Exercises

1 FitzHugh-Nagumo equation

Calculate the FitzHugh-Nagumo equation with $I_{input} = 1.5$. Try also with $I_{input} = 0$.

$$\frac{dV}{dt} = 10\left(V - \frac{V^3}{3} - R + I_{input}\right)$$
$$\frac{dR}{dt} = 0.8(-R + 1.25V + 1.5)$$

Draw the V-R plot.

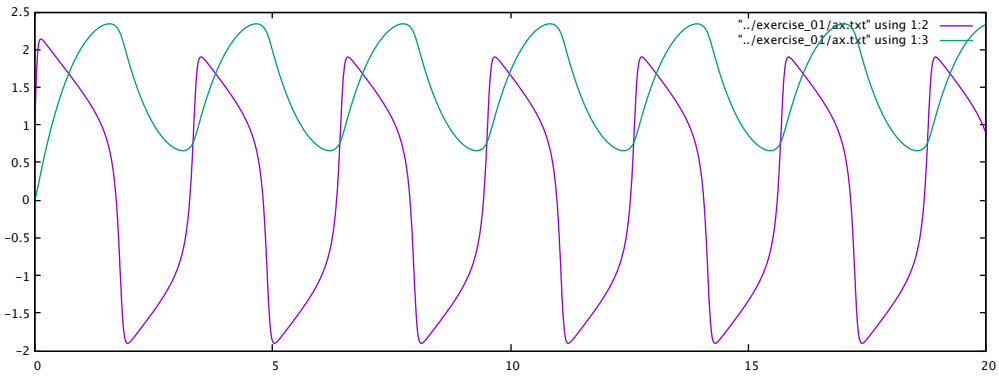
✎ Answer: Modify `ex07` or `ex07a`

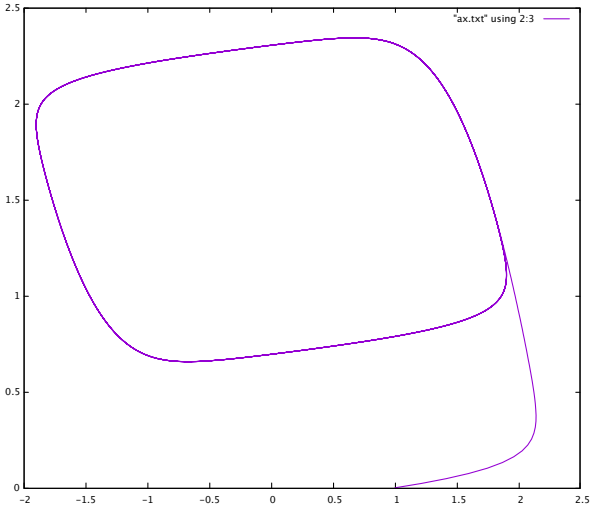
```cpp
//  main.cpp of ode (ex07a)
#include <iostream>
#include <fstream>
#include "Eqs.h"

int main(int argc, char** argv){
    int ndim = 2;           // number of equations
    double ttime = 20.0;    // total time
    double dt = 0.01;       // step size
    int npts = (int)(ttime/dt)+1;
    vector<double>  y={1.0, 0.0};

    Eqs ce;
    ce.initRK4(ndim, dt);

    std::fstream fd("ax.txt", std::ios::out | std::ios::trunc);
    for(int i=0;i<npts;i++){
        double t = dt*i;
        fd << t << " " << y[0] << " " << y[1] << std::endl;
        ce.advanceStep(t, y);
    }
    fd.close();
#if defined(_WIN32)
    system("..\\tools\\gc ax.txt ");
#else
    system("../tools/gc ax.txt ");
#endif
    return 0;
}
```

```cpp
// Eqs.h
#ifndef __included_eqs_h__
#define __included_eqs_h__

#include <vector>
#include "RungeKutta4.h"

using std::vector;

class Eqs : public RungeKutta4
{
public:
```

```
    void rhs(double tt, vector<double> y, vector<double>& dy);
private:
};
#endif
```

```
// Eqs.cpp
#include <cmath>
#include "Eqs.h"
void Eqs::rhs(double tt, vector<double> y, vector<double>& dy){
    double input = 1.5;
    dy[0] = 10.0*(y[0] - pow(y[0],3.0)/3.0 - y[1] + input );
    dy[1] = 0.8*(-y[1] + 1.25*y[0] + 1.5);
}
```



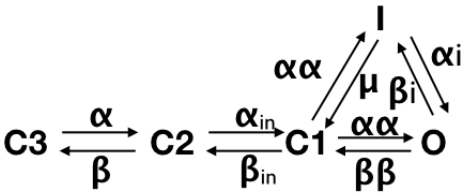


V-R plot.

