Maple 13 における微分方程式 (DE) ソルバの改良点

要約

- Maple 13 における厳密でシンボリックな常微分方程式 (ODE) の求解のテーマは、標準的手法の範囲を超えていたり、標準的手法では過度に複雑な結果が導かれるような微分方程式 (DE) において、これに代わる求解方法を探索することです。たとえば、対称点を持たない場合を含めた主要な非線形 ODE 族の場合、構築し、次に関連する高階の線形 ODE を解くことで解を導きます。同様に、標準以外の手法で対称性を探索することにより、さまざまな ODE の初期値問題は、ODE の一般解が計算できない場合を含め、単純な形式で表現できます。本リリースでは、これらの機能が標準 DE ソルバ (dsolve) に組み込まれており、その手法も各種の新ツールを通じて単独利用できるようになりました。
- 偏微分方程式については、PDEtools の大部分の対称性コマンドの機能があらゆる観点から大きく改良されました。また、新たに 4 種類のコマンドが追加されています。さまざまな新機能のなかでも特筆すべきは、線形および非線形 PDE 系の多項式の解の高速計算を実現するコマンドが 1 つ追加されたことです。さらに、内部 PDEtools ライブラリにはプログラミング専用の新たなルーチンが 45 種類追加されました。
- ODE の数値解については、ODE 問題と添え字が 1 つの DAE 問題で、イベント処理、および精度とエラー制御の機能が改良されました。

厳密解

常微分方程式 (ODE)

本リリースの dsolve コマンドは、以前は適用外であったさまざまな問題を解くことができるようになり、また、これまでの解法では複雑すぎた多様な ODE 族の解を以前よりはるかに簡潔に計算できるようになりました。

1 階から 5 階までの非線形 ODE の新たな求解: 微分法を用いた線形化

- ときには、標準的な方法では解くことができない非線形 ODE でも、適切な因子を用いて乗算し、(1 回か 2 回の) 微分を行うことができることがあり、その場合の結果式には、より高階の線形 ODE が因数として含まれます。この線形 ODE を解くと、オリジナルの問題に対する明示的な解をよりコンパクトな方法で表すことができます。

例

```
> PDEtools:-declare(y(x), prime=x);
```

$$y(x) \text{ will now be displayed as } y$$

$$\text{derivatives with respect to } x \text{ of functions of one variable will now be displayed with } ' \qquad (2.1.1.1)$$

次に示すのは、y' に対する 1 階の非線形の ODE で、2 つの引数の任意関数を G とし、これに対応する新しい解を導く例です。

```
> ode[1] := G((2*y(x)-2+diff(y(x),x)*x)*x^3/(3*x^4+2*x^3),-
  x^3*(-y(x)+1+(x+1)*diff(y(x),x))/(3*x+2)) = 0;
```

$$ode_1 := G\left(\frac{(2y-2+y'x)\,x^3}{3x^4+2x^3}, -\frac{x^3\,(-y+1+(x+1)\,y')}{3x+2}\right) = 0 \qquad (2.1.1.2)$$

```
> sol[1] := dsolve(ode[1], implicit);
```

(2.1.1.3)

$$sol_1 := G\left(\frac{-y + \frac{-C1}{x^2} + 1}{x+1}, \_C1\right) = 0 \qquad (2.1.1.3)$$

ここで、ode[1]における任意形式の G では、上記の出力における同じ G によって解が導かれます。通常、ODE の解は odetest を用いて検証されます。

```
> odetest(sol[1], ode[1]);
```

$$0 \qquad (2.1.1.4)$$

次に、アプリケーションに関する科学出版物から抽出した 3 階と 4 階の 2 種類の非線形 ODE の例を取り上げます。この新しい解法では、微分法の線形化を用いた高速でコンパクトな手法で計算しています。

```
> ode[2] := diff(y(x),x,x,x) = 1/8*(-8*x*diff(y(x),x,x)*diff
  (y(x),x) + 4*x^2*diff(y(x),x,x)^2 + diff(y(x),x)^2 + 12*y
  (x))/x^2/diff(y(x),x);
```

$$ode_2 := y''' = \frac{1}{8}\,\frac{-8\,x\,y''y' + 4\,x^2\,y'^2 + y^2 + 12\,y}{x^2\,y'} \qquad (2.1.1.5)$$

```
> dsolve(ode[2]);
```

$$y = -4\,\_C2\,\sqrt{x} + \frac{4}{3}\,x^{3/2}\,\_C1 + x^2 + \_C3\,x - \frac{4}{3}\,\_C2\,\_C1 - \frac{1}{12}\,\_C3^2 \qquad (2.1.1.6)$$

```
> ode[3] := diff(y(x),x,x,x,x) = 1/2*(2*diff(y(x),x,x,x)*diff
  (y(x),x)-diff(y(x),x,x)^2+c^4*y(x)^2)/y(x);
```

$$ode_3 := y'''' = \frac{1}{2}\,\frac{2\,y'''y' - y''^2 + c^4\,y^2}{y} \qquad (2.1.1.7)$$

```
> dsolve(ode[3]);
```

$$y = \_C1 + \frac{1}{8}\,\frac{(\_C1^2 - 2\,\_C3^2 - 2\,\_C4^2)\,e^{-cx}}{\_C2} + \_C2\,e^{cx} + \_C3\,\sin(c\,x) + \_C4\,\cos(c\,x) \qquad (2.1.1.8)$$

