Validasi Teorema Geometri Pada Kasus Titik Berat Segitiga Secara Induktif Menggunakan Komputasi Simbolik

Vicky Julius Mawuntu

Program Studi Fisika FMIPAK, Universitas Negeri Manado, Kab. Minahasa, 95618, Indonesia

e-mail: vickymawuntu@unima.ac.id

Diterima 15 April 2023; Disetujui 28 April 2023

ABSTRAK

Komputasi simbolik menekankan pada komputasi eksak dari perwakilan data yang ada. Perwakilan eksak menyatakan bahwa sekalipun keluaran (*output*) kecil, data yang dihasilkan dengan menggunakan komputasi dapat menghasilkan ekspresi matematik yang lebih besar. Untuk mengatasi masalah ini, sebuah metode perlu digunakan dalam merepresentasikan data. Proses ini dilakukan untuk menyelesaikan suatu permasalahan fisis pada kasus titik berat pada segitiga. Bahasa pemrograman simbolik dapat mengubah bilangan rasional, sehingga komputasi simbolik ini dapat diaplikasikan untuk memvalidasi teorema geometri pada kasus titik berat segitiga secara induktif.

Kata kunci : Komputasi simbolik, geometri, titik berat segitiga

ABSTRACT

Symbolic computing emphasizes the exact processing of data representations. The exact representative states that even though the output is small, the data generated using processing can produce larger mathematical expressions. To overcome this problem, a method needs to be used in representing data. This process is carried out to solve a physical problem in the case of the center of gravity on a triangle. Symbolic programming languages can convert rational numbers, so that this symbolic processing can be applied to inductively validate geometric theorems in the case of triangular center.

Keywords : *Simbolic computation, geometry, triangle center*

1. PENDAHULUAN

Perkembangan komputasi memungkinkan kita untuk menerapkan beragam pendekatan baru untuk menyelesaikan berbagai masalah matematik maupun nonmatematik. Tren perkembangan komputasi saat ini mengarah pada bagaimana menciptakan model matematika untuk sebuah deskripsi dan simulasi sistem yang riil, serta menggunakan metode statistik yang kompleks untuk mengevaluasi kemungkinan terjadinya kasus-kasus tertentu, serta mereduksi kompleksitas yang ada (Buchberger, 1985; Kerr dkk., 2022).

Komputasi simbolik merupakan komputasi aljabar dalam lingkup studi ilmiah yang mengembangkan algoritma untuk memanipulasi ekspresi matematik dan objek matematik, dan menekankan pada komputasi eksak dari ekspresi variabel yang tidak memiliki angka (Davenport, 1985; Lample & Charton, 2019).

Komputasi simbolik juga bisa didefinisikan sebagai komputasi yang merupakan pemrograman yang tidak hanya mengolah data numerik tapi juga simbolik, terutama yang membentuk ekspresi matematik. Numerik dalam hal ini berarti mengenai bilangan riil. Bahasa pemrograman yang umum digunakan, baik untuk komputasi numerik maupun simbolik, ataupun keduanya antara lain: *Maple*, *Maxima*, *Mathematica*, *Reduce*, *Derive*, dan lain sebagainya.

Komputasi simbolik sudah sangat banyak digunakan dalam eksperimen matematik dan dalam mendesain persamaan yang digunakan dalam program numerik. Angka dasar (*basic number*) yang digunakan dalam komputasi simbolik adalah integer-integer. Integer ini memungkinkan kita untuk menentukan fraksi-fraksi yang sudah tidak bisa lagi direduksi menjadi dua integer (Gathen & Gerhard, 2003).

