

Simulazione dei Sistemi dinamici con Matlab-Simulink

**Soluzione della prova del
20/1/2021**

Ing. Alessandro Pisano
apisano@unica.it

Prova finale di “Simulazione dei Sistemi Dinamici con Matlab Simulink” – 20/1/2021

La dinamica di un motore passo-passo è rappresentata, sotto determinate approssimazioni, dal seguente sistema di equazioni differenziali

$$L \frac{di_a(t)}{dt} = v_a(t) - R i_a(t) + K_m \omega(t) \sin(N_r \theta(t))$$

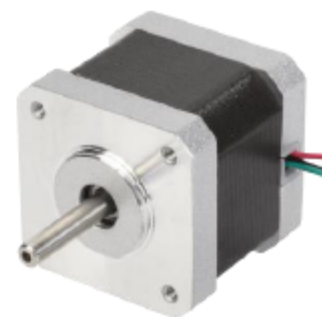
$$L \frac{di_b(t)}{dt} = v_b(t) - R i_b(t) - K_m \omega(t) \cos(N_r \theta(t))$$

$$J \frac{d\omega(t)}{dt} = -B \omega(t) - K_m i_a(t) \sin(N_r \theta(t)) + K_m i_b(t) \cos(N_r \theta(t))$$

$$\frac{d\theta(t)}{dt} = \omega(t)$$

in cui $v_a(t)$ e $v_b(t)$ sono le tensioni applicate, $i_a(t)$ e $i_b(t)$ sono le correnti di fase, $\omega(t)$ e $\theta(t)$ sono rispettivamente la velocità angolare e la posizione angolare dell'albero. I parametri del motore sono i seguenti:

$L = 5 \text{ mH}$	$R = 0.7 \Omega$	$J = 3 \cdot 10^{-2} \text{ kg m}^2$
$K_m = 0.5 \text{ N M / A}$	$N_r = 50$	$B = 0.2 \text{ N M s / rad}$



e le tensioni applicate hanno la seguente espressione

$$v_a(t) = 20 + 10 \cos(15t) \sin(5t)$$

$$v_b(t) = 15 + 5 \sin(10t)$$

Realizzare un modello Simulink del sistema dinamico considerando le condizioni iniziali $i_a(0) = 0.1 A$, $i_b(0) = -0.1 A$, $\omega(0) = 2 \text{ rad/s}$ $\theta(t) = \pi \text{ rad}$.

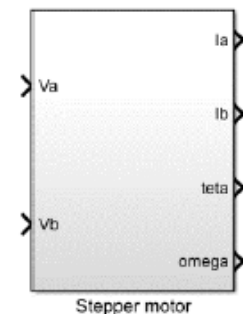
Il modello dovrà contenere un Subsystem che riceve in ingresso i segnali $v_a(t)$ e $v_b(t)$ e produca in uscita i segnali $i_a(t)$, $i_b(t)$, $\omega(t)$ e $\theta(t)$, come mostrato nella figura a lato.

Simulare il sistema per 20 secondi con passo di campionamento $T_s = 0.001 \text{ s}$ e creare un grafico che mostri sovrapposte le due correnti di fase.

Realizzare uno script che avvii in automatico il modello Simulink e crei il grafico richiesto.

Scrivere quindi una **function** che acquisisca in ingresso le due correnti di fase e la posizione angolare e costruisca due grafici che mostrino le evoluzioni temporali dei segnali $i_d(t)$ e $i_q(t)$ (denominati componenti diretta e in quadratura delle correnti) definiti come segue

$$\begin{bmatrix} i_d(t) \\ i_q(t) \end{bmatrix} = \begin{bmatrix} \cos(N_r \theta(t)) & \sin(N_r \theta(t)) \\ -\sin(N_r \theta(t)) & \cos(N_r \theta(t)) \end{bmatrix} \begin{bmatrix} i_a(t) \\ i_b(t) \end{bmatrix}$$



