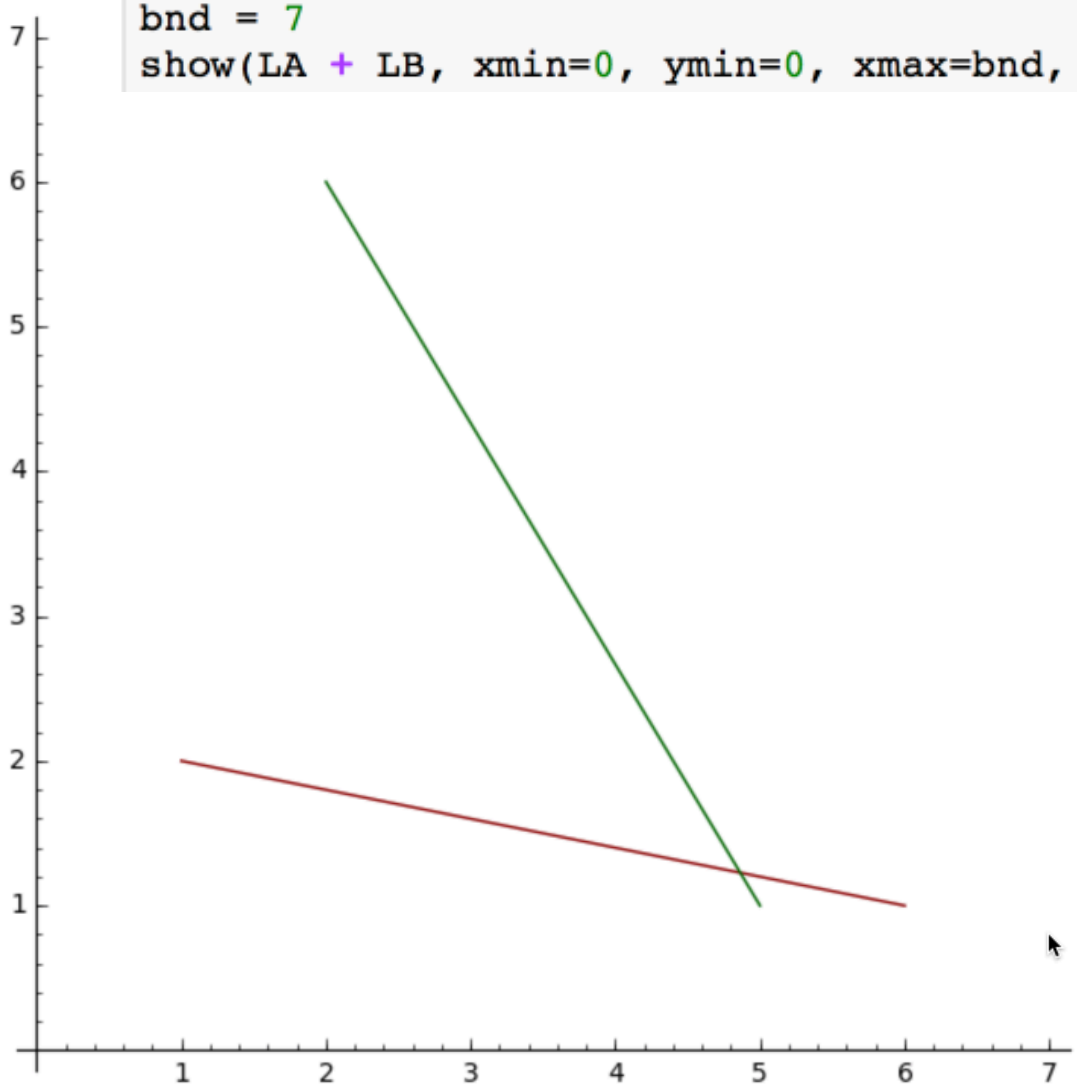
Lecture 8:
Intersecting Parametric Forms

September 14, 2017

```
A1 = [1, 2]; A2 = [6, 1]
B1 = [2, 6]; B2 = [5, 1]
LA = line([A1,A2], color="darkred")
LB = line([B1,B2], color="darkgreen")
bnd = 7
show(LA + LB, xmin=0, ymin=0, xmax=bnd, ymax=bnd, aspect_ratio=1)
```



