

# PROJETO E IMPLEMENTAÇÃO DA MODULAÇÃO *SPACE VECTOR* EM 3 NÍVEIS PARA O CONVERSOR NPC TRIFÁSICO

TIARLES GUTERRES<sup>1</sup>

## SUMÁRIO

1 Objetivos	2
2 Desenvolvimento	2
3 Resultados	10

## LISTA DE FIGURAS

Figura 1	O conversor NPC 3 níveis. . . . .	2
Figura 2	O diagrama Space Vector com realce nos (a) e (b) vetores, (c) setores e (d) sextantes.	3
Figura 3	Código em <i>Python</i> para o cálculo dos <i>dwell times</i> para o sextante um. . . . .	4
Figura 4	Retas r1, r2 e r3 para a identificação dos sextantes de acordo com as equações (4), (5) e (6). . . . .	5
Figura 5	Trecho de código em C que mostra o cálculo das equações da reta, apresentadas na Figura 4 com a lógica de identificação dos sextantes. . . . .	5
Figura 6	Captura de um projeto feito para os setores 1, 2, 3, 4, 5 e 6, correspondente ao mesmo setor quando rotacionados em múltiplos de 60°. . . . .	6
Figura 7	Parte final do algoritmo de modulação com a troca dos valores dos comparadores como mostrou a Figura 6 e a Tabela 2. . . . .	7
Figura 8	Retas R1, R2 e R3 para a identificação dos setores no sextante um de acordo com as equações (7), (8) e (9). . . . .	7
Figura 9	Trecho de código que mostra o cálculo das equações da reta, apresentadas na Figura 8 com a lógica de identificação dos setores para o sextante 1 com as referências rotacionadas para este. . . . .	8
Figura 10	Trecho de código com a atribuição dos valores de <i>dwell times</i> , com projeto que mostra a contribuição de cada um dos vetores para síntese da referência, como foi mostrado na Figura 6 . . . . .	9
Figura 11	O conversor NPC trifásico com as tensões de entrada balanceadas, a rotina C com a SVM e uma carga RL para conexão das fases ( $R = 1\Omega$ e $L = 3mH$ ). . . . .	10
Figura 12	Resultados obtidos para índices de modulação diferentes. . . . .	11

<sup>1</sup> Grupo de Eletrônica de Potência e Controle (GEPOC), UFSM, Santa Maria, Brasil

## 1 OBJETIVOS

O objetivo deste trabalho é projetar e implementar a modulação Space Vector para o conversor NPC trifásico. As plataformas utilizadas foram o Jupyter-Notebook e o Typhoon-HIL Control Center.

## 2 DESENVOLVIMENTO

**O CONVERSOR NPC.** O NPC (em inglês, *Neutral Point Clamped*, Figura 1) é uma topologia conversor multinível. A topologia NPC utilizada possui três níveis.

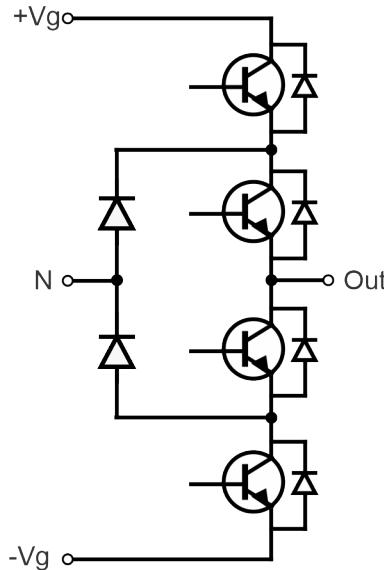


Figura 1: O conversor NPC 3 níveis.

A Tabela 1 mostra como obter a partir dele cada um dos níveis de tensão considerando uma fase do conversor.

Tabela 1: Modos de operação de cada fase do conversor NPC.

Tensão Sintetizada	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub> = $\bar{S}_1$	S <sub>4</sub> = $\bar{S}_2$
-Vg	0	0	1	1
N (Ponto Neutro)	0	1	1	0
+Vg	1	1	0	0

**DIAGRAMA space vector.** O Diagrama *Space Vector* (SV) 3 níveis pode ser visto na Figura 2. A Figura 2a e a 2b mostram os vetores distribuídos no diagrama, a Figura 2c os setores e a Figura 2d os sextante. O diagrama apresenta todas as combinações de síntese possíveis considerando o conjunto trifásico do conversor NPC.

