In[1]:= {NotebookFileName[], DateString[]}

Out[1]= {H:\morpheus\worm4\worm4.nb, Sat 24 Apr 2021 11:15:55}

Setup

```
In[2]:= Needs["Developer`"]

In[3]:= $HistoryLength = 5

Out[3]= 5

In[4]:= curDir = FileNameDrop[NotebookFileName[]]

Out[4]= H:\morpheus\worm4

In[5]:= modelName = FileNameTake[curDir, -1]

Out[5]= worm4

In[6]:= logFile = "logger.csv"

Out[6]= logger.csv

In[7]:= ksfdDir = "D:\\WSL\\KSFD"

Out[7]= D:\WSL\KSFD

In[8]:= ksfdDataDir = FileNameJoin[{ksfdDir, "data"}]

Out[8]= D:\WSL\KSFD\data

In[9]:= FileNames["*", ksfdDataDir]

Out[9]= {D:\WSL\KSFD\data\energies138.mx, D:\WSL\KSFD\data\energies139.mx,
  D:\WSL\KSFD\data\energies140.mx, D:\WSL\KSFD\data\energies141.mx,
  D:\WSL\KSFD\data\energies157.mx, D:\WSL\KSFD\data\images103,
  D:\WSL\KSFD\data\images133, D:\WSL\KSFD\data\images134, D:\WSL\KSFD\data\images135,
  D:\WSL\KSFD\data\images136g, D:\WSL\KSFD\data\images136m, D:\WSL\KSFD\data\images90,
  D:\WSL\KSFD\data\images91, D:\WSL\KSFD\data\images92, D:\WSL\KSFD\data\images93,
  D:\WSL\KSFD\data\images94, D:\WSL\KSFD\data\images95, D:\WSL\KSFD\data\last133plot.png,
  D:\WSL\KSFD\data\last92plot.png, D:\WSL\KSFD\data\o103endPlot.pdf,
  D:\WSL\KSFD\data\o103step0Plot.pdf, D:\WSL\KSFD\data\o134nx128dtndterrs.csv,
  D:\WSL\KSFD\data\o134nx512dtndterrs.csv, D:\WSL\KSFD\data\o134nxndt1errs.csv,
  D:\WSL\KSFD\data\o135ampsPlot.pdf, D:\WSL\KSFD\data\o135step0Plot.pdf,
  D:\WSL\KSFD\data\o136gnx512dtnerrs.csv, D:\WSL\KSFD\data\o136gnxndt4errs.csv,
  D:\WSL\KSFD\data\o136mnx512dtnerrs.csv, D:\WSL\KSFD\data\o136mnxndt4errs.csv,
  D:\WSL\KSFD\data\o93nx512dtndterrs.csv, D:\WSL\KSFD\data\o93nxndt1errs.csv,
  D:\WSL\KSFD\data\o94ampsPlot.pdf, D:\WSL\KSFD\data\o94step0Plot.pdf,
  D:\WSL\KSFD\data\o95dterrs.csv, D:\WSL\KSFD\data\o95nx512dtnerrs.csv,
  D:\WSL\KSFD\data\o95nxerrs.csv, D:\WSL\KSFD\data\o95nxndt2errs.csv,
  D:\WSL\KSFD\data\options138, D:\WSL\KSFD\data\options139, D:\WSL\KSFD\data\options140,
  D:\WSL\KSFD\data\options141, D:\WSL\KSFD\data\options143_6, D:\WSL\KSFD\data\options157}
```

Functions

In[10]:=
```
sweepDir1 = FileNameJoin[{curDir, modelName <> "a"}]
```

Out[10]=
```
H:\morpheus\worm4\worm4a
```

In[11]:=
```
sims1 = FileNames["sim*", sweepDir1];
Short[sims1]
```

Out[12]//Short=
```
{H:\morpheus\worm4\worm4a\sim_0.0_0.0, <<674>>, H:\mo … 8_8000}
```

In[13]:=
```
sim1 = Import[
  FileNameJoin[{sims1[[1]], logFile}],
  {"CSV", "Dataset"}
  , HeaderLines → 1
 ]
```

Out[13]=

time	0
cell.id	1
8 total ›	
time	0
cell.id	2
8 total ›	
time	0
cell.id	3
8 total ›	
time	0
cell.id	4
8 total ›	
time	0
cell.id	5
8 total ›	
time	0
cell.id	6
8 total ›	
time	0
cell.id	7
8 total ›	
time	0
cell.id	8
8 total ›	
time	0
cell.id	9
8 total ›	
time	0
cell.id	10
8 total ›	

rows 1–10 of 202

In[14]:=
```
sim1[[1]]["MKtemp"]
```

Out[14]= 0

In[15]:=
```
sim1[[1]] // Keys
```

Out[15]=

time
cell.id
cell.center.x
cell.center.y
delta_r.x
delta_r.y
MKtemp
cmstrength

In[16]:=
```
sim1[GroupBy["time"]]
```

Out[16]=

0	time	0
	cell.id	1
	8 total ›	
	time	0
	cell.id	2
	8 total ›	
	time	0
	cell.id	3
	8 total ›	
	100 total ›	
—		
10 000	time	10 000
	cell.id	1
	8 total ›	
	time	10 000
	cell.id	2
	8 total ›	
	time	10 000
	cell.id	3
	8 total ›	
	100 total ›	

In[17]:=
```
sim1[GroupBy["time"]][[-1]][StandardDeviation]
```

Out[17]=

time	0.0
cell.id	29.0115
cell.center.x	12.6052
cell.center.y	12.6111
delta_r.x	9.9862
delta_r.y	9.9745
MKtemp	0.0
cmstrength	0.0

In[18]:=
```
MissingQ[ds1[[1, "MKtime"]]]
```

Part: Part specification ds1[[1, MKtime]] is longer than depth of object.

Out[18]=
```
False
```

In[19]:=
```
estimateVelocityDiffusion[
  sweepDir_String,
  logName_ : logFile
 ] := Module[{sweepds, tfinal, MKtime, temp, cms, μx, μy, σx, σy},
  sweepds = Import[
    FileNameJoin[{sweepDir, logName}],
    {"CSV", "Dataset"}
    , HeaderLines → 1
   ];
  sweepds = sweepds[GroupBy["time"]][[-1]];
  temp = sweepds[[1]]["MKtemp"];
  cms = sweepds[[1]]["cmstrength"];
  tfinal = sweepds[[1]]["time"];
  MKtime = sweepds[[1]]["MKtime"];
  If[FailureQ[MKtime] || MissingQ[MKtime], MKtime = tfinal / 10000];
  {σx, σy} = sweepds[All, {"delta_r.x", "delta_r.y"}][StandardDeviation] /@
    {"delta_r.x", "delta_r.y"};
  {μx, μy} = sweepds[All, {"delta_r.x", "delta_r.y"}][Mean] /@ {"delta_r.x", "delta_r.y"};
  <|
   "cmstrength" → cms,
   "temperature" → temp,
   "MKtime" → MKtime,
   "cmtratio" → If[PossibleZeroQ[temp], ∞, cms / temp],
   "tfinal" → tfinal,
   "vmean" → μx / tfinal,
   "Dx" → σx^2 / (2 tfinal),
   "Dy" → σy^2 / (2 tfinal),
   "Dxy" -> (σx^2 + σy^2) / (4 tfinal)
   |>
 ]
```

In[20]:=
```
estimateVelocityDiffusion[sims1[[1]]]
```

Out[20]=
```
<|cmstrength → 0, temperature → 0, MKtime → 1, cmtratio → ∞, tfinal → 10000,
 vmean → 0.00004655, Dx → 0.00498621, Dy → 0.00497453, Dxy → 0.00498037|>
```

Plots

In[21]:=
```
ds1 = Dataset[estimateVelocityDiffusion /@ sims1]
```

Out[21]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy
0	0	1	∞	10 000	0.00004655	0.00498621	0.004
0	0.2	1	0.0	10 000	−0.0000241385	0.00684214	0.008
0	0.4	1	0.0	10 000	−0.000244853	0.0363752	0.038
0	0.6	1	0.0	10 000	0.0000673799	0.0681142	0.060
0	0.8	1	0.0	10 000	0.000129053	0.0972709	0.097
0	1	1	0	10 000	−0.000549561	0.116088	0.109
0	10	1	0	10 000	0.000207029	0.169672	0.239
0	100	1	0	10 000	−0.0000432087	0.16959	0.215
0	1000	1	0	10 000	−0.000705839	0.267672	0.253
0	10 000	1	0	10 000	0.000150761	0.284062	0.280
0	2	1	0	10 000	−0.000341603	0.154172	0.109
0	20	1	0	10 000	0.000160617	0.183204	0.203
0	200	1	0	10 000	0.0010539	0.261324	0.216
0	2000	1	0	10 000	−0.000558769	0.217894	0.226
0	4	1	0	10 000	−0.000302816	0.176871	0.188
0	40	1	0	10 000	0.000346383	0.179997	0.216
0	400	1	0	10 000	0.000392544	0.202208	0.229
0	4000	1	0	10 000	−0.00102218	0.243589	0.292
0	6	1	0	10 000	0.000788141	0.159843	0.153
0	60	1	0	10 000	0.000254052	0.205308	0.182

rows 1–20 of 676

In[22]:=
```
ds1[GroupBy["temperature"]][[-1]]
```

Out[22]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy
0	8000	1	0	10000	−0.000518815	0.258732	0.284503
0.2	8000	1	0.000025	10000	0.000775013	0.251578	0.276628
0.4	8000	1	0.00005	10000	0.00036839	0.282974	0.269314
0.6	8000	1	0.000075	10000	−0.000231954	0.243678	0.308253
0.8	8000	1	0.0001	10000	0.000547282	0.382868	0.310287
10000	8000	1	5/4	10000	0.192566	0.275149	0.275523
1000	8000	1	1/8	10000	0.0401048	0.332414	0.319793
100	8000	1	1/80	10000	0.00423985	0.290144	0.28207
1	8000	1	0.000125	10000	−0.000128128	0.31865	0.358721
10	8000	1	0.00125	10000	0.000188936	0.366922	0.33639
2000	8000	1	1/4	10000	0.0721517	0.324797	0.332349
200	8000	1	1/40	10000	0.00885501	0.258511	0.223076
20	8000	1	0.0025	10000	−0.000773436	0.313912	0.290117
2	8000	1	0.00025	10000	−0.00035091	0.307887	0.301081
4000	8000	1	1/2	10000	0.121491	0.417061	0.295466
400	8000	1	1/20	10000	0.0174633	0.314154	0.401106
40	8000	1	0.005	10000	0.00140717	0.289698	0.336082
4	8000	1	0.0005	10000	0.00121919	0.291916	0.265094
6000	8000	1	3/4	10000	0.154264	0.553458	0.273441
600	8000	1	3/40	10000	0.0259479	0.258868	0.266818

rows 1–20 of 26

In[23]:=
```
{tfinal1, MKtime1} = ds1[[1]] /@ {"tfinal", "MKtime"}
```

Out[23]=
```
{10000, 1}
```

In[24]:=
```
DeleteDuplicates@Flatten@Normal@Keys[ds1]
```

Out[24]=
```
{cmstrength, temperature, MKtime, cmtratio, tfinal, vmean, Dx, Dy, Dxy}
```

In[25]:=
```
Normal[ds1[GroupBy["cmstrength"]][[1, All, "temperature"]]] // Sort // InputForm
```

