Excursions in Computing Science: Book 9c. Heat: Histograms and Gases Part II. Heat.

T. H. Merrett*
McGill University, Montreal, Canada

May 30, 2013

I. Prefatory Notes

12. A gas simulation 1: the collisions The behaviour of gases is a perfect topic to study in aggregate because a) it's impossible for us to follow the collisions of some 10^{23} (0.1 yotta) gas molecules and b) with so many molecules their aggregates can be measured as precisely as simple physical properties.

The important aggregates are averages and totals: the *temperature* of a gas is the mean energy (per unit time per unit volume), and its *pressure* is the total change of the momentum of its molecules at the walls of the container. Other important "thermodynamic" properties are related to our ignorance of the energy, a quantity with the special name of *entropy*.

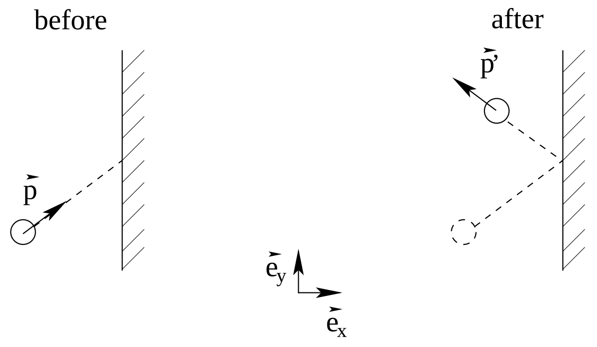
We are best to start with much fewer than 10^{23} molecules and to write a computer simulation to

*Copyleft ©T. H. Merrett, 2012, 2013 Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation in a prominent place. Copyright for components of this work owned by others than T. H. Merrett must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to republish from: T. H. Merrett, School of Computer Science, McGill University, fax 514 398 3883.

study the behaviour of 100 or even 10 molecules.

An *ideal gas* is an abstraction whose molecules do not collide with each other but only with the walls of the container. This is enough to produce the quantities of temperature and pressure, but ignorance must always be constant. With "elastic" collisions at the walls (i.e., collisions in which not only momentum is conserved, as it always must be, but also kinetic energy is conserved—it does not convert to energies of disturbances within the wall) the molecules will just reverse the horizontal components of their motion (at a vertical wall) or the vertical components (at a horiontal wall) and each molecule will keep its original energy so that our ignorance of the energies does not change. Indeed, if we know all the initial energies, our ignorance initially is, and remains, zero.

Here are the results of an elastic collision with a vertical wall.



$$\begin{aligned} \vec{p} &= p_x\vec{e_x} + p_y\vec{e_y} & \qquad \vec{p'} &= -p_x\vec{e_x} + p_y\vec{e_y} \\ K &= \frac{\vec{p}^2}{2m} & \qquad K' &= \frac{\vec{p}^2}{2m} = K \end{aligned}$$

And we see that the kinetic energies before and after (the only kind of energy we are considering) are the same.

(I've cheated here, of course: the *wall* should move when the molecule hits it, for conservation of momentum (Newton's thrd law). So I've pretended either that the wall has infinite mass or that it is rigidly attached to an opposite wall which is struck simultaneously by a molecule with exactly the opposite horizontal momentum.)

An ideal gas can be imagined as having molecules which are all points, so the probability of them hitting each other is zero.

To mix the initial energies up we need a gas of finite-sized molecules which do hit each other. In the ideal world of pre-quantum physics all these collisions can be tracked and the momenta and energies calculated exactly before and after each collision. This will be exactly what we do in our computer simulation of the gas.

But in practice we soon lose track, if only because we could not retain enough significant figures in the calculation to recall every momentum and energy exactly. So our ignorance will increase.

These finite-radius molecules are what we will simulate. For a start, we do make other simplifying assumptions. For instance, we will suppose that the molecules behave like hard billiard balls so that their collisions with each other will be elastic. For example, a collision does not excite internal vibrations in the molecules, which would siphon off some of the kinetic energy. In nature, this assumption is most closely matched by *monatomic* molecules, such as occur in the chemically relatively inert "noble gases", e.g., helium, neon and argon.

We'll need to know radii and masses of these "billiard balls", so we look them up.

Noble gas	He	Ne	Ar
radius, nm	0.049	0.051	0.088
mass, AMU	4	20	40

The Atomic Mass Unit effectively counts nucleons in the atom. So helium, with 2 protons and 2 neutrons, has an AMU of 4.

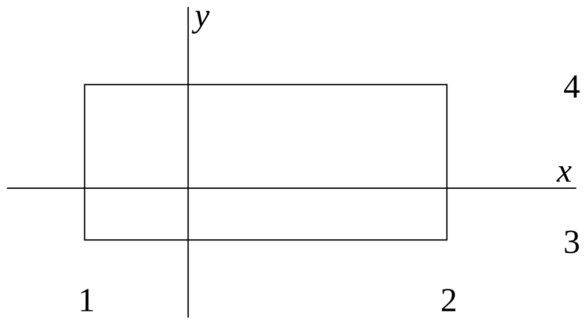
To get kinetic energies in meV (milli electron volts) from speeds of nm/psec (nanometers per picosecond, i.e., kilometers per second) and masses in AMU, we multiply the mass by 10.36.

Here is the MATLAB code we can use for a) collisions with the walls of the container and b) collisions between two molecules.

a) Wall collision code is self-evident: just reverse the appropriate momenta.

```
% function p = molCollWallCountBot(pb,1,1,abcd,r,c,0)           THM     101025
% given vectors momentum pb, box boundaries abcd, position c; scalar radius r
% find new momentum p
function p = molCollWallCountBot(pb,tmult,bmult,abcd,r,c,cntB)
% tmult = 1; bmult = 1; cntB = 0;               % for future refinement
abr = abcd - [-r,r,-r,r];       % extend wall to _centre_ of molecule
p = pb;                         % initialize: momentum out = momentum in
if c(1) < abr(1) | c(1) > abr(2) | c(2) < abr(3) | c(2) > abr(4)
  display(['molCollWallCountBot: ' num2str(c') ' out of box ' num2str(abr)])
elseif (c(1)==abr(1) | c(1)==abr(2)) & (c(2)==abr(3) | c(2)==abr(4)), p = -pb;
                                 % 4 corners: reverse momentum
elseif c(1)==abr(1) | c(1)==abr(2), p(1) = -pb(1);%side wall: reverse x-momentum
elseif c(2) == abr(4), p(2) = -pb(2);   %top: reverse y-momentum
elseif c(2) == abr(3), p(2) = -pb(2);   %bot: reverse y-momentum
else display('molCollWallCountBot: no wall collision')
end %if
```

I've represented the container as a rectangular box from x = `abcd(1)` to x = `abcd(2)` and from y = `abcd(3)` to y = `abcd(4)`.



(It will turn out to be useful to allow this flexibility. The simulation program, however, will *start* with only two numbers, `ab(1)` and `ab(2)` and will initialize

`abcd(1) = −ab(1), abcd(2) = ab(1), abcd(3) = −ab(2), abcd(4) = ab(2)`

thus starting symmetrically about the x- and y-axes.)

I will use a container initially 40×20 nm, specified as `abcd = [−20,20,−10,10]`

b) Intermolecular collisions just follow the physics of the *Billiards* Excursion of of Book 8c, Part IV

```
% function [p1,p2] = moleculeCollide(m1,m2,p1b,p2b,0,C12)  THM          101025
% input masses (units don't matter: only ratios used)
%       initial momenta (typically opposite signs): column vectors
%       C12 is column vector connecting centres at collision time = c2 - c1
% output momentum column vectors
function [p1,p2] = moleculeCollide(m1,m2,p1b,p2b,mhf,C12)
mhf = 0;                % for future refinement
pc = p1b + p2b;         % momentum of CoM
a = m1/(m1+m2);         % dimensionless ratio for conversion to CoM
p1bc = p1b - a*pc;      % CoM momentum of first molecule
p2bc = p2b - (1-a)*pc;  % CoM momentum of second molecule       % p2bc = - p1bc
c12 = C12/sqrt(C12'*C12);       % normalize centre-to-centre vector
dim=size(c12);
reflect = 2*c12*c12' - eye(dim(1));     % reflection matrix: Week 7c Note 8
p1c = -reflect*p1bc;    % CoM momentum after collision
p2c = -p1c;             % CoM momentum after collision
p1 = p1c + a*pc;        % convert back from CoM coordinates
p2 = p2c + (1-a)*pc;    % ditto
```

But these routines are ony two of many building blocks needed for the simulation. The centres shown (c in `molCollWallCountBot()` and *C12* in `moleculeCollide()`) are the positions of the molecules at the moment of collision. We must get the molecules there. And we must figure out, among the 10 or 100 molecules of the simulation, *which* collision will happen next, so that we can move all the molecules up to that point in time and then execute that next collision, be it with a wall (10 or 100 possibilities, respectively) or with another molecule (10!2 or 100!2 possibilities).

To find out when a given molecule at position $\vec{c}$ and with momentum $\vec{p}$ will hit some wall we must find its distance from the wall and its velocity in the direction of the wall. We must do this for each of the four walls and find the minimum time.

The code stores in `record` for each wall: the time to reach the wall and the x- and y-position of the molecule when it does. Then it uses `minrows` to find the shortest of these four times.

```
% function [t,c] = molCollWallWhen(cb,p,M,r,abcd)       THM             101025
% find
%   time of collision, t
%   centre after collision:
%     [a-r;y] for right, [r-a;y] for left, [x;b-r] for top, [x,r-b] for bottom
%     - could use sum of appropriate for corner, but let's default to wall
%       if a tie, then schedule an immediate hit on top or bottom
% given
%   centre of molecule at time 0, column vector cb
%   momentum, column vector p
%   mass (amu), M
%   vector abcd describing box (abcd(1),abcd(2))*(abcd(3),abcd(4))
% uses  v = condif(cond,then,els);
%       z = implies(x,y);
%       found = nonnegrows(matrix);
%       found = minrows(matrix);
function [t,c] = molCollWallWhen(cb,p,M,r,abcd)
 m = M*10.36;           % convert from amu to milli-eV (ps/nm)^2
 v = p/m;               % velocity
% distances from location  cb  to wall or middle horizontal diaphragm
 dap = abcd(2) - r - cb(1);     % positive distance to vertical wall
 dav = abcd(1) + r - cb(1);     % negative distance to vertical wall
 dbp = abcd(4) - r - cb(2);     % positive distance to horizontal wall
 dbv = abcd(3) + r - cb(2);     % negative distance to horizontal wall
% if distance==0 then depend on sign of  v ; time is dist/v; new molecule pos, c
 record(1,:) = condif(implies(dap==0,v(1)>0),[dap/v(1),abcd(2)-r,cb(2)+v(2)*dap/v(1)],[inf,0,0]);
 record(2,:) = condif(implies(dav==0,v(1)<0),[dav/v(1),abcd(1)+r,cb(2)+v(2)*dav/v(1)],[inf,0,0]);
 record(3,:) = condif(implies(dbp==0,v(2)>0),[dbp/v(2),cb(1)+v(1)*dbp/v(2),abcd(4)-r],[inf,0,0]);
 record(4,:) = condif(implies(dbv==0,v(2)<0),[dbv/v(2),cb(1)+v(1)*dbv/v(2),abcd(3)+r],[inf,0,0]);
 next = minrows(nonnegrows(record));    % 2 identical if corner, else 1
 sizeNext = size(next);                 % display test only
 t = next(1,1);                 % time is smallest of the 5 times
 c = [next(1,2);next(1,3)];     % new position of molecule 4 corresponding event
```

Since the `molCollWallWhen()` code also uses four other routines, they are, in order of invocation,

```
% function v = condif(cond,then,els)                    THM             101025
% conditional expression: getting around a MATLAB omission
function v = condif(cond,then,els)
if cond, v = then; else v = els; end % if


% function z = implies(x,y)                     THM                     101025
% the Boolean expression x => y ("if x then y") is formally equivalent to
%                        ~x|y, which is the same as y|~x ("y or not x")
function z = implies(x,y)
z = ~x | y;


% function found = nonnegrows(matrix)                   THM             101025
% find rows of matrix whose first element is nonnegative
function found = nonnegrows(matrix)
sizeMatrix = size(matrix);
found = [];
for j = 1:sizeMatrix(1)
  if matrix(j,1)>=0, found = [found; matrix(j,:)]; end % if
end % for j


% function found = minrows(matrix)                      THM             101025
% return all rows of a rectangular matrix with smallest first element
function found = minrows(matrix)
sizeMatrix = size(matrix);
found = matrix(1,:);
for j = 2:sizeMatrix(1)
  if matrix(j,1) < found(1,1)
    found = matrix(j,:);
  elseif matrix(j,1) == found(1,1)
    found = [found; matrix(j,:)];
  end % if compare
end % for j
```

The tasks performed by these auxiliary functions should be self-evident from the commented code. But I should say something about why they are used.

First, `minrows()` selects the closest wall by finding the least of the times to each wall. If the molecule is going to hit a corner it will hit two walls simultaneously and `minrows()` notes that outcome by returning two rows.

Second, a "time" might be negative because the distance to the wall might have a different sign from the appropriate component of the molecular velocity. The molecule will never reach that wall (on its present trajectory) and `nonnegrows()` discards the record for that wall.

Third, `condif()` allows us to write in one expression what in MATLAB would require two statements. MATLAB supports conditionl *statements* but not the more elegant conditional *expression.*

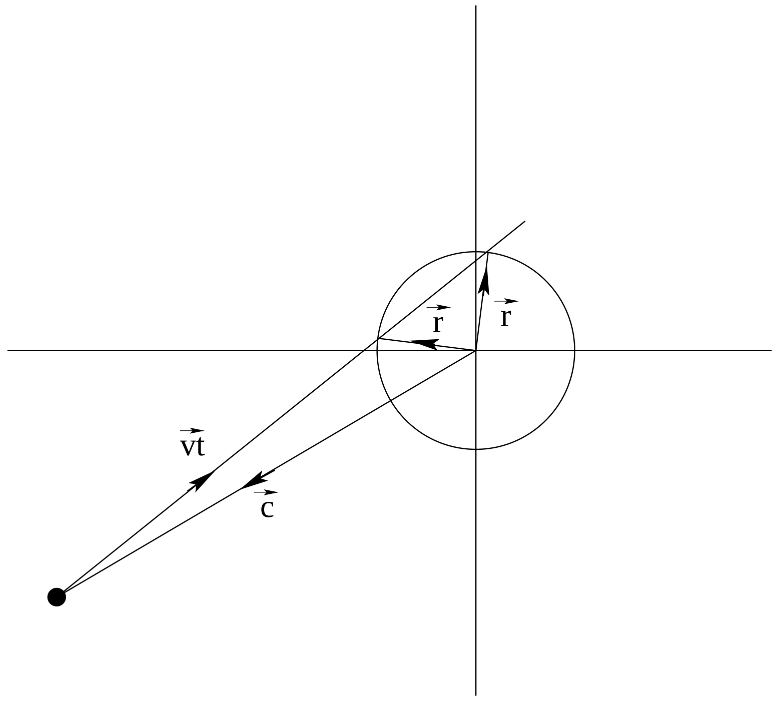
Fourth, the tricky logic of implication addresses the possibility that the molecules may be already at the wall: I used the sign of the velocity in the following way.

IF	molec. at wall	vel. toward wall	THEN
	`F`	`F`	find the hit (time −ve)
	`F`	`T`	find the hit (time +ve)
	`T`	`F`	it's already hit: set time = 0
	`T`	`T`	find the hit (time 0)

This truth table is just that for implication (Week 10, Note 3) so I've called it that.

Finding out when two given molecules will collide with each other is easier if we transform the positions and velocities so that one of the molecules is a) at the origin and b) made stationary. That is, we shift the origin of our coordinate system and we make the coordinate system move along with that molecule: relativity tells us that the physics is unaffected.

We also fatten the now stationary molecule so its radius is the sum of the two radii, and we correspondingly shrink the other molecule to a point. The collision problem has become one of finding the intersection of a line (the trajectory of the centre of the moving molecule) and a circle (the $r_1 + r_2$ circle of the combined radii). There will be two intersection points in general, of which we want the one reached first.



From the diagram, $\vec{c} + \vec{v}t = \vec{r}$, so, in particular, the lengths of these two vector are the same

$$\begin{aligned}\vec{r}.\vec{r} &= (\vec{c} + \vec{v}t).(\vec{c} + \vec{v}t)\\ &= \vec{c}.\vec{c} + 2\vec{c}.\vec{v}t + \vec{v}.\vec{v}t^2\end{aligned}$$

so the quadratic equation we must solve, $\alpha t^2 + \beta t + \gamma = 0$, has

$$\begin{aligned}\alpha &= \vec{v}.\vec{v}\\ \beta &= 2\vec{c}.\vec{v}\\ \gamma &= \vec{c}.\vec{c} - \vec{r}.\vec{r}\end{aligned}$$

and we know $\vec{c}, \vec{v}$ and $\vec{r}$.

This is solved using `quadratic(a,b,c)`. What does it mean if the result has a nonzero imaginary part?

Here is the MATLAB code. The final stage in it, of course, is to restore the coordinate axes to their original position and motionless state.

```
% function [t,c1,c2] = moleculeCollWhen(c1b,c2b,p1b,p2b,M1,M2,r1,r2) THM 101025
% find
%   time of collision, t (if imag(t)~=0 then no collision)
%   centres of each molecule at collision, column vectors c1, c2
% given
%   centres of each molecule at time 0, column vectors c1b, c2b
%   masses and momenta of each molecule, M1, M2, column vectors p1b, p2b
%   radii of each molecule, r1, r2
% uses x = quadratic(a,b,c)
function [t,c1,c2] = moleculeCollWhen(c1b,c2b,p1b,p2b,M1,M2,r1,r2)
 m1 = M1*10.36; % convert from amu to merts (meV(ps/nm)^2)
 m2 = M2*10.36; % convert from amu to merts (meV(ps/nm)^2)
v1b = p1b/m1;           % find velocity of first molecule (nm/ps)
v2b = p2b/m2;           % find velocity of second molecule (nm/ps)
v = v1b - v2b;          % make second molecule stationary
c = c1b - c2b;          % centre second molecule at origin
r = r1 + r2;            % expand 2nd molecule to radius r, shrink 1st to point
if v'*v==0, t = inf; c1 = c1b; c2 = c2b; return, end % if       % parallel vels
t = min(quadratic(v'*v,2*c'*v,c'*c - r^2));            % display test only
c1 = c + v1b*t + c2b;   % move 1st molecule, then transform 2nd away from origin
c2 = v2b*t + c2b;       % transform 2nd away from origin
if t<-eps*10^3 | abs(imag(t))>eps       % flag coincident or no collision
  t = inf;
end % if t<0 |
```

The simulation must start with an initial set of molecules. I've done this with two random number generators, based on the MATLAB `rand(1)`, which returns a single `(1)` pseudorandom number bertween 0 and 1.

The initial positions of the molecules are distributed normally on a line which lies at an angle, itself distributed uniformly between some specified minimum angle `posLo` and maximum angle `posHi`. The normal distribution has a specified standard deviation `stdDev` in nm.

The initial momenta are distributed with a direction given by a second angle distributed uniformly between different specified minimum angle `momLo` and maximum angle `momHi`.

All these parameters are passed using the array

`startRange = [posLo,posHi,stdDev,momLo,momHi]`

for instance, with angles in degrees

`startRange = [0,0,5,40,50]`

Only the range of the uniform distribution of momenta is not specified as a parameter in `startRange`. I have derived momentum from energy, E: $p = \sqrt{2EM}$ and given energy the uniform distribution centered on the specified temperature T (in degrees Kelvin[1]) with a variation within $\pm\sqrt{2}/10 = \pm 0.14$ of this.

The result of the initialization is `molData`, with N rows (one for each molecule) and columns giving centre coordinates, momentum components, molecular radius and mass, and the generated energy.

Here is the code.

```
% function molData = moleculeInit(N,T,startRange,r,M)          THM     101026
% given number of molecules N, temperature T, molecular radius r (nm) and
%  mass M (AMU);
```

[1]Kelvin degrees convert to milli-electron Volts, meV, by the Boltzmann factor $k_B = 0.0862$ meV/degK. Unfortunately, there is a circularity in this way of presenting gas thermostatics because I have anticipated the connection of energy with temperature. I feel it best to be realistic, and so I have put in these numbers, just as I have used real masses and radii for the noble gases. The numbers, however, are not central at this stage of the discussion and you can pass them by for now.

```
%     NB 40 degC == 313.15 degK == 27 meV (26.9935 using k = .0862)
%  startRange is array (posLo, posHi, stdev, momLo, momHi) for the
%  initial positions of the molecules (angles in degrees, stdev in nm)
% find N*7 array molData, columns cx,cy,px,py,r,M,E
% uses genDirection(range)      (uniformly distributed angular direction)
%      genNormal(stdev) (Normally distributed position on line, given stdev)
function molData = moleculeInit(N,T,startRange,r,M)
startDirRange = [startRange(1),startRange(2)]*pi/180;
stdev = startRange(3);
startLoc = [startRange(4),startRange(5)];
momDirRange = [startRange(6),startRange(7)]*pi/180;
%for k = 1:N
k = 1;
% duplicates = [];                      % track duplicates: test only
while true              % loop over N molecules, but pass over any duplicates
  dirStart = genDirection(startDirRange);
  c = genNormal(stdev)*[cos(dirStart),sin(dirStart)] + startLoc;
  dirMom = genDirection(momDirRange);                % changed 100317
  % uniform momenta (changed 100602,100606):
  E = (1 + (2*rand(1)-1)*sqrt(2)/10)*0.0862*T;  % avg +/- (-1 -- 1)rt2/10
  p = sqrt(2*10.36*M*E)*[cos(dirMom),sin(dirMom)];
  moldata = [c(1),c(2),p(1),p(2),r,M,E];
  duplicate = false;
  for j = 1:k-1                 % a very slow way of checking for duplicates
    if min(moldata==molData(j,:))             % min = and for boolean array
%      duplicates = [duplicates;[k,j,moldata]];% track duplicates: test only
      duplicate = true;
      break                                 % full duplicate found
        % NB it doesn't matter if partial duplicate, e.g., positions coincide
    end % if min()
  end % for j
  if ~duplicate | k==1
      molData(k,:) = moldata;               % record unique initial molecule
      k = k + 1;
  end % if ~duplicate
  if k>N, break, end %if
end % while true (aka  for k)
```

Backing it up are self-explanatory routines

```
% function a = genDirection(range)            THM                    100301
% generate random direction in range(1)--range(2)
function a = genDirection(range)
lowangle = range(1);
anglerange = range(2) - lowangle;
a = lowangle + anglerange*rand(1);


% function d = genNormal(stdev)               THM                    100301
% given standerd deviation stdev
% find a distance d from centre
% uses erfNormal(stdev,decpl)  to give cumulative sum of Normal
function d = genNormal(stdev)
% Following [Ross] to generate random variable X from uniform u:
%       generate uniform random u on [0,1];
%       p = erfNormal^{-1}(u)
if stdev == 0, d = 0; else
  u = rand(1);                  % uniform random
  for d = 0:0.03*stdev:3*stdev
    if erfNormal(d,2) > u, break, end %if
  end % for d
  s = rand(1);
  if s<=0.5, signd = -1; else signd = 1; end
  d = signd*d;
end % if stdev
```

(Well, `genNormal` may require a little work and background. And backing it up is `erfNormal` which we discussed in Note 6.
The citation to [Ross] is [Ros02, p.263].)

We now have five building blocks, `moleculeInit()`, `mollCollWallWhen()`, `moleculeCollWhen()`, `molCollWallCountBot()` and `moleculeCollide()`. To incorporate these into the collision simulator we must

- initialize with `moleculeInit()`
- incorporate the others in a loop sequencing through one collision after another, as follows
  - for each molecule (label it `v1`)
    - use `mollCollWallWhen()` to find when it next hits a wall
    - for each other molecule (call it `v2`)
      use `moleculeCollWhen()` to find when `v1` next hits `v2`
  - find the closest of all such times, the associated molecule(s) and position(s) and call this data `next`
  - execute the appropriate collision, using `molCollWallCountBot()` if `v1==v2` (I duplicated `v1` in `next` if the next collision is with a wall) or `moleculeCollide()` otherwise.
  - recalculate the positions of all noncolliding molecules to account for the time lapse represented in `next`, so as to be ready with a complete set of positions and momenta for another iteration of loop 2.

