Maple 14 における効率性の向上

Maple 14 には効率を向上する数多くの改良が含まれています。処理速度が飛躍的に向上した箇所は以下の通りです。

Intel (R) MKL 10.0 (Windows)

- 32- ビット版 Windows 上で動作していた Intel ® Math Kernel Library (Intel ® MKL) の旧バージョンと 64- ビット版 Windows 上で動作していた汎用 BLAS が Intel ® MKL バージョン 10 に置き換えられました。Intel® MKL 10.0 の、高度に最適化されたコアルーチンは Maple のさまざまな箇所で使用されています。
- 次は、Maple 14 ではどれだけパフォーマンスが向上されているかを強調する例です。

```
> M:=LinearAlgebra:-RandomMatrix(2000,datatype=float[8]):
> time( M.M );
```

4.674 (1.1)

2.13 GHz の Core2 Duo 上で 32- ビット版の Windows を動作させている場合、Maple 13.02 だと 7.95 秒必要なのに比べて、Maple 14 だと 2.44 秒で済みます。
2.00 GHz のデュアル Quad-Core Xeon 上で 64- ビット版の Windows を動作させている場合、Maple 13.02 だと 25.593 秒必要なのに比べて、Maple 14 だと 2.437 秒で済みます。

円分多項式

- numtheory[cyclotomic] は新しい実装により、大きな素因数に対する円分多項式の構築にかかる時間が著しく短縮されています。
- 比較のために、次の例を検討します：

```
> t:= time():
> numtheory[cyclotomic](255255,x):
> time() - t;
```

0.228 (2.1)

これは Maple 13 より約 1000 倍速く処理されたことを示しています。問題の規模が大きいほど処理速度が上がります。

多項式の演算

- 改良された実装により、整数体上での多項式の乗算や除算が Maple 13 のときより速く処理できるようになりました。大きな多項式の乗算の場合には、マルチスレッドを使用することも可能になっています。
- 次の例では、計算は Maple 13 のときより約 20 倍速くなっています：

```
> f,g := seq(randpoly([x,y,z],dense,degree=30),i=1..2):
> t:= time():
> assign(p=expand(f*g)):
> divide(p,f,'q');
```

true (3.1)

```
> time() - t;
```

1.600 (3.2)

- 因数分解などの上位多項式演算は、上記改良の恩恵を受けます。次の例では、計算は Maple 13 のときより約 10 倍速くなっています：

```
> f := expand((1+x+y+z+t)^11)+1:
> p:=expand(f*(f+1)):
> time(factor(p));
```

2.828 (3.3)

- Maple 14 では、代数体上で多項式を計算するときに使用される表現である再帰的な稠密多項式の乗算、除算、反転および gcd (最大公約数) 向けのアルゴリズムに改良が加えられています。
- 新しいアルゴリズムでは、メモリが先に割り当てられ、その場で実行されるため、繰り返し実行が必要な場合は処理が著しく速くなります。
- 次の例では、計算は Maple 13 のときより約 2 倍速くなっています：

```
> p := modp1(Prime(1)):
> alpha := RootOf(Randprime(2,x) mod p):
> beta  := RootOf(Randprime(5,x,alpha) mod p):
> cofs  := ()->randpoly([alpha,beta],degree=4,dense):
> f,g,h := seq(Expand(randpoly(x,degree=40,dense,coeffs=cofs)) mod
  p,i=1..3):
> time(assign('a'=Expand(f*g) mod p));
                                0.396                          (3.4)
> time(assign('b'=Expand(g*h) mod p));
                                0.316                          (3.5)
> time(assign('g'=Gcd(a,b) mod p));
                                0.824                          (3.6)
```

has、sort、degree、type による多項式情報の取り出し

- 代数的数または関数が含まれているときの、コマンド has、sort、degree および type の処理速度がかなり上がっています。新しい実装により、これらのコマンドが必要とする計算時間が短縮されました。
- 次の例は、F に関するコマンドについて、Maple 14 の実装は Maple 13 のより 3 秒速いことを示しています。:

```
> c := rand(0..995):
> r := RootOf(numtheory[cyclotomic](997,Z),Z):
> F := [seq(add(c()*Z^c()*x^i, i=0..10^4-1), j=1..10)]:
> G := subs(Z=r, F):
> time(map(degree, eval(F,1), x));
                                0.012                          (4.1)
> time(map(degree, eval(G,1), x));
                                0.012                          (4.2)
> time(map(type, eval(F,1), polynom(anything,x)));
                                0.004                          (4.3)
> time(map(type, eval(G,1), polynom(anything,x)));
                                0.008                          (4.4)
```