Out[25]//InputForm=
```
{0, 0.2, 0.4, 0.6, 0.8, 1, 2, 4, 6, 8, 10, 20, 40, 60, 80, 100,
 200, 400, 600, 800, 1000, 2000, 4000, 6000, 8000, 10000}
```

In[26]:=
```
ds1[GroupBy["cmstrength"]][[1]][SortBy["temperature"]]
```

Out[26]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy
0	0	1	∞	10 000	0.00004655	0.00498621	0.00497
0	0.2	1	0.0	10 000	-0.0000241385	0.00684214	0.00817
0	0.4	1	0.0	10 000	-0.000244853	0.0363752	0.03806
0	0.6	1	0.0	10 000	0.0000673799	0.0681142	0.06008
0	0.8	1	0.0	10 000	0.000129053	0.0972709	0.09794
0	1	1	0	10 000	-0.000549561	0.116088	0.10963
0	2	1	0	10 000	-0.000341603	0.154172	0.10947
0	4	1	0	10 000	-0.000302816	0.176871	0.18867
0	6	1	0	10 000	0.000788141	0.159843	0.15385
0	8	1	0	10 000	0.000209514	0.160657	0.18224
0	10	1	0	10 000	0.000207029	0.169672	0.23937
0	20	1	0	10 000	0.000160617	0.183204	0.20395
0	40	1	0	10 000	0.000346383	0.179997	0.21617
0	60	1	0	10 000	0.000254052	0.205308	0.18284
0	80	1	0	10 000	0.000747709	0.161558	0.20391
0	100	1	0	10 000	-0.0000432087	0.16959	0.21522
0	200	1	0	10 000	0.0010539	0.261324	0.21641
0	400	1	0	10 000	0.000392544	0.202208	0.22995
0	600	1	0	10 000	0.000205537	0.164203	0.17140
0	800	1	0	10 000	-0.00064113	0.212823	0.24169

rows 1–20 of 26

In[27]:=
```
ds1[SortBy["cmstrength"]][GroupBy["cmstrength"]]
```

Out[27]=

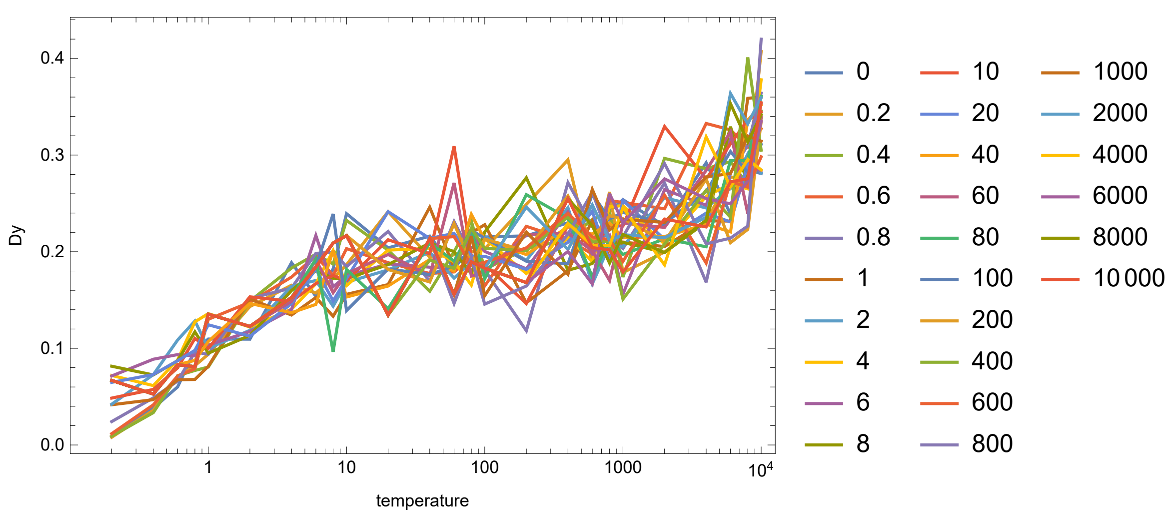
	cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx
0	0	0	1	∞	10 000	0.00004655	0.00498
	0	0.2	1	0.0	10 000	−0.000024138	0.00684
	26 total ›						
0.2	0.2	0	1	∞	10 000	0.0001502	0.00490
	0.2	0.2	1	1.0	10 000	0.00367486	0.00833
	26 total ›						
0.4	0.4	0	1	∞	10 000	0.0001502	0.00490
	0.4	0.2	1	2.0	10 000	0.00917963	0.02085
	26 total ›						
0.6	0.6	0	1	∞	10 000	0.0001502	0.00490
	0.6	0.2	1	3.0	10 000	0.0252299	0.06489
	26 total ›						
0.8	0.8	0	1	∞	10 000	0.0001502	0.00490
	0.8	0.2	1	4.0	10 000	0.0704344	0.16023
	26 total ›						
1	1	0	1	∞	10 000	0.172329	0.22672
	1	0.2	1	5.0	10 000	0.170042	0.18383
	26 total ›						
2	2	0	1	∞	10 000	0.200677	0.13301
	2	0.2	1	10.0	10 000	0.200415	0.09522
	26 total ›						
4	4	0	1	∞	10 000	0.23437	0.11320
	4	0.2	1	20.0	10 000	0.233877	0.13695
	26 total ›						
6	6	0	1	∞	10 000	0.241841	0.1432
	6	0.2	1	30.0	10 000	0.24116	0.14795
	26 total ›						
8	8	0	1	∞	10 000	0.24486	0.12160
	8	0.2	1	40.0	10 000	0.244492	0.16571
	26 total ›						

rows 1–10 of 26

In[28]:=
```
Clear[dsPlots];
dsPlots[
  xname_String,
  yname_String : "Dxy",
  byname_String : "cmstrength",
  dsin_Dataset : ds1
 ] := Block[{is = 400, dss, gds, temp, labels, data},
  gds = dsin[SortBy[byname]][GroupBy[byname]];
  gds = Normal[gds];
  labels = {};
  data = Table[
    temp = ds[[1]][byname];
    AppendTo[labels, temp];
    Transpose@{
      Normal[ds[[All, xname]]],
      Normal[ds[[All, yname]]]
     },
    {ds, gds}
   ];
  ListLogLinearPlot[data
   , PlotRange → {Full, Full}
   , PlotLegends → labels
   , Joined → True
   , Frame → {{True, True}, {True, True}}
   , FrameLabel → {{yname, None}, {xname, None}}
   , ImageSize → is
  ]
 ]
```

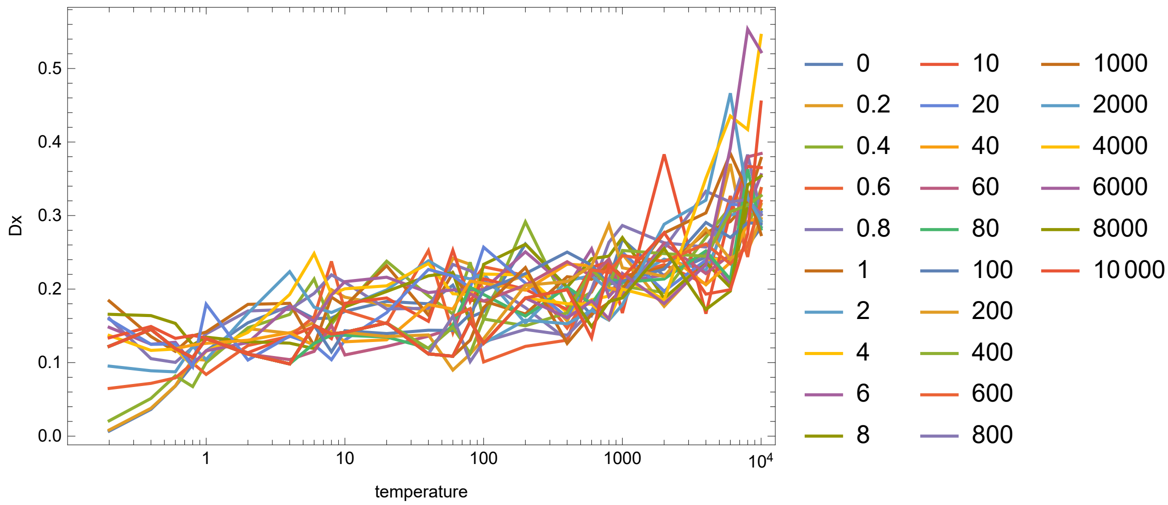
In[30]:=
```
dsPlots["temperature", "Dy"]
```

Out[30]=



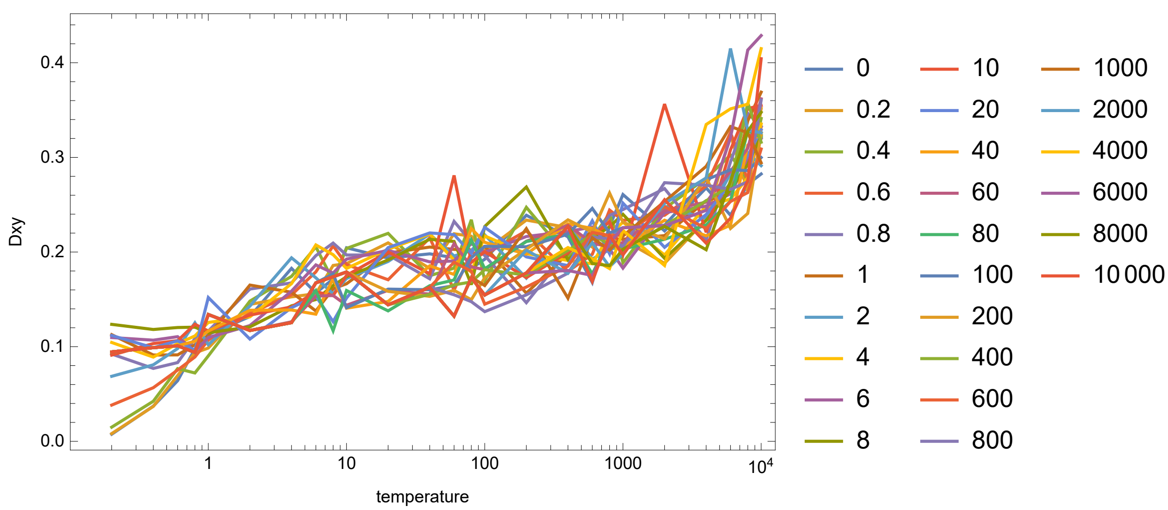
In[31]:= `dsPlots["temperature", "Dx"]`

Out[31]=



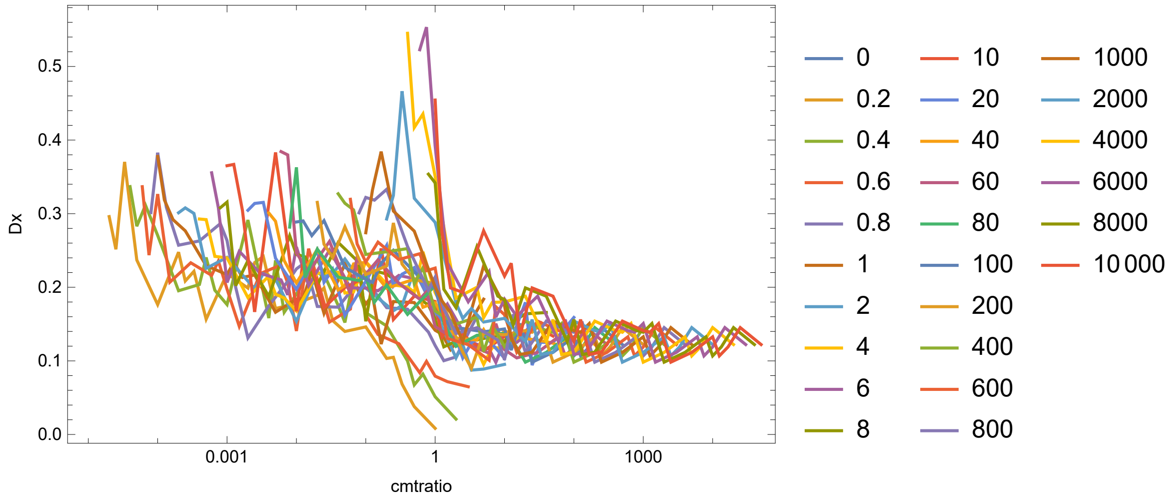
In[32]:= `dsPlots["temperature", "Dxy"]`

