## Matrix solution

First, select the number of answers, $n$, to the question being considered and define the fractions $f_i$ corresponding to answer $i$ and the likelihood, $r_{ij}$, that an object that truly correponds to answer $i$ will be assigned to answer $j$. If $i \neq j$ then that object has been misclassified. Obviously the $r_{ii} = 1 - \sum_{i \neq j} r_{ij}$ , and so are not independent. Similarly, one of the $f_i$ may be eliminated by recognising that $\sum_i f_i = 1$.

In [123]:
```
n = 4
f_i = symbols('f1:%i'%(n+1), real=True, positive=True)
r_ij = [[symbols('r%i%i'%(i, j), real=True, positive=True) for j in range(1,n+1)] for i in
```

Define the vector of observed answer fractions, $\boldsymbol{f}$, and the transfer matrix, $\boldsymbol{r}$. Both are given symbolic contents, $f_i$ and $r_{ij}$ , respectively, which could be replaced with actual values when known.

In [124]:
```
f = Matrix(f_i)
r = Matrix(r_ij)
```

The matrix $\boldsymbol{r}$ is defined with standard index numbering, but in our case we have $f_i = r_{ji}\, p_j$ , so we need to transpose the standard matrix.

In [125]:
```
rT = r.transpose()
rT
```

Out[125]:
$$\begin{pmatrix} r_{11} & r_{21} & r_{31} & r_{41} \\ r_{12} & r_{22} & r_{32} & r_{42} \\ r_{13} & r_{23} & r_{33} & r_{43} \\ r_{14} & r_{24} & r_{34} & r_{44} \end{pmatrix}$$

Following the definition of $\boldsymbol{r}$ above, we have that $\boldsymbol{f} = \boldsymbol{r}^T \cdot \boldsymbol{p}$ , and we wish to solve for $\boldsymbol{p}$. This may be achieved using standard methods, in this case Gaussian elimination utilising LU decomposition.

In [126]:
```
p = rT.LUsolve(f)
```

The products of the above matrix linear algebra operation is a complex expression for each element of $\boldsymbol{p}$, which may be greatly simplified, especially when $r_{ii} = 1 - \sum_{i \neq j} r_{ij}$ and $\sum_i f_i = 1$ are substituted later. The simplification can be very slow for highly nested expressions, but the following approach works ok for $n \leq 3$. For convenience we define a function to perform the "cleaning" of the maths and display of the results.

In [127]:
```
def clean(p):
    for i, pi in enumerate(p):
        p[i] = together(pi).simplify().collect(f_i)
    return p
```

In [128]:
```
for i in range(n):
    x = 1-sum(r[i,:max(0,i)])-sum(r[i,min(n,i+1):])
    p = p.subs(r[i,i], x)
# clean(p)  # beware this can be very slow as the expressions are quite complex for n > 3
```

In [129]:
```
p = p.subs(f[-1], 1-sum(f[:-1]))
#clean(p)  # beware this can be very slow as the expressions are quite complex for n > 3
#[x for x in p]  # display a bit more nicely
```

We can check that this is consistent with the $n = 2$ case computed previously by a non-matrix approach, or using the above approach with $n = 2$, by setting all terms involving $i > 2$ to zero. For convenience we define a function to do this.

In [130]:
```
def check_lower_n(nn):
    subslist = zip(f[nn:], [0]*n) + zip(r[nn:,:], [0]*n*n) + zip(r[:,nn:], [0]*n*n)
    subslist.append((f[nn-1], 1-sum(f[:nn-1])))
    pp = p.subs(subslist)
```

```
    clean(pp)  # this can be very slow as the expressions are quite complex for n > 3
    return [x for x in pp[:nn]]
```

In [131]:
```
p2 = check_lower_n(2)
p2
```

Out[131]:
$$\left[\frac{-f_1 + r_{21}}{r_{12} + r_{21} - 1}, \quad \frac{f_1 + r_{12} - 1}{r_{12} + r_{21} - 1}\right]$$

We can also see that it is consistent for the $n$ = 3 case.

