Записки за упражненията по

“Числени методи за диференциални уравнения”

спец. Информатика, 2019/2020 година

Тихомир Иванов,
Галина Люцканова-Жекова

Съдържание

Глава 1

Увод

1.1 Какво представляват числените методи?

Най-общо казано, числените методи са техники, чрез които математически задачи се представят във вид, в който могат да бъдат решени с помощта на аритметични операции. Въпреки че има много видове числени методи, те имат обща характеристика – изискват голям брой аритметични пресмятания. Ето защо тяхното прилагане става посредством имплементирането им в компютърни програми.

Обикновено числените методи включват **апроксимация** (т.е. приближение) на оригиналната математическа задача. Ето защо голяма част от тях можем да разглеждаме като техники за **приближеното решаване** на дадена математическа задача посредством аритметични операции.

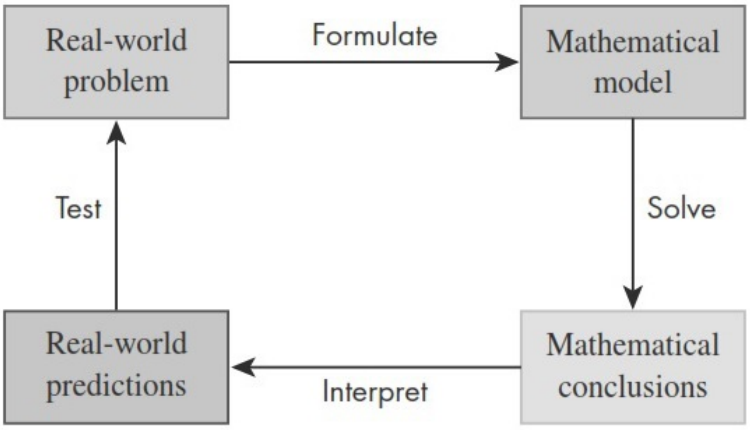
Преди да преминем към разглеждането на конкретни числени методи, нека разгледаме въпроса защо изобщо е необходимо тяхното използване. Математиката е езикът, на който се описват процесите от света около нас. За да изучим даден реален процес или да решим дадена практическа задача, ние трябва да определим кои са основните характеристики, които ги описват – това са някакви величини, които дават информация за съответния процес (например време, скорост, температура, сила, бързодействие на алгоритъм, компресия на данни и др.). Величините се измерват в дадени мерни единици, т.е. им се съпоставят някакви числени стойности. Изучавайки даден процес, ние искаме да изучим зависимостите между величините, които го описват, като за целта създаваме и изследваме математически модел на процеса.

Най-общо казано, **математически модел** е описание на някакъв реален процес или реална задача на езика на математиката. Често това става чрез функция или уравнение (или система от уравнения), много често – диференциално, свързващо величините, описващи процеса. Математическият модел обаче може да представлява и друг математически обект. Например, изследвайки една компютърна мрежа, може да се наложи решаването на задача от теория на графите.

Целта на математическото моделиране е да се опише даденият процес и по-добре да се разберат механизмите, които го обуславят, както и, евентуално, да се направят компютърни симулации и/или предвиждания за бъдещото му поведение.

Често в литературата се дава следната схема, описваща методологията на

математическото моделиране:



Да коментираме накратко етапите, описани в нея.

1. Имайки някаква реална задача, първото, което трябва да направим, е да **формулираме математически модел**, който да я описва. За целта трябва да определим основните величини, които характеризират процеса (от гледна точка на математиката – променливи и параметри), и да съставим математическата задача, която ги свързва (например диференциално уравнение, оптимизационна задача и др.). Важно е да се има предвид, че **всеки математически модел е една абстракция, идеализация на реалния процес**. В него трябва да има баланс – от една страна, моделът трябва достатъчно подробно да описва процеса, така че резултатите от него да бъдат полезни, но, от друга страна, трябва да е достатъчно прост, за да позволява математическо изследване. Всеки модел се базира на някакви допускания (абстракции), които позволяват опростяването на реалната ситуация. При създаването на математически модел използваме физически закони, обуславящи процеса, и математически техники, за да получим уравнения (или други обекти), свързващи променливите. В ситуации, когато не са известни физически закони, които да ни ръководят, може да е необходимо да се съберат данни от експерименти, на базата на които да се състави математическият модел.

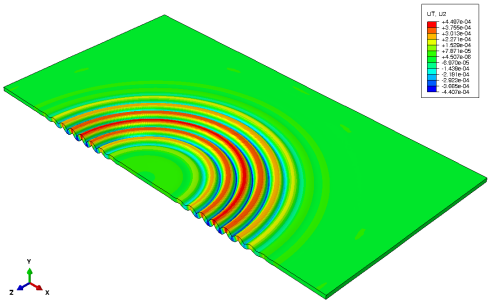
2. Имайки предвид, че математическият модел на един процес представлява математическа задача, **вторият етап е да решим тази задача** и да получим математически заключения. **В настоящия курс ние ще разгледаме именно техники, които ще можем да използваме в този етап**. Важно е да се отбележи, че практическите задачи водят твърде често до математически задачи, които не могат да бъдат решени със стандартните аналитични техники. Както знаем, дори просто изглеждащи алгебрични уравнения като полиномиалните уравнения от пета и по-висока степен в общия случай не могат да бъдат решени точно. Същото се отнася за повечето определени интеграли и др. Въпреки това обаче съществуват техники за тяхното **приближено решаване** и именно с такива ще се запознаем в курса по Числени методи за диференциални уравнения.

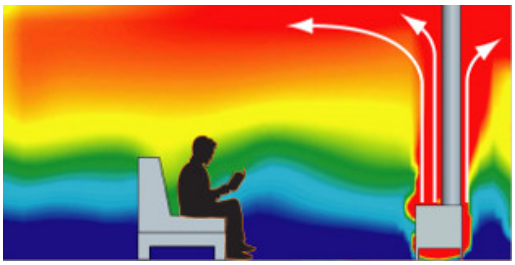
3. След като сме решили (в някакъв смисъл) математическата задача, **следва да интерпретираме резултатите от гледна точка на реалния процес**.

4. Да обърнем внимание, че резултатите за реалния процес, които получихме, са следствие на математическия модел, а не на самия процес. От друга страна, казахме, че математическият модел е една абстракция на реалния процес, т.е. може и да не го описва достатъчно добре. Затова е необходимо да направим **проверка дали тези резултати съответстват на реалността**. Ако това е така, можем да считаме, че моделът ни е удачен. В противен случай се връщаме в началото и трябва да модифицираме модела така, че той да отразява действителността по-добре. С други думи, математическото моделиране е един **итеративен процес**.

1.2 Моделиране с диференциални уравнения

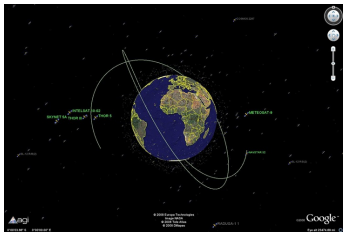
След като се запознахме с методологията на математическото моделиране, нека да се фокусираме върху математическото моделиране с диференциални уравнения. Както споменахме, за да опишем даден реален процес или приложен проблем, ние се стремим да намерим зависимост между основните величини, които го характеризират. Обикновено независимите променливи, които участват в математическия модел, са трите пространствени променливи – x, y, z и времето t.

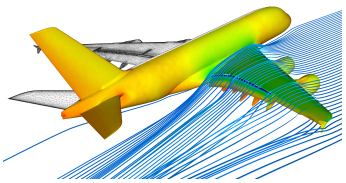


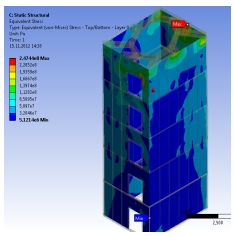


Не е трудно да видим, че нещата около нас се изменят – както с времето, така и при промяната на точката от пространството. От математическа гледна точка, това означава, че повечето математически модели неминуемо ще съдържат производни по отношение на времето и пространствените променливи или по-точно ще представляват закони, свързващи основните величини и техните производни . Това прави естествено разглеждането на диференциалните уравнения като основно средство на математическото моделиране.

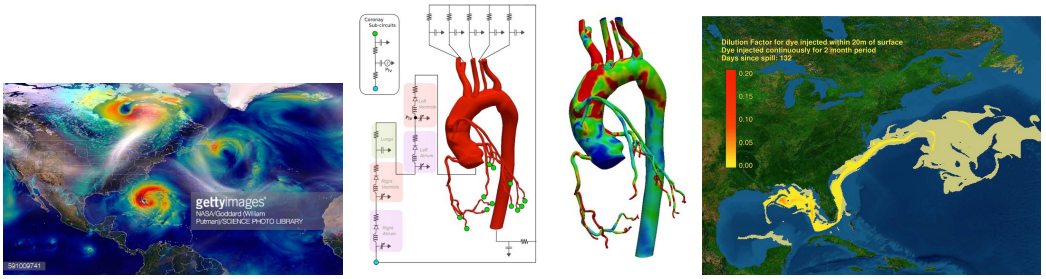
От друга страна, на практика, трудно може да се намери област от науката и технологиите, в която да не се използва апаратът на математическото моделиране.







В реално време се изчисляват траекториите на сателитите. Самолетните инженери симулират обтичането на крилата на самолета от въздушния поток и напреженията, възникващи в неговата структура. Строителните инженери симулират поведението на дадена сграда при различни ситуации. От икономическа гледна точка, тези комютърни симулации позволяват пестенето на милиони, тъй като не е необходимо да се правят прототипи, чрез които да се тестват всички възможни поведения на системата.



Математически модели и компютърни симулации се правят още за предвиждания в областта на метеорологията, за описване на различни процеси в медицината (например потока на кръвта), за опазване на околната среда (например за симулация на развитието на даден петролен разлив) и т.н.

Разбира се, съществуват и други видове математически модели, освен диференциалните уравнения, например в последните години все повече се говори за "big data" и се развиват алгоритми за обработката на тези големи масиви от данни. Все пак обаче диференциалните уравнения описват огромна част от възникващите реални задачи.

1.3 Класове диференциални уравнения и задачи, които те описват

Основните два класа диференциални уравнения са обикновените и частните диференциални уравнения.

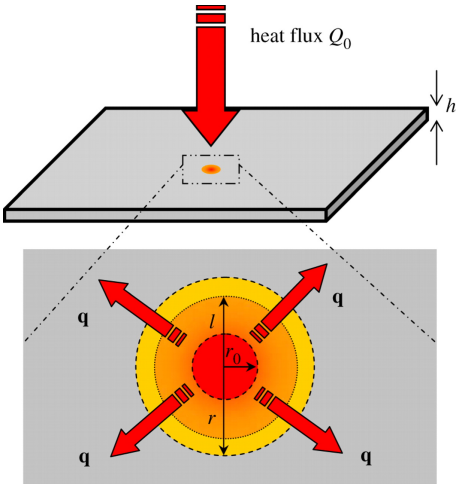
1. **Обикновени диференциални уравнения (ОДУ)**. Това са уравнения, в които участва неизвестна функция на една променлива и нейните производни. Например търсим функция $u(x)$ такава, че

$$u''(x) + u'(x) = x + 1.$$

Разглеждането на ОДУ е естествено например в следните случаи.

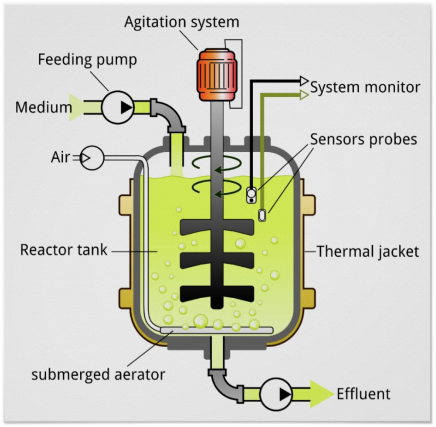
- Разглеждайки стационарното състояние на даден процес, времето не участва като независима променлива.
  - Ако моделираме обект, който може да се разглежда като едномерен (например кабелът на изображението по-долу, който е дълъг и тънък), е естествено неизвестната функция да е на една независима (пространствена) променлива (т.е. търсим $u(x) = ?$).

– Същото се отнася и в случая, когато процесът, който описваме, се характеризира с определена симетрия. Например, ако имаме радиална симетрия, както е в случая с разпространение на топлината по-долу, неизвестната функция ще бъде $u(r)$, където r е разстоянието до центъра.



- Другата принципна ситуация, при която неизвестното е функция на една променлива, е случаят, когато моделираме хомогенен процес. Ако средата е хомогенна, т.е. изследваната величина не зависи от точката в пространството, единствената независима променлива е времето, т.е. търсим $u(t) =?$.

  Като пример можем да дадем процесите в хомогенен биореактор (вж. схемата по-долу). В него има система за хомогенизиране на средата (например бъркалка), която осигурява равномерно достигане на хранителните вещества до организмите, отглеждани в реактора. Тогава концентрацията им ще зависи само от времето, но не и от точката в пространството.



Schematic Structure of a Ba... *by chartsanddiagrams* Zazzle

Често, предположението за хомогенност се прави като първо приближение при моделирането на даден реален процес, който всъщност не е хомогенен.

2. **Частни диференциални уравнения (ЧДУ)**. В частните диференциални уравнения, както подсказва името им, неизвестните са функции на няколко променливи и участват техните частни производни. Например търсим функция $u(x, y)$ такава, че

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0.$$

На този етап ще се ограничим до това да направим класификация на ЧДУ, които ще разглеждаме в курса, по отношение на променливите, участващи в неизвестната функция. В зависимост от това дали участва времевата променлива, или не, говорим за нестационарни и стационарни задачи.

- Стационарните задачи (в които не участва времето t) биват 2-мерни (неизвестната функция е $u(x, y)$) и 3-мерни $u(x, y, z)$. Те описват равновесното състояние на даден процес.
- Нестационарните задачи описват времевата еволюция на даден процес. Те могат да бъдат 1-, 2- или 3-мерни. Съответно неизвестната функция е $u(x, t), u(x, y, t), u(x, y, z, t)$.

Съществуват още диференциални уравнения със закъснения, интегро-диференциални уравнения и др., но те остават извън рамките на настоящия курс.

1.4 Примери за ОДУ

Тук ще разгледаме само няколко примера за ОДУ, които ще използваме по-нататък като модални задачи, върху които ще изложим и изследваме разглежданите числени методи. Повече примери с приложна значимост ще разгледаме в частта “Case Study”. Разнообразни примери могат да бъдат намерени и в [8].

- Важна причина за значимостта на диференциалните уравнения е фактът, че голяма част от законите на физиката, химията и т.н. представляват именно диференциални уравнения. Като пример ще дадем фундаменталния закон на Нютон, на който се основава описването на всяко движение в класическата механика:

$$F(t) = ma(t) \Longrightarrow F(t) = m\frac{d^2u}{dt^2},$$

където m е масата на дадено тяло, $F(t)$ и $a(t)$ са съответно (резултантната) сила, действаща върху тялото, и ускорението в момента от време t, а $u(t)$ е отместването на тялото спрямо началния момент.

И така, Законът на Нютон представлява едно ОДУ от втори ред (т.е. в него участва втората производна на неизвестната функция u).

- Математическото моделиране навлиза все повече в областта на биологията (за повече информация вж. например класическата монография [7]). Ние ще се спрем на два класически модела (моделите на Малтус и Верхълст), описващи развитието на дадена популация, като първо ще изведем модела на Малтус. За целта ще означим с $u(t)$ числеността на популацията в момента от време t. Тогава $u(t + \Delta t)$ ще бъде числеността на популацията в момента от време $t + \Delta t$. Изменението във времевия интервал $[t, t + \Delta t]$ може да се представи в следния вид:

$$u(t + \Delta t) - u(t) = r\Delta t\ u(t), \tag{1.1}$$

където r е коефициентът на естествен прираст за единица време. Можем да разделим двете страни на Δt:

$$\frac{u(t + \Delta t) - u(t)}{\Delta t} = ru(t) \tag{1.2}$$

и ако пуснем Δt да клони към 0, то получаваме следното ОДУ от първи ред по отношение на u:

$$\frac{du}{dt} = ru.$$

Това уравнение, както и всяко друго ОДУ представлява закон, показващ скоростта на изменение на търсената функция и като такова има безброй много решения. За да можем да определим еднозначно едно решение, е необходимо да наложим и начално условие, т.е. в случая да фиксираме стойността на функцията в начален момент, в който тя е известна. Съвкупността от ОДУ и начално условие се нарича задача на Коши. В конкретния случай тя изглежда така

$$\begin{aligned} \frac{du}{dt} &= ru, t \in (t_0, \infty) \\ u(t_0) &= u_0, \end{aligned}$$

където u_0 е константа. Лесно може да се провери, че

$$u(t) = u_0e^{rt}$$

е решение на модела на Малтус. То открива един съществен проблем на този модел – при $r > 0$ той предвижда неограничено нарастване на популацията, което, разбира се, не е реалистично.

- Закон на Верхълст за логистичен растеж. Той се основава на идеята, че съществува максимална численост K, която жизнената среда може да поддържа. Следователно, ако $u = K$, то популацията не би трябвало да се изменя повече, т.е. $du/dt = 0$ при $u = K$. Тя трябва да намалява при $u > K$ и да расте при $u < K$. Тези наблюдения се описват от модела

$$\frac{du}{dt} = ru(K - u), t \in (t_0, \infty)$$
$$u(t_0) = u_0.$$

Точното решение на задачата на Коши е

$$u(t) = \frac{Ku_0e^{Krt}}{K + u_0(e^{Krt} - 1)}.$$

Задача 1. Дадени са данни за развитие на биологичната популация Paramecium aurelia:

брой дни	0	2	4	6	8	10	12	14
бр. Paramecium aurelia (в капка)	4	10	46	66	70	69	71	71

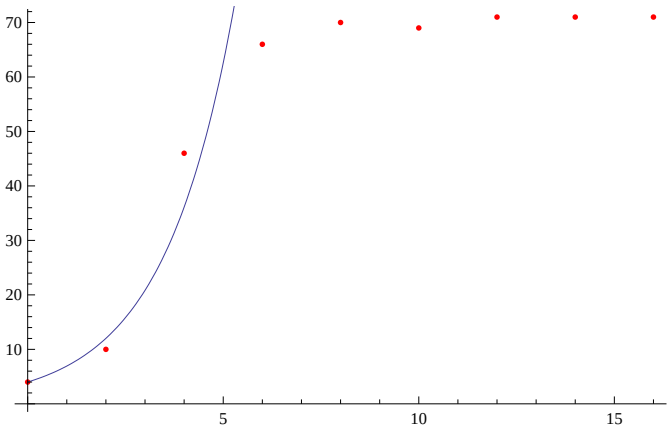
Да се приближат, като се използва:

- моделът на Малтус с коефициент на прираст $r = 0.55$;
- моделът на Верхълст с коефициент на прираст $r = 0.012$.

Решение. От данните в таблицата може да се забележи, че в началния момент е изпълнено $u(0) = 4$ или получаваме следното точно решение за модела на Малтус $u(t) = 4e^{0.55t}$, тъй като $r = 0.55$.

```
In[27]:= plot1 = ListPlot[{{0, 4}, {2, 10}, {4, 46}, {6, 66}, {8, 70}, {10, 69}, {12, 71}, {14, 71}, {16, 71}}, PlotStyle → Red];
         plot2 = Plot[4 * E ^ (0.55 * t), {t, 0, 16}];
         Show[plot1, plot2]
```

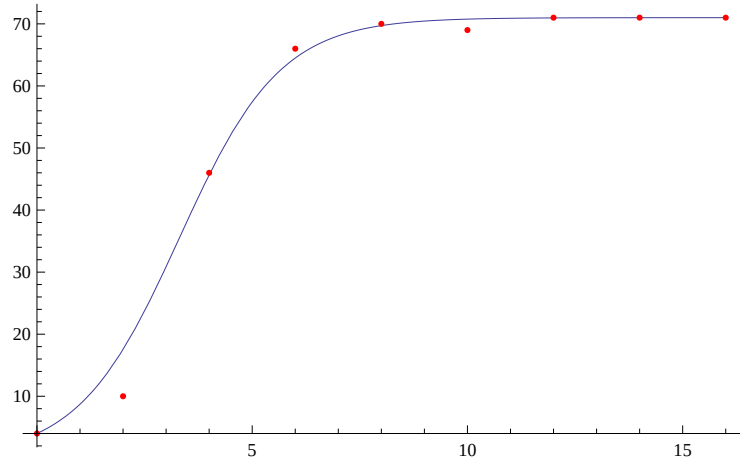
Out[29]=



При модела на Верхълст трябва да пресметнем максималната численост на популацията, която може да поддържа жизнената среда. От данните може да се види, че тя е $K = 71$, тъй като микроорганизмите достигат тази максимална численост и след това не нарастват повече.

In[82]:=
```
plot3 = Plot[(71 * 4 * E ^ (71 * 0.012 * t)) / (71 + 4 * (E ^ (71 * 0.012 * t) - 1)), {t, 0, 16}, PlotRange → All];
Show[plot3, plot1]
```

Out[83]=



Глава 2

Числени методи за ОДУ от първи ред

Твърде малко ОДУ, възникващи в практиката, могат да бъдат решени аналитично. Такива са например уравненията с разделящи се променливи, линейните уравнения и др. (за повече информация – вж. [2, 8]). Ето защо най-често за тяхното решаване е необходимо използването на числени методи. Един начин за численото решаване на ОДУ е да приближим производните в диференциалното уравнение по подходящ начин.

2.1 Приближаване на производни на функция на една променлива.

Производната на функция може да бъде дефинирана като

$$f'(t) = \lim_{\Delta t \to 0} \frac{f(t + \Delta t) - f(t)}{\Delta t}.$$

Тази дефиниция не е възможно да се реализира в компютърната аритметика, тъй като всички числа в компютъра се представят с определен брой значещи цифри след десетичната точка. Следователно бихме могли да приближим производната по следния начин:

$$f'(t) \approx \frac{f(t + \Delta t) - f(t)}{\Delta t}, \tag{2.1}$$

като изберем Δt да бъде фиксирано малко число (например $\Delta t = 0.001$).

Ако положим формално $\Delta t = -\Delta s$ във формула (2.1), то ще получим друг начин за приближаване на производна:

$$f'(t) \approx \frac{f(t + \Delta t) - f(t)}{\Delta t} = \frac{f(t - \Delta s) - f(t)}{-\Delta s} = \frac{f(t) - f(t - \Delta s)}{\Delta s}$$

или

$$f'(t) \approx \frac{f(t) - f(t - \Delta t)}{\Delta t}. \tag{2.2}$$

Като съберем формулите (2.1) и (2.2), получаваме следното:

$$f'(t) \approx \frac{f(t + \Delta t) - f(t - \Delta t)}{2\Delta t}. \tag{2.3}$$

Задача 2. Да се приближи производната на функцията $f(x) = e^x$ в интервала $[0, 10]$, като се използват формули (2.1), (2.2) или (2.3). Да се начертаят на една графика производната и нейните приближения при $\Delta t = 0.5$, $\Delta t = 0.001$ и $\Delta t = 10^{-15}$.

Решение. На графиките по-долу са изобразени производната на e^x (т.е. e^x) в синьо и приближенията ѝ, използвайки съответно формули (2.1), (2.2) и (2.3) с $\Delta t = 0.5$, в оранжево:

- **(f[t+Δt]-f[t])/Δt**

In[81]:=
```
Δt = 0.5;
f[t_] = Exp[t];
DerF[t_] = D[f[t], t];
plotd1 = Plot[{DerF[t], (f[t + Δt] – f[t]) / Δt}, {t, 0, 10}];
```

- **(f[t] - f[t-Δt])/Δt**

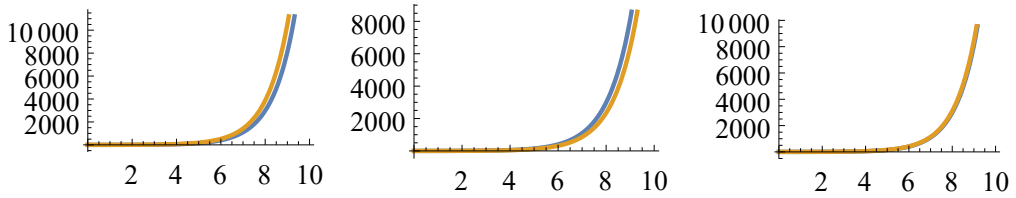
In[85]:=
```
plotd2 = Plot[{DerF[t], (f[t] – f[t – Δt]) / Δt}, {t, 0, 10}];
```

- **(f[x+Δt] - f[x-Δt])/(2Δt)**

In[86]:=
```
plotd3 = Plot[{DerF[t], (f[t + Δt] – f[t – Δt]) / (2 Δt)}, {t, 0, 10}];
```

In[87]:=
```
GraphicsRow[{plotd1, plotd2, plotd3}]
```

Out[87]=



За $\Delta t = 0.05$ формула (2.3) да дава най-добра апроксимация на e^x. Също така в случая формула (2.1) приближава отгоре функцията, а формула (2.2) – отдолу. За произволна функция, ако формула (2.1) приближава функцията отгоре в даден интервал, то формула (2.2) ще я приближава отдолу.

