# ۳ مبانی ومفاهیم مورد نیاز برای کار با سیمولینک

نرم‌افزار سیمولینک ابزاری برای مدل‌سازی، شبیه‌سازی و تحلیل سیستم‌های دینامیکی می‌باشد. برای انجام این کار لازم است که سه مرحله اصلی زیر طی شود:

- تعریف سیستم دینامیکی.
- مدل‌سازی.
- شبیه‌سازی.

بنابراین لازم است که در قدم اول نسبت به تعریف مفاهیمی چون سیستم دینامیکی، مدل‌سازی و شبیه‌سازی اقدام شود. بخش‌های این فصل به تشریح این مفاهیم، اختصاص دارد. البته باید اشاره کرد که جزئیات مرتبط با آن‌ها و نحوه‌ی کاربرد این مفاهیم، در فصل آتی همراه با مثال‌های لازم بیشتر توضیح داده خواهد شد.

## ۱.۳ سیستم دینامیکی

سیستمی که خروجی آن وابسته به زمان است و با آن تغییر می‌کند، سیستم دینامیکی نامیده می‌شود. سیستم‌های دینامیکی در سه گروه دسته‌بندی می‌شود:

- سیستم پیوسته.
- سیستم گسسته.
- سیستم هیبرید.

### ۱.۱.۳ سیستم پیوسته

سیستم پیوسته سیستمی است که می‌توان آن را به‌کمک معادلات دیفرانسیلی نمایش داد. سیستم جرم و فنر به معادله (۱.۳)، یک نمونه سیستم دینامیکی پیوسته می‌باشد.

$$m\ddot{x}(t)+c\dot{x}(t)+kx(t)=f(t) \qquad (۱.۳)$$

### ۲.۱.۳ سیستم گسسته

سیستم گسسته، سیستمی است که می‌توان آن را به‌کمک معادلات تفاضلی نمایش داد. این امکان وجود دارد که با گسسته‌سازی معادلات دیفرانسیلی حاکم بر سیستم‌های پیوسته، آن‌ها را گسسته نمود و به‌صورت معادلات تفاضلی نمایش داد. در برخی موارد، معادلات حاکم بر یک سیستم دینامیکی، اساسا به‌صورت تفاضلی استخراج می‌گردد. این مساله در دینامیک سیالات بسیار متداول می‌باشد. معادلات فضای حالت به‌صورت گسسته خطی برابرند با:

$$x(k+1)=Ax(k)+Bu(k) \qquad (۲.۳)$$

$$y(k)=Cx(k)+Du(k) \qquad (۳.۳)$$

### ۳.۱.۳ سیستم هیبرید

سیستمی که شامل اجزای پیوسته و گسسته باشد، هیبرید نامیده می‌شود. فرآیند پیوسته‌ای که توسط یک کامپیوتر دیجیتال کنترل شود، نمونه‌ای از یک سیستم هیبرید می‌باشد. امروزه استفاده از سیستم‌های هیبرید بسیار متداول می‌باشد.

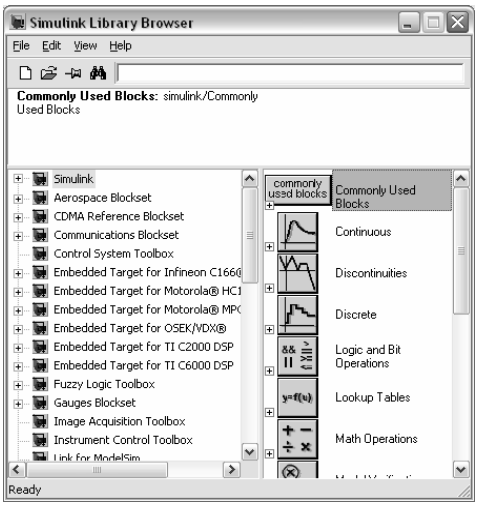
## ۲.۳ مدل‌سازی

مدل‌سازی به‌مفهوم ارائه روابط ریاضی حاکم بر یک سیستم می‌باشد. برای تشریح این روابط، می‌توان از رویکردی که معمولا به‌عنوان کدنویسی یا برنامه‌نویسی شناخته می‌شود، استفاده کرد. اما ساختار مدل‌سازی (برنامه‌نویسی) در سیمولینک به‌صورت خاصی می‌باشد که نمودار بلوکی نامیده می‌شود.

