Bilangan Real pada Analisis Numerik

Komputer yang kita pakai saat ini menyimpan informasi dalam bentuk 1 dan 0 yang dinyatakan dalam tegangan yang disimpan dalam kapasitor. Lantas bagaimana bisa komputer kita saat ini dapat mengerti angka-angka lainnya mulai dari positif, negatif, nol dan bahkan pecahan yang masuk dalam kategori bilangan real? Jawabannya terdapat pada analisis numerik yang akan kita bahas kali ini.

1. Analisis Numerik

Dalam mencari solusi setiap persoalan matematika, tidak semuanya dapat dengan mudah dicari dengan menggunakan otak dan tangan ataupun kalkulator. Mungkin menyelesaikan persoalan $5X = 10$ dapat dengan mudah dan secara intuisi kita langsung tahu bahwa $X = 2$, tetapi bagaimana jika persoalannya $X^{45} + X^{23} + 23 = 0$? Hal ini akan sangat sulit untuk didapatkan jawaban pastinya sehingga baik untuk dilakukan pendekatan yang disebut sebagai analisis numerik. Analisis numerik hadir sebagai bagian studi yang mempelajari algoritma penyelesaian masalah matematika melalaui pendekatan pada jawaban aslinya. Sehingga yang didapat bukanlah jawaban "tepat" tetapi adalah jawaban yang sangat mendekati jawaban itu dengan galat yang sekecil mungkin.

Demikianlah halnya yang dapat dilakukan ketika hendak merepresentasikan bilangan real dalam komputer. Tidak mungkin komputer yang jumlah kapasitornya terbatas dapat berhasil merepresentasikan semua bilangan real yang tidak terbatas. Tidak semua, berarti beberapa dapat direpresentasikan secara sepenuhnya. Ya benar, untuk beberapa jenis bilangan real, komputer dapat merepresentasikannya secara tepat walaupun tetap saja ada batas maksimum dan minimumnya sesuai dengan jumlah kapasitor tadi.

Berikut kita akan membahas bagaimana beberapa jenis bilangan real direpresentasikan dalam komputer melalui metode analisis numerik serta bagaimana error yang mungkin terjadi di dalamnya.

2. Bilangan Bulat

Representasi bilangan bulat dalam computer adalah *exact value*. Operasi aritmatika didalamnya juga menghasilkan hal yang sama dengan prasyarat (i) hasilnya berada dalam rentang nilai angka yang dapat direpresentasikan dengan jumlah kapasitor dan (ii) setiap

operasi pembagian hanya melihat hasil bilangan bulatnya saja dengan mengabaikan angka yang berada dalam koma.

Bilangan bulat non-negatif akan sangat mudah direpresentasikan. Semua digit direpresentasi-kan dengan bit 0 dan 1, sehingga dapat langsung diinterpretasikan sebagai string biner. Sebagai contoh dengan delapan bit, kita bisa menuliskan

$00110111 = (1 + 2 + 4 + 16 + 32)_{10} = 55_{10}$

Agar dapat merepresentasikan bilangan negatif juga, maka harus dilakukan pemisahan antara bilangan negatif dan positif. Idenya adalah menganggap bit pertama sebagai penanda apakah bilangan tersebut negatif (yaitu ditandai angka 1) atau positif (yaitu ditandai angka 0). Metode yang paling sering digunakan untuk menhitung nilai representasi sebuah bilangan negatif adalah dengan metode ***two's complement*** yaitu***:***

Dalam representasi dari sebuah bilangan bulat positif X, inversi setiap bit ($0 \leftrightarrow 1$), dan tambahkan 1 untuk mendapatkan representasi dari $-X$.

Misalkan kita memiliki delapan buah kapasitor yang merepresentasikan masing-masing sebuah bit maka kita dapat merepresentasikan seperti berikut.

$+2 = 0000\ 0010$

$+1 = 0000\ 0001$

$0 = 0000\ 0000$

$-1 = 1111\ 1111$

$-2 = 1111\ 1110$

Dengan angka maksimum yang dapat direpresentasikan adalah sebesar $2^{n-1} - 1$. Sebagai contoh, untuk 32-bit kita dapat interval angka bulat antara

$2^{31-1} = 2147483647 = (01111111\ 11111111\ 11111111\ 11111111)_2$ dan

$-2^{31} = -2147483648 = (10000000\ 00000000\ 00000000\ 00000000)_2$

Sebagian besar *compiler* tidak memberikan pesan error jika ada angka dalam program yang melewati batas interval angka kecuali dalam beberapa kasus.

Demikianlah cara untuk bilangan bulat real dapat direpresentasikan dalam komputer. Error yang mungkin terjadi adalah saat melakukan pembulatan pada saat pembagian ataupun pada saat operasi yang mengakibatkan hasil berada di luat interval angka minimum dan maksimum yang dapat direpresentasikan oleh komputer. Untuk mengatisipasi error ini maka harus dipertimbangkan metode kita mencari hasil dan melihatnya, apakah memang tetap valid apabila menggunakan operasi tersebut dengan menggunakan bilangan bulat apa tidak. Karena masalah ini pastinya ada dan tidak mungkin untuk dihilangan ketika berbicara bilangan bulat dan komputer.

