância média entre as cadeias (B/n) é dada por:

$$\frac{B}{n} = \frac{1}{m-1}\sum_{j=1}^{m}(\bar{\theta}_j - \bar{\bar{\theta}})^2 \tag{4.11}$$

$$\bar{\bar{\theta}} = \frac{1}{m}\sum_{j=1}^{m}\bar{\theta}_j \tag{4.12}$$

Em que $\overline{\overline{\theta}}$ representa a média das médias dos parâmetros das cadeias.

A estimativa da variância da distribuição estacionária é obtida a partir de uma média ponderada de W e B, dada por:

$$\widehat{\mathrm{Var}}(\theta) = \left(1 - \frac{1}{n}\right) W + \frac{B}{n} \tag{4.13}$$

Dessa forma, calcula-se o fator potencial de redução de escala R de Gelman & Rubin (1992) dado por:

$$R = \sqrt{\frac{\widehat{\mathrm{Var}}(\theta)}{W}} \tag{4.14}$$

Segundo os autores, quando o valor de R é alto (acima de 1,2 ou 1,3) deve-se processar a cadeia para mais iterações ou modificar as distribuições de proposição com o intuito de melhorar a convergência para a distribuição estacionária. Gelman & Rubin (1992) sugerem ainda que as cadeias sejam iniciadas com base na moda da distribuição posterior (utilizando métodos de otimização).

4.6 - FUNÇÃO DE VEROSSIMILHANÇA

Considere uma amostra independente de uma variável aleatória X, dada por $x_1...,x_n$, cuja função de densidade de probabilidade é dada por $f(x;\theta)$. A função de densidade conjunta com base nos valores da amostra é denominada função de verossimilhança. A função de verossimilhança consiste em uma função dos parâmetros de um modelo estatístico que permite inferir sobre o seu valor a partir de um conjunto de observações. Dessa forma, sendo $\vec{\boldsymbol{\theta}}$, o vetor de parâmetro da função de verossimilhança L é dada por:

$$L(\theta; x_1, \dots, x_n) = f(x_1; \theta) \times \dots \times f(x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta) \tag{4.15}$$

Para o caso da simulação hidrológica, assumindo-se inicialmente a hipótese de homocedasticidade (variância constante) e independência dos erros (ε) do modelo, estes podem ser representados por uma distribuição normal com média 0 e desvio padrão σ, dado por:

$$\varepsilon \sim N(0,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{1}{2}\left(\frac{\varepsilon}{\sigma}\right)^2\right] \tag{4.16}$$

Aplicando-se o produtório sobre esta função densidade de probabilidade obtêm-se a verossimilhança L, dada por $P(Q|\theta)$, na notação adotada para o emprego em modelos hidrológicos:

$$P(Q|\theta) = (2\pi\sigma^2)^{-\frac{n}{2}} \prod_{t=1}^{n} exp\left[-\frac{[Q_{obs,t} - f(x_t,\theta)]^2}{2\sigma^2}\right] \tag{4.17}$$

em que t representa o indexador da posição da vazão observada, x_t os dados de entrada (usualmente precipitação e evapotranspiração) do modelo hidrológico f e θ representa os parâmetros do modelo hidrológico e da função de verossimilhança e n o número de erros. A Equação (4.17) pode ser reescrita na forma logarítmica assumindo a forma da função log verossimilhança, em que o operador Log, refere-se ao logartítmo neperiano, dada por:

$$LOG\big(P(Q|\theta)\big) = -\frac{n}{2} LN(2\pi\sigma^2) + \sum_{t=1}^{n} -\frac{(Q_{obs,t} - f(x_t,\theta))^2}{2\sigma^2} \tag{4.18}$$

Bates & Campbell (2001) aplicaram a abordagem bayesiana para avaliação de um modelo hidrológico australianao denominado *Australian Water Balance Model* (AWBM) considerando duas funções de verossimilhança, uma assumindo a homocedasticidade e independência dos erros (tal como apresentada acima) e outra assumindo a heterocedasticidade (variância variável) e dependência dos erros. Os resultados dos estudos desses autores indicaram extrema sensibilidade das distribuições posteriores dos parâmetros à função de verossimilhança adotada, além de apontar a não constância da variância dos resíduos. Tal ocorrência que foi relacionada ao aumento da variância das curvas chave para vazões mais altas. Além disso, para os casos estudados pelos pesquisadores é possível observar a existência de correlação entre os parâmetros do modelo adotado.

Dessa forma, a definição da função de verossimilhança mostra-se fundamental na implementação da abordagem bayesiana. Smith *et al.* (2010) ressaltam que embora a consideração de independência e homoscedasticidade dos erros em modelos de simulação hidrológica seja frequente, essas hipóteses quase nuncam se verificam na hidrologia, exceto em casos muito particulares. Os mesmos autores ressaltam ainda que embora tal fato seja de conhecimento dos pesquisadores e modeladores há certo tempo, poucos envolvidos no campo da modelagem hidrológica tem apresentado estudos que examinem e descrevam as propriedades dos resíduos dos modelos aplicados. Assim, verificam-se na literatura poucas referências relativas à formas apropriadas de selecionar funções de verossimilhança adequadas a determinados modelos e dados observados.

Smith *et al.*(2010) apresentam uma abordagem metodológica com o propósito de auxiliar a definição da função de verossimilhança. Os autores apresentam uma metodologia sequencial em que se propõe a aplicação de funções de verossimilhança específicas que levam ou não em conta a independência, a heterocedasticidade e a presença de vazões nulas no histórico de vazões disponível. Dessa forma, são realizados sucessivos testes com diferentes funções de verossimillhança que vão aumentando a complexidade de caracterização dos resíduos, a medida que os testes realizados indicaram a violação das premissas inicialmente assumidas para os erros. Schoups & Vrugt (2010) propuseram uma função de verossimilhança generalizada (GL) bastante flexível capaz de considerar a heterocedasticidade e a dependência dos reziduos em uma única função, a partir da definição de seus parâmetros. Neste trabalho, emprega-se a função de verossimilhança apresentada pelos referidos autores.

4.5.4 - Função de Verossimilhança Generalizada

Schoups & Vrugt (2010) propuseram uma função de verossimilhança generalizada (GL) possível de ser moldada de acordo com as características dos resíduos. A função GL assume que os resíduos são heterocedásticos e autocorrelacionados, contudo, ajustando-se os parâmetros da referida função é possível moldá-la a cada uma das possíveis caracterizações dos resíduos (homocedasticos/não correlacionados; heterocedasticos/não correlacionados, homocedasticos/correlacionados; heterocedasticos/correlacioandos, além da não normalidade).

Reproduz-se, a seguir, a descrição da função GL proposta por Schoups & Vrugt (2010) e, também, empregada no trabalho de Silva (2015). A função baseia-se em um modelo de regressão não linear aditivo, conforme indicado pela equação (3.2), apresentada anteriormente, com a diferença de que as vazões médias simuladas (Q_{sim}) são agora descritas por meio da saída do modelo hidrológico $f(x_t, \theta)$ multiplicado por um fator μ_t que representa o viés introduzido à vazão modelada devido aos erros nos dados observados e da estrutura do modelo, assim:

$$Q_{sim} = f(x_t, \theta) \cdot \mu_t \tag{4.19}$$

O fator μ_t varia com o tempo e pode ser modelado como função da vazão simulada a partir da seguinte equação:

$$\mu_t = exp[\mu_M \cdot f(x_t, \theta)] \tag{4.20}$$

Em que μ_M é um parâmetro que representa o viés estimado a partir dos dados. Esse parâmetro permite amplificar a não linearidade da resposta do modelo ao longo do tempo, o que pode ser válido no caso da existência de resíduos de elevada magnitude. No caso de não se considerar algum viés na vazão modelada, basta assumir $\mu_M = 0 \rightarrow \mu_t = 1$.

Os resíduos ε_t são modelados a partir de uma função de densidade de probabilidade dada por:

$$\Phi_{\mathrm{P}}(B)\varepsilon_t = \sigma_t a_t \tag{4.21}$$

em que:

$$a_t \sim SEP(0,1,\xi,\beta) \tag{4.22}$$

onde $\Phi_{\mathrm{P}}(B) = 1 - \sum_{i=1}^{p} \phi_i \beta^i$ é um polinômio autorregressivo com p parâmetros ϕ_i, B é o operador de defasagem $\left(\beta^i \varepsilon_t = \varepsilon_{t-i}\right)$, σ_t é o desvio padrão no tempo t, a_t caracteriza erros aleatórios independentes e igualmente distribuídos, com média nula e desvio padrão unitário, descritos por uma função de densidade de probabilidade exponencial assimétrica

(SEP- *Skew Exponential Power*), com parâmetros ξ (assimetria) e β (curtose), μ=0 (média) e σ=1 dada por:

$$p(a_t|\xi,\beta) = \frac{2\sigma_\xi}{\xi + \xi^{-1}}\omega_\beta exp(-c_\beta|a_{\xi,t}|^{2/(1+\beta)}) \quad (4.23)$$

Em que $a_{\xi t} = \xi^{-sign(\mu_\xi \cdot a_t)} \cdot (\mu_\xi + \sigma_\xi . a_t)$ e μ_ξ , σ_ξ, c_β, ω_β são obtidos a partir da assimetria, ξ e da curtose, β. A função de densidade de probabilidade será simétrica caso $\xi = 1$ e assimétrica positivamente ou negativamente caso $\xi \geq 1$ ou $\xi \leq 1$, respectivamente. No caso de se assumir a função de densidade de probabilidade simétrica, o modelo probabilístico assume a forma da distribuição uniforme, gaussiana ou de Laplace, adotando-se $\beta = -1, \beta = 0, \beta = 1$, respectivamente.

Dessa forma, a autocorrelação presente na série de resíduos do modelo chuva vazão é supostamente descrita pelo modelo autoregressivo (AR) dado pela Equação (4.21). A heterocedasticidade é considerada admitindo-se que o desvio padrão varie linearmente com a vazão Q_{sim}, tendo-se a seguinte expressão:

$$\sigma_t = \sigma_0 + \sigma_1 \cdot Q_{sim} \quad (4.24)$$

Em que σ_0 e σ_1 são estimados a partir dos dados. O aumento da variância dos erros com o incremento das vazões reflete as incertezas associadas a descargas mais elevadas, tendo em vista que as curvas chave usualmente são extrapoladas para vazões mais altas, sendo estabelecidas a partir de um número limitado de vazões menores.

Com base na função de densidade de probabilidade do modelo de resíduos apresentada Schoups &Vrugt (2010) derivaram a função log de verossimilhança generalizada (GL) com base nos parâmetros θ do modelo hidrológico dada por:

$$LOG\left(P(Q|\theta)\right) = \mathrm{n} \cdot LOG\left(\frac{2\sigma_\xi}{\xi + \xi^{-1}}\right) - \sum_{i=1}^{n} LOG(\sigma_t) - c_\beta \cdot \sum_{i=1}^{n} |a_{\xi,t}|^{2/(1+\beta)} \quad (4.25)$$

Neste trabalho optou-se por trabalhar com a função de verossimilhança generalizada devido a sua flexibilidade em se adaptar, a partir da variação de seus parâmetros

($\sigma_t, \xi, \beta, \phi$), à natureza esperada dos erros (independência/autocorrelação; homocedasticidade/heterocedasticidade; ou o viés da vazão modelada).

Por exemplo, caso se admita $\sigma_1 = 0$; $\phi_i = 0$; $\beta = 0$; $\xi = 1$ e μ_M=0, a função de verossimilhança generalizada passa a representar a distribuição normal, com erros admitidos homocedástico e independentes.

Conforme apresentado em Silva (2015) esta função foi implementada em R, a partir de uma versão original desenvolvida em linguagem MATLAB pelos autores (Shoups & Vrugt, 2010). A Tabela 4.3 apresenta um resumo dos parâmetros a serem calibrados conforme a função de verossimilhança adotada.

Tabela 4.3- Parâmetros das funções de verossimilhança que devem ser estimados

Modelo	**Símbolo**	**Descrição**
Normal (N)	σ	Desvio padrão do erro do modelo
Generalizada (GL)	σ_0	Heterocedastiticade: intercepto
	σ_1	Heterocedasticidade: inclinação
	β	Curtose
	ξ	Assimetria
	ϕ_i	Coeficientes de autocorrelação
	μ_M	Viés da vazão modelada

5 - METODOLOGIA

A metodologia geral de trabalho foi estruturada com base em três grandes etapas de forma a permitir alcançar os objetivos almejados. A primeira etapa consistiu na pesquisa e revisão bibliográfica a respeito dos temas de interesse.

Na segunda etapa, com base na pesquisa realizada, foram desenvolvidos e implementados em linguagem R os algoritmos MCMC (M, MH e AM), o modelo hidrológico (SMAP - Mensal) e o algoritmo PSO de busca global para calibração de modelos.

Complementarmente, foramimplementadas rotinas para avaliação da convergência das cadeias de Markov, com base no exposto no item 4.5 Avaliação da Convergência, além de rotinas gráficas para avaliação dos resultados. Destarte, foi possível atingir o primeiro objetivo específico pleiteado, uma vez que foram estruturados os códigos computacionais em linguagem R para consecução das análises previstas.

Na sequência, com base nos algoritmos implementados, foram realizadas diversas simulações para testes e verificações, tendo-se posteriormente empregado os algoritmos em dados sintéticos e reais. Desse modo, vencida as avaliações de teste para validação dos códigos desenvolvidos, ainda como parte desta segunda etapa, foram realizadas comparações entre as performances dos algoritmos MCMC estudados com fulcro no alcance do segundo objetivo específico delineado.

Para estruturação das avaliações que visaram a comparação entre os algoritmos, primeiramente, foi analisado um caso real de aplicação do modelo SMAP mensal na região do semiárido nordestino brasileiro no estado do Ceará. Nessa região, por haver uma pequena cobertura de solo e estarem localizadas em áreas de embasamento cristalino, pode-se admitir, por simplificação, a não ocorrência do escoamento de base, sendo razoável desconsiderar a influência dos parâmetros (CREC, K e EBin), que se relacionam aesse tipo de escoamento. Assim, considera-se apenas a influência dos parâmetros (SAT, PES e TUin), sendo que para o último, relacionado à taxa de umidade inicial, pode-se optar por calibrá-lo ou prefixá-lo.

Em seguida, foram avaliados mais dois casos, um com dados sintéticos e outro com dados reais, dessa vez, considerando todos os parâmetros calibráveis do modelo.

Como critério de comparação entre os algoritmos, foram considerados o tempo de processamento, bem como o número de iterações para alcance da convergência. Foram avaliados ainda o comportamento da média dos parâmetros ao longo das sucessivas iterações das cadeias, além do comportamento das próprias cadeias e das distribuições posteriores dos parâmetros estudados.

A segunda etapa, além de avaliar de forma comparativa os algoritmos MCMC implementados, buscou uma melhor compreensão do funcionamento desses métodos, investigando o efeito das distribuições adotadas *a priori* e das distribuições propostas na performance e convergência das cadeias.

Para os casos simulados, procurou-se, também, efetuar o procedimento de calibração convencional por meio do algoritmo PSO visando contrapor os resultados da abordagem bayesiana com a determinística, no que concerne a calibração de modelos.

Por fim, a terceira etapa procurou avaliar como as premissas sobre a natureza dos erros influenciam os resultados advindos da calibração, alinhando-se, portanto, com o terceiro objetivo específico pleiteado. Tendo em vista que as premissas assumidas para os erros do modelo estão sintetizadas na escolha da função de verossimilhança, foram adotadas duas funções, a saber:

- função de verossimilhança que assume a normalidade, homocedasticidade e independência entre os erros do modelo; e
- função de verossimilhança generalizada, proposta por Schoups & Vrugt (2010), que permite a consideração da não normalidade, da heterocedasticidade e da autocorrelação entre os erros do modelo.

Assim, para as análises referentes à terceira etapa, optou-se por empregar o algoritmo DREAM. Tal como descrito no item 4.4 - Algoritmo *Differential Evolution Adaptive Metropolis* (DREAM), esse algoritmo foi bastante testado na literatura científica e apresentou a capacidade de ajustar as distribuições propostas durante a evolução das

iterações e trabalhar com proposições baseadas em cadeias mistas, o que confere maior capacidade de convergência ao método.

O algoritmo permite, ainda, empregar a função de verossimilhança generalizada de Schoups & Vrugt (2010) que é, conforme descrito no item 4.5.4 - Função de Verossimilhança Generalizada, uma função bastante flexível, podendo ser ajustada às características dos resíduos a partir da configuração de seus parâmetros.

Dessa forma, os parâmetros da função GL foram configurados de tal maneira a representar a normalidade, a homocedasticidade e a independência, tendo-se após as simulações, verificado o comportamento dos resíduos do modelo confrontandos-o com a distribuição normal padrão que traduz a hipótese inicialmente assumida.

Posteriormente, foram considerados todos os parâmetros da função GL de tal forma a permitir ao modelo abarcar a não normalidade, a heterocedasticidade e a autocorrelação entre os resíduos, sendo posteriormente avaliado o comportamento dos resíduos com base nas premissas assumidas.

Os hidrogramas obtidos com base nas hipóteses inicialmente assumidas foram analisados e comparados com o propósito de verificar o ganho obtido na saída do modelo ao se ajustar aos resíduos do modelo uma função de verossimilhança que melhor represente a natureza dos resíduos.

A Figura 5.1 apresenta o fluxograma com o detalhamento de todo o ciclo de trabalho aqui resumido. Em seguida, ainda como parte desta seção apresenta-se o modelo hidrológico SMAP para passos de tempo mensais, adotado nas simulações realizadas.

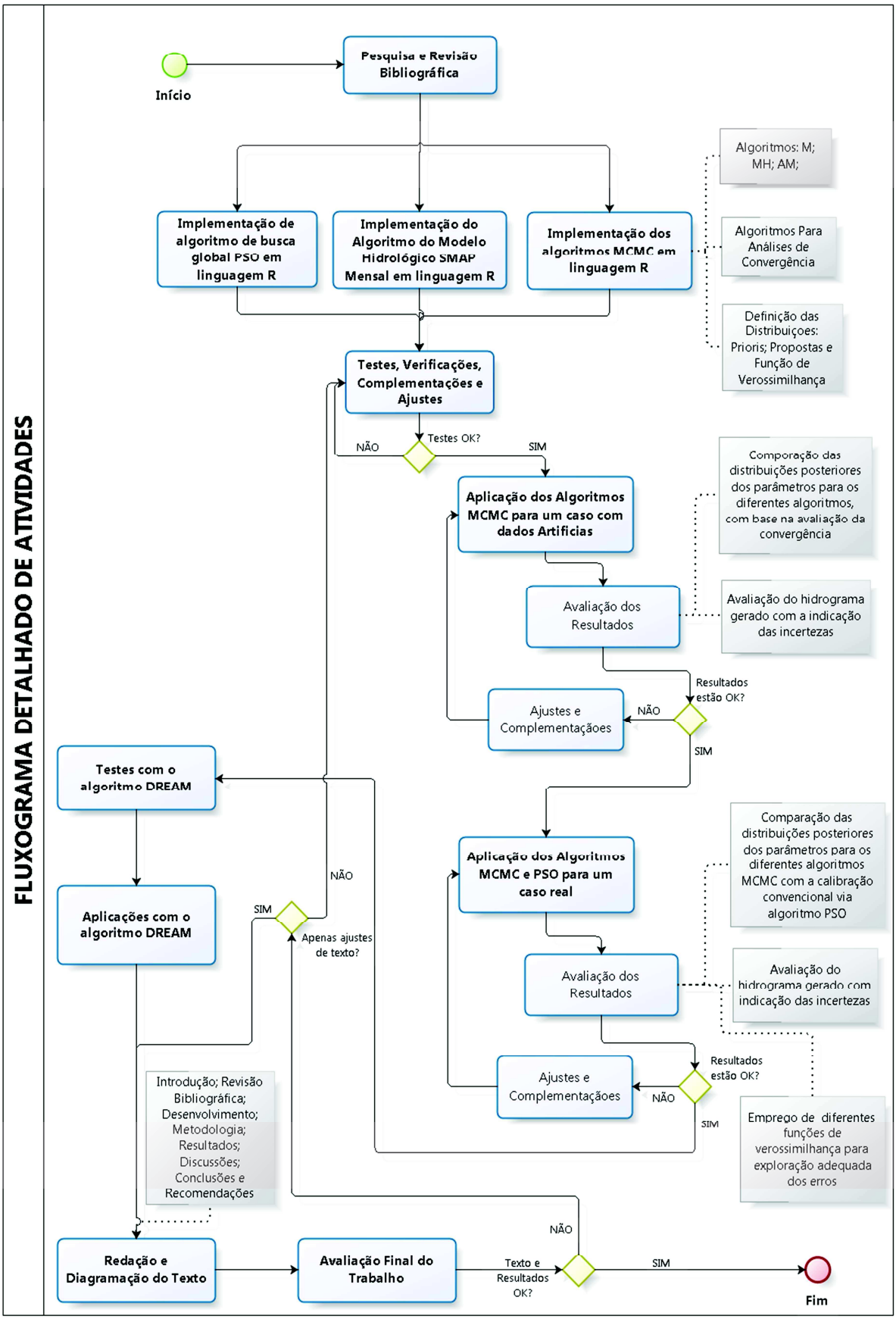

Figura 5.1 - Fluxograma de Atividades

5.1 - MODELO HIDROLÓGICO SMAP MENSAL

O modelo hidrológico SMAP (*Soil Moisture Accounting Procedure*) proposto por Lopes *et al.* (1981) para passos de tempo mensais é um modelo do tipo chuva vazão conceitual que procura representar o armazenamento e os fluxos de água na bacia através de reservatórios lineares fictícios. A estrutura do modelo é composta por dois reservatórios que buscam representar o armazenamento e os fluxos na camada superior do solo e no aquífero, como pode ser visualizado na Figura 5.2.

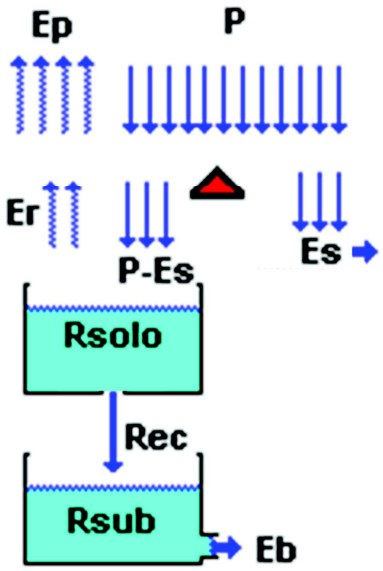

Figura 5.2 - Esquema gráfico da estrutura do modelo SMAP mensal (Nascimento *et al.,* 2009)

A cada evento de precipitação (P), realiza-se um balanço de massa na bacia hidrográfica em estudo. Uma parcela de (P) é transferida como escoamento superficial (ES), estimado por uma equação exponencial que depende de P, da taxa de umidade do solo (TU) e de um expoente (PES). A lâmina restante da precipitação, subtraída do escoamento superficial (P-ES), sofre perda por evaporação (P-ES-EP) e é então adicionada a um reservatório, o qual representa a camada superior do solo. Neste reservatório, a umidade do solo é atualizada ao longo do tempo através das perdas por evapotranspiração real (ER), que dependem do nível do reservatório (RSOLO) e da capacidade de saturação do solo (SAT). A saída deste segundo reservatório representa a recarga do reservatório subterrâneo (REC), estimada com base nataxa de umidade (TU), no nível do reservatório (RSOLO) e no coeficiente de recarga (CREC). O nível d'água no segundo reservatório (RSUB)é então deplecionado a uma taxa constante de recessão do escoamento de base (K), resultando no

escoamento de base (EB) propriamente dito. A soma de ES com EB fornece a vazão total no ponto de controle da bacia.

O modelo possui quatro parâmetros, a saber: capacidade de saturação do solo (SAT); expoente relacionado à geração de escoamento superficial (PES); o coeficiente de recarga do aquífero (CREC), que está relacionado com a permeabilidade da zona não saturada do solo; a taxa de deplecionamento (K) do nível d'água do segundo reservatório (RSUB), responsável pela geração do escoamento de base (EB). Além disso, duas condições iniciais precisam ser definidas, a taxa de umidade inicial do solo (TUin), que determina o nível inicial do segundo reservatório (RSOLO) e o escoamento de base inicial (EBin). Como dados de entrada são necessários um histórico de precipitação média mensal (em mm de chuva), evapotranspiração potencial mensal (em mm) e a área de drenagem da bacia hidrográfica modelada (em km²).

Para a região do semiárido nordestino, devido à presença do embasamento cristalino, o escoamento de base pode ser desprezado, resultando num modelo com menor número de parâmetros, já que CREC e K podem ser considerados nulos. Desta forma, nessas condições, os únicos parâmetros a serem, de fato, calibrados são o SAT e PES, embora o TUin e EBin também possa ser estimado via calibração.

ATabela 5.1 resume os parâmetros a serem calibrados no modelo hidrológico SMAP-Mensal. Para as aplicações deste trabalho, os parâmetros referentes à taxa de umidade inicial (TUin) e ao escoamento de base (EBin) serão pré-definidos para as simulações. Maiores detalhes a respeito do algoritmo do modelo SMAP-Mensal podem ser consultados diretamente no código fonte disponibilizado anexo a esta dissertação.

Tabela 5.1- Parâmetros do modelo hidrológico SMAP-Mensal

Parâmetro	**Descrição**
SAT	Capacidade de saturação do solo (mm)
PES	Expoente da equação exponencial que define o escoamento superficial
CREC	Coeficiente de recarga do reservatório subterrâneo (%)
K	Constante de recessão do escoamento de base (meses)

6 - RESULTADOS E DISCUSSÕES

Esta seção apresenta osresultados obtidos para as etapas metodológicas descritas anteriormente. Os resultados foram estruturados com o propósito de apresentar as análises referentes às comparações entre as performances dos algoritmos MCMC implementados e a avaliação da influência da função de verossimilhança nas distribuições posteriores dosparâmetros do modelo.

6.1 - COMPARAÇÃO ENTRE OS ALGORITMOS METROPOLIS (M), METROPOLIS HASTINGS (MH) E ADAPTIVE METROPOLIS (AM)

A comparação entre as performances dos algoritmos *Metropolis* (M), *Metropolis Hastings* (MH) e *Adaptive Metropolis* (AM) foi realizada para três casos, tendo sido também realizado o procedimento de calibração convencional, tal como descrito na metodologia do trabalho.O primeiro caso consistiu em uma aplicação a uma situação com dados reais e dimensão reduzida dos parâmetros do modelo hidrológico (apenas os parâmetros SAT e PES considerados calibráveis), tendo em vista tratar-se de uma bacia localizada na região do semiárido nordestino. Para a função de verossimilhança, que considerou a normalidade, homocedasticidade e independência dos resíduos, foi calibrado o parâmetro associado ao desvio padrão do erro do modelo (σ).

O segundo e terceiro casos consideraram calibráveis todos os parâmetros do modelo hidrológico (SAT, PES, CREC, K), além do desvio padrão do erro do modelo (σ), tendo sido pré-fixados os valores da taxa de umidade inicial (TUin) e escoamento de base inicial (EBin). O segundo caso, especificamente, referiu-se a uma situação com dados sintéticos de vazões e erros gerados artificialmente, a partir de dados de evapotranspiração e chuva para uma sub-bacia do rio Tapajós. O terceiro caso consistiu em uma aplicação a uma sub-bacia do rio São Francisco com dados reais de chuva, evapotranspiração e vazões observadas.

A apresentação dos resultados para cada um dos casos simulados é realizada inicialmente descrevendo-se os dados de entrada para as simulações, seguido das distribuições adotadas *a priori* e das distribuições propostas para cada um dos algoritmos implementados,

posteriormente são comparadas as distribuições posteriores obtidas para os parâmetros calibrados além do comportamento dos resíduos, que são assumidos normais, homocedásticos e independentes para os três casos apresentados nesta seção.

6.1.1 - Caso real com dimensão reduzida dos parâmetros

Conforme descrito na metodologia anteriormente, este caso aplicou os algoritmos M, MH e AM em uma bacia do semiárido nordestino tendo sido avaliados os parâmetros (SAT) e (PES), além do desvio padrão do erro do modelo (σ). Complementarmente, também foi realizada a calibração convencional com o emprego do algoritmo PSO, seguido da aplicação do método Simplex desenvolvido por Nelder & Mead (1965). A seguir descrevem-se as características da bacia adotada bem como as distribuições, a priori, propostas e os resultados das análises obtidas.

6.1.1.1. - Dados de entrada

Os dados utilizados para simulação deste caso consistiram nos históricos de precipitação, evaporação e vazões médias mensais para o período de setembro/1990 a setembro/1998, para a estação hidrológica Moraújo (Cód. 35125000), localizada no rio Coreaú, no Município de Moraújo, estado do Ceará, com área de drenagem de 1.500 km². O rio Coreaú pertence a sub-bacia 35, dos rios Acara, Pirangi e tributários, que por sua vez pertencem a bacia 3 do Atlântico, trecho Nordeste Oriental. A estação encontra-se às coordenadas: 3º 27' 53" de Latitude Sul e 40º 41' e 7" de Longitude Oeste. A Figura 6.1 apresenta a localização da estação Moraújo com um marcador triangular vermelho.

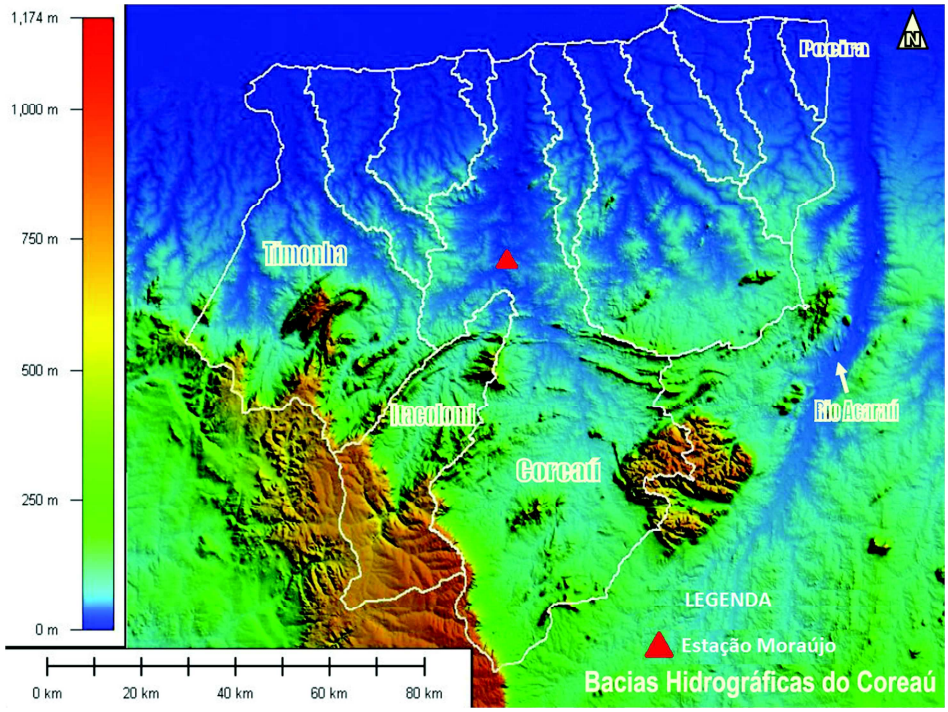

Figura 6.1 - Localização da Estação Fluviométrica Moraújo (adaptado, COGER, 2009)

6.1.1.2. - Distribuições *a priori* e limites de busca dos parâmetros

Para o primeiro caso, nas simulações realizadas com todos os algoritmos (M, MH e AM) optou-se por adotar distribuições *a priori* não informativas dadas por distribuições uniformes. Os limites de variação assumidos para essas distribuições em função dos respectivos parâmetros são apresentados na Tabela 6.10. Para a calibração convencional os limites de busca dos parâmetros do modelo hidrológico foram equivalentes, excetuando-se os do parâmetro referente ao desvio padrão do erro do modelo que não é considerado no procedimento convencional de calibração.

Os valores dos limites indicados foram estabelecidos após diversos procedimentos de calibração convencional (empregando o algoritmo PSO seguido do método Simplex desenvolvido por Nelder & Mead (1965) e inúmeras simulações que avaliaram a sensibilidade do modelo hidrológico.

Os parâmetros referentes à taxa de umidade inicial e ao escoamento de base inicial (TUin, Ebin) foram fixados nos valores 0,4 e 0 m^3/s, respectivamente. Esses valores foram adotados com base em simulações convencionais de calibração que consideraram esses parâmetros calibráveis.

Tabela 6.1- Limites das distribuições uniformes adotadas *a priori* para os parâmetros de interesse e para a busca via algoritmo PSO no caso real com dimensão reduzida de parâmetros

Parâmetro	**Distribuição uniforme**	
	Mínimo	**Máximo**
SAT	500	5000
PES	0,5	6
σ	1	30

6.1.1.3. - Distribuições propostas

É oportuno destacar que não há indicações explícitas a respeito de qual tipo de distribuição proposta deve ser adotada, quando se emprega, por exemplo o algoritmo Metropolis. Contudo, tal como comentado em Marshall (2004) e Bates & Campbell (2001) e verificado nos testes realizados, a variância da distribuição de proposição afeta diretamente a performance das cadeias geradas. Se a variância é relativamente pequena, o processo iterativo induzirá a formação de uma cadeia limitada a um espaço de busca reduzido, delongando a convergência. Por outro lado, se a variância é demasiadamente grande os valorespropostos serão rejeitados com maior frequência, também dificultando a convergência. Variâncias da distribuição de proposição que possuam a mesma ordem de grandeza das distribuições posteriores costumam refletir em taxas de aceitações razoáveis (entre 20 a 70%) (Bates & Campbell, 2001). Como não se sabe a distribuição posterior dos parâmetros, o ajuste da variância da distribuição de proposição acaba sendo realizado por um processo iterativo após sucessivas simulações. A forma adotada para melhorar a maneira de propor os parâmetros por diversos autores consiste em utilizar o valor do passo corrente da cadeia como a média da distribuição de proposição. Assim, com base nessas considerações, foram definidos e ajustados os parâmetros das distribuições de proposição para os algoritmos empregados.

Conforme discutido na Seção 4.2-Algoritmo *Metropolis-Hastings*, o algoritmo *Metropolis* consiste em um caso particular do algoritmo *Metropolis-Hastings*, quando as distribuições de proposição são simétricas. Dessa forma, para as simulações referentes ao algoritmo *Metropolis* foram adotadas distribuições propostas simétricas. Para os parâmetros (SAT e PES) foram consideradas distribuições normais, com média da distribuição igual ao valor do parâmetro do modelo no passo corrente da cadeia de Markov e desvio padrão de 100

(SAT) e 0,1 (PES). Para o desvio padrão do erro do modelo (σ) adotou-se uma distribuição uniforme cujos intervalos de proposição foram redefinidos a cada passo da iteração como a soma e a subtração do valor proposto e sua metade. A Tabela 6.11 apresenta um resumodas distribuições de proposição adotadas para as simulações com o algoritmo *Metropolis*.

Tabela 6.2- Distribuições propostas consideradas para o Algoritmo *Metropolis* no caso real com dimensão reduzida de parâmetros

Parâmetro	**Distribuição**	**Parâmetros**
SAT	Normal	Média=$SAT_{(i)}$; Desvio Padrão=100
PES	Normal	Média=$PES_{(i)}$; Desvio Padrão=0,1
σ	Uniforme	Limite Superior= $\sigma_{(i)} + \sigma_{(i)}$ /2; Limite Inferior= $\sigma_{(i)} - \sigma_{(i)}$ /2

(i) - Representa o passo corrente da cadeia

Para a simulação com o algoritmo *Metropolis-Hastings* foi necessário ajustar as propostas para garantir o emprego de distribuições não simétricas, caso contrário, a situação recairia em um caso do algoritmo *Metropolis* novamente. Para o parâmetro (SAT) adotou-se uma distribuição Log-Normal, com média dada pelo logaritmo Neperiano do valor do parâmetro do modelo no passo corrente da cadeia e desvio padrão de 0,1. As propostas para o parâmetro (PES) foram realizadas por meio de uma distribuição Gama, com parâmetros de forma (α) e de escala (β). O parâmetro (α) da distribuição foi considerado como o produto entre o valor do parâmetro do modelo (PES) no passo corrente da cadeia e o parâmetro da distribuição (β), enquanto o parâmetro (β) foi dado pela razão entre a constante 2.500 e o valor do parâmetro do modelo (PES) no passo corrente da cadeia.

Para a distribuição de proposição do desvio padrão do erro do modelo (σ) optou-se por trabalhar com o parâmetro de precisão (ϕ), dado pelo inverso da variância do erro do modelo ($1/\sigma^2$). Assim, foram obtidas as proposições para o parâmetro (ϕ) e posteriormente os valores foram transformados no desvio padrão do erro do modelo ($\sigma=1/\phi^{0,5}$). A distribuição proposta adotada para o parâmetro de precisão (ϕ) também foi a distribuição Gama, com parâmetro (α) dado pelo produto entre o parâmetro de precisão do modelo (ϕ) no passo corrente da cadeia e o parâmetro (β) da distribuição, enquanto oparâmetro (β) da distribuição foi obtido a partir da razão entre a constante 2025 e o valor do parâmetro de precisão do modelo (ϕ) no passo corrente da cadeia. ATabela 6.3 apresenta um resumo das distribuições de propostas consideradas para as simulações com o Algoritmo *Metropolis-*

Hastings. Destaca-se que os valores dos parâmetros citados foram ajustados após inúmeras simulações testes.

Tabela 6.3- Distribuições propostas consideradas para o Algoritmo *Metropolis Hastings* no caso real com dimensão reduzida de parâmetros

Parâmetro	Distribuição	Parâmetros
SAT	Log-Normal	Média=LN[$SAT_{(i)}$] ; Desvio Padrão=0,1
PES	Gama	$\alpha = PES_{(i)} X \beta$; β=2.500/ $PES_{(i)}$
ϕ	Gama	$\alpha = \phi_{(i)} X \beta$; β=2.025/ $\phi_{(i)}$

(i) - Representa o passo corrente da cadeia; $\sigma = 1/ \phi^{0,5}$

Referente ao algoritmo AM, tal como descrito no item 4.3- Algoritmo *Adaptive Metropolis*, a distribuição de proposição é realizada por meio de uma distribuição multivariada com média dada pelos valores dos parâmetros no passo corrente da cadeia e uma matriz de covariância C_i, que assume um valor fixo até um número inicial i_0 de iterações, a partir do qual a matriz passa a ser atualizada a cada passo da iteração conforme Equação (4.3).

Assim, adotou-se para os 600 passos iniciais da cadeia uma matriz de covariância constante dada pela diagonal principal formada por valores fixos das variâncias dos parâmetros (SAT, PES e σ) e os demais elementos da matriz nulos. Para a diagonal principal da referida matriz adotaram-se os seguintes valores: 900; 0,0064; 225, também definidos a partir de tentativas após sucessivas simulações.

6.1.1.4. - Avaliação do desempenho dos algoritmos e incertezas nos parâmetros do modelo

Com base nas considerações descritas nas seções anteriores foram simuladas duas cadeias de Markov para cada um dos algoritmos totalizando 150.000 iterações. Os valores iniciais da primeira cadeia foram estabelecidos como aqueles obtidos pela calibração convencional com o algoritmo PSO, sendo 1.533,8 (SAT), 3,11 (PES) e 4,42 (σ). Para a segunda cadeia foram adotados os seguintes valores iniciais: 1.000 (SAT), 2 (PES) e 6 (σ). A Figura 6.10 e a Figura 6.3 apresentam o comportamento das cadeias de Markov (gráficos superiores) e das distribuições posteriores (gráficos inferiores) obtidas a partir dos algoritmos simulados. Observa-se que os resultados indicam equivalência das distribuições posteriores obtidas

com os diferentes algoritmos MCMC simulados, tendo em vista a sobreposição das linhas das diferentes cadeias resultantes. Nas figuras citadas apresenta-se ainda com uma linha tracejada azul o valor do parâmetro decorrente do procedimento convencional de calibração. Verifica-se que este valor encontra-se próximo a média da distribuição posterior encontrada.