Out[32]=



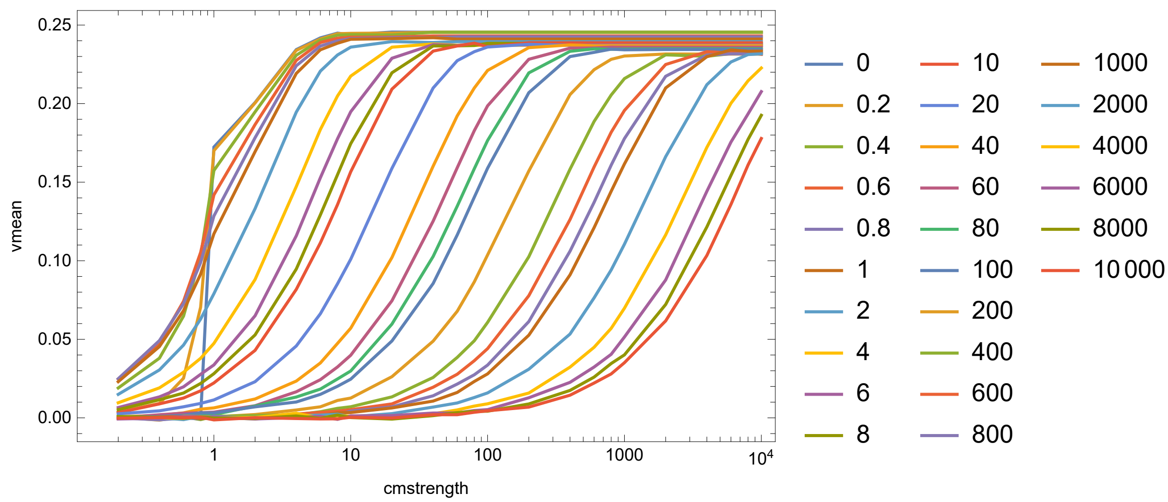
In[33]:=
```
vplot2 = dsPlots["cmtratio", "Dx", "cmstrength"];
Show[vplot2
 , Frame -> True
]
```

Out[34]=



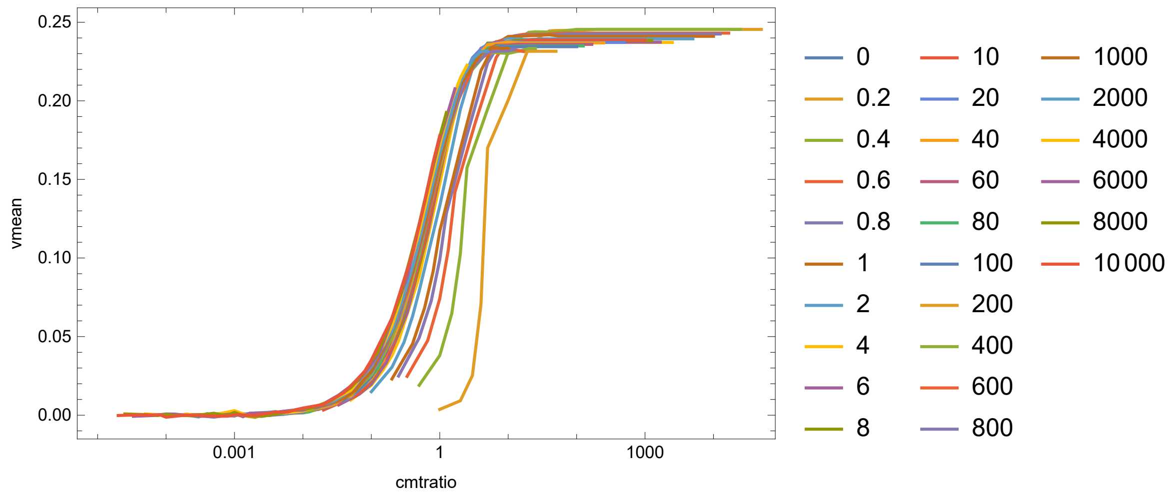
In[35]:=
```
Show[
 dsPlots["cmstrength", "vmean", "temperature"]
 , Frame -> True
]
```

Out[35]=



In[36]:=
```
vplot2 = dsPlots["cmtratio", "vmean", "temperature"];
Show[vplot2
 , Frame -> True
]
```

Out[37]=



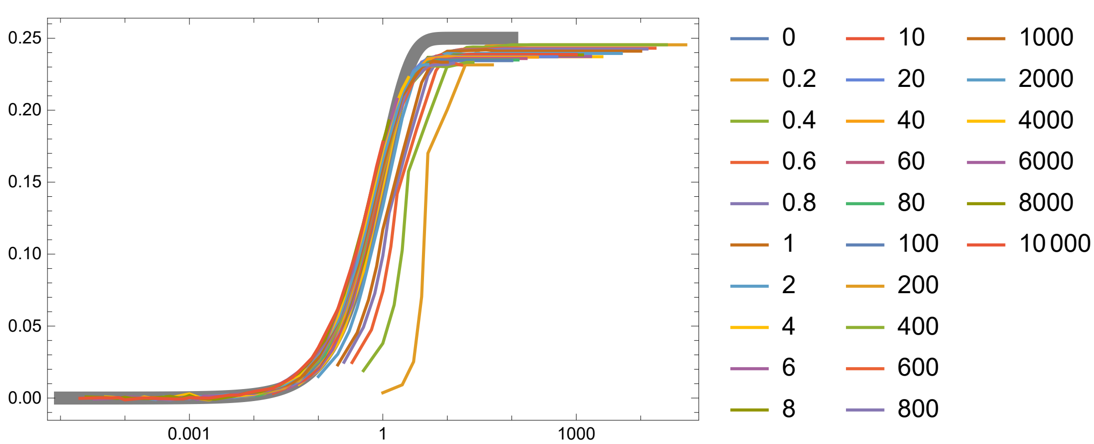
In[38]:=
```
ds1[Select[#["temperature"] == 10 000 && #["cmstrength"] == 10 000 &]]
```

Out[38]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy	Dxy
10 000	10 000	1	1	10 000	0.177795	0.45439	0.354104	0.40424

In[39]:=
```
Show[
 LogLinearPlot[(1 - e^-x) / 4, {x, 10^-5, 100}
  , PlotStyle → {Thickness[0.02], Gray}
 ],
 vplot2
 , Frame -> True
 , PlotRange -> All
]
```

Out[39]=



OK, let's try to understand this. Metropolis works like this. First, we have a set of possible steps we might take. What those steps are for the CPM is not clearly documented in the Morpheus doco anywhere that I can find. This is really the critical question. We then choose one from that set and calculate the ΔE for that step. If $\Delta E < 0$, we take the step. If $\Delta E > 0$, we calculate $P = e^{-\Delta E/T}$, then we take the step with probability P, meaning that we are unlikely to take steps that cause a big positive change in energy, big being assessed relative to the temperature. For chemotaxis, $E = \mu F$, where μ is a parameter, chemotaxis strength, and F is some spatially varying field.

Now, in these calibration runs I set up a field whose gradient is exactly $\nabla F = (F_x, F_y) = (1, 0)$ in grid units. Then I varied μ and T. The first thing you can see from the above description is that the result should depend only on the ratio $\frac{\mu}{T}$. Thus in some of the plots above I plotted diffusion or velocity vs cmtratio, which is $\frac{\mu}{T}$. And if fact it more or less works. The curves more or less superimpose. This is more obvious for velocity. plots of v vs $\frac{\mu}{T}$ for different temperatures almost superimpose, except at $T \leq 1$. The plots of D_x vs $\frac{\mu}{T}$ are a lot noisier, but the curves for different μ match pretty well. (Actually, something simpler is true for D—D and D_y depend only on T to a good first approximation. D_x depends on on T except at $T \leq 1$, where μ makes a difference.) Something weird is going on at low T—I'm guessing that integer arithmetic is used in some part of the computation, so that there is truncation.

Now, there are two limits where the results are sort of predictable. First, in the limit of very high T, every step should be taken. So the velocity should be low. And the diffusion should be maximal, and should correspond to $\frac{1}{4}\left\langle \left\| \Delta\vec{x} \right\|^2 \right\rangle$, where $\Delta\vec{x}$ is the change in cell center position caused by a single step. This maxes out at about 0.3, although it is easy to imagine that it has not saturated even at $T = 10^4$ and would go higher if I went to higher temps. I also noted when I ran the sims that it took a lot longer to run a sim at $T = 10^4$ than at, say, $T = 10$. This suggests that T also affects the sampling strategy, causing a larger sample to be tried.

The other simple limit in principle is $\frac{\mu}{T} \to \infty$. This should cause every step that produces a move in the positive x direction to be accepted and every step in the negative x direction rejected. So the mean velocity should just be the mean size of a positive x step. This is consistently $v \approx 0.25$.