ODE 対称性による新しい ODE-IVP 求解

- さまざまな非線形 ODE の初期値問題 (ODE-IVP) では、有理解が存在します。ただし、一般的な ODE の求解では、積分定数を初期条件に一致させる適合において楕円関数つまり特殊関数が関与することから、簡潔な形式での計算が困難です。Maple 13 では、 dsolve によって ODE-IVP の解を計算するための ODE 対称性が自動的に検索されますが、この検索は自身が持つ、より簡潔な有理形式によって直接的に行われます。一般解が完全に範囲外にある場合でも同様です。

例

```
> ode[4] := diff(y(x), x, x) + 3*(diff(y(x), x))/x - 8*y(x)
  ^3/x^6 = 0;
```

$$ode_4 := y'' + \frac{3\,y'}{x} - \frac{8\,y^3}{x^6} = 0 \qquad (2.1.2.1)$$

```
> ic[4] := y(1) = -1/2, D(y)(1) = -1/2;
```

(2.1.2.2)

$$ic_4 := y(1) = -\frac{1}{2}, D(y)(1) = -\frac{1}{2} \tag{2.1.2.2}$$

ode[4] の一般的な求解では楕円積分の逆計算を行いますが、これは楕円関数 JacobiSN で次のように表されます。

```
> dsolve(ode[4]);
```

$$y = \_C2\ \mathrm{JacobiSN}\left(\left(-\frac{I}{x^2} + \_C1\right)\_C2, I\right) \tag{2.1.2.3}$$

積分定数 *C1*, *C2* の適合は容易ではないため、先行の解で条件 y(1) = -1/2, D(y)(1) = -1/2.が満たされます。

ただし、次のように ODE の対称性を探索することで、

```
> DEtools[symgen](ode[4]);
```

$$[\_\xi = x^3, \_\eta = 0], \left[\_\xi = \frac{1}{2}x, \_\eta = y\right] \tag{2.1.2.4}$$

dsolve は、次の単純な解を ODE 全体の初期値問題で直接計算することができます。

```
> sol_ic[4] := dsolve([ode[4], ic[4]]);
```

$$sol\_ic_4 := y = -\frac{x^2}{x^2+1} \tag{2.1.2.5}$$

odetest を用いると、初期条件に対して解をテストすることができます。

```
> odetest(sol_ic[4], [ode[4], ic[4]]);
```

$$[0, 0, 0] \tag{2.1.2.6}$$

```
> ode[5] := diff(y(x), x,x,x) - diff(y(x), x,x)*y(x) + diff(y(x),x)^2 = 0;
```

$$ode_5 := y''' - y''y + y^2 = 0 \tag{2.1.2.7}$$

```
> ic[5] := y(0) = 1, D(y)(0) = 1/6;
```

$$ic_5 := y(0) = 1, D(y)(0) = \frac{1}{6} \tag{2.1.2.8}$$

dsolve は、唯一 ode[5] では階数の簡約のみ取得します。この場合、解が未知の Abel タイプの ODE、 _g(_f)が含まれます。

```
> dsolve(ode[5]);
```

$$y = \left(e^{\int \_g(\_f)\, d\_f + \_C2}\right) \text{ &where } \left[\left\{\_g_{\_f} = (6\,\_f - 1)\,\_g(\_f)^3 + \frac{(7\,\_f - 1)\,\_g(\_f)^2}{\_f} + \frac{\_g(\_f)}{\_f}\right\}, \left\{\_f = \frac{y'}{y^2}, \_g(\_f) = -\frac{y^2}{2y' - \frac{y''y}{y'}}\right\}, \left\{x = \int \frac{\_g(\_f)}{\_f e^{\int \_g(\_f)\, d\_f + \_C2}}\, d\_f + \_C1, y\right.\right. \tag{2.1.2.9}$$

$$= e^{\int \_g(\_f)\, d\_f + \_C2}\Big\}\Big]$$

この結果は、現時点でただ一つの実現可能な結果ですが、条件 y(0) = 1, D(y)(0) = 1/6との一致を図る有効な適合方法はありません。ただし、Maple 13 では dsolve が最初に ode[5] の対称性を検知し、一般解を計算して失敗に終わるのではなく、全体としてこの問題を解くために、この問題の対称性を探索します。

```
> sol_ic[5] := dsolve([ode[5], ic[5]]);
```

$$sol\_ic_5 := y = -\frac{6}{x-6} \qquad (2.1.2.10)$$

```
> odetest(sol_ic[5], [ode[5], ic[5]]);
```

$$[0, 0, 0] \qquad (2.1.2.11)$$

非有理係数を用いた線形 ODE の新しい求解

- この問題における dsolve の既存ルーチンでは、複雑な問題を簡単な形式で解くために最適化するか、シンプルな特殊関数を用いていました。

例

```
> ode[6] := diff(y(x),x,x) + tan(x)*diff(y(x),x) + a^2*cos(x)
  ^2*sin(x)^(2*n-2)*y(x) = 0;
```

$$ode_6 := y'' + \tan(x)\, y' + a^2 \cos(x)^2 \sin(x)^{2n-2} y = 0 \qquad (2.1.3.1)$$

ode[6]における Maple 12 の求解では、 hypergeometric 関数を使用します。Maple 13 では、この求解にはよりシンプルな Bessel 関数を使用します。

```
> dsolve(ode[6]);
```

$$y = \_C1 \sqrt{\sin(x)}\ \mathrm{BesselJ}\left(\frac{1}{2n}, \frac{\sin(x)^n a}{n}\right) \qquad (2.1.3.2)$$

$$+ \_C2 \sqrt{\sin(x)}\ \mathrm{BesselY}\left(\frac{1}{2n}, \frac{\sin(x)^n a}{n}\right)$$