Pemrograman merupakan implementasi operasi aritmatika yang efisien. Selain angka dan variabel, setiap ekspresi matematik dapat diperlihatkan dalam rentetan operand. Representasi ini sangat fleksibel, sehingga hal-hal yang awalnya tidak memiliki ekspresi matematik kemudian dapat disajikan sebagai

ekspresi matematik (Geddes dkk., 1992; Lample & Charton, 2019)

2. KAJIAN LITERATUR

Dalam komputasi numerik, isi variabel berupa bilangan. Komputer menyediakan tempat yang terbatas untuk bilangan. Data yang akan diproses berupa data bilangan riil. Sedangkan untuk komputasi non-numerik, komputasinya berupa huruf, seperti huruf Latin a sampai dengan z, huruf Yunani α sampai ω, karakter lain yang berupa simbol, dan gambar dengan cara memberi warna pada pixel (unit terkecil pada layar). Komputasi simbolik sendiri memproses bilangan dan huruf yang membentuk ekspresi matematik, seperti: $2x^2 + 4xy$.

Angka pada komputasi perlu untuk dibatasi. Tujuan utamanya adalah untuk mengurangi waktu proses, sehingga mempercepat hasil data yang ingin diperoleh. Ada dua jenis presisi yang ada dalam komputasi, yaitu *single precision*, dan *double precision*. *Single precision* memiliki kira-kira tujuh angka di belakang koma, sementara *double precision* memilliki angka di belakang koma. Dalam komputasi sendiri, bilangan bulat dianggap sebagai simbol. Isi dan variabel pada komputasi simbolik adalah berupa ekspresi matematik, dan bentuk ekspresi matematik paling sederhana adalah bilangan. Contoh ekspresi matematik dalam komputasi simbolik adalah polinom, contohnya:

$$\sum_{i=0}^{N} \boldsymbol{C_i x^i} = \boldsymbol{C_0 + C_1 x + C_2 x^2 + \cdots + C_N x^N} \quad (1)$$

Bilangan pada contoh di atas merupakan polinom orde 0. Contoh ekspresi matematik yang lain adalah teorema Taylor – McLaurin:

$$\boldsymbol{f(x)} = \frac{\boldsymbol{f^i(a)}}{\mathbf{i!}}\boldsymbol{(x-a)^i} \quad (2)$$

$$\boldsymbol{f(x)} = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(\mathrm{x} - \mathrm{a})^2 + \cdots \quad (3)$$

Untuk $a = 0$ deret Taylor-McLaurin menjadi

$$f(x) = f(0) + f'(0)(x) + \frac{f''(0)}{2!}(\mathrm{x})^2 + \cdots \quad (4)$$

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \quad (5)$$

$$\cos(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \quad (6)$$

Kasus tersebut dapat diselesaikan dengan cara komputasi dengan *script* sebagai berikut:

```
> tk := taylor(sin(x^2 - 1), x = 0, 4);
                tk := -sin(1) + cos(1) x^2 + O(x^4)
> tk1 := evalf(tk);
          tk1 := -0.8414709848 + 0.5403023059 x^2 + O(x^4)
```

Ada dua jenis persamaan dalam ekspresi matematik. Yang pertama adalah persamaan sintaktik (*syntactic equality*) yang merupakan persamaan ekspresi matematik yang memiliki arti yang sama sesuai dengan yang ditulis. Persamaan jenis kedua adalah persamaan semantic (*semantic equation*), yang merupakan persamaan dimana dua ekspresi matematik mewakili objek matematik yang sama, seperti:

$$(x+y)^2 = x^2 + 2xy + y^2 \quad (7)$$

Dari teorema Richardson, diketahui bahwa tidak akan muncul algoritma yang memutuskan apakah dua ekspresi matematik yang mewakili angka-angka tertentu merupakan persamaan semantic atau bukan, jika eksponensial dan logaritma muncul dalam ekspresi matematik tersebut.