2 HERG channel simulation

Try voltage-clamp simulation of HERG potassium channel.

A Markov model of the HERG channel is described by Clancy & Rudy. [*1]



$$
\begin{aligned}
I_{Kr} &= G_{Kr} \cdot P(\mathrm{O}) \cdot (\mathrm{V_m} - \mathrm{E_k}) \\
P(\mathrm{O}) &= \text{open probability of } \mathrm{I_{Kr}} \\
G_{Kr} &= 0.0135 \cdot [K^+]_{out}^{0.59} \\
E_K &= (R \cdot T/F) \cdot \ln([K^+]_{out}/[K^+]_{in}) \\
\mathrm{C1} \rightarrow \mathrm{O} \text{ or } \mathrm{C1} \rightarrow \mathrm{I} &\quad \alpha\alpha = 66.5 \cdot 10^{-3} \cdot \exp(0.05547153 \cdot (\mathrm{v} - 36)) \\
\mathrm{C2} \rightarrow \mathrm{C1} &\quad \alpha_{\mathrm{in}} = 2.172 \\
\mathrm{C3} \rightarrow \mathrm{C2} &\quad \alpha = 55.5 \cdot 10^{-3} \cdot \exp(0.05547153 \cdot (\mathrm{v} - 12)) \\
\mathrm{C2} \rightarrow \mathrm{C3} &\quad \beta = 2.357 \cdot 10^{-3} \cdot \exp(-0.036588 \cdot \mathrm{v}) \\
\mathrm{C1} \rightarrow \mathrm{C2} &\quad \beta_{\mathrm{in}} = 1.077 \\
\mathrm{O} \rightarrow \mathrm{C1} &\quad \beta\beta = 2.9357 \cdot 10^{-3} \cdot \exp(-0.02158 \cdot \mathrm{v}) \\
\mathrm{I} \rightarrow \mathrm{O} &\quad \alpha\mathrm{i} = 0.439 \cdot \exp(-0.02352 \cdot (\mathrm{v} + 25))) \cdot 4.5/[\mathrm{K^+}]_{\mathrm{out}} \\
\mathrm{O} \rightarrow \mathrm{I} &\quad \beta\mathrm{i} = 0.656 \cdot \exp(0.000942 \cdot \mathrm{v}) \cdot 4.5^{0.3}/[\mathrm{K^+}]_{\mathrm{out}}^{0.3} \\
\mathrm{I} \rightarrow \mathrm{C1} &\quad \mu = (\alpha\mathrm{i} \cdot \beta\beta \cdot \alpha\alpha)/(\alpha\alpha \cdot \beta\mathrm{i})
\end{aligned}
$$

✎ Answer: Modify `ex08`

```
// main.cpp of channel
#include <iostream>
#include <fstream>
#include <cmath>
#include "Markov.h"

int main(){
    const int ndim = NUM_OF_HERG_STATES;
    const double dt = 0.1;
    const double totalTime = 400.0;
    const int npts = (int)(totalTime/dt)+1;

    std::vector<double> y(ndim);

    y[0] = 1;
    Markov ce;
    ce.initRK4(ndim, dt);
```