یک نمودار بلوکی نمایش گرافیکی ارتباط ریاضی بین ورودی‌ها، حالات و خروجی‌های یک سیستم می‌باشد. بنابراین لازم است در قدم اول تعابیری چون بلوک، حالت، زیرسیستم و ... که مرتبط با مرحله مدل‌سازی می‌باشد، تشریح گردد.

### ۱.۲.۳ بلوک

هر یک از اجزای داخل کتابخانه‌های سیمولینک یک بلوک می‌باشد. بلوک‌ها با اهداف مختلف و به‌صورت یک شئ در دسترس کاربر قرار می‌گیرد. از بلوک‌های پرکاربرد سیمولینک می‌توان به انتگرال‌گیر (Integrator)، بهره (Gain) و جمع (Sum) اشاره کرد.



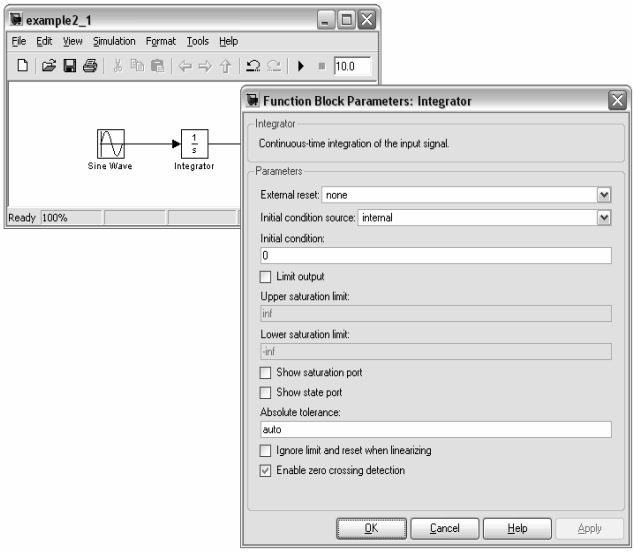

**شکل ۱.۳ مجموعه بلوک‌های متلب.**

کلیه بلوک‌های سیمولینک را می‌توان در Simulink Library Browser مشاهده کنید. مجموعه بلوک‌های سیمولینک –که هر یک متعلق به کارکرد خاصی است– در سمت چپ و بلوک‌های هر مجموعه در سمت راست شکل نشان داده شده‌است (با کلیک بر روی + می‌توانید بلوک‌های هر مجموعه را ببینید). در جلدهای بعدی این کتاب، هر کتابخانه سیمولینک به‌صورت جداگانه تشریح می‌گردد. اما در این جلد به تشریح کلیات کار با سیمولینک و مجموعه بلوک‌های عمومی سیمولینک، که در شکل ۱. ۳ نشان داده‌است، اکتفا می‌گردد.

### ۲.۲.۳ حالت

برخی از بلوک‌ها دارای کمیت‌های داخلی هستند. این کمیت‌ها می‌تواند ثابت یا متغیر با زمان باشد. همچنین تعدادی از این بلوک‌ها، دارای کمیت متغیر با زمانی می‌باشند که حین اجرای شبیه‌سازی تغییر می‌کند. به این کمیت حالت گفته می‌شود. مثلا بلوک انتگرال‌گیر دارای حالت است؛ چراکه دارای کمیتی است که مقدار کمیت ورودی، در هر قدم انتگرال‌گیری به آن افزوده می‌شود. اما بلوک بهره دارای حالت نیست؛ چراکه همواره مقدار ثابتی را در ورودی ضرب و در خروجی نشان می‌دهد.

