UPPER LEVEL FRONTOGENESIS

```
SetDirectory["D:/cygwin/home/legras/h2/cours2/M2-approf/fronto"]
```

```
D:\cygwin\home\legras\h2\cours2\M2-approf\fronto
```

```
<< ColorbarPlot`
```

Basic flow

References: Keyser & Pecnick, 1985, J. Atmos. Sci., 42, pp.1259-1282; Keyser & Pecnick, 1985, J. Atmos. Sci., 42, pp. 1283-1305; Keyser & Shapiro, 1986, Mon. Wea. Rev., 114, pp. 452-499; Hoskins & Bretherton, 1972, J. Atmos. Scio., 29, 11-37; Hoskins, 1982, Ann. Rev. Fluid Mech., 14, pp. 131-151.

Note: Solution exacte de l' équation du mouvement avec confluence et advection de température

$$u = -\alpha x + \gamma (z - z_r)\, e^{\alpha t}$$
$$v = \alpha y$$
$$\Phi = g z + \alpha f x y - \frac{\alpha^2}{2}\left(x^2 + y^2\right) - \gamma f y (z - z_r)\, e^{\alpha t}$$
$$b = g - \gamma y f e^{\alpha t}$$

Basic state at initial time in the y=0 plane

h[x] : hauteur de la tropopause
b[x_, z_] : champ de température
Φ[x_, z_] : géopotentiel
Ug, Vg : vitesse géostrophique

$$h[x\_] := h_0 + K \left(N_2^2 - N_1^2\right)^{-1} \mathrm{ArcTan}\left[\frac{x}{L}\right];$$
$$b[x\_, \xi\_] := g + \left(N_1^2\, \xi + K\, \mathrm{ArcTan}\left[\frac{x}{L}\right]\right) \mathrm{UnitStep}[h[x] - \xi] + \left(N_2^2\, \xi - \left(N_2^2 - N_1^2\right) h_0\right) \mathrm{UnitStep}[\xi - h[x]];$$
$$\Phi[x\_, \xi\_] := g\,\xi + K\, \mathrm{Arc.Tan}\left[\frac{x}{L}\right](\xi - z_c)\, \mathrm{UnitStep}[h[x] - \xi] + \left(\left(N_2^2 - N_1^2\right)\left(\frac{1}{2} h^2 - h z_c - h_0 (\xi - z_c)\right) + \frac{1}{2} N_2^2 z^2\right) \mathrm{UnitStep}[\xi - h[x]];$$
$$Ug[x\_, \xi\_] := -\alpha x + \frac{\gamma}{f}(\xi - z_r);$$
$$Vg[x\_, \xi\_] := \frac{K}{f L}\left(1 + \left(\frac{x}{L}\right)^2\right)^{-1}(\xi - z_c)\, \mathrm{UnitStep}[h[x] - \xi] + \frac{K}{f L}\left(1 + \left(\frac{x}{L}\right)^2\right)^{-1}(h[x] - z_c)\, \mathrm{UnitStep}[\xi - h[x]];$$
$$K = \frac{\Delta\theta\, g}{\pi\, \theta_0};$$

Q1 component of the Q-vector and elements of the Sawyer-Eliassen equation

$$Q1[x\_, \xi\_] = \mathrm{Simplify}[f\,(\partial_x Vg[x, \xi]\, \partial_\xi Ug[x, \xi] - \partial_x Ug[x, \xi]\, \partial_\xi Vg[x, \xi])\ /.\ \mathrm{DiracDelta}[t\_] \to 0];$$
$$Ns2[x\_, \xi\_] = \mathrm{Simplify}[\partial_\xi b[x, \xi]\ /.\ \mathrm{DiracDelta}[t\_] \to 0];$$
$$S2[x\_, \xi\_] = \mathrm{Simplify}[\partial_x b[x, \xi]\ /.\ \mathrm{DiracDelta}[t\_] \to 0];$$
$$F2[x\_, \xi\_] = \mathrm{Simplify}[f\,(f + \partial_x Vg[x, \xi])\ /.\ \mathrm{DiracDelta}[t\_] \to 0];$$
$$Ng2[x\_, \xi\_] = \mathrm{Simplify}\left[N_1^2\, \mathrm{UnitStep}[h[x] - \xi] + N_2^2\, \mathrm{UnitStep}[\xi - h[x]]\right];$$
$$PV[x\_, \xi\_] = \mathrm{Simplify}\left[\frac{1}{f}\left(F2[x, \xi]\, Ns2[x, \xi] - S2[x, \xi]^2\right)\right];$$

Values of the parameters

Parameter list

f: Coriolis parameter
g: gravity
Lx, H: width and height of the computing domain
...

Pure confluence

distances are in km, times in second

```
param1 = {f -> 10^-4, g -> 9.81 × 10^-3,
   h0 -> 8.15, H -> 13.5,
   Lx → 2000., L → 500.,
   N1^2 → 10^-4, N2^2 → 9. × 10^-4,
   zr → 5., zc → 0., (*after Keyser and Pecnick,1985*)
   α -> 10^-5, Δθ → 40., θ0 -> 273,
   γ → 0 };
param = param1;
```

Confluence + cold advection

Keyser & Pecnick use L=796, try also L=525

```
param2 = {f → 1. × 10^-4, g -> 9.81 × 10^-3,
   h0 -> 8.15, H -> 13.5,
   Lx → 2000., L → 500.,
   N1^2 → 1. × 10^-4, N2^2 → 9. × 10^-4,
   zr → 5., zc → 0.,
   α → 1. × 10^-5, Δθ → 40., θ0 -> 273,
   γ → -5.74 10^-7 };
param = param2;
```

Confluence + warm advection

```
param3 = {f → 1. × 10^-4, g -> 9.81 × 10^-3,
   h0 -> 8.15, H -> 13.5,
   Lx → 2000., L → 500.,
   N1^2 → 1. × 10^-4, N2^2 → 9. × 10^-4,
   zr → 5., zc → 0.,
   α → 1. × 10^-5, Δθ → 40., θ0 -> 273,
   γ → 5.74 10^-7 };
param = param3;
```

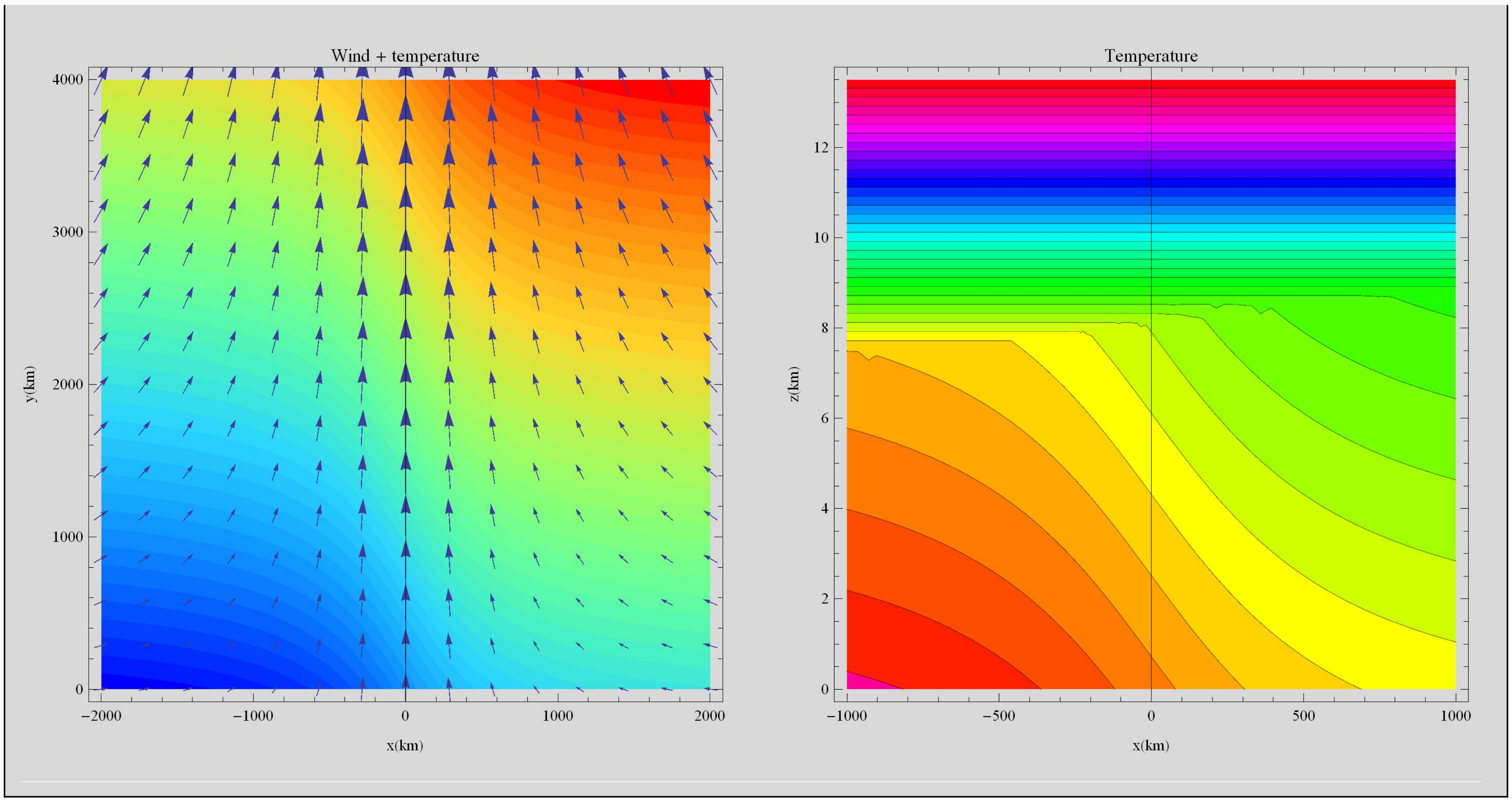
Plots

Horizontal temperature ang geostrophic wind [in the (x,y)-plane] and cross front temperature distribution [in the (x,z)-plane]

