計算量の解析と速度比較

Maple の数式処理機能は、数式に対する記号的な解析演算を適用するだけでなく、計算やシミュレーションに用いる数式の計算量（計算コスト）を求めるためにも適用することが可能です。
この資料では、Maple の計算量解析のための基本機能と、計算量を最適にした計算プログラムの C 言語への変換を通じた実際の速度の比較について解説しています。

```
> restart
```

計算コストの算出

多変数かつ中規模～大規模の数式モデルによるシミュレーションを行うような場合、得てしてその計算コスト（四則演算、関数評価等の回数）について考慮する必要が生じます。

Maple では、与えられた数式の四則演算・関数評価・変数割り当て等の回数を自動でカウントするためのコマンドが用意されています。ここでは、その利用方法について解説します。

いま、ランダムに多項式 P を用意します；

```
> P := randpoly([x, y], degree = 5, dense)
```

$$P := -10 - 7x^5 + 22x^4y - 55x^4 - 94x^3y^2 + 87x^3y - 56x^3 - 62x^2y^2 + 97x^2y - 73x^2 - 4xy^4 - 83xy^3 - 10xy^2 + 62xy - 82x + 80y^5 - 44y^4 + 71y^3 - 17y^2 - 75y \quad \textbf{(1.1)}$$

多項式 P の計算コストとは、その乗算・加算の回数です。codegen パッケージで用意されている cost コマンドを用いると、その乗算・加算の回数を結果として返します。

```
> with(codegen, cost):
> cost(P)
```

$$19\ additions + 65\ multiplications \quad \textbf{(1.2)}$$

ここで、additions は加算、multiplications は乗算を意味します。つまり、この多項式 P については加算：19回、乗算：65 回の計算コストが発生しています。

多項式 P をホーナー則を用いて式の形を変えてみます。convert コマンドを用います。

```
> P2 := convert(P, horner)
```

$$P2 := -10 + (-75 + (-17 + (71 + (-44 + 80y)y)y)y)y + (-82 + (62 + (-10 + (-83 - 4y)y)y)y + (-73 + (97 - 62y)y + (-56 + (87 - 94y)y + (-55 + 22y - 7x)x)x)x)x \quad \textbf{(1.3)}$$

P2 に対して cost コマンドを適用してみましょう。

```
> cost(P2)
```

$$19\ additions + 19\ multiplications \quad \textbf{(1.4)}$$

結果として、乗算の回数を減らすことができています。このように、cost コマンドや関連する Maple のコマンドを用いることで計算回数の少ない効率的な式表現を得ることが可能です。

最適コストとなる計算式への変形

前章では、Maple での数式の計算コストの算出方法について紹介しました。ここでは、Maple でどのようにして最適な計算コストの数式表現を得るかについて解説します。

```
> restart
```

式の形式的な変形

最適な計算コストは、数式の見た目（四則演算、括弧の有無、計算順序、他の変数への置換え有無）などによっても変わってきます。ここでは、三角関数を含む多項式 Q1 を用意し、その見た目を複数用意して計算コストの比較を行います。

まず、元となる式 Q1 を用意します。

```
> Q1 := randpoly([sin(x), cos(y)], degree = 4, dense)
```

$$Q1 := -82 - 7\sin(x)^4 + 22\sin(x)^3\cos(y) - 55\sin(x)^3 - 94\sin(x)^2\cos(y)^2 + 87\sin(x)^2\cos(y) - 56\sin(x)^2 - 62\sin(x)\cos(y)^2 + 97\sin(x)\cos(y) - 73\sin(x) - 4\cos(y)^4 - 83\cos(y)^3 - 10\cos(y)^2 + 62\cos(y) \quad \textbf{(2.1.1)}$$

