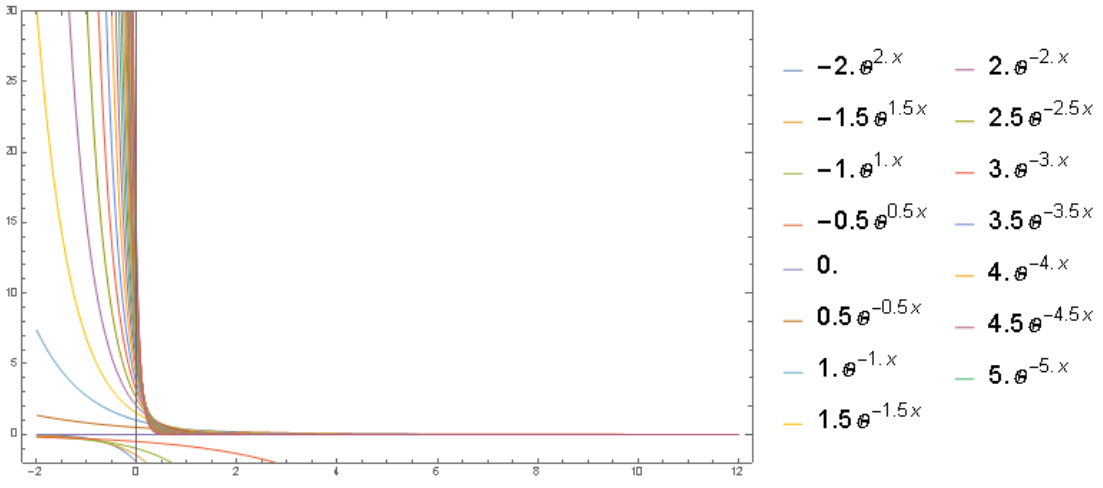
Differentialgleichungen analysieren :

Bei gewünschter Berechnung das rote Semikolon ; am Ende entfernen (falls vorhanden) ! Mit 'Evaluate Cell' oder 'Shift' + 'Enter' starten.

Allgemeine Differentialgleichung

```
(* Hier Gleichung ohne Randbedingungen eingeben *)
DGL := y'[x] == -k * y[x];
(* Lösung als Formel ausgeben *)
Solution = DSolve[{DGL}, y[x], x]
(* Plotten, wahlweise als Tabelle ausgeben *)
ScharUntergrenzeX  = -2;
ScharObergrenzeX   = +12;
ScharSchrittweite  = .5;
ScharUntergrenzeY  = -2;
ScharObergrenzeY   = +30;
(* Tabelle erstellen *)
Loesungsschar = Table[ y[x] /. Solution /. C[1] -> k, {k, ScharUntergrenzeX,
      ScharObergrenzeX, ScharSchrittweite}];
(* Tabelle evtl. ausgeben *)
TabelleSchar = Grid[%, Frame -> All, ItemSize -> All];
(* Funktionsschar plotten *)
GraphSchar = Plot[Loesungsschar // Evaluate, {x, ScharUntergrenzeX, ScharObergrenzeX},
          PlotRange -> {ScharUntergrenzeY, ScharObergrenzeY},
        Frame -> True, PlotLegends → "Expressions", ImageSize → 700]
```

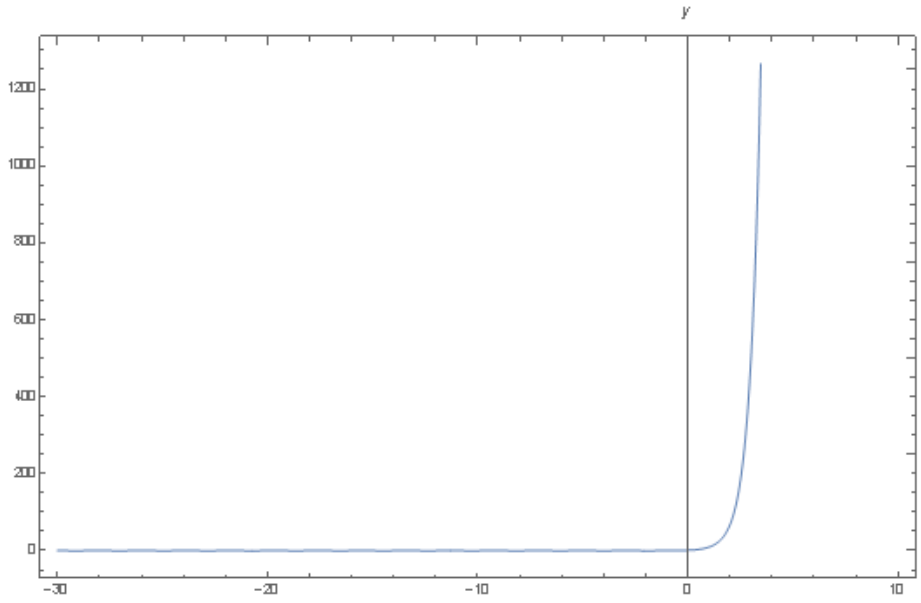
$$\{\{y[x] \to e^{-k x} C[1]\}\}$$



Differentialgleichung mit Randbedingungen

```
(* Hier Gleichung und Randbedingungen eingeben *)
EquationPart = {y'[x] == 2 * y[x] + Sin[3 x], y[0] == 1};
SolutionPart = DSolve [EquationPart, y, x ]
(* Hier wird die Lösung übergeben *)
fpart[x_] := y[x] /. SolutionPart[[1]];
(* Evtl. Probe durchführen. Bei "True" ist alles OK *)
Expand[EquationPart /. SolutionPart];
(* Ausgabe von Tabelle und/oder Graph *)
xmin = -30; xmax = +10; TabSchrittweite = 5;
TabellePart = Grid[Table[{x, fpart[x], fpart[x] // N}, {x, xmin, xmax, TabSchrittweite}],
        Frame -> All, ItemSize -> All];
GraphPart = Plot[ fpart[x], {x, xmin, xmax}, AxesLabel -> {x, y}, Frame -> True,
      ImageSize → 700]
```

