Software para corregir la deriva del máser ajustando su cavidad

D. Cordobés, P. de Vicente, R. Bolaño

Informe Técnico IT - OAN 2007 - 04

CONTENIDO

I. Introducción

Los informes IT/OAN-CAY 2002-2, 2001-8, 2005-5 y 2005-8 describen cómo ajustar la cavidad del máser de Hidrógeno para corregir su deriva en frecuencia. Para ello, se compara la señal de un pulso por segundo (1PPS) generado por un receptor GPS con la señal de 1PPS del máser (generado por el *Station Timing Module*). Si la frecuencia del máser difiere de la del GPS una cantidad invariable, la diferencia entre ambos pulsos es linealmente dependiente con el tiempo y origina una recta en un diagrama en el que el eje de abscisas es el tiempo. La recta es de pendiente positiva si el máser oscila más lento que el GPS y negativa si el máser oscila más rápido que el GPS.

La pendiente de la recta es la diferencia relativa en frecuencia entre el GPS y el máser. Si admitimos que la frecuencia correcta es la del GPS, esta diferencia indica el error relativo en frecuencia del máser, que se corrige ajustando el varactor de la cavidad. El ajuste se hace por medio de un conversor analógico digital, codificado en octal, en el que un valor de 0000 corresponde a un nivel de tensión de 1V y el valor 7777 corresponde a 3V.

Antes de los trabajos realizados en este informe, la pendiente de la recta y el nuevo valor del DAC se estimaban manualmente y de forma aproximada. Este informe se presenta un programa en Python que automatiza el proceso de cálculo del DAC.

II. Funcionamiento del programa

Toda la información referente a la monitorización del máser se almacena en la base de datos *maserdb* que está formada por dos tablas *maserdata* y *maserstatus.* El campo *maser_dps* de la primera tabla ofrece información sobre la diferencia temporal (retardo) entre la señal de 1pps del receptor GPS y la señal de 1pps del máser.

La **Figura 1** representa los valores de retardo comprendidos entre el 31-1-2006 y el 31-1-2007 (la fecha de redacción de este informe). Esta figura se ha generado con *Plotmaser,* un programa en Python desarrollado en el OAN (informe IT/OAN-CAY 2005-5) que permite representar gráficamente la información contenida en la base de datos *maserdb*. El traslado de los equipos de VLBI al radiotelescopio de 40m (informe IT/OAN-CAY 2007-1) se realizó entre octubre de 2006 y diciembre de 2006 y hubo periodos en los que no se pudo monitorizar el máser, por lo que hay zonas donde no hay datos. Los saltos que se observan se deben a reinicios de cuenta en el contador que calcula la diferencia entre el 1PPS del máser y el 1PPS del receptor GPS.

