Symbolic and Numeric Scientific Computation in Maple

K. O. Geddes[*]
Symbolic Computation Group
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
`kogeddes@scg.math.uwaterloo.ca`
`http://www.uwaterloo.ca/~kogeddes`

H. Q. Le[†]
Symbolic Computation Group
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
`hqle@scg.math.uwaterloo.ca`

Abstract

This paper is an exposition of some recent advances in the Maple computer algebra system for both symbolic and numeric scientific computation. Facilities for matrix computations include a convenient syntax for matrix operations, a seamless interface between symbolic and numeric modes, and access to hardware floating point speed for numerical matrices. Techniques for computing exact symbolic solutions of differential equations include decision procedures for special types of DEs, the classification of DEs into known classes, and symmetry-based methods. New numerical solvers which have been incorporated into Maple take full advantage of hardware floating point speed, for definite integrals, IVPs and BVPs. Hybrid symbolic-numeric computational strategies are discussed in the context of evaluating definite integrals in the presence of integrand singularities.

[*]Partially supported by Natural Sciences and Engineering Research Council of Canada Grant No. RGPIN8967-01.

[†]Partially supported by Natural Sciences and Engineering Research Council of Canada Grant No. CRD215442-98.

1 Matrix Computations in Maple

Some of the features appearing in recent versions of Maple [7], [8] for computational linear algebra are highlighted by the following examples. The examples are trivial, chosen simply to illustrate the convenient syntax for matrix operations.

1.1 Matrix Arithmetic

1.1.1 Example 1.1. Matrix initialization and simple operations

```
> A := < <  1 | -1 |  2 > ,
>        < -3 |  4 | -5 > ,
>        <  5 | -6 |  6 > >;
```

$$A := \begin{bmatrix} 1 & -1 & 2 \\ -3 & 4 & -5 \\ 5 & -6 & 6 \end{bmatrix}$$

```
> b := < 2, 0, -3 >;
```

$$b := \begin{bmatrix} 2 \\ 0 \\ -3 \end{bmatrix}$$

```
> A.b;
```

$$\begin{bmatrix} -4 \\ 9 \\ -8 \end{bmatrix}$$

```
> A^2;
```

$$\begin{bmatrix} 14 & -17 & 19 \\ -40 & 49 & -56 \\ 53 & -65 & 76 \end{bmatrix}$$

```
> A^(-1);
```

$$\begin{bmatrix} 2 & 2 & 1 \\ \frac{7}{3} & \frac{4}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} \end{bmatrix}$$

```
> A . A^(-1);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> with(LinearAlgebra):
> Determinant(A);
```

$$-3$$

```
> Eigenvalues(A);
```

$$\begin{bmatrix} -1 \\ 6+\sqrt{33} \\ 6-\sqrt{33} \end{bmatrix}$$

1.1.2 Example 1.2. Solving linear systems

```
> x := LinearSolve(A, b);
```

$$x := \begin{bmatrix} 1 \\ \frac{11}{3} \\ \frac{7}{3} \end{bmatrix}$$

```
> Norm(b - A.x);
```

$$0$$

Of course, multiplying $A^{-1}\,b$ yields the solution x.

```
> A^(-1).b;
```

$$\begin{bmatrix} 1 \\ \frac{11}{3} \\ \frac{7}{3} \end{bmatrix}$$

1.1.3 Example 1.3. Matrix factorization

```
> (P,L,U) := LUDecomposition(A);
```

$$P,\ L,\ U := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 5 & -1 & 1 \end{bmatrix},\ \begin{bmatrix} 1 & -1 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & -3 \end{bmatrix}$$

```
> P.L.U;
```

$$\begin{bmatrix} 1 & -1 & 2 \\ -3 & 4 & -5 \\ 5 & -6 & 6 \end{bmatrix}$$

```
> Norm(% - A);
```

$$0$$

Illustrate the solution of $A\,x = b$ via LU factorization; i.e., solve $P\,L\,U\,x = b$ for x. First solve $L\,y = P^T\,b$ for y.

```
> y := ForwardSubstitute(L, Transpose(P).b);
```

$$y := \begin{bmatrix} 2 \\ 6 \\ -7 \end{bmatrix}$$

Then solve $U\,x = y$ for x.

```
> x := BackwardSubstitute(U, y);
```

$$x := \begin{bmatrix} 1 \\ \frac{11}{3} \\ \frac{7}{3} \end{bmatrix}$$

```
> Norm(A.x - b);
```

$$0$$

1.1.4 Example 1.4. Matrices with hardware floating point entries

```
> F := RandomMatrix(5,5, generator=-10.0..10.0);
```

$$
\begin{aligned}
F \;:=\; & [-1.01439338040108851\,,\ 8.31419636347767721\,,\ -7.13185338667599922\,,\\
& -5.10394459322505156\,,\ -8.61414450398712006]\\
& [-5.90453234289888940\,,\ 1.44472461637079163\,,\ 3.17982409835046552\,,\\
& -9.18952646370712500\,,\ -4.48926596513011590]\\
& [-5.03483867254843176\,,\ 7.72738945913964415\,,\ -2.78006880722472260\,,\\
& -7.07958699730902196\,,\ 3.55313300143521360]\\
& [-7.00754018998686945\,,\ 6.14562115555997934\,,\ 4.53956171215260440\,,\\
& -9.30654392812769515\,,\ 4.68352373073820160]\\
& [-7.83956118483303044\,,\ 4.83363171033040828\,,\ 4.13867569386937362\,,\\
& 4.81114426570961484\,,\ 4.06975811141194300]
\end{aligned}
$$

```
> G := RandomMatrix(5,3, generator=-10.0..10.0);
```

$$
G := \begin{bmatrix}
-1.87081053332853742 & 1.85576635678003754 & -2.60302376525494594\\
7.90224464364470691 & -4.40080088995820518 & 8.46938784380682820\\
-9.80154722959191815 & 8.09780501356328840 & -2.27841067253869766\\
-0.987992355307198978 & 1.17438287631678406 & -9.26079671891372414\\
-1.39021191579296044 & 5.42897504792321683 & -0.854277624065744589
\end{bmatrix}
$$

Operations on matrices containing only hardware floating point entries are performed at hardware floating point speed. Some timing information is presented in examples in the following section.

```
> F.G;
```

$$
\begin{bmatrix}
154.519893878347944 & -148.983918677495922 & 143.931398802227051\\
6.61584551387889786 & -26.7299177546886356 & 109.298041938304450\\
99.7868968956658052 & -54.8869170886515008 & 147.413455526727262\\
19.8629570921824730 & 11.2077608566041978 & 142.132438853521052\\
1.88624867646298000 & 25.4386848311215062 & 3.88313053107022998
\end{bmatrix}
$$

```
> eigs := Eigenvalues(F);
```

$$
eigs := \begin{bmatrix}
13.7807482903409576 + 0.\,I\\
-4.31383031944290441 + 7.40836668667835596\,I\\
-4.31383031944290441 - 7.40836668667835596\,I\\
-6.36980551971295660 + 0.796076015286287908\,I\\
-6.36980551971295660 - 0.796076015286287908\,I
\end{bmatrix}
$$

```
> poly := CharacteristicPolynomial(F, z);
```

$$poly := -41735.16779 - 14773.49427\, z - 1803.549535\, z^2 - 69.84276357\, z^3 + 7.586523388\, z^4 + z^5$$

```
> factor(poly);
```

$$(z - 13.78074829)\,(z^2 + 12.73961104\, z + 41.20815938)(z^2 + 8.627660636\, z + 73.49302899)$$

```
> (Q,R) := QRDecomposition(G):
> Q;
```

$$\begin{bmatrix} -0.145675713676126194 & -0.0891785668660267650 & 0.146813954407106622 \\ 0.615329616547615532 & -0.348467909713608804 & -0.312207720673749134 \\ -0.763223940834159120 & -0.124479406704417442 & -0.376426197253984740 \\ -0.0769326924890997854 & -0.0880871895406016847 & 0.856716373747034576 \\ -0.108252604625798984 & -0.920523124889811917 & 0.0728860406229195624 \end{bmatrix}$$

```
> R;
```

$$\begin{bmatrix} 12.8422953017949110 & -9.83677099319315395 & 8.13453589543462385 \\ 0. & -4.74091182372517040 & -0.833420876930560439 \\ 0. & 0. & -10.1648561180101140 \end{bmatrix}$$

Check that $Q^T\,Q$ is the identity matrix. Use Maple's `fnormal` function to round the entries to 15 digits.

```
> map(fnormal, Transpose(Q).Q, 15);
```

$$\begin{bmatrix} 1.00000000000000 & 0. & 0. \\ 0. & 1.00000000000000 & 0. \\ 0. & 0. & 1. \end{bmatrix}$$

1.2 LinearSolve: Larger Dimensions

```
> with(LinearAlgebra):
> n := 100:
```

Generate a random square matrix of order n and a random vector of order n. By right-clicking on the output structure in an interactive Maple session, one can choose from a *context menu* various operations to be computed or choose to browse the matrix to view its structure.

```
> A := RandomMatrix(n,n, generator=-100.0..100.0);
```

$$A := \begin{bmatrix} \text{100 x 100 Matrix} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

```
> b := RandomVector(n, generator=-100.0..100.0);
```

$$b := \begin{bmatrix} \text{100 Element Column Vector} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

1.2.1 Example 1.5. Hardware versus software floating point

Determine the computation time to solve a linear system of order n in *hf* (hardware float) mode, for comparison with the time for software float mode.

```
> hf_time := time( LinearSolve(A, b) );
```

$$hf\_time := 0.059$$

```
> x := LinearSolve(A, b);
```

$$x := \begin{bmatrix} \text{100 Element Column Vector} \\ \text{Data Type: float[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

The environment variable `UseHardwareFloats` controls whether hardware floats or software floats are used for the computation. Determine the computation time to solve the same linear system in software floats. Note that the speedup obtained by using hardware floats is significant.

```
> UseHardwareFloats := false:
> sf_time := time( LinearSolve(A, b) );
```

$$sf\_time := 2.793$$

```
> hf_SpeedUp := evalf[3]( sf_time / hf_time );
```

$$hf\_SpeedUp := 47.3$$

```
> UseHardwareFloats := true:
```

1.2.2 Example 1.6. A determinant relationship

Here we illustrate the following determinant relationship: if $A = PLU$ then $\det(A) = \det(P)\det(L)\det(U)$.

```
> (P,L,U) := LUDecomposition(A):
> L;
```

$$\left[\begin{array}{c} \text{100 x 100 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: triangular[lower,strict]} \\ \text{Shape: triangular[lower,unit]} \\ \text{Order: Fortran\_order} \end{array}\right]$$

```
> Determinant(L);  # We know that the value is 1
```

$$1.0$$

```
> U;
```

$$\left[\begin{array}{c} \text{100 x 100 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: triangular[upper]} \\ \text{Shape: triangular[upper]} \\ \text{Order: Fortran\_order} \end{array}\right]$$

```
> Determinant(U);
```

$$0.1763815879\,10^{255}$$

$\det(U)$ is simply the product of the diagonal elements:

```
> detU := mul(U[i,i], i=1..n);
```

$$detU := 0.1763815879\,10^{255}$$

```
> time_detA := time( Determinant(A) );
```

$$time\_detA := 0.299$$

```
> Determinant(A);
```

$$-0.1763815879\,10^{255}$$

$\det(A)$ is equal to $\det(U)$ up to the sign determined by the permutation matrix, so $\det(P) = -1$ in this example.

Since P is an exact integer matrix whereas A is a hardware floating point matrix, the computation time for explicitly computing $\det(P)$ is large compared with the computation time for computing $\det(A)$. Of course, one does not explicitly form the permutation matrix and compute its determinant in the manner shown here as part of a method for computing $\det(A)$. The comparison of computation times is shown here to illustrate the speed advantages of hardware floating point mode.

```
> P;
```

$$\begin{bmatrix} \text{100 x 100 Matrix} \\ \text{Data Type: integer[1]} \\ \text{Storage: sparse} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

```
> time_detP := time( Determinant(P) );
```

$$\mathit{time\_detP} := 2.253$$

```
> Determinant(P);
```

$$-1$$

```
> TimeRatio := evalf[3]( time_detP / time_detA );
```

$$\mathit{TimeRatio} := 7.52$$

1.3 QR Decomposition

In this section we illustrate the solution of a *linear least squares* problem via QR decomposition. The problem being solved is simple and the point is to convey some of the facilities in Maple for problem solving (and for teaching).

