MIT AçıkDersSistemi

http://ocw.mit.edu

18.034 İleri Diferansiyel Denklemler

2009 Bahar

Adi Diferansiyel Denklemlerin Çözümleri için

Adım Boyu Uyarlamalı Sayısal Yöntemler

Oleg Golberg

19 Mayıs 2007

1. Giriş

$$y'(x) = f\big(x, y(x)\big), \qquad y(0) = y_0 \tag{1}$$

başlangıç değer problemini ele alalım. $y(t)$ değerinin yaklaşık değerini bulmaya yönelik Runge-Kutta yöntemine benzer pek çok sayısal algoritma, $[0, t]$ aralığında seçilen bir noktalar kümesi için hesaplama yapar. Seçilen noktalar genellikle h adım boyuna sahip bir aritmetik seri oluşturur. h adım boyu küçüldükçe, algoritmalar daha hassas sonuçlar verirler. Ancak, pratikte $y(t)$'nin yaklaşık değerini verilen bir hassasiyete göre bulmak gerektiğinden h için uygun bir değer seçme problemiyle karşılaşırız. Algoritmanın bir iterasyonu izin verilen hatadan daha büyük hata veriyorsa, h'yi ikiye bölmek şeklindeki bir naif yaklaşım çoğunlukla tatmin edici sonuçlar verir. Bu yaklaşımda bir önceki iterasyondan elde edilen tüm sonuçlar göz ardı edilir. Bu bilgiyi izin verilen büyüklükte bir hata üretecek h dereğini tahmin etmekte kullanmak da mümkündür. h için uyarlamalı bir iterasyon kullanarak algoritmayı hesaplama açısından daha verimli hale getirebiliriz. Runge-Kutta ve diğer algoritmalar için de benzer yöntemler bulunsa da, kolay olması açısından, uyarlamalı adım boyu yöntemini Euler integrasyonuna nasıl uygulanacağını göstereceğiz. Euler integrasyonundan kaynaklanan hatayı tahmin ederek h için bir uyarlamalı yineleme (recurrence) bağıntısı elde edip, uyarlamalı ve naif yöntemlerin hesaplama verimliliğini karşılaştıracağız.

2. Standart Euler İntegasyonu

(1) ile aynı şekle sahip bir başlangıç değer problemini ele alalım. y değerinin $x = t$ noktasındaki yaklaşık değerini bulmak istediğimizi varsayalım. Bu amaçla $[0, t]$ aralığını her birinin uzunluğu $h = t/n$ olan n aralığa bölelim. Ortalama değer teoreminden her x için

$$y(x + h) = y(x) + h\, f\big(x, y(x)\big) + c\, h^2 \tag{2}$$

eşitliğini elde ederiz. Burada $\xi \in (x, x + h)$ belli bir sayı olmak üzere,

$$c = \frac{f'\big(\xi, y(\xi)\big)}{2} \tag{3}$$

dir. Bu gözlem, adi diferansiyel denklemlerin çözümünde sıkça kullanılan basit bir sayısal yöntem olan Euler integrasyonunu (metodunu) ortaya çıkarır. $k = 1,2, \dots, n$ için $t_k = kh$ olsun. Bu durumda, yinelemeli dizi

$$y_{k+1} = y_k + hf(t_k, y_k), \;\; k \geq 0 \tag{4}$$

olarak tanımlanır.

(2) den dolayı (4)'ün her adımında ortaya çıkan hata, c değeri $y''(\xi)$ ile orantılı olmak üzere, ch^2 dir. Pratikte, çoğu zaman $y(t)$ çözümünün yaklaşık değerini, verilen bir hassasiyette bulmaya ihtiyacımız vardır. Eğer bir sabit h adım boyu ile standart Euler integrasyonunu kullanırsak, c katsayısı hakkında herhangi bir sonuca ulaşamayız. Bu nedenle, y_n istenen hassasiyetle $y(t)$ ye yaklaşana kadar, n nin büyük değerleri için iterasyon yapmaya ihtiyacımız vardır. Başka bir deyişle, bir h_i dizisi oluşturarak, $i = 1,2, \dots$ için $y_0^{(i)} = y_0$ ve

$$y_{k+1}^{(i)} = y_k^{(i)} + hf\Big(t_k, y_k^{(i)}\Big), \;\; k \geq 0 \tag{5}$$