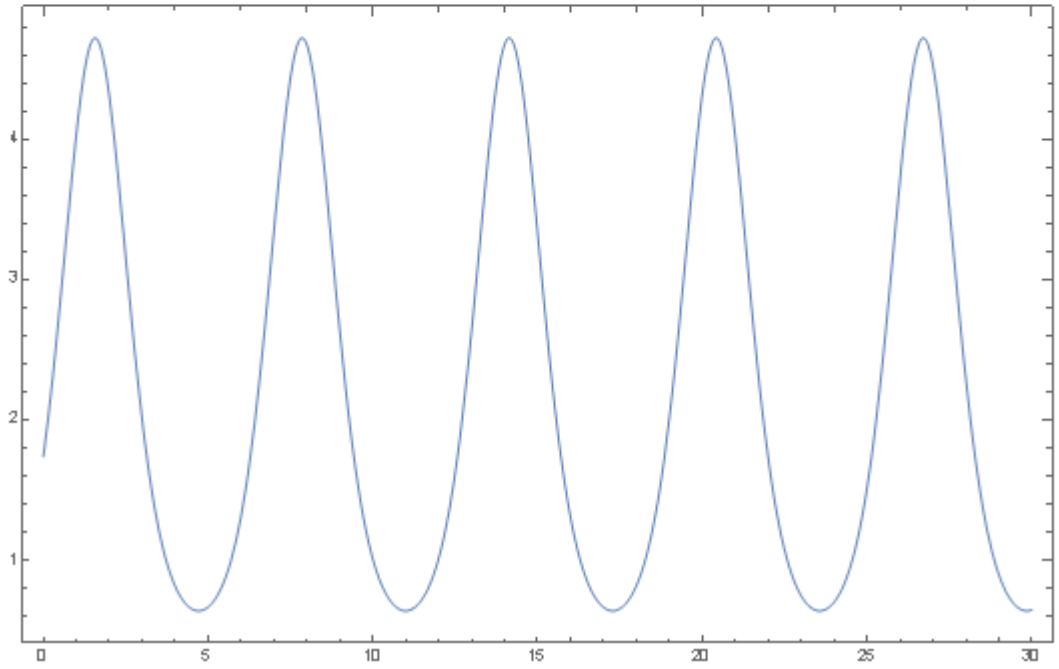
$$\left\{\left\{y \to \text{Function}\left[\{x\}, \frac{1}{13}\left(16\, e^{2x} - 3\cos[3x] - 2\sin[3x]\right)\right]\right\}\right\}$$



Differentialgleichung numerisch lösen

```
(* Hier Gleichung und Randbedingungen eingeben *)
numericalSolution = NDSolve[{y'[x] == y[x] * Cos[x], y[3] == 2}, y, {x, 0, 30}]
(* Ausgabe des Graphen *)
Plot[Evaluate[y[x] /. numericalSolution], {x, 0, 30}, PlotRange → All,
  Frame -> True, ImageSize → 700]
```

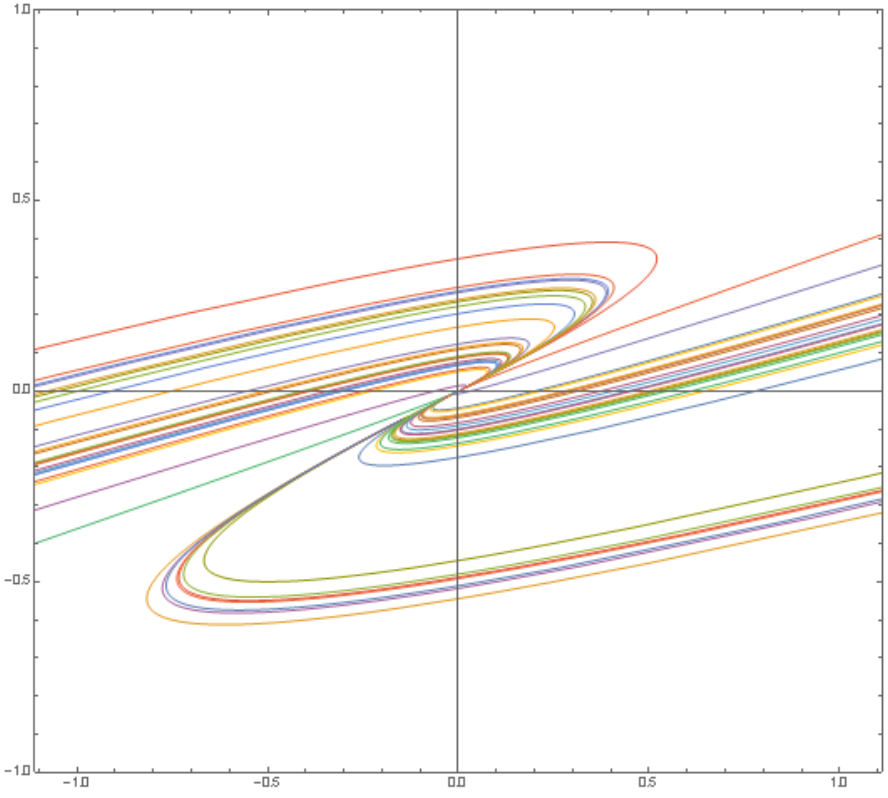
$$\{\{y \to \text{InterpolatingFunction}[\langle\langle \text{Domain: } \{\{0., 30.\}\} \text{ Output: scalar} \rangle\rangle]\}\}$$



