Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе № 1

По курсу: "Математическое моделирование"

Приближенный аналитический метод Пикара в сравнении с численными методами

Студент: Турсунов Жасурбек Рустамович

Группа: ИУ7-66Б

Оценка(баллы):

Преподователь: Градов Владимир Михайлович

Москва, 2021 г.

Содержание

Введение

Цель работы: Получение навыков решения задачи Коши для ОДУ методами Пикара и явным и неявным методами Эйлера.

Обыкновенными дифференциальными уравнениями называются уравнения с одной независимой переменной. Если независимых переменных больше, чем одна, то уравнение называется дифференциальным уравнением с частными производными.

С помощью обыкновенных дифференциальных уравнений строятся модели движения систем взаимодействующих частиц, электротехнических процессов в электрических цепях, кинетики химических реакций, процессов заселения уровней энергии в высокотемпературных средах и многих других объектов и процессов. К задачам для обыкновенных дифференциальных уравнений сводятся некоторые задачи для уравнений в частных производных, когда многомерное уравнение позволяет провести разделение переменных (например, привычислении энергетического спектра частиц в полях определенной симметрии).

Существует три метода решения обыкновенных дифференциальных уравнений:

1. точные;
2. приближенные;
3. численные.

1 Аналитическая часть

Существует задача Коши:

$$\begin{cases} u'(x) = f(x,u) \\ u(\varepsilon) = n \end{cases}$$

Аналитически эту задачу не решить. Но методом Пикара эта задача становится решаемой:

$$\begin{cases} y^{(1)}(x) = n + \int_0^x f(t, y^{(i-1)}(t))dt \\ y^{(0)} = n \end{cases}$$

Рассмотрим пример:

$$\begin{cases} u'(x) = x^2 + u^2 \\ u(0) = 0 \end{cases}$$

Тогда первое приближение метода Пикара:

$$y^{(1)}(x) = 0 + \int_0^x t^2 dt = \frac{x^3}{3}$$

Второе приближение:

$$y^{(2)}(x) = 0 + \int_0^x [t^2 + (\frac{x^3}{3})^2]dt = \frac{t^3}{3} + \frac{t^7}{63}\Big|_0^x = \frac{x^3}{3} + \frac{x^7}{63}$$

Третье приближение:

$$y^{(3)}(x) = 0 + \int_0^x [t^2 + (\frac{x^3}{3} + \frac{x^7}{63})^2]dt = \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535}$$

Четвертое приближение:

$$y^{(4)}(x) = 0 + \int_0^x [t^2 + (\frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535})^2]dt = \frac{x^3}{3} + \frac{x^7}{63} + \frac{2x^{11}}{2079} + \frac{x^{15}}{59535} + \frac{2x^{15}}{93555}$$

$$+ \frac{2x^{19}}{2488563} + \frac{2x^{23}}{86266215} + \frac{x^{23}}{99411543} + \frac{2x^{27}}{3341878155} + \frac{x^{31}}{109876902975}$$

Также эту задачу можно решить численно, воспользовавшись явной и неявной схемой Эйлера:

1. явная: $y_{n+1} = y_n + hf(x_n, y_n)$;

2. неявная: $y_{n+1} = y_n + h(f(x_{n+1}, y_{n+1}‘))$.

Явная схема Эйлера - простейший численный метод. В практике вычислений он употребляется очень редко из-за невысокой точности. Но на его примере удобно пояснить способы построения и исследования численных методов.

Решая алгебраическое уравнение неявной схемой Эйлера, можно определить y_{n+1} которое и будет приближенным значением искомого решения и $u(x_n)$.Но у этой схемы есть серьезные недостатки. Во-первых, неизвестно, имеет ли уравнение вещественный корень, т. е. разрешима ли задача. Во-вторых, даже если корень есть, то как его найти? Метод Ньютона применять нежелательно, так как для этого надо дифференцировать $f(x, u)$. Метод деления пополам не обобщается на системы уравнений. Остается метод последовательных приближений.

2 Технологическая часть

2.1 Листинг кода

В данном пункте представлен листинг кода.

