An open-source approach to automation in organic synthesis: the flow chemical formation of benzamides using an inline liquid-liquid extraction system and a homemade 3-axis autosampling/product-collection device.

SUPPORTING INFORMATION

Matthew O'Brien[a]*, April Hall[a], John Schrauwen[ab] and Joyce van der Made[ab]

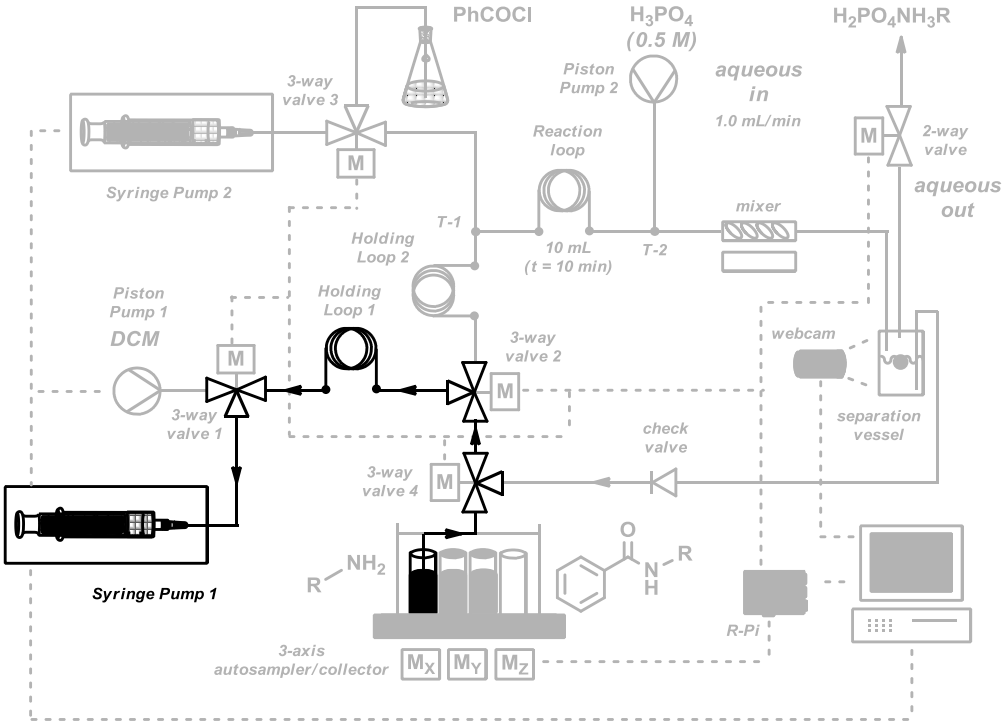
[a]*School of Chemical and Physical Sciences, Lennard-Jones laboratories, Keele University, Borough of Newcastle-under-Lyme, Staffordshire, ST5 5BG, United Kingdom of Great Britain and Northern Ireland.*

[b]*Avans University of Applied Sciences, Lovensdijkstraat 61 – 63, 4818 AJ Breda, North Brabant, the Netherlands*.

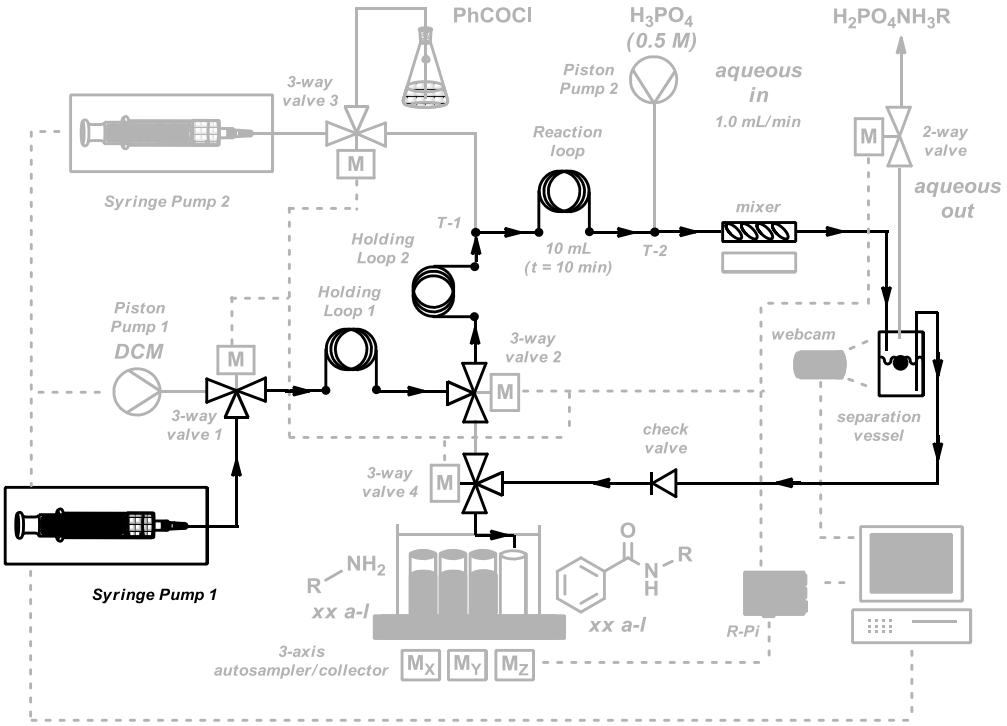
Overview of the Stages of the Flow Sequence

(NB: aqueous in pump also runs continually from the start)

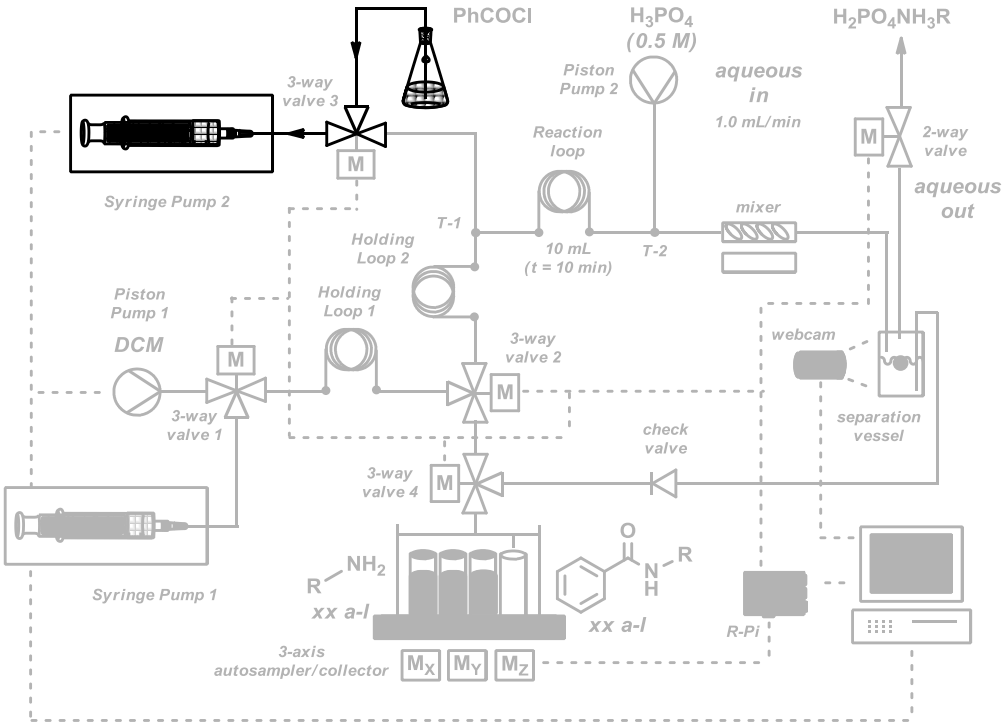
Step 1: Take in 5 mL of DCM to Syringe Pump 1, part filling holding loop 1 (with ca. 4 mL of starting material solution). Aqueous out tap is help open during this step.



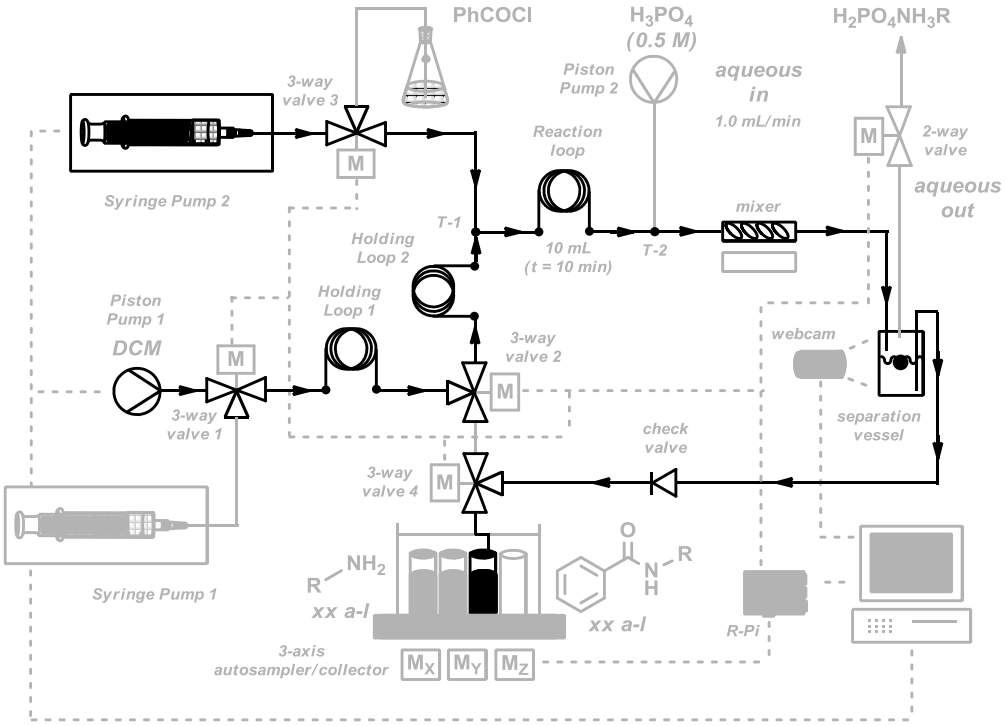
Step 2: Switch Valves. Push out 5 mL of DCM from Syringe Pump 1, transferring starting material in Holding Loop 1 to Holding Loop 2 and flushing residual starting material from take up needle line.



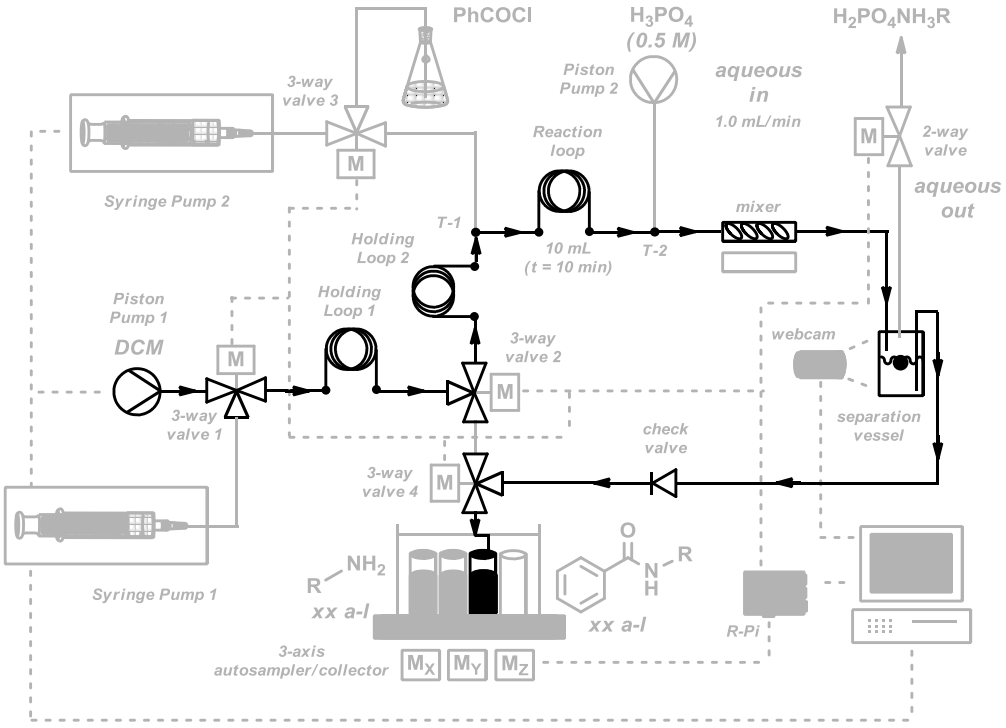
Step 3: Take in 3 mL of acid chloride solution into Syringe Pump 2.



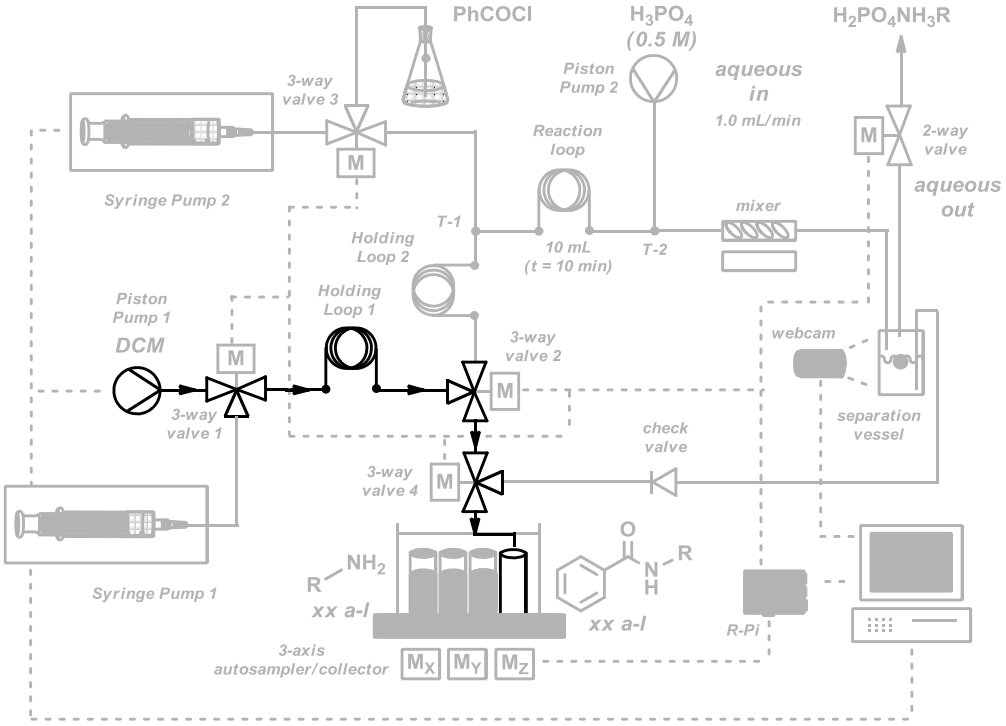
Step 4: Switch Valves, push starting material amine from Holding Loop 2 and acid chloride from Syringe Pump 2, to meet and react at T-1. (Syringe Pump 2 starts pumping only after 1.5 mL of DCM has been dispensed from Piston Pump 1, allowing an overlap of 0.5 mL at start and end of starting material pulse).



Step 5: Continue to pump from Piston Pump 1



Step 6: Switch Valves, pump DCM through Piston Pump 1 to flush residual starting material solution between 3-Way Valve 2 and 3-Way Valve 4.



