Import Bayesian Astrometry package

In [1]:
```
import pyBA
```

Internal package dependencies are numpy, scipy and (for now) pymc. Emcee is used for MCMC.

In [2]:
```
import numpy as np
np.set_printoptions(linewidth=150)
```

Load test matching data; here, an ascii file with objects positions and uncertainties in two frames: x1, y1, sxx1, syy1, sxy1, x2, y2, sxx2, syy2, sxy2

In [3]:
```
!head examples/astrom_match_stats
data = np.loadtxt('examples/astrom_match_stats')
```

```
7.59170202086 39.2105075389 0.600606776843 0.498772876853 0.0772749135275
11.3389704959 1.68639963223 0.510030192875 0.381035008428 0.0290948865951
616.17810378 46.097566192 0.34979037475 0.396889444834 0.0113679491323
619.921658043 8.19724914345 0.431501097366 0.493875233442 -0.00948813542496
1303.8805473 82.986144411 1.25008605095 3.08527558022 -0.0498427884682
1307.82432997 44.8911147307 1.34468337788 2.86625166208 -0.0922806930183
445.769850011 45.8057684734 0.413382859514 0.394839790328 0.0150120344653
449.500174596 7.95294864833 0.489777394098 0.480311253281 -0.00558249167942
1315.72874007 56.2676815101 0.370982840848 0.401928414326 0.0125159948418
1319.51694304 18.1635035301 0.434706196479 0.476403435568 -0.00681030506961
1764.27527392 55.8970061759 0.405691006976 0.398025184995 0.0178786113093
1768.09820558 17.7159679349 0.472298930398 0.478794577697 -0.0211225286251
761.715583027 56.2993805336 0.425873432731 0.412385132197 0.018718631842
765.445254963 18.3914600637 0.470202751598 0.459383331586 -0.00902957821123
760.438997346 63.4416112989 0.429294099722 0.426359302577 0.0341116681046
764.046073677 25.4750103831 0.487377306297 0.436716152985 -0.0325625512263
942.998133819 76.7181347364 0.35071939738 0.406768392207 0.00912171728312
946.726993097 38.7248761981 0.439935865454 0.498503182577 -0.0130934517505
1411.90486531 68.9365073359 0.399006141549 0.423800065671 0.0158064517383
1415.73247186 30.8309485615 0.492342612462 0.491619368463 -0.00768513655543
```

Make object lists

There are 1399 objects in each frame. Here, a random subset of 100 is used.

In [4]:
```
nties = len(data)
print nties

# Use (random) subset of objects
nsamp = 100
ixs = np.random.permutation(nties)[0:nsamp]
```

```
1399
```

Read the data into numpy arrays of Bivarg (bivariate gaussian) objects. ***To add?: a routine can be written to make this a one liner, but that requires the user to provide a more rigid column format.***

In [5]:
```
objectsA = np.array( [ pyBA.Bivarg(mu=data[ix,0:2],sigma=data[ix,2:5]) for ix in ix
objectsB = np.array( [ pyBA.Bivarg(mu=data[ix,5:7],sigma=data[ix,7:10]) for ix in i
```

Regress background (affine) transformation

The first part of the astrometric analysis is to determine the affine part of the mapping. This incorporates translation, scaling and rotation about an arbitrary point; a seven-parameter model. pyBAST returns a 7D multivariate gaussian representing this mapping; the mean of the distribution is the maximum-likelihood affine mapping, and full covariance between the mapping parameters are provided in a 7x7 matrix.

To begin with, we have no information about the mapping between image frames, but the range of coordinates can be used to suggest a transformation. Formally, this is using the data more than once (about 1 + 1/nties times), but it has no practical effect apart from saving time during the background fitting.

