```
In[678]:= ClearAll["Global`*"]
          lösche alle

In[679]:= (* Parameter for FrameStyle *)
                              Rahmenstil
          gr = 14;

In[680]:=
          (* Conversion factor;
          conv = 1 for time unit = day;
          conv = 86400 for time unit = second. Verification of the analysis was only run with conv=1. *)
          conv = 1

Out[680]= 1

In[681]:=
          (* Duration of simulation *)
          dur = 150 * conv

Out[681]= 150

In[682]:=
          (* parameters: Table 2 *)
                              Tabelle
          par = {ALPHA1 → 3 / conv, BETA1 → 0.2 / conv, K1 → 0.0024 / conv, a1 → 1.6, b1 → -0.49, ALPHA2 → 4 / conv,
            BETA2 → 0.02 / conv, K2 → 0.000017 / conv, a2 → -1.6, b2 → 0.6, γ1 → 16.67 / conv, γ2 → 33.37 / conv, rho0 → 500}

Out[682]= {ALPHA1 → 3, BETA1 → 0.2, K1 → 0.0024, a1 → 1.6, b1 → -0.49, ALPHA2 → 4,
           BETA2 → 0.02, K2 → 0.000017, a2 → -1.6, b2 → 0.6, γ1 → 16.67, γ2 → 33.37, rho0 → 500}
```

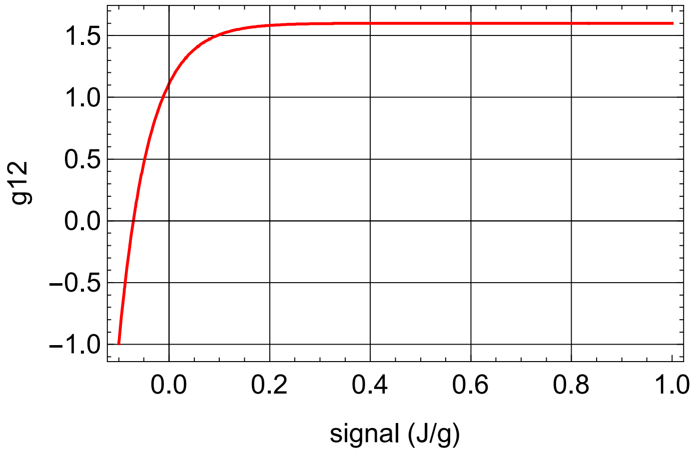
In[683]:=

```
(* Equation 10; ψ is Δψ in the publication *)
g12 = a1 + b1 * Exp[-γ1 * ψ]  /. par
                 Exponentialfunktion
g21 = a2 + b2 * Exp[-γ2 * Abs[ψ]]  /. par
                 Exponentia·· Absolutwert

(* Fig. 4 *)
g12P = Plot[g12, {ψ, -0.1, 1}, PlotStyle → Directive[Red, Thickness[0.005]], Frame → True,
       stelle Funktion graphisch dar  Darstellungsstil Anweisung rot Dicke  Rahmen wahr
  FrameStyle → {Directive[Black, gr], Directive[Black, gr]}, FrameLabel → {"signal (J/g)", "g12"},
  Rahmenstil  Anweisung schwarz  Anweisung schwarz  Rahmenbeschriftung
  PlotRange → All, GridLines → Automatic, GridLinesStyle → Directive[Black], PlotLabel → None]
  Koordinatenbe·· alle Gitternetzlinien automatisch Stil der Gitternetzlinien Anweisung schwarz Beschriftung d··· keine

g21P = Plot[g21, {ψ, -0.1, 1}, PlotStyle → Directive[Red, Thickness[0.005]], Frame → True,
       stelle Funktion graphisch dar  Darstellungsstil Anweisung rot Dicke  Rahmen wahr
  FrameStyle → {Directive[Black, gr], Directive[Black, gr]}, FrameLabel → {"signal (J/g)", "g21"},
  Rahmenstil  Anweisung schwarz  Anweisung schwarz  Rahmenbeschriftung
  PlotRange → All, GridLines → Automatic, GridLinesStyle → Directive[Black], PlotLabel → None]
  Koordinatenbe·· alle Gitternetzlinien automatisch Stil der Gitternetzlinien Anweisung schwarz Beschriftung d··· keine
```