Q1 に対して、式を結合したものを Q2 として用意します。

```
> Q2 := combine(Q1)
```

$$Q2 := -\frac{581}{4}\sin(x) + \frac{173}{4}\cos(y) - \frac{7}{8}\cos(4x) + 55\cos(2x) + \frac{55}{4}\sin(3x) - \frac{11}{4}\sin(y+3x) - \frac{11}{4}\sin(-y+3x) + \frac{227}{4}\sin(x+y) + \frac{227}{4}\sin(x-y) - \frac{61}{2}\cos(2y) + \frac{47}{4}\cos(2x-2y) + \frac{47}{4}\cos(2x+2y) - \frac{87}{4}\cos(-y+2x) - \frac{87}{4}\cos(y+2x) - \frac{31}{2}\sin(x+2y) - \frac{31}{2}\sin(x-2y) - \frac{1}{2}\cos(4y) - \frac{83}{4}\cos(3y) - \frac{1141}{8} \quad \textbf{(2.1.2)}$$

同じく、simplify コマンドで式のサイズ（長さ）を短くしたものを Q3 とします。式のサイズについて簡単化するには size オプションを指定します。

```
> Q3 := simplify(Q1, size)
```

$$Q3 := -4\cos(y)^4 - 83\cos(y)^3 + (-62\sin(x) - 94\sin(x)^2 - 10)\cos(y)^2 + (97\sin(x) + 22\sin(x)^3 + 87\sin(x)^2 + 62)\cos(y) - 82 - 7\sin(x)^4 - 56\sin(x)^2 - 55\sin(x)^3 - 73\sin(x) \quad \textbf{(2.1.3)}$$

次に、ここで用意した各 Q1, Q2, Q3 に対して codegen パッケージの optimize コマンドを適用し、その計算コストを比較します。

式の最適化と計算コスト比較

codegen パッケージには、与えられた数式に対して計算量が少ない表現への変換を求めるための optimize コマンドが用意されています。ただし、この optimize コマンドは、与えられた式の見た目の表現に基づいて最適な計算コストとなる表現を返します。つまり、入力として与える数式がどのような表現になっているか否かで、その計算コストが変わってきます。前節で求めた Q1, Q2, Q3 の各式表現に対して optimize コマンドを適用し、その計算コストを比較してみます。

まず、パッケージ内のコマンドのみを読み込みます。

```
> with(codegen, optimize):
```

Q1 に対して optimize コマンドを適用した結果を Q1opt として変数に割り当てます。

optimizeにより、最終的な式の計算を行う前に部分的な計算を別の変数へと置き換えています。（t + 数値の変数が置き換えて）

```
> Q1opt := optimize(Q1)
```

$$Q1opt := t1 = \sin(x),\ t2 = t1^2,\ t3 = t2^2,\ t5 = t2\ t1,\ t6 = \cos(y),\ t10 = t6^2,\ t21 = t10^2,\ t27 = -82 - 7\ t3 + 22\ t5\ t6 - 55\ t5 - 94\ t2\ t10 + 87\ t2\ t6 - 56\ t2 - 62\ t1\ t10 + 97\ t1\ t6 - 73\ t1 - 4\ t21 - 83\ t10\ t6 - 10\ t10 + 62\ t6 \quad \textbf{(2.2.1)}$$

同様に、Q2, Q3 に対して optimize コマンドを適用し、結果をそれぞれ Q2opt, Q3opt として変数に割り当てます。

```
> Q2opt := optimize(Q2)
```

$$Q2opt := t1 = \sin(x),\ t3 = \cos(y),\ t6 = \cos(4\ x),\ t8 = 2\ x,\ t9 = \cos(t8),\ t11 = 3\ x,\ t12 = \sin(t11),\ t15 = \sin(y + t11),\ t18 = \sin(-y + t11),\ t20 = x + y,\ t21 = \sin(t20),\ t23 = x - y,\ t24 = \sin(t23),\ t26 = 2\ y,\ t27 = \cos(t26),\ t29 = \cos(2\ t23),\ t31 = \cos(2\ t20),\ t34 = \cos(-y + t8),\ t37 = \cos(y + t8),\ t40 = \sin(x + t26),\ t43 = \sin(x - t26),\ t46 = \cos(4\ y),\ t49 = \cos(3\ y),\ t51 = -\frac{581}{4}\ t1 + \frac{173}{4}\ t3 - \frac{7}{8}\ t6 + 55\ t9 + \frac{55}{4}\ t12 - \frac{11}{4}\ t15 - \frac{11}{4}\ t18 + \frac{227}{4}\ t21 + \frac{227}{4}\ t24 - \frac{61}{2}\ t27 + \frac{47}{4}\ t29 + \frac{47}{4}\ t31 - \frac{87}{4}\ t34 - \frac{87}{4}\ t37 - \frac{31}{2}\ t40 - \frac{31}{2}\ t43 - \frac{1}{2}\ t46 - \frac{83}{4}\ t49 - \frac{1141}{8} \quad \textbf{(2.2.2)}$$

```
> Q3opt := optimize(Q3)
```