COMPATIBILITY ISSUE

```
g1 := VectorPlot[Evaluate[{Ug[x, zr], Vg[x, zr] + α y} /.param], {x, -2000, 2000}, {y, 0, 4000}, DisplayFunction → Identity];
g2 := ContourPlot[Evaluate[((b[x, zr] - γ y) θ0)/g /.param], {x, -2000, 2000}, {y, 0, 4000.}, Contours → 50, ContourStyle → None,
     ColorFunction → (RGBColor[Min[2 #1^2, 1], 4 #1 (1 - #1), Min[2 (1 - #1)^2, 1]] &), DisplayFunction → Identity] /.param;
g3 := ContourPlot[Evaluate[(b[x, z] θ0)/g /.param], {x, -1000, 1000}, {z, 0, 13.5},
   Contours → Table[-50 + 5 i, {i, 200}], ColorFunction → Hue, DisplayFunction → Identity,
   FrameLabel → {"x(km)", "z(km)"}, Axes → True, PlotLabel → "Temperature", Exclusions → None];
g4 := Show[GraphicsRow[{Show[g2, g1, FrameLabel → {"x(km)", "y(km)"}, Axes → True, PlotLabel → "Wind + temperature"], g3}]]
```

```
g4
```



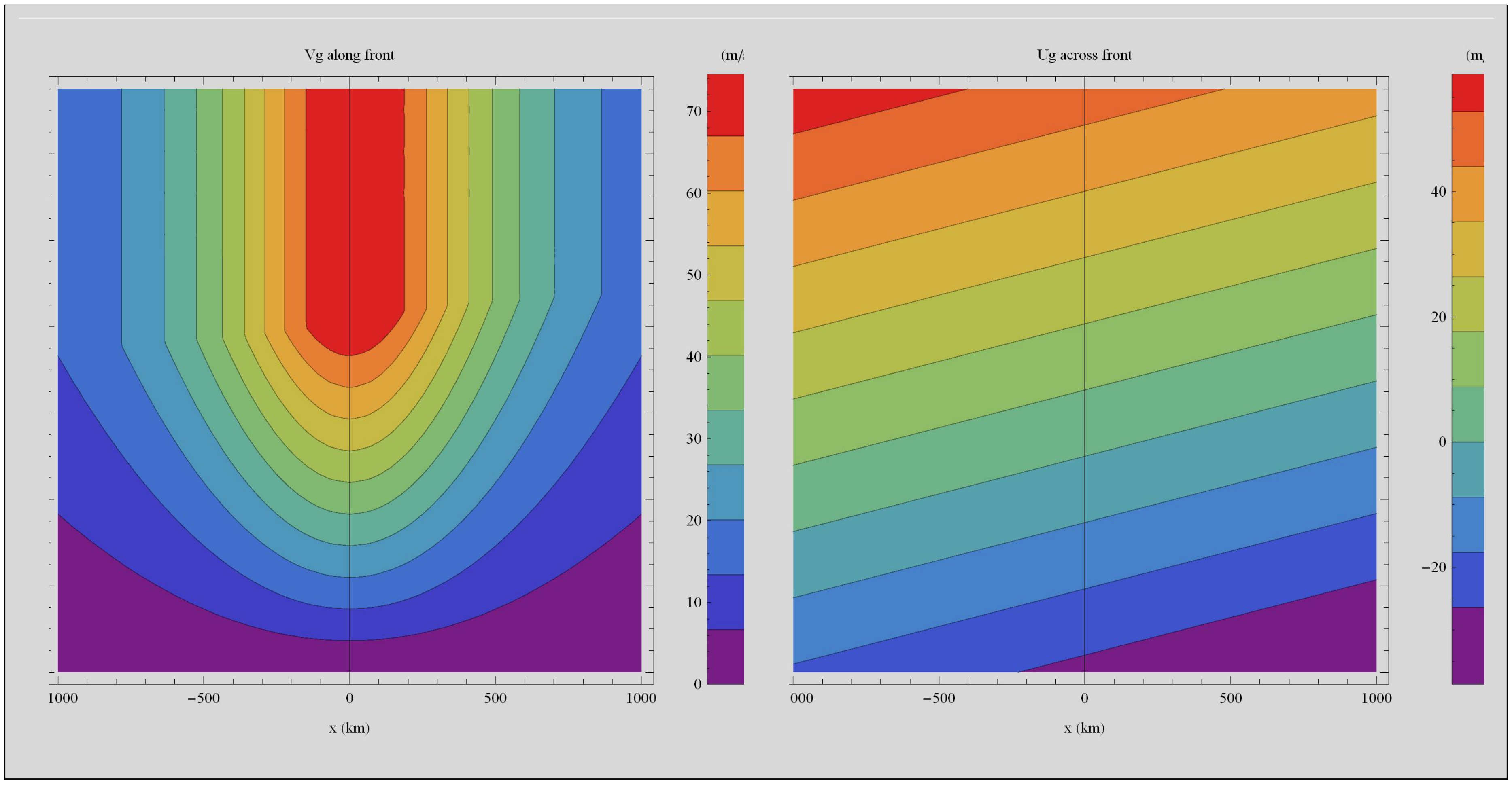
Along front geostrophic wind (x,z)-plane

Cross front geostorphic wind (x,z)-plane

Cross front and along front geostorphic wind (x,z)-plane

```
g5 := Show[GraphicsRow[{ColorbarPlot[Evaluate[1000 Vg[#1, #2] /. param] &, {-1000, 1000},
    {0, 13.5}, PlotType → "Contour", Axes → True, XLabel → "x (km)", YLabel → "z (km)", CLabel → "(m/s)",
    Contours → 10, Title → "Vg along front", ColorFunction → "Rainbow" , Exclusions → None, Height → 400],
   ColorbarPlot[Evaluate[1000 Ug[#1, #2] /. param] &, {-1000, 1000}, {0, 13.5}, PlotType → "Contour",
    Axes → True, XLabel → "x (km)", YLabel → "z (km)", CLabel → "(m/s)", Contours → 10,
    Title → "Ug across front", ColorFunction → "Rainbow" , Exclusions → None, Height → 400]}]]
```

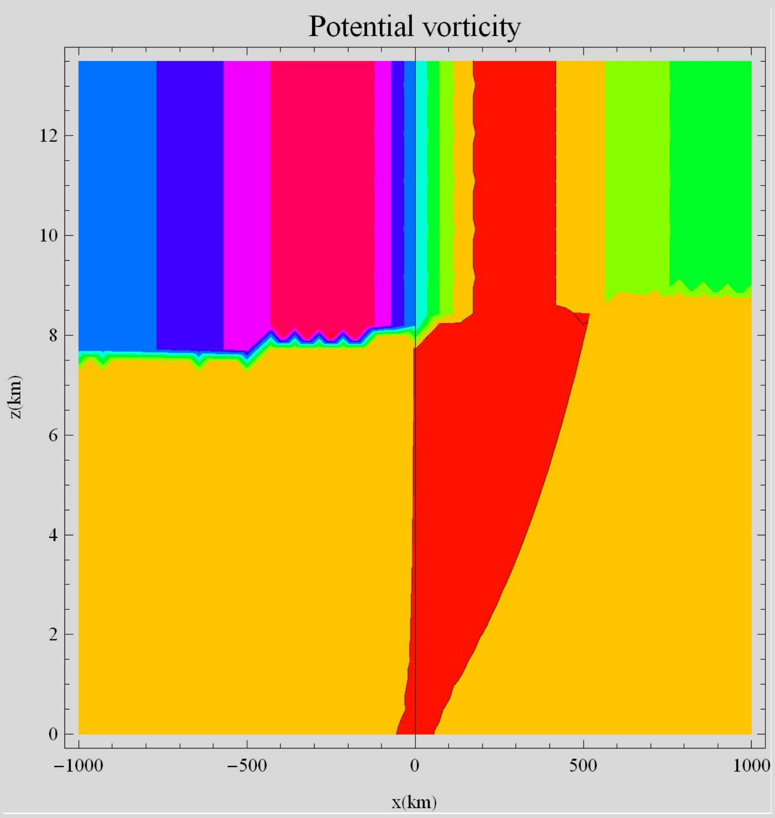
```
g5
```



Potential vorticity (x,z)-plane

```
g6 := Show[ContourPlot[Evaluate[PV[x, z] /.param], {x, -1000, 1000}, {z, 0, 13.5}, ContourStyle → None, ColorFunction → Hue,
   DisplayFunction → Identity, Exclusions → None], ContourPlot[Evaluate[PV[x, z] /.param], {x, -1000, 1000},
   {z, 0, 13.5}, Contours → {0.}, ContourShading → False, DisplayFunction → Identity, Exclusions → None],
  FrameLabel → {"x(km)", "z(km)"}, PlotLabel → "Potential vorticity", Axes → True]
```

```
g6
```

