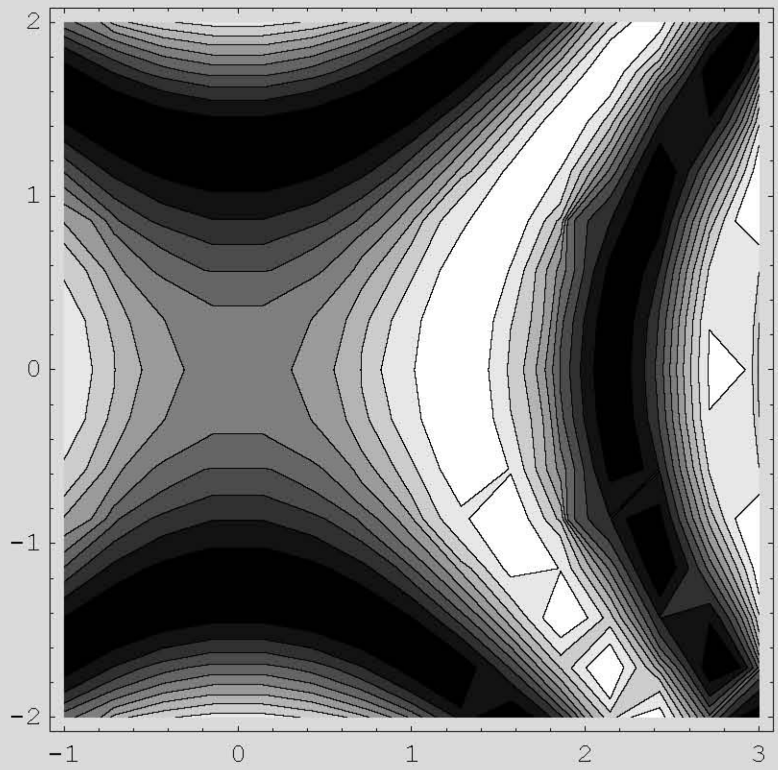
9.2 Μελετώντας τρισδιάστατα γραφικά στο επίπεδο

9.2.1 *Οι συναρτήσεις* Contour Plot *και* DensityPlot

Με την **ContourPlot**[f[x,y], {x,xmin,xmax},{y,ymin,ymax}] σχεδιάζουμε την f[x,y] πάνω στο επίπεδο Oxy, δίνοντας στο σημείο (x,y) ένα χρώμα (συνήθως απόχρωση του γκρίζου) που αντιστοιχεί στην τιμή f[x,y]. Τα σημεία που έχουν μεγαλύτερη τιμή f[x,y] είναι πιο φωτεινά ενώ αυτά που έχουν μικρότερη είναι πιο σκοτεινά. Π.χ

```
ContourPlot[Sin[x^2 - y^2], {x, -1, 3}, {y, -2, 2}];
```



