Relatório de atividades

Integrantes:

- Ana Caroline Saraiva Bezerra - acsb
- Humberto Alves Wanderley Neto - hawn
- Rafael Marinho de Araújo - rma6

Questão 1

A função de transferência da planta proposta é:

$$\frac{1}{s^2+5s+1}$$

Para os experimentos foram criados 3 vetores com valores para Kp, Ki e Kd com o propósito de verificar o comportamento do controle devido às alterações desses parâmetros. A quantidade de experimentos para cada controlador se encontra na tabela abaixo.

Controlador	P	PI	PD	PID	Total
experimentos	3	9	9	27	48

Controlador proporcional (P)

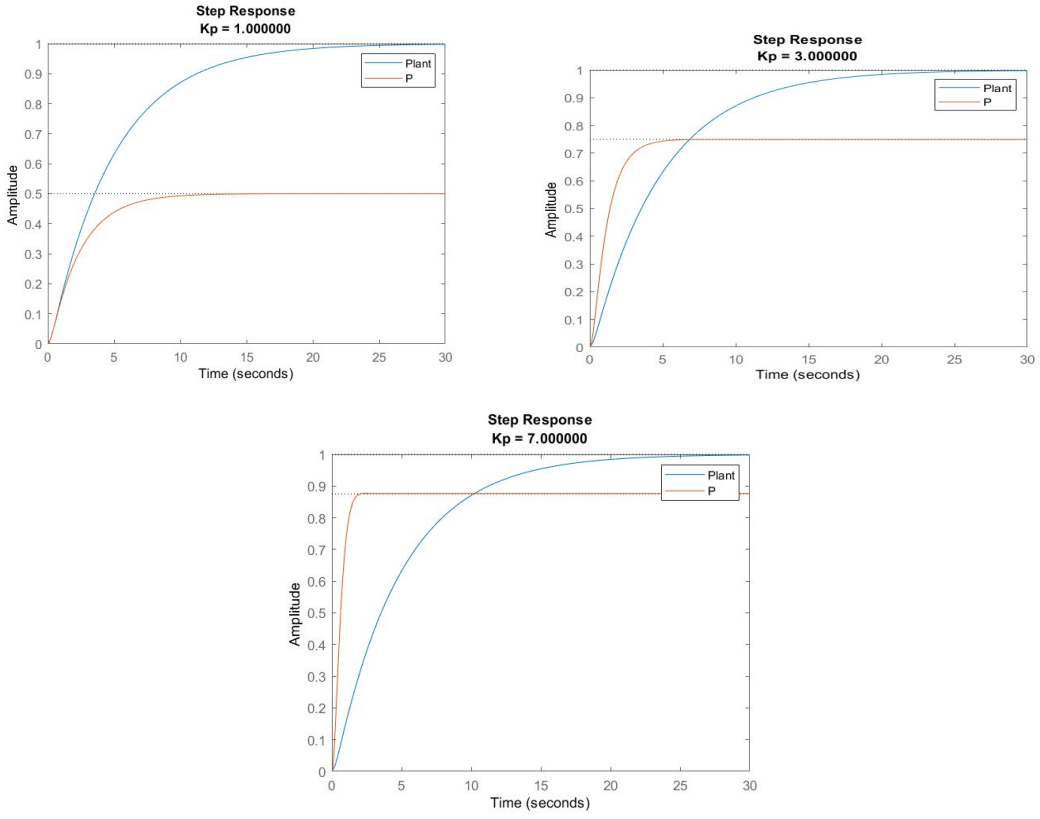
Um controlador proporcional de ganho Kp tem o efeito de reduzir o tempo de resposta, quando comparado a um controle simples do tipo ligado-desligado, e irá também reduzir, mas nunca eliminar, o erro de estado estacionário.

Como pode ser visto nos gráficos obtidos, conforme aumentamos o parâmetro Kp, o tempo de resposta até a convergência diminui. Contudo um Kp muito alto gera um alto sinal de saída, o que pode desestabilizar o sistema. Porém, como é visível nas imagens a seguir, se o Kp é muito baixo, o sistema falha em aplicar a ação necessária para corrigir os distúrbios.

Equação:

$$u(t) = K_p e(t)$$

Resultados obtidos para o controlador proporcional (P):



Controlador proporcional derivativo (PD)

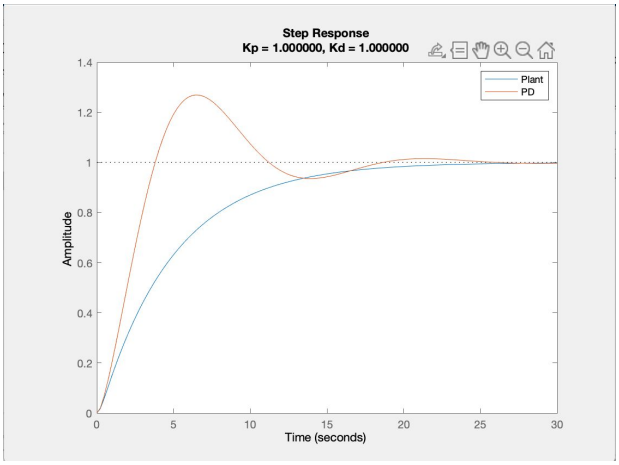
A função principal da ação derivativa é antecipar como o sistema vai reagir e já corrigir essa antecipação. Podemos então perceber que a ação derivativa é muito importante para corrigir desvio em variáveis que variam muito rapidamente, fazendo uma previsão do que acontecerá com o sistema e tomando as devidas medidas para evitar que essas variações continuem acontecendo.

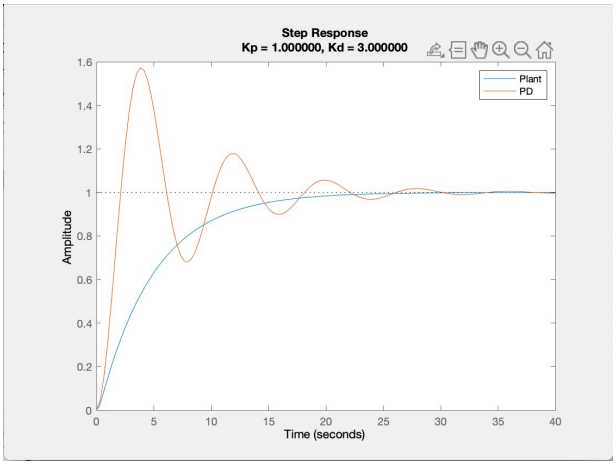
Podemos observar nos resultados obtidos, que se mantermos Kp fixo e variarmos apenas o Kd, o efeito de controle ocorre de forma gradual, diminuindo o erro e convergindo rapidamente. Caso o Kd seja muito alto, o controle começa a passar do ponto de convergência, demorando mais para convergir.

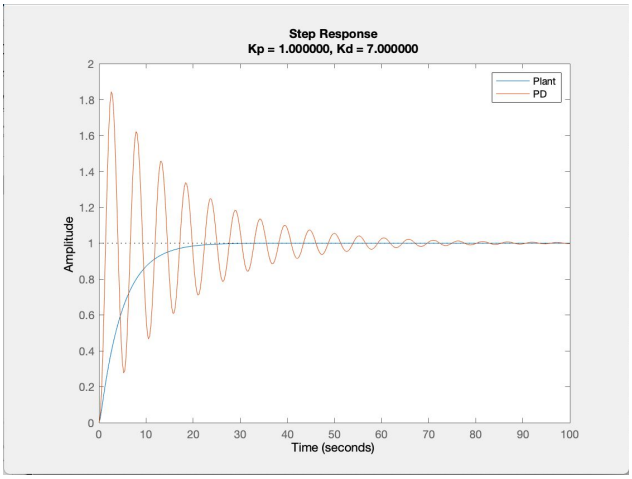
Equação:

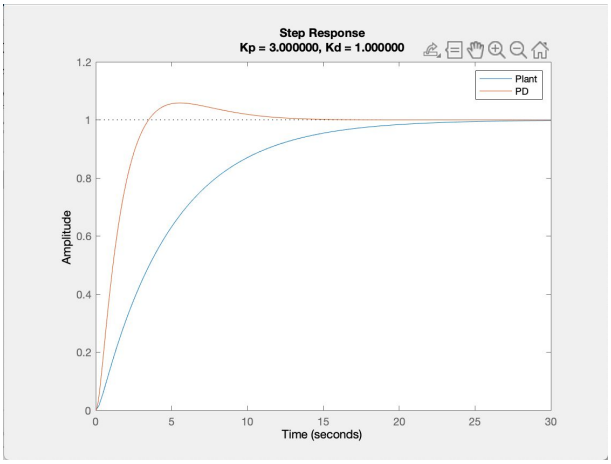
$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt}$$

