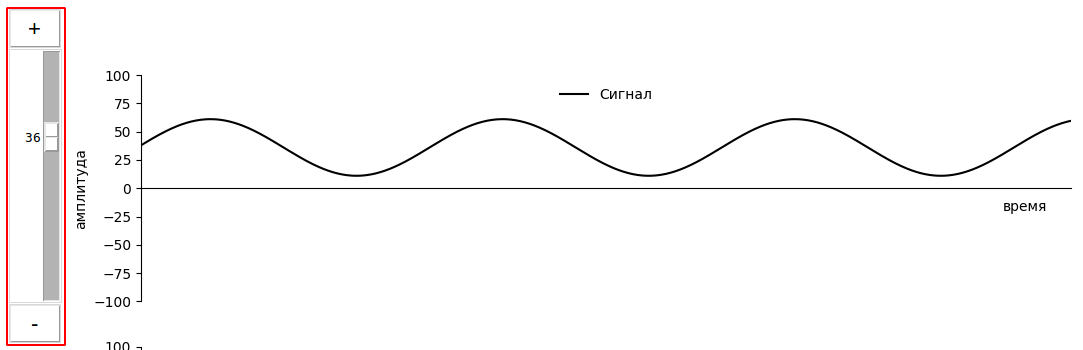
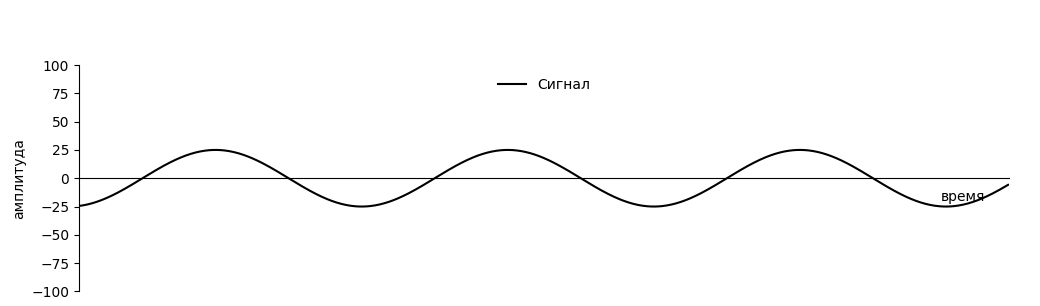
Документация по программе «Модуляция»

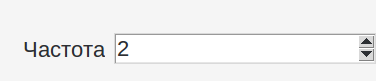
1. Данная программа является учебным пособием и предназначена для визуальной демонстрации различных типов модуляций с целью более глубокого понимания изучаемого материала.
2. Список типов модуляции представленных в программе:
   1. Амплитудная модуляция
   2. Частотная модуляция
   3. Фазовая модуляция
   4. Амплитудно-импульсная модуляция 1-го рода
   5. Амплитудно-импульсная модуляция 2-го рода
   6. Широтно-импульсная модуляция
   7. Фазово-импульсная модуляция
   8. Частотно-импульсная модуляция
   9. Импульсно-кодовая модуляция (в разработке)
   10. Дельта импульсно-кодовая модуляция (в разработке)
3. Элементы управления
   * + Слайдер, управляющий положением сигнала относительно оси времени, имеет значения от -100 до +100. Позволяет «поднимать» и «опускать» уровень графика сигнала.



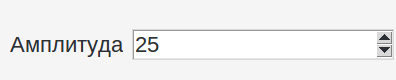
* + - Пространство отрисовки сигнала.



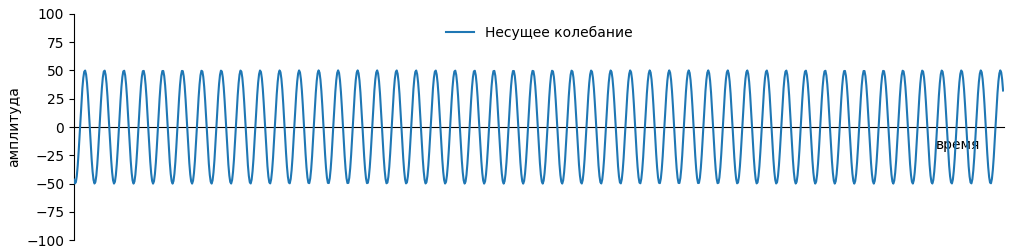
* + - Элемент управления частотой сигнала. Позволяет изменять частоту(расстояние между соотносящимися значениями функции) сигнала. Диапазон от 1 до 100.



* + - Элемент управления амплитудой сигнала. Позволяет изменять амплитуду (разность между минимальным и максимальным значением функции) сигнала. Диапазон от 5 до 95.



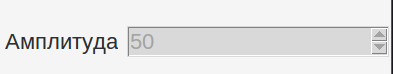
* + - Пространство отрисовки несущего колебания.



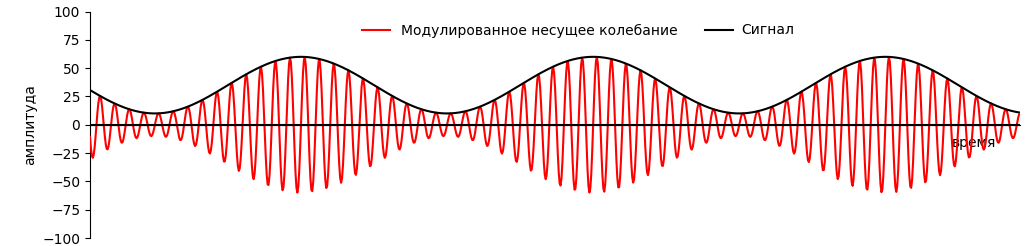
* + - Элемент управления частотой несущего колебания. Позволяет изменять частоту(расстояние между соотносящимися значениями функции) несущего колебания. Диапазон от 10 до 100.



* + - Элемент отображения амплитуды несущего колебания. Синхронизируется с амплитудой сигнала. Диапазон от 5 до 95.



* + - Пространство отрисовки промодулированного несущего колебания.



* + - Переключатель типа модуляции, доступные типы указаны выше.



* + - Поле ввода девиации. Девиация — это коэффициент который показывает степень влияния значения сигнала на несущее колебание при модуляции. Доступные значения от 0.0 и выше, дробные значения с точкой или запятой (4.345 3,675).



* + - Поле ввода ширины несущих импульсов. Характеризует процент расстояния между центрами импульсов, которое заполняют импульсы, т.е чем выше это значение, тем меньше расстояние между одиночными импульсами. Принимает значения от 0 до 100.



1. Листинг кода.

*#Imports*

*import* tkinter *as* tk

*from* tkinter *import* ttk

*import* time, sys, math

*import* numpy *as* np

*from* matplotlib.backends.backend\_tkagg *import* FigureCanvasTkAgg

*from* matplotlib.figure *import* Figure

*import* matplotlib.pyplot *as* plt

*import* matplotlib.animation *as* animation

*from* scipy *import* signal

*#TODO*

*# create new gui without notebook with embeded matplotlib+*

*# create function for changable variables+*

font\_ = ('Arial', 14)

background\_ = '#f5f5f5'

foreground\_ = '#2c2f33'

root = tk.Tk()

root.configure(background = background\_)

root.title('Модуляция')

*#root.geometry('1920x1080')*

root.resizable(1280, 720)

*#style*

style = ttk.Style()

style.theme\_create('mod\_theme', parent = 'alt', settings = {

'TCombobox': {'configure':{

'selectbackground': 'white',

'fieldbackground': 'white',

'selectforeground': foreground\_

}},

'TLabel': {'configure':{

'background': background\_,

'foreground': foreground\_

}}

})

style.theme\_use("mod\_theme")

def func(event=None):

*pass*

*#signal*

def signal\_freq\_change(event=None):

global s\_frq

s\_frq = int(signal\_freq.get())

def signal\_amp\_change(event=None):

global s\_amp, c\_amp, carry\_amp

s\_amp = int(signal\_amp.get())

*if* type\_modulation.current() == 0:

carry\_amp.set(s\_amp\*2)

c\_amp = s\_amp\*2

*elif* type\_modulation.current() >= 2:

carry\_amp.set(s\_amp)

c\_amp = s\_amp

*#carry*

def carry\_freq\_change(event=None):

global c\_frq

c\_frq = int(carry\_freq.get())

def carry\_amp\_change(event=None):

global c\_amp

c\_amp = int(carry\_amp.get())

def carry\_width\_change(event = None):

global c\_width

c\_width = int(carry\_width.get())/100

def change\_modulation(event=None):

global deviation\_lbl

global deviation\_input

global s\_amp, c\_amp, carry\_amp, txt

types = ['Амплитудная модуляция', 'Частотная модуляция', 'Фазовая модуляция',

'Амплитудно-импульсная модуляция 1 рода', 'Амплитудно-импульсная модуляция 2 рода',

'Широтно-импульсная модуляция', 'Фазово-импульсная модуляция', 'Частотно-импульсная модуляция',

'Импульсно-кодовая модуляция', 'Дельта импульсно-кодовая модуляция']

type\_text.set(types[type\_modulation.current()])

txt = 'klfbalksdvbajsdbc'

*if* type\_modulation.current() == 0:

carry\_amp.set(s\_amp\*2)

c\_amp = s\_amp\*2

carry\_amp\_box.configure(state = 'disable' )

*#deviation\_lbl.grid\_remove()*

*#deviation\_input.grid\_remove()*

deviation\_lbl.place\_forget()

deviation\_input.place\_forget()

carry\_width\_lbl.place\_forget()

carry\_width\_box.place\_forget()

carry\_width\_percent\_lbl.place\_forget()

*elif* type\_modulation.current() == 1 or type\_modulation.current() == 2:

carry\_amp.set(s\_amp)

c\_amp = s\_amp

carry\_amp\_box.configure(state = 'disable' )

*#deviation\_lbl.grid(row = 7, column = 1, sticky = ('E'), padx = (40, 5), pady = 5)*

*#deviation\_input.grid(row = 7, column = 2, sticky = ('E', 'W'), padx = 2, pady = 5)*

deviation\_lbl.place(x = 1270, y = 900)

deviation\_input.place(x = 1380, y = 900)

carry\_width\_lbl.place\_forget()

carry\_width\_box.place\_forget()

carry\_width\_percent\_lbl.place\_forget()

*elif* type\_modulation.current() >= 3 and type\_modulation.current() < 6:

carry\_amp.set(s\_amp)

c\_amp = s\_amp

carry\_amp\_box.configure(state = 'disable' )

deviation\_lbl.place\_forget()

deviation\_input.place\_forget()

carry\_width\_lbl.place(x = 1280, y = 625)

carry\_width\_box.place(x = 1380, y = 625)

carry\_width\_percent\_lbl.place(x = 1480, y = 625)

*elif* type\_modulation.current() >= 6:

carry\_amp.set(s\_amp)

c\_amp = s\_amp

carry\_amp\_box.configure(state = 'disable' )

deviation\_lbl.place\_forget()

deviation\_input.place\_forget()

carry\_width\_lbl.place(x = 1280, y = 625)

carry\_width\_box.place(x = 1380, y = 625)

carry\_width\_percent\_lbl.place(x = 1480, y = 625)

deviation\_lbl.place(x = 1270, y = 900)

deviation\_input.place(x = 1380, y = 900)

def scale\_up(event=None):

prev = signal\_lvl.get()

signal\_lvl.set(prev+1)

def scale\_down(event=None):

prev = signal\_lvl.get()

signal\_lvl.set(prev-1)

plot\_container = tk.LabelFrame(root, text = '', height = 960, width = 1280, font = font\_, bg = 'white')

plot\_container.grid(row = 0, column = 0, rowspan = 9, padx = 5, pady = 5)

plot\_container.configure(background = background\_, foreground = foreground\_)

*#main label*

type\_text = tk.StringVar()

type\_text.set('Амплитудная модуляция')

type\_label = ttk.Label(plot\_container, textvariable = type\_text, font = font\_, background = background\_)

type\_label.grid(row = 0, column = 0, sticky = ('N'), padx = 2, pady = 5)

*#vert scale*

signal\_lvl = tk.IntVar()

signal\_scale = tk.Scale(root, from\_ = 100, to = -100, bg = 'white', length = 250, variable = signal\_lvl)

signal\_scale.place(x = 25, y = 140)

up\_button = tk.Button(root, text = '+', command = scale\_up, font = font\_, bg = 'white', width = 2)

up\_button.place(x = 25, y = 100)

down\_button = tk.Button(root, text = '-', command = scale\_down, font = font\_, bg = 'white', width = 2)

down\_button.place(x = 25, y = 395)

*#Signal frequency*

signal\_freq\_lbl = ttk.Label(root, text = 'Частота', font = font\_)

signal\_freq\_lbl.grid(row = 0, column = 1, sticky = ('E', 'S'), padx = (40, 5), pady = 5)

signal\_freq = tk.IntVar()

signal\_freq.set(2)

signal\_freq\_box = tk.Spinbox(root, from\_ = 1, to = 100, textvariable = signal\_freq, font = font\_, foreground = foreground\_, command = signal\_freq\_change)

signal\_freq\_box.grid(row = 0, column = 2, sticky = ('E', 'W', 'S'), padx = 2, pady = 5)

signal\_freq\_box.bind('<Return>', func)

*#Signal amplitude*

signal\_amp\_lbl = ttk.Label(root, text = 'Амплитуда', font = font\_)

signal\_amp\_lbl.grid(row = 1, column = 1, sticky = 'E', padx = (40, 5))

signal\_amp = tk.IntVar()

signal\_amp.set(25)

signal\_amp\_box = tk.Spinbox(root, from\_ = 5, to = 95, textvariable = signal\_amp, font = font\_, foreground = foreground\_, command = signal\_amp\_change, increment = 5.0)

signal\_amp\_box.grid(row = 1, column = 2, sticky = ('E', 'W'), padx = 2, pady = 5)

signal\_amp\_box.bind('<Return>', func)

*#Carrying frequency*

carry\_freq\_lbl = ttk.Label(root, text = 'Частота', font = font\_)

carry\_freq\_lbl.grid(row = 3, column = 1, sticky = ('E', 'S'), padx = (40, 5), pady = 5)

carry\_freq = tk.IntVar()

carry\_freq.set(40)

carry\_freq\_box = tk.Spinbox(root, from\_ = 10, to = 100, textvariable = carry\_freq, font = font\_, foreground = foreground\_, command = carry\_freq\_change, increment = 5.0)

carry\_freq\_box.grid(row = 3, column = 2, sticky = ('E', 'W', 'S'), padx = 2, pady = 5)

carry\_freq\_box.bind('<Return>', carry\_freq\_change)

*#Carrying amplitude*

carry\_amp\_lbl = ttk.Label(root, text = 'Амплитуда', font = font\_)

carry\_amp\_lbl.grid(row = 4, column = 1, sticky = ('E'), padx = (40, 5), pady = 5)

carry\_amp = tk.IntVar()

carry\_amp.set(50)

carry\_amp\_box = tk.Spinbox(root, from\_ = 5, to = 100, textvariable = carry\_amp, font = font\_, foreground = foreground\_, command = carry\_amp\_change, increment = 5.0, state = 'disable')

carry\_amp\_box.grid(row = 4, column = 2, sticky = ('E', 'W'), padx = 2, pady = 5)

carry\_amp\_box.bind('<Return>', func)

*#Carrying impulse width*

carry\_width\_lbl = ttk.Label(root, text = 'Ширина', font = font\_)

carry\_width\_percent\_lbl = ttk.Label(root, text = '%', font = font\_)

carry\_width = tk.IntVar()

carry\_width\_box = tk.Spinbox(root, from\_ = 1, to = 100, textvariable = carry\_width, font = font\_, foreground = foreground\_, command = carry\_width\_change, increment = 5.0, width = 5)

carry\_width.set(50)

*#Type of modulation switcher*

type\_m\_lbl = ttk.Label(root, text = 'Тип:', font = font\_)

type\_m\_lbl.grid(row = 6, column = 1, sticky = ('E', 'S'), padx = (40, 5), pady = 5)

type\_m = tk.StringVar()

type\_modulation = ttk.Combobox(root, state = 'readonly', textvariable = type\_m , font = font\_)

type\_modulation.grid(row = 6, column = 2, sticky = ('E', 'W', 'S'), padx = 2, pady = 5)

type\_modulation['values'] = ('АМ', 'ЧМ', 'ФМ', 'АИМ 1 рода', 'АИМ 2 рода', 'ШИМ', 'ФИМ', 'ЧИМ', 'ИКМ', 'ДИКМ')

type\_modulation.current(0)

root.option\_add('\*TCombobox\*Listbox.font', font\_)

type\_modulation.bind('<<ComboboxSelected>>', change\_modulation)

deviation = tk.StringVar()

deviation\_lbl = ttk.Label(root, text = 'Девиация', font = font\_)

deviation\_input = ttk.Entry(root, textvariable = deviation, font = font\_)

*#variables*

s\_frq = int(signal\_freq.get())

s\_amp = int(signal\_amp.get())

c\_frq = int(carry\_freq.get())

c\_amp = int(carry\_amp.get())

c\_width = int(carry\_width.get())/100

*#plotting*

fig = plt.Figure(figsize=(12, 10))

x = np.arange(0, 10, 0.01)

def s\_ani(i):

*if* type\_modulation.current() != 0 and type\_modulation.current() < 3:

s\_line.set\_ydata(np.sin((x+i/50.0)\*s\_frq)\*s\_amp)

*else*:

y\_signal,\_,\_\_ = calc\_mod\_ani(i)

s\_line.set\_ydata(y\_signal)

*#s\_line.set\_ydata(np.sin((x+i/50.0)\*s\_frq)\*s\_amp+ s\_amp)*

*if* type\_modulation.current() != 6 and type\_modulation.current() != 7:

s\_line.set\_ydata(np.sin((x+i/50.0)\*s\_frq)\*s\_amp+ signal\_lvl.get())

*else*:

s\_line.set\_ydata(np.sin((x+i/50.0)\*s\_frq)\*s\_amp+s\_amp)

*return* s\_line,

def c\_ani(i):

*if* type\_modulation.current() <= 2:

c\_line.set\_ydata(np.sin((x+i/50.0)\*c\_frq)\*c\_amp)

*else*:

y\_carry = signal.square((x+i/50.0)\*c\_frq, duty=c\_width)\*c\_amp

y\_max = max(y\_carry)

c\_line.set\_ydata(y\_carry+y\_max)

*return* c\_line,

def m\_ani(i):

y\_signal, y\_carry, label = calc\_mod\_ani(i)

cm\_line.set\_ydata(y\_carry)

sm\_line.set\_ydata(y\_signal)

sm\_line.set\_color('k')

spm\_line.set\_ydata(0)

*if* type\_modulation.current() == 2:

spm\_line.set\_ydata(np.sin((x+i/50.0)\*s\_frq)\*s\_amp)

*elif* type\_modulation.current() == 6:

spm\_line.set\_ydata(np.sin((x+i/50.0)\*s\_frq)\*s\_amp+s\_amp)

subsignal\_label.set\_text(label)

*if* type\_modulation.current() == 2 or type\_modulation.current() == 6:

sm\_line.set\_color('#1f77b4')

*return* cm\_line, sm\_line, spm\_line, subsignal\_label

def calc\_mod\_ani(i):

*#AM*

*if* type\_modulation.current() == 0:

y\_signal = np.sin((x+i/50.0)\*s\_frq)\*s\_amp

y\_signal = (np.sin((x+i/50.0)\*s\_frq)\*s\_amp)+signal\_lvl.get()

y\_carry = np.sin((x+i/50.0)\*c\_frq)\*c\_amp\*(y\_signal/(s\_amp\*2))

label = 'Сигнал'

*return* y\_signal, y\_carry, label

*#FM*

*elif* type\_modulation.current() == 1:

y\_signal = np.sin((x+i/50.0)\*s\_frq)\*s\_amp

*#dev = 14/y\_max*

dev = 1

*if* not deviation.get().isalpha() and deviation.get() != '':

dev = deviation.get().replace(',', '.')

dev = float(dev)

y\_carry = np.sin((x+i/50.0)\*c\_frq - dev\*np.cos((x+i/50.0)\*s\_frq))\*c\_amp

label = 'Сигнал'

*return* y\_signal, y\_carry, label

*#PM*

*elif* type\_modulation.current() == 2:

dev = 0.1

y\_signal = np.sin((x+i/50.0)\*s\_frq)\*s\_amp

*if* not deviation.get().isalpha() and deviation.get() != '':

dev = deviation.get().replace(',', '.')

dev = float(dev)/10

y\_old\_carry = np.sin((x+i/50.0)\*c\_frq)\*c\_amp

y\_carry = np.sin((x+i/50.0)\*c\_frq - dev\*y\_signal)\*c\_amp

*#label = 'Несущее колебание'*

label = 'Сигнал'

*return* y\_old\_carry, y\_carry, label

*#PAM\_1*

*elif* type\_modulation.current() == 3:

y\_max = s\_amp

y\_signal = (np.sin((x+i/50.0)\*s\_frq)\*s\_amp)+signal\_lvl.get()

y\_carry = signal.square((x+i/50.0)\*c\_frq, duty=c\_width)\*c\_amp+y\_max

y\_carry = y\_signal\*y\_carry/(y\_max\*2)

label = 'Сигнал'

*return* y\_signal, y\_carry, label

*#PAM\_2*

*elif* type\_modulation.current() == 4:

y\_max = s\_amp

y\_signal = (np.sin((x+i/50.0)\*s\_frq)\*s\_amp)+signal\_lvl.get()

y\_carry = signal.square((x+i/50.0)\*c\_frq, duty=c\_width)\*c\_amp+y\_max

y\_carry = average(y\_signal\*y\_carry/(y\_max\*2))

label = 'Сигнал'

*return* y\_signal, y\_carry, label

*#PWM*

*elif* type\_modulation.current() == 5:

y\_signal = np.sin((x+i/50.0)\*s\_frq)\*s\_amp

y\_max = s\_amp

y\_signal = (np.sin((x+i/50.0)\*s\_frq)\*s\_amp)+signal\_lvl.get()

y\_carry = signal.square((x+i/50.0)\*c\_frq, duty=1.5\*c\_width\*y\_signal/(y\_max\*2)+0.08)\*c\_amp+ y\_max

label = 'Сигнал'

*return* y\_signal, y\_carry, label

*#PPM*

*elif* type\_modulation.current() == 6:

inc = i/50

*#inc2 = int(1\*i/np.pi)*

y\_max = s\_amp

y\_signal\_changed = (np.sin((x+inc)\*s\_frq)\*s\_amp)

y\_old\_carry = signal.square((x+inc)\*c\_frq, duty=c\_width)\*c\_amp+ y\_max

y\_generated = signal.square((x+i/50.0)\*c\_frq, duty=c\_width)\*c\_amp+y\_max

y\_carry = np.sin(x)

y\_carry[:] = 0

m\_width = int(np.pi\*200 \*c\_width/c\_frq)

prev = 0

dev = 1

*if* not deviation.get().isalpha() and deviation.get() != '':

dev = deviation.get().replace(',', '.')

dev = float(dev)

*for* i, a in enumerate(y\_generated):

*if* a == 2\*y\_max and prev == 0:

y\_carry[i+int(dev\*y\_signal\_changed[i]):i+int(dev\*y\_signal\_changed[i]) +m\_width] = 2\*y\_max

prev = a

label = 'Несущее колебание'