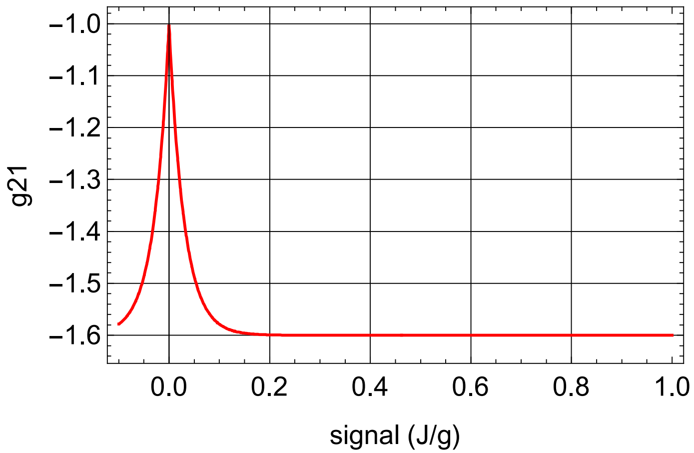
Out[683]= $1.6 - 0.49\, e^{-16.67\, \psi}$

Out[684]= $-1.6 + 0.6\, e^{-33.37\, \mathrm{Abs}[\psi]}$

Out[685]=



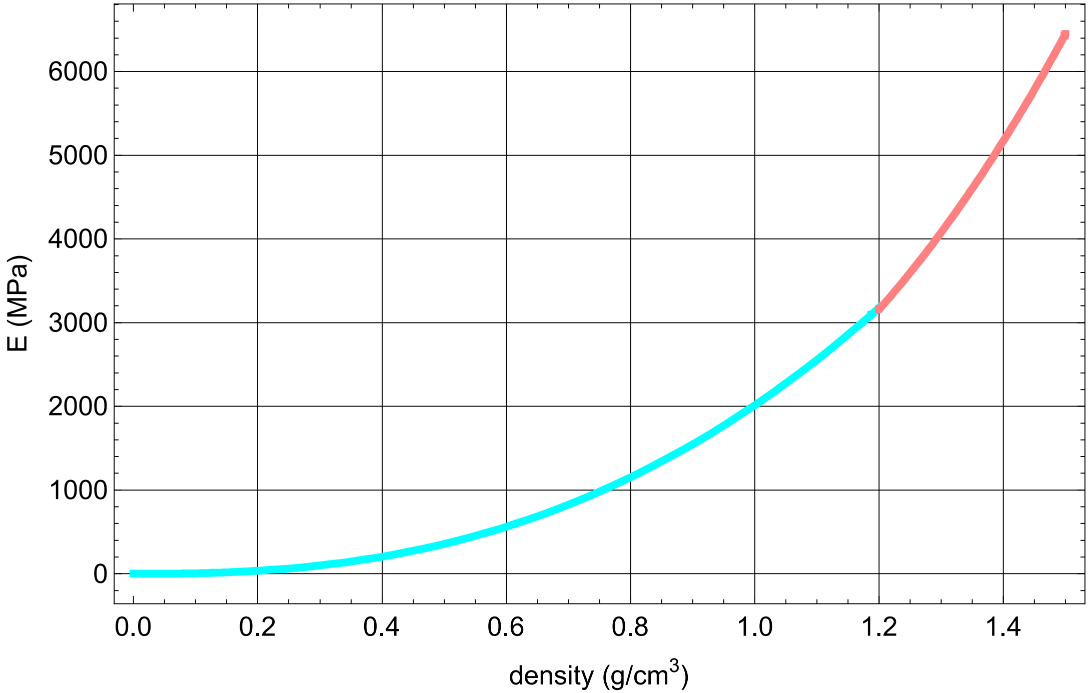
Out[686]=



In[687]:=

```
(* Modulus vs. density *)
   Modulus
E1 = 2014 * ρ^2.5;
E1P = Plot[E1, {ρ, 0, 1.2}, PlotStyle → Directive[Cyan, Thickness[0.008]]];
      stelle Funktion graphisch dar  Darstellungsstil  Anweisung  blaug··  Dicke
E2 = 1762 * ρ^3.2;
E2P = Plot[E2, {ρ, 1.2, 1.5}, PlotStyle → Directive[Pink, Thickness[0.008]]];
      stelle Funktion graphisch dar  Darstellungsstil  Anweisung  rosa  Dicke
EPlot = Show[E1P, E2P, Frame → True,
        zeige an        Rahmen   wahr
  FrameStyle → {Directive[Black, gr], Directive[Black, gr]}, FrameLabel → {"density (g/cm^3)", "E (MPa)"},
  Rahmenstil   Anweisung  schwarz  Anweisung  schwarz  Rahmenbeschriftung  Exponentialkons
  PlotRange → All, GridLines → Automatic, GridLinesStyle → Directive[Black], PlotLabel → None]
  Koordinatenbe··  alle  Gitternetzlinien  automatisch  Stil der Gitternetzlinien  Anweisung  schwarz  Beschriftung d···  keine
```

Out[691]=



In[692]:=
```
E1 /. ρ → 0.5
```
Out[692]= 356.028

In[693]:=
```
(* list of ψ-values *)
Li = {-0.0537, 0, 1.04}
```
Out[693]= {-0.0537, 0, 1.04}

In[694]:=

```
(* Color-coding: Mathematica result for the differential equations *)
col = {Blue, Black, Red};
        blau  schwarz  rot

(* Color-coding: superimpose dots for ABAQUS results *)
colr = {Yellow, Red, Blue};
        gelb    rot  blau
```

In[696]:=

```
H[x_] := (x + Abs[x])/2
```

In[697]:=

```
n1Plotfkt = {};
n2Plotfkt = {};

r1Plotfkt = {};
r2Plotfkt = {};

n1Plotfem = {};
n2Plotfem = {};

r1Plotfem = {};
r2Plotfem = {};

(* initial numbers, eq. 1 *)
```

```
y0 = 10;
z0 = 0.0001;

For[sdv = 1, sdv ≤ 19, sdv++,
For-Schleife
  SDVPlotFEM[sdv] = {};
  fk[sdv] = 1.0;
 ];
fk[7] = 1/rho0 /. par; (* scale density *)
fk[8] = 86400;
fk[16] = 86400;
fk[17] = 86400;

tmin = 0.001;
```

In[713]:=

```
rhoPlotfem = {};
drhodtPlotfkt = {};
drhodtPlotfem = {};
EPlotfem = {};
ErhoPlotfem = {};
n1barPlotfkt = {};
n2barPlotfkt = {};
```

```
For[i = 1, i <= Length[Li], i++,
For–Schleife      Länge

  Print["i = ", i];
  gebe aus

  (* Eq. 14 *)
  expo = 1.0 / (g12 * g21 - 1.0)  /. par /. ψ -> Li[[i]] ;
  n1bar = ( (BETA1 / ALPHA1) * ( (BETA2 / ALPHA2)^g21 ) )^expo  /. par /. ψ -> Li[[i]];
  n2bar = ( (BETA2 / ALPHA2) * ( (BETA1 / ALPHA1)^g12 ) )^expo  /. par /. ψ -> Li[[i]];
  n1barP[i] = Plot[n1bar, {x, tmin, dur}, PlotStyle -> {{col[[i]], Thickness[0.01]}}, PlotRange -> All];
              stelle Funktion graphisch dar     Darstellungsstil     Dicke     Koordinatenbere··  alle
  AppendTo[n1barPlotfkt, n1barP[i]];
  hänge an bei
  n2barP[i] = Plot[n2bar, {x, tmin, dur}, PlotStyle -> {{col[[i]], Thickness[0.01]}}, PlotRange -> All];
              stelle Funktion graphisch dar     Darstellungsstil     Dicke     Koordinatenbere··  alle
  AppendTo[n2barPlotfkt, n2barP[i]];
  hänge an bei

  (* y: osteoclasts, z: osteoblasts;
  Modeling of bone adaptative behavior based on cells activities, Section 3.1: g11=g22=0 *)
  sol[i] = NDSolve[ {y'[x]  == ALPHA1 * (z[x]^g21) - BETA1 * y[x],  z'[x]  == ALPHA2 * (y[x]^g12) - BETA2 * z[x],
           löse Differentialgleichung numerisch
      y[0]  == y0, z[0]  == z0}  /. par /. ψ -> Li[[i]], {y, z}, {x, tmin, dur} ];

  (* eq. 5 *)
  f1 = K1 * H[Evaluate[ y[x]  /. sol[i] ] - n1bar]  /. par;
              werte aus
```

```
                 werte aus
f2 = K2 * H[Evaluate[ z[x] /. sol[i] ] - n2bar] /. par;
                 werte aus
drhodt[i] = rho0 * (f2[[1]] - f1[[1]]) /. par;
drhodtP[i] = Plot[drhodt[i], {x, tmin, dur}, PlotStyle -> {{col[[i]], Thickness[0.01]}}, PlotRange -> All];
              stelle Funktion graphisch dar   Darstellungsstil       Dicke          Koordinatenbere··· alle
AppendTo[drhodtPlotfkt, drhodtP[i]];
hänge an bei

(* number of osteoclasts *)
n1P[i] =
 Plot[Evaluate[ y[x] /. sol[i] ], {x, tmin, dur}, PlotStyle -> {{col[[i]], Thickness[0.01]}}, PlotRange -> All];
 stell··· werte aus                          Darstellungsstil       Dicke          Koordinatenbere··· alle
AppendTo[n1Plotfkt, n1P[i]];
hänge an bei

(* rate of osteoclasts *)
r1P[i] = Plot[Evaluate[ ALPHA1 * (z[x]^g21) - BETA1 * y[x] /. par /. ψ -> Li[[i]] /. sol[i] ],
           stell··· werte aus
  {x, 0.25, dur}, PlotStyle -> {{col[[i]], Thickness[0.01]}}, PlotRange -> All];
                  Darstellungsstil       Dicke          Koordinatenbere··· alle
AppendTo[r1Plotfkt, r1P[i]];
hänge an bei

(* number of osteoblasts *)
n2P[i] =
 Plot[Evaluate[z[x] /. sol[i] ], {x, tmin, dur}, PlotStyle -> {{col[[i]], Thickness[0.01]}}, PlotRange -> All];
 stell··· werte aus                          Darstellungsstil       Dicke          Koordinatenbere··· alle
```

```
 stell    werte aus                                            Darstellungsstil              Dicke                  Koordinatenbere   alle
AppendTo[n2Plotfkt, n2P[i]];
hänge an bei

(* rate of osteoblasts *)
r2P[i] = Plot[Evaluate[ ALPHA2 * (y[x]^g12) - BETA2 * z[x] /. par /. ψ -> Li[[i]] /. sol[i] ],
              stell···  werte aus
   {x, tmin, dur}, PlotStyle -> {{col[[i]], Thickness[0.01]}}, PlotRange -> All];
                   Darstellungsstil          Dicke               Koordinatenbere··  alle
AppendTo[r2Plotfkt, r2P[i]];
hänge an bei

(* read Abaqus results *)
SetDirectory[StringJoin["M:\\auslagern\\Komarova-neu\\FEM\\V0", ToString[i], "\\V0", ToString[i], "_rpt"]];
             vereinige Zeichenketten                            wandle in eine Zeichenkette···  wandle in eine Zeichenkette um
For[sdv = 1, sdv ≤ 19, sdv++, vs = "0";
For–Schleife
 If[sdv ≥ 10, vs = ""];
 wenn
 datF = StringJoin["V0", ToString[i], "-SDV", vs, ToString[sdv], "-element1.rpt"];
        vereinige Zeichenketten  wandle in eine Zeichenkette um    wandle in eine Zeichenkette um
 datD = OpenRead[datF];
        öffne, um zu lesen
 Skip[datD, String, 4];
            Zeichenkette
 datL[i, sdv] = ReadList[datD, Table[Number, {2}]];
                lese Liste          Tabelle Zahl
 datLmod[i, sdv] =
```

```
  Table[{datL[i, sdv][[k]][[1]] * conv / 86400, datL[i, sdv][[k]][[2]] * fk[sdv]}, {k, 1, Length[datL[i, sdv]], 2}];
  Tabelle                                                                          Länge
 Close[datF];
 schließe
 simP[i, sdv] = ListPlot[datLmod[i, sdv], PlotStyle → {colr[[i]], PointSize[0.005]}, Frame → True, FrameLabel →
                listenbezogene Graphik       Darstellungsstil          Punktgröße          Rahmen  wahr  Rahmenbeschriftung
    {"t", ""}, PlotLabel → None, GridLines → Automatic, AxesStyle → {42, FontWeight → Bold}, PlotRange → All];
               Beschriftung d··· keine Gitternetzlinien automatisch Achsenstil  Schriftstärke  fett  Koordinatenbe·· alle
 AppendTo[SDVPlotFEM[sdv], simP[i, sdv]];
 hänge an bei
]; (* sdv *)


(* E in MPa vs. density. Mind that the unit of mass in FEM is 1 kg,
   Exponentialkonstante E
thus scale with 1/1000, and density was scaled above with rho0. *)
ErhoL[i] = Table[{datLmod[i, 7][[j]][[2]] * (1 / 1000) * rho0 /. par, 10^-6 * datLmod[i, 10][[j]][[2]]},
           Tabelle
  {j, 1, Length[datLmod[i, 7]]}];
         Länge
ErhoP[i] = ListPlot[ErhoL[i], PlotStyle → {colr[[i]], PointSize[0.005]}, Frame → True, FrameLabel → {"rho", "E"},
           listenbezogene Graphik  Darstellungsstil    Punktgröße          Rahmen  wahr  Rahmenbeschriftung          Exponenti
  PlotLabel → None, GridLines → Automatic, AxesStyle → {42, FontWeight → Bold}, PlotRange → All];
              keine  Gitternetzlinien  automatisch  Achsenstil     Schriftstärke  fett   Koordinatenbe·· alle
AppendTo[ErhoPlotfem, ErhoP[i]];
hänge an bei
```

```
  Print[""];
  gebe aus

];

i = 1

i = 2

i = 3
```

In[721]:=

```
For[sdv = 1, sdv ≤ 19, sdv++,
For–Schleife
  SDVPlotFEM[sdv] = Show[SDVPlotFEM[sdv]];
                    zeige an
  MMAPlot[sdv] = {};
 ];
(* Only the following MMAPlots are created above *)
MMAPlot[1] = n1Plotfkt;
MMAPlot[2] = n2Plotfkt;
MMAPlot[3] = n1barPlotfkt;
MMAPlot[4] = n2barPlotfkt;
MMAPlot[16] = r1Plotfkt;
MMAPlot[17] = r2Plotfkt;
MMAPlot[8] = drhodtPlotfkt;
```

In[729]:=

```
(* save graphs to files *)
SetDirectory["M:\\auslagern\\Komarova-neu\\Analysis"];


tx = {"number of osteoclasts",
"number of osteoblasts",
"number of inactive osteoclasts",
"number of inactive osteoblasts",
"number of active osteoclasts",
"number of active osteoblasts",
"density",
"temporal variation of the density",
"variation of the density",
"Young's modulus",
"Poisson's ratio",
"density of the elastic energy",
"DELTA_E",
"g12",
"g21",
"dn1dt",
"dn2dt",
"omega",
"zeta"};
```

```
For[sdv = 1, sdv ≤ 19, sdv++,
  vs = "0";
  If[sdv ≥ 10, vs = ""];
  vpr = All; (* vertical plot range *)
  If[sdv == 16, vpr = {-10, 0}];
  pl = None;
  If[sdv == 7, pl = "density scaled"];
  Result[sdv] = Show[MMAPlot[sdv], SDVPlotFEM[sdv], FrameStyle → {Directive[Black, gr], Directive[Black, gr]},
    Frame → True, FrameStyle → {Directive[Black, gr], Directive[Black, gr]}, FrameLabel -> {"t", tx[[sdv]]},
    PlotLabel → pl, GridLines → Automatic, PlotRange → {{-0.001, dur + 0.001}, vpr}, TextStyle -> 20];
  Export[StringJoin["SDV", vs, ToString[sdv], "=", tx[[sdv]], ".png"], Result[sdv], ImageSize → 800];
 ];
```

```
ErhoPlota = Show[EPlot, ErhoPlotfem, FrameStyle → {Directive[Black, gr], Directive[Black, gr]},
   Frame → True, FrameStyle → {Directive[Black, gr], Directive[Black, gr]},
```

```
[wahl   [Rahmenstil        [Anweisung   [schwarz          [Anweisung   [schwarz
    FrameLabel → {"density (g/cm^3)", "E (MPa)"}, PlotLabel → "Modulus vs. density",
    [Rahmenbeschriftung                           [Exponentialko… [Beschriftung der … [Modulus
    GridLines → Automatic, AxesStyle → {42, FontWeight → Bold}, PlotRange → {All, All}, TextStyle -> 20];
               [automatisch  [Achsenstil           [Schriftstärke  [fett   [Koordinatenbere… [alle [alle [Textstil
Export["ErhoPlot.png", ErhoPlota, ImageSize -> 800];
                                  [Bildgröße
```

In[734]:=