Geoffrey Woollard

My code lives in the repo https://github.com/geoffwoollard/prob_prog

Acknowledgments

- discussions with Jordan Lovrod, Ilias Karimalis, Gaurav Bhatt
- starter code for smc.resample_particles and smc.SMC from Masoud Mokhtari

```
In [1]:
         from dill.source import getsource, getsourcelines
In [2]:
         from smc import resample particles, SMC
         list_of_programs = [resample_particles, SMC]
         for program in list_of_programs:
             for line_number, function_line in enumerate(getsourcelines(program)[0]):
                 print(line_number, function_line,end='')
             print()
        0 def resample_particles(particles, log_weights):
              Eq. 4.24 in course textbook (https://arxiv.org/abs/1809.10756v2, pp. 122)
        3
              See Algorithm 15 in the course textbook, section 6.7 Sequantial Monte Carlo, p. 176
        4
              log weights = tensor(log weights)
        5
        6
              n_particles = log_weights.size().numel()
        7
        8
              unnormalized_particle_weights = torch.exp(log_weights).detach().numpy()
        9
        10
              particle idxs = np.random.choice(
        11
                   a=range(n particles),
                   size=n particles,
        13
                   p=unnormalized_particle_weights/unnormalized_particle_weights.sum(),
                   replace=True,
        14
        15
              #print('particle idxs',particle idxs)
        16
        17
        18
              new particles = []
        19
              for idx in range(n particles):
        20
                   new_particles.append(particles[particle_idxs[idxs]]) # TODO: copy?
        21
        22
               log_Z = np.log(np.sum(unnormalized_particle_weights)/n_particles)
        23
               return log_Z, new_particles
        0 def SMC(n_particles, exp,do_log=False):
        2
              particles = []
        3
              weights = []
        4
              logZs = []
        5
              output = lambda x: x
        6
        7
              for i in range(n_particles):
        8
        9
                  res = evaluate(exp, env=None)('addr_start', output)
        10
                   logW = 0.
        11
        12
        13
                  particles.append(res)
        14
                   weights.append(logW)
        15
        16
               #can't be done after the first step, under the address transform, so this should be fine:
        17
               done = False
        18
               smc cnter = 0
        19
               while not done:
                   if do_log: print('In SMC step {}, Zs: '.format(smc cnter), logZs)
        20
        21
                   for i in range(n particles): #Even though this can be parallelized, we run it serially
        22
                        res = run until observe or end(particles[i]) # particle i at next breakbpoint
        23
                        if 'done' in res[2]: #this checks if the calculation is done
        24
                           particles[i] = res[0]
        25
                           if i == 0:
                                done = True #and enforces everything to be the same as the first particle
        26
        27
                                address = ''
        28
                           else:
        29
                                if not done: # triggered when i=0 was not done and i>0 was done
        30
                                    raise RuntimeError('Failed SMC, finished one calculation before the other')
        31
                       else:
        32
                           #TODO: check particle addresses, and get weights and continuations
                           particles[i] = res
        33
                           cont, args, sigma = res
        34
        35
                           assert 'observe' == sigma['type']
        36
                           weights[i] = sigma['distribution'].log_prob(sigma['observed_constant'])
        37
        38
                             # check particle addresses
        39
                           if i == 0:
        40
                                break_point_address = sigma['address']
        41
                           else:
        42
                                if sigma['address'] != break point address:
        43
                                    assert False, 'particles at different break points'
        44
        45
        46
        47
                   if not done:
        48
                       #resample and keep track of logZs
        49
                       logZn, particles = resample_particles(particles, weights)
        50
                       logZs.append(logZn)
                   smc cnter += 1
        51
        52
               logZ = sum(logZs)
        53
               return logZ, particles
```

```
In [2]:
            from daphne import daphne
            import os, json
            import numpy as np
            import torch
            from torch import tensor
            import pandas as pd
            import matplotlib.pyplot as plt
 In [3]:
            def ast helper(fname, directory):
                 sugared_fname = '../prob_prog/hw/hw6/CS532-HW6/{}'.format(directory, fname)
                 desugared_ast_json_fname = '/Users/gw/repos/prob_prog/' + sugared_fname.replace('.daphne','.json')
                 if os.path.isfile(desugared_ast_json_fname):
                      with open(desugared_ast_json_fname) as f:
                          ast = json.load(f)
                 else:
                      #note: the sugared path that goes into daphne desugar should be with respect to the daphne path!
