Supporting Information

Exotic Inverse Kinetic Isotopic Effect in the Thermal Decomposition of Levitated Aluminum Iodate Hexahydrate Particles

Grace L. Rizzo,[1] Souvick Biswas,[1] Ivan Antonov,[1] Kelsea K. Miller,[2] Michelle L. Pantoya,[*2] Ralf I. Kaiser[*1]

[1]Department of Chemistry, University of Hawai'i at Manoa, Honolulu, HI 96822, USA

[2] Department of Mechanical Engineering, Texas Tech University, Lubbock, TX 79409-1021, USA

*E-mail: ralfk@hawaii.edu

*E-mail: michelle.pantoya@ttu.edu

Methods

Chemicals and Gases: Aluminum iodate hexahydrate (AIH) and deuterated – aluminum iodate hexahydrate (AID) particles were synthesized in house as described in detail by Miller et al.[1] Briefly, particles were synthesized using an acid-base precipitation reaction that can be summarized in three steps. Synthesis began by dissolving iodine pentoxide (I_2O_5) powder (Sigma Aldrich) in water (i.e., either H_2O or D_2O) at a 1:1 weight ratio. Following preparation of this highly acidic solution, aluminum hydroxide ($Al(OH)_3$) powder (Sigma Aldrich) was introduced and heated mildly (90°C) to facilitate dissolution. For large crystal formation, the solution was removed from heat and slowly evaporated under ambient conditions precipitating the formation of mm-scale particles.[1] A molecule can be described as an aluminum atom surrounded by a six-member hydroxyl ring that is further surrounded by iodates: $[Al(H_2O)_6](IO_3)_3(HIO_3)_2$ creating (AIH) or its deuterated counterpart (AID). The largest particle that could be steadily levitated in the present apparatus was approximately 2.5 mm in size. The argon (99.9999%, Ar) used to fill the process chamber was obtained from Airgas.

Ultrasonic Levitator Apparatus: The experiments were carried out utilizing an ultrasonic acoustic levitator.[2-5] Ultrasonic sound waves were generated from a piezoelectric transducer oscillating at 58 kHz (Figure S9). A standing wave is generated through multiple reflections between the transducer and concave reflector. The distance between the transducer and reflector is adjusted by a micrometer manipulator to an integral number of half wavelengths which allows for resonance conditions to be maintained following any changes that occur in gas composition, temperature, or pressure. A typical distance between the front plate and reflector is selected to 2.5 times the wavelength (14.8 mm) resulting in five pressure nodes. The amplitude of the oscillations can be optimized through an adjustment of the radio frequency (RF) power between 0.7 and 5.0 W and monitoring on an oscilloscope. The levitator is housed within a pressure compatible stainless-steel process chamber.[2] This enables the AIH and AID particles to be studied in an inert atmosphere and elevated higher pressure of argon inert gas. To stabilize the particles in the third pressure node of the standing wave, experiments were carried out at a pressure of 1200 Torr measured by a MKS 626B series capacitance manometer

Solid Particle Sampling: Solid samples are levitated slightly below the pressure minima of the ultrasonic standing wave.[2,3] This is feasible since the acoustic radiation pressure from the sound

waves counteracts the gravitational force. The horizontal restoring force centers the particle on the axis of the levitator. In the single axis levitator, this horizontal force is one order of magnitude smaller than the axial force. The particles are introduced to the central pressure node via a magnetically coupled wobble stick attached to a side port of the process chamber. A one centimeter by one centimeter stainless-steel wire mesh is attached to the end of the wobble stick shaping an acoustically transparent spatula inside of the process chamber, which holds the AIH or AID sample.[2,3]

Pyrolysis: To initiate decomposition of the levitated particle, a 40 W carbon dioxide laser emitting at 10.6 μm (Synrad, Inc., model FSV40KFD) was used.[2,3] The output power of the laser is adjustable between 1 and 40 W by changing the duty cycle of the discharge by externally triggering the laser using a pulse delay generator (Quantum Composers, 9518 plus). Pending on the desired temperature, the output power of the laser for each experiment ranged between 1-20 W. This output of the laser passes a zinc selenide window and was introduced to the center of the levitator by a planar copper mirror. The diameter of the laser beam at the trap center was optimized to 10 mm to allow a uniform heating of the levitated particle. A higher temperature range could be accessed by focusing the laser beam to a diameter of 0.2 mm onto the particle. This was achieved by using an eight-fold beam expander followed by a parabolic copper mirror with a focal length of 300 mm. The exiting beam from the laser had a diameter of 2.5 mm with a beam divergence angle of less than 7.0 mR. Note that at higher temperatures, the particle became increasingly unstable due to change in the density of the gas and change in the speed of sound. To counterbalance this instability, an arbitrary waveform generator (Keysight, model 33210A) was exploited to modulate the carrier wave through a sine wave with a frequency of 10-100 Hz and an amplitude range of 100-300 mV thus enhancing the lateral stability.

Raman Spectroscopy: To trace the chemical modifications of the trapped particle, Raman spectroscopy was exploited. The Raman transitions were excited by a 532 nm output of a diode-pumped, Q-switched Nd:YAG laser (CrystaLaser, model QL532-1W) at a repetition rate of 1 kHz. This laser operated with a beam diameter of 0.35 mm and a divergence angle of 3.8 mR. This produced an average power output of 200 mW and a pulse width of 13.5 ns. After being reflected from a 45° mirror (Edmund Optics, model NT45-991, >99% reflectance) and a 45° dichroic beamsplitter (Semrock, RazorEdge, model LPD01-532RU-25×36×2.0), the laser beam entered the

process chamber through an antireflection coated window (Figure S10). A plano-convex lens with a focal length of 60 mm was exploited to focus the laser beam onto the particle. The lens then collimates the Raman-shifted photons backscattered from the levitated particles. The beam splitter reflects the incident 532 nm laser beam, but transmits the longer Raman-shifted wavelengths. These passed through a 532 nm RazorEdge ultrasteep long-pass edge filter (Semrock, model LP03-532RE-25), which further decreases the transmitted 532 nm laser light. A 50 mm f/1.8 camera (Nikon, Nikkor 2137) lens focuses the light through the 100 μm entrance slit of the spectrograph; the resolution of the spectrometer is 9 cm^{-1}.

The light then is introduced into a Holospec f/1.8 imaging spectrometer (Kaiser Optical Systems, model 2004500-501), where the beam is collimated by a lens toward two overlaid holographic transmission gratings (Kaiser Optical Systems, model Holoplex HPG-532). Each grating separates the Raman-shift wavenumbers into low and high regions (2400 to 100 cm^{-1}, 4000 to 2200 cm^{-1}). These gratings disperse the light onto spatially distinct halves of a Peltier-cooled charge-coupled device (CCD) detector (Princeton Instruments, PI-MAX2). The CCD detector is composed of 1024 × 256 pixels each having a pixel size of 26 μm. In order to reduce the fluorescence background, it is imperative to conduct pulsed Raman experiments with the pulsed laser and gated detector system described above. The time delay between the laser pulse, opening gate to collect signal, and the period for which the gate is open are optimized to allow for an early detection of the Raman signal, while rejecting the major portion of the 'delayed' fluorescent background. Here, the CCD is kept at a pulse width of 50 ns per pulse and a gate delay of 487 ns with 1000 gates per exposure. The detector operated at a 1 kHz repetition rate. Both the excitation laser and the detector are externally triggered through a pulse delay generator (Quantum Composers, 9518 plus) (Figure S11).

Optical and Infrared Videos: Optical videos were obtained by a Phantom Miro 3a10 camera. This is equipped with a Navitar Zoom 6000 modular lens system. In tandem with the optical videos, thermal imaging videos were collected by a FLIR A6703sc IR camera. Both cameras were operated at a frame rate of 30 Hz. Although these cameras could be operated at higher repetition rates, 30 Hz was used for optimized synchronization and allowing for longer movies to be recorded. Temporal temperature profiles were collected from the maximum temperature readings from the IR camera.

Table S1a. Vibrational mode assignments for the observed peaks in the Raman spectra of AIH at 293 K.

