1 Truth Behind the Scenes

Welcome to challenge the world!

Content under Creative Commons Attribution license CC-BY 4.0, code under MIT license (c)2015 C.C.Huang.

1.1 Who

- cchuang,

and

- You

1.2 How

- Lectures: collected or done by me, someday, by you all;
- Labs: discover the computer's ability;
- Home Work: help you to familar theory and tools we introduce;
- on-line Review Quizzes: Got your points just as you want;
- Group Project: learn how to incorporat with your company.

2 Lecture Resources

- [CGU] http://diffusion.cgu.edu.tw/ (http://diffusion.cgu.edu.tw/)
- [github] https://github.com/cchuang2009 (https://github.com/cchuang2009)

3 Reference Books

3.1 Art:

- **B. Artalay**: Math and the Mona Lisa: The art and Science of Leonardo Da Vinci
- **Jan Steward**, The New Mathematics of the Living World,

3.2 Science and Computer:

- **Hans Petter Langtangen**: Python Scripting for Computational Science, 3rd Ed. 2007, Springer Publ.
- **Hans Petter Langtangen**: A Primer on Scientific Programming with Python, 2012, Springer Publ.
- **Robert K. Adair**: The Physics of Baseball 3rd Ed. 2002, Perennial.

3.3 Course Syllabus

Mathematics is an important field in scientific study. Nowadays, It becomes more vivid than ever due to the progress in innovatory science and introduction of modern technology, such as computers and internet. Such kind of technology also bring the new phenomena which human being always neglect or have not observed in the history.

This course will be an introduction to the mathematical modeling of systems in nature and around our life.

Observations on the specific topics dicussed in the lecture would be progressed by theoretical analysis and computer simulation simulataneously with which we could explore truth and the secret behind the problems deeply more than ever (hope so)..

Though the computer skills is not the necessary in this course, pedagogy always includes an introduction to mathematical model building, exploring simple hands-on and simulated experiments using computers mathematical software.

The models introduced in this course includes:

- Epidemic Model: Fractal dimension leads to vaccination strategies:
- Baseball: The reason way the slider of Chien-Ming Wang is very hard to be hit.
- Random : How to estimate the true value of pi only by tossing a stick on ruled paper?
- Percolation: Why does the forest fire damage human beings?
- Weather prediction: The path of Typhoon affected by Coriolis Force.

Mostly, numerical analysis will be taken to explore reality behind the selected models instead of induced by pure mathematical theory while it is too complicate to lay-in. Students will have the opportunity to develop and analyze mathematical models of real world both in continuous and discrete processes.

3.4 中文課程簡介

自從上世紀以來，數學在定性研究上有許多突破性的發展，再加上計算機軟硬體的快速進展，這就是常被用來探討生活中許多現象產生的原因。這一門課程，我們將結合兩者的長處，藉由探討實際的例子，介紹各種可以應用到學生本業的實用工具。課程涵蓋的例子，主要包含：

- 花朵的模型，為什麼植物知道 Fibonacci 的級數理論?
- 傳染病模型: 2003 SARS 為什麼像森林的火災，來勢洶洶重創台灣, 港, 大陸? Dengue Fever in Tainan.
- 為什麼王建民的 Sinker 那麼厲害?
- 花豹的斑紋，蛇的斑紋，為什麼跟心臟電壓有關係?

這些例子所包含的數學工具，包含甚廣，從數論，微分方程，最佳化等等分析理論到圖學，隨機理論皆涵蓋，也是醫學、工程及管理領域，常見的工具，課程的另一目的，即藉由探討實際問題中，學習如何學以致用的精諦。

[Topics]

- Basics of Python (8/2014-2-8.ipynb)
- Discrete Dynamics (1/2013-2-math-1.ipynb)
- Topology: Dimensions, Chaos, Fractals (2/2014-2-math-2.ipynb)
- Sequences: Basic of Numerics, Sound, and Images (3/2015-3-18-Fourier.ipynb)
- Animation: Pursuit Curves, Blind men and Elephant, and Storage of Radioative Waste (4/2013-2-4-1.ipynb)
- Epidemic Model (14/sir.ipynb)
- Fly Fly to the Sky: Phugoid (5/2013-2-5-1.ipynb)
- Magic (5/magic.ipynb)
- Probability (6/2013-2-6.ipynb)
- Estime π by Buffon Needle Model (6/Buffons_Needle_Sim.ipynb)
- Spiral and Spots (6/reactiondiff-2.ipynb)
- Markov Chain: From Money Ball... (7/2013-2-7.ipynb)
- Lottery (8/lottey.ipynb)
- Greatest Indian Rope-Strick (13/index.ipynb)
- [Trajectory of Whale swimming](10/Analyzing whale tracks.ipynb)
- FIFA the Soccer (11/soccer.ipynb)
- Schrödinger Wall (12/widget.ipynb)
- Awesome Butterfly Ball: Magnus Effect (18/baseball.ipynb)
- Titanics Accident- Seaborn Introduction (17/titanic_seaborn.ipynb)
- Competition among Apple Inc, TMSC, Samsung - Pandas Introduction (20/Final.ipynb)

3.5 How to Run Codes

- get the codes from the following link(ipynb, to run on your site; html, just read)
  - 2016 Sources on GitHub (https://github.com/cchuang2009/math-2016-1/)
  - ~~</a href="http://diffusion.cgu.edu.tw/2014/math/2014-2">diffusion.cgu.edu.tw/2014/math/2014-2 (http://diffusion.cgu.edu.tw/2014/math/2014-2">diffusion.cgu.edu.tw/2014/math/2014-2)~~
- set up the same Python/Jupyter environment to run
  - get ~~</a href="diffusion.cgu.edu.tw/ftp/porteus/">our live linux~~
  - install Anaconda (http://continuum.io/downloads) run as I ran on lecture.
  - register a sagemath account (https://cloud.sagemath.com), upload the ipynb and run.

4 Software Requirements

- Python - ~~2.7~~.x or 3.6.x+, the latter is recommendered and on which we should work;

```
Python-3.6.1 used here
```

- SciPy/NumPy, Pandas, Sympy;
- Matplotlib-2.x, seaborn-0.8,x, pandas;
- IPython(5.+), ipywidgets(5.+), JSAnimation,

[Optional]

Too much to listed in detail!

4.1 Installation Howto

You can use one of the following to install the Python platform on which we would work during this semester and future.

1. install **anaconda** (Linux, Mac, Windows);
2. register an account at **SageCloud** (Any Devices);
3. install the Python and related packages fron scratch if you want to explore system structure and challenge by yourself; it would take plenty of time, day and night, but it is deserves for this.

To make work much easily, install platform on USB stick (>8G bytes) to play on your own computer or laptop.

4.2 Grading

- Lecture Attendance and Computer Practices: 40%
- Quiz: 40%
- Term report in group project: 20%

Final Project

Your chance to show off what you learned

- Hand-ins, written paper ($\leq$ 200 words) • screencast
- poster
- webpage

4.3 Office hour

- Tuesday, 12:00 P.M. ~ 13:00 P.M.
- Wednesday, 12:00 P.M. ~ 13:00 P.M.

