Übung zur Meteorologischen Modellierung (Numerik): Das barotrope Modell

Neben seiner Bedeutung für konzeptionelle und analytische Studien der groß- und synoptisch-skaligen Zirkulation ist das barotope Modell von historischem Interesse für die numerische Meteorologie, da die ersten erfolgreichen Versuche zur numerischen Wettervorhersage mit einem barotropen Modell (der (quasi-)geostrophisch divergenzfreien Version) durchgeführt wurden (Charney, J. G., Fjortoft, R. and von Neumann, J. 1950: Numerical integration of the barotropic vorticity equation. *Tellus*, *2*, 237–54).

Im Folgenden sollen die Grundlagen für die Programmierung eines solchen Modells gegeben werden (in Verbindung mit dem in der Vorlesung Dargestellten). Ziel ist es dann, ein derartiges Modell zu programmieren und zu testen.

1. Die Gleichungen

Das barotrope Modell repräsentiert die Dynamik eines homogenen inkompressiblen Fluids, das sich im hydrostatischen Gleichgewicht befinden soll. Diese kann durch die Bewegungsgleichungen für die horizontalen Geschwindigkeiten (*u*,*v*) und die Kontinuitätsgleichung (Massenerhaltung) beschrieben werden. Die Bewegungsgleichungen für den reibungsfreien Fall lauten in kartesischen Koordinaten (und z-System):

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv &= -\frac{1}{\rho}\frac{\partial P}{\partial x} \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu &= -\frac{1}{\rho}\frac{\partial P}{\partial y} \end{aligned}$$

wobei f den Coriolis-Parameter, P den Druck und ρ die (konstante) Dichte bezeichnet. Hier wurde bereits von der Barotropie (die Geschwindigkeit ändert sich nicht mit der Höhe, also kein thermischer Wind) gebrauch gemacht (die vertikale Advektion von u und v ist entfallen).

Die Kontinuitätsgleichung bei konstantem ρ lautet:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

wobei w die Vertikalgeschwindigkeit bezeichnet. Wird die hydrostatische Grundgleichung

$$\frac{\partial P}{\partial z} = -g\rho$$

integriert, kann P/ρ in den Bewegungsgleichungen durch das Geopotential gh ersetzt werden. Also:

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv &= -\frac{\partial gh}{\partial x} \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu &= -\frac{\partial gh}{\partial y} \end{aligned}$$

Integration der Kontinuitätsgleichung von z=*0* (mit *w(0)=0*) bis h und Berücksichtigung von

$$w(h) = \frac{dh}{dt} = \frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} + v\frac{\partial h}{\partial y}$$

ergibt eine prognostische Gleichung für das, in den Bewegungsgleichungen benötigte Geopotential *gh*.

$$\frac{\partial gh}{\partial t} + u\frac{\partial gh}{\partial x} + v\frac{\partial gh}{\partial y} = -gh(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y})$$

Diese drei Gleichungen bilden ein geschlossenes System mit drei Unbekannten: Die sog. 'primitiven Gleichungen' des divergenten barotropen Modells (oder auch 'Flachwasser-Modell').

Obwohl auch dieses Modell numerisch gelöst werden kann, sollen im Folgenden einige vereinfachte Versionen hergeleitet werden, deren numerische Behandlung deutlich einfacher ist. Die Vereinfachungen ergeben sich aus weiteren Approximationen der Gleichungen (bisher wurde z.B. schon die hydrostatische Approximation verwendet mit der Folge, dass die eigentlich prognostische Gleichung für die vertikale Geschwindigkeit zu einer diagnostischen Beziehung zwischen Druck und Geopotential wurde). Zunächst wird der Coriolis-Parameter *f* durch eine lineare Entwicklung um eine Referenzbreite y_0 (mit $f(y_0)=f_0$) approximiert:

$$f = f_0 + \frac{df}{dy}y = f_0 + \beta y \quad \text{(Beta-Ebenen Approximation)}$$

Eine weitere Approximation mit guter Gültigkeit für die Zirkulation der mittleren Breiten ist die quasi-geostrophische Approximation der Gleichungen: Es kann gezeigt werden (Skalenanalyse; s. Vorlesung und Übungen Theoretische Meteorologie und Literatur), dass für kleine Rossby-Zahlen ($Ro=U/(Lf_0)$ << 1) in erster Näherung das geostrophische Gleichgewicht gilt. Wird nun die Strömung (u,v) und die Abweichung des Geopotentials zu einem mittleren Wert h_0 in einen geostrophischen (u_g,v_g,h_g) und einen (kleinen; O(Ro)) ageostrophischen (u_a,v_a,h_a) Teil aufgespalten ($u=u_g+u_a, v=v_g+v_a, h=h_0+h_g+h_a$), dies in die Flachwasser-Gleichungen eingesetzt, die Skalenanalyse durchgeführt, die Geostrophie eliminiert, die Divergenzfreiheit der geostrophischen Strömung berücksichtigt (die Divergenz steckt nun nur noch in der ageostrophischen Komponente) und alle Terme kleiner als O(Ro) vernachlässigt, so ergibt sich folgendes Gleichungssystem für die zeitliche Änderung der geostophischen Strömung:

$$\begin{aligned}
\frac{\partial u_g}{\partial t} + u_g\frac{\partial u_g}{\partial x} + v_g\frac{\partial u_g}{\partial y} - f_0 v_a - \beta y v_g &= -\frac{\partial gh_a}{\partial x} \\
\frac{\partial v_g}{\partial t} + u_g\frac{\partial v_g}{\partial x} + v_g\frac{\partial v_g}{\partial y} + f_0 u_a + \beta y u_g &= -\frac{\partial gh_a}{\partial y} \\
\frac{\partial gh_g}{\partial t} + u_g\frac{\partial gh_g}{\partial x} + v_g\frac{\partial gh_g}{\partial y} &= -gh_0(\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y})
\end{aligned}$$

Mit Einführung der geostrophischen Vorticity $\zeta_g = \frac{\partial v_g}{\partial x} - \frac{\partial u_g}{\partial y}$ können die ersten beiden Gleichungen (durch Ableitung nach y bzw. x und Subtraktion) auf die (quasi-geostropische) barotrope Vorticity-Gleichung reduziert werden. Es bleibt das System:

$$\frac{\partial \zeta_g}{\partial t} + u_g \frac{\partial \zeta_g}{\partial x} + v_g \frac{\partial \zeta_g}{\partial y} + \beta v_g = -f_0 \left(\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y}\right)$$

$$\frac{\partial gh_g}{\partial t} + u_g \frac{\partial gh_g}{\partial x} + v_g \frac{\partial gh_g}{\partial y} = -gh_0 \left(\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y}\right)$$