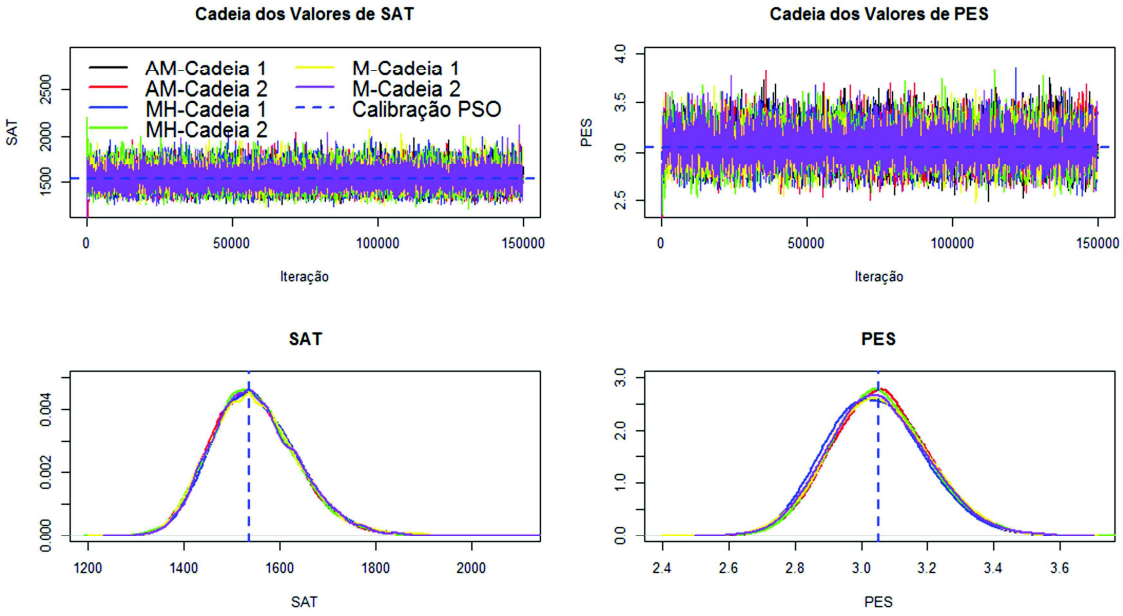

Figura 6.2 - Cadeias e distribuições posteriores para os parâmetros SAT e PES para os algoritmos AM, MH e M e valor obtido por meio da calibração convencional com o algoritmo PSO

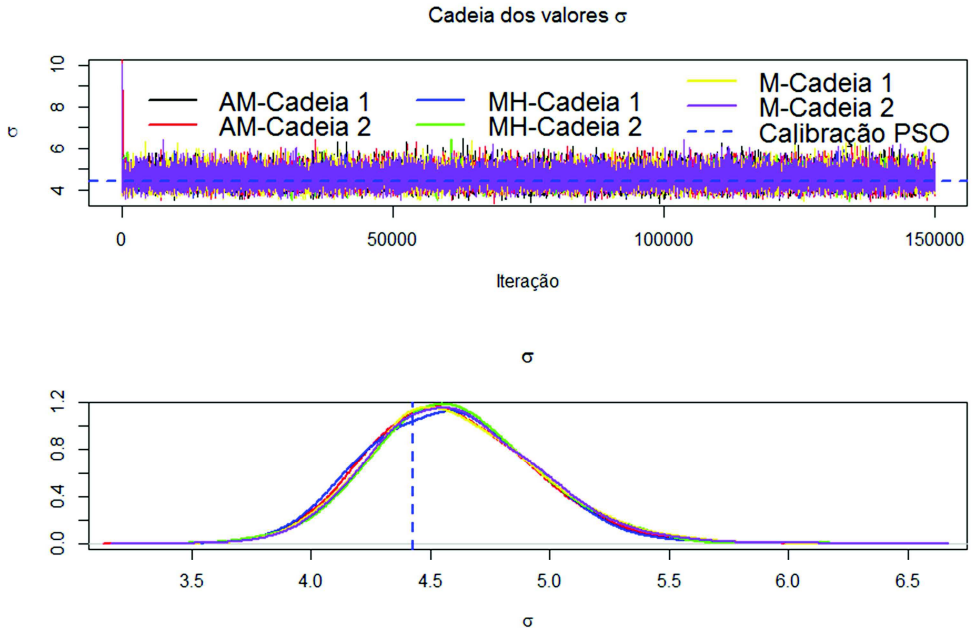

Figura 6.3 - Cadeias e distribuições posteriores para o desvio padrão do erro do modelo (σ) e valor obtido por meio da calibração convencional com o algoritmo PSO

A Tabela 6.4 resume a avaliação comparativa entre as performances dos algoritmos avaliados. As duas primeiras colunas indicam o respectivo algoritmo e as duas cadeias simuladas, respectivamente. A terceira coluna apresenta a média das cadeias descartados os 60.000 primeiros valores para cada um dos 3 parâmetros (SAT, PES e σ). A quarta coluna apresenta para cada um dos parâmetros intervalos de credibilidade de 95%. As três colunas finais mostram respectivamente, a taxa de aceitação, o tempo de processamento para as 150.000 iterações e o fator potencial de redução de escala, denominado fator R. Este último, trata-se do parâmetro referente ao diagnóstico de convergência de Gelman & Rubin (1992), conforme apresentado na Equação (4.14).

Verifica-se que os valores obtidos para os parâmetros (SAT, PES e σ), por meio da calibração convencional com o algoritmo PSO encontram-se dentro do intervalo de credibilidade das distribuições posteriores encontradas a partir de todas as cadeias processadas. É notório ainda observar que as cadeias simuladas por meiodo algoritmo MH apresentaram taxas de aceitação da ordem de 42%, sendo relativamente superiores àquelas obtidas com os demais algoritmos (M e AM) que ficaram entre 14% e 16%.

Tabela 6.4- Resumo da análise de convergência e comparação entre os algoritmos M, MH e AM para o caso real com parâmetros reduzidos

Algoritmo	Cadeia	Média dos Parâmetros			Intervalo de Credibilidade de 95%			Taxa de Aceitação	Tempo de Processamento (min) *Processador Intel-Core i5	Fator R		
		SAT	PES	σ	SAT	PES	σ			SAT	PES	σ
M	1	1.549,7	3,05	4,60	[1.378,0 ; 1.730,9]	[2,77 ; 3,36]	[3,95 ; 5,29]	14,2%	31,6	1,000096	1,000279	1,000246
	2	1.549,7	3,05	4,61	[1.388,7 ; 1.734,4]	[2,76 ; 3,34]	[3,68 ; 5,29]	14,0%	33,6			
MH	1	1.551,1	3,04	4,58	[1.387,2 ; 1.735,9]	[2,76 ; 3,33]	[3,93 ; 5,21]	42,1%	33,6	1,000065	1,00023	1,007297
	2	1.545,7	3,06	4,60	[1.376,3 ; 1.720,0]	[2,77 ; 3,34]	[3,97 ; 5,32]	42,2%	43,6			
AM	1	1.548,6	3,06	4,58	[1.376,6 ; 1.729,1]	[2,77 ; 3,35]	[3,93 ; 5,25]	16,6%	68,0	1,000528	1,000178	1,000263
	2	1.546,4	3,06	4,58	[1.376,2 ; 1.721,0]	[2,77 ; 3,34]	[3,94 ; 5,24]	15,6%	66,9			
PSO	-	1.533,8	3,11	4,42	-			-	0,53	-		
NS	-	0,90			-			-	-	-		

Referente ao tempo de processamento, constata-se que para o caso simulado os algoritmos M e MH tiveram tempos equivalentes, da ordem de 33 min, com um pequeno aumento para a segunda cadeia processada com o algoritmo MH, que resultou em um tempo de simulação de cerca de 43 min. O algoritmo AM levou mais do dobro do tempo de processamento para as duas cadeias simuladas (cerca de 68 min e 67 min, respectivamente).

Referente ao desempenho dos algoritmos, embora tenha sido observada alguma melhora na taxa de convergência do algoritmo MH para o caso simulado, destaca-se que, conformerelatado anteriormente, a distribuição de proposição afeta sobremaneira as taxas de aceitação do método sendo, portanto, um fator que pode influenciar significativamente neste aspecto.

Assim, ainda que o algoritmo AM efetue as propostas de parâmetros sempre com base em uma distribuição normal multivariada, verificou-se nas simulações efetuadas na fase de testes dos algoritmos que o número de passos iniciais estabelecidos para a matriz inicial de covariância, bem como os valores adotados para esta matriz afetam consideravelmente a taxa de aceitação do método. Da mesma forma, também foi possível observar na fase de testes dos algoritmos que as distribuições adotadas *a priori*, bem como as distribuições propostas no caso dos algoritmos M e MH influenciam sobremaneira a taxa de convergência das cadeias.

Avalia-se, portanto, que sob este aspecto é difícil estabelecer um parâmetro equivalente de comparação entre os algoritmos, tendo em vista que são realizadas proposições por distribuições diferentes e que dependendo dosparâmetros adotados podem haver melhora ou redução das taxas de aceitação dos algoritmos. Destaca-se que mesmo no caso do algoritmo MH e M, onde seria possível adotar distribuições *a priori* e propostas equivalentes, torna-se difícil a comparação entre eles sob um mesmo referencial, uma vez que ao se adotar distribuições propostas simétricas o algoritmo MH recai no próprio algoritmo M.

Concernente ao parâmetro de avaliação da convergência (Fator R), observa-se que para todas as cadeias foram obtidos valores inferiores ao limite de 1,01, podendo-se assumir que houve convergência para todas as simulações. Outra avaliação realizada para a

verificação da convergência consistiu na análise do comportamento das médias das cadeias em função do incremento de passos simulados, conforme apresentado na Figura 6.4. É possível perceber por esta avaliação que a partir de cerca de 60.000 iterações, as médias das cadeias tendem a se estabilizar entorno de um valor.

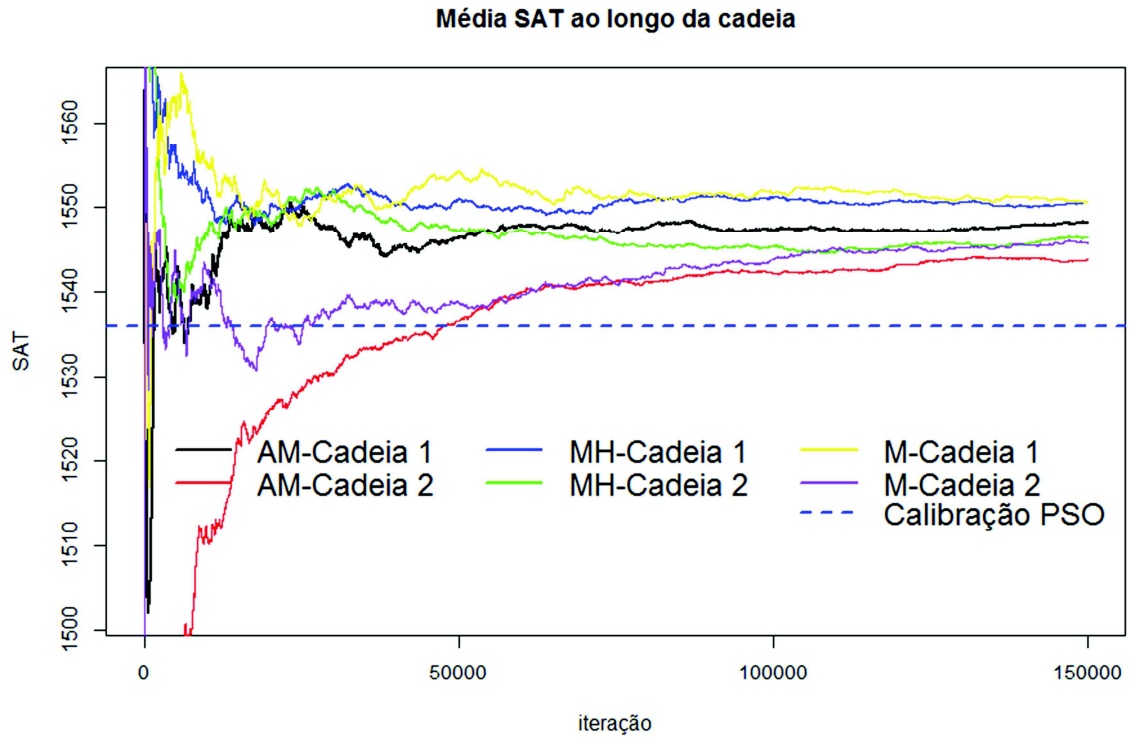

Figura 6.4 - Evolução das médias das cadeias para os diferentes algoritmos processados para o parâmetro SAT

Como foi verificada a equivalência dos resultados para as cadeias simuladas, optou-se por apresentar o diagrama de dispersão para os parâmetros referentes apenas à Cadeia 1 do algoritmo MH. A Figura 6.5 apresenta o gráfico obtido onde se verifica um comportamento que indica forte correlação entre os parâmetros (SAT) e (PES). As linhas traçadas no gráfico indicam regiões com quantidade de ocorrência de valores semelhantes, tratam-se de isolinhas, com os valores das curvas representando a densidade de pontos. A graduação da cor amarela para vermelho mostra o aumento na intensidade de ocorrência de valores.

Observa-se uma correlação inversa entre os valores de (SAT) e (PES), ou seja, maiores valores do parâmetro (SAT) correspondem a valores menores do parâmetro (PES). Uma

explicação física para tal ocorrência está no fato de que maiores valores de (SAT) significam maior capacidade de armazenamento do solo levando a uma redução da quantidade de escoamento superficial, representada pela redução do parâmetro (PES). O estabelecimento de relações dessa natureza é de grande valia em estudos de regionalização, quando se busca a calibração de modelos em locais sem dados.

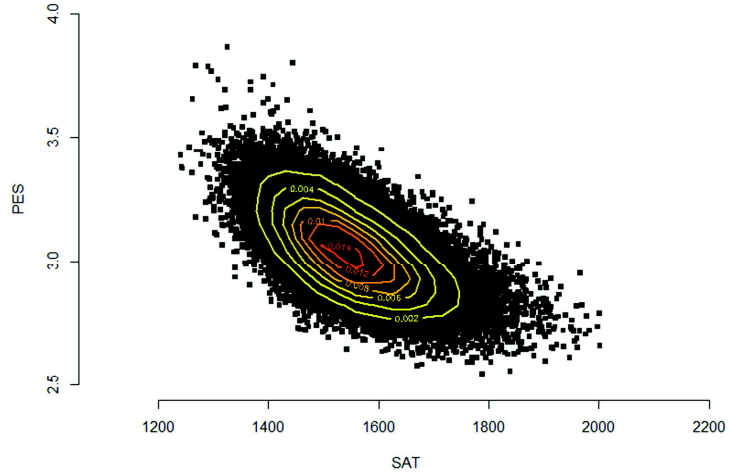

Figura 6.5 - Correlação entre os parâmetros SAT e PES para a cadeia 1 do algoritmo MH

6.1.1.5. - Avaliação dos resíduos

A análise dos resíduos buscou verificar as hipóteses assumidas para a natureza dos erros considerados normais, homocedásticos e independentes. A primeira avaliação consistiu na verificação da normalidade onde se comparou a frequência de distribuição dos resíduos padronizados com a distribuição normal padrão. A Figura 6.6 apresenta com uma linha contínua vermelha os resíduos padronizados do modelo e com uma linha preta tracejada a distribuição normal padronizada. É nítido o contraste entre o comportamento dos resíduos revelando que a hipótese de normalidade é violada. Outra avaliação que comprova a impossibilidade de se admitir a normalidade para descrição dos erros trata-se do gráfico quantil-quantil apresentado na Figura 6.7 onde é possível verificar que para os valores extremos de vazão os pontos não se alinham com a reta referente à normalidade dos resíduos.

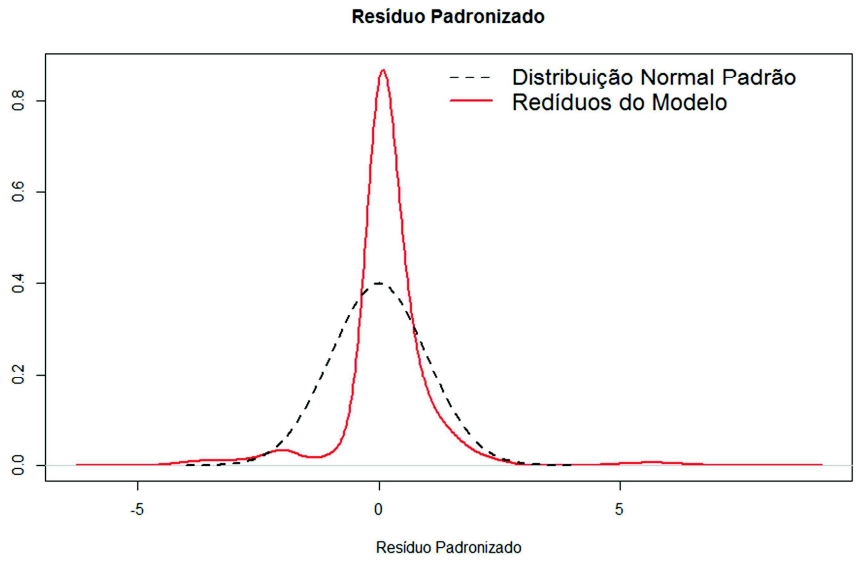

Figura 6.6–Comportamento da média dos resíduos em função das vazões médias simuladas

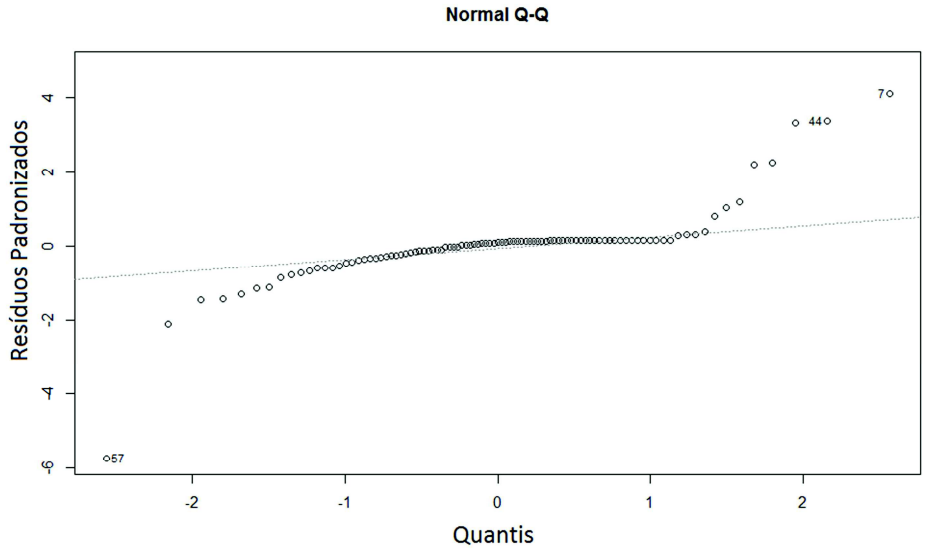

Figura 6.7–Avaliação Quantil-Quantil

Para avaliação da homocedasticidade foi gerado um diagrama de dispersão entre a média dos erros do modelo e a média das vazões simuladas. A Figura 6.8 demonstra o comportamento heterocedástico dos resíduos uma vez que fica claro a tendência de aumento da variância e dispersão dos resíduos a medida que os valores de vazões ficam mais elevados. Como há a ocorrência de erros mais representativos nos tramos superiores dos ajustes das curvas chaves, espera-se a ocorrência desse comportamento nas séries geradas para as estações fluviométricas a partir das curvas chave ajustadas.

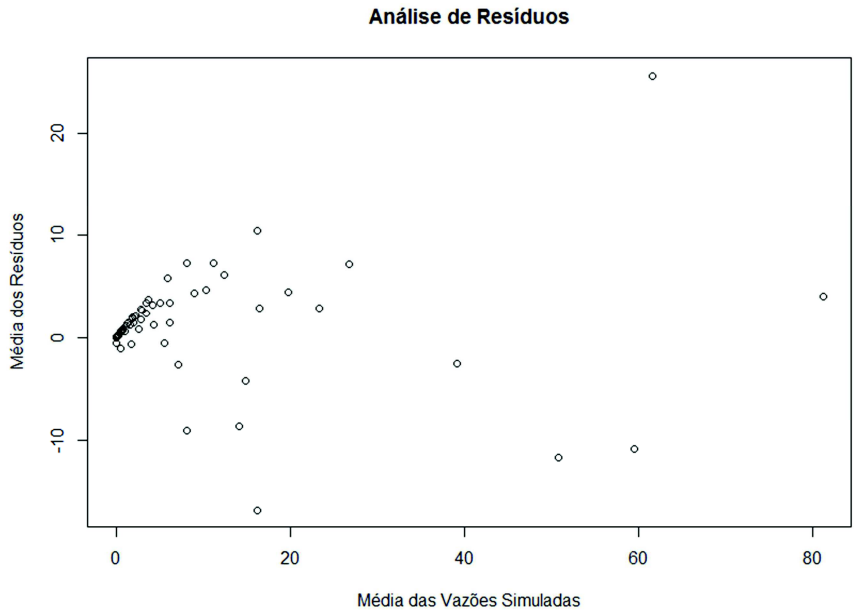

Figura 6.8–Média dos resíduos em função das vazões simuladas

Com o propósito de verificar a independência dos erros, foram calculados os valores de autocorrelação para diferentes lags, conforme apresentado em seção anterior, no item 4.5.2- Autocorrelação-Lag. A Figura 6.9 apresenta os valores de autocorrelação (ACF) para diferentes lags. Nos resultados obtidos não é possível constatar a presença de forte dependência entre os resíduos, exceto para o primeiro lag. Contudo, destaca-se que Silva *et al.* (2014) em simulação de modelo hidrológico com passos de tempo diário identificaram forte correlação entre os erros do modelo. Acredita-se que a escala temporal pode influenciar significativamente neste quesito.

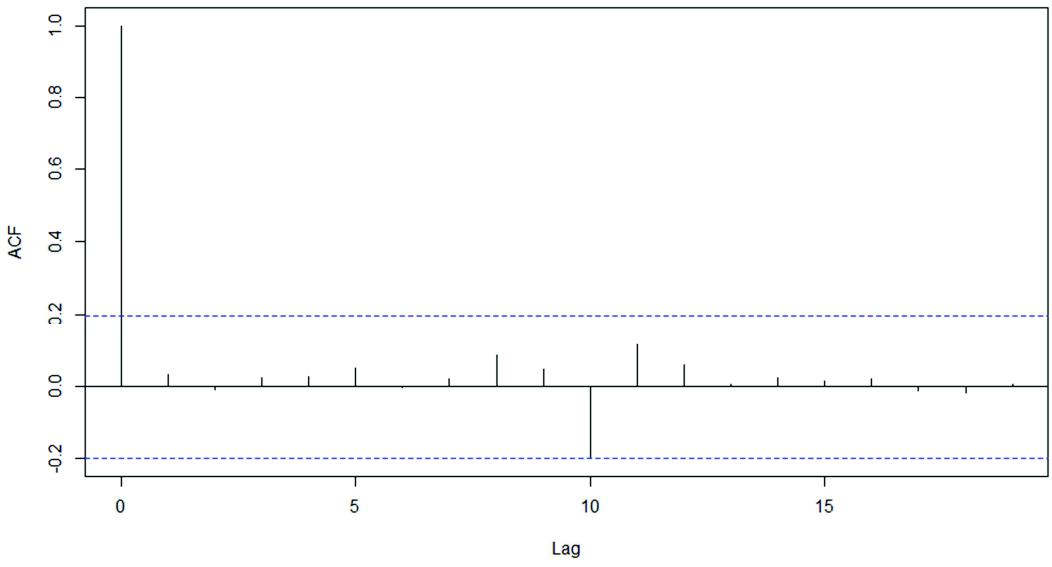

Figura 6.9–Autocorrelação entre os erros para diferentes lags

6.1.1.6. - Incertezas nas vazões modeladas

Além de obter as funções densidade de probabilidade de todos os parâmetros do modelo, a abordagem bayesiana permite obter os intervalos de credibilidade das vazões simuladas, o que é extremamente útil quando se deseja tomar decisões sob uma ótica de análise de risco. Uma vez que foi verificado equivalência entre o comportamento das distribuições posteriores geradas pelos diferentes algoritmos, optou-se por apresentar as incertezas na série de vazões modeladas apenas para os resultados decorrentes da cadeia 1 do algoritmo MH. AFigura 6.10mostra os resultados das incertezas geradas no comportamento da série de vazões simuladas. Pode-se notar que quatro observações estão fora do intervalo de credibilidade de 95%, o que é razoável já que a série histórica contém 97 meses.

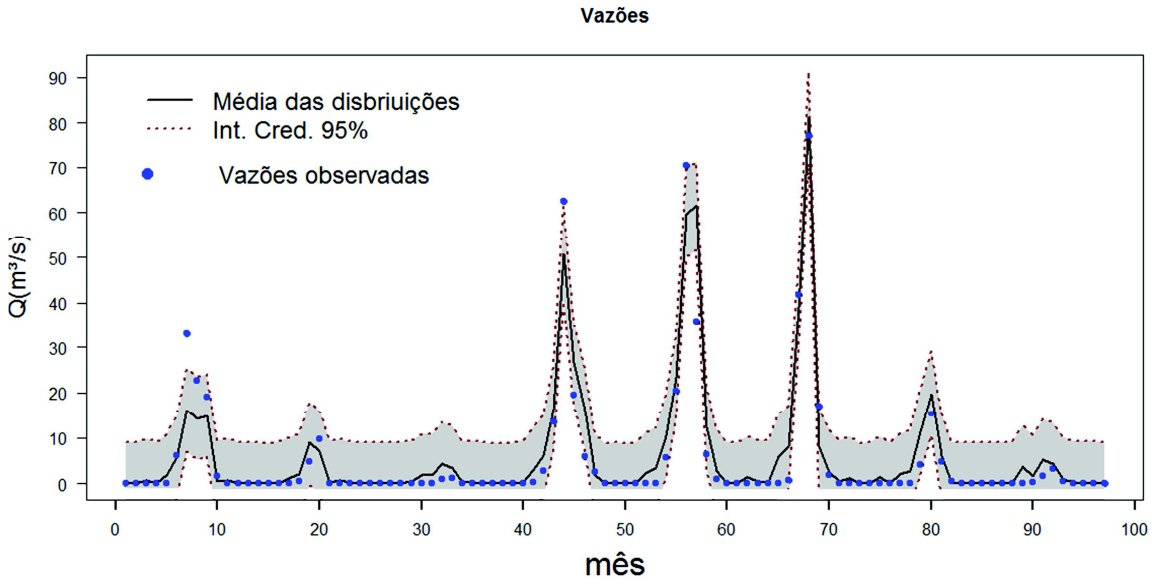

Figura 6.10 - Hidrograma de vazões médias mensais com intervalo de credibilidade de 95%

6.1.2 - Caso com dados sintéticos

O caso com dados sintéticos buscou, sobretudo, validar os códigos computacionais implementados, além de efetuar a análise comparativa entre eles. Ou seja, considerando que os dados de vazões foram gerados sinteticamente, tendo-se inserido ruídos nos valores de vazões para representar erros de natureza normal, homocedástica e indpendente, partiu-se da premissa que uma maneira de validar os algoritmos implementados seria se esses representassem adequadamente a natureza dos resíduos sinteticamente gerados e, portanto, conhecidos.

Assim, constituiu-se uma série de vazões sintéticas gerada a partir do modelo SMAP - Mensal com base em valores pré-fixados para os parâmetros do modelo (SAT=2.200; PES=2,5; CREC=0.3; K=3,5) e com valores iniciais de umidade e escoamento de base, a saber: 0,60 e 35 m^3/s, respectivamente, tendo-se, também, adotado dados de chuva e evapotranspiração potencial observados na bacia hidrográfica de referência. Ressalta-se que para este caso foram admitidos todos os parâmetros do modelo hidrológico calibráveis.

6.1.2.1. - Dados de entrada

Os valores pré-fixados para os parâmetros tomaram como referência valores próximos àqueles obtidos por calibração convencional para um ponto de interesse da bacia hidrográfica do rio Jamanxim (Área de Drenagem = 39.888 km^2), principal afluente da margem direita do rio Tapajós. Esta localidade foi escolhida por coincidir com o ponto previsto para o aproveitamento hidrelétrico Jamanxim, usina hidrelétrica (UHE) prevista no âmbito dos estudos de inventário hidrelétrico dos rios Tapajós e Jamanxim (Maio/2008). Dessa forma, por ser uma bacia detalhadamente estudada no escopo do inventário hidrelétrico, aproveitaram-se os levantamentos e as análises de consistência dos principais dados hidrológicos. Destaca-se que os valores artificiais gerados não buscaram avaliar ou fazer qualquer confrontação com os estudos de inventário. O único intuito foi aproveitardados existentes e consistidos de precipitação e evapotranspiração para a geração de um caso sintético de vazões.

Os dados de chuva adotados tiveram como base a média aritmética das séries de chuvas mensais definidas para os postos pluviométricos Alto Tapajós (Cód. ANA 00757000) e

Diamantino (Cód. ANA 01456005), integrantes da bacia do rio Tapajós. Para a série de evapotranspiração potencial adotou-se a média aritmética de três estações climatológicas, a saber: Belterra, Parintins e Itaituba. A Figura 6.11 apresenta a bacia de referência adotada e a localização dos postos e estações considerados para este caso.

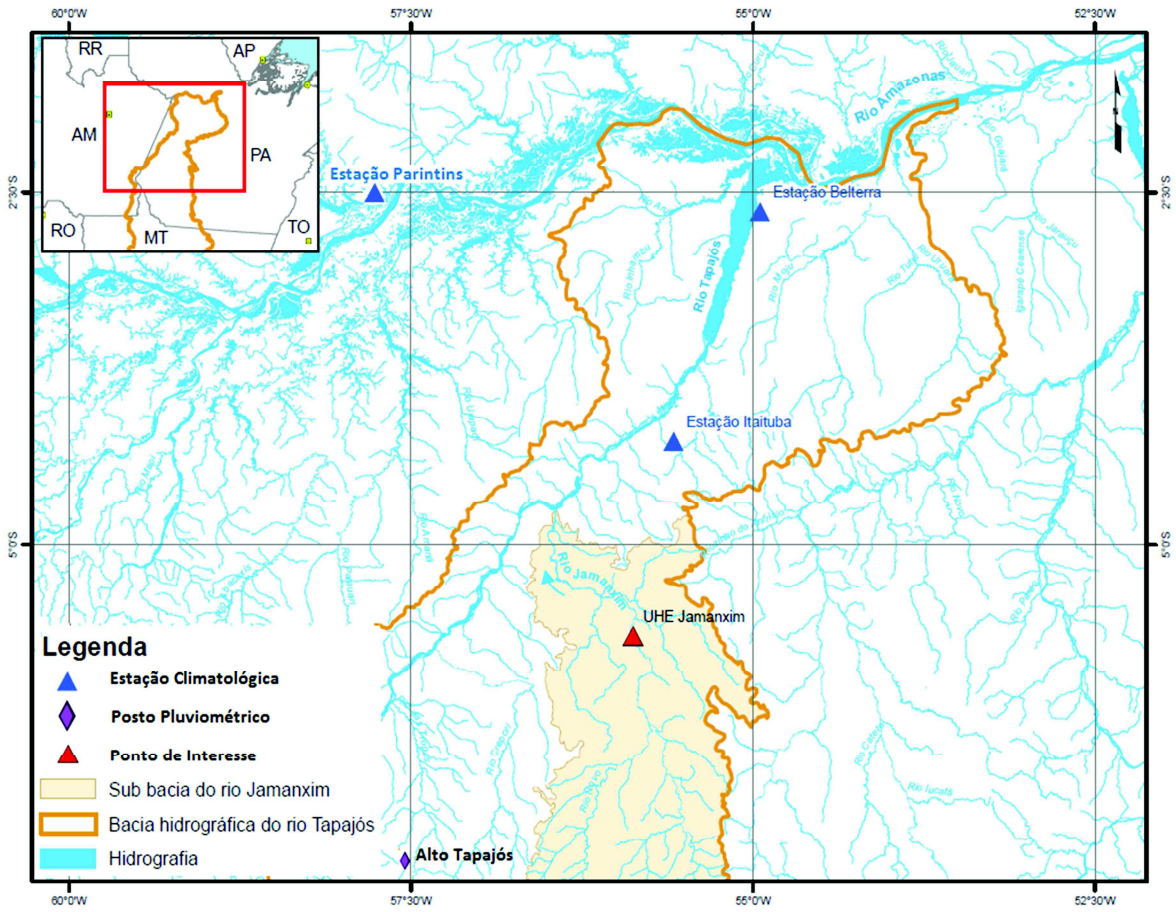

Figura 6.11 - Bacia de referência adotada para a simulação do caso de vazões sintéticas

À série sintética de vazões geradas foram inseridos erros, admitidos normalmente distribuídos, homocedásticos e independentes, definidos a partir de amostras de uma distribuição normal com média zero e desvio padrão 300, $\varepsilon \sim N(0{,}300)$. A taxa de umidade inicial foi fixada em 60% e o escoamento de base foi admitido 35 m³/s.

6.1.2.2. - Distribuições *a priori* e limites de busca dos parâmetros

Conforme efetuado para o primeiro caso simulado, optou-se por adotar distribuições *a priori* não informativas dadas por distribuições uniformes. Os limites de variação assumidos para essas distribuições em função dos respectivos parâmetros são apresentados

naTabela 6.14, tendo-se adotado um intervalo significativo entorno do valor real conhecido, considerado para geração das vazões sintéticas. Para a calibração convencional os limites de busca dos parâmetros do modelo hidrológico foram equivalentes, excetuando-se os do parâmetro referente ao desvio padrão do erro do modelo que não é considerado no procedimento convencional de calibração.

Tabela 6.5- Limites das distribuições uniformes adotadas *a priori* para os parâmetros de interesse e para a busca via algoritmo PSO no caso real com dados sintéticos

Parâmetro	**Distribuição uniforme**	
	Mínimo	**Máximo**
SAT	500	5000
PES	0,5	6
CREC	0	1
K	1	8
σ	10	800

6.1.2.3. - Distribuições propostas

As considerações referentes às distribuições propostas adotadas para este caso são similares àquelas realizadas para o caso anterior com dimensão reduzida de parâmetros. Dessa forma, para o algoritmo Mforam adotadas distribuições propostas simétricas, tendo sido utilizadas distribuições normais para todos os parâmetros (SAT, PES, CREC, K e σ).

Tal como efetuado no caso precedente para as distribuições normais, foram adotadas médias iguais ao valor do parâmetro no passo corrente da cadeia e desvio padrão de 100 (SAT), 0,1 (PES), 0,1 (CREC), 0,5 (K) e 15 (σ).

É oportuno destacar que, inicialmente, buscou-se efetuar as propostas para os parâmetros (CREC, K e σ) com base em distribuições uniformes cujos os intervalos de proposição eram atualizados a cada passo da iteração, estabelecendo os novos limitesdos intervalos como a soma e a subtração do valor proposto para cada parâmetro e sua metade, tal qual realizado na simulação do primeiro caso para o parâmetro (σ). No entanto, o emprego das distribuições uniformes como propostas para esta simulação não funcionaram bem, reduzindo sobremaneira a taxa de aceitação das proposições (abaixo de 1%), mesmo

quando se buscava reduzir os intervalos de proposição reduzindo-se a variância das propostas.

Com o propósito de superar essa dificuldade, foram adotadas distribuições normais como propostas para todos os demais parâmetros, tendo-se ajustado o desvio padrão de cada uma delas para valores que gerassem uma taxa de aceitação superior a 10% em simulações preliminares com apenas 1.000 iterações. Essa constatação corrobora com os registros da literatura, revelando a necessidade de se ajustar as distribuições de proposição e seus parâmetros para cada caso simulado.

A Tabela 6.6 apresenta um resumo das distribuições de proposição adotadas para as simulações com o algoritmo *Metropolis*.

Tabela 6.6- Distribuições propostas consideradas para o Algoritmo *Metropolis* no caso com dados sintéticos

Parâmetro	**Distribuição**	**Parâmetros**
SAT	Normal	Média=$SAT_{(i)}$; Desvio Padrão=100
PES	Normal	Média=$PES_{(i)}$; Desvio Padrão=0,1
CREC	Normal	Média=$CREC_{(i)}$; Desvio Padrão=0,1
K	Normal	Média=$K_{(i)}$; Desvio Padrão=0,5
σ	Normal	Média= $\sigma_{(i)}$; Desvio Padrão=15

(i) - Representa o passo corrente da cadeia

Para o algoritmo MH, as propostas foram ajustadas conforme o caso apresentado anteriormente com o intuito de garantir o emprego de distribuições não simétricas. Para o parâmetro (SAT) adotou-se uma distribuição Log-Normal, com média dada pelo logaritmo Neperiano do valor do parâmetro no passo corrente da cadeia e desvio padrão de 0,1.

As propostas para os demais parâmetros do modelo hidrológico (PES, CREC, K) foram realizadas por meio de uma distribuição Gama, com o parâmetro (α) da distribuição dado pelo produto entre o valor do parâmetro do modelo no passo corrente da cadeia e o parâmetro (β) da distribuição de proposição, enquanto o parâmetro (β) foi dado pela razão entre a constante 2.500 e o valor do parâmetro do modelo no passo corrente da cadeia.

A distribuição de proposição do desvio padrão do erro do modelo (σ) foi realizada de forma indireta, a partir do parâmetro de precisão (ϕ), dado pelo inverso da variância do erro do modelo ($1/\sigma^2$), também por meio de uma distribuição Gama, conforme descrito no caso anterior.

ATabela 6.7apresenta um resumo das distribuições de proposição consideradas para as simulações com o Algoritmo MH.

Tabela 6.7- Distribuições propostas consideradas para o Algoritmo *Metropolis Hastings* no caso com dados sintéticos

Parâmetro	**Distribuição**	**Parâmetros**
SAT	Log-Normal	Média=LN[$SAT_{(i)}$] ; Desvio Padrão=0,1
PES	Gama	α= $PES_{(i)}$ X β; β=2.500/ $PES_{(i)}$
CREC	Gama	α= $CREC_{(i)}$ X β; β=2.500/ $CREC_{(i)}$
K	Gama	α= $K_{(i)}$ X β; β=2.500/ $K_{(i)}$
ϕ	Gama	α= $\phi_{(i)}$X β; β=2.025/ $\phi_{(i)}$

(i) - Representa o passo corrente da cadeia; $\sigma=1/\phi^{0,5}$

Por fim, para o algoritmo AM, adotou-se para os 600 passos iniciais da cadeia uma matriz de covariância constante dada pela diagonal principal formada por valores fixos das variâncias dos parâmetros (SAT, PES, CREC, K e σ) e os demais elementos da matriz nulos. Dessa forma, para a diagonal principal da referida matriz adotaram-se os seguintes valores: 900; 0,0064; 0,0064; 0,01 e 225. Para os passos das cadeias posteriores ao de número 600, a matriz de covariância foi atualizada conforme Equação (4.3)

6.1.2.4. - Avaliação do desempenho dos algoritmos e incertezas nos parâmetros do modelo

Como no caso em questão partiu-se de uma série sintética de dados, cujos erros foram conhecidos, tendo sido gerados a partir de uma distribuição normal com média nula e desvio padrão 300, dispensou-se a simulação de mais de uma cadeia para cada umdos algoritmos estudados, uma vez que,sea primeira cadeia de cada algoritmo reproduzisse adequadamente a natureza dos erros nas vazões simuladas, admitir-se-ia o alcance da convergência.Portanto, feitas as devidas ressalvas, não se aplicou o diagnóstico de Gelaman & Rubin (1992).

Para as cadeias geradas com todos os algoritmos simulados foram processadas 80.000 iterações. O valores iniciais dos parâmetros do modelo para todas as cadeias foram:500 (SAT), 1,1 (PES),0,2 (CREC), 2 (K) e 100 (σ).

As Figuras 6-12 e 6-13 apresentam o comportamento das cadeias de Markov (gráficos superiores) e das distribuições posteriores (gráficos inferiores) obtidas a partir dos algoritmos simulados para todos os parâmetros de interesse.Assim como constatado no primeiro caso simulado, observa-se equivalência das distribuições posteriores obtidas com os diferentes algoritmos. Nas figuras citadas a linha tracejada azul refere-se ao valor do parâmetro decorrente do procedimento convencional de calibração. Verifica-se que este valor encontra-se próximo à média da distribuição posterior encontrada.