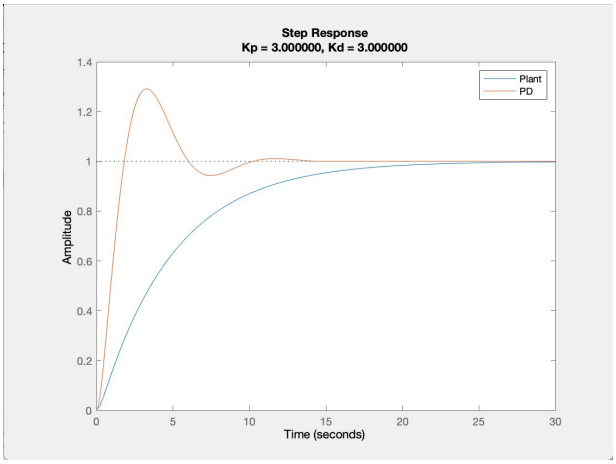
Resultados obtidos para o controlador proporcional derivativo (PD):

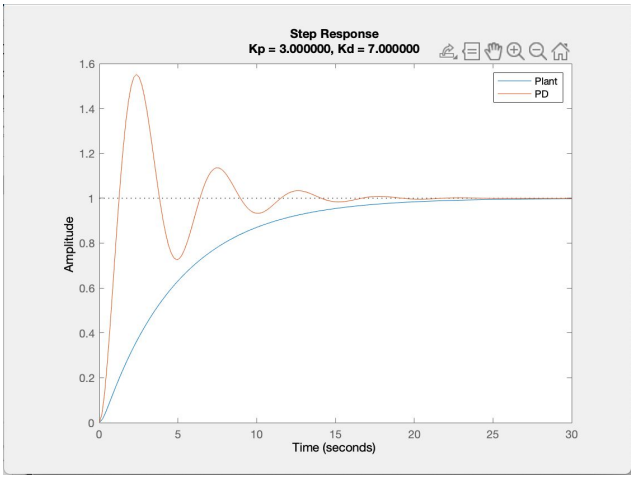


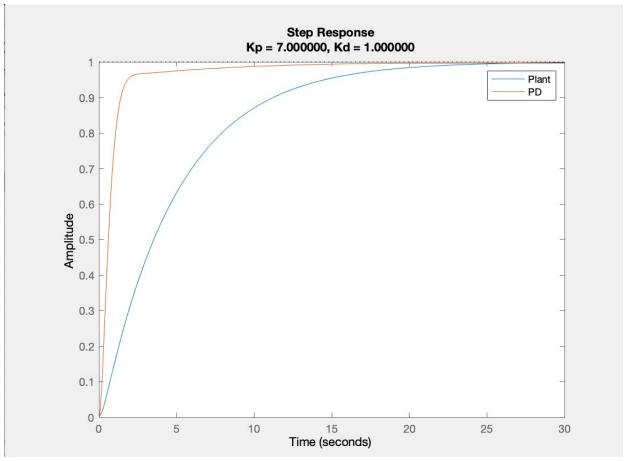


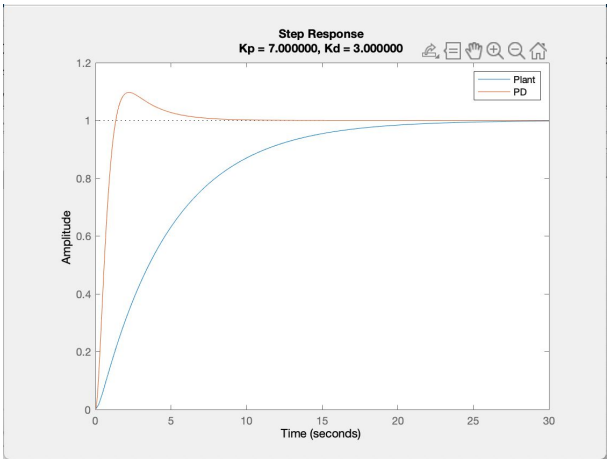


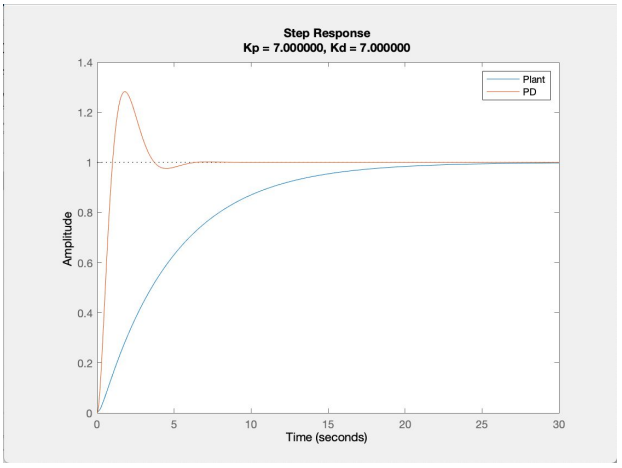












Controlador proporcional integral (PI)

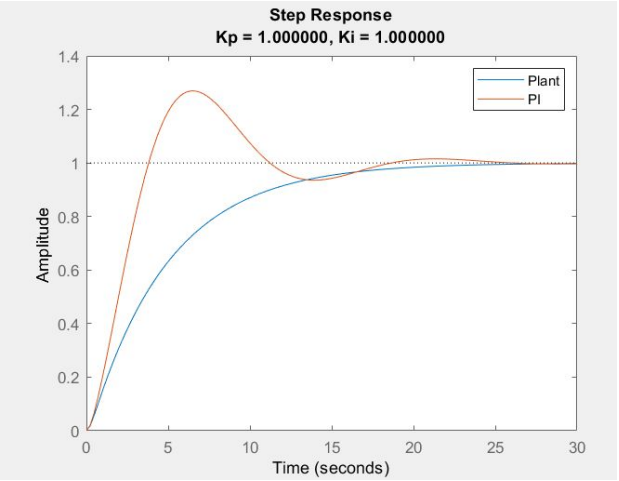
Sistemas de controle PI formam a entrada da planta à partir da soma do erro e da integral do erro no tempo. Diferentemente dos controladores P que fazem uso apenas do erro como variável de controle.

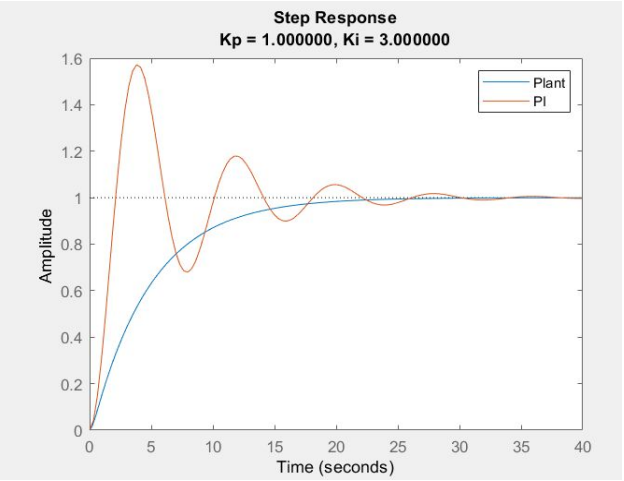
A integral do erro no tempo permite que a saída possa acompanhar a entrada com erro muito menor, chegando a zero, dependendo da planta. Este controlador tende a zerar o erro, porém também pode desestabilizar o sistema se a ação integral for muito acentuada. A ação integral corrige o valor da variável manipulada em intervalos regulares, chamado tempo integral. Se o Ki é baixo, o sistema pode levar muito tempo para atingir o valor de referência. No entanto, se o Ki for muito alto, o sistema pode tornar-se instável.

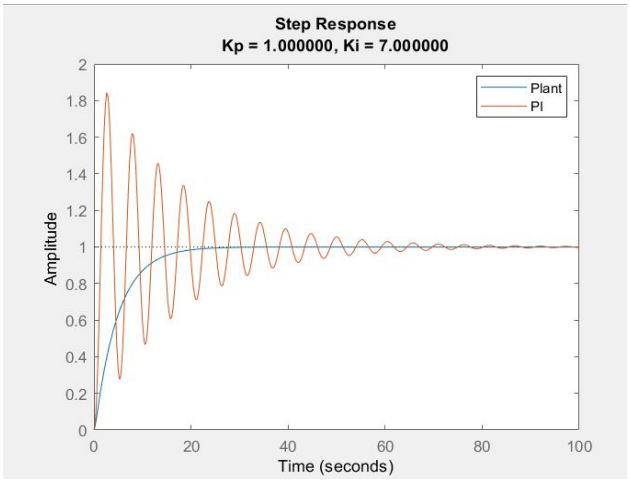
Equação:

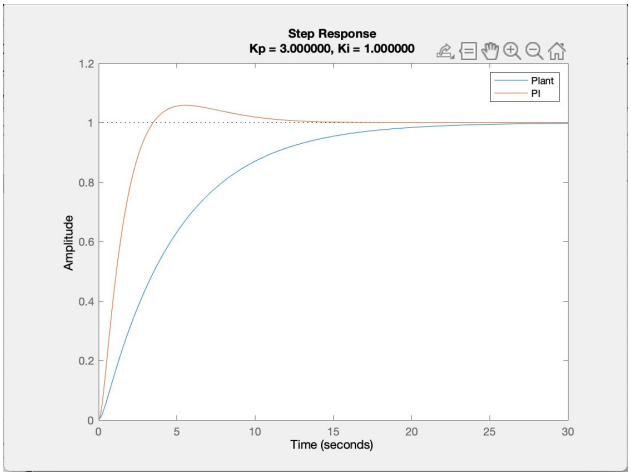
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau$$