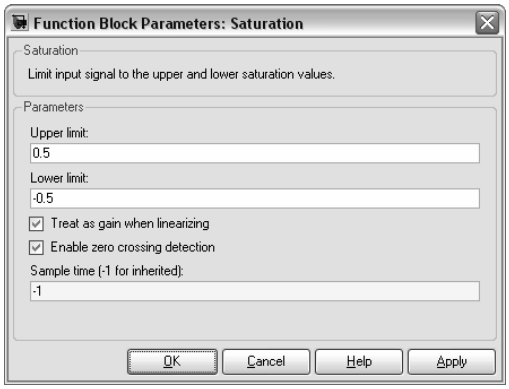
چنان‌چه بر روی هر بلوک دوبار کلیک کنید، می‌توانید کمیت‌های داخلی آن را ببینید. به‌عنوان مثال بر روی بلوک Integrator دوبار کلیک کنید:




**شکل ۲.۳ کمیت‌های داخلی بلوک انتگرال‌گیر.**

### ۳.۲.۳ زمان نمونه‌برداری

هر یک از بلوک‌های داخل کتابخانه گسسته، دارای کمیتی به‌نام زمان نمونه‌برداری (Sample time) می‌باشد. این بلوک‌ها در بازه‌های زمانی برابر با زمان نمونه‌برداری، کمیت‌های خروجی را ثابت درنظر می‌گیرد. البته می‌توان به گونه‌ای این زمان را تنظیم نمود که نرخ نمونه‌برداری مشابه بلوک‌ها قبل و یا بعد از شبیه‌سازی باشد؛ این حالت پیش‌فرض بیشتر بلوک‌ها است و مقدار آن نیز ۱- می‌باشد.




**شکل ۳.۳ زمان نمونه‌برداری.**

### ۴.۲.۳ زیرسیستم‌ها

در محیط سیمولینک این قابلیت وجود دارد که چندین بلوک را به‌صورت یک مجموعه واحد دسته‌بندی کرد تا مدیریت آن‌ها آسان‌تر گردد. این مجموعه زیرسیستم نامیده می‌شود و می‌توان از آن به‌عنوان یک بلوک جدید استفاده کرد. تولید بلوک جدید از یک زیرسیستم نقاب‌گذاری (Mask) نامیده می‌شود.

### ۵.۲.۳ زیرسیستم شرطی

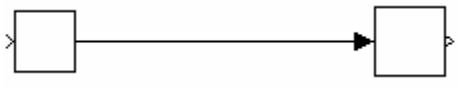
می‌توان زیرسیستم‌هایی تولید کرد که تنها با ارضای یک ورودی به‌عنوان فعال‌ساز یک عملیات خاص، اجرا گردد. از دیدگاه برنامه‌نویسی این زیرسیستم‌ها کار if، for، ... را انجام می‌دهند. به این زیرسیستم‌ها، زیرسیستم شرطی می‌گویند.

### ۶.۲.۳ ایجاد بلوک‌های دلخواه

می‌توان به‌صورت گرافیکی یا با استفاده از برنامه‌نویسی، بلوک‌های دلخواه موردنظر را تولید کرد. این امر به‌کمک استفاده از «m.» فایل‌ها، «S-Function» و «MEX» فایل‌ها قابل انجام است و با امکانات نقاب‌گذاری برای آن بلوک، پنجره رابط تولید کرد. این کار به‌خصوص هنگامی که مدل‌های آماده‌ای وجود دارد، بسیار مفید می‌باشد.

### ۷.۲.۳ سیگنال

سیمولینک برای نمایش خروجی‌ها و ورودی‌های یک بلوک از سیگنال استفاده می‌کند. می‌توان برای سیگنال‌ها خواص متنوعی مانند: نام، نوع داده (۸ بیتی، ۱۶ بیتی و ۳۲ بیتی)، نوع عدد (حقیقی یا مختلط) و بُعد، تعریف کرد.



شکل ۴.۳ سیگنال.

