NE250_HW06_mnegus-prob4

December 2, 2017

1 NE 250 – Homework 6

1.1 Problem 4

12/1/2017

```
In [1]: import numpy as np
        import random
```

a.)

We are considering an infinite, steady-state, monoenergetic, two-region Monte Carlo problem with the following characteristics: (note thate we have renamed region 1 and region 2 as region 0 and region 1 respectively; this allows for simpler calculations) * 1-D problem geometry * Region 0 has $\Sigma_s = 0.5\,\text{cm}^{-1}$ and $\Sigma_t = 1.0\,\text{cm}^{-1}$ (in both regions the only interactions are scattering or absorption). * Region 1 has $\Sigma_s = 0.75\,\text{cm}^{-1}$ and $\Sigma_t = 0.9\,\text{cm}^{-1}$. * Region 0 has $w_{nom} = 1$ and Region 1 has $w_{nom} = 2$. w_{max} and w_{min} can be found as $(w_{nom} \times 2.5)$ or $(w_{nom}\,/\,2.5)$, respectively. * All particles are born in Region 0 with weight 1 at a location that is 1 cm to the left of the interface between Regions 0 and 1. The source is isotropic. * Isotropic scattering.

Problem Statement *Write the algorithm for a Monte Carlo code to solve this specific problem. Include the PDFs required for sampling as well as algorithms for conducting sampling. Use a collision estimator to tally the flux. Include implicit capture, rouletting, and splitting.*

1.1.1 Underlying parameters

Using the above information, we can create a set of dictionaries to describe our physical situation. Each fundamental parameter gets a dictionary, and each dictionary has entries corresponding to each region.

```
In [2]: # Macroscopic Total Cross Sections
        Sigma_t = {0:1.0,1:0.9}

        # Macroscopic Scattering Cross Sections
        Sigma_s = {0:0.5,1:0.75}

        # Nominal Weights
        w_nom = {0:1,1:2}
```

```
    # Max/Min Weights
    w_ext = {region:(w_nom[region]*2.5,w_nom[region]/2.5) for region in w_nom.k
```

In [3]: w_ext

Out[3]: {0: (2.5, 0.4), 1: (5.0, 0.8)}

1.1.2 Tracking the particle

We can track a particle using a particle class, which can model the behavior of each particle as it proceeds through it's lifetime. This particle class will contain methods for each action that the particle will undergo * birth (the class' `__init__` method) * transport * boundary encounter * collision * scoring

