Nanodegree Engenheiro de Machine Learning

Proposta de projeto final: Detecção de Falha em circuitos Elétricos

Gabriel da Silva Costa

26 de agosto de 2018

Histórico do assunto

O desenvolvimento de estratégias de teste para detectar e diagnosticar falhas em circuitos analógicos e de sinais mistos é uma tarefa desafiadora que tem encorajado uma boa quantidade de pesquisas, devido ao aumento do número de aplicações destes circuitos e ao alto custo dos testes. Muitas áreas, tais como, telecomunicações, multimídia e aplicações biomédicas, precisam de bom desempenho em aplicações de alta frequência, baixo ruído e baixa potência, o que só se pode ser alcançado através do uso de circuitos integrados analógicos e de sinais mistos. Assim, uma estratégia para detectar e diagnosticar falhas nesse tipo de circuitos é muito importante (Albustani, 2004). No passado, um circuito integrado era apenas um componente em um sistema, mas hoje o circuito integrado em si é o sistema inteiro (SoC - *system on a chip*). Com esse nível de integração, problemas difíceis de teste e projeto foram gerados. Dentre os vários fatores que aumentam as dificuldades, pode-se citar: a falta de bons modelos de falhas, falta de um padrão de projeto com vistas à facilidade de testes e o aumento da importância das falhas relacionadas ao tempo de vida dos componentes (Claasen, 2003). Assim, a estratégia de testes para detecção e diagnóstico de falhas ainda é severamente dependente da perícia e da experiência que os engenheiros têm sobre as características do circuito. Então a detecção e a identificação de falhas ainda são um processo interativo e que consome bastante tempo. Um estudo na área de detecção e diagnóstico (Fenton, 2001) mostrou que, nas últimas décadas, uma boa quantidade de pesquisa em diagnósticos de falhas foi concentrada em desenvolver ferramentas que facilitassem as tarefas de diagnóstico. Embora progressos importantes tenham sido alcançados, essas novas tecnologias não têm sido largamente aceitas. Isso deve motivar os pesquisadores a investigar outros paradigmas e desenvolver novas estratégias para diagnósticos de falhas.

O uso de técnicas de inteligência computacional para diagnóstico é normalmente baseado na construção de modelos ou no uso de classificadores, cujo sucesso e desempenho depende da qualidade do modelo obtido, o que, no caso de um sistema complexo, pode ser difícil de obter. Os classificadores multiclasse procuram por comportamentos específicos de falhas e se mostram vulneráveis a superposição de padrões de falha ou a padrões de falha que não foram apresentados durante a fase de treinamento. Classificadores de classe única

podem ser treinados para resolver problemas de classificação binária onde apenas uma das classes é bem conhecida (Tax, 2001). Estes podem ser organizados na forma de conjunto (ensemble) de classificadores e com isso reduzir alguns dos problemas encontrados com classificadores multiclasse citados anteriormente. Esta proposta de projeto apresenta o esboço do desenvolvimento de um sistema de detecção de falhas em circuitos lineares e invariantes no tempo, onde os resultados de diferentes métodos serão comparados para a determinação do melhor tipo de classificador.

Descrição do problema

O termo *falha* é definido como uma condição anormal ou defeito (ISO/CD 10303), em um componente, equipamento ou sistema que pode conduzir ao mau funcionamento, isto é, uma diminuição parcial ou total na capacidade de desempenhar a função desejada por certo intervalo de tempo.

Em circuitos analógicos, as falhas podem ser classificadas usando diferentes critérios. O tipo de falha abordada neste projeto é aquela que se dá em função do desvio do parâmetro de um sistema (ou componente do sistema) no tempo, designada falha *paramétrica*, forçando-o a assumir um valor que está fora de sua faixa nominal. Quando existe um desvio repentino muito grande do valor do parâmetro desejado, este é chamado de falha catastrófica. Este tipo de falha está associado à mudança da estrutura do sistema. Alguns exemplos de falhas catastróficas em circuitos elétricos seriam o circuito aberto e o curto-circuito (DUHAMEL E RAULT, 1979).

Conjuntos de dados e entradas

Os conjuntos de dados usados neste projeto são os dados de simulação de tipos diferentes de circuitos elétricos exportados pelo simulador LTSpiceIV, no formato raw, e que serão lidos para tratamento através da biblioteca "LTSpice_RawRead.py", desenvolvida por Nuno Canto Brum" (http://www.nunobrum.com/pyspicer.html).

