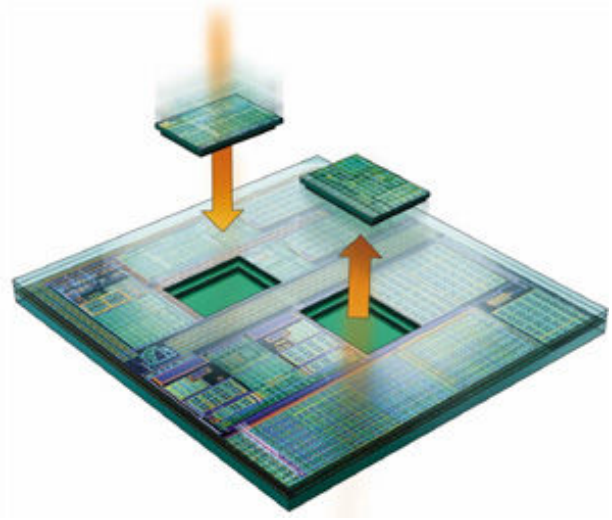
Universidade Federal de Pernambuco
Centro de Informática
Pós-Graduação em Ciência da Computação

Estatística Aplicada
Profª Renata Souza



Modelos Lineares Generalizados para Modelagem e Predição de Desempenho de Sistemas Embarcados

Guilherme Esmeraldo

Recife, março de 2008

Sumário

1. Introdução

Sistemas embarcados estão em todos os lugares, embora, frequentemente, passem despercebidos em nossas vidas cotidianas. Estão presentes em telefonia (celulares e centrais telefônicas), em eletrodomésticos (fornos microondas, máquinas de lavar, televisores e dvd *players*), automóveis (controladores de injeção eletrônica, freios ABS e sensores em geral), vídeo games, calculadoras, PDA's, *drivers* de armazenamento (HDs e disquetes), equipamentos de rede (*hubs*, *switches* e *firewalls*), etc.

Um sistema embarcado é um subsistema eletrônico específico para uma aplicação e é utilizado em uma entidade maior como um componente, um instrumento ou um veículo. O sistema embarcado pode encapsular a funcionalidade do sistema em duas formas: executando software em CPUs ou em hardware especializado. [24]

Devido tanto ao desenvolvimento das tecnologias de circuitos integrados (CIs) quanto à necessidade de novas aplicações, os sistemas embarcados passaram a ficar mais complexos, então surgiram os *Systems-on-Chip* (SoCs).

Um SoC é um CI que implementa muitas ou todas as funções de um sistema eletrônico completo [25]. A característica fundamental de um SoC é sua complexidade. Um chip de memória, por exemplo, pode ter muitos transistores, mas sua estrutura regular faz com que seja um componente e não um sistema. Os componentes de um SoC variam de acordo com a aplicação. Um SoC pode incluir memória *on-chip* (RAM e/ou ROM), microprocessador, interfaces com periféricos, lógica de controle de I/O, conversores de dados, e outros componentes que abrangem sistemas computacionais completos.

Multi-Processor Systems-on-Chip (MPSoCs) são SoCs que incluem vários processadores. Na prática, muitos SoCs são MPSoCs, até mesmo porque é muito difícil projetar um SoC complexo sem utilizar muitos processadores[25].

No projeto de MPSoCs, existem algumas restrições que devem ser observadas, entre elas: alto desempenho, capacidade de executar aplicações de tempo-real, áreas físicas minimizadas e bem utilizadas, baixo consumo de potência ou mecanismos para, dinamicamente, diminuir esse consumo e interfaces com outros dispositivos, visto que um MPSoC é um sistema completo.

Técnicas, como *Plaftorm-Based Design* [41] e *Eletronic System-Level* [26], procuram, através de reuso de componentes (plataformas), bem como projetar em altos níveis de abstração, fornecer mecanismos para simplificar e tornar mais dinâmico o processo de desenvolvimento de MPSoCs, aumentando a produtividade dos projetistas.

A técnica *Platform Based Design* (PBD) é definida como a criação de plataformas estáveis baseadas em microprocessadores que podem rapidamente ser estendidas, adaptadas para uma classe de aplicações, e podem ser utilizadas por projetistas para desenvolvimento rápido [41]. PBD tem sido utilizada com o objetivo de agilizar o processo de desenvolvimento de sistemas dedicados complexos com produtividade e qualidade [40]. Segundo esta metodologia, uma aplicação, definindo unicamente a funcionalidade do sistema que se deseja criar, sem nenhum detalhe de implementação, deve ser mapeada em uma plataforma, que na realidade é pré-concebida para cada grupo, ou família, de aplicações. Após o processo de mapeamento, o sistema é analisado de acordo com determinadas métricas consideradas como essenciais ao projeto. Neste

momento, o projetista tem a liberdade de aplicar ajustes na aplicação, na plataforma ou até mesmo no próprio mapeamento entre ambas as partes. Durante estes ajustes, o foco é a exploração do espaço de alternativas de projeto em busca da obtenção de melhores resultados segundo os requisitos definidos para a aplicação.

Aliada à PBD, outra tendência que permite aumentar a produtividade, é fazer modelagem de plataformas em altos níveis de abstração. Esta técnica é denominada *Eletronic System-Level* (ESL). Modelos ESL permitem abordagens unificadas para desenvolvimento de sistemas de hardware e software. Estes modelos, usualmente, descritos em linguagens de alto nível, como C/C++ [39], SystemC [6] (ver Apêndice A) ou SPecC [14], fornecem uma estimativa das características do sistema no fluxo de projeto antes de passar para o desenvolvimento de baixo nível.

1.1. Definição do Problema

MPSoCs são compostos por muitos componentes heterogêneos, portanto suas arquiteturas de comunicação on-chip devem atender às necessidades de comunicação das aplicações.

Arquitetura de comunicação *on-chip* refere-se a uma estrutura física que integra componentes de um MPSoC e disponibiliza um mecanismo para troca de dados e informações de controle entre eles [25]. Estas interconexões podem variar em complexidade, partindo de canais dedicados até redes intra-chip (*networks-on-chip*) [43].

Vários estudos demonstram o impacto da arquitetura de comunicação *on-chip* no desempenho e consumo de energia do sistema [18, 52, 5, 23, 56, 47]. Segundo [35, 30], a arquitetura de comunicação tem seu consumo de energia equiparável, em magnitude, às fontes primárias de consumo (e.g. processadores e memórias cache).

Null e Lobur em [38] citam que o barramento, por ser um recurso compartilhado, torna-se o gargalo na comunicação.

A utilização de barramentos de maior largura, alta freqüência, presença de DMA e *pipeline* podem aumentar, significantemente, o desempenho da plataforma [9]. Vahid e Givargis citam em [42], também, que a utilização de hierarquia de barramento pode diminuir a latência provinda de periféricos.

A utilização de algoritmos de arbitragem mais eficientes torna possível reduzir a contenção, no caso de vários dispositivos concorrendo pela utilização de um barramento compartilhado, aumentando também, conseqüentemente, o desempenho do sistema [18, 16, 43].

Em contrapartida, quando mal utilizadas, todas estas características podem prejudicar o desempenho e aumentar o consumo de energia, podendo inviabilizar o projeto de um MPSoC [31].

Assim, com tantas características, projetistas têm dificuldades para selecionar a arquitetura de comunicação mais adequada para um MPSoC executando uma aplicação particular.

Técnicas como PBD e ESL visam auxiliar, através de modelos abstratos (software), na escolha dessas características. Porém, dado o aumento das funções e parâmetros das arquiteturas de comunicação on-chip, a necessidade

de examinar milhares de configurações diferentes pode vir a implicar numa busca exaustiva [32, 60]. Dependendo do espaço de projeto, esse processo pode consumir muito tempo, dias ou meses, até se encontrar a melhor arquitetura ou apenas uma que atenda a todas as restrições de projeto.

1.3. Objetivos

Este trabalho tem como objetivo utilizar Modelos Lineares Generalizados (MLG) [1] [3] [4] [17] [21] para estudar a influência das características da estrutura de comunicação on-chip sobre o desempenho de plataformas MPSoC. Um segundo objetivo inclui, através dos MLGs, realizar predição, com precisão, de tempos de execução para construção do espaço de projeto com todas as alternativas de configuração de comunicação.

Para reduzir o escopo deste trabalho, o estudo de caso adota um cenário de simulação MPSoC de transmissão de *streams* (descrito detalhadamente na Seção 3).

Mais especificamente, os objetivos incluem:

- A partir do cenário descrito na Seção 3, formular modelos estatísticos de regressão linear generalizada;
- Ajustes nos modelos para estimação dos parâmetros lineares;
- Verificar a adequação, realizar estudo quanto às discrepâncias locais e realizar refinamentos nos modelos propostos;
- Por fim, realizar previsões para construção do espaço de projeto de configurações da estrutura de comunicação.

1.4. Organização do trabalho

A Seção 2 apresenta vários conceitos sobre barramentos, que serão importantes para melhor compreensão do problema e que serão úteis para formulação do modelo linear generalizado. Na seção seguinte, é apresentado o estudo de caso, o qual adota um cenário de multiprocessamento de streams em sistemas embarcados. A Seção 4 apresenta Modelos Lineares Generalizados, partindo desde os conceitos fundamentais até os componentes dos modelos. Na próxima seção, a metodologia para formulação e avaliação de MLG é explanada. As Seções 6, 7, 8 e 9 apresentam a formulação, ajustes, formadas de avaliação e refinamentos no modelo proposto. Na Seção 10 são realizados testes estatísticos para verificar a adequabilidade do modelo proposto aos dados observados. Seção 11 aborda previsões, através de observações não utilizadas para ajuste dos modelos propostos. Por fim, a Seção 12 apresenta as conclusões deste trabalho.

2. Barramentos

Um barramento é um recurso compartilhado para conectar múltiplos subsistemas [38]. Os barramentos têm baixo custo e são muito versáteis, pois permitem conectar, facilmente, novos dispositivos ao sistema. Em um dado momento, apenas um dispositivo (seja um registrador, ULA, memória, ou algum outro componente) pode utilizá-lo.

Entretanto, este compartilhamento resulta, freqüentemente, em atrasos nas comunicações. Sua velocidade é afetada por seu comprimento, assim como pelo número dos dispositivos que o compartilham [38].

Quanto à sua estrutura, barramentos podem ser ponto-a-ponto, conectando dois componentes específicos (Figura 2.1) ou podem ter um caminho comum que conecta vários dispositivos (Figura 2.2).

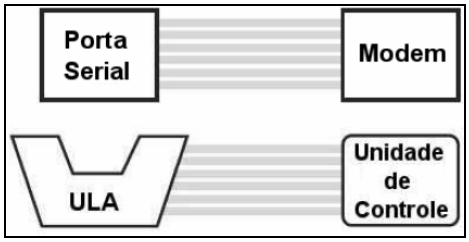


Figura 2.1 Barramentos ponto-a-ponto.

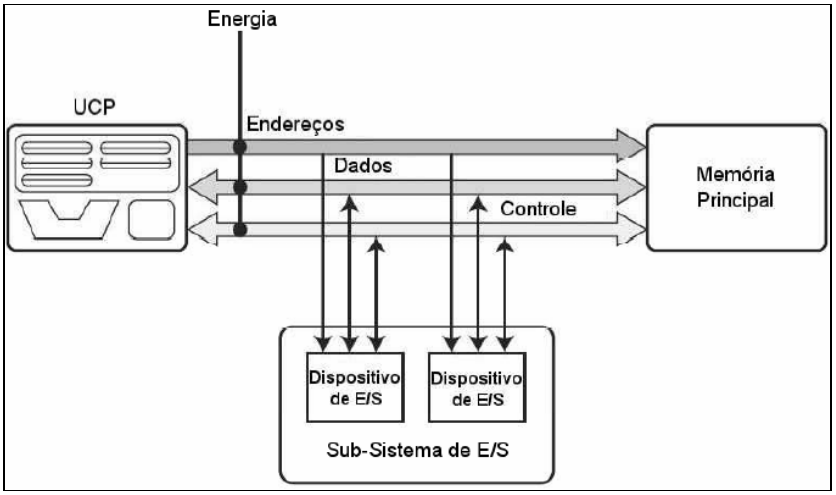


Figura 2.2 Componentes de um barramento típico.

Por causa do compartilhamento, o protocolo (conjunto de regras para uso do barramento) é muito importante. A Figura 2.2 mostra um barramento típico composto por linhas de dados, linhas de endereço, linhas de controle e linhas de energia. As linhas utilizadas para transferências de dados são chamadas, freqüentemente, de barramento de dados. Estas linhas contêm a informação atual que deve ser transferida de um local para outro. Linhas de controle indicam qual dispositivo tem permissão para utilizar o barramento, o tipo de transferência (escrita ou leitura, por exemplo), interrupções, sinais para sincronização por clock, etc. Linhas de endereço indicam o local (na memória, por exemplo) que o dado

deve ser lido ou escrito. As linhas de energia fornecem a alimentação elétrica necessária.

Barramentos podem ser classificados em síncronos e assíncronos. Síncronos são aqueles onde a seqüência de eventos de uma transmissão é controlada pelo clock. Em barramentos assíncronos, as linhas de controle coordenam a transmissão e um protocolo de *handshaking* deve ser utilizado para forçar a temporização. Para ler uma palavra da memória, por exemplo,
o protocolo deve utilizar passos similares aos citados a seguir:

1. ReqREAD: Esta linha de controle do barramento é ativada pelo dispositivo que deseja ler a palavra, e, ao mesmo tempo, o endereço de memória é colocado nas linhas apropriadas do barramento.

2. ReadyDATA: Esta linha de controle é ativada pelo sistema de memória quando coloca o dado pedido nas linhas de dados para o barramento.

3. ACK: Esta linha de controle é utilizada para indicar que o ReqREAD ou o ReadyDATA foram reconhecidos pela memória.

Devido ao fato de utilizar um protocolo de *handshake*, ao invés de clock, para coordenar transações, é possível utilizar dispositivos com períodos diferentes de clock de forma mais eficiente, tornando os barramentos assíncronos mais escalares.

Freqüentemente, os dispositivos são divididos em mestre e escravo, onde um dispositivo mestre é o que inicia as ações e um escravo é que o que responde aos pedidos do mestre.

Em sistemas com mais de um dispositivo mestre, um mecanismo de arbitragem de barramento é necessário. Esquemas de arbitragem são utilizados para tentar minimizar a contenção quando vários dispositivos mestre competem para utilizar o barramento no mesmo instante.

