



Řešení rekurentních problémů

Jitka Kreslíková, Aleš Smrčka

2023

Fakulta informačních technologií
Vysoké učení technické v Brně

IZP – Základy programování



Řešení rekurentních problémů

- ☐ Posloupnosti
- ☐ Řady



Rekurentní vztah

$Y_{i+1} = F(Y_{i-k}, \dots, Y_{i-2}, Y_{i-1}, Y_i)$, kde:

$Y_{i-k}, Y_{i-k+1}, \dots, Y_i, Y_{i+1}$ – zevšeobecněné proměnné,

F – funkce na základě které získáme z hodnot

$Y_{i-k}, \dots, Y_{i-2}, Y_{i-1}, Y_i$ hodnotu Y_{i+1}

$i, k \geq 0$ – počet předchozích kroků řešení

Vlastnosti:

- pro výpočet další hodnoty potřebujeme pouze $k+1$ posledních hodnot.
- musí existovat takové n , že Y_n je požadovanou hodnotou, po jejímž získání iterační výpočet končí.
- musíme umět rozpoznat požadovanou hodnotu.



Posloupnosti

- Uvažujme rekurentní vztah:

$$Y_{i+1} = F(Y_i)$$

- pro výpočet Y_{i+1} je potřeba zjistit hodnotu Y_i .
- na začátku musí být dané Y_0 , ze kterého celý výpočet začíná.
- postupně dostáváme hodnoty Y_1, Y_2, \dots, Y_n , pro které platí:
 1. $Y_{i+1} = F(Y_i)$ pro $i \geq 0$
 2. $Y_i \neq Y_j$ pro $i \neq j$
 3. Y_i pro $i < n$ – nesplňuje podmínky požadované hodnoty.
 4. Y_n – splňuje podmínky požadované hodnoty.



Algoritmické schéma pro posloupnosti

- Algoritmus realizující vztah $Y_{i+1}=F(Y_i)$:
 1. $Y=y_0$;
 2. while ($\neg B(Y)$) $Y=F(Y)$;
- Všeobecné symboly:
 - Proměnná Y
 - Predikátový symbol B
 - Funkční symbol F
- Predikát $B(Y)$ – podmínka požadované hodnoty závislá na hodnotě Y
- Nejde o řešení konkrétního rekurentního vztahu



Algoritmické schéma

- ❑ Algoritmická konstrukce, ve které symboly proměnných, funkcí a predikátů nejsou interpretovány.
- ❑ Pro konkrétní rekurentní vztah uvedeného charakteru stačí interpretace příslušných symbolů.
- ❑ Rekurentní vztahy → iterační výpočty → algoritmické schéma lze použít pro řešení celé řady problémů.



Výpočet druhé odmocniny

Příklad: Výpočet druhé odmocniny reálného čísla $A \geq 0$ lze popsat rekurentním vztahem:

$$y_{i+1} = \frac{1}{2} \left(\frac{A}{y_i} + y_i \right)$$

- y_0 – pro jednoduchost lze volit 1.
- Způsob ukončení algoritmu:
 - Obvykle se výpočet opakuje, dokud $|y_i - y_{i+1}|$ není menší než nějaká zadaná hodnota.
 - Této hodnotě se říká přesnost výpočtu – značíme ji **eps**.
- Pro výpočet nové hodnoty y potřebujeme jednu předcházející hodnotu – použijeme dvě proměnné: `stareY`, `noveY`.



Výpočet druhé odmocniny

Algoritmus:

1. zadej A , ϵ
2. inicializuj $stareY$ (1)
3. vypočítej $noveY$ (podle vzorce)
4. opakuj pokud $abs(noveY - stareY) \geq \epsilon$
 - {
ulož $noveY$ do $stareY$
vypočítej $noveY$
}
5. zobraz výsledek



Řady

- Uvažujme řadu vytvořenou z členů:

$$t_0, t_1, t_2, \dots$$

- Pro členy řady lze napsat rekurentní vztah:

$$t_i = f(t_{i-1}), \text{ pro } i > 0$$

- Necht' částečné součty jsou:

$$s_0, s_1, s_2, \dots$$

$$s_i = t_0 + t_1 + \dots + t_i$$

- Pro částečné součty platí rekurentní vztah:

$$s_0 = t_0$$

$$s_i = s_{i-1} + t_i$$



Řady

- ❑ Při konstrukci algoritmu nutno zohlednit
 - rekurentní vztah pro částečné součty
 - rekurentní vztah pro členy řady, jejich vzájemný vztah
 - způsob ukončení
- ❑ Použití řad pro aproximaci funkcí
 - Počet členů řad není dopředu znám
 - Konec algoritmu \Leftrightarrow přesnost aproximace
 - Přesnost dána buď dosaženou hodnotou částečné sumy nebo posledním sčítancem



