Installing HOPPET

William Woodley
3 August 2018

The following steps can be run on a Mac, on a Linux machine, or from within ACE-NET.

1. Open a terminal.
2. HOPPET is written in Fortran. Check whether a Fortran compiler is installed on the computer:

   ```
   gfortran --version
   ```

3. If one is, skip to step 7. If one is not, and the machine is a Windows machine, install MinGW and MSYS. If the machine is Linux, run `sudo apt-get install gfortran`, and skip to step 7. If the machine is a Mac, install Homebrew:

   ```
   /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
   master/install)"
   ```

4. Install gfortran using Homebrew:

   ```
   brew install gcc
   ```

5. Check that the Homebrew configuration is correct and that everything is up-to-date:

   ```
   brew doctor
   ```

6. Upgrade anything Homebrew's brew doctor says needs upgrading. If XCode needs upgrading, upgrade it from the App Store. If gcc needs upgrading, type `brew upgrade gcc` (or `brew upgrade gcc49`) into the terminal. XCode and `xcode-select` would not upgrade properly for me, so I uninstalled XCode, and then the following steps worked.
7. Download HOPPET:

   https://hoppet.hepforge.org

8. Move into the HOPPET directory:

   ```
   cd /.../hoppet-1.2.0
   ```

9. If changes are to be made to `src` files, such as changing the value of $\Lambda_{\rm QCD}$ (`qcdl5` on line 233 of the `qcd_coupling.f90` file), change them now, before configuring.
10. Follow the instructions in the `INSTALL` file:

    ```
    ./configure
    make
    make check
    make install
    make install-mod
    ```

11. Move into the Fortran-90 example directory:

    ```
    cd example_f90
    ```

12. Run the example:

    ```
    ./tabulation_example
    ```

Using HOPPET

These steps are for editing the tabulation_example.f90 file that comes with HOPPET (the line numbers here refer to the lines on that file). The file tabulation_example for bSAT.f90 is an already-edited version for use with the b-SAT R code, bSAT.Rmd. In order to use this edited file, change the file name to tabulation_example.f90, and follow steps 13 and 14 below. Detailed information on the HOPPET algorithms and numerical methods is in the HOPPET documentation (PDF).

1. In the /hoppet-1.2.0/example_f90 folder, open the (original) tabulation_example file to edit it.

2. The input for this is a parton distribution function (PDF) for a set of x values at a scale Q0 (either as a function of x, or as an array of values), and the output is the PDF for a set of x values at a scale Q.

3. The set of x values for the output PDF is on line 37 in the array heralhc_xvals. Change the values in the array, and change the length of the array (the 9 in brackets on line 36) to however many x values there are.

4. The set of x values that the code uses to evolve the PDF is created as a grid (a HOPPET data type) called grid on lines 59 to 63, and then on lines 81 and 82. If the initial PDF is being input as an array of values, those values must correspond to the x values in xValues(grid). Note that heralhc_xvals is not the same as xValues(grid).

5. The initial PDF grid, pdf0, is created using the XValues(grid) array on line 82 with the unpolarized_dummy_pdf() function at the bottom of the file.

6. If the PDF is being evaluated at very small x values ($x < 6 \times 10^{-6}$), increase the ymax value on line 55 to ymax $= -\ln(x_{\min})$.

7. The code to write the output is on lines 114 to 119, with the array of PDFs pdf_at_xQ of the form:

-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
tbar	bbar	cbar	sbar	ubar	dbar	g	d	u	s	c	b	t

   The indexing of PDFs goes from -6 to 6. Therefore, to return, for example, the values of the gluon density $xg(x,\ \mu) = xg($heralhc_xvals, Q$)$, write pdf_at_xQ(0).

8. For a PDF as a function of x of the form $cx^a(1-x)^b$, change the numbers on lines 140, 141, and 150 to 157. For a different form, change the expressions on lines 150 to 157.

9. For a PDF as a table of values, replace the expressions on lines 150 to 157 with arrays of numbers that correspond to the xValues(grid) values.