A Tabela 6.8 resume a avaliação comparativa entre as performances dos algoritmos estudados. Aprimeira coluna indica o algoritmo considerado, a segunda coluna apresenta a média das cadeias descartados os 60.000 primeiros valores para cada um dos 5 parâmetros (SAT, PES, CREC, K e σ). A terceira coluna apresenta os intervalos de credibilidade de 95%. As duas colunas finais mostram respectivamente, a taxa de aceitação e o tempo de processamento para as 80.000 iterações. Visando facilitar a comparação, a Tabela 6.9 resume os valores reais considerados para a geração das séries de vazões sintéticas.

Embora de uma forma geral as distribuições posteriores tenham se mostrado muito similares, verificando com mais cuidado o comportamento das cadeias geradas pelos algoritmos, constata-se que o algoritmo M promoveu uma mistura mais limitada no universo de valores percorridos pelas cadeias, quando comparado aos algoritmos MH e AM. Tal limitação pode ser melhorada testando-se diferentes valores de variância para as distribuições propostas ou mesmo adotando-se outra distribuição de proposição, contudo, tal comportamento não afetou a convergência da cadeia. Com base em diversas simulações efetuadas percebeu-se que o algoritmo M exige mais cuidado na especificação das distribuições propostas e na especificação de seus parâmetros para propiciar uma maior varredura do espaço amostral de representação dos parâmetros e atingir a convergência.

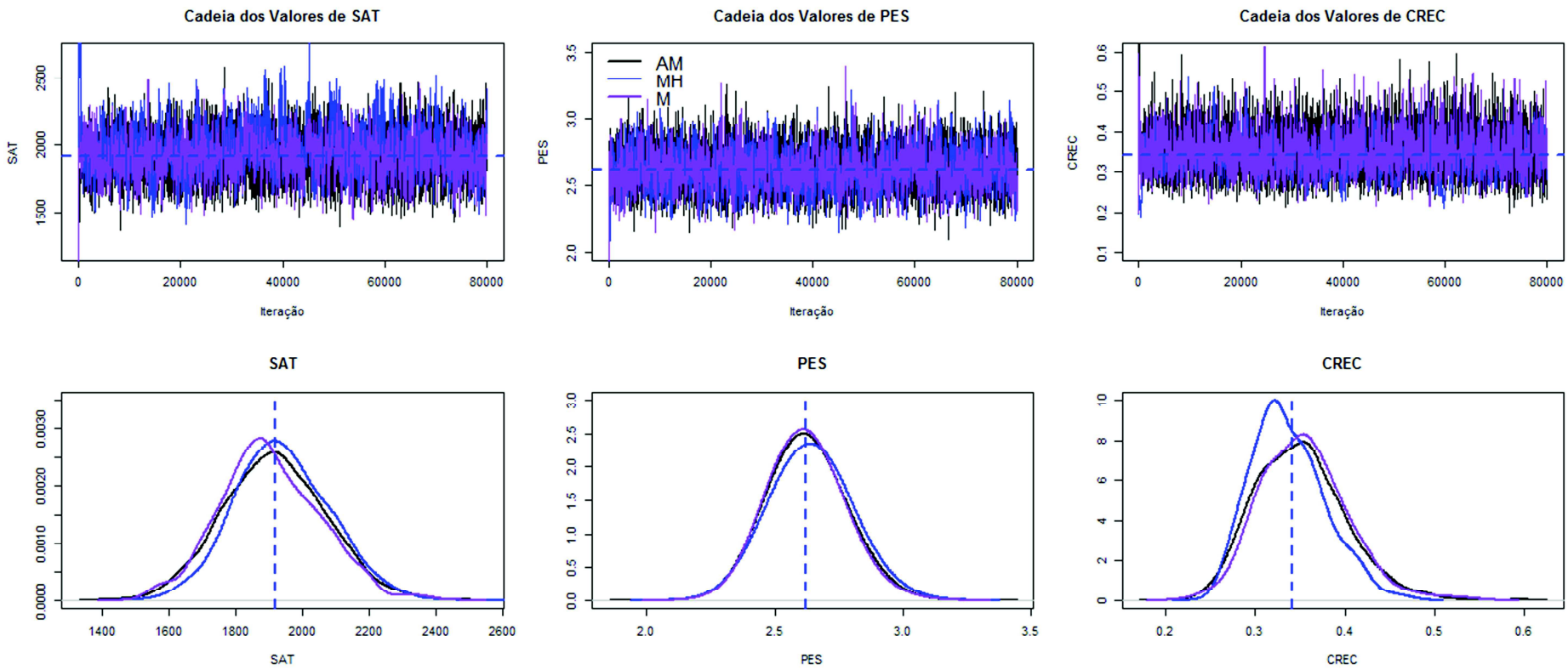

Figura 6.12–Distribuições posteriores e cadeias de Markov para os parâmetros SAT, PES e CREC

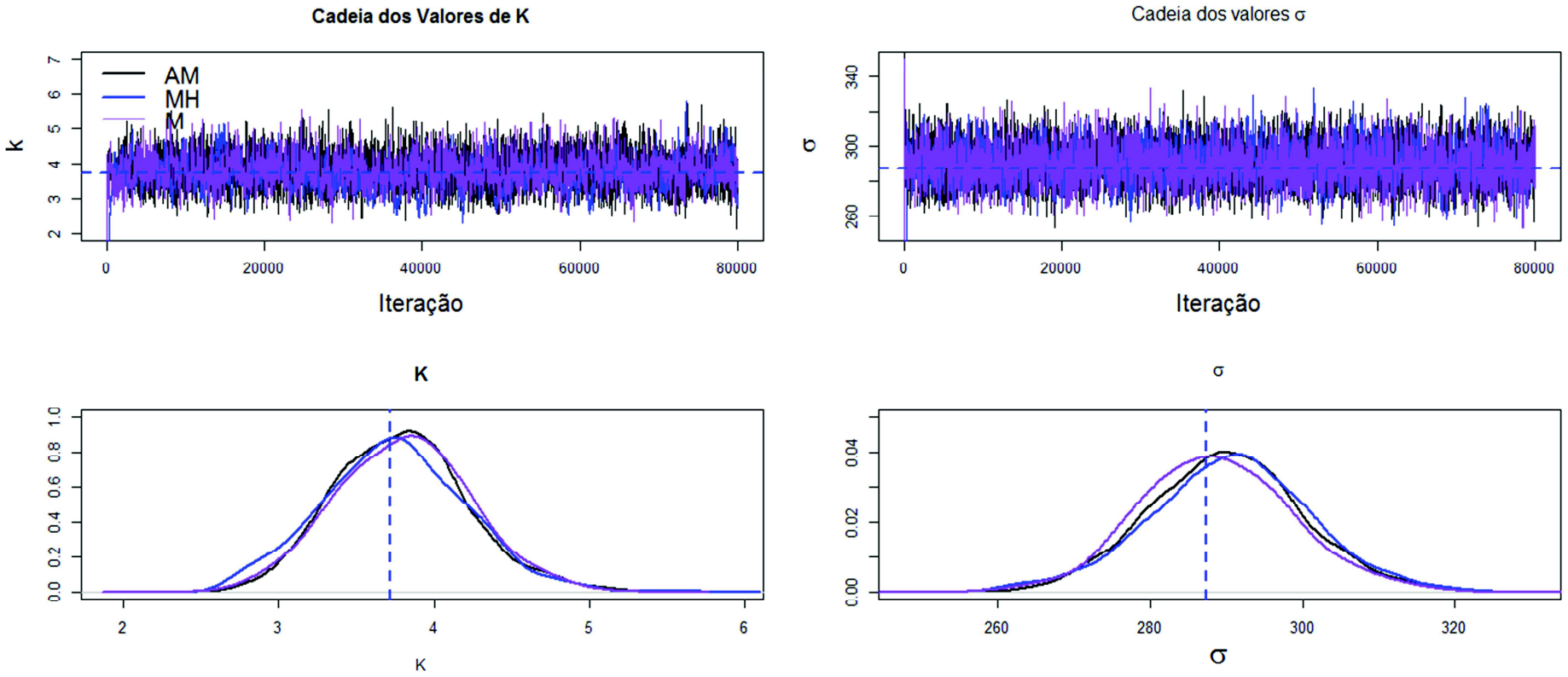

Figura 6.13–Distribuições posteriores e cadeias de Markov para os parâmetros K e σ

Tabela 6.8- Resumo da análise de convergência e comparação entre os algoritmos M, MH e AM para o caso com dados sintéticos

Algoritmo	Média dos Parâmetros					Intervalo de Credibilidade de 95%					Taxa de Aceitação	Tempo de Processamento (hs) *Processador Intel-Core i5
	SAT	PES	CREC	K	σ	SAT	PES	CREC	K	σ		
M	1904,0	2,62	0,35	3,80	289	[1.647,0 ; 2.221,5]	[2,36 ; 2,88]	[0,26 ; 0,45]	[3,00 ; 4,65]	[271 ; 310]	8,0%	1,83
MH	1.941,9	2,64	0,34	3,74	290	[1.682,3 ; 2.243,2]	[2,36 ; 2,93]	[0,27; 0,42]	[2,75; 4,54]	[270 ; 313]	30,7%	1,5
AM	1918,3	2,62	0,35	3,79	290	[1.634,2 ; 2.235,1]	[2,37; 2,91]	[0,26 ; 0,45]	[2,95 ; 4,67]	[269 ; 309]	16,8%	2,13
PSO	1.910,0	2,62	0,34	3,72	287	-					-	0,031

Tabela 6.9- Valores reais dos parâmetros

SAT	PES	CREC	K	σ
2.200	2,5	0,3	3,5	300

Tal como observado no primeiro caso, verifica-se que os valores obtidos para os parâmetros por meio da calibração convencional com o algoritmo PSO encontram-se dentro do intervalo de credibilidade das distribuições posteriores encontradas, assim como os valores reais dos parâmetros.

As correlações entre os parâmetros do modelo foram avaliadas para as cadeias do algoritmo AM, com base nos gráficos de dispersão apresentados na Figura 6.14 e Figura 6.15, sendo mais evidente as correlações entre os parâmetros (SAT x CREC) e (SAT x K).

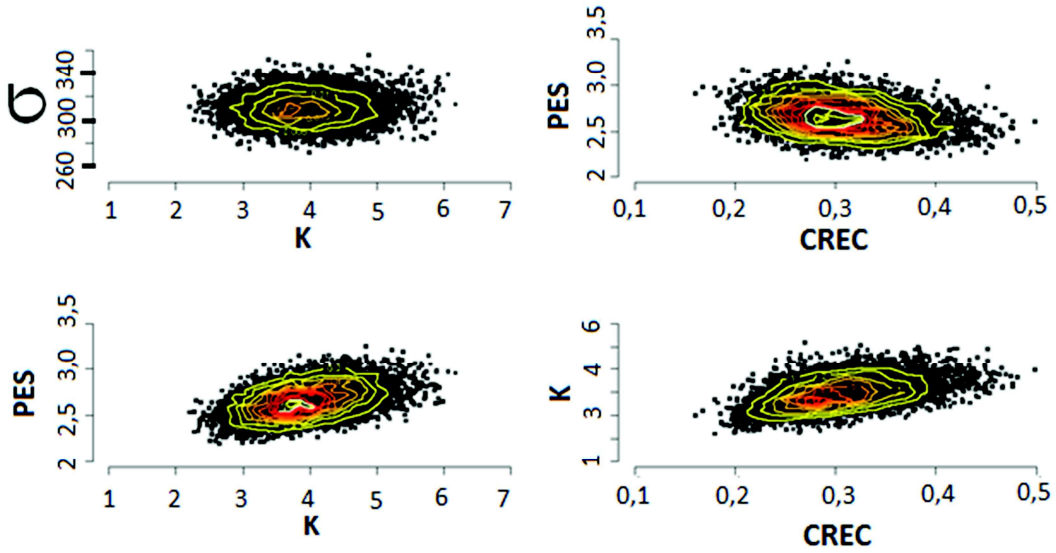

Figura 6.14–Correlações entre os parâmetros do modelo (a)

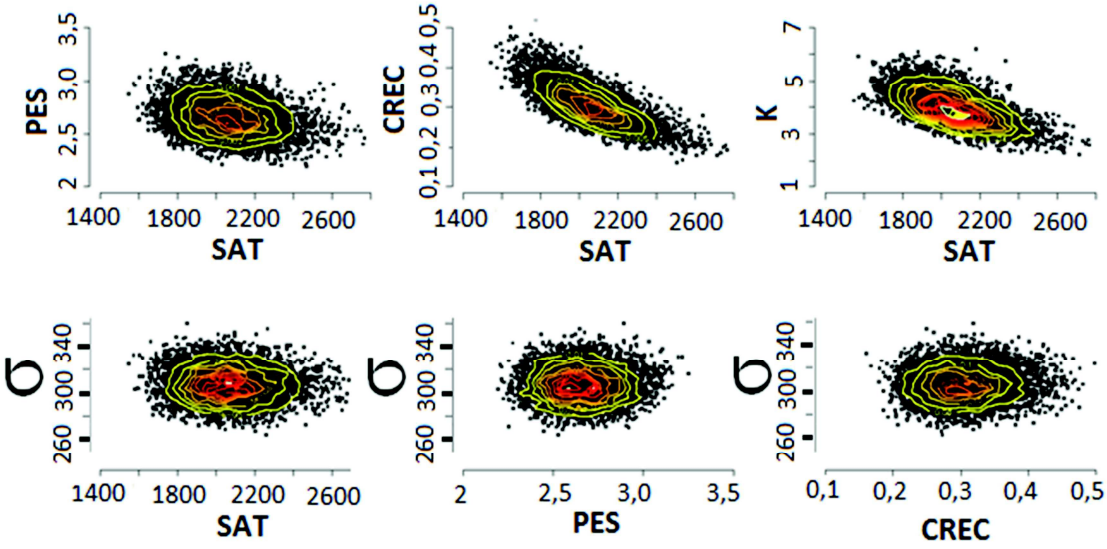

Figura 6.15–Correlações entre os parâmetros do modelo (b)

6.1.2.5. - Avaliação dos resíduos

Considerando a equivalência das distribuições posteriores obtidas, optou-se por realizar a análise dos resíduos apenas para a simulação efetuada com o algoritmo AM. A Figura 6.16 apresenta um gráfico de dispersão dos resíduos em função das vazões simuladas. Tendo em vista que os resíduos foram gerados sinteticamente a partir de uma distribuição normal não é possível observar no gráfico de dispersão qualquer viés relacionado aos resíduos ou indícios de que os mesmos variem em função do valor de vazão.

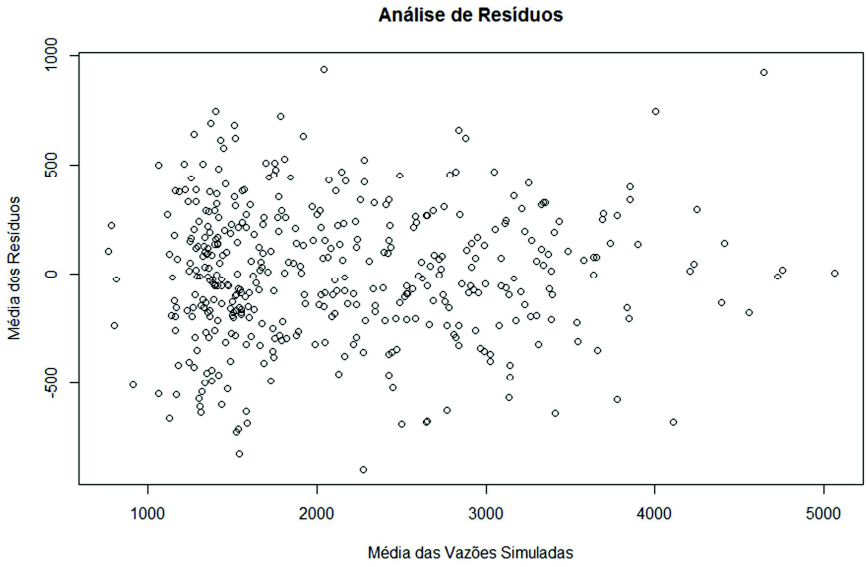

Figura 6.16–Comportamento da média dos resíduos em função das vazões médias simuladas

A Figura 6.17 apresenta o comportamento dos resíduos gerados sinteticamente para a série de vazões artificiais e aqueles obtidos a partir das vazões simuladas com os parâmetros da cadeia convergente. A proximidade entre as duas distribuições mostra que a simulação representou adequadamente o comportamento dos erros sinteticamente gerados a partir de uma distribuição normal com média 0 e desvio padrão 300, confirmando a hipótese de que os resíduos neste caso podem ser representados por uma distribuição normal.

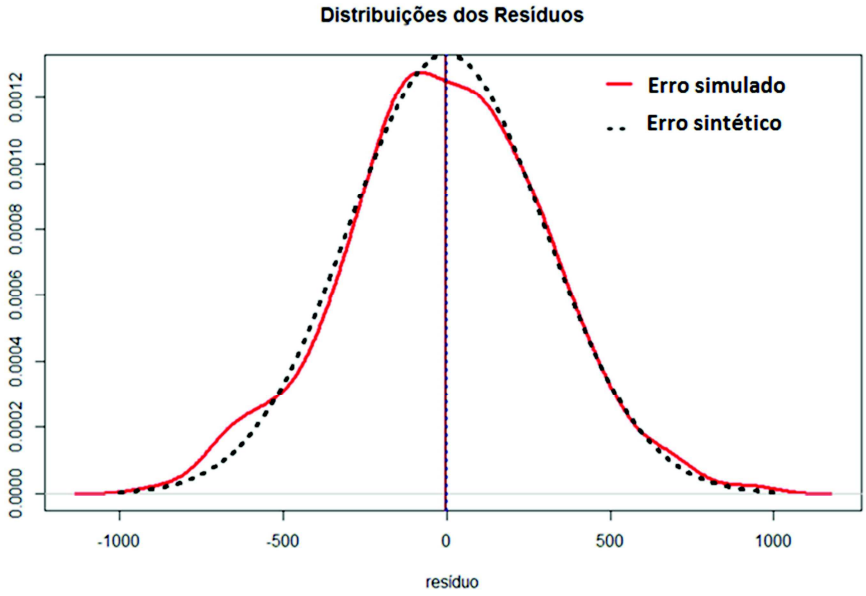

Figura 6.17–Distribuições do erro do modelo simulado e sinteticamente gerado

Ainda com relação aos resíduos, a Figura 6.18 apresenta no gráfico superior esquerdo os resíduos em função dos valores de vazões simuladas, no gráfico superior direito o gráfico quantil-quantil, no gráfico inferior esquerdo os resíduos padronizados, e no gráfico inferior direito os valores das distâncias de Cook em função do número de observação da série de vazões. Nos gráficos de dispersão dos resíduos (dois gráficos da coluna esquerda) não é possível identificar nenhum comportamento que sugira heterocedasticidade ou ainda dependência dos erros, tendo em vista que o padrão de dispersão dos pontos se mantém relativamente equivalente independentemente do valor da vazão modelada. No gráfico quantil-quantil da distribuição normal, observa-se que os pontos dos resíduos padronizados se ajustam perfeitamente a reta que representa a normalidade. No gráfico que representa a distância de Cook's não se observou a presença de número significativo de *outliers*, tendo sido registrados apenas 3 dados com valores significativos. Dessa forma, os resultados da análise de resíduos comprovam a independência e a homocedasticidade dos erros, indicando que o modelo capturou adequadamente as propriedades dos erros sintéticos estabelecidos na simulação.

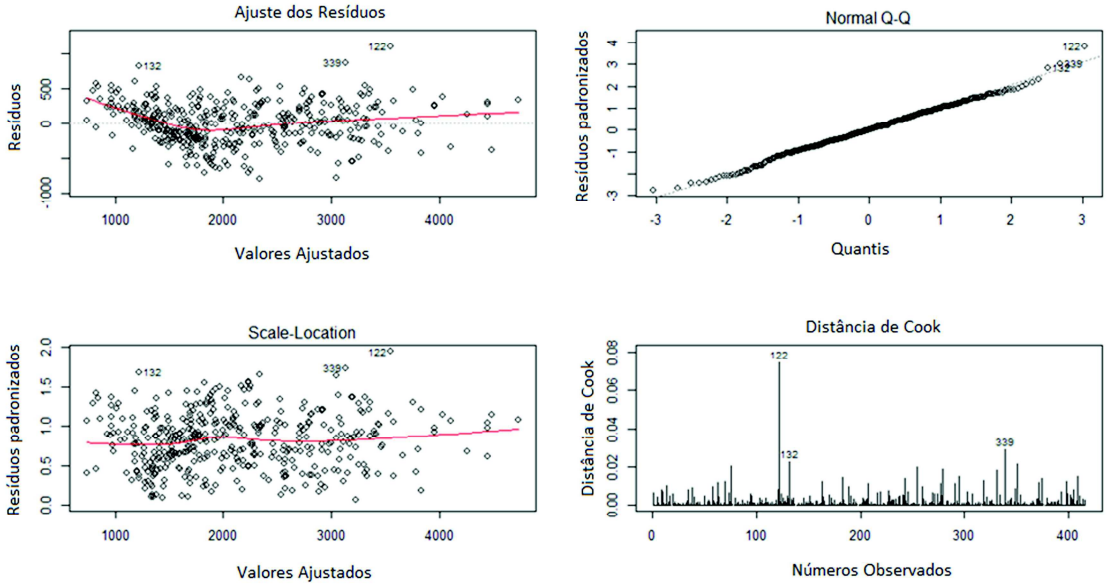

Figura 6.18–Análise de Resíduos

6.1.2.6. - Incertezas nas vazões modeladas

A Figura 6.19 apresenta o resultado final para a série de vazões simuladas indicando os intervalos de credibilidade de 95%, as vazões geradas sinteticamente (pontos azuis) e a média das diversas vazões simuladas a partir dos parâmetros do modelo SMAP decorrentes das cadeias de Markov geradas (linha contínua preta). Como houve correspondência dos resultados para os diferentes algoritmos simulados, por simplificação, optou-se por apresentar as vazões decorrentes da simulação apenas com algoritmo AM. Nota-se no hidrograma mostrado que cerca de 19 observações se mantiveram fora do intervalo de credibilidade de 95%, o que indica adequabilidade do intervalo de credibilidade obtido, uma vez que a série histórica apresentada possui 416 meses.

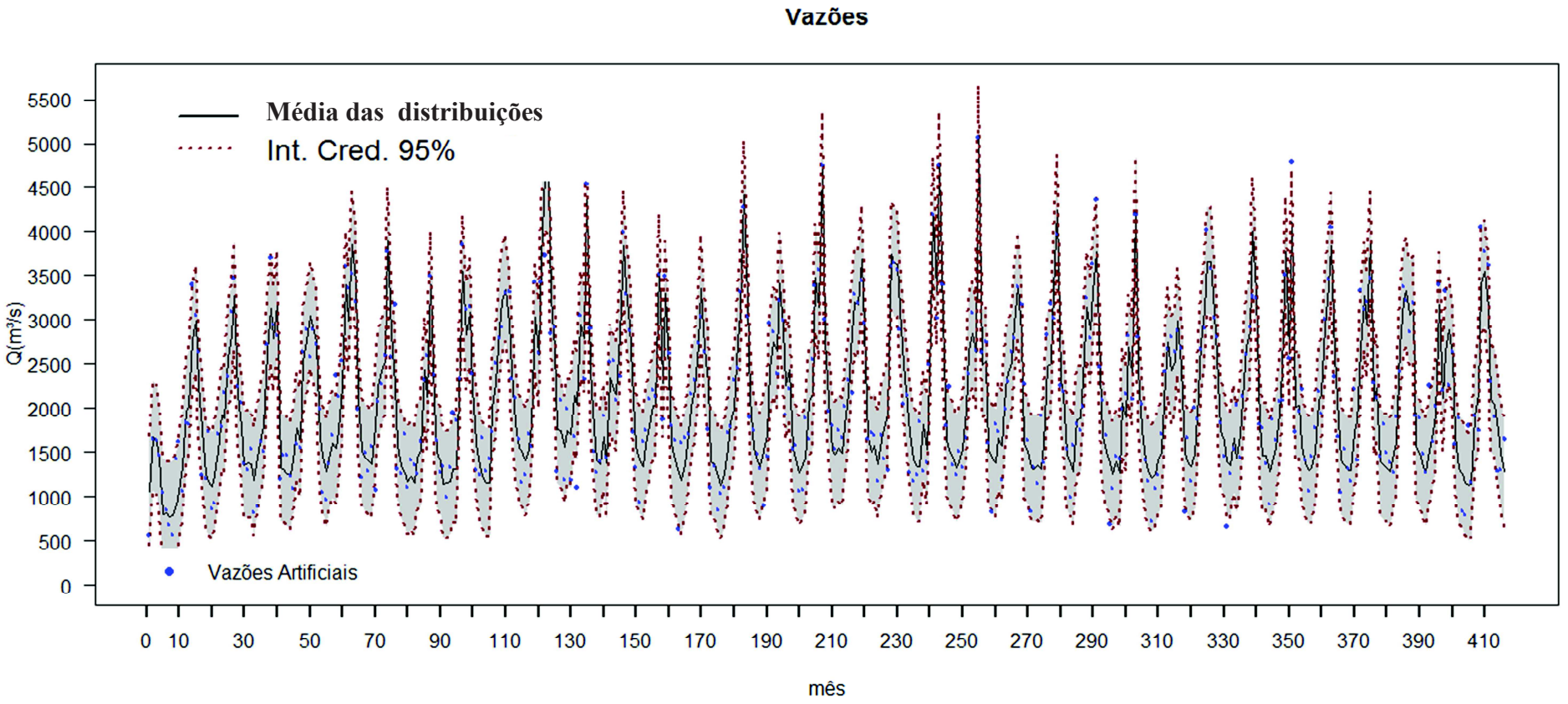

Figura 6.19–Vazões geradas sinteticamente (pontos azuis), com a indicação do intervalo de credibilidade de 95% (hachuras cinzas delimitadas pelas linhas tracejadas vermelhas) e vazões médias decorrentes das simulações realizadas com os parâmetros do modelo hidrológico obtidos das cadeias processadas (linha contínua preta) para o algoritmo AM

6.1.3 - Caso com dados reais

O caso com dados reais empregou os algoritmos AM e MH para avaliação bayesiana das incertezas dos parâmetros bem como o algoritmo PSO seguido da aplicação do método Simplex (Nelder & Mead, 1965) para calibração convencional. Nesta simulação assumiu-se que os resíduos pudessem ser caracterizados por uma distribuição normal com média zero e desvio padrão σ, tendo-se avaliado esta hipótese na análise dos resíduos.

6.1.3.1. - Dados de entrada

Para a localidade de estudo foi escolhida a bacia hidrográfica referente à estação fluviométrica Ponte Raul Soares (Cód. ANA 41340000), localizada no rio das Velhas, afluente do rio São Francisco.

Os dados de vazões observadas, evapotranspiração potencial e precipitação foram aqueles considerados por Otsuki (2015). Tratam-se de dados com intervalo de tempo mensal obtidos para o período de janeiro de 1951 a dezembro de 2007. A Figura 6.20 apresenta a localização geral da bacia considerada para este caso.

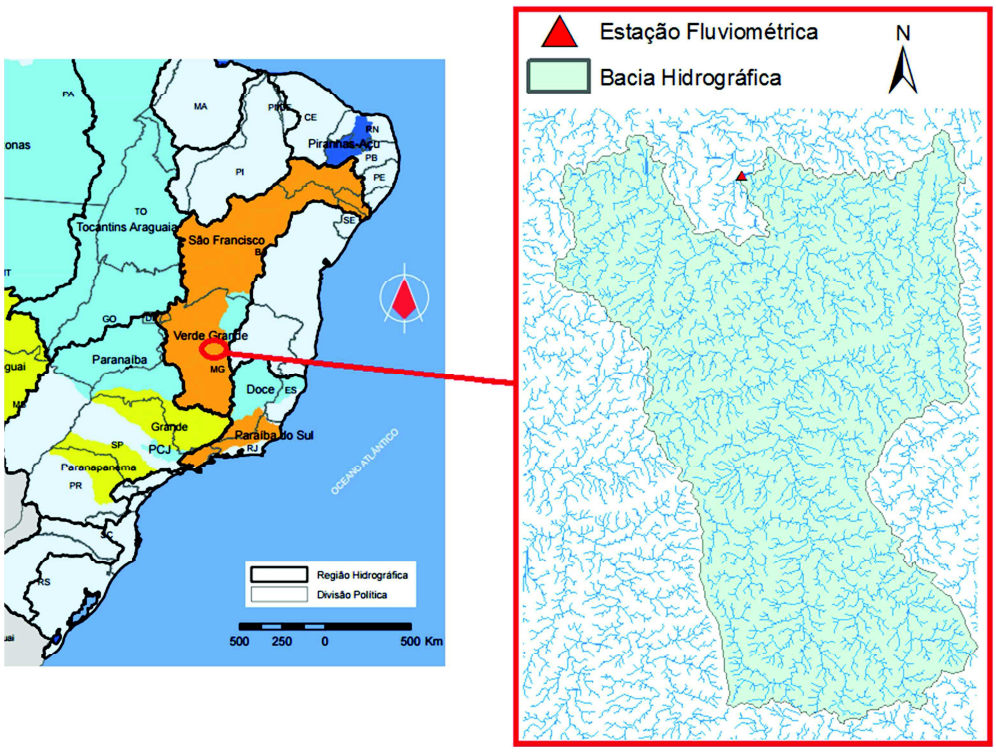

Figura 6.20– Bacia de referência adotada para a simulação do caso com dados reais

A bacia hidrográfica apresenta uma área de 4860 km², os valores da taxa de umidade inicial (TUin) e do escoamento de base inicial (EBin) foram estabelecidos em 0,40 e 20 m³/s, respectivamente. Esses valores foram adotados com base em simulações convencionais de calibração que consideraram esses parâmetros calibráveis.

6.1.3.2. - Distribuições *a priori,* de proposição e limites de busca dos parâmetros

Para ambos os algoritmos (AM e MH) foram adotadas distribuições *a priori* uniformes. Os limites de variação assumidos para essas distribuições em função dos respectivos parâmetros são apresentados naTabela 6.10. Para a calibração convencional os limites de busca dos parâmetros do modelo hidrológico foram equivalentes, excetuando-se do parâmetro referente ao desvio padrão do erro do modelo que não é considerado no procedimento convencional de calibração. Os valores dos limites indicados foram estabelecidos após diversos procedimentos de calibração convencional (empregando o algoritmo PSO, seguido da aplicação do método Simples desenvolvido por Nelder & Mead, 1965) e inúmeras simulações que avaliaram a sensibilidade do modelo hidrológico.

Tabela 6.10- Limites das distribuições uniformes adotadas *a priori* para os parâmetros de interesse e para a busca via algoritmo PSO

Parâmetro	**Distribuição uniforme**	
	Mínimo	**Máximo**
SAT	300	5000
PES	0	10
CREC	0	40
K	0	20
σ	1	50

6.1.3.3. - Distribuições propostas

A distribuição de proposição do algoritmo AM consistiu na distribuição normal multivariada com o valor inicial da matriz de covariância semelhante ao caso com dados sintéticos, ou seja, dada pela diagonal principal formada pelos valores de variância: 900; 0,0064; 0,0064; 0,01; 225, para os parâmetros SAT, PES, CREC, K e σ, respectivamente, sendo o último parâmetro representativo da variância do erro do modelo. Para as

simulações até o passo 600, os demais elementos fora da diagonal principal da matriz de covariância foram assumidos nulo.

Para o algoritmo MH adotaram-se para as proposições as distribuições Log-Normal e Gama. Para as distribuições Log-Normal a média da distribuição de proposiçãoconsiderou o valor do logaritmo natural do parâmetro no passo corrente da cadeia (i) e um valor fixo de desvio padrão. Para as distribuições gama os parâmetros (α) e (β) foram definidos com base no valor de uma constante e no valor do parâmetro no passo corrente da cadeia.

A Tabela 6.11 resume as distribuições de proposição consideradas. Os tipos e os parâmetros das distribuições de proposição foram estabelecidos após diversas simulações experimentais que avaliaram o comportamento das cadeias geradas, tendo-se adotado as distribuições testadas que apresentaram melhor desempenho na formação das cadeias geradas. Para o parâmetro K, inicialmente, adotou-se a distribuição Gama e posteriormente a distribuição Log-Normal.

Tabela 6.11- Distribuições propostas consideradas

Parâmetro	**Distribuição**	**Parâmetros**
SAT	Log-Normal	Média=LN[$SAT_{(i)}$] ; Desvio Padrão=0,1
PES	Gama	$\alpha = PES_{(i)} X \beta$; $\beta = 2.500 / PES_{(i)}$
CREC	Gama	$\alpha = CREC_{(i)} X \beta$; $\beta = 2.500 / CREC_{(i)}$
K	Log-Normal	Média=LN[$K_{(i)}$] ; Desvio Padrão=0,2
	Gama	$\alpha = K_{(i)} X \beta$; $\beta = 2.500 / K_{(i)}$
σ	Gama	$\alpha = \sigma_{(i)} X \beta$; $\beta = 2.500 / \sigma_{(i)}$

(i) - Representa o passo corrente da cadeia

6.1.3.4. - Avaliação do desempenho dos algoritmos e incertezas nos parâmetros

Os algoritmos AM e MH foram processados para 150.000 iterações gerando-se, inicialmente, duas cadeias para o algoritmo AM e duas cadeias a partir do algoritmo MH, considerando cada um dos parâmetros estudados. Para uma das cadeias do algoritmo MH processaram-se 100.000 iterações. Os valores iniciais de partida foram variados para cada uma das cadeias. A Figura 6.21 apresenta as cadeias com base nos algoritmos avaliados.

As cores preta e vermelha indicam as duas cadeias geradas para o algoritmo AM, as cores azul e verde mostram os valores das duas cadeias geradas a partir do algoritmo MH.

Apenas observando os traços das cadeias é possível constatar que, exceto para o parâmetro K, as cadeias geradas para ambos os algoritmos parecem ter promovido uma mistura razoável e equivalente do espaço amostral dos parâmetros. Para o parâmetro K, no entanto, verifica-se nos traços das cadeias que o algoritmo AM (cadeias em vermelho e preto) promoveu uma mistura mais adequada dos valores ao longo do espaço amostral dos parâmetros, enquanto as cadeias obtidas a partir do algoritmo MH (cadeias em azul e verde) sugerem uma maior limitação do espaço amostral percorrido (provavelmente devido a uma variância pequena).

Para uma avaliação objetiva da convergência das cadeias considerou-se a métrica da estatística R de Gelman e Rubin definida como Fator Potencial de Redução de Escala, conforme apresentado na seção 4.5-Avaliação da Convergência. A Tabela 6.12 resume os valores obtidos para este teste de convergência com base nas cadeias simuladas. Considerou-se que houve convergência quando o valor do teste indicou para a métrica R um valor inferior a 1,01, conforme proposto por Gelman *et al.* (2003). Na referida tabela são ainda indicados os tempos de processamento, as respectivas taxas de aceitação obtidas, e o número estimado de iterações a partir do qualse considerou que houve convergência.

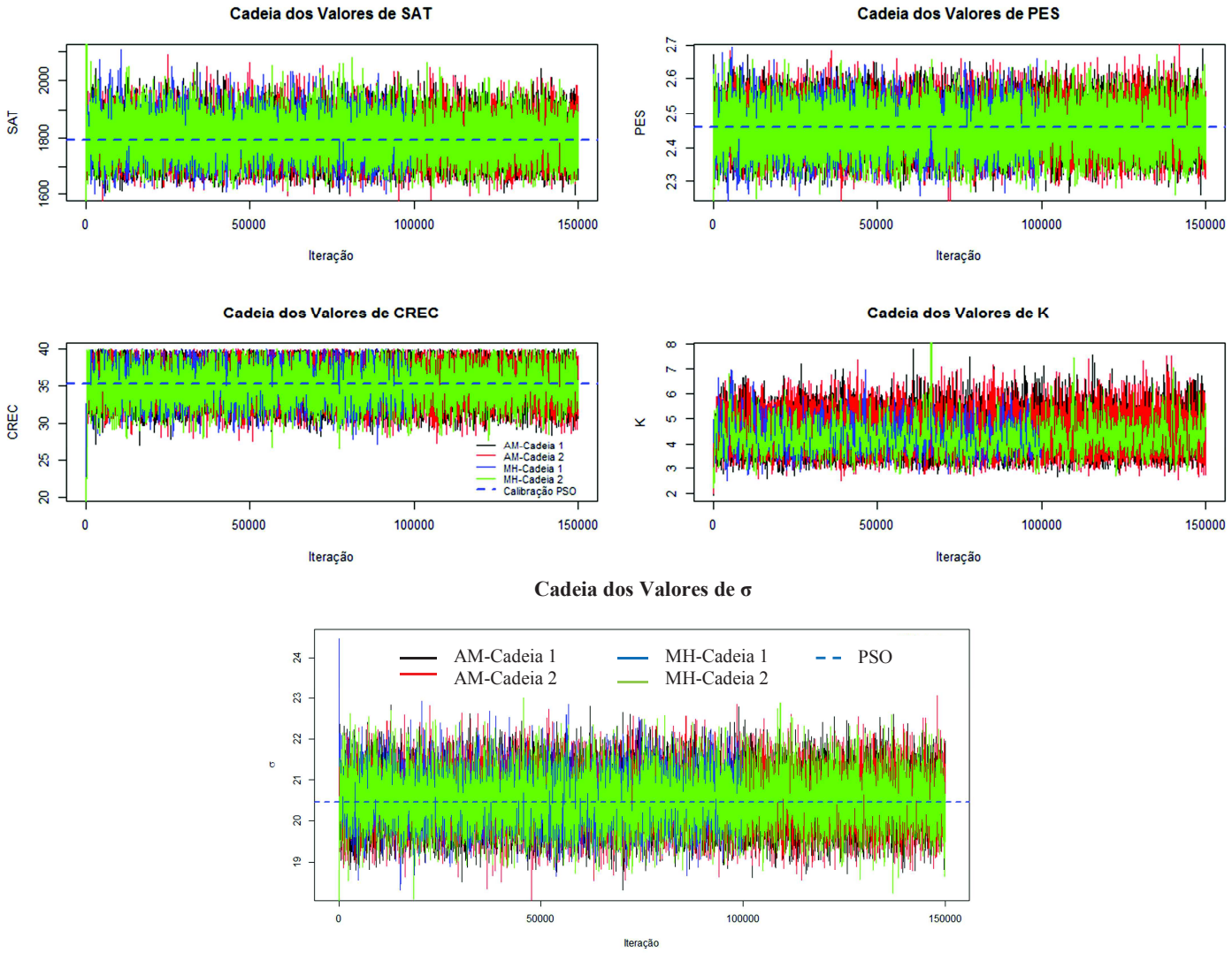

Figura 6.21– Cadeias de Markov dos parâmetros simulados com os algoritmos AM e MH

Tabela 6.12- Resumo da análise de convergência e comparação entre os algoritmos AM e MH

Algoritmo	Parâmetro	Fator R	Nº Iterações para convergência	Taxa de aceitação Cadeia 1	Taxa de aceitação Cadeia 2	Tempo de Processamento Cadeia 1 (h)	Tempo de Processamento Cadeia 2 (h)
AM	SAT	1,000195	18.000	20%	24%	3,1 *Processador Intel-Core i7	3,2 *Processador Intel-Core i7
	PES	1,00074	20.000				
	CREC	1,000346	10.000				
	K	1,000625	30.000				
	SIGMA	1,000302	30.000				
MH	SAT	1,000387	40.000	28%	43%	3,3 *Processador Intel-Core i5	4 *Processador Intel-Core i5
	PES	1,003597	40.000				
	CREC	1,001886	40.000				
	K	**1,025849**	-				
	SIGMA	1,000214	5.000				

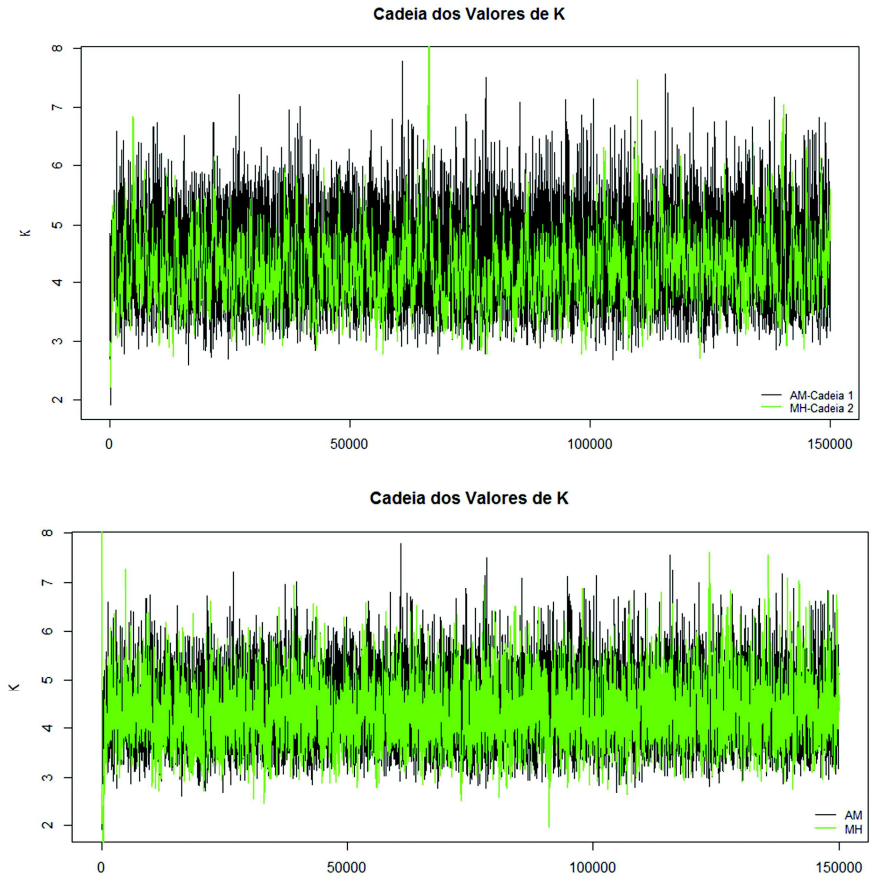

Figura 6.22– Cadeias geradas pelo algoritmo AM (linhas pretas) e pelo algoritmo MH (linhas verdes) com a proposição via distribuição Gama (gráfico acima) e distribuição Log-Normal (gráfico abaixo)

Posteriormente, outras duas simulações foram realizadas com o algoritmo MH substituindo-se a distribuição de proposição de Gama pela Log-Normal, com base nos parâmetros indicados naTabela 6.11. Essa substituição foi realizada com o propósito de buscar melhor mistura entre os valores da cadeia e alcançar a convergência. Após a alteração da distribuição proposta repetiu-se o teste de diagnóstico de Gelman e Rubin, obtendo-se o valor de 1,00017 para o fator R. A Figura 6.22apresenta uma comparação das cadeias obtidas para os algoritmos AM e MH para as situações com e sem convergência.

