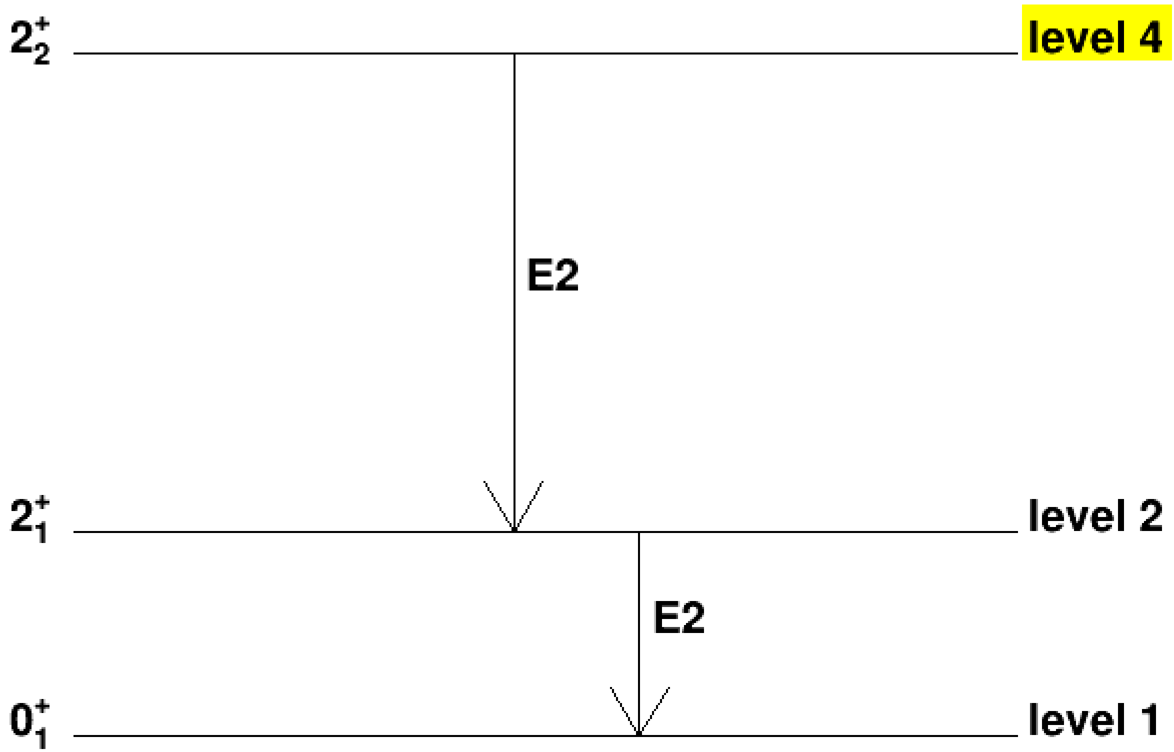
Gosia Gotchas

...Or what goes wrong, when you don't read the Gosia manual **very** carefully!

Nigel Warr, Gosia Workshop, April 2018

Missing level




```
OP,GOSI
...
LEVE
1  1  0  0.00
2  1  2  0.12
4  1  2  0.40
0,0,0,0
ME
2,0,0,0,0
  1  2 0.1      1.0  1.0
  2  4 0.1     -1.0  1.0
0,0,0,0.0
…

Yield file:
1 1 50 132 726 2 1.0
2 1 10  2
4 2 30  4
```

- No level 3 is defined but it is at least self-consistent!
- What does Gosia do?