Сега да видим какво се случва за различни стойности на Δt. От дефиницията на производна следва, че при намаляване на Δt би трябвало да се получи по-добро приближение (в дефиницията за производна $\Delta t \to 0$). Резултатите от експериментите за $\Delta t = 0.5$, $\Delta t = 0.001$ и $\Delta t = 10^{-15}$ са показани по-долу:

- **Δt=0.5**

In[8]:=
```
plotStep5 = Plot[{DerF[t], (f[t + 0.5] − f[t]) / 0.5}, {t, 0, 10}];
```

- **Δt=0.001**

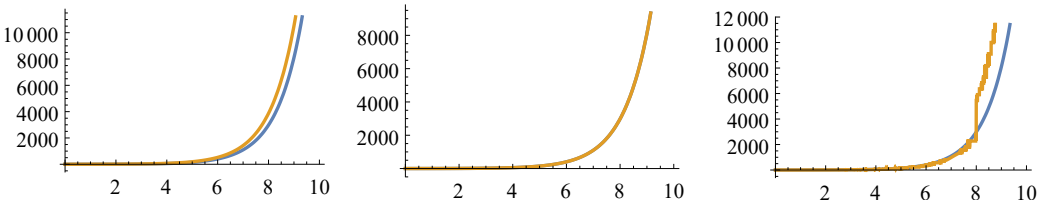
In[9]:=
```
plotStep001 = Plot[{DerF[t], (f[t + 0.001] − f[t]) / 0.001}, {t, 0, 10}];
```

- **Δt=10^(-15)**

In[10]:=
```
plotStepMin15 = Plot[{DerF[t], (f[t + 10^(−15)] − f[t]) / 10^(−15)}, {t, 0, 10}];
```

In[54]:=
```
GraphicsRow[{plotStep5, plotStep001, plotStepMin15}]
```

Out[54]=



Намаляването на грешката се забелява на първите две графики. Но при $\Delta t = 10^{-15}$ полученото приближение не е добро. Причината за това е, че числата в компютъра се представят с фиксиран брой значещи цифри. Това означава, че грешката първоначално ще намалява при намаляване на стъпката, но когато машинната точност бъде достигната (и следователно намаляването на стъпката не може повече да подобрява точността), грешката ще започне да расте заради увеличаването на броя на операциите. Това е и причината за появата на "шума" на третата графика. □

2.2 Абсолютна и относителна грешка. Полином на Тейлър и оценка на грешката.

Както отбелязахме, повечето числените методи включват някаква апроксимация. Ето защо разбирането на идеята за грешка е от много голяма важност за ефективното им използване. Нека да въведем следните дефиниции:

Дефиниция 1. *Абсолютна грешка наричаме разликата между точната и приближената стойност при дадена апроксимация:*

$$\varepsilon_a := exact\ value - approximation.$$

Дефиниция 2. *Относителна грешка дефинираме по следния начин:*

$$\varepsilon_r := \frac{exact\ value - approximation}{exact\ value} = \frac{\varepsilon_a}{exact\ value}.$$

Задача 3. Пресметнете абсолютните и относителните грешке за задача 2.

Решение. Абсолютните и относителните грешки при използването на формули (2.1), (2.2), (2.3) с $\Delta t = 0.001$ за приближение на функцията e^x са изобразени на графиките

- **(f[t+Δt] - f[t])/Δt**

In[114]:=
```
Δt = 0.001;
absErrorD1 = Plot[DerF[t] – (f[t + Δt] – f[t]) / Δt, {t, 0, 10}];
relErrorD1 = Plot[(DerF[t] – (f[t + Δt] – f[t]) / Δt) / DerF[t], {t, 0, 10}];
```

- **(f[t] - f[t-Δt])/Δt**

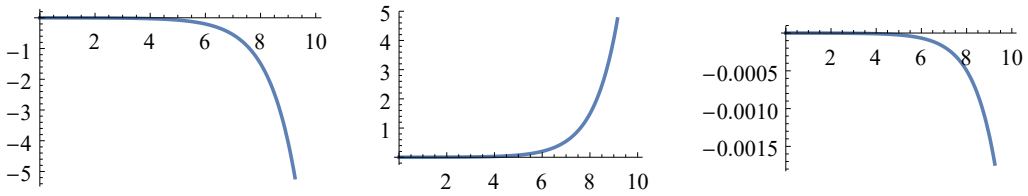
In[117]:=
```
absErrorD2 = Plot[DerF[t] – (f[t] – f[t – Δt]) / Δt, {t, 0, 10}];
relErrorD2 = Plot[(DerF[t] – (f[t] – f[t – Δt]) / Δt) / DerF[t], {t, 0, 10}];
```

- **(f[t+Δt] - f[t-Δt])/(2Δt)**

In[119]:=
```
absErrorD3 = Plot[DerF[t] – (f[t + Δt] – f[t – Δt]) / (2 Δt), {t, 0, 10}];
relErrorD3 = Plot[(DerF[t] – (f[t + Δt] – f[t – Δt]) / (2 Δt)) / DerF[t], {t, 0, 10}];
```

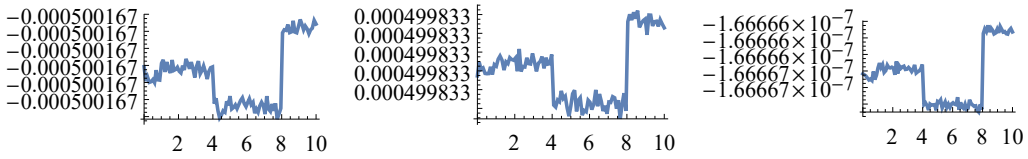
In[121]:=
```
GraphicsRow[{absErrorD1, absErrorD2, absErrorD3}]
```

Out[121]=



In[122]:=
```
GraphicsRow[{relErrorD1, relErrorD2, relErrorD3}]
```

Out[122]=



При една и съща стойност на Δt формула (2.3) дава най-малка абсолютна и относителна грешка, т.е. тя приближава най-добре производната на функция.

Често, като измервател за грешката в даден интервал, се използва максимална абсолютна грешка – колко най-много се различава функцията от нейното приближение в даден интервал. В конкретния случай абсолютната грешка е най-голяма в десния край на интервала. Причината за това е, че функцията e^x расте много бързо. Относителната грешка (или процентната грешка) обаче е приблизително еднаква за целия интервал. □

Абсолютната и относителната грешка представляват добри измерители за това колко е добро едно приближение. В една реална задача обаче обикновено е невъзможно да ги пресметнем, тъй като точното решение не е известно. Затова е важно да можем да оценим грешката, която допускаме при дадена апроксимация. Често в числените методи за диференциални уравнения за оценка

от този вид се използва ред на Тейлър, тъй като много функции могат да се представят чрез развитието си в ред на Тейлър

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k, \tag{2.4}$$

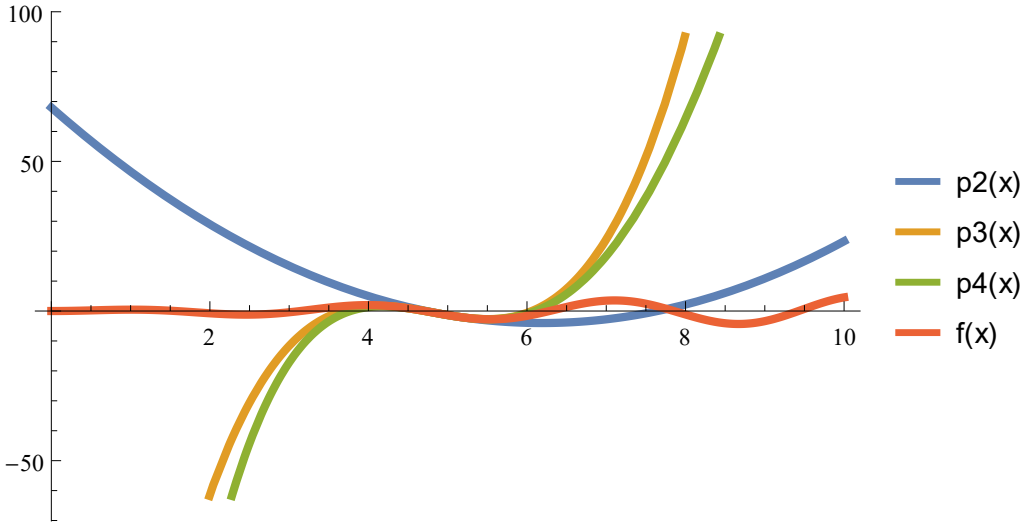
където x_0 е точката, около която развиваме. Тази формула не е възможно да се релизира на компютър, тъй като в нея има сума с безброй много събираеми. Това, което бихме могли да направим, е да приближим функцията с полином от достатъчно голяма степен.

Задача 4. Приближете функцията $x \sin x \cos x$ в интервала $[0, 10]$, като използвате полином на Тейлър от степен 2, 3, 4 около $x_0 = 5$. Начертайте функцията и приближенията на една графика.

Решение. За целта бихме могли да използваме следния код:

In[47]:=
```
f[x_] := Sin[x] * Cos[x] * x
Df1[x_] = D[f[x], {x, 1}];
Df2[x_] = D[f[x], {x, 1}];
Df3[x_] = D[f[x], {x, 1}];

p2[x_, x0_] := f[x0] + f'[x0] (x - x0) + f''[x0] (x - x0)^2 / 2!
p3[x_, x0_] := p2[x, x0] + f'''[x0] (x - x0)^3 / 3!
p4[x_, x0_] := p3[x, x0] + f''''[x0] (x - x0)^4 / 4!

Plot[{p2[x, 5], p3[x, 5], p4[x, 5], f[x]}, {x, 0, 10},
     PlotLegends → {"p2(x)", "p3(x)", "p4(x)", "f(x)"},
     PlotStyle → Table[Thickness[0.01], {i, 4}]]
```

Out[54]=



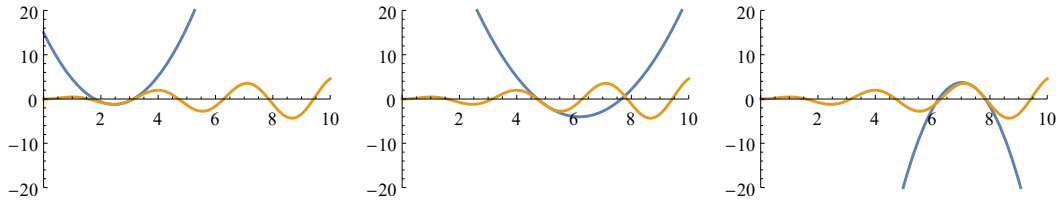
Това, което е важно да отбележим, че при увеличане на степента на полинома, той се „залепва“ все по-добре за функцията около точката, в която развиваме.

Друг интересен въпрос е какво се случва, ако x_0 се променя. Отговор на този въпрос дават следващите графики. На тях е изобразена функцията (в

оранжево) полинома на Тейлър от втора степен, построен с развитие около точките $x_0 = 2.5$, $x_0 = 5$ и $x_0 = 7.5$ (в синьо):

```
In[58]:= GraphicsRow[{Plot[{p2[x, 2.5], f[x]}, {x, 0, 10}, PlotRange → {{0, 10}, {-20, 20}}],
                      Plot[{p2[x, 5], f[x]}, {x, 0, 10}, PlotRange → {{0, 10}, {-20, 20}}],
                      Plot[{p2[x, 7.5], f[x]}, {x, 0, 10}, PlotRange → {{0, 10}, {-20, 20}}]}]
```

Out[58]=



От графиките може да се заключи, че апроксимацията е най-добра за точки, които са близо до точката x_0 (като в нея грешката е 0). Следователно, ако искаме да получим добри резултати в даден интервал, е добра идея да развиваме около средата на този интервал. □

След този кратък обзор върху формулата на Тейлър, сме готови да преминем към темата за оценка на грешката.

Задача 5. Намерете оценка за абсолютната грешка на апроксимациите в задача 2.

Решение. Първо, ще изведем три теоретични оценки на грешката. За целта, ще намерим разликата между точното и приближеното решение и ще развием в ред на Тейлър около точката t. По този начин получаваме следните оценки за формули (2.1), (2.2) и (2.3):

$$err_1 = f'(t) - \frac{f(t+\Delta t) - f(t)}{\Delta t} = f'(t) - \frac{f(t) + \Delta t f'(t) + O(\Delta t^2) - f(t)}{\Delta t} = O(\Delta t)$$

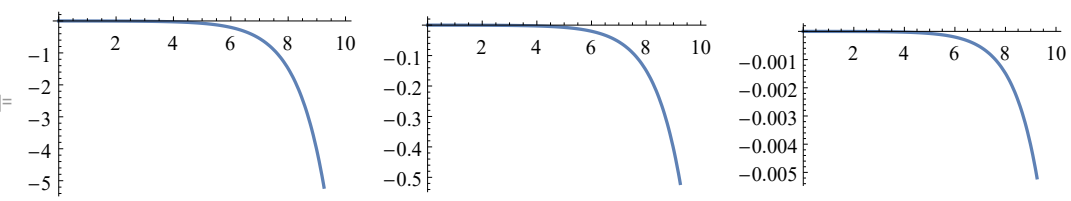
$$err_2 = f'(t) - \frac{f(t) - f(t-\Delta t)}{\Delta t} = f'(t) - \frac{f(t) - (f(t) - \Delta t f'(t) + O(\Delta t^2))}{\Delta t} = O(\Delta t)$$

$$err_3 = f'(t) - \frac{f(t+\Delta t) - f(t-\Delta t)}{\Delta t} = O(\Delta t^2)$$

Абсолютната грешка, която получаваме за формули (2.1) и (2.2), е $O(\Delta t)$. Това означава, че ако намалим Δt десет пъти, грешката ще намалее от порядъка на 10 пъти. За да верифицираме този теоретичен резултат, нека да начертаем графиките на абсолютните грешки при $\Delta t = 0.001$, $\Delta t = 0.0001$, $\Delta t = 0.000001$,

```
In[47]:= f[x_] := E^x
         absErrorD1Step001 = Plot[DerF[t] - (f[t + 0.001] - f[t]) / 0.001, {t, 0, 10}];
         absErrorD1Step0001 = Plot[DerF[t] - (f[t + 0.0001] - f[t]) / 0.0001, {t, 0, 10}];
         absErrorD1Step000001 = Plot[DerF[t] - (f[t + 0.000001] - f[t]) / 0.000001, {t, 0, 10}];
         GraphicsRow[{absErrorD1Step001, absErrorD1Step0001, absErrorD1Step000001}]
```

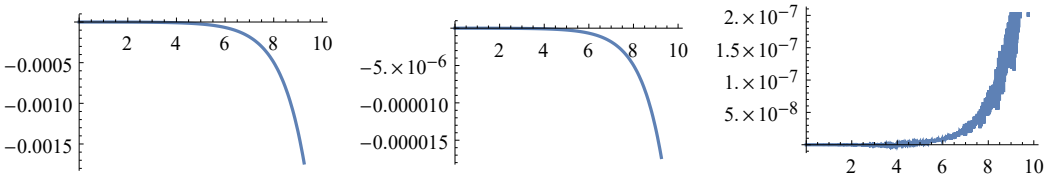
Out[51]=



Аналогично получаваме, че абсолютната грешка при апроксимиране на производната на функцията с формулата (2.3) е $O(\Delta t^2)$. Това означава, че ако намалим Δt десет пъти, грешката ще намалее от порядъка на 100 пъти. При $\Delta t = 0.001$, $\Delta t = 0.0001$, $\Delta t = 0.00001$ получаваме следните графики за абсолютната грешка:

```
In[60]:= absErrorD3Step001 = Plot[DerF[t] - (f[t + 0.001] - f[t - 0.001]) / 0.002, {t, 0, 10}];
         absErrorD3Step0001 = Plot[DerF[t] - (f[t + 0.0001] - f[t - 0.0001]) / 0.0002, {t, 0, 10}];
         absErrorD3Step00001 = Plot[DerF[t] - (f[t + 0.00001] - f[t - 0.00001]) / 0.00002, {t, 0, 10}];
         GraphicsRow[{absErrorD3Step001, absErrorD3Step0001, absErrorD3Step00001}]
```

Out[63]=



Разбира се, тук е мястото да отбележим, че ако изберем много малко Δt грешките от закръгляване ще започнат да доминират над точността на апроксимацията. □

2.3 Задача на Коши за ОДУ от първи ред. Обща теория.

Започваме изучаването на числените методи за решаване на диференциални уравнения с тези за решаване на ОДУ от първи ред, т.е. ще разглеждаме ОДУ от вида

$$\frac{du}{dt} = f(t, u(t)).$$

Уравнения от по-висок ред можем да сведем до такива от първи ред чрез полагане. Например Законът на Нютон

$$\frac{d^2u}{dt^2} = \frac{F(t)}{m}$$

може да се сведе до система ОДУ от първи ред чрез полагането $du/dt = v$. Получаваме системата

$$\begin{aligned} \frac{du}{dt} &= v, \\ \frac{dv}{dt} &= \frac{F(t)}{m}. \end{aligned}$$

От друга страна, системи уравнения от първи ред можем да разглеждаме като едно векторно уравнение. Горния пример можем да запишем във вида

$$\frac{d}{dt}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} v \\ F(t)/m \end{bmatrix}$$

или още

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{u}(t)),$$

където $\mathbf{u} = (u, v)^T \in \mathbb{R}^2$ е вектор и $\mathbf{f}(t, \mathbf{u}) = (v, F(t)/m)^T$ е векторна функция $\mathbf{f} : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}^2$.

Казаното дотук ни дава основание да разглеждаме именно уравнения от първи ред. Общият им вид е

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{u}(t)),$$

където $\mathbf{u} \in \mathbb{R}^n$ и $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$.

Ясно е, че ако една функция $u(t)$ е решение на дадено ОДУ, то и всяка функция от вида $v(t) = u(t) + c$ за някоя константа c ще е решение, тъй като $u'(t) \equiv v'(t)$. Тогава, за да бъде решението определено еднозначно, е необходимо (макар и не достатъчно) да знаем поне една точка от него.

И така, **общата задача, която ще разглеждаме, е т.нар. задача на Коши**:

$$\begin{aligned} &\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{u}(t)), \quad t \in (t_0, +\infty) \\ &\mathbf{u}(t_0) = \mathbf{u}_0. \end{aligned}$$

Решение на задачата на Коши наричаме непрекъснато диференцируема функция $\mathbf{u} : \mathbb{R} \to \mathbb{R}^n$, която удовлетворява уравнението и началното условие.

2.4 Идея на диференчните методи за задачата на Коши за ОДУ от първи ред

Както казахме, най-общо, задачата на Коши за обикновено диференциално уравнение или система от ОДУ от първи ред има вида

$$\begin{aligned} &\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, \mathbf{u}(t)), \ t \in (t_0, +\infty), \\ &\mathbf{u}(t_0) = \mathbf{u}_0, \end{aligned} \tag{2.5}$$

където в общия случай неизвестната функция $\mathbf{u}(t)$ е векторна функция $\mathbf{u} : \mathbb{R} \to \mathbb{R}^n$. Ще приемаме, че функцията $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ е достатъчно гладка, така че съществува единствено решение на задачата на Коши (2.5) и могат да бъдат направени използваните по-нататък Тейлърови развития на $\mathbf{f}$.

Компютърът, разбира се, не може да прави безкрайни пресмятания. С други думи, трябва да знаем докога да интегрираме.

- В някои задачи ще искаме да намерим решението в предварително известен интервал $t \in [t_0, T]$.
- В други задачи може да искаме да интегрираме, докато е изпълнено предварително зададено условие, например докато стойността на u стане 0.
- В някои проблеми искаме да знаем какво е поведението на решенията при $t \to \infty$. В този случай трябва да интегрираме дотогава, когато стане ясно какво е поведението за достатъчно големи времена. Засега ще оставим този доста общ отговор за този случай и ще се върнем на него по-късно.

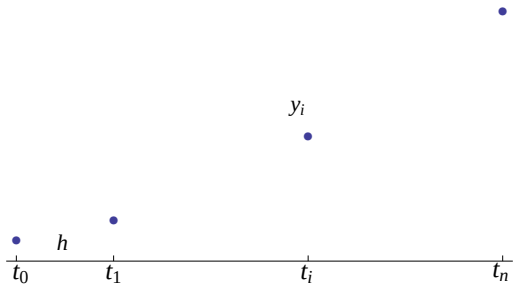
Да обърнем внимание, че променливите t и u са непрекъснати. Нашата цел ще бъде да ги заменим с дискретни, така че задачата да бъде сведена до алгебрична и следователно да може да бъде решена с помощта на стандартни числени методи. Основният въпрос е как тази „замяна" да се направи така, че да получим добро приближение на решението на оригиналната задача на Коши.

Въвеждаме мрежата

$$\overline{\omega}_h = \{t_i = t_0 + ih,\ i = \overline{0,n},\ n = (T - t_0)/h\}.$$

Ще търсим приближените стойности на решението именно в точките от тази мрежа. Стойността на приближеното решение в точката t_i ще бележим с y_i, т.е. искаме

$$y_i \approx u(t_i) = u_i.$$



За построяване на числена схема за намиране на стойностите на приближеното решение можем да подходим, като приближим производната в оригиналната диференциална задача (2.5), използвайки някоя от формулите (2.1), (2.2) или (2.3).

2.5 Методи на Ойлер

2.5.1 Явен и неявен метод на Ойлер

Явният метод на Ойлер е най-простият метод за числено решаване на ОДУ. Използвайки формулата (2.1), която е известна като формула за числено диференциране с разлика напред:

$$u'(t_i) \approx \frac{u(t_i + h) - u(t_i)}{h} = \frac{u(t_{i+1}) - u(t_i)}{h}.$$

И така бихме могли да приближим диференциалната задача (2.5) върху мрежата $\overline{\omega}_h$ с алгебричната задача

$$\begin{aligned} &\frac{y_{i+1} - y_i}{h} = f(t_i, y_i), \quad i = \overline{0, n-1}, \\ &y_0 = u_0. \end{aligned}$$

Тогава за пресмятанията на приближеното решение можем да използваме схемата

$$y_0 = u_0, \quad y_{i+1} = y_i + hf(t_i, y_i), \quad i = \overline{0, n-1}. \tag{2.6}$$

Задача 6. Като използвате явния метод на Ойлер, решете моделното уравнение

$$\frac{du}{dx} = -10u, \quad 0 < x \leq 1,$$
$$u(0) = 1$$

при $n = 4, 20, 100$ и сравнете с точното решение

$$u(x) = e^{-10x}.$$

Решение. За конкретната задача, (2.6) добива вида

$$y_0 = 1, \quad y_{i+1} = y_i - 10hy_i, \quad i = \overline{0, n-1}.$$

Имплементираме числената схема в Mathematica.

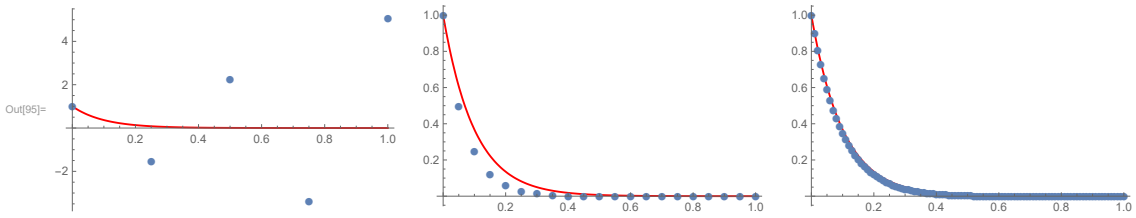
```
In[137]:= explicitEulerMP[n_] := (
           h = 1 / n;
           y = Table[0, {n + 1}];

           y[[1]] = 1;
           For[i = 1, i < n + 1, i++,
            y[[i + 1]] = y[[i]] - 10. * h * y[[i]]
           ];

           y
          )
```

Като използваме реализираната функция, ще изобразим решенията за различни стойности на n.

```
In[51]:= Animate[
          exact[x_] := E^(-10 x);
          plotExact = Plot[exact[x], {x, 0, 1}, PlotStyle → Red, PlotRange → All];
          times = Table[i * 1 / n, {i, 0, n}];
          appr = explicitEuler[n];
          plotAppr = ListPlot[Transpose[{times, appr}], PlotMarkers → ●];
          plotError = ListPlot[Transpose[{times, exact[times] - appr}], PlotRange → {{0, 1}, {0, 1}}, PlotMarkers → ●];

          GraphicsRow[{Show[plotExact, plotAppr], plotError}],
          {n, 4, 100, 4}]
```

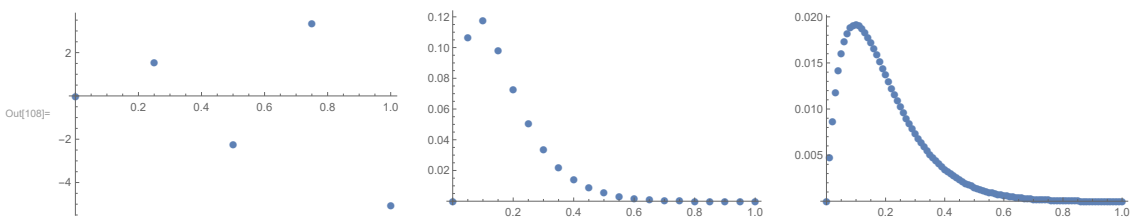
Решенията за $n = 4, 10, 100$ са показани по-долу, като червената крива изобразява точното решение.

Out[95]=



Грешките по абсолютна стойност в точките от мрежата $\overline{\omega}_h$ са изобразени по-долу:

Out[108]=



Виждаме, че с увеличаване на броя на възлите приближеното решение започва да се приближава към точното и съответно абсолютната грешка по модул намалява. Това илюстрира първото важно свойство, което искаме да притежава даден числен метод – да бъде **сходящ към точното решение**, т.е. при $n \to \infty$ грешката да клони към 0.