動的対称性を探索する非線形 ODE の新しい求解

- Maple で dsolve によって動的対称性を探索する手法を導入したのは 1997 年です。しかし、関連の求解手法は代数的に複雑です。Maple 13 では関連のルーチンの機能が大幅に改良され、これらの対称性の下位族の特性を利用してこのような問題に対処できるようになりました (自己不変接触対称性など)。この方法により、新規の解、または明らかに簡潔な解を計算できるようになりました。

例

```
> ode[7] := diff(y(x), x,x) = diff(y(x), x)+(ln(diff(y(x), x)
  )-x)*diff(y(x), x)^2/(y(x)-diff(y(x), x));
```

$$ode_7 := y'' = y' + \frac{(\ln(y') - x)\, y'^2}{y - y'} \qquad (2.1.4.1)$$

この ODE は、1 点のみ対称性を承認していますが、これに加えて、
y'に関する計算可能な自己不変接触 (動的) 対称性を持ちます。

```
> DEtools[symgen](ode[7]);
```

$$[\_\xi = 1, \_\eta = y], [\_\xi = 0, \_\eta = -\ln(\_y1) + x] \qquad (2.1.4.2)$$

対称点 (直前の出力の最初の項) は、方程式の積分にはまったく使用できず、1 階 ODE の求解を困難にする原因になります。一方、動的対称性は以下のように Ei (指数積分) を使用した LambertW 関数で構成される新しい暗黙解を得ることができます。

```
> dsolve(ode[7], implicit);
```

$$\frac{y}{\mathrm{LambertW}\left(-y\,e^{-\left(x e^{x}+e^{x}+\_C1\right)e^{-x}}\right)} - \mathrm{Ei}\left(1, -\mathrm{LambertW}\left(-y\,e^{-\left(x e^{x}+e^{x}+\_C1\right)e^{-x}}\right) - 1 - e^{-x}\,\_C1\right) \_C1 - \_C2 = 0 \qquad (2.1.4.3)$$

非線形 ODE の新しいパラメトリック求解

- 現在、すべての階数の微分法による ODE において、dsolve でその対称性を使用した parametric solutions の直接計算が可能です。これは、特殊関数や積分器の代数反転を伴い、$y(x) = \ldots$の形式の標準的な求解では、あらゆる場合に有用です。また、パラメトリック求解の方が簡潔です。

例

```
> ode[8] := diff(y(x),x,x) = y(x)+_F1(diff(y(x),x)-y(x));
```

$$ode_8 := y'' = y + \_F1(-y + y') \qquad (2.1.5.1)$$

```
> dsolve(ode[8]);
```

$$y = \left(\int \mathrm{RootOf}\left(-x + \int^{\_Z} \frac{1}{-\_f + \_F1(\_f)}\,d\_f + \_C1\right) e^{-x}\,dx + \_C2\right) e^{x} \qquad (2.1.5.2)$$

これは実現可能な最善の求解ですが、複合積分を伴い、内側の積分では積分変数に対する解を RootOf を用いて代数的に求めます。このような解は、RootOf なしでパラメトリック形式で再表示されることが多く、こうなると dsolve を使用して ODE の対称性から直接的に計算できるようになります。

```
> dsolve(ode[8], parametric);
```

$$\left[y(\_T) = \left(\int \frac{e^{\int\left(-\frac{1}{-\_T + \_F1(\_T)}\right)d\_T - \_C2}\,\_T}{-\_T + \_F1(\_T)}\,d\_T + \_C1\right) e^{\int \frac{1}{-\_T + \_F1(\_T)}\,d\_T + \_C2},\ x(\_T) = \int \frac{1}{-\_T + \_F1(\_T)}\,d\_T + \_C2\right] \qquad (2.1.5.3)$$

一般解に特化した新コマンド DEtools[IVPsol] と関連する dsolve の新オプション

- ODE-IVP の解を導く標準的な手順とは、最初により一般的な解を計算し、(これは任意定数によって異なります)、次に、これらの定数の適合により、初期条件または境界条件に合致させる手順です。ただし、初期のより一般的な解は、さ

まざまな方法で記述されることが多く、このまま進めるか、後に解を適合させることでより複合的にすることもできます。

この状況を有効利用するために、ODE 解を指定の初期条件または境界条件に適合させることが目的の新しいコマンド IVPsol が DEtools パッケージに追加されました。また、 dsolve に関連する柔軟な新機能が追加され、指定の解を受け入れて初期/境界条件に適合させる出発点としてこれを直接使用できるようになりました。

例

```
> ode[9] := x*diff(y(x),x,x)+4*y(x)^2 = 0;
```

$$ode_9 := x\,y'' + 4\,y^2 = 0 \qquad (2.1.6.1)$$

ode[9] の一般解は未知です。このため、次に示す例の初期条件でode[9] に関する初期値問題を解くことは原理上不可能です。

```
> ic[9] := y(1) = -1/2, D(y)(1) = 1/2;
```

$$ic_9 := y(1) = -\frac{1}{2}, \mathrm{D}(y)(1) = \frac{1}{2} \qquad (2.1.6.2)$$

ただし、 ode[9] の特殊解が既知であると仮定すれば、次のように、さまざまな方法で解を得ることができる可能性があります：

```
> sol[9] := y(x) = _C1/(2*x);
```

$$sol_9 := y = \frac{1}{2}\,\frac{\_C1}{x} \qquad (2.1.6.3)$$

ここで得られた解を直接 dsolve に渡すことができます。こうすれば、dsolve がこの解を利用して計算できるため、独自に計算を試みて失敗することもなくなります。

```
> dsolve([ode[9], ic[9]], usesolution = sol[9]);
```

$$y = -\frac{1}{2\,x} \qquad (2.1.6.4)$$

同様に、初期条件と解とを直接 DEtools[IVPsol] に渡すことができます。

```
> DEtools[IVPsol]([ic[9]], sol[9]);
```

$$y = -\frac{1}{2\,x} \qquad (2.1.6.5)$$

DEtools: parametricsol および particularsol の機能が拡張

- DEtools コマンドの particularsol と parametricsol では、標準的な手法と対称性を使用した代替的な手法とを組み合わせることで新たに高階かつ非線形の ODE 問題を解くことができるようになりました。

