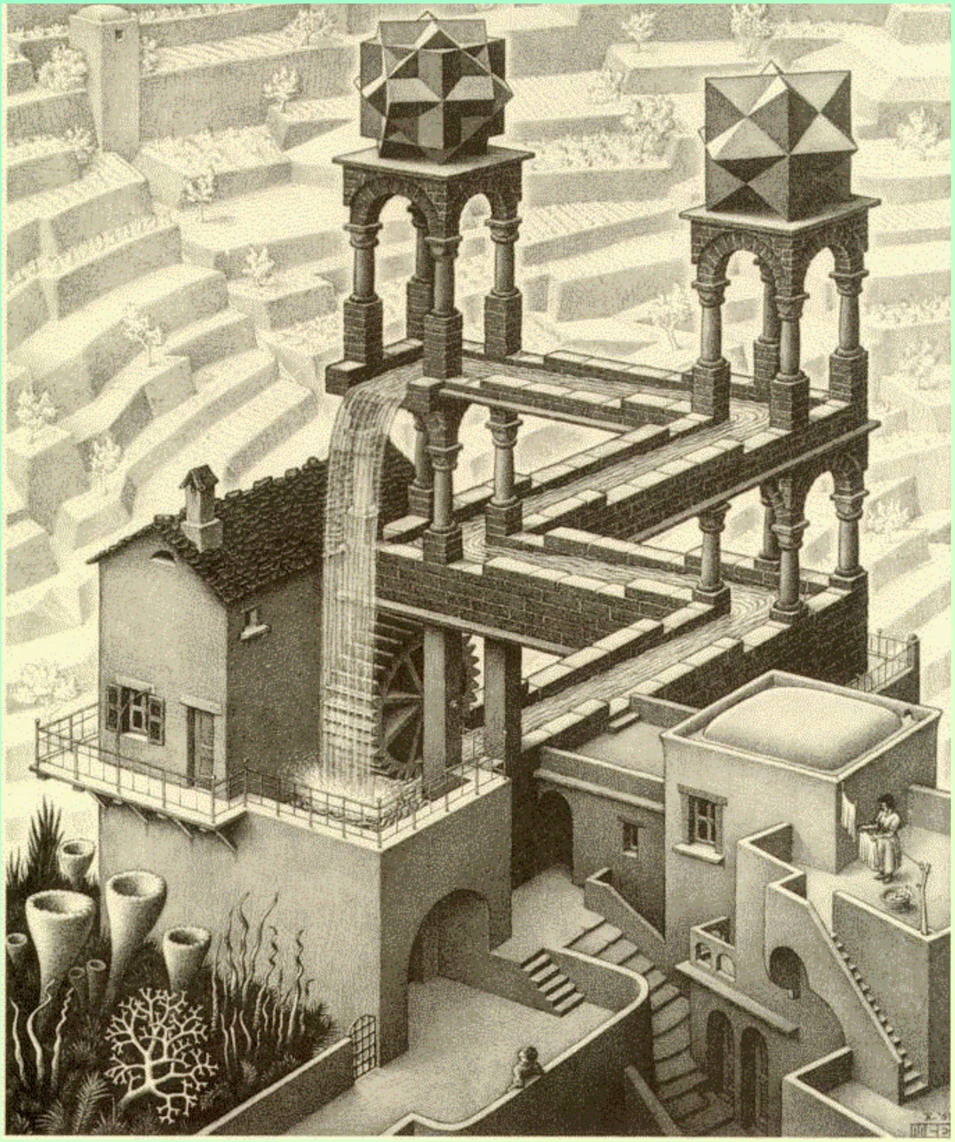
Interferences: Pathological and Otherwise



John Donovan
Micro Analytical Facility
CAMCOR

(Center for
Advanced Materials
Characterization in ORegon)

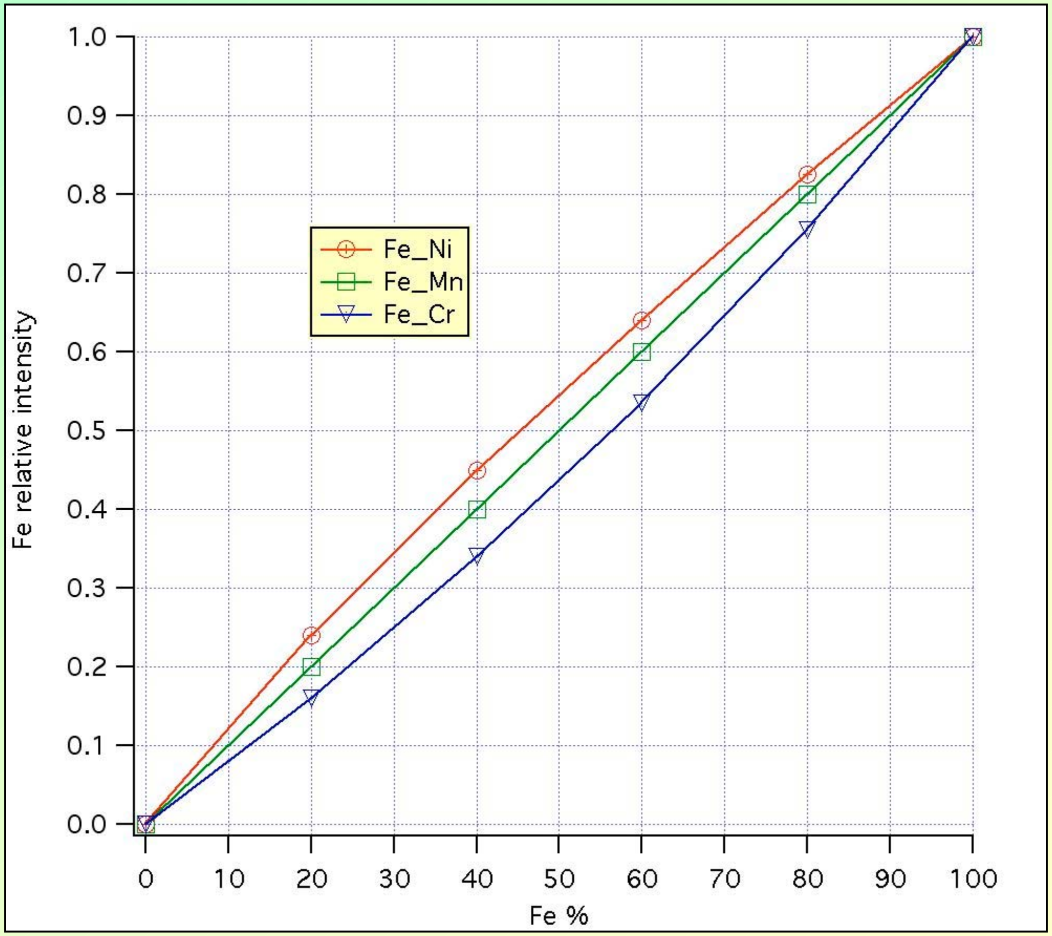
University of Oregon
Eugene, OR

(541)-346-4632
donovan@uoregon.edu
www.epmalab.uoregon.edu

Matrix Correction

$$C_i^{unk} \approx \frac{I_i^{unk}}{I_i^{std}} C_i^{std}$$

$$C_i^{unk} = \frac{I_i^{unk}}{I_i^{std}} \frac{ZAF_i^{unk}}{ZAF_i^{std}} C_i^{std}$$

