# حل معادله‌ی دیفرانسیل مرتبه‌ی اول

درس‌نامه‌ی کاربرد کامپیوتر در فیزیک

(دکتر اجتهادی، دانشگاه شریف، پاییز ۸۵)

همان‌طور که می‌دانیم در انتهای حل بیشتر مسائل فیزیکی به حل یک معادله‌ی دیفرانسیل می‌رسیم. ساده‌ترین معادله‌ای که در فیزیک می‌شناسید قانون دوم نیوتن است که با در نظر گرفتن این نکته که شتاب مشتق دوم مکان است، این معادله به یک معادله دیفرانسیل مرتبه‌ی دوم می‌شود. بنابراین در شبیه‌سازی بسیاری از مسائل فیزیک ناگزیر از حل یک معادله‌ی دیفرانسیل با کامپیوتر خواهیم بود. در این فصل به حل ساده‌ترین معادله دیفرانسیل یعنی معادله‌ی دیفرانسیل مرتبه‌ی اول می‌پردازیم. در بخش‌های بعد به مرور معادلات پیچیده‌تری را بررسی خواهیم کرد.

## ۱ الگوریتم اویلر[۱]

به عنوان یک مثال ساده پدیده‌ی سرد شدن یک لیوان قهوه داغ را در داخل اتاق بررسی می‌کنیم. می‌خواهیم منحنی دمای قهوه بر حسب زمان را به دست بیاوریم که این کار معادل با حل معادله‌ی دیفرانسیل سرد شدن قهوه است. همان‌طور که می‌دانیم دمای قهوه از رابطه‌ی نیوتن پیروی می‌کند. در این رابطه آهنگ سرد شدن قهوه متناسب است با اختلاف دمای قهوه با محیط بیرون،

$$\frac{d\theta}{dt} = -r(\theta - \theta_\circ) \qquad (۱)$$

که در این رابطه $\theta$ دمای قهوه در لحظه‌ی $t$ و $\theta_\circ$ دمای اتاق است. ضریب $r$ نرخ سرد شدن است و به مایع داخل لیوان بستگی دارد. همان‌طور که می‌دانیم حل این معادله‌ی دیفرانسیل به صورت زیر است:

$$\theta = \theta_i + (\theta_\circ - \theta_i)e^{-rt} \qquad (۲)$$

که $\theta_i$ دمای اولیه‌ی قهوه است.

در این‌جا می‌خواهیم ببینیم که معادله‌ی دیفرانسیلی مانند معادله‌ی (۱) را در حالت کلی چگونه می‌توان به کمک کامپیوتر حل کرد. معادله‌ی (۱) را می‌توان در حالت کلی‌تر به صورت زیر نوشت:

[۱] Euler algorithm

$$\frac{dy}{dx} = f(x,y) \tag{۳}$$

که $f(x,y)$ تابعی دل‌خواه از $x$ و $y$ است. از آن‌جا که در کامپیوتر چیزی به نام $\frac{dy}{dx}$ وجود ندارد، بنابراین نماد $d$ را به $\Delta$ تبدیل می‌کنیم:

$$\frac{\Delta y}{\Delta x} = f(x,y) \tag{۴}$$

که این معادله را می‌توان به صورت زیر هم نوشت:

$$\Delta y = f(x,y)\Delta x \tag{۵}$$

در قدم بعد باید تعریفی از $\Delta x$ و $\Delta y$ بکنیم. $\Delta y$ را تعریف می‌کنیم تغییرات $y$ از نقطه‌ی $n$ تا $n+۱$ به ازای جابجایی از $x_n$ به $x_{n+۱}$. برای راحتی کار از این به بعد $\Delta x$ را ثابت و برابر $h$ می‌گیریم.

$$y_{n+۱} - y_n = f(x_n, y_n)h \tag{۶}$$

معادله‌ی بالا خیلی ساده به ما یک الگوریتم می دهد. بدین معنی که مقدار $y$ در هر نقطه را می‌توان از $x$ و $y$ آن در نقطه‌ی قبلی به دست آورد. یعنی:

$$y_۰ \tag{۷}$$

$$y_۱ = y_۰ + f(x_۰, y_۰)h$$

$$y_۲ = y_۱ + f(x_۱, y_۱)h$$

$$y_۳ = y_۲ + f(x_۲, y_۲)h$$

...

به این روش حل معادله‌ی دیفرانسیل، «الگوریتم اویلر» گویند که ساده‌ترین الگوریتم برای حل معادلات دیفرانسیل است:

$$y_{n+۱} = y_n + f(x_n, y_n)h \tag{۸}$$

اجرای برنامه با این الگوریتم بسیار ساده است. تنها کافی است که $y_۰$، محدوده‌ی $x$ و تابع $f$ را بدانیم تا بتوانیم یک لیوان قهوه را به راحتی سرد کنیم.

