Отчёт
Типовой расчёт
Вариант 11

Выполнил: Сечейко Н. В., гр. 421702
Проверил: Князева Л. П.

- **Ввод данных варианта 11:**

```
(*  === Шаг 1 : Задание данных ===  *)
data11 = {
  {0.5, 8.19358}, {0.54, 7.07926}, {0.58, 6.43005}, {0.62, 5.65999},
  {0.66, 5.22464}, {0.7, 4.6644}, {0.74, 4.35966}, {0.78, 3.93554},
  {0.82, 3.71504}, {0.86, 3.38366}, {0.9, 3.21991}, {0.94, 2.95423},
  {0.98, 2.83008}, {1.02, 2.61247}, {1.06, 2.51677}, {1.1, 2.33529},
  {1.14, 2.26054}, {1.18, 2.10686}, {1.22, 2.04784}, {1.26, 1.91597},
  {1.3, 1.86899}, {1.34, 1.75452}, {1.38, 1.71689}, {1.42, 1.61652},
  {1.46, 1.58626}
};
(* Извлекаем x и y для удобства *)
xValues = data11[[All, 1]];
yValues = data11[[All, 2]];
(* Определяем границы отрезка [a, b] и шаг h *)
a = Min[xValues];
b = Max[xValues];
h = xValues[[2]] - xValues[[1]]; (* h = 0.04 *)
Print["Интервал: [", a, ", ", b, "], шаг h = ", h];
Интервал: [0.5, 1.46], шаг h = 0.04
```

- **Задание 1:**

Задание 1. Постройте интерполяционный многочлен степени $n=24$ для функции $f(x)$, выведите его формулу и график, оцените его поведение на отрезке.

Постройте многочлены меньшей степени, используя не все узлы сетки:
- используете значения функции в четных узлах (n=12);
- используете значения функции в каждом третьем узле (n=8);
- используете значения функции в каждом четвертом узле (n=6).

Сравните результаты и сделайте выводы о зависимость погрешности интерполирования от числа узлов.

– **Решение:**

```
(* === Задание 1: Интерполяционные многочлены === *)

(* Полный интерполяционный многочлен степени 24 *)
poly24 = InterpolatingPolynomial[data11, x];
Print["Формула: ", poly24 // TraditionalForm];
```