Peaks	Frequency (cm^{-1}) (this work)	Intensity (this work)	Frequency (cm^{-1}) (Literature[16-22])	Literature Intensity	Carrier	Assignment	Description
1	3137	m	3162	w	$[Al(H_2O)_6]^{3+}$	$\nu_3(H_2O)$	O-H stretching
2	2946	m	3045, 3000 3045	m vw	$[Al(H_2O)_6]^{3+}$ HIO_3	$\nu_1(H_2O)$ ν_1	O-H stretching
3	834	w	839	w	HIO_3	$\nu_8 + \nu_L$	Combination
4	816	w	817	vw	IO_3^-	combination	Combination
5	790	s	808, 806, 789	s	IO_3^-	ν_3	IO_2 antisymmetric stretching
6	777	s	780	vs	HIO_3	ν_8	IO_2 antisymmetric stretching
7	752	m	758, 755, 754, 753, 743	w	IO_3^-	ν_1	IO_2 symmetric stretching
8	713	m	713	vs	HIO_3	ν_3	IO_2 symmetric stretching
9	631	m	631	m	HIO_3	ν_4	IO stretching
10	618	w	622	w	$[Al(H_2O)_6]^{3+}$	ν_L	Lattice modes

Table S1b. Vibrational mode assignments for the observed peaks in the Raman spectra of AID at 293 K.

Peaks	Frequency (cm^{-1}) (this work)	Intensity (this work)	Frequency (cm^{-1}) (Literature[16-22])	Literature Intensity	Carrier	Assignment	Description
1	3084	w	3045	vw	HIO_3	ν_1	O-H stretching
2	2427	s	2416	m	$[Al(D_2O)_6]^{3+}$	2 x $\delta(OD_2)$	O-D bending
3	841	w	839	w	HIO_3	$\nu_8 + \nu_L$	Combination
4	817	w	817	vw	IO_3^-	combination	Combination
5	810	w	789	w	IO_3^-	ν_3	IO_2 antisymmetric stretching
6	772	vs	780	vs	HIO_3	ν_8	IO_2 antisymmetric stretching
7	748	m	758, 755, 754, 753, 743	w	IO_3^-	ν_1	IO_2 symmetric stretching
8	699	w	713	vs	HIO_3	ν_3	IO_2 symmetric stretching
9	661	m	631	m	HIO_3	ν_4	IO stretching
10	607	w	617, 593	w	$[Al(D_2O)_6]^{3+}$	ν_L	Lattice modes

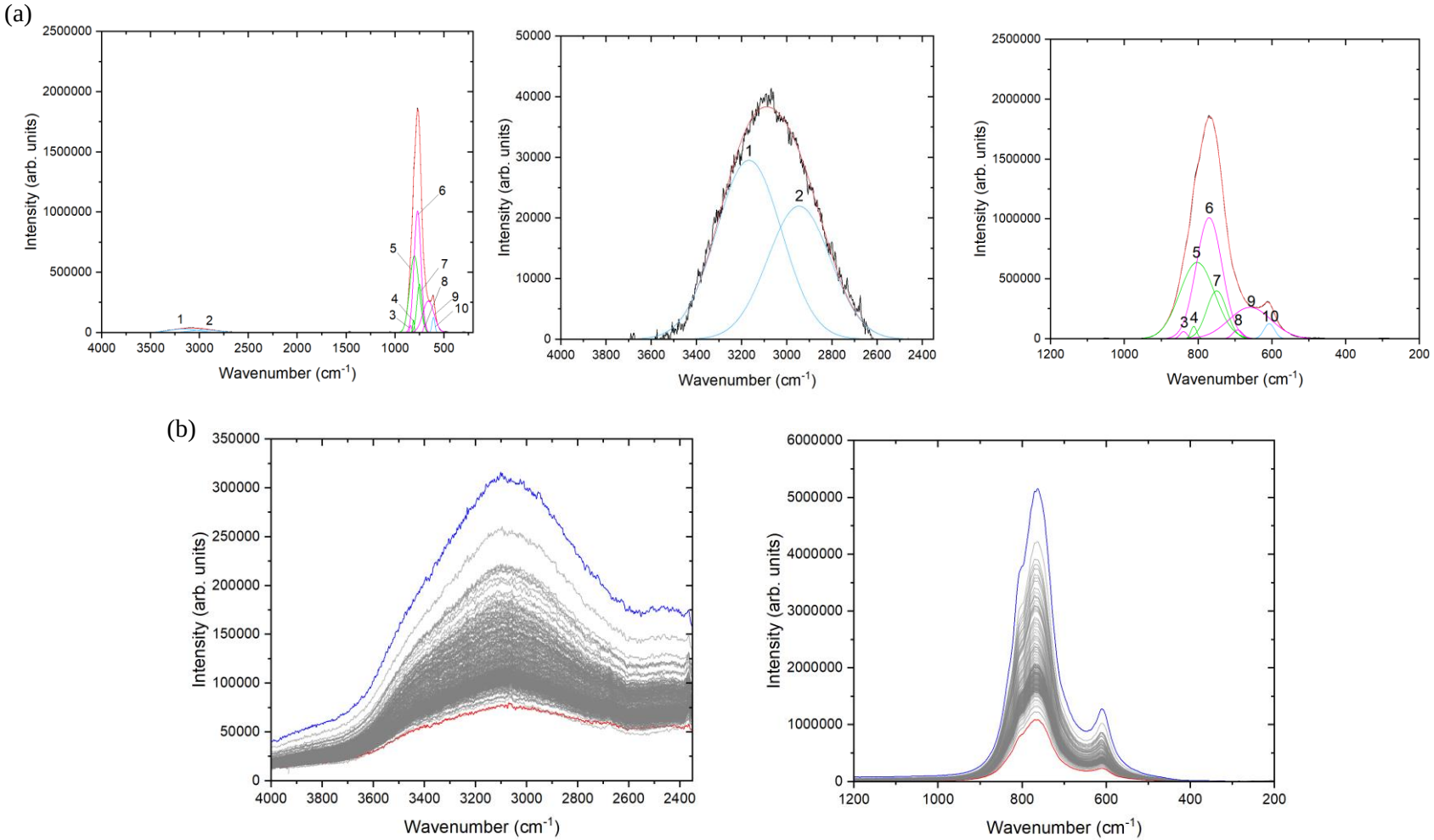

Figure S1. Raman Spectra of AIH at 375 K where (a) shows the full deconvoluted spectrum (left) with detailed views of the high (middle) and low (right) region peaks. (b) represents the high and low regions of the raw data. The spectrum colored in blue corresponds to the start of the constant heating at 375 K. The spectrum colored in red is after heating for two hours.