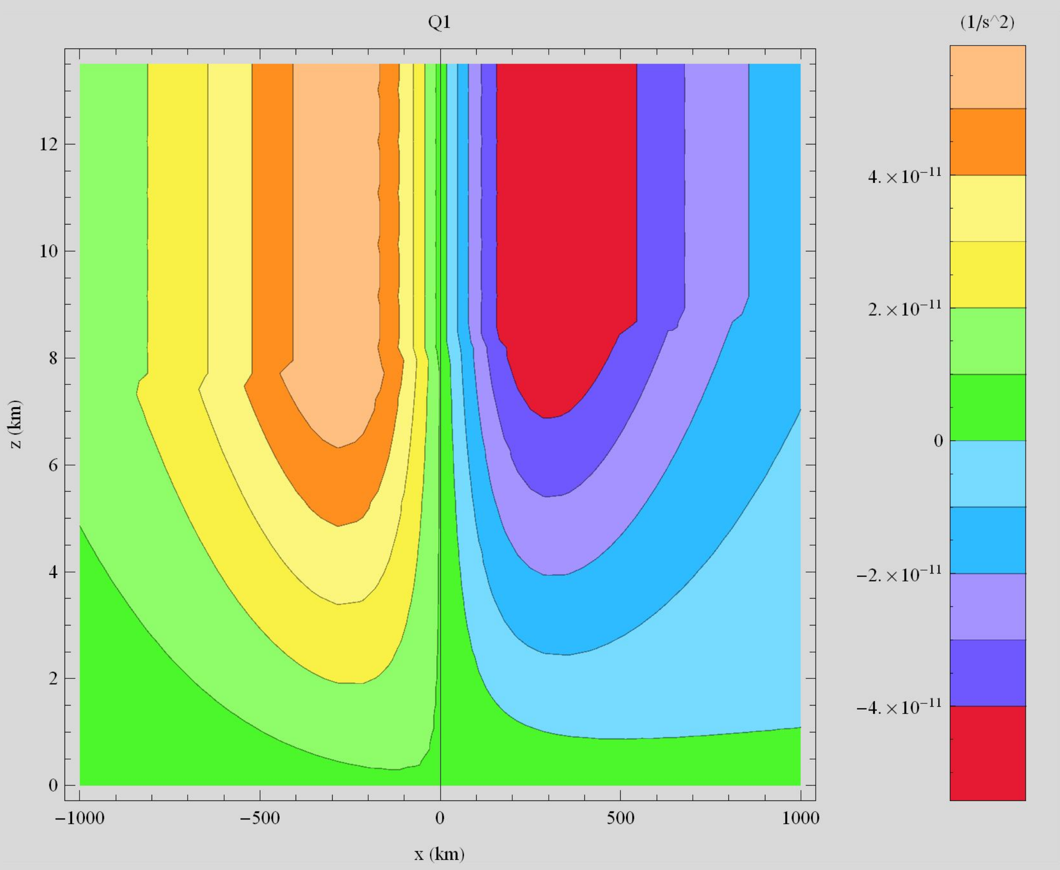


x-component of Q-vector (x,z)-plane

```
ContourPlot[Evaluate[Q1[x, z] /. param], {x, -2000, 2000}, {z, 0, 13.5}, Contours → 10,
  ColorFunction → "BrightBands", FrameLabel → {"x(km)", "z(km)"}, Axes → True, Exclusions → None] /. param
```

```
g7 := ColorbarPlot[Evaluate[Q1[#1, #2] /. param] &, {-1000, 1000}, {0, 13.5},
   PlotType → "Contour", Axes → True, XLabel → "x (km)", YLabel → "z (km)", CLabel → "(1/s^2)",
   Contours → 10, Title → "Q1", ColorFunction → "BrightBands" , Exclusions → None, Height → 400] /. param
```

```
g7
```



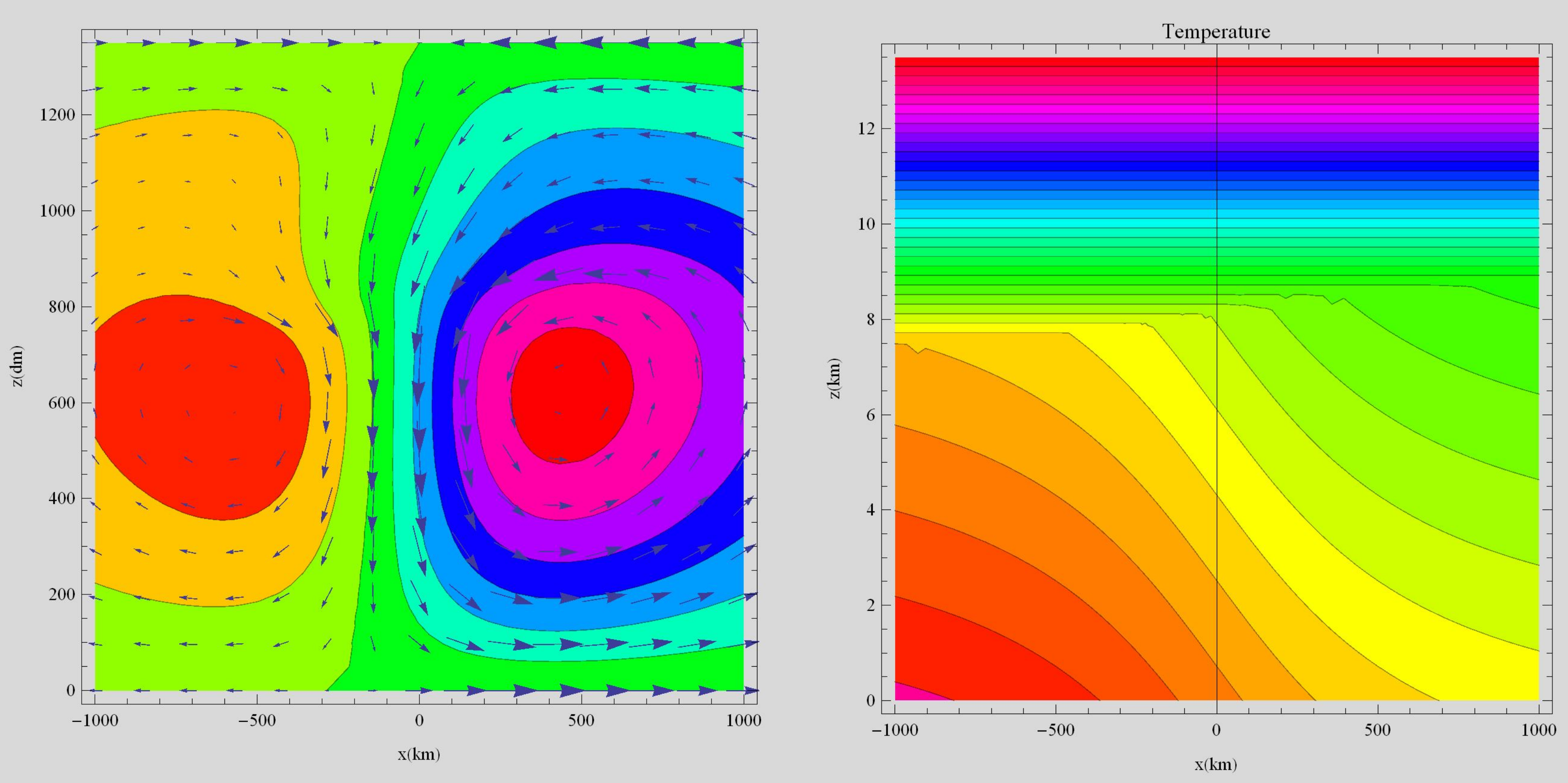
Solution of the QG Sawyer-Eliassen equation

```
SolveQGSE := Module[{}, num = 100; hstep = 2. Lx / (num + 1) /. param; vstep = H / (num + 1) /. param;
vars = Table[a[i, j], {j, num}, {i, num}];
hkern2 = {1, -2, 1} / hstep^2;
vkern2 = {1, -2, 1} / vstep^2;
pQ1[i_, j_] = Simplify[N[Q1[-Lx + i hstep, j vstep] /. param]];
pNg2[i_, j_] = Simplify[N[Ng2[-Lx + i hstep, j vstep] /. param]];
   mNg2 = Table[pNg2[i, j], {j, num}, {i, num}];
mF2 = Table[N[f^2] /. param, {j, num}, {i, num}];
lap = mNg2 Map[ListCorrelate[hkern2, #, {2, 2}, 0] &, vars] +
     mF2 Transpose[Map[ListCorrelate[vkern2, #, {2, 2}, 0] &, Transpose[vars]]];
eqns = Thread[Map[Flatten, lap == Table[-2 pQ1[i, j], {j, num}, {i, num}]]];
{vec, mat} = CoefficientArrays[eqns, Flatten[vars]];
sol2 = Partition[LinearSolve[mat, -vec], num]; // Timing
 ]
```

```
MakePsi := Module[{}, ZeroFrame = Prepend[Append[#1, 0], 0] &;
  sol = Transpose[ZeroFrame /@ Transpose[ZeroFrame /@ sol2]];
  Ψ = ListInterpolation[Transpose[sol], {{-2000., 2000.}, {0, 1350.}}]]
```

```
g8 := ContourPlot[Ψ[x, z], {x, -1000., 1000.}, {z, 0, 1350}, ColorFunction → Hue, DisplayFunction → Identity];
g9 :=
  VectorPlot[Evaluate[{1000 ∂z Ψ[x, z], - 1000 ∂x Ψ[x, z]}], {x, -1000., 1000.}, {z, 0., 1350}, DisplayFunction → Identity];
g10 := Show[GraphicsRow[{Show[{g8, g9}, FrameLabel → {"x(km)", "z(dm)"}], g3}]]
```

```
SolveQGSE;
MakePsi; g10
```