Algoritmické schéma pro řady

□ Na základě uvedené analýzy lze sestavit modifikované algoritmické schéma pro řady:

1. $T = t_0;$

2. $S = T;$

3. while ($\neg B(S, T)$)
 {
 $T = f(T);$
 $S = S + T;$
 }

4. zobraz výsledek



Algoritmické schéma pro řady

□ Analogicky jako pro posloupnosti tak i pro řady musí platit tvrzení:

1. $t_i = f(t_{i-1})$ pro $i > 0$
2. $t_i \neq t_j$ pro $i \neq j$
3. $s_i = s_{i-1} + t_i$ pro $i > 0$
4. $\neg B(s_i, t_i)$ pro $i < n$
5. $B(s_n, t_n)$ změnění se hodnota predikátu B



Algoritmické schéma pro řady

- ❑ Při použití algoritmického schématu pro řady je třeba si uvědomit, že
 - musí existovat **n** , pro které se změní hodnota predikátu **$B(s_i, t_i)$** .
 - při aproximaci funkce je třeba podmínku volit uváženě s ohledem na **funkci**, **argument** a **řadu** → konvergence řady může být velmi pomalá.
 - požadované přesnosti aproximace nemusí být vždy dosažitelné, anebo její cena může být velmi velká.



Aproximace e^x

Příklad: Aproximace $y = e^x$

Exponenciální funkci e^x lze aproximovat řadou:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$$

Pro částečný součet s_i lze napsat:

$$s_i = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^i}{i!}$$



Aproximace e^x

- Závislost sousedních členů řady lze vyjádřit rekurentním vztahem:

$$t_j = t_{j-1} \frac{x}{j} \quad \text{pro } j > 0$$

- Řada konverguje pro $\mathcal{R} \rightarrow$ přírůstek jednotlivých členů řady do celkové sumy bude od určitého i klesat \rightarrow lze využít pro tvorbu koncové podmínky
- Cyklus se bude opakovat, dokud hodnota přírůstku, tedy hodnota členu řady, neklesne pod danou hranici, kterou označíme **eps**.



Aproximace e^x

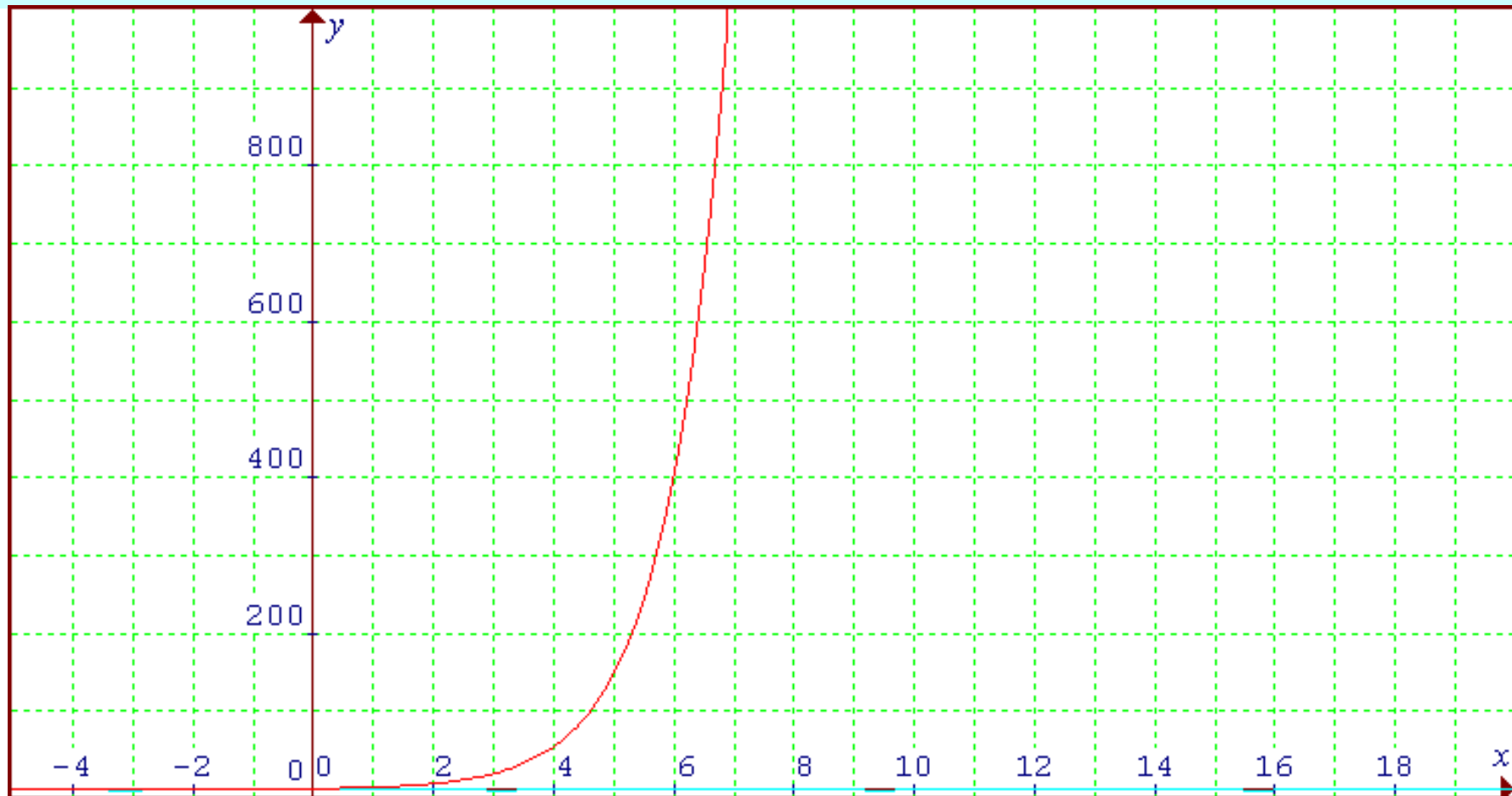
Algoritmus:

1. zadání x , ϵ
2. inicializace: $t=1$, $\text{soucetRady}=t$, $i=0$
3. opakuj pokud $\text{abs}(t) \geq \epsilon$
 - {
 - inkrementace i
 - výpočet dalšího členu t
 - $\text{soucetRady} = \text{soucetRady} + t$
 - }
4. zobrazení výsledku



Aproximace e^x

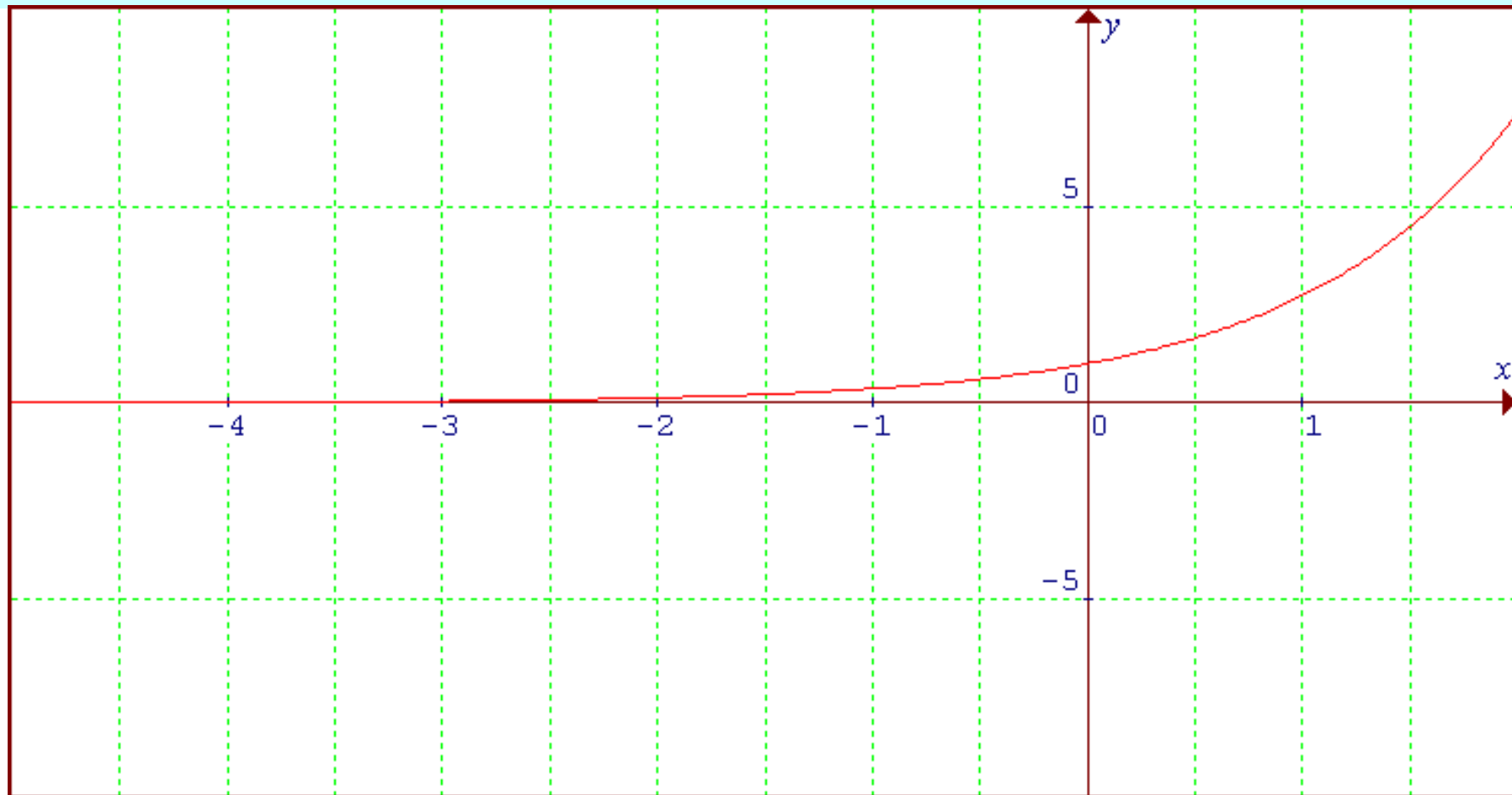
Příklad: $y = e^x, (-\infty < x < \infty)$