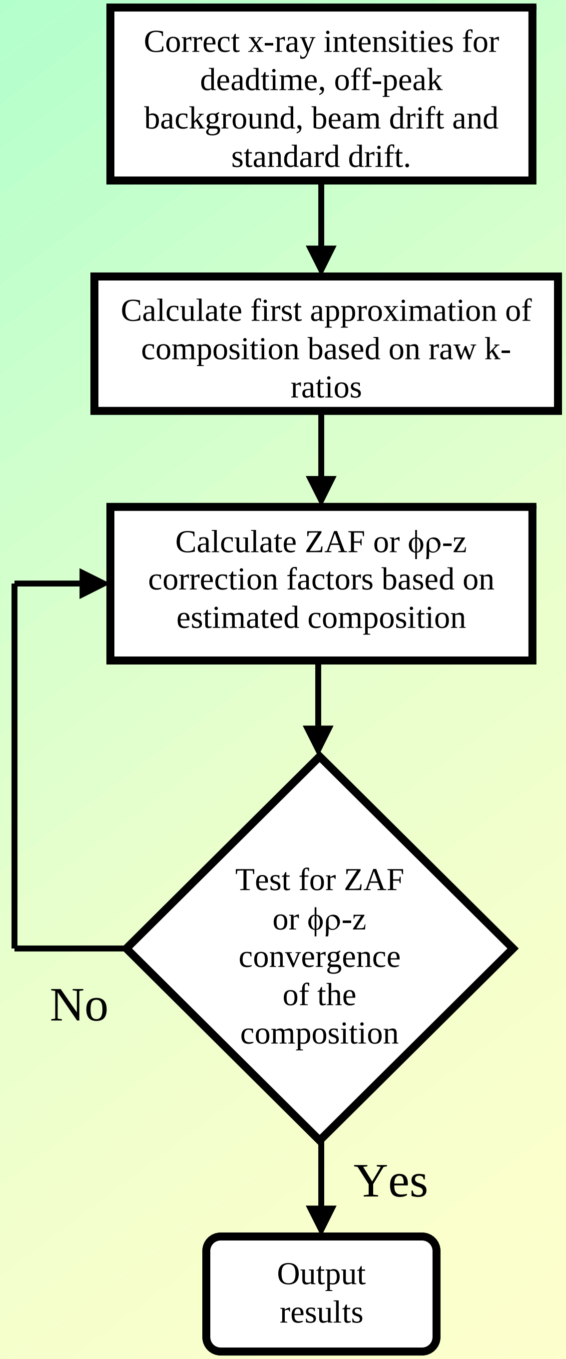


from Fournelle

Matrix Iteration...

```
Sub TestMatrixCalculateMatrix(mLastElm As Long, mLastChan As Long, mElements() As Long, mXrays() As Long, mCations() As Double, mOxygens() As Double, mAtomicWts() As Double, mAlpha1() As Double, mAlpha2() As Double, mAlpha3() As Double, mUnkcts() As Doubl
e, mStdPercents() As Double, mStdcts() As Double, mStdBetas() As Double, mCalculationFlags() As Long, mCalculationPercents() As Double, mUnkBetas() As Double, mUnkPercents() As Double, mOldPercents() As Double, mTotal As Double)
' Calculate matrix correction using a Bence-Albee iteration
' mLastElm& = number of emitting elements (measured intensities)
' mLastChan& = number of absorbing elements (not measured, only specified or calculated weight percents)
' mElements&() = element atomic numbers  (1 to mLastChan&)
' mXrays&() = xray lines (1=Ka, 2=Kb, 3=La, 4=Lb, 5=Ma, 6=Mb, 7=absorber only) (1 to mLastChan&)
' mCations#() = number of stoichiometric cations (1 to mLastChan&)
' mOxygens#() = number of stoichiometric oxygens (1 to mLastChan&)
' mAtomicWts#() = atomic weights (1 to mLastChan&)
' mAlpha1#(), mAlpha2#(), mAlpha3#() = matrix of alpha factor parameters
' mUnkcts#() = measured unknown intensities (1 to mLastElm&)
' mStdcts#() = standard intensities (from call to MatrixGetMatrix, SETUP.MDB database) (1 to mLastElm&)
' mStdPercents#() = standard percents (from call to MatrixGetMatrix, SETUP.MDB database) (1 to mLastElm&)
' mStdbetas#() = standard beta factors (from call to MatrixGetMatrix) (1 to mLastElm&)
' mCalculationFlags&() = absorbing element calculation flags (0 = specified concentration, 1 = by difference, 2 = by stoichiometry)
' mCalculationPercents&() = absorbing element specified weight percents (1 to mLastChan&)
'
' Returned parameters and arrays
' mUnkBetas#() = returned unknown beta factors (1 to mLastElm&) (must be initialized to 1.0)
' mUnkPercents#() = corrected weight percents (matrix corrected) (1 to mLastChan&)
' mOldPercents#() = original weight percents (1 to mLastChan&)
' mTotal# = weight percent total

ierror = False
On Error GoTo TestMatrixCalculateMatrixError

Const MAXMATRIXITER& = 30
Const MATRIXMINTOLER# = 0.005 ' in weight percent
Const MATRIXMINTOTAL# = 0.001 ' in weight percent

Dim bDifferenceElement As Boolean, bStoichiometricElement As Boolean
Dim nDifferenceElement As Long, nStoichiometricElement As Long
Dim chan As Long
Dim MatrixIter As Long

Dim maxdiff As Double, diff As Double
Dim stoichiometricoxygen As Double

ReDim p(1 To mLastChan&) As Double  ' stoichiometric ratio variable

' Determine if there is an element by diference and/or an element by stoichiometry
For chan& = mLastElm& + 1 To mLastChan&
If mCalculationFlags&(chan&) = 1 Then   ' element by difference
bDifferenceElement = True
nDifferenceElement& = chan&
End If

If mCalculationFlags&(chan&) = 2 Then   ' element by stoichiometry
bStoichiometricElement = True
nStoichiometricElement& = chan&
End If
Next chan&

' Bence-Albee iteration begins here
MatrixIter& = 1#
Do Until MatrixIter& >= MAXMATRIXITER&

' Calculate weight percents for analyzed elements
For chan& = 1 To mLastElm&
If mUnkBetas#(chan&) <= 0# Then GoTo TestMatrixCalculateMatrixBadUnkBetas
If mStdPercents#(chan&) <= 0# Then GoTo TestMatrixCalculateMatrixBadStdPercents
If mStdcts#(chan&) <= 0# Then GoTo TestMatrixCalculateMatrixBadStdCounts
If mStdBetas#(chan&) <= 0# Then GoTo TestMatrixCalculateMatrixBadStdBetas

mUnkPercents#(chan&) = (mUnkcts#(chan&) * mStdPercents#(chan&) * mUnkBetas#(chan&)) / (mStdcts#(chan&) * mStdBetas#(chan&))
Next chan&

' Load specified concentrations (could be zero wt% by default)
For chan& = mLastElm& + 1 To mLastChan&
mUnkPercents#(chan&) = 0#