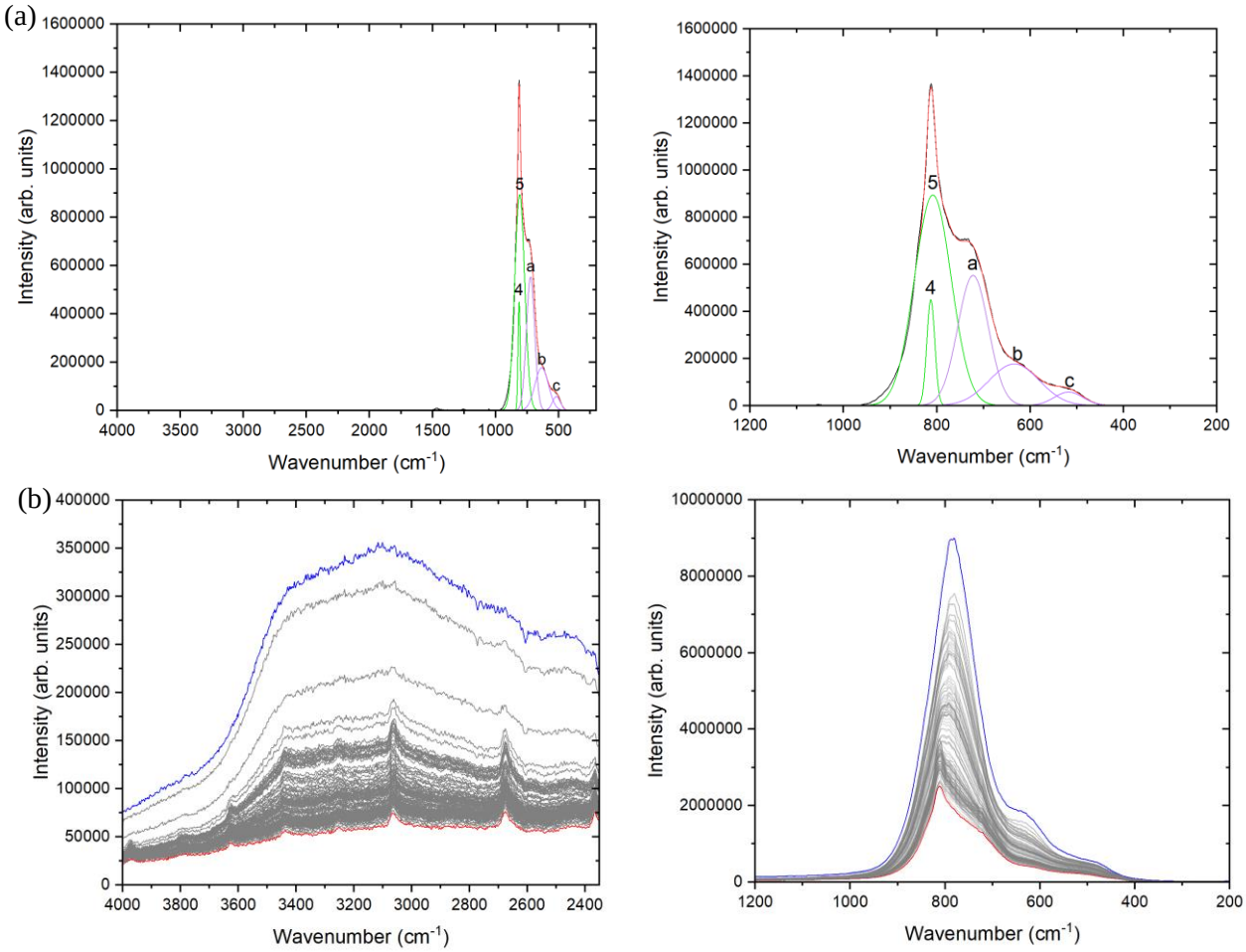

Figure S2. Raman Spectra of AIH at 480 K where (a) shows the full deconvoluted spectrum (left) with a detailed view of the low (right) region peaks. New deconvoluted peaks are assigned to I_2O_5 (purple). See table 3 for new peak assignments. (b) represents the high and low regions of the raw data. The spectrum colored in blue corresponds to the start of the constant heating at 480 K. The spectrum colored in red is after heating for an hour and fifteen minutes. Small, repeated peaks in high region are artificial peaks caused by more surface scattering as the particles surface changes becoming shinier and more reflective.

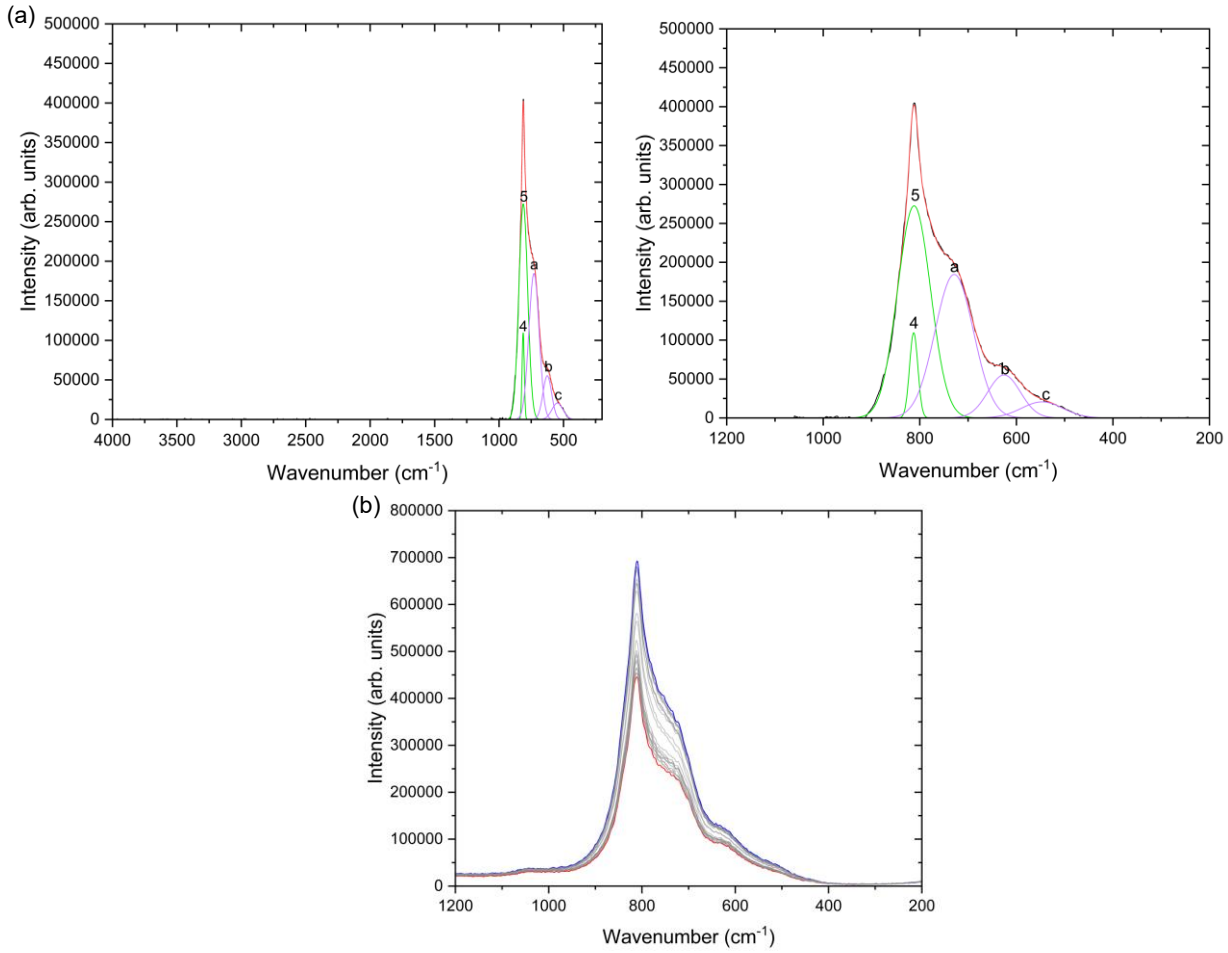

Figure S3. Raman Spectra of AIH at 580 K where (a) shows the full deconvoluted spectrum (left) with a detailed view of the low (right) region peaks. (b) represents the high and low regions of the raw data. The spectrum colored in blue corresponds to the start of the constant heating at 580 K. The spectrum colored in red is after heating for 10 minutes.

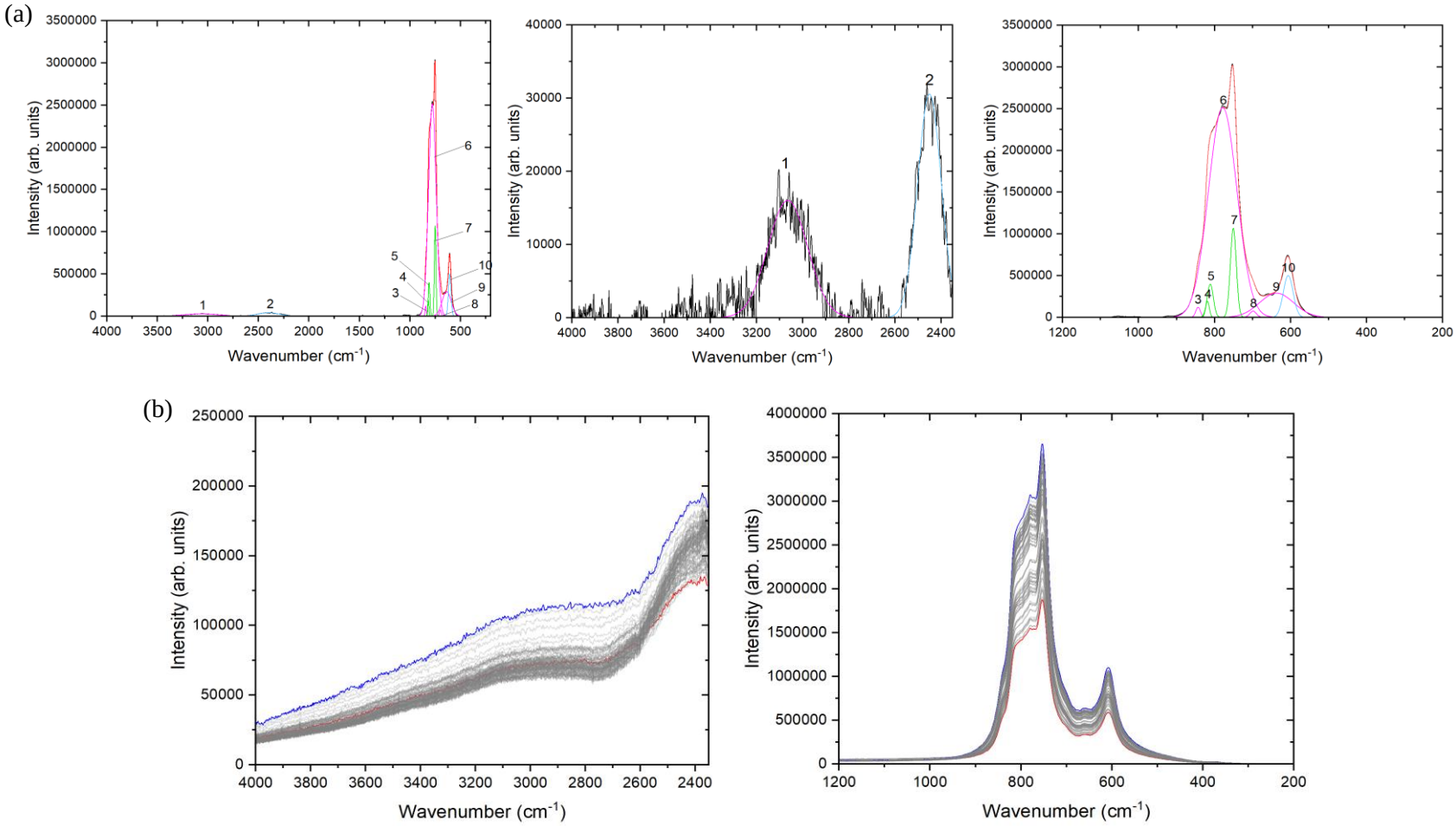

Figure S4. Raman Spectra of AID at 375 K where (a) shows the full deconvoluted spectrum (left) with detailed views of the high (middle) and low (right) region peaks. (b) represents the high and low regions of the raw data. The spectrum colored in blue corresponds to the start of the constant heating at 375 K. The spectrum colored in red is after heating for one hour.