Aproximace e^x

Příklad: $y = e^x, (-\infty < x < \infty)$





Aproximace e^x

- Rychlost konvergence
 - není stejná pro všechna reálná čísla.
 - je velká pro malé hodnoty argumentu x (okolo nuly).
- Pro velké hodnoty x se doporučuje rozdělit argument **x** na celou část **c** a desetinnou část **d** a použít vztah:

$$e^{c+d} = e^c \cdot e^d$$

Hodnota e^c se vypočítá opakovaným násobením e .



Aproximace $\sin(x)$

Příklad: Aproximace $y = \sin(x)$

Pro částečný součet s_i lze napsat:

$$s_i = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots (-1)^i \frac{x^{2i+1}}{(2i+1)!}$$

Členy řady:

$$t_j = -t_{j-1} \frac{x^2}{k_j(k_j - 1)}, \quad \text{pro } j > 0$$

$$k_j = k_{j-1} + 2$$

počáteční hodnoty: $t_0 = x$, $k_0 = 1$

pro $x \in \langle 0; \pi/4 \rangle$ - konverguje nejrychleji (ověřte)



Aproximace $\sin(x)$

- ❑ **Podmínka ukončení** nespecifikuje absolutní hodnotu posledního členu, ale určuje relativně jeho velikost vzhledem k celkové sumě.