□

Неявният метод на Ойлер се основава на формулата (2.2) за числено диференциране с разлика назад

$$u'(t_i) \approx \frac{u(t_i) - u(t_i - h)}{h} = \frac{u(t_i) - u(t_{i-1})}{h}.$$

Следователно апроксимацията на (2.5) има вида

$$\begin{aligned} &\frac{y_i - y_{i-1}}{h} = f(t_i, y_i), \quad i = \overline{1,n}, \\ &y_0 = u_0. \end{aligned}$$

Тогава за пресмятанията на приближеното решение можем да използваме схемата

$$y_0 = u_0, \quad y_i = y_{i-1} + hf(t_i, y_i), \quad i = \overline{1,n}. \tag{2.7}$$

За да получим стойността на приближеното решение в t_i, трябва да решим едно (в общия случай нелинейно) уравнение относно y_i, тъй като y_i участва и в дясната страна.

Задача 7. Да се реши задачата на Коши от задача 6, като се използва неявният метод на Ойлер.

Решение. Започвайки с началното условие $y_0 = 1$, на всяка следваща стъпка трябва да решим уравнението

$$y_i = y_{i-1} - 10hy_i, \quad i = \overline{1,n},$$

в което неизвестното е y_i. Засега ще използваме вградената в Mathematica функция FindRoot, за да решим съответните уравнения.

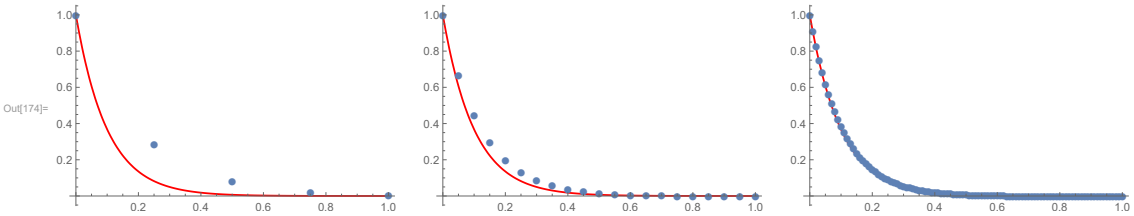
Повечето методи за решаване на нелинейни уравнения са итерационни и се нуждаят от добро начално приближение. Един възможен начин да получим такова е, като използваме явния метод на Ойлер:

$$y_{i,init} = y_{i-1} - 10hy_{i-1}.$$

Имплементираме числената схема в Mathematica.

```
In[21]:= implicitEulerMP[n_] := (
          h = 1 / n;
          y = Table[0, {n + 1}];

          y[[1]] = 1;
          For[i = 1, i < n + 1, i++,
           initialGuess = y[[i]] + h * (-10) y[[i]];
           y[[i + 1]] = yNew /. FindRoot[yNew == y[[i]] + h * (-10) yNew, {yNew, initialGuess}]
          ];

          y
         )
```

Решенията за $n = 4, 20, 100$ са показани по-долу, като червената крива изобразява точното решение.



Виждаме отново, че при увеличаване на n приближеното решение доближава точното. □

За магистрите: Съществуват различни числени методи за приближеното решаване на алгебрични уравнения. Най-често използваният е т.нар. метод на Нютон или метод на допирателните. Той е итерационен метод, т.е. имайки едно начално приближение x_0 на корена на уравнението

$$g(x) = 0,$$

построяваме редица $x_0, x_1, x_2, \ldots$, която искаме да е сходяща към точното решение.

1. Нека имаме едно първоначално приближение x_0. Да построим допирателната към графиката на функцията $g(x)$ в точката $(x_0, g(x_0))$. Имаме

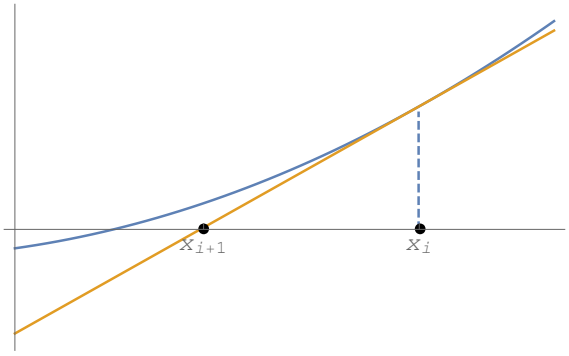
$$g'(x_0) = \operatorname{tg} \varphi = \frac{g(x_0) - 0}{x_0 - x_1}$$

Тогава

$$x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}$$

2. Аналогично намираме следващите приближения по формулата

$$x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)}, \ i = 0, 1, \ldots$$



3. Правим това, докато разликата в две съседни приближения стане по-малка от ϵ (отнапред зададена точност) или надхвърлим предварително зададен брой итерации.

Прилагаме примерна имплементация на метода:

```
In[23]:= Newton[g_, x0_, tol_, maxIter_] := (
          xCurrent = x0;
          xNext = xCurrent - g[xCurrent] / g'[xCurrent];
          iter = 1;

          While[Abs[xCurrent - xNext] ≥ tol && iter < maxIter,
           xCurrent = xNext;
           xNext = xCurrent - g[xCurrent] / g'[xCurrent];
           iter++;
          ];

          xNext
         )
```

Тогава, имайки предвид, че уравнението, което решаваме на всяка стъпка от неявния метод на Ойлер за задачата на Коши от задача 6, има вида

$$g(x) := x - y_{i-1} + 10hx = 0,$$

модифицираме програмата, предложена в предходната задача:

```
In[27]:= implicitEulerWithNewtonMP[n_] := (
          h = 1 / n;
          y = Table[0, {n + 1}];

          y[[1]] = 1;
          For[i = 1, i < n + 1, i++,
           initialGuess = y[[i]] + h * (-10) y[[i]];
           LS[yNew_] := yNew - ( y[[i]] + h * (-10) yNew);
           y[[i + 1]] = Newton[LS, initialGuess, 0.0000001, 100]
          ];

          y
         )
```

2.5.2 Локална и глобална грешка на апроксимация, сходимост, A-устойчивост, монотонност

Ясно е, че за да можем да използваме на практика разглежданите числени методи, трябва да знаем нещо за грешката, която получаваме при приближеното решаване на съответното диференциално уравнение.

При численото решаване на оригиналната диференциална задача има два източника на грешка:

- грешка от апроксимация - поради факта, че вместо оригиналната диференциална задача, решаваме приближена алгебрична задача;
- грешка от закръгляване - поради представянето на числата в компютъра.

Ще разгледаме ефектите от тези две грешки поотделно. Първо, нека приемем, че няма грешки от закръгляване. Искаме да оценим грешката $u_i - y_i$.

За тази цел първо ще въведем следната дефиниция.

Дефиниция 3. *Локална грешка на апроксимация (ЛГА) за диференчния метод*

$$\frac{y_{i+1} - y_i}{h} = \theta_i(t_i, y_i, y_{i+1}) \tag{2.8}$$

наричаме разликата между лявата и дясната страна, пресметнати за точното решение, т.е.

$$\psi_{h,i} := \frac{u_{i+1} - u_i}{h} - \theta_i(t_i, u_i, u_{i+1}).$$

С други думи, ЛГА ни дава информация за това каква грешка внасяме локално, **за една стъпка от метода**. Приемаме, че до i-тата точка сме намерили решението точно, и искаме да видим каква грешка ще получим само в i-тата стъпка.

Задача 8. Да се намери ЛГА на апроксимация на явния и неявния метод на Ойлер.

Решение. За явния метод на Ойлер имаме

$$\psi_{h,i} = \frac{u_{i+1} - u_i}{h} - f(t_i, u_i).$$

Тъй като всичко в горния израз е пресметнато в i-тата точка, освен u_{i+1}, ще изразим u_{i+1}, като развием функцията $u(t)$ в ред на Тейлър около точката t_i.

Получаваме

$$u_{i+1} = u(t_i + h) = u_i + u_i' h + O(h^2).$$

Тогава, като използваме, че $u_i' = f(t_i, u_i)$, получаваме последователно

$$\psi_{h,i} = \frac{1}{h}(\cancel{u_i} + \cancel{u_i' h} + O(h^2) - \cancel{u_i}) - \cancel{f(t_i, u_i)} = O(h).$$

Аналогично, за неявния метод на Ойлер получаваме

$$\psi_{h,i+1} = \frac{u_{i+1} - u_i}{h} - f(t_{i+1}, u_{i+1})$$
$$\frac{1}{h}(\cancel{u_{i+1}} - (\cancel{u_{i+1}} - \cancel{u_{i+1}' h} + O(h^2))) - \cancel{f(t_{i+1}, u_{i+1})} = O(h).$$

Тук използвахме развитието $u_i = u(t_{i+1} - h) = u_{i+1} - u_{i+1}' h + O(h^2)$. □

Локалната грешка на апроксимация е ключова за определяне на глобалната грешка $u_i - y_i$ (която всъщност ни интересува), предвид следната теорема.

Теорема 1. *Ако методът (2.8) има ЛГА от ред p (т.е. $O(h^p)$), то той е сходящ към точното решение със същата скорост, т.е. $\|\boldsymbol{u} - \boldsymbol{y}\| = O(h^p)$ при $h \to 0$.*

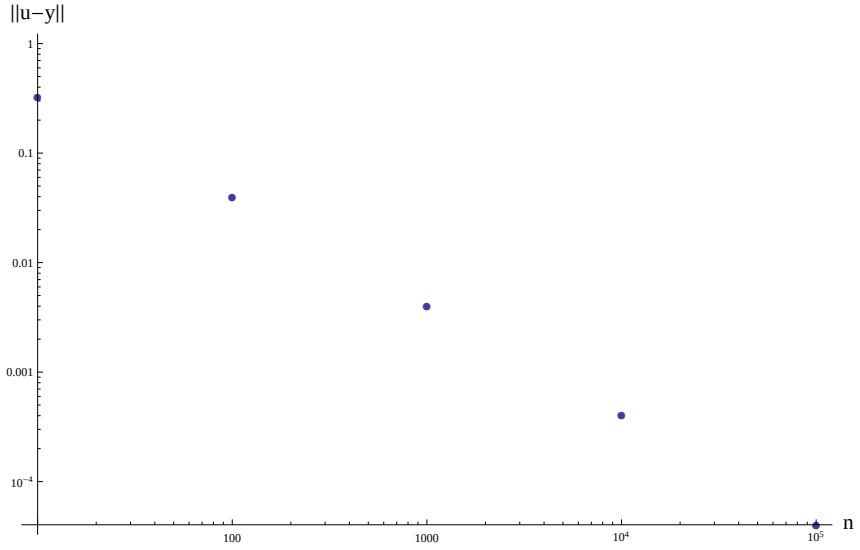
Доказателство. Вж.[3] □

И така, **явният и неявният метод на Ойлер имат първи ред на сходимост**.

Действително, нека разгледаме зависимостта на максималната грешка

$$\|u - y\| := \max_{i=\overline{0,n}} |u_i - y_i|$$

от броя интервали n, например за явния метод на Ойлер, за логистичното уравнение, върху което изложихме методите.



Виждаме, че скоростта, с която намалява грешката, е от първи ред – увеличавайки 10 пъти броя на интервалите, n, грешката намалява също от порядъка на 10 пъти.

Теоретично, сходимостта на един метод означава, че можем да получим произволна точност, стига да изберем достатъчно голямо n. Разбира се, на практика точността е ограничена от максималната точност, с която работим в компютърна аритметика.

Нека разгледаме още два примера за определяна на ЛГА.

Задача 9. Да се пресметне ЛГА за следните диференчни уравнения:

- $\frac{y_{i+1}-y_i}{h} = \frac{1}{3h}(y_i - y_{i-1}) + \frac{2}{3}f_i$,
- $\frac{y_{i+1}-y_i}{h} = \frac{1}{2}(f_{i+1} + f_i)$.

Решение. • В първия случай ЛГА е

$$\psi_h = \frac{1}{h}(u_{i+1} - u_i) - \frac{1}{3h}(y_i - y_{i-1}) + \frac{2}{3}f_i.$$

Развиваме около i-тата точка и получаваме последователно

$$\psi_h = \frac{1}{h}\left(\underbrace{u_i + \frac{u_i'}{1!}h + \frac{u_i''}{2!}h^2 + O(h^3)}_{u_{i+1}} - u_i\right) - \frac{1}{3h}\left(u_i - \underbrace{\left(u_i - \frac{u_i'}{1!}h + \frac{u_i''}{2!}h^2 + O(h^3)\right)}_{u_{i-1}}\right)$$
$$+ \frac{2}{3}f_i = \frac{2h}{3}u_i'' + O(h^2) = O(h).$$

- Във втория пример ЛГА е

$$\begin{aligned}\psi_h &= \frac{1}{h}(u_{i+1}-u_i)-\frac{1}{2}(u'_{i+1}+u'_i)\\ &= \frac{1}{h}\left(\underbrace{u_i+\frac{u'_i}{1!}h+\frac{u''_i}{2!}h^2+\frac{u'''_i}{3!}h^3+O(h^4)}_{u_{i+1}}-u_i\right)\\ &\quad -\frac{1}{2}\left(\underbrace{\left(u'_i+\frac{u''_i}{1!}h+\frac{u'''_i}{2!}h^2+O(h^3)\right)}_{u'_{i+1}}+u'_i\right)\\ &= -\frac{h^2}{6}u'''_i+O(h^3)=O(h^2).\end{aligned}$$

  Методът, зададен в тази подточка, е известен като подобрен метод на Ойлер.

□

И така, ЛГА ни позволява да определим реда на сходимост за даден метод (от вида (2.8)). От друга страна, **това, че за достатъчно голямо n грешката клони към 0, далеч не означава, че за по-малки стойности на n поведението на метода ще е добро.**

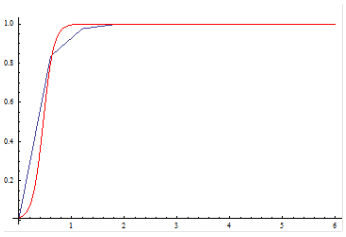
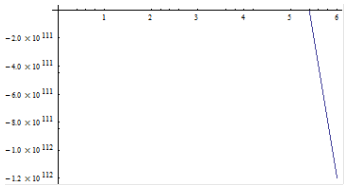
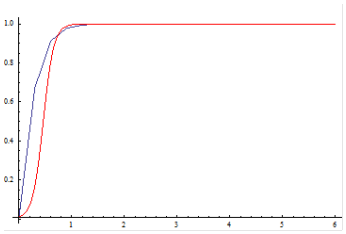
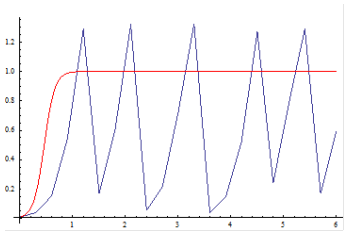
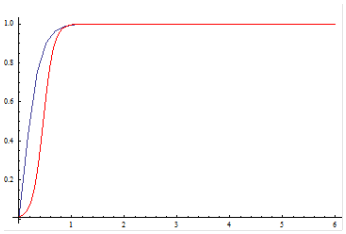
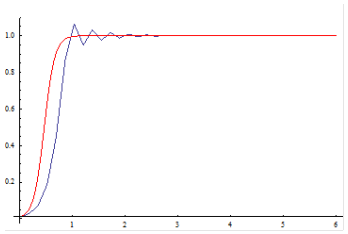
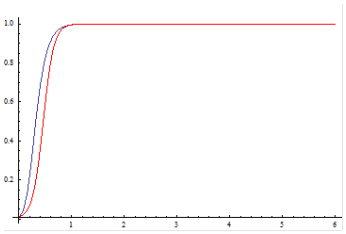
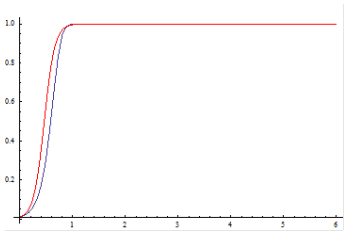
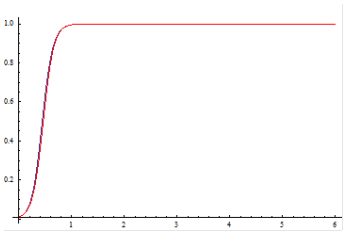
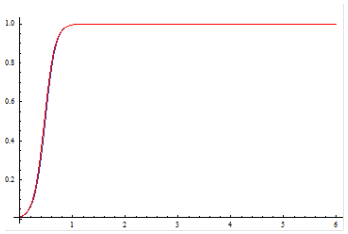
Да разгледаме следния пример.

Задача 10. Да се реши задачата на Коши

$$\begin{aligned}&\frac{du}{dt}=10u(1-u),\ t\in(0,6],\\ &u(0)=0.1\end{aligned}\tag{2.9}$$

с явния и неявния метод на Ойлер за $n=10,20,35,75,500$.

Решение. Привеждаме резултатите от решението на задачата по-долу.

n	Неявен Ойлер	Явен Ойлер
10		
20		
35		
75		
500		

□

И така, очевидно явният метод на Ойлер няма добро поведение за малки стойности на n. Такъв проблем не се наблюдава при неявния метод на Ойлер. Това е свързано с понятията абсолютна устойчивост (A-устойчивост) и монотонност на методите.

За да въведем тези понятия, първо трябва да изясним някои свойства на решенията на едно ОДУ. Ще изложим идеите върху логистичното уравнение

$$\frac{du}{dt} = 10u(1-u).$$

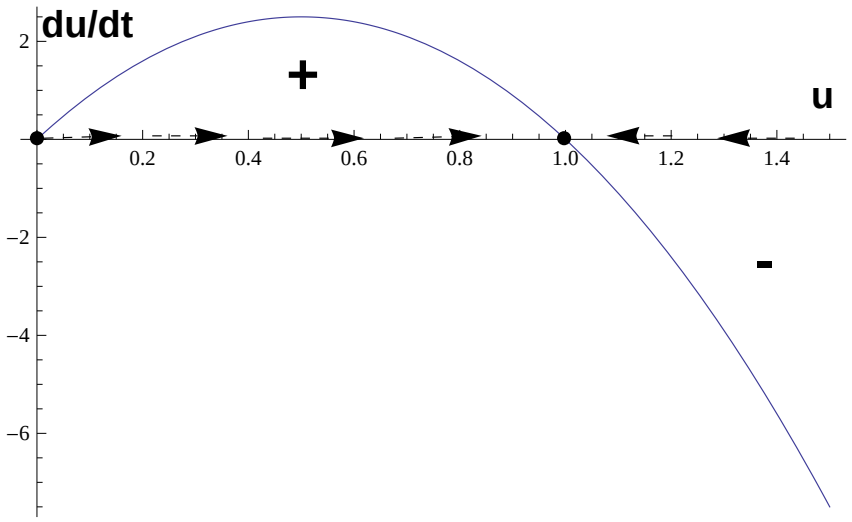
Равновесните точки на едно автономно диференциално уравнение

$$\frac{du}{dt} = f(u)$$

са точките $u = \xi$, за които $f(\xi) = 0$.

Равновесните точки определят еднозначно асимптотичното поведение на решенията (т.е. какво се случва с решенията в крайна сметка, след достатъчно дълъг период от време).

За логистичното уравнение очевидно равновесните точки са $u = 0$ и $u = 1$. Нека разгледаме графиката на du/dt, т.е. параболата $10u(1-u)$:



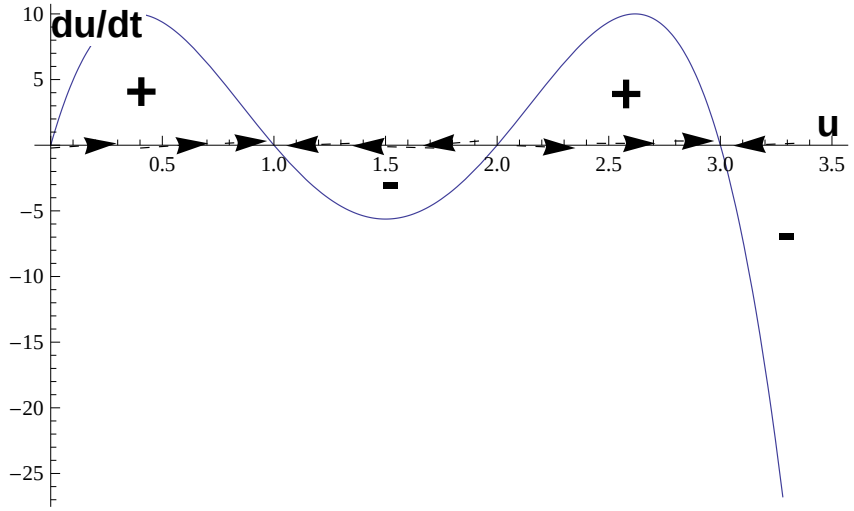
Тъй като производната на u е положителна за $u \in (0, 1)$, то ако u е в този интервал функцията е растяща. В противен случай тя е намаляваща.

От горната фигура лесно се вижда, че всички решения клонят към равновесната точка $u = 1$.

Казваме, че една равновесна точка ξ е асимптотично устойчива, ако за всички достатъчно близки до нея начални условия u_0 следва, че решението u клони към ξ. В противен случай равновесната точка се нарича неустойчива.

За логистичното уравнение точката $u = 0$ е неустойчива, защото всяко изменение на $u > 0$ води до решение, клонящо към $u = 1$. От друга страна точката $u = 1$ е асимптотично устойчива, тъй като при изменение на u решението "се връща" към единицата.

Лесно се съобразява, че за едно автономно уравнение решенията винаги клонят към асимптотично устойчива равновесна точка. Да разгледаме следната диаграма:



От фигурата е ясно, че решенията клонят или към $u = 1$, или към $u = 3$.

От гледна точка на числените методи, това поведение е от съществено значение. Възстановяването на неустойчиви решения с помощта на числени методи е лошо обусловена задача, тъй като всяка грешка от закръгляване означава, че решението, което ще се получи, ще е различно (ще клони към устойчива равновесна точка).

От друга страна, грешките от закръгляване не би трябвало да имат голямо влияние върху решенията, клонящи към устойчивите равновесни точки, тъй като $|u-\xi| \to 0$ при $t \to \infty$. С други думи, грешките от закръгляване, допуснати в даден момент от време, които са неизбежни, би трябвало с времето да стават незначителни.

Ето защо искаме числените методи да запазват асимптотичната устойчивост на решенията. Това свойство на числените методи се нарича абсолютна устойчивост или A-устойчивост.

За да видим как ще изследваме утойчивостта на числения метод, първо трябва да коментираме как аналитично изследваме устойчивостта на една равновесна точка.

Нека разгледаме уравнението

$$\frac{du}{dt} = f(u),$$

за което ξ е равновесна точка, т.е. $f(\xi) = 0$.

Развивайки дясната страна около $u = \xi$ в ред на Тейлър, получаваме

$$\frac{du}{dt} = f(\xi) + f'(\xi)(u - \xi) + O(|2|) = f'(\xi)(u - \xi).$$

Използвайки, че $f(\xi) = 0$ и че членовете от втори и по-висок ред могат да се пренебрегнат за достатъчно малки стойности на $u - \xi$, получаваме, че поведението на решението в достатъчно малка околност на точката ξ се определя от уравнението

$$\frac{du}{dt} = f'(\xi)(u - \xi).$$

Означавайки $\lambda := f'(\xi)$ и $\overline{u} := u - \xi$, записваме уравнението във вида

$$\frac{d\overline{u}}{dt} = \lambda\overline{u}, \tag{2.10}$$

където $\lambda < 0$, тъй като равновесната точка ξ е устойчива.

Дефиниция 4. *Казваме, че даден числен метод е абсолютно устойчив (A-устойчив), ако, приложен върху задачата (2.10), решението му удовлетворява*

$$|y_i| \leq |y_0|.$$

Забележка. С други думи, ако в даден момент от време е допусната грешка (например от закръгляване) y_0, искаме тя да не расте с времето и във всеки следващ момент y_i да остава не по-голяма по абсолютна стойност от y_0. Ние ще поискаме малко по-силното условие $|y_i| \to 0$.

Тогава, ако можем да представим метода във вида $y_{i+1} = Ry_i$, лесно се вижда, че условието е изпълнено, ако $|R| < 1$.

Действително, имаме

$$y_{i+1} = Ry_i = R^2 y_{i-1} = \cdots = R^{i+1} y_0,$$

което клони към 0 точно тогава, когато $|R| < 1$.

Забележка. В зависимост от решаваната задача можем да поискаме методът, който използваме, да запазва различни важни свойства на търсеното решение. A-устойчивостта е свързана със запазване на асимптотичната устойчивост.