In [ ]:

4.4 Survey

You have to follow the link to complete the survey:

- LoginIn (http://120.126.22.84/CGU)

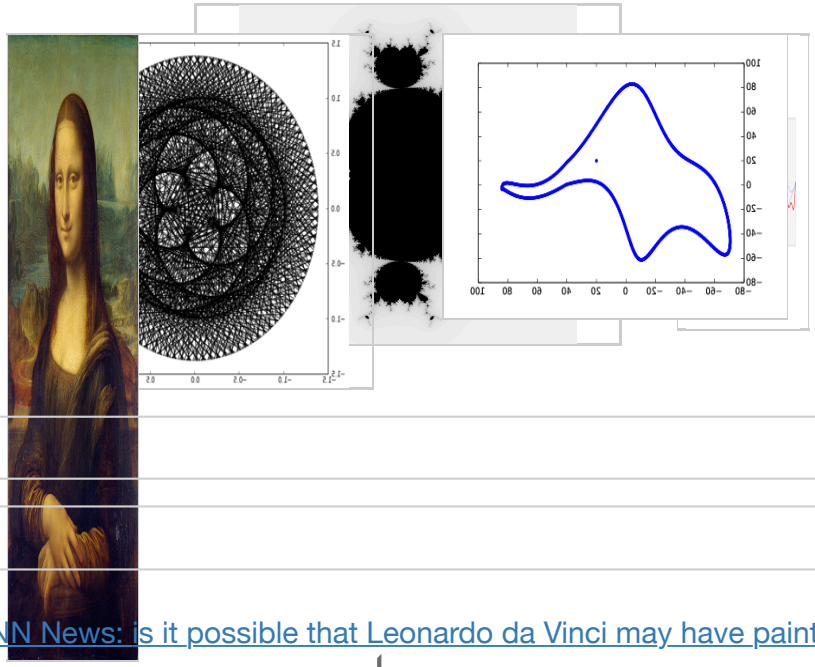
4.5 Marilvn vos Savant

```
To acquire knowledge, one must study;
but to acquire wisdom one must observe
```

.

In [29]:
```
from IPython.core.display import HTML
def css_styling():
    styles = open("styles/custom.css", "r").read()
    return HTML(styles)
css_styling()
```

Out[29]:



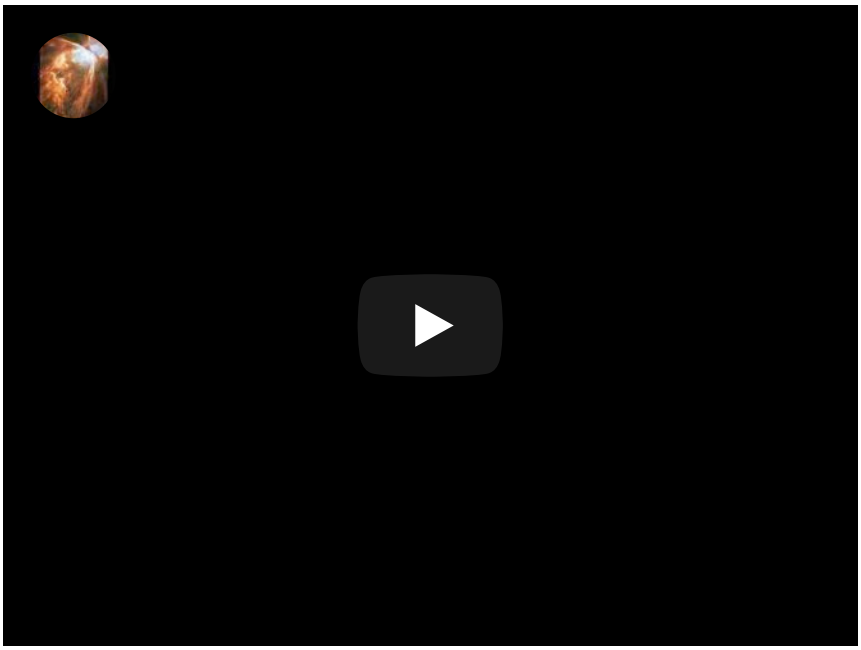
CNN News: is it possible that Leonardo da Vinci may have painted another 'Mona Lisa? Display (https://edition.cnn.com/style/article/isleworth-mona-lisa/index.html)

4.6 Pre-Quizzes

1. ( ) Do you hear Golden Ratio?
2. ( ) Sine function, $\sin(x)$, is a well-known function in the world; how many pictures above are concerned with it?
3. ( ) Almost everyone enjoys music. Could you distinguish what
   - the different intruments playing, flute or guitar, for example;
   - kinds of play, analog or digital?
4. ( ) As human being living in the modern world, are computer game, Pokemon Go for instance, far from you?
5. ( ) Allen Turing broke the myth of German's Enigma and wan the world War II. Does Turing, who is alien to Deutch language, break deutch encryption scheme by hand computation?

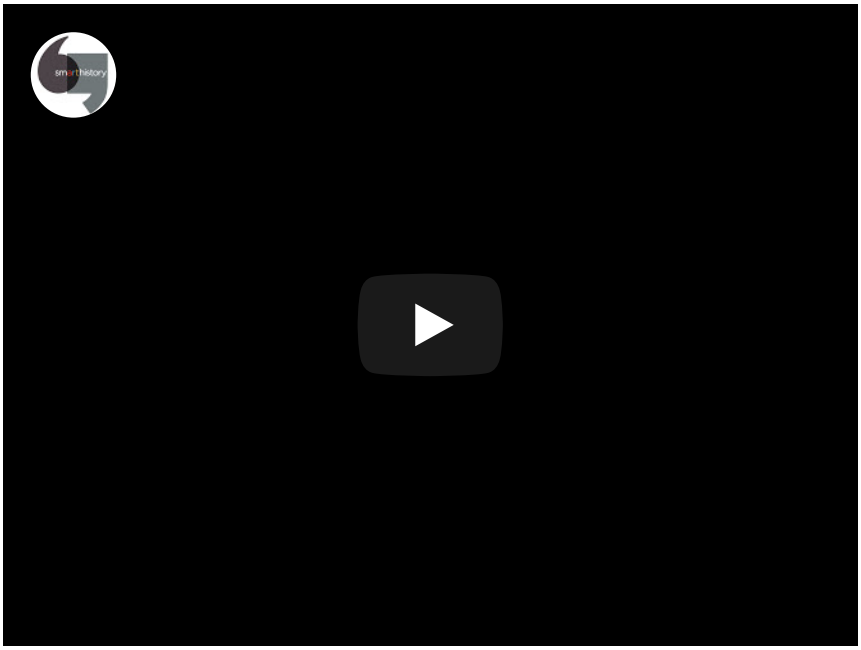
In [30]:
```
from IPython.display import YouTubeVideo
#  Dean Martin - Mona Lisa
YouTubeVideo('iRe6AZlswmw')
```

Out[30]:



In [31]:
```
from IPython.display import YouTubeVideo
#  Dean Martin - Mona Lisa
YouTubeVideo('3kQ_p2EZX4Q')
```

Out[31]:



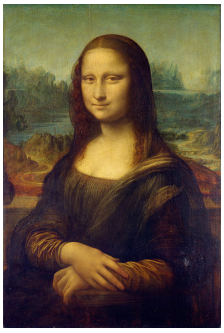
In [ ]:

4.7 What The Golden Ratio is

$$\gamma \sim 1.6180339887498949 \quad \left(= \frac{1+\sqrt{5}}{2} \right)$$

5 Secrect in Art

5.1 Mona Lisa



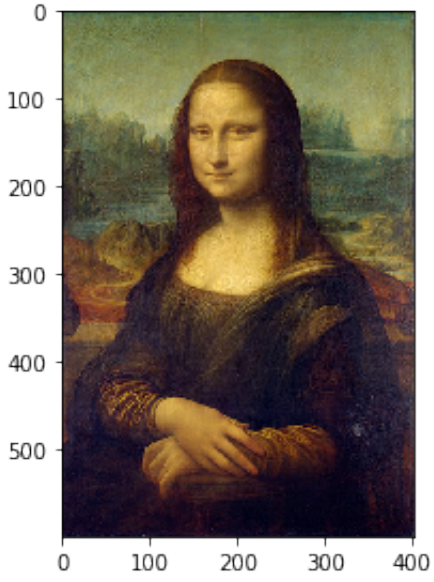
What is the relation between γ and this materpiece?