Surface the Parametric Form

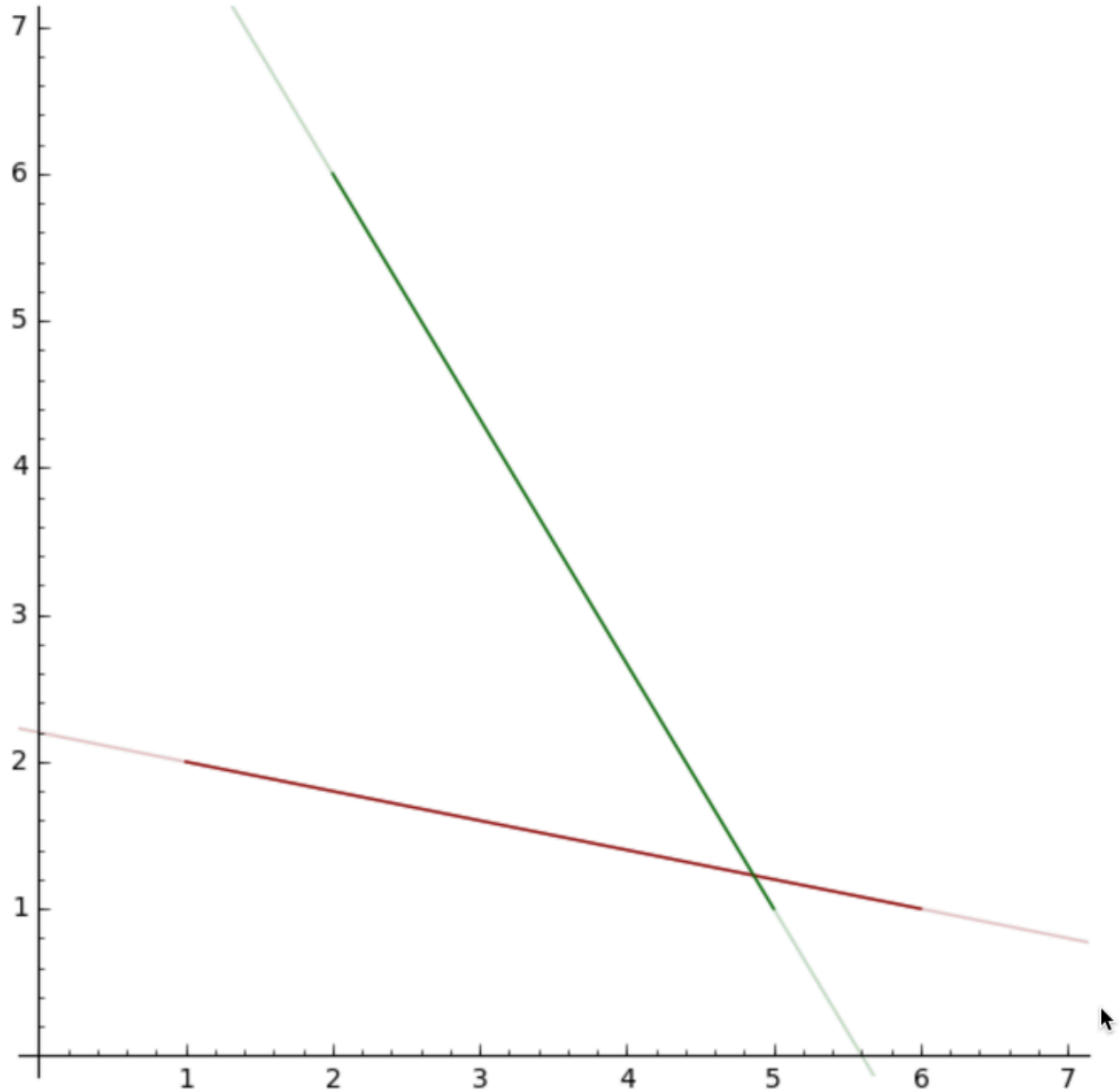
```
t = var('t')
def lapx(t) : return (A1[0]+(A2[0]-A1[0])*t)
def lapy(t) : return (A1[1]+(A2[1]-A1[1])*t)
def lbpx(t) : return (B1[0]+(B2[0]-B1[0])*t)
def lbpy(t) : return (B1[1]+(B2[1]-B1[1])*t)
LAP = parametric_plot((lapx(t),lapy(t)),(t,0.0,1.0),color='darkred')
LBP = parametric_plot((lbpx(t),lbpy(t)),(t,0.0,1.0),color='darkgreen')
bnd = 7
show(LAP + LBP, xmin=0, ymin=0, xmax=bnd, ymax=bnd, aspect_ratio=1)
```