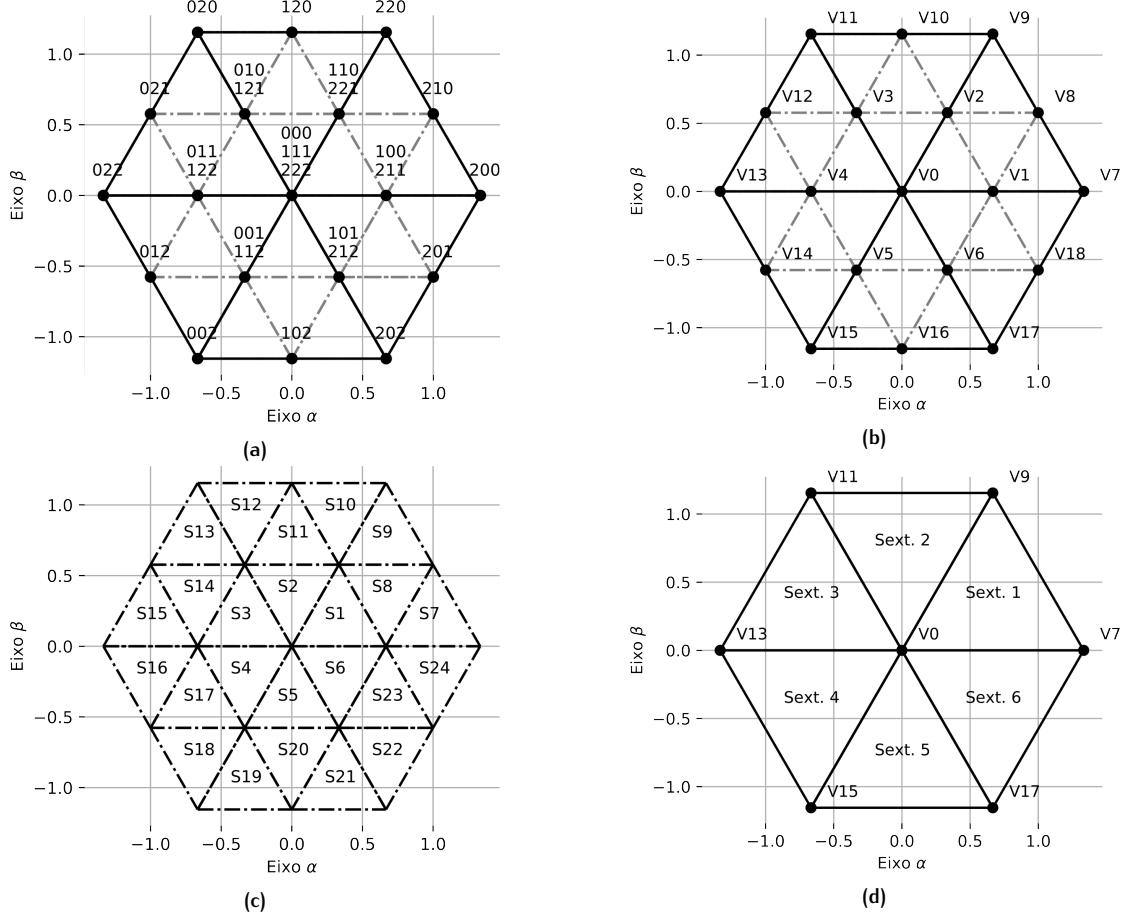


Figura 2: O diagrama Space Vector com realce nos (a) e (b) vetores, (c) setores e (d) sextantes.

**SINTETIZAÇÃO DAS TENSÕES E DWELL TIMES.** Para a SVM precisamos fazer análises do diagrama e a análise da referência para ao fim o conversor poder sintetizar a tensão desejada. A análise do diagrama envolve pré-cálculos que determinam um comportamento (cálculos dos *dwell times*) e a análise da referência deve ser feita em tempo real, considerando a posição desta no *Space Vector* e os *dwell times* citados anteriormente.

Uma simples combinação linear explica como a referência é obtida considerando os *dwell times* ( $dt_{abc}$ ) de cada vetor ( $V_{abc}$ ) em um determinado setor. Como mostra o diagrama, em qualquer setor em que a referência estiver existem três vetores que podem sintetizá-lo, fazendo:

$$dt_a V_a + dt_b V_b + dt_c V_c = V_{ref}, \quad (1)$$

e criar uma segunda relação que determina:

$$dt_a + dt_b + dt_c = 1, \quad (2)$$

ou seja, o tempo somado de todos os vetores deve ser unitário.

