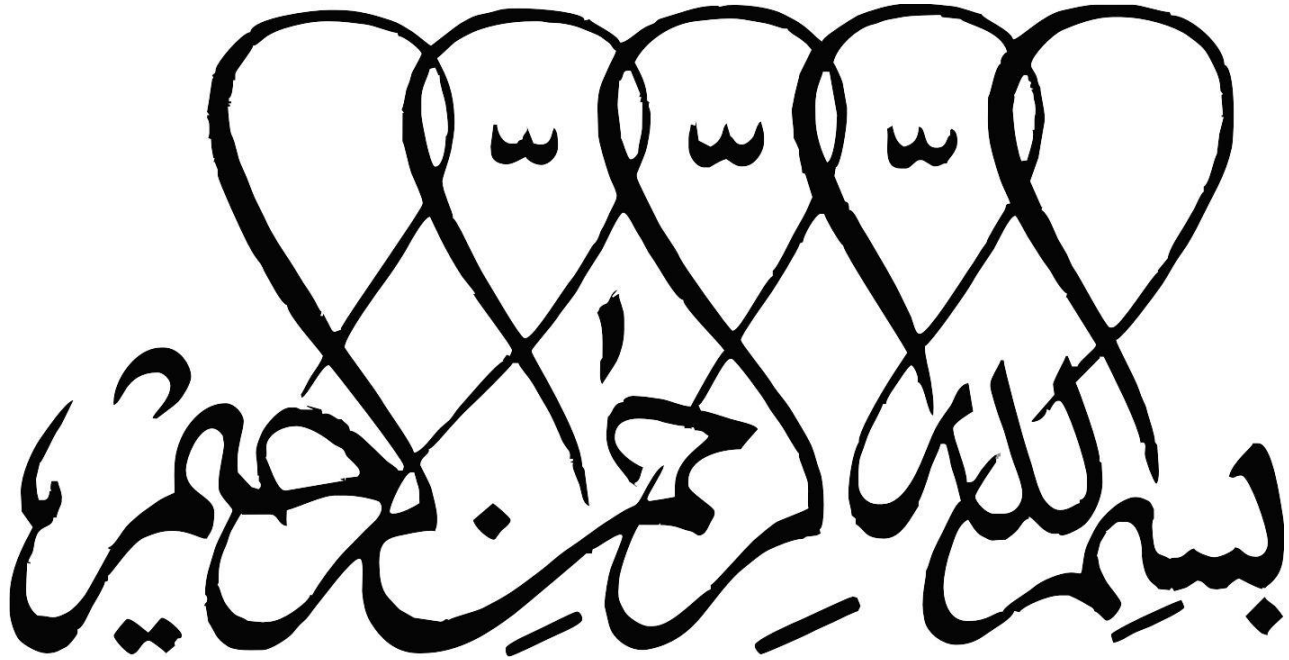
بسم الله الرحمن الرحيم

عناوین

محاسبات عددی

۱- حل معادله دیفرانسیل مرتبه اول

۱-۱ الگوریتم اویلر

۱-۲ انواع خطاها در شبیه سازی

۲- حل معادله دیفرانسیل مرتبه دوم

حل معادله دیفرانسیل مرتبه اول

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،

---

[۱] Euler algorithm

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

که در این رابطه θ 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 θ_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:

$$\frac{dy}{dx} = f(x, y) \qquad (۳)$$

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 Δ 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 Δx و Δy بکنیم. Δy 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 Δx 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.

انواع خطاها در شبیه سازی

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:

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

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

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

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.

حل معادله دیفرانسیل مرتبه دوم

در این بخش به حل معادلات دیفرانسیل مرتبه‌ی دوم می‌پردازیم و علاوه بر الگوریتم اویلر با الگوریتم‌های دیگری هم برای حل معادلات دیفرانسیل آشنا می‌شویم.

۱ الگوریتم اویلر

معادلات دیفرانسیل مرتبه‌ی دوم وقتی که با الگوریتم اویلر حل می‌شود فرق چندانی با حل معادله‌ی دیفرانسیل مرتبه‌ی اول ندارد. معادله‌ی دیفرانسیل مرتبه‌ی دوم زیر را در نظر بگیرید:

$$\ddot{y} = f(x, y, \dot{y}) \tag{۱}$$

این معادله را می‌شود در دو مرحله حل کرد. ابتدا معادله‌ی (۱) را به دو معادله‌ی دیفرانسیل مرتبه‌ی اول زیر تبدیل می‌کنیم:

$$\dot{y} = \frac{dy}{dt} \tag{۲}$$

$$\ddot{y} = \frac{d\dot{y}}{dt}$$

و سپس هر کدام را با الگوریتم اویلر حل می‌کنیم:

$$y_{n+1} = y_n + \dot{y}_n h + O(h^2) \tag{3}$$

$$\dot{y}_{n+1} = \dot{y}_n + f(x_n, y_n, \dot{y}_n)h + O(h^2)$$

که در جمله‌ی دوم به جای مشتق دوم از تابع f استفاده کردیم. همان‌طور که می‌بینید در هر دو جمله بسط تیلور را تا مرتبه‌ی دوم h نوشتیم. اگر معادله‌ی (۱) رابطه‌ی نیوتن باشد، x زمان، y مکان، $\dot{y}$ سرعت، و $\ddot{y}$ شتاب خواهد بود و معادلات (۳) به صورت

$$x_{i+1} = x_i + v_i h \tag{4}$$

$$v_{i+1} = v_i + a_i h$$

در می‌آید. همان‌طور که می‌بینید الگوریتم اویلر را می‌توان به هر معادلات دیفرانسیل با هر مرتبه‌ای تعمیم داد و نوشت. اگرچه الگوریتم اویلر ساده‌ترین الگوریتمی است که برای حل معادلات دیفرانسیل می‌توان پیشنهاد کرد، اما بهترین نیست. در ادامه با الگوریتم‌های دیگری که دارای خطای کمتری هستند، آشنا خواهیم شد.

۲- گسسته سازی

اگر بسط تابع f را در اطراف نقطه‌ی $x_۰$ به صورت زیر نشان دهیم:

$$f(x_۰) = f_۰$$

$$f_{\pm ۱} = f(x_۰ \pm h) = f_۰ \pm f'h + \frac{f''}{۲}h^۲ \pm \frac{f'''}{۶}h^۳ + O(h^۴) \qquad (۵)$$

$$f_{\pm ۲} = f(x_۰ \pm ۲h) = f_۰ \pm ۲f'h + ۲f''h^۲ \pm \frac{۴}{۳}f'''h^۳ + O(h^۴)$$

ساده‌ترین مشتق‌گیری، مشتق دونقطه‌ای است.

$$f' = \frac{f_۱ - f_۰}{h} + O(h) \qquad (۶)$$

می‌توان مشتق را به کمک سه نقطه (نقطه‌ی قبلی، نقطه‌ی مورد نظر و نقطه‌ی بعدی) به دست آورد و مشتق سه‌نقطه‌ای را این گونه نوشت:

$$f' = \frac{f_{+۱} - f_{-۱}}{۲h} + O(h^۲) \tag{۷}$$

و مشتق پنج‌نقطه‌ای را به صورت زیر:

$$f' = \frac{۱}{۱۲h}(f_{-۲} - ۸f_{-۱} + ۸f_{۰} - f_{۲}) + O(h^۴) \tag{۸}$$

همان‌طور که می‌بینید وقتی از تعداد نقاط بیشتری برای محاسبه‌ی مشتق حساب می‌کنیم، مشتق با دقت بیشتری محاسبه می‌شود.

۳- الگوریتم ورلت (verlet)

اگر $x_{n+۱}$ و $x_{n-۱}$ را بنویسیم:

$$x_{n+۱} = x_n + v_n h + \frac{۱}{۲} a_n h^۲ + \beta_n h^۳ + \gamma_n h^۴$$

(۱۶)

$$x_{n-۱} = x_n - v_n h + \frac{۱}{۲} a_n h^۲ - \beta_n h^۳ + \gamma_n h^۴$$

و با هم جمع کنیم:

$$x_{n+۱} = ۲x_n - x_{n-۱} + a_n h^۲ + O(h^۴) \qquad (۱۷)$$

به این روش محاسبه‌ی مکان در هر قدم، «الگوریتم ورلت» گویند. در این الگوریتم مکان در هر قدم از روی دو قدم قبلی به دست می‌آید. با این کار می‌توان خطا را تا مرتبه‌ی $O(h^۴)$ کاهش داد. برای محاسبه‌ی سرعت در این الگوریتم، در معادله (۱۶)اگر مقادیر $x_{n+۱}$ و $x_{n-۱}$ را از هم کم کنیم، داریم:

$$x_{n+۱} - x_{n-۱} = ۲v_n h + O(h^۳)$$

که از روی آن به راحتی می‌توان سرعت را محاسبه کرد:

$$v_n = \frac{x_{n+۱} - x_{n-۱}}{۲h} \qquad (۱۸)$$