Consider the problem of calculating a least squares fit by a polynomial to a set of data. For the purposes of this demonstration, create a set of "experimental data" by first generating a random polynomial and then for a chosen list of data points, let the corresponding function values be perturbations of the values of the polynomial at these data points.

1.3.1 Example 1.7. Generate data values using RandomTools

Generate a random polynomial p(x) of degree $n-1$ (i.e., the number of coefficients is n).

```
> with(RandomTools):
> n := 3:
> p := unapply( Generate(polynom(integer(range=1..10), x, degree=n-1)), x );
```

$$p := x \to 2 + x + 8\,x^2$$

Choose a number, *Npts*, of data points in a specified range *lo*..*hi* for the independent variable. Define the corresponding function values for the "experimental data" to be the values of p(x) at these data points, perturbed by random perturbations in a chosen range $-\varepsilon..\varepsilon$.

```
> Npts := 7:  lo := 0.0:  hi := 3.0:
> incr := evalf[2]( (hi-lo)/(Npts-1) ):
> X := [ seq(lo + (i-1)*incr, i = 1..Npts) ];
```

$$X := [0., 0.50, 1.00, 1.50, 2.00, 2.50, 3.00]$$

```
> epsilon := 0.2;
```

$$\varepsilon := 0.2$$

```
> perturb := Generate(list(float(range=0..2*epsilon, digits=2), Npts)):
> perturb := map(t -> t-epsilon, perturb);
```

$$perturb := [0.03, 0.03, -0.112, -0.187, 0.07, -0.181, 0.19]$$

```
> data := [ seq([X[i], p(X[i])*(1+perturb[i])], i=1..Npts) ];
```

$$\begin{aligned} data \;:=\; & [[0., 2.06], [0.50, 4.635000], [1.00, 9.7680000], [1.50, 17.4795000], \\ & [2.00, 38.520000], [2.50, 44.6355000], [3.00, 91.630000]] \end{aligned}$$

```
> DataPlot := plot(data, x=lo..hi, style=point,symbol=BOX,symbolsize=14):
```

1.3.2 Example 1.8. Polynomial fit to the data via QR decomposition

Let the vector a of size n represent the coefficients of the polynomial fit:
$\mathrm{p}(x) = a_1 + a_2\,x + \ldots + a_n\,x^{n-1}$.
In matrix formulation, we must solve an *overdetermined* linear system $A\,a = b$ for a. More precisely, we wish to find the vector a which minimizes $A\,a - b$ in the least squares norm, where the matrix A and right-hand-side vector b are formed from the data values as shown below. We apply the QR decomposition of A to solve this linear least squares problem.

```
> with(LinearAlgebra):
> A := Matrix(Npts, n, (i,j) -> data[i][1]^(j-1));
```

$$A := \begin{bmatrix} 1. & 0. & 0. \\ 1. & 0.50 & 0.2500 \\ 1. & 1.00 & 1.0000 \\ 1. & 1.50 & 2.2500 \\ 1. & 2.00 & 4.0000 \\ 1. & 2.50 & 6.2500 \\ 1. & 3.00 & 9.0000 \end{bmatrix}$$

```
> b := Vector(Npts, i -> data[i][2]);
```

$$b := \begin{bmatrix} 2.06 \\ 4.635000 \\ 9.7680000 \\ 17.4795000 \\ 38.520000 \\ 44.6355000 \\ 91.630000 \end{bmatrix}$$

```
> (Q,R) := QRDecomposition(A):
> Q;
```

$$\begin{bmatrix} -0.377964473009227308 & -0.566946709513840962 & 0.545544725589981460 \\ -0.377964473009227252 & -0.377964473009227252 & -0.666133814775093924\,10^{-15} \\ -0.377964473009227252 & -0.188982236504613654 & -0.327326835353988544 \\ -0.377964473009227252 & 0. & -0.436435780471984724 \\ -0.377964473009227252 & 0.188982236504613654 & -0.327326835353988654 \\ -0.377964473009227252 & 0.377964473009227252 & 0.138777878078144568\,10^{-15} \\ -0.377964473009227252 & 0.566946709513840962 & 0.545544725589981128 \end{bmatrix}$$

```
> R;
```

$$\begin{bmatrix} -2.64575131106459072 & -3.96862696659688608 & -8.59869176095991960 \\ 0. & 2.64575131106459028 & 7.93725393319377304 \\ 0. & 0. & 2.29128784747792036 \end{bmatrix}$$

Since $A = Q\,R$, solve $R\,a = Q^T\,b$ for the vector a. Note that R is upper triangular.

```
> a := BackwardSubstitute(R, Transpose(Q).b);
```

$$a := \begin{bmatrix} 4.47510714285715672 \\ -9.27664285714288184 \\ 12.0794285714285792 \end{bmatrix}$$

The desired polynomial fit is as follows, where we choose to round the coefficients to four significant digits.

```
> a := map(evalf[4], a):
> poly := add(a[i]*x^(i-1), i=1..n);
```

$$poly := 4.475 - 9.277\,x + 12.08\,x^2$$

Plot both the "experimental data" and the polynomial fit.

```
> PolyPlot := plot(poly, x=lo..hi):
> plots[display]( {DataPlot, PolyPlot} );  # Figure 1
```

Of course, given the matrix A and the vector b, we could have solved the linear least squares problem by directly invoking the `LeastSquares` function as follows.

```
> anew := LeastSquares(A, b):
> map(evalf[4], anew);
```

$$\begin{bmatrix} 4.475 \\ -9.277 \\ 12.08 \end{bmatrix}$$

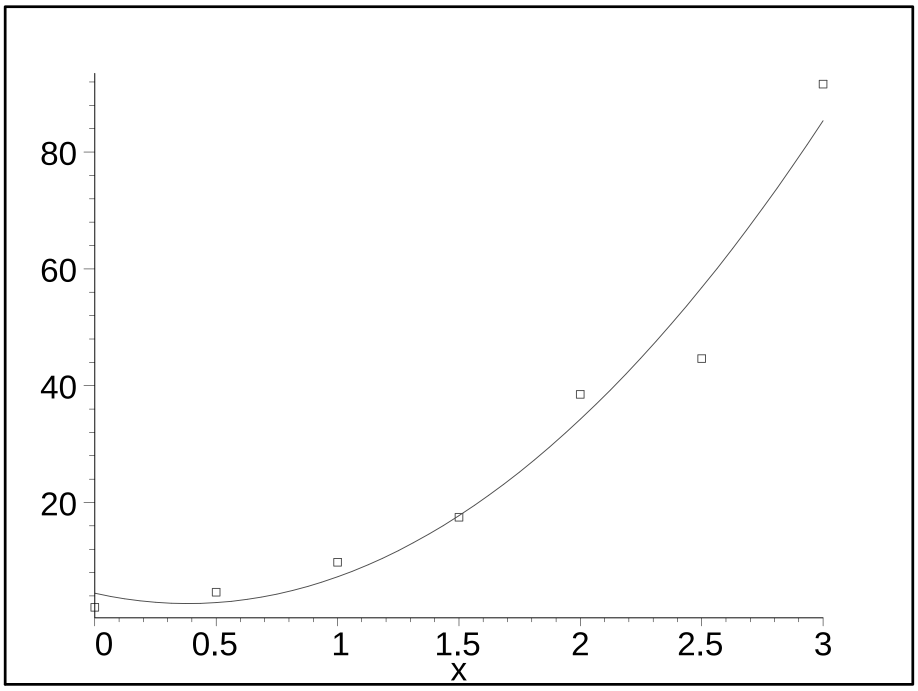


Figure 1: Polynomial fit to data.

1.4 Large (Structured) Matrices

1.4.1 Example 1.9. Large sparse matrices

The following command will generate, very quickly, a random sparse matrix with sparsity specified by the `density` option and with floating point entries in the range specified by the `generator` option.

```
> with(LinearAlgebra):
> n := 1000:
> M1 := RandomMatrix(n, n, generator=0.0..1.0, density=0.05,
        outputoptions=[storage=sparse,datatype=float[8]]);
```

$$M1 := \left[\begin{array}{c} \text{1000 x 1000 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: sparse} \\ \text{Order: Fortran\_order} \end{array}\right]$$

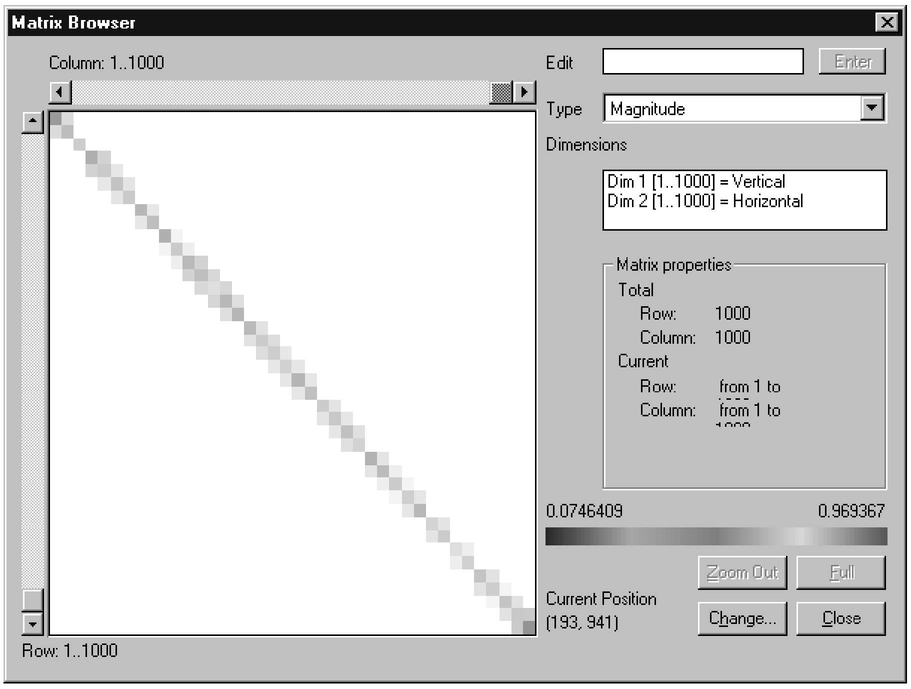
The following command generates a symmetric, banded matrix.

```
> M2 := RandomMatrix(n, n, generator=0.0..1.0, density=0.1,
outputoptions=[shape=symmetric,storage=band[0,5],datatype=float[8]]);
```

$$M2 := \left[\begin{array}{c} \text{1000 x 1000 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: band[0,5]} \\ \text{Shape: symmetric} \\ \text{Order: Fortran\_order} \end{array}\right]$$

In an interactive Maple session, by right-clicking on the placeholder output seen above one can use the Matrix Browser to view the structure of the matrix (Figure 2).