Нека сега изследваме A-устойчивостта на методите на Ойлер. За явния метод на Ойлер имаме

$$y_{i+1} = y_i + hf(t_i, y_i) = y_i + h\lambda y_i = (1 + h\lambda)y_i.$$

Следователно, за да бъде явният метод на Ойлер A-устойчив, трябва да е изпълнено

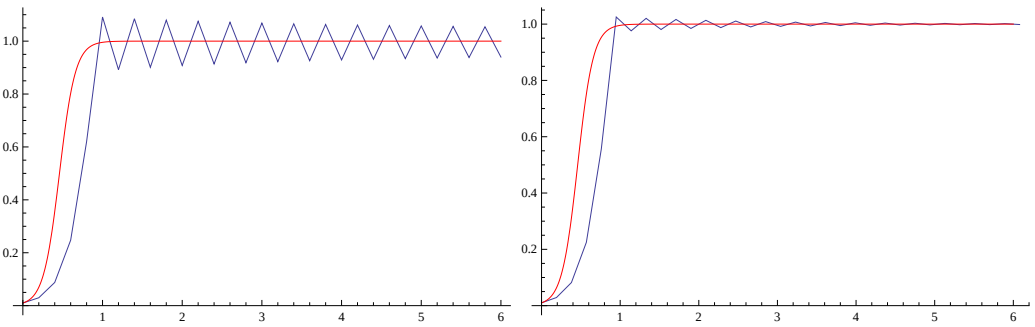
$$|1 + h\lambda| < 1 \Longrightarrow h < -2/\lambda$$

За неявния метод на Ойлер имаме

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}) = y_i + h\lambda y_{i+1} \Longrightarrow y_{i+1} = \frac{1}{1 - \lambda h} y_i.$$

Вземайки предвид, че $\lambda < 0$ е ясно, че $|\frac{1}{1-\lambda h}| < 1$ за всяко h, т.е. неявният метод на Ойлер е A-устойчив за всяко h.

Да илюстрираме казаното върху задачата (2.9). Единствената устойчива равновесна точка е $u = 1$ и $\lambda = f'(1) = -10$. Следователно методът е A-устойчив точно тогава, когато $h \leq 2/10$, като очакваме грешката да клони към 0, ако $h < 2/10$. Действително, нека разгледаме решенията при стъпка $h = 0.2$, $h = 0.19$:



В първия случай грешката осцилира около равновесната точка без да расте, но и без да намалява. Във втория случай грешката клони към 0.

Резултатът при $h = 0.19$ обаче ни показва, че A-устойчивостта на метода не е достатъчна, за да има решението добро поведение. Въпреки че грешката клони към 0, се появява нежелано осцилиране около решението. Следователно е добре да поискаме методът да няма такова поведение.

Дефиниция 5. *Казваме, че даден числен метод е монотонен, ако, приложен върху задачата (2.10), решението му не сменя знака си, т.е.*

$$sgn(y_i) = sgn(y_0).$$

Забележка. С други думи, искаме грешката да не осцилира около нулата.

Ако запишем метода във вида

$$y_{i+1} = Ry_i,$$

е ясно, че последното е изпълнено точно тогава, когато $R > 0$.

За явния метод на Ойлер имаме

$$y_{i+1} = (1 + h\lambda)y_i$$

и следователно методът е монотонен точно тогава, когато $h < -1/\lambda$.

Лесно се вижда, че неявният метод на Ойлер е монотонен за всяко h, вземайки предвид, че

$$y_{i+1} = \frac{1}{1 - \lambda h} y_i.$$

2.6 Методи на Рунге–Кута

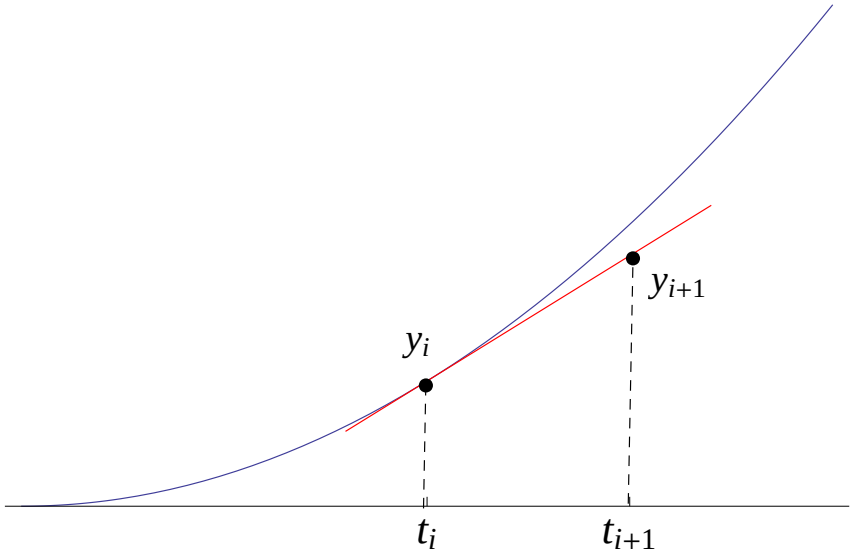
Основен недостатък на методите на Ойлер е, че те са бавно сходящи – да припомним, че те имат първи ред на сходимост, тъй като локалната грешка на апроксимация е $O(h)$. Това означава, че ако искаме да получим висока точност, трябва да работим с много малка стъпка, т.е. да правим голям брой операции, което, разбира се, е нецелесъобразно. Сега ще разгледаме първия от двата основни класа методи, които се използват, когато е необходим по-висок ред на сходимост.

2.6.1 Явни методи на Рунге–Кута

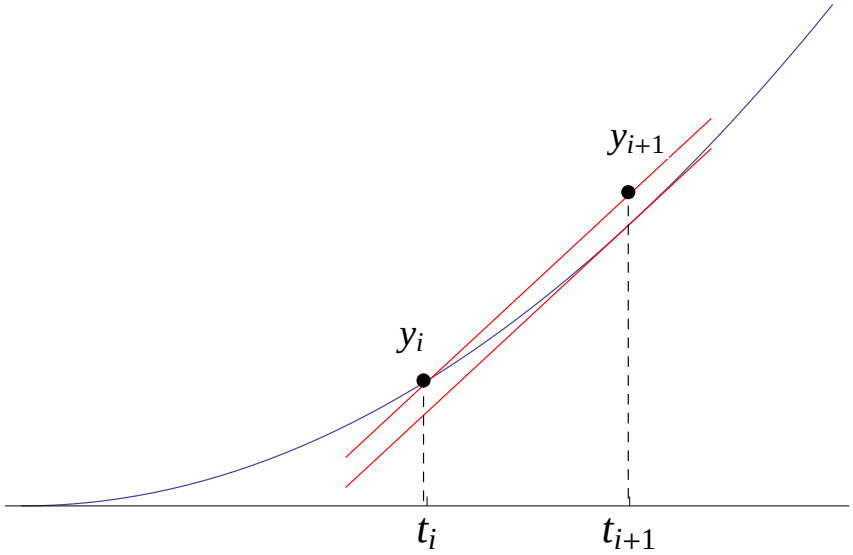
Явните методи на Рунге–Кута са най-често използваният клас методи за решаване на ОДУ. За да мотивираме тяхното използване, нека първо разгледаме от геометрична гледна точка методите на Ойлер. При явния метод на Ойлер имаме

$$y_{i+1} = y_i + hf(t_i, y_i),$$

т.е. правим една стъпка по направление на допирателната в точката t_i, за да получим приближението в точката t_{i+1}:



При неявния метод на Ойлер използваме направлението, зададено от допирателната в точката t_{i+1}:

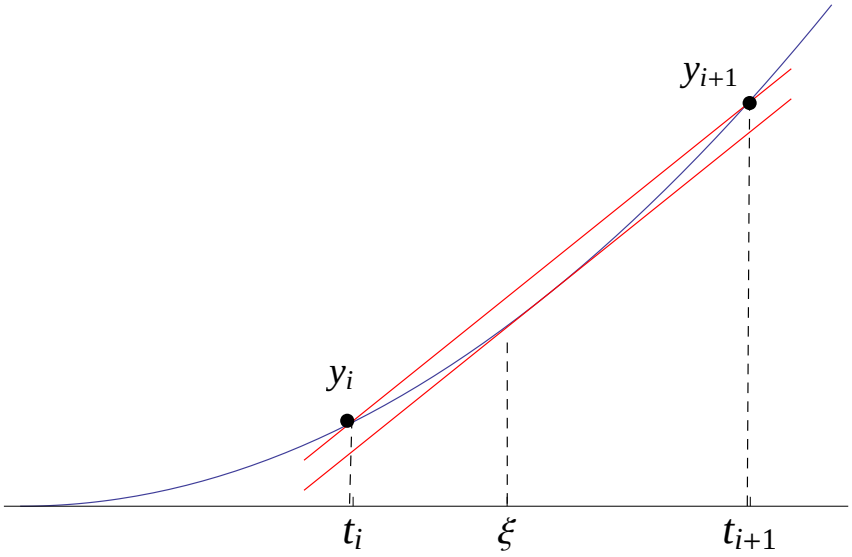


ЛГА за явния и за неявния метод на Ойлер е $O(h)$.

От друга страна, според Теоремата за крайните нараствания, съществува ξ, така че

$$u(t_i + h) - u(t_i) = u'(\xi)(t_{i+1} - t_i) = u'(\xi)h.$$

С други думи, ако използваме направлението, зададено от допирателната в тази точка, ЛГА би била 0:



Ние, разбира се, няма как да намерим точката ξ, но нейното съществуване ни дава основание да вземем производните в различни точки в интервала $[t_i, t_{i+1}]$ и да ги усредним с някакви тегла, така че да получим стойност, близка (в някакъв смисъл) до стойността в ξ. Нека означим стойностите на производните в тези точки (умножени по h) с $k_1, k_2, \ldots, k_s$. Общият вид на s-етапните методи на Рунге-Кута е

$$\frac{y_{i+1} - y_i}{h} = \frac{1}{h}(p_1 k_1 + p_2 k_2 + \cdots + p_s k_s),$$

където $p_1, p_2, \ldots, p_s$ са теглата, с които вземаме всяка от производните. Ще ги определим така, че ЛГА да е възможно най-малка.

k_1 е производната в точката t_i. За нея имаме

$$k_1 = hf(t_i, y_i).$$

k_2 е производната в някоя друга точка $t_i + \alpha_2 h$. За да я пресметнем обаче (т.е. да пресметнем дясната страна в диференциалното уравнение), ни е необходима стойността на решението в тази точка, което ни е неизвестно. Ще го апроксимираме на база на известната информация, като за неговата стойност вземем $y_i + \beta_{2,1} k_1$. Тогава

$$k_2 = hf(t_i + \alpha_2 h, y_i + \beta_{2,1} k_1).$$

Параметрите α_2, $\beta_{2,1}$ отново подлежат на определяне, така че ЛГА да е възможно най-малка.

Разсъждавайки аналогично и за производните в следващите точки, получаваме следния общ вид на методите на Рунге-Кута:

$$\begin{aligned}
&\frac{y_{i+1} - y_i}{h} = \frac{1}{h}(p_1 k_1 + p_2 k_2 + \cdots + p_s k_s),\\
&k_1 = hf(t_i, y_i)\\
&k_2 = hf(t_i + \alpha_2 h, y_i + \beta_{2,1} k_1)\\
&k_3 = hf(t_i + \alpha_3 h, y_i + \beta_{3,1} k_1 + \beta_{3,2} k_2)\\
&k_j = hf(t_i + \alpha_j h, y_i + \beta_{j,1} k_1 + \cdots + \beta_{j,j-1} k_{j-1}), \quad j = 4, \ldots, s.
\end{aligned}$$

Да обърнем внимание, че методите на Рунге-Кута са едностъпкови методи, тъй като при намирането на y_{i+1} използват само стойността на y_i, но не и на стойностите на приближеното решение в предходните точки.

В литературата могат да се открият коефициентите на различни методи от този тип с различен ред на сходимост. Често коефициентите се записват в т.нар. Таблица на Butcher:

$$\begin{array}{c|cccc} 0 & & & & \\ \alpha_2 & \beta_{2,1} & & & \\ \vdots & \vdots & \ddots & & \\ \alpha_s & \beta_{s,1} & \cdots & \beta_{s,s-1} & \\ \hline & p_1 & p_2 & \cdots & p_s \end{array}$$

Нека разгледаме един метод на Рунге-Кута, който има четвърти ред на апроксимация и е най-често използваният метод за решаване на ОДУ при равномерна мрежа. Таблицата на Butcher за метода е

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Записан подробно, методът има следния вид:

$$\begin{aligned} &\frac{y_{i+1}-y_i}{h} = \frac{1}{h}(p_1k_1 + p_2k_2 + p_3k_3 + p_4k_4), \\ &k_1 = hf(t_i, y_i), \\ &k_2 = hf(t_i + 1/2h, y_i + 1/2k_1), \\ &k_3 = hf(t_i + 1/2h, y_i + 1/2k_2), \\ &k_4 = hf(t_i + h, y_i + k_3). \end{aligned}$$

Прилагаме функция в Mathematica, реализираща метода на Рунге-Кута от четвърти ред за произволна задача на Коши (2.5).

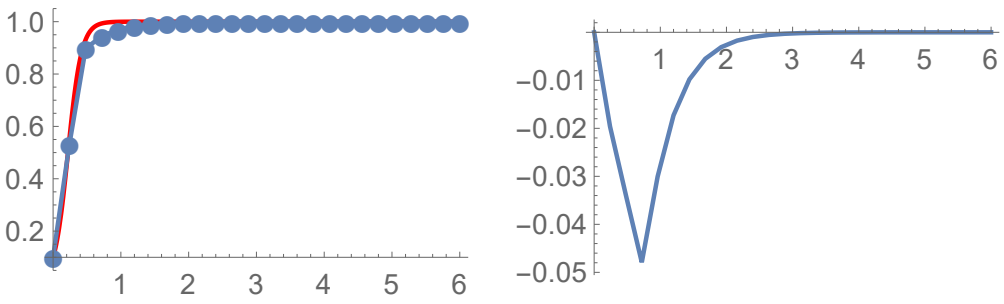
In[1]:=
```
RungeKuttaLogistic[f_, n_, t0_, T_] := (
  h = (T - t0) / n;
  x = Table[i * h, {i, 0, n}];
  y = Table[0, {i, 0, n}];
  y[[1]] = 0.1;
  For[i = 1, i <= n, i++,
   k1 = h * f[x[[i]], y[[i]]];
   k2 = h * f[x[[i]] + h / 2, y[[i]] + k1 / 2];
   k3 = h * f[x[[i]] + h / 2, y[[i]] + k2 / 2];
   k4 = h * f[x[[i]] + h, y[[i]] + k3];
   y[[i + 1]] = y[[i]] + 1 / 6 k1 + 1 / 3 k2 + 1 / 3 k3 + 1 / 6 k4];

  Transpose[{x, y}]
 )
```

Нека я използваме, за да решим задачата на Коши (2.9).

```
Clear[x];
exact[x_] = DSolve[{u'[x] == 10 u[x] (1 - u[x]), u[0] == 0.1}, u[x], x][
plotExact = Plot[exact[x], {x, 0, 6}, PlotStyle → Red, PlotRange → Al
n = 25;
f[x_, u_] := 10 u (1 - u);
appr = RungeKuttaLogistic[f, n, 0, 6];
plotAppr = ListLinePlot[appr, PlotMarkers → ●, PlotRange → All];
GraphicsRow[{Show[plotExact, plotAppr], ListLinePlot[Transpose[{ap
```

Графика с приближеното и точното решение и графика на грешката прилагаме по-долу.

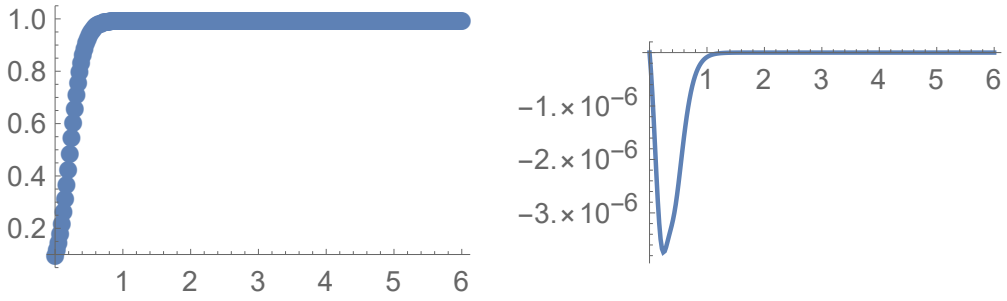
Out[47]=



Визуално, решението при $n = 25$ изглежда много добре (сравнете с решението при методите на Ойлер).

Ако намалим стъпката с един порядък, т.е. $n = 250$, тогава очакваме грешката да намалее с 4 порядъка, тъй като RK4 има четвърти ред на сходимост. Това действително е така, както можем да се убедим от следните графики:

Out[77]=



Повече информация за начина, по който се избират коефициентите в методите на Рунге–Кута може да бъде намерена в [3] и [1].

2.6.2 Метод на Рунге за практическа оценка на реда на сходимост.

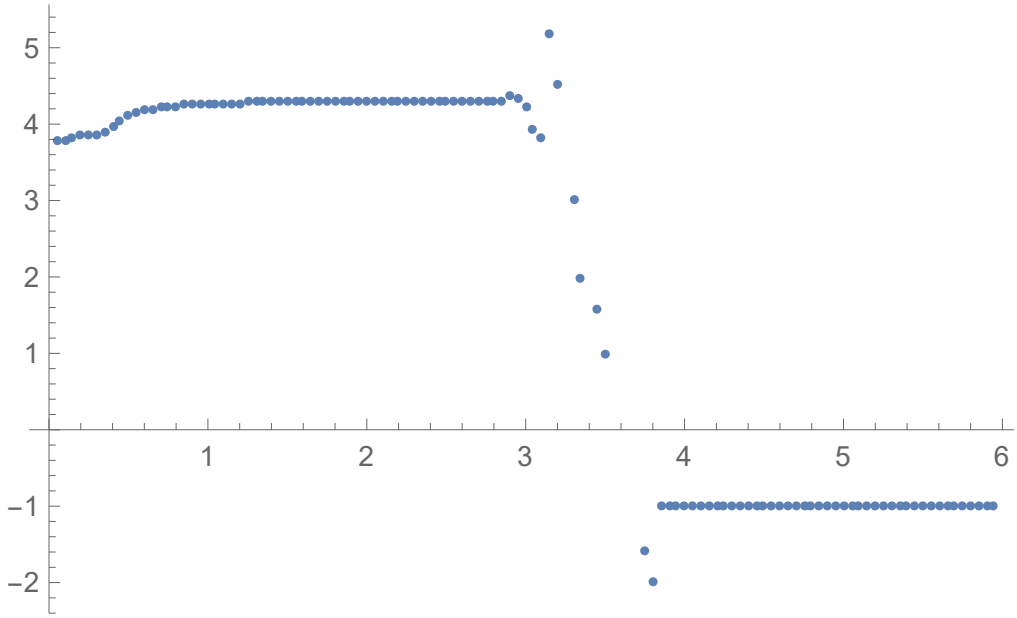
Много често при приближено решаване на дадена задача, бихме искали да оценим реда на сходимост на използвания числен метод, т.е. да намерим колко бързо приближеното решение се доближава до точното решение, като променяме стъпката.

За да пресметнем приближено реда на сходимост α на кой да е метод за решаване на дадена задача, бихме могли да използваме метода на Рунге за практическа оценка, т.е.:

$$\alpha(x) = \frac{1}{\ln 2} \ln \left| \frac{y_h(x) - y_{h/2}(x)}{y_{h/2}(x) - y_{h/4}(x)} \right|,$$

където $y_h(x)$, $y_{h/2}(x)$ и $y_{h/4}(x)$ са съответно приближените стойности на $u(x)$ в точката x върху мрежите ω_h, $\omega_{h/2}$ и $\omega_{h/4}$. Ако за достъчно много общи точки на трите мрежи е изпълнено, че $\alpha \approx \alpha(x)$, то можем да считаме, че реалният ред на сходимост за числения метод е α. Изследвайки редът на сходимост на задачата (2.5), получаваме следните резултати:

Out[69]=



Можем да забележим, че когато сме достатъчно далеч от равновесната точка, то $\alpha \approx 4$, а като се приближим до нея се получава $\alpha \approx -1$. Това означава, че редът на сходимост е 4, тъй като при $x \geq 3$ грешката между точното и приближеното решение е от порядъка на 10^{-16}, т.е. сме достигнали на практика до точното решение $u(x)$.

2.6.3 Изследване на A-устойчивост и монотонност за методите на Рунге–Кута

Задача 11. Да се изследва за A-устойчивост и монотонност методът на Рунге–Кута

$$\begin{aligned}
&\frac{y_{i+1} - y_i}{h} = \frac{1}{2h}(k_1 + k_2),\\
&k_1 = hf(t_i, y_i),\\
&k_2 = hf(t_i + h, y_i + k_1).
\end{aligned}$$

Решение. Изследваме A-устойчивост и монотонност върху моделната задача (2.10). Получаваме последователно

$$\begin{aligned}
&k_1 = h\lambda y_i,\\
&k_2 = h\lambda(y_i + k_1) = h\lambda(y_i + h\lambda y_i) = h\lambda y_i + (h\lambda)^2 y_i,\\
&y_{i+1} = y_i + \frac{1}{2}(h\lambda y_i + h\lambda y_i + (h\lambda)^2 y_i) = \left(1 + h\lambda + \frac{1}{2}(h\lambda)^2\right) y_i.
\end{aligned}$$

Следователно, за да бъде методът A-устойчив, трябва да е изпълнено условието

$$\left|1 + \lambda h + \frac{1}{2}(h\lambda)^2\right| < 1.$$

Решавайки последното уравнение по отношение на $z := h\lambda$, получаваме

$$-2 < z < 0 \Longrightarrow h < -2/\lambda.$$

Условието за монотонност

$$1 + z + \frac{1}{2}z^2 > 0$$

е винаги изпълнено, т.е. методът е монотонен за всяко $h > 0$. □

Задача 12. Да се изследва за A-устойчивост и монотонност методът на Рунге–Кута

$$\begin{aligned}
&\frac{y_{i+1} - y_i}{h} = \frac{1}{6h}(k_1 + 4k_2 + k_3),\\
&k_1 = hf(t_i, y_i),\\
&k_2 = hf(t_i + h/2, y_i + k_1/2),\\
&k_3 = hf(t_i + h, y_i - k_1 + 2k_2).
\end{aligned}$$

Решение. Изследваме А-устойчивост и монотонност върху моделната задача (2.10). Получаваме последователно

$$\begin{aligned}
k_1 &= h\lambda y_i,\\
k_2 &= h\lambda(y_i + k_1/2) = h\lambda(y_i + h\lambda y_i/2) = h\lambda y_i + \frac{(h\lambda)^2}{2} y_i,\\
k_3 &= h\lambda(y_i - k_1 + 2k_2) = h\lambda(y_i - h\lambda y_i + 2h\lambda y_i + (h\lambda)^2 y_i)\\
&= h\lambda y_i + (h\lambda)^2 y_i + (h\lambda)^3 y_i\\
y_{i+1} &= y_i + \frac{1}{6}\left[h\lambda y_i + 4\left(h\lambda y_i + \frac{(h\lambda)^2}{2} y_i\right) + h\lambda y_i + (h\lambda)^2 y_i + (h\lambda)^3 y_i\right],\\
&= \left(1 + h\lambda + \frac{1}{2}(h\lambda)^2 + \frac{1}{6}(h\lambda)^3\right) y_i.
\end{aligned}$$

Следователно, за да бъде методът А-устойчив, трябва да е изпълнено условието

$$\left|1 + h\lambda + \frac{1}{2}(h\lambda)^2 + \frac{1}{6}(h\lambda)^3\right| < 1.$$

Решавайки последното уравнение по отношение на $z := h\lambda$, получаваме

$$-2.51275 < z < 0 \Longrightarrow h < -2.51275/\lambda.$$

Условието за монотонност

$$1 + z + \frac{1}{2}z^2 > 0$$

е изпълнено за $z > -1.59607$, т.е. $h < -1.59607/\lambda$. □

Забележка. Може да се покаже, че за всички явни методи на Рунге–Кута условието за А-устойчивост има вида

$$|1 + z + O(z^2)| < 1.$$

Последното очевидно няма как да бъде изпълнено за всяко z, т.е. **явните методи на Рунге–Кута са условно А-устойчиви.**

2.6.4 За магистрите: Методи с адаптивен избор на стъпката.