Tempo de contenção é definido pelo intervalo entre o instante de solicitação de uso do barramento por um mestre e o instante que a permissão é concedida.

Segundo [38], esquemas de arbitragem podem ser divididos em quatro categorias:

1. Arbitragem Daisy Chain: este esquema utiliza uma linha de controle que dá permissão ao mestre de uso do barramento. Esse sinal é passado do mestre de maior prioridade para o de menor prioridade. Este esquema de arbitragem é simples mas não é justo, pois existe a possibilidade dos mestres de menor prioridade nunca utilizarem o barramento.

2. Arbitragem Paralela Centralizada: cada mestre tem uma linha de controle para o barramento, e um árbitro centralizado seleciona quem deve utilizar o barramento. São exemplos de algoritmos deste esquema:

- **Prioridade Fixa:** cada mestre possui um valor que respresenta sua prioridade na escala de prioridades. Quando há concorrência para utilizar o barramento, o árbitro dá permissão ao de maior prioridade.

• **Round Robin:** mestres que desejam utilizar o barramento entram em uma fila de prioridades, onde o primeiro elemento da fila tem permissão para efetuar transferências. A ordenação da fila segue a ordem de solicitação de uso do barramento, ou seja, o primeiro mestre ao solicitar o uso tem permissão para efetuar transferências e os demais são postos na fila de acordo com a ordem de solicitação. Após efetuar todas as operações, o primeiro mestre da fila é removido, o segundo, que passa a ser o primeiro, recebe permissão para efetuar transferências.

• **Time Division Multiple Access (TDMA):** assim como em Round Robin, os mestres são postos em uma fila de prioridade por ordem de solicitação, porém, cada mestre possui um intervalo de tempo (quantum) específico para realizar suas operações. Após, este intervalo, o mestre que está realizando transferências é removido da primeira posição da fila e inserido no final.

3. Arbitragem Distribuída utilizando Self-Selection: este esquema é semelhante à arbitragem centralizada, mas não utiliza um árbitro central para selecionar quem deve utilizar o barramento, os próprios mestres determinam quem tem a maior prioridade.

4. Arbitragem Distribuída utilizando Detecção de Colisão: cada mestre pode fazer requisições. Se o barramento detectar qualquer colisão (múltiplas requisições simultâneas), o mestre deve fazer outra requisição após uma espera com duração variada (Protocolo Ethernet utiliza este tipo de arbitragem).

Visando aumentar a eficiência do uso do barramento, que é a quantidade de transferências por tempo de uso do barramento por um mestre, na comunicação entre dispositivos, alguns barramentos possuem diferentes tipos de transferências. Dentre estes tipos, podemos destacar os três a seguir:

• **Simples:** este tipo caracteriza as transações mais fundamentais, onde apenas uma transferência é necessária para o tráfego das informações.

• **Rajada:** é caracterizado pela seqüência de várias transferências simples e é bastante utilizado para transferência de blocos de informações. Exemplos de uso deste tipo de transferência incluem aplicações que utilizam transferência de *streams* de dados entre dispositivos ou envio de blocos de dados por processadores para preenchimento de regiões de memória.

• **Split:** é um tipo de transferência que pode ser interrompida por um periférico para ser realizada em outro momento. Normalmente, é utilizado por periféricos com estados de latência, ou seja, que não permitem atender imediatamente requisições.

2.1. Fases dos Protocolos de Barramentos

Como foi visto na seção anterior, em relação à coordenação do protocolo, existem dois tipos de barramentos: síncronos e assíncronos. Se observarmos a seqüência de eventos (sinais de barramento), em ambos, podemos dividi-la basicamente em três fases:

1. Requisição de uso do barramento: O mestre requisita ao barramento permissão para utilizá-lo. Esta permissão será concedida, ou não, pelo mecanismo de arbitragem.

2. Configuração da transação: O mestre envia ao barramento os sinais de controle. Nesta fase será configurado o modo de operação do barramento, como, por exemplo, escrita, leitura, operações de rajada, etc., e o endereço do dispositivo de onde o dado será lido ou escrito.

3. Envio de dados: A depender da operação, o mestre deverá enviar dados (operações de escrita) ou receber (operações de leitura). Esta fase pode ser seguida por uma operação de release ou não. Release caracteriza a liberação do barramento por um mestre para que os outros possam utilizá-lo. No caso de não ocorrer release o mestre continua sendo o proprietário do barramento.

Em barramentos síncronos, a seqüência dos eventos contidos nas fases citadas anteriormente é gerenciada pelo clock. Para barramentos assíncronos, poderão haver mais eventos, caracterizando o *handshaking*, mas a disposição dos eventos continua de acordo com as três fases.

Normalmente, os eventos para a fase de Requisição são basicamente três: *request*, *grant* e *busy*. O primeiro, *request*, é sinalizado pelo mestre e significa que o mesmo deseja fazer transações. Se a permissão for concedida, o barramento sinaliza *grant* para aquele mestre e sinaliza *busy* para os demais dispositivos.

Para a fase de Configuração, o mestre disponibiliza o endereço do dispositivo de/para onde se deseja fazer a transferência e, em paralelo, os sinais de controle. Os sinais de controle podem ser de vários tipos, entre eles:

• **Escrita ou leitura:** se a operação do mestre será de escrita ou leitura. Este sinal é dirigido ao escravo para que ele possa se preparar para o tipo de transferência.

• **Tipo de transferência:** dois tipos são, normalmente, suportados por barramentos:

1. não-seqüencial, para transferências simples e

2. seqüencial, para transferências por rajada.

• **Tipo de rajada:** indica ao escravo o tipo de rajada. É utilizada com dispositivos que fazem endereçamento incremental - cálculo da faixa de endereços de uma rajada a partir do incremento de um endereço inicial - ou *wrapping* - cálculo da faixa de endereços de uma rajada a partir de incremento de um endereço inicial, porém há um limite de incremento que, quando atingido, o endereçamento volta

ao endereço inicial, comportando-se como uma fila circular. Um exemplo bastante prático deste tipo de dispositivo é um sistema de
memória.

• **Tamanho do dado:** tamanho, em bits, do dado a ser transferido. Este sinal é importante para o mestre que deseja enviar um dado de tamanho menor que a largura do barramento.

• **Controle de proteção:** informações adicionais sobre acesso ao barramento. Pode ser utilizado para indicar, por exemplo, um modo de acesso privilegiado ou modo de usuário, para gerenciamento de memória por mestres, indicando se o endereço corrente é armazenável em memória cache ou não, etc.

Na última fase, Envio de dados, o barramento de dados é utilizado para transferência de dados, pelo mestre ou escravo, de acordo com a operação configurada. Quando disponível, o sinal de release é ativado pelo mestre.

3. Estudo de caso

Para o estudo de caso foi o adotado um cenário MPSoC de transmissão de *streams*, que consiste de três módulos processadores de 32-bits que enviam fluxos de dados através de um barramento compartilhado para uma memória externa (Figura 3.1).

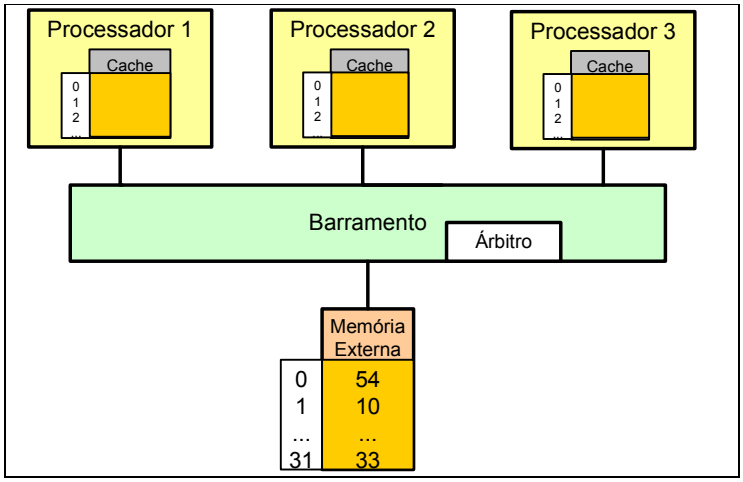


Figura 4.1 – Plataforma MPSoC para transmissão de *streams*.

Para a modelagem dos componentes processadores, barramento e memória externa foi utilizada a biblioteca de modelagem de sistemas embarcados SystemC (ver Apêndice A).

Buscando aleatoriedade nos resultados, as quantidades totais de dados dos *streams* enviados pelos módulos processadores foram definidas diferentemente: 512 bits, 256 bits e 1024 bits para os Processadores 1, 2 e 3, respectivamente.

O modelo de barramento adotado possui várias características inerentes a barramentos reais. Sendo ele síncrono e parametrizável em relação a:

- tempo de requisição de uso do barramento (*request cycle*);
- suporte a operações de *pipeline*(*pipeline cycle*);
- largura do barramento de dados (*bus width*) e
- suporte a operações simples e rajada, esta com diversos tamanhos (*transfer size*).

O mecanismo de arbitragem do barramento é paralelo centralizado com protocolo orientado à prioridade fixa, permitindo aos mestres possuir três níveis de prioridades diferentes e exclusivos.

A memória externa possui tamanho de 1,2 MB e é utilizada para armazenar as aplicações que serão executadas pelos módulos processadores e para armazenar os *streams* enviados.

3.1 Simulações

Como parâmetros do barramento foram utilizados:

rc = tempo de requisição de uso do barramento, variando entre {1, 2, 3} (em microssegundos, ms),

pc = uso ou não de operações com *pipeline*, variando entre {1, 2} (1 indica presença de *pipeline* e 2 ausencia de *pipeline*),

bw = largura do barramento, variando entre {8, 16, 32} (em bits),

ts = Tamanho da transferência, variando entre {1, 4, 8, 16} , onde {1} representa uma transferência simples e {4, 8, 16} representa o tamanho de transferências para compor uma transferência de rajada.

Como parâmetros de configuração do mecanismo de arbitragem, utilizaram-se:

m1p = Prioridade do Processador 1, variando entre {1, 2, 3}

m2p = Prioridade do Processador 2, variando entre {1, 2, 3}

m3p = Prioridade do Processador 3, variando entre {1, 2, 3}

A partir de todos esses parâmetros foram realizadas 432 simulações, com média de cinco minutos para configuração e um minuto e meio para simulação. Ao todo, foram necessárias aproximadamente 47 horas para realizar todas as simulações.

4. Modelos Lineares Generalizados

O Modelo Linear Generalizado (MLG) é um método baseado na técnica de Máxima Verossimilhança [1] pela qual parâmetros do modelo de regressão podem ser estimados quando o modelo da estrutura de erro pertence à família exponencial de distribuições [2]. Os parâmetros são, então, utilizados para calcular os valores esperados da variável resposta e, assim, os resíduos. A homogeneidade dos resíduos é avaliada e utilizada como critério para determinar se o modelo da estrutura de erros é apropriado [2].

Verossimilhança é definida como o produto das probabilidades da observação de cada valor da variável resposta. Para distribuições contínuas, como a distribuição normal e a gama, as funções de densidade são utilizadas no lugar da probabilidade. É usual considerar o logaritmo da verossimilhança desde que ofereça cálculos mais tratáveis (e qualquer máximo da verossimilhança é também máximo da log-verossimilhança). Estimação de máxima verossimilhança, na prática, procura identificar os valores dos parâmetros que maximizem esta log-verossimilhança. [3]

Um modelo linear generalizado possui três componentes. O primeiro é o componente aleatório ***Y***, que é um vetor de observações de *y*, tendo *n* componentes que são independentemente distribuídos pertencentes à uma das distribuições da família exponencial.

O segundo é o componente sistemático, que é a especificação para o vetor $\boldsymbol{\mu}$ em termos de um pequeno número de parâmetros desconhecidos $\beta_1, \beta_2, \ldots, \beta_p$. Um preditor linear $\boldsymbol{\eta}$ é dado por

$$\eta = \sum_{j=1}^{p} X_j \beta_j + \xi_j$$

onde ***X*** é o vetor de variáveis sistemáticas para as observações ***Y*** e $\boldsymbol{\xi}$ é o vetor de erros, também chamados de *offsets*, associados às observações.

O terceiro componente é a ligação entre o componente aleatório e os componentes sistemáticos. Essa ligação é frequentemente escrita como $\eta = g(\mu)$ ou $\mu = g^{-1}(\eta)$, onde *g* é a função de ligação [2]. A tabela a seguir ilustra as funções de ligação mais comuns:

Tabela 3.2 – Funções de ligação

Nome	g(x)	$g^{-1}(x)$
Identidade	x	x
Log	ln(x)	e^x
Logit	ln(x/(1 - x))	$e^x/(1 + e^x)$
Recíproca	1/x	1/x

Modelos de regressão linear clássicos têm uma distribuição normal no primeiro componente e a função de ligação identidade para o terceiro componente.

Formalmente, a família exponencial é um famílias com dois parâmetros [3], definida como

$$f_i(y_i;\theta_i,\phi) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{a_i(\phi)} + c(y_i,\phi)\right\}$$

onde $a_i(\phi)$, $b(\theta_i)$ e $c(y_i, \phi)$ são funções conhecidas; θ_i é um parâmetro relacionado à média; e ϕ é um parâmetro relacionado à variância. Para fins práticos, é útil saber que um membro da família exponencial possui as seguintes propriedades:

1. a distribuição é completamente especificada em termos de sua média e variância,
2. a variância de Y_i é uma função de sua média.

A segunda propriedade é enfatizada expressando-se a variância como

$$Var(Y_i) = \frac{\phi.V(\mu_i)}{\omega_i}$$

onde V(.), chamada de função de variância, é uma função especificada; o parâmetro ϕ refere-se à variância; e ω_i é uma constante que atribui peso, ou credibilidade, à observação i.

Várias distribuições conhecidas pertencem à família exponencial, como exemplo: normal, Poisson, binomial, gama, e Gaussiana inversa. Os valores correspondentes das funções de variância são ilustrados na Tabela 3.1.

Tabela 3.1 – Distribuições e respectivas funções de variância.

Distribuição	V(x)
Normal	1
Poisson	x
Gama	x^2
Binomial	x(1-x)
Gaussiana Inversa	x^3

Cada estrutura de erro é associada a uma função de ligação chamada "canônica", que simplifica a matemática para solução analítica de MLGs. Quando utilizando software de computadores modernos, entretanto, o uso de funções de ligação canônicas não é importante e qualquer par de funções de ligação e de variância pode ser selecionado. A Tabela 3.3 ilustra a função de ligação canônica para algumas distribuições de erro.

Tabela 3.3 – Distribuições e respectivas funções de ligação canônica.

