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.

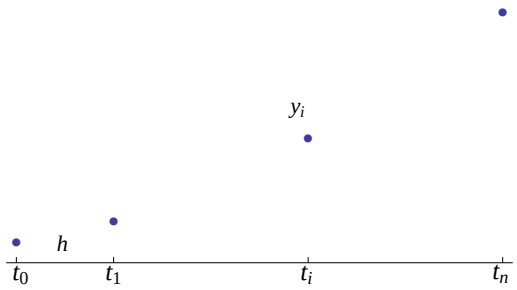
Да обърнем внимание, че променливите t и u са непрекъснати. Нашата цел ще бъде да ги заменим с дискретни, така че задачата да бъде сведена до алгебрична и следователно да може да бъде решена с помощта на стандартни числени методи. Основният въпрос е как тази „замяна" да се направи така, че да получим 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 7. Като използвате явния метод на Ойлер, решете моделното уравнение

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

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
          )
```

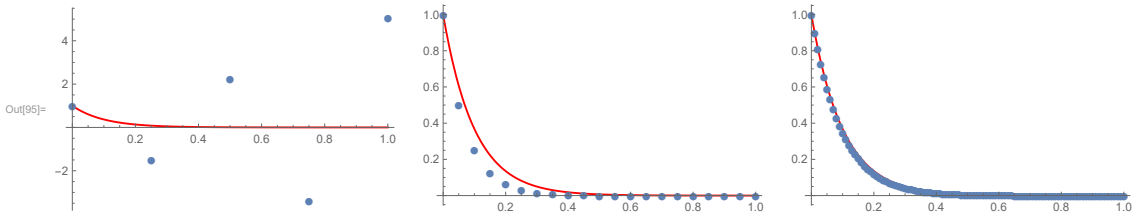
Като използваме реализираната функция, ще изобразим решенията за различни стойности на n.

```
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}]
```

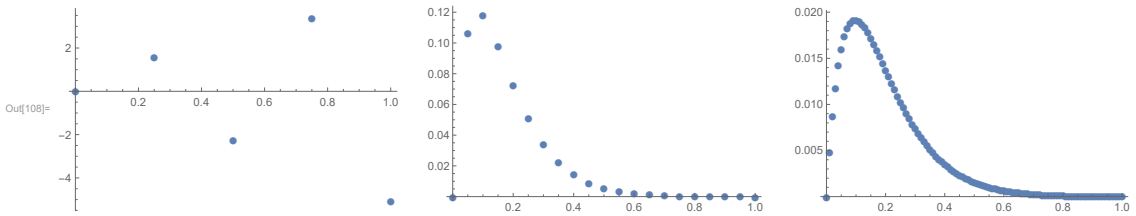
Решенията за $n = 4, 10, 100$ са показани по-долу, като червената крива изобразява точното решение.

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 8. Да се реши задачата на Коши от задача 7, 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:

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

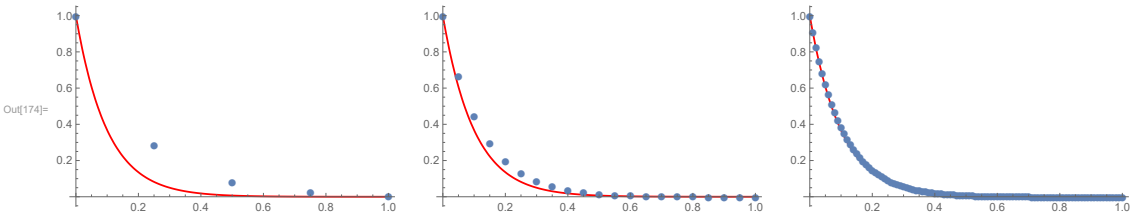
Имплементираме числената схема в Mathematica.

```
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

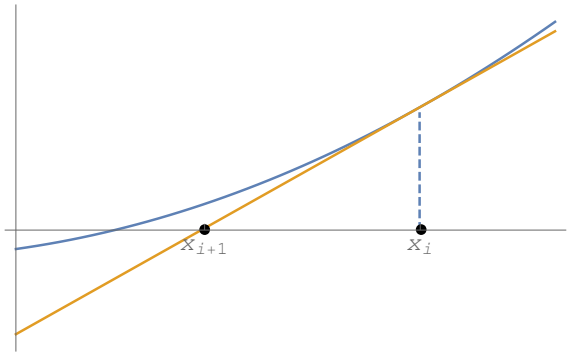
$$g'(x_0) = \operatorname{tg} \varphi = \frac{g(x_0) - 0}{x_0 - x_1}$$

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 7, 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.