Suppose that $\Delta x = \{-1, 0, 1\}$ with probabilities $\left\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right\}$. And let's suppose we always accept the positive step. We accept the negative step with probability $e^{-\mu/T}$. Then the predicted mean velocity is

$$\begin{aligned}\overline{v} &= \frac{1}{4}\cdot 1 + \frac{1}{2}\cdot 0 + \frac{1}{4}\cdot e^{-u/T}\cdot(-1)\\ &= \frac{1}{4}\left(1 - e^{-u/T}\right)\end{aligned}$$

In fact, that fits the v vs $\frac{\mu}{T}$ plot fairly well. The variance, though, ought to be

In[40]:=
```
mdist = EmpiricalDistribution[{1/4, 1/2 + (1/4) (1 - ⅇ^-μ), (1/4) ⅇ^-μ} -> {1, 0, -1}
 ]
```

Out[40]= DataDistribution[Type: Empirical Data points: 3]

In[41]:=
```
Mean[mdist] // Simplify
```

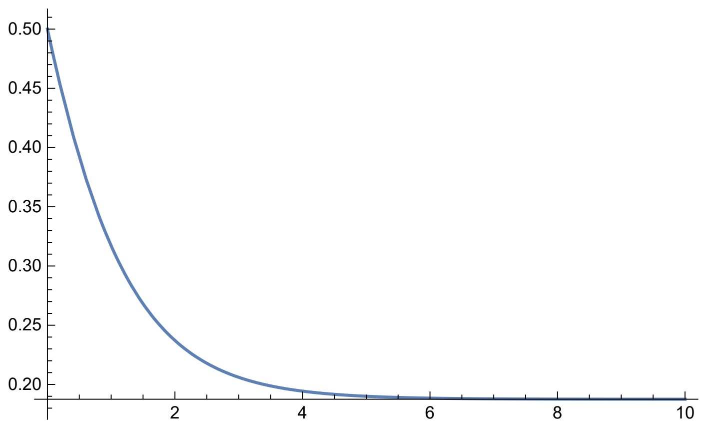
Out[41]= $\frac{1}{4} - \frac{e^{-\mu}}{4}$

In[42]:=
```
Variance[mdist] // FullSimplify
```

Out[42]= $\frac{1}{8} e^{-\mu} (3 + \text{Cosh}[\mu] + 2\,\text{Sinh}[\mu])$

In[43]:=
```
Plot[Variance[mdist], {μ, 0, 10}
 , PlotRange → Full
]
```

Out[43]=



So, the results mostly make sense. At every T, the maximum velocity, that attained when $\frac{\mu}{T}$ is large, approaches $\frac{1}{4}$. Furthermore, the approach to this v_{max} is exponential—$v = \frac{1}{4}(1 - e^{-\mu/T})$ is a fairly good approximation at all but the lowest T.

This dependence is apparently well explained by a sample distribution that that includes steps of $\Delta x \in \{-1, 0, 1\}$ with probabilities $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$. If you choose the positive option at every step where it is available, or 0 otherwise, you get a mean velocity of $\frac{1}{4}$.

The puzzle, though, is the variance. This distribution has a variance of $\frac{1}{2}$. That should lead to a maximum diffusion constant of $\frac{1}{4}$. But the numbers are larger than that for high T.

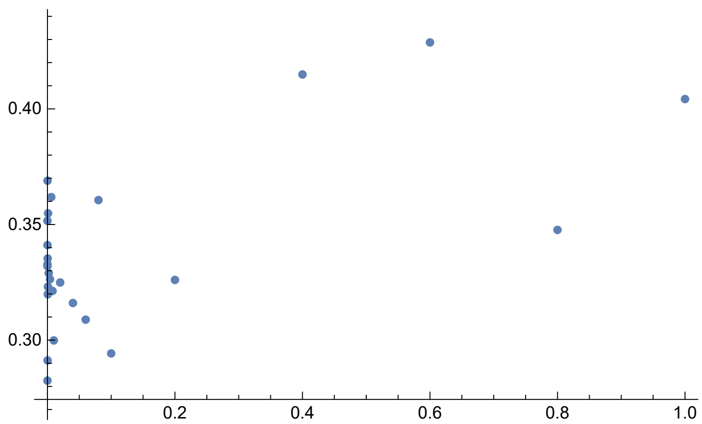
In[44]:=
```
ds1[Select[#["temperature"] == 10000 &]][[All, "Dxy"]]
```

Out[44]=

0.282494	0.35158	0.341085	0.331976	0.332783	0.404247
0.294257	0.299865	0.368932	0.354827	0.326023	0.324923
0.32896	0.291229	0.414863	0.316053	0.326324	0.33532
0.42873	0.308854	0.361861	0.31982	0.347661	0.360548
0.321269	0.323127				

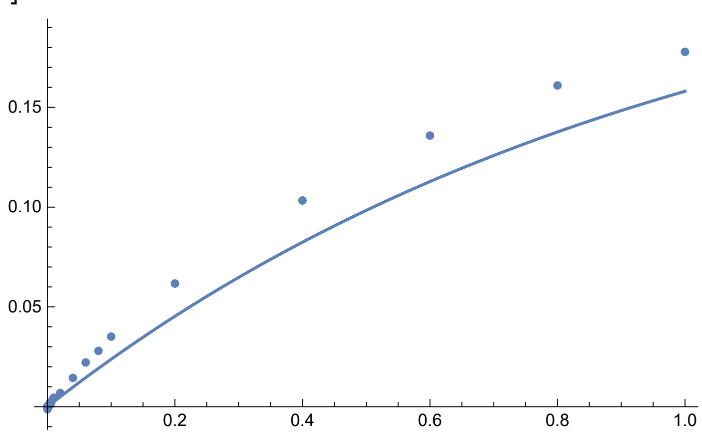
In[45]:=
```
ListPlot[
 ds1[Select[#["temperature"] == 10000 &]][[All, {"cmtratio", "Dxy"}]]
 , PlotRange → All
]
```

Out[45]=

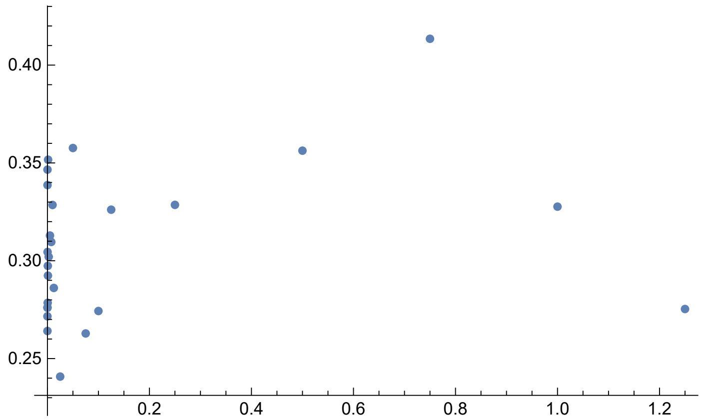


In[46]:=
```
Show[
 ListPlot[
  ds1[Select[#["temperature"] == 10000 &]][[All, {"cmtratio", "vmean"}]]
  , PlotRange → All
 ],
 Plot[(1 - e^-x) / 4, {x, 0, 1}
 ]
]
```

Out[46]=



In[47]:=
```
ListPlot[
 ds1[Select[#["temperature"] == 8000 &]][[All, {"cmtratio", "Dxy"}]]
 , PlotRange → All
]
```

Out[47]=


Together, these numbers suggest that the sample changes as T increases, with the mean of the positive possibilities growing a bit larger than $\frac{1}{4}$ and the variance growing, too.

I probably ought to do this calculation with a diagonal gradient to see if Δx and Δy are correlated, which could cause problems. But if that is true, I'm not sure what I could do with the information. I don't have a way of making the chemotactic strength nonisotropic. So I guess I would just have to live with it.

Simulation parameter values

Let's suppose I set up a toy model with 360 worms on an 0.2×0.2 cm domain with a 75×75 grid.

In[48]:=
```
width2 = height2 = Quantity[0.2, "Centimeters"];
nx2 = ny2 = 75;
dt2 = Quantity[0.15, "Seconds"];
{dx2, dy2} = {width2, height2} / {nx2, ny2}
```

Out[51]= {0.00266667 cm , 0.00266667 cm}

In[52]:=
```
{wormwidth2, wormlength2} = Quantity[{15, 240}, "Micrometers"]
```

Out[52]= {15 μm , 240 μm}

In[53]:=
```
wormsize2 = (wormwidth2 wormlength2) / (dx2 dy2)
```

Out[53]= 5.0625

In[54]:=
```
gridunitD2 = (dx2 dy2) / Quantity[dt2, "Seconds"]
```

Out[54]= $0.0000474074\ \mathrm{cm}^2/\mathrm{s}^2$

That's what a diffusion constant of 1 in grid units would correspond to in $\mathrm{cm}^2\ s^{-1}$.

In[55]:=
```
d139final = Import[
    FileNames["*", FileNameJoin[{ksfdDataDir, "options139"}]][[-1]],
    "Data"
   ];
i139final = d139final["/images"];
p139final = <|ImportString[d139final["/params"], "JSON"]|>
```

Out[57]=
```
<|D_1_1 → 1/1000000, nligands_2 → 1, alpha_2 → 1500, alpha_1 → 1500, Umin → 1/10000000,
 randgridnh → 0, nwidth → 384, variance_interval → 100., variance_timing_function → 2002.03,
 s_1_1 → 0.01, tmax → 200000., ngroups → 1, atol → 0.01, randgridnw → 0,
 CFL_safety_factor → 0.5, dt → 1/100000000, rhomin → 1/10000000, lastvart → 0.,
 s_2_1 → 0.001, ky0 → 2., aUr → 0.120891, kx0 → 3.4641, nligands_1 → 1, aUa → 0.863036,
 maxsteps → 10000, rhomax → 28000, rtol → 1/1000000, maxscale → 2., ndepth → 384,
 dim → 2, murho0 → 9000., rho0 → 9000., srho0 → 90., D_2_1 → 1/100000, nheight → 384,
 sigma → 0.02357, series_1_1 → 1, cushion → 2000, variance_rate → 0., t0 → 0.,
 k0 → 4., lamda → 0.000955351, gamma_2_1 → 0.001, slowdown → 0.02, s2 → 5.55545 × 10^-6,
 beta_2 → -0.0000111109, beta_1 → 0.0000111109, depth_1_1 → 0.4, gamma_1_1 → 0.01,
 conserve_worms → False, randgridnd → 0, U0_1_1 → , width → 1, nelements → 384,
 height → 1, degree → 3, depth → 1., weight_1_1 → 1., Nworms → 0, t → 200203.|>
```

The following is our desired worm diffusion rate in real units.

In[58]:=
```
σ139 = Quantity[p139final["s2"], "Centimeters"^2 / "Seconds"]
```

Out[58]= $5.55545 \times 10^{-6}\ \mathrm{cm}^2/\mathrm{s}$

In grid units, we need

In[59]:=
```
σ139 / gridunitD2
```

Out[59]= 0.117185 s

In[60]:=
```
p139final["D_1_1"] / (dx2 dy2)
```

Out[60]= $0.140625\ /\mathrm{cm}^2$

worm4b

In[61]:=
```
sweepDir2 = FileNameJoin[{curDir, "worm4b2"}]
```

Out[61]= H:\morpheus\worm4\worm4b2

In[62]:=
```
sims2 = FileNames["sim*", sweepDir2];
Short[sims2]
```

Out[63]//Short=
```
{H:\morpheus\worm4\worm4b2\sim_0.0_0.0, <<674>>, H:\mo … _8000}
```

In[64]:=
```
estimateVelocityDiffusion[sims2[[-1]]]
```

Out[64]=
```
<|cmstrength → 8, temperature → 8000, MKtime → 0.15, cmtratio → 1/1000,
 tfinal → 1500, vmean → 0.0031493, Dx → 1.48052, Dy → 2.06624, Dxy → 1.77338|>
```