Distribuição	Função de Ligação Canônica
Normal	$\eta = \mu$
Lognormal	$\eta = \mu$
Gama	$\eta = 1/\mu$
Poisson	$\eta = \log(\mu)$

5. Metodologia

De acordo com Benini e Micheli em [4], na prática o processo de trabalho com os MLG pode ser dividido em três etapas: (i) formulação dos modelos; (ii) ajuste dos modelos e (iii) inferência.

Os MLG formam um ferramental de grande utilidade prática, pois apresentam grande flexibilidade na etapa (i), computação simples em (ii) e critérios razoáveis em (iii). Essas etapas são realizadas sequencialmente. Na análise de dados complexos, após a realização da etapa de inferência, pode-se voltar à etapa (i) e escolher outros modelos, a partir de informações mais detalhadas obtidas do estudo feito em (iii).

A primeira etapa, formulação de modelos, compreende a escolha de opções para a distribuição de probabilidades da variável resposta (componente aleatório), covariáveis (variáveis sistemáticas) e função de ligação. Estas opções visam a descrever as características principais da variável resposta.

A etapa de ajuste representa o processo de estimação dos parâmetros lineares dos modelos e de determinadas funções das estimativas desses parâmetros, que representam medidas de adequação dos valores estimados. Vários métodos podem ser usados para estimar os parâmetros dos MLG. Como o método de máxima verossimilhança nos MLG conduz a um procedimento de estimação bastante simples, esse método é o mais utilizado.

Por fim, a etapa de inferência tem como objetivo principal verificar a adequação do modelo como um todo e realizar um estudo detalhado quanto a discrepâncias locais. Essas discrepâncias, quando significantes, podem implicar na escolha de outro modelo, ou em aceitar a existência de dados aberrantes. Em qualquer caso, toda a metodologia de trabalho deverá ser repetida.

O analista deve, nessa etapa, verificar a precisão e a interdependência das estimativas, construir regiões de confiança e testes sobre os parâmetros de interesse, analisar estatisticamente os resíduos e realizar previsões.

5.1 Ferramentas

As ferramentas utilizadas para apoio às análises, testes e geração de gráficos estatísticos, contidos nas fases da metodologia, foram:

- **Minitab** versão 14 [13];
- **R** versão 2.6.1 [37].

6. Formulação dos modelos

6.1 Identificação das variáveis

O primeiro passo na construção dos MLGs é definição da variável aleatória e identificação as variáveis sistemáticas.

O componente aleatório (Vetor ***Y***) foi definido, de acordo com os objetivos traçados na Subseção 1.3, como os tempos de execução obtidos a partir das configurações da arquitetura de comunicação *on-chip* do modelo de plataforma MPSoC apresentado na Seção 4.

Para as variáveis sistemáticas (vetor ***X***), de acordo com o estudo apresentado na Seção 2, existem muitas características nas arquiteturas de comunicação on-chip que podem ser empregadas para formulação do MLG. Porém, para fins de simplificação, serão extraídas as características parametrizáveis do modelo de comunicação on-chip do cenário apresentado na Seção 3. A Tabela 6.1 mostra as variáveis candidatas ao modelo e suas respectivas descrições.

Tabela 6.1 – Variáveis candidatas ao modelo.

Componente aleatório	**Descrição**
st	tempo de execução da plataforma
Variáveis sistemáticas	**Descrição**
rc	tempo de requisição de uso do barramento
pc	uso ou não de operações com *pipeline*
bw	largura do barramento
ts	tamanho da transferência
m1p	prioridade do Processador 1
m2p	prioridade do Processador 2
m3p	prioridade do Processador 3

6.2 Multicolinearidade

O primeiro estudo a ser realizado sobre as variáveis preditoras é a relação de multicolinearidade.

O termo multicolinearidade (ou colinearidade) refere-se a uma relação aproximadamente linear entre variáveis explicativas (sistemáticas), em outras palavras, quando há forte correlação entre preditores. De acordo com Dobson [17], esta condição tem várias conseqüências indesejáveis. Hocking [19] cita algumas destas: as estimativas $\boldsymbol{\beta}$ podem ser maiores, em magnitude, do que foi previsto, uma estimativa pode ser negativa, contradizendo a regra prevista do preditor, e nenhuma das variáveis pode ser significante apesar da estatística *F* [20], dada pela regressão.

Segundo Hocking [19] uma forma gráfica de identificar relações lineares entre preditores é realizada através da Matriz de Dispersão (Matrix Plot). As

variáveis são arranjadas em uma matriz de forma que em uma dada linha seja possível encontrar os gráficos de dispersão das variáveis da linha contra as variáveis da coluna. A vantagem de apresentar os gráficos em uma matriz é a facilidade de localizar comportamentos incomuns de casos individuais ou outros padrões. A matriz de dispersão para os dados das simulações pode ser vista na Figura 5.1, onde apenas a porção triangular inferior da matriz é mostrada.

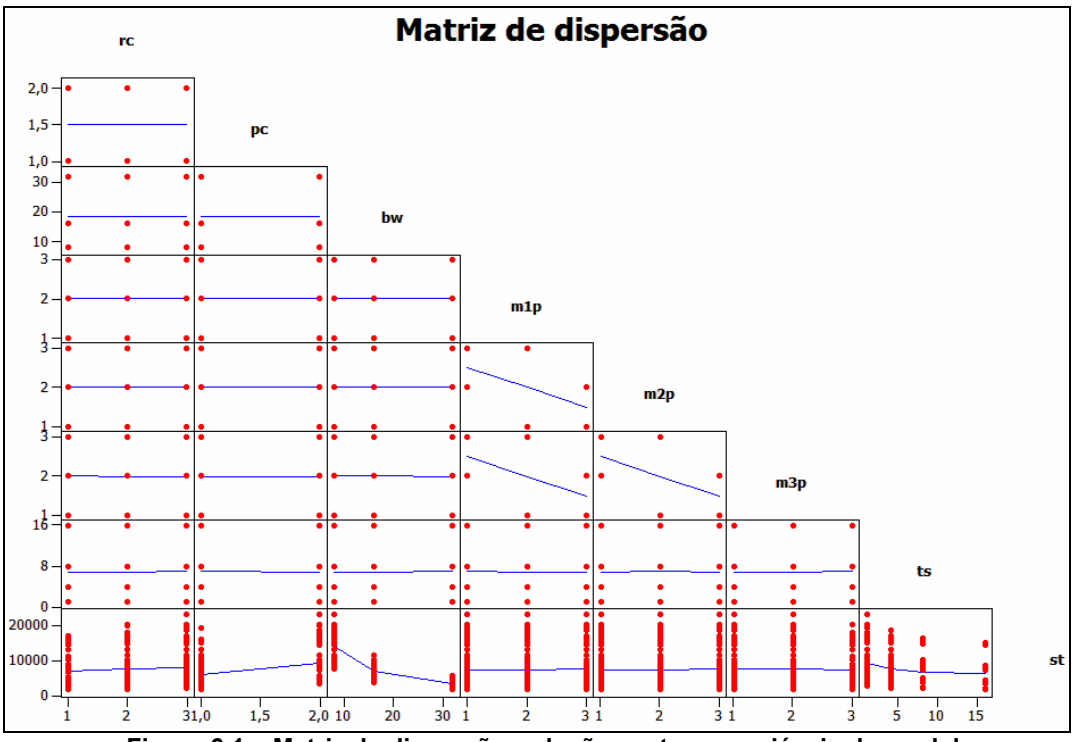


Figura 6.1 – Matriz de dispersão: relações entre as variáveis do modelo.

Pode-se verificar na Figura 5.1 que a maioria dos gráficos de dispersão da matriz mostra relações constantes (curvas horizontais) entre as variáveis preditoras, exceto entre m1p, m2p e m3p. Destas, observa-se também que a correlação é negativa, visto que as curvas decrescem com o aumento dos valores das variáveis.

Para qualificar a intensidade das relações lineares notadas na matriz de dispersão, são computados os coeficientes de correlação para cada par de variáveis. O coeficiente de correlação pode ser dado por

$$r = \frac{n.\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}}$$

onde x e y são as variáveis analisadas.

A Tabela 6.2 mostra a Matriz de Correlação, a qual apresenta os coeficientes de correlação entre todas as variáveis candidatas ao modelo.

Tabela6.2- Matriz de correlação: coeficientes de correlação entre variáveis do modelo.

	Rc	Pc	Bw	Ts	m1p	m2p	M3p	St
rc	1	0	0	0	0	0	0	0,121
pc	0	1	0	0	0	0	0	0,336
bw	0	0	1	0	0	0	0	-0,791
ts	0	0	0	1	0	0	0	-0,229
m1p	0	0	0	0	1	-0,5	-0,5	0,027
m2p	0	0	0	0	-0,5	1	-0,5	0,028
m3p	0	0	0	0	-0,5	-0,5	1	-0,055
st	0,121	0,336	-0,791	-0,229	0,027	0,028	-0,055	1

Pode-se constatar na última linha da Tabela 6.2 que existe correlação linear entre a variável resposta (st) e as variáveis preditoras (rc, pc, bw, ts, m1p, m3p, m3p). É possível verificar, também na mesma tabela, que existe correlação negativa entre m1p, m2p e m3p.

Segundo Weisberg [7], quando um conjunto de preditores são exatamente colineares ou possuem colinearidade aproximada, um ou mais preditores devem ser removidos, de forma que a perda de informação sobre os valores ajustados esperados seja mínima.

6.3 Seleção de Variáveis para o MLG

A seleção de variáveis refere-se ao processo de busca de variáveis, dentre as opções, que ajustam o melhor modelo. O objetivo é dividir $\boldsymbol{X}$ em um conjunto de termos ativos $\boldsymbol{X_A}$ e um conjunto de termos inativos $\boldsymbol{X_I}$.

Para tanto, existem dois aspectos que devem ser abordados. Primeiro, dado um candidato particular $\boldsymbol{X_C}$ para termos ativos, que critério deve ser utilizado para comparar $\boldsymbol{X_C}$ com outras possíveis escolhas de $\boldsymbol{X_A}$? o segundo é computacional: como tratar o potencial alto número de comparações que precisam ser realizadas?

As próximas duas subseções descrevem com mais detalhes estes aspectos.

6.3.1 Critérios de informação

Segundo Weisberg em [7], critérios para comparar vários subconjuntos de candidatos são baseados na falta de ajuste do modelo e sua complexidade. Falta de ajuste do modelo é medido pela Soma dos Quadrados Residuais (SQR). Complexidade é medida pelo número de termos *tc* em $\boldsymbol{X_C}$, incluindo o intercepto.

O critério mais comum é o *Akaike Information Criterion* (AIC) [8], o qual foi adotado neste trabalho, é definido pela equação a seguir

$$AIC = -2.MLL + 2.tc$$

onde MLL é a máxima log-verossimilhança. Pequenos valores são preferidos, assim melhores conjuntos candidatos possuem maior *MLL* e menor número de termos *tc*.

Duas alternativas para o AIC são o *Bayes Information Criterion* (BIC) [10] e o *Malow's* C_p [11].

6.3.2 Critérios computacionais

Além dos critérios de informação, Validação Cruzada (*Cross-Validation*) pode, também, ser utilizada para comparar subconjuntos candidatos de funções da média (modelo ajustado).

O tipo mais direto de validação cruzada é dividir os dados em duas partes aleatórias, chamadas de *conjunto de construção* e *conjunto de validação*. O conjunto de construção é utilizado para estimar os parâmetros da função da média. Valores ajustados desta função da média são então computados para os casos do conjunto de validação, e a média da diferença dos quadrados da resposta e os valores ajustados para o conjunto de validação é utilizada como um resumo do ajuste do conjunto

Neste trabalho, os conjuntos de construção e validação abrangem aproximadamente 80% e 20%, respectivamente, do total de casos (432 casos) obtidos das simulações.

Existem outras técnicas de validação cruzada que podem ser utilizadas neste mesmo contexto, como a técnica *PRESS* [12].

6.4 Métodos Computacionais

Para problemas que não sejam de regressão linear por mínimos quadrados [7], ou se validação cruzada é utilizada como uma função de critério de seleção, métodos exaustivos são geralmente inviáveis e o compromisso computacional é necessário.

Métodos como *stepwise* [7] e *best subsets* [15] examinam somente alguns subconjuntos de cada tamanho. A estimativa de $\boldsymbol{X_A}$ é então selecionada a partir de alguns subconjuntos que foram examinados. Estes métodos não garantem encontrar o candidato que é ótimo de acordo com a função de critério, mas eles frequentemente fornecem resultados úteis na prática.

Este trabalho utilizou critérios informação e computacionais, bem como *stepwise* para seleção de variáveis nos refinamentos dos modelos propostos.

6.5 Transformações

Em modelos lineares clássicos, analistas procuram, algumas vezes, transformar os dados para satisfazer requisitos de normalidade e variância constante.

Para MLGs a transformação da variável independente é dada pela função de ligação. Os pontos importantes observados são: a simetria das distribuições e

a ausência de *outliers*. Em muitos casos, a transformação resolve problemas residuais, como heterocedasticidade (ver Seção xxx).

Para identificar a função de ligação para transformação mais adequada para o modelo proposto, foi utilizada a técnica de transformação *Box-Cox* [22].

Seja y a variável original e f(y) a função de transformação dada por

$$f(y) = \begin{cases} \dfrac{y^{\lambda} - 1}{\lambda}, \lambda \neq 0 \\ \log y, \lambda = 0 \end{cases}$$

Se λ=1 então f(y) = y – 1, que é equivalente a uma função de ligação identidade com um deslocamento ao nível da base. Já se λ = -1 então f(y) = 1 – y^{-1}, que é equivalente a uma função de ligação inversa com um deslocamento ao nível de base.

Pelo ajuste de séries de MLGs para dados com diferentes valores λ (incluindo valores reais entre -1 e 0, bem como entre 0 e 1), é possível estimar que valor de λ é mais adequado, que produz a distribuição mais próxima da normal, ao conjunto de dados em questão observando qual valor de λ fornece a maior log-verossimilhança.

Para os dados de simulação, de acordo com o gráfico apresentado na Figura 5.2, é possível observar que a maior log-verossimilhança é encontrada em λ = 0.