# ۲ انواع خطاها در شبیه‌سازی

سوالی که ممکن است در این‌جا مطرح شود این است که جوابی که با الگوریتم اویلر به دست آمده، چقدر به جواب واقعی نزدیک است. مشکل اول در $h$ است. می‌دانیم که رابطه‌ی (۸) وقتی دقیق است که $h$ به سمت صفر میل کند. ولی وقتی که $h$ محدود است رابطه‌ی (۸) درست نیست. رابطه‌ی (۸) را در واقع می‌توان به صورت زیر نوشت: هست:

$$y(x_n + h) = y(x_n) + y'(x)h|_{x=x_n} + \dots \qquad (۹)$$

که بسط تیلور تا مرتبه‌ی اول بر حسب $h$ است که از مرتبه‌ی دوم $h$ در آن صرف‌نظر کرده‌ایم. اگر بسط را تا مرتبه‌ی دوم ادامه دهیم، به صورت

$$y(x_n + h) = y(x_n) + y'(x)h|_{x=x_n} + \frac{۱}{۲}y''(x)h^۲|_{x=x_n} + \dots \qquad (۱۰)$$

در می‌آید. در واقع در این جا در الگوریتم اویلر بسط تیلور را بریده‌ایم و خطایی که مرتکب می‌شویم از مرتبه‌ی $h^۲$ است. به این خطا اصطلاحاً «خطای قطع بسط تیلور» یا به طور ساده‌تر «خطای قطع»[۲] گویند. در شکل ؟؟ این خطا را نشان داده‌ایم. همان‌طور که می‌بینید این خطای در یک قدم است و در حل معادله‌ی دیفرانسیل درهر قدم چنین خطایی به خطای قدم قبل اضافه می‌شود. دقت کنیم که وقتی $h$ (طول قدم‌ها) را کوچک می‌کنیم، از آن طرف داریم تعداد قدم‌ها را بزرگ می‌کنیم. تعداد کل قدم‌هایی که باید طی کنیم از مرتبه‌ی $\frac{۱}{h}$ است پس خطای نهایی ما از مرتبه‌ی $h^۲ \times \frac{۱}{h} \sim h$ خواهد بود. به این خطا، «خطای سراسری[۳]» (خطای کلی) و به خطای هر قدم «خطای موضعی[۴]» گویند. به توانی که نشان‌دهنده‌ی مرتبه‌ی خطای سراسری است هم «مرتبه‌ی الگوریتم» گویند. بنابراین تا حالا می‌دانید که الگوریتم اویلر، الگوریتمی از مرتبه‌ی ۱ است.[۵]

سوال بعدی که مطرح می‌شود این است که $h$ را تا چه حد می‌توان در کامپیوتر کوچک کرد. دیدیم که با کوچک کردن $h$ تعداد قدم‌هایی شبیه‌سازی زیاد شده و در نتیجه زمان شبیه‌سازی هم زیاد می‌شود. امروزه کامپیوترهای سریعی داریم و قبل از اینکه به این محدودیت زمانی برسیم، به محدودیت مکانی در ذخیره‌ی اطلاعات می‌رسیم. در کامپیوتر همیشه اعداد با تعداد ارقام بامعنی مشخصی ذخیره می‌شوند. در این گونه موارد با خطای دیگری به نام «خطای گردکردن[۶]» مواجه می‌شویم. این خطا از آن‌جا ناشی می‌شود که در کامپیوتر تعداد ارقام بامعنی‌ای که ذخیره می‌شود محدود است. فرض کنید کامپیوتری دارید که تنها اعدادی را با ۴ رقم معنی‌دار ذخیره می‌کند؛ بنابراین اگر عددمان برای مثال ۴۲.۳۳ باشد، کامپیوتر هر عددی را که در ادامه باشد را دور می‌ریزد. بنابراین مثلاً اگر عدد ۰.۰۰۰۴ را با این عدد جمع کنیم، جواب همان ۴۲.۳۳ خواهد بود. یا به عنوان مثالی دیگر فرض کنید در کامپیوتری که تنها تا ۴ رقم بامعنی را نگه می‌دارد، عدد ۱ را یک میلیون بار با خودش جمع کنیم. وقتی در بار ۱۰۰۰۰ ام عدد ۱ به جمع که عدد ۹۹۹۹ است اضافه کنیم، جمع ۱۰۰۰۰ می‌شود؛ ولی چون کامپیوتر تنها تا ۴ رقم بامعنی را نگه می‌دارد، عدد ۱۰۰۰۰ به صورت ۱۰۰۰ در کامپیوتر ذخیره می‌شود و صفر آخر دور ریخته می‌شود. به همین ترتیب جمع یک میلیون ۱ هم همان ۱۰۰۰ خواهد شد.