Systeme von Differentialgleichungen allgemein lösen

```
(* Dies ist die Beispielfunktion. Benötigt wird die Matrix der rechten Seiten *)
VorlageDGLn = {x'[t] == 4 * x[t] - 6 * y[t], y'[t] == 1 * x[t] - 1 * y[t]};
(* Matrix erstellen *)
MatrixSystemDGL = {{4, -6}, {1, -1}};
(* Berechne die Eigenwerte der Matrix *)
Eigenvalues[MatrixSystemDGL];
(* Gerüst für DGLn vorbereiten *)
X[t_] = {x[t], y[t]};
(* Gleichungssystem erstellen *)
GleichungssystemDGLn = X'[t] == MatrixSystemDGL.X[t];
(* Gleichungssystem lösen *)
LösungSystemDGLn = DSolve[GleichungssystemDGLn, {x, y}, t]
(* Ausgabe des Graphen  vorbereiten *)
PartikuläreLösung = Partition[Flatten[Table[{x[t], y[t]} /. LösungSystemDGLn /. {C[1] → 1 / i,
        C[2] → 1 / j}, {i, -20, 20, 6}, {j, -20, 20, 6}]], 2];
(* Ausgabe des Graphen  durchführen *)
ParametricPlot[Evaluate[PartikuläreLösung], {t, -3, 3}, PlotRange → {-1, 1},
 Frame -> True,  ImageSize → 700]
```

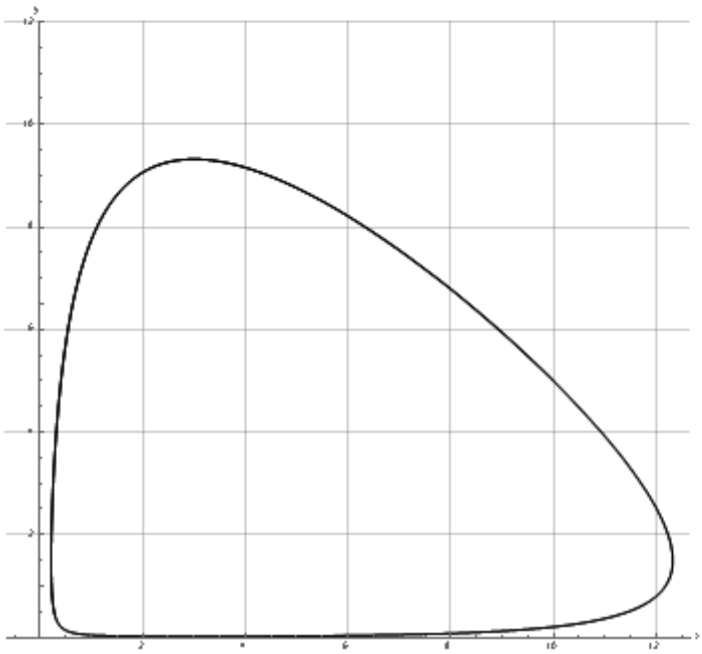
$$\left\{\left\{x \to \text{Function}\left[\{t\}, e^{t}\left(-2+3e^{t}\right)C[1]-6e^{t}\left(-1+e^{t}\right)C[2]\right],\right.\right.$$
$$\left.\left.y \to \text{Function}\left[\{t\}, e^{t}\left(-1+e^{t}\right)C[1]-e^{t}\left(-3+2e^{t}\right)C[2]\right]\right\}\right\}$$



Systeme von Differentialgleichungen mit Randbedingungen lösen

```
(* Hier Gleichung und Randbedingungen eingeben *)
SystemDGLn = {x'[t] == 7 * x[t] - 5 * y[t], y'[t] == 4 * x[t] - 2 * y[t], x[0] == -1 / 2, y[0] == 1};
(* DGLn lösen *)
LösungDGLn = DSolve[SystemDGLn, {x, y}, t]
(* Probe durchführen *)
SystemDGLn /. LösungDGLn /. {t → RandomReal[]};
(* Ausgabe des Graphen *)
Plot[{x[t] /. LösungDGLn, y[t] /. LösungDGLn}, {t, 8, 11}, PlotRange → All,
    Frame -> True, ImageSize → 700];
(* Lösung in parametrisierter Form anzeigen. Beispiel Lotka–Volterra–DGLn *)
a = 1.5; b = 1; c = 3; d = 1;
ParNumericalSolution = NDSolve[{x'[t] == x[t] * (a - b * y[t]),
        y'[t] == y[t] * (-c + d * x[t]), x[0] == 10, y[0] == 5}, {x, y}, {t, 0, 30}];
(* Einzelgleichungen isolieren *)
BeuteGlng[t_] = x[t] /. ParNumericalSolution[[1]];
RäuberGlng[t_] = y[t] /. ParNumericalSolution[[1]];
(* Ausgabe des Graphen *)
ParameterPlot = ParametricPlot[{BeuteGlng[t], RäuberGlng[t]}, {t, 0, 10},
    PlotRange → {0, 12}(* {-2, +2} *), PlotStyle → {Thick, Black},
    GridLines → Automatic, AxesLabel → {"x", "y"}, Frame -> None,
    PlotLegends → "Expressions", Frame → True, ImageSize → 700]
```

$$\left\{\left\{x \to \text{Function}\left[\{t\}, -\frac{1}{2} e^{2t}\left(-14 + 15\, e^{t}\right)\right],\right.\right.$$
$$\left.\left. y \to \text{Function}\left[\{t\}, -e^{2t}\left(-7 + 6\, e^{t}\right)\right]\right\}\right\}$$



