Metoda Runge-Kutta

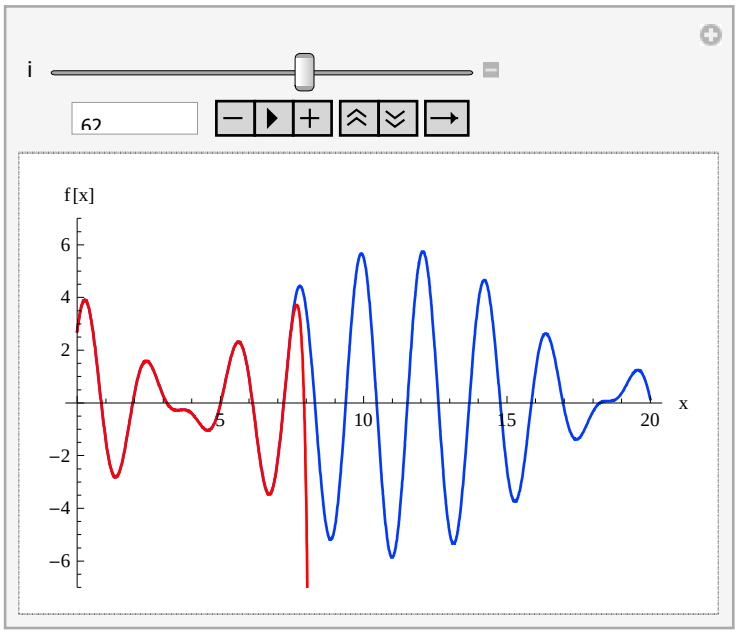
In[1]:=
```
ClearAll["Global`*"];
```

Taylorův rozvoj

... U mnoha funkcí platí, že více členů Taylorova rozvoje poskytne lepší přesnost náhrady funkce. Ukázka zpřesňování náhrady je v notebooku **CAO-RungeKutta.nb.** ...

In[2]:=
```
f[x_] := π * Sin[e * x] + e * Cos[π * x];
fTaylor[n_Integer] := N[Normal[Series[f[x], {x, 0, n}]]];
nmax = 100;
rozvoje = fTaylor /@ Range[nmax];
pl[i_Integer] := Plot[{f[x], rozvoje[[i]]}, {x, 0, 20}, AxesLabel → {"x", " f[x]"},
   PlotStyle → {{Hue[0.63], Thickness[0.0045]}, {Hue[0.], Thickness[0.0045]}},
   PlotRange → {Automatic, {-7, 7}}];
Manipulate[pl[i], {i, 1, nmax, 1}, ControlPlacement → Top]
```

Out[7]=



Odvození metody Runge-Kutta

... V notebooku **CAO-RungeKutta.nb** je ukázán postup, jak odvodit iterační krok Eulerovy metody využívající první tři členy Taylorova rozvoje. Použijeme ale trik: budeme hledat vyjádření tohoto rozvoje tak, abychom nemuseli vyčíslovat vyšší derivace než $y'(t) = f(t, y(t))$.
Hledejme náhradu ve tvaru:

$$y(t0 + \Delta t) \approx y(t0) + v1 \cdot p1 + v2 \cdot p2,$$

kde $v1$ a $v2$ jsou váhy přírůstku $p1$ a $p2$:

$$p1 = \Delta t \cdot f(t0, y(t0)),$$
$$p2 = \Delta t \cdot f(t0 + a \cdot \Delta t, y(t0) + b \cdot p1)$$

In[8]:=
```
p1 = Δt * f[t0, y[t0]];
p2 = Δt * f[t0 + a * Δt, y[t0] + b * p1];
yDalRunge = y[t0] + v1 * p1 + v2 * p2;
yDalTaylorPom = (Normal[Series[y[t], {t, t0, 2}]] /. t → Δt + t0);
dosders = D[y'[t0] -> f[t0, y[t0]], {t0, #}] & /@ Range[0, 2];
yDalTaylor = yDalTaylorPom //. dosders;
Normal[Series[yDalRunge, {Δt, 0, 2}]];
rce = Coefficient[% - yDalTaylor, Δt^#] == 0 & /@ Range[1, 2];
```