Python Control Script (pycontrol.py):

```
import cv2
import numpy as np
import time
from matplotlib import pyplot as plt
import serial
import threading
from Queue import Queue
import sys

aqhold = False

dim = 30
step = 0.0318

startpos = (1,1) #waste
currentpos = (1,1)

#positions of starting material vials
#change to suit arrangement
pickup = [(1,2), (1,3), (1,4)]

#positions of product collection vials
#change to suit arrangement
putdown = [(1,6), (2,2), (2,3)]


camport = 0

cam = cv2.VideoCapture(camport)

ramp_frames = 20

posflag = 'high'
tapflag = 'closed'

startflag = False


drag_start = None
drag_end = None
box = (0,0,0,0)

interface = 0.5

kernel = np.ones((5,5),np.uint8)
kernel2 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))

huevar = 10

rpos = None

ser_rpi = serial.Serial() #to rpi
ser_rpi.port = 20
ser_rpi.baudrate = 9600
ser_rpi.open()

ser_sp1 = serial.Serial() #to syringe pump 1
ser_sp1.port = 57
ser_sp1.open()
```

```
#NB: serials commands were sent to pumps multiple times
#this avoided rare occurences of 'dropped' signals.
ser_sp1.write('00 rat 1\x0D')
time.sleep(0.1)
ser_sp1.write('00 rat 1\x0D')
time.sleep(0.1)
ser_sp1.write('00 rat 1\x0D')
time.sleep(0.1)
ser_sp1.write('00 dir wrd\x0D')
time.sleep(0.1)
ser_sp1.write('00 dir wrd\x0D')
time.sleep(0.1)
ser_sp1.write('00 dir wrd\x0D')
time.sleep(0.1)

ser_sp2 = serial.Serial() #to syringe pump 2
ser_sp2.port = 58
ser_sp2.open()
ser_sp2.write('01 rat 1\x0D')
time.sleep(0.1)
ser_sp2.write('01 rat 1\x0D')
time.sleep(0.1)
ser_sp2.write('01 rat 1\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir wdr\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir wdr\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir wdr\x0D')
time.sleep(0.1)

ser_p1 = serial.Serial() #to piston pump
ser_p1.port = 59
ser_p1.open()
ser_p1.write('flow 1000 \x0D')
time.sleep(0.1)
ser_p1.write('flow 1000 \x0D')
time.sleep(0.1)
ser_p1.write('flow 1000 \x0D')
time.sleep(0.1)

print 'serials opened ok'

rpiq = Queue(maxsize=0)

def dostuff():
    global currentpos
    global pickup
    global putdown
    global aqhold
    global rpiq

    for i in range(len((pickup))):

        print 'current pos = ' + str(currentpos)

        #move to next pickup:
        deltax = pickup[i][0] - currentpos[0]
        deltay = pickup[i][1] - currentpos[1]
```

```
xsteps = int(dim*deltax/step)
print 'xsteps = ' + str(xsteps)
ysteps = int(dim*deltay/step)
print 'ysteps = ' + str(ysteps)

if abs(xsteps)>0.1:
    t = ('mvx%s' % str(xsteps))
    t = t + '\n'
    rpiq.put(t)
    print 'moving x'
    time.sleep((abs(xsteps))/400.0)

if abs(ysteps)>0.1:
    t = ('mvy%s' % str(ysteps))
    t = t + '\n'
    rpiq.put(t)
    print 'moving y'
    time.sleep((abs(ysteps))/400.0)

rpiq.put('mvz-2000\n') #lower needle
print 'lowering needle'
time.sleep(5)

print 'deltax = ' + str(deltax)
print 'deltay = ' + str(deltay)
currentpos = pickup[i]
print 'pickup %d' %(i+1) + str(currentpos)



#do stuff with syringes
rpiq.put('twb-1600\n') #opens V2-V4
time.sleep(0.1)
rpiq.put('twd1600\n') #opens V2-V4
print 'opening t2-t4'
aqhold = True #flag to hold tpo open
print 'aqhold = True'
time.sleep(10)


ser_sp1.write('00 dir wdr\x0D')
time.sleep(0.1)
ser_sp1.write('00 dir wdr\x0D')
time.sleep(0.1)
ser_sp1.write('00 dir wdr\x0D')
time.sleep(0.1)
ser_sp1.write('00 run\x0D')
time.sleep(0.1)
ser_sp1.write('00 run\x0D')
time.sleep(0.1)
ser_sp1.write('00 run\x0D')
print 'taking up into sp1'
time.sleep(300) #5 mins (5ml at 1ml/min) (load speed)

ser_sp1.write('00 stp\x0D')
time.sleep(0.1)
ser_sp1.write('00 stp\x0D')
time.sleep(0.1)
ser_sp1.write('00 stp\x0D')
print 'stopping sp1'
time.sleep(0.1)
```

```
rpiq.put('twb1600\n') #closes V2-V4
time.sleep(0.1)
rpiq.put('twd-1600\n') #closes V2-V4
print 'closing t2-t4'
time.sleep(10)
aqhold = False #allows aq. out to open/close
print 'aqhold = False'

rpiq.put('mvz2000\n') #raise needle
print 'raising needle'
time.sleep(5)

#move to waste
print 'moving to waste'
deltax = startpos[0] - currentpos[0]
deltay = startpos[1] - currentpos[1]

xsteps = int(dim*deltax/step)
print 'xsteps = ' + str(xsteps)
ysteps = int(dim*deltay/step)
print 'ysteps = ' + str(ysteps)

if abs(xsteps)>0.1:
    t = ('mvx%s' % str(xsteps))
    t = t + '\n'
    rpiq.put(t)
    print 'moving x'
    time.sleep((abs(xsteps))/400.0)

if abs(ysteps)>0.1:
    t = ('mvy%s' % str(ysteps))
    t = t + '\n'
    rpiq.put(t)
    print 'moving y'
    time.sleep((abs(ysteps))/400.0)

print 'deltax = ' + str(deltax)
print 'deltay = ' + str(deltay)
currentpos = (currentpos[0]+deltax,currentpos[1]+deltay)
print 'wastepos =  %d' %(i+1) + str(currentpos)

#do stuff with syringes:
print 'pushing from 1st loop to 2nd loop'
ser_sp1.write('00 dir inf\x0D')
time.sleep(0.1)
ser_sp1.write('00 dir inf\x0D')
time.sleep(0.1)
ser_sp1.write('00 dir inf\x0D')
time.sleep(0.1)
ser_sp1.write('00 run\x0D')
time.sleep(0.1)
ser_sp1.write('00 run\x0D')
time.sleep(0.1)
ser_sp1.write('00 run\x0D')
time.sleep(300)
#5 mins, flush s.m. in needle to waste. (5ml at 1ml/min)
ser_sp1.write('00 stp\x0D')
time.sleep(0.1)
ser_sp1.write('00 stp\x0D')
time.sleep(0.1)
ser_sp1.write('00 stp\x0D')
```

```
time.sleep(0.1)
ser_sp2.write('01 dir wdr\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir wdr\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir wdr\x0D')
time.sleep(0.1)
ser_sp2.write('01 run\x0D')
time.sleep(0.1)
ser_sp2.write('01 run\x0D')
time.sleep(0.1)
ser_sp2.write('01 run\x0D')
print 'taking up into sp2'
time.sleep(180) #3 mins, 3ml at 1ml/min
ser_sp2.write('01 stp\x0D')
time.sleep(0.1)
ser_sp2.write('01 stp\x0D')
time.sleep(0.1)
ser_sp2.write('01 stp\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir inf\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir inf\x0D')
time.sleep(0.1)
ser_sp2.write('01 dir inf\x0D')
time.sleep(0.1)

rpiq.put('twa1600\n')
time.sleep(0.1)
rpiq.put('twc1600\n')
print 'connecting sp2 to t-piece'
print 'connecting p1 to 2nd loop'
time.sleep(10)
#move to putdown 1
print 'moving to putdown position'
deltax = putdown[i][0] - currentpos[0]
deltay = putdown[i][1] - currentpos[1]

xsteps = int(dim*deltax/step)
print 'xsteps = ' + str(xsteps)
ysteps = int(dim*deltay/step)
print 'ysteps = ' + str(ysteps)

if abs(xsteps)>0.1:
    t = ('mvx%s' % str(xsteps))
    t = t + '\n'
    rpiq.put(t)
    print 'moving x'
    time.sleep((abs(xsteps))/400.0)

if abs(ysteps)>0.1:
    t = ('mvy%s' % str(ysteps))
    t = t + '\n'
    rpiq.put(t)
    print 'moving y'
    time.sleep((abs(ysteps))/400.0)

print 'deltax = ' + str(deltax)
print 'deltay = ' + str(deltay)
currentpos = putdown[i]
print 'putdown %d' %(i+1) + str(currentpos)
```

```
#do reaction:

ser_p1.write('flow 500 \x0D') #0.5 mL/min for amine
time.sleep(0.1)
ser_p1.write('flow 500 \x0D')
time.sleep(0.1)
ser_p1.write('flow 500 \x0D')
time.sleep(0.1)
print 'setting p1 flow to 0.5 ml per min'
ser_p1.write('on \x0D') #start injecting amine s.m.
time.sleep(0.1)
ser_p1.write('on \x0D')
time.sleep(0.1)
ser_p1.write('on \x0D')
print 'starting injecting amine'
print 'injecting for 5min30'
time.sleep(180) #p1 at 0.5 mL/min for 3 min
ser_sp2.write('01 rat 0.5\x0D')
time.sleep(0.1)
ser_sp2.write('01 rat 0.5\x0D')
time.sleep(0.1)
ser_sp2.write('01 rat 0.5\x0D')
time.sleep(0.1)
print 'starting injection acid chloride'
ser_sp2.write('01 run\x0D') #start injecting acid chloride
time.sleep(0.1)
ser_sp2.write('01 run\x0D')
time.sleep(0.1)
ser_sp2.write('01 run\x0D')
time.sleep(360) #inject acic chloride
#for 6 min (3ml at 0.5 ml/min)
print 'stopping injecting common'
ser_sp2.write('01 stp\x0D') #stop injecting acid chloride
time.sleep(0.1)
ser_sp2.write('01 stp\x0D')
time.sleep(0.1)
ser_sp2.write('01 stp\x0D')
time.sleep(0.1)
print 'increasing p1 flow to 1ml per min'
print 'flowing p1 for 30 mins'
ser_p1.write('flow 1000 \x0D') #increase flow rate of p1
time.sleep(0.1)
ser_p1.write('flow 1000 \x0D')
time.sleep(0.1)
ser_p1.write('flow 1000 \x0D')
time.sleep(1800) #30 mins for p1 to pump at 1mL/min
print 'stopping p1'
ser_p1.write('off \x0D')
time.sleep(0.1)
ser_p1.write('off \x0D')
time.sleep(0.1)
ser_p1.write('off \x0D')
time.sleep(1)

aqhold = True #holds aq. out tap open
print 'aqhold = True'
time.sleep(1)
print 'opening t2-t4'
#open V2-V4 for flush:
rpiq.put('twd1600\n')
```

```
time.sleep(0.1)
rpiq.put('twb-1600\n')
time.sleep(15)
#move to waste:
print 'moving to waste'

deltax = startpos[0] - currentpos[0]
deltay = startpos[1] - currentpos[1]

xsteps = int(dim*deltax/step)
print 'xsteps = ' + str(xsteps)
ysteps = int(dim*deltay/step)
print 'ysteps = ' + str(ysteps)

if abs(xsteps)>0.1:
    t = ('mvx%s' % str(xsteps))
    t = t + '\n'
    rpiq.put(t)
    print 'moving x'
    time.sleep((abs(xsteps))/400.0)

if abs(ysteps)>0.1:
    t = ('mvy%s' % str(ysteps))
    t = t + '\n'
    rpiq.put(t)
    print 'moving y'
    time.sleep((abs(ysteps))/400.0)

print 'deltax = ' + str(deltax)
print 'deltay = ' + str(deltay)
currentpos = (currentpos[0]+deltax,currentpos[1]+deltay)
print 'wastepos =  %d' %(i+1) + str(currentpos)

#flush residue in V2-V4:
print 'flushig t2-t4 tubing'
print 'starting p1 for 4 min'
ser_p1.write('on \x0D')
time.sleep(0.1)
ser_p1.write('on \x0D')
time.sleep(0.1)
ser_p1.write('on \x0D')
time.sleep(240) #flush for 4 min
ser_p1.write('off \x0D')
time.sleep(0.1)
ser_p1.write('off \x0D')
time.sleep(0.1)
ser_p1.write('off \x0D')
time.sleep(1)
print 'stopping p1'
print 'closing t2-t4 section'
print 'moving sp2 back to reservoir'
rpiq.put('twa-1600\n')
time.sleep(0.1)
rpiq.put('twc-1600\n')
time.sleep(0.1)
#close V2-V4:
rpiq.put('twb1600\n')
time.sleep(0.1)
rpiq.put('twd-1600\n')
time.sleep(10)
aqhold = False #allow aq. out to close/open
```

```
        print 'aqhold = False'


listenflag = True

def listenfunc():
    global listenflag
    while listenflag:
        if startflag:
            print 'startflag'
            dostuff()
            listenflag = False


serflag = True

def myserial():
    #print 'myserial'
    global serflag
    global rpiq
    global ser_rpi
    while serflag:
        ser_rpi.write(rpiq.get())
        rpiq.task_done()
    #print 'returning myserial'
    return


mythread2 = threading.Thread(target = myserial)

listenthread = threading.Thread(target = listenfunc)

def poscalc(refpt1, refpt2, scanpt):

    percent = (float(scanpt[1])-float(refpt2[1]))/(float(refpt1[1])-
float(refpt2[1]))
    return percent

def posdo(fract):

    global aqhold
    global ser_rpi
    global tapflag
    global posflag

    if aqhold:

        if tapflag == 'open':
            pass
        if tapflag == 'closed':
            tapflag = 'open'
            this = (str('tpo888').zfill(1) + '\n')
            rpiq.put(this)
            print 'open tap'

    else:

        if posflag == 'high':
            if fract < 0:
                if tapflag == 'closed':
```

