Необходимо сделать упрощенную версию расчетного графа, аналогичную расчетному графу в программе MATLAB Simulink и ему подобных, где функциональность представляется в виде графа блоков и связей между ними (для понимания как это выглядит см. рис).

Для этого предлагается реализовать расчетный граф как набор из вершин, реализующих интерфейс абстрактного класса Vertex. Класс Vertex должен предоставлять такой интерфейс как получение количества портов ввода вывода, установка значении входа, расчет результата, возврат результата для выхода. Каждой вершине ставиться в соответствие индекс, каждому порту входа и выхода ставится в соответствие индекс от 0 до числа портов этого типа в Vertex. Граф должен быть ациклическим. Расчетный граф сам должен реализовывать интерфейс Vertex чтобы его можно было как вершину вставлять в другой граф, при этом его портами входа выхода являются свободные (не связанные связями) порты входа выхода вершин внутри него. Значения портов только double. При упорядочивании графа использовать топологическую сортировку.

Например, что-то типа этого

```
class Vertex
{
public:
    virtual ~Vertex();
    virtual int num_inputs() const = 0;
    virtual void set_input(int inp_idx,double inp_val) = 0;
    virtual int num_outputs() const = 0;
    virtual double get_output(int out_idx) = 0;
    virtual void calc_value() = 0;
};

struct Edge
{
    int out_vertex_id;
    int out_port_id;
    int inp_vertex_id;
    int inp_port_id;
};

class CalcGraph : public Vertex
{
public:
    int num_inputs() const override;
    void set_input(int inp_idx,double inp_val) override;
    int num_outputs() const override;
    double get_output(int out_idx) override;
    void calc_value() override;
    void set_data(const std::vector<Vertex*> &vertex,const std::vector<Edge>
&edges);
    ...//Тут еще код.
};
```

Как видно сам CalcGraph может быть добавлен в качестве вершины. Типовая вершина, например оператор «+» может выглядеть так.

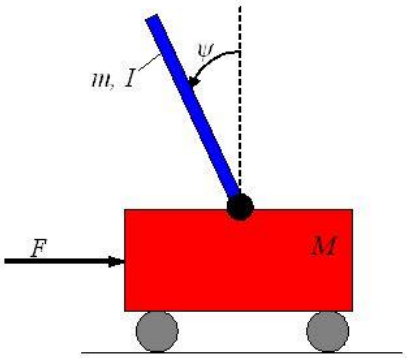
```
class PlusOperator : public Vertex
{
    double m_inp_val[2] = {0};
    double m_out_val = 0;
public:
    int num_inputs() const override{
        return 2;
    }
    void set_input(int inp_idx, double inp_val) override
    {
```

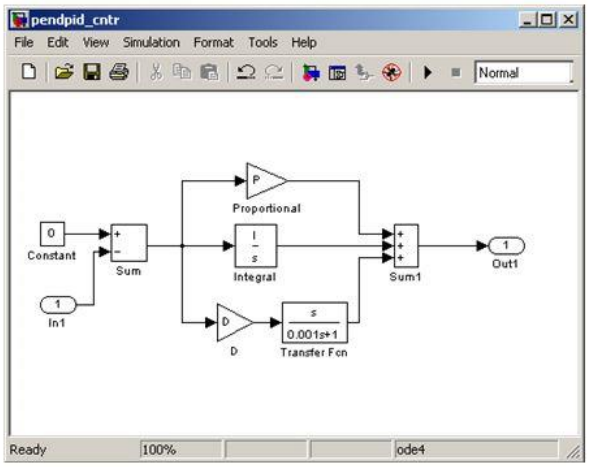
```
        if(inp_idx <= 0 || inp_idx >= 2)
            throw std::out_of_range("Bad inp_idx in the
PlusOperator::set_input");
        m_inp_val[inp_idx] = inp_val;
    }
    void calc_value() override
    {
        m_out_val = m_inp_val[0] + m_inp_val[1];
    }
    int num_outputs() const override
    {
        return 1;
    }
    double get_output(int out_idx) override
    {
        return m_out_val;
    }
};
```

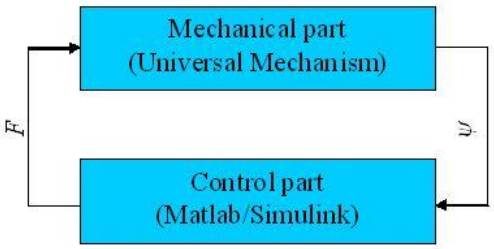
Пример использования

```
CalcGraph cg;
std::vector< Vertex* > vertex;
std::vector< Edge > edges;
vertex.push_back(new PlusOperator);//vertex_id = 0
vertex.push_back(new PlusOperator);//vertex_id = 1
edges.push_back(Edge{0,0,1,0});//First output to second inp0
cg.set_data(vertex,edges);
std::cout << cg.num_inputs() << std::endl;//3
std::cout << cg.num_outputs() << std::endl;//1
cg.set_input(0,1.0);
cg.set_input(1,2.0);
cg.set_input(1,3.0);
cg.calc_value();
std::cout << cg.get_output(0) << std::endl;//1.0 + 2.0 + 3.0 = 6
```