На листинге 1 представлен листинг явной схемы Эйлера.

```
def euler(x, y, n, h):
  outY = []
  for i in range(n):
    try:
      y += h * f(x, y)
      outY.append(y)
      x += h
    except OverflowError:
      outY.append('overflow')
      for _ in range(i, n-1):
        outY.append('none')
      break
  return outY

```

Листинг 1: явная схема Эйлера

На листинге 2 представлен листинг неявной схемы Эйлера.

```
  def explicit_euler(x, y, n, h):
    outY = [y]
    for i in range(n):
      D = 1 - 4*h*(y + h*((x + h)**2))
      if D < 0:
        outY.append('D < 0')
        for _ in range(i, n-2):
          outY.append('-----')
        break
      y = (1 - sqrt(D)) / (2*h)
      x += h
      outY.append(y)
    return outY
```

Листинг 2: неявная схема Эйлера

На листинге 3 представлен листинг первого, второго, третьего и четвертого приближения метода Пикара.

```
def picar1(x):
  return x ** 3 / 3
def picar2(x):
  return picar1(x) + x ** 7 / 63
def picar3(x):
  return picar2(x) +  (x ** 11) * (2 / 2079) + (x ** 15) / 59535
def picar4(x, picar3):
  return picar3 + (x ** 15)*(2 / 93555) + (x ** 19)*(2 / 3393495) + \
  (x ** 19)*(2 / 2488563) + (x ** 23)*(2 / 86266215) + \
  (x ** 23)*(1 / 99411543) + (x ** 27)*(2 / 3341878155) + \
  (x ** 31)*(1 / 109876902975)
```

3 Исследовательская часть

3.1 Пример работы

Ниже на Рисунке 1 и 2 представлены примеры работы программы.

x	Пикар 1	Пикар 2	Пикар 3	Пикар 4	Явный	Неявный
0.00000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0
0.02000	0.00000267	0.00000267	0.00000267	0.00000267	0.00000267	0.00000267
0.04000	0.00002133	0.00002133	0.00002133	0.00002133	0.00002134	0.00002135
0.06000	0.00007200	0.00007200	0.00007200	0.00007200	0.00007202	0.00007203
0.08000	0.00017067	0.00017067	0.00017067	0.00017067	0.00017070	0.00017071
0.10000	0.00033333	0.00033333	0.00033333	0.00033333	0.00033338	0.00033340
0.12000	0.00057600	0.00057601	0.00057601	0.00057601	0.00057608	0.00057609
0.14000	0.00091467	0.00091468	0.00091468	0.00091468	0.00091478	0.00091479
0.16000	0.00136533	0.00136538	0.00136538	0.00136538	0.00136550	0.00136552
0.18000	0.00194400	0.00194410	0.00194410	0.00194410	0.00194426	0.00194427
0.20000	0.00266667	0.00266687	0.00266687	0.00266687	0.00266707	0.00266708
0.22000	0.00354933	0.00354973	0.00354973	0.00354973	0.00354997	0.00354998
0.24000	0.00460800	0.00460873	0.00460873	0.00460873	0.00460902	0.00460903
0.26000	0.00585867	0.00585994	0.00585994	0.00585994	0.00586028	0.00586029
0.28000	0.00731733	0.00731948	0.00731948	0.00731948	0.00731987	0.00731988