Here is the code for simulating collisions forever. Well, I've put in a lookaround after multiples of 100 intermolecular collisions to ask if we want to continue.

```
% function molData = collSim(N,T,r,M,ab) THM 101026
% N molecules each of radius r (nm), mass M (AMU) at temperature T (degK)
% Helium 0.049nm (1), 4AMU; Neon 0.051 m (1), 20AMU; Argon 0.088nm (1.8), 40AMU)
% find; molecules in box [-ab(1),ab(1)]x[-ab(2),ab(2)] nanometers
% (initially symmetrical, stored abcd [abcd(1),abcd(2)]x[abcd(3),abcd(4)])
% Return nothing (collision simulation only, no stats); molData is sop;see below
% uses molData = moleculeInit(N,T,startRange,r,M)
% [delTime,centre] = molCollWallWhen(cb,p,M,r,ab);
% [delTime,cntr1,cntr2] = moleculeCollWhen(c1b,c2b,p1b,p2b,M,M,r,r);
% p1= molCollWallCountBot(p1b,1,1,abcd,r1,c1,0);
% [p1,p2] = moleculeCollide(M,M,p1b,p2b,0,C12);
function molData = collSim(N,T,r,M,ab)
tic % elapsed time

% initialize
startRange = [0,0,5,0,0,40,50];
%       l h s x y  l h s is stdev in nm of spread along line of angle
%       ang   cntr from ang_l to ang_h centred at x,y
%      ----pos---  mom mom_l to mom_h is angle of motion direction
% for box -20:20*-10:10 try exactly 45 degrees from exactly (0,0)
% generates molData[N * cx,cy,px,py,r,M,E]
%  1  2  3  4 5 6 7
% L: nm   T: ps  M: meV(ps/nm)^2   L/T: nm/ps   ML/T: meV(ps/nm)
```

```
molData = moleculeInit(N,T,startRange,r,M) % display test only
oldMolData = molData;
sizeMolData = size(molData);
noVals = sizeMolData(1);
abcd(1) = -ab(1); abcd(2) = ab(1); abcd(3) = -ab(2); abcd(4) = ab(2);
happIndx = 0; % keep track of number of steps
time = 0.0; % keep track of time
stepChunk = 100; % iterate this many steps then check if more wanted
firstChunk = true; % to detect first <stepChunk> collisions at resume time
%stepChunk = 1; % test only
chunkNo = 0;

% Loop starts here
while(true)
  collision = false;
  % find time of next collision
  record = []; % record of collision times, results
  for v1 = 1:noVals
    c1b = [molData(v1,1); molData(v1,2)];
    p1b = [molData(v1,3); molData(v1,4)];
    M1 = molData(v1,6);
    r1 = molData(v1,5);
    [delTime,centre] = molCollWallWhen(c1b,p1b,M1,r1,abcd);  % display test only
    record = [record;[delTime,v1,v1,centre(1),centre(2),centre(1),centre(2)]];
    for v2 = v1+1:noVals
      c2b = [molData(v2,1); molData(v2,2)];
      p2b = [molData(v2,3); molData(v2,4)];
      M2 = molData(v2,6);
      r2 = molData(v2,5);
      [delTime,cntr1,cntr2] = moleculeCollWhen(c1b,c2b,p1b,p2b,M1,M2,r1,r2);
      record = [record;[delTime,v1,v2,cntr1(1),cntr1(2),cntr2(1),cntr2(2)]];
    end % for v2
  end % for v1

  %sqrt choose and execute next collision
  next = minrows(record); % smallest delTime wins  display test only
  time = time + next(1);
  v1 = next(2);
  v2 = next(3);
  c1 = [next(4);next(5)];
  c2 = [next(6);next(7)];
  deltaX1 = c1(1) - molData(v1,1);
  deltaY1 = c1(2) - molData(v1,2);
  deltaX2 = c2(1) - molData(v2,1);
  deltaY2 = c2(2) - molData(v2,2);
  r1 = molData(v1,5);
  r2 = molData(v2,5);
  M1 = molData(v1,6);
  M2 = molData(v2,6);
  p1b = [molData(v1,3); molData(v1,4)];
  p2b = [molData(v2,3); molData(v2,4)];
  if v1==v2 % collision is with wall
    p1 = molCollWallCountBot(p1b,1,1,abcd,r1,c1,0);
    deltaPx1 = p1(1) - p1b(1); % change of momentum at wall
    deltaPy1 = p1(2) - p1b(2);
    % molData for v1 is common to both these branches: done afterwards
  else % v1~=v2
    collision = true; % test only: used to update afterColl
    C12 = c2 - c1;
    [p1,p2] = moleculeCollide(M1,M2,p1b,p2b,0,C12); % display test only
    molData(v2,1) = c2(1);
    molData(v2,2) = c2(2);
    molData(v2,3) = p2(1);
    molData(v2,4) = p2(2);
    molData(v2,7) = (p2(1).^2 + p2(2).^2)/(2*M2*10.36); % energy
  end % if v1==v2
  molData(v1,1) = c1(1);
  molData(v1,2) = c1(2);
  molData(v1,3) = p1(1);
  molData(v1,4) = p1(2);
  molData(v1,7) = (p1(1).^2 + p1(2).^2)/(2*M1*10.36); % energy
  for v = 1:noVals % update all other molecules' positions
    if v==v1 | v==v2, continue, end % if
    deltaX = molData(v,3)/(molData(v,6)*10.36)*next(1);
    deltaY = molData(v,4)/(molData(v,6)*10.36)*next(1);
    molData(v,1) = molData(v,1) + deltaX; % update position
    molData(v,2) = molData(v,2) + deltaY;
  end % for v
  if collision
    happIndx = happIndx + 1;
    if mod(happIndx,stepChunk)==0
      elapsed = toc; % elapsed time
      chunkNo = chunkNo + 1;
      if firstChunk
firstChunk = false;
      end % if firstChunk
      beep
      reply = input([num2str(elapsed) ' elapsed, ' num2str(time)...
 ' ps.: another ' num2str(stepChunk) ' collisions? y/n '],'s');
      if isempty(reply)
          reply = 'y';
      end
      if reply~='y' % not 'y'
        break
      else tic % elapsed time
      end % if reply
    end % if mod()
  end % if collision
end % while(true)
```

13. A gas simulation 2: statistics. We wrote in the previous Note a simulation of molecules colliding in two dimensions but we did not do anything about looking inside to see what is happening. We could animate the molecules and show them bouncing around, the way Blauch [Bla09] does.

But, for one thing, I have not written the simulator to make this easy because it does not step in equal increments of time but rather looks ahead to the next collision(molecule-molecule or molecule-wall) and abruptly advances the clock to that time.

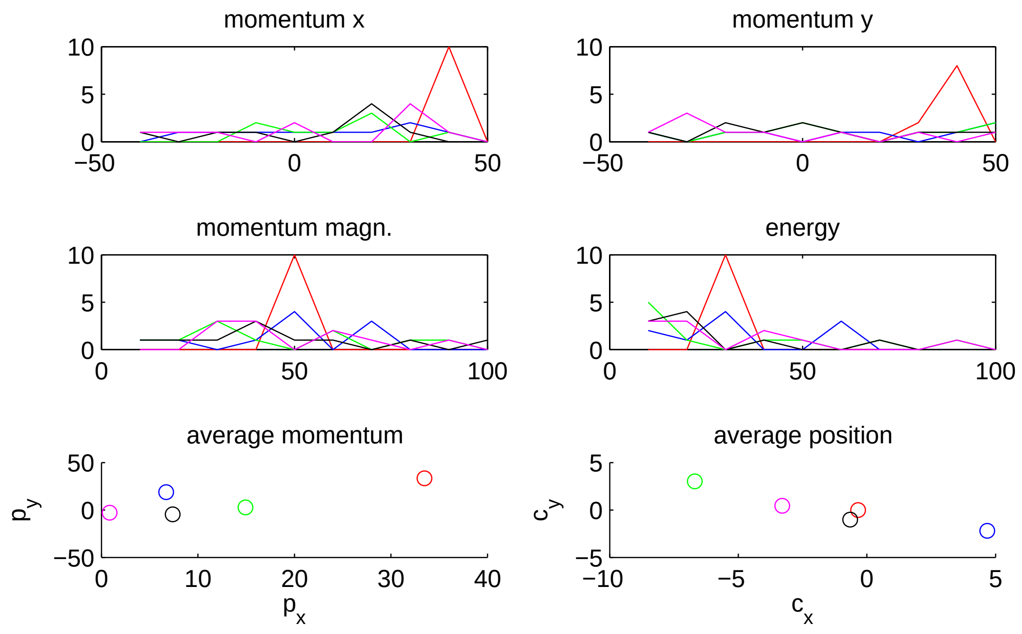
And for anothr thing, such an animation is much less helpful in advising our thinking than looking at the right *statistics*. The Excursion *First simulation* makes a start. Let's now incorporate this statistics-making and -plotting within the simulation.

First here is the function that gathers and prints the statistics.

```
% function molDatAverage = molDataHisto(molData,molDatAverage) THM 101028
% plot histogram of statistics from statSim()
% given
%       molData, N*7 array of [cx,cy,px,py,4,M,E] to which we'll add p
%       molDatAverage, initially [], average of all these
% find
%       molDatAverage = [molDatAverage;molDatAvg]
function molDatAverage = molDataHisto(molData,molDatAverage)
molData(:,8) = sqrt(molData(:,3).^2 + molData(:,4).^2); % momentum magnitude
sizeMolData = size(molData);
% firstInvocation = size(molDatAverage)==[0,0];% initially [0,0] (or invokNo==1)
sizeMolDataAvg = size(molDatAverage);
invokNo = sizeMolDataAvg(1) + 1;
firstInvocation = invokNo==1;                   % initially
molDatAvg = sum(molData)/sizeMolData(1)
molDatAverage = [molDatAverage;molDatAvg];
%molDataStdev = sqrt(sum(molData.^2-molDatAverage.^2)/sizeMolData(1))
molDatBucket = zeros(10,5);
bw = 10;                % bucket width
for k = 1:4                            % momentum (1,2), energy (3), |p| (4)
  k1 = condif(k<3,k+2,k+4);            % 1->3 (px), 2->4 (py), 3->7 (E), 4->8
  bs = condif(k<3,-5*bw,0);            % bucket start
  for j = 1: sizeMolData(1) % this could be coded better: now?
    if molData(j,k1) < bs + bw, molDatBucket(1,k) = molDatBucket(1,k) + 1;
    elseif molData(j,k1) < bs + 2*bw, molDatBucket(2,k) = molDatBucket(2,k) + 1;
    elseif molData(j,k1) < bs + 3*bw, molDatBucket(3,k) = molDatBucket(3,k) + 1;
    elseif molData(j,k1) < bs + 4*bw, molDatBucket(4,k) = molDatBucket(4,k) + 1;
    elseif molData(j,k1) < bs + 5*bw, molDatBucket(5,k) = molDatBucket(5,k) + 1;
    elseif molData(j,k1) < bs + 6*bw, molDatBucket(6,k) = molDatBucket(6,k) + 1;
    elseif molData(j,k1) < bs + 7*bw, molDatBucket(7,k) = molDatBucket(7,k) + 1;
    elseif molData(j,k1) < bs + 8*bw, molDatBucket(8,k) = molDatBucket(8,k) + 1;
    elseif molData(j,k1) < bs + 9*bw, molDatBucket(9,k) = molDatBucket(9,k) + 1;
    elseif molData(j,k1) < bs + 10*bw, molDatBucket(10,k) = molDatBucket(10,k) + 1;
    else % ignore energentum over 100
    end % if molData
  end % for j
end % for k
colour = 'rbgkm';
invokNo,clr = colour(mod(invokNo-1,5)+1);       % display test only
j=1:10;
subplot(3,2,1)
if firstInvocation, hold off, end
plot(bw*(j-5),molDatBucket(:,1),clr)
title('momentum x')
hold on
subplot(3,2,2)
if firstInvocation, hold off, end
plot(bw*(j-5),molDatBucket(:,2),clr)
title('momentum y')
hold on
subplot(3,2,3)
if firstInvocation, hold off, end
plot(bw*j,molDatBucket(:,4),clr)
title('momentum magn.')
hold on
subplot(3,2,4)
if firstInvocation, hold off, end
plot(bw*j,molDatBucket(:,3),clr)
title('energy')
hold on
subplot(3,2,5)
if firstInvocation, hold off, end
scatter(molDatAvg(3),molDatAvg(4),clr)
xlabel('p_x')
ylabel('p_y')
```

```
title('average momentum')
hold on
subplot(3,2,6)
if firstInvocation, hold off, end
scatter(molDatAvg(1),molDatAvg(2),clr)
xlabel('c_x')
ylabel('c_y')
title('average position')
hold on
```

To explain it, let's look at the figure it generates.



These plots are made in the last 37 lines of the code: the first time `molDataHisto()` is invoked, it draws the red plots. The second time it draws blue, the third time green, then black then finally magenta. (It will cycle through these colours repeatedly, but we don't need that ability.)

The six plots it makes are (1, 2) the histogram of x- and y-momenta, (4) the histogram of energies, and (5, 6) scatter plots of average momenta for all 10 molecules, x vs. y, and of average position for all 10 molecules, x vs. y, respectively. And, because it might be more useful than the x- and y-components of the momenta, I added (3) the histogram of the magnitudes of the momenta, $\sqrt{p_x^2 + p_y^2}$.

Plots 1–4 are histograms, so really should be bar charts, but to show all the colours I made them with `plot()`.

The histograms are derived from `molDatBucket`, which in the 18 lines of the nested `for` loop is filled in as follows. For the energy and the magnitude of the momentum, 10 buckets of width 10 from 0 to 100 count the numbers of molecules that fall in each bucket range. For the x and y components of the momenta, 10 buckets of width 10 from -50 to 50 do the same.

The scatter plots are derived from `molDatAvg` which is the average of the input, `molData`. Only the x and y components of the momenta and of the position are used, although `molDatAvg` finds the averages of all eight values in `molData` (augmented by the momenum magnitudes). Also `molDatAverage` is assembled from `molDatAvg` in previous invocations of the `molDatHisto()` function: this is output but not otherwise used, except to find out which invocation `molDatHisto()` is at, from the number of rows in `molDatAverage`.

To incorporate invocations of function `molDatHisto()` in the simulation is very straightforward. I put it into the code in two places. First

```
molDatAverage = molDatHisto(molData,[])
```

appears right after the first `molData` is generated by `moleculeInit()`. Second

```
molDatAverage = molDatHisto(molData,molDatAverage)
```

appears in the main loop in the part I wrote to execute whenever the numbers of intermolecular collisions is a multiple of 100.

Now let's look more closely at the results of running `statSim(10,300,0.05,4,[20;10])` where the modified simulation `statSim()` has the same parameters as `CollSim()` of the previous Note. First, all the red histograms are quite different from the rest. They show the initial squirt of 10 molecules at about 45 degrees "north-east".

Second, the other colours (blue-green-black-magenta) in each histogram seem to resemble each other somewhat. Apart from what seem to be echoes of the initial peaks, the x- and y-momenta seem to be sort of uniformly distributed but there isn't enough data to tell. But there may be different patterns in the other colours for the momentum magnitude and for the energy.

There seem to be more energies near zero (to the left) than with higher values (green-black-magenta). And there seem to be peaks at low momentum magnitude, but between zero and the middle 50—maybe around 40—and below that tapers to no occurrences of momentum magnitude (green-black-magenta). We'll explore these patterns further next.

(The scatterplots don't show much except for maybe a tendency towards (0,0) in both cases (magenta for momenta, black for energies). But these are snapshots, taken every 100 collisions, so may never tell us much.)

To refine these results we need more molecules. But `moleculeCollWhen` runs in time proportional to the square of the number of molecules, so simulating 1000 molecules may cost us up to 100 times more than simulating just 10. So instead of changing N from 10 to more molecules, we rewrite `statSim()` to `cumuSim()` which *accumulates* statistics on momentum magnitude and energy, thus smoothing out the distributions and averages.

We need a new histogram- and average-making function

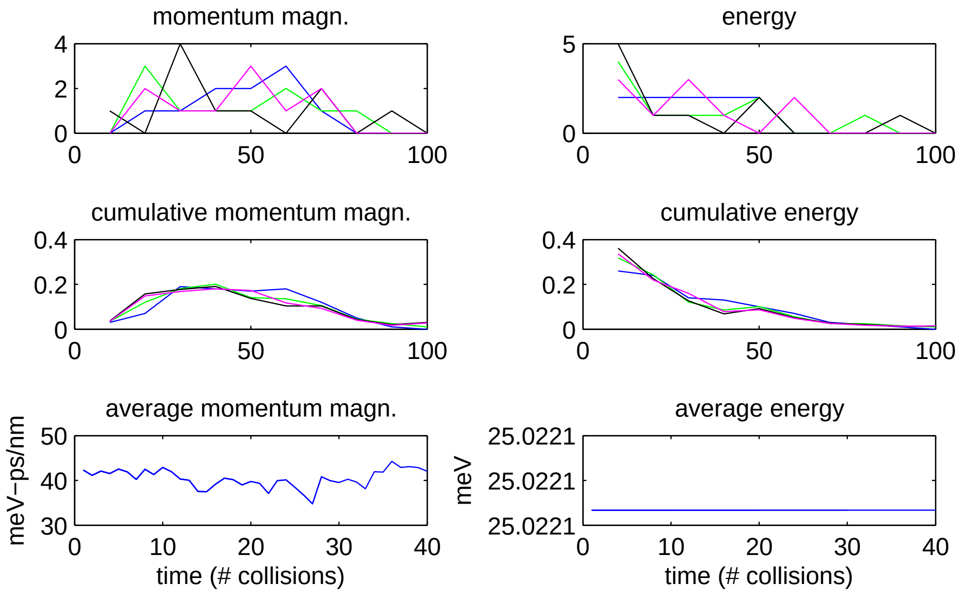
```
[momenCum,momenAvg] = momenCumHisto(molData,momenCum,momenAvg)
```

which accumulates columns 8 and 9 of `molData` into the histograms for momentum magnitude and energy, respectively, in `momenCum`. It also records the averages of these two quantities in the array `momenAvg` which increases steadily in size the same way `molDatAverage` grew with repeated invocations of `molDataHisto()` (above, in this Note).

We must also modify `molDataHisto()` to `molDataDistr()` with two more inputs, `momenCum` and `momenAvg`: this now plots (1 and 2) the same snapshots of momentum magnitude and energy, plus (3 and 4) two new plots of *distributions* made from the new input `momenCum`. (I normalize the histograms to distributions because the histograms increase in size—every 10 collisions in fact—and I want to show them all the same size for comparison.)

Finally `molDataDistr()` provides (5 and 6) two more plots, from the new input `momenAvg`, of the average momentum magnitude and energies as *timeseries*.

We'll look at the results first, before saying how `cumuSim()` differs from `statSim()`.



(Note that there are no red lines: to let the gas settle down, I've let the simulation run through 100 intermolecular collisions before starting to accumulate statistics. This avoids distortion from the initial momenta and energies. The snapshts look somewhat like those from `statSim()`—they differ in detail because `moleculeinit()` has generated a whole new set of starting conditions due to its use of `rand(1)`. I'll discuss the new plots shortly.)

I've modified `statSim()` to `cumuSim()` in the following ways. First, instead of the initial invocation of `molDataHisto()` I just set `molDatAverage = []`. Elsewhere I replace `molDataHisto()` with `molDataDistr()`.

Second, just before the code that detects if each chunk (of 100 intermelecular collisions) is finished, I use `firstChunk` to avoid collecting statistics for the first chunk.

Third, in this same place I write a test for smaller chunks, of size N ($= 10$), and invoke `momenCumHisto()` to accumulate momentum and energy statistics every 10 intermolecular collisions. (Warning: N is a factor of `chunkSize` (10 and 100 respectively) in this use of `cumuSim()`: it may not work well if `chunkSize` is not a multiple of 10.)

In the last four plots in the results we notice the following things. First, the accumulted momentum and energy distributions (3 and 4) clearly settle down to fixed shapes as more and more collisions are accumuated (in the sequence blue-green-black-magenta).

The energies tend to a familiar distribution: it appears to decline exponentially with energy. So the probabilities could possibly be written something like

$$p_E = \frac{e^{-E\beta}}{Z}$$

as at the end of Note 7. This distribution is called the *Boltzmann* distribution.

The momenta settle into a different distribution, called the *Maxwell* distrinution. It rises in the middle and decreases at both ends. But it is not the normal distribution because it cannot be symmetrical: only non-negative values are possible for the momentum magnitude.

Scond, the timeseries plots (5 and 6) of average momentum and energy show two characteristic features.

The average energy is completely constant in time, consistent with the law of conservation of energy.

In the simulations, `moleculeCollide()` ensures that energy is conserved in each collision, so of course the total energy of all the molecules, and hence the average energy of each, remains constant.

The average momentum magnitude, on the other hand, *fluctuates*: it varies about an apparently fixed value. Since `moleculeCollide()` also conserves momenta at each intermolecular collision, why is the average momentum not constant, like the averge energy?

There are two considerations. First, momenta are not conserved, but rather reversed, in collisions with the *walls*. However we did discuss the averaging out of these reversals by collisions of different molecules with opposite walls.

The second consideration is more significant: it is the *individual components* of momentum that are conserved by `moleculeCollide()`, not the *magnitude*. So we have no reason to say that average momentum magnitude is constant.

14. The Boltzmann and Maxwell distributions. If we suppose that energies *do* follow the Boltzmann distribution

$$B(E) = \frac{1}{Z}e^{-\beta E}$$

can we figure out the form of the Maxwell distribution for the momentum magnitudes?

As in Note 7, the partition factor, Z, is the normalizing factor so that $B(E)$ sums to 1. Quantum energy levels are discrete but their separations are so small (and zero in the pre-quantum approximation) that we can switch from sums to antislopes as a mathematically easier approximation.

We also need to work with momenta, because the possible kinetic energies are due to momenta

$$E = \frac{p^2}{2m}$$

where p is the momentum magnitude, $p^2 = p_x^2 + p_y^2$, and m is the molecular mass, fortunately the same for every molecule in the present simulation.

So Z will be the sum of all possible $e^{-\beta p^2/(2m)}$, i.e., the antislope of this with respect to p_x and p_y.

$$\begin{aligned}
Z &= \text{antislope}_{p_x}\text{antislope}_{p_y}e^{-\beta p^2/(2m)} \\
&= \text{antislope}_{p_x}\text{antislope}_{p_y}e^{-\beta(p_x^2+p_y^2)/(2m)} \\
&= (\text{antislope}_{p_x}e^{-\beta p_x^2/(2m)})(\text{antislope}_{p_y}e^{-\beta p_y^2/(2m)}) \\
&= (\text{antislope}_{p_q}e^{-\beta p_q^2/(2m)})^2 \\
&= 2\pi m/\beta
\end{aligned}$$

where the last line is just I from Note 6 with the parameter a there being $2m/\beta$ here (and p_q in the line before is "p_{whatever}", i.e., p_x or p_y).

In going from sums to antislopes we have gone from probability distribution to probability density. So we must express the probability in terms of a small increment of momentum magnitude, Δp. Since we used 2-dimensional antislopes for Z, over p_x and p_y, to go to the 1-dimensional momentum magnitude p we sum over all values of the other polar coordinate to get a factor $2\pi p$ (also as in Note 6). So the probability density

$$\begin{aligned}
M_{2\text{D}}(p)\Delta p &= \frac{1}{Z}e^{-\beta p^2/(2m)}2\pi p\Delta p \\
&= \frac{2\pi}{2\pi m/\beta}pe^{-\beta p^2/(2m)}\Delta p \\
&= \frac{\beta}{m}p\Delta pe^{-\beta p^2/(2m)}
\end{aligned}$$

The factor of p in the result forces the probability to go to zero as p goes to zero, which is the significant difference we observed in the results of the simulation.

This is the 2-dimensional variant of the Maxwell distribution of momentum magnitudes (or velocity magnitudes). The true Maxwell distribution is 3-dimensional. You can find it in the Excursions.

Let's see if the Boltzmann and Maxwell (2D) distributions match the distributions of simulated energies and momenta, respectively. There is one parameter to determine: what is β?

We must explore a little further. We have two new distributions: what are their moments? (Don't confuse "moment" with "momentum"!) Their modes?

First, Boltzmann: its mean is its first moment.

$$\mu_1 = \text{antislope}_E \frac{E}{Z} e^{-\beta E} \mid_0^\infty$$

We'll integrate by parts (Note 6) with

$$\begin{array}{rcl} u = & E & \text{slope}_E u = 1 \\ v = & -\frac{1}{\beta Z} e^{-\beta E} & \text{slope}_E v = \dfrac{1}{Z} e^{-\beta E} \end{array}$$

Since

$$\text{slope}_E uv = u\text{slope}_E v + v\text{slope}_E u$$

so

$$uv = \text{antislope}_E(u\,\text{slope}_E v) + \text{antislope}_E(v\,\text{slope}_E u)$$

and

$$\begin{array}{rcl} \mu_1 & = & \text{antislope}_E \dfrac{E}{Z} e^{-\beta E} \mid_0^\infty \\ & = & \text{antislope}_E(u\,\text{slope}_E v) \mid_0^\infty \\ & = & -\dfrac{E}{\beta Z} e^{-\beta E} \mid_0^\infty + \text{antislope}_E(\dfrac{1}{\beta Z} e^{-\beta E} \mid_0^\infty) \\ & = & uv \mid_0^\infty - \text{antislope}_E(v\,\text{slope}_E u) \mid_0^\infty \\ & = & 0 - 0 + \dfrac{1}{\beta Z} Z \\ & = & \dfrac{1}{\beta} \end{array}$$

since $Z = \text{antislope}_E e^{-\beta E} \mid_0^\infty$ by definition.

This tells us immediately that β is the reciprocal of the mean energy μ_1, determined by the Boltzmann energy distribution.

But let's go on. The second moment

$$\mu_3 = \text{antislope}_E \frac{E^2}{Z} e^{-\beta E} \mid_0^\infty$$

Integrating by parts with

$$\begin{array}{rcl} u = & E^2 & \text{slope}_E u = 2E \\ v = & -\frac{1}{\beta Z} e^{-\beta E} & \text{slope}_E v = \dfrac{1}{Z} e^{-\beta E} \end{array}$$

$$\begin{aligned}
\mu_2 &= \text{antislope}_E(u\,\text{slope}_E v)\ |_0^\infty \\
&= uv\ |_0^\infty - \text{antislope}_E(v\,\text{slope}_E u)\ |_0^\infty \\
&= -\frac{E^2}{\beta Z}e^{-\beta E}\ |_0^\infty + \text{antislope}_E(\frac{2E}{\beta Z}e^{-\beta E}\ |_0^\infty) \\
&= 0 - 0 + \frac{2}{\beta}\mu_1 \\
&= \frac{2}{\beta^2}
\end{aligned}$$

So the variance is

$$\sigma^2 = \mu_2 - \mu_1^2 = \frac{1}{\beta^2}$$

and the standard deviation $\sigma = 1/\beta = m\mu_1$.

Second, Maxwell (2D). Integration by parts with

$$\begin{aligned}
u &= p & \text{slope}_p u &= 1 \\
v &= -e^{-p^2/a} & \text{slope}_p v &= \frac{2p}{a}e^{-p^2/a}
\end{aligned}$$

where a abbreviates $2m/\beta$ for simplicity, so

$$\begin{aligned}
\mu_1 &= \text{antislope}_p \frac{2}{a}p^2 e^{-p^2/a}\ |_0^\infty \\
&= \text{antislope}_p(u\,\text{slope}_p v)\ |_0^\infty \\
&= uv\ |_0^\infty - \text{antislope}_p(v\,\text{slope}_p u)\ |_0^\infty \\
&= -pe^{-p^2/a}\ |_0^\infty + \text{antislope}_p e^{-p^2/a}\ |_0^\infty \\
&= 0 - 0 + \frac{\sqrt{\pi a}}{2} \\
&= \frac{\sqrt{\pi a}}{2} \\
&= \sqrt{\frac{\pi m}{2\beta}}
\end{aligned}$$

Where I have used the result $I = \sqrt{\pi a}$ from Note 6 and cut it in half because that range was from $-\infty$ to ∞ while this range is from 0 to ∞.

Similarly, using

$$\begin{aligned}
u &= p^2 & \text{slope}_p u &= 2p \\
v &= -e^{-p^2/a} & \text{slope}_p v &= \frac{2p}{a}e^{-p^2/a}
\end{aligned}$$

we integrate by parts to get the second moment

$$\begin{aligned}
\mu_2 &= \text{antislope}_p \frac{2}{a}p^3 e^{-p^2/a}\ |_0^\infty \\
&= \text{antislope}_p(u\,\text{slope}_p v)\ |_0^\infty \\
&= uv\ |_0^\infty - \text{antislope}_p(v\,\text{slope}_p u)\ |_0^\infty \\
&= -p^2 e^{-p^2/a}\ |_0^\infty + \text{antislope}_p 2pe^{-p^2/a}\ |_0^\infty
\end{aligned}$$

$$\begin{aligned} &= 0 - 0 - ae^{-p^2/a}\,|_0^\infty \\ &= a \\ &= \frac{2m}{\beta} \end{aligned}$$

because slope$(-ae^{-p^2/a}) = 2pe^{-p^2/a}$, and because $e^{-p^2/a}$ is zero at $p = \infty$ and 1 at $p = 0$.

The variance is $\mu_2 - \mu_1^2 = a(1 - \pi/4)$ and the standard deviation is the square root of this.

Finally, we can find the mode. This is where the Maxwell (2D) distribution goes horizontal at its peak so its slope is zero.

$$\begin{aligned} 0 &= \text{slope}_p \frac{p}{a} e^{-p^2/a} \\ &= \frac{1}{a} e^{-p^2/a} - \frac{2p}{a^2} e^{-p^2/a} \end{aligned}$$

so

$$0 = 1 - \frac{2p^2}{a}$$

i.e.,

$$p = \sqrt{\frac{a}{2}} = \sqrt{\frac{m}{\beta}}$$

is the mode—the most probable momentum.

Note that the mean is above the mode.

$$\mu_1 = \frac{\sqrt{\pi a}}{2} = \sqrt{\frac{\pi}{2}}\sqrt{\frac{a}{2}} > \sqrt{\frac{a}{2}}$$

These results are for the Maxwell distribution in two dimensions. The three-dimensional Maxwell distribution, which applies to real gases (as opposed to billiard balls) is different and given in an Excursion.