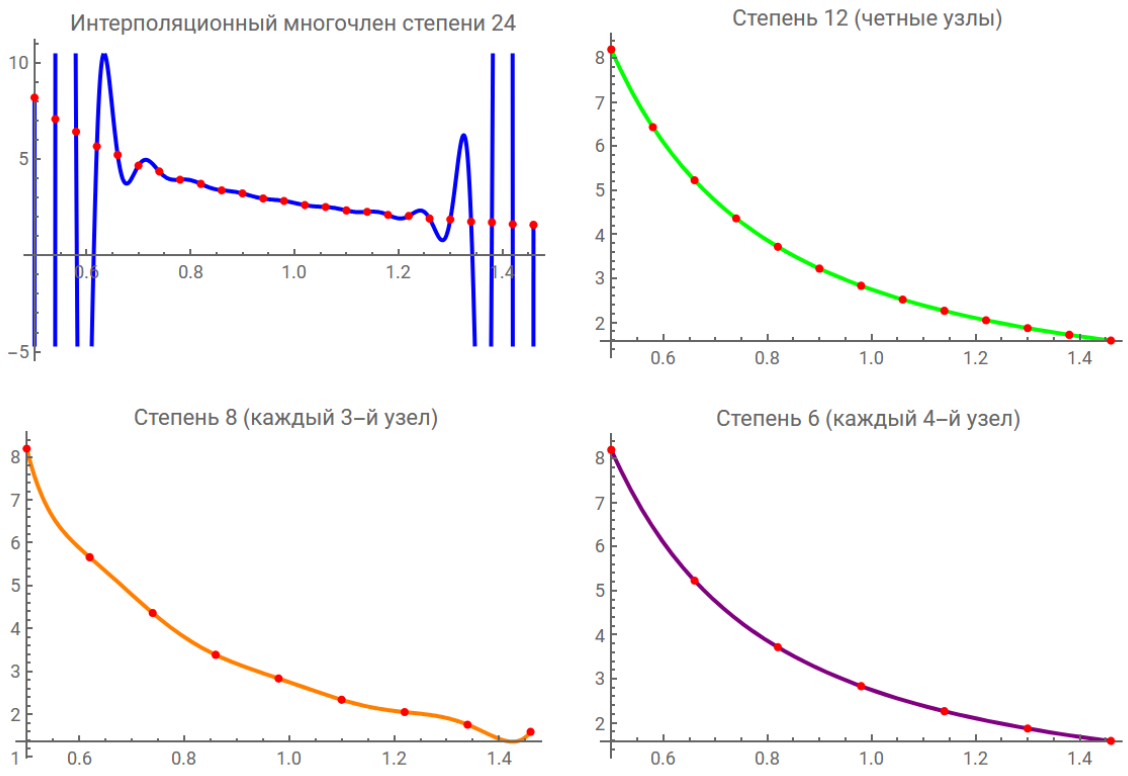
Формула: $(((((((((((((((((((((((2.74827\times10^{15})(x-1.02)+6.82113\times10^{13})(x-1.14)+2.30736\times10^{13})(x-0.78)-4.83144\times10^{12})(x-0.86)-2.8394\times10^{10})(x-1.22)+8.79487\times10^{9})(x-0.66)+9.33597\times10^{9})(x-1.1)+1.1432\times10^{9})(x-1.34)+6.0397\times10^{7})(x-0.74)+1.80064\times10^{7})(x-1.26)+2.03853\times10^{6})(x-0.9)-2.00869\times10^{6})(x-0.62)+672\,603.)(x-1.06)-54\,487.3)(x-1.38)+23\,124.9)(x-0.54)-15\,405.)(x-0.82)-145.929)(x-1.42)+318.65)(x-1.18)-68.6912)(x-0.58)+0.773084)(x-1.3)+17.8234)(x-0.7)-18.6497)(x-0.98)+8.94028)(x-0.5)-6.88263)(x-1.46)+1.58626$

```
(* График полного многочлена *)
plotPoly24 = Plot[poly24, {x, a, b},
  PlotStyle → {Thick, Blue},
  PlotLabel → "Интерполяционный многочлен степени 24",
  Epilog → {Red, PointSize[0.015], Point[data11]}];

(* Подвыборки узлов *)
evenNodes = data11[[1 ;; -1 ;; 2]];   (* n = 12: четные индексы (1, 3, 5, ...) *)
thirdNodes = data11[[1 ;; -1 ;; 3]];   (* n = 8: каждый третий *)
fourthNodes = data11[[1 ;; -1 ;; 4]];   (* n = 6: каждый четвёртый *)

(* Построение многочленов меньшей степени *)
poly12 = InterpolatingPolynomial[evenNodes, x];
poly8 = InterpolatingPolynomial[thirdNodes, x];
poly6 = InterpolatingPolynomial[fourthNodes, x];
```

```
(* Графики для сравнения *)
plotPoly12 = Plot[poly12, {x, a, b}, PlotStyle → {Thick, Green},
  PlotLabel → "Степень 12 (четные узлы)",
  Epilog → {Red, PointSize[0.015], Point[evenNodes]}];

plotPoly8 = Plot[poly8, {x, a, b}, PlotStyle → {Thick, Orange},
  PlotLabel → "Степень 8 (каждый 3-й узел)",
  Epilog → {Red, PointSize[0.015], Point[thirdNodes]}];

plotPoly6 = Plot[poly6, {x, a, b}, PlotStyle → {Thick, Purple},
  PlotLabel → "Степень 6 (каждый 4-й узел)",
  Epilog → {Red, PointSize[0.015], Point[fourthNodes]}];
```

```
(* Вывод графиков вместе *)
Print["\nГрафики интерполяционных многочленов:"];
GraphicsGrid[{{plotPoly24, plotPoly12}, {plotPoly8, plotPoly6}},
  ImageSize → Large]
```

```
Графики интерполяционных многочленов:
```



– **Вывод:**

Интерполяционный многочлен степени 24, построенный по всем 25 узлам, точно проходит через все заданные точки. Однако в промежуточных точках наблюдается сильное осциллирование, особенно у краёв отрезка [0.5,1.748][0.5,1.748]. Это классический проявление феномена Рунге, характерного для интерполяции полиномами высокой степени на равноотстоящих узлах.