A Figura 6.23 e a Figura 6.24 apresentam as distribuições posteriores dos parâmetros com base nas cadeias convergentes simuladas, considerando a hipótese de resíduos normais homocedásticos e independentes (N). Para a composição das distribuições posteriores foram descartadas as 60.000 primeiras iterações. A Figura 6.13resume as estatísticas dos

parâmetros modelados, incluindo os resultados obtidos ao se empregar a calibração convencional com o algoritmo PSO seguido do método Simplex (Nelder & Mead, 1965) com base na métrica Nash-Suticliffe (NS).

As médias das distribuições posteriores se mantiveram próximas dos valores obtidos por meio da calibração convencional. Cada uma das cadeias geradas é representada por um tipo e cor de linha. O resultado obtido por meio da calibração convencional é representado pela linha vertical tracejada azul. Houve significativa proximidade entre as distribuições posteriores dos parâmetros para as cadeias de ambos os algoritmos avaliados.

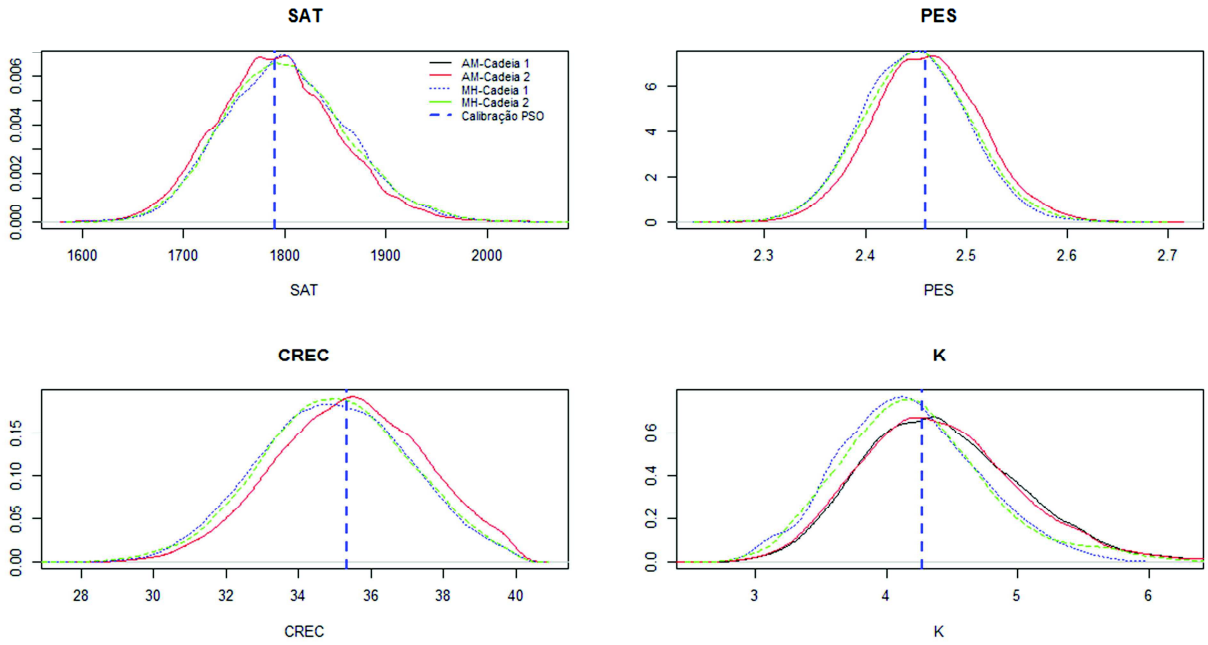

Figura 6.23– Distribuições posteriores dos parâmetros do modelo hidrológico para as cadeias dos algoritmos AM e MH

Tabela 6.13- Estatísticas de referência dos parâmetros avaliados com base nos algoritmos implementados

Parâmetro	Calibração Convencional (PSO/Nelder & Mead)	Algoritmo AM		Algoritmo MH	
		Média distribuição posterior	Intervalo de Credibilidade (95%)	Média distribuição posterior	Intervalo de Credibilidade (95%)
SAT	1.790,18	1.793,16	[1.678,78; 1.911,39]	1.799,25	[1.680,62; 1.920,51]
PES	2,46	2,46	[2,36; 2,56]	2,46	[2,35; 2,55]
CREC	35,35	35,43	[31,65; 39,50]	35,17	[31,19; 39,11]
K	4,27	4,43	[3,31; 5,65]	4,32	[3,15; 5,54]
σ	20,45	20,53	[19,47; 21,67]	20,53	[19,47; 21,64]
NS	0,88	-		-	

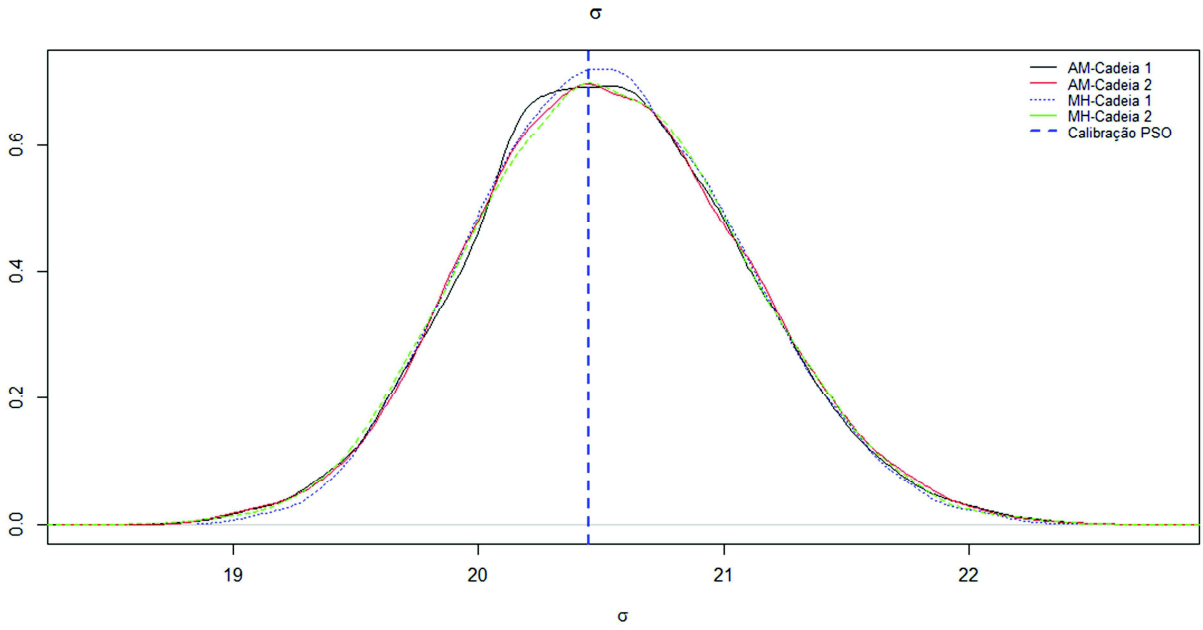

Figura 6.24– Distribuição posterior do desvio padrão do erro do modelo para as cadeias dos algoritmos AM e MH

No que concerne ao desempenho dos algoritmos implementados, conforme verificado nos gráficos apresentados (Figura 6.23 e Figura 6.24) e no resumo das distribuições posterioresnaTabela 6.13, destaca-se que ambos resultaram em distribuições posteriores equivalentes para os parâmetros.

O algoritmo MH apresentou para o caso simulado taxas de aceitação ligeiramente superiores ao algoritmo AM, contudo, tal constatação não pode ser entendida como melhor eficiência do algoritmo MH, tendo em vista que no caso para as vazões sintéticas a taxa de aceitação do algoritmo AM foi superior. Ademais, tal como relatado anteriormente a distribuição de proposição afeta sobremaneira as taxas de aceitação do método sendo, portanto, um fator que pode influenciar significativamente nesse aspecto. Avalia-se, portanto, que sob este aspecto é difícil estabelecer um parâmetro equivalente de comparação entre os dois algoritmos tendo em vista que ambos realizam a proposição por distribuições diferentes e que dependendo dos parâmetros adotados podem melhorar ou reduzir as taxas de aceitação do algoritmo.

6.1.3.5. - Avaliação dos resíduos

Para este caso admitiu-se o modelo probabilístico dos resíduos que considera a normalidade, independência e homocedasticidade. Observa-se que o modelo conseguiu reproduzir razoavelmente as vazões observadas. As médias das distribuições posteriores foram praticamente equivalentes aos valores obtidos para o vetor de parâmetros ótimos pela calibração convencional. As vazões geradas pelo modelo por meio desses parâmetros apresenta um valor de 0,88 para métrica NS, indicando uma aproximação razoável entre as vazões simuladas e as vazões observadas.

Contudo, avaliando os resíduos do modelo, verifica-se que as hipóteses de normalidade, independência e homocedasticidade não se confirmam. A Figura 6.25 apresenta a distribuição padronizada obtida para os erros do modelo simulado confrontada com a distribuição normal padrão evidenciando que os resíduos não seguem o modelo Gaussiano. O gráfico quantil-quantil da distribuição normal apresentado na Figura 6.26 confirma o desvio em relação a normalidade tendo em vista que os pontos não aderem a reta representativa da distribuição normal. Essa constatação é importante, uma vez que assumir uma premissa equivocada para o comportamento dos erros interfere na série de vazões modeladas.

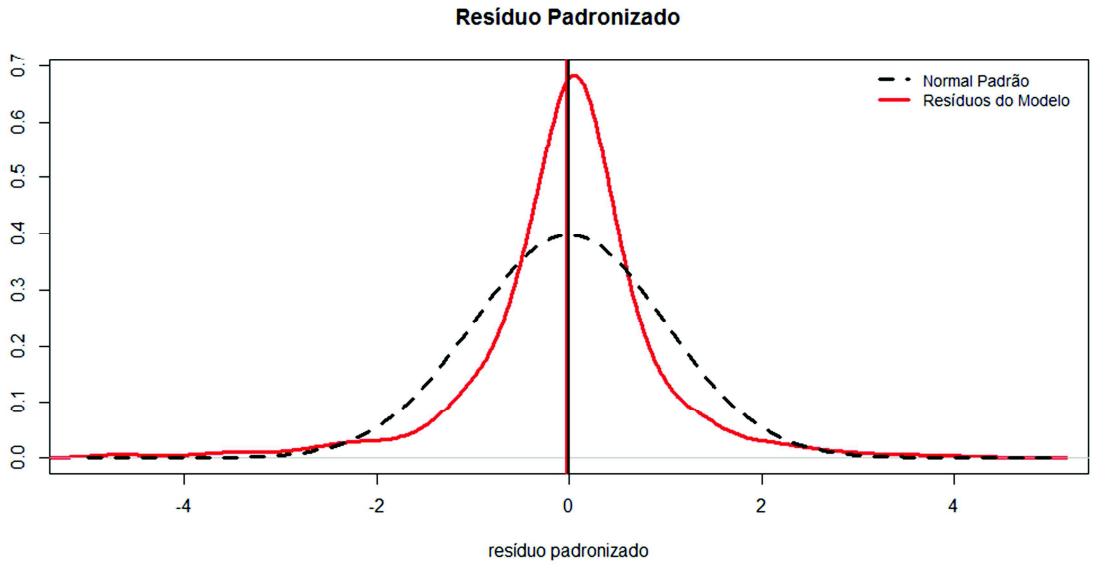

Figura 6.25– Distribuições do erro padronizado do modelo simulado comparado com uma distribuição normal padrão com média 0 e desvio padrão 1

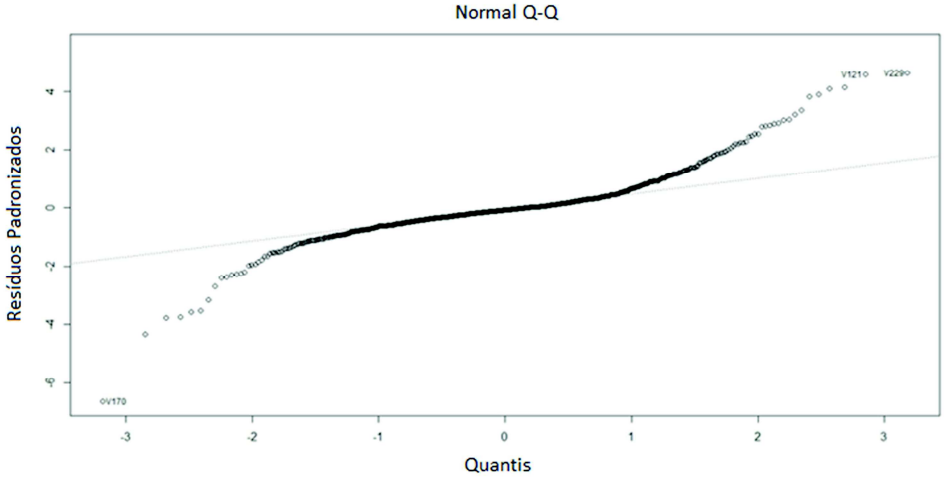

Figura 6.26– Resíduos do modelo em função da vazão simulada

A Figura 6.27 apresenta os resíduos do modelo em função da vazão simulada. É perceptível na imagem uma tendência a maior dispersão a medida que o valor da vazão aumenta. Tal ocorrência suscita a violação da homocedasticidade assumida, indicando que a variância dos erros certamente deve aumentar com o incremento da vazão.

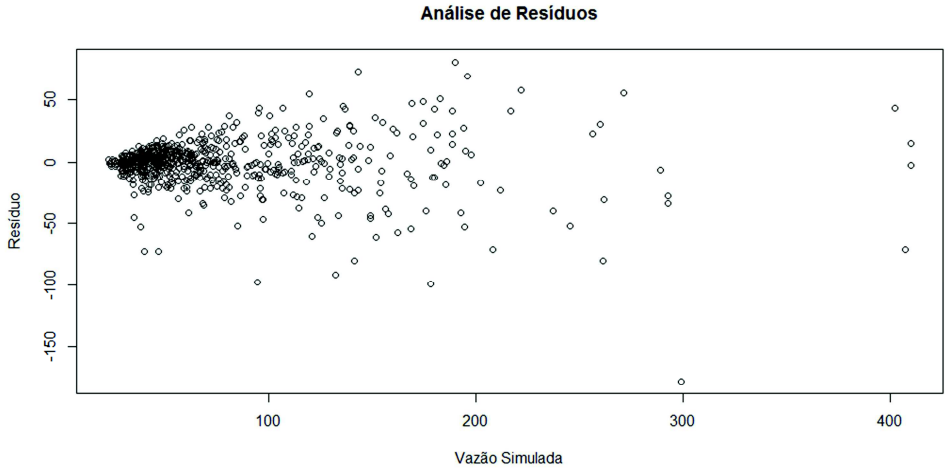

Figura 6.27– Resíduos do modelo em função da vazão simulada

Quanto à independência dos erros, os valores da autocorrelação (ACF) para diferentes lags indicam dependência entre os resíduos, conforme mostrado na Figura 6.28. Destaca-se, contudo, que Silva *et al.* (2014) em simulação de modelo hidrológico com passos de tempo diário identificaram forte correlação entre os erros do modelo. Acredita-se que a escala temporal pode influenciar neste quesito.

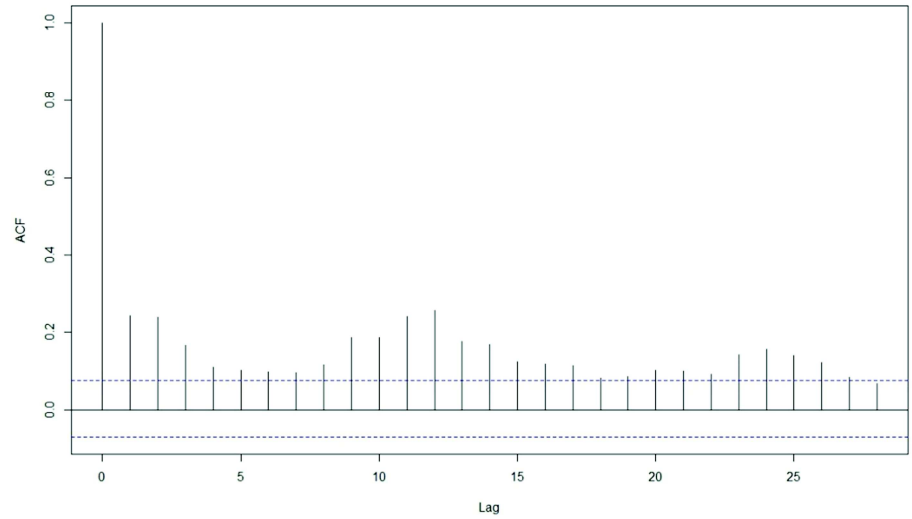

Figura 6.28– Autocorrelação dos resíduos do modelo normal

Outros estudos (Schoups & Vrugt, 2010; Silva *et al.,* 2014; Smith *et al.,* 2015) apontam para comportamentos semelhantes dos resíduos em modelos hidrológicos, ou seja, a presunção de que os resíduos podem ser considerados homocedásticos, independentes e gaussianos não é válida para a maior parte dos casos de simulação hidrológica.

Com o intuito de buscar o segundo objetivo específico do trabalho, na Seção 6.2 - Influência da Função de Verossimilhança, emprega-se a função de verossimilhança generalizada proposta por Schoups & Vrugt (2010) com o intuito de avaliar o impacto da função de verossimilhança na estimativa dos parâmetros do modelo bem como na representação dos resíduos.

6.1.3.6. - Incertezas nas vazões simuladas

Com as cadeias dos valores de parâmetros obtidos foram processadas 90.000 séries de vazões mensais, das quais se extraiu o intervalo de credibilidade de 95%. Os intervalos de credibilidade obtidos foram confrontados com os dados observados e a série obtida a partir do vetor de parâmetros ótimos definido a partir da calibração convencional com o algoritmo PSO. A Figura 6.29 apresenta os resultados para a série completa simulada. A Figura 6.30 apresenta um detalhe de um período de 100 meses da série. Tendo em vista a extensão da série, visualmente não fica nítido quantos valores observados estão contidos no intervalo de credibilidade obtido.

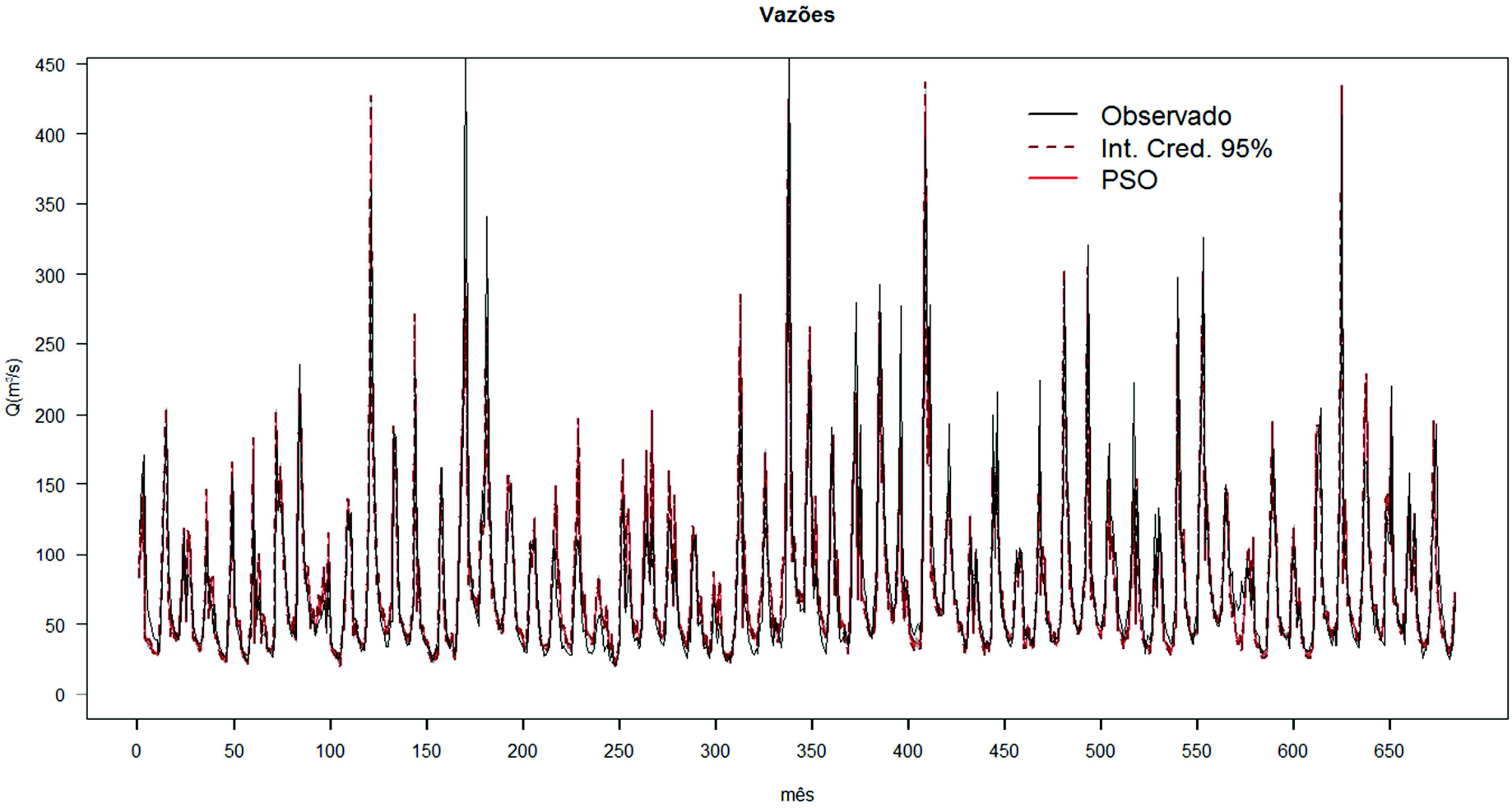

Figura 6.29– Vazões modeladas com intervalos de credibilidade, com parâmetros ótimos da calibração convencional e valores observados

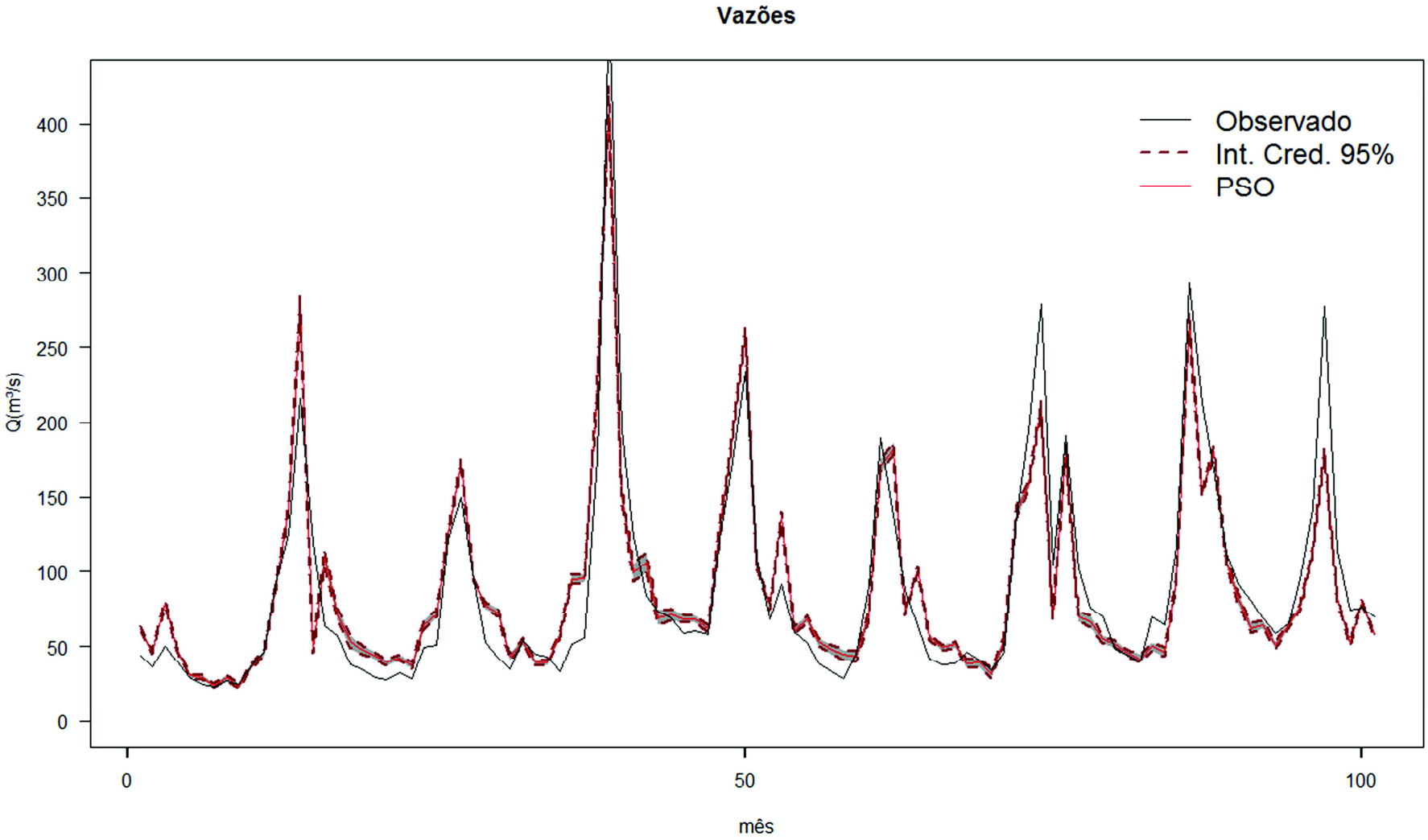

Figura 6.30– Detalhe de um trecho de 100 meses do histórico modelado indicado

Contudo, contabilizando-se individualmente cada um dos valores e comparando-os com os respectivos intervalos de incertezas verificou-se que em cerca 80% do tempo simulado os dados observados estão entre os limites de incertezas associados. Essa constatação coaduna com a situação observada em simulação similar com passos de tempo diário apresentada por Silva (2015). É conveniente destacar que, tal como ressaltado pelo referido autor, a abordagem aqui apresentada foca na incerteza advinda especificamente dos parâmetros do modelo desconsiderando àquelas oriundas dos dados observados e da própria estrutura do modelo, não obstante constata-se que métricas usuais para procedimentos tradicionais de calibração tal como o parâmetro NS indicou um bom ajuste do modelo.

6.2 - INFLUÊNCIA DA FUNÇÃO DE VEROSSIMILHANÇA

A última simulação visou a consecução do terceiro objetivo almejado que consiste na análise da influência das premissas adotadas em relação à natureza dos erros do modelo no processo de calibração do modelo hidrológico SMAP mensal. Ou seja, buscou-se avaliar a influência da função de verossimilhnaça na estimativa dos parâmetros do modelo e na descrição das incertezas das vazões.

O algoritmo DREAM foi empregado para o mesmo caso com dados reais apresentados na seção anterior. O principal objetivo desta simulação foi avaliar o efeito da função de verossimilhança nas distribuições posteriores dos parâmetros do modelo hidrológico e na modelagem dos resíduos, além de validar os algoritmos implementados AM e MH.

Foi utilizada a função de verossimilhança generalizada (GL) para o modelo dos resíduos. Os parâmetros da função generalizada também foram ajustados($\sigma_1 = 0$; $\phi_i = 0$; $\beta = 0$; $\xi = 1$ e μ_M=0) para representar a normalidade (N) sendo possível, portanto, contrapor os resultados obtidos para o modelo de resíduos via função generalizada (GL) e gaussiana (N). Dessa forma, ao se configurar a função generalizada para representar a normalidade, independência e homocedasticidade de resíduos estabelece-se a mesma hipótese considerada na simulação anterior com os algoritmos AM e MH, com a diferença de que neste caso fez-se uso do algoritmo DREAM.

Os limites de busca dos parâmetros do modelo hidrológico foram definidos conforme descrito no caso anterior, tendo-se mantido os limites de busca dos parâmetros do modelo

hidrológico tal como apresentados naTabela 6.10. Os parâmetros referentes à taxa de umidade inicial e o escoamento de base (TUin, Ebin) também foram os mesmos fixados no caso anterior, ou seja, os valores 0,4 e 20 m³/s, respectivamente.

No que se refere às funções de verossimilhança de modelagem dos resíduos, para o modelo que assume a normalidade e independência dos erros (N), o parâmetro a ser obtido consiste no desvio padrão do erro do modelo (σ). A fim de se estabelecer os limites de busca deste parâmetro, partiu-se do valor obtido para o desvio padrão dos resíduos quando se consideram as vazões processadas com o modelo hidrológico adotando-se os parâmetros ótimos definidos via da calibração convencional. Assim, considerou-se como valores superiores e inferiores a ordem de grandeza deste valor acrescido e decrescido de seu próprio módulo.

Para função de verossimilhança generalizada (N) os parâmetros a serem definidos consistem nos coeficientes angular (σ_1) e linear (σ_0) da reta definida pela Equação (4.24), que simula a heterocedasticidade dos resíduos, os parâmetros de assimetria (ξ) e curtose (β) e o parâmetro de autocorrelação (ϕ_i). Admitiu-se como nulo o parâmetro de viés da vazão modelada (μ_M). Ainda com relação ao parâmetro de autocorrelação do polinômio do modelo autoregressivo(4.21), considerou-se apenas um parâmetro, dado por ϕ_1.

Os limites superior e inferior dos parâmetros foram estabelecidos após diversas simulações testes realizadas com base nos limites adotados por Silva (2015) e Shoups & Vrugt (2010).

A Tabela 6.14 resume os limites estabelecidos para os parâmetros comentados.

Tabela 6.14- Limites de avaliação dos parâmetros das funções de verossimilhança

Modelo	Símbolo	Descrição	Limite inferior	Limite superior	Valor fixo
Normal (N)	σ	Desvio padrão do erro do modelo	0	50	-
Generalizada (GL)	σ_0	Heterocedastiticade: intercepto	-10	10	-
	σ_1	Heterocedasticidade: inclinação	0	1	-
	β	Curtose	-0,99	2,5	-
	ξ	Assimetria	0.1	10	
	ϕ_1	Coeficientes de autocorrelação	0	1	-
	μ_M	Viés da vazão modelada	-	-	0

Os parâmetros referentes a simulação Monte Carlo via cadeias de Markov (número de cadeias, número máximo de passos, limite da estatística R de Gelman e Rubin e número de intervalos de avaliação da função de verossimilhança) foram também definidos após diversas simulações testes e com base nas observações destacadas por Silva (2015). Para o número de cadeias (*nseq*), o autor mencionado reportou não ter identificado maiores ganhos ou anomalias em função deste parâmetro.

Destaca-se, contudo, que como o método DREAM faz uso da estatística R de Gelman e Rubin para avaliação de convergência, o valor mínimo para este parâmetro deve ser 2. Realizaram-se testes com os valores deste parâmetro variando de 2,3 e 5, tendo-se adotado para apresentação neste trabalho os resultados obtidos para 3 cadeias.

No que concerne ao número máximo de passos nas cadeias (*ndraw*) observou-se, para o caso simulado, que valores abaixo de 10.000 iterações, para os testes realizados, nem sempre garantiam o atendimento do critério de convergência da estatística R de Gelman e Rubin.

Tendo-se como referência a simulação apresentada na seção anterior, que aplicou algoritmo AM ao mesmo caso observando-se um desempenho razoável desse algoritmo para 150.000 iterações (ou seja, atendimento ao critério de convergência com base no diagnóstico da estatística R de Gelman e Rubin), adotou-se para o número de passos máximo das cadeias o valor de 150.000 iterações, tendo-se confirmado após a simulação o atendimento ao critério de convergência da estatística R (*Rthres*), definido como valor máximo admissível 1,01, conforme sugerido por Gelman (1996).

O valor do parâmetro definido para o número de intervalos entre as avaliações da função de verossimilhança (*thin.t*) foi estabelecido após a verificação dos indicadores de autocorrelação obtidos para a simulação que assumiu a normalidade do erro, tendo-se admitido que a partir do décimo parâmetro da amostra não haja mais autocorrelação significativa. O número de cadeias a serem descartadas (*burn in*) da amostra adotada para composição das distribuições posteriores foi definida em 10.000 iterações. A Tabela 6.15 resume os parâmetros considerados para o processamento do método DREAM.

Tabela 6.15- Parâmetros considerados para as cadeias de Markov via simulação Monte Carlo

Parâmetro	**Descrição**	**Valor**
nseq	Número de cadeias	3
ndraw	Número de iterações	150.000
Rthres	Estatística R de Gelman-Rubin	1,01
thin.t	Número de intervalos de avaliação da verossimilhança	10
burn-in	Aquecimento da cadeia	10.000

6.1.3.7. - Avaliação das incertezas nos parâmetros simulados

A Figura 6.31apresenta as distribuições posteriores para os parâmetros do modelo hidrológico considerando a hipótese de resíduos modelados pela distribuição normal (linha contínua escura) e pela distribuição exponencial potência simétrica (SEP) (linha tracejada vermelha) considerada na função de verossimilhança generalizada (GL).

A Tabela 6.16 resume as estatísticas dos parâmetros modelados para cada um dos modelos simulados, incluindo os resultados obtidos para os parâmetros ao se empregar a calibração convencional com o algoritmo PSO seguido do método Simplex de busca local (Nelder & Mead, 1965) com base na métrica Nash-Suticliffe (NS), tradicionalmente empregada em modelos de simulação hidrológica (Equação (3.1). Esses valores tratam-se dos mesmos apresentados na simulação anterior e foram repetidos na tabela para facilitar a comparação com os novos resultados.

Observa-se quanto às distribuições posteriores obtidas para os parâmetros SAT e CREC que o comportamento é semelhante para os dois modelos de resíduos (N e GL), sendo os intervalos de credibilidade equiparáveis, assim como as médias das distribuições que se aproximam, inclusive, dos valores ótimos obtidos pelo processo de calibração convencional. Contudo, para os parâmetros PES e K, embora as feições das distribuições posteriores apresentem certa similaridade para os dois modelos de resíduos, seus limites de credibilidade e as médias das distribuições se distanciam sobremaneira.

A média das distribuições posteriores para o modelo que assume a normalidade dos resíduos se mostrou muito próxima dos valores encontrados para os parâmetros ótimos via calibração convencional. No que concerne, especificamente, às distribuições posteriores dos parâmetros do modelo hidrológico obtidas por meio do modelo N dos resíduos verifica-se equivalência dos resultados obtidos para condições similares processadas com os algoritmos AM e MH descritos no caso anterior.

A Figura 6.32 e a Figura 6.33 apresentam as distribuições posteriores dos parâmetros da função de verossimilhança generalizada (GL) e a função que assume a normalidade dos resíduos (N) respectivamente.

