

# Exercises for Parallel Cython

## Computing the Mandelbrot set

1. Go into the `cython-mandel` directory and have quick a look at the sources:
  - `mandel.pyx` -- The Cython algorithm for computing the Mandelbrot set
  - `runmandel.py` -- Creates the fractal arrays and displays it as ASCII art
  - `setup.py` -- The distutils script for compiling the applicationand identify where the parallel loop is located.
2. Compile the package and run it:

```
python setup.py build_ext --inplace # just compile, not install
time python runmandel.py           # run and measure the (real) time spent
```

Note down the time taken for the default run. Now, run it again, but forcing the use of one single thread, by using the `OMP_NUM_THREADS` environment variable:

```
time OMP_NUM_THREADS=1 python runmandel.py
```

- Which is the speed-up of the latter with respect to the default run?
  - How many threads do you think is the default for OpenMP?
3. Re-run the mandel application using a different number of threads.
    - Do you find that it scales smoothly?

## Computing a polynomial

4. Use the sources in `cython-poly` directory to evaluate the next polynomial:

```
((.25*x + .75)*x - 1.5)*x - 2
```

with:

```
time OMP_NUM_THREADS=1 python runpoly.py
```

- Does it scale the same way than the mandel program?
- Why do you think this is so?

## Static vs dynamic scheduling

5. In sources above, the OpenMP scheduling for threads has been set to its optimal value. Play with 'static' and 'dynamic' values for it.
  - Which one is better for each case?
  - Can you explain why?