In[65]:=
```
ds2 = Dataset[estimateVelocityDiffusion /@ sims2]
```

Out[65]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx
0	0	0.15	∞	1500	0.0000586667	0.00472924
0	0.2	0.15	0.0	1500	0.000133778	0.0202838
0	0.4	0.15	0.0	1500	−0.000402444	0.13506
0	0.6	0.15	0.0	1500	−0.000951619	0.299735
0	0.8	0.15	0.0	1500	−0.00166478	0.406173
0	1	0.15	0	1500	0.00325044	0.627768
0	10	0.15	0	1500	0.000444487	0.950335
0	100	0.15	0	1500	0.00292561	1.19553
0	1000	0.15	0	1500	0.00422094	1.24679
0	10000	0.15	0	1500	0.00604645	2.40613
0	2	0.15	0	1500	0.00242859	0.641311
0	20	0.15	0	1500	−0.00609222	0.987535
0	200	0.15	0	1500	−0.000310101	0.938662
0	2000	0.15	0	1500	0.00832676	1.61467
0	4	0.15	0	1500	−0.00153308	0.567142
0	40	0.15	0	1500	−0.0000319382	1.19626
0	400	0.15	0	1500	0.00226206	1.17687
0	4000	0.15	0	1500	0.00419683	1.61481
0	6	0.15	0	1500	−0.00116685	1.00051
0	60	0.15	0	1500	0.00672245	0.781455

rows 1–20 of 676

In[66]:=
```
{tfinal2, MKtime2} = ds2[[1]] /@ {"tfinal", "MKtime"}
```

Out[66]=
```
{1500, 0.15}
```

In[67]:=
```
ds2[GroupBy["temperature"]][[-1]]
```

Out[67]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy
0	8000	0.15	0	1500	0.00786922	2.09757	2.22
0.2	8000	0.15	0.000025	1500	0.00476517	1.89948	1.78
0.4	8000	0.15	0.00005	1500	-0.00104947	1.33388	1.86
0.6	8000	0.15	0.000075	1500	0.00559173	1.8551	1.85
0.8	8000	0.15	0.0001	1500	-0.00063802	1.75056	1.83
10 000	8000	0.15	5/4	1500	1.28184	1.78372	1.83
1000	8000	0.15	1/8	1500	0.271511	2.33548	1.39
100	8000	0.15	1/80	1500	0.0304989	1.62275	2.10
1	8000	0.15	0.000125	1500	0.00640358	2.00112	1.98
10	8000	0.15	0.00125	1500	0.0061581	1.3857	1.74
2000	8000	0.15	1/4	1500	0.486441	2.77179	1.32
200	8000	0.15	1/40	1500	0.0692196	2.1099	2.26
20	8000	0.15	0.0025	1500	0.00729434	1.89216	1.55
2	8000	0.15	0.00025	1500	-0.00222092	1.82031	1.90
4000	8000	0.15	1/2	1500	0.782442	3.53221	2.34
400	8000	0.15	1/20	1500	0.110522	1.90741	1.68
40	8000	0.15	0.005	1500	0.0094493	1.6366	1.90
4	8000	0.15	0.0005	1500	-0.00232761	2.19666	1.90
6000	8000	0.15	3/4	1500	1.00969	3.15898	2.04
600	8000	0.15	3/40	1500	0.163292	2.66306	1.57

rows 1–20 of 26

In[68]:=
```
DeleteDuplicates@Flatten@Normal@Keys[ds2]
```

Out[68]=
```
{cmstrength, temperature, MKtime, cmtratio, tfinal, vmean, Dx, Dy, Dxy}
```

In[69]:=
```
Normal[ds2[GroupBy["cmstrength"]][[1, All, "temperature"]]] // Sort // InputForm
```

Out[69]//InputForm=
```
{0, 0.2, 0.4, 0.6, 0.8, 1, 2, 4, 6, 8, 10, 20, 40, 60, 80, 100,
 200, 400, 600, 800, 1000, 2000, 4000, 6000, 8000, 10000}
```

In[70]:=
```
ds2[GroupBy["cmstrength"]][[1]][SortBy["temperature"]]
```

Out[70]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy
0	0	0.15	∞	1500	0.0000586667	0.00472924	0.
0	0.2	0.15	0.0	1500	0.000133778	0.0202838	0.
0	0.4	0.15	0.0	1500	−0.000402444	0.13506	0.
0	0.6	0.15	0.0	1500	−0.000951619	0.299735	0.
0	0.8	0.15	0.0	1500	−0.00166478	0.406173	0.
0	1	0.15	0	1500	0.00325044	0.627768	0.
0	2	0.15	0	1500	0.00242859	0.641311	0.
0	4	0.15	0	1500	−0.00153308	0.567142	0.
0	6	0.15	0	1500	−0.00116685	1.00051	0.
0	8	0.15	0	1500	0.00355325	0.797968	0.
0	10	0.15	0	1500	0.000444487	0.950335	0.
0	20	0.15	0	1500	−0.00609222	0.987535	0.
0	40	0.15	0	1500	−0.0000319382	1.19626	0.
0	60	0.15	0	1500	0.00672245	0.781455	1.
0	80	0.15	0	1500	0.0027837	0.838457	0.
0	100	0.15	0	1500	0.00292561	1.19553	1.
0	200	0.15	0	1500	−0.000310101	0.938662	1.
0	400	0.15	0	1500	0.00226206	1.17687	1.
0	600	0.15	0	1500	0.00502242	1.53833	1.
0	800	0.15	0	1500	−0.00647744	1.33995	1.

rows 1–20 of 26

In[71]:=
```
ds2[SortBy["cmstrength"]][GroupBy["cmstrength"]]
```

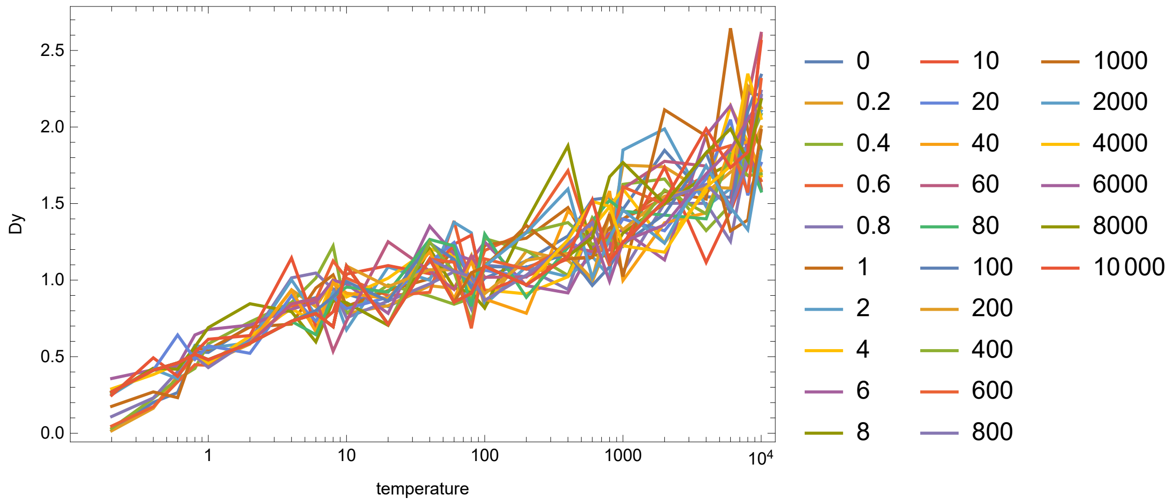
Out[71]=

	cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	D
0	0	0	0.15	∞	1500	0.0000586667	0.
	0	0.2	0.15	0.0	1500	0.000133778	0.
	26 total ›						
0.2	0.2	0	0.15	∞	1500	0.001328	0.
	0.2	0.2	0.15	1.0	1500	0.021898	0.
	26 total ›						
0.4	0.4	0	0.15	∞	1500	0.001328	0.
	0.4	0.2	0.15	2.0	1500	0.0597404	0.
	26 total ›						
0.6	0.6	0	0.15	∞	1500	0.001328	0.
	0.6	0.2	0.15	3.0	1500	0.174522	0.
	26 total ›						
0.8	0.8	0	0.15	∞	1500	0.001328	0.
	0.8	0.2	0.15	4.0	1500	0.465822	0.
	26 total ›						
1	1	0	0.15	∞	1500	1.10091	0.
	1	0.2	0.15	5.0	1500	1.09067	0.
	26 total ›						
2	2	0	0.15	∞	1500	1.28761	0.
	2	0.2	0.15	10.0	1500	1.28052	0.
	26 total ›						
4	4	0	0.15	∞	1500	1.47197	0.
	4	0.2	0.15	20.0	1500	1.47432	0.
	26 total ›						
6	6	0	0.15	∞	1500	1.51869	0.
	6	0.2	0.15	30.0	1500	1.51245	0.
	26 total ›						
8	8	0	0.15	∞	1500	1.52225	0.
	8	0.2	0.15	40.0	1500	1.52269	0.
	26 total ›						

rows 1–10 of 26

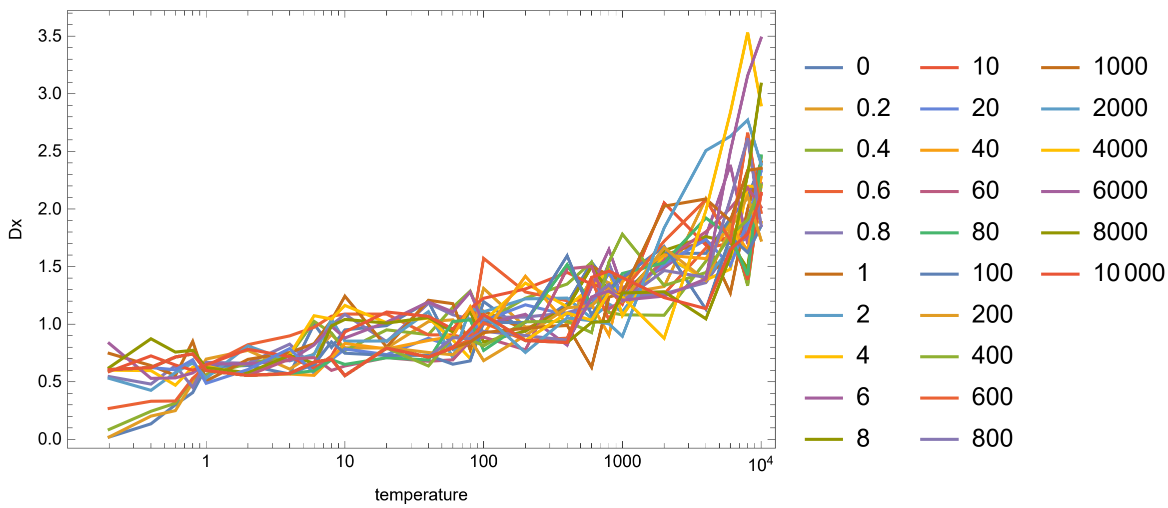
In[72]:=
```
dsPlots["temperature", "Dy", "cmstrength", ds2]
```