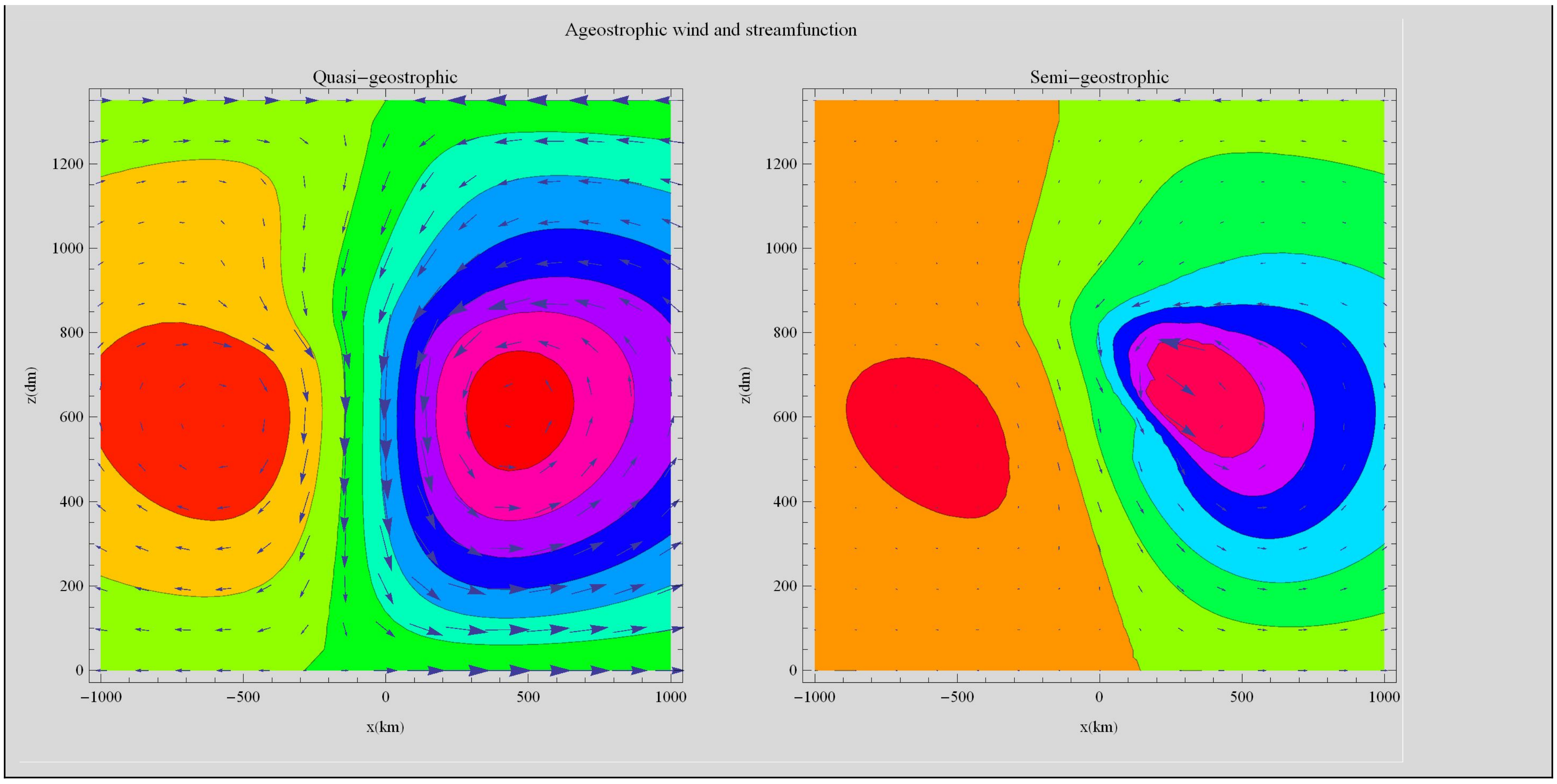
Solution of the full Sawyer-Eliassen equation

```
SolveSE := Module[{}, num = 100; hstep = 2. Lx / (num + 1) /. param; vstep = H / (num + 1) /. param;
vars = Table[a[i, j], {j, num}, {i, num}];
hkern2 = {1, -2, 1} / hstep^2; hkern1 = {1, 0, -1} / (2 hstep);
vkern2 = {1, -2, 1} / vstep^2; vkern1 = {1, 0, -1} / (2 vstep);
pQ1[i_, j_] = Simplify[N[Q1[-Lx + i hstep, j vstep] /. param]];
pNg2[i_, j_] = Simplify[N[Ng2[-Lx + i hstep, j vstep] + Ns2[-Lx + i hstep, j vstep] /. param]];
   pF2[i_, j_] = Simplify[N[F2[-Lx + i hstep, j vstep] /. param]];
   pS2[i_, j_] = Simplify[N[S2[-Lx + i hstep, j vstep] /. param]];
   mNg2 = Table[pNg2[i, j], {j, num}, {i, num}];
mF2 = Table[pF2[i, j], {j, num}, {i, num}];
mS2 = Table[pS2[i, j], {j, num}, {i, num}];
   SE = mNg2 Map[ListCorrelate[hkern2, #, {2, 2}, 0] &, vars] +
     mF2 Transpose[Map[ListCorrelate[vkern2, #, {2, 2}, 0] &, Transpose[vars]]] - 2 mS2
      Transpose[Map[ListCorrelate[vkern1, #, {2, 2}, 0] &, Transpose[Map[ListCorrelate[hkern1, #, {2, 2}, 0] &, vars]]]];
eqns = Thread[Map[Flatten, SE == Table[-2 pQ1[i, j], {j, num}, {i, num}]]];
{vec, mat} = CoefficientArrays[eqns, Flatten[vars]];
sol2 = Partition[LinearSolve[mat, -vec], num]; // Timing
]
```

```
MakePsif := Module[{}, ZeroFrame = Prepend[Append[#1, 0], 0] &;
  sol = Transpose[ZeroFrame /@ Transpose[ZeroFrame /@ sol2]];
  Ψf = ListInterpolation[Transpose[sol], {{-2000., 2000.}, {0, 1350.}}]]
```

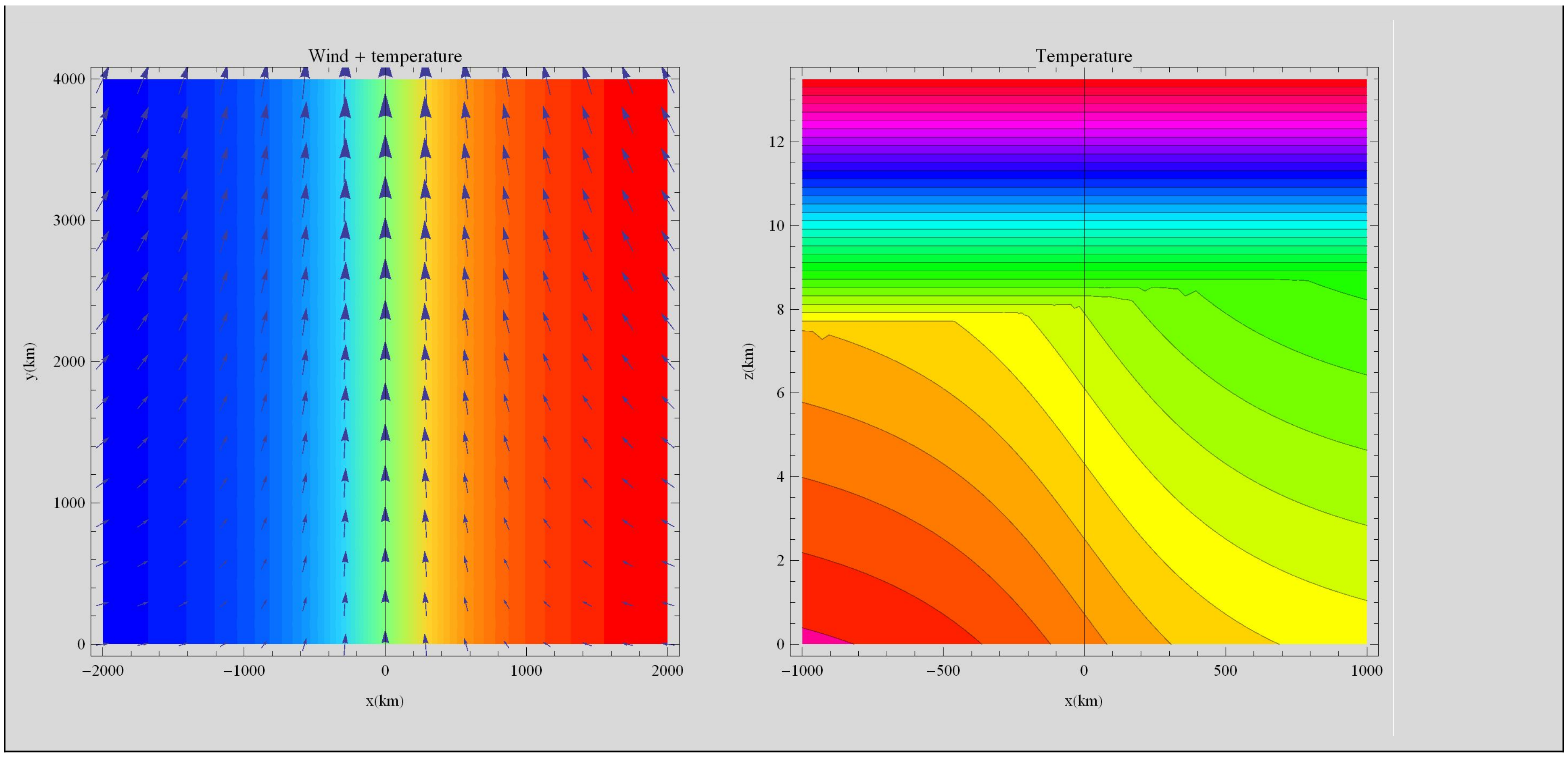
```
g8f := ContourPlot[Ψf[x, z], {x, -1000.`, 1000.`}, {z, 0, 1350`}, ColorFunction → Hue, DisplayFunction → Identity];
g9f := VectorPlot[Evaluate[{1000 ∂{z} Ψf[x, z], -1000 ∂{x} Ψf[x, z]}],
   {x, -1000.`, 1000.`}, {z, 0.`, 1350`}, AspectRatio → 1, DisplayFunction → Identity];
g11 := Show[GraphicsRow[{Show[{g8f, g9f}, FrameLabel → {"x(km)", "z(dm)"}], g3}]];
g12 := Show[GraphicsRow[{Show[{g8, g9}, FrameLabel → {"x(km)", "z(dm)"}, PlotLabel → "Quasi-geostrophic"],
    Show[{g8f, g9f}, FrameLabel → {"x(km)", "z(dm)"}, PlotLabel → "Semi-geostrophic"]}],
  PlotLabel → "Ageostrophic wind and streamfunction"]
```

