JPolynome5D.cc

The JPolynome5D.cc program can be used to test interpolation in a 5-dimensional space.

Definition of a 5-dimensional function:

```
/**
 * 5D function.
 */
inline double f5(const double x0,
                 const double x1,
                 const double x2,
                 const double x3,
                 const double x4)
{
  static const double a = 1.0;
  static const double b = 1.0;
  static const double c = 1.0;

  return
    x0 * (a + x0 * (b + c*x0))  *
    x1 * (a + x1 * (b + c*x1))  *
    x2 * (a + x2 * (b + c*x2))  *
    x3 * (a + x3 * (b + c*x3))  *
    x4 * (a + x4 * (b + c*x4));
}
```

Definition of a 5-dimensional interpolator with 3-degree polynomial interpolation in each dimension:

```
  typedef JGridPolint3Function1D_t                          JFunction1D_t;

  typedef JMAPLIST<JPolint3FunctionalGridMap,
                   JPolint3FunctionalGridMap,
                   JPolint3FunctionalGridMap,
                   JPolint3FunctionalGridMap>::maplist      JMaplist_t;

  typedef JMultiFunction<JFunction1D_t, JMap_t>             JMultiFunction_t;

  JMultiFunction_t g5;
```

Construction of 5-dimensional interpolator.

```
const int    N    = 11;
const double xmin = -1.0;
const double xmax = +1.0;

for (double x0 = xmin; x0 <= xmax; x0 += (xmax - xmin) / (N - 1)) {
  for (double x1 = xmin; x1 <= xmax; x1 += (xmax - xmin) / (N - 1)) {
    for (double x2 = xmin; x2 <= xmax; x2 += (xmax - xmin) / (N - 1)) {
      for (double x3 = xmin; x3 <= xmax; x3 += (xmax - xmin) / (N - 1)) {
        for (double x4 = xmin; x4 <= xmax; x4 += (xmax - xmin) / (N - 1)) {
          g5[x0][x1][x2][x3][x4] = f5(x0, x1, x2, x3, x4);
        }
      }
    }
  }
}

g5.compile();
```

Test of 5-dimensional interpolator:

```
  JQuantile Q;

  for (int i = 0; i != numberOfEvents; ++i) {

    const double x0  = gRandom->Uniform(xmin, xmax);
    const double x1  = gRandom->Uniform(xmin, xmax);
    const double x2  = gRandom->Uniform(xmin, xmax);
    const double x3  = gRandom->Uniform(xmin, xmax);
    const double x4  = gRandom->Uniform(xmin, xmax);

    Q.put(f5(x0,x1,x2,x3,x4) - g5(x0,x1,x2,x3,x4));
  }

  cout << "mean  " << SCIENTIFIC(10,2) << Q.getMean()  << endl;
  cout << "RMS   " << SCIENTIFIC(10,2) << Q.getSTDev() << endl;
```

Typical output is:

```
mean   -1.20e-17
RMS     1.66e-17
```