... V notebooku **CAO-RungeKutta.nb** je ukázán numerický postup vyřešení rovnosti koeficientů: Za hodnoty funkce ***f*** a jejích derivací jsou dosazována náhodná čísla, vytvořeny rozdíly pravých a levých stran a sečteny přes všechny rovnice a hledáno minimum výsledného výrazu. Je-li blízké nule, jsou rovnice vyřešeny dobře. Je jedno, jak k výsledku dospějeme, tohle se autorům zdálo rychlejší naprogramovat. ...

In[16]:=
```
trya = 1;
SeedRandom[1];
tryn = 10;
numRce = Flatten[rce /. {f^(_,_)[_, _] -> Random[], f[_, _] -> Random[]} & /@ Range[tryn]];
norm := Abs[#]^2 &;
promin = Expand[Plus @@ (numRce /. lhs_ == rhs_ ↦ norm[lhs - rhs])] /. a → trya;
proms = {#, Random[]} & /@ Union[Cases[promin, _Symbol, {0, ∞}]];
dos = Union[(min = FindMinimum[promin, proms])[[2]], {a → trya}];
Print["Pro hodnoty neznamych ", dos]
Print["je soucet normovanych odchylek splneni rovnic roven ", min[[1]] ]
```

```
Pro hodnoty neznamych {a → 1, b → 1., v1 → 0.5, v2 → 0.5}
```

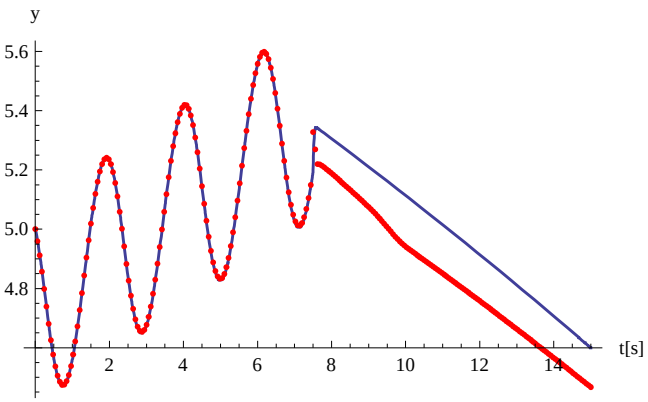
```
je soucet normovanych odchylek splneni rovnic roven 1.17616 × 10^-18
```

Porovnání odvozené metody Runge-Kutta s metodou NDSolve

... V další části notebooku **CAO-RungeKutta.nb** je ukázáno porovnání výsledků odvozené metody s NDSolve. Doporučujeme pohrát si s různými hodnotami počtu kroků **n**, případně měnit zadanou funkci *f* (v notebooku se jmenuje **myf**). Všimněte si zejména, jak „zlobí“ skok v derivaci (proto je **myf** zadána pomocí **If**). Tam, kde hrozí nespojitost *f*, hrozí nespolehlivost numerických rešení diferenciálních rovnic. Nejen tam ... Naštestí v oblasti elektrických obvodů nám příroda takové funkce *f* v námi zvoleném spojitém popisu světa zakazuje. ...

In[26]:=
```
tStart = 0.;
tEnd = 15;
yStart = 5;
myf[t_, y_] := If[t < 0.5 * (tStart + tEnd), Sin[3 t - .5 y], -y * Cos[t + y^2]];
n = 250;
Δt = (tEnd - tStart)/n;
dal[{t_, y_}] = {t + Δt, ((yDalRunge /. dos) /. f :> myf) /. {t0 → t, y[t0] → y}};
res = NestList[dal, {tStart, yStart}, n];
resNDS = NDSolve[{y'[t] == myf[t, y[t]], y[tStart] == yStart}, y, {t, tStart, tEnd}][[1]];
Show[Plot[y[t] /. resNDS, {t, tStart, tEnd}, PlotRange → All, PlotStyle → Thickness[0.005],
  AxesLabel → {"t[s]", "y"}], ListPlot[res, PlotStyle → {Red, PointSize[0.01]}]]
```