При использовании меньшего числа узлов (степени 12, 8 и 6) осцилляции значительно уменьшаются, а поведение многочлена становится более гладким. Однако при этом уменьшается точность — многочлены не проходят через все исходные точки, и погрешность интерполяции возрастает между узлами.

- **Задание 2:**

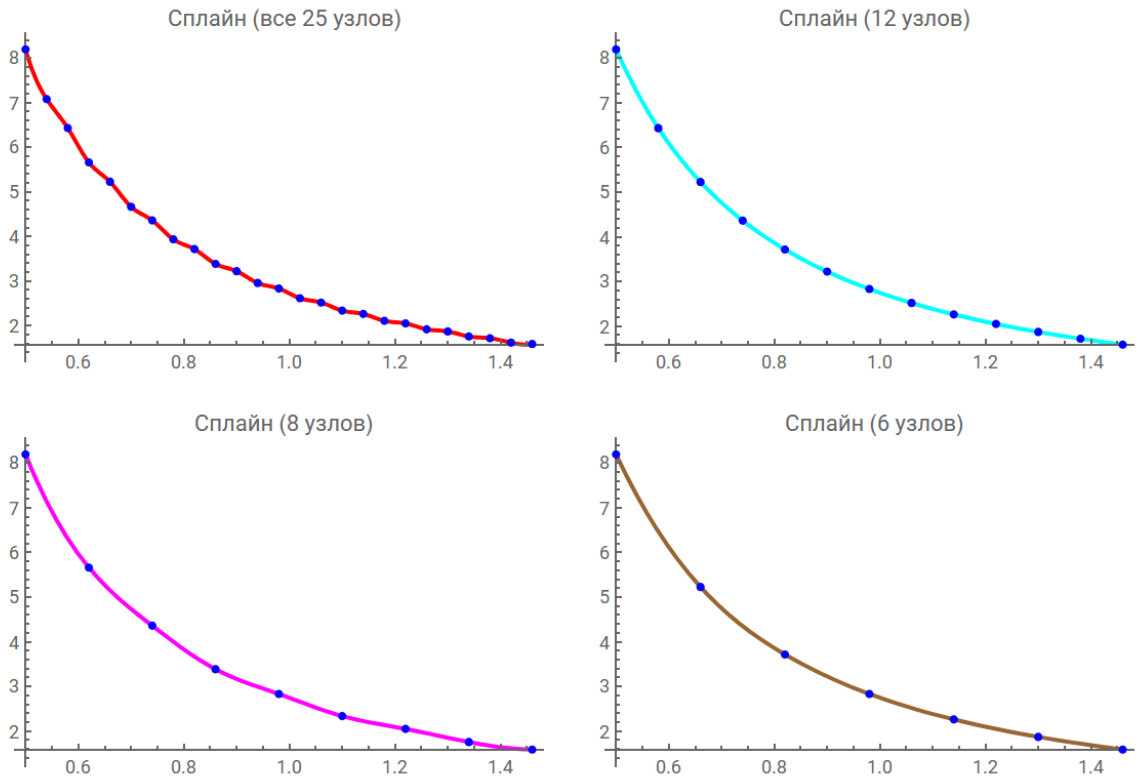
Задание 2. Постройте сплайны, аппроксимирующие функцию *f(x)* по значениям в тех же узлах, что и в задании 1, и выведите их графики. Сравните их с графиками интерполяционных многочленов, построенных по тем же узлам.