The mean of the Boltzmann distribution gave us the parameter β

$$\beta = \frac{1}{\mu_1}$$

where μ_1 is the mean energy. For our simulation we found $\mu_1 = 25.0$meV. So we can plug this into the Boltzmann and Maxwell disteributions to get the actual curves

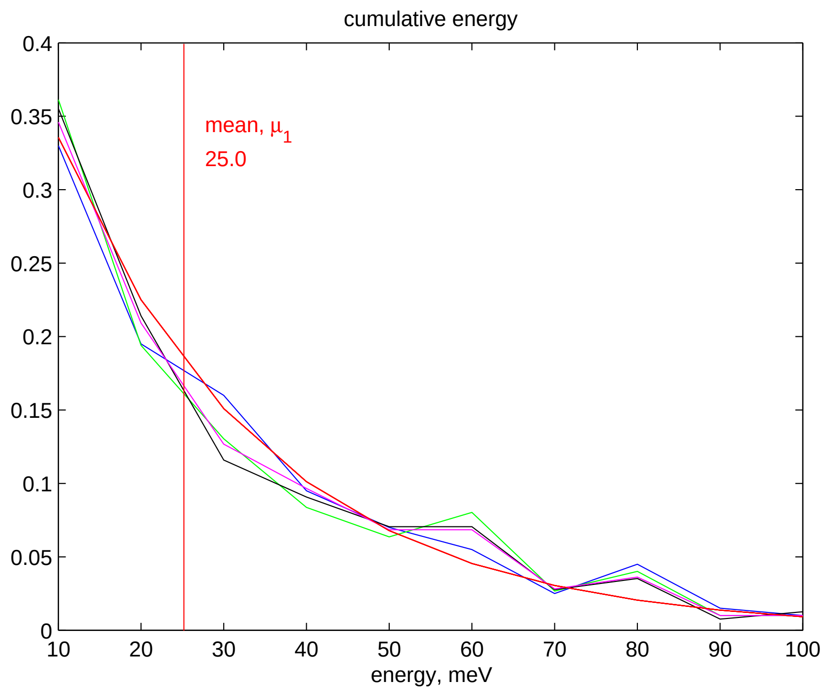
$$B(E) = \frac{1}{Z_B} e^{-\beta E}$$

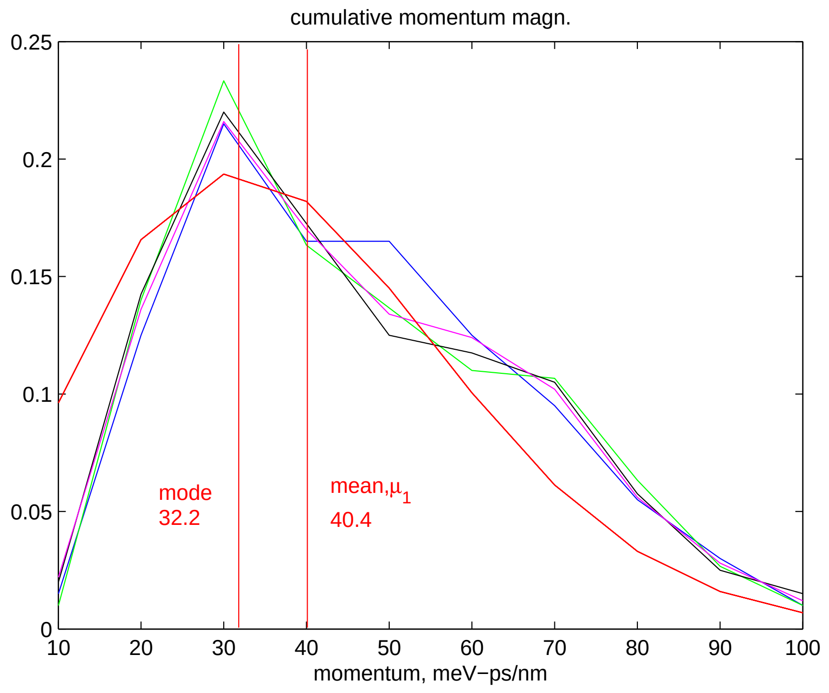
and

$$M_{2D}(p) = \frac{p}{Z_M} e^{-\beta p/(2m)}$$

where in a calculation we'll just use Z_B and Z_M as the normalizing denominators, i.e., the sums of all the unnormalized terms, to make $B(E)$ and $M_{2D}(p)$ sets of probabilities which sum to 1.

We didn't capture all the intermediate data in the simulation `cumuSim()`, so we rewrite it as `maxbSim()` to give the following results.





The red curves are the theoretical distributions we've just worked out. You can see that the distributions of the simulated 2D gas get closer to the red curves as we accumulate the data (blue to green to black to magenta).

The simulation `maxbSim()` differs from `cumuSim()` in its initialization and in its plot routine `molDataMaxBo()`.

The initializing function, `molecuReInit()`, replacing `moleculeInit()`, simply returns the `molData` array that `moleculeInit()` generated in our above invocation of `cumuSim()`.

The plot routine `molDataMaxBo()` replaces `molDataDistr()` and plots only the cumulative energy distribution and the cumulative momentum magnitude distribution. It plots them as separate figures so we can see them better than the single figure with six subplots. And it adds the plots of the distribution functions $B(E)$ and $M_{2D}(p)$ we've just worked out.

We can also compare average momentum magnitudes between theory

$$\sqrt{\frac{\pi m}{2\beta}} = 40.36 \text{ meV-ps/nm}$$

and simulation. The Note 13 `cumuSim()` recorded a sequence of forty average momentum magnitudes, one for every ten intermolecular collisions (these are what it plotted as the momentum timeseries), so we can accumulate these and find overall averages corresponding to each of the blue, green, black and magenta distributions. These are, respectively, 41.8, 40.7, 40.0 and 40.5 meV-ps/nm.

15. Fluctuations, variations and samples. We can learn more from the `cumuSim()` rseults of Note 13. The fifth plot showed fluctuations in the average momentum magnitude as time progressed.

Fluctuations are variations in a *timeseries*. Since they are also variations of the quantity plotted in the timeseries, they have statistical properties such as means, standard decviations, and, of course, histograms. We'll talk about these later.

First, I'd like to focus on the sequence aspect of fluctuations. (The sequence need not be a sequence in time, by the way, although this is the most common situation in which fluctuations are discussed, but it must be a sequence as opposed to a simple set of values.)

When I searched the McGill library catalogue for a book on fluctuations, I came across the title "Robertsonian Economcs: An Examination of the Work of Sir D. H. Robertson on Industrial Fluctuations" (by John R Presley, 1979). This book considers fluctuations as cycles or even waves. This suggests Fourier analyzing our timeseries to find what wave components it has.

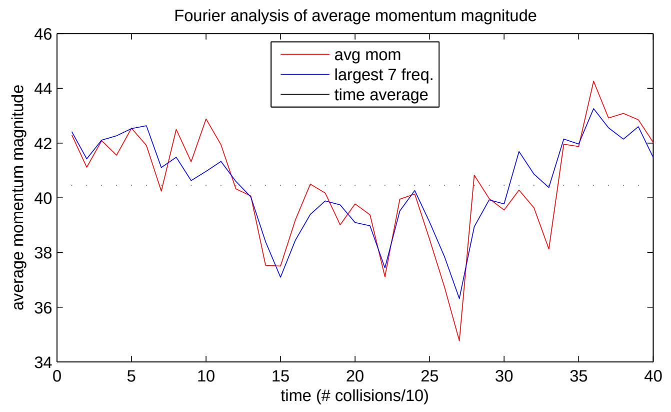
So I used `makeDFT(40)` from Week 9 Note 5 to extract the frequency components from the 40 momentum magnitude averages sequenced by `cumuSim()`.

The seven largest magnitudes of these (2-number) components were

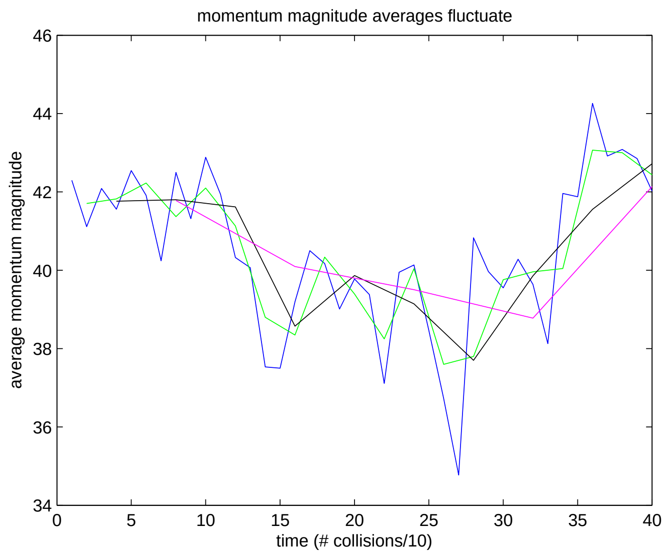
component	magnitude	$\div\sqrt{40}$
1	255.89	40.4598
2	6.42	1.0151
7	2.82	0.4459
4	2.28	0.3605
8	2.01	0.3178
17	1.74	0.2751
5	1.71	0.2704

I've put in a third column the second column divided by $\sqrt{40}$: the first component is just the overall average of the momentum magnitudes, and the others are scaled commensurately.

Setting all but these seven to zero and Fourier-transforming back, the way we did with the *leftBlack* image in Week 9 Note 5, we can capture most of the fluctuations.



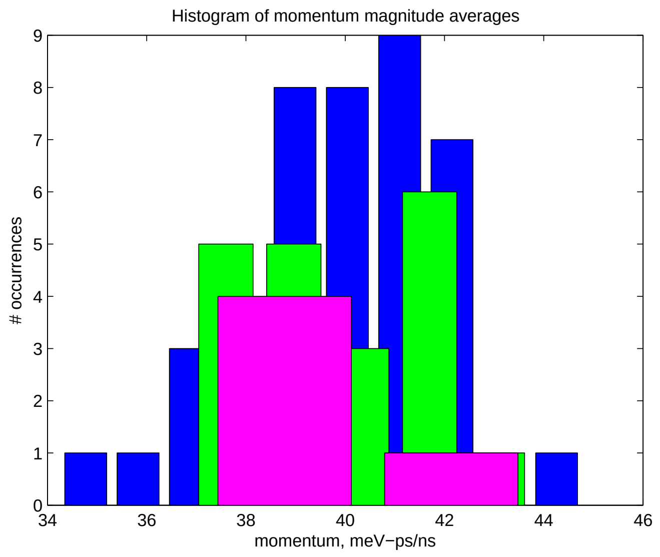
We can reduce the fluctuations by increasing the sample size. The averages shown so far were taken after 10 intermolecular collisions, and `cumuSim()` generated 40 of them (blue in the figure below). We can group these into 20 averages of 20 collisions (green), 10 averages of 40 collisions (black), 5 averages of 80 collisions (magenta).



(The program is straightforward which takes the 40 averages as produced by `cumuSim()` and groups them into 2s, 4s and 8s, and builds up the cumulative plots above.)

We see that increasing the sample size reduces the variability. We can see this from a statistical point of view with histograms and moments, too.

If we turn those 40 momentum magnitude averages into a histogram we can get a distribution of averages (blue in the figure below). Moreover we can aggregate in 2s and get another histogram (green). And we can skip to groups of 8 and get a third histogram (magenta).



Notice that the histograms get narrower as we aggregate more. We can confirm this by reporting the standard deviations.

grouped by	averge	variance	std.dev.
1s	40.4590	4.0	2.0
2s	40.4590	2.9	1.7
8s	40.4590	1.7	1.3

The Fourier analysis gives almost exactly the same numbers for variability as the standard deviation, using the following reasoning.

The first Fourier component, corresponding to zero frequency, is always the mean (or a multiple of it if we must still divide by $\sqrt{n}$ for n samples). The remaining Fourier components must give the variability about the mean. How do we add them up? Well, each component corresponds to a vector in an n-dimensional space, as we saw in Week 9 Note 1. So to find the length of the vector (of variability) they make up, we must find the sum of the squares of their magnitudes then take the square root.

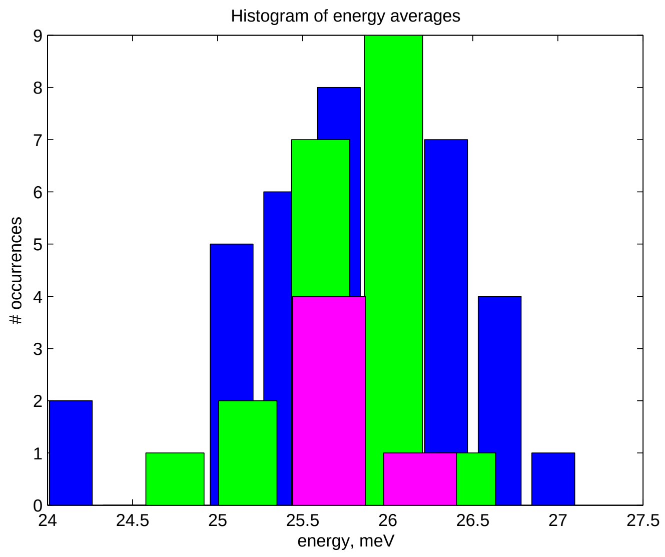
When we do this with another straightforward program (not forgetting to divide lengths by $\sqrt{n}$) we get the following, which I compare with more precise numbers for the standard deviation above.

grouped by	Fourier variability	std.dev.
1s	1.9934	1.9937
2s	1.6927	1.6929
4s	1.5713	
8s	1.2989	1.2989

Finally, we can do the same statistics (but not Fourier transforms) for the average *energies*, which can hardly be said to fluctuate since energy conservation forces the average energy to plot as a straight horizontal line in Note 13.

Although the energy is fixed during one simulation, it will change under different initializations. A third simple program can invoke `moleculeInit()` forty times to generate 40 energy averages.

We can modify the previous program which generated histograms for momentum magnitude averages to do the same for these energy averages. (In fact we could have written it to accept the set of averages as a parameter, and so not have to modify it at all.)



The colours mean the same as before (blue: 40 groups of 1; green: 20 groups of 2; magenta: 5 groups of 8) and we see the same narrowing of the histograms. Standard deviations again confirm this trend.

grouped by	averge	variance	std.dev.
1s	25.9490	0.40	0.63
2s	25.9490	0.17	0.41
8s	25.9490	0.04	0.21

I have made this treatment of energy averages along the same lines as the momentum magnitude averages earlier because the first fluctuates and the second does not, but both can be analyzed statistically as *sets* although only the timeseries can be Fourier-trnsformed as a fluctuating sequence.

Note that β would take on different values for each of these 40 initial conditions, because the mean energy is different in each case. We have ahead of us the problem of finding a single value for β for all such initial conditions.

The narrowing of the histograms as the sample size increases is generally true. Let's work with samples of size 2 from the distribution

values	1	2	3
frequencies	1/4	2/4	1/4

which has mean $m = (1 + 4 + 3)/4 = 2$ and variance $\sigma^2 = (1 + 8 + 9)/4 - m^2 = 1/2$.

There are nine possible samples of two values each, with total values

+	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6

$$(m_{ij}) = \frac{1}{2}\begin{pmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{pmatrix}$$

or
because the sample means are half these numbers. (Note that I'm distinguishing, say, 1 + 2 from 2 + 1, so the answers are symmetrical about the diagonal

The frequencies of these samples are the products of the frequencies of the two values making up the sample.

$\times$	1/4	2/4	1/4
1/4	1/16	2/16	1/16
2/4	2/16	4/16	2/16
1/4	1/16	2/16	1/16

or

$$(p_i p_j) = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

So the average mean over all the samples is the sum of the means weighted by the frequencies

$$\begin{aligned} < m > &= \sum_{ij}(m_{ij})(p_i p_j) \\ &= \frac{1}{2}\frac{1}{16}\sum\begin{pmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{pmatrix} .\times \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \\ &= \frac{1}{32}(2+12+24+20+6) \\ &= 2 \\ &= m \end{aligned}$$

And the average of the sample means is just the mean of the distribution they are drawn from.

But the variance of the sample means is half the variance of the original distribution. The second moments come from the sums of squares

$+$	1^2	2^2	3^2
1^2	2	5	10
2^2	5	8	13
3^2	10	13	18

or

$$(\mu_{2ij}) = \frac{1}{2}\begin{pmatrix} 2 & 5 & 10 \\ 5 & 8 & 13 \\ 10 & 13 & 18 \end{pmatrix}$$

To find the variances, subtract the squares of the means

$$\begin{aligned} (\sigma^2_{ij}) &= \frac{1}{2}\begin{pmatrix} 2 & 5 & 10 \\ 5 & 8 & 13 \\ 10 & 13 & 18 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 2^2 & 3^2 & 4^2 \\ 3^2 & 4^2 & 5^2 \\ 4^2 & 5^2 & 6^2 \end{pmatrix} \\ &= \frac{1}{4}\begin{pmatrix} 0 & 1 & 4 \\ 1 & 0 & 1 \\ 4 & 1 & 0 \end{pmatrix} \end{aligned}$$

The average of all of these again needs the frequencies

$$\begin{aligned} < \sigma^2 > &= \sum_{ij}(\sigma^2_{ij})(p_i p_j) \\ &= \frac{1}{4}\frac{1}{16}\sum\begin{pmatrix} 0 & 1 & 4 \\ 1 & 0 & 1 \\ 4 & 1 & 0 \end{pmatrix} .\times \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \\ &= \frac{1}{4}\frac{1}{16}(4+8+4) \\ &= \frac{1}{4} \\ &= \frac{\sigma^2}{2} \end{aligned}$$

Sticking with sample size 2 we can generalize this argument to any distribution, p_i, summed over numerical values i. The average of the sample means

$$< m > = \sum_{ij} p_i p_j \frac{i+j}{2}$$

$$\begin{aligned}
&= \frac{1}{2}(\sum_{ij} ip_ip_j + \sum_{ij} p_ijp_j) \\
&= \frac{1}{2}(\sum_i ip_i + \sum_j jp_j) \\
&= \sum_i ip_i \\
&= m
\end{aligned}$$

where $\sum_{ij} ip_ip_j =\sim_i ip_i \sum_j p_j = \sum_i ip_i$ to get the third line and $\sum_j jp_j = \sum_i ip_i$ to get the fourth. The average of the sample variances

$$\begin{aligned}
< \sigma^2 > &= \sum_{ij} p_ip_j(\frac{i^2+j^2}{2} - (\frac{i+j}{2})^2) \\
&= \sum_{ij} p_ip_j(\frac{i^2+j^2}{4} - \frac{ij}{2}) \\
&= \frac{1}{2}\sum_i i^2p_i - \frac{1}{2}\sum_{ij} ijp_ip_j \\
&= \frac{1}{2}\sum_i i^2p_i - \frac{1}{2}\sum_i ip_i\sum_j jp_j \\
&= \frac{1}{2}\mu_2^2 - \frac{1}{2}m^2 \\
&= \frac{1}{2}\sigma^2
\end{aligned}$$

In general, for the means of a sample of size N from any distribution,

$$\begin{aligned}
< m > &= m \\
< \sigma^2 > &= \sigma^2/N
\end{aligned}$$

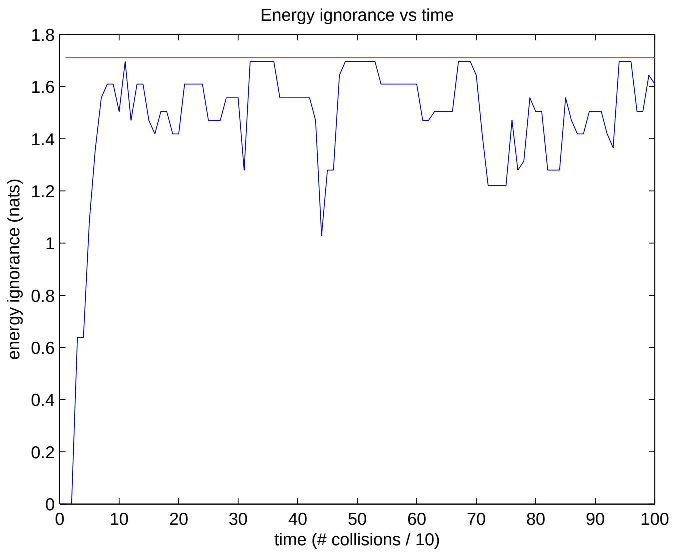
so the standard deviation of the sample means is $1/\sqrt{N}$ times the standard deviation of the distribution the samples are taken from.

This is called the $\sqrt{N}$ law for distributions of sample means.

The term "fluctuation" is often used, confusingly, to mean variability. We have seen in this Note that fluctuations give rise to variance, but we will restrict "fluctuations" to apply to *sequences* of values, often timeseries. Variability, and "variance" in particular, have more general meanings.

16. Entropy. It is significant that the distribution of molecular energies of Note 13 settles down into the distribution of Mom's maximized ignorance of Note 8. We saw this in Notes 13 and 14, where we named it the Boltzmann distribution.

Let's explore the development of the energy ignorance from the initial jet of molecules from the centre of the box.



The ignorance, $-\sum_j p_j \ln p_j$, starts at 0 because all the molecules have the same energy (so one $p_j = 1$ and the rest are 0). Within a dozen collisions it has reached its maximum, and thereafter fluctuates below this max. The first 15 values are, starting before any collisions and recording after every intermolecular collision,

0	0	0	0.64	0.64	1.09	1.36	1.56	1.61	1.61	1.50	1.70	1.47	1.61	1.61

The red line is the maximum ignorance, calculated using the same step size of 10 as used by the statistics gathering.

(This calculation is dicey because, as we saw in the Excursion *Ignorance and continuuous distributions*, the ignorance calculated will grow without limit as the step size decreases. In fact, the max is 4.2 for a step size of 1. Here's the math, the first part of which underlies the above calculation.

$$\begin{aligned} p_j &= e^{-\beta E_j}/Z \\ -\ln p_j &= \beta E_j + \ln Z \\ -\sum_j \ln p_j &= \sum \beta E_j p_j - \ln Z \sum_j p_j \\ &= \beta E + \ln Z \\ &= 1 + \ln Z \end{aligned}$$

where $E = 1/\beta$ is the mean energy.

To get a number for this,

$$\begin{aligned} Z &= \sum_j e^{-\beta E_j} \\ &\approx \frac{1}{\Delta E}\text{antislope}_E e^{-\beta E} \mid_0^\infty \\ &= -\frac{1}{\beta \Delta E} e^{-\beta E} \mid_0^\infty \\ &= \frac{1}{\beta \Delta E} \\ &= \frac{E}{\Delta E} \end{aligned}$$

where I replaced the sum by the corresponding area (the antislope) divided by the distribution step size, $\Delta E = 10$, say.

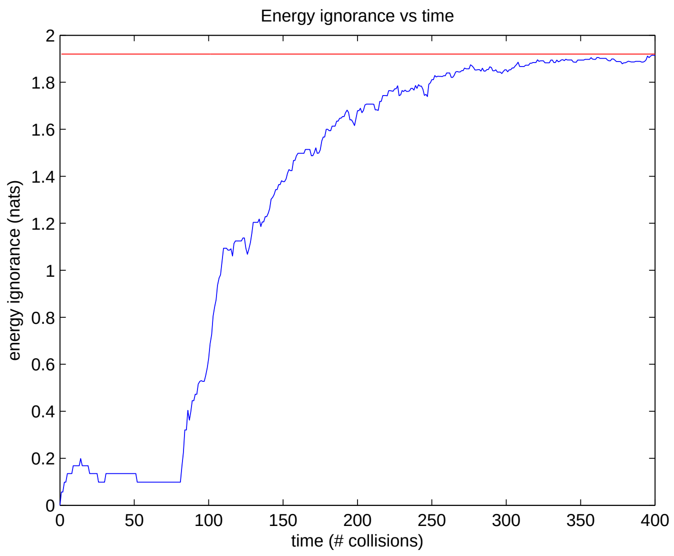
So the ignorance of the final state, which should be the maximum, is

$$\begin{aligned} 1 + \ln Z &= 1 + \ln \frac{E}{\Delta E} \\ &= 1 + \ln \frac{25.0221}{10} \\ &= 1.92 \end{aligned}$$

All these ignorance numbers are rough, but they seem to be about the same size.)

The program is a modification of `cumuSim()` from Note 14 in which the plotting routine is replaced by one which calculates and draws the entropy. It is invoked right at the beginning and after every intermolecular collision.

The fluctuations in the energy ignorance are large for this simulation of 10 molecules. To reduce the fluctuations and show that the increase of ignorance from 0 is real, here is a simulation of 100 molecules.



After starting at 0, nothing muc happens until most of the 100 molecules have collided at least once, then the ignorance climbs steadily towards the theoretical maximum of 1.92.

At this point of equilibrium, the energy distribution has become Boltzmann and the momentum magnitude distribution has become Maxwell.

I have enttled this Note "entropy" but have so far discussed only energy ignorance. The latter is a "dimensionless" quantity, a function only of probabilities. What physics calls entropy has the physical dimensions of energy divided by temperature, say meV per degree Kelvin. So there must be some constant we can find, k_B, with these physical dimensions, which we can multiply by to get entropy

$$S = k_B(\beta E + \ln Z)$$

where S is the conventional symbol in thermodynamice for entropy.

Finding this special constant is still ahead of us.

So we define *entropy* to be k_B times the energy ignorance. Until we find out more about k_B however we'll leave it out and work with energy ignorance alone.

The word "entropy" is from the Greek word $\epsilon\nu\tau\rho\omega\pi\eta$ for "evolution": a system naturally increases its entropy until a maximum is reached, defining a state of equilibrium.

17. Temperature. At last we can understand what temperature is.

You may with *some* justification accuse me here of circularity. Right from the beginning of this Part, from Note 12, I have used temperature (in absolute or Kelvin degrees) as a parameter to the gas simulations. I need not have done this. All along I could have used mean energy instead. But we are all much more familiar with the idea of temperature (at least in degrees Celcius or possibly Fahrenheit) than with mean energy (in milli-electron-Volts). So I used temperature for familiarity even though we do not *understand* what it is.

Temperature is a property of two or more systems in *equilibrium*. That is, systems which have settled down with each other and are no longer making macroscopically observable adjustments. When they are in the kind of equilibrium I am thinking of, they have the same temperature.

For example, two objects at two different temperatures when place in contact with each other will eventually come to have the same temperature—when they are in equilibrium.

We'll use the adjective "thermal" to pin this down a little more because it is still ambiguous: a wet cloth placed on top of a dry cloth will equilibrate to the same degree of dampness (in the absence of evaporation or runoff) but dampness is not what we mean by temperature.

Let's use our gas simulation to show precisely what we mean. Because temperature compares two systems in equilibrium, we'll need two gases. I'm going to let these mix together because we have routines to track their distributions and their energy ignorances. This is the extreme case of putting them in contact through some heat-conducting barrier but not letting them mix—and there are subtle differences which I'm going to gloss over. Simulating the mixture is much easier than simulating a "heat-conducting barrier".

The new simulation makes changes right down to the routines that anticipate and execute collisions with the walls (`molCollWallWhen()` and `molCollWallCountBot()`). These now include means of recognizing when a molecule crosses the x-axis because this will be the initial separator between the two gases, one in the lower half of the box and one in the upper half. The executing routine, `molCollWallCountBot()`, has an extra parameter, `mix`, which allows us to keep the two gases separate (`mix` = 0: the molecules bounce off the x-axis from above and below) or to mix them (`mix` = 1: the molecules cross the x-axis as they did in our earlier simulations of a single gas.