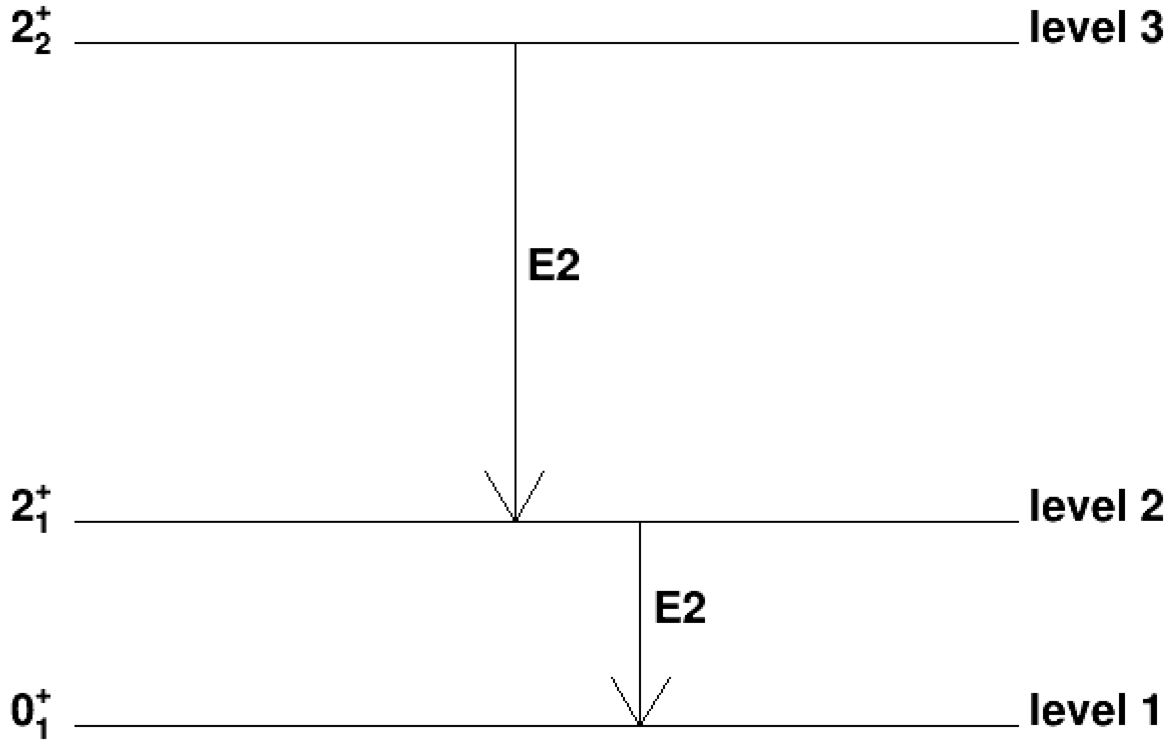
Missing level

- Gosia gets confused and gives the message:

```
ERROR-NO MATRIX ELEMENT BETWEEN STATES   4 AND   2
     THIS TRANSITION IGNORED
```

- In fact there is a matrix element defined between states 4 and 2, but internally, Gosia has got mixed up! It ends up looking up the wrong yield and actually accesses the 0th element of a Fortran array (which starts from 1!) This means it is accessing whatever happens to be in memory before that array!
- Anything it calculates is probably nonsense!

Invalid yield data




```
OP,GOSI
...
LEVE
1  1  0  0.00
2  1  2  0.12
3  1  2  0.40
0,0,0,0
ME
2,0,0,0,0
  1  2 0.1      1.0  1.0
  2  3 0.1     -1.0  1.0
0,0,0,0.0
…
```

Yield file:

```
1 1 50 132 726 3 1.0
2 1 10  2
3 2 30  4
3 1 20  4
```

The yield data refers to the $2_2^+ \rightarrow 0_1^+$, but the this transition does not have a corresponding matrix element! What does Gosia do?

Invalid yield data

- Gosia gives a **warning** in the output file

```
ERROR-NO MATRIX ELEMENT BETWEEN STATES   3 AND   1
     THIS TRANSITION IGNORED
```

- So you might think it is OK… but no, internally Gosia looks up the matrix element between levels 3 and 1 and gets index zero. Again this results in Gosia accessing the zeroth element of an array, which starts at one.
- This probably also gives nonsense!

Invalid yield data

- If you compile with bounds checking (e.g. *gfortran -fbounds-check*) it will stop with a runtime error:

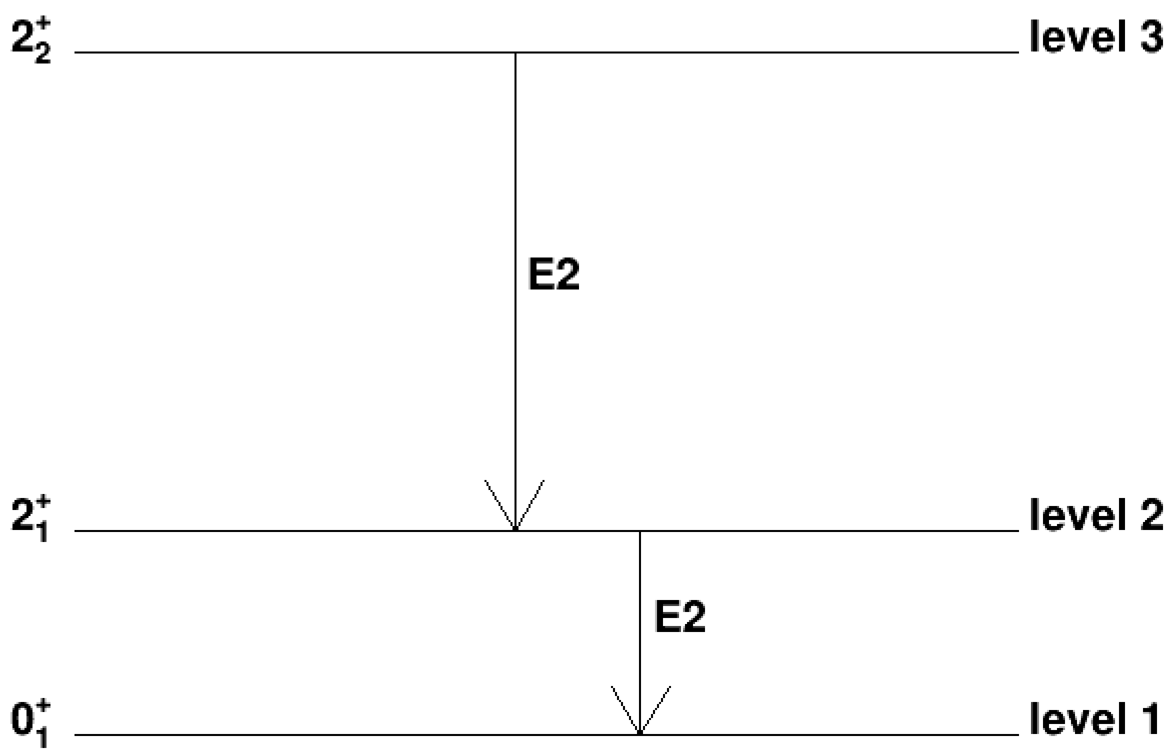
```
At line 10674 of file gosia_20081208.13.f
Fortran runtime error: Index '0' of dimension 1 of array 'iy' below lower bound of 1

Error termination. Backtrace:
#0  0x7fcc97295f7a
#1  0x7fcc97296b25
#2  0x7fcc97296ef7
#3  0x44d4e2
#4  0x41281e
#5  0x44aa69
#6  0x45f627
#7  0x484691
```

Invalid yield data

- If you compile with bounds checking (e.g. *gfortran -fbounds-check*) it will stop with a runtime error:

```
At line 10674 of file gosia_20081208.13.f
Fortran runtime error: Index '0' of dimension 1 of array 'iy' below lower bound of 1

Error termination. Backtrace:
#0  0x7f3979e49f3a in ???
#1  0x7f3979e4ab25 in ???
#2  0x7f3979e4aef7 in ???
#3  0x44d4e2 in szereg_ at gosia_20081208.13.f:10674
#4  0x41281e in ready_ at gosia_20081208.13.f:10483
#5  0x44aa69 in adhoc_ at gosia_20081208.13.f:13032
#6  0x45f627 in gosia at gosia_20081208.13.f:1276
#7  0x484691 in main at gosia_20081208.13.f:3542
```

Invalid branching ratios




```
OP,GOSI
...
LEVE
1  1  0  0.00
2  1  2  0.12
3  1  2  0.40
0,0,0,0
ME
2,0,0,0,0
  1  2 0.1      1.0  1.0
  2  3 0.1     -1.0  1.0
0,0,0,0.0
…

OP,YIEL
…
3 2 3 1 0.10 0.01
```

The branching ratio relates the $2_2^+ \rightarrow 2_1^+$ to the $2_2^+ \rightarrow 0_1^+$, but the latter transition does not have a corresponding matrix element! What does Gosia do?

Invalid branching ratios

- Gosia does **not** give an error or even a warning!
- Gosia happily writes a message into the output file:

```
                    BRANCHING RATIOS

   NS1   NF1   NS2   NF2   RATIO(1:2)      ERROR
    3     2     3     1     0.10000       0.01000
   BRANCHING RATIOS ARE TAKEN WITH WEIGHT    0.100000E+01
```

- However, Gosia stores these indices in its internal data. Later, when it actually tries to calculate the branching ratio, it looks up the matrix elements corresponding to the two transitions. When it does this one of the indices is zero and once again it accesses the zeroth element of a Fortran array.
- Once again, you probably calculate nonsense!