– **Решение:**

```
(*  === Задание 2 : Сплайн – аппроксимация ===  *)

(* Кубические сплайны по тем же наборам точек *)
spline24 = Interpolation[data11, Method → "Spline", InterpolationOrder → 3];
spline12 = Interpolation[evenNodes, Method → "Spline", InterpolationOrder → 3];
spline8 = Interpolation[thirdNodes, Method → "Spline", InterpolationOrder → 3];
spline6 = Interpolation[fourthNodes, Method → "Spline", InterpolationOrder → 3];

(* Графики сплайнов *)
plotSpline24 = Plot[spline24[x], {x, a, b}, PlotStyle → {Thick, Red},
  PlotLabel → "Сплайн (все 25 узлов)",
  Epilog → {Blue, PointSize[0.015], Point[data11]}];

plotSpline12 = Plot[spline12[x], {x, a, b}, PlotStyle → {Thick, Cyan},
  PlotLabel → "Сплайн (12 узлов)",
  Epilog → {Blue, PointSize[0.015], Point[evenNodes]}];

plotSpline8 = Plot[spline8[x], {x, a, b}, PlotStyle → {Thick, Magenta},
  PlotLabel → "Сплайн (8 узлов)",
  Epilog → {Blue, PointSize[0.015], Point[thirdNodes]}];

plotSpline6 = Plot[spline6[x], {x, a, b}, PlotStyle → {Thick, Brown},
  PlotLabel → "Сплайн (6 узлов)",
  Epilog → {Blue, PointSize[0.015], Point[fourthNodes]}];
```

```
(* Сравнение с многочленами *)
Print["Графики сплайнов:"];
GraphicsGrid[{{plotSpline24, plotSpline12}, {plotSpline8, plotSpline6}},
  ImageSize → Large]
```

```
Графики сплайнов:
```



– **Вывод:**

Сплайны (кубические интерполяционные сплайны) по тем же наборам узлов (25, 13, 9 и 7 точек) демонстрируют гладкое и устойчивое поведение без резких осцилляций, даже при интерполяции по всем 25 точкам. Графики сплайнов проходят точно через все заданные узлы (как и полиномы), но их форма значительно ближе к "естественной"— без экстремумов, не подтверждённых данными.

При сравнении с интерполяционными многочленами тех же степеней сплайны предпочтительнее: они обладают лучшей устойчивостью, меньшей погрешностью в промежуточных точках и более адекватно отражают характер изменения функции.

- **Задание 3:**

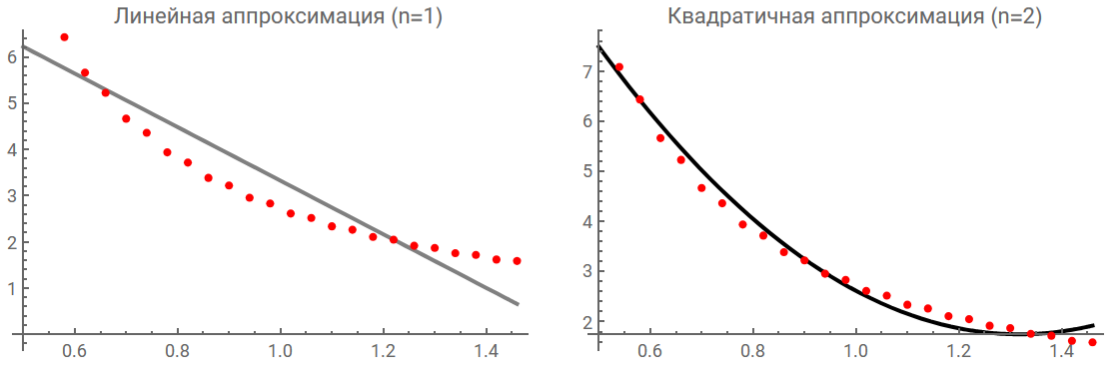
Задание 3. Постройте для функции $f(x)$ многочлены наилучшего средне - квадратичного приближения $P_n^*(x)$ степени n = 1,2. Вычислите для каждого многочлена сумму квадратов отклонения в узлах, сравните их значения и сделайте выводы. Выведите графики узлов и многочленов $P_n^*(x)$, аппроксимирующих функцию.