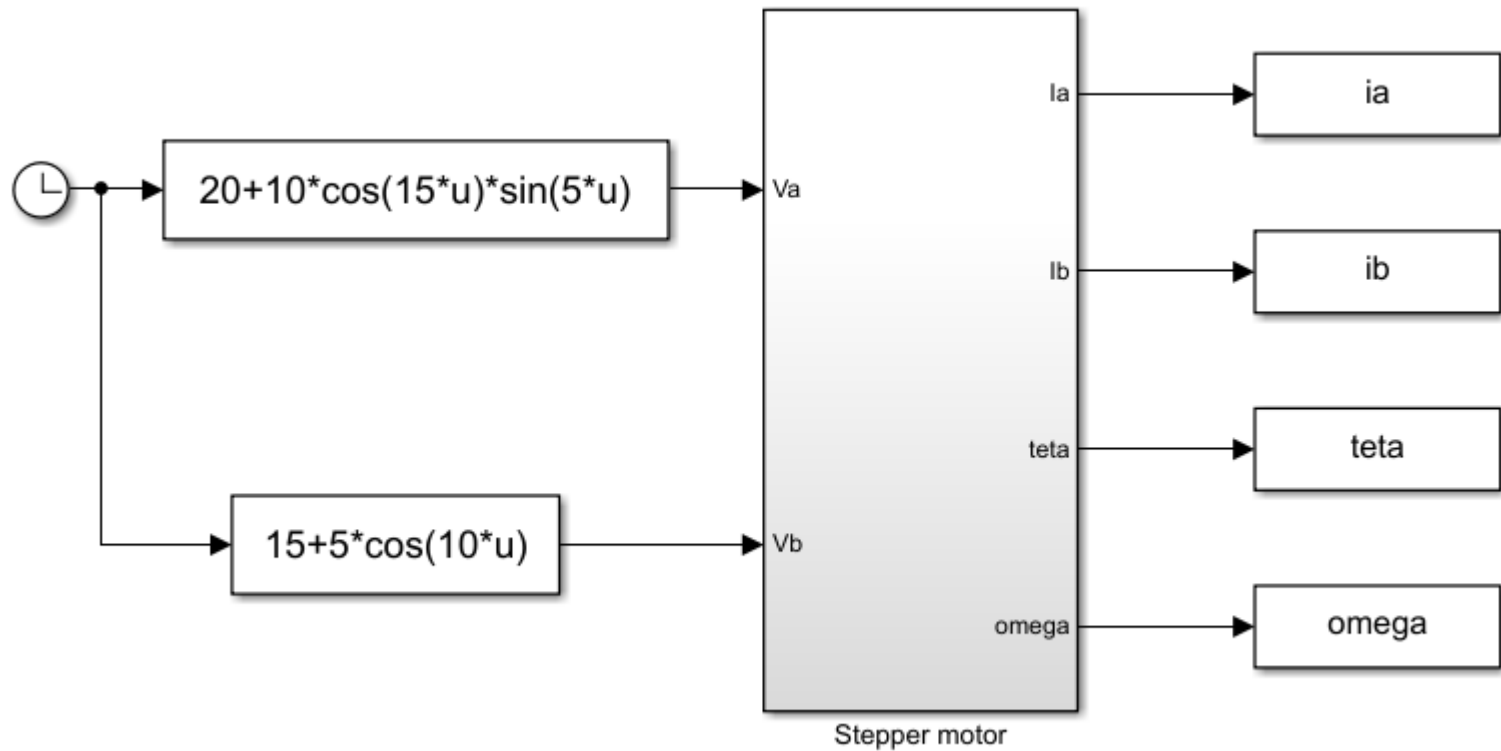
Presentiamo diverse possibili soluzioni sia in merito alla struttura del modello Simulink che della Function richiesta.

