Crlcalc - Berechnungen der Ausgabewerte

**Lens Properties**

**f -Fokus**

Berechnet die Brennweite der Linse (korreliert)

Unkorreliert: f = R/(2. \* N \* delta)

f\_corr = f / ( 1 - W \* (N\*\*2-1) / (6. \* f \* N ) )

Korrektur: Ab jetzt die allgemeingültige Formel.

factor = numpy.sqrt(f/(L\*1.0))

f\_corr = L \* factor / numpy.sin(1/factor))

**H – Verschiebung der Hauptebene**

Länge aller Linsen: L = N \* W

H = - L\*\*2 / f / 24

**D\_eff – „effective aperture“**

Energie → Wellenzahl: e2k = 2 \* numpy.pi \* energy / 12398.52 \* 10\*\*7

Zwischenrechnung: deff\_part = mu \* N \* R + 2\*N\*(e2k \* delta \* rough)\*\*2

D\_eff = 2 \* float( numpy.sqrt(2\*R\*\*2 / deff\_part \*

(1 - numpy.exp(-deff\_part / 2. \* (R\_0/R)\*\*2)) ) )

Neu:

a\_p = mu \* N \* R\_0\*\*2 / 2. / R + N \* delta \* e2k \* rough \* (R\_0/R)\*\*2

D\_eff = 2\*R\_0 \* numpy.sqrt( (1-numpy.exp(-a\_p)) /a\_p )

**T\_p – Transmission**

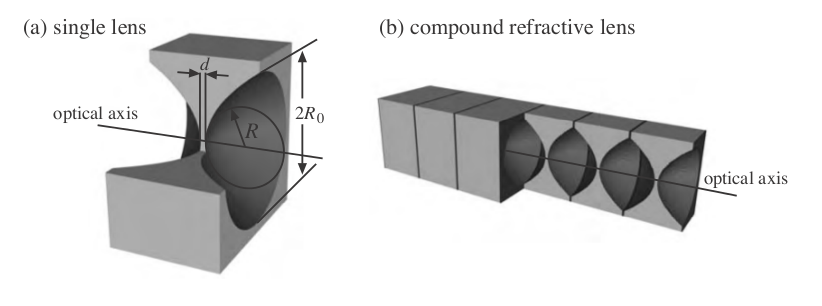
Zwischenrechnung:

Energie → Wellenzahl: e2k = 2 \* numpy.pi \* energy / 12398.52 \* 10\*\*7

a\_p = mu \* N \* R\_0\*\*2 / 2. / R + N \* delta \* e2k \* rough \* (R\_0/R)\*\*2

Korrektur: Faktor 2 in T\_p & roughness Zusatz in a\_p

T\_p: numpy.exp(-mu \* N \* d) \* (1 - numpy.exp(-2 \* a\_p)) / (2\*a\_p)

**N.A. - numeric aperture**

L2\_corr = (g \* f\_corr) /

(g - f\_corr)

N.A. = D\_eff /

L2\_corr / 2.

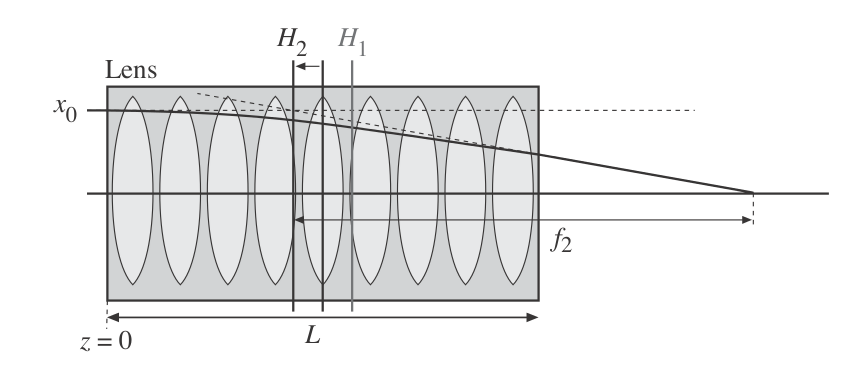
**Cross section**

Entspricht Sigma

T = T\_p

Sigma = numpy.pi \*

R\_0\*\*2 \* T



**results for given geometry**

**L2 – Entfernung Image zu Hauptebene**

L2\_corr = (g \* f\_corr) / (g - f\_corr)