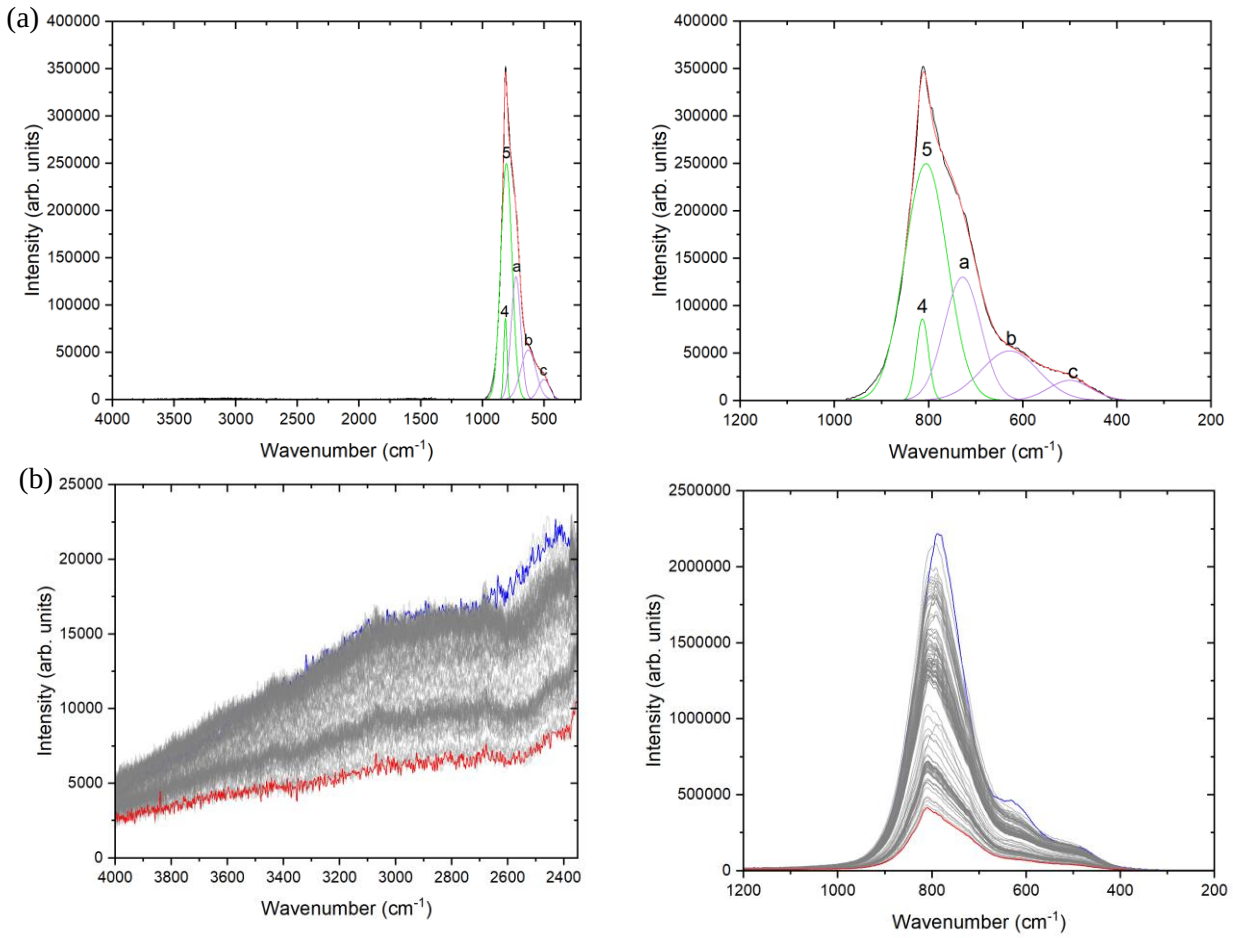

Figure S5. Raman Spectra of AID at 480 K where (a) shows the full deconvoluted spectrum (left) with a detailed view of the low (right) region peaks. New deconvoluted peaks are assigned to I_2O_5 (purple). See table 4 for new peak assignments. (b) represents the high and low regions of the raw data. The spectrum colored in blue corresponds to the start of the constant heating at 480 K. The spectrum colored in red is after heating for an hour and fifteen minutes.

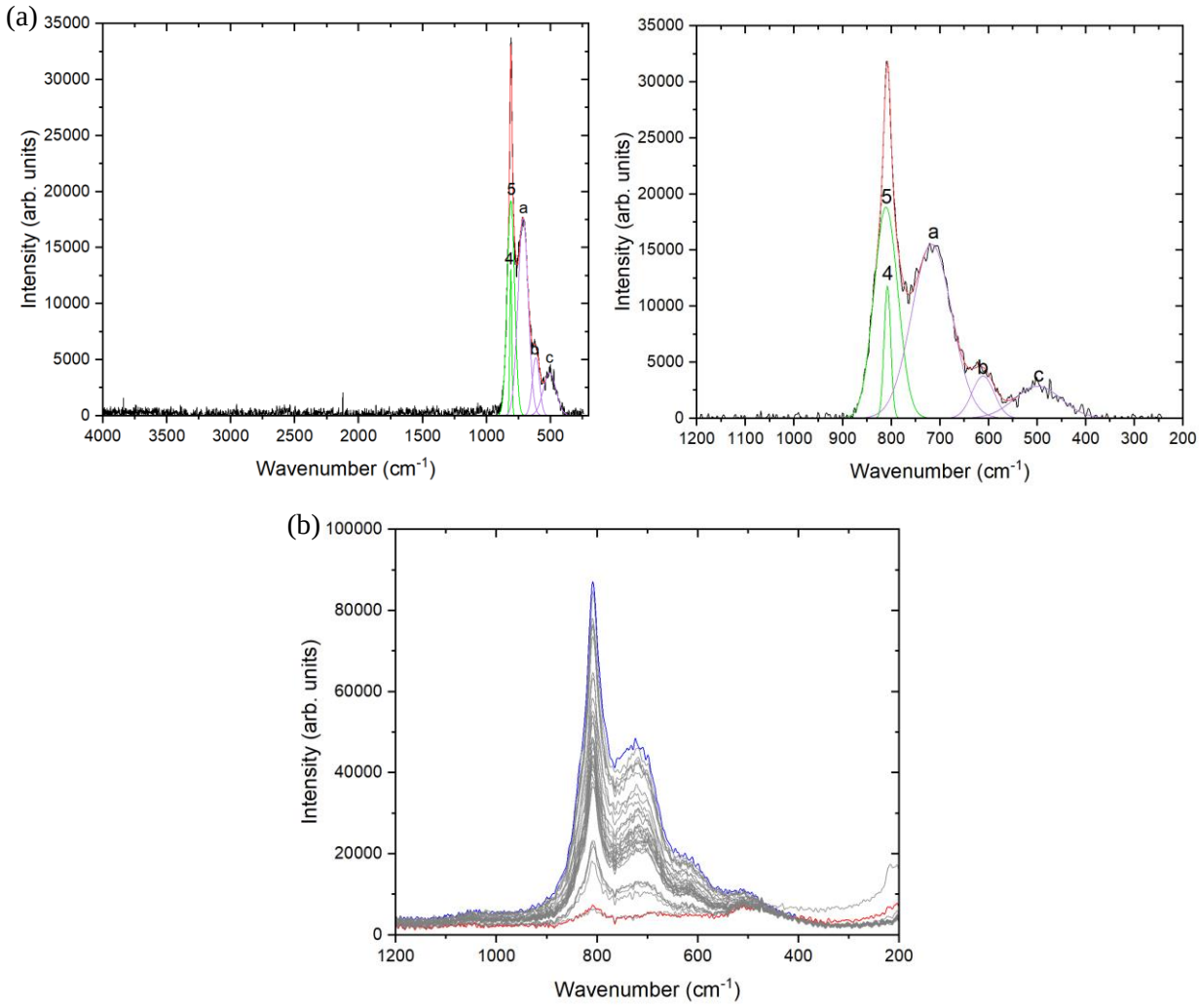

Figure S6. Raman Spectra of AID at 580 K where (a) shows the full deconvoluted spectrum (left) with a detailed view of the low (right) region peaks. (b) represents the low regions of the raw data. The spectrum colored in blue corresponds to the start of the constant heating at 580 K. The spectrum colored in red is after heating for 20 minutes.