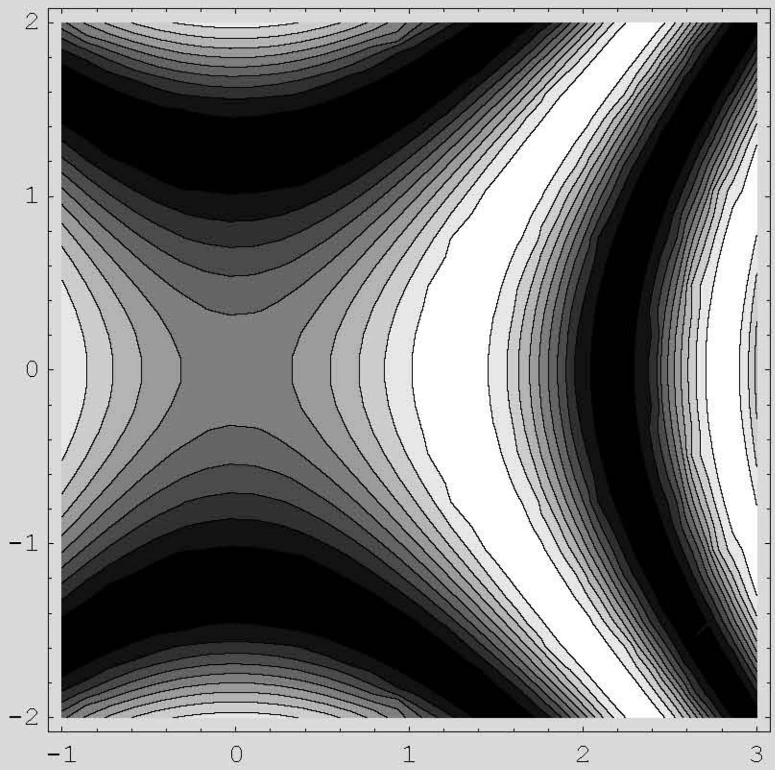
Παρατηρούμε 10 αποχρώσεις του γρί(το λευκό δεν το μετράμε, ουσιαστικά έχουμε μαζί με το λευκό 11). Η να το πούμε διαφορετικά: Το πεδίο τιμών(στον άξονα Oz) έχει χωριστεί σε 10+1 ίσου μήκους διαστήματα έστω Δ1,Δ2,..,Δ11. Κάθε διάστημα παίρνει ένα χρώμα του γκρί ξεκινώντας απο το μαύρο. Όσα σημεία (x,y) του επιπέδου απεικονίζονται(μέσω της f) **μέσα** στο ίδιο διάστημα Δ_i θα πάρουν την **ίδια** απόχρωση! 'Ετσι πάνω στο επίπεδο Oxy εμφανίζονται χρωματικές λωρίδες,τα **ισουψή επίπεδα,** τα οποία διαχωρίζονται μεταξύ τους απο κάποιες καμπύλες που λέγονται **ισουψείς**(Contours). Όλα τα σημεία μιας ισουψή καμπύλης παίρνουν **την ίδια ακριβώς τιμή** με την f ! Φυσικά επειδή υπάρχουν άπειρα σημεία (x,y) μέσα στο ορθογώνιο σχεδιασμού D={x,xmin,xmax}X{y,ymin,ymax}, το *Mathematica* θα διαλέξει δειγματολειπτικά λίγα σημεία απο το D και με βάση τις τιμές τους θα σχεδιάσει τις ισουψείς καμπύλες. Τα σημεία αυτά λέγονται Plot-Points. Φυσικά το αποτέλεσμα που παίρνουμε έχει όπως βλέπουμε πολλές ατέλειες. Για παράδειγμα οι ισουψείς καμπύλες δεν είναι όσο θα περιμέναμε **ομαλές**. Επίσης μπορούμε να παρατηρήσουμε ότι η λευκή λωρίδα στα δεξιά είναι στο κάτω μέρος της κομματιασμένη! Αυτό ωφείλεται κυρίως στο γεγονός ότι η προεπιλεγμένη τιμή του PlotPoints είναι 15. Οπότε απο το διάστημα D επιλέγονται 15X15 το πλήθος σημεία **που δεν είναι αρκετά** αν η f[x,y] έχει απότομες "λακούβες" και "λοφίσκους" στο D. Θα προσπαθήσουμε να αντιμετωπίσουμε αυτές τις ατέλειες. Ας δούμε όμως πρώτα, ποιές είναι οι επιλογές της ContourPlot:

```
Options[ContourPlot]

{AspectRatio → 1, Axes → False, AxesLabel → None,
 AxesOrigin → Automatic, AxesStyle → Automatic,
 Background → Automatic, ColorFunction → Automatic,
 ColorFunctionScaling → True, ColorOutput → Automatic,
 Compiled → True, ContourLines → True, Contours → 10,
 ContourShading → True, ContourSmoothing → True,
 ContourStyle → Automatic, DefaultColor → Automatic, Epilog → {},
 Frame → True, FrameLabel → None, FrameStyle → Automatic,
 FrameTicks → Automatic, ImageSize → Automatic, PlotLabel → None,
 PlotPoints → 15, PlotRange → Automatic, PlotRegion → Automatic,
 Prolog → {}, RotateLabel → True, Ticks → Automatic,
 DefaultFont :→ $DefaultFont, DisplayFunction :→ $DisplayFunction,
 FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```

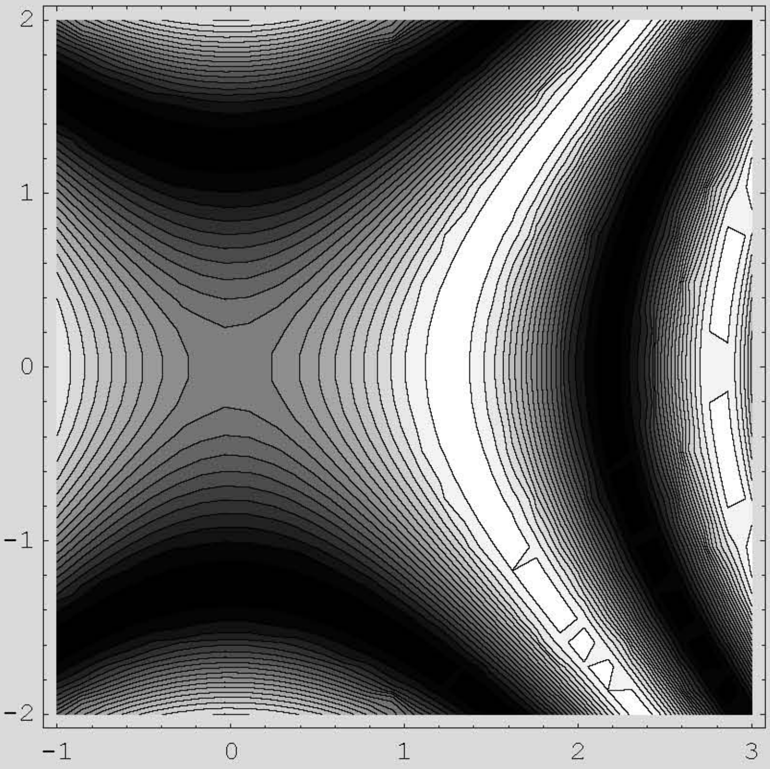
Αλλάζοντας κάποια απο τα παραπάνω χαρακτηριστικά μπορούμε να έχουμε ένα καλό αποτέλεσμα. Π.χ *μπορούμε να επιτρέψουμε στο Mathematica να κάνει καλύτερη δειγματοληψία παίρνοντας περισσότερα σημεία.* *Ως αποτέλεσμα θα έχουμε πιο ακριβείς ισουψείς καμπύλες.* Π.χ

```
gr = ContourPlot[Sin[x^2 - y^2],
   {x, -1, 3}, {y, -2, 2}, PlotPoints → 30];
```



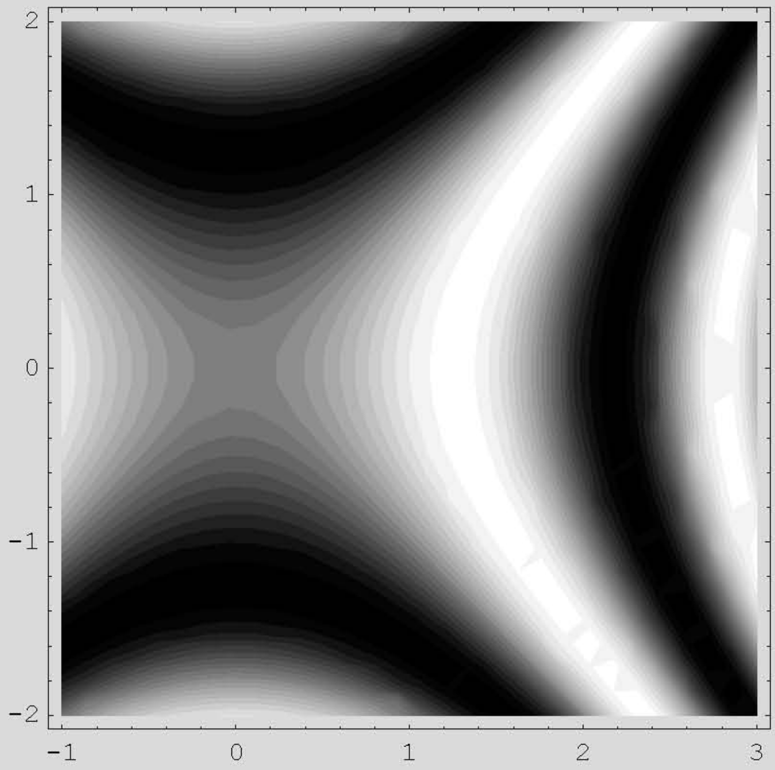
Μπορούμε επίσης να ζητήσουμε περισσότερες ισουψείς(και άρα ***περισσότερες αποχρώσεις****) μεγαλώνοντας την προεπιλεγμένη τιμή του* Contours *που είναι* 10(ή *ακριβέστερα* 10+1 *όπως προαναφέραμε*). Π.χ

