1. Background: Who Will Find This Useful?

Stick models are collections of lines in $P^n$. A stick model more specifically tells us which of the lines intersect in points and which do not. Here is an example of a stick model:

```
  1
 /|
/  |
  2 3
  |
  4
```

This stick model has four lines. Line 1 intersects lines 2 and 3 in a point but does not intersect line 4. Line 4 intersects lines 2 and 3. Lines 2 and 3 do not intersect. As can be seen, all of the necessary intersection information is contained in the stick model.

In order to study a stick model, it is useful to look at the sets of projective lines which obey the intersections given by the stick model. These sets of projective lines can be studied algebraically, giving a means by which the stick model can be studied. “randsticks” is a function which generates a random set of lines which obey the intersections given by the input of a stick model. Call this random set of lines an embedding of the stick model. “stickideal” is a function which, given

---

Thanks to the NSF grant number DMS-0849637 for supporting this work.
the output of “randsticks” as input, produces an ideal for the embedding of the stick model. These two functions are all that is needed by the user to generate generic embeddings of stick models.

2. Code

While the user need only understand the functionality of “randsticks” and “stickideal”, all other “black box” code is included for completeness.

-- Function: randpoint(pDim)
-----------------------------
-- Input: pDim: dimension of projective space.
--
-- Output: list of length pDim+1 of a point in ZZ/32003^(pDim+1)
--
randpoint = (pDim) ->
(
    pt = {};
    for i from 0 to pDim do(
        pt = append(pt,random(ZZ/32003));
    );
    return pt;
);

-- Function: remFirst(L)
-------------------------
-- Input: L: a list
--
-- Output: The list L with the first element removed
--
remFirst = (L) ->
(
    ret = {};
    for i from 1 to #L-1 do(
        ret = append(ret,L_i);
    );
    return ret;
);
-- Function: randVectorInPlane(L)
-----------------------------
-- Input: L: a list of 2 vectors (represented in lists)
-- in the same vector space ex: {{1,0,0},{1,1,1}}
-- Output: a random vector in the span of the two given
-- vectors
--
randVectorInPlane = (L) ->
(  
a = L_0;
  b = L_1;
  c = (random(ZZ/32003))\cdot a + (random(ZZ/32003))\cdot b;
  return c;
);

-- Function: contains(L,el)
-----------------------------
-- Input: L: A list
-- el: An object
-- Output: true--if el is contained in L
-- false-- if el is not in L
--
contains = (L,el) ->
(  
  for i from 0 to #L-1 do(
    if L_i == el then(return true;);
  );
  return false;
);

-- Function: numIntersections(L1,L2)
-----------------------------
-- Input: L1: a list
-- L2: a list
-- Output: the number of elements the two
-- lists have in common.
--
numIntersections = (L1,L2) ->
(  
  numInt = 0;
  for i from 0 to #L1-1 do(
cur = L1_i;
for j from 0 to #L2-1 do(
    if L1_i == L2_j then(numInt = numInt+1;);
);
return numInt;

-- Function: randsticks(sNum,intr,pDim)
---------------------------------------
-- Note that the stick model needs to be fully
-- connected, i.e. no disjoint parts.
--
-- Input: sNum: an integer representing the number
-- of sticks.
-- intr: a list of points (a,b) where a!=b
-- and a,b are in {1,...,sNum}
-- (a,b) implies that line a
-- intersects line b.
-- pDim: dimension of the projective space
-- the embeddings will take place in.
-- Output: A list of pairs of vectors. Each pair
-- of vectors spans a plane in R^(pDim+1)
-- The first pair corresponds to the first
-- stick, the second pair to the second,
-- etc.. ex: {{{1, 0, 0, 0, 0},
-- {0, 1, 0, 0, 0}},
-- {{.49471, -.797041, 0, 0, 0},
-- {-.476157, .00570924, -.0633647,
-- -.364895, -.198453}}}
-- represents two planes in R^5
-- representing lines in P^4

randsticks = (sNum,intr,pDim) ->
(  --the next 6 lines create our first plane in R^(pDim+1)
  f1 = {1,0};
  f2 = {0,1};
  for i from 1 to (pDim -1) do(
      f1 = append(f1,0);
      f2 = append(f2,0);
RANDOM EMBEDDINGS OF STICK MODELS IN MACAULAY 2