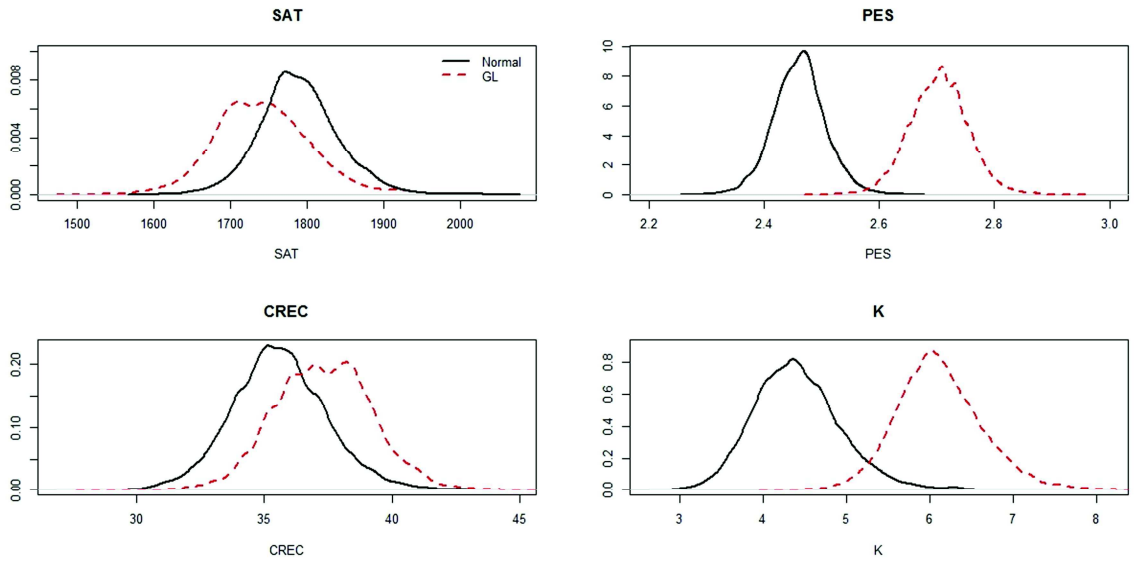

Figura 6.31– Distribuições posteriores para os parâmetros do modelo hidrológico SMAP-Mensal para os modelos de resíduos avaliados

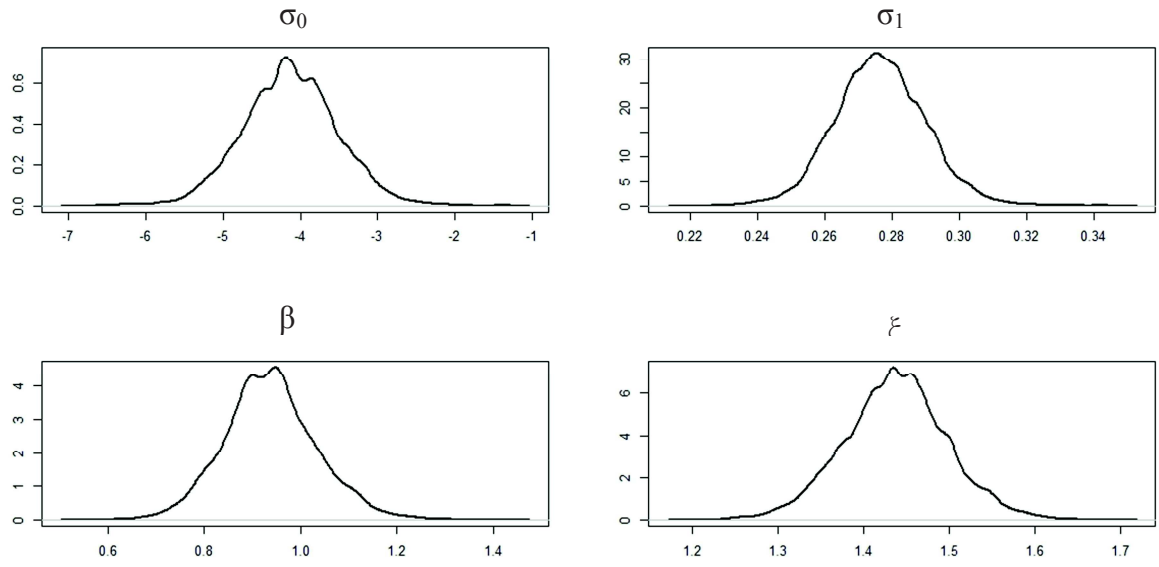

Figura 6.32– Distribuições posteriores para os parâmetros da função de verossimilhança generalizada (GL)

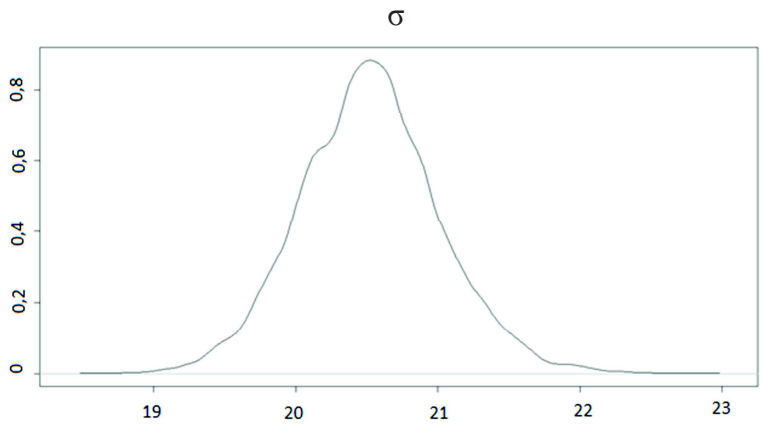

Figura 6.33– Distribuição posterior do parâmetro da função de verossimilhança assumida a normalidade dos resíduos (N)

6.1.3.8. - Avaliação dos resíduos dos modelos simulados

O principal propósito desta análise foi verificar o ganho na modelagem dos resíduos ao se adotar a função de verossimilhança generalizada. A Figura 6.34 apresenta o modelo dado pela função SEP e os resíduos padronizados indicando uma melhora expressiva no ajuste, quando comparado ao modelo de resíduos que assume a normalidade (Figura 6.25).

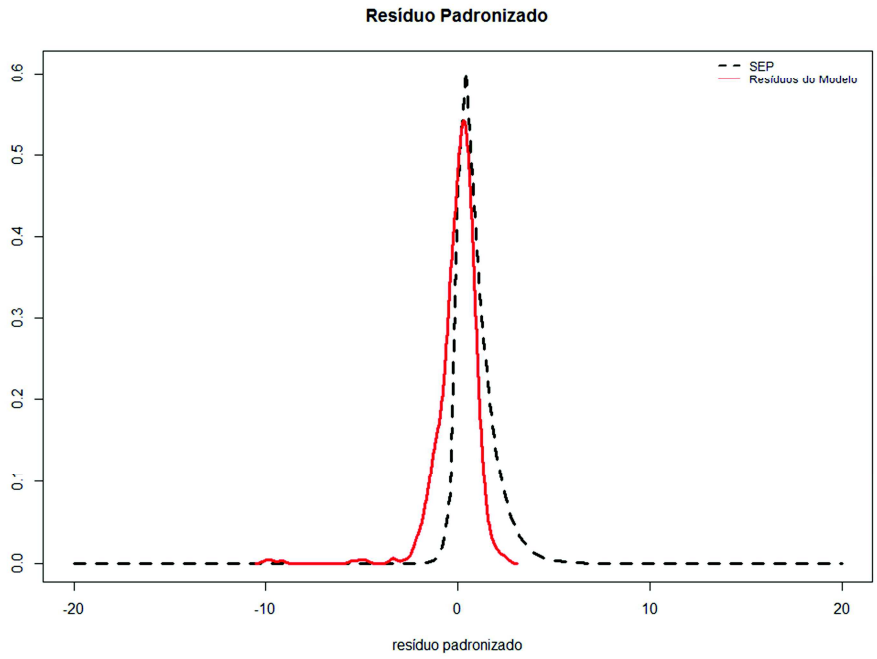

Figura 6.34– Distribuições do erro padronizado do modelo simulado comparado com a distribuição SEP

Verifica-se ainda que o modelo GL permitiu a remoção da heterocedasticidade dos resíduos, bem como reduziu significativamente a correlação de ordem 1 entre os valores do erro, conforme pode ser observado na Figura 6.35 e Figura 6.36, quando comparadas a Figura 6.27 e Figura 6.28 que apresentam os mesmos resultados para o modelo que assume a normalidade e independência.

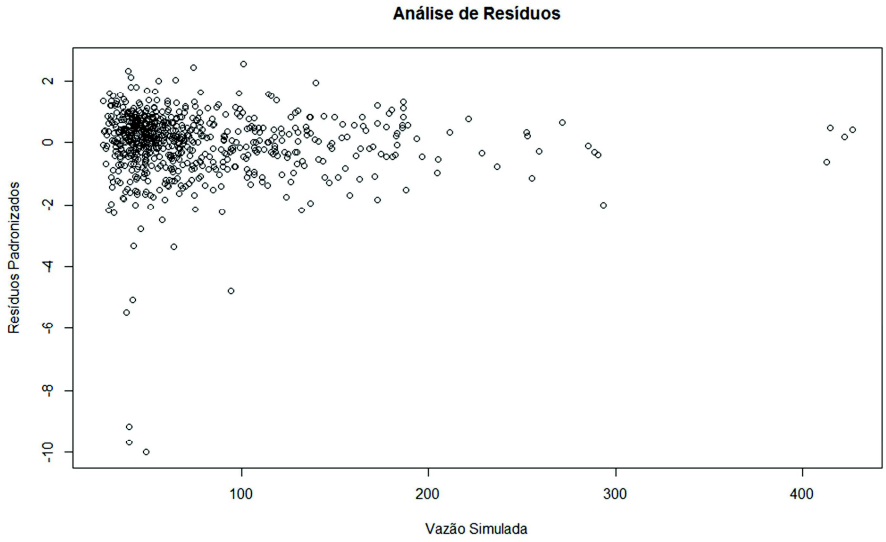

Figura 6.35– Resíduos do modelo GL em função da vazão simulada

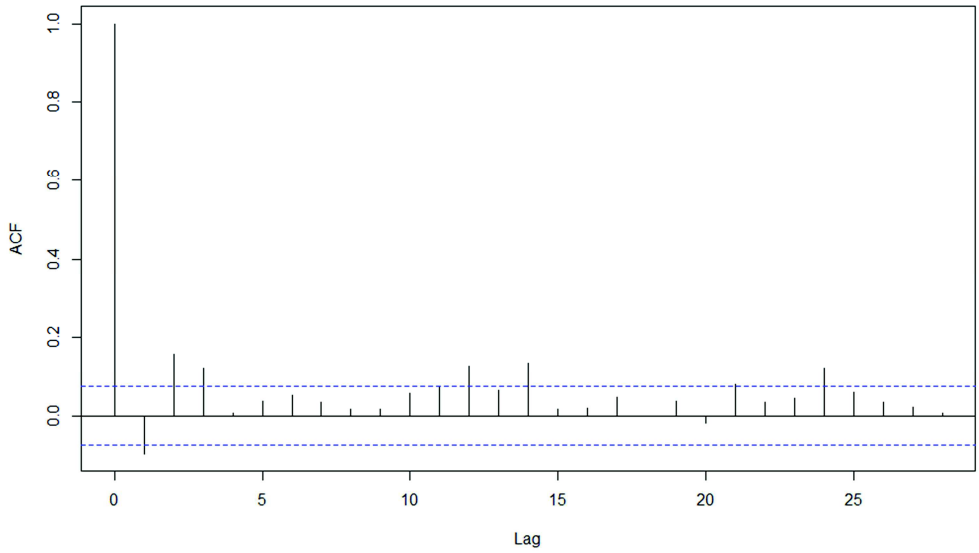

Figura 6.36– Autocorrelação dos resíduos do modelo GL

Dessa forma, verifica-se que o modelo GL mostra-se superior revelando que os erros podem ser bem descritos a partir de função exponencial de potência assimétrica (SEP), supondo heterocedasticidade e autocorrelção entre os resíduos.

Considerando a influência expressiva da função de verossimilhança nas distribuições posteriores obtidas para os parâmetros simulados, ressalta-se a relevância de se avaliar este fator quando da calibração dos parâmetros de modelos hidrológicos. Esta situação se impõe tendo vista que nos casos de simulação hidrológica usualmente empregam-se os parâmetros calibrados paraefetuar previsões.

Verifica-se que mesmo sob a égide da percepção bayesiana, que trata os parâmetros de forma probabilística e transcende a busca por um conjunto ótimo de parâmetros, pode-se assumir distribuições para os parâmetros de forma incipiente, caso não se avalie adequadamente o comportamento dos resíduos do modelo após os ajustes efetuados.

6.1.3.9. - Avaliação das incertezas nas vazões modeladas

A série de vazões mensais com intervalos de credibilidade de 95%obtidos para o modelo N e GL é apresentada na Figura 6.37. A Figura 6.38 apresenta um detalhe em um trecho da série para melhor visualização. Embora não seja perceptível visualmente o ganho na estimativa das incertezas das vazões modeladas, ao se empregar a função GL verificou-se que em cerca de 90% do tempo simulado as vazões observadas permaneceram nos intervalos de incerteza previstos, enquanto para o modelo N este valor ficou em cerca de 80%.

Em estudos realizados por Schoups & Vrugt (2010) para dois casos com discretização temporal diária observou-separa as séries simuladas uma redução mais expressiva dos intervalos de credibilidade ao se adotar o modelo GL. Contudo, espera-se que o ganho obtido no intervalo de predição dependa da sensibilidade dos parâmetros do modelo associadas às alterações decorrentes nas distribuições posteriores dos parâmetros. Nos casos apresentados pelos referidos autores as diferenças observadas nas distribuições posteriores dos parâmetros foram mais significativas tendo afetado todos os parâmetros do modelo adotado no estudo.

Tabela 6.16- Estatísticas de referência dos parâmetros considerados para simulação mensal

Parâmetro	Calibração Convencional (PSO/ Nelder & Mead)	Modelo N			Modelo GL		
		Média distribuição posterior	Intervalo de Credibilidade (95%)	Estatística R (Gelman-Rubin)	Média distribuição posterior	Intervalo de Credibilidade (95%)	Estatística R (Gelman-Rubin)
SAT	1.790,18	1.791,06	[1.693,12; 1.894,38]	1,0002	1.742,40	[1624,35; 1.865,50]	1,0002
PES	2,46	2,46	[2,37; 2,55]	1,0001	2,70	[2,60; 2,80]	1,0004
CREC	35,35	35,53	[32,02; 39,46]	1,0006	37,35	[33,72; 41,07]	1,0002
K	4,27	4,40	[3,43; 5,50]	1,0000	6,09	[5,16; 7,24]	1,0014
σ	20,45	20,52	[19,53; 21,51]	1,0005	-		
NS	0,88	-			-		
σ_0	-				-4,12	[-5,36; -2,97]	1,0006
σ_1					0,28	[0,25; 0,30]	1,0004
β					0,93	[0,75; 1,13]	0,9999
ξ					1,44	[1,32; 1,56]	1,0009
ϕ_1					0,42	[0,37; 0,47]	1,0010

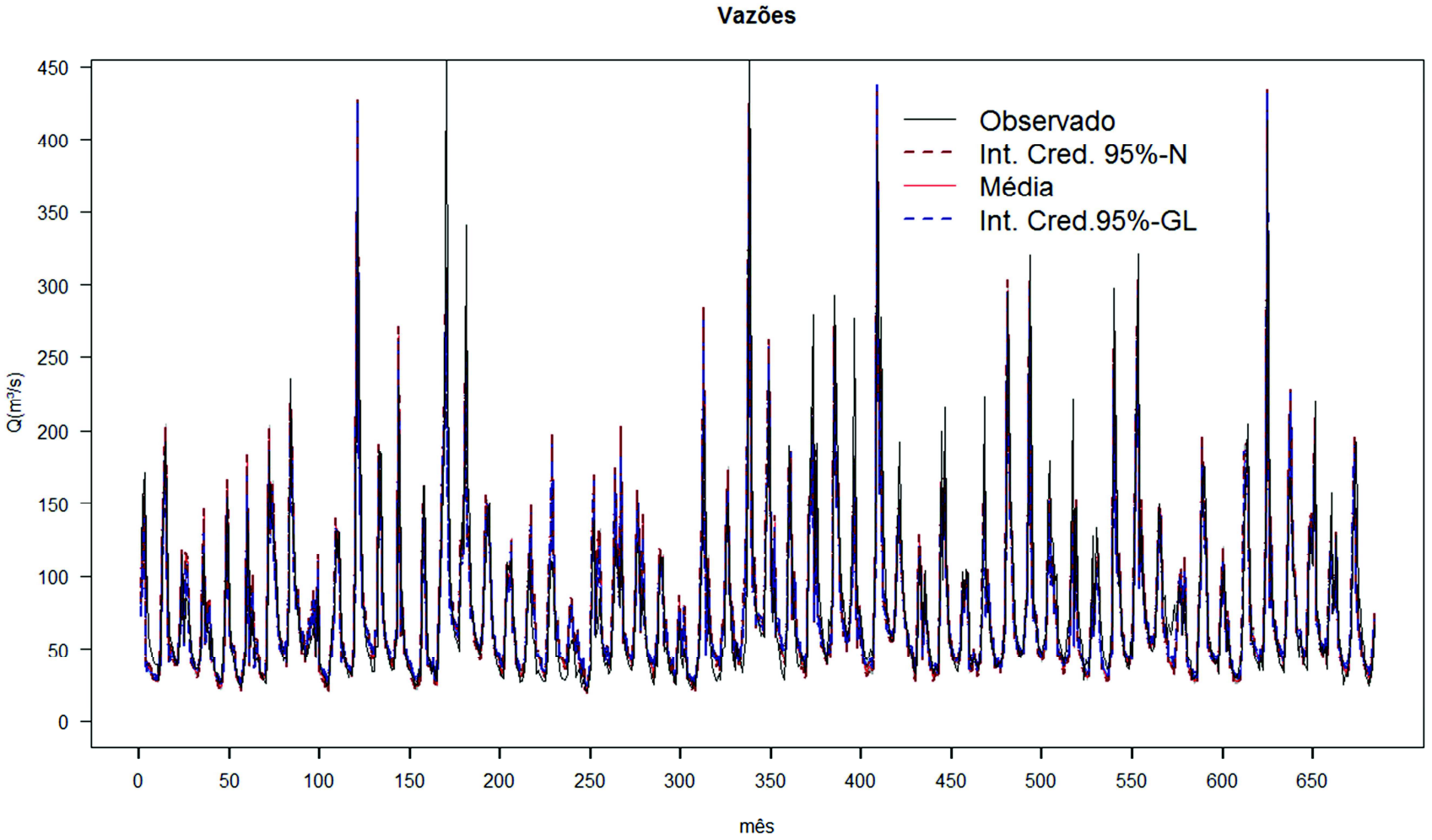

Figura 6.37– Vazões modeladas com intervalos de credibilidade, com parâmetros ótimos da calibração convencional e valores observados

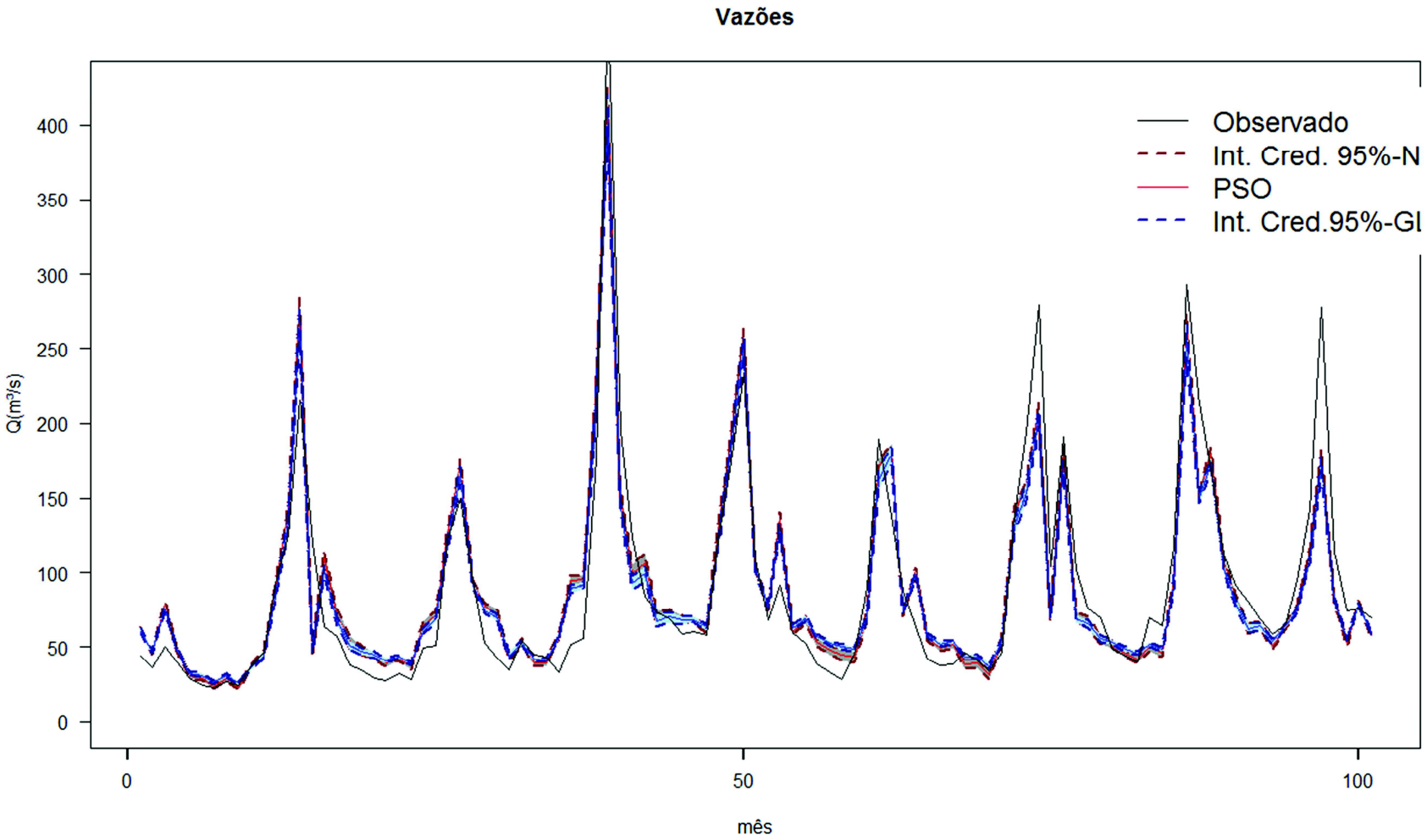

Figura 6.38– Detalhe de um trecho de 100 meses do histórico modelado indicado

7 - CONCLUSÕES E RECOMENDAÇÕES

Esta dissertação tratou da análise de incertezas associadas ao processo de predição de vazões a partir de modelos hidrológicos do tipo chuva vazão, tema de crescente interesse no campo da hidrologia. Modelar e descrever adequadamente o grau de precisão das vazões simuladas por um modelo hidrológico é uma tarefa fundamental na tentativa de fornecer aos interessados informação relevante que subsidiem o processo de tomada de decisão e de dimensionamento em diversas áreas da engenharia de recursos hídricos.

O estudo aqui apresentado se baseou numa abordagem Bayesiana para modelar as incertezas envolvidas no processo de modelagem hidrológica. Tal abordagem faz uso de técnicas MCMC (cadeias de Markov via simulação Monte Carlo) que permitem modelar e descrever as incertezas envolvidas no processo de modelagem, tendo como base um conhecimento *a priori* acerca dos parâmetros do modelo, assim como as informações hidrológicas disponíveis ao modelador. O resultado final dessa estratégia é a obtenção de distribuições probabilísticas de todos os parâmetros envolvidos no processo, não apenas daqueles inerentes ao do modelo hidrológico, mas também daqueles relacionados com o modelo previamente assumido para o erro. Com base nessas distribuições é possível estimar as incertezas nos valores de vazão simulados.

Este trabalho analisou (a) o desempenho de diferentes algoritmos MCMC na calibração de modelo hidrológicos e na descrição das incertezas envolvidas, e avaliou (b) a importância das premissas previamente adotadas em relação à natureza dos erros envolvidos. Para que essas questões pudessem ser exploradas, foram realizados quatro casos de simulação hidrológica, sendo dois sintéticos e dois empregando dados reais.

Os três primeiros casos buscaram validar e comparar o desempenho de três algoritmos MCMC (*Metropolis*, *Metropolis Hastings* e *Adaptive Metropolis*), sempre realizando uma comparação com um procedimento de calibração convencional, que emprega o algoritmo *Particle Swarm Optimization (PSO)*.

O primeiro caso consistiu em uma situação com dados reais e dimensão reduzida dos parâmetros do modelo hidrológico (SAT e PES) para uma situação de aplicação em região

do semiárido nordestino. Os resultados obtidos indicaram equivalência entre as distribuições posteriores obtidas para os parâmetros do modelo hidrológico, observando-se uma maior dificuldade do algoritmo M em atingir a convergência sob as mesmas considerações empregadas para os demais algoritmos.

O segundo caso consistiu em uma situação com dados sintéticos,em que os dados de vazão foram gerados artificialmente, a partir de valores pré-definidos para os parâmetros do modelo hidrológico, e depois corrompidos com erros advindos de uma distribuição normal com média nula e desvio padrão conhecido. Os resultados decorrentes dessa simulação indicaram que os algoritmos implementados conseguiram reproduzir adequadamente as características dos erros gerados sinteticamente, ou seja, foram capazes de estimar corretamente a variância do erro do modelo, elemento chave para a adequada descrição das incertezas nas vazões simuladas, assim como obtiveram distribuições posteriores, cujos intervalos de credibilidade continham os valores reais dos parâmetros do modelo hidrológico.

O terceiro caso aplicou os algoritmos MH e AM a uma situação com dados reais. A função de verossimilhança empregada na análise assumiu que os erros eram normalmente distribuídos, independentes e homocedásticos. Contudo, a análise dos resíduos obtidos indicou violação dessa hipótese. Ambos os algoritmos obtiveram distribuições posteriores similares para todos os parâmetros do modelo hidrológico, sendo que a média das distribuições possuíam valores próximos àqueles obtidos por meio da calibração convencional a partir do algoritmo PSO.

O último caso simulado buscou avaliar a influência da função de verossimilhança na estimativa das incertezas dos parâmetros do modelo, bem como o seu reflexo na predição da vazão para o modelo SMAP com discretização temporal mensal. Essas simulações foram realizadas com o algoritmo DREAM.

Foram avaliados dois modelos probabilísticos para os erros do modelo. Um primeiro que assume a normalidade dos erros, e outro que emprega uma função generalizada, que permite a consideração da não normalidade, heterocedasticidade e autocorrelação dos resíduos. Os resultados indicaram alta sensibilidade das distribuições dos parâmetros e das incertezas na predição das vazões à escolha função de verossimilhança. A função de

verossimilhança generalizada demonstrou-se mais adequada e bastante flexível para representação do comportamento dos resíduos em detrimento da hipótese que assume a normalidade dos resíduos.

No que concerne aos algoritmos empregados, o algoritmo DREAM mostrou-se mais eficiente do que os algoritmos M, MH e AM, tendo em vista que aquele permite um ajuste da distribuição proposta a partir de cadeias múltiplas perfazendo uma busca mais ampla do espaço amostral e, consequentemente, aumentando a efetividade do método.

Trabalhos futuros poderão agregar à estimativa das incertezas das vazões modeladas os erros advindos dos dados de entrada do modelo, bem como as informações de vazões observadas, com base nas incertezas associadas às curvas chave.

A consideração dos casos com vazões nulas e uma possível adaptação da função de verossimilhança generalizada para contemplar esses tipos de ocorrência em bacias semiáridas também consiste em uma possibilidade de pesquisa promissora. Outra abordagem de trabalho relevante é o emprego da inferência bayesiana para a avaliação das incertezas em procedimentos de regionalização de parâmetros de modelos hidrológicos

REFERÊNCIAS BIBLIOGRÁFICAS

Abbott, M. B., Bathurst, J. C., Cunge, J. A., O'Connell, P. E., & Rasmussen, J. (1986). An introduction to the European Hydrological System. Journal of Hydrology, 87, pp. 45-59.

Arnold, J. G., Srinivasan, R., Muttiah, R. S., & Williams, J. R. (1998). Large area hydrologic modeling and assessment. Part I: Model development. Journal of The American Water Resources Association, 34, pp. 73-89.

Bárdossy, A. (2007). Calibration of hydrological model parameters for ungauged catchments, 11, pp. 703-710.

Barros, F. V. (2012). Estudos de Calibração dos Postos Fluviométricos do Estado do Ceará.Fortaleza: FUNCEME-COGERH.

Bates, B. C., & Campbell, E. P. (April de 2001). A Markov chain Monte Carlo scheme for parameter estimation and inference in conceptual rainfall-runoff modeling. Journal of Water Resources Research, 37, pp. 937-947.

Bekman, O. R., & Neto, P. L. (2009). Análise Estatística da Decisão (2ª Edição ed.). São Paulo: Blucher.

Beven, K. (2009). Environmental Modelling: An Uncertain Future? (1ª Edição ed.). Abingdon, Oxfordshire, England: Routledge.

Beven, K. J. (2001). Rainfall-Runoff Modelling - The Primer. England: John Willey & Sons, LTD.

Beven, K. J., & Binley, A. M. (1992). The future of distributed models: Model calibration and uncertainty prediction. Journal of Hydrological Processes, 6, pp. 279-298.

Beven, K., & Hall, J. (2014). Applied Uncertainty Analysis for Flood Risk Management. Imperial College Press.

Boyle, D. P. (2001). Multicriteria calibration of hydrologic Models. Tucson: Ph.D. Dissertation, Department of Hydrology and Water Resources, University of Arizona.

Calver, A. (1988). Calibration, sensitivity and validation of a physically based rainfall-runoff model. Journal of Hydrology, 103, pp. 102-115.

Campbell, E. P., & Bates, B. C. (2001b). Regionalization of rainfall-runoff model parameters using Markov Chain Monte-Carlo samples. Journal of Water Resources Research, 37, pp. 731-739.

Chib, S., & Greenberg, E. Understanding the Metropolis-Hastings Algorithm. The American Statistician, 49, pp. 327-335.

Collischonn, W., Allasia, D., da Silva, B. C., & Tucci, C. E. (2007). The MGB-IPH model for large-scale rainfall-runoff. Hydrological Sciences Journal ,12, pp. 878-895.

Cordeiro, J. d. (2009). Estimação Bayesiana de Parâmetros envolvidos em Modelos Determinísticos. Rio de Janeiro: Dissertação submetida ao Corpo Docente do instituto de Matemática - Departamento de Métodos Estatísticos da Universidade Federal do Rio de Janeiro - UFRJ, como parte dos requisitos necessários à obtenção do grau de Mestre em Estatística.

Costa, V. A. (2011). Modelos regionais de curvas de permanência de vazões para rios perenes, intermitentes e efêmeros, e seu emprego na calibração indireta de modelos de simulação hidrológica. Belo Horizonte: Dissertação apresentada ao Programa de Pós-graduação em Saneamento, Meio Ambiente e Recursos Hídricos da Universidade Federal de Minas Gerais.

Cowles, M. K., & Carlin, B. P. (1996). Markov Chain Monte Carlo Convergence Diagnostics: A comparative Review. Journal of the American Statistical Association, 91, pp. 883-904.

Duan, Q. Y., Gupta, V. K., & Sorooshian, S. (1993). Shuffled complex evolution approach for effective and efficient global minimization. Journal of Optimization Theory and Applications, 76, pp. 501-521.

Duan, Q., Gupta, H. V., Sorooshian, S., Rousseau, A. N., & Turcotte, R. (2002). Calibration of Watershed Models. Washington-DC: AGU Books Board.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. Proceedings of the Sisth International Symposium on Micromachine and Human Science , pp. 39-43.

Fernandes, W. S. (2009). Métodos para a estimação de quantis de enchentes extremas com o emprego conjunto de análise bayseiana, de informações não sistemáticas e de distribuições limitadas superiormente. Belo Horizonte: Tese de Doutorado apresentada ao Programa de Pós-Graduação em Saneamento, Meio Ambiente e Recursos Hídricos da Universidade Federal de Minas Gerais, como requisito à obtenção do título de Doutor em Saneamento, Meio Ambiente e Recursos Hídricos.

Fernandez, W., Vogel, R.,M., Sankarasubramanian, A. (2000). Regional calibration of a watershed model. Hydrological Sciences-Journal, 45, pp. 689-707.

Franchini, M. (1996). Use of a genetic algorithm combined with a local search method for the automatic calibration of conceptual rainfall-runoff models. Hydrological Sciences Journal, 41, pp. 21-37.

Freeze, R. A., & Harlan, R. L. (1969). Blueprint for a physically-based digitally simulated hydrologic response model. Hournal of Hydrology, 9, pp. 2161-2173.

FUNCEME - Fundação Cearense de Meteorologia e Recursos Hídricos. (2010). www.funceme.br. Acesso em 04 de Abril de 2013, disponível em http://www.funceme.br/index.php/areas/acudes-e-rios/regionalizacao

Gelman, A. (1996). Inference and monitoring convergence. In: W. R. Gilks, S. Richardson, & D. J. Spiegelhalter (Eds.), Markov Chain Monte Carlo in Practice (pp. 43-131). London: Chapman & Hall.

Gelman, A., & Rubin, D. (1992). Inference from Iterative Simulation Using Multiple Sequences. Statistical Science, 7, pp. 457-511.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2003). Bayesian Data Analysis (Second Edition ed.). London\Boca Raton: Chapman & Hall\CRC.

Guillaume, J., & Andrews, F. (2012). http://dream.r-forge.r-project.org/. Fonte: DiffeRentialEvolution Adaptive Metropolis. R packge version 0.4-2.

Gupta, V., Sorooshian, S., & Yapo, P. (1998). Toward improved calibration of hydrologic models: Multiple and noncommensurable measures of information. Water Resources Research, 34, pp. 751-763.

Haario, H., Eero, S., & Tamminen, J. (2001). An adaptive Metropolis algorithm. Bernoulli, 7, pp. 223-242.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. Biometrika, 57, pp. 97-190.

Hooke, R., & Jeeves, T. A. (1961). Direct search solution of numerical and statistical problems. J. Assoc. Comput. Mash, 8, pp. 212-229.

Hsu, K., Gupta, H. V., & Sorooshian, S. (1995). Artificial Neural Network Modeling of the Rainfall-Runoff Process. Water Resources Research, 31, pp. 2517-2530.

Jameson, A. (1995). Gradient Based Optimization Methods. MAE Technical Report No. 2057, Princeton University.

Kuczera, G., & Parent, E. (1998). Monte Carlo assessment of parameter uncertainty in conceptual catchment models: the Metropolis algorithm. Journal of Hydrology, 211, pp. 69-85.

Kuczera, G., Kavetski, D., Renard, B., & Thyer, M. (2010). A limited-memory acceleration strategy for MCMC sampling in hierarchical bayesian calibration of hydrological models. Journal of Water Resources Research, 7.

Liu, Y., & Gupta, H. V. (2007). Uncertainty in hydrologic modeling: Toward an integrated data assimilation framework. Journal of Water Research, 43, p. W07401.

Lopes, J. C., Braga Jr., B. F., & Conejo, J. L. (1981). Simulação Hidrológica: Aplicações de um modelo simplificado. Anais do III Simpósio Brasileiro de Recursos Hídricos, 2, pp. 42-62.

Madsen, H. (2000). Automatic calibration of a conceptual rainfall-runoff model using multiple objectives. Journal of Hydrology , pp. 276-288.

Marshall, L. (2005). Bayesian analysis of rainfall-runoff models: Insights to parameter estimation, model comparison and hierarchical model development. Sydney, Australia: A thesis sbmitted for the degree of Doctor of Philosophy at the University of New South Wales.

Marshall, L., Nott, D., & Sharma, A. (2004). A comparative study of Markov chain Monte Carlo methods for conceptual rainfall-runoff modeling. Journal of Water Resources Reserach, 40, pp. W0251.

Martinez, W. L., & Martinez, A. R. (2002). Computational Statistics Handbook with MATLAB. Chapman & Hall/CRC.

McIntyre, N., Young, P., Orellana, B., Marshall, M., Reynolds, B., & Wheater, H. (2011). Identification of nonlinearity in rainfall-flow response using data-based mechanistic modeling. Water Resources Research, 47, pp. W03515-1 a 14.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., & Teller, A. H. (1953). Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics, 21, pp. 1087-1092.

Montanari, A., Shoemaker, C. A., & van de Giesen, N. (2009). Introduction to special section on Uncertainty Assessment in Surface and Subsurface Hydrology: An overview of issues and challenges. Water Resources Research, 45, pp. W00B00-1-4.

Murray, W. (2010). Newton-type Methods. Department of Management Science and Engineering, Stanford University,CA.

Nascimento, L. V., Reis, D. J., & Martins, E. S. (2009). Avaliação do algoritmo evolutivo Mopso na calibração multiobjetivo do modelo SMAP no estado do Ceará. Revista Brasileira de Recursos Hídricos - ABRH, 14, 85-97.

Nash, J. E. (1960). A unit hidrograph study with particular reference to British catchments. Institution of Civil Engineers - Proceedings, 17, pp. 249-282.

Nash, J. E., & Sutcliffe, J. V. (1970). River flow forecasting through conceptual models, Part I - A discussion of principles. Jounal of Hydrology, 282-290.

Nelder, J. A., & Mead, R. (1965). A simplex algorithm for function minimization. Computer Journal, 7, pp. 308-313.

Neuman, S. P. (2003). Maximum likelihood Bayesian averaging of uncertain model predictions. Journal of Stochastic Environmental Research and Risk Assessment, 17, pp. 291-305.

Otsuki, R. G., & Reis, D. S. (Novembro/Dezembro de 2011). Análise comparativa de metodologias de estimativa de séries de vazões médias mensais aplicadas a estudos de aproveitamentos hidrelétricos. Anais do XIX Simpósio Brasileiro de Recursos Hídricos .

Parajka, J., Merz, R., Bloschl, R. (2005). A comparison of regionalisation methods for catchment model parameters. Hydrology and Earth System Sciences Discussions, 2, pp. 509-542.

Pinheiro, V. B. (2009). Calibração de um modelo chuva-vazão em bacias sem monitoramento fluviométrico a partir de curvas de permanência sintéticas. Belo Horizonte: Dissertação de Mestrado submetida ao Programa de Pós-graduação em Saneamento, Meio Ambiente e Recursos Hídricos da Universidade Federal de Minas Gerais.

R Core Team. (2013). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing.

Reis, D. S., & Stedinger, J. R. (November de 2005). Bayesian MCMC flood frequency analysis with historical information. Journal of Hydrology, 313, pp. 97-116.

Schoups, G., & Vrugt, J. A. (2010). A formal likelihood function for parameter and predictive inference of hydrologic models with correlated, heteroscedastic, and non-Gaussian errors. Water Resources Research, 46, pp. W10531-1-17.

Seibert, J. (1999). Regionalisation of parameters for a conceptual rainfall-runoff model. Agricultural and Forest Meteorology, 98-99, pp. 279-293.

Shu, C., & Burn, D. H. (2003). Spatial patterns of homogeneous pooling groups for flood frequency analysis. Hydrological Sciences Journal , pp. 601-618.

Silva, F. E. (2015). Assimilação do Padrão de Variabilidade das Variáveis de Estado de um Modelo Chuva-Vazão, em Esquemas de Simulação/Previsão Hidrológica. Belo

Horizonte: Tese de Doutorado apresentada ao Programa de Pós-Graduação em Saneamento, Meio Ambiente e Recursos Hídricos da Universidade Federal de Minas Gerais, como requisito à obtenção do título de Doutor em Saneamento, Meio Ambiente e Recursos Hídricos.

Silva, F. E., Naghettini, M., & Fernandes, W. (2014). Avaliação bayesiana das incertezas nas estimativas dos parâmetros de um modelo chuva-vazão conceitual. Revista Brasileira de Recursos Hídricos - RBH, 19, 148-159.

Singh, V. P., & Woolhiser, D. A. (Julho/Agosto de 2002). Mathematical Modeling of Watershed hydrology. Journal of Hydrologic Engineering , pp. 270-292.

Sivapalan, M., Takeuchi, K., fRANKS, S. W., Gupta, V. K., Karambiri, H., Lakshmi, V., et al., (06 de Dezembro de 2003). IAHS Decade on Predictions in Ungauged Basins (PUB), 2003-2012: Shaping an exciting future for the hydrological sciences. Hydrological Sciences Journal , pp. 857-880.

Smith, T. J. (2012). Conceptual hydrologic modeling: insights into bayesian analysis, model development, and predictions in ungauged basins. Bozeman: A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Ecology and Environmental Sciences.Monta State University.

Smith, T. J. (2012). Conceptual hydrologic modeling: Insights into Bayesian analysis, model development, and predictions in ungauged basins. Bozeman, Montana, USA: A dissertation submitted in partial fulfillment of the requirements for the degreee of Doctor of Philosophy in Ecology and Environmental Sciences.

Smith, T., Marshall, L., & Sharma, A. (2015). Modeling residual hydrologic erros with Bayesian inference. Journal of Hydrology, 528, pp. 29-37.

Smith, T., Sharma, A., Marshall, L., Mehrotra, R., & Sisson, S. (2010). Development of a formal likelihood function for improved Bayesian inference of ephemeral catchments. Water Resources Research, 46, pp. W12551-1-12.

Sumner, N. R. (1995). Calibration of a Conceptual Rainfall-Runoff Model using Simulated Annealing. A Thesis Submitted in Partial Fulfilment of the Requirements for the Award of Master of Science at the Faculty of Science and Technology, Edith Cowan University.

ter Braak, C. (2006). A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: easy Bayesian computing for real parameter spaces. Statistics and Computing, 16, pp. 239-249.

Thiemann, M., Trosset, M., Gupta, H., & Sorooshian, S. (2001). Bayesian recursive parameter estimation for hydrologic models. Journal of Water Resources Research, 37, pp. 2521-2535.

Todini, E. (1996). The ARNO rainfall-runoff model. Journal of Hydrology , pp. 339-382.

Tucci, C. E. (2012). Hidrologia: ciência e aplicação (4ª Edição ed.). Porto Alegre: UFRGS/ABRH.

Vrugt, J. A. (2015). Markov chain Monte Carlo Simulation Using the DREAM Software Package: Theory, Concepts, and MATLAB Implementation. Dream Manual.

