Stochastic processes

Lab items

- Random Walks
  - Plotting functions
  - Walk functions
  - Random walk experiments with different walk functions
  - Statistical behavior of random walks
- Stochastic Differential Equations
  - Simple stock growth
  - Fixed points, attractors, repellers
  - Random walk with the Ito process

Random Walks

We examine a few stochastic processes, and practice with random variables.

In[108]:=
```
ClearAll["Global`*"]
```

Generate a pseudorandom real number between 0 and 2 π.

In[109]:=
```
RandomReal[{0, 2 Pi}]
```

Out[109]=
```
3.16175
```

Appropriate square function

We want to plot our solution in a square grid.

In[132]:=
```
IdealSquare[xMin_, xMax_, yMin_, yMax_] := Module[{},
  dx = (xMax - xMin)/2;
  cx = Mean[{xMin, xMax}];
  dy = (yMax - yMin)/2;
  cy = Mean[{yMin, yMax}];

  Which[
   dx ≥ dy,
   yNewMin = cy - dx; yNewMax = cy + dx; {xMin, xMax, yNewMin, yNewMax},
   dy > dx,
   xNewMin = cx - dy; xNewMax = cx + dy; {xNewMin, xNewMax, yMin, yMax}
  ]
 ]
```

Walk functions

Here are a few ways a random walk could be generated.

In[111]:=
```
randomGrid[] := RandomChoice[{{{1}, {0}}, {{-1}, {0}}, {{0}, {1}}, {{0}, {-1}}}];
randomPolar[] := Module[{}, t = RandomReal[{0, 2 Pi}];
   {{Cos[t]}, {Sin[t]}}];
randomGridGeneral[n_] :=
  N[RandomChoice[{{Cos[2 Pi #/n]}, {Sin[2 Pi #/n]}} & /@ Range[n]]];
```

Random walk

We create a random walk function which takes in initial conditions, number of time-steps, and a fixed distance traveled per time-step governed by the walk functions before.

In[114]:=
```
RandomWalk[init_, steps_, distance_, OptionsPattern[
   {plot → True, degrees → {1, -1}, WalkSpace → randomPolar}]] := Module[{},
  solutions = {init};
  (* The purpose of degrees is to select a range
   of points with respect to degree away from the first. *)

  {start, end} = OptionValue[degrees];

  curStep = 2;

  (* To ensure that our final plot has a 1:1 coordinate ratio,
  we find the smallest and largest x and y values, respectively. *)
  xMin = init[[1]];
  xMax = init[[1]];
```

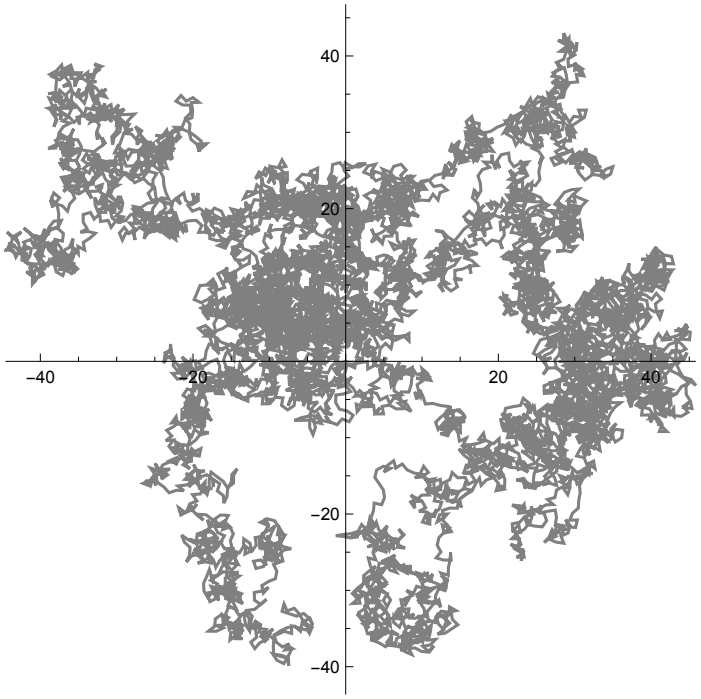
```
  yMin = init[[2]];
  yMax = init[[2]];

  direction = OptionValue[WalkSpace];

  While[curStep ≤ steps,
   curSol = solutions[[curStep - 1]];
   dir = direction[];
   newSol = curSol + distance direction[];

   (* xMin, xMax, yMin, yMax are re-evaluated at every iteration. *)
   xMin = Min[xMin, newSol[[1]]];
   xMax = Max[xMax, newSol[[1]]];
   yMin = Min[yMin, newSol[[2]]];
   yMax = Max[yMax, newSol[[2]]];
   AppendTo[solutions, newSol];
   curStep += 1];

  (* Calculate ideal square. *)
  {xMin, xMax, yMin, yMax} = IdealSquare[xMin, xMax, yMin, yMax];

  solutions = Level[solutions[[#]], {-1}] & /@ Range[steps];
  If[OptionValue[plot],
   ListPlot[
    solutions[[start ;; end]],
    AspectRatio → 1,
    (*Axes→False,*)
    (*ImageSize→1024,*)
    Joined → True,
    PlotRange → {{xMin, xMax}, {yMin, yMax}},
    PlotStyle → Gray
   ],
   {solutions[[start ;; end]], xMin, xMax, yMin, yMax}]
 ]
```

