Part I

The Taylor series method and **TIDES**

Chapter 2

The Taylor series method integrator

In this section we present the Taylor series method to integrate ordinary differential equations. We show the method with all its variants, options and additions.

2.1 The Taylor Series Method integrator (**TSM Integrator**) and its parameters

Let us consider the initial value problem:

$$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{f}(t, \boldsymbol{y}(t); \boldsymbol{p}), \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \qquad t \in \mathbb{R},\ \boldsymbol{y} \in \mathbb{R}^n,\ \boldsymbol{p} \in \mathbb{R}^m, \tag{2.1}$$

where $\boldsymbol{y}_0$ are the initial conditions and $\boldsymbol{p}$ the parameters.

Now, the value of the solution at $t_{i+1} = t_i + h_{i+1}$ (that is, $\boldsymbol{y}(t_{i+1})$) is approximated from the N-th degree Taylor series of $\boldsymbol{y}(t)$ developed at t_i and evaluated at $t = t_{i+1}$. The function $\boldsymbol{f}$ has to be a smooth function, in this tutorial we consider that $\boldsymbol{f}$ is analytic.

$$\begin{aligned}
\boldsymbol{y}(t_0) &\overset{\text{def}}{=} \boldsymbol{y}_0, \\
\boldsymbol{y}(t_{i+1}) &\simeq \boldsymbol{y}(t_i) + \frac{d\boldsymbol{y}(t_i)}{dt}\, h_{i+1} + \frac{1}{2!}\frac{d^2\boldsymbol{y}(t_i)}{dt^2}\, h_{i+1}^2 + \ldots + \frac{1}{N!}\frac{d^N\boldsymbol{y}(t_i)}{dt^N}\, h_{i+1}^N, \\
&\simeq \boldsymbol{y}_i + \boldsymbol{f}(t_i, \boldsymbol{y}_i)\, h_{i+1} + \frac{1}{2!}\frac{d\boldsymbol{f}(t_i, \boldsymbol{y}_i)}{dt}\, h_{i+1}^2 + \ldots + \frac{1}{N!}\frac{d^{N-1}\boldsymbol{f}(t_i, \boldsymbol{y}_i)}{dt^{N-1}}\, h_{i+1}^N \overset{\text{def}}{=} \boldsymbol{y}_{i+1}.
\end{aligned} \tag{2.2}$$

From the formulation of the TSM, the problem is reduced to the determination of the Taylor coefficients $\{d^j\boldsymbol{y}(t_i)/dt^j\}$ by means of the use of automatic differentiation (AD) techniques.

The TSM presents several peculiarities. One of them is that it gives directly a dense output in the form of a power series and therefore we can evaluate the solution at any time just by using the Horner algorithm. Also, as TSM of degree N are also of order N, the use of TSMs of high degree give us numerical methods of high order. Therefore, they are very useful for high-precision solution of ODEs.

In the practical implementation of a numerical method for the solution of ODEs the use of variable stepsizes is a crucial point because it permits to automatize the control of the error. In TIDES we use an absolute error tolerance `tolabs` and a relative tolerance `tolrel`. With both we construct the error tolerance

$$\texttt{TOL} = \texttt{tolabs} + \max(\|\boldsymbol{y}(t_i)\|, \|\boldsymbol{y}(t_{i-1})\|) \times \texttt{tolrel}.$$

Another crucial point in the TSM is the selection of the order of the method, that is, N. In TIDES we adopt a modification of the *optimal order*. On one hand, when we use an order that depends only on the requested tolerance `tolabs`, we adopt the simple formula

$$\hat{n} = \lceil -\ln(\texttt{tolabs})/2 \rceil + \texttt{nordinc},$$

where `maxord` is the maximum order and `nordinc` is an increment of the order with respect to the asymptotic formula (this may be adjusted by the user). This is the case on the dp-tides and mp-tides programs, where the complexity of the extended Taylor series algorithm does not justify to use a more adaptive algorithm. In the minf-tides and minc-tides programs we use a slightly more sophisticated formula

$$\begin{aligned}
\texttt{tolorder}(i) &= \min\left(\texttt{tolabs}/\min(\|\boldsymbol{y}(t_i)\|, \|\boldsymbol{y}(t_{i-1})\|), \texttt{tolrel}\right), \\
\hat{n} &= \lceil -\ln(\texttt{tolorder}(i))/2 \rceil + \texttt{nordinc}.
\end{aligned}$$