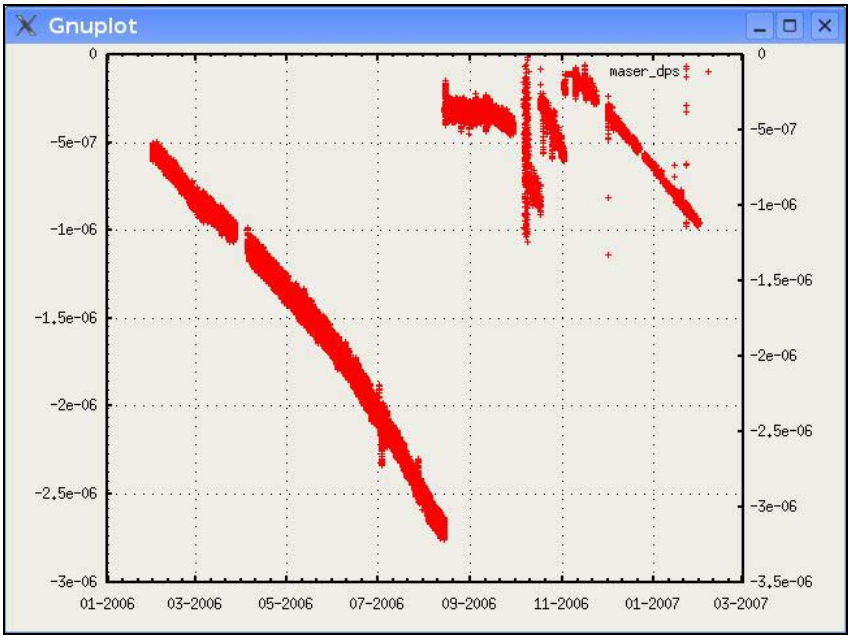

Figura 1. Datos de retardo del máser desde el 31/01/2006 hasta el 30/01/2007

A continuación se realiza un ejemplo que ilustra cómo manejar el programa. Se va a estimar la corrección (a la fecha de redacción de este informe: 31-1-2007) que habría que hacer en el valor del DAC de la cavidad del máser.

1) En primer lugar se identifica el tramo de fechas idóneo. Para ello se emplea el programa *PlotMaser* para estudiar la evolución del retardo del máser en un año: desde el 30-1-2006 hasta el 31-1-2007 (ver Figura 1). A continuación se selecciona el intervalo de fechas (fecha inicial y final) del tramo recto que esté más próximo a la fecha actual: en nuestro caso, a la vista de la Figura 1, seleccionaríamos el tramo comprendido entre las fechas del 1-1-2007 y 31-1-2007 (**Figura 2**). Estas fechas son las que se introducen en los cajetines "*Fecha inicial*" y "*Fecha final*" de la interfaz gráfica.

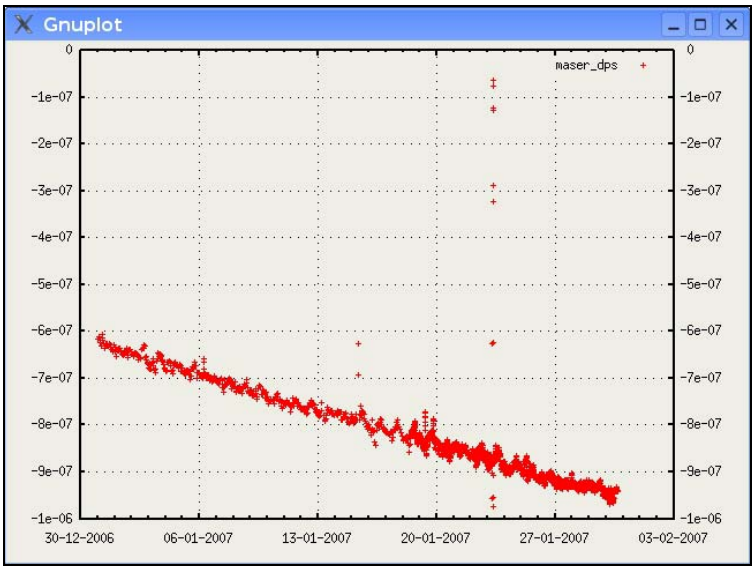

Figura 2. Datos de retardo del máser desde el 1/1/2007 hasta el 31/1/2007 (Es una ampliación del último tramo de la Figura 1).

2) Tras seleccionar las fechas, el programa rellena los cajetines "Retardo inicial", "Retardo final" y "DAC inicial" con valores obtenidos de la base de datos, tal y como se explica en las secciones 3.2.1 y 3.2.3 (**Figura 3**).

3) Para obtener el DAC corregido se pulsa el botón "Calcular", que pone en marcha el algoritmo descrito en la sección 3.2. Con los valores de este ejemplo se obtiene un error relativo en frecuencia de $\approx$ -1.2 10^{-13} y un valor en octal del DAC de 3752.

Figura 3. El programa rellena automáticamente los cajetines de retardo inicial, retardo final y DAC inicial.

4) Además se presenta una gráfica con los valores de retardo entre las fechas inicial y final obtenidos directamente de la base de datos y, superpuesta sobre esta, una recta que une los valores de iniciales y finales de retardo, tal y como se describe en la sección 3.2.4 (**Figura 4**).