Vrugt, J. A., Diks, C. G., Gupta, H. V., Bouten, W., & Verstraten, J. M. (2005). Improved treatment of uncertainty in hydrologic modeling: Combining the strengths of global optimization and data assimilation. Journal of Water Resources Research, 41, p. W01017.

Vrugt, J. A., Gupta, H. V., Bouten, W., & Sorooshian, S. (2003). A Shuffled Complex Evolution Metropolis algorithm for optimization and uncertainty assessment of hydrologic model parameters. Journal of Water Resources Research, 39, p. 1201.

Vrugt, J. A., ter Braak, C. J., Clark, M. P., Hyman, J. M., & Robinson, B. A. (2008). Treatment of input uncertainty in hydrologic modeling: Doing hydrology backward with Markov chain Monte Carlo simulation. Water Resources Research, 44, p. W00B09.

Vrugt, J. A., ter Braak, C. J., Diks, C. G., Higdon, D., Robinson, B. A., & Hyman, J. M. (2009). Accelerating Markov chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling. International Journal of Nonlinear Sciences and Numerical Simulation, 10, pp. 273-290.

Wagener, T., & Montanari, A. (2011). Convergence of approaches toward reducing uncertainty in predictions in ungauged basins. Jounarl of Water Resources Research, 47, p. W06301.

Wagener, T., & Wheater, H. S. (2006). Parameter estimation and regionalization for continuous rainfall-runoff models including uncertainty. Journal of Hydrology , pp. 132-154.

Wagener, T., Boyle, D. P., Matthew, J. L., Wheater, H. S., Gupta, H. V., & Sorooshian, S. (2011). A framework for development and application of hydrological models. Hydrology and Earth Sciences, 5, pp. 13-26.

Wagener, T., McIntyre, N., Lees, M. J., Wheater, H. S., & Gupta, H. V. (2003). Towards reduced uncertainty in conceptual rainfall-runoff modelling: Dynamic identifiability analysis. Journal of Hydrological Processess, 17, pp. 455-476.

Wagener, T., Sivapalan, M., Troch, P., & Woods, R. (2007). Catchment Classification and Hydrologic Similarity. Geography Compass , pp. 901-931.

Wagener, T., Wheater, H. S., & Gupta, H. V. (2004). Rainfall-Runoff Modelling in Gauged and Ungauged Catchments. London: Imperial College Press.

Wang, Q. J. (1991). The genetic algorithms and its application to calibrating conceptual rainfall-runoff models. Water Resources Research, 27, pp. 2467-2471.

Woolhiser, D. A., & Brakensiek, D. L. (1982). Hydrologic modelling of small watersheds. Hydrologic system synthesis, 5.

Yapo, P. O., Gupta, H. V., & Sorooshian, S. (1998). Multi-objective global optimization for hydrologic model. Journal of Hydrology, 204, pp. 83-97.

Young, P. (1988). Data-Based Mechanistic Modelling of Environmental, Ecological, Economic and Engineering Systems. 1867 a 1874. Lancaster, UK: Control and System Group, Centre for Research on Environmental Systems and Statistics (CRES).

Zhang, X., Srinivasan, R., Zhao, K., & Van Liew, M. (2009). Evaluation of global optimization algorithms for parameter calibration of a computationally intensive hydrologic model. Hydrological Process, 23, pp. 430-441.

APÊNDICE A – CÓDIGO FONTE: CASO COM DIMENSÃO REDUZIDA DE PARÂMETROS

```
#####################################################################
#UNIVERSIDADE DE BRASÍLIA-UnB
#FACULDADE DE TECNOLOGIA
#DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL
#PÓS GRADUAÇÃO EM TECNOLOGIA AMBIENTAL E RECURSOS HÍDRICOS
#Data: 01/08/2016
#Autor: Cássio Guilherme Rampinelli
#Orientador: Dirceu Silveira Reis Jr.
#Coorientador:Carlos Henrique Ribeiro Lima
#email:cassiorampinelli@gmail.com
#####################################################################
#Código desenvolvido no âmbito das atividades do trabalho
#de dissertação de mestrado do referido autor.
#Obs:Para solicitar os dados EPQ desta simulação enviar email
#para o referido autor
#_____________________________________________________________#

#####################################################################################
#################Caso real com dimensão reduzida de parâmetros######################
#####################################################################################

#Limpa a memória do R
rm(list=ls(all=TRUE))

#Seta arquivo de trabalho para o R
setwd("E:/HD_CÁSSIO/PASTA_CASSIO/UnB/Pós-
Graduação/Mestrado/Dissertação/Simulacoes-Metodologias/R/Bayesian-
Calibration/SimulacoesFinaisDissertacao/Simulacao_6_1_1")

#Carregando pacotes
library(MASS)#Pacote necessário para chamar a função normal multivariada

#Carrega Matriz com dados de Evaporação (E), Precipitação (P) e vazões
#Observadas(Q)
EPQ<-read.table(file="EPQ_6.1.1.txt",header=T)


##########################################
#PARTE 1: FUNÇÕES BÁSICAS
##########################################

########################################
#FUNÇÃO SMAP#############################
########################################
#INPUT-Argumento de entrada
#EPQ-Matriz com os dados de Evaporação, Precipitação e Vazão Observada
#para-Vetor de parâmetros do modelo, no caso:sat;pes
```

```
#########################################################
#OUTPUT
#Qcalc-Vetor com vazões simuladas
#########################################################


smap<-function(EPQ,para,Area) #Declaração da função smap

{

#########################
 #Inicializando variáveis#
 #########################

 E<-EPQ[,1] # Armazena a coluna E (evaporação), do data.frame EPQ no vetor E
 Prec<-EPQ[,2] # Armazena a coluna P (precipitação), do data.frame EPQ no vetor #Prec
 n<-length(Prec) # Armazena o tamanho do vetor Prec na variável n
 sat=para[1] #Armazena o elemento 1 do vetor para na variável sat
 pes=para[2] #Armazena o elemento 2 do vetor para na variável pes
 #crec=para[3] #Armazena o elemento 3 do vetor para na variável crec
 crec=0#para região do semiárido
 crecp=crec/100 #Transforma crec de percentual para centesimal
 #crecp=crec
 #k=para[4] #Armazena o elemento 4 do vetor para na variável k
 #ke=0.5^(1/k) # Calcula a constante de recessão do escoamento superficial
 ke=0#para região do semiárido
 #tuin=para[3] #Armazena o elemento 5 do vetor para na variável tuin
 #tuinp=tuin/100;#Transforma tuin de percentual para centesimal

 tuinp=0.4

 #ebin=para[6]#Armazena o elemento 6 do vetor para na variável ebin
 ebin=0

 # As variáveis seguintes consistem em vetores e serão inicializados como nulos, pois
 # o R requer uma definição a priore.
 Rsol<-0
 rsub<-0
 tu<-0
 dsol<-0
 Qcalc<-0
 es<-0
 er<-0
 eb<-0
 rec<-0

 #Inicialização dos primeiros elementos dos respectivos vetores
Rsol[1]=tuinp*sat
 rsub[1]=ebin/(1-ke)/Area*2630
tu[1]=tuinp
```

```
  ########################################
  #Implementação do Algoritmo do Modelo Smap#
  ########################################

  i<-2 # Variável i é inicializada como 2, pois os primeiros elementos dos
  # vetores de interesse já foram inicializados nas linhas anteriores

   while (i<=n+1) #Loop while que é processado até que o contador i atinga n+1

  {

   # Calcula a variação de nível do reservatorio do solo
   dsol[i]=0.5*(Prec[i-1]-Prec[i-1]*(tu[i-1]^pes)-E[i-1]*tu[i-1]-Rsol[i-1]*crecp*(tu[i-
1]^4))
   #Calcula a taxa de umidade no passo i
   tu[i]=(Rsol[i-1]+dsol[i])/sat
   #Calcula o escoamento de base

   if(is.na(tu[i])){tu[i]=0}
   if(tu[i]<0){tu[i]=0}
   if(tu[i]>1){tu[i]=1}

   es[i]=Prec[i-1]*tu[i]^pes
   #Calcula a evaporação real
   er[i]=E[i-1]*tu[i]
   #Calcula a recarga do reservatorio subterrâneo
   rec[i]=Rsol[i-1]*crecp*(tu[i]^4)
   #Calcula o escoamento de base
   eb[i]=rsub[i-1]*(1-ke)
   #Atualiza o nível do reservatório subterrâneo
   rsub[i]=rsub[i-1]-eb[i]+rec[i]
   #Atualiza o nível do reservatorio do solo
   Rsol[i]=Rsol[i-1]+Prec[i-1]-es[i]-er[i]-rec[i]


if(is.na(Rsol[i])){Rsol[i]=0}
if(Rsol[i]>sat)
   {es[i]=es[i]+Rsol[i]-sat
   Rsol[i]=sat
   }

   if(Rsol[i]<0){Rsol[i]=0}

#Calcula vazão total simulada
   Qcalc[i-1]=(es[i]+eb[i])*Area/2630

    i<-i+1 #atualiza o contador
   }
  return(Qcalc) # retorna a série simulada
```

```
 }
########################################
#FUNÇÃO OBJETIVO 1:NASH-SUTICLIFFE#####
########################################
#INPUT-Argumento de entrada
#para-Vetor de parâmetros do modelo, no caso:sat;pes
#########################################################
#OUTPUT
#NS1-Valor da função objetivo x (-1)
#########################################################

FO1<-function (para)

{
 pen=0 #Inicializa penalidade

 #Verifica se os parâmetros estão dentro dos limites, caso não estejam
 #atribui-se penalidade à função objetivo
 if(para[1]>=limites[2,1]) pen=10000000000000000000
 if(para[1]<=limites[1,1])pen=10000000000000000000
 if(para[2]>=limites[2,2]) pen=10000000000000000000
 if(para[2]<=limites[1,2]) pen=10000000000000000000
 #if(para[3]>=limites[2,3]) pen=10000000000000000000
 #if(para[3]<=limites[1,3]) pen=10000000000000000000
 #if(para[4]>=limites[2,4]) pen=10000000000000000000
 #if(para[4]<=limites[1,4]) pen=10000000000000000000
 #if(para[5]>=limites[2,5])pen=10000000000000000000
 #if(para[5]<=limites[1,5]) pen=10000000000000000000
 #if(para[6]>=limites[2,6]) pen=10000000000000000000
 #if(para[6]<=limites[1,6]) pen=10000000000000000000

 Qcalc<-smap(EPQ,para,Area)#Executa a função SMAP
 Qobs<-EPQ[,3]#Armazena os valores das vazões observadas na variável Qobs

 #Calcula a função objetivo
 NS1<-(-1)*(1-((sum((Qobs-Qcalc)^2))/(sum((Qobs-mean(Qobs))^2))))+pen

 #Caso a função objetivo retorne um valor não numerico, atribui-se uma
 #penalidade
 return(ifelse(is.na(NS1), 10000000000000000000, NS1))

}

########################################
#FUNÇÃO PSO#############################
########################################
#INPUT-Argumento de entrada
#funcao-Função objetivo a ser considerada
#limites-limites de busca do vetor de parâmetros
#########################################################
```

```
#OUTPUT
#result-Lista com o vetor de parâmetros ótimos e o valor da função objetivo
#########################################################

#Define a função PSO,que tem como argumento uma função e uma matriz com os limites
#de busca tal qual como indicado acima.
PSO<-function(funcao,limites)
{

 n<-ncol(limites)#Denife o numero de colunas=par ordenado
 npar<-50#Número total de partículas a serem geradas
 maxger<-100#Número total de gerações, ou iterações
 POP<-matrix(0,nrow=npar,ncol=n)#Cria matriz que armazenará a população inicial
#############################################################################
 ####Geração da PopulaçãoInicial#########################################
#############################################################################

for (i in 1:npar) # Percorre o número de particulas até npar

 {
  for (j in 1:n)# Cria as coordenadas do par ordenado para cada particula

  {
   #Forma de gerar as coordenadas dos pontos aleatoriamente
   POP[i,j]= (limites[2,j]-limites[1,j])*runif(1)+limites[1,j]
  }

 }

#############################################################################
###Inicialização e definição de algumas variáveis e constantes de interesse###
#############################################################################

 criterio<-0#Define o critério de parada
 geracao<-0#Inicializa um contador para o número de gerações ou iterações

 #Matriz que armazena o menor valor da função obtido pela particula
 pbestAJ<-matrix(0,nrow=1,ncol=npar)+9999999999999999999

 #Matriz que armazena as coordenadas (posição) do menor valor de funçao que
 # a particula já obteve
 pbestXX<-matrix(9999999999999999999,nrow=npar,ncol=n)

 #Armazena velocidade
 vant<-matrix(0,nrow=npar,ncol=n)

 #Constantes ou parâmetros de confiança: indicam o quanto uma partícula confia
 #na sua melhor posição(c1) ou na melhor posição ja obida por todas as particulas(c2)
 c1=2
 c2=2
```

```
#Inércia da particula, que controla a capacidade de exploração do algoritmo.
#Um valor alto facilita um comportamento mais global. Um valor baixo facilita
#um comportamento mais local.
w=1
#forma proposta para atualizar o parâmetro w a cada passo da iteração
passo=(w-0.4)/maxger

#Inicializando matriz auxiliar para armazenar os valores máximos da velocidade
vmax=(limites[2,]-limites[1,])*0.2
#Inicializando matriz velocidade
v<-matrix(0,nrow=npar,ncol=n)
#Inicializando AJ, que consiste em um vetor que armazena os valores da função
AJ<-0
#Inicializando variável auxiliar x
x<-0
#Contador, utilizado apenas no caso de querer gerar figuras sequenciais para
#posteriormente gerar imagens animadas:.gifs
#count<-1

#############################################################################
#######Implementação do processo iterativo do algoritmo#####################
#############################################################################


while (!criterio) #Enquanto critério for igual a zero o loop é processado.
  #criterio, foi definido como sendo zero em linhas anteriores.

{

  geracao=geracao+1 #atualiza o numero de gerações, ou iterações.
  w=w-passo        #atualiza o parâmetro de inércia

  for (i in 1:npar)#For que percorre o número total de particulas

  {

    for(j in 1:n)#For que percorre as coordenadas de cada particula
    {
     x[j]<-POP[i,j]#Armazena as coordenadas da partícula em um vetor auxiliar x
    }
    AJ[i]=funcao(x) #Aplica a função nas coordenadas de x e o valor retornado é
    #armazenado no vetor AJ

    if(AJ[i]<pbestAJ[i])#Caso o valor obtido para a função AJ seja menor que o
      #menor valor já encontrado por esta partícula processa-se o if
    {
      pbestAJ[i]=AJ[i]  #O melhor valor da particula passa a ser o valor encontrado
      pbestXX[i,]=POP[i,] #Guarda as coordenadas da partícula com o melhor valor
encontrado
```

```
    }
   }

   #Armazena o índice (ou a linha) do melhor valor de todos os melhores individuais
   #já encontrados por cada particula.
gbestAJindice<-which(pbestAJ==min(pbestAJ),arr.ind=TRUE)

#Armazena as coordenadas do melhor valor já encontrado
   gbestXX=pbestXX[gbestAJindice[1,2],]

   ######################################################################
   ####################ATUALIZAÇÃO#######################################

   for (i in 1:npar) # For que percorre o número de partículas
   {

    #Atualiza o vetor velocidade de acordo com formulação do algoritmo PSO
v[i,]=(w*vant[i])+(c1*runif(1)*(pbestXX[i,]-POP[i,]))+(c2*runif(1)*(gbestXX-POP[i,]))

for(j in 1:n) #For que percorre as coordenadas de cada particula
    {

     if(v[i,j]>vmax[j])#Se a velocidade atual for maior que a velocidade máxima
     {
      v[i,j]=vmax[j]#Velocidade atual recebe velocidade máxima
     }
    }


    POP[i,]=POP[i,]+v[i,]#Atualiza a posição da particula a partir da nova velocidade
    vant[i]=v[i]#Atualiza a velocidade da partícula para o passo posterior.

   }

     if (geracao==maxger)#Se o contador geracao atingir o número total de gerações
   {criterio=1}#Sera atribuído 1 a variável critério e interrompe-se o loop while

  }

  #Armazena em result uma lista que contem o vetor de parâmetros ótimos e
  # o respectivo valor da função objetivo.

  result<-list("Par_Otimo:",gbestXX,"FO:",funcao(gbestXX))
  return(result)#Imprime no prompt a melhor posição de todas após a finalização do
algoritmo

}
########################################
#FUNÇÃO SOMA############################
########################################
```

```
#INPUT-Argumento de entrada
#Qobs-vetor com vazões observadas
#Qcalc-vetor com vazões calculadas
#########################################################
#OUTPUT
#result-soma dos desvios ao quadrado (variância)
#########################################################

#Soma os desvios ao quadrado
#############################
soma = function(Qobs,Qcalc)
{
 result = 0
 for (t in 1:length(Qcalc))
 {
  result = result + (Qobs[t]-Qcalc[t])^2
}
 result
}

##########################################
#PARTE 2: CALIBRAÇÃO CONVENCIONAL
##########################################

#Definição os limtes do espaço de busca para calibração.
limites<-rbind(c(500,0.5),c(5000,6))
#A ordem dos parâmetros no vetor e indicada abaixo:
#limites<-rbind(c(SAT_min,pes_min,CREC_min,k_min,tuin_min,ebin_min),
#c(SAT_max,pes_max,CREC_max,k_max,tuin_max,ebin_max))

#Área da bacia em km²
Area=1500

#Chama a função PSO e armazena o resultado na variável:resultadoCalibracaoPSO
resultadoCalibracaoPSO<-PSO(FO1,limites)

#Mostra os valores calibrados no prompt do R
resultadoCalibracaoPSO

#vetor inicial de parâmetros para calibração via NelderMead
#Digitar os valores obtidos a partir da calibração PSO
parai<-c(1533.759573 ,3.107788)

#Aplica a função optim que contem o algoritmo de Nelder & Mead
# e armazena a saída da busca local na variável result1

result1<-optim(parai, FO1)#Armazena na variável result1 a saída da rotina
a <- proc.time()
X<-PSO(FO1,limites)
proc.time() - a
```

```
#mostra o vetor final de parâmetros calibrados
result1

#armazena o vetor de parãmetros ótimos na vairável paraf1
paraf1<-result1$par

#Armazena as vazões calculadas com base no vetor ótimo de parâmetros
Qcalc=smap(EPQ,paraf1,Area)

#Armazena valores de vazões observadas
Qobs=EPQ[,3]

#Plota as vazões observadas e as vazões calibradas pela calibração convencional
plot(Qcalc,type="l",col="red",lwd=2,xlab="mês",ylab="Vazão(m³/s)",main="Vazões
Observadas e Simuladas")
lines(Qobs,col="black",type="l",lty=3,lwd=2)
legend("topright",lty=c(1,3),lwd=c(2,2),bty="n",cex=0.8,col=c("red","black"),
legend=c("Vazões simuladas","Vazões observadas"))

#resíduosPSO
residuosPSO=Qcalc-Qobs

#desvioPadrão dos resíduos
sdResiduosPSO=sd(residuosPSO)


#########################################################################
#####PARTE 3: FUNÇÕES AUXILIARES E ALGORITMOS MCMC
#########################################################################

#####################################################################
##FUNÇÕES A PRIORI
#####################################################################

prior <- function(param){

  #passa os elementos do vetor param para as variáveis sat, pes, crec, k e sd
  sat = param[1]
  pes = param[2]
  sd=param[3]


#PRIORI SAT
  satprior = dunif(sat,min=500, max = 5000,log = T)


#PRIORI PES
  pesprior = dunif(pes,min=0.5,max=6,log = T)
```

```
  #PRIORI SD
  sdprior = dunif(sd,min=1,max=30,log = T)

  #retorna o ln do produto das prioris
  return(satprior+pesprior+sdprior)
}

#############################################################################
#FUNÇÃO DE LOG-VEROSSIMILHANÇA ASSUMINDO NORMALIDADE E
#HOMOCEDASTICIDADE
#############################################################################


##Função Verossimilhança
## Verossimilhança: Adotou-se o Ln da verossimilhança
###################################
likelihood <- function(param){

  #armazena os parâmetros de entrada
  sat = param[1]
  pes = param[2]
  sd=param[3]

  phi=1/sd^2 # Parâmetro de precisão

  Prec<-EPQ[,2] # Armazena a coluna P (precipitação), do data.frame EPQ no vetor #Prec
  N<-length(Prec) # Armazena o tamanho do vetor Prec na variável n

likelihood=0
  somadesv=0


  A=Area
  para<-c(sat,pes)
  EPQ<-EPQ
  Qcalc=smap(EPQ,para,Area)
Qobs<-EPQ[,3]
  somadesv=soma(Qobs,Qcalc)

likelihood=(N/2)*log(phi)-(phi/2)*somadesv
return(likelihood)

}
```

```
#####################################################################
#FUNÇÕES PROPOSTAS AUXILIARES PARA O ALGORITMO DE METROPOLIS-
HASTINGS
#####################################################################
##Funções propostas -ALGORITMO M
#########################################
proposalfunctionM <- function(param){

  sat = param[1]
  pes = param[2]
  sd = param[3]

satp=rnorm(1,sat,100)
  #plot(function(satx){dnorm(satx,mean=sat,sd=100)},xlim=c(sat-
500,sat+500),ylab="f(sat)",xlab="sat",lty=1)

  pesp=rnorm(1,pes,0.1)
  #plot(function(pesx){dnorm(pesx,mean=pes,sd=0.1)},xlim=c(pes-
0.5,pes+0.5),ylab="f(pes)",xlab="pes",lty=1)

limiteinferior=sd-sd/2
  limitesuperior=sd+sd/2
  sdp=runif(1, min=limiteinferior, max=limitesuperior)

  return(c(satp,pesp,sdp))

}

##Funções propostas -ALGORITMO MH
#########################################

proposalfunctionMH <- function(param){

   sat = param[1]
pes = param[2]
  sd=param[3]

  satp=rlnorm(1,log(sat),sd=0.1)

  #PROPOSTA PES
  CV=50
  EX=pes
  BETTA=CV^2/EX
  ALFFA=EX*BETTA
  pesp = rgamma(1,ALFFA,BETTA)
#plot(function(pes){dgamma(pes,ALFFA,BETTA)},xlim=c(pes-
#2,pes+2),ylab="f(pes)",xlab="pes",lty=1)

#PROPOSTA sd
  phi=1/sd^2
```

```
  CV=45
  EX=phi
  BETTA=CV^2/EX
  ALFFA=EX*BETTA

phip=rgamma(1,ALFFA,BETTA)
  sdp=sqrt(1/phip)


  #plot(function(sd){dgamma(sd,ALFFA,BETTA)},xlim=c(1/(sd-
30)^2,1/(sd+30)^2),ylab="f(sd)",xlab="sd",lty=1)

  return(c(satp,pesp,sdp))

}

#propostas aplicadas nos valores correntes com os parâmetros nos valores

proposal1 <- function(param){

  sati=param[1]
  pesi=param[2]
  sdi=param[3]

  satp=param[4]
  pesp=param[5]
  sdp=param[6]


  #Proposta do SAT nos valores correntes com base nos valores propostos
satproposal=dlnorm(sati,mean=log(satp),sd=0.1,log=T)

#Proposta do PES nos valores correntes com base nos valores propostos
  #CVi=8
  CVi=50
  EXi=pesi
  BETTAi=CVi^2/EXi
ALFFAi=EXi*BETTAi

  #CVp=8
  CVp=50
  EXp=pesp
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp

  pesproposal=dgamma(pesi,ALFFAp,BETTAp,log=T)
```

```
#Proposta do SD nos valores correntes com base nos valores propostos

phii=1/sdi^2
  #CVi=10
  CVi=45
  EXi=phii
  BETTAi=CVi^2/EXi
  ALFFAi=EXi*BETTAi

   phip=1/sdp^2
  #CVp=10
  CVp=45
  EXp=phip
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp

  phiproposal=dgamma(phii,ALFFAp,BETTAp,log=T)

sum=satproposal+pesproposal+phiproposal

  return(sum)

}

#propostas aplicadas nos valores propostos com os parâmetros nos valores
#correntes
proposal2 <- function(param){

  sati=param[1]
  pesi=param[2]
  sdi=param[3]

  satp=param[4]
  pesp=param[5]
  sdp=param[6]

  #Proposta do SAT nos valores correntes com base nos valores propostos
satproposal=dlnorm(satp,mean=log(sati),sd=0.1,log=T)

#Proposta do PES nos valores correntes com base nos valores propostos
  #CVi=8
  CVi=50
  EXi=pesi
  BETTAi=CVi^2/EXi
ALFFAi=EXi*BETTAi

  #CVp=8
  CVp=50
  EXp=pesp
  BETTAp=CVp^2/EXp
```

```
  ALFFAp=EXp*BETTAp

pesproposal=dgamma(pesp,ALFFAi,BETTAi,log=T)

  #Proposta do SD nos valores correntes com base nos valores propostos

phii=1/sdi^2
  #CVi=10
  CVi=45
  EXi=phii
  BETTAi=CVi^2/EXi
  ALFFAi=EXi*BETTAi

  phip=1/sdp^2
  #CVp=10
  CVp=45
  EXp=phip
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp

  phiproposal=dgamma(phip,ALFFAi,BETTAi,log=T)

sum=satproposal+pesproposal+phiproposal

  return(sum)

}

##############################
#FUNÇÃO POSTERIOR
##############################

##Função Posterior
##Ln do produto da verossimilhança pela priori

posterior <- function(param){
  return (likelihood(param) + prior(param))
}

#########################################################################
##Algoritmo Metropolis - M
#########################################################################


run_Metropolis <- function(startvalue, iterations){


#cria a matriz que armazenará os valores da cadeia para os 3 parâmetros
  #com a dimensão: nº linhas=Número de iterações +1; nº colunas=nº de parâmetros (3)
chain = array(dim = c(iterations+1,3))
```

```
#inicializa a primeira linha da matriz com o vetor de valores iniciais para os #parâmetros
  chain[1,] = startvalue

  #inicializa as variáveis que armazenarão o número de aceitação e rejeição dos
  #passos nas cadeias
  naceito=0
  nrejeitado=0

  #Inicializa o avanço ao longo dos passos da cadeia até o número total de iterações
  for (i in 1:iterations){

    #Chama a função proposalfuncion que retorna o vetor de parâmetros propostos que é
    #armazenado na variável proposal
    proposal= proposalfunctionM(chain[i,])

    #calcula a razão de aceitação de Hastings para o algoritmo M
    razao=posterior(proposal)-posterior(chain[i,])

    #condicional que faz um tratamento de erro, ou seja, caso algum valor não
    #numérico surja na razão de aceitação se mantém no passo posterior da cadeia
    #os valores do passo atual
    if (exp(razao) == "NaN"){

      chain[i+1,]=chain[i,]
      nrejeitado=nrejeitado+1
    }

    else

      #Condicional que permite a aceitação do valor proposto
      #sorteia-se um número de 0 a 1 e compara-se o valor
      #sorteado com o exponencial da razão de aceitação (ou a unidade, caso
      #o valor do exponencial da razão de aceitação seja maior). Se o valor sorteado
      #for menor, aceita-se o valor proposto

if(runif(1,0,1)<min(1,exp(razao))){

#armazena-se no passo seguinte da cadeia o valor proposto
        chain[i+1,]=proposal

        #contabiliza-se o número de aceitação
        naceito=naceito+1
      }

    else{

      #caso contrário rejeita-se o valor proposto e armazena-se no passo seguinte
      #da cadeia o valor do passo corrente
      chain[i+1,] = chain[i,]
```

```
    #contabiliza-se o número de rejeição
    nrejeitado=nrejeitado+1

   }
  }

   #retorna uma lista contendo todos os valores da cadeia (chain), o número de valores
  #aceitos (naceito) e o número de valores rejeitados (nrejeitado)
  lista=list(chain,naceito,nrejeitado)

  return(lista)
}

################################################
## ALGORITMO METROPOLIS HASTINGS - MH
################################################

#Função do algoritmo Metropolis Hastings
#Argumentos de entrada:
#vetor de parâmetros iniciais- startvalue=(SAT,PES,CREC,K,Desvio Padrão)
#Número de iterações- iterations=(Número de Iterações)
run_MetropolisHastings <- function(startvalue, iterations){

#cria a matriz que armazenará os valores da cadeia para os 3 parâmetros
  #com a dimensão: nº linhas=Número de iterações +1; nº colunas=nº de parâmetros (3)
chain = array(dim = c(iterations+1,3))

#inicializa a primeira linha da matriz com o vetor de valores iniciais para os #parâmetros
  chain[1,] = startvalue

  #inicializa as variáveis que armazenarão o número de aceitação e rejeição dos
  #passos nas cadeias
  naceito=0
  nrejeitado=0

  #Inicializa o avanço ao longo dos passos da cadeia até o número total de iterações
  for (i in 1:iterations){

    #Chama a função proposalfuncion que retorna o vetor de parâmetros propostos que é
    #armazenado na variável proposal
    proposal= proposalfunctionMH(chain[i,])

    #armazena na variável valores o vetor com os valores dos parâmetros no passo
    #corrente na cadeia e o vetor de parâmetros propostos
    valores=c(chain[i,],proposal)

    #calcula a razão de aceitação de Hastings fazendo uso das funções propostas
    #auxiliares
    razao=posterior(proposal)+proposal1(valores)-posterior(chain[i,])-proposal1(valores)
```

```
    #condicional que faz um tratamento de erro, ou seja, caso algum valor não
    #numérico surja na razão de aceitação se mantém no passo posterior da cadeia
    #os valores do passo atual
    if (exp(razao) == "NaN"){

      chain[i+1,]=chain[i,]
      nrejeitado=nrejeitado+1
    }

    else

      #Condicional que permite a aceitação do valor proposto
      #sorteia-se um número de 0 a 1 e compara-se o valor
      #sorteado com o exponencial da razão de aceitação (ou a unidade, caso
      #o valor do exponencial da razão de aceitação seja maior). Se o valor sorteado
      #for menor, aceita-se o valor proposto
if (runif(1,0,1)<min(1,exp(razao))){

#armazena-se no passo seguinte da cadeia o valor proposto
       chain[i+1,]=proposal
       #contabiliza-se o número de aceitação
       naceito=naceito+1

      }
    else{

      #caso contrário rejeita-se o valor proposto e armazena-se no passo seguinte
      #da cadeia o valor do passo corrente
      chain[i+1,] = chain[i,]
      #contabiliza-se o número de rejeição
      nrejeitado=nrejeitado+1
    }

  }

  #retorna uma lista contendo todos os valores da cadeia (chain), o número de valores
  #aceitos (naceito) e o número de valores rejeitados (nrejeitado)
  lista=list(chain,naceito,nrejeitado)

  return(lista)
}
```

```
#########################################################################
#ALGORITMO ADAPTIVE METROPOLIS - AM
#########################################################################

run_AdaptiveMetropolis_MCMC <- function(startvalue, iterations){

#Cria a matriz que irá armazenar os valores das cadeias para os 3 parâmetros
#SAT, PES,SD
  chain = array(dim = c(iterations+1,3))

#Inicializa os elementos da primeira linha da matriz
chain[1,] = startvalue

  #Variância SAT
  varSat=30^2

#Variância PES
  varPes= 0.08^2

  #Variância SD
  varSd=15^2

  #Cria a matriz de variância/covariância C
  C=matrix(0,nrow=3,ncol=3)


  #Atribui as variâncias para a diagonal da matriz C
diag(C)=c(varSat,varPes,varSd)

#Matriz Identidade
  Id=diag(x=1,nrow=3,ncol=3)

  #Parâmetros do algoritmo AM
  sd=(2.4)^2/5
  E=0.001

  #Variáveis que armazenam números de aceitação e rejeição na cadeia
  naceito=0
  nrejeitado=0

  #Matriz que armazena os valores propostos da cadeia
p=matrix(0,nrow=iterations,ncol=3)

# Valores iniciais para a matriz C, conforme algoritmo Metropolis
  # para os 600 passos iniciais
  for(i in 1:600)
  {


    #proposta a partir de uma distribuição normal multivariada
```

```
proposal=mvrnorm(1,chain[i,],C)

#armazena os valores propostos na matriz p
   p[i,]=proposal

   #calcula a razão de aceitação
   razao=posterior(proposal)-posterior(chain[i,])

   if (exp(razao) == "NaN"){

     chain[i+1,]=chain[i,]
     nrejeitado=nrejeitado+1
}

   else

     if(runif(1,0,1)<min(1,exp(razao))){

chain[i+1,]=proposal
       naceito=naceito+1
}

   else{

     chain[i+1,] = chain[i,]
nrejeitado=nrejeitado+1

   }

  }

  #Passos seguintes após a iteração inicial calculando a matriz de variância covariância
  #com base na matriz C do passo anterior

for (i in 601:iterations){

   chainMed1=c(mean(chain[1:i,1]),mean(chain[1:i,2]),mean(chain[1:i,3]))
   chainMed2=c(mean(chain[1:(i-1),1]),mean(chain[1:(i-1),2]),mean(chain[1:(i-1),3]))
   C=((i-1)/i)*C+(sd/i)*(i*chainMed2%*%t(chainMed2)-
(i+1)*chainMed1%*%t(chainMed1)+chain[i,]%*%t(chain[i,])+E*Id)

#valores propostos a partir de uma distribuição normal multivariada
   #com base na matriz C
   proposal=mvrnorm(1,chain[i,],C)
   #Armazena o valor proposto na matriz p
   p[i,]=proposal

   #Calcula a razão de aceitação
   razao=posterior(proposal)-posterior(chain[i,])
```

```
    if (exp(razao) == "NaN"){

     chain[i+1,]=chain[i,]
     nrejeitado=nrejeitado+1
}

   else

     if(runif(1,0,1)<min(1,exp(razao))){

chain[i+1,]=proposal
       naceito=naceito+1
}

   else{

     chain[i+1,] = chain[i,]
nrejeitado=nrejeitado+1

   }
  }

  #Retorna uma lista com a cadeia final, número de aceitação, número de rejeição
  #e a matriz de valores propostos

  lista=list(chain,naceito,nrejeitado,p)

  return(lista)
}

#####################################################################
##PARTE 4: RODANDO OS ALGORITMOS MCMC
#####################################################################

#####################################################################
######## Cadeia 1 - Algoritmo M
#####################################################################

# Valores iniciais para o primeiro conjunto de cadeias
startvalue <- c(1534,3.1,4.4)
iterations=150000

#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoM1<-run_Metropolis(startvalue,iterations)
proc.time() - a

#Número de valores aceitos
```

```
resultadoM1[[2]]
#Número de valores rejetiados
resultadoM1[[3]]

#Salva na variável chainAM1 a cadeia final
chainM1=resultadoM1[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainM1, "chainM1.txt",sep="\t",row.names=F )

#####################################################################
######## Cadeia 2 - Algoritmo M
#####################################################################

# Valores iniciais para o primeiro conjunto de cadeias
startvalue <- c(1000,2,6)
iterations=150000

#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável #e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoM2<-run_Metropolis(startvalue,iterations)
proc.time() - a

#Número de valores aceitos
resultadoM2[[2]]
#Número de valores rejetiados
resultadoM2[[3]]

#Salva na variável chainAM1 a cadeia final
chainM2=resultadoM2[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainM2, "chainM2.txt",sep="\t",row.names=F )


#####################################################################
######## Cadeia 1 - Algoritmo MH
#####################################################################

# Valores iniciais para o primeiro conjunto de cadeias
startvalue <- c(1534,3.1,4.4)
iterations=150000

#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável #e
#contabiliza o tempo de processamento
a <- proc.time()
resultadoMH1<-run_MetropolisHastings(startvalue,iterations)
proc.time() - a
```

```
#Número de valores aceitos
resultadoMH1[[2]]
#Número de valores rejetiados
resultadoMH1[[3]]

#Salva na variável chainAM1 a cadeia final
chainMH1=resultadoMH1[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainMH1, "chainMH1.txt",sep="\t",row.names=F )


#####################################################################
######## Cadeia 2 - Algoritmo MH
#####################################################################

# Valores iniciais para o primeiro conjunto de cadeias
startvalue <- c(1000,2,6)
iterations=150000

#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável #e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoMH2<-run_MetropolisHastings(startvalue,iterations)
proc.time() - a

#Número de valores aceitos
resultadoMH2[[2]]
#Número de valores rejetiados
resultadoMH2[[3]]

#Salva na variável chainAM1 a cadeia final
chainMH2=resultadoMH2[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainMH2, "chainMH2.txt",sep="\t",row.names=F )

#####################################################################
######## Cadeia 1 - Algoritmo AM
#####################################################################

# Valores iniciais para o primeiro conjunto de cadeias
startvalue <- c(1534,3.1,4.4)
iterations=150000
#Chama o pacote MASS que habilita a função normal multivariada, usada no algoritmo
#AM
library(MASS)
```

```
#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável #e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoAM1<-run_AdaptiveMetropolis_MCMC(startvalue,iterations)
proc.time() - a


#Número de valores aceitos
resultadoAM1[[2]]
#Número de valores rejetiados
resultadoAM1[[3]]

#Salva na variável chainAM1 a cadeia final
chainAM1=resultadoAM1[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainAM1, "chainAM1.txt",sep="\t",row.names=F )

#################################################################
######## Cadeia 2 - Algoritmo AM
#################################################################

# Valores iniciais para o primeiro conjunto de cadeias
startvalue <- c(1000,2,6)
iterations=150000

#Chama o pacote MASS que habilita a função normal multivariada, usada no algoritmo
AM
library(MASS)

#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável #e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoAM2<-run_AdaptiveMetropolis_MCMC(startvalue,iterations)
proc.time() - a

#Número de valores aceitos
resultadoAM2[[2]]
#Número de valores rejetiados
resultadoAM2[[3]]

#Salva na variável chainAM1 a cadeia final
chainAM2=resultadoAM2[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainAM2, "chainAM2.txt",sep="\t",row.names=F )
```

```
#####################################################################
##PARTE 5: RESULTADOS GERAIS
#####################################################################

#valores para descartar
burnIn <- 60000

#Função que define o intervalo de credibilidade de 95%
HDIofMCMC = function( sampleVec , credMass=0.95 ) {
  sortedPts = sort( sampleVec )
  ciIdxInc = floor( credMass * length( sortedPts ) )
  nCIs = length( sortedPts ) - ciIdxInc
  ciWidth = rep( 0 , nCIs )
  for ( i in 1:nCIs ) {
    ciWidth[ i ] = sortedPts[ i + ciIdxInc ] - sortedPts[ i ]
  }
  HDImin = sortedPts[ which.min( ciWidth ) ]
  HDImax = sortedPts[ which.min( ciWidth ) + ciIdxInc ]
  HDIlim = c( HDImin , HDImax )
  return( HDIlim )
}

#Média das Cadeias

mean(chainM1[-(1:burnIn),1])
mean(chainM2[-(1:burnIn),1])
mean(chainMH1[-(1:burnIn),1])
mean(chainMH2[-(1:burnIn),1])
mean(chainAM1[-(1:burnIn),1])
mean(chainAM2[-(1:burnIn),1])

mean(chainM1[-(1:burnIn),2])
mean(chainM2[-(1:burnIn),2])
mean(chainMH1[-(1:burnIn),2])
mean(chainMH2[-(1:burnIn),2])
mean(chainAM1[-(1:burnIn),2])
mean(chainAM2[-(1:burnIn),2])

mean(chainM1[-(1:burnIn),3])
mean(chainM2[-(1:burnIn),3])
mean(chainMH1[-(1:burnIn),3])
mean(chainMH2[-(1:burnIn),3])
mean(chainAM1[-(1:burnIn),3])
mean(chainAM2[-(1:burnIn),3])

#Intervalos de Credibilidade de 95%

#M-Cadeia1
intconf=0
intconf<-HDIofMCMC(chainM1[-(1:burnIn),1])
```

```
intconf

intconf=0
intconf<-HDIofMCMC(chainM1[-(1:burnIn),2])
intconf

intconf=0
intconf<-HDIofMCMC(chainM1[-(1:burnIn),3])
intconf


#M-Cadeia2
intconf=0
intconf<-HDIofMCMC(chainM2[-(1:burnIn),1])
intconf

intconf=0
intconf<-HDIofMCMC(chainM2[-(1:burnIn),2])
intconf

intconf=0
intconf<-HDIofMCMC(chainM2[-(1:burnIn),3])
intconf


#MH-Cadeia1
intconf=0
intconf<-HDIofMCMC(chainMH1[-(1:burnIn),1])
intconf

intconf=0
intconf<-HDIofMCMC(chainMH1[-(1:burnIn),2])
intconf

intconf=0
intconf<-HDIofMCMC(chainMH1[-(1:burnIn),3])
intconf

#MH-Cadeia2
intconf=0
intconf<-HDIofMCMC(chainMH2[-(1:burnIn),1])
intconf


intconf=0
intconf<-HDIofMCMC(chainMH2[-(1:burnIn),2])
intconf

intconf=0
intconf<-HDIofMCMC(chainMH2[-(1:burnIn),3])
```

```
intconf

#AM-Cadeia1
intconf=0
intconf<-HDIofMCMC(chainAM1[-(1:burnIn),1])
intconf

intconf=0
intconf<-HDIofMCMC(chainAM1[-(1:burnIn),2])
intconf

intconf=0
intconf<-HDIofMCMC(chainAM1[-(1:burnIn),3])
intconf

#AM-Cadeia2
intconf=0
intconf<-HDIofMCMC(chainAM2[-(1:burnIn),1])
intconf

intconf=0
intconf<-HDIofMCMC(chainAM2[-(1:burnIn),2])
intconf

intconf=0
intconf<-HDIofMCMC(chainAM2[-(1:burnIn),3])
intconf

#####################################################################
## Gráficos Cadeias e Posteriores: ##
#####################################################################


#Olhando todos os parametros
par(mfrow = c(2,2))


plot(chainAM1[,1],type="l",xlab="Iteração",ylab="SAT",main="Cadeia dos Valores de
SAT",col="black",ylim=c(1200,2900))
lines(chainAM2[,1],col="red")
lines(chainMH1[,1],col="blue")
lines(chainMH2[,1],col="green")
lines(chainM1[,1],col="yellow")
lines(chainM2[,1],col="purple")
abline(h = paraf1[1],lwd=2,lty=2,col="blue")

legend("topleft",lty=c(1,1,1,1),lwd=c(2,2,2,2),bty="n",cex=1.5,col=c("black","red","blue",
"green"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2"))
```

```
legend("topright",lty=c(1,1,2),lwd=c(2,2,2),bty="n",cex=1.5,col=c("yellow","purple","blue
"),
legend=c("M-Cadeia 1","M-Cadeia 2","Calibração PSO"))


plot(chainAM1[,2],type="l",xlab="Iteração",ylab="PES",main="Cadeia dos Valores de
PES",col="black",ylim=c(2.4,4))
lines(chainAM2[,2],col="red")
lines(chainMH1[,2],col="blue")
lines(chainMH2[,2],col="green")
lines(chainM1[,2],col="yellow")
lines(chainM2[,2],col="purple")
abline(h = paraf1[2],lwd=2,lty=2,col="blue")


plot(density(chainM1[-
(1:burnIn),1],bw=10),type="l",xlab="SAT",ylab="",main="SAT",col="black",lwd=2,ylim
=c(0,0.005))
lines(density(chainAM2[-(1:burnIn),1],bw=10),col="red",lwd=2)
lines(density(chainMH1[-(1:burnIn),1],bw=10),col="blue",lwd=2)
lines(density(chainMH2[-(1:burnIn),1],bw=10),col="green",lwd=2)
lines(density(chainM1[-(1:burnIn),1],bw=10),col="yellow",lwd=2)
lines(density(chainM2[-(1:burnIn),1],bw=10),col="purple",lwd=2)
abline(v = paraf1[1],lwd=2,lty=2,col="blue")


plot(density(chainM1[-
(1:burnIn),2],bw=0.03),type="l",xlab="PES",ylab="",main="PES",col="black",lwd=2,yli
m=c(0,3))
lines(density(chainAM2[-(1:burnIn),2],bw=0.03),col="red",lwd=2)
lines(density(chainMH1[-(1:burnIn),2],bw=0.03),col="blue",lwd=2)
lines(density(chainMH2[-(1:burnIn),2],bw=0.03),col="green",lwd=2)
lines(density(chainM1[-(1:burnIn),2],bw=0.03),col="yellow",lwd=2)
lines(density(chainM2[-(1:burnIn),2],bw=0.03),col="purple",lwd=2)
abline(v = paraf1[2],lwd=2,lty=2,col="blue")


#Olhando todos os parametros
par(mfrow = c(2,1))


plot(chainAM1[,3],type="l",xlab="Iteração",ylab=expression(paste(sigma[])),main=expres
sion(paste("Cadeia dos valores ",sigma[])),col="black",ylim=c(3.5,10))
lines(chainAM2[,3],col="red")
lines(chainMH1[,3],col="blue")
lines(chainMH2[,3],col="green")
lines(chainM1[,3],col="yellow")
lines(chainM2[,3],col="purple")
abline(h = sdResiduosPSO,lwd=2,lty=2,col="blue")
```

```
legend(x=500,y=9.5,lty=c(1,1),lwd=c(2,2),bty="n",cex=1.5,col=c("black","red"),
legend=c("AM-Cadeia 1","AM-Cadeia 2"))

legend(x=50000,y=9.5,lty=c(1,1),lwd=c(2,2),bty="n",cex=1.5,col=c("blue","green"),
legend=c("MH-Cadeia 1","MH-Cadeia 2"))

legend(x=100000,y=10.4,lty=c(1,1,2),lwd=c(2,2,2),bty="n",cex=1.5,col=c("yellow","purpl
e","blue"),
legend=c("M-Cadeia 1","M-Cadeia 2","Calibração PSO"))


plot(density(chainM1[-
(1:burnIn),3],bw=0.08),type="l",xlab=expression(paste(sigma[])),main=expression(paste(,
sigma[])),col="black",lwd=2,ylab="")
lines(density(chainAM2[-(1:burnIn),3],bw=0.08),col="red",lwd=2)
lines(density(chainMH1[-(1:burnIn),3],bw=0.08),col="blue",lwd=2)
lines(density(chainMH2[-(1:burnIn),3],bw=0.08),col="green",lwd=2)
lines(density(chainM1[-(1:burnIn),3],bw=0.08),col="yellow",lwd=2)
lines(density(chainM2[-(1:burnIn),3],bw=0.08),col="purple",lwd=2)
abline(v = sdResiduosPSO,lwd=2,lty=2,col="blue")


##########################################################################
######Avaliação da convergência
##########################################################################

#chama o pacote Coda que possui a função para o diagnóstico de Gelman
library("coda")


################################################
######ALGORITMO M#################################
################################################

#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as
matrizes com as cadeias
mh.draws1M <- mcmc(chainM1[-(1:burnIn),1])
mh.draws2M <- mcmc(chainM2[-(1:burnIn),1])

#cria um alista com as cadeias geradas
mh.listM <- mcmc.list(list(mh.draws1M , mh.draws2M))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R
para as cadeias
M.SAT.Diag=gelman.diag(mh.listM,confidence=0.95)
M.SAT.Diag$psrf

#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listM,main="Teste de Convergência - AM (SAT)",ylab="Fator
R",xlab="Iteração")
```

```
#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as matrizes com as cadeias
mh.draws1M <- mcmc(chainM1[-(1:burnIn),2])
mh.draws2M <- mcmc(chainM2[-(1:burnIn),2])

#cria um alista com as cadeias geradas
mh.listM <- mcmc.list(list(mh.draws1M , mh.draws2M))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R para as cadeias
M.PES.Diag=gelman.diag(mh.listM,confidence=0.95)
M.PES.Diag$psrf

#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listM,main="Teste de Convergência - AM (SAT)",ylab="Fator R",xlab="Iteração")



#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as matrizes com as cadeias
mh.draws1M <- mcmc(chainM1[-(1:burnIn),3])
mh.draws2M <- mcmc(chainM2[-(1:burnIn),3])

#cria um alista com as cadeias geradas
mh.listM <- mcmc.list(list(mh.draws1M , mh.draws2M))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R para as cadeias
M.SIGMA.Diag=gelman.diag(mh.listM,confidence=0.95)
M.SIGMA.Diag$psrf

#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listM,main="Teste de Convergência - AM (SAT)",ylab="Fator R",xlab="Iteração")


##################################################
######ALGORITMO MH#################################
##################################################

#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as matrizes com as cadeias
mh.draws1MH <- mcmc(chainMH1[-(1:burnIn),1])
mh.draws2MH <- mcmc(chainMH2[-(1:burnIn),1])

#cria um alista com as cadeias geradas
mh.listMH <- mcmc.list(list(mh.draws1MH , mh.draws2MH))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R para as cadeias
```

```
M.SAT.Diag=gelman.diag(mh.listMH,confidence=0.95)
M.SAT.Diag$psrf

#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listMH,main="Teste de Convergência - AM (SAT)",ylab="Fator R",xlab="Iteração")



#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as matrizes com as cadeias
mh.draws1MH <- mcmc(chainMH1[-(1:burnIn),2])
mh.draws2MH <- mcmc(chainMH2[-(1:burnIn),2])

#cria um alista com as cadeias geradas
mh.listMH <- mcmc.list(list(mh.draws1MH , mh.draws2MH))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R para as cadeias
M.PES.Diag=gelman.diag(mh.listMH,confidence=0.95)
M.PES.Diag$psrf

#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listMH,main="Teste de Convergência - AM (SAT)",ylab="Fator R",xlab="Iteração")



#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as matrizes com as cadeias
mh.draws1MH <- mcmc(chainMH1[-(1:burnIn),3])
mh.draws2MH <- mcmc(chainMH2[-(1:burnIn),3])

#cria um alista com as cadeias geradas
mh.listMH <- mcmc.list(list(mh.draws1MH , mh.draws2MH))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R para as cadeias
M.SIGMA.Diag=gelman.diag(mh.listMH,confidence=0.95)
M.SIGMA.Diag$psrf

#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listM,main="Teste de Convergência - AM (SAT)",ylab="Fator R",xlab="Iteração")
```

```
################################################
######ALGORITMO AM################################
################################################

#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as matrizes com as cadeias
mh.draws1AM <- mcmc(chainAM1[-(1:burnIn),1])
mh.draws2AM <- mcmc(chainAM2[-(1:burnIn),1])

#cria um alista com as cadeias geradas
mh.listAM <- mcmc.list(list(mh.draws1AM , mh.draws2AM))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R para as cadeias
M.SAT.Diag=gelman.diag(mh.listAM,confidence=0.95)
M.SAT.Diag$psrf

#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listAM,main="Teste de Convergência - AM (SAT)",ylab="Fator R",xlab="Iteração")

#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as matrizes com as cadeias
mh.draws1AM <- mcmc(chainAM1[-(1:burnIn),2])
mh.draws2AM <- mcmc(chainAM2[-(1:burnIn),2])

#cria um alista com as cadeias geradas
mh.listAM <- mcmc.list(list(mh.draws1AM , mh.draws2AM))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R para as cadeias
M.PES.Diag=gelman.diag(mh.listAM,confidence=0.95)
M.PES.Diag$psrf

#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listAM,main="Teste de Convergência - AM (SAT)",ylab="Fator R",xlab="Iteração")

#salva as cadeias como variáveis do tipo mcmc, com o emprego da função mcmc sobre as matrizes com as cadeias
mh.draws1AM <- mcmc(chainAM1[-(1:burnIn),3])
mh.draws2AM <- mcmc(chainAM2[-(1:burnIn),3])

#cria um alista com as cadeias geradas
mh.listAM <- mcmc.list(list(mh.draws1AM , mh.draws2AM))
#Aplica a função gelman.diag na lista com as cadeias geradas e verifica o valor do fator R para as cadeias
M.SIGMA.Diag=gelman.diag(mh.listAM,confidence=0.95)
M.SIGMA.Diag$psrf
```

```
#Plota o valor do fator R em função do número de elementos da cadeia
gelman.plot(mh.listAM,main="Teste de Convergência - AM (SAT)",ylab="Fator R",xlab="Iteração")

medias.SAT.M1<-0

for(i in 1:nrow(chainM1))

{
   medias.SAT.M1[i]=mean(chainM1[1:i,1])

}

medias.SAT.M2<-0
for(i in 1:nrow(chainM1))

{
 medias.SAT.M2[i]=mean(chainM2[1:i,1])

}

medias.SAT.MH1<-0

for(i in 1:nrow(chainMH1))

{
 medias.SAT.MH1[i]=mean(chainMH1[1:i,1])

}

medias.SAT.MH2<-0

for(i in 1:nrow(chainMH2))

{
 medias.SAT.MH2[i]=mean(chainMH2[1:i,1])

}

medias.SAT.AM1<-0

for(i in 1:nrow(chainAM1))

{
 medias.SAT.AM1[i]=mean(chainAM1[1:i,1])
}
medias.SAT.AM2<-0
```

```
for(i in 1:nrow(chainAM2))

{
  medias.SAT.AM2[i]=mean(chainAM2[1:i,1])
}

#Cria um arquivo txt  para a cadeia gerada
write.table( medias.SAT.M1, "medias.SAT.M1.txt",sep="\t",row.names=F )
write.table( medias.SAT.M2, "medias.SAT.M2.txt",sep="\t",row.names=F )

write.table( medias.SAT.MH1, "medias.SAT.MH1.txt",sep="\t",row.names=F )
write.table( medias.SAT.MH2, "medias.SAT.MH2.txt",sep="\t",row.names=F )

write.table( medias.SAT.AM1, "medias.SAT.AM1.txt",sep="\t",row.names=F )
write.table( medias.SAT.AM1, "medias.SAT.AM1.txt",sep="\t",row.names=F )

#Olhando todos os parametros
par(mfrow = c(1,1))

plot(medias.SAT.AM1, main="Média SAT ao longo da
cadeia",xlab="iteração",ylab="SAT",lwd=2,type="l")

lines(medias.SAT.AM2,col="red")
lines(medias.SAT.MH1,col="blue")
lines(medias.SAT.MH2,col="green")
lines(medias.SAT.M1,col="yellow")
lines(medias.SAT.M2,col="purple")
abline(h = paraf1[1],lwd=2,lty=2,col="blue")

legend(x=500,y=1525,lty=c(1,1),lwd=c(2,2),bty="n",cex=1.5,col=c("black","red"),
legend=c("AM-Cadeia 1","AM-Cadeia 2"))

legend(x=50000,y=1525,lty=c(1,1),lwd=c(2,2),bty="n",cex=1.5,col=c("blue","green"),
legend=c("MH-Cadeia 1","MH-Cadeia 2"))

legend(x=100000,y=1525,lty=c(1,1,2),lwd=c(2,2,2),bty="n",cex=1.5,col=c("yellow","purp
le","blue"),
legend=c("M-Cadeia 1","M-Cadeia 2","Calibração PSO"))


plot(chainAM1[1:10000,1],type="l",xlab="Iteração",ylab="SAT",main="Cadeia dos
Valores de SAT",col="black",ylim=c(1200,2900))
lines(chainAM2[1:10000,1],col="red")
lines(chainMH1[1:10000,1],col="blue")
lines(chainMH2[1:10000,1],col="green")
lines(chainM1[1:10000,1],col="yellow")
lines(chainM2[1:10000,1],col="purple")
abline(h = paraf1[1],lwd=2,lty=2,col="blue")
```

```
legend("topleft",lty=c(1,1,1,1),lwd=c(2,2,2,2),bty="n",cex=1.5,col=c("black","red","blue",
"green"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2"))

legend("topright",lty=c(1,1,2),lwd=c(2,2,2),bty="n",cex=1.5,col=c("yellow","purple","blue
"),
legend=c("M-Cadeia 1","M-Cadeia 2","Calibração PSO"))

#Evolução da cadeia
plot(chainMH1[-(1:burnIn),1],chainMH1[-
(1:burnIn),2],type="p",frame=F,xlab="SAT",ylab="PES",cex=0.8,pch=15,xlim=c(1100,22
00),ylim=c(2.5,4))
library(MASS)
contornos<-kde2d(chainMH1[-(1:burnIn),1],chainMH1[-(1:burnIn),2])
contour(contornos,add=T,lwd=2,nlevels = 10,col=rev(heat.colors(10)))

############################################################
####VAZÕES E RESÍDUOS#######################################
############################################################

AD=Area
Qobs=EPQ[,3]

Vazoes3PAR=matrix(nrow=nrow(chainMH1),ncol=length(Qobs))

for(i in 1:nrow(chainMH1))
{
  Vazoes3PAR[i,]=smap(EPQ,chainMH1[i,1:2],AD)
}

write.table( Vazoes3PAR, "Vazoes3PAR.txt",sep="\t",row.names=F )#Cria um arquivo

residuos3PAR=matrix(0,nrow=nrow(chainMH1),ncol=length(Qobs))

for(i in 1:nrow(Vazoes3PAR))
{
  residuos3PAR[i,]=Vazoes3PAR[i,]-Qobs
 }

plot(colMeans(Vazoes3PAR),colMeans(residuos3PAR),main="Análise                    de
Resíduos",ylab="Média dos Resíduos",xlab="Média das Vazões Simuladas")

#RESÍDUOS - FUNÇÃOD E VEROSSIMILHANÇA ERROS GAUSSIANOS

standardResiduos3PAR=matrix(0,nrow=nrow(residuos3PAR),ncol=length(Qobs))

for(i in 1:nrow(standardResiduos3PAR))
{
  standardResiduos3PAR[i,]=residuos3PAR[i,]/chainMH1[i,3]
 }
```

```
plot(density(standardResiduos3PAR,bw=0.3),xlab="Resíduo
Padronizado",ylab="",main="Resíduo Padronizado",lty=1,lwd=2,col="red")
absc=seq(-4,4,by=0.1)
ord=dnorm(absc,0,1)
lines(absc,ord,lwd=2,type="l",col="black",lty=2)
legend("topright",lty=c(2,1),lwd=c(2,2),bty="n",cex=1.5,col=c("black","red"),
legend=c(expression("Distribuição Normal Padrão","Redíduos do Modelo")))

x=colMeans(Vazoes3PAR)
y=colMeans(standardResiduos3PAR)

x=Vazoes3PAR[1,]
y=standardResiduos3PAR[1,]

plot(x,y,main="Análise de Resíduos",ylab="Resíduo Padronizado",xlab="Vazão
Simulada")

my.acf<-acf(colMeans(standardResiduos3PAR))
ac<-(my.acf$acf)
plot(ac,main="")

resu<-lm(Qobs~colMeans(Vazoes3PAR))
par(mfrow=c(2,2))
plot(resu, which=c(1,2,3,4))
par(mfrow=c(1,1))
plot(resu, which=c(2),main="Normal Q-Q",xlab="Quantis")

##########################################
##Gráficos Incertezas nas Vazões Modeladas
##########################################

cadeiaFinal=chainMH1[-(1:burnIn),]
vazoesFinal=Vazoes3PAR[-(1:burnIn),]

intconf=0
intconf<-HDIofMCMC(chainMH1[-(1:burnIn),1])
intconf

lowSAT=intconf[1]
meanSAT=mean(chainMH1[-(1:burnIn),1])
upSAT=intconf[2]

intconf=0
intconf<-HDIofMCMC(chainMH1[-(1:burnIn),2])
intconf

lowPES=intconf[1]
meanPES=mean(chainMH1[-(1:burnIn),2])
upPES=intconf[2]
```

```
intconf=0
intconf<-HDIofMCMC(chainMH1[-(1:burnIn),3])
intconf

AD=1500
paraLow<-c(lowSAT,lowPES)

intconfMatrix=matrix(nrow=2,ncol=97)

vazoesErros=matrix(nrow=nrow(vazoesFinal),ncol=ncol(Vazoes3PAR))

for(i in 1:nrow(vazoesErros) )
{
  erro=rnorm(97,0,cadeiaFinal[i,3])
 vazoesErros[i,]=vazoesFinal[i,]+erro
 }

intconfMatrix=matrix(nrow=2,ncol=97)

for(i in 1:ncol(vazoesErros))
{
  intconfMatrix[,i]=HDIofMCMC(vazoesErros[,i])
}

Qlow<-intconfMatrix[1,]
Qmean<-colMeans(Vazoes3PAR)
Qup<-intconfMatrix[2,]

par(mfrow = c(1,1))

# Calculate range from 0 to max value of cars and trucks
g_range <- range(0, Qup)

Qobs=EPQ[,3]

plot(Qobs, type="p", col="blue", ylim=g_range,
    axes=FALSE, ann=FALSE,cex=1,pch=19)

newx=seq(1:length(Qmean))

polygon(c(rev(newx), newx), c(rev(Qlow), Qup), col = 'grey80', border = NA)

lines(Qmean, type="l",col="black",lty=1,lwd=2,cex=2)
lines(Qup, type="l",col="darkred",lty=3,lwd=2,cex=1)
lines(Qlow, type="l",col="darkred",lty=3,lwd=2,cex=1)
lines(Qobs, type="p",col="blue",lty=3,lwd=2,cex=0.8,pch=19)
```

```
# Make x axis using Mon-Fri labels
axis(1,at=seq(0,200,by=10),labels=F,lwd=1,lwd.ticks=1,cex=3)
par(tcl=-0.5)
axis(1,at=seq(0,200,by=10),lwd=0,lwd.ticks=2,cex=8)

# Make y axis with horizontal labels that display ticks at
# every 4 marks. 4*0:g_range[2] is equivalent to c(0,4,8,12).
axis(2, las=1, at=10*0:g_range[2],cex=2)

# Create box around plot
box()

# Graph trucks with red dashed line and square points
#lines(QSIMAGVALID, type="o", pch=22, lty=2, col="red")

# Graph trucks with red dashed line and square points
#lines(QSIMPSOVALID, type="o", pch=25, lty=5, col="darkgreen")

# Create a title with a red, bold/italic font
title(main="Vazões",cex.lab=4)

# Label the x and y axes with dark green text
title(xlab="mês",cex.lab=2)
title(ylab="Q(m³/s)",cex.lab=1.5)

legend(1, g_range[2], c("Média das disbriuições","Int. Cred. 95%"), cex=1.5,
col=c("black","darkred"), lty=c(1,3),lwd=c(2,2),box.col="white")

legend(1, 75,"       Vazões observadas", cex=1.5, col="blue",pch=19,box.col="white")
```