3. Bilangan Pecahan

Dalam komputer tidak ada bilangan rasional, namun semuanya dapat direpresentasikan dengan presisi yang sangat terbatas. Metode yang digunakan adalah dengan membagi setiap bit dengan aturan posisi tertentu seperti dalam standar IEEE 1 digit pertama merepresentasikan tanda negatif atau positif (S). Kemudian dilanjutkan 8 atau 11 digit berikutnya merupakan representasi eksponen dari angka (E). HIngga 23 atau 52 bit terakhir disebut sebagai *mantissa* yang merepresentasikan angka tersebut dengan digit paling kanan bernilai satu, kemudian digit kedua bernilai setengah, kemudian sebelahnya seperempat, demikian dijumlahkan apabila bernilai 1 dan diabaikan apabila bernilai 0 (F). Berikut adalah contoh konversinya.

Bit = 1 1000 0001 011 0000 0000 0000 0000 0000

S = 1 E = 1000 0001 F = 011 0000 0000 0000 0000 0000

(negatif) pangkat 2 representasi 1.375

Sehingga, dalam pecahan bernilai $-1{,}375\times2^2 = -5{,}5$

Hal ini tentu saja memiliki dampak error yang cukup besar karena lompatan antar satu bilangan ke bilangan lainnya hanya dapat dilakukan mencapai 0,5 sehingga disinilah sebenarnya peran analisis numerik itu sendiri banyak dipakai. Namun tetap saja, walaupun sudah dilakukan banyak pendekatan, sangat dimungkinkan kesalahan. Sehingga dalam dunia nyata, tidak disarankan sebenarnya menggunakan perhitungan bilangan pecahan menggunakan computer.

4. Bilangan Real pada Interpolasi

Interpolasi adalah suatu kegiatan berupa penyisipan suatu nilai diantara dua bagian berbeda untuk memperkirakan nilai suatu fungsi antara dua nilai yang telah diketahui. Pada komputer grafik, interpolasi digunakan untukmenggabungkan beberapa efek yang ingin dilakukan pada suatu obyek. Beberapa jenis interpolasi adalah : interpolasi linier dan interpolasi kuadratik.

4.1 Interpolasi Linear

Bentuk interpolasi yang paling sederhana adalah menghubungkan dua titik data dengan garis lurus. tehnik ini dinamakan interpolasi linear dan biasanya diilustrasikan dengan dua segitiga siku – siku yang sebangun. Persamaan dari interpolasi linier adalah :

$$f_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0).$$

Cara penulisan $f_1(x)$ menunjukkan bahwa ini adalah polinom interpolasi orde pertama (interpolasi lanjar).Perhatikan bahwa disamping menyatakan kemiringan garis y angmenghubugkan titik-titik, bentuk $\frac{f(x_1) - f(x_0)}{x_1 - x_0}$ adalah hampiran (aproksimasi) beda hingga terbagi dari turunan pertama.Umumnya semakin kecil selang diantara titik-titik data, semakin baik hampirannya.

Algoritma Interpolasi

1) Tentukan dua titik P1 dan P2 dengan koordinatnya masing-masing (x0,y0) dan (x1,y1)2)
2) Tentukan nilai x dari titik yang akan dicari

3) Hitung nilai y dengan : $f_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$.
4) Tampilkan nilai titik (x,y) yang baru.

4.2 Interpolasi Kudratik

Menentukan titik-titik antara 3 buah titik dengan menggunakan pendekatan fungsikuadrat 3 titik yang diketahui: P1(x1,y1), P2(x2,y2) dan P3(x3,y3).

Untuk memperoleh titik (x,y) digunakan rumus interpolasi kuadratik:

$$Y = Y1\frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + Y2\frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} + Y3\frac{(x-x_1)(x-x_2)}{(x_3-x_2)(x_3-x_1)}$$

Algoritma Interpolasi Kuadratik:

1) Tentukan 3 titik P1, P2 dan P3 dengan koordinatnya masing-masing(x1,y1),(x2,y2),dan(x3,y3).
2) Tentukan titik x dari titik yang akan dicari
3) Hitung nilai y dengan : $Y = Y1\frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + Y2\frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} + Y3\frac{(x-x_1)(x-x_2)}{(x_3-x_2)(x_3-x_1)}$
4) Tampilkan nilai titik yang terbaru.

Daftar Pustaka

https://www.academia.edu/7026353/Penelitian_Matematika_Suatu_Contoh_di_Analisa_Numerik. Diakses pada tanggal 2 Juni 2018, pukul 16.53.

http://www.lce.hut.fi/teaching/S-114.1100/lect_1.pdf. Diakses pada tanggal 2 Juni 2018, pukul 15.23.

Haruna, Ibrahim Umar. 2015. "Minimizing Error in Scientific Numerical Computation". Novelty Journals.

Pembagian Tugas

Eka (13516061) :

- Bagian 4 dan revisi bagian lainnya
- Finalisasi artikel

Hagai (13516136) :

- Bagian 1, 2 dan 3
- Posting di Github Pages