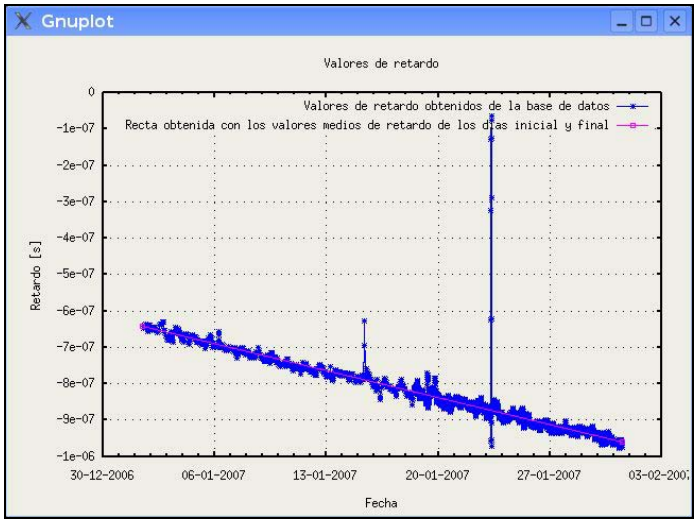

Figura 4. La interfaz representa los valores de retardo junto con una recta (en magenta) que une los valores de retardo inicial y final con el propósito de comprobar la linealidad de los datos.

III. Descripción del código

El programa está integrado por dos módulos: interfaz gráfica y código en lenguaje Python.

3.1) Diseño gráfico

3.1.1) Interfaz gráfica

La interfaz gráfica se muestra en la **Figura 5**. Ha sido diseñada con el entorno QtDesigner y se puede subdividir en dos partes:

A) Entradas

- Cajetín "Fecha inicial": Establece el día inicial del tramo de la curva *retardo-fecha* deseada.

- Cajetín "Fecha final": Establece el día final del tramo de la curva *retardo-fecha* deseada.

- Cajetín "Retardo inicial": Media de los valores de retardo para el día establecido en fecha inicial. El programa accede a la base de datos, calcula la media de los valores de retardo para ese día y lo presenta en este cajetín automáticamente.

- Cajetín "Retardo final": Media de los valores de retardo para el día establecido en fecha final. El programa accede a la base de datos, calcula la media de los valores de retardo para ese día y lo presenta en este cajetín automáticamente.

Figura 5. Aspecto final de la interfaz gráfica

- Cajetín "DAC inicial": Valor en octal del DAC más reciente almacenado en la base de datos. El programa accede a la base de datos, obtiene ese valor y lo presenta en este cajetín automáticamente.

B) Salidas

Tras pulsar el botón "Calcular" se obtiene:

- Cajetín "Df/f": Pendiente de la curva retardo-fecha seleccionada.

- Cajetín "DAC final": Valor en octal del DAC corregido.

3.1.2) Gráfica para comprobar la linealidad del tramo de retardo

Tras pulsar el botón "Calcular", se abre una nueva ventana con los valores de retardo -obtenidos de la base de datos-, comprendidos entre las fechas inicial y final, junto con una recta que une los valores de retardo presentados en los cajetines "Retardo inicial" y "Retardo final" para verificar así la linealidad del tramo elegido.

3.2) Algoritmos

El lenguaje de programación empleado ha sido Python y el código está modularizado en funciones lo cual facilita su comprensión y la depuración de errores. Se hace uso de la librería *gnuplot* para representar gráficamente los valores de retardo. Para no bloquear la ventana de introducción de datos y poder repintar las gráficas cada vez que se pulsa el botón "Calcular" se emplean hilos (*threads)* y para refrescar los cajetines *retardo inicial, retardo final y DAC inicial* con sus nuevos valores cuando se cambian los valores de fecha inicial y/o fecha final se emplean *slots.* El código completo se muestra en el Apéndice I.

3.2.1) Cálculo de los valores de retardo iniciales y finales

Al introducir las fechas iniciales y finales del tramo que se desea, el programa automáticamente rellena los campos "Retardo inicial" y "Retardo final" tras hacer una consulta a la base de datos y obtener una media con los valores de retardo para dichos días.