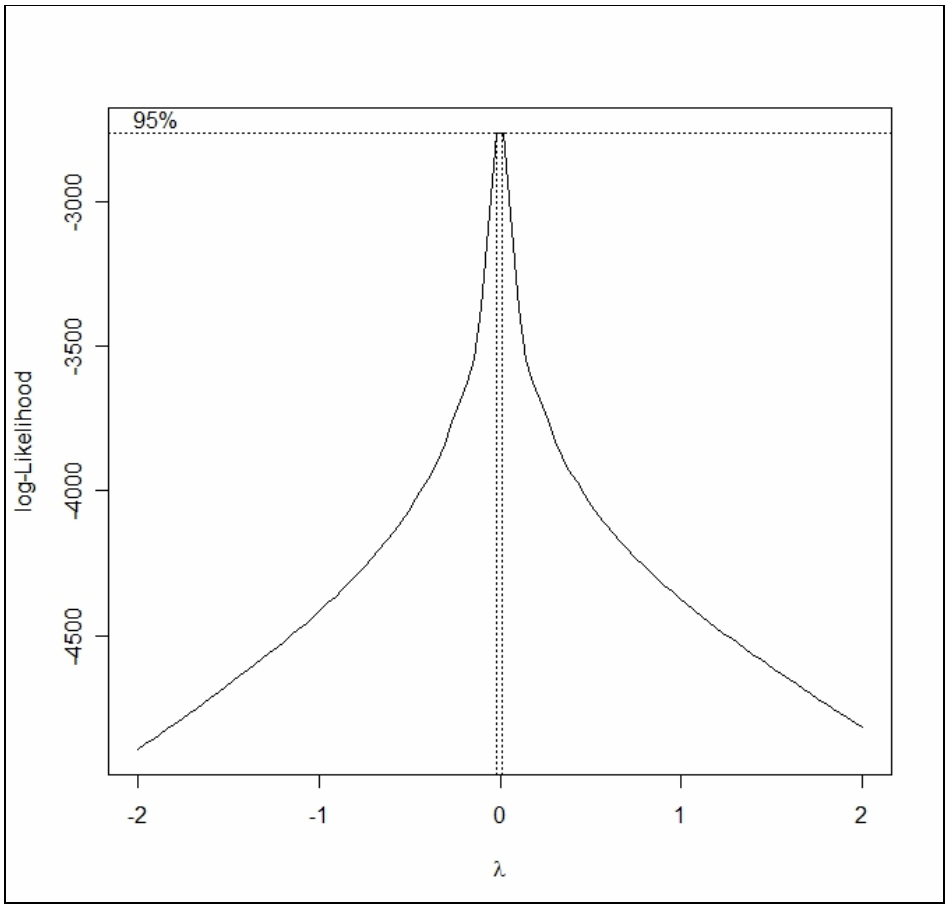


Figura 6.2 – Gráfico com resultados das transformações Box-Cox.

Assim, com λ = 0, a função de ligação logarítmica pode ser adotada.

Outra forma de realizar transformações em MLGs, envolve utilizar a função de transformação para modificar a escala dos valores da variável aleatória e, então, utilizar a função identidade como nova função de ligação. Assim, temos:

$$f(y) = g(\eta)$$

onde f(y) é a função de transformação da escala da variável aleatória y e g(η) é a função de ligação identidade.

Por fim, aplicando a função de transformação logarítmica em st e utilizando a função de ligação identidade, temos o modelo dado por:

$$\log(st) = \eta$$

6.6 Fatores

Algumas vezes é necessário atribuir categorias ou classificações a variabilidade evidente em dados. Por exemplo, neste estudo, as simulações podem ser classificadas pela presença ou ausência de operações com *pipeline*. Em outro exemplo, os processadores podem executar suas aplicações em três níveis de prioridade diferentes.

Estas duas variáveis são chamadas fatores [21]. As classes individuais de uma classificação são chamadas de *níveis de um fator* ou *classes da variável dummy*. Na classificação de dados em termos de fatores e seus níveis, a característica importante observada é a extensão de quais níveis diferentes de um fator podem influenciar a variável de interesse. [21]

Observando as variáveis preditoras candidatas ao modelo, é possível verificar que todas podem ser classificadas como fator, dado que assumem os valores discretos a seguir:

Tabela 6.3 – Fatores candidatos e respectivos níveis

Fator	Níveis
rc	1, 2, 3
pc	1, 2
bw	8, 16, 32
ts	1, 4, 8, 16
m1p	1, 2, 3
m2p	1, 2, 3
m3p	1, 2, 3

6.6.1 Representação de fatores

Esta discussão foi iniciada por quanto a média dos tempos de simulação variam em função das categorias de suas variáveis preditores (fatores). É desejável escrever uma função da média que permita que para cada nível dos fatores tenha sua própria média.

Fatores são frequentemente representados por variáveis *dummy* [3]. Seja D um fator com cinco níveis, a j-ésima variável dummy U_j para o fator, j=1,...,5 possui o i-ésimo valor u_{ij}, para i=1,...,n, dado por

$$u_{ij} = \begin{cases} 1, \text{se } \mathrm{D}_i = \text{j - ésima categoria de D} \\ 0, \text{do contrário.} \end{cases}$$

Se um fator tem três níveis, por exemplo, e o tamanho da amostra n=7 com os casos 1,2 e 7 do primeiro nível do fator, casos 4 e 5 na segundo nível e casos 3 e 6 do terceiro nível, assim a árvore de variáveis dummy seria:

U1	U2	U3
1	0	0
1	0	0
0	0	1
0	1	0
0	1	0
0	0	1
1	0	0

Se essas três variáveis são inseridas com fator aditivo no modelo, a média será dada apenas por uma das colunas. Esta é uma característica importante de um conjunto de variáveis *dummy* para um fator: sua soma sempre converge para o mesmo valor, normalmente um para cada caso, visto que cada caso possui somente um nível do fator.

Pode-se, então, escrever a função da média como

$$E(y|D) = \beta_1 U_1 + \beta_2 U_2 + \beta_3 U_3$$

Esta função aparenta não incluir um intercepto. Desde que a soma de U_j é dada por apenas umas das colunas, o intercepto é implícito. Desde que comum ter o intercepto incluso explicitamente, pode-se deixar de fora do modelo uma das variáveis dummy, utilizando-se a *regra de fator* [7]:

> *Um fator com d níveis pode ser representado por pelo menos d variáveis dummy. Se o intercepto está na função da média, pelo menos d – 1 variáveis dummy podem ser utilizadas na função da média.*

Uma escolha comum é remover a primeira variável dummy, assim a função da média será dada por

$$E(y|D) = \eta_0 + \eta_2 U_2 + \eta_3 U_3$$

onde foram mudados os nomes dos parâmetros, de β para η, porque agora obtêm-se valores diferentes.

7. Ajuste dos modelos

7.1 Estimação dos parâmetros lineares

A partir do estudo realizado na seção anterior, foi possível identificar os seguintes modelos, com os respectivos AIC, obtidos a partir do método de máxima verossimilhança, utilizando as distribuições Gaussiana e gama para modelar a estrutura de erro:

Tabela 7.1 – Relação dos MLGs e respectivos AIC, obtidos com

Modelo	Equação	AIC - Gaussiana	AIC- Gama
M1	log(st) ~ rc + pc + bw + ts + m1p	-637,33	-643,07
M2	log(st) ~ rc + pc + bw + ts + m2p	-636,15	-641,88
M3	log(st) ~ rc + pc + bw + ts + m3p	-691,76	-698,14
M4	log(st) ~ rc + pc + bw + ts + m1p + m2p	-688,05	-692,43
M5	log(st) ~ rc + pc + bw + ts + m1p + m3p	-688,05	-692,43
M6	log(st) ~ rc + pc + bw + ts + m2p + m3p	-688,05	-692,43

Dentre os modelos candidatos, foi escolhido o terceiro modelo (M3), contido na Tabela 7.1, o qual obteve melhor índice AIC, tanto para distribuição Gaussiana quanto para gama, além de ser um dos com menor número de termos *tc*.

A Figura 7.1 ilustra os resultados obtidos da regressão do terceiro modelo com distribuição Gaussiana para a estrutura de erro.

```
Call:
glm(formula = log(st) ~ rc + pc + bw + m3p + ts, family = gaussian(link = "identity"))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.18143  -0.04252  -0.01775   0.03371   0.47264

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)
(Intercept)  9.4883134  0.0161584  587.207  < 2e-16 ***
rc2          0.0970558  0.0114936    8.444 9.48e-16 ***
rc3          0.1870664  0.0114936   16.276  < 2e-16 ***
pc2          0.4511324  0.0106196   42.481  < 2e-16 ***
bw16        -0.6916081  0.0136544  -50.651  < 2e-16 ***
bw32        -1.3805206  0.0136544 -101.105  < 2e-16 ***
m3p2         0.0006681  0.0117080    0.057    0.955
m3p3        -0.0868261  0.0117080   -7.416 1.00e-12 ***
ts4         -0.2457612  0.0132143  -18.598  < 2e-16 ***
ts8         -0.3983106  0.0132143  -30.142  < 2e-16 ***
ts16        -0.4890862  0.0133383  -36.668  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.00759591)

    Null deviance: 101.2109  on 344  degrees of freedom
Residual deviance:   2.5370  on 334  degrees of freedom
AIC: -691.76

Number of Fisher Scoring iterations: 2
```

Figura 7.1 - Resultados da regressão para o modelo log(st) ~ rc + pc + bw + ts + m3p com erro Gaussiano.

Na Figura 7.1 pode-se observar os coeficientes para cada nível dos fatores (coluna *Estimate*), o erro padrão para cada coeficiente (coluna Std. Error), estatística *t-student* (coluna *t value*), a estatística *p* (coluna *Pr(>|t|)*) e o critério AIC.

8. Inferência

8.1 Estatística p para redução do modelo

Na Figura 7.1 pode-se observar que a estatística *p* para o nível 2 de m3p (m3p2) tem valor muito próximo de 1 (0,955), sinalizando que este fator não é significante para o MLG.

A partir de uma nova regressão, removendo-se o segundo nível de m3p do modelo (este novo modelo agora é chamado de M3.1) , obteve-se novos resultados, apresentados na Figura 8.1.

```
Call:
glm(formula = log(st) ~ rc + pc + bw + I(m3p == 3) + ts, family = gaussian(link = "identity"))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.18112  -0.04286  -0.01773   0.03404   0.47268

Coefficients:
                  Estimate Std. Error  t value Pr(>|t|)
(Intercept)       9.488652   0.015005  632.385  < 2e-16 ***
rc2               0.097056   0.011476    8.457  8.6e-16 ***
rc3               0.187066   0.011476   16.300  < 2e-16 ***
pc2               0.451175   0.010578   42.654  < 2e-16 ***
bw16             -0.691587   0.013629  -50.744  < 2e-16 ***
bw32             -1.380542   0.013629 -101.295  < 2e-16 ***
I(m3p == 3)TRUE -0.087186   0.009853   -8.849  < 2e-16 ***
ts4              -0.245761   0.013195  -18.626  < 2e-16 ***
ts8              -0.398311   0.013195  -30.187  < 2e-16 ***
ts16             -0.489072   0.013316  -36.728  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.00757331)

    Null deviance: 101.2109  on 344  degrees of freedom
Residual deviance:   2.5371  on 335  degrees of freedom
AIC: -693.76

Number of Fisher Scoring iterations: 2
```

Figura 8.1 - Resultados da regressão para o modelo M3.1.

Verifica-se, agora, que os valores dos coeficientes mudaram, porém, de acordo com a estatística *p*, todos os níveis dos fatores utilizados são bastante significativos (estatísticas p muito próximas de zero) para o modelo. Observa-se, também, que o critério AIC diminuiu de -691,76 para -693,76, indicando que o segundo modelo se ajusta sensivelmente melhor aos dados.

8.2 Análise residual

De acordo com Anderson et al. em [3], existem testes diagnósticos que permitem avaliar se as suposições sobre um modelo são apropriadas. Testes que ajudam nesta investigação incluem:

- resíduos, para avaliar se a estrutura do erro é apropriada.

- influências, que identificam observações que possuem excessiva influência sobre o modelo;
- transformações Box-Cox, que examinam as funções de ligação mais apropriadas (ver Seção 6.5).

8.2.1 Resíduos

Resíduos podem ser utilizados para explorar a adequação do valor ajustado de um modelo, em relação à: escolha da função de variância, função de ligação e

termos no preditor linear. [1] Resíduos podem indicar a presença de valores anormais, neste caso, necessitando, então, de mais investigação.

Para modelos lineares clássicos, é possível expressar a variável aleatória na forma

$$y = \mu + (y - \mu)$$

onde *y* é a variável aleatória, *μ* é o valor ajustado e *y – μ* é o resíduo linear.

Para MLGs, é necessária uma definição estendida de resíduos, aplicável a todas as distribuições que podem substituir a normal. Existem vários tipos de resíduos, entre eles: *Pearson*, *Anscombe* e *Resíduos de Desvio*.[1]

Neste trabalho foram utilizados resíduos de *Anscombe* quando a estrutura de erro segue a distribuição gama. *Anscombe* propõe uma transformação na distribuição dos resíduos de forma a fazer a distribuição na escala transformada o mais normal possível. Para gama, a função de transformação pode ser definida por

$$r_P = \frac{3(y^{1/3} - \mu^{1/3})}{\mu^{1/3}}$$

8.2.2 Análise gráfica residual

Para o diagnóstico residual, basicamente podem ser utilizados os seguintes gráficos:

- (1) Q-Q Plots e (2) Histogramas, para verificar suposições sobre a distribuição de probabilidade do erro;
- (3) Diagrama de dispersão dos resíduos contra os valores ajustados, para verificar não-linearidade e não-constância de variância;
- (4) Diagrama de dispersão dos resíduos contra o tempo ou ordem de coleta, para verificar se existem correlação entre os erros;
- (5) Diagrama de dispersão ou Diagrama de séries temporais das distâncias de *Cook* [27] contra a ordem de coleta, para verificar a presença de pontos de influência.

As Figuras 8.2 e 8.3 mostram os gráficos (1), (2), (3) e (4) para o modelo M3.1 com distribuições Gaussiana (resíduo linear) e gama (resíduo de *Anscombe*) para a estrutura de erro.