Com ambas equações e isolando parte delas podemos montar o sistema abaixo:

$$\begin{bmatrix} dt_a \\ dt_b \\ dt_c \end{bmatrix} = \begin{bmatrix} V_a & V_b & V_c \\ 1 & 1 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} V_{ref} \\ 1 \end{bmatrix}, \quad (3)$$

obtendo uma relação que, a partir do valor dos vetores conhecidos (Figura 2a), resulta nos dwell times para o setor formado pelos vetores  $V_{abc}$ . A Figura 3 mostra o código *Python* descrito para o cálculo dos *dwell times* para todos os setores do sextante um.

```
In [1]: from numpy.linalg import inv
import numpy as np

In [2]: # Considering the sextant 1
V_sx1 = {
    'V0': [0,0,0],
    'V1': [1,0,0],
    'V2': [1,1,0],
    'V7': [2,0,0],
    'V8': [2,1,0],
    'V9': [2,2,0]}
sx1 = {
    'S1': ['V0', 'V1', 'V2'],
    'S7': ['V1', 'V7', 'V8'],
    'S8': ['V1', 'V2', 'V8'],
    'S9': ['V2', 'V8', 'V9']}
for sector in sx1:
    Vs = []
    for vector_name in sx1[sector]:
        vector = V_sx1[vector_name]
        T = 2/3 * np.array([[1, -1/2, -1/2],
                            [0, np.sqrt(3)/2, -np.sqrt(3)/2]])
        VT = vector @ T.transpose()

        Vs.append(VT)
    M = np.vstack([np.array(Vs).transpose(), [1, 1, 1]])
    print('Dwell Time matrix to sector', sector)
    print(inv(M))

Dwell Time matrix to sector S1
[[[-1.5      -0.8660254   1.       ],
 [ 1.5      -0.8660254   0.       ],
 [ 0.       1.73205081  0.       ]]]
Dwell Time matrix to sector S7
[[[-1.5      -0.8660254   2.       ],
 [ 1.5      -0.8660254  -1.       ],
 [ 0.       1.73205081  0.       ]]]
Dwell Time matrix to sector S8
[[[ 0.       -1.73205081  1.       ],
 [-1.5      0.8660254   1.       ],
 [ 1.5      0.8660254  -1.       ]]]
Dwell Time matrix to sector S9
[[[-1.5      -0.8660254   2.       ],
 [ 1.5      -0.8660254   0.       ],
 [ 0.       1.73205081  -1.       ]]]
```

Figura 3: Código em *Python* para o cálculo dos *dwell times* para o sextante um.

O código descreve os vetores do primeiro sextante, define quais destes pertence aos quatro setores deste e, dentro do laço de repetição, verifica cada um dos vetores do setor, converte em coordenadas alfa-beta e faz uma operação de empilhamento matricial vertical (função `np.vstack`) entre os vetores de cada setor e um vetor de '1's'. Por fim, a inversa é calculada com a função `inv()` e impresso para leitura no console.

**IDENTIFICAÇÃO DOS SEXTANTES.** Com noção das dimensões do diagrama e considerando a entrada trifásica de referências convertidas para coordenadas alfa-beta podemos posicioná-la no diagrama SV para sintetizá-lo. A segunda etapa é a análise da referência começando pela definição do sextante onde se encontra a referência.

$$r_1 = V_\beta \quad (4)$$

$$r_2 = V_\beta - \sqrt{3}V_\alpha \quad (5)$$

$$r_3 = V_\beta + \sqrt{3}V_\alpha \quad (6)$$

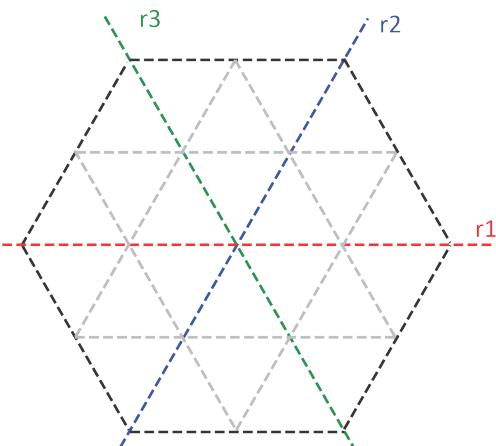


Figura 4: Retas  $r_1$ ,  $r_2$  e  $r_3$  para a identificação dos sextantes de acordo com as equações (4), (5) e (6).

A Figura 4 e as equações (4), (5) e (6) ilustram as retas  $r_1$ ,  $r_2$  e  $r_3$ , onde com as quais, podemos identificar qual a posição angular que a referência se encontra, podendo assim nomear um sextante para esta referência que será utilizado para sintetize simplificada das tensões.

A Figura 5 mostra o trecho de código em C com a implementação desta lógica.

```

8   r1 = V_beta;
9   r2 = V_beta - sqrt3*V_alpha;
10  r3 = V_beta + sqrt3*V_alpha;
11
12 if(r1 > 0 && r2 < 0) sextant = 1;
13 else if(r1 > 0 && r2 > 0 && r3 > 0) sextant = 2;
14 else if(r1 > 0 && r3 < 0) sextant = 3;
15 else if(r1 < 0 && r2 > 0) sextant = 4;
16 else if(r1 < 0 && r3 < 0 && r2 < 0) sextant = 5;
17 else sextant = 6;
```

Figura 5: Trecho de código em C que mostra o cálculo das equações da reta, apresentadas na Figura 4 com a lógica de identificação dos sextantes.