El código está en la función `computeDelayMeanForDay` y se muestra en el Apéndice I.

3.2.2) Cálculo del error relativo en frecuencia

El error relativo en frecuencia df/f es la pendiente de la recta que une los puntos $(\Delta t_1, t_1)$, $(\Delta t_2, t_2)$, donde Δt_1 es el retardo inicial, Δt_2 es el retardo final (obtenidos en la sección 3.2.1), t_1 es la fecha inicial y t_2 la fecha final (ambas en formato de día juliano). Matemáticamente,

$$\frac{df}{f} = \frac{\Delta t_2 - \Delta t_1}{86400 \cdot (t_2 - t_1)} \qquad [1]$$

El resultado se presenta en la interfaz gráfica en el cajetín “Df/f”.

El código está en la función `calcula_dac` y se muestra en el Apéndice I.

3.2.3) Cálculo del DAC final

El algoritmo calcula el nuevo DAC en los siguientes pasos.

1) Accede a la base de datos para obtener el valor más reciente del DAC y lo presenta en la interfaz gráfica en el cajetín “Dac inicial”.

   El código está en la sección de inicialización del programa y se muestra en el Apéndice I.

2) Se calcula el valor (paso) con el que hay que incrementar o decrementar el valor del DAC inicial. Para ello hay que utilizar la curva respuesta en frecuencia – tensión del varactor, obtenida en Nizhny Novgorod (Rusia) y presentada en el informe IT/OAN-CAY 2005-8 (**Figura 6**).

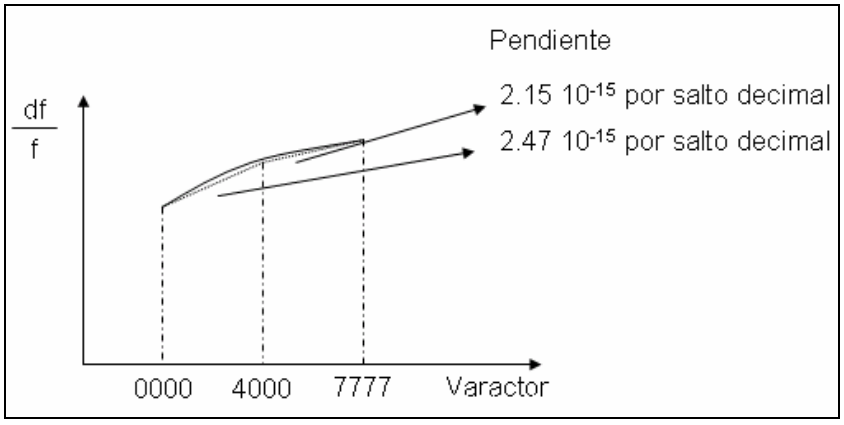

Figura 6. Dependencia relativa de la frecuencia con la posición del varactor

Según la Figura 6, se tiene:

$$\text{- Si DAC inicial} < 4000 \Rightarrow \text{paso} = \left| \frac{df/f}{2.47 \cdot 10^{-15}} \right| \qquad [2]$$

$$\text{- Si DAC inicial} \geq 4000 \Rightarrow \text{paso} = \left| \frac{df/f}{2.15 \cdot 10^{-15}} \right| \qquad [3]$$

El código está en la función `calcula_dac` y se muestra en el Apéndice I.

3) El DAC inicial se pasa de octal a decimal. La conversión de octal a decimal se realiza multiplicando los dígitos del número por su peso octal y sumándolos posteriormente.

Ejemplo: el número octal 3670 convertido a decimal es: $3*8^3 + 6*8^2 + 7*8^1 + 0*8^0 = 1976$

El código está en la función `calcula_dac` y se muestra en el Apéndice I.

