1 ПО для моделирования нанесения покрытия

1.1 Краткое резюме

Описанное ниже программное обеспечение является учебным и позволяет смоделировать нанесение покрытия на внутреннюю часть тонкой цилиндрической трубки, расположенной под наклоном методом Монте-Карло.

Программа генерирует случайные числа (в данном случае случайным числом является синус угла распыления).

Для записи результатов применяется словарь (число-структура). Ключ словаря – нижнее значение интервала разбиения длины трубки. Например, если молекула попала на отрезок $[10, 20]$, то ключём будет 10.

Значением является структура *values* (см. листинг **1**). Структура содержит в себе поле значения количества попавших в интервал молекул, а также координаты каждой из молекул по стороне L.

```
1  struct value
2  {
3    int i;
4    vector<double> vec;
5  };
6  map<int, value> right_values;
7  map<int, value> left_values;
```

Листинг 1. Запись значений

Программа совершает большое количество итераций по описанному далее методу и выводит результаты в файл (см. листинг **7**). В файле содержится количество попавших в каждый интервал на каждой стороне молекул.

1.2 Расчёт геометрии

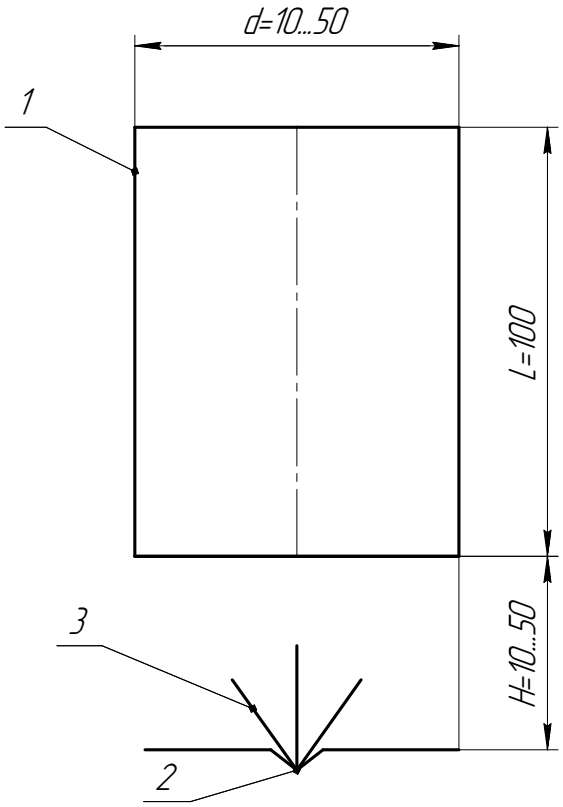


Рис. 1. Экскиз расчёта

1.2.1 Эскиз

На рис. **1** изображён эскиз будущего расчёта.

1. Трубка, на внутреннюю поверхность которой наносятся частицы;
2. Точечный источник, из которого вылетают частицы;
3. Частицы.

1.2.2 Граничные условия

Молекулы будут попадать на внутреннюю поверхность в том случае, если (см. рис. **2**):

— **слева:** угол распыления

$$\psi_{\text{расп}} \in \left[\mu_{\text{пр}}, \psi_{\text{пр}}\right] \tag{1}$$