0.30000	0.00900000	0.00900347	0.00900347	0.00900347	0.00900392	0.00900394
0.32000	0.01092267	0.01092812	0.01092812	0.01092812	0.01092864	0.01092865
0.34000	0.01310133	0.01310967	0.01310968	0.01310968	0.01311026	0.01311027
0.36000	0.01555200	0.01556444	0.01556445	0.01556445	0.01556510	0.01556511
0.38000	0.01829067	0.01830883	0.01830885	0.01830885	0.01830957	0.01830959
0.40000	0.02133333	0.02135934	0.02135938	0.02135938	0.02136018	0.02136020
0.42000	0.02469600	0.02473259	0.02473266	0.02473266	0.02473354	0.02473356
0.44000	0.02839467	0.02844535	0.02844546	0.02844546	0.02844643	0.02844645
0.46000	0.03244533	0.03251451	0.03251470	0.03251470	0.03251576	0.03251578
0.48000	0.03686400	0.03695719	0.03695749	0.03695749	0.03695864	0.03695866
0.50000	0.04166667	0.04179067	0.04179114	0.04179115	0.04179240	0.04179243
0.52000	0.04686933	0.04703252	0.04703324	0.04703324	0.04703460	0.04703463
0.54000	0.05248800	0.05270053	0.05270162	0.05270163	0.05270309	0.05270313
0.56000	0.05853867	0.05881281	0.05881445	0.05881445	0.05881602	0.05881607
0.58000	0.06503733	0.06538781	0.06539022	0.06539022	0.06539191	0.06539196
0.60000	0.07200000	0.07244434	0.07244784	0.07244785	0.07244966	0.07244973
0.62000	0.07944267	0.08000165	0.08000667	0.08000669	0.08000863	0.08000870
0.64000	0.08738133	0.08807944	0.08808656	0.08808658	0.08808866	0.08808875
0.66000	0.09583200	0.09669790	0.09670789	0.09670793	0.09671015	0.09671026
0.68000	0.10481067	0.10587781	0.10589169	0.10589175	0.10589413	0.10589426
0.70000	0.11433333	0.11564054	0.11565965	0.11565975	0.11566230	0.11566245
0.72000	0.12441600	0.12600816	0.12603421	0.12603437	0.12603712	0.12603730
0.74000	0.13507467	0.13700344	0.13703868	0.13703892	0.13704189	0.13704210
0.76000	0.14632533	0.14864997	0.14869724	0.14869760	0.14870085	0.14870109
0.78000	0.15818400	0.16097219	0.16103514	0.16103567	0.16103924	0.16103952
0.80000	0.17066667	0.17399548	0.17407871	0.17407948	0.17408346	0.17408379
0.82000	0.18378933	0.18774625	0.18785553	0.18785665	0.18786115	0.18786153
0.84000	0.19756800	0.20225197	0.20239454	0.20239615	0.20240132	0.20240176
0.86000	0.21201867	0.21754133	0.21772617	0.21772847	0.21773452	0.21773503
0.88000	0.22715733	0.23364425	0.23388248	0.23388575	0.23389297	0.23389355
0.90000	0.24300000	0.25059201	0.25089736	0.25090195	0.25091072	0.25091139
0.92000	0.25956267	0.26841737	0.26880664	0.26881305	0.26882388	0.26882466
0.94000	0.27686133	0.28715463	0.28764833	0.28765722	0.28767082	0.28767171
0.96000	0.29491200	0.30683974	0.30746283	0.30747507	0.30749238	0.30749340
0.98000	0.31373067	0.32751044	0.32829315	0.32830991	0.32833216	0.32833333