Soluzione 1

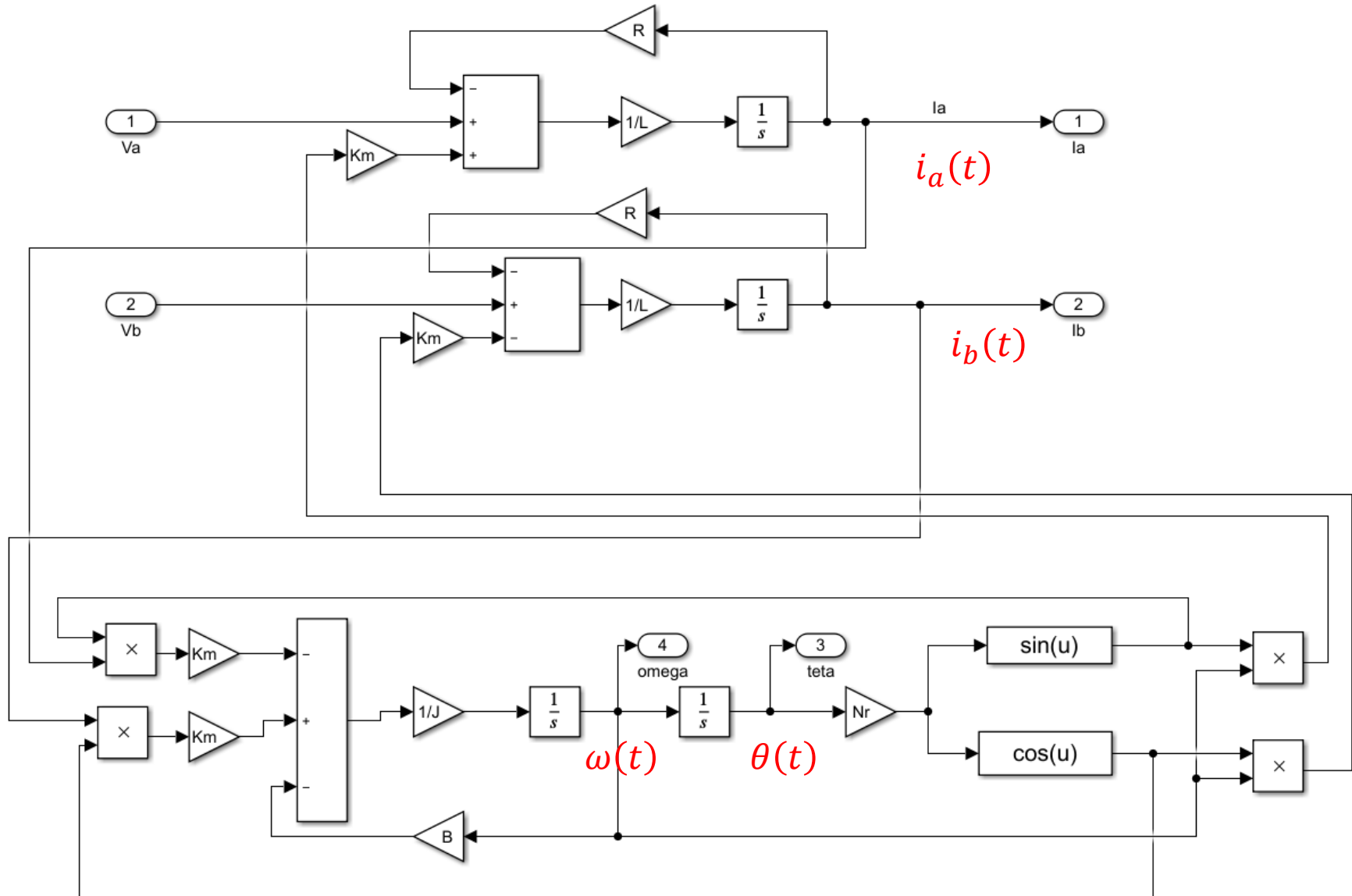
Files: `steppermotor_model01.slx`
`steppermotor_Script01.m`

Lo Script contiene due diverse implementazioni alternative della function, chiamate «Park1» e «Park2» (in quanto le correnti $i_d(t)$ e $i_q(t)$ si ottengono applicando ad $i_a(t)$ e $i_b(t)$ la cosiddetta «trasformazione di Park»)

Modello Simulink



Contenuto del Subsystem «Stepper Motor»



```
clear all  
close all  
clc
```

Prima parte dello script

```
L=0.005;  
R=0.7;  
J=0.03;  
Km=0.5;  
Nr=50;  
B=0.2;
```

Definizione parametri

```
ia0=0.1;  
ib0=-0.1;  
teta0=pi;  
omega0=2;
```

Definizione condizioni iniziali

```
sim('steppermotor_model01')
```

Avvio del modello

```
figure  
plot(ia.time,ia.data,ib.time,ib.data),grid  
legend('ia','ib')
```