10. The number of loops, nloop, on line 66 determines whether the evolution is done to LO, NLO, or NNLO.

11. The initial scale, Q0, is set on line 83.

12. The final scale, Q, in set on line 107.

13. Save the file, and remake it from a terminal:

```
make tabulation_example
```

14. Run the file:

```
./tabulation_example
```

Calling HOPPET from Maple

Using Maple Output in HOPPET:

1. Export the Maple variable, `output`, by writing the following in Maple:

   ```
   writedata("Maple_output.txt", output):
   ```

   This creates a file `Maple_output.txt` in the working directory.

2. Prepare the data for use in HOPPET by declaring a variable for it at the top of `tabulation_example.f90`. If it is a single number named `Maple_input`, write the following:

   ```
   real(dp) ::  Maple_input
   ```

3. Import the data into HOPPET by adding the following lines to `tabulation_example.f90`:

   ```
   open(25, file = "Maple_output.txt", status = 'old', access = 'sequential', form =
   'formatted', action = 'read')

   read(25, fmt = *) Maple_input

   close(25)
   ```

   The number `25` is the unit number that identifies the file `Maple_output.txt`.

Using HOPPET Output in Maple (Method 1):

1. Create a blank text file called `HOPPET_output.txt`, and place it in the `example_f90` folder.

2. Remove lines 108 to 111, line 104, and line 69 of the unedited `tabulation_example.f90` file.

3. Replace what is on line 114 of the unedited `tabulation_example.f90` file with the following:

   ```
   open(26, file = "HOPPET_output.txt", status = 'old', access = 'sequential', form =
   'formatted', action = 'write')

   write(26, fmt = *) &

   close(26)
   ```

   The number `26` is the unit number that identifies the file `HOPPET_output.txt`. If the number of PDFs that are being returned is known (for example, if it is the PDFs for up and down quarks that are being evolved), then `*` in the `write()` statement can be replaced with something more specific, such as `'(2(es11.5))'`, which specifies two (`2`) scientific (`es`) outputs of width eleven (`11`) with five (`5`) digits after the decimal place.

4. Save and remake the file:

   ```
   make tabulation_example
   ```

5. If the Maple file is not in the same folder as the HOPPET file, type the following into Maple, replacing the path in the `currentdir()` function with the correct directory location:

   ```
   currentdir("/.../hoppet-1.2.0/example_f90"):
   ```

6. Run the Fortran code by typing the following into Maple:

   ```
   system("./tabulation_example"):
   ```

7. Read in the HOPPET output and store it in a list called pdf:

```
pdf := readdata("/.../hoppet-1.2.0/example_f90/HOPPET_output.txt", N_col);
```

N_col is the number of columns in HOPPET_output.txt. To evolve the gluon density only (meaning the statement following the write() statement from step 2 above is simply pdf_at_xQ(0), as in tabulation_example for bSAT.f90), then set N_col := 1: in Maple.

8. If necessary, reset the working directory with currentdir() back to where the Maple file is located.

Using HOPPET Output in Maple (Method 2):

1. Remove lines 108 to 111, line 104, and line 69 of the unedited tabulation_example.f90 file, save it, and remake it:

```
make tabulation_example
```

2. Load the StringTools package for the Split() function, and load the Statistics package for the Remove() function:

```
with(StringTools):
with(Statistics):
```

3. If the Maple file is not in the same folder as the HOPPET file, type the following into Maple, replacing the path in the currentdir() function with the correct directory location:

```
currentdir("/.../hoppet-1.2.0/example_f90"):
```

4. Run the Fortran code by typing the following into Maple, which stores the output in pdf:

```
pdf := ssystem("./tabulation_example"):
```

The result is a list. The first element should be a 0, indicating the Fortran code was run successfully. The second element is a string containing the output from HOPPET.

5. Split the string into multiple strings, each of which will either contain a single number, or will be an empty string:

```
pdf := Split(pdf[2]):
```

6. Write a function to identify the empty strings:

```
isEmpty := input -> evalb(input = ""):
```

7. Remove the empty strings from the PDF vector:

```
pdf := Remove(isEmpty, pdf):
```

8. Parse the strings into numbers:

```
for i from 1 to nops(pdf) do:

   pdf[i] := parse(pdf[i]):

od:
```

9. If necessary, convert pdf, which is now a list of numbers, into a matrix with the correct dimensions to match what is output by running ./tabulation_example in the terminal.

10. If necessary, reset the working directory with currentdir() back to where the Maple file is located.