CARDIFF

UNIVERSITY

INSPIRAL

CODES.

ASIS

- Theoretical calculations have given us post–Newtonian expansions of an energy function $E(x \equiv v^2)$ and a GW luminosity function $\mathcal{F}(v)$, together which define the gravitational wave phasing formula:

$$\begin{aligned} t(v) &= t_{\rm ref} + m \int_v^{v_{\rm ref}} \frac{E'(v)}{\mathcal{F}(v)}\, dv, \\ \phi(v) &= \phi_{\rm ref} + 2 \int_v^{v_{\rm ref}} v^3 \frac{E'(v)}{\mathcal{F}(v)}\, dv, \end{aligned} \tag{1}$$

$v = (\pi m F)^{1/3}$,
F is the instantaneous GW frequency,
m is the total mass of the binary.

- T–approximants are Taylor approxiants of the energy and flux functions,

$$\dot{E}_{T_n}(x) \equiv E_N(x) \sum_{k=0}^{n} \hat{E}_k(\eta) x^k \tag{2}$$

$$\mathcal{F}_{T_n}(x) \equiv \mathcal{F}_N(x) \left[\sum_{k=0}^{n} \hat{\mathcal{F}}_k(\eta) v^k + \sum_{k=6}^{n} \hat{L}_k(\eta) \log(v/v_0) v^k \right], \tag{3}$$

where

$$E_N(x) = -\frac{1}{2}\eta x; \quad \mathcal{F}_N(x) = \frac{32}{5}\eta^2 x^5. \tag{4}$$

- There are two possible approaches now:

  1. Re–expand $E'_{T_n}(v)/\mathcal{F}_{T_n}(v)$ appearing in the phasing formula, in which case the integrals in Eq.(1) can be done analytically to obtain a *parametric* representation of the phasing formula in terms of polynomial expressions in the auxiliary variable v

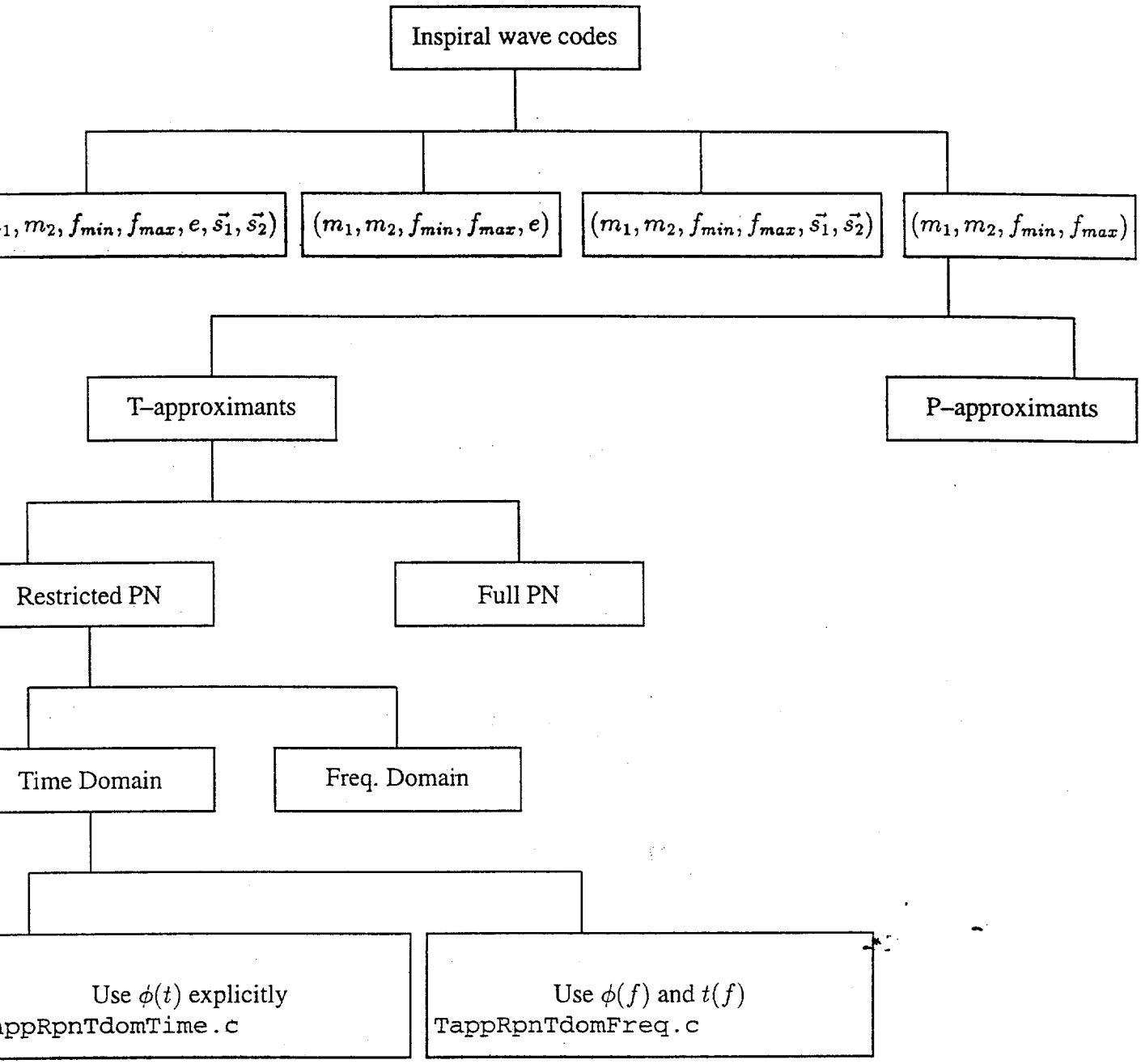
$$\begin{aligned} \phi_{T_n}(v) &= \phi_{\rm ref} + \phi^v_N(v) \sum_{k=0}^{n} \hat{\phi}^v_k v^k, \\ t_{T_n}(v) &= t_{\rm ref} + t^v_N(v) \sum_{k=0}^{n} \hat{t}^v_k v^k, \end{aligned} \tag{5}$$