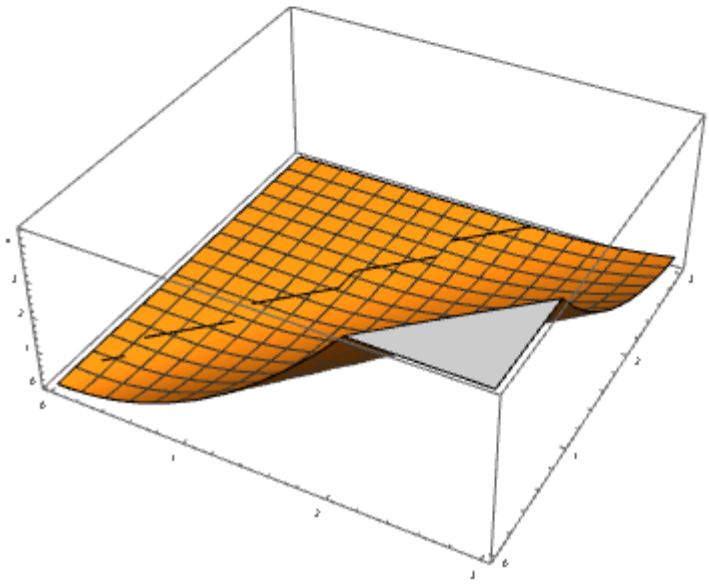
Partielle Differentialgleichungen lösen

```
(* PDGLn Arbeitsbereich *)
(* Definiert die Lösungsfunktion in 2 oder 3 Variablen *)
fn := u[x, y]; fn3 := u[x, y, z];
(* Definiert die partiellen Ableitung *)
u'x := D[fn, x]; u'y := D[fn, y]; u'z := D[fn, z];
u''xx := D[fn, {x, 2}]; u''yy := D[fn, {y, 2}]; u''zz := D[fn, {z, 2}];
u''xy := D[D[fn, x], y]; u''yz := D[D[fn, y], z]; u''xz := D[D[fn, x], z];
(* Definiert die Nebenbedingungen *)
Anfangsbedingung = {u[x, 0] == x^2};
Randbedingung = {u[0, y] == 0};
(* Definiert die eigentliche PDGL *)
PDGL = 2 * u'x + 3 * u'y + fn == 0;(* Im Beispiel: Linear I. Ordnung *)
(* Befehl zum Lösen der PDGL *)
Lösung = DSolve[PDGL, fn, {x, y}]
LösungNebenbedingungen = DSolveValue[{PDGL, Randbedingung,
      Anfangsbedingung}, u[x, y], {x, y}] // FullSimplify
(* Verifiziert die Korrektheit der Lösung *)
PDGL /. Lösung[[1]] // Simplify
(* Plottet die Lösung für Probleme mit Nebenbedingungen. Ansonsten nächste Zelle benutzen *)
Plot3D[LösungNebenbedingungen // Evaluate, {x, 0, 3},
 {y, 0, 3}, Exclusions → None, ImageSize → 700]
```

$$\left\{\left\{u[x, y] \to e^{-x/2}\, C[1]\left[\frac{1}{2}\,(-3x + 2y)\right]\right\}\right\}$$

$$-\frac{1}{9}\, e^{-y/3}\,(3x - 2y)^2\left(-1 + \text{HeavisideTheta}\left[-\frac{3x}{2} + y\right]\right)$$

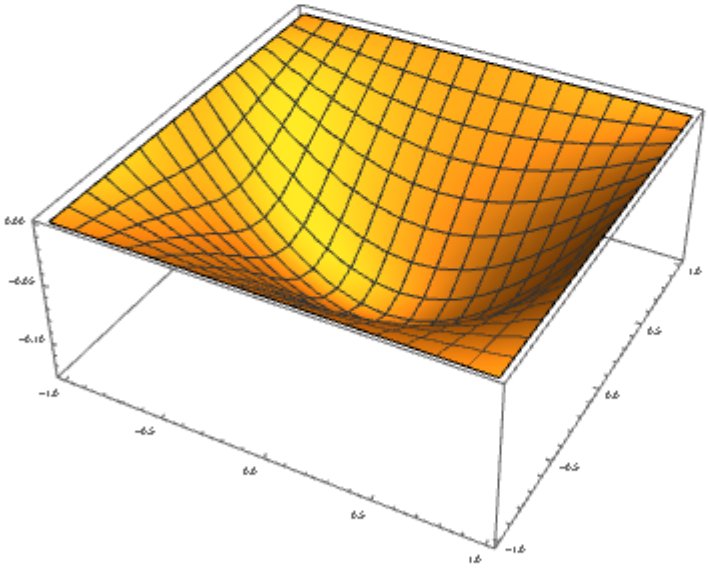
$$e^{-x/2}\, C[1]\left[-\frac{3x}{2} + y\right] + 3\, u^{[0,1)}[x, y] + 2\, u^{[1,0)}[x, y] == 0$$