(I have also incuded in `molCollWallCountBot()` three additional parameters which were hitherto frozen to value 1: `tmult, bmult` and `cntB` will be used later as mechanisms for heating and expanding the gases. But we don't use them yet.)

The main simulation, now called `tmptureSim()`, differs in two major ways from previous simulations. First, the parameters describing the gas, M (mass of molecule in AMU), r (radius of molecule in nm), N (number of molecules in the gas) and T (temperature: yes, here it is again—see the second paragraph of this Note) must now describe two gases, so they are now column arrays of two elements each.

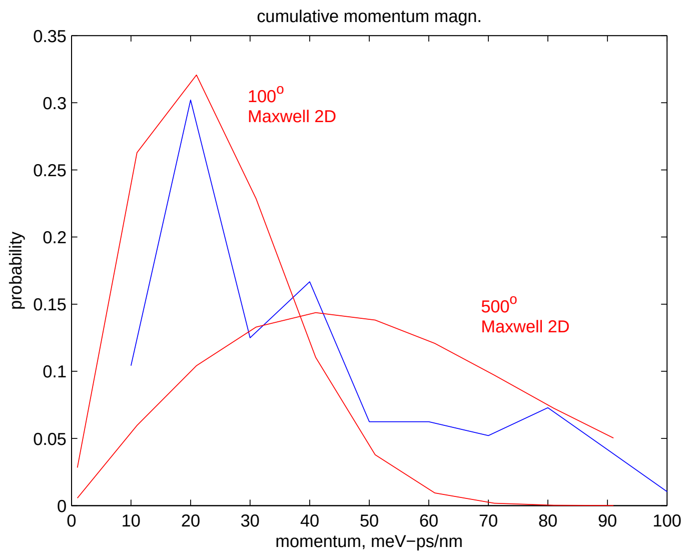
Second, there is now a third possible response to the interactive question, do we want to continue after some multiple of 100 intermolecular collisions. This allows us to switch on gas mixing during the simulation. (It's written as a toggle, so we can switch it off again later, but there is no present need for this.)

The new simulation, `tmptureSim()`, gathers statistics on the energy ignorance (combined for both gases) as a function of time, and on the distribution of momentum magnitudes as a sequence of snapshots after every collision.

I ran it for 1000 collisions, keeping the gases separate for the first 500 then allowing them to mix for the next 500. I made the two gases equal in number (50), mass and radius (4 AMU, 0.05 nm:

i.e., helium) but at temperatures 100 and 500.

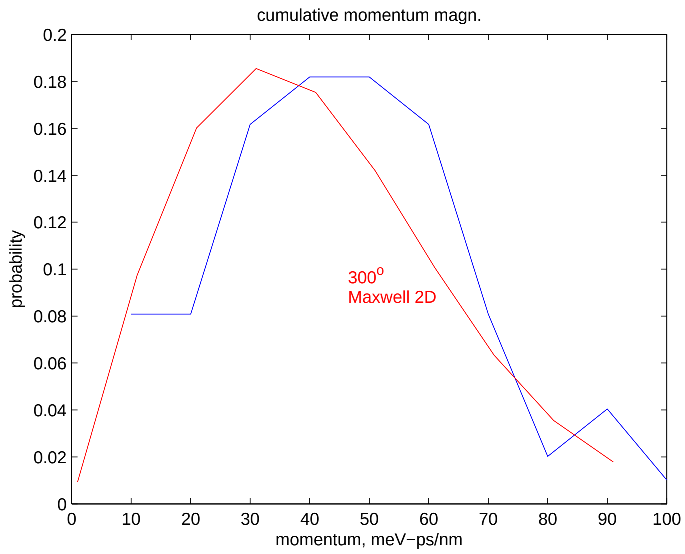
Here is the snapshot of the distribution of momentum magnitudes after the first 500 intermolecular collisions, just before switching on the mixing.



In red are the two corresponding Maxwell distributions, one for the 100°K gas and one for the 500°K gas. The snapshot (blue) is reasonably close to the sum of these.

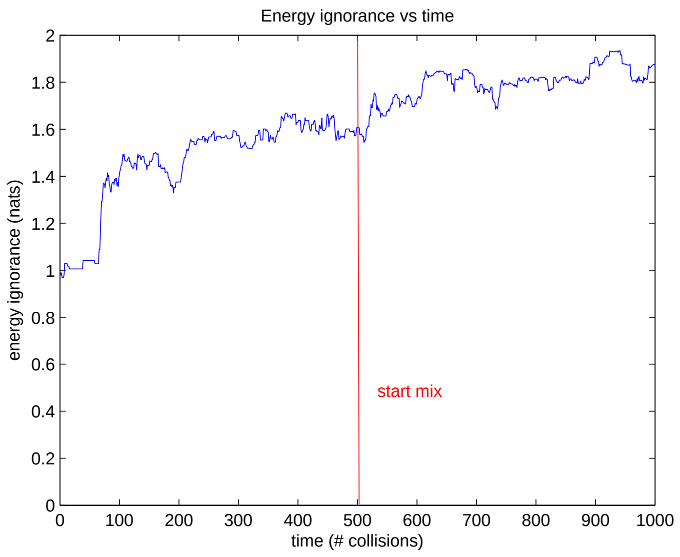
After 500 intermolecular collisions the two gases (He at 100°K and He at 500°K) seem to have settled into individual equilibria—although it is hard to tell from the statistics on so few molecules (50) each—and it seemed a good time to switch on the mixing.

Here is the snapshot after 1000 intermolecular collisions i.e., 500 after allowing the gases to start mixing (blue)



and the Maxwell distribution for momentum magnitudes at 300°K (red).

And finally, here is the time dependency of the energy ignorance.

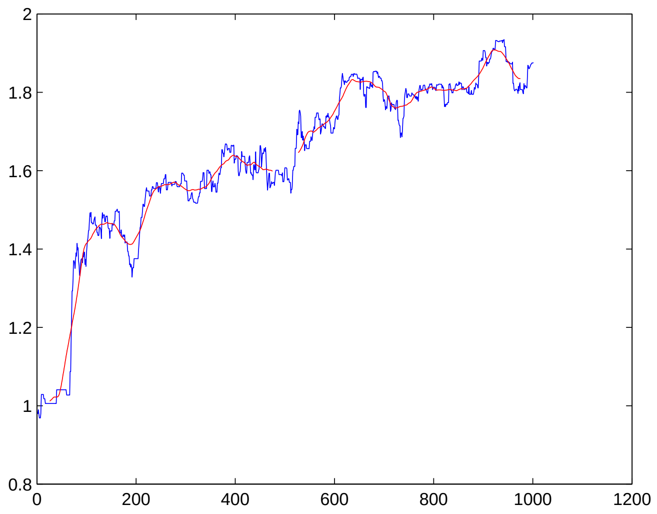


It was this plot that I watched to decide when to turn on the mixing. It seemed to have levelled out to a maximum at the point (5000 intermolecular collisions) when I decided to allow mixing. But it was hard to tell if it wasn't still climbing.

You can see a new climb start just after 500 collisions—although it might be a large fluctuation. This in turn seems to level off to a maximum after another 500.

To try to see the trends of this time-plot of information ignorance I smoothed the data in a simpler way than lossy Fourier transform (which does not permit extrapolation at the ends).

Here is the result for a smoothing with a running average over every 50 points in the timeseries (see the code for `smooth2enerIgn(runlength)` in MATLABpak).



Because I knew where I switched the mixing on, and so broke the smoothing across that point, this plot begs the question somewhat. But perhaps it helps to justify, after the fact, 500 collisions as a good moment to turn on the gas mixing.

Without mentioning temperature, then, the upshot of all this is that the two separate gases, at first, had the two separate mean energies of 8.54 meV and 43.61 meV respectively. The average of this is 26.07 meV and this is the mean energy of the combined gases.

The simulation `tmptureSim()` records all the energies in the array `molData` and reports `molData` for each of the two gases after initialization and for both at the end of the simulation.

It is inevitable that the final mean energy is the average of the two initial mean energies, given that the two initial gases are the same quantities of the same type of gas (helium). (If the quantities had been different, N_1 and N_2, the mean energies $\overline{E_1}, \overline{E_2}$ and $\overline{E}$ of initial gases 1 and 2 and the final gas, respectively, are related as follows.

$$\begin{aligned}\overline{E_1} &= \frac{1}{2m_1}\sum_{j=1}^{N_1}\frac{p_{1j}^2}{N_1}\\ \overline{E_2} &= \frac{1}{2m_2}\sum_{j=1}^{N_2}\frac{p_{2j}^2}{N_2}\\ \overline{E} &= (N_1\overline{E_1}+N_2\overline{E_2})/(N_1+N_2).)\end{aligned}$$

So what is more interesting is the two mean energies of the *separate* gases *after* they have thoroughly mixed. This would be what we observe of two gases in thermal contact but not mixed.

The simulation does not change the locations, as they are recorded in `molData`, of the molecules so we must just compare the first half with the second half of `molData`. The mean energies we get are 22.99 meV and 29.15 meV respectively for the initially cooler and initially warmer gases.

The two gases are converging to the *same mean energy.* This makes mean energy a candidate for what we mean by temperature. You can run the simulation now for different gases, say helium (M = 4 AMU, r = 0.049 nm) and neon (M = 20 AMU, r = 0.051 nm) or argon (M = 40 AMU, r = 0.088 nm) and show that these also converge to the same mean energy aftermixing, So mean energy is a property *independent* of the type of gas. This is also true of temperature.

We could just identify temperature with mean energy and measure it in meV. But we are used to measuring temperature in degrees and so we need a conversion constant. This is called the Boltzmann constant, k_B, which, to convert from meV to degrees Kelvin, is

$$k_B = 0.0862\text{meV/K}^\circ$$

So we get, by dividing by k_B,

meV	°K
8.54	99.1
43.61	506
26.07	302

These are close to, but not exactly, the temperatures I set as parameters, namely 100°K and 500°K, averaging to 300°K. This is because the 50 molecules of each gas in the simulation are *samples* drawn from these temperatures. So they will be a little off, especially as they are *small* samples—not like the 10^{23} molecules of macroscopic amounts of gas.

This motivates a change in the meaning of the parameter β. We have so far considered β to be the reciprocal of the mean energy of the sample. To be more universal, we will from now on define β in terms of temperature, T:

$$\beta = \frac{1}{k_BT}$$

Let's look at this from the point of view of maximizing the combined information ignorance as the system proceeds to equilibrium from the moment when we started to allow the gases to mix.

First, the energy ignorance,

$$I = -\sum_j p_i \ln p_j$$

$$\begin{aligned} &= -\sum_j p_i \ln \frac{e^{\beta E_j}}{Z} \\ &= \beta \sum_j p_j E_j + \sum_j p_j \ln Z \\ &= \beta \overline{E} + \ln Z \end{aligned}$$

because $\sum_j p_j E_j = \overline{E}$ and $\sum_j p_j = 1$ (and I have not gone on to say $\beta = 1/\overline{E}$) where $\overline{E}$ is the mean energy and $Z = \sum_j e^{-\beta E_j}$.

If we want to maximize this, we'll need to set a small change, ΔI, to zero.

I depends on $\beta, \overline{E}$ and e_j so we can write ΔI formally in terms of $\Delta\beta, \Delta\overline{E}$, all the ΔEj and the appropriate slopes.

$$\begin{aligned} \Delta I &= \Delta(\beta\overline{E} + \ln Z) \\ &= \text{slope}_{\overline{E}}(\beta\overline{E})\Delta\overline{E} + \text{slope}_\beta(\beta\overline{E})\Delta\beta + \frac{1}{Z}(\text{slope}_\beta Z \Delta\beta + \sum \text{slope}_{E_j} e^{-\beta E_j}\Delta E_j) \\ &= \beta\Delta\overline{E} + \overline{E}\Delta\beta + \frac{1}{Z}(\sum_j(-E_j)e^{-\beta E_j})\Delta\beta - \frac{\beta}{Z}\sum_j e^{-\beta E_j}\Delta E_j \\ &= \beta\Delta\overline{E} + \overline{E}\Delta\beta - \overline{E}\Delta\beta - \frac{\beta}{Z}\sum_j e^{-\beta E_j}\Delta E_j \\ &= \beta\Delta\overline{E} - \frac{\beta}{Z}\sum_j e^{-\beta E_j}\Delta E_j \end{aligned}$$

The second two terms describe changes to the energy *levels*, E_j, permitted to the gas. If we view the gas quantum-mechanically, these levels are affected by geometry and numbers of molecules, for instance, which, in the experiment of two containers of gas in termal contact, will not change. So $\Delta E_j = 0$.

Second, consider the two gases. As we saw in Note 8, the combined ignorance of two independent systems is the sum of the individual ignorances

$$I = I_1 + I_2$$

We are going to be interested in the total energy ignorance for all the gas, not just per molecule, so we can use this additivity to see

$$I^{\text{G}} = NI^{\text{M}} = N_1 I^{\text{M}} + N_2 I^{\text{M}}$$

where I^{M} is the ignorance per gas molecule and I^{G} is the total for $N = N_1 + N_2$ molecules.

Third, for the two systems, the total energy is the sum of the two energies. We'll write U for the total energy: it relates to the mean energy per molecule by $U = N\overline{E}$. Thus

$$U = U_1 + U_2 = N_1\overline{E_1} + N_2\overline{E_2}$$

Fourth we maximize the total energy ignorance for equilibrium by setting $\Delta I^{\text{G}} = 0$.

$$\begin{aligned} 0 = \Delta I^{\text{G}} &= N_1\Delta I_1^{\text{M}} + N_2\Delta I_2^{\text{M}} \\ &= N_1\beta_1\Delta\overline{E_1} + N_2\beta_2\Delta\overline{E_2} \\ &= \beta_1\Delta U_1 + \beta_2\Delta U_2 \end{aligned}$$

Notice that we have two different values of β, in general, for the two gas systems.

Since the total energy, U, is constant by conservation of energy, while the component energies U_1 and U_2 are adjusting

$$0 = \Delta U = \Delta U_! + \Delta U_2$$

so $\Delta U_2 = -\Delta U_1$ and

$$\begin{aligned} 0 = \Delta I^{\mathrm{G}} &= \beta_1 \Delta U_1 - \beta_2 \Delta U_2 \\ &= (\beta_1 - \beta 2)\Delta U_1 \end{aligned}$$

This is true for any ΔU_1 which we can suppose is not zero itself, so

$$\beta_1 = \beta_2$$

is the condition for equilibrium. This corresponds to our idea about temperature: temperature must be some function of β so that two equal values of β give equal temperatures.

To figure out what this function is, suppose the two gases are not quite at equilibrium so $\Delta I^{\mathrm{G}} > 0$. We know that $\beta > 0$ because $e^{-\beta E}$ decreases as E increases (and anyway, $\beta < 0$ would make $e^{-\beta E}$ increase without limit as E increases).

So think about a positive ΔU_1, i.e., energy flowing *into* gas 1:

$$0 < \Delta I^{\mathrm{G}} = (\beta_1 - \beta_2)\Delta U_1$$

means $\beta_1 > \beta_2$. But experience tells us that for the second gas to be heating the first gas, its temperature must be higher: $\beta_1 > \beta_2$ corresponds to $T_1 < T_2$.

So the simplest function must be

$$\beta = \frac{1}{kT}$$

where k is some constant. This k could be 1, but we've already noted that the Boltzmann constant k_B connects temperature to mean energy, and so we take $k = k_B$:

$$\beta = \frac{1}{k_B T}$$

We now have both simulation and theory pointing to the same definition of temperature.

One more thing. We can relate temperature to the slope of entropy. This will justify our use of k_B in the definition of entropy in the previous Note.

We had

$$\Delta I = \beta \Delta \overline{E} - \frac{\beta}{Z}\sum_j e^{-\beta E_j} \Delta E_j$$

so

$$\mathrm{slope}_{\overline{E}} I = \beta$$

Now increase the energy ignorance I ($= I^{\mathrm{M}}$, per molecule) to the whole gas, $I^{\mathrm{G}} = N I^{\mathrm{M}}$, and convert it to entropy for the whole gas

$$\begin{aligned} \Delta S^{\mathrm{G}} &= k_B N \Delta I^{\mathrm{M}} \\ &= k_B N \beta \Delta \overline{E} + \text{terms not in } \Delta \overline{E} \\ &= k_B \frac{1}{k_B T} \Delta(N\overline{E}) + \text{terms not in } \Delta(N\overline{E}) \\ &= \frac{1}{T}\Delta U + \text{terms not in } \Delta U \end{aligned}$$

so

$$\text{slope}_U S^{\text{G}} = \frac{1}{T}$$

This could also be taken as a definition of temperature if we wanted to start abstractly. We'll find that thermodynamics is based on such abstractions.

The quantities that increase by a factor N in going from individual molecules to the whole gas are called *extensive*.

$$\begin{aligned} S^{\text{G}} &= NS^{\text{M}} \\ V^{\text{G}} &= NV^{\text{M}} \\ N &= N \times 1 \end{aligned}$$

The complemetary quantities, P, T and μ, do not behave this way, and are called *intensive*.

From now on I will stop writing the superscripts M for individual molecules and G for the whole gas: extensive variables will always refer to the whole gas (G is implicit) unless I explicitly say otherwise.

18. Pressure. The energies of the molecules have given us a statistic, the mean energy, which is an observable and measurable quantity, the temperature. What about their momenta?

The mean momentum is, of course, zero, or the whole box would be moving. (The mean momentum *magnitude* is not zero because all the minus signs disappear when we take the magnitudes.) What is interesting about the momenta of a stationary box of gas?

Every time a molecule hits a side of the box its component of momentum normal to the side is reversed. Ths gives rise to a *pressure* on the sides due to the gas. To ensure that this pressure does not depend on the size of the side (length in 2D, area in 3D) we divide the change of momentum by that. To ensure that the pressure does not depend on how long we keep the gas in the box, we divide again by the length of time.

So pressure is a rate of change of momentum per unit side. We'll use capital P for pressure because we have used small $\vec{p}$ for momentum. Here are possible units

$\vec{p}$ meV-ps/nm

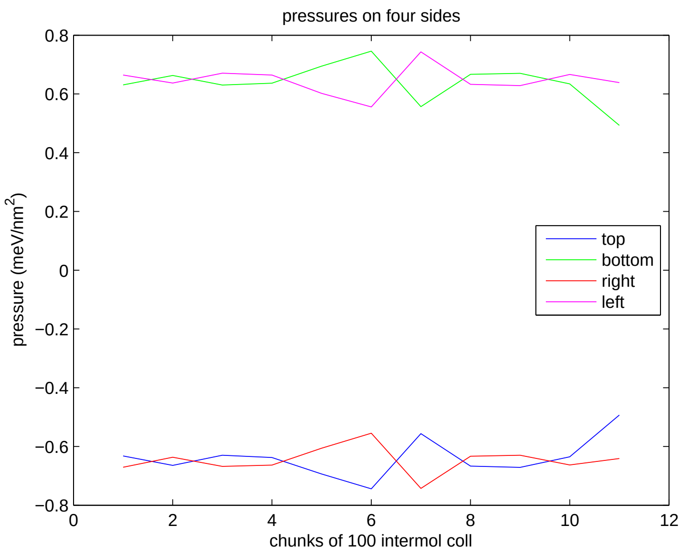
P meV-ps/nm (/mn/ps in 2D; /nm^2/ps in 3D)

i.e., pressure is meV/nm^2 in 2D and meV/nm^3 in 3D.

For comparison in 3D, 1 atmosphere is 101325 J/m^3 and a milli-eV is 1.6×10^{-22} J. "Standard pressure" is defined to be 10^5 J/m^3 because it is close enough to 1 atm, so we will use that. The conversion is to multiply meV/nm^3 by 1.6×10^{-5} to get J/m^3 and simply by 1.6 (or divide by 0.624) to get standard pressure.

The simulation `presSim(N,T,r,M,ab)` is a modification of `statSim()` which no longer tracks or plots (mean) energies or (mean) momentum magnitudes but instead sums up the momentum reversals for each of the four sides, then, every 100 intermolecular collisions (and this should correspond to a lot more collisions with the walls), divide by the simulated time that has elapsed and by the respective lengths of the sides.

For 1100 intermolecular collisions of 10 molecules at 300°K the 11 averages are



We see that the pressures are the same on all four sides (to plot them more clearly I've kept the minus signs for momentum reversals off the top and right sides).

If we do the theory, we can relate pressure to temperature. Here is Feynman's argument [FLS64, §39-2; pp. 39-2–6] using $\vec{p} = m\vec{v}$.

For molecules hitting the top (leaving off the minus sign) the change of momentum is $2mv_y$ and the rate of change in time is $2mv_y/t$.

How many molecues travelling with vertical speed v_y will hit the top in time t? They must be within a distance v_yt of the top, or, if the top has area A, within a volume v_ytA. And if there are $n = N/V$ molecules per unit volume, this is a *count* of nv_ytA molecules.

(The area A would be the length L in 2D, but it's going to cancel out, so the discussion will be valid for both 3D and 2D, up to a point.)

So the rate of change of momentum of molecules moving upwards at speed v_y is $2mv_ynv_ytA$ and per unit time and per unit area it is the pressure

$$P = 2nmv_y^2$$

But of course molecules do not all move at the same speed v_y so we must use the average of v_y^2 which we'll write $< v_y^2 >$. Then

$$P = nm < v_y^2 >$$

Why, Feynman asks, did we drop the 2? Because the average of v_y^2 includes the half of the molecules moving *away* from the top.

Now physics has observed that pressures are the same in all directions, so

$$< v_x^2 >=< v_y^2 >$$

This is handy because we'd like to use $< v^2 >=< v_x^2 > + < v_y^2 >$ instead of $< v_y^2 >$:

$$< v^2 >=< v_x^2 > + < v_y^2 >= 2 < v_y^2 >$$

Thus

$$P = n\frac{1}{2}m < v^2 >= n\overline{E} = nk_BT$$

(In 3D, $< v^2 >=< v_x^2 > + < v_y^2 > + < v_z^2 >= 3 < v_y^2 >$, so $P = \frac{2}{3}n\frac{1}{2}m < v^2 >= \frac{2}{3}n\overline{E} = nk_BT$)

Finally we defined $n = N/V$ so

$$PV = Nk_BT$$

This is the "ideal gas law". An *ideal gas* is a gas of monatomic molecules which don't even collide with each other. Although `pressSim()` kept the intermolecular collisions of `statSim()` and all the other gas simulators, it gathered data only at the walls and it should be obvious that the intermolecular collisions will not affect them. Indeed, pressure depends only on mean energy $\overline{E}$ and density n, and collisions do not change this.

Remarkably, the ideal gas law is the same (for monatomic gases) in 3D as in the 2D gas for which we derived it. This is because of *equipartition of energy*: the mean energies of each direction of motion are the same (as are the total energies in each direction).

We can check this in the simulation, since we have the final snapshot of `molData` (or any other snapshot will do). If we average

$$p_x^2/(2m) \qquad \text{and} \qquad p_y^2/(2m)$$

for all molecules, we get (for the simulation I made in Note 17 with `tmptureSim()`: but it could have been done in any of our simulations)

$$\begin{aligned}\overline{E_x} &= 12.33\,\text{meV}\\ \overline{E_y} &= 13.74\,\text{meV}\end{aligned}$$

These are close enough for a snapshot of only 100 molecules.

Here is the theory. The mean energy in the x direction is

$$\frac{\text{antislope}p_x^2e^{-p_x^2/a}/(\beta a)\mid_{-\infty}^{\infty}}{\text{antislope}e^{-p_x^2/a}\mid_{-\infty}^{\infty}} = \frac{\sqrt{\pi a}/(2\beta)}{\sqrt{\pi a}} = \frac{1}{2\beta} = \frac{1}{2}k_BT$$

(We did the numerator as μ_2 of Maxwell in Note 14 and the denominator as I in Note 6.)

The mean energy in the y direction is the same. So is the mean energy in the z direction if there is one.

So each direction has a mean energy of $k_BT/2$.

Thus

	2D	3D
$\overline{E}$	k_BT	$\frac{2}{3}k_BT$
PV	$N\overline{E} = Nk_BT$	$\frac{3}{2}N\overline{E} = Nk_BT$

Of course, volums V in 3D is measured in length^3. "Volume" in 2D is really area and measured in length^2.

The equipartition of energy happens to hold for other kinds of energy, such as rotation or vibration of polyatomic molecules. Each "degree of freedom" adds another $k_BT/2$ to the mean energy $\overline{E}$. For instance, a diatomic molecule in 3D has six degrees of freedom (x, y and z for each atom: these can be looked at as x, y or z for the centre of mass, plus two axes of rotation plus a third one of vibration) and so mean energy of $3k_BT$.

Thus for a general gas we can write

$$PV = (\gamma - 1)U$$

where $\gamma = 5/3$ for monatomic gases in 3D since $U = N\overline{E} = (3/2)Nk_BT$, i.e., $PV = (2/3)U$.

For monatomic gases in 2D, we can see that $\gamma = 2$.

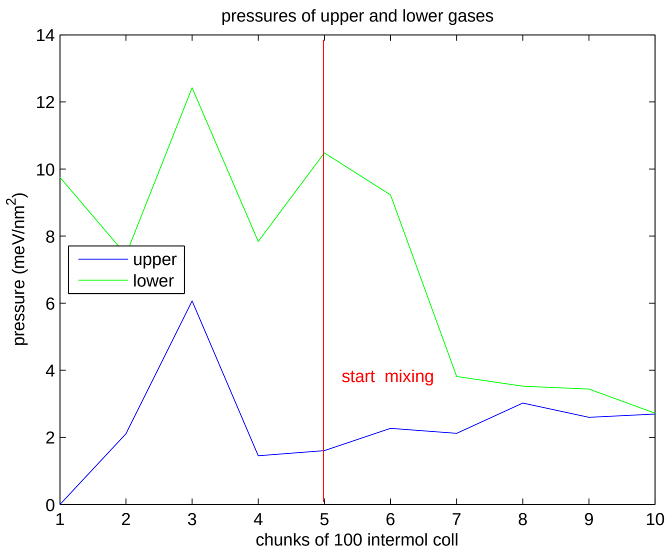
Feynman [FLS64, §39-2] and Note 22 show that this means

$$PV^\gamma = \text{constant}$$

for any process, such as our simulations so far, in which no energy is lost through the walls—although it might be lost through work done by the gas such as expanding the container, causing volume V to change. We'll come to the subtle discussion of heat and work in Note 22.

What happens to the pressures of two gas samples as they reach equilibrium? We can simulate. Once again we can avoid the difficulties of inventing a mechanism to keep the gases separate and at the same time transmit temperature and pressure, because the simulation can track the molecules and distinguish each gas.

Here is the convergence of pressures of helium (50 molecules) at 100°K ("upper") and helium (50 molecules) at 500°K ("lower") after mixing starts after 500 intermolecular collisions.



The theory for pressure equalization extends the theory in Note 17 for temperature equalization and is a little more abstract.

In Note 17 we showed that the change in energy ignorance (per molecule)

$$\Delta I^{\mathrm{M}} = \beta\Delta\overline{E} - \frac{\beta}{Z}\sum_j e^{-\beta E_j}\Delta E_j$$

and I said that the energy levels E_j depend on geometry (and other things). Inasfar as E_j depends on, say, volume per molecule, V^{M}

$$\Delta E_j = \mathrm{slope}_{V^{\mathrm{M}}} E_j \Delta V^{\mathrm{M}} + \text{other terms}$$

So whatever

$$-\frac{\beta}{Z}\sum_j e^{-\beta E_j}\Delta E_j = -\frac{\beta}{Z}(\sum e^{-\beta E_j}\mathrm{slope}_{V^{\mathrm{M}}} E_j)\Delta V^{\mathrm{M}}$$

is, we can invent a name, say P for everything but the $\beta\Delta V$.