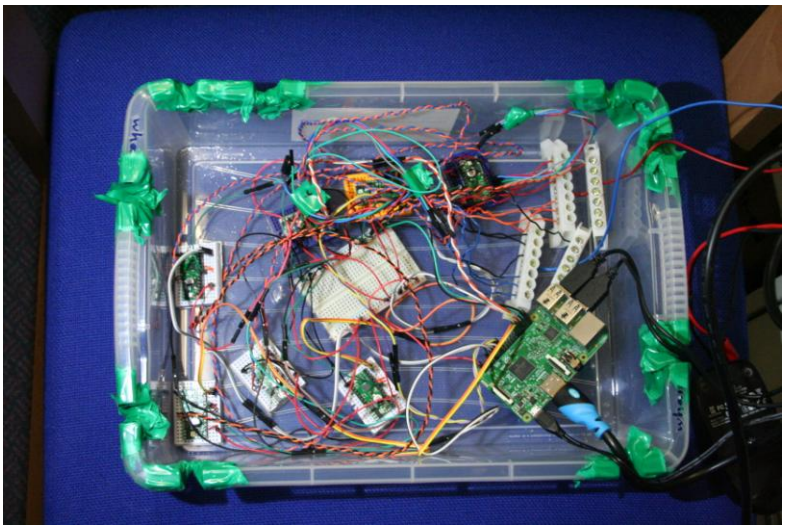
```
                    pass
                if tapflag == 'open':
                    tapflag = 'closed'
                    this = (str('tpo-888').zfill(1) + '\n')
                    rpiq.put(this)
                    print 'close tap'
            if fract > 1:
                posflag = 'low'
        if posflag == 'low':
            if fract > 1:
                if tapflag == 'open':
                    pass
                if tapflag == 'closed':
                    tapflag = 'open'
                    this = (str('tpo888').zfill(1) + '\n')
                    rpiq.put(this)
                    print 'open tap'
            if fract < 0:
                posflag = 'high'


def trackcam(cam, hueval):
    global huevar
    global startflag
    global serflag

    hue1 = hueval - huevar
    hue2 = hueval + huevar

    go = True
    while go:
        retval, f = cam.read()
        hsv = cv2.cvtColor(f, cv2.COLOR_BGR2HSV)
        lower = np.array([hue1, 50, 50])
        higher = np.array([hue2, 255, 255])

        mask = cv2.inRange(hsv, lower, higher)

        opened2 = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel2)

        _, contours, heirarchy = cv2.findContours(opened2, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        areas = [cv2.contourArea(c) for c in contours]

        if len(areas) > 0:

            max_index = np.argmax(areas)
            cnt = contours[max_index]

            M = cv2.moments(cnt)

            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])
            cv2.circle(f, (cx, cy), 10, (255,255,0), 3)

        cv2.imshow('mask', mask)

        fract = poscalc(refpos2, refpos1, (cx, cy))
        posdo(fract)
```

```
        cv2.circle(f, refpos1, 5, (255,100,100), thickness = 2)
        cv2.circle(f, refpos2, 5, (255,100,100), thickness = 2)

        cv2.imshow('frame', f)

        l = cv2.waitKey(5)
        if l == ord('s'):
            print 'pressed s'
            startflag = True
        if l == 27:
            cv2.destroyAllWindows()
            serflag = False
            break


def takepic():
    retval, im = cam.read()
    return im

def on_mouse2(event, x, y, flags, param):
    global rpos
    if event == cv2.EVENT_LBUTTONDOWN:
        rpos = (x,y)

def getrefpos(image):
    global rpos
    rpos = None
    im = image
    cv2.namedWindow('select refpos', 1)
    cv2.setMouseCallback('select refpos', on_mouse2)
    a = True
    while a:
        if rpos:
            a = False
            cv2.destroyWindow('select refpos')
            return rpos
        cv2.imshow('select refpos', im)
        k = cv2.waitKey(7)
        if k == 27:
            cv2.destroyWindow('select refpos')
            return None

def on_mouse(event, x, y, flags, param):
    global drag_start
    global drag_end
    global box
    if event == cv2.EVENT_LBUTTONDOWN:
        drag_end = False
        drag_start = (x,y)
        print drag_start
        print 'left down'
    if event == cv2.EVENT_LBUTTONUP:
        drag_end = (x,y)
        print drag_end
        print 'left up'

    if drag_start and drag_end:

        xmin = min (drag_start[0], drag_end[0])
        ymin = min (drag_start[1], drag_end[1])
        xmax = max (drag_start[0], drag_end[0])
```

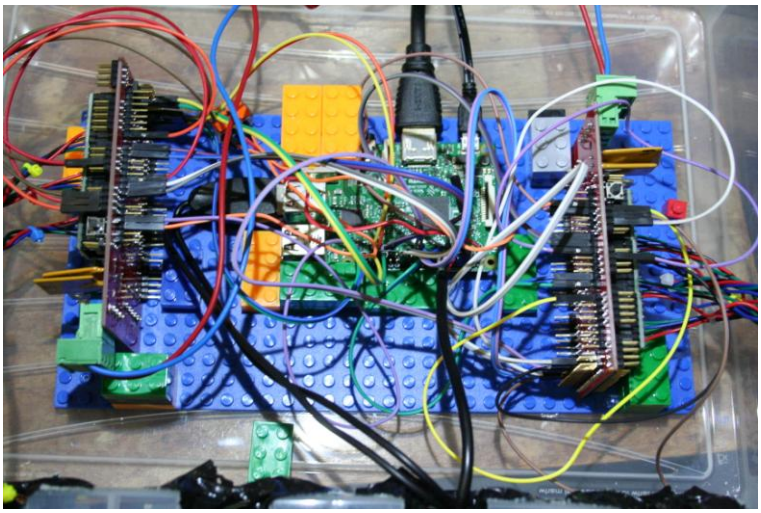
```
            ymax = max (drag_start[1], drag_end[1])
            box = (xmin, ymin, xmax-xmin, ymax-ymin)
        if box:
            print box

def getbox(image):
    print getbox
    im = image
    cv2.namedWindow('select', 1)
    cv2.setMouseCallback('select', on_mouse)
    a = True
    while a:
        if box[2]>0 and box[3]>0:
            cv2.destroyWindow('select')
            return box
        cv2.imshow('select', im)
        k = cv2.waitKey(7)
        if k == 27:
            cv2.destroyWindow('select')
            return None

def gethist(image):
    global box
    im = image
    im2 = im[box[1]:(box[1] + box[3]), box[0]: (box[0]+box[2])]
    hsv = cv2.cvtColor(im2, cv2.COLOR_BGR2HSV)
    hist = cv2.calcHist([hsv], [0], None, [180], [0, 180])

    return hist



for i in range(ramp_frames):
    temp = takepic()

capture = takepic()

refpos1 = getrefpos(capture)
print refpos1
refpos2 = getrefpos(capture)
print refpos2

box1 = getbox(capture)
print box1


hist = gethist(capture)

m = np.argmax(hist)
print 'strongest hue is ' + str(m)

mythread2.start()
listenthread.start()

trackcam(cam, m)

serflag = False

if mythread2.isAlive():
    print 'mythread2 is alive'
if listenthread.isAlive():
```

```
    print 'listenthread is alive'

del (cam)
ser_rpi.close()
del ser_rpi
ser_sp1.close()
del ser_sp1
ser_sp2.close()
del ser_sp2
ser_p1.close()
del ser_p1
sys.exit()
```

Previous RasPi motor control circuitry using individual breadboards for Pololu A4988 motor drivers:



Current RasPi circuitry using 2 RAMPS boards with 8 x Pololu A4988 motor drivers:



Raspberry Pi Python Script (rpi.py):

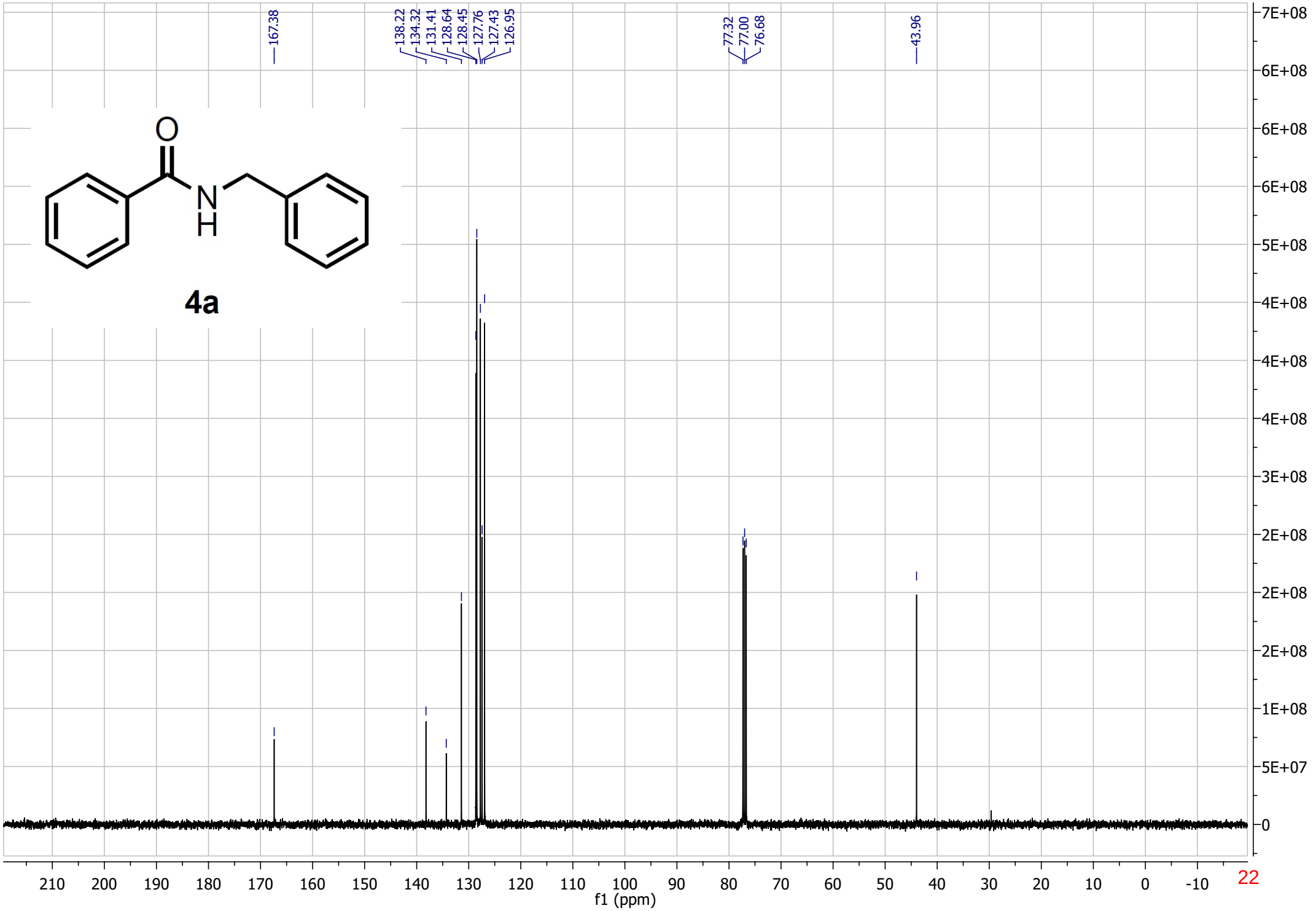
```
import time
import serial
import threading
from Queue import Queue
import RPi.GPIO as gpio

import os

#microstepping on ALL motors set on RAMPS boards

gpio.setmode (gpio.BOARD)

xstep = 24
xdir = 26

ystep = 29
ydir = 31

zstep = 32
zdir = 36

t1step = 18
t1dir = 22

t2step = 12
t2dir = 16

t3step = 13
t3dir = 15

t4step = 7
t4dir = 11

tastep = 19
tadir = 21

steptime = 0.002

gpio.setup(xstep, gpio.OUT)
gpio.setup(xdir, gpio.OUT)
gpio.setup(ystep, gpio.OUT)
gpio.setup(ydir, gpio.OUT)
gpio.setup(zstep, gpio.OUT)
gpio.setup(zdir, gpio.OUT)
gpio.setup(t1step, gpio.OUT)
gpio.setup(t2step, gpio.OUT)
gpio.setup(t3step, gpio.OUT)
gpio.setup(t4step, gpio.OUT)
gpio.setup(tastep, gpio.OUT)
gpio.setup(t1dir, gpio.OUT)
gpio.setup(t2dir, gpio.OUT)
gpio.setup(t3dir, gpio.OUT)
gpio.setup(t4dir, gpio.OUT)
gpio.setup(tadir, gpio.OUT)


gpio.output(xstep, False)
gpio.output(ystep, False)
```

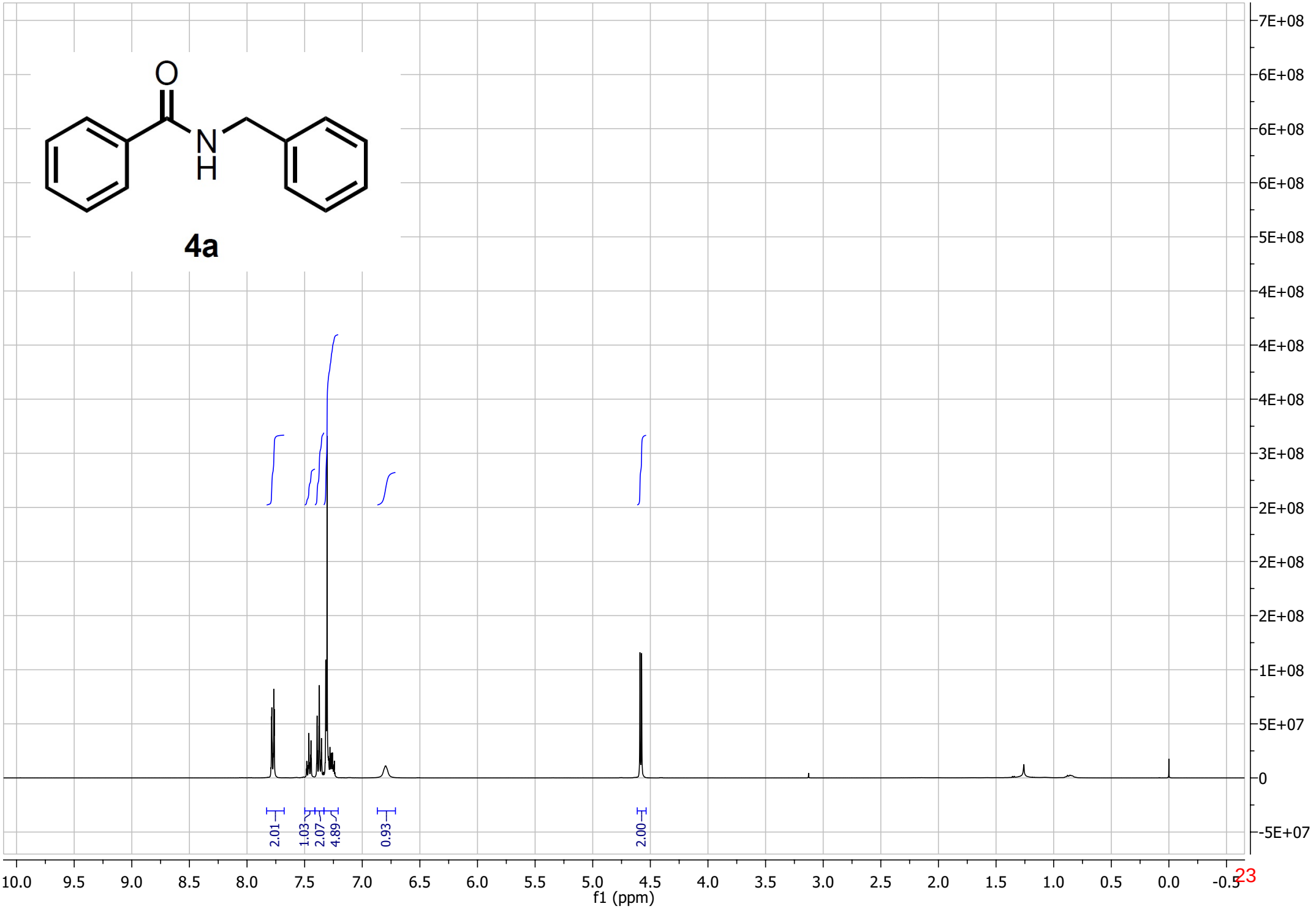
```
gpio.output(zstep, False)
gpio.output(xdir, False)
gpio.output(ydir, False)
gpio.output(zdir, False)
gpio.output(t1dir, False)
gpio.output(t2dir, False)
gpio.output(t3dir, False)
gpio.output(t4dir, False)
gpio.output(tadir, False)


#motors:
#x, y, z, four x 3-way valves, tap-outlet. 8 in total.
#8 functions/threads/queues.
#movex (xq), movey (yq), movez(zq),
#turn1 (t1q), turn2 (t2q), turn3 (t3q),
#turn4 (t4q), turna(taq)


ser = serial.Serial('/dev/ttyAMA0', 9600, timeout = 0.01)

flag = True

xq = Queue(maxsize=0)
yq = Queue(maxsize=0)
zq = Queue(maxsize=0)
#aq = Queue(maxsize=0)
#bq = Queue(maxsize=0)
#cq = Queue(maxsize=0)
#tpoq = Queue(maxsize=0)
t1q = Queue(maxsize=0)
t2q = Queue(maxsize=0)
t3q = Queue(maxsize=0)
t4q = Queue(maxsize=0)
taq = Queue(maxsize=0)

def movex():
    global xq
    while flag:
        a = xq.get()
        print ('mvx ' + str(a)).strip()
        print ('yes')
        if int(a)<0:
            print 'less than 0'
            gpio.output(xdir, False)
        if int(a)>0:
            print 'greater than 0'
            gpio.output(xdir, True)
        for x in range(abs(int(a))):
            gpio.output(xstep, True)
            time.sleep(0.001)
            gpio.output(xstep, False)
            time.sleep(0.001)


def movey():
    global yq
    while flag:
        a = yq.get()
        print ('mvy ' + str(a)).strip()
        if int(a)<0:
```