Out[35]=

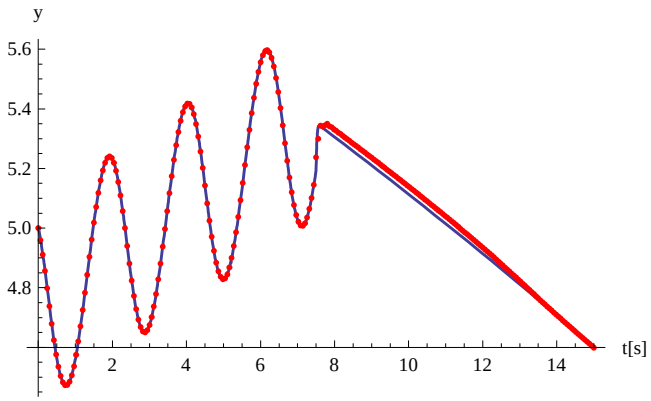


Použití oblíbené metody Runge-Kutta 4. řádu

... V předposlední části je ukázáno použití oblíbené metody Runge-Kutta čtvrtého řádu (shodovalo by se pět členů Taylorova rozvoje, tedy až do čtvrté mocniny *t* včetne). ...

In[36]:=
```
derivace[t_, y_] := myf[t, y];
RungeKuttaKrok[{t_, y_}] := Module[
   {k1, k2, k3, k4},
   k1 = derivace[t, y];
   k2 = derivace[t + Δt/2, y + Δt/2 * k1];
   k3 = derivace[t + Δt/2, y + Δt/2 * k2];
   k4 = derivace[t + Δt, y + Δt * k3];
   {t + Δt, y + Δt/6 * (k1 + 2 k2 + 2 k3 + k4)}
  ];
res2 = NestList[RungeKuttaKrok, {tStart, yStart}, n];
Show[Plot[y[t] /. resNDS, {t, tStart, tEnd}, PlotRange → All,
  PlotStyle → Thickness[0.005], AxesLabel → {"t[s]", "y"}],
 ListPlot[res2, PlotStyle → {Red, PointSize[0.01]}]]
```

Out[39]=



Grafická demonstrace metody Runge-Kutta 4. řádu

Zkoumaná funkce bude definovaná svojí derivací a počátečním bodem.

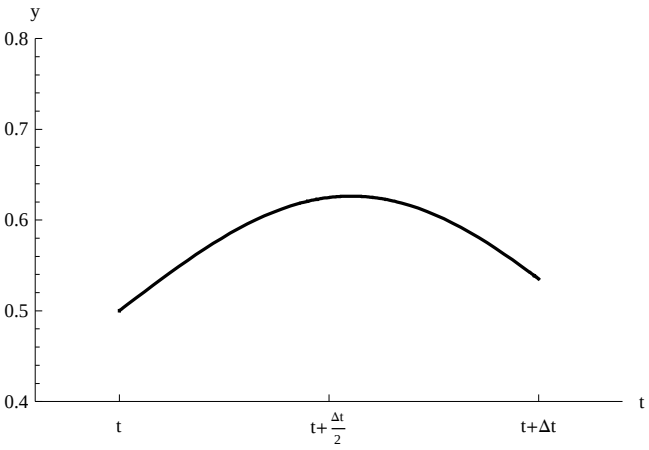
In[40]:=
```
(*der[{t_,y_}]:=-Sin[t^2+y^3-6];*)
der[{t_, y_}] := -Sin[t^3 + y + 0.8];

tStart = 1.5;
yStart = 0.5;
Δt = 0.35;
```