                      ast = daphne(['desugar-hoppl-cps', '-i', sugared_fname])
                      with open(desugared_ast_json_fname, 'w') as f:
                          json.dump(ast, f)
                 return ast
            i=1
            fname = '{}.daphne'.format(i)
            exp = ast_helper(fname, directory='programs')
            %cat programs/1.daphne
            (defn until-success [p n]
               (if (sample (flip p))
                 (until-success p (+ n 1))))
            (let [p 0.01]
              (until-success p 0))
 In [8]:
            import smc, evaluator
            import importlib
            importlib.reload(smc)
           <module 'smc' from '/Users/gw/repos/prob prog/hw/hw6/CS532-HW6/smc.py'>
 Out[8]:
In [11]:
            output = lambda x: x
            evaluator.evaluate(exp, env=None)('addr start', output)
           (<function primitives.push addr(alpha, value, k)>,
Out[11]:
            ['addr_start', '0', <evaluator.Procedure at 0x135e73640>],
            {'type': 'proc'})
In [14]:
            n_particles=3
            logZ, particles = smc.SMC(n_particles, exp)
            particles
           In SMC step 0, Zs: []
           [tensor(42), tensor(122), tensor(88)]
Out[14]:
          Note that there are no observed in this program, and thus Z is undefined
In [15]:
            particle_counts = [1,10,100,1000,10000,100000]
            fig, axes = plt.subplots(nrows=len(particle counts),figsize=(30,20))
            # fig.tight_layout()
            plt.subplots adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.5) # https://stackoveri
            for idx, n_particles in enumerate(particle_counts):
                 logZ, particles = smc.SMC(n_particles, exp)
                 samples_array = np.array([sample.item() for sample in particles])
                 mean = samples_array.mean()
                 var = samples_array.var()
                 pd.Series(samples_array).plot.hist(ax=axes[idx], bins=50, title='Program {} | {} particles | mean {:1.3f}
           In SMC step 0, Zs: []
           In SMC step 0, Zs:
                                  []
                                 []
           In SMC step 0, Zs:
           In SMC step 0, Zs: []
           In SMC step 0, Zs: []
           In SMC step 0, Zs: []
                                                    Program 1.daphne | 1 particles | mean 457.000 | var / std 0.000e+00 / 0.000e+00 | Evidence: logZ 0.000 / Z 1.000e+00
            0.6
             0.2
             0.0
                                                    Program 1.daphne | 10 particles | mean 63.200 | var / std 4.400e+03 / 6.633e+01 | Evidence: logZ 0.000 / Z 1.000e+00
            j 10
                                                   Program 1.daphne | 100 particles | mean 100.150 | var / std 8.675e+03 / 9.314e+01 | Evidence: logZ 0.000 / Z 1.000e+00
          12..
10.0
7
                                                   Program 1.daphne | 1000 particles | mean 93.865 | var / std 9.031e+03 / 9.503e+01 | Evidence: logZ 0.000 / Z 1.000e+00
             100
                                                   Program 1.daphne | 10000 particles | mean 99.084 | var / std 1.031e+04 / 1.015e+02 | Evidence: logZ 0.000 / Z 1.000e+00
            1500
            2 1000
                                                  Program 1.daphne | 100000 particles | mean 98.877 | var / std 9.891e+03 / 9.945e+01 | Evidence: logZ 0.000 / Z 1.000e+00
            15000
           를 10000
```

```
In [2]:
          from daphne import daphne
          import os, json
          import numpy as np
          import torch
          from torch import tensor
          import pandas as pd
          import matplotlib.pyplot as plt
In [3]:
          def ast helper(fname, directory):
              sugared fname = '../prob prog/hw/hw6/CS532-HW6/{}/{}'.format(directory,fname)
              desugared ast json fname = '/Users/gw/repos/prob prog/' + sugared fname.replace('.daphne','.json')
              if os.path.isfile(desugared ast json fname):
                   with open (desugared ast json fname) as f:
                       ast = json.load(f)
                   #note: the sugared path that goes into daphne desugar should be with respect to the daphne path!