In [8]:
```
S = pyBA.background.suggest_mapping(objectsA,objectsB)
print S.mu
print S.sigma
```

```
[ -4.26178143  38.03466637   0.           0.           0.           0.99972514
0.99994634]
[[ inf   0.   0.   0.   0.   0.   0.]
 [  0.  inf   0.   0.   0.   0.   0.]
 [  0.   0.  inf   0.   0.   0.   0.]
 [  0.   0.   0.  inf   0.   0.   0.]
 [  0.   0.   0.   0.  inf   0.   0.]
 [  0.   0.   0.   0.   0.  inf   0.]
 [  0.   0.   0.   0.   0.   0.  inf]]
```

So S.mu is the maximum likelihood mapping based on a cursory inspection of the data. S.sigma can be interpreted as an uninformative (uniform) prior, so that the widths in the parameters space are infinite. But we're only going to use the mean value (S.mu), as a starting point for exploration of parameter space.

Find maximum *a posteriori* affine mapping between frames, assuming a uniform prior

In [10]:
```
%timeit P = pyBA.background.MAP( objectsA, objectsB, mu0=S.mu, prior=pyBA.Bgmap(),
```

```
1 loops, best of 3: 6.52 s per loop
```

For 100 objects, it takes a few seconds to find the peak of the likelihood distribution and estimate the covariance between parameters. This is an O(n) process, so for 1400 objects it takes about a minute and a half.

In [11]:
```
print P.mu
print P.sigma
```

```
[ -3.64892497e+00   3.77235824e+01   2.55989013e-04   3.58632278e-04
3.91894791e-04   9.99929722e-01   1.00000670e+00]
[[  5.51030108e-01  -7.70927169e-02   6.80347253e-05   8.31222301e+00
-4.66091022e+02  -2.05442167e-04   1.10006121e-06]
 [ -7.70927169e-02   1.35083575e-01  -3.41649564e-05  -4.56846428e+02
-7.60992201e+00   3.20245947e-06  -7.60437531e-05]
 [  6.80347253e-05  -3.41649564e-05   2.90231727e-08   7.18372551e-05
-3.41228669e-05  -3.58192835e-09   6.56521118e-10]
 [  8.31222301e+00  -4.56846428e+02   7.18372551e-05  -1.78428540e+06
-3.09506530e+04  -2.11054736e-04   2.65141384e-06]
 [ -4.66091022e+02  -7.60992201e+00  -3.41228669e-05  -3.09506530e+04
1.82045314e+06  -3.33196680e-07  -7.65610736e-05]
 [ -2.05442167e-04   3.20245947e-06  -3.58192835e-09  -2.11054736e-04
-3.33196689e-07   1.74098850e-07   2.57373226e-10]
 [  1.10006121e-06  -7.60437531e-05   6.56521118e-10   2.65141384e-06
-7.65610736e-05   2.57373226e-10   3.49059099e-08]]
```

So the ML mapping value is not so different from the initial guess. But, now there is a full covariance matrix available as well.

Regressing the distortion map using gaussian processes

pyBAST treats astrometric mapping as a gaussian process, where the mean function to the GP is the affine transformation and the residuals of the observed tie objects relative to this background mapping are used to regress the remaining distortions. One of the benefits of this is that the uncertainty in the background mapping can be propagated through trivially: a gaussian process GP(M,C) = M + GP(0,C), so that a different choice of background mapping does not require a re-evaluation of the gaussian process.