Out[72]=



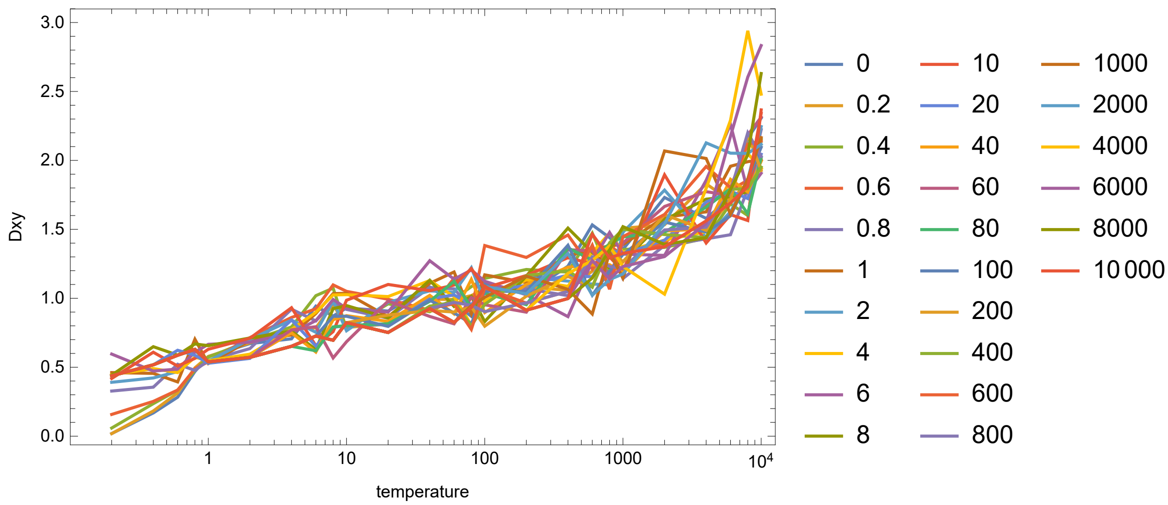
In[73]:=
```
dsPlots["temperature", "Dx", "cmstrength", ds2]
```

Out[73]=



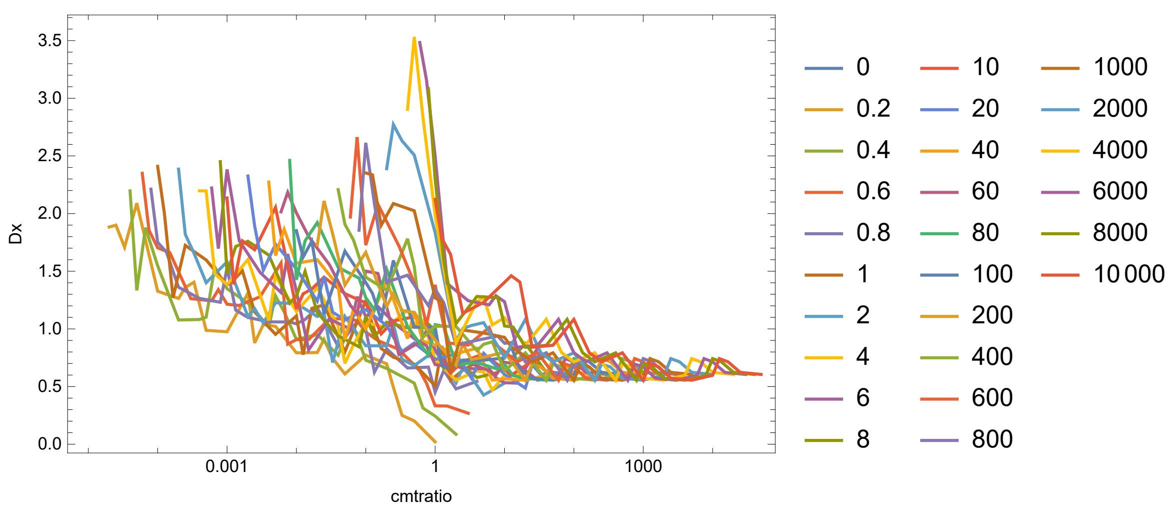
In[74]:=
```
dsPlots["temperature", "Dxy", "cmstrength", ds2]
```

Out[74]=



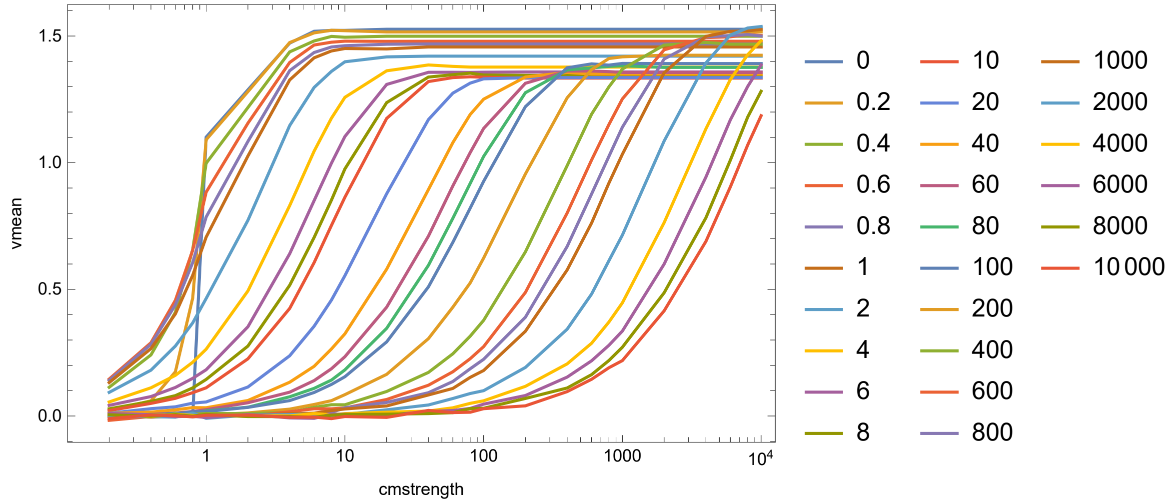
In[75]:=
```
vplot2 = dsPlots["cmtratio", "Dx", "cmstrength", ds2];
Show[vplot2
 , Frame -> True
]
```

Out[76]=



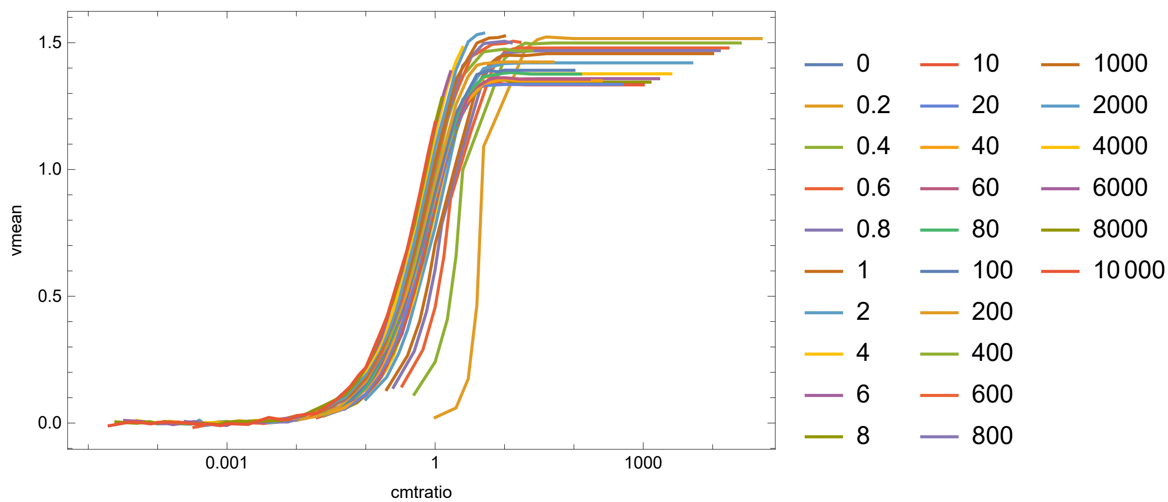
In[77]:=
```
Show[
 dsPlots["cmstrength", "vmean", "temperature", ds2]
 , Frame -> True
]
```

Out[77]=



In[78]:=
```
vplot2 = dsPlots["cmtratio", "vmean", "temperature", ds2];
Show[vplot2
 , Frame -> True
]
```

Out[79]=



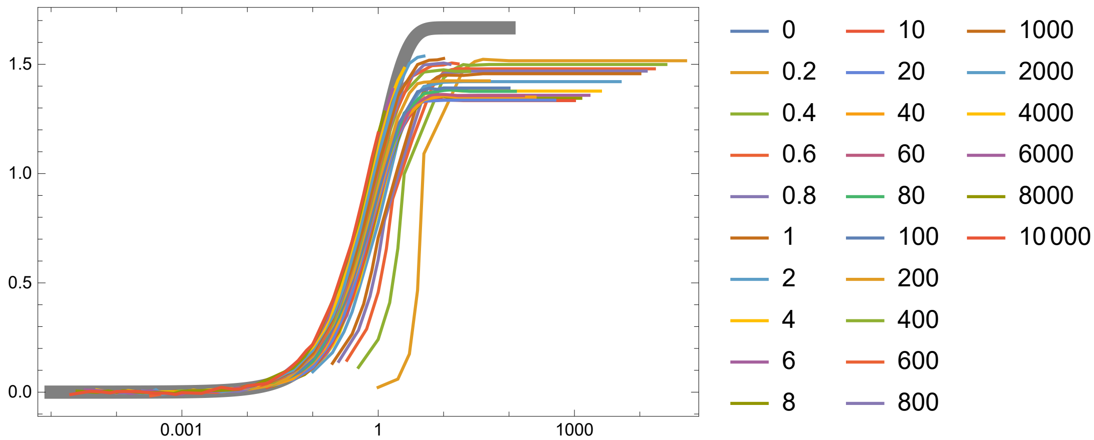
In[80]:=
```
ds2[Select[#["temperature"] == 10000 && #["cmstrength"] == 10000 &]]
```

Out[80]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy	Dxy
10 000	10 000	0.15	1	1500	1.18536	2.12323	2.55923	2.3412

In[81]:=
```
Show[
 LogLinearPlot[(1 - e^-x) / (4 MKtime2), {x, 10^-5, 100}
  , PlotStyle → {Thickness[0.02], Gray}
 ],
 vplot2
 , Frame -> True
 , PlotRange -> All
]
```

Out[81]=



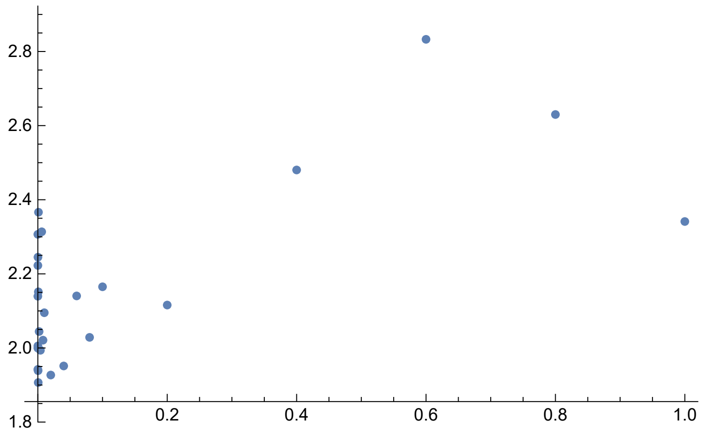
In[82]:=
```
ds2[Select[#["temperature"] == 10000 &]][[All, "Dxy"]]
```

Out[82]=

2.30664	1.94221	2.13957	1.99959	2.22267	2.34123
2.16509	2.09507	2.0055	2.36627	2.11582	1.92697
2.04434	2.245	2.4803	1.95141	1.99374	1.93879
2.83283	2.14044	2.31357	1.90675	2.62995	2.0286
2.02104	2.15148				