4) Se añade el valor del paso calculado al valor del DAC inicial en decimal.

$$\text{- Si } \frac{df}{f} < 0 \Rightarrow \text{DAC final decimal} = \text{DAC inicial decimal} + \text{paso} \qquad [4]$$

$$\text{- Si } \frac{df}{f} > 0 \Rightarrow \text{DAC final decimal} = \text{DAC inicial decimal} - \text{paso} \qquad [5]$$

El código está en la función `calcula_dac` y se muestra en el Apéndice I.

5) Por último se convierte el valor final del DAC de decimal a octal. La conversión de decimal a octal es un poco más laboriosa que la de octal a decimal y se realiza dividiendo el número por 8 y escribiendo el resto como el dígito menos significativo, proceso que se repite hasta que el cociente es 0.

- Ejemplo: la conversión en octal del número decimal 2026 es:

División	Cociente	Resto	Número octal
2026 / 8	253	2	2
253 / 8	31	5	52
31 / 8	3	7	752
3 / 8	0	3	3752

Tabla 1. Conversión del número decimal 2026 en octal

El DAC octal así calculado se presenta en la interfaz gráfica en el cajetín "DAC final".

El código está en la función `calcula_dac` y se muestra en el Apéndice I.

3.2.4) Representación gráfica

Una vez que el programa calcula el DAC muestra a su vez una gráfica con los datos de retardo entre las fechas inicial y final obtenidos directamente de la base de datos. Superpuesta a esta, el programa también dibuja una recta que une los valores medios de retardo para la fecha inicial y final para poder verificar de forma visual la linealidad de los valores de retardo.

La representación gráfica se ejecuta en un proceso independiente al de la interfaz gráfica gracias al empleo de hilos (*threads*) de Linux. Esto permite cambiar los valores de la interfaz sin tener que cerrar la ventana gráfica.

El código está en la función `plotDac` y se muestra en el Apéndice I.

IV. Bibliografía

[1] P. de Vicente, A. Garrigues, *Monitorización del máser de hidrógeno del CAY*, Informe Técnico IT- OAN 2005-5.

[2] A. Barcia, P. de Vicente, *Caracterización del máser de hidrógeno del CAY*, Informe Técnico IT-OAN 2005-8.

[3] P. de Vicente, I. López, *Marcas temporales en el sistema VLBI del CAY***,** Informe Técnico IT-OAN 2001-8

[4] P. de Vicente**,** *Programas de control y monitorización de la estación meteorológica y del terminal GPS del CAY*, Informe Técnico IT- OAN 2002-2.

[5] D. Cordobés, P. de Vicente, J. Fernández, C. Almendros, J.Mª Yagüe, *Traslado de los equipos de VLBI al radiotelescopio de 40m*, Informe Técnico IT- OAN 2007-1.