### ۸.۲.۳ نوع داده

نوع داده، خاصیتی است که به ارائه یک داده در حافظه مربوط می‌شود. سیمولینک انواع داده‌هایی را که توسط متلب قابل استفاده باشد، می‌شناسد. این داده‌ها عبارتند از: ۸ بیتی، ۱۶ بیتی، ۳۲ بیتی و

double. علاوه‌بر این، سیمولینک دارای دو نوع داده مخصوص به‌خود می‌باشد که داده‌های شئ‌گرا محسوب می‌شود. این دو عبارتند از:

- کمیت‌های سیمولینک.
- سیگنال‌های سیمولینک.

سیمولینک به شما توانایی می‌دهد که با استفاده از این دو نوع داده، بتوان داده‌های شئ‌گرای جدید و مورد نیاز را تولید کرد و از آن‌ها بهره‌برداری کرد.

## ۳.۳ شبیه‌سازی

پس از مدل‌سازی نوبت به اجرای شبیه‌سازی می‌رسد. به فرآیند محاسبه حالت‌ها و خروجی‌های یک سیستم دینامیکی، که به‌کمک یک سری معادله زمان وابسته مدل‌سازی شده‌است، در یک بازه‌ی زمانی خاص، شبیه‌سازی گفته می‌شود. به‌منظور انجام شبیه‌سازی، سیمولینک عملیاتی را بر روی مدل تولیدشده انجام می‌دهد. در این بخش برخی از مفاهیم مرتبط با این فرآیند تحت عناوین زیر تشریح می‌شود:

- تألیف مدل.
- برقراری اتصال.
- حلقه شبیه‌سازی.
- حل‌کننده.
- کشف عبور از صفر.
- حلقه جبری.

البته اکثر این مراحل به‌صورت خودکار توسط سیمولینک انجام می‌گیرد؛ اما آگاهی از این مراحل جهت ساختن مدل صحیح و یافتن خطاهای شبیه‌سازی بسیار مفید است و در مدل‌های پیچیده با ایجاد تغییراتی می‌توان یک شبیه‌سازی بهتر داشت (لذا اگر خواننده مبتدی هستید، می‌توانید از این بخش گذر کنید.). به‌هر حال عمل شبیه‌سازی تنها با طی مرحله زیر شروع می‌شود (Simulation\ Start).

### ۱.۳.۳ تألیف مدل (Compile)

هنگامی که یک مدل سیمولینک تولید شد، با صدور فرمان اجرا، یک سری عملیات بر روی مدل صورت می‌گیرد. اولین قدم این فرآیند Compile نمودن مدل است. این عملیات شامل مراحل زیر می‌باشد:

- ارزیابی بلوک‌ها.

- ارزیابی سیگنال‌ها.
- انتشار خواص سیگنالی.
- بهینه‌سازی.
- تسطیح برنامه.
- مرتب‌سازی.
- محاسبه بازده نمونه‌برداری.

- **ارزیابی بلوک‌ها:**

مقادیر کمیت‌های مربوط به بلوک‌های مدل مورد ارزیابی قرار می‌گیرد.

- **ارزیابی سیگنال‌ها:**

خواص سیگنال‌ها (نوع داده، بعد و ...) مورد ارزیابی قرار می‌گیرد. ارزیابی فقط به‌صورت صریح (یعنی با استفاده از خواص واردشده توسط کاربر) نخواهد بود؛ بلکه این ارزیابی همخوانی بین سیگنال و محل تولید و مصرف آن را نیز دربر می‌گیرد.

- **انتشار خواص سیگنالی:**

در این مرحله خواص مربوط به سیگنال‌های مولد (سیگنالی که توسط یک بلوک به‌عنوان ورودی تلقی می‌شود)، به ورودی بلوک‌های مصرف‌کننده آن سیگنال انتشار می‌یابد.

- **بهینه‌سازی:**

در این مرحله مدل تا حد امکان کوچک می‌شود و تا حد امکان خلاصه‌سازی صورت می‌گیرد.