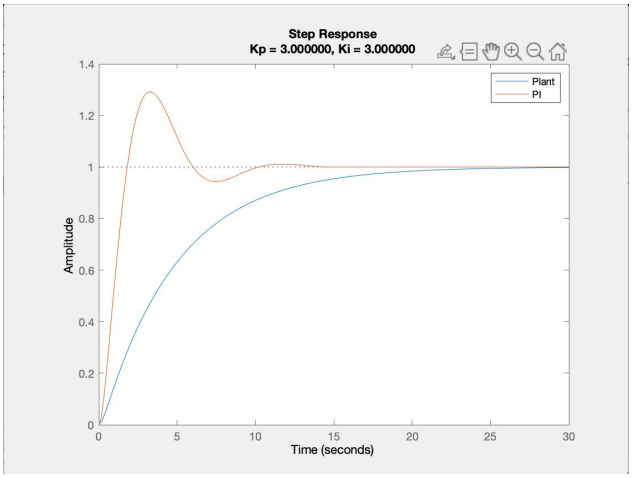
Resultados obtidos para o controlador proporcional integral (PI)

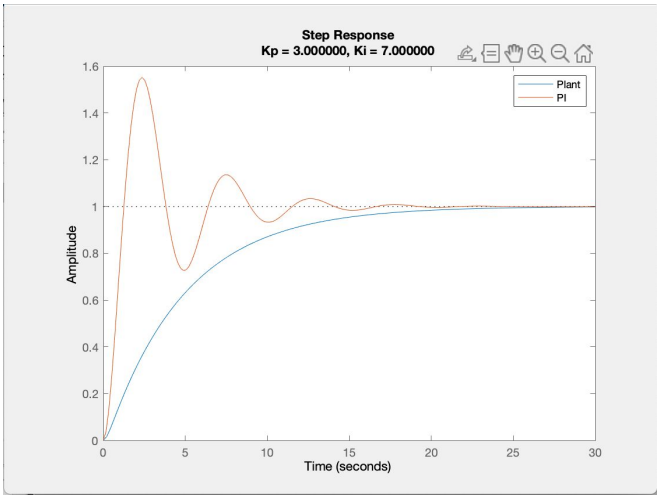


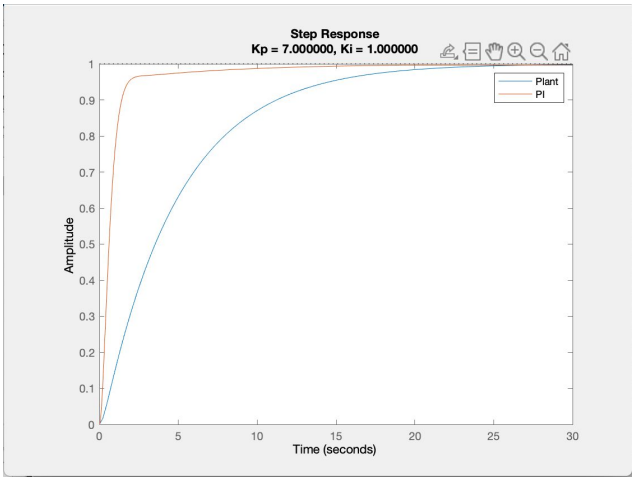


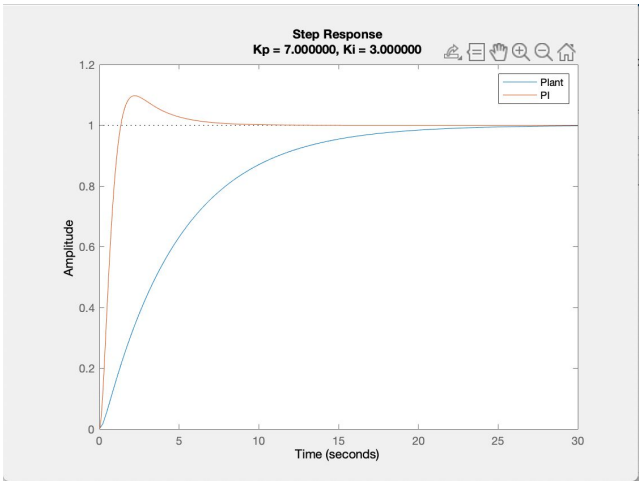


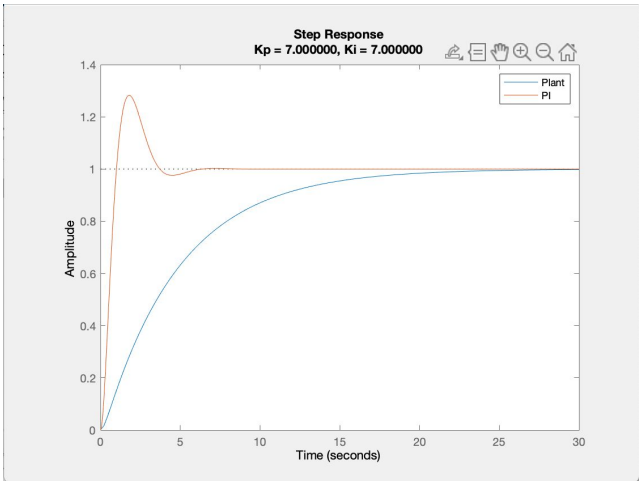












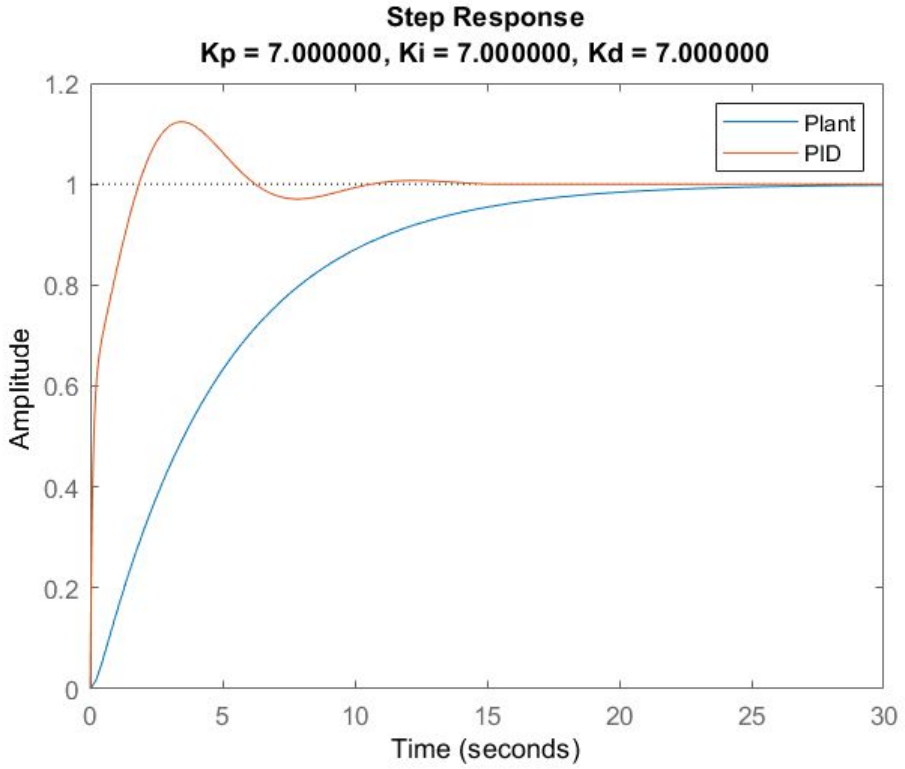
Controlador proporcional integral derivativo (PID)