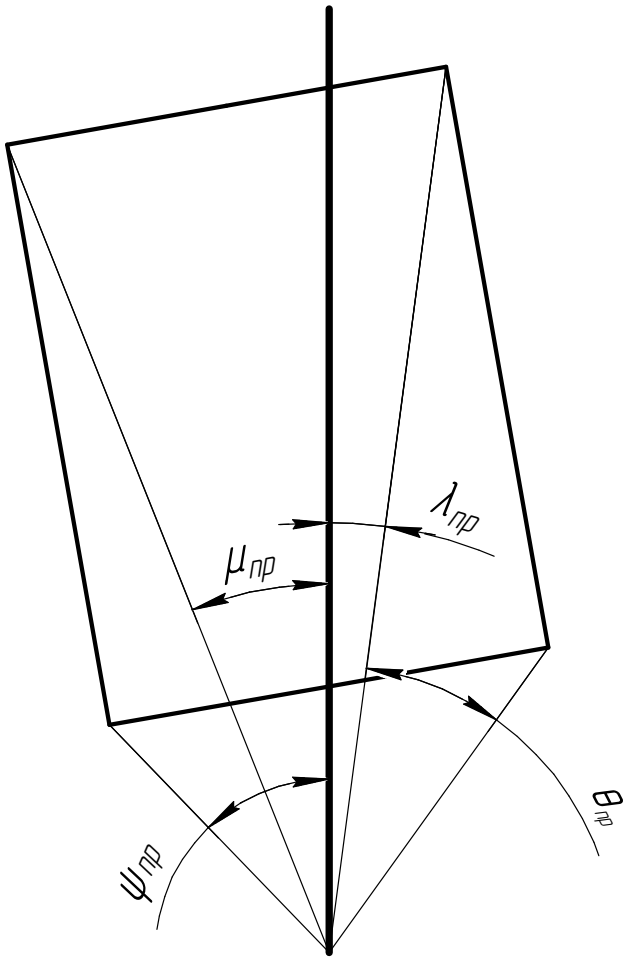


Рис. 2. Граничные условия

— **справа:** угол распыления

$$\psi_{\text{расп}} \in \left[\lambda_{\text{пр}}, \theta_{\text{пр}}\right] \tag{2}$$

1.2.2.1 Нижние граничные условия

Рассмотрим нижнюю часть трубки для нахождения углов $\mu_{\text{пр}}$ и $\lambda_{\text{пр}}$ (см. рис. **3**). В треугольнике ACB:

$$\psi_{\text{пр}} = \operatorname{arctg}\left(\frac{AM}{BM}\right) = \operatorname{arctg}\left(\frac{\frac{d\cos\varphi}{2}}{H - \frac{d}{2}\sin\varphi}\right) = \operatorname{arctg}\left(\frac{d}{\left(2\left(H - \frac{d}{2}\sin\varphi\right)\right)}\right) \tag{3}$$

Похожим образом найдём правое граничное условие из треугольника BNE:

$$\theta_{\text{пр}} = \operatorname{arctg}\left(\frac{NE}{BN}\right) = \operatorname{arctg}\left(\frac{\frac{d\cos\varphi}{2}}{H + \frac{d}{2}\sin\varphi}\right) = \operatorname{arctg}\left(\frac{d}{\left(2\left(H + \frac{d}{2}\sin\varphi\right)\right)}\right) \tag{4}$$