Invalid branching ratios

- If you compile with bounds checking (e.g. *gfortran -fbounds-check*) it doesn't crash at the OP,YIEL stage, but will probably crash in another OP, making debugging very hard!
- Moreover, even if it doesn't crash, there's no guarantee that it is performing the correct calculation!

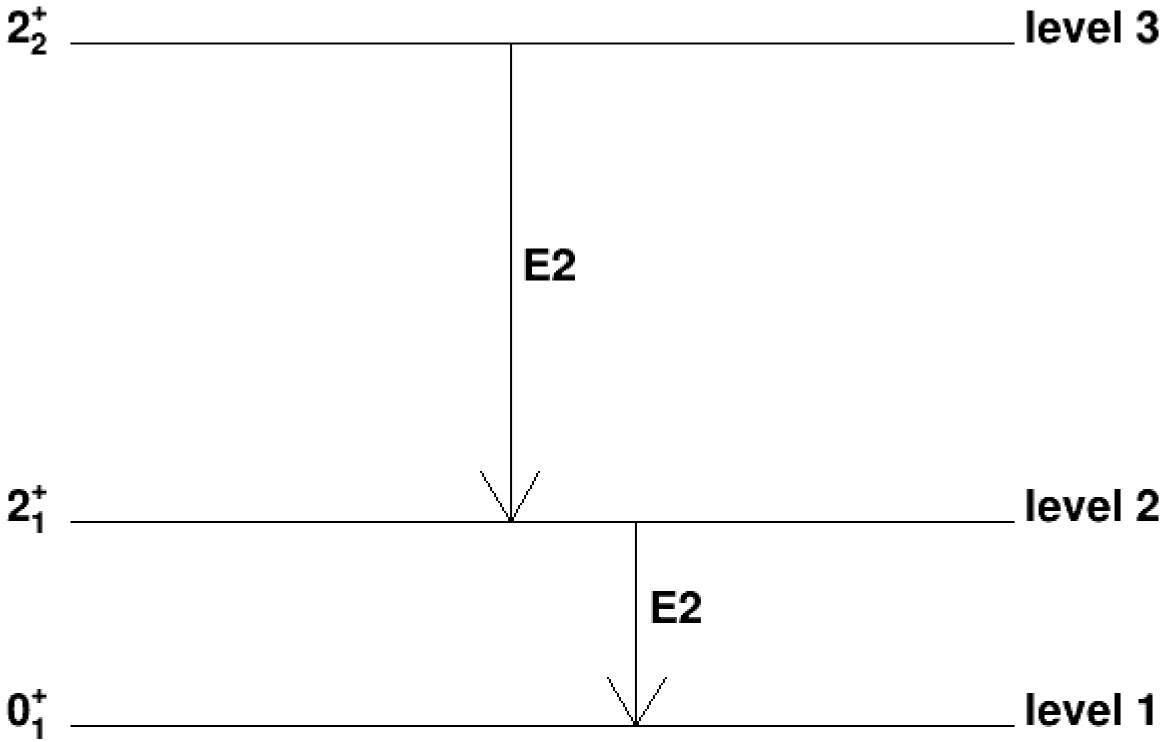
Invalid branching ratios

- Specifying a branching ratio involving an E4, E5 or E6 is not allowed. Gosia can use matrix elements with these multipolarities in the **excitation** but not in the de-excitation! This is implied by the text on page 2 of the manual:

> Abstract
>
> The wealth of Coulomb excitation data collected using present day experimental techniques makes feasible a model independent determination of almost all E1, E2, E3, M1, and M2 matrix elements connecting low-lying collective nuclear levels populated by electromagnetic excitation. A semiclassical coupled-channel Coulomb excitation least-squares search code, GOSIA, has been developed to simulate...

- If you define a branching ratio with, for example, an E4, it will look up the decay, but as it only has E1-3 and M1, M2 it will get zero. It then accesses the 0^{th} element of an array again!
- Using -fbounds-check also catches this.

Invalid lifetime data




```
OP,GOSI
...
LEVE
1  1  0  0.00
2  1  2  0.12
3  1  2  0.40
0,0,0,0
ME
2,0,0,0,0
  1  2 0.1      1.0  1.0
  2  3 0.1     -1.0  1.0
0,0,0,0.0
…

OP,YIEL
…
4,2,0.1
```

- What happens if a lifetime entry in OP,YIEL refers to a non-existant level? Here we have set the lifetime of level 4 to 2.0(1) ps.