5

oneInt = {}; twoInt = {}; completed = {1}; L = new MutableList from {{f1,f2}};
pair = {}; for i from 1 to sNum do(
    curpair = {}; for j from 1 to #intr do(
        if (intr_(j-1))_0 == i then(
            curpair = append(curpair, (intr_(j-1))_1);
        );
        if (intr_(j-1))_1 == i then(
            curpair = append(curpair, (intr_(j-1))_0);
        );
    );
    pair = append(pair,curpair);
); finished = false;
while finished == false do(
    happened =false;
    oneInt = {}; twoInt = {}; for i from 1 to sNum do(
        a = pair_(i-1);
        if numIntersections(completed,a)==1 and contains(completed,i) == false
            then(oneInt=append(oneInt,i));
        if numIntersections(completed,a)==2 and contains(completed,i) == false
            then(twoInt=append(twoInt,i));
    );
    if #twoInt > 0 then(
        happened=true;
        pop = twoInt_0;
        twoInt = remFirst(twoInt);
        comp = pair_(pop-1);
        interplanes = {}; for j from 0 to #comp-1 do(
            if contains(comp,j) == true
                then(interplanes = append(interplanes,comp_j));
        );
    );
x1 = L#(interplanes_0 -1); --the vectors spanning one
   -- constraining plane.
x2 = L#(interplanes_1 -1); --the vectors spanning the
   -- other constraining plane.
x1 = randVectorInPlane(x1);
x2 = randVectorInPlane(x2);
L#(pop-1) = {x1,x2};
completed = append(completed,pop);

if #oneInt > 0 and #twoInt == 0 and happened==false then(
   pop = oneInt_0;
   oneInt = remFirst(oneInt);
   comp = pair_(pop-1);
   interplanes = {};
   for j from 0 to #comp-1 do(
      if contains(completed,comp_j) == true
         then(interplanes = append(interplanes,comp_j));
   );
   x1 = L#(interplanes_0 -1); --The vectors spanning
   --the constraining plane
   x1 = randVectorInPlane(x1);
x2 = randpoint(pDim);
L#(pop-1) = {x1,x2};
completed = append(completed,pop);
);

if #oneInt == 0 and #twoInt == 0 then(finished = true;);
);
finL = {};
for i from 1 to #L do(
   finL = append(finL,L#(i-1));
);
return finL;

-- Function: stickideal(sticks)
-----------------------------
-- Input: sticks: a list of pairs of vectors of equal
-- dimension. The vectors described a
-- basis for planes.
-- Output: The intersection of the ideals of all of
-- the planes.
RANDOM EMBEDDINGS OF STICK MODELS IN MACAULAY 2

stickideal = (sticks) -> (  
    len = length sticks_0_0;  
    T = ZZ/32003[s,t];  
    S = ZZ/32003[x_1..x_len];  
    sepidl = {};  
    i=0;  
    for i from 0 to (length sticks)-1 do(  
        sepidl = append(sepidl,ker map(T,S,s*sticks_i_0+t*sticks_i_1));  
    );  
    --return sepidl; --For the list of ideals before intersection, uncomment.  
    return intersect sepidl;  
);

3. Bugs and Miscellaneous

It is important to note that not all stick models can be embedded correctly with the above code in Macaulay 2. Stick models with the constraint that all sticks have three intersections have a rigidity which creates an error when trying to embed the last stick of the stick model. For this reason, only use this code for the purpose of embedding less rigid stick models.

Also, originally the code was written over the real numbers, but the concatenation of the decimals ultimately eliminated intersections which were intended to exist. In this first approach, a slower but more random vector generation algorithm was used (using Monte Carlo methods). In this algorithm, it was useful to find orthonormal bases for planes. For the reason that this process may be needed by someone using Macaulay 2 in the future, I have included the code for it despite it being disjoint from the above code for stick model embeddings:

-- Function: orthobasis(L)  
--------------------------
-- Input: L: a list of 2 vectors in the same vector space  
-- ex: {{1,0,0},{1,1,1}}  
-- Output: a list of 2 vectors spanning the same plane as L  
-- but which are perpindicular and of length 1.  
-- ex: {{1,0,0},{0,-.707107,-.707107}}  
--
orthobasis = (L) ->
\[
\begin{align*}
\text{a} &= L_0; \\
\text{b} &= L_1; \\
\text{dotab} &= 0; \\
\text{dotaa} &= 0; \\
\text{for } i \text{ from } 0 \text{ to } \#a-1 \text{ do}( \\
    &\quad \text{dotab} = \text{dotab} + (a_i)*(b_i); \\
    &\quad \text{dotaa} = \text{dotaa} + (a_i)*(a_i); \\
\); \\
\text{if } \text{dotab} == 0 \text{ then } (\text{return L}); \\
\text{c} &= (-\text{dotaa/\text{dotab}})*b+a; \\
\text{tot} &= 0; \\
\text{for } i \text{ from } 0 \text{ to } \#a-1 \text{ do}( \\
    &\quad \text{tot} = \text{tot} + (c_i)^2; \\
\); \\
\text{tot} &= \text{tot}^.5; \\
\text{c} &= \text{c/tot}; \\
\text{tot} &= 0; \\
\text{for } i \text{ from } 0 \text{ to } \#a-1 \text{ do}( \\
    &\quad \text{tot} = \text{tot} + (a_i)^2; \\
\); \\
\text{tot} &= \text{tot}^.5; \\
\text{a} &= \text{a/tot}; \\
\text{return } \{\text{c},\text{a}\}; \\
\end{align*}
\]