Cada circuito simulado no LTSpiceIV é submetido a variações nos parâmetros do seus componentes. Conforme os parâmetros são submetidos a variações, algumas das grandezas do circuito são observadas como, por exemplo, tensão do sinal de entrada, corrente de entrada, tensões e correntes individuais de cada componente e,

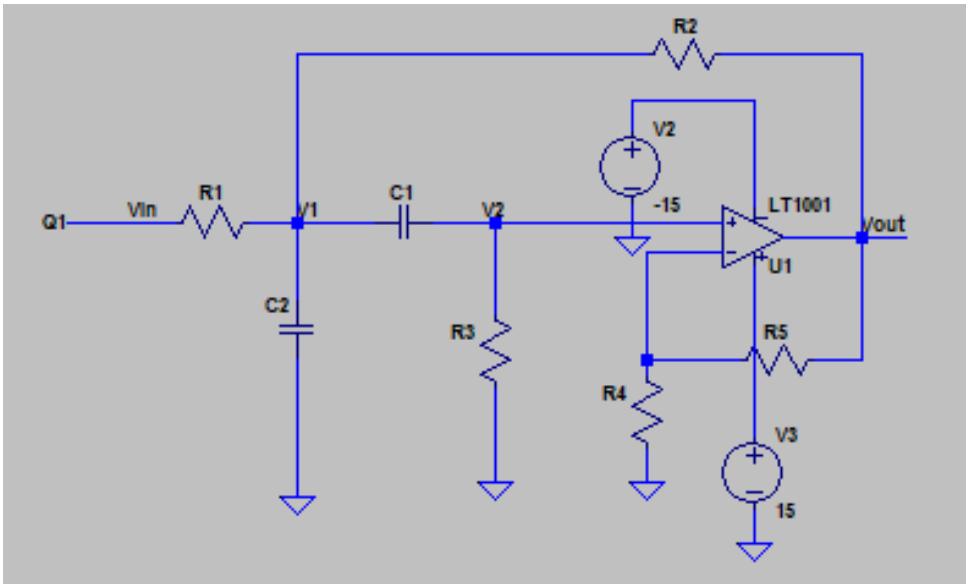
principalmente, tensão de saída. Desta forma, é fácil prever que a quantidade de dados gerados a cada simulação é enorme.

Os dados de entrada podem ser observados (todos) no formato “.raw” no link a seguir, e um arquivo “.csv” referente a um dos circuitos (Sallen Key mc + 4bitPRBS [FALHA]). Demais arquivos “.csv” serão disponibilizados conforme gerados.

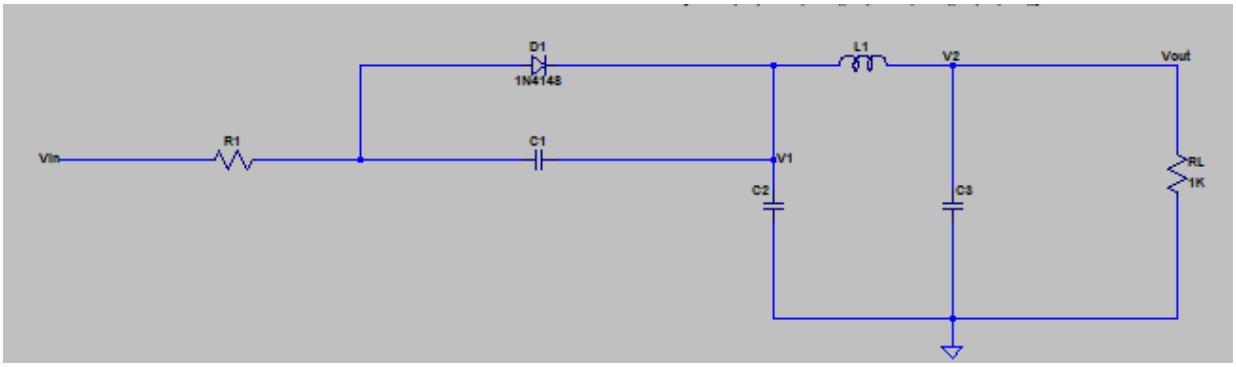
- https://drive.google.com/drive/folders/1KSEIhl6s4vkloGJGuobKPDel8952DXE4

Circuitos utilizados:

