lec_act_4_systems PDF

October 31, 2023

[ ]:
```
# Initialize Otter
import otter
grader = otter.Notebook("lec_act_4_systems.ipynb")
```

1 Predator-prey iterative functions

We're going to start implementing a simple version of the predator/prey relationship, a classic differential equations problem also known as the Lotka-Volterra equations. You don't need to know a whole lot about them, except that there are two variables to track - prey and predator - and that how the numbers of prey/predator changes depends on both the current prey values AND the predator values (the differential equations part)

Resources: - https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations for a more general theoretical introduction - https://www.kristakingmath.com/blog/predator-prey-systems a better "what is it" description - https://scientific-python.readthedocs.io/en/latest/notebooks_rst/3_Ordinary_Differential_Equations/02_Examples/Lotka_Volt - implementation in sci-py (which is what you should really use for this type of problem - scipy's ode solver)

Also see the slides for the lab: https://docs.google.com/presentation/d/1wd1SpTJiezfroDizaFkA6UxCgMQE_A4Z

In this lecture activity: You're going to - practice writing a function from a function description - practice turning a bit of math into an iterative function - practice calling functions from functions

Note: You can write the entire solver in one function. But that's a) rather hard to debug and b) can result in making the most common mistake with iterative functions - not computing the new values from the old ones.

[ ]:
```
# Doing the imports for you
import numpy as np
import matplotlib.pyplot as plt
```

1.1 First function - compute the new prey value from the prey and predator

Function input: - Current prey value (number) - current predator value (number) - the parameters as a dictionary (see calling function code)

Output - New prey value (number)

Equation: - dprey/dt = "Prey reproduce" * prey - "Prey eaten" * prey * predator - prey = prey + delta_t * dprey/dt

Function name: Use **compute_prey_from_prey_and_predator**

[ ]:
```
# TODO: Fill in the parameters (see description above). Don't forget to comment␣
 ↪what the input paramters are
def compute_prey_from_prey_and_predator( prey , predator , params):
    # TODO: Calculate the new prey value from the input prey/predator values␣
 ↪and delta t (see equation)
    prey_reproduce = params["Prey reproduce"]
    prey_eaten = params["Prey eaten"]
    delta_t = params["delta t"]

    dprey_dt = prey_reproduce * prey - prey_eaten * prey * predator

    new_prey = prey + delta_t * dprey_dt

    return new_prey
    #  Note: To get, eg, the "Prey reproduce" value use params["Prey␣
 ↪reproduce"].
    pass
```

[ ]:
```
# Test code
# Tie the number of time steps to the total number of days
delta_t = 0.1
n_days = 40
n_time_steps = int(n_days / delta_t)
params = {"Prey reproduce":1.0,
          "Prey eaten":0.02,
          "Predator loss":1.2,
          "Predator reproduce":0.03,
          "delta t": delta_t,    # unit: days
          "n days": n_days,      # unit: days
          "n time steps": n_time_steps}

prey_initial = 100
predator_initial = 100

prey_new = compute_prey_from_prey_and_predator(prey=prey_initial,␣
 ↪predator=predator_initial, params=params)

print(f"Checking prey new {prey_new}, should be 90")
```

```
Checking prey new 90.0, should be 90
```

[ ]:
```
grader.check("prey_from_prey_and_predator")
```

[ ]:
```
prey_from_prey_and_predator results: All test cases passed!
```

2 Second function (predator)

Compute the new predator value from the prey and predator.

Input: - Current prey value (number) - current predator value (numer) - the parameters as a dictionary (see calling function)

Output: - New predator value (number)

Equation: - dpredator/dt = - “Predator loss” * predator + “Predator reproduce” * prey * predator - predator = predator + delta_t * dpredator/dt

Function name

Use **compute_predator_from_prey_and_predator**

[ ]:
```
# TODO: Fill in the parameters (see description above). Don't forget to comment␣
 ↪what the input paramters are
def compute_predator_from_prey_and_predator ( prey , predator , params):
    # TODO: Calculate the new predator value from the input prey/predator␣
 ↪values and delta t (see equation)
    predator_loss = params["Predator loss"]
    predator_reproduce = params["Predator reproduce"]
    delta_t = params["delta t"]

    dpredator_dt = -predator_loss * predator + predator_reproduce * prey *␣
 ↪predator

    new_predator = predator + dpredator_dt * delta_t

    return new_predator
    #  Note: To get, eg, the "Prey reproduce" value use params["Prey␣
 ↪reproduce"].
    pass
```

[ ]:
```
# Check code - uses parameters defined in question 1
predator_new = compute_predator_from_prey_and_predator (prey=prey_initial,␣
 ↪predator=predator_initial, params=params)

print(f"Checking predator new {predator_new}, should be 118")
```

```
Checking predator new 118.0, should be 118
```

[ ]:
```
grader.check("predator_from_prey_and_predator")
```

[ ]:
```
predator_from_prey_and_predator results: All test cases passed!
```

3 Third function (call both)

Put the two functions together.