In [32]:
```
%matplotlib inline

import matplotlib.pyplot as plt
from skimage.segmentation import slic,mark_boundaries
#from skimage.data import lena
from skimage.util import img_as_float
import skimage.io as io
import numpy as np
from numpy import pi,sin,cos,sqrt
```

In [33]:
```
Monalisa = io.imread('imgs/Monalisa.png')
print('image shape:', Monalisa.shape)
io.imshow(Monalisa)
#io.show()
```

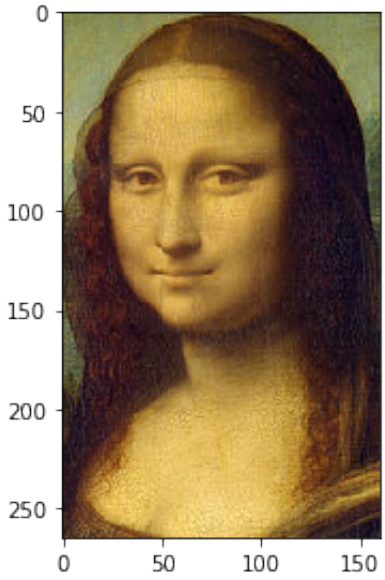
```
image shape: (599, 402, 3)
```

Out[33]:
```
<matplotlib.image.AxesImage at 0x1305c4940>
```



In [34]:
```
import skimage
Monalisa_zoom = Monalisa[55:320, 120:280]
#Monalisa_filter=skimage.restoration.denoise_tv_chamb
io.imshow(Monalisa_zoom) #io.imshow(Monalisa_filter)
```

Out[34]:
```
<matplotlib.image.AxesImage at 0x13062f320>
```



In [35]:
```
# Estimate the ratio of height/width of beautiful fac
RR=(320-55)/(280.-120)
Error=(RR-1.618)/1.618
print(RR)
print('error = %s' %Error)
```

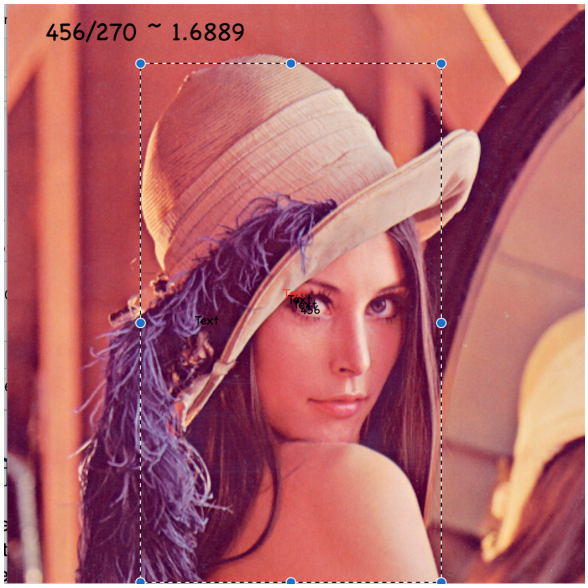
```
1.65625
error = 0.023640296662546287
```

In [4]:
```
2.8/1.7, 3.7/2.8,3.4/2.8
```

Out[4]:
```
(1.6470588235294117, 1.3214285714285716, 1.2142857142
857144)
```

5.2 Famous Photo,

Below photo is generally, [Lenna (https://en.wikipedia.org/wiki/Lenna)](https://en.wikipedia.org/wiki/Lenna), found in image manipulation:



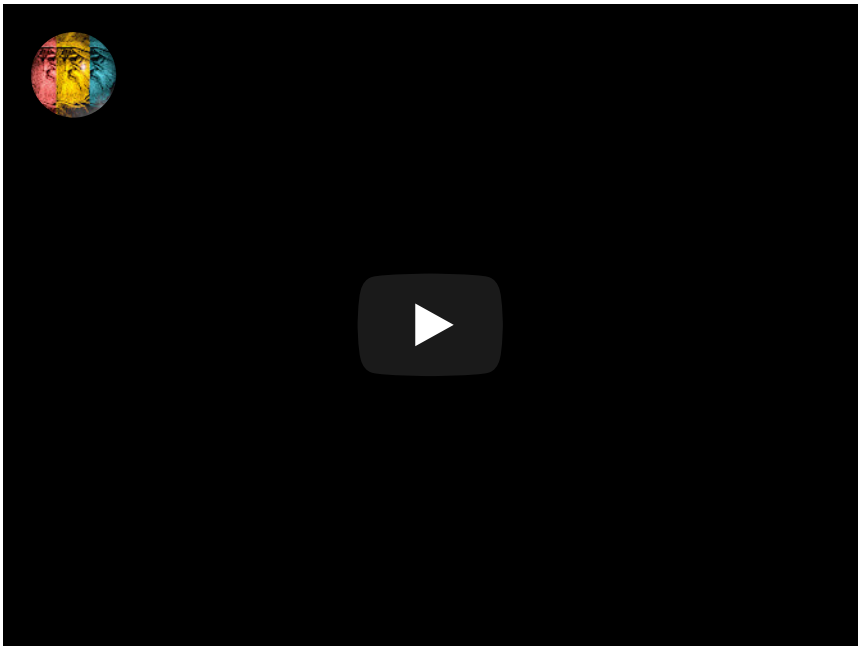
5.3 Sunflower

Within the continuous evolution with wild world from far far ago and are still alive today, it's amazing the fact that plants have known how to overcome the deficiencies of necessay supplements, such like sun light , water for instance.

> And how do they do?

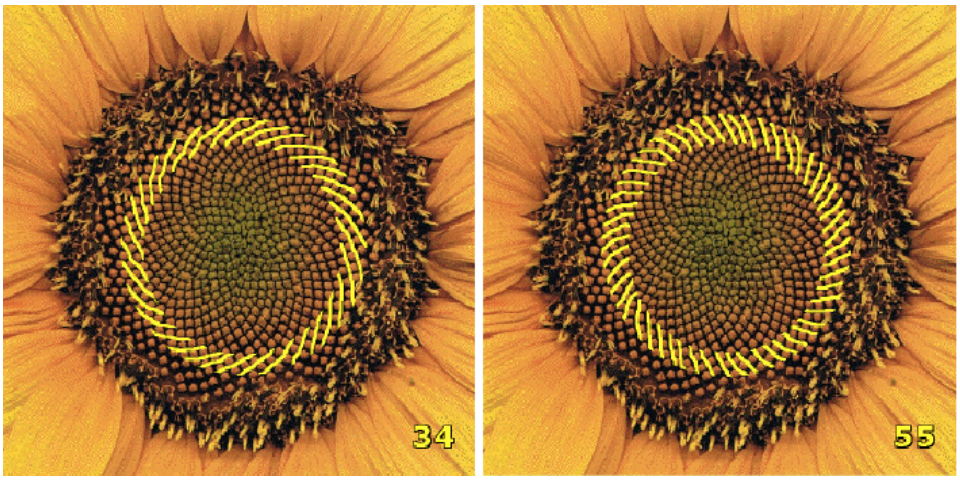
In [36]:
```
YouTubeVideo('4VrcO6JaMrM')
```

Out[36]:



In [10]:
```
from IPython.display import Image
Image("imgs/sunflower121.png")
```

Out[10]:



Type *Markdown* and LaTeX: α^2

In [ ]:
```

```

It is very close to γ!

What kind of infomation hidden in the pair (34,55)?

5.4 Fibonicci Sequence

The term follows the following rules:

$$\begin{aligned} f_0, f_1 &= 1, \\ f_2 &= f_0 + f_1 = 2, \\ &\vdots \\ f_{n+1} &= f_n + f_{n-1}, n = 1, 2, 3, \end{aligned}$$

This means:

$$\{f_n\}_0^\infty = 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \cdots$$

5.5 Limit of Rotio of Sequences

As well-known:

$$\mathbf{f_n = C_1\left(\frac{1+\sqrt{5}}{2}\right)^n + C_2\left(\frac{\sqrt{5}-1}{2}\right)^n}$$

5.5.1 Conclusion

$$\lim_{\mathbf{n\to\infty}} \frac{\mathbf{f_{n+1}}}{\mathbf{f_n}} = \frac{\mathbf{1+\sqrt{5}}}{\mathbf{2}} \sim \mathbf{1.6180339887498949}$$

Just the Golden Ratio!