**b – Entfernung Image zu verschobener Hauptebene**

b\_corr = L2\_corr + H

**L1 – Entfernung Source zu Hauptebene**

L1 = g – H

**Ish/Isv – Image Size**

Source Size b\_h/b\_v = b\_x

Is(x) = b\_x \* f\_corr / (g – f\_corr)

Korrektur: Nehme statt g L1.

**magn – Magnitude**

magn\_pre = mag = (L1 - f) / f

Wenn magn\_pre >=1 : magn = magn\_pre

Sonst: magn = 1 / magn\_pre

Korrektur: Vereinfachte Formel.

magn\_pre = L2\_corr / L1

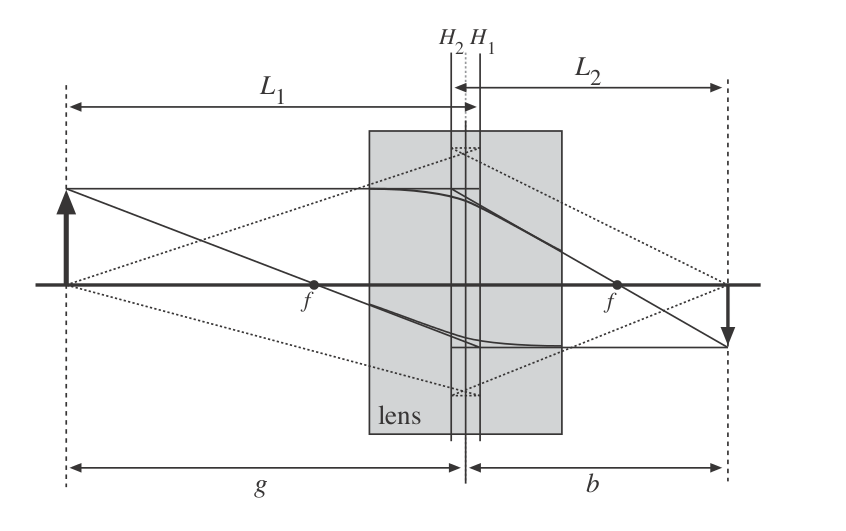
**gain**

gain = 4 \* R\_0 \*\* 2 \* T / b\_v / b\_h \* (g - f) \*\* 2 / f \*\* 2

**flux**

I = Intensity

Cross Section = sigma

flux = sigma \* intensity

**Correlated for diffraction, psf and roughness**

**diff limit - Auflösungslimit**

Im Programm hieß es Beugungsschreibchen

e2k siehe D\_eff

FWHM(s) = (2 \* s \* float(numpy.sqrt(2 \* numpy.log(2))))

*diff\_limit =* FWHM(2 / e2k \* L2/ Deff)

**depth of field**

sigma\_new = diff\_limit\_corr / NA\_corr

dof = 4 \* sigma\_new / 2. / float( numpy.sqrt(2 \* numpy.log(2))

**u\_dof - microbeam depth of field**

bg\_x = bgh/bgv = ISv/ISh

Bbgh/Bbgv = numpy.sqrt(diff\_limit \*\* 2 + bg\_x \*\* 2)

Wenn Bbgv >= bbgh: Bbgx = Bbgh

Wenn Bbgv < Bbgh: Bbgx = bbgv

sigma\_new = Bbg\_x / N.A.

u\_dof = 4 \* sigma\_new / 2. / float( numpy.sqrt(2 \* numpy.log(2))

**Isv\_corr / Ish\_corr – Image Size diffraction-korrigiert**

IS\_x\_corr = numpy.sqrt(diff\_limit\_corr \*\* 2 + bg\_x\_corr \*\* 2)