If mCalculationFlags&(chan&) = 0 Then
mUnkPercents#(chan&) = mCalculationPercents#(chan&)
End If
Next chan&

' Sum stoichiometric oxygen (do not include oxygen from element by difference)
If bStoichiometricElement Then
stoichiometricoxygen# = 0#
For chan& = 1 To mLastChan&
If chan& <> nStoichiometricElement& Then
p#(chan&) = mOxygens#(chan&) / mCations#(chan&) * mAtomicWts#(nStoichiometricElement&) / mAtomicWts#(chan&)
stoichiometricoxygen# = stoichiometricoxygen# + mUnkPercents#(chan&) * p#(chan&)
End If
Next chan&
mUnkPercents#(nStoichiometricElement&) = stoichiometricoxygen#
End If

' Total all weight percents for difference element calculation
mTotal# = 0#
For chan& = 1 To mLastChan&
mTotal# = mTotal# + mUnkPercents#(chan&)
Next chan&

' Calculate element by difference based on current weight percents
If bDifferenceElement Then
If mTotal# < 100# Then
mUnkPercents#(nDifferenceElement&) = 100# - mTotal#
If bStoichiometricElement Then
p#(nDifferenceElement&) = mOxygens#(nDifferenceElement&) / mCations#(nDifferenceElement&) * mAtomicWts#(nStoichiometricElement&) / mAtomicWts#(nDifferenceElement&)
mUnkPercents#(nDifferenceElement&) = mUnkPercents#(nDifferenceElement&) / p#(nDifferenceElement&)
stoichiometricoxygen# = stoichiometricoxygen# + mUnkPercents#(nDifferenceElement&) * p#(nDifferenceElement&)
mUnkPercents#(nStoichiometricElement&) = stoichiometricoxygen#
End If
End If
mTotal# = 100#
End If

' Check for decent total
If mTotal# <= MATRIXMINTOTAL# Then GoTo TestMatrixCalculateMatrixBadTotal

' Compute change in weight percents since last iteration
maxdiff = 0#
For chan& = 1 To mLastChan&
diff# = Abs(mUnkPercents#(chan&) - mOldPercents#(chan&))
If diff# > maxdiff# Then maxdiff# = diff#
mOldPercents#(chan&) = mUnkPercents#(chan&)
Next chan&

' Check for convergence
If maxdiff# <= MATRIXMINTOLER# Then GoTo 6000

' Compute sum of alpha factors for each element (beta factors)
Call TestMatrixCalculateBetaFactors(mLastElm&, mLastChan&, mAlpha1#(), mAlpha2#(), mAlpha3#(), mUnkPercents#(), mUnkBetas#(), mTotal#)
If ierror Then Exit Sub

' Not converged, try again
MatrixIter& = MatrixIter& + 1
Loop

' If we get here, convergence failed
MsgBox "WARNING in TestMatrixCalculateMatrix- Convergence failed", vbOKOnly + vbExclamation, "TestMatrixCalculateMatrix"

' Converged, calculate total
6000:
mTotal# = 0#
For chan& = 1 To mLastChan&
mTotal# = mTotal# + mUnkPercents#(chan&)
Next chan&

Exit Sub

' Errors
TestMatrixCalculateMatrixError:
MsgBox Error$, vbOKOnly + vbCritical, "TestMatrixCalculateMatrix"
ierror = True
Exit Sub

TestMatrixCalculateMatrixBadUnkBetas:
msg$ = "WARNING in TestMatrixCalculateMatrix- Bad unknown beta factor for element" & Str$(mElements&(chan&)) & " " & Str$(mXrays&(chan&))
MsgBox msg$, vbOKOnly + vbExclamation, "TestMatrixCalculateMatrix"
ierror = True
Exit Sub

TestMatrixCalculateMatrixBadTotal:
msg$ = "WARNING in TestMatrixCalculateMatrix- Insufficient total"
MsgBox msg$, vbOKOnly + vbExclamation, "TestMatrixCalculateMatrix"
ierror = True
Exit Sub

TestMatrixCalculateMatrixBadStdPercents:
msg$ = "Insufficient standard percents for element" & Str$(mElements&(chan&)) & " " & Str$(mXrays&(chan&))
MsgBox msg$, vbOKOnly + vbExclamation, "TestMatrixCalculateMatrix"
ierror = True
Exit Sub

TestMatrixCalculateMatrixBadStdCounts:
msg$ = "Insufficient standard counts for element" & Str$(mElements&(chan&)) & " " & Str$(mXrays&(chan&))
MsgBox msg$, vbOKOnly + vbExclamation, "TestMatrixCalculateMatrix"
ierror = True
Exit Sub

TestMatrixCalculateMatrixBadStdBetas:
msg$ = "Bad standard beta factor for element" & Str$(mElements&(chan&)) & " " & Str$(mXrays&(chan&))
MsgBox msg$, vbOKOnly + vbExclamation, "TestMatrixCalculateMatrix"
ierror = True
Exit Sub

End Sub
```