Apéndice I: Código en Python

```
#!/usr/bin/env python
# -*- coding: ISO-8859-15 -*

#/* Programa para corregir la deriva del maser
# *
# * Copyright (C) 2006
# * Observatorio Astronomico Nacional, Spain
# *
# * This library is free software; you can redistribute it and/or modify it
under
# * the terms of the GNU Library General Public License as published by the
Free
# * Software Foundation; either version 2 of the License, or (at your option)
any
# * later version.
# *
# * This library is distributed in the hope that it will be useful, but WITHOUT
# * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY FITNESS
# * FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for
more
# * details.
# *
# * You should have received a copy of the GNU Library General Public License
# * along with this library; if not, write to the Free Software Foundation,
Inc.,
# * 675 Massachusetts Ave, Cambridge, MA 02139, USA. Correspondence concerning
# * APEX should be addressed as follows:
# *
# *  who                when          what
# * --------        ----------  --------------------------------
# * David Cordobes    09/12/2006      created
# */


from plotOptions import *
from threading import *
import sys,time, _mysql
from qt import *
from string import *
from deriva_maser_gui import *
from mx.DateTime import *
import Gnuplot, Gnuplot.funcutils
import math,os


exceptionFlag = 0

class gplotX(Thread):
        def __init__(self):
               Thread.__init__(self)
               self.g0 = Gnuplot.Gnuplot()

        def run(self):
               self.g0.reset()


class maserDelay(deriva_maser_gui):

       def __init__(self):

               deriva_maser_gui.__init__(self)

               now = QDate()
```

```
        now.setYMD(time.localtime()[0],time.localtime()[1],time.localtime()[2])

        self.fecha_ini_gui.setDate(now.addDays(-30))
        self.fecha_fin_gui.setDate(now)

        self.slotFillStartDelay()
        self.slotFillEndDelay()

        try:
               db = _mysql.connect(host = "localhost", user = "maserdbuser",passwd =
               "kvarzch1-75", db = "maserdb"
        except:
               print "Error conectando a la base de datos."
               sys.exit(-1)

        query_str = "select dac from maserstatus ORDER BY mjd desc"

        db.query(query_str)

        r = db.store_result()

        filas = r.fetch_row(0)
        db.close()

        str_dac_ini = filas[0][0]

        self.dac_ini_gui.setText(str_dac_ini)

def fromQDateToMxDateTime(self, dtqt):
        ''' Returns a MxDatetime object from a QDate object
        @param dtqt: QDate object
        @return MxDatetime object
        '''
        year = dtqt.year()
        month = dtqt.month()
        day = dtqt.day()
        dtmx = DateTime(year, month, day)

        return dtmx

def fromQDateTomySQLChar(self, dtqt):
        ''' Returns a string from a QDateTime object
        @param dtqt: QDate object
        @return SQL string
        '''
        mysqlChar = dtqt.toString("yyyyMMddhhmmss"

        return str(mysqlChar)

def fromMxDateTimeTomySQLChar(self, date):
        '''Computes a string from a mx.DateTime object. This
        string will be used when querying the MYSQL database
        @param date : mxDateTime date object
        @return string with the formatted date and time:
          Year + month + day + hour + minute + second
        '''
        mysqlChar = date.strftime("%Y%m%d%H%M%S"

        return mysqlChar

def computeDelayMeanForDay(self, dateIn):
        '''Computes the mean delay for a given date. It uses all values
        betweeen 00:00 and 23:59
        @param dateIn: mx.DateTime object
        @return mean value in seconds
        '''
        startDate = DateTime(dateIn.year, dateIn.month, dateIn.day,0,0,0)
        endDate = DateTime(dateIn.year, dateIn.month, dateIn.day, 23, 59, 59)

        fechaInicial = self.fromMxDateTimeTomySQLChar(startDate)
```

```
        fechaFinal = self.fromMxDateTimeTomySQLChar(endDate)

        query_str = "select avg(maser_dps) from maserdata WHERE abs(maser_dps)<1
        and ts >= " + fechaInicial + " && ts < " + fechaFinal

        try:
                db = _mysql.connect(host = "localhost", user = "maserdbuser",passwd =
                "kvarzch1-75", db = "maserdb"
        except:
                print "Error conectando a la base de datos."
                sys.exit(-1)

        db.query(query_str)