So we get

$$\Delta I^{\mathrm{M}} = \beta\Delta\overline{E} + \beta P\Delta V^{\mathrm{M}} + \text{other terms not in } \Delta\overline{E}, \Delta V$$

I didn't yet claim that P is a pressure, but it does have the physical dimensions of pressure: PV is energy, because β is 1/energy and I is dimensionless, so the physical dimensions of P is energy/volume, just like pressure, above.

Furthermore, for equilibrium we have maxium ignorance and, switching to the whole gas, $I^{\rm G} = NI^{\rm M}$, $U = N\overline{E}$ and $V = NV^{\rm M}$, therefore

$$\begin{aligned} 0 = \Delta I^{\rm G} &= \beta\Delta U + \beta P\Delta V \\ &= \Delta I_1^{\rm G} + \Delta I_2^{\rm G} \\ &= (\beta_1 - \beta_2)\Delta U_1 + (\beta_1 P_1 - \beta_2 P_2)\Delta V_1 \end{aligned}$$

where, again, the total energy $U_1 + U_2$ is constant and so is the total volume $V_1 + V_2$, but $\Delta U_1 \neq 0$ and $\Delta V_1 \neq 0$, so, as before

$$\beta_1 = \beta_2$$

and now, also

$$P_1 = P_2$$

Furthermore, at thermal equilibrium ($\beta_1 = \beta_2$) but with the pressures not matched, and for a positive $\Delta I^{\rm G}$ and a positive ΔV_1

$$0 < \Delta I^{\rm G} = \beta_1(P_1 - P_2)\delta V_1$$

means that $P_1 > P_2$ which we would expect if P were pressure: the gas at the greater pressure expands into the other (think of blowing up a balloon).

So P a) has the physical dimension of pressure, b) equalizes at equilibrium, and c) causes expansion from the gas with greater P. Let's conclude that P = pressure.

Notice that this discussion also tells us that

$$\text{slope}_V S^{\rm G} = \frac{P}{T}$$

for $S^{\rm G} = k_B I^{\rm G}$.

19. State function for monatomic gases. In the last two Notes we have found both temperature and pressure as slopes of the entropy S, with respect to total energy U and total volume V, respectively.

This suggests that we might be able to write $S(U, V, ...)$, i.e., entropy as a *function* of U, V and other variables. In general we cannot, but for monatomic gases whose molecules do not interact beyond colliding with each other, and which do not lose energy to or gain energy from the outside, we can.

We must work with probabilities in a different way. Instead of the probability that a *molecule*, say the jth, has energy E_j, we'll consider the probability that the whole gas, in *state* j, has energy U_j, and we'll call this probability p_j. For equilibrium we maximize the whole-gas energy ignorance

$$I^{\rm G} = -\sum p_j \ln p_j$$

subject to

$$1 = \sum p_j$$

and

$$< U >= \sum p_j U_j$$

But, given our specification that no energy leaves or enters, all the U_j are the same, so the second constraint is equivalent to the first.

Maximizing $I^{\rm G}$ makes all the probabilities the same

$$p_j = \frac{1}{\Omega}$$

where Ω is the total number of possible states of the gas with energy $U_j =< U >$ which we'll just call U.

Thus

$$I^{\mathrm{G}} = \ln \Omega$$

and

$$S = k_B \ln \Omega$$

and we must just figure out what Ω is.

The states with kinetic energy U are given by quantum mechanics. In Week 7a Note 3 we used $e^{-i(\omega t - kx)}$ to describe a one-dimensional particle, where k is the wavenumber (*not* the Boltzmann constant), related to momentum by $p = \hbar k$.

In two or three dimensions, k becomes a vector $\vec{k} = (k_x, k_y)$ or $\vec{k} = (k_x, k_y, k_z)$ and kx becomes the dot product $\vec{k}.\vec{x} = xk_x + yk_y + ...$

Now we want to put the particle (a molecule) in a box of sides $a, b, ..$: the space part of $e^{-i(\omega t - \vec{k}.\vec{x})}$, namely $e^{i\vec{k}.\vec{x}}$, must have *nodes* at the sides of the box. This means that xk_x, yk_y, etc. must each be an integer multiple of π at $x = a$, $y = b$, etc.:

$$\begin{aligned} ak_x &= q\pi \\ bk_y &= r\pi \end{aligned}$$

So the particle energy, $\vec{p}^2/(2m)$

$$\begin{aligned} \frac{\vec{p}^2}{2m} &= \frac{\hbar \vec{k}^2}{2m} \\ &= \frac{\hbar^2 \vec{k}^2}{2m}(\frac{q^2}{a^2} + \frac{r^2}{b^2} + ..) \\ &= \frac{h^2}{8m}(\frac{q^2}{a^2} + \frac{r^2}{b^2} + ..) \end{aligned}$$

and the total energy is the sum over all particles

$$U = \frac{h^2}{8m}\sum_{j=1}^{N}(\frac{q_j^2}{a^2} + \frac{r_j^2}{b^2} + ..).$$

This describes an ellipse in $2N$ (or $3N$) dimensions—a hyperellipse. We explored higher-dimensional spheres in Part I, where we found the volume to be

$$\mathrm{Vol}_{2N\mathrm{dim}} = \frac{\pi^N R^{2N}}{N!}$$

and

$$\mathrm{Vol}_{3\mathrm{dim}N} = \frac{\pi^N R^{\frac{3N}{2}}}{(\frac{3N}{2})!}$$

(provided we insist in the $3N$dim case that N is an even number of particles—but it won't be too wrong if N is odd).

So let's make the box square, or cubic, with every side of length a. This gives a hypersphere

$$\sum_{j=1}^{N}(q_j^2 + r_j^2 + ..) = \frac{8ma^2U}{h^2} = R^2$$

(rearranged to bring out the radius R).

The allowed states, of energy U, will be points on the *surface* of this hypersphere with integer values of $q_j, r_j, \ldots$ But in the *Hypersphere* Excursion of Week9cII we saw that the entire volume of a d-dimensional hypersphere equals the volume of a shell of thickness R/d at the surface and so for large d (such as $d = 2N$ or $d = 3N$) the states at the surface effectively equal the states in the whole hypersphere.

So we'll use $\mathrm{Vol}_{2N\mathrm{dim}}$ or $\mathrm{Vol}_{3N\mathrm{dim}}$, above, to count the states. This count is

$$\Omega = \mathrm{Vol}_{2N\mathrm{dim}}/(2^{2N}N!) \qquad \text{or} \qquad \mathrm{Vol}_{3N\mathrm{dim}}/(2^{3N}N!)$$

where we divided a) by 2^{2N} (2^{3N}) to restrict the $q_j, r_j, ..$ to *positive* integers, and b) by $N!$ because this is the number of ways we can arrange N molecules in all the states, but quantum mechanics says we cannot distinguish these arrangements from each other.

Focussing now on 2D

$$\begin{aligned}\Omega &= \frac{\pi^N}{2^{2N}(N!)^2}R^{2N} \\ &= \frac{\pi^N}{2^{2N}(N!)^2}\left(\frac{8ma^2U}{h^2}\right)^N\end{aligned}$$

(the extra $N!$ is from the hypersphere volume).

To find the logarithm of this we'll need another way to write $N!$. *Stirling's approximation*

$$N! = N^N e^{-N}\sqrt{2\pi N}$$

can be derived with much the same math as the hypersphere volume calculations, but we don't do it here. It is plausible that $N!$ should be somewhat less than N^N and the e^{-N} does the reduction. When we come to take logarithms, $\sqrt{2\pi N}$ will be negligible compared with N^N and e^{-N}.

So

$$\begin{aligned}S &= k_B \ln\Omega \\ &= k_B \ln\left(\left(\frac{8\pi ma^2U}{4h^2}\right)^N \frac{e^{2N}}{2\pi N N^{2N}}\right) \\ &= 2Nk_B + Nk_B\ln\left(\frac{2\pi mV}{h^2}\frac{UV}{N^2}\right) - k_B\ln 2\pi N\end{aligned}$$

($a^2 = V$, the "volume", and we neglect the third term.

Let's see if the slopes we found in Notes 17 and 18 give the results we had then.

$$\frac{1}{T} = \mathrm{slope}_U S = Nk_B \mathrm{slope}_U \ln(C_U U) = \frac{Nk_B}{C_U U}C_U = \frac{Nk_B}{U}$$

where I've stuck everything that does not contain U into the constant C_U.

Thus

$$U = Nk_BT$$

and in Note 18 we had the same thing, in the form $\overline{E} = k_BT$. Next

$$\frac{P}{T} = \mathrm{slope}_V S = Nk_B \mathrm{slope}_V \ln(C_V V) = \frac{Nk_BC_V}{C_V V} = \frac{Nk_B}{V}$$

This is the ideal gas law of Note 18:

$$PV = Nk_BT$$

Thus we have the function of state

$$S(U, V, N)$$

depending on U and V as we wished, and also on N.

This is the two-dimensional variant of the *Sackur-Tetrode* equation. See the Excursions for Sackur-Tetrode itself.

(In Note 26 of Part III we'll extend this to include electric charge, and other extensions are also interesting and important.)

This suggests that we should wonder about the significance of

$$\begin{aligned}\text{slope}_N S &= k_B \ln(\frac{2\pi m}{h^2}\frac{UV}{N^2}) \\ &= k_B \ln(\frac{UV}{N^2}\frac{N_0^2}{U_0 V_0})\end{aligned}$$

Where I've invented $N_0^2/(U_0 V_0) = 2\pi m/h^2$ to simplify the coming calculations, and where you should confirm that the $2k_B$ term is cancelled by a $-2k_B$ term.

By analogy with $P/T = \text{slope}_V S$ we'll define a new quantity μ

$$\frac{\mu}{T} = -\text{slope}_N S = -k_B \ln(\frac{UV}{N^2}\frac{N_0^2}{U_0 V_0}) = k_B \ln(\frac{N^2}{UV}\frac{U_0 V_0}{N_0^2})$$

where the minus sign will make sense soon.

Whatever the quantity μ means, for equilibrium of two samples of the same gas, by the same argument that established $P_1 = P_2$ in Note 18, we get, with both temperatures and pressures matched

$$\mu_1 = \mu_2$$

and, just before equilibrium, positive ΔS makes matter flow from larger μ to smaller μ:

$$\begin{aligned}0 < \Delta S &= \frac{1}{T}\Delta U + \frac{P}{T}\Delta V - \frac{\mu}{T}\Delta N \\ &= \frac{\Delta U_1}{T_1} + \frac{\Delta U_2}{T_2} + \frac{P_1 \Delta V_1}{T_1} + \frac{P_2 \Delta V_2}{T_2} - \frac{\mu_1 \Delta N_1}{T_1} - \frac{\mu_2 \Delta N_2}{T_2}\end{aligned}$$

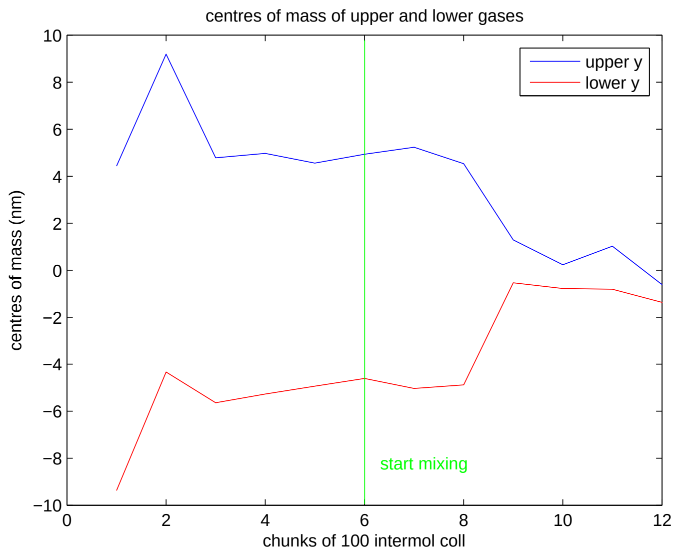
With $T_1 = T_2$ and for constant U, V and N, $\Delta U_2 = -\Delta U_1$, $\Delta V_2 = -\Delta V_1$, $\Delta N_2 = -\Delta N_1$, then

$$0 < S = (\mu_2 - \mu_1)\frac{\Delta N_1}{T}$$

so for a flow of (positive) ΔN_1 particles from gas sample 2 to gas sample 1,

$$\mu_2 > \mu_1$$

Here is a simulation. I've tracked the y component of the centre of mass of two gas samples. Before mixing, these settle down to $y_{\text{cm1}} = 5$ nm for the upper gas (confined to $0 \le y \le 10$) and $y_{\text{cm1}} = -5$ nm for the lower gas (confined to $-10 \le y \le 0$). On opening the barrier the gases start mixing: the upper centre of mass moves downwards to $y_{\text{cm1}} = 0$ and the lower centre of mass moves upwards to $y_{\text{cm2}} = 0$.

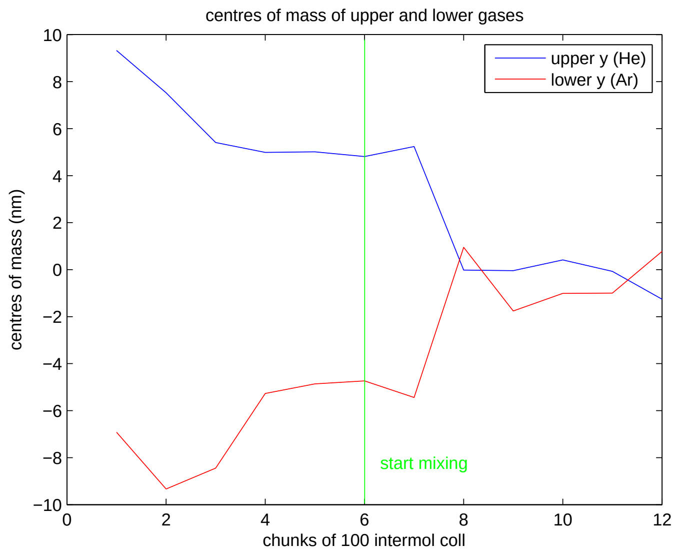


This process is *diffusion* but alternatively we can think of something *attracting* the centres of mass in each case.

It is important that we do not think of the upper and lower centres of mass attracting *each other*, although the plot suggests that they do. If one of the halves of the container were initially empty, the gas in the other half would diffuse into the empty half on opening the barrier, thus moving its centre of mass in the same way as here, but all by itself.

The formalism we used to describe the attraction of a planet to the sun (Excursion *Kepler I* in Part III of Book 8c in Symmetry) was a *potential*. This was elaborated in Note 16, Part IV, Book 8c. (We called it "potential energy" then; now I'm shortening it to one word.) So we'll say that this attraction of the gas (i.e., of its centre of mass) is also due to a potential.

We'll call it the *chemical potential* since it is different for different kinds of molecule. Here is the result of simulating helium in the upper compartment and argon in the lower compartment.



The difference with the helium-helium simulation is perhaps not convincing. We will come to a further understanding of the chemical potential, μ, only in Note 26 of Part III, when we deal with diffusion, and in the Excursion *Chemistry in the language of physics* in Part III.

We can relate the complicated expression defining the chemical potential to the pressure of the gas as follows. For an ideal gas (in 2D)

1) $V = Nk_BT/P$

2) $U = Nk_BT$

So $N^2/(UV) = P/(k_B^2T^2)$

and $U_0V_0/N_0^2 = k_B^2T_0^2/P_0$

So

$$\begin{aligned}\frac{\mu}{T} &= k_B\ln(\frac{P}{P_0}\frac{T_0}{T}) \\ &= k_B\ln(\frac{P}{P_0})\end{aligned}$$

if we say T is fixed, $T = T_0$.

It makes sense that diffusion, described by chemical potential, should depend on pressure.

Much of the time we wil be interested in *small changes*, $\Delta S, \Delta U, \Delta V$, etc., of the respective quantities. This, plus our discovery that the complementary quantities, T, P, etc., are simply related to the *slopes* of S, say, with respect to U, V, etc., enables us to work with "equations of state" which, unlike Sackur-Tetrode for the ideal gas, are too complicated to know exactly.

20. Thermostatic equations of state. With a monatomic gas we could write the entropy, $S(U, V, N)$, explcitly as a function of the state (in 3D, the Sackur-Tetrode function, from the Excursion of that name).

This is hard to do for gases of molecules that rotate, vibrate, or have other forms of internal energy, or for gases whose molecules interact with each other in ways more complicated than simply bouncing off each other elastically—e.g., through electric fields.

But if we content ourselves with *changes* of values of S, U, V, N and the other quantities we derived in the last two Notes—especially very *small* changes—we need only know that S *is* a function of U, V, and N. Without knowing the form of $S(U, V, N)$ we can still write

$$\Delta S = \frac{1}{T}\Delta U + \frac{P}{T}\Delta V - \frac{\mu}{T}\Delta N$$

because

$$\begin{aligned}\text{slope}_U S &= \frac{1}{T} \\ \text{slope}_V S &= \frac{P}{T} \\ \text{slope}_N S &= -\frac{\mu}{T}\end{aligned}$$

Slopes permit *linear approximations* to functions, whatever the functions are, and since we know the slopes of S, we can get results even without knowing what S is in general.

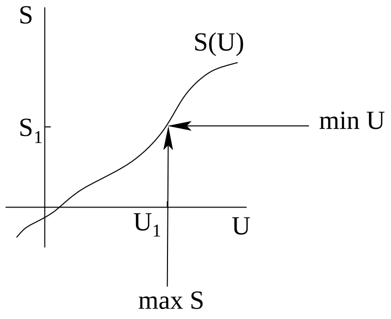
First, we can "invert" $S(U, V, N)$, or at least turn it "inside-out", to find $U(S, V, N)$—i.e., that U is a function of S, V and N, even though we do not know it completely.

$$\Delta U = T\Delta S - P\Delta V + \mu\Delta N$$

The Sackur-Tetrode equation, and its two-dimensional analogue, is a *monotonic* relationship between entropy S and energy U. While the relationship is not always monotonic, when it *is*—as for ideal gases—it guarantees gobal, not just local, inverses, $S(U, ..)$ and $U(S, ..)$.

A monotonic function either never decreases or never increases as its argument increases. S is a monotonically increasing function of U for ideal gases.

Whenever this is the case we can also see that maximizing S for a fixed U corresponds to minimizing U for a fixed S. Here is why.



Equilibrium is given by the curve $S(U)$. Since equlibrium corresponds to maximum S, systems not in equilibrium all correspond to (U, S) points *below* the curve. Maximizing S for a given U, say U_1, brings the system up to the curve at $S_1 = S(U)$. But clearly this is the same point reached by fixing S and S_1 and minimizing U to try to bring it to $U_1 = S^{-1}(S_1)$, the same point (U_1, S_1) on the curve.

Second, we can replace variables by their complements to get a suite of other functions related to $U(S, V, N)$ by sharing slopes.

The *complement* of a variable x with respect to a function F is the slope of the function with respect to the variable, $y = slope_x f$, which we then treat as a variable in its own right.

We can get new functions, depending on different variables, but sharing slopes with $U(S, V, N)$ by *Legendre transformations.* For example, the *enthalpy*

$$H = U + PV$$

uses the complement of V in U, $P = -\text{slope}_V U$ (see ΔU above), and is a function $H(S, P, N)$ which we can now show (locally).

$$\begin{aligned} \Delta H &= \Delta U + \Delta(PV) \\ &= T\Delta S - P\Delta V + P\Delta V + V\Delta P + \mu\Delta N \\ &= T\Delta S + V\Delta P + \mu\Delta N \end{aligned}$$

so H is $H(S, P, N)$.

But H and U share the slopes with respect to their common variables.

$$\begin{aligned} \text{slope}_S H &= T = \text{slope}_S U \\ \text{slope}_N H &= \mu = \text{slope}_N U \end{aligned}$$

Besides the enthalpy we can define six other related functions. Here are the common three, plus the two we alredy know.

internal energy	$U(S, V, N)$
enthalpy	$H(S, P, N) = U + PV$
(Helmholtz) free energy	$A(T, V, N) = U - TS$
Gibbs free energy	$G(T, P, N) = U - TS + PV$
grand potential	$\Phi(T, V, \mu) = U - TS - \mu N$

Here are all eight functions (columns), all three pairs of complements (rows) and all the slopes.

$\text{slope}_r c$ | $c =$	U	A	H	X	G	Φ	Y	Z
$r = S$	T		T	T			T	
T		$-S$			$-S$	$-S$		$-S$
V	$-P$	$-P$				$-P$	$-P$	
P			V	V	V			V
N	μ	μ	μ		μ			
μ				$-N$		$-N$	$-N$	$-N$

What is the use of all these functions? They are not the same as each other—"energy" and "potential" appear in the names of most of the named ones because they all have the physical dimensions of energy—but they do share slopes, as the table shows.

In general thermostatics where we know only the slopes and not the functions themselves, these variants give useful different perspectives on the system.

For example, we can use enthalpy to express the chemical potential as a function of pressure in equilibrium ($\Delta S = 0$):

$$\mu = \text{slope}_N H$$

Since

$$\begin{aligned} \Delta H &= T\Delta S + V\Delta P + \mu\Delta N \\ &= V\Delta P + \mu\Delta N \\ &= V\Delta P \end{aligned}$$

if we also conserve the number of particles N so $\Delta N = 0$.

Now choose an ideal gas, $\text{slope}_P H = V = Nk_BT/P$

$$\begin{aligned} H &= \text{antislope}_P \frac{Nk_BT}{P} \big|_{P_0}^{P} \\ &= Nk_BT(\ln P - \ln P_0) \\ &= Nk_BT \ln \frac{P}{P_0} \end{aligned}$$

supposing we take the antislope over range P_0 to P.

This is a much simpler derivation of the result we got at the end of Note 19.

While none of these state functions is the energy U, small *differences* in them are identical to small differences in energy because they share slopes. This is the basis for their practical value.

If pressure is fixed, for example, as in open chemical reactions (most chemical reactions in basic labs take place at fixed pressures of about one atmosphere) then the four functions that depend explicitly on P simplify because $\Delta P = 0$, and we could write them, for fixed P, without the P:

$$\begin{gathered} H(S,N) \\ G(T,N) \\ X(S,\mu) \\ Z(T,\mu) \end{gathered}$$

Furthermore, for equilibrium (which may nearly be the case for a chemical reaction in progress) $\Delta S = 0$ and we can also drop S from two of these

$$\begin{gathered} H(N) \\ X(\mu) \end{gathered}$$

Of these two, the enthalpy, H, is commonly used (X doesn't even have a name), and this is why college chemistry texts talk about the enthalpy when they want to calculate changes in energy and heat.

On the other hand, if $\Delta S \neq 0$ but $\Delta T = 0$, we would have the other two, $G(N)$ or $Z(\mu)$. *Changes of phase* occur at constant pressure and temperature and the Gibbs function is used to calculate their properties.

In our simulations so far (except for the mixing experiments) volume has been fixed, so we can use similar considerations to select

$$\begin{gathered} U(S,N) \\ A(T,N) \\ \Phi(T,\mu) \\ Y(S,\mu) \end{gathered}$$

The we could further stipulate equilibrium, $\Delta S = 0$, to pinpoint U or Y, or we might stipulate fixed temperature, $\Delta T = 0$, focussing on A and Φ.

We might consider U to be the basic quantity for fixed volume and H to be basic for fixed pressure. Note the relationship between these and the respective "free energies" (so-called because they tell us about energy available for work and not dissipated as heat)

$$\begin{aligned} A &= U - TS \\ G &= H - TS \end{aligned}$$

Some adjectives ($\iota\sigma os$ same)

isobaric	$P =$ const.	$\beta\alpha\rho os$ weight
isochoric	$V =$ const.	$\chi\omega\rho os$ space
		or isovolumetric, isometric
isothermal	$T =$ const.	$\theta\epsilon\rho\mu os$ hot
isentropic	$S =$ const.	or adiabatic: a (not) dia (through) $\beta\alpha\tau os$ passable
equilibrium	$S =$ const.	(because $S =$ max)

Note that each of these functions is *stationary* (i.e., conserved) under different conditions. For example $G(T,P,N)$ does not change for isothermal, isobaric processes on a fixed number of molecules.

The discussion in this Note of local inverses of functions (or "turning them inside-out" in the case of multivariate functions) and of using complementary variables to generate related functions with different local dependences, are generally applicable to any function, although we have applied them here to the thermostatic energy-like functions that depend on variables S and T, P and V, and N and μ.

The next Note will investigate further general ideas, then I'll add what we know about ideal gases and can suppose about non-ideal gases to explore general thermodynamics.

This Note has "thermostatics", not "thermodynamics", in the title just because of the term "local": the math applies only to the physics of *small* changes from equilibrium. (Processes that involve only insignificant departures from equilibrium are called *reversible.*) Although we can arrive at remarkably general results this way, the math does not apply to the process of arriving at equilibrium, for which we have the much more difficult task of knowing the functions themselves, not just their slopes.

As a final note, for future reference, only two of the named thermostatic functions in this Note depend explicitly on entropy, namely $U(S,V,N)$ and $H(S,P,N)$: we will have special uses for the internal energy and the enthalpy.

21. More on multivariate slopes. Useful but not obvious transformations on slopes can be derived from the local invertibility of multivariate functions.

Let's work these out with four variables (since gas laws such as the ideal gas law depend on four variables T, V, P and N). Here are w, x, y and z as (local) functions of each other.

$$z(w,x,y) \qquad w(x,y,z) \qquad x(w,y,z) \qquad y(w,x,z)$$

From the first

$$\Delta z = W_z\Delta w + X_z\Delta x + Y_z\Delta y$$

where I've written W, X, Y and Z for the complementary variabkes, i.e., the slopes, and since we have four functions, I've used a subscript z to distinguish which function we are taking the slope of. Thus

$$W_z = \text{slope}_w z$$

etc.

From the second function

$$\Delta w = Z_w\Delta z + X_w\Delta x + Y_w\Delta y$$

which we can also write in terms of W_z, X_z, Y_z because of the first function

$$\Delta w = \frac{1}{W_z}(\Delta z - X_z\Delta x - Y_z\Delta y)$$

So we have essentially two results from this

$$Z_w = \frac{1}{W_z}$$

and

$$X_w = -\frac{X_z}{W_z}$$
$$Y_w = -\frac{Y_z}{W_z}$$

The first says

$$\text{slope}_z w = \frac{1}{\text{slope}_w z}$$

and this encourages the use of conventional notation for slopes

$$\frac{\mathrm{d}w}{\mathrm{d}z} = \frac{1}{\mathrm{d}z/\mathrm{d}w}$$

(or, since w and z depend on other variables besides each other, this is conventionally written using "partial derivatives"

$$\frac{\partial w}{\partial z} = \frac{1}{\partial z/\partial w}.)$$

The second pair of results are both of the general form

$$A_b = -\frac{A_c}{B_c}$$

i.e.,

$$\text{slope}_a b = -\frac{\text{slope}_a c}{\text{slope}_b c}$$

Conventional notation is a source of confusion here:

$$\frac{\partial b}{\partial a} = -\frac{\partial c/\partial a}{\partial c/\partial b}$$

would tempt us to cancel the ∂c numerators to get the contradiction

$$\frac{\partial b}{\partial a} = -\frac{\partial b}{\partial a}$$

For this reason, conventional notation is forced to add a tag to say what does *not* vary when the different slopes are being taken.

$$\left(\frac{\partial w}{\partial x}\right)_{y,z} = -\frac{(\partial z/\partial x)_{w,y}}{(\partial z/\partial w)_{x,y}}$$

The tags have the effect of making $\partial z/\partial x$, etc., an inviolable unit, not just a fraction. (Or, we could write

$$\text{slope}_{x@y,z} w = -\frac{\text{slope}_{x@w,y} z}{\text{slope}_{w@x,y} z}$$

but this is so far not necessary for the slope$_a b$ notation.)