```
gr1 = Show[gr, Contours → 20];
```



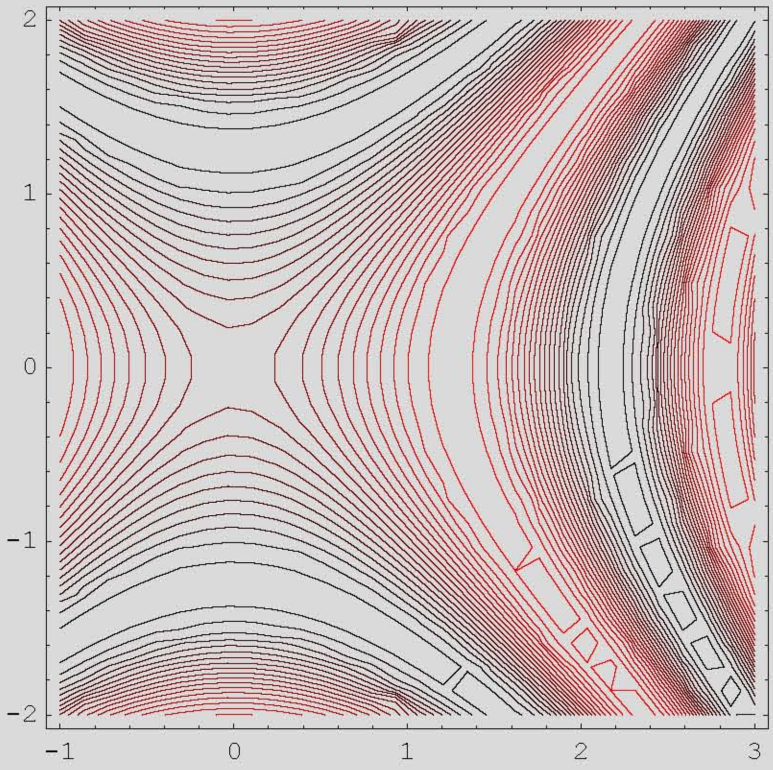
Προσέξτε ότι *αυξάνοντας το πλήθος των* Contours *χάνεται η ακρίβεια στο σχεδιασμό των ισουψών*!!! Άρα *θα πρέπει να αυξήσουμε ξανά τα* PlotPoints *για να πετύχουμε την ακρίβεια στον σχεδιασμό*! Αν *τώρα θέλουμε μόνο τις διαβαθμίσεις χωρίς τις ισουψείς θα γράφαμε* ContourLines->False *π.χ*

```
gr2 = Show[gr1, ContourLines → False];
```



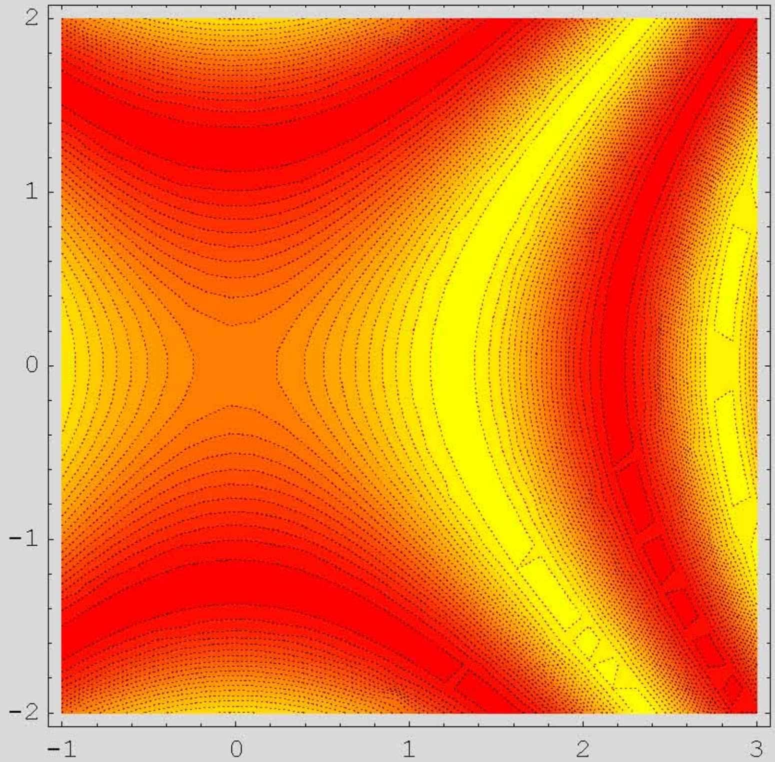
Σίγουρα πολύ πιο κατανοητό αποτέλεσμα! Μ*ερικές φορές δεν μας ενδιαφέρουν τόσο οι αποχρώσεις* όσ*ο οι ίδιες οι ισουψ*είς! Π*αρακάτω δίνουμε ένα τέτοιο παράδειγμα*. Με **ContourShading->False** *εξαφανίζουμε τις αποχώσεις ενώ με* **ContourStyle**->({RGBColor[#,0,0]}&)/@Range[0,0.95,.05] *δίνουμε* 20 ***διαφορετικές αποχρώσεις(όσα** είναι **και** τα* Contours στην g1) ***του κόκκινου*** (δηλ. RGBColor[0,0,0], RGBColor[.05,0,0],...,-RGBColor[0.95,0,0])*στις αντίστοιχες ισουψείς καμπ*ύλες.

