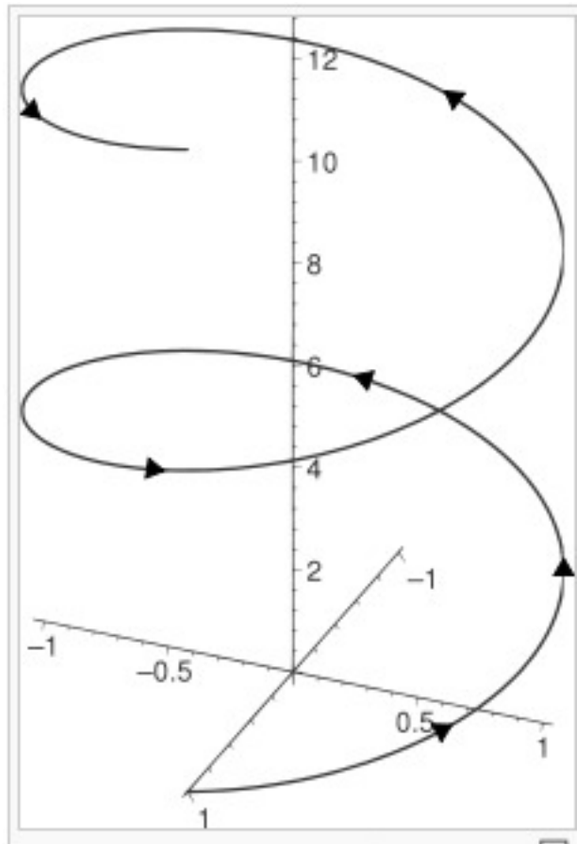
# The Helix Object

E.P.

## Helix

A **helix** (pl: **helixes** or **helices**) is a type of smooth space curve, i.e. a curve in three-dimensional space. It has the property that the tangent line at any point makes a constant angle with a fixed line called the *axis*.