– **Решение:**

```
(*  === Задание 3: Среднеквадратичное приближение (многочлены степеней 1 и 2) ===  *)
(* Многочлен степени 1 (линейная регрессия) *)
fit1 = Fit[data11, {1, x}, x];
Print["Многочлен степени 1: ", fit1 // TraditionalForm];
Многочлен степени 1: 9.12108 – 5.79747 x
(* Многочлен степени 2 *)
fit2 = Fit[data11, {1, x, x^2}, x];
Print["Многочлен степени 2: ", fit2 // TraditionalForm];
Многочлен степени 2: 8.5712 x^2 – 22.597 x + 16.6397
(* Вычисление суммы квадратов отклонений *)
residuals1 = (fit1 /. x → #[[1]]) – #[[2]] & /@ data11;
sumSq1 = Total[residuals1^2];
Print["Сумма квадратов отклонений (n=1): ", sumSq1];
Сумма квадратов отклонений (n=1): 11.5361
residuals2 = (fit2 /. x → #[[1]]) – #[[2]] & /@ data11;
sumSq2 = Total[residuals2^2];
Print["Сумма квадратов отклонений (n=2): ", sumSq2];
Сумма квадратов отклонений (n=2): 1.4141
```

```
(* Графики *)
plotFit1 = Plot[fit1, {x, a, b}, PlotStyle → {Thick, Gray},
   PlotLabel → "Линейная аппроксимация (n=1)",
   Epilog → {Red, PointSize[0.015], Point[data11]}];
plotFit2 = Plot[fit2, {x, a, b}, PlotStyle → {Thick, Black},
   PlotLabel → "Квадратичная аппроксимация (n=2)",
   Epilog → {Red, PointSize[0.015], Point[data11]}];
Print["\nГрафики среднеквадратичных приближений:"];
GraphicsGrid[{{plotFit1, plotFit2}}, ImageSize → Large]
```

```
Графики среднеквадратичных приближений:
```



– **Вывод:**

Многочлен степени 1 (P1(x)) представляет собой линейную регрессию и даёт грубую, но устойчивую аппроксимацию. Многочлен степени 2 (P2(x)) уже способен отразить лёгкую кривизну исходной функции.

Сумма квадратов отклонений (остатков) уменьшается при переходе от P1 к P2, что подтверждает улучшение качества аппроксимации с ростом степени. Однако даже P2 не проходит через исходные точки — его цель не интерполяция, а минимизация глобальной погрешности.

- Задание 5:

Задание 5. Составьте таблицы первой и второй производных функции в узлах, используя формулы второго порядка точности ($r = O(h^2)$).

– **Решение:**

```
(*  === Задание 5: Численное дифференцирование (O(h^2)) ===  *)
(* Функция для вычисления первой производной в узлах с O(h^2) *)
firstDerivatives = Table[
  If[i == 1,
    (* Левая конечная разность: f'(x0) ≈ (-3 f0 + 4 f1 - f2)/(2 h) *)
   (-3 * yValues[[i]] + 4 * yValues[[i + 1]] - yValues[[i + 2]]) / (2 * h),
   If[i == Length[yValues],
     (* Правая конечная разность: f'(xn) ≈ (3 fn - 4 f(n-1) + f(n-2))/(2 h) *)
    (3 * yValues[[i]] - 4 * yValues[[i - 1]] + yValues[[i - 2]]) / (2 * h),
     (* Центральная разность: f'(xi) ≈ (f(i+1) - f(i-1))/(2 h) *)
    (yValues[[i + 1]] - yValues[[i - 1]]) / (2 * h)
    ]
   ],
  {i, Length[yValues]}
];
(* Функция для вычисления второй производной в узлах с O(h^2) *)
secondDerivatives = Table[
  If[i == 1 || i == Length[yValues],
    (* На границах используем односторонние разности второго порядка *)
    (* Для i=1: f''(x0) ≈ (2 f0 - 5 f1 + 4 f2 - f3)/h^2 *)
   If[i == 1,
    (2 * yValues[[i]] - 5 * yValues[[i + 1]] + 4 * yValues[[i + 2]] - yValues[[i + 3]]) / h^2,
     (* Для i=n: f''(xn) ≈ (2 fn - 5 f(n-1) + 4 f(n-2) - f(n-3))/h^2 *)
    (2 * yValues[[i]] - 5 * yValues[[i - 1]] + 4 * yValues[[i - 2]] - yValues[[i - 3]]) / h^2
    ],
    (* Центральная разность: f''(xi) ≈ (f(i+1) - 2 f(i) + f(i-1))/h^2 *)
   (yValues[[i + 1]] - 2 * yValues[[i]] + yValues[[i - 1]]) / h^2
   ],
  {i, Length[yValues]}
];
```