```
gr3 = Show[gr1, ContourShading -> False,
  ContourStyle → ({RGBColor[#, 0, 0]} &) /@ Range[0, 0.95, .05]];
```



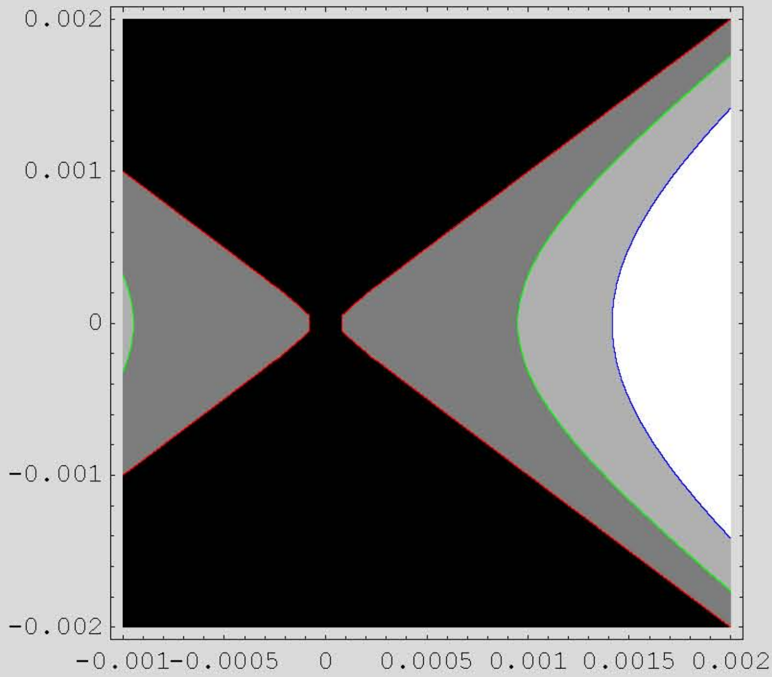
Εδώ *με* έ*ντονο κόκκινο* είναι *οι* ***ισουψείς*** *που βρίσκονται πιο ψηλά απο τις* άλλες. Θα *μπορούσαμε τώρα να* ***εμφανίσουμε*** *και τα* ***ισουψή*** *επίπεδα με διαβαθμίσεις του κίτρινου-κόκκινου(με την βοήθεια της* Color-Function→(RGBColor[1,#,0]&)) ***και τις*** ContourLines(ισουψεις καμπύλες) ***κόκκινες και διακεκομμένες*** *π.χ*

```
gr4 = Show[gr1, ContourShading → True,
  ColorFunction → (RGBColor[1, #, 0] &),
  ContourStyle → ({RGBColor[#, 0, 0], Dashing[{0.0015, 0.005}]} &) /@
    Range[0, 0.95, .05]];
```



Δεν πρέπει να ξεχάσουμε να αναφέρουμε την Contours->{z1,z2,z3,...} *με την οποία επιλέγουμε να μπούν ισουψείς μόνο στις συγκεκριμμένες τιμές του* z. Π.*χ θα προσπαθήσουμε να διερευνίσουμε την* f *κοντά στο σημείο* (0,0) *μελετώντας μόνο κάποιες θετικές ισουψείς με τιμές κοντά στο* f[x,y]=0 *π.χ* Contours→{4 10^{-9}, 9 10^{-7}, 2 10^{-6}} *Για ακρίβεια μεγαλώνουμε και το πλήθος των δειγματολειπτικών σημείων*(PlotPoints→40)

```
gr5 =
  ContourPlot[Sin[x^2 - y^2], {x, -0.001, 0.002}, {y, -0.002, 0.002},
   Contours → {4 10^-9, 9 10^-7, 2 10^-6}, PlotPoints → 40, ContourStyle →
    {{RGBColor[1, 0, 0]}, {RGBColor[0, 1, 0]}, {RGBColor[0, 0, 1]}}];
```

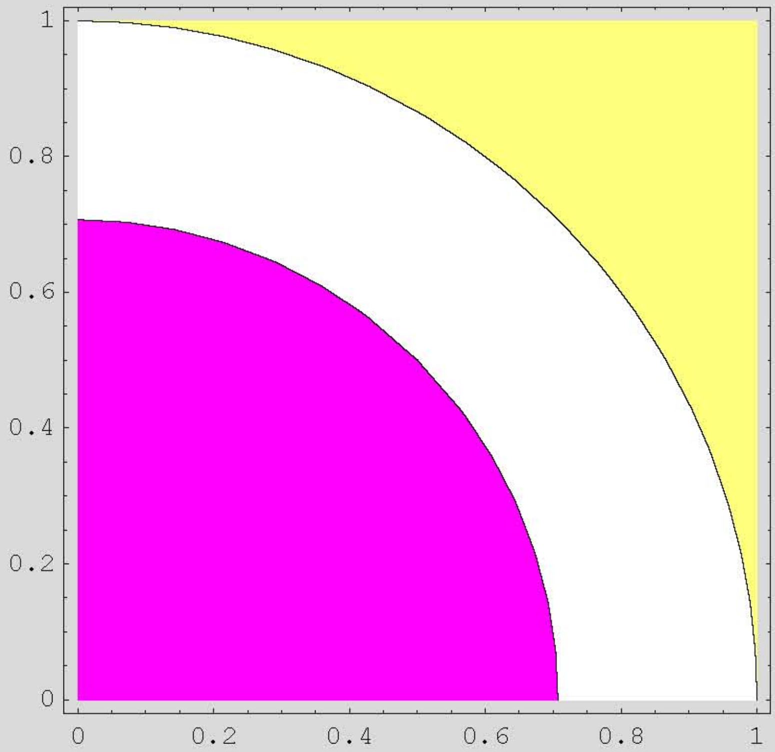


Με *μαύρο χρώμα* *είναι* όλες *οι τιμές της συνάρτησης* $<4\ 10^{-9}$, ***με ενδιάμεσο γκρί οι τιμές μεταξύ*** $4\ 10^{-9}$ *και* $9\ 10^{-7}$***κ.ο.κ***

Σχόλια για την ColorFunction. Η ColorFunction πρέπει να είναι κάποια συνάρτηση που να περιέχει εντολές χρωμάτων. Ο σκοπός της είναι να χρωματίσει με κάποιο τρόπο τα ισουψή επίπεδα. Ο τρόπος λειτουργίας της είναι ο εξής. Το πεδίο τιμών **κανονικοποιείται** στο διάστημα [0,1]. Έτσι για παράδειγμα, αν θέλουμε να σχεδιαστεί η f[x,y] για [x,y] σε κάποιο διάστημα του R^2, τότε υπολογίζεται η μέγιστη τιμή max και η ελάχιστη τιμή min της f στο διάστημα αυτό. Στη συνέχεια το max θεωρείται ως η μονάδα(1) και το min ως το μηδέν(0) και όλες οι άλλες τιμές της συνάρτησεις αντιστοιχίζονται γραμμικά σε τιμές μεταξύ του 0 και του 1. Ας κάνουμε ένα παράδειγμα. Έστω η max=2 και min=1. Αν, για παράδειγμα, επιλέξουμε κάποια Contours στις τιμές 0.5 και 1 του πεδίου τιμών της f, τότε η πρώτη τιμή του Contours θα αντιστοιχηθεί στην 0.5/(2-0)=0.25 και η δεύτερη στην 1/(2--0)=0.5 δηλ. η 1 θεωρείται απο την ColorFunction ως 0.5 και η 0.5 ως 0.25. Οπότε για να ορίσουμε σωστά ενα style για την ColorFunction σε αυτό το συγκεκριμμένο παράδειγμα θα πρέπει να δώσουμε τρία διαφορετικά χρώματα σε τρεις διαφορετικές περιοχές: Για τα σημεία που οι τιμές τους z (δηλ. τα ύψη) βρίσκονται στο πιο βαθύ ισουψές επίπεδο δηλ. για εκείνα που (μετά την κανονικοποίηση)παίρνουν τιμές z< 0.25, για τα σημεία που βρίσκονται μεταξύ των δύο ισουψών καμπυλών δηλ. για τα 0.25<=z<0.5 και τέλος για τα σημεία με z>0.5. Κοιτάξτε προσεκτικά πως φτιάχνουμε ένα style για να χρωματίσουμε τα ισουψή επίπεδα στο παράδειγμα που ακολουθεί.