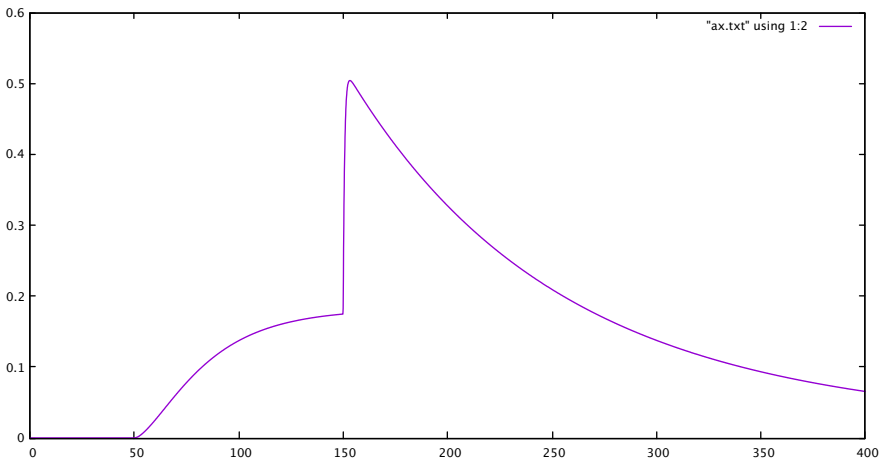
[*1] Cardiovascular Research 50:301 - 313, 2001.

```
    ce.setInitialValues(y);

    std::fstream fd("ax.txt", std::ios::out | std::ios::trunc);
    for(int i=0; i<npts; i++){
        double t = dt*i;
        fd << t << " " << y[3] << std::endl;
//        fd << t << " " << y[0]<< " " << y[1]<< " " << y[2]<< " "
//           << y[3] << " " << y[4] << std::endl;
        ce.advanceStep(t, y);
    }
    fd.close();
#if defined(_WIN32)
    system("..\\tools\\gc ax.txt ");
#else
    system("../tools/gc ax.txt ");
#endif
    return 0;
}
```

```
// markov.h
#ifndef __included__markov_h__
#define __included__markov_h__

#include <vector>
#include "RungeKutta4.h"

#define NUM_OF_HERG_STATES 5
#define HERG_OPEN_STATE 3

using std::vector;

class Markov : public RungeKutta4 {
public:
    void setInitialValues(vector<double>& y);
    void rhs(double t, vector<double> y, vector<double>& dy);
private:
    double a, b, ain, bin, aa, bb, ai, bi, mu;
    double vold;
    double vhold,vpulse,vhold2;
    double Kout;
};
#endif
```

```
// markov.cpp
#include <cmath>
#include "Markov.h"

void Markov::setInitialValues(vector<double>& y){
    Kout = 4.5;
    vold = -1000.0;
    vhold = -80.0;
    vpulse = 20.0;
    vhold2 = -50.0;

    for(int i = 0; i< NUM_OF_HERG_STATES ; i++){
        y[i] = 0.0;
    }
/* more modern way of writhing for loop
    for (vector<double>::iterator i = y.begin() ; i != y.end(); ++i){
        *i = 0.0;
```

```
    }
*/
    y[0] = 1.0;
};

void Markov::rhs(double t, vector<double> y, vector<double>& dy){

    double v;
    if(t<50.0){
        v = vhold;
    }else if(t<150.0){
        v = vpulse;
    }else{
        v =  vhold2;
    }
    if(v != vold){
        aa = 66.5E-3 * exp(0.05547153 * (v-36.0));
        ain = 2.172;
        a = 55.5E-3 * exp(0.05547153 * (v - 12.0));
        b = 2.357E-3 * exp(-0.036588 * v);
        bin = 1.077;
        bb = 2.9357E-3 * exp(-0.02158 * v);
        ai = 0.439 * exp(-0.02352*(v + 25.0)) * 4.5/Kout;
        bi = 0.656 * exp(0.000942 * v) * pow(4.5/Kout, 0.3);
        mu =(ai*bb)/bi;
        vold = v;
    }
    dy[0] = -a*y[0] + b*y[1];
    dy[1] = -(ain + b)*y[1] + a*y[0] + bin*y[2];
    dy[2] = -(bin + 2.0*aa)*y[2] + ain*y[1] + bb*y[3] + mu*y[4];
    dy[3] = -(bb + bi)*y[3] + aa*y[2] + ai*y[4];
    dy[4] = -(mu + ai)*y[4] + aa*y[2] + bi*y[3];
}
```



Open probability in response to the depolarizing pulse (+20 mV) from the holding potential (-80 mV). Note that repolarization (to -50 mV) leads to more channel opening.

