Exercise set 5

Introduction to Python Programming for Economics & Finance

Richard Foltyn
University of Glasgow

June 2, 2023

Contents

1 Optimisation with interpolation

In this exercise you are asked to find the maximum of a function that is defined by a handful of points defined on a grid:

[1]:
```
grid = [0., 1., 2., 3., 4., 5.]
values = [-1.625, -1.225, -1.025, -1.025, -1.225, -1.625]
```

You will use interpolation to evaluate the function at values that are not on this grid.

Perform the following points:

1. Plot `values` against `grid` using a line plot.
2. Write an objective function that can be used to find the maximum of this function:

   ```
   def objective(x, grid, values):
       ...
   ```

   It should take a scalar argument `x` and evaluate the function value using interpolation.

   Hint: You can use `np.interp()` to perform *linear* interpolation.
3. Use SciPy's `minimize_scalar()` to perform the maximisation. You can pick a specific method or just use the default settings. Report the maximum and the maximiser at which it is attained.

   Hint: You need to pass the additional arguments to `objective` using the `args` argument:

   ```
   minimize_scalar(..., args=(grid, values))
   ```

   Hint: The minimum is stored in the `fun` attribute of the object returns by the minimiser, while the value at which the minimum it attained is stored in the `x` attribute.
4. **[Advanced]** NumPy's `np.interp()` only supports linear interpolation (and does not allow for extrapolation, which, however, is not needed in this case.) Rewrite your objective function using SciPy's `interp1d()` to perform *quadratic* interpolation by specifying `kind='quadratic'`.

   Hint: Unlike NumPy's `np.interp()`, the SciPy variant returns a *function* which can be used to interpolate arbitrary points.