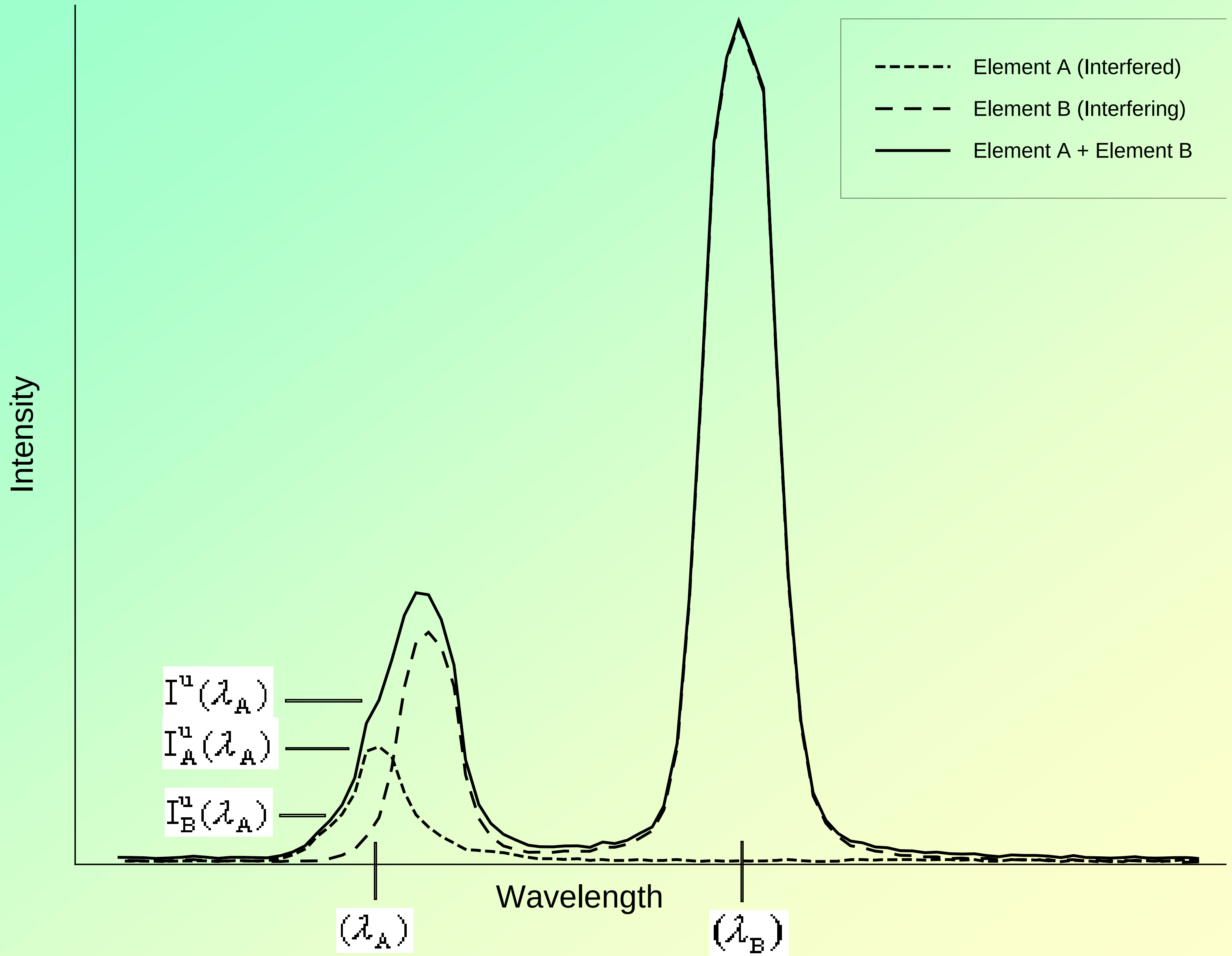
```
Sub TestMatrixCalculateBetaFactors(mLastElm As Long, mLastChan As Long, mAlpha1() As Double, mAlpha2() As Double, mAlpha3() As Double, mUnkPercents() As Double, mUnkBetas() As Double, mTotal As Double)
' This routine accepts an array of weight percents and alpha factors and returns an array of beta factors

ierror = False
On Error GoTo TestMatrixCalculateBetaFactorsError

Dim i As Long
Dim emitter As Long, absorber As Long
Dim betafraction As Double

ReDim wtfractions(1 To mLastChan&) As Double

' Convert to weight fractions
If mTotal# = 0# Then GoTo TestMatrixCalculateBetaFactorsBadTotal
For i& = 1 To mLastChan&
wtfractions#(i&) = mUnkPercents#(i&) / mTotal#
Next i&

' Calculate beta factors for "mLastElm" emitters and "mLastChan" absorbers
For emitter& = 1 To mLastElm&
mUnkBetas#(emitter&) = 0#

For absorber& = 1 To mLastChan&
betafraction# = (mAlpha1#(emitter&, absorber&) + wtfractions#(absorber&) * mAlpha2#(emitter&, absorber&) + wtfractions#(absorber&) ^ 2 * mAlpha3#(emitter&, absorber&)) * wtfractions#(absorber&)

' Sum beta factors fractions
mUnkBetas#(emitter&) = mUnkBetas#(emitter&) + betafraction#

Next absorber&
Next emitter&

Exit Sub

' Errors
TestMatrixCalculateBetaFactorsError:
MsgBox Error$, vbOKOnly + vbCritical, "TestMatrixCalculateBetaFactors"
ierror = True
Exit Sub

TestMatrixCalculateBetaFactorsBadTotal:
msg$ = "Total percent equals zero, cannot calculate beta factors"
MsgBox msg$, vbOKOnly + vbExclamation, "TestMatrixCalculateBetaFactors"
ierror = True
Exit Sub

End Sub
```



Other Compositionally Dependent Corrections

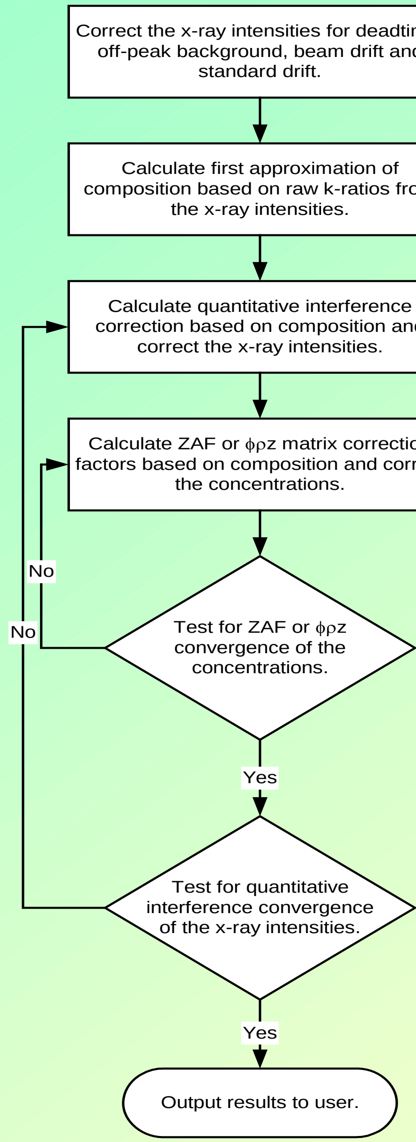
- Quantitative Spectral Interference Calculations
- Mean Atomic Number (MAN) Based Backgrounds
- Volatile Intensity Corrections
- Water by Difference (specified element effects)
- Compound Area-Peak Factor (APF) Calculations
- Blank (Zero) Value Corrections for Trace Elements

Spectral Interferences



Matrix Iteration

Flow Diagram of the Quantitative Iterated Interference Correction



Eq. 1

$$I_B^u(\lambda_A) \approx \frac{I_B^{\bar{s}}(\lambda_A)}{C_B^{\bar{s}}} \frac{I_B^u(\lambda_B)}{I_B^s(\lambda_B)} C_B^s$$

Gilfrich, *et al.*, 1978

Eq. 2

$$I_B^U(\lambda_A) = \frac{[ZAF]^{\bar{s}}_{\lambda_A}}{C_B^{\bar{s}}} \frac{C_B^u}{[ZAF]^u_{\lambda_A}} I_B^{\bar{s}}(\lambda_A)$$

Donovan, Rivers and Snyder, 1993

$$C_A^u = \frac{C_A^s}{[ZAF]^s_{\lambda_A}} [ZAF]^u_{\lambda_A} \frac{I^u(\lambda_A) - \frac{[ZAF]^{\bar{s}}_{\lambda_A}}{C_B^{\bar{s}}} \frac{C_B^u}{[ZAF]^u_{\lambda_A}} I_B^{\bar{s}}(\lambda_A)}{I_A^s(\lambda_A)}$$

Differences Between Eq. 1 and Eq. 2

Self-Interfering Analyses

	wt. % (nominal)	wt. % (uncorrected)	wt. % (Eq. 1)	wt. % (Eq. 2)
Ba Lα ↔ Ti Kα (PET)	Ba 33.15[3] Ti 11.69	33.26 ± 0.18 11.71 ± 0.08	33.08 11.59	33.08 ± 0.18 11.59 ± 0.08
Pb Lα ↔ As Kα	Pb 59.69[4] As 21.58	106.20 ± 0.33 41.38 ± 0.27	19.64 6.60	61.25 ± 1.97 22.15 ± 1.04

[3] Benitoite (BaTiSi3O9) is assumed stoichiometric : Si 20.38, Ba 33.15, Ti 11.69, O 34.896
[4] Shultenite ($HAsPbO_4$) is assumed stoichiometric : Pb 59.69, As 21.58, O 18.44. The oxygen concentration was measured at 19.8 wt. % and included in the matrix correction calculations.