Untuk menguji kesamaan dua ekspresi matematik, biasanya kita dapat membentuk persamaan tersebut ke dalam bentuk kanonik, atau membuat diferensiasinya dalam bentuk normal kemudian menguji persamaan sintatik dari hasil yang diperoleh. Kita dapat menggunakan *script* berikut:

```
> restart;
> prop01 := ((x - c)^2 / (a^2)) + (y^2 / b^2) = 1;
                prop01 := (x - c)^2 / a^2 + y^2 / b^2 = 1
> x := r·cos(theta); y := r·sin(theta);
                          x := r cos(θ)
                          y := r sin(θ)
> prop01;
                (r cos(θ) - c)^2 / a^2 + r^2 sin(θ)^2 / b^2 = 1
```

Dalam komputasi, biasanya kita perlu untuk melakukan simplifikasi terhadap sutau ekspresi matematik (Bruno dkk., 1983). Contoh aplikasi dari simplifikasi ini mengenai diferensiasi variabel x terhadap ekspresi matematik a^x dari ekspresi matematika berikut:

$$x.a^x.0 + a^x.\left(1.\log a + x.\frac{0}{a}\right) \quad (8)$$

Saat suatu ekspresi matematik ini merupakan ekspresi yang sulit dan tidak dapat diterima, maka prosedur simplifikasi perlu dilakukan. Simplifikasi normalnya dilakukan melalui aturan penulisan ulang (*rewriting rules*). Ada beberapa tahap penulisan ulang yang perlu dilakukan. Simplifikasi yang paling sederhana adalah dengan mereduksi ukuran ekspresi matematik, seperti $E - E \to 0$ atau $\sin(0) \to 0$. Simplifikasi ini akan diaplikasikan pada sistem aljabar komputer.

Kesulitan pertama akan muncul saat kita memasukkan operasi asosiatif seperti penjumlahan dan perkalian. Cara standar untuk menyelesaikan masalah sehubungan dengan operasi asosiatif ini adalah dengan mempertimbangkan angka sebarang dari *operand*, yaitu dengan menuliskan $a + b + c$ menjadi "+"(a, b, c), sehingga $a + (b + c)$ dan $(a + b) + c$ keduanya disimplifikasi menjadi "+"(a, b, c). Representasi internal dari ekspresi matematik, harusnya tidak memiliki substraksi atau divisi atau minus, diluar representasi angka.

Kesulitan kedua yang timbul adalah saat kita menggunakan sifat komutatif dari operasi penjumlahan dan perkalian. Masalahnya yaitu bagaimana mengenal bagian (*terms*) yang mirip dengan maksud mengkombinasikan ataupun menghilangkan bagian tersebut (*cancelling*). Dalam aplikasi *Maple*, fungsi *hash* (*hash function*) didesain untuk menggenerasi *collision* akibat kesamaan bagian (*term*) ini, yaitu dengan cara mengkombinasikan keduanya. Fungsi ini kemudian dapat mengenali ekspresi ataupun subekspresi matematik yang muncul beberapa kali dalam komputasi dan menyimpannya sekali. Hal ini tidak hanya berimbas pada penghematan ruang data, tapi juga akan mempercepat proses komputasi.

3. METODE PENELITIAN

Aplikasi *software* komputasi digunakan untuk melakukan kalkulasi simbolik sistem komputer aljabar (*computer algebra systems*), dimana sistem disini merupakan kompleksitas dari aplikasi utama yang digunakan untuk menyajikan data matematik pada komputer, yaitu sebuah bahasa pemrograman untuk memproses masukan menjadi keluaran dari bentuk ekspresi matematik. Operasi sistem akan berupa simplifikasi ekspresi matematika.

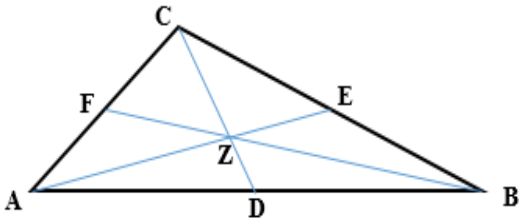
Bahasa pemrograman yang digunakan untuk komputasi simbolik pada studi ini adalah *Maple*. Program *Maple* memiliki kemampuan relatif baik untuk perhitungan matematis secara eksak maupun numerik. Dengan kemampuan yang dimiliki *Maple* menjadi program yang sangat interaktif. Komputasi yang ditawarkan berupa solusi aritmatika dasar, teori grafik, dan analis vektor, termasuk sarana visualisasi matematika dan *wordprocessing* khusus untuk notasi matematika dan bahasa pemrograman modern.

