```
In [1]: run p1.py
```

```
**********
dt1: 1.0 tf1: 36000.0 df1: 324310917.977126
dt2: 0.5 tf2: 36000.0 df2: 324813020.5007068
delta t: 0.5
final distance: 324813020.5007068
error: -502102.52358078957
relative error: -0.0015458201854309502
**********
dt1: 0.25 tf1: 36000.0 df1: 325063935.01676553
dt2: 0.125 tf2: 36000.0 df2: 325189358.1714124
delta t: 0.125
final distance: 325189358.1714124
error: -125423.15464687347
relative error: -0.00038569267872769984
**********
dt1: 0.0625 tf1: 36000.0 df1: 325252061.2334843
dt2: 0.03125 tf2: 36000.0 df2: 325283410.63699913
delta t: 0.03125
final distance: 325283410.63699913
error: -31349.403514802456
relative error: -9.637566039230604e-05
```

The smallest delta t I could calculate is as follows (it took a while)
dt1: 0.015625 tf1: 36000.0 df1: 325299084.8071171
dt2: 0.0078125 tf2: 36000.0 df2: 325306921.75922555
delta t: 0.0078125
final distance: 325306921.75922555
error: -7836.952108442783
relative error: -2.409094791485337e-05

That's
$\Delta t = 0.0078125$
$d(36000) = 325306921.75922555$

I cut Δt in half every time and noted the difference from the previous answer as error. Notice that $error$ and Δt are being cut in half every time (since error is linear, Δt and $error$ are directly proportional, so this makes sense). Thus,
$\frac{|error_0|}{2^n} < 1000 \implies log_2(\frac{error_0}{1000}) < n$
$error_0 = -502102.5$
$\implies log_2(\frac{error_0}{1000}) = 8.97 < 9 = n$
In order to achieve an accuracy less than 1000 meters, we need
$\Delta t = \frac{\Delta t_0}{2^n}$ where $\Delta t_0 = 0.5$
$\implies \Delta t = \frac{.5}{2^9} = \frac{1}{2^{10}} = \frac{1}{1024} < 0.001$

```
In [1]: run p1_b.py
```

```
Expected value: 325306921.75922555
dt: 200.0 d1(36000): 72414136.96110632 err: -252892784.79811925
dt: 210.0 d2(36000): 250667465.28667754 err: -74639456.47254801
d1 more accurate than d2: False
dt: 100.0 d1(36000): 217164438.65054289 err: -108142483.10868266
dt: 105.0 d2(36000): 385529825.5312745 err: 60222903.77204895
d1 more accurate than d2: False
dt: 66.66666666666667 d1(36000): 255035831.8619846 err: -70271089.89724
094
dt: 70.0 d2(36000): 434429513.1276867 err: 109122591.36846113
d1 more accurate than d2: True
dt: 50.0 d1(36000): 273247626.2796922 err: -52059295.479533374
dt: 52.5 d2(36000): 242282579.4292694 err: -83024342.32995614
d1 more accurate than d2: True
dt: 40.0 d1(36000): 283968596.50312746 err: -41338325.25609809
dt: 42.0 d2(36000): 293943303.1236355 err: -31363618.635590076
d1 more accurate than d2: False
dt: 33.333333333333336 d1(36000): 291031076.07875544 err: -34275845.680
47011
dt: 35.0 d2(36000): 329030518.6313102 err: 3723596.872084677
d1 more accurate than d2: False
dt: 28.571428571428573 d1(36000): 296033944.4591349 err: -29272977.3000
9067
dt: 30.0 d2(36000): 294536752.51838416 err: -30770169.24084139
d1 more accurate than d2: True
dt: 25.0 d1(36000): 299763118.7767933 err: -25543802.982432246
dt: 26.25 d2(36000): 320678553.22968876 err: -4628368.529536784
d1 more accurate than d2: False
dt: 22.2222222222222 d1(36000): 325324089.1149546 err: 17167.355729043
484
dt: 23.333333333333332 d2(36000): 341430927.52462715 err: 16124005.7654
01602
d1 more accurate than d2: True
dt: 20.0 d1(36000): 304950710.87446606 err: -20356210.884759486
dt: 21.0 d2(36000): 358390411.66992694 err: 33083489.910701394
d1 more accurate than d2: True
dt: 18.181818181818183 d1(36000): 306827465.76543397 err: -18479455.993
79158
dt: 19.09090909090909 d2(36000): 295221535.13281524 err: -30085386.6264
10306
d1 more accurate than d2: True
dt: 16.666666666666668 d1(36000): 308387497.8372482 err: -16919423.9219
7734
dt: 17.5 d2(36000): 312513536.1190847 err: -12793385.640140831
d1 more accurate than d2: False