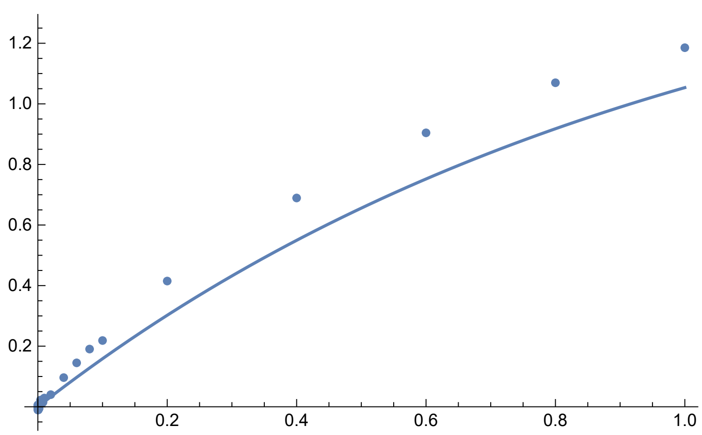
In[83]:=
```
ListPlot[
 ds2[Select[#["temperature"] == 10 000 &]][[All, {"cmtratio", "Dxy"}]]
 , PlotRange → All
]
```

Out[83]=



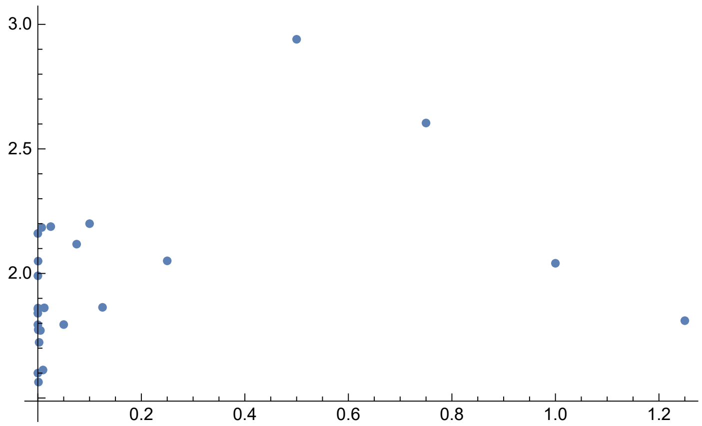
In[84]:=
```
Show[
 ListPlot[
  ds2[Select[#["temperature"] == 10 000 &]][[All, {"cmtratio", "vmean"}]]
  , PlotRange → All
 ],
 Plot[(1 - e^-x) / (4 MKtime2), {x, 0, 1}]
]
```

Out[84]=



In[85]:=
```
ListPlot[
 ds2[Select[#["temperature"] == 8000 &]][[All, {"cmtratio", "Dxy"}]]
 , PlotRange → All
]
```

Out[85]=



OK, let’s figure out what T will give me the closest match to the σ I want. I’ll try to match it at $\mu = 0$.

In[86]:=
```
μ02ds = ds2[Select[PossibleZeroQ[#["cmstrength"]] &]];
μ02ds = μ02ds[SortBy["temperature"]]
```

Out[87]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy
0	0	0.15	∞	1500	0.0000586667	0.00472924	0.
0	0.2	0.15	0.0	1500	0.000133778	0.0202838	0.
0	0.4	0.15	0.0	1500	−0.000402444	0.13506	0.
0	0.6	0.15	0.0	1500	−0.000951619	0.299735	0.
0	0.8	0.15	0.0	1500	−0.00166478	0.406173	0.
0	1	0.15	0	1500	0.00325044	0.627768	0.
0	2	0.15	0	1500	0.00242859	0.641311	0.
0	4	0.15	0	1500	−0.00153308	0.567142	0.
0	6	0.15	0	1500	−0.00116685	1.00051	0.
0	8	0.15	0	1500	0.00355325	0.797968	0.
0	10	0.15	0	1500	0.000444487	0.950335	0.
0	20	0.15	0	1500	−0.00609222	0.987535	0.
0	40	0.15	0	1500	−0.0000319382	1.19626	0.
0	60	0.15	0	1500	0.00672245	0.781455	1.
0	80	0.15	0	1500	0.0027837	0.838457	0.
0	100	0.15	0	1500	0.00292561	1.19553	1.
0	200	0.15	0	1500	−0.000310101	0.938662	1.
0	400	0.15	0	1500	0.00226206	1.17687	1.
0	600	0.15	0	1500	0.00502242	1.53833	1.
0	800	0.15	0	1500	−0.00647744	1.33995	1.

rows 1–20 of 26

In[88]:=
```
σ139 / (gridunitD2 MKtime2)
```

Out[88]= 0.781235 s

In[89]:=
```
width2 = height2 = Quantity[0.2, "Centimeters"];
nx2 = ny2 = 75;
dt2 = Quantity[MKtime2, "Seconds"];
{dx2, dy2} = {width2, height2} / {nx2, ny2}
```

Out[92]= {0.00266667 cm , 0.00266667 cm}

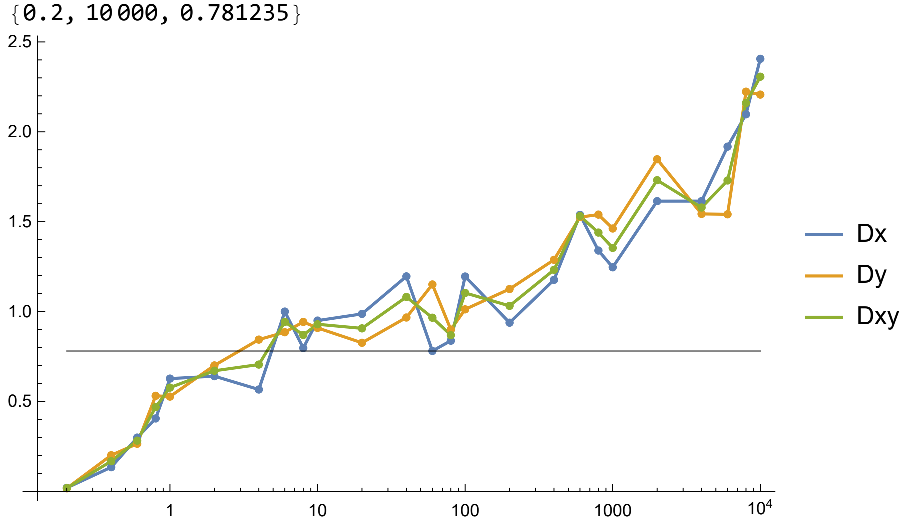
In[93]:=
```
{wormwidth2, wormlength2} = Quantity[{15, 240}, "Micrometers"]
```
Out[93]= {15 μm , 240 μm }

In[94]:=
```
wormsize2 = (wormwidth2 wormlength2) / (dx2 dy2)
```
Out[94]= 5.0625

In[95]:=
```
σ139
```
Out[95]= $5.55545\times10^{-6}\ \mathrm{cm}^2/\mathrm{s}$

In[96]:=
```
gridunitD2 = (dx2 dy2) / dt2
```
Out[96]= $0.0000474074\ \mathrm{cm}^2/\mathrm{s}$

In[97]:=
```
Block[{is = 400, ds = μ02ds, dfs = {"Dx", "Dy", "Dxy"}, data, mint, maxt, targetD},
  ds = ds[Select[! PossibleZeroQ[#["temperature"]] &]];
  targetD = (σ139 / (dx2 dx2)) Quantity[1, "Seconds"];
  data = Table[
    ds[[All, {"temperature", d}]][Values] // Normal,
    {d, dfs}
   ];
  {mint, maxt} = MinMax[data[[All, All, 1]]];
  {
   {mint, maxt, targetD},
   ListLogLinearPlot[data
    , PlotRange → All
    , Joined → True
    , Mesh → All
    , ImageSize → is
    , PlotLegends → dfs
    , Epilog → {Line[{{Log[mint], targetD}, {Log[maxt], targetD}}]}
   ]
  }
 ] // Column
```
Out[97]=
{0.2, 10 000, 0.781235}



Looks like I want $T \approx 6$. I might use $T = 10$ just to have a round number. It's probably close enough, and I can always adjust MKtime if I want to.

In[98]:=
```
μ02ds[[All, {"temperature", "Dxy"}]] // Transpose
```

Out[98]=

temperature	0	0.2	0.4	0.6	0.8	1
	2	4	6	8	10	20
	40	60	80	100	200	400
	600	800	1000	2000	4000	6000
	8000	10 000				
Dxy	0.0049999	0.01902	0.168373	0.282612	0.469109	0.577738
	0.671213	0.705879	0.943108	0.870461	0.929956	0.907204
	1.08189	0.966382	0.869398	1.10423	1.03224	1.23262
	1.532	1.43984	1.35463	1.73114	1.5792	1.72958
	2.1605	2.30664				

OK, now for the velocity, i.e. chemotaxis strength,

In[99]:=
```
dx2
```

Out[99]=
```
0.00266667 cm
```

Here I have an analytical expression that is accurate enough to go on with. In grid units, the velocity, for chemotaxis strength μ, field gradient ∇F, and temperature T, is

$$\begin{aligned} \|v\| &= \frac{1}{4}\left(1 - e^{-\mu\|\nabla F\|/T}\right) \\ &\approx \frac{\mu\|\nabla F\|}{4T} \end{aligned}$$

Strictly speaking, I know this only for the case of $\nabla F = (1, 0)$. But I'm assuming (because I have no good alternative) that the chemotactic response is isotropic and that only the product $\mu \nabla F$ matters, i.e., the argument of the exponential is bilinear in μ and ∇F.

Suppose I have a potential field V in units of $\mathrm{cm}^2\, s^{-1}$. The gradient of this potential has units of velocity, specifically $\mathrm{cm}s^{-1}$. What do I choose for μ? T is given—it was chosen to produce the right diffusion rate σ.) I'm going to assume the velocities are small.

$$\begin{aligned} \nabla F &= \nabla V \mathrm{dx} \\ v &= \nabla V\left(\frac{\mathrm{dt}}{\mathrm{dx}}\right) \\ &\approx \frac{\mu \nabla F}{4T} \end{aligned}$$

$$= \frac{\mu \nabla V \mathrm{dx}}{4\,T}$$

$$\nabla V\left(\frac{\mathrm{dt}}{\mathrm{dx}}\right) \approx \frac{\mu \nabla V \mathrm{dx}}{4\,T}$$

$$1 \approx \frac{\mu\,\mathrm{dx}^2}{4\,T\mathrm{dt}}$$

$$\mu \approx \frac{4\,T\mathrm{dt}}{\mathrm{dx}^2}$$

Here dt is the Metropolis kinetics time, e.g. 0.15 s for the case I'm working on, and dx is the grid spacing. This doesn't seem right. It seems like dx ought to show up in the final computation of μ.

Well, let's try an example. Suppose U_α changes from 0 to 25 000 over a distance of 0.01 cm, which is plausible. Then V_{U_α} changes from 0 to -3×10^{-5} over that distance. So the gradient is $\|\nabla V\| \approx 3 \times 10^{-3}$ cm/s.