Mithilfe des geostrophischen Gleichgewichts kann die geostrophische Stromfunktion $\psi = gh_g/f_0$ eingeführt werden, für die gilt:

$$\begin{aligned} u_g &= -\frac{\partial \Psi}{\partial y} \\ v_g &= \frac{\partial \Psi}{\partial x} \\ \zeta_g &= \nabla^2 \Psi = \frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} \end{aligned}$$

Elimination der Divergenz der ageostrophischen Strömung aus den beiden Gleichungen und Einsetzen der geostrophischen Stromfunktion ergibt das quasigeostrophische divergente barotope Modell (quasigeostrophisches Flachwasser-Modell), das nun nur noch aus einer prognostischen Variablen, der Stromfunktion, besteht, die die Strömung (im Rahmen der Approximationen) vollständig beschreibt:

$$\frac{\partial (\nabla^2 - \lambda^{-2})\Psi}{\partial t} = -\frac{\partial \Psi}{\partial x}\frac{\partial (\nabla^2 - \lambda^{-2})\Psi}{\partial y} + \frac{\partial \Psi}{\partial y}\frac{\partial (\nabla^2 - \lambda^{-2})\Psi}{\partial x} - \beta \frac{\partial \Psi}{\partial x} = -J(\Psi, (\nabla^2 - \lambda^{-2})\Psi) - \beta \frac{\partial \Psi}{\partial x}$$

Hierbei bezeichnet λ den Rossby-Deformations-Radius ($\lambda = (gh_0)^{1/2}/f_0$) und $J(a,b) = (\partial a/\partial x\, \partial b/\partial y - \partial a/\partial y\, \partial b/\partial x)$ den Jacobi-Operator. Zu beachten sei hierbei noch, dass der Term $\lambda^{-2}\psi$ im Jacobi-Operator (die Advektion der Schichtdickenanomalie durch die geostrophische Strömung) nur formal in der Gleichung enthalten geblieben ist (es gilt $J(\psi, -\lambda^{-2}\psi) = 0$).

Eine noch tiefgreifendere Vereinfachung ergibt sich durch die vollständige Vernachlässigung der Divergenz der Strömung. Die Dynamik reduziert sich so auf das divergenzfreie barotrope Modell (die divergenzfreie barotrope Vorticitygleichung), das, wie oben erwähnt, das erste numerische Wettervorhersagemodell war:

$$\frac{\partial \zeta_g}{\partial t} + u_g \frac{\partial \zeta_g}{\partial x} + v_g \frac{\partial \zeta_g}{\partial y} + \beta v_g = 0$$

bzw.

$$\frac{\partial \nabla^2 \Psi}{\partial t} = -\frac{\partial \Psi}{\partial x}\frac{\partial \nabla^2 \Psi}{\partial y} + \frac{\partial \Psi}{\partial y}\frac{\partial \nabla^2 \Psi}{\partial x} - \beta \frac{\partial \Psi}{\partial x} = -J(\Psi, \nabla^2 \Psi) - \beta \frac{\partial \Psi}{\partial x}$$

Eine dritte, in den Anfängen der Wettervorhersage und für theoretische Studien häufig verwendete Version des barotropen Modells ist das sog. äquivalent-barotrope Modell: Um die

Barotropie-Bedingung (der Wind ändert sich nicht mit der Höhe) abzuschwächen aber trotzdem ein einfach zu behandelndes System zu erhalten wird hierbei angenommen, dass sich der Betrag der Strömung, jedoch nicht ihre Richtung, mit der Höhe ändern darf. Das vertikale Profil *A(p)* der Strömung sei dabei unabhängig vom Ort *(x,y)* und der Zeit *(t)*, Die horizontale Strömung *(u,v)* kann so als *A(p)(<u>(x,y,t),<v>(x,y,t))* geschrieben werden, wobei <> das vertikale Mittel bezeichnet (Es gilt <A> =1). Hier wurde (wie in meteorologischen Anwendungen üblich und um den Vergleich mit der Literatur zu erleichtern) in der vertikalen das p- anstelle des z-Systems verwendet (die Gleichungen sind hier aber nahezu identisch). Wird dieser Ansatz eingesetzt und die Variable $\psi^*=<A^2>\psi$ eingeführt, so ergibt sich für das quasi-geostrophisch äquivalent barotrope Modell (auf der β-Ebene) die Gleichung:

$$\frac{\partial(\nabla^2 - A(p_s)\lambda^{-2})\Psi^*}{\partial t} = -J(\Psi^*, \nabla^2\Psi^*) - \beta\frac{\partial\Psi^*}{\partial x}$$

Gültig ist dieses Modell für das Niveau *p** für das gilt: $A(p^*)=<A^2>$, das sog. äquivalent-barotrope Niveau. Typische Windprofile ergeben ein *p** von 600-500hPa. Dieses Niveau ist praktisch identisch mit dem divergenzfreien Niveau, d.h. der Schicht, in der die Divergenz im Mittel die kleinsten Werte annimmt.

Formal besteht der Unterschied zwischen den drei hier vorgestellten Versionen des quasi-geostrophischen barotropen Modells im Faktor $A(p_s)$ im Term, der die zeitliche Änderung des Geopotentials (also die Erzeugung von relativer Vorticity durch Stretching) beschreibt: $A(p_s)=0$ ergibt das divergenzfreie Modell, in dem die relative Vorticity lokal allein durch die Advektion von absoluter (relativer plus planetarer) Vorticity geändert wird. $A(p_s)=1$ liefert die Flachwassergleichungen; neben der Advektion von absoluter Vorticity liefert hier die Änderung der Schichtdicke durch die Divergenz/Konvergenz der Strömung einen Beitrag zur lokalen Vorticity-Tendenz ('Pirouetten-Effekt'). Der Parameter $A(p_s)$ bestimmt die Wirkung der Divergenz (gemittelt über die gesamte Mächtigkeit des Fluids) auf die Vorticity-Produktion in der äquivalent barotropen Schicht. In der Praxis kann durch die Wahl von $A(p_s)$ vor allem die Phasengeschwindigkeit der Rossby-Wellen im Modell modifiziert werden, was natürlich Konsequenzen für die Güte der Vorhersagen mit diesem Modell hat. So wurde dieser Parameter in den damaligen Wettervorhersagen als 'tuning' Parameter eingesetzt und in Hinblick auf eine möglichst gute Vorhersage eingestellt.