                   ast = daphne(['desugar-hoppl-cps', '-i', sugared fname])
                   with open(desugared ast json fname, 'w') as f:
                       json.dump(ast, f)
              return ast
          fname = '{}.daphne'.format(i)
          exp = ast helper(fname, directory='programs')
          %cat programs/2.daphne
         (defn marsaglia-normal [mean var]
            (let [d (uniform-continuous -1.0 1.0)
                   x (sample d)
                   y (sample d)
                   s (+ (* x x ) (* y y ))]
              (if (< s 1)
                  (+ mean (* (sqrt var)
                              (* x (sqrt (* -2 (/ (log s) s))))))
                  (marsaglia-normal mean var))))
         (let [mu (marsaglia-normal 1 5)
               sigma (sqrt 2)
               lik (normal mu sigma)]
           (observe lik 8)
           (observe lik 9)
In [4]:
          import smc
          import importlib
          importlib.reload(smc)
         <module 'smc' from '/Users/gw/repos/prob_prog/hw/hw6/CS532-HW6/smc.py'>
Out[4]:
In [5]:
          n particles=3
          logZ, particles = smc.SMC(n particles, exp)
         In SMC step 0, Zs: []
         In SMC step 1, Zs: [-8.746667201168327]
         In SMC step 2, Zs: [-8.746667201168327, -11.342506857689752]
In [7]:
          particle counts = [1,10,100,1000,10000,100000]
          fig, axes = plt.subplots(nrows=len(particle_counts),figsize=(30,20))
          # fig.tight layout()
          plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.5) # https://stackoveri
          for idx, n particles in enumerate(particle counts):
             logZ, particles = smc.SMC(n particles, exp)
              samples_array = np.array([sample.item() for sample in particles])
              mean = samples array.mean()
              var = samples array.var()
              pd.Series(samples array).plot.hist(ax=axes[idx], bins=50, title='Program {} | {} particles | mean {:1.3f}
         In SMC step 0, Zs: []
         In SMC step 1, Zs: [-20.574844409164438]
         In SMC step 2, Zs: [-20.574844409164438, -25.219081845127047]
         In SMC step 0, Zs:
                              []
                               [-3.8916753382007996]
         In SMC step 1, Zs:
         In SMC step 2, Zs:
                              [-3.8916753382007996, -2.641245161237912]
         In SMC step 0, Zs:
                              []
                              [-5.708884842017928]
         In SMC step 1, Zs:
         In SMC step 2, Zs:
                              [-5.708884842017928, -3.9738090087117763]
                              []
         In SMC step 0, Zs:
                              [-5.07073953887065]
         In SMC step 1, Zs:
         In SMC step 2, Zs: [-5.07073953887065, -2.9832770513027356]
         In SMC step 0, Zs: []
                              [-5.4072599330000415]
         In SMC step 1, Zs:
         In SMC step 2, Zs: [-5.4072599330000415, -2.9642962996086535]
         In SMC step 0, Zs: []
         In SMC step 1, Zs:
                              [-5.382312301483098]
         In SMC step 2, Zs: [-5.382312301483098, -2.790267592243308]
                                              Program 2.daphne | 1 particles | mean -0.788 | var / std 0.000e+00 / 0.000e+00
          0.8
          0.6
          0.4
0.4
          0.0
                                               Program 2.daphne | 10 particles | mean 6.744 | var / std 0.000e+00 / 0.000e+00
                                               Program 2.daphne | 100 particles | mean 6.107 | var / std 1.687e-01 / 4.108e-01
                                              Program 2.daphne | 1000 particles | mean 7.178 | var / std 1.024e+00 / 1.012e+00
         j 100
                                              Program 2.daphne | 10000 particles | mean 7.042 | var / std 5.960e-01 / 7.720e-01
                                              Program 2.daphne | 100000 particles | mean 7.326 | var / std 8.845e-01 / 9.405e-01
          6000
          4000
```

```
In [20]:
                             from daphne import daphne
                             import os, json
                             import numpy as np
                             import torch
                             from torch import tensor
                             import pandas as pd
                             import matplotlib.pyplot as plt
                             import seaborn as sns
   In [2]:
                             def ast helper(fname, directory):
                                        sugared fname = '../prob prog/hw/hw6/CS532-HW6/{}/{}'.format(directory,fname)
                                        desugared_ast_json_fname = '/Users/gw/repos/prob_prog/' + sugared_fname.replace('.daphne','.json')
                                        if os.path.isfile(desugared ast json fname):
                                                    with open (desugared ast json fname) as f:
                                                               ast = json.load(f)
                                        else:
                                                   #note: the sugared path that goes into daphne desugar should be with respect to the daphne path!