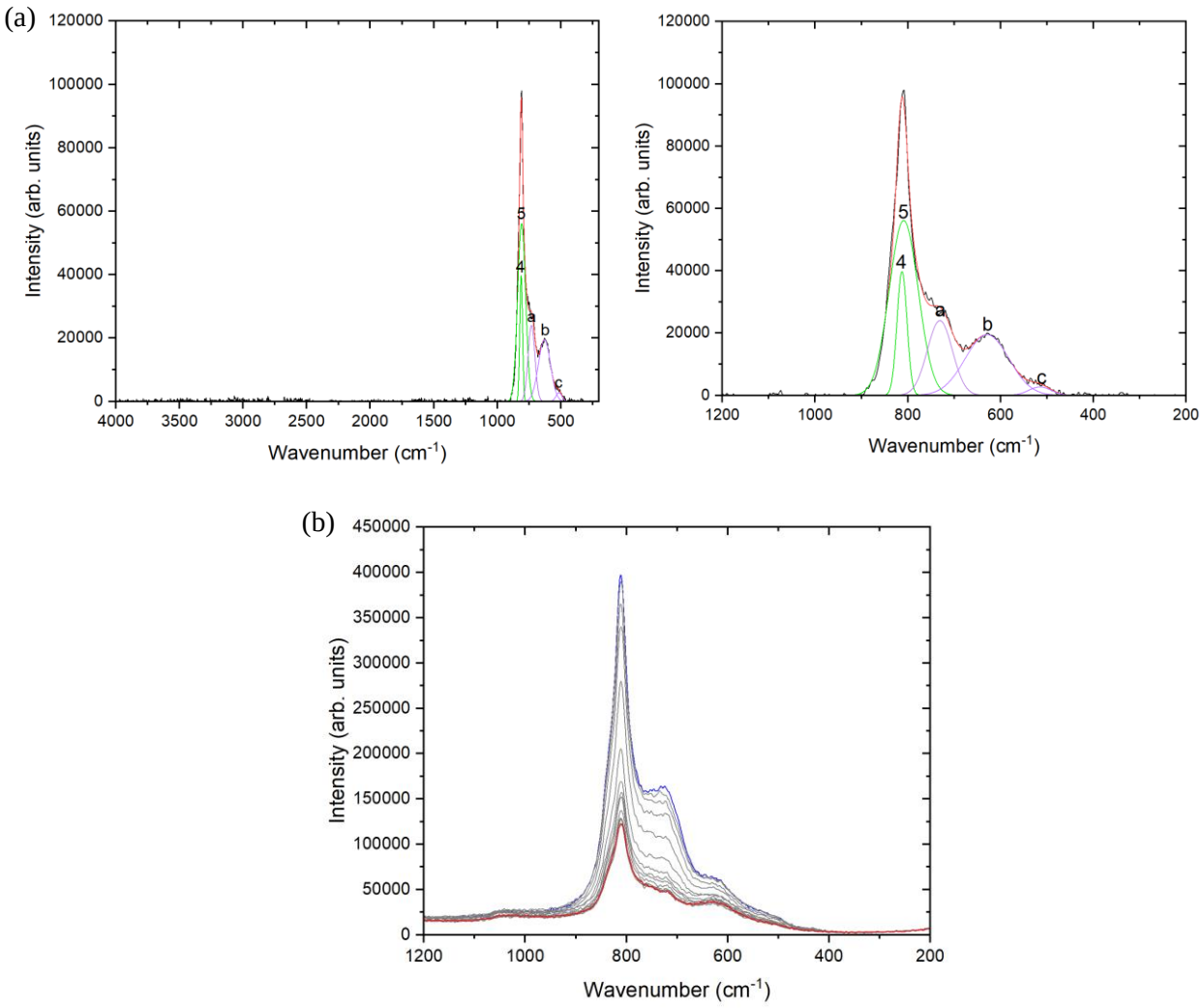

Figure S7. Raman Spectra of AIH at 600 K where (a) shows the full deconvoluted spectrum (left) with a detailed view of the low (right) region peaks. (b) represents the high and low regions of the raw data. The spectrum colored in blue corresponds to the start of the constant heating at 600 K. The spectrum colored in red is after heating for 10 minutes.

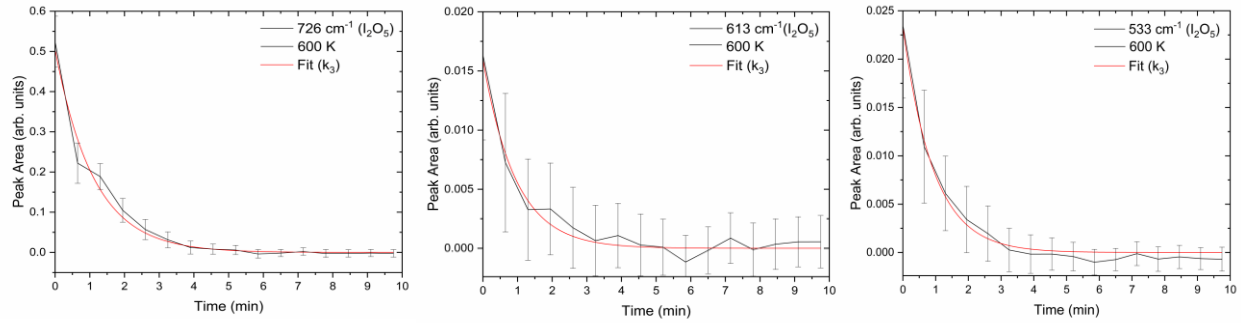

Figure S8. Fitted time traces of selected bands in the Raman spectra for AIH at 600K. The component/species are also mentioned in the parentheses of the corresponding band positions. See table S2 for rate constants.

Table S2: AIH rate constants for each step in the decomposition process. Selected peak wavenumbers are listed in parentheses. Here rate constants for 600 K are included.

Step	T, K	k (min^{-1})	Averaged k (min^{-1})
1 (3137 cm^{-1}) 1 (618 cm^{-1})	375 ± 5	0.046 ± 0.001 0.047 ± 0.001	0.047 ± 0.001
2 (777 cm^{-1}) (726 cm^{-1})	480 ± 5	0.020 ± 0.001 0.020 ± 0.001	0.020 ± 0.001
3 (726 cm^{-1}) (613 cm^{-1}) (533 cm^{-1})	580 ± 10	0.075 ± 0.038 0.118 ± 0.035 0.182 ± 0.081	0.125 ± 0.051
3 (726 cm^{-1}) (613 cm^{-1}) (533 cm^{-1})	600 ± 10	0.90 ± 0.06 1.06 ± 0.09 1.08 ± 0.05	1.01 ± 0.07

Table S3: AID rate constants for each step in the decomposition process. Selected peak wavenumbers are listed in parentheses.