2. Die numerische Lösung

Natürlich können auch beim barotropen Modell unterschiedliche Verfahren (Gitterpunkts, Spektrale, Semi-Lagrange, fininite Elemente, etc.) verwendet werden, um das System numerisch zu lösen. Hier soll auf das 'klassische' Gitterpunkts-Verfahren zurückgegriffen werden. Exemplarisch behandelt werden soll das divergenzfreie barotrope Modell. Wie schon aus den Gleichungen zu vermuten, ist die Erweiterung auf das quasi-geostrophische Flachwasser-Modell bzw. das quasi-geostrophische äquivalent-barotrope Modell relativ einfach, während die Lösung der 'primitiven' Flachwasser-Gleichungen aufwändiger (und vom Ansatz verschieden) ist.

wie aus der Gleichung für das divergenzfreie barotrope Modell

$$\frac{\partial\nabla^2\Psi}{\partial t} = -\frac{\partial\Psi}{\partial x}\frac{\partial\nabla^2\Psi}{\partial y} + \frac{\partial\Psi}{\partial y}\frac{\partial\nabla^2\Psi}{\partial x} - \beta\frac{\partial\Psi}{\partial x} = -J(\Psi, \nabla^2\Psi) - \beta\frac{\partial\Psi}{\partial x}$$

zu ersehen, gilt es die zeitliche Entwicklung der Stromfunktion zu berechnen, die nur von der Stromfunktion selbst abhängt. Aus der Stromfunktion können dann diagnostisch alle relevanten Parameter (Vorticity, Geopotential-Anomalie, Wind, etc.) bestimmt werden. Die numerische Aufgabe ist also aus einem gegebenen Stromfunktionsfeld (und geeigneten Randbedingungen) das Stromfunktionsfeld zum nächsten Zeitpunkt (auf einer diskreten Zeitachse) zu berechnen (analog zur numerischen Lösung der Advektions- oder der Evolutions-Gleichung (Meteorologische Modellierung/Numerik erster Teil), wobei in diesem Fall die rechte Seite (der Antrieb) eine nichtlineare Funktion der vorherzusagenden Größe ist).

Konzeptionell ist das Modell wie folgt zu behandeln: Unter Vorgabe geeigneter Randbedingungen in der x,y-Ebene muss zunächst die Modellgleichung als Randwertaufgabe numerisch für $(\partial\psi/\partial t)_{t=0}$ gelöst werden. D.h. aus einem gegeben ψ Feld (und Bedingungen am äußeren Rand zur Bestimmung der räumlichen Ableitungen) wird der Antrieb (die rechte Seite) berechnet. Hierzu werden numerische Approximationen der räumlichen Ableitungen (bzw. des Jacobi-Operators) benutzt. Die Lösung der Poisson- (oder Helmholtz-) Gleichung

$$\nabla^2\Theta = G(x, y)$$

ergibt das $(\partial\psi/\partial t)_{t=\Delta t}$. Dann erfolgt eine zeitliche Extrapolation aus der $\psi_{t=\Delta t}$ folgt. Aus diesem Feld wird ein neuer Antrieb bestimmt, wiederum die Poisson-Gleichung gelöst und ψ in der Zeit extrapoliert so ergibt sich ψ zum Zeitpunkt $t=2\Delta t$. Das Verfahren wird auf diese Weise fortgeführt, bis der erwünschte Vorhersagezeitpunkt T erreicht ist.

Bevor das Modell numerisch mit einem Gitterpunkts-Verfahren gelöst werden kann, muss zunächst das Modellgebiet festgelegt und auf diesem ein entsprechendes Gitter definiert werden. Zusätzlich müssen die Werte für die freien Parameter (f_0, β, g, etc.) festgelegt werden. Da hier die Gleichungen in kartesischen Koordinaten und für die β-Ebene formuliert sind, kommt als Modellgebiet ein Kanal in zonaler Richtung (für meteorologische Anwendungen) oder ein rechteckiges Becken (Ozeanographie) in Frage. Als Gitter eignet sich ein reguläres kartesisches Gitter mit NxM Gitterpunkten, dessen Maschenweite in x- und y-Richtung nach den zu simulierenden Prozessen (bzw. von der Rechenkapazität des Computers) bestimmt wird. Die Maschenweite und die Prozesse bestimmen auch den längstmöglichen Zeitschritt dem das Modell integriert werden kann/sollte, da natürlich das Courant-Friedrich-Levy Kriterium für numerische Stabilität (s. Advektions-Gleichung) erfüllt werden muss. Beim divergenzfreien (und quasi-geostrophischen) Modell wird der Zeitschritt durch die Wanderung der (relativ langsamen) Rossby-Wellen bestimmt; bei einer Maschenweite von 300km ist deshalb, je nach Länge der längsten Rossby-Wellen und Geschwindigkeit des Grundstroms, ein Zeitschritt von bis zu ca. einer Stunde möglich (in den primitiven Gleichungen kämen wesentlich schnellere Schwere-Wellen hinzu; der Zeitschritt müsste also deutlich kürzer gewählt werden).

Da das Modellgebiet beschränkt ist, müssen noch geeignete Randbedingungen an den äußeren Rändern des Gebiets festgelegt werden, da ja Ableitungen zu bilden sind, für die Werte (Gitterpunkte) außerhalb (an den Rändern) des definierten Gebiets benötigt werden. Im Kanal werden typischerweise zyklische Randbedingungen in x-Richtung verwendet (also $\psi(0,y) = \psi(N,y)$ und $\psi(N+1,y) = \psi(1,y)$, wenn 1 der erste und N der letzte Gitterpunkt des Gitters ist und y alle Gitterpunkte in y-Richtung bezeichnet). An den meridionalen (y) Rändern (bzw. im Becken) wird typischerweise angenommen, dass die Normal-Komponente der Strömung am Rand verschwindet (kein Transport über den Rand). Das kann erreicht werden, indem ψ als konstant (=0) am Rand angenommen wird (also im Kanal $\psi(x,0)=0$, $\psi(x,M+1)=0$.).

Bemerkung: da für die Strömung nur die Stromfunktionsanomalien (Gradienten) zählt und das räumliche Mittel keine Rolle spielt, wird dieses zumeist auf 0 gesetzt.