4. HASIL DAN PEMBAHASAN

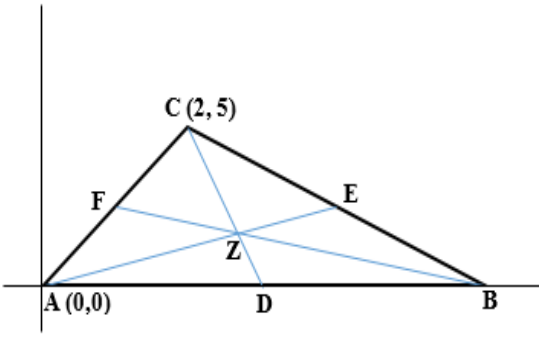
Komputasi Simbolik

Bahasa pemrograman simbolik dapat mengubah bilangan rasional. Pengolahan bilangan rasional dapat diaplikasikan untuk pembuktian teorema geometri secara induktif, yaitu pembuktian sebuah teorema dengan meninjau kasus-kasus khusus.

Secara matematis pembuktian perlu dilakukan secara deduktif, atau dilakukan dengan mengacu dari aksioma. Dalam segitiga, tiga garis berat bertemu di satu titik (Gambar 1).



Gambar 1. Titik potong dari garis tengah tiap sisi segitiga merupakan titik berat benda



Gambar 2. Masing-masing sudut pada segitiga diberi koordinat

Titik D di tengah-tengah AB → CD = garis perpotongsn titik berat;
Titik E di tengah-tengah AB → AE = garis perpotongan titik berat;

Titik F di tengah-tengah AC → BF = garis perpotongan titik berat;
CD dan AD berpotongan di Z (Gambar 2).

Variabel Z merupakan titik berat atau pusat massa. Titik berat ini merupakan jumlahan dari semua gaya berat (vektor) partikel penyusun segitiga yang nilainya sama dengan gaya yang bekerja pada benda, dan dengan demikian berat total segitiga bertitik tangkap di Z.

Penyelesaian masalah matematika di atas dapat didekati dengan penyelesaian induktif, yaitu dengan memilih kasus-kasus tertentu kemudian diberi angka.

Sehingga dapat dihitung persamaan garis CD sebagai berikut:

$$\mathrm{y} - \mathrm{y_C} = \frac{\mathrm{y_D} - \mathrm{y_C}}{\mathrm{x_D} - \mathrm{x_C}}(\mathrm{x} - \mathrm{x_C})$$

$$\mathrm{y} - 5 = \frac{0 - 5}{3 - 2}(\mathrm{x} - 2)$$

$$\mathrm{y} = 5 - \big(5(\mathrm{x} - 2)\big)$$

$$\mathrm{A_{1x}} + \mathrm{B_{1y}} + \mathrm{C_1} = 0 \qquad (9)$$

Prosedur di atas diulang hingga ketemu persamaan garis **AE:**

$$\mathrm{A_{2x}} + \mathrm{B_{2y}} + \mathrm{C_2} = 0 \qquad (10)$$

Perpotongan kedua garis **CD** dan **AE** akan menentukan solusi dari permasalahan titik berat segitiga.

Kalau dalam kasus tersebut kita menggunakan bilangan rasional, maka tidak aka nada pembulatan, sehingga ruas kiri akan bernilai sama dengan ruas kanan. Nilai yang diperoleh adalah eksak sehingga teorema dinyatakan benar untuk kasus ini. Untuk kasus-kasus yang lain, variabel A, B, C, dapat diganti untuk menyelesaikan persamaan tersebut.