より高速な GraphTheory アルゴリズム

- 新しいアルゴリズムの採用により、Maple 14 における BipartiteMatching、ConnectedComponents および MaxFlow の時間効率およびメモリ効率が向上されました。
- 次の例の計算については、Maple 14 では Maple 13 より約 200 倍速く、メモリの使用量は約 150 MB 少なく済んでいます。

```
> with(GraphTheory):
> with(SpecialGraphs):
> bt:=CompleteBinaryTree(8):
> time(BipartiteMatching(bt));
                                0.048                          (5.1)
```

• 次の例の計算については、Maple 14 では Maple 13 より約 9 秒早く終了し、メモリの使用量は約 250 MB 少なく済んでいます。

```
> with(GraphTheory):
> cg:=CycleGraph(4000):
> cgc:=GraphComplement(cg):
> time(ConnectedComponents(cgc));
```

0.112 (5.2)

• 次の例については、Maple 14 では Maple 13 より 9 倍速くなっています。

```
> with(GraphTheory):
> with(SpecialGraphs):
> sb:=SoccerBallGraph():
> sbw:=MakeWeighted(sb):
> time(MaxFlow(sbw,1,20));
```

0.020 (5.3)

• さらに、CompleteGraph、DisjointUnion および AddVertex を含む基本コマンド GraphTheory は、多くの頂点と辺を持つ大きなグラフをより効率的に処理できるように改良されています。

CUDA 加速

• Maple では、NVIDIA の CUDA 技術を使用して LinearAlgebra ルーチンの加速化をサポートしています。さまざまなデータ型および形の行列で乗算を加速することができます。詳細については、CUDA パッケージおよび CUDA,supported_routines を参照してください。注意：次の例では、処理が実行されるコンピュータで CUDA がサポートされている場合に限って処理速度の向上が見られます。詳細については、CUDA,supported_hardware を参照してください。

```
> N := 20:
> m1 := LinearAlgebra:-RandomMatrix( 2000,2000,outputoptions=
  [datatype=float[4]] ):
> m2 := LinearAlgebra:-RandomMatrix( 2000,2000,outputoptions=
  [datatype=float[4]] ):
> t := time[real]():
> to N do mNoCuda := m1.m2: end:
> tNoCuda := time[real]()-t;
```

tNoCuda := 24.942 (6.1)

```
> CUDA:-Enable( true );
> t := time[real]():
> to N do mCuda := m1.m2: end:
> tCuda := time[real]()-t;
```

tCuda := 2.202 (6.2)

.(ドット)演算子の改良

Maple 14 では、.(ドット)演算子を呼び出す際のオーバーヘッドを削減しています。これは特に小さな行列を扱うときに重要です。

• Maple 13 では、次のコマンドが終了するまでに 9 秒強かかっていました。

```
> A := Matrix([[9,5],[6,4]]):
> B := Matrix([[6,7],[3,10]]):
> n := 100:
> st := time():
> to n do to n do A.B; od; od:
> t1 := time()-st;
```

(7.1)

$t1 := 0.072$ (7.1)

ソルバの直接利用

- 新規コマンド SolveTools[Polynomial] を使用すると、1 変数多項式ソルバを直接利用することができ、SolveTools[PolynomialSystem] を使用すると、多変数多項式ソルバを直接利用することができます。純粋な多項式に関しては、多くの前処理を省略し、SolveTools[Linear] 並みのオーバーヘッドしか送らないこれらのコマンドは solve より効率的です。詳細については、Enhancements to Symbolic Capabilities in Maple 14 を参照してください。

関連項目

Index of New Maple 14 Features