Figure 2: Matrix browser for $M2$


Matrix Browser

Column: 1..1000

Edit

Type: Magnitude

Dimensions

Dim 1 [1..1000] = Vertical
Dim 2 [1..1000] = Horizontal

Matrix properties
Total
Row: 1000
Column: 1000
Current
Row: from 1 to
Column: from 1 to

0.0746409 0.969367

Current Position
(193, 941)

Row: 1..1000

Operations on structured matrices are performed efficiently. Here we multiply matrix *M2* by the vector of size n with all entries 1.0 .

```
> b := Vector(1..n, 1.0):
> M2.b;
```

$$\left[\begin{array}{c} \text{1000 Element Column Vector} \\ \text{Data Type: float[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{array}\right]$$

1.4.2 Example 1.10. Operations on a banded matrix

For this example, consider the following banded matrix of order 100.

```
> n := 100:
> B := LinearAlgebra:-RandomMatrix(n, n, generator=0.0..1.0, density=0.05,
outputoptions=[shape=symmetric,storage=band[0,5],datatype=float[8]]);
```

$$B := \begin{bmatrix} \text{100 x 100 Matrix} \\ \text{Data Type: float[8]} \\ \text{Storage: band[0,5]} \\ \text{Shape: symmetric} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

By right-clicking on the output B in an interactive Maple session, the following are some of the operations which can be invoked from the context menu:

- Browse (to see the band structure)
- Determinant
- Rank
- Singular Values (and Browse the vector of singular values)

For example, let us compute the singular values of B and then calculate the *numerical rank* [5] of the matrix as follows.

```
> singvals := LinearAlgebra:-SingularValues(B);
```

$$singvals := \begin{bmatrix} \text{100 Element Column Vector} \\ \text{Data Type: float[8]} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix}$$

```
> delta := evalhf(DBL_EPSILON) * LinearAlgebra:-Norm(B);
```

$$\delta := 0.3161404587\,10^{-15}$$

Since the singular values are ordered from largest to smallest, we may apply the following bisection method to find the last “numerically nonzero” singular value.

```
> lo := 1:  hi := n:
> while hi-lo > 1 do
    mid := iquo(lo+hi, 2);
    if singvals[mid] > delta then lo := mid else hi := mid end if
  end do:
> num_rank := lo;
```

$$num\_rank := 43$$

By looking at a few singular values near the cutoff, we see that the numerical rank is well-defined in this example.

```
> singvals[num_rank-1 .. num_rank+2];
```

$$\begin{bmatrix} 0.04973062190055448 78 \\ 0.0098239501148672681 8 \\ 0.315472208202137882\,10^{-17} \\ 0.493038065763132379\,10^{-31} \end{bmatrix}$$

2 Symbolic Solution of ODEs

The `dsolve` command in Maple is a general ODE solver which handles various types of ODE problems including:

- closed form solutions of a single ODE or a system of ODEs
- closed form solutions of ODEs with given initial or boundary conditions
- formal power series solutions of a linear ODE with polynomial coefficients
- solutions obtained via integral transforms
- truncated series solutions
- numerical solutions

Some of these types of solutions are illustrated in the following sections.

2.1 Solution Techniques for a Single ODE

For the task of computing closed form solutions of a single ODE, two general strategies are employed [2]:

- **Classification methods**
    - determine whether the given ODE matches a recognizable pattern

  - if yes, apply a known solution method for that pattern
  - Special case: Linear ODE, for which decision procedures are implemented

- **Symmetry methods**
  - look for generators of the symmetry groups of the given ODE
  - use this information to integrate the ODE, or at least to reduce the order of the ODE

2.1.1 The odeadvisor command and the classification types

The `odeadvisor` command is an innovative addition to Maple:

- especially for teaching and learning
- classifies a given ODE according to standard textbook classifications
- provides user access to the classification strategies used by the `dsolve` command

The goal of the `odeadvisor` command is to classify a given ODE according to standard textbooks [6], [9]. The classification types known to this command are described in the help page `?odeadvisor` and they are as follows.

- First order ODEs

Abel,	Abel2A,	Abel2C,	Bernoulli,	Chini,
Clairaut,	dAlembert,	exact,	homogeneous,	homogeneousB,
homogeneousC,	homogeneousD,	homogeneousG,	linear,	patterns,
quadrature,	rational,	Riccati,	separable,	sym_implicit

- Second order ODEs

Bessel,	Duffing,	ellipsoidal,	elliptic,	Emden,
erf,	exact_linear,	exact_nonlinear,	Gegenbauer,	Halm,
Hermite,	Jacobi,	Lagerstrom,	Laguerre,	Lienard,
Liouville,	linear_ODEs,	linear_sym,	missing,	Painleve,
quadrature,	reducible,	sym_Fx,	Titchmarsh,	Van_der_Pol

- High order ODEs

quadrature,	missing,	exact_linear,	exact_nonlinear,	reducible,
linear_ODEs				

2.1.2 Example 2.1. A Riccati equation

Consider the following first-order nonlinear equation.

```
> with(DEtools):
> ode[1] := x*diff(y(x),x)+a*y(x)^2-y(x)+b*x^2;
```

$$ode_1 := x\,\frac{d}{dx}\,\mathrm{y}(x) + a\,\mathrm{y}(x)^2 - \mathrm{y}(x) + b\,x^2$$

Ask for information about the ODE.

```
> odeadvisor(ode[1]);
```

$$[[\_homogeneous,\ class\ D],\ \_rational,\ \_Riccati]$$

Solve the ODE. Note: It is not necessary to invoke `odeadvisor` prior to invoking `dsolve`.

```
> dsolve(ode[1], y(x));
```

$$\mathrm{y}(x) = -\frac{\tan\left(x\,\sqrt{b\,a} + \_C1\,\sqrt{b\,a}\right)\,x\,\sqrt{b\,a}}{a}$$

2.1.3 Example 2.2. An Abel equation

This is another first-order nonlinear equation.

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

$$ode_2 := (2\,\mathrm{y}(x) - x)\,\frac{d}{dx}\,\mathrm{y}(x) - \mathrm{y}(x) - 2\,x$$

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

$$[[\_homogeneous,\ class\ A],\ \_exact,\ \_rational,\ [\_Abel,\ 2nd\ type,\ class\ A]]$$

```
> dsolve(ode[2], y(x));
```

$$\mathrm{y}(x) = \frac{\frac{x\,\_C1}{2} + \frac{\sqrt{5\,x^2\,\_C1^2 + 4}}{2}}{\_C1},\ \mathrm{y}(x) = \frac{\frac{x\,\_C1}{2} - \frac{\sqrt{5\,x^2\,\_C1^2 + 4}}{2}}{\_C1}$$

2.1.4 Example 2.3. A Bessel equation

Consider the following second-order linear equation.

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

$$ode_3 := x^2 \frac{d^2}{dx^2} \mathrm{y}(x) + x \frac{d}{dx} \mathrm{y}(x) - (x^2 + n^2) \mathrm{y}(x)$$

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

$$[[\_Bessel, \_modified]]$$

```
> dsolve(ode[3], y(x));
```

$$\mathrm{y}(x) = \_C1 \, \mathrm{BesselI}(n, x) + \_C2 \, \mathrm{BesselK}(n, x)$$

2.1.5 Example 2.4. A Van der Pol equation

The following second-order nonlinear equation can be classified, but the general symbolic solution can only be expressed in an implicit form.

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

$$ode_4 := \frac{d^2}{dx^2} \mathrm{y}(x) - \mu (1 - \mathrm{y}(x)^2) \frac{d}{dx} \mathrm{y}(x) + \mathrm{y}(x) = 0$$

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

$$[[\_2nd\_order, \_missing\_x], \_Van\_der\_Pol]$$

Numerical solutions of Van der Pol's equation are considered in Example 4.4.

2.1.6 Example 2.5. Solving via Lie symmetry methods

```
> ode[5] := diff(y(x),x) = F((y(x)-x*ln(x))/x) + ln(x);
```

$$ode_5 := \frac{d}{dx} \mathrm{y}(x) = \mathrm{F}\left(\frac{\mathrm{y}(x) - x \ln(x)}{x}\right) + \ln(x)$$

Lie symmetry methods [2] are used to express the solution, in this case in an implicit form in terms of Maple's `RootOf` construct.

```
> dsolve(ode[5], y(x));
```

$$\mathrm{y}(x) = \mathrm{RootOf}\left(\ln(x) - \int^{\_Z} \frac{1}{-\_a - 1 + \mathrm{F}(\_a)}\, d\_a - \_C1\right) x + x\ln(x)$$

For the following particular choice of the function F, the above ODE is a Riccati equation.

```
> F := u -> u^2:
> ode[5];
```

$$\frac{d}{dx}\mathrm{y}(x) = \frac{(\mathrm{y}(x) - x\ln(x))^2}{x^2} + \ln(x)$$

```
> odeadvisor(ode[5], y(x));
```

$$[[\_1st\_order,\ \_with\_linear\_symmetries],\ \_Riccati]$$

Request that Lie symmetry methods be used to compute the solution.

```
> dsolve(ode[5], y(x), 'Lie');
```

$$\mathrm{y}(x) = \frac{1}{10}\, x\, (2\sqrt{5}\ln(x) + \sqrt{5} - 5\tanh\left(\frac{1}{2}\left(\ln(x) + \_C1\right)\sqrt{5}\right)\right)\sqrt{5}$$

Show the symmetry generators (a set of infinitesimals which leave the ODE invariant).

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

$$[\_\xi = x,\ \_\eta = x + y]$$

2.2 Hypergeometric Solutions of Second-order Linear ODEs

Maple's `dsolve` can resolve the *equivalence* between a given ODE and one having ${}_2F_1, {}_1F_1$ or ${}_0F_1$ hypergeometric solutions, whenever that equivalence involves power composed with Möbius transformations (see the help page `?dsolve,hypergeometric`).

2.2.1 Example 2.6. ${}_2F_1$ hypergeometric type of solution

```
> PDEtools[declare](y(x), prime=x);
```

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

$$\textit{derivatives with respect to}:\ x \textit{ of functions of one variable will now be displayed with }'$$

```
> ode := diff(y(x),x,x) =
  3/16*(x^2+10)*(4*x^6+16*x^4+23*x^2-10)/(x^2+2)^2/(x^2+5)^2/x^2*y(x);
```

$$ode := y'' = \frac{3\,(x^2+10)\,(4\,x^6+16\,x^4+23\,x^2-10)\,y}{16\,(x^2+2)^2\,(x^2+5)^2\,x^2}$$

```
> sol := dsolve(ode);
```

$$sol := y = \_C1 \operatorname{hypergeom}\left(\left[\frac{1}{2}, \frac{1}{2}\right], \left[\frac{3}{4}\right], \frac{2\,(x^2+5)}{5\,(x^2+2)}\right) x^{(3/4)}\,(x^2+5)^{(3/8)}$$

$$+ \frac{\_C2 \operatorname{hypergeom}\left(\left[\frac{3}{4}, \frac{3}{4}\right], \left[\frac{5}{4}\right], \frac{2\,(x^2+5)}{5\,(x^2+2)}\right) x^{(3/4)}\,(-x^2-5)^{(1/4)}\,(x^2+5)^{(3/8)}}{(x^2+2)^{(1/4)}}$$

2.2.2 Example 2.7. ${}_1F_1$ hypergeometric type of solution

```
> ode := diff(y(x),x,x) = 27*x*y(x)/(2*x+1)/(x-1)^4;
```

$$ode := y'' = \frac{27\,x\,y}{(1+2\,x)\,(x-1)^4}$$

```
> sol := dsolve(ode);
```

$$sol := y = \_C1\ (1+2\,x)\,e^{\left(\frac{-1-2\,x}{x-1}\right)} \operatorname{KummerU}\left(\frac{3}{2}, 2, \frac{2+4\,x}{x-1}\right)$$

$$+ \_C2\ (1+2\,x)\,e^{\left(\frac{-1-2\,x}{x-1}\right)} \operatorname{hypergeom}\left(\left[\frac{3}{2}\right], [2], \frac{2+4\,x}{x-1}\right)$$

```
> convert(sol, StandardFunctions);
```

$$y = \_C1\ (1+2\,x)\,e^{\left(\frac{-1-2\,x}{x-1}\right)} \operatorname{KummerU}\left(\frac{3}{2}, 2, \frac{2+4\,x}{x-1}\right) + \_C2\ (1+2\,x)\,e^{\left(\frac{-1-2\,x}{x-1}\right)}$$

$$\left(e^{\left(\frac{2+4\,x}{2\,(x-1)}\right)} \operatorname{BesselI}(0, \frac{2+4\,x}{2\,(x-1)}) + e^{\left(\frac{2+4\,x}{2\,(x-1)}\right)} \operatorname{BesselI}\left(1, \frac{2+4\,x}{2\,(x-1)}\right)\right)$$