Jika koordinat A, B, C bukan merupakan angka atau bilangan rasional, tapi berupa huruf dan simbol, maka kita perlu memasukkan koordinat A $(\mathrm{x}_a - \mathrm{y}_a)$, B $(\mathrm{x}_b - \mathrm{y}_b)$, dan C $(\mathrm{x}_c - \mathrm{y}_c)$ ke dalam program yang sudah disusun untuk angka atau bilangan rasional.

Ketika koordinat masuk pada perhitungan, maka meskipun $\mathrm{A_3X_z} + \mathrm{B_3Y_z} + \mathrm{C_3} = 0$ merupakan ekspresi matematika (dalam bentuk $\mathrm{x_a}, \mathrm{y_a}, \mathrm{x_b}, \mathrm{y_b}, \mathrm{x_c}, \mathrm{y_c}$ yang rumit), belum tentu komputer bisa menyimpulkan bahwa persamaan tersebut sama dengan nol.

Untuk setiap bilangan riil selalu bisa ditentukan bilangan rasional yang sedekat mungkin dengan bilangan riil itu, dengan syarat himpunan bilangan rasional itu padat (*dense*), contohnya

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \qquad (11)$$

yang meskipun $\sin(x)$ riil, tapi bisa didekati dengan bilangan rasional sampai ketelitian tertentu.

Jika suatu fungsi yang memiliki beberapa masukan, maka kita hanya dapat memperoleh satu keluaran (*output*). Dalam komputasi, keluaran ini tersimpan dalam suatu variabel yang namanya adalah nama fungsi. Contohnya seperti berikut:

plus (*a*,*b*) → masukan: *a* dan *b*
keluaran: *a* + *b*
(tersimpan dalam variabel yang namanya "plus")

Jika *a* + *b* + *c* = (*a* + *b*) + *c*, maka:

d = *plus* (*a* + *b*) (12)
e = *plus* (*d* + *c*) (13)

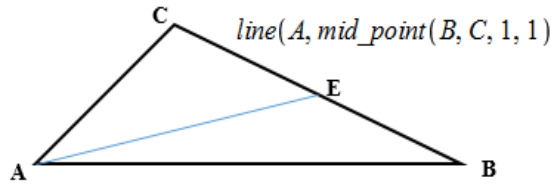
Fungsi *d* = plus (*a* + *b*) tidak diperlukan, karena plus (*a*, *b*) sendiri sudah menyimpan hasil, sementara *e* = plus (*d* + *c*) akan mencetak *e*. Fungsi tersebut akhirnya dapat disimpifikasi dengan *print plus*(*plus*(*a*,*b*),*c*).

Komputasi simbolik dan numerik

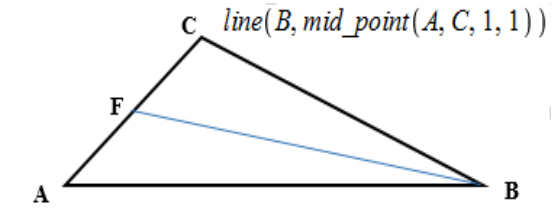
Komputasi numerik bersifat otomatis. Program yang diberi masukan berupa bilangan riil dalam bentuk desimal (seperti $\frac{1}{3} = 0{,}333$), hanya akan menyatakan bilangan riil melalui satu cara. Setelah program dimulai (*run*), maka akan keluar bilangan riil dalam bentuk desimal (otomatis), sehingga kita tidak perlu mengintervensi jalannya program. Untuk program komputasi *Fortran*, program yang sudah jadi dari bahasa teks (*file text*) akan dikompilasi menjadi bahasa mesin (*file exe*).