	Pb Lα (cps)	As Kα (cps)	S Kα (cps)
PbS	1473.3 ± 11.5	1213.0 ± 3.8	1453.3 ± 9.3
GaAs	1624.7 ± 29.9	1771.7 ± 8.2	2.5 ± 1.2
FeS	14.0 ± 3.3	13.9 ± 3.7	4986.9 ± 26.3

Cascade Interference Analyses

	wt. % (nominal)	wt. % (uncorrected)	wt. % (Eq. 1)	wt. % (Eq. 2)
Ni K → Fe Kα Fe Kβ → Co Kα	Co 0.022[1]	0.089 ± 0.008	0.010	0.022 ± 0.008
Ti Kβ → V Kα V Kβ → Cr Kα	Cr 0.025[2]	0.268 ± 0.01	-0.020	0.021 ± 0.010

[1] SRM 1159 includes : Ni 48.2, Fe 51.0, C 0.007, Mn 0.30, P 0.003, S 0.003, Si 0.32, Cu 0.038, Cr 0.06, Mo 0.01
[2] SRM 654b includes : Ti 88.974, Al 6.34, V 4.31, Fe 0.23, Si 0.045, Ni 0.028, Sn 0.023, Cu 0.004, Mo 0.013, Zr 0.008

Pathological Interferences

Re la 1.43298
Zn ka 1.43652

“Self-Interferring”

Un   10 Zn-ReSCN gr2
TakeOff = 40 KiloVolts = 20 Beam Current = 20 Beam Size = 0

Results in Elemental Weight Percents

SPEC:	O	N	C	H
TYPE:	SPEC	SPEC	SPEC	SPEC
AVER:	1.900	5.000	4.200	.200
SDEV:	.000	.000	.000	.000

ELEM:	Cs	Fe	Zn	Re	S	Se	SUM
53	.000	.000	19.463	74.142	17.309	.000	122.214
55	.000	.007	20.459	74.986	16.357	.000	123.108
56	.000	.019	19.578	75.195	17.997	.000	124.089
AVER:	.000	.009	19.833	74.774	17.221	.000	123.137
SDEV:	.000	.010	.545	.558	.824	.000	
SERR:	.000	.006	.314	.322	.476	.000	
%RSD:	.1	113.3	2.7	.7	4.8	.1	
STDS:	834	730	660	575	730	660	

Results Based on 6 Atoms of re

SPEC:	O	N	C	H
TYPE:	SPEC	SPEC	SPEC	SPEC
AVER:	1.774	5.334	5.225	2.965
SDEV:	.013	.040	.039	.022

ELEM:	Cs	Fe	Zn	Re	S	Se	SUM
53	.000	.000	4.486	6.000	8.134	.000	34.048
55	.000	.002	4.663	6.000	7.600	.000	33.518
56	.000	.005	4.450	6.000	8.339	.000	34.005
AVER:	.000	.002	4.533	6.000	8.025	.000	33.857
SDEV:	.000	.003	.114	.000	.382	.000	
SERR:	.000	.001	.066	.000	.220	.000	
%RSD:	.8	113.2	2.5	.0	4.8	.8	

With Iterated Interference Correction

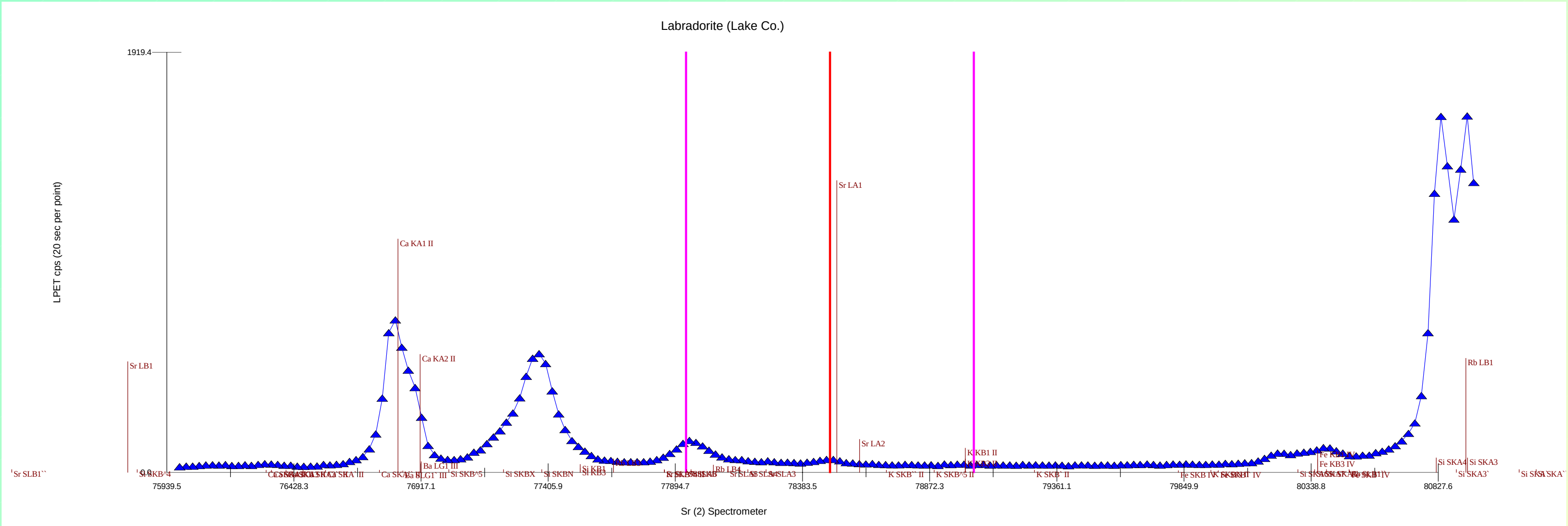
```
Un   10 Zn-ReSCN gr2
TakeOff = 40  KiloVolts = 20  Beam Current = 20  Beam Size = 0

Results in Elemental Weight Percents
SPEC:        O       N       C       H
TYPE:     SPEC    SPEC    SPEC    SPEC

AVER:    1.900   5.000   4.200    .200
SDEV:     .000    .000    .000    .000

ELEM:       Cs      Fe      Zn      Re       S      Se   SUM
    53    .000    .000   6.325  65.726  17.333    .000 100.683
    55    .000    .007   7.471  65.113  16.343    .000 100.233
    56    .000    .019   6.188  66.949  18.029    .000 102.486

AVER:     .000    .009   6.661  65.929  17.235    .000 101.134
SDEV:     .000    .010    .704    .935    .848    .000
SERR:     .000    .006    .407    .540    .489    .000
%RSD:       .1   113.3    10.6     1.4     4.9      .0
STDS:      834     730     660     575     730     660

Results Based on 6 Atoms of re
SPEC:        O       N       C       H
TYPE:     SPEC    SPEC    SPEC    SPEC

AVER:    2.013   6.050   5.926   3.363
SDEV:     .028    .085    .084    .047

ELEM:       Cs      Fe      Zn      Re       S      Se   SUM
    53    .000    .000   1.645   6.000   9.189    .000  34.236
    55    .000    .002   1.961   6.000   8.745    .000  34.274
    56    .000    .006   1.580   6.000   9.383    .000  34.053

AVER:     .000    .003   1.728   6.000   9.106    .000  34.188
SDEV:     .000    .003    .204    .000    .327    .000
SERR:     .000    .002    .118    .000    .189    .000
%RSD:      1.3   112.3    11.8      .0     3.6     1.4
```