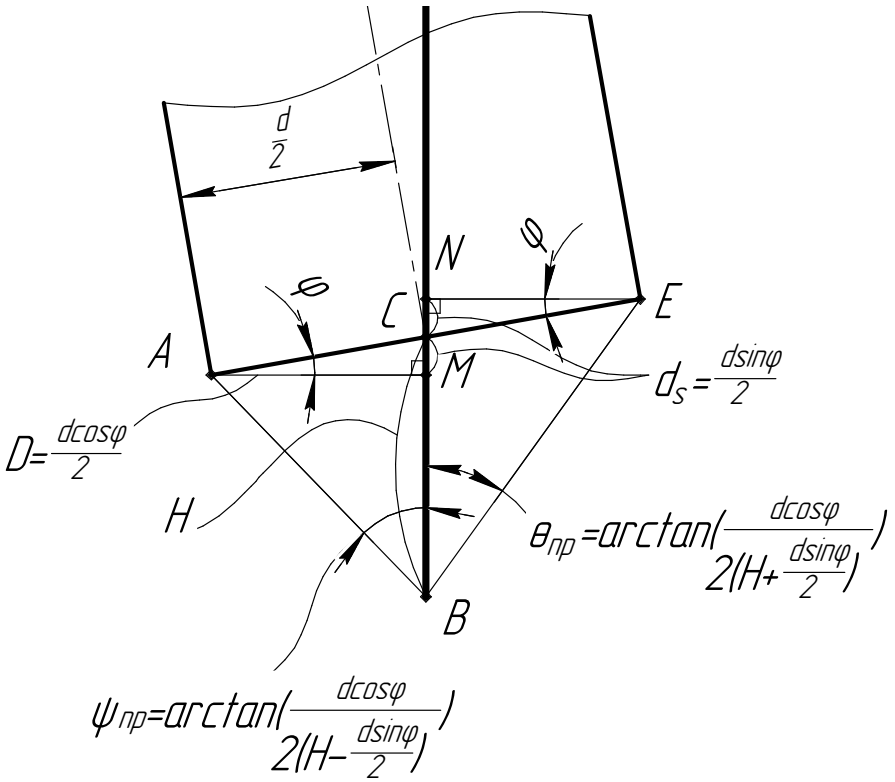


Рис. 3. Нижние граничные условия

1.2.2.2 Верхние граничные условия

Рассмотрим верхнюю часть трубки (см. рис. **4**). Найдём левую границу (рис. **4а**):

1. Угол γ из треугольника *BHF*:

$$\gamma = \operatorname{arctg}\left(\frac{BF}{BH}\right) = \operatorname{arctg}\left(\frac{\frac{d}{2}}{b}\right) = \operatorname{arctg}\left(\frac{d}{2\sqrt{\left(\frac{d}{2}\right)^2 + L^2}}\right) \tag{5}$$

2. Угол *AHB*:

$$AHB = \delta = 180^\circ - \varphi - \gamma \tag{6}$$

3. В треугольнике *AHB* по теореме косинусов находим сторону a:

$$a = b^2 + H^2 - 2bH\cos\left(180^\circ - \varphi - \delta\right) \tag{7}$$

4. По теореме косинусов треугольника *AHB* находим искомый угол:

$$\begin{gathered} b^2 = a^2 + H^2 = 2aH\cos\left(\mu_{\text{пр}}\right) \\ \mu_{\text{пр}} = \arccos\left(\frac{a^2 + H^2 - b^2}{2aH}\right) \end{gathered} \tag{8}$$

Правое граничное условие может быть найдено по аналогии (см. рис. **4б**):

$$\begin{gathered} b^2 = a^2 + H^2 = 2aH\cos\left(\lambda_{\text{пр}}\right) \\ \lambda_{\text{пр}} = \arccos\left(\frac{a^2 + H^2 - b^2}{2aH}\right) \end{gathered} \tag{9}$$

1.2.2.3 Нахождение интервала попадания

Как уже было сказано в **1.1. Краткое резюме**, программа вычисляет координату попадания молекулы на сторону. На рис. **5** изображены расчётные схемы для координат молекул.

1. В треугольнике *BCD* (рис. **5а**) известны стороны *AB* (можно найти из рис. **3**), $BD = \frac{d}{2}$ и граничный угол $\psi_{\text{пр}}$ (см. уравнение (3)). Находим углы ε и $\angle CAB$:

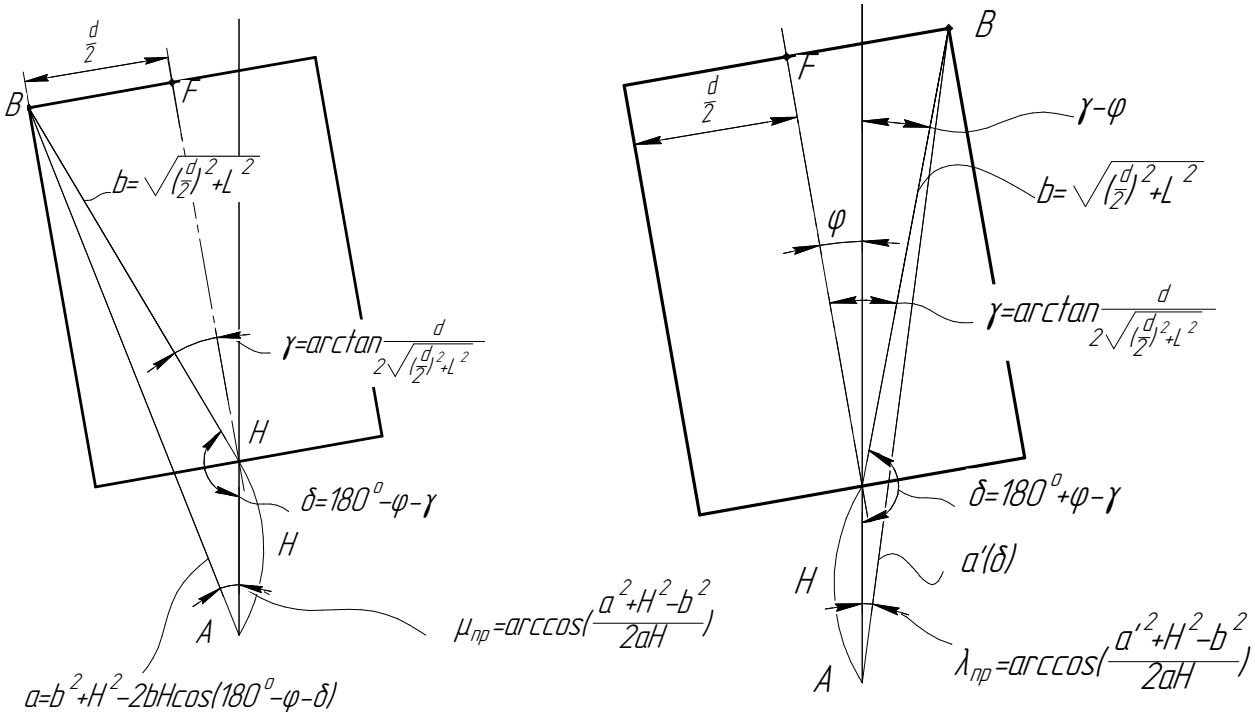
$$\begin{gathered} \varepsilon = \arcsin\left(\frac{2k\sin\psi_{\text{пр}}}{d}\right) \\ \angle CAB = 180^\circ - \psi_{\text{пр}} - \varepsilon \end{gathered} \tag{10}$$

2. В треугольнике *DAB* (см. рис. **5б**) находим углы $\angle ADB$ и $\angle ADB$:

$$\begin{aligned} &\angle ABD = \psi_{\text{пр}} - \psi_{\text{расп}} \\ &\angle DAB = 90^\circ + \angle CAB = 90^\circ + 180^\circ - \psi_{\text{пр}} - \varepsilon = 270^\circ - \psi_{\text{пр}} - \varepsilon \\ &\angle ADB = 180^\circ - \angle DAB - \angle ABD = \\ &\qquad = 180^\circ - 270^\circ + \psi_{\text{пр}} + \varepsilon - \psi_{\text{пр}} + \psi_{\text{расп}} = \varepsilon + \psi_{\text{расп}} - 90^\circ \end{aligned} \tag{11}$$

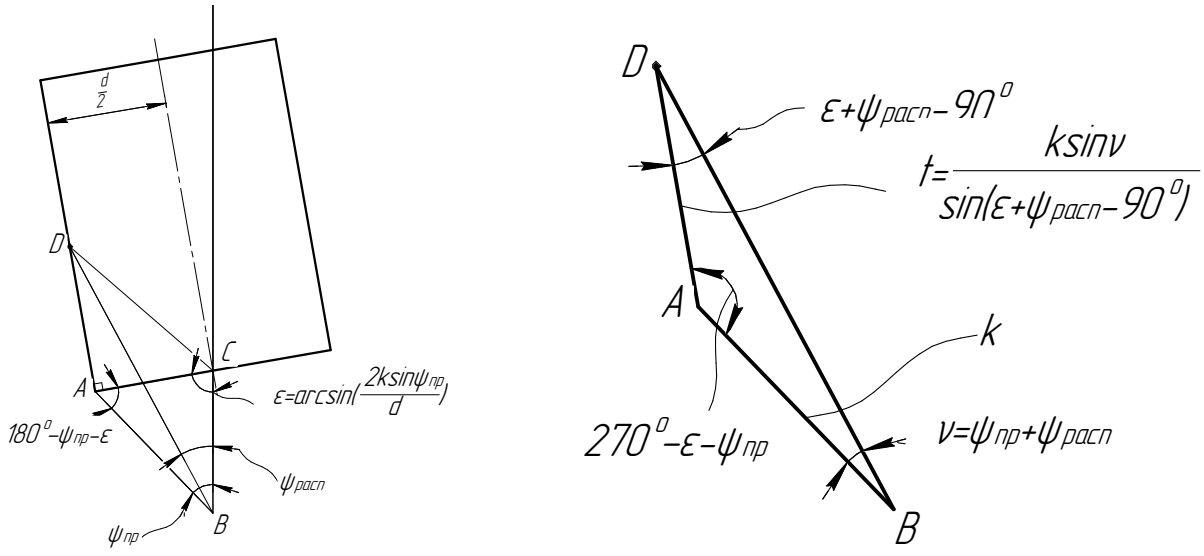
3. По теореме синусов треугольника *ADB* находим искомую сторону (координату распыления):