- **تسطیح برنامه:**

در این مرحله زیرسیستم‌های موقتی ایجادشده توسط نرم‌افزار حذف می‌شوند، بلوک‌های داخل آن جایگزین می‌شود. در نتیجه این مرحله، کل برنامه یکپارچه می‌شود.

- **مرتب‌سازی:**

در این مرحله بلوک‌ها برحسب ترتیب مورد نیاز برای اجرا، مرتب می‌شود.

- **محاسبه بازه نمونه‌برداری:**

مرحله نهایی تعیین بازه نمونه‌برداری تمام بلوک‌هاست؛ به‌نحوی که نیاز به تعیین صریح این کمیت نباشد.

### ۲.۳.۳ برقراری اتصال

در این مرحله، موتور سیمولینک حافظه مورد نیاز برای اجرای مدل تولیدشده را تخصیص می‌دهد. این حافظه به سیگنال‌ها، حالت‌ها و کمیت‌های مربوط به زمان اجرا اختصاص می‌یابد. علاوه‌بر این، ساختمان‌هایی که برای ذخیره‌سازی اطلاعات هر بلوک در بازه زمانی شبیه‌سازی نیاز است، در این مرحله تخصیص داده می‌شود. برای بلوک‌های داخلی، اصلی‌ترین ساختمانی که به کمیت‌های مربوط به زمان اجرا اختصاص می‌یابد، **SimBlock** نامیده می‌شود. این ساختمان، حاوی اشاره‌گر (Pointer)های مربوط به حافظه‌های میانی (Buffers) ورودی و خروجی بلوک‌ها، حالت‌ها و بردارها می‌باشد.

### ۳.۳.۳ حلقه شبیه‌سازی

اکنون نوبت به حلقه شبیه‌سازی می‌رسد. در این مرحله، موتور سیمولینک حالت‌ها و خروجی‌ها را، در بازه زمانی شروع تا خاتمه شبیه‌سازی محاسبه می‌کند. محاسبات در قدم‌هایی، که اصطلاحا Step Time گفته می‌شود، انجام می‌پذیرد؛ طول این قدم‌ها Step Size نام دارد. طول قدم‌ها بسته به نوع حل‌کننده مورد استفاده برای محاسبه حالت‌های پیوسته، نمونه زمانی (Sample Time) برای سیستم‌های گسسته و شرایط عبور از صفر (Zero-Crossing Detection) یک حالت پیوسته (درصورت وجود) تعیین می‌گردد. این مرحله دارای دو قدم می‌باشد:

۱-مقداردهی اولیه (Initialization).

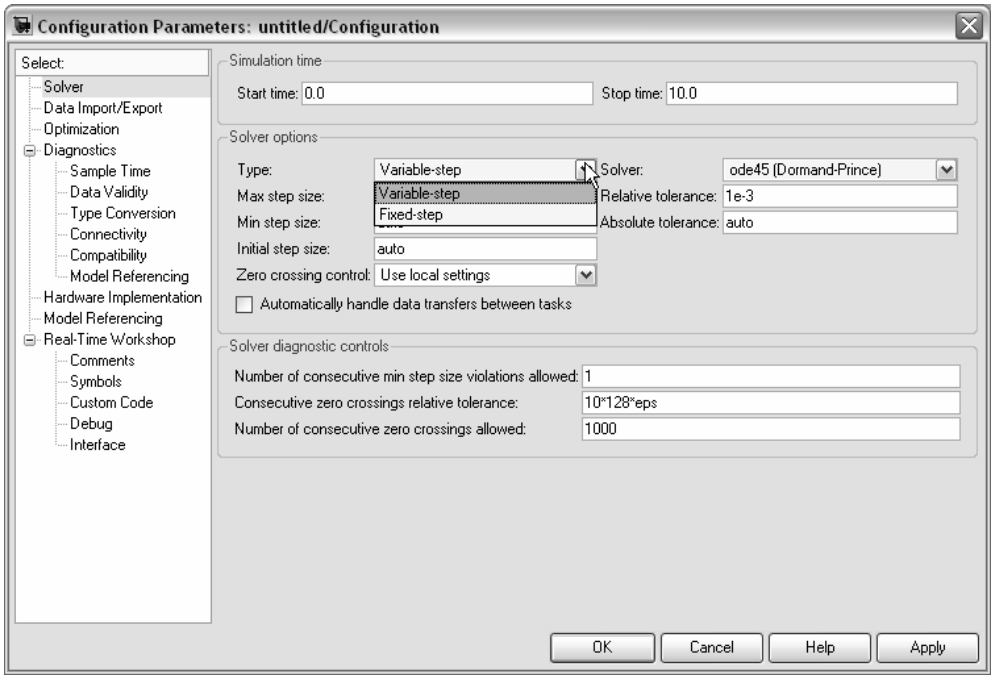
۲-تکرار (Iteration).