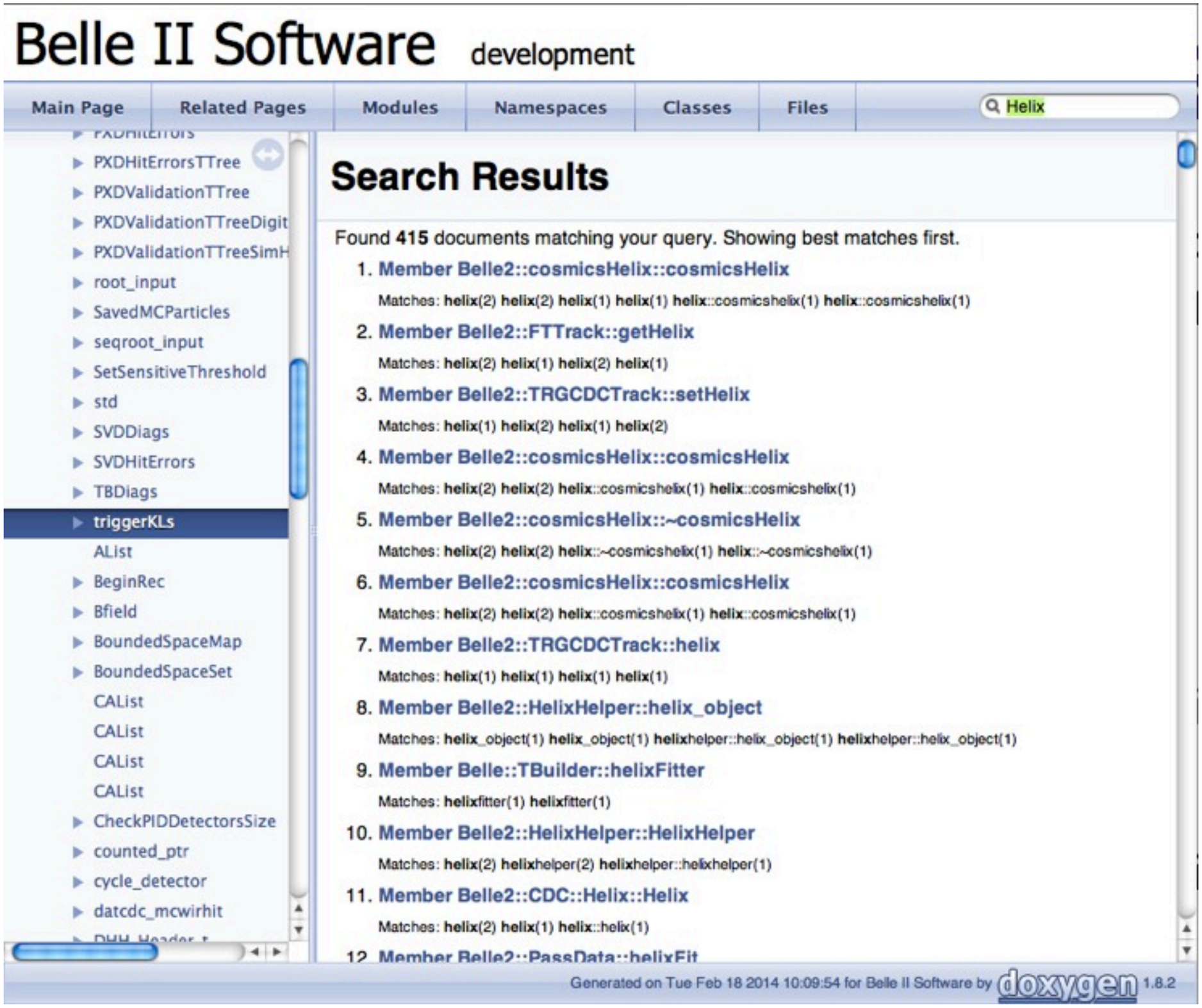



The helix (cos *t*, sin *t*, *t*) from $t = 0$ to $4\pi$ with arrowheads showing direction of increasing *t*

# Why do we need an Helix?

- The helix is the trajectory followed by a charged particle in a uniform magnetic field in vacuum neglecting energy losses
  - It is an excellent approximation for particles moving inside the vacuum chamber of the beam pipe
  - It is a good local approximation of the actual trajectory outside the vacuum chamber
  - As a mater of fact almost everyone implemented it for its own purposes

# Do we really need an additional Helix Object?



# *Yes if users are going to use only the mdst package for their analysis.*

- We need a light weight (i.e. TObject free) helix object that will reside in the mdst package.
  - The analysis EDM object should be able to build it and to be built from it. ( Rationale: You can build as many Helices without paying the TObject penalty burden. Once you are satisfied by your helix you can store it on Root files. )
- It will be very useful if you (tracking developers) will refrain by writing your own helix, by telling what do you need/like from an Helix.
- N.B. The Helix is not going to provide a "fit my hits" methods.

# The Official Belle2 Helix. ©The Tracking Group

- Parametrization: we are going to use the 5 parameters of the mdst package
- What the hel(l)ix have to provide?
  - position and momentum at a given path length (optionally with 6x6 cov matrix )
- Point/momentum of closest approach to a given point.
- Point/momentum of closest approach to a given straight line.
- other requests? (A part a good documentation.)