Grafico delle correnti di fase

Seconda parte dello script

```
%% Utilizzo della function "Park1"
```

```
[id iq]=Park1(ia,ib,teta);
```

```
t=ia.Time;
```

```
figure
```

```
plot(t,id,t,iq),grid
```

```
legend('id','iq')
```

```
%% Definizione della function Park1
```

```
function [id iq]=Park1(ia,ib,teta)
```

```
Nr=50;
```

```
id=cos(Nr*teta.data).*ia.data+sin(Nr*teta.data).*ib.data;
```

```
iq=-sin(Nr*teta.data).*ia.data+cos(Nr*teta.data).*ib.data;
```

```
end
```



```
%% Utilizzo della function "Park2"
```

```
[id iq]=Park2(ia,ib,teta);  
t=ia.Time;  
figure  
plot(t,id,t,iq),grid  
legend('id','iq')
```

```
%% Definizione della function Park2
```

```
function [id iq]=Park2(ia,ib,teta)
```

```
Nr=50;
```

```
Ia=ia.data;
```

```
Ib=ib.data;
```

```
Teta=teta.data;
```

```
t=ia.time;
```

```
for i=1:length(t)
```

```
Idqi=[cos(Nr*Teta(i)) sin(Nr*Teta(i)); -sin(Nr*Teta(i)) cos(Nr*Teta(i))]*[Ia(i);Ib(i)];
```

```
id(i)=Idqi(1);
```

```
iq(i)=Idqi(2);
```

```
end
```

```
end
```

Commenti

Il modello Simulink è realizzato in forma «basica», mediante l'utilizzo di blocchi elementari Simulink

Le funzioni Park1 e Park2 fanno la stessa cosa: generano i campioni delle correnti trasformate $i_d(t)$ e $i_q(t)$

La funzione **Park1** si basa sulle espressioni separate delle due correnti $i_d(t)$ e $i_q(t)$

$$i_d(t) = \cos(N_r \theta(t)) i_a(t) + \sin(N_r \theta(t)) i_b(t)$$

$$i_q(t) = -\sin(N_r \theta(t)) i_a(t) + \cos(N_r \theta(t)) i_b(t)$$

ed implementa tali espressioni mediante gli operatori di moltiplicazione elemento per elemento (il parametro N_r deve essere ridefinito internamente alla funzione in quanto le funzioni non possono in generale accedere alle variabili presenti nel workspace)

La funzione **Park2** sfrutta invece l'espressione matriciale della relazione

$$\begin{bmatrix} i_d(t) \\ i_q(t) \end{bmatrix} = \begin{bmatrix} \cos(N_r\theta(t)) & \sin(N_r\theta(t)) \\ -\sin(N_r\theta(t)) & \cos(N_r\theta(t)) \end{bmatrix} \begin{bmatrix} i_a(t) \\ i_b(t) \end{bmatrix}$$

Il ciclo for determina ad ogni iterazione la quantità

$$I_{dqi} \equiv \begin{bmatrix} i_d(iT_c) \\ i_q(iT_c) \end{bmatrix} = \begin{bmatrix} \cos(N_r\theta(iT_s)) & \sin(N_r\theta(iT_s)) \\ -\sin(N_r\theta(iT_s)) & \cos(N_r\theta(iT_s)) \end{bmatrix} \begin{bmatrix} i_a(iT_s) \\ i_b(iT_s) \end{bmatrix}$$

N.B. la i-esima componente del vettore \mathbf{T}_{θ} coincide con $\theta(iT_s)$, dove T_s è il passo di campionamento (pari a 0.001 s)

Le ultime due istruzioni interne al ciclo for estraggono dal vettore I_{dqi} la prima e la seconda componente, «scrivendo» tali valori come i-esime componenti dei vettori i_d e i_q che al termine della esecuzione del ciclo for vengono restituiti all'esterno della funzione.

Le funzioni Park1 e Park 2 possono indifferentemente essere definite mediante file dedicati con estensione .m oppure (come fatto nei files forniti) inserite al termine dello script.