Um den Modellablauf in die Tat umzusetzen müssen nun noch die numerischen Approximationen der einzelnen Komponenten festgelegt werden, also für die Berechnung des Jakobi-Operators, des Laplace-Operators, der horizontalen Ableitungen, der Lösung der Poisson- (Helmholtz-) Gleichung und der Extrapolation in der Zeit:

Der Jacobi-Operator

Es gibt verschiedene Möglichkeiten den Jacobi-Operator durch Differenzen zu approximieren. Diese folgen aus den unterschiedlichen analytischen Formulierungen:

$$\begin{aligned} J(a,b) &= \frac{\partial a}{\partial x}\frac{\partial b}{\partial y} - \frac{\partial a}{\partial y}\frac{\partial b}{\partial x} \\ &= \frac{\partial}{\partial x}(a\frac{\partial b}{\partial y}) - \frac{\partial}{\partial y}(a\frac{\partial b}{\partial x}) \\ &= \frac{\partial}{\partial y}(b\frac{\partial a}{\partial x}) - \frac{\partial}{\partial x}(b\frac{\partial a}{\partial y}) \end{aligned}$$

Zur Diskretisierung werden jeweils der Gitterpunkt selbst, 0=(i,j), sowie die 8 umliegenden Gitterpunkte (1-8) verwendet, die wie in der nachfolgenden Skizze gezeigt angeordnet sind:

6 = (i-1,j+1)	2 = (i,j+1)	5 = (i+1,j+1)
3 = (i-1,j)	0 = (i,j)	1 = (i+1,j)
7 = (i-1,j-1)	4 = (i,j-1)	8 = (i+1,j-1)

So ergeben sich folgende Differenzenoperatoren für die drei oben angegebenen Formulierungen:

$$\begin{aligned} J_1 &= \frac{1}{4\Delta x\Delta y}\{(a_1-a_3)(b_2-b_4)-(a_2-a_4)(b_1-b_3)\} \\ J_2 &= \frac{1}{4\Delta x\Delta y}\{a_1(b_5-b_8)-a_3(b_6-b_7)-a_2(b_5-b_6)+a_4(b_8-b_7)\} \\ J_3 &= \frac{1}{4\Delta x\Delta y}\{b_2(a_5-a_6)-b_4(a_8-a_7)-b_1(a_5-a_8)+b_3(a_6-a_7)\} \end{aligned}$$

Die wirklich verwendete Approximation (J) ergibt sich dann als Linearkombination von J_1, J_2 und J_3: $J = \alpha J_1 + \beta J_2 + \gamma J_3$. Arakawa (1966) konnte zeigen, dass die Kombination mit $\alpha=\beta=\gamma=1/3$ die geeignetste ist, da so die wichtigen Integral-Eigenschaften, Energie- und Enstrophie-Erhaltung, des Systems gewährleistet sind.

Der Laplace-Operator

Der Laplace-Operator, $\nabla^2 = \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, kann durch zentrierte Differenzen approximiert werden:

$$\nabla^2\Psi(i,j) = \nabla^2\Psi_0 = \frac{1}{\Delta x^2}(\Psi_1+\Psi_3-2\Psi_0) + \frac{1}{\Delta y^2}(\Psi_2+\Psi_4-2\Psi_0)$$

mit den Indizes wie für den Jacobi-Operator.

Horizontale Ableitungen

Auch für die Ableitungen in x- und y-Richtung $(\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ können zentrierte Differenzen verwendet werden:

$$\frac{\partial \Psi}{\partial x} = \frac{1}{2\Delta x}(\Psi_1 - \Psi_3)$$
$$\frac{\partial \Psi}{\partial y} = \frac{1}{2\Delta y}(\Psi_2 - \Psi_4)$$

Lösung der Poisson- (Helmholtz-) Gleichung

Ein wichtiger Bestandteil des barotropen quasi-geostrophischen Modells ist die Lösung einer Poisson- $(\nabla^2\Theta = G(x,y))$ bzw. Helmholtz- $((\nabla^2 + \lambda)\Theta = G(x,y))$ Gleichung, um zu jedem Zeitschritt Θ aus einem bekannten *G(x,y)* zu bestimmen. Eine Möglichkeit bietet hier das sog. 'Gauß-Seidel' Verfahren, bzw. seine als 'Successive Over-Relaxation' (SOR) bekannte Verbesserung, die hier an der Poisson-Gleichung erklärt werden sollen:

Der Ausgangspunkt ist diskrete Darstellung des Laplace-Operators (s. oben). setzt man diesen ein, so ergibt sich für die diskrete Poisson-Gleichung

$$\nabla^2\Theta_0 - G_0 = \frac{1}{\Delta x^2}(\Theta_1 + \Theta_3 - 2\Theta_0) + \frac{1}{\Delta y^2}(\Theta_2 + \Theta_4 - 2\Theta_0) - G_0 = 0$$

formal lässt sich hieraus für jeden Gitterpunkt 0 das gesuchte Θ_0 bestimmen. Dieses hängt jedoch von den andern (ebenfalls unbekannten) Θ an den Gitterpunkten 1,2,3 und 4 ab. Eine direkte Lösung ist also nicht möglich, eine approximative Lösung kann aber iterativ gefunden werden. Hierzu wird zunächst für einen beliebig vorgegebenes Anfangsfeld für Θ (z.B. überall 0 oder, vielleicht günstiger, das Θ zum vorhergehenden Zeitschritt) der Fehler $\varepsilon_0 = (\nabla^2\Theta_0 - G_0)$ nach obiger Formel bestimmt und jedes Θ_0 um diesen Fehler korrigiert. Es ergibt sich so ein neues Θ_0 (Θ'_0) zu

$$\Theta'_0 = \Theta_0 + \varepsilon_0 / (\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2})$$
$$= (\frac{1}{\Delta x^2}(\Theta_1 + \Theta_3) + \frac{1}{\Delta y^2}(\Theta_2 + \Theta_4) - G_0) / (\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2})$$

Ist mit Hilfe dieser Formel für jeden Gitterpunkt ein neues Θ bestimmt, beginnt das Verfahren von vorn (Bestimmung des Fehlers und Berechnung des neuen Θ), bis ein Maß des gesamten Fehlers (z.B. die Summe der Fehlerquadrate über alle Gitterpunkte) einen vorgegebenen Wert unterschreitet, bzw. bis eine vorher festgelegte Anzahl an Iterationen erreicht ist. (Bemerkung: wie zu ersehen ist, kann ein neues Θ auch ohne vorherige Berechnung des Fehlers bestimmt werden).