مرحله اول، تنها یک بار و در آغاز شبیه‌سازی صورت می‌گیرد. در این مرحله خروجی‌ها و حالت‌ها مقداردهی اولیه می‌شود. در مرحله تکرار، ورودی‌ها، حالت‌ها و خروجی‌ها در هر قدم زمانی به‌روز می‌شود. در پایان شبیه‌سازی سیمولینک مقادیر نهایی ورودی‌ها، خروجی‌ها و حالت‌ها را ارائه می‌نماید. توجه داشته باشید که سیمولینک دارای بلوک‌هایی است که به‌کمک آن‌ها (با افزودن آن‌ها به مدل) می‌توان تاریخچه زمانی تغییرات هر سیگنال دلخواهی را در آن‌ها ذخیره کرد و در پایان شبیه‌سازی مشاهده نمود.

### ۴.۳.۳ حل‌کننده

سیمولینک یک سیستم دینامیکی را، با محاسبه حالت‌های آن در قدم‌های زمانی مختلف شبیه‌سازی می‌نماید. به این فرآیند حل‌کردن مدل گفته می‌شود. برای محاسبه این حالت‌ها می‌توان از روش‌های مختلفی استفاده کرد و به‌طور قطع نمی‌توان گفت که یک روش خاص می‌تواند برای تمامی سیستم‌ها مناسب باشد. به‌همین دلیل سیمولینک دارای مجموعه‌ای از برنامه‌ها برای محاسبه حالت‌ها می‌باشد که اصطلاحا **حل‌کننده** گفته می‌شود. انتخاب نوع حل‌کننده از طریق پنجره Configuration Parameters که در منوی Simulation، واقع در

پنجره ویرایشگر محیط سیمولینک قرار دارد، امکان‌پذیر است. در این حالت می‌توان با استفاده از حل‌کننده‌های قدم ثابت یا قدم متغیر، و در شرایط پیوسته یا گسسته، اقدام به حل مدل نمود.




**شکل ۵.۳ تنظیمات حل‌کننده.**

با توجه به توضیحات بالا، می‌توان حل‌کننده‌های سیمولینک را در دو گروه اصلی به‌صورت زیر دسته‌بندی کرد:

- قدم ثابت (Fixed Step).
- قدم متغیر (Variable Step).

در نتیجه با توجه به پیوسته یا گسسته بودن سیستم، چهار حالت زیر ایجاد می‌گردد:

- حل‌کننده قدم ثابت برای سیستم‌های پیوسته (Fixed Step Continuous Solvers).
- حل‌کننده قدم متغیر برای سیستم‌های پیوسته (Variable Step Continuous Solvers).
- حل‌کننده قدم ثابت برای سیستم‌های گسسته (Fixed Step Discrete Solvers).
- حل‌کننده قدم متغیر برای سیستم‌های گسسته (Variable Step Continuous Solvers).