Разгледаните дотук методи използват равномерна мрежа. Това обаче невинаги е целесъобразно. Нека разгледаме следната задача на Коши.

$$\begin{aligned}
&\frac{du}{dt} = -b(t)u + t, \quad t \in (0, 10],\\
&u(0) = 1,
\end{aligned} \tag{2.11}$$

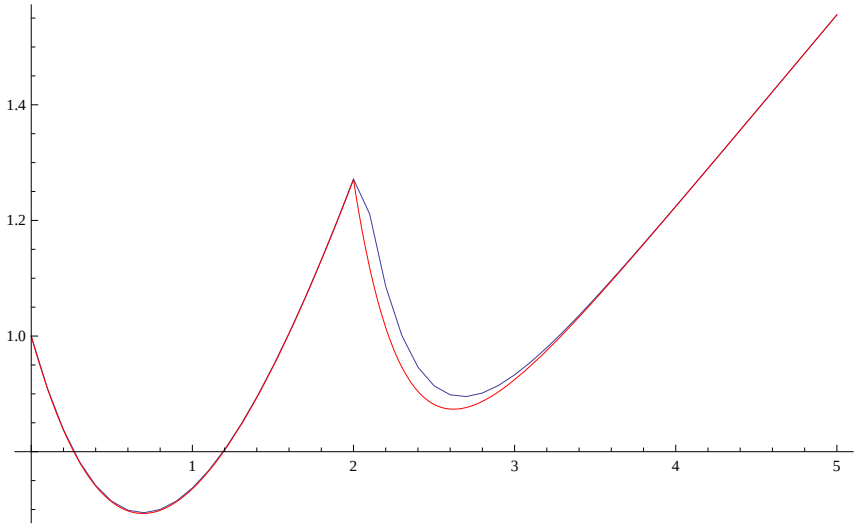
където

$$b(t) = \begin{cases} 1 & 0 \le t \le 2 \\ 3 & 2 < t \end{cases}.$$

имаща точно решение

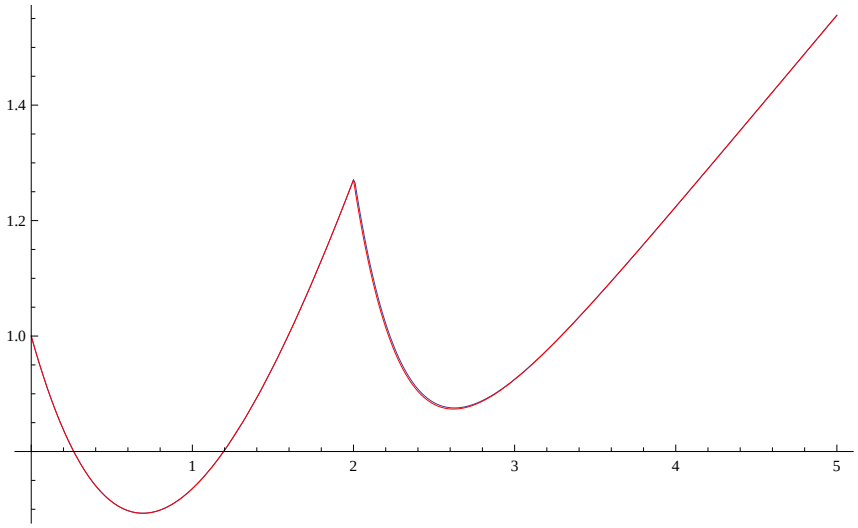
$$u(t) = \begin{cases} t - 1 + 2e^{-t} & 0 \leq t \leq 2 \\ \frac{1}{3}t - \frac{1}{9} + e^{-3t}\left(\frac{4}{9}e^6 + 2e^4\right) & t > 2 \end{cases}.$$

Решавайки задачата с метод на Рунге-Кута от трети ред със стъпка $h = 0.1$, получаваме следния резултат;



Както виждаме от графиката, методът дава добър резултат до особеността на решението в $t = 2$, след което обаче графиките на приближеното и точното решение сериозно се разминават.

Значително по-добър резултат получаваме със стъпка $h = 0.01$:



Възниква обаче логичният въпрос защо е необходимо да използваме толкова малка стъпка в целия интервал, след като в част от него 10 пъти по-голяма стъпка дава добри резултати.

При използването на равномерна мрежа стъпката трябва да се съобрази с най-лошия случай.

Идеята на методите с адаптивен избор на стъпката е да използваме малка стъпка там, където това е необходимо (обикновено, където решението се изменя бързо) и по-голяма стъпка, където поведението на решението го позволява.

Най-общо, идеята на методите с адаптивен избор на стъпката можем да изложим със следния алгоритъм.

Нека означим $h_i := t_{i+1} - t_i$, $i = 0, 1, 2, ...$

АЛГОРИТЪМ:

Избираме първоначална стойност за h_0; инициализираме $y_0 = u_0$, $t_0 = 0$.

Докато $t_i < T$:

1. Оценяваме локалната грешка, err_i, която бихме допуснали със стъпка h_i.
2. Докато $err_i > tol$, намаляваме стъпката h_i.
3. Пресмятаме y_{i+1}, t_{i+1}, както и първоначална стойност за h_{i+1}; увеличаваме i с 1.

Нека разгледаме последователно стъпки 1, 2 и 3.

1. Оценката на грешката зависи от конкретния метод, който използваме.

   Един възможен начин за практическа оценка на грешката е да решим задачата с два различни метода от различен ред, например втори и трети. Съответните приближени решения ще означим с $y_i^{<2>}$ и $y_i^{<3>}$. За оценка на грешката вземаме

   $$err_i := y_i^{<3>} - y_i^{<2>}.$$

   Неудобството при тази оценка е, че трябва да се пресмята приближеното решение по два различни начина, тоест да се правят допълнителни операции. Съществуват обаче двойки методи на Рунге-Кута, при които този проблем бива решен, тъй като методът от по-висок ред преизползва всички пресмятания на метода от по-нисък ред. Да разгледаме следната двойка методи на Рунге–Кута от втори и трети ред:

0			
1	1		
1/2	1/4	1/4	
	1/6	1/6	2/3
	1/2	1/2	0

   За метода от втори ред се използват коефициентите в първите два реда, а за метода от трети ред – в първите три.

   Коефициентите в първия ред под чертата се отнасят за метода от трети ред, а във втория ред – за метода от втори ред.

   Така получаваме

   $$y_{i+1}^{<3>} = y_i + 1/6k_1 + 1/6k_2 + 2/3k_3,$$
   $$y_{i+1}^{<2>} = y_i + 1/2k_1 + 1/2k_2.$$

   Следователно за оценката на грешката err_i имаме

   $$err_i = y_{i+1}^{<3>} - y_{i+1}^{<2>} = -\frac{1}{3}(k_1 + k_2 - 2k_3).$$

2. Сега ще изведем формула, по която ще променяме стъпката. Нека текущата стъпка е $h_{i,temp}$. Тъй като използваме метод от трети ред, имаме

$$err_i \approx Ch_{i,temp}^3 \tag{2.12}$$

за някоя константа C.

Искаме да изберем нова стойност на стъпката, $h_{i,new}$ така, че грешката да е приблизително равна на зададения толеранс tol, т.е.

$$tol \approx Ch_{i,new}^3 \Longrightarrow h_{i,new} = \left(\frac{tol}{C}\right)^{1/3}.$$

Изразявайки C от (2.12), получаваме окончателно

$$h_{i,new} = h_{i,temp}\left(\frac{tol}{err_i}\right)^{1/3}.$$

Да обърнем внимание, че ако $err_i < tol$, то горната формула води до увеличаването на стъпката. С други думи, ако сме използвали в текущата итерация по-малка стъпка, отколкото е било необходимо, за следващата итерация стъпката ще се увеличи. Затова същата формула използваме и за определянето на начална стойност за h_{i+1}.

Прилагаме примерна имплементация на метода в Mathematica:

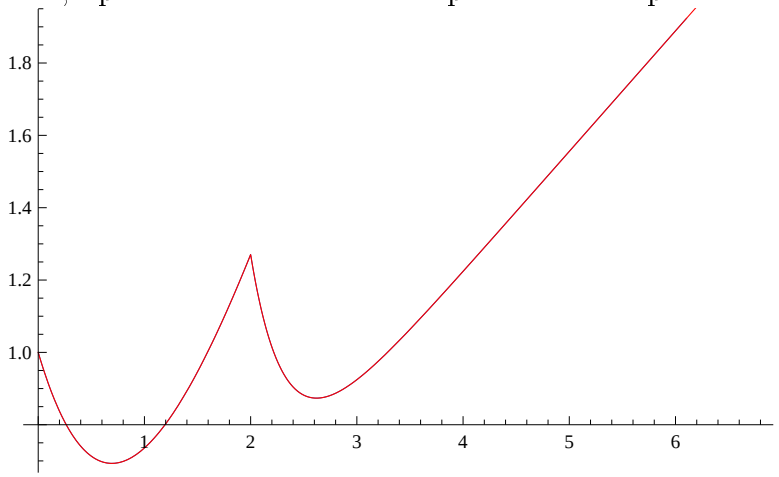
In[48]:=
```
RK23[f_, t0_, T_, h0_, tol_] := (
  h = h0;
  t = t0;
  y = {0};
  y[[1]] = 1;
  i = 1;
  t = {0};
  While[t[[i]] < T,
   k1 = h f[t[[i]], y[[i]]];
   k2 = h f[t[[i]] + h, y[[i]] + k1];
   k3 = h f[t[[i]] + h/2, y[[i]] + 1/4 k1 + 1/4 k2];
   While[Abs[1/3 (k1 + k2 - 2 k3)] > tol,
    h = h (tol/Abs[1/3 (k1 + k2 - 2 k3)])^(1/3);
    k1 = h f[t[[i]], y[[i]]];
    k2 = h f[t[[i]] + h, y[[i]] + k1];
    k3 = h f[t[[i]] + h/2, y[[i]] + 1/4 k1 + 1/4 k2];
   ];
   y = Append[y, y[[i]] + 1/6 k1 + 1/6 k2 + 2/3 k3];
   t = Append[t, t[[i]] + h];
   h = h (tol/(1/3 Abs[(k1 + k2 - 2 k3)]))^(1/3);
   i++
  ];
  {t, y}
 )
```

Да приложим реализираната функция за решаването на задачата (2.11).

In[50]:=
```
b[t_] := If[0 ≤ t ≤ 2, 1, 3]
f[t_, y_] := -b[t] y + t;
t = RK23[f, 0, 10, 0.1, 10^-5][[1]];
y = RK23[f, 0, 10, 0.1, 10^-5][[2]];
plotAppr = ListLinePlot[Table[{t[[i + 1]], y[[i + 1]]}, {i, 0, Length[y] - 1}]];
exactSolution[t_] := If[0 ≤ t ≤ 2, t - 1 + 2 E^-t, 1/3 t - 1/9 + E^-3 t (4/9 E^6 + 2 E^4)]
plotExact = Plot[exactSolution[t], {t, 0, 10}, PlotStyle → Red];
Show[plotAppr, plotExact]
```

Визуално, приближеното и точното решение са неразличими:

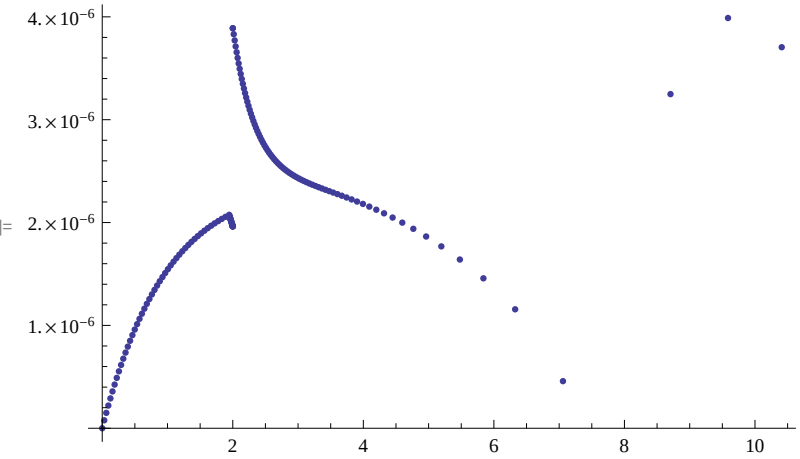
Out[57]=



Можем да видим, че грешката действително е ограничена от 10^{-5}:

In[107]:=
```
plotErr = ListPlot[Table[{t[[i + 1]], Abs[y[[i + 1]] - exactSolution[t[[i + 1]]]]},
    {i, 0, Length[y] - 1}], PlotRange → All]
```

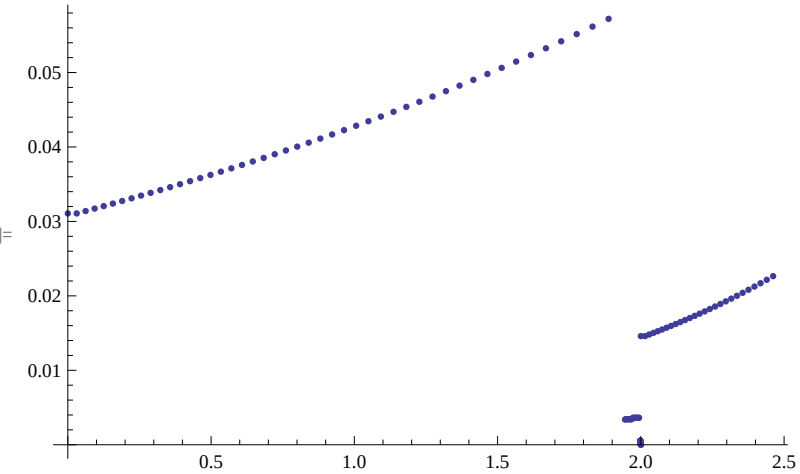
Out[107]=



Нека видим как се е изменяла стъпката. На следващата графика изобразяваме стъпката в интервала $0 \leq t \leq 2.5$.

In[115]:=
```
ListPlot[Table[{t[[i]], t[[i + 1]] - t[[i]]}, {i, 1, (Length[y] - 1) / 1.5}]
 , PlotRange → All]
```

Out[115]=

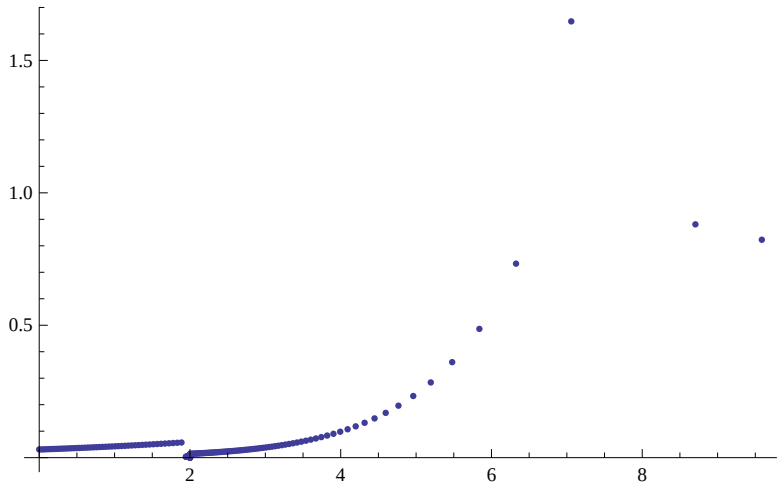


Виждаме, че около особеността на решението стъпката е намалена значително. След тази особеност методът увеличава стъпката, тъй като решението

става близо до линейна функция:

```
ListPlot[Table[{t[[i]], t[[i + 1]] - t[[i]]}, {i, 1, (Length[y] - 1)}],
 PlotRange → All]
```

Out[116]=



2.7 Многостъпкови методи. Методи на Адамс-Башфорт и Адамс-Мултон.

Вторият основен клас методи за решаване на ОДУ от първи ред, освен методите на Рунге–Кута, са т.нар. многостъпкови методи.

При едностъпковите методи, например методите на Рунге–Кута и Ойлер, пресмятаме стойността на приближеното решение y_{i+1}, използвайки само стойността y_i. Например по явния метод на Ойлер имаме

$$y_{i+1} = y_i + hf(t_i, y_i).$$

Идеята на многостъпковите методи е да използваме информацията, която сме получили за решението в няколко предходни точки, като по този начин целим да получим по-висока точност. Един възможен подход за получаване на такива методи е следният. Както казахме, диференциалното уравнение

$$\frac{du}{dt} = f(t, u(t))$$

може да се сведе до еквивалентно на него интегрално уравнение. За тази цел интегрираме двете страни в граници от t_i до t_{i+1}[1] и получаваме

$$u_{i+1} - u_i = \int_{t_i}^{t_{i+1}} f(t, u(t))dt$$

или, което е същото,

$$u_{i+1} = u_i + \int_{t_i}^{t_{i+1}} f(t, u(t))dt. \tag{2.13}$$

[1] По-общо, можем да интегрираме в произволен интервал $[t_{i-k}, t_i]$, но най-често използваните методи се получават при избор на интервала $[t_{i-1}, t_i]$

За да получим диференчно уравнение, апроксимиращо горното интегрално уравнение и следователно оригиналното диференциално уравнение, ще апроксимираме интеграла в дясната страна.

За тази цел ще използваме интерполационна квадратурна формула. Нека означим $\overline{f}(t) = f(t, u(t))$. Да припомним, че идеята на интерполационните квадратурни формули е да апроксимираме подинтегралната функция с интерполационния ѝ полином на Лагранж, $\overline{f} \approx L_n(\overline{f}, t)$ и тогава получаваме приближение на интеграла, като интегрираме L_n, вместо $\overline{f}$. Интерполационния полином на Лагранж построяваме, като за интерполационни възли вземем точките $t_i, t_{i-1}, t_{i-2}, \ldots, t_{i-s}$.

Забележка. Естествено е, разбира се, да използваме именно точките от мрежата, тъй като за построяване на интерполационния полином ни е необходимо да знаем стойността на интерполираната функция, а това е изпълнено именно в точките от мрежата.

Забележка. Използвайки възела t_i и предходните възли, получаваме **явен метод**, тъй като не използваме стойност на решението, която все още не е намерена.

След тези уточнения, нека разгледаме няколко конкретни метода, получени по този начин.

Едностъпков метод. При $s = 0$, единственият интерполационен възел е t_i. Търсим полинома от нулева степен, който интерполира функцията $\overline{f}(t)$ (т.е. съвпада с нея) в точката t_i. Очевидно това е константата $\overline{f}(t_i) = f(t_i, u(t_i)) =: f_i$. Имаме

$$\overline{f}(t) \approx f_i \Longrightarrow \int_{t_i}^{t_{i+1}} \overline{f}(t)dt \approx hf_i.$$

Заместваики апроксимацията на интеграла в (2.13), получаваме

$$y_{i+1} = y_i + hf_i,$$

т.е. отново получихме явния метод на Ойлер, за който знаем, че има ЛГА $O(h)$.

Двустъпков метод. При $s = 1$ търсим полинома от първа степен, който интерполира функцията $\overline{f}(t)$ в точките t_i, t_{i-1}. Ще използваме формулата на Нютон с разделени разлики. Пресмятаме необходимите ни разделени разлики:

	$f[\cdot]$	$f[\cdot,\cdot]$
t_i	f_i	$\frac{f_{i-1}-f_i}{-h}$
t_{i-1}	f_{i-1}	

Тогава за интерполационния полином получаваме

$$\begin{aligned}
&\overline{f}(t) \approx f_i + \frac{f_{i-1} - f_i}{-h}(t - t_i) \\
&\Longrightarrow \int_{t_i}^{t_{i+1}} \overline{f}(t)dt \approx \int_{t_i}^{t_{i+1}} \left(f_i + \frac{f_{i-1} - f_i}{-h}(t - t_i) \right) dt \\
&= f_i h - \frac{f_{i-1} - f_i}{h} \left. \frac{(t - t_i)^2}{2} \right|_{t_i}^{t_{i+1}} = f_i h - \frac{h}{2}(f_{i-1} + f_i).
\end{aligned}$$

Замествайки апроксимацията на интеграла в (2.13), получаваме метода

$$y_{i+1} = y_i + \frac{h}{2}(3f_i - f_{i-1}),$$

за който може да се покаже, че има ЛГА $O(h^2)$.

Получените по този начин методи се наричат **методи на Адамс-Башфорт** и са явни линейни многостъпкови методи за решаване на задачата (2.5).

Аналогично могат да се получат неявни методи, като при построяването на интерполационния полином, апроксимиращ подинтегралната функция в (2.13), използваме възли $t_{i+1}, t_i, \ldots, t_{i+1-s}$. Тези методи се наричат **методи на Адамс-Мултон**.

Формулите, както за методите на Адамс-Башфорт, така и за методите на Адамс-Мултон, могат да се намерят лесно в литературата.

Ще предложим функция, имплементираща метод на Адамс-Башфорт от четвърти ред:

$$y_{i+1} = y_i + \frac{h}{24}\left(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}\right)$$

Нека отбележим, че за пресмятането на приближеното решение в $i+1$-вата точка е необходимо решението в предходните 4 точки. Следователно стойностите в първите четири точки от мрежата не можем да намерим директно чрез горната формула. За тази цел се използва едностъпков метод, имащ същия ред на апроксимация. Ние ще използваме метода на Рунге–Кута от четвърти ред, който разгледахме по-рано.

```
In[1]:= AB4[f_, u0_, t0_, T_, n_] := (
          h = (T - t0) / n;
          t = Table[i h, {i, 0, n}];
          y = Table[0, {n + 1}];
          fi = Table[0, {n + 1}]; (*list of values of the right-hand side f*)
          y[[1]] = u0;
          (*Compute the first approximate values, using the RK4 method*)
          For[i = 1, i ≤ 3, i++,
           k1 = h f[t[[i]], y[[i]]];
           k2 = h f[t[[i]] + h / 2, y[[i]] + k1 / 2];
           k3 = h f[t[[i]] + h / 2, y[[i]] + k2 / 2];
           k4 = h f[t[[i]] + h, y[[i]] + k3];
           y[[i + 1]] = y[[i]] + 1/6 (k1 + 2 k2 + 2 k3 + k4);
           fi[[i + 1]] = f[t[[i + 1]], y[[i + 1]]];

          ];
          (*Compute the remaining values, using the AB4 formula*)
          For[i = 4, i < n + 1, i++,
           y[[i + 1]] =
            y[[i]] + h/24 (55 fi[[i]] - 59 fi[[i - 1]] + 37 fi[[i - 2]] - 9 fi[[i - 3]]);
           fi[[i + 1]] = f[t[[i + 1]], y[[i + 1]]];
          ];
          y
         )
```

Важно е да отбележим, че, **за да бъде имплементацията на метода ефективна, е необходимо да пазим не само стойностите на приближеното решение, но и на дясната страна в точките от мрежата**, за да избегнем многократното пресмятане на десните страни в едни и същи точки. **Би било сериозна грешка да не използваме този факт, тъй като в противен случай методът ще изпълнява много по-голям брой операции** (сравними с метода на Рунге–Кута).

Методите на Адамс–Башфорт са по-бързи от методите на Рунге–Кута със същия ред на точност, но обикновено методите на Рунге–Кута имат по-добро поведение от гледна точка на устойчивостта на метода. Последното е важно за задачи, при които условието за устойчивост може да означава, че методът на Адамс трябва да се използва с твърде малка стъпка, което да го направи неприложим на практика. **Изборът на метод за решаването на всяка конкретна задача зависи от това какво се изисква от него – бързодействие, висока точност, устойчивост и т.н.**

2.8 Някои приложения на числените методи за решаване на ОДУ

Ще разгледаме няколко примера, на базата на които ще илюстрираме приложението на разглежданите методи на практика.

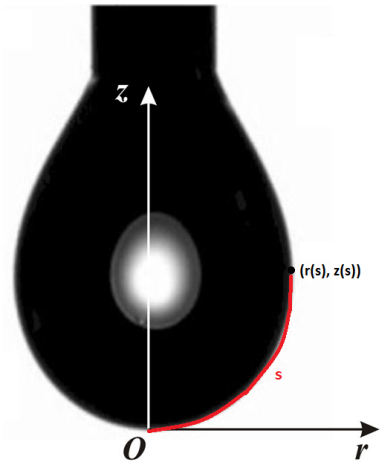
Анализ на формата на осево-симетрична капка.

Методът на Адамс–Башфорт датира от края на 19-ти век, като първо той е бил приложен за определяне профила на капка от даден флуид. Ето защо ние ще го илюстрираме именно върху тази задача.

Разглеждаме системата

$$\begin{aligned}\frac{dr}{ds} &= \cos\varphi,\\ \frac{dz}{ds} &= \sin\varphi,\\ \frac{d\varphi}{ds} &= 2b + cz - \frac{\sin\varphi}{r},\end{aligned}$$

която описва профила на капка, пусната от даден капиляр, под действието на гравитацията. Профилът се описва като параметрична крива $l = (r(s), z(s))$, която е параметризирана по дължината на дъгата, s, измерена от върха на капката където φ е ъгълът, който допирателната в съответната точка сключва с абсцисната ос. Параметърът b описва кривината във върха на капката, а c е параметър, който характеризира веществото.



Началните условия са $r(0) = z(0) = \varphi(0) = 0$. Да отбележим, че системата има особеност при $s = 0$, тъй като дясната страна на последното уравнение не е дефинирана. Може да се покаже обаче, че дясната граница съществува и

$$\left.\frac{d\varphi}{ds}\right|_{0+} = b.$$

Тогава е естествено да приемем последното като дефиниция на дясната страна за $s = 0$.

За да можем да използваме вече написаната в Mathematica функция $AB4$, първо трябва да запишем системата като едно векторно уравнение. Полагайки

$\mathbf{u} = (r, z, \varphi)^T$ и $\mathbf{f}(\mathbf{u}) = (f_1, f_2, f_3)^T$, където

$$\begin{aligned} f_1(\mathbf{u}) &= \cos\varphi, \\ f_2(\mathbf{u}) &= \sin\varphi, \\ f_3(\mathbf{u}) &= \begin{cases} 2b + cz - \frac{\sin\varphi}{r}, & s > 0 \\ b, & s = 0 \end{cases}. \end{aligned}$$

получаваме системата

$$\begin{aligned} &\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}), \quad s > 0, \\ &\mathbf{u}(0) = (0, 0, 0)^T. \end{aligned}$$

Както казахме, ще използваме функцията $AB4$ за нейното решаване, като малко ще я модифицираме. Това се налага, предвид факта, че обикновено задачата е скалирана така (т.е. са избрани такива мерни единици), че радиусът на капиляра да бъде единица. Това означава, че трябва да интегрираме, докато r стане 1.