In both cases we use

$$N = \max\left(\texttt{minord}, \hat{n}\right).$$

We use two strategies for selecting the stepsize. The first one is based on estimating the error just by taking the last term in the Taylor series (in order to avoid problems with odd/even functions we take the last two terms different from zero, which avoid also problems with polynomial solutions). Note that this strategy is also equivalent to the concept of RK pairs (two RK methods, one of lower order than the other, which permits to estimate the error). So,

$$\begin{aligned}
\hat{h}_{i+1} &= \min\left\{ \left(\frac{\texttt{TOL}}{\|\boldsymbol{y}^{[N-1]}(t_i)\|_\infty}\right)^{1/(N-1)}, \left(\frac{\texttt{TOL}}{\|\boldsymbol{y}^{[N]}(t_i)\|_\infty}\right)^{1/N} \right\}, \\
h_{i+1} &= \texttt{fac1} \times \max\Big(\min(\texttt{rmaxstep} \times h_i, \hat{h}_{i+1}), \texttt{rminstep} \times h_i\Big),
\end{aligned} \tag{2.3}$$

with $\boldsymbol{y}^{[N]}$ the normalised derivative $\boldsymbol{y}^{[N]} = \boldsymbol{y}^{(N)}/N!$, `fac1` a safety factor (we use $\texttt{fac1} = 0.9$), and `rmaxstep` and `rminstep` stands for the maximum and minimum ratio between the actual stepsize and the previous one.

After this selection of the stepsize we may enter, or not, in a refinement process which is based on the `defect error control`. Note that, “a priori”, in the TSM there is no rejected step as occurs in any variable-stepsize formulation for Runge-Kutta or multistep methods because we choose the stepsize once the series are generated in order to obtain the required precision level. But, in order to give more guarantee about the stepsize we may analyse the agreement between the tangent vector to the Taylor polynomial and the vector field at the end of the step, that is, given the Taylor approximation of the solution on the interval $[t_i,\, t_{i+1}] = [t_i,\, t_i + h_{i+1}]$

$$\boldsymbol{y}(t) \simeq \sum_{k=0}^{N} \boldsymbol{y}^{[k]}(t_i) \cdot (t - t_i)^k, \qquad \boldsymbol{y}'(t) \simeq \sum_{k=1}^{N} k\, \boldsymbol{y}^{[k]}(t_i) \cdot (t - t_i)^{k-1}$$

then evaluating at the end of the interval $\boldsymbol{y}'_{i+1} \equiv \sum_{k=1}^{N} k\, \boldsymbol{y}^{[k]}(t_i) \cdot (h_{i+1})^{k-1}$ and the criteria for rejecting the stepsize is

$$\text{if} \quad \|\boldsymbol{y}'_{i+1} - \boldsymbol{f}(t_{i+1},\, \boldsymbol{y}_{i+1})\|_\infty > \texttt{fac2} \times \texttt{TOL} \quad \text{then} \quad \widetilde{h}_{i+1} = \texttt{fac3} \cdot h_{i+1}, \tag{2.4}$$

where `fac2` and `fac3` are control factors that reduce the stepsize (we have taken $\texttt{fac2} = 10$, $\texttt{fac3} = 0.8$). It is important to remark that although the stepsize may be rejected we do not have to recalculate the Taylor coefficients, we only have to consider the new stepsize and enter again in the criteria for rejecting the stepsize. Therefore we cannot say that we reject a complete step, we just reject the estimation of the stepsize, and so the computational cost is not very hight (in fact the cost of evaluating $\boldsymbol{y}'_{i+1}$ and $\boldsymbol{f}(t_{i+1},\, \boldsymbol{y}_{i+1})$). This process is done a maximum of `nitermax` times. If the TSM Integrator reaches the maximum iteration number without achieving the condition, it shows an error message.

The TSM Integrator computes Taylor series around a point that is computed by adding the step to the previous point. In order to avoid the accumulated small errors in that addition we implement, as an option, the `Kahan summation` algorithm.