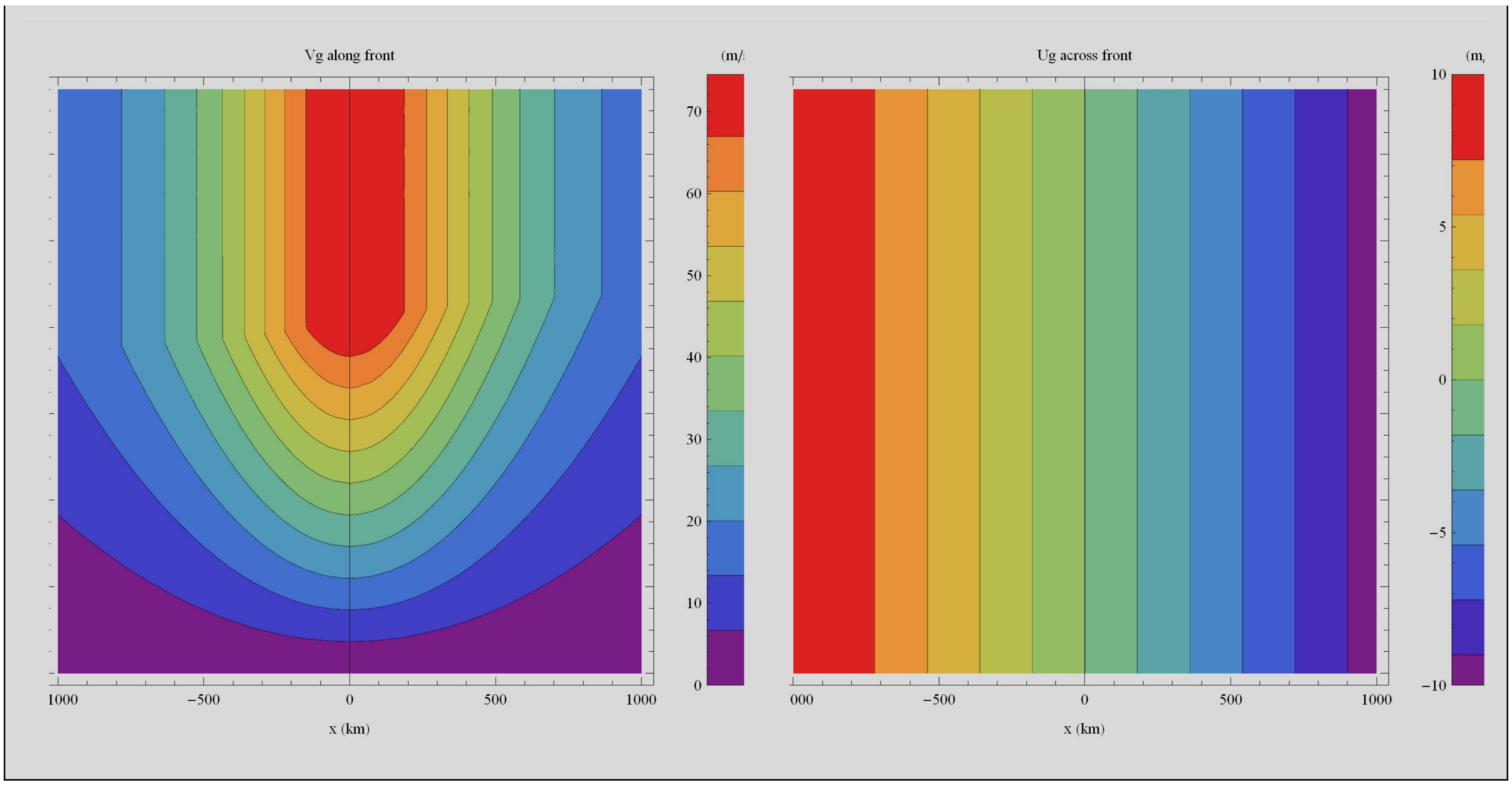
```
SolveSE;
MakePsif; g12
```

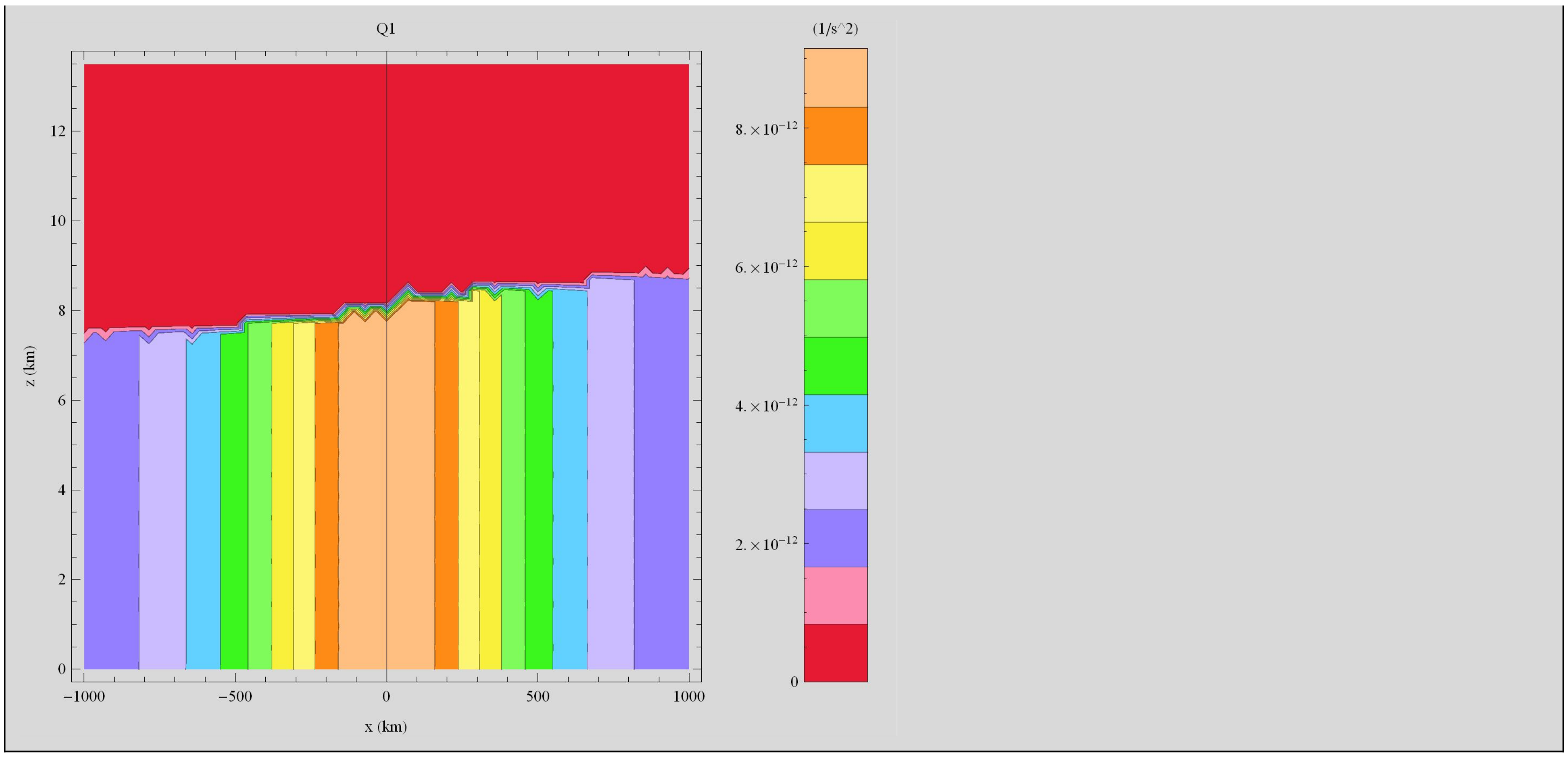


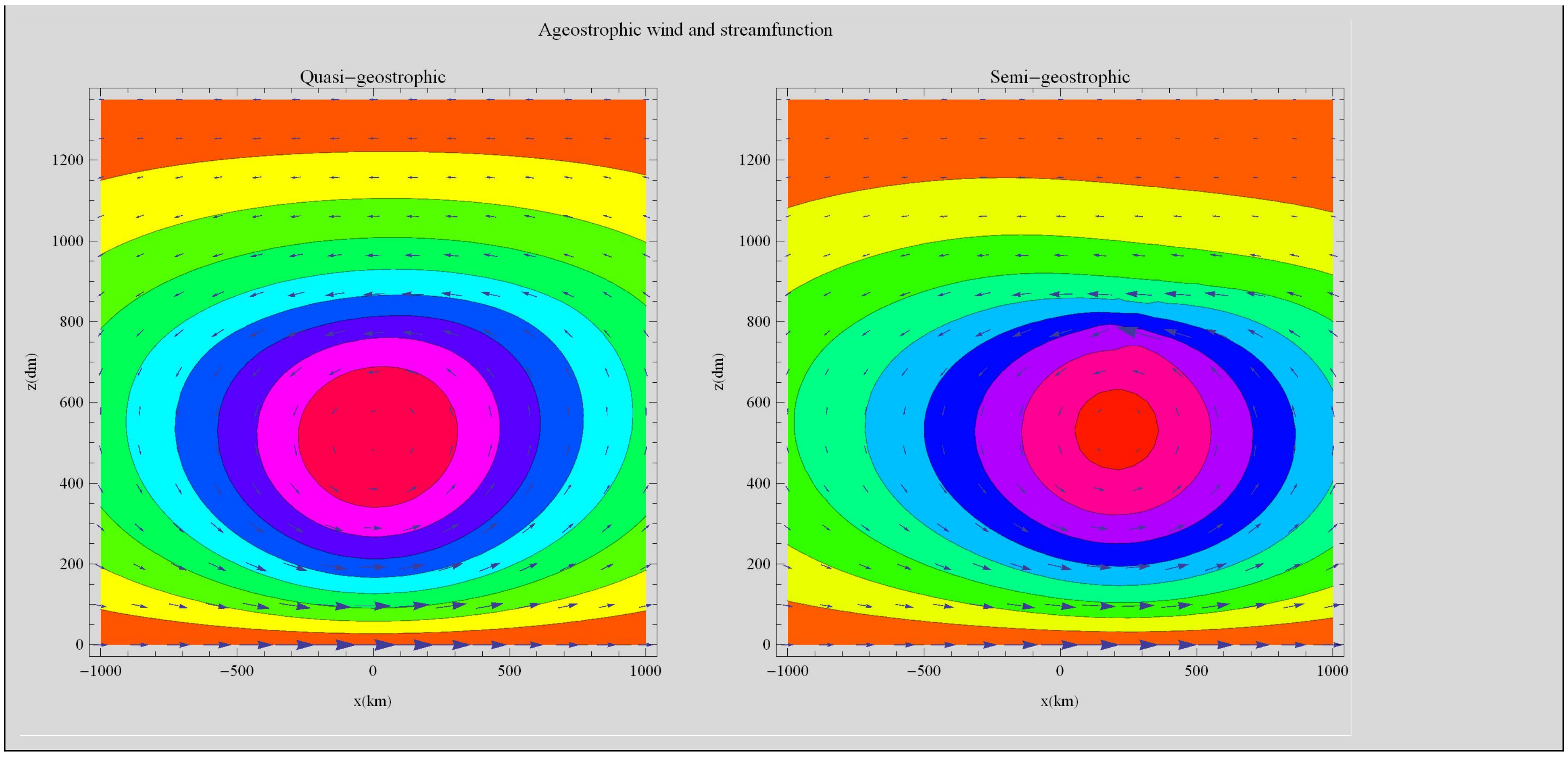
Cas de la confluence

```
param = param1;
g4
g5
g7
SolveQGSE; MakePsi;
SolveSE; MakePsif;
g12
```


Wind + temperature
y(km)
x(km)
Temperature
z(km)
x(km)

