question 2 views

Daily Challenge 16.6

(**Due: Friday 11/2 at 12:00 noon Eastern**)

What I cannot code, I do not understand.

(1) Problem: numerical integration.

Write a Python program to approximate the lower and upper sums $L(f,P)$ and $U(f,P)$ given a Python function f and a partition P. Recall that these are given by

$$L(f,P) = \sum_{i=1}^{n} m_i(t_i - t_{i-1}),$$
$$U(f,P) = \sum_{i=1}^{n} M_i(t_i - t_{i-1})$$

where m_i and M_i are the infimum and supremum of f, respectively, on each sub-interval $[t_i, t_{i-1}]$.

I've started the code for your below with some helper functions. **Fill in the rest** to make it work. Then **test your code** on some simple functions and see whether the results make sense.

```python
import numpy as np

def approx_inf_sup(f, a, b, n_tries=10):
    """
    Approximates the infimum and supremum of f on [a,b]

    f: a Python function mapping floats to floats
    a: left endpoint
    b: right endpoint:
    n_tries: optional input, number of points to try

    returns: a tuple (inf, sup) of the approximate inf and sup
    """
    tries = (b-a)*np.random.random(size=n_tries) + a ## Samples points in [a,b)
    f_values = f(tries)

    approx_inf = f_values.min()
    approx_sup = f_values.max()

    return approx_inf, approx_sup

def approx_lower_upper(f, P):
    """
    Approximates L(f,P) and U(f, P)

    f: a Python function mapping floats to floats
    P: a list of points in a partition

    returns: a tuple (inf, sup) of the approximate inf and sup
    """

    my_partition=sorted(P)
    ## Don't assume the user gave you a sorted partition

    lower_sum = 0
    upper_sum = 0

    for t_i, t_i_plus_1 in zip(my_partition[:-1], my_partition[1:]):

        ## your awesome code goes here

    return lower_sum, upper_sum
```

daily_challenge

Updated 5 months ago by Christian Ferko

the students' answer, *where students collectively construct a single answer*

That was as fun as I expected actually.

```
import numpy as np
def findInf(f,a,b):
    infSet = []
    for x in range(101):
        infSet.append(f(a+x*((b-a)/100)))
    return min(infSet)

def findSup(f,a,b):
    infSet = []
    for x in range(101):
        infSet.append(f(a+x*((b-a)/100)))
    return max(infSet)

def cube(x):
    return x*x*x*x

def findInfArea(f,P):
    areaSum = 0
    for index in range(1,(len(P))):
        areaSum += (findInf(f,P[index-1],P[index]) * (P[index]-P[index-1]))
    return areaSum

def findSupArea(f,P):
    areaSum = 0
    for index in range(1,(len(P))):
        areaSum += (findSup(f,P[index-1],P[index]) * (P[index]-P[index-1]))
    return areaSum

partition = np.linspace(0,5,101)
print(str(partition))
print("The area under the curve is: " + str(findInfArea(cube,partition)))
```

Updated 5 months ago by Logan Pachulski

the instructors' answer, *where instructors collectively construct a single answer*

My solution:

```
import numpy as np

def approx_inf_sup(f, a, b, n_tries=10):
    """
    Approximates the infimum and supremum of f on [a,b]

    f: a Python function mapping floats to floats
    a: left endpoint
    b: right endpoint:
    n_tries: optional input, number of points to try

    returns: a tuple (inf, sup) of the approximate inf and sup
    """
    tries = (b-a)*np.random.random(size=n_tries) + a ## Samples points in [a,b)
    f_values = f(tries)

    approx_inf = f_values.min()
    approx_sup = f_values.max()

    return approx_inf, approx_sup

def approx_lower_upper(f, P):
    """
    Approximates L(f,P) and U(f, P)

    f: a Python function mapping floats to floats
    P: a list of points in a partition

    returns: a tuple (inf, sup) of the approximate inf and sup
    """
    my_partition=sorted(P)
    ## Don't assume the user gave you a sorted partition

    lower_sum = 0
    upper_sum = 0

    for t_i, t_i_plus_1 in zip(my_partition[:-1], my_partition[1:]):

        this_inf, this_sup = approx_inf_sup(f, t_i, t_i_plus_1)
        delta_t = t_i_plus_1 - t_i

        lower_sum += this_inf*delta_t
        upper_sum += this_sup*delta_t

    return lower_sum, upper_sum
```

Updated 5 months ago by Christian Ferko

followup discussions *for lingering questions and comments*