bağıntılarını kullanarak $|y_{n_{i+1}}^{(i+1)} - y_{n_i}^{(i)}|$ değeri yeterince küçük oluncaya kadar $y_n^{(i)}$ değerini hesaplamaya ihtiyacımız vardır. Eğer algoritmanın uygulamasında reel sayıların sabit hassasiyetli gösterimleri kullanılırsa, ki pratikte genelde yapılan budur, (5)'in her adımının karmaşıklığı sabittir. Bu nedenle $y_n^{(i)}$ değerini n-adım integrasyonla hesaplamanın karmaşıklığı n ile doğrusal olarak artar. n_{min} istenen hassasiyete ulaşmak için gerekli en küçük adım sayısı olsun. Bu durumda hedefimiz (5)'in toplam iterasyon sayısını azaltmak ve n_{min} den çok büyük olan n sayılarından kaçınmak olur. Bir doğal yol, her iterasyonda n adım sayısını iki katına çıkarmaktır. Başka bir deyişle $h_{i+1} = h_i/2$ şeklinde bir basit yineleme ile verilir. Bu şekilde $\log_2 n_{min}$ iterasyon gerekir ve kullandığımız n'lerin toplamı $2n$'i geçmez.

3. Uyarlamalı Adım Boyu Yöntemi

Euler intregrasyon algoritmasının her iterasyonunun, n değerinin her adımda iki katına çıkarıldığı basit yaklaşımda kullanılandan daha fazla veri sağladığı kolaylıkla görülebilir. Üstelik, istenen hassasiyet düzeyi bir sonraki iterasyon için adım boyu hesaplanırken kullanılmaz. Bu probleme getirilebilecek ve algoritmayı optimize edecek çözümlerden biri her adımın meydana getireceği hataya bakmaktır. (5) iterasyonunun her adımının ch^2 şeklinde bir hataya neden olduğunu ve c'nin değişim hızının $f''(\xi)$ ile orantılı olduğunu bulmuştuk. Her adım için hatayı

$$\epsilon_k^{(i)} = y(t) - y_k^{(i)} \tag{6}$$

şeklinde tanımlayalım. O zaman $\left|\epsilon_k^{(i)}\right| < \epsilon$ eşitsizliğinin sağlandığı durumları arıyoruz. Eğer (2)'deki c katsayısının sabit olduğunu kabul edersek, $y(t)$'nin yaklaşık değerini daha etkili verecek bir h_i dizisi oluşturabiliriz. Eğer c sabit olursa, $y_n^{(i)}$ değerini hesaplarken oluşan birikmiş hata, a bir sabit olmak üzere,

$$\epsilon_{n_i}^{(i)} = n_i c h_i^2 = \frac{t}{h_i} c h_i^2 = t c h_i = a h_i \tag{7}$$

olur. h_1 ve h_2 adım boyuna sahip iki farklı iterasyonu ele alalım. Her iterasyondaki hatanın tanımını hatırlayacak olursak,

$$y(t) - y_{n_1}^{(1)} = \epsilon_{n_1}^{(1)} = a h_1 \tag{8}$$

$$y(t) - y_{n_2}^{(2)} = \epsilon_{n_2}^{(2)} = a h_2 \tag{9}$$

denklemlerini elde ederiz. Birinciden ikinciyi çıkararak,

$$y_{n_2}^{(2)} - y_{n_1}^{(1)} = a(h_1 - h_2) \tag{10}$$

$$a = \frac{y_{n_2}^{(2)} - y_{n_1}^{(1)}}{h_1 - h_2} \tag{11}$$

buluruz.

Sonraki iterasyonda istenen aralıkta bir hata elde etmek için h_3'ün hangi şartı sağlaması gerektiğini bulalım.

$$\epsilon > |\epsilon_{n_3}^{(3)}| = |a h_3| = \frac{h_3 |y_{n_2}^{(2)} - y_{n_1}^{(1)}|}{h_1 - h_2} \tag{12}$$

$$h_3 > \frac{(h_1 - h_2)\epsilon}{|y_{n_2}^{(2)} - y_{n_1}^{(1)}|} \tag{13}$$

Böylelikle, h_i dizisi tarafından belirlenen ve bir $q < 1$ katsayısı için

$$h_{i+2} = q \frac{(h_i - h_{i+1})\epsilon}{|y_{n_{i+1}}^{(i+1)} - y_{n_i}^{(i)}|} \tag{14}$$

şeklinde tanımlanan basit uyarlamalı adım boyu algoritması elde etmiş oluruz.