5.6 Math Description

Replace the $\mathbf{f_n = ar^n}$:

$$\begin{aligned} \mathbf{f_{n+1} = f_n + f_{n-1}} &\Longrightarrow r^2 = r + 1 \\ r &= \left(\frac{1+\sqrt{5}}{2}\right) \text{ or } \left(\frac{\sqrt{5}-1}{2}\right) \end{aligned}$$

This implies the solution is in the following form:

$$\mathbf{f_n = C_1\left(\frac{1+\sqrt{5}}{2}\right)^n + C_2\left(\frac{\sqrt{5}-1}{2}\right)^n}$$

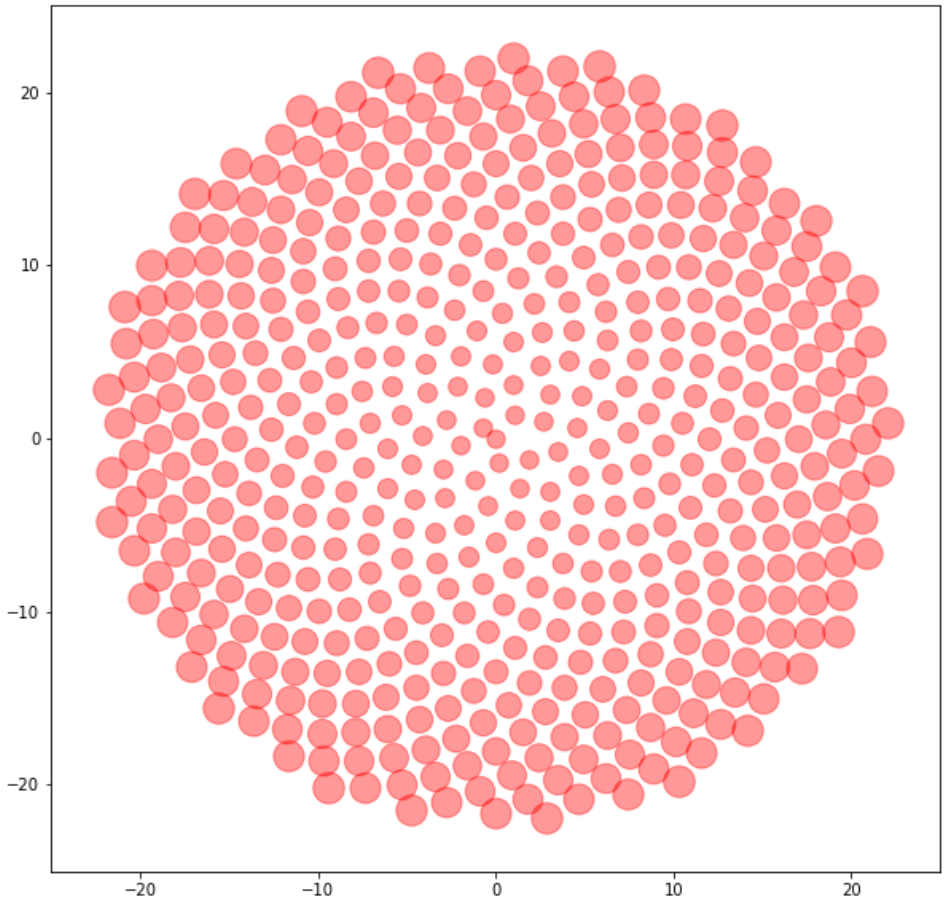
5.7 Unit In Radical Degree

It is congrant to 137.52°

$$
\begin{aligned}
1.618 \times 360^{\circ} &= 582.48^{\circ} \\
(-360^{\circ}) &\rightarrow 222.48^{\circ} \\
(-360^{\circ}) &\rightarrow -137.52^{\circ} \\
(\times -1) &\rightarrow 137.52^{\circ}
\end{aligned}
$$

In [37]:

```
plt.figure(figsize=(10,10))
plt.xlim(-25,25)
plt.ylim(-25,25)

c=1;r=1;num=500;

for n in np.arange(1,num):
  R=c*sqrt(n-1);
  rr=float(n)/float(num)+0.3;
  size=0.8*(2+4*n/float(num))

  angle=(n-1)*pi*137.5/180. #best angle about 137.5
  x=R*cos(angle);y=R*sin(angle)

  plt.scatter(x, y,c='r', s=size*80, alpha=0.4)
```

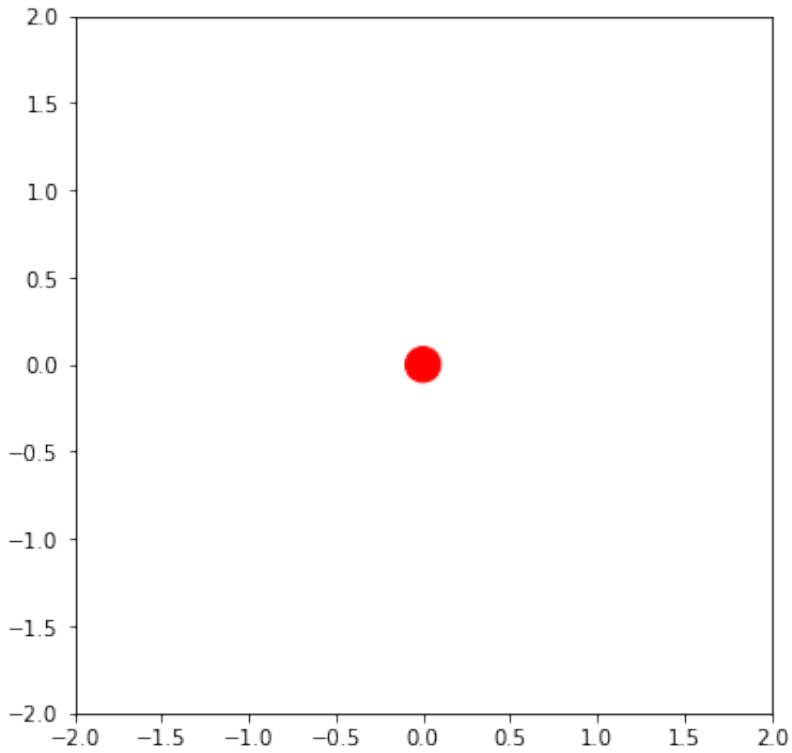


5.8 Replica the Plant (Animation)

One Pictire is better than one thousand words, and one animation is better than one thousand pictures.

To simulate the plants' growth model, let the points expand by degree, 137.52° , toward outside.

In [38]:
```
from JSAnimation.IPython_display import display_animat
import matplotlib.pyplot as plt
from matplotlib import animation
import numpy as np

fig = plt.figure(figsize=(6,6))

ax = plt.axes(xlim=(-2, 2), ylim=(-2, 2),aspect="equal
#ar=137.52
ar=139.5
r0=0.1
r=0.03
deg2rad=np.pi/180.
c = ax.add_patch(plt.Circle((0, 0), radius=r0,color="#
Nframes = 50
def animate(nframe):
    f = nframe
    rm=r0+np.sqrt(f)*0.014
    ang=f*ar*deg2rad
    gam= 0.5*(1.4-f/float(Nframes))
    c = ax.add_patch(plt.Circle((r*f*np.cos(ang), r*f*
    return c
```



```
#To-Do
anim = animation.FuncAnimation(fig, animate,
frames=Nframes, interval=100)
display_animation(anim, default_mode='once')
```

5.9 Review Exercise:

1. Consider the image

Dive out the concern golden ratio.

