Basic Data Type for KAGALI PE

By KGWG DAS team

Hyung Won Lee, Chunglee Kim, Yeong-Bok Bae, Jeongcho Kim

Date 20 Nov. 2017

Overall Idea

- Use a single structure pointer passing to waveform function
- Use a union data type containing all types of necessary data types for waveform calculation
- Parameter estimation algorithm implementations and overall management : Hyung Won Lee and Chunglee Kim
- Waveform implementation : Jeongcho Kim
- Likelihood function implementation : Yeong-Bok Bae
- All other necessary functions : all of us based on the necessity

KGLWaveformModelParamsType

- Defined in KGLWaveformModelParams.h.in
- Role
  - Enumerator for all possible data types
- Currently defined data types
  - char, int, int64, uint, uint64, float, double, double complex, string(char *), void pointer

KGLWaveformModelParamsType

```
typedef enum KGLWaveformModelParamsType_tag {
  KGLWAVEFORM_MODEL_PARAMS_char_t,
  KGLWAVEFORM_MODEL_PARAMS_int_t,
  KGLWAVEFORM_MODEL_PARAMS_int64_t,
  KGLWAVEFORM_MODEL_PARAMS_uint_t,
  KGLWAVEFORM_MODEL_PARAMS_uint64_t,
  KGLWAVEFORM_MODEL_PARAMS_float_t,
  KGLWAVEFORM_MODEL_PARAMS_double_t,
  KGLWAVEFORM_MODEL_PARAMS_double_complex_t,
  KGLWAVEFORM_MODEL_PARAMS_string_t,
  KGLWAVEFORM_MODEL_PARAMS_void_ptr_t,
  KGLWAVEFORM_MODEL_PARAMS_LAST_t
} KGLWaveformModelParamsType;
```

KGLWaveformModelParamsValue

- Defined in KGLWaveformModelParams.h.in
- Role
  - Unified a single data type for waveform model parameter values
- Currently defined field values
  - char_value, string_value, int64_value, uint_value, uint64_value, float_value, double_value, double_complex_value, void_ptr_value

KGLWaveformModelParamsValue

```
typedef union KGLWaveformModelParamsValue_tag {
  char char_value;
  char * string_value;
  int int_value;
  int64_t int64_value;
  uint uint_value;
  uint64_t uint_64_value;
  float float_value;
  double double_value;
  double complex double_complex_value;
  void * void_ptr_value;
} KGLWaveformModelParamsValue;
```

KGLWaveformModelParamsIndex

- Defined in KGLWaveformModelParams.h.in
- Role
  - Index for physical parameters in parameter array
- Currently defined field values

KGLWaveformModelParamsIndex

```
typedef enum KGLWaveformModelParamsIndex_tag {
  KGLWAVEFORM_MODEL_PARAMS_distance,
  KGLWAVEFORM_MODEL_PARAMS_iota,
  KGLWAVEFORM_MODEL_PARAMS_psi,
  KGLWAVEFORM_MODEL_PARAMS_ra,
  KGLWAVEFORM_MODEL_PARAMS_dec,
  KGLWAVEFORM_MODEL_PARAMS_m1,
  KGLWAVEFORM_MODEL_PARAMS_m2,
  KGLWAVEFORM_MODEL_PARAMS_spin1x,
  KGLWAVEFORM_MODEL_PARAMS_spin1y,
  KGLWAVEFORM_MODEL_PARAMS_spin1z,
  KGLWAVEFORM_MODEL_PARAMS_spin2x,
  KGLWAVEFORM_MODEL_PARAMS_spin2y,
  KGLWAVEFORM_MODEL_PARAMS_spin2z,
  KGLWAVEFORM_MODEL_PARAMS_eccentricity,
  KGLWAVEFORM_MODEL_PARAMS_tidal_lambda1,
  KGLWAVEFORM_MODEL_PARAMS_tidal_lambda2,
  KGLWAVEFORM_MODEL_PARAMS_phaseOrder,
  KGLWAVEFORM_MODEL_PARAMS_ampOrder,
  KGLWAVEFORM_MODEL_PARAMS_eccOrder,
  KGLWAVEFORM_MODEL_PARAMS_spinOrder,
  KGLWAVEFORM_MODEL_PARAMS_tidalOrder,
  KGLWAVEFORM_MODEL_PARAMS_coal_phase,
  KGLWAVEFORM_MODEL_PARAMS_f_ecc,
  KGLWAVEFORM_MODEL_PARAMS_f_ref,
  KGLWAVEFORM_MODEL_PARAMS_phi_ref,
  KGLWAVEFORM_MODEL_PARAMS_f_min,
  KGLWAVEFORM_MODEL_PARAMS_f_max,
  KGLWAVEFORM_MODEL_PARAMS_domain,
  KGLWAVEFORM_MODEL_PARAMS_approximant,
  KGLWAVEFORM_MODEL_PARAMS_NUMBER
} KGLWaveformModelParamsIndex;
```

KGLWaveformModelParamsItem

- Defined in KGLWaveformModelParams.h.in
- Role
  - Actual parameter value and name
- Currently defined field values

```
typedef struct KGLWaveformModelParamsItem_tag {
  char name[VAR_NAME_MAX];
  KGLWaveformModelParamsValue value;
  KGLWaveformModelParamsType type;
} KGLWaveformModelParamsItem;
```

KGLWaveformModelParams

- Defined in KGLWaveformModelParams.h.in
- Role
  - Array of all parameters
- Currently defined field values

```
typedef struct KGLWaveformModelParams_tag {
  KGLWaveformModelParamsItem variables[KGLWAVEFORM_MODEL_PARAMS_NUMBER];
  int dimension;
} KGLWaveformModelParams;
```

Waveform function pointers

- KGLFDWaveform

```
typedef void (*KGLFDWaveform)(KGLStatus *status, KGLWaveformModelParams *params, double complex *hp,
double complex *hc, double f);
```

- KGLFDWaveformArray

```
typedef void (*KGLFDWaveformArray)(KGLStatus *status, KGLWaveformModelParams *params, double complex *hp,
 double complex *hc, double *f, int n_start, int n_end, int N, KGLFDWaveform func);
```

- Defined in KGLWaveforms.h.in

KGLInspiralFDTaylorF2

- KGLInspiralFDTaylorF2.c

```
/**
 * KGLInspiralFDTaylorF2
 *
 *  default waveform function prototype, supplied only for example
 */
void KGLInspiralFDTaylorF2( //begin{proto}
    KGLStatus *status,    /**< [in, out] kgl status pointer */
    KGLWaveformModelParams * params, /**< [in] waveform model parameters to print */
    double complex *hp, /**< [out] calculated h_plus value */
    double complex *hc, /**< [out] calculated h_cross value */
    double f /**< [in] the given frequency value  */
    ) //end{proto}
{
  *hp = cos(f) + sin(f)*I;
  *hc = cos(f) - sin(f)*I;
}
```

What to do

- generateTemplate
  - Make a simple application to generate data for the given waveform
  - Possibly useful for search pipeline
  - Generate temple banks
- inferenceMCMC
  - Parallel tempering MCMC parameter estimation for CBC signal
  - Minimum implementation of lalsuite
  - Include only one wave for inspiral TaylorF2 including extended para meters, eccentricity, tidal, etc.

Discussions

- how to utilize data read lib and a waveform lib from inference lib