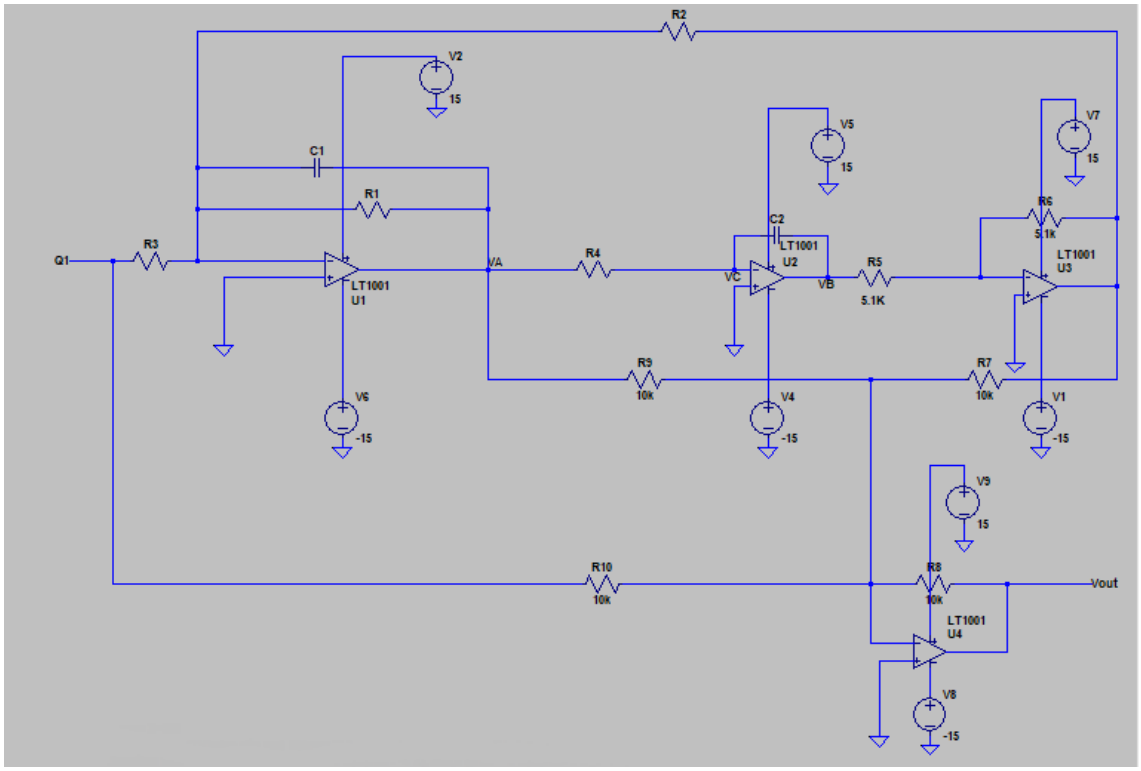
Sallen Key mc + 4bitPRBS [FALHA]:



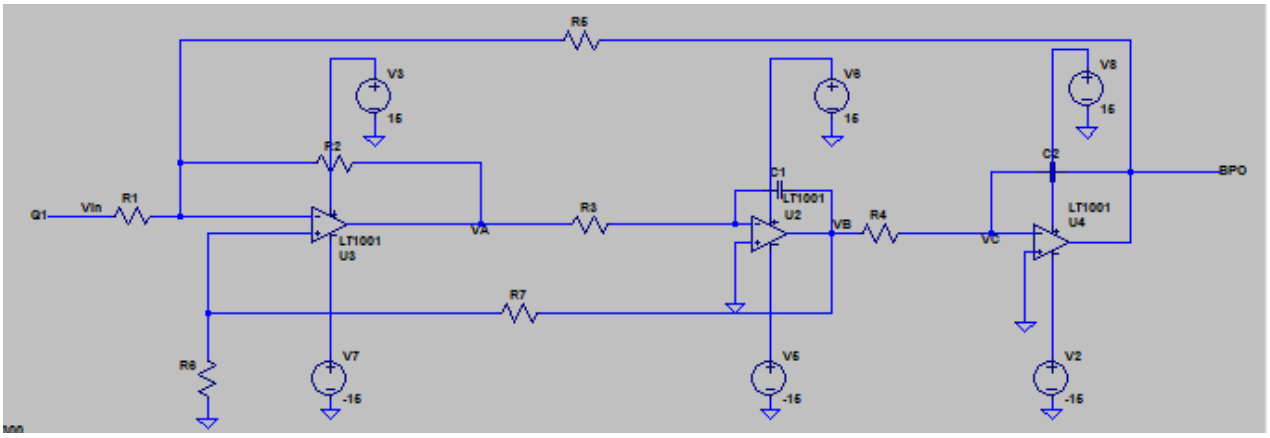
Nonlinear Rectfier + 4bit PRBS [FALHA] - 300 - 0.2s:



Biquad Highpass Filter mc + 4bitPRBS [FALHA]:



CTSV mc + 4bitPRBS [FALHA]



Descrição da solução

A solução para o problema de detecção de falhas consiste em observar os resultados da simulação, e treinar um modelo de predição e aprendizagem para que um possível componente com parâmetros alterados seja identificado como causador da falha no circuito. Este modelo treinado poderá ser submetido posteriormente a outras amostras de grandezas de um novo estado do mesmo circuito, para diagnóstico de falhas.