$$Q3opt := t1 = \cos(y),\ t2 = t1^2,\ t3 = t2^2,\ t7 = \sin(x),\ t9 = t7^2,\ t14 = t9\ t7,\ t19 = t9^2,\ t24 = -4\ t3 - 83\ t2\ t1 + (-62\ t7 - 94\ t9 - 10)\ t2 + (97\ t7 + 22\ t14 + 87\ t9 + 62)\ t1 - 82 - 7\ t19 - 56\ t9 - 55\ t14 - 73\ t7 \quad \textbf{(2.2.3)}$$

optimizeコマンドには、変数への置換え等も含めて式の計算コスト最適化をより強く適用するための **tryhard** オプションが用意されています。このオプションを用いた結果をそれぞれ Q1hard, Q2hard, Q3hard として求めてみます。

```
> Q1hard := optimize(Q1, tryhard)
```

$$Q1hard := t5 = \cos(y),\ t4 = \sin(x),\ t3 = t5^2,\ t2 = t4^2,\ t1 = t4\ t2,\ t6 = -82 + (22\ t1 + 97\ t4 + 62)\ t5 - 55\ t1 - 73\ t4 + (-56 + 87\ t5 - 7\ t2)\ t2 + (-83\ t5 - 94\ t2 - 62\ t4 - 10 - 4\ t3)\ t3 \quad \textbf{(2.2.4)}$$

```
> Q2hard := optimize(Q2, tryhard)
```

$$Q2hard := t14 = 2\ x,\ t13 = 3\ x,\ t12 = 2\ y,\ t11 = x - y,\ t10 = x + y,\ t1 = -\frac{581}{4}\sin(x) + \frac{173}{4}\cos(y) - \frac{7}{8}\cos(4\ x) + 55\cos(t14) + \frac{55}{4}\sin(t13) - \frac{1141}{8} - \frac{83}{4}\cos(3\ y) - \frac{61}{2}\cos(t12) - \frac{1}{2}\cos(4\ y) + \frac{227}{4}\sin(t10) + \frac{227}{4}\sin(t11) - \frac{87}{4}\cos(-y + t14) - \frac{87}{4}\cos(y + t14) + \frac{47}{4}\cos(2\ t10) + \frac{47}{4}\cos(2\ t11) - \frac{31}{2}\sin(x - 2\ y) - \frac{31}{2}\sin(x + t12) - \frac{11}{4}\sin(-y + t13) - \frac{11}{4}\sin(y + t13) \quad \textbf{(2.2.5)}$$

```
> Q3hard := optimize(Q3, tryhard)
```

$$Q3hard := t18 = \cos(y),\ t21 = t18^2,\ t17 = \sin(x),\ t16 = t17^2,\ t15 = t17\ t16,\ t1 = -82 \quad \textbf{(2.2.6)}$$

$$-73\,t17-55\,t15+(97\,t17+22\,t15+62)\,t18+(87\,t18-56-7\,t16)\,t16$$
$$+(-83\,t18-62\,t17-94\,t16-10-4\,t21)\,t21$$