Input: - Current prey value (number) - current predator value (numer), the parameters as a dictionary (see calling function)

Output: New prey, predator values (tuple)

Functionality: Should just call the two functions, one after the other

Function name: use **compute_one_time_step**

TODO: Once you have this working correctly make one small change - use the new prey value you calculate in the call to calculate the predator value (instead of the input prey value). How much is the result off by? This is a really, really common error and one that is difficult to catch. By writing the code this way (two functions, then calling one function after the other) you're less likely to make that mistake in the first place

Don't forget to put it back to the correct answer

[ ]:
```
...

# TODO: Fill in the parameters (see description above). Don't forget to comment␣
 ↪what the input paramters are
def compute_one_time_step (prey , predator , params):
    # TODO: Calculate the new prey/predator values from the input prey/predator␣
 ↪values and delta t (see equation)
    prey_new2 = compute_prey_from_prey_and_predator (prey, predator, params)
    predator_new2 = compute_predator_from_prey_and_predator ( prey , predator ,␣
 ↪params)
    return prey_new2, predator_new2
    # Do NOT re-write the equations - call the functions you already wrote
    pass
```

[ ]:
```
# Test code
prey_new2, predator_new2 = compute_one_time_step(prey_initial,␣
 ↪predator_initial, params)

print(f"Checking prey new {prey_new2}, should be 90 predator new␣
 ↪{predator_new2}, should be 118")
```

```
Checking prey new 90.0, should be 90 predator new 118.0, should be 118
```

[ ]:
```
grader.check("compute_one_time_step")
```

[ ]:
```
compute_one_time_step results: All test cases passed!
```

3.1 Hours and collaborators

Required for every assignment - fill out before you hand-in.

Listing names and websites helps you to document who you worked with and what internet help you received in the case of any plagiarism issues. You should list names of anyone (in class or not)

who has substantially helped you with an assignment - or anyone you have *helped*. You do not need to list TAs.

Listing hours helps us track if the assignments are too long.

[ ]:
```
# List of names (creates a set)
worked_with_names = {}
# List of URLS (creates a set)
websites = {}
# Approximate number of hours, including lab/in-class time
hours = 10
# for all row, column in all_indices_from_where
#.   if this is the column for wrist torque
#.      print(f"Row: {r}, Time step: {c // n_time_steps} Successful y/n:␣
 ↪{pick_data[r, -1] == 1}, value: {pick_data[r, c]}")
for r in range(len(worked_with_names)):
    for c in range(len(websites)):
         wrist_torque_column = 2
         if c == wrist_torque_column:
            print(f"Row: {r}, Time step: {c // n_time_steps}, Successful y/n:␣
 ↪{pick_data[r][-1] == 1}, value: {pick_data[r][c]}")
            # Assuming 'your_column_for_wrist_torque' is the column index you␣
 ↪are interested in
```

[ ]:
```
grader.check("hours_collaborators")
```

[ ]:
```
hours_collaborators results: All test cases passed!
```

[ ]:
```

```

3.2 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

Submit through gradescope, Lecture activity week 4, iterative systems.