                                                   ast = daphne(['desugar-hoppl-cps', '-i', sugared fname])
                                                    with open (desugared ast json fname, 'w') as f:
                                                              json.dump(ast, f)
                                        return ast
                             i=3
                             fname = '{}.daphne'.format(i)
                             exp = ast helper(fname, directory='programs')
                             %cat programs/3.daphne
                            (defn reduce [f x values]
                                                                     (if (empty? values)
                                                                              (reduce f (f x (first values))) (rest values))))
                            (let [observations [0.9 0.8 0.7 0.0 -0.025 -5.0 -2.0 -0.1 0.0 0.13 0.45 6 0.2 0.3 -1 -1]
                                            init-dist (discrete [1.0 1.0 1.0])
                                            trans-dists {0 (discrete [0.1 0.5 0.4])
                                                                                 1 (discrete [0.2 0.2 0.6])
                                                                                 2 (discrete [0.15 0.15 0.7])}
                                            obs-dists {0 (normal -1 1)
                                                                           1 (normal 1 1)
                                                                           2 (normal 0 1)}]
                                             (reduce
                                                  (fn [states obs]
                                                       (let [state (sample (get trans-dists
                                                                                                                              (peek states)))]
                                                              (observe (get obs-dists state) obs)
                                                             (conj states state)))
                                                  [(sample init-dist)]
                                                  observations))
In [64]:
                             import smc, evaluator
                             import importlib
                             importlib.reload(smc)
                           <module 'smc' from '/Users/gw/repos/prob prog/hw/hw6/CS532-HW6/smc.py'>
Out[64]:
In [75]:
                             %%time
                             n particles=100
                             logZ, particles = smc.SMC(n particles, exp)
                             samples array = torch.stack(particles).detach().numpy()
                             samples array.mean(0)
                           CPU times: user 2.97 s, sys: 10.2 ms, total: 2.98 s
                           Wall time: 2.98 s
                           array([ 1.420, 1.600, 1.470, 1.480, 1.030, 1.540, 1.930, 1.930,
                                                  1.730,
                                                                                            0.260, 1.750, 1.670, 1.930, 2.000, 2.000,
                                                  0.940])
In [76]:
                             %%time
                             #40s / 1k samples
                             particle counts = [1,10,100,1000,10000,100000]
                             fig, axes = plt.subplots(nrows=len(particle counts), figsize=(30,20))
                             plt.subplots adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=1.75) # https://stackovel
                             np.set printoptions(formatter={'float': '{: 1.3f}'.format}) # https://stackoverflow.com/questions/2891790/how-
                             for idx, n particles in enumerate(particle counts):
                                        logZ, particles = smc.SMC(n particles, exp)
                                         samples array = torch.stack(particles).detach().numpy()
                                         # mean var
                                        mean = samples array.mean(0)
                                        var = samples array.var(0)
                                        title='Program \{\}\ |\ \{\}\ particles \ n \ mean <math>\{\}\ n \ title= \ n \
                                                    fname, n particles, mean, var, np.sqrt(var), logZ, np.exp(logZ))
                                        df = pd.DataFrame(samples array)
                                        df wide = pd.melt(df.reset index(),id vars='index')
                                        ax1=sns.countplot(x="value", hue="variable", data=df wide,ax=axes[idx])
                                        axes[idx].set title(title)
                                        if idx == 0:
                                                    ax1.legend(bbox to anchor=(1.2, 1), loc='upper right', borderaxespad=0,fontsize=20)
                                        else:
                                                   axes[idx].legend([],[], frameon=False)
                           CPU times: user 1h 4min 59s, sys: 18 s, total: 1h 5min 17s
                           Wall time: 1h 5min 26s
                                                                                                                                Program 3 daphne | 1 particles
mean [ 2.000 1.000 0.000 2.000 2.000 2.000 2.000 2.000 2.000
2.000 2.000 2.000 2.000 0.000 2.000 1.000]
var [ 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
std [ 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
Evidence: logZ -52.953 / Z 1.006e-23
                                                                                                                                                                                                                                                                                                                                          0
                                                                                                                                                                                                                                                                                                                                          1