例

```
> ode[10] := diff(y(x), x,x,x) - diff(y(x), x,x)*y(x) + diff
  (y(x),x)^2 = 0;
```

$$ode_{10} := y''' - y''y + y^2 = 0 \qquad (2.1.7.1)$$

ode[10]には既知の一般解はなく、特殊解が 1 つ存在します。ただし、この 1 つの任意定数によっては、次のように計算できます。

```
> DEtools[particularsol](ode[10]);
```

$$y = 0, y = \frac{6}{-x + \_C1} \qquad (2.1.7.2)$$

ode[8]については、前述のパラグラフで紹介したパラメトリック求解を使用して次のように計算できます。

```
> DEtools[parametricsol](ode[8]);
```

$$\left\{\left[y(\_T) = \left(\int \frac{e^{\int\left(-\frac{1}{-\_T + \_F1(\_T)}\right) d\_T - \_C2}\,\_T}{-\_T + \_F1(\_T)}\, d\_T + \_C1\right) e^{\int \frac{1}{-\_T + \_F1(\_T)}\, d\_T + \_C2},\ x(\_T) = \int \frac{1}{-\_T + \_F1(\_T)}\, d\_T + \_C2\right]\right\} \qquad (2.1.7.3)$$

この機能は、 dsolve にオプションとして追加されています。前述のパラグラフの説明を参照してください。

偏微分方程式 (PDE)

PDEtools パッケージには、本リリースで大規模な改良と追加が施されました。これによって、シンボリック演算および偏微分方程式の分野に最新鋭の新たな基準が打ち立てられました。

PDEtool の新コマンド

- PDEtools: CharacteristicQInvariants、 ConsistencyTest、PolynomialSolutions および SymmetrySolutions の 4 種類のコマンドが新たに追加されました。コマンドの名前が示すとおり、これらのコマンドにはそれぞれ次のような機能があります。整合性の検証、PDE 系の多項解の計算、問題の対称性探索により与えられた解を他の微分法の解に変換、系の対称性関数 CharacteristicQ で探索することで実施する、簡潔な手法を用いた微分方程式系の不変式の計算です。

例

```
> with(PDEtools);
```

[*CanonicalCoordinates*, *ChangeSymmetry*, *CharacteristicQ*, *CharacteristicQInvariants*, *ConservedCurrentTest*, *ConservedCurrents*, *ConsistencyTest*, *D_Dx*, *DeterminingPDE*, *Eta_k*, *Euler*, *FromJet*, *InfinitesimalGenerator*, *Infinitesimals*, *IntegratingFactorTest*, *IntegratingFactors*, *InvariantSolutions*, *InvariantTransformation*, *Invariants*, *Laplace*, *Library*, *PDEplot*, *PolynomialSolutions*, *ReducedForm*, *SimilaritySolutions*, *SimilarityTransformation*, *SymmetrySolutions*, *SymmetryTest*, *SymmetryTransformation*, *TWSolutions*, *ToJet*, *build*, *casesplit*, *charstrip*, *dchange*, *dcoeffs*, *declare*, *diff_table*, *difforder*, *dpolyform*, *dsubs*, *mapde*, *separability*, *splitstrip*, *splitsys*, *undeclare*] (2.2.1.1)

この新しい PolynomialSolutions コマンドを使用して PDE 系の多項解を計算する例を次に示します。

```
> PDEtools:-declare((f, g)(x,y));
```

$$f(x, y) \text{ will now be displayed as } f$$

$$g(x, y) \text{ will now be displayed as } g \qquad (2.2.1.2)$$

```
> sys[1] := diff(f(x,y),x)*diff(g(x,y),x) + diff(f(x,y),y)*
  diff(g(x,y),y) + g(x,y)*(diff(f(x,y),x,x) + diff(f(x,y),y,
  y)) = -1;
```

$$sys_1 := f_x g_x + f_y g_y + g\left(f_{x,x} + f_{y,y}\right) = -1 \qquad (2.2.1.3)$$

```
> psol[1] := PolynomialSolutions(sys[1]);
```

$$psol_1 := \left\{f = \frac{\left(\left(-2y^2 + x^2\right)\_C3 + \_C1 + \_C2\, y\right)\_C5 + 2x\,\_C4\,\_C3 - x}{\_C5}, g\right.$$

$$= \_C4 + \_C5\, x\Big\}, \left\{f = \frac{1}{2} \frac{1}{\_C6}\left(\left(-2y^2\,\_C5 + (2x\,\_C4\right.\right.$$

$$\left.\left. + 2\,\_C2)\, y + 2\,\_C1 + 2\,\_C5\, x^2 + 2\,\_C3\, x\right)\_C6 - y^2\right), g = \_C6\Big\},$$

$$\left\{f = \frac{1}{\_C5}\left(-4y\left(\left(\frac{1}{2}\,\_C2 + \_C3\, x\right)\_C5 + \frac{1}{2}\right.\right.\right.$$

$$\left.\left.\left. + \_C4\,\_C3\right) RootOf\left(2\,\_Z^2 - 1\right) + \_C5\left(\_C1 + \_C3\, y^2 + x\,\_C2\right)\right), g\right.$$

$$\left. = \_C4 + RootOf\left(2\,\_Z^2 - 1\right)\_C5\, y + \_C5\, x\right\}, \left\{f\right.$$