The essentially same thing written out in linear algebraic format.

$$\begin{vmatrix} x(t) \\ y(t) \end{vmatrix} = \begin{vmatrix} a1x \\ a1y \end{vmatrix} + \left(\begin{vmatrix} ax2 \\ ay2 \end{vmatrix} - \begin{vmatrix} a1x \\ a1y \end{vmatrix} \right) t$$

SageMath draws an identical figure.

Lines Know No Bounds

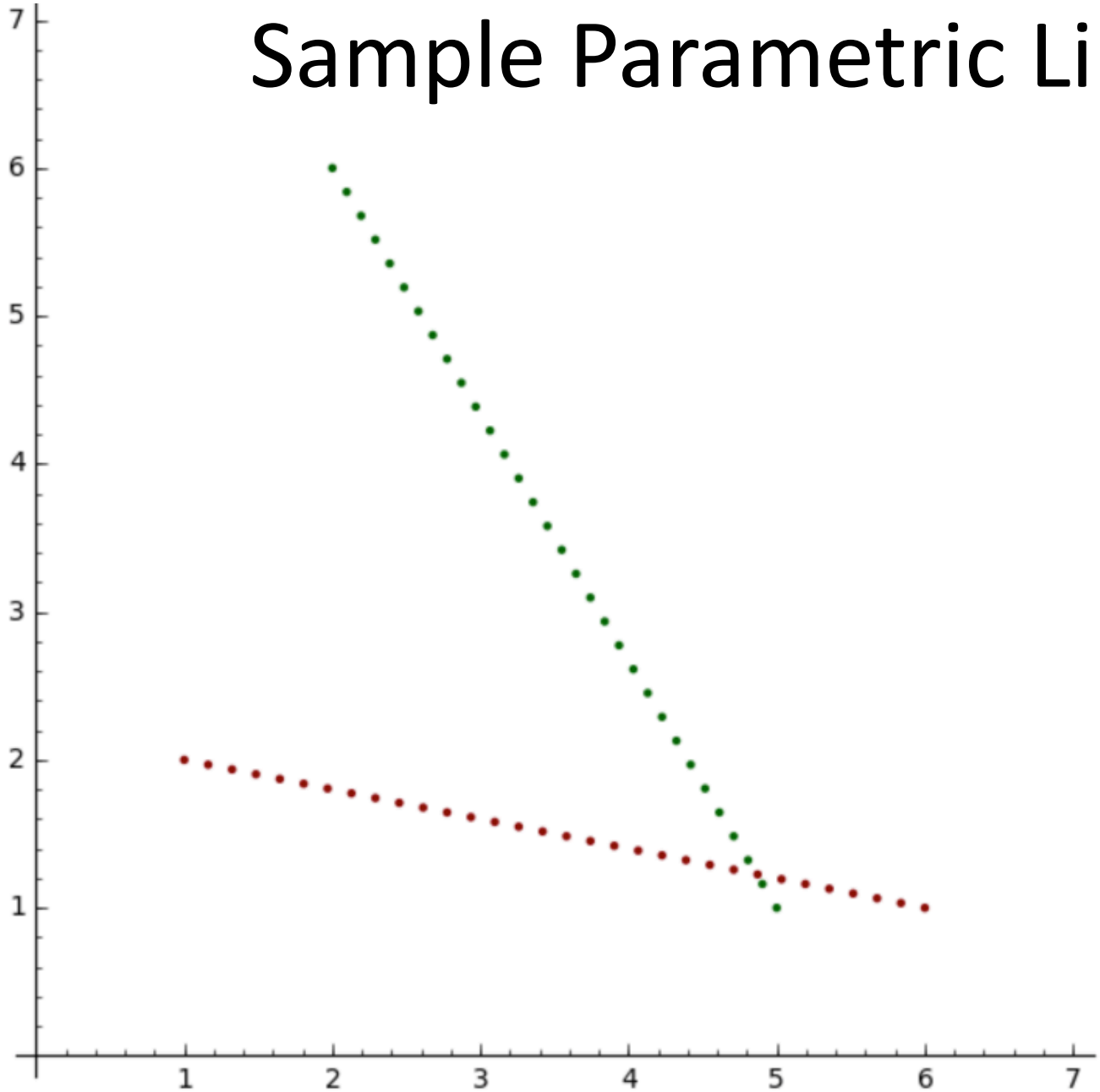


Any Value of *t* Will Do

```
LA = line([A1,A2], color="darkred")
LB = line([B1,B2], color="darkgreen")
LAP = parametric_plot((lapx(t),lapy(t)),(t,-10.0,10.0),color='darkred', alpha=0.25)
LBP = parametric_plot((lbpx(t),lbpy(t)),(t,-10.0,10.0),color='darkgreen', alpha=0.25)
bnd = 7
show(LA + LB + LAP + LBP, xmin=0, ymin=0, xmax=bnd, ymax=bnd, aspect_ratio=1)
```

- Consider how to draw un-bounded line
- Not really possible
  - This is suggesting graphics concept of clipping
- But for the illustration large bounds work
  - So notice t-values are beteen -10 and 10

Sample Parametric Line



Code to Sample

```
k  = 32.0
A1v = vector(A1); A2v = vector(A2);
B1v = vector(B1); B2v = vector(B2);
tss = [i/(k-1) for i in range(k)]
ptsa = [A1v + (A2v - A1v) * t for t in tss]
ptsb = [B1v + (B2v - B1v) * t for t in tss]
gptsa = [point(p,color='darkred') for p in ptsa]
gptsb = [point(p,color='darkgreen') for p in ptsb]
bnd = 7
show(sum(gptsa) + sum(gptsb), xmin=0, ymin=0, xmax=bnd, ymax=bnd, aspect_ratio=1)
```