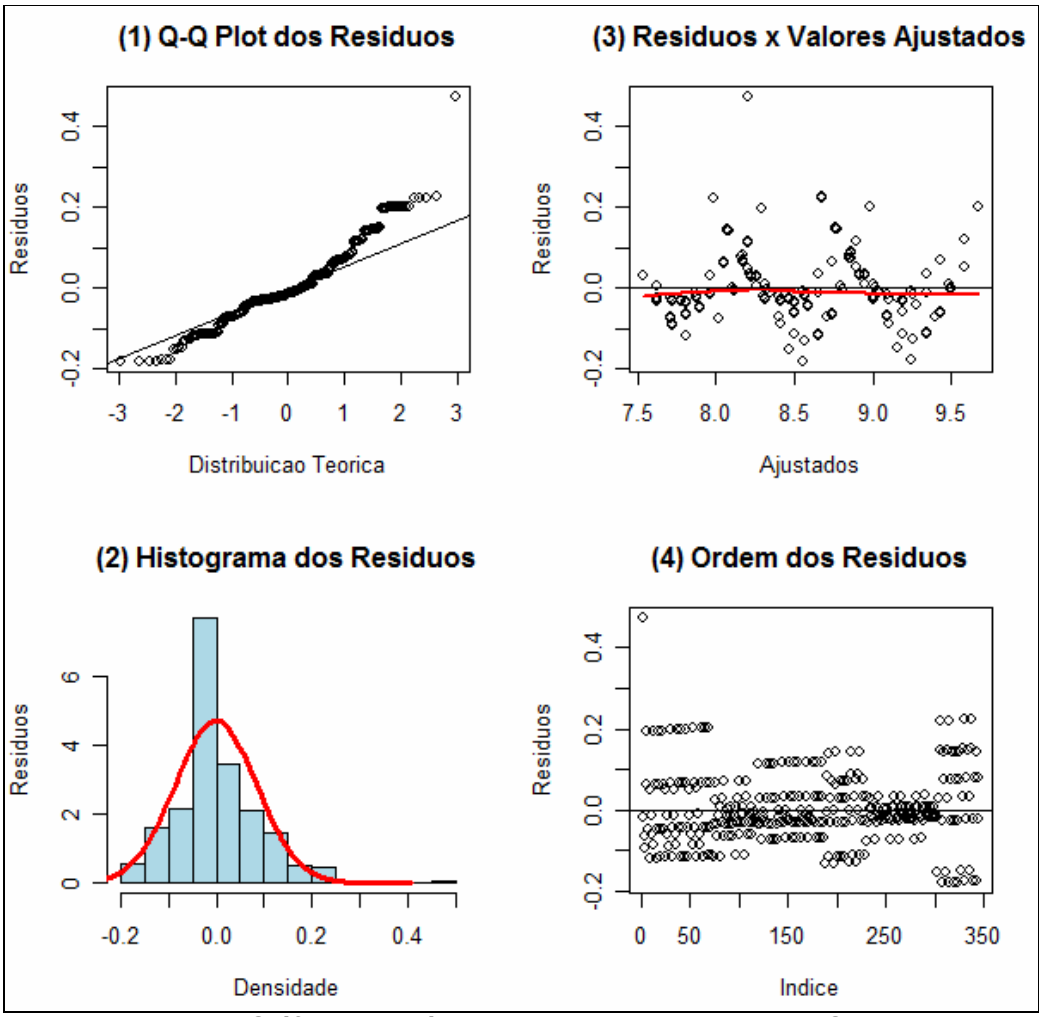


Figura 8.2 – Gráficos de análise residual para M3.1 com erro Gaussiano.

Observando os gráficos (1) e (2) nas Figura 8.2 e 8.3 observa-se que a distribuição dos resíduos não segue uma normal, visto que em (1) os pontos não seguem a linha e em (2) existem uma tendência, dos dados, à esquerda do escore 0.

Para os gráficos (3) é verificado que existe um padrão, curva que se repete ao longo dos valores ajustados. Este padrão indica não-linearidade ou curvatura, sugerindo que a função da média foi especificada incorretamente [7]. Observa-se também que a variância, distância entre os pontos, aumenta com o aumento dos valores ajustados, indicado por curvas mais grossas, caracterizando variância não-constante.

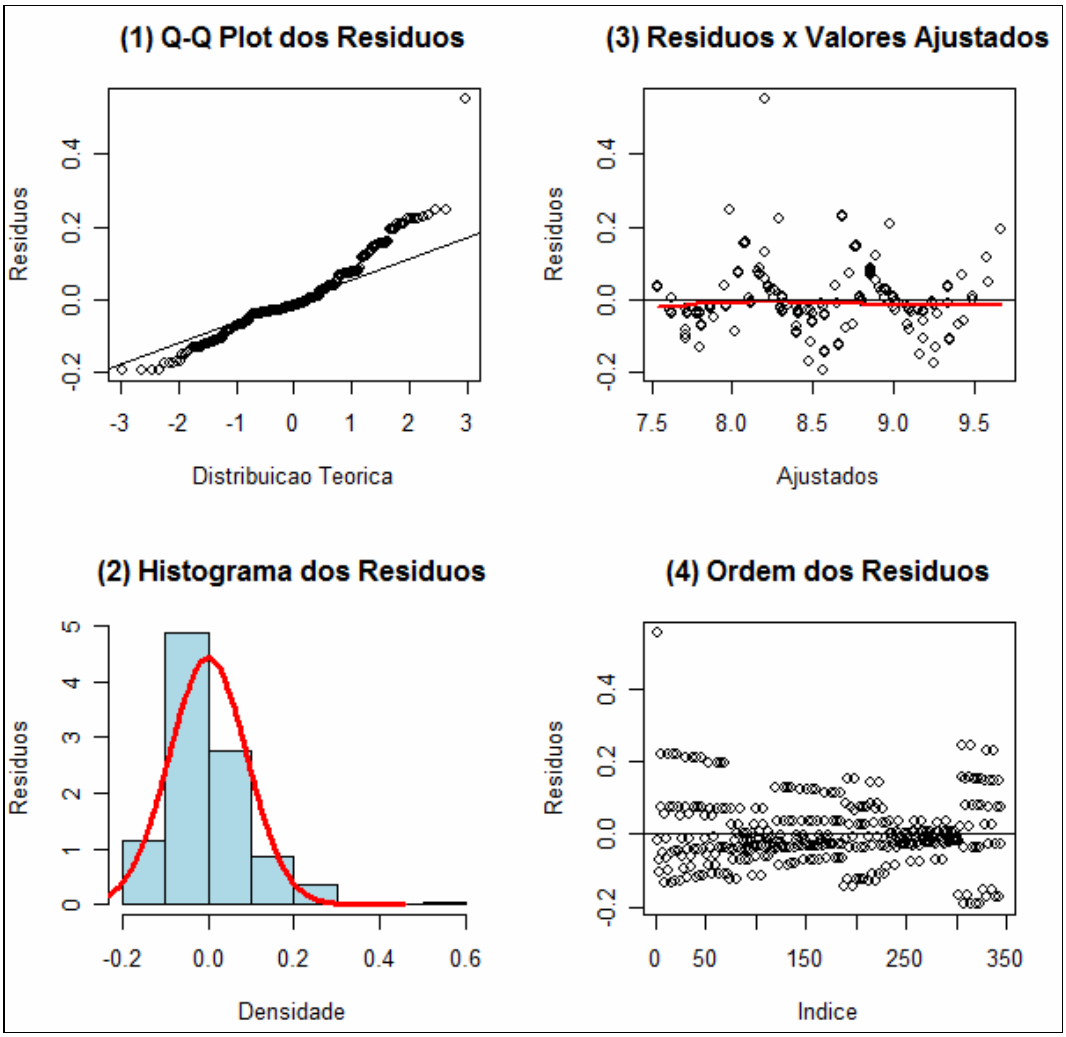


Figura 8.3 – Gráficos de análise residual para M3.1 com erro gama.

Os gráficos (4) não apresentam nenhum padrão linear entre pontos subseqüentes, o que classifica a não-correlação entre os erros.

Para os erros gaussianos e gama os gráficos de análise residual apresentaram resultados semelhantes, assim deste ponto no trabalho em diante, será dada mais ênfase nas análises sobre erros gaussianos.

Ainda observando esses gráficos, constata-se que existe um ponto distante dos demais, sendo classificado como candidato a ponto de influência. Isto é verificado no gráfico das distâncias de *Cook* contido na Figura 8.4.

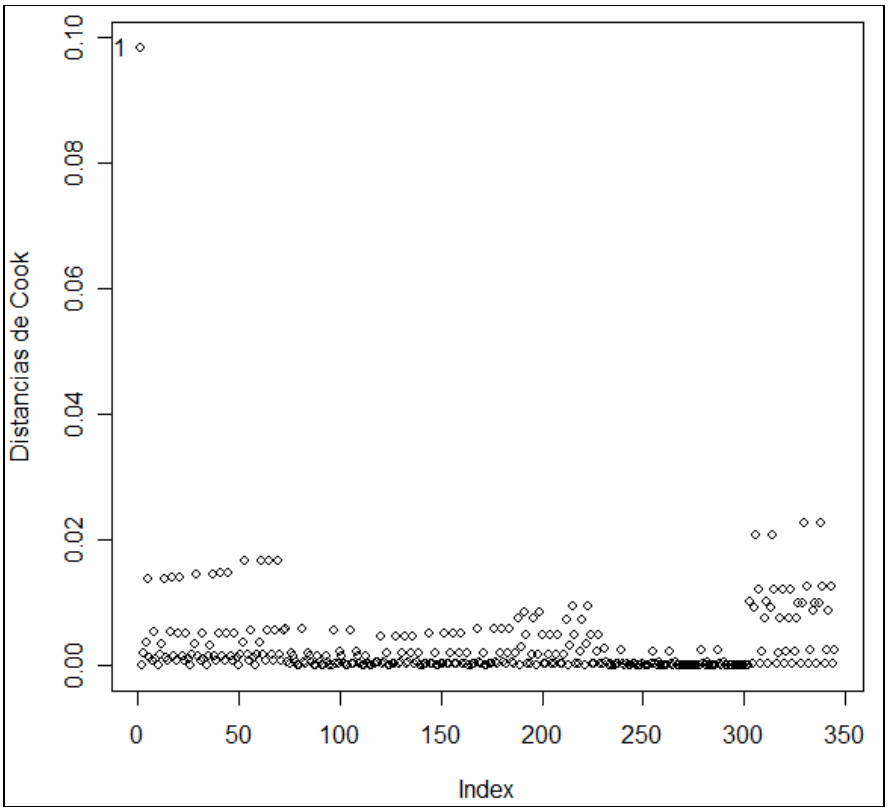


Figura 8.4 – Distancias de *Cook* com um ponto de influência em M3.1

Verifica-se na Figura 8.4, no canto superior esquerdo, a existência de um ponto de influência. Este ponto refere-se à primeira observação, dado que a numeração indicada é 1.

Removendo a primeira observação do conjunto de construção e realizando nova regressão para o modelo M3.1, obtém-se novos resultados apresentados na Figuras 8.5. Destes, a análise residual indica que os erros não seguem a normal, a variância é não-constante e existe um padrão de curvatura na gráfico (3).

Enfim, apesar de não existirem pontos de influência, como é confirmado na Figura 8.6, os resultados indicam que a função da média é incorreta, sendo assim o modelo M3.1 inadequado.

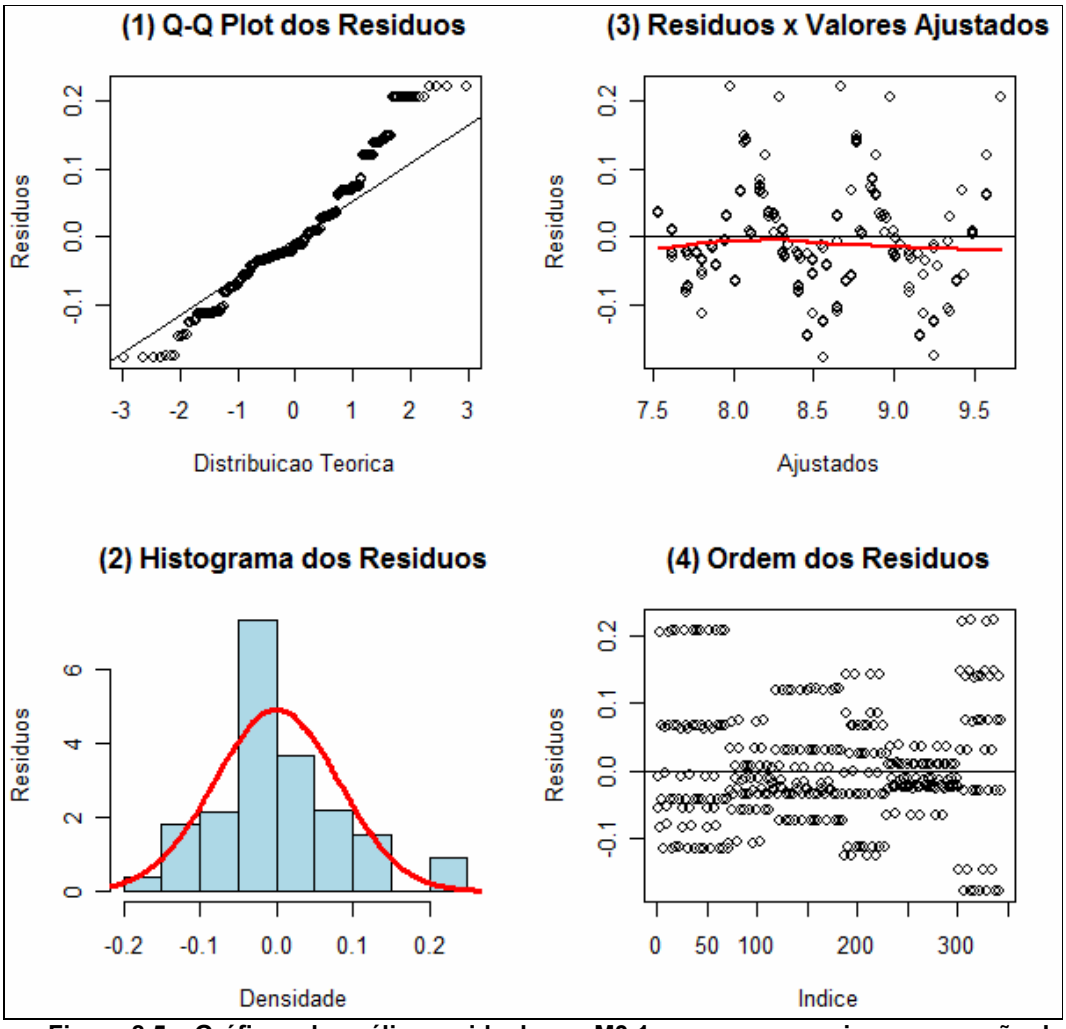


Figura 8.5 – Gráficos de análise residual para M3.1 com erro gaussiano e remoção de um ponto de influência.