1.2 Survival Biasing

In [4]:
```
class particle:
    """
    A class to model a single Monte Carlo particle over it's lifetime
    """
    def __init__(self,verbose=False):
        """The particle is born. Assign a position [cm], angle (cos θ), ene
        self.x = -1
        self.mu = 2*random.random()-1
        self.E = 1
        self.w = 1
        self.region = 0
        self.score = np.zeros(2)
        self.verbose = verbose
        if verbose:
            print('The particle was born at x = -1 cm (region 0), with weig

    def transport(self,sample=True):
        """Transport the particle through the problem geometry"""
        # Sample to find the number of mean free paths that traveled by the
        if sample:
            xi = random.random()
            self.mfp_x = -np.log(xi)*self.mu
            if self.verbose: print('This particle will travel {} MFPs in th
        # Determine the number of mean free paths to a boundary in the curr
        boundary_mfp = -self.x*Sigma_t[self.region]
        if self.verbose: print('The distance to the boundary is {} MFPs.'.f
        # If the particle reaches a boundary before the collision, stop and
        if boundary_mfp == 0:  # (the particle was at the boundary, so must
            if self.verbose: print('(the particle is at the boundary)')
            self.collision()
        elif self.mfp_x > boundary_mfp:
            self.boundary(boundary_mfp)
```

```python
        else:
            self.collision()

    def boundary(self,boundary_mfp):
        """The particle reached a boundary: reevaluate the particle's track
        if self.verbose: print('The particle travels to the boundary.')
        self.x = 0
        self.region = 1-self.region
        self.mfp_x -= boundary_mfp
        self.update_weight(collision=False)
        self.transport(sample=False)

    def collision(self):
        """The particle collided: use survival biasing to continue followi
        self.x += self.mfp_x
        if self.verbose: print('The particle collides at x = {}'.format(rou
        self.score_particle()
        self.update_weight(collision=True)
        if self.w == 0:
            if self.verbose: print('\t\t\t\t The particle is killed (w=0).'
            return
        self.mu = 2*random.random()-1
        if self.verbose: print('The particle scatters, and is now traveling
        self.transport(sample=True)

    def update_weight(self,collision=False):
        """Update the particle's weight using survival biasing, splitting,
        if collision: # adjust weight with survival biasing
            # New weight:     w_{i+1} = w(1 - Σ_a / Σ_t)
            self.w *= (Sigma_t[self.region]-Sigma_s[self.region])/Sigma_t[s
            if self.verbose: print('The particle is survival biased, now wi
        # Splitting
        if self.w > w_ext[self.region][0]:
            SR = self.w/w_nom[self.region]
            xi = random.random()
            if xi >= SR - int(SR):
                b = 0
            else:
                b = 1
            n_new_particles = int(SR) + b
            if self.verbose: print('Splitting the particle into {} new part
            global particles
            for i in range(int(SR)):
                particles.append(particle())
                particles[-1].x = self.x
                particles[-1.].w = self.w/n_new_particles
        # Russian Roulette
        if self.w < w_ext[self.region][1]:
```

```
                    if self.verbose: print('Rouletting the particles...')
                    xi = random.random()
                    RR = self.w/w_nom[self.region]
                    if xi >= RR :
                        self.w = 0
                        if self.verbose: print('\t\t\t ... uh-oh...')
                    else:
                        self.w = w_nom[self.region]
                        if self.verbose: print('Phew. Particle survived, and now it

        def score_particle(self):
            self.score[self.region] += self.w
            if self.verbose:
                print('Score! Particle with weight {} added to the tally.'.form
                print('\tCurrent score: \t Region 0: {}    Region 1: {}'.format
```

```
In [5]: ## Survival biasing, rouletting, and splitting MC run
        N = 20000
        tally = np.zeros(2)
        particles = [particle() for n in range(N)]
        for p in particles:
            p.transport()
            tally += p.score
        norm_collisions = tally/N
```

```
In [6]: print('Average collisions per particle, region 0: {} \t Average collisions
        print('Ratio: ',norm_collisions[0]/norm_collisions[1])
```

```
Average collisions per particle, region 0: 1.7315          Average collisions per p
Ratio:  10.2698695136
```

Interestingly, it appears that in our specific problem splitting is never used. This fact could be deduced, however, by noting that splitting only occurs when $w_i > w_{max}$. For both regions, $w_{max} \geq 2.5$. Upon birth in either region, a particle's weight will never be more than 2. Collisions will only trigger a reduction in weight, and rouletting produces particles also with a maximum weight of 2 (roulette products have weight w_{nom}).

1.3 The story of a Monte Carlo Particle

b.)

Let's follow just 1 particle.

```
In [7]: random.seed(1)
        N = 1
        tally = np.zeros(2)
        p = particle(verbose=True)
```

```
        p.transport()
        tally += p.score
```

```
The particle was born at x = -1 cm (region 0), with weight 1.
This particle will travel -0.121 MFPs in the x-direction.
The distance to the boundary is 1.0 MFPs.
The particle collides at x = -1.121
Score! Particle with weight 1 added to the tally.
        Current score:          Region 0: 1.0   Region 1: 0.0
The particle is survival biased, now with weight 0.5
The particle scatters, and is now traveling with mu = 0.528
This particle will travel 0.721 MFPs in the x-direction.
The distance to the boundary is 1.121 MFPs.
The particle collides at x = -0.4
Score! Particle with weight 0.5 added to the tally.
        Current score:          Region 0: 1.5   Region 1: 0.0
The particle is survival biased, now with weight 0.25
Rouletting the particles...
                         ... uh-oh...
                                 The particle is killed (w=0).
```