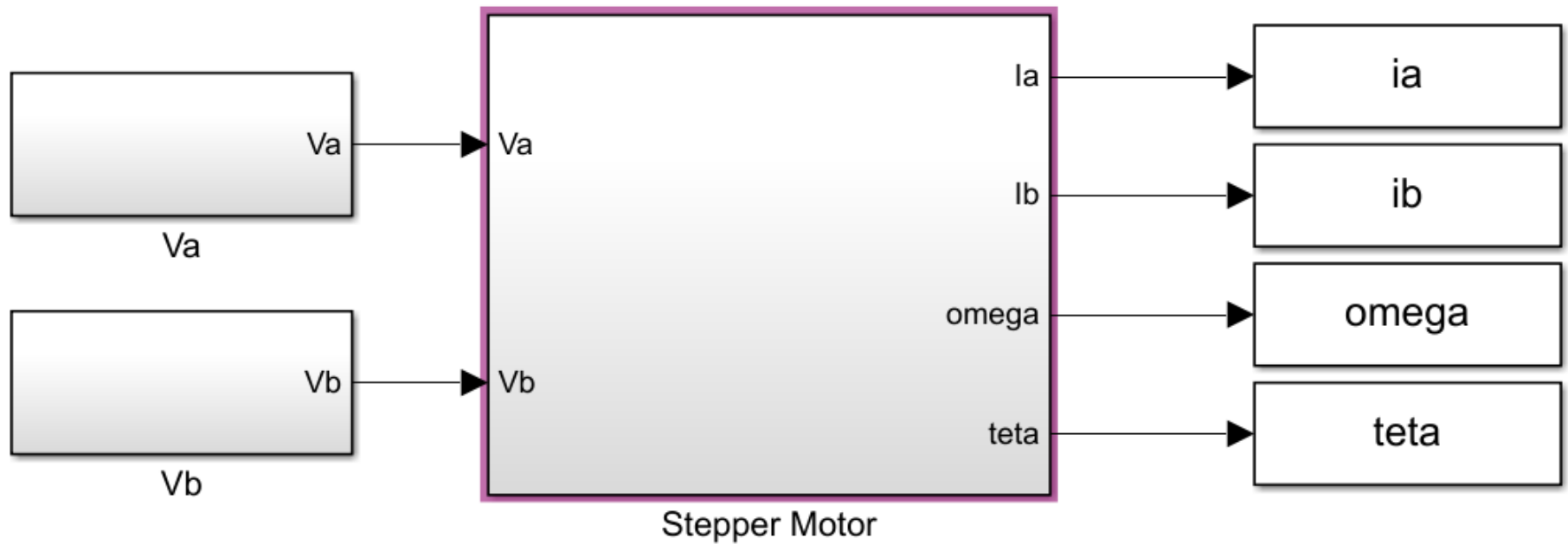
Nel secondo caso le funzioni possono ovviamente essere impiegate unicamente all'interno dello script che ne contiene la definizione.