3 Synaptic protein

Synaptic proteins can be counted using the SDS-digested freeze-fracture replica labeling electron-microscopy. If n protein particles are observed, how do you estimate the surface area of the synaptic contact?

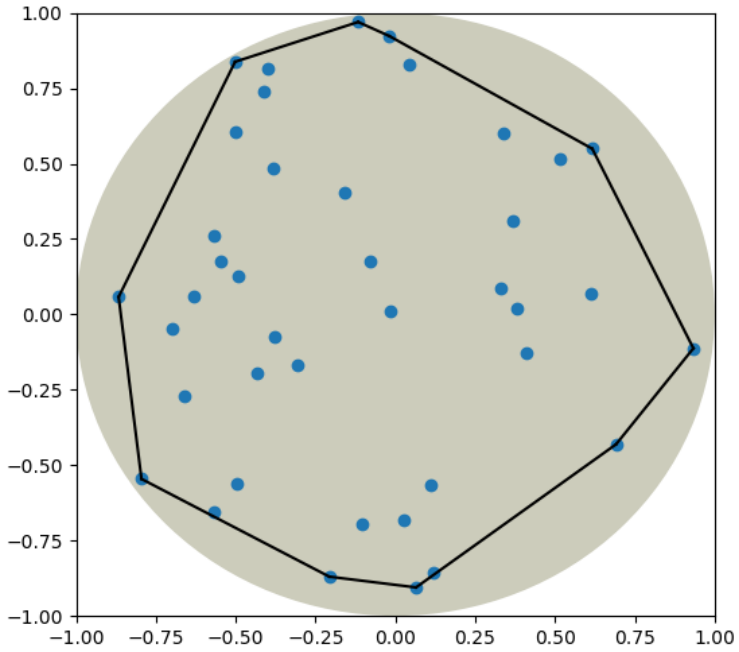
Assume the synaptic contact is round, and the particles are distributed randomly.

Python or Matlab is more convenient for this kind of calculation.
Use the Monte-Carlo method (`ex10`). See Hagiwara et al (2005)[*2]

✎ Answer

```python
import numpy as np
import math
from scipy.spatial import ConvexHull
import matplotlib.pyplot as plt

npts  = 20
niter = 10
mean  = 0.0

for i in range (niter):
    points = np.zeros((npts,2))
    j = 0
    while j < npts:
        a = 2.0*np.random.rand(1, 2)-1.0
        if np.sum( a**2 )<= 1.0 :
            points[j,:] = a[:,:]
            j = j+1
    hull = ConvexHull(points)
    area = hull.volume/math.pi
    print(area)
    mean = mean + area
mean = mean/niter
print("Mean = %2.5f" % mean)

# plot an example (the last trial)
fig = plt.figure()
ax = fig.add_subplot(111)
plt.xlim([-1,1])
plt.ylim([-1,1])
plt.plot(points[:,0], points[:,1], 'o')
for simplex in hull.simplices:
    plt.plot(points[simplex, 0], points[simplex, 1], 'k-')
c = plt.Circle((0.0,0.0), 1.0, fc="#ccccbb")
ax.add_patch(c)
plt.show()
```

[*2] J Comp Neurol. 489:195-216, 2005.



A figure to demonstrate a single trial. Thousands of trials are calculated to get an average area of the convex shapes.

4 Free calcium concentration in the presence of multiple calcium buffers

In the presence of the buffers listed below,

a. Calculate the total calcium concentration when the free calcium concentration is 100 nM.

b. Calculate the free calcium concentration when 100 μM EGTA is added.

	concentration (μM)	K_D (μM)
mobile buffer	290	200
fixed buffer	160	2
EGTA	(100)	0.07

✎ Solve the equation using the regula falsi method.