dt: 15.384615384615385 d1(36000): 309704735.8727113 err: -15602185.8865
14246
dt: 16.153846153846153 d2(36000): 327331617.2724673 err: 2024695.513241
7679
d1 more accurate than d2: False
dt: 14.285714285714286 d1(36000): 310831764.25794446 err: -14475157.501
281083
dt: 15.0 d2(36000): 310099409.1257071 err: -15207512.633518457
d1 more accurate than d2: True
dt: 13.333333333333334 d1(36000): 311807006.3833444 err: -13499915.3758
81135
dt: 14.0 d2(36000): 323064379.1309787 err: -2242542.628246844
```

```
d1 more accurate than d2: False
dt: 12.5 d1(36000): 312659190.3459367 err: -12647731.413288832
dt: 13.125 d2(36000): 334538614.0418977 err: 9231692.282672167
d1 more accurate than d2: False
dt: 11.764705882352942 d1(36000): 313410225.1936504 err: -11896696.5655
75123
dt: 12.352941176470589 d2(36000): 344781623.98469967 err: 19474702.2254
7412
d1 more accurate than d2: True
dt: 11.1111111111111 d1(36000): 314077110.6904409 err: -11229811.06878
4654
dt: 11.666666666666666 d2(36000): 306924726.83303124 err: -18382194.926
19431
d1 more accurate than d2: True
dt: 10.526315789473685 d1(36000): 314673239.35358745 err: -10633682.405
638099
dt: 11.052631578947368 d2(36000): 317286024.1293592 err: -8020897.62986
6362
d1 more accurate than d2: False
dt: 10.0 d1(36000): 315209305.0033824 err: -10097616.755843163
dt: 10.5 d2(36000): 326690967.3190968 err: 1384045.5598712564
d1 more accurate than d2: False
dt: 9.523809523809524 d1(36000): 315693949.8967954 err: -9612971.862430
155
dt: 10.0 d2(36000): 315209305.0033824 err: -10097616.755843163
d1 more accurate than d2: True
dt: 9.090909090909092 d1(36000): 316134234.17687774 err: -9172687.58234
781
dt: 9.545454545454545 d2(36000): 323834427.5132367 err: -1472494.245988
8458
d1 more accurate than d2: False
dt: 8.695652173913043 d1(36000): 325314988.8144326 err: 8067.0552070736
885
dt: 9.130434782608695 d2(36000): 331771573.0208454 err: 6464651.2616198
66
d1 more accurate than d2: True
dt: 8.333333333333334 d1(36000): 316904040.94940704 err: -8402880.80981
8506
dt: 8.75 d2(36000): 339105608.602025 err: 13798686.842799425
d1 more accurate than d2: True
dt: 8.0 d1(36000): 317242477.956744 err: -8064443.802481532
dt: 8.4 d2(36000): 312073772.90333813 err: -13233148.855887413
d1 more accurate than d2: True
dt: 7.6923076923076925 d1(36000): 317554730.85038847 err: -7752190.9088
3708
dt: 8.076923076923077 d2(36000): 319467928.5346449 err: -5838993.224580
646
d1 more accurate than d2: False
dt: 7.407407407407407 d1(36000): 317843725.2435264 err: -7463196.515699
148
dt: 7.777777777777778 d2(36000): 326358194.9955624 err: 1051273.2363368
273
d1 more accurate than d2: False
dt: 7.142857142857143 d1(36000): 318111966.5463912 err: -7194955.212834
358
dt: 7.5 d2(36000): 317749815.6253986 err: -7557106.133826971
d1 more accurate than d2: True
```

```
dt: 6.896551724137931 d1(36000): 325314850.0292366 err: 7928.2700110673
9
dt: 7.241379310344827 d2(36000): 324212684.8513136 err: -1094236.907911
9563
d1 more accurate than d2: True
Where delta t divides 600, it was more accurate 15 times.
Where delta t doesn't divide 600, it was more accurate 14 times.
```

The above output is messy, but it compares the computed d(36000) with the very small time step from (a). If error was smaller, it converges faster. d1 had Δt divide 600 but d2 did not. However, it was fairly even in terms of which converged quicker (the former 15 times and the latter 14).

Despite that, it makes sense why we would want our time steps to step on discontinuities. Around the discontinuity, it uses the left side of the discontinuity to predict past it. Any portion of delta t going over the discontinuity is being modeled incorrectly by the left side. If it steps on the discontinuity, it minimizes the extrapolation based on incorrect values.