```
style = Which[0 <= # < 0.25, RGBColor[1, 0, 1], 0.25 <= # < 0.5,
   RGBColor[1, 1, 1], # >= 0.5, RGBColor[1, 1, 0.5]] &
ContourPlot[x^2 + y^2, {x, 0, 1}, {y, 0, 1},
 Contours → {0.5, 1}, ColorFunction → style]
```

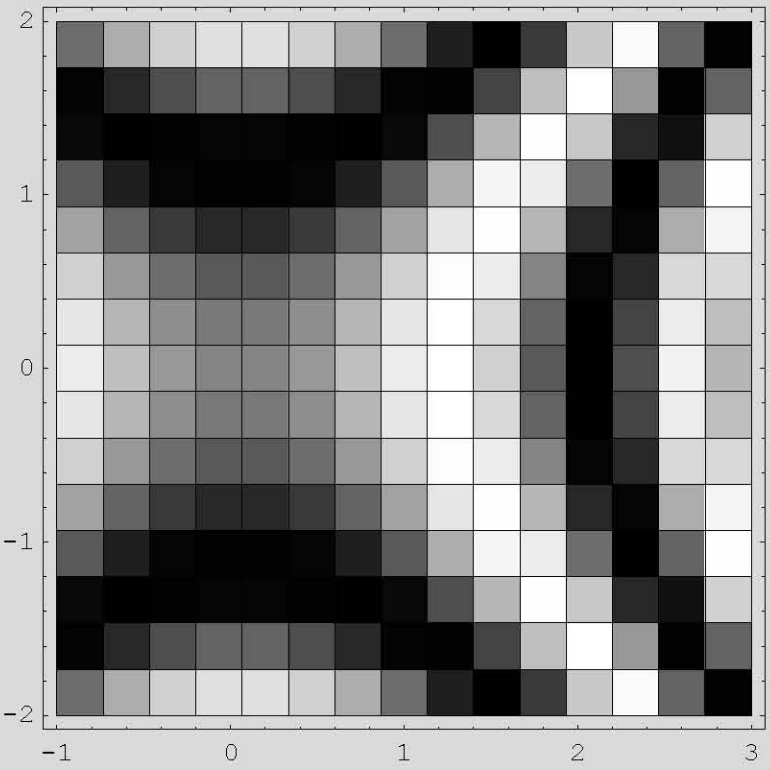
```
Which[0 ≤ #1 < 0.25, RGBColor[1, 0, 1], 0.25 ≤ #1 < 0.5,
  RGBColor[1, 1, 1], #1 ≥ 0.5, RGBColor[1, 1, 0.5]] &
```



- ContourGraphics -

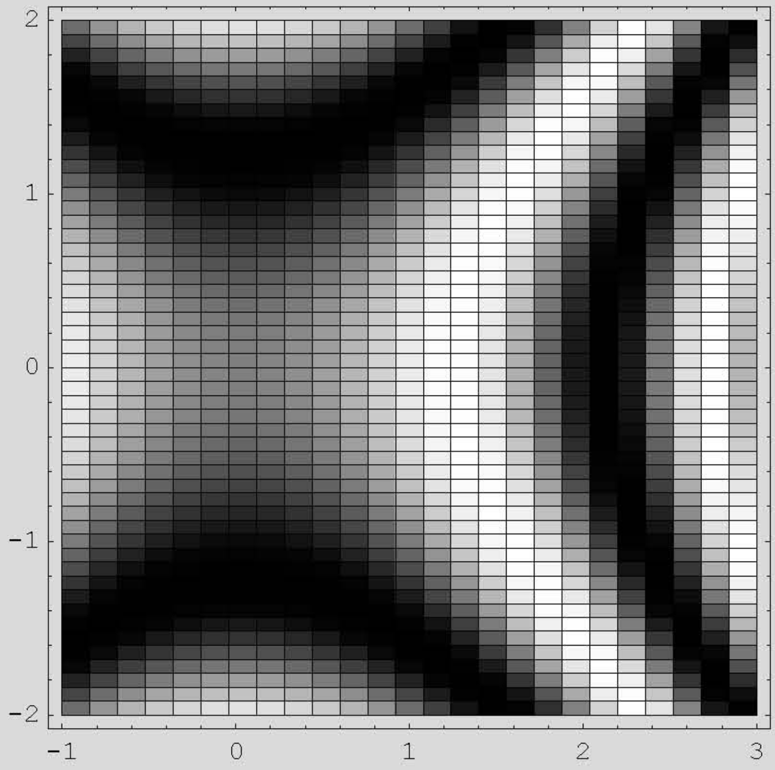
Η **DensityPlot** ***δεν προσπαθεί να σχεδιάσει κάποιες ισουψείς καμπύλες*** όπως ***η*** ContourPlot. Απλώς ***παράγει ένα πλέγμα***(mesh)***και κάποιες αποχρώσεις μέσα σε αυτό***. Η ***προεπιλεγμένες αποχρώσεις είναι του γκρί***. Αυτό το γεγονός φυσικά έχει και πλεονεκτήματα και μειονεκτήματα ως προς την ContourPlot. ουσιαστικά η DensityPlot[f[x,y],{x,xmin,xmax}, {y,ymin,ymax}]μας **προβάλει** κάθετα την επιφάνεια y=f[x,y] στο επίπεδο χωρίο [xmin,xmax]X[ymin,ymax] χωρίς να γίνεται κάποια προσπάθεια "ερμηνείας" της επιφάνειας.. ***Σκοτεινά γκρί χρησιμοποιούνται για βαθουλώματα της*** f[x,y] ***δηλ. για μικρές τιμές και ανοικτα γκρί για μεγάλες τιμές***. Π.χ