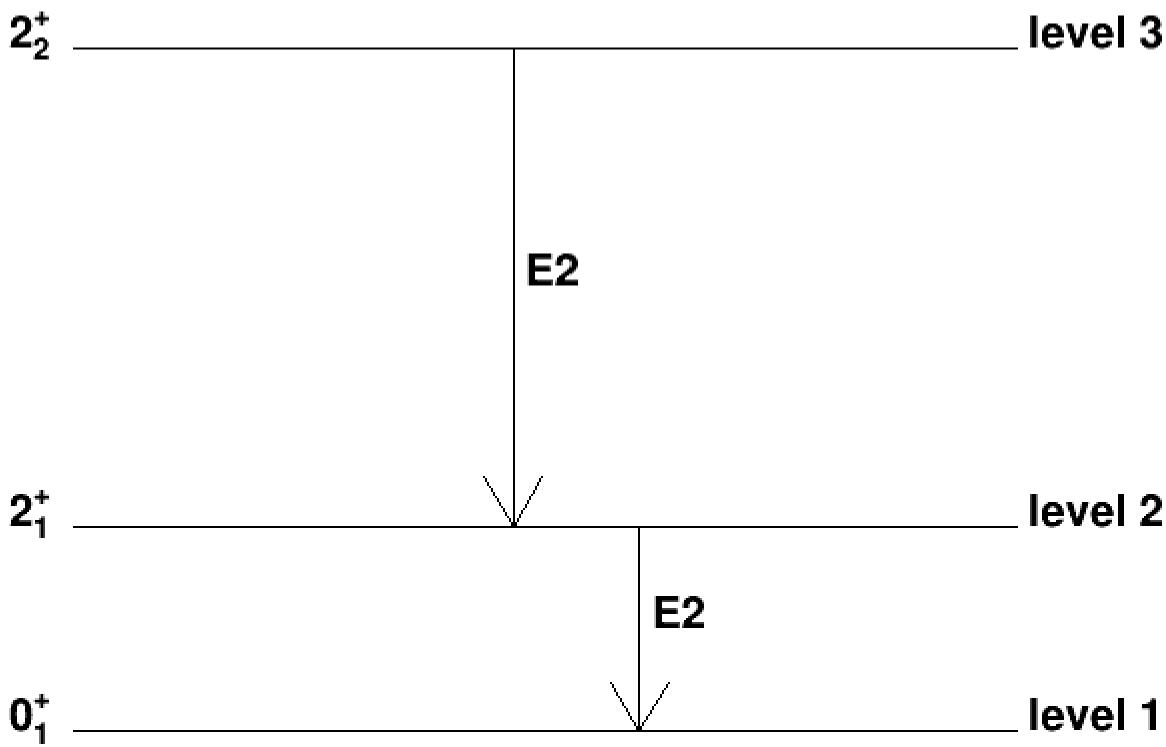
Invalid lifetime data

- Again, Gosia gives no error or warning, just the message in the output file:

```
                    LIFETIMES(PSEC)


    LEVEL        LIFETIME     ERROR

      4          2.00         0.10

           LIFETIMES ARE TAKEN WITH WEIGHT    0.100000E+01
```

- And again, it doesn't fail in the OP,YIEL step, but it can cause memory corruption or incorrect calculation later on.

Invalid mixing ratio data




```
OP,GOSI
...
LEVE
1  1  0  0.00
2  1  2  0.12
3  1  2  0.40
0,0,0,0
ME
2,0,0,0,0
  1  2 0.1      1.0  1.0
  2  3 0.1     -1.0  1.0
0,0,0,0.0
…

OP,YIEL
…
3,2,1.5,0.1
```

- What if we provide the mixing ratio for the $2_2^+ \rightarrow 2_1^+$ state, but we forgot to add the M1 matrix element?