        r = db.store_result()
        filas = r.fetch_row(0)

        db.close()

        #try:
        meanValue=atof((filas[0][0]))

        #except TypeError:

        #       raise "no_hay_datos"
        #       return

        return meanValue

def calcula_dac(self):
        '''Calculates the new DAC and prints it on the GUI
        '''

        str_dac_ini=str((self.dac_ini_gui.text()))
        int_dac_ini = atoi(str_dac_ini) #Obtengo el valor actual del DAC

        if (str(self.ret_ini_gui.text())=="" and str(self.ret_fin_gui.text())=="":

                QMessageBox.warning(self,"Error","Has de introducir una fecha inicial
                y final validas")
                return
        else:
                if str(self.ret_ini_gui.text())=="":
                        QMessageBox.warning(self,"Error","Has de introducir una fecha
                        inicial valida")
                        return
                if str(self.ret_fin_gui.text())=="":
                        QMessageBox.warning(self,"Error","Has de introducir una fecha
                        final valida")
                        return

        ret_ini=atof(str(self.ret_ini_gui.text()))*1.e-9
        ret_fin=atof(str(self.ret_fin_gui.text()))*1.e-9

        value = self.fecha_ini_gui.date() #Value es un objeto QDate
        dtmx = self.fromQDateToMxDateTime(value) #dtmx es un objeto mxDateTime
        djm_ini=dtmx.mjd

        value = self.fecha_fin_gui.date() #Value es un objeto QDate
        dtmx = self.fromQDateToMxDateTime(value) #dtmx es un objeto mxDateTime
        djm_fin=dtmx.mjd

        # Calculo la pendiente
        deltaf=(ret_fin-ret_ini)/(86400.0*(djm_fin-djm_ini))

        #self.dac_deltaf_gui.setText("%12.11f"%(deltaf) )
        self.dac_deltaf_gui.setText(str(deltaf) )

        # Calculo el paso que tengo que sumar o restar al valor del DAC actual
        if int_dac_ini<4000:
                paso=abs(deltaf//2.47e-15)
```

```
        else:
                paso=abs(deltaf//2.15e-15)

        #Convierto el valor del DAC de octal a decimal

        dac_ini_decimal=0

        for i in range(len(str_dac_ini)):

                dac_ini_decimal=dac_ini_decimal+float(str_dac_ini[len(str_dac_ini)-i-
                1])*pow(8,i)

        # Obtengo el nuevo DAC en decimal

        if deltaf<0:
                dac_fin_decimal=dac_ini_decimal+paso
        else:
                dac_fin_decimal=dac_ini_decimal-paso

        #Convierto el valor del DAC de decimal a octal
        """
        dac_octal_fin=""

        for i in range(10): #Asumo que en la conversion en un caso extremo se
        tengan 10 iteraciones como mucho

                dac_octal_cociente=int(dac_fin_decimal//8)
                dac_octal_resto=int(dac_fin_decimal%8)

                dac_octal_fin=str(dac_octal_resto)+dac_octal_fin

                if i==0:
                        dac_octal_fin=str(dac_octal_resto)

                if dac_octal_cociente==0:
                        break

                dac_fin_decimal=dac_octal_cociente
        """

        dac_octal_fin= "%o" % dac_fin_decimal
        self.dac_fin_gui.setText(dac_octal_fin)

        #La paso las fechas en formato QDate
        self.plotDac(self.fecha_ini_gui.date(),self.fecha_fin_gui.date())

def slotFillStartDelay(self):
        '''Reads the date from the fecha_ini_gui widget and
        computes the corresponding mx.DateTime object and gets
        the mean delay for that date reading the database
        '''
        global exceptionFlag
        value = self.fecha_ini_gui.date() #Value es un objeto QDate
        dtmx = self.fromQDateToMxDateTime(value) #dtmx es un objeto mxDateTime

        try:
                delay = self.computeDelayMeanForDay(dtmx)
                delay_ns=delay*1e9
                self.ret_ini_gui.setText("%6.3f"%(delay_ns) )
        except TypeError:

                if exceptionFlag==0:

                        exceptionFlag=1
                        QMessageBox.warning(self,"Error","No hay datos para la fecha
                        indicada")
                        self.ret_ini_gui.setText("")
                        return

        exceptionFlag=0
```

```
def slotFillEndDelay(self):
        '''Reads the date from the fecha_ini_gui widget and
        computes the correspondinf mx.DateTime object and computes
        the mean delay for that date reading the database
        '''