In [15]:
```
D = pyBA.Dmap(P, objectsA, objectsB)
```

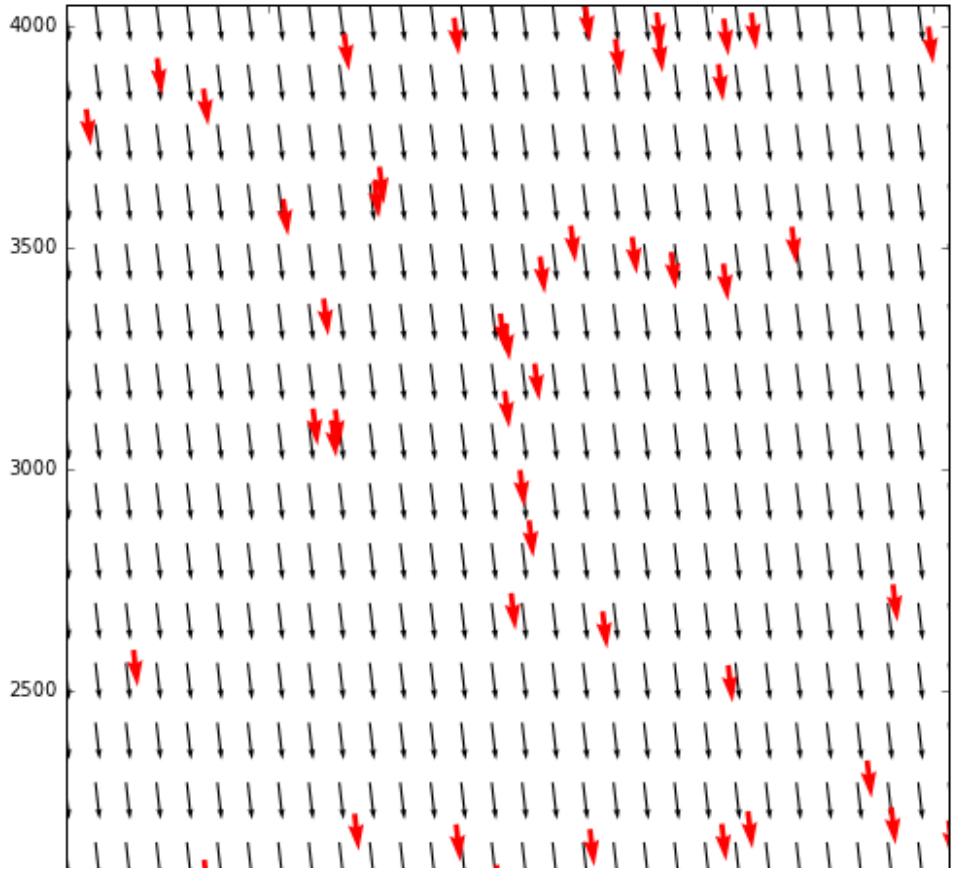
D is a distortion map object, which is a gaussian process with the background mapping attached as the mean function, and the sets of objects available for regression. At first, D knows nothing about the observed objects (except via the background mapping), so we condition the gaussian process based on these data.

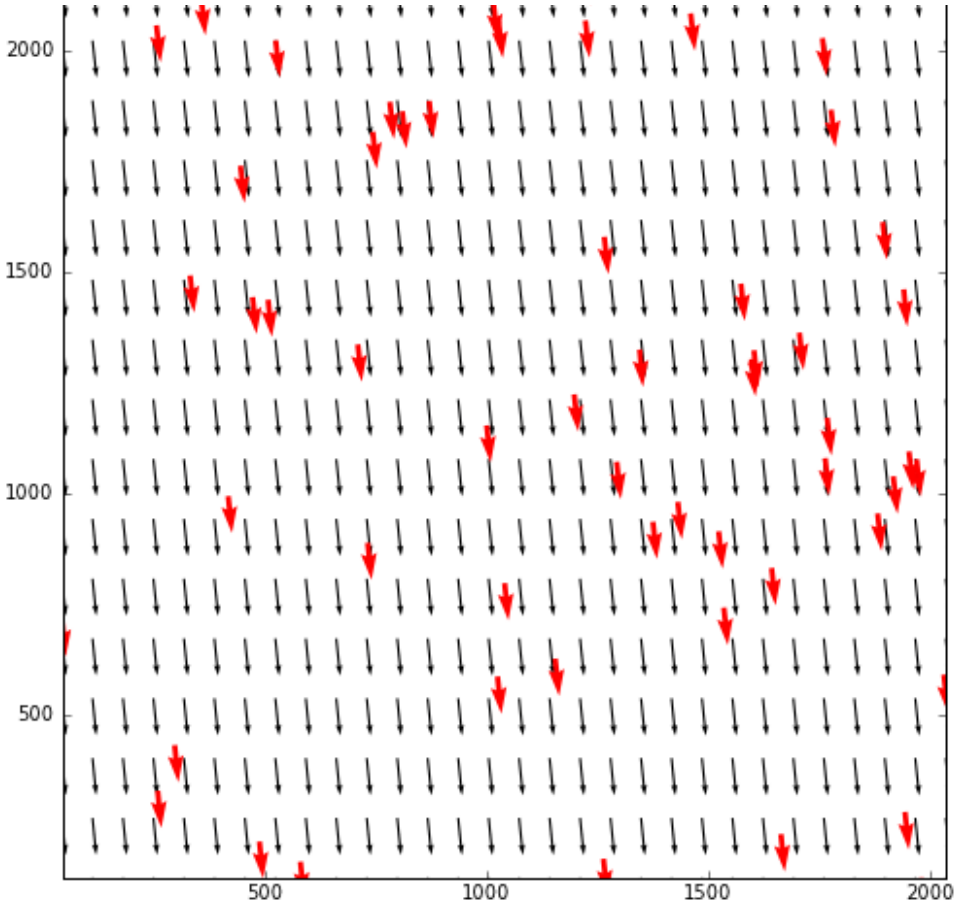
In [21]:
```
%timeit D = pyBA.Dmap(P, objectsA, objectsB); D.condition()
print D.hyperparams
```

```
1 loops, best of 3: 2.72 s per loop
{'amp': 1.4121788419148369, 'scale': 55.796346151927494}
```