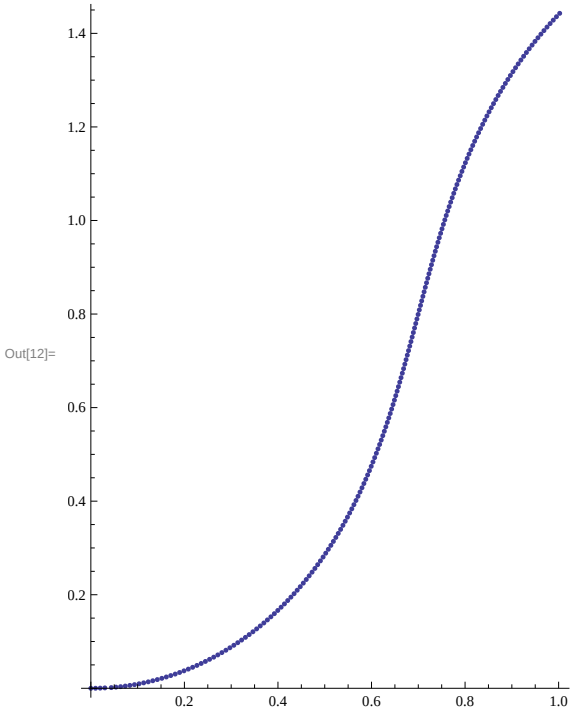
Вземайки предвид, че капката е осево-симетрична, то е достатъчно да вземем само половината от нея.

Прилагаме по-долу модифицирания код, заедно с приложението му за определяне на профила на капката при стойности на параметрите $b = 1.84366$ и $c = -2.9$.

In[7]:=
```
AB4[f_, u0_, h_] := (
  y = Table[0, {4}];
  fi = Table[0, {4}]; (*list of values of the right-hand side f*)
  y[[1]] = u0;
  (*Compute the first approximate values, using the RK4 method*)
  For[i = 1, i ≤ 3, i++,
   k1 = h f[y[[i]]];
   k2 = h f[y[[i]] + k1 / 2];
   k3 = h f[y[[i]] + k2 / 2];
   k4 = h f[y[[i]] + k3];
   y[[i + 1]] = y[[i]] + 1/6 (k1 + 2 k2 + 2 k3 + k4);
   fi[[i + 1]] = f[y[[i + 1]]];

  ];
  i = 4;
  (*Compute the remaining values, using the AB4 formula*)
  While[y[[i, 1]] ≤ 1,
   y = Append[y,
     y[[i]] + h/24 (55 fi[[i]] - 59 fi[[i - 1]] + 37 fi[[i - 2]] - 9 fi[[i - 3]])];
   fi = Append[fi, f[y[[i + 1]]]];
   i++;
  ];
  y
 )
```

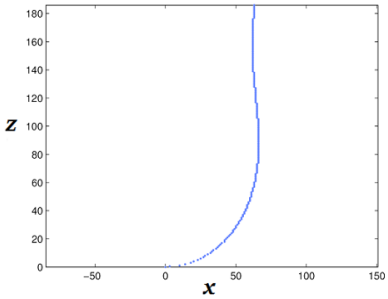
In[8]:=
```
b = 1 / 0.5424
c = -2.9;
f[u_] := {
   Cos[u[[3]]],
   Sin[u[[3]]],
   If[u[[1]] ≠ 0, 2 b + c u[[2]] - Sin[u[[3]]] / u[[1]], b]
  }
res = AB4[f, {0, 0, 0}, 0.01];
ListPlot[Table[{res[[i, 1]], res[[i, 2]]}, {i, 1, Length[res]}],
 AspectRatio → Automatic]
```

В резултат от изпълнението на горния код, получаваме следния профил на капка.



Решаването на горната система ОДУ е част от т.нар. Axisymmetric Drop Shape Analysis. Това е метод, на базата на който работят много уреди за определяне на повърхностно напрежение на течности (тенсиометри).

Повърхностното напрежение е характеристика на течностите, която ги кара да възприемат форма, при която повърхността им е минимална. Направата на измервателни уреди за измерване на повърхностно напрежение е голяма индустрия. За тази цел капка от дадено вещество се пуска от капиляр. На капката се прави снимка, която се дигитализира и така профилът се описва с множество от точки (заради симетрията разглеждаме само половината).



От друга страна, отчитайки силите, действащи върху капката, както видяхме, може да се определи профилът при зададени параметри. Софтуерът, управляващ уредите, определя стойността на параметрите (в частност параметърът c, в който участва повърхностното напрежение) така, че експерименталният и теоретичният профил да съвпадат.

Решаване на твърди системи.

Нека разгледаме задачата на Коши

$$\frac{du}{dt} = \lambda(-u + \sin t), \quad 0 < t \leq 10,$$
$$u(0) = 0,$$

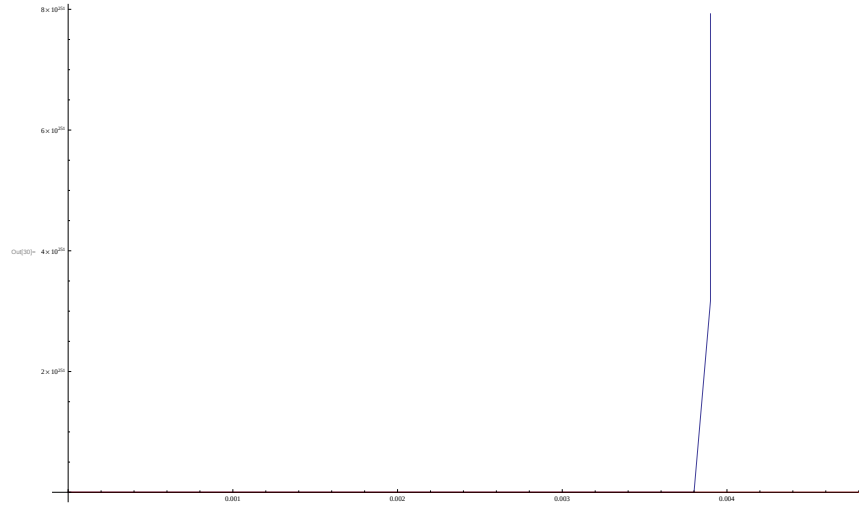
имаща точно решение

$$u(t) = \frac{\lambda}{1+\lambda^2} e^{-\lambda t} + \frac{\lambda^2}{1+\lambda^2} \sin t - \frac{\lambda}{1+\lambda^2} \cos t.$$

Да решим задачата при $\lambda = 1000000$. Първата ни идея може да бъде да използваме явен метод, поради относителната простота и бързодействие спрямо неявните. Нека приложим например метода на Рунге-Кута от четвърти ред.

```
In[24]:= λ = 1 000 000;
         f[t_, y_] := λ (-y + Sin[t]);
         y = RK4[f, 0, 0.0001, 0, 10];
         plotAppr = ListLinePlot[Table[{t[[i + 1]], y[[i + 1]]}, {i, 0, n}]];
         c = λ / (1 + λ^2);
         plotExact = Plot[c E^(-λ t) + λ^2 / (1 + λ^2) Sin[t] - λ / (1 + λ^2) Cos[t], {t, 0, 10}, PlotStyle → Red];
         Show[plotAppr, plotExact]
```

Да отбележим, че сме използвали доста малка стъпка – 10^{-4}. Резултатът от изпълнението на горния код е следният:

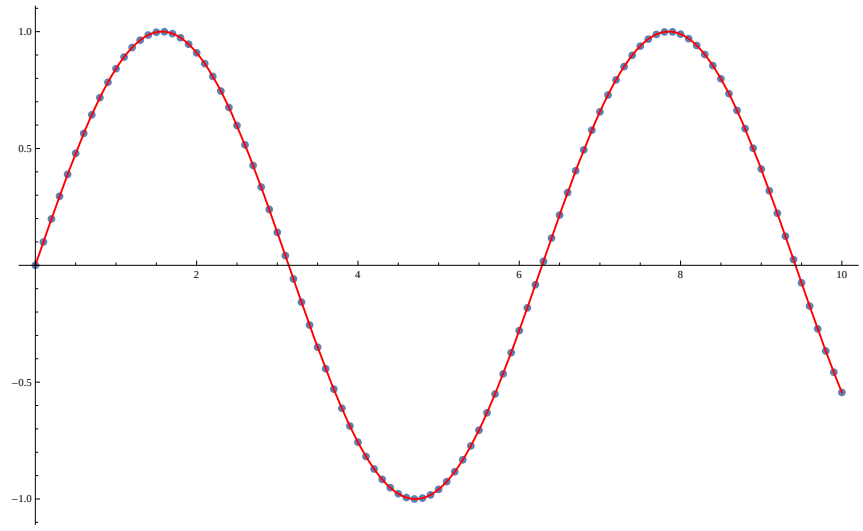


Ясно е, че причината за получения резултат е условната устойчивост на метода. За конкретната задача условието за устойчивост означава, че стъпката трябва да е твърде малка, за да бъде методът устойчив, което го прави на практика неприложим.

Това е пример за т.нар. **твърди задачи**. Естествен подход тогава е да опитаме да решим задачата с неявен метод. Ще изпробваме неявния метод на Ойлер за целта.

```
λ = 1 000 000;
f[t_, y_] := λ (-y + Sin[t]);
y = implicitEuler[f, 0, 0.1, 0, 10];
plotAppr = ListPlot[Table[{t[[i + 1]], y[[i + 1]]}, {i, 0, n}],
    PlotRange → All];
c = λ / (1 + λ^2);
plotExact = Plot[c E^(-λ t) + λ^2 / (1 + λ^2) Sin[t] - λ / (1 + λ^2) Cos[t], {t, 0, 10}, PlotStyle → Red];
Show[plotAppr, plotExact, PlotRange → All]
```

Резултатът (при използвана стъпка $h = 0.1$) е следният:



Виждаме, че дори неявният метод на Ойлер, който има първи ред на сходимост, се справя добре с решаването на задачата.

Анимация на свободно падащо тяло.

Ще моделираме поведението на топче, което пада свободно. На топчето му действат силата на тежестта $F = mg$, където m е масата на тялото, а g – земното ускорение. Ако с x означим отместването на топчето, то от втория закон на Нютон е известно, че:

$$\begin{aligned} ma &= F, \\ mx'' &= mg, \\ x'' &= g. \end{aligned}$$

Така получихме следното диференциално уравнение от втори ред:

$$x'' = g.$$

За да анимираме поведението топчето в даден времеви интервал (например $0 < t \leq 10$), са ни необходими 2 начални условия. Нека топчето в момент 0 се намира в позиция 0 и началната му скорост е 0. Така получаваме задачата:

$$\begin{aligned} x'' &= g,\ 0 < t \leq 10, \\ x(0) &= 0, \\ x'(0) &= 0. \end{aligned}$$

Ще сведем задачата от втори ред към задача на Коши за системата ОДУ от първи ред. За целта ще положим $x' = v$, което води до следната еквивалетна задача:

$$\begin{aligned} x' &= v,\ 0 < t \leq 10, \\ v' &= g,\ 0 < t \leq 10, \\ x(0) &= 0, \\ v(0) &= 0. \end{aligned} \tag{2.14}$$

Ако означим $\mathbf{u} := (x, v)^T$, то тогава задачата може да се запише и като

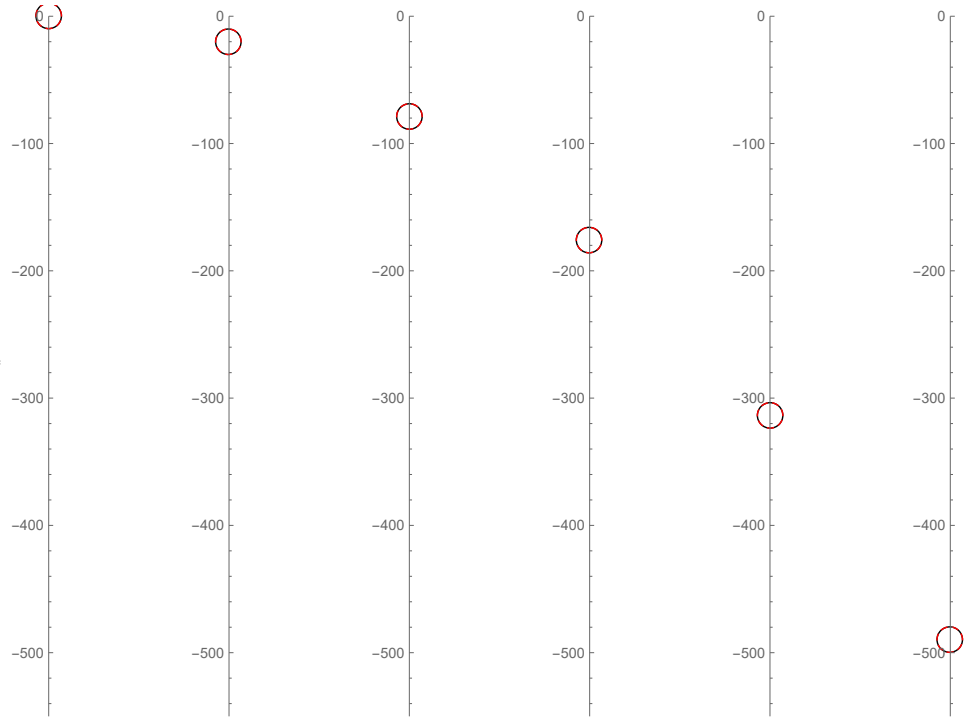
$$\begin{aligned} \frac{d\mathbf{u}}{dt} &= \begin{pmatrix} v \\ g \end{pmatrix},\ 0 < t \leq 10, \\ \mathbf{u}(0) &= 0. \end{aligned}$$

Примерна имплементация на решението е представена по-долу:

```
explicitEuler[f_, h_, t0_, T_, u0_] := (
  n = Ceiling[(T - t0) / h];
  t = Table[t0 + i * h, {i, 0, n}];
  y = Table[0, {n + 1}];

  y[[1]] = u0;
  For[i = 1, i < n + 1, i++,
   y[[i + 1]] = y[[i]] + h * f[t[[i]], y[[i]]]
  ];

  Transpose[{t, y}]
 )

f[x_, u_] := {u[[2]], 9.8}
h = 0.00001;
apprPos2 = explicitEuler[f, h, 0, 10, {0, 0}][[All, 2, 1]];
Animate[Graphics[{Circle[{0, -exact1[t[[i]]]}, 5], {Red, Dashed, Circle[{0, -apprPos2[[i]]}, 5]}},
                                 PlotRange → {{0, 10}, {-550, 0}}, Axes → {False, True}],
         {i, 1, Length[t], 1}]
```

Резултати от експериментите в 5 различни момента от време са показани по-долу. Като червено топче е анимирано, използвайки точното решение на (2.14), а синьото – използвайки приближеното решение, намерено по явния метод на Ойлер.

Out[71]=



2.8.1 Сравнение на методите за числено решаване на ОДУ

В настоящия параграф ще направим едно кратко сравнение на база на няколко признака между методите за числено решаване на ОДУ, без да претендираме за каквато и да е изчерпателност.

1. Явни методи
   (а) Метод на Ойлер
   + Бързи и лесни за имплементиране
   - Бавно сходящи.
   (б) Методи на Рунге-Кута
   + Съществуват методи с висок ред на сходимост
   + По-добри свойства от гледна точка на устойчивост спрямо методите на Адамс-Башфорт
   + Удобни са в случаи, когато искаме да изпълним дадена точност и за използване с адативен избор на стъпката
   - По-бавни от методите на Адамс-Башфорт
   (в) Методи на Адамс-Башфорт
   + Съществуват методи с висок ред на сходимост
   + По-бързи спрямо методите на Рунге-Кута
   + Често се комбинират с неявните методи на Адамс-Мултон като предикторно-коректорен метод
   - По-лоши свойства от гледна точка на устойчивост спрямо методите на Рунге-Кута
2. Явни срещу неявни методи
   - Явните методи са значително по-бързи.
   - За твърди задачи явните методи са практически неприложими.
   - Често неявните методи се използват заедно с явен метод в двойка предиктор-коректор.

На база на горното сравнение още веднъж ще подчертаем, че познаването на различни методи ни позволява да използваме метод, който е подходящ, от гледна точка на изискванията на конкретната приложна задача, която решаваме.

2.9 Допълнителни задачи

Задача 13. Диференциалното уравнение

$$\frac{dy}{dx} = \frac{-x + \sqrt{x^2 + 4y^2}}{2y}$$

описва формата на равнинна крива, която отразява всички лъчи, успоредни на абсцисата, в една и съща точка (началото на координатната система). Такива модели намират приложение при огледалата в телескопите, сателитните

антени, соларните колектори и др. Решете задачата на Коши, съответстваща на горното уравнение

Задача 14. Разглеждаме числения метод с тегло θ

$$y_{j+1} = y_j + h\theta f_{j+1} + h(1-\theta)f_j.$$

- Кои методи се получават при $\theta = 0$ и $\theta = 1$?
- Как даденият метод може да бъде изведен от явния и неявния метод на Ойлер?
- За кои стойности на θ методът е неявен?
- Изведете условие за A-устойчивост за произволно θ.
- За кои стойности на θ методът е A-устойчив (за всяко $h > 0$)?
- За кои стойности на θ методът е монотонен (за всяко $h > 0$)?
- Покажете, че за всяко θ ЛГА е $O(h)$.
- Съществува ли стойност на θ, за която ЛГА е $O(h^2)$?

Задача 15. Разглеждаме уравнението

$$u'(t) = -300t^2y^3, \quad 0 \leq t \leq 3$$

с начално условие $u(0) = 1$.

- Покажете, че точното решение на горната задача на Коши е

$$u(t) = \frac{1}{\sqrt{200t^3 + 1}}.$$

- Приложете метода с адаптивен избор на стъпката RK23, разглеждан на упражнения с толеранс 0.0001.
- Намерете допуснатата грешка във всяка точка и сравнете с толеранса.
- Покажете как се е изменяла стъпката при решаването на задачата. Къде стъпката е малка? Защо?
- Каква равномерна стъпка очаквате да е необходима за метод на Рунге-Кута от трети ред, за да бъде постигната същата точност?
- Приложете метод на Рунге-Кута от трети ред с равномерна мрежа, така че да получите същата точност. Сравнете времената за изпълнение на програмата при равномерна и адаптивна стъпка, като използвате вградената функция TimeUsed
- Направете същото, като в предишната подточка, за явния и неявния метод на Ойлер и за методи на Рунге-Кута от четвърти и пети ред (за съответните таблици на Butcher вижте книгата на Butcher, качена в Moodle).
- Потърсете информация в интернет за метода на Runge-Kutta-Fehlberg (RKF45), който е най-често използваният на практика метод с адаптивен избор на стъпката. Решете задачата, като го използвате. Сравнете времето за изпълнение на програмат с това при RK23.

Глава 3

Диференчни методи за ЧДУ

3.1 Явни диференчни схеми за нестационарни задачи

Ще започнем изложението, като представим идеята за построяване на устойчиви диференчни схеми за нестационарни ЧДУ. За илюстрация ще използваме двата основни класа линейни нестационарни ЧДУ от втори ред – параболични и хиперболични задачи. По-конкретно, ще разгледаме уравненията на топлопроводността, преноса и струната.

3.1.1 Явна диференчна схема за едномерното линейно уравнение на топлопроводността

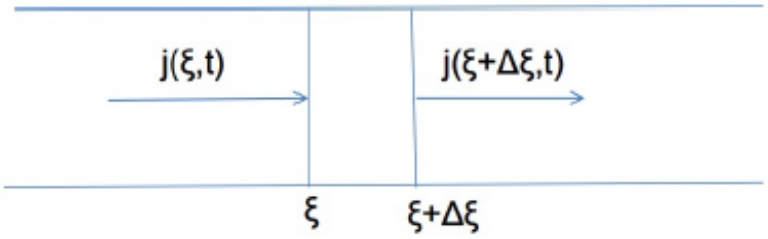
Дифузията е единият основен процес, който обуславя движението в природата. При него веществата се разпространяват от места с по-голяма концентрация (температура, енергия,...) към места с по-малка концентрация (температура, енергия,...). Ще изведем уравнението на дифузията (топлопроводността) в едномерния случай (т.е. когато процесът зависи само от една пространствена променлива). Математическият модел се базира на закон за запазване (на концентрация, на топлина и др.) **Голяма част от законите във физиката са именно отражение на някакъв закон за запазване.**

Ще изведем уравнението, разсъждавайки за концентрацията на дадено вещество, но с аналогични разсъждения можем да изведем същото уравнение, интерпретирано от гледна точка разпространението на топлина и др.

Разглеждаме дълга и тънка тръба, в която приемаме, че разпределението на вещество в едно напречно сечение е хомогенно, т.е. концентрацията на веществото (нека я бележим с u) зависи само от една пространствена променлива x, т.е. $u = u(x,t)$.

Нека с $j(x,t)$ означим потока на веществото в точката x вследствие на дифузията, т.е. това е скоростта, с която веществото преминава през тази точка. Нека приемем, че тръбата е добре уплътнена, т.е. през околната ѝ повърхнина не преминава вещество.

Нека разгледаме един достатъчно малък интервал $[\xi, \xi + \Delta\xi]$:



Изменението на концентрацията в дадения интервал е равно на разликата на входящия и изходящия поток, т.е.

$$\text{изменението на концентрацията} = j(\xi, t) - j(\xi + \Delta\xi, t).$$

Изменението за единица време в дадена точка от интервала е $\frac{\partial u}{\partial t}$. За да получим общото изменение в интервала, трябва да сумираме измененията във всяка точка и получаваме

$$\int_{\xi}^{\xi+\Delta\xi} \frac{\partial u}{\partial t} dx = j(\xi, t) - j(\xi + \Delta\xi, t).$$

Това е **интегралната форма на закона за запазване.** Обикновено, тя не се използва в този вид, а се свежда до диференциална. Един възможен подход е следният.

Пресмятаме интеграла в лявата страна по формулата на централните правоъгълници и получаваме

$$\frac{\partial u}{\partial t}\left(\xi + \frac{\Delta\xi}{2}, t\right)\Delta\xi + O(\Delta\xi^3) = j(\xi, t) - j(\xi + \Delta\xi, t).$$

Делим двете страни на $\Delta\xi$ и получаваме

$$\frac{\partial u}{\partial t}\left(\xi + \frac{\Delta\xi}{2}, t\right) + O(\Delta\xi^3) = \frac{j(\xi, t) - j(\xi + \Delta\xi, t)}{\Delta\xi}.$$

И сега, пускайки $\Delta\xi$ да клони към 0, получаваме **диференциалната форма на закона за запазване:**

$$\frac{\partial u}{\partial t} = -\frac{\partial j}{\partial x}(\xi, t).$$

В последното уравнение все още присъстват две неизвестни функции – $u(x,t)$ и $j(x,t)$. Оказва се обаче, че съществува връзка между тях. Тя е експериментално установена и носи името Закон на Фик (в топлопроводността – Закон на Фурие). Според него потокът е пропорционален на разликата в концентрациите (по-точно на градиента), т.е.

$$j(x, t) = -D\frac{\partial u}{\partial x}.$$

Заместващи последното в диференциалната форма на Закона за запазване, получаваме окончателно уравнението на дифузията (още, уравнение на топлопроводността)

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}.$$

Още веднъж ще подчертаем, че това уравнение описва много различни процеси – разпространение на топлина, дифузия на вещество, разпространение на заразни заболявания и др. Това е едно от големите достойнства на математическите модели – те са абстрактни задачи, които могат да се интерпретират по много различни начини – един и същ математически обект може да описва различни реални процеси.

И така, ще изложим основните идеи на диференчните методи за решаване на ЧДУ върху хомогенното линейно едномерно уравнение на топлопроводността

$$\frac{\partial u}{\partial t} - D\frac{\partial^2 u}{\partial x^2} = 0. \tag{3.1}$$

Имайки предвид, че в уравнението присъстват производна от втори ред по пространството и от първи ред, е ясно, че за да затворим диференциалната задача, е необходимо да наложим две (гранични) условия по пространството и едно (начално) по времето:

$$u(x,0) = u_0(x), \tag{3.2}$$
$$u(0,t) = u_L(t), \quad u(l,t) = u_R(t). \tag{3.3}$$

Наложените гранични условия се наричат **гранични условия от първи род (гранични условия на Дирихле).** По-нататък ще разгледаме и по-общи гранични условия.

За самото построяване на диференчната схема подхождаме по подобен начин на това, което направихме при методите за ОДУ от първи ред. Ще дискретизираме диференциалната задача, като въведем мрежа от възли, в които ще търсим стойностите на приближеното решение, след което ще апроксимираме производните чрез известните ни формули за числено диференциране.

Областта, в която разглеждаме задачата е правоъгълникът

$$\Re := \{(x,t) : 0 \le x \le l,\ 0 \le t \le T\}.$$

Простата геометрия на дефиниционната област на $u(x,t)$ (която е напълно естествена за разглежданата едномерна по пространството нестационарна задача) ни позволява да въведем **равномерна мрежа**[1] със стъпки по пространството и по времето, съответно h и τ, по следния начин:

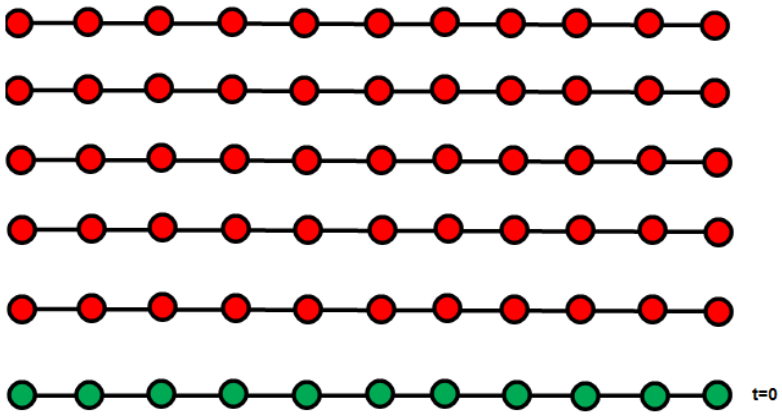
$$\overline{\omega}_{h,\tau} := \{(x_i,t_j) : x_i = ih, t_j = j\tau, i = \overline{0,n}, j = \overline{0,m}, n = l/h, m = T/\tau\}.$$

Стойността на приближеното решение в точката (x_i,t_j) ще бележим с y_i^j. Множеството от възли, за които $t = t_j$ ще наричаме j-ти **слой по времето**.

