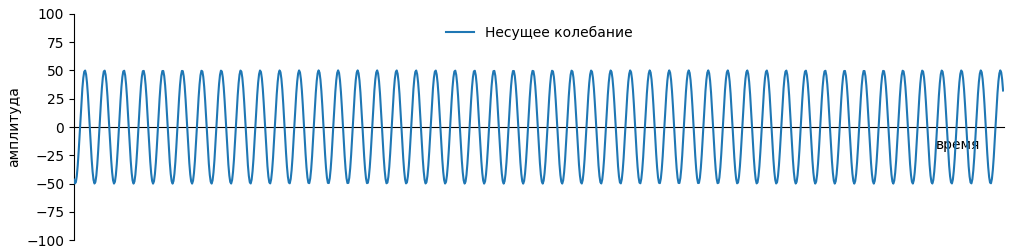
Документация по программе «Манипуляция»

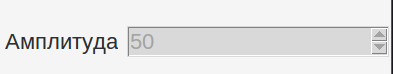
1. Данная программа является учебным пособием и предназначена для визуальной демонстрации различных типов модуляций с целью более глубокого понимания изучаемого материала.
2. Список типов модуляции представленных в программе:
   1. Амплитудная манипуляция
   2. Частотная манипуляция
   3. Фазовая манипуляция
   4. Относительно-фазовая манипуляция
3. Элементы управления
   * + Пространство отрисовки дискретного сигнала.
     + 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 отображение несущего колебания. Неизменяемое значение в 50 единиц.



* + - Пространство отрисовки манипулированного несущего колебания.
    - Переключатель типа манипуляции, доступные типы указаны выше.

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

font\_ = ('Arial', 14)

background\_ = '#f5f5f5'

foreground\_ = '#2c2f33'

root = tk.Tk()

root.configure(background = background\_)

root.title('Манипуляция')

*#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")

*'''#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 change\_manipulation(event = None):

types = ['Амплитудная манипуляция', 'Частотная манипуляция', 'Фазовая манипуляция', 'Относительно-фазовая манипуляция',]

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

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

*pass*

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

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 манипуляция')

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)

*#Signal parameters*

*#NOT TODAY!!!*

*#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(10)

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 = 300, 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)*

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

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

type\_manipulation['values'] = ('АМт', 'ЧМт', 'ФМт', 'ОФМт')

type\_manipulation.current(0)

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

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

*#variables*

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

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

s\_amp = 25

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

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

*#plotting*

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

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

def discr\_signal\_gen(i):

global bit\_0

*#inc = int(i/1.5712)*

inc = i/20

inc2 = int(5\*i/np.pi)

*#inc2 = int(2\*i)*

*if* 300-inc2 < -1000:

inc2 -= 2000

*if* 300-inc2 > 0:

y\_signal = signal.square((np.pi\*x+inc + 0 )\*1)\*s\_amp + s\_amp

y\_signal[300-inc2:] = signal.square((np.pi\*x[300-inc2:]+inc + np.pi)\*1)\*s\_amp + s\_amp

*else*:

y\_signal = signal.square((np.pi\*x+inc + np.pi )\*1)\*s\_amp + s\_amp

y\_signal[300-inc2:] = signal.square((np.pi\*x[300-inc2:]+inc + 0)\*1)\*s\_amp + s\_amp

*'''position = 949 #629*

*s\_frq = 5*

*phi = 0*

*if position-inc2 < -1000:*

*inc2 -= 2000*

*if i == 815:*

*phi = -10*

*if position-inc2 > 0:*

*y\_signal = signal.square((np.pi\*x+inc)\*s\_frq)\*s\_amp + s\_amp*

*y\_signal[position-inc2:] = signal.square((x[position-inc2:]+inc + np.pi)\*s\_frq)\*s\_amp + s\_amp*

*else:*

*y\_signal = signal.square((x+inc + np.pi)\*s\_frq)\*s\_amp + s\_amp*

*y\_signal[position-inc2:] = signal.square((x[position-inc2:]+inc - 4.9)\*s\_frq)\*s\_amp + s\_amp'''*

*#print(i, 300-inc2)*

*return* y\_signal

def s\_ani(i):

inc2 = int(5\*i/np.pi)

y\_signal = discr\_signal\_gen(i)