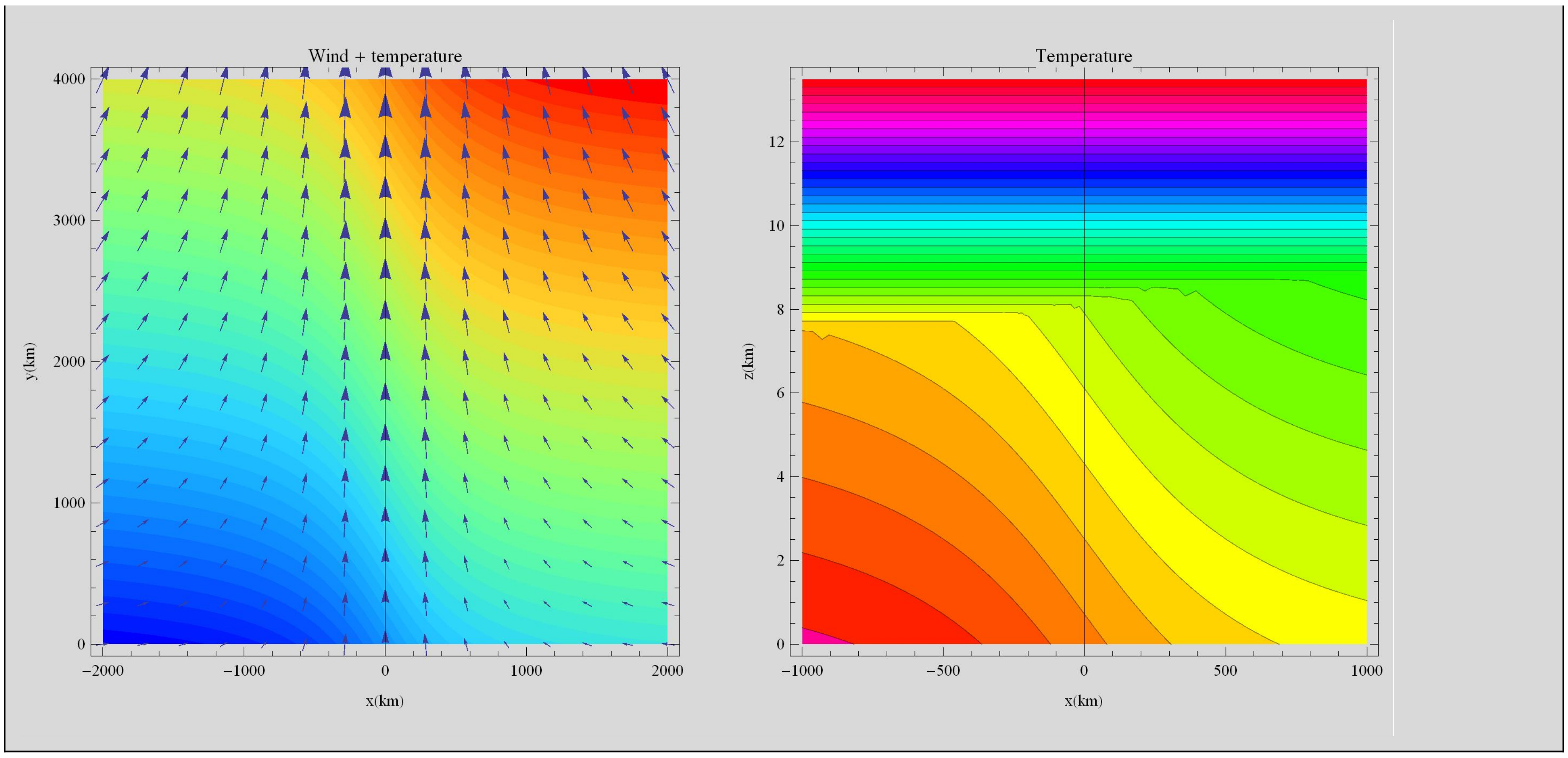

Vg along front
x (km)
Ug across front
x (km)