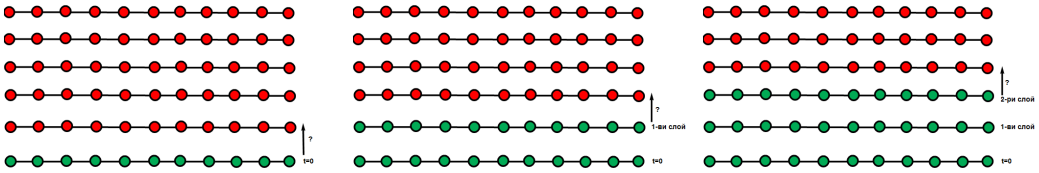
От началното условие ни е известна стойността на решението на 0-вия слой по времето

$$y_i^0 = u_0(x_i), \quad i = \overline{0,n}. \tag{3.4}$$

[1] Да отбележим, че е възможно използването на неравномерни и адаптивни мрежи, които да бъдат подходящи (в някакъв смисъл) за конкретната решавана задача, но това излиза съществено извън рамките на настоящия уводен курс.



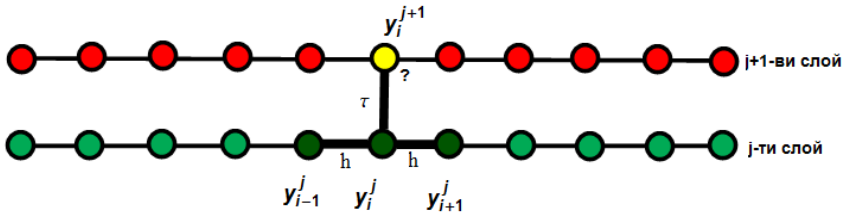
Тогава, ако на базата на информацията за 0-вия слой, можем да пресметнем стойностите на приближеното решение на 1-вия, оттам на 2-рия и т.н., или изобщо, ако, знаейки стойностите на приближеното решение на j-тия слой, можем да намерим тези на $j+1$-вия ($j = 0, \ldots, m-1$), ще можем да намерим всички търсени стойности.



И така, нека приемем, че са ни известни стойностите на j-тия слой по времето. Ще търсим тези на $j+1$-вия. Връзката между стойностите на приближеното решение, разбира се, ще получим като апроксимираме диференциалното уравнение (3.1) в точката (x_i, t_j). Апроксимирайки производната по времето с формулата с разлика напред и производната по пространството с формула с централна разлика от втори ред, получаваме диференчното уравнение с ЛГА $O(h^2 + \tau)$

$$\frac{y_i^{j+1} - y_i^j}{\tau} - D\frac{y_{i-1}^j - 2y_i^j + y_{i+1}^j}{h^2} = 0, \quad i = \overline{1, n-1}, \ j = \overline{0, m-1}. \tag{3.5}$$

В него участват стойностите на приближеното решение в точките с индекси $(i-1, j)$, (i, j), $(i+1, j)$ и $(i, j+1)$, като само последната стойност е неизвестна. Това означава, че **получената схема е явна**, тъй като от горното уравнение можем директно да намерим стойността y_i^{j+1}.



Тези четири точки образуват т.нар. **шаблон** за точката (x_i, t_j), в която апроскимираме оригиналната диференциална задача.

Шаблонът можем да поставим така, че да намерим всички вътрешни точки на $j+1$-вия слой по времето.

За граничните точки ще използваме граничните условия:

$$y_0^{j+1} = u_L(t_{j+1}), \quad y_n^{j+1} = u_R(t_{j+1}), \quad j = \overline{0, m-1}. \tag{3.6}$$

И така, уравненията (3.4), (3.5), (3.6) апроксимират оригиналната диференциална задача с диференчна такава. Решавайки уравнение (3.5) по отношение на единственото неизвестно y_i^{j+1}, получаваме т.нар. **каноничен вид на диференчната схема**:

$$\begin{aligned} &y_i^{j+1} = \left(1 - 2\frac{D\tau}{h^2}\right) y_i^j + \frac{D\tau}{h^2}(y_{i-1}^j + y_{i+1}^j), \quad i = \overline{1, n-1}, \ j = \overline{0, m-1}, \\ &y_i^0 = u_0(x_i), \quad i = \overline{0, n}, \\ &y_0^{j+1} = u_L(t_{j+1}), \quad y_n^{j+1} = u_R(t_{j+1}), \quad j = \overline{0, m-1}. \end{aligned} \tag{3.7}$$

Имплементирането на една диференчна схема от горния вид трябва да следва следния алгоритъм:

1. Използвайки началното условие, намираме стойностите на първия слой по времето.

2. Итерираме по $j = \overline{0, m-1}$. На всяка итерация от стойностите на j-тия слой намираме тези на $j+1$-вия.

   - От апроксимацията на основното диференциално уравнение намираме стойностите във вътрешните точки от мрежата.
   - От граничните условия намираме стойностите на приближеното решение в граничните точки.

Ще предложим функция в Mathematica, която имплементира горната диференчна схема.

```
In[48]:= heatEquation[l_, T_, h_, u0_, uL_, uR_, d_] := (
  τ = h^2 / (3 d);
  n = Ceiling[l / h];
  m = Ceiling[T / τ];
  t = Table[(j - 1) τ, {j, 1, m + 1}];
  x = Table[(i - 1) h, {i, 1, n + 1}];
  y = Table[0, {n + 1}, {m + 1}];
  (*Initial condition*)
  For[i = 1, i ≤ n + 1, i++,
   y[[i, 1]] = u0[x[[i]]]
  ];
  (*Iterate over the time nodes*)
  For[j = 1, j ≤ m, j++,
   (*Compute the internal values from the main PDE*)
   For[i = 2, i ≤ n, i++,
    y[[i, j + 1]] = (1 - 2 d τ / h^2) y[[i, j]] + d τ / h^2 (y[[i - 1, j]] + y[[i + 1, j]])
   ];
   (*Compute the boundary values from the boundary conditions*)
   y[[1, j + 1]] = uL[t[[j + 1]]];
   y[[n + 1, j + 1]] = uR[t[[j + 1]]]
  ];
  {x, t, y}
 )
```

Когато изследваме един диференчен метод, е добре да можем да го тестваме в частен случай на задачата, за който е известно точно решение (когато това е възможно). За тази цел ще приложим имплементираната диференчна схема за следната задача:

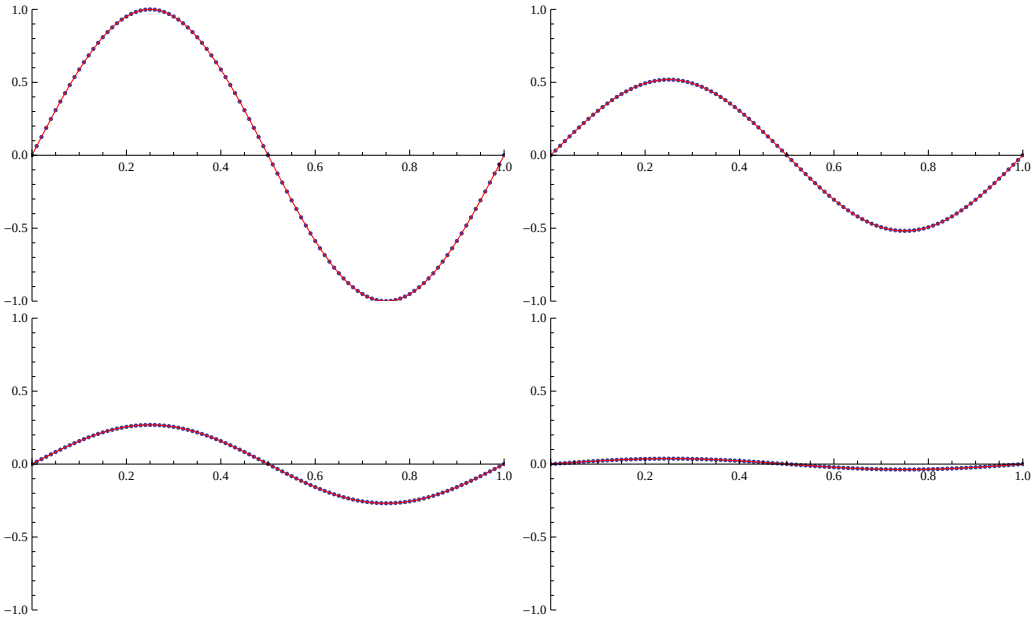
$$\begin{aligned} &\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \quad 0 < x < 1,\ 0 < t \leq 0.1 \\ &u(x,0) = \sin(2\pi x), \\ &u(0,t) = u(1,t) = 0, \end{aligned} \tag{3.8}$$

която има точно решение

$$u(x,t) = e^{-4\pi^2 t} \sin(2\pi x). \tag{3.9}$$

```
u0[x_] := Sin[2 Pi x]
uL[t_] := 0
uR[t_] := 0
resX = heatEquation[1, 0.5, 0.01, u0, uL, uR, 1][[1]];
resT = heatEquation[1, 0.5, 0.01, u0, uL, uR, 1][[2]];
resY = heatEquation[1, 0.5, 0.01, u0, uL, uR, 1][[3]];
Manipulate[
 Show[
  ListPlot[
   Table[{resX[[i]], resY[[i, j]]}, {i, 1, Length[resX]}], PlotRange → All],
  Plot[E^-4 Pi^2 resT[[j]] Sin[2 Pi x], {x, 0, 1}, PlotStyle → Red]
 ],
 {j, 1, Length[resT], 1}]
```

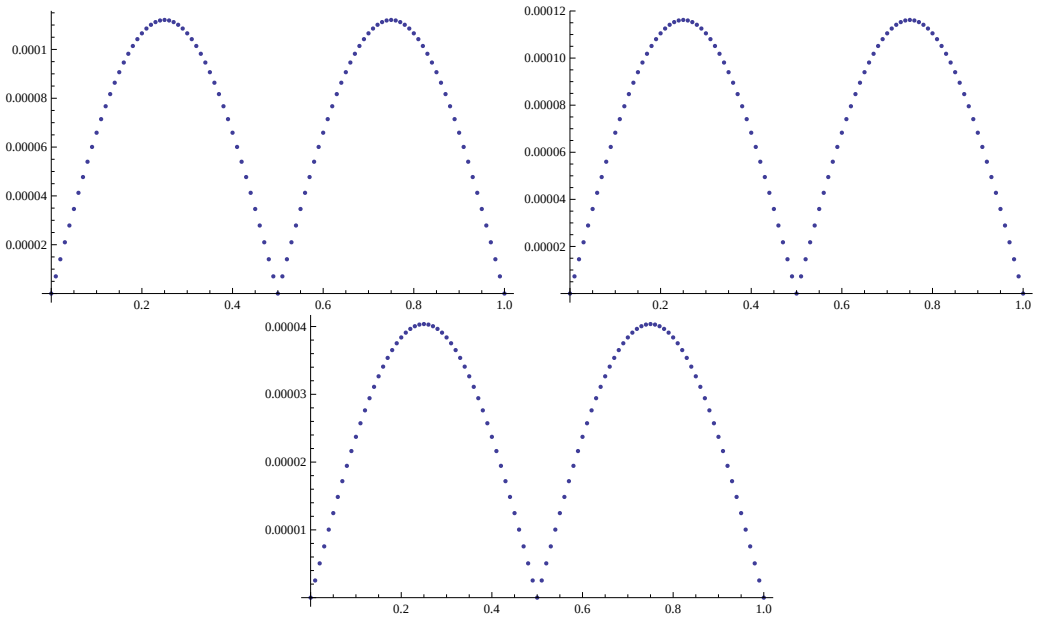
Графиките на приближеното и точното решение (съответно със сини точки и червена линия) за $t = 0, 0.015, 0.03, 0.08$ са приложени по-долу:



Визуално, приближеното и точното решение съвпадат.

В началния момент от време температурата в първата половина от интервала е положителна, а във втората – отрицателна. Граничните условия означават, че в двата края на областта има поглъщател на топлина, който поддържа температурата 0 (нека за определеност мислим, че температурата е в $°C$). С течение на времето, вследствие на дифузията, температурата в лявата половина намалява, а в дясната страна расте, като окончателно се установява около равновесно състояние $0°C$ в цялата дефиниционна област.

Съответните стойности на грешката по абсолютна стойност в точките от мрежата (пропускаме $t = 0$, тъй като там грешката очевидно е 0) са:



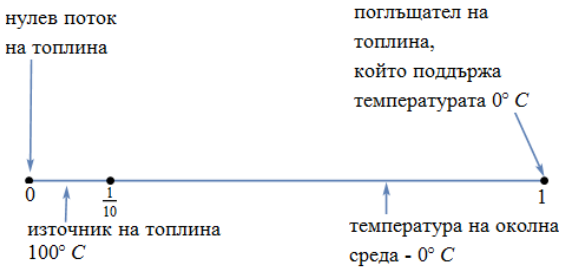
За разгледания времеви интервал максимумът на грешката намалява с цял порядък – от 10^{-4} до 10^{-5}. Това е свързано със свойството на параболичния диференциален оператор, който има изглаждащ ефект. Засега ще се ограничим с този лаконичен коментар. По-подробно ще коментираме въпроса, когато разглеждаме хиперболичните уравнения.

Числените експерименти върху разгледаната моделна задача дават добри резултати и поведението на числения метод върху нея е добро, което ни дава основание да го използваме и върху други задачи.

Преди да преминем към такива обаче, е необходимо да коментираме въпроса за устойчивостта на разглежданите диференчни методи. В конкретния случай изследването за устойчивост е свързано с т.нар. условие за положителност на коефициентите. Тук ще дадем формулировка, която не е много точна, но е достатъчна за целите на разглежданите задачи.

Ако всички коефициенти в каноничния вид (3.7) са положителни числа, то методът е устойчив. По-точна формулировка на цитираното условие може да се намери в [3].

Нека сега разгледаме задача за едномерно тяло с начална температура $0°C$, която схематично е представена по-долу:



Освен дифузията, в този случай ще разгледаме топлообмен с външната среда. Тогава

изменението на температурата = дифузионен член+топлообмен с външната среда.

Топлообменът с външната среда е пропорционален на разликата в температурата на тялото и външната среда, т.е. на $v(x) - u(x,t)$, където

$$v(x) := \begin{cases} 100, & x \leq 1/10 \\ 0, & x > 1/10 \end{cases}.$$

Нека означим коефициента на пропорционалност с $\varkappa$. В числените експерименти ще приемем, че $\varkappa = 1$.

Тогава диференциалното уравнение, описващо процеса, е

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + \varkappa(v(x) - u).$$

Дясното гранично условие е от първи род – $u(1,t) = 0$.

За да наложим условие за нулев поток на топлина на лявата граница, ще вземем предвид, че потокът на топлина е пропорционален на $\partial u/\partial x$, т.е. налагаме условието

$$\partial u/\partial x(0,t) = 0.$$

Гранични условия от вида

$$\partial u/\partial x(0,t) = b_0$$

се наричат гранични условия от втори род или гранични условия на фон-Нойман.

И така, разглеждаме диференциалната задача

$$\begin{aligned} &\frac{\partial u}{\partial t} - D\frac{\partial^2 u}{\partial x^2} + \varkappa u = v(x)\varkappa, \quad 0 < x < 1,\ 0 < t \leq 0.1 \\ &u(x,0) = 0, \\ &\frac{\partial u}{\partial x}(0,t) = 0, \\ &u(1,t) = 0. \end{aligned} \tag{3.10}$$

За апроксимацията на диференциалното уравнение, началното и дясното гранични условия подхождаме, както и в предишния пример.

Получаваме

$$\begin{aligned} &\frac{y_i^{j+1} - y_i^j}{\tau} - D\frac{y_{i+1}^j - 2y_i^j + y_{i-1}^j}{h^2} + \varkappa y_i^j = v(x_i)\varkappa,\ i = \overline{1, n-1},\ j = \overline{1, m-1} \\ &y_i^0 = 0,\ i = \overline{0, n}, \\ &y_n^j = 0,\ j = \overline{1, m}. \end{aligned} \tag{3.11}$$

Записано в каноничен вид, първото уравнение е

$$y_i^{j+1} = \frac{D\tau}{h^2}(y_{i+1}^j + y_{i-1}^j) + \left(1 - \frac{2D\tau}{h^2} - \tau\varkappa\right) y_i^j + v(x_i)\tau\varkappa.$$

Нека отбележим, че неговата ЛГА е $O(h^2+\tau)$.

От условието за положителност на коефициентите получаваме

$$1-\frac{2D\tau}{h^2}-\tau\varkappa\leq 0\Longrightarrow\tau\leq\frac{h^2}{2D+\varkappa h^2}.$$

Нека сега разгледаме въпроса за апроксимирането на лявото гранично условие, което е от втори род. Бихме искали да използваме формулата за числено диференциране с централна разлика, тъй като тя има втори ред на точност. Това обаче, разбира се, е невъзможно, тъй като нямаме точка от мрежата вляво от границата. За да го апроксимираме, бихме могли да използваме единствено формулата за числено диференциране с разлика напред. Получаваме диференчното уравнение

$$\frac{y_1^j-y_0^j}{h}=0,$$

което има ЛГА $O(h)$. Ако в граничните точки обаче въвеждаме грешка, която е от по-нисък ред, то е безполезен фактът, че в останалите точки грешката ни е от втори ред. Сега ще видим как можем да повишим реда на апроксимация за граничното условие от втори род. Бихме могли да запишем следното равенство

$$\frac{u_1^j-u_0^j}{h}=0+O(h).$$

Бихме могли обаче да развием остатъчния член и да получим за него вида $\diamond+O(h^2)$ тогава последното уравнение добива вида

$$\frac{u_1^j-u_0^j}{h}=0+\diamond+O(h^2)$$

или

$$\frac{u_1^j-u_0^j}{h}-\diamond=0+O(h^2)$$

и следователно диференчното уравнение

$$\frac{y_1^j-y_0^j}{h}-\diamond=0 \tag{3.12}$$

ще апроксимира лявото гранично условие с ЛГА $O(h^2)$. Остатъчният член $O(h)$, който искаме да представим в друг вид, е всъщност ЛГА. Нека я разпишем, като развием в ред на Тейлър около $x=0$:

$$\begin{aligned}\psi_{h,\tau}&=\frac{1}{h}\left(u_0^j+\left.\frac{\partial u}{\partial x}\right|_0^j h+\left.\frac{\partial^2 u}{\partial x^2}\right|_0^j\frac{h^2}{2}+O(h^3)-u_0^j\right)\\&=\left.\frac{h}{2}\,\frac{\partial^2 u}{\partial x^2}\right|_0^j+O(h^2).\end{aligned}$$

В последното използвахме, че $\frac{\partial u}{\partial x}(0,t)=0$. И така, получихме представяне на ЛГА в желания вид. Тогава, ако заместим $\diamond$ с $\left.\frac{h}{2}\frac{\partial^2 u}{\partial x^2}\right|_0^j$ (по-точно, апроксимацията на последното) в лявата страна на (3.12), ще получим апроксимация с втори ред на точност. Проблемът обаче е, че по абсолютно същата причина, поради

която не можахме да използваме формулата с централна разлика за апроксимация на производната от първи ред, не можем да апроксимираме и втората производна. За да решим този проблем правим едно съществено допускане, а именно – че основното диференциално уравнение (3.10) е изпълнено и при $x = 0$. Това допускане е обосновано, тъй като физическите процеси имат достатъчно гладко поведение и следователно фактът, че (3.10) е изпълнено върху отворения интервал $x \in (0, 1)$, ни дава основание да приемем, че в граничните точки също е изпълнено с достатъчно малка грешка.

Тогава можем да изразим втората производна по пространството чрез производната по времето. Получаваме

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{D}\left(\frac{\partial u}{\partial t} - \varkappa(v(0) - u)\right).$$

Тогава, като използваме последното, заместваме $\diamond$ в (3.12) и получаваме окончателно апроксимация на граничното условие с грешка $O(h^2 + \tau)$:

$$\frac{y_1^j - y_0^j}{h} - \frac{h}{2}\frac{1}{D}\left(\frac{y_0^{j+1} - y_0^j}{\tau} - \varkappa(v(0) - y_0^j)\right) = 0.$$

Записано в каноничен вид, диференчното уравнение добива вида

$$y_0^{j+1} = \left(1 - \frac{2\tau D}{h^2} - \varkappa\tau\right) y_0^j + \frac{2D\tau}{h^2} y_1^j + v(0)\varkappa\tau. \tag{3.13}$$

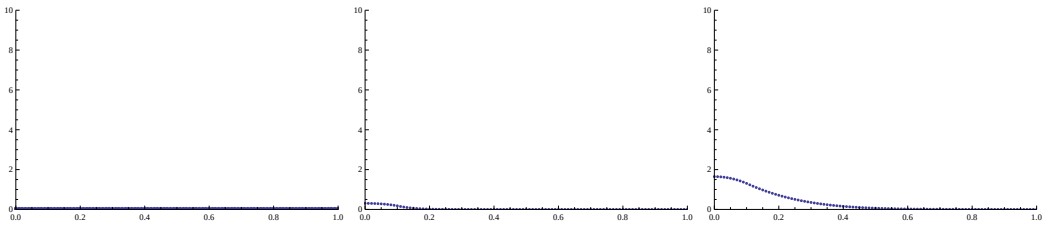
И така, получихме диференчната схема (3.11), (3.13).

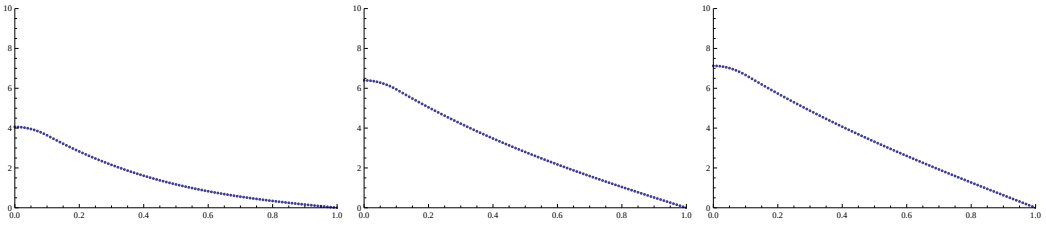
Прилагаме нейна примерна имплементация в Mathematica:

In[1]:=
```
heatEquation[l_, T_, h_, u0_, d_, κ_] := (
  τ = h^2 / (3 d + κ h^2);
  n = Ceiling[l / h];
  m = Ceiling[T / τ];
  t = Table[(j - 1) τ, {j, 1, m + 1}];
  x = Table[(i - 1) h, {i, 1, n + 1}];
  y = Table[0, {n + 1}, {m + 1}];
  (*Initial condition*)
  For[i = 1, i ≤ n + 1, i++,
   y[[i, 1]] = u0[x[[i]]]
  ];
  (*Iterate over the time nodes*)
  For[j = 1, j ≤ m, j++,
   (*Compute the internal values from the main PDE*)
   For[i = 2, i ≤ n, i++,
    y[[i, j + 1]] =
     (1 - 2 d τ / h^2 - τ κ) y[[i, j]] + d τ / h^2 (y[[i - 1, j]] + y[[i + 1, j]]) + v[x[[i]]] τ κ
   ];
   (*Compute the boundary values from the boundary conditions*)
   y[[1, j + 1]] = (1 - 2 d τ / h^2 - τ κ) y[[1, j]] + 2 d τ / h^2 y[[2, j]] + 100 τ κ;
   y[[n + 1, j + 1]] = 0
  ];
  {x, t, y}
 )
```

Нека разгледаме приближеното решение за $t = 0, 0.003, 0.03, 0.17, 0.5, 0.83$:

In[2]:=
```
u0[x_] := 0
v[x_] := If[x ≤ 1 / 10, 100, 0]
resX = heatEquation[1, 1, 0.01, u0, 1, 1][[1]];
resT = heatEquation[1, 1, 0.01, u0, 1, 1][[2]];
resY = heatEquation[1, 1, 0.01, u0, 1, 1][[3]];
Manipulate[
 ListPlot[Table[{resX[[i]], resY[[i, j]]}, {i, 1, Length[resX]}],
  PlotRange → {{0, 1}, {0, 10}}],
 {j, 1, Length[resT], 1}]
```





Вследствие на източника на топлина, температурата се повишава в началото на интервала, като дифузията я разпространява в целия интервал. Поради поглъщателя на топлина на дясната граница, температурният профил достига равновесното си състояние, изобразено на последната графика.

3.1.2 Чисто неявна схема за уравнението на дифузията

Условието за устойчивост при явната схема означава, че стъпката по времето, τ, трябва да е от порядъка на h^2. Това може да доведе до необходимостта от работата с много малка стъпка по времето.