$$= \frac{1}{2\,\_C5^3 - \_C5\,\_C6^2}\left(\left(2\,\_C1 + 2x\,\_C2 - \_C3\, y^2 + 2x^2\,\_C3\right)\_C5^3\right.$$

$$- 6y\left(\left(\_C3\, x + \frac{1}{3}\,\_C2\right)\_C6 + \frac{1}{3}\,\_C4\,\_C3 + \frac{1}{3}\right)\_C5^2$$

$$- \_C6^2\left(x^2\,\_C3 + \_C1 + x\,\_C2 - 2\,\_C3\, y^2\right)\_C5 + y\,\_C6^2\,(\_C2\,\_C6$$

$$\left. - 2\,\_C4\,\_C3 + 1)\right), g = \_C4 + \_C5\, y + \_C6\, x\Big\}$$

```
> map(pdetest, [psol[1]], sys[1]);
```

$$[0, 0, 0, 0] \qquad (2.2.1.5)$$

本リリースの新しい PDE 系の求解では、従来の既知の解からその対称性を探索して計算できるようになりました。たとえば、次のような 2 次元の波動方程式があるとします：

```
> pde[2] := diff(u(x, t), x,x) - diff(u(x, t), t,t) = 0;
```

$$pde_2 := u_{x,x} - u_{t,t} = 0 \qquad (2.2.1.6)$$

次に、pdsolve を使用した、積によって可分な求解の例を示します。.

```
> sol[2] := pdsolve(pde[2], HINT = `*`, build);
```

$$sol_2 := u(x,t) = e^{\sqrt{-c_1}\,x}\,\_C1\,\_C3\,e^{\sqrt{-c_1}\,t} + \frac{e^{\sqrt{-c_1}\,x}\,\_C1\,\_C4}{e^{\sqrt{-c_1}\,t}} + \frac{\_C2\,\_C3\,e^{\sqrt{-c_1}\,t}}{e^{\sqrt{-c_1}\,x}} + \frac{\_C2\,\_C4}{e^{\sqrt{-c_1}\,x}\,e^{\sqrt{-c_1}\,t}} \qquad (2.2.1.7)$$

新機能を使用した polynomial タイプの対称性は次のように求められます：

```
> S[2] := Infinitesimals(pde[2], typeofsymmetry = polynomial)
  [-1..-1];
```

$$S_2 := \left[\_\xi_1(x,t,u) = 2\,t\,x, \_\xi_2(x,t,u) = t^2 + x^2, \_\eta_1(x,t,u) = 0\right] \qquad (2.2.1.8)$$

新しい解は、 sol[2] の変換と対称性 S[2]の探索によって導かれます。

```
> newsol[2] := SymmetrySolutions(sol[2], S[2]);
```

$$newsol_2 := \left\{ u(x,t) = \left(\_C2\,\_C3\,e^{\frac{2\sqrt{-c_1}\,((-x^2+t^2)\,\_\varepsilon + t)}{1+(-x^2+t^2)\,\_\varepsilon^2 + 2\,\_\varepsilon t}} + \_C1\,\_C3\,e^{\frac{2\,(x+t)\sqrt{-c_1}\,(1+(t-x)\,\_\varepsilon)}{1+(-x^2+t^2)\,\_\varepsilon^2+2\,\_\varepsilon t}} + \_C4\left(e^{\frac{2\sqrt{-c_1}\,x}{1+(-x^2+t^2)\,\_\varepsilon^2+2\,\_\varepsilon t}}\,\_C1 + \_C2\right)\right) e^{-\frac{(x+t)\sqrt{-c_1}\,(1+(t-x)\,\_\varepsilon)}{1+(-x^2+t^2)\,\_\varepsilon^2+2\,\_\varepsilon t}} \right\} \qquad (2.2.1.9)$$

この新たな解は pdetest を使用して検証します。

```
> pdetest(newsol[2], pde[2]);
```

$$0 \qquad (2.2.1.10)$$

対称性を使用した解の計算に追加された重要な機能

- PDEtool の対称性コマンドのほとんどは機能が大幅に改善したことで、難解な問題においても解を得られるだけの十分な柔軟性を提供し、必要な形式のいずれかを備えるようになっています。特筆される新機能を次に取り上げます：
  - 対称性手法の主要ソルバ InvariantSolutions にパッケージ化されたオプションにより、非常に多様な形式の解を取得できるようになりました。
  - InvariantSolutions、、SimilaritySolutions、、および Infinitesimals は、いずれもより高速な求解、多項タイプの対称性や制限された依存性を持つ対称性の検索が可能であり、解を容易に計算できるようになりました。
  - InvariantSolutions および SimilaritySolutions は、PDE 系の無限小を入力するオプションを含む任意関数を自動的に特殊化して、必要な場合に特殊化されるのを防ぎます。
  - 対称性の手法の主要コマンドである DeterminingPDE は、決定系についても integrating factors および conserved currents の計算ができるようになりま