*return* y\_old\_carry, y\_carry, label

*#PFM repair it*

*elif* type\_modulation.current() == 7:

inc = i/50

*#inc2 = int(1\*i/np.pi)*

y\_max = s\_amp

y\_signal\_changed = (np.sin((x+inc+np.pi/4)\*s\_frq)\*s\_amp)

y\_generated = signal.square((x+i/50.0)\*c\_frq, duty=c\_width)\*c\_amp+y\_max

y\_carry = np.sin(x)

y\_carry[:] = 0

m\_width = int(np.pi\*200 \*c\_width/c\_frq)

prev = 0

dev = 1

*if* not deviation.get().isalpha() and deviation.get() != '':

dev = deviation.get().replace(',', '.')

dev = float(dev)

*for* i, a in enumerate(y\_generated):

*if* a == 2\*y\_max and prev == 0:

y\_carry[i+int(dev\*y\_signal\_changed[i]):i+int(dev\*y\_signal\_changed[i]) +m\_width] = 2\*y\_max

prev = a

label = 'Сигнал'

y\_signal = (np.sin((x+inc)\*s\_frq)\*s\_amp)+y\_max

*return* y\_signal, y\_carry, label

def average(y\_mod):

s, counter, prev = 0, 0, 0

flag = False

*for* i, a in enumerate(y\_mod):

*if* a > 0:

flag = True

s += a

counter += 1

*else*:

*if* flag:

y\_mod[prev:i] = s/counter

s, counter = 0, 0

flag = False

prev = i+1

*return* y\_mod

def normalize(ax):

ax.set\_xlim(0, 10)

ax.set\_ylim(-100, 100)

ax.tick\_params(axis='x', which='both', bottom=False, top=False, labelbottom=False)

*#ax.tick\_params(axis='y', which='both', right=False, left=False, labelleft=False)*

ax.spines['bottom'].set\_position('center')

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

ax.xaxis.set\_label\_coords(0.95, 0.45)

ax.set\_xlabel('время')

ax.set\_ylabel('амплитуда')

canvas = FigureCanvasTkAgg(fig, master=plot\_container)

canvas.get\_tk\_widget().grid(row=1,column=0)

s\_ax = fig.add\_subplot(311)

c\_ax = fig.add\_subplot(312)

m\_ax = fig.add\_subplot(313)

root.update()

y\_signal = np.sin((x)\*s\_frq)\*s\_amp

y\_max = max(y\_signal)

y\_carry = np.sin((x)\*c\_frq)\*c\_amp

s\_line, = s\_ax.plot(x, y\_signal, 'k')

c\_line, = c\_ax.plot(x, y\_carry)

y\_signal = (np.sin((x)\*s\_frq)\*s\_amp)+y\_max

y\_max = max(y\_signal)

y\_carry = y\_carry\*(y\_signal/y\_max)

cm\_line, = m\_ax.plot(x, y\_carry, 'r')

spm\_line, = m\_ax.plot(x, y\_signal, 'k')

sm\_line, = m\_ax.plot(x, y\_signal, 'k')

s\_ax.cla()

c\_ax.cla()

m\_ax.cla()

normalize(s\_ax)

normalize(c\_ax)

normalize(m\_ax)

s\_ax.legend([s\_line], ['Сигнал'], loc = 'upper center', frameon=False)

c\_ax.legend([c\_line], ['Несущее колебание'], loc = 'upper center', frameon=False)

m\_ax.legend([cm\_line, sm\_line], ['Модулированное несущее колебание', ' '], loc = 'upper center', frameon=False, ncol=2)

subsignal\_label = m\_ax.text(0.7, 0.90, '', transform=m\_ax.transAxes)

*#c\_repeatable\_point = int(250 \* c\_frq / 10)*

a1 = animation.FuncAnimation(fig, s\_ani, np.arange(1, 315), interval=20, blit=True)

a2 = animation.FuncAnimation(fig, c\_ani, np.arange(1, 127), interval=20, blit=True)

a3 = animation.FuncAnimation(fig, m\_ani, np.arange(1, 315), interval=20, blit=True)

root.mainloop()