2.2.3 Example 2.8. ${}_0F_1$ hypergeometric type of solution

```
> ode := diff(y(x),x,x) =
  1/20/x^2*(405*x^6-5670*x^4+58604*x^2+13720)/(3*x^2-14)^3*y(x);
```

$$ode := y'' = \frac{(405\,x^6 - 5670\,x^4 + 58604\,x^2 + 13720)\,y}{20\,x^2\,(3\,x^2 - 14)^3}$$

```
> sol := dsolve(ode);
```

$$sol := y = \_C1\,\sqrt{9\,x^2 - 42}\,\mathrm{hypergeom}\left([],\,[1],\,\frac{14\,x^2}{5\,(3\,x^2 - 14)}\right)\sqrt{x}$$
$$+\,\_C2\,\sqrt{x}\,\sqrt{9\,x^2 - 42}\,\mathrm{BesselK}\left(0,\,\frac{2\sqrt{\dfrac{x^2}{3\,x^2 - 14}}\,\sqrt{70}}{5}\right)$$

```
> convert(sol, Bessel);
```

$$y = \_C1\,\sqrt{9\,x^2 - 42}\,\mathrm{BesselJ}\left(0,\,\frac{28\,x}{\sqrt{-210\,x^2 + 980}}\right)\sqrt{x}$$
$$+\,\_C2\,\sqrt{x}\,\sqrt{9\,x^2 - 42}\,\mathrm{BesselK}\left(0,\,\frac{2\sqrt{\dfrac{x^2}{3\,x^2 - 14}}\,\sqrt{70}}{5}\right)$$

An example with fractional power coefficients, whose solution can be expressed in elementary form.

```
> ode := diff(y(x),x,x) = 1/4*(2/3*1/(x^(5/6))+2/x^(2/3))/x*y(x) +
  1/2*(-1+x^(1/6))/x*diff(y(x),x);
```

$$ode := y'' = \frac{\left(\dfrac{2}{3\,x^{(5/6)}} + \dfrac{2}{x^{(2/3)}}\right)y}{4\,x} + \frac{\left(-1 + x^{(1/6)}\right)y'}{2\,x}$$

```
> sol := dsolve(ode);
```

$$sol := y = \_C1\,e^{(-3\,x^{(1/6)})} + \_C2\,\sqrt{x}\,\mathrm{hypergeom}\left([3],\,[4],\,9\,x^{(1/6)}\right)\,e^{(-3\,x^{(1/6)})}$$

```
> convert(sol, elementary);
```

$$y = \_C1\,e^{(-3\,x^{(1/6)})} - \frac{2}{243}\,\_C2\left(1 - e^{(9\,x^{(1/6)})}\left(1 - 9\,x^{(1/6)} + \frac{81\,x^{(1/3)}}{2}\right)\right)e^{(-3\,x^{(1/6)})}$$

2.3 Formal Power Series Solutions

There are algorithms in Maple to compute various types of formal power series solutions [1] for a linear ODE with polynomial coefficients. The following examples will serve to illustrate.

2.3.1 Example 2.9. Formal power series with polynomial coefficients

```
> ode := (3*x^2-6*x+3)*diff(y(x),x,x) + (12*x-12)*diff(y(x),x) + 6*y(x);
```

$$ode := (3\,x^2 - 6\,x + 3)\,\frac{d^2}{dx^2}\,\mathrm{y}(x) + (12\,x - 12)\,\frac{d}{dx}\,\mathrm{y}(x) + 6\,\mathrm{y}(x)$$

```
> dsolve(ode, y(x), formal_series, coeffs=polynomial);
```

$$\mathrm{y}(x) = \sum_{n=0}^{\infty} (\_C1 + n\,\_C2)\,x^n$$

2.3.2 Example 2.10. Formal power series with hypergeometric coefficients

```
> ode := 2*x*(x-1)*diff(y(x),x,x) + (7*x-3)*diff(y(x),x) + 2*y(x) = 0;
```

$$ode := 2\,x\,(x-1)\,\frac{d^2}{dx^2}\,\mathrm{y}(x) + (7\,x - 3)\,\frac{d}{dx}\,\mathrm{y}(x) + 2\,\mathrm{y}(x) = 0$$

```
> dsolve(ode, y(x), type=formal_series, coeffs=hypergeom);
```

$$\mathrm{y}(x) = \frac{\_C1\left(\sum_{n=0}^{\infty} \frac{\Gamma\left(n+\frac{1}{2}\right)\,(x+1)^n}{n!}\right)}{\sqrt{\pi}},\ \mathrm{y}(x) = \_C1\left(\sum_{n=0}^{\infty}\frac{(n+1)\,x^n}{2\,n+1}\right),$$

$$\mathrm{y}(x) = \frac{\_C1\left(\sum_{n=0}^{\infty} \frac{(-1)^n\,\Gamma\left(n+\frac{1}{2}\right)\,(x-1)^n}{\Gamma(n+1)}\right)}{\sqrt{\pi}}$$

2.3.3 Example 2.11. Formal m-sparse m-hypergeometric power series

```
> ode := diff(y(x),x,x) + (x-1)*y(x);
```

$$ode := \frac{d^2}{dx^2}\,\mathrm{y}(x) + (x-1)\,\mathrm{y}(x)$$

```
> dsolve(ode, y(x), type=formal_series, coeffs=mhypergeom);
```

$$\mathrm{y}(x) = \frac{2}{9}\,\frac{\_C1\,\pi\,\sqrt{3}\left(\displaystyle\sum_{n=0}^{\infty}\frac{\left(-\frac{1}{3}\right)^n (x-1)^{(3\,n+1)}}{3^n\,\Gamma\left(\frac{4}{3}+n\right)\,\Gamma(n+1)}\right)}{\Gamma\left(\frac{2}{3}\right)},$$

$$\mathrm{y}(x) = \_C1\,\Gamma\left(\frac{2}{3}\right)\left(\sum_{n=0}^{\infty}\frac{\left(-\frac{1}{3}\right)^n (x-1)^{(3\,n)}}{\Gamma(n+1)\,3^n\,\Gamma\left(n+\frac{2}{3}\right)}\right)$$

2.4 The DEtools Package

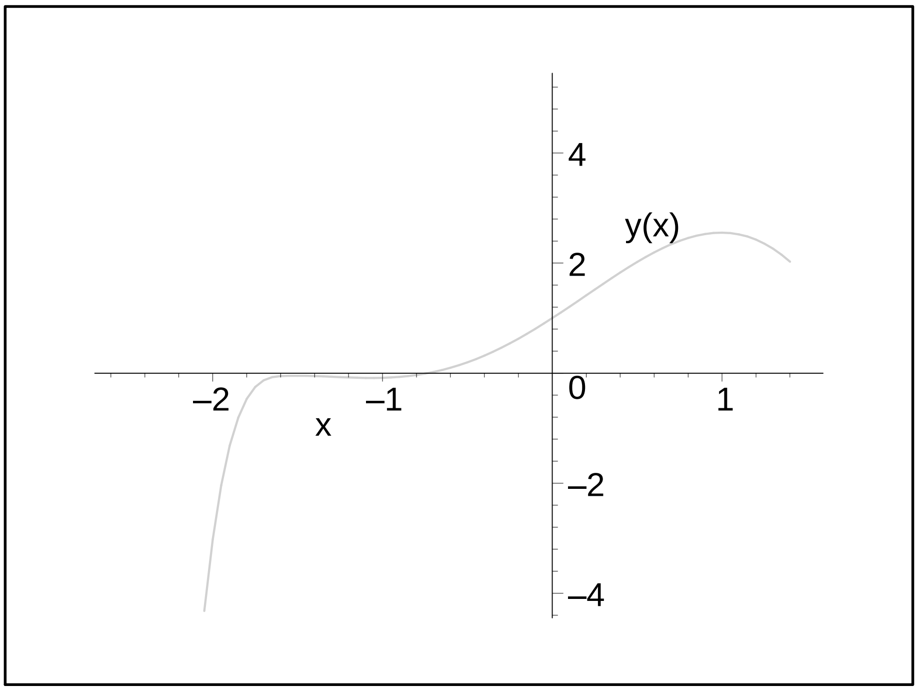


Figure 3: DEplot for an IVP.

The `DEtools` package provides user tools for manipulating ODEs including:

- functions for visualization
  - e.g. `DEplot`, `dfieldplot`, `phaseportrait`
- general manipulation of differential equations
  - e.g. `Dchangevar`, `convertsys`, `indicialeq`, `reduceOrder`
- functions for constructing closed form solutions

  – e.g. `constcoeffsols`, `exactsol`, `expsols`, `linearsol`, `ratsols`, `separablesol`

- functions for differential operators

  – e.g. `DFactor`, `adjoint`, `formal_sol`

Some examples using the functions for visualization are presented below.

2.4.1 Example 2.12. DEplot

```
> with(DEtools):
> de1 :=
  cos(x)*diff(y(x),x$3)-diff(y(x),x$2)+Pi*diff(y(x),x)=y(x)-x;
```

$$de1 := \cos(x)\,\frac{d^3}{dx^3}\,\mathrm{y}(x) - \frac{d^2}{dx^2}\,\mathrm{y}(x) + \pi\,\frac{d}{dx}\,\mathrm{y}(x) = \mathrm{y}(x) - x$$

```
> DEplot(de1, y(x), x=-2.5..1.4,
  [[y(0)=1,D(y)(0)=2,(D@@2)(y)(0)=1]], y=-4..5, stepsize=.05);  # Figure 3
```

2.4.2 Example 2.13. dfieldplot

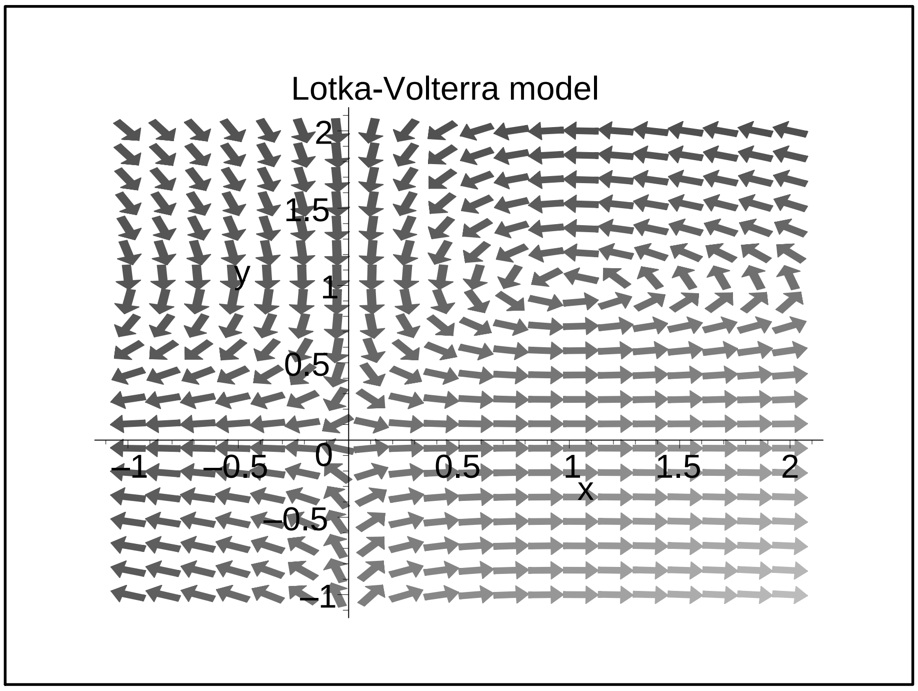


Figure 4: dfieldplot: Lotka-Volterra model.

```
> de2a := diff(x(t),t) = x(t)*(1-y(t));
  de2b := diff(y(t),t) = .3*y(t)*(x(t)-1);
```

$$\mathit{de2a} := \frac{d}{dt}\,\mathrm{x}(t) = \mathrm{x}(t)\,(1 - \mathrm{y}(t))$$

$$\mathit{de2b} := \frac{d}{dt}\,\mathrm{y}(t) = 0.3\,\mathrm{y}(t)\,(\mathrm{x}(t) - 1)$$

```
> dfieldplot([de2a,de2b], [x(t),y(t)],
  t=-2..2, x=-1..2, y=-1..2,
  arrows=LARGE, title="Lotka-Volterra model",
  color=[.3*y(t)*(x(t)-1),x(t)*(1-y(t)),.1]);  # Figure 4
```

```
> de3 := diff(y(x),x) = 1/2*(-x-(x^2+4*y(x))^(1/2));
```

$$\mathit{de3} := \frac{d}{dx}\,\mathrm{y}(x) = -\frac{x}{2} - \frac{1}{2}\,\sqrt{x^2 + 4\,\mathrm{y}(x)}$$

```
> dfieldplot(de3, y(x), x=-3..3, y=-3..2,
  title="Restricted domain", color=1/2*(-x-(x^2+4*y)));  # Figure 5
```