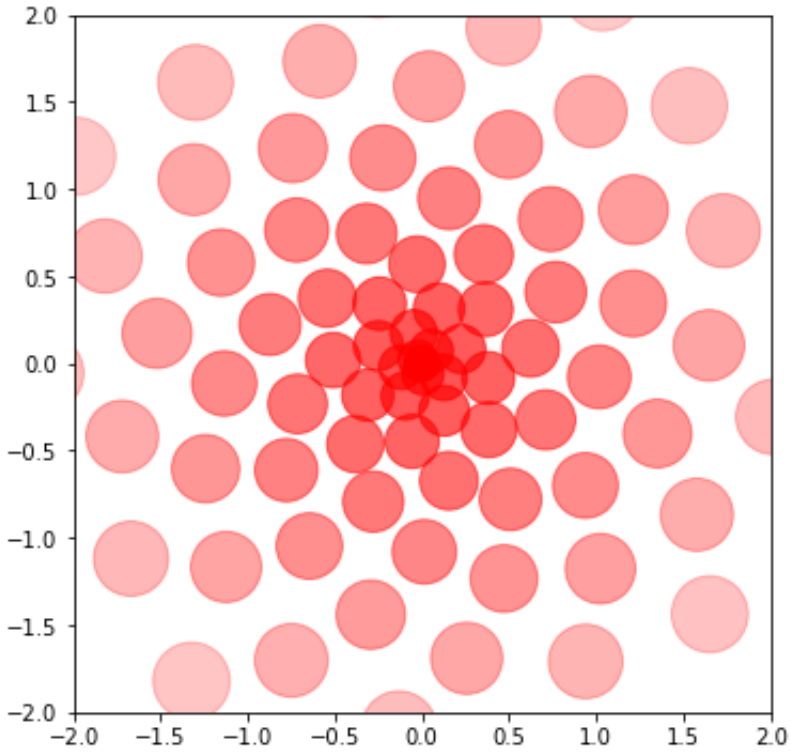
5.10 All Perfeact?

Not all plants grow by using the congrant angle, 137.52° , reference above. But how does the quantity affect the plants' shape?

In [41]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from ipywidgets import interactive,interact,FloatSlid
from IPython.display import display, clear_output
```

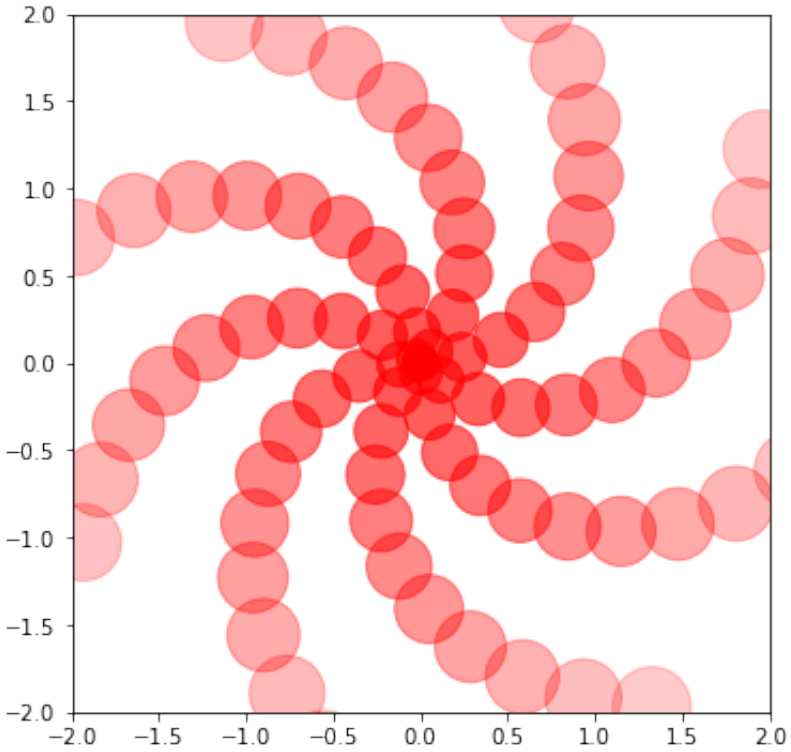
In [42]:
```
def flowerReplicate(arg=137.52):
    fig = plt.figure(figsize=(6,6))
    ax = plt.axes(xlim=(-2, 2), ylim=(-2, 2),aspect="

    r0=0.1
    r=0.03
    deg2rad=np.pi/180.
    c = ax.add_patch(plt.Circle((0, 0), radius=r0,col
    N = 80
    for i in np.arange(N):
        rm=r0+np.sqrt(i)*0.014
        ang=i*arg*deg2rad
        gam= 0.5*(1.4-i/float(N))
        c = ax.add_patch(plt.Circle((r*i*np.cos(ang),
    return c
```

In [43]:
```
FlowerSlider = FloatSlider(value=137.52, min=136., ma

# initial plot
flowerReplicate(arg=137.52)

def on_slider_change(change):
    clear_output(wait=True)
    flowerReplicate(change['new'])

FlowerSlider.observe(on_slider_change, names='value')
#display(FlowerSlider)
```



In [44]:
```
w = interactive(flowerReplicate, arg=(136.,140.,0.2))
display(w)
```

arg 136.00



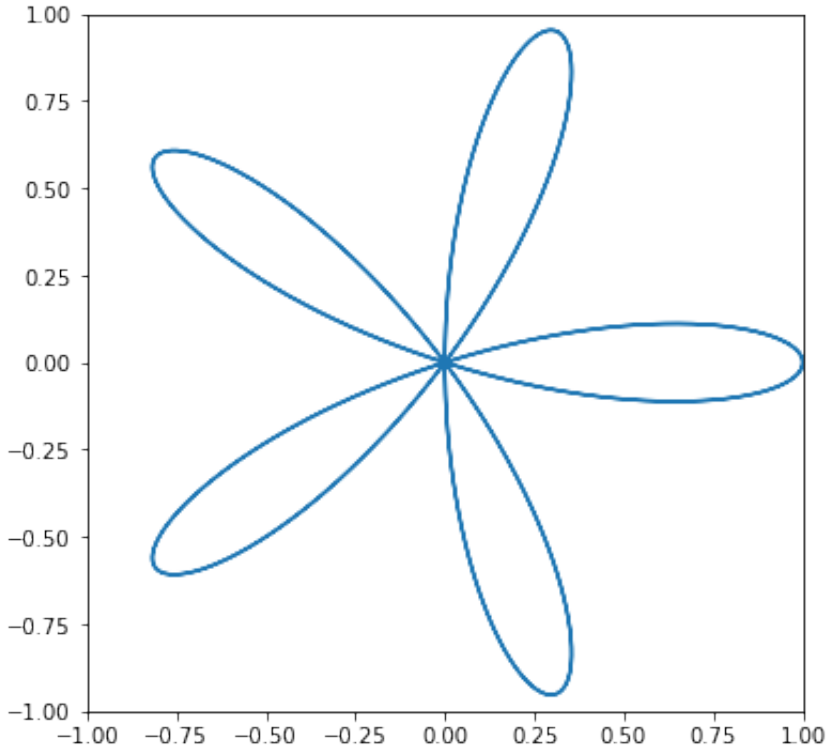
5.11 Image

$\cos(2n\theta)$ or $\sin(2n\theta)$ where n being natural number gives $n/2n$-pedal rose curve respectively.

In [45]:
```
from numpy import sqrt,pi,sin,cos
```

In [46]:
```
t=np.arange(0,2*pi,0.01)