Dieses Verfahren ist jedoch noch nicht das 'Gauß-Seidel' Verfahren, sondern wird als Jacobi Methode bezeichnet. Die Jacobi Methode ist zumeist nicht praktikabel, da das Ergebnis viel zu langsam gegen die wirkliche Lösung konvergiert. Das 'Gauß-Seidel' Verfahren verbessert die Konvergenz erheblich. Der 'Trick' ist, dass zur Bestimmung von Θ'_0 nicht allein die 'alten' Θ verwendet werden, sondern, sobald verfügbar, die neuen Θ' eingehen. Wird z.B. das dargestellte Gebiet bei jedem Iterationsschritt von oben links zeilenweise nach unten rechts durchlaufen, so sind bei der Bestimmung von Θ'_0 bereits neue Θ' an den Gitterpunkten 2 und 4 vorhanden und das neue Θ'_0 ergibt sich zu

$$\Theta'_0 = \left(\frac{1}{\Delta x^2}(\Theta_1 + \Theta'_3) + \frac{1}{\Delta y^2}(\Theta_2 + \Theta'_4) - G_0\right) / \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)$$

Im Gauß-Seidel Verfahren sind typischerweise $N^2\ p/4$ Iterationen für ein Gitter der Größe $N \times N$ erforderlich um den Anfangsfehler um den Faktor 10^{-p} zu reduzieren (was in etwa eine Verbesserung um den Faktor 2 zur Jacobi Methode entspricht), was immer noch ein relativ großer Aufwand ist. Eine deutliche Reduzierung dieses Aufwands kann durch das 'Successive Over-Relaxation' (SOR) Verfahren erreicht werden. Hier wird so vorgegangen, dass das alte Θ nicht mit dem Fehler ε_0 sondern mit $\omega\,\varepsilon_0$ korrigiert, wobei $1 < \omega < 2$. Der Fehler wird also (wie der Name des Verfahrens schon sagt) 'überkorregiert'. Es folgt somit:

$$\Theta'_0 = \Theta_0 + \omega\varepsilon_0 \Big/ \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)$$

$$= \Theta_0 + \omega\left(\frac{1}{\Delta x^2}(\Theta_1 + \Theta'_3 - 2\Theta_0) + \frac{1}{\Delta y^2}(\Theta_2 + \Theta'_4 - 2\Theta_0) - G_0\right) / \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)$$

Auch hier werden wiederum alle bereits verfügbaren neuen Θ verwendet. Die Anzahl der benötigten Iterationen zur Reduzierung des Anfangsfehlers um den Faktor 10^{-p} verringert sich bei optimaler Wahl von ω theoretisch auf etwa $N\ p/3$, also etwa um den Faktor N im vergleich zum Gauß-Seidel Verfahren. Leider ist die Wahl des optimalen ω problematisch und für 'normale' meteorologische Probleme nur 'empirisch' möglich.

Die Lösung der Helmholtz Gleichung erfolgt analog wobei der diskrete Laplace-Operator um den Term $\lambda\Theta_0$ zu erweitern ist.

Natürlich gibt es auch andere Methoden zur Lösung der Poisson- (Helmholz-) Gleichung, wie z.B. Verfahren mit Hilfe der Fourierzerlegung oder die Mehrgitterverfahren.

Die zeitliche Ableitung

Zur Extrapolation in der Zeit, d.h. Berechnung des Wertes zum neuen Zeitpunkt, muss auch das Differential in der Zeit numerisch approximiert werden. wie bereits aus der Evolutions- und Advektions-Gleichung bekannt gibt es auch hier unterschiedliche Verfahren. Prinzipiell kann jedes numerisch stabile Verfahren verwendet werden. Durch die Komplexität (Nichtlinearität) der Gleichung sind implizite Verfahren zumeist aber nicht praktikabel. In der Meteorologie ist das Leapfrog Schema (ein Zweischrittverfahren, s. Evolutionsgleichung), zusammen mit dem Robert-Asselin Filter zur Verhinderung des künstlichen 'Computational-Modes', weit verbreitet und soll deshalb auch hier angewendet werden:

$$\Psi(t+\Delta t) = \Psi^*(t-\Delta t) + 2\Delta t\left(\frac{\partial \Psi}{\partial t}\right)$$

$$\Psi^*(t) = \Psi(t) + F\left[\Psi(t+\Delta t) + \Psi^*(t-\Delta t) - 2\Psi(t)\right]$$

Mit der Filterkonstanten *F* (typischerweise 0.1). Da beim Leapfrog zwei Zeitebenen zur Berechnung der nächsten benötigt werden, muss am Anfang ein anderes Verfahren (z.B. explizit Euler) eingesetzt werden.

3. Umsetzung in die Praxis

Die praktische Umsetzung des oben vorgestellten barotropen Modells erfordert relativ umfangreiche Programmierarbeit, bei der naturgemäß viele Fehler begangen werden können (gedankliche wie technische). Es bietet sich an, das Modell nicht in einem Stück zu schreiben, sondern in kleinen Abschnitten, die dann separat und in Zusammenarbeit getestet werden können (modulare Vorgehensweise). So werden nach und nach Subroutinen einzelner Komponenten des Modells erstellt und im Hauptprogramm zusammengeführt. Diese Komponenten können sein:

- Eingabe und Ausgabe
- Erstellung des Gitters
- Bildung von räumlichen Ableitungen
- Bildung des Laplace-Operators
- Bildung des Jacobi-Operators
- Zeitliche Extrapolation
- Lösung der Poisson- (Helmholtz-) Gleichung
- Berücksichtigung der Randbedingungen
- usw.

Die Erstellung des Hauptprogramms kann am Beginn der Arbeiten stehen. Hier gilt es den späteren Ablauf im Modell (wie oben beschrieben) in eine Programmstruktur um zu setzten, auch dies kann durch den Einsatz von Subroutinen für einzelne Programmblöcke erleichtert werden. Ferner muss überlegt (und umgesetzt) werden, welche (globalen) Variablen (Skalare und Felder) benötigt werden und welche Namen (möglichst selbsterklärend) diese erhalten sollen. Diese Variablen werden am besten in einem 'modul' definiert und vorbelegt. Es kann nicht schaden, schon Aufrufe der späteren Subroutinen für die einzelnen Komponenten an die entsprechenden Stellen einzufügen, die in dieser Phase noch Platzhalter Routinen (die nichts tun) aufrufen. Diese Platzhalter können dann später nach und nach durch die wirklichen Komponenten ersetzt werden. Sobald ein solches 'fertiges' Hauptprogramm erstellt wurde, sollte es getestet (kompiliert) werden.

Als zweiter Schritt bietet sich die Ein- und Ausgabe an. Parameter, die das Modell später von außen steuern müssen eingelesen werden. Möglicherweise sollen auch Startdaten des vorherzusagenden Feldes (z.B. der Stromfunktion) eingelesen werden. Herausgeschrieben werden natürlich die Ergebnisse (welche, wann, etc.?), aber auch die Ausgabe von Kontrollvariablen ist sicher sinnvoll. Die Wahl des Formats des Outputs ist eine wichtige Entscheidung.