**A MATRIZ DE ROTAÇÃO E A RELAÇÃO GERAL CONSIDERANDO O SEXTANTE UM.** Uma prática que otimiza o projeto de síntese da tensão de referência é trazer a referência para único sextante, no qual o projeto será de fato feito, fazendo apenas simples modificações para que as tensões sejam sintetizadas corretamente.

As Figuras 6a, 6b e 6c mostram o conteúdo dos comparadores considerando o setores  $S_1, S_2, S_3, S_4, S_5$  e  $S_6$  da Figura 2c, quando aplicada a rotação estes se sobrepõem.

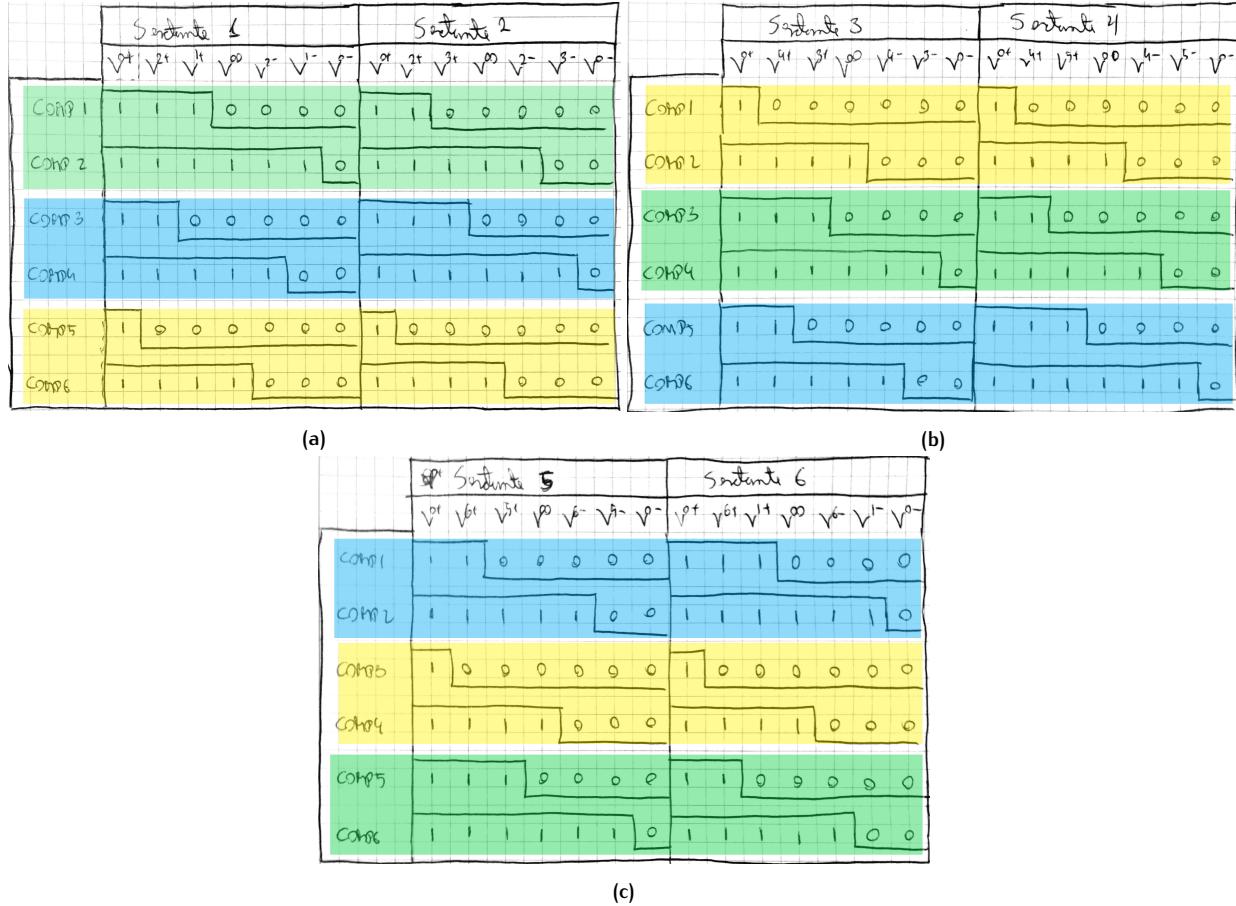


Figura 6: Captura de um projeto feito para os setores 1, 2, 3, 4, 5 e 6, correspondente ao mesmo setor quando rotacionados em múltiplos de  $60^\circ$ .