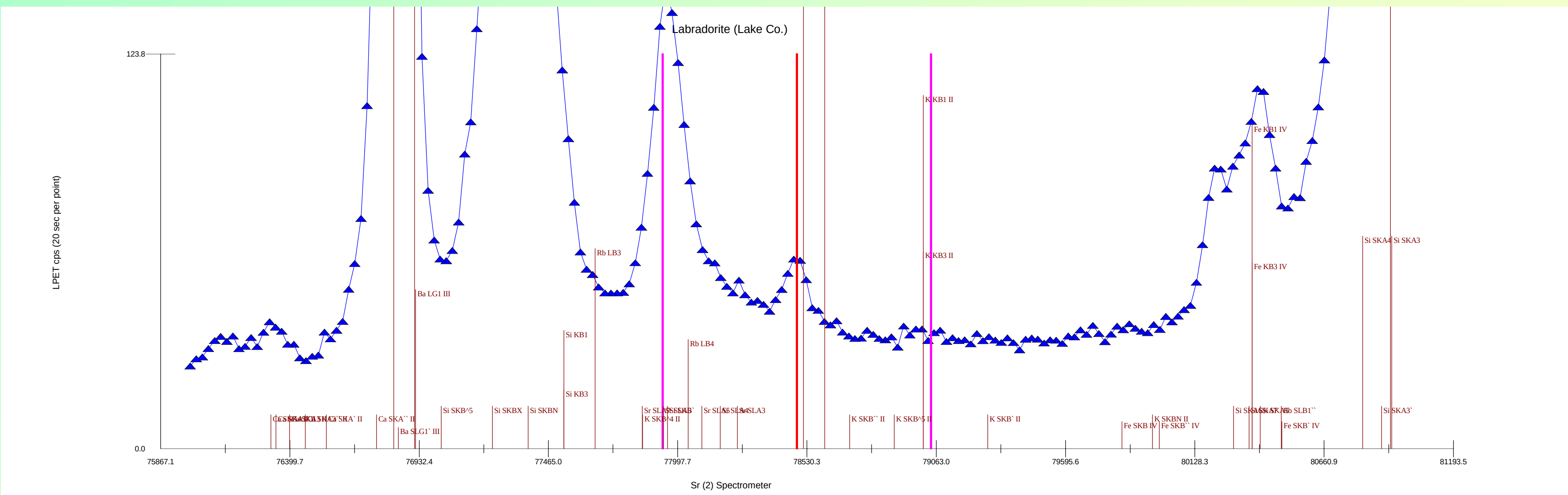
6 rhenium to 9 sulfur

Large magnitude “Self-Interferences”

Interfering Pair	Wavelength Region (Å)	Approximate Overlap (% @ 50/50)
Ba Lα ↔ Ti Kα	2.7	0.8 - 0.2
Pb Lα ↔ As Kα	1.17	150 - 65
Hg Lα ↔ Ge Kα	1.25	120 - 15
Ir Lα ↔ Ga Kα	1.34	70 - 30
Re Lα ↔ Zn Kα	1.43	140 - 60
Er Lα ↔ Co Kα	1.78	110 - 50
Eu Lα ↔ Mn Kα	2.1	15 - 5
In Lα ↔ K Kα	3.74	50 - 20
Th Mα ↔ Ag Lα	4.13	30 - 60
Bi Mα ↔ Tc Lα	5.1	50 - 70
Mo Lα ↔ S Kα	5.4	30 - 15

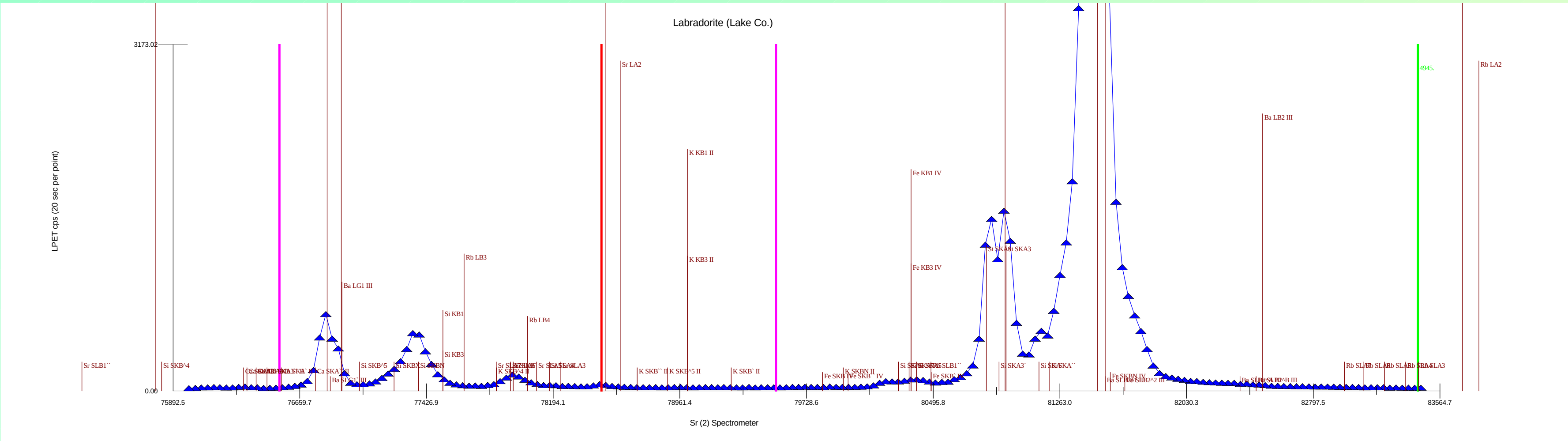
Typical (nasty) Interference, but first...