Nach diesen mehr technischen Vorbereitungen kann mit der eigentlichen 'Physik' des Modells begonnen werden. Zunächst wird das Modell initialisiert. D.h. Parameter, die noch nicht belegt sind, bzw. die von der Eingabe abhängen und/oder berechnet werden müssen, werden gesetzt. So wird z.B. hier auch das Gitter auf dem gerechnet wird initialisiert, d.h. seine

numerische Dimension (meist schon als Parameter im 'modul' festgelegt) in physikalische Größen (Länge, Breite, Gitterpunktsabstände, etc.) umgesetzt. Natürlich müssen auch die Startfelder belegt werden (soweit sie nicht von außen eingelesen werden).

Es folgt dann der Einbau aller anderen Komponenten (Subroutinen), die noch benötigt werden. Dabei sollte auf jeden Fall ständig getestet werden. Teilweise (bei komplizierten Komponente, wie dem Lösen der Poisson-Gleichung) kann es sinnvoll sein die Komponente unabhängig vom Rest des Modells zu testen.

Wenn alle Komponenten zusammen und getestet sind, sollte vor den eigentlichen Rechnungen noch ein Test der 'physikalischen' Richtigkeit des Modells erfolgen. D.h. die numerische Lösungen bestimmter Probleme sollten mit bekannten (im Idealfall analytischen, ansonsten mit Arbeiten Anderer) Lösungen verglichen werden. Für das barotrope Modell bietet sich die Rossby-Haurwitz Welle (s. Übungen 'Theoretische Meteorologie') als ein Testfall an, da sie ja eine analytische Lösung der nichtlinearen Gleichung ist. Stimmen die Lösungen des Modells mit der bekannten Lösungen überein, so kann 'guten Gewissens' mit den wirklichen Experimenten begonnen werden.

4. Der Code

Ausgehend von dem oben Geschriebenen und der Vorlesung soll ein barotropes Modell programmiert werden, das die divergenzfreie Vorticity-Gleichung auf der Beta-Ebene mithilfe der Gitterpunktsmethode in einem Kanal löst. Als Ausgangspunkt kann das vorgefertigte Programm 'baro.f90' verwendet werden, das bereits einige Komponenten des kompletten Modells enthält (wird in den Übungen/der Vorlesung besprochen).

Um ein vollständiges System zu erhalten, müssen hier noch folgende Subroutinen geschrieben und in den Code integriert werden, die zur Zeit nur als 'Rumpf' vorhanden sind:

- ***sor***: Lösung einer Poisson-Gleichung (inverser Laplace) mithilfe der 'Successive Over-Relaxation' (SOR)
- ***mkdfdx***: Berechnung der Ableitungen in x-Richtung
- ***laplace***: Anwendung des Laplace Operators auf ein Feld
- ***jacobi***: Anwendung des Jacobi Operators auf zwei Felder
- ***euler***: Durchführung eines expliziten (Euler) Zeitschritts
- ***leapfrog***: Durchführung eines Leapfrog Zeitschritts (mit Filter)

sor

Die Lösung der Poisson-Gleichung mit dem SOR Verfahren ist der wohl aufwändigste und zentralste Teil des Modells. Von den Modellparametern und Variablen werden hier die Dimensionen der Felder, die Gitterpunktsabstände in x- und y-Richtung sowie ein Input-Feld (im späteren Modell die Vorticity-Tendenzen) benötigt. Als Output wird der inverse Laplace des Inputs an das Modell zurückgegeben (d.h. die Stromfunktionstendenzen). Im der vorhandenen 'dummy' Routine wird dem Rechnung getragen:

```
      subroutine sor(pdf,pf,pdx,pdy,kx,ky)
      implicit none
!
!     subroutine sor compute the inverse Laplacian from a given field
!     by using the Successive OverRelaxation method (SOR)
!
      integer :: kx                ! x dimension
      integer :: ky                ! y dimension
      real :: pdx                  ! x grid point distance
      real :: pdy                  ! y grid point distance
```

```
      real :: pdf(0:kx+1,0:ky+1)   ! input: field
      real :: pf(0:kx+1,0:ky+1)    ! output: inverse Laplacian of input
!
      return
      end
```

Beachte, dass hier die Felder inklusive der Ränder dimensioniert sind.

Um aus diesem Rumpf die gewünschte Subroutine zu entwickeln, kann dieser Code nun um folgende Teile ergänzt werden:

1. berechne einen Anfangsfehler ($\varepsilon_0=(\nabla^2\Theta_0 - G_0)$, s. Abschnitt SOR oben)
2. (über-) korrigiere die Anfangslösung
3. berechne einen neuen Fehler
4. wiederhole 2 und 3 bis ein globales Maß des Fehlers 'klein genug' ist (bzw. um den Faktor x kleiner als der Anfangsfehler)

Es ist günstig hierbei u. a. Folgendes zu beachten: a) Setzte nach jeder Iteration die Randbedingungen neu (z.B. durch Aufruf der entsprechenden Subroutine). b) Die Genauigkeit des Computers ist nicht beliebig hoch, der Fehler kann also nicht beliebig klein werden (starte zunächst mit einer kleinen Verbesserung, z.B. Faktor 10). c) Setzte ein Maximum der Iterationen die durchgeführt werden sollen, um nicht eine 'Todschleife' zu produzieren (Theoretisch sollten bei NxN Gitterpunkten (N p/3) Iterationen nötig sein, um den Fehler um den Faktor 10^p zu reduzieren).

mkdfdx

Zur Berechnung der x-Ableitung mithilfe zentrierter Differenzen werden folgende Modellparameter und Variablen benötigt: Die Dimensionen der Felder, der Gitterpunktsabstand in x-Richtung sowie ein Input-Feld. Als Output wird die x-Ableitung des Inputs an das Modell zurückgegeben. Dies sieht die dummy-Routine folgendermaßen vor:

```
      subroutine mkdfdx(pf,pdfdx,pdx,kx,ky)
      implicit none
!
!     subroutine mkdfdx computes x derivation from a field
!     using central differences
!
      integer :: kx                 ! x-dimension
      integer :: ky                 ! y-dimension
      real :: pdx                   ! x grid distance
      real :: pf(0:kx+1,0:ky+1)     ! input: field
      real :: pdfdx(0:kx+1,0:ky+1)  ! output: dfield/dx
!
      return
      end
```