Dengan menggunakan komputasi, permasalahan titik berat segitiga di atas dapat diselesaikan dengan penyelesaian berikut; pertama kita mengambil titik tengah dari garis

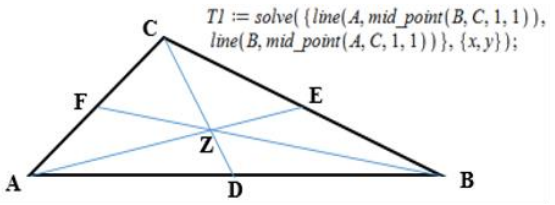
BC, kemudian diberi garis menuju sudut A. Langkah yang sama juga dilakukan dari titik tengah garis **AC** menuju sudut B.



Gambar 3. Gambar dan *script* garis tengah sisi segitiga **BC.**



Gambar 4. Gambar dan *script* garis tengah sisi segitiga **AC**



Gambar 5. Gambar dan *script* perpotongan garis **AE**, **CD**, dan **BF**

Dengan demikian, untuk menghitung titik berat segitiga di atas, kita dapat menggunakan *script* berikut:

```
> line := (A, B) → if true then
    y − A[2] = ((A[2] − B[2])/(A[1] − B[1])) · (x − A[1])
  fi:
> mid_point := (A, B, mA, mB) → if true then
    xC := (mB·A[1] + mA·B[1])/(mA + mB):
    yC := (mB·A[2] + mA·B[2])/(mA + mB):
    [xC, yC]
  fi:
Warning, `xC` is implicitly declared local
to procedure `mid_point`
Warning, `yC` is implicitly declared local
to procedure `mid_point`
> A := [1, 1]: B := [4, 5]: C := [3, 15]:
> T1 := solve({line(A, mid_point(B, C, 1, 1)),
  line(B, mid_point(A, C, 1, 1))}, {x, y});
                T1 := {x = 8/3, y = 7}                (1)
```

Titik perpotongan dari kedua garis ini merupakan titik berat benda sehingga untuk sebarang koordinat yang kita berikan pada masing-masing sudut segitiga, kita dapat menghitung koordinat titik berat benda.

Titik perpotongan dari kedua garis ini merupakan titik berat benda sehingga untuk sebarang koordinat yang kita berikan pada masing-masing sudut segitiga, kita dapat menghitung koordinat titik berat benda.

5. KESIMPULAN

Semua objek matematik dapat direpresentasikan sebagai objek simbolik melalui pemodelan dalam bentuk ekspresi matematik. Dalam komputasi, kita perlu untuk melakukan simplifikasi ekspresi matematik. Komputasi simbolik sendiri memproses bilangan dan huruf yang membentuk ekspresi matematik. Bahasa pemrograman simbolik dapat mengubah bilangan rasional, sehingga komputasi simbolik ini dapat diaplikasikan untuk pembuktian teorema geometri secara induktif, yaitu pembuktian sebuah teorema melalui peninjauan kasus titik berat pada segitiga.

6. REFERENSI

Buchberger, B. (1985) *Symbolyc computation (An Editorial)*, 1, pp. 1-6.

Buchberger, B., Collins, G. E., Loos Rudiger, Albrecht Rudulf (1983). *Computer Algebra*. Computing Suplementa

Davenport, J. H., Siret, Y., Tournier, E., (1985). *Computer Algebra: Systems and Algorithms for Algebraic Computation*. Davenport: Academic Press.

Gathen von zur, J., Gerhard, J. (2003). *Modern Computer Algebra (Second Edition)*. Cambridge University Press.

Geddes, K. O., Czapor, S. R., Labahn, G. (1992) *Algorithms for Computer Algebra*.

Kerr, G., González-Parra, G., Sherman, M. (2022). A new method based on the Laplace transform and Fourier series for solving linear neutral delay differential equations. *Applied Mathematics and Computation*, 420, 126914.

Lample, G., Charton, F. (2019). Deep Learning for Symbolic Mathematics. *International Conference on Learning Representations*.