Invalid mixing ratio data

- Again, Gosia gives no warning or error and happily writes a message in the output file:

```
                    EXPERIMENTAL E2/M1 MIXING RATIOS


          TRANSITION          DELTA        ERROR

           3---  2            1.5000       0.1000

          E2/M1 MIXING RATIOS ARE TAKEN WITH WEIGHT    0.100000E+01
```

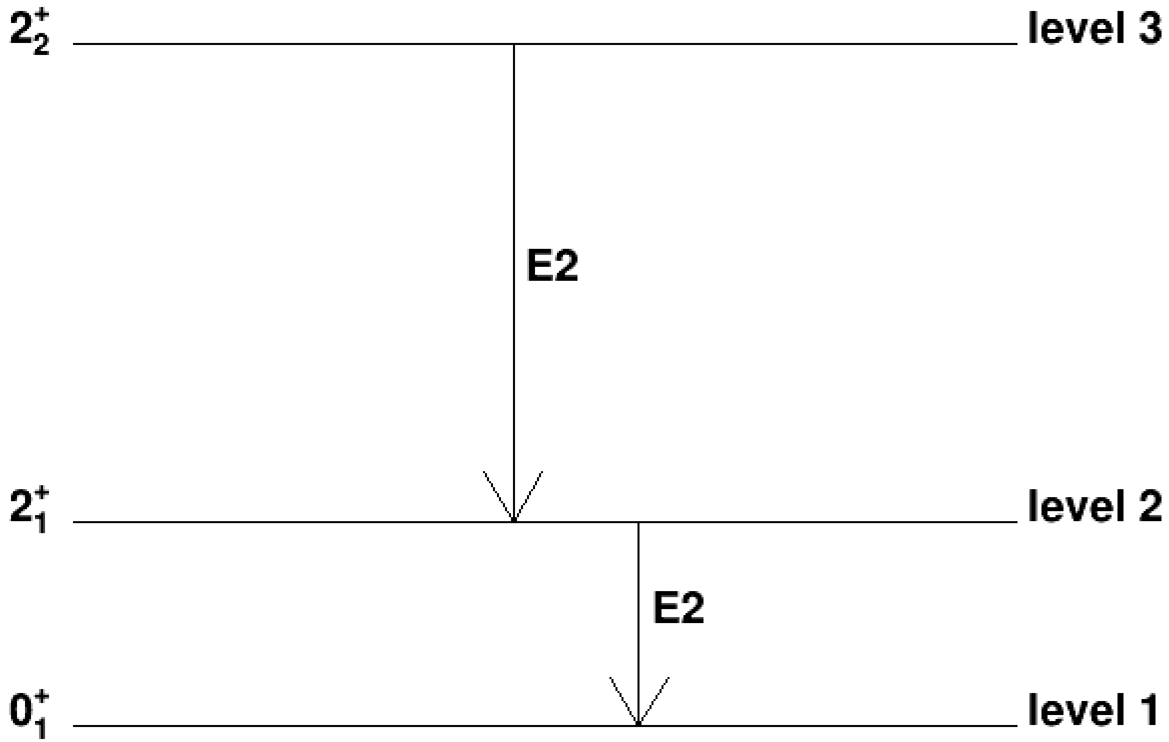
- And again, it corrupts the memory.

Invalid mixing ratio data

- If we turn on bounds checking, we get a runtime error in the OP,YIEL step:

```
At line 11814 of file gosia_20081208.13.f
Fortran runtime error: Index '0' of dimension 1 of array 'elm' below lower bound of 1

Error termination. Backtrace:
#0  0x7f8e4231ef3a in ???
#1  0x7f8e4231fb25 in ???
#2  0x7f8e4231fef7 in ???
#3  0x4026d5 in mixr_
        at gosia_20081208.13.f:11814
#4  0x483db9 in gosia
        at gosia_20081208.13.f:3484
#5  0x484691 in main
        at gosia_20081208.13.f:3542
```

Invalid experimental matrix elements




```
OP,GOSI
...
LEVE
1  1  0  0.00
2  1  2  0.12
3  1  2  0.40
0,0,0,0
ME
2,0,0,0,0
  1  2 0.1      1.0  1.0
  2  3 0.1     -1.0  1.0
0,0,0,0.0
…

OP,YIEL
…
2,1,3,0.10,0.01
```

- What if we provide a value for the E2 matrix element for the $2_2^+ \rightarrow 0_1^+$ state, but we forgot to declare it in the first place?

Invalid experimental matrix elements

- This is another case, where there is no warning and no error in the OP,YIEL step.
- Actually, it gets totally mixed up and looks up a valid matrix element, but not the one you wanted! This will get included in the χ^2 during minimisation. So the fit will probably work, but you have created an incorrect constraint on one of the matrix elements!

CONT options with integers

- If you read the manual carefully (page 121) there's an important comment written in bold:

CONT

.

. Input control parameters needed for the job.

.

END, Marks the end of the CONT input. End must be followed by a comma and a blank line.