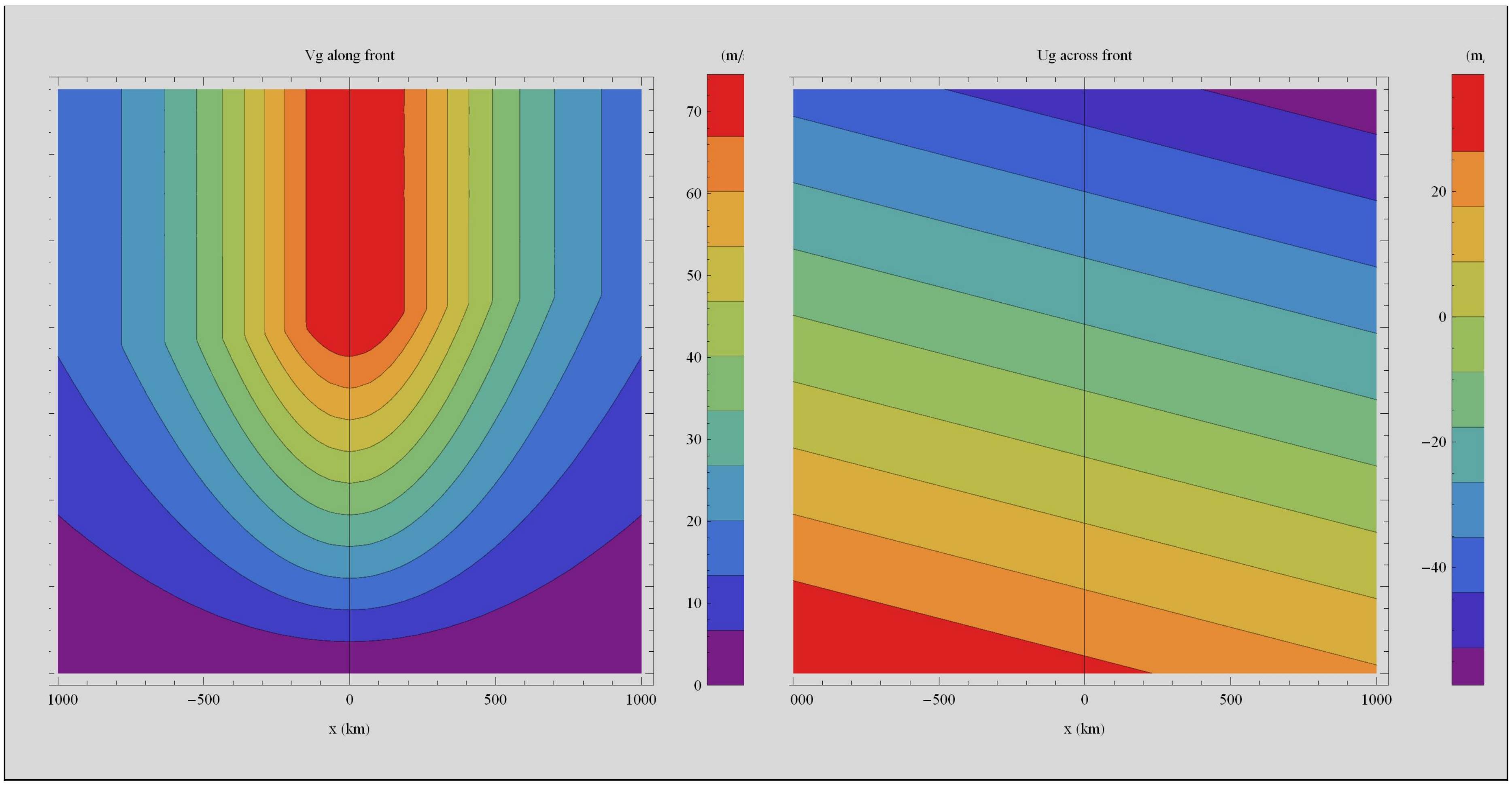


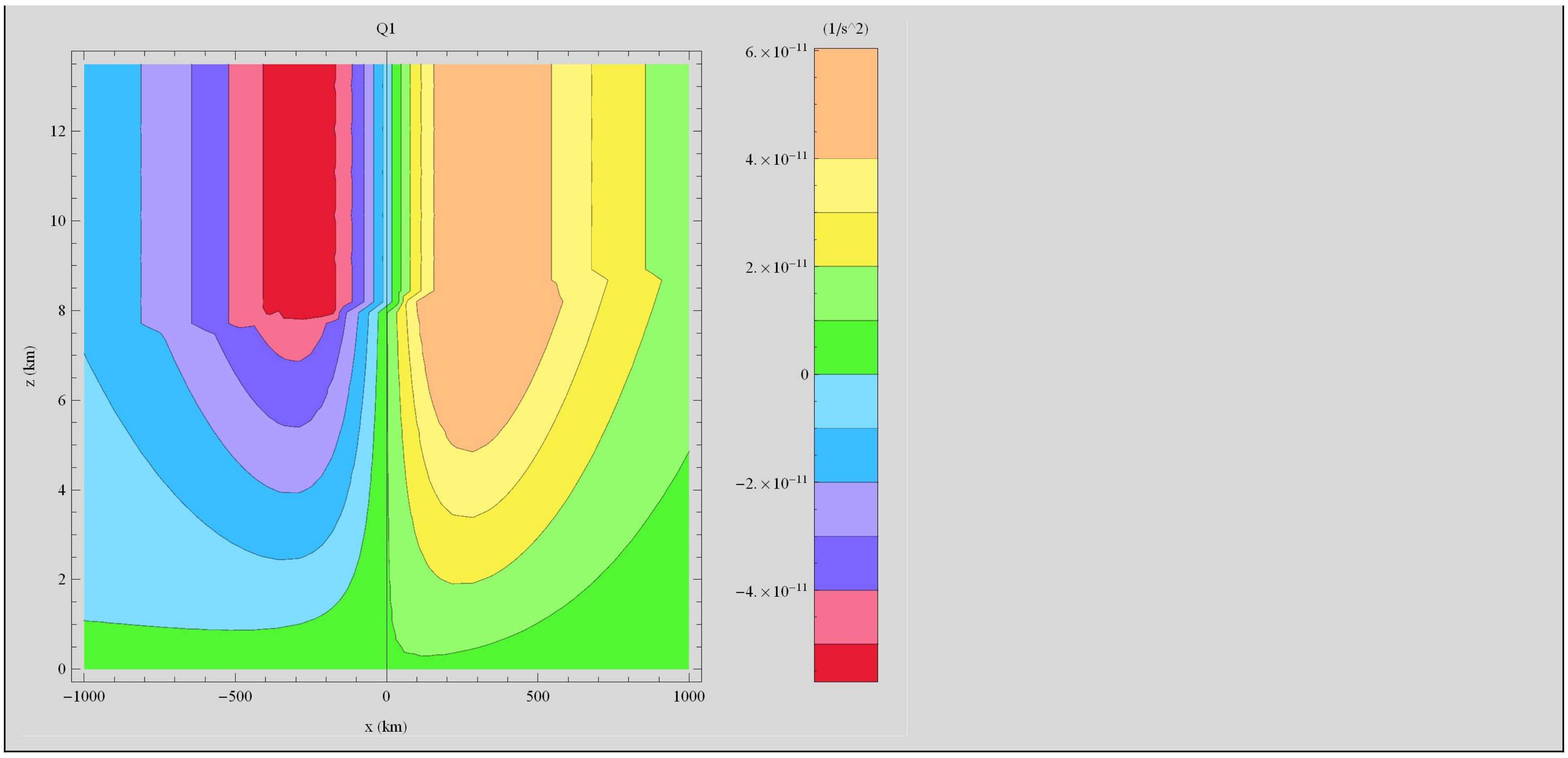


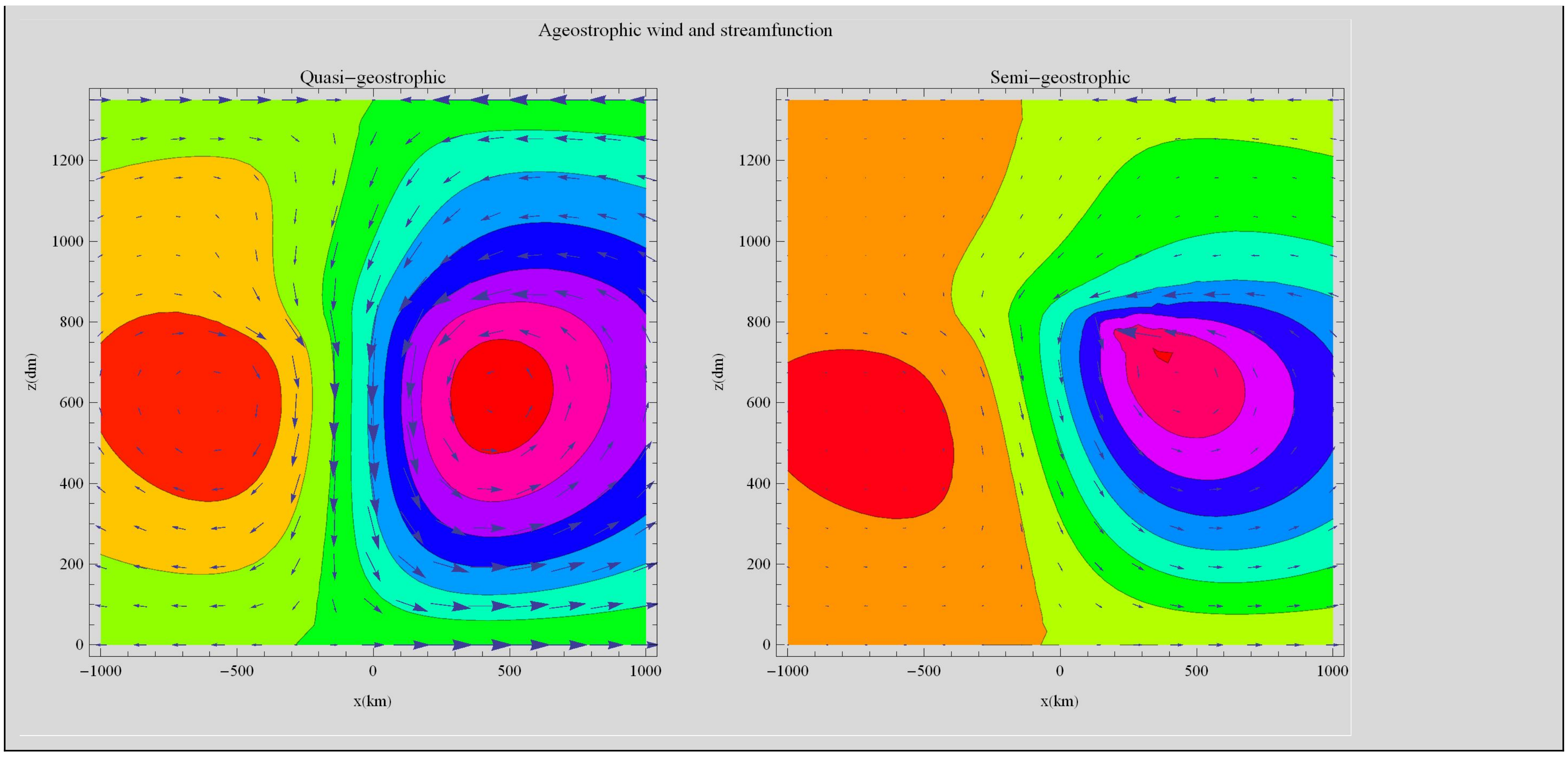
Cas de l'advection froide

```
param = param2;
g4
g5
g7
SolveQGSE; MakePsi;
SolveSE; MakePsif;
g12
```


Wind + temperature
y(km)
x(km)
Temperature
z(km)
x(km)