した。

例

PDE 系では、特殊解 (一般解より迅速な計算が可能) を計算すれば解を得るのに十分な場合がよくあります。本リリースの新機能では、これらの特殊解をさまざまな方法で探索できます。たとえば、先のセクションで説明した 2 次元の波動方程式を思い出してください：

```
> pde[2];
```

$$u_{x,x} - u_{t,t} = 0 \qquad (2.2.2.1)$$

次のようにすると、 t に基づいて求解するだけで十分であることが指定されます。

```
> InvariantSolutions(pde[2], numberofsolutions = 1,
  dependency = t);
```

$$u(x,t) = t\_C1 + \_C2 \qquad (2.2.2.2)$$

次のような別の入力では、 x または t のいずれか 1 つの変数だけに基づいて無限小を使用して解を導くように指定されます。この方が計算は容易です。

```
> InvariantSolutions(pde[2], dependencyofinfinitesimals = {x,
  t});
```

$$u(x,t) = \_C2 + x\_C1,\ u(x,t) = t\_C1 + \_C2 \qquad (2.2.2.3)$$

このように入力することで、多項タイプの無限小を、jet 変数で線形に使用することが指定されます (行列転化のみを必要とした最もシンプルな計算が可能)。ここでは display オプションによって対応する invariants と解が表示されることに注意してください。

```
> InvariantSolutions(pde[2], degreeofinfinitesimals = 1,
  display);
```

$$invariants = [-x^2 + t^2, u(x,t)]$$

$$solutions = [\{u(x,t) = \_C1 + \ln((t-x)\ (x+t))\ \_C2\}]$$

$$invariants = \left[\frac{t}{x}, u(x,t)\right]$$

$$solutions = \left[\left\{u(x,t) = \_C3 - \frac{1}{2}\ \_C4 \ln\left(\frac{x+t}{x}\right) + \frac{1}{2}\ \_C4 \ln\left(\frac{t-x}{x}\right)\right\}\right]$$

$$invariants = [t, u(x,t)]$$

$$solutions = [\{u(x,t) = t\_C1 + \_C2\}]$$

$$invariants = [x, u(x,t)]$$

$$solutions = [\{u(x,t) = \_C3\, x + \_C4\}]$$

$u(x,t) = \_C2 + x\_C1,\ u(x,t) = t\_C1 + \_C2,\ u(x,t) = \_C1 + \ln((t-x)\ (x + t))\ \_C2,\ u(x,t) = \_C1 - \frac{1}{2}\ \_C2 \ln\left(\frac{x+t}{x}\right) + \frac{1}{2}\ \_C2 \ln\left(\frac{t-x}{x}\right)$ (2.2.2.4)

このように表示される不変式は、各解の明確な特徴を示すもので、画面からこ

れを直接コピーして InvariantSolutions の呼び出しに貼り付けることで対応する解を正確に再現できます。

```
> InvariantSolutions(pde[2], invariants = [t/x, u(x,t)]);
```

$$u(x,t) = \_C1 - \frac{1}{2}\,\_C2\,\ln\left(\frac{x+t}{x}\right) + \frac{1}{2}\,\_C2\,\ln\left(\frac{t-x}{x}\right) \qquad (2.2.2.5)$$

pde[2] に関するInfinitesimals の一般形式には、任意関数 $\_Fn(x+t)$ および $\_Fm(-t+x)$ が含まれます。

```
> Infinitesimals(pde[2]);
```

$$[\_\xi_1(x,t,u) = \_F1(x+t) + \_F2(t-x), \_\xi_2(x,t,u) = \_F1(x+t) - \_F2(t - x) + 1, \_\eta_1(x,t,u) = \_F3(x+t) + \_F4(t-x)\,], [\_\xi_1(x,t,u) = \_F5(x+t) + \_F6(t-x), \_\xi_2(x,t,u) = \_F5(x+t) - \_F6(t-x), \_\eta_1(x,t,u) = u + \_F7(x+t) + \_F8(t-x)\,] \qquad (2.2.2.6)$$

SimilarityTransformation 、さらには、これに関するすべての対称性関連の変換を計算することは任意関数のために不可能です。このような無限小を使用するために、 Infinitesimals を除くすべての対称性コマンドにより、この任意関数は自動的に特殊化されるようになりました。例に示すように Infinitesimals については、このコマンドの新しいオプション specialize_Fn を指定して対話的に計算します。

```
> S := Infinitesimals(pde[2], specialize_Fn,
  displayfunctionality = false);
```

$$S := [\_\xi_1 = 1, \_\xi_2 = 0, \_\eta_1 = 0], [\_\xi_1 = 1, \_\xi_2 = 2, \_\eta_1 = 0], [\_\xi_1 = 0, \_\xi_2 = 1, \_\eta_1 = 1], [\_\xi_1 = 1, \_\xi_2 = 1, \_\eta_1 = u], [\_\xi_1 = 1, \_\xi_2 = -1, \_\eta_1 = u], [\_\xi_1 = 0, \_\xi_2 = 0, \_\eta_1 = u + 1], [\_\xi_1 = 0, \_\xi_2 = 1, \_\eta_1 = x + t], [\_\xi_1 = 0, \_\xi_2 = 1, \_\eta_1 = t - x], [\_\xi_1 = 0, \_\xi_2 = 0, \_\eta_1 = u + x + t], [\_\xi_1 = 0, \_\xi_2 = 0, \_\eta_1 = u + t - x], [\_\xi_1 = x + t, \_\xi_2 = x + t, \_\eta_1 = u], [\_\xi_1 = t - x, \_\xi_2 = -t + x, \_\eta_1 = u], [\_\xi_1 = x + t, \_\xi_2 = x + t + 1, \_\eta_1 = 0], [\_\xi_1 = t - x, \_\xi_2 = 1 - t + x, \_\eta_1 = 0] \qquad (2.2.2.7)$$

先のすべてのコマンドは、このようにして pde[2] の対称性を適切に取り扱うことができます。

```
> SymmetryTransformation(S[5], u(x, t), v(r, s));
```

$$\{r = x + \_\varepsilon, s = -\_\varepsilon + t, v(r,s) = u(x,t)\,e^{-\varepsilon}\} \qquad (2.2.2.8)$$