Because of this possible notational confusion, let's work an example. And since gas laws have motivated this discussion, let's use the ideal gas law $PV = Nk_BT$.

Consider $V = Nk_BT/P$, and translate it $V \to w, P \to x, N \to y$ and $T \to z$:

$$w = yk_Bz/x \qquad\qquad z = xw/(zk_B)$$

Now

$$\text{slope}_z w = yk_B/x \qquad\qquad \text{slope}_w z = x/(yk_B)$$
$$\text{slope}_x w = -yk_Bz/(x^2) \qquad\qquad \text{slope}_x z = w/(yk_B)$$

So

$$\text{slope}_z w = \frac{yk_B}{x} = \frac{1}{\frac{x}{yk_B}} = \frac{1}{\text{slope}_w z}$$

and

$$\text{slope}_x w = -\frac{yk_Bz}{x^2} = -\frac{w}{x} = -\frac{\text{slope}_x z}{\text{slope}_w z}$$

as required.

The use of such transformations becomes apparent with a more sophisticated gas law. The *van der Waals gas law* (normally written for a fixed $N = N_A$) is

$$(P + \frac{a}{V^2})(V - b) = RT$$

or

$$P = -\frac{a}{V^2} + \frac{RT}{V-b}$$

where a and b are constants which depend on the kind of gas, and $R = N_Ak_B$.

Translated, it is hard to write the function $w(x,z)$ but we *can* write

$$x = -\frac{a}{w^2} + \frac{Rz}{w-b} \qquad\qquad z = \frac{1}{R}(x + \frac{a}{w^2})(w - b)$$

For this, slope$_w z$ and slope$_x w$ cannot be found directly, but

$$\begin{aligned}
\text{slope}_z w &= \frac{1}{\text{slope}_w z} = \frac{1}{\frac{1}{R}(x - \frac{a}{w^3}(w-2b))} = \frac{\frac{R}{w-b}}{\frac{Rz}{(w-b)2} - \frac{2a}{w^3}} \\
\text{slope}_x w &= -\frac{\text{slope}_x z}{\text{slope}_w z} = -\frac{(w-b)/R}{(x - \frac{a}{w^3}(w-2b))/R} = \frac{1}{\frac{a}{w^3} - \frac{Rz}{(w-b)^2}}
\end{aligned}$$

(Note the replacement of x in both of the above.)

Alternatively we could use x

$$\begin{aligned}
\text{slope}_z w &= -\frac{\text{slope}_z x}{\text{slope}_w x} = -\frac{\frac{R}{w-b}}{\frac{2a}{w^3} - \frac{Rz}{(w-b)^2}} \\
\text{slope}_x w &= \frac{1}{\text{slope}_w x} = \frac{1}{\frac{a}{w^3} - \frac{Rz}{(w-b)^2}}
\end{aligned}$$

A second set of transformations involves complementary variables and slopes of slopes.

In general, for a function $z(x, y, ..)$,

$$\text{slope}_x(\text{slope}_y z) = \text{slope}_y(\text{slope}_x z)$$

so, if X and Y are the complementary variables of x and y, respectively

$$\Delta z = X\Delta x + Y\Delta y$$

then this implies

$$\text{slope}_x Y = \text{slope}_y X$$

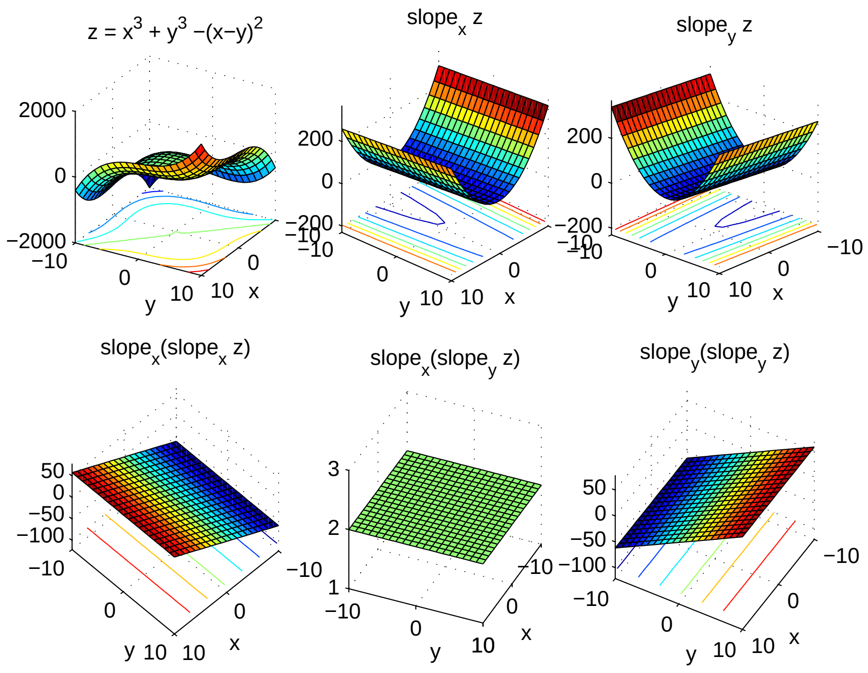
Before we apply this to the thermostatic functions of state we can look at an example to convince ourselves of this first. It is probably a good idea also to visualize slopes and slopes of slopes for multivariate functions while we are thinking about this concrete example.

Here is a function with nontrivial slopes and slopes of slopes, all different.

$$\begin{aligned}
z &= x^3 + y^3 - (x-y)^2 \\
\text{slope}_x z &= 3x^2 - 2(x-y) \\
\text{slope}_y z &= 3y^2 + 2(x-y) \\
\text{slope}_x(\text{slope}_x z) &= 6x - 2 \\
\text{slope}_y(\text{slope}_x z) &= 2 \\
\text{slope}_x(\text{slope}_y z) &= 2 \\
\text{slope}_y(\text{slope}_y z) &= 6y - 2
\end{aligned}$$

We see the equality of $\text{slope}_x(\text{slope}_x z)$ with $\text{slope}_y(\text{slope}_x z)$.

Here are plots of all six different functions, showing them in 3D against both x and y



Now consider the eight state functions, $U(S,V,N), H(S,P,N), A(T,V,N), G(T,P,N), \Phi(T,V,\mu)$ as well as $X(S,P,\mu), Y(S,V,\mu)$ and $Z(T,P,\mu)$. Recall from Note 20 that the complementary variables are

var	S	T	V	P	N	μ
compl	T	$-S$	$-P$	V	μ	$-N$

so we must keep track of the minus signs for S, P and N.

Looking at

$$\Delta U = T\Delta S - P\Delta V + \mu\Delta N$$

we get

$$\begin{aligned}\text{slope}_S P &= -\text{slope}_V T\\ \text{slope}_S \mu &= -\text{slope}_N T\\ \text{slope}_V \mu &= -\text{slope}_N P\end{aligned}$$

Looking at

$$\Delta H = T\Delta S + V\Delta P + \mu\Delta N$$

we get

$$\begin{aligned}\text{slope}_S V &= -\text{slope}_P T\\ \text{slope}_S \mu &= -\text{slope}_N T\\ \text{slope}_P \mu &= -\text{slope}_N V\end{aligned}$$

which has one overlap with what we found from ΔU.

There is a general rule. Break variables S, T, V, P, N, μ into complementary pairs X and X^C, Y and Y^C with X and Y drawn from different sets of complements. Then

$$\text{slope}_X Y = \pm\text{slope}_{Y^C} X^C$$

where the sign is positive if both X^C and Y have the same sign. (T, V, μ are positive; S, P, N are negative.)

What is more, the variables that are fixed for these slopes are either Z or Z^C, where Z is drawn from the third set of complements, other than X or Y; and, of course, Y^C is fixed on the left-hand side and X on the right-hand side. Thus the above expands to

$$\begin{aligned}
\text{slope}_{X@Y^C,Z}Y &= \pm\text{slope}_{Y@X,Z}Y \\
\text{slope}_{X@Y^C,Z^C}Y &= \pm\text{slope}_{Y@X,Z^C}Y
\end{aligned}$$

But this explicit fixing is not needed in the $\text{slope}_a b$ notation.

So, for example, we can construct from this

$$\text{slope}_S P = -\text{slope}_V T$$

which we already worked out above, or

$$\text{slope}_T N = -\text{slope}_\mu T$$

a new relationship (which can be obtained from either Φ or Z).

Four of these relationships, derived from U, H, A and G with N fixed, are called the *Maxwell relations.*

$$\begin{aligned}
\text{slope}_S P &= -\text{slope}_V T \\
\text{slope}_S V &= -\text{slope}_P T \\
\text{slope}_T P &= -\text{slope}_V S \\
\text{slope}_T V &= -\text{slope}_P S
\end{aligned}$$

are frequently used.

22. Work and heat. We can extract some useful work from our gas by letting it expand. By pushing on a piston the expanding gas can move it and this motion could drive wheels to pump water or grind grain or move a steam locomotive.

We'll define work as useful energy. (Or maybe, better, as *used* energy.) This definition depends on what we consider to be useful and so is certainly not a state function of the gas the way the internal energy and all its variants in Note 20 do.

The internal energy of our gases is (a little bit generalized from the ideal, monatomic gas, but not much)

$$U(S, V, N)$$

Of these variables we'll focus on the volume V as our hope for providing useful work. (We'll keep N, the number of molecules, fixed, so the only other'variable is S and we'll come to that.)

When our gas expands by ΔV the change in internal energy is

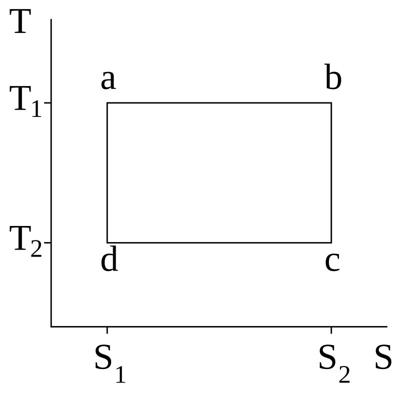
$$\Delta U = \text{slope}_V U \Delta V = -P\Delta v$$

from Note 20. This change is an energy too, and we can identify it with the work done *on* the gas. (If the gas expands, ΔV is positive and ΔU is negative: positive work is actually done *by* the gas.)

To get more work out of the gas, we'll have to compress it back again (do work *on* the gas) and then cycle 'round again. Can we do this in such a way that the work doesn't all cancel, but that we get a net benefit?

This deends on the temperature at which we do the expansion and compression: if the temperatures for the two processes are different we actually can get net work.

Here is a picture of such a cycle



The work we get per cycle is the area of the rectangle

$$(T_1 - T_2)(S_2 - S_1)$$

(This is at least plausible since

$$TS = Tk_B \times \text{ignorance}$$

is also an energy. We'll now show that this area is indeed the extracted work.)

Firat let's derive the relationship mentioned in Note 18

$$pV^\gamma = \text{const.}$$

for a gas that does not lose energy except by doing work by volume change. We know two things. First, above

$$\Delta U = \text{slope}_V u \Delta V$$

with

$$\text{slope}^V U = -p$$

The second thing we know, from Note 18, is that

$$PV = (\gamma - 1)U$$

where $\gamma - 1$ is just a way of writing the factor, which differs for different gases. For a monatomic ideal gas we had

	2D	3D
$\gamma - 1$	1	2/3
γ	2	5/3

and other values arise from the equipartitin of energy for polyatomic gases.

From this

$$\begin{aligned}(\gamma - 1)\Delta U &= \text{slope}_V U \Delta V + \text{slope}_P U \Delta P \\ &= P\Delta V + V\Delta P\end{aligned}$$

Putting the two equations for ΔU together we get

$$(1 - \gamma)P\Delta V = P\Delta V + V\Delta P$$

or

$$0 = \gamma P\Delta V + V\Delta P$$

Let's compare this with

$$\begin{aligned}\text{slope}PV^\gamma &= \gamma P\text{slope}V^\gamma + V^\gamma\text{slope}P \\ &= \gamma PV^{\gamma-1}\text{slope}V + V^\gamma\text{slope}P\end{aligned}$$

(I've used slope for slope_x, the slope with respect to *any* variable—we don't care which.)

If PV^γ is constant, its slope is zero, and so, too,

$$0 = \gamma P\text{slope}V + V\text{slope}P$$

But what we know is the same thing

$$\begin{aligned}0 &= \gamma P\Delta V + V\Delta P \\ &= \gamma P\text{slope}_x V\Delta x + V\text{slope}_x P\Delta x\end{aligned}$$

i.e.,

$$0 = \gamma P\text{slope}V + V\text{slope}P$$

So, for many gases, and certainly ideal monatomic gases

$$PV^\gamma = \text{constant}.$$

For the 2D monatomic gas, $\gamma = 2$.

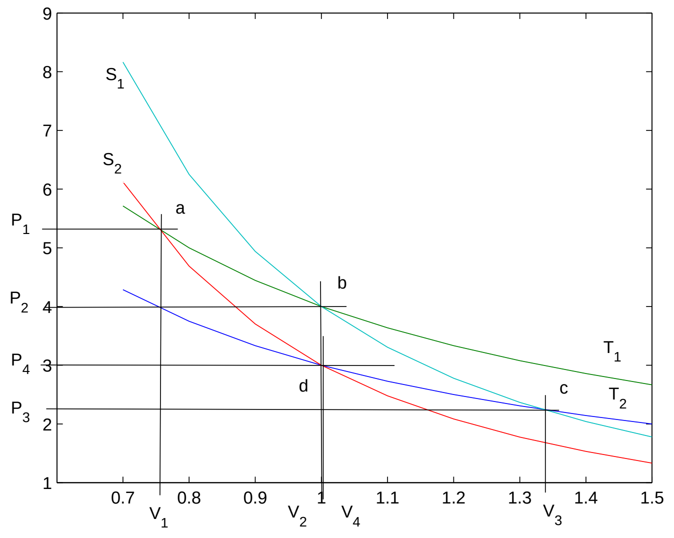
Now specialize to ideal gases for which, in addition,

$$PV = Nk_BT$$

and we have two relationships to work with.

The French engineer, Nicolas Léonard Sadi Carnot, in 1824 figured out how to get the most work possible out of a cycle such as the one we drew earlier. We'll work with the above two equations, although Carnot had available neither ideal gases nor conservation of energy.

We can show that the following "P–V" diagram is equivalent to our earlier "T–S" diagram.



where ab and cd are segments of two different isoterms corresponding to temperatures T_1 and T_2 respectively, and bc and da are segments of two different isentropes corresponding to entropies S_1 and S_2 respectively.

(We will see soon that the entropies are indeed fixed along bc and da.)

The isotherms give P vs. V as

$$P = Nk_BT_1/V \qquad \text{and} \qquad P = Nk_BT_2/V$$

and the isentropes give P vs. V as

$$P = \text{const}_1/V \qquad \text{and} \qquad P = \text{const}_2/V$$

At the corners we have the following values

a	P_1	V_1	T_1	S_1
b	P_2	V_2	T_1	S_2
c	P_3	V_3	T_2	S_2
d	P_4	V_4	T_2	S_1

where we must show that there are only the two values for S.

Given the connections between isotherms and isentropes at the four corners we can see (using $\gamma = 2$ for 2D ideal monatomic gas)

$$\frac{(Nk_BT_1)^2}{P_1} = \text{const}_1 = \frac{(Nk_BT_2)^2}{P_4}$$

and

$$\frac{(Nk_BT_1)^2}{P_2} = \text{const}_1 = \frac{(Nk_BT_2)^2}{P_3}$$

We also have

$$\frac{V_4^2}{V_1^2} = \frac{P_1}{P_4} = \frac{T_1^2}{T_2^2} = \frac{P_2}{P_3} = \frac{V_3^2}{V_2^2}$$

where I've worked out from the middle: the relationships between T^2 and P come from const$_1$ above (going to the left) and from const$_2$ above (going to the right); the relationships between P and V^2 come from $PV^2 = \text{const}_1$ (going to the left) and from $PV^2 = \text{const}_2$ (going to the right).

Thus

$$T_2V_4 = T_1V_1$$

and

$$T_2V_3 = T_1V_2$$

Now let's calculate the entropies at the four corners using the 2D Sackur-Tetrode equation of Note 19

$$S = 2Nk_B + Nk_B \ln CTV$$

where V replaces U since $U = Nk_BT$ for a 2D ideal gas (Note 19) and where C absorbs all the constant terms

$$C = \frac{2\pi mk_B}{h^2N}$$

Then, labelling the entropies by the four corners,

$$\begin{aligned} \frac{S_a}{Nk_B} - 2 &= \ln CT_1V_1 \\ \frac{S_b}{Nk_B} - 2 &= \ln CT_1V_2 > \ln CT_1V_1 \end{aligned}$$

$$\begin{aligned}\frac{S_c}{Nk_B}-2 &= \ln CT_2V_3=\ln CT_1V_2\\ \frac{S_d}{Nk_B}-2 &= \ln CT_2V_4=\ln CT_1V_1\end{aligned}$$

So

$$S_a=S_d<S_b=S_c$$

and we may call them S_1 and S_2 as we did.

This also establishes that PV^γ = constant gives isentropes, at least for this case: unchanging entropy is equivalent to loss or gain of internal energy only through doing or receiving work.

Now let's show that the work done is the negative of the area drawn in the P–V diagram and that this area equals $(T_1-T_2)(S_2-S_1)$, the area of the rectangle in the T–S diagram.

For a small change in volume, ΔV, we had that the work done on the gas is $-P\Delta V$. So the total work is

$$-\text{antislope}_V P$$

with $P(V)$ a function of V. In the P–V diagram the (positive) antislope comes in four parts

$$\begin{aligned}&\text{antislope}_V\frac{Nk_BT}{V}|_a^b\\ +\ &\text{antislope}_V\frac{\text{const}_1}{V^2}|_b^c\\ +\ &\text{antislope}_V\frac{Nk_BT}{V}|_c^d\\ +\ &\text{antislope}_V\frac{\text{const}_1}{V^2}|_d^a\end{aligned}$$

Each of these represents the area below the corresponding curve, and since V decreases from c to a, the latter two are negative and so subtracted, leaving the area bounded by the curves, as I said.

The isotherm antislopes are

$$\text{antislope}_V\frac{Nk_BT_1}{V}|_a^b=Nk_BT_1\ln\frac{V_2}{V_1}$$

and

$$\text{antislope}_V\frac{Nk_BT_2}{V}|_c^d=Nk_BT_2\ln\frac{V_4}{V_3}=Nk_BT_2\ln\frac{V_1}{V_2}$$

using the $V_4^2/V_1^2=V_3^2/V_2^2$ relationship above.

The isentrope antislopes are

$$\text{antislope}_V\frac{\text{const}_2}{V^2}|_b^c=-\frac{\text{const}_2}{V}|_b^c=\frac{(Nk_VT_2)^2}{P_3VC_3}-\frac{(Nk_VT_1)^2}{P_2VC_2}$$

and

$$\text{antislope}_V\frac{\text{const}_1}{V^2}|_d^a=-\frac{\text{const}_1}{V}|_d^a=\frac{(Nk_VT_1)^2}{P_1VC_1}-\frac{(Nk_VT_2)^2}{P_4VC_4}$$

Since $P_1V_1=Nk_BT_1=P_2V_2$ and $P_3V_3=Nk_BT_2=P_4V_4$, these two cancel and their sum is zero.

The isotherm antislopes sum to

$$Nk_B(T_1-T_2)\ln\frac{V_2}{V_1}$$

and this is exactly equal to $(T_1 - T_2)(S_2 - S_1)$, from the 2D Sackur-Tetrode equation for S.

So the work performed is the negative of the enclosed area in either diagram. (Or the result of doing the antislopes counterclockwise, instead of clockwise as we did them.)

Now Carnot was interested in building a machine, even if a theoretical one, and he did not do this particular math. How does it all turn into a machine?

Let's imagine two large "thermal baths" at temperatures T_1 (higher) and T_2 (lower), respectively: a thermal bath (or "heat reservoir") is just an object large enough that what we are about to do won't noticeably change its temperature.

Let's also imagine a cylinder full of gas with a moveable piston to allow the gas to expand or to compress it. The cylinder is itself moveable so that we can take it back and forth between the two reservoirs. When it is in contact with the reservoir, the cylinder allows the gas to remain in thermal equilibrium at the temperature of the reservoir (isothermal). When it is in transit between the reservoirs, the cylinder allows no energy to enter or leave the gas save through the work of moving the piston (isentropic).

The work cycle corresponds to the following sequence of operations (Feynman [FLS64, §44-3] shows pictures).

ab Cylinder sits on T_1 reservoir and pressure on the piston is relaxed from P_1 to P_2, allowing the gas to expand from V_1 to V_2, all at T_1.

bc Cylinder is in transit from T_1 to T_2 and at the same time the piston pressure is dropped from P_2 to P_3 allowing the gas to expand from V_2 to V_3 and thus (since the internal energy must drop) to drop temperature from T_1 to T_2.

cd Cylinder sits on T_2 reservoir and pressure on the piston is increased from P_3 to P_4, compressing the gas from V_3 to V_4, all at T_2.

da Cylinder is in transit from T_2 to T_1 and at the same time the piston pressure is increased from P_4 to P_1 compressing the gas from V_4 to V_1, thus increasing the temperature from T_2 to T_1, just as the cylinder arrives at bath T_1.

At this point, Carnot did his really inspired thing. He let the machine be *reversible*. This means that none of the work done was dissipated by friction (a practically unavoidable way of turning work back into heat). And it means that heat could flow either way at any point in the operation.

Reversible heat flow gets around Carnot's true stroke of genius, the *Second Law of thermodynamics.* (He didn't call it that because the first law, which is essentially energy conservation, was not yet formulated.)

The second law says *Heat cannot of itself pass from one body to a hotter body* [FS64, First and Second Law], which we can agree from experience top be true. What is remarkable is what can be inferred from it.

For example, it implies that "heat cannot be taken in at a certain temperature and converted into work with *no other change* in the system or its surroundings" [FLS64, §44-2].

Here is the deduction. If we could extract work from a heat reservoir at temperature T_2, say, then we could convert that work into heat at a higher temperature, T_1, say, by friction at T_1 (e.g., by rubbing the T_1 reservoir). But that would contradict the second law.

This means that Carnot's machine needs at least two heat reservoirs at different temperatures $T_1$and T_2.

It also means that Carnot's machine might not be reversible because the work extracted from the reservoir T_1 (in Carnot's machine we extract work from the higher temperature, not as in the above deduction) may lower the gas temperature and so, if reconverted to heat in the gas, will not flow back to the reservoir at the higher temperature T_1.

Carnot gets around the last by taking things so *slowly* that the temperature differences are miniscule and the argument does not apply. (Think of fluctuations.) Moreover, the slowness also keeps friction negligible.

With this caveatof sufficient slowness, Carnot's machine is reversible.

So what? Reversibility means we can run the whole cycle backwrds (using work to extract heat from T_2 to put it into T_1—a "heat pump": this does not violate the second law because it requires work from the surroundings). That means nothing is lost, which in turn means the process is as efficient as possible.

But the efficiency of *converting* heat to work is not 100% even in this best possible scenario. To see this, we will finally have to define "heat".

First, like work, heat is not a state function. We cannot assign a total quantity of heat to a body. This was a stumbling block for the now obsolete caloric theory of heat in which heat was taken to be a "fluid" contained in bodies, which can flow between them and which moreover was thought to be conserved.

We cannot deal with absolute heat levels but only with heat *transferred* in a given process.

In Carnot's reversible heat engine, no heat is transferred in the isentropic operations, and all the work produced or consumed by the isothermal operations is due to heat received from or given to the reservoirs.

We saw that the total work was

$$Nk_BT_1 \ln\frac{V_2}{V_1} - Nk_BT_2 \ln\frac{V_2}{V_1}$$

And we can break this into the heat transferred from the T_1 reservoir

$$Q_1 = Nk_BT_1 \ln\frac{V_2}{V_1}$$

and the heat transferred to the T_2 reservoir

$$Q_2 = Nk_BT_2 \ln\frac{V_2}{V_1}$$

Then the work is

$$W = Q_1 - Q_2$$

and the *efficiency* of the conversion is defined in terms of the heat transferred from the hotter reservoir. (That is, we draw heat Q_1 from the T_1 reservoir in order to do work W. But we cannot use it all and must give up heat Q_2 to reservoir T_2.)

$$\text{efficiency} = \frac{W}{Q_1} = \frac{Q_1 - Q_2}{Q_1} = \frac{T_1 - T_2}{T_1}$$

Note that this depends *only* on the temperatures.

So if $T_1 = 400°$K and $T_2 = 300°$K, the efficiency is 25%. Remember, this is for the *best possible* engine, the reversible Carnot engine. (Feynman [FLS64, §44-4] shows that our results, based on ideal monatomic gas—in 2D, even—are valid for *any* reversible engine of any design using any kind of gas. This is the power of Carnot's idea.)

Thus $(T_1 - T_2)/T_1$ is the highest efficiency that can in principle be achieved by *any* heat engine: a steam locomotive, your car, or any engine that extracts work from the difference of two temperatures.

We can identify heat with entropy change. For example in the isothermal operations the change in the internal energy of the gas is zero. With internal energy $U(S, V)$—remember we're keeping N fixed so it can be ignored as a variable

$$\begin{aligned} 0 = \Delta V &= \text{slope}_S U \Delta S + \text{slope}_V U \Delta V \\ &= T\Delta S - P\Delta V \end{aligned}$$

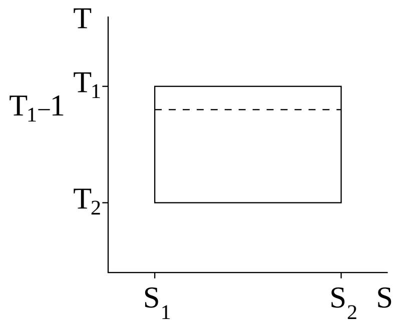
So, since $-P\Delta V$ is work, the heat transferred can be taken to be $\Delta Q = T\Delta S$.

If the temperature is not constant but is a function $T(S)$ then the heat transferred by a change from S_1 to S_2 is the

$$\text{antislope}_S T \mid_{S_1}^{S_2}$$

But we often do not know $T(S)$ whereas the gas laws do give us $P(V)$, which we can use if applicable.

We can identify entropy with the *unavailability* of energy in the following sense. In the T–S diagram



if we lower the upper temperature by 1°K, energy $S_2 - S_1$ has become unavailable to do work.