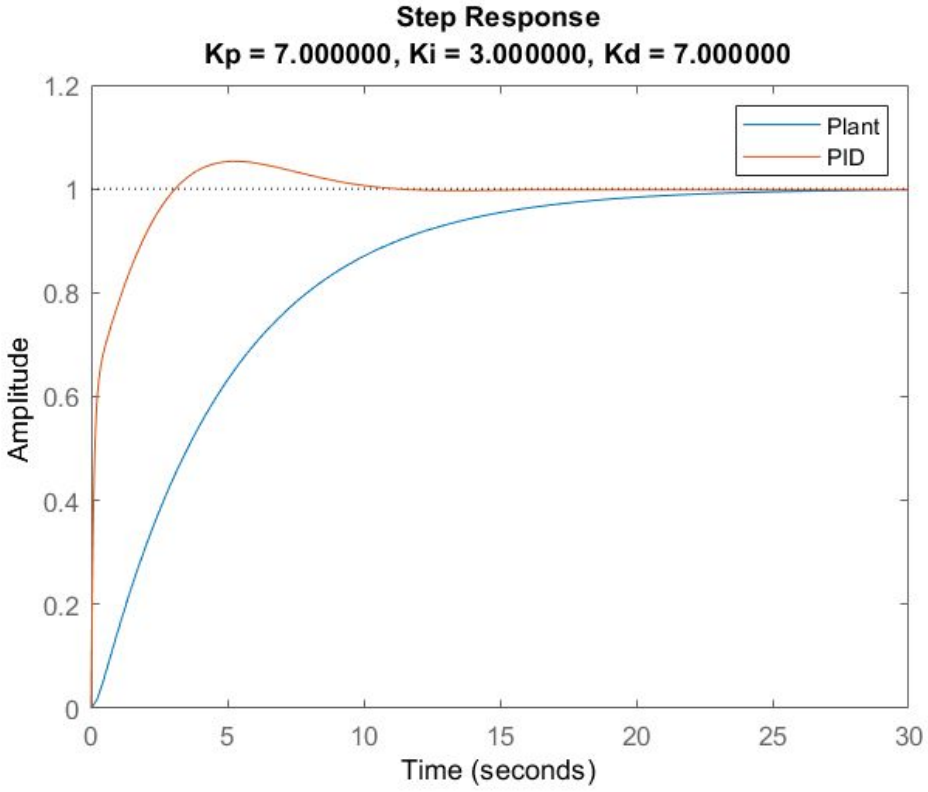
No controlador PID o valor de Kd não surtiu muito efeito, apenas mudando levemente o tempo e a resposta e o pico máximo, mas seu efeito é praticamente irrelevante em comparação à alterações em Kd e Ki. Para esses parâmetros o comportamento foi semelhante ao controlador PD e PI, onde aumentar o valor de Ki diminui o tempo de resposta, mas acrescenta overshooting e oscilação. Da mesma forma, diminuir o Kp acentua esses problemas.

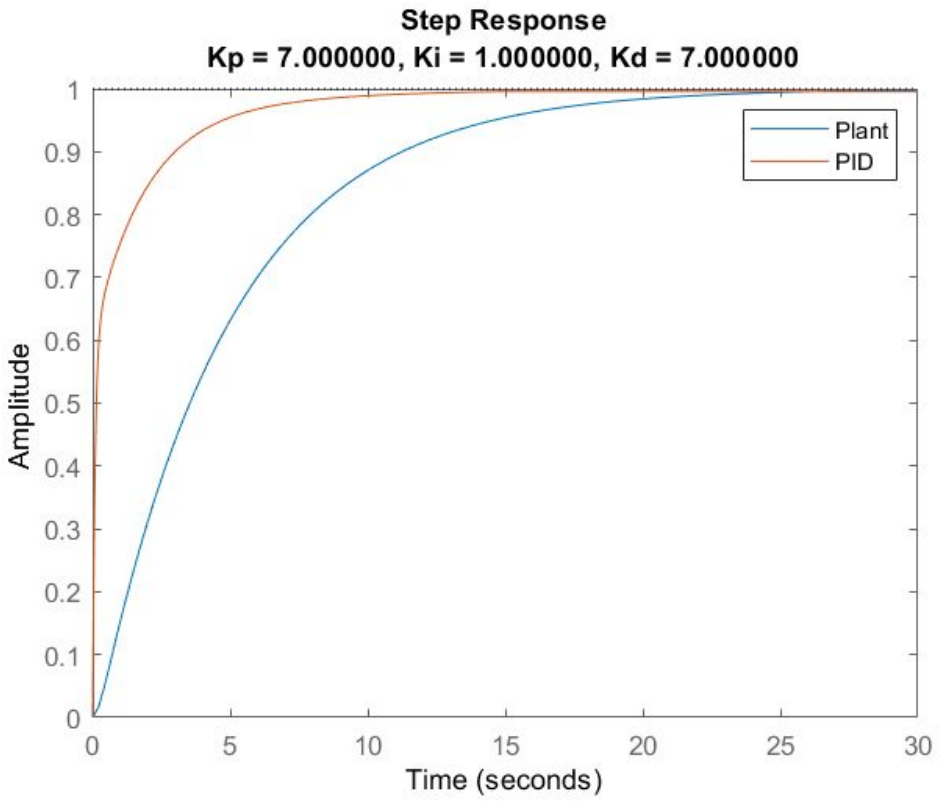
Equação:

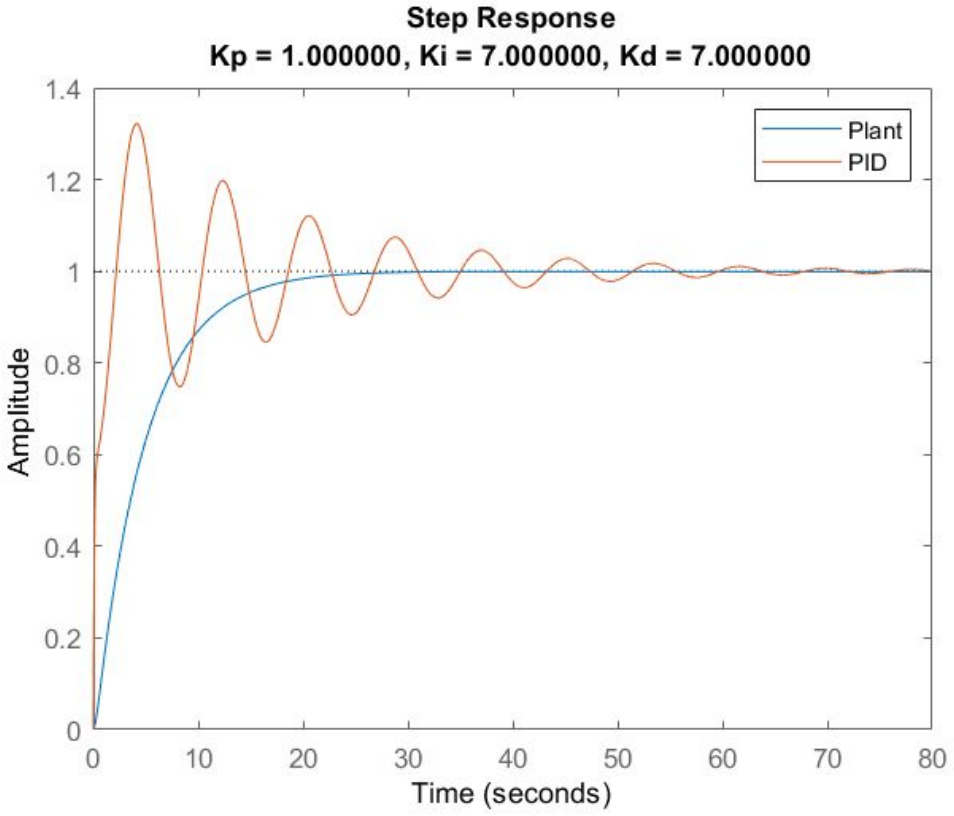
$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int_0^t e(\tau) d\tau$$

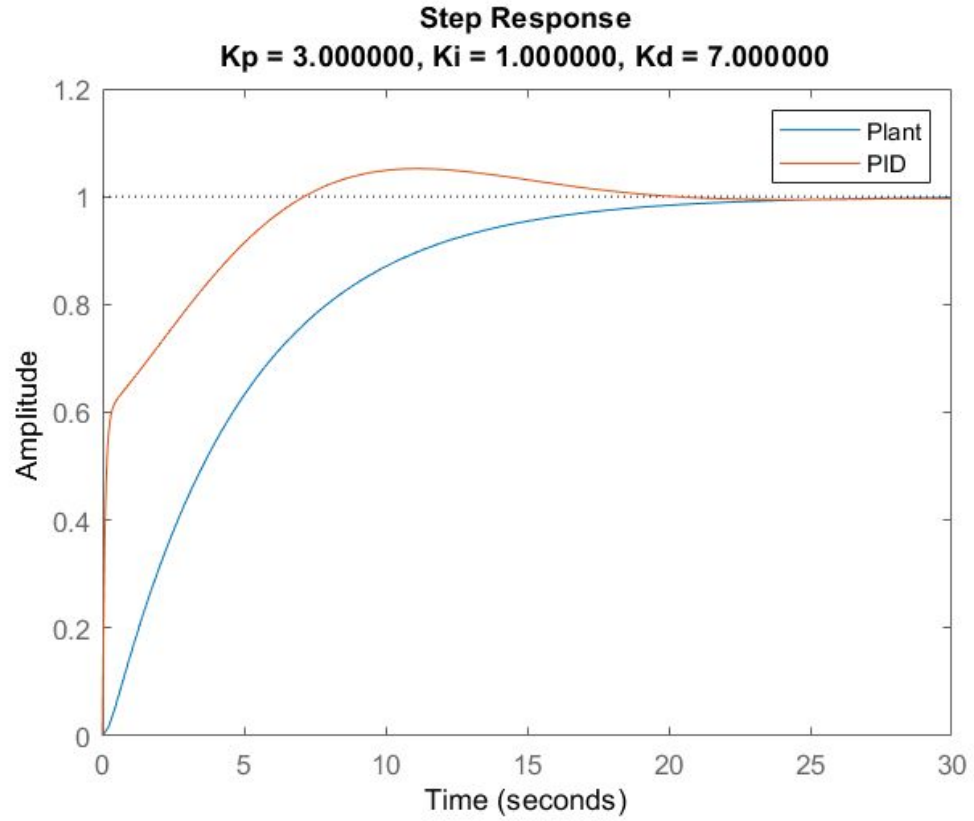
Resultados obtidos para o controlador proporcional integral derivativo (PID)

