# 4.1.4 Problem Set: IVP implementation

Bookmark this page

To begin, you will complete the implementation of `IVPlib_pset1.py`. Specifically:

1. **Complete `solve(thisIVP, dt, method)` by following the specifications in the docstring.**

   This function integrates `thisIVP` forward in time from $t = t_I$ to $t_F$ taking timesteps of size $\Delta t$ (within the code, the timestep is called `dt`). The integration over one timestep from $t^n$ to $t^{n+1} = t^n + \Delta t$ is done by `method`.

   For example, one possible method is `step_FE` which applies a single step of Forward Euler. Note that `step_FE` is already implemented for you.

   Run `IVP_scenarios.py` to check that your implementation of `solve` is working. This tester uses the `IVPlib_pset1` module to approximate the solution to the five increasingly complex IVP which are described in as an appendix in Section 4.1.7. If you have correctly implemented `solve`, the output from `IVP_scenarios.py` should be very similar to the following:

   ```
   run_test0 results:
       step_FE    : Max error = 2.665e-15

   run_test1 results:
       step_FE    : Max error = 1.518e+00

   run_test2 results:
       step_FE    : Max error = [2.665e-15,1.518e+00]

   run_test3 results:
       step_FE    : Max error = 2.966e-01

   run_test4 results:
       step_FE    : Max error = [8.663e-02,3.215e-02]
   ```

2. **Complete `step_RK4(thisIVP, dt, un, tn)` following the docstring.**

   Now that `solve` is working, you will implement a new method to work with it, specifically the classical 4-stage Runge-Kutta algorithm (RK4).

   Again, use `IVP_scenarios.py` to check your implementation of `step_RK4` by updating the `methods_to_test` list in the `__main__` body of `IVP_scenarios.py`. If you have correctly implemented `step_RK4` and `solve`, the output from `IVP_scenarios.py` should be very similar to the following:

   ```
   run_test0 results:
       step_FE    : Max error = 2.665e-15
       step_RK4   : Max error = 2.665e-15

   run_test1 results:
       step_FE    : Max error = 1.518e+00
       step_RK4   : Max error = 3.553e-15

   run_test2 results:
       step_FE    : Max error = [2.665e-15,1.518e+00]
       step_RK4   : Max error = [2.665e-15,3.553e-15]

   run_test3 results:
       step_FE    : Max error = 2.966e-01
   ```

```
    step_RK4   : Max error = 3.976e-08

run_test4 results:
    step_FE    : Max error = [8.663e-02,3.215e-02]
    step_RK4   : Max error = [1.283e-05,2.710e-05]
```

WARNING: the exact order of operations you used to implement your methods can cause machine precision level differences in the results above. Thus, for numbers such as `3.553e-15`, which are at machine precision, you could see different machine precision values (e.g. `7.105e-15`)

‹ Previous Next ›