plt.figure(figsize=(6,6))
plt.plot(cos(t)*cos(5*t),sin(t)*cos(5*t))
plt.xlim(-1,1)
plt.ylim(-1,1)
```

Out[46]: (-1, 1)



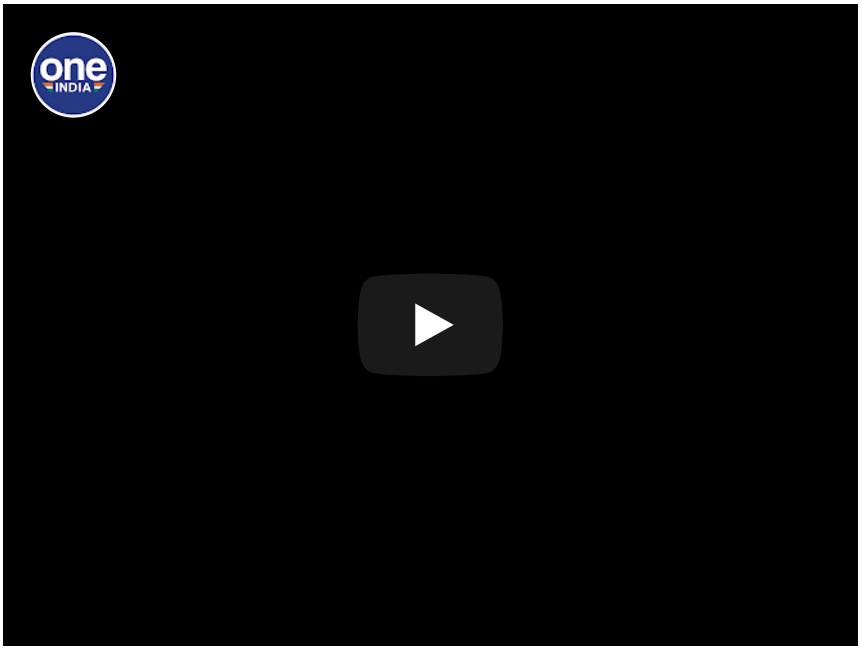
Prelude: "La Vie En Rose", Duck for the space ship duck in GUNDAM 0083, named as its shape, like a rose blossom.

In [47]:
```
from IPython.display import display,HTML,YouTubeVideo
```

Gandum 0082

In [48]:
```
YouTubeVideo('p9mG149Nfrk')
```

Out[48]:



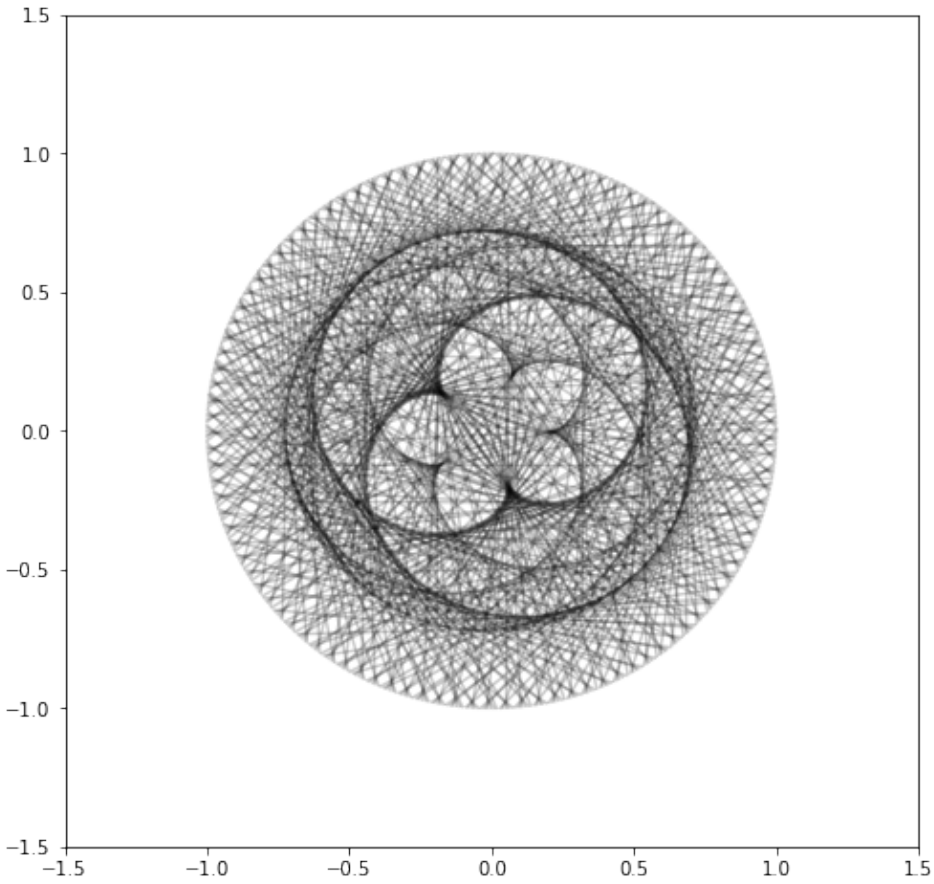
In [7]:
```
x1,x2,y1,y2=0,1,0,1
t1=365.256    # one year for Earth
t2=224.701    # one year for Venus
R1=149.6      # 1,000,000 km distance between Earth an
R2=108.21     # 1,000,000 km distance between Venus an
Ratio=R1/R2
```

In [8]:
```
from numpy import pi,cos,sin
def Cartesian(n, year=365.256, R=R1):
    nn=days*np.arange(0,n+1)
    theta= 2*pi/year;
    R=R/R1
    return np.array([R*cos(nn*theta),R*sin(nn*theta)]
```

In [9]:
```
days=7
n=600
# trajectory coordinates of Earth
Xs=Cartesian(n)
# trajectory coordinates of Venus
Ys=Cartesian(n,year=t2,R=R2)
```

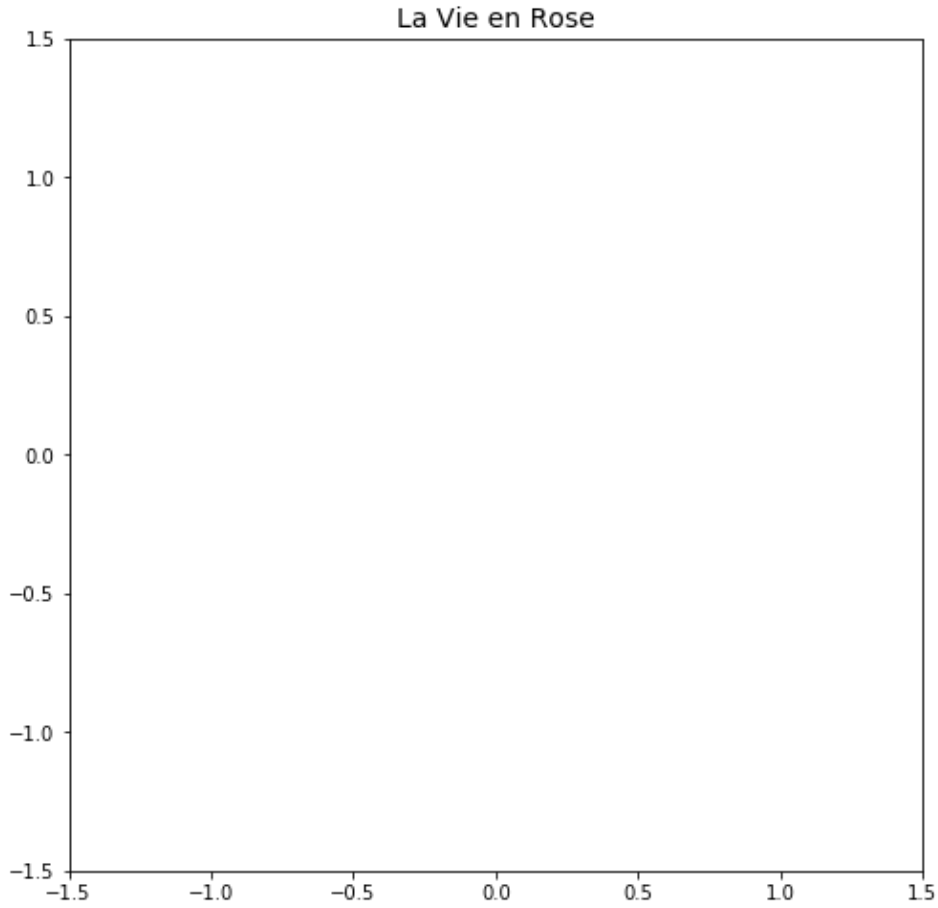
In [32]:

```
plt.figure(figsize=(8,8))
[plt.plot([Xs[0,i],Ys[0,i]],[Xs[1,i],Ys[1,i]],'k',alp
plt.xlim(-1.5,1.5)
plt.ylim(-1.5,1.5);
plt.savefig("imgs/planet.png")
```



In [33]:

```
from matplotlib import animation
from JSAnimation import IPython_display
from JSAnimation.IPython_display import display_anima
k=20
anim=plt.figure(figsize=(8,8));
ax = anim.add_subplot(111)
plt.title("La Vie en Rose",fontsize=14);

plt.ylim([-1.5,1.5]);
plt.xlim([-1.5,1.5]);
def init():
    #return  plt.plot([P[0],Q[0]],[P[1],Q[1]], color=
    return  plt.plot([Xs[0,0],Ys[0,0]],[Xs[1,0],Ys[1,
#PP=[[P[2*k],Q[2*k]] for k in np.arange(n)];
#QQ=[[P[2*k+1],Q[2*k+1]] for k in np.arange(n)];

PP=[[Xs[0,k],Ys[0,k]] for k in np.arange(n)];
QQ=[[Xs[1,k],Ys[1,k]] for k in np.arange(n)];
def animate(i):
    return [ax.plot(PP[k],QQ[k],color='0.2') for k in
```




```
anim=animation.FuncAnimation(anim, animate,
frames=120,interval=10)
display_animation(anim, default_mode='once')
```

In [40]:

```
k=20

PP=[[Xs[0,k],Ys[0,k]] for k in np.arange(n)];
QQ=[[Xs[1,k],Ys[1,k]] for k in np.arange(n)];

duration=20

fig, ax = plt.subplots(figsize=(8,8))

ax.set_title("La Vie en Rose",fontsize=14)
ax.set_aspect(aspect=1)
ax.set_xlim((-1.5,1.5));
ax.set_ylim((-1.5,1.5));
def animate(t):
    len=int(duration*t)
    for i in range(len):
        ax.plot(PP[i],QQ[i],color='0.2');
    return mplfig_to_npimage(fig)

animation = VideoClip(animate, duration=duration)
animation.ipython_display(fps=20, loop=True, autoplay
```

```
0%|          | 0/401 [00:00<?, ?it/s]
1%|          | 3/401 [00:00<00:14, 26.81it/s]
1%|▏         | 6/401 [00:00<00:14, 26.64it/s]
2%|▏         | 9/401 [00:00<00:15, 25.43it/s]
3%|▎         | 11/401 [00:00<00:16, 23.22it/s]
3%|▎         | 13/401 [00:00<00:18, 21.17it/s]
4%|▍         | 15/401 [00:00<00:19, 19.82it/s]
4%|▍         | 17/401 [00:00<00:20, 18.53it/s]
5%|▍         | 19/401 [00:01<00:29, 13.02it/s]
```

In [26]:
```
# save each connected trajector in png format
PP=[[Xs[0,k],Ys[0,k]] for k in np.arange(n)];
QQ=[[Xs[1,k],Ys[1,k]] for k in np.arange(n)];

fig, ax = plt.subplots(figsize=(8,8))

ax.set_title("La Vie en Rose",fontsize=14)
ax.set_aspect(aspect=1)
ax.set_xlim((-1.5,1.5));
ax.set_ylim((-1.5,1.5));
for i in range(len(PP)):
    ax.plot(PP[i],QQ[i],alpha=0.2);

    #name='{:03d}.png'.format(i)
    fig.savefig("images/{:03d}.png".format(i))
```

On the old days, JSAnimation made the data visualization work easily and directly; but comes with the Python progress itself, some libraries depressed and syntax had been changed, something does not work. Thus, more effective libray, *moviepy*, is led to be in place of the JSAnimation's work.

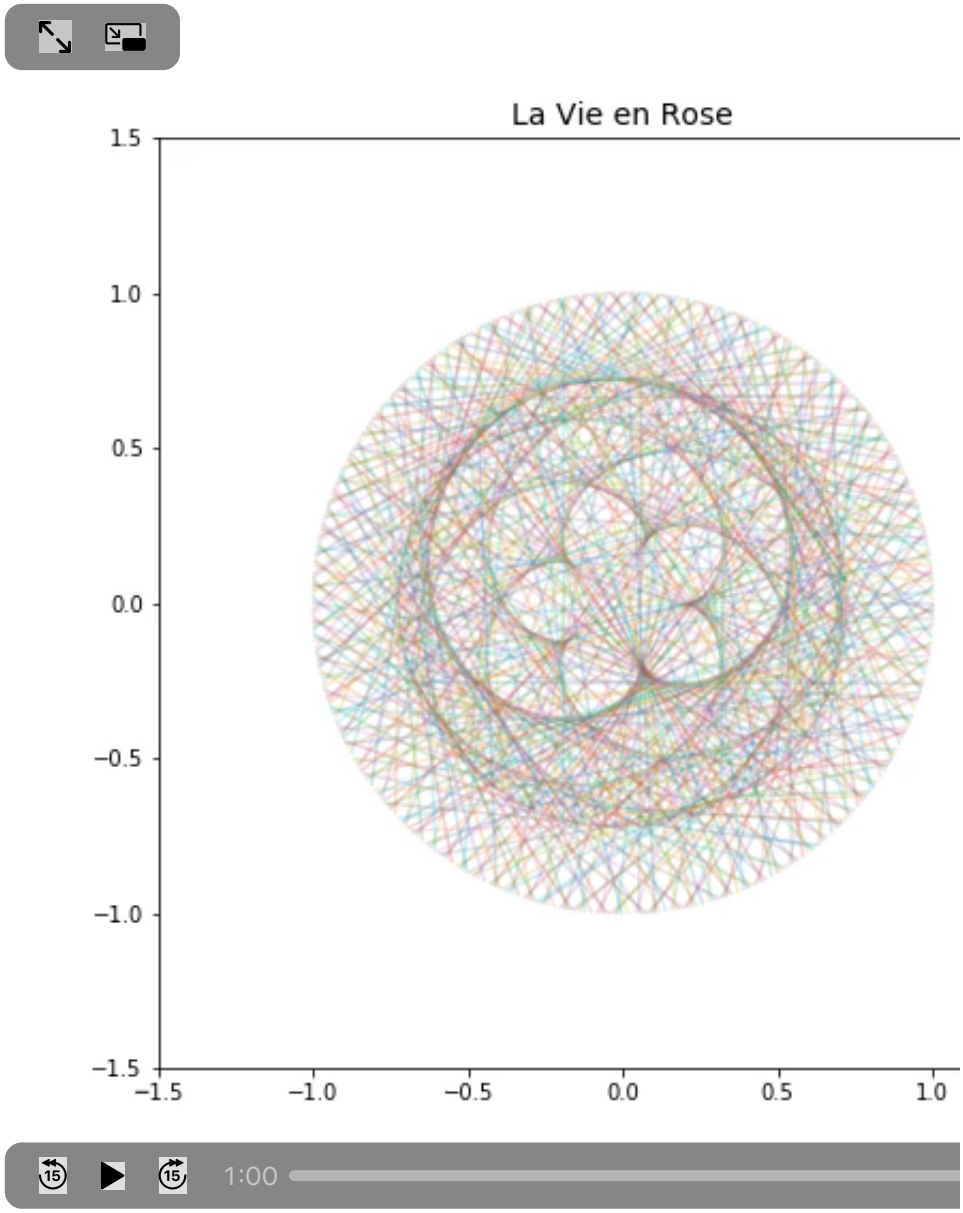
In [27]:

```
from moviepy.editor import *
import glob
directory="./images"
clips=[]
import numpy as np

file_list = np.sort(glob.glob('images/*.png'))
for file in file_list:
    clips.append(ImageClip(file).set_duration(0.1))
# compose all the png files
video = concatenate(clips, method="compose")
video.write_videofile('test1.mp4', fps=20)
```