With 100 objects the conditioning is very fast, but this is an O(n^3) process. (Once a GP is conditioned, trying to condition it again will throw an error, which is why D is recreated inside the timeit call.) The hyperparameters of the gaussian process are the overall amplitude of the variation and the scale of the covariance function. The process of conditioning also prepares the GP for regression at arbitrary locations.

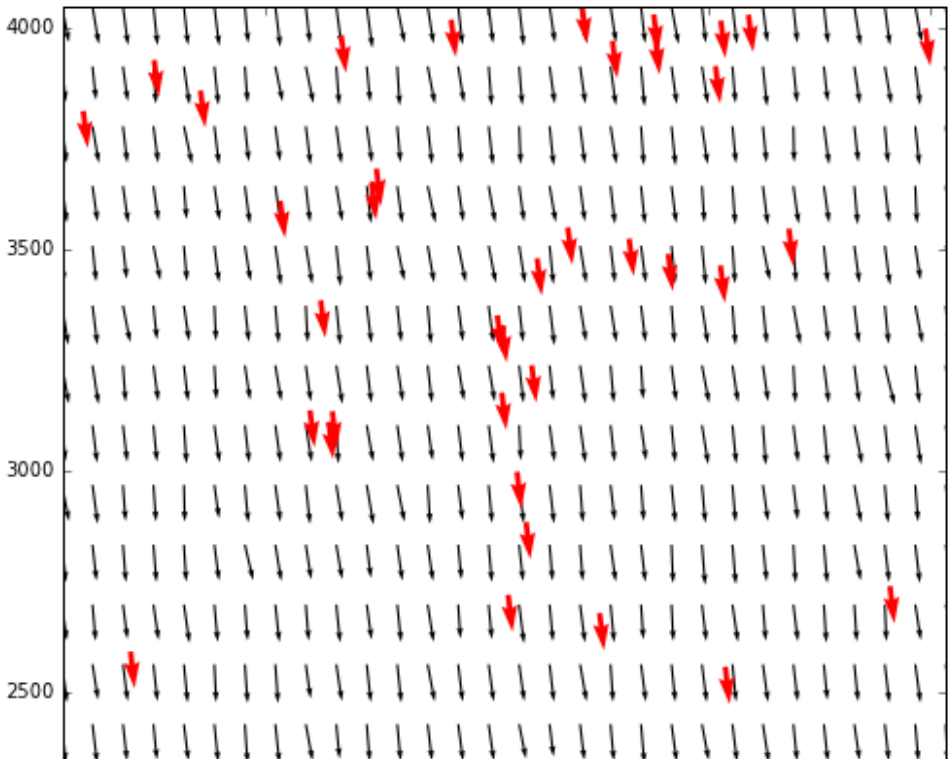
In [22]:
```
nres = 30 # Density of interpolation grid points
D.draw_background(res=nres)
```