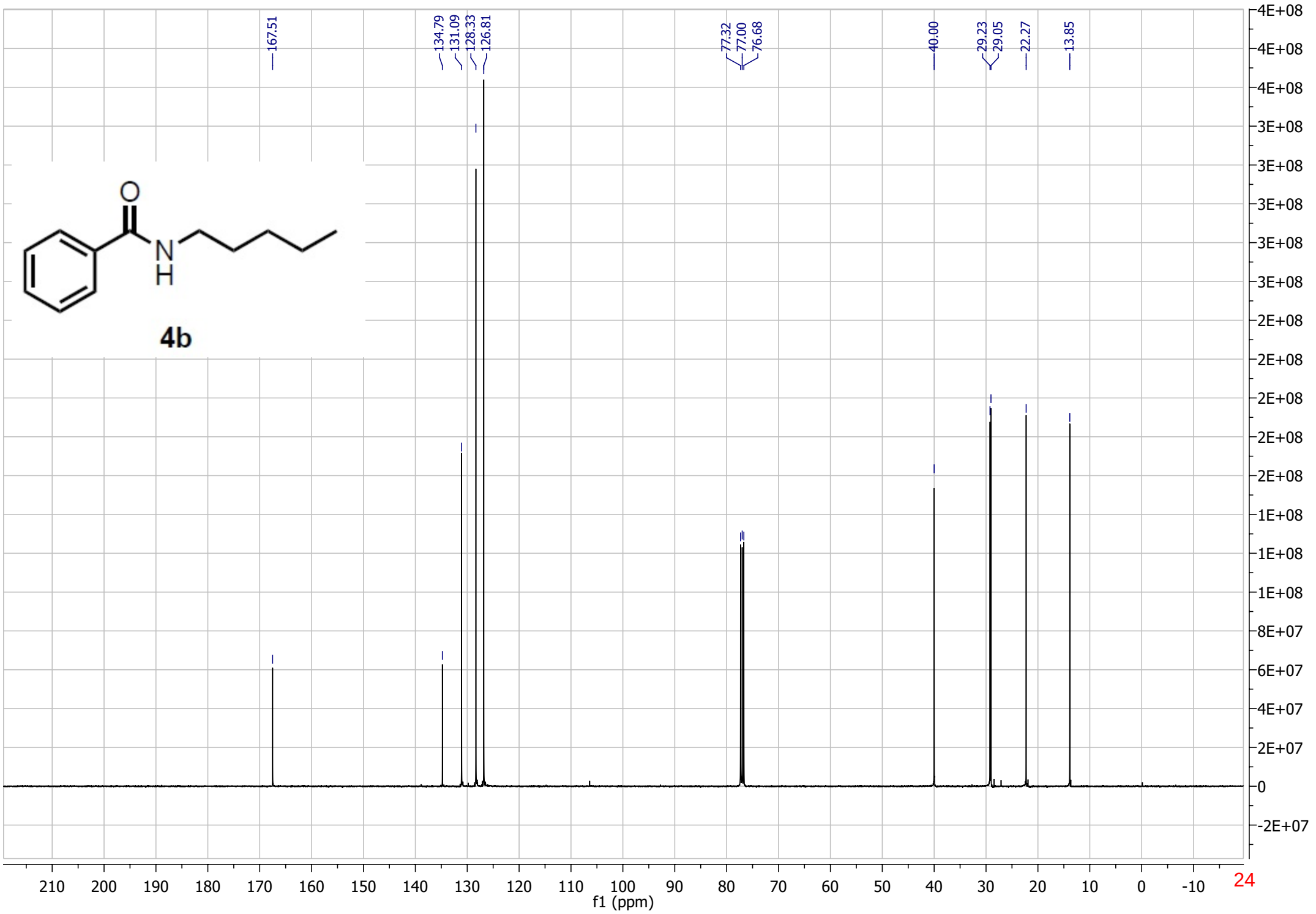
```
            gpio.output(ydir, False)
        if int(a)>0:
            gpio.output(ydir, True)
        for x in range(abs(int(a))):
            gpio.output(ystep, True)
            time.sleep(0.001)
            gpio.output(ystep, False)
            time.sleep(0.001)


def movez():
    global zq
    while flag:
        a = zq.get()
        print ('mvz ' + str(a)).strip()
        if int(a)<0:
            gpio.output(zdir, False)
        if int(a)>0:
            gpio.output(zdir, True)
        for x in range(abs(int(a))):
            gpio.output(zstep, True)
            time.sleep(0.001)
            gpio.output(zstep, False)
            time.sleep(0.001)



def turn1():
    global t1q
    while flag:
        a = t1q.get()
        print ('twa ' + str(a)).strip()
        if int(a)<0:
            gpio.output(t1dir, False)
        if int(a)>0:
            gpio.output(t1dir, True)
        for x in range(abs(int(a))):
            gpio.output(t1step, True)
            time.sleep(steptime)
            gpio.output(t1step, False)
            time.sleep(steptime)



def turn2():
    global t2q
    while flag:
        a = t2q.get()
        print ('twb ' + str(a)).strip()
        if int(a)<0:
            gpio.output(t2dir, False)
        if int(a)>0:
            gpio.output(t2dir, True)
        for x in range(abs(int(a))):
            gpio.output(t2step, True)
            time.sleep(steptime)
            gpio.output(t2step, False)
            time.sleep(steptime)

def turn3():
```

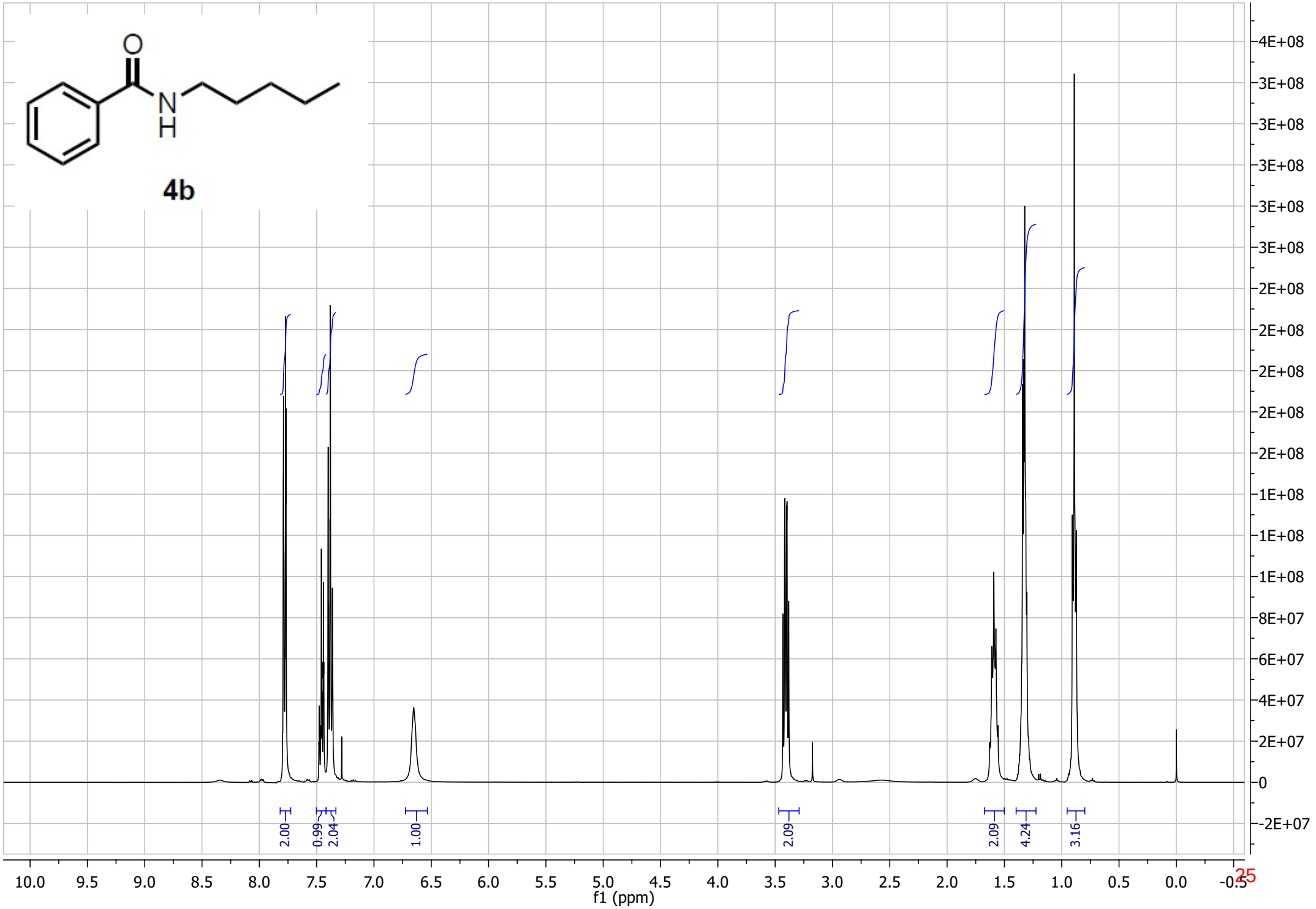
```
    global t3q
    while flag:
        a = t3q.get()
        print ('twc ' + str(a)).strip()
        if int(a)<0:
            gpio.output(t3dir, False)
        if int(a)>0:
            gpio.output(t3dir, True)
        for x in range(abs(int(a))):
            gpio.output(t3step, True)
            time.sleep(steptime)
            gpio.output(t3step, False)
            time.sleep(steptime)

def turn4():
    global t4q
    while flag:
        a = t4q.get()
        print ('twd ' + str(a)).strip()
        if int(a)<0:
            gpio.output(t4dir, False)
        if int(a)>0:
            gpio.output(t4dir, True)
        for x in range(abs(int(a))):
            gpio.output(t4step, True)
            time.sleep(steptime)
            gpio.output(t4step, False)
            time.sleep(steptime)


def turna():
    global taq
    while flag:
        a = taq.get()
        print ('tpo ' + str(a)).strip()
        if int(a)<0:
            gpio.output(tadir, False)
        if int(a)>0:
            gpio.output(tadir, True)
        for x in range(abs(int(a))):
            gpio.output(tastep, True)
            time.sleep(steptime)
            gpio.output(tastep, False)
            time.sleep(steptime)

def listen():
    global xq
    global yq
    global zq
    print 'listen'
    global flag
    while flag:
        time.sleep(0.001) # reduces cpu load
        if (ser.inWaiting()>0):
            x = ser.readline()
            print x


            if x.rstrip() == 'x':
                flag = False
```

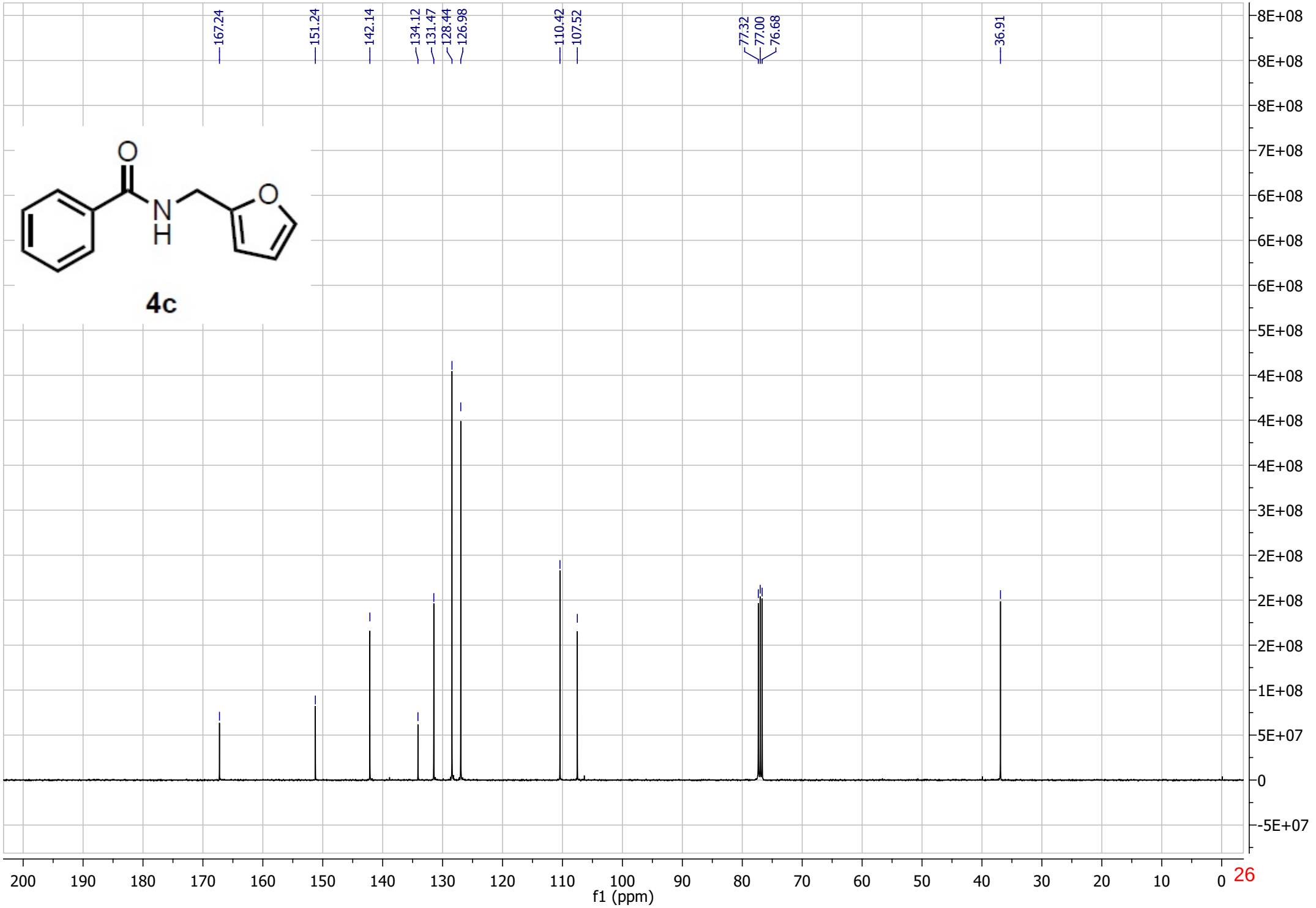
```
            if x.rstrip()[:3] == 'mvx':
                xq.put(x[3:])

            if x.rstrip()[:3] == 'mvy':
                yq.put(x[3:])

            if x.rstrip()[:3] == 'mvz':
                zq.put(x[3:])

            if x.rstrip()[:3] == 'twa':
                t1q.put(x[3:])

            if x.rstrip()[:3] == 'twb':
                t2q.put(x[3:])

            if x.rstrip()[:3] == 'twc':
                t3q.put(x[3:])

            if x.rstrip()[:3] == 'twd':
                t4q.put(x[3:])

            if x.rstrip()[:3] == 'tpo':
                taq.put(x[3:])


listenthread = threading.Thread(target = listen)
xthread = threading.Thread(target = movex)
ythread = threading.Thread(target = movey)
zthread = threading.Thread(target = movez)
t1thread = threading.Thread(target = turn1)
t2thread = threading.Thread(target = turn2)
t3thread = threading.Thread(target = turn3)
t4thread = threading.Thread(target = turn4)
tathread = threading.Thread(target = turna)

xthread.start()
ythread.start()
zthread.start()
t1thread.start()
t2thread.start()
t3thread.start()
t4thread.start()
tathread.start()

listenthread.setDaemon(True)
listenthread.start()
listenthread.join()

ser.close()
del ser
gpio.cleanup()
os._exit(0)
```