The evaluation of the Taylor series (polynomials) is made by using the Horner scheme. However, when a polynomial is bad conditioned this evaluation may give very unaccurate results. In order to minimize such effects we implement, as an option, the `Compensated Horner` algorithm.

2.2 Inputs and Outputs of the TSM Integrator

To integrate the ODE (1.1) the TSM Integrator needs the numerical value of the initial conditions of the variables $\boldsymbol{y}_0$ and the numerical value of the parameters $\boldsymbol{p}$. These values must be passed to the TSM Integrator as the main input.

We may choose between a dense output, i.e. the solution in a list of equidistant (or not) points $\{t_0, t_1, \ldots, t_f\}$, or a non dense output, i.e. only at the final point t_f.

The basic output of a TSM Integrator is the result of the integration, i.e. the value of the variables $\boldsymbol{y}(t)$ at the desired points: $\{t_0, t_1, \ldots, t_f\}$. Likewise, we may add to the output the values of a function $G(\boldsymbol{y}(t))$ and the values of the partial derivatives of $\boldsymbol{y}(t)$

and $G(\boldsymbol{y}(t))$ with respect to the initial conditions or the parameters evaluated in the same points $\{t_0, t_1, \ldots, t_f\}$.

The previous output has the format of a matrix in which each row i represents the solution in t_i. The elements of the row are: $t_i, \boldsymbol{y}(t_i)$, and depending on the case, $G(\boldsymbol{y}(t_i))$, $\partial \boldsymbol{y}(t_i)/\partial s_j$, $\partial G(\boldsymbol{y}(t_i))/\partial s_j$, with s_i the elements, in order, with respect to we compute the partial derivatives.

The output can be written on a data matrix (only in the standard versions 3.2) and into a file or the screen (all versions).

Instead to compute the solution at the desired points, sometimes we want to compute when determinate events occur. An event is a zero, local extremum, local maximum or local minimum of a function $G(\boldsymbol{y}(t))$.

We may summarise the options of a TSM Integrator, to obtain the desired solution, in the following scheme

- A vector with the `initial conditions` $\boldsymbol{y}_0$.
- A vector with the `parameters` $\boldsymbol{p}$.
- The list $\{t_0, t_1, \ldots, t_f\}$ of `integration points` for the dense output or the initial and the final point $\{t_0, t_f\}$ for a non dense output.
- The `list of extra functions` $G(\boldsymbol{y}(t))$ that we want to evaluate.
- The `list of partials` with respect to we want to compute the solution and the extra functions.
- The `event` and the `function event` $G(\boldsymbol{y}(t))$ that we want to compute.
- The way in which we want the output: `file`, `screen` or `data matrix`.

Chapter 3

About TIDES

3.1 What is TIDES?

TIDES (Taylor series Integrator for Differential EquationS) is a software to integrate, by using the Taylor Series method, systems (1.1) of first order differential equations (ODEs). The Taylor Series Method (TSM) is based on the evaluation of the time Taylor series of the variables. These series are obtained by an iterative way that uses the decomposition of the derivatives by automatic differentiation (AD) methods.

TIDES has two different parts (pieces of software): The MATHEMATICA package MathTIDES and the C library LibTIDES.

TIDES		
Product		Language
MathTIDES	preprocessor	MATHEMATICA (version 6.0 or higher)
LibTIDES	library (objects or source code)	C FORTRAN

The preprocessor MathTIDES helps the user to write the code to integrate a particular ODE.

To integrate the code wrote with MathTIDES we must compile and link it together with the library LibTIDES that contains the kernel of the Taylor Series Method integrator (TSM Integrator).

There are four versions of the TSM Integrator: two minimal versions in C and Fortran respectively and two standard C versions in double and multiple precision. The multiple precision version of the integrator requires MPFR (version 2.4 or higher) and GMP (version 4.1 or higher) libraries (`http://www.mpfr.org/`, and `http://gmplib.org`).

3.2 The four versions of the TSM Integrator

When we integrate an ODE with TIDES we may choose among four different versions of the kernel: two minimal (faster) versions in FORTRAN (minf-tides) and C (minc-tides) respectively, and two standard (more complete) versions in C, with double (dp-tides) or arbitrary precision (mp-tides) respectively.

