Jupiter’s red spot and 2D turbulence

Here we implement Onsager’s vortex model for two-dimensional turbulent flow. Two-dimensional turbulence is quite different from turbulence in three dimensions. In three dimensions, large eddies stretch and fold into smaller eddies, leading to whirly motion on all scales. In two dimensions, one often finds that the swirly eddies combine into one big eddy. Jupiter’s red spot and hurricanes are thought to be approximately explained as the result of 2D turbulence in their atmospheres.

Jupiter’s Red Spot. Jupiter’s atmosphere is stormy, with winds that can exceed 400 miles per hour in a complex, turbulent pattern. Jupiter’s `red spot’ is a giant storm 3 1/2 times the size of the earth, that has lasted for at least 186 years. (Image from Voyager I,NASA’s Goddard Space Flight Center, courtesy of NASA/JPL, http://www.nasa.gov/content/jupiters-great-red-spot-viewed-by-voyager-i.

Setting up the Hamiltonians, the equations of motion, and various routines for solving and plotting...

WARNING: Code will break if you assign anything to x, y, Γ, or t. Instead, use something like x0, y0, Γ20, ...

Apologies for my inelegant formulation...

```
ClearAll[x, y, Γ, t]
```

```
H[x_, y_, Γ_] := Sum[
   - (Γ[[i]] Γ[[j]] / π) Log[ (x[[i]] - x[[j]]) ^2 + (y[[i]] - y[[j]]) ^2],
   {i, 1, Length[Γ]}, {j, i + 1, Length[Γ]}]

(* H = 1/2 sum(H[i]) because of double counting in sum *)
H[i_][x_, y_, Γ_] :=
  - (Γ[[i]] / π)
   (Sum[Γ[[j]] Log[ (x[[i]] - x[[j]]) ^2 + (y[[i]] - y[[j]]) ^2], {j, 1, i - 1}] +
     Sum[Γ[[j]] Log[ (x[[i]] - x[[j]]) ^2 + (y[[i]] - y[[j]]) ^2], {j, i + 1, Length[Γ]}])

xOft[nVortices_] := Table[x[i][t], {i, 1, nVortices}]
yOft[nVortices_] := Table[y[i][t], {i, 1, nVortices}]
ΓList[nVortices_] := Table[Γ[i], {i, 1, nVortices}]

AnalyticH[nVortices_] := AnalyticH[nVortices] =
  H[xOft[nVortices], yOft[nVortices], ΓList[nVortices]] // Simplify

Equations[Γ0_] :=
 Flatten[Table[{x[i] '[t] == - (1 / (2 π)) (Sum[Γ0[[j]] (y[i][t] - y[j][t]) /
          ((x[i][t] - x[j][t]) ^2 + (y[i][t] - y[j][t]) ^2), {j, 1, i - 1}] +
        Sum[Γ0[[j]] (y[i][t] - y[j][t]) / ((x[i][t] - x[j][t]) ^2 +
            (y[i][t] - y[j][t]) ^2), {j, i + 1, Length[Γ0]}]),
     y[i] '[t] == (1 / (2 π)) (Sum[Γ0[[j]] (x[i][t] - x[j][t]) /
          ((x[i][t] - x[j][t]) ^2 + (y[i][t] - y[j][t]) ^2), {j, 1, i - 1}] + Sum[Γ0[[j]]
         (x[i][t] - x[j][t]) / ((x[i][t] - x[j][t]) ^2 + (y[i][t] - y[j][t]) ^2),
        {j, i + 1, Length[Γ0]}])}, {i, 1, Length[Γ0]}]]

InitialConditions[x0_, y0_] :=
 Flatten[Table[{x[i][0] == x0[[i]], y[i][0] == y0[[i]]}, {i, 1, Length[x0]}]]

RunSimulation[x0_, y0_, Γ0_, tMax_] :=
 NDSolve[Evaluate[Flatten[{Equations[Γ0], InitialConditions[x0, y0]}]],
   Flatten[{xOft[Length[Γ0]], yOft[Length[Γ0]]}], {t, 0, tMax}][[1]]

RandomScatter[nVortices_] :=
 Block[{Γ = RandomReal[{-1, 1}, nVortices], r = Sqrt[RandomReal[{0, 1}, nVortices]],
   θ = RandomReal[{-π, π}, nVortices], x0 = r Cos[θ], y0 = r Sin[θ]}, {Γ, x0, y0}]

MetropolisStep[i_, β_] :=
 Module[{Ei = H[i][x0, y0, Γ0], r = Sqrt[RandomReal[]], θ = RandomReal[{-π, π}],
   xp = x0, yp = y0, Ef, βΔE, rand = RandomReal[]}, xp[[i]] = r Cos[θ];
  yp[[i]] = r Sin[θ]; Ef = H[i][xp, yp, Γ0]; βΔE = β (Ef - Ei);
  If[βΔE < 0 || Exp[-βΔE] > rand, {x0[[i]] = r Cos[θ]; y0[[i]] = r Sin[θ]}];]

Metropolis[nSweeps_, β_] :=
  {nTries = Length[x0] nSweeps; For[try = 1, try ≤ nTries,
    try += 1, MetropolisStep[RandomInteger[{1, Length[x0]}], β]];};

ShowSimulation[x0_, y0_, Γ0_] :=
 Module[{colors = Table[Blend[{Red, Blue}, (1 + Γ0[[i]]) / 2], {i, 1, Length[Γ0]}]},
  Show[Graphics[{White, Rectangle[{-1, -1}, {1, 1}]}], Graphics[{Circle[{0, 0}, 1],
     {PointSize[0.1], Point[Transpose[{x0, y0}], VertexColors → colors]}}]]]
```

```
AnimateSimulation[x0_, y0_, Γ0_, tMax_] :=
 Module[{sol = RunSimulation[x0, y0, Γ0, tMax],
   colors = Table[Blend[{Red, Blue}, (1 + Γ0[[i]]) / 2], {i, 1, Length[Γ0]}]},
  xt = (x0ft[Length[Γ0]] /. sol);
  yt = (y0ft[Length[Γ0]] /. sol);
  Manipulate[Show[Graphics[{White, Rectangle[{-2, -2}, {2, 2}]}],
    Graphics[{Circle[{0, 0}, 1], {PointSize[0.05], Point[Transpose[{xt, yt}] /.
         t → time, VertexColors → colors]}}]], {time, 0, tMax}]]

minDistance[x0_, y0_] := Min[
  Table[Sqrt[(x0[[i]] - x0[[j]])^2 + (y0[[i]] - y0[[j]])^2 + KroneckerDelta[i, j]],
   {i, 1, Length[x0]}, {j, 1, Length[x0]}]]
```