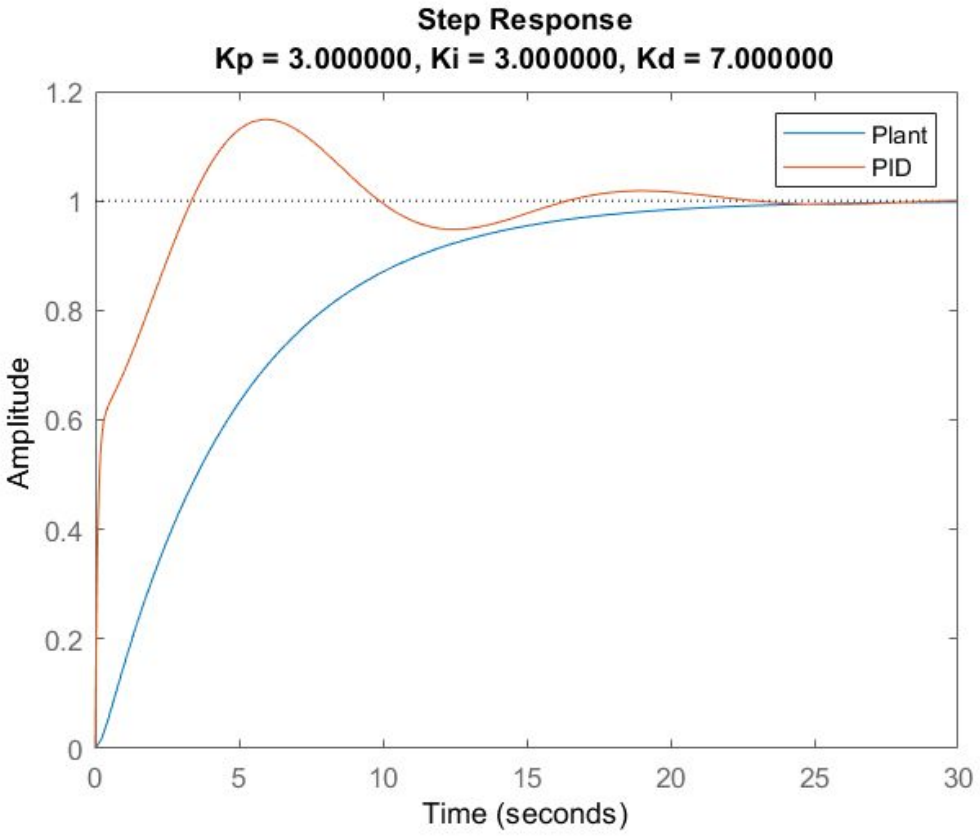


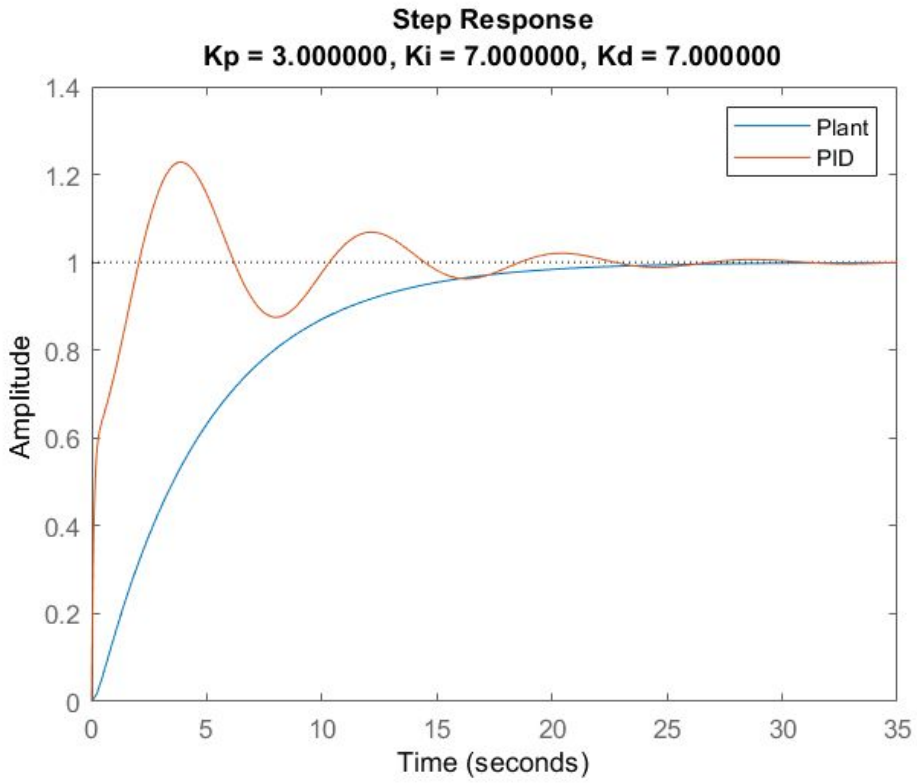



Step Response
Kp = 7.000000, Ki = 1.000000, Kd = 7.000000
Plant
PID
Amplitude
Time (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5
10
15
20
25
30


Step Response
Kp = 1.000000, Ki = 7.000000, Kd = 7.000000
Plant
PID
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
10
20
30
40
50
60
70
80


Step Response
Kp = 3.000000, Ki = 1.000000, Kd = 7.000000
Plant
PID
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
0
5
10
15
20
25
30


Step Response
Kp = 3.000000, Ki = 3.000000, Kd = 7.000000
Plant
PID
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
0
5
10
15
20
25
30



No controlador tipo P, é notável que quanto maior o valor de Kp mais rápida é a resposta do sistema, assim como a amplitude da convergência.

Os controladores PI e PD se comportaram de forma muito semelhante. Nesse controladores quanto maior o valor de Ki e Kd, mais rápido o sistema respondeu, porém a resposta começa a apresentar problemas de overshooting e oscilação. Se o valor de P fica muito baixo os efeitos oscilatórios aparecem mais.

Questão 2

link do dataset escolhido: https://archive.ics.uci.edu/ml/datasets/Ionosphere

Informações do dataset

Esses dados de radar foram coletados por um sistema em Goose Bay, Labrador. Este sistema consiste em uma matriz em fase de 16 antenas de alta frequência com uma potência total de transmissão da ordem de 6,4 quilowatts. Os alvos eram elétrons livres na ionosfera. Retornos de radar "bons" são aqueles que mostram evidências de algum tipo de estrutura na ionosfera. Retornos "ruins" são aqueles que seus sinais não passam pela ionosfera.

Os sinais recebidos foram processados usando uma função de autocorrelação cujos argumentos são o tempo de um pulso e o número do pulso. Havia 17 números de pulso para o sistema Goose Bay. As instâncias nesta base de dados são descritas por 2

atributos por número de pulso, correspondendo aos valores complexos retornados pela função resultante do sinal eletromagnético complexo.

Os parâmetros utilizados para montar os cenários de treinamento e validação foram:

- Taxa de aprendizagem: 0.1, 0.5 ou 1;
- Função de treinamento: **Scaled conjugate gradient backpropagation** (trainscg) ou **Gradient descent with momentum** (traingdx);
- Tamanho da camada escondida: 20 ou 50;
- Número de épocas: 50 ou 100;

Foi realizado uma combinação simples com todos os parâmetros, totalizando 24 cenários.