        global exceptionFlag
        value = self.fecha_fin_gui.date() #Value es un objeto QDate
        dtmx = self.fromQDateToMxDateTime(value) #dtmx es un objeto mxDateTime

        try:
                delay = self.computeDelayMeanForDay(dtmx)
                delay_ns=delay*1e9
                self.ret_fin_gui.setText("%6.3f"%(delay_ns) )
        except TypeError:
                if exceptionFlag==0:

                        exceptionFlag=1
                        QMessageBox.warning(self,"Error","No hay datos para la fecha
                        indicada")
                        self.ret_fin_gui.setText("")
                        return

        exceptionFlag=0

def plotDac(self, dateIn, dateEnd):

        '''Plots all the delay values between dateIn@12h00m00s and dateEnd@12h00m00s.
        Also, it prints a straight line beginning in the mean delay of dateIn and
        ending in the mean delay of dateEnd
        @param dateIn: QDate object
        @param dateEnd: QDate object
        '''
        self.gp = gplotX()
        self.gp.start()

        startDate = DateTime(dateIn.year(), dateIn.month(), dateIn.day(),12,0,0)
        endDate = DateTime(dateEnd.year(), dateEnd.month(), dateEnd.day(), 12, 0, 0)

        delta=endDate.mjd-startDate.mjd

        startDate=self.fromMxDateTimeTomySQLChar(startDate)
        endDate=self.fromMxDateTimeTomySQLChar(endDate)

        try:
                db = _mysql.connect(host = "localhost", user = "maserdbuser",passwd =
                "kvarzch1-75", db = "maserdb"
        except:
                print "Error conectando a la base de datos."
                sys.exit(-1)

        ret_ini=atof(str(self.ret_ini_gui.text()))*1.e-9
        ret_fin=atof(str(self.ret_fin_gui.text()))*1.e-9


        query_str = "select maser_dps,mjd,ts from maserdata WHERE ts >= " + startDate
        + " && ts < " + endDate+" ORDER BY mjd"

        db.query(query_str)

        r = db.store_result()

        filas = r.fetch_row(0)
        db.close()


        temp_file="/home/maser/data/temp.log"
        temp_file2="/home/maser/data/temp2.log"

        fOut = open(temp_file,'w')

        for f in filas:
```

```
                        x = atof(f[2])
                        y1 = atof(f[0])
                        fOut.write("%f %18.16f\n" %(x, y1))

        fOut.close()

        self.gp.g0.title('Valores de retardo')

        self.gp.g0.xlabel('Fecha')
        self.gp.g0.ylabel('Retardo [s]')

        self.gp.g0('set xdata time')
        self.gp.g0('set timefmt "%Y%m%d%H%M%S"')

        if delta > 180:
                self.gp.g0('set format x "%m-%Y"')
        else:
                self.gp.g0('set format x "%d-%m-%Y"')

        self.gp.g0('set grid x y')

        self.gp.g0.plot(Gnuplot.File(temp_file, using=(1,2),with = 'lp 3
        3',title='Valores de retraso obtenidos de la base de datos'))

        fOut = open(temp_file2,'w')

        fOut.write("%f %18.16f\n" %(atof(startDate), atof(ret_ini)))
        fOut.write("%f %18.16f\n" %(atof(endDate), atof(ret_fin)))

        fOut.close()

        self.gp.g0.replot(Gnuplot.File(temp_file2, using=(1,2),with = 'lp 4
        4',title='Recta obtenida con los valores medios de retardo de los dias
        inicial y final'))


        """
        #Calculo la recta
        pendiente=(ret_fin-ret_ini)/(djm_fin-djm_ini)
        constante=ret_fin-pendiente*djm_fin


        for f in filas:
                y=pendiente*atof(f[1])+constante
                pairs.append([atof(f[1]),y])

        """

def slotRmFiles(self):

        line =  os.popen('ls /home/maser/data/ | grep temp.log').readlines()
        if line != []:
                os.system('rm /home/maser/data/temp.log')
                os.system('rm /home/maser/data/temp2.log')

def main(args):

        ap = QApplication(sys.argv)

        sWindow = maserDelay()
        ap.setMainWidget(sWindow)

        sWindow.show()

        sWindow.connect(sWindow.fecha_ini_gui,SIGNAL("valueChanged(const QDate
        &",sWindow.slotFillStartDelay)
        sWindow.connect(sWindow.fecha_fin_gui,SIGNAL("valueChanged(const QDate
        &",sWindow.slotFillEndDelay)
        sWindow.connect(sWindow.calculaButton,SIGNAL("clicked()",
        sWindow.calcula_dac)
        sWindow.connect(ap, SIGNAL('lastWindowClosed()'), sWindow.slotRmFiles)
```

```
                ap.exec_loop()

if __name__=="__main__":
        main(sys.argv)
```