Neutrino oscillation probability from a Jpp perspective

M. de Jong

Neutrino oscillation probability

$$P(\nu_1 \to \nu_2) \equiv f(\sin^2\theta_{12}, \sin^2\theta_{23}, \sin^2\theta_{13}, \delta m_{21}, \delta m_{31}, \delta_{CP}; E_\nu, \cos\theta)$$

Is it possible to
tabulate $f(\sin^2\theta_{12}, \sin^2\theta_{23}, \sin^2\theta_{13}, \delta m_{21}, \delta m_{31}, \delta_{CP})$ and
interpolate *a posteriori* between $(E_\nu, \cos\theta)$?

Jpp

```
typedef JMAPLIST< JMap,
                  JMap,
                  JMap,
                  JMap,
                  JMap,
                  JMap,
                  JMap,
                  JMap>::maplist  JMaplist_t;

JMultiMap<double, double, JMaplist_t>  zmap;
```

```
for (E_ν) {
  for (cos θ) {
    for (sin²θ_12) {
      for (sin²θ_23) {
        for (sin²θ_13) {
          for (δm_21) {
            for (δm_31) {
              for (δ_CP) {
                zmap[sin²θ_12][sin²θ_23][sin²θ_13][δm_21][δm_31][δ_CP][E_ν][cos θ] =
                    f(sin²θ_12, sin²θ_23, sin²θ_13, δm_21, δm_31, δ_CP; E_ν, cos θ);
              }
            }
          }
        }
      }
    }
  }
}
```

2D interpolator type

```
typedef   JPolint0Function1D_t                                JFunction1D_t;
typedef   JFunction1D_t::abscissa_type                        abscissa_type;
typedef   JFunction1D_t::value_type                           value_type;

typedef   JMAPLIST<JPolint0FunctionalGridMap>::maplist        JMaplist1D_t;
typedef   JMultiFunction<JFunction1D_t, JMaplist1D_t>         JFunction2D_t;
```

6D interpolator type – 2D return type

```
typedef   JMap<abscissa_type, JCollection<value_type> >      JMap2D_t;
typedef   JConstantFunction1D<abscissa_type, JMap2D_t>      JCollection2D_t;

typedef   JMAPLIST<JPolint1FunctionalGridMap,
                   JPolint1FunctionalGridMap,
                   JPolint1FunctionalGridMap,
                   JPolint1FunctionalGridMap,
                   JPolint1FunctionalGridMap,
                   JPolint1FunctionalGridMap>::maplist      JMaplist6D_t;

typedef   JMultiFunction<JCollection2D_t, JMaplist6D_t>      JFunction6D_t;
```

6D interpolator returning 2D interpolator

```
JFunction6D_t  g6;        // 6D-function returning 2D-collection
JFunction2D_t  g2;        // 2D-function returning single value

copy(g6(sin^2θ_12, sin^2θ_23, sin^2θ_13, δm_21, δm_31, δ_CP), g2);
```

```
for (E_ν) {
  for (cos θ) {
    g2(E_ν, cos θ);       // evaluate P for the two observables
  }
}
```

Get oscillation probability... OK

1983.323 ms elapsed
1975.319 ms user
0.990 ms system

Get 2D-function... OK

1.105 ms elapsed
1.067 ms user
0.032 ms system

Get interpolation... OK

58.681 ms elapsed
58.471 ms user
0.000 ms system