                                                                                                                                                                                                                                                                                                                                          2
                                                                                                                                Program 3 daphne | 10 particles
mean [ 1.00 1.400 2.000 1.600 2.000 2.000 1.700 1.000 1.000 0.000 0.000 2.000 2.000 2.000 2.000 2.000 2.000 2.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.00
                                                                                                                                                                                                                                                                                                                                          3
                                                                                                                                                                                                                                                                                                                                          4
                                                                                                                                                                                                                                                                                                                                          ____5
                                                                                                                                                                                                                                                                                                                                          6
                                                                                                                                                                                                                                                                                                                                          7
                                                                                                                                                                                                                                                                                                                                          8
                                                                                                                                 Program 3 daphrie | 100 particles
mean [1.410 1.570 1.720 1.810 1.900 1.160 1.780 1.610 1.740 1.140
0.300 1.690 1.530 1.920 1.520 1.600 1.550]
var [0.802 0.645 0.242 0.154 0.000 0.694 0.232 0.538 0.252 0.980
0.510 0.214 0.569 0.074 0.730 0.560 0.247]
std [0.895 0.803 0.492 0.392 0.000 0.833 0.481 0.733 0.502 0.990
0.714 0.462 0.754 0.271 0.834 0.748 0.497]
Evidence: logz -44.510 / Z 4.675e-20
                                                                                                                                                                                                                                                                                                                                         13
                                                                                                                                                                                                                                                                                                                                         15
                                                                                                                                mean [ 1.355 1.636 1.729 1.590 1.625 1.481 1.557 1.635 1.610 1.074 0.771 1.688 1.766 1.722 1.671 1.464 0.947] var [ 0.817 0.552 0.284 0.480 0.24 0.560 0.487 0.438 0.344 0.949] std [ 0.904 0.749 0.533 0.632 0.156 0.748 0.698 0.662 0.586 0.974 0.535 0.632 0.625 0.625 0.343 0.550 0.789] Evidence: logZ -44.174 / Z 6.538e-20
                                                                                                                                Program 3 daphne | 10000 particles
mean [ 1.417 1.544 1.709 1.613 1,620 1.442 1.675 1.668 1.619 1.063
0.172 1.712 1.682 1.754 1.677 1.526 0.889]
var [ 0.763 0.657 0.313 0.455 0.02 0.613 0.406 0.421 0.422 0.958
0.314 0.330 0.407 0.242 0.340 0.322 0.701]
std [ 0.873 0.810 0.559 0.674 0.141 0.783 0.637 0.649 0.650 0.979
0.505 0.574 0.638 0.452 0.538 0.567 0.837]
Evidence: log2 -44.442 / Z 5.004e-20
                                                                                                                                Program 3 daphnej 100000 particles
mean [1.424 | 1.549 | 1.708 | 1.611 | 1,004 | 1.421 | 1.651 | 1.683 | 1.593 | 1.030
0.133 | 1.677 | 1.673 | 1.687 | 1.636 | 1.517 | 0.938 |
var [0.759 | 0.651 | 0.316 | 0.459 | 0.016 | 0.042 | 0.423 | 0.430 | 0.432 | 0.954
0.248 | 0.371 | 0.430 | 0.302 | 0.345 | 0.328 | 0.690 |
std [0.871 | 0.807 | 0.552 | 0.677 | 0.152 | 0.776 | 0.651 | 0.635 | 0.657 | 0.977
0.498 | 0.699 | 0.656 | 0.549 | 0.587 | 0.373 | 0.8311 |
Evidence: 0.022 | 4.44 | 2.67 | 2.5084 | 2.08
```

```
In [11]:
            from daphne import daphne
            import os, json
            import numpy as np
            import torch
            from torch import tensor
            import pandas as pd
            import matplotlib.pyplot as plt
In [34]:
            def ast helper(fname, directory):
                sugared fname = '../prob prog/hw/hw6/CS532-HW6/{}/{}'.format(directory,fname)
                desugared ast json fname = '/Users/gw/repos/prob_prog/' + sugared_fname.replace('.daphne','.json')
                if os.path.isfile(desugared ast json fname):
                     with open(desugared_ast_json_fname) as f:
                          ast = json.load(f)
                     #note: the sugared path that goes into daphne desugar should be with respect to the daphne path!