Version	Contents	MathTIDES generates	linked with
minf-tides	basic TSM	FORTRAN files	LibTIDES
minc-tides	basic TSM	C files	LibTIDES
dp-tides	complete TSM + partial derivatives	C files	LibTIDES
mp-tides	complete TSM + partial derivatives + arbitrary precision	C files	LibTIDES GMP library MPFR library

The limitations of the minimal versions permit more simple data structures in their implementation. Taking advantage of this implementation we obtain a faster code.

3.2.1 Minimal versions (minf-tides, minc-tides)

The minimal versions of the TSM Integrator produce a basic Taylor series integrator characterized by the following points

- In the mathematical expression of $\boldsymbol{f}$ in (1.1) may appear the following functions:
  - The usual operators: $+, -, *, /$
  - A number(or constant parameter) power to a variable: $a^x, a > 0$.
  - A variable power to a number(or constant parameter): $x^r, r \in \mathbb{R}$.
  - A variable power to a variable : x^y.
  - Functions: $\sin, \cos, \tan, \log$.
- The minimal version integrates only one differential system on each main problem.
- The minimal version writes the output, dense or not, into a file or on the screen.

- minf-tides is based on two FORTRAN files generated by MathTIDES. The first file (ODE file), with extension `.f`, contains the iterative procedure to construct the function $\boldsymbol{f}$. The second file, whose name begins by `dr_`, contains the driver (main program) to call the integrator.

- minc-tides is based on three C files generated by MathTIDES. Two files (ODE files), with the same names and extensions `.c, .h`, contain the iterative procedure to construct the function $\boldsymbol{f}$. The third file, whose name begins by `dr_`, contains the driver (main program) to call the integrator.

- To integrate the ODE we only need to compile and run these files. It can be done in two different ways:

  - linking them together with the library LibTIDES,

  - compiling them together with the file `minf_tides.f` (in the minf-tides version) or `minc_tides.c` (in the minc-tides version). In this case it is not necessary to link them with LibTIDES. Later, in this chapter, we will explain how to obtain these files.

3.2.2 Standard versions (dp-tides, mp-tides)

The standard versions of the integrator produce a complete Taylor series integrator characterized by the following points

- In the mathematical expression of $\boldsymbol{f}$ in (1.1) the following functions may appear:

  - The usual operators: $+, -, *, /$
  - A number(or constant parameter) power to a variable: $a^x, a > 0$.
  - A variable power to a number (or constant parameter): $x^r, r \in \mathbb{R}$.
  - A variable power to a variable : x^y.
  - Functions: $\sin, \cos, \tan, \sinh, \cosh, \tanh, \mathrm{asin}, \mathrm{acos}, \mathrm{atan}, \mathrm{asinh}, \mathrm{acosh}, \mathrm{atanh}, \log$.

- The minimal version integrates one or more differential systems on each main problem.

- The minimal version writes the output, dense or not, into a file or on the screen and/or a bidimensional array.

- Simultaneously with the integral of the variables they may obtain :

  - The integral of functions of the variables.

  - The integral of the partials of the variables with respect to the initial conditions.
  - The integral of the partials of the variables with respect to the parameters.
  - The integral of the partials of functions of the variables with respect to the initial conditions.
  - The integral of the partials of functions of the variables with respect to the parameters.

- We may compute *events*: points where a function of the solution is a zero or an extremum.

- Both versions are based on three C files: the driver (basic main program) an two files (ODE files), with the same names and extensions `.c, .h`, that contains the iterative procedure to construct the function $\boldsymbol{f}$. These files must be compiled and linked with the library LibTIDES (kernel of the integrator) to integrate the ODE.

- mp-tides uses the MPFR and GMP libraries (**libmpfr.a**, **libgmp.a**) to integrate in multiple precision with any number of precision digits. In this case we must link the files with the libraries LibTIDES, MPFR and GMP.

Chapter 4

Installing TIDES

Uncompress the TIDES distribution in your home directory, then you have the directory `tides-2.0`. TIDES has two different parts: MathTIDES and LibTIDES, you must install both. The installation process is different for each part. If you plan to use the multiple precision version of TIDES you must install previously the libraries GMP and MPFR.