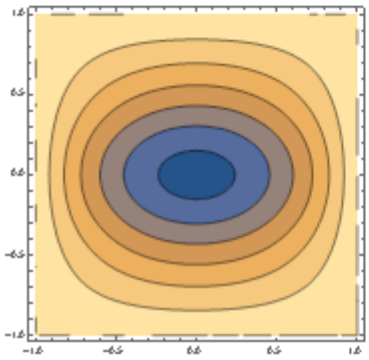


```
(* PDGLn Numerisch lösen: Arbeitsbereich *)
(* Definiert die Lösungsfunktion in 2 oder 3 Variablen *)
fn := u[x, y]; fn3 := u[x, y, z];
(* Definiert die partiellen Ableitung *)
u'x := D[fn, x]; u'y := D[fn, y]; u'z := D[fn, z];
u''xx := D[fn, {x, 2}]; u''yy := D[fn, {y, 2}]; u''zz := D[fn, {z, 2}];
u''xy := D[D[fn, x], y]; u''yz := D[D[fn, y], z]; u''xz := D[D[fn, x], z];
(* Definiert die Nebenbedingungen *)
Anfangsbedingung = {u[x, 0] == x^2};(* Wird hier nicht gebraucht *)
Randbedingung = {u[x, -1] == u[x, 1] == u[-1, y] == u[1, y] == 0};
(* Definiert und löst die eigentliche PDGL *)
(* Im Beispiel: Linear II. Ordnung , inhomogen *)(* Hier wird numerisch approximiert *)
NumLösung = NDSolveValue[{u''xx + u''yy == Exp[-(x^2 + 10 y^2)],
    Randbedingung}, u, {x, -1, 1}, {y, -1, 1}]
(* 3d-Pot der Lösung für Probleme mit Nebenbedingungen *)
Plot3D[NumLösung[x, y], {x, -1, +1}, {y, -1, +1},
 Exclusions → None, ImageSize → 700]
(* Konturen-Pot der Lösung für Probleme mit Nebenbedingungen *)
ContourPlot[NumLösung[x, y], {x, -1, +1}, {y, -1, +1}]
```

```
Null²
```

```
InterpolatingFunction[  ]
```

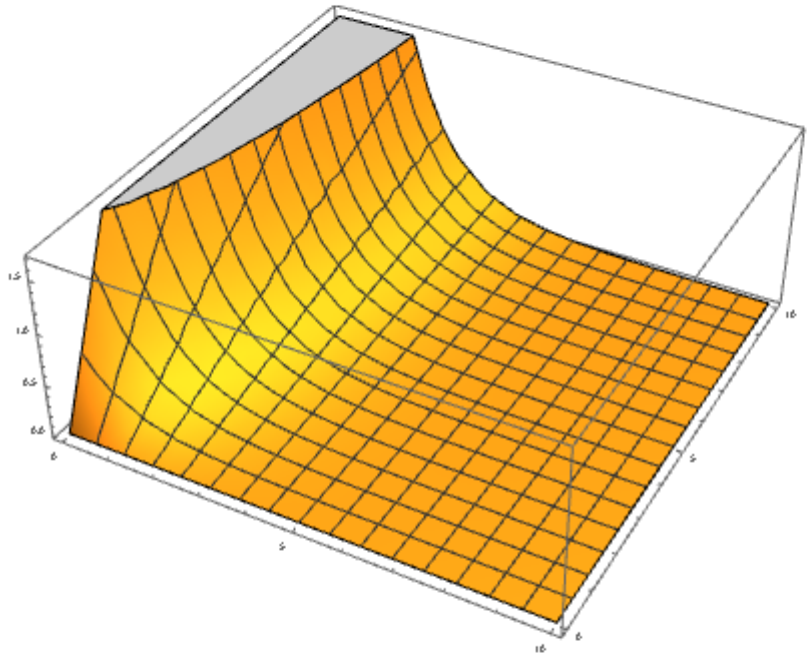





```
(* Linear I. Ordnung ohne Nebenbedingungen – Transportgleichung *)
PDGL1 = D[u[t, x], t] + x * D[u[t, x], x] == 0;
(* Lösung berechnen *)
Lösung = DSolve[{PDGL1}, u[t, x], {t, x}]
sol2 = Lösung[[1, 1, 2]]; sol2 /. {C[1] → 1}; Lösungsfunktion = sol2[[1]]
Plot3D[Lösungsfunktion, {t, 0, 10}, {x, 0, 10}, ImageSize → 800]
```

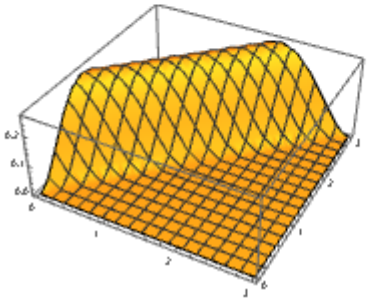
$$\{\{u[t, x] \to C[1]\left[e^{-t}\, x\right]\}\}$$

$$e^{-t}\, x$$



```
(* Linear I. Ordnung mit Anfangs– und Randwertbedingung– Transportgleichung *)(* Erst ab Mathematica 11! *)
PDGL2 = D[u[t, x], t] + D[u[t, x], x] == 0;
Nebenbedingungen = {u[t, 0] == 0, u[0, x] == E^(-x) Sin[x]^2};
Lösung = DSolveValue[{PDGL2, Nebenbedingungen}, u[t, x], {t, x}] // FullSimplify
Plot3D[Lösung // Evaluate, {t, 0, 3}, {x, 0, 3}, Exclusions → None]
```