In[100]:=
```
Vu[
   U_,
   α_ : 1500,
   β_ : 2 σ139
  ] :=
  - β Log[1 + U / α];
```

In[101]:=
```
{
  "V" → Vu /@ {0, 25 000},
  "∇V" -> Abs[Vu[25 000]] / Quantity[0.01, "Centimeters"],
  "∇F" → (QuantityMagnitude@Abs[Vu[25 000]] / Quantity[0.01, "Centimeters"]) dx2,
  "v" → (Abs[Vu[25 000]] / Quantity[0.01, "Centimeters"]) (dt2 / dx2),
  "μ / T" → (4 dt2) / dx2^2,
  "(μ ∇F) / T" -> ((4 dt2) / dx2^2) (Abs[Vu[25 000]] / Quantity[0.01, "Centimeters"]) dx2,
  "(μ ∇F) / (4 T)" ->
   ((4 dt2) / (4 dx2^2)) (Abs[Vu[25 000]] / Quantity[0.01, "Centimeters"]) dx2
 } //
 Column
```

Out[101]=
```
V → {0. cm²/s , -0.0000319069 cm²/s }
∇V → 0.00319069 cm/s
∇F → 8.50852 × 10^-6
v → 0.179477
μ / T → 84 375. s/cm²
(μ ∇F) / T → 0.717906
(μ ∇F) / (4 T) → 0.179477
```

Yup, looks like that works out.

dt = 0.05

In[102]:=
```
μ01ds = ds1[Select[PossibleZeroQ[#["cmstrength"]] &]];
μ01ds = μ01ds[SortBy["temperature"]]
```

Out[103]=

cmstrength	temperature	MKtime	cmtratio	tfinal	vmean	Dx	Dy
0	0	1	∞	10 000	0.00004655	0.00498621	0.00497
0	0.2	1	0.0	10 000	−0.0000241385	0.00684214	0.00817
0	0.4	1	0.0	10 000	−0.000244853	0.0363752	0.03806
0	0.6	1	0.0	10 000	0.0000673799	0.0681142	0.06008
0	0.8	1	0.0	10 000	0.000129053	0.0972709	0.09794
0	1	1	0	10 000	−0.000549561	0.116088	0.10963
0	2	1	0	10 000	−0.000341603	0.154172	0.10947
0	4	1	0	10 000	−0.000302816	0.176871	0.18867
0	6	1	0	10 000	0.000788141	0.159843	0.15385
0	8	1	0	10 000	0.000209514	0.160657	0.18224
0	10	1	0	10 000	0.000207029	0.169672	0.23937
0	20	1	0	10 000	0.000160617	0.183204	0.20395
0	40	1	0	10 000	0.000346383	0.179997	0.21617
0	60	1	0	10 000	0.000254052	0.205308	0.18284
0	80	1	0	10 000	0.000747709	0.161558	0.20391
0	100	1	0	10 000	−0.0000432087	0.16959	0.21522
0	200	1	0	10 000	0.0010539	0.261324	0.21641
0	400	1	0	10 000	0.000392544	0.202208	0.22995
0	600	1	0	10 000	0.000205537	0.164203	0.17140
0	800	1	0	10 000	−0.00064113	0.212823	0.24169

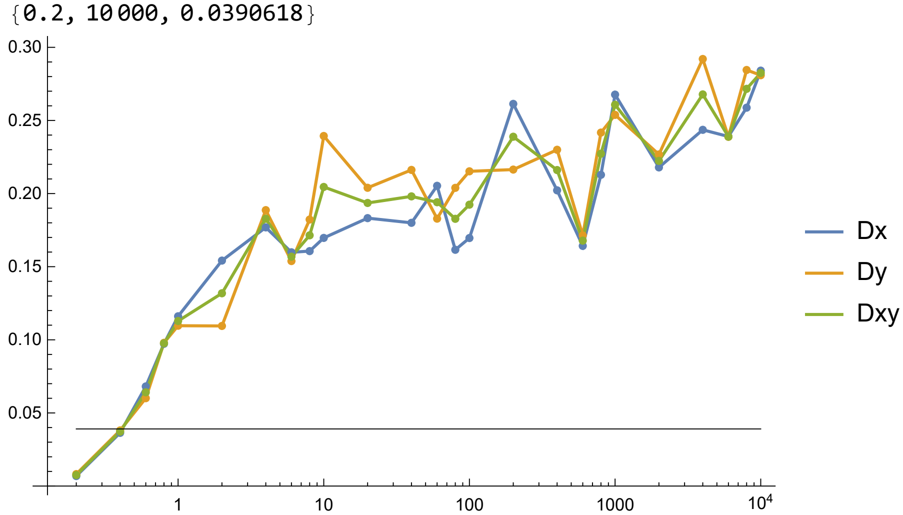
rows 1–20 of 26

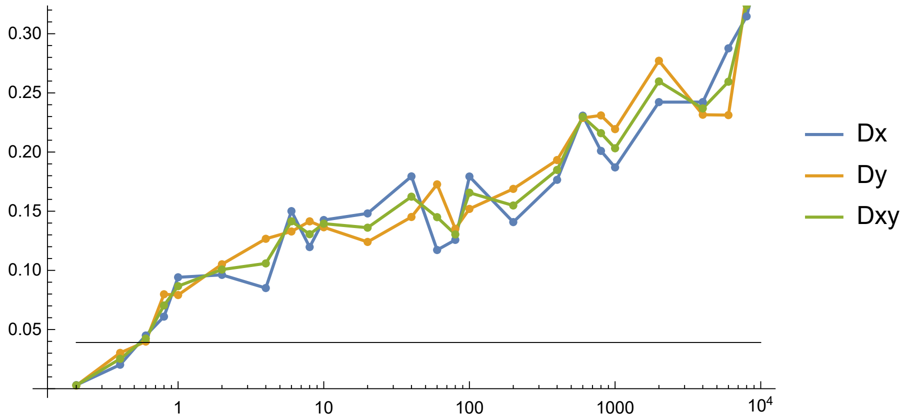
In[104]:=
```
dt3 = Quantity[0.05, "Seconds"]
```

Out[104]= 0.05 s

```
In[105]:= Block[{is = 400, dss = {μ01ds, μ02ds}, dfs = {"Dx", "Dy", "Dxy"}, data, mint, maxt, targetD},
  dss = Table[
    ds[Select[! PossibleZeroQ[#["temperature"]] &]],
    {ds, dss}
   ];
  targetD = (σ139 / (dx2 dx2)) dt3;
  data = Table[
    ds[[All, {"temperature", d}]][Values] // Normal,
    {ds, dss}, {d, dfs}
   ];
  data[[2, All, All, 2]] *= 0.15;
  {mint, maxt} = MinMax[data[[All, All, All, 1]]];
  {
   {mint, maxt, targetD},
   ListLogLinearPlot[data[[1]]
    , PlotRange → All
    , Joined → True
    , Mesh → All
    , ImageSize → is
    , PlotLegends → dfs
    , Epilog → {Line[{{Log[mint], targetD}, {Log[maxt], targetD}}]}
   ],
   ListLogLinearPlot[data[[2]]
    , PlotRange → All
    , Joined → True
    , Mesh → All
    , ImageSize → is
    , PlotLegends → dfs
    , Epilog → {Line[{{Log[mint], targetD}, {Log[maxt], targetD}}]}
   ],
   data
  }
 ] // Column
```

{0.2, 10 000, 0.0390618}





Out[105]=

```
{{{{0.2, 0.00684214}, {0.4, 0.0363752}, {0.6, 0.0681142},
   {0.8, 0.0972709}, {1, 0.116088}, {2, 0.154172}, {4, 0.176871}, {6, 0.159843},
   {8, 0.160657}, {10, 0.169672}, {20, 0.183204}, {40, 0.179997}, {60, 0.205308},
   {80, 0.161558}, {100, 0.16959}, {200, 0.261324}, {400, 0.202208},
   {600, 0.164203}, {800, 0.212823}, {1000, 0.267672}, {2000, 0.217894},
   {4000, 0.243589}, {6000, 0.239021}, {8000, 0.258732}, {10000, 0.284062}},
  {{0.2, 0.00817876}, {0.4, 0.0380614}, {0.6, 0.0600807}, {0.8, 0.0979445},
   {1, 0.109638}, {2, 0.109479}, {4, 0.188675}, {6, 0.153853}, {8, 0.182244},
   {10, 0.239375}, {20, 0.203953}, {40, 0.216172}, {60, 0.182847},
   {80, 0.20391}, {100, 0.215223}, {200, 0.21641}, {400, 0.229952},
   {600, 0.171406}, {800, 0.241698}, {1000, 0.253822}, {2000, 0.226766},
   {4000, 0.292021}, {6000, 0.238811}, {8000, 0.284503}, {10000, 0.280925}},
  {{0.2, 0.00751045}, {0.4, 0.0372183}, {0.6, 0.0640975}, {0.8, 0.0976077},
   {1, 0.112863}, {2, 0.131825}, {4, 0.182773}, {6, 0.156848}, {8, 0.17145},
   {10, 0.204524}, {20, 0.193578}, {40, 0.198085}, {60, 0.194078},
   {80, 0.182734}, {100, 0.192407}, {200, 0.238867}, {400, 0.21608},
   {600, 0.167805}, {800, 0.22726}, {1000, 0.260747}, {2000, 0.22233},
   {4000, 0.267805}, {6000, 0.238916}, {8000, 0.271617}, {10000, 0.282494}}},
 {{{0.2, 0.00304257}, {0.4, 0.020259}, {0.6, 0.0449603}, {0.8, 0.060926},
   {1, 0.0941652}, {2, 0.0961967}, {4, 0.0850713}, {6, 0.150077},
   {8, 0.119695}, {10, 0.14255}, {20, 0.14813}, {40, 0.179439}, {60, 0.117218},
   {80, 0.125769}, {100, 0.17933}, {200, 0.140799}, {400, 0.17653},
   {600, 0.23075}, {800, 0.200992}, {1000, 0.187019}, {2000, 0.242201},
   {4000, 0.242221}, {6000, 0.287654}, {8000, 0.314636}, {10000, 0.360919}},
  {{0.2, 0.00266343}, {0.4, 0.030253}, {0.6, 0.0398233}, {0.8, 0.0798068},
   {1, 0.0791562}, {2, 0.105167}, {4, 0.126692}, {6, 0.132855}, {8, 0.141443},
   {10, 0.136437}, {20, 0.124031}, {40, 0.145128}, {60, 0.172696},
   {80, 0.135051}, {100, 0.151939}, {200, 0.168871}, {400, 0.193256},
   {600, 0.22885}, {800, 0.230961}, {1000, 0.219369}, {2000, 0.277141},
   {4000, 0.231539}, {6000, 0.231221}, {8000, 0.333514}, {10000, 0.331074}},
  {{0.2, 0.002853}, {0.4, 0.025256}, {0.6, 0.0423918}, {0.8, 0.0703664},
   {1, 0.0866607}, {2, 0.100682}, {4, 0.105882}, {6, 0.141466}, {8, 0.130569},
   {10, 0.139493}, {20, 0.136081}, {40, 0.162284}, {60, 0.144957},
   {80, 0.13041}, {100, 0.165634}, {200, 0.154835}, {400, 0.184893},
   {600, 0.2298}, {800, 0.215977}, {1000, 0.203194}, {2000, 0.259671},
   {4000, 0.23688}, {6000, 0.259437}, {8000, 0.324075}, {10000, 0.345996}}}}
```

Equilibrium

Here’s another idea: perhaps I shouldn’t try to match the velocity, but merely the equilibrium. This is fairly straightforward. I know that at equilibrium, the is a Boltzmann equilibrium with temperature σ. But the Metropolis kinetics used for the cellular Potts model is based on a Boltzmann distribution. Thus I get the following simple answer for the chemotactic strength:

$$\mu = \frac{T}{\sigma}$$

I tried that out, and it seems to work pretty well.