Step	T, K	k (min^{-1})	Averaged k (min^{-1})
1 (2427 cm^{-1}) (607 cm^{-1})	375 ± 5	0.20 ± 0.01 0.28 ± 0.02	0.24 ± 0.02
2 (772 cm^{-1}) (728 cm^{-1})	480 ± 5	0.020 ± 0.001 0.020 ± 0.001	0.020 ± 0.001
3 (728 cm^{-1}) (627 cm^{-1}) (501 cm^{-1})	580 ± 10	0.112 ± 0.005 0.101 ± 0.004 0.083 ± 0.003	0.099 ± 0.004

S1. Python Script:

```
# -*- coding: utf-8 -*-
"""
"""

import numpy as np
import matplotlib.pyplot as plt
#import scipy.integrate as inte
#import scipy.special as spec
#import scipy.optimize as opt
import scipy.signal as sgnl
import scipy.interpolate as inter
from sklearn.decomposition import NMF

def bs_poly(data,chi_sq,deg=3):
    x=np.arange(0,0.1*len(data),0.1)
    res=np.zeros(len(data))
    fres=np.polyfit(x,data,deg,full=True)
    #print(fres[1][0])
    #plt.plot(x,data)
    res=np.zeros(len(data))
    conv=(chi_sq-fres[1][0])/max(chi_sq,fres[1][0])
    #print(conv)
    for i in range(deg+1):
        res+=fres[0][i]*x**(deg-i)
    res=res
    #plt.plot(x,res)
    for i in range(len(data)):
```

```
        res[i]=min(res[i],data[i])
    #plt.plot(x,res)
    return res,fres[1][0],conv


def bsbin(data,binsize=100,dev=1,wscale=0.1):
    res=np.zeros(len(data))
    for i in range(len(data)):
      start=max(0,i-binsize)
      end=min(len(data),i+binsize)
      databin=data[start:end]#np.sort(data[start:end])
      w_array=np.exp(-(databin-
np.mean(databin[0:min(binsize,int(binsize*wscale))]))**2/2/(wscale*np.std(databin)**2))
      #exp(-(np.arange(end-start)-(end-start)/2)**2/2/(wscale*(end-start))**2)
      av=np.average(databin,axis=0,weights=w_array)
      av2=np.average(databin**2,axis=0,weights=w_array)
      res[i]=av-dev*np.sqrt(av2-av**2)
    return res


lcor=[-1.088999E-03,3.660977,-1.427E+02]
#[-9.514080E-04,3.571146,-1.346522E+02]
hcor=[-3.040E-04,2.912E+00,2.191E+03]
#[-9.834099E-04,3.235987,2.154054E+03]
file='Raman Data/AlH_{ind}_{region}.txt'
ind=3
region='low'#'low'#
rawdata=[]
skipdata=[]#np.array([6,19,30,44,45])#np.array([18,19,20,32,33,34,35,56,68,69,70])#np.array([6
,19,30,44,45])#
for i in [3]:
```

```
    print(np.shape(rawdata))
    if len(rawdata)==0:
        rawdata=np.loadtxt(file.format(ind=i,region=region))
    else:
        rawdata=np.vstack((rawdata,np.loadtxt(file.format(ind=i,region=region))))
xlist=np.arange(np.min(rawdata[:,0]),np.max(rawdata[:,0]))
xsize=len(xlist)+1
lrange=np.linspace(240,2450,2000)#(900,1200,301)#(900,1200,301)#
hrange=np.linspace(2400,4380,2000)
miny=0
maxy=300
maxy=min(maxy,int(len(rawdata)/xsize))
print('maxy =',maxy)
nvec=2
deg=1
thresh=0.1
normalize=False
subtract=False
DoNMF=True
data=[]#np.zeros((maxy-miny,xsize))
for i in np.arange(miny,maxy):
    k=0
    for j in skipdata:
        if i==j:
            k+=1
            print(i,k)
    if k==0:
        if len(data)==0:
```

```
            data=np.asarray([rawdata[i*xsize:(i+1)*xsize,3]-
np.min(rawdata[i*xsize:(i+1)*xsize,3])])
        else:
            data=np.vstack((data,np.asarray([rawdata[i*xsize:(i+1)*xsize,3]-
np.min(rawdata[i*xsize:(i+1)*xsize,3])])))
ylist=np.arange(miny,maxy-len(skipdata))
if region=='low':
    i1=50
    i2=960
    cor=lcor
    ran=lrange
elif region=='high':
    i1=0
    i2=960
    cor=hcor
    ran=hrange
pdata=np.zeros((len(ylist),len(ran)))
xlist=cor[0]*xlist**2+cor[1]*xlist+cor[2]
for i in ylist:
    #print(i)
    if subtract:
        res,chi_sq,conv=bs_poly(data[i,i1:i2],0,deg=deg)
        res,chi_sq,conv=bs_poly(res,chi_sq,deg=deg)
        while conv>thresh:
            res,chi_sq,conv=bs_poly(res,chi_sq,deg=deg)
            print(conv)
        data[i,i1:i2]=data[i,i1:i2]-res
    spl=inter.splrep(xlist[i1:i2],data[i,i1:i2])
    pdata[i,:]=inter.splev(ran,spl,ext=3)
```

```
    if normalize:
        pdata[i,:]=pdata[i,:]/np.linalg.norm(pdata[i,:])
u,s,v=np.linalg.svd(pdata)#data[0:maxy,i1:i2])
if DoNMF:
    model=NMF(n_components=nvec, init='nndsvd')

W=model.fit_transform(abs(pdata))#,W=np.ones((100,3)),H=pdata[2:99:40])#data[0:maxy,i1:i2]
)
    H=model.components_
fig,ax=plt.subplots(2,2)
ax[0,0].plot(s,'ok')
for i in range(nvec):
    if DoNMF:
        ax[1,0].plot(ylist,W[:,i])
        ax[0,1].plot(ran,H[i,:])
    else:
        sgn=np.sign(v[i,abs(v[i,:]).argmax()])
        ax[1,0].plot(ylist,sgn*u[:,i]*s[i])
        ax[0,1].plot(ran,sgn*v[i,:]*s[i])
X,Y=np.meshgrid(ran, ylist)
cp = ax[1,1].contourf(X, Y, np.log(pdata+300))
#fig.colorbar(cp)
plt.savefig('test.jpg', dpi = 300)
plt.show()
"""
#HNO3 Raman - March4th_low
#for i in np.arange(0,maxy-3,2):plt.plot(ran+50*i,(pdata[i]-
bsbin(pdata[1],100,0.5,0.25))/1e4+i,label='t = '+str(i)+' min')
#plt.legend(fontsize='x-small',frameon=False,handlelength=1,loc='upper left')
```

```
#plt.xlabel('Energy, cm$^{-1}$')

#plt.ylabel('Raman signal, x10$^4$ counts')

#Pure IL March2nd

fig,ax=plt.subplots(4,1,sharex=True,figsize=(10,5),gridspec_kw={'hspace':0.0,'height_ratios':
[7,1,1,1]})

ax[3].invert_xaxis()

il_low=np.loadtxt('Raman_IL_low.txt').T

no3m=np.loadtxt('NO3_minus_low.txt').T

hno3=np.loadtxt('HNO3_low.txt').T

#il_high=np.loadtxt('Raman_IL_high.txt').T

pdata[:,1657]=0.5*(pdata[:,1655]+pdata[:,1659])

pdata[:,1656]=0.5*(pdata[:,1655]+pdata[:,1657])

pdata[:,1658]=0.5*(pdata[:,1657]+pdata[:,1659])

#for i in np.arange(0,len(pdata)-2,1):

#    ax.plot(ran-35*i,1e-4*(pdata[i]-bsbin(pdata[i],100,1,1)+10000*i))

#    ax.plot(np.arange((i+1)*200,(i+1)*200-(maxy-2)*35,-35),np.arange(0,maxy-
2),'k:',linewidth=0.5)

#ax.plot([-1000,-1000],[0,0],label='HNO$_3$ 20%')

ax[1].plot(il_low[0,100:],il_low[1,100:]/np.max(il_low[1,100:]),label='[EMIM]$^+$[CBH]$^-
$')

ax[2].plot(no3m[0],no3m[1],label='NO$_3^-$')

ax[3].plot(hno3[0],hno3[1],label='HNO$_3$')

#savedata=np.reshape(ran[100:],(1,len(ran[100:])))

for i in np.arange(0,45,3):

    tempdata=np.mean(pdata[i:i+3,100:],axis=0)/5000+(i-31)

    #savedata=np.vstack((savedata,np.reshape(tempdata,(1,len(ran[100:])))))

    tempdata=tempdata-bsbin(tempdata,200,1,1)

    ax[0].plot(ran[100:],tempdata+i-31)#,label='t = '+str(2*(i-32))+' min')

#ax[0].legend(fontsize='x-small',frameon=False,handlelength=1,loc='upper left')
```

```
ax[3].set_xlabel('Wavenumber, cm$^{-1}$')
#ax[0].set_xlim((2416,354))
ax[2].set_ylabel('Reactant spectra')
ax[0].set_ylabel('Signal, arb. units')
ax[3].set_xticks(np.arange(0,2500,250))
for i in ax[1:]:
    i.legend(fontsize='x-small',frameon=False,handlelength=0,loc='upper left')
    i.set_yticks([])
"""
```