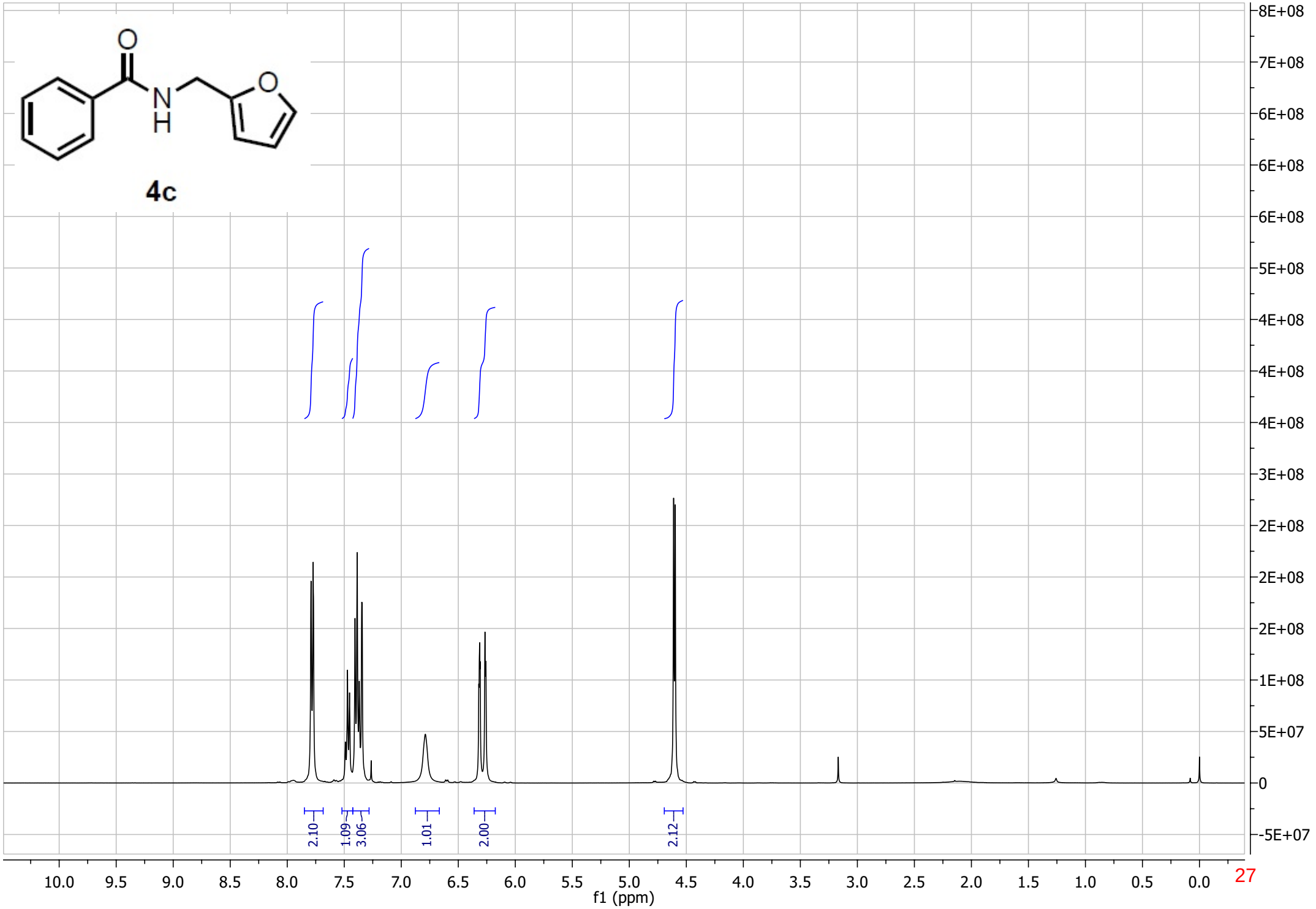

167.38
138.22
134.32
131.41
128.64
128.45
127.76
127.43
126.95
77.32
77.00
76.68
43.96
O
N
H
4a
7E+08
6E+08
6E+08
6E+08
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
210 200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10 0 -10
f1 (ppm)

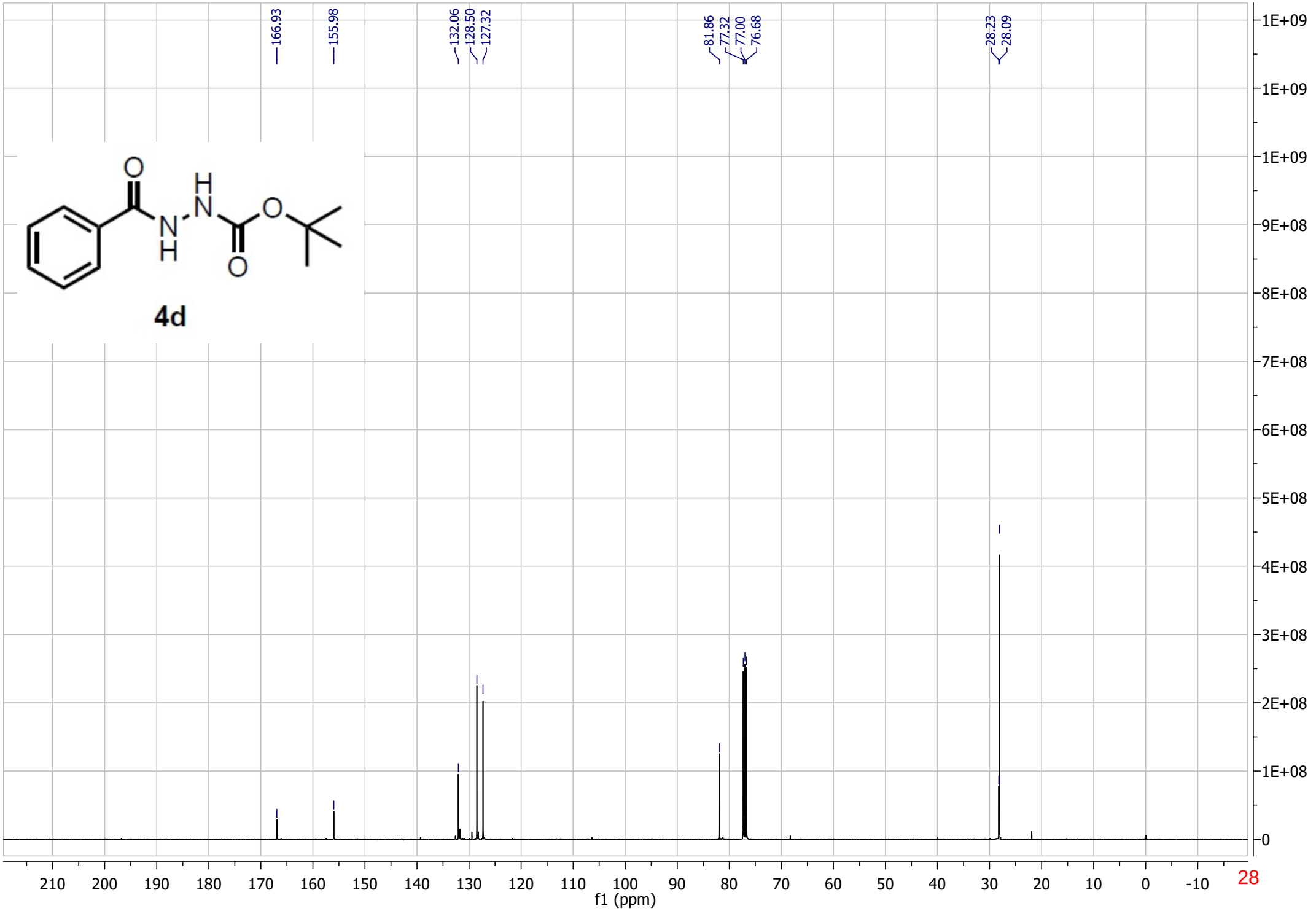

O
N
H
4a
2.01
1.03
2.07
4.89
0.93
2.00
7E+08
6E+08
6E+08
6E+08
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
-5E+07
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
f1 (ppm)

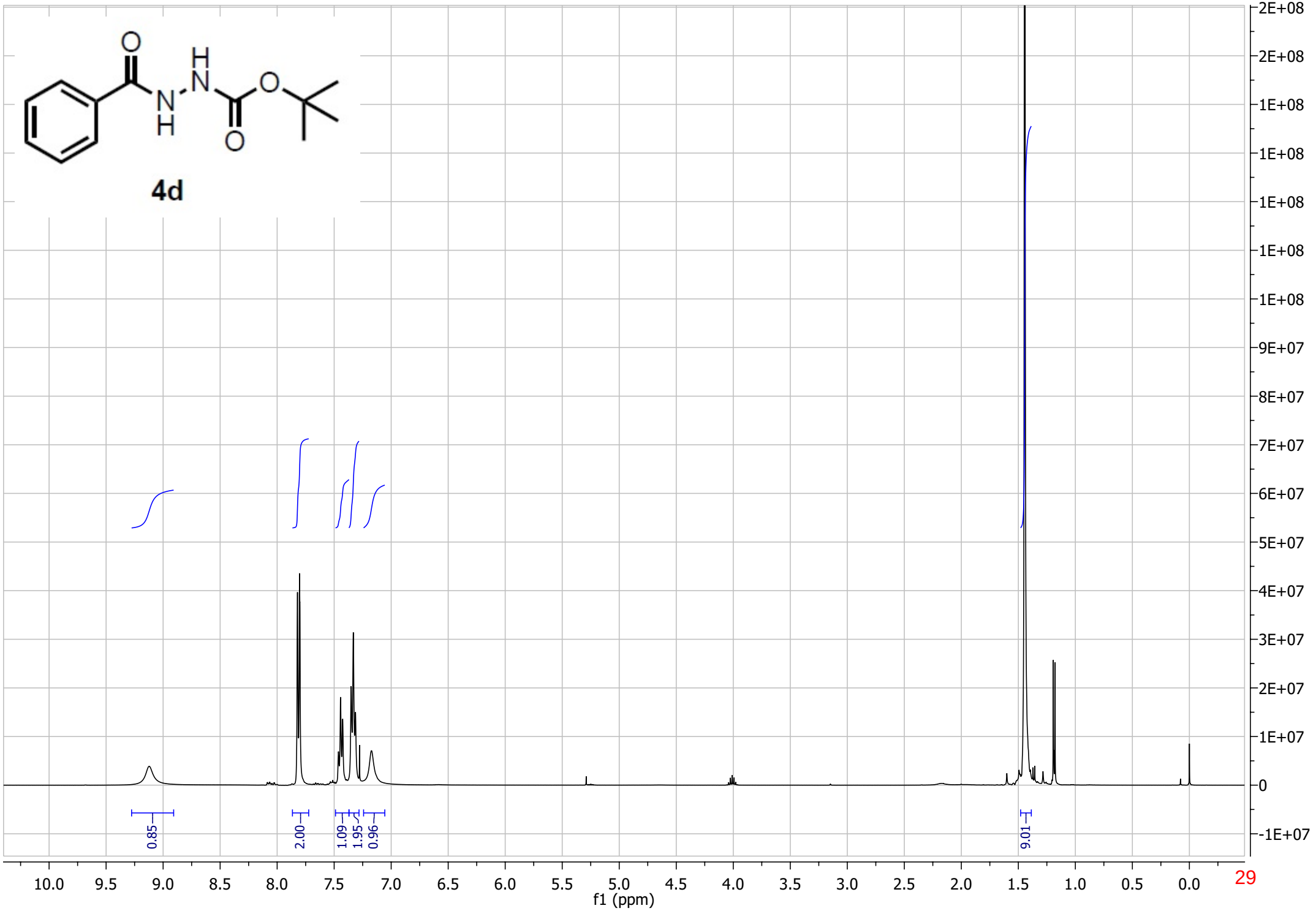

167.51
134.79
131.09
128.33
126.81
77.32
77.00
76.68
40.00
29.23
29.05
22.27
13.85
O
N
H
4b
4E+08
4E+08
4E+08
3E+08
3E+08
3E+08
3E+08
3E+08
2E+08
2E+08
2E+08
2E+08
2E+08
2E+08
1E+08
1E+08
1E+08
8E+07
6E+07
4E+07
2E+07
0
-2E+07
210 200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10 0 -10
f1 (ppm)