$$AD = \frac{AB\sin\left(\angle ABD\right)}{\sin\left(\angle ADB\right)} \tag{12}$$



а) Верхнее левое граничное условие б) Верхнее правое граничное условие

Рис. 4. Верхние граничные условия



а) Вычисление координаты молекулы б) Треугольник BAD

Рис. 5. Расчётная схема вычисления координаты молекулы

1.3 Принцип работы программы

1.3.1 Преобразование Бокса-Мюллера

Преобразование Бокса-Мюллера – метод моделирования стандартных нормально распределённых случайных величин:

1. Выбираются 2 независимые случайные величины, равномерно распределённые на отрезке $[-1, 1]$:

$$x \in [-1, 1], \quad y \in [-1, 1] \tag{13}$$

2. Вычисляем s:

$$s = x^2 + y^2 \tag{14}$$

3. Если

$$0 < s \leqslant 1 \tag{15}$$

то вычисляем z_0 и z_1:

$$\begin{aligned} z_0 &= x\sqrt{\frac{-2\ln s}{s}} \\ z_1 &= y\sqrt{\frac{-2\ln s}{s}} \end{aligned} \tag{16}$$

в противном случае повторяем пункт **1** и уравнение (13).

В листинге **2** приведена реализация алгоритма Бокса - Мюллера на языке C++.

```
1  double box_muller ()
2  {
3    double s;
4    static quint64 i = 0;
5    double x;
6    double y;
7    double q;
8    double w;
```

```
9       do
10      {
11        x = randomnum();
12        y = -randomnum();
13        s = x * x + y * y;
14        q = x * qSqrt((-2 * qLn(s))/s);
15        w = y * qSqrt((-2 * qLn(s))/s);
16      }
17      while (s>1 || s==0 || qAbs(q)>1 || qAbs(w) > 1||qAbs(q)>1);
18      i++;
19      if (i % 2 == 0)
20      return w;
21      else
22      return q;
23    }
```

Листинг 2. Реализация алгоритма Бокса-Мюллера

1.3.2 Генерация случайных чисел в библиотеке Qt

Qt предоставляет класс *QRandomGenerator* для генерации случайных чисел. Функция *randomnum()* из листинга **2** реализовывается как раз при помощи этого класса. Реализация функции приведена в листинге **3**.

```
1  double randomnum()
2  {
3    QRandomGenerator generator;
4    return generator.global()->generateDouble();
5  }
```

Листинг 3. Генерация случайных чисел

1.3.3 Реализация программы на языке C++

Первым делом задаются начальные значения. Было принято, что расстояние от источника до центра основания H, угол наклона оси трубки φ и её диа-

метр d лежат в следующих интервалах:

$$\begin{aligned} H &\in [10, 50]\,\text{мм} \\ \varphi &\in [0, 10]^{\circ} \\ d &\in [10, 50]\,\text{мм} \end{aligned} \tag{17}$$

```
1  double H = 30;
2  double L = 100;
3  double fi_pi = 5;//tut
4  double fi = fi_pi*M_PI/180;
5  double d = 40;//tut
6  double r = 145e-12;
```