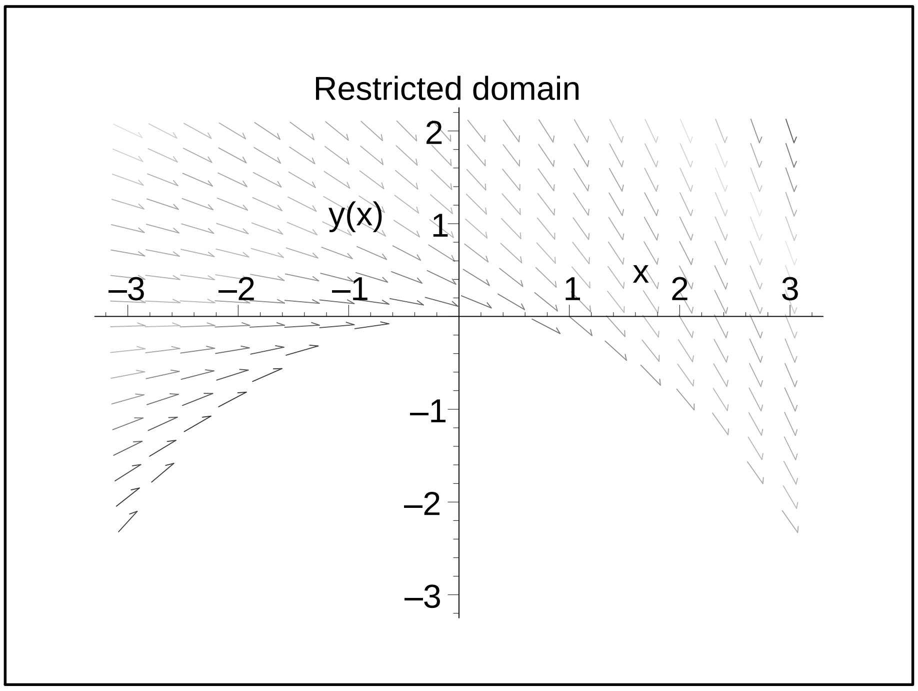


Figure 5: dfieldplot: Restricted domain.

2.4.3 Example 2.14. phaseportrait

```
> de4a := D(x)(t) = y(t)-z(t);
  de4b := D(y)(t) = z(t)-x(t);
  de4c := D(z)(t) = x(t)-y(t)*2;
```

$$de4a := \mathrm{D}(x)(t) = \mathrm{y}(t) - \mathrm{z}(t)$$

$$de4b := \mathrm{D}(y)(t) = \mathrm{z}(t) - \mathrm{x}(t)$$

$$de4c := \mathrm{D}(z)(t) = \mathrm{x}(t) - 2\,\mathrm{y}(t)$$

```
> phaseportrait([de4a,de4b,de4c], [x(t),y(t),z(t)],
  t=-2..2, [[x(0)=1,y(0)=0,z(0)=2]], stepsize=.05,
  scene=[z(t),x(t)], linecolour=sin(t*Pi/2),
  method=classical[foreuler]);                     # Figure 6
```

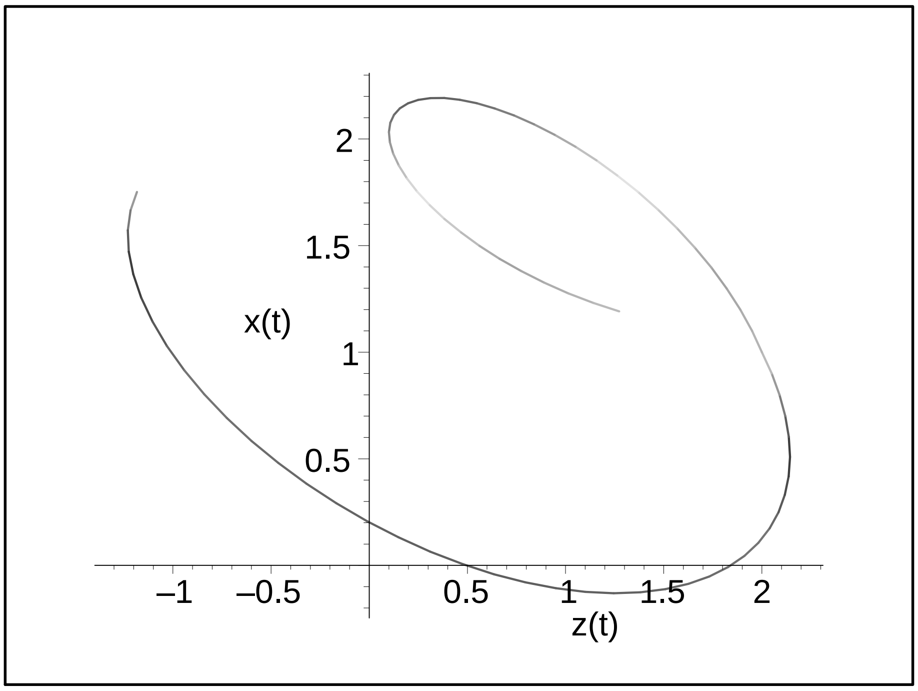


Figure 6: An example of phaseportrait.

```
> de5 := D(y)(x) = -y(x)-x^2;
```

$$de5 := \mathrm{D}(y)(x) = -\mathrm{y}(x) - x^2$$

```
> phaseportrait(de5, y(x), x=-1..2.5,
  [[y(0)=0],[y(0)=1],[y(0)=-1]],
  title="Asymptotic solution",colour=magenta,
  linecolor=[gold,yellow,wheat]);              # Figure 7
```

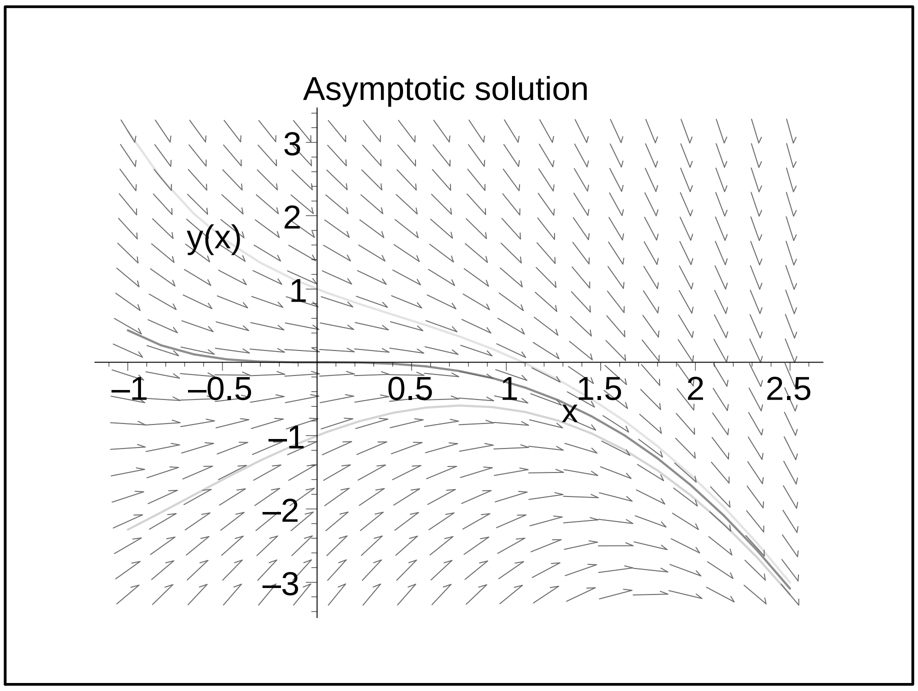


Figure 7: phaseportrait: Asymptotic solution.

3 Hybrid Symbolic-Numeric Integration

The Maple scientific computation environment supports both symbolic and numeric mathematical computation, as previous sections have illustrated. In this section a new paradigm, *hybrid symbolic-numeric computation*, is employed to achieve an enhanced level of computational power. The overall strategy is to apply an appropriate combination of symbolic mathematical analysis and numerical computation.

The hybrid symbolic-numeric methods illustrated here are part of an automated numerical integration strategy [3], [4] implemented in Maple for the numerical evaluation of definite integrals. In particular, symbolic analysis techniques are used to deal with integrand singularities.

3.1 Example 3.1. Finite and infinite interval of integration

```
> f := x^2*ln(x)*exp(-x^2);
```

$$f := x^2 \ln(x)\, e^{(-x^2)}$$

```
> plot( f, x=0..4 );  # Figure 8
```

Form the integral on $(0, 4)$ and attempt to evaluate in symbolic mode.

```
> intf := Int( f, x=0..4 ):
> value( intf );
```

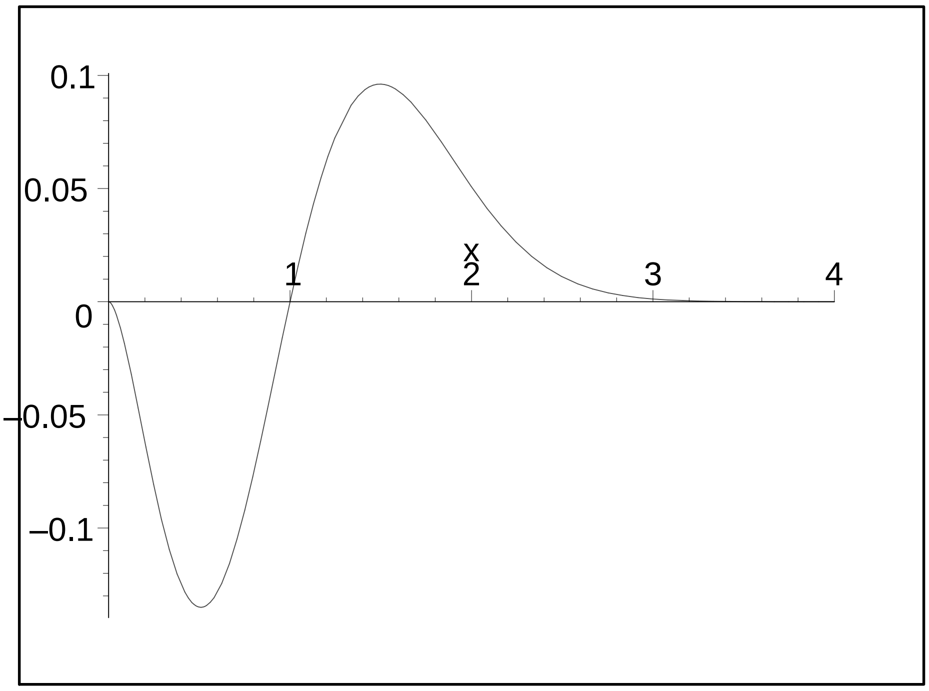


Figure 8: Graph of integrand for Example 3.1.

$$\int_0^4 x^2 \ln(x)\, e^{(-x^2)}\, dx$$

The integral does not evaluate in symbolic mode. Apply direct numerical integration (at standard precision).

```
> evalf( intf );  # Invokes compiled NAG routines.
```

$$0.008084270850$$

Or at higher precision.

```
> evalf[25]( intf );
```

$$0.008084270849929287342315724$$

Notice from the graph that the integral on $(0, \infty)$ will have approximately the same numerical value. Curiously, for the infinite integral a closed form expression can be obtained!

```
> newintf := Int( f, x=0..infinity ):
> newintf = value( newintf );
```

$$\int_0^\infty x^2 \ln(x)\, e^{(-x^2)}\, dx = \frac{\sqrt{\pi}}{4} - \frac{\sqrt{\pi}\,\gamma}{8} - \frac{1}{4}\sqrt{\pi}\ln(2)$$

Of course, this symbolic formula can be evaluated numerically.

```
> evalf( rhs(%) );
```

$$0.0080845993$$

Or, we may apply direct numerical integration to the infinite integral.

```
> evalf( newintf );  # Invokes compiled NAG routines.
```

$$0.008084599362$$

Note: Numerical evaluation of the exact formula loses some precision. The result from numerical integration is correct to 10 significant digits.