```
[MoviePy] >>>> Building video test1.mp4
[MoviePy] Writing video test1.mp4

100%|██████████| 1201/1201 [00:34<00:00, 34.91it/s]

[MoviePy] Done.
[MoviePy] >>>> Video ready: test1.mp4
```

In [28]:
```
from IPython.display import Video

Video("test1.mp4")
```

Out[28]:



6 Sound (../2/Sinusoid.ipynb)

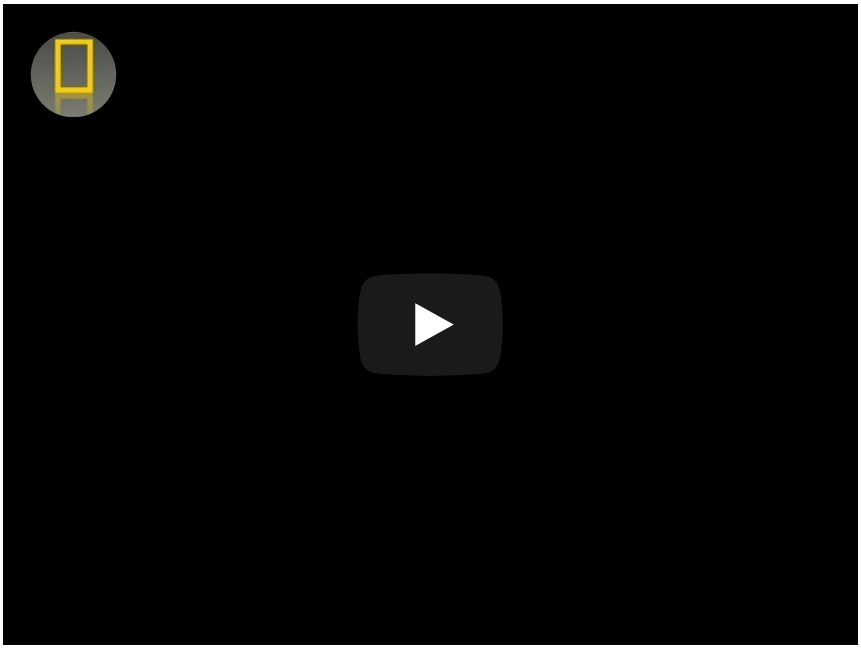
6.1 Recall of Nature

Befitting the largest mammal on earth, the call of an adult blue whale is loud (amptitude) and low (frequencies). The sound carries farther than any other animal sound, and can be detected over a thousand miles away. The call has a distinct pattern, which usually precedes a series of B *moans*.

In [49]:
```
from IPython.display import YouTubeVideo
```

In [50]:
```
YouTubeVideo('i8MTsgdWuU0')
```

Out[50]:



6.2 Pattern of Tones

The B call is simpler and easier to analyze, which consists of:

- a fundamental frequency around *16-17* hertz
- and a series of harmonics(multiples) of the fundamental frequency.

Here, the amplitude of the call is modulated to produce a loud moan followed by weaker moan.

6.3 Simulation of Sound

- The signal itself is treated as a sum of harmonics

$$y_0 = \sum_{n=0}^{\infty} \sin(2\pi n f_0 t)$$

- The envelope of the signal is treated as a decaying sine wave of the form

$$A = A_0 e^{-Bt} \sin(2\pi f_m t)$$

where the amplitude A_0, the decay rate B, and the modulating frequency f_m.

- Whale call:

$$A \cdot y_0$$

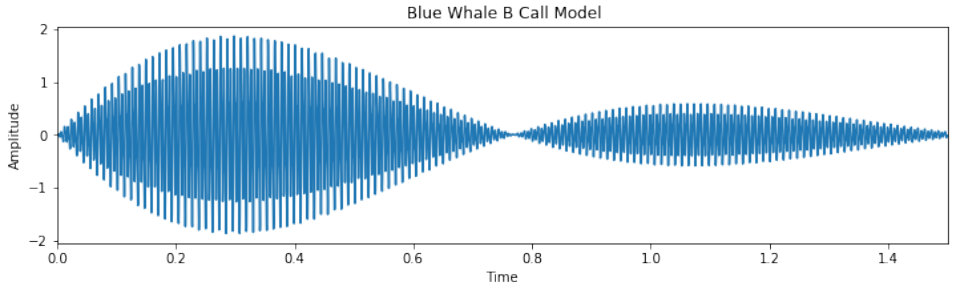
- Because blue whale calls are so low, they are barely audible to humans (within 20Hz to 20kHz). The time scale in the data being modeled is compressed by a factor of 10 to raise the pitch and make it more clearly audible. The model works with the same $\times 10$ audible frequencies.

In [51]:
```
%matplotlib inline

import numpy as np
from numpy import sin,pi,exp
import matplotlib.pylab as plt
from IPython.display import Audio, Image
```

In [52]:
```
fs = 4000;
xmax=1.5
t = np.linspace(0,xmax,fs);
# Set the fundamental frequency of the call.
f0 = 175;
# compose with harmonics.
y0 = sin(2*pi*f0*t) + 0.5*sin(2*pi*2*f0*t) + 0.5*sin(
# Set the additional parameters in the model.
A0 = 2; #Initial amplitude.
B = 1.5; # Amplitude decay rate.
fm = 0.65; # Frequency of the modulating envelope.
# Create the envelope

A = A0*exp(-B*t)*sin(2*pi*fm*t);
```

In [53]:
```
call = A*y0;
# Plot the model call and listen to it.
plt.figure(figsize=(12,3))
plt.plot(t,call)
plt.xlabel('Time');plt.ylabel('Amplitude');plt.xlim(0
plt.title(' Blue Whale B Call Model')

Audio(call.astype(np.int16),rate=fs)
```

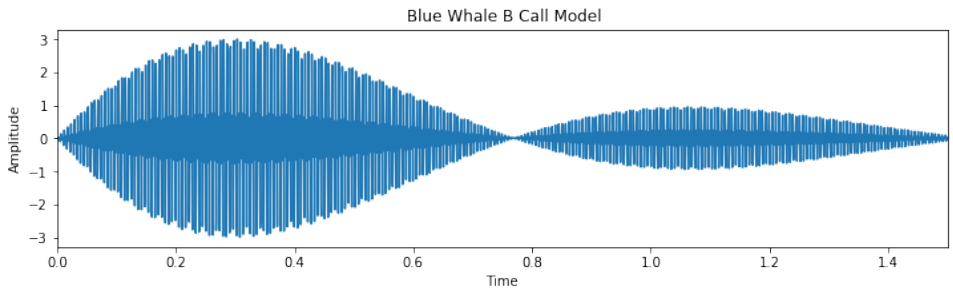
Out[53]:



In [54]:
```
fs = 4000;
xmax=1.5
t = np.linspace(0,xmax,fs);
# Set the fundamental frequency of the call.
f0 = 175;
# compose with harmonics.
y0 = sin(2*pi*f0*t) + sin(2*pi*2*f0*t) + sin(2*pi*3*f
# Set the additional parameters in the model.
A0 = 2; #Initial amplitude.
B = 1.5; # Amplitude decay rate.
fm = 0.65; # Frequency of the modulating envelope.
# Create the envelope

A = A0*exp(-B*t)*sin(2*pi*fm*t);
```

In [55]:
```
call = A*y0;
# Plot the model call and listen to it.
plt.figure(figsize=(12,3))
plt.plot(t,call)
plt.xlabel('Time');plt.ylabel('Amplitude');plt.xlim(0
plt.title(' Blue Whale B Call Model')

Audio(call.astype(np.int16),rate=fs)
```

Out[55]:



In [41]:
```
!jupyter nbconvert index.ipynb
```

```
[NbConvertApp] Converting notebook index.ipynb to htm
l
[NbConvertApp] Writing 5023064 bytes to index.html
```

In [ ]:

In [ ]: