The packages in MATHEMAGIX

21 November 2007

The structure of a package

- Autonomous software component.
- Installation `./configure && make`; with options:

```
--enable-glue        compile glue for interpreter [yes]
--enable-debug       install a debugging enable executable [-ggdb]
--enable-optimize    compile with optimizations [guessed]
--enable-verify      verify correctness of low-level operations [no]
--enable-test        compile additional test programs [no]
--enable-bench       compile additional benchmark programs [no]
```

- Usually, it provides
  - a library `libpackage.so` of the exported functions,
  - a library `libmmxpackage.so` for the interaction with the interpreter based on `libpackage.so`.
- Same general structure (to simplify the maintenance).

```
|-- LICENSE
|-- Makefile
|-- ...
|-- build
|    |-- Makefile
|    `-- ...
|-- doc
|    |-- Makefile
|    |-- doxyfile
|    |-- basix.tm
|    |-- ...
|    `-- html
|    `-- ...
|-- glue
|    |-- basix_glue.cpp
|    `-- ...
|-- include
|    `-- basix
|    `-- ...
|-- macros
|    |-- ...
|    `-- mmx_module.m4
|-- script
|    |-- basix-config
|    `-- basix-config.in
|-- src
|    `-- ...
`-- test
`-- ...
```

- Place where the compilation is run
- The documentation part (TEXMACS and DOXYGEN files)
- The connection to the interpreter
- The header files
- The macros which specify the behavior of `configure`
- The files to be compiled
- The test files

The existing packages

- `basix:` strings, lists, vectors, tables, symbols, generics, ...

- `numerix:` extended arithmetic based on GMP `integer`, `rational`, MPFR `floating` with rounding modes.

- `algebrix/algebramix:` vectors & matrices, dense univariate polynomials with fast arithmetic,skew polynomials, univariate series, multivariate polynomials, dual/inverse systems, Sturm sequence, univariate resultant, ...

- `symbolix:` arithmetic trees for symbolic manipulations.

❒ `subdivix:` Bernstein basis representation of univariate and multivariate polynomials; subdivision solvers with sleeve approximations in 1D, ND; continued fraction expansion of roots of univariate polynomials.

❒ `realroot:` manipulation of real roots of univariate polynomials, sign evaluation, static Sturm sequence for small degree (E. Tsigaridas).

❒ `mps:` Approximation of complex roots of univariate polynomials based on Aberth method (G. Fiorentino & D. Bini).

❒ `nla:` numerical linear algebra, connections to LAPACK: LU, QR, SVD, eigenvalues and eigenvectors.

❒ `analyzit:` basic algebraic structures with error bounds (ball).

❒ `shape:` curves and surfaces of different type: parameterised, implicit, polygonal with algorithms to compute topology, intersection, self-intersection, ...

❒ `pack:` package to install and use packages (R. Soum).

The package PACK

- Help to install precompiled or source versions of packages

```
> search "algebrix"
> install "algebrix"
> remove "algebrix"
```

- Take into account dependancies.

- Use external servers (defined in `sources.mmx`) with a joomla interface.

- Upgrade `index.mmx` to get information on how to install packages (version, architecture, dependancies, url, ...)

How to write your own module MYMOD

- use "mymod" evaluates the function
  - void(*define_mymod) (void) ...;
- Export (parameterised) **types**:
  - define_type<always,cpp_type>("mmx_type")
  - define_type<always,cpp_ptype<C>>(gen("A_type",NAME(C)))
- Export **functions**:
  - define<always>("eval",my_cpp_eval_function);
- Export **operators**:
  - define_binary<always,cpp_type,add_op>();
- Export implicit **converters**:
  - define_caster<Cond,cpp_type1,cpp_type2>(...);
- Export general **signatures** under **conditions**:
  - lift_scalar_field_ops<IsField,polynomial<C>,C> ();