S2. Fitting python script:

```
# -*- coding: utf-8 -*-
"""
"""

import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt
import scipy.interpolate as inter
import scipy.misc as ms

gauss=lambda x,x0,A,w:A*np.exp(-0.5*((x-x0)/w)**2)/np.sqrt(2*np.pi)/w#gaussian lineshape function
lorentz=lambda x,x0,A,w:A/np.pi/w/(1+((x-x0)/w)**2)#lorentzian lineshape function
agauss=lambda x,x0,s,w: gauss(x,x0,1e3,w/(np.exp(-s*(x-x0))+1))#asymmetric gaussian lineshape function
BBR=lambda x,b,T: 1e45*b*2*h*c**2/x**6/(np.exp(1e9*h*c/x/kb/T)-1)#modified 1/lambda^6 Planck's function
l_bbr=lambda T,x,b:1e45*2*h*c**2*b/x**6/(np.exp(1e9*h*c/x/kb/T)-1)#formulation of modified Planck's function for numerical evaluation of temperature derivative
lin=lambda x,x0,A,w:A*(x-x0)
```

```
quad=lambda x,x0,A,w:A*(x-x0)**2
poly=lambda x,x0,A,w:A*(x-x0)**w


class linpeak:#spectral peak class
    def __init__(self,x0,A,w,name,fun='gauss'):#initialization
        self.name=name#peak name
        self.x0=x0#peak linecenter
        self.A=A#peak area
        self.w=w#peak width
        self.fun=fun#peak lineshape function
    def val(self,x,x0=None,A=None,w=None):#evavaluation of a peak at wavelength x
        if x0==None: x0=self.x0
        if A==None: A=self.A
        if w==None: w=self.w
        return self.fun(x,x0,A,w)
    def der(self,x,n=1,x0=None,A=None,w=None):#evaluation of nth derivative of a peak
lineshape function
        return ms.derivative(self.val,x,dx=1e-3,n=n)


def BBR_fit(xdata,ydata,sigma,init=[1e-6,2000]):#non-linear fit of spectral data (ydata) defined
at wavelength xdata to a modified Plank's formula


fres,pcov=opt.curve_fit(BBR,xdata,ydata,sigma=sigma,absolute_sigma=True,p0=(init[0],init[1])
)
    perr=np.sqrt(np.diag(pcov))
    return fres,BBR(xdata,fres[0],fres[1]),perr


def dirfit(xdata,ydata,peaks,bs=0,der=0,fit_bbr=1,bg=(0,-1),NoPlot=True): #constrained linear
least-square fit of spectral data
    a=np.zeros((len(xdata),(der+1)*len(peaks)+2*bs+1+int(fit_bbr)))
```

```
    for i in range(der+1):
        for j in range(len(peaks)):
            a[:,j+i*len(peaks)]=peaks[j].der(xdata,n=i)
    for i in range(2*bs+1):
        a[:,(der+1)*len(peaks)+i]=np.real((1j)**(-i)*np.exp(-2*np.pi*((i+1)//2)*1j*(xdata-
xdata[0])/(xdata[-1]-xdata[0]))+(-1j)**(-i)*np.exp(2*np.pi*((i+1)//2)*1j*(xdata-
xdata[0])/(xdata[-1]-xdata[0])))
    for i in range(int(fit_bbr)):
        a[:,-fit_bbr+i]=poly(xdata,2350,1,i)#ms.derivative(l_bbr,2500,n=i,args=(xdata,1e-
6),order=max(3,2*i-1))
    b1=np.zeros((der+1)*len(peaks)+2*bs+1+int(fit_bbr))
    b2=b1+np.inf
    #b1=-np.inf
    cov=np.linalg.inv(np.dot(a.T,a))
    res=opt.lsq_linear(a,ydata,bounds=(b1,b2))
    fitres=np.zeros(len(xdata))
    if not NoPlot:
        plt.plot(xdata,ydata,'k')
        #fitres=np.zeros(len(xdata))
        for i in np.arange(np.shape(a)[1]):
            fitres+=res.x[i]*a[:,i]
        plt.plot(xdata,fitres,'r')
        for i in range(len(peaks)):
            plotdata=np.zeros(len(xdata))
            for j in range(der+1):
                plotdata+=res.x[i+j*len(peaks)]*a[:,i+j*len(peaks)]
            plt.plot(xdata,plotdata,'g')
    err=np.sqrt(np.diag(cov)*res.cost/len(xdata))
    bsdata=np.zeros(len(xdata))
```

```
    for i in range(2*bs+1):
        bsdata+=res.x[i+(der+1)*len(peaks)]*a[:,i+(der+1)*len(peaks)]
    for i in range(int(fit_bbr)):
        bsdata+=res.x[-fit_bbr+i]*a[:,-fit_bbr+i]
    #if fit_bbr>0:
    #    try:
    #
fitted_BBR=BBR_fit(xdata[bg[0]:bg[1]],bsdata[bg[0]:bg[1]],sigma=np.sqrt(ydata[bg[0]:bg[1]]+
1000)/inter.splev(xdata[bg[0]:bg[1]],cs),init=[1e-6,2500])
    #    except:
    #        fitted_BBR=([0,300],[None],[0,0])
    #    if not NoPlot:
    #        plt.plot(xdata,BBR(xdata,fitted_BBR[0][0],fitted_BBR[0][1]),'b')
    #else:
    #    fitted_BBR=([0,300],[None],[0,0])
    if not NoPlot:
        plt.plot(xdata,ydata-bsdata,'purple')
    return res.x,err,fitres,ydata-bsdata


'''
First define list of peaks, e.g. peaks=list()
Populate peaks with lineshape functions via peaks.append(linpeak(x0,A,w,name,function))
To use a spectral overlay [ovr_x,ovr_y], define an interpolation function
ovr_fun(x,x0=None,A=None,w=None), where x0,A,w are dummy arguments.
Populate peaks with the function as peaks.append(linpeak(x0,A,w,name,function=ovr_fun)) to
ensure that that it return interpolated value upon peaks[i].val(x) call
Run the main program via


out = dirfit(xdata,ydata,peaks,bs=0,der=0,fit_bbr=1,bg=(1000,-250),NoPlot=True)
```

- xdata is the wavelengths grid,

- ydata is the spectral intensity

- bs is the number of sin, cos functions to use for the background (0 is constant offset, integer values n > 0 add sin and cos functions with period of

(lambda_max - lambda_min)/n)

- der is the highest order of wavelength derivatives for the peaks lineshape function to use (0 - use only functions themselves, 1 - use 1st derivative, 2 - use 2nd edrivative and so on)

set der = 0 if positions and lineshapes are well-known. Otherwise set der = 1 to allow small corrections for x0 position, der = 2 to allow correcting x0 and w.

Higher der values provide better fit but may return non-physical peaks.

- fit_bbr is the maximum order of Planck's function temperature derivative to use for baseline fitting

(0 - to not use the modified Plank's function, 1 - use the function itself, 2 - use the function + 1st derivative and so on)

- bg is the wavelength range used for the non-linear fit of the baseline to the Planck's formula for temperature determination

The function returns out = (out[0],out[1],out[2],out[3])

where out[0] is array containing peak areas, out[1] is 1-sigma uncertrainty for out[0], out[2] is amplitude,temperature determined

in the modified Planck's formula fit and their uncertainties, out[3] is baseline obtained in the linear fit

Set NoPlot to False to enable plotting spectral data, peaks and the Planck's baseline.

'''

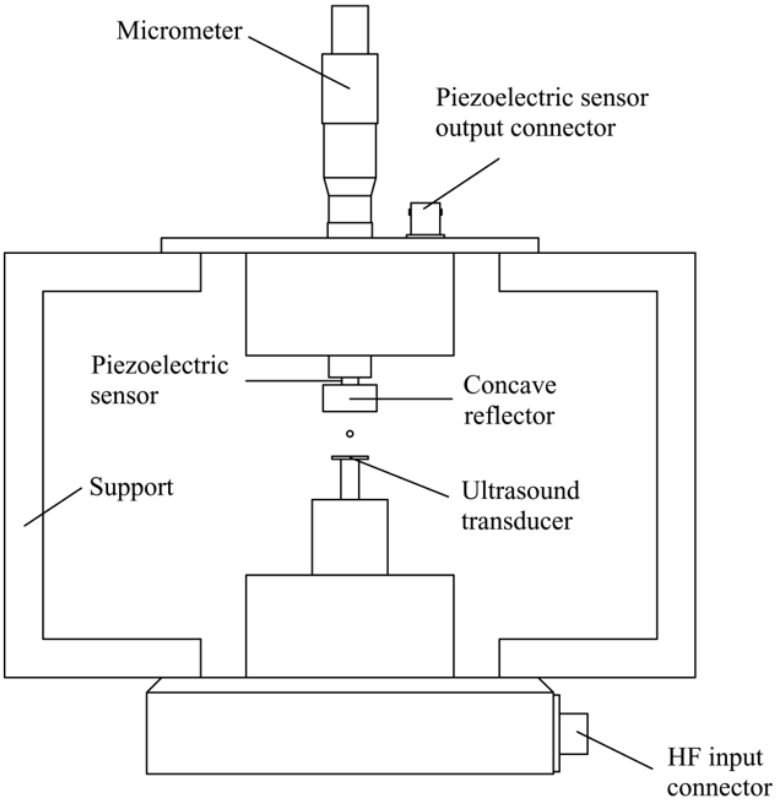

Figure S9. A schematic diagram of the levitator. Ultrasonic sound waves are generated by the piezoelectric transducer. A standing wave is generated due to multiple reflections between the transducer and the concave reflector. The micrometer displayed allows the distance between the transducer plate and the reflector to be adjusted. The pressure amplitude of the standing wave is monitored by connecting the output of the piezoelectric sensor via connector to an oscilloscope. The RF power to the transducer is input via the HF connector.