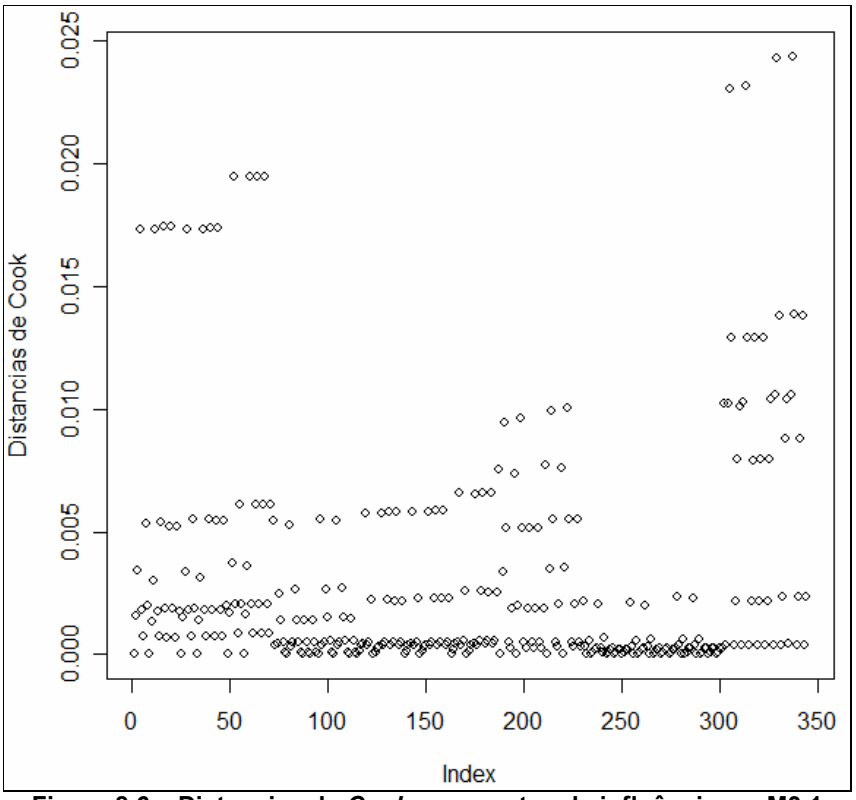


Figura 8.6 – Distancias de *Cook* sem pontos de influência em M3.1

9. Refinamento do modelo

9.1 Interações

MLGs podem também considerar a combinação entre dois ou mais fatores. Quando o modelo possui vários fatores, o efeito da combinação de dois ou mais fatores é chamado *efeito de interação*. Interações ocorrem quando o efeito de um fator varia de acordo com o nível de outro fator. Em contraste o efeito de um fator simples é chamado de *efeito principal*.

O conceito de interação é dado como segue: *se a mudança na média da variável resposta entre dois níveis de um fator A é a mesma para níveis diferentes de um fator B, pode-se dizer que não há interação; mas se a mudança é diferente para diferentes níveis de B, pode-se dizer que há interação*. [7]

Interações relatam o efeito que fatores têm sobre o risco do modelo, que não são relatados na correlação exposta entre os fatores. Neste trabalho, a necessidade de interações é indicada pela impossibilidade de representar, com precisão, os tempos de simulação utilizando apenas o efeito principal dos fatores. Conclusão obtida a partir da análise residual apresentada na seção anterior.

Na prática, de acordo com Anderson et al. em [3], o efeito de interação pode ser observado em gráficos com os valores ajustados contra os efeitos de fatores. Se não houver nenhum efeito significante de interação entre fatores, as linhas dos gráficos serão paralelas.

Na Figura 9.2 são mostrados os gráficos dos efeitos de interação dos fatores ts, bw, rc e pc. O fator m3p não exerce efeitos de interação, pois no modelo M3.1, é utilizado apenas o nível 3.

Observa-se que dentre os quatro gráficos, o único que não apresenta efeito de interação é o Gráfico (2) (referente aos efeitos do fator bw). Pode-se concluir então que os demais fatores devem possuir efeitos de interação no modelo de regressão.

Um termo de interação pode ser incluindo em um MLG simplesmente pela definição de uma nova variável preditora, que é dada em função do efeito multiplicativo entre dois ou mais fatores com efeitos de iteração.

Assim, a partir dos fatores ts, rc, pc e o nível 3 de m3p, se obtêm um novo modelo, chamado M3.2, a seguir

```
log(st) ~ rc + pc + bw + m3p3 + ts + rc*pc + rc*m3p3 + rc*ts + pc*m3p3 + pc*ts
          + m3p3*ts + rc*pc*ts + pc*m3p3*ts
```

Figura 9.1 – Modelo de regressão M3.2

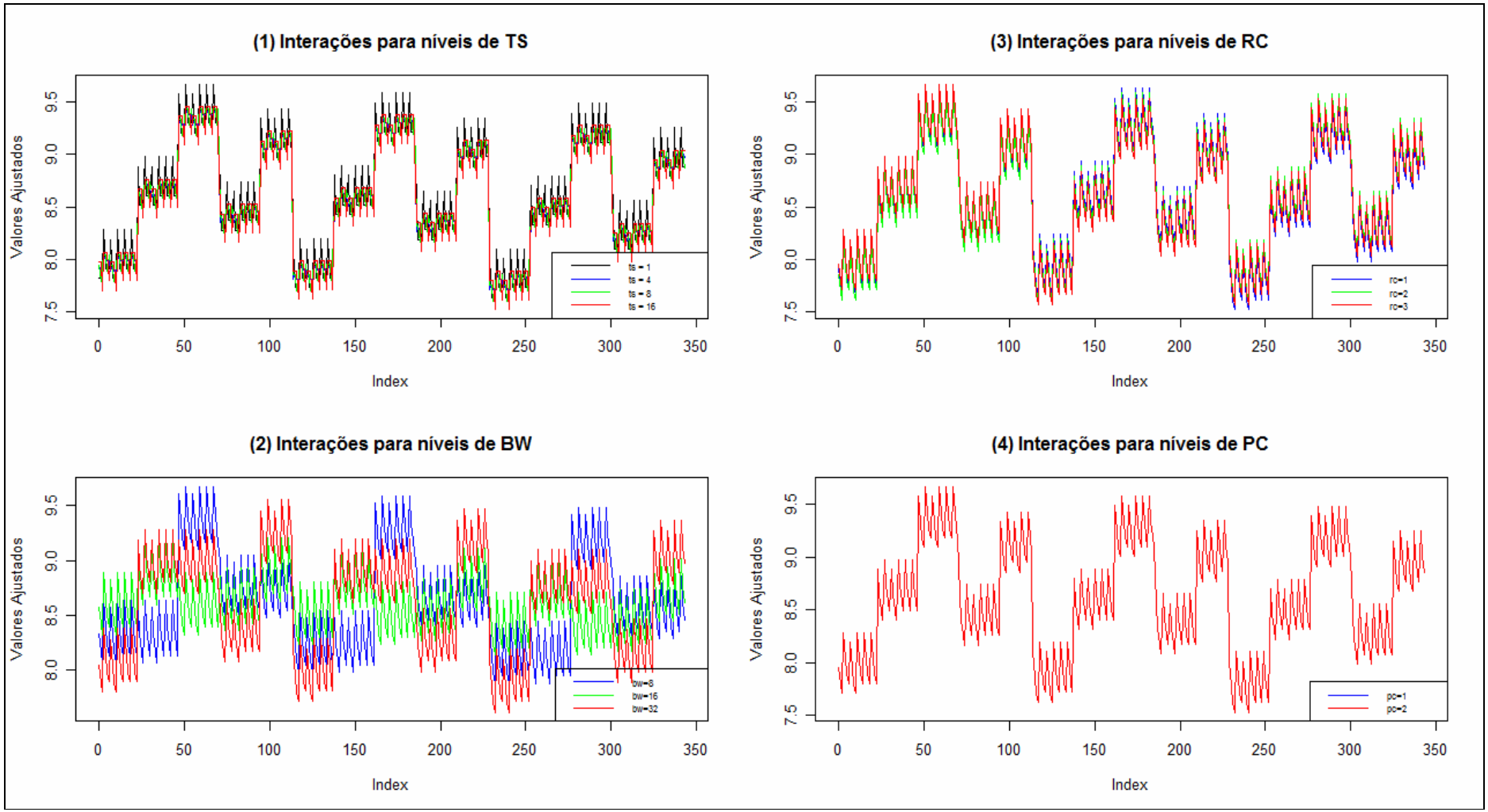


Figura 9.2 – Gráficos dos efeitos de interação para os fatores ts, bw, rc e pc.

As Figuras 9.3 e 9.4 mostram os gráficos de análise residual para o modelo M3.2 com erro gaussiano.

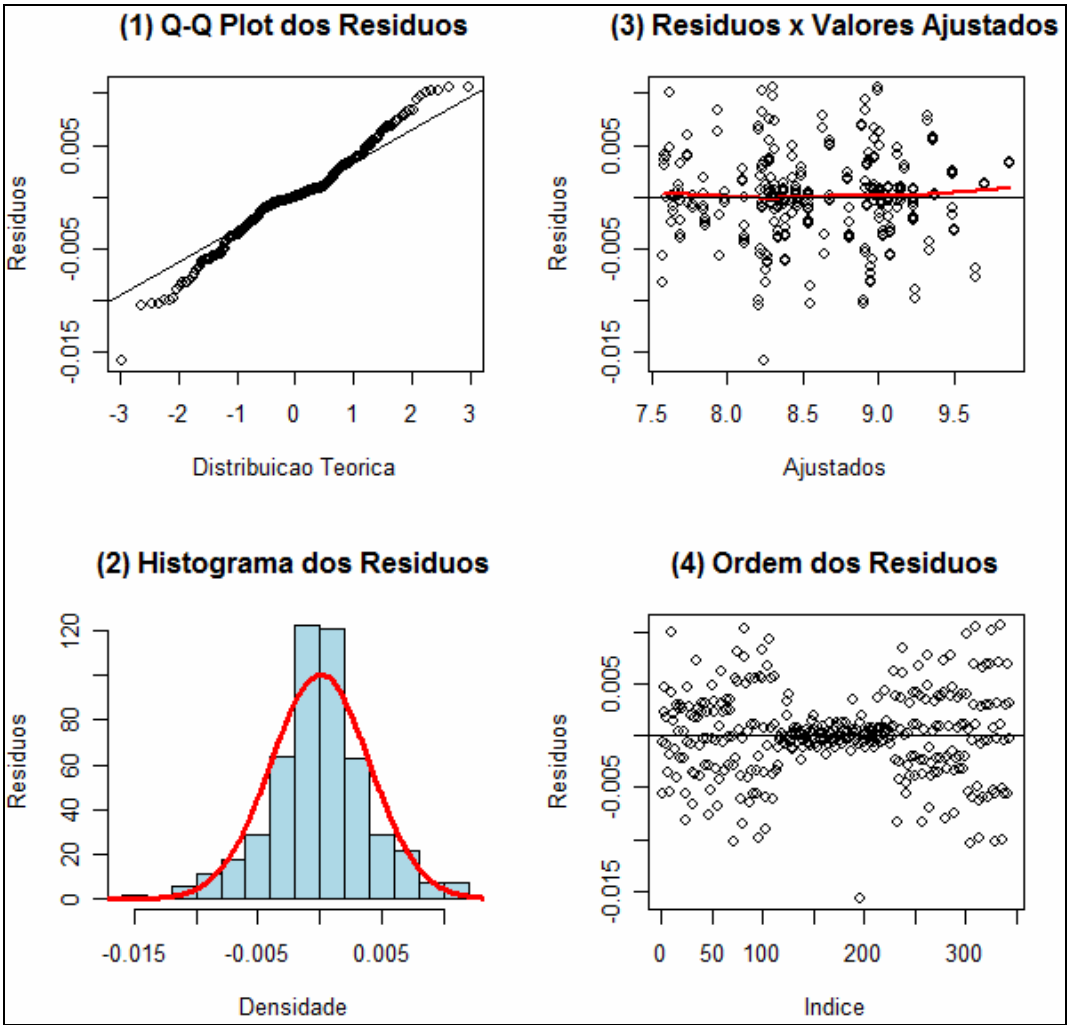


Figura 9.3 - Gráficos de análise residual para M3.2 com erro gaussiano.

Pode-se observar no gráfico (1), Figura 9.3, que o erro ainda não segue a distribuição normal. Porém, no gráfico (2), percebe-se, no histograma, uma tendência a normal. O gráfico (3) mostra que não existe um padrão de curvatura e que a variância tornou-se constante. Nos gráficos (1), (3) e (4) é possível verificar que existe um candidato a ponto de influência, que é constatado na Figura 9.4. Este ponto é a observação de número 227.

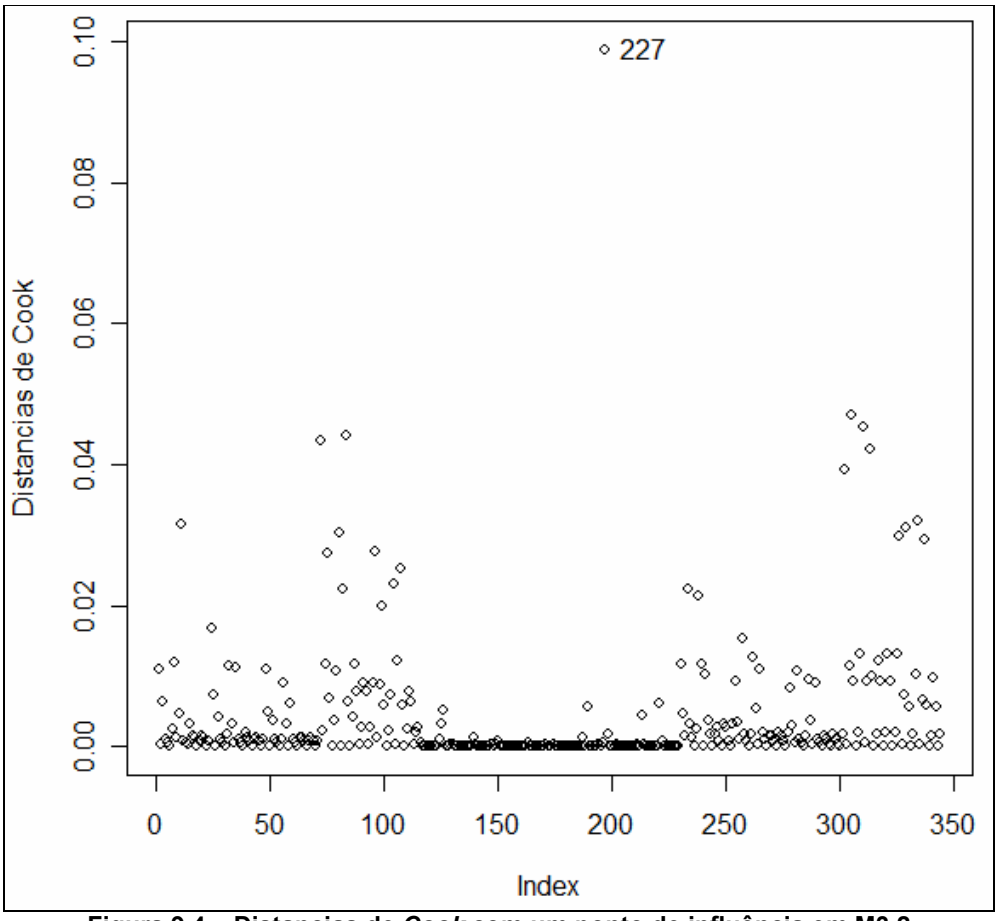


Figura 9.4 – Distancias de *Cook* com um ponto de influência em M3.2

A Figura 9.5 mostra novos gráficos de análise residual para o modelo M3.2, também com erro gaussiano, porém sem a observação 227. Pode-se perceber que não mais existem candidatos a ponto de influência.