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

bit\_0.set\_x((1 - 0.965 - inc2/1000)%1)

bit\_0.set\_text(str(1 *if* y\_signal[(1000-965-inc2)%1000]>0 *else* 0))

bit\_1.set\_x((1 - 0.865 - inc2/1000)%1)

bit\_1.set\_text(str(1 *if* y\_signal[(1000-865-inc2)%1000]>0 *else* 0))

bit\_2.set\_x((1 - 0.765 - inc2/1000)%1)

bit\_2.set\_text(str(1 *if* y\_signal[(1000-765-inc2)%1000]>0 *else* 0))

bit\_3.set\_x((1 - 0.665 - inc2/1000)%1)

bit\_3.set\_text(str(1 *if* y\_signal[(1000-665-inc2)%1000]>0 *else* 0))

bit\_4.set\_x((1 - 0.565 - inc2/1000)%1)

bit\_4.set\_text(str(1 *if* y\_signal[(1000-565-inc2)%1000]>0 *else* 0))

bit\_5.set\_x((1 - 0.465 - inc2/1000)%1)

bit\_5.set\_text(str(1 *if* y\_signal[(1000-465-inc2)%1000]>0 *else* 0))

bit\_6.set\_x((1 - 0.365 - inc2/1000)%1)

bit\_6.set\_text(str(1 *if* y\_signal[(1000-365-inc2)%1000]>0 *else* 0))

bit\_7.set\_x((1 - 0.265 - inc2/1000)%1)

bit\_7.set\_text(str(1 *if* y\_signal[(1000-265-inc2)%1000]>0 *else* 0))

bit\_8.set\_x((1 - 0.165 - inc2/1000)%1)

bit\_8.set\_text(str(1 *if* y\_signal[(1000-165-inc2)%1000]>0 *else* 0))

bit\_9.set\_x((1 - 0.065 - inc2/1000)%1)

bit\_9.set\_text(str(1 *if* y\_signal[(1000-65-inc2)%1000]>0 *else* 0))

*return* s\_line, bit\_0, bit\_1, bit\_2, bit\_3, bit\_4, bit\_5, bit\_6, bit\_7, bit\_8, bit\_9

def c\_ani(i):

inc = i/(10\*np.pi)

inc = i/20

c\_line.set\_ydata(np.sin((np.pi\*x+inc)\*c\_frq/2)\*c\_amp)

*return* c\_line,

def m\_ani(i):

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

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

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

*return* cm\_line, sm\_line,

def calc\_mod\_ani(i):

inc = i/(10\*np.pi)

inc = i/20

*#ASK*

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

y\_signal = discr\_signal\_gen(i)

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

*return* y\_signal, y\_carry

*#FSK*

*if* type\_manipulation.current() == 1:

y\_signal = discr\_signal\_gen(i)

y\_carry = np.sin((np.pi\*x+inc)\*(c\_frq/2+8\*(y\_signal/(2\*s\_amp))))\*c\_amp

*return* y\_signal, y\_carry

*#PSK*

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

y\_signal = discr\_signal\_gen(i)

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

*return* y\_signal, y\_carry

*#OPSK*

*if* type\_manipulation.current() == 3:

inc2 = int(5\*i/np.pi)

y\_signal = discr\_signal\_gen(i)

*if* 300-inc2 < -1000:

inc2 -= 2000

*if* 300-inc2 > 0:

mask\_signal = signal.square((np.pi\*x+inc + 0 )\*0.5)\*s\_amp + s\_amp

mask\_signal[300-inc2:] = signal.square((np.pi\*x[300-inc2:]+inc +np.pi)\*0.5)\*s\_amp + s\_amp

*else*:

mask\_signal = signal.square((np.pi\*x+inc + np.pi )\*0.5)\*s\_amp + s\_amp

mask\_signal[300-inc2:] = signal.square((np.pi\*x[300-inc2:]+inc + 0)\*0.5)\*s\_amp + s\_amp

y\_carry = np.sin((np.pi\*x+inc + np.pi\*(mask\_signal/(2\*s\_amp)))\*(c\_frq/2))\*c\_amp

*#print(i, inc2, 300-inc2)*

*if* y\_signal[0] == 0:

*#sys.pause()*

*pass*

*return* y\_signal, y\_carry

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))

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 = discr\_signal\_gen(0)

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

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

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

bit\_0 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_1 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_2 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_3 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_4 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_5 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_6 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_7 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_8 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

bit\_9 = s\_ax.text(0, 0.6, '', transform = s\_ax.transAxes, fontsize = 18)

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', 'Сигнал'], loc = 'upper center', frameon=False, ncol=2)

ani1 = animation.FuncAnimation(fig, s\_ani, np.arange(1, 1258), interval=20, blit=True)

ani2 = animation.FuncAnimation(fig, c\_ani, np.arange(1, 127), interval=20, blit=True) *#if speed = 10.0 => range = 125, interval =20*

ani3 = animation.FuncAnimation(fig, m\_ani, np.arange(1, 1258), interval=20, blit=True)

*'''import inspect*

*print(inspect.getmembers(bit\_0, predicate=inspect.ismethod))'''*

root.mainloop()