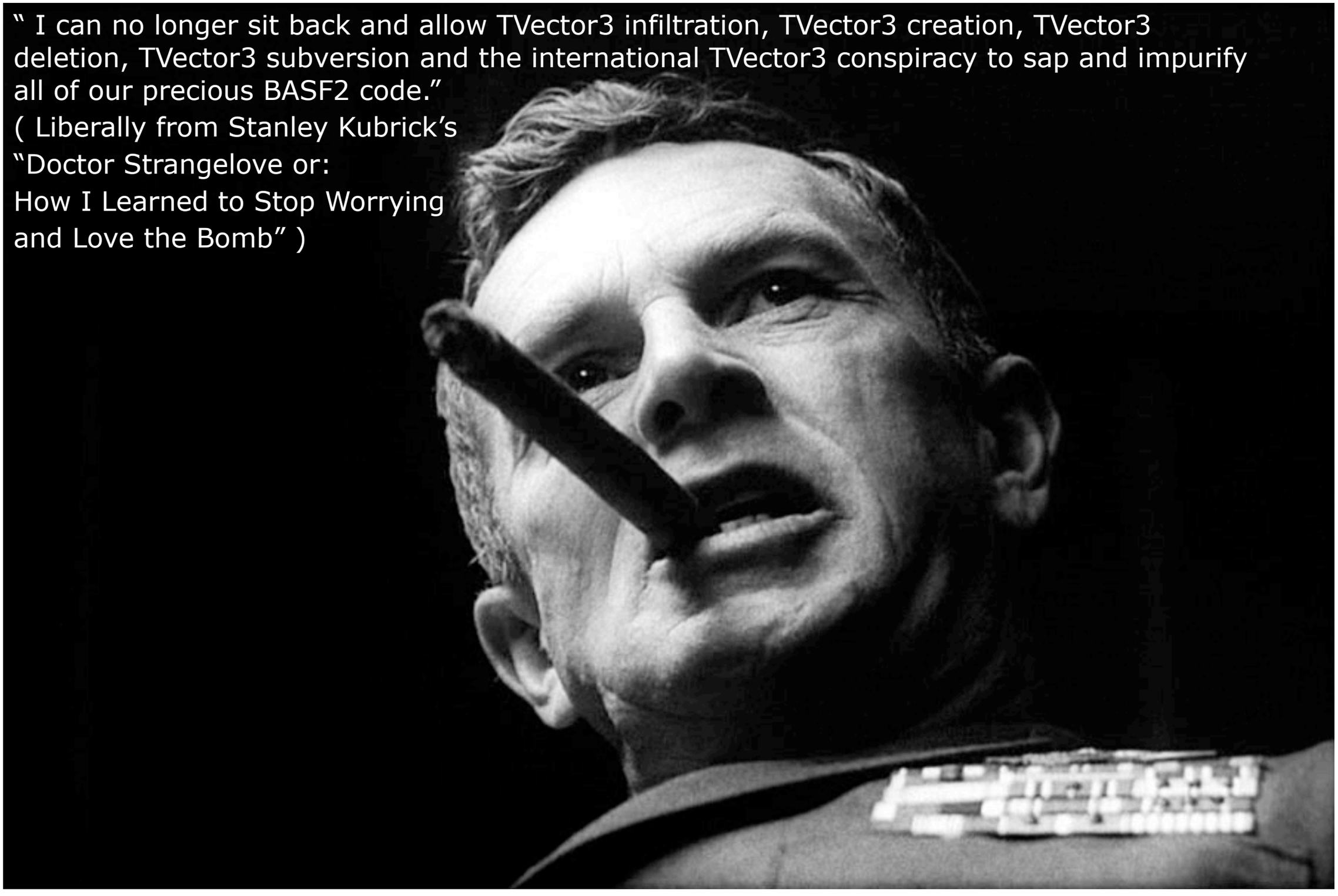
# Action Items

- 1) Collects the users request
  - Jakob, Oliver, Physics group?
- 2) Parametrization: any suggestion from Rudy?
- 3) Can we use an Automatic Differentiation tool to simplify our error propagation task?

# TVector3 is polluting our Code

" I can no longer sit back and allow TVector3 infiltration, TVector3 creation, TVector3 deletion, TVector3 subversion and the international TVector3 conspiracy to sap and impurify all of our precious BASF2 code."