                     ast = daphne(['desugar-hoppl-cps', '-i', sugared fname])
                     with open (desugared ast json fname, 'w') as f:
                          json.dump(ast, f)
                return ast
            fname = '{}.daphne'.format(i)
            exp = ast helper(fname, directory='programs')
            %cat programs/4.daphne
           (let [mu (sample (normal 1 (sqrt 5)))
                        sigma (sqrt 2)
                        lik (normal mu sigma)]
                   (observe lik 8)
                   (observe lik 9)
                   mu)
In [35]:
            import smc
            import importlib
            importlib.reload(smc)
           <module 'smc' from '/Users/gw/repos/prob_prog/hw/hw6/CS532-HW6/smc.py'>
Out[35]:
In [45]:
            particle counts = [1,10,100,1000,10000,100000]
            fig, axes = plt.subplots(nrows=len(particle counts), figsize=(30,20))
            # fig.tight layout()
            plt.subplots adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=0.5) # https://stackoveri
            for idx, n particles in enumerate(particle counts):
                logZ, particles = smc.SMC(n particles, exp)
                samples array = np.array([sample.item() for sample in particles])
                mean = samples array.mean()
                var = samples array.var()
                pd.Series(samples_array).plot.hist(ax=axes[idx], bins=50, title='Program {} | {} particles | mean {:1.3f}
           In SMC step 0, Zs: []
           In SMC step 1, Zs: [-5.426987590543291]
           In SMC step 2, Zs: [-5.426987590543291, -7.7169570978697495]
           In SMC step 0, Zs: []
           In SMC step 1, Zs: [-4.409021968208568]
           In SMC step 2, Zs: [-4.409021968208568, -3.281747838494585]
           In SMC step 0, Zs: []
           In SMC step 1, Zs: [-5.44325713520603]
           In SMC step 2, Zs: [-5.44325713520603, -2.7813596798573204]
           In SMC step 0, Zs: []
           In SMC step 1, Zs: [-5.348063793569588]
           In SMC step 2, Zs: [-5.348063793569588, -3.34402504427412]
           In SMC step 0, Zs: []
           In SMC step 1, Zs: [-5.384503472706044]
           In SMC step 2, Zs: [-5.384503472706044, -2.8596724029796277]
           In SMC step 0, Zs: []
           In SMC step 1, Zs: [-5.406247981671558]
           In SMC step 2, Zs: [-5.406247981671558, -2.857562885985456]
                                                  Program 4.daphne | 1 particles | mean 3.920 | var / std 0.000e+00 / 0.000e+00 | Evidence: logZ -13.144 / Z 1.957e-06
           Frequency
0.4
                                                   Program 4.daphne | 10 particles | mean 6.160 | var / std 0.000e+00 / 0.000e+00 | Evidence: logZ -7.691 / Z 4.570e-04
                                                   Program 4.daphne | 100 particles | mean 6.865 | var / std 1.919e-01 / 4.380e-01 | Evidence: logZ -8.225 / Z 2.680e-04
                                                  Program 4.daphne | 1000 particles | mean 6.497 | var / std 2.420e-01 / 4.919e-01 | Evidence: logZ -8.692 / Z 1.679e-04
           텔 150
                                                  Program 4.daphne | 10000 particles | mean 7.209 | var / std 8.558e-01 / 9.251e-01 | Evidence: logZ -8.244 / Z 2.628e-04
                                                  Program 4.daphne | 100000 particles | mean 7.225 | var / std 7.930e-01 / 8.905e-01 | Evidence: logZ -8.264 / Z 2.577e-04
           들 4000
```