Nejprve si zobrazme průběh zkoumané funkce "přesně" spočítaný pomocí metody **NDSolve** (každopádně spočítaný s krokem mnohem menším nežli Δt):

In[44]:=
```
rNDS =
  NDSolve[{y'[t] == der[{t, y[t]}], y[tStart] == yStart}, y, {t, tStart, tStart + Δt}][[1]];
plNDS = Plot[y[t] /. rNDS, {t, tStart, tStart + Δt},
  Ticks → {{{tStart, "t"}, {tStart + Δt/2, "t+Δt/2"}, {tStart + Δt, "t+Δt"}}},
  AxesOrigin → {tStart - 0.2 * Δt, 0.4},
  PlotRange → {{tStart - 0.2 * Δt, tStart + 1.2 * Δt}, {0.4, 0.8}},
  AxesLabel → {"t", "y"}, PlotStyle → {Thickness[0.0055], Black}]
```

Out[45]=

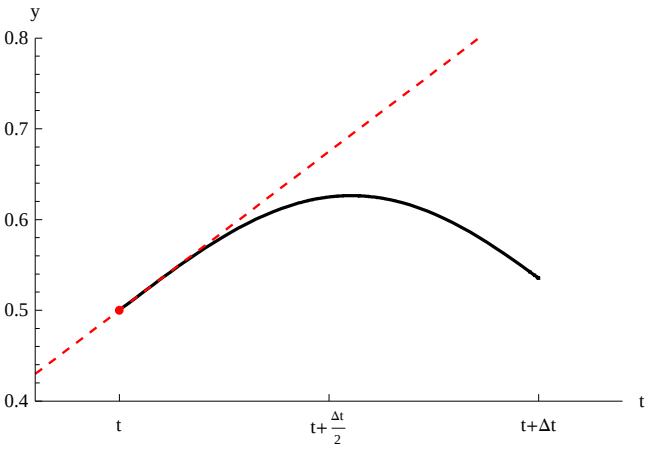


Tento průběh budeme aproximovat metodou Runge-Kutta. Z bodu t do bodu $t+\Delta t$ skočíme na jeden krok Δt.

Spočítáme derivaci $k1$ v počátku (červený bod), tedy v bodě $[t, y(t)]$. Derivace $k1$ v počátku má směr podle červené čárkované úsečky.

In[46]:=
```
bodStart = {tStart, yStart};
znackaBodStart = Graphics[{Red, PointSize[0.015], Point[bodStart]}];
k1 = der[bodStart];
plk1 = Plot[yStart + (t - tStart) * k1,
   {t, tStart - 0.2 Δt, tStart + Δt}, PlotStyle → {Red, Dashed, Thickness[0.004]}];
Show[plNDS, znackaBodStart, plk1]
```

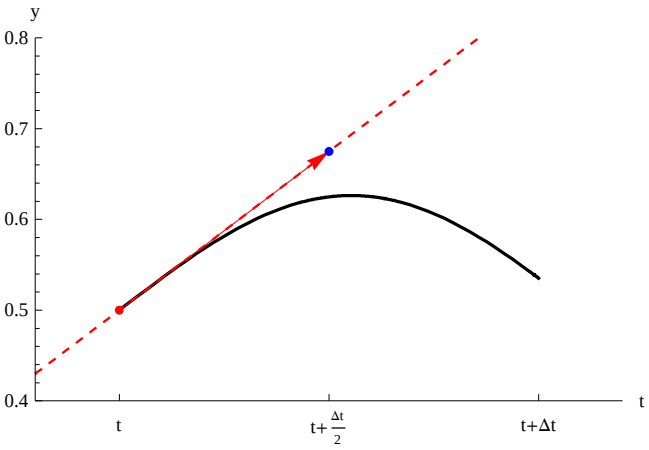
Out[50]=



Vyrazíme z počátku (červený bod) ve směru derivace $k1$ (červená šipka).
Popojdeme o $\frac{\Delta t}{2}$, tedy dojdeme do bodu $[t+\frac{\Delta t}{2}, y(t) + k1\ \frac{\Delta t}{2}]$ (modrý bod).

In[51]:=
```
bodProk2 = {tStart + Δt/2, yStart + k1 * Δt/2};
plk1krok = Graphics[{Red, Arrow[{bodStart, bodProk2}]}];
znackaBodProk2 = Graphics[{Blue, PointSize[0.015], Point[bodProk2]}];
Show[plNDS, znackaBodStart, plk1, plk1krok, znackaBodProk2]
```

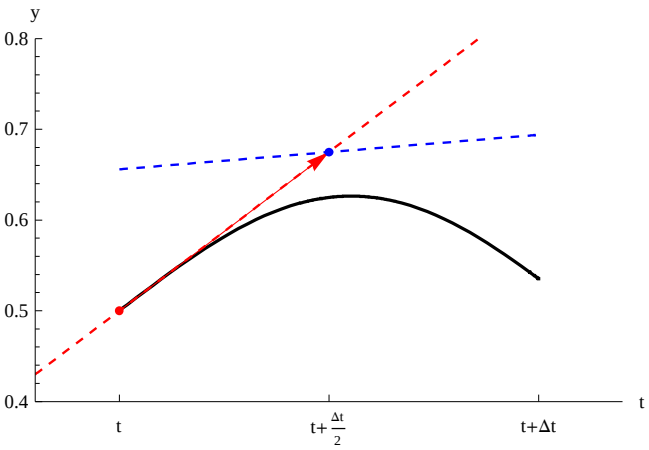
Out[54]=



V modrém bodě spočítáme novou derivaci $k2$. Derivace $k2$ v modrém bodě má směr podle modré čárkované úsečky.

In[55]:=
```
k2 = der[bodProk2];
plk2 = Plot[bodProk2[[2]] + (t - tStart - Δt/2) * k2,
   {t, tStart, tStart + Δt}, PlotStyle → {Blue, Dashed, Thickness[0.004]}];
Show[plNDS, znackaBodStart, plk1, plk1krok, znackaBodProk2, plk2]
```

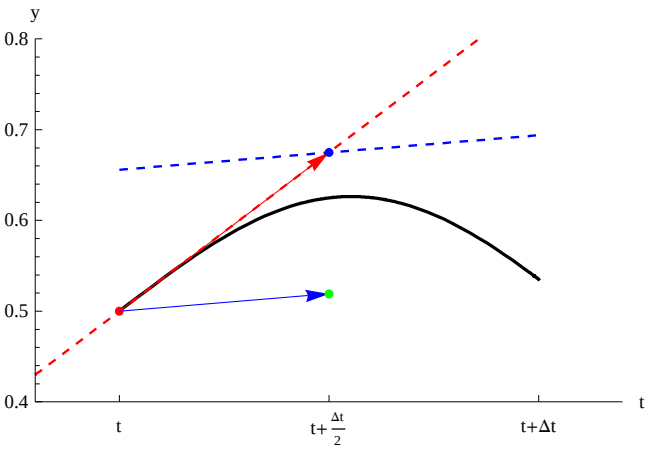
Out[57]=



Opět vyrazíme z počátku, ale tentokrát ve směru derivace $k2$, tedy po rovnoběžce s modrou úsečkou (modrá šipka). Opět popojdeme o $\frac{\Delta t}{2}$, tedy dojdeme do bodu $[t+\frac{\Delta t}{2}, y(t) + k2\ \frac{\Delta t}{2}]$ (zelený bod).

In[58]:=
```
bodProk3 = {tStart + Δt/2, yStart + k2 * Δt/2};
plk2krok = Graphics[{Blue, Arrow[{bodStart, bodProk3}]}];
znackaBodProk3 = Graphics[{Green, PointSize[0.015], Point[bodProk3]}];
Show[plNDS, znackaBodStart, plk1, plk1krok,
 znackaBodProk2, plk2, plk2krok, znackaBodProk3]
```

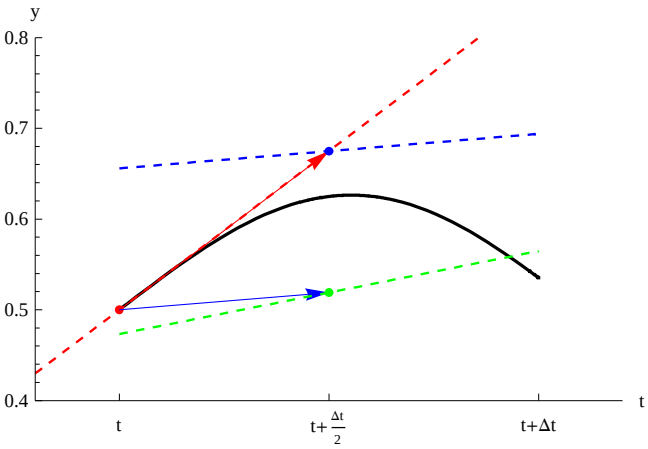
Out[61]=



V zeleném bodě spočítáme novou derivaci *k*3. Derivace *k*3 v zeleném bodě má směr podle zelené čárkované úsečky.

In[62]:=
```
k3 = der[bodProk3];
plk3 = Plot[bodProk3[[2]] + (t - tStart - Δt/2) * k3,
   {t, tStart, tStart + Δt}, PlotStyle → {Green, Dashed, Thickness[0.004]}];
Show[plNDS, znackaBodStart, plk1, plk1krok, znackaBodProk2,
 plk2, plk2krok, znackaBodProk3, plk3]
```

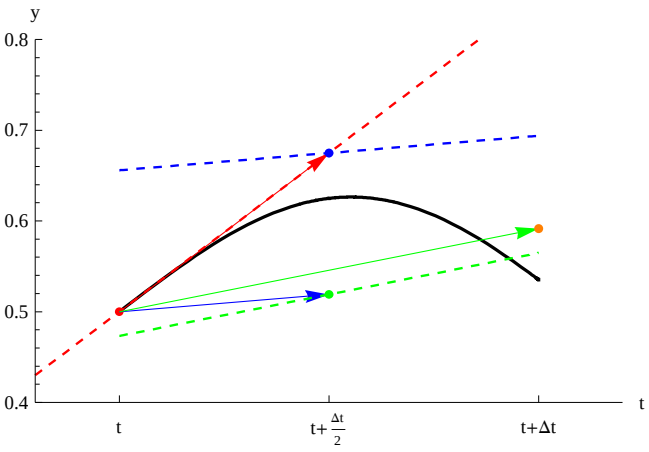
Out[64]=



Opět vyrazíme z počátku, ale tentokrát ve směru derivace *k*3, tedy po rovnoběžce se zelenou úsečkou (zelená šipka). Tentokrát však popojdeme o Δt, tedy dojdeme do bodu $[t+\frac{\Delta t}{2}, y(t) + k3\ \Delta t]$ (oranžový bod).

In[65]:=
```
bodProk4 = {tStart + Δt, yStart + k3 * Δt};
plk3krok = Graphics[{Green, Arrow[{bodStart, bodProk4}]}];
znackaBodProk4 = Graphics[{Orange, PointSize[0.015], Point[bodProk4]}];
Show[plNDS, znackaBodStart, plk1, plk1krok, znackaBodProk2,
 plk2, plk2krok, znackaBodProk3, plk3, plk3krok, znackaBodProk4]
```