Log de execução no matlab

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: trainscg
Taxa de aprendizado: 0.100000
Número de épocas: 50
taxa de acerto - média: 91.132075
taxa de acerto - desvio padrão: 3.778822

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: trainscg
Taxa de aprendizado: 0.100000
Número de épocas: 100
taxa de acerto - média: 91.132075
taxa de acerto - desvio padrão: 4.539643

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: trainscg
Taxa de aprendizado: 0.500000
Número de épocas: 50
taxa de acerto - média: 90.000000
taxa de acerto - desvio padrão: 4.625956

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: trainscg
Taxa de aprendizado: 0.500000
Número de épocas: 100
taxa de acerto - média: 86.981132
taxa de acerto - desvio padrão: 4.307143

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20

Função de treinamento: trainscg
Taxa de aprendizado: 1.000000
Número de épocas: 50
taxa de acerto - média: 91.698113
taxa de acerto - desvio padrão: 3.579937

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: trainscg
Taxa de aprendizado: 1.000000
Número de épocas: 100
taxa de acerto - média: 89.622642
taxa de acerto - desvio padrão: 3.902413

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: traingdx
Taxa de aprendizado: 0.100000
Número de épocas: 50
taxa de acerto - média: 82.830189
taxa de acerto - desvio padrão: 9.721000

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: traingdx
Taxa de aprendizado: 0.100000
Número de épocas: 100
taxa de acerto - média: 88.113208
taxa de acerto - desvio padrão: 8.716990

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: traingdx
Taxa de aprendizado: 0.500000
Número de épocas: 50
taxa de acerto - média: 87.735849
taxa de acerto - desvio padrão: 4.810396

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: traingdx
Taxa de aprendizado: 0.500000
Número de épocas: 100
taxa de acerto - média: 83.773585
taxa de acerto - desvio padrão: 7.236805

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: traingdx

Taxa de aprendizado: 1.000000
Número de épocas: 50
taxa de acerto - média: 88.679245
taxa de acerto - desvio padrão: 5.186296

Treinando rede com os parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: traingdx
Taxa de aprendizado: 1.000000
Número de épocas: 100
taxa de acerto - média: 85.471698
taxa de acerto - desvio padrão: 7.006301

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: trainscg
Taxa de aprendizado: 0.100000
Número de épocas: 50
taxa de acerto - média: 90.188679
taxa de acerto - desvio padrão: 3.182166

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: trainscg
Taxa de aprendizado: 0.100000
Número de épocas: 100
taxa de acerto - média: 90.188679
taxa de acerto - desvio padrão: 4.517808

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: trainscg
Taxa de aprendizado: 0.500000
Número de épocas: 50
taxa de acerto - média: 89.622642
taxa de acerto - desvio padrão: 3.902413

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: trainscg
Taxa de aprendizado: 0.500000
Número de épocas: 100
taxa de acerto - média: 90.000000
taxa de acerto - desvio padrão: 5.835854

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: trainscg
Taxa de aprendizado: 1.000000

Número de épocas: 50
taxa de acerto - média: 88.867925
taxa de acerto - desvio padrão: 3.138358

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: trainscg
Taxa de aprendizado: 1.000000
Número de épocas: 100
taxa de acerto - média: 92.264151
taxa de acerto - desvio padrão: 2.427706

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: traingdx
Taxa de aprendizado: 0.100000
Número de épocas: 50
taxa de acerto - média: 86.603774
taxa de acerto - desvio padrão: 10.945926

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: traingdx
Taxa de aprendizado: 0.100000
Número de épocas: 100
taxa de acerto - média: 82.830189
taxa de acerto - desvio padrão: 13.265819

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: traingdx
Taxa de aprendizado: 0.500000
Número de épocas: 50
taxa de acerto - média: 89.056604
taxa de acerto - desvio padrão: 5.681303

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: traingdx
Taxa de aprendizado: 0.500000
Número de épocas: 100
taxa de acerto - média: 85.849057
taxa de acerto - desvio padrão: 7.664187

Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: traingdx
Taxa de aprendizado: 1.000000
Número de épocas: 50

taxa de acerto - média: 90.566038
taxa de acerto - desvio padrão: 2.178680

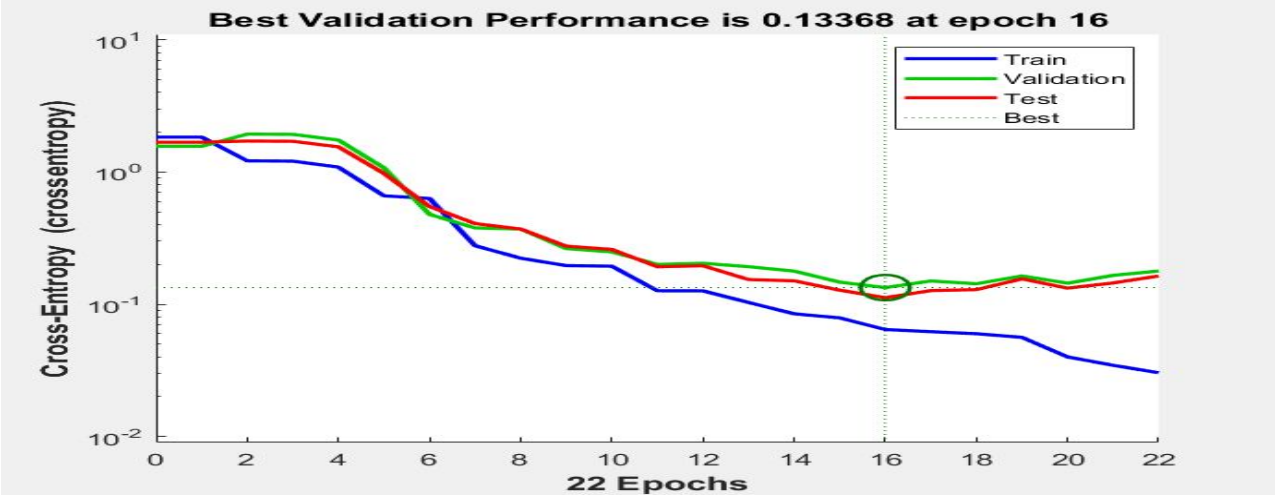
Treinando rede com os parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: traingdx
Taxa de aprendizado: 1.000000
Número de épocas: 100
taxa de acerto - média: 86.415094
taxa de acerto - desvio padrão: 6.401515

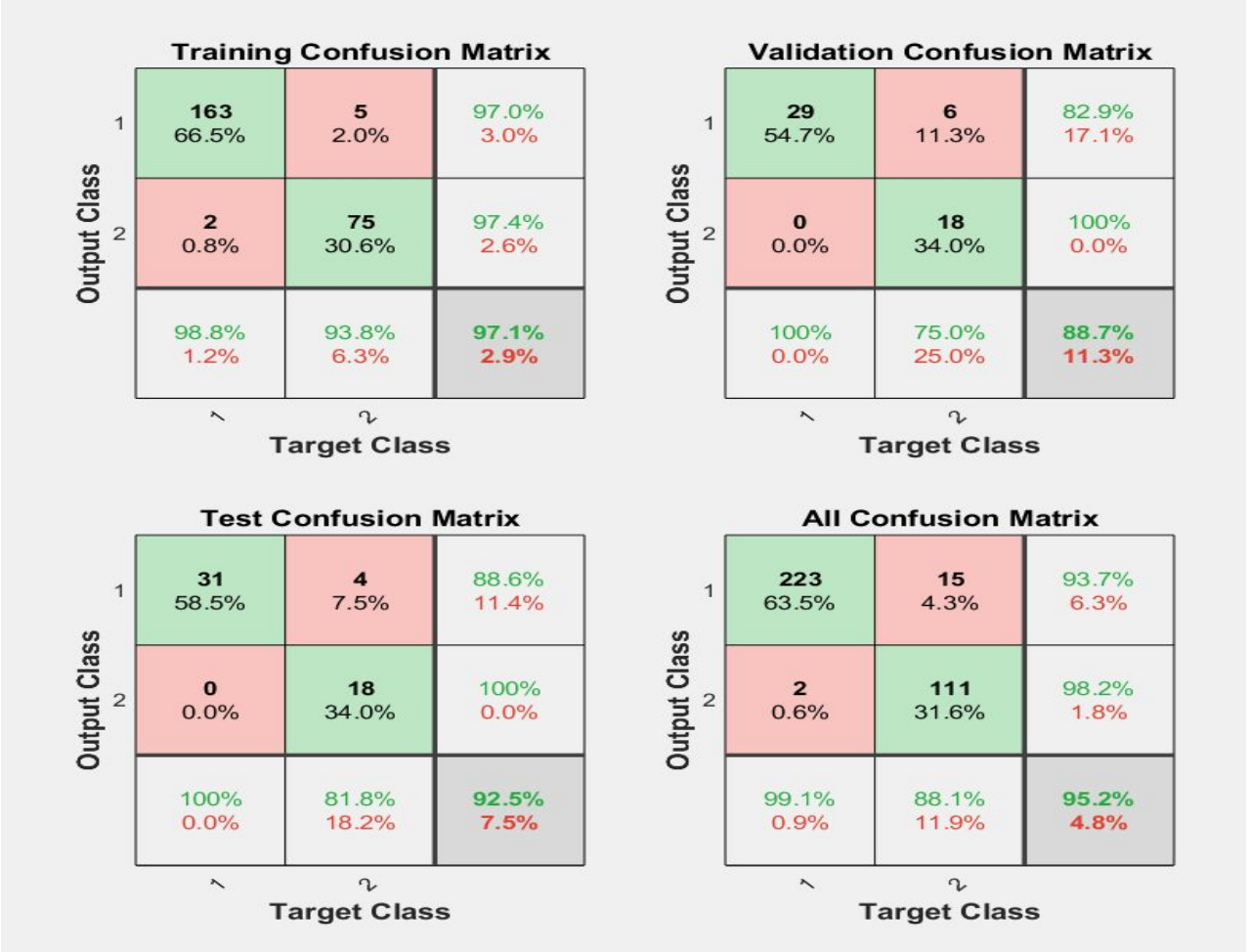
Total de 24 cenários avaliados.