The control switches are specified by a three-character alphanumeric string followed by a comma and, in some cases, by a floating point number X (as a convention, X will be used as the initial symbol for floating point entries while symbols beginning with I are specified to be integer). **Even if X has the meaning of an integer it should be entered as a floating point number, the conversion being done by the code.** Some of the switches require additional input. The available control switches are as follows:

- So what does Gosia do if you give a CONT option with an integer by mistake?

CONT options with integers

- e.g. the manual says that DIP sets the E1 dipole polarization parameter to X/1000. The default value is X = 5. So DIPOL is 5×10^{-3}. The manual describes it on page 122.

DIP,X. Sets the $E1$ dipole polarization parameter to X/1000 (See equations 3.25-26). Default value is 5.

- So what does the code here do to DIPOL?

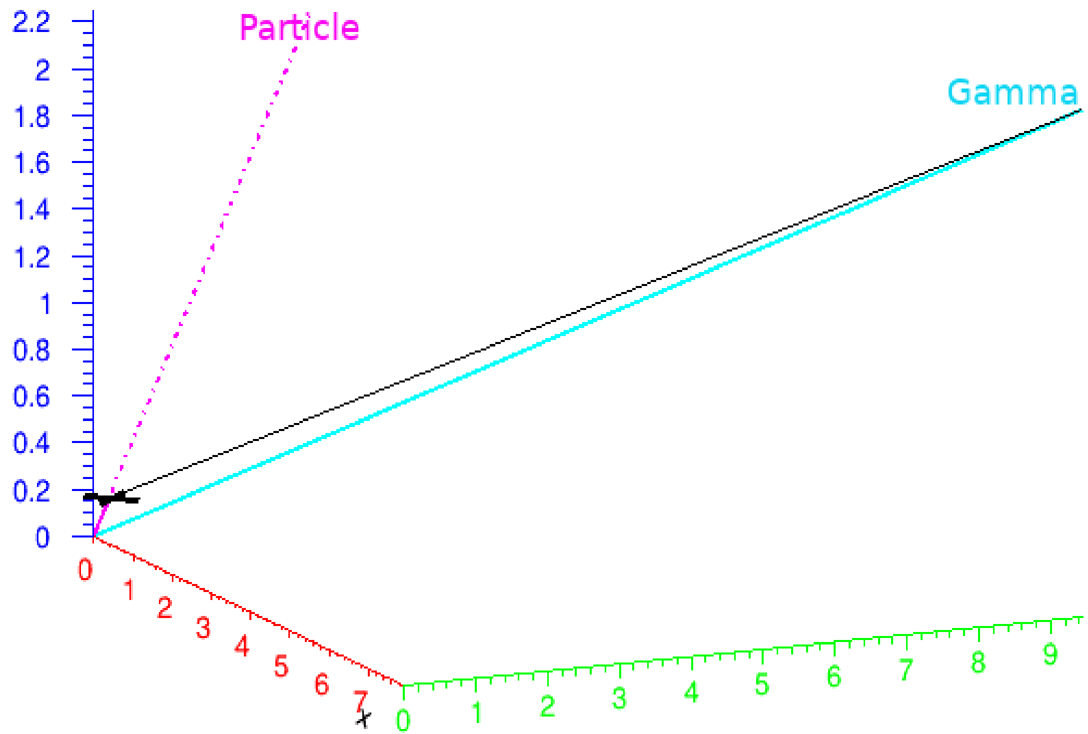
```
OP,GOSI
CONT
DIP,5
END,
```

CONT options with integers

- It sets DIPOL to 5×10^{-4}! Exactly a tenth of what we expect!
- The reason is that the Fortran expects a REAL number and it interprets the 5 as 0.5!
- If you use ACC,5. it sets ACCUR to 10^{-5}, but ACC,5 sets ACCUR to $10^{-0.5} = 0.316$!
- For other options, where the X corresponds to a number of lines to read, X is interpreted as zero and the lines of data are silently ignored!

IFLAG in OP,YIEL

- IFLAG is set in OP,YIEL and it determines whether the effect of the finite distance travelled by the decaying nucleus on the γ-ray angular distribution is to be included.



IFLAG in OP,YIEL

- Page 177 of the manual says:

IFLAG Assumes the values of 0 or 1. $IFLAG = 1$ means that the correction to the angular distribution of the γ-rays due to a finite distance traveled by the decaying nucleus will be included in the calculation (see Section 6.4). $IFLAG = 0$ switches off this correction.