これまでに得られた Q1, Q2, Q3, Q1opt, Q2opt, Q3opt, Q1hard, Q2hard, Q3hard の各数式の計算コストを求めてみます。

```
> with(codegen, cost):
```

```
> Q1cost := cost(Q1)
```

$$Q1cost := 13\ additions + 18\ functions + 36\ multiplications \quad \textbf{(2.2.7)}$$

```
> Q2cost := cost(Q2)
```

$$Q2cost := 28\ additions + 18\ functions + 34\ multiplications \quad \textbf{(2.2.8)}$$

```
> Q3cost := cost(Q3)
```

$$Q3cost := 13\ additions + 29\ multiplications + 13\ functions \quad \textbf{(2.2.9)}$$

```
> Q1optcost := cost(Q1opt)
```

$$Q1optcost := 2\ functions + 8\ assignments + 24\ multiplications + 13\ additions \quad \textbf{(2.2.10)}$$

```
> Q2optcost := cost(Q2opt)
```

$$Q2optcost := 18\ functions + 24\ assignments + 26\ multiplications + 27\ additions \quad \textbf{(2.2.11)}$$

```
> Q3optcost := cost(Q3opt)
```

$$Q3optcost := 2\ functions + 8\ assignments + 19\ multiplications + 13\ additions \quad \textbf{(2.2.12)}$$

```
> Q1hardcost := cost(Q1hard)
```

$$Q1hardcost := 2\ functions + 6\ assignments + 16\ multiplications + 13\ additions \quad \textbf{(2.2.13)}$$

```
> Q2hardcost := cost(Q2hard)
```

$$Q2hardcost := 27\ multiplications + 6\ assignments + 26\ additions + 18\ functions \quad \textbf{(2.2.14)}$$

```
> Q3hardcost := cost(Q3hard)
```

$$Q3hardcost := 2\ functions + 6\ assignments + 16\ multiplications + 13\ additions \quad \textbf{(2.2.15)}$$

得られた計算コストに対して、加算、乗算、関数評価、計算結果の変数への保存、変数への割り当てをそれぞれ重み付けし、量的に比較してみます。いま、各演算の重みを以下の変数 w で割り当てられている数量とします。

```
> w := [additions = 1, multiplications = 20, functions = 100, storage = 2, assignments
      = 1]:
```

for 文で各計算量を求めます。

```
> for c in [Q1cost, Q2cost, Q3cost, Q1optcost, Q2optcost, Q3optcost, Q1hardcost,
      Q2hardcost, Q3hardcost] do
   eval(c, w)
   end do
```

$$2533$$
$$2508$$
$$1893$$
$$701$$
$$2371$$
$$601$$
$$539$$
$$2372$$
$$539 \quad \textbf{(2.2.16)}$$

以上の結果から、計算コストが最適（最小）なのは、Q1hard と Q3hard であることがわかります。最適となる計算式の表現を確認してみます。

```
> Q1hard
```

$$t5=\cos(y),\ t4=\sin(x),\ t3=t5^2,\ t2=t4^2,\ t1=t4\,t2,\ t6=-82+(22\,t1+97\,t4 \quad \textbf{(2.2.17)}$$
$$+62)\,t5-55\,t1-73\,t4+(-56+87\,t5-7\,t2)\,t2+(-83\,t5-94\,t2-62\,t4$$
$$-10-4\,t3)\,t3$$

```
> Q3hard
```

$$t18 = \cos(y),\ t21 = t18^2,\ t17 = \sin(x),\ t16 = t17^2,\ t15 = t17\,t16,\ t1 = -82 - 73\,t17 - 55\,t15 + (97\,t17 + 22\,t15 + 62)\,t18 + (87\,t18 - 56 - 7\,t16)\,t16 + (-83\,t18 - 62\,t17 - 94\,t16 - 10 - 4\,t21)\,t21 \qquad \textbf{(2.2.18)}$$

若干の違いはありますが、変数への置換や計算結果の変数への保存などは非常に似ています。

Q3 は、元々 Q1 を長さに関して簡単化（simplify コマンドを適用）したものです。すなわち、今回の例においては、simplify コマンドを size オプションで使用し長さを短くなるようにした式をさらに最適化した表現式が、もっとも計算量上で最小となる式であることが判明しました。

ただし、数式によっては必ずしも simplify コマンド(size オプション付き)を適用したものが計算量最小であるとは限りません。この点では、その分野における式変形のルールを上手に活用することが必要となります。