[ ]:
```
# Save your notebook first, then run this cell to export your submission.
grader.export(run_tests=True)
```

```
---------------------------------------------------------------------------
LatexFailed                               Traceback (most recent call last)
File c:
 ↪\Users\user10\anaconda3\Lib\site-packages\otter\export\exporters\via_latex.py
 ↪66, in PDFViaLatexExporter.convert_notebook(cls, nb_path, dest, xecjk,␣
 ↪**kwargs)
     64         output_file.write(latex_output[0])
---> 66 pdf_output = nbconvert.export(pdf_exporter, nb)
```

```
     67 with open(dest, "wb") as output_file:

File c:\Users\user10\anaconda3\Lib\site-packages\nbconvert\exporters\base.py:82
 ↪in export(exporter, nb, **kw)
     81 if isinstance(nb, NotebookNode):
---> 82     output, resources = exporter_instance.from_notebook_node(nb,
 ↪resources)
     83 elif isinstance(nb, (str,)):

File c:\Users\user10\anaconda3\Lib\site-packages\nbconvert\exporters\pdf.py:200
 ↪in PDFExporter.from_notebook_node(self, nb, resources, **kw)
    199 if not os.path.isfile(pdf_file):
--> 200     raise LatexFailed("\n".join(self._captured_output))
    201 self.log.info("PDF successfully created")

LatexFailed: PDF creating failed, captured latex output:
This is BibTeX, Version 0.99d (MiKTeX 23.10)

The top-level auxiliary file: notebook.aux

I found no \citation commands---while reading file notebook.aux

I found no \bibdata command---while reading file notebook.aux

I found no \bibstyle command---while reading file notebook.aux

(There were 3 error messages)



During handling of the above exception, another exception occurred:

ExportFailedException                     Traceback (most recent call last)
c:\Users\user10\Desktop\ME
 ↪203\IntroPythonProgramming\IntroPythonProgramming\Week_4_systems\lec_act_4_sy
 ↪ipynb Cell 21 line 2

      <a href='vscode-notebook-cell:/c%3A/Users/user10/Desktop/ME%20203/
 ↪IntroPythonProgramming/IntroPythonProgramming/Week_4_systems/lec_act_4_system
 ↪ipynb#X26sZmlsZQ%3D%3D?line=0'>1</a> # Save your notebook first, then run thi
 ↪cell to export your submission.
----> <a href='vscode-notebook-cell:/c%3A/Users/user10/Desktop/ME%20203/
 ↪IntroPythonProgramming/IntroPythonProgramming/Week_4_systems/lec_act_4_system
 ↪ipynb#X26sZmlsZQ%3D%3D?line=1'>2</a> grader.export(run_tests=True)

File c:\Users\user10\anaconda3\Lib\site-packages\otter\check\utils.py:184, in
 ↪grading_mode_disabled(wrapped, self, args, kwargs)
    182 if type(self)._grading_mode:
    183     return
--> 184 return wrapped(*args, **kwargs)
```

```
File c:\Users\user10\anaconda3\Lib\site-packages\otter\check\utils.py:166, in␣
 ↪incompatible_with.<locals>.incompatible(wrapped, self, args, kwargs)
    164     else:
    165         return
--> 166 return wrapped(*args, **kwargs)

File c:\Users\user10\anaconda3\Lib\site-packages\otter\check\utils.py:217, in␣
 ↪logs_event.<locals>.event_logger(wrapped, self, args, kwargs)
    215 except Exception as e:
    216     self._log_event(event_type, success=False, error=e)
--> 217     raise e
    219 if ret is None:
    220     ret = LoggedEventReturnValue(None)

File c:\Users\user10\anaconda3\Lib\site-packages\otter\check\utils.py:213, in␣
 ↪logs_event.<locals>.event_logger(wrapped, self, args, kwargs)
    208 """
    209 Runs a method, catching any errors and logging the call. Returns the␣
 ↪unwrapped return value
    210 of the wrapped function.
    211 """
    212 try:
--> 213     ret: Optional[LoggedEventReturnValue[T]] = wrapped(*args, **kwargs)
    215 except Exception as e:
    216     self._log_event(event_type, success=False, error=e)

File c:\Users\user10\anaconda3\Lib\site-packages\otter\check\notebook.py:462, i␣
 ↪Notebook.export(self, nb_path, export_path, pdf, filtering, pagebreaks, files␣
 ↪display_link, force_save, run_tests)
    460 pdf_created = True
    461 if pdf:
--> 462     pdf_path = export_notebook(nb_path, filtering=filtering,␣
 ↪pagebreaks=pagebreaks)
    463     if os.path.isfile(pdf_path):
    464         pdf_created = True

File c:\Users\user10\anaconda3\Lib\site-packages\otter\export\__init__.py:36, i␣
 ↪export_notebook(nb_path, dest, exporter_type, **kwargs)
     33     pdf_name = os.path.splitext(nb_path)[0] + ".pdf"
     35 Exporter = get_exporter(exporter_type=exporter_type)
---> 36 Exporter.convert_notebook(nb_path, pdf_name, **kwargs)
     38 return pdf_name

File c:
 ↪\Users\user10\anaconda3\Lib\site-packages\otter\export\exporters\via_latex.py
 ↪77, in PDFViaLatexExporter.convert_notebook(cls, nb_path, dest, xecjk,␣
 ↪**kwargs)
```

```
    73     if xecjk:
    74         message += "\n\nIf the error above is related to xeCJK or fando
 ↪in LaTeX " \
    75             "and you don't require this functionality, try running agai
 ↪without " \
    76             "xecjk set to True or the --xecjk flag."
---> 77     raise ExportFailedException(message)
    79 finally:
    80     if NBCONVERT_6:

ExportFailedException: There was an error generating your LaTeX; showing full
 ↪error message:
   This is BibTeX, Version 0.99d (MiKTeX 23.10)

   The top-level auxiliary file: notebook.aux

   I found no \citation commands---while reading file notebook.aux

   I found no \bibdata command---while reading file notebook.aux

   I found no \bibstyle command---while reading file notebook.aux

   (There were 3 error messages)
```