پاسخ تمرین سری یکم مبانی سیستم‌های نهفته و بیدرنگ

۱. زبان StateChart و SDL از مکانیزم broadcast برای آپدیت کردن متغیرهای خود استفاده می‌کنند؛ در حالیکه petri net از این مکانیزم استفاده نمی‌کند.

StateChartها از مکانیزم broadcast استفاده می‌کنند؛ بدین گونه که مقادیر متغیرهای موجود برای تمامی قسمت‌های مدل StateChart قابل مشاهده است. به عبارتی دیگر، هرگونه سیگنالی که در یک clock cycle تولید شود، توسط تمامی بخش‌های آن مدل StateChart قابل مشاهده است.

SDL به گونه‌ای می‌تواند از این مکانیزم استفاده کند. یک سیگنال SDL برای در لحظه فقط می‌تواند برای یک نمونه پراسس ارسال شود. برای اینکه بتوان قابلیت broadcasting را به وجود آورد، کاربر باید یک package حاوی تعدادی تابع general-purpose را include کند تا بتواند این قابلیت را فراهم کند. بدین ترتیب، یک سری متغیرهای قابل broadcast شدن خواهیم داشت که داده‌های shared شده را بین nodeها یا اجراکننده‌ها ارسال می‌کند.

اما petri netها از این مکانیزم استفاده نمی‌کنند. در عوض، از یک روش دیگر استفاده می‌کنند که در آن یک event در شبکه می‌تواند وضعیت استیت‌های سیستم را با اضافه یا حذف کردن یک سری token از بافرها، استیت‌ها و ... تغییر دهد. این تغییرات به صورت local است و تنها استیت‌هایی را می‌توان تغییر داد که به آن event مشخص متصل باشند و در نتیجه از مکانیزم broadcast پشتیبانی نمی‌کند.

۲. a) از دو طرف معادله انتگرال می‌گیریم:

$$\int_0^t I\frac{d}{d\tau}\omega(\tau)d\tau = \int_0^t \big(k_T i(\tau) - x(\tau)\big)d\tau \rightarrow\ I\big(\omega(t)-\omega(0)\big)$$
$$= \int_0^t \big(k_T i(\tau) - x(\tau)\big)d\tau$$

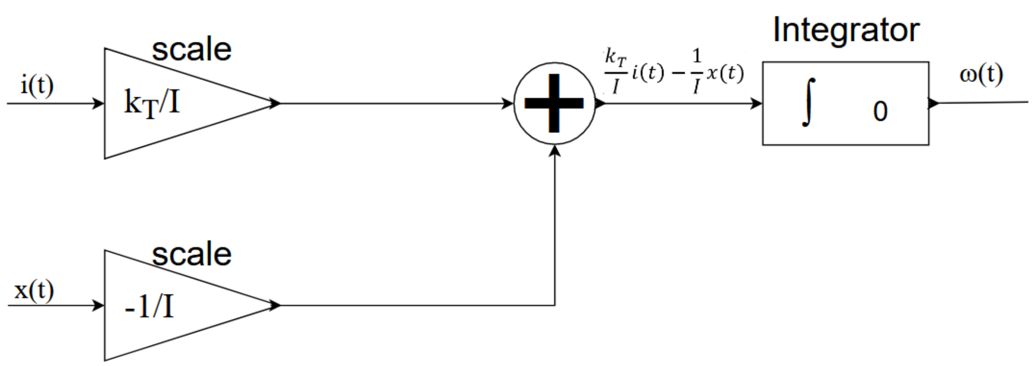
همانطور که در صورت سوال ذکر شده، موتور در لحظه *initial* در حالت استراحت است پس *ω(0)* برابر صفر است و در نتیجه عبارت بالا به صورت زیر تبدیل می‌شود:

$$I\omega(t) = \int_0^t (k_T i(\tau) - x(\tau))d\tau$$

با توجه به اینکه در بخش بعد *ω(t)* به عنوان خروجی actor model ما است، پس عبارت بالا را به صورت زیر می‌نویسیم:

$$\omega(t) = \int_0^t \left(\frac{k_T}{I}i(\tau) - \frac{1}{I}x(\tau)\right)d\tau$$

b) همانطور که در انتهای بخش قبل ذکر شد، از عبارت نهایی برای ساخت actor model استفاده می‌کنیم. ابتدا از دو *scalar* برای تولید $\frac{k_T}{I}i(t)$ و $-\frac{1}{I}x(t)$ استفاده می‌کنیم. سپس این دو مقدار را با کمک یک جمع‌کننده این دو مقدار را با همدیگر جمع می‌کنیم. در نهایت، از مقدار به دست آمده با کمک کامپوننت انتگرال‌گیر، انتگرال می‌گیریم تا مقدار *ω(t)* به دست بیاید. در نتیجه شماتیک *actor model* ما به صورت زیر خواهد شد:



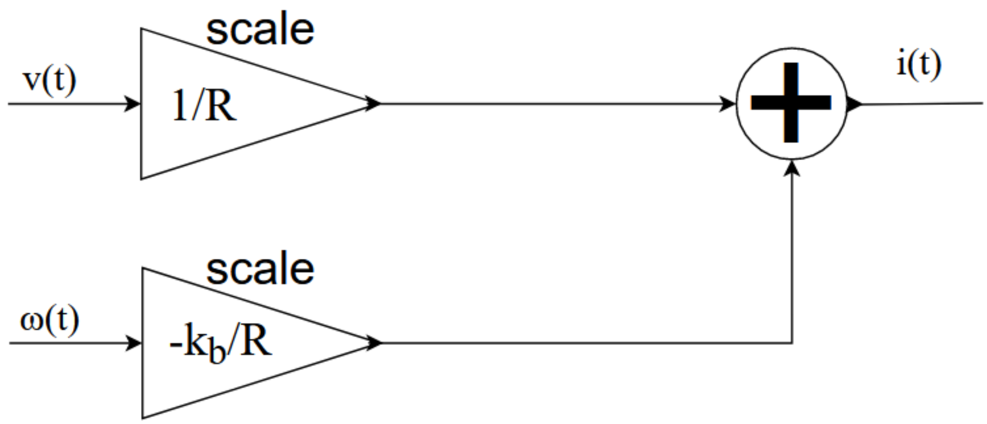
شکل ۱: شماتیک actor model رابطه داده شده

c) این *actor model* نیز تا حدودی مشابه *actor model* قبلی است؛ با این تفاوت که به جای مقدار جریان، مقدار ولتاژ را داریم و باید با استفاده از رابطه داده شده در این بخش، سیگنال *i(t)* را نیز بسازیم. پس انگار یک *sub model* به مدل قبلی اضافه می‌شود که خروجی آن به *model* ساخته شده در بخش قبل وارد می‌شود.

رابطه داده شده در این بخش را به صورت زیر بازنویسی می‌کنیم:

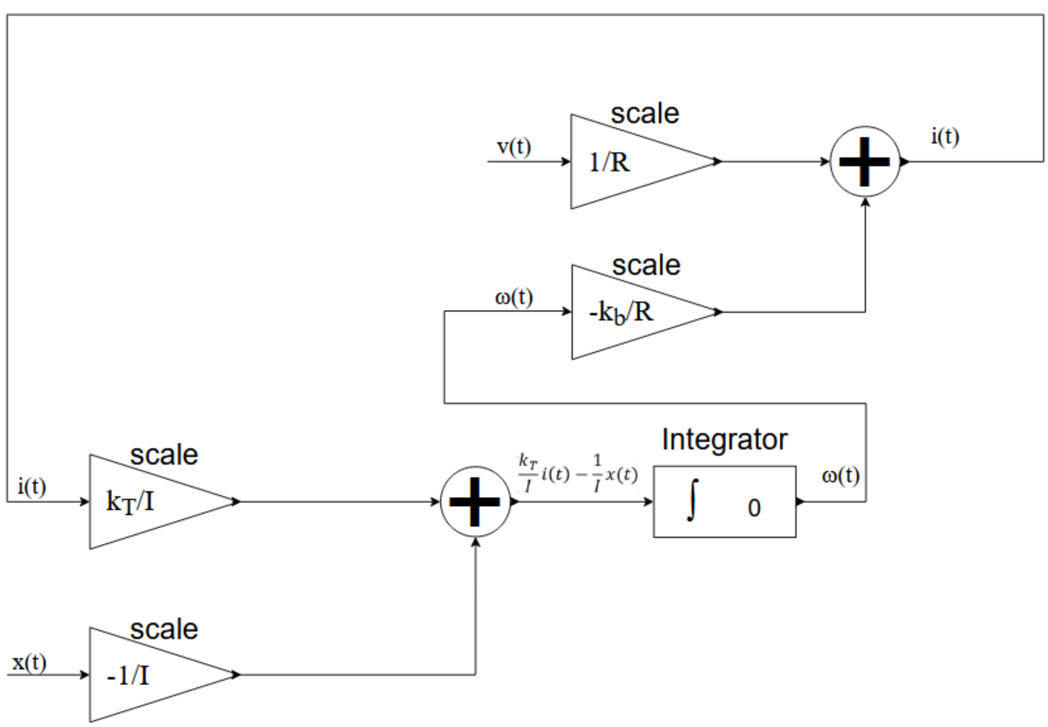
$$v(t) = Ri(t) + k_b\omega(t) \rightarrow i(t) = \frac{1}{R}v(t) - \frac{k_b}{R}\omega(t)$$

حالا این بخش از مدل به صورت زیر خواهد شد:



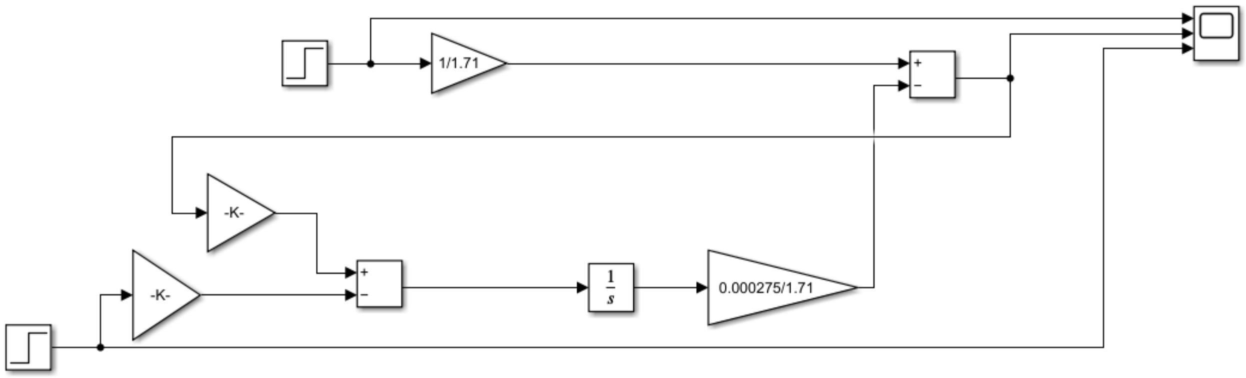
شکل ۲: شماتیک رابطه بین $\omega(t)$ و $v(t)$ و $i(t)$

اما می‌دانیم مقدار $\omega(t)$ از بخش قبلی مدل به دست آمده است. پس انگار مقدار $\omega(t)$ به عنوان ورودی بخش دوم زیرمدل استفاده می‌شود و در عوض، خروجی این بخش که $i(t)$ است به ورودی زیر مدل بخش اول وصل می‌شود. در نتیجه یک مدل با فیدبک خواهیم داشت که به صورت زیر خواهد بود:



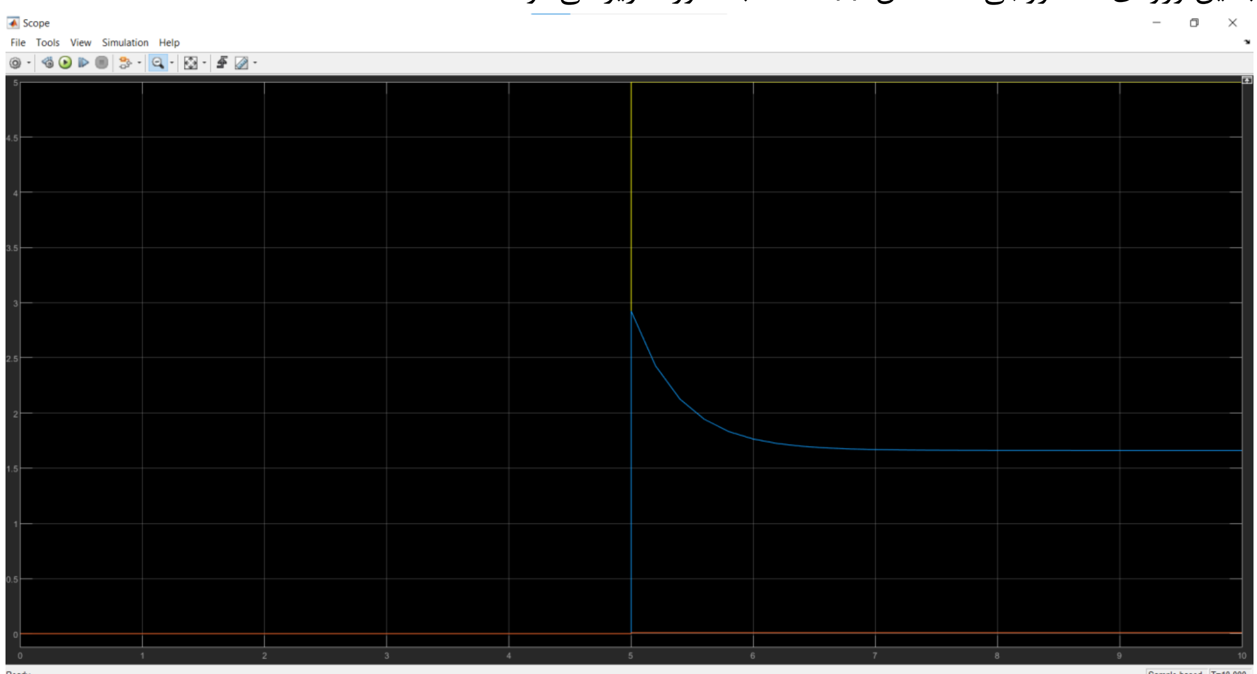
شکل ۳: شماتیک actor model جدید با آپدیت $v(t)$ به جای $i(t)$ به عنوان ورودی

کامپوننت‌ها را مطابق مدل به دست آمده از قسمت قبل قرار می‌دهیم. مدل ساخته شده به صورت زیر خواهد بود:



ضرایب (gainها) را مطابق صورت سوال مقداردهی کرده و به v(t) یک سیگنال پله با مقدار ۵ ولت را اساین می‌کنیم. مقدار x(t) را نیز یک سیگنال پله در نظر می‌گیریم، که در ثانیه‌ی ۵ مقدارش ۰,۰۱ می‌شود (مطابق صورت سوال).

با این ورودی‌ها، خروجی که همان i(t) است به صورت زیر می‌شود:



باید در نظر داشت که مقدار i در ابتدا همان ۰ در نظر گرفته می‌شود، ولی در لحظاتی که مقدار v و x تغییر می‌کند، مقدار i نیز تغییر کرده و از جایی که i ورودی نیز هست، مقدار لحظه‌ی قبلش روی مقدر لحظه‌ای فعلی‌اش تاثیر می‌گذارد.