APÊNDICE B – CÓDIGO FONTE: CASO COM DADOS SINTÉTICOS

```
#####################################################################
#UNIVERSIDADE DE BRASÍLIA-UnB
#FACULDADE DE TECNOLOGIA
#DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL
#PÓS GRADUAÇÃO EM TECNOLOGIA AMBIENTAL E RECURSOS HÍDRICOS
#Data: 25/11/2015
#Autor: Cássio Guilherme Rampinelli
#Orientador: Dirceu Silveira Reis Jr.
#Coorientador:Carlos Henrique Ribeiro Lima
#email:cassiorampinelli@gmail.com

#Obs:Para solicitar os dados EPQ desta simulação enviar email para o referido autor
#####################################################################
#Código desenvolvido no âmbito das atividades do trabalho
#de dissertação de mestrado do referido autor.
#_____________________________________________________________#

#####################################################################
################Caso com dados Sintéticos#####################
#####################################################################

rm(list=ls(all=TRUE)) #Limpa a memória do R

#Carregando pacotes
library(MASS)#Pacote necessário para chamar a função normal multivariada

##########################################
#PARTE 1: DEFINIÇÃO DA FUNÇÃO SMAP- MENSAL
##########################################

#INPUT-Argumento de entrada

#EPQ-Matriz com os dados de Evaporação, Precipitação e Vazão Observada

#para-Vetor de parâmetros do modelo, no caso:sat;pes;crec;k

#########################################################

#OUTPUT

#Qcalc-Vetor com vazões simuladas

#########################################################

smap<-function(EPQ,para,Area) #Declaração da função smap

{

  ########################
#Inicializando variáveis#
  ########################
```

```
  E<-EPQ[,1] # Armazena a coluna E (evaporação), do data.frame EPQ no vetor E
Prec<-EPQ[,2] # Armazena a coluna P (precipitação), do data.frame EPQ no vetor Prec
  n<-length(Prec) # Armazena o tamanho do vetor Prec na variável n
sat=para[1] #Armazena o elemento 1 do vetor para na variável sat
pes=para[2] #Armazena o elemento 2 do vetor para na variável pes
crec=para[3] #Armazena o elemento 3 do vetor para na variável crec
#crec=0#para região do Ceará
#crecp=crec/100 #Transforma crec de percentual para centesimal
crecp=crec
  k=para[4] #Armazena o elemento 4 do vetor para na variável k
ke=0.5^(1/k) # Calcula a constante de recessão do escoamento superficial
#ke=0#para região do Ceará
#tuin=para[3] #Armazena o elemento 5 do vetor para na variável tuin
#tuinp=tuin/100;#Transforma tuin de percentual para centesimal

tuinp=0.6

#ebin=para[6]#Armazena o elemento 6 do vetor para na variável ebin
ebin=35

# As variáveis seguintes consistem em vetores e serão inicializados como nulos, pois
# o R requer uma definição a priore.
Rsol<-0
rsub<-0
  tu<-0
dsol<-0
Qcalc<-0
  es<-0
er<-0
eb<-0
rec<-0

#Inicialização dos primeiros elementos dos respectivos vetores
Rsol[1]=tuinp*sat
rsub[1]=ebin/(1-ke)/Area*2630
  tu[1]=tuinp

  ########################################
#Implementação do Algoritmo do Modelo Smap#
  ########################################

  i<-2# Variável i é inicializada como 2, pois os primeiros elementos dos
# vetores de interesse já foram inicializados nas linhas anteriores

while (i<=n+1) #Loop while que é processado até que o contador i atinga n+1

  {

# Calcula a variação de nível do reservatorio do solo
   dsol[i]=0.5*(Prec[i-1]-Prec[i-1]*(tu[i-1]^pes)-E[i-1]*tu[i-1]-Rsol[i-1]*crecp*(tu[i-1]^4)
)
```

```
#Calcula a taxa de umidade no passo i
   tu[i]=(Rsol[i-1]+dsol[i])/sat
#Calcula o escoamento de base
   es[i]=Prec[i-1]*tu[i]^pes
#Calcula a evaporação real
er[i]=E[i-1]*tu[i]
#Calcula a recarga do reservatorio subterrâneo
rec[i]=Rsol[i-1]*crecp*(tu[i]^4)
#Calcula o escoamento de base
eb[i]=rsub[i-1]*(1-ke)
#Atualiza o nível do reservatório subterrâneo
rsub[i]=rsub[i-1]-eb[i]+rec[i]
#Atualiza o nível do reservatorio do solo
Rsol[i]=Rsol[i-1]+Prec[i-1]-es[i]-er[i]-rec[i]
#Calcula vazão total simulada
Qcalc[i-1]=(es[i]+eb[i])*Area/2630
   i<-i+1#atualiza o contador

 }

return(Qcalc) # retorna a série simulada

}


##############################################
#PARTE 2: GERANDO AS VAZÕES SINTÉTICAS
##############################################

#Área de drenagem adotada
Area=39888

#Atribui diretório de trabalho onde os dados de entrada serão armazenados
#Configurar para pasta onde o arquivo EPQ será inserido
setwd("C:/Users/ADM/Desktop/MarkdownR")

#Dados de Evapotranspiração; Precipitação
EPQ<-read.table(file="EPQCaso1.txt",header=T)

#Vetor de Parâmetros do modelo SMAP pré definidos
param<-c(2200,2.5,0.3,3.5)

#Vazões Geradas
y<-smap(EPQ,param,Area)

#Visualização das Vazões Geradas
plot(y,type="l",main="Vazões Artificiais sem Erros",ylab="Vazões (m³/s)",xlab="MÊs")
```

```
#Inicializa o vetor que armazenará as vazões artificiais
VazoesArtificiais=0


# Insere os resíduos nas vazões sintéticas a partir de uma
#distribuição normal com média 0 e desvio padrão 300
for(i in 1:length(y) )
{

  erro=rnorm(1,0,300)
VazoesArtificiais[i]=y[i]+erro

}

#Gráfico das vazões artificiais com resíduos
plot(VazoesArtificiais,type="l",main="Vazões Artificiais com Erros",ylab="Vazões (m³/s)
",xlab="MÊs")


#Salva na terceira coluna da matriz EPQ as vazões sintéticas com os resíduos
EPQ[,3]<-VazoesArtificiais


#####################################################################
#PARTE 3: FUNÇÃO AUXILIAR PARA SOMA DOS QUADRADOS DOS RESÍDUOS
#####################################################################

########################################
#FUNÇÃO SOMA############################
########################################
#Soma os desvios ao quadrado
############################
soma =function(Qobs,Qcalc)
{
result =0
  for (t in 1:length(Qcalc))
  {
result =result +(Qobs[t]-Qcalc[t])^2
  }
result
}



#########################
##PARTE 4: FUNÇÕES PRIORIS
#########################

#Distribuição Prior
#################################
prior <-function(param){
```

```
#armazena parâmetros de entrada
sat =param[1]
pes =param[2]
crec =param[3]
  k=param[4]
sd=param[5]


#PRIORI SAT
satprior =dlnorm(sat,mean=8.2, sd =0.7,log = T)

#PRIORI PES
pesprior =dunif(pes,min=0.5,max=5,log = T)

#PRIORI CREC
crecprior =dbeta(crec,2,2,log = T)

#PRIORI K
  CV=2
  EX=3.5
  BETTA=CV^2/EX
  ALFFA=EX*BETTA
kprior =dgamma(k,ALFFA,BETTA,log = T)

#PRIORI SD
sdprior =dunif(sd,min=1,max=800,log = T)

return(satprior+pesprior+crecprior+kprior+sdprior)
}


#########################################################################
#########
##PARTE 5: FUNÇÃO DE LOG-VEROSSIMILHANÇA ASSUMINDO NORMALIDAD
E E HOMOCEDASTICIDADE
#########################################################################
#########


##Função Verossimilhança
## Verossimilhança: Adotou-se o Ln da verossimilhança
################################
likelihood<-function(param){

#armazena os parâmetros de entrada
sat =param[1]
pes =param[2]
crec =param[3]
  k=param[4]
sd=param[5]
```

```
phi=1/sd^2# Parâmetro de precisão

Prec<-EPQ[,2] # Armazena a coluna P (precipitação), do data.frame EPQ no vetor Prec
  N<-length(Prec) # Armazena o tamanho do vetor Prec na variável n

likelihood=0
somadesv=0


  A=Area
para<-c(sat,pes,crec,k)
  EPQ<-EPQ
Qcalc=smap(EPQ,para,Area)
Qobs<-EPQ[,3]
somadesv=soma(Qobs,Qcalc)

likelihood=(N/2)*log(phi)-(phi/2)*somadesv
return(likelihood)

}

############################
##PARTE 6: FUNÇÃO POSTERIOR
############################

##Função Posterior
##Ln do produto da verossimilhança pela priori

posterior <-function(param){
return (likelihood(param) +prior(param))
}


#############################################
##PARTE 7: ALGORITMO ADAPTIVE METROPOLIS - AM
#############################################

run_AdaptiveMetropolis_MCMC<-function(startvalue, iterations){

#Cria a matriz que irá armazenar os valores das cadeias para os 5 parâmetros
#SAT, PES, CREC, K, SD
chain =array(dim =c(iterations+1,5))

#Inicializa os elementos da primeira linha da matriz
chain[1,] =startvalue

#Variância SAT
varSat=30^2

#Variância PES
varPes=0.08^2
```

```
#VariânciaCrec
varCrec=0.08^2

#Variância K
varK=0.1^2

#Variância SD
varSd=15^2

#Cria a matriz de variância/covariância C
  C=matrix(0,nrow=5,ncol=5)


#Atribui as variâncias para a diagonal da matriz C
diag(C)=c(varSat,varPes,varCrec,varK,varSd)



#Matriz Identidade
  Id=diag(x=1,nrow=5,ncol=5)

#Parâmetros do algoritmo AM
sd=(2.4)^2/5
  E=0.001

#Variáveis que armazenam números de aceitação e rejeição na cadeia
naceito=0
nrejeitado=0

#Matriz que armazena os valores propostos da cadeia
  p=matrix(0,nrow=iterations,ncol=5)

# Valores iniciais para a matriz C, conforme algoritmo Metropolis
# para os 600 passos iniciais
  for(i in 1:600)
  {


#proposta a partir de uma distribuição normal multivariada
proposal=mvrnorm(1,chain[i,],C)

#armazena os valores propostos na matriz p
    p[i,]=proposal

#calcula a razão de aceitação
razao=posterior(proposal)-posterior(chain[i,])

if (exp(razao) == "NaN"){

chain[i+1,]=chain[i,]
nrejeitado=nrejeitado+1
```

```
   }

else

if(runif(1,0,1)<min(1,exp(razao))){

chain[i+1,]=proposal
naceito=naceito+1
     }

else{

chain[i+1,] =chain[i,]
nrejeitado=nrejeitado+1

   }


  }

#Passos seguintes após a iteração inicial calculando a matriz de variância covariância
#com base na matriz C do passo anterior

  for (i in 601:iterations){

   chainMed1=c(mean(chain[1:i,1]),mean(chain[1:i,2]),mean(chain[1:i,3]),mean(chain[1:
i,4]),mean(chain[1:i,5]))
   chainMed2=c(mean(chain[1:(i-1),1]),mean(chain[1:(i-1),2]),mean(chain[1:(i-1),3]),me
an(chain[1:(i-1),4]),mean(chain[1:(i-1),5]))
   C=((i-1)/i)*C+(sd/i)*(i*chainMed2%*%t(chainMed2)-(i+1)*chainMed1%*%t(chainMe
d1)+chain[i,]%*%t(chain[i,])+E*Id)

#valores propostos a partir de uma distribuição normal multivariada
#com base na matriz C
proposal=mvrnorm(1,chain[i,],C)
#Armazena o valor proposto na matriz p
   p[i,]=proposal

#Calcula a razão de aceitação
razao=posterior(proposal)-posterior(chain[i,])


if (exp(razao) == "NaN"){

chain[i+1,]=chain[i,]
nrejeitado=nrejeitado+1
   }

else

if(runif(1,0,1)<min(1,exp(razao))){
```

```
chain[i+1,]=proposal
naceito=naceito+1
     }

else{

chain[i+1,] =chain[i,]
nrejeitado=nrejeitado+1

   }
  }

#Retorna uma lista com a cadeia final, número de aceitação, número de rejeição
#e a matriz de valores propostos

  lista=list(chain,naceito,nrejeitado,p)

return(lista)
}



##################################
##PARTE 8: RODANDO O ALGORITMO
##################################

#Inicializa a primeira linha da cadeia
startvalue<-c(500,1.1,0.2,2,100)

#Definie o Número de iterações
nIter=50000
iterations=nIter

#Confirma a área da bacia
Area=39888

#inicializa a primeira linha da cadeia
chain=0

#Variável para armazenar o tempo de processamento
a <-proc.time()

#chama o algoritmo e armazena a saída na variável resultadoAM
resultadoAM<-run_AdaptiveMetropolis_MCMC(startvalue,iterations)

#Mostra o tempo de processamento após rodar o algoritmo
proc.time() -a

##usersystemelapsed
## 4484.1410.58 4580.72
```

```
#Armazena a matriz de valores propostos na variável propostas
propostas<-resultadoAM[[4]]

#Armazena as cadeias na variável chainAM
chainAM=resultadoAM[[1]]
chainAM=matrix(unlist(chainAM),ncol=5,byrow=F)

#Mostra o número de valores aceitos
resultadoAM[[2]]

## [1] 7005

#Mostra o número de valores rejeitados
resultadoAM[[3]]

## [1] 42995

##########################################################
##PARTE 9: Visualizando as cadeias-Resultados básicos
##########################################################

#valores para descartar
burnIn<-10000



#Função que define o intervalo de credibilidade de 95%
HDIofMCMC =function( sampleVec , credMass=0.95 ) {
sortedPts =sort( sampleVec )
ciIdxInc =floor( credMass *length( sortedPts ) )
nCIs =length( sortedPts ) -ciIdxInc
ciWidth =rep( 0 , nCIs )
  for ( i in 1:nCIs ) {
ciWidth[ i ] =sortedPts[ i +ciIdxInc ] -sortedPts[ i ]
  }
HDImin =sortedPts[ which.min( ciWidth ) ]
HDImax =sortedPts[ which.min( ciWidth ) +ciIdxInc ]
HDIlim =c( HDImin , HDImax )
return( HDIlim )
}

################
## Gráficos: ##
################



#Olhando todos os parametros
par(mfrow =c(2,3))

intconf=0
intconf<-HDIofMCMC(chainAM[-(1:burnIn),1])
```

```
#hist(chain[-(1:burnIn),1],ylab="Frequência na cadeia final",xlab="Probabilidades",xlim
=c(0,1))
plot(density(chainAM[-(1:burnIn),1],bw=35),xlab="Sat",ylab="",main="Sat" )
abline(v =mean(chainAM[-(1:burnIn),1]),lwd=2,col="red")
abline(v =2200,lwd=2,lty=2,col="blue")
lines(c(intconf[1],intconf[2]),c(0,0),lwd=2,lty=5,col="purple")
legend("topleft",lty=c(5,1,2),lwd=c(2,2,2),bty="n",cex=0.8,col=c("purple","red","blue"),
legend=c("Intervalo de cred. 95%","Média da Cadeia","Valor real"))


inconf=0
intconf<-HDIofMCMC(chainAM[-(1:burnIn),2])
plot(density(chainAM[-(1:burnIn),2],bw=0.03),xlab="pes",ylab="",main="pes" )
abline(v =mean(chainAM[-(1:burnIn),2]),lwd=2,col="red")
abline(v =2.5,lwd=2,lty=2,col="blue")
lines(c(intconf[1],intconf[2]),c(0,0),lwd=2,lty=5,col="purple")
legend("topleft",lty=c(5,1,2),lwd=c(2,2,2),bty="n",cex=0.8,col=c("purple","red","blue"),
legend=c("Intervalo de cred. 95%","Média da Cadeia","Valor real"))


inconf=0
intconf<-HDIofMCMC(chainAM[-(1:burnIn),3])
plot(density(chainAM[-(1:burnIn),3],bw=0.01),xlab="Crec",ylab="",main="Crec" )
abline(v =mean(chainAM[-(1:burnIn),3]),lwd=2,col="red")
abline(v =0.3,lwd=2,lty=2,col="blue")
lines(c(intconf[1],intconf[2]),c(0,0),lwd=2,lty=5,col="purple")
legend("topleft",lty=c(5,1,2),lwd=c(2,2,2),bty="n",cex=0.8,col=c("purple","red","blue"),
legend=c("Intervalo de cred. 95%","Média da Cadeia","Valor real"))


plot(chainAM[-(1:burnIn),1],type="l",xlab="Iteração",ylab="Sat",main="Cadeia dos Valo
res de Sat")
abline(h =mean(chainAM[10000:50000,1]),lwd=2,col="darkred")
abline(h =2200,lwd=2,lty=2,col="blue")
legend("bottomright",lty=c(1,2),lwd=c(2,2),bty="n",cex=0.8,col=c("darkred","blue"),
legend=c("MédiaCadeia -AM","Valor real"))


plot(chainAM[-(1:burnIn),2],type="l",xlab="Iteração",ylab="pes",main="Cadeia dos Valo
res de pes")
abline(h =mean(chainAM[10000:50000,2]),lwd=2,col="darkred")
abline(h =2.5,lwd=2,lty=2,col="blue")
legend("bottomright",lty=c(1,2),lwd=c(2,2),bty="n",cex=0.8,col=c("darkred","blue"),
legend=c("MédiaCadeia -AM","Valor real"))


plot(chainAM[-(1:burnIn),3],type="l",xlab="Iteração",ylab="crec",main="Cadeia dos Val
ores de crec")
```

```
abline(h =mean(chainAM[10000:50000,3]),lwd=2,col="darkred")
abline(h =0.3,lwd=2,lty=2,col="blue")
legend("bottomright",lty=c(1,2),lwd=c(2,2),bty="n",cex=0.8,col=c("darkred","blue"),
legend=c("MédiaCadeia -AM","Valor real"))

#Olhandotodososparametros
par(mfrow =c(2,2))

inconf=0
intconf<-HDIofMCMC(chainAM[-(1:burnIn),4])
plot(density(chainAM[-(1:burnIn),4],bw=0.1),xlab="k",ylab="",main="k" )
abline(v =mean(chainAM[-(1:burnIn),4]),lwd=2,col="red")
abline(v =3.5,lwd=2,lty=2,col="blue")
lines(c(intconf[1],intconf[2]),c(0,0),lwd=2,lty=5,col="purple")
legend("topleft",lty=c(5,1,2),lwd=c(2,2,2),bty="n",cex=0.8,col=c("purple","red","blue"),
legend=c("Intervalo de cred. 95%","Média da Cadeia","Valor real"))

inconf=0
intconf<-HDIofMCMC(chainAM[-(1:burnIn),5])
plot(density(chainAM[-(1:burnIn),5],bw=1.5),xlab=expression(paste(sigma[M])),ylab=""
,main=expression(paste(sigma[M])) )
abline(v =mean(chainAM[-(1:burnIn),5]),lwd=2,col="red")
abline(v =300,lwd=2,lty=2,col="blue")
lines(c(intconf[1],intconf[2]),c(0,0),lwd=2,lty=5,col="purple")
legend("topleft",lty=c(5,1,2),lwd=c(2,2,2),bty="n",cex=0.8,col=c("purple","red","blue"),
legend=c("Intervalo de cred. 95%","Média da Cadeia","Valor real"))


plot(chainAM[-(1:burnIn),4],type="l",xlab="Iteração",ylab="k",main="Cadeia dos Valore
s de k")
abline(h =mean(chainAM[-(1:burnIn),4]),lwd=2,col="darkred")
abline(h =3.5,lwd=2,lty=2,col="blue")
legend("bottomright",lty=c(1,2),lwd=c(2,2),bty="n",cex=0.8,col=c("darkred","blue"),
legend=c("MédiaCadeia -AM","Valor real"))


plot(chainAM[-(1:burnIn),5],type="l",xlab="Iteração",ylab=expression(paste(sigma[M]))
,main=expression(paste("Cadeia dos valores ",sigma[M])))
abline(h =mean(chainAM[-(1:burnIn),5]),lwd=2,col="darkred")
abline(h =300,lwd=2,lty=2,col="blue")
legend("bottomright",lty=c(1,2),lwd=c(2,2),bty="n",cex=0.8,col=c("darkred","blue"),
legend=c("MédiaCadeia -AM","Valor real"))
```