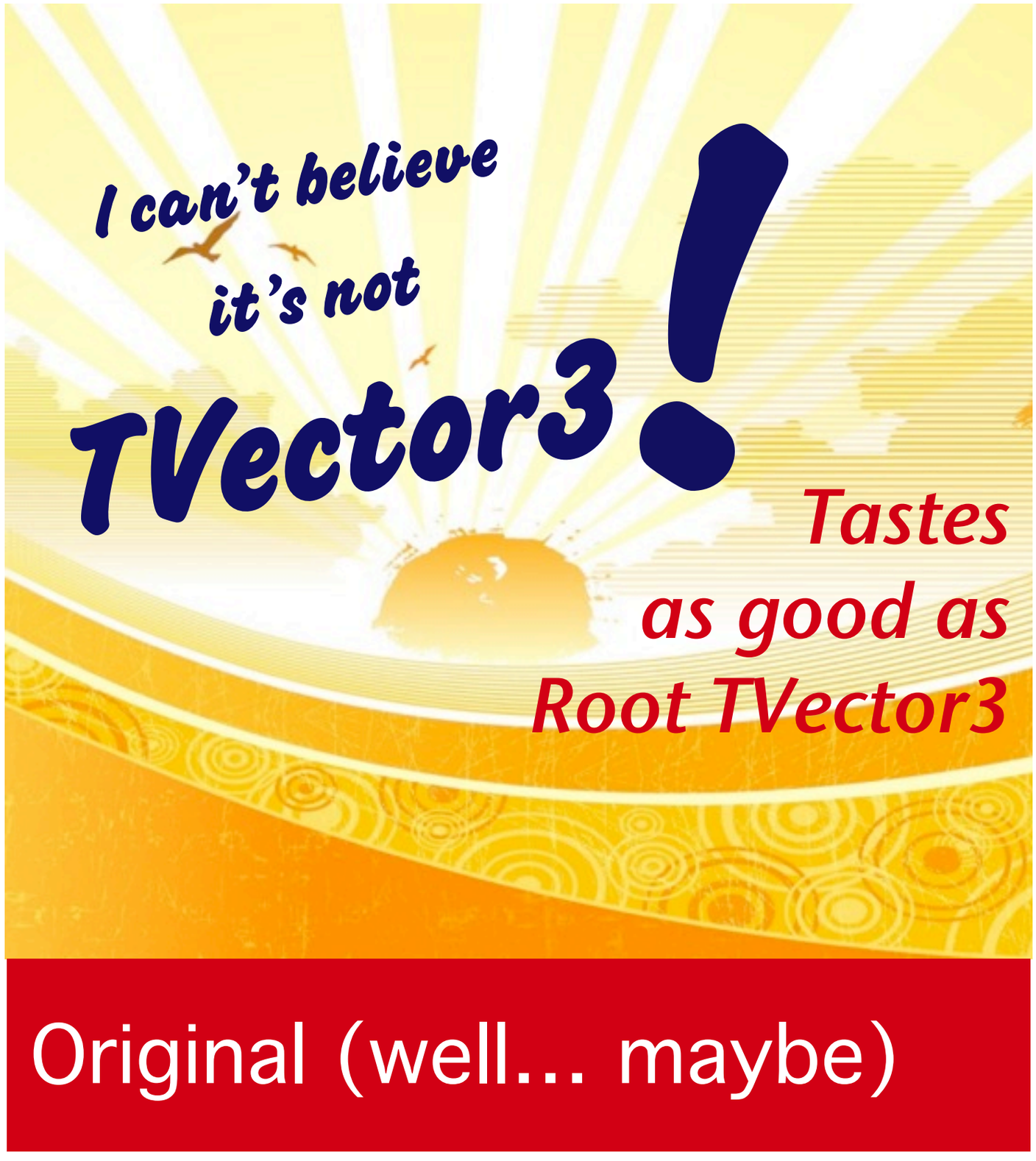
( Liberally from Stanley Kubrick's "Doctor Strangelove or: How I Learned to Stop Worrying and Love the Bomb" )



# Why is TVector3 so evil?

- TVector3 are sooo handy! We all love to write
  a += b
  instead of tediously adding component by component
  a[1] += b[1]
- **BUT** TVector3 inherit from **TObject**, each time we create/delete it we have to create/delete the whole TObject.
  - In many cases you need just a double[3] ( perhaps a float[3] )
  - We need a light weight TVector3 with all the goodies and no fat.


I can't believe
it's not
Butter!
Tastes
as good as
fresh butter
according to 4 out of
butter users
70% Less Saturated
Fat than Butter*
Original


I can't believe
it's not
TVector3!
Tastes
as good as
Root TVector3
Original (well... maybe)

# Simple solution

- Peek the TVector3 class, rename it, get rid of the TObject, inline the getters and setters, provides const accessors then migrate the code.
  For emacs users:
  M-x replace-string
  TVector3
  ICantBelieveItsNotTVector3

- Seriously, let us decide a name
  - BVector3 (B for Basf2)
  - tVector3 (t for tracking)
  - And NO, I do not want to use Belle2::TVector3 just in case you were thinking about it (As I did for quite a while).

# Action Item

- Let us decide a name
- Who