4.1 Installing MathTIDES

To install MathTIDES you need to copy the folder MathTIDES inside a directory that is in the `$Path` of MATHEMATICA. You can do it manually, or by opening the notebook `InstallMathTIDES.nb` that makes automatic the installation process following the instructions of the notebook.

4.2 Installing GMP and MPFR

This is only necessary if you need to work with multiple precision.

LibTIDES library uses MPFR (version 2.4 or higher) and GMP (version 4.1 or higher) libraries for multiple precision computations, so you need to have both installed. Then, you must install GMP and MPFR, in this order, if your system does not have them. You can download GMP from http://www.gmplib.org and MPFR from http://www.mpfr.org. Then you must uncompress them and run on the terminal the following four orders

```
./configure
make
make check
```

```
sudo make install
```

inside each of its directories. The last order supposes that you have administrator privileges.

4.3 Installing LibTIDES

The next installation procedure creates and installs LibTIDES in a Unix environment (Macos X and Windows with MinGW included).

To install LibTIDES you must uncompress it and run on the terminal the following five orders

```
cd $HOME/tides-2.0
./configure
make
make check
sudo make install
```

Later you may remove the directory `tides-2.0`.

4.3.1 Changing the work directory

The first order

```
cd $HOME/tides-2.0
```

changes the work directory to the TIDES directory.

4.3.2 Configuring the installation

To configure the complete installation type on the terminal

```
./configure
```

Depending on where you installed GMP and/or MPFR, you may need to specify its installation directories. For example, if you put GMP in /usr/local, then you need to do the following

```
./configure --with-gmp=/usr/local
```

If MPFR is also in a non-standard directory, you may have to do the same thing with it:

```
./configure --with-gmp=/usr/local --with-mpfr=/usr/local
```

If you don't have GMP and/or MPFR installed, or you are not interested in having multiple precision capacities in your program, pass the following option to configure:

```
./configure --disable-multiple-precision
```

This will create the needed Makefiles to compile a reduced version of `libTIDES.a` without the MPFR extensions.

By default, the library will be installed in `/usr/local/lib`. If you prefer another installation directory, specify it by adding the prefix option to configure.

```
./configure --prefix=......
```

By default the FORTRAN files of LibTIDES are created. If you do not have a FORTRAN compiler or you do not plan to use the minimal FORTRAN version of TIDES you may use the following configure option

```
./configure --disable-fortran
```

4.3.3 Making the library

To build the library, type on the terminal:

```
make
```

This will create the complete library or only the double precision version of the library depending on the options of `configure`.

The library `libTIDES.a` is created inside the directory `"$HOME/tides-2.0/libTIDES"`.

4.3.4 Checking the library before installation

Before to install LibTIDES it is useful to check the build library. To check it (run the test files) type:

```
make check
```

The test includes the double precision test and the multiple precision test when available. If everything is OK, you can install it.

4.3.5 Installing and uninstalling the library

If you have administrator privileges you can install the library by typing on the terminal

```
sudo make install
```

To uninstall the library just type

```
sudo make uninstall
```

The word `sudo` it is not necessary if you are `root user`.

If you have not administrator privileges take the library `libTIDES.a` created on the TIDES directory and copy it on your desired directory. You can do the same with the libraries GMP and MPFR. In this case you need too the header files `dp_tides.h`, `mp_tides.h` and `mpfr.h`.

4.3.6 Checking the library after installation

```
make installcheck
```

verifies LibTIDES after the installation process.

4.3.7 Working with Mac OS X

Taking into account that Mac OS X is based on a Unix system you can install LibTIDES on Mac OS X by following all the previous steps from the terminal, and using the `gcc` compiler installed on Mac OS X with the Developer tools.

If you prefer to use Xcode, instead working from the terminal, follow the previous steps except the order `sudo make install`. Then take the library `libTIDES.a` and the header file `dp_tides.h`, and include them in your XCode project. If you work with multiple precision do the same with the files `mp_tides.h` and `mpfr.h` and the MPFR and GMP libraries.

4.3.8 Working with Windows

The installation has been tested with `MinGW` and `Msys`. The GMP and MPFR libraries are not installed, so you have to build and install them. They will be installed at `/usr/local`, but `Msys` does not have it in the path, so you have to use:

```
./configure --with-gmp=/usr/local --with-mpfr=/usr/local
```