[Analytically, do part (a), discussing the flow field around a point vortex.]

(b) Run the simulation with one vortex with Γ=1 and a `tracer vortex' with Γ=0. Does the tracer vortex rotate around the test vortex with velocity u0?

```
nVortices = 2;
Γ0 = {1., 0.};
x0 = {0., 0.5};
y0 = {0., 0.};
tMax = 100.;
AnimateSimulation[x0, y0, Γ0, tMax]
```

Do parts (c) and (d) analytically.

(e) Start with n=20 vortices with a random distribution of vortex strengths Γ in [−1,1] and random positions within the unit circle. Print the original configuration. Run for a time t=10, and print the final configuration. Do you see the spontaneous formation of a giant whirlpool? Are the final positions roughly also randomly arranged? Measure and report the energy H of your vortex configuration. (Warning: Sometimes the differential equation solver crashes. Just restart again with a new set of random initial conditions.)

```
{Γ0, x0, y0} = RandomScatter[20];
tMax = 10.;
AnimateSimulation[x0, y0, Γ0, tMax]
```

Do parts (f), (g), and (h) analytically.

(i) Thermalize the Monte-Carlo simulation for your n=20 vortices at temperature β=1/(kBT)=2, report the final energy, and print out your configuration. Does the vortex simulation look similar to the ones you found in part~(e)?

Thermalize again at temperature β=−2, report the energy, and print out your final configuration. Do the vortices separate out into clumps of positive and negative vorticity?

```
{Γ0, x0, y0} = RandomScatter[20];
tMax = 10.;
H[x0, y0, Γ0]
Metropolis[1000, 2];
H[x0, y0, Γ0]
ShowSimulation[x0, y0, Γ0]
```

(j) Re-run the Monte Carlo simulation with n=20 vortices until you find a configuration with a clear separation of positive and negative vortices (say, by examining the energy), but where the minimum distance between vortices is not too small (say, bigger than 0.01). Print this initial configuration of vortices. Run the simulation for t=10 with this state as the initial condition. How many hurricanes do you find? Print out the final configuration of vortices.

```
{Γ0, x0, y0} = RandomScatter[20];
tMax = 10.;
{H[x0, y0, Γ0], minDistance[x0, y0]}
Metropolis[1000, -2.];
{H[x0, y0, Γ0], minDistance[x0, y0]}
```

```
{Γ0, x0, y0} = RandomScatter[20];
tMax = 10.;
{H[x0, y0, Γ0], minDistance[x0, y0]}
Metropolis[1000, ...];
{H[x0, y0, Γ0], minDistance[x0, y0]}
```

```
AnimateSimulation[x0, y0, Γ0, tMax]
```