Die Berechnung zentrierter Differenzen wurde bereits im letzten Semester (Advektionsgleichung) geübt und sollte hier kein Problem sein.

laplace

Auch der Laplace Operator soll durch zentrierte Differenzen dargestellt werden (s. Text oben). Die Dimensionen der Felder, die Gitterpunktsabstände in x-und y-Richtung und das Inputfeld werden an die Subroutine übergeben, das Ergebnis wird zurückgeliefert:

```
      subroutine laplace(pf,pdf,pdx,pdy,kx,ky)
      implicit none
```

```
!
!     subroutine laplace computes the laplacian from a field
!
      integer :: kx                ! x-dimension
      integer :: ky                ! y-dimension
      real :: pdx                  ! x grid distance
      real :: pdy                  ! y grid distance
      real :: pf(0:kx+1,0:ky+1)    ! input: field
      real :: pdf(0:kx+1,0:ky+1)   ! output: Laplacian
!
      return
      end
```

Die Berechnung des Laplace Operators kann nun leicht implementiert werden.

jacobi

Die Programmierung des erhaltenden Jacobi-Operators nach Arakawa (s. oben) erfordert etwas mehr Programmieraufwand und besondere Sorgfalt bei den Indizes. Übergabeparameter an/von der Subroutine sind hier die Dimensionen der Felder, die Gitterpunktsabstände in x- und y-Richtung, die beiden Input-Felder und (als output) das Ergebnis:

```
      subroutine jacobi(p1,p2,pj,pdx,pdy,kx,ky)
      implicit none
!
!     subroutine jacobi computes the jacobi operator according to
!     Arakawa (1966)
!
!     jacobi(1,2)=(j1+j2+j3)/3. after Arakawa 1966
!
      integer :: kx              ! x-dimension
      integer :: ky              ! y-dimension
      real :: p1(0:kx+1,0:ky+1)  ! input: field 1
      real :: p2(0:kx+1,0:ky+1)  ! input: field 2
      real :: pj(0:kx+1,0:ky+1)  ! output: jacobi(1,2)
      real :: pdx                ! x grid distance
      real :: pdy                ! y grid distance
!
      return
      end
```

Bei der Implementierung des Jacobi-Operators ist es günstig Hilfsfelder (für die drei unterschiedlichen Darstellungsweisen) zu verwenden, die dann natürlich entsprechend deklariert werden müssen.

euler

Die Durchführung eines (expliziten) Euler Zeitschritts ist leicht zu programmieren. Von den Modellparametern und Variablen werden hier die Dimensionen der Felder, der Zeitschritt, das Feld der Tendenzen und das aktuelle Feld (Zeitpunkt t) benötigt. Als Output wird das Feld zum Zeitpunkt (t+1) an das Modell zurückgegeben. Im der vorhandenen 'dummy' Routine wird dem Rechnung getragen:

```
      subroutine euler(pfm,pdfdt,pf,pdelt,kx,ky)
      implicit none
!
!     subroutine euler does an explicit Euler time step
!
      integer :: kx                  ! x dimension
      integer :: ky                  ! y dimension
      real    :: pdelt               ! time step [s]
      real    :: pfm(0:kx+1,0:ky+1)  ! input f(t)
      real    :: pdfdt(0:kx+1,0:ky+1) ! input tendency
      real    :: pf(0:kx+1,0:ky+1)   ! output f(t+1)
!
```

```
      return
      end
```

Um diese dummy Routine zu komplettieren ist eigentlich nur noch eine Zeile nötig.

leapfrog

Im Vergleich zum Euler ist das (gefilterte) Leapfrog Verfahren ein wenig schwieriger zu implementieren (auch wenn es bereits im letzten Semester anhand der Advektionsgleichung geübt wurde). Von den Modellparametern und Variablen werden hier die Dimensionen der Felder, der zweifache Zeitschritt, das Feld der Tendenzen, das aktuelle Feld (Zeitpunkt t) und das (gefilterte) Feld zum Zeitpunkt (t-1) benötigt. Als Output werden das neue Feld zum Zeitpunkt t und das neue gefilterte Feld zum Zeitpunkt t-1 an das Modell zurückgegeben. Im der vorhandenen 'dummy' Routine ist vorgesehen, dass die alten Felder mit den neuen überschrieben werden:

```
      subroutine leapfrog(pf,pfm,pdfdt,pdelt2,kx,ky)
      implicit none
!
!     subroutine leapfrog does a leapfrog time step
!     with Robert Asselin filter
!
      integer :: kx                ! x dimension
      integer :: ky                ! y dimension
      real :: pdelt2               ! 2.* timestep
      real :: pf(0:kx+1,0:ky+1)    ! input/output f(t)
      real :: pfm(0:kx+1,0:ky+1)   ! input/output f(t-1) (filtered)
      real :: pdfdt(0:kx+1,0:ky+1) ! input tendency
!
      return
      end
```

Das leapfrog Verfahren kann hier nun relativ leicht nach den oben im Text stehenden (und in der Vorlesung besprochenen) Formeln implementiert werden. Beachte die am Ende zu erfolgende Umspeicherung (t) -> (t-1) und (t+1) -> (t).

5. Die Aufgaben

5.1. *Erstellung des Modells:* Erstelle mit Hilfe des oben Geschrieben, des in der Vorlesung gelernten und (u.U.) der Beispiel-Programme auf der 'homepage' das barotrope divergenzfreie Modell und teste es anhand der Rossby-Haurwitz Welle.

Im code 'baro.f90' sind für diesen Test bereits (fast) alle Einstellungen vorhanden: Im Module *baromod* sind die Dimensionen *NX* und *NY* des Kanals auf 64 bzw. 32 gesetzt, die Länge (*xchannel*) und Breite (*ychannel*) auf 360° und 40°, die zentrale Breite (*rlat*) auf 50°N und der Zeitschritt (*delt*) auf 1200s. In subroutine *initial* wird die Anfangs-Stromfunktion mit einer Welle der Wellenzahl 1 in x-Richtung (0.5 in y-Richtung) belegt. Es muss nur noch die Anzahl der zu rechnenden Zeitschritte (*nrun*) festgelegt werden. Am Anfang sollte *nrun* auf 1 belassen werden. Sollte hier das Ergebnis richtig aussehen, kann ein entgültiger Test mit *nrun*=100 erfolgen.

5.2. *Versuche zur barotropen Instabilität:* Es kann gezeigt werden (z.B. Holton) dass unter gewissen Umständen eine zonal symmetrische Strömung barotrop instabil werden kann, d.h. dass (anfangs kleine) Störungen dem Grundstrom kinetische Energie entziehen und dadurch anwachsen. Desgleichen ergeben sich auch Situationen, in denen Störungen ihre kinetische Energie an den Grundstrom abgeben (barotrope Stabilität; auf der globalen Skala ist dieser Prozess dominierend).