Modelo de referência (benchmark)

Os circuitos são simulados inicialmente em estado de funcionamento normal, e cada estado de falha é causado forçadamente, isto é, existe uma função no LTSpiceIV que causa a alteração dos parâmetros dos componentes de forma ordenada, prevista. Desta forma, para um dado conjunto de passos de simulação é possível prever qual foi o componente causador da falha, e qual o valor que levou àquela falha.

No caso específico do circuito “Sallen Key mc + 4bitPRBS [FALHA]”:

```
ac dec 100 10k 1meg
.step param run 1 3300 1
.tran 300us
.function falhaR1(baixo,alto,mc)  if((run>X)&(run<=2*X), alto,if (run<=X,baixo,mc))
.function falhaR2(baixo,alto,mc)  if((run>3*X)&(run<=4*X), alto,
        if ((run<=3*X)&(run>2*X),baixo,mc))
.function falhaR3(baixo,alto,mc)  if((run>5*X)&(run<=6*X), alto,
        if ((run<=5*X)&(run>4*X),baixo,mc))
.function falhaC2(curto,aberto,normal)  if((run>9*X)&(run<=10*X),aberto,
        if((run<=9*X)&(run>8*X),curto,normal))
.function falhaC1(curto,aberto,normal)  if((run>7*X)&(run<=8*X),aberto,
        if((run<=7*X)&(run>6*X),curto,normal))
.param X=300
```

Onde serão executados 3300 passos de simulação, e a cada grupo de 300 passos um dos componentes tem seu valor alterado conforme código predefinido.

Métricas de avaliação

Como a alteração causadora da falha no circuito é previamente determinada, é possível usar um "gabarito" para qualificar e quantificar a precisão do modelo de predição. Desta forma é possível empregar as métricas do próprio scikit-learn, como o fbeta_score, ou o silhouette_score caso se mostre necessário empregar modelos não-supervisionados.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

O F-beta score é um método de avaliação de desempenho que representa o desempenho do modelo de predição com maior precisão.

Design do projeto

O projeto se inicia após a simulação dos circuitos, onde o LTSpiceIV gera os arquivos ".raw" com todas as grandezas dos circuitos. Este arquivo é então lido através da biblioteca LTSpice_RawRead.py, que organiza os dados em dicionários contendo os tempos e passos de simulação. As principais informações sobre as simulações para este projeto são os valores do sinal de entrada, Vin, e os de saída, Vout, então os dados fornecidos pela biblioteca de leitura do arquivo ".raw" são filtrados e organizados em um dataframe. Este dataframe possui suas linhas referentes aos passos de entrada, e suas colunas referentes aos valores de Vout. A exemplo do circuito "Sallen Key mc + 4bitPRBS [FALHA]", este dataframe possuirá 1200 linhas e 3300 colunas.

Devido à grande quantidade de dados neste dataframe, e tendo em vista a velocidade do treinamento, um método de redução de dados no espaço amostral pode ser empregado, como o PAA (PAA - *Piecewise Aggregate Approximation*). A aplicação da redução de elementos é possível pois os circuitos são lineares e, por muitas vezes, passos

sucessivos resultam em valores semelhantes. É a partir da representação dos dados por PAA que serão aplicados os métodos de classificação e predição.

Os resultados da classificação serão comparados com os parâmetros predefinidos da simulação dos circuitos. Como explicado anteriormente, a simulação como um todo é composta de grupos menores de simulações, grupos estes que são predefinidos no arquivo principal de simulação do LTSpiceIV, então a predição para um dado elemento do dataframe, que foi submetido ao PAA, deve ser coincidente com aquele predeterminado. Por exemplo, no caso do circuito "Sallen Key mc + 4bitPRBS [FALHA]", um elemento entre as colunas 0 e 300 é sabido como uma simulação com "falha em R1, valor alto", e a previsão será positiva se assim o classificar e negativa caso contrário. A métrica F-beta score será empregada para quantificar a precisão do classificador e os resultados do treino serão plotados para uma conferência mais direta do resultado.

Após avaliado o desempenho do modelo de predição, o classificador será submetido ao refino de parâmetros através da aplicação do *GridSearchCV*, ferramenta do scikit-learn usada para otimizar a configuração dos hiperparâmetros do classificador. Esta etapa é de suma importância para o máximo desempenho do projeto, e impede que tempo precioso seja desperdiçado em tentativas "manuais" de otimização.

Posteriormente, e fugindo um pouco ao escopo do projeto, seria possível aplicar métodos para determinar se existem relações entre grandezas distintas como, por exemplo, entre Vo e VR1, o que ampliaria a funcionalidade do projeto.