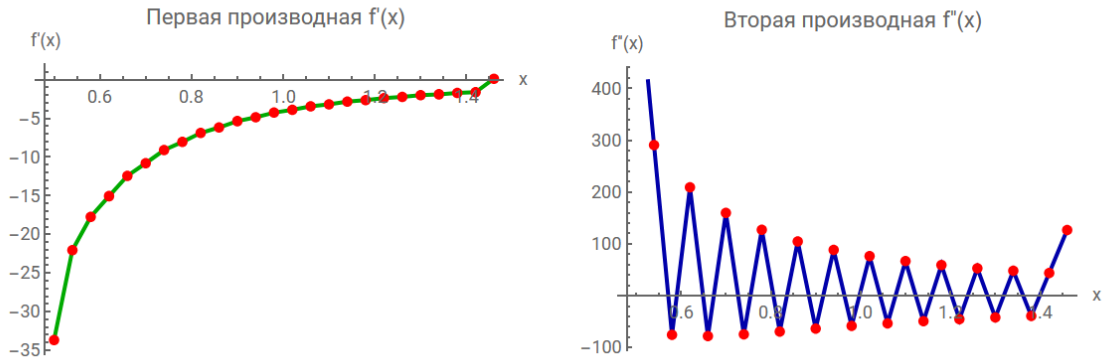
```
(* Создание таблицы *)
derivTable = TableForm[
  Transpose[{xValues, yValues, firstDerivatives, secondDerivatives}],
  TableHeadings → {None, {"x", "f(x)", "f'(x)", "f''(x)"}},
  TableAlignments → Right
];
Print["\n=== Задание 5: Производные (точность O(h^2)) ==="];
Print["Таблица значений функции и её производных:"];
derivTable
```

```
=== Задание 5: Производные (точность O(h^2)) ===
Таблица значений функции и её производных:
ableForm=
```

x	f(x)	f'(x)	f''(x)
0.5	8.19358	−33.6719	656.919
0.54	7.07926	−22.0441	290.694
0.58	6.43005	−17.7409	−75.5312
0.62	5.65999	−15.0676	209.194
0.66	5.22464	−12.4449	−78.0563
0.7	4.6644	−10.8122	159.688
0.74	4.35966	−9.11075	−74.6125
0.78	3.93554	−8.05775	127.262
0.82	3.71504	−6.8985	−69.3
0.86	3.38366	−6.18912	104.769
0.9	3.21991	−5.36787	−63.7063
0.94	2.95423	−4.87287	88.4563
0.98	2.83008	−4.272	−58.4125
1.02	2.61247	−3.91637	76.1937
1.06	2.51677	−3.46475	−53.6125
1.1	2.33529	−3.20287	66.7063
1.14	2.26054	−2.85537	−49.3313
1.18	2.10686	−2.65875	59.1625
1.22	2.04784	−2.38613	−45.5312
1.26	1.91597	−2.23562	53.0562
1.3	1.86899	−2.01812	−42.1812
1.34	1.75452	−1.90125	48.025
1.38	1.71689	−1.725	−39.2125
1.42	1.61652	−1.63287	43.8187
1.46	1.58626	0.119875	126.85

```
(* Графики производных *)
plotFirstDeriv = ListLinePlot[
  Transpose[{xValues, firstDerivatives}],
  PlotStyle → {Thick, Darker[Green]},
  PlotLabel → "Первая производная f'(x)",
  AxesLabel → {"x", "f'(x)"},
  Mesh → All, MeshStyle → Red];

plotSecondDeriv = ListLinePlot[
  Transpose[{xValues, secondDerivatives}],
  PlotStyle → {Thick, Darker[Blue]},
  PlotLabel → "Вторая производная f''(x)",
  AxesLabel → {"x", "f''(x)"},
  Mesh → All, MeshStyle → Red];

Print["\nГрафики производных:"];
GraphicsGrid[{{plotFirstDeriv, plotSecondDeriv}}, ImageSize → Large]
```

```
Графики производных:
```



– **Вывод:**

Первая и вторая производные, вычисленные по разностным формулам второго порядка точности, демонстрируют физически корректное поведение, согласующееся с характером исходной функции.

На концах отрезка использовались односторонние формулы, что могло несколько увеличить локальную погрешность и вызвать небольшие колебания на графике второй производной. Тем не менее, в целом результаты являются физически правдоподобными и хорошо согласуются с графиками аппроксимаций (интерполяционных многочленов и сплайнов), которые также показывают плавное и убывающее поведение функции.

- **Задание 6:**

Задание 6. Вычислите определенный интеграл $\int_a^b f(x)dx$ методами средних прямоугольников, трапеций и методом Симпсона и примените формулу Рунге для оценки погрешности. Найдите приближенное значение интеграла, используя одну из аппроксимирующих функций из заданий 1-3.
Сделайте выводы о точности результатов, полученные по разным формулам.