  2. the second of the polynomials in Eq. (5) can be inverted and the resulting polynomial for v in terms of t can be substituted in $\phi^{(2)}(v)$ to arrive at an explicit time-domain phasing formula

$$\phi_{T_n}(t) = \phi_{\rm ref} + \phi^t_N \sum_{k=0}^{n} \hat{\phi}^t_k \theta^k \tag{6}$$

$$F_{T_n}(t) = F^t_N \sum_{k=0}^{n} \hat{F}^t_k \theta^k, \tag{7}$$

  where
  $\theta = [\eta(t_{\rm ref} - t)/(5m)]^{-1/8}$,
  $F \equiv d\phi/2\pi dt = v^3/(\pi m)$ is the instantaneous GW frequency.


Inspiral wave codes
$_1, m_2, f_{min}, f_{max}, e, \vec{s_1}, \vec{s_2})$
$(m_1, m_2, f_{min}, f_{max}, e)$
$(m_1, m_2, f_{min}, f_{max}, \vec{s_1}, \vec{s_2})$
$(m_1, m_2, f_{min}, f_{max})$
T–approximants
P–approximants
Restricted PN
Full PN
Time Domain
Freq. Domain
Use $\phi(t)$ explicitly
ppRpnTdomTime.c
Use $\phi(f)$ and $t(f)$
TappRpnTdomFreq.c

MakeTemplate.c

The function header is of the form:

```
void MakeTemplate (Status *status,
                   REAL8Vector *waveform,
                   InspiralTemplate *params)
```

The output structure has the form

```
typedef struct tagREAL8Vector{
               INT4 length;
               REAL8 *data;
               }REAL8Vector;
```

The input structure is of the form

```
typedef struct tagInspiralTemplate{
                INT4 number;
                REAL4 m1;
                REAL4 m2;
                REAL4 spin1[3];
                REAL4 spin2[3];
                REAL4 inclination;
                REAL4 eccentricity;
                REAL4 totalMass;
                REAL4 mu;
                REAL4 eta;
                REAL4 fLower;
                REAL4 fCutoff;
                REAL4 tSampling;
                REAL4 phaseShift;
                REAL4 nStartPad;
                REAL4 nEndPad;
                InputMasses massChoice;
                InspiralMethod method;
                } InspiralTemplate;
```

The parameter `MassChoice` is of type `enum InputMasses`, which determines which pair of input masses the user has defined. This typedef is as follows:

```
typedef enum {
              m1Andm2,
              totalMassAndEta,
              totalMassAndMu
              } InputMasses;
```

The parameter `method` is of type `enum InspiralMethod`, which tells the function which code is to be used to generate the waveform. This typedef is (at the moment) as follows:

```
typedef enum {
              Best,
              TappRpnTdomFreq20,
              TappRpnTdomTime20,
              } InspiralMethod;
```

More choices will be added as more codes are written.

Operating Instructions

```
/* Declare the structures to be used  */

InspiralTemplate templateIn;
REAL8Vector *waveform;
Status status;
waveform=(REAL8Vector *)LALMalloc(sizeof(REAL8Vector));

/* Initialize the inputs  */

templateIn.number = 1;
templateIn.mass1 = 10.0;
templateIn.mass2 = 10.0;
templateIn.fLower = 40.0;
templateIn.fCutoff = 1000.0;
templateIn.tSampling = 1.0/4000.0;
templateIn.phaseShift = 0.0;
templateIn.nStartPad = 100;
templateIn.nEndPad = 0;
templateIn.massChoice = m1Andm2
templateIn.method = TappRpnTdomFreq20

/* Call the function  */

MakeTemplate (&status, waveform, &templateIn);
```

Now the waveform will be pointed to by the pointer `waveform->data`.

Error checks

```
void MakeTemplate (Status *status,
                   REAL8Vector *waveform,
                   InspiralTemplate *params)

   ASSERT (waveform!=NULL,
           status,
           MAKETEMPLATE_ENULL,
           MAKETEMPLATE_MSGENULL1);
```

This above example checks whether the pointer `waveform` is a NULL pointer or not. If it is a NULL pointer, then an error message which is defined by the character string `MAKETEMPLATEC_MSGENULL1` is sent to the screen.

Note 1, Linda Turner, 05/09/00 09:00:53 AM
LIGO-G000048-00-D