$$e^{t-x}\, \text{HeavisideTheta}[-t + x]\, \text{Sin}[t - x]^2$$

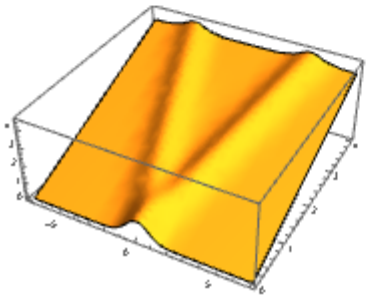


```
(* Linear II. Ordnung Allgemein - Wärmeleitung -
 Wellengleichung*) (* Erst ab Mathematica 11! *)
(* Definiert die eigentliche PDGL  II. Ordnung in x und t *)
fn := u[x, t]; k := 1;
u't := D[fn, t]; u''tt := D[fn, {t, 2}]; u''xx := D[fn, {x, 2}];
Wärmegleichung = u't == k * u''xx;
Wellengleichung = u''tt == u''xx; ic = u[x, 0] == E^(-x^2);
(* Definition der Anfangsbedingung *)
AnfangsbedingungWärme = u[x, 0] == x;
AnfangsbedingungWelle = {u[x, 0] == E^(-x^2), Derivative[0, 1][u][x, 0] == 1};
(* Anfangsbedingung={u[x,0]==UnitBox[x]+UnitTriangle[x/3],Derivative[0,1][u][x,0]==0};
 Anfangsbedingung gebietsweise *)
(* Anfangsbedingung={u[x,0]==E^(-(x-6)^2)+E^(-(x+6)^2),Derivative[0,1][u][x,0]==1/2};
Anfangsbed. Summe exp. Fktn.  *)
(* Lösung  abrufen *)
DSolveValue[{Wärmegleichung, AnfangsbedingungWärme}, fn, {x, t}]
DSolve[Wellengleichung, fn, {x, t}]
DSolveValue[{Wellengleichung, AnfangsbedingungWelle}, fn, {x, t}]
(* Plotten *)
Plot3D[%, {x, -7, 7}, {t, 0, 4}, Mesh → None]
```

x

```
{{u[x, t] → C[1][t - x] + C[2][t + x]}}
```

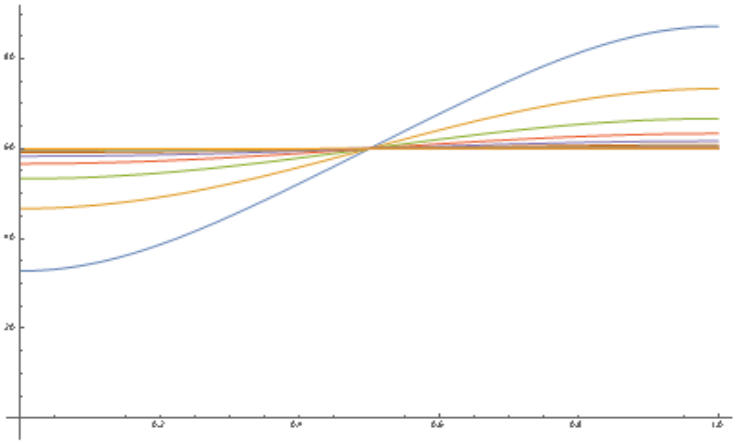
$$\frac{1}{2}\left\{e^{-(-t+x)^2} + e^{-(t+x)^2}\right\} + t$$

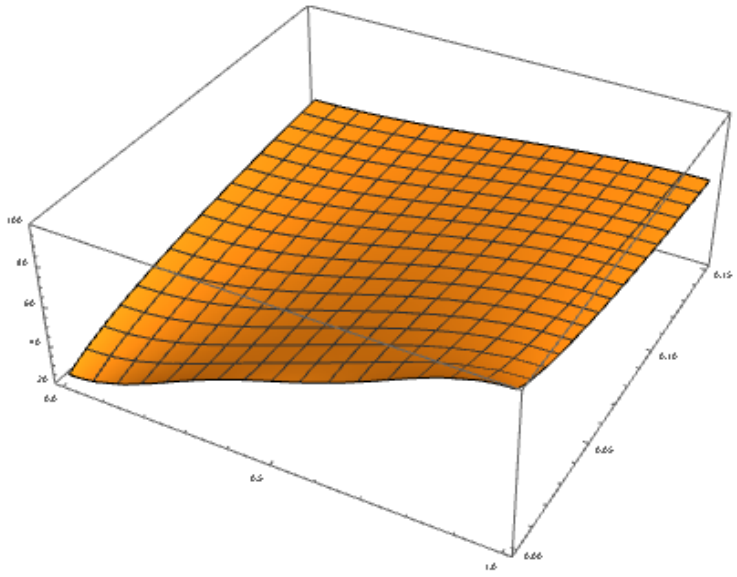


```
(* Wärmeleitungsgleichung mit Ableitung als Randwertbedingung *)(* Erst ab Mathematica 11! *)
(* Allgemeine Definition der Wärmeleitungsgleichung *)
WLGL = D[u[x, t], t] == D[u[x, t], {x, 2}];
(* Definition der RWBn *)
Randwertbedingungen = {Derivative[1, 0][u][0, t] == 0, Derivative[1, 0][u][1, t] == 0};
(* An den Rändern 0 und 1 ist die I. Ableitung Null → keine Wärmeänderung ! *)
(* Definition der AWBn *)
Anfangswertbedingung = u[x, 0] == 20 + 80 x;
(* Anfangszustand = lineare Funktion *)
(* Lösung wird abgerufen *)
Lösung = DSolve[{WLGL, Randwertbedingungen, Anfangswertbedingung}, u[x, t], {x, t}]
(* Die Lösung wird als allgemeine Fourier-Reihe ausgegeben *)
(* Aus der - eigentlich unendlichen - Summe werden die ersten 4 Terme extrahiert *)
Näherungslösung = u[x, t] /. Lösung[[1]] /. {Infinity → 4} // Activate // Expand
(* Ausgabe als 2d-Plot *)
Plot[Table[Näherungslösung, {t, 0.02, 0.9, 0.07}] // Evaluate, {x, 0, 1},
 AxesOrigin → {0, 0}, PlotRange → All, ImageSize → 800]
(* Ausgabe als 3d-Plot *)
Plot3D[Näherungslösung, {x, 0, 1}, {t, 0, .15}, PlotRange → {20, 100}, ImageSize → 800]
```

$$\left\{\left\{u[x, t] \to 60 + 2 \sum_{K[1]=1}^{\infty} \frac{80 \left(-1 + (-1)^{K[1]}\right) e^{-\pi^2 t K[1]^2} \text{Cos}[\pi x K[1]]}{\pi^2 K[1]^2}\right\}\right\}$$