Analisando os resultados obtidos após a combinação dos parâmetros, é possível notar que a função de treinamento tem grande impacto na eficiência geral da rede. Os dados utilizados para o treinamento foram escolhidos de forma aleatória, logo os valores presentes no log podem sofrer variações consideráveis em consequentes execuções no matlab. Para cada cada uma das funções de treinamento, foi selecionado o melhor e o pior cenário de desempenho para essa execução. As matrizes de confusão bem como os parâmetros do cenário podem ser observados abaixo:

Melhor resultado obtido com a função trainscg:

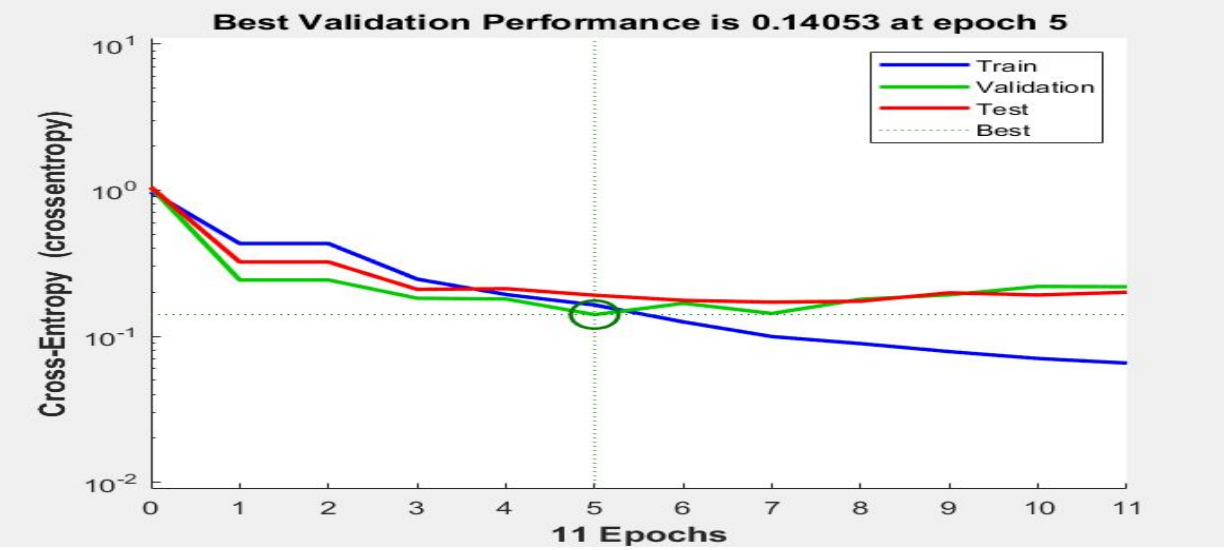
Parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: trainscg
Taxa de aprendizado: 1.000000
Número de épocas: 100
taxa de acerto - média: 92.264151
taxa de acerto - desvio padrão: 2.427706

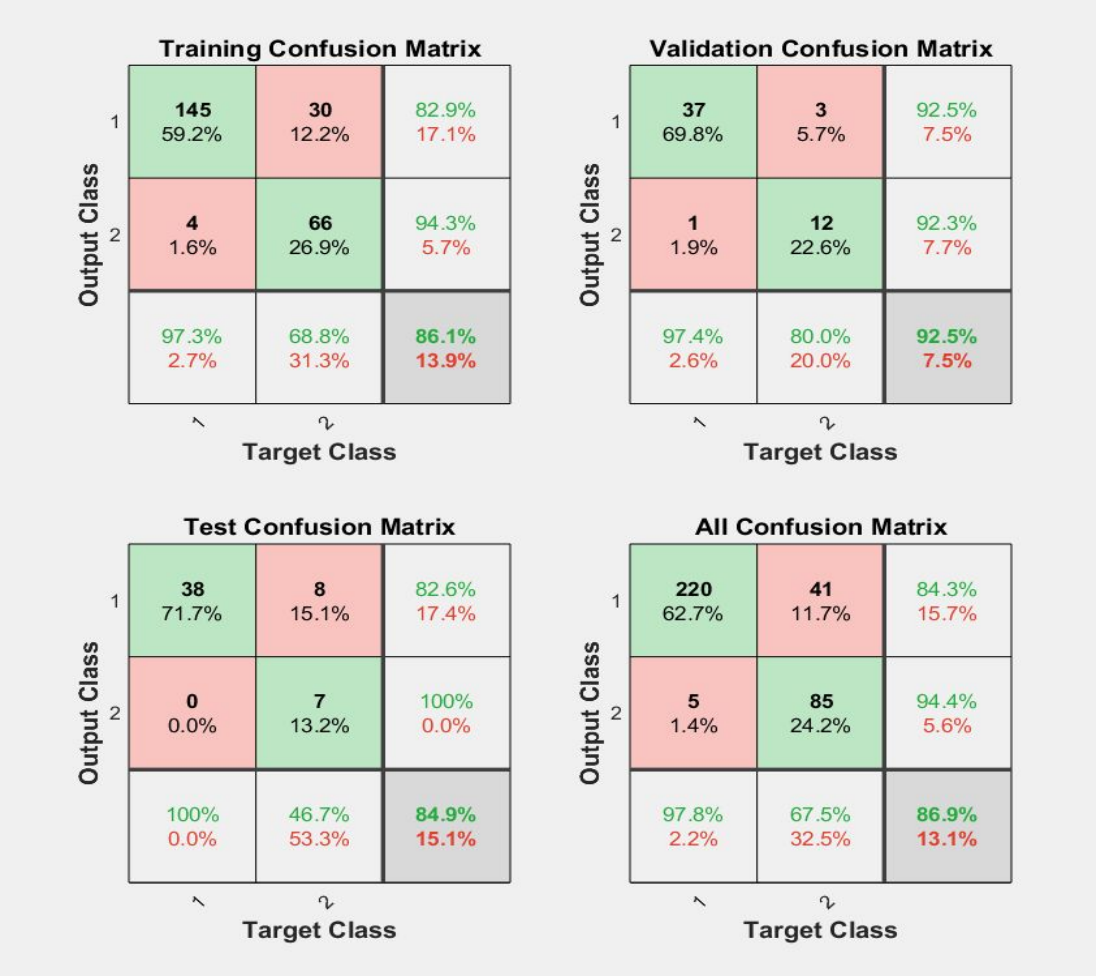




Pior resultado obtido com a função trainscg:

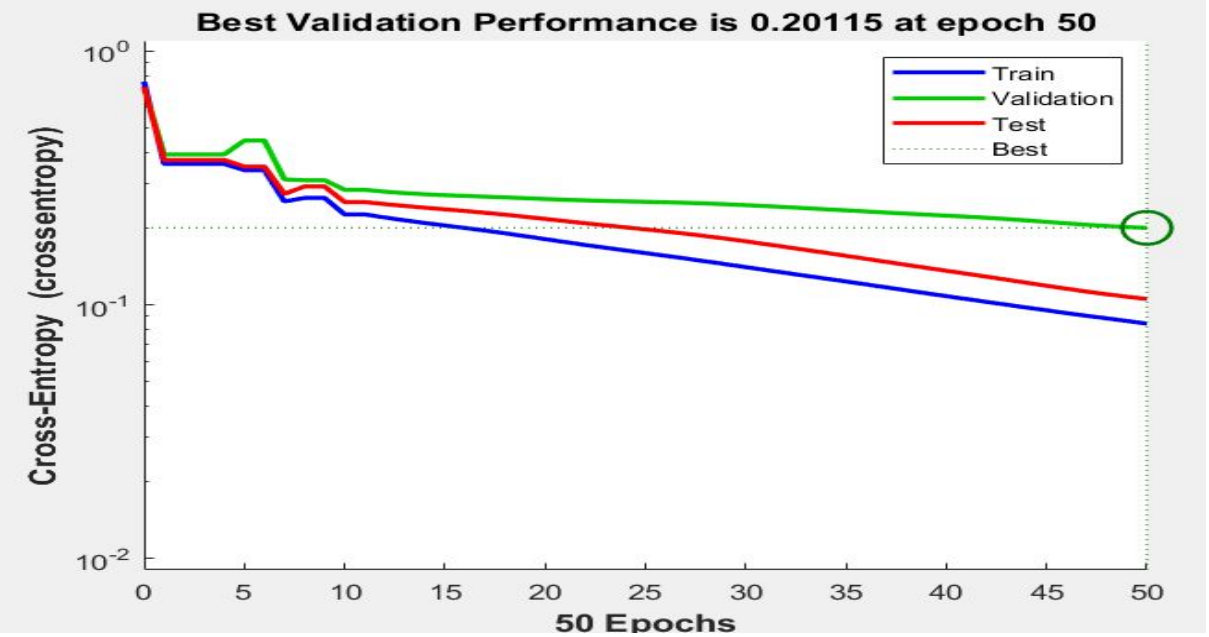
Parâmetros:
Tamanho da camada escondida: 20
Função de treinamento: trainscg
Taxa de aprendizado: 0.500000
Número de épocas: 100
taxa de acerto - média: 86.981132
taxa de acerto - desvio padrão: 4.307143