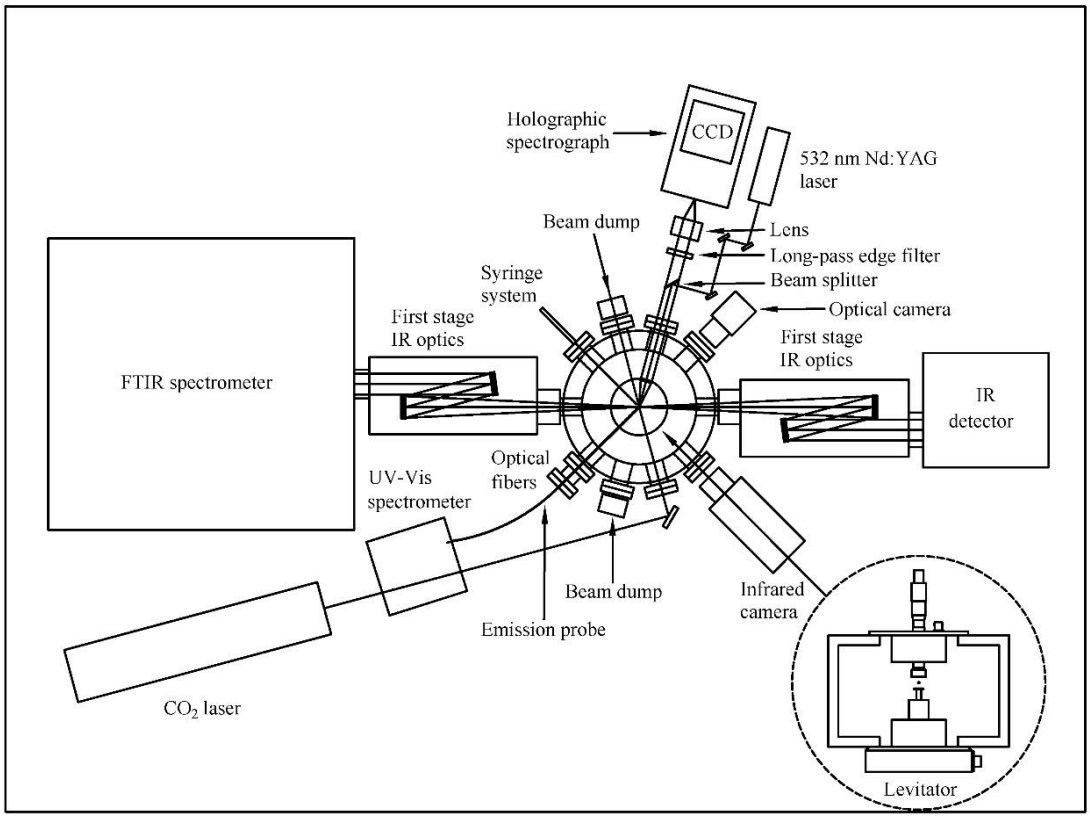

Figure S10. Schematic top view of the complete levitator apparatus displaying the ultrasonic levitator, process chamber, carbon dioxide laser, Raman spectrometer, infrared camera, optical camera, and some other complementary spectroscopy tools (FTIR spectrometer and fiber optic UV-vis spectrometer).[6]

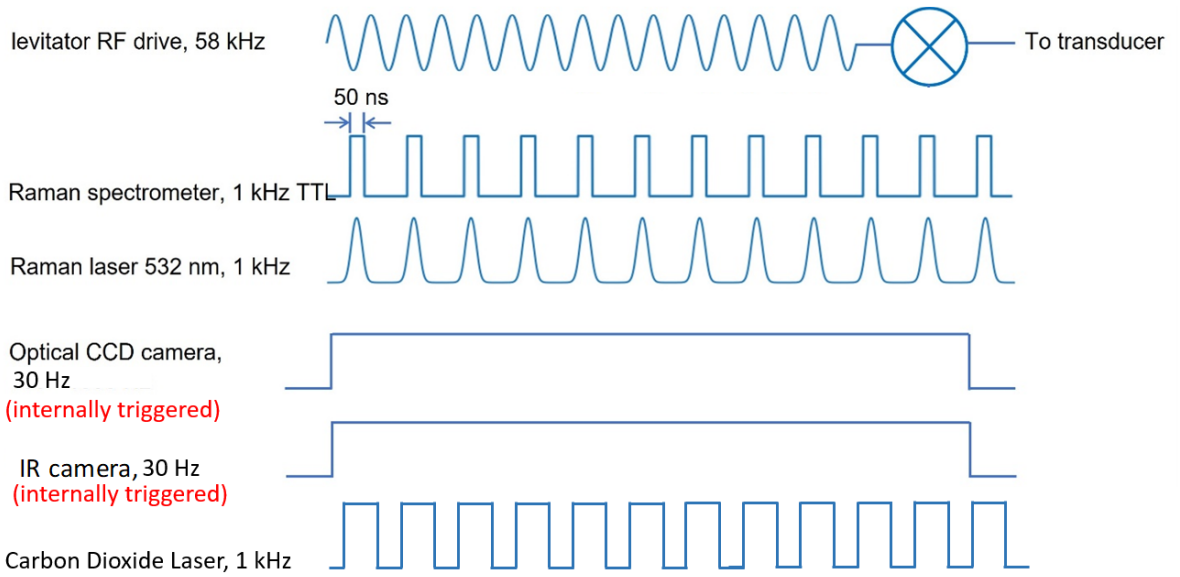

Figure S11. Typical pulse sequence used for operation of the ultrasonic levitator. Both Optical and IR camera were internally triggered and run at 30 Hz.

References

(1) Miller, K.K.; Creegan, S.E.; Unruh, D.K.; Pantoya, J.D.; Hill, K.J.; Tran, Q.; Pantoya, M.L. Acid Base Synthesis of Aluminum Iodate Hexahydrate Powder as a Promising Propellant Oxidizer. *Chem. Eng. J.* **2023**, *453*, 139953.

(2) Brotton, S.J.; Kaiser, R.I. Novel High-Temperature and Pressure-Compatible Ultrasonic Levitator Apparatus Coupled to Raman and Fourier Transform Infrared Spectrometers. *Rev. Sci. Instrum.* **2013**, *84*, 055114.

(3) Brotton, S.J.; Kaiser, R.I. In Situ Raman Spectroscopic Study of Gypsum ($CaSO_4 \cdot 2H_2O$) and Epsomite ($MgSO_4 \cdot 7H_2O$) Dehydration Utilizing an Ultrasonic Levitator. *J. Phys. Chem. Lett.* **2013**, *4*, 669-673.

(4) Lucas, M.; Brotton, S.J.; Min, A.; Woodruff, C.; Pantoya, M.L.; Kaiser, R.I. Effects of Size and Prestressing of Aluminum Particles on the Oxidation of Levitated exo-Tetrahydrodicyclopentadiene Droplets. *J. Phys. Chem. A* **2020**, *124*, 1489-1507.

(5) Brotton, S.J.; Perera, S.D.; Misra, A.; Kleimeier, N.F.; Turner, A.M.; Kaiser, R.I.; Palenik, M.; Finn, M.T.; Epshteyn, A.; Sun, B.J.; Zhang, L.J.; Chang, A.H.H. A Spectroscopic Investigation on the Oxidation of exo-Tetrahydrodicyclopentadiene (JP-10; $C_{10}H_{16}$) Doped with Titanium-Aluminum-Boron Reactive Metal Nanopowder (RMNP). *J. Phys. Chem. A* **2022**, *126*, 125-144.

(6) Brotton, S. J.; Kaiser, R. I. Effects of Nitrogen Dioxide on the Oxidation of Levitated exo-Tetrahydrodicyclopentadiene (JP-10) Droplets Doped with Aluminum Nanoparticles. *J. Phys. Chem. A* **2021**, *125*, 2727-2742.