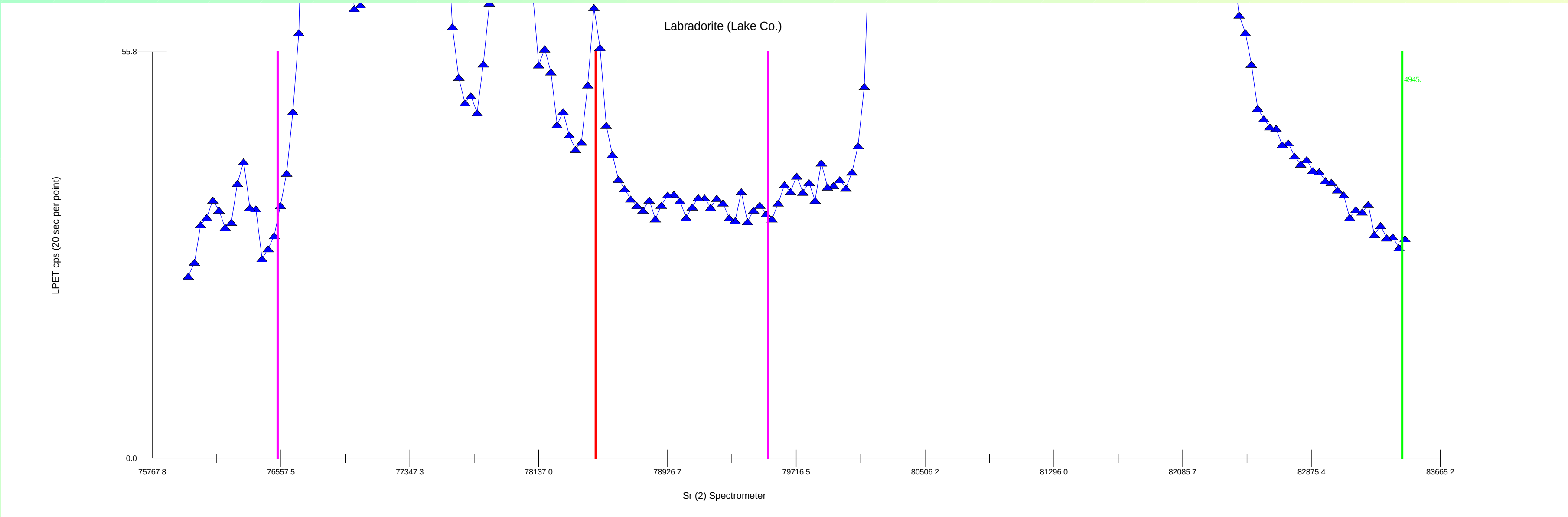




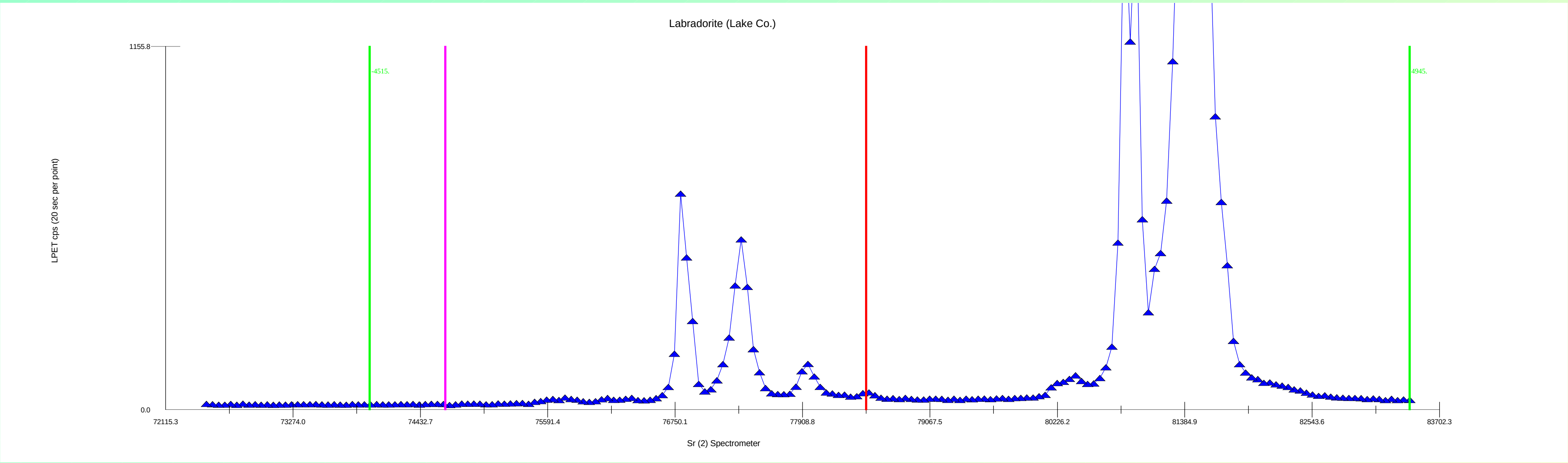
20 keV
100 nA
PET

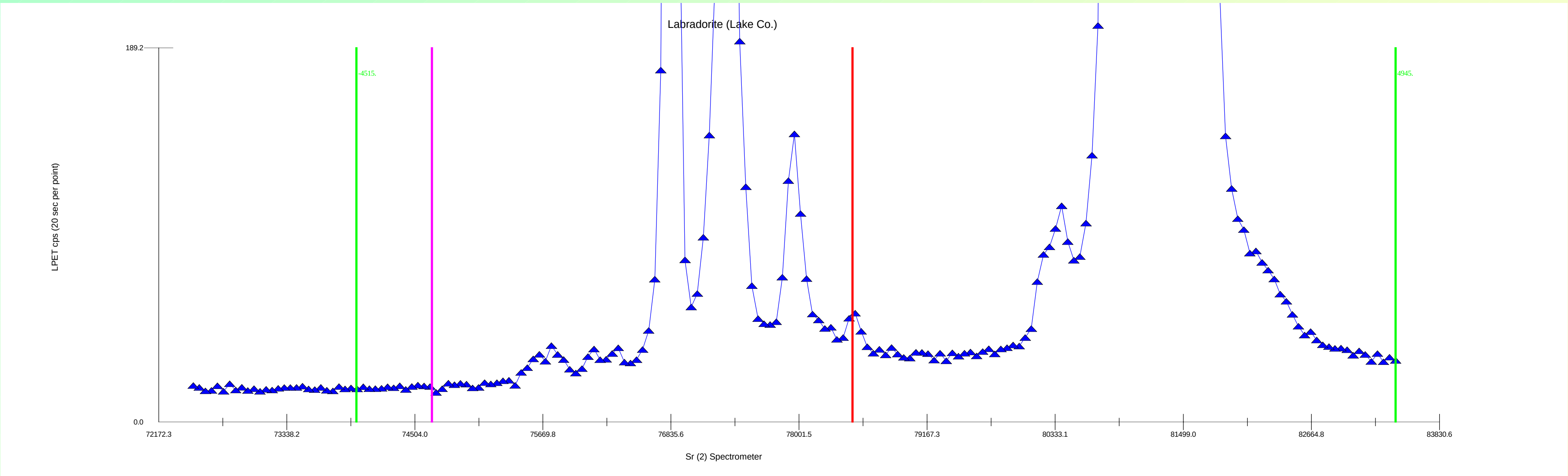
Are we measuring background yet?