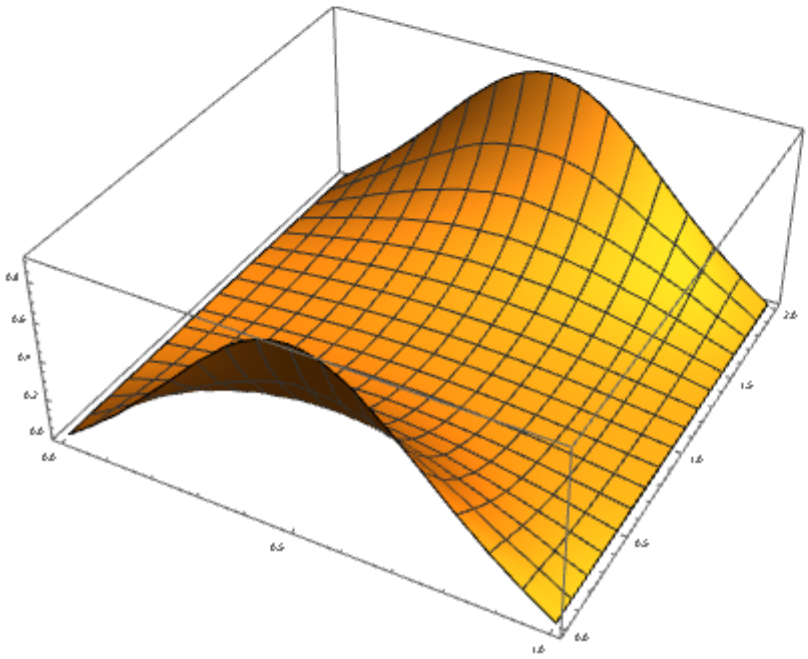
$$60 - \frac{320\, e^{-\pi^2 t} \text{Cos}[\pi x]}{\pi^2} - \frac{320\, e^{-9\pi^2 t} \text{Cos}[3\pi x]}{9\pi^2}$$





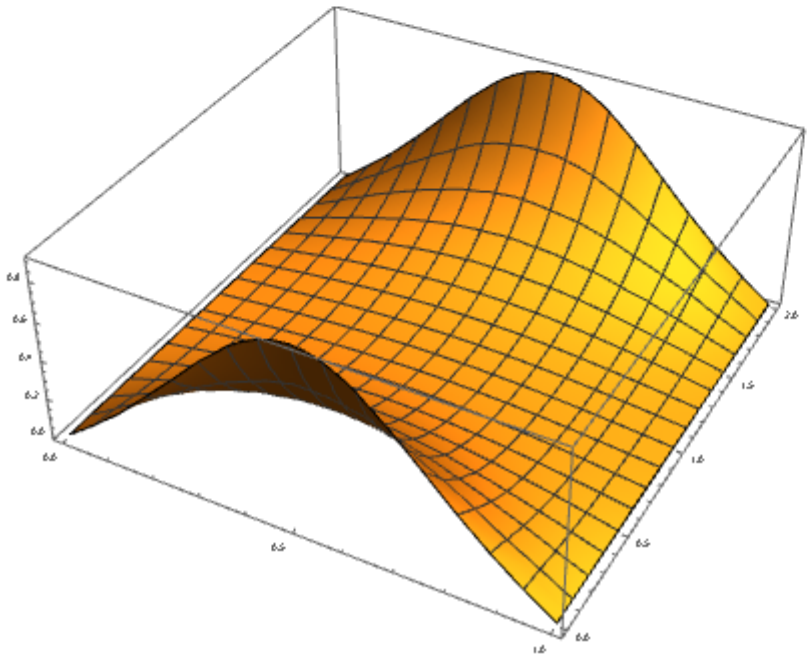
```
(* Laplace–Gleichung – Dirichlet–Bedingung *)(* Erst ab Mathematica 11! *)
(* Allgemeine Definition der Laplace–Gleichung *)
LaGL = Laplacian[u[x, y], {x, y}] == 0;
(* Definition des Gebietes Ω *)
Ω = Rectangle[{0, 0}, {1, 2}];
(* Definition der RWBn *)RWB = DirichletCondition[u[x, y] ==
    Piecewise[{{UnitTriangle[2 x - 1], y == 0 || y == 2}}, 0], True];
(* Lösung wird abgerufen *)
Lösung = DSolveValue[{LaGL, RWB}, u[x, y], {x, y} ∈ Ω] // FullSimplify
(* Die Lösung wird als allgemeine Fourier-Reihe ausgegeben *)
(* Aus der - eigentlich unendlichen - Summe werden die ersten
  4 Terme extrahiert *)
Näherungslösung = Lösung /. {∞ → 4} // Activate;
(* Ausgabe als 3d-Plot *)
Plot3D[Näherungslösung // Evaluate, {x, y} ∈ Ω, PlotRange → All,
 ImageSize → 800]
(* ContourPlot: PlotStyle heißt hier ContourStyle!*)
ContourPlot[Näherungslösung, {x, 0, 1}, {y, 0, 2}]

(* LaPlace auf der Scheibe mit NDSolve *)
f = NDSolveValue[{Laplacian[u[x, y], {x, y}] == 1,
    DirichletCondition[u[x, y] == Sin[5 x * y], True]}, u[x, y], {x, y} ∈ Disk[]];
Plot3D[f, {x, y} ∈ Disk[]]
```

$$\sum_{K[1]=1}^{\infty} \frac{8\,\text{Cosh}[\pi\,(-1+y)\,K[1]]\,\text{Sech}[\pi\,K[1]]\,\text{Sin}\left[\frac{1}{2}\,\pi\,K[1]\right]\,\text{Sin}[\pi\,x\,K[1]]}{\pi^2\,K[1]^2}$$