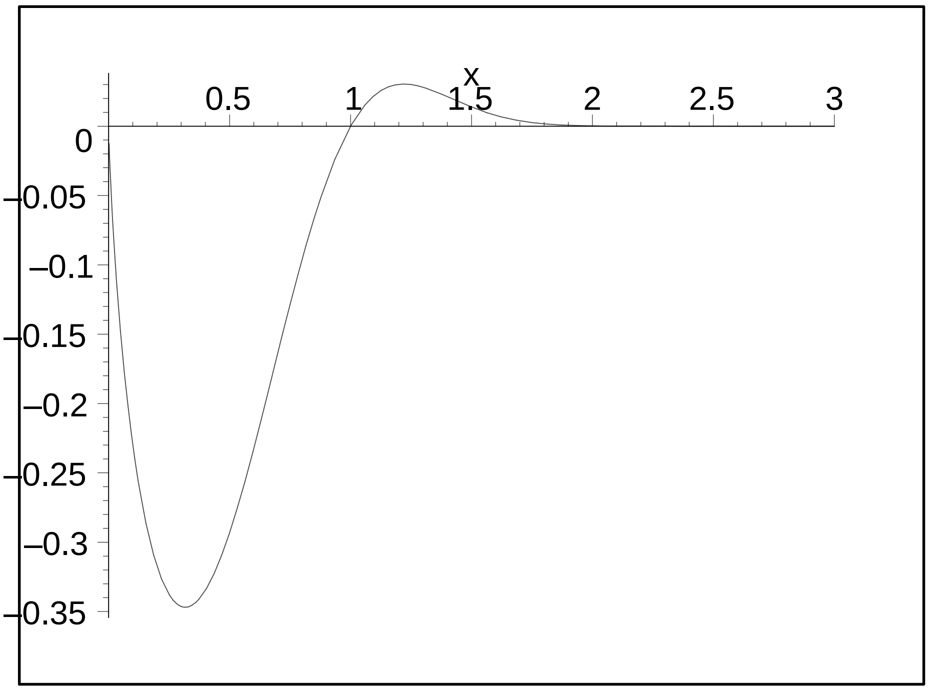


Figure 9: Graph of integrand for Example 3.2.

3.2 Example 3.2. Logarithmic singularity at left endpoint

```
> g := sin(x)*ln(x)*exp(-x^3);
```

$$g := \sin(x)\ln(x)\,e^{(-x^3)}$$

```
> plot( g, x=0..3 );  # Figure 9
```

Form the integral on $(0, 3)$, attempt symbolic evaluation, and apply numerical integration.

```
> intg := Int( g, x=0..3 ):
> value( intg );
```

$$\int_0^3 \sin(x)\ln(x)\,e^{(-x^3)}\,dx$$

```
> evalf( intg );
```

$$-0.1957885158$$

Evaluating numerically to 25 digits yields the following result.

```
> evalf[25]( intg );
```

$$-0.1957885158488077265323200$$

Note that the integrand has a logarithmic singularity at x=0 as the following series expansion shows.

```
> series( g, x=0 );
```

$$\ln(x)\,x - \frac{1}{6}\ln(x)\,x^3 - \ln(x)\,x^4 + \frac{1}{120}\ln(x)\,x^5 + \mathrm{O}(x^6)$$

The hybrid symbolic-numeric technique includes the concept of term-by-term integration of such a series expansion.

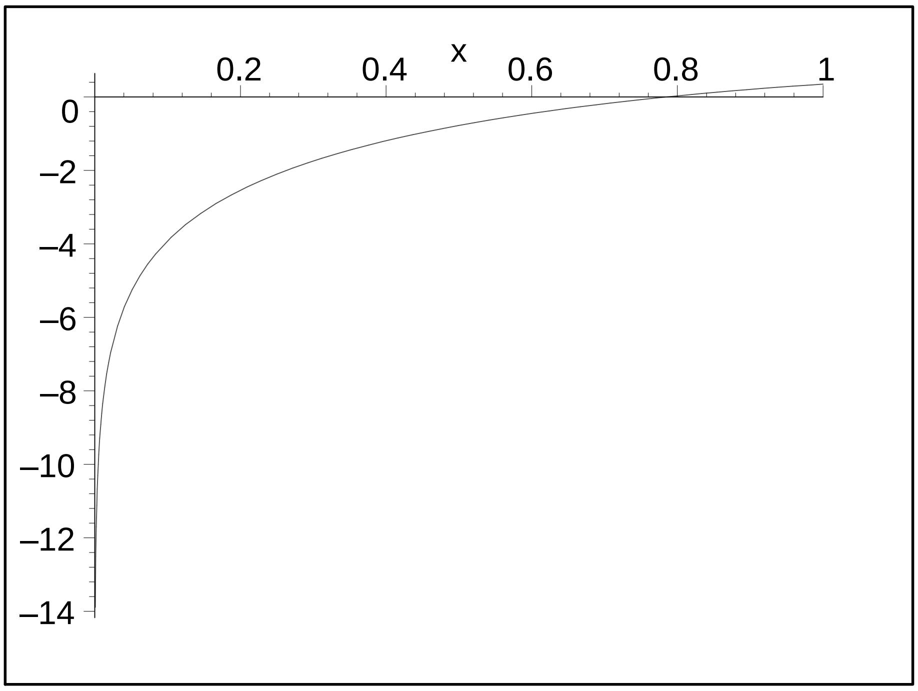


Figure 10: Integrand with logarithmic singularity.

3.3 Example 3.3. Subtracting off a singularity

```
> h := ln(1 - cos(2*x)):
> Int(h, x=0..1);
```

$$\int_0^1 \ln(1 - \cos(2\,x))\,dx$$

Note that the graph of this integrand goes to $-\infty$ at $x = 0$. However, this is an integrable singularity – it behaves like $\ln(x)$ near $x = 0$.

```
> plot(h, x=0..1);  # Figure 10
```

Note: The technique of *subtracting off a singularity* illustrated below, takes place automatically within Maple's numerical integration routines.

Suppose it is desired to compute the result to 25 digits of accuracy. The generalized series expansion of h at $x = 0$ takes the following form.

```
> Digits := 25:
> series(h, x=0, 8);
```

$$(\ln(2) + 2\ln(x)) - \frac{1}{3}\,x^2 - \frac{1}{90}\,x^4 - \frac{2}{2835}\,x^6 + \mathrm{O}(x^8)$$

The non-regular part is

```
> q := 2*ln(x);
```

$$q := 2\ln(x)$$

The new expression

```
> newh := h - q;
```

$$newh := \ln(1 - \cos(2x)) - 2\ln(x)$$

is analytic on the interval [0, 1]. Thus it can be integrated easily by the default numerical integration method.

```
> r1 := evalf(Int(newh, x=0..1));
```

$$r1 := 0.57970676857677544431529296$$

Integrating q is easy because it has the indefinite integral

```
> int(q, x);
```

$$2x\ln(x) - 2x$$

and its definite integral can therefore be computed symbolically.

```
> r2 := int(q, x=0..1);
```

$$r2 := -2$$

Finally, summing the two values, we obtain the value for the original definite integration problem.

```
> Int(h, x=0..1)  =  r1 + r2;
```

$$\int_0^1 \ln(1 - \cos(2x))\,dx = -1.42029323142322455684707 0$$

3.4 Example 3.4. Algebraic transformation of variables

```
> F := sqrt(sin(x)):
> Int(F, x=0..2);
```

$$\int_0^2 \sqrt{\sin(x)}\,dx$$

Note: The *change of variables* illustrated below takes place automatically within Maple's numerical integration routines.

The generalized series expansion of the integrand F at $x = 0$ is of the form

```
> series(F, x=0, 5);
```

$$\sqrt{x} - \frac{x^{(5/2)}}{12} + \mathrm{O}(x^{(9/2)})$$

Applying the change of variables $t = \sqrt{x}$ yields

```
> r3 := Int( eval(F, x=t^2) * diff(t^2, t), t = 0..sqrt(2) );
```

$$r3 := \int_0^{\sqrt{2}} 2\sqrt{\sin(t^2)}\,t\,dt$$

The new integrand is analytic on the interval of integration. Therefore it can be integrated easily by the default numerical integration method, even at high accuracy. Suppose that the result is desired to 50 digits of accuracy.

```
> evalf[50](r3);
```

$$1.6207234083199665050793106079159335362211371877592$$

3.5 Example 3.5. Integrating a series term-by-term

```
> G := exp(v - v^2/2) / (1 + 1/2*exp(v)):
> Int(G, v=0..infinity);
```

$$\int_0^\infty \frac{e^{(v-1/2\,v^2)}}{1+\frac{1}{2}e^v}\,dv$$

Note: The development of the integrand in a generalized series expansion and its integration term-by-term as illustrated below takes place automatically within Maple's numerical integration routines.

Suppose it is desired to compute the integral to 20 digits of accuracy. First, the interval of integration is split into 0..1 and 1..∞. For the finite interval, the default numerical integration method has no difficulty.

```
> Digits := 20:
> r01 := evalf(Int(G, v=0..1));
```

$$\mathit{r01} := 0.75805648290182466780$$

For the infinite interval, the change of variables $v = \frac{1}{x}$ transforms the problem into the following new integration problem.

```
> r1inf := Int( -eval(G, v=1/x) * diff(1/x, x), x = 0..1  );
```

$$\mathit{r1inf} := \int_0^1 \frac{e^{\left(\frac{1}{x}-\frac{1}{2\,x^2}\right)}}{\left(1+\frac{1}{2}\,e^{\left(\frac{1}{x}\right)}\right)\,x^2}\,dx$$

Let the integrand appearing here be named g. The first few terms of the generalized series expansion of g are as follows.

```
> g := op(1, r1inf):
> n := 6:
> s := `evalf/int/genseries`(g, x, n);
```

$$s := \frac{2\sqrt{e^{\left(-\frac{1}{x^2}\right)}}}{x^2} - \frac{4\sqrt{e^{\left(-\frac{1}{x^2}\right)}}\,e^{\left(-\frac{1}{x}\right)}}{x^2} + \frac{8\sqrt{e^{\left(-\frac{1}{x^2}\right)}}\,(e^{\left(-\frac{1}{x}\right)})^2}{x^2} - \frac{16\sqrt{e^{\left(-\frac{1}{x^2}\right)}}\,(e^{\left(-\frac{1}{x}\right)})^3}{x^2}$$
$$+\frac{32\sqrt{e^{\left(-\frac{1}{x^2}\right)}}\,(e^{\left(-\frac{1}{x}\right)})^4}{x^2} - \frac{64\sqrt{e^{\left(-\frac{1}{x^2}\right)}}\,(e^{\left(-\frac{1}{x}\right)})^5}{x^2} + \mathrm{O}((e^{\left(-\frac{1}{x}\right)})^6)$$

Note that s is a series in increasing powers of $e^{-\frac{1}{x}}$. Maple's symbolic integrator determines that the first two terms of s have the following indefinite integrals.

```
> terms := [seq(simplify(op(i,s),symbolic), i=1..n)]:
> int(terms[1], x);
```

$$-\sqrt{\pi}\,\sqrt{2}\,\mathrm{erf}\left(\frac{\sqrt{2}}{2\,x}\right)$$

```
> int(terms[2], x);
```

$$2\sqrt{\pi}\,e^{(1/2)}\,\sqrt{2}\,\mathrm{erf}\left(\frac{\sqrt{2}}{2\,x}+\frac{\sqrt{2}}{2}\right)$$

Similarly, the integral of each term can be computed symbolically. If we compute the definite integral over [0, 0.25] of the successive terms of the generalized series expansion of g, we find that the successive values are as follows. For the numerical evaluation of the symbolic formula for each term, and for the addition of the terms to get an accurate floating point result, it is generally necessary to set the working precision higher for this stage of the computation.

```
> Digits := 2*Digits:
> seq( int(terms[i], x=0..0.25), i = 1..n ):
> evalf( [ % ] );
```

$$\begin{aligned}&[0.00015877606054314910751237036266503283736 37,\\ &-0.47386157552545686895055017987048468418 32\,10^{-5},\\ &0.14618558751626870771092071306440237944\,10^{-6},\\ &-0.46204199238888121408794721890945 76\,10^{-8},\\ &0.14874871295997595129888742403 91\,10^{-9},\\ &-0.48583244697608264682314096 9\,10^{-11}]\end{aligned}$$

Clearly, the series representation is converging reasonably quickly on this interval (the magnitude of successive terms is decreasing at a good rate).