4. Karşılaştırma

Aşağıdaki başlangıç değer problemini kullanarak naif iterasyon çiftleme kullanılan Euler integralleme algoritması ile yukarıda gösterilen uyarlamalı adım boyu algoritmasının verimliliğini karşılaştıralım:

Euler integrasyonun verimliliğini, naif katlama iterasyonu ve uyarlamalı adım boyu metodunu kullanarak,

$$y'(x) = 1 - x + 4y(x), \qquad y(0) = 1 \tag{15}$$

başlangıç değer problemine uygulayarak karşılaştıralım

Amaç 10^{-3}'ten küçük bir hatayla $y(1)$ değerini yaklaşık olarak bulmaktır. Aşağıda Euler integrasyonunun basit bir uygulamasını 8-ondalıklı kayan noktalı sayılar kullanan Common Lisp programını kullanarak yapacağız.

```
(defun euler (f tn n y0)
  "Approximate y(tn) with n-step Euler method"
  (declare (type double-float x y0)
           (type integer n))
  (let ((h (/ tn n)) (y y0))
    (progn
      (loop for i from 0 to (- n 1)
         do (incf y (* h (funcall f (* i h) y))))
      y)))
```

(15) başlangıç değer probleminin tam çözümü kolaylıkla

$$y(x) = -\frac{3}{16} + \frac{x}{4} + \frac{19}{16}\, e^{4x} \tag{16}$$

şeklinde bulunabilir. Bu durumda y(1) ~ 64.897 olur. İlk adım boyunu $h_1 = 0.1$ seçelim ve $h_{i+1} = 2h_i$ ile tanımlanan h_i dizisi için önce $y_n^{(i)}$ değerlerini hesaplayalım.

$$y_{10}^{(1)} \approx 34.411$$
$$y_{20}^{(2)} \approx 45.588$$
$$y_{40}^{(3)} \approx 53.807$$
$$y_{80}^{(4)} \approx 58.916$$
$$y_{160}^{(5)} \approx 61.786$$
$$y_{320}^{(6)} \approx 63.310$$
$$y_{640}^{(7)} \approx 64.095$$
$$y_{1280}^{(8)} \approx 64.494$$
$$y_{2560}^{(9)} \approx 64.695$$
$$y_{5120}^{(10)} \approx 64.796$$
$$y_{10240}^{(11)} \approx 64.847$$
$$y_{20480}^{(12)} \approx 64.872$$
$$y_{40960}^{(13)} \approx 64.885$$
$$y_{81920}^{(14)} \approx 64.891$$
$$y_{163840}^{(15)} \approx 64.894$$
$$y_{327680}^{(16)} \approx 64.896$$
$$y_{655360}^{(17)} \approx 64.897$$

Bu nedenle, 17 iterasyon gerekir ve yaklaşık değeri hesaplamanın maliyeti, (5) deki bir adımın hesaplama maliyeti m (bizim uygulamamızda sabit) olmak üzere,

$$10m + 20m + \cdots + 655360m = 10(2^{16} - 1)m = 1310710m \tag{17}$$

olarak bulunur. Şimdi ilk adım boylarını $h_1 = 0.1, h_2 = 0.05$ muhafaza edelim fakat (14) uyarlamalı yinelemesini $q = 0.9$ katsayısıyla kullanalım. İlk iki yaklaşık değer

$$y_{10}^{(1)} \approx 34.411, \; y_{20}^{(2)} \approx 45.588$$

aynıdır. İzin verilen hata $\epsilon = 0.001$ olduğu için

$$h_3 = 0.9\frac{(0.1 - 0.05) \times 0.001}{|45.588 - 34.411|} \approx 4.026 \times 10^{-6} \tag{18}$$

elde ederiz. Bu durumda $n_3 = [1/h_3] = 248378$ olur. Beklendiği gibi iterasyon izin verilen sınıra yakın bir hata verdi.

$$y_{248378}^{(3)} \approx 64.896 \tag{19}$$

olduğu görülür.

Bir sonraki iterasyonda

$$h_4 = 0.9\frac{h_2 - h_3}{|64.896 - 45.588|} \approx 2.331 \times 10^{-6} \tag{20}$$

olduğundan $n_4 = [1/h_4] = 429086$ olur. Dördüncü iterasyon istenen duyarlılıkta

$$y_{429086}^{(4)} \approx 64.897 \tag{21}$$

yaklaşık değerini verir. Böylece, bir basit uyarlamalı adım yöntemi kullanıldığında, gereken iterasyon sayısı sadece 4 ve toplam hesap maliyeti

$$10m + 20m + 248378m = 677494m \tag{22}$$

olur. Bu maliyet adım sayısı katlanarak elde edilen yaklaşımın maliyetinin yaklaşık yarısıdır.

5. Sonuç

Uyarlamalı adım boyu yöntemini Euler integrasyonuna nasıl başarıyla uygulayacağımızı ve hesaplamayı daha verimli yapabileceğimizi gördük. Benzer fakat daha gelişmiş yöntemler, Runge-Kutta gibi daha verimli sayısal yöntemlere uygulanarak, pratikte de sıkça kullanılan

Runge-Kutta-Fehlberg ve Dormand-Prince yöntemleri gibi yöntemler geliştirilebilir. Örneğin Dormand-Prince yöntemi Matlab'ın adi diferansiyel denklem çözücülerinin birinde kullanılmaktadır.