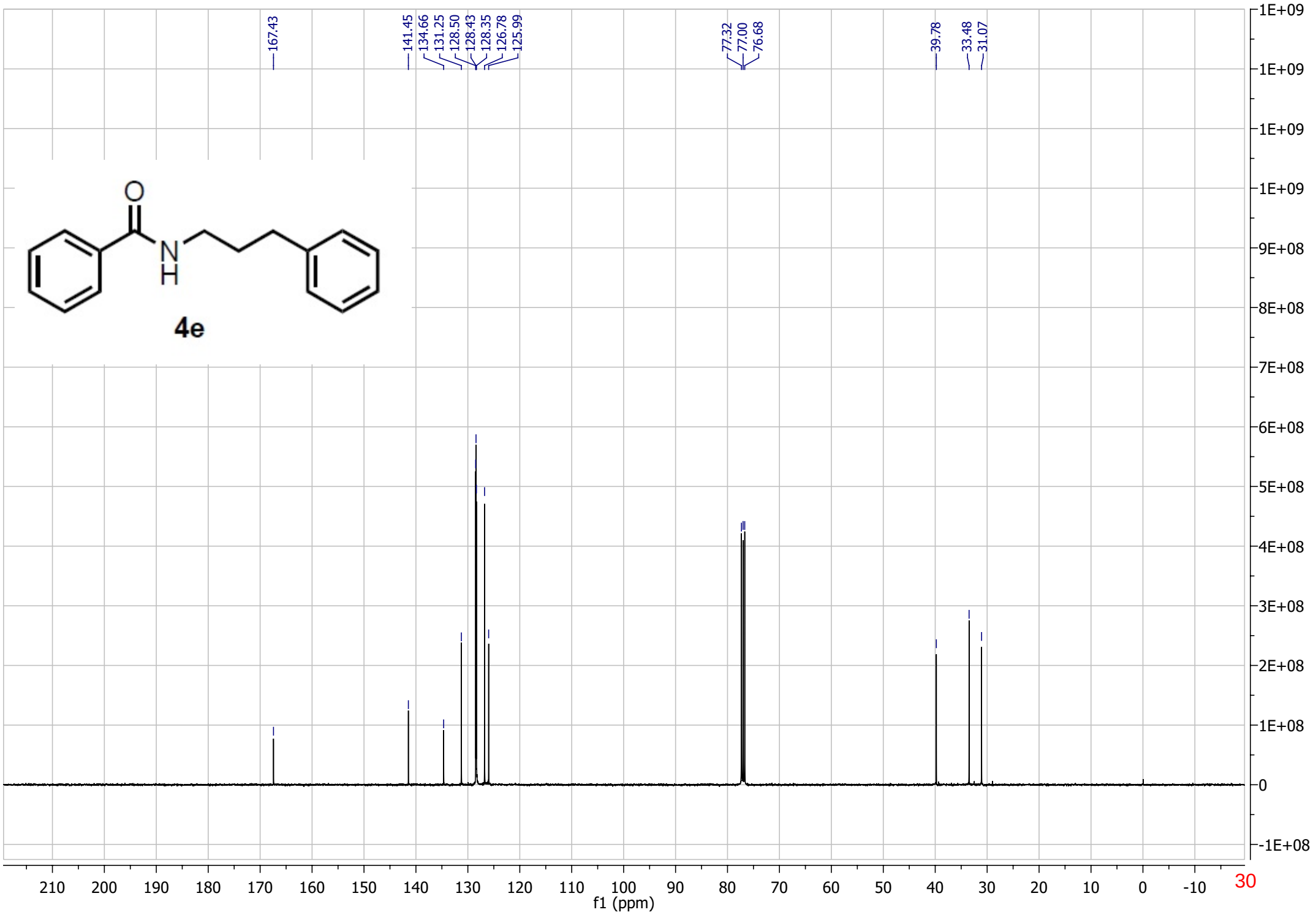

O
N
H
4b
2.00
0.99
2.04
1.00
2.09
2.09
4.24
3.16
f1 (ppm)
25

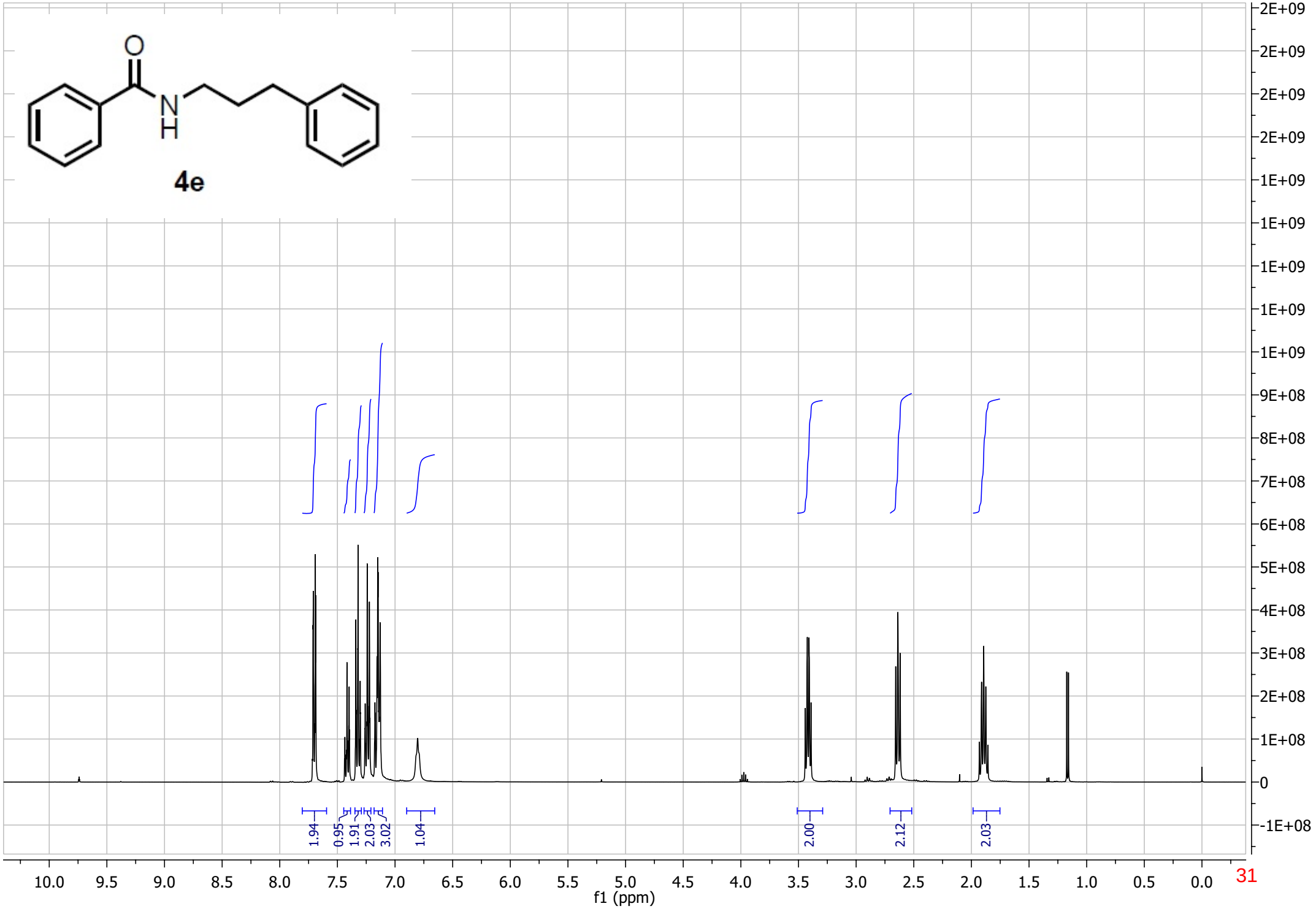

167.24
151.24
142.14
134.12
131.47
128.44
126.98
110.42
107.52
77.32
77.00
76.68
36.91
O
N
H
O
4c
8E+08
8E+08
8E+08
7E+08
6E+08
6E+08
6E+08
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
-5E+07
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
f1 (ppm)

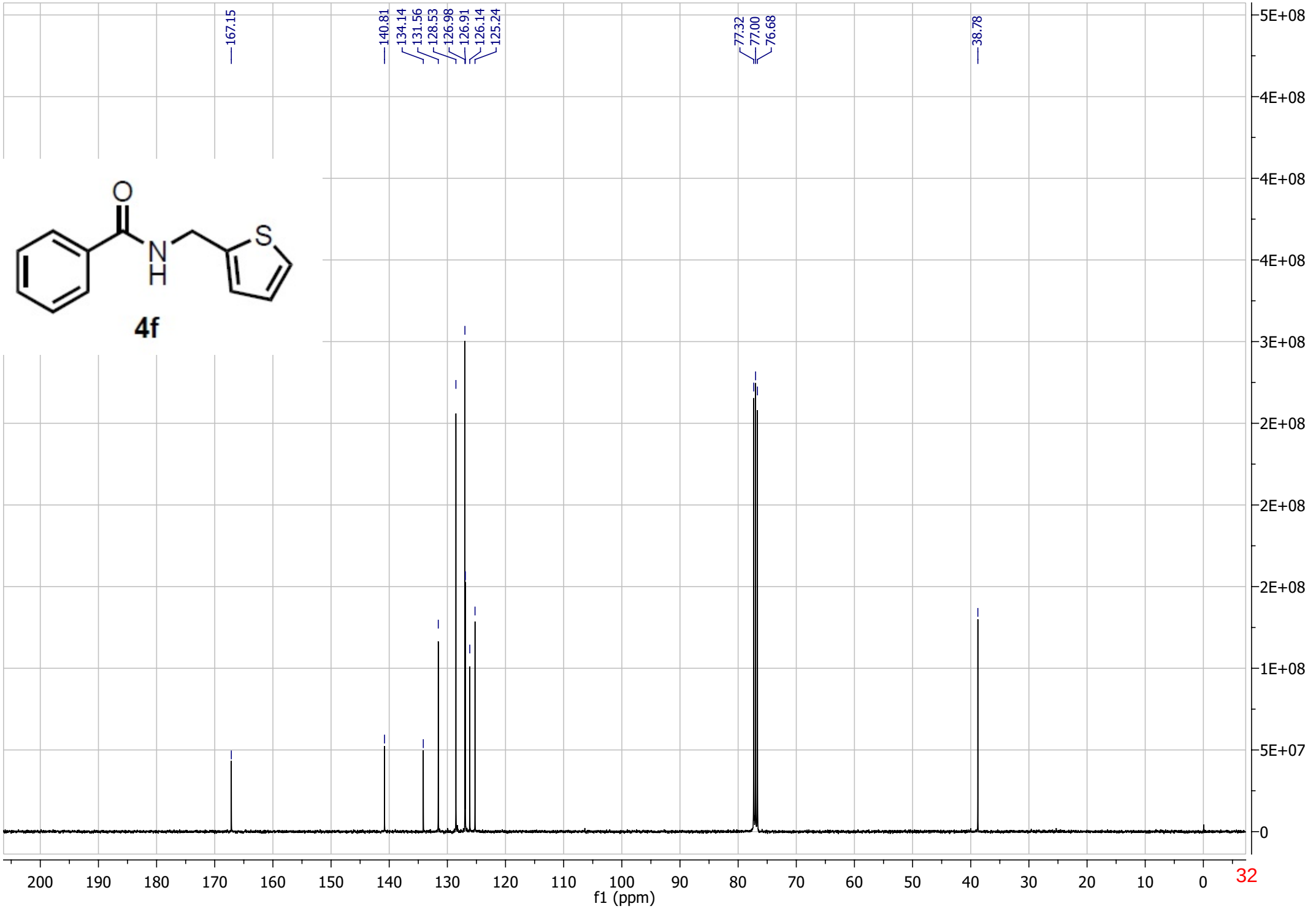

O
N
H
O
4c
2.10
1.09
3.06
1.01
2.00
2.12
8E+08
7E+08
6E+08
6E+08
6E+08
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
-5E+07
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
f1 (ppm)

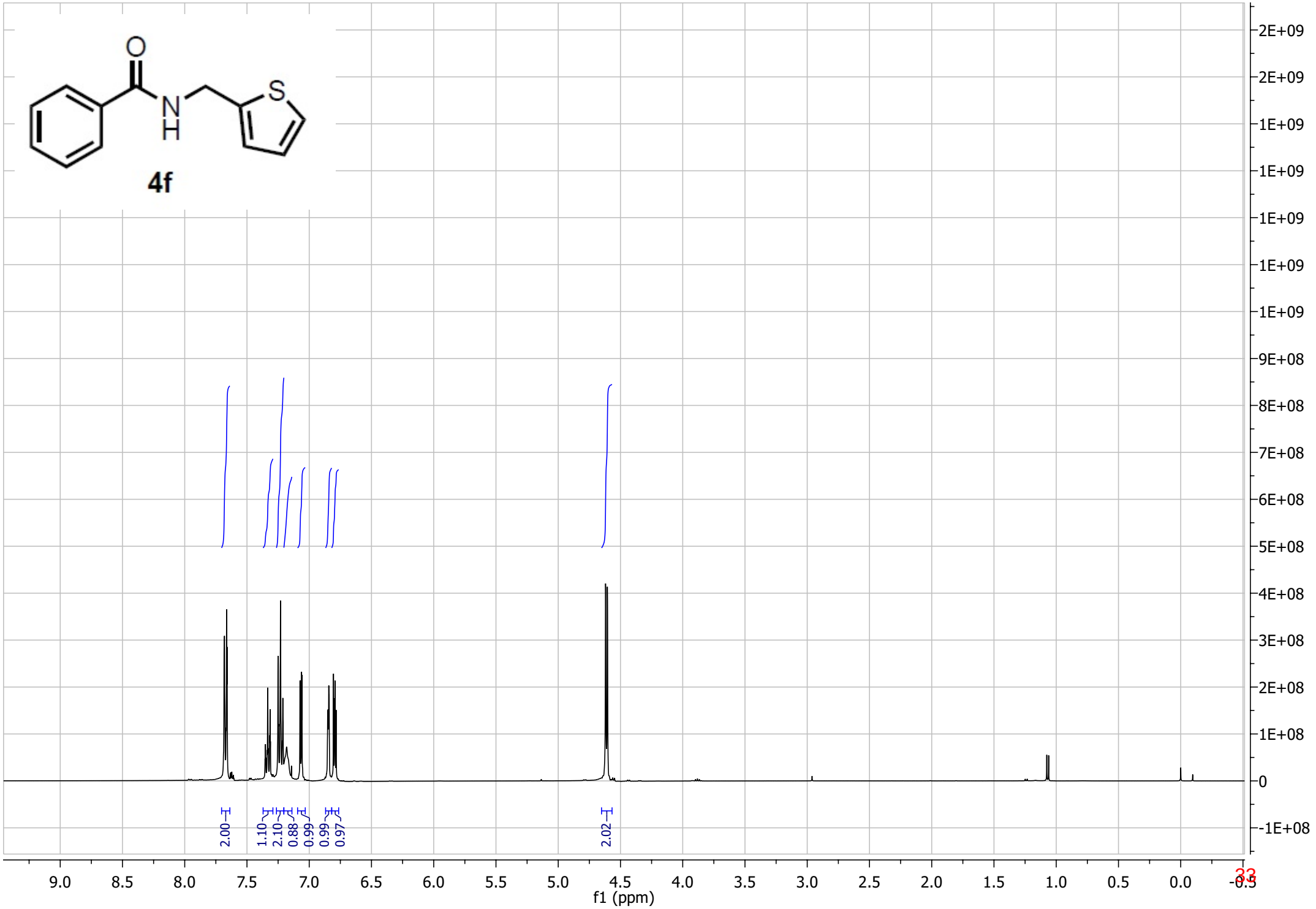

166.93
155.98
132.06
128.50
127.32
81.86
77.32
77.00
76.68
28.23
28.09
4d
1E+09
1E+09
1E+09
9E+08
8E+08
7E+08
6E+08
5E+08
4E+08
3E+08
2E+08
1E+08
0
210 200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10 0 -10
f1 (ppm)