Experiments with RandomWalk

A random walk of 10,000 steps, using **randomPolar**.

In[115]:=
```
RandomWalk[{{0}, {0}}, 10 000, 1]
```

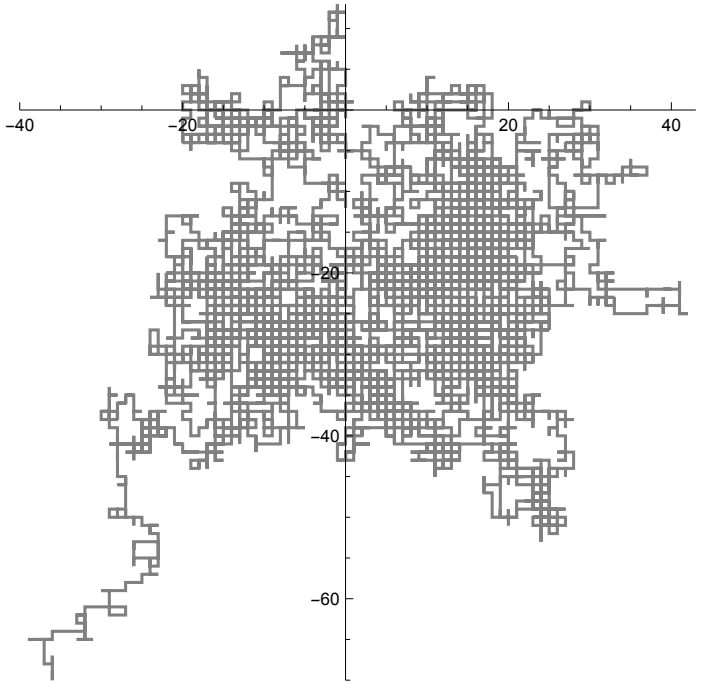
Out[115]=



A random walk of 10,000 steps, using **randomGrid** orientations.

In[117]:=
```
RandomWalk[{{0}, {0}}, 10 000, 1, WalkSpace → randomGrid]
```

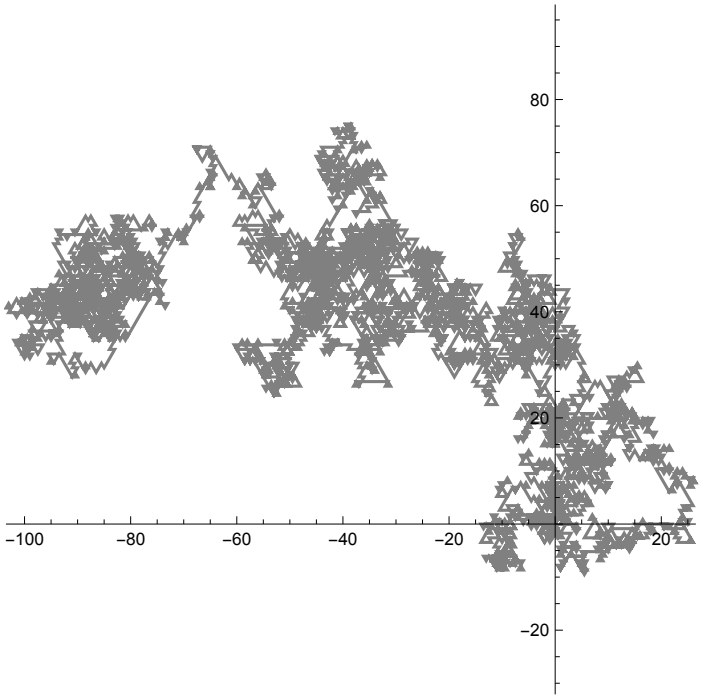
Out[117]=



A random walk of 10,000 steps, in which the possible walk orientations are $\left\{0, \frac{2\pi}{3}, \frac{4\pi}{3}\right\}$.