これで不変式の解は、このように任意関数の特殊化が自動化されたことで自動的に計算できるようになります。

```
> InvariantSolutions(pde[2]);
```

$$u(x,t) = \frac{\_C1}{\sqrt{t-x}}, u(x,t) = \_C1\sqrt{x+t}, u(x,t) = t + x\,\_C1 + \_C2, u(x,t) = \_C1\,(-2x+t) + \_C2, u(x,t) = \frac{\_C1\,e^{\frac{1}{2}t - \frac{1}{2}x}}{e^{-x}}, u(x,t) \qquad (2.2.2.9)$$

$$= \frac{\_C1\, e^{-\frac{1}{2}x-\frac{1}{2}t}}{e^{-x}}, u(x,t) = \frac{1}{2}x^2 + \frac{1}{2}(-2t+2\_C1)x + \frac{1}{2}t^2$$

$$+ \_C2, u(x,t) = \frac{1}{2}x^2 + \frac{1}{2}(2t+2\_C1)x + \frac{1}{2}t^2 + \_C2, u(x,t)$$

$$= \_C1 + \_C2 \ln\left((-1+2t-2x)\, e^{2x+2t}\right), u(x,t) = \_C1$$

$$+ \_C2 \ln\left((2x+2t+1)\, e^{2x-2t}\right)$$

内部 PDEtool ルーチンライブラリのプログラミング利用が実現

- 微分方程式の操作およびプログラミングを目的とする PDEtools internal library が Maple 13 から利用できるようになりました。このライブラリには 45 種類のサブルーチンが用意されています。

例

PDEtool ルーチンライブラリは、主にプログラミング目的で利用するために作成されたものです。ただし、このライブラリのルーチンの一部は、演算利用に適しています。たとえば、sys[1] について新コマンドの PolynomialSolutions を使用して直前に得られる結果は、解の次数について最初に予測された上限によって次のように計算できます。

```
> sys[1];
```

$$f_x g_x + f_y g_y + g\left(f_{x,x} + f_{y,y}\right) = -1 \qquad (2.2.3.1)$$

```
> Describe(PDEtools:-Library:-UpperBounds);

# Input: a PDE system PDESYS and a list of dependent variables F
# Output: a sequence of sets of equations, where the lhs is a function of F
and the rhs is an upper bound for the degree of a solution polynomial in the
independent variables
#
UpperBounds( PDESYS, F,
          { X::list(name) := GetIndepVars(F) } )
```

```
> PDEtools:-Library:-UpperBounds(sys[1], [f, g](x,y));
```

$$\{f=2, g=1\} \qquad (2.2.3.2)$$

dpolyform および casesplit に関する機能拡張

- 本リリースの casesplit コマンドと dpolyform コマンドは、代数式 (シンボル変数以外) の任意関数を使用して計算できるようになりました。この機能は次のように説明できます。一定の未知の要素 (つまり任意関数の非有理の要素で、その引数は独立変数において非有理) についての式が与えられたときに、微分多項表現 (PDE 系) を計算し、指定の式を用いて求解する場合、PDE 系は未知の要素およびその導関数について有理となり有理係数を持つことになります。

動機付け： 代数式の任意関数 (通常は微分不変式) は、公式を適用するか、より大規模な PDE 問題に発展するような中間物の PDE 問題を解くときに頻繁に現れます。本リリースの新機能では、これらのオブジェクトでの微分除去が許容されており、それに伴ってこれらが関与する非線形 PDE 系の脱共役も可能となったことから、PDE 系およびその解で実現できる内容が大幅に拡大しました。

例

次のように、任意関数 $_F1(x+t) + _F2(-t+x)$が関与する式 $g(x,t)$があるとします。

```
> sol[3] := g(x,t) = _F1(x+t) + _F2(x-t);
```

$$sol_3 := g(x, t) = \_F1(x + t) + \_F2(-t + x) \qquad (2.2.4.1)$$

この式で満足するのはどのような PDE でしょうか。次のように計算できます。2 次元の波動方程式を使用します。

```
> pde[3] := dpolyform(sol[3], no_Fn);
```

$$pde_3 := [g_{x,x} = g_{t,t}] \ \&\text{where}\ [\,] \qquad (2.2.4.2)$$

変数 $x, t, \ldots$ の一定の写像に対する differential polynomial representations を計算する手法は、変数 $(\sin(x + t)$ など)に適用される写像の演算上の特性の知識を利用して行う従来の機能です。一方、Maple 13 の新機能には、特定の演算上の特性を持たない *任意写像* ($\_F1(x + t)$ など) を用いてこれらの PDE 表現を計算する機能があります。式で評価される任意写像の導関数も有理 PDE 形式で表すことができます。

```
> sol[4] := g(x,t) = D(_F1)(ln(x*t) + _F2(1/exp(t)));
```

$$sol_4 := g(x, t) = \mathrm{D}(\_F1)\left(\ln(t\,x) + \_F2\left(\frac{1}{e^t}\right)\right) \qquad (2.2.4.3)$$

```
> pde[4] := dpolyform(sol[4], no_Fn);
```

$$pde_4 := \left[g_{t,x} = \frac{g_t\, g_{x,x}}{g_x} + \frac{g_t}{x}\right] \ \&\text{where}\ [g_t \neq 0, g_x \neq 0] \qquad (2.2.4.4)$$