A faixa colorida nos sinais apresenta uma relação entre os setores dos diferentes sextantes e que se repete para os demais setores quando o diagrama é analisado por completo, podendo assim, montar a Tabela 2 com os valores considerados dos comparadores para os diferentes sextantes. A Tabela leva em conta também as diferenças entre os sextantes que são pares dos que são ímpares, que será mostrado com mais detalhes na implementação.

Tabela 2: Tabela com o conteúdo dos comparadores para os seis sextantes considerando o cálculo dos dois primeiros.

	Sextante 1	Sextante 2	Sextante 3	Sextante 4	Sextante 5	Sextante 6
COMP1	COMP1 Sx. 1	COMP1 Sx.2	COMP5 Sx. 1	COMP5 Sx. 2	COMP3 Sx. 1	COMP3 Sx. 2
COMP2	COMP2 Sx. 1	COMP2 Sx. 2	COMP6 Sx. 1	COMP6 Sx. 2	COMP4 Sx. 1	COMP4 Sx. 2
COMP3	COMP3 Sx. 1	COMP3 Sx. 2	COMP1 Sx. 1	COMP1 Sx. 2	COMP5 Sx. 1	COMP5 Sx. 2
COMP4	COMP4 Sx. 1	COMP4 Sx. 2	COMP2 Sx. 1	COMP2 Sx. 2	COMP6 Sx. 1	COMP6 Sx. 2
COMP5	COMP5 Sx. 1	COMP5 Sx. 2	COMP3 Sx. 1	COMP3 Sx. 2	COMP1 Sx. 1	COMP1 Sx. 2
COMP6	COMP6 Sx. 1	COMP6 Sx. 2	COMP4 Sx. 1	COMP4 Sx. 2	COMP2 Sx. 1	COMP2 Sx. 2

A Figura 7 mostra um trecho final da rotina em C que considera a lógica da Tabela 2.

```

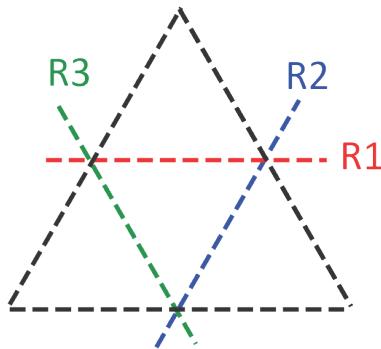
134 if(sextant == 1 || sextant == 2){
135     ... COMP1 = t_COMP1;
136     ... COMP2 = t_COMP2;
137     ... COMP3 = t_COMP3;
138     ... COMP4 = t_COMP4;
139     ... COMP5 = t_COMP5;
140     ... COMP6 = t_COMP6;
141 } else if(sextant == 3 || sextant == 4){
142     ... COMP1 = t_COMP5;
143     ... COMP2 = t_COMP6;
144     ... COMP3 = t_COMP1;
145     ... COMP4 = t_COMP2;
146     ... COMP5 = t_COMP3;
147     ... COMP6 = t_COMP4;
148 } else { //Sextant 5 and 6
149     ... COMP1 = t_COMP3;
150     ... COMP2 = t_COMP4;
151     ... COMP3 = t_COMP5;
152     ... COMP4 = t_COMP6;
153     ... COMP5 = t_COMP1;
154     ... COMP6 = t_COMP2;
155 }

```

**Figura 7:** Parte final do algoritmo de modulação com a troca dos valores dos comparadores como mostrou a Figura 6 e a Tabela 2.

Com isto pode-se apenas rotacionar o vetor de referência para um determinado sextante e identificar, mapear e projetar os setores para apenas aquele.

**IDENTIFICAÇÃO DOS SETORES.** A Figura 8 e as equações (7), (8) e (9) ilustram as retas R1, R2 e R3, onde com as quais, podemos identificar qual a posição no sextante que a referência se encontra.



**Figura 8:** Retas R1, R2 e R3 para a identificação dos setores no sextante um de acordo com as equações (7), (8) e (9).

$$R_1 = V_\beta - \frac{3\sqrt{2}}{8} \quad (7)$$

$$R_2 = V_\beta - \sqrt{3} V_\alpha + \frac{3\sqrt{2}}{4} \quad (8)$$

$$R_3 = V_\beta + \sqrt{3} V_\alpha - \frac{3\sqrt{2}}{4} \quad (9)$$

Para a identificação da referência no sextante um uma rotação utilizando a relação mostrada nas equações (10) e (11) deve ser aplicada.

$$\theta = 60^\circ \times (\text{sextante} - 1) \quad (10)$$

$$\begin{bmatrix} V_{\alpha,1} \\ V_{\beta,1} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} V_\alpha \\ V_\beta \end{bmatrix} \quad (11)$$