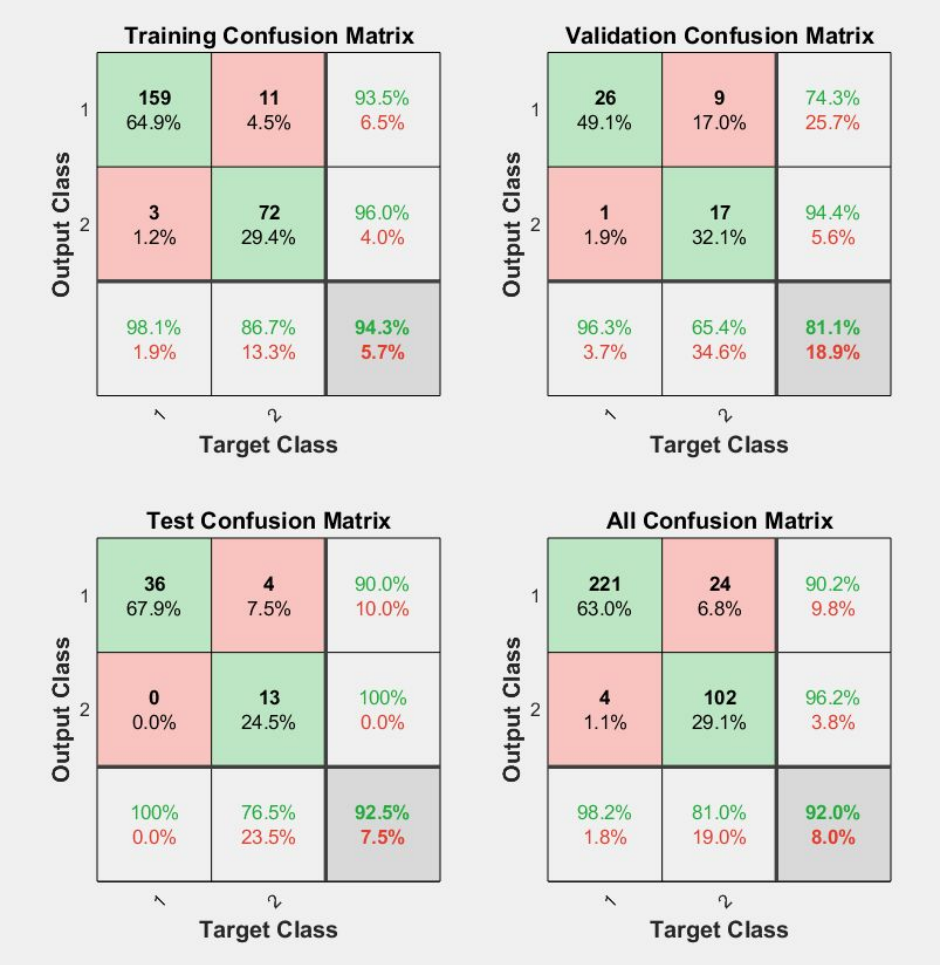




Melhor resultado obtido com a função traingdx:

Parâmetros:
Tamanho da camada escondida: 50
Função de treinamento: traingdx
Taxa de aprendizado: 1.000000
Número de épocas: 50
taxa de acerto - média: 90.566038
taxa de acerto - desvio padrão: 2.178680





Pior resultado obtido com a função traingdx:

Parâmetros:

Tamanho da camada escondida: 20

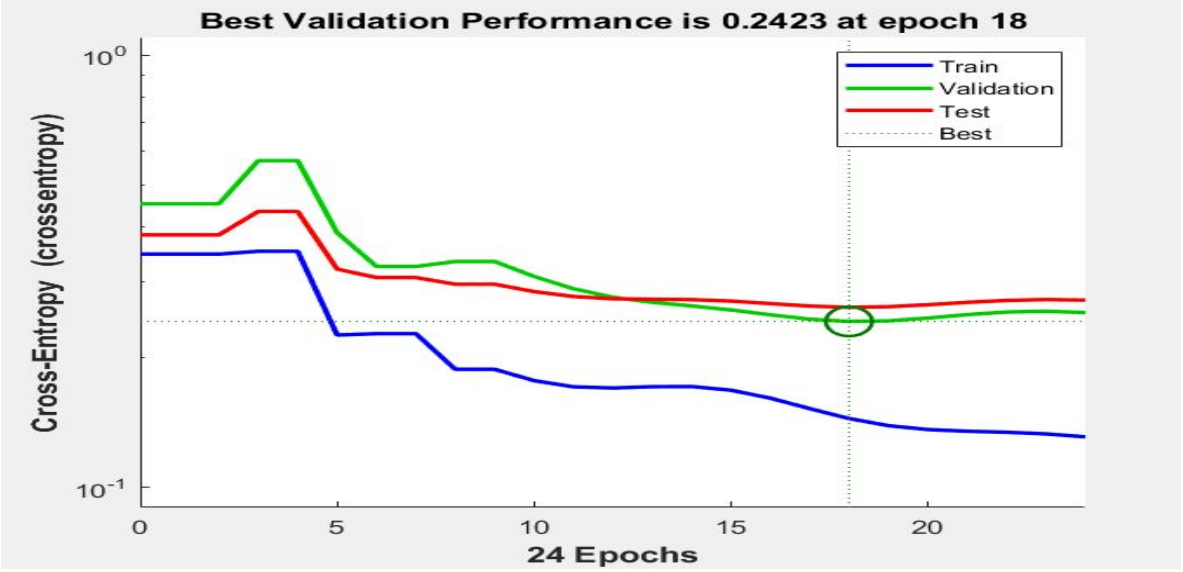
Função de treinamento: traingdx

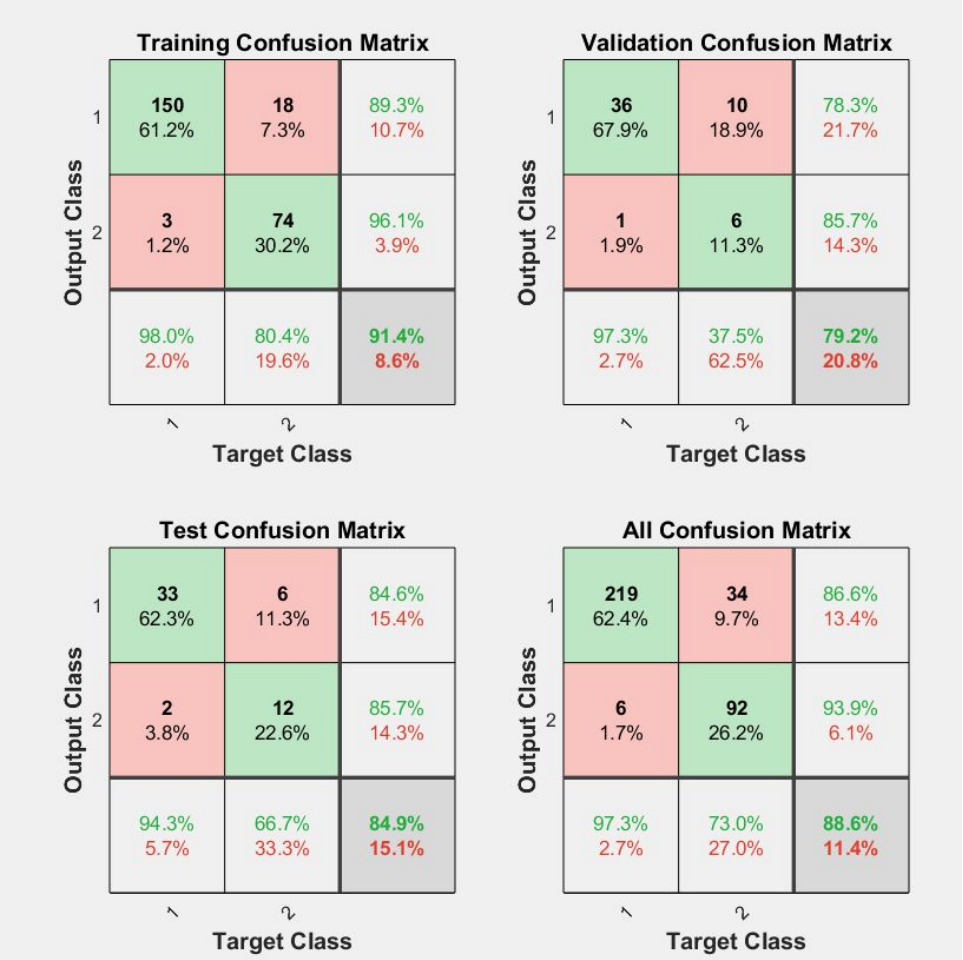
Taxa de aprendizado: 1.000000

Número de épocas: 100

taxa de acerto - média: 85.094340

taxa de acerto - desvio padrão: 4.908079





Analisando os 2 algoritmos de treinamento, percebemos que o número de neurônios na camada escondida também influenciou no melhor e pior resultado para ambos os algoritmos de treinamento. Além disso, é constatado através dos logs do matlab que a função de treinamento **trainscg** se mostrou mais eficiente na maioria das vezes que a **traingdx** para a classificação dos dados no dataset escolhido.