Soluzione 2

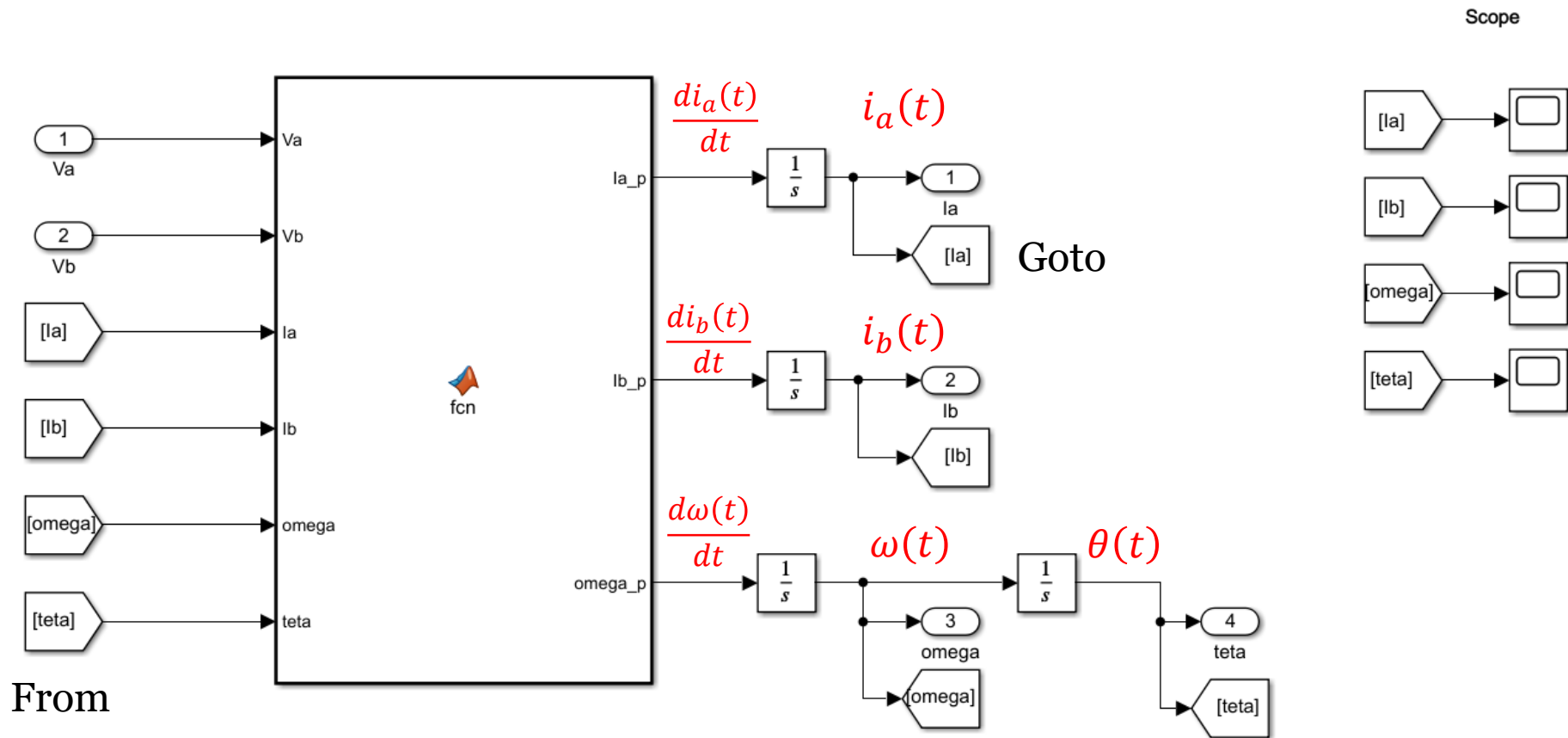
Files: `steppermotor_model02.slx`
`steppermotor_Script02.m`

La soluzione 2 differisce unicamente nella struttura del modello Simulink.

Modello Simulink



Contenuto del Subsystem «Stepper Motor»



Le espressioni delle derivate $\frac{di_a(t)}{dt}$, $\frac{di_b(t)}{dt}$, $\frac{d\omega(t)}{dt}$ sono implementate mediante un blocco Matlab Function. Per ridurre il numero di linee di collegamento, si fa anche uso di 4 coppie di blocchi Goto e From.

Matlab Function

```
function [Ia_p, Ib_p, omega_p] = fcn(Va, Vb, Ia, Ib, omega, teta)

L=0.005;
R=0.7;
J=0.03;
Km=0.5;
Nr=50;
B=0.2;

Ia_p=(Va-R*Ia+Km*omega*sin(Nr*teta))/L;
Ib_p=(Vb-R*Ib-Km*omega*cos(Nr*teta))/L;
omega_p=(-B*omega-Km*Ia*sin(Nr*teta)+Km*Ib*cos(Nr*teta))/J;
```

I parametri del motore devono essere definiti all'interno del blocco Matlab function. Non è più necessario definirli all'interno dello Script, che contiene unicamente le definizioni delle condizioni iniziali.