That is the meaning of the rearrangement of $\Delta Q = T\Delta S$,

$$\Delta S = \frac{\Delta Q}{T}$$

When you sit by a campfire or a warm radiator you feel heat. We can relate this colloquial meaning of "heat" to entropy by imagining two gas containers side by side but initially at different temperatures. If they are ideal gases we can show explicitly that the entropy increases.

$$\begin{aligned} S_1 &= N_1 k_B (2 + \ln C \frac{T_1 V_1}{N_1}) \\ S_2 &= N_2 k_B (2 + \ln C \frac{T_2 V_2}{N_2}) \\ S_3 &= (N_1 + N_2) k_B (2 + \ln C \frac{T_3 (V_1 + V_2)}{N_1 + N_2}) \end{aligned}$$

where, from conservation of (kinetic) energy

$$T_3 = \frac{N_1 T_1 + N_2 T_2}{N_1 + N_2}$$

and I've written the 2D Sackur-Tetrode equation with

$$C = \frac{2\pi m k_B}{h^2}$$

To simplify the math, let's suppose $N_1 = N_2 = N$. Then

$$\begin{aligned}
\Delta S &= S_3 - (S_1 + S_2) \\
&= 2Nk_B \ln C\frac{(T_1+T_2)(V_1+V_2)}{4N} \\
&\quad -Nk_B \ln C\frac{T_1V_1}{N} \\
&\quad -Nk_B \ln C\frac{T_2V_2}{N} \\
&= Nk_B(\ln\frac{(T_1+T_2)^2}{4T_1T_2} + \ln\frac{(V_1+V_2)^2}{4V_1V_2})
\end{aligned}$$

after all the simplifying.

You can show that $\frac{(a+b)^2}{4ab}$ is never less than 1 (and equals 1 when $a = b$) so the logarithms are never negative. Thus $\Delta S > 0$ and the entropy increases unless $T_1 = T_2$ and $V_1 = V_2$, i.e., the gases were already in equilibrium.

We cannot work out the *amount* of heat using $\Delta Q = T\Delta S$ because the transfer takes place at different temperatures, ranging from T_2 to T_3 and from T_1 to T_3. But we can calculate the heat transferred directly using $PV = (\gamma - 1)U$ and reasoning that, since no work is done, ΔQ from the hotter gas equals ΔU for that reservoir. We'll use $\gamma = 2$ again, and an ideal gas $PV = Nk_BT$.

(That is the heat given off by gas at T_1. For conservation of energy, this equals the heat absorbed by gas at T_2 and the total change in U is zero.)

For the gas initially at T_1 and finally at T_3

$$\begin{aligned}
\Delta Q = \Delta U &= N_1k_BT_1 - N_1k_BT_3 \\
&= \frac{N_1N_2}{N_1+N_2}k_B(T_1 - T_2)
\end{aligned}$$

and this is the heat that's transferred.

This relationship between heat and temperature

$$\Delta Q = \frac{Nk_B}{\gamma - 1}\Delta T$$

(putting back the general γ) enters the province of specific heats. Feynman [FLS64, §§40-5,40-6] tells how the disagreement between theory and experiment led Maxwell as early as 1859 to the failure of pre-quantum physics that was finally fixed by quantum physics.

The *specific heat* is the rate of increase of energy with temperature and so it can be written as

$$\text{slope}_T U$$

except that $U(S, V)$ does not explicitly depend on temperature T.

So we can use the Legendre transformation of Note 20 to swap T for S and get the Helmholtz free energy $A(T, V) = U - TS$ and

$$\text{slope}_T U = \text{slope}_T A = C_V$$

But we can also use the Gibbs free energy, which swaps P for V, $G(T,P) = A + PV$ so

$$\text{slope}_T U = \text{slope}_T G = C_P$$

The difference between these two is made clear if we explicitly write the fixed variables, respectively volume V and pressure P:

$$\begin{aligned} C_V &= \text{slope}_{T@V} U = \text{slope}_{T@V} A \\ C_P &= \text{slope}_{T@P} U = \text{slope}_{T@P} G \end{aligned}$$

Furthermore

$$\begin{aligned} C_P &= \text{slope}_T G \\ &= \text{slope}_T (A + PV) \\ &= \text{slope}_T (A + Nk_BT) \\ &= \text{slope}_T A + Nk_B \\ &= C_V + Nk_B \end{aligned}$$

where ths switch from PV to Nk_BT assumes an ideal gas, and where we don't need to specify @ when finding slope_T of A or G because the respective other variables are V and P and these are fixed implicitly. We *do* need the @ for $\text{slope}_T U$ because it's not clear what the fixed variable is to be since U is not an explicit function of T.

The Carnot relationship between work and heat can be written

$$\Delta A = \Delta Q \frac{\Delta T}{T}$$

and if these quantities are small enough the Carnot cycle has the shape of a trapezoid in P–V space, whose area is the same as that of the rectangle

$$\Delta W = \Delta P \Delta V$$

This relationship can now be written variously and the limits taken:

$$\Delta Q = T \frac{\Delta P}{\Delta T} \Delta V \to T \text{slope}_T P \Delta V$$

or

$$\Delta Q = T \frac{\Delta V}{\Delta T} \Delta P \to T \text{slope}_T V \Delta P$$

These are useful if we are working with state functions in which V or P, respectively, are variables.

Emergence. The ideal gas law relates quantities which are removed from the underlying mechanics of bouncing molecules conserving energy and momentum. P and T are only averages of these quantities—in the end, averages of kinetic energy, which is why they are related.

It could be said that P and T are *emergent* quantities, resulting only from *aggregates* of the bouncing molecules. A single molecule does not have a pressure or a temperature (or even a volume, in some sense). Furthermore, because there are so many molecules in a visible-sized sample of gas, the $\sqrt{N}$ law (Part I Excursion *Root N law*) says $PV = Nk_BT$ is a very precise relationship even for arbitrary samples of gas: these averages take on the role of precise quantities in their own right.

The idea is that there are two levels of discourse, each with quite different laws. The "lower" level is the billiard-ball physics of bouncing molecules. The "upper" level is the ideal gas and its law.

The argument so far, in terms of P, T, etc., may not yet persuade us that the upper level can show behaviour which is utterly new. We *can* imagine a single molecule having a pressure or a temperature (or a volume or a density) because, after all, we came up with these quantities by averaging over many molecules. So for true emergence we must look further.

Entropy is a truly emergent quantity. It is inherent in a multiplicity of bouncing molecules and has no meaning at all for a single molecule.

Without violating the "lower" law of energy conservation, entropy introduces a major new concept—the dissipation of unavailability of energy. This is a truly emergent law, making the "upper" level more than the sum of its "lower"-level parts. It leads to a new mode of thinking based not on the (conserved) internal energy U but on the never-increasing and often diminishing free energies, A (for fixed volue) and, most used, G (for fixed pressure).

The most vexing property of entropy is its irreversible increase (apart from fluctuations) in isolated systems. This appears to contradict the reversibility of the laws of the molecular level below it. It is liberating, though, to think of the irreversibility as emergent rather than as contradiction.

The possibility of emergent quantities and laws completely undoes any deterministic extrapolation of science, and even the possibility of a "theory of everything". As far as we know, for example, courage, responsibility or free will are emergent at the level of "moral person" and go above and beyond any deterministic entanglement of molecules.

Part of the copnceptual difficulty of thermodynamics is surely its flat contradiction of the philosophies of determinism and the reducibility of higher levels to lower. Being free of such philosophy is liberating, too.

It is appropriate to conclude this Note with the First Law of thermodynamics, since it was formulated, as the law of conservation of energy, after the second law. Many statements of the First Law (the simplest being *Heat is work and work is heat* [FS64, First and Second Law]) must be hedged.

Work can indeed be converted entirely to heat. Both are energy and the work-energy that dissipates equals in magnitude the heat-energy that results.

The second law, however, tells us that not all heat can be converted to work—unless we consider "heat" to be only and exactly that disordered energy that *can* be extracted as work, but this is not consistent with our above reasoning in which work was the difference between the heat in, at the higher temperature, and the heat out, at the lower temperature.

23. Correlation.
24. Collision theory.
25. Mobility, diffusivity and Brownian motion.
26. Three potentials and dissipation.

27. Active transport and biochemistry.
28. Combined transport.
29. Phase transitions.
30. Phase transitions in random graphs.
31. Point-to-point resistance in a network.
32. Van der Waals.
33. Sublimation.
34. Ferromagnets.
35. Particle individuality and Bose-Einstein condensation.

II. **The Excursions**

You've seen lots of ideas. Now *do* something with them!

1. On keeping with advice to have at least three different books open on any difficult topic, I can say that my principal sources for the next three Parts have been Robertson [Rob93], Wannier [Wan87] and Feynman [FLS64, Ch.39–46]. All three are disorderly, however. Feynman is, as usual, accessible, and after you have mastered the present Book you should find Robertson and Wannier accessible in places. I hope I have saved you much of the work of ordering the subject into a narrative, but I am unable to save you assimilating for yourself the strange connections between ignorance and entropy (Part II on equilibrium and theromstatics), between autocorrelation and transport (Part III on linear thermodynamics)

2. a) Show that a 27 meV helium molecule is moving at 1.14 nm/ps, i.e., 1.14 km/sec.
   b) What is the momentum in meV-ps/nm for a 1 km/sec helium molecule?

3. Explore the invocations (Note 12)

   ```
       molCollWallCountBot([1;1],1,1,[-20,20,-10,10],0.05,[19.95,9.95],0)
       molCollWallCountBot([1;1],1,1,[-20,20,-10,10],0.05,[18.00,9.95],0)
       molCollWallCountBot([1;1],1,1,[-20,20,-10,10],0.05,[19.95,8.00],0)
   ```

   a) Draw the configurations these inputs describe and guess what the output will be.
   b) Execute `molCollWallCountBot()` by hand and see if you get the same results.
   c) Run `molCollWallCountBot()` in MATLAB with these inputs and compare with (a) and (b)

4. Explore the invocations (Note 12)

   ```
               [p1,p2] = moleculeCollide(4,4,[1;1],[0;0],0,[0.1;0]
               [p1,p2] = moleculeCollide(4,4,[1/2;1],[-1/2;0],0,[0.1;0]
   ```

   a) Draw the configurations these inputs describe and guess what the output will be.
   b) Execute `moleculeCollide()` by hand and see if you get the same results.
   c) Run `moleculeCollide()` in MATLAB with these inputs and compare with (a) and (b)

5. Explore the invocations (Note 12)

   ```
   [t,c] = molCollWallWhen([−0.05;−0.05],41.44*[1;1],4,0.05,[−20,20,−10,10])
   [t,c] = molCollWallWhen([−14.95;−0.05],41.44*[1;1],4,0.05,[−20,20,−10,10])
   ```

   a) Draw the configurations these inputs describe and guess what the output will be.
   b) Execute `molCollWallWhen()` by hand and see if you get the same results.
   c) Run `molCollWallWhen()` in MATLAB with these inputs and compare with (a) and (b) item Explore the invocations (Note 12)

   ```
     [t,c1,c2] = moleculeCollWhen([−10.1;0],[0;0],41.44*[1;1],4,4,0.05,0.05)
     [t,c1,c2] = moleculeCollWhen([−10;0.1],[0;0],41.44*[1;1],4,4,0.05,0.05)
   ```

   a) Draw the configurations these inputs describe and guess what the output will be.
   b) Execute `moleculeCollWhen()` by hand and see if you get the same results.
   c) Run `moleculeCollWhen()` in MATLAB with these inputs and compare with (a) and (b)

6. Why does the running time of `moleculeCollWhen()` in Note 12 depend on the square of the number N of molecules? What is the exact dependence, theoretically? Can you observe it by running `moleculeCollWhen()` for different values of N?

7. Show that a molecule at 40°C can be expected to have energy 27 meV.

8. In Note 12, how can you set `startRange` to make `moleculeInit` launch a set of molecules with strictly deterministic initial positions and momenta? What would the consequence of this be?

9. Explore the invocation (Note 12) `moleculeInit(10,300,[0,0,5,0,0,40,50],0.05,4)`
a) How many rows will there be in the resulting `molData`? What values will the 2nd, 5th and 6th columns have? What can you say about the range of values in the 1st and 2th columns? How should the values in the 3rd and 4th compare with each other?
b) Run this invocation in MATLAB and check your answers to (a).

10. **First simulation.** Explore the invocation `molData = collSim(10,300,0.05,4,[20;10])` (Note 12) How long does it take to simulate 100 collisions of 10 molecules a) simulation time? b) elapsed time? How do the columns of the resulting `molData` compare with the initial `molData`, especially c) cols 1, 2? d) cols 3, 4?
e) Write a function `molDataHisto(molData)` which plots histograms of the resulting momenta (`molData(:,3), molData(:,4)`) and energies (`molData(:,7)`) in buckets of width 10. Run it on `molData` from `collSim()` above. Discuss the plots you get.
f) Modify `molDataHisto()` to add `molData(:,8)` initially holding the magnitudes $\sqrt{p_x^2 + p_y^2}$ of the momenta.

11. **Doves and hawks.** (A kinetic theory of conflict, 1.) A simulation by Maynard Smith and Price [SP73] of conflicts among five different types within a population can be simplified [HL11] to two different types with the following characteristics in disputes over a resource worth 10 arbitrary units.
A *dove* will always give in and so will share the resource with another dove (score: 5 each) or yield it to a hawk (score: 0).
A *hawk* will always fight and so will take the resource from a dove (score 10) and either will take it from another hawk (score 10), or, if hey loses the fight with that hawk, will suffer damage costing, say, 30 of those units.
a) Write a simulation `[doves,hawks] = doveHawk(D,H,G,L)` which, given the number of doves `D` and the number of hawks `H` in the population, and scores `G` for gain (say, 10) and `L` for loss (say 30), returns the minimum, average and maximum scores for all the `doves` and for all the `hawks`.
Note that deciding "conflicts", i.e., who encounters whom over a resource, is no longer deterministic as it is for molecules but requires `rand()` to select each of the two parties from the population. How do you ensure that an individual does not conflict with hemself?
Before you write the rest of the simulation, ensure, by accumulating and displaying a matrix of encounters, that your randomizing works and omits self-conflicts. Use a small population for this test because the matrix will be `popSize` × `popSize`.
Deciding who wins the fight, if there is one, also requires `rand()`: I made the chances 50-50. My simulations showed that doves generally come out ahead, unless there are very few hawks. Here's what happened in 161051 iterations.

```
[doves,hawks] = doveHawk(1000,1000,10,30)
doves =  1.9211    2.4975    3.1046
hawks = -3.9535    0.0065    4.0940
[doves,hawks] = doveHawk(1000,500,10,30)
doves = 2.8333    3.3273    3.8265
hawks = 0.1075    3.3410    6.3793
[doves,hawks] = doveHawk(1000,200,10,30)
doves = 3.7549    4.1657    4.5217
hawks = 4.8548    6.6204    8.2671
[doves,hawks] = doveHawk(1000,100,10,30)
```

```
doves = 4.2553    4.5370    4.7810
hawks = 6.7213    8.1673    9.2908
```

(Here, `doves` and `hawks` return the minimum, average and maximum statistics of counts made on the individual scores of doves and hawks, respectively, taken after the individual counts are divided by the number of encounters that individual has had. These are the values that can be compared with the theory coming up.)
Obviously a single hawk will always win. Speculate on strongmen and tyrants, or on corporations, or on the government, or the law, as the single hawk.
What happens when the populations are all reduced 100-fold?
What happens when the relative loss is changed from 30/10?
b) Show that the *expected* scores for large populations where the probability of being a dove is p and the probability of being a hawk is q are, for the three ratios of doves to hawks simulated above,

p	q	dovesAvg	hawksAvg
1/2	1/2	2.5	0
2/3	1/3	$3.\overline{3}$	$3.\overline{3}$
5/6	1/6	$4.1\overline{6}$	$6.\overline{6}$
10/11	1/11	4.5455	8.1818

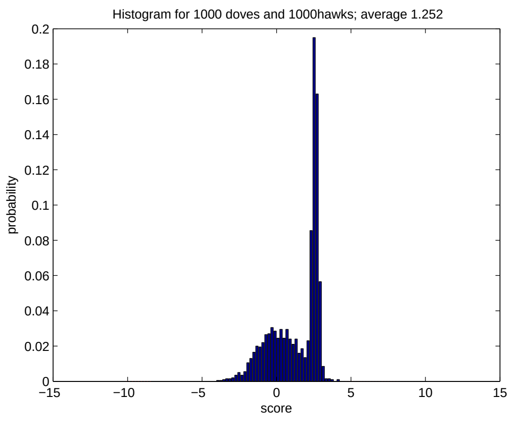
Hint. Here are the scores that you would get if you were dove or hawk (rows) up against dove or hawk (columns). Multiply by p and q respectively and sum.

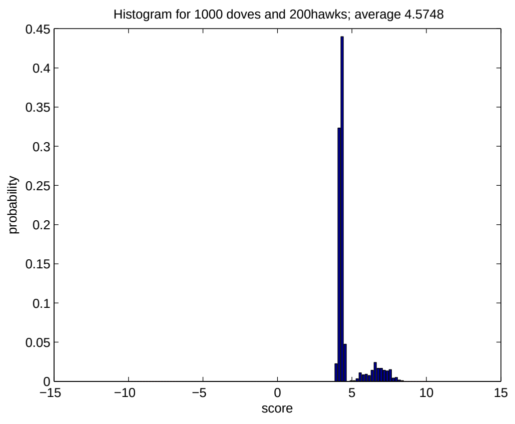
	doves(p)	hawks(q)
dove	5	0
hawk	10	-10

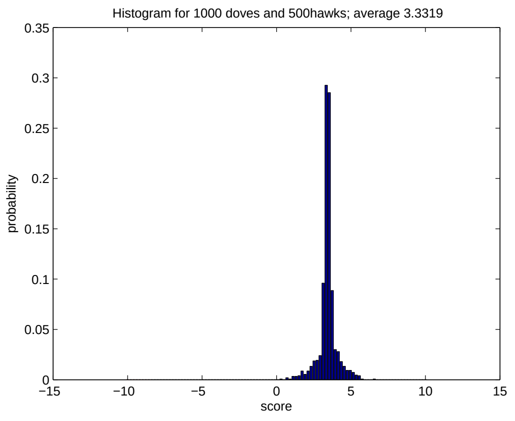
Note that hawks do not do better than if they constitute a third or less of the population.
c) Re-do this theory for small populations, say 10:10, 10:5, 10:2. What is the effect of excluding self-conflicts?

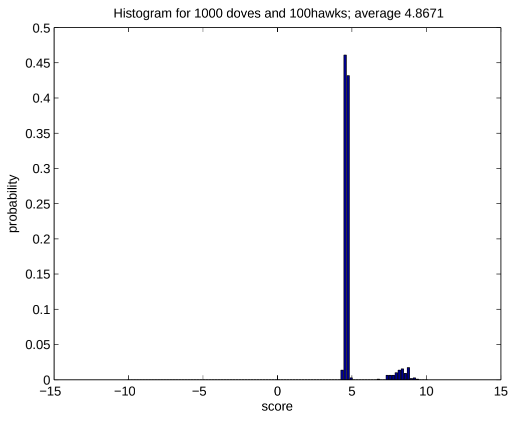
p	q	dovesAvg	hawksAvg
10/20	10/20	2.25	0.5
10/15	5/15	3	4
10/12	2/12	3.75	7.5

d) What happens when there are no hawks? Compare the population mean scores.
e) Consider that each iteration adds 10 to the total resources of the society, less any "dissipation" caused by fighting, and for large populations (such as the first simulations above with 1000 doves) calculate what the society should gain (or lose) in the simulation (`D*doves(2) + H*hawks(2)`).
f) Modify your program to display the *distributions* of scores per encounter. Here are some results. Can you reproduce them?









Relate these distributions to the ranges reported in (a). What will happen if there are 1000 doves and no hawks?
g) Show, for the gains of 10 and possible losses of 30 used above, and for proportion p of doves in the population (and hence $q = 1-p$ hawks), that the higher moments of the distributions of scores for doves and hawks respectively, after centralizing them on the means $5p$ and $10(p-q)$ respectively are

moment	doves	hawks
2	$25pq$	100
4	$625p(1-4p+6p^2-3p^3)$	10000

with all odd moments zero. What happens for $p = 0, 1/2, 1$?
Use Note 6 of Part I to show that neither distribution is normal.
h) The simulation so far has imagined unlimited resources, since every iteration adds a score of 10 to the population. What is the average score after N iterations? What is a simple way to report on fixed resources? What is realistic and what unrealistic about this fix?

12. **Dominance.** (A kinetic theory of conflict, 2.) The previous Excursion, *Doves and hawks*, assumes static characteristics for the members of a population. How do such characteristics develop? Let's take the zoological concept of *dominance* and define it for each individual as the proportion of fights that individual wins. Then we can assign a probability of fighting, as opposed to yielding, equal to the individual's dominance, and, if both decide to fight, the chance of winning is determined by the dominances of the two combatants.
(We can think of dominance as a generalization of the previous Excursion, giving a spectrum between doves and hawks, with doves having dominance 0 and hawks having dominance 1.)

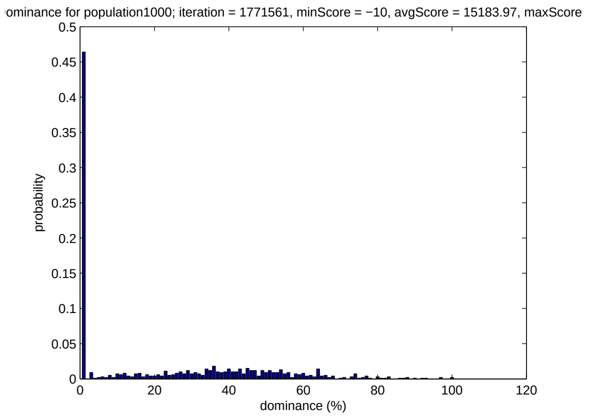
If it comes to a fight between left, with dominance ℓ, and right, with dominance r, the chances are the following.

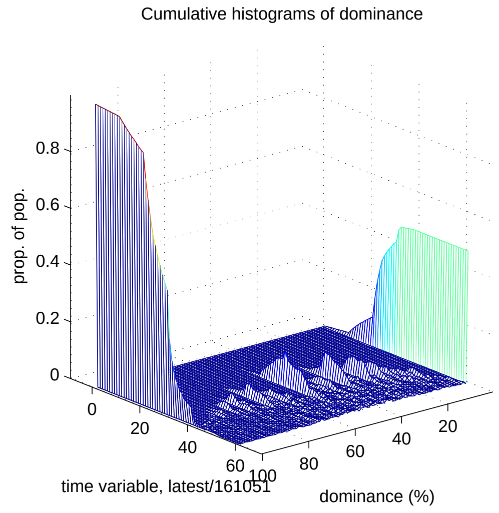
both win	left wins	left loses	both lose
ℓr	$\ell(1-r)$	$(1-\ell)r$	$(1-\ell)(1-r)$

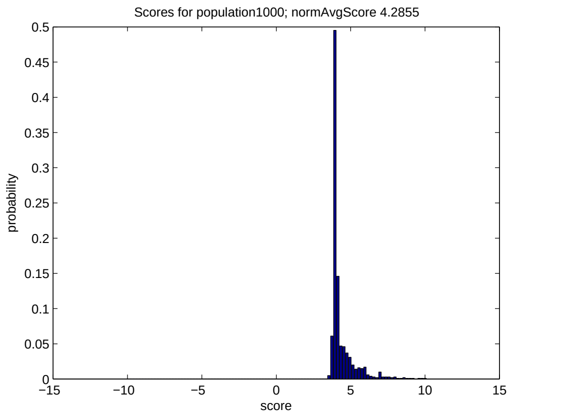
(check that these sum to 1) but we exclude the possibility of both winning or losing, so the chances I've used are (you can simplify the denominators)

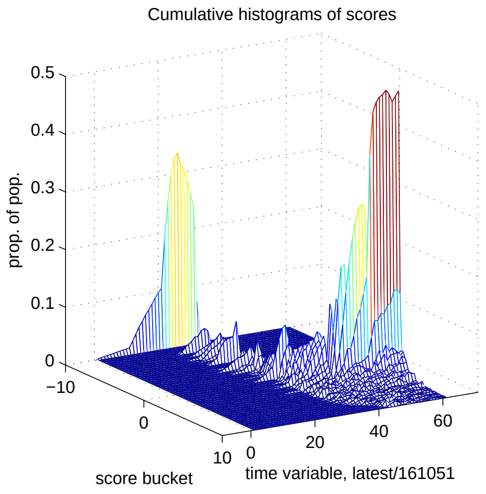
left wins	left loses
$\ell(1-r)/(\ell(1-r)+(1-\ell)r)$	$(1-\ell)r/(\ell(1-r)+(1-\ell)r)$

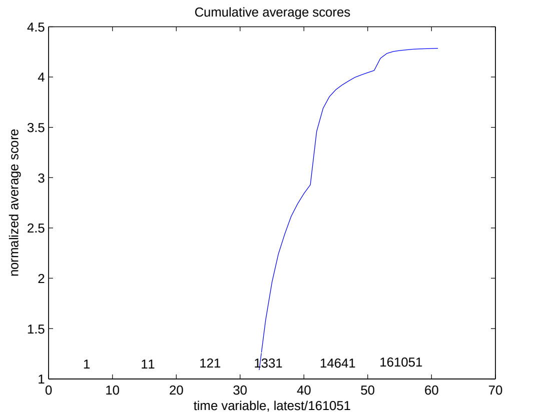
We can also keep scores as in `dovesHawks()` but these will have no effect on the outcome, initially.
To start the simulation, give everybody the same initial dominance (greater than zero: what will happen if every individual starts with dominance 0?).
a) Before running the simulation, what do you think will happen?
b) Write and run the simulation.
Here are sample results for a population of 1000.







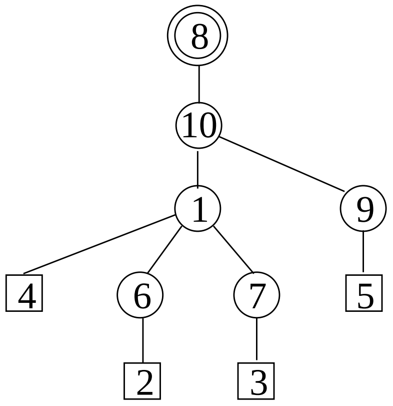




The figures on the left are the resulting (i) distribution of dominance, (ii) distribution of scores, and (iii) the evolution of the average score.
Note that time is variable along the axis: my simulation plotted each iteration up to 11, then each 11th iteration up to 121, then each 121st up to 1331, and so on. So points 1 to 11 are separated by 1 iteration (and show no result because the score for each individual is divided by the number of encounters, and these are initially zero); points 12 to 21 are separated by 11 iterations, and so on. This shows in detail the early evolution of the simulation.
The figures on the right show the evolution of (i) the distribution of dominance, from an initial dominance of 1 for everybody, evolving to a very small dominance for a large proportion of the population; (ii) the distribution of scores.
c) Compare your results here with those from the *Doves and hawks* Excursion. How far can the theory we developed there be applied?
Show that at least one dove (dominance zero) and at most `popSize` $- 1$ doves will result per-

manently from the initial conflicts. Typically, will the number of doves be about `popSize`/2? What is the expectation?
Show that if more than one hawk (dominance one) remains in the population it is because they have not encountered each other. Is it possible to have no hawks? What is the expectation?
d) Note the very large number of iterations (1771561) I ran. The (very few) individual(s) of high dominance lasted a long time. What would the effects of mortality be on the results? (Say, replace individuals after a certain amount of time—or after a certain amount of damage in fights—by new individuals with randomly chosen dominance.) Can you think of examples of (effectively) immortal individuals in society?
e) A way of doing this would be to allow the scores to affect the dominance. Setting an individual's dominance to zero when heir score drops below, say, -100, will save hem from continual fights and the risk of greater losses. It will also reduce cost to society. What will this do to the cycles (see Excursion *Cycles* below)?
Note that this adds two parameters to the theory of dominance, where there were none before: the ratio of losses to gains (which was hitherto used only to establish scores, not to affect dominance), and the ratio of this new value to gains. By tweaking the parameters, such a "phenomenological" theory might be made to "fit" a variety of empirical observations on herd or flock animals.
So far, individuals fight based on their own dominance but do not take into account the dominance of their opponent. How would you insert a "display" mechanism (raising hackles, pounding one's chest, flashing one's Porsche) to dissuade the opponent before an actual fight begins? Or some other mechanism to avoid fights based on reputation?