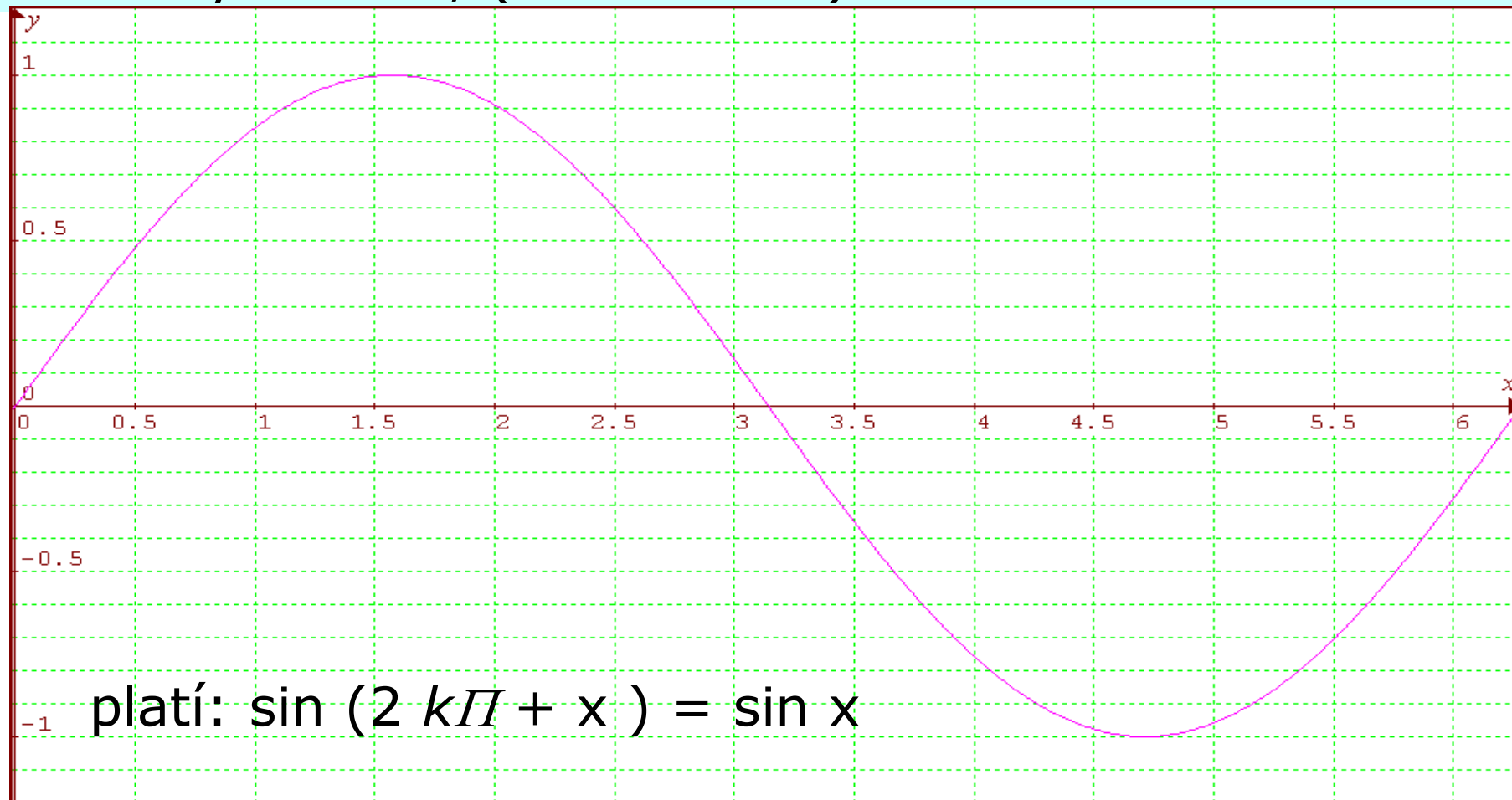
Algoritmus:

1. zadání x , ϵ
2. inicializace $t=x$, $\text{soucetRady}=t$, $k=1$
3. opakuj pokud $\text{abs}(t) \geq \epsilon \times \text{abs}(\text{soucetRady})$
 - {
 - $k = k + 2$
 - výpočet dalšího členu t
 - $\text{soucetRady} = \text{soucetRady} + t$
 - }
4. zobrazení výsledku



Řešení rekurentních problémů

Příklad: $y = \sin x, (-\infty < x < \infty)$





Heuristika

- ❑ Opatření:
 - k snížení náročnosti výpočtu
 - k zvýšení efektivity výpočtu
- ❑ Posun výpočtu do intervalu nejrychlejší konvergence
 - $e^{c+d} = e^c e^d$
 - využití periodicity u goniometrických funkcí
- ❑ Odstranění zbytečných výpočtů
 - Zejména z těla cyklu – odsunout mimo!



Řešení rekurentních problémů





Kontrolní otázky

1. Jak je definován rekurentní vztah?
2. Vysvětlete postup řešení problémů, které jsou definované rekurentním vztahem.
3. Jakým způsobem se řeší přesnost výpočtu u problémů zadaných rekurentním vztahem?



Úkoly k procvičení

1. Vytvořte program v jazyku C se standardními knihovnamí, který pomocí iteračních výpočtů vypočítá funkce o neznámém počtu hodnot na standardním vstupu a vypíše výsledky na standardní výstup. Každá hodnota bude vypočtena zvlášť ve vlastní funkci. Algoritmy musí řešit heuristiku a práci s nekonečnými a nečíselnými hodnotami podobně jako to řeší knihovna `<math.h>`.