A Figura 9 mostra a implementação da rotação, considerando graus radianos ( $60^\circ = \frac{\pi}{3}$ ), o cálculo das retas de referência para definição do setor em que a referência rotacionada se encontra e a lógica para decidir a qual setor ela pertence.

```

28 theta = M_PI/3 * (sextant - 1);
29
30 V_alfa1 = V_alfa*cos(theta) + V_beta*sin(theta);
31 V_beta1 = -V_alfa*sin(theta) + V_beta*cos(theta);
32
33 R1 = V_beta1 - 3*sqrt2/8;
34 R2 = V_beta1 - sqrt3*V_alfa1 + 3*sqrt2/4;
35 R3 = V_beta1 + sqrt3*V_alfa1 - 3*sqrt2/4;
36
37 if(R1>=0)
38 ...sector = 4;
39 else
40 {
41 ... if (R2>=0)
42 {
43 ..... if(R3 >= 0)
44 ..... ....sector = 2;
45 ..... else
46 ..... ....sector = 1;
47 ..... }
48 ... else
49 ... sector = 3;
50 }
51

```

**Figura 9:** Trecho de código que mostra o cálculo das equações da reta, apresentadas na Figura 8 com a lógica de identificação dos setores para o sextante 1 com as referências rotacionadas para este.

A Figura 10 mostra a rotina para cada um dos quatro setores que é definido no código mostrado na Figura 9, neste podemos ver a presença dos *dwell times* calculados no código da Figura 3 e o projeto dos comparadores para os setores S1, S8, S7 e S9 como foi feito na Figura 6.

```
52 if(sector == 1){ //Sector 1
53     dt0 = -1.5*V_alfa1 - sqrt3/2*V_beta1 + 1;
54     dt1 = 1.5*V_alfa1 - sqrt3/2*V_beta1;
55     dt2 = 1.-dt0 - dt1;
56     ...
57     if(sextant == 1 || sextant == 3 || sextant == 5 ){
58         t_COMP1 = .25*dt0+.5*dt2+.5*dt1;
59         t_COMP2 = .75*dt0+ dt2 + dt1;
60         t_COMP3 = .25*dt0+.5*dt2;
61         t_COMP4 = .75*dt0+dt2+.5*dt1;
62         t_COMP5 = .25*dt0;
63         t_COMP6 = .75*dt0+.5*dt1+.5*dt2;
64     } else { //sextant = [2,6]
65         t_COMP1 = -.25*dt0 + .5*dt1;
66         t_COMP2 = .75*dt0+dt1+.5*dt2;
67         t_COMP3 = .25*dt0+.5*dt1+.5*dt2;
68         t_COMP4 = .75*dt0+dt1+dt2;
69         t_COMP5 = .25*dt0;
70         t_COMP6 = .75*dt0+.5*dt1+.5*dt2;
71     }
72 }
```

**(a) Setor 1**

```
72 } else if(sector == 8){ //Sector 8
73     dt1 = - sqrt3*V_beta1 + .1;
74     dt2 = -1.5*V_alfa1 + sqrt3/2*V_beta1 + .1;
75     dt8 = 1 - dt1 - dt2;
76     ...
77     if(sextant == 1 || sextant == 3 || sextant == 5 ){
78         t_COMP1 = .5*dt2 + .5*dt1 + dt8;
79         t_COMP2 = 1;
80         t_COMP3 = .5*dt2;
81         t_COMP4 = dt2+.5*dt1+dt8;
82         t_COMP5 = 0;
83         t_COMP6 = .5*dt2+.5*dt1;
84     } else { //sextant =[2,4,6]
85         t_COMP1 = .5*dt1;
86         t_COMP2 = .5*dt1 + dt2 + dt8;
87         t_COMP3 = .5*dt1+dt2+dt8;
88         t_COMP4 = 1;
89         t_COMP5 = 0;
90         t_COMP6 = .5*dt1+.5*dt2;
91     }
}
```

(b) Setor 8

```
92 } else if(sector == 7){ //Sector 7
93     dt1 = -1.5*V_alpha1 - sqrt3/2*V_beta1 + 2;
94     dt7 = -1.5*V_alpha1 - sqrt3/2*V_beta1 - 1;
95     dt8 = 1 - dt1 - dt7;
96     ...
97     if(sextant == 1 || sextant == 3 || sextant == 5){
98         t_COMP1 = .5*dt1+dt8+dt7;
99         t_COMP2 = 1;
100        t_COMP3 = 0;
101        t_COMP4 = .5*dt1+dt8;
102        t_COMP5 = 0;
103        t_COMP6 = .5*dt1;
104    } else { //sextant=[2,4,6]
105        t_COMP1 = .5*dt1+dt7;
106        t_COMP2 = 1;
107        t_COMP3 = .5*dt1+dt7+dt8;
108        t_COMP4 = 1;
109        t_COMP5 = 0;
110        t_COMP6 = .5*dt1;
111    }
}
```