C言語への変換と速度比較

計算量を最適にした数式は、Maple 上での計算速度もさることながら、C 言語や他の言語のソースにおいても計算量最適化の改善効果は非常に重要です。ここでは、前節で求めた計算量の異なる２つの数式 Q1 及び Q1 を強制的に最適化した Q3hard を用いて、それぞれを Compiler パッケージにより外部実行ライブラリ化し計算速度を比較してみます。

まず、Q1 及び Q3hard をそれぞれ Maple のプロシージャに変換します。これには codegen パッケージの makeproc コマンドを用います。

```
> with(codegen, makeproc):
```

Q1 を、引数 x, y を取るプロシージャ Q1func とします。

```
> Q1func := makeproc(Q1, [x, y])
```

```
Q1func := proc(x, y)
    -82 - 7*sin(x)^4 + 22*sin(x)^3*cos(y) - 55*sin(x)^3 - 94*sin(x)^2
    *cos(y)^2 + 87*sin(x)^2*cos(y) - 56*sin(x)^2 - 62*sin(x)*cos(y)^2 + 97
    *sin(x)*cos(y) - 73*sin(x) - 4*cos(y)^4 - 83*cos(y)^3 - 10*cos(y)^2
    + 62*cos(y)
end proc
```
(3.1)

同様に、Q3hard についてもプロシージャ化します。

```
> Q3hardfunc := makeproc([Q3hard], [x, y])
```

```
Q3hardfunc := proc(x, y)
    local t1, t15, t16, t17, t18, t21;
    t18 := cos(y);
    t21 := t18^2;
    t17 := sin(x);
    t16 := t17^2;
    t15 := t17*t16;
    t1 := -82 - 73*t17 - 55*t15 + (97*t17 + 22*t15 + 62)*t18 + (87
    *t18 - 56 - 7*t16)*t16 + (-83*t18 - 62*t17 - 94*t16 - 10 - 4*t21)
    *t21
end proc
```
(3.2)

プロシージャに変換した２つの数式を Compiler パッケージを用いて外部ライブラリ化します。(Compiler パッケージは、Maple 言語で記述されたプロシージャを C 言語ソースへと変換し、かつコンパイル、ビルド及び Maple とのリンクを自動で行います)

```
> with(Compiler)
```

```
Warning, Compiler is not a correctly formed package - option `package`
is missing
```
[BuildTools, Compile, NativeProcs, Setup]
(3.3)

```
> Q1funcCompiled := Compile(Q1func)
```

```
Q1funcCompiled := proc( )
    option call_external, define_external(_mf258b4e3a7b648c602447ae2f9d37d0a,
    MAPLE, LIB
     = "C:\DOCUME~1\tetsuy\LOCALS~1\Temp\tetsuy-2840\_mf258b4e3a7b648c6024\
    47ae2f9d37d0at1kgKiOO.dll");
    call_external(0, 77206336, true, false, args)
end proc
```
(3.4)

```
> Q3funcCompiled := Compile(Q3hardfunc)
```

(3.5)

```
Q3funcCompiled := proc( )                                                    (3.5)
    option call_external, define_external( _mb0ed71303da04923901f1f19afaae243,
    MAPLE, LIB
     = "C:\DOCUME~1\tetsuy\LOCALS~1\Temp\tetsuy-2840\_mb0ed71303da04923901\
    f1f19afaae243RkUkxVYj.dll");
    call_external(0, 77336816, true, false, args)
end proc
```

外部ライブラリにした２つの数式を用いて、計算速度を求めてみます。計算（コール）回数は 100 万回です。

```
> st := time( ):
  data1 := seq(seq(Q1funcCompiled(x, y), x = 0 ..1.0, 0.001), y = 0 ..1.0, 0.001):
  time( ) - st
                                  14.812                                     (3.6)
> st := time( ):
  data2 := seq(seq(Q3funcCompiled(x, y), x = 0 ..1.0, 0.001), y = 0 ..1.0, 0.001):
  time( ) - st
                                  10.704                                     (3.7)
```

Q3hard を用いた計算が高速であることがわかります。

なお、念のため計算結果を確認します。data1 と data2 を引き算し、その最大値を確認します。(結果は 0 となっており、同じであることがわかります)

```
> err := [data1] - [data2]:
> max(err)
                                    0.                                       (3.8)
```