Chapter 11

# Power Series Methods

## Application 11.2
## Automatic Computation of Series Coefficients

Repeated application of the recurrence relation to grind out successive coefficients is — especially in the case of a recurrence relation with three or more terms — a tedious aspect of the infinite series method. Here we illustrate the use of a computer algebra system for this task. In Example 7 of Section 11.2 in the text we saw that the coefficients in the series solution $y = \sum c_n x^n$ of the differential equation $y'' - x\,y' - x^2 y = 0$ are given in terms of the two arbitrary coefficients $c_0$ and $c_1$ by

$$c_2 = 0, \quad c_3 = \frac{c_1}{6}, \text{ and } c_{n+2} = \frac{n\,c_n + c_{n-2}}{(n+2)(n+1)} \quad \text{for } n \geq 2. \tag{1}$$

It looks like a routine matter to implement such a recurrence relation, but a twist results from the fact that a typical computer system array is indexed by the subscripts $1, 2, 3, \cdots$ rather than by the subscripts $0, 1, 2, \cdots$ that match the exponents in the successive terms of a power series that begins with a constant term. For this reason we rewrite our proposed power series solution in the form

$$y = \sum_{n=0}^{\infty} c_n x^n = \sum_{n=1}^{\infty} b_n x^{n-1} \tag{2}$$

where $b_n = c_{n-1}$ for each $n \geq 1$. Then the first two conditions in (1) say that $b_3 = 0,\ b_4 = b_2/6,$ and the recurrence relation (with $n$ replaced with $n - 1$) yields the new recurrence relation

$$b_{n+2} = c_{n+1} = \frac{(n-1)c_{n-1} + c_{n-3}}{(n+1)(n)} = \frac{(n-1)b_n + b_{n-2}}{n(n+1)} \tag{3}$$

Now we're ready to go. The *Maple*, *Mathematica*, and MATLAB implementations of this recurrence relation illustrated in the paragraphs below yield the 11-term partial sum

$$y(x) = 1 + x + \frac{x^3}{6} + \frac{x^4}{12} + \frac{3x^5}{40} + \frac{x^6}{90} + \frac{13x^7}{1008} + \frac{3x^8}{1120} + \frac{119x^9}{51840} + \frac{41x^{10}}{113400} + \cdots. \tag{4}$$

You can apply this method to any of the examples and problems in Section 11.2 of the text.

## Using *Maple*

Suppose we want to calculate the terms through the 10th degree — that is, 11 terms

```
k := 11:        # k terms
```

— in (2) with the initial conditions $b_1 = b_2 = 1$. We begin by setting up an array of $k$ terms and specifying the given (initial) values of the first two terms.

```
b := array(1..k):
b[1] := 1:      # given  y(0) value
b[2] := 1:      # given y'(0) value
```

Also, because we know also that $b_3 = 0$ and $b_4 = b_2/6$, we define the additional values

```
b[3] := 0:
b[4] := b[2]/6:
```

Using the recurrence relation in (3), subsequent coefficient values are now calculated by the loop

```
for n from 3 by 1 to k-2 do
    b[n+2] := ((n-1)*b[n]+b[n-2])/(n*(n+1));
    od;
```

which quickly yields the $b$-coefficient values corresponding to the solution given in (4). You might note that the even- and odd-degree terms there agree with those shown in Eqs. (18) and (19) of Example 7 in the text. You can substitute $b_1 = 1,\ b_2 = 0$ and $b_1 = 0,\ b_2 = 1$ separately (instead of $b_1 = b_2 = 1$) in the commands above to derive these two solutions separately.

## Using *Mathematica*

Suppose we want to calculate the terms through the 10th degree — that is, 11 terms

```
k = 11;            (*  k terms  *)
```

— in (2) with the initial conditions $b_1 = b_2 = 1$. We begin by setting up an array of $k$ terms and specifying the given (initial) values of the first two terms.

```
b = Table[0,{n,1,k}];
b[[1]] = 1;        (* given  y(0) value *)
b[[2]] = 1;        (* given y'(0) value *)
```

Also, because we know also that $b_3 = 0$ and $b_4 = b_2 / 6,$ we define the additional values

```
b[[3]] = 0;
b[[4]] = b[[2]]/6;
```

Using the recurrence relation in (3), subsequent coefficient values are now calculated by the loop

```
For[n = 3, n <= k-2,
    b[[n+2]] = ((n-1)b[[n]] + b[[n-2]])/(n*(n+1));
    n = n+1];
b
```

which quickly yields the $b$-coefficient values corresponding to the solution given in (4). You might note that the even- and odd-degree terms there agree with those shown in Eqs. (18) and (19) of Example 7 in the text. You can substitute $b_1 = 1,\ b_2 = 0$ and $b_1 = 0,\ b_2 = 1$ separately (instead of $b_1 = b_2 = 1$) in the commands above to derive these two solutions separately.

## Using MATLAB

Suppose we want to calculate the terms through the 10th degree — that is, 11 terms

```
k = 11;      % k terms
```

— in (2) with the initial conditions $b_1 = b_2 = 1$. We begin by setting up an array of $k$ terms and specifying the given (initial) values of the first two terms.

```
b = 0*(1:k);
b(1) = 1;    % given y(0) value
b(2) = 1;    % given y'(0) value
```

Also, because we know also that $b_3 = 0$ and $b_4 = b_2 / 6,$ we define the additional values

```
b(3) = 0;
b(4) = b(2)/6;
```

Using the recurrence relation in (3), subsequent coefficient values are now calculated by the loop

```
for n = 3:k-2
    b(n+2) = ((n-1)*b(n)+b(n-2))/(n*(n+1));
    end
```

When we display the resulting coefficient values

```
format rat, b
```

we see the same coefficients as displayed in the solution (4), *except that* the coefficient $b_{10}$ of $x^9$ is shown as 73/31801 rather than the correct value 119/51840 shown in (4). It happens that

$$\frac{73}{31801} \approx 0.0022955253 \text{ while } \frac{119}{51840} \approx 0.0022955247$$

so the two rational fractions agree when rounded off to 9 decimal places. The explanation is that MATLAB (as opposed to *Maple* and *Mathematica*) works internally with decimal rather than exact arithmetic. But at the end of the computation, its **`format rat`** algorithm converts a correct 14-place approximation for $b_{10}$ to an incorrect rational fraction that's "close but no cigar."

You can substitute $b_1 = 1,\ b_2 = 0$ and $b_1 = 0,\ b_2 = 1$ separately (instead of $b_1 = b_2 = 1$ in the commands above) to derive partial sums of the two linearly independent solutions displayed in Eqs. (18) and (19) of Example 7.