Eine Bedingung für barotrope Instabilität eines zonalen Jets ist das Vorhandensein einer Nullstelle der absoluten Vorticity (allgemeiner, der potentiellen Vorticity) d.h.

$$\beta - \frac{\partial^2 [U]}{\partial y^2} = 0$$

irgendwo (bei einer kritischen Breite y_c). Hierbei ist [U] der zonal gemittelte Zonalwind (Jet).

Um barotrope Instabilität/Stabilität zu untersuchen, können unterschiedliche idealisierte Zonalwindjets analytisch oder mithilfe numerischer Modelle studiert werden. Während (relativ leicht) gezeigt werden kann, dass die Annahme eines einfachen Cosinus-Profils ([U]=$cos(\pi y/B)$; mit B=Breite des zu untersuchenden Kanals) nur stabile Strömungen liefert, kann das Profil

$$[U] = \frac{U_{\max}}{2}(1 - \cos(\frac{2\pi}{B} y))$$

barotrop instabil sein (U_{max} ist hierbei die Windgeschwindigkeit im Zentrum des Jets). Untersuchungen zeigen, dass je nach Stärke des Jets und Wellenzahl der Störung sowohl Stabilität als auch Neutralität und Instabilität auftreten kann. Wobei gilt: Ab einer Mindestgeschwindigkeit des Jets sind Wellen bis zu einer bestimmten Wellenlänge (L_c) stabil (geben ihre Energie an den Grundstrom ab), Wellen länger als L_c und kürzer als eine zweite kritische Wellenlänge (L_0) sind instabil (gewinnen Energie aus dem Grundstrom) und sehr lange Wellen ($L > L_0$) sind neutral (ihre Energie bleibt konstant). Für diese drei Klassen gilt folgendes:

1. Stabil: Phasengeschwindigkeit der Welle ist größer als die Jet-Geschwindigkeit bei der kritischen Breite
2. Instabil: Phasengeschwindigkeit der Welle ist positiv aber kleiner als die Jet-Geschwindigkeit bei der kritischen Breite
3. Neutral: Phasengeschwindigkeit der Welle ist negativ.

Wobei sich die Jet-Geschwindigkeit bei der kritischen Breite, und damit die Phasengeschwindigkeit der Welle, die Bereich 1 und Bereich 2 trennt (*Cn*), aus der Instabilitätsbedingung und dem Jetprofil ermitteln lässt:

$$[U](y_c) = Cn = \frac{U_{\max}}{2} - \frac{\beta B^2}{4\pi^2}$$

Untersuche/Verifiziere mithilfe des barotropen Modells die Stabilitätseigenschaften des o.a. Jets. Variiere hierzu bei festgehaltenen U_{max} die Wellenzahl der aufgeprägten Störung und betrachte das zeitliche Verhalten der kinetischen Energie der Störung und des Grundstroms. bei welchen Werten liegen die kritischen zonalen Wellenlängen? Wie entwickelt sich die horizontale Struktur (Phasenlage) der instabilen und der stabilen Welle mit der Zeit? Hierbei sollen folgende Parameter gewählt werden:

- Kanal wie im Original-Modell.
- U_{max}=100m/s
- Meridionale Wellenzahl der Störung=0.5 (wie im Original)

- 1 Tag Integrationszeit bei 20 Minuten Zeitschritt

Anmerkungen zur Durchführung: a) Um das vorgegeben Jet-Profil in das Modell einzubauen, müssen die Randbedingungen geändert werden (subroutine boundary). Bisher wurde dort die Stromfunktion an den meridionalen Rändern auf Null gesetzt (was keinen Jet zulässt). Als neue Randbedingung (in y) kann das zonale Mittel der Stromfunktion an den Punkten y=1 (für 0) und y=NY (für NY+1) genommen werden.
b) In subroutine initial muss die Anfangsstromfunktion um den Jet erweitert werden. Beachte: In Stromfunktion schreibt sich das Profil:

$$[\Psi] = -\frac{U_{\max}}{2}(y - \frac{B}{2\pi}\sin(\frac{2\pi}{B}y))$$

c) In subroutine initial kann auch die x-Wellenzahl der Störung eingestellt werden (parameter zk)

d) Es ist möglicherweise günstig, die Energiediagnostik direkt in das Modell einzubauen (z.B. in subroutine baroout). Aufteilen in Grundstrom + Störung nicht vergessen.

e) Durch numerisches Rauschen (Ungenauigkeiten des Verfahrens und des Rechners) sollte nicht unbedingt erwartet werden, dass neutrale Wellen wirklich neutral sind.

5.3. *Eine Wettervorhersage:* Das divergenzfreie barotrope Modell war das erste funktionierende numerische 'Wettervorhersagemodell. Durch die große Anzahl an Vereinfachungen, die in das Modell einfließen, kann es natürlich nicht mit den heutigen Modellen der Wetterdienste konkurrieren. Trotzdem soll zu Testzwecken eine Vorhersage mit dem in der Übung programmierten Modell gewagt werden. Dazu stehen auf dem file input.dat auf der 'homepage', als Anfangs und Verifikationsdatensatz, 6-stündliche Geopotentialdaten auf dem Modellgitter (64x34; d.h. gesamte Kanalbreite incl. Rand) für den Zeitraum 14.02.1962 00:00 bis 24.02.1962 00:00 (große Hamburger Sturmflut 15.-16.02.1962) bereit. zusätzlich ist mit dem file input.ctl ein grads-control file zum plotten dieser Daten gegeben.

Anmerkungen zur Durchführung: a) Zum Einlesen kann die subroutine initial verwendet werden. Bei den Daten handelt es sich um einen unformatierten file, der mit den Befehlen

```
open(11,file='input.dat',form='unformatted')
read(11) f(1:NX,0:NY+1)
close(11)
```

gelesen werden kann (ein mehrfaches read überspringt Zeitpunkte und so können andere Anfangswerte gewählt werden).

b) Da es sich um Geopotential handelt, müssen die Werte durch f0 geteilt werden, um eine Stromfunktion zu erhalten.

c) Wichtig sind hier die Randbedingungen. Um das Modell stabil zu halten, sollte die Stromfunktion am Nord- und Südrand (0 und NY+1) auf den Anfangswert festgehalten werden. Dazu reicht es in der subroutine boundary das Setzten der Randbedingungen für diese Punkte zu entfernen.