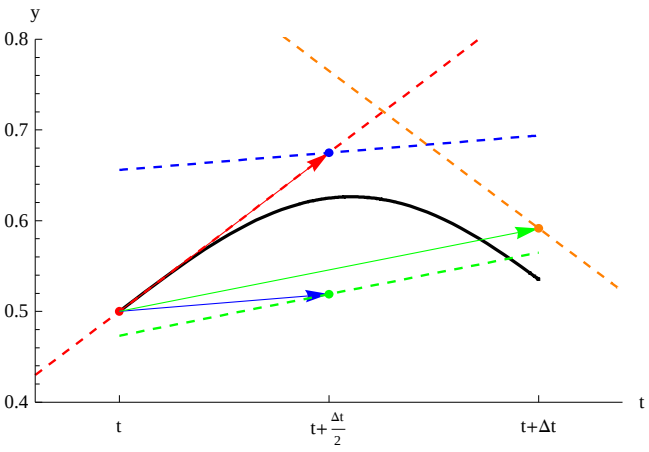
Out[68]=



V oranžovém bodě spočítáme novou derivaci *k*4. Derivace *k*4 v oranžovém bodě má směr podle oranžové čárkované úsečky.

In[69]:=
```
k4 = der[bodProk4];
plk4 = Plot[bodProk4[[2]] + (t - tStart - Δt) * k4,
    {t, tStart, tStart + 1.2 Δt}, PlotStyle → {Orange, Dashed, Thickness[0.004]}];
Show[plNDS, znackaBodStart, plk1, plk1krok, znackaBodProk2, plk2,
 plk2krok, znackaBodProk3, plk3, plk3krok, znackaBodProk4, plk4]
```

Out[71]=



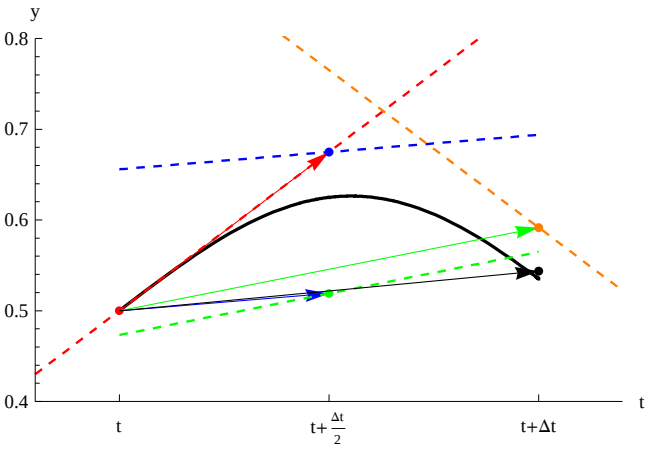
Výslednou derivaci *kVys* spočítáme jako vážený průměr derivací *k*1, *k*2, *k*3 a *k*4.

In[72]:=
```
kVys = 1/6 * (k1 + 2 k2 + 2 k3 + k4);
```

Podle této derivace vyrazíme z počátku, tedy ve směru černé šipky. Popojdeme o Δ*t* a dorazíme do černého bodu.

In[73]:=
```
bodVys = {tStart + Δt, yStart + kVys * Δt};
znackaBodVys = Graphics[{Black, PointSize[0.015], Point[bodVys]}];
plkVyskrok = Graphics[{Black, Arrow[{bodStart, bodVys}]}];
Show[plNDS, znackaBodStart, plk1, plk1krok, znackaBodProk2, plk2, plk2krok,
 znackaBodProk3, plk3, plk3krok, znackaBodProk4, plk4, plkVyskrok, znackaBodVys]
```

Out[76]=



Přestože jsme zvolili poměrně velký krok Δt, spočítali jsme hodnotu funkce v bodě $t+\Delta t$ poměrně přesně.

Jednotlivé kroky si můžete projít ještě jednou postupným přepínáním tlačítek nad grafem níže.

In[77]:=
```
grafy = {plNDS, znackaBodStart, plk1, plk1krok, znackaBodProk2, plk2, plk2krok,
    znackaBodProk3, plk3, plk3krok, znackaBodProk4, plk4, plkVyskrok, znackaBodVys};
```

In[78]:=
```
kroky = Table[Take[grafy, i], {i, 1, 14}];
```

In[86]:=
```
Manipulate[Show[kroky[[krok]]], {krok, Range[14], Setter}]
```

Out[86]=