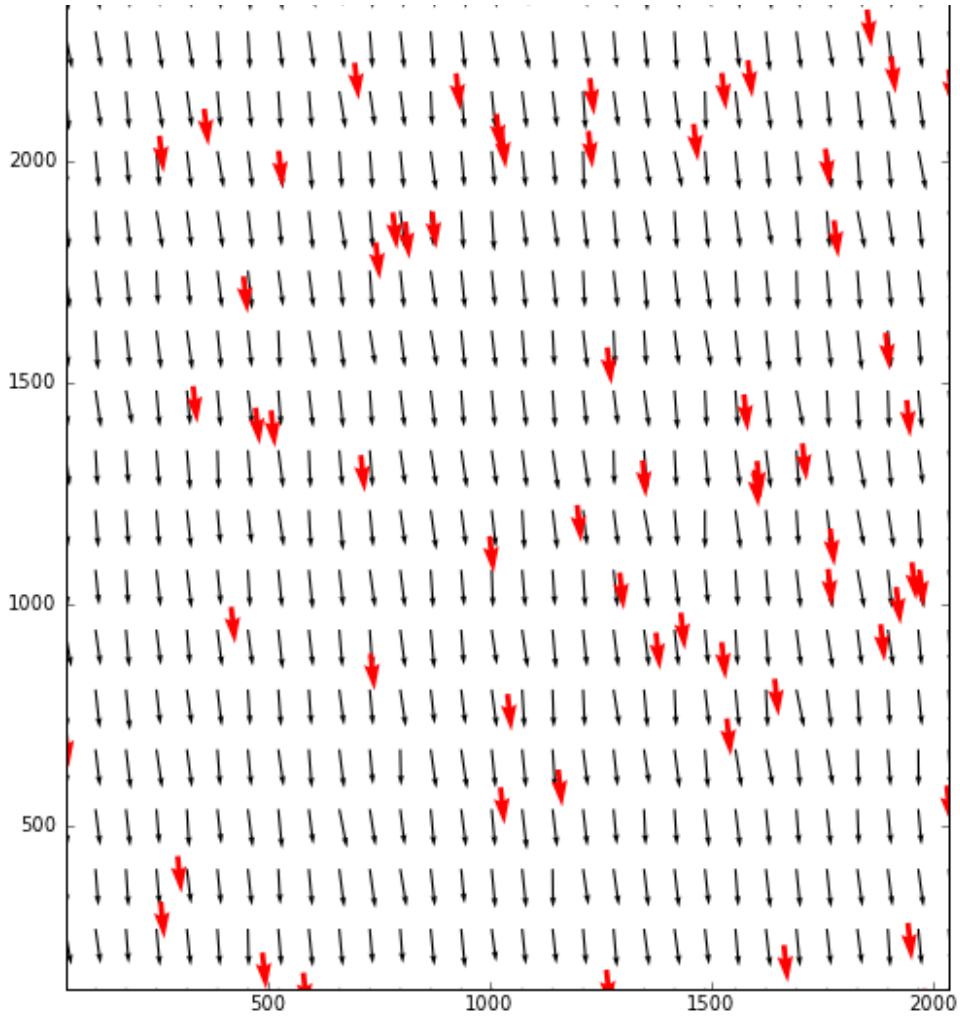




This plot shows the mean regression of the gaussian process on a 30 x 30 grid of points (the black arrows), with the mapping between observed tie objects shown in red. Error ellipses are included at the tip of each vector, but they are a little difficult to see. To get a better sense of the variation, we can draw a realisation from the gaussian process.

In [23]:
```
D.draw_realisation(res=nres)
```





Note that the vectors now wiggle around a bit. And, in particular, that the wiggling is much more stable in the vicinity of the observed data. You can think of a realisation as being a mapping between frames that gets generated each time a request for the astrometric mapping is made: by drawing many such realisations one can map a point in one frame (or set of points) to a distribution of points in the other.