Рис. 1: пример работы программы

1.00000	0.33333333	0.34920635	0.35018515	0.35020795	0.35023680	0.35023813
1.02000	0.35373600	0.37196911	0.37318784	0.37321869	0.37325631	0.37325782
1.04000	0.37495467	0.39584247	0.39735368	0.39739520	0.39744442	0.39744615
1.06000	0.39700533	0.42087248	0.42273890	0.42279448	0.42285904	0.42286101
1.08000	0.41990400	0.44710756	0.44940388	0.44947791	0.44956267	0.44956490
1.10000	0.44366667	0.47459868	0.47741355	0.47751168	0.47762295	0.47762549
1.12000	0.46830933	0.50339951	0.50683782	0.50696730	0.50711326	0.50711614
1.14000	0.49384800	0.53356655	0.53775209	0.53792217	0.53811338	0.53811665
1.16000	0.52029867	0.56515930	0.57023774	0.57046023	0.57071028	0.57071399
1.18000	0.54767733	0.59824041	0.60438278	0.60467264	0.60499899	0.60500320
1.20000	0.57600000	0.63287589	0.64028242	0.64065858	0.64108361	0.64108837
1.22000	0.60528267	0.66913522	0.67803984	0.67852617	0.67907847	0.67908388
1.24000	0.63554133	0.70709160	0.71776693	0.71839343	0.71910953	0.71911565
1.26000	0.66679200	0.74682208	0.75958516	0.76038944	0.76131585	0.76132279
1.28000	0.69905067	0.78840780	0.80362651	0.80465559	0.80585149	0.80585936
1.30000	0.73233333	0.83193415	0.85003452	0.85134703	0.85288763	0.85289656
1.32000	0.76665600	0.87749101	0.89896540	0.90063428	0.90261501	0.90262515
1.34000	0.80203467	0.92517293	0.95058931	0.95270507	0.95524698	0.95525850
1.36000	0.83848533	0.97507939	1.00509171	1.00776641	1.01102292	1.01103602
1.38000	0.87602400	1.02731499	1.06267489	1.06604696	1.07021244	1.07022735
1.40000	0.91466667	1.08198969	1.12355960	1.12779973	1.13312041	1.13313741
1.42000	0.95442933	1.13921908	1.18798689	1.19330514	1.20009294	1.20011235
1.44000	0.99532800	1.19912458	1.25622012	1.26287447	1.27152473	1.27154694
1.46000	1.03737867	1.26183375	1.32854715	1.33685391	1.34786798	1.34789343
1.48000	1.08059733	1.32748049	1.40528277	1.41562911	1.42964333	1.42967257
1.50000	1.12500000	1.39620536	1.48677133	1.49963038	1.51745349	1.51748716
1.52000	1.17060267	1.46815583	1.57338969	1.58933874	1.61200012	1.61203903
1.54000	1.21742133	1.54348657	1.66555039	1.68529284	1.71410524	1.71415033
1.56000	1.26547200	1.62235975	1.76370521	1.78809688	1.82473826	1.82479071
1.58000	1.31477067	1.70494533	1.86834897	1.89842980	1.94505080	1.94511206
1.60000	1.36533333	1.79142136	1.98002380	2.01705593	2.07642182	2.07649371
1.62000	1.41717600	1.88197432	2.09932376	2.14483718	2.22051689	2.22060169
1.64000	1.47031467	1.97679943	2.22689997	2.28274735	2.37936700	2.37946762
1.66000	1.52476533	2.07610097	2.36346614	2.43188854	2.55547499	2.55559516
1.68000	1.58054400	2.18009264	2.50980470	2.59351036	2.75196140	2.75210601
1.70000	1.63766667	2.28899789	2.66677353	2.76903228	2.97276814	2.97294366
1.72000	1.69614933	2.40305031	2.83531328	2.96006956	3.22294839	3.22316353
1.74000	1.75600800	2.52249393	3.01645537	3.16846368	3.50908868	3.50935538
1.76000	1.81725867	2.64758369	3.21133085	3.39631779	3.83993926	3.84027428
1.78000	1.87991733	2.77858572	3.42117999	3.64603823	4.22738340	4.22781084
1.80000	1.94400000	2.91577783	3.64736281	3.92038323	4.68797913	4.68853470
1.82000	2.00952267	3.05944982	3.89137060	4.22252006	5.24551102	5.24624952
1.84000	2.07650133	3.20990395	4.15483849	4.55609211	5.93541842	5.93642760
1.86000	2.14495200	3.36745536	4.43955924	4.92529790	6.81293511	6.81436317
1.88000	2.21489067	3.53243245	4.74749820	5.33498416	7.96915847	7.97127304
1.90000	2.28633333	3.70517736	5.08080977	5.79075553	9.56580723	9.56913644
1.92000	2.35929600	3.88604643	5.44185526	6.29910416	11.92012148	11.92584304
1.94000	2.43379467	4.07541061	5.83322245	6.86756288	15.75100689	15.76226777
1.96000	2.50984533	4.27365594	6.25774685	7.50488646	23.10913445	23.13727750
1.98000	2.58746400	4.48118407	6.71853497	8.22126628	43.12006260	43.24094179
2.00000	2.66666667	4.69841270	7.21898959	9.02858493	313.03513881	323.63495515

Рис. 2: пример работы программы

4 Ответы на вопросы

1) Каково значение функции при x = 2, т.е. привести значение $u(2)$?

Около 313 +-10.

2) Пояснить, каким образом можно доказать правильность полученного результата при фиксированном значении аргумента в численных методах?

Для того чтобы доказать правильность полученного результата при фиксированном значении аргумента, мы должны сравнить значения Эйлера при разном шаге, так как численные методы зависят от шага. Например возьмём шаг 10^{-4}, получим ответ в районе 310, при шаге 10^{-5} результат будет около 500. А при шаге равном 10^{-6} получим ответ в районе 510. Если сравнивать результаты при шаге 10^{-4} и 10^{-5}, то разница в значении слишком велика, а это нехорошо. Отсюда получаем, чем меньше шаг, тем точнее результат.