### ۵.۳.۳ کشف عبور از صفر

زمانی که یک برنامه توسط سیمولینک اجرا می‌شود، سیمولینک درباره گسستگی متغیرهای حالت در هر قدم زمانی سیستم بررسی می‌کند. تکنیکی که توسط سیمولنیک برای این کار استفاده می‌شود، **کشف عبور از صفر** یا **Zero Crossing Detection** نامیده می‌شود. هنگامی که سیمولینک یک ناپیوستگی را در بازه‌ی زمانی خاصی کشف کند، قدم‌های زمانی را حول آن قدم زمانی افزایش خواهد داد. برای درک بهتر اهمیت کشف عبور از صفر، به مثال زیر توجه کنید:

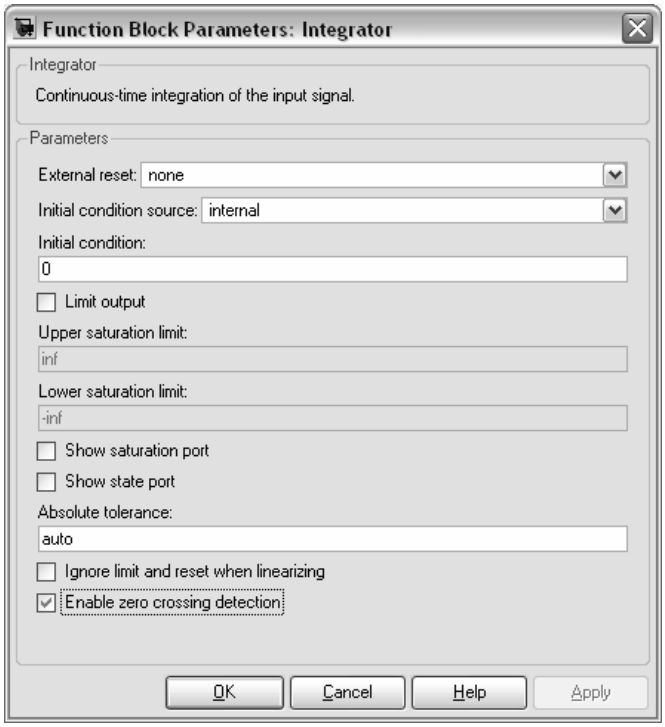
فرض کنید قصد دارید یک حرکت سقوط آزاد یک توپ لاستیکی را شبیه‌سازی نمایید. هنگامی که توپ با زمین برخورد کند، روند تغییرات موقعیت جسم در یک لحظه تغییر می‌نماید. چنان‌چه نتوان زمان این گسستگی را شناسایی کرد، علاوه‌بر این که در مورد لحظه برخورد توپ با زمین نمی‌توان اظهار نظر کرد، مطالعه پروفیل زمانی موقعیت نیز یک تغییر مسیر حرکت برای توپ در فضا بدون هیچ دلیل منطقی را نشان می‌دهد. در حالی‌که با استفاده از تکنیک یادشده، می‌توان به‌خوبی لحظه برخورد توپ با زمین را مشخص نمود.

برای استفاده از این تکنیک لازم است که از متغیرهای با گام متغیر استفاده شود؛ چرا که همان‌طور که تشریح شد، با ثابت‌کردن گام زمانی امکان دارد نقطه گسستگی بین دو سر یک قدم زمانی قرار گیرد، و اصطلاحا انتگرال‌گیر از روی آن جهش نماید. البته با کاهش بازه زمانی قدم‌ها می‌توان احتمال این رویداد را کم کرد، ولی هرگز به صفر نخواهد رسید. کوچک‌کردن بازه‌های زمانی منجر به افزایش زمان شبیه‌سازی می‌شود. در حالی که با استفاده از حل‌کننده‌های با قدم متغیر، بازه زمانی در اطراف نقاط گسست کوچک است و در نقاطی که تغییرات کند است، بزرگ می‌شود. این مساله باعث افزایش سرعت قابل توجه شبیه‌سازی، در عین کشف نقاط گسست می‌شود.