(c) Setor 7

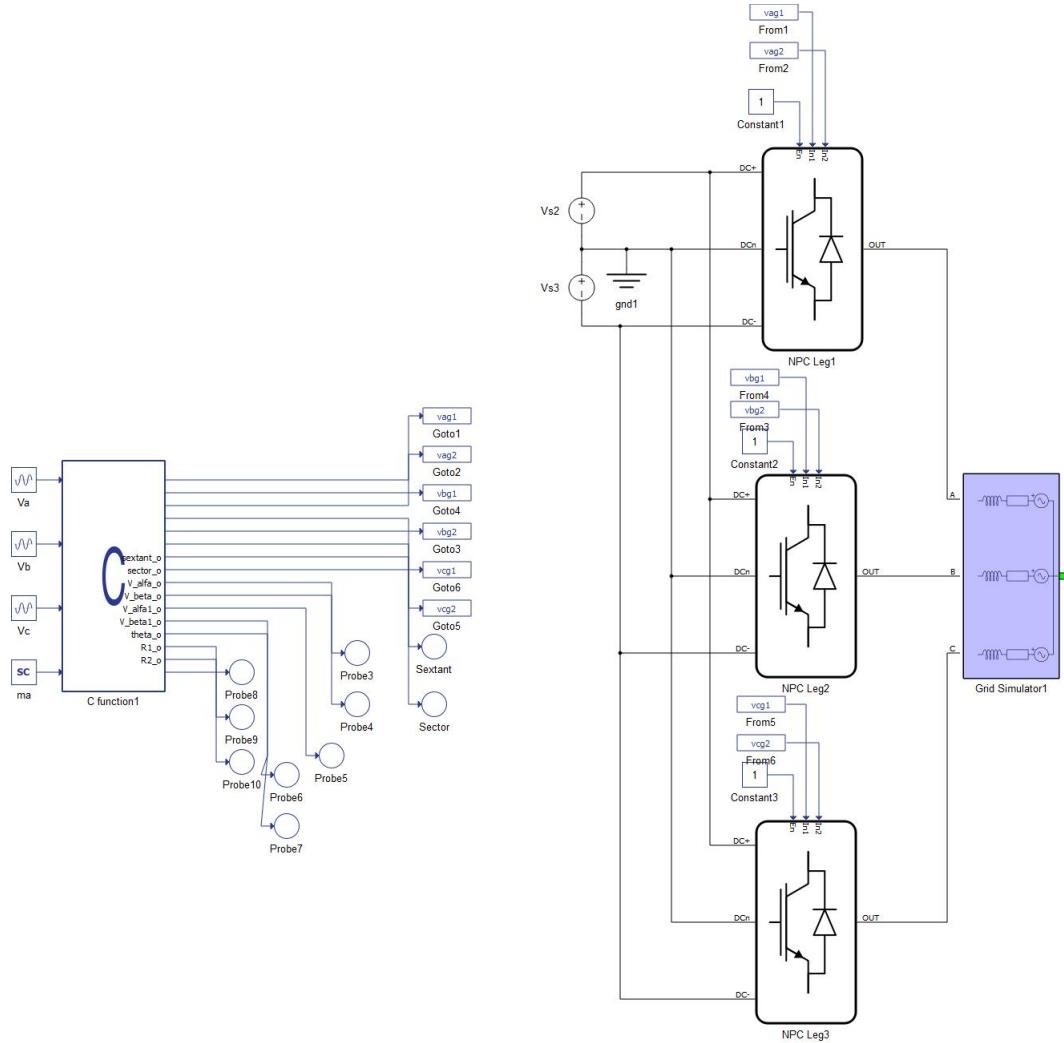
```
112 } else if(sector == 9) //Sector 9....  
113 dt2 = -.5*V_alfa1 - sqrt3/2*V_beta1 + 2;  
114 dt8 = -.5*V_alfa1 - sqrt3/2*V_beta1;  
115 dt9 = 1 - dt2 - dt8;  
116  
117 if(sextant == 1 || sextant == 3 || sextant == 5) {  
118 .....t_COMP1 = .5*dt2 + dt9+dt8;  
119 .....t_COMP2 = 1;  
120 .....t_COMP3 = .5*dt2+dt9;  
121 .....t_COMP4 = 1;  
122 .....t_COMP5 = 0;  
123 .....t_COMP6 = .5*dt2;  
124 } else { //sextant = [2,4,6]  
125 .....t_COMP1 = 0;  
126 .....t_COMP2 = .5*dt2+dt8;  
127 .....t_COMP3 = .5*dt2+dt8+dt9;  
128 .....t_COMP4 = 1;  
129 .....t_COMP5 = 0;  
130 .....t_COMP6 = .5*dt2;  
131 }
```