O gráfico (1) ainda não sugere normalidade dos erros, porém a gráfico (2) apresenta uma tendência razoável. Existem testes formais para normalidade, como *kolmogorov-Smirnov* [29] e *Shapiro-Wilk* [33], mas estes testes não são tão flexíveis quanto o gráfico Q-Q Plot, pois a estatística *p* não é muito útil como indicador de que ação tomar. [28]

O primeiro, dos dois testes de aderência supracitados, foi aplicado aos resíduos, com estrutura de erro gaussiana e gama, este com transformação de *Anscombe*, e obtiveram estatísticas *p* iguais a 4,3e-03 e 1,452e-02, respectivamente, indicando que as hipóteses nulas, hipótese de normalidade, devem ser descartadas.

Para o segundo teste, também aplicado às duas estruturas de erro, gaussiana e gama, os valores da estatística *p* foram 3.220e-05 e 1,426e-4, respectivamente, indicando que as hipóteses nulas também devem ser descartadas.

Porém, Faraway, em [28], afirma que para curvas com pouca cauda, que é dada pelo padrão em “S” apontado no gráfico (1), os efeitos da não-normalidade não são tão sérios e podem ser razoavelmente ignorados.

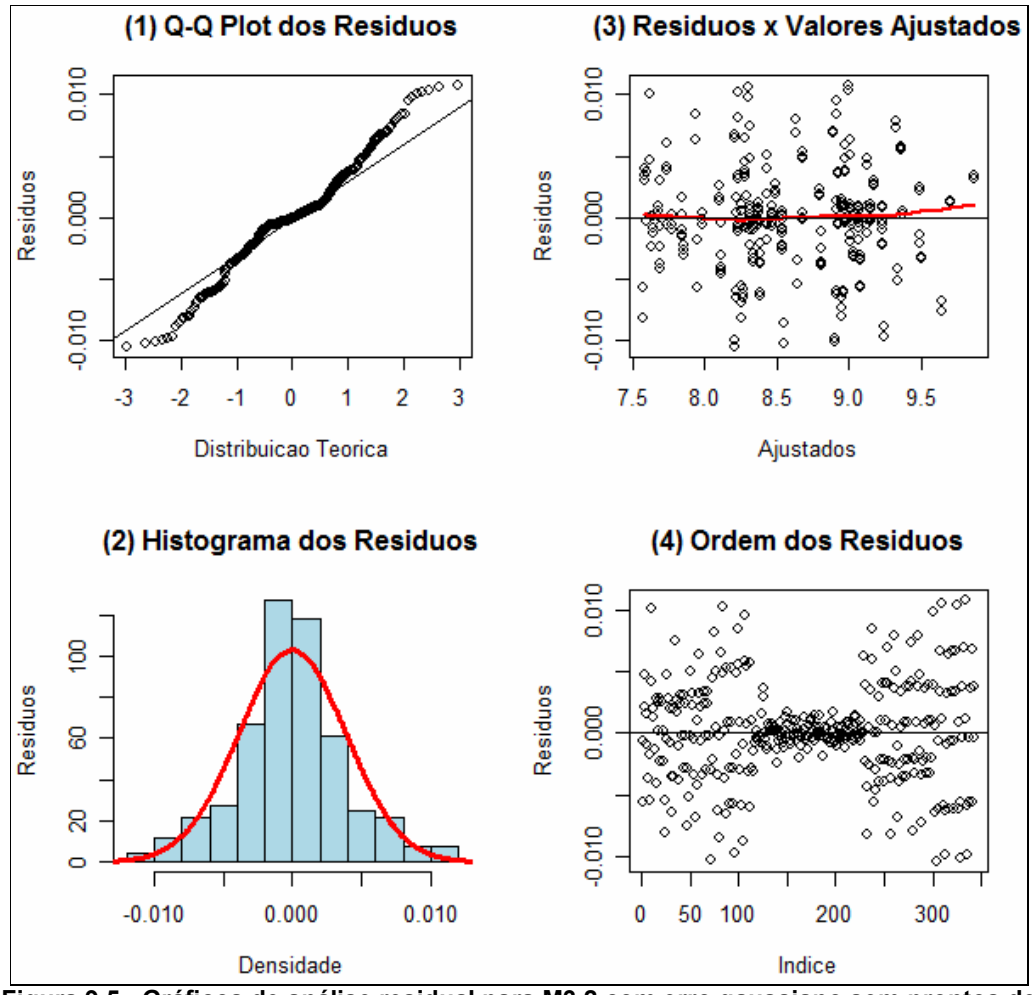


Figura 9.5 - Gráficos de análise residual para M3.2 com erro gaussiano sem prontos de influência.

O gráfico (3) apresenta a ausência de curva e variância constante e o gráfico (4) mostra não-correlação entre os erros.

10. Estimação da discrepância do modelo

Para testar a adequabilidade do modelo proposto (M3.2) ao conjunto de dados observados, utilizou-se os testes de *kolmogorov-Smirnov*, comparando os valores ajustados aos valores observados, e *Qui-Quadrado* [34].

No primeiro teste, o modelo M3.2 obteve estatística *p* igual a 0,9847, indicando que este modelo se adequou quase que perfeitamente aos dados.

Para segundo teste, obteve ótima aceitação, pois o resultado encontrado para estatística *p* é igual a 1.

Graficamente também é possível avaliar a discrepância do modelo. A Figura 10.1 mostra um gráfico com duas curvas: a curva de cor vermelha refere-se aos valores da variável aleatória de cada observação do conjunto de construção e a azul refere-se aos respectivos valores obtidos pelo modelo M3.2.

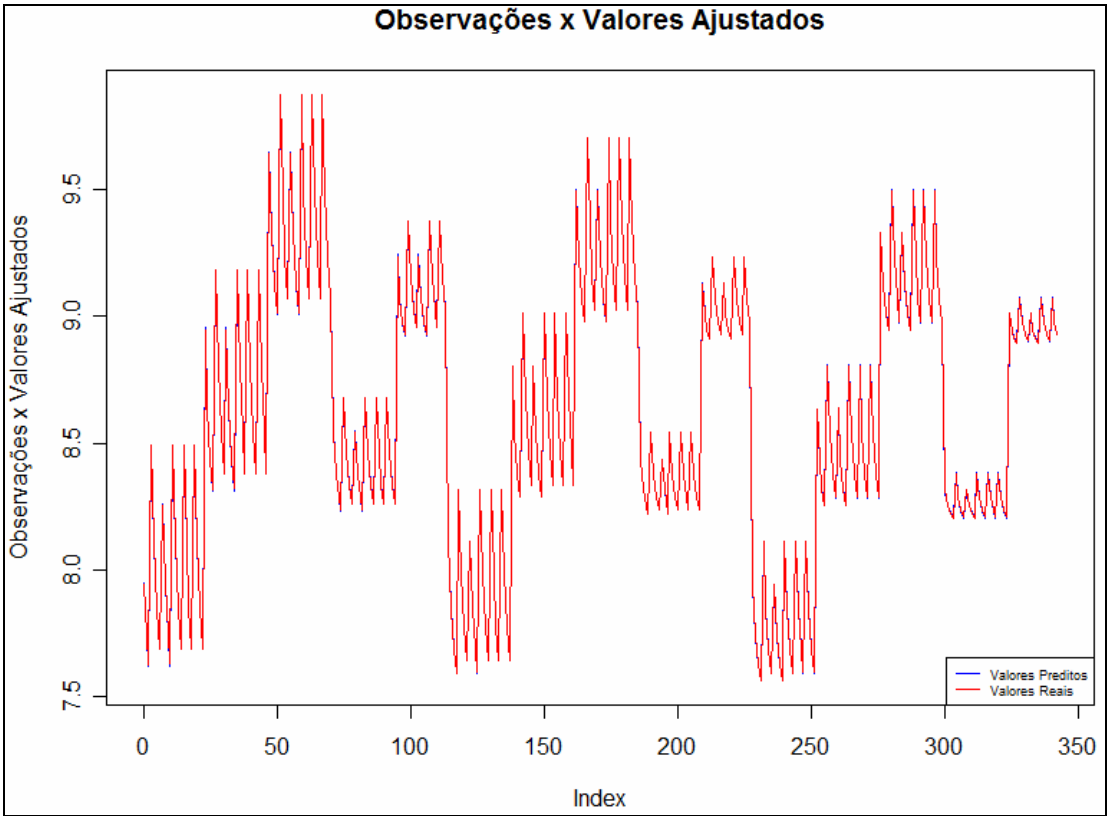


Figura 10.1 – Gráfico comparativo entre valores observados e ajustados de acordo com as observações.

De acordo com a Figura 10.1, pode-se observar que as curvas em sua quase totalidade se sobrepõem. Isto indica que o modelo M3.2 pode ser utilizado para modelar o problema, dado que os valores ajustados são muito próximos (ou idênticos, na grande maioria dos casos) aos valores observados.

11. Predições

Em predições, têm-se novos casos, possivelmente, um valor futuro, que não foi utilizado para estimação de parâmetros, com valor observado do preditor X_*. Deseja-se conhecer o valor de Y_*, a resposta correspondente, mas que não foi ainda observado. Assim, pode-se utilizar a função da média estimada para predizê-lo. Assume-se que os dados usados para estimar a função da média são relevantes para o novo caso, assim o modelo ajustado aplica-se a ele. [7]

Dado que o MLG proposto foi devidamente avaliado e validado, pode-se utiliza-lo para obter os valores médios da variável resposta a partir dos valores das variáveis sistemáticas contidos no conjunto de teste. Deste, os valores da variável resposta (tempo de simulação) observados serão extraídos e utilizados para fins de comparação com os valores preditos.

O gráfico contido na Figura 11.1 ilustra a comparação entre os valores observados da variável resposta do conjunto de teste e os valores preditos.

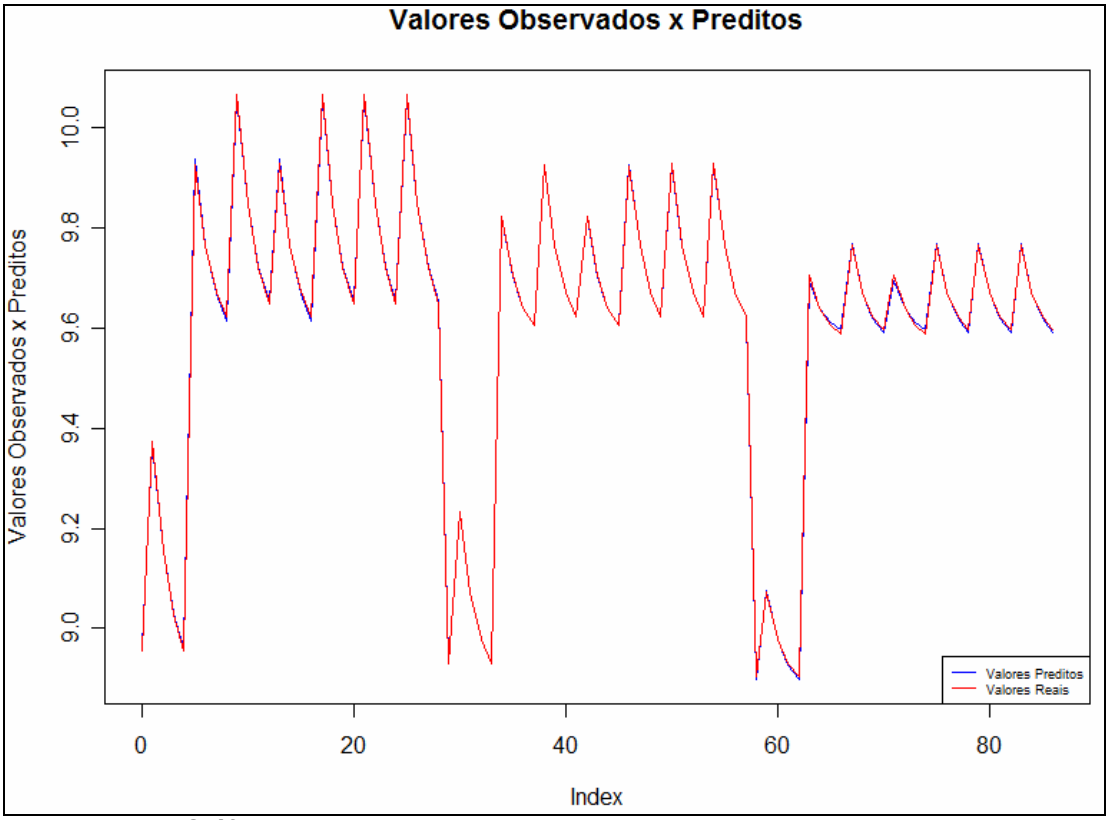


Figura 11.1 – Gráfico comparativo entre valores observados e preditos de acordo com as observações.

A curva vermelha representa os valores observados no conjunto de teste para a variável resposta e a azul representa os valores preditos. Observa-se que as curvas se sobrepõem na maioria do gráfico, indicando que os valores preditos são compatíveis com os observados.

A precisão dos resultados, indicada pela comparação no gráfico da Figura 11.1, afirma que o modelo proposto é preciso e que os resultados podem ser

utilizados para compor o espaço de opções de configuração da estrutura de comunicação do sistema embarcado do estudo de caso.