In[118]:=
```
RandomWalk[{{0}, {0}}, 10 000, 1, WalkSpace → (randomGridGeneral[3] &)]
```

Out[118]=

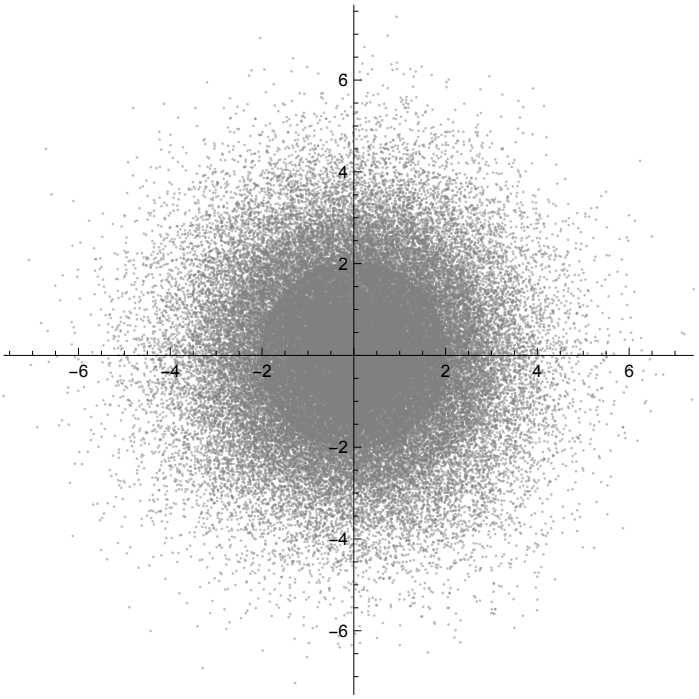


Density of a random walk

Perhaps we are less concerned about the path, and more about the points traversed by many walks. What does the point distribution look like for 10,000 walks?

In[119]:=
```
With[{packages = RandomWalk[{{0}, {0}}, 10, 1, plot → False] & /@ Range[10 000]},
 {solutions, xMins, xMaxs, yMins, yMaxs} = Transpose[packages];
 solutions = Flatten[solutions, 1];
 xMin = Min[xMins];
 xMax = Max[xMaxs];
 yMin = Min[yMins];
 yMax = Max[yMaxs];
 {xMin, xMax, yMin, yMax} = IdealSquare[xMin, xMax, yMin, yMax];

 Show[
  ListPlot[
   solutions,
   AspectRatio → 1,
   (*ImageSize→1024,*)
   PlotRange → {{xMin, xMax}, {yMin, yMax}},
   PlotStyle → {Opacity[.5], Gray}
  ]]]
```

Out[119]=



Stochastic differential equations

Stock growth

The DE $dx = a\,dt$ has the general solution $x(t) = a\,t + x_0$ for some initial condition x_0; in this example, we interpret the constant a as the growth rate of some market. We are interested in the stochastic case of

the simple interest equation, $dx = a\,dt + b\,dW$. In this case, we interpret the constant b as the volatility of the market.

In[120]:=
```
ClearAll["Global`*"]
```

In[121]:=
```
proc = ItoProcess[dx[t] == 1 dt + 1 dw[t], x[t], {x, 0}, t, w ≈ WienerProcess[]]
```

Out[121]=
```
ItoProcess[{{1}, {{1}}, x[t]}, {{x}, {0}}, {t, 0}]
```

In[122]:=
```
Mean[proc[t]]
```

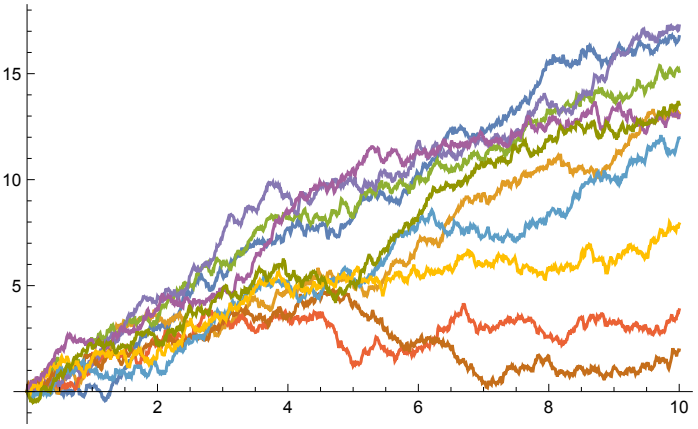
Out[122]=
```
t
```

In[123]:=
```
Variance[proc[t]]
```

Out[123]=
```
t
```

In[124]:=
```
ListLinePlot[RandomFunction[proc, {0, 10, .01}, 10](*,ImageSize→1024*)]
```