(d) Setor c

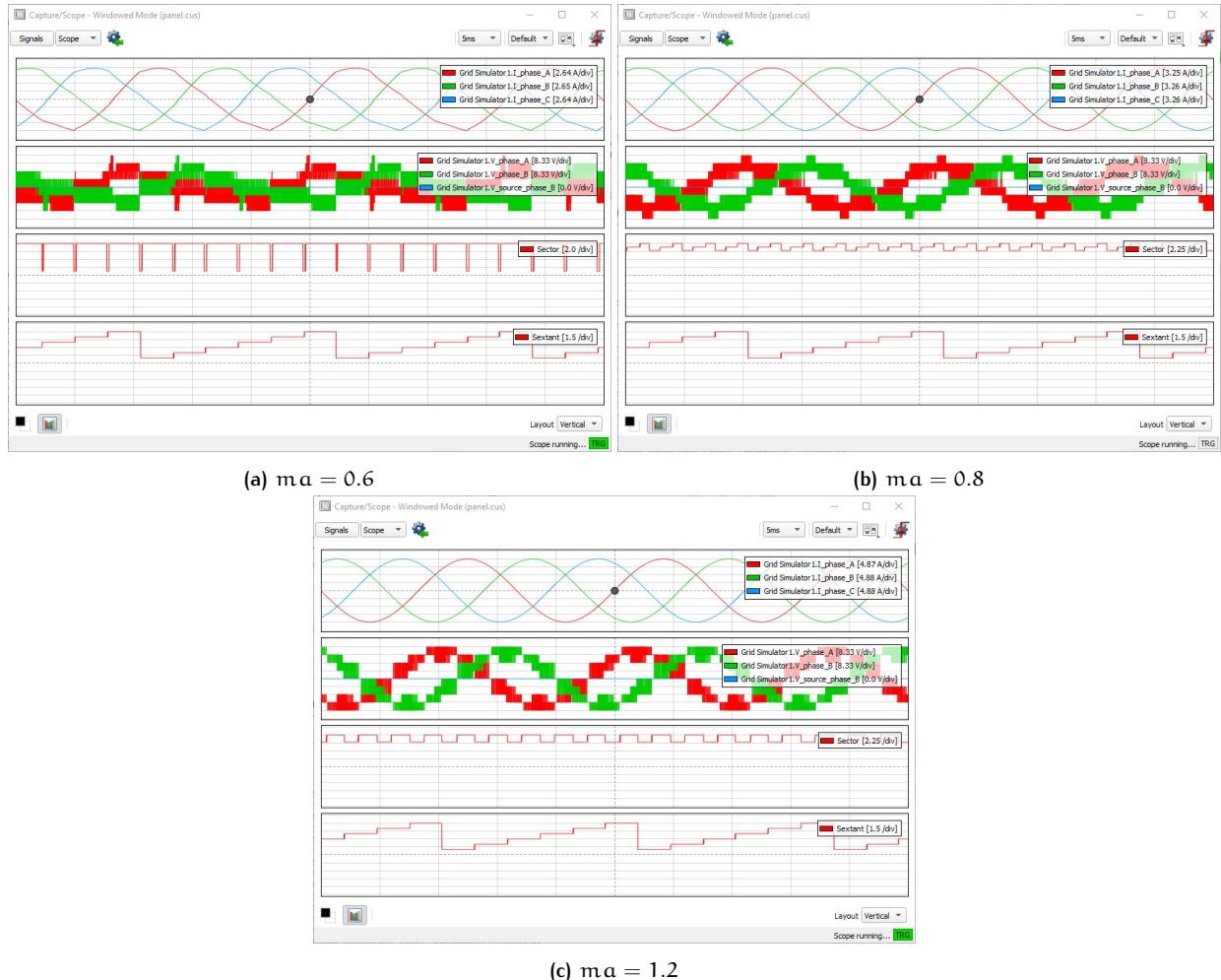
**Figura 10:** Trecho de código com a atribuição dos valores de *dwell times*, com projeto que mostra a contribuição de cada um dos vetores para síntese da referência, como foi mostrado na Figura 6.

### 3 RESULTADOS

Por fim, os resultados mostram a implementação do conversor trifásico ao lado da rotina em C como o algoritmo de modulação (Figura 11) e alguns resultados variando o índice modulação (Figura 12).



**Figura 11:** O conversor NPC trifásico com as tensões de entrada balanceadas, a rotina C com a SVM e uma carga RL para conexão das fases ( $R = 1\Omega$  e  $L = 3mH$ ).



**Figura 12:** Resultados obtidos para índices de modulação diferentes.