In [132]:
```
p3 = check_lower_n(3)
p3[0]
```

Out[132]:
$$\frac{f_1(-r_{21} - r_{23} - r_{32} + 1) + f_2(-r_{21} + r_{31}) + r_{21}r_{31} + r_{21}r_{32} + r_{23}r_{31} - r_{31}}{r_{12}r_{23} + r_{12}r_{31} + r_{12}r_{32} - r_{12} + r_{13}r_{21} + r_{13}r_{23} + r_{13}r_{32} - r_{13} + r_{21}r_{31} + r_{21}r_{32} - r_{21} + r_{23}r_{31} - r_{23} - r_{31} - r_{32} + 1}$$

In [133]:
```
p3[1]
```

Out[133]:
$$\frac{f_1(-r_{12} + r_{32}) + f_2(-r_{12} - r_{13} - r_{31} + 1) + r_{12}r_{31} + r_{12}r_{32} + r_{13}r_{32} - r_{32}}{r_{12}r_{23} + r_{12}r_{31} + r_{12}r_{32} - r_{12} + r_{13}r_{21} + r_{13}r_{23} + r_{13}r_{32} - r_{13} + r_{21}r_{31} + r_{21}r_{32} - r_{21} + r_{23}r_{31} - r_{23} - r_{31} - r_{32} + 1}$$

In [134]:
```
p3[2]
```

Out[134]:
$$\frac{f_1(r_{12} + r_{21} + r_{23} - 1) + f_2(r_{12} + r_{13} + r_{21} - 1) + r_{12}r_{23} - r_{12} + r_{13}r_{21} + r_{13}r_{23} - r_{13} - r_{21} - r_{23} + 1}{r_{12}r_{23} + r_{12}r_{31} + r_{12}r_{32} - r_{12} + r_{13}r_{21} + r_{13}r_{23} + r_{13}r_{32} - r_{13} + r_{21}r_{31} + r_{21}r_{32} - r_{21} + r_{23}r_{31} - r_{23} - r_{31} - r_{32} + 1}$$

## Comparing with Bamford et al. (2009)

Take $i = 1$: spiral; $i = 2$: early-type; $i = 3$: other, then we can apply the above approach, with $n = 3$, together with the assumptions made in Bamford et al. (2009), and compare.

If we assume that the bias only results in some proportion of true spiral answers being assigned to the early-type option, then all the $r_{ij}$ are zero for $i \neq j$ and one for $i = j$, except for $r_{12}$ and $r_{11} = 1 - r_{12}$.

In [135]:
```
new_r = eye(n)
new_r[0,1] = r[0,1]
new_r[0,0] = 1 - r[0,1]
subslist = zip(r.reshape(1, n*n)[:], new_r.reshape(1, n*n)[:])
subslist.append((f[3], 0))
r.subs(subslist), f.subs(subslist)
```

Out[135]:
$$\left(\begin{pmatrix} -r_{12}+1 & r_{12} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ 0 \end{pmatrix}\right)$$

In [136]:
```
p_gz = p.subs(subslist)[:3]
clean(p_gz)
```

Out[136]:
$$\left[\frac{f_1}{-r_{12} + 1}, \quad \frac{f_1 r_{12} + f_2(r_{12} - 1)}{r_{12} - 1}, \quad f_3\right]$$

Bamford et al. (2009) determines and applies the correction in terms of the early-type to spiral ratio.

$$1/K = 10^C = \left\langle \frac{n_{\rm el}}{n_{\rm sp}} \right\rangle \Big/ \left\langle \frac{n_{\rm el}}{n_{\rm sp}} \right\rangle_{\rm true}$$

where the number ratios are determined in bins over which the correction is expected to be roughly constant.

In terms of the quantiites used above,

$$1/K = \left\langle \frac{f_2}{f_1} \right\rangle \Big/ \left\langle \frac{p_2}{p_1} \right\rangle.$$

Again, $\left\langle f_2/f_1 \right\rangle$ and $\left\langle p_2/p_1 \right\rangle$ are bin averages. However, rearranging:

$$K = \frac{p_2 f_1}{f_2 p_1}$$

In [168]:
```
K = (p_gz[1] * f[0] / (f[1] * p_gz[0]))
(K.subs(f[0], 1-f[1]).simplify())
```

Out[168]: $\frac{f_2 - r_{12}}{f_2}$

The correction is then applied as,

$$\frac{p_2}{p_1} = \frac{f_2}{f_1} K$$

and

$$p_1 = \frac{1}{p_2/p_1 + f_3/f_1 + 1} = \frac{1}{(f_2 - r_{12})/f_1 + f_3/f_1 + 1} = \frac{f_1}{(f_2 - r_{12}) + f_3 + f_1} = \frac{f_1}{1 - r_{12}}$$