Random seed 4 gives a much more dynamic plot...

```
In [8]: random.seed(4)
        N = 1
        tally = np.zeros(2)
        p = particle(verbose=True)
        p.transport()
        tally += p.score
```

```
The particle was born at x = -1 cm (region 0), with weight 1.
This particle will travel -1.199 MFPs in the x-direction.
The distance to the boundary is 1.0 MFPs.
The particle collides at x = -2.199
Score! Particle with weight 1 added to the tally.
        Current score:          Region 0: 1.0   Region 1: 0.0
The particle is survival biased, now with weight 0.5
The particle scatters, and is now traveling with mu = -0.208
This particle will travel -0.388 MFPs in the x-direction.
The distance to the boundary is 2.199 MFPs.
The particle collides at x = -2.587
Score! Particle with weight 0.5 added to the tally.
        Current score:          Region 0: 1.5   Region 1: 0.0
The particle is survival biased, now with weight 0.25
Rouletting the particles...
Phew. Particle survived, and now it has weight 1
The particle scatters, and is now traveling with mu = -0.197
This particle will travel -0.017 MFPs in the x-direction.
```

```
The distance to the boundary is 2.587 MFPs.
The particle collides at x = -2.604
Score! Particle with weight 1 added to the tally.
        Current score:          Region 0: 2.5   Region 1: 0.0
The particle is survival biased, now with weight 0.5
The particle scatters, and is now traveling with mu = 0.601
This particle will travel 0.161 MFPs in the x-direction.
The distance to the boundary is 2.604 MFPs.
The particle collides at x = -2.443
Score! Particle with weight 0.5 added to the tally.
        Current score:          Region 0: 3.0   Region 1: 0.0
The particle is survival biased, now with weight 0.25
Rouletting the particles...
Phew. Particle survived, and now it has weight 1
The particle scatters, and is now traveling with mu = 0.073
This particle will travel 0.094 MFPs in the x-direction.
The distance to the boundary is 2.443 MFPs.
The particle collides at x = -2.348
Score! Particle with weight 1 added to the tally.
        Current score:          Region 0: 4.0   Region 1: 0.0
The particle is survival biased, now with weight 0.5
The particle scatters, and is now traveling with mu = -0.655
This particle will travel -1.468 MFPs in the x-direction.
The distance to the boundary is 2.348 MFPs.
The particle collides at x = -3.817
Score! Particle with weight 0.5 added to the tally.
        Current score:          Region 0: 4.5   Region 1: 0.0
The particle is survival biased, now with weight 0.25
Rouletting the particles...
Phew. Particle survived, and now it has weight 1
The particle scatters, and is now traveling with mu = 0.855
This particle will travel 0.16 MFPs in the x-direction.
The distance to the boundary is 3.817 MFPs.
The particle collides at x = -3.656
Score! Particle with weight 1 added to the tally.
        Current score:          Region 0: 5.5   Region 1: 0.0
The particle is survival biased, now with weight 0.5
The particle scatters, and is now traveling with mu = 0.613
This particle will travel 0.137 MFPs in the x-direction.
The distance to the boundary is 3.656 MFPs.
The particle collides at x = -3.52
Score! Particle with weight 0.5 added to the tally.
        Current score:          Region 0: 6.0   Region 1: 0.0
The particle is survival biased, now with weight 0.25
Rouletting the particles...
Phew. Particle survived, and now it has weight 1
The particle scatters, and is now traveling with mu = -0.38
This particle will travel -0.178 MFPs in the x-direction.
```

```
The distance to the boundary is 3.52 MFPs.
The particle collides at x = -3.697
Score! Particle with weight 1 added to the tally.
        Current score:          Region 0: 7.0   Region 1: 0.0
The particle is survival biased, now with weight 0.5
The particle scatters, and is now traveling with mu = 0.464
This particle will travel 0.073 MFPs in the x-direction.
The distance to the boundary is 3.697 MFPs.
The particle collides at x = -3.624
Score! Particle with weight 0.5 added to the tally.
        Current score:          Region 0: 7.5   Region 1: 0.0
The particle is survival biased, now with weight 0.25
Rouletting the particles...
                         ... uh-oh...
                                 The particle is killed (w=0).
```