Prima parte dello script

```
clear all
close all
clc

ia0=0.1;
ib0=-0.1;
teta0=pi;
omega0=2;

Ts=0.001;

sim('steppermotor_model02')

figure
plot(ia.time,ia.data,ib.time,ib.data),grid
legend('ia','ib')
```

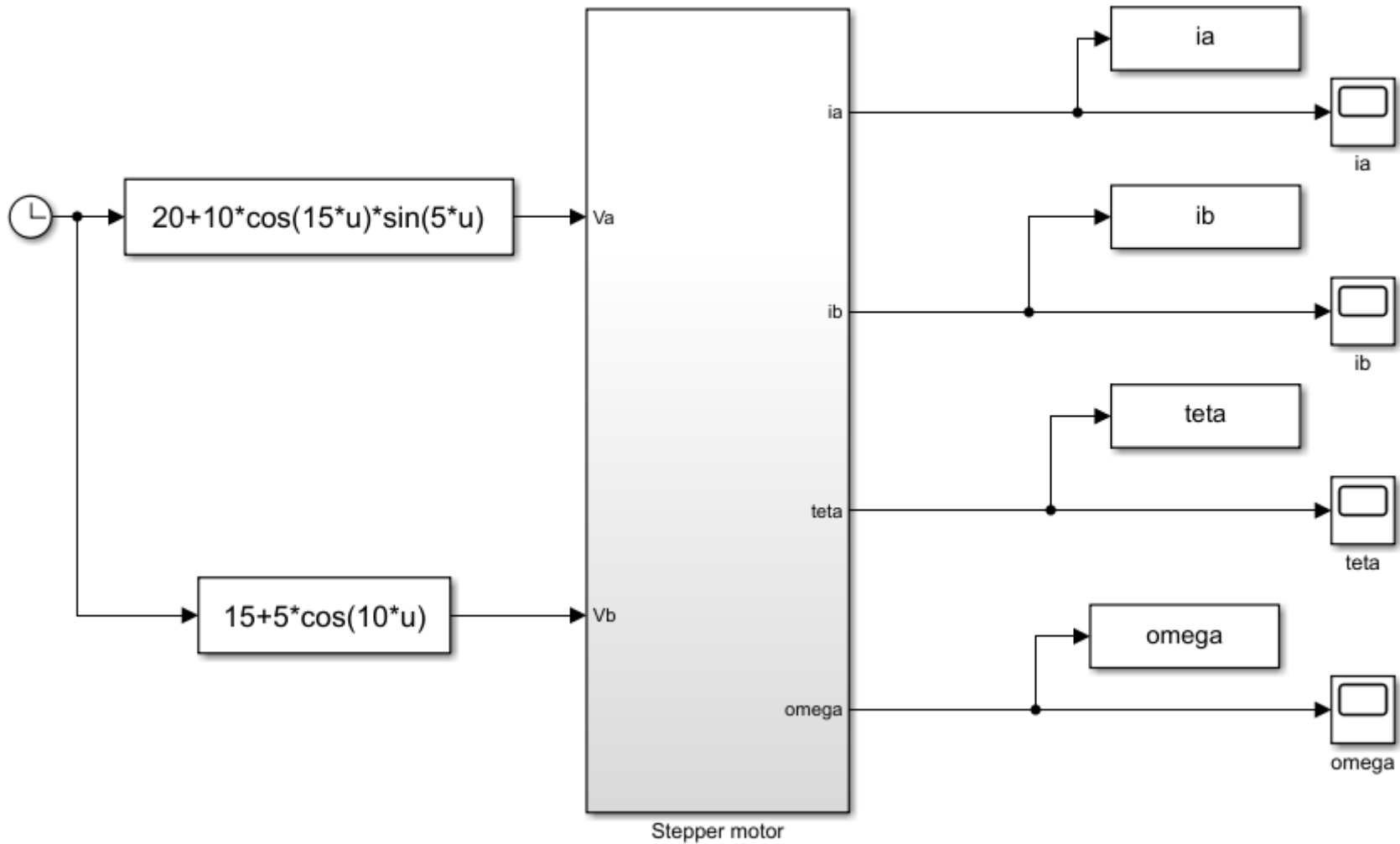
La prosecuzione dello script è analoga alla precedente Soluzione 1