To obtain a numerical result accurate to 20 digits, it is necessary to sum more than six terms of the series as can be seen from the magnitude of the terms above. For this example, 20 terms is more than necessary.

```
> n := 20:
> s := `evalf/int/genseries`(g, x, n):
> terms := [seq(simplify(op(i,s),symbolic), i=1..n)]:
```

The following loop adds successive terms until the magnitude of the terms becomes negligible for 20 digit accuracy.

```
> Digits := 2*Digits:
> epsilon := 1e-20:
> t := evalf( int(terms[1], x=0..0.25) ):  val := t:
> for i from 2 to n while abs(t/val) > epsilon do
    t := evalf( int(terms[i], x=0..0.25) );
    val := val + t
  end do:
> number_of_terms := i-1;
```

$$number\_of\_terms := 15$$

We now have the following result for the integral of g over [0, 0.25].

```
> Digits := 20:
> r1 := evalf(val);
```

$$r1 := 0.00015417915400122159463$$

For the remaining interval [0.25, 1], ordinary numerical integration methods encounter no difficulties because there are no nearby singularities.

```
> r2 := evalf(Int(g, x=0.25..1));
```

$$r2 := 0.54730623706268017603$$

Add these two values to get the value of the integral *r1inf*.

```
> r1inf := r1 + r2;
```

$$r1inf := 0.54746041621668139762$$

Finally, we have obtained the answer for the original integration problem.

```
> Int(G, v=0..infinity) = r01 + r1inf;
```

$$\int_0^\infty \frac{e^{(v-1/2\,v^2)}}{1+\frac{1}{2}\,e^v}\,dv = 1.30551689911850606 54$$

3.6 Multidimensional Numerical Integration

The most significant improvement in numerical integration capabilities for Maple 8 is the addition of numerical methods for multiple integration. In previous releases, one could form a multiple integral using nested `Int` expressions, and then invoke the `evalf` command on the multiple integration problem. The only solution method was to invoke, recursively, one-dimensional numerical integration methods. This was an inefficient approach to numerical multiple integration problems which would succeed only on the simplest problems.

In Maple 8, compiled C routines which implement numerical multiple integration methods are automatically invoked for such problems whenever the desired precision is in hardware floating point range (typically about 15 decimal digits).

3.6.1 Special (list) syntax for multiple integrals

A numerical multiple integration problem can be specified in a natural way using nested one-dimensional integrals, for example:

```
evalf( Int(...(Int(Int(f, x1=a1..b1), x2=a2..b2), ...), xn=an..bn) )
```

where the integrand f depends on *x1*, *x2*, ..., *xn*. Such a problem can also be specified using the following special multiple integration notation with a list as the second argument:

```
evalf( Int(f, [x1=a1..b1, x2=a2..b2, ..., xn=an..bn]) .
```

Additional optional arguments can be stated just as in the case of one-dimensional integration. Also as in one-dimensional integration, the integrand f can be specified as a procedure in which case the second argument must be a list of ranges: `[a1..b1, a2..b2, ..., an..bn]` . Whether a multiple integration problem is stated using nested integrals or using the list notation, the arguments are extracted so as to invoke the same numerical multiple integration routines. See the help page `?evalf/int` for further details and examples.

3.6.2 Example 3.6. Multidimensional integration

Multiple integrals may be expressed as nested one-dimensional integrals.

```
> Int(Int(Int(exp(x+y+z)/((5*x+1)*(10*y+2)*(15*z+3)),
                         x=0..4), y=0..3), z=0..sqrt(2));
```

$$\int_0^{\sqrt{2}}\int_0^3\int_0^4 \frac{e^{(x+y+z)}}{(5\,x+1)\,(10\,y+2)\,(15\,z+3)}\,dx\,dy\,dz$$

```
> evalf(%);
```

$$0.9331611325$$

Numerical multiple integration may also be invoked using a list syntax.

```
> d := (1 - w^2*x^2*y^2*z^2):
  g := d * cos(w*x*y*z) - d * w*x*y*z * sin(w*x*y*z);
```

$$g := (1-w^2\,x^2\,y^2\,z^2)\cos(w\,x\,y\,z) - (1-w^2\,x^2\,y^2\,z^2)\,w\,x\,y\,z\sin(w\,x\,y\,z)$$

```
> DesiredIntegral := Int(Int(Int(Int(g, w=0..1),
                                 x=0..1), y=0..1), z=0..1);
```

$$\mathit{DesiredIntegral} :=$$

$$\int_0^1\int_0^1\int_0^1\int_0^1 (1-w^2\,x^2\,y^2\,z^2)\cos(w\,x\,y\,z) - (1-w^2\,x^2\,y^2\,z^2)\,w\,x\,y\,z\sin(w\,x\,y\,z)\,dw\,dx\,dy\,dz$$

Here we use a list syntax to give the integration command, which would avoid the need to form the nested integral above.

```
> evalf(Int(g, [w=0..1, x=0..1, y=0..1, z=0..1]));
```

$$0.9717798177$$

Equivalently, we could have requested numerical evaluation of the nested integral formed above.

```
> evalf(DesiredIntegral);
```

$$0.9717798177$$

When low accuracy is sufficient, the Monte Carlo method may be used.

```
> h := 1/(2 + sin(Pi*sqrt(87)*(x1+x2+x3+x4+x5+x6)));
```

$$h := \frac{1}{2+\sin(\pi\,\sqrt{87}\,(\mathit{x1}+\mathit{x2}+\mathit{x3}+\mathit{x4}+\mathit{x5}+\mathit{x6}))}$$

```
> evalf(Int(h, [x1=-1..1, x2=-1..1, x3=-1..1, x4=-1..1,
        x5=-1..1, x6=-1..1], method=_MonteCarlo, epsilon=0.5e-2)):
```

Only trust about 3 digits when `epsilon=0.5e-2` .

```
> evalf[3](%);
```

$$36.9$$

4 Numerical Solution of IVPs

One may consult the help page `?dsolve,numeric` for details about the numerical ODE solvers available in Maple. The examples below illustrate some of the capabilities for numerically solving initial-value problems.

4.1 Example 4.1. A Riccati equation

Consider the following Riccati equation *de* with initial condition *ic*.

```
> de := diff(y(x),x) = (y(x) - x*ln(x))^2/x^2 + ln(x);
```

$$de := \frac{d}{dx}\,\mathrm{y}(x) = \frac{(\mathrm{y}(x) - x\ln(x))^2}{x^2} + \ln(x)$$

```
> ic := y(1)=1/2;
```

$$ic := \mathrm{y}(1) = \frac{1}{2}$$

For this example, it is possible to compute an exact symbolic solution.

```
> soln := dsolve({de,ic}, y(x));
```

$$soln := \mathrm{y}(x) = \frac{1}{10}\,x\,\left(2\,\sqrt{5}\ln(x) + \sqrt{5} - 5\tanh\left(\frac{1}{2}\,\sqrt{5}\ln(x)\right)\right)\,\sqrt{5}$$

```
> y_exact := unapply(rhs(soln), x):
```

Compute a solution using the default numerical method which is a Runge-Kutta Fehlberg method `rkf45`. Look at the solution at some specific points.

```
> y_numeric := dsolve({de,ic}, type=numeric, range=1..4):
> y_numeric(1.5);
```

$$[x = 1.5,\ \mathrm{y}(x) = 0.646309187179596068]$$

```
> y_numeric(4.0);
```

$$[x = 4.0,\ \mathrm{y}(x) = 3.45869339601487890]$$

Comparing with the exact answer at $x = 4$, we see that the default numerical solution is accurate to about 5 digits.

```
> evalf[15]( y_exact(4) );
```

$$3.45866042407752$$

The numerical solution can be plotted efficiently because all necessary data has already been computed and stored in the procedure *y_numeric*.

```
> plots[odeplot](y_numeric);  # Figure 11
```

4.2 Example 4.2. A pendulum problem

```
> de := diff(x(t),t,t) = g/l*sin(x(t));
```

$$de := \frac{d^2}{dt^2}\mathrm{x}(t) = \frac{g\sin(\mathrm{x}(t))}{l}$$

```
> g := 32;  l := 2;
```

$$g := 32$$

$$l := 2$$

```
> inits := x(0)=0, D(x)(0)=1/10;
```

$$inits := \mathrm{x}(0) = 0,\ \mathrm{D}(x)(0) = \frac{1}{10}$$

For this example, the symbolic solution is expressed in a somewhat complicated implicit form which involves the `RootOf` construct and an unevaluated integral.

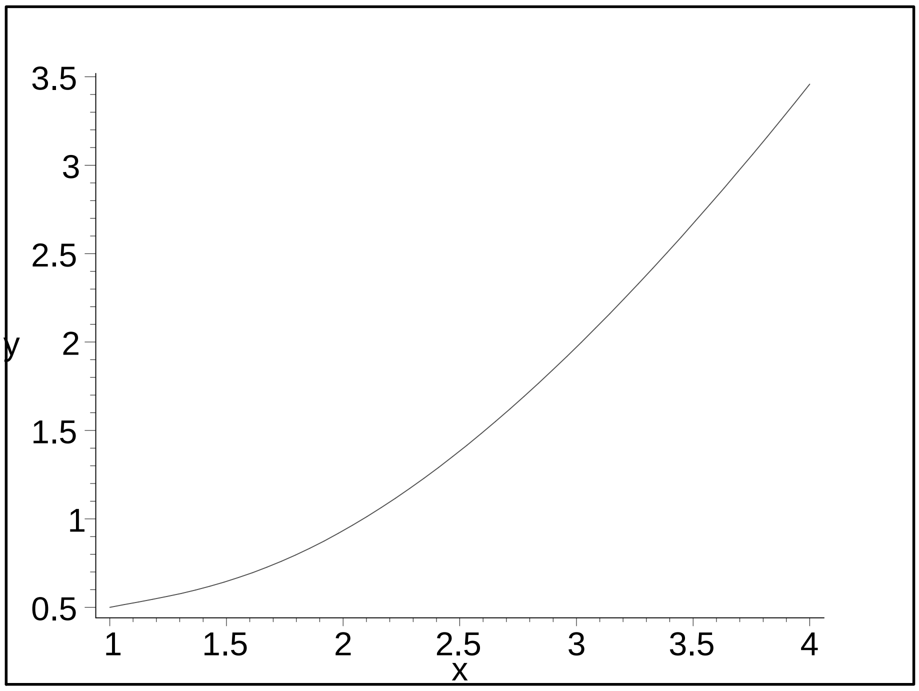


Figure 11: Numerical solution of a Riccati equation.

```
> soln := dsolve({de,inits}, x(t));
```

$$soln := \mathrm{x}(t) = \mathrm{RootOf}\left(-\int_0^{\_Z} \frac{10}{\sqrt{-3200\cos(\_f)+3201}}\, d\_f + t\right)$$

```
> x_exact := unapply(rhs(soln), t):
```

Computing numerical values of *x_exact* is quite inefficient because `fsolve` and `evalf/int` are repeatedly invoked!

```
> for tval from 0.0 by 0.2 to 0.6 do
    't' = tval, 'x_exact' = evalf( x_exact(tval) )
  end do;
```

$$t = 0.,\ x\_exact = 0.$$

$$t = 0.2,\ x\_exact = 0.02220259912$$

$$t = 0.4,\ x\_exact = 0.05938654256$$

$$t = 0.6,\ x\_exact = 0.1366102790$$

In this case, a numerical solution can be computed much more efficiently.

```
> x_numeric := dsolve({de,inits}, type=numeric, range=0..4):
> for tval from 0.0 by 0.2 to 0.6 do
    't' = tval, 'x_numeric' = eval(x(t), x_numeric(tval))
  end do;
```

$$t = 0.,\ x\_numeric = 0.$$

$$t = 0.2,\ x\_numeric = 0.0222025997741837761$$

$$t = 0.4,\ x\_numeric = 0.0593864864151269420$$

$$t = 0.6,\ x\_numeric = 0.136610116474530196$$

Plot the numerical solution.

```
> plots[odeplot](x_numeric);  Figure 12
```

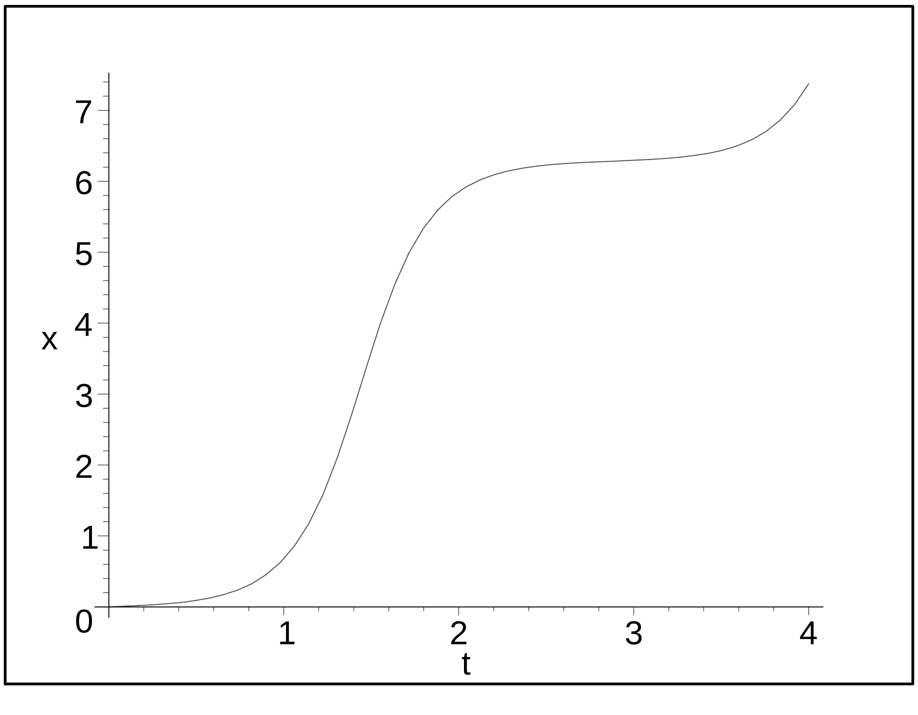


Figure 12: Numerical solution of a pendulum problem.