Листинг 4. Исходные значения

Затем вычисляются величины по формулам, указанным в **1.2. Расчёт геометрии**.

```
1  double left_down_angle = qAtan(d*qCos(fi)/(2*(H-d/2*qSin(fi)))
   );
2  double right_down_angle = qAtan(d*qCos(fi)/(2*(H+d/2*qSin(fi))
   ));
3  double left_down_dlina = d*qCos(fi)/(2*(qSin(left_down_angle))
   );
4  double right_down_dlina = d*qCos(fi)/(2*(qSin(right_down_angle
   )));
5  double left_up_dlina = sqrt(pow(d/2,2)+pow(L,2)+pow(H,2)-2*
   sqrt(pow(d,2)/4+pow(L,2))*H*qCos(M_PI-fi-qAtan(d/(2*L))));
6  double left_up_angle = qAcos(-(pow(d,2)/4+pow(L,2)-pow(
   left_up_dlina,2)-pow(H,2))/(2*H*left_up_dlina));
7  double right_up_dlina = sqrt(pow(d/2,2)+pow(L,2)+pow(H,2)-2*
   sqrt(pow(d,2)/4+pow(L,2))*H*qCos(M_PI+fi-qAtan(d/(2*L))));
8  double right_up_angle = qAcos(-(pow(d,2)/4+pow(L,2)-pow(
   right_up_dlina,2)-pow(H,2))/(2*H*double left_down_dlina = d
   *qCos(fi)/(2*(qSin(left_down_angle)));
```

```
9   double right_down_dlina = d*qCos(fi)/(2*(qSin(right_down_angle
        )));
10  qDebug() << right_down_dlina << "right_down_dlina";
11  double left_up_dlina = sqrt(pow(d/2,2)+pow(L,2)+pow(H,2)-2*
        sqrt(pow(d,2)/4+pow(L,2))*H*qCos(M_PI-fi-qAtan(d/(2*L))));
12  double left_up_angle = qAcos(-(pow(d,2)/4+pow(L,2)-pow(
        left_up_dlina,2)-pow(H,2))/(2*H*left_up_dlina));
13  double right_up_dlina = sqrt(pow(d/2,2)+pow(L,2)+pow(H,2)-2*
        sqrt(pow(d,2)/4+pow(L,2))*H*qCos(M_PI+fi-qAtan(d/(2*L))));
14  double right_up_angle = qAcos(-(pow(d,2)/4+pow(L,2)-pow(
        right_up_dlina,2)-pow(H,2))/(2*H*right_up_dlina));_up_dlina
        ));
15  double eps_left = qAsin(left_down_dlina*2*qSin(left_down_angle
        )/d);
16  double eps_right = M_PI-eps_left;
```

Листинг 5. Вычисление граничных условий

Далее начинается цикл из десяти миллионов итераций (листинг **6**). Именно здесь реализуются алгоритмы, описанные выше.

```
1   for (int j = 0; j < 10000000; ++j)
2   {
3     double buff = box_muller();
4     if (j%1000000 == 0)
5     qDebug() << j/1000000 <<"/10";
6     if (qAsin(buff)>-right_down_angle && qAsin(buff)<
          -right_up_angle)
7     {
8       double rh = dlina(right_down_angle,right_down_dlina,
            eps_right,qAsin(qAbs(buff)));
9       right_values[address(rh)].i++;
10      right_values[address(rh)].vec.push_back(rh);
11    }
12    else if (qAsin(buff)<left_down_angle && qAsin(buff)>
          left_up_angle)
```

```
13      {
14        double lh = dlina(left_down_angle,left_down_dlina,eps_left
              ,qAsin(buff));
15        left_values[address(lh)].i++;
16        left_values[address(lh)].vec.push_back(lh);
17      }
18    }
```

Листинг 6. Тело программы

После исполнения программы данные из массивов записываются в файл (см. листинг 7):

```
1   QFile left_file(filename);
2   QTextStream left_data(&left_file);
3   for (const auto& amount:left_values) {
4     left_data << amount.second.i*2*r*pow(10,6)/10 << "\n";
5   }
6   left_data<< "sep" << "\n"<<"\n";
7   for (const auto& amount:right_values) {
8     left_data << amount.second.i*2*r*pow(10,6)/10 << "\n";
9   }
10  left_file.close();
```

Листинг 7. Запись данных в файл