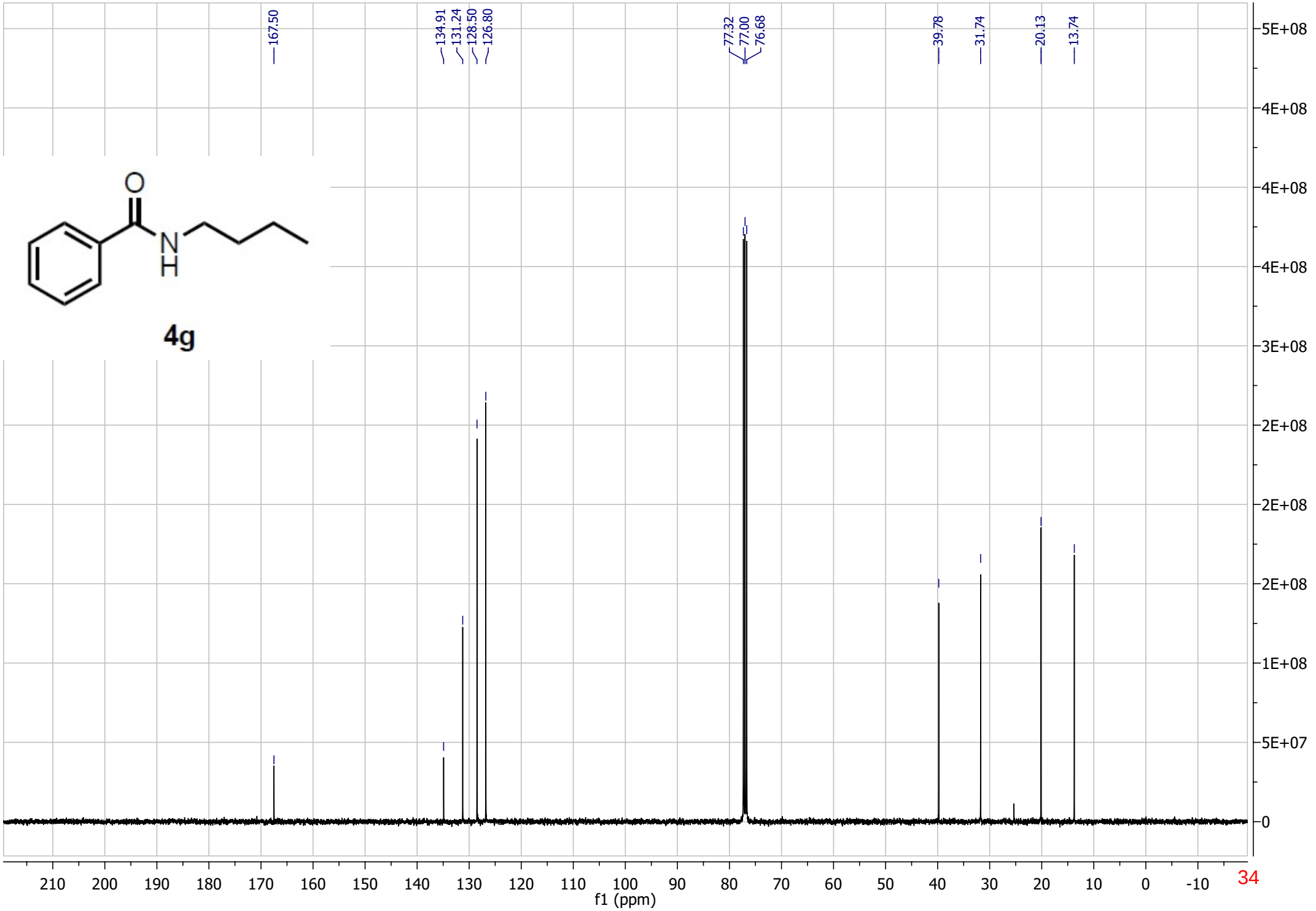

4d
0.85
2.00
1.09
1.95
0.96
10.16
f1 (ppm)
29

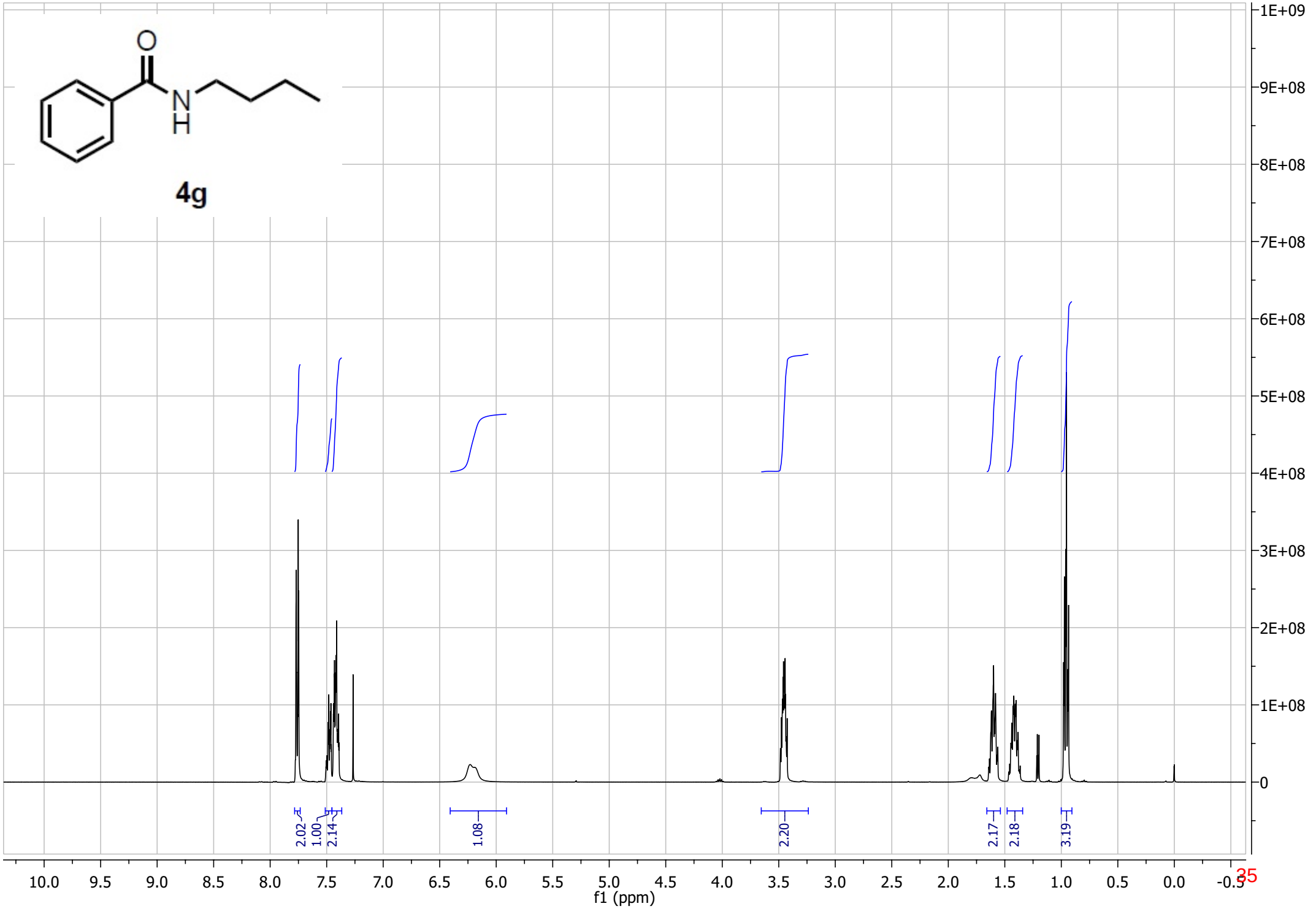

167.43
141.45
134.66
131.25
128.50
128.43
128.35
126.78
125.99
77.32
77.00
76.68
39.78
33.48
31.07
O
N
H
4e
1E+09
1E+09
1E+09
1E+09
9E+08
8E+08
7E+08
6E+08
5E+08
4E+08
3E+08
2E+08
1E+08
0
-1E+08
210 200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10 0 -10
f1 (ppm)

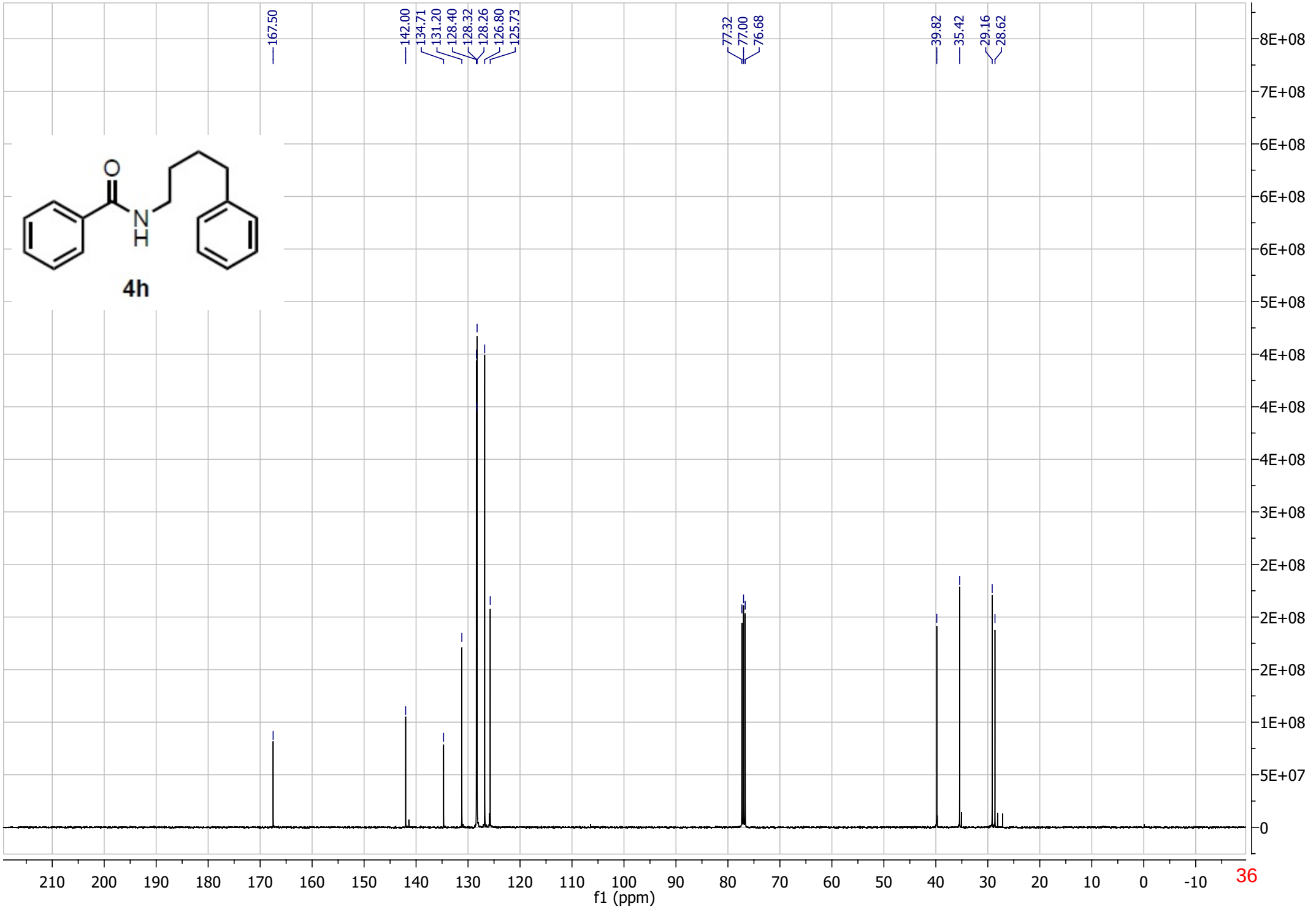

O
N
H
4e
1.94
0.95
1.91
2.03
3.02
1.04
2.00
2.12
2.03
f1 (ppm)
31

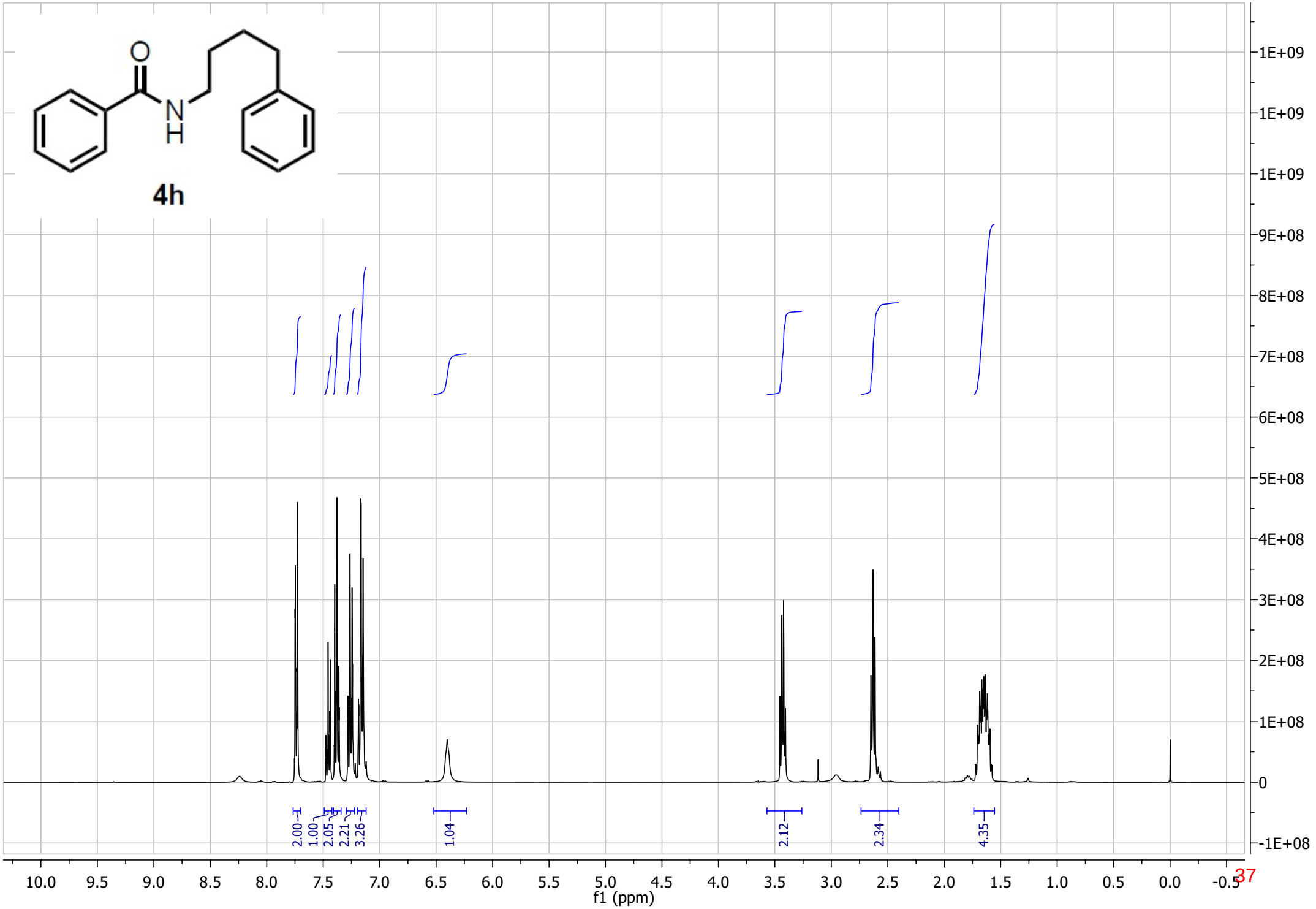

167.15
140.81
134.14
131.56
128.53
126.98
126.91
126.14
125.24
77.32
77.00
76.68
38.78
O
N
H
S
4f
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
f1 (ppm)