Out[124]=



Fixed points, attraction, repulsion

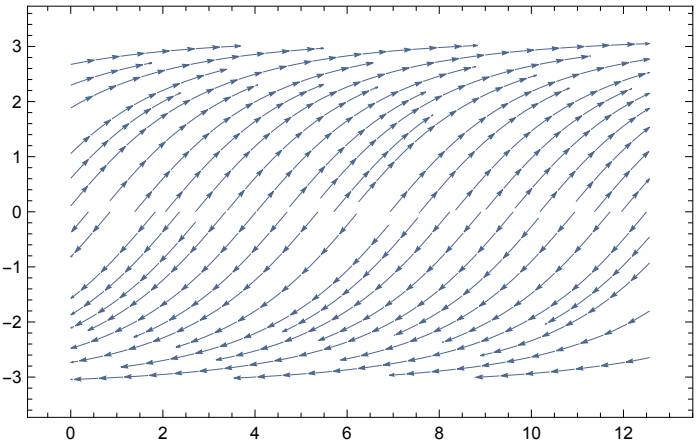
We compute some solutions to the SDE $dx = \sin x\,dt + \sin t\,dW$, where W(t) is a standard Wiener process. (Jun. 14)

In[125]:=
```
ClearAll["Global`*"]
```

First, consider the ODE $dx = \sin x\,dt$. The system is described by the following phase space.

In[126]:=
```
StreamPlot[{x, Sin[x]}, {t, 0, 4 Pi}, {x, -Pi, Pi},
 StreamScale → Tiny, AspectRatio → GoldenRatio^-1 (*,ImageSize→1024*)]
```

Out[126]=

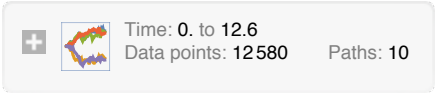


Next, consider the stochastic counterpart.

In[127]:=
```
proc =
 ItoProcess[dx[t] == Sin[x[t]] dt + Sin[t] dw[t], x[t], {x, 0}, t, w ≈ WienerProcess[]]
```

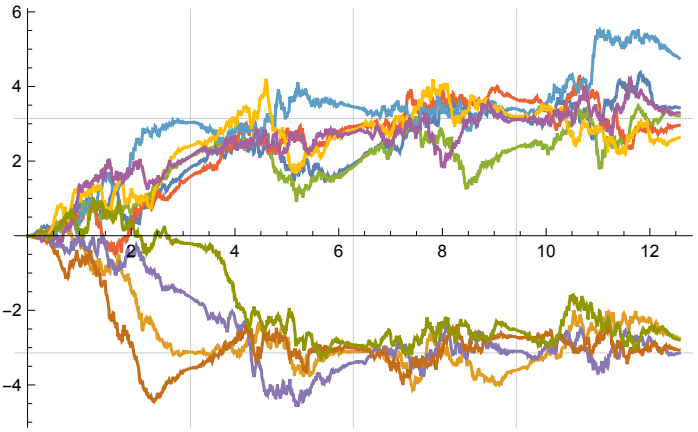
Out[127]=
```
ItoProcess[{{Sin[x[t]]}, {{Sin[t]}}, x[t]}, {{x}, {0}}, {t, 0}]
```

In[128]:=
```
RandomFunction[proc, {0, 4 Pi, .01}, 10]
ListLinePlot[%, GridLines → {(Pi #) & /@ Range[3], {-Pi, Pi}} (*,ImageSize→1024*)]
```

Out[128]=
```
TemporalData[
```


```
]
```

Out[129]=



Parametric Ito process

We try a 2-dimensional random walk using the Wiener process.

In[130]:=
```
ClearAll["Global`*"]
```

In[131]:=
```
proc = ItoProcess[d x[t] == d w[t], x[t], {x, 0}, t, w ≃ WienerProcess[]]
```

Out[131]=
```
ItoProcess[{{0}, {{1}}, x[t]}, {{x}, {0}}, {t, 0}]
```

In[136]:=
```
{xx, yy} = Normal[RandomFunction[proc, {0, 5, .0001}, 2]];
{xMin, xMax, yMin, yMax} = IdealSquare[Min[xx], Max[xx], Min[yy], Max[yy]];
```

In[138]:=
```
ListLinePlot[Transpose[{Transpose[xx][[2]], Transpose[yy][[2]]}],
 PlotRange → {{xMin, xMax}, {yMin, yMax}}, AspectRatio → 1(*,ImageSize→1024*)]
```

Out[138]=