در مورد نحوه عملکرد و پیاده‌سازی این تکنیک مطالب گسترده و تخصصی زیاد قابل بیان است که می‌توان با مراجعه به Help نرم‌افزار از آن‌ها اطلاع کامل یافت. اما در این‌جا به‌صورت گذرا مطالبی بیان می‌شود:

به‌منظور پیاده‌سازی تکنیک یادشده، سیمولینک به‌همراه هر بلوک، متغیرهایی را که به بحث عبور از صفر مربوط می‌شود، ذخیره می‌نماید. هر یک از این متغیرها تابعی از مقدار متغیر حالت مربوط به آن بلوک می‌باشد که می‌تواند مقداری مثبت یا منفی را به‌خود اختصاص دهد. این توابع به‌نحوی تعریف می‌شود که هنگامی که پدیده گسستگی برای متغیر حالت پدید آمد، تابع مورد نظر، از یک مقدار مثبت یا منفی به صفر رسیده باشد و از آن عبور کرده باشد. در پایان هر قدم زمانی، سیمولینک تمامی متغیرهای مربوط به توابع عبور از صفر را به‌روز می‌نماید و در مورد تغییر علامت آن‌ها بررسی می‌کند. زمانی که متغیری تغییر علامت داده باشد یعنی با پدیده گسستگی ایجاد شده‌است. در این حالت سیمولینک با استفاده از میان‌یابی بین مقدار متغیر یادشده، زمان عبور از صفر آن را تخمین

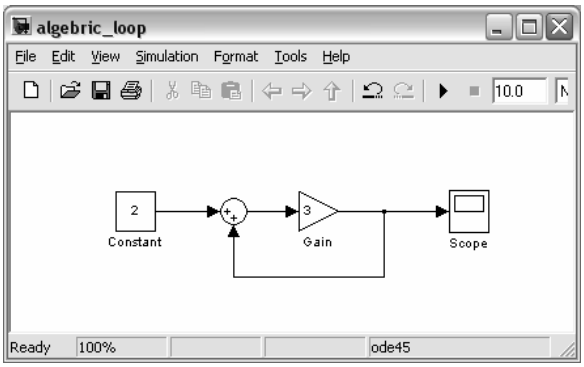
می‌زند. با مشخص‌شدن این زمان، سیمولینک درعین شناسایی نقطه گسست برای اجتناب از بروز مشکل از روی آن نقطه جهش می‌نماید ولی برای نقاط قبل و بعد از آن، قدم‌های زمانی کوچک‌تری درنظر گرفته می‌شود. در شکل ۳. ۶ توانایی کشف عبور از صفحه در تنظیمات بلوک انتگرال‌گیر با گزینه **Enable zero crossing detection** فعال شده‌است.




**شکل ۳. ۶ تشخیص کشف عبور از صفر.**

### ۳.۳.۶ حلقه جبری

در برخی بلوک‌های سیمولینک برای محاسبه خروجی، لازم است که حتما مقدار سیگنال ورودی آن‌ها مشخص باشد. این بلوک‌ها اصطلاحا **feedthrough** نامیده می‌شود. بلوک‌های جمع، بهره، ضرب و ... از این نوع می‌باشد. حال اگر در یک مدل سیمولینک خروجی این بلوک در محاسبه ورودی آن نقش داشته باشد آن مدل دارای حلقه جبری می‌باشد. در شکل ۳. ۷ یک نمونه ساده از یک مدل که دارای حلقه جبری می‌باشد، نمایش داده شده‌است.




**شکل ۷.۳ حلقه جبری.**

دو دلیل از دلائل عمده‌ای که باعث بروز حلقه‌ی جبری می‌شود عبارت است از:

۱-نیاز به مقدار اولیه برای برخی سیگنال‌ها.

۲-حل دستگاه معادلات.

مشکل اول، با استفاده از بلوک **IC**[1] واقع در کتابخانه بلوکی **Signal Attributes** قابل حل است. این بلوک مقدار اولیه اختصاص داده شده را در اولین گام شبیه‌سازی، به سیگنالی که به ورودی آن متصل شود، اختصاص می‌دهد. برای حل دستگاه معادلات نیز می‌توان از بلوک **Algebraic Constraint** واقع در کتابخانه بلوکی **Math Operation** استفاده کرد.

1 Initial Condition.