4.3 Example 4.3. Numerical solution as a piecewise-polynomial interpolant

The result computed by the numerical solver `rkf45` is a piecewise-polynomial interpolant. It is the interpolant which gets plotted by `odeplot`. For instructional purposes, we can request the Maple solver to return the piecewise-polynomial solution as the output. Here we restrict to a smaller range so that the output is not too large.

```
> x_pwpoly := dsolve({de,inits}, type=numeric, output=piecewise,
                                                      range=0..0.2):
```

Extract the x(t) solution and display it using 5-digit format.

```
> x_pwpoly := eval(x(t), x_pwpoly):
> evalf[5](x_pwpoly);
```

$$\begin{cases} \mathrm{Float(undefined)} & t < 0. \\ -0.85776\,10^{-5} + 0.10051\,t + 0.020216\,(t - 0.025238)^2 & \\ \quad +0.26830\,(t - 0.025238)^3 + 0.034011\,(t - 0.025238)^4 & t \le 0.050477 \\ -0.00029911 + 0.10545\,t + 0.066910\,(t - 0.082173)^2 & \\ \quad +0.28165\,(t - 0.082173)^3 + 0.097913\,(t - 0.082173)^4 & t \le 0.11387 \\ -0.0013632 + 0.11503\,t + 0.11369\,(t - 0.13540)^2 & \\ \quad +0.30695\,(t - 0.13540)^3 + 0.15776\,(t - 0.13540)^4 & t \le 0.15694 \\ -0.0031888 + 0.12658\,t + 0.15520\,(t - 0.17847)^2 & \\ \quad +0.33774\,(t - 0.17847)^3 + 0.21345\,(t - 0.17847)^4 & t \le 0.20000 \\ \mathrm{Float(undefined)} & \mathit{otherwise} \end{cases}$$

Compare values of *x_pwpoly* with *x_exact*.

```
> for tval from 0.0 by 0.1 to 0.2 do
     't' = tval, 'x_pwpoly' = eval(x_pwpoly, t=tval),
                 'x_exact' = evalf( x_exact(tval) )
  end do;
```

$$t = 0.,\ x\_pwpoly = 0.30\,10^{-15},\ x\_exact = 0.$$

$$t = 0.1,\ x\_pwpoly = 0.01026880738,\ x\_exact = 0.01026880675$$

$$t = 0.2,\ x\_pwpoly = 0.02220259336,\ x\_exact = 0.02220259912$$

Compute the piecewise-polynomial solution for `range=0..4` and plot it. Of course, it is the same plot as previously displayed.

```
> x_pwpoly := dsolve({de,inits}, type=numeric, output=piecewise,
                                                        range=0..4):
> x_pwpoly := eval(x(t), x_pwpoly):
> plot(x_pwpoly, t=0..4):  # Output suppressed; same as Figure 12.
```

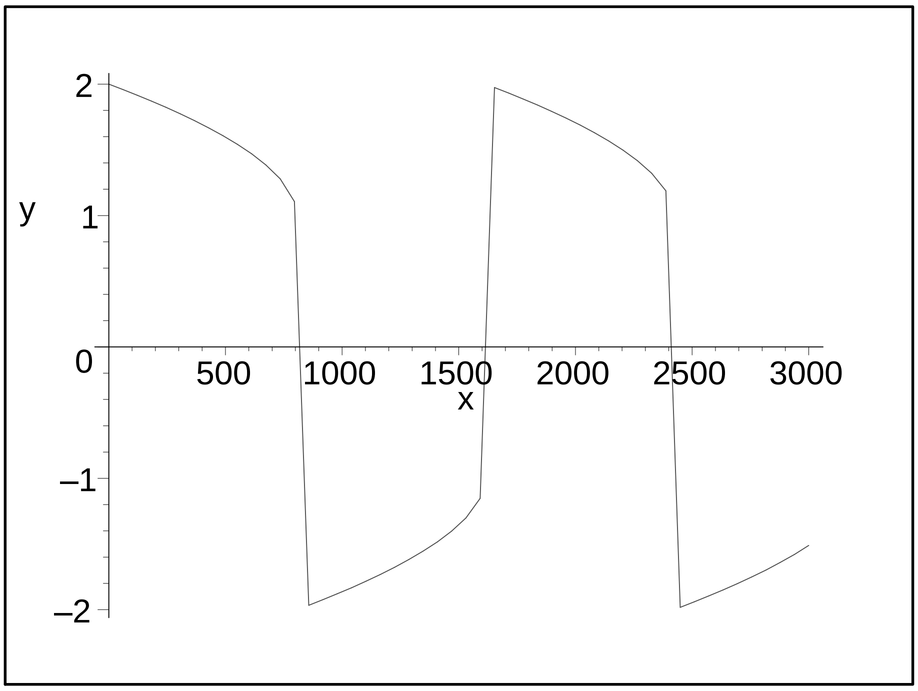


Figure 13: Numerical solution of a stiff IVP.

4.4 Example 4.4. A Stiff Problem: van der Pol's equation

The Van der Pol equation in relaxation oscillation is a famous example of a stiff nonlinear problem.

```
> mu := 1000:
> de := diff(y(x),x,x)-mu*(1-y(x)^2)*diff(y(x),x)+y(x) = 0;
```

$$de := \frac{d^2}{dx^2}\,\mathrm{y}(x) - 1000\,(1-\mathrm{y}(x)^2)\,\frac{d}{dx}\,\mathrm{y}(x) + \mathrm{y}(x) = 0$$

```
> inits[1] := y(0)=2, D(y)(0)=0;
```

$$inits_1 := \mathrm{y}(0) = 2,\ \mathrm{D}(y)(0) = 0$$

The following attempt to compute a numerical solution by the default non-stiff method leads to trouble.

```
> soln[0] := dsolve({de,inits[1]}, type=numeric, range=0..3000):

   Warning, cannot evaluate the solution further right of 6.1339278,
   maxfun limit exceeded (see ?dsolve,maxfun for details)
```

Specify the option `stiff=true` so that the default stiff method will be used, namely an implicit Rosenbrock third-fourth order Runge-Kutta method.

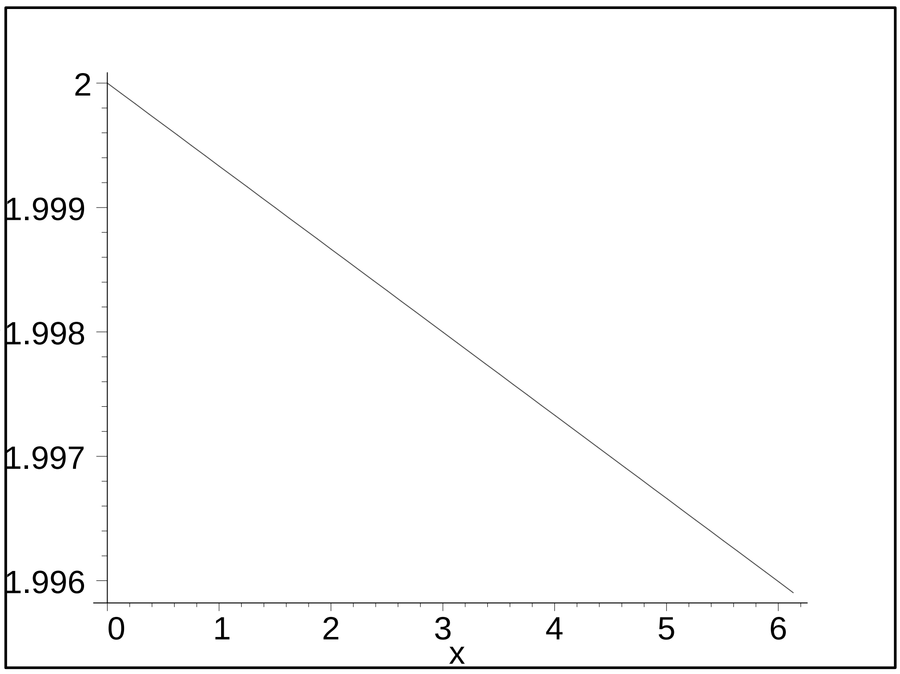


Figure 14: Aborted attempt to solve a stiff IVP.

```
> soln[1] := dsolve({de,inits[1]}, type=numeric, range=0..3000,
                                                     stiff=true):
```

A plot of the solution shows that there are regions of sharp change. The IVP is stiff where the solution is slowly varying.

```
> plots[odeplot](soln[1], [x, y(x)]);  # Figure 13
```

By experimenting with other initial conditions, one can see that all nontrivial solutions converge very rapidly to this same limit cycle.

Finally, let us check what was computed in the aborted attempt which used the default non-stiff method. We see that it was unable to compute beyond approximately $x = 6$.

```
> plots[odeplot](soln[0], [x, y(x)]);  # Figure 14
```

References

[1] S.A. Abramov, M. Petkovšek and A. Ryabenko, Special formal series solutions of linear operator equations. *Discrete Math.* 210, 2000, pp. 3-25.

[2] E.S. Cheb-Terrab, ODEtools: A Maple Package for Studying and Solving Ordinary Differential Equations. To appear in *Handbook of Computer Algebra*, Ed. J. Grabmeier, E. Kaltofen, V.Weispfenning, Springer. [http://lie.uwaterloo.ca/description/odetools].

[3] K.O. Geddes, Numerical integration in a symbolic context. *Proceedings of SYMSAC'86*, B.W. Char (ed.), ACM Press, New York, 1986, pp. 185-191.

[4] K.O. Geddes and G.J. Fee, Hybrid symbolic-numeric integration in Maple. *Proceedings of ISSAC'92*, P.S. Wang (ed.), ACM Press, New York, 1992, pp. 36-41.

[5] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, 2nd edition. The Johns Hopkins University Press, Baltimore, MD, 1989.

[6] E. Kamke, *Differentialgleichungen: Losungsmethoden und Losungen.* New York: Chelsea Publishing Co, 1959.

[7] Michael B. Monagan, Keith O. Geddes, K. Michael Heal, George Labahn, Stefan M. Vorkoetter, James McCarron and Paul DeMarco, *Maple 8 Introductory Programming Guide.* Waterloo Maple Inc., Waterloo, Ontario, Canada, 2002.

[8] Michael B. Monagan, Keith O. Geddes, K. Michael Heal, George Labahn, Stefan M. Vorkoetter, James McCarron and Paul DeMarco, *Maple 8 Advanced Programming Guide.* Waterloo Maple Inc., Waterloo, Ontario, Canada, 2002.

[9] D. Zwillinger, *Handbook of Differential Equations*, 2nd edition. Academic Press, 1992.