11. Conclusões

Este trabalho apresentou um estudo sobre as influências das características da estrutura de comunicação sobre o desempenho de sistemas embarcados, utilizando Modelos Lineares Generalizados. Para tanto, foi utilizado um estudo de caso que consistiu de um cenário de multiprocessamento de *streams* de dados, onde foram utilizados modelos de simulação de três processadores, um barramento e uma memória externa.

O modelo de barramento utilizado é parametrizável e, assim, permitiu realizar diversas simulações, com diferentes configurações. Os dados das configurações e os respectivos tempos de simulação dos modelos de simulação MPSoC foram divididos em dois conjuntos: teste e construção. O conjunto de construção foi utilizado para, através do método de máxima verossimilhança, estimar os parâmetros que compõem o modelo de comportamento do sistema embarcado. O conjunto de teste foi utilizado para previsões, onde estas foram comparadas com as observações de desempenho do conjunto de teste.

Classificou-se o modelo final como um bom modelo para predições de desempenho do cenário do estudo de caso. Com a precisão dos valores preditos é possível utilizar os resultados para compor totalmente o espaço opções de configuração de comunicação para o projeto de sistemas embarcados.

Como trabalhos futuros destacam-se:

- Utilizar outras distribuições da família exponencial para modelar a distribuição dos erros;
- Utilizar outros tipos de resíduos, de forma a normalizá-los totalmente, quando utilizadas outras distribuições diferentes da normal;
- Utilizar a técnica PRESS [12] para validação cruzada,
- Utilizar outras técnicas para obter modelos de comportamento e comparação de resultados, como: métodos Monte Carlo [44], Análise de Variância [19] e Análise de Fatores [45].

Referências Bibliográficas

[1] P. MCCULLAGH and J. A. NELDER. Generalized Linear Models. Chapman and Hall. 1989.

[2] Y. JIAO, Y. CHEN, D. SHNEIDER and J. WROBLEWSKI. A simulation study of impacts of error structure on modeling stock-recruitment data using generalized linear models. NRC Research Press Website: http://cjfas.nrc.ca, 2004.

[3] D. ANDERSON, S. FELDBLUM, C. MODLIN, D. SCHIRMACHER, E. SCHIRMACHER, N. THANDI. A Practitioner's Guide to Generalized Linear Models. Watson Wyatt Worldwide, 2007.

[4] NELDER, J. A. and WEDDERBURN, R. W. M. Generalized linear models. Journal of the Royal Statistical Society, A, 135, 370{384.25, 1972.

[5] L. BENINI and G. D. MICHELI. Powering networks on chips. In Proceedings of the Int. Symp. Syst. Level Synthesis, pages 33–38, 2001.

[6] D. C. BLACK and J. DONOVAN. SystemC: From the Groung Up. In Kluwer Academic Publishers., 2004.

[7] S. WEISBERG. Applied Linear Regression. John Wiley and Sons, 2005.

[8] SAKAMOTO, Y., ISHIGURO, M., and KITAGAWA, G. Akaike Information Criterion Statistics. Dordrecht: Reidel, 1987.

[9] Y.-S. CHO, E.-J. CHOI, and K.-R. CHO. Modeling and analysis of the system bus latency on the SoC platform. In Proceedings of the international workshop on System-level interconnect prediction, pages 67–74, 2006.

[10] SCHWARZ, G. Estimating the dimension of a model. Annals of Statistics, 6, 461–464, 1978.

[11] Mallows, C. Some comments on Cp. Technometrics, 15, 661–676, 1973

[12] ALLEN, D. M. The relationship between variable selection and prediction. Technometrics,16, 125–127, 1974

[13] Minitab. Meet MINITAB. Minitab Inc, 2003.

[14] R. DÖMER, A. GERSTLAUER, and D. GAJSKI. Specc methodology for high-level modeling. In 9th IEEE/DATC Electronic Design Processes Workshop, 2002.

[15] R.R. Hocking. A Biometrics Invited Paper: The Analysis and Selection of Variables in Linear Regression. Biometrics, 32, 1-49, 1976.

[16] H. EL-REWINI and M. ABD-EL-BARR. Advanced computer architecture and parallel processing. In John Wiley & Sons, Inc., 2005.

[17] A. J. Dobson. An Introduction to Generalized Linear Models. 2nd. Ed. Chapman and Hall/CRC, 2002.

[18] F. E. GUIBALY. Design and analysis of arbitration protocols. In IEEE Transactions on Computers, pages 161–171, 1989.

[19] R. R. HOCKING. Methods and Applications of Linear Models Regression and The Analysis of Variance. 2nd ed. Wiley and Sons, inc, 2003.

[20] J. NETER, W. WASSERMAN and M.H. KUTNER. Applied Linear Statistical Models, Second Edition. Irwin, Inc, 1985.

[21] C. E. McCULLOCH and S. R. SEARLE. Generalized, Linear and Mixed Models.Willey and Sons, Inc, 2001.

[22] BOX, G. E. P. and COX, D. R. An analysis of transformations (with discussion). Journal of the Royal Statistical Society B, 26, 211–252, 1964

[23] R. HO, K. W. MAI, and M. A. HOROWITZ. The future of wires. In Proceedings of the IEEE, vol. 89, pages 490–504, April 2001.

[24] A. A. JERRAYA. Long term trends for embedded system design. In Proceedings of the CEPA 2 Workshop - Digital Platforms for Defence, pages 15–16, 2005.

[25] A. A. JERRAYA andW.WOLF. Multiprocessor systems-on-chips. In Morgan Kaufmann, September 2004.

[26] K. KEUTZER. System-level design: Orthogonalization of concerns and platform- based design. In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2000.

[27] COOK, R. D. Detection of influential observations in linear regression. Technometrics, 19, 15–18, 1977.

[28] FARAWAY, J. J. Practical regression and Anova using R. Website: www.stat.lsa.umich.edu/˜faraway/book, 2002.

[29] Durbin, J. Distribution theory for tests based on the sample distribution function. SIAM, 1973.

[30] K. LAHIRI and A. RAGHUNATHAN. Power analysis of system-level on-chip communication architectures. In Proceedings of the Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pages 236–241, September 2004.

[31] K. LAHIRI, A. RAGHUNATHAN, and S. DEY. Efficient exploration of the SoC communication architecture design space. In Proceedings of the Intl. Conf. on Computer Aided Design, pages 424–430, 2000.

[32] K. LAHIRI, A. RAGHUNATHAN, and S. DEY. System-level performance analysis for designing on-chip communication architectures. In IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, pages 768–783, June 2001.

[33] SHAPIRO, S. S. and WILK, M. B. An analysis of variance test for normality (complete samples). Biometrika, 52:591–611, 1965.

[34] CHERNOFF, H. and LEHMANN, E.L. The use of maximum likelihood estimates in χ2 tests for goodness-of-fit. The Annals of Mathematical Statistics; 25:579-586, 1954.

[35] D. L. LIU and C. SVENSSON. Power consumption estimation in CMOS VLSI chips. In IEEE Journal SSC, pages 1531–1549, 1994.

[36] HASTIE, T. J. and PREGIBON, D. Generalized linear models. Chapter 6 of Statistical Models in S. Eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole, 1992.

[37] “The R Project for Statistical Computing”, Website: http://www.r-project.org/.

[38] L. NULL and J. LOBUR. The Essencials of Computer Organization and Architecture. In Jones and Bartlett Publishers, 2003.

[39] S. PRATA. C++ primer plus. 5th edition. sams. 2004.

[40] H. ROCHA. SCExamine: Um mecanismo para introspecção de sistemas em SystemC. Dissertação de Mestrado - CIn/UFPE. 2006.

[41] A. SANGIOVANNI-VINCENTELLI. Defining platform-based design. In EDesign of EETimes, 2002.

[42] F. VAHID and T. GIVARGIS. Embedded System Design: A Unified Hardware/Software Introduction. In John Wiley & Sons, Inc., 2002.

[43] J. XU,W.WOLF, J. HENKEL, and S. CHAKRADHAR. A methodology for design, modeling, and analysis of networks-on-chip. In IEEE International Symposium on Circuits and Systems ISCAS., pages 1778–1781, 2005.

[44] ROBERT, C. P. and CASELLA, G. Monte Carlo statistical methods. Springer-Verlag, 1999.

[45] TUCKER, L. and MACCALLUM, R. C. Exploratory Factor Analysis. 1997. Website: http://quantrm2.psy.ohio-state.edu/maccallum/factornew.htm.

Apendice A

SystemC

SystemC é uma linguagem de projeto de sistemas embarcados que surgiu da necessidade pervasiva por uma linguagem que aumentasse a produtividade do projetista de sistemas eletrônicos [6]. Na realidade, SystemC não é uma linguagem, mas sim uma biblioteca de classes C++ que permite modelagem de software e hardware, criando, assim, um modelo simulável do sistema.

Suas principais características incluem:

• **Modelo de tempo:** SystemC define um modelo de tempo com 64 bits de resolução utilizando uma classe conhecida como sc_time. Possui um micro-kernel de simulação que permite modelar e avançar tempo global.

• **Tipos de dados de hardware:** não é possível ter-se a variedade de tipos de dados necessários para um hardware digital dentro dos limites dos tipos de dados nativos de C++. Assim, SystemC provê tipos de dados compatíveis com hardware que dão suporte à configuração em bits da largura do dado. Além disto, permite representar dados *four-state* (0,1,X,Z) e todas os métodos necessários para utilizar estes tipos de dados, incluindo conversão entre os tipos de dados de hardware e conversão de tipos de dados de hardware para software.

• **Hierarquia e estrutura:** para modelar hierarquia de hardware, SystemC utiliza uma entidade chamada módulo. Estes módulos são interconectados através de canais de comunicação. A hierarquia surge da instanciação de classes de módulos dentro de outros módulos.

• **Gerenciamento de comunicação:** canais de comunicação representam um poderoso mecanismo para modelagem de comunicações. Conceitualmente, um canal é mais do que um simples sinal ou fio. Canais podem representar esquemas de comunicação mais complexos que eventualmente pode ser mapeados para hardware.

• **Concorrência:** simulação de execução paralela é efetuada pela simulação de cada unidade concorrente. Cada unidade pode ser executada até que a simulação de outras unidades seja necessária para manter os comportamentos alinhados no tempo.

Para modelar todas essas características, SystemC define vários tipos de componentes. A Figura A.1 ilustra um sistema com os principais tipos.

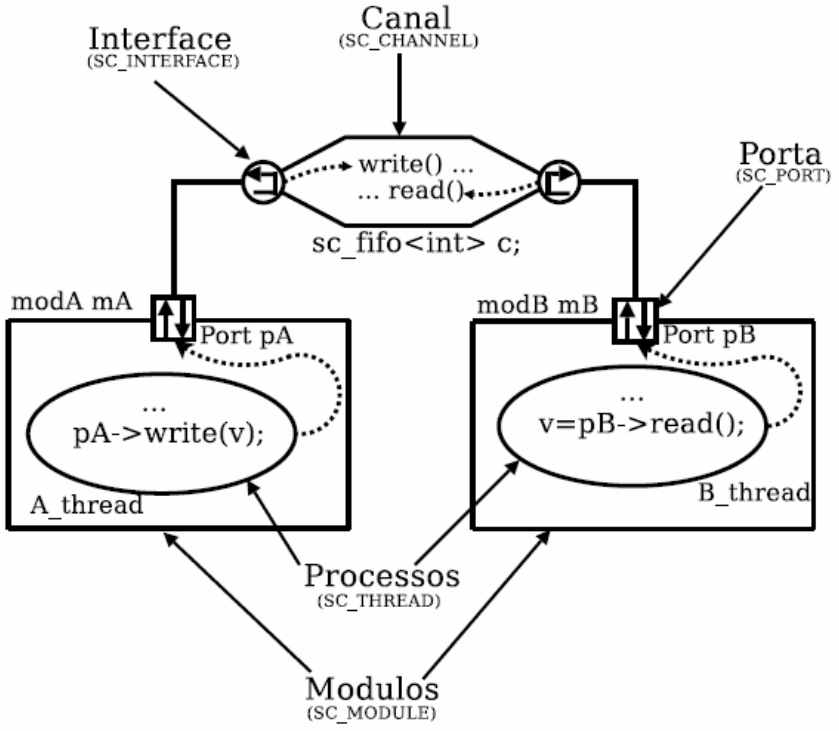


Figura A.1 – Principais componentes de SystemC.

Como indicado anteriormente, o micro-kernel escalona os processos de simulação de execução. Processos de simulação são simplesmente funções membro, que na Figura A.1 é representado por uma SC_THREAD, de módulos, que, por sua vez, são definidos pela diretiva SC_MODULE. Para comunicarem-se, módulos utilizam portas (SC_PORTS). As portas são ligadas aos canais de comunicação, que são definidos por SC_CHANNEL e SC_INTERFACE.

Em SystemC, além de porta e canal, outra estrutura utilizada para comunicação entre módulos é o SC_EXPORT. Este, conceitualmente, é bastante semelhante a SC_PORT, apenas se diferencia pelo fato de não necessitar de um canal para comunicação. Módulos que se comunicam através de SC_PORT e SC_EXPORT, têm suas interfaces ligadas diretamente, como se estivem acessando um canal de comunicação. A Figura A.2 ilustra a conexão de dois módulos, A e B, através de SC_PORT e SC_EXPORT.

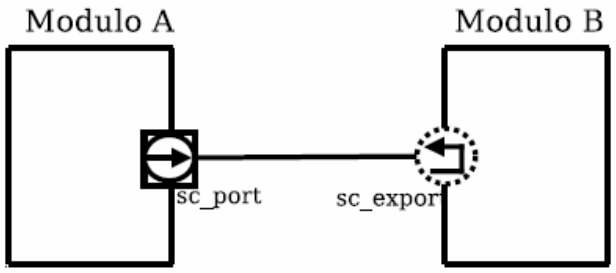


Figura A.2 Comunicação de dois módulos através de SC_PORT e SC_EXPORT.

SystemC, por ser uma biblioteca de classes C++, permite modelar componentes de hardware em diferentes níveis de abstração. Estes níveis podem ser divididos, basicamente, de acordo com as informações temporais que fornecem: precisão de ciclo (*cycle accurate*), tempo estimado (*time estimated*) e comportamental (*behavioral*). No primeiro, que possui o menor desempenho de simulação em relação aos demais modelos de alto nível, sua funcionalidade é regida por um *clock* de sistema. Por causa desta estrutura, os modelos possuem informações bastante precisas de temporização. O segundo tipo não possui *clock*, porém os modelos fazem estimativas dos tempos necessários para cada operação. O último tipo não possui informações temporais. Geralmente, é utilizado para validar as funções do componente e possui o maior desempenho de simulação.