In case of a single buffer,

$$
\begin{aligned}
\frac{d[\mathrm{CaB}]}{dt} &= k^+[\mathrm{Ca}^{2+}]_f[\mathrm{B}] - k^-[\mathrm{CaB}] \\
[\mathrm{Ca}^{2+}]_f &\to c \\
[\mathrm{B}] &\to b_f \\
[\mathrm{CaB}] &\to b_b \\
B_t &= b_f + b_b \\
\frac{cb_f}{b_b} &= \frac{k^-}{k^+} = K_D \\
\frac{c(B_t - b_b)}{b_b} &= K_D \\
c(B_t - b_b) &= K_D \\
(c + K_D)b_b &= cB_t \\
b_b &= \frac{cB_t}{c + K_D} \\
&= \frac{B_t}{1 + \frac{K_D}{c}} \\
&= B_t(1 - \frac{K_D}{c + K_D}) \\
c_t &= c + B_t(1 - \frac{K_D}{c + K_D})
\end{aligned}
$$

If there are n kinds of buffers,

$$
\begin{aligned}
\frac{d[\mathrm{CaB_i}]}{dt} &= k_i^+[\mathrm{Ca}^{2+}]_f[\mathrm{B_i}] - k_i^-[\mathrm{CaB_i}] \\
b_{i,b} &= \frac{B_{i,t}}{1 + \frac{K_{i,D}}{c}} \\
c_t &= c + \sum_i^n B_{i,t}(1 - \frac{K_{i,D}}{c + K_{i,D}})
\end{aligned}
$$

Question a is simple. Just calculate $c_t = c + \sum_i^n B_{i,t}(1 - \frac{K_{i,D}}{c+K_{i,D}})$.

```
#include <iostream>
#include <fstream>
using namespace std;

double getTotalCalcium(double x){
    const int m = 2;   // number of buffers
    const double B[] = {290.0, 160.0};
    const double Kd[] = {200.0,2.0};

    double y;    // total calcium concdntration

    y = x;
    for(int i=0;i<m;i++){
        y += B[i]*(1.0 - Kd[i]/(x+Kd[i]));
    }
    return y;
}


int main()
{
    fstream fd("ax.txt", ios::out | ios::trunc);
    for(int i = 0;i<100;i++){
        double x = 0.002*i;
        double y = getTotalCalcium(x);
        fd << x << " " << y << endl;
    }
    fd.close();

    cout << "Total calcium = " << getTotalCalcium(0.1) << " microM, when free calcium = 0.1 microM" <<

    system("~/bin/gc ax.txt");
    return 0;
}
```

Question b is to find the c value with added EGTA so that c_t is equal to the value obtained in a. The value of c is changed systematically, and c_t value is obtained. Given c, if a calculated c_t is larger, then reduce c. If a calculated c_t is smaller, then increase c. Finish the calculation when the error is smaller than a very small value (here, $1.0e^{-7}$)

```
//
// regula falsi
//
#include <iostream>
#include <cmath>

using namespace std;
int main()
{
    const double c_total  = 7.87;    // total Ca, in μM
    const double Bm_total = 290.0;   // total mobile buffer, in μM
    const double Bf_total = 160.0;   // total fixed buffer, in μM
    const double E_total  = 100.0;     //100.0; // total EGTA, in μM

    const double Kd_m = 200.0;       // in μM
    const double Kd_f  = 2.0;        // in μM
    const double Kd_E  = 0.07;       // in μM
    const double cerror = 1.0e-7;

```

```
    double c, Bmc, Bfc, Ec;     // free concentration
    double cx;

    int flag = 1;
    double dc = 0.01;
    double cfac = -0.3;
    c = 0.0;
    while(true){
        c += dc;
        Bmc = Bm_total*(1.0 - Kd_m/(c+Kd_m));
        Bfc = Bf_total*(1.0 - Kd_f/(c+Kd_f));
        Ec  = E_total *(1.0 - Kd_E/(c+Kd_E));
        cx  = c + Bmc + Bfc + Ec;
        cout << "c=" << c << ", cx = " << cx << ", dc = " << dc << endl;

        if(fabs(c_total - cx )<cerror)break;

        if((flag==1) &&(cx>c_total)){
            dc *= cfac;
            flag = 0;
        }else if((flag==0)&&(cx<c_total)){
            dc *=cfac;
            flag = 1;
        }
    }
    cout << "Ans: c = " << c << endl;
    return 0;
}
```