Soluzione 3

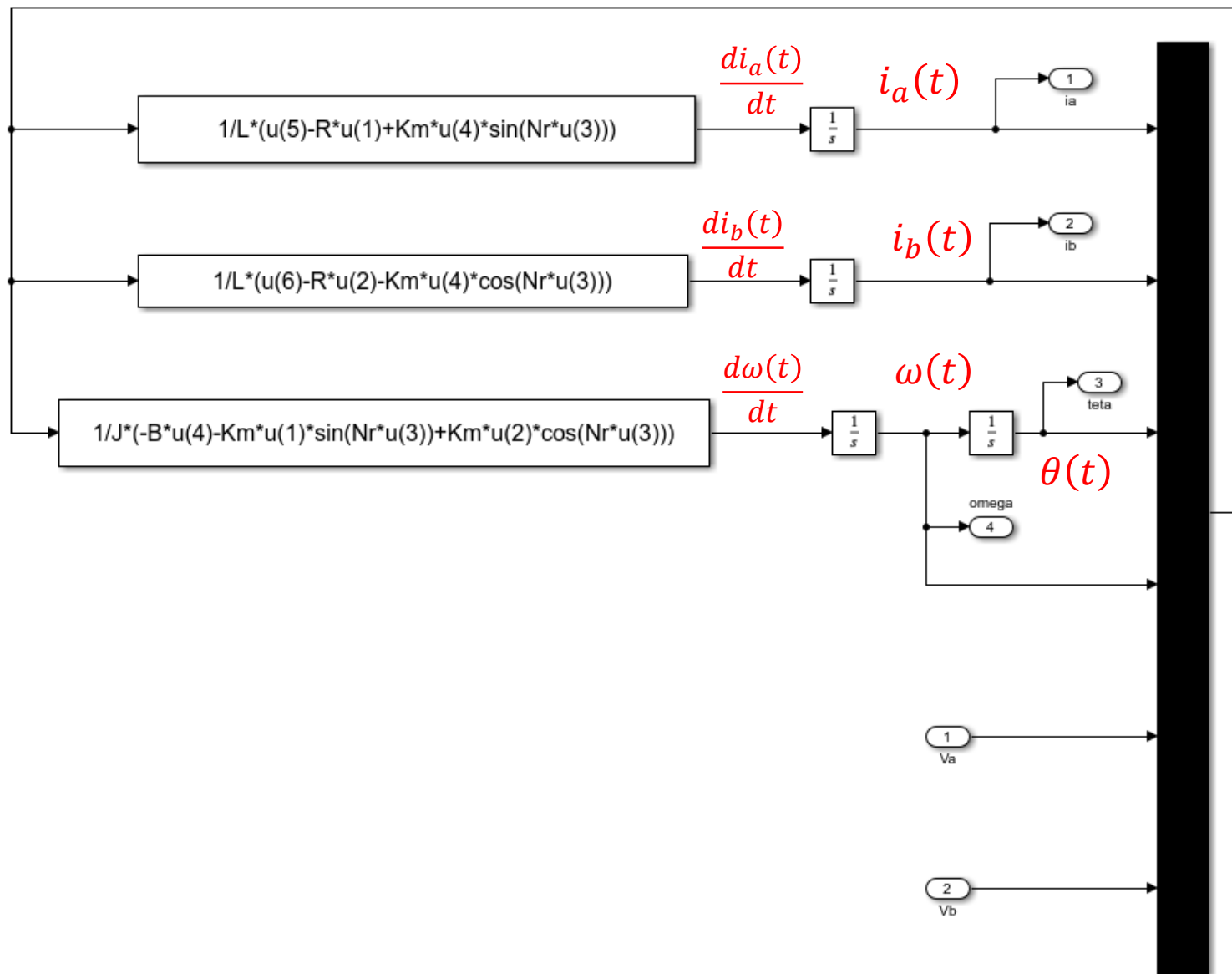
Files: `steppermotor_model03.slx`
`steppermotor_Script03.m`

La soluzione 3 differisce dalle due precedenti unicamente nella struttura del modello Simulink.

Modello Simulink



Contenuto del Subsystem «Stepper Motor»



Commenti

Il modello Simulink prevede l'impiego di vari blocchi Fcn che generano i segnali con i quali alimentare gli integratori

Tutti i blocchi Fcn sono alimentati dal medesimo segnale vettoriale che contiene, nell'ordine, le due correnti di fase, la posizione angolare, la velocità angolare, e le due tensioni di alimentazione.

La generazione dei segnali $v_a(t)$ e $v_b(t)$ è identica alla soluzione 1, così come lo Script di gestione.