13. **Hierarchies.** Without altering the initial postulates of the previous Excursion (dominance is the proportion of the fights you win, probability of fighting equals dominance, and probability of winning when you do fight is proportional to dominance), find out what sort of hierarchies develop.
A *hierarchy* can be represented as a matrix, in this case with one element for each pair of individuals in the population, and with that element counting the number of fights won (or the scores): element (j, k) counts the fights j has won over k.
How do we detect hierarchy in this matrix? Let's start with a *tree*. The tree



could be represented as

	1	2	3	4	5	6	7	8	9	10	
1				1		1	1				3
2											
3											
4											
5											
6		1									1
7			1								1
8										1	1
9					1						1
10	1								1		2
	1	1	1	1	1	1	1		1	1	9

or, after rearrangement to reveal the structure a little more clearly,

	8	10	1	4	9	5	6	7	2	3	
8		1									1
10			1		1						2
1				1			1	1			3
4											
9						1					1
5											
6									1		1
7										1	1
2											
3											
		1	1	1	1	1	1	1	1	1	9

In the drawing, I double-circled the *root* node of the tree, and showed the *leaf* nodes as squares. In the matrices, I included a column and a row at left and on top to give the node labels, and I added a column and a row at right and on the bottom giving, respectively, the row sums and the column sums. From these sums you can identify the root, whose column sum is 0 (blank for clarity), and the leaves, whose row sums are 0.
(The convention is that lines are "directed" from above to below in the drawing, and that these directions are given in the matrices by considering rows are directed towards columns, in the sense of $j \to k$ if $(j,k) \neq 0$. So 8→10, etc.)
Thus, with a tree, we can identify the root by its zero column sum. We can represent a node as a row vector of zeros except for a 1 in the position corresponding to the node. We can pick up the hierarchy from this by matrix multiplication. For example, node 8 leads to node 10

$$(0,0,0,0,0,0,0,1,0,0)\begin{pmatrix} &&&1&&1&1&&& \\ &&&&&&&&& \\ &&&&&&&&& \\ &&&&&&&&& \\ &&&&&&&&& \\ &1&&&&&&&& \\ &&1&&&&&&& \\ &&&&&&&&&1 \\ &&&&1&&&&& \\ 1&&&&&&&&1& \end{pmatrix} = (0,0,0,0,0,0,0,0,0,1)$$

and similarly, node 10 leads to nodes 1 and 9, etc.
We can write a loop which picks up each of these levels and accumulates the following hierarchy

for this case

$$\begin{pmatrix} 8 & 10 & 1 & 9 & 4 & 5 & 6 & 7 & 2 & 3 \\ 0 & 1 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 \end{pmatrix}$$

where the upper row labels the nodes in ascending order of level in the tree (trees in computer science are drawn upside-down, so the root on top ascends to the leaves on the bottom: sorry!) and the lower row gives those levels.
a) Write a function `[newGraph,hier] = toposort(graph)` which inputs the 1–10 matrix I gave first above and outputs the 8-10-1-9-4-5-6-7-2-3 matrix I gave next above, and the hierarchy represented by `hier`. How would you stop the loop that multiplies the node vector by the `graph` matrix to get the next node vector? How would you permute the matrix `graph` to get `newGraph`?
Now, if we added a link, say from node 7 to node 5, the level of node 5 would have to increase to 4 because it now depends on a node (7) of level 3. The loop that would implement the level-finding for a tree, above, would continue on to find this new candidate for the level of node 5

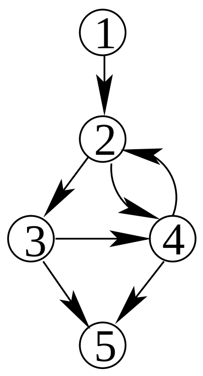
$$\begin{pmatrix} 8 & 10 & 1 & 9 & 4 & 5 & 6 & 7 & 2 & 3 & 5 \\ 0 & 1 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 4 \end{pmatrix}$$

and you'll have to work backwards along this `hier` array to eliminate the

$$\begin{pmatrix} 5 \\ 2 \end{pmatrix}$$

claimant.
b) Modify your `toposort()` to handle such extra links. The example I've given is no longer a "tree" but a "directed acyclic graph (DAG)". If there are no cycles, we still have a hierarchy, just a little more complicated than a tree. But we might also introduce cycles. What will a cycle do to your loop? How can you stop it?
c) Modify the dominance simulation of the previous Excursion to tally the victories of j over k. The resulting matrix will have entries that are 0, 1 or greater: make sure your `toposort()` still behaves.
d) When you run this modified simulation (which you should do for small populations initially, or your program will be reporting on `popSize`^{2}-size arrays) do you discover hierarchies? Cycles? Discuss the implications for zoology of your results.
e) The simplest kind of cycle involves only two nodes, $j \to k$ and $k \to j$. These can easily be eliminated by, say, diminishing the larger of j, k and k, j by the smaller, and making the smaller 0: it's plausible that the individual of a pair who wins more fights is dominant. Try getting your `toposort()` to do this within your simulation, but use the resulting hierarchy to rearrange the entire *original* matrix.
(Your `toposort()` is executing a "topological sort". Note that the order assigned to nodes at the same level is arbitrary. What will happen if the graph is a chain of n nodes including all levels 0 to $n-1$? The algorithm I have pointed you to will require $\mathrm{O}(n^3)$ work: find the part of the implementation that costs this many multiplications. ("Big-O" is computer science notation meaning, in this case, that the work as a function of n, $w(n) \le kn^3$ for some constant k once n is big enough that the n^3 is the most important n-dependence in finding the value of the function.) Topological sort can be written with $\mathrm{O}(n \log n)$ complexity. If $n = 1000$ and the logarithm is to base 10, compare n^3 and $n \log n$.)

14. **Cycles.** Find all the cycles in the matrix recording fights won pairwise by individuals in the previous Excursion.
Here is a five-node graph with two cycles and its matrix representation.



$$G = \begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

a) Show that you can identify cycles of length ℓ by looking at the diagonal of G^ℓ.
Unfortunately the diagonal of G^ℓ does not tell us the intermediate nodes in the cyclic path, because matrix multiplication throws away this information. We need a more fundamental operation than matrix multiplication, which can keep the intermediate steps. The most straightforward way to create such an operation is to change from matrices to *relations.*
Let's walk through the thinking, and a MATLAB (matrix) implementation of relations and the *join,* a central operation of the "relational algebra".
The relational form of the above graph is the "binary relation"

```
1     2
2     3
2     4
3     4
3     5
4     2
4     5
```

b) See how each of the seven "tuples" of this binary relation represents one of the edges of the graph (or one of the nonzero entries of the matrix). Write a MATLAB function `rel = mat2rel(matrix)` which will create a tuple of `rel` from each of the nonzero entries of `matrix`.
The relational operation underlying matrix multiplication is the *join.* For a start, here is G joined with itself.

```
    G                  G                G2 = join(G,G)
1       2          1       2             1     2     3
4       2          2       3             1     2     4
2       3          2       4             4     2     3
2       4          3       4             4     2     4
3       4          3       5             2     3     4
3       5          4       2             2     3     5
4       5          4       5             2     4     2
                                         2     4     5
                                         3     4     2
                                         3     4     5
```

I've rearranged the tuples in G the first time it appears. This does not change the meaning of the relation: it still would give the same matrix and the same graph. But it facilitates understanding the join.

The join combines tuples from its two arguments (in this case, both G) by looking for matches between the second column of the first argument and the first column of the second argument. The join combines all matching tuples from one side with all matching tuples from the other. Thus, since G has two tuples ending in 2 and two tuples beginning in 2, `join(G,G)` will have four tuples with 2 in the middle. How about matches on 3? on 4?

Note that `join(G,G)` is also a relation, but not a binary one: it is a "ternary relation" because it has three columns.

c) Show how this join gives the same result as multiplying the matrix version of G with itself, with the additional information about the intermediate step. Basically $G \times G$ and `join(G,G)` give the paths of length 2 in the graph: the matrix multiplication without, and the join with, the intermediate node.

d) Write a MATLAB function `R3 = join(R1,R2)` to join the two relations `R1` and `R2`. Beware that the relations need not be binary and need not have the same number of columns as each other: make your implementation robust against this. The easiest way is to write a double loop, circulating through each tuple of each relation. But note that the cost is $O(e \times f)$ if `R1` has e tuples and `R2` has f tuples. Note also that these quantities are the numbers of *edges* in the graphs if the two relations represent graphs (e.g., they are binary): what is this cost in terms of the number of *nodes* of each graph (what is the worst case)?

The tuples of `G2` corresponding to the diagonal elements of G^2 are the two that begin and end on the same node: `4 2 4` and `2 4 2`. These both correspond to the (24) cycle (2→4 and 4→2). So they can be detected by looking for repeats within the tuple, specifically repeats of the same node at the beginning and at the end of the tuple.

We are going to want to remove such cycles from the result and report them separately. Let's suppose we have done this with a MATLAB function `[cyc,rest] = nextCycles(G2)` where `cyc` contains the cycles and `rest` is the remaining tuples of `G2`. But we won't try to build this function yet, because we are going to encounter some more subtleties.

We'll explore the `join` operation some more first.

e) Show that the third and fourth steps, if (changing the definition of `G2` and ignoring `cyc` in the following for the time being) `[cyc,G2] = nextCycles(join(G,G))`, `[cyc,G3] = nextCycles(join(G2,G))`, (or `[cyc,G3] = nextCycles(join(G,G2))`: show it's the same) and `[cyc,G4] = nextCycles(join(G3,G))` are

```
             G3                              G4
    1     2     3     4          1     2     3     4     2
    1     2     3     5          1     2     3     4     5
    4     2     3     5
    1     2     4     5
    2     3     4     5
```

where `nextCycles()` has removed tuples with *any* duplicate node.

(Show that `3 4 2 4` and `1 2 4 2` arise in the join leading to `G3`: these just describe the beginning of an infinite loop around the cycle (24), which is why they must be discarded.)

And show that the next join, `join(G4,G)`, will produce an empty relation: there is no path of length 5+1 in the graph.

f) Now that you know that `nextCycles()` must eliminate all tuples with repeated nodes from `rest`, putting only those with repeats only in the first and last positions into `cyc`, write function `[cyc,rest] = nextCycles(rel)`.

You should probably write some auxiliary functions first:

`rept = repeats(tuple)` constructs a ternary relation on `(value,1st,2nd)` where `value` is the repeating value, `1st` is the position where it is first found, and `2nd` is the position where it is found again—`rept` may have several such tuples if there are several repeating values or

if they repeat in more than one place;
`cyc = addunique(cyc,cycle)` adds a `cycle` to the output relation `cyc` only if it is not already in `cyc`; and
`cycle = canonCycl(tuple(1,1:q-1))` rearranges the cycle of values so that the same cycle is not described in different ways, e.g., `3 4 3 2`, `2 3 4 2` and `4 2 3 4` all give the same cycle `2 3 4`.
g) Finally, write function `cycles = findCycles(matrix)` which puts all these pieces together to find *all* the cycles in the graph described by `matrix`. It must loop to find cycles of length 2, of length 3, and so on. Use `findCycles()` in the simulation of Excursion *Dominance* above to report on cycles as they develop in the matrix of fights won.
h) Look up [Mer08] for accounts of a full relational programming language. Implement it and rewrite all of *Excursions in Computing Science* in Aldat instead of in MATLAB.

15. Look up Ludwig Eduard Boltzmann (1844–1916). How does the *Boltzmann constant* appear in the simulations of these Notes? What is a *Boltzmann brain*?

16. Wannier [Wan87, p.476] gives Perrin's calculation of Boltzmann's constant from data on *Brownian motion*: rework it into the units (nm, meV) used in these Notes. (This requires perhaps more physics than I've covered.)

17. Look up James Clerk Maxwell (1831–79). What did he have to say about the rings of Saturn?

18. A second puzzling connection between energies, which follow a Boltzmann distribution, and momentum magnitudes, which have a Maxwell distribution, is that the energy is just the square of the momentum magnitude (divided by twice the mass, which is the same for all molecules in the simulation so far). Because of this simple relationship shouldn't they have the same distribution? (Note 13)
a) Suppose a quantity has each value from 1 to 10 occurring once, so that it has the following five-bucket histogram.

values	1–2	3–4	5–6	7–8	9–10
# occurrences	2	2	2	2	2

What does the five-bucket histogram look like for the *square* of the quantity?
b) How does this argument depend on dimensionality? Note 14 and the Excursion *Maxwell distribution* show differences between momentum magnitude distributions in 2 and 3 dimensions. Why?

19. **Maxwell distribution.** (Note 14) Recalculate the partition function and the distribution of momentum magnitudes in 3 dimensions to show

$$
\begin{aligned}
Z &= (\sqrt{\pi(2m/\beta)})^3 \\
M(p)\Delta p &\approx \frac{p^2\Delta p}{\sqrt{\pi/2}(m/\beta)^{3/2}} e^{-\beta p^2/(2m)} \\
&= \sqrt{\frac{2}{\pi}}(\frac{\beta}{m})^{\frac{3}{2}} p^2 \Delta p e^{-\beta p^2/(2m)}
\end{aligned}
$$

with the approximation getting better as Δp gets smaller.

20. Why is $Ee^{-\beta E}$ zero at $E = 0$ and $E = \infty$ in Note 14? How about $E^2e^{-\beta E}$?

21. Why did I not try to find the mode of the Boltzmann distributio in Note 14?

22. Show that for Maxwell (3D)

$$\begin{aligned}
\mu_1 &= 2\sqrt{\frac{a}{\pi}} = 2\sqrt{\frac{2m}{\pi\beta}} \\
\mu 2 &= \frac{3a}{2} = \frac{3m}{\beta} \\
\sigma^2 &= (3 - \frac{8}{\pi})\frac{a}{2} \\
\text{mode} &= \sqrt{a} = \sqrt{\frac{2m}{\beta}}
\end{aligned}$$

For μ_1 you can take direct advantage of results in Note 14. For μ_2 you'll have to find a new u in order to integrate by parts.
Look up Maxwell's distribution to check these results.

23. a) Generalize to three-samples the results of Note 15 that

$$\begin{aligned}
\sum_{ij} p_i p_j \frac{i+j}{2} &= m \\
\sum_{ij} p_i p_j (\frac{i^2+j^2}{2} - (\frac{i+j}{2})^2) &= \frac{\sigma^2}{2}
\end{aligned}$$

b) Generalize this to N-samples ($i_1, i_2, ...i_N$, and corresponding frequencies).

24. Look up Feynman's discussion of temperatures via thermal equilibrium of two mixed gases [FLS64, §39-4] and the same with the gases separated. (Feynman requires a prior discussion of pressure.) Compare the treatment in Note 17.

25. Why are the two Maxwell distributions for the combined statistics from Helium at 100°K and 500°K in Note 17 both twice too high? Rerun the simulation and correct the theoretical plots.

26. Make discrete Fourier transforms in 501 and 500 dimensions and use these to reduce the variance of the energy ignorance before and after gas mixing starts in Note 17. Note the behaviour at the ends of these intervals when you retain only the largest four or five amplitudes.

27. From what height must we drop a body so that on impact with the ground its temperature rises by 1K°, assuming all its acquired energy converts to a temperature rise?

28. The discussion in Note 19 of the state function $S(U, V, N)$ is based on Robertson [Rob93, pp.111,113–5]. The N! dividing the volume of the hypersphere is discussed on p.114. Look up the further discussion of the "Gibbs Paradox" on pp.154–6, 540–1.

29. **Sackur-Tetrode equation.** Complete the derivation in Note 19 to show that the number of states in 3D for a monatomic gas isolated from outside is

$$\Omega = (\frac{2\pi m}{h^2})^{\frac{3N}{2}} \frac{V^N U^{\frac{3N}{2}}}{\frac{3N}{2} N!(\frac{3N}{2} - 1)!}$$

and so

$$S = \frac{5k_B N}{2} + k_B N \ln((\frac{4\pi m}{3h^2})^{\frac{3}{2}} \frac{VU^{\frac{3}{2}}}{N^{\frac{5}{2}}})$$

Thus show

$$U = \frac{3}{2}Nk_BT$$

and

$$PV = Nk_BT.$$

30. Using the Sackur-Tetrode equation in either 2D or 3D and the work of either Note 19 or the previous Excursion, find explicit functions $S(P, V, N)$, $S(T, V, N)$ and $S(T, \mu, N)$. How do these contrast with the (energy) functions of state discussed in Note 20?

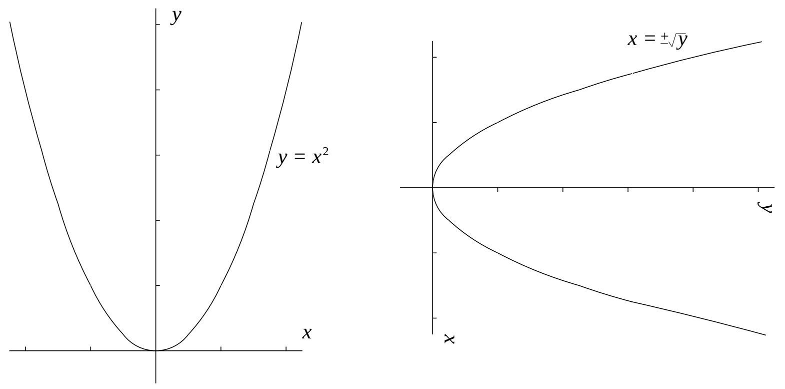
31. In 3D, using Sackur-Tetrode, show that

$$\frac{\mu}{T} = -\text{slope}_N S = -k_B \ln(\frac{VU^{\frac{3}{2}}}{N^{\frac{5}{2}}}\frac{N_0^{\frac{5}{2}}}{V_0U_0^{\frac{3}{2}}}) = k_B \ln(\frac{N^{\frac{5}{2}}}{VU^{\frac{3}{2}}}\frac{V_0U_0^{\frac{3}{2}}}{N_0^{\frac{5}{2}}})$$

and so, for ideal gases with $V = Nk_BT/P$ and $U = (3/2)Nk_BT$ and, with $T = T_0$

$$\mu = k_BT\ln(P/P_0)$$

32. **Local inverses of functions.**[2] A *function* is defined in mathematics to be a many-to-one mapping from the possible values of its argument to the possible values of the result. Thus, $f(x) = x^2$ gives 9 if x is either 3 or -3: 2-to-1. One-to-one mappings are also functions, in the special case that "many" becomes "one", e.g. the cubing function.
The *inverse* of a function is the function which reverses this mapping. So the inverse of cubing, $y = x^3$, takes the cube root, $x = \sqrt[3]{y}$.
But $f(x) = x^2$ has no inverse, because $\pm\sqrt{y}$ (where $y = x^2$) has two possible values (except when $y = 0$) not just the one value required for it to be a function.



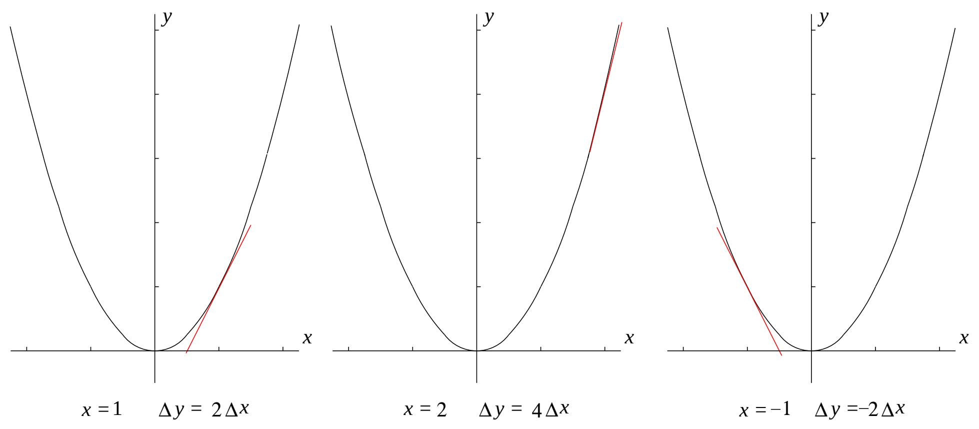
a) Find a function which is $\infty : 1$ and so, of course, has no inverse.
However, *locally* any function has an inverse. In the case of squaring, "locally" means either non-negative x or non-positve x. In other cases, "locally" might be more restricted.
b) What does "locally" mean for the function you found in (a)?
Certainly, "locally" is satisfied by a range Δx where Δx is made arbitrarily small.
But we can extract this from *any* function for which we know the slope

$$\Delta f = \text{slope}_x f \Delta x$$

Thus, for $f(x) = x^2$ in particular

$$\Delta y = 2x\Delta x$$

[2]This Excursion is intended to help with Note 20.



And of course

$$\Delta x = \frac{1}{2x}\Delta y$$

is the local inverse. Slopes permit *linear approximations* for any function, and these can always be inverted.
c) When does $\Delta x = \Delta y/(2x)$ fail?
d) Rewrite the $2x$ in terms of y, so that no x appears on the right-hand side of the equation in (c). What additional conditions must be stated?
All this works also for functions of *more than one variable*. Try

$$z = f(x,y) = x^2 + y^2$$

So

$$\begin{aligned}\Delta z &= \text{slope}_x f\Delta x + \text{slope}_y f\Delta y \\ &= 2x\Delta x + 2y\Delta y\end{aligned}$$

Turning this inside-out (it is unsatisfactory to talk about an "inverse"—why?)

$$\Delta x = \frac{1}{2x}\Delta z - \frac{y}{x}\Delta y$$

e) Using the local "inverse"

$$x = g(y,z) = +\sqrt{z - y^2}$$

find the slope$_y g$ and slope$_z g$ and confirm the above result for Δx. (You'll have to convert $\sqrt{z-y^2}$ back to x to make the direct comparison.)

33. **Complementary variables.**[3] Using a function of one variable, $f(x) = x^2$, explore the replacement of x by $y = \text{slope}_x f$ in $g(y) = f - yx$.
a) Show that, for small $\Delta x, \Delta f, \Delta g$

$$\begin{aligned}\Delta f &= y\Delta x \\ \Delta g &= y\Delta x - y\Delta x - x\Delta y - \text{slope}_y f\Delta y \\ &= -x\Delta y\end{aligned}$$

Does this justify our saying above that g is a function of y but not of x?
b) Plot $g(x,y) = x^2 - yx$ as a function of both x and y

[3]This Excursion is intended to help with Note 20.

i) parametrically as g versus x for $y = 0, y = 1, y = 2$
ii) parametrically as g versus x for $x = 0, x = 1, x = 2$
iii) fully in 3D
c) In(b)(i) above, with y fixed, what are the values of x at which slope$_x g = 0$?
d) Discuss the difference between y as an independent variable and y as slope$_x g$
e) Revisit the relationship between Lagrangian and Hamiltonian in Note 39 of Book 8c Part IV and discuss it in the light of complementary variables as examined in Note 20 and this Excursion.

34. Verify the dependences of F, G and Φ along the lines of the treatment of $H(S, P, N)$ in Note 20.

35. What are the functions $X(S, P, \mu)$, '$Y(S, V, \mu)$ and $Z(T, P, \mu)$ in terms of U, TS, PV and μN in Note 20?

36. What other function besides enthalpy can be used in Note 20 to find $\mu(P)$?

37. Which four functions of state would be candidates to describe isothermal processes (Note 20)?

38. Which function of state in Note 20 is conserved under adiabatic, isochoric processes on a fixed number of molecules? Isothermal, isochoric processes on a fixed number of molecules?

39. In Note 22 we are using generalizations of what we learned in our computer simulations of a monatomic gas. Further simulations are bulky to describe and I am not sure they are instructive. But you might like to rewrite the routines to allow the bottom wall of the container to be heated and the top wall to move like a piston, and thereby confirm the ideal gas law$PV = Nk_BT$.

40. Redo the calculations of Note 22 for 3D gases.

41. Because Carnot engine efficiency depends only on temperatures, this gives another (but equivalent) way of *defining* temperature. Look up "thermodynamic temperature",

42. What is the maximum possible efficiency of your car? Look up the operating temperature of its motor.

43. When we calculated ΔS from the Sackur-Tetrode equation in Note 22 we got a dependence on $\ln \frac{(V_1+V_2)^2}{4V_1V_2}$ which is zero only when $V_1 = V_2$. Why should the entropy increase if the volumes are different even if the two gases were initially at the same temperature?

44. Using $\Delta V \Delta P = \Delta Q \Delta T/T$ from Note 22 and the ideal gas law, come as close as you can to the Sackur-Tetrode equation a) for volume V compared with V_0 b) for pressure P compared with P_0.

45. **Causality.** One of the hangups of some aspects of science is the philosophical obligation to trace the causes of every phenomenon. This is appropriate for engineering (which is parodied by Rube Goldberg and Heath Robinson with their machines of ultimate causal chains) and evidently attractive for biology with its biochemical cycles and ecological food chains.
But causality is a diminishing philosophy in physics where conservation laws and laws of stationary quantities are more useful than force calculations, for instance.
a) Review all the physics in these Notes to see how many references you can find to causality. Can you find any way to fit, say, conservation of energy and momentum into a causal framework?
b) Here's some more physics. Tidal forces between the Earth and the Moon have stopped the Moon's rotation relative to the earth, so that the same side of the Moon always faces us.

Conservation of angular momentum requires that the lost angular momentum of the Moon's rotation not disappear. So the Moon has moved further away from the Earth than it was when it was rotating. Find a mechanical explanation of this.
c) Re-calculate the billiard-ball collisions of Excursion *Billiards* in Book 8c Part IV using Newtonian forces. Was it easier than using the conservation laws of the Excursion?
d) What does the word "cause" mean? Where must "first cause" come in? What does the light cone of special relativity require?

46. Any part of the Preliminary Notes that needs working through.

References

[Bla09] David N. Blauch. Kinetic molecular theory: Maxwell distribution. http://www.chm.davidson.edu/vce/KineticMolecularTheory/Maxwell.html, 2009.

[FLS64] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics, Volume I*. Addison-Wesley, 1964.

[FS64] Michael Flanders and Donald Swan. At the drop of another hat. PMC 1216, E.M.I. Records Ltd., Middlesex, England, 1964.

[HL11] Terry Hunt and Carl Lipo. *The Statues That Walked: Unravelling the Mystery of Easter Island*. Free Press, New York, 2011.

[Mer08] T. H. Merrett. Aldat. URL http://www.cs.mcgill.ca/ tim/aldat/, 2008.

[Rob93] Harry S Robertson. *Statistical ThermoPhysics*. P T R Prentice-Hall, Inc., Engelwood Cliffs, NJ, 1993.

[Ros02] Sheldon M Ross. *Probability Models for Computer Science*. Harcourt/Academic Press, San Diego, 2002.

[SP73] J Maynard Smith and G R Price. The logic of animal conflict. *Nature*, 246:15–18, Nov. 2 1973.

[Wan87] Gregory H Wannier. *Statistical Physics*. Dover Publications, Inc., Mineola, N.Y., 1987. originally New York 1966 John Wiley and Sons, Inc.