```
DensityPlot[Sin[x^2 - y^2], {x, -1, 3}, {y, -2, 2}];
```



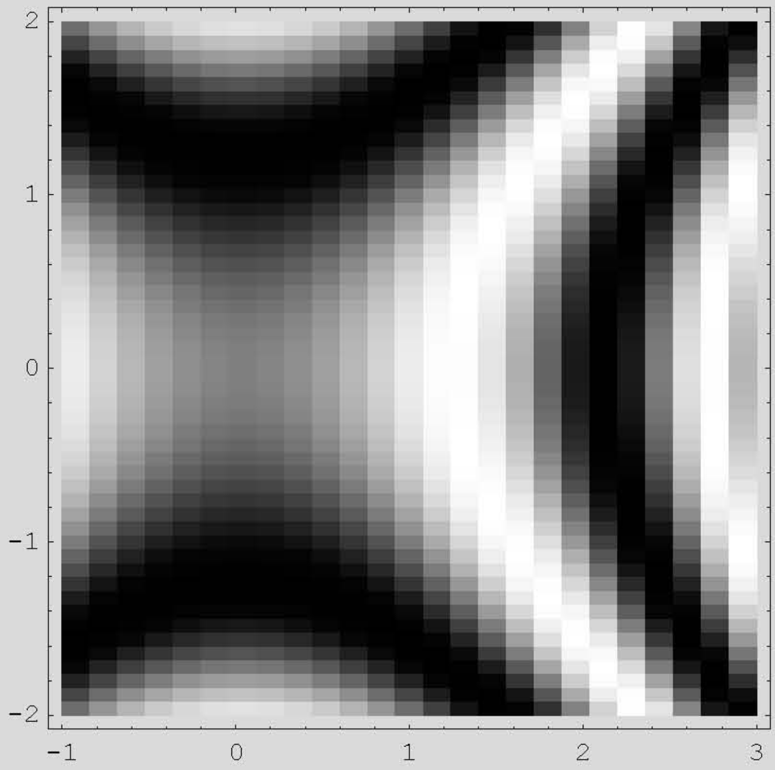
Όπως βλέπουμε χρησιμοποιούνται 15 PlotPoints *σε κάθε ένα απο τα διαστήματα των* x *και* y *αντίστοιχα*. Για *μεγαλύτερη ακρίβεια στα χρώματα μπορούμε να βάλουμε μεγαλύτερη τιμή* *π.χ* PlotPoints->{25,50}

```
gr1 = DensityPlot[Sin[x^2 - y^2],
  {x, -1, 3}, {y, -2, 2}, PlotPoints → {25, 50}];
```



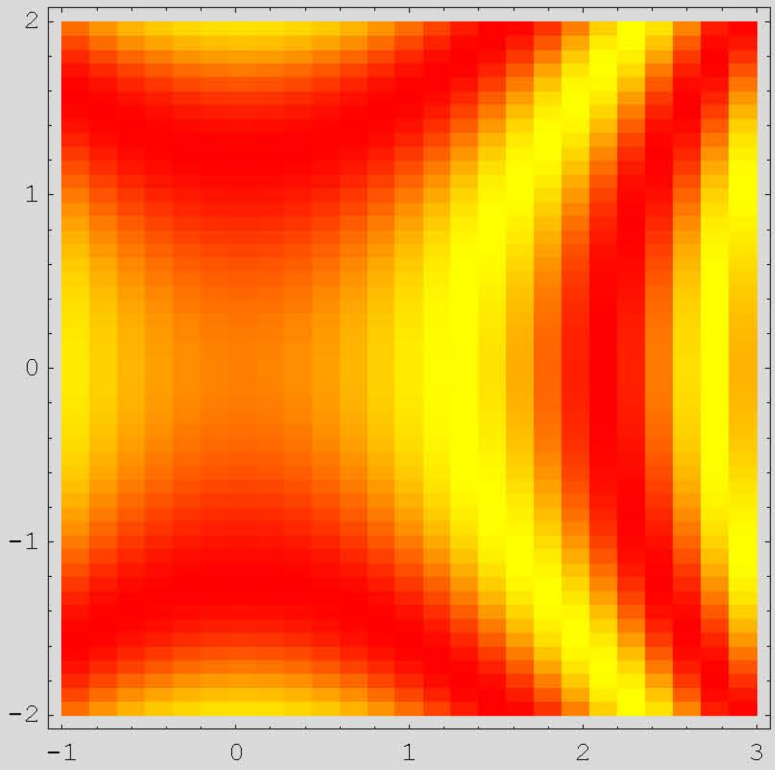
Όπως βλέπουμε δεν γίνεται καμμία προσπάθεια να σχηματιστούν καποια ισουψή επίπεδα Απλώς *σε κάθε δειγματοληπτικό σημείο απο τα* 25 ×50 *υπολογίζεται η αντίστοιχη τιμή της* f *και στην συνέχεια αυτή μετατρέπεται σε μια απόχρωση του γκρί.* Με Mesh->False *μπορούμε να εξαφανίσουμε το πλέγμα και να μείνει μόνο η απόχρωση:*