APÊNDICE C – CÓDIGO FONTE: CASO COM DADOS REAIS

```
################################################################
#UNIVERSIDADE DE BRASÍLIA-UnB
#FACULDADE DE TECNOLOGIA
#DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL
#PÓS GRADUAÇÃO EM TECNOLOGIA AMBIENTAL E RECURSOS HÍDRICOS
#Data: 25/11/2015
#Autor: Cássio Guilherme Rampinelli
#Orientador: Dirceu Silveira Reis Jr.
#Coorientador:Carlos Henrique Ribeiro Lima
#email:cassiorampinelli@gmail.com
############################################################
#Código desenvolvido no âmbito das atividades do trabalho
#de dissertação de mestrado do referido autor.
#Obs:Para solicitar os dados EPQ desta simulação enviar email
#para o referido autor
#_____________________________________________________________#

################################################################
################DADOS REAIS- SMAP MENSAL#################
################################################################

#Limpa a memória do R
rm(list=ls(all=TRUE))

#Seta arquivo de trabalho para o R
setwd("C:/Users/ADM/Desktop/Simulacao_03_05_2016_Caso  Dados  Reais_Algoritmos
Implementados")

#Carrega Matriz com dados de Evaporação (E), Precipitação (P) e vazões Observadas (Q)
EPQ<-read.table(file="EPQCaso2.txt",header=T)

########################################
#PARTE 1: FUNÇÕES BÁSICAS
########################################

#####################################
#FUNÇÃO SMAP#########################
#####################################
#INPUT-Argumento de entrada
#EPQ-Matriz com os dados de Evaporação, Precipitação e Vazão Observada
#para-Vetor de parâmetros do modelo, no caso:sat;pes
#####################################################
#OUTPUT
#Qcalc-Vetor com vazões simuladas
#####################################################

smap<-function(EPQ,para,Area) #Declaração da função smap
```

```
{

#########################
#Inicializando variáveis#
#########################

E<-EPQ[,1] # Armazena a coluna E (evaporação), do data.frame EPQ no vetor E
Prec<-EPQ[,2] # Armazena a coluna P (precipitação), do data.frame EPQ no vetor Prec
n<-length(Prec) # Armazena o tamanho do vetor Prec na variável n
sat=para[1] #Armazena o elemento 1 do vetor para na variável sat
pes=para[2] #Armazena o elemento 2 do vetor para na variável pes
crec=para[3] #Armazena o elemento 3 do vetor para na variável crec
#crec=0#para região do Ceará
crecp=crec/100 #Transforma crec de percentual para centesimal
#crecp=crec
k=para[4] #Armazena o elemento 4 do vetor para na variável k
ke=0.5^(1/k) # Calcula a constante de recessão do escoamento superficial
#ke=0#para região do Ceará
#tuin=para[3] #Armazena o elemento 5 do vetor para na variável tuin
#tuinp=tuin/100;#Transforma tuin de percentual para centesimal

tuinp=0.4

#ebin=para[6]#Armazena o elemento 6 do vetor para na variável ebin
ebin=20

# As variáveis seguintes consistem em vetores e serão inicializados como nulos, pois
# o R requer uma definição a priore.
Rsol<-0
rsub<-0
tu<-0
dsol<-0
Qcalc<-0
es<-0
er<-0
eb<-0
rec<-0

#Inicialização dos primeiros elementos dos respectivos vetores
Rsol[1]=tuinp*sat
rsub[1]=ebin/(1-ke)/Area*2630
tu[1]=tuinp

#######################################
#Implementação do Algoritmo do Modelo Smap#
#######################################

i<-2 # Variável i é inicializada como 2, pois os primeiros elementos dos
# vetores de interesse já foram inicializados nas linhas anteriores
```

```
  while (i<=n+1) #Loop while que é processado até que o contador i atinga n+1

  {

    # Calcula a variação de nível do reservatorio do solo
    dsol[i]=0.5*(Prec[i-1]-Prec[i-1]*(tu[i-1]^pes)-E[i-1]*tu[i-1]-Rsol[i-1]*crecp*(tu[i-
1]^4))
    #Calcula a taxa de umidade no passo i
    tu[i]=(Rsol[i-1]+dsol[i])/sat
    #Calcula o escoamento de base

    if(tu[i]<0){tu[i]=0}
    if(tu[i]>1){tu[i]=1}

    es[i]=Prec[i-1]*tu[i]^pes
    #Calcula a evaporação real
    er[i]=E[i-1]*tu[i]
    #Calcula a recarga do reservatorio subterrâneo
    rec[i]=Rsol[i-1]*crecp*(tu[i]^4)
    #Calcula o escoamento de base
    eb[i]=rsub[i-1]*(1-ke)
    #Atualiza o nível do reservatório subterrâneo
    rsub[i]=rsub[i-1]-eb[i]+rec[i]
    #Atualiza o nível do reservatorio do solo
    Rsol[i]=Rsol[i-1]+Prec[i-1]-es[i]-er[i]-rec[i]

if(Rsol[i]>sat)
    {es[i]=es[i]+Rsol[i]-sat
    Rsol[i]=sat
    }

    if(Rsol[i]<0){Rsol[i]=0}


#Calcula vazão total simulada
    Qcalc[i-1]=(es[i]+eb[i])*Area/2630


    i<-i+1 #atualiza o contador


  }

  return(Qcalc) # retorna a série simulada

}

########################################
#FUNÇÃO OBJETIVO 1:NASH-SUTICLIFFE#####
```

```
#####################################
#INPUT-Argumento de entrada
#para-Vetor de parâmetros do modelo, no caso:sat;pes
#####################################################
#OUTPUT
#NS1-Valor da função objetivo x (-1)
#####################################################

FO1<-function (para)

{
 pen=0 #Inicializa penalidade

 #Verifica se os parâmetros estão dentro dos limites, caso não estejam
 #atribui-se penalidade à função objetivo
 if(para[1]>=limites[2,1]) pen=10000000000000000000
 if(para[1]<=limites[1,1])pen=10000000000000000000
 if(para[2]>=limites[2,2]) pen=10000000000000000000
 if(para[2]<=limites[1,2]) pen=10000000000000000000
 if(para[3]>=limites[2,3]) pen=10000000000000000000
 if(para[3]<=limites[1,3]) pen=10000000000000000000
 if(para[4]>=limites[2,4]) pen=10000000000000000000
 if(para[4]<=limites[1,4]) pen=10000000000000000000
 #if(para[5]>=limites[2,5])pen=10000000000000000000
 #if(para[5]<=limites[1,5]) pen=10000000000000000000
 #if(para[6]>=limites[2,6]) pen=10000000000000000000
 #if(para[6]<=limites[1,6]) pen=10000000000000000000

 Qcalc<-smap(EPQ,para,Area)#Executa a função SMAP
 Qobs<-EPQ[,3]#Armazena os valores das vazões observadas na variável Qobs

 #Calcula a função objetivo
 NS1<-(-1)*(1-((sum((Qobs-Qcalc)^2))/(sum((Qobs-mean(Qobs))^2))))+pen

 #Caso a função objetivo retorne um valor não numerico, atribui-se uma
 #penalidade
 return(ifelse(is.na(NS1), 10000000000000000000, NS1))

}

#####################################
#FUNÇÃO PSO#########################
#####################################
#INPUT-Argumento de entrada
#funcao-Função objetivo a ser considerada
#limites-limites de busca do vetor de parâmetros
#####################################################
#OUTPUT
#result-Lista com o vetor de parâmetros ótimos e o valor da função objetivo
#####################################################
```

```
#Define a função PSO,que tem como argumento uma função e uma matriz com os limites
#de busca tal qual como indicado acima.
PSO<-function(funcao,limites)
{

 n<-ncol(limites)#Denife o numero de colunas=par ordenado
 npar<-50#Número total de partículas a serem geradas
 maxger<-100#Número total de gerações, ou iterações
 POP<-matrix(0,nrow=npar,ncol=n)#Cria matriz que armazenará a população inicial


#########################################################################
 ####Geraçãoda População Inicial
 #########################################################################
 for (i in 1:npar) # Percorre o número de particulas até npar

 {
  for (j in 1:n)# Cria as coordenadas do par ordenado para cada particula

  {
   #Forma de gerar as coordenadas dos pontos aleatoriamente
   POP[i,j]= (limites[2,j]-limites[1,j])*runif(1)+limites[1,j]
  }

 }

#########################################################################
 ###Inicialização e definição de algumas variáveis e constantes de interesse###
 #########################################################################

 criterio<-0#Define o critério de parada
 geracao<-0#Inicializa um contador para o número de gerações ou iterações

 #Matriz que armazena o menor valor da função obtido pela particula
 pbestAJ<-matrix(0,nrow=1,ncol=npar)+99999999999999999999

 #Matriz que armazena as coordenadas (posição) do menor valor de funçao que
 # a particula já obteve
 pbestXX<-matrix(99999999999999999999,nrow=npar,ncol=n)

 #Armazena velocidade
 vant<-matrix(0,nrow=npar,ncol=n)

 #Constantes ou parâmetros de confiança: indicam o quanto uma partícula confia
 #na sua melhor posição(c1) ou na melhor posição ja obida por todas as particulas(c2)
 c1=2
 c2=2
```

```
#Inércia da particula, que controla a capacidade de exploração do algoritmo.
#Um valor alto facilita um comportamento mais global. Um valor baixo facilita
#um comportamento mais local.
w=1
#forma proposta para atualizar o parâmetro w a cada passo da iteração
passo=(w-0.4)/maxger

#Inicializando matriz auxiliar para armazenar os valores máximos da velocidade
vmax=(limites[2,]-limites[1,])*0.2
#Inicializando matriz velocidade
v<-matrix(0,nrow=npar,ncol=n)
#Inicializando AJ, que consiste em um vetor que armazena os valores da função
AJ<-0
#Inicializando variável auxiliar x
x<-0
#Contador, utilizado apenas no caso de querer gerar figuras sequenciais para
#posteriormente gerar imagens animadas:.gifs
#count<-1


##############################################################################
#######Implementação do processo iterativo do algoritmo#####################
##############################################################################


while (!criterio) #Enquanto critério for igual a zero o loop é processado.
  #criterio, foi definido como sendo zero em linhas anteriores.

{

  geracao=geracao+1 #atualiza o numero de gerações, ou iterações.
  w=w-passo        #atualiza o parâmetro de inércia

  for (i in 1:npar)#For que percorre o número total de particulas

  {

    for(j in 1:n)#For que percorre as coordenadas de cada particula
    {
     x[j]<-POP[i,j]#Armazena as coordenadas da partícula em um vetor auxiliar x
    }
    AJ[i]=funcao(x) #Aplica a função nas coordenadas de x e o valor retornado é
    #armazenado no vetor AJ

    if(AJ[i]<pbestAJ[i])#Caso o valor obtido para a função AJ seja menor que o
      #menor valor já encontrado por esta partícula processa-se o if
    {
      pbestAJ[i]=AJ[i]  #O melhor valor da particula passa a ser o valor encontrado
      pbestXX[i,]=POP[i,] #Guarda as coordenadas da partícula com o melhor valor
encontrado
```

```
    }
   }

   #Armazena o índice (ou a linha) do melhor valor de todos os melhores individuais
   #já encontrados por cada particula.
gbestAJindice<-which(pbestAJ==min(pbestAJ),arr.ind=TRUE)

#Armazena as coordenadas do melhor valor já encontrado
   gbestXX=pbestXX[gbestAJindice[1,2],]

   ######################################################################
   ######################ATUALIZAÇÃO#####################################

   for (i in 1:npar) # For que percorre o número de partículas
   {


     #Atualiza o vetor velocidade de acordo com formulação do algoritmo PSO
v[i,]=(w*vant[i])+(c1*runif(1)*(pbestXX[i,]-POP[i,]))+(c2*runif(1)*(gbestXX-POP[i,]))

for(j in 1:n) #For que percorre as coordenadas de cada particula
     {

      if(v[i,j]>vmax[j])#Se a velocidade atual for maior que a velocidade máxima
      {
       v[i,j]=vmax[j]#Velocidade atual recebe velocidade máxima
      }
     }


     POP[i,]=POP[i,]+v[i,]#Atualiza a posição da particula a partir da nova velocidade
     vant[i]=v[i]#Atualiza a velocidade da partícula para o passo posterior.

   }


   if (geracao==maxger)#Se o contador geracao atingir o número total de gerações
   {criterio=1}#Sera atribuído 1 a variável critério e interrompe-se o loop while


  }

  #Armazena em result uma lista que contem o vetor de parâmetros ótimos e
  # o respectivo valor da função objetivo.

  result<-list("Par_Otimo:",gbestXX,"FO:",funcao(gbestXX))
  return(result)#Imprime no prompt a melhor posição de todas após a finalização do
algoritmo

}
```

```
########################################
#FUNÇÃO SOMA############################
########################################
#INPUT-Argumento de entrada
#Qobs-vetor com vazões observadas
#Qcalc-vetor com vazões calculadas
#########################################################
#OUTPUT
#result-soma dos desvios ao quadrado (variância)
#########################################################

#Soma os desvios ao quadrado
############################
soma = function(Qobs,Qcalc)
{
 result = 0
 for (t in 1:length(Qcalc))
 {
  result = result + (Qobs[t]-Qcalc[t])^2
}
 result
}


########################################
#PARTE 2: CALIBRAÇÃO CONVENCIONAL
########################################

#Definição os limtes do espaço de busca para calibração.
limites<-rbind(c(300,0.5,0.01,0.01),c(5000,8,20,10))
#A ordem dos parâmetros no vetor e indicada abaixo:
#limites<-rbind(c(SAT_min,pes_min,CREC_min,k_min,tuin_min,ebin_min),
#c(SAT_max,pes_max,CREC_max,k_max,tuin_max,ebin_max))

#Área da bacia em km²
Area=4860

#Chama a função PSO e armazena o resultado na variável:resultadoCalibracaoPSO
resultadoCalibracaoPSO<-PSO(FO1,limites)

#Mostra os valores calibrados no prompt do R
resultadoCalibracaoPSO

#vetor inicial de parâmetros para calibração via NelderMead
#Digitar os valores obtidos a partir da calibração PSO
parai<-c(2721.251008 ,2.164960,4.552448,5.633842)

#Aplica a função optim que contem o algoritmo de Nelder & Mead
# e armazena a saída da busca local na variável result1
```

```
result1<-optim(parai, FO1)#Armazena na variável result1 a saída da rotina

#mostra o vetor final de parâmetros calibrados
result1

#armazena o vetor de parãmetros ótimos na vairável paraf1
paraf1<-result1$par

#Armazena as vazões calculadas com base no vetor ótimo de parâmetros
Qcalc=smap(EPQ,paraf1,Area)

#Armazena valores de vazões observadas
Qobs=EPQ[,3]

#Plota as vazões observadas e as vazões calibradas pela calibração convencional
plot(Qcalc,type="l",col="red",lwd=2,xlab="mês",ylab="Vazão(m³/s)",main="Vazões
Observadas e Simuladas")
lines(Qobs,col="black",type="l",lty=3,lwd=2)
legend("topright",lty=c(1,3),lwd=c(2,2),bty="n",cex=0.8,col=c("red","black"),
legend=c("Vazões simuladas","Vazões observadas"))

#resíduosPSO
residuosPSO=Qcalc-Qobs

#desvioPadrão dos resíduos
sdResiduosPSO=sd(residuosPSO)

##############################################################################
#####PARTE 3: FUNÇÕES AUXILIARES E ALGORITMOS MCMC
##############################################################################

#####################################################################
##FUNÇÕES A PRIORI
#####################################################################

prior <- function(param){

  #passa os elementos do vetor param para as variáveis sat, pes, crec, k e sd
  sat = param[1]
  pes = param[2]
  crec = param[3]
  k=param[4]
  sd=param[5]


  #PRIORI SAT
satprior = dunif(sat,min=500, max = 5000,log = T)

  #PRIORI PES
```

```
  pesprior = dunif(pes,min=0.5,max=10,log = T)

  #PRIORI CREC
  crecprior = dunif(crec,min=0.5,max=40,log = T)

  #PRIORI K
  kprior = dunif(k,min=0,max=20,log = T)

  #PRIORI SD
  sdprior = dunif(sd,min=1,max=50,log = T)


#retorna o ln do produto das prioris
  return(satprior+pesprior+crecprior+kprior+sdprior)
}

#############################################################################
#FUNÇÃO DE LOG-VEROSSIMILHANÇA ASSUMINDO NORMALIDADE E HOMOCEDASTICIDADE
#############################################################################


##Função Verossimilhança
## Verossimilhança: Adotou-se o Ln da verossimilhança
################################
likelihood <- function(param){

  #armazena os parâmetros de entrada
  sat = param[1]
  pes = param[2]
  crec = param[3]
  k=param[4]
  sd=param[5]

  phi=1/sd^2 # Parâmetro de precisão

  Prec<-EPQ[,2] # Armazena a coluna P (precipitação), do data.frame EPQ no vetor Prec
  N<-length(Prec) # Armazena o tamanho do vetor Prec na variável n

  likelihood=0
  somadesv=0


  A=Area
  para<-c(sat,pes,crec,k)
  EPQ<-EPQ
  Qcalc=smap(EPQ,para,Area)
  Qobs<-EPQ[,3]
  somadesv=soma(Qobs,Qcalc)
```

```
  likelihood=(N/2)*log(phi)-(phi/2)*somadesv
  return(likelihood)

}


#######################################################################
#FUNÇÕES PROPOSTAS AUXILIARES PARA O ALGORITMO DE METROPOLIS-
HASTINGS
#######################################################################
##Funções propostas
##########################################
proposalfunction <- function(param){


  sat = param[1]
  pes = param[2]
  crec = param[3]
  k=param[4]
  sd=param[5]



  #PROPOSTA SAT
  #CV=50
  #EX=sat
  #BETTA=CV^2/EX
  #ALFFA=EX*BETTA
  #satp = rgamma(1,ALFFA,BETTA)

#plot(function(sat){dgamma(sat,ALFFA,BETTA)},xlim=c(sat-
1000,sat+1000),ylab="f(sat)",xlab="sat",lty=1)

  satp=rlnorm(1,log(sat),sd=0.1)


#PROPOSTA PES
  CV=50
  EX=pes
  BETTA=CV^2/EX
  ALFFA=EX*BETTA
  pesp = rgamma(1,ALFFA,BETTA)
#plot(function(pes){dgamma(pes,ALFFA,BETTA)},xlim=c(pes-
2,pes+2),ylab="f(pes)",xlab="pes",lty=1)

#PROPOSTA CREC
  CV=50
  EX=crec
  BETTA=CV^2/EX
  ALFFA=EX*BETTA
```

```
  crecp = rgamma(1,ALFFA,BETTA)
  #plot(function(crecp){dgamma(crecp,ALFFA,BETTA)},xlim=c(crec-
0.3,crec+0.3),ylab="f(crecp)",xlab="crecp",lty=1)



  #PROPOSTA k
  #CV=50
  #EX=k
  #BETTA=CV^2/EX
  #ALFFA=EX*BETTA
  #kp = rgamma(1,ALFFA,BETTA)
  #plot(function(kp){dgamma(kp,ALFFA,BETTA)},xlim=c(kp-
1,kp+1),ylab="f(kp)",xlab="kp",lty=1)
kp = rlnorm(1,mean=log(k), sd = 0.2)


#PROPOSTA sd
  phi=1/sd^2
  CV=45
  EX=phi
  BETTA=CV^2/EX
  ALFFA=EX*BETTA

phip=rgamma(1,ALFFA,BETTA)
  sdp=sqrt(1/phip)


  #plot(function(sd){dgamma(sd,ALFFA,BETTA)},xlim=c(1/(sd-
30)^2,1/(sd+30)^2),ylab="f(sd)",xlab="sd",lty=1)

  return(c(satp,pesp,crecp,kp,sdp))

}


#propostas aplicadas nos valores correntes com os parâmetros nos valores
proposal1 <- function(param){

  sati=param[1]
  pesi=param[2]
  creci=param[3]
  ki=param[4]
  sdi=param[5]

  satp=param[6]
  pesp=param[7]
  crecp=param[8]
  kp=param[9]
  sdp=param[10]
```

```
  #Proposta do SAT nos valores correntes com base nos valores propostos
satproposal=dlnorm(sati,mean=log(satp),sd=0.1,log=T)

#Proposta do PES nos valores correntes com base nos valores propostos
  #CVi=8
  CVi=50
  EXi=pesi
  BETTAi=CVi^2/EXi
ALFFAi=EXi*BETTAi

  #CVp=8
  CVp=50
  EXp=pesp
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp

  pesproposal=dgamma(pesi,ALFFAp,BETTAp,log=T)



#Proposta do CREC nos valores correntes com base nos valores propostos
  #CVi=5
  CVi=50
  EXi=creci
  BETTAi=CVi^2/EXi
  ALFFAi=EXi*BETTAi

  #CVp=5
  CVp=50
EXp=crecp
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp

  crecproposal=dgamma(creci,ALFFAp,BETTAp,log=T)


#Proposta do K nos valores correntes com base nos valores propostos
#CVi=10
  #CVi=50
  #EXi=ki
  #BETTAi=CVi^2/EXi
  #ALFFAi=EXi*BETTAi

  #CVp=10
  #CVp=50
  #EXp=kp
  #BETTAp=CVp^2/EXp
  #ALFFAp=EXp*BETTAp
```

```
  #kproposal=dgamma(ki,ALFFAp,BETTAp,log=T)

  kproposal=dlnorm(ki,mean=log(kp),sd=0.2,log=T)


#Proposta do SD nos valores correntes com base nos valores propostos

phii=1/sdi^2
  #CVi=10
  CVi=45
  EXi=phii
  BETTAi=CVi^2/EXi
  ALFFAi=EXi*BETTAi


  phip=1/sdp^2
  #CVp=10
  CVp=45
  EXp=phip
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp


  phiproposal=dgamma(phii,ALFFAp,BETTAp,log=T)

  sum=satproposal+pesproposal+crecproposal+kproposal+phiproposal

return(sum)

}

#propostas aplicadas nos valores propostos com os parâmetros nos valores
#correntes
proposal2 <- function(param){

  sati=param[1]
  pesi=param[2]
  creci=param[3]
  ki=param[4]
  sdi=param[5]

  satp=param[6]
  pesp=param[7]
  crecp=param[8]
  kp=param[9]
  sdp=param[10]

  #Proposta do SAT nos valores correntes com base nos valores propostos
satproposal=dlnorm(satp,mean=log(sati),sd=0.1,log=T)
```

```
#Proposta do PES nos valores correntes com base nos valores propostos
  #CVi=8
  CVi=50
  EXi=pesi
  BETTAi=CVi^2/EXi
ALFFAi=EXi*BETTAi

  #CVp=8
  CVp=50
  EXp=pesp
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp

pesproposal=dgamma(pesp,ALFFAi,BETTAi,log=T)



  #Proposta do CREC nos valores correntes com base nos valores propostos
  #CVi=5
  CVi=50
  EXi=creci
  BETTAi=CVi^2/EXi
  ALFFAi=EXi*BETTAi

  #CVp=5
  CVp=50
EXp=crecp
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp

crecproposal=dgamma(crecp,ALFFAi,BETTAi,log=T)


  #Proposta do K nos valores correntes com base nos valores propostos
#CVi=10
  #CVi=50
  #EXi=ki
  #BETTAi=CVi^2/EXi
  #ALFFAi=EXi*BETTAi

  #CVp=10
  #CVp=50
  #EXp=kp
  #BETTAp=CVp^2/EXp
  #ALFFAp=EXp*BETTAp

  #kproposal=dgamma(kp,ALFFAi,BETTAi,log=T)
```

```
  kproposal=dlnorm(kp,mean=log(ki),sd=0.2,log=T)



#Proposta do SD nos valores correntes com base nos valores propostos

phii=1/sdi^2
  #CVi=10
  CVi=45
  EXi=phii
  BETTAi=CVi^2/EXi
  ALFFAi=EXi*BETTAi


  phip=1/sdp^2
  #CVp=10
  CVp=45
  EXp=phip
  BETTAp=CVp^2/EXp
  ALFFAp=EXp*BETTAp


  phiproposal=dgamma(phip,ALFFAi,BETTAi,log=T)

  sum=satproposal+pesproposal+crecproposal+kproposal+phiproposal

return(sum)

}


############################
#FUNÇÃO POSTERIOR
############################

##Função Posterior
##Ln do produto da verossimilhança pela priori

posterior <- function(param){
  return (likelihood(param) + prior(param))
}



#########################################################################
#ALGORITMO ADAPTIVE METROPOLIS - AM
#########################################################################
```

```
run_AdaptiveMetropolis_MCMC <- function(startvalue, iterations){

#Cria a matriz que irá armazenar os valores das cadeias para os 5 parâmetros
#SAT, PES, CREC, K, SD
  chain = array(dim = c(iterations+1,5))

#Inicializa os elementos da primeira linha da matriz
chain[1,] = startvalue

  #Variância SAT
  varSat=30^2

#Variância PES
  varPes= 0.08^2

  #VariânciaCrec
  varCrec=0.08^2

  #Variância K
  varK=0.1^2

  #Variância SD
  varSd=15^2

  #Cria a matriz de variância/covariância C
  C=matrix(0,nrow=5,ncol=5)


  #Atribui as variâncias para a diagonal da matriz C
diag(C)=c(varSat,varPes,varCrec,varK,varSd)



#Matriz Identidade
  Id=diag(x=1,nrow=5,ncol=5)

  #Parâmetros do algoritmo AM
  sd=(2.4)^2/5
  E=0.001

  #Variáveis que armazenam números de aceitação e rejeição na cadeia
  naceito=0
  nrejeitado=0

  #Matriz que armazena os valores propostos da cadeia
p=matrix(0,nrow=iterations,ncol=5)

# Valores iniciais para a matriz C, conforme algoritmo Metropolis
  # para os 600 passos iniciais
  for(i in 1:600)
```

```
  {


   #proposta a partir de uma distribuição normal multivariada
proposal=mvrnorm(1,chain[i,],C)

#armazena os valores propostos na matriz p
   p[i,]=proposal

   #calcula a razão de aceitação
   razao=posterior(proposal)-posterior(chain[i,])

   if (exp(razao) == "NaN"){

     chain[i+1,]=chain[i,]
     nrejeitado=nrejeitado+1
}

   else

     if(runif(1,0,1)<min(1,exp(razao))){

chain[i+1,]=proposal
       naceito=naceito+1
}

   else{

     chain[i+1,] = chain[i,]
nrejeitado=nrejeitado+1

   }


  }

 #Passos seguintes após a iteração inicial calculando a matriz de variância covariância
 #com base na matriz C do passo anterior

for (i in 601:iterations){


chainMed1=c(mean(chain[1:i,1]),mean(chain[1:i,2]),mean(chain[1:i,3]),mean(chain[1:i,4])
,mean(chain[1:i,5]))
   chainMed2=c(mean(chain[1:(i-1),1]),mean(chain[1:(i-1),2]),mean(chain[1:(i-
1),3]),mean(chain[1:(i-1),4]),mean(chain[1:(i-1),5]))
   C=((i-1)/i)*C+(sd/i)*(i*chainMed2%*%t(chainMed2)-
(i+1)*chainMed1%*%t(chainMed1)+chain[i,]%*%t(chain[i,])+E*Id)

#valores propostos a partir de uma distribuição normal multivariada
```

```
   #com base na matriz C
   proposal=mvrnorm(1,chain[i,],C)
   #Armazena o valor proposto na matriz p
   p[i,]=proposal

   #Calcula a razão de aceitação
   razao=posterior(proposal)-posterior(chain[i,])


   if (exp(razao) == "NaN"){

     chain[i+1,]=chain[i,]
     nrejeitado=nrejeitado+1
}

   else

     if(runif(1,0,1)<min(1,exp(razao))){

chain[i+1,]=proposal
       naceito=naceito+1
}

   else{

     chain[i+1,] = chain[i,]
nrejeitado=nrejeitado+1

   }
  }

  #Retorna uma lista com a cadeia final, número de aceitação, número de rejeição
  #e a matriz de valores propostos

  lista=list(chain,naceito,nrejeitado,p)

  return(lista)
}


################################################
##PARTE 8: ALGORITMO METROPOLIS HASTINGS - MH
################################################

#Função do algoritmo Metropolis Hastings
#Argumentos de entrada:
#vetor de parâmetros iniciais- startvalue=(SAT,PES,CREC,K,Desvio Padrão)
#Número de iterações- iterations=(Número de Iterações)
run_MetropolisHastings <- function(startvalue, iterations){
```

```
#cria a matriz que armazenará os valores da cadeia para os 5 parâmetros
  #com a dimensão: nº linhas=Número de iterações +1; nº colunas=nº de parâmetros (5)
chain = array(dim = c(iterations+1,5))

#inicializa a primeira linha da matriz com o vetor de valores iniciais para os parâmetros
  chain[1,] = startvalue

  #inicializa as variáveis que armazenarão o número de aceitação e rejeição dos
  #passos nas cadeias
  naceito=0
  nrejeitado=0

  #Inicializa o avanço ao longo dos passos da cadeia até o número total de iterações
  for (i in 1:iterations){

    #Chama a função proposalfuncion que retorna o vetor de parâmetros propostos que é
    #armazenado na variável proposal
    proposal= proposalfunction(chain[i,])

    #armazena na variável valores o vetor com os valores dos parâmetros no passo
    #corrente na cadeia e o vetor de parâmetros propostos
    valores=c(chain[i,],proposal)

    #calcula a razão de aceitação de Hastings fazendo uso das funções propostas
    #auxiliares
    razao=posterior(proposal)+proposal1(valores)-posterior(chain[i,])-proposal1(valores)

    #condicional que faz um tratamento de erro, ou seja, caso algum valor não
    #numérico surja na razão de aceitação se mantém no passo posterior da cadeia
    #os valores do passo atual
    if (exp(razao) == "NaN"){

      chain[i+1,]=chain[i,]
      nrejeitado=nrejeitado+1
    }

    else

      #Condicional que permite a aceitação do valor proposto
      #sorteia-se um número de 0 a 1 e compara-se o valor
      #sorteado com o exponencial da razão de aceitação (ou a unidade, caso
      #o valor do exponencial da razão de aceitação seja maior). Se o valor sorteado
      #for menor, aceita-se o valor proposto
if (runif(1,0,1)<min(1,exp(razao))){

#armazena-se no passo seguinte da cadeia o valor proposto
       chain[i+1,]=proposal
       #contabiliza-se o número de aceitação
       naceito=naceito+1
```

```
    }
   else{

    #caso contrário rejeita-se o valor proposto e armazena-se no passo seguinte
    #da cadeia o valor do passo corrente
    chain[i+1,] = chain[i,]
    #contabiliza-se o número de rejeição
    nrejeitado=nrejeitado+1
   }

 }

 #retorna uma lista contendo todos os valores da cadeia (chain), o número de valores
 #aceitos (naceito) e o número de valores rejeitados (nrejeitado)
 lista=list(chain,naceito,nrejeitado)

 return(lista)
}


#####################################################################
##PARTE 4: RODANDO OS ALGORITMOS MCMC
#####################################################################

#Número de iterações
nIter=150000
iterations=nIter

#Chama o pacote MASS que habilita a função normal multivariada, usada no algoritmo
AM
library(MASS)

#####################################################################
######## Cadeia 1 - Algoritmo AM
#####################################################################

# Valores iniciais para o primeiro conjunto de cadeias
startvalue <- c(1700,2.6,21,4.5,24)
#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoAM1<-run_AdaptiveMetropolis_MCMC(startvalue,iterations)
proc.time() - a

#Número de valores aceitos
resultadoAM1[[2]]
#Número de valores rejetiados
resultadoAM1[[3]]
```

```
#Salva na variável chainAM1 a cadeia final
chainAM1=resultadoAM1[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainAM1, "chainAM1.txt",sep="\t",row.names=F )


#####################################################################
######## Cadeia 1 - Algoritmo MH
#####################################################################

# Valores iniciais para o primeiro conjunto de cadeias
startvalue <- c(1700,2.6,21,4.5,24)

#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoMH1<-run_MetropolisHastings(startvalue,iterations)
proc.time() - a

#Número de valores aceitos
resultadoMH1[[2]]
#Número de valores rejetiados
resultadoMH1[[3]]

#Salva na variável chainAM1 a cadeia final
chainMH1=resultadoMH1[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainMH1, "chainMH1.txt",sep="\t",row.names=F )


#####################################################################
######## Cadeia 2 - Algoritmo AM
#####################################################################

# Valores iniciais para o segundo conjunto de cadeias
startvalue <- c(800,1.8,15,3,17)

#Roda a segunda cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoAM2<-run_AdaptiveMetropolis_MCMC(startvalue,iterations)
proc.time() - a

#Número de valores aceitos
resultadoAM2[[2]]
#Número de valores rejetiados
```

```
resultadoAM2[[3]]

#Salva na variável chainAM1 a cadeia final
chainAM2=resultadoAM2[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainAM2, "chainAM2.txt",sep="\t",row.names=F )

#################################################################
######## Cadeia 2 - Algoritmo MH
#################################################################

# Valores iniciais para o segundo conjunto de cadeias
startvalue <- c(1200,1.8,15,3,15)

#Roda a primeira cadeia do algoritmo Adaptive Metropolis e salva o tempo na variável e
#contabiliza o tempo de processamento

a <- proc.time()
resultadoMH2<-run_MetropolisHastings(startvalue,iterations)
proc.time() - a

#Número de valores aceitos
resultadoMH2[[2]]
#Número de valores rejetiados
resultadoMH2[[3]]

#Salva na variável chainAM1 a cadeia final
chainMH2=resultadoMH2[[1]]

#Cria um arquivo txt  para a cadeia gerada
write.table( chainMH2, "chainMH2.txt",sep="\t",row.names=F )

#################################################################
##PARTE 5: GRÁFICOS DE SAÍDA
#################################################################

#valores para descartar
burnIn <- 60000

#Olhando todos os parametros
par(mfrow = c(2,2))

plot(chainAM1[,1],type="l",xlab="Iteração",ylab="SAT",main="Cadeia dos Valores de
SAT",col="black")
lines(chainAM2[,1],col="red")
lines(chainMH1[,1],col="blue")
lines(chainMH2[,1],col="green")
abline(h = paraf1[1],lwd=2,lty=2,col="blue")
```

```
legend("topright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("black",
"red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))


plot(chainAM1[,2],type="l",xlab="Iteração",ylab="PES",main="Cadeia dos Valores de
PES",col="black")
lines(chainAM2[,2],col="red")
lines(chainMH1[,2],col="blue")
lines(chainMH2[,2],col="green")
abline(h = paraf1[2],lwd=2,lty=2,col="blue")
legend("topright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("black",
"red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))


plot(chainAM1[,3],type="l",xlab="Iteração",ylab="CREC",main="Cadeia dos Valores de
CREC",col="black")
lines(chainAM2[,3],col="red")
lines(chainMH1[,3],col="blue")
lines(chainMH2[,3],col="green")
abline(h = paraf1[3],lwd=2,lty=2,col="blue")
legend("bottomright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("bla
ck","red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))

par(mfrow = c(1,1))
plot(chainAM1[,4],type="l",xlab="Iteração",ylab="K",main="Cadeia dos Valores de
K",col="black")
lines(chainAM2[,4],col="red")
lines(chainMH1[,4],col="blue")
lines(chainMH2[,4],col="green")
abline(h = paraf1[4],lwd=2,lty=2,col="blue")
legend("bottomright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("bla
ck","red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))


#Olhando todos os parametros
par(mfrow = c(1,1))
plot(chainAM1[,5],type="l",xlab="Iteração",ylab=expression(paste(sigma[])),main=expres
sion(paste("Cadeia dos valores ",sigma[])),col="black")
lines(chainAM2[,5],col="red")
lines(chainMH1[,5],col="blue")
lines(chainMH2[,5],col="green")
abline(h = 20.45,lwd=2,lty=2,col="blue")
```

```
legend("topright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("black",
"red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))
#Olhando todos os parametros
par(mfrow = c(2,2))

plot(chainAM1[,1],type="l",xlab="Iteração",ylab="SAT",main="Cadeia dos Valores de
SAT",col="black")
lines(chainAM2[,1],col="red")
lines(chainMH1[,1],col="blue")
lines(chainMH2[,1],col="green")
abline(h = 1790.18,lwd=2,lty=2,col="blue")
legend("topright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("black",
"red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))


plot(chainAM1[,2],type="l",xlab="Iteração",ylab="PES",main="Cadeia dos Valores de
PES",col="black")
lines(chainAM2[,2],col="red")
lines(chainMH1[,2],col="blue")
lines(chainMH2[,2],col="green")
abline(h = 2.46,lwd=2,lty=2,col="blue")
legend("topright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("black",
"red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))


plot(chainAM1[,3],type="l",xlab="Iteração",ylab="CREC",main="Cadeia dos Valores de
CREC",col="black")
lines(chainAM2[,3],col="red")
lines(chainMH1[,3],col="blue")
lines(chainMH2[,3],col="green")
abline(h = 35.35,lwd=2,lty=2,col="blue")
legend("bottomright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("bla
ck","red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))

par(mfrow = c(1,1))
plot(chainAM1[,4],type="l",xlab="Iteração",ylab="K",main="Cadeia dos Valores de
K",col="black")
lines(chainAM2[,4],col="red")
lines(chainMH1[,4],col="blue")
lines(chainMH2[,4],col="green")
abline(h = 4.27,lwd=2,lty=2,col="blue")
```

```
legend("bottomright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("bla
ck","red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))


#Olhando todos os parametros
par(mfrow = c(1,1))
plot(chainAM1[,5],type="l",xlab="Iteração",ylab=expression(paste(sigma[])),main=expres
sion(paste("Cadeia dos valores ",sigma[])),col="black")
lines(chainAM2[,5],col="red")
lines(chainMH1[,5],col="blue")
lines(chainMH2[,5],col="green")
abline(h = sdResiduosPSO,lwd=2,lty=2,col="blue")
legend("topright",lty=c(1,1,1,1,2),lwd=c(1.5,1.5,1.5,1.5,2),bty="n",cex=0.8,col=c("black",
"red","blue","green","blue"),
legend=c("AM-Cadeia 1","AM-Cadeia 2","MH-Cadeia 1","MH-Cadeia 2","Calibração
PSO"))
```