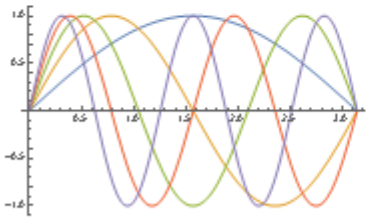
```
(* Laplace-Gleichung - Dirichlet-Bedingung *)(* Erst ab Mathematica 11! *)
(* Allgemeine Definition der Laplace-Gleichung *)
LaGL = Laplacian[u[x, y], {x, y}] == 0;
(* Definition des Gebietes Ω *)
Ω = Rectangle[{0, 0}, {1, 2}];
(* Definition der RWBn *)RWB = DirichletCondition[u[x, y] ==
    Piecewise[{{UnitTriangle[2 x - 1], y == 0 || y == 2}}, 0], True];
(* Lösung wird abgerufen *)
Lösung = DSolveValue[{LaGL, RWB}, u[x, y], {x, y} ∈ Ω] // FullSimplify
(* Die Lösung wird als allgemeine Fourier-Reihe ausgegeben *)
(* Aus der - eigentlich unendlichen - Summe werden die ersten
  4 Terme extrahiert *)
Näherungslösung = Lösung /. {∞ → 4} // Activate;
(* Ausgabe als 3d-Plot *)
Plot3D[Näherungslösung // Evaluate, {x, y} ∈ Ω, PlotRange → All,
 ImageSize → 800]
(* ContourPlot: PlotStyle heißt hier ContourStyle!*)
ContourPlot[Näherungslösung, {x, 0, 1}, {y, 0, 2}]

(* LaPlace auf der Scheibe mit NDSolve *)
f = NDSolveValue[{Laplacian[u[x, y], {x, y}] == 1,
    DirichletCondition[u[x, y] == Sin[5 x * y], True]}, u[x, y], {x, y} ∈ Disk[]];
Plot3D[f, {x, y} ∈ Disk[]]
```

$$\sum_{K[1]=1}^{\infty} \frac{8\,\text{Cosh}[\pi\,(-1+y)\,K[1]]\,\text{Sech}[\pi\,K[1]]\,\text{Sin}\left[\frac{1}{2}\,\pi\,K[1]\right]\,\text{Sin}[\pi\,x\,K[1]]}{\pi^2\,K[1]^2}$$



```
(* Sturm-Liouville *)(* Erst ab Mathematica 11! *)
(* Gleich mit Nebenbedingungen *)
Lösung = DSolve[{y''[x] + λ y[x] == 0, y[0] == 0, y[π] == 0}, y[x], x]
(* Ausgabe der Eigenfunktionen *)
Eigenfunktionen = Table[y[x] /. Lösung[[1]] //. {n → i, λ → n^2} /. {C[1] → 1}, {i, 5}];
(* Plotten der Eigenfunktionen *)
Plot[Evaluate[Eigenfunktionen], {x, 0, Pi}]
```

$$\left\{\left\{y[x] \to \begin{cases} C[1]\ \text{Sin}\left[x\sqrt{\lambda}\right] & n \in \text{Integers} \;\&\&\; n \geq 1 \;\&\&\; \lambda == n^2 \\ 0 & \text{True} \end{cases}\right\}\right\}$$



```
(* Vibrationen gespannte Saite *)(* Erst ab Mathematica 11! *)
(* Dauert sehr lange und ist instabil gegenüber vorherigen Berechnungen!
  Ggfs. neu und solo starten !!! *)
(* Wieder Wellengleichung wie vor *)
Wellengleichung = D[u[x, t], {t, 2}] == D[u[x, t], {x, 2}];
(* Saite ist auf beiden Seiten eingespannt *)
Randbedingung = {u[0, t] == 0, u[π, t] == 0};
(* Auslenkung an bestimmten Stellen am Anfang *)
Anfangsbedingung = {u[x, 0] == x^2 (π - x), u^(0,1)[x, 0] == 0};
(* Lösung mit Nebenbedingungen *)
Lösung = DSolve[{Wellengleichung, Randbedingung, Anfangsbedingung},
    u, {x, t}] /. {K[1] → m}
Teillösungen[x_, t_] = u[x, t] /. Lösung[[1]] /. {∞ → 4} // Activate
(* Plotten der Eigenfunktionen *)
Table[Show[Plot[Table[Teillösungen[x, t][[m]], {t, 0, 4}] // Evaluate, {x, 0, Pi}, Ticks → False],
   ImageSize → 150], {m, 4}]
```

$$\left\{\left\{u \to \text{Function}\left[\{x, t\}, \sum_{m=1}^{\infty} -\frac{4\,(1 + 2\,(-1)^m)\,\text{Cos}[t\,m]\,\text{Sin}[x\,m]}{m^3}\right]\right\}\right\}$$

$$4\,\text{Cos}[t]\,\text{Sin}[x] - \frac{3}{2}\,\text{Cos}[2\,t]\,\text{Sin}[2\,x] + \frac{4}{27}\,\text{Cos}[3\,t]\,\text{Sin}[3\,x] - \frac{3}{16}\,\text{Cos}[4\,t]\,\text{Sin}[4\,x]$$