HW10: Numerical Differentiation and NR3D

Date Due: 11:59pm, Wed – 2015-04-29

1. **(10 Points)** Write a function with the header **`[df] = myCentralDiff(x,y)`** which approximates the derivative dx/dy for equal sized vectors x and y. The first and last elements of **`df`** should be **`NaN`**.

2. **(10 Points)** Write a function with the header

`[df] = myPartial(f, x, ind, eps)`

which calculates the partial derivative using the central difference method of function **`f`** with respect to **`x(ind)`**. The central difference approximation of a function evaluated at **`x`** is:

$$f' = \frac{f(x+h) - f(x-h)}{2\epsilon}$$

and **`eps`** is used to permute **`x(ind)`** such that

when **`ind`** = 1: $h = \begin{bmatrix} \epsilon \\ 0 \\ 0 \end{bmatrix}$, when **`ind`** = 2: $h = \begin{bmatrix} 0 \\ \epsilon \\ 0 \end{bmatrix}$, and when **`ind`** = 3: $h = \begin{bmatrix} 0 \\ 0 \\ \epsilon \end{bmatrix}$

Test Case:

$f_1(x,y,z) = 9x^2 + 36y^2 + 4z^2 - 36$

$f_2(x,y,z) = x^2 - 2y^2 - 20z$

$f_3(x,y,z) = x^2 - y^2 + z^2$

```
>> f1 = @(x) 9*x(1).^2 + 36*x(2)^2 + 4*x(3).^2 - 36;
>> f2 = @(x) x(1).^2 - 2*x(2).^2 - 20*x(3);
>> f3 = @(x) x(1).^2 - x(2).^2 + x(3).^2;
>> x = [-1; 0; -1];
>> s = myPartial(f1, x, 1, 1e-7)

s =

         -17.9999999971869
>> s = myPartial(f2, x, 3, 1e-7)

s =

         -20.0000000027956
>> s = myPartial(f3, x, 1, 1e-7)

s =

         -2.00000000116773
>> s = myPartial(f3, x, 2, 1e-7)

s =

     0
```

Your can implement this function with three lines of code. (Or fewer.)

3. **(10 Points)** Write a function with the header

```
function [integral, w] = myTrapRule(f, a, b, n)
```

which implements the trapezoidal rule for numerical integration. `w` is the rectangle width as determined by `a`, `b`, and `n`.

Test Case 1:

```
>> f1 = @(x) x.^5 + x.^3./sin(x).^4 + 1;
>> [a,b] = myTrapRule(f1, 1, 2, 1000)

a =
     15.902027561738555

b =
   1.000000000000000e-03
```

Test Case 2:

```
>> f2 = @(bob) sin(bob).*cos(bob).*exp(bob).*tan(bob);
>> [a,b] = myTrapRule(f2, 0, 5, 1000)

a =

     1.024075996301055e+02

b =

   0.005000000000000
1)
```

4. **(10 Points)** Write a function with the header

```
function [integral, w] = myMidPoint(f, a, b, n)
```

which implements the mid-point rule for numerical integration. `w` is the rectangle width as determined by `a`, `b`, and `n`.

Test Case 1:

```
>> f1 = @(x) x.^5 + x.^3./sin(x).^4 + 1;
>> [a,b] = myMidPoint(f1, 1, 2, 1000)

a =
     15.902013422407348

b =
   1.000000000000000e-03
```

Test Case 2:

```
>> f2 = @(bob) sin(bob).*cos(bob).*exp(bob).*tan(bob);
>> [a,b] = myMidPoint(f2, 0, 5, 1000)

a =

     1.024074254688705e+02


b =

   0.005000000000000
```

2)

Deliverables: Submit the following m-files (separately, not zipped) onto Blackboard. **Be sure that the functions are named *exactly* as specified, including spelling and case**.

```
myCentralDif.m
myPartial.m
myTrapRule.m
myMidPoint.m
```