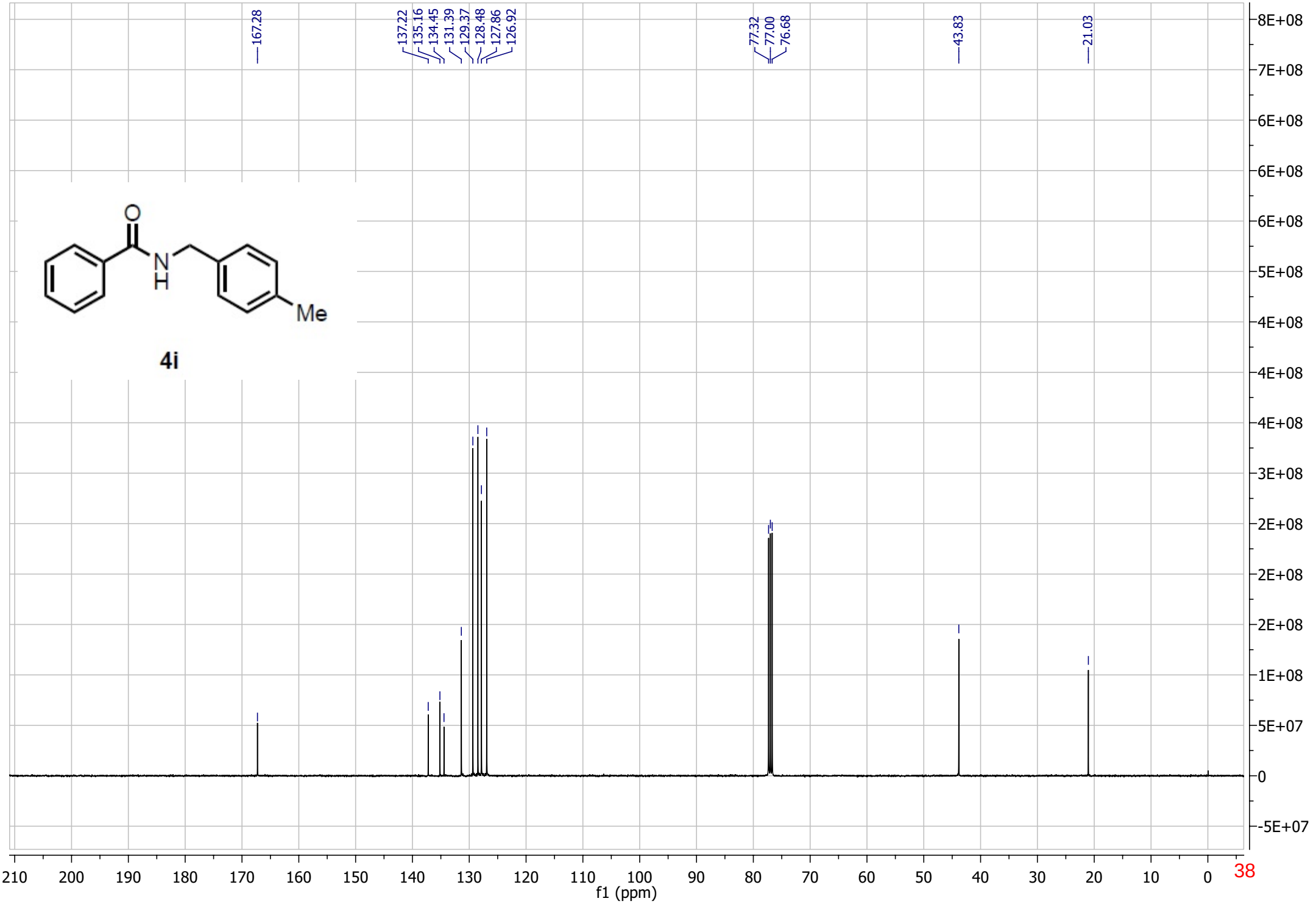

O
N
H
S
4f
2.00
1.10
2.10
0.88
0.99
0.99
0.97
2.02
2E+09
2E+09
1E+09
1E+09
1E+09
1E+09
1E+09
1E+09
9E+08
8E+08
7E+08
6E+08
5E+08
4E+08
3E+08
2E+08
1E+08
0
-1E+08
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
f1 (ppm)

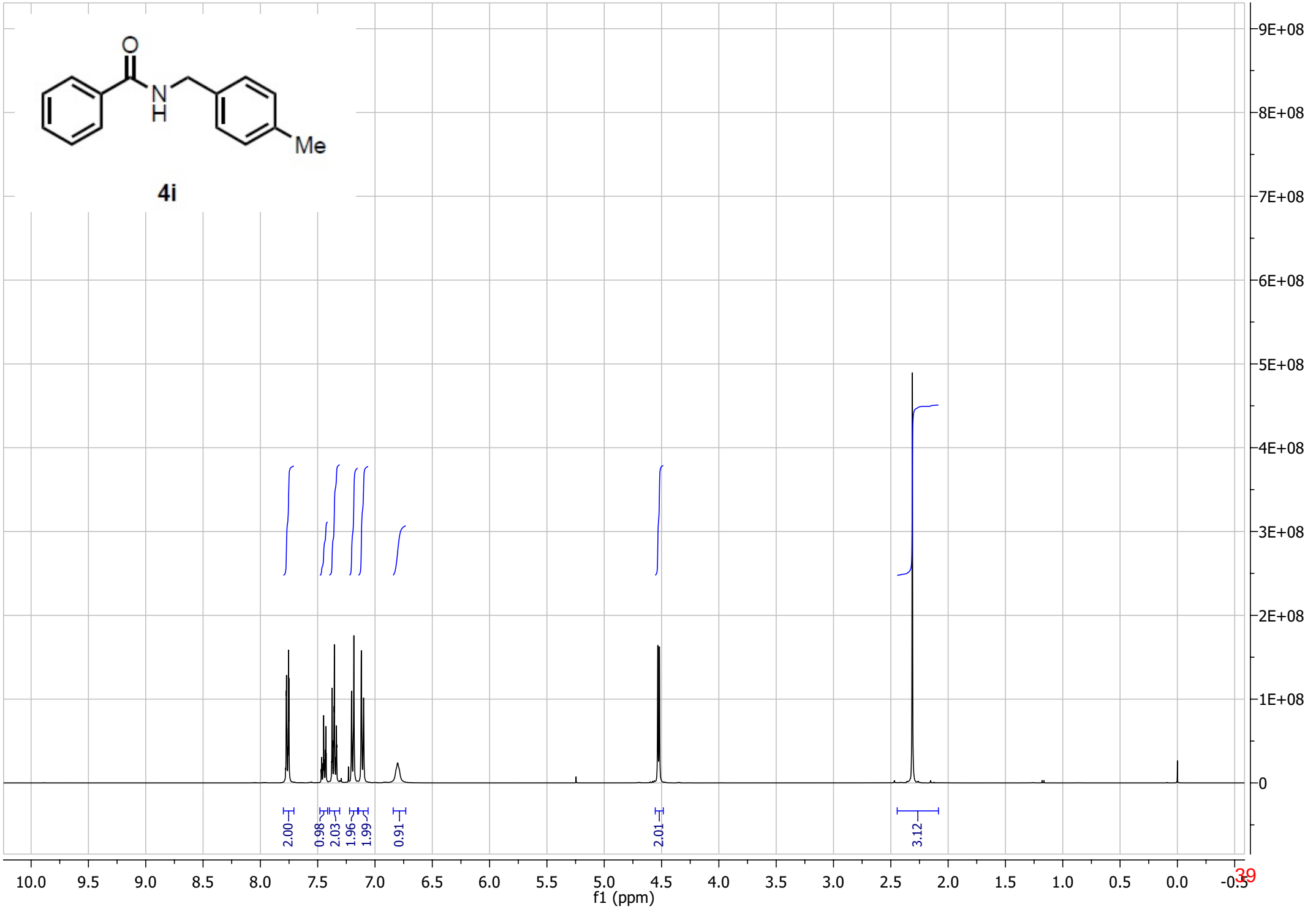

167.50
134.91
131.24
128.50
126.80
77.32
77.00
76.68
39.78
31.74
20.13
13.74
O
N
H
4g
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
210 200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10 0 -10
f1 (ppm)

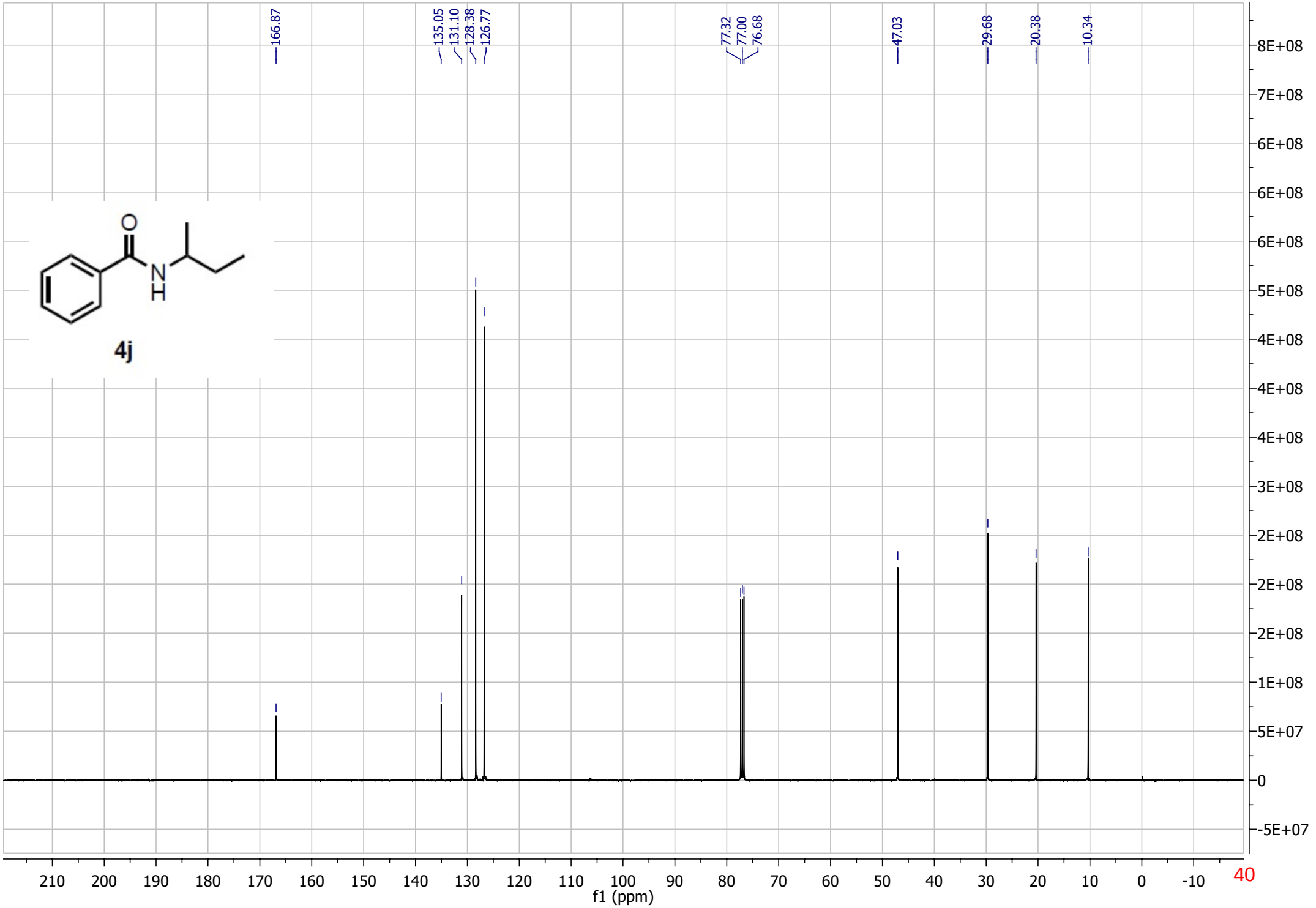

O
N
H
4g
1E+09
9E+08
8E+08
7E+08
6E+08
5E+08
4E+08
3E+08
2E+08
1E+08
0
2.02
1.00
2.14
1.08
2.20
2.17
2.18
3.19
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
f1 (ppm)

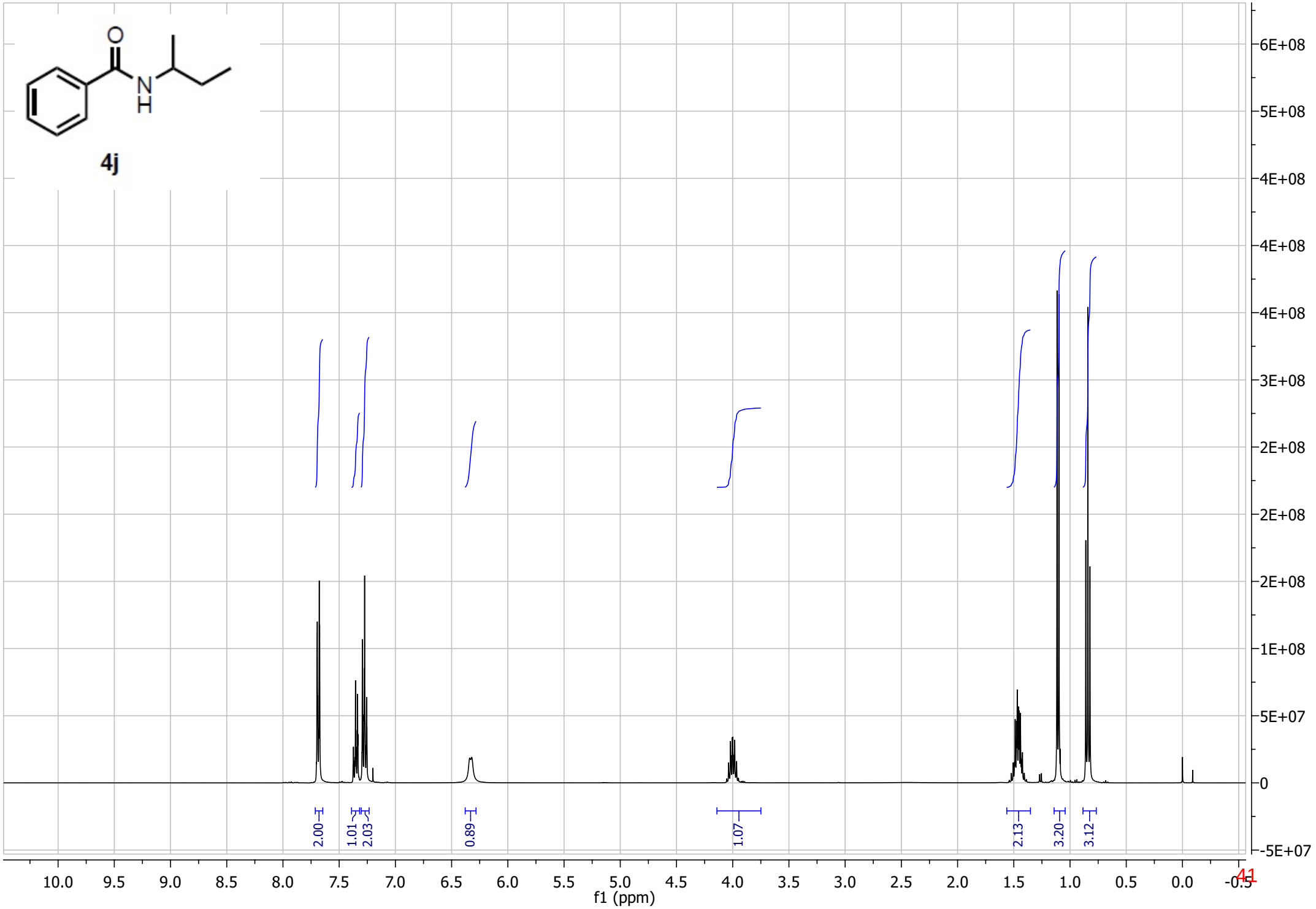

167.50
142.00
134.71
131.20
128.40
128.32
128.26
126.80
125.73
77.32
77.00
76.68
39.82
35.42
29.16
28.62
O
N
H
4h
8E+08
7E+08
6E+08
6E+08
6E+08
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
210 200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10 0 -10
f1 (ppm)


O
N
H
4h
1E+09
1E+09
1E+09
9E+08
8E+08
7E+08
6E+08
5E+08
4E+08
3E+08
2E+08
1E+08
0
-1E+08
2.00
1.00
2.05
2.21
3.26
1.04
2.12
2.34
4.35
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
f1 (ppm)


167.28
137.22
135.16
134.45
131.39
129.37
128.48
127.86
126.92
77.32
77.00
76.68
43.83
21.03
O
N
H
Me
4i
8E+08
7E+08
6E+08
6E+08
6E+08
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
-5E+07
210
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
f1 (ppm)


O
N
H
Me
4i
9E+08
8E+08
7E+08
6E+08
5E+08
4E+08
3E+08
2E+08
1E+08
0
2.00
0.98
2.03
1.96
1.99
0.91
2.01
3.12
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
f1 (ppm)


166.87
135.05
131.10
128.38
126.77
77.32
77.00
76.68
47.03
29.68
20.38
10.34
O
N
H
4j
8E+08
7E+08
6E+08
6E+08
6E+08
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
-5E+07
210 200 190 180 170 160 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10 0 -10
f1 (ppm)


O
N
H
4j
6E+08
5E+08
4E+08
4E+08
4E+08
3E+08
2E+08
2E+08
2E+08
1E+08
5E+07
0
-5E+07
2.00
1.01
2.03
0.89
1.07
2.13
3.20
3.12
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
f1 (ppm)