- In the full description two pages later, the manual points out that if you are scanning over matrix elements, you can get the situation, where a matrix element is close to zero and the lifetime becomes unreasonable.

In this case, “unreasonable” means that the code calculates that the nucleus emitting the γ-ray has travelled many tens of centimetres.

IFLAG in OP,YIEL

- The theory behind the correction is explained on page 85 of the manual:

For a given direction of the recoil the observed yield of the decay of a state having the decay constant λ can be written as:

$$Y = \lambda \int_0^\infty \exp(-\lambda t) \cdot S(t)Y(t)dt \tag{6.39}$$

where $S(t)$ denotes the time dependence of the solid angle factor, while $Y(t)$ stands for the time dependence of the "point" angular distribution. To the lowest order, the product $S(t)Y(t)$ is expressed as:

$$S(t)Y(t) \approx Y(0) + pt \tag{6.40}$$

where p stands for the time derivative of this product taken at $t = 0$ (note that $S(0) = 1$, thus $S(0)Y(0) = Y(0)$). Inserting Eq. 6.40 into Eq. 6.39 and using the displacement distance, s, as an independent variable instead of time, we finally obtain for mean lifetime τ:

$$Y = Y(0) + \tau p \tag{6.41}$$

where p is calculated numerically using a second set of yields evaluated in a point shifted by s in the recoil direction, i.e.:

$$p = \frac{S(s)Y(s) - Y(0)}{s} \tag{6.42}$$

where $S(s)$, is calculated assuming that the displacement is small compared to the distance to the detector, r_0:

$$S(s) = \frac{r_0^2}{(\bar{r_0} - \bar{s})^2} \tag{6.43}$$

The displacement correction requires the γ yields to be calculated twice for each evaluation, thus should be requested only when necessary to avoid slowing down the execution. The user is responsible to check that the first-order correction is adequate.

IFLAG in OP,YIEL

- Internally, Gosia calculates for a γ-ray emitted at the origin and one emitted 0.25 cm along the direction of motion of the particle and then applies the correction described in the manual, interpolating between those two points, or extrapolating to the point where the γ-ray is emitted based on the lifetime.
- If, however, a matrix element gets too small and the lifetime gets large, the extrapolation is huge. Depending on the direction, this can either increase or decrease the yield. It can even result in Gosia calculating **negative yields**!

IFLAG in OP,YIEL

- If you are minimizing using Gosia, the code checks if $\tau > 10^4$ ps and if this happens, it gives a message and turns off the correction:

```
DURING THE MINIMIZATION IT WAS NECESSARY TO SWITCH OFF THE
TIME-OF-FLIGHT CORRECTION
```

- Note, however, that the test is against τ not $\tau\beta$, which would probably be more appropriate.
- If you use an external code to do the fit, there is no check!

What should users do?

- Compiling with bounds checking catches bad inputs, which result in memory corruption, but not the case, where Gosia just gets the calculation “wrong” (i.e. it isn't performing the calculation you think it is!). It is, however, **highly recommended** to do bounds checking. With gfortran this is the *-fbounds-check* option. You might want to add *-g* so you get symbols as well as hex dumps, but you have to be an expert to interpret the symbols correctly!

What can Gosia developers do?

- With the February 2018 version:
  - ➔ Error messages for mistakes in the input for yields, branching ratios, lifetimes, mixing ratios and known matrix elements.
  - ➔ CONT accepts the “X” parameter as a REAL or an INTEGER.

What about Rachel?

- Another option for the user is to use Rachel to generate the input. This should ensure that the inputs follow the specifications correctly. It should eliminate inconsistencies such as referencing levels or matrix elements, that don't exist.
- However, there is no manpower working on Rachel.

What about Rachel?

- Rachel is written in python2 (python3 is nearly 10 years old), which will be obsoleted next year.
- It also uses gtk2, which is also rather old (gtk3 is already 7 years old).
- Moreover, there are problems with Rachel and the current version of matplotlib, so it doesn't display the level scheme window properly.

Conclusions

- Users have to be very careful with their inputs. Gosia doesn't check anything!
- We can, however, try to improve this.
- But more manpower is needed to do this…
  - A nice project for someone might be to create a test suite of inputs (good and bad) that can be used to verify that the good inputs produce good outputs and the bad ones produce useful error messages.
  - A more ambitious one, would be to update Rachel.