بنابراین اگر برای مثال سرد شدن قهوه، که جواب واقعی را می‌دانیم، خطای شبیه‌سازی (اختلاف دمای محاسبه شده از شبیه‌سازی و جواب واقعی) را برای $h$ های مختلف رسم کنیم نمودار ۱ به دست می‌آید. همان‌طور که می‌بینید برای $h$

[۲] truncation error
[۳] global error
[۴] local error
[۵] دقت کنیم که فرض ما این است با توابع تعیینی (deterministic) کار می‌کنیم. اگر تابع $y$ تابعی تصادفی باشد، می‌شود نشان داد که $\delta y \sim h^{۱/۲}$. در بخش‌های بعد به این‌گونه توابع می‌پردازیم.
[۶] rounding error

شکل ۱: نمودار خطای شبیه‌سازی بر حسب مقدار طول قدم $h$.

های بزرگ رابطه‌ای خطی برقرار است. این به خاطر این است که الگوریتم اویلر از مرتبه‌ی ۱ است. همین‌طور که $h$ را کوچک می‌کنیم، خطا هم کم می‌شود تا جایی که خطای گرد کردن ظاهر می‌شود و از آن‌جا به بعد خطا شروع به بزرگ شدن می‌کند. این نمودار کوچکترین مقدار $h$ که با آن الگوریتم اویلر کمترین خطا را دارد، نشان می‌دهد.

## ۳ پایداری الگوریتم

گاهی در حل بعضی معادلات دیفرانسیل دیده می‌شود که الگوریتم‌ها ناپایدار هستند. ناپایداری در برنامه بدین صورت ظاهر می‌شود که تا مدتی برنامه به درستی کار می‌کند ولی ناگهان جواب‌های بی‌نهایت و غیرفیزیکی از حل معادلات دیفرانسیل به دست می‌آید. این ناپایداری‌ها دلایل مختلفی دارند که عموماً یا به خاطر خود الگوریتم است یا به خاطر نوع گسسته‌سازی ما. بعضی الگوریتم‌ها ذاتاً ناپایدارند و بعضی دیگر برای بعضی مسائل پایدار و برای بعضی دیگر از خود ناپایداری نشان می‌دهند.

یکی از دلایل ناپایداری می‌تواند این باشد که الگوریتمی که برای مسئله می‌نویسیم دارای نقاطی باشد که از طرفی جاذب است و از طرفی دافع؛ الگوریتم اویلر برای مسئله سرد شدن قهوه پایدار است چون وقتی که قهوه به دمای اتاق رسید اگر به خاطر خطاهای کامپیوتری مقداری کمتر از دمای اتاق بگیرد در قدم بعد دمای قهوه به سمت دمای اتاق گرم می‌شود. در این مسئله دمای اتاق یک جواب پایدار است یعنی از هر سمتی که به آن نزدیک شویم به سمت آن خواهیم رفت. ولی اگر شکل معادله‌ی دیفرانسیل به صورتی بود که از طرفی به جواب نزدیک شویم ولی از طرف دیگر از جواب ما را دور بکند. برای مثال اگر به جای معادله (۱) معادله

$$\frac{d\theta}{dt} = -r|\theta - \theta_0| \tag{۱۱}$$

را داشتیم، اگر از $\theta_0$ رد شویم، مقدار $\theta$ در هر قدم از جواب دور خواهد شد. به چنین جواب‌هایی جواب ناپایدار گویند. الگوریتم اویلر برای این مسئله ناپایدار است چون تا زمانی جواب درست می‌دهد ولی از زمانی به بعد جواب اشتباهی می‌دهد. به عنوان مثالی دیگر معادله‌ی دیفرانسیل زیر را در نظر بگیرید:

$$\dot{y} = -\sqrt{|y|} \tag{۱۲}$$

اگر این معادله را نیز با الگوریتم اویلر حل کنیم، داریم:

$$y_{n+۱} = y_n - h\sqrt{|y_n|} \tag{۱۳}$$

بنابراین اگر $h\sqrt{|y_n|} > y_n$ برقرار شود (یا $y_n < h^۲$)، از آن جا به بعد الگوریتم ناپایدار می‌شود.
ناپایداری‌ها ممکن است به دلایل دیگری هم بوجود بیایند که در بخش‌های بعد به آن‌ها خواهیم پرداخت.