Опитът ни от решаването на ОДУ подсказва, че този проблем би могъл да се реши, ако използваме неявни методи за решаването на задачата. Ще изложим идеята отново върху задачата (3.8), имаща точно решение (3.9). За удобство, нека тук отново приложим основното диференциално уравнение в (3.8):

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \quad 0 < x < 1, \ 0 < t \leq 0.1 \tag{3.14}$$

При извеждането на явната схема, апроксимирахме основното уравнение (3.14) на j-тия слой по времето (вж. (3.5)). Получихме

$$\frac{y_i^{j+1} - y_i^j}{\tau} - \frac{y_{i-1}^j - 2y_i^j + y_{i+1}^j}{h^2} = 0, \quad i = \overline{1, n-1}, \ j = \overline{0, m-1}, \tag{3.15}$$

където единственото неизвестно е y_i^{j+1}.

Можем обаче да подходим по аналогичен начин, като направим апроксимацията на $j+1$-вия слой, като за производната по времето използваме формулата с разлика назад. Получаваме

$$\frac{y_i^{j+1} - y_i^j}{\tau} - \frac{y_{i-1}^{j+1} - 2y_i^{j+1} + y_{i+1}^{j+1}}{h^2} = 0, \quad i = \overline{1, n-1}, \ j = \overline{0, m-1}. \tag{3.16}$$

Очевидно получената апроксимация е неявна, тъй като в уравнението участват 3 неизвестни – $y_{i-1}^{j+1}, y_i^{j+1}, y_{i+1}^{j+1}$. Ако разгледаме каноничния вид на (3.16) обаче, става ясно какво е предимството в този случай, а именно – устойчивост при произволен избор на стъпката по времето (вж. [3]).

Няма да се спираме по-подробно на чисто неявната схема, тъй като нейната имплементация е аналогична на тази при схемата на Кранк–Никълсън, която ще разгледаме в следващата секция.

3.1.3 Схеми с тегло. Метод на Кранк–Никълсън.

От (3.15) и (3.16) можем да получим клас от схеми с тегло. Умножавайки двете страни на (3.15) със σ, а тези на (3.5) с $1-\sigma$ и събирайки почленно, получаваме

$$\frac{y_i^{j+1}-y_i^j}{\tau}-\sigma\frac{y_{i-1}^j-2y_i^j+y_{i+1}^j}{h^2}-(1-\sigma)\frac{y_{i-1}^{j+1}-2y_i^{j+1}+y_{i+1}^{j+1}}{h^2}=0,\ \ i=\overline{1,n-1},\ j=\overline{0,m-1}. \tag{3.17}$$

Може да се покаже (вж. [3]), че най-добрият избор на σ, от гледна точка на точност на метода, е $\sigma=1/2$. Получената схема се нарича схема на Кранк–Никълсън и има ЛГА $O(h^2+\tau^2)$.

Очевидно, за да намерим стойностите на приближеното решение на $j+1$-вия слой по времето, трябва да решаваме система от $n+1$ уравнения (включвайки и уравненията, получени от граничните условия).

За схемата на Кранк–Никълсън линейната система, която трябва да се решава на всеки слой по времето за зад. (3.8), има вида (проверете!):

$$\begin{bmatrix} 1 & 0 & & & \\ -\frac{1}{2h^2} & \frac{1}{\tau}+\frac{1}{h^2} & -\frac{1}{2h^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2h^2} & \frac{1}{\tau}+\frac{1}{h^2} & -\frac{1}{2h^2} \\ & & & 0 & 1 \end{bmatrix} \begin{bmatrix} y_0^{j+1} \\ y_1^{j+1} \\ \vdots \\ y_{n-1}^{j+1} \\ y_n^{j+1} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{y_1^j}{\tau}+\frac{y_0^j-2y_1^j+y_2^j}{2h^2} \\ \vdots \\ \frac{y_{n-1}^j}{\tau}+\frac{y_{n-2}^j-2y_{n-1}^j+y_n^j}{2h^2} \\ 0 \end{bmatrix} \tag{3.18}$$

За да проверим кога ще е устойчива схемата, привеждаме диференчната схема в каноничен вид (изразяваме y_i^{j+1}):

$$\left(\frac{1}{\tau}+\frac{1}{h^2}\right)y_i^{j+1}=\frac{1}{2h^2}(y_{i-1}^j+y_{i+1}^j+y_{i-1}^{j+1}+y_{i+1}^{j+1})+\left(\frac{1}{\tau}-\frac{1}{h^2}\right)y_i^j. \tag{3.19}$$

Коефициентите са положителни, когато $\frac{\tau}{h^2}\leq 1$ и следователно схемата ще е устойчива, ако изберем $\tau=\frac{h^2}{2}$.

Имплементирането на една неявна диференчна схема от горния вид трябва да следва следния алгоритъм:

1. Използвайки началното условие, намираме стойностите на първия слой по времето.

2. Итерираме по $j=\overline{0,m-1}$. На всяка итерация от стойностите на j-тия слой намираме тези на $j+1$-вия, като решаваме системата (3.18).

Прилагаме следната примерна имплементация:

In[16]:=
```
CrankNicolson[l_, T_, h_, u0_, uL_, uR_] := (
  τ = h^2/2;
  n = Ceiling[l/h];
  m = Ceiling[T/τ];
  x = Table[(i-1)*h, {i, 1, n+1}];
  t = Table[(i-1)*τ, {i, 1, m+1}];
  y = Table[0, {n+1}, {m+1}];

  (*Initial condition*)
  For[i = 1, i ≤ n+1, i++,
    y[[i, 1]] = u0[x[[i]]];
  ];

  (*generate the matrix of the coefficients of the system. It
    is the same for every time step therefore it is not needed to be generated in the loop.*)
  A = Table[0, {n+1}, {n+1}];
  A[[1, 1]] = 1;
  A[[n+1, n+1]] = 1;

  For[i = 2, i ≤ n, i++,
    A[[i, i-1]] = -1/(2 h^2);
    A[[i, i]] = 1/τ+1/h^2;
    A[[i, i+1]] = -1/(2 h^2);
  ];

  (*Iterate over time*)
  For[j = 1, j ≤ m, j++,
      (*Boundary conditions*)
      y[[1, j+1]] = uL[t[[j]]];
      y[[n+1, j+1]] = uR[t[[j]]];

      (*Internal values from the main PDE*)
      For[i = 2, i ≤ n, i++,
        (*generate the vector of the RHS*)
        b = Table[y[[i, j]]/τ+(y[[i-1, j]]-2 y[[i, j]]+y[[i+1, j]])/(2 h^2), {i, 2, n}];
        b = Flatten[{0, b, 0}];
      ];

      (*solve the system*)
      y[[All, j+1]] = LinearSolve[A, b];
  ];

  {x, t, y}
)
```

3.1.4 Явна диференчна схема за едномерното уравнение на преноса

Наред с дифузията, другият основен процес, който обуславя движението на вещество, темепература и др. е конвекцията или простия пренос. Като най-прост пример за такъв тип движение ще дадем концентрацията на вещество, пуснато в течаща вода (например река). Причината за преноса на веществото (ако пренебрегнем дифузията) е самото движение на водата, т.е. тя ще го пренася със скорост c, с която се движи реката.

Математическият модел, описващ този процес, е хиперболичното частно диференциално уравнение от първи ред

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0. \tag{3.20}$$

Последното се нарича уравнение на преноса или уравнение на конвекцията.

Ще илюстрираме идеята за числено решаване, като дадем пример с гранична задача в областта $\mathbb{D} = \{(x,t) : -10 \leq x \leq 10, 0 \leq t \leq T\}$. За да затворим задачата, са ни необходими едно начално условие и едно гранично условие (в диференциалното уравнение има първа производна по пространството и първа производна по времето). Нека да изберем следните условия:

$$u(x,0) = u_0(x) = \begin{cases} 25, & x \leq 0 \\ 25 - 25x, & 0 \leq x \leq 1 \\ 0, & x \geq 1 \end{cases}$$
$$u(0,t) = u_L(t) = 25.$$

Ще построим явна двуслойна схема за решаване на горната задача. За тази цел нека разгледаме два произволни слоя по времето j, $j+1$ от мрежата

$$\overline{\omega}_{h,\tau} := \{(x_i, t_j) : x_i = -10 + ih, t_j = j\tau, i = \overline{0,n}, j = \overline{0,m}, n = 20/h, m = T/\tau\}.$$

Първата ни цел ще бъде да апроксимираме основното диференциално уравнение (3.20) с **устойчиво** диференчно уравнение, като апроксимацията ще направим в точката (x_i, t_j).

Предвид факта, че разглеждаме двуслойна диференчна схема, е ясно, че за апроксимацията на производната по времето ще използваме формула за числено диференциране с разлика напред:

$$\frac{\partial u}{\partial t} = \frac{u_i^{j+1} - u_i^j}{\tau} + O(\tau).$$

За апроксимацията на производната по пространството обаче имаме три варианта – формулите за числено диференциране с централна разлика, с разлика напред и разлика назад. Да разгледаме какви апроксимации можем да получим, използвайки трите формули.

1. **Формула с централна разлика.** Имайки предвид, че тази формула има най-добър ред на апроксимация, бихме искали да започнем с нея. Тогава апроксимацията на (3.20) има вида

$$\frac{y_i^{j+1} - y_i^j}{\tau} + c\frac{y_{i+1}^j - y_{i-1}^j}{2h} = 0,$$

което, записано в каноничен вид, е

$$y_i^{j+1} = y_i^j + \frac{c\tau}{2h}(y_{i-1}^j - y_{i+1}^j).$$

Очевидно е, че не могат да се изберат стойности на стъпките τ и h така, че условието за положителност на коефициентите да е изпълнено.

2. **Формула с разлика напред.** В този случай получаваме диференчното уравнение
$$\frac{y_i^{j+1} - y_i^j}{\tau} + c\frac{y_{i+1}^j - y_i^j}{h} = 0,$$
което, записано в каноничен вид, е
$$y_i^{j+1} = \left(1 + \frac{c\tau}{h}\right) y_i^j - \frac{c\tau}{h} y_{i+1}^j.$$
И в този случай се оказва, че не съществуват стойности на стъпките τ и h такива, че условието за положителност на коефициентите да е изпълнено.

3. **Формула с разлика назад.** В този случай получаваме диференчното уравнение
$$\frac{y_i^{j+1} - y_i^j}{\tau} + c\frac{y_i^j - y_{i-1}^j}{h} = 0,$$
каноничният вид, на което е
$$y_i^{j+1} = \left(1 - \frac{c\tau}{h}\right) y_i^j + \frac{c\tau}{h} y_{i-1}^j. \tag{3.21}$$
Сега вече е възможно да изберем стъпки, които да удовлетворяват условието за положителност на коефициентите. За тази цел е необходимо да бъде изпълнено неравенството
$$1 - \frac{c\tau}{h} \geq 0,$$
т.е. $\tau \leq h/c$.

И така, за да получим устойчива диференчна схема, ще използваме апроксимацията (3.21), която има локална грешка $O(h + \tau)$. Началното и лявото гранични условия се апроксимират тривиално:

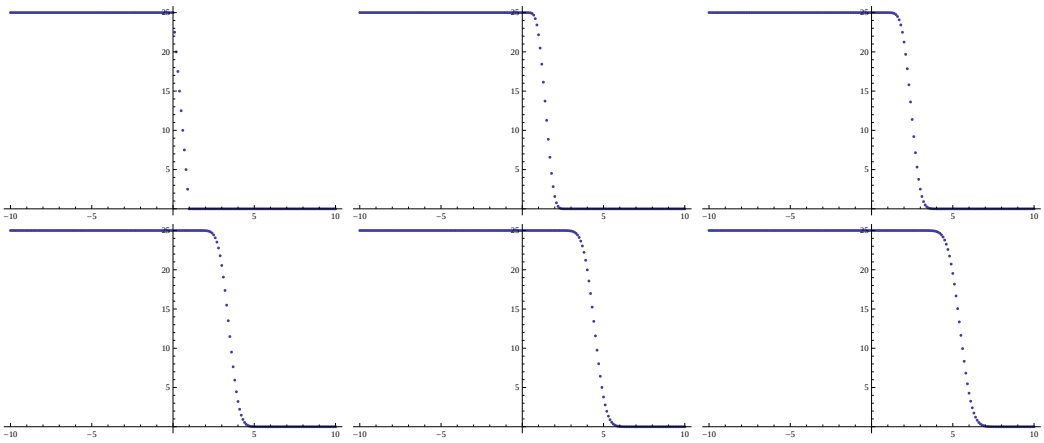
$$\begin{aligned} y_i^0 &= u_0(x_i),\ i = \overline{0,n}, \\ y_0^j &= 25,\ j = \overline{0,m}. \end{aligned} \tag{3.22}$$

И така, получихме явна двуслойна диференчна схема с ЛГА $O(h + \tau)$, чийто каноничен вид се състои от диференчните уравнения (3.21) и (3.22) и която е устойчива при $\tau \leq h/c$.

Да разгледаме една примерна имплементация на диференчната схема:

```
Transport[l_, T_, h_, u0_, uL_, c_] := (
    τ = h / (2 c);
    n = Ceiling[l / h];
    m = Ceiling[T / τ];
    x = Table[-10 + (i - 1) * h, {i, 1, n + 1}];
    t = Table[(j - 1) * τ, {j, 1, m + 1}];
    y = Table[0, {n + 1}, {m + 1}];

    (*Initial condition*)
    For[i = 1, i ≤ n + 1, i++,
      y[[i, 1]] = u0[x[[i]]];
    ];

    (*Iterate over time*)
    For[j = 1, j ≤ m,  j++,
            (*Boundary condition*)
            y[[1, j + 1]] = uL[t[[j]]];

            (*Internal values from the main PDE*)
            For[i = 2,  i ≤ n,  i++,
                    y[[i, j + 1]] = (1 - c * τ / h) y[[i, j]] + c τ / h * y[[i - 1, j]]
              ];
    ];

    {x, t, y}
  )
u0[x_] := Which[x ≤ 0,  25,  x ≤ 1, 25 - 25 * x,  True,  0]
uL[x_] := 25
apprSol  =  Transport[20, 5, 0.01, u0, uL, 1];
apprX  =  apprSol[[1]];
apprT  =  apprSol[[2]];
apprY  =  apprSol[[3]];

Manipulate[
  Show[
      Plot[u0[x - apprT[[j]]], {x, 0, 10}, PlotStyle → Red, PlotRange → {{0, 10}, {0, 25}}],
    ListPlot[Transpose[{apprX, apprY[[All, j]]}], PlotRange → {{0, 10}, {0, 1.5}}]
  ],
   {j, 1, Length[apprT], 1}]
```

Използвайки горния код, ще представим решението за $t = 0, 1, 2, 3, 4, 5$:



Профилът на решението се запазва с течение на времето, като се пренася

надясно със скорост c. Това е пример за т.нар. решения от тип "бягаща вълна".

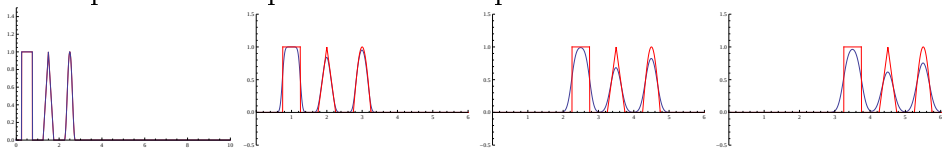
Както видяхме, при параболичните уравнения особеностите в началното условие не водят до съществени проблеми при числено решаване на дадена диференциална задача, предвид изглаждащия ефект на параболичния оператор. Нека видим дали това е така и при хиперболичните уравнения. Да разгледаме за целта следната задача:

$$\begin{aligned} &\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \quad -10 < x < 10,\ 0 < t \leq 5, \\ &u(x,0) = \begin{cases} 1, & 1/4 < x < 3/4, \\ 1 - |4x - 6|, & 5/4 < x < 7/4, \\ \cos^2[\pi(2x-5)], & 9/4 < x < 11/4, \\ 0, & \text{иначе}, \end{cases} \\ &u(0,t) = 0. \end{aligned} \tag{3.23}$$

Преизползваме функцията, която написахме в зад. 3.20:

```
u0[x_] := Which[And[x ≥ 1/4, x ≤ 3/4], 1, And[x ≥ 5/4, x ≤ 7/4],
   1 - Abs[4 * x - 6], And[x ≥ 9/4, x ≤ 11/4], (Cos[Pi (2 x - 5)])^2, True, 0]
uL[x_] := 0
apprSol = Transport[20, 5, 0.01, u0, uL, 1];
apprX = apprSol[[1]];
apprT = apprSol[[2]];
apprY = apprSol[[3]];

Manipulate[
  Show[
    Plot[u0[x - apprT[[j]]], {x, 0, 10}, PlotStyle → Red, PlotRange → {{0, 10}, {0, 1.5}}],
    ListPlot[Transpose[{apprX, apprY[[All, j]]}]]
  ],
  {j, 1, Length[apprT], 1}]
```

Резултатите за $j = 0, 100, 400, 600$ са изобразени на следващите графики, като с червено е изобразено точното решение.



Както можем да забележим, приближеното решение с течение на времето губи от първоначалната енергия и намалява, като се отдалечава от точното. Освен това, правоъгълната част от сигнала се заглажда и загубва първоначалната си форма. Този ефект е известен като "числена дифузия".

3.1.5 Диференчни схеми за едномерното уравнение на преноса с втори ред на апроксимация, устойчиви в l_2-норма

За да избегнем съществената загуба на точност, която наблюдавахме в последния пример, бихме искали да използваме схеми с по-висок ред на апроксимация, т.е. $O(\tau^2 + h^2)$. Една такава схема за решаване на уравнението на преноса е схемата на Lax-Wendroff. Тя може да бъде изведена и като се използва подход,

аналогичен на това, което сме разгледали досега в курса, но тук ще приложим едно различно извеждане на схемата.

За да приближим решението на уравнението (3.20) на следващия слой по времето u_i^{j+1} с ЛГА $O(h^2+\tau^2)$, развиваме в ред на Тейлър около t_j:

$$u_i^{j+1} = u_i^j + \left.\frac{\partial u}{\partial t}\right|_i^j \tau + \left.\frac{\partial^2 u}{\partial t^2}\right|_i^j \frac{\tau^2}{2} + O(\tau^3). \tag{3.24}$$

От (3.20) имаме

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x},$$

и диференцирайки двете страни на последното уравнение по t, получаваме последователно

$$\begin{aligned}\frac{\partial^2 u}{\partial t^2} &= -c\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}\right) = -c\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial t}\right)\\ &= -c\frac{\partial}{\partial x}\left(-c\frac{\partial u}{\partial x}\right) = c^2\frac{\partial^2 u}{\partial x^2}.\end{aligned}$$

От заместването на последните резултати в (**??**) следва

$$u_i^{j+1} = u_i^j - c\left.\frac{\partial u}{\partial x}\right|_i^j \tau + \frac{c^2\tau^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_i^j + O(\tau^3). \tag{3.25}$$

Пренебрегвайки остатъчния член и минавайки към приближеното решение, получаваме следното диференчно уравнение:

$$y_i^{j+1} = y_i^j - \frac{c\tau}{2h}(y_{i+1}^j - y_{i-1}^j) + \frac{c^2\tau^2}{2h^2}(y_{i+1}^j - 2y_i^j + y_{i-1}^j).$$

Получената апроксимация има ЛГА $O(h^2+\tau^2)$, тъй като (**??**) има ЛГА $O(\tau^2)$ и производните по x в **??** са приближени с ЛГА $O(h^2)$. Привеждаме в каноничен вид

$$y_i^{j+1} = \frac{c\tau}{2h}\left(1+\frac{c\tau}{h}\right)y_{i-1}^j + \left(1-\frac{c\tau}{h}\right)\left(1+\frac{c\tau}{h}\right)y_i^j - \frac{c\tau}{2h}\left(1-\frac{c\tau}{h}\right)y_{i+1}^j.$$

Следователно схемата не е устойчива в C-норма, тъй като не удовлетворява условията за положителност на коефициентите. Интуитивно[2], това означава, че големината на грешка, допусната на j-тия слой по времето, може да стане по-голяма на следващия, т.е. $\|\varepsilon^{j+1}\|_C > \|\varepsilon^j\|_C$, където

$$\|\varepsilon^j\|_C := \max_{i=0,\ldots,n} |\varepsilon_i^j|.$$

Това, че максимумът на грешката може да расте обаче, не означава, че задължително това нарастване ще бъде драстично и методът ще бъде неизползваем. По-слабо изискване е устойчивостта в мрежова l_2-норма. В някакъв смисъл това означава устойчивост на средната грешка. Максимумът може и да расте, но средната грешка остава контролирана. По-точно мрежовата l_2-норма се дефинира така:

$$\|\varepsilon^j\|_{l_2} := \sqrt{\sum_{i=0}^{n}(\varepsilon_i^j)^2}$$

[2]Дадените обяснения за устойчивост не са точни. Те имат за цел единствено добиването на някаква интуиция за разликата между устойчивост в мрежова C-норма и мрежова l_2-норма.

.

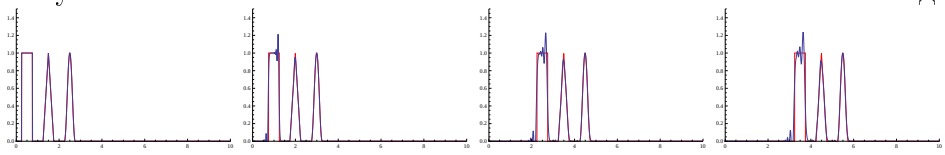
Схемата на Lax–Wenderoff е устойчива в l_2-норма, ако за числото на Courant е изпълнено $CourantNumber := \frac{c\tau}{h} \leq 1$. Това твърдение се доказва по метода на хармониките (вж. [3]).

По аналогичен начин може да се докаже, че централна разлика води до числена схема, която е устойчива в l_2-норма.

Прилагаме примерна имплементация на схемата на Lax-Wendroff за задачата (3.23).

```
Transport2Order[l_, T_, h_, u0_, uL_, c_] := (
    τ = h/(2 c);
    n = Ceiling[l/h];
    m = Ceiling[T/τ];
    x = Table[-10 + (i-1)*h, {i, 1, n+1}];
    t = Table[(j-1)*τ, {j, 1, m+1}];
    y = Table[0, {n+1}, {m+1}];

    (*Initial condition*)
    For[i = 1, i ≤ n+1, i++,
      y[[i, 1]] = u0[x[[i]]];
    ];

    (*Iterate over time*)
    For[j = 1, j ≤ m, j++,
            (*Boundary condition*)
            y[[1, j+1]] = uL[t[[j]]];

            (*Internal values from the main PDE*)
            For[i = 2, i ≤ n, i++,

      y[[i, j+1]] = c τ/(2 h) (1 + c*τ/h)*y[[i-1, j]] + (1 - c*τ/h) (1 + c*τ/h) y[[i, j]] + c τ/(2 h) (c*τ/h - 1)*y[[i+1, j]]
                ];
    ];

    {x, t, y}
  )
```

Резултатите от използването на схемата на Lax-Wendroff са следните:



Както виждаме, при нея приближеното решение не се отдалечава значително от точното за разглеждания интервал от време, т.е. този метод запазва много по-добре енергията в системата. Отново се наблюдава числена дифузия, особено в точките на прекъсване, но този ефект е значително по-малък. От друга страна обаче, схемата губи монотонност и се наблюдават нефизични осцилации около особеностите на решението. Причината за това е, че тя вече не е устойчива в мрежова C-норма, т.е. няма контрол върху максимума на грешката, а само върху "общото" ѝ поведение. Съществуват различни подходи за намаляването на този ефект, но този въпрос излиза извън рамките на настоящия курс.

И така, при решаването на хиперболични уравнения трябва да се имат предвид съответните особености и трудности. Това е тема, която и днес продължава

да бъде активна област на изследване.

3.2 Допълнителни задачи

Задача 16. Като се използва условието за положителност на коефициентите, да се изследва устойчивостта на чисто неявна двуслойна схема за уравнението на преноса, която използва за апроксимиране на производната по пространството формула с разлика напред / разлика назад/ централна разлика.

Да се имплементира една от така получените схеми и да се приложи за решаването на задачата от секцията "Явна диференчна схема за едномерното уравнение на преноса".

Задача 17. Да се построи явна диференчна схема с ЛГА $O(h^2 + \tau)$ за диференциалната задача

$$\begin{aligned}
&\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = \sin(xt), \quad 0 < x < 1,\ 0 < t \leq 0.1,\\
&u(x,0) = x^2 - x,\\
&\frac{\partial u}{\partial x}(0,t) + u(0,t) = -1\\
&\frac{\partial u}{\partial x}(1,t) = 1.
\end{aligned}$$

Библиография

[1] J. Butcher, Numerical Methods for Ordinary Differential Equations. Wiley, 2nd edition, 2009.

[2] E. Coddington, An Introduction to Ordinary Differential Equations. Dover Publications, Unabridged edition, 1989.

[3] С. Димова, Т. Черногорова, А. Йотова, Числени методи за диференциални уравнения. Университетско издателство "Св. Климент Охридски", София, 2010.

[4] J. Hale, Ordinary Differential Equations. Dover Publications, 2009.

[5] M. Hirsh, S. Smale, R. Devaney, Differential Equations, Dynamical Systems, and an Introduction to Chaos. Academic Press, 3rd edition, 2012.

[6] M.H. Holmes, Introduction to Numerical Methods in Differential Equations. Springer, 2007.

[7] J. Murray, Mathematical Biology I. An Introduction. Springer, 3rd edition, 2002.

[8] M. Tenenbaum, H. Pollard, Ordinary Differential Equations. Dover Publications, Revised ed., 1985.