- Python is making this ‘easy’
- How, by allowing enumeration of list elements
- Here 32 points are created that are evenly sampled along the two line segments

How About Intersection

- Pair of equations in two unknowns

$$-(ax_1 - ax_2)t + ax_1 = -(bx_1 - bx_2)s + bx_1$$

$$-(ay_1 - ay_2)t + ay_1 = -(by_1 - by_2)s + by_1$$

- We will shortly also consider turning this into a matrix inversion problem, but not yet.

Solution

- Courtesy of SageMath solve command

$$s = -\frac{ax_2(ay_1 - by_1) - ax_1(ay_2 - by_1) - (ay_1 - ay_2)bx_1}{(ay_1 - ay_2)bx_1 - (ay_1 - ay_2)bx_2 - ax_1(by_1 - by_2) + ax_2(by_1 - by_2)}$$

$$t = \frac{(ay_1 - by_2)bx_1 - (ay_1 - by_1)bx_2 - ax_1(by_1 - by_2)}{(ay_1 - ay_2)bx_1 - (ay_1 - ay_2)bx_2 - ax_1(by_1 - by_2) + ax_2(by_1 - by_2)}$$

- For our specific example

$$\left[s = \left(\frac{21}{22}\right), t = \left(\frac{17}{22}\right)\right]$$

Now Draw It

```
ax1  = A1[0]; ax2 = A2[0]; ay1 = A1[1]; ay2 = A2[1]
bx1  = B1[0]; bx2 = B2[0]; by1 = B1[1]; by2 = B2[1]
eq1v = ax1 + (ax2 - ax1) * t == bx1 + (bx2 - bx1) * s
eq2v = ay1 + (ay2 - ay1) * t == by1 + (by2 - by1) * s
resv = solve([eq1v,eq2v], s, t)
```

```
tstar = resv[0][1].rhs()
LA  = line([A1,A2], color="darkred")
LB  = line([B1,B2], color="darkgreen")
poi = point((lapx(tstar), lapy(tstar)),size=64,color='orange')
bnd = 7
show(LA + LB + poi, xmin=0, ymin=0, xmax=bnd, ymax=bnd, aspect_ratio=1)
```

Viola