この結果は、通常どおり pdetest を使用して検証できます。

```
> pdetest(sol[4], pde[4]);
```

$$[0] \qquad (2.2.4.5)$$

- casesplit コマンドは、新しいオプション引数の caseplotも許可するため、複数のケースへの分割に加え、この分割をグラフィカルに表現するプロットも生成されます。

```
> ode[11] := diff(y(t),t,t)^2 + 2*diff(y(t),t,t)*y(t)^3*diff
  (y(t),t) - 4*y(t)^2*diff(y(t),t)^3;
```

$$ode_{11} := y_{t,t}^2 + 2\, y_{t,t}\, y(t)^3\, y_t - 4\, y(t)^2\, y_t^3 \qquad (2.2.4.6)$$

```
> casesplit(ode[11], caseplot);
```

$$========= \textit{Pivots Legend} =========$$

$$p1 = y_{t,t} + y(t)^3\, y_t$$

$$p2 = y(t)$$

$$p3 = y_t$$

$$PLOT(\ldots)$$

$$\left[y_{t,t}^2 = -2\, y_{t,t}\, y(t)^3\, y_t + 4\, y(t)^2\, y_t^3\right] \ \&\text{where}\ \left[y_{t,t} + y(t)^3\, y_t \neq 0\right], \left[y_t = -\frac{1}{4}\, y(t)^4\right] \ \&\text{where}\ [y(t) \neq 0], [y_t = 0] \ \&\text{where}\ [y(t) \neq 0], [y(t) = 0] \ \&\text{where}\ [\,] \qquad (2.2.4.7)$$

境界条件に関する pdetest の機能拡張

- 本リリースの pdetest コマンドは、解の正確性を境界条件の観点から検証できるようになりました。

例

次の PDE の例では、境界条件と解を検討します。

```
> pde[3] := diff(u(x,t),t) = k*diff(diff(u(x,t),x),x)+Q;
```

$$pde_3 := u_t = k\,u_{x,x} + Q \qquad (2.2.5.1)$$

```
> bc[3] := u(0,t) = 2*exp(k*t)-1/k*Q;
```

$$bc_3 := u(0,t) = 2\,e^{kt} - \frac{Q}{k} \qquad (2.2.5.2)$$

```
> sol[3] := u(x,t) = _C1^2*exp(x+k*t)-(_C1^2-2)*exp(-x+k*t)
  -1/2/k*Q*x^2+1/_C1^2/k*Q*(_C1^2-2)*x-1/k*Q;
```

$$sol_3 := u(x,t) = \_C1^2\,e^{x+kt} - (\_C1^2-2)\,e^{-x+kt} - \frac{1}{2}\,\frac{Q\,x^2}{k} + \frac{Q\,(\_C1^2-2)\,x}{\_C1^2\,k} - \frac{Q}{k} \qquad (2.2.5.3)$$

pdetest を使用して sol[1] で pde[1] が求解されるかどうか検証します。本リリースには、ここで境界条件の bc[1] が導かれるかどうかを検証できる新機能があります。

```
> pdetest(sol[3], [pde[3], bc[3]]);
```

$$[0, 0] \qquad (2.2.5.4)$$

境界条件には、次のように導関数が関与する場合があります：

```
> bc[3.1] := D[1,1](u)(0,t) = 2*exp(k*t)-1/k*Q;
```

$$bc_{3.1} := D_{1,1}(u)(0,t) = 2\,e^{kt} - \frac{Q}{k} \qquad (2.2.5.5)$$

```
> pdetest(sol[3], [pde[3], bc[3.1]]);
```

$$[0, 0] \qquad (2.2.5.6)$$

数値解

Maple 13 では、ODE の数値解において、いくつかの側面で機能が改良されました。

- Maple 12 で導入されたイベント処理機能が拡張され、イベントの初期トリガーを制御できる機能が追加されました。さらに、新規イベントのプログラミング構造体 (tobegin、toend、breakiteration、delayhalt) や丸め制御に関するサイド構造体が追加されました。また、根探索トリガーを制限するトリガー表現の領域の増減を行う機能やイベント発生時に対話的にクエリーを提出する機能 (eventfired) も追加されました。詳細については、 dsolve,Events を参照してください。
- 中核ソルバの堅牢性が向上しました。純粋な ODE 問題および添え字 1 つの DAE 変数に関する正確性とエラー制御の機能が改良されました。これによって解の不連続性の検知機能が改善しました (詳細については、examples, dsolve_numeric_NewErrorControl を参照してください)。

エラー制御機能が改善したため、Maple は以下のような問題でより正確な解を得ることができるようになりました：

```
> dsn:=dsolve({diff(x(t),t)=1,y(t)=sin(Pi*t),x(0)=0},numeric):
```

```
> dsn(10);
```

$$[t = 10., x(t) = 10., y(t) = 3.77931112383999429\ 10^{-8}] \quad (3.1)$$

明示的な Runga-Kutta 技法のエラー制御メカニズムに修正が加えらたことから、Maple は次のような問題でより正確な解を得ることができるようになりました：

```
> dsn2:=dsolve({diff(x(t),t)=x(t)/(1-t),x(0)=1},numeric):
```

```
> dsn2(0.99);
```

$$[t = 0.99, x(t) = 99.999993181301562] \quad (3.2)$$

Maple では、より多くの洗練されたエラー検知も可能になりました：

```
> dsn2(1.01);
```

```
Error, (in dsn2) cannot evaluate the solution further right of .99999999, probably a singularity
```

ここでは $t = 1$ における ODE の解が特異であるとしてエラーが返されます。

- 投影技法の改良により、相対的な少数自由度を持つ DAE 問題のパフォーマンスが改善されました。
- optimize=true オプションの最適化の効率が向上しました。また、デフォルトの rkf45 ソルバおよび rosenbrock ソルバを使用してハードウェア精度を持つ evalhf 対応の問題を扱う場合に、Compiler で使用する新オプション compile が追加されました。
- 対話型クエリーオプション numfun も追加されました。これにより、IVP 問題における解の値を計算する方法で行う、ODE の評価手法が数種類提供されます (maxfunをサポートするすべての IVP ソルバでサポートされます)。

参照

Index of New Maple 13 Features