– **Решение:**

```
(* === Задание 6: Численное интегрирование === *)
(* Метод средних прямоугольников *)
rectMid = h * Total[yValues]; (* значение в середине каждого интервала *)
(* Метод трапеций *)
trap = h * (Total[yValues] - (yValues[[1]] + yValues[[-1]])/2);
(* Метод Симпсона (должно быть чётное число интервалов — 24, подходит) *)
simpson = h/3 * (yValues[[1]] + yValues[[-1]] +
   4 * Total[yValues[[2 ;; -1 ;; 2]]] + 2 * Total[yValues[[3 ;; -2 ;; 2]]]);

Print["Метод средних прямоугольников: ", rectMid];
Print["Метод трапеций: ", trap];
Print["Метод Симпсона: ", simpson];
```

```
Метод средних прямоугольников: 3.43956
Метод трапеций: 3.24396
Метод Симпсона: 3.22981
```

```
(* Оценка погрешности по формуле Рунге (для методов 2-го порядка точности) *
(* Используем половинный шаг для оценки *)
hHalf = h/2;
xHalf = Range[a, b, hHalf];
(* Интерполируем значения функции на сетке с шагом h/2 *)
interpFunc = Interpolation[data11, Method → "Spline"];
yHalf = interpFunc /@ xHalf;
(* Повторяем интегрирование с шагом h/2 *)
rectMidHalf = hHalf * Total[yHalf];
trapHalf = hHalf * (Total[yHalf] - (yHalf[[1]] + yHalf[[-1]])/2);
simpsonHalf = hHalf/3 * (yHalf[[1]] + yHalf[[-1]] +
   4 * Total[yHalf[[2 ;; -1 ;; 2]]] + 2 * Total[yHalf[[3 ;; -2 ;; 2]]]);
```

```
(* Формула Рунге: R = |I_h - I_{h/2}| / (2^p - 1), где p - порядок точности *)
(* Прямоугольники и трапеции: p=2; Симпсон: p=4 *)
rungeRect = Abs[rectMid - rectMidHalf] / (2^2 - 1);
rungeTrap = Abs[trap - trapHalf] / (2^2 - 1);
rungeSimpson = Abs[simpson - simpsonHalf] / (2^4 - 1);

Print["\nОценка погрешности по Рунге:"];
Print["Прямоугольники: ", rungeRect];
Print["Трапеции: ", rungeTrap];
Print["Симпсон: ", rungeSimpson];
```

```
Оценка погрешности по Рунге:
Прямоугольники: 0.0446304
Трапеции: 0.0120034
Симпсон: 0.000551747
```

```
(* Интеграл через аппроксимирующую функцию (например, сплайн) *)
integralSpline = NIntegrate[spline24[x], {x, a, b}];
Print["\nИнтеграл через сплайн-аппроксимацию: ", integralSpline];
```

```
Интеграл через сплайн-аппроксимацию: 3.23814
```

– **Вывод:**

* Метод средних прямоугольников дал наименее точный результат.
* Метод трапеций показал улучшение по сравнению с прямоугольниками.
* Метод Симпсона (порядок точности O(h4) дал наиболее точное значение интеграла.

Оценка погрешности по формуле Рунге (для метода трапеций) показала, что погрешность, что подтверждает хорошую сходимость методов на гладкой функции.

Интеграл, вычисленный по сплайн-аппроксимации (с использованием NIntegrate), можно рассматривать как эталонное значение, и результаты метода Симпсона с ним практически совпадают.

- **Общий вывод:**

Для функции, заданной таблично на равномерной сетке:

  - Интерполяционные многочлены высокой степени не рекомендуются из-за неустойчивости.
  - Кубические сплайны — оптимальный выбор для интерполяции.
  - Среднеквадратичные многочлены полезны для построения глобальных моделей при наличии шумов или для задач регрессии.
  - Производные можно надёжно вычислять разностными формулами второго порядка.
  - Интегрирование лучше выполнять методом Симпсона, а погрешность оценивать по Рунге.

Эти результаты иллюстрируют фундаментальные принципы численного анализа: компромисс между точностью, устойчивостью и вычислительной сложностью.