```
gr2 = Show[gr1, Mesh → False];
```



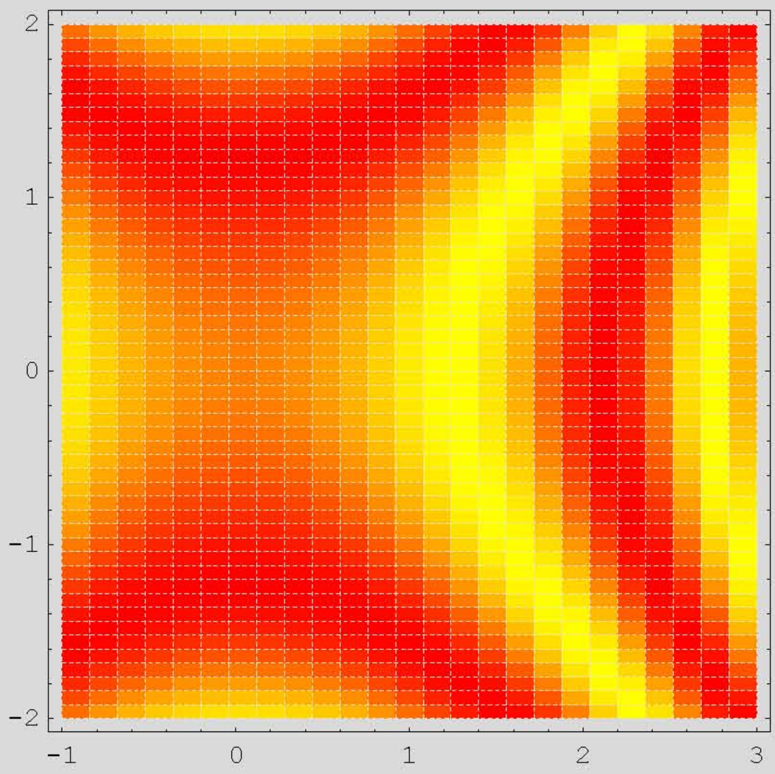
Με *την* ColorFunction *μπορούμε να αλλάξουμε κατά βούληση τις αποχρώσεις*:

```
gr3 = Show[gr2, ColorFunction → (RGBColor[1, #, 0] &)];
```



Αν θέλουμε να εμφανίζονται οποσδήποτε και το πλέγμα θα ήταν σκόπιμο να διαλέγαμε με την βοήθεια της MeshStyle *ένα διαφορετικό χρώμα γραμμών πλέγματος* ή *πιο λεπτές γραμμές πλέγματος* ή *διακεκομμένες* ή *κάποια απο τα προηγούμενα*:

```
gr4 =
  Show[gr1, MeshStyle → {GrayLevel[0.9], Dashing[{0.005, 0.005}]},
   ColorFunction → (RGBColor[1, #, 0] &)];
```



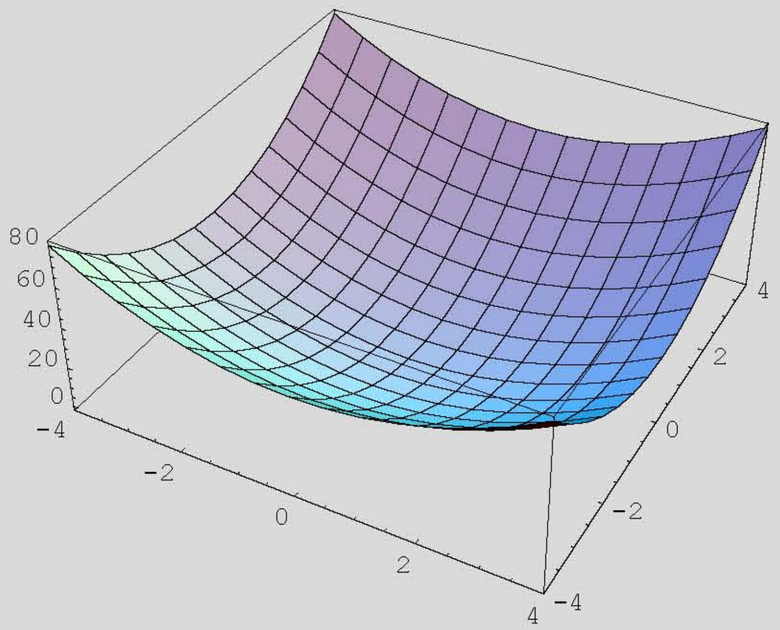
Επαναλαμβάνουμε ότι η DensityPlot μας σχεδιάζει μια επιφάνεια του χώρου όπως θα την έβλεπε ένας παρατηρητής που βρισκόταν ακριβώς απο πάνω της!

Ίσως θα αναρωτιέστε γιατί να χρησιμοποιήσουμε την DensityPlot αφού υπάρχει η ContourPlot. Η απάντηση είναι ότι υπάρχουν κακές περιπτώσεις που η ContourPlot στην προσπάθεια της να ζωγραφίσει τα ισουψή επίπεδα δεν βγάζει κάποια κάποιο κατανοητό γράφημα δηλ. μας επιστρέφει ανακριβές γράφημα οπότε θα πρέπει να αρκεστούμε στην DensityPlot ή σε συνδυασμό των δύο. Γενικά θα πρέπει να είμαστε σε θέση να παίρνουμε όλες τις πληροφορίες που μας χρειάζονται στην μελέτη μας κάνοντας κατάλληλο συνδυασμό όλων των δυνατοτήτων π.χ

Ασκηση: Να μελετήσουμε την συμπεριφορά της επιφάνειας με εξίσωση z= 2 x^2 + 3 y^2 για {x,-4,4} και {y,-4,4}.

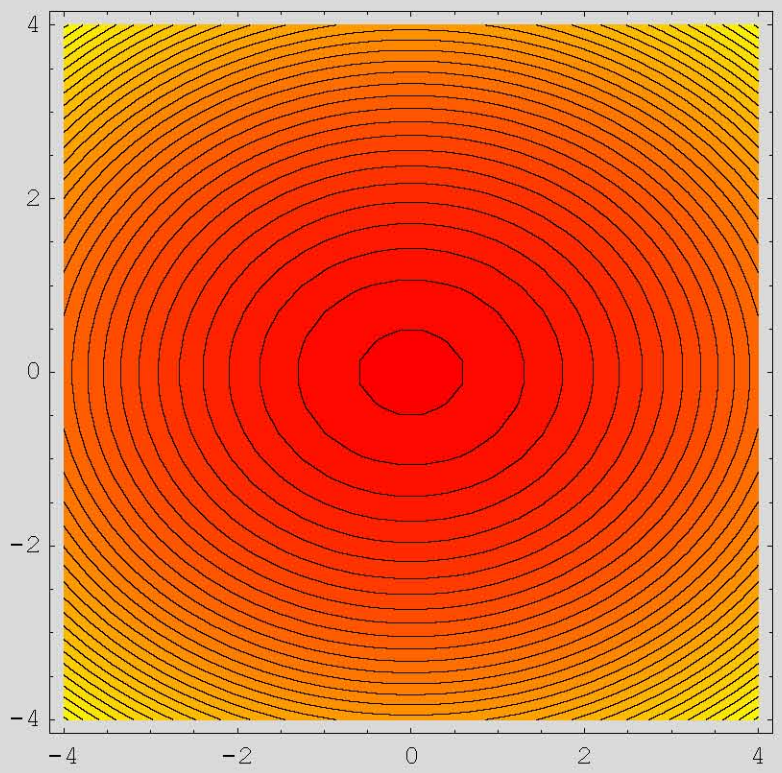
Λύση:Χρησιμοποιούμε την Plot3D σε συνδυασμό με την ContourPlot:

```
Plot3D[2 x^2 + 3 y^2 , {x, -4, 4} , {y, -4, 4}];
```



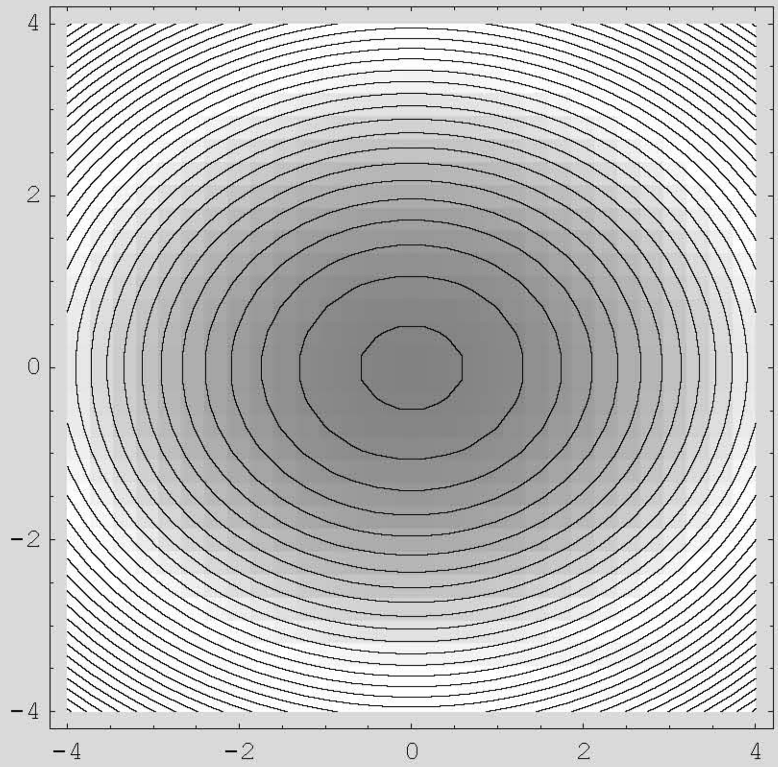
Βλέπουμε ότι υπάρχει ένα βαθούλωμα(ελάχιστη τιμή της z) αλλά δεν ξέρουμε που ακριβώς. Η ContourPlot θα βοηθήσει στον εντοπισμό του:

```
ContourPlot[2 x^2 + 3 y^2 , {x, -4, 4} ,
  {y, -4, 4}, ColorFunction → (RGBColor[1, #, 0] &),
  Contours → 30, PlotPoints → 30];
```



Δηλ. *είναι το σημείο* (0,0)! *Θα μπορούσαμε να χρησιμοποίησουμε και την* ContourPlot *σε συνδυασμό με την* DensityPlot *ως εξής*: Θέτουμε την DensityPlot κάτω απο την ContourPlot και στην πρώτη βάζουμε Mesh→False ενώ στην δεύτερη ContourShading→False(για να εμφανιστούν μόνο οι ισουψείς καμπύλες)

```
cg = ContourPlot[2 x^2 + 3 y^2 ,
   {x, -4, 4} , {y, -4, 4}, ContourShading → False,
   DisplayFunction → Identity, PlotPoints → 30, Contours → 30];
dg = DensityPlot[2 x^2 + 3 y^2 , {x, -4, 4} , {y, -4, 4},
   PlotPoints → 30, Mesh → False, DisplayFunction → Identity,
   ColorFunction → (GrayLevel[If[0.5 + # > 1, 1, 0.5 + #]] &)];
Show[dg, cg, DisplayFunction → $DisplayFunction];
```



Με If[0.5+#>1,1,0.5+#] *φωτίσαμε κατά* 0.5 *περισσότερο τα σκοτεινά μερη του* DensityPlot *για να έχουμε περισσότερη φωτεινότητα στα σημεία γύρω απο το* (0,0). Στο Show γράψαμε πρώτα την dg και μετά την cg για να μπεί η dg απο κάτω από την cg.

Τελειώνουμε εδώ με τις επιλογές της DensityPlot για να μπορέσετε να τις συγκρίνετε με τις αντίστοιχες της ContourPlot.

```
Options[DensityPlot]

{AspectRatio → 1, Axes → False, AxesLabel → None,
 AxesOrigin → Automatic, AxesStyle → Automatic,
 Background → Automatic, ColorFunction → Automatic,
 ColorFunctionScaling → True, ColorOutput → Automatic,
 Compiled → True, DefaultColor → Automatic, Epilog → {},
 Frame → True, FrameLabel → None, FrameStyle → Automatic,
 FrameTicks → Automatic, ImageSize → Automatic, Mesh → True,
 MeshStyle → Automatic, PlotLabel → None, PlotPoints → 15,
 PlotRange → Automatic, PlotRegion → Automatic,
 Prolog → {}, RotateLabel → True, Ticks → Automatic,
 DefaultFont :→ $DefaultFont, DisplayFunction :→ $DisplayFunction,
 FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```