FFT usando FFTW3

El encabezado del programa debe incluir (en máquinas de 64bits):

```
use, intrinsic         ::  iso_c_binding

include "/usr/include/fftw3.f03"          ! verficar que exista
type(C_ptr)       ::  plan_rc,plan_cr    ! nombre de los planes  a utilizar
```

para transformar un arreglo real (dando como resultado un complejo) creo el plan usando la subrutina (el plan se crea una sola vez, y es mejor hacerlo antes de dar valores al arreglo in)

```
plan_rc = fftw_plan_dft_r2c_1d(N , in , out  , FFTW_MEASURE)
```

y luego ejecuto la transfomación llamando a

```
call fftw_execute_dft_r2c(plan_rc, in , out)
```

Notar que in y out son los arreglos de entrada y de salidad respectivamente y tendrán las siguientes características:

```
real(C_double)              ::   in(N)
complex(C_double_complex)   ::   out(N/2 + 1)
```

donde N es el tamaño del arreglo. Observar que la transformada tiene tamaño N/2+1, pues la transfomada discreta de Fourier, *tf(1:N),* de un arreglo real, *in(1:N)*, satisface

$$tf(N-j) \equiv tf(-j) = tf(j)^*$$

donde *, significa complejo conjugado. Por lo tanto, con sólo guardar N/2+1 datos (el +1 es por el 0), puedo reconstuir todo el arreglo de la transformada de Fourier. En el caso de FFTW, para llevar el resultado a una salida que vaya de -N/2/rango a N/2/rango, podría hacer lo siguiente:

```
factor = 1._pr/rango
do i = -N/2, N/2
  if ( i < 0 ) then
    write(*,*) factor*real(i,pr),conjg( out(-i+1) )/dble(N)
  else
    write(*,*) factor*real(i,pr),out(i+1) /dble(N)
  endif
enddo
```

Notar que se incluyen los factores para que den bien las frecuencias y para la normalización de la transformada.

Finalmente, si no se van a hacer más transformadas, se puede destruir el plan:

```
call fftw_destroy_plan(plan_rc)
```