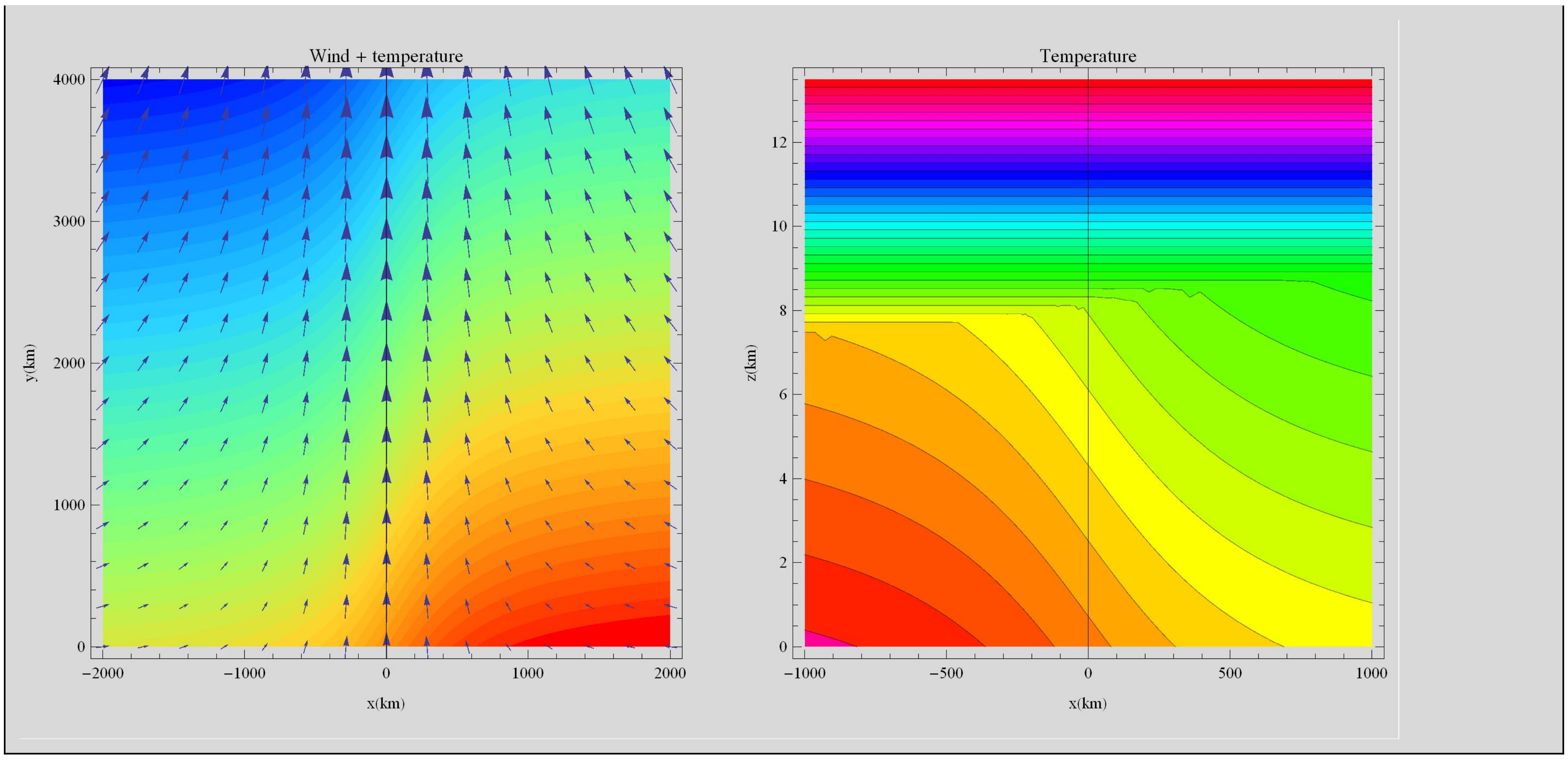

Vg along front
x (km)
(m/
Ug across front
x (km)
(m,



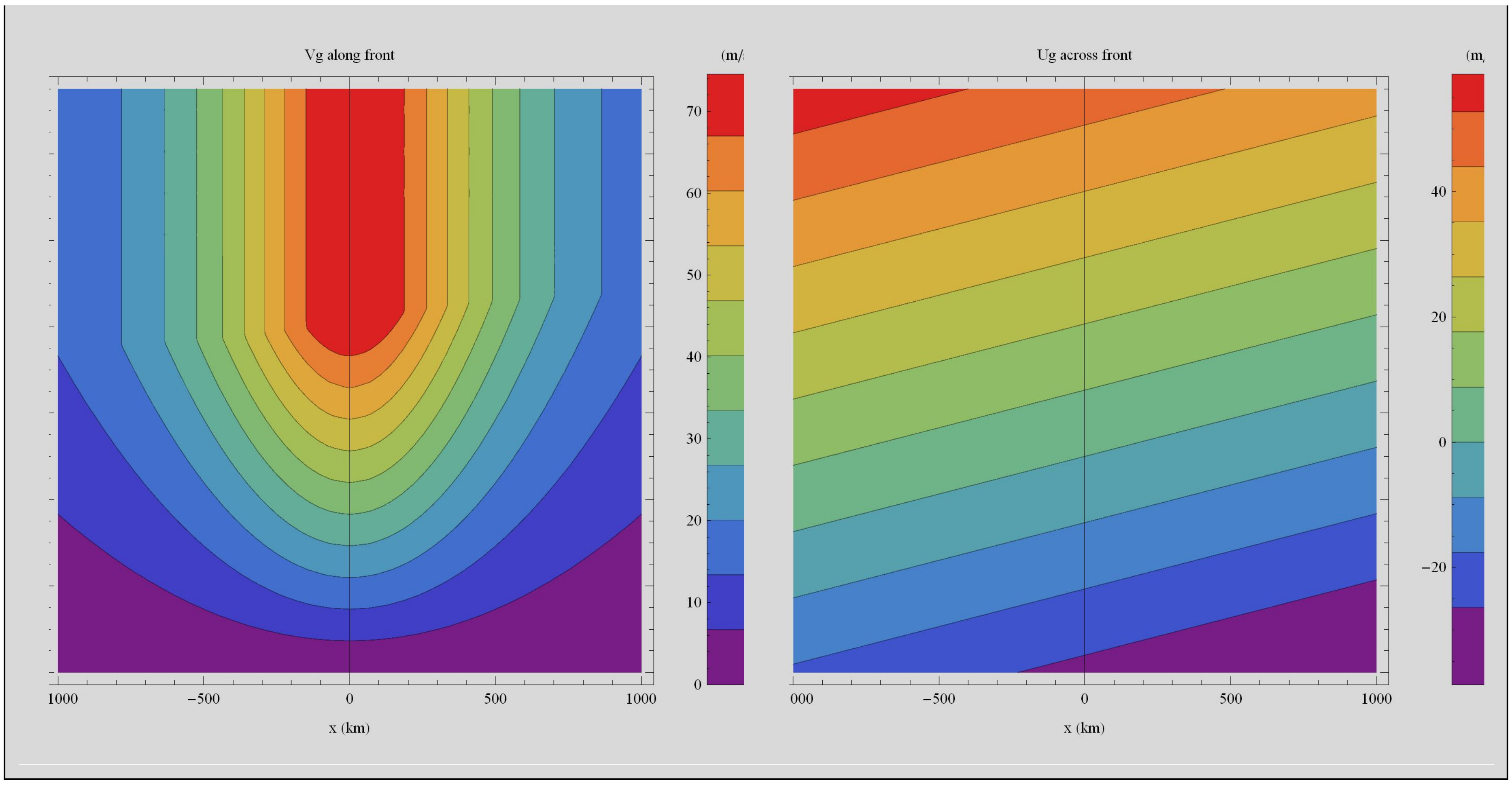


Cas de l’advection chaude

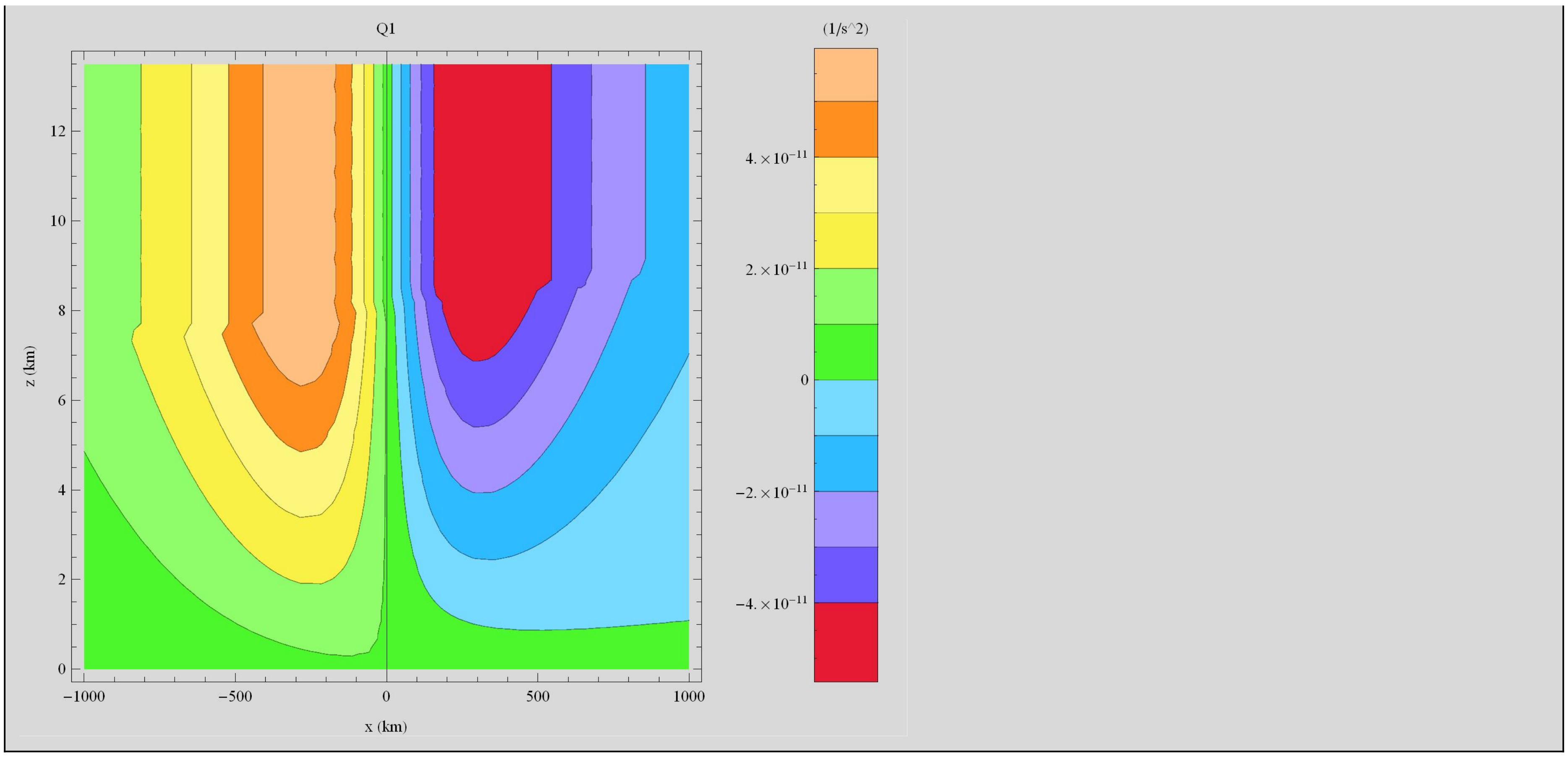
```
param = param3;
g4
g5
g7
SolveQGSE; MakePsi;
SolveSE; MakePsif;
g12
```

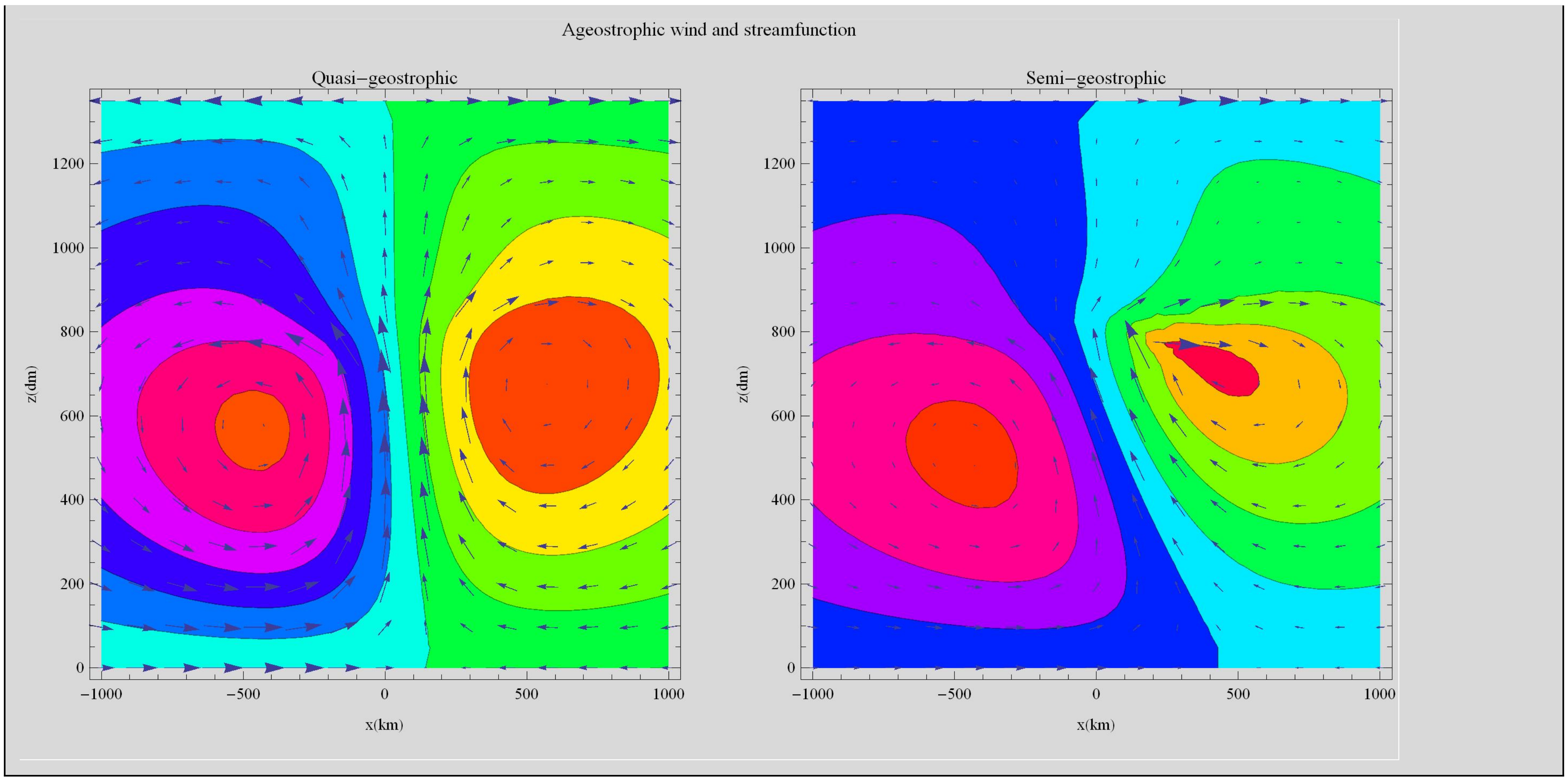


Wind + temperature
y(km)
x(km)
Temperature
z(km)
x(km)



Vg along front
x (km)
Ug across front
x (km)





Solution using NDSolve with bad boundary conditions

DRAFT AREA