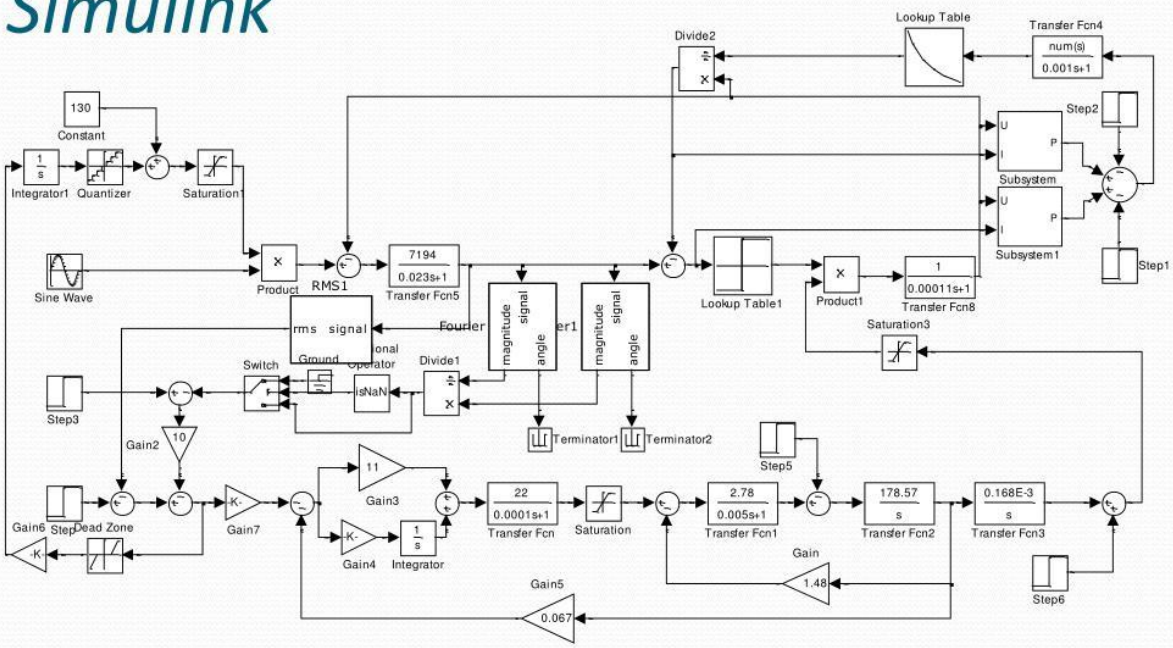
Example 1. Stabilization of the inverted pendulum





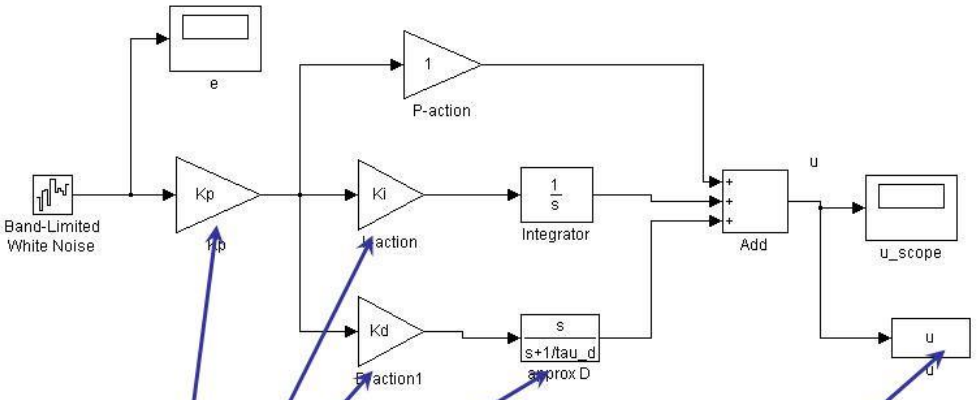


Моделирование в среде *Matlab Simulink*



Simulink example: PID controller

$$K_{PID}(s) = K_P\left(1 + \frac{K_I}{s} + \frac{K_D s}{s + \frac{1}{\tau_D}}\right), \qquad K_P = 0.8357,\ K_I = 5.2070,\ K_D = 0.0171, \tau_D = 0.3413$$



parameters are set/changed in the workspace

In workspce: tout as simulation time, retrieve results via u.signals.values