How about now?





Polygonization...

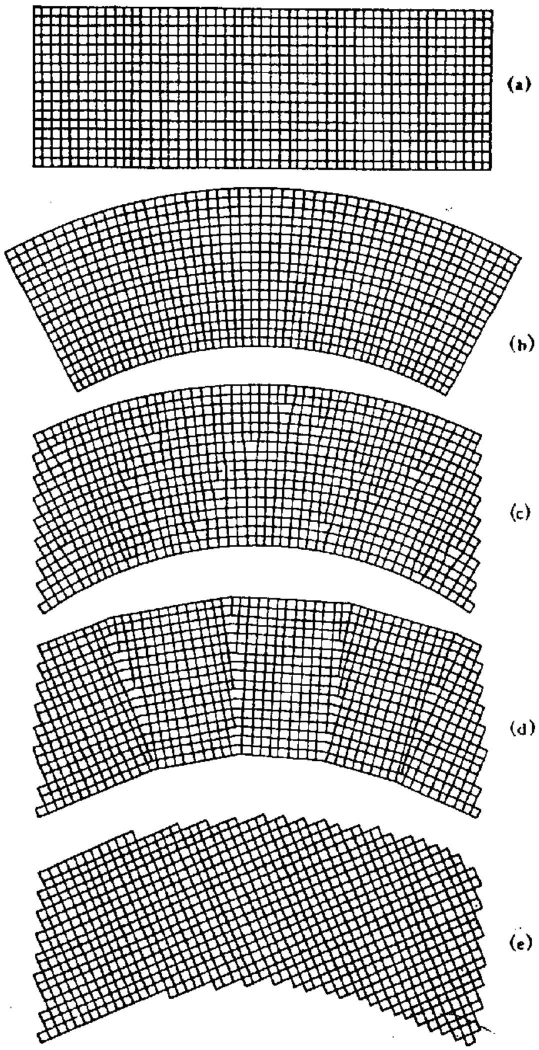
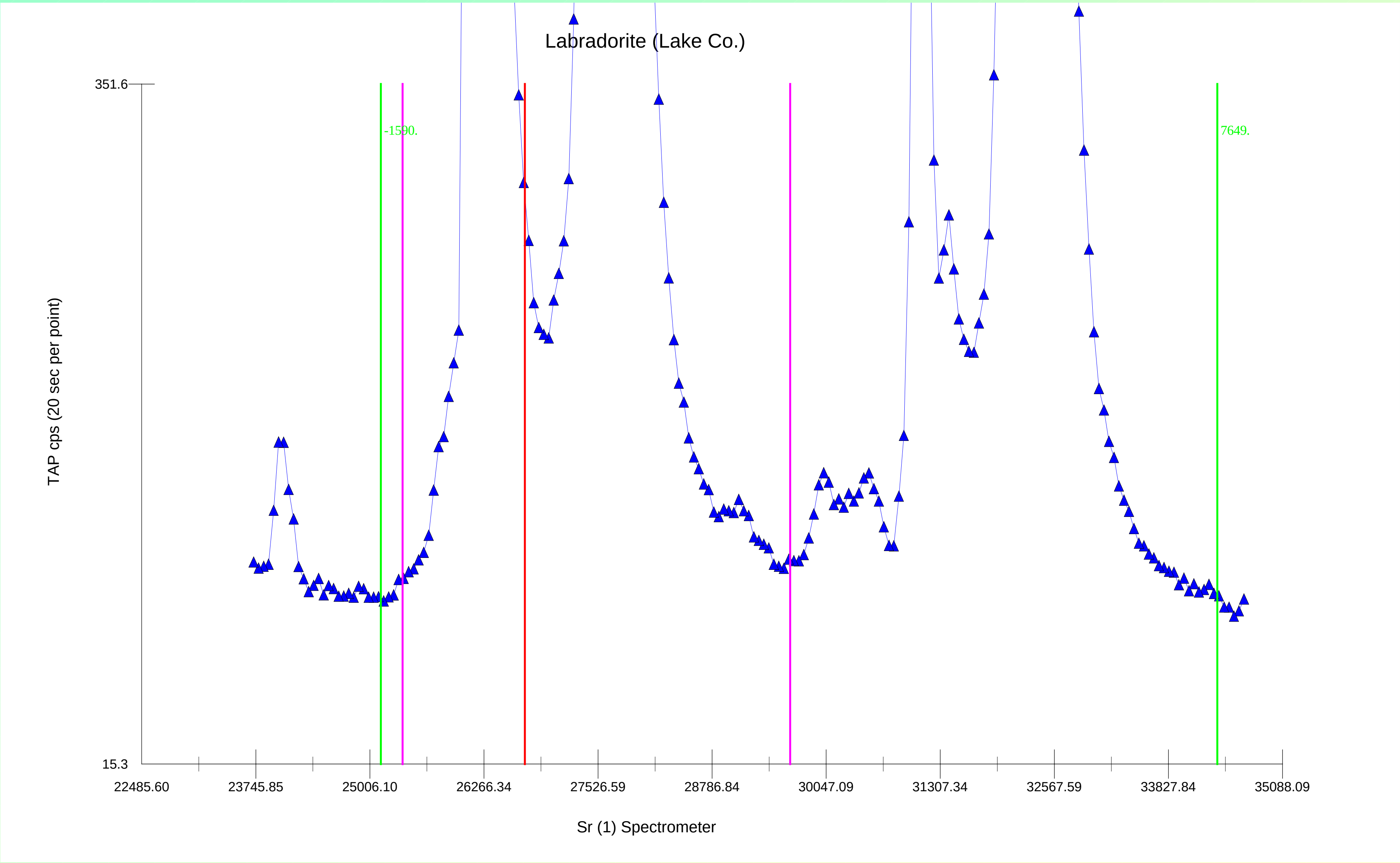


FIG. 1.27. Five states of a single crystal. (*a*) Unstrained. (*b*) Elastically bent. (*c*) Plastically bent. (*d*) Polygonized. (*e*) Recrystallized.

Nabarrro, F.R.N. (1967) Theory of Crystal Dislocations, Oxford, 821 pp.

Can we measure “true” background between Al and Si ka peaks?



Results

SiO2, WITHOUT Interference Correction:

	Sr (la) TAP	Sr (la) LPET	Sr (la) LPET	Sr (la) TAP
Average:	0.505	0.071	0.070	0.399
Std Dev:	0.010	0.001	0.003	0.004

SiO2, WITH Interference Correction:

	Sr (la) TAP	Sr (la) LPET	Sr (la) LPET	Sr (la) TAP
Average:	0.003	0.002	0.000	-0.001
Std Dev:	0.012	0.001	0.004	0.003

Conclusions

•Quantitative interference corrections require accurate background measurements

•Even “pathological” cascade and self-interfering interference correction are routine using iterative correction methods